

IBM Content
Manager for Multiplatforms/IBM Information
Integrator for Content



Manual de Programação de Aplicação de Estações de Trabalho

Versão 8 Edição 2

IBM Content
Manager for Multiplatforms/IBM Information
Integrator for Content



Manual de Programação de Aplicação de Estações de Trabalho

Versão 8 Edição 2

Nota

Antes de utilizar estas informações e o produto a que se referem, leia as informações incluídas em “Informações” na página 513.

Segunda Edição (Março 2003)

Esta edição aplica-se à Versão 8 Edição 2 de IBM Enterprise Information Portal for Multiplatforms, Content Manager para Multiplataformas da IBM (números de produto 5724-B43, 5724-B19) e a todas as edições e modificações subsequentes até indicação em contrário em novas edições.

Partes dest produto são: Copyright © 1990-2000 ActionPoint, Inc. e/ou os seus licenciadores, 1299 Parkmoor Drive, San Jose, CA 95126 U.S.A. Todos os direitos reservados.

Aviso de Copyright, Licença e Renunciador do CUP Parser Generator

© Copyright 1996-1999 por Scott Hudson, Frank Flannery, C. Scott Ananian

Permissão para utilizar, copiar, modificar e distribuir o software e a documentação do CUP Parser Generator para qualquer propósito e sem qualquer pagamento é por este meio concedida, desde que o aviso de copyright acima mencionado surja em todas as cópias e que tanto o aviso de copyright como o aviso de permissão e o renunciador de garantia surjam na documentação de suporte e que os nomes dos autores e dos seus colaboradores não sejam utilizados para efeitos de publicidade que diga respeito à distribuição do software sem uma permissão prévia e específica por escrito.

Os autores e seus empregadores renunciam todas as garantias respeitantes a este software, incluindo todas as garantias de padrão de qualidade do produto e de boas condições. Em circunstância alguma serão os autores e seus empregadores responsáveis por danos específicos ou indirectos ou quaisquer tipos de danos que resultem da perda de utilização, dados ou lucros, quer seja numa acção de contrato, negligência ou outras acções lesivas, consequentes ou relacionados com a utilização ou desempenho deste software.

© Copyright International Business Machines Corporation 1996, 2003. Todos os direitos reservados.

Índice

| | |
|---|------------|
| Acerca deste manual | vii |
| Quem deve utilizar este manual | vii |
| Onde encontrar mais informações | vii |
| As informações incluídas na embalagem do produto | vii |
| Suporte disponível na Web | viii |
| Como enviar comentários | ix |
| Novidades no Enterprise Information Portal Versão 8 | ix |
| Novidades no Content Manager Versão 8 | xi |

Introdução ao Enterprise Information

| | |
|---|----------|
| Portal | 1 |
| Pesquisar informações de clientes | 1 |
| A necessidade | 2 |
| A solução | 2 |
| A solução de Enterprise Information Portal | 2 |
| Componentes do IBM Enterprise Information Portal for Multiplatforms | 4 |
| Novidades nas APIs da Versão 8.2 | 6 |
| Novidades nas APIs da Versão 8.1 | 7 |
| Classes de Java novas e alteradas | 8 |
| Classes novas e alteradas de C++ | 9 |

Conceitos de programação de aplicações do Enterprise Information Portal

| | |
|--|-----------|
| Portal | 11 |
| Compreender acesso a dados através de servidores de conteúdos | 11 |
| Compreender conceitos de objectos de dados dinâmicos | 12 |
| Objectos de dados dinâmicos (DDO) | 12 |
| Objectos de dados expandidos (XDO) | 13 |
| Representar conteúdo multimédia | 13 |
| Compreender servidores de conteúdo e DDOs | 14 |
| Comparar DDO/XDOs com valores de atributos e partes de artigos | 14 |
| Compreender identificadores permanentes (PID) | 14 |

Trabalhar com um servidor de conteúdos associado e pesquisa associada

| | |
|--|-----------|
| associada | 15 |
| Definição de correspondências do esquema associado | 17 |
| Utilizar componentes de definição de correspondências do servidor de conteúdos associado | 18 |
| Executar consultas associadas | 18 |
| Sintaxe de consultas associadas | 19 |
| Armazenar resultados da consulta em pastas associadas (Apenas para Java) | 22 |
| Trabalhar com a administração do sistema | 22 |
| Personalizar o cliente de administração do sistema de EIP | 23 |

Programar com as interfaces de programação de aplicações (APIs)

| | |
|--|-----------|
| programação de aplicações (APIs) | 25 |
| Compreender as diferenças entre as APIs de Java e C++ | 25 |
| Compreender a arquitectura cliente/servidor (Apenas para Java) | 25 |
| Criação de pacotes para o ambiente Java | 26 |
| Sugestões de programação | 26 |
| Configurar o ambiente de Java (Apenas para Java) | 26 |
| Definir as variáveis de ambiente de Java para Windows | 27 |
| Definir as variáveis de ambiente de Java para AIX | 27 |
| Definir as variáveis de ambiente de Java para Solaris | 28 |
| Utilizar Remote Method Invocation (RMI) com servidores de conteúdos | 28 |
| Configurar o ambiente de C++ (Apenas para C++) | 28 |
| Definir as variáveis de ambiente de C++ em Windows | 30 |
| Definir as variáveis de ambiente de C++ em AIX | 30 |
| Construir programas C++ | 31 |
| Definir o subsistema da consola para conversão de página de códigos em Windows | 32 |
| Compreender várias opções de pesquisa | 32 |
| Rastreio | 33 |
| Rastrear consultas de texto utilizando o Motor de Pesquisa de Texto | 33 |
| Rastrear consultas paramétricas | 34 |
| Tratar excepções | 34 |
| Constantes | 35 |
| Ligação a servidores de conteúdos | 36 |
| Estabelecer uma ligação | 36 |
| Ligar e desligar de um servidor de conteúdos num cliente | 37 |
| Definir e obter opções do servidor de conteúdos | 37 |
| Enumerar servidores de conteúdos | 38 |
| Enumerar as entidades e os atributos de um servidor de conteúdos | 39 |
| Trabalhar com objectos de dados dinâmicos (DDOs) | 41 |
| Criar um DKDDO | 42 |
| Adicionar propriedades a um DDO | 43 |
| Criar um identificador permanente (PID) | 44 |
| Trabalhar com artigos e propriedades de dados | 44 |
| Obter o DKDDO e as propriedades de atributo | 46 |
| Apresentar todo o DDO | 48 |
| Eliminar um DDO (Apenas para C++) | 49 |
| Trabalhar com objectos de dados expandidos (XDOs) | 50 |
| Utilizar um identificador permanente de XDO (PID) | 50 |
| Compreender as propriedades do XDO | 51 |
| Cadeias de configuração de DB2, ODBC e DataJoiner (Apenas para C++) | 51 |
| Sugestões de programação de Java | 52 |

| | | | |
|--|------------|---|-----|
| Sugestões de programação de C++ | 52 | Trabalhar com exemplos de Content Manager | 141 |
| Programar um XDO como parte de DDO | 52 | O exemplo de hipótese de seguro | 142 |
| Programar um XDO autónomo | 54 | Criar uma aplicação do Content Manager | 142 |
| Exemplos de trabalho com um XDO | 58 | Compreender os componentes do software | 142 |
| Trabalhar com XML (Apenas para Java) | 80 | Representar artigos através de DDOs | 143 |
| Expandir a capacidade de importação e | | Ligação ao sistema do Content Manager | 144 |
| exportação de XML. | 80 | Trabalhar com artigos | 145 |
| Importar documentos XML | 81 | Trabalhar com pastas | 167 |
| Exportar XML | 85 | Definir ligações entre artigos | 173 |
| Criar documentos e utilizar o atributo DKPARTS | 86 | Trabalhar com o controlo de acesso | 175 |
| Criar pastas e utilizar o atributo DKFOLDER | 89 | Criar um privilégio | 176 |
| Utilizar DKAny (Apenas para C++) | 92 | Criar um conjunto de privilégios | 177 |
| Utilizar código de tipo. | 92 | Visualizar propriedades do conjunto de | |
| Gerir memória em DKAny | 93 | privilégios | 179 |
| Utilizar construtores | 93 | Definir uma lista de controlo de acesso (ACL) | 180 |
| Obter o código de tipo | 93 | Obter e visualizar informações sobre a ACL | 182 |
| Atribuir um novo valor a DKAny | 93 | Atribuir uma ACL a um tipo de artigo | 183 |
| Atribuir um valor de DKAny | 94 | Atribuir uma ACL a um artigo | 184 |
| Apresentar DKAny | 94 | Compreender a linguagem da consulta | 185 |
| Destruir DKAny | 95 | Consultar o servidor de Content Manager | 186 |
| Sugestões de programação | 95 | Aplicar a linguagem da consulta ao modelo de | |
| Utilizar conjuntos e iteradores | 95 | dados do Content Manager. | 187 |
| Utilizar métodos de recolha sequencial | 96 | Compreender a pesquisa paramétrica | 188 |
| Utilizar o iterador sequencial | 96 | Compreender a pesquisa de texto | 189 |
| Gerir memória em recolhas (Apenas para C++) | 98 | Pesquisar conteúdos de objectos | 189 |
| Ordenar a recolha | 99 | Pesquisar documentos | 190 |
| Compreender a recolha e o iterador associados | 100 | Tornar os atributos definidos pelo utilizador | |
| Consultar um servidor de conteúdos | 101 | pesquisáveis por texto | 190 |
| Diferenças entre dkResultSetCursor e DKResults | 102 | Compreender a sintaxe de pesquisa de texto | 190 |
| Utilizar consultas paramétricas | 102 | Criar pesquisa de texto e paramétrica combinada | 191 |
| Utilizar a consulta de texto | 109 | Exemplos de consultas | 194 |
| Utilizar o cursor do conjunto de resultados | 121 | Exemplos de consultas | 197 |
| Abrir e fechar o cursor do conjunto de | | Compreender a linguagem da consulta | 203 |
| resultados para voltar a executar a consulta | 122 | Utilizar sequências de abandono com | |
| Definir e obter posições num cursor do conjunto | | operadores de comparação ("=", "!=", ">", "<", | |
| de resultados | 122 | "BETWEEN" e outros) | 203 |
| Criar uma recolha a partir de um cursor do | | Utilizar sequências de abandono com o | |
| conjunto de resultados | 124 | operador LIKE | 204 |
| Consultar recolhas. | 125 | Utilizar sequências de abandono com pesquisa | |
| Obter o resultado de uma consulta | 125 | de texto avançada (funções "contains" e "score"). | 205 |
| Avaliar uma nova consulta | 126 | Utilizar sequências de abandono com pesquisa | |
| Utilizar a recolha passível de consulta em | | de texto básica (funções "contains-text-basic" e | |
| alternativa à consulta combinada | 127 | "score-basic") | 206 |
| | | Utilizar sequências de caracteres globais em | |
| | | Java e C++ | 207 |
| | | A gramática da linguagem da consulta | 208 |
| | | Trabalhar com o gestor de recursos | 210 |
| | | Trabalhar com objectos do gestor de recursos | 211 |
| | | Gerir documentos em Content Manager | 211 |
| | | Criar o modelo de dados de gestão de documentos | 213 |
| | | Criar um tipo de artigo de documento | 213 |
| | | Criar um documento | 215 |
| | | Actualizar um documento | 217 |
| | | Obter e eliminar um documento | 219 |
| | | Elaborar versões de partes no modelo de dados | |
| | | de gestão de documentos | 219 |
| | | Trabalhar com transacções | 220 |
| | | Pontos a ter em consideração ao conceber | |
| | | transacções na aplicação do utilizador | 221 |
| | | Precauções ao utilizar transacções explícitas | 222 |
| Trabalhar com o Content Manager | | | |
| Versão 8.2. | 129 | | |
| Compreender o sistema Content Manager | 129 | | |
| Compreender os conceitos do Content Manager | 130 | | |
| Artigos | 130 | | |
| Atributos. | 131 | | |
| Tipos de artigos | 131 | | |
| Componentes de raiz e descendentes | 132 | | |
| Objectos | 133 | | |
| Ligações e Referências | 133 | | |
| Documentos. | 134 | | |
| Pastas | 134 | | |
| Elaboração de versões | 134 | | |
| Controlo de acesso | 135 | | |
| Planear uma aplicação do Content Manager | 140 | | |
| Determinar as funções da sua aplicação | 140 | | |
| Tratamento de erros | 140 | | |

| | |
|--|-----|
| Utilizar operações de dar entrada e saída em transacções | 222 |
| Processamento de transacções | 223 |
| Encaminhar um documento através de um processo | 224 |
| Compreender o processo do encaminhamento de documentos | 224 |
| Configurar um processo de encaminhamento de documentos | 225 |

Trabalhar com outros servidores de conteúdos. 245

| | |
|--|-----|
| Trabalhar com o Content Manager anterior | 247 |
| Manusear objectos grandes | 247 |
| Utilizar DDOs para representar o conteúdo do Content Manager anterior | 248 |
| Criar, actualizar e eliminar documentos ou pastas | 249 |
| Obter um documento ou pasta | 257 |
| Compreender pesquisa de texto (Motor de Pesquisa de Texto). | 260 |
| Pesquisar imagens pelo conteúdo. | 283 |
| Utilizar aplicações de pesquisa de imagens | 286 |
| Estabelecer uma ligação em QBIC | 291 |
| Enumerar servidores de pesquisa de imagens | 292 |
| Enumerar bases de dados, catálogos e funções de pesquisa de imagens | 293 |
| Representar informações de pesquisa de imagens com um DDO | 296 |
| Trabalhar com consultas de imagens. | 297 |
| Utilizar o motor de pesquisa de imagens | 300 |
| Indexar um XDO existente através de motores de pesquisa | 301 |
| Utilizar consulta combinada | 303 |
| Compreender as funções de fluxo de trabalho e de cesto de trabalho do Content Manager anterior | 307 |
| Trabalhar com OnDemand | 315 |
| Representar servidores e documentos OnDemand | 315 |
| Estabelecer e anular ligações ao servidor de OnDemand | 316 |
| Enumerar informações em OnDemand | 316 |
| Obter um documento OnDemand | 319 |
| Activar o modo de pastas OnDemand | 326 |
| Pesquisa assíncrona | 326 |
| Pastas OnDemand como modelos de pesquisa | 327 |
| Pastas OnDemand como entidades nativas | 327 |
| Criar e modificar anotações. | 327 |
| Rastreio | 327 |
| Trabalhar com Content Manager ImagePlus for OS/390 | 329 |
| Enumerar entidades e atributos | 330 |
| Sintaxe da consulta de ImagePlus para OS/390 | 334 |
| Trabalhar com o Content Manager for AS/400 | 336 |
| Enumerar entidades (classes de índices) e atributos | 336 |
| Executar uma consulta | 338 |
| Executar uma consulta paramétrica | 343 |
| Trabalhar com Domino.Doc. | 344 |
| Enumerar entidades e sub-entidades | 346 |

| | |
|--|-----|
| Enumerar atributos de armário | 348 |
| Construir consultas em Domino.Doc. | 349 |
| Utilizar sintaxe de consulta. | 349 |
| Trabalhar com Extended Search (ES). | 350 |
| Enumerar servidores de Extended Search | 351 |
| Enumerar entidades (bases de dados) e atributos (campos). | 351 |
| Utilizar a Generalized Query Language (GQL) | 355 |
| Identificar o artigo de tipo de DDO em Extended Search | 356 |
| Criar PIDs na Extended Search | 357 |
| Processar os conteúdos de um documento de Extended Search | 357 |
| Obter um documento. | 362 |
| Obter um objecto binário grande (BLOB) | 362 |
| Associar tipos de MIME com documentos. | 363 |
| Utilizar pesquisa associada em Extended Search | 364 |
| Trabalhar com Panagon Image Services (Apenas para Java) | 364 |
| Modelar dados | 364 |
| Documentos e páginas em Panagon Image Services | 365 |
| Enumerar entidades e atributos | 366 |
| Consultar. | 368 |
| Trabalhar com bases de dados relacionais | 370 |
| Ligação a bases de dados relacionais | 371 |
| Enumerar entidade e atributos de entidades | 373 |
| Executar uma consulta | 375 |
| Criar conectores de servidores de conteúdos personalizados | 378 |
| Desenvolver conectores de servidores de conteúdos personalizados | 378 |
| Utilizar a classe FeServerDefBase (apenas Java) | 394 |

Construir aplicações de fluxo de trabalho de EIP 395

| | |
|--|-----|
| Ligação aos serviços do fluxo de trabalho | 395 |
| Iniciar um fluxo de trabalho | 396 |
| Terminar um fluxo de trabalho | 397 |
| Enumerar todos os fluxos de trabalho | 398 |
| Suspender um fluxo de trabalho | 399 |
| Retomar um fluxo de trabalho. | 400 |
| Enumerar todas as listas de trabalhos | 400 |
| Aceder a uma lista de trabalhos | 401 |
| Aceder a artigos de trabalho | 402 |
| Mover artigos no fluxo de trabalho | 403 |
| Enumerar todos os modelos de fluxo de trabalho | 404 |
| Criar as suas próprias acções (Apenas para Java) | 405 |

Construir aplicações com JavaBeans visuais e não visuais 407

| | |
|---|-----|
| Compreender conceitos básicos de beans | 407 |
| Utilizar JavaBeans em construtores | 408 |
| Utilizando o IBM Websphere Studio Application Developer | 409 |
| Invocar os beans de Java do EIP | 409 |
| Beans não visuais | 410 |
| Configurações de beans não visuais | 411 |
| Compreender as funções dos beans não visuais | 411 |
| Categorias de beans não visuais | 411 |

| | |
|--|-----|
| Considerações a ter aquando da utilização de beans não visuais | 416 |
| Rastrear e iniciar sessão nos Beans | 417 |
| Compreender propriedades e eventos de beans não visuais | 417 |
| Construir uma aplicação através de beans não visuais | 417 |
| Trabalhar com beans visuais | 418 |
| Bean CMBLogonPanel | 419 |
| Bean CMBSearchTemplateList | 420 |
| Bean CMBSearchTemplateViewer | 421 |
| Validar ou editar campos de CMBSearchTemplateViewer | 422 |
| Bean CMBSearchPanel | 422 |
| Bean CMBSearchResultsViewer | 422 |
| Substituir menus emergentes | 424 |
| Bean CMBFolderViewer | 424 |
| Bean CMBDocumentViewer | 425 |
| Especificações do visualizador | 425 |
| Visualizadores predefinidos | 426 |
| Iniciar visualizadores externos | 426 |
| Bean CMBItemAttributesEditor | 426 |
| Proibir alterações no CMBItemAttributesEditor | 427 |
| Bean CMBVersionsViewer | 427 |
| Comportamentos gerais de beans visuais | 427 |
| Substituir um bean visual | 428 |
| Construir uma aplicação através de beans visuais | 428 |

Trabalhar com o conjunto de ferramentas do visualizador de documentos de Java 431

| | |
|---|-----|
| Arquitectura do visualizador | 432 |
| Os motores de documentos | 432 |
| O motor de anotações | 433 |
| Criar um visualizador de documentos genérico | 433 |
| Personalizar o visualizador genérico de documentos | 433 |
| Aplicações exemplo | 435 |
| Visualizador autónomo | 435 |
| Aplicação Java | 436 |
| Cliente com poucos recursos | 436 |
| Applet ou servlet | 437 |
| Modo dual e applet ou servlet | 437 |
| Trabalhar com serviços de anotação | 438 |
| Utilizar interfaces de serviços de anotação | 439 |
| Compreender o suporte de edição de anotações | 440 |
| Construir uma aplicação utilizando os serviços de anotação | 441 |
| Adicionar um tipo de anotação personalizado à aplicação do utilizador | 441 |

Trabalhar com a biblioteca de identificadores e o servlet controlador do Enterprise Information Portal . . . 443

| | |
|--|-----|
| Configurar a biblioteca do identificador e o servlet | 443 |
|--|-----|

| | |
|--|-----|
| Utilizar a biblioteca de identificadores | 443 |
| Convenções utilizadas na biblioteca do identificador | 444 |
| Resumo do identificador | 444 |
| Identificadores relacionados com ligações | 444 |
| Identificadores relacionados com esquemas | 445 |
| Identificadores relacionados com pesquisa | 446 |
| Identificadores relacionados com artigos | 446 |
| Identificadores relacionados com pastas | 447 |
| Identificadores relacionados com documentos | 448 |
| Servlet controlador do EIP | 448 |
| O que o servlet pode fazer | 448 |
| Referência de servlet | 450 |
| Matrix de função de toolkit do servlet | 455 |

Trabalhar com extracção de informações 457

| | |
|--|-----|
| Construir uma aplicação de Information Mining (extracção de informações) utilizando os beans | 457 |
| Localização dos ficheiros exemplo | 460 |
| Categorizar documentos | 461 |
| Resumir documentos | 469 |
| Extrair informações | 475 |
| Agrupamento | 480 |
| Importar documentos a partir de um espaço da web | 485 |
| Pesquisar documentos por categoria | 494 |
| Construir o seu fornecedor de conteúdos | 499 |
| Utilizar a API de serviço | 500 |
| Estabelecer ligação à API de serviço de extracção de informações | 501 |
| Gerir a biblioteca, taxonomias e catálogos | 502 |
| Utilizar as ferramentas de Information Mining | 505 |
| Criar registos e armazenar metadados em catálogos | 507 |
| Pesquisar documentos | 509 |
| Executar uma tarefa de servidor | 509 |
| Exemplo de uma aplicação de Information Mining baseada em JSPs | 511 |
| Instalar as JSPs | 512 |

Informações 513

| | |
|-----------------------------|-----|
| Marcas Comerciais | 515 |
|-----------------------------|-----|

Glossário 517

Índice Remissivo 527

Acerca deste manual

Este manual descreve como utilizar Java, JavaBeans, e interfaces de programação de aplicação (APIs) de C++ com Enterprise Information Portal Versão 8 Edição 2 (EIP) e Content Manager (CM) Versão 8 Edição 2. As APIs e beans facultam blocos de construção para criar aplicações para aceder ao conteúdo armazenado em servidores de conteúdo heterogéneos.

Em versões anteriores, o CM e o EIP possuíam manuais de programação de aplicações diferentes. Na Versão 8 Edição 2, os dois produtos partilham muitas das APIs e têm por base muitos dos mesmos conceitos de programação. Para além disso, devido ao facto de grande parte da nova funcionalidade do Enterprise Information Portal Versão 8 Edição 2 relacionar o novo conector ao CM Versão 8 Edição 2, ambos os manuais foram fundidos num só.

Este manual inclui:

- Uma introdução aos conceitos de programação de aplicações de Enterprise Information Portal e CM, incluindo conceitos de objecto de dados dinâmico no contexto de Java e C++
- Uma descrição da função acessível através do conector da Versão 8 Edição 2 CM
- Documentação sobre todos os outros conectores de Enterprise Information Portal dos servidores de conteúdo
- Actualizações de JavaBeansvisual e não-visual
- Actualizações para programar informações em Information Mining, IBM Web Crawler e fluxo de trabalho

As ilustrações referentes a Content Manager implicam tanto a pré-Versão 8.1 como a Versão 8 Edição 2 do produto.

Quem deve utilizar este manual

Este manual destina-se a programadores de aplicações com algumas ou todas as seguintes capacidades:

- Experiência com C++, Java, JavaBeans ou HTML
- Familiaridade com conceitos de base de dados relacionais
- Conhecimento do protocolo de DDO/XDO

Onde encontrar mais informações

O pacote do produto inclui um conjunto completo de informações para o ajudar a planear, instalar, administrar e utilizar o sistema. A documentação e assistência do produto estão também disponíveis na Web.

As informações incluídas na embalagem do produto

O pacote de produto contém um centro de informações e todas as publicações em formato de documento portátil (.PDF).

O centro de informações

O pacote de produto contém um centro de informações que pode ser instalado quando instalar o produto. Para obter informações sobre o centro de informações, consulte *Planejar e Instalar o Content Management System*.

O centro de informações contém a documentação para Content Manager, Enterprise Information Portal e Content Manager VideoCharger. As informações referentes aos tópicos estão organizadas por produto e por tarefa (por exemplo, Administração). Para além dos índices e dos mecanismos de navegação, está disponível uma função de pesquisa que facilita o acesso à informação.

Publicações em PDF

Pode visualizar os ficheiros PDF online utilizando o Adobe Acrobat Reader para o seu sistema operativo. Se não tiver o Acrobat Reader instalado, pode descarregá-lo do site da Web da Adobe em www.adobe.com.

A Tabela 1 mostra as publicações do Content Manager incluídas no Content Manager para Multiplataformas da IBM.

Tabela 1. Publicações do Content Manager

| Nome do ficheiro | Título | Número da publicação |
|------------------|---|----------------------|
| instalar | <i>Planejar e Instalar o Content Management System</i> ¹ | SC17-5411-01 |
| migrar | <i>Migração para o Content Manager Versão 8</i> | SC17-5425-01 |
| sysadmin | <i>Manual de Administração do Sistema</i> | SC17-5428-01 |

Quando encomenda Content Manager para Multiplataformas da IBM, recebe também IBM Enterprise Information Portal for Multiplatforms. Ou pode encomendar IBM Enterprise Information Portal for Multiplatforms separadamente. Tabela 2 apresenta as publicações Enterprise Information Portal incluídas com o produto.

Tabela 2. Publicações do Enterprise Information Portal

| Nome do ficheiro | Título | Número da publicação |
|------------------|---|----------------------|
| apgwork | <i>Manual de Programação de Aplicações de Estações de Trabalho</i> ¹ | SC17-5414-01 |
| ecliinst | <i>Instalação, Configuração e Gestão do eClient</i> | SC17-5416-02 |
| eipinst | <i>Planejar e Instalar o Enterprise Information Portal</i> | GC17-5406-01 |
| eipmanag | <i>Gerir o Enterprise Information Portal</i> | SC17-5413-01 |
| messcode | <i>Mensagens e Códigos</i> ² | SC17-5415-01 |

Notas:

1. O *Manual de Programação de Aplicações de Estações de Trabalho* contém informações acerca das aplicações de programação para Content Manager e Enterprise Information Portal.
2. *Mensagens e Códigos* contém as mensagens e códigos para Content Manager e Enterprise Information Portal.

Suporte disponível na Web

O suporte do produto está disponível na Web. Faça clique sobre **Support** no sítio da Web do produto em:

www.ibm.com/software/data/cm/

www.ibm.com/software/data/eip/

A documentação está incluída sob a forma de suporte electrónico no produto. Para aceder à documentação sobre o produto na Web, clique em **Library** sítio da Web do produto.

Uma interface de documentação baseada em HTML, denominada Enterprise Documentation Online (EDO), está também disponível a partir da Web. Contém actualmente as informações de referência da API. Consulte a página da Web Library do Enterprise Information Portal para obter informações sobre como aceder ao EDO.

Como enviar comentários

Os comentários do utilizador ajudam a IBM a fornecer informações de qualidade. Envie quaisquer comentários que tenha sobre esta publicação ou outra documentação do Content Manager ou do Enterprise Information Portal. Pode utilizar um dos seguintes métodos para enviar os seus comentários:

- Através da Web. Visite a página do formulário para comentários do IBM Data Management Online Reader (RCF) em:
www.ibm.com/software/data/rcf
Pode utilizar a página para introduzir e enviar comentários.
- Envie os seus comentários por correio electrónico para comments@vnet.ibm.com. Certifique-se de que inclui o nome do produto, o número da versão e o nome e parte do manual (se aplicável). Se pretender tecer comentários acerca de um texto específico, inclua a localização do texto (por exemplo, o título de um capítulo ou de uma secção, o número de uma tabela, de uma página ou o título de um tópico da ajuda).

Novidades no Enterprise Information Portal Versão 8

Versão 8.2: A Versão 8.2 inclui uma variedade de melhorias. A Versão 8.2 adiciona mais funcionalidade ao fluxo de trabalho de administração do sistema e suporta a mais recente tecnologia de base de dados, o DB2 Universal Database Versão 8.1. Estes destaques e outras melhorias efectuadas no produto Versão 8.2, são resumidos em baixo:

Mudança de nome do Enterprise Information Portal para IBM Information Integrator for Content

Ao Enterprise Information Portal foi atribuído outro nome: Information Integrator for Content. Apesar dos nomes dos manuais terem sido alterados na Versão 8.2, o texto incluído nos manuais continua a apresentar o nome de produto Enterprise Information Portal. Ao pesquisar mais informações na Web, pode continuar a utilizar Enterprise Information Portal ou EIP, até que a transição para o novo nome seja concluída.

Suporte para DB2 UDB V8.1

Suportes de Enterprise Information Portal V8.2. A função de concentração de ligação de DB2 V8.1 faculta escalabilidade aumentada para aplicações e clientes de dois escalões.

Pastas associadas

O eClient actualmente possui a capacidade de organizar documentos e

pastas nativas de vários repositórios numa única pasta associada e iniciar essa pasta num fluxo de trabalho. As pastas associadas também permitem aos utilizadores armazenar permanentemente resultados de pesquisa na base de dados associada de EIP, na qual os utilizadores os podem obter em qualquer altura. As opções completas de CRUD (criação, obtenção, actualização e eliminação) estão disponíveis nestas pastas associadas sem re-indexação.

Pontos de recolha de fluxo de trabalho avançado

O fluxo de trabalho é agora totalmente suportado em AIX e Solaris. O construtor de fluxo de trabalho, as APIs, o Supervisor de Pontos de Recolha e os JavaBeans facultam uma função e utilização melhorada do fluxo de trabalho.

Microsoft Visual Studio .NET para construir aplicações

O Enterprise Information Portal 8.1 e APIs de versões posteriores actualmente suportam o Microsoft Visual Studio .NET para escrever aplicações de gestão de conteúdo ou para integrar aplicações contruídas utilizando o Microsoft Visual Studio .NET.

Versão 8.1: A Versão 8.1 inicia uma herança de integração e versatilidade. Um dos muitos destaques e melhorias realizadas em relação a produtos anteriores de Content Manager é a nova estrutura de modelo de dados, que permite uma maior personalização dos documentos. As alterações realizadas ao produto Content Manager na Versão 8.1 encontram-se resumidas de seguida:

Suporte para Sun Solaris

Pode instalar conectores, funções e bases de dados de Java em sistemas Solaris.

Administração de sistema comum

Uma aplicação cliente única faculta acesso separado à administração do Content Manager e Enterprise Information Portal.

Novos conectores

- O conector de ICM para o Content Manager Versão 8 Edição 1 permite-lhe tirar partido das eficazes funções de armazenamento de documentos do Content Manager Versão 8.
- O novo conector de C++ Extended Search Versão 3.7 é executável em AIX.

Conectores melhorados

- As pesquisas paramétricas de texto são suportadas a partir do nível associado e através de uma ligação directa de Extended Search.
- Os melhoramentos funcionais e os melhoramentos de rendimento realizados no conector de OnDemand incluem:
 - Modificações na estrutura de um DDO de OnDemand.
 - Pesquisa assíncrona suportada

IBM Web Crawler

O IBM Web Crawler é uma função que permite aos utilizadores pesquisar e resumir informações na Web e em bases de dados de Lotus Notes.

Melhoramentos de fluxo de trabalho

O fluxo de trabalho é agora totalmente suportado em AIX e Solaris. O construtor, APIs e JavaBeans do fluxo de trabalho facultam uma função e utilização melhoradas de fluxo de trabalho.

Centro de informações

O centro de informações com base em browser inclui a documentação de Content Manager, Enterprise Information Portal e Content Manager VideoCharger. As informações referentes aos tópicos estão organizadas por produto e por tarefa (por exemplo, Administração). Para além dos índices e dos mecanismos de navegação, está disponível uma função de pesquisa que facilita o acesso à informação.

Acessibilidade

As funções de acessibilidade auxiliam um utilizador que possui uma deficiência física como, por exemplo, mobilidade restringida ou visão limitada, a utilizar produtos de software com sucesso. As principais funções de acessibilidade deste produto incluem:

- A capacidade de funcionar com todas as funções utilizando o teclado em vez do rato
- Suporte para propriedades de visualização melhoradas.
- Opções para sinais de alerta vídeo e áudio
- Compatibilidade com tecnologias auxiliares
- Compatibilidade com funções de acessibilidade do sistema operativo
- Formatos de documentação acessíveis

Novidades no Content Manager Versão 8

Versão 8.2: A Versão 8.2 inclui uma variedade de melhorias acima da Versão 8.1. A Versão 8.2 adiciona mais funções de fluxo de trabalho ao eClient, aumenta a função de gestão de recursos e suporta a mais recente tecnologia de base de dados e cliente, incluindo DB2 Universal Database Versão 8.1, Oracle Versão 8.1.7.4 e Versão 9.2.0.1 e WebSphere Versão 5. Estes destques e outras melhorias efectuadas ao produto Versão 8.2, encontram resumidas em baixo:

Mudança de nome do Enterprise Information Portal paraIBM Information Integrator for Content

Ao Enterprise Information Portal foi atribuído outro nome: Information Integrator for Content. Apesar dos nomes dos manuais terem sido alterados na Versão 8.2, o texto incluído nos manuais continua a apresentar o nome de produto Enterprise Information Portal. Ao pesquisar mais informações na Web, pode continuar a utilizar Enterprise Information Portal ou EIP, até que a transição para o novo nome seja concluída.

Suporte para Oracle Versão 8.1.7.4, Versão 9.2.0.1 ou posterior

O Content Manager V8.2 adiciona suporte para bases de dados de Oracle que gerem os metadados armazenados no servidor de bibliotecas e no gestor de recursos. As ferramentas de migração estão incluídas para utilizadores de Oracle do Content Manager Versão 7. **Nota:** O Oracle não gere conteúdos de servidor de bases de dados do Enterprise Information Portal.

Replicação

O Content Manager V8.2 inclui replicação de gestor de recursos, que consiste na capacidade de armazenar objectos em várias localizações, geridas por gestor de recursos de replicação. As réplicas de objectos irão comportar-se como objectos de memória cache de LAN para equilíbrio melhorado de carregamento.

Memória cache de LAN

O suporte de memória de cache de LAN no Content Manager V8.2 faculta uma colocação em memória cache transparente a nível de aplicações, utilizando servidores locais, tal como está definido pelo administrador do sistema.

Suporte para DB2 UDB V8.1

Os Content Manager V8.2 e Enterprise Information Portal V8.2 suportam DB2/UDB V8.1. A função de concentração de ligação de DB2 V8.1 faculta escalabilidade aumentada para aplicações e clientes de dois escalões (como, por exemplo o Content Manager V8 Client for Windows). O DB2/UDB V8.1 substituiu o DB2 Universal Database Text Information Extender (TIE) pelo Net Search Extender (NSE).

Suporte para WebSphere Application Server Versão 4 e Versão 5

O WebSphere Application Server Versão 5 apresenta a implementação de servidores e acesso e gestão de dados a partir de qualquer browser da web.

Pastas associadas

O eClient actualmente possui a capacidade de organizar documentos e pastas nativas de vários repositórios numa única pasta associada e iniciar essa pasta num fluxo de trabalho. As pastas associadas também permitem aos utilizadores armazenar permanentemente resultados de pesquisa na base de dados associada de EIP, na qual os utilizadores os podem obter em qualquer altura. As opções completas de CRUD (criação, obtenção, actualização e eliminação) estão disponíveis nestas pastas associadas sem re-indexação.

Pontos de recolha de fluxo de trabalho avançado

O fluxo de trabalho é agora totalmente suportado em AIX e Solaris. O construtor de fluxo de trabalho, as APIs, o Supervisor de Pontos de Recolha e os JavaBeans facultam uma função e utilização melhorada do fluxo de trabalho.

Microsoft Visual Studio .NET para construir aplicações

O Content Manager e Enterprise Information Portal 8.1 e APIs de versões posteriores actualmente suportam o Microsoft Visual Studio .NET para escrever aplicações de gestão de conteúdo ou para integrar aplicações contruídas utilizando o Microsoft Visual Studio .NET.

Versão 8.1: A Versão 8.1 inicia uma herança de integração e versatilidade. Um dos muitos destaques e melhorias realizadas em relação a produtos anteriores de Content Manager é a nova estrutura de modelo de dados, que permite uma maior personalização dos documentos. As alterações realizadas ao produto Content Manager na Versão 8.1 encontram-se resumidas de seguida:

Rendimento melhorado

O servidor de bibliotecas e o gestor de recursos utilizam procedimentos armazenados de DB2 e potenciam tecnologia de DB2 para reduzir significativamente o tráfego de rede e melhorar o rendimento e a escalabilidade.

Suporte para Sun Solaris

Tanto o servidor de bibliotecas como o gestor de recursos podem ser instalados em Sun Solaris.

Modelo de dados avançado

O novo modelo de dados hierárquico faculta a base para soluções personalizadas de gestão de documentos compostos.

Fluxo de trabalho melhorado

Através do encaminhamento de documentos integrados, as capacidades de fluxo de trabalho foram melhoradas com o encaminhamento sequencial, o encaminhamento dinâmico e os pontos de recolha.

Pesquisa de texto integrada

Para além da pesquisa com base em atributos, os utilizadores clientes podem agora executar pesquisas completas de texto em informações de documentos com base em texto. A função de pesquisa de texto actualmente utiliza o DB2 Universal Database Text Information Extender, que contribui para um processo que evita obstáculos para configurar a pesquisa de texto.

Administração de sistema comum

Uma aplicação cliente única faculta acesso separado ao Content Manager e Enterprise Information Portal. Dentro do Content Manager, os domínios administrativos facultam uma forma de limitar o acesso administrativo a sub-secções do servidor de bibliotecas.

Cliente de ambiente de trabalho de função total e eclient melhorado

Os melhoramentos ao cliente facultam aos utilizadores uma aplicação pronta a utilizar para uma implementação rápida ou para uma integração de aplicação de actividade. O Client for Windows suporta pesquisa de texto integrada, encaminhamento de documentos, o modelo de dados hierárquico (Para um nível de componente descendente único), elaboração de versões e índice durante a importação. O eClient inclui pesquisa de texto integrada, fluxo de trabalho avançado de EIP, controlo de versões e atributos de vários valores.

Instalação mais simples

A instalação é coerente em sistemas operativos suportados e as informações relativas à instalação personalizada são facultadas pelo Assistente de Planeamento do CD Iniciar Aqui. As instalações não assistida e de consola são também facultadas.

Centro de informações

O centro de informações com base em browser inclui a documentação de Content Manager, Enterprise Information Portal e Content Manager VideoCharger. As informações referentes aos tópicos estão organizadas por produto e por tarefa (por exemplo, Administração). Para além dos dos índices e dos mecanismos de navegação, está disponível uma função de pesquisa que facilita a obtenção.

Acessibilidade

As funções de acessibilidade auxiliam um utilizador que possui uma deficiência física como, por exemplo, mobilidade restringida ou visão limitada, a utilizar produtos de software com sucesso. As principais funções de acessibilidade deste produto incluem:

- A capacidade de funcionar com todas as funções utilizando o teclado em vez do rato
- Suporte para propriedades de visualização melhoradas.
- Opções para sinais de alerta vídeo e áudio
- Compatibilidade com tecnologias auxiliares
- Compatibilidade com funções de acessibilidade do sistema operativo

- Formatos de documentação acessíveis

Integrações PeopleSoft e Siebel

Os utilizadores de aplicações de PeopleSoft e Siebel já podem configurar estas aplicações para acederem ao conteúdo armazenado numa variedade de servidores de conteúdo utilizando eClient.

Introdução ao Enterprise Information Portal

Muitas empresas com muito movimento de documentos, como, por exemplo, companhias de seguros e instituições financeiras, administram grandes volumes de conteúdo relacionado com negócios. A necessidade de uma solução de empresa para gerir e aceder a informações empresariais é comum a muitas indústrias.

Um *servidor de conteúdos* é um sistema de software que guarda ficheiros de multimédia, formulários de negócios, documentos e dados relacionados, com metadados que permitem aos empregados processar e trabalhar com o conteúdo. Quando não existe um modo de ligar eficazmente servidores de conteúdos diferentes, uma empresa pode perder tempo e dinheiro com a duplicação de informações ou a formação de empregados para a execução de múltiplas pesquisas.

O Enterprise Information Portal fornece tecnologia de ponta para reunir todos os recursos no ambiente de trabalho da sua estação de trabalho. O IBM Enterprise Information Portal for Multiplatforms pode ajudar a maximizar o valor das informações e dos recursos multimédia ligando servidores de conteúdos diferentes através de um único cliente. Com um cliente do IBM Enterprise Information Portal for Multiplatforms, os utilizadores podem rapidamente e simultaneamente aceder a todos os servidores de conteúdos ligados. Os utilizadores podem também efectuar a extracção de informações ou pesquisas “inteligentes” em servidores de conteúdos, incluindo a Web ou uma intranet e podem executar tarefas do fluxo de trabalho nos processos de negócios.

Com o IBM Enterprise Information Portal for Multiplatforms, o utilizador pode personalizar aplicações para a sua empresa. Através dos conjuntos de ferramentas de do IBM Enterprise Information Portal for Multiplatforms, os programadores de aplicações podem escrever aplicações de ambiente de trabalho e aplicações baseadas na Web.

Este capítulo apresenta uma descrição geral de cada IBM Enterprise Information Portal for Multiplatforms. Uma situação acerca de uma companhia de seguros fictícia, a Seguradora XYZ, demonstra as funções e capacidades do IBM Enterprise Information Portal for Multiplatforms.

Pesquisar informações de clientes

A Seguradora XYZ (XYZ), uma grande companhia de seguros multi-riscos, tem uma extensa colecção de fotografias, queixas, apólices, notas de avaliadores, relatórios de peritos e outros documentos empresariais.

A XYZ mantém todos os memorandos enviados aos titulares de apólices, junto com formulários médicos e de avaliação electrónicos em arquivos de ficheiros Lotus Domino.Doc. A XYZ arquiva todas as declarações de apólice, notificações e facturas num servidor Content Manager OnDemand para armazenamento a longo prazo e acesso rápido. A XYZ armazena todos os impressos, fotografias e cartas de participação recebidas dos detentores de apólice numa pasta de sistema de Content Manager para AS/400. A XYZ mantém relatórios de peritos num DB2 Universal Database (DB2 UDB) Data Warehouse Center Information Catalog Manager. A XYZ armazena também conteúdos multimédia, tais como gráficos de alta resolução num sistema Content Manager para serem partilhados pelos departamentos de

publicidade, relações públicas e novos negócios. Além disso, a XYZ mantém informações, como por exemplo os procedimentos empresariais, na intranet da empresa.

A necessidade

As participações, as chamadas de clientes e a assistência geral a titulares de apólices não pode ser tratada com o conteúdo de um servidor porque os empregados têm necessidade de aceder a todas as informações de cliente. Para fornecer assistência a clientes, os empregados necessitam simultaneamente de aceder a uma série de servidores. A Seguradora XYZ precisa de uma solução que ligue todos os seus servidores de conteúdos e a intranet da empresa para pesquisa e obtenção de informação. Também pretendem expandir a sua utilização do processamento de fluxos de trabalho.

Muitos empregados necessitam de aceder a documentos, dos funcionários aos peritos de seguros aos agentes. A XYZ tem de limitar o acesso a certos artigos e facultar acesso ilimitado a outros. A XYZ também precisa de uma interface de fácil utilização para reduzir a necessidade de formação.

A solução

A Seguradora XYZ prefere a distribuição do IBM Enterprise Information Portal for Multiplatforms porque as tecnologias de pesquisa exaustiva lhe permite ligar-se e pesquisar todos os servidores de conteúdos para obtenção de dados. Actualmente, quando um funcionário do Centro de Atendimento da XYZ recebe uma chamada, basta uma única pesquisa associada para obter todas as informações necessárias acerca do titular da apólice.

A Seguradora XYZ também recorre à função de Information Mining do Enterprise Information Portal para pesquisar e obter informações da intranet da empresa. Pretendem também expandir a sua utilização dos processos de fluxo de trabalho.

A solução de Enterprise Information Portal

O IBM Enterprise Information Portal for Multiplatforms é um produto global: os seus componentes trabalham em conjunto para fornecer uma solução exclusiva adequada à sua empresa. Centrado numa arquitectura de múltiplos níveis, o IBM Enterprise Information Portal for Multiplatforms fornece um cliente de administração para gerir pesquisas, clientes (como exemplos) para executar pesquisas e conectores para ligar a servidores de conteúdos diferentes, tais como o Content Manager, Content Manager ImagePlus para OS/390, Content Manager OnDemand, Lotus Domino.Doc, DB2 UDB, DB2 DataJoiner e o DB2 Data Warehouse Center Information Catalog Manager. Pode escrever conectores adicionais para servidores de conteúdos adicionais.

O Figura 1 na página 3 mostra o conceito de arquitectura de escalões múltiplos do IBM Enterprise Information Portal for Multiplatforms.

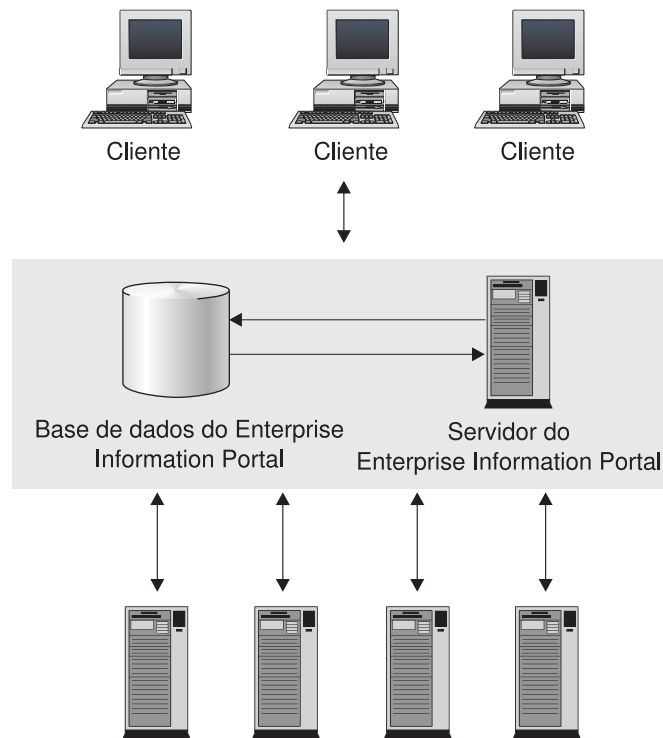


Figura 1. Arquitectura de múltiplos níveis

A arquitectura do IBM Enterprise Information Portal for Multiplatforms permite às aplicações cliente efectuar pesquisas únicas num ou mais servidores de conteúdos. Para executar pesquisas, um cliente utiliza modelos de pesquisas que foram definidos pelo administrador do IBM Enterprise Information Portal for Multiplatforms.

Ao utilizar um modelo de pesquisa, o cliente executa uma pesquisa associada. Uma *pesquisa associada* é uma pesquisa que é executada simultaneamente em vários servidores de conteúdos, cujos atributos nativos foram previamente correlacionados aos atributos associados utilizados no modelo de pesquisa. Os modelos de pesquisa do IBM Enterprise Information Portal for Multiplatforms contêm critérios de pesquisa que fazem referência aos atributos associados, que correspondem aos atributos nativos de cada um dos servidores de conteúdos. O administrador do IBM Enterprise Information Portal for Multiplatforms cria os modelos de pesquisa. A IBM Enterprise Information Portal for Multiplatforms faculta conectores como uma interface comum para interfaces heterogéneas de servidores de conteúdos. Os servidores de conteúdos devolvem então objectos de dados ao cliente.

A arquitectura do IBM Enterprise Information Portal for Multiplatforms fornece as seguintes vantagens:

- Aceder utilizando uma consulta única para servidores múltiplos e de conteúdo variável que suportam transacções de e-business e aplicações de assistência ao cliente.
- A capacidade de extracção de informações em vários servidores de conteúdos, incluindo a Web.
- Acesso de processo de fluxo de trabalho a dados em servidores de conteúdos múltiplos e heterogéneos.

- Suporte para o desenvolvimento de aplicações cliente independentes da localização de dados em qualquer servidor de conteúdos, devido à separação de aplicações cliente, índices remissivos e dados.

Componentes do IBM Enterprise Information Portal for Multiplatforms

Esta secção descreve cada componente do IBM Enterprise Information Portal for Multiplatforms. Estes componentes são fornecidos como parte do produto IBM Enterprise Information Portal for Multiplatforms.

Base de dados de administração

A base de dados de IBM Enterprise Information Portal para Multiplatforms é uma base de dados de DB2 UDB. Esta armazena todas as informações de que necessita para gerir o EIP e os seus componentes.

Migrar a base de dados de Enterprise Information Portal Versão 7.1 do utilizador:

Deve migrar os dados do utilizador de Enterprise Information Portal Versão 7.1 *antes* de utilizar a base de dados de administração de Enterprise Information Portal Versão 8 Edição 2.

Cliente de Administração

O administrador do sistema utiliza o cliente de administração do IBM Enterprise Information Portal for Multiplatforms para:

- Definir cada servidor de conteúdos para pesquisa associada.
- Identificar entidades e atributos nativos em servidores de conteúdos e correlacioná-los a entidades associadas.
- Criar modelos de pesquisa.
- Identificar e gerir utilizadores que podem aceder a modelos de pesquisa, à função extracção de informações e a processos de fluxo de trabalho.
- Definir processos de fluxo de trabalho.

Estas informações estão armazenadas na base de dados do IBM Enterprise Information Portal for Multiplatforms.

Por uma questão de comodidade, recomenda-se que instale o cliente de administração na mesma estação de trabalho ou servidor da base de dados IBM Enterprise Information Portal for Multiplatforms.

Pode também ter tantos clientes de administração quantos quiser noutras estações de trabalho. Para efectuar esta configuração, é necessário proceder de um dos seguintes modos:

- Instale o suporte de DB2 Client e utilize o Assistente de Configuração de DB2 Client para configurar o acesso à base de dados de administração do sistema em todas as estações de trabalho nas quais o cliente de administração está instalado.
- Utilize uma configuração de Remote Method Invocation (RMI), iniciando o servidor de RMI em que a base de dados do IBM Enterprise Information Portal for Multiplatforms está instalada. O utilizador deve certificar-se de que o ficheiro `\CMBROOT\cmbclient.ini` aponta para este servidor. A localização deste ficheiro INI é o local onde o directório está especificado por uma palavra-chave CMCOMMON no ficheiro `cmbcmenv.properties`. Este ficheiro também se pode encontrar num URL especificado pela palavra-chave CMCOMMON_URL no ficheiro `cmbcmenv.properties`.

Conectores

As classes de conectores permitem às aplicações cliente aceder a servidores de conteúdos e à base de dados do IBM Enterprise Information Portal for Multiplatforms. O IBM Enterprise Information Portal for Multiplatforms fornece os seguintes conectores:

- Conectores de base de dados relacional (DB2, JDBC, ODBC)
- Conector associado (à base de dados do IBM Enterprise Information Portal for Multiplatforms)
- Content Manager Versão 8 Edição 2
- Conector do Content Manager Versão 7 Edição 1
- Conector de Content Manager OnDemand
- Conector do Content Manager ImagePlus for OS/390
- Conector do Content Manager for AS/400
- Conector do Lotus Domino.Doc
- Conector de Extended Search

O conector associado contém a classe de conectores para a base de dados do IBM Enterprise Information Portal for Multiplatforms. Cada servidor de conteúdos contém as classes adequadas do conector.

Num ambiente de Java, as versões Java dos conectores são locais ou remotas. Em C++, existem apenas conectores locais. Os conectores locais são um conjunto de classes de conector utilizados para ligar correctamente a vários servidores de conteúdos. Os conectores locais podem residir num cliente de ambiente de trabalho do IBM Enterprise Information Portal for Multiplatforms ou num servidor de RMI do IBM Enterprise Information Portal for Multiplatforms. Os conectores remotos são utilizados para estabelecer ligação a um servidor de conteúdos através de um servidor RMI ou de um membro de conjunto de servidores RMI. A utilização do conector remoto elimina a necessidade de ligar directamente ao servidor de conteúdos.

IBM eClient

O eClient é uma interface de utilizador com base em browser para aceder a documentos armazenados em Content Manager (todas as plataformas), Content Manager OnDemand (todas as plataformas) e Content ManagerImagePlus for OS/390.

Information Mining

O Information Mining fornece serviços linguísticos para localizar informações ocultas em documentos ou em servidores de conteúdos. Durante o processamento dos documentos de texto, são criados metadados (informações sobre dados) que podem ser resumidos, classificados e pesquisados. O IBM Enterprise Information Portal for Multiplatforms fornece exemplos que demonstram a forma de utilização das capacidades do Information Mining num cliente com poucos recursos. O utilizador pode construir um cliente de ambiente de trabalho ou um cliente com poucos recursos para trabalhar com Information Mining.

Web Crawler da IBM

O Web Crawler da IBM permite-lhe pesquisar e importar conteúdos da web. Os resultados da pesquisa e os metadados podem então ser analisados e catalogados através das ferramentas de Information Mining. O Web Crawler da IBM pode ir buscar servidores de http, ftp, ntp, lotus e de domino.

Fluxo de trabalho

Com o Enterprise Information Portal, pode controlar o fluxo de trabalho da sua empresa. Utilizando a função fluxo de trabalho do Enterprise Information Portal, pode definir e executar o processo de fluxo de trabalho de um grupo de trabalho, departamento ou empresa. Utilizando um construtor gráfico, é possível construir uma representação gráfica global e fácil de compreender de um processo de fluxo de trabalho no construtor do fluxo de trabalho do Enterprise Information Portal. Os utilizadores podem utilizar o processo de fluxo de trabalho definido para executar tarefas, utilizando um cliente desenvolvido ou a exemplos de cliente de poucos recursos do Enterprise Information Portal.

Motor de Pesquisa de Texto do Content Manager Versão 7

Pode utilizar esta função para indexar, pesquisar e obter automaticamente documentos armazenados no Content Manager Versão 7. Os utilizadores podem localizar documentos pesquisando palavras ou expressões.

Restrição: O servidor e cliente de pesquisa de texto é uma função facultativa do Content Manager que o utilizador pode configurar e executar apenas com servidores de pré-Versão 8.1 Content Manager. Se não utilizar servidores de pré-Versão 8.1 Content Manager, não instale esta função.

Cliente e servidor de pesquisa de imagens do Content Manager Versão 7

Esta função utiliza tecnologia IBM QBIC (consulta por conteúdo de imagem) com a qual pode pesquisar objectos por determinadas propriedades visuais, como a cor e a textura.

Restrição: A funcionalidade do servidor de pesquisa de imagens não é suportada no Enterprise Information Portal Versão 8 e Content Manager Versão 8. A sua funcionalidade ainda está disponível através da utilização de pré-Versão 8.1 Content Manager.

Novidades nas APIs da Versão 8.2

Exemplos de código abrangentes

Os exemplos de código foram actualizados para as funções do Content Manager Versão 8.2. Para além disso, todos os exemplos de código actualmente são apresentados em caixas (identificados segundo o idioma) para uma melhor legibilidade.

Restrições do DB2 DataJoiner

Se o seu servidor de conteúdos necessitar de DB2 Universal Database Versão 8.1, nesse caso, o conector DataJoiner do EIP não funcionará devido a problemas de associação. Isto deve-se ao facto do DataJoiner 2.1 não reconhecer instruções de SQL nos ficheiros de enlace do DB2 Versão 7.

Para solucionar este problema, leia a pergunta FAQ no site da Web do DataJoiner (<http://www.software.ibm.com/data/datajoiner/>) : "Why am I having problems with binding on DB2 Universal Database Version 7 when connecting to a DataJoiner server?". A resposta irá dar-lhe instruções relativas aos passos seguintes de ligação ao servidor de DataJoiner Versão 2.1.1:

1. Descarregue os dois ficheiros de enlace e atribua-lhes os nomes `db2cliws_dj.bnd` e `db2clprp_dj.bnd`.
2. Introduza os seguintes comandos de DB2:

```
db2cmd
db2 connect to user using
db2 bind db2cliws_dj.bnd grant public
db2 bind db2cliprr_dj.bnd grant public
```

Restrições do Gestor de Catálogo de Informações de DB2 Data Warehouse Manager

O concetor de IC necessita de DB2 Universal Database Versão 7.2 e não funcionará com DB2 UDB Versão 8.1.

Pastas associadas (Apenas para Java)

O Enterprise Information Portal Versão 8 Edição 2 actualmente faculta entidades associadas especiais que suportam *pastas associadas*. Estas pastas associadas podem armazenar os resultados combinados de uma consulta associada como, por exemplo, um documento do Content Manager e um documento relacionado do OnDemand. O utilizador pode enviar os resultados directamente para um fluxo de trabalho.

Suporte de Microsoft Visual Studio .NET

As APIs do Enterprise Information Portal e do Content Manager versão 8.2 (e versões posteriores) actualmente suportam o Microsoft Visual Studio .NET.

Novidades nas APIs da Versão 8.1

O Enterprise Information Portal Versão 8 Edição 2 fornece um acesso sem precedentes a diferentes servidores de conteúdos. Os novos componentes e funções incluem:

- Capacidades de importação de XML:
Agora pode utilizar XML para importar e exportar conteúdos para o Content Manager através de DDOs e XDOs (utilizando APIs Java).
- Procedimentos de instalação melhorados
- Conectores adicionais para bases de dados relacionais:
O Enterprise Information Portal fornece conectores de bases de dados relacionais para DB2 UDB, DB2 DataJoiner, DB2 Data Warehouse Manager Information Catalog Manager e outras bases de dados através de controladores JDBC ou ODBC.
- extracção de informações e capacidades de pesquisa avançadas:
A Information Mining oferece pesquisa de texto avançada utilizando uma consulta flexível que pode restringir a documentos de determinadas categorias.
- Capacidades de fluxo de trabalho:
Utilizando a função fluxo de trabalho do Enterprise Information Portal, pode definir e executar o processo de fluxo de trabalho de um grupo de trabalho, departamento ou empresa.
- Controlo de acesso de nível associado:
Pode controlar o acesso ao Enterprise Information Portal extracção de informações; e aos processos de fluxo de trabalho através da utilização de conjuntos de privilégios e listas de controlo de acesso. O controlo de acesso adicional aos dados pode ser gerido pelas funções de controlo de acesso de cada servidor de conteúdos.
- Suporte adicional para o Content Manager:
 - Enumerar, adicionar, obter, actualizar e eliminar a classe de conteúdo
 - Obtenção assíncrona de conteúdo de objectos

Classes de Java novas e alteradas

As classes comuns de Java são utilizadas por todos os conectores. Este pacote contém interfaces (como dkDatastore) e classes abstractas (como dkAbstractDatastore e dkXDO), bem como classes concretas (como DKDDO) utilizadas pelos conectores.

Novas classes:

- dkAbstractAccessControlList
- dkAbstractAttrGroupDef
- dkAbstractAuthorizationMgmt
- dkAbstractConfigurationMgmt
- dkAbstractDatastoreAdmin
- dkAbstractDataObjectBase
- dkAbstractPrivilege
- dkAbstractPrivilegeGroup
- dkAbstractPrivilegeSet
- dkAbstractResultSetCursor
- dkAbstractUserDef
- dkAbstractUserMgmt
- dkAttrGroupDef
- dkAuthorizationMgmt
- dkCheckableObject
- DKChildCollection
- dkConfigurationMgmt
- DKLinkCollection
- dkPersistentCheckableObject
- dkPrivilege
- dkPrivilegeGroup
- dkUserDef
- dkUserGroupDef

Classes alteradas:

- dkAbstractDatastore
- dkAbstractDatastoreDef
- dkAbstractDatastoreExt
- dkAbstractEntityDef
- dkAccessControlList
- dkDataObjectBase
- dkDatastore
- dkDatastoreAdmin
- dkDatastoreDef
- dkDatastoreExt
- DKDDO
- dkEntityDef
- dkPersistentObject
- dkPrivilegeSet

- dkSearchTemplate
- dkSchemaMapping
- dkUserManagement

Alterações no comportamento de classes de Enterprise Information Portal Versão 8

Entre o EIP 7.1 e o EIP 8.2, algumas classes ou componentes de classe alteraram o seu comportamento. Esta secção descreve essas alterações. Para obter informações mais detalhadas, consulte *referência de API online*.

- dkIterator: next() avança o iterador para o artigo seguinte e obtém esse artigo e previous() regressa ao artigo anterior e obtém esse artigo.
- dkXDO: getPidObject() e setPidObject(DKPid pid)

Classes novas e alteradas de C++

O pacote comum de classes de C++ é utilizado pelos conectores. Este pacote contém interfaces (como dkDatastore) e classes abstractas (como dkAbstractDatastore e dkXDO), bem como classes concretas (como DKDDO) utilizadas pelos conectores.

Novas classes:

- dkAbstractAccessControlList
- dkAbstractAttrGroupDef
- dkAbstractAuthorizationMgmt
- dkAbstractConfigurationMgmt
- dkAbstractDataObjectBase
- dkAbstractDatastoreAdmin
- dkAbstractPrivilege
- dkAbstractPrivilegeGroup
- dkAbstractPrivilegeSet
- dkAbstractResultSetCursor
- dkAbstractUserDef
- dkAbstractUserMgmt
- dkAttrGroupDef
- dkAuthorizationMgmt
- dkCheckableObject
- DKChildCollection
- dkConfigurationMgmt
- DKLinkCollection
- dkPersistentCheckableObject
- dkPrivilege
- dkPrivilegeGroup
- dkUserDef
- dkUserGroupDef

Classes alteradas:

- dkAbstractDatastore
- dkAbstractDatastoreDef
- dkAbstractDatastoreExt

- dkAbstractEntityDef
- dkAccessControlList
- dkDataObjectBase
- dkDatastore
- dkDatastoreAdmin
- dkDatastoreDef
- dkDatastoreExt
- DKDDO
- dkEntityDef
- dkPersistentObject
- dkPrivilegeSet
- dkSchemaMapping
- dkSearchTemplate
- dkUserManagement

Alterações no comportamento de classe de Enterprise Information Portal 8.2

Entre o EIP 7.1 e o EIP 8.2, algumas classes ou componentes de classe alteraram o seu comportamento. Esta secção descreve essas alterações. Para obter informações mais detalhadas, consulte *referência de API online*.

- A utilização de DKString com DKAny em AIX foi alterada. Para obter uma DKString de uma DKAny, agora deve colocar DKAny numa variável DKAny e atribuí-la a uma variável DKString.

Exemplo: Este exemplo parte do princípio que o DDO foi obtido através de uma chamada de API de dkResultSetCursor fetchNext() de um servidor de conteúdos e que a variável do DDO foi definida, tal como é demonstrado de seguida. Este exemplo realiza ciclos nos artigos de dados do DDO para localizar os valores da cadeia.

```
DKDDO *ddo = NULL; DKAny any; DKString strTmp;

unsigned short data_id = 0;
unsigned short count = 0;

count = ddo->dataCount();
for (data_id = 1; data_id <= count; data_id++)
{
    any = ddo->getDataId(data_id);
    if (any.typeCode() == DKAny::tc_string)
    {
        strTmp = any.toString(); // This is how to get a DKString from a DKAny
    }
}
```

- As interfaces que implementam dkXDO sofreram alterações no que diz respeito aos métodos getPidObject e setPidObject.

Conceitos de programação de aplicações do Enterprise Information Portal

O Enterprise Information Portal facilita interfaces de programação de aplicações (APIs) direccionadas para objectos (OO), que pode utilizar para criar aplicações de consulta que acessem e apresentem dados relacionais, bem como dados de multimédia. Este capítulo fornece uma breve descrição geral de como estas APIs se adequam à arquitectura do Enterprise Information Portal e descreve os conceitos de programação orientados para objectos em que se baseiam as APIs.

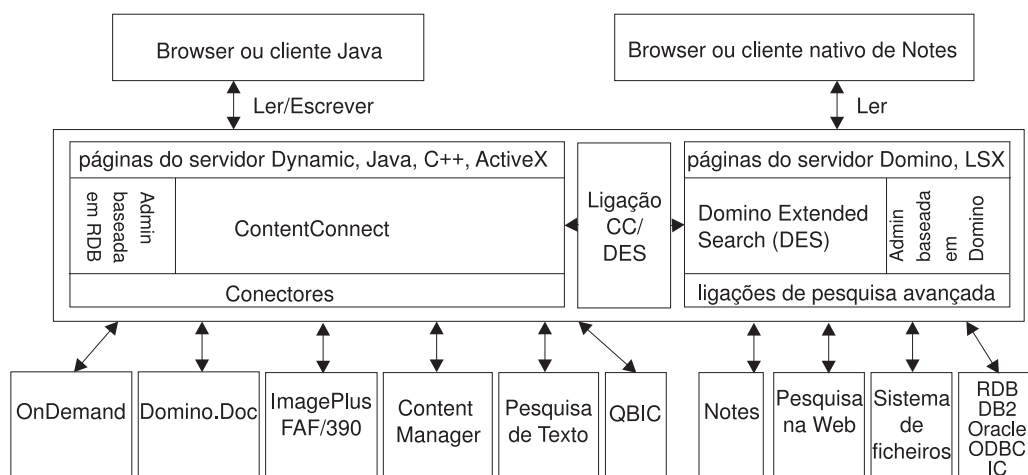


Figura 2. Organização do Enterprise Information Portal

Compreender acesso a dados através de servidores de conteúdos

Um servidor de conteúdos é um depósito de dados que é compatível com o protocolo DDO/XDO. Um servidor de conteúdos suporta sessões, ligações, transacções, cursores e consultas de utilizadores. As aplicações que utilizam as interfaces de programação de aplicações (APIs) e as bibliotecas de classes descritas neste manual, podem executar funções suportadas pelos servidores de conteúdos, tais como adicionar, obter, actualizar e eliminar DDOs. O Enterprise Information Portal suporta os seguintes servidores de conteúdos:

- Content Manager Versão 8 Edição 2
- Content Manager Versão 7 Edição 1
- Domino.Doc
- Pesquisa alargada
- ImagePlus para OS/390
- Content Manager OnDemand
- VisualInfo para AS/400
- DB2
- DB2 DataJoiner
- Catálogo de Informações
- DB2 Warehouse Manager Information Catalog Manager
- Servidores JDBC/ODBC

As aplicações que utilizam o Enterprise Information Portal podem criar um servidor de conteúdos associado que funciona como um servidor comum. As classes associadas ao Enterprise Information Portal permitem a pesquisa, obtenção e actualização associadas em vários servidores de conteúdos.

O servidor de conteúdos associado do Enterprise Information Portal e cada um dos servidores de conteúdo possuem esquemas diferentes. Ao integrar servidores de conteúdos múltiplos e heterogéneos num sistema associado exige conversão e definição de correspondências.

As funções de definição de correspondências do esquema fornecem as informações do esquema para cada servidor de conteúdos. As informações fornecidas pela definição de correspondências do esquema são utilizadas durante a pesquisa associada e a recolha associada, bem como a administração do sistema de Enterprise Information Portal. O Enterprise Information Portal mantém o esquema e as definições de correspondências, bem como as outras informações da administração na base de dados da administração.

Compreender conceitos de objectos de dados dinâmicos

Em coordenação com o CORBA Persistent Object Service e Object Query Service Specification de Object Management Groups (OMG), o Enterprise Information Portal fornece uma implementação do objecto de dados dinâmicos (DDO) e a sua extensão, o objecto de dados expandidos (XDO), que fazem parte dos protocolos de CORBA Persistent Data Service (PDS). Os conceitos de DDO e XDO não são específicos de um único servidor e podem ser utilizados para representar objectos de dados em qualquer sistema de gestão de bases de dados suportados pelo Enterprise Information Portal.

O objecto de dados dinâmico é uma interface utilizada para retirar e introduzir dados num servidor de conteúdos. Os DDOs existem na aplicação e não existem após a finalização de uma aplicação.

Objectos de dados dinâmicos (DDO)

O DDO é uma representação neutra a servidores de conteúdos dos dados permanentes de um objecto. O seu objectivo é conter todos os dados para um único objecto permanente. É também uma interface para obter ou carregar dados permanentes num servidor de conteúdos.

Um DDO tem um único ID permanente (PID), um tipo de objecto e um conjunto de artigos de dados cuja cardinalidade se chama contagem de dados. Cada artigo de dados tem um nome, valor, ID, uma ou mais propriedades de dados e contagem da propriedade de dados. Cada propriedade de dados pode ter um ID, um nome e um valor.

Por exemplo, um DDO pode representar uma fila de tabelas de base de dados, cujas colunas são representadas pelos artigos de dados do DDO e pelas suas propriedades. Um DDO pode conter um ou mais objectos de dados expandidos (XDOs) que representam tipos de dados não tradicionais. O Figura 3 na página 13 demonstra objectos de dados dinâmicos e artigos de dados.

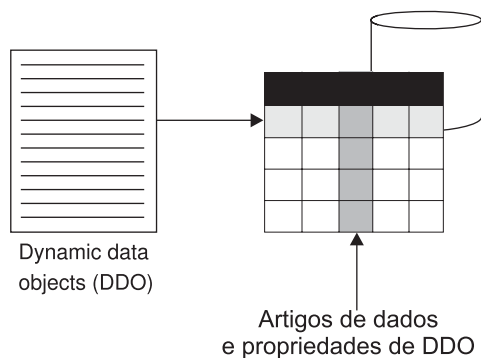


Figura 3. Objectos de dados dinâmicos e artigos

Objectos de dados expandidos (XDO)

Um XDO é uma representação de dados complexos de multimédia como, por exemplo, um artigo de recurso a armazenar uma imagem ou documento no Content Manager ou um novo tipo de dados introduzido pelos serviços de uma base de dados relacional como, por exemplo, IBM DB2 Extenders.

Os XDOs complementam os DDOs armazenando dados multimédia de tipos complexos e oferecendo funções que implementam os comportamentos dos tipos de dados na aplicação. Os XDOs podem estar contidos em, ou pertencer a um DDO de forma a representar um objecto de dados multimédia complexo.

Os XDOs têm um conjunto de propriedades para representar essas informações como tipos de dados e IDs. Os XDOs também podem ser objectos de dados dinâmicos autónomos. A Figura 4 demonstra um exemplo de XDOs.

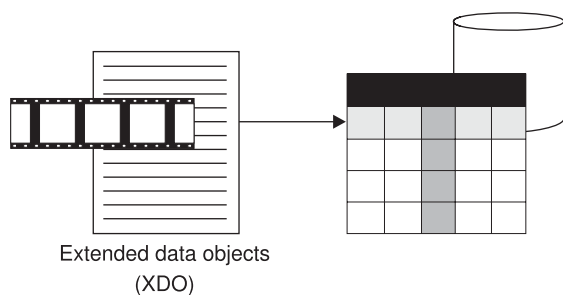


Figura 4. Objectos de dados expandidos (XDOs)

Representar conteúdo multimédia

Os DDOs e os XDOs podem representar objectos de dados de qualquer tipo e estrutura. Por exemplo, um filme pode ser representado por um DDO. Este DDO contém vários artigos de dados, que representam atributos do filme, como Nome_Realizador ou Titulo_Filme e XDOs multimédia, que representam os dados multimédia do filme, tais como vídeo clips ou galerias fotográficas.

No Content Manager 8.2, um DDO consiste em todos os metadados que descrevem o objecto como, por exemplo, um documento ou imagem. Um XDO expande a funcionalidade de DDO para suportar o conteúdo de recurso. O conteúdo de recurso é qualquer tipo de conteúdo, desde texto ou dados binários a sequências de vídeo e áudio. Um artigo que implementa (e deve ser) um XDO é também um

DDO e suporta toda a funcionalidade facultada por um DDO, bem como a funcionalidade adicional facultada por (e deve ser) um XDO.

Compreender servidores de conteúdo e DDOs

Os DDOs são criados edinamicamente associados a um servidor de conteúdos. A associação entre um DDO e um servidor de conteúdos é estabelecida com o PID dos DDOs.

Em geral, uma aplicação do Enterprise Information Portal executa os cinco passos enumerados de seguida, de modo a introduzir e retirar dados de um servidor de conteúdos.

1. Criar um servidor de conteúdos.
2. Estabelecer uma ligação ao servidor de conteúdos.
3. Criar os DDOs que seram utilizados e associar o servidor de conteúdos aos DDOs.
4. Adicionar, obter, actualizar e eliminar os DDOs através de métodos apropriados.
5. Fechar a ligação e destruir o servidor de conteúdos.

Comparar DDO/XDOs com valores de atributos e partes de artigos

Um DDO corresponde a um artigo no Enterprise Information Portal. O tipo de objecto do DDO corresponde ao tipo de artigo associado do artigo. Os artigos de dados de um DDO corresponde aos atributos de um artigo. Por exemplo, no Content Manager um tipo de artigo é criado utilizando um conjunto de atributos e um artigo é sempre indexado por um tipo de artigo.

Um DDO pode manter um ou mais XDOs que correspondam a partes de artigo no Enterprise Information Portal.

Compreender identificadores permanentes (PID)

O identificador permanente (PID) identifica exemplarmente um objecto permanente em qualquer servidor de conteúdos. O PID de um DDO é formado por um ID de artigo, um nome de servidor de conteúdos e outras informações relacionadas. Quando um DDO é adicionado a um servidor de conteúdos, o sistema atribui um PID único ao DDO.

Devido ao facto de um DDO ser uma interface dinâmica de dados permanentes que são introduzidos e retirados de servidores de conteúdos, diferentes DDOs podem representar as mesmas entidades de dados permanentes e, consequentemente, os DDOs podem ter o mesmo PID. Por exemplo, um DDO pode ser criado para introduzir uma entidade de dados num servidor de conteúdos para armazenar permanentemente dados e outro DDO pode ser criado para manter a mesma entidade de dados, em relação à qual é dada saída do mesmo servidor de conteúdos para ser modificada. Neste caso, estes dois DDOs partilham o mesmo valor de PID.

Trabalhar com um servidor de conteúdos associado e pesquisa associada

A *Pesquisa associada* é o processo de pesquisar dados num ou mais servidores de conteúdos. Utilize um objecto de DKDatastoreFed para a pesquisa associada. A pesquisa associada trabalha com classes que são implementações específicas de dkDatastore, dkDatastoreDef e outras classes relacionadas que suportam pesquisas associadas. As classes associadas específicas trabalham em conjunto com outras classes comuns, tais como consultas, recolhas e objectos de dados, fazendo parte da estrutura do Enterprise Information Portal.

As classes associadas funcionam em diferentes servidores de conteúdos como, por exemplo, Content Manager ImagePlus for OS/390 ou Domino.Doc. As classes fornecem um conjunto de funções genéricas para pesquisa associada e acesso através de servidores de conteúdos. Esta vista comum, denominada *modelo de documento associado*, é ilustrada na Figura 5.

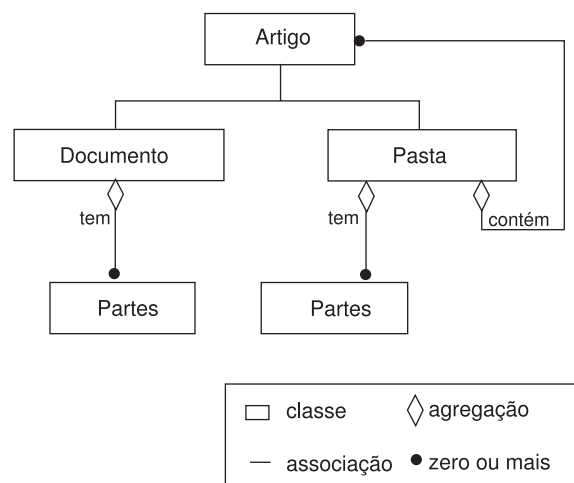


Figura 5. Vista de documento associado

Um artigo pode ser um documento ou uma pasta. Um documento pode conter zero ou mais partes. Uma pasta pode ter zero ou mais artigos que podem ser documentos ou outras pastas.

Nem todos os servidores de conteúdos podem suportar o modelo de documento associado. Por exemplo, uma base de dados DB2 não tem pastas ou partes. Um artigo define correspondências a uma linha numa DB2 ou noutra tabela de base de dados relacional e é utilizado caso um servidor de conteúdos não suporte documentos ou pastas.

Em geral, um documento é representado no seu programa através de um objecto de dados dinâmico (DDO), que é um objecto de dados que se descreve a ele mesmo, para transferir dados para o interior ou para o exterior do servidor. O próprio DDO tem uma estrutura geral e suporta uma variedade de modelos. Não está limitado ao modelo de documento associado. Esta flexibilidade permite a um DDO representar dados em servidores de conteúdos diferentes, cada um com o seu modelo de dados.

Uma *entidade* é um objecto de servidor de conteúdos que abrange atributos. Um *atributo* é um identificador utilizado para metadados em servidores de conteúdos. Por exemplo, os perfis, campos ou palavras-chave são atributos em servidores de conteúdo Domino.Doc.

Cada servidor de conteúdos possui a sua terminologia para explicar o modelo que suporta. Tabela 3 relaciona a terminologia utilizada para vários servidores de conteúdos no modelo associado:

Tabela 3. Definir correspondências de terminologia para cada servidor de conteúdos

| Servidor de conteúdos | Fonte de dados | Entidade | Atributo | Vista |
|---------------------------------|---|--|--|--|
| Content Manager 8.2 | Servidor de bibliotecas | tipo de artigo | atributo | vista de tipo de artigo ou subconjunto de tipo de artigo |
| Content Manager 7.1 | Servidor de bibliotecas | classe de índices | <ul style="list-style-type: none"> • atributo • atributo chave | vista de classe de índices |
| OnDemand | Servidor OnDemand | <ul style="list-style-type: none"> • grupo de aplicações • pasta | <ul style="list-style-type: none"> • campo • critério | N/A |
| ImagePlus | Servidor ImagePlus para OS/390 | entidade | atributo | N/A |
| Content Manager for AS/400 | Servidor Content Manager for AS/400 | classe de índices | atributo | vista de classe de índices |
| Domino.Doc | Servidor de Domino | biblioteca divisão armário processador de enlaces | perfil campo palavra-chave | N/A |
| Extended Search | Servidor de Extended Search | nome da base de dados | nome da base de dados | N/A |
| Base de dados relacional | IBM DB2 UDB, JDBC, ODBC, IBM DB2 DataJoiner | tabela | coluna | vista |
| Catálogo de Informações | DB2 Warehouse Manager Information Catalog Manager | classe de índices | propriedade | |
| Servidor de conteúdos associado | servidor de definição de correspondências | entidade associada com definição de correspondências | atributo associado com definição de correspondências | modelo de pesquisa |

Tabela 3. Definir correspondências de terminologia para cada servidor de conteúdos (continuação)

| Servidor de conteúdos | Fonte de dados | Entidade | Atributo | Vista |
|---|----------------|--------------------|--------------------|-----------------|
| Servidor de conteúdos associado que suporta pastas associadas | servidor | entidade associada | atributo associado | pasta associada |

O Figura 6 ilustra uma pesquisa associada. A pesquisa associada utiliza o servidor de conteúdos associado do Enterprise Information Portal working through modelos de pesquisa. O servidor de conteúdos associado chama, de seguida, as pesquisas para os servidores de conteúdo individuais, de modo a realizarem a pesquisa propriamente dita nos servidores de conteúdos. Esta associação é estabelecida através da definição de correspondências do esquema.

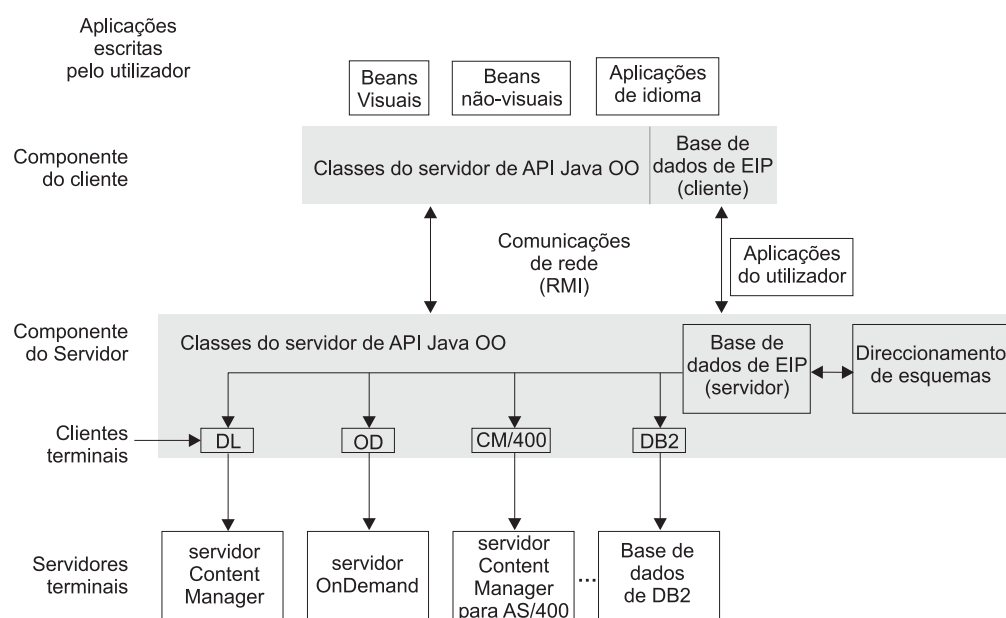


Figura 6. Estrutura de pesquisas associadas

O servidor de conteúdos associado pode utilizar os conectores local ou remoto para estabelecer ligação a servidores de conteúdos e pode utilizar RMI nesta comunicação. O utilizador pode desenvolver aplicações recorrendo a classes de API.

Definição de correspondências do esquema associado

Uma *definição de correspondências do esquema* representa uma definição de correspondências entre o esquema no servidor de conteúdos e a estrutura dos artigos que o utilizador pretende processar na aplicação. Um *esquema associado* constitui o esquema conceptual de um servidor de conteúdos associado de Enterprise Information Portal e define correspondências entre os conceitos no servidor de conteúdos associado e os conceitos em cada servidor de conteúdos participante. A definição de correspondências do esquema opera a diferença entre a forma dos dados serem armazenados fisicamente e a forma como o utilizador pretende processar os dados numa aplicação.

As informações de definição de correspondências estão representadas na memória das classes de definição de correspondências do esquema.

Utilizar componentes de definição de correspondências do servidor de conteúdos associado

Para além das informações relativas à definição de correspondências de esquema destinadas à definição de correspondência de entidades e atributos, um servidor de conteúdos associado deve igualmente possuir acesso às seguintes informações:

Definição de correspondências do ID de utilizador e palavra-passe

Para suportar uma única função de início de sessão, cada ID de utilizador no Enterprise Information Portal pode ter a correspondência definida para o ID de utilizador correspondente em cada servidor de conteúdos.

Registo do servidor de conteúdos

Cada servidor de conteúdos terá de ser registado de forma a poder ser localizado e ter a sessão iniciada pelo Enterprise Information Portal.

O ID de utilizador e as informações do servidor de conteúdos são mantidas na base de dados da administração do Enterprise Information Portal.

Executar consultas associadas

Para executar uma pesquisa associada utilizando APIs, comece por criar uma cadeia de consulta associada. Em seguida poderá criar e consultar a consulta de várias formas:

- Pode criar um objecto de consulta associada, `DKFederatedQuery`, passando-lhe a cadeia de consulta; em seguida invocar o método `executar` ou `avaliar` no objecto para processar a consulta.
- O utilizador pode transmitir a cadeia de consulta ao método `executar` ou `avaliar` do servidor de conteúdos associada de modo a processar directamente a consulta.

A cadeia de consulta é interpretada num formulário de consulta associada que constitui, essencialmente, uma representação neutra do servidor de conteúdos da consulta. O formulário da consulta associada é o input da pesquisa associada.

Se a consulta tiver origem numa aplicação baseada numa interface gráfica de utilizador (GUI), a consulta não tem de ser analisada e o formulário da consulta associada correspondente poder ser construído directamente.

À medida que é processada uma pesquisa associada, o Enterprise Information Portal executa os passos seguintes:

- Traduz o formulário canónico da consulta para várias consultas nativas que se executam em cada servidor de conteúdos. As informações de tradução são obtidas a partir da definição de correspondências do esquema.
- Converte entidades e atributos associados em entidades e atributos nativos para cada um dos servidores de conteúdos. Este processo utiliza os mecanismos de definição de correspondências e conversão descritos na definição de correspondências do esquema.
- Filtra apenas os dados relevantes durante a construção de consultas nativas.
- Formula consultas nativas e submete-as aos servidores de conteúdos individuais.

Cada servidor de conteúdos executa a consulta emitida. Os resultados são devolvidos numa consulta associada, que os pode processar da seguinte forma:

- Converte entidades e atributos nativos em entidades e atributos associados segundo as informações de definição de correspondências.
- Filtra os resultados para incluir apenas os dados solicitados.
- Intercala os resultados de vários servidores de conteúdos numa recolha associada.

O resultado de uma pesquisa associada é devolvido como uma recolha associada. O utilizador poderá criar um iterador para aceder a membros de recolha individual. Cada chamada ao método seguinte no iterador devolve um objecto DKDDO, que consiste num objecto de dados dinâmico neutral do servidor de conteúdos.

A recolha associada fornece a possibilidade de separar os resultados da consulta segundo o servidor de conteúdos. Criar um iterador sequencial ao invocar o método `creatememberIterator` na recolha associada. Ao utilizar este iterador sequencial, o utilizador poderá aceder a cada recolha de membro, que é um objecto de `DKResults`, e processá-la separadamente.

Os componentes de uma pesquisa associada e as suas relações está ilustrada em Figura 7.

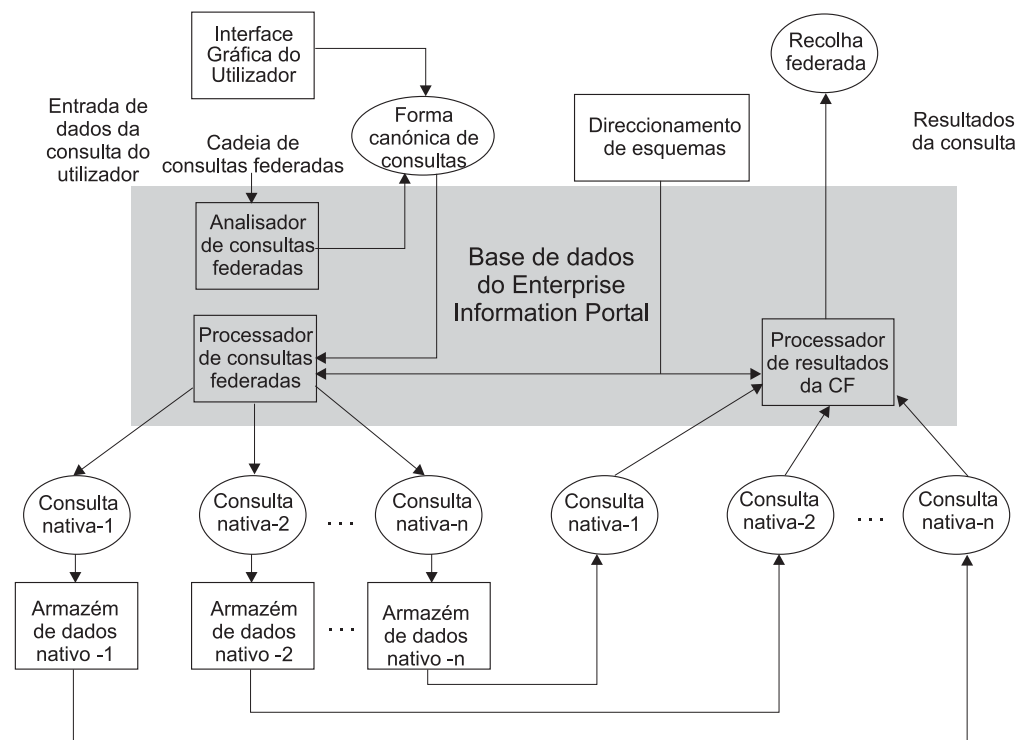


Figura 7. Processar consultas associadas

Sintaxe de consultas associadas

Quando criar uma consulta associada, terá de estar na sintaxe apropriada, como é demonstrado mais abaixo. O servidor de conteúdos associado não suporta consulta de imagens.

```

PARAMETRIC_SEARCH=([ENTITY=entity_name,]
                    [MAX_RESULTS=maximum_results,]
                    [COND=(conditional_expression)]
                    [; ...]
                    );
[OPTION=([CONTENT=yes_no_attronly]
        )]

[and

TEXT_SEARCH=(COND=(text_search_expression)
              );
[OPTION=([SEARCH_INDEX={search_index_name | (index_list) };]
        [ASSOCIATED_ENTITY={associated_entity_name});]
        [MAX_RESULTS=maximum_results;]
        [TIME_LIMIT=time_limit]
        )]
]

```

O operador NOT não é suportado em pesquisas associadas.

Exemplos de cadeias de consulta federadas

Consulta paramétrica associada a utilizar o operador LIKE

```
"PARAMETRIC_SEARCH = ( ENTITY = F_DGSAMP71, MAX_RESULTS = 5,
COND = (fName LIKE '%'))"
```

Consulta paramétrica associada a utilizar os operadores LIKE e >

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 5,
COND = (fJTitle LIKE 'Java%' AND fJNumPages > 20) )"
```

Consulta paramétrica associada a utilizar os operadores LIKE e <

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 5,
COND = (fJTitle LIKE 'Java%' AND fJNumPages < 20) )"
```

Consulta paramétrica associada a utilizar o operador BETWEEN

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJNumPages BETWEEN 5 200) )"
```

MAX_RESULTS devolve todos os resultados quando é definido como zero.

Consulta paramétrica associada a utilizar o operador NOTBETWEEN

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJNumPages NOTBETWEEN 5 100) )"
```

Consulta paramétrica associada a utilizar o operador IN

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJArticleTitle IN ('Java', 'Multi-Disk B-trees.',
'On Beyond Data.', 'IBM')) )"
```

Consulta paramétrica associada a utilizar o operador NOTIN

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJArticleTitle NOTIN ('Java', 'Multi-Disk B-trees.',
'On Beyond Data.', 'IBM')) )"
```

Consulta paramétrica associada a utilizar o operador ==

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJEditorName == 'Harth') )"
```

Consulta paramétrica associada a utilizar o operador <>

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJSectionTitle <> 'not available') )"
```

Consulta paramétrica associada a utilizar os operadores AND e OR

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = ((fJTtitle LIKE '%Java%') OR ((fJEditorName<NULL) AND
(fJArticleTitle LIKE 'Computer%')))) ); OPTION = (CONTENT = YES)"
```

Consulta paramétrica associada a utilizar o operador

CONTAINS_TEXT_IN_CONTENT

Este exemplo pesquisa texto no conteúdo. O conteúdo pode ser uma palavra ou uma expressão. Esta apenas é válida quando a entidade associada pesquisável por texto (FedTextResource) é correlacionada a um tipo de artigo pesquisável por texto do Content Manager Versão 8 ou uma entidade pesquisável por texto do Extended Search.

```
"PARAMETRIC_SEARCH = ( ENTITY = FedTextResource,MAX_RESULTS = 6,
COND = ( CONTAINS_TEXT_IN_CONTENT 'XML' ) ); OPTION =
( CONTENT = YES )"

```

Consulta paramétrica associada a utilizar o operador CONTAINS_TEXT

Pesquisa texto em valores de atributo. O texto pode ser uma palavra ou expressão. Esta apenas é válida quando o atributo associado pesquisável por texto (fJTtitle) é correlacionado a um atributo pesquisável por texto do Content Manager Versão 8 ou um atributo pesquisável por texto de Extended Search.

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal, MAX_RESULTS = 0,
COND = (fJTtitle CONTAINS_TEXT 'Java') )"

```

Consulta de texto associada

Pesquisa texto no conteúdo. O texto pode ser uma palavra ou expressão. A palavra-passe ASSOCIATED_ENTITY só se aplica quando uma entidade associada é pesquisável por texto. Esta apenas é válida quando a entidade associada pesquisável por texto (FedEntity) é correlacionada a um tipo de artigo pesquisável por texto do Content Manager Versão 8 ou uma entidade pesquisável por texto do Extended Search.

```
"TEXT_SEARCH = ( COND = ('XML') ); OPTION = ( ASSOCIATED_ENTITY=FedEntity )"

```

Consulta de texto associada

Pesquisa texto no conteúdo. O texto pode ser uma palavra ou uma expressão. O índice de texto associado, FedTMINDEX, é correlacionado a um índice de pesquisa de Text Miner do Content Manager Versão 7. A palavra-chave SEARCH_INDEX só é aplicável a este tipo de correlação. Para especificar uma palavra ou expressão na condição, o utilizador deve definir a cadeia de configuração como GENFEDTEXTQRY=YES, ao definir um servidor de Content Manager Versão 7 que suporta pesquisa de texto.

```
"TEXT_SEARCH = ( COND = ('sistema operativo') );
OPTION = ( SEARCH_INDEX = FedTMINDEX)"

```

Consulta de texto associada

Pesquisa texto no conteúdo em Content Manager Versão 7, Content Manager Versão 8 e Extended Search. O texto pode ser uma palavra ou uma expressão. O índice de texto associado, FedTMINDEX, é correlacionado a um índice de pesquisa de Text Miner do Content Manager Versão 7. A palavra-chave SEARCH_INDEX só é aplicável a este tipo de correlação. Para especificar uma palavra ou expressão na condição, o utilizador deve definir a cadeia de configuração como GENFEDTEXTQRY=YES, ao definir um servidor de Content Manager Versão 7 que suporta pesquisa de texto. A palavra-passe ASSOCIATED_ENTITY só se aplica quando uma entidade associada é pesquisável por texto. Esta apenas é válida quando a entidade associada pesquisável por texto (FedTextResource) é correlacionada a um tipo de artigo pesquisável por texto do Content Manager Versão 8 ou uma entidade pesquisável por texto do Extended Search.

```
"TEXT_SEARCH = ( COND = ( 'operating system' ) ); OPTION =
( SEARCH_INDEX = FedTMINDEX; ASSOCIATED_ENTITY = FedTextResource;
MAX_RESULTS = 5 )"
```

Consulta paramétrica associada a utilizar o operador OR

```
"PARAMETRIC_SEARCH = ( ENTITY = FedTextResource,
AX_RESULTS = 0,COND = (FedTextResourceJTitle LIKE '%test%'
) ) OR TEXT_SEARCH =( COND = ('UNIX') );
OPTION = ( ASSOCIATED_ENTITY =
FedTextResource; MAX_RESULTS = 4 )"
```

Consulta paramétrica associada a utilizar o operador AND

```
"PARAMETRIC_SEARCH = ( ENTITY = FedTextResource, COND =
(FedTextResourceJTitle LIKE '%test%') ) AND TEXT_SEARCH =
( COND = ('UNIX') ); OPTION = ( ASSOCIATED_ENTITY =
FedTextResource)"
```

Consulta paramétrica e consulta de texto em atributos a utilizar o operador OR

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal,
COND = (fJTitle LIKE 'Java%' OR fJArticleTitle
CONTAINS_TEXT 'Database') );OPTION = ( CONTENT = ATTRONLY )"
```

Consulta paramétrica e consulta de texto em atributos a utilizar o operador OR

```
"PARAMETRIC_SEARCH = ( ENTITY = F_ICMNLSDb_Journal,
COND = (fJTitle LIKE 'Java%' OR fJArticleTitle
CONTAINS_TEXT 'Database') );OPTION = ( CONTENT = YES )"
```

Armazenar resultados da consulta em pastas associadas (Apenas para Java)

O Enterprise Information Portal Versão 8 Edição 2 actualmente faculta entidades associadas especiais que suportam *pastas associadas*. Estas pastas associadas podem armazenar os resultados combinados de uma consulta associada como, por exemplo, um documento do Content Manager e um documento relacionado de OnDemand. O utilizador pode enviar os resultados directamente para um fluxo de trabalho.

O EIP armazena as pastas como DDOs, que poderá adicionar a um conjunto de DKFolder. Pode também armazenar as pastas como XDOs num conjunto de DKParts.

A entidade associada especial utiliza tabelas adicionais para deter as pastas. Qualquer outra funcionalidade (como, por exemplo, as consultas, APIs e atributos) tem comportamentos idênticos a uma entidade associada normal. A entidade especial apenas armazena PIDs de DDO nas pastas.

Se optar por não definir correspondências de uma entidade associada especial, a consulta associada apenas pesquisa as entidades associadas especiais. Se optar por definir correspondências entre uma entidade associada especial e outras entidades nativas, a consulta associada também pesquisa essas entidades.

Para obter os exemplos de código, consulte o directório CMBROOT\dk\samples.

Trabalhar com a administração do sistema

O Enterprise Information Portal faculta as classes e APIs para que o utilizador possa aceder às funções de administração do sistema. Consulte *referência de API online* para obter informações relativas às classes específicas.

Personalizar o cliente de administração do sistema de EIP

O cliente de administração do sistema de EIP suporta a expansão da aplicação de administração do sistema para incluir as funções de personalização:

- O utilizador pode substituir os diálogos de utilizador e de grupo de utilizadores no cliente de administração do sistema de EIP por diálogos personalizados.
- Pode adicionar novos nós à hierarquia no cliente de administração do sistema de EIP.
- Pode adicionar novos artigos de menu ao menu Ferramentas no cliente de administração do sistema.

Pode chamar saídas de utilizador antes e depois de iniciar uma sessão na administração do sistema de EIP.

Programar com as interfaces de programação de aplicações (APIs)

As interfaces de programação de aplicações (APIs) são um conjunto de classes que acedem e manipulam dados locais ou remotos. Esta secção descreve as APIs, a implementação de várias funções de pesquisa e a conectividade da Internet.

As APIs suportam:

- Um modelo de objectos comum para o acesso a dados.
- Pesquisas e actualizações múltiplas numa combinação heterogénea de servidores de conteúdo.
- Um mecanismo flexível para utilizar uma combinação de motores de pesquisa; por exemplo, a função de pesquisa de texto do Content Manager.
- Capacidade de fluxo de trabalho.
- Funções de administração.
- Apenas para **Java**: Implementação cliente/servidor para a aplicação Java.

O suporte de sequência múltipla para as APIs de Java está totalmente activado apenas para servidores de Windows. Os servidores e clientes de AIX e os clientes de Windows não suportam sequência múltipla.

Compreender as diferenças entre as APIs de Java e C++

A lista que se segue descreve as diferenças entre os conjuntos de API de IBM Enterprise Information Portal for Multiplatforms Java e de C++:

- Os operadores definidos na API de C++ não estão definidos na API de Java. São suportados como funções de Java.
- O objecto de classe (java.lang.Object) Java é utilizado em substituição da classe DKAny de C++ para representar um objecto genérico.
- As constantes comuns e globais são definidas na interface DKConstant na API de Java; em C++ são definidas em DKConstant.h.
- As APIs de Java utilizam o colector de dados caducados de Java.
- As funções DKDDO.toXML() e DKDDO.fromXML() de Java não estão disponíveis em C++.

Compreender a arquitectura cliente/servidor (Apenas para Java)

As APIs facultam uma interface de programação conveniente para programadores de aplicações. As APIs podem residir no servidor ou no cliente de EIP (ambos facultam a mesma interface). A API cliente comunica com o servidor para aceder a dados através da rede, por intermédio de RMI (Remote Method Invocation) de Java. A comunicação entre o cliente e o servidor é executada pelas classes; não é necessário adicionar quaisquer programas adicionais.

As classes de API são formadas pelos seguintes pacotes: servidor, cliente, cs e comum. As classes de servidor e de cliente fornecem as mesmas APIs, mas têm implementações diferentes.

- O pacote do servidor é com.ibm.mm.sdk.server. As classes do pacote do servidor comunicam directamente com o servidor de conteúdos associado ou de fundo.

- O pacote do cliente é `com.ibm.mm.sdk.client`. As classes do pacote do cliente comunicam com as classes do pacote do servidor por intermédio de RMI.
- As classes comuns são partilhadas pelo cliente e pelo servidor. Por vezes uma aplicação não sabe onde reside o conteúdo. Por exemplo, uma aplicação pode ter conteúdo que reside no cliente numa dada altura e no servidor noutra altura. O pacote `cs` liga dinamicamente o cliente e o servidor.

A aplicação cliente deve importar o pacote cliente, a aplicação dinâmica deve importar o pacote `cs` e a aplicação do servidor deve importar o pacote do servidor.

Apesar de se fornecer a mesma API para cliente e servidor, o pacote cliente tem um artigo de excepção adicional, visto que comunica com o pacote do servidor. No entanto, tenha em atenção que a interface cliente/servidor não é suportada por todos os conectores. Por exemplo, esta não é suportada pelo CM V8.

Criação de pacotes para o ambiente Java

As APIs de Enterprise Information Portal estão contidas em quatro pacotes como partes de `com.ibm.mm.sdk`: `comum`, `servidor`, `cliente` e `cs`.

servidor (`com.ibm.mm.sdk.server`)

Aceder e manipular informações do servidor de conteúdos

cliente (`com.ibm.mm.sdk.client`)

Comunicar com o pacote do servidor através do Remote Method Invocation (RMI)

comum (`com.ibm.mm.sdk.common`)

Classes comuns para o pacote de servidor, pacote de cliente e pacote de `cs`.

cs (`com.ibm.mm.sdk.cs`)

Ligar o cliente ou servidor de forma dinâmica

A sua aplicação tem de utilizar o `comum` com o pacote do servidor para aplicações dinâmicas, o pacote do `cliente` para aplicações que acedem ao servidor remoto ou o pacote `cs`.

Sugestões de programação

Não importe pacotes de cliente e de servidor no mesmo programa. Se estiver a desenvolver uma aplicação cliente, importe o pacote cliente. Caso contrário, importe o pacote do servidor. Se não souber onde reside o conteúdo, utilize o pacote `cs` (com os pacotes do servidor ou do cliente). A importação de vários pacotes pode provocar erros de compilação.

Alguns conectores contêm chamadas ao código C da implementação. No caso destes conectores, utilize o pacote cliente para aplicações da Web que necessitam de interfaces puras de Java. O pacote cliente é criado com programas puros de Java; o pacote do servidor pode incluir chamadas de JNI.

Visto que um cliente exige a excepção `java.rmi.RemoteException`, anexe sempre esta excepção à aplicação, quer seja executada num servidor ou num cliente.

Configurar o ambiente de Java (Apenas para Java)

Ao configurar o seu ambiente de Windows, AIX ou Solaris, deverá ter importado os seguintes pacotes:

pacote do servidor

Importe quando um servidor de conteúdos e uma aplicação estão do lado do servidor

- com.ibm.mm.sdk.common
- com.ibm.mm.sdk.server

pacote do cliente

Importe quando um servidor de conteúdos e uma aplicação estão do lado do cliente.

- com.ibm.mm.sdk.common
- com.ibm.mm.sdk.client

pacote cs

Importe quando a localização de um servidor de conteúdos é diferente da localização da aplicação.

- com.ibm.mm.sdk.common
- com.ibm.mm.sdk.cs

Definir as variáveis de ambiente de Java para Windows

Pode abrir uma linha de comandos de Windows com o ambiente configurado para desenvolver aplicações de Enterprise Information Portal seleccionando **Iniciar → Programas → IBM Enterprise Information Portal for Multiplatforms 8.2 → Janela de Desenvolvimento**. Em alternativa, poderá executar cmbenv81.bat numa linha de comandos de Windows para configurar o ambiente.

Caso pretenda modificar as variáveis de ambiente, altere o seguinte:

PATH Certifique-se de que o seu PATH contém *X:\CMBROOT\DLL*; onde *X* é a unidade em que instalou o Enterprise Information Portal.

CLASSPATH

Certifique-se de que o seu CLASSPATH contém *X:\CMBROOT\LIB\xxx*, onde *X* é a unidade em que instalou o Enterprise Information Portal e *xxx* são os ficheiros .jar (por exemplo, cmbfed81.jar).

Definir as variáveis de ambiente de Java para AIX

No ambiente de AIX, pode utilizar um procedimento de base, cmbenv81.sh, para configurar o ambiente de desenvolvimento de aplicações de EIP.

Se não utilizar o procedimento, deve definir as seguintes variáveis de ambiente:

PATH Certifique-se de que o seu PATH contém */usr/lpp/cmb/lib*

LIBPATH

Certifique-se de que o seu LIBPATH contém */usr/lpp/cmb/lib*

LD_LIBRARY_PATH

Certifique-se de que o seu LD_LIBRARY_PATH contém */usr/lpp/cmb/lib*

CLASSPATH

Certifique-se de que o seu CLASSPATH contém */usr/lpp/cmb/lib/xxx*, em que *xxx* corresponde aos ficheiros .jar (por exemplo, cmbfed81.jar)

Utilize a opção do compilador *-qalign=packed* de forma a que os objectos se possam alinhar devidamente.

Definir as variáveis de ambiente de Java para Solaris

No ambiente Solaris, pode utilizar um shell script, `cmbenv81.sh`, para configurar o ambiente de desenvolvimento para desenvolver aplicações do EIP.

Se não utilizar o procedimento, deve definir as seguintes variáveis de ambiente:

PATH Certifique-se de que o seu PATH contém `/opt/cmb/lib`

LIBPATH

Certifique-se de que o seu LIBPATH contém `/opt/cmb/lib`

LD_LIBRARY_PATH

Certifique-se de que o seu LD_LIBRARY_PATH contém `/opt/cmb/lib`

CLASSPATH

Certifique-se de que o seu CLASSPATH contém `/opt/cmb/lib/xxx`, em que xxx corresponde aos ficheiros `.jar` (por exemplo, `cmbfed81.jar`)

Utilize a opção do compilador `-qalign=packed` de forma a que os objectos se possam alinhar devidamente.

Utilizar Remote Method Invocation (RMI) com servidores de conteúdos

Visto que as classes de clientes nas APIs de Java têm de comunicar com as classes de servidor para aceder a dados através da rede, tanto o servidor como o cliente têm de estar preparados para a execução cliente/servidor. Na máquina servidor, o servidor de RMI tem de estar em execução para receber o pedido do cliente que utiliza um número de porta especificado. O programa cliente exige o nome do servidor e o número da porta. Para realizar comunicações entre cliente e servidor, o cliente deve saber qual o número da porta do servidor ao qual necessita estabelecer uma ligação.

Um servidor de RMI pode estabelecer ligação a um número infinito (apenas limitado pelos recursos do sistema) de servidores de conteúdo, mas cada servidor deve estar ligado a, pelo menos, um servidor de conteúdos. Um servidor de RMI principal pode remeter para outros servidores de RMI no conjunto de servidores. Quando um cliente de RMI pesquisa pela primeira vez um servidor de conteúdos, este inicia um servidor de RMI. Se o servidor de conteúdos não for encontrado, os servidores de conjunto de RMI são pesquisados de seguida.

Se o mesmo cliente de RMI pesquisar novamente o servidor de conteúdos, o cliente pesquisa o servidor de RMI no qual encontrou pela primeira vez o servidor de conteúdos.

Para iniciar o servidor de RMI, utilize o `cmbregist81.bat` em Windows ou `cmbregist81.sh` em AIX ou Solaris. Antes de iniciar o servidor de RMI, defina o número de porta correcto e o tipo de servidor. Para obter informações relativas à configuração e administração de servidores de RMI, consulte *Planear e Instalar o Enterprise Information Portal* e *Gerir o Enterprise Information Portal*.

Configurar o ambiente de C++ (Apenas para C++)

Ao configurar o ambiente do Windows ou o ambiente de AIX, tem de estabelecer as definições descritas nesta secção. A Tabela 4 na página 29 enumera os requisitos de Biblioteca, AIX shared e DLL.

Requisito: Para utilizar C++, deve instalar o Suporte de Cliente e Assistente de Configuração de Cliente de DB2 em todos os servidores remotos que acedem à base de dados do Enterprise Information Portal. O ID de utilizador e palavra-passe utilizados para estabelecer uma ligação à base de dados devem ser iguais ao ID de utilizador e palavra-passe que utiliza na base de dados do Enterprise Information Portal. Para obter mais detalhes, consulte *Gerir o Enterprise Information Portal*.

Attention: cmbcm81x.lib is for release build and cmbcm81xd.lib is for debug build, where *x* represents either Version 6 or 7 (.Net) of the Microsoft Visual C++ compiler.

Tabela 4. Informações relativas a objectos partilhados e ambiente de DLL

| Conector | Biblioteca | DLLs do Windows | Objectos partilhados em AIX |
|--|-----------------------------------|--|--------------------------------------|
| Nota Comum: Isto não é um conector. É um conjunto comum de APIs utilizado por todos os conectores. | cmbcm81x.lib cmbcm81xd.lib | cmbcm81x.dll cmbcm81xd.dll | libcmbcm815.a |
| Content Manager Versão 8 | cmbicm81x.lib cmbicm81xd.lib | cmbicm81x.dll cmbicm81xd.dll cmbicmfac81x.dll cmbicmfac81xd.dll | libcmbicm815.a libcmbicmfac815.so |
| Content Manager anterior | cmbdl81x.lib cmbdl81xd.lib | cmbdl81x.dll cmbdl81xd.dll cmbdlfac81x.dll cmbdlfac81xd.dll de_db2.dll de_db2_d.dll de_ora.dll de_ora_d.dll | libcmbdl815.a libcmbdlfac815.so |
| Associado | cmbfed81x.lib cmbfed81xd.lib | cmbfed81x.dll cmbfed81xd.dll cmbfedfac81x.dll cmbfedfac81xd.dll | libcmbfed815.a libcmbfedfac815.so |
| DB2 Universal Database Versão 8.1 | cmbdb281x.lib cmbdb281xd.lib | cmbdb281x.dll cmbdb281xd.dll cmbdb2fac81x.dll cmbdb2fac81xd.dll | libcmbdb2815.a libcmbdb2fac815.so |
| ODBC | cmbodbc81x.lib cmbodbc81xd.lib | cmbodbc81x.dll cmbodbc81xd.dll cmbodbcfac81x.dll cmbodbcfac81xd.dll | Não suportado |
| OnDemand | cmbod81x.lib cmbod81xd.lib | cmbod81x.dll cmbod816xd.dll cmbodfac81x.dll cmbodfac81xd.dll | libcmbod815.a libcmbodfac815.so |
| ImagePlus for OS/390 | cmbip81x.lib cmbip81xd.lib | cmbip81x.dll cmbip81xd.dll cmbipfac81x.dll cmbipfac81xd.dll | Não suportado |
| VisualInfo | cmbv481x.lib cmbv481xd.lib | cmbv481x.dll cmbv481xd.dll cmbv4fac81x.dll cmbv4fac81xd.dll | Não suportado |

Tabela 4. Informações relativas a objectos partilhados e ambiente de DLL (continuação)

| Conector | Biblioteca | DLLs do Windows | Objectos partilhados em AIX |
|------------------------|---------------------------------|--|-----------------------------------|
| Domino.Doc | cmbdd81x.lib cmbdd81xd.lib | cmbdd81x.dll cmbdd81xd.dll cmbddf81x.dll cmbddf81xd.dll | Não suportado |
| Domino Extended Search | cmbdes81x.lib cmbdes81xd.lib | cmbdes81x.dll cmbdes81xd.dll cmbdes81x.dll cmbdes81xd.dll | libcmbdes815.a libcmbdes815.so |

Definir as variáveis de ambiente de C++ em Windows

Pode abrir uma linha de comandos de DOS com o ambiente configurado para desenvolver aplicações de Enterprise Information Portal seleccionando **Iniciar → Programas → IBM Enterprise Information Portal for Multiplatforms 8.1 → Janela de Desenvolvimento**. Em alternativa, poderá executar CMBenv81.bat numa linha de comandos de DOS para configurar o ambiente.

Caso pretenda modificar as variáveis de ambiente, altere o seguinte:

PATH

definir PATH=x:\CMBROOT\DLL

Em que x é a unidade na qual o EIP está instalado.

INCLUDE

definir INCLUDE=x:\CMBROOT\INCLUDE

em que x é a unidade na qual o EIP está instalado.

Definir as variáveis de ambiente de C++ em AIX

Consulte os ficheiros MAK exemplo no directório de exemplos para obter mais informações.

Defina as seguintes variáveis de ambiente:

No ambiente AIX, pode utilizar um dos três ficheiros batch para configurar o ambiente de programação.

1. Para uma interface Bourne, utilize cmbenv81.sh
2. Para uma interface C, utilize cmbenv81.csh
3. Para uma interface Korn, utilize cmbenv81.ksh

Defina as seguintes variáveis de ambiente:

caminho de NLS

exportar NLSPATH=\${NLSPATH}:/usr/lpp/cmb/msg/En_US/%N

PATH

exportar PATH=/usr/lpp/cmb/lib

LIBPATH

exportar LIBPATH=/usr/lpp/cmb/lib

INCLUDE

exportar INCLUDE=/usr/lpp/cmb/INCLUDE

Construir programas C++

Siga os procedimentos relativos ao seu compilador e ambiente de desenvolvimento para criar os ficheiros MAK e construir a sua aplicação.

Ao conceber uma aplicação através das APIs de C++ que irá utilizar para depuração, ligue a aplicação à versão de depuração das bibliotecas da API, ou seja, as bibliotecas ***d.lib**. Ao conceber a aplicação da produção final, ligue-a às bibliotecas que não sejam de depuração (***.lib**).

Trabalhar com Microsoft Visual Studio .NET

As APIs do Enterprise Information Portal e do Content Manager versão 8.2 e versões posteriores suportam o Microsoft Visual Studio .NET. No entanto, ao utilizar o Microsoft Visual Studio .NET para construir as suas aplicações, deve estabelecer ligação à biblioteca adequada, a qual é determinada pelo conector e pela versão de Visual Studio C++ que está a utilizar na altura da compilação.

Os ficheiros MAK exemplo (que suportam tanto o Visual Studio Versão 6 como o Visual Studio .NET) são facultados com os exemplos de API de ICM.

Para determinar a biblioteca à qual necessita de estabelecer ligação, na altura da compilação, utilize o formato *nome do conector*816.lib, se usar o Microsoft Visual Studio C++ 6 e *nome do conector*817.lib se usar o Microsoft Visual Studio .NET, tal como é demonstrado nas tabelas seguintes.

Tabela 5. Nomes exemplo de bibliotecas de Microsoft Visual Studio C++ 6

| Conector | Biblioteca |
|---------------------------------|---------------|
| Comum | cmbcm816.lib |
| Content Manager 8.1 e posterior | cmbicm816.lib |
| Associado | cmbfed816.lib |

Tabela 6. Nomes de bibliotecas de Microsoft Visual Studio .NET

| Conector | Biblioteca |
|---------------------------------|---------------|
| Comum | cmbcm817.lib |
| Content Manager 8.1 e posterior | cmbicm817.lib |
| Associado | cmbfed817.lib |

Definir o subsistema da consola para conversão de página de códigos em Windows

C++

```
#include <DKConstant.h>
#include <DKEnvironment.hpp>

void main(int argc, char *argv[])
{
    // definir o sub-sistema para a consola no início de programa
    // isto irá converter as mensagens de erro devolvidas da página de códigos
    // por DKEExceptions do Windows Graphical
    // Interface do Utilizador (formato ANSI) para a Consola (formato OEM)
    // Se não estiver especificada, a predefinição é DK_SS_WINDOWS
    DKEnvironment::setSubSystem(DK_SS_CONSOLE);
    ...
}
```

Compreender várias opções de pesquisa

As pesquisas podem ser executadas com base em quase todas as partes de um artigo ou componente, texto dentro de um artigo ou componente, ou texto dentro dos conteúdos de recursos.

O Content Manager Versão 8 faculta uma função de texto integrado, a qual já não necessita de uma função de pesquisa de texto diferente (no caso do Content Manager ainda necessita). Consulte “Compreender a pesquisa de texto” na página 189.

Utilize as várias opções de pesquisa para pesquisar num servidor de conteúdos, utilizando uma consulta suportada ou uma combinação de várias, enumeradas de seguida, ou pesquise nos resultados de uma pesquisa anterior. Cada tipo de pesquisa é suportado por um ou mais motores de pesquisa. Nem todos os servidores de conteúdos suportam várias opções de pesquisa.

Pesquisa paramétrica

Pesquisa texto como, por exemplo, propriedades de artigos e componentes, atributos, referências, ligações e conteúdos de pastas. Por exemplo, recorre a uma consulta paramétrica para pesquisar documentos utilizando o nome de um cliente. A consulta necessita de uma correspondência exacta na

condição especificada no predicado da consulta e dos valores de dados armazenados no servidor de conteúdos.

Pesquisa de texto

Pesquisa todo o texto marcado como `textSearchable(true)` utilizando um motor de Pesquisa de Texto como, por exemplo, o DB2 Net Search Engine (NSE). A consulta baseia-se no conteúdo de campos de texto para efectuar uma correspondência aproximada à expressão de pesquisa de texto especificada; por exemplo, a existência (ou inexistência) de determinadas expressões ou raízes de palavras.

Nota: O DB2 Net Search Engine (NSE) foi denominado DB2 Text Information Extender (TIE) em versões anteriores de DB2.

Pesquisa de imagens

Pesquisa características dentro de imagens. A consulta baseia-se no conteúdo de imagens para efectuar uma correspondência aproximada à expressão de pesquisa de imagens especificada; por exemplo, a presença de uma determinada cor nas imagens.

Pesquisa combinada

Pesquisa utilizando a pesquisa paramétrica e a pesquisa de texto.

Apenas para Content Manager: O CM possui um motor de pesquisa e três opções relativas ao modo e altura em que as pesquisas devem ser executadas (e os resultados devem ser devolvidos): Executar, Avaliar e Executar com Repetição de Marcação. Consulte o exemplo `SSearchICM` para obter a tabela e as explicações.

Rastreio

Para tratar problemas que surgem nas aplicações de API do utilizador, pode utilizar o rastreio e o tratamento de excepções.

Rastrear consultas de texto utilizando o Motor de Pesquisa de Texto

O Motor de Pesquisa de Texto (TSE) e todas as suas funções só podem ser utilizados com o Content Manager anterior. O Content Manager Versão 8 faculta uma função de texto integrada, a qual não necessita de uma função de pesquisa de texto diferente. Consulte “Compreender a pesquisa de texto” na página 189.

As seguintes definições de variáveis de ambiente escrevem o rastreio da consulta de Motor de Pesquisa de Texto, em formato binário, num ficheiro especificado:

`CMBTMDSTREAMTRACE=nomeFicheiro`

(por exemplo, `.\tm.out` para Windows ou `./tm.out` para AIX)

As seguintes definições de variáveis de ambiente escrevem o rastreio para as chamadas da API de Motor de Pesquisa de Texto utilizadas durante uma consulta de texto para um ficheiro especificado:

`CMBTMTRACE=nomeFicheiro`

A seguinte definição de ambiente grava os termos de pesquisa de texto num ficheiro especificado: `CMBMTMTERM=fileNome` (por exemplo, `.\tmterm.out`)

Nota: O Content Manager Versão 8 utiliza uma pesquisa de texto integrado. Se estiver a utilizar o Content Manager anterior, ainda pode utilizar o TSE.

Rastrear consultas paramétricas

Em versões anteriores do Content Manager que utilizam o Text Search Engine, utilize a seguinte definição de variável de ambiente para escrever a consulta paramétrica transmitida ao gestor de pastas:

`CMBDLQRYTRACE=nomeFicheiro`

(por exemplo, `<.\dlqry.out>` para Windows ou `<./dlqry.out>` para AIX)

Tratar exceções

Quando as APIs detectam um problema, estas devolvem uma excepção. Enviar uma excepção cria um objecto de excepção da classe de `DKException` ou numa das suas subclasses.

Quando é criada `DKException`, o nível do conector regista informações de diagnóstico num ficheiro de registo, partindo do princípio que é utilizada a configuração de registo. Consulte *Mensagens e Códigos* para obter mais informações relativas aos ficheiros de registo e configuração utilizados pelas APIs de EIP.

Quando é capturada uma excepção de `DKException`, esta irá permitir-lhe que veja todas as mensagens de erro, códigos de erro e estados de erro que ocorrem durante a execução. Quando um erro é capturado, é emitida uma mensagem de erro juntamente com a localização do surgimento da excepção. Também são facultadas informações adicionais como, por exemplo, de ID de erro. O código seguinte apresenta um exemplo do processo de libertação e obtenção em EIP e CM:

```
Java
try{
    ... Operações de API de EIP ...
}
catch (DKException exc) {
    // NOTE: Print Function Provided in SConnectDisconnectICM API Sample.
    System.out.println("");
    System.out.println("XXXXXXXXXXXXXXXXXXXXXXXXXXXX");
    System.out.println("X    !!! Exception !!!    X");
    System.out.println("XXXXXXXXXXXXXXXXXXXXXXXXXXXX");
    System.out.println("    Name: " + exc.name());
    System.out.println("    Message: " + exc.getMessage());
    System.out.println(" Message ID: " + exc.getErrorId());
    System.out.println("Error State: " + exc.errorState());
    System.out.println(" Error Code: " + exc.errorCode());
    exc.printStackTrace();
    System.out.println("-----");
} catch (Exception exc) {
    // NOTE: Print Function Provided in SConnectDisconnectICM API Sample.
    System.out.println("");
    System.out.println("XXXXXXXXXXXXXXXXXXXXXXXXXXXX");
    System.out.println("X    !!! Exception !!!    X");
    System.out.println("XXXXXXXXXXXXXXXXXXXXXXXXXXXX");
    System.out.println("    Name: " + exc.getClass().getName());
    System.out.println(" Message: " + exc.getMessage());
    exc.printStackTrace();
    System.out.println("-----");
}
```

C++

```
try{
    ... Operações de API de EIP ...
}
catch(DKException &exc) {
    // NOTE: Print Function Provided in SConnectDisconnectICM API Sample.
    cout << endl;
    cout << "XXXXXXXXXXXXXXXXXXXXXXXXXXXX" << endl;
    cout << "X      !!! Exception !!!      X" << endl;
    cout << "XXXXXXXXXXXXXXXXXXXXXXXXXXXX" << endl;
    cout << "      Name: " << exc.name() << endl;
    cout << " Message ID: " << exc.errorId() << endl;
    cout << "Error State: " << exc.errorState() << endl;
    cout << " Error Code: " << exc.errorCode() << endl;
    // Print API Location(s) Detecting Error
    for(unsigned int j=0; j < exc.locationCount(); j++){ //Print all locations
        const DKExceptionLocation* p = exc.locationAtIndex(j);
        cout << " Location " << j << ": " << p->fileName() << " :: "
            << p->functionName() << '[' << p->lineNumber() << ']' << endl;
    }
    if(exc.textCount()<=0) // Write statement if no locations.
        cout << " Locations: <none> " << endl;
    // Error Message(s)
    for(unsigned int i=0; i < exc.textCount(); i++) // Print All Messages
        cout << " Message " << i << ": " << exc.text(i) << endl;
    if(exc.textCount()<=0) // Notify user if no messages.
        cout << " Messages: <none> " << endl;
    cout << "-----" << endl;
}
}
```

Para obter mais informações relativas à detecção e tratamento de erros, consulte o Exemplo de Instrução de API SConnectDisconnectICM disponível em CMBROOT\Samples\java\icm ou CMBROOT\Samples\cpp\icm.

Constantes

As constantes especificadas encontram-se na forma de DK_CM_ (Constantes comuns) ou DK_XX_ (em que XX indica diferentes servidores de conteúdo). Consulte Tabela 7 na página 36 para obter uma lista das extensões (extensões anexadas a cada DKDatastore).

Ao especificar constantes de DDO, utilize DK_CM_DATAITEM_TYPE_ ... (por exemplo, DK_CM_DATAITEM_TYPE_STRING) para tipos de propriedades. Para tipos de atributos utilize as constantes DK_CM_...*tipo* (por exemplo, DK_CM_INTEGER).

Java

As constantes comuns estão definidas em DKConstant.java. Também pode rever uma versão de texto destas constantes em DKConstant.txt. As constantes são servidores de conteúdo específicos em DKConstantXX.java; por exemplo, as constantes exclusivas de Content Manager encontram-se em DKConstantICM.java.

C++

As constantes comuns estão definidas em `DKConstant.h`. Para obter uma lista de constantes e dos valores correspondentes, visualize `DKConstant2.h` (mas não o inclua no seu programa). As constantes de servidores de conteúdos específicos são definidas em ficheiros de cabeçalho de formato `DKConstantXX.h`; por exemplo, as constantes exclusivas de Content Manager encontram-se em `DKConstantICM.h`.

Ligação a servidores de conteúdos

Um objecto da classe de `DKDatastorexx` (em que `xx` indica um servidor de conteúdos específico) representa e gere uma ligação a um servidor de conteúdos, faculta suporte de transacção e executa comandos do servidor. Consulte Tabela 7 para obter as extensões exactas.

Tabela 7. Tipo de servidor e terminologia do nome da classe

| Servidor de conteúdos | Nome da classe |
|---|--|
| Content Manager Versão 8.2 | DKDatastoreICM |
| Content Manager anterior | DKDatastoreDL |
| Content Manager OnDemand | DKDatastoreOD |
| Content Manager for AS/400 (VisualInfo para AS/400) | DKDatastoreV4 |
| Content Manager ImagePlus for OS/390 | DKDatastoreIP |
| Domino.Doc | DKDatastoreDD |
| Extended Search | DKDatastoreDES |
| Panagon Image Services (FileNET) | DKDatastoreFN |
| Bases de dados relacionais | DKDatastoreDB2, DKDatastoreJDBC (para Java) DKDatastoreODBC (para C++) |

Estabelecer uma ligação

Cada classe de `DKDatastorexx` fornece métodos de ligação e de desligação a ela. O exemplo seguinte utiliza um servidor de bibliotecas do Content Manager denominado `ICMNLSDb`, o ID de utilizador `ICMADMIN` e a palavra-passe `PASSWORD`. Para obter informações relativas ao Content Manager, consulte *Ligação ao sistema do Content Manager*; para obter outros servidores de conteúdo, consulte *Trabalhar com outros servidores de conteúdos*. O exemplo cria um objecto de `DKDatastoreICM` para o servidor de conteúdos do CM, estabelece-lhe uma ligação, trabalha a partir do mesmo e, finalmente, anula a ligação estabelecida.

Java

```
DKDatastoreICM dsICM = new DKDatastoreICM(); //Create datastore object
dsICM.connect("ICMNLSDb","ICMADMIN","PASSWORD",""); //Connect to datastore

System.out.println("Connected to datastore dbase: '"+dsICM.datastoreName()+
    "', UserName '"+dsICM.userName()+"'");

dsICM.disconnect(); // Disconnect from datastore
dsICM.destroy(); // Destroy reference
```

Para obter a aplicação exemplo completa, consulte
SConnectDisconnectICM.java no directórioCMBROOT\Samples\java\icm.

C++

```
DKDatastoreICM* dsICM = new DKDatastoreICM(); //Create datastore object
dsICM->connect("ICMNLSDb","ICMADMIN","PASSWORD",""); //Connect to datastore

cout << "Connected to datastore dbase: '" << dsICM->datastoreName() <<
    "', UserName '" << dsICM->userName() << "')." << endl;

dsICM->disconnect(); //Disconnect from datastore
delete(dsICM); //Destroy reference
```

Para obter a aplicação exemplo completa, consulte
SConnectDisconnectICM.cpp no directórioCMBROOT\Samples\cpp\icm.

Ao estabelecer ligação a um servidor de conteúdos o utilizador terá de ter em atenção os requisitos de cada servidor de conteúdos; por exemplo, a palavra-passe para o ImagePlus para OS/390 pode não permitir mais do que oito caracteres de comprimento.

Ligar e desligar de um servidor de conteúdos num cliente

Utiliza o mesmo código em “Estabelecer uma ligação” na página 36 para aceder a um servidor de conteúdos a partir de uma aplicação cliente. Para isso, basta substituir `import com.ibm.mm.sdk.server.*;` por `import com.ibm.mm.sdk.client.*;`. A sua aplicação cliente terá de manusear quaisquer erros de comunicações que ocorram.

Definir e obter opções do servidor de conteúdos

O utilizador pode aceder ou definir as opções de processamento num servidor de conteúdos através dos métodos em `DKDatastorexx`. O exemplo seguinte mostra como definir e obter a opção de estabelecimento de uma sessão administrativa num servidor de bibliotecas do Content Manager. Consulte a *referência de API online* para obter a lista de opções e as respectivas descrições.

Neste exemplo de definição e obtenção de uma opção do servidor de conteúdos no Content Manager, a colocação em memória-cache é desactivada. No caso de servidores de conteúdos que suportam esta opção, recomenda-se que seja utilizada a predefinição (ON). **Requisito:** Já existe um objecto de `DKDatastoreICM` válido numa variável denominada `dsICM`.

Java

```
dsICM.setOption(DKConstant.DK_CM_OPT_CACHE,  
    new Integer(DKConstant.DK_CM_FALSE));  
Object val = dsICM.getOption(DKConstant.DK_CM_OPT_CACHE);
```

C++

```
DKAny inVal = DK_CM_FALSE  
DKAny outVal;  
dsICM->setOption(DK_CM_OPT_CACHE,inVal);  
dsICM->getOption(DK_CM_OPT_CACHE,outVal);
```

Ao obter uma opção de servidor de conteúdos, `output_option` normalmente é um número inteiro, mas o utilizador pode convertê-lo para um objecto.

Enumerar servidores de conteúdos

`DKDatastorexx` fornece um método de listagem dos servidores a que se pode ligar. A lista de servidores é devolvida numa `DKSequentialCollection` de objectos de `DKServerDefxx` (em que `xx` identifica o servidor de conteúdos específico).

Restrição: O servidor de conteúdos do Domino.Doc não faculta um método que enumera os servidores.

Depois de obter um objecto de `DKServerDefxx`, o utilizador poderá obter o nome do servidor e o tipo de servidor e utilizar o nome do servidor para com ele estabelecer uma ligação.

O exemplo seguinte enumera os servidores aos quais está configurado para estabelecer ligação.

Java

```
DKDatastoreICM dsICM = new DKDatastoreICM(); // Create a datastore object  
dkCollection coll = dsICM.listDataSources(); // Obtain data source list  
dkIterator iter = coll.createIterator(); // Create an iterator  
while(iter.more()){ // While there are more  
    DKServerDefICM srvrDef = (DKServerDefICM) iter.next();  
    System.out.println("Found server '"+srvrDef.getName()+"");  
}
```

C++

```
DKDatastoreICM* dsICM = new DKDatastoreICM(); // Create a datastore object  
dkCollection* coll = (dkCollection*)dsICM->listDataSources(); // Obtain list  
dkIterator* iter = coll->createIterator(); // Create an iterator  
while(iter->more()){ // While there are more  
    DKServerDefICM* srvrDef = (DKServerDefICM*) iter->next()->value();  
    cout << "Found server '" << srvrDef->getName() << "' << endl;  
    delete(srvrDef); // Free memory  
}  
delete(iter); // Free memory  
delete(coll);  
delete(dsICM);
```

Enumerar as entidades e os atributos de um servidor de conteúdos

O `DKDatastorexx` fornece métodos de listagem de entidades e os seus atributos, para um servidor de conteúdos. Cada nome de atributo faz parte de um espaço de nome. O espaço de nome predefinido é utilizado para todos os atributos em que não está especificado um espaço de nome.

A lista de entidades é devolvida num objecto de `DKSequentialCollection` de objectos de `dkEntityDef`. Os atributos para uma entidade são devolvidos num objecto de `DKSequentialCollection` dos objectos `dkAttrDef`. Após obter um objecto de `dkAttrDef`, o utilizador poderá obter informações sobre o atributo, tais como o seu nome e tipo, e utilizar as informações para formar uma consulta.

Para obter mais informações sobre estes dois métodos, consulte *referência de API online*.

O exemplo seguinte demonstra como obter a lista de tipos de artigos, bem como a lista de atributos de um servidor Content Manager.

Java

```
. . .
try {
    DKSequentialCollection pCol = null;
    dkIterator pIter = null;
    DKSequentialCollection pCol2 = null;
    dkIterator pIter2 = null;
    DKServerDefICM pSV = null;
    String strServerName = null;
    String strItemType = null;
    DKComponentTypeDefICM itemTypeDef = null;
    DKAttrDefICM attrDef = null;
    DKDatastoreDefICM dsDefICM = null;
    int i = 0;
    int j = 0;
    // ----- Create the datastore and connect (assumes the
    //      parameters for the connection are previously set)
    DKDatastoreICM dsICM = new DKDatastoreICM();
    dsICM.connect(db,userid,pw,"");
    // ----- List the item types
    pCol = (DKSequentialCollection) dsICM.listEntities();
    pIter = pCol.createIterator();
    i = 0;
    while (pIter.more() == true)
    {
        i++;
        itemTypeDef = (DKComponentTypeDefICM)pIter.next();
        strItemType = itemTypeDef.getName();
        System.out.println("item type name [" + i + "] - " + strItemType);
        System.out.println("    type " + itemTypeDef.getType());
        System.out.println("    itemTypeId " + itemTypeDef.getId());
        System.out.println("    compID " + itemTypeDef.getComponentTypeId());
        //continued . . .
    }
}
```

Java (continuação)

```
// ----- List the attributes
pCol2 = (DKSequentialCollection) dsICM.listEntityAttrs(strItemType);
pIter2 = pCol2.createIterator();
j = 0;
while (pIter2.more() == true)
{
    j++;
    attrDef = (DKAttrDefICM)pIter2.next();
    System.out.println("Attr name [" + j + "] - " + attrDef.getName());
    System.out.println("    datastoreType " + attrDef.datastoreType());
    System.out.println("    attributeOf " + attrDef.getEntityName());
    System.out.println("    type " + attrDef.getType());
    System.out.println("    size " + attrDef.getSize());
    System.out.println("    id " + attrDef.getId());
    System.out.println("    nullable " + attrDef.isNullable());
    System.out.println("    precision " + attrDef.getPrecision());
    System.out.println("    scale " + attrDef.getScale());
    System.out.println("    stringType " + attrDef.getStringType());
    System.out.println("    sequenceNo " + attrDef.getSequenceNo());
    System.out.println("    userFlag " + attrDef.getUserFlag());
}
dsICM.disconnect();
}
catch(DKException exc)
{
    // ----- Handle the exceptions
```

Um exemplo completo, `SItemTypeRetrievalICM`, inclui como enumerar definições de tipos de artigo. Outro exemplo completo, `SAttributeDefinitionRetrievalICM`, inclui como enumerar definições de atributos. Ambos os exemplos estão disponíveis no directório `CMBROOT\Samples\java\icm`.

O exemplo seguinte de C++ mostra como obter a lista de classes e atributos de índice a partir de um servidor do Content Manager:

C++

```
// Get a collection containing all Item Type Definitions.
DKSequentialCollection * itemTypeColl = (DKSequentialCollection *)
    dsICM->listEntities();
// Accessing each and printing the name & description.
cout << "\nItem Type Names in System:
    (" << itemTypeColl->cardinality() << ')' << endl;
// Create an iterator to iterate through the collection
dkIterator* iter = itemTypeColl->createIterator();
// while there are still items in the list, continue
while(iter->more()){
    DKItemTypeDefICM* itemType = (DKItemTypeDefICM*) iter->next()->value();
    cout << " - " << itemType->getName() << ": " <<
        itemType->getDescription() << endl;
    delete(itemType); // Free Memory

cout << endl;
delete(iter);
delete(itemTypeColl);
```


Para obter mais informações, consulte o exemplo `SItemTypeInfoRetrievalICM`, o qual inclui como enumerar definições de tipo de artigo. O `SAttributeDefinitionRetrievalICM`, inclui como enumerar definições de atributos. Os exemplos encontram-se disponíveis no directório `CMBROOT\Samples\cpp\icm`.

Sugestão: Em vez de eliminar imediatamente a `pEnt`, o utilizador pode deferi-la e utilizar a função `aplicar` para eliminar a definição do atributo que está dentro da recolha. Consulte “Gerir memória em recolhas (Apenas para C++)” na página 98 para obter mais informações.

C++

```
...
pCol2->apply(deleteDKAttrDefICM);
delete pCol2;
...
```

Trabalhar com objectos de dados dinâmicos (DDOs)

Esta secção descreve como utilizar um DDO e contém exemplos que ajudam o utilizador a compreender como:

1. Associar um DKDDO a um servidor de conteúdos.
2. Criar um DKDDO.
3. Criar Identificadores Permanentes (PIDs) para atributos DKDDO.
4. Adicionar atributos e definir propriedades de atributo.
5. Definir o DKDDO como uma pasta ou um documento.
6. Definir e visualizar valores das propriedades de atributo.
7. Verificar as propriedades de DKDDO.
8. Verificar as propriedades dos atributos.
9. Apresentar o conteúdo de DKDDO.
10. Eliminar o DKDDO.

Utilize a classe de DKDDO para objectos de dados dinâmicos (DDOs) nas suas aplicações da IBM Enterprise Information Portal for Multiplatforms. Um objecto DKDDO representa um artigo que, por exemplo, poderá ser um documento ou uma pasta de Content Manager ou um objecto definido pelo utilizador. Um objecto de DKDDO contém atributos. Cada atributo tem um nome, valor e propriedades. Cada atributo é identificado por um ID de dados. Os atributos são numerados consecutivamente, começando pelo 1; o número do atributo é o ID dos dados.

Visto que o nome, valor e propriedade de um atributo pode variar, o DKDDO fornece mecanismos flexíveis para representar dados que têm origem numa variedade de servidores de conteúdos e de formatos. Por exemplo, artigos de diferentes tipos de artigos em Content Manager ou linhas de tabelas diferentes numa base de dados relacional. O próprio DKDDO pode ter propriedades que se aplicam a todo o DKDDO, em vez de apenas a um atributo em particular.

O utilizador irá associar um DKDDO a um servidor de conteúdos antes de chamar os métodos adicionar, obter, actualizar e eliminar para colocar os seus atributos num servidor de conteúdos ou recuperá-los. Defina o servidor de conteúdos como parâmetro que irá criar o objecto de DKDDO ou chamando o método `setDatastore`.

Cada DKDDO tem um identificador de objectos permanente (PID) que contém as informações para localização dos atributos no servidor de conteúdos.

Criar um DKDDO

O DKDDO tem vários construtores. Pode criar um DKDDO chamando o seu construtor sem quaisquer parâmetros.

Java

```
DKDDO ddo = new DKDDO();
```

C++

```
DKDDO ddo;
```

O ddo desde DDO deve aumentar dinamicamente para acomodar mais atributos. Para obter um construtor mais eficiente, transmita o número exacto de atributos que pretende (por exemplo, dez).

Java

```
DKDDO ddo = new DKDDO(10);
```

C++

```
DKDDO ddo = new DKDDO((short)10);
```

Importante: Em APIs como DKDatastoreICM e DKDatastoreOD, crie um DKDDO utilizando os métodos createDDO() na classe de DKDatastore XX. No CM V8, os DDOs têm de ser criados utilizando estes métodos. O exemplo seguinte cria um DKDDO transmitindo o servidor de conteúdos e o tipo de objecto ao Content Manager Versão 8:

Java

```
DKDatastoreICM dsICM = new DKDatastoreICM(); //create a CM datastore  
DKDDO ddo=dsICM.createDDO("ICMSAMPLE", //create a DDO to hold an object type  
DKConstant.DK_CM_DOCUMENT);
```

Para obter mais informações relativas à criação de DDO, consulte o exemplo SItemCreationICM disponível no directório CMBROOT\Samples\java\icm.

C++

```
// create a Content Manager datastore
DKDatastoreICM* dsICM = new DKDatastoreICM();
// create a DDO to hold an object type
DKDDO* ddo = dsICM->createDDO("ICMSAMPLE",
DK_CM_DOCUMENT);
```

Para obter mais informações relativas à criação de DDO, consulte o exemplo `SItemCreationICM` disponível no directório `CMBROOT\Samples\cpp\icm`.

Para obter outros conectores (como, por exemplo, versões anteriores do Content Manager), o utilizador pode criar um `DKDDO` facultando o servidor de conteúdos e o tipo de objecto com o seguinte exemplo:

Java

```
DKDatastoreDL dsDL=new DKDatastoreDL(); //create a CM datastore
DKDDO ddo=new DKDDO(dsDL, "DLSAMPLE"); //create a DDO to hold an object type
//DLSAMPLE in dsDL
```

C++

```
// create an earlier Content Manager datastore
DKDatastoreDL dsDL;
// create a DDO to hold an object type DLSAMPLE in dsDL
DKDDO* cddo = new DKDDO(&dsDL, "DLSAMPLE");
```

O construtor que é usado pelo utilizador irá depender da aplicação deste último; consulte *referência de API online* para obter informações relativas a construtores.

Adicionar propriedades a um DDO

Ao criar um objecto de `DKDDO` para representar um DDO, tem de especificar a sua propriedade de tipo de artigo (um documento, pasta ou artigo).

Pode transmitir esta propriedade como uma das opções do método `DKDatastoreXX.createDDO(itemTypeName,itemPropertyType/SemanticType)`.

`DKDatastoreICM.createDDO(itemTypeName,itemPropertyType/SemanticType)`

Ou, caso já tenha criado o `DKDDO` sem definir a sua propriedade de tipo de artigo, pode utilizar o exemplo seguinte para definir o tipo de DDO como "documento".

Java

```
//----- Add the property that it is a document
ddo.addProperty(DK_CM_PROPERTY_ITEM_TYPE, new Short(DK_CM_DOCUMENT));
```

C++

```
any = DK_CM_DOCUMENT; // it is a document
ddo->addProperty(DK_CM_PROPERTY_ITEM_TYPE, any);
```

Criar um identificador permanente (PID)

Cada DDO terá de ter um identificador permanente (PID). O PID contém informações sobre o nome, tipo, ID e tipo de objectos do servidor de conteúdos. O PID identifica a localização de dados permanentes do DDO. Por exemplo, em servidores de conteúdo de versões anteriores do Content Manager, o PID corresponde ao ID do artigo. O ID do artigo é um dos parâmetros mais importantes para as funções de obtenção, actualização e eliminação. No Content Manager V8, o PID possui cinco partes (consulte *Trabalhar com o Content Manager 8.2* para obter mais informações).

Para a função adicionar, o servidor de conteúdos cria e devolve o ID do artigo. Por exemplo, os conectores que facultam o método `DKDatastoreXX.createDDO()` criam automaticamente o objecto de PID na operação `DKDDO.add()`.

O exemplo seguinte cria um DDO para obter um artigo conhecido:

Java

```
// Given a connected DKDatastoreICM object named "dsICM"
// Create a new DDO
DKDDO ddo = dsICM.createDDO("book",DKConstant.DK_CM_DOCUMENT);
// PID automatically created by the function above.
ddo.add(); // Add the new item to the datastore.
// PID Completed by the System
DKPidICM pid = (DKPidICM)ddo.getPidObject();
```

C++

```
// Given a connected DKDatastoreICM object named "dsICM"
// Create a new DDO
DKDDO * ddo = dsICM->createDDO("book",DK_CM_DOCUMENT);
// PID automatically created by the function above.
ddo->add(); // Add the new item to the datastore.
// PID Completed by the System
DKPidICM * pid = (DKPidICM *)ddo->getPidObject();
```

Ligue-se ao servidor de conteúdos e chame a função de obtenção para obter o DDO criado no exemplo.

O conector de Content Manager 8.2 possui uma classe de PID que é uma sub-classe de `DKPid`. Denomina-se `DKPidICM` que corresponde ao pid utilizado por `DKDDOs` e `dkResources`.

Trabalhar com artigos e propriedades de dados

O `DKDDO` fornece métodos para adicionar propriedades de atributos a um objecto de `DKDDO`.

Parta do princípio que um tipo de artigo `NameOfItemType` possui os atributos `Name` de tipo `integer` e está definido como nulo no repositório do CM V8. O utilizador cria um objecto de `DKDDO` para manusear um artigo dessa entidade e pretende adicionar dois artigos de dados ao `DKDDO`.

O exemplo seguinte cria um artigo, define propriedades de atributos e guarda-as no servidor de conteúdos permanente do Content Manager. **Requisito:** Parte do princípio que possui uma `DKDatstoreICM` ligada na variável `dsICM`. O tipo de artigo definido pelo utilizador `S_withChild` deve ser definido com `varchar` `S_varchar`, `long integer` `S_integer`, `short integer` `S_short` e `time` `S_time`, tal como está definido no Exemplo de Instrução de API `SItemCreationICM`.

Java

```
// Create a new item in memory
DKDDO ddo = dsICM.createDDO("S_withChild", DKConstant.DK_CM_DOCUMENT);

// Set attributes (Additional attributes set in SItemCreationICM sample)
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,"S_varchar"),
    "abcdefg");
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,"S_integer"),
    new Integer("123"));
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,"S_short"),
    new Short("5"));
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,"S_time"),
    java.sql.Time.valueOf("10:00:00"));

// Add to datastore
ddo.add();
```

A aplicação exemplo completa da qual este exemplo foi retirado(`SItemCreationICM`) encontra-se disponível no directório `CMBROOT\Samples\java\icm`.

C++

```
// Create a new item in memory
DKDDO* ddo = dsICM->createDDO("S_withChild", DK_CM_DOCUMENT);

// Set attributes (Additional attributes set in SItemCreationICM sample)
// NOTE: Values below are automatically converted to type DKAny
ddo->setData(ddo->dataId(DK_CM_NAMESPACE_ATTR,"S_varchar"),
    DKString("this is a string value"));
ddo->setData(ddo->dataId(DK_CM_NAMESPACE_ATTR,"S_integer"),
    (long) 123);
ddo->setData(ddo->dataId(DK_CM_NAMESPACE_ATTR,"S_short"),
    (short) 5);
ddo->setData(ddo->dataId(DK_CM_NAMESPACE_ATTR,"S_time"),
    DKTime("10.00.00"));

// Add to datastore
ddo->add();
```

A aplicação exemplo completa da qual este exemplo foi retirado(`SItemCreationICM`) encontra-se disponível no directório `CMBROOT\Samples\cpp\icm`.

O utilizador deve definir o tipo da propriedade de um atributo; passível de ser convertido em nulo e outras propriedades são opcionais.

O exemplo seguinte acede a valores de atributos de um DDO.

Java

```
// Cast operations will enable access as subclass of Object type returned.  
// NOTE: Additional attributes accessed in SItemRetrievalICM sample.
```

```
String attrVal1 = (String) ddo.getData(ddo.dataId(  
    DKConstant.DK_CM_NAMESPACE_ATTR,"S_varchar"));  
Integer attrVal2 = (Integer) ddo.getData(ddo.dataId(  
    DKConstant.DK_CM_NAMESPACE_ATTR,"S_integer"));  
Short attrVal3 = (Short) ddo.getData(ddo.dataId(  
    DKConstant.DK_CM_NAMESPACE_ATTR,"S_short"));  
Time attrVal4 = (Time) ddo.getData(ddo.dataId(  
    DKConstant.DK_CM_NAMESPACE_ATTR,"S_time"));  
  
System.out.println("Attr 'S_varchar' value: "+attrVal1);  
System.out.println("Attr 'S_integer' value: "+attrVal2);  
System.out.println("Attr 'S_short' value: "+attrVal3);  
System.out.println("Attr 'S_time' value: "+attrVal4);
```

A aplicação exemplo completa da qual este exemplo foi retirado (SItemRetrievalICM) encontra-se disponível no directório CMBROOT\Samples\java\icm.

C++

```
// Assignment and cast operations converts values from DKAny to each type.  
// NOTE: Additional attributes accessed in SItemRetrievalICM sample.
```

```
DKString attrVal1 = ddo->getData(ddo->dataId(DK_CM_NAMESPACE_ATTR,  
    DKString("S_varchar"))).toString();  
long attrVal2 = (long) ddo->getData(ddo->dataId(DK_CM_NAMESPACE_ATTR,  
    DKString("S_integer")));  
short attrVal3 = (short) ddo->getData(ddo->dataId(DK_CM_NAMESPACE_ATTR,  
    DKString("S_short")));  
DKTimestamp attrVal4 = (DKTimestamp) ddo->getData(ddo->dataId(  
    DK_CM_NAMESPACE_ATTR, DKString("S_time")));  
  
cout << "Attr 'S_varchar' value: " << attrVal1 << endl;  
cout << "Attr 'S_integer' value: " << attrVal2 << endl;  
cout << "Attr 'S_short' value: " << attrVal3 << endl;  
cout << "Attr 'S_time' value: " << attrVal4 << endl;
```

A aplicação exemplo completa da qual este exemplo foi retirado (SItemRetrievalICM) encontra-se disponível no directório CMBROOT\Samples\cpp\icm.

Obter o DKDDO e as propriedades de atributo

Ao processar um DKDDO, o utilizador deve primeiro saber qual é o seu tipo: documento, pasta ou artigo. O código exemplo seguinte demonstra como determinar o tipo de DDO.

Java

```
short prop_id = ddo.propertyId(DK_CM_PROPERTY_ITEM_TYPE);
if (prop_id > 0) {
    short type = ((Short) ddo.getProperty(prop_id)).shortValue();
    switch(type) {
        case DK_CM_DOCUMENT:
            // --- process a document
            ....
            break;
        case DK_CM_FOLDER:
            // --- process a folder
        case DK_CM_ITEM:
            // --- Process an item in Content Manager
            ....
            break;
    }
}
```

Para obter mais informações relativas a aceder a propriedades de tipo de artigo, consulte o Exemplo de Instrução de API SItemRetrievalICM, que se encontra disponível no directório CMBROOT\Samples\java\icm.

C++

```
unsigned short prop_id =
    ddo->propertyId(DK_CM_PROPERTY_ITEM_TYPE);
if (prop_id > 0) {
    unsigned short type = (unsigned short) ddo->getProperty(prop_id);
    switch(type) {
        case DK_CM_DOCUMENT:
            // process document
            ...
            break;
        case DK_CM_FOLDER:
            // process folder
            ...
            break;
    }
}
```

Para obter mais informações relativas a aceder a propriedades de tipo de artigo, consulte o Exemplo de Instrução de API SItemRetrievalICM, que se encontra disponível no directório CMBROOT\Samples\cpp\icm.

Para obter propriedades de um atributo, o utilizador deve obter o data_id do atributo e, de seguida, poderá obter as propriedades.

Tanto o data_id como o property_id começam por 1. Se especificar 0, receberá uma excepção.

Java

```
data_id = ddo.dataId("Title"); // get data_id of Title
// ----- Get the number of properties for the attribute
short number_of_data_prop = ddo.dataPropertyCount(data_id);
// ----- Display all data properties belonging to this attribute
//          using a loop; the index starts at 1
for(short i = 1; i <= number_of_data_prop; i++) {
    System.out.println(i + " Property Name = " +
        ddo.getDataPropertyName(data_id,i)
        + " value = " + ddo.getDataProperty(data_id,i));
}
```

Para obter a aplicação exemplo completa, consulte SItemRetrievalICM no directórioCMBROOT\Samples\java\icm.

C++

```
// get data_id of Title
data_id = ddo->dataId("Title");
// how many props does it have?
unsigned short number_of_data_prop = ddo->dataPropertyCount(data_id);
// displays all data properties belonging to this attribute
// notice that the loop index starts from 1, where
// 1 <= i <= number_of_data_prop
for (unsigned short i = 1; i <= number_of_data_prop; i++) {
    cout << i << " Property Name = " << ddo->
        getDataPropertyName(data_id, i) << " value = " << ddo->
        getDataProperty(data_id, i) << endl;
}
```

Para obter a aplicação exemplo completa, consulte SItemRetrievalICM no directórioCMBROOT\Samples\cpp\icm.

Apresentar todo o DDO

Durante o desenvolvimento da aplicação, poderá ter de apresentar o conteúdo de um DKDDO com a finalidade de depuração.

Java

```
short number_of_attribute = ddo.dataCount();
short number_of_prop      = ddo.propertyCount();
short number_of_data_prop;
// list DDO properties
for (short k = 1; k <= number_of_prop; k++) {
    System.out.println( k + " Property Name = " + ddo.getPropertyName(k) +
        ",\t value = " + ddo.getProperty(k));
}
// list data-items and their properties
for (short i = 1; i <= number_of_attribute; i++) {
    System.out.println( i + " Attr. Name = " + ddo.getDataName(i) +
        ",\t value = " +
        number_of_data_prop = ddo.dataPropertyCount(i);
    for (short j = 1; j <= number_of_data_prop; j++) {
        System.out.println( "\t" + j + " Data Prop. Name = " +
            ddo.getDataPropertyName(i,j) +
            ",\t value = " +
            ddo.getDataProperty(i,j));
    }
}
```

Para obter um exemplo completo de acesso e impressão de um DDO (e todos os sub-componentes), consulte `SItemRetrievalICM` e a sua função estática `printDDO()` no directório `CMBROOT\Samples\java\icm`.

C++

```
unsigned short number_of_attribute = ddo->dataCount();
unsigned short number_of_prop;
unsigned short number_of_data_prop;
// list DDO properties
for (short k = 1; k <= number_of_prop; k++) {
    cout << k << " Property Name = " << ddo->getPropertyName(k) <<
        ",\t value = " << ddo->getProperty(k) << endl;
}
// list data-items and their properties
for (unsigned short i = 1; i <= number_of_attribute; i++) {
    cout << i << " Attr. Name = " << ddo->getDataName(i) <<
        "<< ",\t value = " << ddo->getData(i) << endl;
    number_of_data_prop = ddo->dataPropertyCount(i);
    for (unsigned short j = 1; j <= number_of_data_prop; j++) {
        cout << "\t" << j << " Nome Prop. Dados = "
            << ddo->getDataPropertyName(i, j)
            << ",\t value = " << ddo->getDataProperty(i, j)
            << endl;
    }
}
```

Para obter um exemplo completo de acesso e impressão de um DDO (e todos os sub-componentes), consulte `SItemRetrievalICM` e a sua função estática `printDDO()` no directório `CMBROOT\Samples\cpp\icm`.

Eliminar um DDO (Apenas para C++)

Um DKDDO tem duas representações: a da memória e a cópia permanente. Para eliminar o DKDDO da memória, chame o seu destruidor. Tenha em atenção que esta acção não altera a cópia permanente no servidor de conteúdos.

Para eliminar a cópia permanente no servidor de conteúdos, utilize a função `dkddo:del()`. Isto não afecta a representação DKDDO na memória (os valores de atributo encontram-se num objecto de DKAny). O destruidor elimina as referências de objectos a `dkCollection` e a `dkDataObjectBase`, incluindo referências e `DKParts`, `DKFolder`, `DKDDO` e `DKBlob`.

Trabalhar com objectos de dados expandidos (XDOs)

Um XDO representa um componente que pode armazenar conteúdos de recursos como, por exemplo, um artigo de recurso ou uma parte de documento.

Os artigos de recurso (XDOs) expandem artigos que não são de recurso (DDOs). A criação de artigos de recurso é idêntica aos artigos comuns. A área de Artigos de Recurso é criada de forma idêntica à área dos Artigos comuns. Dependendo do tipo de Artigo de recurso, o XDO pode ser ainda mais expandido.

Java

Class Hierarchy

| Type | DDO | XDO | Extension |
|--------|-------|-------------|----------------|
| ----- | ----- | ----- | ----- |
| Lob | DKDDO | -> DKLobICM | |
| Text | DKDDO | -> DKLobICM | -> DKTextICM |
| Image | DKDDO | -> DKLobICM | -> DKImageICM |
| Stream | DKDDO | -> DKLobICM | -> DKStreamICM |

Apenas para Content Manager: Visto que o CM 8 requer que os XDOs sejam da sub-classe correcta, os DKDDOs devem sempre ser criados utilizando os métodos `createDDO()` de `DKDatastoreICM`. Isto permite ao sistema configurar automaticamente as informações na estrutura de DKDDO e faculta uma maior funcionalidade como, por exemplo, recursos, modelo de documentos do CM e pastas. Os artigos de tipo recurso (devolvidos de `DKDatastoreICM.createDDO()`) podem ser convertidos em XDO ou sub-classe correctos, dependendo da classificação de XDO. Para obter mais informações relativas à criação de artigos em geral e da função `DKDatastoreICM.createDDO()`, consulte o exemplo `SItemCreationICM`.

Para criar um XDO para objectos binários utilize `DKBlobxx`, em que `xx` corresponde ao sufixo que representa o servidor específico. Por exemplo, utilize `DKBlobICM` para Content Manager, `DKBlobOD` para OnDemand ou `DKBlobIP` para ImagePlus for OS/390. Quando cria um objecto de `DKBlobxx`, o utilizador deve trasmiti-lo ao servidor de conteúdos `DKDatastorexx`. Para Content Manager, utiliza `DKLobICM` para criar o XDO.

Utilizar um identificador permanente de XDO (PID)

Um XDO necessita de um PID para armazenar os dados de forma permanente. Para utilizar um XDO de forma a localizar e armazenar dados, terá de fornecer um PID para o `DKBlobxx`, através de um `DKPidXDOxx`. As Bases de Dados relacionais necessitam de uma tabela, uma coluna e de uma cadeia de predicado de dados para localizar dados num servidor de conteúdos. Para bases de dados relacionais (RDB), são necessários o nome da tabela, o nome da coluna e o predicado de dados para o `DKPidXDOxx`.

No Conector de ICM, utilize `DKPidICM` para representar o pid de um objecto `dkResource` que consiste num XDO.

Compreender as propriedades do XDO

Utilize os métodos do `DKBlobxx` para definir as propriedades de um XDO onde quer que se apliquem; não estão disponíveis todas as propriedades para todos os servidores de conteúdos. Ao carregar, os valores predefinidos para as propriedades são definidos, caso não sejam especificados valores específicos. Por exemplo, as seguintes predefinições são utilizadas em versões anteriores do Content Manager:

TipoRep (tipo de representação)

A predefinição é `FRN$NULL`. Para o `VisualInfo` for AS/400, terá de utilizar " ", oito espaços em branco entre aspas.

ClasseConteúdos

A predefinição é `DK_CM_CC_UNKNOWN`. Para os valores válidos, consulte `DKConstant2DL.h` no directório `\cmbroot\include` para o Enterprise Information Portal.

TipoAssociado

A predefinição é: `DK_DL_BASE`.

DadosAssociados

A predefinição é: `NULL`.

Para indexar correctamente o conteúdo dos objectos com o Content Manager anterior, terá de definir `SearchEngine`, `SearchIndex` e `SearchInfo` no objecto de extensão `DKSearchEngineInfoDL`.

Para trabalhar com XDOs no Content Manager, consulte “Trabalhar com artigos” na página 145.

Sugestão de C++: Para obter os valores válidos de `ContentClass`, consulte o ficheiro `INCLUDE/DKConstant2DL.h` facultado com o Content Manager.

Cadeias de configuração de DB2, ODBC e DataJoiner (Apenas para C++)

Esta secção define as cadeias de configuração de C++ DB2, ODBC e de DataJoiner.

CC2MIMEFILE=(nomeficheiro)

Especifica o ficheiro `cmbcc2mime.ini` (opcional).

DSNAME=(nome do servidor de conteúdos)

Especifique o nome do servidor de conteúdos (facultativo). Quando este servidor de conteúdos é utilizado por Associado, esta opção é automaticamente definida.

AUTOCOMMIT=ON | OFF

Especifique se a consolidação automática está ligada ou desligada. A predefinição é desligada (facultativo). Quando este servidor de conteúdos é utilizado por Associado, a consolidação automática está sempre activada. Esta é definida automaticamente.

Esta secção define as cadeias de ligação de C++ DB2, ODBC e de DataJoiner.

NATIVECONNECTSTRING=(cadeia de ligação nativa)

Especifique a passagem de uma cadeia de ligação nativa para a chamada de ligação nativa (opcional).

SCHEMA=nome

Especifique o esquema a ser utilizado nas funções de `listEntities`, `listEntityAttrs`, `listPrimaryKeyNames`, `listForeignKeyNames` (facultativo).

Sugestões de programação de Java

Para o Content Manager V8 e versões posteriores, um XDO é um objecto dkResource. DKPidICM é utilizado para representar o PID do objecto de recurso.

No caso de versões anteriores do Content Manager, Content Manager for AS/400 e IP 390, o utilizador identifica um XDO através da combinação do ID de artigo, do ID de parte e do TipoRep. Para RDB, a chave para a identificação de um XDO é a combinação da cadeia de tabela, coluna e predicado de dados. Para manusear um XDO autónomo, o utilizador irá fornecer o ID do artigo e o ID da parte. O TipoRep é opcional, uma vez que o sistema lhe fornece um valor predefinido.

Utilize o método de adição de DKBlobxx para adicionar o actual conteúdo ao servidor de conteúdos. Poderá obter o valor do ID da parte depois de adicionar caso pretenda efectuar mais alguma operação com esse objecto posteriormente.

Utilize o método getPidObject() em dkXDO para obter o objecto DKPid.

Poderá utilizar a seguinte instrução depois de adicionar para obter o ID da parte atribuído ao sistema:

Java

```
int partID = ((DKPidXDOICM)(axdo.getPidObject())).getPartId();
```

Atenção: Em versões anteriores do Content Manager, o utilizador necessita de um ID de parte válido para adicionar uma parte para ser indexada pelo gestor de pesquisas (não pode definir o ID de parte como 0).

Nesta edição, foram modificados dois métodos em dkXDO: DKPid dkXDO.getPid() é desactualizado e substituído por getPidObject. DKPid dkXDO.getPidObject() Estes métodos devolviam um DKPidXDO, mas actualmente devolvem um objecto DKPid.

Sugestões de programação de C++

No Content Manager, VI400 e IP390, o utilizador irá identificar um XDO através da combinação do ID do artigo, do ID da parte e do TipoRep. Para Bases de Dados Relacionais, a combinação entre o nome da tabela, o nome da coluna e o predicado de dados é a chave para identificar um XDO. Para um XDO autónomo, terá de fornecer o ID do artigo e o ID da parte. O TipoRep é opcional, visto que o sistema fornece um valor predefinido (FRN\$NULL).

Para a função de add, deve facultar um ID de parte. Poderá obter o valor do ID da parte depois de adicionar caso pretenda efectuar mais alguma operação com esse objecto posteriormente.

Importante: Ao adicionar uma parte para o gestor de pesquisas indexar num servidor de conteúdos do Content Manager, deve possuir um ID de parte válido e não pode definir o ID de parte como 0.

Programar um XDO como parte de DDO

Um XDO representa um objecto de parte único, se possuir um DDO a representar um documento, que consiste num conjunto de objectos de conteúdo de recurso. Pode manipular o XDO enquanto componente do DDO ou como um objecto

autónomo. Quando aceder ao XDO como parte do DDO, o DDO fornece o ID do artigo. Ao utilizar o XDO como um objecto autónomo, terá de utilizar o ID do artigo existente para o XDO.

O exemplo seguinte cria um documento e adiciona partes de documentos ao Content Manager. **Requisito:** O tipo de artigo definido pelo utilizador, `S_docModel`, deve ser definido no sistema, classificado como Modelo de Documentos e suportar tipos de parte `ICMBASE` e `ICMBASETEXT`, tal como está definido no Exemplo de Instrução de API `SItemTypeCreationICM`.

Java

```
// Create a document
DKDDO ddoDocument = dsICM.createDDO("S_docModel", DKConstant.DK_CM_DOCUMENT);

// Create parts
DKLobICM base = (DKLobICM) dsICM.createDDO("ICMBASE",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASE);
DKTextICM baseText1 = (DKTextICM) dsICM.createDDO("ICMBASETEXT",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASETEXT);
DKTextICM baseText2 = (DKTextICM) dsICM.createDDO("ICMBASETEXT",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASETEXT);

// Set parts' MIME type (SResourceItemMimeTypesICM.txt sample)
base.setMimeType("application/msword");
baseText1.setMimeType("text/plain");
baseText2.setMimeType("text/plain");

// Load content into parts (SResourceItemCreationICM sample)
base.setContentFromClientFile("SResourceItemICM_Document1.doc");
// Load file
baseText1.setContentFromClientFile("SResourceItemICM_Text1.txt");
// into memory
baseText2.setContentFromClientFile("SResourceItemICM_Text2.txt");

// Access the DKParts attribute
DKParts dkParts = (DKParts) ddoDocument.getData(ddoDocument.dataId(
DKConstant.DK_CM_NAMESPACE_ATTR, DKConstant.DK_CM_DKPARTS));

// Add parts to document
dkParts.addElement(base);
dkParts.addElement(baseText1);
dkParts.addElement(baseText2);

// Add new document to persistent datastore
ddoDocument.add();
```

Para obter a aplicação exemplo completa, consulte `SDocModelItemICM.java` no directório `CMBROOT\Samples\java\icm`. `SResourceItemCreationICM` apresenta mais exemplos da utilização de XDO.

C++

```
// Create a document
DKDDO* ddoDocument = dsICM->createDDO("S_docModel", DK_CM_DOCUMENT);
// Create Parts
DKLobICM* base = (DKLobICM*) dsICM->createDDO("ICMBASE",
                                                DK_ICM_SEMANTIC_TYPE_BASE);
DKTextICM* baseText1 = (DKTextICM*) dsICM->createDDO("ICMBASETEXT",
                                                       DK_ICM_SEMANTIC_TYPE_Basetext);
DKTextICM* baseText2 = (DKTextICM*) dsICM->createDDO("ICMBASETEXT",
                                                       DK_ICM_SEMANTIC_TYPE_Basetext);

// Set parts' MIME type (SResourceItemMimeTypesICM.txt sample)
base->setMimeType("application/msword");
baseText1->setMimeType("text/plain");
baseText2->setMimeType("text/plain");
// Load content into parts (SResourceItemCreationICM sample)
// Load the file into memory.
base->setContentFromClientFile("SResourceItemICM_Document1.doc");
baseText1->setContentFromClientFile("SResourceItemICM_Text1.txt");
baseText2->setContentFromClientFile("SResourceItemICM_Text2.txt");
// Access the DKParts attribute
DKParts* dkParts = (DKParts*)(dkCollection*) ddoDocument->getData(
    ddoDocument->dataId(DK_CM_NAMESPACE_ATTR, DK_CM_DKPARTS));

// Add parts to document
dkParts->addElement((dkDataObjectBase*) (DKDDO*)base);
dkParts->addElement((dkDataObjectBase*) (DKDDO*)baseText1);
dkParts->addElement((dkDataObjectBase*) (DKDDO*)baseText2);
// Add new document to persistent datastore
ddoDocument->add();
```

Para obter a aplicação exemplo completa, consulte SDocModelItemICM.java no directório CMBROOT\Samples\cpp\icm. SResourceItemCreationICM apresenta mais exemplos da utilização de XDO.

Programar um XDO autónomo

Todos os exemplos que se seguem são específicos do Content Manager 8.2. Para obter exemplos de versões anteriores do Content Manager e de outros servidores de conteúdos, consulte “Representar artigos através de DDOs” na página 143, “Trabalhar com outros servidores de conteúdos” na página 245 e consulte os programas exemplo no directório CMBROOT\Samples.

Adicionar um XDO a partir da memória tampão

Este exemplo mostra como adicionar um XDO a partir de uma memória tampão no Content Manager. Cria um XDO, carrega conteúdo na memória e armazena permanentemente o conteúdo na memória. **Requisito:** O tipo de artigo definido pelo utilizador S_lob de recurso de classificação deve ser definido no sistema, tal como está definido no Exemplo de Instrução de API SItemTypeCreationICM. Para além disso, as definições do gestor de recursos e do conjunto de SMS devem ser configuradas e estabelecidas como predefinições do tipo de artigo ou do utilizador, tal como os exemplos SResourceMgrDefCreationICM, SSMSCollectionDefCreationICM, SResourceMgrDefSetDefaultICM e SSMSCollectionDefSetDefaultICM demonstram.

Java

```
// Create an empty resource object
DKLobICM lob = (DKLobICM) dsICM.createDDO("S_lob", DKConstant.DK_CM_DOCUMENT);

// Set the MIME type (SResourceItemMimeTypesICM.txt sample)
lob.setMimeType("application/msword");

// Load content into item's local memory
lob.setContentFromClientFile("SResourceItemICM_Document1.doc");

// Add to datastore with content already in memory
lob.add();
```

Para obter a aplicação exemplo completa, consulte SResourceItemCreationICM no directórioCMBROOT\Samples\java\icm.

C++

```
// Create an empty resource object
DKLobICM* lob = (DKLobICM*) dsICM->createDDO("S_lob", DK_CM_DOCUMENT);

// Set the MIME type (SResourceItemMimeTypesICM.txt sample)
lob->setMimeType("application/msword");

// Load content into item's local memory
lob->setContentFromClientFile("SResourceItemICM_Document1.doc");

// Add to datastore With content already in memory
lob->add();
```

Para obter a aplicação exemplo completa, consulte SResourceItemCreationICM no directórioCMBROOT\Samples\cpp\icm.

Adicionar um XDO a partir de um ficheiro

O exemplo seguinte adiciona um XDO ao servidor de conteúdos (armazenar o conteúdo directamente do ficheiro) do Content Manager. **Requisito:** O tipo de artigo definido pelo utilizador S_lob de recurso de classificação deve ser definido no sistema, tal como está definido no Exemplo de Instrução de API SItemTypeCreationICM. Para além disso, as definições do gestor de recursos e do conjunto de SMS devem ser configuradas e estabelecidas como predefinições do tipo de artigo ou do utilizador, tal como os exemplos SResourceMgrDefCreationICM, SSMSCollectionDefCreationICM, SResourceMgrDefSetDefaultICM e SSMSCollectionDefSetDefaultICM demonstram.

Java

```
// Create an empty resource object
DKTextICM text = (DKTextICM) dsICM.createDDO("S_text", DKConstant.DK_CM_ITEM);

// Set the MIME type (SResourceItemMimeTypesICM.txt sample)
text.setMimeType("text/plain");

// Store content directly from a file
text.add("SResourceItemICM_Text1.txt");
```

Para obter a aplicação exemplo completa, consulte SResourceItemCreationICM no directórioCMBROOT\Samples\java\icm.

C++

```
// Create an empty resource object
DKTextICM* text = (DKTextICM*) dsICM->createDDO("S_text", DK_CM_ITEM);

// Set the MIME type (SResourceItemMimeTypesICM.txt sample)
text->setMimeType("text/plain");

// Store content directly from a file
text->add("SResourceItemICM_Text1.txt");
```

Para obter a aplicação exemplo completa, consulte SResourceItemCreationICM no directórioCMBROOT\Samples\cpp\icm.

Adicionar um objecto de anotação a um XDO

O exemplo seguinte adiciona uma parte de anotação a um documento no Content Manager. **Requisito:** O tipo de artigo definido pelo utilizador, S_docModel, deve ser definido no sistema, classificado como Modelo de Documentos e suportar tipos de parteICMANNOTATION, tal como está definido no exemplo SItemTypeCreationICM. Parta do princípio que lhe foi atribuída uma instância de um documento já armazenado permanentemente na variável ddoDocument. Parta também do princípio que lhe é atribuído um objecto de DKDatastoreICM na variável dsICM.

Java

```
// Check out / lock the item for update
dsICM.checkOut(ddoDocument);

// Create annotation part
DKLobICM annot = (DKLobICM) dsICM.createDDO("ICMANNOATATION",
                                             DKConstantICM.DK_ICM_SEMANTIC_TYPE_ANNOTATION);

// Set annotatoin MIME type (SResourceItemMimeTypesICM.txt sample)
annot.setMimeType("image/bmp");

// Load content into parts (SResourceItemCreationICM sample)
annot.setContentFromClientFile("myAnnotation.bmp");

// Access the DKParts attribute
DKParts dkParts = (DKParts) ddoDocument.getData(ddoDocument.dataId(
    DKConstant.DK_CM_NAMESPACE_ATTR,DKConstant.DK_CM_DKPARTS));

// Add parts to document
dkParts.addElement(annot);

// Save the changes to the persistent datastore
ddoDocument.update();

// Check in / unlock the item after update
dsICM.checkIn(ddoDocument);
```

Para obter a aplicação exemplo completa, consulte SDocModelItemICM no directórioCMBROOT\Samples\java\icm.

C++

```
// Check out / lock the item for update
dsICM->checkOut(ddoDocument);

// Create annotation part
DKLobICM* annot = (DKLobICM*) dsICM->createDDO("ICMANNOATATION",
                                             DK_ICM_SEMANTIC_TYPE_ANNOTATION);

// Set annotatoin MIME type (SResourceItemMimeTypesICM.txt sample)
annot->setMimeType("image/bmp");

// Load content into parts (SResourceItemCreationICM sample)
annot->setContentFromClientFile("myAnnotation.bmp");

// Access the DKParts attribute
DKParts* dkParts = (DKParts*)(dkCollection*) ddoDocument->getData(
    ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKPARTS));

// Add parts to document
dkParts->addElement((dkDataObjectBase*)(DKDDO*)annot);

// Save the changes to the prsistent datastore
ddoDocument->update();

// Check in / unlock the item after update
dsICM->checkIn(ddoDocument);
```

Para obter a aplicação exemplo completa, consulte SDocModelItemICM no directórioCMBROOT\Samples\cpp\icm.

Exemplos de trabalho com um XDO

O exemplo seguinte ilustra a utilização de um XDO autónomo.

Obter, actualizar e eliminar um XDO

Para obter, actualizar ou eliminar um objecto num servidor de conteúdos, forneça o ID do artigo, o ID da parte e o TipoRep correctos que representam o objecto.

O exemplo seguinte obtém, actualiza e elimina um XDO no Content Manager.

Requisito: O tipo de artigo definido pelo utilizador S_text de recurso de classificação deve ser definido no sistema, tal como está definido no Exemplo de Instrução de API SItemTypeCreationICM. Para além disso, as definições do gestor de recursos e do conjunto de SMS devem ser configuradas e estabelecidas como predefinições do tipo de artigo ou do utilizador, tal como os exemplos SResourceMgrDefCreationICM, SSMSCollectionDefCreationICM, SResourceMgrDefSetDefaultICM e SSMSCollectionDefSetDefaultICM demonstram. Parta também do princípio que lhe é atribuída uma cadeia de PID na variável pidString para um artigo de recurso que já existe no servidor de conteúdos.

Java

```
// Given: String pidString

// Re-create Blank DDOs for Existing Item (SItemRetrievalICM sample)

DKLobICM lob = (DKLobICM) dsICM.createDDO(pidString);

// Retrieve the item with the resource content
lob.retrieve(DKConstant.DK_CM_CONTENT_YES);

// Check out / lock the item for update (SItemUpdateICM sample)
dsICM.checkOut(lob);

// Set the new MIME type (SResourceItemMimeTypesICM.txt sample)
lob.setMimeType("application/msword");

// Update datastore with new content
lob.update("SResourceItemICM_Document2.doc");

// Check in / unlock the item after update
dsICM.checkIn(lob);

// Delete item
lob.del();
```

Este exemplo de código foi retirado de SResourceItemRetrievalICM, SResourceItemUpdateICM e SResourceItemDeletionICM no directório CMBROOT\Samples\java\icm.

C++

```
// Given: DKString pidString

// Re-create Blank DDOs for Existing Item (SItemRetrievalICM sample)
DKLobICM* lob = (DKLobICM*) dsICM->createDDO(pidString);

// Retrieve the item with the resource content
lob->retrieve(DK_CM_CONTENT_YES);

// Check out / lock the item for update (SItemUpdateICM sample)
dsICM->checkOut(lob);

// Set the new MIME type (SResourceItemMimeTypesICM.txt sample)
lob->setMimeType("application/msword");

// Update datastore with new content
lob->update("SResourceItemICM_Document2.doc");

// Check in / unlock the item after update
dsICM->checkIn(lob);

// Delete item
lob->del();

// Free Memory
delete(lob);
```

Este exemplo de código foi retirado de `SResourceItemRetrievalICM`, `SResourceItemUpdateICM` e `SResourceItemDeletionICM` no directório `CMBROOT\Samples\cpp\icm`.

Invocar uma função de XDO

Este exemplo demonstra como deve testar a classe de `DKBlob` através de um servidor do Content Manager anterior. Para este exemplo, terá de saber o ID do artigo e o ID da parte do XDO.

Java

```
public class txdomiscDL implements DKConstantDL
{
    public static void main(String args[])
    {
        int    partId = 5;
        String itemId = "GAWCVGGVFUG428UJ";
        String repType = "";
        // Check the number of arguments for main and determine what to do
        if (args.length == 3)
        {
            partId = (short)Integer.parseInt(args[0], 10);
            repType = args[1];
            itemId = args[2];
            System.out.println("Insira: java txdomiscDL " +
                + partId + " " + repType + " " + itemId);
        }
        if (args.length == 2)
        {
            partId = (short)Integer.parseInt(args[0], 10);
            repType = args[1];
            System.out.println("Insira: java txdomiscDL " +
                + partId + " " + repType);
        }
        if (args.length == 1)
        {
            partId =(short)Integer.parseInt(args[0], 10);
            System.out.println("Insira: java txdomiscDL " + partId );
            System.out.println("0 tipoRep predefinido fornecido = " + repType);
            System.out.println("0 Idartigo predefinido fornecido = " + itemId);
        }
        if (args.length == 0)
        {
            System.out.println("invocar: java txdomiscDL ");
            System.out.println("No parameter, following defaults provided:");
            System.out.println("    partId predefinido = " + partId);
            System.out.println("    repType predefinido = " + repType);
            System.out.println("    default itemId = " + itemId);
        }

        try
        {
            DKDatastoreDL dsDL = new DKDatastoreDL();
            System.out.println("a ligar ao armazenamento de dados");
            dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");
            System.out.println("armazenamento de dados ligado");

            DKBlobDL axdo = new DKBlobDL(dsDL);
            DKPidXDODL apid = new DKPidXDODL();
            apid.setPartId(partId);
            apid.setPrimaryId(itemId);
            apid.setRepType(repType);
            axdo.setPidObject(apid);
            System.out.println("repType=" + apid.getRepType());
            System.out.println("itemid=" + apid.getItemId());
            System.out.println("partId=" + apid.getPartId());
        }
        // continued...
    }
}
```

Java (continuação)

```
// ----- Before retrieve
System.out.println("before retrieve:");
System.out.println("  contentclass=" + axdo.getContentClass());
System.out.print("  content length=" + axdo.length());
System.out.println(" (the length of this object instance - in memory)");
System.out.print("  getSize=" + axdo.getSize());
System.out.println(" (get the object size without retrieving object)");
System.out.println("  createdTimestamp=" + axdo.getCreatedTimestamp());
System.out.println("  updatedTimestamp=" + axdo.getUpdatedTimestamp());
axdo.retrieve();

// ----- After retrieve
System.out.println("after retrieve:");
System.out.println("  contentclass=" + axdo.getContentClass());
System.out.print("  content length=" + axdo.length());
System.out.println(" (the length of this object instance - in memory)");
System.out.print("  getSize=" + axdo.getSize());
System.out.println(" (get the object size without retrieving object)");
System.out.println("  createdTimestamp=" + axdo.getCreatedTimestamp());
System.out.println("  updatedTimestamp=" + axdo.getUpdatedTimestamp());
System.out.println("  affiliatedTyp=" + axdo.getAffiliatedType());
if (axdo.getAffiliatedType() == DK_DL_ANNOTATION)
{
    DKAnnotationDL ann =
(DKAnnotationDL)(axdo.getExtension("DKAnnotationDL"));
    System.out.println("affil pageNumber=" + ann.getPageNumber());
    System.out.println("affil X=" + ann.getX());
    System.out.println("affil Y=" + ann.getY());
}
System.out.println("about to do open()...");
axdo.setInstanceOpenHandler("notepad", true);
int cc = axdo.getContentClass();
if ( cc == DK_DL_CC_GIF)
    axdo.setInstanceOpenHandler("lviewpro", true);
else if (cc == DK_DL_CC_ASCII)
    axdo.setInstanceOpenHandler("notepad", true);
else if (cc == DK_DL_CC_AVI)
    axdo.setInstanceOpenHandler("mplay32 ", true);
axdo.open();
dsDL.disconnect();
    dsDL.destroy();
}
catch (DKException exc)
{
}
// ----- Handle the exceptions
}
```

C++

```
void main(int argc, char *argv[])
{
    DKDatastoreDL dsDL;
    long hsession;
    DKString itemId, repType;
    int partId;
    itemId = "GAWCVGGVFUG428UJ";
    repType = "FRN$NULL";
    partId = 2;

    cout << "argc is " << argc << endl;
    if (argc == 1)
    {
        cout << "invoke: txdomisc <partId> <repType> <itemId>" << endl;
        cout << " no parameter, following default will be provided:" << endl;
        cout << "The supplied default partId = " << partId << endl;
        cout << "The supplied default repType = " << repType << endl;
        cout << "The supplied default itemId = " << itemId << endl;
    }
    else if (argc == 2)
    {
        partId = atoi(argv[1]);
        cout << "you enter: txdomisc " << argv[1] << endl;
        cout << "The supplied default repType = " << repType << endl;
        cout << "The supplied default itemId = " << itemId << endl;
    }
    else if (argc == 3)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        cout << "you enter: txdomisc " << argv[1] << " " << argv[2] << endl;
        cout << "The supplied default itemId = " << itemId << endl;
    }
    else if (argc == 4)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        itemId = DKString(argv[3]);
        cout << "you enter: txdomisc " << argv[1] << " " << argv[2] << " " << argv[3] << endl;
    }

    cout << "connecting Datastore" << endl;
    try
    {
        dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
        cout << "datastore connected" << endl;
        hsession = (long) (dsDL.connection()->handle());
        cout << "datastore handle" << hsession << endl;
    }
    // continued...
```

C++ (continuação)

```
DKBlobDL* axdo = new DKBlobDL(&dsDL);
DKPidXDODL* apid = new DKPidXDODL;
apid ->setPartId(partId);
apid ->setPrimaryId(itemId);
apid ->setRepType(repType);
axdo ->setPidObject(apid);
cout<<"itemId= "<<axdo->getItemId()<<endl;
cout<<"partId= "<<((DKPidXDODL*) (axdo->getPidObject()))
->getPartId()<<endl;
cout<<"repType= "<<axdo->getRepType()<<endl;

//== before retrieve
cout<<"before retrieve:"<<endl;
cout<<" content class="<<axdo->getContentClass()<<endl;
cout<<" content length="<<axdo->length();
cout<<" (the length of this object instance - in memory)"<<endl;
cout<<" getSize="<<axdo->getSize();
cout<<" (get the object size without retrieving object)"<<endl;
cout<<" createdTimestamp="<<axdo->getCreatedTimestamp()<<endl;
cout<<" updatedTimestamp="<<axdo->getUpdatedTimestamp()<<endl;
axdo->retrieve();

//== after retrieve
cout<<"after retrieve:"<<endl;
cout<<" content class="<<axdo->getContentClass()<<endl;
cout<<" content length="<<axdo->length();
cout<<" (the length of this object instance - in memory)"<<endl;
cout<<" getSize="<<axdo->getSize();
cout<<" (get the object size without retrieving object)"<<endl;
cout<<" createdTimestamp="<<axdo->getCreatedTimestamp()<<endl;
cout<<" updatedTimestamp="<<axdo->getUpdatedTimestamp()<<endl;
cout<<" mimeType="<<axdo->getMimeType()<<endl;
int atype = axdo->getAffiliatedType();
cout<<" affiliatedType= "<<axdo->getAffiliatedType()<<endl;
if (atype == DK_DL_ANNOTATION)
{
    DKAnnotationDL* ann=(DKAnnotationDL*)axdo
    ->getExtension("DKAnnotationDL");
    cout <<" pageNumber= "<<ann->getPageNumber()<<endl;
    cout <<" partId= "<<ann->getPart()<<endl;
    cout <<" X="<<ann->getX()<<endl;
    cout <<" Y="<<ann->getY()<<endl;
}
//== open content
int concls = axdo->getContentClass();
if (concls == DK_DL_CC_ASCII)
    axdo->setInstanceOpenHandler("notepad", TRUE);
else if (concls == DK_DL_CC_GIF)
    axdo->setInstanceOpenHandler("lviewpro", TRUE);
else if (concls == DK_DL_CC_AVI)
    axdo->setInstanceOpenHandler("mplay32", TRUE);
axdo->open();
// continued...
```

C++ (continuação)

```
        delete apid;
        delete axdo;
        dsDL.disconnect();
        cout<<"datastore disconnected"<<endl;
    }
    catch(DKException &exc)
    {
        cout << "Error id" << exc.errorId() << endl;
        cout << "Exception id " << exc.exceptionId() << endl;
        for(unsigned long i=0;i< exc.textCount();i++)
        {
            cout << "Error text:" << exc.text(i) << endl;
        }
        for (unsigned long g=0;g< exc.locationCount();g++)
        {
            const DKExceptionLocation* p = exc.locationAtIndex(g);
            cout << "Filename: " << p->fileName() << endl;
            cout << "Function: " << p->functionName() << endl;
            cout << "LineNumber: " << p->lineNumber() << endl;
        }
        cout << "Exception Class Name: " << exc.name() << endl;
    }
    cout << "done ..." << endl;
}
```

Adicionar um objecto de suporte de dados de XDO em versões anteriores doContent Manager

Para cada objecto de suporte adicionado, é criada uma entrada na tabela FRN\$MEDIA. Esta entrada contém as informações sobre os dados do utilizador do suporte. O objecto de suporte físico é armazenado no servidor de conteúdos de VideoCharger especificado na tabela da rede. Para o exemplo seguinte, terá de saber o ID do artigo do XDO.

Java

```
public class txdoAddVSDL implements DKConstantDL
{
    // ----- Main method
    public static void main(String[] args)
    {
        String fileName = "/icing1.mpg1";           //a media object
        String itemId = "K1A04EWBVHJAV1D7";        //a known itemId
        int partId = 45;
        // ----- Check the arguments for main
        if (args.length == 3)
        {
            fileName = args[0];
            partId = (int)Integer.parseInt(args[1], 10);
            itemId = args[2];
            System.out.println("You enter: java txdoAddVSDL " +
                fileName + " " + partId + " " + itemId);
        }
        if (args.length == 2)
        {
            fileName = args[0];
            partId = (int)Integer.parseInt(args[1], 10);
            System.out.println("You enter: java txdoAddVSDL " +
                fileName + " " + partId );
            System.out.println("The supplied default itemId = " + itemId);
        }
        if (args.length == 1)
        {
            fileName = args[0];
            System.out.println("You enter: java txdoAddVSDL " + fileName);
            System.out.println("The supplied default partId = " + partId);
            System.out.println("The supplied default itemId = " + itemId);
        }
        if (args.length == 0)
        {
            System.out.println("invoke: java txdoAddVSDL <filename> <part ID> <item ID>");
            System.out.println("Sem parâm., as predefinições seguintes serão fornecidas:");
            System.out.println("    default fileName = " + fileName);
            System.out.println("    default partId = " + partId);
            System.out.println("    default itemId = " + itemId);
        }
        // ----- Processing
        try
        {
            // ----- connect to datastore
            DKDatastoreDL dsDL = new DKDatastoreDL();
            // replace following with your library server, userid, password
            System.out.println("connecting to datastore...");
            dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");
            System.out.println("datastore connected");
            // ----- create xdo and pid
            DKBlobDL axdo = new DKBlobDL(dsDL);
            DKPidXDODL apid = new DKPidXDODL();
            apid.setPartId(partId);
            apid.setPrimaryId(itemId);
            axdo.setPidObject(apid);
            // you must use the content class DK_DL_CC_IBMVSS for a media object
            axdo.setContentClass(DK_DL_CC_IBMVSS);
            System.out.println("contentClass=" + axdo.getContentClass());
            System.out.println("partId = " + axdo.getPartId());
        }
        // continued...
    }
}
```

Java (continuação)

```
// ----- configurar DKMediaStreamInfoDL
DKMediaStreamInfoDL aVS = new DKMediaStreamInfoDL();
aVS.setMediaFullFileName(fileName);
// if fileName contain a list of media segments then use following
//      aVS.setMediaObjectOption(DK_VS_LIST_OF_OBJECT_SEGMENTS);
aVS.setMediaObjectOption(DK_DL_VS_SINGLE_OBJECT);
aVS.setMediaHostName("<insert hostname here>");
aVS.setMediaUserId("<insert user ID here>");
aVS.setMediaPassword("<insert password here>");
// following are optional, if not set default value will be provided
aVS.setMediaNumberOfUsers(2);
aVS.setMediaAssetGroup("AG");
// ----- same as defined in VideoCharger server
aVS.setMediaType("MPEG1");
aVS.setMediaResolution("SIF");
aVS.setMediaStandard("NTSC");
aVS.setMediaFormat("SYSTEM");
axdo.setExtension("DKMediaStreamInfoDL", (dkExtension)aVS);
System.out.println("about to call add()");
axdo.add();
System.out.println("add successfully....");
System.out.println("after added check for status:");
boolean flag2 = axdo.isCategoryOf(DK_DL_MEDIA_OBJECT);
if (flag2)
{
    DKMediaStreamInfoDL media = (DKMediaStreamInfoDL)
        axdo.getExtension("DKMediaStreamInfoDL");
    System.out.println(" mediaformat=" + media.getMediaFormat());
    System.out.println(" mediaBitRate=" + media.getMediaBitRate());
    System.out.println(" mediastate(dynamic)=" +
        axdo.retrieveObjectState(DK_MEDIA_OBJECT));
}
dsDL.disconnect();
dsDL.destroy();
}
catch (DKException exc) {
    try {
        dsDL.destroy();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("Exception name " + exc.name());
    System.out.println("Exception message " + exc.getMessage());
    exc.printStackTrace();
}
catch (Exception exc){
    try {
        dsDL.destroy();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("Exception message " + exc.getMessage());
    exc.printStackTrace();
}
}
}
```

C++

```
void main(int argc, char *argv[])
{
    DKString itemId, fileName;
    int partId;
    itemId = "K1A04EWBVHJAV1D7";
    partId = 22;
    fileName = "/icing1.mpg1";
    if (argc == 1)
    {
        cout<<"invoke: txdoAddVSDL <fileName> <partId> <itemId>"<<endl;
        cout<<" no parameter, following default will be provided:"<<endl;
        cout<<"The supplied default fileName = "<<fileName<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 2)
    {
        fileName = DKString(argv[1]);
        cout<<"you enter: txdoAddVSDL "<<argv[1]<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"O IDartigo da predefinição fornecida = "<<itemId<<endl;
    }
    else if (argc == 3)
    {
        fileName = DKString(argv[1]);
        partId = atoi(argv[2]);
        cout<<"you enter: txdoAddVSDL "<<argv[1]<<" "<<argv[2]<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 4)
    {
        fileName = DKString(argv[1]);
        partId = atoi(argv[2]);
        itemId = DKString(argv[3]);
        cout<<"enter: txdoAddVSDL "<<argv[1]<<" "<<argv[2]<<" "<<argv[3]<<endl;
    }
    try
    {
        // connect to datastore
        cout << "Connecting datastore ..." << endl;
        DKDatastoreDL dsDL;
        // substituir o seguinte pelo servidor de bibliotecas, id utilizador,
        // palavra-passe
        dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");
        cout << "datastore connected" << endl;
        // *** create xdo and pid
        DKBlobDL* axdo = new DKBlobDL(&dsDL);
        DKPidXDODL* apid = new DKPidXDODL;
        apid ->setPartId(partId);
        apid ->setPrimaryId(itemId);
        axdo ->setPidObject(apid);
        // you must use the content class DK_DL_CC_IBMVSS for a media object
        axdo ->setContentClass(DK_DL_CC_IBMVSS);
        cout <<"itemId= "<<axdo->getItemId()<<endl;
        cout <<"partId= "<<axdo->getPartId()<<endl;
        cout <<"repType= "<<axdo->getRepType()<<endl;
        cout <<"content class="<< axdo->getContentClass()<<endl;
    }
    // continued...
```

C++ (continuação)

```
// *** setup DKMediaStreamInfoDL
DKMediaStreamInfoDL aVS;
aVS.setMediaFullFileName(fileName);
aVS.setMediaObjectOption(DK_DL_VS_SINGLE_OBJECT);
aVS.setMediaHostName("<insert hostname here>");
aVS.setMediaUserId("<insert user ID here>");
aVS.setMediaPassword("<insert password here>");

//following are optional, if not set then default value will be provided
aVS.setMediaNumberOfUsers(1);
aVS.setMediaAssetGroup("AG");
// *** same as defined in VideoCharger server
aVS.setMediaType("MPEG1");
aVS.setMediaResolution("SIF");
aVS.setMediaStandard("NTSC");
aVS.setMediaFormat("SYSTEM");

axdo ->setExtension("DKMediaStreamInfoDL", (dkExtension*)&aVS);
cout <<"about to do add()"<<endl;
axdo ->add();
cout<<"Object added successfully "<<endl;

cout<<"after added check for status:"<<endl;
DKBoolean flag2 = axdo->isCategoryOf(DK_DL_MEDIA_OBJECT);
if (flag2)
{
    DKMediaStreamInfoDL* mediaInfo = (DKMediaStreamInfoDL*)
        axdo->getExtension("DKMediaStreamInfoDL");
    cout<<" copyRate="<<mediaInfo->getMediaCopyRate()<<endl;
    cout<<" mediaType="<<mediaInfo->getMediaType()<<endl;
    cout<<" mediaFrameRate="<<mediaInfo->getMediaFrameRate()<<endl;
    cout<<" mediaState="<<mediaInfo->getMediaState()<<endl;
    cout<<" mediaTimestamp="<<mediaInfo->getMediaTimestamp()<<endl;
    cout<<" MediaState(dynamic)="<<
        axdo->retrieveObjectState(DK_MEDIA_OBJECT)<<endl;
}
dsDL.disconnect();
cout<<"datastore disconnected"<<endl;
}
catch(DKException &exc)
{
    cout << "Error id" << exc.errorId() << endl;
    cout << "Exception id " << exc.exceptionId() << endl;
    for(unsigned long i=0;i< exc.textCount();i++)
    {
        cout << "Error text:" << exc.text(i) << endl;
    }
    for (unsigned long g=0;g< exc.locationCount();g++)
    {
        const DKExceptionLocation* p = exc.locationAtIndex(g);
        cout << "Filename: " << p->fileName() << endl;
        cout << "Function: " << p->functionName() << endl;
        cout << "LineNumber: " << p->lineNumber() << endl;
    }
    cout << "Exception Class Name: " << exc.name() << endl;
}
cout << "done ..." << endl;
}
```

Eliminar um objecto de suporte do XDO

O exemplo seguinte demonstra como eliminar um objecto de suporte do XDO.

Para este exemplo terá de conhecer o ID do artigo, o ID da parte e o TipoRep (tipo de representação) do XDO.

Java

```
public class txdoDelVSDL implements DKConstantDL
{
    public static void main(String args[])
    {
        int    partId = 45;
        String repType = "";
        String itemId = "K1A04EWBVHJAV1D7";
        if (args.length == 3)
        {
            partId = (short)Integer.parseInt(args[0], 10);
            repType = args[1];
            itemId = args[2];
            System.out.println("Insira: java txdoDelVSDL " +
                + partId + " " + repType + " " + itemId);
        }
        // ----- Verificar os argumentos para principal
        if (args.length == 2)
        {
            partId = (short)Integer.parseInt(args[0], 10);
            repType = args[1];
            System.out.println("Insira: java txdoDelVSDL " +
                + partId + " " + repType);
        }

        if (args.length == 1)
        {
            partId =(short)Integer.parseInt(args[0], 10);
            System.out.println("Insira: java txdoDelVSDL " + partId );
            System.out.println("0 tipoRep predefinido fornecido = " + repType);
            System.out.println("0 Idartigo predefinido fornecido = " + itemId);
        }
        if (args.length == 0)
        {
            System.out.println("invocar: java txdoDelVSDL <part ID> <RepType> <
                item ID>");
            System.out.println("Sem parâmetro, as predefinições seguintes serão
                fornecidas:");
            System.out.println("    partId predefinido = " + partId);
            System.out.println("    repType predefinido = " + repType);
            System.out.println("    default itemId = " + itemId);
        }

        // ----- Processar
        try
        {
            DKDatastoreDL dsDL = new DKDatastoreDL();
            System.out.println("a ligar ao armazenamento de dados...");
            // substituir o seguinte pelo servidor de bibliotecas, id de utilizador,
            // palavra-passe
            dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");
            System.out.println("armazenamento de dados ligado");

            DKBlobDL axdo = new DKBlobDL(dsDL);
            DKPidXDODL apid = new DKPidXDODL();
            apid.setPartId(partId);
            apid.setPrimaryId(itemId);
            apid.setRepType(repType);
            axdo.setPidObject(apid);
            boolean flag2 = axdo.isCategoryOf(DK_DL_MEDIA_OBJECT);
            System.out.println("isMediaObject?="+ flag2);
        }
        // continued...
    }
}
```

Java (continuação)

```
if (flag2)
{
    DKMediaStreamInfoDL media = (DKMediaStreamInfoDL)
        axdo.getExtension("DKMediaStreamInfoDL");
    System.out.println(" mediaformat=" + media.getMediaFormat());
    System.out.println(" mediaBitRate=" + media.getMediaBitRate());
    System.out.println(" mediastate(dynamic)=" +
        axdo.retrieveObjectState(DK_MEDIA_OBJECT));
    // ----- set delete option for media object
    axdo.setOption(DK_DL_OPT_DELETE_OPTION,
        (Object)new Integer(DK_DL_DELETE_NO_DROPITEM_MEDIA_AVAIL));
    System.out.println("The delete option =" +
        (Integer)(axdo.getOption(DK_OPT_DL_DELETE_OPTION)));
}

System.out.println("about to call del().. ");
axdo.del();
System.out.println("del successfully....");
flag2 = axdo.isCategoryOf(DK_DL_MEDIA_OBJECT);
System.out.println("after delete isMediaObject? = " + flag2);
System.out.println("about to call dsDL.disconnect()");
dsDL.disconnect();
dsDL.destroy();
}
// ----- Handle exceptions
catch (DKException exc) {
    try {
        dsDL.destroy();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("Exception name " + exc.name());
    System.out.println("Exception message " + exc.getMessage());
    exc.printStackTrace();
}
catch (Exception exc){
    try {
        dsDL.destroy();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("Exception message " + exc.getMessage());
    exc.printStackTrace();
}
}
```

C++

```
void main(int argc, char *argv[])
{
    DKDatastoreDL dsDL;
    DKString itemId, repType;
    int partId;
    itemId = "Y68M1I@VYDG8SPQ4";
    partId = 1;
    repType = "FRN$NULL";
    if (argc == 1)
    {
        cout<<"invoke: txdoDelVSDL <partId> <repType> <itemId>"<<endl;
        cout<<" no parameter, following default will be provided:"<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 2)
    {
        partId = atoi(argv[1]);
        cout<<"you enter: txdoDelVSDL "<<argv[1]<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 3)
    {
        repType = DKString(argv[2]);
        partId = atoi(argv[1]);
        cout<<"you enter: txdoDelVSDL "<<argv[1]<<" "<<argv[2]<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 4)
    {
        itemId = DKString(argv[3]);
        repType = DKString(argv[2]);
        partId = atoi(argv[1]);
        cout<<"you enter: txdoDelVSDL "<<argv[1]<<" "<<argv[2]<<" "
            <<argv[3]<<endl;
    }

    try
    {
        cout << "Connecting datastore ..." << endl;
        // substituir o seguinte pelo servidor de bibliotecas, id utilizador,
        // palavra-passe
        dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
        cout << "datastore connected" << endl;

        DKBlobDL* axdo = new DKBlobDL(&dsDL);
        DKPidXDODL* apid = new DKPidXDODL;
        apid ->setPartId(partId);
        apid ->setPrimaryId(itemId);
        apid ->setRepType(repType);
        axdo ->setPidObject(apid);
        cout <<"itemId= "<<axdo->getItemId()<<endl;
        cout <<"partId= "<<((DKPidXDODL*)(axdo->getPidObject()))
            ->getPartId()<<endl;
        DKBoolean flag2 = axdo->isCategoryOf(DK_DL_MEDIA_OBJECT);
        cout <<"isMediaObject? = "<<flag2<<endl;
    }
    // continued...
```


C++ (continuação)

```
        if (flag2)
        {
            DKMediaStreamInfoDL* mediaInfo = (DKMediaStreamInfoDL*)
                axdo->getExtension("DKMediaStreamInfoDL");
            cout<<" copyRate="<<mediaInfo->getMediaCopyRate()<<endl;
            cout<<" mediaType="<<mediaInfo->getMediaType()<<endl;
            cout<<" mediaFrameRate="<<mediaInfo->getMediaFrameRate()<<endl;
            cout<<" mediaState="<<mediaInfo->getMediaState()<<endl;
            cout<<" mediaTimestamp="<<mediaInfo->getMediaTimestamp()<<endl;
            cout<<" MediaState(dynamic)=
                "<<axdo->retrieveObjectState(DK_MEDIA_OBJECT)<<endl;

            cout<<"about to set the delete option for media object..."<<endl;
            DKAny delOpt = DK_DL_DELETE_NO_DROPITEM_MEDIA_AVAIL;
            axdo->setOption(DK_DL_OPT_DELETE_OPTION, delOpt);
            DKAny opt;
            axdo->getOption(DK_DL_OPT_DELETE_OPTION, opt);
            long lopt = opt;
            cout<<"The setted delete option = "<<lopt<<endl;

        }
        cout<<"about to do del()"<<endl;
        axdo->del();
        cout<<"del successfully..."<<endl;
        flag2 = axdo->isCategoryOf(DK_DL_MEDIA_OBJECT);
        cout<<"after delete isMediaObject? = "<<flag2<<endl;
        delete axdo;
        delete apid;
        dsDL.disconnect();
        cout<<"datastore disconnected"<<endl;
    }
    catch(DKException &exc)
    {
        cout << "Error id" << exc.errorId() << endl;
        cout << "Exception id " << exc.exceptionId() << endl;
        for(unsigned long i=0;i< exc.textCount();i++)
        {
            cout << "Error text:" << exc.text(i) << endl;
        }
        for (unsigned long g=0;g< exc.locationCount();g++)
        {
            const DKExceptionLocation* p = exc.locationAtIndex(g);
            cout << "Filename: " << p->fileName() << endl;
            cout << "Function: " << p->functionName() << endl;
            cout << "LineNumber: " << p->lineNumber() << endl;
        }
        cout << "Exception Class Name: " << exc.name() << endl;
    }
    cout << "done ..." << endl;
}
```

Obter um objecto de suporte do XDO

O exemplo seguinte demonstra como obter um objecto de suporte do XDO. O objecto obtido contém apenas os metadados de suporte, não o próprio objecto de suporte. Para este exemplo terá de saber o ID do artigo e o ID da parte do XDO.

Java

```
public class txdoretxsDL implements DKConstantDL
{
    public static void main(String args[])
    {
        int    partId = 45;
        String itemId = "K1A04EWBVHJAV1D7";
        String repType = "";
        System.out.println("Processar através dos valores seguintes: ");
        System.out.println("    Part Id = " + partId);
        System.out.println("    RepType = " + repType);
        System.out.println("    Item Id = " + itemId);
        try
        {
            DKDatastoreDL dsDL = new DKDatastoreDL();
            System.out.println("a ligar ao armazenamento de dados...");
            // ----- substituir o seguinte pelo seu servidor de bibliotecas,
            // idutilizador, palavra-passe
            dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");
            System.out.println("armazenamento de dados ligado");
            DKBlobDL axdo = new DKBlobDL(dsDL);
            DKPidXDODL apid = new DKPidXDODL();
            apid.setPartId(partId);
            apid.setPrimaryId(itemId);
            apid.setRepType(repType);
            axdo.setPidObject(apid);
            System.out.println("repType=" + apid.getRepType());
            System.out.println("objectType=" + axdo.getObjectType());
            System.out.println("itemid=" + apid.getItemId());
            System.out.println("partId=" + apid.getPartId());

            boolean flag = axdo.isCategoryOf(DK_DL_INDEXED_OBJECT);
            boolean flag2 = axdo.isCategoryOf(DK_DL_MEDIA_OBJECT);
            System.out.println("isIndexedObject?" + flag);
            System.out.println("isMediaObject?" + flag2);
            if (flag)
            {
                DKSearchEngineInfoDL srch = (DKSearchEngineInfoDL)
                    axdo.getExtension("DKSearchEngineInfoDL");
                System.out.println("    serverName=" + srch.getServerName());
                System.out.println("    textIndex=" + srch.getTextIndex());
                System.out.println("    timeStamp=" + srch.getSearchTimestamp());
                System.out.println("    searchIndex=" + srch.getSearchIndex());
                System.out.println("    indexedState=" +
                    axdo.retrieveObjectState(DK_INDEXED_OBJECT));
            }
        }
    }
}
```

Java (continuação)

```
if (flag2)
{
    DKMediaStreamInfoDL media = (DKMediaStreamInfoDL)
        axdo.getExtension("DKMediaStreamInfoDL");
    System.out.println(" mediaformat=" + media.getMediaFormat());
    System.out.println(" mediaBitRate=" + media.getMediaBitRate());
    System.out.println(" mediastate(dynamic)=" +
        axdo.retrieveObjectState(DK_MEDIA_OBJECT));
}
System.out.println("before retrieve.....");
System.out.println(" lob length=" + axdo.length());
System.out.println(" size=" + axdo.getSize());
System.out.println(" createdTimestamp="+axdo.getCreatedTimestamp());
System.out.println(" updatedTimestamp="+axdo.getUpdatedTimestamp());
// ----- Perform the retrieve call
axdo.retrieve();

System.out.println("after retrieve.....");
System.out.println(" lob length=" + axdo.length());
System.out.println(" size=" + axdo.getSize());
System.out.println(" mimeType=" + axdo.getMimeType());
System.out.println(" createdTimestamp=" + axdo.getCreatedTimestamp());
System.out.println(" updatedTimestamp=" + axdo.getUpdatedTimestamp());
System.out.println("affiliatedType=" + axdo.getAffiliatedType());
if (axdo.getAffiliatedType() == DK_DL_ANNOTATION)
{
    DKAnnotationDL ann =
        (DKAnnotationDL)(axdo.getExtension("DKAnnotationDL"));
    System.out.println("affil pageNumber=" + ann.getPageNumber());
    System.out.println("affil X=" + ann.getX());
    System.out.println("affil Y=" + ann.getY());
}
System.out.println("about to do open()...");
axdo.setInstanceOpenHandler("notepad", true); //default for Windows
int cc = axdo.getContentClass();
if ( cc == DK_DL_CC_GIF)
    axdo.setInstanceOpenHandler("lviewpro ", true); //use lviewpro
else if (cc == DK_DL_CC_AVI)
    axdo.setInstanceOpenHandler("mplay32 ", true); //use mplay32
else if (cc == DK_DL_CC_IBMVSS)
    axdo.setInstanceOpenHandler("iscoview ", true); //use iscoview
axdo.open();

dsDL.disconnect();
dsDL.destroy();
}
catch (DKException exc)
{
    ... \\ handle exceptions and destroy the datastore+
}
}
```

C++

```
void main(int argc, char *argv[])
{
    DKDatastoreDL dsDL;
    DKString itemId, repType;
    int partId;
    itemId = "K1A04EWBVBHJAV1D7";
    partId = 1;
    repType = "FRN$NULL";
    if (argc == 1)
    {
        cout<<"invoke: txdoRetxsDL <partId> <repType> <itemId>"<<endl;
        cout<<" no parameter, following default will be provided:"<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 2)
    {
        partId = atoi(argv[1]);
        cout<<"you enter: txdoRetxsDL "<<argv[1]<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 3)
    {
        repType = DKString(argv[2]);
        partId = atoi(argv[1]);
        cout<<"you enter: txdoRetxsDL "<<argv[1]<<" "<<argv[2]<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 4)
    {
        itemId = DKString(argv[3]);
        repType = DKString(argv[2]);
        partId = atoi(argv[1]);
        cout<<"you enter: txdoRetxsDL "<<argv[1]
            <<" "<<argv[2]<<" "<<argv[3]<<endl;
    }
    try
    {
        cout << "Connecting datastore ..." << endl;
        // replace following with your library server, userid, password
        dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");
        cout << "datastore connected" << endl;

        DKBlobDL* axdo = new DKBlobDL(&dsDL);
        DKPidXDODL* apid = new DKPidXDODL;
        apid ->setPartId(partId);
        apid ->setPrimaryId(itemId);
        apid ->setRepType(repType);
        axdo ->setPidObject(apid);
        cout <<"itemId= "<<axdo->getItemId()<<endl;
        cout <<"partId= "
            <<((DKPidXDODL*)(axdo->getPidObject()))->getPartId()<<endl;
    }
    // continued...
```

C++ (continuação)

```
        DKBoolean flag = axdo->isCategoryOf(DK_DL_INDEXED_OBJECT);
        DKBoolean flag2 = axdo->isCategoryOf(DK_DL_MEDIA_OBJECT);
        cout <<"isIndexed? = "<<flag<<endl;
        cout <<"isMediaObject? = "<<flag2<<endl;
        if (flag)
        {
            DKSearchEngineInfoDL* srchInfo = (DKSearchEngineInfoDL*)
                axdo->getExtension("DKSearchEngineInfoDL");
            cout<<" ServerName="<<srchInfo->getServerName()<<endl;
            cout<<" TextIndex="<<srchInfo->getTextIndex()<<endl;
            cout<<" srchEngine="<<srchInfo->getSearchEngine()<<endl;
            cout<<" srchIndex="<<srchInfo->getSearchIndex()<<endl;
            cout<<" indexedState="<<axdo
                ->retrieveObjectState(DK_DL_INDEXED_OBJECT)<<endl;
        }

        if (flag2)
        {
            DKMediaStreamInfoDL* mediaInfo = (DKMediaStreamInfoDL*)
                axdo->getExtension("DKMediaStreamInfoDL");
            cout<<" copyRate="<<mediaInfo->getMediaCopyRate()<<endl;
            cout<<" mediaType="<<mediaInfo->getMediaType()<<endl;
            cout<<" mediaFrameRate="<<mediaInfo->getMediaFrameRate()<<endl;
            cout<<" mediaState="<<mediaInfo->getMediaState()<<endl;
            cout<<" mediaTimestamp="<<mediaInfo->getMediaTimestamp()<<endl;
            cout<<" MediaState(dynamic)= "
                <<axdo->retrieveObjectState(DK_DL_MEDIA_OBJECT)<<endl;
        }

        cout<<"before retrieve..."<<endl;
        cout <<" length of lobdata = "<<axdo->length()<<endl;
        cout<<" size of lobdata = "<<axdo->getSize()<<endl;
        cout<<" created Timestamp = "<<axdo->getCreatedTimestamp()<<endl;
        cout<<" updated Timestamp = "<<axdo->getUpdatedTimestamp()<<endl;
        axdo->retrieve();
        cout<<"after retrieve..."<<endl;
        cout <<" length of lobdata = "<<axdo->length()<<endl;
        cout <<" mimeType = "<<axdo->getMimeType()<<endl;
        cout <<" size of lobdata = "<<axdo->getSize()<<endl;
        cout<<" created Timestamp = "<<axdo->getCreatedTimestamp()<<endl;
        cout<<" updated Timestamp = "<<axdo->getUpdatedTimestamp()<
            <endl;
        // continued...
```

C++ (continuação)

```
int atype = axdo->getAffiliatedType();
cout <<"affiliatedType= "<<axdo->getAffiliatedType()<<endl;
if (atype == DK_ANNOTATION)
{
    DKAnnotationDL* ann =
        (DKAnnotationDL*)axdo->getExtension("DKAnnotationDL");
    cout<<"  pageNumber= "<<ann->getPageNumber()<<endl;
    cout<<"  partId= "<<ann->getPart()<<endl;
    cout<<"  X= "<<ann->getX()<<endl;
    cout<<"  Y= "<<ann->getY()<<endl;
}
cout<<"about to do open()..."<<endl;
axdo->setInstanceOpenHandler("notepad", TRUE);
//default use Notepad in Windows
int concls = axdo->getContentClass();
if (concls == DK_DL_CC_GIF)
    axdo->setInstanceOpenHandler("lviewpro", TRUE);
//use lviewpro in Windows
else if (concls == DK_DL_CC_AVI)
    axdo->setInstanceOpenHandler("mplay32", TRUE);
//use mplay32 in Windows
else if (concls == DK_DL_CC_IBMVSS)
    axdo->setInstanceOpenHandler("iscoview", TRUE);
//use iscoview in Windows
axdo->open();

delete axdo;
delete apid;
dsDL.disconnect();
cout<<"datastore disconnected"<<endl;
}
catch(DKException &exc)
{
    cout << "Error id" << exc.errorId() << endl;
    cout << "Id exceção" << exc.exceptionId() << endl;
    for(unsigned long i=0;i< exc.textCount();i++)
    {
        cout << "Error text:" << exc.text(i) << endl;
    }
    for (unsigned long g=0;g< exc.locationCount();g++)
    {
        const DKExceptionLocation* p = exc.locationAtIndex(g);
        cout << "Filename: " << p->fileName() << endl;
        cout << "Function: " << p->functionName() << endl;
        cout << "LineNumber: " << p->lineNumber() << endl;
    }
    cout<<"Exception Class Name: "<<exc.name()<<endl;
}
cout << "done ..." << endl;
}
```

Adicionar um XDO a uma recolha de memória

Para adicionar um objecto de XDO associado aos nomes de conjunto de memória definidos, utilize o objecto da extensão `DKStorageManageInfoxx`, onde `xx` é o sufixo que representa o servidor específico.

O exemplo seguinte utiliza `DKStorageManageInfoDL`. No caso de versões anteriores do servidor de Content Manager, do Content Manager Versão 8 e versões posteriores, consulte “Trabalhar com o Content Manager Versão 8.2” na página 129.

Java

```
String fileName = "e:\\test\\notepart.txt"; //file for add
int partId = 0; //let system decide the partId
String itemId = "V5SPB$WBLOHIQ4YI"; //an existing itemId
DKDatastoreDL dsDL = new DKDatastoreDL(); //required datastore
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD",""); //connect to dstore
DKBlobDL axdo = new DKBlobDL(dsDL); //criar XDO
DKPidXDODL apid = new DKPidXDODL(); //create PID
apid.setPartId(partId); //set partId
apid.setPrimaryId(itemId); //set itemId
axdo.setPidObject(apid); //set PID object
axdo.setContentClass(DK_DL_CC_ASCII); //set ContentClass

// ----- Criar o DKStorageManageInfoDL
StorageManageInfoDL aSMS = new DKStorageManageInfoDL();
aSMS.setRetention(888); //optional
aSMS.setCollectionName("TESTCOLLECT1"); //already defined in DL SMS
aSMS.setManagementClass("TESTMGT1"); //opcional
aSMS.setStorageClass("FIXED"); //optional
axdo.setExtension("DKStorageManageInfoDL", (dkExtension)aSMS);
axdo.add(fileName); //adicionar do ficheiro
System.out.println("after add partId = " + axdo.getPartId());
//display the partId after add

dsDL.disconnect(); // disconnect from and destroy datastore
dsDL.destroy();
// ----- Handle the exceptions
```

C++

```
DKString fileName="e:\\test\\notepart.txt"; //file for add
int partId = 0; //let system decide the partId
DKString itemId = "V5SPB$WBLOHIQ4YI"; //an existing itemId
DKString rtype = "FRN$NULL"; //optional
DKDatastoreDL dsDL; //required datastore
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD"); //connect to dstore
DKBlobDL* axdo = new DKBlobDL(&dsDL); //create XDO
DKPidXDODL* apid = new DKPidXDODL; //create Pid
apid->setPartId(partId); //set partId
apid->setPrimaryId(itemId); //set itemId
apid->setRepType(rtype); //set repType
axdo->setPidObject(apid); //set pid object
axdo->setContentClass(DK_DL_CC_ASCII); //set ContentClass

//---set DKStorageManageInfoDL-----
DKStorageManageInfoDL aSMS = new DKStorageManageInfoDL();
aSMS.setRetention(888); //optional
aSMS.setCollectionName("TESTCOLLECT1"); //already defined in DL SMS
aSMS.setManagementClass("TESTMGT1"); //optional
aSMS.setStorageClass("FIXED"); //optional
axdo->setExtension("DKStorageManageInfoDL", (dkExtension)aSMS);
axdo->add(fileName); //add from file
System.out.println("after add partId = " + axdo->getPartId());
//display the partId after add

dsDL.disconnect(); //disconnect from datastore
System.out.println("armazenamento de dados desligado");
```

Consulte os seguintes exemplos de código no directório CMBROOT\Samples para obter exemplos relativos a adicionar objectos de pesquisa indexados e objectos de suporte de dados ao Content Manager.

- TxdoAddBsmsDL

- TxdosAddBsmsDL
- TxdoAddFsmsDL
- TxdosAddFsmsDL
- TxdomAddsmsDL

Alterar a recolha de memória de um XDO

Poderá alterar o conjunto de armazenamento de um XDO existente. Após configurar o objecto de extensão DKStorageManageInfoICM, chame o método `changeStorage`.

Java

```
System.out.println("about to call changeStorage().....");
axdo.changeStorage();
System.out.println("changeStorage() success.....");
```

C++

```
System.out.println("about to call changeStorage().....");
axdo->changeStorage();
System.out.println("changeStorage() success.....");
```

A aplicação exemplo completa da qual foi retirado este exemplo (TxdoChgSmsICM) encontra-se no directório CMBROOT\Samples\java\d1 ou CMBROOT\Samples\cpp\d1.

Trabalhar com XML (Apenas para Java)

O Enterprise Information Portal suporta a importação e exportação de conteúdos de documentos de XML para e de Content Manager como DDOs e XDOs, utilizando as APIs de Java. Esta função permite importar, armazenar e obter uma grande variedade de objectos em Content Manager—, tais como conteúdos de dados ou multimédia— de diferentes sistemas de informação sem desenvolver interfaces separadas para cada sistema. Por exemplo, se possui um objecto armazenado num sistema de dados, pode convertê-lo num ficheiro XML e, de seguida, importá-lo para o Content Manager utilizando APIs do Enterprise Information Portal Java. Uma vez no Content Manager, o utilizador poderá fazer com o objecto o que podia fazer com outro objecto qualquer do Content Manager.

Expandir a capacidade de importação e exportação de XML

Actualmente, as funções de importação e exportação de XML, `DKDDO.toXML()` e `DKDDO.fromXML()`, não suportam artigos com valores de atributo de referência, ligações ou conteúdos de pastas. A operação de exportação é concluída, mas utiliza o PID como referência ao artigo no servidor de conteúdos específico. Se forem importados para um servidor de conteúdos diferente, esses artigos não existirão no sistema destino, pois as informações de PID são únicas do servidor de conteúdos das quais provieram. Estas duas referências apenas criam novos artigos e não suportam várias versões ao mesmo tempo.

Como solução, deve construir uma ferramenta à parte para tratar artigos referidos, detectar e sobrepor artigos já existentes ou para executar outras funções não suportadas por esta interface.

A Versão 8.1 com o Pacote de Correções 1 e a Versão 8.2 do EIP faculta novas ferramentas exemplo TImportICM e TExportICM) e a API de ferramenta exemplo

(TExportPackageICM) que executam a funcionalidade de importação e exportação com todas as funções. A nova funcionalidade exemplo expande significativamente as interfaces DKDDO.toXML() e DKDDO.fromXML() demonstradas em SItemXMLImportExportICM.

As novas ferramentas estão incluídas na Versão 8.1 com o Pacote de Correções 1 e na Versão 8.2 como exemplos de Java. Para obter mais informações, consulte as Ferramentas de Importação / Exportação Exemplo, o documento de API e TExportPackageICM.doc, localizados no directório CMBROOT\Samples\java\icm.

Importar documentos XML

Pode importar XML de fontes diferentes, incluindo entrada padrão, ficheiros, memórias tampão e endereços da Web (URLs). Também é possível importar um ficheiro XML na sua totalidade. Estes construtores extraem o conteúdo de um documento de XML, criar um DKDDO correspondente e um dkXDO qualquer a ele associado. Depois pode chamar o método adicionar no DDO para adicionar o objecto ao Content Manager. O novo DDO pertence a um tipo de artigo do Content Manager Versão 8 ou a uma classe de índice de uma versão anterior do Content Manager e só pode ser armazenado no Content Manager. A importação de um ficheiro XML auto-referenciador permite-lhe armazenar o ficheiro XML original como um XDO; ou seja, o utilizador não perde o XML no processo de importação, tornando o próprio XML disponível para uma futura utilização.

Ao importar o conteúdo de XML, utilize estes métodos em DKDDO:

```
toXML(DKNVPair xmlDestination, java.lang.String path, int options)
fromXML(DKNVPair xmlSource, int options)
```

A utilização dos construtores de DKDDO com o Content Manager anterior para importar XML é abandonada.

Ao importar o conteúdo do XML, tenha em atenção os seguintes parâmetros:

1. Tenha em atenção que apenas pode importar para o Content Manager ou para o Content Manager anterior.
2. Os ficheiros XML que tenham conteúdos para importação têm de estar de acordo com a definição do tipo de documento XML, demonstrada mais abaixo.
3. A importação e exportação de XML são suportadas apenas pelas APIs de Java.

As secções seguintes descrevem os pré-requisitos e os métodos de importação do conteúdo de XML:

- A definição do tipo de documento de XML (DTD)
- Armazenar conteúdo em documentos XML
- Extrair conteúdo de diferentes fontes de XML
- Importar conteúdo de XML para o Content Manager.

A Definição do Tipo de Documento XML (DTD)

Para importar conteúdos para o Content Manager para armazenar como XML, o utilizador deve armazenar o conteúdo em documentos de XML, que estejam em conformidade com ddo.dtd, que se encontra em CMROOT\samples\java\dl\ddo.dtd.

```
<!ELEMENT ddo (pid?, pidIdStrings*, propertyCount?, property*, dataCount?, dataItem*, xdoValue?)>
<!ATTLIST ddo      entityName CDATA #REQUIRED
                  xmlns      CDATA #FIXED "http://www.omg.org/pub/docs/formal/97-12-12.pdf#ddo/EIP-7.1"
                  xdoType    CDATA #IMPLIED
                  dsType     CDATA #IMPLIED
<!--
<!ELEMENT pid EMPTY>
<!ATTLIST pid      dsType      CDATA #IMPLIED
                  dsName      CDATA #IMPLIED
                  objectType   CDATA #IMPLIED
                  pidString    CDATA #IMPLIED
-->
```

```

<!ELEMENT pidIdStrings EMPTY>
<!-- ATTLIST pidIdStrings idPosition CDATA #REQUIRED
idValue CDATA #REQUIRED -->
<!-- ELEMENT propertyCount (#PCDATA)>
<!-- ELEMENT property EMPTY>
<!-- ATTLIST property propertyId CDATA #IMPLIED
propertyName CDATA #IMPLIED
propertyValue CDATA #IMPLIED
propertyType CDATA #IMPLIED -->
<!-- ELEMENT dataCount (#PCDATA)>
<!-- ELEMENT dataItem (dataPropertyCount?, dataProperty+, (dataValue | dataValues))>
<!-- ATTLIST dataItem dataId CDATA #IMPLIED
dataName CDATA #REQUIRED
dataNameSpace CDATA #IMPLIED -->
<!-- ELEMENT dataPropertyCount (#PCDATA)>
<!-- ELEMENT dataProperty EMPTY>
<!-- ATTLIST dataProperty propertyId CDATA #IMPLIED
propertyName CDATA #IMPLIED
propertyValue CDATA #IMPLIED
propertyType CDATA #IMPLIED -->
<!-- ELEMENT dataValues (dataValueCount?, dataValue*)>
<!-- ATTLIST dataValues collectionName CDATA #IMPLIED -->
<!-- ELEMENT dataValueCount (#PCDATA)>
<!-- ELEMENT dataValue (#PCDATA | ddo | xdoRef | linkRef)*>
<!-- ELEMENT linkRef (linkSource, linkTarget, linkItem?)>
<!-- ATTLIST linkRef linkTypeName CDATA #REQUIRED -->
<!-- ELEMENT linkSource (ddo)>
<!-- ATTLIST linkSource sameAsParentDDO (yes | no) "no">
<!-- ELEMENT linkTarget (ddo)>
<!-- ATTLIST linkTarget sameAsParentDDO (yes | no) "no">
<!-- ELEMENT linkItem (ddo)>
<!-- ELEMENT xdoRef (xdoPid, xdoIdStrings*, xdoValue)>
<!-- partId abandonado é substituído por xdoIdString numa xdoRef -->
<!-- repType abandonado é substituído por xdoIdString numa xdoRef -->
<!-- ELEMENT xdoPid EMPTY>
<!-- ATTLIST xdoPid dsType CDATA #REQUIRED
dsName CDATA #IMPLIED
xdoType CDATA #REQUIRED
objectType CDATA #IMPLIED
partId CDATA #IMPLIED
repType CDATA #IMPLIED
pidString CDATA #IMPLIED -->
<!-- ELEMENT xdoIdStrings EMPTY>
<!-- ATTLIST xdoIdStrings idPosition CDATA #REQUIRED
idValue CDATA #REQUIRED -->
<!-- ELEMENT xdoValue (contentType?, specificInfo*, searchEngineInfo?, smsInfo?, xdoContent?)>
<!-- ATTLIST xdoValue refType CDATA #REQUIRED
refEncoding CDATA #IMPLIED
mimeType CDATA #REQUIRED
XML-LINK CDATA #IMPLIED
HREF CDATA #IMPLIED -->
<!-- ELEMENT contentType (#PCDATA)>
<!-- ELEMENT specificInfo EMPTY>
<!-- ATTLIST specificInfo infoGroup CDATA #REQUIRED
infoName CDATA #REQUIRED
infoValue CDATA #REQUIRED -->
<!-- searchEngineInfo deprecated it is replaced by specificInfo -->
<!-- ELEMENT searchEngineInfo EMPTY>
<!-- ATTLIST searchEngineInfo searchEngine CDATA #REQUIRED
searchIndex CDATA #REQUIRED
searchInfo CDATA #REQUIRED -->
<!-- smsInfo abandonada é substituída por specificInfo -->
<!-- ELEMENT smsInfo EMPTY>
<!-- ATTLIST smsInfo smsRetention CDATA #IMPLIED
smsCollection CDATA #IMPLIED
smsMgmtClass CDATA #IMPLIED
smsStorageClass CDATA #IMPLIED
smsObjServer CDATA #IMPLIED -->
<!-- ELEMENT xdoContent (#PCDATA)>

```

Armazenar conteúdo em documentos XML

Os ficheiros XML podem representar de diversas formas documentos ou pastas para importação para o Content Manager. Estes documentos e pastas também podem conter partes. O exemplo que se segue demonstra primeiro um artigo de dados típico de XML, dataItem dataId="1", cujo valor é Basuki. O ItemDados 13, no entanto, utiliza o nomeDados DKParts, que se relaciona com um XDO auto-referenciador.

Exemplo para Content Manager

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ddo SYSTEM "ddo.dtd">
<ddo entityName="MagXML3" dsType="ICM" xdoType="DKLobICM">
  <pid dsType="ICM" dsName="ICMNLSDb"/>
  <property propertyId="1" propertyName="item-type"
    propertyValue="document"/>
  <dataItem dataName="Classification3">

```

```

        <dataProperty propertyName="type" propertyValue="string" />
        <dataValue>B</dataValue>
    </dataItem>
    <dataItem dataName="PublisherName3">
        <dataProperty propertyName="type" propertyValue="string"/>
        <dataValue>PublisherName3</dataValue>
    </dataItem>
    <dataItem dataName="MagEdrXML3" dataNameSpace="CHILD">
        <dataProperty propertyName="type" propertyValue="collection+ddo"/>
    <dataValues collectionName="DKChildCollection">
        <dataValue>
            <ddo entityName="MagEdrXML3" dsType="ICM" xdoType="DKLobICM">
                <pid dsType="ICM" dsName="ICMNLSDDB"/>
                <dataItem dataName="Sophias3.USPostal3">
                    <dataProperty propertyName="type" propertyValue="string"/>
                    <dataValue>Title of Book</dataValue>
                </dataItem>
                <dataItem dataName="Sophias3.Association3">
                    <dataProperty propertyName="type" propertyValue="string"/>
                    <dataValue>Sophias3.Association3</dataValue>
                </dataItem>
            </ddo>
        </dataValue>
    </dataValues>
</dataItem>
    <xdoValue mimeType="text/plain" refType="file"
        XML-LINK="SIMPLE" HREF="TSophiaBefore.txt" >
    </xdoValue>
</ddo>

```

Para obter alguns exemplos de como o XML armazena objectos com o Content Manager, consulte os exemplos no directório DK\Samples\java\icm\.

Exemplo para o Content Manager anterior

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ddo SYSTEM "ddo.dtd">
<ddo entityName="DLSAMPLE">
    <pid dsType="DL" dsName="LIBSRVRN"/>
    <property propertyId="1" propertyName="item-type" propertyValue="document"/>
    <dataItem dataId="1" dataName="DLSEARCH_Author">
        <dataProperty propertyId="1" propertyName="type" propertyValue="string"/>
        <dataValue>Basuki</dataValue>
    </dataItem>
    . . . .
    <dataItem dataId="13" dataName="DKParts">
        <dataProperty propertyId="1" propertyName="type" propertyValue="collection+xdo"/>
        <dataProperty propertyId="2" propertyName="nullable" propertyValue="false"/>
        <dataValues>
            <dataValue>
                <xdoRef>
                    <xdoPid dsType="DL" xdoType="DKBlobDL"/>
                    <xdoValue refType="self" mimeType="text/xml">
                        <contentType>XML</contentType>
                    </xdoValue>
                </xdoRef>
            </dataValue>
        </dataValues>
    </dataItem>
</ddo>

```

Para obter alguns exemplos de como o XML armazena objectos com o Content Manager anterior, consulte os seguintes exemplos de código no directório de exemplos.

- dlsamp01.xml

- dlsamp02.xml
- dlsamp03.xml
- dlsamp04.xml
- dlsamp05.xml
- dltypes01.xml

Extrair conteúdo de diferentes fontes de XML

Os métodos DKDDO podem extrair conteúdos de uma variedade de fontes de XML, incluindo entrada de dados standard, ficheiros, memórias tampão e endereços da Web (URLs). Chame os métodos DKDDO para extrair conteúdos da fonte de XML do utilizador e para iniciar o processo de importação.

Aqui estão alguns exemplos de cada fonte de XML:

XML de um ficheiro:

Java

```
xmlSource = new DKNVPair("FILE", "dlsamp01.xml");
```

XML de uma memória tampão:

Java

```
File file = new File("dlsamp01.xml");
int fileSize = (int) file.length();
byte[] data = new byte[fileSize];
DataInputStream dis = new DataInputStream(new FileInputStream(file));
dis.readFully(data);
String strBuffer = new String(data);
DKNVPair xmlSource = new DKNVPair("BUFFER", strBuffer);
int importOptions=DK_CM_XML_VALIDATION;
```

XML de um endereço da Web (URL):

Java

```
xmlSource = new DKNVPair("URL", "file:///d://myxml//dlsamp01.xml");
// replace file:///d:// with http://www.webaddress.com/ for URL
Int importOptions=0;
```

Importar conteúdo de XML para o Content Manager

O exemplo seguinte percorre estes passos básicos:

1. Crie um DKDDO.
2. Crie e estabeleça ligação a um servidor de conteúdos, neste caso o Content Manager ou versões anteriores do Content Manager.
3. Adicione o novo DKDDO novamente ao servidor de conteúdos que, neste caso, é o Content Manager.
4. Utilize `dkDataObjectBase fromXML(DKNVPair xmlSource)` ou `dkDataObjectBase fromXML(DKNVPair xmlSource, int options)` para importar a fonte de XML.

O DKDDO resultante está em conformidade com as especificações de ddo.dtd e pertence a um tipo de artigo do Content Manager ou a uma classe de índice do Content Manager anterior.

Java

```
// ----- Construct a DDO by importing the XML document
xmlSource = new DKNVPair("FILE", "icmsamp01.xml");
int importOptions = DK_CM_XML_VALIDATION;
DKDDO ddo = new DKDDO();
ds = new DKDatastoreICM();
// ..... connect to the datastore
ddo.setDatastore (ds);
// ----- Add the DDO to the datastore
ddo.add()
// ----- Import the XML
ddo.fromXML(xmlSource, importOptions);
```

Para obter a aplicação exemplo completa, consulte
SItemXMLImportExportICM.java no directórioCMBROOT\Samples\java\icm.

Exportar XML

Pode exportar dados de DDO e XDO como um documento de XML do Content Manager ou do Content Manager anterior. Utilize o método toXML(DKNVPair xmlDestination, String path, int options) do DKDDO para exportar.

Restrição: As interfaces DKDDO.toXML() e DKDDO.fromXML() demonstradas neste exemplo não podem ser utilizadas para exportar um artigo de um servidor de conteúdos e importá-lo para outro, caso façam referência a outros artigos através de ligações, conteúdos de pastas ou atributos de referência. A operação de importação irá criar novos artigos e não actualizará ou sobreporá artigos existentes, nem suportará várias versões de um artigo. Para obter informações relativas a ferramentas que expandam esta capacidade, consulte “Expandir a capacidade de importação e exportação de XML” na página 80.

O exemplo seguinte mostra como exportar XML:

Java

```
DKNVPair xmlDestination = null;
//----- Use this line to export to STDOUT
//xmlDestination = new DKNVPair("STDOUT", null);

//----- Use this line to export to a buffer
//xmlDestination = new DKNVPair("BUFFER", new Object());

//----- Use this line to export to a file
String xmlFile = "export.xml";
if(!fileName.equals("")){
    xmlFile = fileName;
}

String strOS = System.getProperty("os.name");
if (strOS.indexOf("Win") != -1) { // Windows OS
    xmlFile = outputPath + "\\\" + xmlFile;
} else { // other than Win OS
    xmlFile = outputPath + "/" + xmlFile;
}

xmlDestination = new DKNVPair("FILE", xmlFile);

ddo.toXML(xmlDestination, outputPath, DKConstant.DK_CM_XDO_REFERENCE);
```

Para obter a aplicação exemplo completa, consulte
SItemXMLImportExportICM.java no directório CMBROOT\Samples\java\icm.

Criar documentos e utilizar o atributo DKPARTS

O atributo DKPARTS num DDO representa a recolha de partes num documento. O valor deste atributo é um objecto de DKParts, que consiste num conjunto de DKPart. Os objectos de DKPart são artigos de tipos de artigos classificados de *parte de documento* e contêm conteúdo de recurso.

Apenas para Content Manager: Um documento é um artigo que pode ser armazenado, obtido e comutado entre sistemas do Content Manager e utilizadores como uma unidade diferente. Um artigo ao qual é atribuído o tipo semântico de *documento* deve conter informações que compõem um documento, o que não significa uma implementação de um modelo de documentos específico. Um artigo criado a partir de um tipo de artigo classificado de documento (também conhecido como modelo de documentos) significa que o artigo irá conter partes de documento, uma implementação específica de um documento de modelos facultado pelo Content Manager. Os tipos de artigo classificados de documento podem criar artigos com o tipo semântico de documento ou de pasta. As partes de documento incluem vários tipos de conteúdo, incluindo, por exemplo, texto, imagens e folhas de cálculo.

O exemplo seguinte cria um documento e adiciona partes de documento ao Content Manager. **Requisito:** O tipo de artigo definido pelo utilizador S_docModel, deve ser definido no sistema, classificado como Modelo de Documentos. Também suporta tipos de parte ICMBASE e ICMBASETEXT, tal como o Exemplo de API de Instrução SItemTypeCreationICM demonstra.

Java

```
// Create a document
DKDDO ddoDocument = dsICM.createDDO("S_docModel", DKConstant.DK_CM_DOCUMENT);

// Create parts
DKLobICM base = (DKLobICM) dsICM.createDDO("ICMBASE",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASE);
DKTextICM baseText1 = (DKTextICM) dsICM.createDDO("ICMBASETEXT",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASETEXT);
DKTextICM baseText2 = (DKTextICM) dsICM.createDDO("ICMBASETEXT",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASETEXT);

// Set parts' MIME type (SResourceItemMimeTypesICM.txt sample)
base.setMimeType("application/msword");
baseText1.setMimeType("text/plain");
baseText2.setMimeType("text/plain");

// Load content into parts (SResourceItemCreationICM sample)
base.setContentFromClientFile("SResourceItemICM_Document1.doc");
// Load file
baseText1.setContentFromClientFile("SResourceItemICM_Text1.txt");
// into memory
baseText2.setContentFromClientFile("SResourceItemICM_Text2.txt");

// Access the DKParts attribute
DKParts dkParts = (DKParts) ddoDocument.getData(ddoDocument.dataId(
DKConstant.DK_CM_NAMESPACE_ATTR, DKConstant.DK_CM_DKPARTS));

// Add parts to document
dkParts.addElement(base);
dkParts.addElement(baseText1);
dkParts.addElement(baseText2);

// Add new document to persistent datastore
ddoDocument.add();
```

Para obter mais informações relativas à criação de documentos com partes, consulte o Exemplo de Instrução de API SDocModelItemICM no directório CMBROOT\Samples\java\icm.

C++

```
// Create a document
DKDDO* ddoDocument = dsICM->createDDO("S_docModel", DK_CM_DOCUMENT);

// Create Parts
DKLobICM* base = (DKLobICM*) dsICM->createDDO("ICMBASE",
DK_ICM_SEMANTIC_TYPE_BASE);
DKTextICM* baseText1 = (DKTextICM*) dsICM->createDDO("ICMBASETEXT",
DK_ICM_SEMANTIC_TYPE_BASETEXT);
DKTextICM* baseText2 = (DKTextICM*) dsICM->createDDO("ICMBASETEXT",
DK_ICM_SEMANTIC_TYPE_BASETEXT);

// Set parts' MIME type (SResourceItemMimeTypesICM.txt sample)
base->setMimeType("application/msword");
baseText1->setMimeType("text/plain");
baseText2->setMimeType("text/plain");

// Load content into parts (SResourceItemCreationICM sample)
base->setContentFromClientFile("SResourceItemICM_Document1.doc");
// Load the file into memory.
baseText1->setContentFromClientFile("SResourceItemICM_Text1.txt");
baseText2->setContentFromClientFile("SResourceItemICM_Text2.txt");

// Access the DKParts attribute
DKParts* dkParts = (DKParts*)(dkCollection*) ddoDocument->getData(
ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKPARTS));

// Add parts to document
dkParts->addElement((dkDataObjectBase*)(DKDDO*)base);
dkParts->addElement((dkDataObjectBase*)(DKDDO*)baseText1);
dkParts->addElement((dkDataObjectBase*)(DKDDO*)baseText2);

// Add new document to persistent datastore
ddoDocument->add();
```

Para obter mais informações relativas à criação de documentos com partes, consulte o Exemplo de Instrução de API SDocModelItemICM no directório CMBROOT\Samples\cpp\icm.

O DDO possui todas as partes do conjunto de partes. Actualize e elimine as partes através do DDO de documento.

O exemplo seguinte demonstra como obter e aceder a partes a partir de um DDO.

Java

```
// NOTE: Print function provided in SDocModelItemICM sample

// Get the DKParts object.
short dataid = ddoDocument.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,
if(dataid==0)
    throw new Exception("No DKParts Attribute Found! Either item type does not
        support parts or the document has not been explicitly retrieved.");
DKParts dkParts = (DKParts) ddoDocument.getData(dataid);

// Go through part list
dkIterator iter = dkParts.createIterator(); // Create an Iterator
while(iter.more()){                        // While there are items left
    DKDDO part = (DKDDO) iter.next();      // Move pointer & return next
    System.out.println("Item Id: "+((DKPidICM)part.getPidObject()).getItemId());
}
```

Para obter a aplicação exemplo completa, consulte SDocModelItemICM no directório CMBROOT\Samples\java\icm.

C++

```
// NOTE: Print function provided in SDocModelItemICM sample

// Get the DKParts object.
short dataid = ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKPARTS);
if(dataid==0)
    throw DKException("No DKParts Attribute Found! Either item type does not
        support parts or the document has not been explicitly retrieved.");
DKParts* dkParts = (DKParts*)(dkCollection*) ddoDocument->getData(dataid);

// Go through part list
dkIterator* iter = dkParts->createIterator(); // Create an Iterator
while(iter->more()){                          // While there are items
    DKDDO* part = (DKDDO*) iter->next()->value(); // Move pointer & return next
    cout << "Item Id:" << ((DKPidICM*)part->getPidObject())->getItemId()<< endl;
}
delete(iter);                                // Free Memory
```

Para obter mais informações relativas à criação de documentos com partes, consulte o Exemplo de Instrução de API SDocModelItemICM no directório CMBROOT\Samples\cpp\icm.

Criar pastas e utilizar o atributo DKFOLDER

Num DDO de pasta, utilize o atributo DKFOLDER para representar a recolha de documentos e outras pastas que pertencem a uma pasta. O valor deste atributo é um objecto de DKFolder, que é uma recolha de DDOs. Tal como é demonstrado de seguida, o atributo DKFolder é definido da mesma forma que o atributo DKParts.

Apenas para Content Manager: Uma pasta é um artigo de qualquer tipo de artigo, independentemente da sua classificação, com o tipo semântico de *pasta*. Qualquer artigo com o tipo semântico de pasta irá conter uma funcionalidade de pasta específica facultada pelo Content Manager, para além de todas as capacidades de artigo que não é de recurso e todas as funcionalidades adicionais disponíveis a partir de uma classificação de tipo de artigo como, por exemplo, modelo de

documentos ou recurso. As pastas podem conter um número indeterminado de artigos de qualquer tipo, incluindo documentos e sub-pastas. Uma pasta é indexada por atributos.

O exemplo seguinte cria uma pasta e adiciona conteúdos ao Content Manager.

Requisito: O tipo de artigo definido pelo utilizador, `S_simple`, deve ser definido no sistema, tal como está definido no Exemplo de Instrução de API `SItemTypeCreationICM`.

Java

```
// Create new folder in memory
DKDDO ddoFolder = dsICM.createDDO("S_simple", DKConstant.DK_CM_FOLDER);

// Create and save contents to place in folder
DKDDO ddoDocument = dsICM.createDDO("S_simple", DKConstant.DK_CM_DOCUMENT);
DKDDO ddoFolder2 = dsICM.createDDO("S_simple", DKConstant.DK_CM_FOLDER);
DKDDO ddoItem = dsICM.createDDO("S_simple", DKConstant.DK_CM_ITEM);
ddoDocument.add();
ddoItem.add();
ddoFolder2.add();

// Access the DKFolder attribute
DKFolder dkFolder = (DKFolder)
ddoFolder.getData(ddoFolder.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,
                                   DKConstant.DK_CM_DKFOLDER));

// Add contents to folder
dkFolder.addElement(ddoDocument);
dkFolder.addElement(ddoItem);
dkFolder.addElement(ddoFolder2); // Note, Folders can contain sub-folders.

// Save the folder in the persistent datastore.
ddoFolder.add();
```

Para obter mais informações relativas à criação de Pastas, consulte o Exemplo de Instrução de API `SFolderICM` no directório `CMBROOT\Samples\java\icm`.

C++

```
// Create new folder in memory
DKDDO* ddoFolder = dsICM->createDDO("S_simple", DK_CM_FOLDER);

// Create and save contents to place in folder
DKDDO* ddoDocument = dsICM->createDDO("S_simple", DK_CM_DOCUMENT);
DKDDO* ddoFolder2 = dsICM->createDDO("S_simple", DK_CM_FOLDER);
DKDDO* ddoItem = dsICM->createDDO("S_simple", DK_CM_ITEM);
ddoDocument->add();
ddoItem->add();
ddoFolder2->add();

// Access the DKFolder attribute
DKFolder* dkFolder = (DKFolder*)(dkCollection*) ddoFolder->getData(
    ddoFolder->dataId(DK_CM_NAMESPACE_ATTR, DK_CM_DKFOLDER));

// Add contents to folder
dkFolder->addElement(ddoDocument);
dkFolder->addElement(ddoItem);
dkFolder->addElement(ddoFolder2); // Note, Folders can contain sub-folders.

// Save the folder in the persistent datastore.
ddoFolder->add();
```

Para obter mais informações relativas à criação de Pastas, consulte o Exemplo de Instrução de API SFolderICM no directório CMBROOT\Samples\cpp\icm.

No Content Manager Versão 8, o artigo de pasta não possui os conteúdos de pasta. Um artigo pode ser adicionado a várias pastas. Remover um artigo de uma pasta apenas quebra a relação pasta-conteúdo gerida pelo sistema. Os artigos da pasta devem ser actualizados e eliminados independentemente. Em versões anteriores do Content Manager, o DDO possui os conteúdos no conjunto.

O exemplo seguinte demonstra como obter e aceder a conteúdos de pastas a partir de um DDO.

Java

```
// NOTE: Print function provided in SFolderICM API Education Sample

// Get the DKFolder object.
short dataid = folder.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,
    DKConstant.DK_CM_DKFOLDER);

if(dataid==0)
    throw new Exception("No DKFolder Attribute Found! DDO is either not a
        Folder or Folder Contents have not been explicitly retrieved.");
DKFolder dkFolder = (DKFolder) folder.getData(dataid);

// Access contents
dkIterator iter = dkFolder.createIterator(); // Create an Iterator
while(iter.more()){ // While there are items left
    DKDDO ddo = (DKDDO) iter.next(); // Move to & return next element
    System.out.println("Item Id: "+((DKPidICM) ddo.getPidObject()).getItemId());
}
```

For the complete sample application, refer to the SFolderICM education sample in the CMBROOT\Samples\java\icm directory.

C++

```
// NOTE: Print function provided in SFolderICM API Education Sample

// Get the DKFolder object.
short dataid = folder->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKFOLDER);
if(dataid==0)
    throw DKException("No DKFolder Attribute Found! DDO is either not a Folder
    or Folder Contents have not been explicitly retrieved.");
DKFolder* dkFolder = (DKFolder*)(dkCollection*) folder->getData(dataid);

// Access contents
dkIterator* iter = dkFolder->createIterator(); //Create an Iterator
while(iter->more()){                          //While there are items left
    DKDDO* ddo = (DKDDO*) iter->next()->value(); //Move to & return next element
    cout << "Item Id: " << ((DKPidICM*)ddo->getPidObject())->getItemId()<< endl;
}
```

Para obter a aplicação exemplo completa, consulte o exemplo de instrução SFolderICM no directório CMBROOT\Samples\cpp\icm.

Utilizar DKAny (Apenas para C++)

O DKAny contém objectos cujo tipo pode variar em tempo de execução. Um objecto de DKAny pode ser qualquer um dos tipos seguintes:

- null
- (unsigned) short
- (unsigned) long
- double
- char
- TypeCode
- DKBoolean
- DKString
- DKDate
- DKTime
- DKTimestamp

Um DKAny só pode ser NULL caso não lhe tenha sido atribuído um valor (DKAny any). Após ter-lhe sido atribuído um valor, `DKAny::isNull()` irá devolver FALSE.

Para além dos tipos acima apresentados, um objecto de DKAny também pode conter os seguintes tipos de referência de objectos:

- `dkDataObjectBase*`
- `dkCollection*`
- `void*`

Utilizar código de tipo

O utilizador poderá determinar o tipo actual do objecto de DKAny chamando a função `typeCode`, que devolve um objecto `TypeCode`, ou seja, `tc_null` para nulo, `tc_short` para curto e assim por diante. Consulte a *referência de API online* para obter uma listagem completa de tipos de códigos.

Gerir memória em DKAny

O DKAny gere a memória para o objecto que contém, a não ser que o objecto que contém seja um tipo de referência de objectos. A cópia de operações relacionadas que envolvem referências de objectos irá criar apenas uma cópia do indicador. O utilizador terá de ter sempre presente a localização dos tipos de referência de objectos durante a cópia e a eliminação.

Utilizar construtores

O DKAny fornece um construtor para cada tipo que suporta. O exemplo seguinte mostra como deve criar um objecto de DKAny que contenha alguns dos tipos enumerados na secção anterior.

C++

```
DKAny any1((unsigned short) 10);           // contains unsigned short 10
DKAny any2((long) 200);                     // contains long 200
DKAny any3(DKString("any string"));         // contains DKString
DKAny any4(DKTime(10,20,30));               // contains DKTime
DKAny any5((dkDataObjectBase*) new DKDDO); // contains DKDDO
DKAny any6(new MyObject(5,"abc"));          // contains MyObject
DKAny any7(new DKDDO);                     // shorter form of any5
```

Obter o código de tipo

Utilize a função `typeCode` para descobrir o código de tipo do objecto dentro de DKAny.

C++

```
DKAny::TypeCode type_code;
type_code = any1.typeCode(); // type_code is tc_ushort
type_code = any4.typeCode(); // type_code is tc_time
type_code = any5.typeCode(); // type_code is tc_dobase (object ref)
type_code = any6.typeCode(); // type_code is tc_voidptr since
                             // MyObject is not recognized by DKAny
```

Atribuir um novo valor a DKAny

Para atribuir um novo valor a um objecto de DKAny existente, utilize o operador de atribuição do sinal de igual (=). O DKAny fornece uma atribuição para cada código de tipo.

C++

```
DKAny any;           // any contains null
long vlong = 300;
DKTimestamp vts(1997,8,28,10,11,12,999);
dkDataObjectBase* dobase =
(dkDataObjectBase*) new DKDDO;
any = vlong;         // any contains long 300
any = vts;           // any contains timestamp
any = dobase;        // any contains ddo
any = new DKDDO;     // any contains ddo
```

Atribuir um valor de DKAny

Voltar a atribuir um DKAny para um tipo normal exige um operador de conversão. Por exemplo:

C++

```
vlong      = (long) any2;                // sets vlong to 200
DKTime at  = (DKTime) any4;              // sets at to (10,20,30)
DKDDO* ddo = (DKDDO*) ((dkDataObjectBase*) any5); // extract the ddo
dkDataObjectBase* dobase = any7;        // extract the DDO
```

O utilizador irá gerar uma excepção de conversão de tipo inválido caso o tipo não seja correspondente. Deste modo, terá de verificar o código de tipo antes de converter o DKAny para um tipo normal:

C++

```
if (any5.typeCode() == DKAny::tc_dobase)
    dobase = (dkDataObjectBase*) any5;
```

O utilizador pode criar uma instrução de tipo de letra para verificar o tipo de DKAny, da seguinte forma:

C++

```
switch(any.typeCode()) {
    case DKAny::tc_short:
        // operation for short
        ...
        break;
    case DKAny::tc_ushort:
        // operation for unsigned short
        ...
        break;
    ... etc.
}
```

Se o objecto de DKAny contiver uma referência de objectos, o utilizador poderá obter o conteúdo de DKAny como um indicador de vazio, convertendo-o depois para o tipo apropriado. No entanto, utilize esta operação apenas caso conheça o código de tipo que é utilizado no DKAny:

C++

```
// knows exactly any5 contains DKDDO
ddo = (DKDDO*) any5.value();
```

Apresentar DKAny

O utilizador poderá utilizar cout para apresentar o conteúdo de um objecto de DKAny:

C++

```
cout << any3 << endl; // displays "any string"
cout << any4 << endl; // displays "10:20:30"
cout << any5 << endl; // displays "(dkDataObjectBase*) <address>",
                        // onde endereço é a localização na memória do ddo
```

Destruir DKAny

Visto que o DKAny pode manter uma referência de objectos mas não gere a memória para tipos de referência de objectos, o utilizador terá de gerir a memória para estes tipos. O exemplo seguinte gere a memória de um objecto de DKAny:

C++

```
DKDDO* ddo = new DKDDO; // creates a DKDDO in the heap
DKAny anyA((dkDataObjectBase*)ddo);
DKAny* anyB = new DKAny(anyA); // creates anyB in the heap
                                // anyA and anyB contains a
                                // reference to the same ddo
...
delete anyB; // delete anyB, does not delete ddo
if (anyA.typeCode() == DKAny::tc_dobase)
    delete ((dkDataObjectBase*) anyA.value()); // deletes the ddo
```

A última instrução de eliminação tem de ser executada antes de sair do âmbito, caso contrário anyA é eliminado, deixando o DDO como uma fuga na memória.

Sugestões de programação

Recomendação: Ao converter um literal inteiro num DKAny, é aconselhável que indique explicitamente qual o tipo para evitar a conversão num tipo que não é o pretendido. Mude para

C++

```
any = 10; // ambiguous
any = (unsigned long) 10; // unambiguous
any = (short) 4; // unambiguous
```

Utilizar conjuntos e iteradores

dkCollection é uma classe abstracta que fornece os métodos para trabalhar com um conjunto. DKSequentialCollection é uma implementação concreta de dkCollection. Outros conjuntos são implementados como subclasses de DKSequentialCollection. Estes conjuntos contêm os objectos de dados como membros.

Os membros do conjunto são normalmente objectos do mesmo tipo; no entanto, um conjunto pode conter membros de tipos diferentes.

Apenas para C++: Quando um novo número é adicionado, o conjunto passa a detê-lo. Quando o membro é obtido, o utilizador poderá obter um indicador para o objecto de DKAny dentro do conjunto. Este objecto pertence ao conjunto, o que significa que o conjunto gere a memória dos seus membros de DKAny. Um objecto

de DKAny pode manter uma referência de objectos mas não pode gerir memória para tipos de referência de objectos, o utilizador terá de gerir a memória para estes.

Utilizar métodos de recolha sequencial

A DKSequentialCollection fornece métodos para adicionar, obter, remover e substituir os seus membros. Para além disso, também tem um método ordenar. O exemplo seguinte ilustra como adicionar um novo membro a uma recolha (o método addElement assume um objecto como o parâmetro).

Java

```
DKSequentialCollection sq = new DKSequentialCollection();
String str = " first member ";
sq.addElement(str);           // add a new element at the last position
```

C++

```
DKSequentialCollection sq;
DKAny any = DKString(" first member ");
sq.addElement(any);           // add a new element at last position
                                // any will be copied into the collection
                                // you own the original any, the collection
                                // owns the copy
```

Utilizar o iterador sequencial

O utilizador irá iterar sobre membros da recolha através de iteradores. As APIs têm dois tipos de iteradores: dkIterator e DKSequentialIterator.

Java

dkIterator, o iterador base, suporta os métodos seguinte, mais e repor. A subclasse DKSequentialIterator contém mais métodos. Um iterador é criado chamando o método createIterator da recolha. Após criar o iterador, utilize o método setToFirst() para indicar o primeiro artigo. O exemplo seguinte demonstra a utilização de um iterador:

```
dkIterator iter = sq.createIterator(); // create an iterator for sq
Object member;
iter.setToFirst();
member = iter.at();
while(iter.more()) {                // While there are more members
    member = iter.next();            // move to the next member and get it

    System.out.println(member);
    ....
}
```


C++

Os iteradores são fornecidos para lhe permitir a iteração sobre membros da recolha. Existem dois tipos de iteradores: o iterador base `dkIterator`, que suporta as funções `seguite`, `mais` e `repor`, além do `DKSequentialIterator` da subclasse, que contém mais funções. É criado um iterador através da chamada da função `createIterator` na recolha. Esta função cria um novo iterador e devolve-o ao utilizador. Utilize o código seguinte para iterar numa recolha:

```
dkIterator* iter = sq.createIterator(); // create an iterator for sq
DKAny* member;

// while there are more members
// get the current member and
// advance iter to the next member
while(iter->more()) {
    member = iter->next();

    cout << *member << endl; // display it, if you want to
    ...                     // do other processing
}
delete iter;                // do not forget to delete iter
```

O `DKSequentialIterator` fornece métodos adicionais para mover o iterador em qualquer direcção. O exemplo seguinte poderia ter sido reescrito da seguinte forma:

Java

```
// ----- Create a sequential iterator for sq
DKSequentialIterator iter =
    (DKSequentialIterator) sq.createIterator();
Object member;
iter.setToFirst();
while(iter.more()) {
    member = iter.at(); // get the current member
    ...               // do other processing
    iter.setToNext(); // advance to the next position
}
```

C++

```
DKSequentialIterator* iter = // create an iterator for sq
(DKSequentialIterator*) sq.createIterator();
DKAny* member;
while(iter->more()) {
    member = iter->at(); // get the current member
    ...               // do other processing
    iter->setToNext(); // advance to the next position
}
delete iter;
```

Este código permite-lhe executar algumas operações no membro actual antes de mover para o membro seguinte. Uma operação deste tipo poderá ser a substituição de um membro por outro novo, ou removê-lo.

Java

```
String st1 = "the new first member";
sq.replaceElementAt(st1, iter); // replace current member with a new one
....                          // or
sq.removeElementAt(iter);      // remove the current member
....
```

C++

```
any = DKString("the new first member");

sq.replaceElementAt(any, *iter); // replace current member with a new one
...                          // or
sq.removeElementAt();           // remove the current member
...
```

Sugestão: Quando remover o membro actual, o iterador avança para o membro seguinte. Ao remover um membro dentro de um ciclo, verifique-o segundo a forma ilustrada no exemplo seguinte. Deve verificar a condição de remoção para que não ignore o membro seguinte após remover o membro actual.

Java

```
....
if (removeCondition == true)
    sq.removeElementAt(iter); // remove current member, do not advance the
                              // iterator since it is advanced to the next
                              // after the removal operation
else
    iter.setToNext();         // if no removal, advance the iterator to the
....                          // next position
```

C++

```
...
if (removeCondition == TRUE)
    sq.removeElementAt(*iter); // remove current member, do not advance iter
                              // since it is advanced to the next after
                              // the removal operation
else
    iter->setToNext();          // no removal, advance the iterator
...                          // to the next position
```

Gerir memória em recolhas (Apenas para C++)

O conjunto gere a recolha para os seus membros, que são objectos de DKAny. As mesmas regras que regem os objectos de DKAny aplicam-se aqui, caso o objecto dentro de DKAny seja um tipo de referência de objectos então o utilizador é responsável pela gestão da memória quando:

- Destrói a recolha.
- Substitui um membro.
- Remove um membro.

Este exemplo demonstra como gerir a memória nestas situações:

```
C++
// retrieve the member and hang-on to it
member = iter->at();

// code to handle this member as to prevent memory leaks
if (member->typeCode() == DKAny::tc_dobase) {
    // delete it if no longer needed
    delete ((dkDataObjectBase*) member->value());
}

sq.removeElementAt(*iter);           // remove it from the collection
```

Em vez de eliminar o membro, poderá adicioná-lo a outra recolha. Deve executar passos idênticos antes de utilizar as funções `replaceElementAt` e `removeAllElement`.

Antes de destruir uma recolha, elimine os seus membros. Poderá escrever uma função de forma a executar esta tarefa e passar esta função para a função aplicar da recolha. Parta do princípio que possui uma recolha de objectos de `DKAny` que contém objectos de `DKAttributeDef`. O exemplo seguinte elimina a recolha:

```
C++
DKDatastoreICM dsICM;
...
DKAny any = dsICM.listSchemaAttributes("GRANDPA");
dkCollection* acoll = (dkCollection*) any;
...
acoll->apply(deleteDKAttributeDef);           // use the attributes
delete acoll;                               // deletes all members
```

Neste exemplo, `deleteDKAttributeDef` é uma função que assume o objecto `DKAny` como parâmetro. Está definido da seguinte forma:

```
C++
void deleteDKAttributeDef(DKAny& any) {
    delete ((DKAttributeDef*) any.value());
    any.setNull();                       // good practice
}
```

O utilizador poderá escrever a sua função de eliminação para eliminar a recolha ou remover alguns membros antes de eliminar a recolha.

Os destruidores de algumas recolhas conhecidas, como `DKParts`, `DKFolder` e `DKResults`, executam estas medidas de remoção necessárias. Contudo, estes não gerem a memória ao executar as funções `replaceElementAt`, `removeElementAt` ou `removeAllElement`.

Ordenar a recolha

Utilize a função `ordenar` para ordenar os membros da recolha por ordem ascendente ou por ordem descendente com base numa tecla especificada. O utilizador terá de passar um objecto de ordenação e a ordem pretendida. A

interface para objectos de ordenação está definida em `dkSort.java` (Java) ou `dkSort.hpp` (C++). Pode escrever a sua própria função de ordenação para ordenar a sua recolha específica. O exemplo seguinte ilustra como ordenar uma recolha de DDOs:

Java

```
DKResults rs;
....           // Execute a query to fill DKResults with DDOs
....
DKSortDDOId sortId; // Declare the sort function object; sort on item-id
rs.sort(sortId);    // by default, sort in ascending order
....
```

C++

```
DKResults* rs;
....
// Execute a query to fill DKResults with DDOs
....
DKSortDDOId* sortId; // Declare the sort function object; sort on item-id
rs->sort(sortId); // by default, sort in ascending order
....
```

Sugestão: O objecto de ordenação é criado na pilha de memória e por isso não tem de ser explicitamente eliminado. A função é de utilização simultânea, o que significa que uma única cópia pode ser partilhada, reutilizada ou passada para outra função.

Compreender a recolha e o iterador associados

Utilize uma recolha associada na sua aplicação para processar objectos de dados resultantes de uma consulta como um conjunto. A recolha associada preserva as relações de sub-agrupamentos que existem entre os objectos de dados.

Uma recolha associada é uma recolha de objectos de `DKResults`. Esta é criada para manter os resultados da `DKFederatedQuery`, que pode ter origem nos vários servidores de conteúdos heterogéneos. Cada objecto de `DKResults` contém os resultados de pesquisa de um servidor de conteúdos específico. Uma recolha associada pode conter um número infinito de recolhas imbricadas.

Para passar as várias etapas de uma recolha associada, crie e utilize um `dkIterator` ou um `DKSequentialIterator`. Em seguida crie outro `dkIterator` para passar em revista cada objecto de `DKResults`, de forma a iterá-lo e processá-lo segundo o servidor de conteúdos que lhe deu origem.

Também pode criar um iterador associado, `dkFederatedIterator`, e utilizá-lo para passar em revista todos os membros da recolha, independentemente do servidor de conteúdos que deu origem aos resultados.

Restrição: Não pode executar uma consulta numa recolha associada.

A Figura 8 na página 101 revela a estrutura e o comportamento de `DKFederatedCollection`.

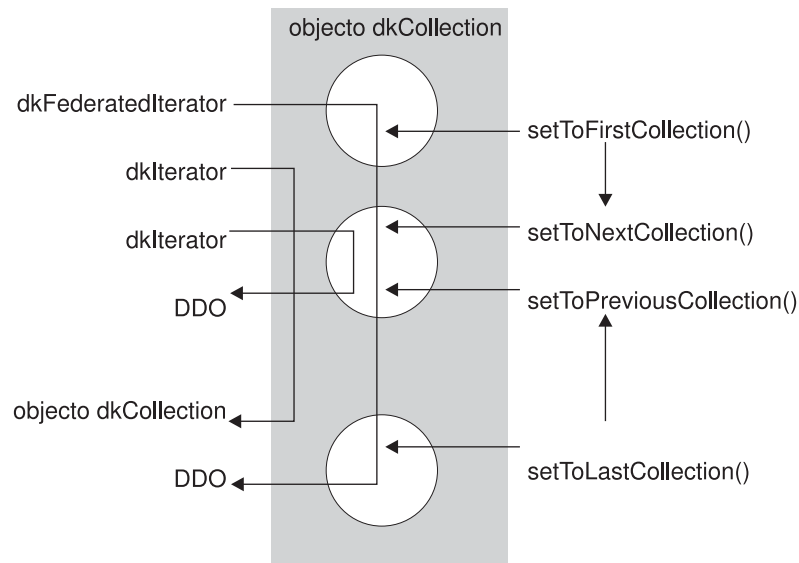


Figura 8. Estrutura e comportamento de DKFederatedCollection

Na Figura 8, o que está a oval representa a DKFederatedCollection que contém vários círculos menores que são os objectos de DKResults. O dkFederatedIterator transpõe os limites da recolha e devolve sempre um DDO.

O primeiro dkIterator é um iterador para a DKFederatedCollection e devolve sempre um objecto de DKResultse. O segundo dkIterator é um iterador para o segundo objecto de DKResults; este devolve um DDO para cada membro da recolha de DKResults.

A função setToFirstCollection no dkFederatedIterator define a posição para o primeiro DDO de DKFederatedCollection. Neste caso, é o primeiro elemento do primeiro objecto da recolha de DKResults. Nesta altura, se for invocada a função setToNextCollection, este irá definir a posição do iterador para o primeiro DDO da segunda recolha de DKResults.

A função setToLastCollection em dkFederatedIterator define a posição do iterador como o último DDO de DKFederatedCollection. Neste caso, é o último elemento do primeiro objecto da recolha de DKResults. Se for invocada a função setToPreviousCollection, este irá definir a posição do iterador para o último DDO da recolha de DKResults anterior.

Consultar um servidor de conteúdos

O utilizador pode pesquisar um servidor de conteúdos e receber resultados num objecto de dkResultSetCursor ou de DKResults. No caso de alguns servidores, o utilizador pode criar um objecto de consulta para representar a sua consulta e, de seguida, invocar as funções executar ou avaliar do objecto de consulta. Com o auxílio do servidores de conteúdos, o objecto da consulta executa tarefas de processamento de consultas, tais como preparar e executar uma consulta, supervisionar o estado da execução de uma consulta e armazenar os resultados. Alguns servidores de conteúdo suportam a utilização de um objecto de consulta como alternativa; dois destes servidores são o das versões anteriores do Content Manager e o servidor de conteúdos associado.

Para os servidores de conteúdos que suportam objectos de consulta, existem quatro tipos de objectos de consulta: paramétricas, de texto, de imagem e combinadas. A consulta combinada é composta por consultas de texto e paramétricas. Nem todos os servidores de conteúdos podem executar consultas combinadas. O Content Manager anterior suporta a consulta de imagens.

Relativamente ao Content Manager, estão integradas as consultas paramétricas e de texto. Não deve utilizar objectos de consulta; para obter informações relativas a efectuar consultas no Content Manager, consulte “Consultar o servidor de Content Manager” na página 186. Para obter informações relativas a consultas em servidores de versões anteriores do Content Manager, consulte *Trabalhar com outros servidores de conteúdos*.

Um servidor de conteúdos utiliza dois métodos para executar uma consulta: executar e avaliar. A função executar devolve um objecto de `dkResultSetCursor`, a função avaliar devolve um objecto de `DKResults`. O objecto `dedkResultSetCursor` é utilizado para manusear grandes conjuntos de resultados e executar as funções eliminar e actualizar na posição actual do cursor do conjunto de resultados. Poderá utilizar a função `fetchNextN` para obter um grupo de objectos numa recolha.

`dkResultSetCursor` também pode ser utilizado para voltar a executar uma consulta através da chamada dos métodos fechar e abrir. Esta acção está descrita em “Utilizar o cursor do conjunto de resultados” na página 121.

`DKResults` contém todos os resultados de uma consulta. O utilizador pode iterar os artigos na recolha para a frente ou para trás e pode executar a recolha ou utilizá-la como um âmbito para outra consulta.

Consulte “Abrir e fechar o cursor do conjunto de resultados para voltar a executar a consulta” na página 122 para obter mais informações.

Restrição: Quando consultar um servidor de conteúdos do Domino.Doc, é devolvido um objecto de `DKResults`. No entanto, não pode consultá-lo ou utilizá-lo como âmbito para outra consulta.

Diferenças entre `dkResultSetCursor` e `DKResults`

Uma recolha de `dkResultSetCursor` e de `DKResults` têm as seguintes diferenças:

- O `dkResultSetCursor` funciona como um cursor do servidor de conteúdos; pode ser utilizado para grandes conjuntos de resultados porque os DKDDOs que contém são obtidos um de cada vez. Também pode ser utilizado para voltar a executar uma consulta, de forma a obter os resultados mais recentes.

Restrição: Não pode voltar a executar uma consulta num servidor de conteúdos do Domino.Doc, mesmo quando estiver a utilizar um `dkResultSetCursor`.

- O `DKResults` contém todo o conjunto de resultados e suporta um iterador bidireccional.
- Deixar um `dkResultSetCursor` aberto durante longos períodos de tempo pode prejudicar o rendimento do utilizadores simultâneos em alguns servidores de conteúdos.

Utilizar consultas paramétricas

Uma consulta paramétrica é uma consulta que necessita de uma correspondência exacta em relação à condição especificada no predicado da consulta e aos valores de dados armazenados no servidor de conteúdos.

Nota: Os exemplos de consulta das secções seguintes aplicam-se a versões anteriores do Content Manager. Para obter informações relativas a consultas no Content Manager V8, consulte “Compreender a linguagem da consulta” na página 185.

Formular uma cadeia de consultas paramétricas

Para criar uma consulta primeiro tem de formular uma cadeia de consultas. No exemplo seguinte, a cadeia e consulta é definida para representar uma consulta na classe de índice denominada GP2DLS2 no Content Manager anterior. Para obter exemplos de cadeias de consulta no Content Manager, consulte “Exemplos de consultas” na página 194. A condição da consulta é pesquisar todos os documentos ou pastas em que o atributo DLSEARCH_DocType não é nulo. O número máximo de resultados está limitado em cinco e o conteúdo está definido como YES, de forma a que sejam devolvidos os conteúdos do documento e da pasta.

Java

```
String cmd = "SEARCH=(INDEX_CLASS=GP2DLS2," +  
             "MAX_RESULTS=5," +  
             "COND=(DLSEARCH_DocType > null));" +  
             "OPTION=(CONTENT=YES;" +  
             "TYPE_QUERY=DYNAMIC;" +  
             "TYPE_FILTER=FOLDERDOC)";
```

C++

```
DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE,";  
cmd += "MAX_RESULTS=5,";  
cmd += "COND=(DLSEARCH_DocType <> NULL));";  
cmd += "OPTION=(CONTENT=YES,";  
cmd += "TYPE_QUERY=DYNAMIC,";  
cmd += "TYPE_FILTER=FOLDERDOC)";
```

O exemplo especifica que o servidor do Content Manager anterior utiliza SQL dinâmica para esta consulta e que todas as pastas e documentos devem ser pesquisados. Diferentes servidores de conteúdos utilizam sintaxes de cadeia de consulta diferentes: a consulta associada possui a sua própria sintaxe de cadeia de consulta. Consulte as informações do servidor de conteúdos que pretende pesquisar ou consulte *referência de API online* para obter mais informações. Se o nome do atributo possuir mais do que uma palavra ou caso se encontre em linguagem de DBCS, deve ser colocado entre apóstrofes ('). Se o valor do atributo estiver em DBCS, este deve ser colocado entre aspas (").

Formular uma consulta paramétrica com vários critérios

Poderá especificar mais do que um critério de pesquisa para uma consulta paramétrica. O exemplo seguinte mostra como especificar uma consulta em duas classes de índice para o Content Manager anterior:

Java

```
String cmd = "SEARCH=(INDEX_CLASS=GP2DLS1,MAX_RESULTS=3," +
             "COND=(DLSEARCH_DocType <> null);" +
             "INDEX_CLASS=GP2DLS1,MAX_RESULTS=8," +
             "COND=('First name'==\"Robert\"));" +
             "OPTION=(CONTENT=YES;" +
             "TYPE_QUERY=DYNAMIC;" +
             "TYPE_FILTER=FOLDERDOC)";
```

C++

```
DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE,MAX_RESULTS=3,";
cmd += "COND=(DLSEARCH_DocType <> NULL);";
cmd += "INDEX_CLASS=DLSAMPLE,MAX_RESULTS=8,";
cmd += "COND=('First name' == \"Robert\"));";
cmd += "OPTION=(CONTENT=YES;";
cmd += "TYPE_QUERY=DYNAMIC;";
cmd += "TYPE_FILTER=FOLDERDOC)";
```

Executar uma consulta paramétrica

Após a cadeia de consultas o utilizador irá criar o objecto da consulta. O `DKDatastorexx` que representa um servidor de conteúdos contém um método para a criação de um objecto da consulta. Utilize o objecto da consulta para executar a consulta e obter os resultados. O exemplo seguinte mostra como criar um objecto de consulta paramétrica e como executar a consulta num servidor de versões anteriores do Content Manager; não deverá utilizar um objecto de consulta no Content Manager Versão 8 ou em versões posteriores. Após a execução da consulta, os resultados são devolvidos numa recolha de `DKResults`.

Atenção: Quando eliminar um objecto de `DKResults`, também são eliminados todos os seus membros. Certifique-se de que não elimina o elemento duas vezes.

Java

```
// ----- Create the datastore, the query object, and the results set
DKDatastoreDL dsDL = new DKDatastoreDL();
dkQuery pQry = null;
DKResults pResults = null;
DKNameValuePair parms[] = null;
// ----- Connect to the datastore
dsDL.connect(libSrv,userid,pw,"");
// ----- Formulate the query string
String cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE," +
             "MAX_RESULTS=5," +
             "COND=(DLSEARCH_DocType <> NULL));" +
             "OPTION=(CONTENT=YES;" +
             "TYPE_QUERY=STATIC;" +
             "TYPE_FILTER=FOLDERDOC)";
// ----- Create the query using the query string
pQry = dsDL.createQuery(cmd, DK_CM_PARAMETRIC_QL_TYPE, parms);
// ----- Execute the query
pQry.execute(parms);
// ----- Process the results
pResults = (DKResults)pQry.result();
processResults((dkCollection)pResults);
// ----- Disconnect when you are through
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual foi retirada este exemplo (TSamplePQryDL.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
dkQuery* pQry;
DKAny any;
DKResults* pResults;

cout << "connecting to datastore" << endl;
dsDL.connect(libsrv,userid,pw);
cout << "datastore connected libsrv: " <<libsrv<< " userid: "<<userid<<endl;

DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE,";
cmd += "MAX_RESULTS=5,";
cmd += "COND=(DLSEARCH_DocType <> NULL));";
cmd += "OPTION=(CONTENT=YES);";
cmd += "TYPE_QUERY=STATIC;TYPE_FILTER=FOLDERDOC";
cout << "query string " << cmd << endl;
cout << "create query" << endl;
pQry = dsDL.createQuery(cmd);
cout << "executing query" << endl;
pQry->execute();
cout << "query executed" << endl;
cout << "get query results" << endl;
any = pQry->result();
pResults = (DKResults*)((dkCollection*) any);

processResults(pResults);

dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TSamplePQryDL.cpp) está disponível no directório Cmbroot/Samples/cpp/dl.

Executar uma consulta paramétrica a partir de um servidor de conteúdos

O DKDatastorexx que representa um servidor de conteúdos tem um método para executar a consulta. O exemplo seguinte demonstra como executar uma consulta paramétrica num servidor de conteúdos do Content Manager anterior. Após a execução da consulta, os resultados são devolvidos num objecto de dkResultSetCursor.

Java

```
// ----- Create the datastore and cursor
DKDatastoreDL dsDL = new DKDatastoreDL();
dkResultSetCursor pCur = null;
DKNVPair parms[] = null;
// ----- Connect to the content server
dsDL.connect(libSrv,userid,pw,"");
// ----- Formulate the query string
String cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE," +
             "MAX_RESULTS=5," +
             "COND=((DLSEARCH_DocType <> NULL)" +
             "AND (DLSEARCH_Date >= 1995)));"+
             "OPTION=(CONTENT=YES;" +
             "TYPE_QUERY=DYNAMIC;" +
             "TYPE_FILTER=FOLDERDOC)";

...
// ----- Execute the query using the query string
pCur = dsDL.execute(cmd, DK_CM_PARAMETRIC_QL_TYPE, parms);
// ----- Process query results as you want
...
// ----- When finished with the cursor, delete it, and disconnect
pCur.destroy();
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDL.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
...
DKDatastoreDL dsDL;
dkResultSetCursor* pCur = 0;
cout << "Datastore DL created" << endl;
cout << "connecting to datastore" << endl;
dsDL.connect(libsrv,userid,pw);
cout << "datastore connected " << libsrv << " userid - " << userid << endl;
// DKString cmd = "SEARCH=(COND=('DLSEARCH_DocType' == \"html\"));";
DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE,";
cmd += "MAX_RESULTS=5,";
cmd += "COND=(DLSEARCH_DocType <> NULL));";
cmd += "OPTION=(CONTENT=YES;";
cmd += "TYPE_QUERY=STATIC;TYPE_FILTER=FOLDERDOC)";
cout << "query string " << cmd << endl;
cout << "executing query" << endl;
pCur = dsDL.execute(cmd);
cout << "query executed" << endl;
...
...
if (pCur != 0)
delete pCur;
dsDL.disconnect();
...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDL.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Avaliar uma consulta paramétrica a partir de um servidor de conteúdos

O DKDatastorexx que representa um servidor de conteúdos tem um método para avaliar uma consulta. Os resultados são devolvidos numa recolha de DKResults. O exemplo seguinte demonstra como avaliar uma consulta paramétrica num servidor de conteúdos do Content Manager anterior.

Java

```
// ----- Create the query string
String cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE," +
              "COND=((DLSEARCH_Date >= \"1995\") AND " +
              "      (DLSEARCH_Date <= \"1996\")))";
              "OPTION=(CONTENT=NO;" +
              "      TYPE_QUERY=DYNAMIC;" +
              "      TYPE_FILTER=FOLDERDOC)";

DKNVPair parms[] = null;
DKDDO item = null;
// ----- Create the datastore and connect
// replace following with your library server, user ID,
// password DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");

// ----- Call evaluate, get the results, and create an
// iterator to process them
DKResults pResults =
    (DKResults)dsDL.evaluate(cmd, DK_CM_PARAMETRIC_QL_TYPE, parms);
dkIterator pIter = pResults.createIterator();
while (pIter.more()) {
    item = (DKDDO)pIter.next();
    ... // ----- Process the DKDDO as appropriate
}
dsDL.disconnect();
dsDL.destroy();
```

C++

```
DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKAny *element;
DKDDO *item;
DKString cmd = "SEARCH=(INDEX_CLASS=GP2DLS5,";
cmd += "COND=((DLSEARCH_Date >= \"1995\") AND ";
cmd += "      (DLSEARCH_Date <= \"1996\")))";
cmd += "OPTION=(CONTENT=NO;";
cmd += "TYPE_QUERY=DYNAMIC;TYPE_FILTER=FOLDERDOC)";

...
DKAny any = dsDL.evaluate(cmd);
DKResults* pResults = (DKResults*)((dkCollection*) any);
dkIterator* pIter = pResults->createIterator();
while (pIter->more()) {
    element = pIter->next();
    item = (DKDDO*)element->value();
    // Process the DKDDO
}
delete pIter;
delete pResults;
dsDL.disconnect();
```

Utilizar a consulta de texto

No Content Manager Versão 8 e versões posteriores, as consultas de texto e paramétricas são integradas; consulte “Criar pesquisa de texto e paramétrica combinada” na página 191.

No Content Manager anterior, o utilizador pode executar pesquisas de texto e paramétricas. As pesquisas de texto consultam os índices de texto criados pelo Motor de Pesquisa de Texto para pesquisar o documento de texto actual.

Formular uma cadeia de consultas de texto

O utilizador irá iniciar uma pesquisa de texto através da formulação de uma cadeia de consultas. No exemplo que se segue, é criada uma cadeia de consultas que representa uma consulta contra o índice de texto TMINDEX. A cadeia de consulta contém critérios para pesquisar todos os documentos de texto com a palavra UNIX ou membro. O número máximo de resultados devolvidos é de cinco.

Java

```
String cmd = "SEARCH=(COND=(UNIX OR member));" +  
            "OPTION=(SEARCH_INDEX=TMINDEX; MAX_RESULTS=5)";
```

C++

```
DKString cmd = "SEARCH=(COND=(UNIX OR member));";  
cmd += "OPTION=(SEARCH_INDEX=TMINDEX;MAX_RESULTS=5)";
```

Formular uma consulta de texto em vários índices

Poderá utilizar a consulta de texto para pesquisar mais do que um índice. O exemplo seguinte demonstra como especificar uma consulta para dois índices.

Importante: Caso especifique mais do que um índice de pesquisa de texto na consulta, os índices terão de ser do mesmo tipo. Por exemplo, poderá especificar dois índices precisos na consulta, mas não poderá especificar um índice preciso e um índice linguístico na mesma consulta.

Java

```
DKString cmd = "SEARCH=(COND=(UNIX OR member));";  
cmd += "OPTION=(SEARCH_INDEX=(TMINDEX,TMINDEX2); MAX_RESULTS=5)";
```

C++

```
String cmd = "SEARCH=(COND=(UNIX OR member));" +  
            "OPTION=(SEARCH_INDEX=(TMINDEX, INDEX2); MAX_RESULTS=5)";
```

Executar uma consulta de texto

Após a consulta de texto o utilizador irá criar o objecto da consulta. O DKDatastorexx que representa um servidor de conteúdos contém um método para a criação de um objecto da consulta. Os resultados são devolvidos numa recolha de DKResults. Utilize o objecto da consulta para executar a consulta e obter os resultados. O exemplo seguinte mostra como criar um objecto de consulta de texto e executar uma consulta.

Java

```
// ----- Create the datastore; declare query and the results
DKDatastoreTS dsTS = new DKDatastoreTS();
dkQuery pQry = null;
DKResults pResults = null;
DKNameValuePair parms[] = null;
// ----- Connect to the datastore
//         for example, dsTS.connect("zebra","7502",DK_CTYP_TCPIP);
dsTS.connect(srchSrv,"","");
// ----- Formulate the query string
String cmd = "SEARCH=(COND=(member AND UNIX));" +
             "OPTION=(SEARCH_INDEX=TMINDEX)";
// ----- Create and execute the query
pQry = dsTS.createQuery(cmd, DK_CM_TEXT_QL_TYPE, parms);
pQry.execute(parms);
// ----- Process the results
pResults = (DKResults)pQry.result();
processResults((dkCollection)pResults);
// ----- When finished, disconnect
dsTS.disconnect();
dsTS.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TSampleTQryTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
dkQuery* pQry;
DKAny any;
DKResults* pResults;

cout << "connecting to datastore" << endl;
//dsTS.connect("zebra","7502",DK_CTYP_TCPIP);
dsTS.connect(srchSrv,"","");
cout << "connected to datastore srchSrv: " << srchSrv << endl;

DKString cmd = "SEARCH=";
cmd += "(COND=(UNIX OR member));";
cmd += "OPTION=(SEARCH_INDEX=";
cmd += srchIndex;
cmd += ")";
cout << "query string " << cmd << endl;
cout << "create query" << endl;
pQry = dsTS.createQuery(cmd);
cout << "executing query" << endl;
pQry->execute();
cout << "query executed" << endl;
cout << "get query results" << endl;
any = pQry->result();
pResults = (DKResults*)((dkCollection*) any);

processResults(pResults);

dsTS.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TSampleTQryTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Executar uma consulta de texto a partir de um servidor de conteúdos

O DKDatastorexx utilizado para representar um servidor de conteúdos faculta um método para executar uma consulta. Os resultados são devolvidos num objecto de dkResultSetCursor. O exemplo seguinte demonstra como executar uma consulta num servidor de versões anteriores do Content Manager.

Java

```
// ----- Create the datastore; declare query and the results
DKDatastoreTS dsTS = new DKDatastoreTS();
dkResultSetCursor pCur = null;
DKNVPair parms[] = null;
// ----- Connect to the datastore
//         for example, dsTS.connect
("zebra", "7502", DK_TS_CTYP_TCPIP);
dsTS.connect(srchSrv, "", "", "");

// ----- Formulate the query string
String cmd = "SEARCH=(COND=(internet OR UNIX));" +
             "OPTION=(SEARCH_INDEX=TMINDEX);" +
             "MAX_RESULTS=5";

...
// ----- Execute the query and process the results
as appropriate
pCur = dsTS.execute(cmd, DK_CM_TEXT_QL_TYPE, parms);
...
// ----- When finished, delete the cursor and disconnect
pCur.destroy();
dsTS.disconnect();
dsTS.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM", "", ' ');
DKString cmd = "SEARCH=(COND=('UNIX operating' AND system));";
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";
...

dkResultSetCursor* pCur = dsTS.execute(cmd);
DKDDO *item = 0;
while (pCur->isValid()) {
    item = pCur->fetchNext();
    if (item != 0) {
        // Process the DKDDO
        ...
        delete item;
    }
}
delete pCur;
dsTS.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Avaliar uma consulta de texto a partir de um servidor de conteúdos

O DKDatastorexx que utiliza para representar um servidor de conteúdos, fornece um método de avaliação para executar uma consulta e devolver uma recolha de DKResults. O exemplo seguinte demonstra como avaliar uma consulta de texto num servidor de conteúdos de versões anteriores do Content Manager.

Java

```
// ----- Create the datastore and the query string
DKDatastoreTS dsTS = new DKDatastoreTS();
String cmd = "SEARCH=(COND=($MC=*$ UN*));" +
             "OPTION=(SEARCH_INDEX=TMINDEX)";

DKNVPair parms[] = null;
DKDDO item = null;
DKDatastoreTS dsTS;
// ----- Connect to the datastore
dsTS.connect("TM","", ' ');
...
// ----- Call evaluate, get the results, and process
// as appropriate
DKResults pResults = (DKResults)dsTS.evaluate(cmd,DK_CM_TEXT_QL_TYPE,parms);
dkIterator pIter = pResults.createIterator();
while (pIter.more()) {
    item = (DKDDO)pIter.next();
    // ----- Process the individual DKDDO objects
}
// ----- Disconnect
dsTS.disconnect();
dsTs.destroy();
```

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM", "", ' ');
DKAny *element;
DKDDO *item;
DKString cmd = "SEARCH=(COND=('UNIX operating' AND system));";
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";

...
DKAny any = dsTS.evaluate(cmd);
DKResults* pResults = (DKResults*)((dkCollection*) any);
dkIterator* pIter = pResults->createIterator();
while (pIter->more()) {
    element = pIter->next();
    item = (DKDDO*) element->value();
    // Process the DKDDO
    ...
}
delete pIter;
delete pResults;
dsTS.disconnect();
```

Obter informações destacadas para correspondência

As informações correspondentes contêm o texto do documento e as informações destacadas para cada correspondência da respectiva consulta.

Ao formular a cadeia de consultas o utilizador irá definir as opções MATCH_INFO e MATCH_DICT. Defina MATCH_INFO como YES para devolver as informações

destacadas correspondentes. A opção MATCH_DICT especifica se as informações destacadas correspondentes serão obtidas através de um dicionário. As informações correspondentes são devolvidas no atributo DKMATCHESINFO no DKDDO devolvido de uma consulta de texto. O valor do atributo DKMATCHESINFO será um objecto DKMatchesInfoTS.

Obter informações destacadas correspondentes é bastante moroso porque o documento é obtido do servidor de conteúdos e analisado de forma linguística para determinar potenciais correspondências. A execução deste processo causa impacto no rendimento de uma consulta de texto.

Obter informação destacada correspondente para cada artigo de resultados da consulta de texto: O exemplo seguinte obtém informações destacadas correspondentes para cada artigo de resultados da consulta de texto enquanto esta decorre. Visto que a opção MATCH_DICT está definida como NO, o dicionário não é utilizado.

Java

```
// ----- Create the datastore
DKDatastoreTS dsTS = new DKDatastoreTS();
dkResultSetCursor pCur = null;
DKNVPair parms[] = null;
// ----- Connect to the content server
//      replace following with your library server,
//      user ID, password
dsTS.connect("TM","", "", "LIBACCESS=(LIBSRVRN,
FRNADMIN, PASSWORD)");
// ----- Formulate the query string
String cmd = "SEARCH=(COND=('UNIX operating' AND system));" +
            "OPTION=(SEARCH_INDEX=TMINDEX; MAX_RESULTS=5; +
            "MATCH_INFO=YES; MATCH_DICT=NO)";

...

pCur = dsTS.execute(cmd,DK_CM_TEXT_QL_TYPE,parms);
DKDDO item = null;
DKMatchesInfoTS pMInfo = null;
DKMatchesDocSectionTS pMSect = null;
DKMatchesParagraphTS pMPara = null;
DKMatchesTextItemTS pMText = null;
int i = 0;
int j = 0;
int k = 0;
int m = 0;
int lCCSID = 0;
int lLang = 0;
int lOffset = 0;
int lLen = 0;
int numberSections = 0;
int numberParagraphs = 0;
int numberTextItems = 0;
int numberNewLines = 0;
String strDoc = "";
String strSection = "";
String strText = "";
Object anyObj = null;
while (pCur.isValid())
{
    // ----- Get the next DKDDO
    item = pCur.fetchNext();
    if (item != null)
    {
        // continued...
    }
}
```

Java (continuação)

```
// ----- Process the DKDDO
for (i = 1; i <= item.dataCount(); i++)
{
    anyObj = item.getData(i);
    if (anyObj instanceof String)
    {
        ...
    }
    else if (anyObj instanceof Integer)
    {
        ...
    }
    else if (anyObj instanceof Short)
    {
        ...
    }
    else if (anyObj instanceof DKMatchesInfoTS)
    {
        pMInfo = (DKMatchesInfoTS)anyObj;
        // ----- process the Match Hightlighting information
        if (pMInfo != null)
        {
            strDoc = pMInfo.getDocumentName();
            numberSections = pMInfo.numberOfSections();
            // ----- loop thru document sections
            for (j = 1; j <= numberSections; j++)
            {
                pMSect = pMInfo.getSection(j);
                strSection = pMSect.getSectionName();
                numberParagraphs = pMSect.numberOfParagraphs();
                // ----- loop thru section paragraphs
                for (k = 1; k <= numberParagraphs; k++)
                {
                    pMPara = pMSect.getParagraph(k);
                    lCCSID = pMPara.getCCSID();
                    lLang = pMPara.getLanguageId();
                    numberTextItems = pMPara.numberOfTextItems();
                    // ----- loop thru paragraph text items
                    for (m = 1; m <= numberTextItems; m++)
                    {
                        pMText = pMPara.getTextItem(m);
                        strText = pMText.getText();
                        // ----- if match found in text item get offset
                        //          and length of match in text item
                        if (pMText.isMatch() == true)
                        {
                            lOffset = pMText.getOffset();
                            lLen = pMText.getLength();
                        }
                        numberNewLines = pMText.numberOfLines();
                    }
                }
            }
        }
    }
}
dsTS.disconnect();
```

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM", "", "", "LIBACCESS=(LIBSRVRN,FRNADMIN,PASSWORD)");
DKString cmd = "SEARCH=(COND=('UNIX operating' AND system));"
cmd += "OPTION=(SEARCH_INDEX=TMINDEX;MAX_RESULTS=5;"
      "MATCH_INFO=YES;MATCH_DICT=NO)";
...
dkResultSetCursor* pCur = dsTS.execute(cmd);
DKDDO *item = 0;
DKAny anyObj;
dkDataObjectBase *pDOBase = 0;
DKMatchesInfoTS *pMInfo = 0;
DKMatchesDocSectionTS *pMSect = 0;
DKMatchesParagraphTS *pMPara = 0;
DKMatchesTextItemTS *pMText = 0;
long i = 0;
long j = 0;
long k = 0;
long m = 0;
long lCCSID = 0;
long lLang = 0;
long lOffset = 0;
long lLen = 0;
long numberSections = 0;
long numberParagraphs = 0;
long numberTextItems = 0;
long numberNewLines = 0;
DKString strDoc;
DKString strSection;
DKString strText;
while (pCur->isValid())
{
    item = pCur->fetchNext();
    if (item != 0)
    {
        // Process the DKDDO
        for (i = 1; i <= item->dataCount(); i++)
        {
            anyObj = item->getData(i);
            switch (anyObj.typeCode())
            {
                case DKAny::tc_string :
                {
                    ...
                    break;
                }
                case DKAny::tc_long :
                {
                    ...
                    break;
                }
                case DKAny::tc_short :
                {
                    ...
                    break;
                }
                case DKAny::tc_dobase :
                {
                    ...
                    break;
                }
            }
        }
    }
}
// continued...
```

C++ (continuação)

```
// process the Match Highlighting information
pD0Base = a;
pMInfo = (DKMatchesInfoTS*)pD0Base;
if (pMInfo != 0)
{
    strDoc = pMInfo->getDocumentName();
    numberSections = pMInfo->numberOfSections();
    // loop thru document sections
    for (j = 1; j <= numberSections; j++)
    {
        pMSect = pMInfo->getSection(j);
        strSection = pMSect->getSectionName();
        numberParagraphs = pMSect->numberOfParagraphs();
        // loop thru section paragraphs
        for (k = 1; k <= numberParagraphs; k++)
        {
            pMPara = pMSect->getParagraph(k);
            lCCSID = pMPara->getCCSID();
            lLang = pMPara->getLanguageId();
            numberTextItems = pMPara->numberOfTextItems();
            // loop thru paragraph text items
            for (m = 1; m <= numberTextItems; m++)
            {
                pMText = pMPara->getTextItem(m);
                strText = pMText->getText();
                // if match found in text item get offset and
                // length of match in text item
                if (pMText->isMatch() == TRUE)
                {
                    00000000    lOffset = pMText->getOffset();
                    lLen = pMText->getLength();
                }
                numberNewLines = pMText->numberOfNewLines();
            }
        }
        break;
    }
    default :
    {
        break;
    }
}
...
delete item;
}
}
delete pCur;
dsTS.disconnect();
```

Obter informação destacada correspondente para um artigo de resultados da consulta de texto: O exemplo seguinte obtém informações destacadas correspondentes para um artigo específico obtido a partir de uma consulta de texto. O `dkResultSetCursor` transmitido nesta rotina terá de ter um estado aberto.

Java

```
DKDatastoreTS dsTS = new DKDatastoreTS();
dkResultSetCursor pCur = null;
DKNVPair parms[] = null;

dsTS.connect("TM","","","LIBACCESS=(LIBSRVRN,FRNADMIN,PASSWORD)");
String cmd = "SEARCH=(COND=('UNIX operating' AND system));" +
             "OPTION=(SEARCH_INDEX=TMINDEX;MAX_RESULTS=5)";

...
pCur = dsTS.execute(cmd);
DKDDO item = null;
Object anyObj = null;
DKMatchesInfoTS pMInfo = null;
DKMatchesDocSectionTS pMSect = null;
DKMatchesParagraphTS pMPara = null;
DKMatchesTextItemTS pMText = null;
int i = 0;
int j = 0;
int k = 0;
int m = 0;
int lCCSID = 0;
int lLang = 0;
int lOffset = 0;
int lLen = 0;
int numberSections = 0;
int numberParagraphs = 0;
int numberTextItems = 0;
int numberNewLines = 0;
String strDoc;
String strSection;
String strText;
String strDID = "";
String strXNAME = "";
String strDataName = "";
DKPid pid = null;
while (pCur.isValid()) {
    item = pCur.fetchNext();
    if (item != null)
    {
        pid = item.getPid();
        // Process the DKDDO
        for (i = 1; i <= item.dataCount(); i++)
        {
            anyObj = item.getData(i);
            strDataName = item.getDataName(i);
            if (strDID.equals(""))
            {
                strDID = pid.getId();
            }
            if (strXNAME.equals(""))
            {
                strXNAME = p.getObjectType();
            }
            ...
        }
    }
}
// continued...
```

Java (continuação)

```
// Get Match Highlighting Information
pMInfo = dsTS.getMatches(pCur, strDID, strXNAME, false);
strDID = "";
strXNAME = "";
if (pMInfo != null)
{
    strDoc = pMInfo.getDocumentName();
    numberSections = pMInfo.numberOfSections();
    // loop thru document sections
    for (j = 1; j <= numberSections; j++)
    {
        pMSect = pMInfo.getSection(j);
        strSection = pMSect.getSectionName();
        numberParagraphs = pMSect.numberOfParagraphs();
        // loop thru section paragraphs
        for (k = 1; k <= numberParagraphs; k++)
        {
            pMPara = pMSect.getParagraph(k);
            lCCSID = pMPara.getCCSID();
            lLang = pMPara.getLanguageId();
            numberTextItems = pMPara.numberOfTextItems();
            // loop thru paragraph text items
            for (m = 1; m <= numberTextItems; m++)
            {
                pMText = pMPara.getTextItem(m);
                strText = pMText.getText();
                // if match found in text item get offset and
                // length of match in text item
                if (pMText.isMatch() == true)
                {
                    lOffset = pMText.getOffset();
                    lLen = pMText.getLength();
                }
                numberNewLines = pMText.numberOfNewLines();
            }
        }
    }
}
dsTS.disconnect();
dsTS.destroy();
```

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM","","","LIBACCESS=(LIBSRVRN,FRNADMIN,PASSWORD)");
DKString cmd = "SEARCH=(COND=('UNIX operating' AND system));"
cmd += "OPTION=(SEARCH_INDEX=TMINDEX;MAX_RESULTS=5)";
...
dkResultSetCursor* pCur = dsTS.execute(cmd);
DKDDO *item = 0;
DKAny anyObj;
dkDataObjectBase *pDOBase = 0;
DKMatchesInfoTS *pMInfo = 0;
DKMatchesDocSectionTS *pMSect = 0;
DKMatchesParagraphTS *pMPara = 0;
DKMatchesTextItemTS *pMText = 0;
long i = 0;
long j = 0;
long k = 0;
long m = 0;
long lCCSID = 0;
long lLang = 0;
long lOffset = 0;
long lLen = 0;
long numberSections = 0;
long numberParagraphs = 0;
long numberTextItems = 0;
long numberNewLines = 0;
DKString strDoc;
DKString strSection;
DKString strText;
DKString strDID;
DKString strXNAME;
DKString strDataName;
DKPid pid;
while (pCur->isValid())
{
    item = pCur->fetchNext();
    if (item != 0)
    {
        pid = item->getPid();
        // Process the DKDDO
        for (i = 1; i <= item->dataCount(); i++)
        {
            anyObj = item->getData(i);
            strDataName = item->getDataName(i);
            if (strDataName == "")
            {
                strDID = pid.getId();
            }
            if (strXNAME == "")
            {
                strXNAME = p->getObjectType();
            }
            switch (anyObj.typeCode())
            {
                ...
            }
        }
    }
}
// continued...
```


C++ (continuação)

```
// Get Match Highlighting Information
pMInfo = dsTS.getMatches(pCur, strDID, strXNAME, FALSE);
strDID = "";
strXNAME = "";
if (pMInfo != 0)
{
    strDoc = pMInfo->getDocumentName();
    numberSections = pMInfo->numberOfSections();
    // loop thru document sections
    for (j = 1; j <= numberSections; j++)
    {
        pMSect = pMInfo->getSection(j);
        strSection = pMSect->getSectionName();
        numberParagraphs = pMSect->numberOfParagraphs();
        // loop thru section paragraphs
        for (k = 1; k <= numberParagraphs; k++)
        {
            pMPara = pMSect->getParagraph(k);
            lCCSID = pMPara->getCCSID();
            lLang = pMPara->getLanguageId();
            numberTextItems = pMPara->numberOfTextItems();
            // loop thru paragraph text items
            for (m = 1; m <= numberTextItems; m++)
            {
                pMText = pMPara->getTextItem(m);
                strText = pMText->getText();
                // if match found in text item get offset and
                // length of match in text item
                if (pMText->isMatch() == TRUE)
                {
                    lOffset = pMText->getOffset();
                    lLen = pMText->getLength();
                }
                numberNewLines = pMText->numberOfNewLines();
            }
        }
    }
    delete pMInfo;
}
...
delete item;
}
}
delete pCur;
dsTS.disconnect();
```

Utilizar o cursor do conjunto de resultados

dkResultSetCursor é um servidor de conteúdos que gere uma recolha virtual de objectos de DDO. Isto significa que a recolha não se materializa enquanto não capturar um elemento dela. A recolha é o conjunto de artigos resultante que corresponde aos critérios da consulta. Após terminar a utilização do cursor, chame o método de destruição para libertar a memória que utilizou.

Importante: As informações contidas nesta secção não se aplicam ao Content Manager 8.2. Consulte “Compreender a linguagem da consulta” na página 185 para obter detalhes.

Abrir e fechar o cursor do conjunto de resultados para voltar a executar a consulta

Ao criar um cursor do conjunto de resultados, este está num estado aberto. Para voltar a executar a consulta, feche e volte a abrir o cursor.

Java

```
String cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE);" +
             "OPTION=(CONTENT=YES);" +
             "TYPE_QUERY=DYNAMIC;" +
             "TYPE_FILTER=FOLDERDOC);";
DKNameValuePair parms[] = null;
...

dkResultSetCursor pCur = dsDL.execute(cmd,DK_CM_PARAMETRIC_QL_TYPE,parms);

pCur.close();
pCur.open();           //voltar a executar a consulta
```

C++

```
DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE)";
cmd += "OPTION=(CONTENT=YES)";
cmd += "TYPE_QUERY=DYNAMIC";
cmd += "TYPE_FILTER=FOLDERDOC";
...

dkResultSetCursor* pCur = dsDL.execute(cmd);
// voltar a executar a consulta
pCur->close();
pCur->open();
```

Definir e obter posições num cursor do conjunto de resultados

Poderá utilizar o cursor do conjunto de resultados para definir e obter a posição actual. O exemplo seguinte cria e executa a consulta. Dentro de um ciclo while, a posição do cursor está definida para a primeira (ou seguinte) posição válida. Em seguida um DDO é capturado dessa posição. É devolvido um nulo do método fetchObject caso o cursor tenha passado o último artigo.

Java

```
// ----- Formulate the query string
String cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE);" +
             "OPTION=(CONTENT=YES);" +
             "TYPE_QUERY=DYNAMIC;" +
             "TYPE_FILTER=FOLDERDOC";

DKNVPair parms[] = null;
DKDDO item = null;
int i = 0;
...
// ----- Execute the query; the result cursor is returned
dkResultSetCursor pCur = dsDL.execute(cmd,DK_CM_PARAMETRIC_QL_TYPE,parms);
// ----- Use a while loop to iterate thru the collection
while (pCur.isValid())
{
    pCur.setToNext();
    item = pCur.fetchObject();
    if (item != null)
    {
        i = pCur.getPosition();
    }
}
}
```

C++

```
DKString cmd = "SEARCH=(INDEX_CLASS=DLSAMPLE)";
cmd += "OPTION=(CONTENT=YES)";
cmd += "TYPE_QUERY=DYNAMIC";
cmd += "TYPE_FILTER=FOLDERDOC";

pCur = 0;
DKDDO *item = 0;
long i = 0;
...

dkResultSetCursor* pCur = dsDL.execute(cmd);
while (pCur->isValid()) {
    pCur->setToNext();
    item = pCur->fetchObject();
    if (item != 0) {
        i = pCur->getPosition();
        delete item;
    }
}
delete pCur;
```

Uma outra forma de executar esta operação:

Java

```
Object a = null;
pCur = dsDL.execute(cmd,DK_CM_PARAMETRIC_QL_TYPE,parms);
while (pCur.isValid()) {
    pCur.setPosition(DK_CM_NEXT,a);
    item = pCur.fetchObject();
    if (item != null) {
        i = pCur.getPosition();
    }
}
}
```

C++

```
DKAny a;
pCur = dsDL.execute(cmd);
while (pCur->isValid()) {
    pCur->setPosition(DK_CM_NEXT,a);
    item = pCur->fetchObject();
    if (item != 0) {
        i = pCur->getPosition();
        delete item;
    }
}
delete pCur;
```

Poderá utilizar um posicionamento relativo quando iterar através dos artigos. No exemplo seguinte, é ignorado um artigo ou outro no cursor do conjunto de resultados.

Java

```
Object a = null;
pCur = dsDL.execute(cmd,DK_CM_PARAMETRIC_QL_TYPE,parms);
a = new Integer(2);
while (pCur.isValid()) {
    pCur.setPosition(DK_CM_RELATIVE,a); // move cursor 2 positions forward
    item = pCur.fetchObject();          // from the current position
    if (item != null) {                  // (relative)
        i = pCur.getPosition();
    }
}
}
```

C++

```
DKAny a;
long increment = 2;
pCur = dsDL.execute(cmd);
a = increment;
while (pCur.isValid()) {
    pCur->setPosition(DK_CM_RELATIVE,a);
    item = pCur->fetchObject();
    if (item != 0) {
        i = pCur->getPosition();
        delete item;
    }
}
delete pCur;
```

Criar uma recolha a partir de um cursor do conjunto de resultados

Poderá utilizar o cursor do conjunto de resultados para inserir numa recolha um número específico de artigos do cursor do conjunto de resultados. O primeiro parâmetro do método `fetchNextN` especifica quantos artigos devem ser colocados na recolha. Colocar um zero no primeiro parâmetro para indicar que todos os artigos serão colocados na recolha.

No exemplo seguinte, todos os artigos do cursor do conjunto de resultados são capturados para uma recolha sequencial. Caso fItems seja Verdadeiro, é devolvido pelo menos um artigo.

Java

```
DKSequentialCollection seqColl = new DKSequentialCollection();
boolean fItems = false;
int how_many = 0;
fItems = pCur.fetchNextN(how_many, seqColl);
```

C++

```
DKSequentialCollection seqColl;
DKBoolean fItems = FALSE;
long how_many = 0;
fItems = pCur->fetchNextN(how_many, seqColl);
```

Consultar recolhas

Uma *recolha passível de consulta* é uma recolha que poderá ser consultada depois, logo fornece um conjunto mais reduzido e resultados mais precisos. Uma implementação concreta de uma recolha passível de consulta é um objecto de DKResults, devolvido como o resultados de uma avaliação da consulta. DKResults é uma subclasse de dkQueryableCollection e é uma recolha de DDOs.

Obter o resultado de uma consulta

O exemplo seguinte ilustra como emitir uma consulta paramétrica e obter resultados. Os resultados estão em rs, sendo este um objecto de DKResults. Poderá utilizar exemplos de códigos anteriores para processar a recolha e obter o DDO.

Java

```
// ----- Create a datastore and establish a connection
DKDatastoreDL dsDL = new DKDatastoreDL();
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");

// ----- Create and execute a query object
String query1 = "SEARCH=(INDEX_CLASS=GRANDPA, COND=(Title <> null));";
DKParametricQuery pq =
    (DKParametricQuery) dsDL.createQuery(query1, DK_CM_PARAMETRIC_QL_TYPE, null);
pq.execute();
// ----- Get the result
DKResult rs = (DKResults) pq.result();
```

C++

```
// establish a connection
DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");
// create a query object
DKString query1 = "SEARCH=(INDEX_CLASS=GRANDPA,COND=(Title <> NULL));";
DKParametricQuery* pq =(DKParametricQuery*)
dsDL.createQuery(query1,DK_PARAMETRIC_QL_TYPE, NULL);
pq->execute();
DKAny any = pq->result();
DKResult* rs = (DKResults*) any.value();
```

Avaliar uma nova consulta

Poderá consultar o resultado a partir de uma consulta para depois o aperfeiçoar. O código seguinte, baseado no exemplo anterior, demonstra a reavaliação de uma consulta:

Java

```
String query2 = "SEARCH=(INDEX_CLASS=GRANDPA, COND=(Subject == 'Mystery'))";
Object obj = rs.evaluate(query2,DK_CM_PARAMETRIC_QL_TYPE, null);
....
```

C++

```
DKString query2 = "SEARCH=(INDEX_CLASS=GRANDPA, COND=(Subject == 'Mystery'))";
any = rs->evaluate(query2,DK_PARAMETRIC_QL_TYPE, NULL);
...
```

A segunda consulta devolve obj, um objecto de DKResults que contém os resultados aperfeiçoados. Os resultados combinados de ambas as consultas seriam equivalentes a:

```
"SEARCH=(INDEX_CLASS=GRANDPA, COND=(Title <> NULL AND Subject == 'Mystery'))";
```

Poderá repetir a consulta até obter resultados satisfatórios. Após iniciar um tipo de consulta, as consultas seguintes têm de ser do mesmo tipo. Se misturar tipos de consultas, o resultado pode ser nulo.

O exemplo seguinte demonstra consultas de texto sequenciais:

Java

```
DKDatastoreTS dsTS = new DKDatastoreTS();
dsTS.connect("TM","","","");

// ----- A primeira consulta
String tquery1 = "SEARCH=(COND=(IBM)); OPTION=(SEARCH_INDEX=TMINDEX)";
DKTextQuery tq =
    (DKTextQuery) dsTS.createQuery(tquery1, DK_CM_TEXT_QL_TYPE, null);
tq.execute();
DKResults trs = (DKResults) tq.result();
// ----- A segunda consulta
String tquery2 = "SEARCH=(COND=(Tivoli)); OPTION=(SEARCH_INDEX=TMINDEX)";
Object obj = trs.evaluate(tquery2, DK_CM_TEXT_QL_TYPE, null);
```

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM","","","");

DKString tquery1 = "SEARCH=(COND=(IBM)); OPTION=(SEARCH_INDEX=TMINDEX)";
DKTextQuery* tq =
    (DKTextQuery*) dsTS.createQuery(tquery1,DK_TEXT_QL_TYPE, NULL);
tq->execute();
any = tq->result();
DKResults* trs = (DKResults*) any.value();

DKString tquery2 = "SEARCH=(COND=(Tivoli));OPTION=(SEARCH_INDEX=TMINDEX)";
any = trs->evaluate(tquery2,DK_TEXT_QL_TYPE, NULL);
```

A segunda consulta devolve obj, um objecto de DKResults que contém os resultados aperfeiçoados. Os resultados combinados de ambas as consultas seriam equivalentes a:

```
"SEARCH=(COND=(IBM AND Tivoli));OPTION=(SEARCH_INDEX=TMINDEX)";
```

Utilizar a recolha passível de consulta em alternativa à consulta combinada

Uma consulta combinada fornece a flexibilidade de submeter uma combinação de consultas paramétricas e de texto, com ou sem âmbitos. No entanto, todas estas consultas terão de ser submetidas ao mesmo tempo e não uma de cada vez, como faria se estivesse a avaliar uma recolha passível de consulta.

Uma consulta combinada devolve um objecto de DKResults; no entanto, não pode avaliar outra consulta paramétrica contra esta. Não poderá utilizar consultas combinadas em todos os servidores de conteúdos.

Avaliar uma recolha passível de consulta com consultas subsequentes dá ao utilizador a flexibilidade necessária para aperfeiçoar os resultados de uma consulta anterior, passo a passo, até obter um resultado final satisfatório. As consultas subsequentes são bastante úteis para procurar de forma dinâmica num servidor de conteúdos e formular a consulta seguinte com base nos resultados anteriores. No entanto, se já souber a totalidade da consulta, é mais eficaz submeter uma vez a consulta completa ou utilizar uma consulta combinada.

Trabalhar com o Content Manager Versão 8.2

Esta secção descreve as interfaces de programação de aplicações (APIs) do conector (conector de ICM) do Content Manager Versão 8 Edição 2. O conector de ICM é uma extensão da estrutura (EIP) do Enterprise Information Portal (EIP), por isso é essencial que o utilizador compreenda os conceitos de estrutura de EIP descrito s em “Conceitos de programação de aplicações do Enterprise Information Portal” na página 11 antes de começar a ler estas informações.

Pode utilizar as APIs do conector de ICM para construir e implementar aplicações personalizadas que acedem a um servidor de conteúdos do Content Manager. Pode também utilizar as APIs para integrar as aplicações existentes num servidor de conteúdos do Content Manager,

Esta secção contém as seguintes informações:

- Compreender o sistema Content Manager
- Compreender os conceitos do Content Manager
- Planear uma aplicação do Content Manager
- Criar uma aplicação de Content Manager
- Controlar o acesso a informações
- Processar transacções
- Pesquisar artigos

Compreender o sistema Content Manager

Os componentes principais do sistema Content Manager incluem um servidor de bibliotecas, um ou mais objectos de gestor de recursos e um conjunto de interfaces de programação de aplicações (APIs) orientadas para objectos. Para administrar o sistema Content Manager, é facultado também ao utilizador uma cliente de administração de sistemas com base em-Java.

O servidor de bibliotecas fornece-lhe capacidades de exemplificação de dados flexíveis, acesso protegido ao seu sistema, gestão eficiente do conteúdo e outras funções. O servidor de bibliotecas gere as relações entre artigos no sistema e controla o acesso a todas as informações do sistema, incluindo as informações armazenadas nos gestores de recursos configurados no sistema.

O gestor de recursos é o componente que armazena o conteúdo actual de qualquer objecto binário, como uma imagem digitalizada, um documento de office ou um video. Pode integrar outros gestores de recursos, como, por exemplo, Content Manager VideoCharger ou outros produtos que não sejam IBM, no sistema do Content Manager. Com o gestor de recursos poderá concluir as seguintes tarefas:

- Mova automaticamente o conteúdo do dispendioso suporte de dados de alta velocidade para um suporte de dados mais lento menos dispendioso utilizando o System Managed Storage (SMS).
- Aceda ao gestor de recursos directamente a partir do browser da Web.
- Obtenha a totalidade ou parte de um objecto.
- Sincronize os seus dados com o servidor de bibliotecas.

As APIs facultam aplicações com acesso ao sistema do Content Manager. As APIs estão disponíveis para Java e C++. Utilizando as APIs, as aplicações do utilizador podem tirar partido de toda a funcionalidade do Content Manager como, modelação de dados, pesquisa de texto e pesquisa paramétrica integradas, acesso e facultamento de dados de terceiros, etc.

O diagrama de Figura 9 ilustra como os componentes do sistema se encaixam. Tenha presente que está é apenas uma implementação de um sistema Content Manager. Noutra configuração de sistema, o utilizador poderá ter quatro gestores de recursos, por exemplo.

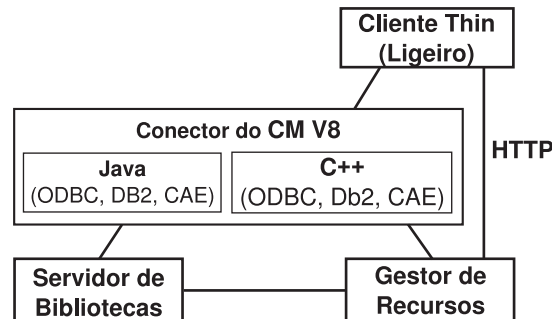


Figura 9. Configuração do sistema

Compreender os conceitos do Content Manager

Esta secção descreve conceitos importantes do Content Manager. É importante que compreenda os conceitos do Content Manager antes de prosseguir com as tarefas de programação. As informações descritas nesta secção incluem:

- Artigos.
- Atributos.
- Tipos de artigo.
- Componentes raiz e descendentes.
- Objectos.
- Ligações e Referências.
- Documentos.
- Pastas.
- Elaboração de versões.
- Controlo de acesso.
- Modelo de dados de gestão de documentos.

Artigos

Um artigo é a entidade básica gerida pelo servidor de bibliotecas. Os exemplos de artigos incluem uma política, participação, número de telefone, etc. Um *artigo* é um termo genérico para uma ocorrência de um tipo de artigo. Caso um objecto seja uma peça discreta de conteúdo digital, então um artigo é uma representação desse objecto. O artigo não é o objecto, mas identifica de forma precisa como e onde o localizar. No sistema, os artigos representam objectos incluindo documentos e pastas. Para definir objectos empresariais, como um documento, trabalhe com as definições do artigo.

Quando uma aplicação cria um artigo, o Content Manager atribui ao artigo vários atributos definidos pelo sistema e permite ao utilizador definir os seus próprios atributos.

Os atributos definidos pelo sistema incluem uma marca de hora de criação e um identificador de artigo (ID do artigo). O ID do artigo é único para cada artigo. O ID do artigo é armazenado pelo Content Manager e utilizado para localizar o artigo dentro do servidor de bibliotecas. Ao gravar a aplicação, utilizará o ID do artigo para aceder a todos os dados associados com esse artigo.

Atributos

Um *atributo* é uma unidade de dados que descreve uma determinada característica ou propriedade (por exemplo, nome, endereço, idade, etc.) de um artigo e pode ser pesquisada de modo a localizar o artigo.

Pode agrupar atributos para constituir grupos de atributos. Por exemplo, o atributo endereço pode ser constituído por um grupo de atributos que incluem a rua, cidade, distrito e código postal.

O utilizador pode também definir atributos que possuem vários valores. Estes atributos são denominados atributos de vários valores, que são implementados como componentes descendentes. Por exemplo, pode armazenar vários endereços, o endereço particular, o endereço do local de trabalho, etc., para um proprietário de política.

Para obter informações adicionais, consulte o exemplo `SAttributeDefinitionCreationICM`.

Tipos de artigos

Um *tipo de artigo* (classe de índice em versões anteriores do Content Manager) é, essencialmente, um modelo para definir e, posteriormente, localizar artigos semelhantes. Um tipo de artigo é formado por um componente de raiz, zero ou mais componentes descendentes e uma classificação. Um tipo de artigo é a estrutura geral que contém todos os componentes e dados associados. Por exemplo, num caso de seguro, um tipo de artigo de apólice contém artigos com atributos como, por exemplo, o número da apólice, o nome, participação, etc.

No Content Manager, existem quatro classificações de tipos de artigo: não de recurso, de recurso, documento (também denominados modelo de documentos) e parte de documento. Um tipo de artigo que não é de recurso representa entidades que não se encontram armazenadas num gestor de recursos. Um tipo de artigo de recurso representa objectos armazenados num gestor de recursos como, por exemplo, ficheiros num sistema de ficheiros, vídeo clips num servidor de vídeo, LOBs (objectos volumosos) em tabelas de base de dados, etc. Um tipo de artigo de documento representa entidades que contêm partes de documentos, que por sua vez contêm conteúdos de recurso, tal como um tipo de artigo de recurso. Um tipo de artigo de parte de documento representa objectos armazenados num gestor de recursos, mas são partes de um documento, contidas e detidas por um tipo de artigo de documento. O Content Manager faculta um conjunto base de tipos de artigo de recurso: objectos de LOB, de texto, imagem, sequência e de vídeo.

Para obter mais informações, consulte o exemplo `SItemTypeCreationICM`.

Componentes de raiz e descendentes

Um tipo de artigo é composto por componentes: um componente raiz e qualquer número de componentes descendentes, que são opcionais.

Um *componente de raiz* é o primeiro ou único nível de um tipo de artigo hierárquico. Um tipo de artigo é formado por atributos definidos pelo sistema e pelo utilizador. Internamente, o tipo de artigo mais básico contém apenas um componente.

Um *componente descendente* é um segundo nível opcional(ou um nível mais baixo) de um tipo de artigo hierárquico. Cada componente descendente está directamente associado ao seu nível superior. A Figura 10 apresenta o diagrama do meta-modelo do Content Manager. Apresenta os componentes raiz e descendente e as suas relações na formação da hierarquia de um artigo. O diagrama também mostra ligações, referências, artigos de recurso e objectos de recurso.

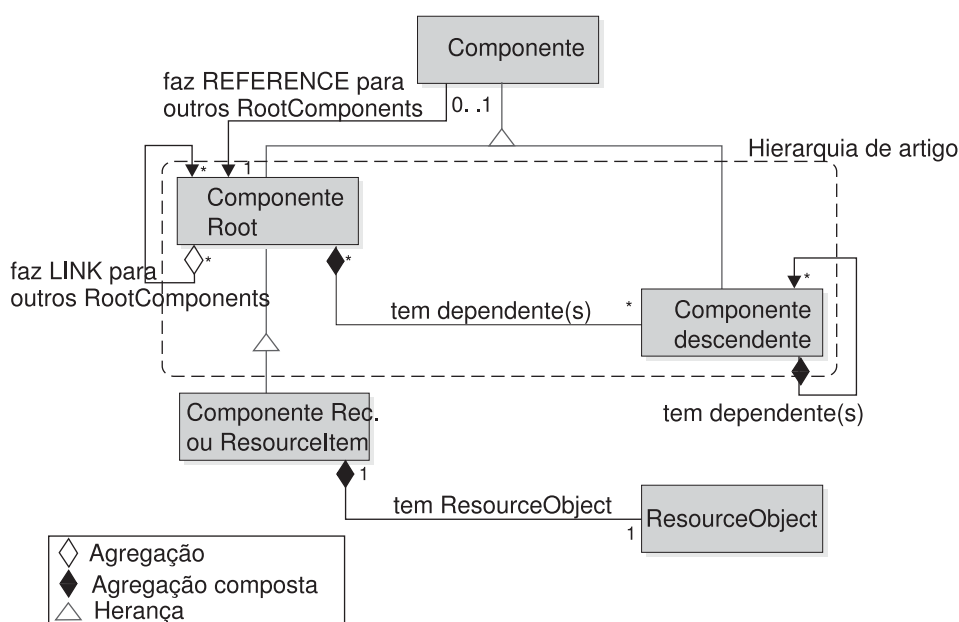


Figura 10. O meta-modelo de Content Manager: Uma apresentação lógica.

A Figura 11 na página 133 apresenta um exemplo do modelo de dados para uma aplicação de seguro com a apólice como componente raiz, tendo a participação, o relatório da Polícia e a estimativa de danos como componentes descendentes.

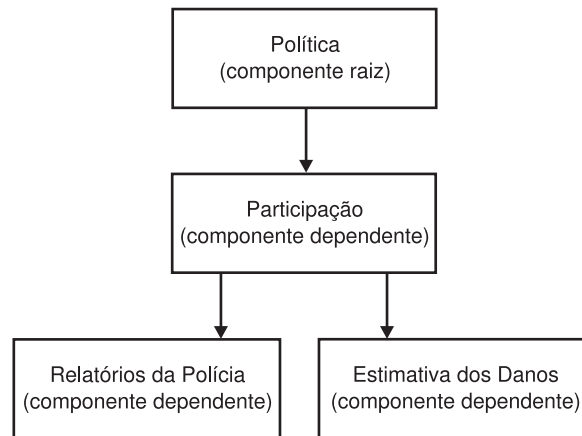


Figura 11. Hierarquia de componentes

Para obter mais informações relativas aos conceitos de modelação de dados, do Content Manager, consulte o exemplo `SItemTypeCreationICM`.

Objectos

Um *objecto* (também denominado por conteúdo de recurso) é qualquer conteúdo digital que um utilizador pode armazenar, obter e manipular como uma única unidade (por exemplo, imagens de JPEG, áudio MP3, vídeo AVI e um bloco de texto de um livro). Um objecto é sempre armazenado num gestor de recursos. O acesso a um objecto é controlado pelo servidor de bibliotecas.

Para obter mais informações, consulte o exemplo `SResourceItemCreationICM`.

Ligações e Referências

Pode utilizar uma *ligação* para modelar uma ou várias associações entre artigos. Tal como foi demonstrado na Figura 10, as ligações apenas podem ser utilizadas para associar componentes raiz de artigos. As ligações não estão relacionadas com uma versão específica de um artigo. Pode definir um número indeterminado de ligações. As ligações são uma forma flexível de ligar uma fonte a um destino. Uma ligação apenas associa dois artigos e faculta formas de aceder aos artigos que liga ou de aceder a artigos que podem estabelecer ligação a esses dois artigos. A utilização de ligações é determinada no tempo de execução pela aplicação do utilizador. Pode utilizar um número indeterminado de ligações na sua aplicação.

As ligações são blocos de construção flexíveis, abertos e irrestritos, que pode utilizar nas suas aplicações. Como tal, o utilizador tem como opção colocar restrições a ligações da sua aplicação.

Uma das formas através das quais pode utilizar uma ligação é representando a utilização de pastas ou a relação contentor-contido. Se optar por implementar a utilização de pastas recorrendo a ligações, tenha em atenção que o contentor não *possui* o contido, o que significa que os artigos do contentor não são eliminados quando o contentor é eliminado. Para obter mais informações relativas a ligações, consulte o exemplo `SLinksICM`.

Uma *referência* encontra-se entre um componente (quer seja raiz ou descendente) e outro componente raiz. Uma referência é representada como um atributo de referência num componente definido na altura da concepção. Uma definição de componente pode possuir qualquer número, especificado durante a definição do

tipo de componente, de atributos de referência que remetem para outros componentes raiz. Uma referência normalmente não indica propriedade, mas o utilizador pode implementar uma relação de propriedade, caso seja necessário.

Ao adicionar uma referência a um tipo de componente, os artigos desse tipo de componente podem fazer referência a outro artigo. Em termos do DDO, este possui um atributo que é identificado pelo nome da referência. O valor do atributo pode ser definido para outro DDO. O valor do atributo para DDO consiste no DDO que é referido pela referência.

As referências podem ser definidas tanto nos tipos de componentes raiz e descendente, fazendo referência a outro tipo de componente raiz. Consulte Figura 10 na página 132. As referências podem também referir uma versão específica de um artigo, enquanto as ligações fazem referência a todas as versões. Para obter mais informações relativas a atributos de referência, consulte o exemplo `SReferenceAttrDefCreationICM`.

Documentos

Existem dois tipos de documentos que poderá possuir no sistema. O primeiro tipo de documento é um artigo de tipo semântico "documento," que deve conter informações que componham um documento. Este tipo de "documento" pode ser autónomo ou conter partes, caso tenha implementado o modelo de documentos no seu modelo de dados. Para obter mais informações relativas a este tipo de documento, consulte o Exemplo de Instrução de API `SItemCreationICM`.

O outro tipo de documento é um artigo criado a partir de um tipo de artigo classificado como "documento" (também denominado como "modelo de documentos"). este tipo de documento contém partes de documento, uma implementação específica do modelo de documentos do Content Manager. As partes de documento podem incluir vários tipos de conteúdos, incluindo, por exemplo, texto, imagens e folhas de cálculo. Para obter mais informações relativas ao modelo de documentos, consulte o exemplo `SDocModelItemICM`.

Pastas

Uma *pasta* é um artigo que pode conter outros artigos de qualquer tipo. Isto implica que poderá também incluir outras pastas. No Content Manager Versão 8, o conceito de pastas é implementado utilizando relações de ligação entre artigos. Os artigos podem conter outros artigos para formar uma hierarquia de contenção, denominada hierarquia de pastas. Por exemplo, um artigo de políticas pertence ao tipo de artigo de políticas e, potencialmente, possui várias participações, fazendo das políticas uma pasta que detém outros artigos como, por exemplo, uma fotografia, um número de segurança social, etc. As pastas são muito flexíveis porque um artigo qualquer pode ser uma pasta e pode conter um número qualquer de outros artigos. Para obter mais informações, consulte o exemplo `SFolderICM`.

Elaboração de versões

A elaboração de versões é a capacidade de armazenar e manter várias versões de um artigo, incluindo versões dos componentes descendentes do artigo. O utilizador especifica as regras de elaboração de versões quando define um tipo de artigo. Se um tipo de artigo for activado para elaboração de versões, serão elaboradas versões de todos os artigos desse tipo de artigo.

Existem dois tipos de elaboração de versões: constante ou por aplicação. Quando um artigo é activado para elaborar constantemente versões, é automaticamente criada uma nova versão do artigo sempre que o artigo é actualizado e armazenado no servidor de conteúdos. Quando um artigo é activado para elaboração de versões por aplicação, o sistema apenas cria uma nova versão quando especificado pela aplicação do utilizador.

A elaboração de versões é processada pelo servidor de bibliotecas do Content Manager. Cada versão de um artigo, cujo conteúdo é armazenado no gestor de recursos, possuirá a sua própria cópia do conteúdo de RM. As características importantes do suporte de elaboração de versões incluem:

- A elaboração de versões envolve um componente raiz e toda a sua hierarquia.
- Os tipos de artigos podem ter uma das três políticas de elaboração de versões possíveis: elaborar sempre versões, nunca elaborar versões (predefinição) e elaboração de versões controlada pela aplicação.
- Todas as versões de um artigo do sistema do CM são pesquisáveis e passíveis de serem obtidas.
- Todas as versões de um artigo podem ser actualizadas e eliminadas.
- Para tipos de artigo com elaboração de versões controlada pela aplicação, quando o artigo é actualizado, o utilizador pode optar entre aplicar as actualizações à versão existente ou criar uma nova versão com base nas actualizações.
- Cada versão de um artigo possui o seu identificador permanente (PID). O PID possui várias partes, das quais duas são relevantes no contexto actual. A primeira parte relevante é o ID do Artigo, que é igual em todas as versões diferentes do artigo. A outra parte é o número da versão. Todas as versões do artigo possuem um número de versão diferente que pode ser obtido e definido como uma cadeia. De seguida encontra-se um exemplo que demonstra como trabalhar com números de versões.

```
DKPidICM pid = (DKPidICM)ddo.getPidObject();
Versão da cadeia = pid.getVersionNumber();
....
pid.setVersionNumber(version);
```
- Um tipo de artigo pode ser configurado para manter apenas um número limitado de versões de cada artigo. Se a actualização de um artigo exceder o número máximo permitido, a versão guardada mais antiga é largada e é criada uma nova versão por parte do sistema.
- Se um artigo activado para versões for novamente indexado, todas as versões anteriores do artigo são automaticamente eliminadas.
- Os componentes descendentes de um artigo herdam a versão do componente ascendente.
- A versão de um tipo de componente descendente não pode ser alterada, já que este segue a elaboração de versões do seu tipo ascendente.
- As regras de elaboração de versões nível de parte podem ser obtidas no objecto de relação do tipo de artigo que representa os tipos.

Para obter informações relativas à elaboração de versões, consulte os exemplos `SItemUpdateICM` e `SItemTypeCreationICM`.

Controlo de acesso

O modelo de controlo de acesso do Content Manager é formado pelos seguintes elementos fundamentais:

- Privilégios e conjuntos de privilégios.

- Entidades controladas.
- Utilizadores e grupos de utilizadores.
- Listas de controlo de acesso.

Os vários elementos de controlo de acesso funcionam de acordo com a seguinte descrição. A cada utilizador do Content Manager é concedido um conjunto de privilégios de utilizador. Estes privilégios definem a operação que um utilizador pode executar. Os direitos de acesso efectivos de um utilizador nunca excederão os privilégios definidos pelo utilizador.

O modelo de controlo de acesso do Content Manager é aplicado à entidade controlada. Uma *entidade controlada* é uma unidade de dados protegidos do utilizador. No Content Manager, a entidade controlada pode estar nível de um artigo, de tipo de artigo, ou ao nível de toda a biblioteca. Por exemplo, pode associar uma ACL a um tipo de artigo para reforçar o controlo de acesso a nível do tipo do artigo. As operações sobre entidades controladas são reguladas por uma ou mais regras de controlo, denominadas listas de controlo de acesso (ACLs). Todas as entidades controladas no sistema do Content Manager devem estar ligadas a uma ACL.

Quando um utilizador indica uma operação sobre um artigo, o sistema verifica o privilégio do utilizador e a ACL associada ao artigo para determinar se o utilizador tem direito de efectuar uma operação no artigo. Evidentemente, o direito de aceder a um artigo também requer o direito de aceder ao tipo de artigo em que o artigo se encontra definido. Figura 12 demonstra um exemplo de como o sistema determina os direitos de acesso do utilizador a um artigo com base nos privilégios e ACLs.

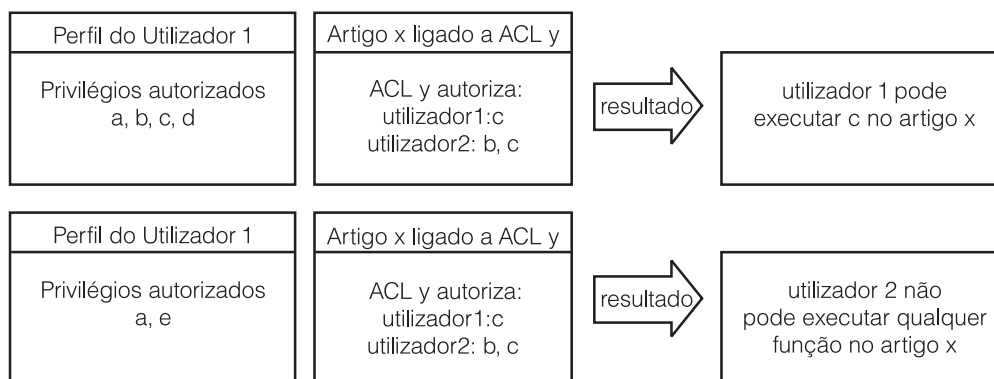


Figura 12. Diagrama de controlo de acesso

Privilégios e conjuntos de privilégios

Os privilégios permitem a um utilizador executar uma acção específica num artigo do sistema como, por exemplo, criá-lo ou eliminá-lo. A cada utilizador do Content Manager é concedido um conjunto de privilégios de utilizador. Os privilégios definem o máximo de operações que um utilizador pode executar em informações do sistema Content Manager. Os direitos de acesso de um utilizador não excedem os privilégios definidos pelo utilizador.

O Content Manager faculta um determinado número de privilégios predefinidos que o utilizador não pode alterar, denominados privilégios definidos pelo sistema. Pode também definir os privilégios do utilizador, denominados privilégios definidos pelo utilizador. O utilizador aplica os privilégios definidos pelo utilizador na aplicação utilizando as rotinas de saída do utilizador.

Todos os privilégios possuem um código único gerado pelo sistema, denominado código de definição de privilégios. Os códigos de definição de privilégios entre 0 e 999 estão reservados para privilégios definidos pelo sistema. Pode utilizar códigos de 1000, e acima de 1000, para privilégios definidos pelo utilizador.

Os privilégios definidos pelo sistema são classificados em duas categorias: privilégios de administração do sistema e privilégios de acesso a dados. Pode utilizar os privilégios de administração do sistema para modelar dados do utilizador e administrar e manter o sistema Content Manager. O utilizador necessita de privilégios de administração para concluir tarefas como, por exemplo, configurar o sistema, gerir a configuração de servidor de bibliotecas e gerir tipos de artigo. Pode utilizar os privilégios de acesso a dados para aceder e alterar os dados do sistema como, por exemplo, artigos e tipos de artigos.

Um grupo de privilégios atribuído a um utilizador é um *conjunto de privilégios*. Por exemplo, um conjunto de privilégios pode conter os privilégios de criação, actualização e eliminação. Os conjuntos de privilégios permitem uma administração do sistema mais fácil. O utilizador deve agrupar privilégios num conjunto antes de os utilizar. Não existe um limite de privilégios que um conjunto pode conter.

Os conjuntos de privilégios predefinidos do Content Manager incluem: privilégio Administração do Sistema

AllPrivSet; PrivSetCode: 1

Um utilizador com este conjunto de privilégios pode executar todas as funções em todas as entidades do Content Manager. Os privilégios contidos neste conjunto incluem: Todos os privilégios definidos pelo sistema e pelo utilizador.

NoPrivSet; PrivSetCode: 2

Os utilizadores que possuam este conjunto de privilégios não podem executar quaisquer acções em entidades do Content Manager. Os privilégios contidos neste conjunto incluem: Nenhum.

SystemAdminPrivSet; PrivSetCode: 3

Os utilizadores com este conjunto de privilégios podem executar todas as funções de administração do sistema e modelação de dados do Content Manager. Os privilégios contidos neste conjunto incluem:

ItemAdminPrivSet; PrivSetCode: 4

Os utilizadores com este conjunto de privilégios podem executar todas as funções de modelação de dados e acesso a artigo do Content Manager. Os privilégios contidos neste conjunto incluem:

- Privilégio de tipo de artigo definido pelo sistema
- Privilégio de selecção de SQL de artigo
- Privilégio de consulta de tipo de artigo
- Privilégios de consulta de artigo
- Privilégio de adição de artigo
- Privilégio de atributo definido pelo utilizador de conjunto de artigos
- Privilégio de atributo definido pelo sistema de conjunto de artigos
- Privilégio de eliminação de artigo
- Privilégio de deslocação de artigo
- Privilégio de artigo de ligação a
- Privilégio de artigo ligado

- Privilégio de artigo detentor
- Privilégio de artigo detido
- Privilégio de artigo de adição de ligação
- Privilégio de artigo de alteração de ligação
- Privilégio de artigo de remoção de ligação
- Privilégio de saída de artigo

Tabela 8. Códigos dos Privilégios

| Nome do privilégio | Código |
|----------------------|--------|
| ICMLogon | 1 |
| SystemAdmin | 40 |
| SystemDefineItemType | 45 |
| ItemSQLSelect | 121 |
| ItemTypeQuery | 122 |
| ItemQuery | 123 |
| ItemAdd | 124 |
| ItemSetUserAttr | 125 |
| ItemSetSysAttr | 126 |

Utilizadores e grupos de utilizadores

É quase certo que o utilizador tem um grupo de utilizadores que exigem o mesmo tipo de acesso ao sistema. Por exemplo, todos os seguradores de uma empresa de seguros necessitam de privilégios de pesquisa, obtenção e actualização para o tipo de artigo de participações. Poderá agrupar os seguradores e outros utilizadores com necessidades comuns de acesso num grupo de utilizadores. No entanto, não poderá colocar um grupo de utilizadores noutro grupo de utilizadores.

Um grupo de utilizadores é apenas um agrupamento de conveniência de utilizadores individuais que executam tarefas semelhantes. Um grupo de utilizadores é formado por nenhum ou mais utilizadores. Não tem de atribuir um conjunto de privilégios a um grupo de utilizadores. Cada utilizador num grupo de utilizadores possui um conjunto de privilégios. Um grupo de utilizadores facilita a criação de listas de controlo do acesso a objectos no seu sistema. Um grupo de utilizadores não pode pertencer a outros grupos.

Listas de controlo de acesso (ACLs)

Quando um utilizador cria um artigo no sistema Content Manager, esse utilizador deve definir o acesso que outros utilizadores possuirão a esse artigo e quais as operações que nele poderão executar. A lista de utilizadores que possuem acesso ao artigo e as operações que estes podem executar no artigo são denominadas *lista de controlo de acesso* (ACL). Uma ACL é uma lista que contém num ou mais IDs de utilizador individual ou grupos de utilizadores e os seus privilégios associados. O utilizador pode associar artigos, tipos de artigos e listas de trabalho a uma ACL. Os conjuntos de privilégios definem a capacidade máxima de utilização do sistema de um utilizador individual, uma ACL restringe o acesso desse utilizador individual a um artigo. Por exemplo, se a ACL permitir que o artigo fotografia seja eliminado mas o João não tem o privilégio de eliminação no seu conjunto de privilégios, então o João não pode eliminar a fotografia.

Uma entidade controlada está associada a uma ACL específica através do código de ACL. Quando associadas a entidades controladas, as ACLs definem a

autorização das entidades associadas e não englobam os privilégios do utilizador. É aplicada uma ACL e os privilégios de utilizador são verificados.

Os utilizadores especificados nas regras de controlo de acesso podem ser utilizadores individuais, grupos de utilizadores ou públicos. A interpretação é determinada pelo campo Tipo de Utilizador de uma regra. Os tipos de regras, por uma questão de ilustração, podem possuir os nomes Regra de ACL para Utilizador e Regra de ACL para Grupo e Regra de ACL para Público, respectivamente. Ao especificar público, a Regra de ACL para Público autoriza todos os utilizadores a executar operações especificadas nos Privilégios de ACL na entidade associada, desde que o utilizador passe na verificação de Privilégios do Utilizador. Os privilégios de ACL na entidade limite em Público podem ser configurados no nível Sistema. A capacidade de abrir uma entidade limite em Público pode ser configurada a nível do sistema. O parâmetro de configuração denomina-se PubAccessEnabled (definido na tabela ICMSTSysControl). Quando desactivadas, todas as Regras de ACL para Público são ignoradas durante o processo de controlo de acesso.

Dentro da mesma ACL, um utilizador pode ser especificado em mais do que um tipo de regra. A prioridade dos três tipos, do mais alto ao mais baixo, é Regra de ACL para Público, Regra de ACL para Utilizador e Regra de ACL para Grupo. Ao aplicar verificações de ACL, se passar um tipo de regra de prioridade superior, a autorização é processada e o processo pára. Se a verificação de Regra de ACL para Público tiver falhado, o seguinte processo irá continuar nos tipos de regra de prioridade inferior.

Se a verificação de Regra de ACL para Utilizador tiver falhado, a verificação pára. A Regra de ACL para Grupo não é verificada. Não há necessidade de prosseguir com a verificação no tipo Grupo pois, caso um utilizador efectue uma verificação de utilizador individual, este será excluído do acesso ao tipo de grupo com base no algoritmo de controlo de acesso. A verificação de controlo de acesso de Tipo de utilizador individual e Tipo de grupo não é um processo sequencial. Trata-se de uma situação com duas opções, apesar de não ser prejudicial a realização de uma verificação sequencial.

Se o utilizador não tiver passado na verificação de tipo de utilizador individual (ou o utilizador não possuir uma regra na tabela Lista de Acessos), o processo de verificação irá avançar para o tipo de grupo. Se o utilizador pertencer a um dos grupos e a verificação do privilégio passar, a autorização é processada e o processo pára. Caso contrário, é negado acesso e o processo pára igualmente. Quando se especifica um utilizador em mais do que uma Regra de ACL para um Grupo, o utilizador é autorizado pela união de todos os Privilégios de ACL dessas regras. Nunca se especifica um utilizador em mais do que uma regra de ACL para Utilizador.

O sistema do CM faculta as seguintes ACLs pré-configuradas: SuperUserACL , NoAccessACL e PublicReadACL.

SuperUserACL

Esta ACL é formada por uma única regra que autoriza o utilizador pré-configurado ICMADMIN do CM a executar todas as funções de CM (AllPrivSet) nas entidades associadas.

NoAccessACL

Esta ACL é formada por uma única regra que não permite, a todos os utilizadores de CM (público), a execução de acções (NoPrivSet).

PublicReadACL

Esta ACL é formada por uma única regra que permite, a todos os utilizadores de CM (ICMPUBLIC), a capacidade de leitura (ItemReadPrivSet). Este é o valor predefinido atribuído ao DfltACLCode de um utilizador.

Planear uma aplicação do Content Manager

Esta secção ajuda-o a identificar os requisitos para a criação de uma aplicação do Content Manager e fornece informações sobre a forma de operar do Content Manager. Uma parte essencial do planeamento da sua aplicação está em criar um modelo de dados que corresponde às necessidades da sua empresa. Para obter mais informações acerca do modelo de dados do Content Manager, consulte *Planear e Instalar o Content Management System* e o exemplo `tSItemTypeCreationICM` no directório de exemplos.

Os tópicos seguintes são abordados nesta secção:

- Determinar as funções da sua aplicação.
- Tratamento de erros.

Determinar as funções da sua aplicação

A abordagem que tiver para desenvolver a sua aplicação varia com base nas necessidades da sua organização. Para produzir uma aplicação efectiva, todas as partes interessadas na sua organização devem contribuir para o planeamento e concepção da aplicação. Para obter ajuda adicional relativa ao planeamento, consulte *Planear e Instalar o Content Management System*.

Antes de poder criar a aplicação, deverá poder responder a todas ou a maior parte das seguintes questões:

- Quais os tipos de documentos que a sua organização utiliza?
- Que tipos de conteúdo têm os documentos existentes?
- Como processa documentos?
- Pode automatizar o seu processamento de documentos?
- Como recebe, apresenta, armazena e distribui os documentos?
- Com que frequência obtém os documentos após serem armazenados?
- Qual é o volume de documentos que a sua organização gere?
- Quais os tipos de suporte de memória que pretende utilizar para armazenar os seus objectos grandes?
- Existem outras aplicações que a sua organização utilize?
- Quantos utilizadores e qual o tipo de controlo de acesso que pretende utilizar?

Utilize as respostas para as perguntas colocadas acima para o ajudar a determinar quais as funções a incluir na sua aplicação.

Tratamento de erros

Ao tratar erros, a excepção mais importante a deter é a classe `DKException`. Não utilize excepções para lógicos de programa e não tenha em conta excepções de detenção para detectar se existe algo no servidor de conteúdos ou noutra situação que não seja pontual. Utilizar excepções no lógico do programa diminui o rendimento e pode inutilizar as informações de rastreio e registo para a depuração e suporte.

Reveja cuidadosamente todas as informações da excepção. Existem várias sub-classes de `DKException` e dependendo do programa, o mais indicado será tratar cada excepção individualmente. A Tabela 9 contém informações de `DKException`.

Tabela 9. Informações de `DKException`

| DKException | Descrição |
|------------------------------|---|
| Nome | Nome da Classe de Excepção. Contém o nome da sub-classe. |
| Mensagem | Uma mensagem específica explica o erro. A mensagem pode conter uma grande quantidade de informações, encapsulando, por vezes, estados importantes de variáveis na altura em que o erro foi detectado. |
| ID da Mensagem | Um ID de Mensagem único identifica este tipo de erro e estabelece uma correspondência entre este e uma mensagem de núcleo utilizada acima. |
| Estado do Erro | Poderá conter informações adicionais relativas ao estado da API de OO ou ao erro de servidor de bibliotecas. Se o servidor de bibliotecas detectar um erro, as seguintes quatro informações são agrupadas aqui: Código de retorno Código da razão Código de retorno de Ext / SQL Código da razão de Ext / SQL |
| Código do Erro | Poderá conter o código de retorno do servidor de bibliotecas. |
| Rastreio da Pilha de Memória | Informações importantes que indicam o ponto da falha no programa do utilizador e o local exacto onde o erro foi detectado ou tratado pela última vez pela API de OO. |

Ao trabalhar em Java, também deve tratar `java.lang.Exception`. O exemplo `SConnectDisconnectICM` do directório de exemplos demonstra como deter e imprimir erros. Para obter informações relativas ao registo e rastreio, consulte *Mensagens e Códigos*.

Trabalhar com exemplos de Content Manager

O Content Manager faculta um conjunto abrangente de exemplos de códigos para auxiliar o utilizador a concluir as tarefas essenciais do Content Manager. Os exemplos são uma ótima fonte de informações sobre API pois facultam informações de consulta, orientações de programação, exemplos de utilização de API e ferramentas.

Pode visualizar os exemplos na *Online application Programming Reference* no Centro de Informações do produto. Para além disso, os exemplos encontram-se nos directórios `CMBROOT\Samples\java\icm` e `CMBROOT\Samples\cpp\icm`. No entanto, tenha em conta que deve seleccionar o componente Exemplos e Ferramentas durante a instalação do EIP de modo a dispor dos exemplos no directório.

Para tirar o maior partido possível dos exemplos leia *Readme de Exemplos*. Contém um índice completo de consulta para o ajudar a encontrar rapidamente o exemplo que contém o conceito, ou tópico, que procura. Todos os exemplos são documentados com precisão e facultam informações conceptuais aprofundadas e uma explicação de todos os passos de uma tarefa. As informações adicionais contidas em cada exemplo incluem:

- Informações de cabeçalho detalhadas que explicam os conceitos apresentados no exemplo.
- Uma descrição do ficheiro de exemplos incluindo informações de pré-requisitos e utilização de linhas de comandos.
- Um código totalmente comentado que facilmente pode ser eliminado, personalizado e utilizado nas suas aplicações.
- Uma função de utilitário que pode utilizar ao desenvolver as suas aplicações.

A secção *Introdução* em *Readme de Exemplos* ajuda-o a aprender rapidamente a concluir as seguintes tarefas gerais:

- Modelação de dados.
- Estabelecer ligação a um servidor e tratar erros.
- Definir atributos e grupos de atributos.
- Trabalhar com atributos de referência.
- Definir o seu modelo de dados.
- Trabalhar com artigos.
- Trabalhar com artigos de recurso.
- Trabalhar com pastas.
- Trabalhar com ligações.
- Definir o gestor de recursos.
- Definir uma recolha SMS.
- Pesquisar artigos.

O exemplo de hipótese de seguro

O Content Manager faculta exemplos de códigos para uma possível implementação prática em que se recorre a uma seguradora. As informações utilizadas para criar o exemplo de seguradora são concebidas e criadas com o único intuito de ajudar a clarificar as funções essenciais do Content Manager. Para obter uma lista completa dos exemplos que compõem o cenário de seguro, consulte o *Readme de Exemplos*.

Criar uma aplicação do Content Manager

As APIs que implementam a funcionalidade do Content Manager Versão 8 Edição 2 estão agrupadas no que é denominado como conector de ICM. As APIs do conector de ICM possuem um sufixo de ICM, como no exemplo `DKDatastoreICM`.

As informações nesta secção incluem:

- Compreender os componentes de software.
- Representar artigos utilizando DDOs.
- Estabelecer ligação ao sistema de Content Manager.
- Trabalhar com artigos.

Compreender os componentes do software

Para fins conceptuais, pode categorizar as APIs de OO nos seguintes grupos de serviços:

- Modelação de documentos e dados.
- Pesquisa e obtenção.
- Importação e apresentação de dados.
- Gestão de sistemas.
- Encaminhamento de documentos.

O módulo de elaboração de modelos de dados e de documentos contém as APIs que lhe permitem definir as correspondências dos seus modelos de dados empresariais para o modelo de dados hierárquico do Content Manager subjacente. Por exemplo, o modelo de dados de uma companhia de seguros inclui *apólices*, em que o modelo de dados do Content Manager é essencialmente artigos. As APIs do módulo de modelação de dados e documentos facultam interfaces para definir artigos que representam *políticas*.

O módulo de pesquisa e obtenção processa pedidos relativos a artigos geridos como, por exemplo, documentos e pastas. As APIs do módulo de pesquisa permite-lhe executar pesquisas de texto e paramétricas combinadas para artigos que constam no sistema Content Manager. Os resultados da pesquisa são devolvidos à aplicação na forma de conjuntos de resultados da pesquisa.

O módulo de importação e entrega de dados faculta as APIs que permitem ao utilizador importar dados para o sistema e entregar esses dados através de vários meios de transmissão como, por exemplo, uma rede ou a Web.

O módulo de gestão do sistema fornece-lhe as interfaces para configurar e manter um sistema do Content Manager eficiente e protegido. Por exemplo, pode incorporar as APIs de gestão do sistema na sua aplicação de forma a poder ajustar as definições de controlo do sistema, gerir utilizadores, atribuir privilégios de utilizador, permitir acesso ao sistema e assim por diante.

As APIs do módulo de encaminhamento de documentos ajudam o utilizador a encaminhar objectos empresariais como, por exemplo, documentos, através de um processo, segundo as necessidades definidas pela empresa do utilizador.

Representar artigos através de DDOs

Antes de poder criar uma aplicação, o utilizador deve compreender os conceitos do protocolo de DDO/XDO explicados em “Compreender conceitos de objectos de dados dinâmicos” na página 12. As informações nesta secção são específicas do Content Manager Versão 8 Edição 2.

Um DDO é essencialmente um contentor de atributos. Um atributo tem um nome, valor e várias propriedades. Uma das propriedades mais importantes dos atributos é o tipo de atributo. Um DDO possui um identificador permanente (PID) para indicar onde o objecto reside no armazenamento permanente. Um DDO tem alguns métodos para se multiplicar e métodos correspondentes para obter as informações de um artigo. Os métodos de DDO incluem adicionar, obter, actualizar e eliminar. O utilizador recorre a estes métodos para colocar e retirar os dados de um artigo do armazenamento permanente como, por exemplo, Content Manager.

Na memória, os artigos do Content Manager são representados como DDOs. Os atributos do artigo são representados como atributos do DDO com um nome, tipo e valor. As ligações e referências são representadas como tipos especiais de atributos. No entanto, a diferença entre um atributo de ligação e um atributo de referência é que o atributo de referência remete para outro (único) DDO ou XDO, e um atributo de ligação remete para uma recolha (vários) de DDOs ou XDOs. Os XDOs são utilizados para representar objectos volumosos (LOBs).

Uma referência a um artigo, quer seja XDO ou outro DDO, tem um nome com a propriedade tipo definida como referência do objecto e o valor definido como a ocorrência do objecto referenciado. Os componentes descendentes e as ligações são também representados como atributos do DDO com a propriedade tipo definida

como uma recolha de objectos de dados e o valor definido como uma recolha de DDOs. No caso de um componente descendente, o nome do atributo é o nome do componente descendente. O valor é a recolha de componentes descendentes que pertencem ao componente de raiz. Se o artigo raiz for eliminado, todos os componentes de raiz do artigo raiz são também eliminados.

Ligação ao sistema do Content Manager

Uma das primeiras coisas que o utilizador deverá efectuar quando constrói uma aplicação do Content Manager é estabelecer ligação ao servidor. Esta secção auxilia o utilizador na execução de várias tarefas incluídas no estabelecimento e na conclusão da ligação a um servidor de Content Manager.

Para aceder a um servidor do Content Manager, a sua aplicação deve criar um servidor de conteúdos, que funciona como um servidor comum. Para criar e estabelecer ligação a um servidor de conteúdos:

1. Crie um objecto de servidor de conteúdos.

Java

```
DKDatastoreICM dsICM = new DKDatastoreICM();
```

C++

```
DKDatastoreICM *dsICM =new DKDatastoreICM();
```

2. Configure os parâmetros de ligação.

Java

```
String database = "icmnlsdb";  
String userName = "icmadmin";  
String password = "password";
```

C++

```
char * database = "icmnlsdb";  
char * userName = "icmadmin";  
char * password ="password";
```

3. Chame a operação de ligação no servidor de conteúdos. `databaseNameStr` é o nome da base de dados à qual pretende estabelecer ligação.

Java

```
dsICM.connect(databaseNameStr,usridStr,pwStr,"");
```

C++

```
dsICM->connect(database, userName, password, "");
```


Dependendo da configuração do seu sistema, poderá ter vários servidores de bibliotecas e gestores de recursos a que se pode ligar. Para obter uma lista de nomes dos servidores de bibliotecas aos quais pode estabelecer ligação, utilize `DKDatastoreICM`, chame o método `listDataSourceNames()` e, de seguida, chame o método `listDataSources()`. O método `listDataSources()` lista os servidores de bibliotecas que actualmente estão disponíveis para ligação.

Após estabelecer ligação a um servidor de bibliotecas, utilize `DKRMConfiguration` e chame o método `listResourceMgrs()` para obter uma lista de gestores de recursos associados a esse servidor de bibliotecas.

To disconnect from the system, call the `disconnect` operation in the content server.

Para obter mais informações, consulte o `SConnectDisconnectICM`, que consiste no exemplo completo do qual os snippets de código acima referidos foram extraídos.

Alterar uma palavra-passe

Pode permitir aos utilizadores a alteração da palavra-passe sempre que iniciem uma nova sessão do servidor de bibliotecas. To implement the change password option, use `DKDatastoreICM` and call the `changePassword()` method.

Java

```
changePassword(String userID, String oldPwd, String newPwd)
```

C++

```
changePassword(const char* userID, const char* oldPwd, const char* newPwd)
```

Trabalhar com artigos

Esta secção descreve os processos envolvidos na criação, actualização e eliminação de artigos.

For additional information about working with items see the `SItemCreationICM` sample.

Criar um tipo de artigo

Antes de criar quaisquer artigos, deve criar tipos de artigo. Deve definir um tipo de artigo para artigo que criar. Por exemplo, um artigo de participação é o componente descendente do tipo de item apólice

Quando cria um tipo de artigo, pode definir uma classificação para o tipo de artigo. Uma *classificação de tipo de artigo* é a categorização dentro de um tipo de artigo que aprofunda a identificação dos artigos desse tipo de artigo. Todos os artigos do mesmo tipo de artigo possuem a mesma classificação de tipo de artigo. O Content Manager fornece as seguintes classificações de tipo de artigo: artigo, artigo de recurso e parte de documento. Se não especificar uma classificação de tipo de artigo quando cria um tipo de artigo, a classificação do tipo de artigo assume a predefinição do artigo (DDO). As classificações de tipo de artigo predefinidas e as suas constantes de ID correspondentes encontram-se enumeradas em Tabela 10 na página 146.

Tabela 10. Classificações de tipo de artigo

| Classificação | Constante de ID | Número | Descrição |
|---------------------|---|--------|---|
| Item | DK_ICM_ITEMTY PE_CLASS_ITEM | 0 | Um artigo standard (DDO). |
| Recurso | DK_ICM_ITEMTY PE_CLASS_RES OURCE_ITEM | 1 | Um artigo de recurso que descreve e contém dados que se encontram armazenados no gestor de recursos. Os dados de vídeo, imagens, documento e outros dados arquivados no gestor de recursos constituem exemplos de artigos de recurso. |
| Modelo de documento | DK_ICM_ITEMT YPE_CLASS_DOC _MODEL | 2 | Um artigo que modela documentos utilizando partes. Um documento é composto por um determinado número de partes, que se encontram contidas no atributo DKConstant.DK_CM_DKPARTS. |
| Parte | DK_ICM_ITEMTY PE_CLASS_DOC_ PART | 3 | Os artigos (partes) na classificação do modelo de documento. |

Nota: As constantes encontram-se no directório com\ibm\mm\sdk\common\DKConstantICM.java de Java e dk\icm\DKConstantICM.h de C++.

Para criar um tipo de artigo (consulte o exemplo de código em baixo):

1. Create an item type and pass it a reference to the content server.
2. Atribua um nome ao tipo de artigo.
3. Adicione atributos ao tipo de artigo e defina os qualificadores pretendidos como, por exemplo, anulável, pesquisável por texto e único.
4. Add the item type to the persistent content server.

Java

```
//This example creates an item type definition and names it.
//The item type name must be less than 15 characters in length.
DKItemTypeDefICM bookItemType = new DKItemTypeDefICM(dsICM);
bookItemType.setName("livro");
bookItemType.setDescription("É um exemplo de um nome de tipo de artigo.");
//Below, a text-searchable attribute called book title is added. The
//attribute is defined to require a unique name and also a value. The value
//does not have to be unique.
DKAttrDefICM attr = (DKAttrDefICM)_dsDefICM.retrieveAttr("book_title");
attr.setTextSearchable(true);attr.setUnique(true);
attr.setNullable(false);
bookItemType.add(attr);
//Here, a book_num_pages attribute is added to the book item
//type with the following characteristics: text searchable, unique,
//and a value is not required.
DKAttrDefICM attr = (DKAttrDefICM)_dsDefICM.retrieveAttr("book_num_pages");
attr.setTextSearchable(false);
attr.setUnique(false);
attr.setNullable(false);
bookItemType.addAttr(attr);
bookItemType.add();
```

C++

```
DKDatastoreICM* pDs;  
DKDatastoreICM* pDs;  
...  
DKDatastoreDefICM* dsDefICM = (DKDatastoreDefICM*)pDs->datastoreDef();  
//create new ItemType  
DKItemTypeDefICM * bookItemType = new DKItemTypeDefICM(pDs);  
bookItemType->setName("book");  
bookItemType->setDescription("This is an example item type name.");  
//Create new Attribute; add it to datastore and to the ItemType  
DKAttrDefICM * attr = (DKAttrDefICM *)dsDefICM->createAttr();  
    attr->setName("book_title");  
    attr->setType(DK_CM_VARCHAR);  
    attr->setSize(80);  
    attr->setTextSearchable(TRUE);  
    attr->setUnique(TRUE);  
    attr->setNullable(FALSE);  
//Persist the attribute to the datastore  
attr->add();  
//Add the newly created attribute to the item type.  
bookItemType->addAttr(attr);  
//Create new Attribute; add it to datastore and to ItemType  
    attr = (DKAttrDefICM *)dsDefICM->createAttr();  
    attr->setName("book_num_pages");  
    attr->setType(DK_CM_INTEGER);  
    attr->setTextSearchable(FALSE);  
    attr->setUnique(FALSE);  
    attr->setNullable(FALSE);  
    attr->add();  
bookItemType->addAttr(attr);  
//Add the entity, bookItemType, to the datastore.  
bookItemType->add();
```

Consulte o exemplo `SitemTypeCreationICM` para obter mais informações.

Enumerar tipos de artigos

Para obter uma lista de tipos de artigo definidos:

1. Estabeleça ligação ao servidor de conteúdos `DKDatastoreICM`.
2. Obtenha uma referência à definição do servidor de conteúdos.
3. Chame o método `listEntityNames` no objecto de definição do servidor de conteúdos para obter uma matriz de cadeias dos nomes dos tipos de artigo.
4. Utilize um ciclo para enumerar todos os nomes.

Java

```
String itemTypeName[] = dsICM.listEntityNames();  
DKSequentialCollection itemTypeColl = (DKSequentialCollection) dsICM.listEntities();  
dkIterator iter = itemTypeColl.createIterator();  
while(iter.more()){  
    dkEntityDef itemType = (dkEntityDef) iter.next();  
    System.out.println(" Item type name : " + itemType.getName()); }  
}
```

C++ Exemplo 1

```
long larraySize = 0;
DKString * itemTypeName = dsICM->listEntityNames(larraySize);
for (int i = 0; i < larraySize; i++) {
    cout <<(char*)itemTypeName[i] <<endl;
}
delete [] itemTypeName;
```

C++ Exemplo 2

```
DKSequentialCollection * itemTypeColl = (DKSequentialCollection *)
dsICM->listEntities();
dkIterator * iter = itemTypeColl->createIterator();
while (iter->more()) {
    dkEntityDef* itemType=(dkEntityDef*)((void*)(*iter->next()));
    cout <<(char*)itemType->getName() < delete(itemType);
}
delete(iter);
delete(itemTypeColl);
```

Para obter mais informações, consulte o exemplo SItemTypeRetrievalICM.

Criar atributos

Para criar atributos:

1. Crie um objecto de definição de atributos.
2. Descreva o objecto que cria definido o seu nome, descrição, tipo, tamanho, etc.
Os exemplos de tipos de atributo que pode criar incluem:
 - DKConstant.DK_CM_BLOB.
 - DKConstant.DK_CM_CHAR.
 - DKConstant.DK_CM_CLOB.
 - DKConstant.DK_CM_DATE.
 - DKConstant.DK_CM_DECIMAL.
 - DKConstant.DK_CM_INTEGER.
 - DKConstant.DK_CM_LONG.
 - DKConstant.DK_CM_SHORT.
 - DKConstant.DK_CM_TIME.
 - DKConstant.DK_CM_TIMESTAMP.
 - DKConstant.DK_CM_VARCHAR.
3. Add the new definition to the persistent content server.

Java

```
//This example defines an attribute for the title of a book.
attr = new DKAttrDefICM(dsICM);
attr.setName("book_title");
attr.setDescription("The title of the book.");
attr.setType(DKConstant.DK_CM_VARCHAR);
attr.setSize(100);
attr.add();
//This example defines an attribute for the number of pages in a book.
attr = new DKAttrDefICM(dsICM);
attr.setName("book_num_pages");
attr.setDescription("The number of pages in the book.");
attr.setType(DKConstant.DK_CM_INTEGER);
attr.setMin((short) 0);
attr.setMax((short) 100000);
attr.add();
```

C++

```
//This example defines an attribute for the title of a book.
DKDatastoreICM * dsICM; .....
DKAttrDefICM * attr = new DKAttrDefICM(dsICM);
    attr->setName("book_title");
    attr->setDescription("The title of the book.");
    attr->setType(DK_CM_VARCHAR);
    attr->setSize((long) 100);
    attr->add();
//This example defines an attribute for the number of pages in a book.
DKAttrDefICM * attr = new DKAttrDefICM(dsICM);
    attr->setName("book_num_pages");
    attr->setDescription("The number of pages in the book.");
    attr->setType(DK_CM_INTEGER);
    attr->setMin((long) 0);
    attr->setMax((long) 100000);
    attr->add();
```

Para obter mais informações relativas à criação de atributos, consulte o exemplo `SAttributeDefinitionCreationICM`.

Criar, actualizar e eliminar grupos de atributos

Os grupos de atributos tornam mais fácil a tarefa de adicionar todo um grupo de atributos a artigos e sub-componentes. Um atributo pode pertencer a qualquer número de grupo de atributos, desde zero até qualquer número. Um atributo pode ser adicionado a grupos de atributos múltiplos. Um artigo de dados (DDO) pode conter grupos de atributos múltiplos. O mesmo nome de atributo pode aparecer no DDO, mas cada atributo é completamente independente, com base no espaço do nome. When an attribute is added to an attribute group, it impacts only the component types that are created after the addition. Os tipos de componentes pré-existentes permanecem inalterados. O exemplo seguinte demonstra como criar um grupo de atributos:

Java

```
// Create a datastore definition object given the connected datastore
DKDatastoreDefICM dsDefICM = (DKDatastoreDefICM)dsICM.datastoreDef();
//Creating a new attribute group
DKAttrGroupDefICM attrGroup = new DKAttrGroupDefICM(dsICM);
// Set a name, maximum 15 characters
attrGroup.setName("Book_Details");
//Set a description for the new attribute group
attrGroup.setDescription("Detailed book information");
// Retrieve the definition of an attribute that will be
//added to this attribute group.
DKAttrDefICM title = (DKAttrDefICM) dsDefICM.retrieveAttr("book_title");
// Retrieve the definition of another attribute that will be added
//to this attribute group.
DKAttrDefICM publisher = (DKAttrDefICM) dsDefICM.retrieveAttr("publisher");
// Add the Attribute Definitions to the Attribute Group
attrGroup.addAttr(title);
attrGroup.addAttr(publisher);
// add it to the persistent datastore
attrGroup.add();
```

C++

```
// Create a datastore definition object given the connected datastore
DKDatastoreDefICM* dsDefICM = (DKDatastoreDefICM*) dsICM->datastoreDef();
//Creating a new attribute group
DKAttrGroupDefICM* attrGroup = new DKAttrGroupDefICM(dsICM);
// Set a name, maximum 15 characters
attrGroup->setName("Book_Details");
//Set a description for the new attribute group
attrGroup->setDescription("Detailed book information");
// Retrieve the definition of an attribute that will be
//added to this attribute group.
DKAttrDefICM* title = (DKAttrDefICM *) dsDefICM->retrieveAttr("book_title");
// Retrieve the definition of another attribute that will be added
//to this attribute group.
DKAttrDefICM* publisher = (DKAttrDefICM *) dsDefICM->retrieveAttr("publisher");
// Add the Attribute Definitions to the Attribute Group
attrGroup->addAttr(title);
attrGroup->addAttr(publisher);
// add it to the persistent datastore
attrGroup->add();

delete(attrGroup);
```

Para actualizar o nome e descrição e um grupo de atributos, chame o método `update()` em `DKAttrGroupDefICM` e faculte uma tabela de IDs de atributos. Se este grupo de atributos já tiver sido associado a um tipo de componente, não poderá actualizar este grupo de atributos.

Para eliminar um grupo de atributos, o utilizador também trabalha com a classe `DKAttrGroupDefICM`. Ao eliminar o grupo de atributos, os atributos primários que anteriormente formavam o grupo de atributos permanecem no servidor de bibliotecas. As seguintes excepções aplicam-se quando elimina um grupo de atributos:

- Um grupo de atributos não pode ser eliminado, caso esteja associado a um tipo de componente e seja constante.

- Um atributo não pode ser removido de um grupo de atributos caso este grupo esteja associado a um tipo de componente e seja constante.
- O utilizador não pode adicionar um atributo a um grupo de atributos caso este grupo esteja associado a um tipo de componente e seja constante.

Para obter mais informações, consulte o exemplo `SAttributeGroupDefCreationICM`.

Enumerar os atributos num servidor de conteúdos

O exemplo seguinte demonstra como obter uma lista de atributos num servidor de conteúdos.

Java

```
DKDatastoreICM dsICM;
DKDatastoreDefICM dsDefICM=DKDatastoreDefICM(dsICM.datastoreDef());
//Get a collection containing all Attribute Definitions.
DKSequentialCollection attrDefColl =
    (DKSequentialCollection)dsDefICM.listAttrs();
if ((attrDefColl!=null) &&(attrDefColl.cardinality(>0))){
    //Create an iterator to iterate through the collection
    dkIterator iter = attrDefColl.createIterator();
    while(iter.more()){
        //while there are still items in the list,continue
        dkAttrDef attrDef = (dkAttrDef) iter.next();
        System.out.println("-"+attr.getName()+":"+attr.getDescription());
    }
}
```

C++

```
DKDatastoreICM * dsICM; .....
DKDatastoreDefICM*dsDefICM =(DKDatastoreDefICM *)dsICM->datastoreDef();
//Get a collection containing all Attribute Definitions.
DKSequentialCollection*attrDefColl =
    (DKSequentialCollection *)dsDefICM->listAttrs();
if (attrDefColl &&(attrDefColl->cardinality(>0)){
    //Create an iterator to iterate through the collection
    dkIterator*iter =attrDefColl->createIterator();
    while(iter->more()){
        //while there are still items in the list,continue
        dkAttrDef* attrDef =(dkAttrDef *)iter->next()->value();
        cout <<"- "<<attrDef->getName()<<":"<<attrDef->getDescription()<<endl;
        delete(attrDef);
    }
    delete(iter);
    delete(attrDefColl);
}
delete(dsDefICM);
```

Enumerar nomes de atributos para um tipo de artigo

O exemplo seguinte demonstra como obter uma lista de nomes de atributos para um tipo de artigo. Para obter mais informações, consulte o Exemplo de Instrução de API `SItemTypeRetrievalICM`.

Java

```
//Get a collection containing all attr. defs for the Item Type.
DKSequentialCollection attrColl = (DKSequentialCollection)
    dsICM.listEntityAttrs(itemTypeName);
//Accessing attribute each and printing the name and description.
System.out.println("\nAttributes of Item Type '"+itemTypeName+":'
    (" +attrColl.cardinality()+')');
//Create an iterator to iterate through the collection
dkIterator iter = attrColl.createIterator();
while(iter.more()) {
    //while there are still items in the list, continue
    DKAttrDefICM attr = (DKAttrDefICM)iter.next();
    System.out.println("-"+attr.getName()+":"+attr.getDescription());
}
```

C++

```
//Get a collection containingall Attribute Definitions for the Item Type.
DKSequentialCollection*attrColl =(DKSequentialCollection*)
    dsICM->listEntityAttrs(itemTypeName);
//Accessing attribute each and printing the name and description.
cout <<"\nAttributes of Item Type '"<<itemTypeName <<":'
    ("<<attrColl->cardinality()<<')'<<
//Create an iterator to iterate through the collection
dkIterator* iter =attrColl->createIterator();
while(iter->more()) {
    //while there are still items in the list, continue
    DKAttrDefICM* attr =(DKAttrDefICM*)iter->next()->value();
    cout <<"- "<<attr->getName()<<": "<<attr->getDescription()<<endl;
    delete(attr);
}
delete(iter);
delete(attrColl);
```

Criar um artigo

Um artigo corresponde à árvore completa de DDOs componentes com, pelo menos, um componente raiz. A esse DDO componente raiz deve ser atribuído um Tipo de Propriedade de Artigo ou Tipo Semântico. Quando cria um artigo, o utilizador deve atribuir-lhe uma propriedade de tipo de artigo. As propriedades válidas de tipo de artigo são documento, pasta ou artigo. Tem de especificar a propriedade de tipo de artigo como o segundo parâmetro da função `createDDO` do servidor de conteúdos. O valor do tipo de propriedade é armazenado na propriedade do DDO denominada `DK_CM_PROPERTY_ITEM_TYPE`. Não confunda esta propriedade de tipo de artigo com a definição geral de tipo de artigo que descreve a estrutura do artigo. A Tabela 11 apresenta uma lista dos tipos de propriedades de artigo disponíveis.

Tabela 11. Definições de propriedade de tipo de artigo

| Tipo de propriedade | Constante | Definição |
|---------------------|----------------|--|
| Documento | DK_CM_DOCUMENT | O artigo representa um documento ou dados armazenados. As informações contidas nesta artigo podem formar um documento. Este artigo pode ser considerado um documento comum, pois isto não significa obrigatoriamente uma implementação de um modelo de documento específico. |

Tabela 11. Definições de propriedade de tipo de artigo (continuação)

| Tipo de propriedade | Constante | Definição |
|---------------------|--------------|---|
| Pasta | DK_CM_FOLDER | O artigo representa um objecto que contém ou faz referência a conteúdos ou objectos. Este artigo pode ser considerado uma pasta comum, pois isto não significa obrigatoriamente uma implementação de um modelo de documento específico. |
| Item (predefinição) | DK_CM_ITEM | Item genérico. Este artigo não se adequa a tipos semânticos definidos pelo sistema ou definidos pelo utilizador. |

Os artigos são criados como DKDDOs. Utilize sempre os métodos `DKDatastoreICM's createDDO()` para criar DKDDOs, pois o sistema utiliza os métodos `DKDatastoreICM's createDDO` para configurar automaticamente informações importantes na estrutura de DKDDO.

Quando cria um artigo, o utilizador define os componentes que formam o artigo. Por exemplo, se optar por criar um artigo hierárquico, terá de criar componentes descendentes.

1. Ao utilizar os métodos `createDDO` e `createChildDDO` do servidor de conteúdos, crie um DDO de artigo e defina todos os seus atributos e outras informações necessárias. Este exemplo utiliza o objecto de `DKDatastoreICM` com sessão iniciada e ligado denominado `dsICM`.
2. Crie um componente de raiz.

Exemplo 1 de Java

```
DKDDO myDocumentDDO = dsICM.createDDO("EmployeeDoc",
DKConstantICM.DK_CM_DOCUMENT);
```

Exemplo 2 de Java

```
DKDDO myFolderDDO = dsICM.createDDO("DeptFolder",
DKConstantICM.DK_CM_FOLDER);
```

Exemplo 1 de C++

```
DKDDO* myDocumentDDO = dsICM->createDDO("EmployeeDoc",DK_CM_DOCUMENT);
```

Exemplo 2 de C++

```
DKDDO* myFolderDDO = dsICM->createDDO("DeptFolder",DK_CM_FOLDER);
```

3. Crie um componente descendente sob o componente raiz "EmployeeDoc", como, por exemplo, "Dependent".

Java

```
DKDDO myDDO =dsICM.createChildDDO("EmployeeDoc","Dependent");
```

C++

```
DKDDO* myDDO = dsICM.createChildDDO("EmployeeDoc","Dependent");
```

4. Add the child component to the parent.

Java

```
short dataid = myDocumentDDO.dataId(DK_CM_NAMESPACE_CHILD,"Dependent");
DKChildCollection children =
    (DKChildCollection) myDocumentDDO.getData(dataid);
children.addElement(myDDO);
```

C++

```
short dataid = myDocumentDDO->dataId(DK_CM_NAMESPACE_CHILD,"Dependent");
DKChildCollection* children =
    (DKChildCollection*)(dkCollection*)
myDocumentDDO->getData(dataid);
children->addElement(myDDO);
```

5. Utilize o método setData para preencher o DDO com os valores adequados.

Java

```
myDDO.setData(.....);
```

C++

```
myDDO->setData(.....);
```

6. Guarde este artigo na memória permanente.

Java

```
myDocumentDDO.add();
```

C++

```
myDocumentDDO->add();
```

No exemplo anterior, o último passo criou um documento no servidor de conteúdos. Quando um DDO de documento é adicionado a um servidor de conteúdos, todos os seus atributos são adicionados, incluindo todas as partes dentro da recolha de DKParts. Quando um DDO de documento é adicionado a um

servidor de conteúdos, todos os seus atributos são adicionados, incluindo todos os descendentes e partes dentro da recolha de DKParts.

O *Tipo semântico* define a utilização ou regras de um artigo. O Content Manager já possui alguns tipos semânticos pré-definidos, mas o utilizador pode também definir os seus próprios tipos semânticos.

Pode especificar o tipo semântico como o segundo parâmetro da função createDDO do servidor de conteúdos. O valor do tipo semântico é armazenado na propriedade DKConstantICM.DK_ICM_PROPERTY_SEMANTIC_TYPE do DDO, que pode conter o mesmo valor do tipo de propriedade do artigo. O conector de ICM suporta tipos semânticos de pasta e documento. O utilizador pode também definir os seus próprios tipos semânticos. A Tabela 12 enumera os tipos semânticos disponíveis.

Tabela 12. Tipos semânticos pré-definidos

| Tipo semântico | Constante | Definição |
|--------------------------|---------------------------------|---|
| Documento | DK_ICM_SEMANTIC_TYPE_DOCUMENT | Indica que este artigo é um documento |
| Pasta | DK_ICM_SEMANTIC_TYPE_FOLDER | Indica que este artigo é uma pasta |
| Anotação | DK_ICM_SEMANTIC_TYPE_ANNOTATION | Indica que este artigo ou parte do mesmo é uma anotação da parte base |
| Contentor | DK_ICM_SEMANTIC_TYPE_CONTAINER | Indica que este artigo é um contentor, que pode conter outros artigos. Esta relação contentor-contido é representada utilizando ligações. |
| Histórico | DK_ICM_SEMANTIC_TYPE_HISTORY | Indica que este artigo ou parte do mesmo é um histórico da parte base. |
| Nota | DK_ICM_SEMANTIC_TYPE_NOTE | Indica que este artigo ou parte do mesmo é uma nota da parte base. |
| Base | DK_ICM_SEMANTIC_TYPE_BASE | Indica que este artigo ou parte do mesmo é a parte base que pode ter uma anotação, nota ou histórico associados ao artigo. |
| Definido pelo utilizador | Definido pelo utilizador | Um tipo semântico definido pelo utilizador interpretado pela aplicação. |

Notas:

- Em Java, as constantes estão definidas em DKConstantICM.java.
- Em C++, as constantes estão definidas em DKConstantICM.h.
- No Content Manager Versão 7, o tipo semântico é denominado tipo afiliado.

Se criar um artigo num tipo de artigo que foi definido como tipo de artigo de recurso, é devolvido o XDO correcto. Para obter mais informações relativas a recursos, consulte o exemplo SResourceItemCreationICM. Poderá também revelar-se útil rever as informações relativas à criação de artigos no exemplo SItemCreationICM.

Definir e obter valores de atributo de artigo

Java

Os valores de atributo são armazenados e obtidos como `java.lang.Objects`. Para definir ou obter valores de atributo, utilize os métodos `setData` e `getData()` respectivamente, do DDO. Defina o valor utilizando um objecto com o tipo que corresponde ao tipo de atributo. **Exemplo:**

```
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,nameOfAttrStr), valueObj);  
Object obj = ddo.getData(ddo.dataId(DK_CM_NAMESPACE_ATTR,nameOfAttrStr));
```

A instrução anterior define o valor de atributo como o valor transmitido como um `java.lang.Object valueObj`. O `nameOfAttrStr` é o nome de cadeia do atributo. Este atributo foi definido e especificado no tipo de artigo quando foi definido o tipo de artigo deste DDO. `valueObj` é o valor que deve definir e que deve ser o tipo certo para este atributo.

C++

Ao definir valores para atributos individuais, utilize o nome de definição do atributo individual. Para aceder a atributos individuais que pertencem a um grupo de atributos, utilize este formato: <Nome Grupo Atributos>.<Nome Atributo>. **Exemplo:**

```
//O snippet de código seguinte apresenta o valor do atributo do carácter  
//O "book_title" relativo ao artigo representado pelo ddoDocument de DDO  
//é definido como o valor "Effective C++"  
ddoDocument->setData(ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,  
                                         DKString("book_title")),DKString("Effective C++"));  
//O snippet de código seguinte apresenta o valor do atributo do número inteiro  
//"book_num_pages" para o artigo representado  
//pelo ddoDocument de DDO é definido com o valor "250"  
ddoDocument->setData(ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,  
                                         DKString("book_num_pages")), (long)250);  
//Este snippet de código mostra como o valor do atributo "book_title" do  
//artigo representado pelo ddoDocument de DDO é obtido para a variável de cadeia  
//"title".  
DKString title =(DKString) ddoDocument->getData(ddoDocument->  
dataId(DK_CM_NAMESPACE_ATTR,DKString("book_title")));
```

Alterar atributos de um artigo

Para modificar os atributos de um artigo, utilize o método `setData` de DDO e defina-o como o valor necessário especificando esse valor, utilizando `java.lang.Object`, tal como o exemplo seguinte demonstra. No exemplo seguinte, `nameOfAttrStr` é o nome de cadeia do atributo e `valueObj` corresponde ao valor.

Java

```
ddo.setData(ddo.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,nameOfAttrStr), valueObj);
```

C++

```
ddoDocument->setData(ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,"book_title"),  
DKString("More Effective C++"));
```

Actualizar um artigo

Para actualizar um artigo:

1. Obtenha um artigo ou crie um artigo e adicione-o ao servidor de conteúdos.
2. Modifique o artigo, os seus atributos, recolhas de ligações e assim por diante.
3. Chame a operação de actualização do DDO.

Java

```
ddo.update();
```

C++

```
ddo->update();
```

Se um artigo for activado para elaboração de versões, o utilizador pode criar uma nova versão do artigo em vez de actualizar o artigo actual, tal como o exemplo seguinte demonstra.

Java

```
ddo.update(DKConstant.DK_CM_VERSION_NEW);
```

C++

```
ddo->update(DK_CM_VERSION_NEW);
```

Para actualizar a versão mais recente de um artigo, utilize o formato apresentado no exemplo seguinte:

Java

```
ddo.update(DKConstant.DK_CM_VERSION_LATEST);
```

C++

```
ddo->update(DK_CM_VERSION_LATEST);
```

Também pode actualizar um DDO chamando o método `updateObject` em `DKDatastoreICM` com as opções adequadas, tal como o exemplo seguinte demonstra:

Java

```
// Connected DKDatastoreICM object named ds;  
// DKDDO object named ddo;  
int options = DK_CM_VERSION_NEW + DK_CM_CHECKIN;  
ds.updateObject(ddo, options);
```

C++

```
int options = DK_CM_VERSION_NEW + DK_CM_CHECKIN;  
ds->updateObject(ddo, options);
```

Importante: O utilizador deve dar saída do artigo antes de chamar o método actualizar e voltar a dar entrada do artigo quando finalizar a actualização. Consulte “Dar entrada e dar saída de artigos” na página 164. Para obter mais informações relativas à actualização de artigos, consulte o Exemplo de Instrução de API SItemUpdateICM.

Definir um tipo de artigo de recurso

Um artigo de recurso é um artigo com atributos adicionais definidos pelo sistema, que define a localização, tipo, tamanho, etc., do objecto representado pelo artigo. O objecto, por vezes, é denominado por “recurso” e pode ser um ficheiro de vídeo, uma imagem, um documento de processador de texto, etc. Para obter informações adicionais, consulte o exemplo SItemTypeCreationICM (no directório de exemplos).

Os passos seguintes explicam-lhe o processo de definição de um tipo de artigo de recurso.

1. Obtenha o objecto de definição do servidor de conteúdos a partir do servidor de conteúdos ligado.

Java

```
DKDatastoreDefICM dsDefICM = (DKDatastoreDefICM)dsICM.datastoreDef();
```

C++

```
DKDatastoreDefICM* dsDefICM = (DKDatastoreDefICM*) dsICM->datastoreDef();
```

2. Crie uma nova definição de tipo de artigo.

Java

```
DKItemTypeDefICM itemType = new DKItemTypeDefICM(dsICM);  
itemType.setName("SampleResource");  
itemType.setDescription("Simple Resource Lob Item Type");
```

C++

```
DKItemTypeDefICM * itemType = new DKItemTypeDefICM(dsICM);  
itemType->setName("SampleResource");  
itemType->setDescription("Simple Resource Lob Item Type");
```

3. Adicione um atributo.

Java

```
DKAttrDefICM attr = (DKAttrDefICM)dsDefICM.retrieveAttr("S_varchar");
itemType.addAttr(attr);
//A classificação de recursos indica que esta classe irá
//conter o ficheiro de dados
itemType.setClassification(DKConstantICM.DK_ICM_ITEMTYPE_CLASS_RESOURCE_ITEM);
```

C++

```
DKAttrDefICM * attr = (DKAttrDefICM*) dsDefICM->retrieveAttr("S_varchar");
itemType->addAttr(attr);
//A classificação de recursos indica que esta classe irá conter
//o ficheiro de dados
itemType->setClassification(DK_ICM_ITEMTYPE_CLASS_RESOURCE_ITEM);
```

4. Especifique a classe de XDO e o tipo de recurso deste tipo de artigo.

Java

```
itemType.setXDOClassName(DKConstantICM.DK_ICM_XDO_LOB_CLASS_NAME);
itemType.setXDOClassID(DKConstantICM.DK_ICM_XDO_LOB_CLASS_ID);
itemType.add();
```

C++

```
itemType->setXDOClassName(DK_ICM_XDO_LOB_CLASS_NAME);
itemType->setXDOClassID(DK_ICM_XDO_LOB_CLASS_ID);
itemType->add();
```

Criar um artigo de recurso

Criar artigos de recurso é muito semelhante a criar artigos comuns. Os XDOs expandem os DDOs e, dependendo do tipo de artigo de recurso, o XDO pode ser ainda mais expandido. A Tabela 13 contém os tipos de artigo do recurso e a hierarquia de classe utilizada para os criar. Para obter informações, consulte o exemplo `SResourceItemCreationICM` (no directório de exemplos).

Tabela 13. Hierarquia de classe do tipo de artigo do recurso

| Tipo | DDO | XDO | Extensão |
|-----------|-----|----------------------|-------------|
| LOB | | DKDDO -> DKLobICM | |
| Texto | | DKDDO -> DKLobICM -> | DKTextICM |
| Imagem | | DKDDO -> DKLobICM -> | DKImageICM |
| Sequência | | DKDDO -> DKLobICM -> | DKStreamICM |

Os passos seguintes explicam-lhe o processo de criação de um artigo de recurso. Há várias formas de definir e armazenar conteúdos de recurso. O processo seguinte constitui um exemplo de armazenamento directo a partir do ficheiro, na altura em que o artigo é adicionado ao servidor de conteúdos.

1. Crie o artigo do recurso. Tenha em atenção que pode ser qualquer tiposemântico e que a chamada de `DKDatastoreICM::createDDO` também é utilizada paracriar um artigo de recurso da mesma forma que é utilizada para

criar um artigo comum. O tipo de recurso devolvido por `DKDatastoreICM.createDDO` é difundido para a sub-classe correcta (neste caso é `DKLobICM`), com base na classificação de XDO definida durante a criação do tipo de artigo no qual se baseia este artigo de recurso.

Java

```
DKLobICM    lob = (DKLobICM)dsICM.createDDO  
            ("SampleResource",DKConstant.DK_CM_DOCUMENT);
```

C++

```
DKLobICM*    lob = (DKLobICM*)  
dsICM->createDDO("SampleResource",DK_CM_DOCUMENT);
```

2. Defina o conteúdo ou dados no objecto. Após a criação do lob, pode definir o tipo de MIME do recurso. Neste caso, o recurso é um documento de MS Word. O tipo MIME descreve o tipo de conteúdo que está a ser armazenado.

Java

```
lob.setContentFromClientFile(fileName);  
lob.setMimeType("application/msword");
```

C++

```
lob->setContentFromClientFile(fileName);  
lob->setMimeType("application/msword");
```

Para obter informações adicionais relativas a tipos de MIME, consulte o exemplo `SResourceItemMimeTypeICM`.

3. Adicione os dados ao servidor de conteúdos. Neste caso, trata-se de um documento de Word exemplo. Tenha em atenção que o conteúdo do artigo de recurso é armazenado no gestor de recursos.

Java

```
lob.add("SResourceItemICM_Document1.doc");
```

C++

```
lob->add("SResourceItemICM_Document1.doc");
```

Importante: Em C++, deve limpar a memória.
`delete(lob);`

Para obter mais informações, consulte o exemplo `SItemUpdateICM`.

Pesquisar artigos

Para localizar artigos que correspondam a um conjunto de critérios, conclua os seguintes passos:

1. Estabeleça ligação a um objecto DKDatastoreICM.
2. Processe a consulta correcta. Certifique-se de que a sua consulta está de acordo com o idioma da consulta. Para obter mais informações, consulte “Compreender a linguagem da consulta” na página 185.
3. Guarde os resultados da consulta num objecto de DKResult.

Java

```
DKNVPair options[] = new DKNVPair[3];
options[0] =
    new DKNVPair(DKConstant.DK_CM_PARM_MAX_RESULTS, "0"); // No Max (Default)
options[1] = new DKNVPair(DKConstant.DK_CM_PARM_RETRIEVE,
    new Integer(DKConstant.DK_CM_CONTENT_ATTRONLY));
options[2] = new DKNVPair(DKConstant.DK_CM_PARM_END, null);
DKResults results = (DKResults)dsICM.evaluate(query,
    DKConstantICM.DK_CM_XQPE_QL_TYPE, options);
```

C++

```
DKNVPair *options    = new DKNVPair[3];
// only one result will be returned
options[0].set(DK_CM_PARM_MAX_RESULTS, "1");// Retrieve only one item
options[1].set(DK_CM_PARM_RETRIEVE , (long)DK_CM_CONTENT_ATTRONLY);
// Last option has to be this value
options[2].set(DK_CM_PARM_END , NULL);
//Note if a query is expected to return more than one result,
//the DKDatastoreICM::execute method
//should be used. The execute method returns a dkResultSetCursor.
DKResults* results = (DKResults*)(dkCollection*)dsICM->evaluate
    (query, DK_CM_XQPE_QL_TYPE, options);
```

Para obter informações adicionais, consulte o exemplo SSearchICM.

Obter um artigo

Para obter explicitamente um artigo ou actualizar um artigo através de uma consulta, o utilizador deve aceder ao objecto de PID ou cadeia de PID do artigo a ser obtido ou actualizado. Se apenas dispuser do ID de Artigo, pode executar uma consulta para obter as informações de PID completas. Independentemente do nome de tipo de artigo e do ID de artigo, o cenário seguinte mostra como obter o DDO, caso pretenda obtê-lo explicitamente.

Java

1. Após estabelecer ligação a um objecto de `datastoreICM`, pesquise o artigo utilizando o ID de artigo adequado. Para obter informações relativas a escrever consultas, consulte “Consultar o servidor de Content Manager” na página 186.
2. Guardar os resultados da consulta num objecto `DKResults`. No exemplo de código seguinte, `queryStr` corresponde à cadeia de pesquisa no formato de linguagem de consulta correcta.

```
DKResults results = (DKResults)dsICM.evaluate(queryStr, DKConstantICM.DK_CM_XQPE_QL_TYPE, null);
```
3. Crie um iterador em `DKResults` e utilize-o para obter DDOs, que agora pode obter um a um.

```
ddo.retrieve();
```
4. Suponha que possui a cadeia de PID obtida de um DDO, tal como é demonstrado neste exemplo:

```
DKPidICM pid = ddo.getPidObject();  
String pidStr = pid.pidString();
```

Ao utilizar a cadeia de PID, pode executar uma obtenção concluindo os seguintes passos.

1. Crie um DDO utilizando o método `createDDO` de `DKDatastoreICM`. Não utilize o conector `DKDDO`. No seguinte exemplo, o objecto `dsICM` já se encontra ligado a um armazenamento de dados do Content Manager. Para além disso, o PID é uma cadeia denominada `pidStr`.

```
DKDDO ddo = dsICM.createDDO(pidStr);
```
2. Chame a operação de obtenção do DDO.

```
ddo.retrieve();
```

C++

1. Após estabelecer ligação a um objecto de `datastoreICM`, pesquise o artigo utilizando o ID de artigo adequado. Para obter informações relativas a escrever consultas, consulte Consultar o servidor de Content Manager.
2. Guarde os resultados da consulta num objecto de `DKResult`. No exemplo de código abaixo apresentado, `queryStr` é a cadeia de consulta no formato correcto do idioma de consulta.

```
DKResults* results = (DKResults*)(dkCollection*)  
dsICM->evaluate(queryStr, DK_CM_XQPE_QL_TYPE, NULL);
```
3. Criar um DDO utilizando o método `createDDO` do `DKDatastoreICM`. Não utilize o conector `DKDDO`. No exemplo a seguir apresentado, o objecto `dsICM` já está ligado a um armazenamento de dados do Content Manager. Além disso, o PID é uma cadeia com o nome `pidStr`.

```
DKDDO* ddo = dsICM->createDDO(pidStr);
```
4. Chame a operação de obtenção do DDO.

```
ddo->retrieve();
```

Se um artigo for activado para elaboração de versões, o utilizador pode obter uma versão específica do artigo utilizando o método de obtenção da classe `DKDDO` com as opções adequadas. Para obter a versão mais recente de um artigo, utilize a

constante `DK_CM_VERSION_LATEST` como opção de obtenção. Por predefinição, (se não forem especificadas opções), o `ddo.retrieve()` obtém a versão mais recente de um artigo.

Exemplo:

```
DKDDO ddo = ds.createDDO(...);
.... ddo.retrieve(DK_CM_VERSION_LATEST);
```

Também pode obter um DDO chamando o método `retrieveObject` no objecto `DKDatastoreICM`.

Exemplo:

```
DKDatastoreICM ds = new DKDatastoreICM();
//ligar, etc ...
DKDDO ddo = ds.createDDO(itemType,option);
//define o PID tal como é apresentado em cima...
ds.retrieveObject(ddo);
```

Para obter mais informações relativas à obtenção de artigos, consulte os Exemplos de Instrução de API `SItemRetrievalICM` e `SResourceItemRetrievalICM`.

Se um artigo for activado para elaboração de versões e pretender obter a versão mais recente, incluindo os seus atributos e descendentes, utilize o seguinte formato:

Java

```
int options =DK_CM_CONTENT_ATTRONLY + DK_CM_CONTENT_CHILDREN +
                                                    DK_CM_VERSION_LATEST;
ds.retrieveObject(ddo, options);
```

C++

```
DKDDO* ddo;
long options =DK_CM_CONTENT_ATTRONLY +DK_CM_CONTENT_CHILDREN +DK_CM_VERSION_LATEST;
dsICM->retrieveObject(ddo,options);
```

Para obter mais informações relativas à obtenção como, por exemplo, opções de obtenção, consulte o exemplo `SItemRetrievalICM`.

Eliminar artigos

Utilize o método de eliminação no DDO para eliminar um artigo do servidor de conteúdos.

Java

```
ddoDocument.del();
```

C++

```
ddoDocument->del();
```

O DDO deve ter definidos o seu ID de artigo e tipo de objecto, bem como possuir uma ligação válida ao servidor de conteúdos.

Para obter mais informações relativas à eliminação de artigos, consulte o Exemplo de Instrução de API `SItemDeletionICM`.

Dar entrada e dar saída de artigos

Para impedir que dois utilizadores efectuem alterações ao mesmo artigo ao mesmo tempo, deverá dar saída de um artigo antes de o actualizar. Dar saída de um artigo concede os direitos exclusivos de actualização ao utilizador que tenha dado saída do artigo. Quando dá saída de um artigo, o utilizador mantém um bloqueio de escrita permanente sobre esse artigo. O artigo é bloqueado e desbloqueado como um todo, incluindo todas as versões e componentes descendentes. No entanto, os artigos ligados e referenciados não são bloqueados juntamente com o artigo. O bloqueio permanente sobre o artigo é libertado quando o utilizador dá entrada do artigo.

Para dar entrada e saída de artigos, utilize as operações `checkIn` e `checkOut` do `DKDatastoreICM`, tal como o exemplo seguinte demonstra.

1. Tendo estabelecido a ligação a um objecto de `DKDatastoreICM` intitulado `dsICM` e utilizado um DDO intitulado `myDataObject`, dê saída do artigo.

Java

```
dsICM.checkOut(myDataObject);
```

C++

```
dsICM->checkOut(myDataObject);
```

2. Dê entrada do artigo.

Java

```
dsICM.checkIn(myDataObject);
```

C++

```
dsICM->checkIn(myDataObject);
```

Para obter mais informações relativas a dar entrada e saída de artigos, consulte o Exemplo de Instrução de API `SItemUpdateICM`.

Definir e obter as propriedades de atribuição de versão de um artigo

O exemplo a seguir apresentado demonstra como definir as propriedades de atribuição de versão de um artigo e como obter as propriedades de atribuição de versão de um artigo.

Java

```
DKPidICM pid = (DKPidICM)ddo.getPidObject();
// Accessing version information
String version = pid.getVersionNumber();
...
// Setting the version number in the DDO
pid.setVersionNumber(version);
```

C++

```
DKPidICM * pid = (DKPidICM *)ddo->getPidObject();
// Accessing version information
DKString version = pid->getVersionNumber();
....
//Setting the version number for the PID associated with an item (DDO).
pid->setVersionNumber((char *)version);
```

Trabalhar com propriedades de elaboração de versões

Esta secção faculta exemplos para auxiliar o utilizador a trabalhar com propriedades de elaboração de versões.

Obter a política de controlo de versões de um tipo de artigo

Um artigo tem uma política de controlo de versão, que contém a propriedade de atribuição de versão para o artigo. Apresenta-se a seguir a lista das três propriedades de atribuição de versão disponíveis e o valor utilizado para representar cada propriedade na política de controlo de versões.

DK_ICM_VERSION_CONTROL_ALWAYS

Elaboração constante de versões.

DK_ICM_VERSION_CONTROL_NEVER

A elaboração de versões não é suportada neste tipo de artigo (predefinição).

DK_ICM_VERSION_CONTROL_BY_APPLICATION

A aplicação determina o esquema de elaboração de versões.

Java

```
short versionControlPolicy;
DKItemTypeDefICM item = newDKItemTypeDefICM();
....
versionControlPolicy = item.getVersionControl();
```

C++

```
short versionControlPolicy = 0;
DKItemTypeDefICM * item = NULL;
...
versionControlPolicy = item->getVersionControl();
```

Definir o controlo de versão para um tipo de artigo

Java

```
DKItemTypeDefICM item = new DKItemTypeDefICM();
short versionControl = DK_ICM_VERSION_CONTROL_ALWAYS;
....
item.setVersionControl(versionControl);
```

C+

```
DKItemTypeDefICM * item = NULL;
short versionControl = DK_ICM_VERSION_CONTROL_ALWAYS;
...
item->setVersionControl(versionControl);
```

Para obter um exemplo completo de acesso a informações de controlo de versões de definições de tipo de artigo no sistema, consulte o Exemplo de Instrução de API `SItemTypeRetrievalICM`.

Obter o número máximo permitido de versões para um tipo de artigo

Java

```
short versionMax;
DKItemTypeDefICM item = new DKItemTypeDefICM();
....
versionMax = item.getVersionMax();
```

C++

```
short versionMax = 0;
DKItemTypeDefICM * item = NULL;
....
versionMax = item->getVersionMax();
```

Definir o número máximo permitido de versões permitido para um tipo de artigo Neste exemplo, apenas dez versões de um artigo com base neste tipo de artigo são mantidas pelo sistema.

Java

```
short versionMax=10;
DKItemTypeDefICM item = new DKItemTypeDefICM();
...
item.setVersionMax(versionMax);
```

C++

```
short versionMax = 10;
DKItemTypeDefICM * item = NULL;
....
item->setVersionMax(versionMax);
```

Definir o valor do tipo de elaboração de versões para um tipo de artigo

C++

```
DKItemTypeDefICM * item = NULL;
short versionType = DK_ICM_ITEM_VERSIONING_OPTIMIZED;
...
item->setVersioningType(versioningType);
```

Trabalhar com pastas

Uma pasta é um DDO totalmente suportado que representa um artigo. Uma pasta possui a estrutura hierárquica completa de dados do tipo de artigo no qual é criada. Uma pasta é um DDO de pasta de tipo semântico e possui um atributo, `DKConstant.DK_CM_DKFOLDER`, que contém um `DKFolder`, independentemente da classificação do tipo de artigo. O objecto `DKFolder`, um `DKSequentialCollection`, pode conter um número indeterminado de DDOs de componente raiz que representem outros artigos.

Os procedimentos seguintes incluem fragmentos de códigos apenas para fins informativos. Não copie estes fragmentos de códigos. Cole-os directamente no programa da aplicação, pois estes não funcionarão no formato actual. Para obter um exemplo completo de pastas que possa executar, consulte o exemplo `SFolderICM` no directório de exemplos.

Criar uma pasta

O método `DKDatastoreICM.createDDO()` é utilizado para criar DDOs durante o tempo de execução. Tal como é demonstrado no exemplo `SItemCreationICM` (no directório de exemplos), ao criar artigos de pasta, o tipo de propriedade ou tipo semântico do artigo necessita de ser especificado como `DKConstant.DK_CM_FOLDER`.

É criado um artigo de pasta utilizando o método `DKDatastoreICM.createDDO` com o tipo semântico `DK_CM_FOLDER`. O utilizador pode criar uma pasta especificando `DKConstant.DK_CM_FOLDER` como o segundo parâmetro do método `createDDO`.

Java

```
DKDDO ddoFolder = dsICM.createDDO("S_simple", DKConstant.DK_CM_FOLDER);
```

C++

```
DKDDO* ddoFolder = dsICM->createDDO("S_simple", DK_CM_FOLDER);
```

Utilize o método `setData` para preencher o `ddoFolder`. Após a criação de um DDO de pasta, este é guardado no servidor de conteúdos permanente.

Java

```
ddoFolder.add();
```

C++

```
ddoFolder->add();
```

Adicionar conteúdo a uma pasta

Todos os conteúdos que serão colocados em DKFolder devem ser permanentes no servidor de conteúdos antes de chamar o método add() ou update() no DDO de pasta. Para tornar os conteúdos permanentes, chame os seus métodos DKDDO.add().

Para adicionar artigos a uma pasta, utilize a função addElement() de DKFolder. Após adicionar membros suficientes à pasta, pode chamar o método de adição ou actualização para guardar as relações pasta-conteúdo no armazenamento permanente. Esta acção agrupa todas as modificações de pasta e numa única chamada do servidor de bibliotecas. Ao adicionar artigos a uma pasta, não adicione várias cópias do mesmo artigo a DKFolder. Visto que todas as modificações realizadas às pastas são identificadas, não realize modificações duplicadas ou que entrem em conflito. Por exemplo, não adicione várias cópias do mesmo artigo à pasta.

Siga os passos seguintes para adicionar conteúdos a uma pasta. Tenha em atenção que uma pasta pode conter outra pasta (sub-pasta) como um dos seus artigos de conteúdo.

Java

1. Crie três novos artigos. Estes artigos serão adicionados a uma pasta como sendo os seus conteúdos. Os conteúdos de pasta podem ser de qualquer tiposemântico.

```
DKDDO ddoDocument=dsICM.createDDO("S_simple",DKConstant.DK_CM_DOCUMENT);  
DKDDO ddoFolder2=dsICM.createDDO("S_simple",DKConstant.DK_CM_FOLDER);  
DKDDO ddoItem=dsICM.createDDO("S_simple",DKConstant.DK_CM_ITEM);
```

2. Utilize o método setData para preencher os DDOs. Os artigos que serão adicionados como conteúdos de pasta devem ser tornados permanentes no armazenamento de dados.

```
ddoDocument.add();  
ddoItem.add();  
ddoFolder2.add();
```

3. Obtenha o atributo dkFolder do DDO de pasta anteriormente criado e tornado permanente.

```
short dataid = ddoFolder.dataId  
    (DKConstant.DK_CM_NAMESPACE_ATTR,DKConstant.DK_CM_DKFOLDER);  
dkFolder = (DKFolder) ddoFolder.getData(dataid);
```

4. Deve ser dada saída da pasta do armazenamento de dados antes de ser actualizada (por intermédio da adição de conteúdos).

```
dsICM.checkOut(ddoFolder);
```

5. Adicione os artigos anteriormente criados à pasta.

```
dkFolder.addElement(ddoDocument);  
dkFolder.addElement(ddoItem);  
dkFolder.addElement(ddoFolder2);
```

6. Consolide as alterações efectuadas à pasta e dê entrada explícita das alterações.

```
ddoFolder.update();  
dsICM.checkIn(ddoFolder);
```

C++

1. Criar os artigos que irão ser adicionados à pasta.

```
DKDDO * ddoDocument = dsICM->createDDO("book", DK_CM_DOCUMENT);  
DKDDO * ddoFolder2 = dsICM->createDDO("book", DK_CM_FOLDER);  
DKDDO * ddoItem = dsICM->createDDO("book", DK_CM_ITEM);
```
2. Persistir os artigos criados ao armazenamento de dados.

```
ddoDocument->add();  
ddoItem->add();  
ddoFolder2->add();
```
3. Obter o atributo DKFolder para a pasta.

```
DKFolder * dkFolder;  
short dataid = ddoFolder->dataId(DK_CM_NAMESPACE_ATTR, DK_CM_DKFOLDER);  
if (dataid!=0) dkFolder = (DKFolder*)(dkCollection*) ddoFolder->  
getData(dataid);
```
4. Retirar a pasta (Uma pasta deve ser retirada antes de ser actualizada)

```
dsICM->checkOut(ddoFolder);
```
5. Adicionar os artigos criados à pasta.

```
dkFolder->addElement(ddoDocument);  
dkFolder->addElement(ddoItem);  
dkFolder->addElement(ddoFolder2); // Folders can contain sub-folders.
```
6. Actualizar a pasta. Isto também insere implicitamente a pasta (desbloqueei-a).

```
ddoFolder->update();
```
7. Inserir explicitamente a pasta.

```
dsICM->checkIn(ddoFolder);
```

Remover conteúdos de uma pasta

Os artigos podem ser removidos de uma pasta de duas formas. Uma remoção diferida (a remoção, na prática, é efectuada quando o DDO de pasta é actualizado e várias chamadas de servidor de conteúdos são combinadas numa só) é efectuada utilizando o(s) método(s) `removeElementXX`. Uma remoção imediata (e dispendiosa, já que várias chamadas são efectuadas ao servidor de conteúdos) pode ser efectuada utilizando o método `dkFolder.removeMember`. Consulte o exemplo `SFolderICM` (no directório de exemplos) para obter mais detalhes relativos a trabalhar com pastas.

Conclua os seguintes passos para remover conteúdos de uma pasta:

Java

1. Dê saída do DDO de pasta do qual pretende remover um artigo.
`dsICM.checkOut(ddoFolder);`
2. Procure o artigo a remover. Neste caso, procure o DDO `ddoItem` anteriormente criado. Crie um iterador para efectuar a iteração do conteúdo da pasta.

```
dkIterator iter = dkFolder.createIterator();
String itemPidString = ddoItem.getPidObject().pidString();
while(iter.more()) {
    // Avance com o indicador para o elemento seguinte e devolva esse objecto.
    // Compare as cadeias de PID do DDO devolvido do iterador
    // com o DDo que deverá ser removido.
    DKDDO ddo = (DKDDO) iter.next();
    if(ddo.getPidObject().pidString().equals(itemPidString)) {
        // O artigo a ser removido é detectado.
        // Remova-o (o elemento actual no iterador)
        dkFolder.removeElementAt(iter);
    }
}
```
3. Torne a alteração permanente e dê entrada no DDO de pasta.
`ddoFolder.update();`
`dsICM.checkIn(ddoFolder);`

C++

1. Crie um artigo e adicione-o a uma pasta. Note que a pasta foi criada antes.

```
DKDDO * ddoItem = dsICM->createDDO("book", DK_CM_ITEM);
ddoItem->add();
DKFolder * dkFolder;
short dataid = ddoFolder->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKFOLDER);
if (dataid!=0) dkFolder = (DKFolder*)(dkCollection*) ddoFolder->
getData(dataid);
dsICM->checkOut(ddoFolder);
dkFolder->addElement(ddoItem);
ddoFolder->update();
```
2. Inserir explicitamente a pasta.
`dsICM->checkIn(ddoFolder);`
3. Criar um iterador para a pasta.
`dkIterator* iter = dkFolder->createIterator();`
4. Utilize a iteração através do conteúdo da pasta até que seja localizado o artigo que irá ser removido. Remova o artigo.

```
while (iter->more())
{
    DKDDO* ddo = (DKDDO*) iter->next()->value();
    if ( ((DKPidICM*) ddo->getPidObject())->pidString() ==
        ((DKPidICM*) ddoItem->getPidObject())->pidString() )
    {
        //Found the element to be removed. Remove it
        dkFolder->removeElementAt(*iter);
        // Now that we found the ddoItem, remove it.
    }
}

delete(iter);
```

Obter os conteúdos de uma pasta

Para obter os conteúdos de uma pasta, o utilizador deve definir a opção de obtenção. Por predefinição, a opção de obtenção não se encontra definida. Se não definir a opção de obtenção, o objecto só irá conter um atributo DKFolder ou DK_CM_DKFOLDER quando a obtenção for chamada e quando forem definidas as opções correctas. As opções de obtenção incluem:

- DKConstant.DK_CM_CONTENT_LINKS_OUTBOUND
- DKConstant.DK_CM_CONTENT_ITEMTREE

Para obter um artigo de pasta numa recolha de atributoDKFolder com Ligações Externas especificado, utilize o seguinte formato:

Java

```
ddoFolder.retrieve(DKConstant.DK_CM_CONTENT_LINKS_OUTBOUND);
```

C++

```
ddoFolder->retrieve(DK_CM_CONTENT_LINKS_OUTBOUND);
```

Para obter um artigo de pasta onde a árvore de artigos inclui ligações e conteúdos de pasta, utilize o seguinte formato:

Java

```
ddoFolder.retrieve(DKConstant.DK_CM_CONTENT_ITEMTREE);
```

C++

```
ddoFolder->retrieve(DK_CM_CONTENT_ITEMTREE);
```

Obter todas pastas que contêm um DDO específico

Várias pastas podem conter o mesmo artigo (DDO). Isto permite-lhe associar o artigo a diferentes aplicações ou utilizadores. Para determinar quais as pastas que actualmente contêm um DDO específico, consulte o exemplo seguinte.

Os passos seguintes demonstram como localizar pastas que contêm um objecto de ddoDocument previamente criado. Consulte o exemplo SFolderICM (no directório de exemplos) para obter detalhes adicionais relativos a trabalhar com pastas.

Java

1. Obtenha o objecto de extensão do armazenamento de dados.

```
DKDatastoreExtICM dsExtICM =(DKDatastoreExtICM)
dsICM.getExtension(DKConstant.DK_CM_DATASTORE_EXT);
```
2. Chame o método `getFoldersContainingDDO` no objecto de extensão para localizar as pastas que contêm o objecto `ddoDocument`. Esta acção devolve uma recolha de DDOs em que cada DDO devolvido consiste numa pasta que contém o objecto de `ddoDocument`.

```
DKSequentialCollection list = dsExtICM.getFoldersContainingDDO(ddoDocument);
```
3. Crie um iterador para efectuar ciclos através da recolha de pastas devolvida.

```
iter =list.createIterator();
while(iter.more()) {
    // Avance com o iterador para o elemento seguinte e devolva
    // esse objecto.
    DKDDO ddo =(DKDDO) iter.next();
    // Imprima o ID do artigo do DDO de pasta de modo a identificar
    // o objecto de pasta devolvido.
    System.out.println("-Item ID: " +
        ((DKPidICM)ddo.getPidObject()).getItemId());
}
```

C++

```
DKDDO* ddoDocument = ....;
//Create datastore object,connect to datastore,create DDO etc.
....
//Create a new datastore extension object from the connected object
DKDatastoreExtICM* dsExtICM = (DKDatastoreExtICM*)
    dsICM->getExtension(DK_CM_DATASTORE_EXT);
//Retrieve the PID for the created DDO
DKPidICM* pid = (DKPidICM*) ddoDocument->getPidObject();
//Retrieve a list of folders containing the DDO created earlier.
DKSequentialCollection *list = dsExtICM->getFoldersContainingDDO(ddoDocument);
//Create a iterator for the returned DDO collection
dkIterator* iter =list->createIterator();
while(iter->more()) {
    DKDDO* ddo =(DKDDO*) iter->next()->value();
    pid = (DKPidICM*) ddo->getPidObject();
    cout << "-Item ID:" << pid->getItemId() << endl;
    delete ddo;
}
delete iter;
```

Definir ligações entre artigos

Pode utilizar ligações para associar um artigo origem a um destino com um artigo de descrição facultativo. O utilizador define ligações em objectos de `DKLink`, onde pode especificar os seguintes tipos de elementos de ligação.

Tabela 14. Estrutura de dados de ligação

| Ligação DK | Definição |
|-------------------------|----------------------------------|
| Nome do Tipo de Ligação | O tipo de ligação. |
| Origem | O artigo origem de uma ligação. |
| Destino | O artigo destino de uma ligação. |

Tabela 14. Estrutura de dados de ligação (continuação)

| Ligação DK | Definição |
|-----------------|-------------------------------------|
| Item de Ligação | A descrição do artigo. Facultativo. |

Visto que uma ligação representa uma relação de um para muitos, esta é representada num DDO por um artigo de dados sob o nome de espaço LINK, como valor DKLinkCollection. Para obter mais informações relativas a trabalhar com ligações, consulte o exemplo SLinksICM no directório de exemplos.

Uma ligação não pertence à origem nem ao destino. Uma ligação apenas liga uma origem e um destino. Por exemplo, se a origem A for ligada ao destino B, A será sempre a origem e B será sempre o destino, independentemente do DDO a que A ou B estão a ser referenciados.

Na memória, tanto os DDOs origem como os DDOs destino contêm cópias do mesmo objecto DKLink. Devido ao facto do objecto DKLink conter uma referência à origem e ao destino, um DDO que contenha uma ligação contém também uma ligação que faz referência a si mesmo, bem como a outro DDO. Por exemplo, se a Origem A estiver ligada ao Destino B, tanto A como B contêm a mesma ligação, como mostra o exemplo da Tabela 15.

Tabela 15. Exemplo de definição de DKLink

| DKLink Definido de A para B | DDO A | DDO B |
|-----------------------------|---------------|---------------|
| A origem é A | A origem é A | A origem é A |
| O destino é B | O destino é B | O destino é B |

Ao adicionar ou remover ligações da memória permanente, o utilizador apenas tem de executar a operação num dos dois artigos, no origem ou no destino. A ligação é automaticamente actualizada para o outro artigo.

Para obter mais informações relativas a ligações e um exemplo de ligações completo que possa executar, consulte o exemplo SLinksICM no directório de exemplos.

Ligações internas e externas

Ao utilizar ligações para referenciar um DDO específico, seja de origem ou destino, o utilizador pode considerar as ligações como sendo *internas* ou *externas*. Ao considerar uma ligação segundo a perspectiva de um determinado DDO, caso esse DDO seja o destino, então este último considera-a uma ligação interna. Entretanto, segundo a perspectiva do DDO no outro extremo dessa ligação, a mesma ligação é externa.

No exemplo em Tabela 15, a ligação do DDO A ao DDO B é externa, enquanto a mesma ligação ao DDO B é externa.

Nomes de tipos de ligação

As relações entre ligações possuem nomes. Dentro de um DDO, estão agrupadas ligações em conjuntos de ligações. Existem nomes de tipos de ligação definidos pelo sistema e o utilizador pode definir os seus próprios nomes. Pode utilizar todos os nomes de tipos de ligação definidos pelo utilizador ou os nomes definidos pelo sistema. Os seguintes nomes de tipo de artigo definidos pelo sistema incluem:

O nome de tipo de ligação contém

Constante Java: DKConstant.DK_ICM_LINKTYPENAME_CONTAINS

Constante C++: DK_ICM_LINKTYPENAME_CONTAINS

Nome do tipo de ligação: DKFolder

Constante Java: DKConstant.DK_ICM_LINKTYPENAME_DKFOLDER

Constante C++: DK_ICM_LINKTYPENAME_DKFOLDER

O nome de tipo de ligação DKFolder é facultado e utilizado pelo sistema para gerir pastas. O objecto DKFolder constitui uma interface simplificada a um conjunto de ligações externas para facilitar o trabalho com pastas. Consequentemente, não deverá utilizar o nome de tipo de ligação DKFolder para definir ou eliminar ligações. O utilizador também jamais deverá utilizar o nome de tipo de ligação DKFolder com um DKLinkCollection. No entanto, deve especificar o nome de tipo de ligação DKFolder de forma a pesquisar pastas utilizando ligações.

Consulte o exemplo SLinksICM no directório cmbroot\samples, para obter informações adicionais relativas a ligações. Para obter informações relativas à criação de tipos de ligação definidos pelo utilizador, consulte o exemplo SLinkTypeDefinitionCreationICM.

Obter artigos ligados

Quando o utilizador obtém ligações, possui a opção de obter apenas ligações internas, apenas externas, ou ambos os tipos. De forma a obter tipos internos e externos, deve definir uma opção de obtenção. Pode definir as seguintes opções para obter ligações:

Java

- DKConstant.DK_CM_CONTENT_LINKS_OUTBOUND
- DKConstant.DK_CM_CONTENT_LINKS_INBOUND
- DKConstant.DK_CM_CONTENT_LINKS_OUTBOUND + DKConstant.DK_CM_CONTENT_LINKS_INBOUND
- DKConstant.DK_CM_CONTENT_ITEMTREE

C++

- DK_CM_CONTENT_LINKS_OUTBOUND
- DK_CM_CONTENT_LINKS_INBOUND
- DK_CM_CONTENT_LINKS_OUTBOUND + DK_CM_CONTENT_LINKS_INBOUND
- DK_CM_CONTENT_ITEMTREE

Tenha em mente que, antes de efectuar uma chamada aos métodos add() ou update() de DDO, todos os artigos associados às ligações já devem ser permanentes.

Trabalhar com o controlo de acesso

Se aceder ao cliente de administração de sistemas do Content Manager ou às APIs do Content Manager para escrever o programa de administração do utilizador, pode utilizar as funções de controlo de acesso para controlar o acesso às informações do sistema de Content Manager. As várias APIs de controlo de acesso permitem ao utilizador controlar o acesso a todo o sistema, a um conjunto de operações intimamente relacionado do sistema ou a um artigo individual.

Algumas das tarefas que pode concluir utilizando as APIs de controlo de cesso incluem:

- Criar privilégios.
- Associar uma lista de acções a informações no sistema.
- Dar permissão a utilizadores para executar acções nas informações de servidor de bibliotecas.

Criar um privilégio

Um privilégio é representado pela classe `DKPrivilegeICM`. Os passos e exemplos de código seguintes mostram como criar um novo objecto de privilégio, torná-lo permanente e obter um privilégio. Para criar um privilégio, siga os seguintes passos:

Java

1. Estabeleça uma ligação a um armazenamento de dados

```
DKDatastoreICM ds = new DKDatastoreICM();
ds.connect("ICMNLSDb","icmadmin",password,"");
dkDatastoreDef dsDef = (dkDatastoreDef)ds.datastoreDef();
DKDatastoreAdminICM dsAdmin = (DKDatastoreAdminICM)dsDef.datastoreAdmin();
```
2. Obtenha um objecto `DKAuthorizationMgmtICM`. Esta classe processa tarefas de gestão de autorizações.

```
DKAuthorizationMgmtICM aclMgmt = (DKAuthorizationMgmtICM)
dsAdmin.authorizationMgmt();
```
3. Crie um novo objecto de privilégio.

```
DKPrivilegeICM priv = new DKPrivilegeICM(ds);
```
4. Atribua um nome ao objecto de privilégio.

```
priv.setName("UserPriv");
```
5. Atribua uma descrição ao objecto de privilégio.

```
priv.setDescription("Este é um privilégio definido pelo utilizador");
```
6. Adicione o novo privilégio objecto do gestor de autorizações.

```
aclMgmt.add(priv);
```
7. Obtenha o privilégio recém-criado do objecto do gestor de autorizações.

```
dkPrivilege aPriv = aclMgmt.retrievePrivilege("UserPriv");
```
8. Visualize informações relativas ao objecto de privilégio.

```
System.out.println("privilege name = " + aPriv.getName());
System.out.println("privilege description = " + aPriv.getDescription());
```


C++

1. Criar o objecto do armazenamento de dados, bem como os objectos de gestão de administração relacionados.

```
//Create the datastore object
DKDatastoreICM * ds = new DKDatastoreICM();
//Connect to the underlying datastore
ds->connect("ICMNLSDb", "icmdmin", "password", "");
//Retrieve the datastore definition object (used to access
//and manipulate CM meta-data)
dkDatastoreDef * dsDef = (dkDatastoreDef *) ds->datastoreDef();
//Create the class used to represent and process datastore
//administration functions.
DKDatastoreAdminICM * dsAdmin = (DKDatastoreAdminICM *)
    dsDef->datastoreAdmin();
//Retrieve the class used to represent and manage the authorization
//related functionality of the ICM datastore
DKAuthorizationMgmtICM * aclMgmt = (DKAuthorizationMgmtICM *)
    dsAdmin->authorizationMgmt();
```

2. Criar um novo objecto de privilégio e definir as suas propriedades

```
DKPrivilegeICM * priv = new DKPrivilegeICM(ds);
//Set the name of the privilege object
priv->setName("UserPriv");
//Set the privilege description
priv->setDescription("This is user-defined privilege");
```

3. Adicione o privilégio ao armazenamento de dados através do objecto de gestão de autorização.

```
aclMgmt->add(priv);
```

Criar um conjunto de privilégios

Um conjunto de privilégios é representado pela classe `DKPrivilegeSetICM`. Antes de poder iniciar a criação de conjuntos de privilégios, deve estar ligado a um servidor de conteúdos. Para criar um conjunto de privilégios, conclua os seguintes passos:

Java

1. Crie novos privilégios (ou obtenha privilégios) para adicionar ao novo conjunto de privilégios.

```
DKPrivilegeICM priv_1 = new DKPrivilegeICM(ds);
priv_1.setName("ItemCheckOut");
DKPrivilegeICM priv_2 = new DKPrivilegeICM(ds);
priv_2.setName("ItemQuery");
DKPrivilegeICM priv_3 = new DKPrivilegeICM(ds);
priv_3.setName("ItemAdd");
```
2. Crie um novo conjunto de privilégios.

```
DKPrivilegeSetICM privSet1 = new DKPrivilegeSetICM(ds);
```
3. Atribua um nome ao conjunto de privilégios.

```
privSet1.setName("UserPrivSet");
```
4. Atribua uma descrição ao conjunto de privilégios.

```
privSet1.setDescription("Este é um conjunto de privilégios definido pelo
utilizador");
```
5. Adicione os privilégios ao conjunto de privilégios.

```
privSet1.addPrivilege(priv_1);
privSet1.addPrivilege(priv_2);
privSet1.addPrivilege(priv_3);
```
6. Adicione o conjunto de privilégio recém criado ao gestor de autorizações.

```
AcIMgmt.add(privSet1);
```
7. Visualize informações relativas ao conjunto de privilégios recém criado.

```
DKPrivilegeSetICM aPrivSet = (DKPrivilegeSetICM)
acIMgmt.retrievePrivilegeSet("UserPrivSet");
System.out.println("privilege set name = " + aPrivSet.getName());
System.out.println("privilege set description =
" + aPrivSet.getDescription());
dkCollection coll = aPrivSet.listPrivileges();
dkIterator iter = coll.createIterator();
while(iter.more()){
DKPrivilegeICM _priv = (DKPrivilegeICM) iter.next();
System.out.println("  privilege name = " + _priv.getName());
}
```

C++

1. Criar o objecto do armazenamento de dados, bem como os objectos relacionados com a gestão da administração.

```
//Create the datastore object
DKDatastoreICM * ds = new DKDatastoreICM();
//Connect to the underlying datastore
ds->connect("ICMNLSDb", "icmdmin", "password", "");
//Retrieve the datastore definition object
//(used to access and manipulate CM meta-data)
dkDatastoreDef * dsDef = (dkDatastoreDef *) ds->datastoreDef();
//Create the class used to represent and process datastore administration
// functions.
DKDatastoreAdminICM * dsAdmin = (DKDatastoreAdminICM *)
    dsDef->datastoreAdmin();
//Retrieve the class used to represent and manage the authorization
//related functionality of the ICM datastore
DKAuthorizationMgmtICM * aclMgmt = (DKAuthorizationMgmtICM *)
    dsAdmin->authorizationMgmt();
```

2. Criar três privilégios e definir as suas propriedades.

```
DKPrivilegeICM * priv_1 = new DKPrivilegeICM(ds);
priv_1->setName("ItemCheckOut");
DKPrivilegeICM * priv_2 = new DKPrivilegeICM(ds);
priv_2->setName("ItemQuery");
DKPrivilegeICM * priv_3 = new DKPrivilegeICM(ds);
priv_3->setName("ItemAdd");
```

3. Criar um novo conjunto de privilégios e definir as suas propriedades.

```
DKPrivilegeSetICM * privSet1 = new DKPrivilegeSetICM(ds);
privSet1->setName("UserPrivSet");
privSet1->setDescription("This is a user-defined priv set");
```

4. Adicionar os privilégios criados ao novo conjunto de privilégios.

```
privSet1->addPrivilege(priv_1);
privSet1->addPrivilege(priv_2);
privSet1->addPrivilege(priv_3);
```

5. Adicionar o conjunto de privilégios ao armazenamento de dados utilizando o objecto de gestão de autorização.

```
aclMgmt->add(privSet1);
```

Visualizar propriedades do conjunto de privilégios

O exemplo seguinte demonstra como apresentar as propriedades de um conjunto de privilégios.

C++

```
//Retrieve a privilege set using its name
DKPrivilegeSetICM * sPrivSet = (DKPrivilegeSetICM *)
    aclMgmt->retrievePrivilegeSet("UserPrivSet");
//Display privilege set properties
cout<<"privilege set name = "<< (char *)sPrivSet->getName() << endl;
cout<<"priv set descrip="<< (char *)sPrivSet->getDescription() << endl;
//Retrieve the list of privileges that are a part of this privilege set
dkCollection * coll = sPrivSet->listPrivileges();
dkIterator * iter = coll->createIterator();
while (iter->more())
{
    DKPrivilegeICM* _priv = (DKPrivilegeICM *) (void *) (*iter->next());
    cout<<"  privilege name = "<< (char *)_priv->getName() << endl;
}
delete(iter);
```

Definir uma lista de controlo de acesso (ACL)

O modelo de controlo de acesso do Content Manager é aplicado ao nível da entidade controlada. Uma entidade controlada é uma unidade de dados de utilizador protegidos. Uma entidade controlada pode ser um artigo individual, um tipo de artigo ou uma biblioteca completa. As operações sobre entidades controladas são reguladas por uma ou mais regras de controlo. A ACL é o contentor destas regras de controlo. A classe DKAccessControlListICM representa uma ACL. Cada entidade controlada num sistema Content Manager deve estar associada a uma ACL.

O exemplo seguinte demonstra como definir uma ACL.

Java

```
//Define a new DKACLData object.
//A DKACLData class is used to hold ACL related data.
DKACLData aclData1 = new DKACLData();
//set the user group name for the ACL
aclData1.setUserGroupName("ICMADMIN");
//set ACL patron type.
aclData1.setPatronType(DK_CM_USER_KIND_USER);
//Set the privilege set associated with this ACL
aclData1.setPrivilegeSet(privSet_1);
//Create another DKACLData object.
DKACLData aclData2 = new DKACLData();
aclData2.setUserGroupName("ICMPUBLIC");
aclData2.setPatronType(DK_CM_USER_KIND_GROUP);
aclData2.setPrivilegeSet(privSet_2);
//Create a new ACL. A DKAccessControlListICM represents a CM v8.2. ACL
DKAccessControlListICM acl1 = new DKAccessControlListICM(ds);
//Assign a new to the newly-created ACL
acl1.setName("UserACL");
//Assign a description to the newly-created ACL
acl1.setDescription("This is a user-defined ACL");
//Add the previously created ACL data objects to the ACL
acl1.addACLData(aclData1);
acl1.addACLData(aclData2);
//Add the newly-created ACL to the authorization manager
aclMgmt.add(acl1);
//Retrieve and display information about the newly created ACL
DKAccessControlListICM acl_1 = (DKAccessControlListICM)
aclMgmt.retrieveAccessControlList("UserACL");
System.out.println("ACL name = " + acl_1.getName());
System.out.println("    desc = " + acl_1.getDescription());
dkCollection coll = acl_1.listACLData();
dkIterator iter = coll.createIterator();
while (iter.more()) {
    DKACLData aclData = (DKACLData) iter.next();
    DKPrivilegeSetICM _privSet = (DKPrivilegeSetICM) aclData.getPrivilegeSet();
    System.out.println("  PrivSet name = " + _privSet.getName());
    System.out.println("  PrivSet desc = " + _privSet.getDescription());
    String usrGrpName = aclData.getUserGroupName();
    System.out.println("    UserGroupName = " + usrGrpName);
    short patronType = aclData.getPatronType();
    System.out.println("    Patron type = " + patronType);
}
```

C++

1. Definir novos objectos de DKACLData. Cada objecto de DKACLData é utilizado para conter dados relacionados com a ACL.

```
DKACLData * aclData1 = new DKACLData();
// set the name of the user group to be associated with this ACL
aclData1->setUserGroupName("ICMADMIN");
// Set the ACL patron type. Can be one of 3 possible
//values:DK_CM_USER_KIND_USER or DK_CM_USER_KIND_GROUP or
//DK_CM_USER_KIND_PUBLIC.
aclData1->setPatronType(DK_CM_USER_KIND_USER);
// Set the privilege set associated with the ACL.
DKPrivilegeSetICM * privSet_1 = (DKPrivilegeSetICM *)
    aclMgmt->retrievePrivilegeSet("UserPrivSet");
aclData1->setPrivilegeSet(privSet_1);
```

2. Criar outro objecto de DKACLdata.

```
DKACLData * aclData2 = new DKACLData();
aclData2->setUserGroupName("ICMPUBLIC");
aclData2->setPatronType(DK_CM_USER_KIND_GROUP);
DKPrivilegeSetICM * privSet_2 = (DKPrivilegeSetICM *)
    aclMgmt->retrievePrivilegeSet("PublicPrivSet");
aclData2->setPrivilegeSet(privSet_2);
```

3. Crie uma nova ACL. Defina os valores de propriedade para este nova ACL.

```
DKAccessControlListICM * acl1 = new DKAccessControlListICM(ds);
//Assign a new name to the newly-created ACL.
acl1->setName("UserACL");
// Assign a description to the newly-created ACL.
acl1->setDescription("This is a user-defined ACL");
```

4. Adicione os objectos de dados da ACL, criada no passo três, à ACL.

```
acl1->addACLData(aclData1);
acl1->addACLData(aclData2);
```

5. Adicione a ACL, criada no passo três, ao gestor de autorização.

```
aclMgmt->add(acl1);
```

O programa de exemplo completo encontra-se no directório de exemplos.

Obter e visualizar informações sobre a ACL

Para obter e visualizar as informações sobre o ACL, conclua os passos seguintes.

1. Obter a ACL a partir do objecto de gestão de autorização utilizando o nome da ACL.

C++

```
DKAccessControlListICM * acl_1 = (DKAccessControlListICM *) aclMgmt->
    retrieveAccessControlList("UserACL");
cout<<"ACL name = "<< (char *)acl_1->getName() << endl;
cout<<"    desc = "<< (char *)acl_1->getDescription() << endl;
```

2. Obter os dados da ACL associados à ACL.

C++

```
dkCollection * coll = acl_1->listACLData();
dkIterator * iter = coll->createIterator();
char szpatronType[12] = {0x00};
while (iter->more())
{
    DKACLData * aclData = (DKACLData *) (void *) (*iter->next());
    DKPrivilegeSetICM * _privSet = (DKPrivilegeSetICM *) aclData->
        getPrivilegeSet();
    cout<<"privSet name = "<< (char *) _privSet->getName() << endl;
    cout<<"privSet desc = "<< (char *) _privSet->getDescription() << endl;
    DKString usrGrpName = aclData->getUserGroupName();
    cout<<"    UserGroupName = "<< (char *) usrGrpName << endl;
    short patronType = aclData->getPatronType();
    sprintf(szpatronType, "%d", patronType);
    cout<<"    Patron type    = "<< szpatronType << endl;
}
delete(iter);
```

Atribuir uma ACL a um tipo de artigo

Após uma ACL ser criada, pode associá-la a um tipo de artigo específico. De seguida são apresentados os passos que terá de seguir para atribuir uma ACL a um tipo de artigo:

1. Configure um novo tipo de artigo.

Java

```
DKItemTypeDefICM it = new DKItemTypeDefICM(dsICM);
//Atribua um nome a este tipo de artigo
it.setName("TextResource1");
//Atribua uma descrição a este tipo de artigo
it.setDescription("CMv8.2 Text Resource Item Type.");
//Atribua um código de ACL a este tipo de artigo
it.setItemTypeACLCode((int)DK_ICM_ITEMACL_BIND_AT_ITEM);
....
it.add(); // torne o tipo de artigo permanente
...
```

C++

```
DKItemTypeDefICM * itemType = new DKItemTypeDefICM(dsICM);
// Atribua um nome a este tipo de artigo.
itemType->setName("TextResource1");
// Atribua uma descrição a este tipo de artigo.
itemType->setDescription("CMv8.2 Text Resource Item Type.");
```

2. Recupere os dados da ACL associados à ACL.

Java

```
int itemTypeACLCode = it.getItemTypeACLCode();
// Ao definir o identificador de ACL do tipo de artigo como TRUE(1),
// confirmamos que a associação de ACL se encontra ao nível do tipo de
// artigo. Ao definir o identificador de ACL do tipo de artigo como
// FALSE(0), estará implícito que a associação de ACL não se encontra
// ao nível do tipo de artigo
it.setItemLevelACLFlag(1);// - verdadeiro
// Determine se a associação da ACL se encontra ou não ao nível do
// tipo de artigo
int itemTypeACLFlag = it.getItemLevelACLFlag();
```

C++

```
itemType->setItemTypeACLCode((long) 1);
// Ao definir o identificador de ACL do tipo de artigo como TRUE (1),
//confirmamos que a associação de ACL se encontra ao nível do tipo de artigo.
//Ao definir o identificador de ACL do tipo de artigo como FALSE(0),
// estará implícito que a associação de ACL não se encontra ao nível do tipo de artigo.
itemType->setItemLevelACLFlag((short)1);
itemType->add();
// torne o tipo de artigo permanente.
```

O programa de exemplo completo está disponível no directório de exemplos.

Atribuir uma ACL a um artigo

Para activar o controlo de acesso a nível de artigo, o utilizador deve associar um ACL ao artigo. Para isso, deve criar o artigo utilizando o método add() no DDO, tal como é demonstrado no seguinte fragmento de código. No fragmento de código seguinte, parte-se do princípio que já possui uma ligação ao servidor de conteúdos e que criou a ACL. O programa completo encontra-se no directório de exemplos.

Java

```
//Criar um novo DDO
DKDDO ddoItem =dsICM.createDDO("myItemType",DKConstantICM.DK_CM_ITEM);
//criar uma nova ACL (lista de controlo de acesso)
DKAccessControlListICM acl1 = new DKAccessControlListICM(ds);
//Atribuir um nome à ACL
acl1.setName("MyACL");
//Adicione uma nova propriedade ao DDO. Esta propriedade irá
//chamar-se DK_ICM_PROPERTY_ACL
int propId = ddoItem.addProperty(DK_ICM_PROPERTY_ACL);
//Defina a ACL anteriormente criada (ou obtida) como valor desta propriedade
ddoItem.setProperty(propId,"MyACL");
//Consolide o DDO no armazenamento de dados
ddoItem.add();
```


C++

1. Criar um novo artigo (DDO) baseado num tipo de artigo existente.
`DKDDO * ddoItem = dsICM->createDDO("book", DK_CM_ITEM);`
2. Criar uma nova ACL (Lista de Controlo de Acesso) e definir as propriedades da ACL.
`DKAccessControlListICM * acl1 = new DKAccessControlListICM(dsICM);`
3. Atribuir um nome à ACL.
`acl1->setName("MyACL");`
4. Adicionar uma nova propriedade ao DDO. Esta propriedade irá ter o nome DK_ICM_PROPERTY_ACL.
`ushort propId = ddoItem->addProperty(DK_ICM_PROPERTY_ACL);`
5. Definir a ACL, criada anteriormente, como o valor para esta propriedade. Note que um utilizador pode optar por obter uma ACL existente e utilizá-la como a ACL deste artigo.
`ddoItem->setProperty(propId, "MyACL");`
6. Tornar persistente o DDO no armazenamento de dados.
`ddoItem->add();`

Compreender a linguagem da consulta

Uma poderosa linguagem de consulta com base em XML está disponível no Content Manager 8.2. Quando uma aplicação pesquisa artigos armazenados no sistema do Content Manager, o motor subjacente executa o processamento da pesquisa com base numa consulta. Para examinar de forma eficiente o modelo de dados hierárquico do Content Manager, irá utilizar a linguagem da consulta do Content Manager. A linguagem de consulta facultar os seguintes benefícios:

- Suporta o modelo de dados completo.
- Suporta versões como, por exemplo, pesquisar uma versão específica e a versão mais recente.
- Permite pesquisas em hierarquias de vista de tipo de componente em artigos ligados e referências.
- Combina a pesquisa paramétrica e de texto.
- Faculta capacidades de SORTBY.
- Aplica controlo de acesso do Content Manager.
- Está em conformidade com XQuery Path Expressions (XQPE), um sub-conjunto da minuta de trabalho de Consulta de XML W3C.
- Executa pesquisas de alto rendimento.

A linguagem de consulta do Content Manager é uma linguagem com base em XML que, ao contrário das linguagens do proprietário, adapta-se ao XQuery Path Expressions (XQPE), um subconjunto do esboço de trabalho de W3C XML Query. A linguagem de consulta pesquisa tipos de artigos hierárquicos e localiza artigos rápida e facilmente. Antes de iniciar a escrita de consultas, o utilizador deve compreender os conceitos, sintaxe e gramática da linguagem de consulta.

Todas as consultas são realizadas em vistas de tipos de componente. Assim sendo, os nomes que o utilizador irá utilizar para os componentes raiz ou para os componentes descendentes nas cadeias de consulta podem ser os nomes de vistas de tipo de componente base que são criados pelo sistema (por exemplo, Journal,

Journal_Article) ou os nomes de vistas de tipo de componentes definidos pelo utilizador (por exemplo, My_Journal, My_Book_Section). Em administração do sistema, as vistas de tipo de componente são denominadas como subconjuntos de tipo de componente.

Quando o utilizador submete uma consulta de um documento, pasta, objecto e assim por diante, o seu pedido é direccionado para o motor de consulta do Content Manager, que processa a consulta e a traduz para a SQL apropriada. Então a biblioteca adiciona as verificações de segurança (ACLs) e executa a consulta.

Consultar o servidor de Content Manager

Para consultar o servidor de bibliotecas do Content Manager:

1. Consulte o seu servidor de conteúdos criando uma cadeia de consulta para representar as suas condições de pesquisa.
2. Chame o método avaliar, executar ou executeWithCallback juntamente com a cadeia e opções da consulta.
3. Receba os resultados através de um objecto DKResults, um dkResultSetCursor, ou um objecto dkCallback.

As APIs de consulta executam as tarefas de processamento de consulta como, por exemplo, preparar e executar uma consulta, supervisionar o estado da execução de uma consulta e obter os resultados.

Evidentemente, uma cadeia de consulta pode conter um de três tipos de consultas: paramétrica, de texto e combinada. Uma consulta paramétrica utiliza condições como a igualdade e comparação. Uma consulta de texto utiliza as funções de pesquisa de texto para tornar a pesquisa mais eficaz. Uma consulta combinada é composta por consultas de texto e paramétricas.

Para executar uma consulta, pode utilizar um dos seguintes métodos: avaliação, execução ou executeWithCallback. O método de avaliação devolve todos os resultados como uma recolha de DDOs e é útil para pequenos conjuntos de resultados. O método de execução devolve um objecto dkResultSetCursor, que possui as seguintes características:

- O cursor dkResultSetCursor funciona como um cursor de servidor de conteúdos.
- Poderá utilizá-la para grandes conjuntos de resultados porque os DDOs que devolve são obtidos em blocos, tal como o utilizador os solicitar.
- Poderá também utilizar dkResultSetCursor para voltar a executar uma consulta, chamando os métodos de encerramento e abertura.
- Pode utilizar dkResultSetCursor para eliminar e actualizar a posição actual do cursor de conjunto de resultados.

O método executeWithCallback executa uma consulta de modo assíncrono e envia em blocos os resultados para o objecto de repetição de marcação especificado. Assim, o método executeWithCallback liberta o módulo principal de execução da tarefa de obter resultados de consulta. Para obter mais informações relativas a métodos de consulta, consulte o exemplo SSearchICM. Consulte a Referência de Programação de Aplicações para obter os métodos de avaliação, execução e executeWithCallback na classe DKDatastoreICM.

Aplicar a linguagem da consulta ao modelo de dados do Content Manager

Para o ajudar a compreender a linguagem da consulta, pode ver conceptualmente o servidor de bibliotecas como um documento XML. A analogia do documento XML é apenas utilizada com o objectivo de explicar a linguagem da consulta. Deste modo, é muito importante lembrar que a representação XML do servidor de bibliotecas é apenas uma representação virtual e os artigos num servidor de bibliotecas não são elementos XML, nem o utilizador irá obter elementos XML quando executar uma consulta. Na representação XML do servidor de bibliotecas, os elementos do modelo de dados do Content Manager são representados da seguinte forma:

Artigos

Em geral, cada artigo do CM é representado por elementos de XML imbricados, em que o elemento XML de nível de topo representa o componente raiz e os elementos XML imbricados representam os componentes descendentes. A imbricação de elementos de XML representa, assim, a hierarquia de componentes.

Componentes de raiz

Um componente de raiz é representado pelo primeiro nível de um elemento XML. Um componente de raiz possui os seguintes atributos de XML: ID ITEMID , STRING COMPONENTID , INTEGER VERSIONID, INTEGER SEMANTICTYPE e outros atributos definidos pelo utilizador do componente. Em servidor de bibliotecas, o IDITEM é único.

Componentes descendentes

Um componente descendente é representado por um elemento de XML imbricado e possui os seguintes atributos: STRING ITEMID, STRING COMPONENTID, INTEGER VERSIONID e outros atributos definidos pelo utilizador do componente.

Tenha em atenção que o COMPONENTID só é único dentro um componente de Content Manager. O ITEMID e o VERSIONID são iguais ao componente raiz do descendente ITEMID e VERSIONID.

Atributos definidos pelo utilizador

Cada atributo definido pelo utilizador é representado por um atributo XML imbricado no elemento XML, que representa o componente que o contém.

Ligações

Apesar das ligações internas e externas não fazerem parte de um artigo incluído no modelo de dados do CM (são armazenados separadamente na tabela de ligações), caso pretenda consultá-las, é conveniente encará-las conceptualmente como fazendo parte do elemento XML que representa o artigo. Isto faz com que as aplicações não tenham de escrever explicitamente junções nas consultas. As ligações representam uma relação de um-para-muitos entre artigos. Tenha em atenção que esta relação só existe entre componentes raiz (uma ligação não pode originar ou produzir um componente descendente).

As ligações que dão origem a um artigo são representadas por elementos XML<OUTBOUNDLINK> com os seguintes atributos: IDREF LINKITEMREF, IDREF TARGETITEMREF e STRING LINKTYPE. O LINKITEMREF é uma referência a um artigo que contém metadados para a ligação. O TARGETITEMREF é uma referência ao artigo indicado pela ligação. O LINKTYPE é o tipo da ligação. De modo idêntico, as ligações

que indicam um artigo são representadas por elementos XML <INBOUNDLINK> com os seguintes atributos: IDREF LINKITEMREF, IDREF SOURCEITEMREF e STRING LINKTYPE. O SOURCEITEMREF é uma referência ao artigo em que a ligação é originada. Para obter mais informações relativas à semântica das ligações, consulte os exemplos SLinksICM e SSearchICM .

Referências (atributos de referências)

Os atributos de referência são representados por atributos XML de tipo IDREF. Uma referência representa uma relação de um-para-um entre um artigo ou um componente e outro artigo. Consequentemente, o destino de uma referência apenas pode ser um componente raiz e não um componente descendente. Um atributo de referência, no entanto, pode ter origem tanto em componentes de raiz como componentes descendentes.

Os atributos de referência podem ser definidos pelo sistema (SYSREFERENCEATTRS) ou definidos pelo utilizador (PublicationRef nas seguintes consultas de exemplo). As referências podem ser cruzadas em ambas as direcções.

O cruzamento transversal inverso de referências é executado de uma forma semelhante ao cruzamento transversal inverso de ligações, tal como foi descrito acima. O utilizador pode conceber o artigo que está a ser referenciado comum elemento de XML denominado REFERENCEDBY, que contém um atributo XML denominado REFERENCER (de tipo IDREF), que indica o componente que faz referência a este artigo. É semelhante ao elemento INBOUNDLINK com o atributo SOURCEITEMREF para o cruzamento transversal inverso de ligações.

As referências também suportam semânticas de eliminação (restrição, cascata, definição de nulo ou sem acção). Para obter mais informações relativas a atributos de referência, consulte os exemplos SReferenceAttrDefCreationICM e SSearchICM.

Documentos

A funcionalidade de pasta facultada no Content Manager através de DKFolder é armazenada como um conjunto de ligações do tipo de ligação "DKFolder" (DK_ICM_LINKTYPE_NAME_DKFOLDER). Todas as relações pasta-conteúdo são modeladas como ligações externas da pasta e como ligações internas aos conteúdos, transformando a pasta na origem e os conteúdos nos destinos de cada ligação. Siga a semântica de pesquisa e de percorrimto de ligações para pesquisar e percorrer pastas. Para obter mais informações, consulte os exemplos SFolderICM e SSearchICM.

Compreender a pesquisa paramétrica

Os artigos são frequentemente obtidos através da iniciação de uma pesquisa de atributos seleccionados. Uma única consulta pode examinar atributos definidos pelo sistema e atributos definidos pelo utilizador dos artigos no servidor de conteúdos. As condições de pesquisa simples são formadas por um nome de atributo, um operador e um valor que são combinados numa cláusula. O Content Manager faculta ao utilizador muitos operadores de comparação para concluir as pesquisas paramétricas. Os operadores incluem:

"="

"< "

"<="

"> "

">="

"!="

"LIKE"

"NOT LIKE"

"BETWEEN"

"NOT BETWEEN"

"IS NULL"

"IS NOT NULL"

Pode especificar condições de pesquisa complexas combinando condições de pesquisa simples numa cláusula, utilizando os operadores Booleanos AND, OR e NOT. Consulte os exemplos de consulta para obter mais detalhes.

Compreender a pesquisa de texto

Ao utilizar o DB2 Universal Database Net Search Extender (NSE), anteriormente conhecido como Text Information Extender (TIE) no CM 8.1, o Content Manager faculta dois tipos de pesquisa de texto: a pesquisa de texto de atributos que contém texto em componentes e a pesquisa de texto de objectos. A principal diferença entre os dois tipos de pesquisa de texto é a forma como o conteúdo é armazenado. Quando define um atributo para que seja pesquisável por texto, está a indicar que se pode pesquisar texto contido na coluna desse atributo. Para tornar um atributo (coluna) pesquisável por texto, o NSE cria um índice de texto. O índice de texto detém informações relativas ao texto que será pesquisado. Estas informações são utilizadas para executar a pesquisa de texto de forma eficaz. Por exemplo, Fred, um administrador de sistemas, cria um tipo de artigo denominado `Journal` que possui uma vista de tipo de componente descendente `Journal_Article`, que este pretende activar para pesquisa de texto. Um dos atributos para `Journal_Article` é `Title`, que Fred activa para pesquisa de texto. Quando Lily, uma agente de seguros, pesquisa o Título que contém a palavra "Java", o sistema pesquisa no índice de texto Título algumas pistas relativas a "Java".

Para obter mais informações relativas ao TIE utilizado no CM 8.1, consulte a documentação do DB2 Universal Database Text Information Extender (TIE). Para obter informações relativas ao NSE utilizado no CM 8.2, consulte a documentação de DB2 Universal Database Net Search Extender (NSE). Na questão da pesquisa de texto na linguagem de consulta de ICM, o NSE e TIE podem ser utilizados alternadamente. Da perspectiva da linguagem de consulta de ICM, não existem diferenças entre a sintaxe de pesquisa de texto e a funcionalidade entre NSE e TIE."

Pesquisar conteúdos de objectos

A pesquisa de conteúdos de objectos funciona de forma diferente. Em vez de indexar directamente uma coluna, o sistema utiliza uma referência à localização do objecto num gestor de recursos. O NSE utiliza a referência para obter o conteúdo quando cria um índice de texto. E um utilizador final que execute uma pesquisa não nota diferenças ao pesquisar objectos armazenados num gestor de recursos. Um administrador de sistemas, no entanto, tem de configurar uma vista de tipo de artigo de recurso de texto, de forma a que o mecanismo de pesquisa localize o conteúdo no gestor de recursos. A pesquisa de texto é executada no atributo "TIREF" do tipo de artigo de recurso, que faz referência aos conteúdos armazenados no gestor de recursos para fins de pesquisa de texto.

Pesquisar documentos

O utilizador pode executar pesquisa de texto nos conteúdos de partes de documento. Uma vista "ICMPARTS" de tipo de componente virtual é suportada na consulta como descendente de todos os documentos do sistema. O atributo "TIEREF" na vista "ICMPARTS" de tipo de componente faz referência aos conteúdos de todas as partes pesquisáveis por texto desse documento, para fins de pesquisa de texto. Consulte os exemplos de consulta para obter a utilização específica desta funcionalidade.

Tornar os atributos definidos pelo utilizador pesquisáveis por texto

Pode tornar os atributos definidos pelo utilizador pesquisáveis por texto, utilizando a APIs DKAttrDefICM e DKItemTypeDefICM. As propriedades predefinidas do índice de texto criado podem ser modificadas utilizando a classe DKTextIndexDefICM. Para obter mais informações nas APIs, consulte a referência de API online ou o exemplo SItemCreationICM.

Compreender a sintaxe de pesquisa de texto

O utilizador pode executar consultas de pesquisa de texto utilizando uma sintaxe pesquisa de texto básica ou avançada.

Pesquisa de texto básica

Visto que a maioria das pesquisas de texto são efectuadas através da listagem de algumas palavras seguidas, a sintaxe de pesquisa de texto básica (simplificada) foi especificamente concebida para simplificar este caso mais comum junto dos utilizadores. A sintaxe também permite a utilização de "+" e "-", bem como a utilização de expressões entre aspas. A pesquisa de texto simplificada é realizada pela utilização de funções "contains-text-basic" e "score-basic". A função "contains-text-basic" é utilizada para pesquisar dentro de atributos e dentro de conteúdos de recursos ou documentos. A função "score-basic" utiliza a mesma sintaxe da função "contains-text-basic" e é utilizada para ordenar resultados com base na ordenação dos resultados da pesquisa de texto. O utilizador não poderá deixar de equiparar a função "contains-text-basic" para 1, caso seja verdadeira, e equipará-la a 0, caso seja falsa. Consulte os exemplos de consulta para obter instruções relativas à forma como estas funções são utilizadas.

As informações adicionais relativas à sintaxe de pesquisa de texto básica incluem:

- Execução de pesquisa de texto não sensível a letras minúsculas e maiúsculas (tal como a sintaxe avançada, por predefinição). Consulte a documentação de NSE para obter as opções de pesquisa sensíveis a letra minúscula e maiúscula.
- Parte-se do princípio que os termos entre aspas são uma expressão.
- Utilização de + (mais) - (menos)
 - + (mais) = o documento deve incluir esta palavra.
 - - (menos) = o documento não deve incluir esta palavra.
 - Quando não se especifica um + ou -, o motor de consulta utiliza um algoritmo para fazer corresponder as palavras ao texto.
- Os operadores booleanos (AND, OR, NOT) não são válidos e são ignorados.
- Os parêntesis na sintaxe básica não são suportados.
- Caracteres globais válidos
 - ? (ponto de interrogação) = representa um carácter único
 - * (asterisco) = representa um determinado número de caracteres arbitrários

Para obter mais informações relativas à pesquisa de texto básica, consulte o exemplo SSearchICM.

Pesquisa de texto avançada

A sintaxe de pesquisa de texto avançada é utilizada para permitir ao utilizador especificar condições mais complexas para a pesquisa de texto. A pesquisa de texto utiliza a sintaxe de pesquisa de texto de NSE e permite funções poderosas como, por exemplo, a pesquisa de proximidade e a pesquisa aproximada. A sintaxe de pesquisa de texto avançada utiliza as funções "contém texto" e "resultado" de forma idêntica ao modo como as funções "contém básico de texto" e "básico de resultado" são utilizadas pela pesquisa de texto básica. As cadeias facultadas às funções avançadas devem encontrar-se em sintaxe de NSE, exceptuando a seguinte situação: altere as aspas para plicas e vice-versa. Por exemplo, a condição CONTAINS (descrição, "IBM")=1 em NSE seria contém texto (@descrição, " 'IBM' ")=1 na linguagem de consulta de CM. Esta acção deve ser efectuada para suportar a simplicidade de escrita em consultas com uma utilização mínima de caracteres de abandono. O utilizador não poderá deixar de equiparar a função "contains-text" para 1, para verificar se é verdadeira, e equipará-la a 0, para verificar se é falsa. Consulte os exemplos de consulta para obter mais detalhes relativos à pesquisa de texto avançada.

Para obter mais informações relativas à pesquisa de texto avançada, consulte o exemplo SSearchICM.

Criar pesquisa de texto e paramétrica combinada

As pesquisas podem ser executadas com base, virtualmente, em qualquer parte de um artigo ou componente, texto dentro de um artigo ou componente, ou texto dentro dos conteúdos de recursos. A pesquisa pode ser executada de um dos seguintes modos:

Pesquisa paramétrica

Uma pesquisa que é baseada nas propriedades do artigo e do componente, atributos, referências, ligações, conteúdos de pastas, etc.

Pesquisa de Texto

Pesquisar no texto.

Pesquisa Combinada

Uma pesquisa utilizando a pesquisa paramétrica e de texto.

Siga os passos seguintes para criar uma pesquisa paramétrica e de texto combinada.

Java

1. Crie um armazenamento de dados de ICM e estabeleça-lhe ligação.
2. Gere uma cadeia de consulta combinada.

```
String queryString =  
    "//Journal_Article [Journal_Author/@LastName = \"Richardt\" +  
    \" AND contains-text (@Text, \" 'Java' & 'XML' \")=1]\";
```
3. Se pretender utilizar as opções de obtenção, diferentes das opções predefinidas, utilize a tabela DKNVPair para agrupar as opções.

```
DKNVPair parms[] = new DKNVPair[3];  
String strMax = "5";  
parms[0] = new DKNVPair(DKConstant.DK_CM_PARM_MAX_RESULTS, strMax);  
parms[1] = new DKNVPair(DKConstant.DK_CM_PARM_RETRIEVE,  
    DKConstant.DK_CM_CONTENT_ATTRONLY | DKConstant.DK_CM_CONTENT_LINKS_OUTBOUND);  
parms[2] = new DKNVPair(DKConstant.DK_CM_PARM_END, null);
```
4. Execute a consulta combinada segundo uma das seguintes três formas distintas: avaliação, execução e executeWithCallback.

```
DKResults resultsCollection =  
    (DKResults)dsICM.evaluate(queryString,  
    DKConstant.DK_CM_XQPE_QL_TYPE,parms);
```
5. Processe os resultados. O procedimento de processamento dos resultados depende do método de execução que utilizou.

Para obter um exemplo completo e documentação adicional, consulte o Exemplo de Instrução de API SSearchICM em CMBROOT\Samples\java\icm.

C++

1. Especificar as opções de pesquisa. Note que o conjunto de opções tem sempre de ter um elemento terminal. Por exemplo, caso pretenda especificar duas opções, a matriz de opções deve possuir três elementos.

```
DKNVPair * parms = new DKNVPair[3];
DKNVPair * pparm = NULL;
    DKString strMax = "5";
DKAny * anyNull = new DKAny();
//Permita a devolução de um máximo de 5 artigos de uma pesquisa
pparm = new DKNVPair(DK_CM_PARM_MAX_RESULTS, strMax);
parms[0] = *pparm;
delete pparm;
//Especifique o conteúdo a ser obtido
pparm = new DKNVPair((long)DK_CM_PARM_RETRIEVE,
    DK_CM_CONTENT_ATTRONLY | DK_CM_CONTENT_LINKS_OUTBOUND);
parms[1] = *pparm;
delete pparm;
pparm = new DKNVPair(DK_CM_PARM_END, *anyNull);
parms[2] = *pparm;
delete pparm;
```

2. Executar a pesquisa. Existem três modos para executar uma pesquisa:

evaluate

Devolve todos os resultados como um conjunto; é bom para conjuntos pequenos.

execute

Devolve um cursor de conjunto de resultados, que o programa de chamada utilizar para executar uma iteração sobre os resultados.

executeWithCallback

Cria um módulo que executa uma iteração sobre o conjunto de resultados e chama o objecto de repetição de marcação para cada bloco de resultados. O programa de chamada utiliza o objecto de repetição de marcação para obter os resultados.

No exemplo a seguir apresentado, apenas se pretendem cinco resultados, por isso é utilizado o método DKDatastoreICM.evaluate.

```
DKResults * resultsCollection = (DKResults *) (dkCollection *)
    dsICM->evaluate(queryString, DK_CM_XQPE_QL_TYPE, parms);
```

3. Visualizar os resultados da pesquisa.

```
// Create an iterator to go through Results collection.
dkIterator* iter = resultsCollection->createIterator();

cout << "Results:" << endl;
cout << "    - Total: " << results->cardinality() << endl;

while (iter->more())
{
    //Each element in the returned array is an item (DDO)
    DKDDO* ddo = (DKDDO*) iter->next()->value();
    cout << "    - Item ID: " << ((DKPidICM*) ddo->getPidObject())
        ->getItemId() << " (" << ((DKPidICM*) ddo->getPidObject())
        ->getObjectType() << ")" << endl;
}
```

4. Limpar.

```
delete(iter);
delete[] parms;
....
```

Para obter um exemplo completo e documentação adicional, consulte o Exemplo de Instrução de API SSearchICM em CMBROOT\Samples\cpp\icm.

Exemplos de consultas

Para o ajudar a compreender melhor a linguagem da consulta e para que possa dar início à escrita de consultas, esta secção fornece-lhe as informações seguintes:

- Um modelo de dados exemplo.
- Uma representação de documentos de XML do modelo de dados.
- Consultas exemplo.
- Gramática da linguagem de consulta.

As consultas exemplo das secções seguintes têm por base o modelo de dados de exemplos de consulta descrito na Figura 13 na página 195. Consulte a ilustração do modelo de dados quando revir as consultas exemplo.

Para obter sintaxe de consulta e exemplos adicionais com base num modelo de dados alternativo, consulte o exemplo SSearchICM.

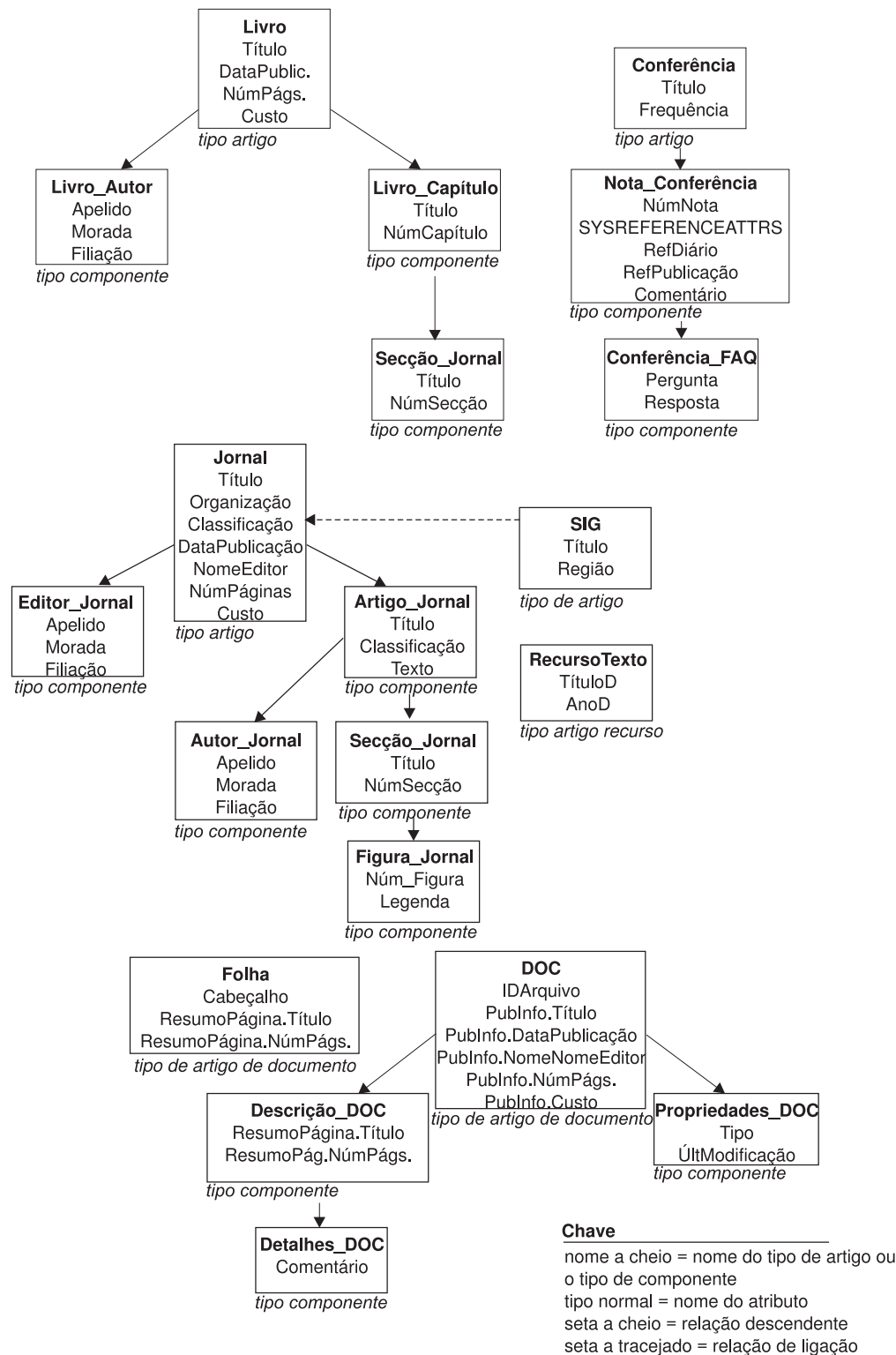


Figura 13. Modelo de dados de exemplos de consulta

O documento de XML seguinte é uma representação do modelo de dados da Figura 13.

Representação XML do modelo de dados de exemplos de consulta:

```

<Journal (ID ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
          INTEGER SEMANTICTYPE, Title, Organization, Classification,
          PublishDate, PublisherName, NumPages, Cost)>
  <Journal_Editor (STRING ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
                  LastName, Address, Affiliation)>
</Journal_Editor>
... (repeating <Journal_Editor>)

  <Journal_Article (STRING ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
                  Title, Classification, Text)>
    <Journal_Section (STRING ITEMID, STRING COMPONENTID,
                     INTEGER VERSIONID, Title, SectionNum)>
      <Journal_Figure (STRING ITEMID, STRING COMPONENTID,
                      INTEGER VERSIONID, FigureNum, Caption)>
</Journal_Figure>
      ... (repeating <Journal_Figure>)
    </Journal_Section>
    ... (repeating <Journal_Section>)

    <Journal_Author (STRING ITEMID, STRING COMPONENTID,
                    INTEGER VERSIONID, LastName, Address, Affiliation)>
</Journal_Author>
    ... (repeating <Journal_Author>)
  </Journal_Article>
  ... (repeating <Journal_Article>)

  <OUTBOUNDLINK (IDREF LINKITEMREF, IDREF TARGETITEMREF,
                STRING LINKTYPE) >
</OUTBOUNDLINK>
  ... (repeating <OUTBOUNDLINK>)

  <INBOUNDLINK (IDREF LINKITEMREF, IDREF SOURCEITEMREF,
                STRING LINKTYPE)>
</INBOUNDLINK>
  ... (repeating <INBOUNDLINK>)

  <REFERENCEDBY (IDREF REFERENCER)>
</REFERENCEDBY>
  ... (repeating <REFERENCEDBY>)
</Journal>
...(repeating <Journal>)

<Book (ID ITEMID, STRING COMPONENTID, INTEGER VERSIONID, INTEGER
        SEMANTICTYPE, Title, PublishDate, NumPages, Cost)>
  <Book_Author (STRING ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
               LastName, Address, Affiliation)>
</Book_Author>
  ... (repeating <Book_Author>)
  <Book_Chapter (STRING ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
                 Title, ChapterNum)>
    <Book_Section (STRING ITEMID, STRING COMPONENTID,
                   INTEGER VERSIONID, Title, SectionNum)>
</Book_Section>
    ... (repeating <Book_Section>)
  </Book_Chapter>
  ... (repeating <Book_Chapter>)

  <OUTBOUNDLINK (IDREF LINKITEMREF, IDREF TARGETITEMREF,
                STRING LINKTYPE) >
</OUTBOUNDLINK>
  ... (repeating <OUTBOUNDLINK>)

  <INBOUNDLINK (IDREF LINKITEMREF, IDREF SOURCEITEMREF,
                STRING LINKTYPE)>
</INBOUNDLINK>
  ... (repeating <INBOUNDLINK>)

```

```

        <REFERENCEDBY (IDREF REFERENCER)>
        </REFERENCEDBY>
        ... (repeating <REFERENCEDBY>)
    </Book>
    ... (repeating <Book>)

<SIG (ID ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
INTEGER SEMANTICTYPE, Title, Region)>
    <OUTBOUNDLINK (IDREF LINKITEMREF, IDREF TARGETITEMREF,
        STRING LINKTYPE) >
    </OUTBOUNDLINK>
    ... (repeating <OUTBOUNDLINK>)

    <INBOUNDLINK (IDREF LINKITEMREF, IDREF SOURCEITEMREF,
        STRING LINKTYPE)>
    </INBOUNDLINK>
    ... (repeating <INBOUNDLINK>)

    <REFERENCEDBY (IDREF REFERENCER)>
    </REFERENCEDBY>
    ... (repeating <REFERENCEDBY>)
</SIG>
... (repeating <SIG>)

<TextResource (ID ITEMID, STRING COMPONENTID, INTEGER VERSIONID,
        INTEGER SEMANTICTYPE, JTitle, JYear)>
    <OUTBOUNDLINK (IDREF LINKITEMREF, IDREF TARGETITEMREF,
        STRING LINKTYPE) >
    </OUTBOUNDLINK>
    ... (repeating <OUTBOUNDLINK>)

    <INBOUNDLINK (IDREF LINKITEMREF, IDREF SOURCEITEMREF,
        STRING LINKTYPE)>
    </INBOUNDLINK>
    ... (repeating <INBOUNDLINK>)

    <REFERENCEDBY (IDREF REFERENCER)>
    </REFERENCEDBY>
    ... (repeating <REFERENCEDBY>)
</TextResource>
... (repeating <TextResource>)

```

Exemplos de consultas

As consultas exemplo facultadas nesta secção têm por base o modelo de dados exemplo, Figura 13 na página 195, e o documento XML exemplo da página 195. Estas são algumas indicações para ajudar o utilizador a compreender os exemplos de consulta:

- Siga a cadeia da consulta como seguiria uma estrutura de directório
- "/" uma única barra indica uma relação descendente directa
- "/" duas barras indicam uma relação descendente
- "." (PONTO) representa o componente actual da hierarquia
- ".." (PONTO-PONTO) representa o ascendente do componente actual
- "@" (Arroba) denota um atributo
- "[]" (parêntesis quadrados) denotam uma instrução condicional ou uma lista
- "=>" (operador DEREERENCE) representa uma acção de ligação ou de referência
- O resultado da consulta deve ser um componente (por exemplo, um atributo não poderá ser o último elemento do caminho)

São facultados exemplos e documentação adicionais no exemplo SSearchICM sample.

Exemplo 1: acesso a componentes

Esta consulta localiza todos os diários.

```
/Diário
```

Explicação: A “/” começa na raiz implícita do documento de XML, que neste caso é o servidor de bibliotecas completo. Cada tipo de artigo é um elemento sob esta raiz. Se LS.xml for o documento de XML que contém o modelo completo, tal como foi descrito anteriormente, então a raiz de documento explícita é o documento (LS.xml).

Exemplo 2: acesso a atributos

Esta consulta localiza todos os artigos de jornal com um total de 50 páginas cada.

```
/Journal[@NumPages=50]
```

Explicação: O predicado @NumPages = 50 avaliar para verdadeiro para todos os jornais que possuem o atributo de Content Manager “NumPages” definido para 50.

Exemplo 3: vários tipos de artigo

Esta consulta localiza todos os livros ou jornais que possuem “Williams” como um dos autores e que possuem um título de secção começado por “XML”.

```
(/Livro | /Jornal)
[(./Journal_Author/@LastName = "Williams"
OR ./Book_Author/@LastName = "Williams")
AND (./Book_Section/@Title LIKE "XML%"
OR ./Journal_Section/@Title LIKE "XML%")]
```

OR

```
(/Book[./Book_Author/@LastName = "Williams"
AND ./Book_Section/@Title LIKE "XML%"])
| (/Journal[./Journal_Author/@LastName = "Williams"
AND ./Journal_Section/@Title LIKE "XML%"])
```

Explicação: As duas consultas acima descritas produzem o mesmo resultado. “./Journal_Author” significa que o componente Journal_Author deverá ser localizado directamente no componente actual no caminho (que, no primeiro caso, é Livro ou Jornal) ou mais abaixo na hierarquia. Tenha em atenção que o operador LIKE é utilizado em conjunção com um carácter global, neste caso “%”.

Exemplo 4: operações aritméticas em condições

Esta consulta localiza todos os jornais com um número de páginas entre 45 e 200.

```
/Journal[@NumPages BETWEEN 49-4 AND 2*100]
```

Explicação: Tenha em atenção que pode executar operações aritméticas para calcular os valores resultantes que serão utilizados com o operador BETWEEN.

Exemplo 5: cruzamento de ligações em sentido descendente

Esta consulta localiza todos os artigos de jornal editados por “Williams” que estão contidos em SIGs com o título “SIGMOD”.

```

/SIG[@Title = "SIGMOD"]/OUTBOUNDLINK
  [@LINKTYPE = "contains"]/@TARGETITEMREF =>
  Journal[Journal_Editor/@LastName = "Williams"]
/Journal_Article

```

Explicação: Este é um exemplo de seguimento de ligações no sentido descendente. O componente virtual OUTBOUNDLINK de XML e o seu atributo TARGETITEMREF são utilizados para cruzar para todos os Jornais e, finalmente, os Journal_Articles subjacentes. O último componente do caminho consiste no que é devolvido como resultado da consulta. O resultado pode ser forçado cruzando apenas tipos de ligação específicos (neste exemplo é “contains”) para um tipo específico de artigos (Jornal, neste exemplo). Devido ao facto da representação conceptual de XML do servidor de bibliotecas encarar as ligações internas e externas como sendo partes de artigos, o operador de anulação de referência pode ser utilizado para que as aplicações não tenham de efectuar a escrita de junções explícitas.

Exemplo 6: cruzamento de ligações em sentido inverso

Esta consulta localiza todos os artigos de qualquer tipo que possuam jornais que custem menos de cinco dólares, com artigos cujo autor é “Nelson”.

```

/Journal[@Cost < 5
AND ../Journal_Author/@LastName = "Nelson"]
/INBOUNDLINK[@LINKTYPE = "contains"]
@SOURCEITEMREF => *

```

Explicação: Este é um exemplo de seguimento de ligações no sentido inverso. O carácter global “*”, a seguir ao operador de anulação de referência “=>” garante que os artigos de QUALQUER tipo são devolvidos como resultado.

Exemplo 7: pesquisa de texto (funções contains-text e score)

Esta consulta localiza artigos de artigo cujo autor é “Richardt” e que contêm o texto “Java” e o texto “XML”. Os resultados são ordenados pelo resultado da pesquisa de texto.

```

//Journal_Article[Journal_Author/@LastName = "Richardt"
AND contains-text(@Text, " 'Java' & 'XML' ")=1]
SORTBY(score(@Text, " 'Java' & 'XML' "))

```

Explicação: Este é um exemplo de execução de uma pesquisa de texto com a função contains-text. Para obter a sintaxe suportada por esta função, consulte a documentação de DB2 Net Search Engine (NSE). Tenha em atenção que a função contains-text deve ser equiparada a 1 para ser verdadeira e a 0 para ser falsa. A função de resultado utiliza a informação de ordenação devolvida por NSE, a qual neste caso é utilizada para ordenar os artigos de diário resultantes por intermédio de SORTBY.

Exemplo 8: pesquisa de texto (função contains-text e ordenação de atributos)

Esta consulta localiza todos os jornais que possuem a palavra “Design” ou a palavra “Index” nos títulos e ordena os resultados por título, em ordem descendente.

```

/Journal
[Journal_Article[contains-text(@Title, " 'Design' | 'Index' ")=1]]
SORTBY (@Title DESCENDING)

```

Explicação: Este é outro exemplo de execução de uma pesquisa de texto com a função contains-text. A ordenação, neste caso, utiliza o operador DESCENDING no atributo "Título". A predefinição de SORTBY é ASCENDING.

Exemplo 9: pesquisa de texto (funções contains-text-basic e score-basic)

Esta consulta localiza todos os artigos de jornal que contêm o texto "Java" e o texto "JDK 1.3", mas não contêm o texto "XML", utilizando a sintaxe de pesquisa de texto simplificada (básica) e ordena os resultados segundo o resultado da pesquisa de texto.

```
//Journal_Article  
[contains-text-basic(@Title, " +Java -XML +'JDK 1.3'")=1]  
SORTBY (score-basic(@Title, " +Java -XML +'JDK 1.3' "))
```

Explicação: Este é um exemplo de execução de uma pesquisa de texto utilizando a sintaxe de pesquisa de texto simplificada. Utilize um "+" para indicar as palavras ou expressões que devem estar presentes no atributo "Título" e, de forma idêntica, utilize um "-" para excluir outras palavras ou expressões. A função score-basic funciona de forma idêntica à função score do exemplo anterior, mas utiliza uma sintaxe simplificada.

Exemplo 10: pesquisa de texto em artigos de recurso

Esta consulta localiza recursos de texto num tipo de artigo de recurso de texto "TextResource" que contém o texto "Java" e o texto "XML".

```
/TextResource[contains-text(@TIEREF, " 'Java' & 'XML'  
")=1]
```

Explicação: Este é um exemplo de execução de uma pesquisa de texto utilizando o recursos do gestor de recursos. Tenha em atenção que o atributo "TIEREF" é utilizado como uma representação do recurso que é representado pelo artigo de tipo "TextResource". A sintaxe de TIE é normalmente utilizada dentro da função contains-text. Para obter a sintaxe suportada por esta função, consulte a documentação de DB2 Net Search Engine (NSE).

Exemplo 11: cruzamento de referências em sentido descendente

Esta consulta localiza todas as perguntas que são colocadas com maior frequência em conferências, para as quais as notas da conferência referem livros com títulos que mencionam o EIP.

```
/Conference/Conference_Note [@PublicationRef =>  
Book[@Title LIKE "%EIP%"]]  
/Conference_FAQ
```

Exemplo 12: cruzamento de referências em sentido descendente

Esta consulta localiza todos os capítulos de livros referidos nas notas das conferências e que estão relacionados com a Internet.

```
/Conference[@Title LIKE "%Internet%"]  
/Conference_Note/@PublicationRef =>  
*/Book_Chapter
```

Exemplo 13: cruzamento de referências na direcção inversa +

Esta consulta localiza todos os componentes que possuem referências que indicam quaisquer livros.

```
/Book/REFERENCEDBY/@REFERENCER => *
```

Exemplo 14: cruzamento de referências na direcção inversa +

Esta consulta localiza todas as perguntas que são colocadas com maior frequência em notas de conferências e que fazem referência a livros sobre XML.


```
/Book[@Title LIKE "XML"]/REFERENCEDBY/@REFERENCER =>  
Conference_Note/Conference_FAQ
```

Explicação: Tenha em atenção que os atributos da referência têm origem dentro do componente Conference_Note, que é o componente que deve ser apresentado primeiro, após o operador de anulação de referência. Esta consulta produz um conjunto de resultados vazio caso, por exemplo, Conference surgir a seguir ao operador “=>”.

Exemplo 15: cruzamento de referências na direcção inversa +

Esta consulta localiza todos os componentes que contêm XML nos seus comentários e que possuem referências que indicam livros.

```
/Book/REFERENCEDBY/@REFERENCER =>  
*[@Remark LIKE "%XML%"]
```

Exemplo 16: função de última versão

Esta consulta localiza todos os jornais da última versão. Por predefinição, são devolvidas todas as versões da vista de tipo do componente indicado que correspondam à consulta. VERSIONID é um atributo definido pelo sistema possuído por todos os tipos de componente.

```
/Journal[@VERSIONID = latest-version (.)]
```

Exemplo 17: função de última versão no destino da anulação de referência

Esta consulta localiza todos os livros da última versão que são referidos nas notas de quaisquer conferências.

```
/Conference/Conference_Note/@SYSREFERENCEATTRS =>  
Book[@VERSIONID = latest-version(.)]
```

Exemplo 18: função de última versão em componentes de carácter global

Esta consulta localiza todos os componentes da última versão que possuem referências que indicam quaisquer livros.

```
/Book/REFERENCEDBY/@REFERENCER => *  
[@VERSIONID = latest-version(.)]
```

Exemplo 19: atributos definidos pelo sistema

Esta consulta localiza todos os componentes raiz com um ID de artigo específico.

```
/*[@ITEMID =  
"A1001001A01J09B00241C95000"]
```

Exemplo 20: pesquisa de texto no modelo de documentos

Esta consulta localiza todos os documentos que contêm a palavra “XML” em qualquer uma das suas partes.

```
/Doc[contains-text(./ICMPARTS/@TIEREF, " 'XML' ")=1]
```

Explicação: A linguagem de consulta faculta um componente virtual denominado “ICMPARTS” que permite o acesso a todos os tipos de artigo de Partes de ICM contidos num tipo de artigo específico de Classificação de Documento.

Exemplo 21: modelo de documentos (acesso a Partes de ICM)

Esta consulta localiza todas as partes do documento com o ID de memória de 555.

```
/Doc[@ArchiveID = 555]/ICMPARTS/  
@SYSREFERENCEATTRS => *
```

Exemplo 22: modelo de documentos (acesso a Partes de ICM)

Esta consulta localiza todas as partes de todos os documentos do sistema.

```
//ICMPARTS/@SYSREFERENCEATTRS => *
```

Explicação: Devido ao facto dos tipos de artigo Documento e Papel terem sido definidos como Documentos no sistema, as Partes de ICM de ambos os artigos são devolvidas no resultado.

Exemplo 23: existência de atributos

Esta consulta localiza todos os componentes de raiz que possuem título.

```
/*[@Title]
```

Explicação: Para eliminar a restrição que estipula apenas a devolução de componentes de raiz, a consulta pode ser novamente escrita para iniciar com duas barras.

```
//*[@Title]
```

Exemplo 24: lista de literais e expressões

Esta consulta localiza todos os jornais que possuem um título igual ao título do artigo, ao título da secção ou IBM Systems Journal".

```
/Journal[@Title = [Journal_Article/@Title,  
./Journal_Section/@Title,"IBM Systems Journal"]]
```

Exemplo 25: lista de literais numéricos

Esta consulta localiza todos os livros que custam 10, 20 ou 30 dólares.

```
/Book[@Cost = [10, 20, 30]]
```

Exemplo 26: lista de um resultado de consulta

Esta consulta localiza todos os jornais ou livros com o título "Star Wars".

```
[/Journal, /Book[@Title = "Star Wars"]]
```

Exemplo 27: grupos de atributos

Esta consulta localiza todos os detalhes em documentos cuja descrição é de, pelo menos, 20 páginas.

```
/Doc[Doc_Description/@PageSummary.NumPages >=  
20]//Doc_Details
```

Explicação: Tenha em atenção, que se um atributo (por exemplo, "NumPages") fizer parte de um grupo de atributos (por exemplo "PageSummary"), o utilizador deve fazer referência a esse atributo como GroupName.AttrName (por exemplo, PageSummary.NumPages). O atributo "@NumPages" não seria localizado em Doc_Description.

Os resultados intermédios obtidos por INTERSECT/EXCEPT não podem ser combinados com operadores aritméticos (monádicos/binários) ou operadores de comparação. Podem ser combinados por operadores de conjunto (UNION/INTERSECT/EXCEPT) ou podem ser apresentados isoladamente.

Exemplos de utilização válida de UNION/INTERSECT/EXCEPT:

1. (/Journal/Journal_Article[@Title = "Content Management"]
EXCEPT
//Journal_Article[@Classification =
"Security"])/Journal_Section

Esta consulta é válida porque o resultado de EXCEPT é o resultado de toda a consulta - não é combinada utilizando operadores.

2. /Journal[(Journal_Editor/@LastName
UNION ./Journal_Author/@LastName) = "Davis"]

Esta consulta é válida porque não existe uma restrição sobre o operador UNION.

3. /Journal[Journal_Article[Journal_Section/@Title INTERSECT
./Journal_Figure/@Caption]/@Title = "Content Management"]

Esta consulta é válida porque o resultado de INTERSECT não é combinado utilizando operadores.

```
4. /Journal[@Title = "VLDB"]  
   UNION /Journal[@Cost = 20]  
   INTERSECT /Journal[@Organization = "ACM"]
```

Esta consulta é válida porque o resultado do operador INTERSECT é combinado utilizando um operador de conjunto (UNION).

Exemplos de utilização válida de INTERSECT/EXCEPT:

```
1. /Journal[(Journal_Editor/@LastName  
   INTERSECT .//Journal_Author/@LastName) = "Davis"]
```

Esta consulta é inválida porque o resultado do operador INTERSECT é combinado utilizando um operador de comparação (=).

```
2. /Journal[(.//Journal_Section/@SectionNum  
   EXCEPT .//Journal_Figure/@FigureNum) + 5 = 10]
```

Esta consulta é inválida porque o resultado de EXCEPT é combinado utilizando um operador aritmético (+).

Compreender a linguagem da consulta

Para suportar funções avançadas da linguagem de consulta (como, por exemplo, os caracteres globais "%" ou "_" dentro de cadeias de texto), são utilizadas sequências de abandono para diferenciar entre os casos em que os caracteres globais são tratados como caracteres normais e os casos em que lhes é atribuído o significado especial de caracteres globais. Para o utilizador, é importante saber quais os caracteres utilizados como globais, pois quando se pretende tratar os caracteres globais como caracteres normais, estes devem ser precedidos por um carácter de abandono. As sequências de abandono também são utilizadas para processar plicas e aspas.

Necessita de adicionar sequências de abandono quando as cadeias utilizadas em consultas contêm caracteres especiais (aspas, apóstrofos) ou caracteres globais (sinal de percentagem, traço de sublinhado, asterisco, ponto de interrogação) ou um carácter de abandono predefinido (uma barra invertida). Este processamento é o mais simples relativamente a cadeias utilizadas em condições de comparação e envolve-se mais no operador LIKE e nas funções de pesquisa de texto. O processamento devido dos caracteres especiais irá garantir uma execução bem sucedida de consultas e a exactidão dos resultados das consultas.

Importante: Utilize os caracteres globais de forma regrada, pois a sua utilização em consultas do utilizador pode aumentar significativamente o tamanho da lista de resultados, o que, por sua vez, pode reduzir o rendimento e devolver resultados de pesquisa inesperados.

Utilizar sequências de abandono com operadores de comparação ("=", "!=" , ">" , "<" , "BETWEEN" e outros)

Aspas "

Coloque dois caracteres de aspas seguidos.

Exemplo:

```
//Journal_Article[@Title = "Analysis of ""The Time Machine"" by H.  
G. Wells himself"]
```

Uma vez que o título do artigo contém o nome do livro entre aspas duplas, "The Time Machine", estas aspas duplas internas devem ser abandonadas.

Plica (apóstrofo) '

Neste caso, não necessita de efectuar o abandono.

Exemplo:

```
/Book[@Title != "Uncle Tom's Cabin"]
```

Utilizar sequências de abandono com o operador LIKE

Aspas "

Coloque dois caracteres de aspas seguidos.

Exemplo:

```
//Journal_Article[@Title LIKE "Analysis of ""The Time Machine"" %"]
```

Uma vez que o título do artigo contém o nome do livro entre aspas duplas, "The Time Machine", estas aspas duplas internas devem ser abandonadas.

Plica (apóstrofo) '

Neste caso, não necessita de efectuar o abandono.

Exemplo:

```
/Book[@Title LIKE "Uncle Tom's Cabin"]
```

Caracteres globais ("% ", "_")

O sinal de percentagem "%" é um carácter global utilizado para representar um número indeterminado de caracteres arbitrários numa cadeia, que é utilizada com o operador LIKE. O traço de sublinhado "_" é um carácter global utilizado para representar um único carácter arbitrário. Se pretender que estes caracteres globais sejam tratados como caracteres normais, deverá efectuar o seguinte:

1. Coloque um carácter de abandono antes do carácter global.
2. Adicione uma cláusula ESCAPE com o carácter de abandono após a expressão LIKE.

Exemplo A:

```
/Book[@Title LIKE "Plato%S%S_mposium"]
```

Este exemplo mostra como os caracteres globais "%" e "_" são utilizados para localizar um livro cuja ortografia do título é incerta.

Exemplo B:

```
//Journal_Article[@Title LIKE "Usage of underscore !_ in query"  
ESCAPE "!"]
```

Visto que a cadeia de pesquisa, neste exemplo, contém o traço de sublinhado "_" como um carácter normal (e não como um carácter global), o utilizador pode abandonar o traço de sublinhado com um carácter de ponto de exclamação "!". Qualquer carácter único pode ser utilizado como um carácter de abandono.

Exemplo C:

```
//Journal_Article[@Title LIKE "_sage of underscore \_ in%" ESCAPE "\"]
```

Nesta consulta, os caracteres globais são utilizados tanto como caracteres normais ("_" abandonado por "\") e como caracteres globais ("_") para deter versões em letra maiúscula e minúscula da palavra "Utilização", bem como "%" para deter várias finalizações da cadeia.

Exemplo D:

```
//Journal_Article[@Title LIKE "Usage of underscore !_ on Yahoo!!" ESCAPE "!"]
```

Pode também utilizar um carácter de abandono como um carácter normal. Para esse efeito, coloque dois caracteres de abandono iguais seguidos, tal como no exemplo de pesquisa de "Yahoo!" apresentado em baixo.

Utilizar sequências de abandono com pesquisa de texto avançada (funções "contains" e "score")

Aspas "

Coloque dois caracteres de aspas seguidos.

Exemplo:

```
//Journal_Article[contains-text (@Title, " 'Analysis of '"The Time Machine'" %' ")=1]
```

Uma vez que o título do artigo contém o nome do livro entre aspas duplas, "The Time Machine", estas aspas duplas internas devem ser abandonadas.

Plica (apóstrofo) '

Coloque dois apóstrofos seguidos. Não é permitido apenas um apóstrofo na pesquisa de texto avançada, pois um conjunto de apóstrofos é utilizado para assinalar um termo ou expressão. Se for apresentado um apóstrofo dentro de um termo, então o apóstrofo deve ser abandonado para o diferenciar do apóstrofo que termina o termo ou expressão.

Exemplo A:

```
/Book[contains-text (@Title, " 'Uncle Tom''s Cabin' ")=1] SORTBY (score (@Title, " 'Uncle Tom''s Cabin' "))
```

Tenha em atenção que Tom''s possui dois apóstrofos.

Exemplo B:

```
/Book[contains-text (@Title, " ('Greek' & 'Plato''s Symposium') & NOT ' Socrates' ")=1] SORTBY (score (@Title, " ('Greek' & 'Plato''s Symposium') & NOT ' Socrates' "))
```

Tenha em atenção que Plato''s possui dois apóstrofos.

Caracteres globais ("%", "_")

Tal como o operador LIKE, a sintaxe avançada utiliza "%" e "_" como caracteres globais. O sinal de percentagem "%" é um carácter global utilizado para representar um número indeterminado de caracteres arbitrários. O traço de sublinhado "_" é um carácter global utilizado para representar um único carácter arbitrário. Se pretender que um carácter global seja tratado como um carácter normal, deverá efectuar o seguinte:

1. Coloque um carácter de abandono antes do carácter global.

2. Adicione uma cláusula ESCAPE após CADA termo em que utiliza o carácter de abandono.

Exemplo A:

```
/Book[contains-text (@Title, " 'Usage of underscore !_ in query'  
ESCAPE '!' ")=1] SORTBY (score (@Title, " 'Usage of underscore !_ in  
query' ESCAPE '!' "))
```

Neste exemplo, é utilizado um ponto de exclamação "!" como carácter de abandono antes do traço de sublinhado.

Exemplo B:

```
/Book[contains-text (@Title, " 'Usage of underscore !_ in query'  
ESCAPE '!' | 'Yahoo! For Dummies' | 'Usage of underscore !_ on  
Yahoo!!' ESCAPE '!' | 'War and Peace' ")=1]
```

Tenha em atenção que deve ser adicionada uma cláusula ESCAPE após cada termo da cadeia de pesquisa de texto em que efectua o escape dos caracteres globais, mesmo que o carácter de abandono seja igual em todos os termos.

Utilizar sequências de abandono com pesquisa de texto básica (funções "contains-text-basic" e "score-basic")

Aspas "

Coloque dois caracteres de aspas seguidos.

Exemplo:

```
//Journal_Article[contains-text-basic (@Title, "Analysis of ""The  
Time Machine"" ")=1]
```

Visto que o título do artigo contém o nome do livro entre aspas, "The Time Machine", estas aspas internas devem ser abandonadas. O título do livro encontra-se entre apóstrofos para o distinguir como expressão.

Plica (apóstrofo) '

Coloque dois apóstrofos seguidos. A sintaxe de pesquisa de texto básica permite termos entre plicas, de modo que um termo pode conter um espaço. Consequentemente, é necessário colocar dois apóstrofos seguidos para diferenciar o caso de um apóstrofo que surge dentro de um termo e o caso de um apóstrofo que inicia um termo novo.

Exemplo A:

```
/Book[contains-text-basic (@Title, "Uncle Tom''s Cabin")=1]SORTBY  
(score-basic (@Title, "Uncle Tom''s Cabin"))
```

Tenha em atenção que Tom''s possui dois apóstrofos.

Exemplo B:

```
/Book[contains-text-basic (@Title, " +Greek +'Plato''s Symposium'  
-Socrates ")=1] SORTBY (score-basic (@Title, " +Greek +'Plato''s  
Symposium' -Socrates "))
```

Tenha em atenção que Plato''s possui dois apóstrofos e que 'Plato's Symposium' se encontra entre plicas, pois trata-se de uma expressão.

Caracteres globais ("*", "?" e "\\")

Antes dos caracteres "*", "?" e "\\", coloque uma barra invertida "\", caso estes caracteres não devam ser tratados como caracteres globais. O asterisco "*" é um carácter global utilizado para representar um número indeterminado de caracteres arbitrários numa pesquisa de texto básica para as funções contains-text-basic e score-basic. O ponto de interrogação "?" é um carácter global utilizado para representar um único carácter arbitrário. No caso da pesquisa de texto básica, a linguagem de consulta faculta uma barra invertida de carácter de abandono "\", para ser utilizado quando o termo a ser pesquisado contém no seu interior o carácter global e o utilizador pretende tratar esse carácter global como um carácter normal.

Exemplo A:

```
/Book[contains-text-basic (@Title, " +Greek +'Plato*s*S?mposium'
-Socrates ")=1] SORTBY (score-basic (@Title, " +Greek
+'Plato*s*S?mposium' -Socrates ))
```

Este exemplo mostra como utilizar a pesquisa de texto básica quando a ortografia de um termo não é certa. Os caracteres "*" e "?" devem, neste caso, ser globais e, por isso, não devem ser abandonados.

Exemplo B:

```
/Book[contains-text-basic (@Title, "Why forgive\?")=1] SORTBY
(score-basic (@Title, "Why forgive\?"))
```

Neste exemplo, o título contém o ponto de interrogação "?" como carácter normal e, por isso, este carácter pode ser abandonado com uma barra invertida.

Exemplo C:

```
//Journal_Section[contains-text-basic (@Title,
"C:\\OurWork\\IsNeverDone")=1] SORTBY (score-basic (@Title,
"C:\\OurWork\\IsNeverDone"))
```

Cada barra invertida que ocorre naturalmente no termo de pesquisa "C:\\OurWork\\IsNeverDone" deve ser abandonada com outra barra invertida.

Utilizar sequências de caracteres globais em Java e C++

Coloque uma barra invertida antes dos caracteres especiais (por exemplo, as aspas e a barra invertida).

Exemplo:

Consulta:

```
/Book[contains-text-basic (@Title, "Why forgive\?")=1]
```

Java

```
String query = "/Book[contains-text-basic (@Title, \"Why forgive\\?\")=1]";
```

C++

```
DKString query ("/Book[contains-text-basic (@Title, \"Why forgive\\?\")=1]");
```

Tenha em atenção que as aspas internas e a barra invertida antes do ponto de interrogação são precedidas por uma barra invertida. Este processamento é inerente às linguagens de programação de Java e C++. Para obter mais informações, consulte as especificações relativas a estas linguagens.

A gramática da linguagem da consulta

A gramática formal da linguagem da consulta é a seguinte:

- (* palavras-chave *)
- AND = ("a" | "A"), ("n" | "N"), ("d" | "D") ;
- ASCENDING = ("a" | "A"), ("s" | "S"), ("c" | "C"), ("e" | "E"), ("n" | "N"), ("d" | "D"), ("i" | "I"), ("n" | "N"), ("g" | "G") ;
- BETWEEN = ("b" | "B"), ("e" | "E"), ("t" | "T"), ("w" | "W"), ("e" | "E"), ("e" | "E"), ("n" | "N") ;
- DESCENDING = ("d" | "D"), ("e" | "E"), ("s" | "S"), ("c" | "C"), ("e" | "E"), ("n" | "N"), ("d" | "D"), ("i" | "I"), ("n" | "N"), ("g" | "G") ;
- DIV = ("d" | "D"), ("i" | "I"), ("v" | "V") ;
- EXCEPT = ("e" | "E"), ("x" | "X"), ("c" | "C"), ("e" | "E"), ("p" | "P"), ("t" | "T") ;
- INTERSECT = ("i" | "I"), ("n" | "N"), ("t" | "T"), ("e" | "E"), ("r" | "R"), ("s" | "S"), ("e" | "E"), ("c" | "C"), ("t" | "T") ;
- LIKE = ("l" | "L"), ("i" | "I"), ("k" | "K"), ("e" | "E") ;
- MOD = ("m" | "M"), ("o" | "O"), ("d" | "D") ;
- NOT = ("n" | "N"), ("o" | "O"), ("t" | "T") ;
- OR = ("o" | "O"), ("r" | "R") ;
- SORTBY = ("s" | "S"), ("o" | "O"), ("r" | "R"), ("t" | "T"), ("b" | "B"), ("y" | "Y") ;
- UNION = ("u" | "U"), ("n" | "N"), ("i" | "I"), ("o" | "O"), ("n" | "N") ;
- IS = ("i" | "I"), ("s" | "S") ;
- NULL = ("n" | "N"), ("u" | "U"), ("l" | "L"), ("l" | "L") ;
- ESCAPE_KEYWORD = ("e" | "E"), ("s" | "S"), ("c" | "C"), ("a" | "A"), ("p" | "P"), ("e" | "E") ;
- KEYWORD = (AND | ASCENDING | BETWEEN | DESCENDING | DIV | EXCEPT | INTERSECT | LIKE | MOD | NOT | OR | SORTBY | UNION | IS | NULL) ;
- (* literais *)
- DIGIT = ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9") ;
- NONZERO_DIGIT = ("1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9") ;
- Exponent = (e | E), ["+" | "-"], DIGIT, { DIGIT }
- INTEGER_LITERAL = "0" | NONZERO_DIGIT, { DIGIT } ;
- FLOAT_LITERAL = DIGIT, { DIGIT }, ".", { DIGIT }, [Exponent] | ["."], DIGIT, { DIGIT }, [Exponent] ;
- (* UNICODE_CHARACTER é o conjunto de todos os caracteres de unicode e sequências de abandono. A sua definição não está incluída neste documento *) (* Os literais de cadeia são delimitados por aspas e podem conter todos os caracteres, excepto aspas. Para incluir aspas como parte do literal da cadeia, especifique duas aspas consecutivas, i.e. um carácter de aspas é abandonado por outro carácter de aspas. Estes serão tratados como um carácter de aspas *)
- STRING_LITERAL = "'", { (UNICODE_CHARACTER - "'") | ("'", "'") }, "'", ;

- (* A sequência de abandono é um único carácter delimitado por aspas. Para especificar as próprias aspas como carácter de escape, especifique duas aspas consecutivas, i.e., um carácter de aspas é abandonado por outro carácter de aspas. Estes serão tratados como um carácter de aspas. Para obter uma explicação completa dos valores legais de ESCAPE_CHARACTER, consulte a Secção de consulta de DB2 SQL no Predicado LIKE. *)
- `ESCAPE_LITERAL = ''' , ((ESCAPE_CHARACTER - ''') | (''' , ''')), ''';`
- `LETTER = ("a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "_" | "$");`
- (* Um IDENTIFIER começa com uma letra (a-z, A-Z) ou um traço de sublinhado, seguido por zero ou mais letras, traços de sublinhado, caracteres de dólar ou dígitos (0-9). Uma palavra-chave só poderá ser um IDENTIFIER caso se encontre entre plicas *)
- `IDENTIFIER = (LETTER, { LETTER | DIGIT }) - KEYWORD | ''' , LETTER, { LETTER | DIGIT }, ''';`
- `ExpressionWithOptionalSortBy = LogicalOrSetExpression, SORTBY, "(", SortSpecList, ")" | Expression ;`
- `Expression = LogicalOrSetExpression ;`
- `SortSpecList = SortSpec, { ",", SortSpec } ;`
- `SortSpec = Expression, [ASCENDING | DESCENDING] ;`
- `LogicalOrSetExpression = LogicalOrSetTerm | LogicalOrSetExpression, (OR | UNION | "|" | EXCEPT), LogicalOrSetTerm ;`
- `LogicalOrSetTerm = LogicalOrSetPrimitive | LogicalOrSetTerm, (AND | INTERSECT), LogicalOrSetPrimitive ;`
- `LogicalOrSetPrimitive = [NOT], SequencedValue ;`
- `SequencedValue = ValueExpression ;`
- `ValueExpression = Comparison ;`
- `Comparison = ArithmeticExpression | Comparison, CompareOperator, ArithmeticExpression, ESCAPE_KEYWORD, ESCAPE_LITERAL | Comparison, CompareOperator, ArithmeticExpression | Comparison, [NOT], BETWEEN, ArithmeticExpression, AND, ArithmeticExpression | Comparison, IS, [NOT], NULL ;`
- `ArithmeticExpression = ArithmeticTerm | ArithmeticExpression, ("+" | "-"), ArithmeticTerm ;`
- `ArithmeticTerm = ArithmeticFactor | ArithmeticTerm, ("*" | DIV | MOD), ArithmeticFactor ;`
- `ArithmeticFactor = ArithmeticPrimitive | ("+" | "-"), ArithmeticFactor ;`
- `ArithmeticPrimitive = BasicExpression, OptionalPredicateList | PathExpression ;`
- `PathExpression = Path | ("/" | "/" / "/"), Path | BasicExpression, OptionalPredicateList, ("/" | "/" / "/"), Path ;`
- `Path = Step | Path, ("/" | "/" / "/"), Step ;`
- `Step = NodeGenerator, OptionalPredicateList ;`
- `NodeGenerator = NameTest | "@", NameTest | "@", NameTest, "=>", NameTest | ".." ;`
- `OptionalPredicateList = {Predicate} ;`
- `Predicate ::= [", Expression, "]" ;`

- BasicExpression = Literal | FunctionName, "(" , OptionalExpressionList, ")" | "(" Expression ")" | ListConstructor | "." ;
- FunctionName = QName ;
- Literal = STRING_LITERAL | INTEGER_LITERAL | FLOAT_LITERAL ;
- OptionalExpressionList = [ExpressionList] ;
- ExpressionList = Expression, { ",", Expression } ;
- ListConstructor = "[", [ListContent], "]" ;
- ListContent = Expression, { ",", Expression } ;
- NameTest = QName | "*" ;
- QName = LocalPart ;
- LocalPart = IDENTIFIER ;
- CompareOperator = "=" | "<" | "<=" | ">" | ">=" | "!=" | [NOT] LIKE ;

Trabalhar com o gestor de recursos

Um gestor de recursos de Content Manager controla uma recolha de recursos geridos (objectos). Gere também a memória necessária a infra-estrutura de Gestão de Memória Hierárquica (HSM), mas o utilizador deve primeiro configurar o gestor de recursos para suportar HSM. Os gestores de recursos possuem funções para suportar serviços específicos de tipos para mais do que um tipo de objecto como, por exemplo, sequenciar, compactar, descompactar, codificar, converter códigos, pesquisar ou extrair texto.

Um único gestor de recursos é utilizado exclusivamente por um servidor da biblioteca. Cada gestor de recursos facultado pelo sistema de Content Manager faculta um sub-conjunto comum de APIs de acesso a dados nativos, que pode ser acedido através da servidor de bibliotecas controladora, de outros componentes de Content Manager e de aplicações, quer local (no mesmo nó de rede) ou remotamente.

Outras APIs de acesso a dados permitem acesso remoto ao gestor de recursos utilizando o suporte de cliente do próprio gestor de recursos e um protocolo padrão de acesso de rede como, por exemplo, CIFS, NFS ou FTP. Para obter o acesso remoto, utilize uma ligação entre cliente e servidor. Os clientes comunicam com os gestores de recursos do Content Manager que utilizam HTTP recorrendo a um servidor standard da Web. A entrega de dados tem por base os protocolos de transferência de dados de HTTP, FTP e FS. Utilizando HTTP, todas as aplicações ou componentes do Content Manager que necessitem de aceder aos conteúdos geridos do Content Manager podem formar dinamicamente um triângulo com um servidor de bibliotecas e um gestor de recursos. Este triângulo forma um caminho de acesso a dados directo entre a aplicação e cada gestor de recursos, mas também um caminho de controlo entre o servidor de bibliotecas e o gestor de recursos. Pode definir as correspondências deste triângulo conceptual para qualquer configuração de rede, que vá desde a configuração de apenas um nó até uma distribuída geograficamente.

A nova arquitectura também acomoda gestores de recursos a que uma aplicação não consegue aceder directamente, tais como um subsistema baseado no sistema central, um sistema de um único utilizador que não implique o controlo de acesso ou um sistema que contém informações de alta sensibilidade, em que o acesso directo de uma aplicação não é permitido pela política empresarial. Neste caso, o acesso a gestores de recursos é indirecto. Tanto o paradigma pull, como o

paradigma push, da transferência de dados são acomodados pelo sistema do Content Manager, bem como as chamadas síncronas e assíncronas.

Para obter informações sobre como configurar um gestor de recursos consulte *Planear e Instalar o Content Management System* e o exemplo `SResourceMgrDefCreationICM` no directório de exemplos, `cmbroot\samples\java\icm`.

Trabalhar com objectos do gestor de recursos

Num Content Manager, cada entidade gerida é intitulada um artigo. Existem dois tipos de artigos, o tipo que representa entidades lógicas puras como, por exemplo, documentos ou pastas, ou entidades que representam objectos de dados físicos como, por exemplo, os dados de texto de um documento de processamento de texto, a imagem digitalizada de uma participação ou o vídeo clip de um acidente de viação. Os objectos possuem um estado e comportamento necessários para processar os dados físicos associados a um documento lógico.

Os objectos de recurso também representam coisas como ficheiros num sistema de ficheiros, clips de vídeo num servidor de vídeo e BLOBs. Durante o tempo de execução, os objectos de recurso são utilizados para aceder aos dados físicos que indicam. Por essa razão, os objectos de recurso no Content Manager possuem um tipo. Ou seja, têm um determinado estado e comportamento. O servidor de bibliotecas e o gestor de recursos gerem um esquema para armazenar o estado de um objecto. Os tipos de objectos de base facultados pelo Content Manager são: BLOBs ou CLOBs genéricos e objectos de conteúdo de Texto, Imagem e Vídeo. O utilizador pode também criar sub-classes dos tipos pré-definidos. Um objecto de recurso também pode ter atributos definidos pelo utilizador, que são utilizados para pesquisar e obter.

Segunda a perspectiva do sistema do Content Manager, cada objecto é representado por um identificador lógico único, o seu Identificador de Recursos Uniforme (URI). O servidor de bibliotecas gere o espaço de nome de URI. Se assim for solicitado, o servidor de bibliotecas correlaciona URIs a Localizadores de Recursos Uniformes (URL). Os URLs são utilizados para obter acesso aos dados físicos. Os URLs não indicam directamente uma área de memória gerida pelo gestor de recursos. Pelo contrário, o gestor de recursos utiliza um espaço de nome local para converter nomes de objectos lógicos em nomes de ficheiros físicos. Os URIs de objectos são criados pelo gestor de recursos específico. O servidor de bibliotecas ou o utilizador final podem sugerir um URI de objecto (o seu nome), mas a decisão é tomada pelo gestor de recursos.

O utilizador pode aceder a um objecto utilizando as APIs do gestor de recursos do Content Manager (armazenar, obter, actualizar, eliminar, etc.) Em alguns casos, pode utilizar APIs que sejam nativas do objecto (sequência, difusão selectiva e etapa) ou o sistema de ficheiros.

Para obter informações relativas a trabalhar com objectos de gestor de recursos, consulte o exemplo `SResourceItemCreationICM` no directório de exemplos, `CMBROOT\Samples\java\icm` ou `CMBROOT\Samples\cpp\icm`.

Gerir documentos em Content Manager

O Content Manager implementa um flexível modelo de dados de gestão de documentos que pode utilizar para gerir objectos empresariais. Os elementos básicos do modelo de dados incluem pastas, documentos e objectos.

Como já foi anteriormente mencionado, os documentos, pastas e outros objectos são todos representados por artigos do sistema de Content Manager. No nível de API, a única diferença entre um documento e uma pasta é o tipo semântico e a respectiva funcionalidade de DKFolder. Um documento é formado por atributos de metadados, que descrevem o documento, incluindo os atributos de valor único (documento, nome, data, assunto), os atributos de valores múltiplos (palavras-chave) e conjuntos de atributos de valores múltiplos (endereço, formado por rua, cidade, estado e código postal).

O modelo de dados de gestão de documentos utiliza partes de documento para associar objectos (artigos de recurso) ao documento. Este modelo suporta mais do que uma parte para construção de um documento. Por exemplo, cada página pode ser uma parte distinta. Para que uma aplicação determine a ordem das partes que compõem um documento, um número de parte é armazenado nas partes de documento. As partes de documento contêm um indicador do objecto (um atributo de referência) que contém outras informações relativas à parte como, por exemplo, o tipo de MIME, o tamanho, o ID de gestor de recursos que contém a parte, o nome de recolha desse gestor de recursos, etc. Cada objecto pode ter diferentes atributos. Por exemplo, uma anotação pode ter X e Y coordenadas, enquanto um registo de nota pode ter o CCSID do texto da nota.

Para ajudar o utilizador a compreender o modelo de dados de gestão de documentos, tenha em consideração a seguinte hipótese de um utilizador que importa um documento utilizando uma aplicação cliente:

- É apresentada uma janela ao utilizador.
- O utilizador introduz (ou selecciona) o nome do ficheiro a importar para o sistema. Por exemplo, um relatório da Polícia.
- O utilizador selecciona o tipo de documento (memorando, participação, plano).
- É aberta uma nova janela, em que o utilizador pode introduzir atributos que descrevem o documento. Por exemplo, a data, número de participação e número da apólice de seguro.
- O utilizador define as partes de um documento e introduz alguns valores destinados aos atributos. Por exemplo, o relatório de polícia é uma parte de documento de uma participação.
- O utilizador termina de inserir as descrições do documento e conclui a tarefa. O relatório da Polícia é criado.

Utilizando as APIs ou JavaBeans, a aplicação cliente então estabelece ligação ao servidor de bibliotecas e ao gestor de recursos. O sistema cria dois artigos (um artigo sem recursos e um artigo de recursos) para armazenar o documento. São criados dois artigos porque o relatório da Polícia contém uma fotografia, que é armazenada no gestor de recursos. O documento é criado com uma única chamada de API. O objecto é então armazenado no gestor de recursos e o gestor de recursos devolve a marca de hora e outros metadados relativos ao objecto. O gestor de recursos cria um atributo de referência para o objecto e insere o atributo de referência no componente descendente do documento. É efectuada uma última chamada ao servidor de bibliotecas para armazenar componentes descendentes e para actualizar os atributos. Todo o processo está limitado por uma transacção, de forma a que nenhuma API gere um documento parcialmente criado.

Após ter criado um documento, pode actualizá-lo. Pode executar dois tipos de actualizações: alterar os metadados ou alterar o conteúdo. O servidor de bibliotecas cria automaticamente um novo registo de artigo com o número de versão seguinte

(se o artigo estiver activado para versões) e copia todos os componentes descendentes que estão associados ao artigo.

Criar o modelo de dados de gestão de documentos

Esta secção ajuda o utilizador a concluir as tarefas principais associadas com o modelo de dados de gestão de documentos:

- Criar um tipo de artigo de documento.
- Criar um documento.
- Actualizar um documento.
- Obter e eliminar um documento.
- Elaborar versões de partes no modelo de dados de gestão de documentos.

Criar um tipo de artigo de documento

Java:

1. Crie um Tipo de Artigo de documento com a classificação de Tipo de Artigo = `DK_ICM_ITEMTYPE_CLASS_DOC_MODEL`, defina o seguinte.

```
docItemTypeDef.setVersionControl
((short)DKConstantICM.DK_ICM_VERSION_CONTROL_ALWAYS);
docItemTypeDef.setVersioningType(DKConstantICM.DK_ICM_ITEM_VERSIONING_FULL);
docItemTypeDef.setDeleteRule(DKConstantICM.DK_ICM_DELETE_RULE_CASCADE);
```
2. Crie uma relação de Tipo de Item para adicionar Partes ao documento. Obtenha `EntityDef` para cada Parte.

```
Parttype = (DKItemTypeDefICM) dsDef.retrieveEntity(PartName);
```
3. Para cada parte, crie uma relação de Tipo de Item e defina os valores.

```
DKItemTypeRelationDefICM itRel = new DKItemTypeRelationDefICM(ds);
itRel.setTargetItemTypeID(PartName);
itRel.setDefaultRMCode((short)1);
itRel.setDefaultACLCode(DKConstantICM.DK_ICM_SUPER_USER_ACL);
itRel.setDefaultCollCode((short)1);
itRel.setDefaultPrefetchCollCode((short)1);
itRel.setVersionControl(DK_ICM_VERSION_CONTROL_NERVER);
```
4. Adicione a relação de Tipo de Item ao Documento (Origem).

```
docItemTypeDef.addItemTypeRelation(itRel);
```
5. Adicione o Tipo de Item de documento à memória permanente.

```
docItemTypeDef.add();
```

C++:

1. Obtenha o objecto de definição do servidor de conteúdos.

```
DKDatastoreDefICM * dsDefICM = (DKDatastoreDefICM *)dsICM->datastoreDef();
```
2. Obtenha o objecto de administração do servidor de conteúdos.

```
DKDatastoreAdminICM * pdsAdmin = (DKDatastoreAdminICM *)dsDefICM->datastoreAdmin();
DKItemTypeDefICM * itemType = NULL;
DKItemTypeRelationDefICM * itemTypeRel = NULL;
DKAttrDefICM* attr = NULL;
```
3. Obtenha um atributo para o tipo de artigo de documento. Se o atributo não existir, crie-o.

```
dkAttrDef *pAttr = dsDefICM->retrieveAttr("docTitle1");
if(pAttr == NULL)
{
    attr = new DKAttrDefICM(dsICM);
    attr->setName("docTitle1"); //attribute name column name
    attr->setType(DK_CM_CHAR);
```

```

        attr->setSize(100);
        attr->setNullable(false);
        attr->setUnique(false);
        attr->add();
        pAttr = attr;
    }
    itemType = new DKItemTypeDefICM(dsICM);
    itemType->setName("DocModelTest");
    itemType->setDescription("This is a test Item Type");
    itemType->setClassification(DK_ICM_ITEMTYPE_CLASS_DOC_MODEL);
    itemType->setAutoLinkEnable(false);
    itemType->setVersionControl((short)DK_ICM_VERSION_CONTROL_ALWAYS);
    itemType->setVersioningType(DK_ICM_ITEM_VERSIONING_FULL);
    itemType->addAttr(pAttr);

```

4. Crie uma relação entre o documento que acabou de criar e parte1. Para isso, obtenha primeiro EntityDef para cada Parte.

```

DKItemTypeDefICM *itemTypePart1 = (DKItemTypeDefICM *)
    dsDefICM->retrieveEntity("ICMBASE");
//int part1ITypeid = itemTypePart1->getItemTypeId();
int part1ITypeid = itemTypePart1->getIntId();

```

5. Para cada parte, crie uma relação de tipo de artigo e defina os valores.

```

DKItemTypeRelationDefICM *itemTypeRelPart1= new DKItemTypeRelationDefICM(dsICM);
itemTypeRelPart1->setTargetItemTypeID(part1ITypeid);
//define o gestor de recursos predefinido
itemTypeRelPart1->setDefaultRMCode((short)1);
//define o código de ACL predefinido
itemTypeRelPart1->setDefaultACLCode(1);

```

6. Defina o conjunto predefinido no qual os recursos de artigo correspondentes a este tipo de artigo devem ser armazenados.

```

itemTypeRelPart1->setDefaultCollCode((short)1);

```

7. Defina o conjunto de obtenção prévia predefinida na qual os recursos de artigo correspondentes a este tipo de artigo devem ser armazenados.

```

itemTypeRelPart1->setDefaultPrefetchCollCode((short)1);
itemTypeRelPart1->setVersionControl((short)DK_ICM_VERSION_CONTROL_NEVER);
itemTypeRelPart1->setSourceItemTypeID(itemType->getIntId());

```

8. adicione a relação de tipo de artigo ao documento (origem).

```

itemType->addItemTypeRelation(itemTypeRelPart1);

```

9. Crie uma relação entre o documento que acabou de criar e a parte2. Para isso, obtenha primeiro EntityDef para cada parte.

```

DKItemTypeDefICM *itemTypePart2 = (DKItemTypeDefICM *)
    dsDefICM->retrieveEntity("ICMANNOTATION");
int part2ITypeid = itemTypePart2->getIntId();

```

10. Para cada parte, crie uma relação de tipo de artigo e defina os valores.

```

DKItemTypeRelationDefICM * itemTypeRelPart2= new
    DKItemTypeRelationDefICM(dsICM);
itemTypeRelPart2->setTargetItemTypeID(part2ITypeid);
itemTypeRelPart2->setDefaultRMCode((short)1);
itemTypeRelPart2->setDefaultACLCode(1);
itemTypeRelPart2->setDefaultCollCode((short)1);
itemTypeRelPart2->setDefaultPrefetchCollCode((short)1);

itemTypeRelPart2->setVersionControl((short)DK_ICM_VERSION_CONTROL_NEVER);

```

```

itemTypeRelPart2->setSourceItemTypeID(itemType->getIntId());

```

11. Adicione a relação de tipo de artigo ao documento (origem).

```

itemType->addItemTypeRelation(itemTypeRelPart2);

```

12. Atualize a definição do tipo de artigo no servidor de bibliotecas.

```

itemType->add();

```

Para obter informações adicionais, consulte o exemplo `SItemTypeCreationICM`.

Criar um documento

Um artigo com o tipo semântico "Documento" pode conter atributos (como os artigos dos mesmos tipos semânticos) e várias "partes" (ao contrário de artigos de outros tipos semânticos) no seu interior. Os passos seguintes explicam ao utilizador o processo de criação de um artigo (com base num tipo de artigo de documento pré-definido) que contém um atributo e uma "parte". Tenha em mente que nos passos seguintes se parte do princípio de que um tipo de artigo denominado "s_simple," com um atributo, denominado "S_varchar, " e uma parte ("ICMBASE") já foi definido.

Java

1. Crie o DDO do documento.

```
DKDDO ddoDocument = dsICM.createDDO("S_simple",
DKConstant.DK_CM_DOCUMENT);
short dataId = 0;
String attrValue = "Test";
```
2. Defina o atributo do documento. Neste caso, partimos do princípio que o tipo de artigo só possui um atributo.

```
dataId = ddoDocument.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,"S_varchar");
ddoDocument.setData(dataId,attrValue);
DKParts parts = null;
// Partes do documento
```
3. Aceda ao conjunto de partes do documento.

```
dataId = ddoDocument.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,
DKConstantICM.DK_CM_DKPARTS);
if (dataId == 0) {
    dataId = ddoDocument.addData(DKConstant.DK_CM_NAMESPACE_ATTR,
DKConstantICM.DK_CM_DKPARTS);
    parts = new DKParts();
    ddoDocument.setData(dataId, parts);
}
else
{
    parts = (DKParts)ddoDocument.getData(dataId);
    if (parts == null)
    {
        parts = new DKParts();
        ddoDocument.setData(dataId, parts);
    }
}
```
4. Crie uma parte de tipo pré-definido "ICMBASE". Esta parte será adicionada ao documento criado. Parte-se do princípio que o documento abaixo criado tem por base um tipo de artigo com apenas uma parte.

```
DKLobICM pLobPart = (DKLobICM) dsICM.createDDO("ICMBASE",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASE);
pLobPart.setPartNumber(1);
// Defina o tipo mime para a parte adicionada
pLobPart.setMimeType("text/plain");
String partValue = "This is a base part";
pLobPart.setContent(partValue.getBytes());
```
5. Adicione a parte criada ao conjunto de "partes". Tenha em atenção que esta acção de guardar é diferida (a alteração não é consolidada no armazenamento de dados até que o DDO do documento seja tornado permanente).

```
parts.addElement((dkDataObjectBase)((DKDDO) pLobPart));
```
6. Torne permanente o documento no armazenamento de dados.

```
ddoDocument.add();
```


C++

```
DKDatastoreDefICM* pdsDef = (DKDatastoreDefICM*) dsICM->datastoreDef();
// Create a new DDO of type DocModelTest and semantic type DK_CM_DOCUMENT
DKDDO* ddoDocument = dsICM->createDDO("DocModelTest",DK_CM_DOCUMENT);
ddoDocument->setData(ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,
DKString("docTitle1")),DKString("this is a string value"));
DKItemTypeDefICM* itemType = (DKItemTypeDefICM*)
    pdsDef->retrieveEntity("DocModelTest");
// Retrieve the collection of DKItemTypeRelationDefICM object for given source
// item type from the persistent store
dkCollection* pRelationColl = itemType->retrieveItemTypeRelations();
dkIterator* pIter = pRelationColl->createIterator();
int noOfPartsToCreate = pRelationColl->cardinality();
DKParts* pPartColl= NULL;
// Create the parts collection for the object if it does not exist
short dataId = ddoDocument->dataId(DK_CM_NAMESPACE_ATTR,DK_CM_DKPARTS);
if (dataId == 0) {
    dataId = ddoDocument->addData(DK_CM_NAMESPACE_ATTR,DK_CM_DKPARTS);
    pPartColl = new DKParts();
    ddoDocument->setData(dataId,pPartColl);
}
else {
    pPartColl =(DKParts*) (ddoDocument->getData(dataId).value());
    if (pPartColl ==NULL) {
        pPartColl = new DKParts();
        ddoDocument->setData(dataId,pPartColl);
    }
}
int i=0;
DKItemTypeRelationDefICM* itemTypeRelPart =NULL;
DKItemTypeDefICM* pEnt =NULL;
// CV v8 BLOB
DKLobICM* pPart =NULL;
DKString str = "This is to test the document model with two parts";
while(pIter->more()) {
    i=i+1;
    itemTypeRelPart=(DKItemTypeRelationDefICM*) pIter->next()->value();
    pEnt = (DKItemTypeDefICM*)
        ((DKDatastoreDefICM*) pdsDef)->retrieveEntity(
            (long)itemTypeRelPart->getTargetItemTypeID());
    pPart =(DKLobICM*) dsICM->createDDO(pEnt->getName(), DK_CM_RESOURCE);
    pPart->setPartNumber(i);
    pPart->setContent(str);

    DKAny any = (dkDataObjectBase*) pPart;
    pPartColl->addElement(any);
}
//Add the DDO to the datastore
ddoDocument->add();
```

Para obter mais informações, consulte o exemplo SDocModelItemICM.

Actualizar um documento

Os passos seguintes explicam ao utilizador o processo de actualização de um artigo de tipo semântico "Documento". Nos passos seguintes, uma parte nova é adicionada e o valor do atributo é actualizado.

Java

1. Atualize o valor do atributo do artigo de documento.

```
String attrValue = "New Value";
short dataId=ddoDocument.dataId
(DKConstant.DK_CM_NAMESPACE_ATTR,"S_varchar");
ddoDocument.setData(dataId,attrValue);
```

2. Aceda ao conjunto de partes do documento.

```
DKParts parts = null;
// Partes do documento
dataId = ddoDocument.dataId(DKConstant.DK_CM_NAMESPACE_ATTR,
DKConstantICM.DK_CM_DKPARTS);
if (dataId == 0) {
    dataId = ddoDocument.addData(DKConstant.DK_CM_NAMESPACE_ATTR,
DKConstantICM.DK_CM_DKPARTS);
    parts = new DKParts();
    ddoDocument.setData(dataId, parts);
}
else
{
    parts = (DKParts)ddoDocument.getData(dataId);
    if (parts == null)
    {
        parts = new DKParts();
        ddoDocument.setData(dataId, parts);
    }
}
```

3. Crie dados para a nova parte.

```
String partValue = "This is an annotation";
```

4. Crie uma parte de tipo pré-definido "ICMANNOTATION". Esta parte será adicionada ao documento criado. Aqui, parte-se do princípio que o documento a ser criado tem por base um tipo de artigo com apenas uma parte. Após a nova parte ser adicionada, o documento terá duas partes.

```
DKLobICM pLobPart = (DKLobICM)dsICM.createDDO("ICMANNOTATION",
DKConstantICM.DK_ICM_SEMANTIC_TYPE_BASE);
pLobPart.setContent(partValue.getBytes());
pLobPart.setPartNumber(2);
```

5. Adicione a parte criada ao conjunto de "partes". Tenha em atenção que esta acção de guardar é diferida (a alteração não é consolidada no armazenamento de dados até que o DDO do documento seja tornado permanente).

```
parts.addElement((dkDataObjectBase)((DKDDO) pLobPart));
```

6. Torne permanente o documento alterado no armazenamento de dados.

```
ddoDocument.update();
```

C++

```
DKDatastoreDefICM* pdsDef = (DKDatastoreDefICM*) dsICM->datastoreDef();
// Create a Document DDO from a PID string
DKString pidString = ....;
//...
DKDDO* ddoDocument = dsICM->createDDO(pidString);
DKPidICM* pPID = (DKPidICM*) ddoDocument->getPidObject();
DKString* pstrItemType = &pPID->getObjectType();
// Retrieves the definition for the item type "DocModelTest" from the
// persistent datastore. The definition is returned as a the *dkEntityDef
// object that is in turn typecast to a DKItemTypeDefICM object
DKItemTypeDefICM* itemType = (DKItemTypeDefICM*)
    pdsDef->retrieveEntity(*pstrItemType);
ddoDocument->setData(ddoDocument->dataId(
    DK_CM_NAMESPACE_ATTR,DKString("docTitle1")),
    DKString("this is a new string value"));

DKString updateString = "This is updated part";
DKSequentialIterator* pSeqIter = NULL;
DKLobICM* pPart = NULL;
short dataId = ddoDocument->dataId (DK_CM_NAMESPACE_ATTR, DK_CM_DKPARTS);
DKParts* pParts = (DKParts*) &ddoDocument->getData(dataId);
if (pParts != NULL) {
    pSeqIter = (DKSequentialIterator*) pParts->createIterator();
}
else return; // quit
pPart = (DKLobICM *) pSeqIter->next();
// Update the existing part with the new content
pPart->setContent(updateString);
// Add a new part to the Document
DKLobICM* pPart1 = (DKLobICM*) dsICM->createDDO ("ICMNOTELOG", DK_CM_RESOURCE);
pPart1->setPartNumber(3);
DKString pTempData = "This is to test the document model with two parts";
pPart1->setContent(pTempData);
DKAny any = (dkDataObjectBase*)(DKDDO*) pPart1;
pParts->addElement(any);
//Update the DDO information in the datastore.
ddoDocument->update();
delete pSeqIter;
```

Para obter mais informações, consulte o exemplo `SDocModelItemICM`.

Obter e eliminar um documento

Para obter um documento, chame `ddo.retrieve(opção)`. Se definir a opção como `DK_ICM_CONTENT_YES`, é obtida a lista TOC de partes, bem como as partes. Caso contrário, é obtida apenas a lista TOC de partes.

Para eliminar um documento, chame `ddo.del()`. O documento e as partes a este anexadas são eliminados. Para obter mais informações relativas à obtenção e eliminação de documentos com partes, consulte o exemplo de API `SDocModelItemICM`.

Elaborar versões de partes no modelo de dados de gestão de documentos

As propriedades da elaboração de partes de documento são determinadas pelas propriedades de elaboração de versões da relação de tipo de artigo com cada tipo de parte seleccionado no tipo de artigo do documento. As características da elaboração de versões de partes de documento incluem:

- Tal como os documentos normais, as partes podem ter um dos três modelos de elaboração de versões: elaborar sempre versões, nunca elaborar versões (predefinição) e elaboração de versões controlada pela aplicação.
- Se um tipo de artigo possuir uma política de versões nunca elaborar versões, as partes do tipo de artigo possuem a mesma política de versões.
- Se um tipo de artigo T possuir uma política de versões elaborar sempre versões e um artigo I de tipo de artigo T e o utilizador modificar (adicionando/eliminando ou actualizando uma parte) os seus atributos ou a sua recolha de partes, é criada uma nova versão do artigo I.
- As partes de documentos, ao contrário dos próprios documentos, não possuem um número máximo de versões.
- Pode obter regras de elaboração de versões, a nível de partes, no objecto de relação do tipo de artigo relativas à parte em questão (base, nota, anotação, etc.).

Os exemplos seguintes mostram como obter as regras de elaboração de versões para uma parte de base do tipo de artigo.

Java

```
String itemTypeName="book"; //tipo de artigo exemplificativo
long partId = DK_ICM_PART_BASE;
DKItemTypeDefICM item =null;
DKDatastoreICM ds = new DKDatastoreICM();
...
item = (DKItemTypeDefICM)ds.datastoreDef.retrieveEntity(itemTypeName;
DKItemTypeRelationDefICM itemRelation =
(DKItemTypeRelationDefICM)item.retrieveItemTypeRelation(partId);
versionControlPolicy = itemRelation.getVersionControl();
```

C++

```
DKString itemTypeName="book"; //example item type
//Especifique o id de parte em relação ao qual necessitamos de informações
//de elaboração de versões
long partId = DK_ICM_PART_BASE;
DKItemTypeDefICM * itemType;
//Retrieve the entity corresponding to the "book" item type
itemType = (DKItemTypeDefICM *)dsICM->datastoreDef()->
retrieveEntity(itemTypeName);
//Retrieve the relation object for the specified part id
DKItemTypeRelationDefICM * itemTypeRelation =
(DKItemTypeRelationDefICM*)itemType->retrieveItemTypeRelation(partId);
//Retrieve the version control policy for the specified part
int versionControlPolicy = itemTypeRelation->getVersionControl();
```

Trabalhar com transacções

As transacções permitem ao Content Manager manter a coerência entre o servidor de bibliotecas e todos os gestores de recursos adjacentes. Uma transacção é uma unidade de trabalho recuperável determinada pelo utilizador, que é formada por uma ou mais chamadas consecutivas de APIs efectuadas através de uma única ligação ao servidor de bibliotecas. A sequência de chamadas consecutivas do método DKDatastoreICM é efectuada directa ou indirectamente, através dos DDOs e XDOs.

O âmbito de uma transacção e a quantidade de trabalho envolvida na mesma é, por predefinição, o trabalho executado por um único método de API (transacção implícita). Este tipo de transacção é recomendado e consiste no melhor âmbito de rendimento de uma transacção. O utilizador pode, no entanto, alterar o âmbito de uma unidade de trabalho, tornando-a maior para incluir várias chamadas de método (transacção explícita), mas a utilização deste tipo de transacção pode gerar algum tempo sistema de rendimento.

Quando termina uma transacção, toda a transacção é consolidada ou sofre um retrocesso. Se for consolidada, todas as alterações ao servidor do Content Manager, efectuadas por chamadas de API dentro da transacção, passam a ser permanentes. Se forem removidas alterações a uma transacção ou esta falhar, todas as alterações efectuadas dentro da transacção são invertidas durante o processamento da remoção de alterações.

A consolidação e a remoção de uma transacção são executadas automaticamente no caso de transacções implícitas. No caso das transacções explícitas que estão a ser utilizadas, a consolidação da transacção é controlada pela aplicação, enquanto que a remoção de alterações da transacção pode ser iniciada por uma aplicação ou de forma automática por parte do sistema do Content Manager. O sistema do Content Manager inicia uma remoção de alterações quando ocorre um erro grave ou quando é necessário processar um bloqueio entre o servidor de bibliotecas e a base de dados.

Dentro de uma transacção, as alterações não consolidadas ao gestor de recursos só são visíveis à aplicação que efectuou as alterações após a transacção estar consolidada. Por exemplo, o utilizador efectua alterações a um artigo do gestor de recursos e armazena-o. Se obtiver esse artigo antes da transacção estar consolidada, o artigo não irá reflectir as alterações que acabou de efectuar. Só verá o artigo actualizado depois da consolidação da transacção.

Não são suportadas transacções concorrentes ou sobrepostas através de uma única ligação do servidor de bibliotecas. Para manter transacções concorrentes, o utilizador deve estabelecer várias ligações entre o servidor de bibliotecas e a base de dados ou iniciar vários processos ou módulos clientes, se estiver a trabalhar com uma aplicação cliente. As aplicações como a IBM WebSphere[®] Application Server operam processos, ligações e sessões.

Quando invocados, os métodos `execute()` e `executeWithCallback()` em `DKDatastoreICM` criam automaticamente uma ligação adicional à base de dados. A nova ligação à base de dados é então utilizada para executar a consulta. Visto que as consultas utilizam uma ligação diferente à base de dados, também possuem um âmbito de transacção diferente de outras operações de servidor de conteúdos. A ligação à base de dados é fechada (ou devolvida ao conjunto, se a partilha estiver activada) quando `DKResultSetCursor` é fechado.

Pontos a ter em consideração ao conceber transacções na aplicação do utilizador

Se um nó cliente ou um servidor de bibliotecas falhar antes da consolidação da transacção, a função de obtenção da base de dados remove imediatamente as alterações à transacção no servidor de bibliotecas. As alterações ao gestor de recursos efectuadas durante a falha são imediatamente desfeitas se o nó cliente e o gestor de recursos estiverem ambos activos. Se a falha tiver ocorrido no nó cliente, o utilizador deve submeter o gestor de recursos a um ciclo do Utilitário de Obtenção Assíncrona, de modo a repor a coerência entre o gestor de recursos e a

servidor de bibliotecas. Antes da execução do utilitário, os servidores ainda possuem integridade de dados. Serão afectadas as operações dos artigos em progresso que evidenciaram a falha e serão também rejeitadas até que o RM seja obtido. Uma falha no decorrer de uma actualização de um objecto impede que se realize outra actualização desse mesmo objecto antes da primeira falha estar consolidada.

Se o gestor de recursos falhar, o utilizador deve executar o utilitário de obtenção assíncrona para remover incoerências. No OS/390, o gestor de recursos tem capacidades de transacção nativas, tal como o Object Access Method (OAM), que são utilizadas para obter mais expedientemente.

Precauções ao utilizar transacções explícitas

No caso de transacções explícitas, nas quais controla o âmbito da transacção utilizando `DKDatastoreICM.startTransaction()` e `DKDatastoreICM.commit()`, deverá ter precaução ao desenvolver uma aplicação na qual trabalha com Documentos do Content Manager com partes e ao executar operações de criação, obtenção, actualização e eliminação (CRUD) de `DKLobICM`. Ao executar estas operações, deverá executar operações de CRUD próximas, o mais possível, do final da transacção. Também deverá manter uma transacção o mais breve possível, pois uma transacção longa aumenta o potencial da ocorrência de problemas de bloqueio da base de dados.

Os problemas de bloqueio são mais notórios ao actualizar um artigo e quando a aplicação opta por não consolidar imediatamente a transacção. Enquanto a transacção não for consolidada, o artigo que está a ser actualizado continua a ser visível a outras aplicações. Quando outro utilizador tenta aceder ou visualizar o artigo, esse utilizador fica bloqueado até que a transacção de actualização seja consolidada. O mesmo problema (bloqueio da base de dados) ocorre ao criar novos artigos numa pasta. Se a pasta for visível a outro utilizador, e esse utilizador tentar obter o novo artigo, o utilizador é bloqueado até que a transacção seja consolidada. O tempo anterior à consolidação da transacção corresponde ao tempo que o utilizador está bloqueado.

A melhor abordagem para evitar o bloqueio da base de dados é consolidar frequentemente as transacções e evitar transacções de execução longa. Se o utilizador tiver de executar operações de CRUD, recomenda-se que execute estas operações quando tiver a certeza de que ninguém irá aceder aos artigos a serem actualizados.

Utilizar operações de dar entrada e saída em transacções

O Content Manager suporta operações de dar entrada e saída em artigos. A operação de dar saída é chamada para adquirir um bloqueio de escrita permanente para artigos. Quando é dada saída de um artigo por parte de um utilizador, outros utilizadores não podem actualizá-lo, apesar de o poderem obter e visualizar. O utilizador deverá chamar a operação para dar saída antes de actualizar ou reindexar um artigo, independentemente do modo da transacção (implícito ou explícito) que utilizar. Após terminar a acção no artigo, chame a operação de dar entrada para libertar o bloqueio permanente e disponibilizar o artigo para que possa ser actualizado por parte de outros utilizadores. Após criar um artigo, o utilizador possui a opção de o manter em estado de saída para impedir que outros utilizadores o alterem antes de concluir o trabalho. Se der saída (ou entrada) de um artigo utilizando uma transacção explícita, esta operação será anulada caso sejam removidas as alterações à transacção. Se der saída de um artigo utilizando

uma transacção implícita, esta acção é consolidada. É da responsabilidade da aplicação voltar a dar entrada do artigo, utilizando opções ou métodos de entrada.

Processamento de transacções

O âmbito da transacção pode ser controlado por uma chamada de API cliente, mas este controlo deve ser cuidadosamente concebido. Para agrupar um conjunto de chamadas de API numa transacção, o utilizador deve construí-lo explicitamente concluindo os seguintes passos:

1. Chame o método `startTransaction()` da classe `DKDatastoreICM`. O utilizador trabalhará com métodos `DKDatastoreICM` para concluir todos os passos da transacção.
2. Chame todas as APIs que pretende incluir na transacção segundo a ordem que pretende que estas sejam chamadas.
3. Chame os métodos de consolidação ou remoção de alterações para terminar a transacção.

Todas as chamadas de API efectuadas entre os métodos `startTransaction()` e `commit()` ou `rollback()`, são tratadas como uma transacção.

Todas as APIs podem ser incluídas em transacções, excepto se existir indicação em contrário. Recorra à Consulta Online de API para obter detalhes. Algumas APIs administrativas não podem ser incluídas em transacções explícitas. Por exemplo, o método para definir ou actualizar tipos de artigos.

De seguida encontra-se a lista de métodos de classe envolvidos em transacções do Content Manager em relação à criação e actualização de artigos.

DKDatastoreICM.startTransaction()

Inicia uma transacção explícita.

DKDatastoreICM.commit()

Consolida as alterações da transacção efectuadas na base de dados.

DKDatastoreICM.rollback()

Remove alterações à transacção da base de dados.

DKDatastoreICM.checkOut()

Adquire um bloqueio de escrita permanente sobre um artigo.

DKDatastoreICM.checkIn()

Liberta um bloqueio de escrita permanente anteriormente adquirido.

DKDatastoreICM.add()

Cria um novo artigo na base de dados.

DKDatastoreICM.updateObject()

Actualiza um artigo. Deve ser dada saída do artigo antes de chamar este método.

DKDatastoreICM.retrieveObject()

Obtém um artigo da base de dados.

DKDatastoreICM.deleteObject()

Elimina um artigo da base de dados.

DKDatastoreICM.moveObject()

Reindexa um artigo. Move um artigo de um tipo de artigo para outro. Deve ser dada saída do artigo antes de chamar este método.

Outra excelente fonte de informações relativas a transacções é o exemplo `SItemUpdateICM`.

Encaminhar um documento através de um processo

O Content Manager faculta um serviço de encaminhamento de documentos integrado para o auxiliar a encaminhar documentos através de um processo empresarial. As APIs de encaminhamento de documentos permite-lhe construir novas aplicações utilizando encaminhamento de documentos, ou adicionar a funcionalidade de encaminhamento de documentos às aplicações do utilizador. Encaminhamento de documentos faculta ao utilizado as seguintes funções:

- Sincronização de todos os artigos num processo de encaminhamento de documentos porque as funções de encaminhamento de documentos estão incluídas em transacções do Content Manager.
- Apresentação apenas do trabalho ao qual o utilizador tem acesso.
- Pista de auditoria única que inclui registos da criação, modificação e encaminhamento de documentos.

Para obter os conceitos básicos de encaminhamento de documentos, consulte o *Manual de Administração do Sistema*. Consulte também os exemplos, pois estes constituem uma excelente fonte de informações adicionais relativas ao encaminhamento de documentos.

Compreender o processo do encaminhamento de documentos

O encaminhamento de documentos é formado por processos, nós de trabalho, listas de trabalho e pacotes de trabalho. O administrador do sistema cria os nós de trabalho, processos e listas de trabalho por intermédio do cliente de administração do sistema. Um processo é formado por nós de trabalho. Cada nó de trabalho do processo consiste num passo diferente do processo. O utilizador pode criar um processo que se ramifique em várias direcções. O utilizador determina qual a ramificação para a qual segue o nó de trabalho. O utilizador pode escolher numa lista de selecções possíveis definida pelo administrador do sistema. Pode definir uma saída de servidor quando define um nó de trabalho. Pode definir saídas de servidor para entrar num nó de trabalho, sair de um nó de trabalho e para notificar o utilizador quando o limite de sobrecarga é atingido. Quando é iniciado um processo, é criado um pacote de trabalho. O pacote de trabalho é o elemento de encaminhamento e contém os atributos do trabalho. Os atributos do pacote de trabalho são formados pelo PID de artigo, pela prioridade, proprietário, etc.

Os pontos de recolha são nós de trabalho com função adicional. Um pacote de trabalho num nó de ponto de recolha prossegue para o nó de trabalho seguinte do processo, assim que existir o número de artigos especificado de um tipo de artigo especificado na pasta especificada. As listas de trabalho definem os pacotes de trabalho atribuídos a um utilizador. Pode ter uma ou mais do que uma lista de trabalhos. Cada lista de trabalhos pode incluir um ou mais do que um nó de trabalho. Pode especificar a ordem dos pacotes de trabalho na lista de trabalhos segundo a prioridade ou data. Pode também definir a ordem de nós de trabalho na lista de trabalhos.

Ao obter listas de trabalho, pode filtrar os resultados para incluir ou excluir trabalho suspenso. Os pacotes de trabalho também podem encontrar-se em estado de notificação. O estado de *Notificação* sucede quando os pacotes de trabalho estão no nó para além do tempo especificado pelo administrador. Tenha em mente que

um nó de trabalho pode fazer parte de mais do que uma lista de trabalhos. O número de pacotes devolvidos numa lista de trabalhos é definido pelo administrador do sistema.

Essas operações básicas que pode executar utilizando encaminhamento de documentos incluem:

- Iniciar um processo
- Terminar um processo
- Continuar um processo
- Suspender um processo
- Retomar um processo
- Obter trabalho de uma lista de trabalhos.
- Obter o artigo seguinte numa lista de trabalhos.
- Definir, actualizar e eliminar um processo
- Definir, actualizar e eliminar um nó de trabalho
- Definir, actualizar e eliminar uma lista de trabalhos.

Configurar um processo de encaminhamento de documentos

Existem nove APIs que pode utilizar para implementar a funcionalidade de encaminhamento de documentos na aplicação do utilizador. Poderá encontrar os detalhes relativos a estas APIs e a métodos em *referência de API online*. As nove APIs de encaminhamento de documentos incluem:

DKDocRoutingServiceICM

Esta classe fornece os métodos de gestão de um processo: iniciar, terminar, continuar, suspender e retomar.

DKDocRoutingServiceMgmtICM

Esta classe fornece os métodos para gerir as classes auxiliares: DKProcessICM, DKWorkNodeICM e DKWorkListICM. Pode aceder ao objecto de DKDocRoutingServiceMgmtICM a partir do objecto de DKDocRoutingServiceICM.

DKProcessICM

Esta classe representa um processo no servidor da biblioteca.

DKWorkNodeICM

Esta classe representa um nó de trabalho no servidor de bibliotecas.

DKWorkListICM

Esta classe representa uma lista de trabalhos no servidor de bibliotecas.

DKRouteListEntryICM

Esta classe define o encaminhamento que o processo pode realizar (de, para). Um objecto de processo (ou seja, DKProcessICM) contém uma recolha de objectos de encaminhamento (ou seja, DKRouteListEntryICM).

DKCollectionResumeListEntryICM

Esta classe representa uma entrada na lista de continuação para os nós de trabalho. Um nó de trabalho contém uma recolha de objectos DKCollectionResumeListEntryICM.

DKWorkPackageICM

Esta classe representa um pacote de trabalho no servidor de bibliotecas. Quando é iniciado um processo, é criado um pacote de trabalho.

DKResumeListEntryICM

Esta classe representa uma lista de continuação. Um pacote de trabalho pode conter uma recolha de listas de retoma.

Criar objectos de serviço de encaminhamento de documentos

Os exemplos seguintes demonstram como criar um objecto de encaminhamento de documentos.

Java

```
//The DKDocRoutingServiceMgmtICM object is a helper class that provides
// methods to manage DKProcessICM, DKWorkNodeICM, and DKWorkListICM
// and the meta-data required to define these objects
DKDocRoutingServiceMgmtICM routingMgmt = new
    DKDocRoutingServiceMgmtICM(dsICM);

//The DKDocRoutingServiceICM class provides the core routing services
//like starting, terminating, continuing, suspending, and resuming a process.

DKDocRoutingServiceICM routingService = new DKDocRoutingServiceICM(dsICM);
```

C++

```
//provides methods to manage DKProcessICM, DKWorkNodeICM,
//and DKWorkListICM and the meta-data required to define these objects
DKDocRoutingServiceMgmtICM* routingMgmt = new
    DKDocRoutingServiceMgmtICM(dsICM);

//The DKDocRoutingServiceICM class provides the core routing services
// such as starting, terminating, continuing, suspending, and resuming
//a process.
DKDocRoutingServiceICM* routingService = new
    DKDocRoutingServiceICM(dsICM);
```

Para obter um exemplo completo, consulte o exemplo `SDocRoutingDefinitionCreationICM`.

Definir um novo nó de trabalho normal

Um nó de trabalho constitui um passo na definição de um processo de encaminhamento de documentos. A aplicação do utilizador pode sair do nó em qualquer altura. A aplicação do utilizador é responsável pela validação dos seus critérios de saída.

Java

```
// Create new Work Node Object.
DKWorkNodeICM workNode1 = new DKWorkNodeICM();
//Choose a Name that is 15 characters or less length
workNode1.setName("S_fillClaim");
//Choose a Description for more information than the name.
workNode1.setDescription("Claimant Fills Out Claim");
// Sets the value of the maximum time that an item can spend at this
// work node (in minutes).
workNode1.setTimeLimit(100);
// Specify max number of work packages that can be at this node
//at any given time
workNode1.setOverloadLimit(200);

// Set the type to be a regular work node
workNode1.setType(0);

//Add the new work node definition to the document routing
//managment object
routingMgmt.add(workNode1);
```

C++

```
// Create new Work Node Object.
DKWorkNodeICM* workNode1 = new DKWorkNodeICM();
// Choose a Name that is 15 characters or less length
workNode1->setName("ValidateCreditCard");
// Choose a Description for more information than the name.
workNode1->setDescription("Buyer's credit card is validated with
the credit card agency");

// Sets the value of the maximum time that an item can spend at
//this work node (in minutes).
workNode1->setTimeLimit(100);
// Specify max number of work packages that can be at this node
// at any given time
workNode1->setOverloadLimit(200);

// Set the type to be a regular work node
workNode1->setType(0);

//Add the new work node definition to the document routing
//management object
routingMgmt->add(workNode1);

// Free memory. This object will no longer be needed.
delete(workNode1);
```

Para obter um exemplo completo relativo à criação de nós de trabalho, consulte o exemplo `SDocRoutingDefinitionCreationICM`.

Enumerar nós de trabalho

O método `listWorkNodeNames` enumera todos os nomes de nós de trabalho no servidor de bibliotecas e o método `listWorkNodes` devolve uma recolha de objectos de `DKWorkNodeICM` representando um nó de trabalho no servidor de bibliotecas.

Java

```
// Obtain the document routing management object.
// Obtain the Routing Management object.
DKDocRoutingServiceMgmtICM routingMgmt = new
    DKDocRoutingServiceMgmtICM(dsICM);

// Obtain all Work Nodes in the System.
dkCollection workNodes = routingMgmt.listWorkNodes();
System.out.println("Work Nodes in System: (" +workNodes.cardinality()+")");
dkIterator iter = workNodes.createIterator();
while(iter.more())
{
    DKWorkNodeICM workNode = (DKWorkNodeICM) iter.next();
    if(workNode.getType()==0)
        System.out.println(" Normal Node - "+workNode.getName()+":
            "+workNode.getDescription());
    else
        System.out.println(" Collection Pt - "+workNode.getName()+":
            "+workNode.getDescription());
}
```

C++

```
// Obtain the document routing management object.
DKDocRoutingServiceMgmtICM* routingMgmt =
    new DKDocRoutingServiceMgmtICM(dsICM);

// Obtain the collection containing all the work nodes in the system.
dkCollection* workNodes = routingMgmt->listWorkNodes();
if (workNodes && (workNodes->cardinality()>0) )
{
    cout << "Work Nodes in System: (" << workNodes->cardinality() << ")"
        << endl;
    dkIterator* iter = workNodes->createIterator();
    while(iter->more())
    {
        DKWorkNodeICM* workNode = (DKWorkNodeICM*) iter->next()->value();
        if(workNode->getType()==0)
        {
            cout << " Normal Node - " << workNode->getName() << ": " <<
                workNode->getDescription() << endl;
        }
        else
        {
            cout << " Collection Pt - " << workNode->getName() << ": " <<
                workNode->getDescription() << endl;
        }
        delete(workNode);
    }
    delete(iter);
    delete(workNodes);
}
delete(routingMgmt);
```

Para obter mais informações relativas a enumerar nós de trabalho, consulte o exemplo SDocRoutingListingICM.

Definir um novo ponto de recolha

Um ponto de recolha é um nó de trabalho com critérios de saída definidos pelo sistema, especificamente aplicáveis a pastas de encaminhamento. Um conjunto de

requisitos pode ser especificado, o qual deve estar em conformidade, antes do processo poder resumir ou avançar para além deste ponto.

Java

```
// Create a new Work Node Object. This will be
//the collection point
DKWorkNodeICM collectionPoint = new DKWorkNodeICM();

// Choose a Name with 15 characters or less.
collectionPoint.setName("S_gatherAll");

// Choose a Description for more information than the name.
collectionPoint.setDescription("Gather Claim,Police Report,Policy,& Photos");

// Sets the value of the maximum time that an item can spend at this
// work node (in minutes).
collectionPoint.setTimeLimit(100);

// Specify max number of work packages that can be at this node.
collectionPoint.setOverloadLimit(200);
// Set the type of node to be a collection point.
collectionPoint.setType(1);

// Create the "Resume" List, which is the list of document types
//that the process must wait for before moving on to the next node.
//A list will be created to hold "resume entries" which are descriptions
//of requirements that must be met before the process can move on.
// Create a List/Collection to hold all Resume Entries.
dkCollection resumeList = new DKSequentialCollection();
// Create as many requirements, or "Resume List Entries", which
//specify what Item Types it must wait for. The process cannot
//pass this collection point unless the specified number of Item
//(DDO) of the specified Item Type reaches this collection point.
DKCollectionResumeListEntryICM resumeRequirement = new
    DKCollectionResumeListEntryICM();
// Set the Item Type Name Folder Item that is being routed.
resumeRequirement.setFolderItemTypeName("S_simple");
// Make the collection wait for an Item of the specified Item Type
//to be added to the folder before proceeding.
resumeRequirement.setRequiredItemTypeName("S_autoClaim");
// Specify the number of Items of the specified Item Type that it
//must wait for.
resumeRequirement.setQuantityNeeded(1);
// Add the requirement (Entry) to the List of Requirements (Resume List).
resumeList.addElement(resumeRequirement);
resumeRequirement = new DKCollectionResumeListEntryICM();
resumeRequirement.setFolderItemTypeName("S_simple");
resumeRequirement.setRequiredItemTypeName("S_policeReport");
resumeRequirement.setQuantityNeeded(1);
// When all requirements (resume list entries) have been added to the
//list of requirements (resume list), set the resume list in the
//collection point.
collectionPoint.setCollectionResumeList(resumeList);
// Add the new collection point definition to the document routing
// management object
routingMgmt.add(collectionPoint);
```

C++

```
// Create a new Work Node Object.This will be the collection point
DKWorkNodeICM* collectionPoint = new DKWorkNodeICM();
// Choose a Name with 15 characters or less.
collectionPoint->setName("GatherOrderDetails");
// Choose a Description for more information than the name.
collectionPoint->setDescription("Gather all the information related to the
order, shipping mechanism and shipping address");
// Sets the value of the maximum time that an item can spend at this
//work node (in minutes).
collectionPoint->setTimeLimit(100);
// Specify max number of work packages that can be at this node.
collectionPoint->setOverloadLimit(200);

// Set the type of node to be a collection point.
collectionPoint->setType(1);
// Create the "Resume" List, which is the list of document types
//that the process must wait for before moving on to the next node.
//A list will be created to hold "resume entries" which are descriptions
//of requirements that must be met before the process may move on.

// Create a List / Collection to hold all Resume Entries.
dkCollection* resumeList = new DKSequentialCollection();

// Create as many requirements, or "Resume List Entries", which specify
//what Item Types it must wait for. The process cannot pass this
//collectionpoint unless the specified number of Item (DDO) of the
//specified Item Type reaches this collection point.
DKCollectionResumeListEntryICM* resumeRequirement = new
DKCollectionResumeListEntryICM();
// Set the Item Type Name Folder Item that is being routed.
resumeRequirement->setFolderItemTypeName("book");

// Make the collection wait for an Item of the specified Item Type to
//be added to the folder before proceeding.
resumeRequirement->setRequiredItemTypeName("AnItemType");

//Specify the number of Items of the specified Item Type that
//it must wait for.
resumeRequirement->setQuantityNeeded(1);

// Add the requirement (Entry) to the List of Requirements (Resume List).
resumeList->addElement(resumeRequirement);
resumeRequirement = new DKCollectionResumeListEntryICM();
resumeRequirement->setFolderItemTypeName("book");
resumeRequirement->setRequiredItemTypeName("AnotherItemType");
resumeRequirement->setQuantityNeeded(1);
resumeList->addElement(resumeRequirement);

// When all requirements (resume list entries) have been added to
//the list of requirements (resume list), set the resume list in the
//collection point.
collectionPoint->setCollectionResumeList(resumeList);
// Add the new collection point definition to the document routing
// de encaminhamento de documentos.
routingMgmt->add(collectionPoint);

//Free the memory associated with this collection point
delete(collectionPoint);
```

Para obter mais informações relativas à definição de pontos de recolha, consulte o exemplo SDocRoutingDefinitionCreationICM.

Definir uma lista de trabalho

Uma lista de trabalhos é composta por um ou mais nós de trabalho a partir dos quais um utilizador obtém uma lista de pacotes de trabalho ou o pacote de trabalho "seguinte". Um nó de trabalho pode fazer parte de mais do que uma lista de trabalhos. A utilização de uma lista de trabalhos permite a um administrador, ou a uma aplicação do utilizador, alterar dinamicamente atribuições de trabalho sem contactar utilizadores finais.

Java

```
// Create a new work list.
DKWorkListICM workList = new DKWorkListICM();
// Choose a name of 15 characters or less.
workList.setName("S_fillClaimWL");
workList.setDescription("Work List Covering Fill/Submit Claim Work Node.");
//Specify that work packages returned by the work list will be sorted by time
workList.setSelectionOrder(DKConstantICM.DK_ICM_DR_SELECTION_ORDER_TIME);
//Specify that the work packages returned will be the one that are not
// in the suspend state
workList.setSelectionFilterOnSuspend
    (DKConstantICM.DK_ICM_DR_SELECTION_FILTER_NO);

//Specify that work packages returned are not the ones in the notify state
workList.setSelectionFilterOnNotify
    (DKConstantICM.DK_ICM_DR_SELECTION_FILTER_NO);

// Specify that at most 100 work packages should be listed in
// this work list
workList.setMaxResult(100);
String[] wnNames = {"S_fillClaim"};
workList.setWorkNodeNames(wnNames);

//Add the new work list definition to the document routing
//management object
routingMgmt.add(workList);
```

C++

```
// Create a new worklist.
DKWorkListICM* workList = new DKWorkListICM();
//Choose a name of 15 characters or less.
workList->setName("ValidateWorkList");
workList->setDescription("worklist Covering the credit card
    validation work node.");

//Specify that work packages returned by the worklist will be
//sorted by time
workList->setSelectionOrder(DK_ICM_DR_SELECTION_ORDER_TIME);

//Specify that the work packages returned will be the one that are not
//in the suspend state
workList->setSelectionFilterOnSuspend(DK_ICM_DR_SELECTION_FILTER_NO);

//Specify that work packages returned are not the ones in the notify state
workList->setSelectionFilterOnNotify(DK_ICM_DR_SELECTION_FILTER_NO);

//Specify that at most 100 work packages should be listed in this worklist
workList->setMaxResult(100);

DKString* wnNames = new DKString[1];
wnNames[0] = "ValidateCreditCard";
//Add the work node to the worklist
workList->setWorkNodeNames(wnNames,1);

//Add the new worklist definition to the document routing management
//object
routingMgmt->add(workList);

//Free the memory associated with this worklist
delete(workList);
```

Para obter mais informações relativas à definição de listas de trabalho, consulte o exemplo `SDocRoutingDefinitionCreationICM`.

Enumerar listas de trabalho

O método `listWorkListNames` enumera todos os nomes de listas de trabalho no servidor de bibliotecas e o método `listWorkLists` devolve uma recolha de objectos de `DKWorkListICM` que representam uma lista de trabalhos no servidor de bibliotecas.

Java

```
// Obtain the document routing management object.
// Obtenha o objecto de Gestão de Encaminhamento.
DKDocRoutingServiceMgmtICM routingMgmt =
    new DKDocRoutingServiceMgmtICM(dsICM);
// Obtain all Work Lists in the System.
dkCollection workLists = routingMgmt.listWorkLists();
System.out.println("Work Lists in System: (" +workLists.cardinality()+")");
dkIterator iter = workLists.createIterator();
while(iter.more())
{
    DKWorkListICM workList = (DKWorkListICM) iter.next();
    System.out.println("    - " +workList.getName()+":
        "+workList.getDescription());
}
```


C++

```
dkCollection* workLists = routingMgmt->listWorkLists(); // Obtain all
// Work Lists in the System.

if (workLists && (workLists->cardinality()>0) )
{
    cout<<"Work Lists in System: ("<<workLists->cardinality()<<")"<<endl;
    dkIterator* iter = workLists->createIterator();
    while(iter->more()){
        DKWorkListICM* workList = (DKWorkListICM*) iter->next()->value();
        cout << " - " << workList->getName() << ": "
            << workList->getDescription() << endl;
        delete(workList); // Free Memory
    }
    delete(iter); // Free Memory
    delete(workLists);
}
```

Para obter um exemplo completo, consulte o exemplo SDocRoutingListingICM.

Definir um novo processo e encaminhamento associado

Um processo de encaminhamento de documentos consiste nos percursos definidos que um pacote de trabalhos segue. Vários processos de encaminhamento podem reutilizar os mesmos nós e o utilizador também pode utilizar vários percursos entre nós.

Java

```
// Create a new Process Definition
DKProcessICM process = new DKProcessICM();
process.setName("S_claimProcess");
process.setDescription("Process for an Insurance Claim");

// Define all possible Routes.

// Create a list of all possible routes between nodes.
dkCollection routes = new DKSequentialCollection();

// Connect the Work Nodes using Route List Entries. A simple route
//between two work nodes is specified by associating a 'From' work
//node and a 'To' work node.
// A Route List Entry simply connects two nodes with an implied direction.
// Multiple routes may exist between nodes. A specific route may be selected
// by a user-defined "selection" keyword. Examples might be "Continue", "Go",
// "Accept", "Reject", "Complete", etc.
// Create a new connection between two nodes.
DKRouteListEntryICM nodeRoute = new DKRouteListEntryICM();

// Every process must start with the start node.
nodeRoute.setFrom(DKConstantICM.DK_ICM_DR_START_NODE);
nodeRoute.setTo("S_fillClaim");
// Choose any user-defined name for an action that will make the
//transition from the 1st node to the second
nodeRoute.setSelection("Continue");

// Add the individual route to the collection of all possible routes.
routes.addElement(nodeRoute);
nodeRoute = new DKRouteListEntryICM();
nodeRoute.setFrom("S_fillClaim");
nodeRoute.setTo("S_gatherAll");
// Choose any user-defined name for an action that will make the
// transition take place.
nodeRoute.setSelection("Continue");

// Add the individual route to the collection of all possible routes.
routes.addElement(nodeRoute);
nodeRoute = new DKRouteListEntryICM();
nodeRoute.setFrom("S_gatherAll");
nodeRoute.setTo(DKConstantICM.DK_ICM_DR_END_NODE);

// Choose any user-defined name for an action that will make
//the transition take place.
nodeRoute.setSelection("Complete");
// Add the individual route to the collection of all possible routes.
routes.addElement(nodeRoute);
// Set the route in the process.
process.setRoute(routes);
// Adicione o processo à Gestão de encaminhamentos.
routingMgmt.add(process);
```

C++

```
//A document routing process is the defined routes that a work
//package being routed will follow. Multiple routing processes may
//re-use the same nodes and multiple routes between nodes may be used.

// Create a new Process Definition
DKProcessICM* process = new DKProcessICM();
process->setName("Buy_Book");
process->setDescription("Purchase a book online");
// Define all possible Routes.

// Create a list of all possible routes between nodes.
dkCollection* routes = new DKSequentialCollection();

// Connect the Work Nodes using Route List Entries. A simple route
// between two work nodes is specified by associating a 'From' work node
// and a 'To' work node.
// A Route List Entry simply connects two nodes with an implied direction.
// Multiple routes may exist between nodes. A specific route may be selected
// by a user-defined "selection" keyword. Examples might be "Continue", "Go",
// "Accept", "Reject", "Complete", etc.

// Create a new connection between two nodes.
DKRouteListEntryICM* nodeRoute = new DKRouteListEntryICM();

// Every process must start with the start node.
nodeRoute->setFrom(DK_ICM_DR_START_NODE);
nodeRoute->setTo("ValidateCreditCard");
// Choose any user-defined name for an action that will make the transition
// from the 1st node to the 2nd
nodeRoute->setSelection("Continue");

// Add the individual route to the collection of all possible routes.
routes->addElement(nodeRoute);

nodeRoute = new DKRouteListEntryICM();
nodeRoute->setFrom("ValidateCreditCard");
nodeRoute->setTo("GatherShippingDetails");

// Choose any user-defined name for an action that will make the transition
// take place.
nodeRoute->setSelection("Continue");

// Add the individual route to the collection of all possible routes.
routes->addElement(nodeRoute);

nodeRoute = new DKRouteListEntryICM();
nodeRoute->setFrom("GatherOrderDetails");
nodeRoute->setTo(DK_ICM_DR_END_NODE);

// Choose any user-defined name for an action that will make the transition
// take place.
nodeRoute->setSelection("Complete");

// Add the individual route to the collection of all possible routes.
routes->addElement(nodeRoute);

// Set the route in the process.
process->setRoute(routes);

// Add the process to the routing Management.
routingMgmt->add(process);

delete(process);
```

For the complete example, see the SDocRoutingDefinitionCreationICM sample.

Iniciar um processo de encaminhamento de documentos

O exemplo seguinte mostra como iniciar um processo de encaminhamento de documentos.

Java

```
//Primeiro crie um documento ou pasta que será encaminhado.  
//Deve ser pré-definido um tipo de artigo de nome "s_simple" antes de se poder  
// criar o DDO desse nome.  
  
DKDDO ddoFolder = dsICM.createDDO("S_simple", DKConstant.DK_CM_FOLDER);  
  
// Guarde a pasta criada no armazenamento de dados permanente.  
ddoFolder.add();  
  
//Crie o objecto base de serviço de encaminhamento de documentos.  
DKDocRoutingServiceICM routingService = new DKDocRoutingServiceICM(dsICM);  
  
//Inicie um processo com o nome "S_claimProcess" (que deve ser pré-definido).  
  
//A cadeia de PID do pacote de trabalho que será encaminhada é devolvida por  
//esta chamada.  
String workPackagePidStr = routingService.startProcess  
("S_claimProcess", ddoFolder.getPidObject().pidString(),1,"icmadmin");
```

C++

```
//Primeiro crie um documento ou pasta que será encaminhado.  
//An item type of name "book" must be pre-defined before a DDO of  
// that name can be created.  
DKDDO* ddoFolder = dsICM->createDDO("book", DK_CM_FOLDER);  
  
// Guarde a pasta criada no armazenamento de dados permanente.  
ddoFolder->add();  
  
//Set the priority for this document routing process  
int Priority = 1;  
  
//Crie o objecto base de serviço de encaminhamento de documentos.  
DKDocRoutingServiceICM* routingService = new DKDocRoutingServiceICM(dsICM);  
  
//Start a process with the name "Buy_Book" (which must be pre-defined.  
//A cadeia de PID do pacote de trabalho que será encaminhada é devolvida por  
//esta chamada.  
DKString workPackagePidStr = routingService->  
startProcess("Buy_Book", ((DKPidICM*)  
ddoFolder->getPidObject())->pidString(), Priority, "icmadmin");
```

Para obter um exemplo completo, consulte o exemplo SDocRoutingProcessingICM.

Terminar um processo

Um processo pode ser explicitamente terminado antes deste alcançar um nó final. Quando termina um processo, o pacote de trabalho a ser enviado é removido do sistema. Para terminar um processo, o utilizador deve saber qual a cadeia de PID do pacote de trabalho a ser enviado pela instância do processo

Java

```
routingService.terminateProcess(workPackagePidStr);
```

C++

```
routingService->terminateProcess(workPackagePidStr);
```

Para obter mais informações relativas a terminar um processo, consulte o exemplo `S DocRoutingProcessingICM`.

Continuar um processo

O método `continueProcess()` encaminha o artigo, referido pelo PID do artigo no pacote de trabalho especificado, do actual nó de trabalho para o nó de trabalho seguinte, que é determinado pela selecção. O pacote de trabalho especificado é removido do servidor de bibliotecas e é criado um novo pacote de trabalho para o proprietário especificado. É dada entrada do artigo referido pelo PID de artigo, caso tenha sido dada saída do mesmo. O PID do novo pacote de trabalho é devolvido. É devolvido um nulo caso o processo tenha terminado.

No exemplo de código que a seguir se apresenta, o nome da selecção que irá causar a transição do nó de trabalho actual para o nó de trabalho seguinte é "Continue". Tenha em atenção que a cadeia de PID do pacote de trabalho do actual pacote de trabalho é especificada na chamada do método. A chamada do método devolve a cadeia de PID do novo pacote de trabalho.

Java

```
workPackagePidStr = routingService.continueProcess  
(workPackagePidStr, "Continue", "icmadmin");
```

C++

```
char * userName = "icmadmin";  
  
workPackagePidStr = routingService->continueProcess(workPackagePidStr,  
"Continue", userName);
```

Para obter um exemplo completo, consulte o exemplo `S DocRoutingProcessingICM`.

Suspender um processo

Uma instância do processo de encaminhamento de documentos pode ser suspensa durante determinado período de tempo (em minutos) ou pode efectuar a pendência de um conjunto de requisitos que devem ser preenchidos antes que se possa retomar o processo. Isto não diz respeito a processos e módulos num ambiente de programação. O módulo ou processo no ambiente de tempo de execução de C++ não irá ser parado.

Java

```
dkCollection requirements = new DKSequentialCollection();
//O processo será suspenso durante dois minutos.
routingService.suspendProcess(workPackagePidStr, 2, requirements);
```

C++

```
dkCollection * requirements = new DKSequentialCollection();
//If no requirements are to be provided and the process is only to be
//suspended for a fixed period of time, the user can also pass in a
//NULL collection to this method.
//dkCollection * requirements = NULL;

//O processo será suspenso durante dois minutos.
routingService->suspendProcess(workPackagePidStr, 2, requirements);
delete(requirements);
```

Para obter um exemplo completo, consulte o exemplo SDocRoutingProcessingICM.

Retomar um processo

Um processo suspenso (processo em estado de suspensão) pode ser explicitamente ou implicitamente retomado após a expiração do tempo especificado ou após os requisitos definidos terem sido preenchidos. Esta acção anula o estado de suspensão do processo, devolvendo-o ao seu funcionamento normal. O método `resumeProcess` redefine o identificador suspenso do pacote de trabalho especificado como falso antes da suspensão atingir o tempo de duração especificado. Não irão ser executados quaisquer encaminhamentos ou retiradas do artigo de trabalho associado.

Java

```
routingService.resumeProcess(workPackagePidStr);
```

C++

```
routingService->resumeProcess(workPackagePidStr);
```

Para obter mais informações relativas à retoma de um processo, consulte o exemplo SDocRoutingProcessingICM.

Enumerar cadeias de identificador permanente de pacotes de trabalho numa lista de trabalhos

O exemplo de código seguinte mostra como enumerar as cadeias de PID de todos os pacotes de trabalho numa lista de trabalhos especificada.

Java

```
String[] workPackagePIDs =  
    routingService.listWorkPackagePidStrings(workListName,processOwner);  
  
// Imprima os PIDs do Pacote de Trabalho  
System.out.println("Work Packages in Work List: (+workPackagePIDs.length+));  
for(int i=0; i< workPackagePIDs.length; i++)  
    System.out.println(" - PID: +workPackagePIDs[i]);
```

C++

```
long arraySize = -1; // Tamanho a ser definido pela API.  
DKString* workPackagePIDs = routingService->  
    listWorkPackagePidStrings("workListName",processOwner,arraySize);  
  
// Imprima os PIDs do Pacote de Trabalho  
cout << "Work Packages in Work List: (" << arraySize << ")" << endl;  
for(int i=0; i< arraySize; i++)  
    cout << " - PID: " << workPackagePIDs[i] << endl;  
  
delete[] workPackagePIDs; // Liberte memória
```

Para obter um exemplo completo, consulte o exemplo `SDocRoutingListingICM`.

Obter informações relativas ao pacote de trabalho

Quando uma instância de um processo de encaminhamento de documentos se encontra a decorrer, um pacote de trabalho constitui o veículo através do qual um artigo (instância de um tipo de artigo) avança no processo de encaminhamento. Um pacote de trabalho contém todas as informações necessárias relativas ao processo e ao artigo que transporta. O pacote de trabalho corresponde ao objecto que uma aplicação utiliza e manipula como for necessário.

O método `retrieveWorkPackage` devolve o objecto de `DKWorkPackageICM` referido pelo PID do pacote de trabalho especificado (`wpPidStringStr`).

Java

```
//Utilize um serviço de encaminhamento de documentos estabelecido
//Ao especificar falso nesta chamada de método, o utilizador estará a garantir
//que não será dada saída
//do pacote de trabalho
DKWorkPackageICM workPackage =
    routingService.retrieveWorkPackage(workPackagePidStr,false);
System.out.println("-----");
System.out.println("                Work Package");
System.out.println("-----");
System.out.println(" Process Name: " + workPackage.getProcessName());
System.out.println(" work Node Name: " + workPackage.getWorkNodeName());
System.out.println(" Owner: " + workPackage.getOwner());
System.out.println(" Priority: " + workPackage.getPriority());
System.out.println(" User Last Moved: " + workPackage.getUserLastMoved());
System.out.println(" Time Last Moved: " + workPackage.getTimeLastMoved());
System.out.println(" Suspend State: " + workPackage.getSuspendState());
System.out.println(" Notify State: " + workPackage.getNotifyState());
System.out.println(" Notify Time: " + workPackage.getNotifyTime());
System.out.println(" Resume Time: " + workPackage.getResumeTime());
System.out.println("Work Package Pid: " + workPackage.getPidString());
System.out.println(" Item Pid: " + workPackage.getItemPidString());
```

C++

```
cout << "-----" << endl;
cout << " Work Package" << endl;
cout << "-----" << endl;
cout << " Process Name: " << workPackage->getProcessName() << endl;
cout << " work Node Name: " << workPackage->getWorkNodeName() << endl;
cout << " Owner: " << workPackage->getOwner() << endl;
cout << " Priority: " << workPackage->getPriority() << endl;
cout << " User Last Moved: " << workPackage->getUserLastMoved() << endl;
cout << " Time Last Moved: " << workPackage->getTimeLastMoved() << endl;
cout << " Suspend State: " << workPackage->getSuspendState() << endl;
cout << " Notify State: " << workPackage->getNotifyState() << endl;
cout << " Notify Time: " << workPackage->getNotifyTime() << endl;
cout << " Resume Time: " << workPackage->getResumeTime() << endl;
cout << "Work Package Pid: " << workPackage->getPidString() << endl;
cout << " Item Pid: " << workPackage->getItemPidString() << endl;
```

Para obter um exemplo completo, consulte o exemplo SDocRoutingProcessingICM.

Enumerar processos de encaminhamento de documentos

Os exemplos seguintes mostram como enumerar processos de encaminhamento de documentos.

Java

```
//The listProcessNames method lists all process names in the
//Library Server, and the listProcesses method returns a collection
//of DKProcessICM objects representing a process in the Library Server.
// Obtain the document routing management object.
DKDocRoutingServiceMgmtICM routingMgmt =
    new DKDocRoutingServiceMgmtICM(dsICM);
// Obtain the list of all running document routing processes
// Obtain the Routing Management object.
DKDocRoutingServiceMgmtICM routingMgmt =
    new DKDocRoutingServiceMgmtICM(dsICM);
// Obtain list of all routing processes running.
dkCollection processes = routingMgmt.listProcesses();
System.out.println("Running Processes: (" + processes.cardinality() + ")");

dkIterator iter = processes.createIterator();
while(iter.more())
{
    // Move pointer to next element and obtain that next element.
    DKProcessICM proc = (DKProcessICM) iter.next();
    System.out.println(" - " + proc.getName() + ": " + proc.getDescription());
}
```

C++

```
// Obtain the document routing management object.
DKDocRoutingServiceMgmtICM* routingMgmt =
    new DKDocRoutingServiceMgmtICM(dsICM);

// Obtain the list of all running document routing processes
dkCollection* processes = routingMgmt->listProcesses();
if (processes && (processes->cardinality() > 0))
{
    cout << "Running Processes: (" << processes->cardinality() << ")"
        << endl;
    dkIterator* iter = processes->createIterator();
    while(iter->more())
    {
        DKProcessICM* proc = (DKProcessICM*) iter->next()->value();
        cout << " - " << proc->getName() << ": "
            << proc->getDescription() << endl;
        delete(proc);
    }
    delete(iter);
    delete(processes);
}
delete(routingMgmt);
```

Uma função de impressão é facultada no exemplo SDocRoutingListingICM.

Encaminhamento ad hoc

De seguida é apresentado um procedimento exemplo de encaminhamento ad hoc. No exemplo, o cliente de administração de sistemas é utilizado para configurar nós de trabalho, processos e listas de trabalho.

1. Crie dois nós de trabalho, N1 e N2, por exemplo.
2. Crie dois processos de um nó, P1 e P2, de forma a que P1 possua um nó de trabalho, N1 e P2 tenham um nó de trabalho, N2.

P1 terá o seguinte aspecto:

| De: | Acção: | Para: |
|-------|-----------|-------|
| START | Continuar | N1 |
| N1 | Continuar | END |

P2 terá o seguinte aspecto:

| De: | Acção: | Para: |
|-------|-----------|-------|
| START | Continuar | N2 |
| N2 | Continuar | END |

3. Crie duas listas de trabalhos, WL1 e WL2, de forma a que WL1 possua um nó de trabalho, N1 e WL2 tenham um nó de trabalho, N2.

Na duração da execução, conclua os seguintes passos para implementar o encaminhamento ad hoc:

1. Inicie o processo P1 com um PID de documento (Exemplo, ABC). É criado um pacote de trabalho, o WP1. A lista de trabalhos WL1 apresenta o pacote de trabalhos WP1 no nó de trabalho N1.
2. Para mover o documento ABC do processo P1 para o processo P2, termine o pacote de trabalho WP1 e inicie o processo P2 com o mesmo documento (ABC). É criado o pacote de trabalho, o WP2.

A lista de trabalhos WL2 apresenta o pacote de trabalho WP2 no nó de trabalho N2.

Para visualizar exemplos adicionais, consulte o exemplo SDocRoutingDefinitionCreationICM.

Consultas exemplo de encaminhamento de documentos

Esta secção contém consultas exemplo. Para obter mais informações relativas a escrever em consultas, consulte "Compreender a linguagem da consulta" na página 185.

Exemplo 1

Localiza documentos de veículos e devolve apenas os pacotes de trabalho associados que estão activos.

```
/Car[@VERSIONID = latest-version(.) AND @SEMANTICTYPE =  
1]/REFERENCEDBY/@REFERENCER => WORKPACKAGE[@SUSPENDFLAG =  
0]
```

Exemplo 2

Localiza documentos de veículos e devolve os pacotes de trabalho associados que se encontram no processo "Investigação de Acidente".

```
/Car[@VERSIONID = latest-version(.) AND @SEMANTICTYPE =  
1]/REFERENCEDBY/@REFERENCER => WORKPACKAGE[@PROCESSITEMID =  
/ROUTINGPROCESS[@PROCESSNAME =  
"Investigação do Acidente"]/@ITEMID]
```

Exemplo 3

Localiza documentos de veículos em que o nome é "Honda" e devolve os pacotes de trabalho associados que se encontram no processo "Investigação de Acidente".

```
/Car[@Name = "Honda" AND @VERSIONID = latest-version(.) AND  
@SEMANTICTYPE = 1]/REFERENCEDBY/@REFERENCER =>  
WORKPACKAGE[@PROCESSITEMID = /ROUTINGPROCESS[@PROCESSNAME =  
"Investigação do Acidente"]/@ITEMID]
```

Exemplo 4

Localiza documentos de veículos e devolve os pacotes de trabalho associados que se encontram no passo "Em Análise" do processo "Investigação de Acidente".

```
/Car[@VERSIONID = latest-version(.) AND @SEMANTICTYPE =  
1]/REFERENCEDBY/@REFERENCER => WORKPACKAGE[@PROCESSITEMID =  
/ROUTINGPROCESS[@PROCESSNAME = "AccidentInvestigation"]/@ITEMID  
AND ../@WORKNODENAME = "Sob Avaliação"]
```

Exemplo 5

Localiza documentos de veículos e devolve apenas os pacotes de trabalho associados que se encontram no passo "Em Análise" do processo "Investigação de Acidente".

```
/Car[@VERSIONID = latest-version(.) AND @SEMANTICTYPE =  
1]/REFERENCEDBY/@REFERENCER => WORKPACKAGE[@PROCESSITEMID =  
/ROUTINGPROCESS[@PROCESSNAME = "Investigação do Acidente"]  
/@ITEMID AND ../@WORKNODENAME = "Sob Avaliação" AND  
@SUSPENDFLAG = 1]
```

Conceder privilégios para o encaminhamento de documentos

Para que um utilizador possa executar operações de encaminhamento de documentos, este deve possuir os privilégios adequados. Os privilégios associados ao encaminhamento de documentos encontram-se enumerados na tabela seguinte. Os privilégios gerais dos artigos são aplicáveis a processos, nós de trabalho e listas de trabalhos.

Tabela 16. Privilégios de encaminhamento de documentos

| Privilégio | Descrição | API relacionada |
|----------------------------|---|---|
| ICM_PRIV_ITEM_UPDATE_WORK | Utilizado para verificar se o utilizador possui autorização para executar as seguintes acções relacionadas com o pacote de trabalho: definir a prioridade definir o proprietário definir a lista de retoma definir a duração da suspensão | suspendProcess resumeProcess setWorkPackagePriority setWorkPackageOwner |
| ICM_PRIV_ITEM_ROUTE_START | Utilizado para verificar se o utilizador possui autorização para iniciar um processo. | startProcess |
| ICM_PRIV_ITEM_ROUTE_END | Utilizado para verificar se o utilizador possui autorização para terminar um processo. | terminateProcess |
| ICM_PRIV_ITEM_GET_WORKLIST | Utilizado para verificar se o utilizador possui autorização para obter a contagem de pacotes de trabalho de uma lista de trabalhos. | getCount listWorkPackagePidStrings |
| ICM_PRIV_ITEM_GET_WORK | Utilizado para verificar se o utilizador possui autorização para obter um pacote de trabalho. | getNextWorkPackagePidString getNextWorkPackage checkOutItemInWorkPackage retrieveWorkPackage |

Tabela 16. Privilégios de encaminhamento de documentos (continuação)

| Privilégio | Descrição | API relacionada |
|---------------------------------|--|--|
| ICM_PRIV_ITEM _GET_ASGN_WORK | Utilizado para verificar se o utilizador possui autorização para obter um pacote de trabalho possuído por um utilizador diferente. | getNextWorkPackagePidString getNextWorkPackage getCount listWorkPackagePidStrings |
| ICM_PRIV_ITEM _ROUTE | Utilizado para verificar se o utilizador possui autorização para encaminhar um pacote de trabalho. | continueProcess |

Trabalhar com listas de controlo de acesso para o encaminhamento de documentos

Ao definir uma ACL para uma entidade de encaminhamento de documentos como, por exemplo, um processo, nós de trabalho e lista de trabalhos, as operações permitidas na entidade são afectadas. O efeito das ACLs na entidade de encaminhamento de documento do Content Manager e os seus privilégios associados encontram-se enumerados na seguinte tabela.

Tabela 17. Listas de controlo de acesso e encaminhamento de documentos

| Privilégio | Descrição | API relacionada |
|--------------------|---|--|
| Processo | startProcess | ICM_PRIV_ITEM_ROUTE_START |
| Nó de trabalho | continueProcess suspendProcess resumeProcess terminateProcess setWorkPackagePriority setWorkPackageOwner | ICM_PRIV_ITEM_ROUTE ICM_PRIV_ITEM_UPDATE_WORK ICM_PRIV_ITEM_UPDATE_WORK ICM_PRIV_ITEM_ROUTE_END ICM_PRIV_ITEM_UPDATE_WORK ICM_PRIV_ITEM_UPDATE_WORK |
| Lista de trabalhos | getNextWorkPackagePidString getNextWorkPackage getCount listWorkPackagePidStrings checkOutItemInWorkPackage | ICM_PRIV_ITEM_GET_WORK ICM_PRIV_ITEM_GET_ASGN_WORK ICM_PRIV_ITEM_GET_WORK ICM_PRIV_ITEM_GET_ASGN_WORK ICM_PRIV_ITEM_GET_WORKLIST ICM_PRIV_ITEM_GET_ASGN_WORK ICM_PRIV_ITEM_GET_WORKLIST ICM_PRIV_ITEM_GET_ASGN_WORK ICM_PRIV_ITEM_GET_WORK |

Constantes de encaminhamento de documentos

O utilizador define as constantes de encaminhamento de documentos em DKConstantICM. De seguida é apresentada a lista de constantes de encaminhamento de documentos:

- public final static int DK_ICM_DR_SELECTION_FILTER_NO = 0;
- public final static in DK_ICM_DR_SELECTION_FILTER_YES = 1;
- public final static int DK_ICM_DR_SELECTION_FILTER_EITHER = 2;
- public final static int DK_ICM_DR_SELECTION_ORDER_PRIORITY = 0;
- public final static int DK_ICM_DR_SELECTION_ORDER_TIME = 1;
- public final static int DK_ICM_DR_MAX_RESULT_ALL = 0;
- public final static String DK_ICM_DR_START_NODE = "START";
- public final static String DK_ICM_DR_END_NODE = "END";

Trabalhar com outros servidores de conteúdos

As classes `dkDatastore` são utilizadas para definir um servidor de conteúdos adequado aos servidores de conteúdo da sua aplicação. O servidor de conteúdos é a interface primária do Enterprise Information Portal. Cada servidor de conteúdos possui uma classe de servidor de conteúdos diferente.

Para criar um servidor de conteúdos, utilize as classes de `DKDatastorexx`, em que `xx` identifica o servidor de conteúdos específico. A Tabela 7 na página 36 apresenta estas classes.

Tabela 18. Tipo de servidor e terminologia do nome da classe

| Servidor de conteúdos | Nome da classe |
|---|---|
| Content Manager Versão 8.2 | <code>DKDatastoreICM</code> |
| Content Manager anterior | <code>DKDatastoreDL</code> |
| Content Manager OnDemand | <code>DKDatastoreOD</code> |
| Content Manager for AS/400 (VisualInfo para AS/400) | <code>DKDatastoreV4</code> |
| Content Manager ImagePlus for OS/390 | <code>DKDatastoreIP</code> |
| Domino.Doc | <code>DKDatastoreDD</code> |
| Extended Search | <code>DKDatastoreDES</code> |
| Panagon Image Services (FileNET) | <code>DKDatastoreFN</code> |
| Bases de dados relacionais | <code>DKDatastoreDB2</code> , <code>DKDatastoreJDBC</code> (for Java), <code>DKDatastoreODBC</code> |

Ao criar um servidor de conteúdos para um servidor de conteúdos, implemente cada uma das seguintes classes e interfaces:

dkDatastore

Representa o servidor de conteúdos e gere a ligação, as comunicações e a execução de comandos do servidor de conteúdos. O `dkDatastore` é uma versão abstracta da classe do gestor de consultas. Suporta o método `avaliar`.

dkDatastoreDef

Utiliza os métodos para aceder a artigos armazenados no servidor de conteúdos. Também cria, enumera e elimina as suas entidades. Mantém uma recolha de `dkEntityDefs`. Os exemplos de classes concretas para esta interface são:

- `DKDatastoreDefDL`
- `DKDatastoreDefOD`

dkEntityDef

Utilize os métodos para aceder a informações da entidade. Também cria e elimina entidades e atributos. Os métodos desta classe suportam aceder a entidades de múltiplos níveis. Se um servidor de conteúdos não suportar sub-entidades, estas geram objectos de `DKUsageError`. Se um servidor de conteúdos suportar entidades de vários níveis, o utilizador deve

implementar métodos para sobrepor as excepções de sub-classes destes servidores de conteúdo. Os exemplos de classes concretas para a interface de dkEntityDef são:

- DKIndexClassDefDL
- DKAppGrpDefOD

A hierarquia de classes para uma definição de entidade está ilustrada em Figura 14:

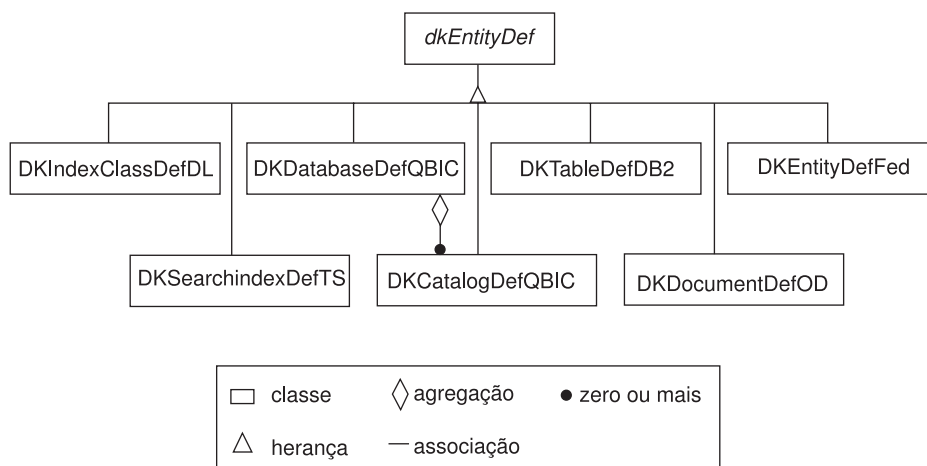


Figura 14. Hierarquia de classes

dkAttrDef

Define métodos para aceder a informações de atributos e para criar e eliminar atributos. Os exemplos de classes concretas de dkAttrDef são:

- DKAttributeDefDL
- DKFieldDefOD

dkServerDef

Define métodos de acesso a informações do servidor. Os exemplos de classes concretas de dkServerDef são:

- DKServerDefDL
- DKServerDefOD

dkResultSetCursor

Cria um cursor de servidor de conteúdos que gere uma recolha de objectos DDO. Para utilizar os métodos addObject, deleteObject e updateObject, define a opção do servidor de conteúdos DK_CM_OPT_ACCESS_MODE como DK_CM_READWRITE.

dkBlob

Declara uma interface pública comum para tipos de dados de objectos binários de grandes dimensões (BLOB) em cada servidor de conteúdos. As classes concretas derivadas de dkBlob partilham esta interface comum, permitindo um processamento de BLOBs com origem em servidores de conteúdos heterogéneos. Os exemplos de classes concretas de dkBlob são:

- DKBlobDL
- DKBlobOD

As classes de definição de dados e a sua hierarquia de classes estão representadas em Figura 15:

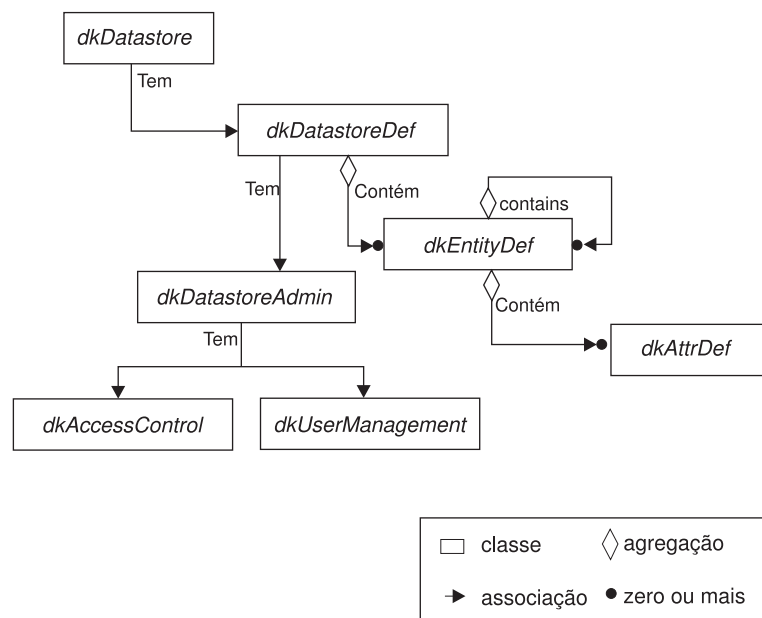


Figura 15. Hierarquia de classes de definição de dados

Para obter mais informações sobre dkDatastore e outras classes comuns, consulte “Desenvolver conectores de servidores de conteúdos personalizados” na página 378.

Trabalhar com o Content Manager anterior

Esta secção descreve como aceder a dados em servidores do Content Manager e como executar as tarefas seguintes:

- Manusear objectos grandes
- Utilizar DDOs.
- Utilizar XDOs num motor de pesquisa.
- Utilizar consulta combinada.
- Utilizar Motor de Pesquisa de Texto.
- Utilizar pesquisa de imagens QBIC).
- Utilizar fluxos de trabalho e cestos de trabalho.

Manusear objectos grandes

No conector do anterior Content Manager, o utilizador pode obter objectos grandes por partes utilizando a obtenção assíncrona. Para obter a aplicação exemplo, consulte TxdoAsyncRetDL no directório CMBROOT\dk\samples.

Definir o tamanho da pilha de memória de Java (apenas Java)

Java tem uma limitação para o tamanho inicial e máximo predefinido da pilha. O tamanho inicial predefinido da pilha é 1 048 576 e o tamanho máximo predefinido da pilha é 16 777 216 bytes. Se o seu programa da aplicação de Java tentar utilizar objectos maiores que o tamanho da pilha, o seu programa irá falhar durante a execução. Para aumentar o tamanho máximo da pilha na sua aplicação, utilize a opção -mx quando executar o programa da aplicação de Java. Por exemplo:

Java

```
java -mx40000000 yourApplication
```

Utilizar DDOs para representar o conteúdo do Content Manager anterior

Um DDO associado a DKDatastoreDL tem informações específicas para representar o modelo de documento do Enterprise Information Portal: documento, pasta, partes, artigo, ID de artigo, ordem e assim por diante. As secções seguintes descrevem como deve aceder a estas informações.

Propriedades do DDO

O tipo de um artigo, quer seja um documento ou uma pasta, é uma propriedade com o nome DK_CM_PROPERTY_ITEM_TYPE. Para obter o tipo de artigo do DDO, deverá chamar:

Java

```
DKDDO addo = new DKDDO(dsDL, pid);
Object obj = addo.getPropertyByName(DK_CM_PROPERTY_ITEM_TYPE);
if (obj != null) {
    short item_type = ((Short) obj).shortValue();
}
```

C++

```
DKAny any = cddo->getPropertyByName(DK_CM_PROPERTY_ITEM_TYPE);
if (!any.isNull()) {
    unsigned short item_type = (unsigned short) any;
}    ...           // execute uma acção
```

Após a chamada da propriedade, o item_type é igual ao DK_CM_DOCUMENT para um documento ou DK_CM_FOLDER para uma pasta. A instrução if assegura que a propriedade existe. Consulte “Adicionar propriedades a um DDO” na página 43 e “Obter o DKDDO e as propriedades de atributo” na página 46 para obter mais informações.

Identificador permanente (PID)

O PID contém informações importantes específicas do Enterprise Information Portal: o tipo de objecto indica a classe de índice à qual o DDO pertence; o PID contém o ID de artigo do artigo associado do servidor de conteúdos. Consulte “Criar um identificador permanente (PID)” na página 44.

Representar documentos

Um DDO que representa um documento tem a propriedade DK_CM_PROPERTY_ITEM_TYPE definida como DK_CM_DOCUMENT. O seu PID contém o nome da classe de índices como tipo de objecto. O ID do PID é o mesmo que o ID do artigo.

As partes existentes num documento são representadas como objectos DKPartsDL, que consistem em conjuntos de objectos binários grandes (BLOBs), em que cada uma delas é representada como um objecto DKBlobDL.

Um DDO de documento tem um atributo específico denominado DKPARTS, cujo valor é um objecto de DKParts.

Para alcançar cada uma das partes no documento, primeiro o utilizador terá de obter DKParts, para em seguida criar um iterador para iterar as partes. Se o documento não possuir partes, DKParts é nulo ou a cardinalidade de DKParts é zero.

Os documentos associados a uma consulta combinada (uma combinação entre uma consulta paramétrica e de texto) poderão ter um atributo transitório denominado DKRANK, cujo valor é um objecto que contém uma ordem inteira informatizada pelo Motor de Pesquisa de Texto.

Para obter mais informações sobre a criação e processamento de um objecto de DKParts, consulte “Criar, actualizar e eliminar documentos ou pastas”, “Obter um documento ou pasta” na página 257 e “Criar documentos e utilizar o atributo DKPARTS” na página 86.

Representar pastas

Um DDO que representa uma pasta tem uma propriedade DK_CM_PROPERTY_ITEM_TYPE igual a DK_CM_FOLDER. Tal como um DDO de documento, o seu PID contém o nome da classe de índices como tipo de objecto e o ID do artigo está no ID do PID.

Um objecto de DKFolder representa o índice remissivo dentro de uma pasta. Um objecto de DKFolder é uma recolha de DDOs. Cada DDO representa um artigo na pasta, quer seja um documento ou outra pasta. Um DDO de pastas tem um atributo denominado DKFOLDER, cujo valor é um objecto de DKFolder.

Para alcançar cada um dos membros DDO da pasta, terá primeiro de obter o objecto de DKFolder; em seguida irá criar um iterador para aceder a cada membro do artigo. Se a pasta não possuir um membro, DKFolder é nulo, mas o atributo DKFOLDER está sempre presente num DDO de pasta criado pelo servidor de conteúdos.

Para obter mais informações sobre a criação e processamento de um objecto de DKFolder, consulte “Criar, actualizar e eliminar documentos ou pastas”, “Obter um documento ou pasta” na página 257 e “Criar pastas e utilizar o atributo DKFOLDER” na página 89.

Criar, actualizar e eliminar documentos ou pastas

Esta secção descreve os processos que envolvem a criação, actualização e eliminação de documentos e pastas.

Criar um documento

Para criar um documento e guardar os seus dados permanentes num servidor de conteúdos, o utilizador deve criar um DDO e definir todos os seus atributos (e outras informações), exceptuando o ID do artigo. O ID do artigo é atribuído e devolvido pelo servidor de conteúdos. Alguns dos exemplos anteriores estão combinados na demonstração seguinte:

Java

```
// ----- Step 1: create a datastore and connect to it
DKDatastoreDL dsDL = new DKDatastoreDL();
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");

// ----- Step 2: create a document (or folder) DDO
//         and set all its attributes and other required information
DKPid pid = new DKPid();
pid.setObjectType("GRANDPA"); // Set the index-class name it belongs to
DKDDO ddo = new DKDDO(dsDL,pid); // Create a DDO with PID and
...                               // associate it to dsDL

// ----- Step 2.a: add attributes according to index class GRANDPA
Object obj, vstr;
Boolean yes = new Boolean(true);
Boolean no = new Boolean(false);

short data_id = cddo.addData("Title"); // add new attribute "Title"
vstr = new Short(DK_CM_DATAITEM_TYPE_STRING);
// ----- Add type properties VSTRING and nullable
cddo.addDataProperty(data_id, DK_CM_PROPERTY_TYPE, vstr);
cddo.addDataProperty(data_id, DK_CM_PROPERTY_NULLABLE, no);

data_id = cddo.addData("Subject"); // add new attribute "Subject"
cddo.addDataProperty(data_id, DK_CM_PROPERTY_TYPE, vstr);
cddo.addDataProperty(data_id, DK_CM_PROPERTY_NULLABLE, yes);

// ----- Add some more attributes as necessary
....

// ----- Step 2.b: add DKPARTS attribute
DKParts parts = new DKParts(); // create a new DKParts, collection of parts
DKBlobDL blob = new DKBlobDL(dsDL); // create a new XDO blob
DKPidXDODL pidXDO = new DKPidXDODL(); // create PID for this XDO object

pidXDO.setPartId(5); // set part number to 5
blob.setPidObject(pidXDO); // set the PID for the XDO blob
blob.setContentClass(DK_DL_CC_GIF); // set content class type GIF
blob.setRepType(DK_REP_NULL); // set rep type for the part
blob.setContentFromClientFile("choice.gif"); // set the blob's content
blob.setInstanceOpenHandler("xv"); // the viewer program on AIX

parts.addElement(blob); // add the blob to the parts collection

.... // create and add some more blobs to
.... // to the collection as necessary

// ----- Create DKPARTS attribute and set it to refer to the DKParts object
short data_id = ddo.addData(DKPARTS); // add attribute "DKParts"
obj = new Short(DK_CM_COLLECTION_XDO); // add type property
ddo.addDataProperty(data_id, DK_CM_PROPERTY_TYPE, obj);
ddo.addDataProperty(data_id, DK_CM_PROPERTY_NULLABLE, yes); // add nullable prop
ddo.setData(data_id, parts); // sets the attribute value

// ----- Step 2.c: sets the item type : document
obj = new Short(DK_CM_DOCUMENT);
ddo.addProperty(DK_CM_PROPERTY_ITEM_TYPE, obj);

// ----- Step 3: make item persistent; add item to the datastore
ddo.add(); // document created in datastore
```

C++

```
// step 1: create a datastore and connect to it
DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");

// step 2: create a document (or folder) DDO
//          and set all its attributes and other required information
//          See the section on "Using DDO"
DKPid pid;
// set the index-class name it belongs to
pid.setObjectType("GRANDPA");
// create a DDO with PID and associated with dsDL
DKDDO* ddo = new DKDDO(&dsDL,pid);

// step 2.a: add attributes according to index-class GRANDPA
DKAny any;
DKBoolean yes = TRUE;
DKBoolean no = FALSE;
// add a new attribute named "Title"
unsigned short data_id = ddo->addData("Title");
// add type property VSTRING
any = DK_VSTRING;
ddo->addDataProperty(data_id, DK_CM_PROPERTY_TYPE, any);
// add nullable property: non-nullable
any = no;
ddo->addDataProperty(data_id, DK_CM_PROPERTY_NULLABLE , any);

// add a new attribute named "Subject"
data_id = ddo->addData("Subject");

any = DK_VSTRING;
ddo->addDataProperty(data_id, DK_CM_PROPERTY_TYPE, any);
any = yes;
ddo->addDataProperty(data_id, DK_CM_PROPERTY_NULLABLE , any);

// add some more attributes as necessary
// ...

// step 2.b: add DKPARTS attribute
// create a new XDO blob
DKParts* parts = new DKParts;
DKBlobDL* blob = new DKBlobDL(&dsDL);

DKPidXDODL pidXDO;                                // create PID for this XDO object

pidXDO.setPartId(5);                                // set part number to 5
blob->setPid(&pidXDO);                                // set the PID for the XDO blob
blob->setContentClass(DK_CC_GIF);                     // set content class type GIF
blob->setRepType(DK_REP_NULL);                         // set rep type for the part
blob->setContentFromClientFile("choice.gif"); // set the blob's content

DKAny any = (dkDataObjectBase*) blob;
parts->addElement(any);                                // add the blob to the parts collection

...                                                    // create and add some more blobs
...                                                    // to the collection as necessary
// continued...
```

C++ (continuação)

```
// create DKPARTS attribute and sets it to refer to the DKParts object
// add attribute "DKParts"
unsigned short data_id = ddo->addData(DKPARTS);
any = DK_COLLECTION_XDO;
// add type property
ddo->addDataProperty(data_id,DK_CM_PROPERTY_TYPE,any);
any = (DKBoolean) TRUE;
// add nullable property
ddo->addDataProperty(data_id,DK_CM_PROPERTY_NULLABLE ,any);
any = (dkCollection*) parts;
// sets the attribute value
ddo->setData(data_id, any);

// step 2.c: sets the item type : document
any = DK_CM_DOCUMENT;
ddo->addProperty(DK_CM_PROPERTY_ITEM_TYPE, any);

// passo 3: torná-lo permanente
// a document is created in the datastore
ddo->add();
```

O último passo do exemplo anterior criou um documento no servidor de conteúdos (com as informações). Sempre que um DDO de documento é adicionado a um servidor de conteúdos, todos os seus atributos são adicionados, incluindo todas as partes que fazem parte da recolha de DKParts.

Para adicionar um DDO de pasta o processo é o mesmo. Os membros da recolha de DKFOLDER são adicionados ao servidor de conteúdos como um componente da pasta. A pasta contém uma tabela de conteúdos dos seus membros, que são documentos e pastas existentes. Por este facto, crie todos os membros de pasta no servidor de conteúdos antes de adicionar um DDO de pasta.

Java

Pode adicionar o mesmo documento a servidores de conteúdos do mesmo tipo. Para adicionar este documento a LIBSRVRN do servidor do Content Manager, que tem uma classe de índices LIBSV2 com a mesma estrutura que LIBSV, utilize o exemplo seguinte:

```
// ----- Create datastore and connect to LIBSRVRN
DKDatastoreDL dsN = new DKDatastoreDL();
dsN.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");

// ----- Update the PID
pid = ddo.getPidObject();
pid.setObjectType("LIBSV2");           // set the new index class
pid.setPrimaryId("");                  // make the item ID blank
pid.setDatastoreName("LIBSRVRN");      // set the new datastore name
ddo.setPidObject(pid);                 // update the PID
ddo.setDatastore(dsN);                 // re-associate the DDO with dsN
ddo.add();                             // add the DDO
```

C++

Pode adicionar o mesmo documento a servidores de conteúdos do mesmo tipo. Por exemplo, para adicionar o documento ao servidor LIBSRVRN, que tenha uma classe de índices GRANDPA2 com a mesma estrutura que GRANDPA:

```
// create datastore and connect to LIBSRVRN
DKDatastoreDL dsN;
dsN.connect("LIBSRVRN","FRNADMIN","PASSWORD");
// update the Pid
pid = ddo->getPid();
pid.setObjectType("GRANDPA2");           // set the new index-class
pid.setId("");                           // blank the item-id
pid.setDatastoreName("LIBSRVRN");        // set the new datastore name
ddo->setPid(pid);                         // update the PID
ddo->setDatastore(&dsN);                  // re-associate it with dsN
ddo->add();                               // add it
```

Actualizar um documento ou uma pasta

Para actualizar um documento ou pasta:

1. Defina o ID de artigo e o tipo de objecto.
2. Actualize os atributos adequados ou adicione-os à recolha de DKParts.
3. Chame o método actualizar para guardar a alteração.

Java

```
// ----- Update the value of attribute Title
String newTitle = "Accident Report";
short data_id = ddo.getDataByName("Title");
ddo.setData(data_id, newTitle);
ddo.update();
```

C++

```
// update the value of attribute Title
DKAny any = DKString("Guess who is behind all this");
unsigned short data_id = ddo->getDataByName("Title");
ddo->setData(data_id, any);
ddo->update();
```

Após efectuar a chamada ao método de actualização, o valor do atributo Title no servidor de conteúdos é actualizado. As partes neste documento não são actualizadas excepto se o seu conteúdo for alterado. A ligação ao servidor tem de ser válida quando chamar o método actualizar.

Actualize um DDO de pastas através de passos semelhantes: actualize os valores do atributo ou adicione ou remova elementos de DKFolder; em seguida chame o método actualizar.

Actualizar partes

Represente a recolha de partes num documento através de um objecto de DKParts.

DKParts é uma subclasse de DKSequentialCollection. Para além de herdar as funções de recolha sequencial, o DKParts tem dois membros adicionais para adicionar e remover uma parte da recolha. Estes métodos também guardam imediatamente as alterações ao servidor de conteúdos.

O documento já deve existir no servidor de conteúdos.

Adicionar ou remover um membro: Os exemplos seguintes adicionam uma parte ao documento.

Java

```
DKDDO addo = new DKDDO();    // create a document DDO
DKBlobDL newPart = new DKBlobDL(); // create the new part to be added
....                          // initialized the DDO and new part
DKParts parts = (DKParts) addo.getDataByName(DKPARTS); // get DKParts
parts.addMember(ddo, newPart); // assume none of these values are NULL
```

C++

```
// a document DDO
DKDDO* ddo;
// a new part to be added
DKBlobDL* newPart;
// ddo and newPart are
// initialized somewhere along the line
...
...
// get DKParts
DKAny any = ddo->getDataByName(DKPARTS);
DKParts* parts = (DKParts*) any.value();
// assume none of these values are NULL
parts->addMember(ddo, newPart);
```

Para remover newPart da recolha e do servidor de conteúdos, deve utilizar:

Java

```
parts.removeMember(addo, newPart);
```

C++

```
parts->removeMember(ddo, newPart);
```

O método removeMember em DKParts elimina a cópia permanente da parte do servidor de conteúdos.

Diferenças entre actualizar, adicionar e remover num DDO de documentos: Os métodos addMember e removeMember de DKParts facultam utilitários para adicionar e remover uma parte da recolha e do servidor de conteúdos. São mais rápidos do que o método actualizar num DDO de documento. O método actualizar num DDO actualiza todos os atributos no DDO, incluindo DKParts de todos os seus membros alterados. As etapas são:

Java

```
....  
// ----- Obter DKParts, presumir que existe e não é nulo  
DKParts parts = (DKParts) addo.getDataByName(DKPARTS);  
parts.addElement(newPart);           // adicionar uma nova partes a partes  
addo.update();                       // actualiza todo o ddo  
....
```

C++

```
...  
DKAny any = ddo->getDataByName(DKPARTS);  
// obter DKParts, presumir que existe  
DKParts* parts = (DKParts*) any.value();  
// presumir que não é NULL  
any = (dkDataObjectBase*) newpart;  
parts->addElement(any);  
// actualiza todo o ddo  
ddo->update();  
...
```

Actualizar pastas

O utilizador irá representar a recolha de documentos e pastas dentro de uma pasta através de um objecto de DKFolder. No servidor de conteúdos, uma pasta contém uma tabela de conteúdos que fazem referência aos seus objectos, em vez de deter os objectos em si.

DKFolder é uma subclasse de DKSequentialCollection. Para além de herdar os métodos de recolha sequencial, tem dois membros adicionais de forma a adicionar ou remover um membro (um documento ou pasta) da recolha e armazena imediatamente essas alterações.

O documento ou pasta a ser adicionado ou removido já deve existir no servidor de conteúdos.

Adicionar ou remover um membro: O exemplo seguinte ilustra a adição de outro DDO de documento ou pasta a um DDO de pasta:

Java

```
DKDDO folderDDO = new DKDDO(); // Created the folder DDO  
DKDDO newMember = new DKDDO(); // Create the new DDO to be added  
....                          // The folder DDO and newMember are  
....                          // initialized  
// ----- Get the DKFolder, assuming it exists, and the value not null  
DKFolder folder = (DKFolder) folderDDO.getDataByName(DKFOLDER);  
folder.addMember(folderDDO, newMember);
```

C++

```
// a folder DDO
DKDDO* folderDDO;
// a new DDO to be added as a member of this folder
DKDDO* newMember;
...      // folderDDO and newMember are
...      // initialized somewhere along the line
DKAny any = folderDDO->getDataByName(DKFOLDER);
// get DKFolder, assume it exists
DKFolder* folder = (DKFolder*) any.value();
// assume non NULL
folder->addMember(folderDDO, newMember);
```

newMember e folderDDO devem existir no servidor de conteúdos para que outro documento ou pasta lhes seja adicionada.

De forma idêntica, para remover newMember da recolha e do servidor de conteúdos, utilize o seguinte exemplo:

Java

```
folder.removeMember(folderDDO, newMember);
```

C++

```
folder->removeMember(folderDDO, newMember);
```

Importante: Ao remover um membro de uma pasta apenas irá remover esse membro do índice da pasta. Caso utilize a função `removeElementAt`, esta não irá eliminar o membro da memória ou do servidor de conteúdos.

Diferenças entre actualizar, adicionar e remover num DDO de pasta: Os métodos `addMember` e `removeMember` de `DKFolder` facultam utilitários para adicionar e remover um documento ou uma pasta da recolha e do servidor de conteúdos. São mais rápidos do que o método actualizar num DDO de pasta

O método actualizar num DDO actualiza todos os atributos no DDO, incluindo `DKFolder` e todos os seus membros, enquanto que os métodos `addMember` e `removeMember` apenas adicionam ou removem um membro no índice remissivo da pasta.

Eliminar um documento ou uma pasta

Utilize o método `elim` no DDO para eliminar um documento ou pasta do servidor de conteúdos.

Java

```
ddo.del();
```


C++

```
ddo->del();
```

O DDO tem de possuir o seu ID do artigo e conjunto do tipo de objectos e deve possuir uma ligação válida ao servidor de conteúdos.

Utilize a instrução acima também para eliminar uma pasta. Apenas os dados permanentes são eliminados; a cópia em memória do DDO não é alterada. Consequentemente, pode voltar a adicionar posteriormente este DDO ao mesmo ou a outro servidor de conteúdos na aplicação. Consulte “Criar um documento” na página 249 para obter mais informações.

Obter um documento ou pasta

Para obter um documento deDKDatastoreDL (representando um servidor de conteúdos de versões anteriores do Content Manager), o utilizador deve saber qual o nome de classe de índice e o ID de artigo do documento. Deve também associar o DDO ao servidor de conteúdos e estabelecer uma ligação.

Java

```
DKDDO ddo = new DKDDO(dsDL,pid);
// ----- Create the datastore and establish a connection
DKDatastoreDL dsDL = new DKDatastoreDL();
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");

DKPid pid = new DKPid();
pid.setObjectType("Claim"); // set the index-class name it belongs to
pid.setPrimaryId("LN#U5K6ARLGM3DB4"); // set the item-id
// ----- create a DDO with the PID and associated with the datastore

ddo.retrieve(); // retrieve the document
```

C++

```
DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");
DKPid pid;
// set the index-class name it belongs to
pid.setObjectType("GRANDPA");
// set the item-id
pid.setId("LN#U5K6ARLGM3DB4");
// create a DDO with pid and associated to dsDL
DKDDO* ddo = new DKDDO(&dsDL, pid);
// retrieve it
ddo->retrieve();
```

Após efectuar uma chamada para obter, todos os valores de atributo do DDO são definidos como o valor dos dados permanentes armazenados no servidor de conteúdos. Caso o documento tenha partes, o atributo DKPARTS é definido como um objecto de DKParts. Contudo, não é obtido o conteúdo de cada parte nesta recolha. Devido ao facto de uma parte poder ser grande, o utilizador não deve obter todas ao mesmo tempo para a memória. É mais aconselhável obter de forma explícita a parte que pretende.

Caso o DDO seja um resultado de consulta paramétrica que foi executado com a opção de consulta CONTENT=NO, o DDO está vazio (não possui os valores de atributo). No entanto, todas as informações necessárias para o obter já estão definidas.

Obter partes

Após obter um DDO, pode obter as suas partes armazenadas no atributo DKPARTS, da seguinte forma:

Java

```
DKParts parts = (DKParts) ddo.getDataByName(DKPARTS);
```

C++

```
DKAny any = ddo->getDataByName(DKPARTS);  
DKParts* parts = (DKParts*) any.value();
```

Este exemplo parte do princípio que o atributo DKPARTS existe. Caso não exista, é gerada uma exceção. Consulte “Obter uma pasta” na página 259 para obter um exemplo da extracção do valor do atributo, obtendo primeiro o ID dos dados e testá-lo com o valor zero.

Para obter cada parte, terá de criar um iterador para avançar na recolha e obter cada parte. Consulte “Criar documentos e utilizar o atributo DKPARTS” na página 86.

Java

```
// ----- Create an iterator and process the part collection members  
if (parts != null) {  
    DKBlobDL blob;  
    dkIterator iter = parts.createIterator();  
    while (iter.more()) {  
        blob = (DKBlobDL) iter.next();  
        if (blob != null) {  
            blob.retrieve(); // retrieve the blob's content  
            blob.open();  
            .... // other processing, as needed  
        }  
    }  
}
```

C++

```
// create iterator and process the part collection member one by one
if (parts != NULL) {
    DKAny* element;
    DKBlobDL* blob;
    dkIterator* iter = parts->createIterator();
    while (iter->more()) {
        element = iter->next();
        blob = (DKBlobDL*) element->value();
        if (blob != NULL) {
            // retrieve the blob's content
            blob->retrieve();
            // other processing, as needed
            blob->open();
        }
    }
    delete iter;
}
```

Tal como nos resultados do DDO numa consulta paramétrica, cada XDO da parte que existe dentro da recolha de DKParts está vazio (não tem o conteúdo definido). Contudo, tem todas as informações necessárias para a obtenção. Para colocar o seu conteúdo e as informações relacionadas na memória, efectue a chamada do método de obtenção:

Java

```
blob.retrieve();
```

C++

```
blob->retrieve();
```

Obter uma pasta

Obtenha um DDO de pasta da mesma forma que obtém um DDO de documento. Após ser obtido, todos os atributos do DDO de pasta ficam definidos, incluindo o atributo, DKFOLDER. Este valor de atributo é definido para um objecto de DKFolder, uma recolha dos membros de DDO na pasta. Tal como as partes num objecto de DKParts, estes DDOs membros contêm apenas informação suficiente para as obter. Poderá obter um DDO de pastas da seguinte forma:

Java

```
data_id = ddo.dataId(DKFOLDER);    // get DKFOLDER data-id
if (data_id == 0)                  // folder not found
    throw new DKException(" folder data-item not found");

DKFolder fCol = (DKFolder) ddo.getData(data_id); // get the folder collection

// ----- Create iterator and process the DDO collection members one by one
if (fCol != null) {
    DKDDO item;
    dkIterator iter = fCol.createIterator();
    while (iter.more()) {
        item = (DKDDO) iter.next();
        if (item != null) {
            item.retrieve();    // retrieve the member DDO
            ....                // other processing
        }
    }
}
```

C++

```
// get DKFOLDER data-id
data_id = ddo->dataId(DKFOLDER);
// folder not found
if (data_id == 0) {
    DKException exc(" folder data-item not found");
    DKTHROW exc;
}
// get the folder collection
any = ddo->getData(data_id);
DKFolder* fCol = (DKFolder*) any.value();
// create iterator and process the DDO collection member one by one
if (fCol != NULL) {
    DKAny* element;
    DKDDO* item;
    dkIterator* iter = fCol->createIterator();
    while (iter->more()) {
        element = iter->next();
        item = (DKDDO*) element->value();
        if (item != NULL) {
            // retrieve the member DDO
            item->retrieve();
            // other processing
            ...
        }
    }
    delete iter;
}
```

Consulte também “Criar pastas e utilizar o atributo DKFOLDER” na página 89.

Compreender pesquisa de texto (Motor de Pesquisa de Texto)

O produto Motor de Pesquisa de Texto suporta vários tipos de consulta:

- “Consulta booleana” na página 261
- “Consulta de texto livre” na página 261
- “Consulta híbrida” na página 262
- “Consulta de proximidade” na página 262

- “Consulta de obtenção global de texto (GTR)” na página 262

Pode utilizar o ID de artigo, o part number e as informações de ordenação (devolvidas pela consulta) de pesquisa de texto para criar um XDO que obtenha o documento de um servidor de versões anteriores do Content Manager.

Utilize um objecto de DKDatastoreTS para representar o Motor de Pesquisa de Texto. O Motor de Pesquisa de Texto não armazena realmente os dados, mas sim limita-se a indexar os dados armazenados no Content Manager anterior para suportar nestes um pesquisa de texto. O resultado de um pesquisa de texto é um identificador de artigos a descrever a localização do documento no Content Manager. Utilize estes identificadores para obter o documento.

O objecto de DKDatastoreTS não suporta as funções adicionar, actualizar, obter e eliminar. Poderá consultar este servidor de conteúdos. Consulte o “Carregar dados para indexação através do Motor de Pesquisa de Texto” na página 272 para obter informações sobre a adição de dados ao Content Manager que está indexado pelo Motor de Pesquisa de Texto.

Consulta booleana

Uma consulta booleana é constituída por palavras e frases, separadas por operadores booleanos. Coloque uma frase entre apóstrofes ('). As frases são encaradas como cadeias literais.

O exemplo seguinte cria uma cadeia de consultas para pesquisar todos os documentos de texto com a palavra *www* ou a expressão *web site* no índice *TMINDEX* de pesquisa de texto:

Java

```
String cmd = "SEARCH=(COND=(www OR 'web site'));" +
             "OPTION=(SEARCH_INDEX=TMINDEX)";
```

C++

```
DKString cmd = "SEARCH=(COND=(www OR 'web site'))";
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Consulta de texto livre

Uma consulta de texto livre é constituída por palavras, frases ou expressões entre chavetas ({}). As palavras não têm de ser contíguas. O exemplo seguinte cria uma cadeia de consultas para pesquisar todos os documentos de texto com o texto livre *web site* no índice *TMINDEX* de pesquisa de texto:

Java

```
String cmd = "SEARCH=(COND=({web site}));" +
             "OPTION=(SEARCH_INDEX=TMINDEX)";
```

C++

```
DKString cmd = "SEARCH=(COND=({Web site}))";
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Consulta híbrida

Uma consulta híbrida é constituída por uma consulta booleana seguida de uma consulta de texto livre. O exemplo seguinte cria uma cadeia de consultas para pesquisar em todos os documentos de texto as palavras IBM e UNIX, bem como o texto livre web site no índice TMINDEX de pesquisa de texto:

Java

```
String cmd = "SEARCH=(COND=(IBM AND UNIX {web site}));" +  
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

C++

```
DKString cmd = "SEARCH=(COND=(IBM AND UNIX {Web site}));";  
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Consulta de proximidade

Uma consulta de proximidade procura uma sequência de argumentos de pesquisa localizados no mesmo documento, parágrafo ou frase. O exemplo seguinte cria uma cadeia de consultas para pesquisar todos os documentos de texto com a frase números racionais e a palavra matemática no mesmo parágrafo, utilizando o índice TMINDEX de pesquisa de texto:

Java

```
String cmd = "SEARCH=(COND=($PARA$ {'rational numbers' math}));" +  
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

C++

```
DKString cmd = "SEARCH=(COND=($PARA$ {'rational numbers' math}));";  
cmd += "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Este tipo de consulta exige pelo menos dois argumentos de pesquisa.

Consulta de obtenção global de texto (GTR)

Uma consulta de GTR é otimizada para idiomas de conjunto de caracteres de duplo byte (DBCS) como, por exemplo, o Japonês ou o Chinês. A GTR também suporta idiomas de conjunto de caracteres de byte único (SBCS). Coloque todos os caracteres de duplo byte entre plicas ('). Certifique-se de que a expressão a ser pesquisada possui o conjunto de códigos de caracteres e o idioma especificados.

O exemplo seguinte demonstra uma pesquisa GTR de todos os documentos de texto que contenham a frase comercialização da IBM. A palavra-chave CORRESPONDENTE é definida para indicar o grau de semelhança para a frase.

Java

```
String cmd = "SEARCH=(COND=($CCSID=850,LANG=6011,MATCH=1$ " +  
            "'comercialização da IBM'));" +  
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

C++

```
DKString cmd = "SEARCH=(COND=($CCSID=850, LANG=6011,MATCH=1$ ";  
cmd += "'IBM marketing')");  
cmd += "OPTION=(SEARCH_INDEX=TMGTRX)";
```

Certifique-se de que as opções do servidor de conteúdos de pesquisa de texto DK_OPT_TS_CCSID (identificadores de conjunto de caracteres codificados) e DK_OPT_TS_LANG (identificadores de idioma) estão correctamente definidos. A predefinição para DK_OPT_TS_CCSID é DK_CCSID_00850. A predefinição para DK_OPT_TS_LANG é DK_LANG_ENU. Estes valores são utilizados como predefinições globais para a consulta de texto. Para obter mais informações, consulte a referência de API online.

Também pode inserir informações específicas de CCSID e LANG como é demonstrado no exemplo seguinte. Terá de especificar CCSID e LANG; um valor não pode ser especificado sem o outro.

Representar informações de Motor de Pesquisa de Texto através de DDOs

Um DDO é utilizado (em associação com um objecto de DKDatastoreTS) para representar os resultados de pesquisas de texto.

O DKDatastoreTS não tem um tipo de artigo de propriedade como tem um objecto de DKDatastoreDL. O formato do seu ID também é diferente. Um DDO resultante de uma consulta de texto corresponde a uma parte de texto dentro de um artigo. Contém os seguintes atributos padrão:

DKDLITEMID

O ID do artigo de que faz parte este texto. Utilize este ID de artigo para obter todo o artigo do servidor de conteúdos.

DKPARTNO

Um part number inteiro para esta parte de texto. Utilize o part number com o ID do artigo para obter a parte de texto do servidor de conteúdos.

DKREPTYPE

O TipoRep desta parte de texto. Utilize este atributo com o ID do artigo e o part number para obter a parte de texto do servidor de conteúdos.

DKRANK

Uma ordem de inteiro que demonstra a relevância desta parte para os resultados de uma consulta de texto. Uma ordem mais elevada significa uma correspondência mais exacta. Consulte a referência de API online para obter mais informações.

DKDSIZE

Um número inteiro que representa as ocorrências de palavras (nos resultados de consultas booleanas). Consulte a referência de API online para obter mais informações.

DKRCNT

Um número inteiro que representa correspondências da pesquisa booleana. Consulte a referência de API online para obter mais informações.

O PID para um DDO de pesquisa de texto tem as informações seguintes:

tipo de servidor de conteúdos
TS.

nome do servidor de conteúdos
O nome utilizado para ligar ao servidor de conteúdos.

tipo de objectos
Índice de Pesquisa de texto.

ID ID de documento de Motor de Pesquisa de Texto.

Estabelecer uma ligação

O objecto de DKDatastoreTS fornece duas funções para ligação e um método para desligação. Normalmente, cria-se um objecto de DKDatastoreTS, estabelece-se-lhe uma ligação, executa-se uma consulta e, de seguida, anula-se a ligação após concluir. O exemplo seguinte demonstra a primeira função de ligação através do servidor de pesquisa de texto TM.

Java

```
// ----- Create the datastore
DKDatastoreTS dsTS = new DKDatastoreTS();
dsTS.connect("TM", "", "", "");
.... // run a query
dsTS.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TConnectTS.java) está disponível no directórioCMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
dsTS.connect("TM","", "", "");
... // do some work
dsTS.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TConnectTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

O exemplo seguinte demonstra a segunda função de ligação através do servidor de pesquisa de texto com o nome de sistema central apollo, número da porta 7502e tipo de comunicação TCP/IP DK_CTYP_TCPIP:

```
dsTS.connect("apollo", "7502", DK_CTYP_TCPIP);
```

O exemplo seguinte demonstra a primeira função de ligação através do servidor de pesquisa de texto com nome de sistema central apollo, número da porta 7502, tipo de comunicação T (TCP/IP):

```
dsTS.connect("apollo", "", "", "PORT=7502; COMMTYPE=T");
```

O exemplo seguinte demonstra o primeiro método de ligação através de TM do nome do servidor de pesquisa de texto e através do servidor de bibliotecas LIBSRVR2 , ID de utilizador FRNADMIN e palavra-passe PASSWORD:

O exemplo seguinte demonstra a primeira função de ligação através do nome do servidor de pesquisa de texto TM, do servidor de bibliotecas LIBSRVRN, do ID de utilizador FRNADMIN e da palavra-passe PASSWORD:

```
dsTS.connect("TM", "", "", "LIBACCESS=(LIBSRVRN, FRNADMIN, PASSWORD)");
```

Pode utilizar o último parâmetro LIBACCESS, também designado por cadeia de ligação, para transmitir uma sequência de parâmetros.

Sugestão: Para evitar que expire o tempo da ligação de Motor de Pesquisa de Texto, ligue-se ao Motor de Pesquisa de Texto, execute as consultas e desligue imediatamente. Não deixe a ligação aberta.

Obter e definir opções de pesquisa de texto

O Pesquisa de texto fornece algumas opções que poderá definir ou obter através das suas funções. Consulte a referência de API online para obter a lista de opções e as respectivas descrições. O exemplo seguinte demonstra como definir ou obter a opção para um conjunto de código de caracteres de pesquisa de texto.

Java

```
DKDatastoreTS dsTS = new DKDatastoreTS();
Integer input_option = new Integer(DK_TS_CCSSID_00850);
Integer output_option = null;

dsTS.setOption(DK_TS_OPT_CCSSID, input_option);
output_option = (Integer) dsTS.getOption(DK_OPT_TS_CCSSID);
```

C++

```
DKAny input_option = DK_CCSSID_00850;
DKAny output_option;
dsTS.setOption(DK_OPT_TS_CCSSID, input_option);
dsTS.getOption(DK_OPT_TS_CCSSID, output_option);
```

A opção_output é um objecto, mas normalmente é convertida para um Inteiro.

Sugestões: As opções de CCSID e LANG são indissociáveis. Caso uma seja especificada, a outra também terá de o ser. São especificadas as predefinições CCSID e LANG pelas opções DKDatastoreTS, DK_OPT_TS_CCSSID e DK_OPT_TS_LANG. Consulte a referência de API online para obter a lista das opções do servidor de conteúdos e as respectivas descrições.

Poderá especificar mais do que uma opção de pesquisa para uma condicionante da consulta. As opções de pesquisa são separadas por vírgulas. É dado um exemplo de várias condicionantes de pesquisa em “Consulta de obtenção global de texto (GTR)” na página 262.

Se ambas as opções de pesquisa SC (carácter único exigido) e MC (sequência de caracteres), o utilizador terá de especificar primeiro a opção de pesquisa SC . Por exemplo, \$SC=?,MC=*\$ U?I*.

Listagem de servidores

O objecto de DKDatastoreTS fornece uma função de listagem dos servidores de pesquisa de texto a que se pode ligar. O exemplo que se segue demonstra como obter a lista de servidores.

Java

```
DKServerDefTS pSV = null;
DKIndexTS pIndx = null;
String strServerName = null;
char chServerLocation = ' ';
String strLoc = null;
String strIndexName = null;
String strLibId = null;
int i = 0;
DKDatastoreTS dsTS = new DKDatastoreTS();
System.out.println("list servers");
pCol = (DKSequentialCollection)dsTS.listDataSources();
pIter = pCol.createIterator();
while (pIter.more() == true)
{
    i++;
    pSV = (DKServerDefTS)pIter.next();
    strServerName = pSV.getName();
    chServerLocation = pSV.getServerLocation();
    if (chServerLocation == DK_TS_SRV_LOCAL)
        strLoc = "LOCAL SERVER";
    else if (chServerLocation == DK_TS_SRV_REMOTE)
        strLoc = "REMOTE SERVER";
    System.out.println("Server Name [" + i + "] - " + strServerName +
        " Localização Servidor - " + strLoc);
}
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TListCatalogTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
DKSequentialCollection *pCol = 0;
dkIterator *pIter = 0;
DKString strServerName;
char chServerLocation = ' ';
DKString strLoc;
DKServerDefTS *pSV = 0;
long i = 0;
DKAny a;
cout << "list servers" << endl;
a = dsTS.listDataSources();
pCol = (DKSequentialCollection*)((dkCollection*)a);
pIter = pCol->createIterator();
while (pIter->more() == TRUE)
{
    i++;
    pSV = (DKServerDefTS*)((void*)(*pIter->next()));
    strServerName = pSV->getName();
    chServerLocation = pSV->getServerLocation();
    if (chServerLocation == DK_SRV_LOCAL)
    {
        strLoc = "LOCAL SERVER";
    }
    else if (chServerLocation == DK_SRV_REMOTE)
    {
        strLoc = "REMOTE SERVER";
    }
    cout << "Server Name [" << i << "] - " << strServerName
        << " Server Location - " << strLoc << endl;
    delete pSV;
}
delete pIter;
delete pCol;
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TListCatalogTS.cpp) está disponível no directório Cmbroot/Samples/cpp/dl.

A lista de servidores é devolvida numa DKSequentialCollection de objectos de DKServerInfoTS. Após ter obtido um objecto de DKServerInfoTS, o utilizador poderá obter o nome do servidor e a localização. Depois poderá utilizar o nome do servidor para estabelecer uma ligação a ele.

Enumerar esquemas

Um objecto de DKDatastoreTS fornece funções para a listagem de esquemas. Na pesquisa de texto, são índices de pesquisa de texto. O exemplo que se segue demonstra como obter a lista de índices.

A lista de índices é devolvida num objecto de DKSequentialCollection de objectos de DKIndexTS. Após obter um objecto de DKIndexTS, o utilizador poderá obter informações sobre o índice, tais como o ID do nome e da biblioteca, que poderá utilizar para formar uma consulta.

Java

```
tsCol = (DKSequentialCollection) dsTS.listEntities();
tsIter = pCol.createIterator();
int i = 0;
while (tsIter.more()) {
    i++;
    TsIndx = (DKSearchIndexDefTS)tsIter.next();
    strIndexName = TsIndx.getName();
    strLibId = TsIndx.getLibraryId();
    ...           \\ Process the list as appropriate
}
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TListCatalogTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
DKSequentialCollection *pCol = 0;
dkIterator *pIter = 0;
DKString strIndexName;
DKString strLibId;
DKServerDefTS *pSV = 0;
DKSearchIndexDefTS *pIndx = 0;
long i = 0;
DKAny a;
cout << "connecting to datastore" << endl;
dsTS.connect("TM","","");
cout << "list search indexes" << endl;
pCol = (DKSequentialCollection*)((dkCollection*)dsTS.listEntities());
pIter = pCol->createIterator();
i = 0;
while (pIter->more() == TRUE)
{
    i++;
    pIndx = (DKSearchIndexDefTS*)((void*)(*pIter->next()));
    strIndexName = pIndx->getName();
    strLibId = pIndx->getLibraryId();
    cout << "index name [" << i << "] - " << strIndexName
         << " Library - " << strLibId << endl;
    delete pIndx;
}
delete pIter;
delete pCol;
dsTS.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TListCatalogTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1. Consulte também “Gerir memória em recolhas (Apenas para C++)” na página 98 para obter informações relativas a eliminar a recolha.

Indexação de XDOs através do motor de pesquisa

Antes de pesquisar artigos de dados com Motor de Pesquisa de Texto e pesquisa de dados, primeiro tem de indexar os dados. Os índices requerem três valores: SearchEngine, SearchIndex e SearchInfo.

O valor da propriedade SearchIndex é uma combinação de dois nomes: o nome do serviço de pesquisa e o nome do índice de pesquisa. Por exemplo, se tiver definido um servidor de pesquisa de texto denominado TM no cliente de administração de sistemas e um índice de pesquisa denominado TMINDEX associado ao primeiro, o valor de SearchIndex é TM-TMINDEX.

Para um objecto que vai ser indexado pelo Motor de Pesquisa de Texto, o valor de SearchEngine terá de ser SM, para um artigo de dados ser indexado por consulta pelo pesquisa de imagens, o valor de SearchEngine terá de ser QBIC (para obter mais informações sobre pesquisa de imagens, consulte “Compreender as condições e os conceitos da pesquisa de imagens” na página 284).

O SearchIndex para QBIC é uma combinação de três valores: QBIC nome da base de dados, QBIC nome do catálogo e QBIC nome do servidor. Por exemplo, se o QBIC nome da base de dados for SAMPLEDB, o QBIC nome do catálogo for SAMPLECAT e o QBIC nome do servidor for QBICSRV, então o valor correcto para SearchIndex seria SAMPLEDB-SAMPLECAT-QBICSRV.

Consulte LoadSampleTSQBICDL e LoadFolderTSQBICDL no directório CMBR00T\Samples para obter exemplos relativos a carregar dados, a criar uma pasta ou carregar dados.

Importante: Ao adicionar um objecto da parte para indexação por um motor de pesquisa, não defina o TipoRep. Actualmente, o Motor de Pesquisa de Texto funciona apenas com o TipoRep predefinido FRN\$NULL.

Adicionar um XDO para indexação através do Motor de Pesquisa de Texto: O exemplo seguinte demonstra como adicionar um XDO que será indexado:

Java

```
// ----- Declare variables for part ID, item ID, and file
int    partId = 20;
String itemId = "CPPIORH4JBIXWIY0";
String fileName = "g:\\test\\testsrch.txt";
try {
    DKDatastoreDL dsDL = new DKDatastoreDL(); // create datastore
    ...                                     // connect to datastore
    dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");
    DKBlobDL axdo = new DKBlobDL(dsDL);      // create XDO
    DKPidXDODL apid = new DKPidXDODL();      // create PID
    apid.setPartId(partId);                  // set partId
    apid.setPrimaryId(itemId);               // set itemId
    axdo.setPidObject(apid);                 // setPid to XDO
    axdo.setContentClass(DK_DL_CC_ASCII);    // set ContentClass to text

    // --- set the searchEngine
    DKSearchEngineInfoDL aSrchEx = new DKSearchEngineInfoDL();
    aSrchEx.setSearchEngine("SM");
    aSrchEx.setSearchIndex("TM-TMINDEX");
    aSrchEx.setSearchInfo("ENU");
    axdo->setExtension("DKSearchEngineInfoDL", (dkExtension)aSrchEx);
    ...
    // ----- Set file content to buffer area
    axdo.setContentFromClientFile(fileName);
    axdo.add();                             //add from buffer
    ...
    // ----- Display the partId after add
    System.out.println("after add partId = " + ((DKPidXDODL)
        (axdo.getPidObject())).getPartId());

    dsDL.disconnect();                      //disconnect from datastore
    dsDL.destroy();
}
// ----- Catch any exception
catch (...)
```

C++

```
void main(int argc, char *argv[])
{
    DKDatastoreDL dsDL;
    DKString itemId, repType;
    int partId;
    itemId = "N2JJBERBQFK@WTVL";
    repType = "FRN$NULL";
    partId = 10;
    if (argc == 1)
    {
        cout<<"invoke: indexPartxsDL <partId> <repType> <itemId>"<<endl;
        cout<<" no parameter, following default will be provided:"<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 2)
    {
        partId = atoi(argv[1]);
        cout<<"you enter: indexPartxsDL "<<argv[1]<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 3)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        cout<<"you enter: indexPartxsDL "<<argv[1]<<" "<<argv[2]<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 4)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        itemId = DKString(argv[3]);
        cout<<"you enter: indexPartxsDL "<<argv[1]<<" "<<argv[2]<<" "<<argv[3]<<endl;
    }

    cout << "connecting Datastore" << endl;
    try
    {
        //replace following with your library server, user ID, password
        dsDL.connect("LIBSRVN","FRNADMIN","PASSWORD");

        cout << "datastore connected" << endl;

        DKBlobDL* axdo = new DKBlobDL(&dsDL);
        DKPidXDODL* apid = new DKPidXDODL;
        apid ->setPartId(partId);
        apid ->setPrimaryId(itemId);
        apid ->setRepType(repType);
        axdo ->setPidObject(apid);
        cout<<"itemId= "<<axdo->getItemId()<<endl;
        cout<<"partId= "<<axdo->getPartId()<<endl;
        cout<<"repType= "<<axdo->getRepType()<<endl;
    }

    // continued...
```

C++ (continuação)

```
//--- set searchEngine -----
cout<<"set search engine and setToBeIndexed()"<<endl;
DKSearchEngineInfoDL aSrchEx;
aSrchEx.setSearchEngine("SM");
aSrchEx.setSearchIndex("TM-TMINDEX");
aSrchEx.setSearchInfo("ENU");
axdo->setExtension("DKSearchEngineInfoDL", (dkExtension*)&aSrchEx);
axdo->setToBeIndexed();
cout<<"setToBeIndexed() done..."<<endl;

delete apid;
delete axdo;
dsDL.disconnect();
cout<<"datastore disconnected"<<endl;
}
catch(DKException &exc)
{
    cout << "Error id" << exc.errorId() << endl;
    cout << "Exception id " << exc.exceptionId() << endl;
    for(unsigned long i=0;i< exc.textCount();i++)
    {
        cout << "Error text:" << exc.text(i) << endl;
    }
    for (unsigned long g=0;g< exc.locationCount();g++)
    {
        const DKExceptionLocation* p = exc.locationAtIndex(g);
        cout << "Filename: " << p->fileName() << endl;
        cout << "Function: " << p->functionName() << endl;
        cout << "LineNumber: " << p->lineNumber() << endl;
    }
    cout << "Exception Class Name: " << exc.name() << endl;
}
cout << "done ..." << endl;
}
```

Importante: Quando adicionar um objecto da parte para indexação através de um motor de pesquisa, não defina o TipoRep (tipo de representação). O Motor de Pesquisa de Texto funciona apenas com o TipoRep predefinido FRN\$NULL.

Carregar dados para indexação através do Motor de Pesquisa de Texto: Para carregar dados num Content Manager para ser indexado pelo Motor de Pesquisa de Texto, o utilizador terá de criar um índice e um índice de pesquisa de texto.

Antes que possa criar um índice de pesquisa de texto, o servidor de pesquisa de texto tem de estar em execução. Certifique-se de que o seu ambiente está devidamente definido. Para isso, pode personalizar e executar os exemplos: TListCatalogDL eTListCatalogTS no directório CMBROOT\Samples.

Para criar partes no Content Manager que são indexadas pelo Motor de Pesquisa de Texto, consulte “Trabalhar com objectos de dados expandidos (XDOs)” na página 50.

Depois de carregar os dados para o Content Manager, utilize a função wakeUpService na classe de DKDatastoreDL para substituir os documentos na fila de documentos. Esta função assume um nome de motor de pesquisa como um parâmetro.

De seguida, a partir da janela de Administração de pesquisa de texto do Content Manager, conclua os seguintes passos:

1. Faça duplo clique sobre o servidor de pesquisa de texto.
2. Faça duplo clique sobre o índice de pesquisa de texto.
3. Faça clique sobre **INDEX**.

Esta acção irá indexar os documentos numa consulta de documentos. Terminada a indexação, já poderá executar consultas de pesquisa de texto.

Para obter mais informações sobre a administração de pesquisa de texto, consulte o *Manual de Administração do Sistema*.

Utilizar o suporte de documentos estruturados por texto

Os documentos estruturados por texto são compostos por textos identificados (como, por exemplo, um ficheiro HTML). Um modelo de documentos define a estrutura do documento e o Motor de Pesquisa de Texto pode pesquisar palavras ou expressões entre os identificadores.

Poderá executar consultas de texto em documentos estruturados da seguinte forma:

1. Crie um modelo de documento. O modelo de documentos contém secções e cada secção inclui o nome de secção e o identificador de documento utilizados. Por exemplo:

```
<HTML>
<HEAD>
<TITLE>Acme Corp<br></TITLE>
</HEAD>
<BODY>
<H1>Acme Corp<BR></H1>
<P><B>Acme Corp<BR></B><BR>
<P>John Smith <BR>
<P><ADDRESS>Acme Corporation<BR></ADDRESS>
<HR>
<H2>Acme Corp Business Objectives</H2>
<HR>
<P>
<H2><A NAME="Header_Test" HREF="#ToC_Test">Marketing</A></H2>
<P>Temos de aumentar a nossa oportunidade de mercado em 25%.
<P>Temos de ir de encontro às necessidades dos nossos clientes.
</BODY>
</HTML>
```

2. Crie um índice de pesquisa de texto que utilize o modelo de documentos do Content Manager.
3. Defina as regras de indexação para o índice de pesquisa de texto e especificar o formato do documento predefinido (por exemplo, DK_TS_DOCFMT_HTML para ficheiros HTML)
4. Adicione objectos de parte ao servidor de Content Manager.
5. Inicie o processo de indexação para os índices de pesquisa de texto.

Este exemplo mostra como deve enumerar os modelos de documentos definidos no seu sistema.

Java

```
// ----- Initialize the variables
DKSequentialCollection pCol = null;
DKDatastoreDefTS dsDef = null;
DKDatastoreAdminTS dsAdmin = null;
dkIterator pIter = null;
DKDocModelTS pDocModel = null;
int ccsid = 0;
String strDocModelName = null;
int i = 0;

// ----- Create the datastore and connect
DKDatastoreTS dsTS = new DKDatastoreTS();
dsTS.connect(srchSrv,"",' ');

dsDef = (DKDatastoreDefTS)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS)dsDef.datastoreAdmin();

// ----- Get list of document models
pCol = (DKSequentialCollection) dsAdmin.listDocModels("");
pIter = pCol.createIterator();
i = 0;
while (pIter.more() == true)
{
    i++;
    pDocModel = (DKDocModelTS)pIter.next();
    strDocModelName = pDocModel.getName();
    ccsid = pDocModel.getCCSID();
}
dsTS.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado(TListDocModelsTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
DKDatastoreDefTS* dsDef = 0;
DKDatastoreAdminTS* dsAdmin = 0;
DKDocModelTS* docModel = 0;
DKSequentialCollection *pCol = 0;
long ccsid = 0;
DKString strDocModelName;
dkIterator *pIter = 0;
long i = 0;
DKAny a;

dsTS.connect(srchSrv,"","");

dsDef = (DKDatastoreDefTS*)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS*)dsDef->datastoreAdmin();

// list document models
pCol = (DKSequentialCollection*)dsAdmin->listDocModels("");
pIter = pCol->createIterator();
while (pIter->more() == TRUE)
{
    i++;
    docModel = (DKDocModelTS*)((void*)(*pIter->next()));
    strDocModelName = docModel->getName();
    ccsid = docModel->getCCSID();
    delete docModel;
}
delete pIter;
delete pCol;

dsTS.disconnect();
```

A aplicação exemplo completa da qual foi retirada este exemplo (TListDocModelsTS.cpp) está disponível no directório Cmbroot/Samples/cpp/dl.

O exemplo seguinte demonstra como deve criar um modelo de documento:

Java

```
// ----- Create datastore and connect
DKDatastoreTS dsTS = new DKDatastoreTS();
DKDatastoreDefTS dsDef = null;
DKDatastoreAdminTS dsAdmin = null;

// ----- Create an instance of a document model object
DKDocModelTS docModel = new DKDocModelTS();

// ----- Create 2 instances of a document section objects for the model
DKDocSectionTS docSection = new DKDocSectionTS();
DKDocSectionTS docSection2 = new DKDocSectionTS();

// ----- Describe the document model for text document structure
//          for files like tstruct.htm above
docModel.setCCSID(DK_TS_CCSID_00850);
docModel.setName(docModelName);
docSection.setName("SAMPITITLE");
docSection.setTag("TITLE");
docModel.addDocSection(docSection);
docSection2.setName("SAMPCORPBODY");
docSection2.setTag("BODY");
docModel.addDocSection(docSection2);

dsTS.connect("TMMUF","", "", "");

dsDef = (DKDatastoreDefTS)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS)dsDef.datastoreAdmin();

// ----- Create the document model
dsAdmin.createDocModel("", docModel);

dsTS.disconnect();
dsTS.destroy();

Refer to TCreateDocModelTS.java and TCreateStructDocIndexTS.java in the
CMBROOT\Samples\java\d1 directory for more examples.
```

C++

```
DKDatastoreTS dsTS;
DKDatastoreDefTS* dsDef = 0;
DKDatastoreAdminTS* dsAdmin = 0;

// create an instance of a document model object
DKDocModelTS* docModel = new DKDocModelTS();

// create 2 instances of a document section objects for the model
DKDocSectionTS* docSection = new DKDocSectionTS();
DKDocSectionTS* docSection2 = new DKDocSectionTS();

// Describe the document model for text document structure
// for files like tstruct.htm above

docModel->setCCSID(DK_TS_CCSID_00850);
docModel->setName("SAMP CORPMOD");
docSection->setName("SAMP CORPTITLE");
docSection->setTag("TITLE");
docModel->addDocSection(docSection);
docSection2->setName("SAMP CORP BODY");
docSection2->setTag("BODY");
docModel->addDocSection(docSection2);

dsTS.connect("TMMUF","", "", "");

dsDef = (DKDatastoreDefTS*)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS*)dsDef->datastoreAdmin();

// create doc model
dsAdmin->createDocModel("", docModel);

// delete document model & sections
delete docModel;

dsTS.disconnect();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TCreateDocModelTS.cpp) e (TCreateStructDocIndexTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

O exemplo seguinte demonstra como deve criar e definir as regras de indexação de um índice de pesquisa de texto que utiliza um modelo de documento:

Java

```
// ----- Create the datastore and index rules object
DKDatastoreTS dsTS = new DKDatastoreTS();
DKDatastoreDefTS dsDef = null;
DKDatastoreAdminTS dsAdmin = null;
DKIndexingRulesTS indexRules = new DKIndexingRulesTS();

// ----- Create an instance of a document model object
DKDocModelTS docModel = new DKDocModelTS();

// ----- Create 2 instances of a document section objects for the model
DKDocSectionTS docSection = new DKDocSectionTS();
DKDocSectionTS docSection2 = new DKDocSectionTS();

// ----- Create the document model instance for indexing rules
DKDocModelTS docModel2 = new DKDocModelTS();
docModel2.setCCSID(DK_TS_CCSID_00850);
docModel2.setName("SAMPCORPMOD");

// ----- Describe the document model for text document structure
//           for files like tstruct.htm above
docModel.setCCSID(DK_TS_CCSID_00850);
docModel.setName("SAMPCORPMOD");
docSection.setName("SAMPITITLE");
docSection.setTag("TITLE");
docModel.addDocSection(docSection);
docSection2.setName("SAMPCORPBODY");
docSection2.setTag("BODY");
docModel.addDocSection(docSection2);

// ----- Connect to the datastore
dsTS.connect("TMMUF","", "", "");

dsDef = (DKDatastoreDefTS)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS)dsDef.datastoreAdmin();

DKSearchIndexDefTS pEnt = new DKSearchIndexDefTS((dkDatastore)dsTS);
pEnt.setName("TSTRUCT");
pEnt.setIndexType(DK_TS_INDEX_TYPE_PRECISE);
pEnt.setIndexProperty(DK_TS_PROPERTY_SECTIONS_ENABLED);
pEnt.setLibraryId("LIBSUM");
pEnt.setLibraryClientServices("IMLLSCDL");
pEnt.setLibraryServerServices("IMLLSSDL");
String strIndexFileDir = "e:\\tsindex\\index\\tstruct";
// ----- For AIX us the following form for the file
//   String strIndexFileDir = "/home/cltadmin/tsindex/tstruct";
pEnt.setIndexDataArea(strIndexFileDir);
String strWorkFileDir = "e:\\tsindex\\work\\tstruct";
// ----- For AIX us the following form for the file
//   String strWorkFileDir = "/home/cltadmin/work/tstruct";
pEnt.setIndexWorkArea(strWorkFileDir);

// ----- Associate document model with index
pEnt.addDocModel(docModel);

// ----- Create text search index that supports sections
dsDef.add((dkEntityDef)pEnt);

// continued...
```

Java (continuação)

```
indexRules.setIndexName("TSTRUCT");
indexRules.setDefaultDocFormat(DK_TS_DOCFMT_HTML);
indexRules.setDefaultDocModel(docModel2);

dsAdmin.setIndexingRules(indexRules);

dsTS.disconnect();
dsTS.destroy();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TCreateStructDocIndexTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
DKDatastoreDefTS* dsDef = 0;
DKDatastoreAdminTS* dsAdmin = 0;
DKIndexingRulesTS* indexRules = new DKIndexingRulesTS();

// create an instance of a document model object
DKDocModelTS* docModel = new DKDocModelTS();

// create 2 instances of a document section objects for the model
DKDocSectionTS* docSection = new DKDocSectionTS();
DKDocSectionTS* docSection2 = new DKDocSectionTS();

// doc model instance for indexing rules
DKDocModelTS* docModel2 = new DKDocModelTS();
docModel2->setCCSID(DK_TS_CCSID_00850);
docModel2->setName("SAMPCORPMOD");

// Describe the document model for text document structure
// for files like tstruct.htm above

docModel->setCCSID(DK_TS_CCSID_00850);
docModel->setName("SAMPCORPMOD");
docSection->setName("SAMPCORPTITLE");
docSection->setTag("TITLE");
docModel->addDocSection(docSection);
docSection2->setName("SAMPCORPBODY");
docSection2->setTag("BODY");
docModel->addDocSection(docSection2);

// continued...
```

C++ (continuação)

```
dsTS.connect("TMMUF","", "", "");

dsDef = (DKDatastoreDefTS*)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS*)dsDef->datastoreAdmin();

DKSearchIndexDefTS* pEnt = new DKSearchIndexDefTS(&dsTS);

pEnt->setName("TSTRUCT");
pEnt->setIndexType(DK_TS_INDEX_TYPE_PRECISE);

// This index is text structure document section enabled
pEnt->setIndexProperty(DK_TS_PROPERTY_SECTIONS_ENABLED);

pEnt->setLibraryId("LIBSUM");
pEnt->setLibraryClientServices("IMLLSCDL");
pEnt->setLibraryServerServices("IMLLSSDL");
DKString strIndexFileDir = "e:\\tsindex\\index\\tstruct";
//**** for AIX ****
//DKString strIndexFileDir = "/home/cltadmin/tsindex/index/tstruct";
pEnt->setIndexDataArea(strIndexFileDir);
DKString strWorkFileDir = "e:\\tsindex\\work\\tstruct";
//**** for AIX ****
//DKString strWorkFileDir = "/home/cltadmin/tsindex/work/tstruct";
pEnt->setIndexWorkArea(strWorkFileDir);

// Associate document model with index
pEnt->addDocModel(docModel);

// Create text search index that supports sections
dsDef->add(pEnt);

delete pEnt;

indexRules->setIndexName("TSTRUCT");
indexRules->setDefaultDocFormat(DK_TS_DOCFMT_HTML);
indexRules->setDefaultDocModel(docModel2);
dsAdmin->setIndexingRules(indexRules);

delete indexRules;

dsTS.disconnect();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TCreateStructDocIndexTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

O exemplo seguinte demonstra como deve iniciar o processo de indexação, que é assíncrono. Ao utilizar o programa de administração do sistema, pode iniciar o processo de indexação e verificar o seu estado.

Java

```
// ----- Declare datastore and administration
DKIndexFuncStatusTS pIndexFuncStatus = null;
DKDatastoreTS dsTS = new DKDatastoreTS();
DKDatastoreDefTS dsDef = null;
DKDatastoreAdminTS dsAdmin = null;

dsTS.connect("TMMUF","","","");

dsDef = (DKDatastoreDefTS)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS)dsDef.datastoreAdmin();

// ----- Start the indexing process
dsAdmin.startUpdateIndex(indexName);

// ----- Get indexing status
pIndexFuncStatus = dsAdmin.getIndexFunctionStatus(indexName,
                                                    DK_TS_INDEX_FUNCID_INDEX_DOCUMENTS);
.... // Process the status as appropriate

// ----- Show the scheduled document queue
System.out.println("*getScheduledDocs "
    + pIndexFuncStatus.getScheduledDocs());

// ----- Show the primary document queue
System.out.println("*getDocsInPrimaryIndex "
    + pIndexFuncStatus.getDocsInPrimaryIndex());

// ----- Shows the secondary document queue
System.out.println("*getDocsInSecondaryIndex " +
    pIndexFuncStatus.getDocsInSecondaryIndex());
System.out.println("*getDocMessages "
    + pIndexFuncStatus.getDocMessages());
if (pIndexFuncStatus.isCompleted() == true)
{
    // ---- Processing if indexing is completed
}

if (pIndexFuncStatus.getReasonCode() != 0)
{
    dsAdmin.setIndexFunctionStatus(indexName,
        DK_TS_INDEX_FUNCID_INDEX_DOCUMENTS, DK_TS_INDEX_ACTID_RESET);
}

dsTS.disconnect();
dsTS.destroy();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TIndexingTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
DKDatastoreDefTS* dsDef = 0;
DKDatastoreAdminTS* dsAdmin = 0;
DKIndexFuncStatusTS* pIndexFuncStatus = 0;

dsTS.connect(srchSrv,"","");

dsDef = (DKDatastoreDefTS*)dsTS.datastoreDef();
dsAdmin = (DKDatastoreAdminTS*)dsDef->datastoreAdmin();

// starts the indexing process
dsAdmin->startUpdateIndex(srchIndex);

// Get indexing status
pIndexFuncStatus = dsAdmin->getIndexFunctionStatus(srchIndex,
    DK_TS_INDEX_FUNCID_INDEX_DOCUMENTS);

cout << "***** index status *****" << endl;
cout << ".*isCompleted " << pIndexFuncStatus->isCompleted() << endl;
cout << ".*getEnabledId " << pIndexFuncStatus->getEnabledId() << endl;
cout << ".*getReasonCode " << pIndexFuncStatus->getReasonCode()
    << endl;
cout << ".*getFuncStopped " << pIndexFuncStatus->getFunctionStopped()
    << endl;
cout << ".*getStartedLast " << pIndexFuncStatus->getStartedLast()
    << endl;
cout << ".*getEndedLast " << pIndexFuncStatus->getEndedLast() << endl;
cout << ".*getStartedExecution " << pIndexFuncStatus->getStartedExecution()
    << endl;
cout << ".*getScheduledDocs " << pIndexFuncStatus->getScheduledDocs()
    << endl;
cout << ".*getDocsInPrimaryIndex " << pIndexFuncStatus->getDocsInPrimaryIndex()
    << endl;
cout << ".*getDocsInSecondIndex " << pIndexFuncStatus->getDocsInSecondIndex()
    << endl;
cout << ".*getDocMessages " << pIndexFuncStatus->getDocMessages()
    << endl;
if (pIndexFuncStatus->isCompleted() == TRUE)
{
    // indexing completed
}
if (pIndexFuncStatus->getReasonCode() != 0)
{
    dsAdmin->setIndexFunctionStatus(srchIndex,
        DK_TS_INDEX_FUNCID_INDEX_DOCUMENTS,DK_TS_INDEX_ACTID_RESET);
}
delete pIndexFuncStatus;
dsTS.disconnect();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TIndexingTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Consulte TCheckStatusTS no directórioCMBROOT\Samples para obter um exemplo de verificação de estado. O exemplo verifica se um pedido colocado em fila foi removido da fila de documento programada para a fila primária ou secundária. Se ocorrer um erro de indexação, pode verificar o ficheiro imldiag.log no directório de instâncias de pesquisa de texto.

O exemplo seguinte demonstra como deve executar uma consulta de texto de documentos de estrutura com base no modelo de documento e no índice de pesquisa de texto definido anteriormente.

Java

```
// ----- Create the datastore
DKDatastoreTS dsTS = new DKDatastoreTS();
dkResultSetCursor pCur = null;
DKNameValuePair parms[] = null;
// ----- Connect
dsTS.connect("TMMUF","", "", "");
// ----- Generate the query string
String cmd = "SEARCH=(COND=($CCSID=850," +
             "DOCMOD=(DOCMODNAME=SAMPCORPMOD," +
             "SECLIST=(SAMPCORPTITLE,SAMPCORPBODY))$ Corp));" +
             "OPTION=(SEARCH_INDEX=TMSTRUCT;MAX_RESULTS=5)";
// ----- Execute the query
pCur = dsTS.execute(cmd,DK_CM_TEXT_QL_TYPE,parms);

// ----- Process the results
.....
dsTS.disconnect();
dsTS.destroy();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TExecuteStructDocTS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreTS dsTS;
dkResultSetCursor* pCur = 0;

dsTS.connect("TMMUF","", "", "");

DKString cmd = "SEARCH=(COND=($CCSID=850,";
cmd += "DOCMOD=(DOCMODNAME=SAMPCORPMOD,";
cmd += "SECLIST=(SAMPCORPTITLE,SAMPCORPBODY))$ Corp));";
cmd += "OPTION=(SEARCH_INDEX=TMSTRUCT;MAX_RESULTS=5)";

pCur = dsTS.execute(cmd);

// process the results

dsTS.disconnect();
```

A aplicação exemplo completa da qual foi retirado este exemplo (TExecuteStructDocTS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Pesquisar imagens pelo conteúdo

O servidor de Pesquisa de Imagens pode pesquisar imagens armazenadas quando o utilizador especifica o tipo de imagem ou faculta um exemplo da imagem.

O Figura 16 na página 284 demonstra uma aplicação exemplificativa que se liga ao servidor de pesquisa de imagens.O servidor de pesquisa de imagens utiliza

tecnologia de Pesquisa Por Conteúdo de Imagem (QBIC) para pesquisar cores semelhantes, esquemas ou padrões.

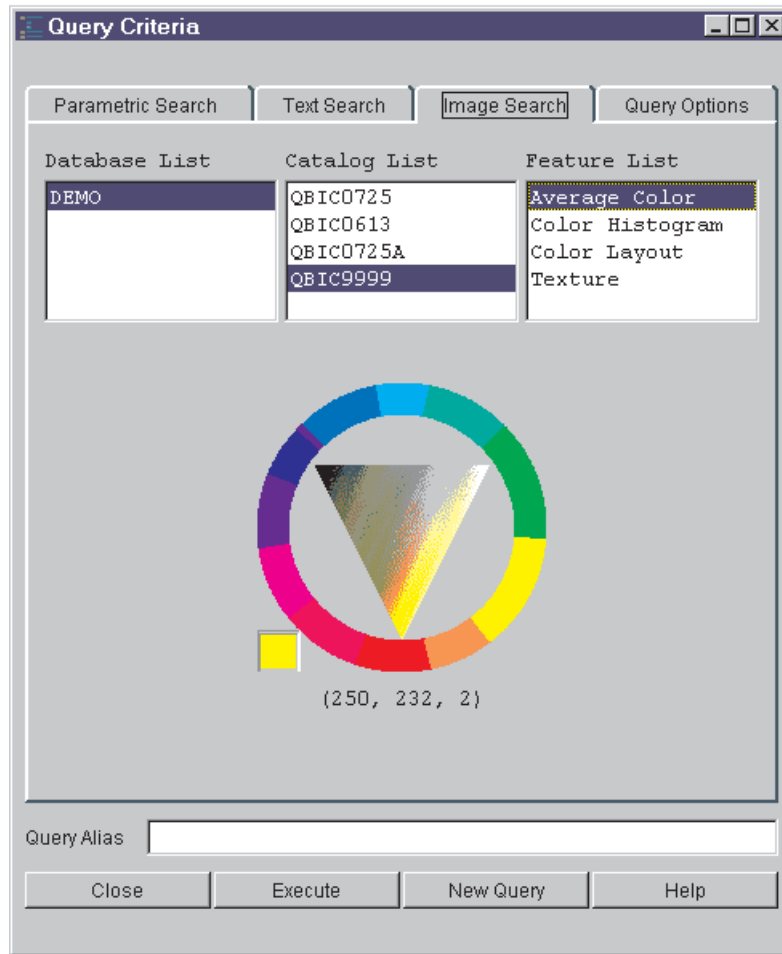


Figura 16. Cliente exemplificativo de pesquisa de imagens

Compreender as condições e os conceitos da pesquisa de imagens

Esta secção descreve os componentes de pesquisa de imagens: o servidor, as bases de dados, catálogos e a relação do servidor de pesquisa de imagens com versões anteriores do Content Manager. Também descreve as *funções* que são as características visuais passíveis de pesquisa das imagens.

Compreender servidores, bases de dados e catálogos de pesquisa de imagens: A versão anterior do Content Manager utiliza um servidor de pesquisa de imagens para pesquisar imagens. As aplicações do Content Manager armazenam objectos de imagem no servidor de objectos. O servidor de pesquisa imagens analisa e indexa as informações da imagem. O servidor de pesquisa de imagens não armazena as imagens em si.

Um servidor de conteúdos definido por um objecto DKDatastoreQBIC representa o servidor de pesquisa de imagens. Os resultados de uma pesquisa de imagens incluem identificadores (IDs de artigo) que descrevem a localização da imagem no servidor do Content Manager. Poderá utilizar estes identificadores com outros resultados, tais como o part number e TipoRep para obter a imagem.

Pode executar consultas no servidor de conteúdos. No entanto, o servidor de conteúdos de pesquisa de imagens não suporta operações de adição, actualização, obtenção e eliminação. O Figura 17 demonstra um exemplo de um servidor de pesquisa de imagens.

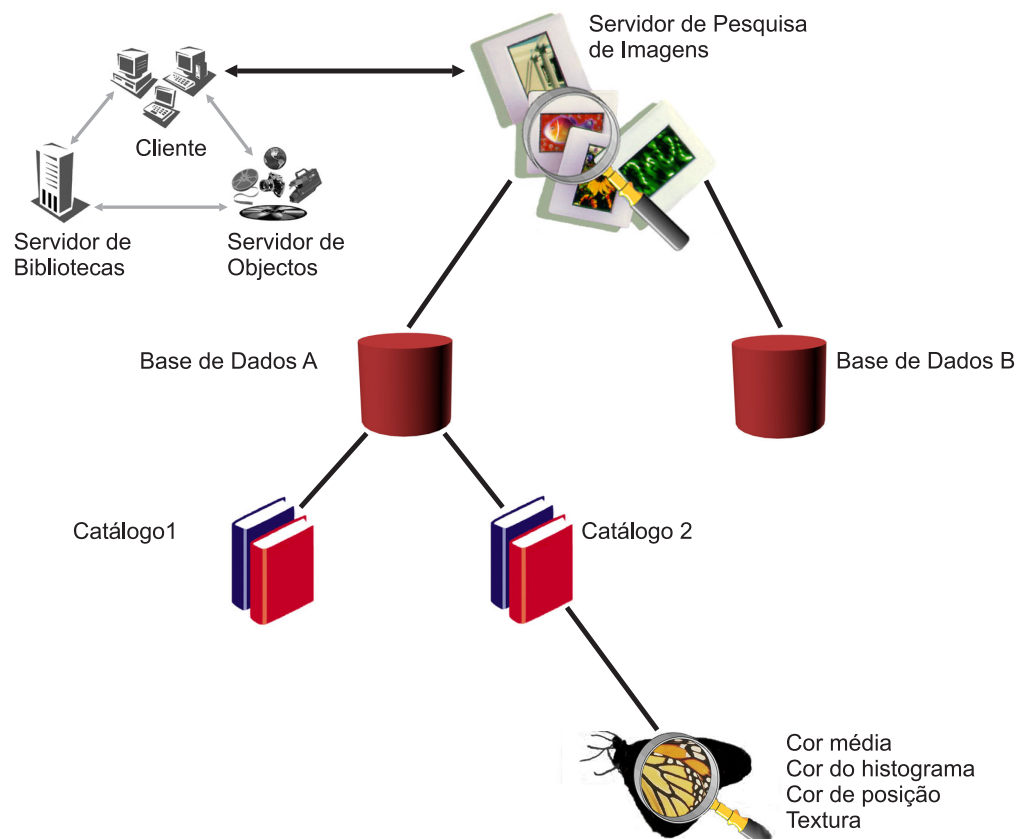


Figura 17. Um servidor de pesquisa de imagens num sistema Content Manager anterior

O servidor de pesquisa de imagens pode conter uma ou mais do que uma base de dados. Cada base de dados pode conter um ou mais do que um catálogo, que contenha informações relativas às características visuais das imagens. Estas quatro características de pesquisa de imagem são:

- Cor média.
- Cor de histograma.
- Cor posicional (esquema da cor).
- Textura.

Compreender as características da pesquisa de imagens: As quatro características de pesquisa de imagens e os seus objectivos estão definidos nesta secção:

Cor Média Utilize a cor média para pesquisar imagens que tenham uma cor predominante. As imagens com cores predominantes similares têm cores médias similares. Por exemplo, as imagens que contêm porções iguais de vermelho e amarelo possuem uma cor média laranja.

QbColorFeatureClass é o nome da característica de cor média.

Cor de Histograma

Mede a percentagem de distribuição da cor numa imagem. A análise do histograma mede separadamente as diferentes cores de uma imagem. Por exemplo, uma paisagem campestre tem uma cor de histograma que revela uma alta frequência de azul, verde e cinzento.

Utilize a cor de histograma para pesquisar imagens que contenham uma variedade semelhante de cores.

`QbColorHistogramFeatureClass` é o nome da característica de cor de histograma.

Cor posicional (esquema da cor)

As cores posicionais medem o valor da cor média em pixels numa área especificada de uma imagem. Por exemplo, as imagens com objectos que no meio apresentem a cor vermelho vivo têm uma cor posicional de vermelho vivo.

`QbDrawFeatureClass` é o nome da característica de cor posicional.

Textura

Utilize a textura para pesquisar imagens que tenham um padrão em particular. A textura mede a espessura, contraste e direcionalidade de uma imagem. A espessura indica o tamanho de artigos que se repetem numa imagem. O contraste identifica as variações de luminosidade numa imagem. A direcionalidade indica se uma direcção é predominante numa imagem. Por exemplo, uma imagem que contenha fibra de madeira tem uma textura semelhante a outras imagens que contenham fibra de madeira.

`QbTextureFeatureClass` é o nome da característica de textura.

Utilizar aplicações de pesquisa de imagens

As aplicações cliente de pesquisa de imagens criam consultas de imagens, executam-nas e depois avaliam as informações devolvidas pelo servidor de pesquisa de imagens. Antes que uma aplicação possa pesquisar imagens pelo conteúdo, as imagens terão de ser indexadas e terão de se armazenar as informações do conteúdo numa base de dados de pesquisa de imagens.

Restrição: Não poderá indexar imagens existentes no seu servidor de objectos. Apenas poderá indexar as imagens que criar no seu servidor de objectos após a instalação do servidor de pesquisa de imagens e do cliente. O Figura 18 na página 287 demonstra um exemplo do cliente e de imagens obtidas.

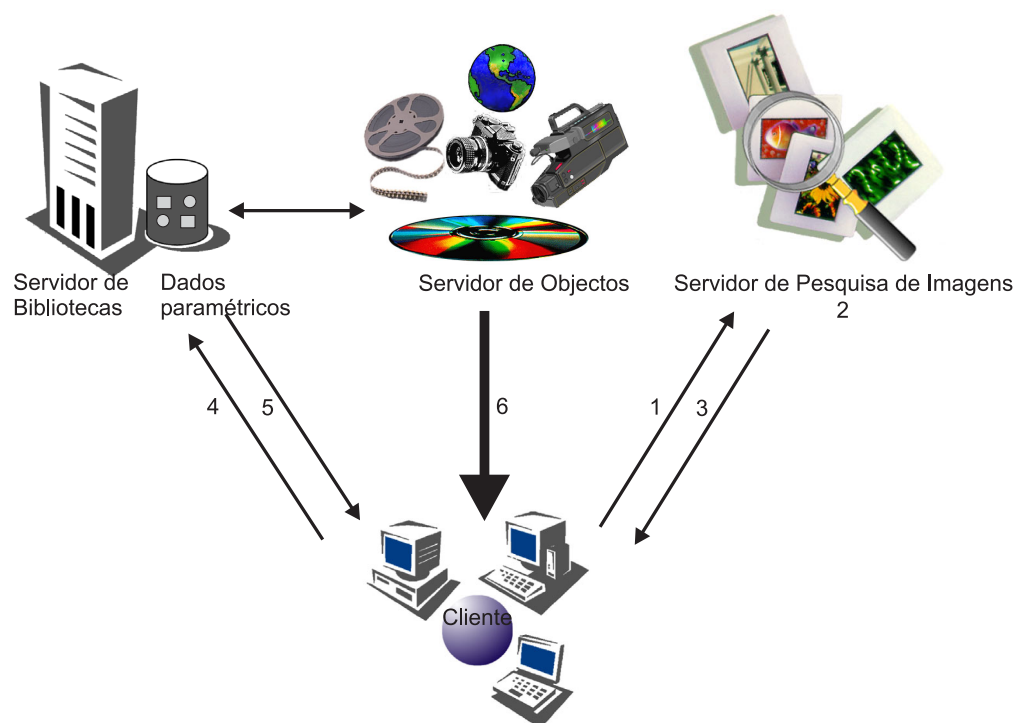


Figura 18. Como os clientes de pesquisa de imagens pesquisam e obtêm imagens

Para executar uma pesquisa de imagens:

1. Um cliente constrói uma cadeia de consultas QBIC e envia-a para um servidor de pesquisa de imagens.
2. O servidor de pesquisa de imagens recebe a cadeia de consultas e pesquisa as imagens catalogadas para obter uma correspondência.
3. O cliente recebe as correspondências como uma lista de identificadores. O identificador para cada imagem correspondente é formado por:
 - ID de Artigo.
 - Part number.
 - TipoRep.
 - Ordenação.
4. O cliente solicita as informações da parte e do índice da imagem a partir de um servidor de bibliotecas.
5. O servidor de bibliotecas devolve informações de índice, bem como de descrição de texto, ao cliente. O servidor de bibliotecas também exige que um servidor de objectos envie partes de imagens especificadas ao cliente.
6. O servidor de objectos envia partes de imagens e o cliente reconhece que as recebeu.

Criar consultas

Quando o utilizador criar consultas, terá de construir uma cadeia de consultas que a aplicação passe para o servidor de pesquisa de imagens. Uma consulta de imagens é uma cadeia de caracteres que especifica o critério de pesquisa. Os critérios de pesquisa consistem em:

Uma consulta de imagens é uma cadeia de caracteres que especifica o critério de pesquisa. Os critérios de pesquisa consistem em:

Nome da função

As funções utilizadas na pesquisa.

Valor da função

Os valores dessas funções. O Tabela 19 na página 289 revela os nomes e os valores da função de pesquisa de imagens que se podem passar numa cadeia de consulta.

Peso relativo O peso relativo ou ênfase colocados em cada função. O peso de uma função indica o ênfase que um servidor de pesquisa de imagens coloca numa função, ao calcular resultados por semelhança e ao devolver resultados de uma consulta. Quanto maior for o peso especificado, maior é o ênfase dado.

Máximo de resultados

Para além de definir o tipo de imagens que uma consulta irá procurar, poderá especificar o número máximo de correspondências que a consulta irá devolver.

Uma cadeia de consultas tem a forma: `valor nome_função`, onde `nome_função` é um nome da função de pesquisa de imagens e `valor` é o valor associado à função. Caso utilize mais do que uma função numa consulta, então terá de especificar um par nome função-valor para cada função. A cadeia "e" separa cada par.

As consultas de pesquisa de imagens têm a seguinte sintaxe:

valor nome_função
peso do valor nome_função

Não pode repetir uma função numa única consulta. Pode especificar várias funções numa consulta. Quando especificar várias funções numa consulta, poderá atribuir uma peso a uma ou mais funções. As consultas que incluem o ênfase para cada função têm o formato: `nome_função valor peso`, onde `nome_função` é um nome da função de pesquisa de imagens, `valor` é o valor associado à função e `peso` é o peso atribuído à função. O peso é a combinação da palavra chave peso, um sinal de igual (=) e um número real maior que 0.0.

Também poderá especificar o número máximo de correspondências que a consulta devolve. Para especificar o máximo de resultados, adicione à consulta e `max_resultados`. O `max_resultados` consiste na palavra chave max, um sinal de igual (=) e um inteiro maior do que zero 0. Tabela 19 na página 289 descreve os nomes e valores das funções.

Tabela 19. Consulta da pesquisa de imagens: valores de função válidos

| Nome da função | Valores |
|---|---|
| QbColorFeatureClass ou QbColor | <p>cor = < Valorrrgb , Valorrrgb , Valorrrgb > onde Valorrrgb é um inteiro de 0 a 255.</p> <p>ficheiro = < localizaçãoFicheiro , " nomeFicheiro " > onde localizaçãoFicheiro é servidor ou cliente, nomeFicheiro é o caminho de ficheiro completo especificado no formato para o sistema em que reside o ficheiro. Por exemplo, poderá pesquisar através de uma cor média e de um valor de textura. O valor de textura é fornecido por uma imagem num ficheiro no cliente. O peso da textura é o dobro do da cor média:</p> <p>QbColorFeatureClass cor= <50, 50, 50> e QbTextureFeatureClass ficheiro=<cliente, "\patterns\pattern1.gif"> peso=2.0</p> |
| QbColorHistogramFeatureClass ou QbHistogram | <p>histograma = < listaHist > onde listaHist consiste numa ou mais cláusulaHist separadas por uma vírgula (,).</p> <p>Uma cláusulaHist é especificada como (valorHist , Valorrrgb , Valorrrgb , Valorrrgb), onde valorHist é um inteiro de 1 para 100 (um valor percentual) e Valorrrgb é um inteiro de 0 para 255.</p> <p>ficheiro = < localizaçãoFicheiro , " nomeFicheiro " > onde localizaçãoFicheiro é servidor ou cliente, nomeFicheiro é o caminho de ficheiro completo especificado no formato para o sistema em que reside o ficheiro.</p> |

Tabela 19. Consulta da pesquisa de imagens: valores de função válidos (continuação)

| Nome da função | Valores |
|---------------------------------------|---|
| QbDrawFeatureClass ou QbDraw | <p>descrição = < " cadeiaDesc " > onde cadeiaDesc é uma cadeia codificada especial que descreve um ficheiro selectivo. Formato da cadeia de descrição:</p> <ol style="list-style-type: none"> 1. Dw,h especificam as dimensões exteriores da própria imagem com largura w e altura h. 2. Rx,y,w,h,r,g,b especificam que um rectângulo de largura w e altura h irá ser posicionado com o canto superior esquerdo nas coordenadas(x,y)—respeitante a uma origem no canto superior esquerdo do rectângulo da imagem—e este rectângulo deverá ter os valores de cor r (vermelho), g (verde) e b (azul). 3. O carácter dois pontos (:) é utilizado como separador. <p>Por exemplo, pode pesquisar o esquema da cor (QbDrawFeatureClass) descrito pela cadeia da descrição: QbDrawFeatureClass description= <"D100,50:R0,0,50,50,255,0,0"</p> <p>ficheiro = < localizaçãoFicheiro , " nomeFicheiro " > onde localizaçãoFicheiro é servidor ou cliente, nomeFicheiro é o caminho de ficheiro completo especificado no formato para o sistema em que reside o ficheiro.</p> |
| QbTextureFeatureClass ou QbTexture | <p>ficheiro = < localizaçãoFicheiro , " nomeFicheiro " > onde localizaçãoFicheiro é servidor ou cliente, nomeFicheiro é o caminho de ficheiro completo especificado no formato para o sistema em que reside o ficheiro.</p> |

Exemplos de consultas:

1. Pesquisar a cor média vermelho:
QbColorFeatureClass cor=<255,0,0>
2. Pesquisar através de um histograma de três cores, 10% vermelho, 50% azul e 40% verde:
QbColorHistogramFeatureClass histograma=
<(10, 255, 0, 0) (50, 0, 255, 0), (40, 0, 0, 255)>
3. Pesquisar através de uma cor média e de um valor de textura. O valor de textura é fornecido pela imagem num ficheiro no cliente. O peso da textura é o dobro do da cor média:
QbColorFeatureClass cor=
<50, 50, 50> e QbTextureFeatureClass ficheiro=<cliente, "\patterns\pattern1.gif">
peso=2.0
4. Pesquisar o esquema da cor descrita:
QbDrawFeatureClass descrição=<"D100,50:R0,0,50,50,255,0,0">
5. Pesquisar a cor média vermelho e limitar as correspondências devolvidas em cinco:
QbColorFeatureClass cor=<255,0,0> e max=5

Executar consultas e avaliar resultados da pesquisa

As aplicações utilizam a API de pesquisa de imagens para emitir consultas e avaliar resultados da pesquisa. Se a informação na base de dados da pesquisa de imagens corresponder ao critério de pesquisa de imagens, então é devolvido um identificador da imagem ou imagens correspondentes. Este identificador é um objecto de dados dinâmicos (DDO) que corresponde a uma parte da imagem dentro de um objecto do Content Manager.

Estabelecer uma ligação em QBIC

A pesquisa de imagens fornece funções de ligação e de desligação ao servidor de conteúdos. O exemplo que se segue demonstra como se deve ligar a um servidor de pesquisa de imagens denominado QBICSRV através do ID de utilizador QBICUSER e a palavra-passe PASSWORD.

Java

```
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
...           // Process as appropriate
dsQBIC.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TConnectQBIC.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
...           // do some work
dsQBIC.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TConnectQBIC.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

A ligação de pesquisa de imagens permite que uma aplicação se ligue a um servidor de pesquisa de imagens.

Após a ligação, o programa poderá utilizar funções que acedem a servidores de pesquisa de imagens, excepto as funções que não estão relacionados com os catálogos de pesquisa de imagens, tais como listarBasedados. É exigido uma função abrirCatálogo para abrir um catálogo para processamento. É chamado uma função fecharCatálogo após a finalização do processamento. O exemplo seguinte demonstra como deve ligar, abrir um catálogo, fechar um catálogo e desligar.

Java

```
// ----- Create a QBIC datastore and connect
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
// ----- open the catalog
dsQBIC.openCatalog("DEMO", "QBIC0725");
...           // Do some processing
dsQBIC.closeCatalog();
dsQBIC.disconnect();
dsQBIC.destroy();
```

C++

```
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
dsQBIC.openCatalog("DEMO", "QBIC0725");
... // do some work
dsQBIC.closeCatalog();
dsQBIC.disconnect();
```

Enumerar servidores de pesquisa de imagens

O servidor de pesquisa de imagens fornece uma função para enumerar os servidores de pesquisa de imagens a que se pode ligar. O exemplo seguinte demonstra como obter (num objecto de `DKSequentialCollection`) a lista de servidores que contém objectos `DKServerInfoQBIC`. Após ter obtido um objecto de `DKServerInfoQBIC`, poderá obter o nome do servidor, o nome do sistema central e o número da porta.

Java

```
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
.....
DKServerInfoQBIC pSV = null;
String strServerName = null;
String strHostName = null;
String strPortNumber = null;
pCol = (DKSequentialCollection)dsQBIC.listDataSources();
iter = pCol.createIterator();
while (iter.more()) {
    srvDef = (DKServerDefQBIC)iter.next();
    ..... // Process each server as appropriate
}
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (`TListCatalogQBIC.java`) está disponível no directório `CMBROOT\Samples\java\d1`.

C++

```
DKDatastoreQBIC dsQBIC;
DKSequentialCollection *pCol = 0;
dkIterator *pIter = 0;
DKServerDefQBIC *pSV = 0;
DKString strServerName;
DKAny a;
long i = 0;
cout << "list servers" << endl;
a = dsQBIC.listDataSources();
pCol = (DKSequentialCollection*)((dkCollection*)a);
pIter = pCol->createIterator();
while (pIter->more() == TRUE)
{
    i++;
    pSV = (DKServerDefQBIC*)((void*)(*pIter->next()));
    strServerName = pSV->getName();
    cout << "Server Name [" << i << "] - " << strServerName << endl;
    delete pSV;
}
delete pIter;
delete pCol;
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TListCatalogQBIC.cpp) está disponível no directório Cmbroot/Samples/cpp/dl.

Enumerar bases de dados, catálogos e funções de pesquisa de imagens

O DKDatastoreQBIC fornece uma função de listagem de todas as bases de dados, catálogos e funções de pesquisa de imagens num servidor de pesquisa de imagens. A lista é devolvida num objecto de DKSequentialCollection que contenha objectos de DKIndexQBIC. Após ter obtido um objecto de DKIndexQBIC, poderá obter o nome da base de dados, do catálogo e da função. O exemplo que se segue demonstra como obter a lista de bases de dados, catálogos e funções.

Java

```
// ----- Create the datastore and connect
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");

// ---- Get the list of servers
col = (DKSequentialCollection)dsQBIC.listDataSources();
iter = col.createIterator();
while (iter.more()) {
    srvDef = (DKServerDefQBIC)iter.next();
    .....    // Process each server as appropriate
}

// ----- Get the list of QBIC Databases
col = (DKSequentialCollection)dsQBIC.listEntities();
iter = col.createIterator();
while (iter.more()){
    dbDef = (DKDatabaseDefQBIC)iter.next();
    // ----- Get the list of catalogs for the database
    col2 = (DKSequentialCollection)dbDef.listSubEntities();
    iter2 = col2.createIterator();
    while (iter2.more()){
        catDef = (DKCatalogDefQBIC)iter2.next();
        // ----- Get the list of features for the catalog
        col3 = (DKSequentialCollection)catDef.listAttrs();
        iter3 = col3.createIterator();
        while (iter3.more()){
            featDef = (DKFeatureDefQBIC)iter3.next();
            .... // Process the features as appropriate
        }
    }
}
dsQBIC.disconnect();
dsQBIC.destroy();
.....
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TListCatalogQBIC.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreQBIC dsQBIC;
DKSequentialCollection *pCol = 0;
dkIterator *pIter = 0;
DKSequentialCollection *pCol2 = 0;
dkIterator *pIter2 = 0;
DKSequentialCollection *pCol3 = 0;
dkIterator *pIter3 = 0;
DKDatabaseDefQBIC *pEntDB = 0;
DKCatalogDefQBIC *pEntCat = 0;
DKString strCatName;
DKString strDBName;
DKString strFeatName;
DKFeatureDefQBIC *pAttr = 0;
DKAny a;
DKAny *pA = 0;
long i = 0;
long j = 0;
long k = 0;
cout << "connecting to datastore" << endl;
dsQBIC.connect("QBICSRV","USERID","PW");
cout << "list databases " << endl;
pCol = (DKSequentialCollection*)((dkCollection*)dsQBIC.listEntities());
pIter = pCol->createIterator();
i = 0;
while (pIter->more() == TRUE)
{
    i++;
    pEntDB = (DKDatabaseDefQBIC*)((void*)(*pIter->next()));
    strDBName = pEntDB->getName();
    cout << "database name [" << i << "] - " << strDBName << endl;
    cout << " list catalogs for DB " << strDBName << endl;
    pCol2=(DKSequentialCollection*)((dkCollection*)pEntDB->listSubEntities());
    pIter2 = pCol2->createIterator();
    j = 0;
    while (pIter2->more() == TRUE)
    {
        j++;
        pA = pIter2->next();
        pEntCat = (DKCatalogDefQBIC*) pA->value();
        strCatName = pEntCat->getName();
        cout << "catalog name [" << j << "] - " << strCatName << endl;
        pCol3=(DKSequentialCollection*)((dkCollection*)pEntCat->listAttrs());
        pIter3 = pCol3->createIterator();
        k = 0;
        while (pIter3->more() == TRUE)
        {
            k++;
            pA = pIter3->next();
            pAttr = (DKFeatureDefQBIC*) pA->value();
            cout << "    Attribute name [" << k << "] - "
                << pAttr->getName() << endl;
            cout << "    datastoreName " << pAttr->datastoreName()
                << endl;
            cout << "    datastoreType " << pAttr->datastoreType()
                << endl;
            cout << "    attributeOf " << pAttr->getEntityName()
                << endl;
            delete pAttr;
        }
    }
}
// continued...
```

C++ (continuação)

```
        delete pIter3;
        delete pCol3;
        delete pEntCat;
    }
    cout << " " << j << " features listed for catalog: "
        << strCatName << endl;
    delete pIter2;
    delete pCol2;
    delete pEntDB;
}
delete pIter;
delete pCol;
cout << i << " databases listed" << endl;
dsQBIC.disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TListCatalogQBIC.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Representar informações de pesquisa de imagens com um DDO

Um DDO associado a DKDatastoreQBIC contém informações específicas para representação de resultados de pesquisa de imagens. Um DDO resultante de uma consulta de imagens corresponde a uma parte de imagem dentro de um artigo, tendo este o seguinte conjunto de atributos padrão:

DKDLITEMID

O ID do artigo para o artigo a que pertence esta parte da imagem. Utilize o ID do artigo para obter todo o artigo do servidor de conteúdos.

DKPARTNO

Um part number inteiro desta parte da imagem. Utilize isto com o ID do artigo para obter esta parte do servidor de conteúdos.

DKREPTYPE

Uma cadeia para tipo de representação (TipoRep). O valor predefinido é FRN\$NULL. Este atributo é reservado.

DKRANK

Uma ordem de inteiro que demonstra a relevância desta parte para o conjunto de resultados da consulta de imagem. A ordem está entre o âmbito 0 até 100. Uma ordem mais elevada significa uma correspondência mais exacta.

O PID para um DDO de pesquisa de imagens tem as informações seguintes:

tipo de servidor de conteúdos

QBIC.

nome do servidor de conteúdos

O nome de servidor utilizado para estabelecer ligação ao servidor de conteúdos.

ID O número de sequência com base em zero do DDO no conjunto de resultados.

Como convenção, o valor do atributo é sempre um objecto.

Trabalhar com consultas de imagens

Esta secção descreve como executar e avaliar consultas de imagens.

Executar uma consulta de imagens

Ao utilizar uma instância de `dkQuery` de `DKDatastoreQBIC`, pode criar um objecto de consulta para executar a consulta e obter os resultados. O exemplo seguinte demonstra como criar e executar um objecto de consulta de imagens. Após executar uma consulta, os resultados são devolvidos numa recolha de `DKResults`.

Java

```
// ----- Generate a query string; then create the datastore and connect
String cmd = "QbColor color=<255, 0, 0>";
DKNameValuePair parms[] = null;
DKDDO item = null;
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
// ----- Open the catalog
dsQBIC.openCatalog("DEMO", "qbic0725");

...    // Process as appropriate

// ----- Create the query and run it
dkQuery pQry = dsQBIC.createQuery(cmd, DK_IMAGE_QL_TYPE, parms);
pQry.execute(parms);
// ----- Get the results and process
DKResults pResults = (DKResults)pQry.result();
dkIterator pIter = pResults.createIterator();
while (pIter.more())
{
    item = (DKDDO)pIter.next();
    // Process the DKDDO
}
dsQBIC.closeCatalog();
dsQBIC.disconnect();
dsQBIC.destroy();
...
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (`SampleIQryQBIC.java`) está disponível no directório `CMBROOT\Samples\java\d1`.

C++

```
DKDatastoreQBIC* dsQBIC;  
dsQBIC = new DKDatastoreQBIC();  
dsQBIC->connect("QBICSRV", "QBICUSER", "PASSWORD");  
dsQBIC->openCatalog("DEMO", "qbic0725");  
DKAny* element;  
DKDDO* item;  
DKString cmd = "QbColor color=<255, 0, 0>";  
dkQuery* pQry = dsQBIC->createQuery(cmd);  
pQry->execute();  
DKAny any = pQry->result();  
DKResults* pResults = (DKResults*)((dkCollection*)any);  
dkIterator* pIter = pResults->createIterator();  
while (pIter->more())  
{  
    element = pIter->next();  
    item = (DKDDO*)element->value();  
    // Process the DKDDO  
    ...  
}  
delete pIter;  
delete pResults;  
delete pQry;  
dsQBIC->closeCatalog();  
dsQBIC->disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TSampleIQryQBIC.cpp) está disponível no directório Cmbroot/Samples/cpp/dl.

Executar uma consulta de imagens a partir do servidor de conteúdos

Como alternativa, pode utilizar a função de execução de DKDatastoreQBIC para executar uma consulta. Os resultados são devolvidos num objecto de dkResultSetCursor. O exemplo seguinte demonstra como executar uma consulta de imagens no servidor de conteúdos. Os resultados são devolvidos num objecto de dkResultSetCursor.

Java

```
// ----- Generate a query string; then create the datastore and connect
String cmd = "QbColorFeatureClass color=<255, 0, 0>";

DKNameValuePair parms[] = null;
DKDDO item = null;
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
dsQBIC.openCatalog("DEMO", "qbic0725");
// ----- Execute the query from the datastore
dkResultSetCursor pCur = dsQBIC.execute(cmd, DK_IMAGE_QL_TYPE, parms);
while (pCur.isValid())
{
    item = pCur.fetchNext();
    ....    // Process the DKDDO
}
pCur.destroy();
dsQBIC.closeCatalog();
dsQBIC.disconnect();
dsQBIC.destroy();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TExecuteQBIC.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreQBIC* dsQBIC;
dsQBIC = new DKDatastoreQBIC();
dsQBIC->connect("QBICSRV", "QBICUSER", "PASSWORD");
cout << "datastore connected" << endl;
dsQBIC->openCatalog("DEMO", "qbic0725");
DKString cmd = "QbColorFeatureClass color=<255, 0, 0>";
dkResultSetCursor* pCur = dsQBIC->execute(cmd);
DKDDO* item = 0;
while (pCur->isValid())
{
    item = pCur->fetchNext();
    if (item != 0)
    {
        // Process the DKDDO
        ...
        delete item;
    }
}
delete pCur;
dsQBIC->closeCatalog();
dsQBIC->disconnect();
```

A aplicação exemplo completa a partir da qual este exemplo foi retirado (TExecuteQBIC.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Avaliar uma consulta de imagens a partir de um servidor de conteúdos

DKDatastoreQBIC também faculta uma função para avaliar uma consulta. O exemplo seguinte demonstra como avaliar uma consulta de imagens a partir do servidor de conteúdos. Os resultados são devolvidos numa recolha de DKResults.

Java

```
// ----- Generate a query string; then create the datastore and connect
String cmd = "QbColorFeatureClass color=<255, 0, 0>";
DKNameValuePair parms[] = null;
DKDDO item = null;
DKDatastoreQBIC dsQBIC = new DKDatastoreQBIC();
dsQBIC.connect("QBICSRV", "QBICUSER", "PASSWORD", "");
dsQBIC.openCatalog("DEMO", "qbic0725");

// ----- Use evaluate to run the query
DKResults pResults=(DKResults) dsQBIC.evaluate(cmd,DK_IMAGE_QL_TYPE,parms);
dkIterator pIter = pResults.createIterator();
while (pIter.more())
{
    item = (DKDDO)pIter.next();
    ...    // Process the DKDDO
}
dsQBIC.closeCatalog();
dsQBIC.disconnect();
dsQBIC.destroy();
```

C++

```
DKDatastoreQBIC* dsQBIC;
dsQBIC = new DKDatastoreQBIC();
dsQBIC->connect("QBICSRV", "QBICUSER", "PASSWORD");
dsQBIC->openCatalog("DEMO", "qbic0725");
DKAny* element;
DKDDO* item;
DKString cmd = "QbColor color=<255, 0, 0>";
DKAny any = dsQBIC->evaluate(cmd);
DKResults* pResults = (DKResults*)((dkCollection*)any);
dkIterator* pIter = pResults->createIterator();
while (pIter->more())
{
    element = pIter->next();
    item = (DKDDO*)element->value();
    // Process the DKDDO
    ...
}
delete pIter;
delete pResults;
dsQBIC->closeCatalog();
dsQBIC->disconnect();
```

Utilizar o motor de pesquisa de imagens

Poderá utilizar o servidor de pesquisa de imagens para especificar uma consulta baseada nas seguintes funções: cor média, percentagem de cor, esquema de cor e texturas. Também poderá especificar várias funções numa consulta. Os resultados da consulta contêm o ID de artigo, o part number, o tipo de representação e as informações de ordenação. Pode utilizar estas informações para criar um XDO para obter os conteúdos de imagens.

Carregar dados para indexação na pesquisa de imagens

Para carregar dados num servidor de Content Manager para indexação através do servidor de pesquisa de imagens, terá de criar uma classe de índices de Content Manager, uma base de dados de pesquisa de imagens e um catálogo de pesquisa

de imagens. A base de dados é uma recolha de catálogos de pesquisa de imagens. Um catálogo contém dados sobre as funções visuais das imagens.

As funções da pesquisa de imagens têm de ser adicionadas ao catálogo para indexação. O utilizador deverá adicionar ao catálogo todas as funções suportadas.

O servidor de pesquisa de imagens tem de estar em execução quando criar uma base de dados e um catálogo de pesquisa de imagens. Certifique-se de que o ambiente está devidamente configurado.

Após o carregamento de dados para o Content Manager, já pode colocar a imagem na fila de imagens. No programa de administração do sistema, seleccione **Processar Fila de Imagens**. Terminada a indexação, já poderá executar pesquisas de imagens.

Indexar um XDO existente através de motores de pesquisa

O utilizador poderá indexar um XDO existente através de um motor de pesquisa especificado. O exemplo seguinte chama a função `setToBeIndexed` da classe `DKBlobDL`.

```
Java
try
{
    // ----- Create the datastore and connect
    DKDatastoreDL dsDL = new DKDatastoreDL();
    dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD","");

    // ----- Create the XDO and PID and set attributes
    DKBlobDL axdo = new DKBlobDL(dsDL);
    DKPidXDODL apid = new DKPidXDODL();
    apid.setPartId(partId);
    apid.setPrimaryId(itemId);
    axdo.setPidObject(apid);

    // ----- Set search engine information
    DKSearchEngineInfoDL aSrchEx = new DKSearchEngineInfoDL();
    aSrchEx.setSearchEngine("SM");
    aSrchEx.setSearchIndex("TM-TMINDEX");
    aSrchEx.setSearchInfo("ENU");
    axdo.setExtension("DKSearchEngineInfoDL", (dkExtension)aSrchEx);
    // ----- Call setToBeIndexed on the XDO
    axdo.setToBeIndexed();

    dsDL.disconnect();
    dsDL.destroy();
}
catch (DKException exc)
{
    ... // Handle the DKException
}
catch (Exception exc)
{
    ... // Handle the Exception
}
```

C++

```
void main(int argc, char *argv[])
{
    DKDatastoreDL dsDL;
    DKString itemId, repType;
    int partId;
    itemId = "N2JJBERBQFK@WTVL";
    repType = "FRN$NULL";
    partId = 10;
    if (argc == 1)
    {
        cout<<"invoke: indexPartxs <partId> <repType> <itemId>"<<endl;
        cout<<" no parameter, following default will be provided:"<<endl;
        cout<<"The supplied default partId = "<<partId<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 2)
    {
        partId = atoi(argv[1]);
        cout<<"you enter: indexPartxs "<<argv[1]<<endl;
        cout<<"The supplied default repType = "<<repType<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 3)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        cout<<"you enter: indexPartxs "<<argv[1]<<" "<<argv[2]<<endl;
        cout<<"The supplied default itemId = "<<itemId<<endl;
    }
    else if (argc == 4)
    {
        partId = atoi(argv[1]);
        repType = DKString(argv[2]);
        itemId = DKString(argv[3]);
        cout<<"you enter: indexPartxs "<<argv[1]<<" "<<argv[2]<<" "<<argv[3]<<endl;
    }

    cout << "connecting Datastore" << endl;
    try
    {
        //replace following with your library server, userid, password
        dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");
        cout << "datastore connected" << endl;

        DKBlobDL* axdo = new DKBlobDL(&dsDL);
        DKPidXDODL* apid = new DKPidXDODL;
        apid ->setPartId(partId);
        apid ->setId(itemId);
        axdo ->setPid(apid);
        axdo ->setRepType(repType);
        cout<<"itemId= "<<(axdo->getPid())->getId()<<endl;
        cout<<"partId= "<<((DKPidXDODL*)(axdo->getPid()))->getPartId()<<endl;
        cout<<"repType= "<<axdo->getRepType()<<endl;

        // continued...
```

C++ (continuação)

```
//--- set searchEngine -----
cout<<"set search engine and setToBeIndexed()"<<endl;
DKSearchEngineInfoDL aSrchEx;
aSrchEx.setSearchEngine("SM");
aSrchEx.setSearchIndex("TM-TMINDEX");
aSrchEx.setSearchInfo("ENU");
axdo->setExtension("DKSearchEngineInfoDL", (dkExtension*)&aSrchEx);
axdo->setToBeIndexed();
cout<<"setToBeIndexed() done..."<<endl;

delete apid;
delete axdo;
dsDL.disconnect();
cout<<"datastore disconnected"<<endl;
}
catch(DKException &exc)
{
    cout << "Error id" << exc.errorId() << endl;
    cout << "Exception id " << exc.exceptionId() << endl;
    for(unsigned long i=0;i< exc.textCount();i++)
    {
        cout << "Error text:" << exc.text(i) << endl;
    }
    for (unsigned long g=0;g< exc.locationCount();g++)
    {
        const DKExceptionLocation* p = exc.locationAtIndex(g);
        cout << "Filename: " << p->fileName() << endl;
        cout << "Function: " << p->functionName() << endl;
        cout << "LineNumber: " << p->lineNumber() << endl;
    }
    cout << "Exception Class Name: " << exc.name() << endl;
}
cout << "done ..." << endl;
}
```

Utilizar consulta combinada

Utilize uma *consulta combinada* para executar uma combinação de consultas paramétricas e de texto, com ou sem um âmbito. Um *âmbito* é um objecto de DKResults que é formado a partir de uma consulta paramétrica ou de texto anterior. O resultado é uma intersecção entre os âmbitos e os resultados de cada consulta. Deste modo, se o utilizador não prestar atenção ao formular a consulta e incluir âmbitos, os resultados da consulta individual podem não sofrer uma intersecção, logo o resultado da consulta combinada é vazio.

Caso exista pelo menos uma consulta paramétrica e outra de texto, o DDO daí resultante tem o atributo DKRANK, o que indica a ordem mais elevada da parte correspondente que pertence ao documento.

Restrição: Para cada consulta numa consulta combinada, terá de utilizar uma ligação diferente ao motor de pesquisa, não poderá encaminhar várias consultas através da mesma ligação.

Consultas paramétricas e de texto combinadas

Para executar uma consulta combinada constituída por uma consulta paramétrica e uma de texto, sem âmbito, terá de criar um objecto de consulta combinada e passar as duas consultas como parâmetros de input que vão ser executados pela consulta combinada. Por exemplo:

Java

```
// ----- Create a pré-Versão 8.1 Content Manager datastore and connect
DKDatastoreDL dsDL = new DKDatastoreDL();
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
// ----- Create a text search datastore and connect
DKDatastoreTS dsTS;
dsTS.connect("TM", "", ' '); // TM is a local alias for
...                          // the Motor de Pesquisa de Texto server

// ----- Generate the parametric query string and create the query
String pquery = "SEARCH=(INDEX_CLASS=GRANDPA, COND=(DLSEARCH_Date > 1994));";
DKParametricQuery pq =
    (DKParametricQuery) dsDL.createQuery(pquery, DK_CM_PARAMETRIC_QL_TYPE, null);

// ----- Generate the text query string and create the query
String tquery = "SEARCH=(COND=(Tivoli)); OPTION=(SEARCH_INDEX=TMINDEX)";
DKTextQuery tq =
    (DKTextQuery) dsTS.createQuery(tquery, DK_CM_TEXT_QL_TYPE, null);

// ----- Create the combined query
DKCombinedQuery cq = new DKCombinedQuery();

// ----- Package the queries in DKNVPair as input parameters
DKNVPair par[] = new DKNVPair[3];
par[0].set(DK_PARM_QUERY, pq);
par[1].set(DK_TEXT_QUERY, tq);
par[2].setName(DK_PARM_END); // to signal the end of parameter list

// ----- Execute the combined query
cq.execute(par);

// ----- Get the results
DKResults res = (DKResults) cq.result();
if (res != null) {
    ... // process the results
}
```


C++

```
DKDatastoreDL dsDL;
dsDL.connect("LIBSRVRN","FRNADMIN","PASSWORD");

DKDatastoreTS dsTS;
// TM is a local alias for the Motor de Pesquisa de Texto server
dsTS.connect("TM",""," ' ');
// create a parametric query
DKString pquery="SEARCH=(INDEX_CLASS=GRANDPA,COND=(DLSEARCH_Date > 1994));";
DKParametricQuery* pq =
    (DKParametricQuery*) dsDL.createQuery(pquery,DK_PARAMETRIC_QL_TYPE, NULL);

// create a text query
DKString tquery = "SEARCH=(COND=(Tivoli));OPTION=(SEARCH_INDEX=TMINDEX)";
DKTextQuery* tq =
    (DKTextQuery*) dsTS.createQuery(tquery,DK_TEXT_QL_TYPE, NULL);

// create a combined query
DKCombinedQuery* cq = new DKCombinedQuery();

// package the queries in DKNVPair as input parameters
DKNVPair par[3];
par[0].set(DK_PARM_QUERY, pq);
par[1].set(DK_TEXT_QUERY, tq);
// to signal the end of parameter list
par[2].setName(DK_PARM_END);

// execute the combined query
cq->execute(par);

// get the results
DKAny any = cq->result();
DKResults* res = (DKResults*) any.value();
if (res != NULL) {
    // process the results
    ...
}
```

A última instrução `if` é necessária para assegurar que o objecto de `DKResults` não é nulo.

Utilizar um âmbito

Se pretender utilizar um objecto de `DKResults` como âmbito, transmita-o como um parâmetro adicional de consulta. O exemplo seguinte ilustra a utilização de um objecto de `DKResults` para funcionar como âmbito de uma consulta combinada:

Java

```
// ----- This scope is the result of a parametric query
DKResults scope;
// ----- This scope is the result of a previous text query
DKResults tscope;

// ----- Package the query in array of DKNVPairs as input parameters
DKNVPair par[] = new DKNVPair[4];
par[0].set(DK_PARM_QUERY, pq);
par[1].set(DK_TEXT_QUERY, tq);
par[2].set(DK_SCOPE_DL, scope);
par[3].set(DK_SCOPE_TS, tscope);
par[4].setName(DK_PARM_END);

// ----- Execute the combined query
cq.execute(par);
....
```

C++

```
DKResults* scope;    // assume that this is the scope
                     // initialized somewhere as a result of
                     // some parametric query
DKResults* tscope    // assume that this is the scope
                     // initialized somewhere as a result of
                     // some text query

...
// package the query in DKNVPair as input parameters
DKNVPair par[4];
par[0].set(DK_PARM_QUERY, pq);
par[1].set(DK_TEXT_QUERY, tq);
par[2].set(DK_SCOPE_DL, scope);
par[3].set(DK_SCOPE_TS, tscope);
par[4].setName(DK_PARM_END);
// execute the combined query
cq->execute(par);
...
```

Os resultados de uma consulta combinada também podem ser utilizados como âmbito de outra consulta combinada, podendo por vezes o utilizador consultar os resultados.

Ordenar

Caso a consulta combinada contenha pelo menos uma consulta de texto, então o DDO resultante tem o atributo DKRANK. Este atributo não é armazenado, mas sim informatizado sempre pelo Motor de Pesquisa de Texto. O valor da ordem corresponde à ordem mais elevada da parte no documento que preenche as condições da consulta de texto.

Sugestões

Caso tenha várias consultas e vários âmbitos paramétricos, é mais eficiente executar uma consulta completa. Isto também se aplica a consultas de texto.

A opção de consulta "MAX_RESULTS=nn" limita o número de resultados obtidos. Normalmente, esta opção aplica-se mais a consultas de texto, visto que o resultado

é armazenado por ordem decrescente. Por exemplo, caso esta opção esteja definida como 10, significa que o chamador apenas pretende os 10 resultados com correspondência mais elevada.

O significado da opção de consulta "MAX_RESULTS=nn" é diferente para consultas paramétricas. Visto que não existe a noção de ordem, o chamador irá obter os 10 primeiros resultados. Os resultados irão sofrer uma intersecção com o resultados da consulta de texto. Deste modo, quando combinar consultas paramétricas e de texto, é aconselhável que não especifique a opção de consulta "MAX_RESULTS=nn" na consulta paramétrica.

Compreender as funções de fluxo de trabalho e de cesto de trabalho do Content Manager anterior

Esta secção descreve as funções de fluxo de trabalho e de cesto de trabalho da versão anterior do Content Manager.

Compreender o serviço do fluxo de trabalho do Content Manager anterior

Um *cesto de trabalho* é um contentor que contém documentos e pastas que o utilizador pretende processar. Um *fluxo de trabalho* é um conjunto ordenado de cestos de trabalho que representam um processo específico empresarial. As pastas e documentos movem-se entre os cestos de trabalho num fluxo de trabalho, permitindo que as aplicações criem modelos empresariais simples e encaminhem trabalho através do processo até este terminar.

O modelo do fluxo de trabalho no Content Manager segue estas regras:

- Um cesto de trabalho não necessita de se localizar num fluxo de trabalho.
- Um cesto de trabalho pode localizar-se em um ou mais do que um fluxo de trabalho.
- Um cesto de trabalho pode pertencer aos mesmo fluxo de trabalho mais do que uma vez.
- Um documento ou pasta apenas só podem ser armazenados num fluxo de trabalho de cada vez.
- Um documento ou pasta podem ser armazenados num cesto de trabalho que não faz parte de fluxo de trabalho.

As APIs de Enterprise Information Portal fornecem classes para trabalhar com o fluxo de trabalho do Content Manager.

A classe de DKWorkflowServiceDL representa o serviço de fluxo de trabalho do Content Manager. Esta classe fornece a possibilidade de iniciar, alterar, remover e completar um documento ou pasta num fluxo de trabalho. Além disso, a classe de DKWorkflowServiceDL permite-lhe obter informações sobre cestos de trabalho e fluxos de trabalho.

As classes DKWorkflowDL e DKWorkBasketDL são as representações orientadas para objectos de um artigo de fluxo de trabalho e de um artigo de cesto de trabalho, respectivamente.

Estabelecer uma ligação

O utilizador terá de estabelecer uma ligação ao servidor de Content Manager antes de poder utilizar o serviço de fluxo de trabalho. O servidor de conteúdos fornece as funções de ligação e de desligação.

O exemplo que se segue demonstra como se deve ligar a um servidor de Content Manager denominado LIBSRVRN, através do ID de utilizador FRNADMIN e da palavra-passe PASSWORD.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
... // Processar conforme for apropriado
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual foi retirada este exemplo (TListWorkFlowWFS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKWorkflowServiceDL wfDL(&dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
... // do some work
dsDL.disconnect();
```

A aplicação exemplo completa da qual foi retirada este exemplo (TListWorkFlowWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Criar um fluxo de trabalho

Utilize o DKWorkflowServiceDL para criar um fluxo de trabalho. Para o fazer, conclua normalmente as seis etapas seguintes:

1. Crie uma ocorrência de DKWorkflowDL.
2. Defina o nome do fluxo de trabalho ("GOLF").
3. Defina a sequência do cesto de trabalho ("NULL") para indicar que este fluxo de trabalho não contém cestos de trabalho.
4. Defina o privilégio ("All Privileges").
5. Defina a disposição (DK_WF_SAVE_HISTORY).
6. Chame a função de adição add ().

O exemplo segue as seis etapas de criação de um fluxo de trabalho.

Java

```
// ----- Create the datastore and the CM workflow services
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);

// ----- Set the access option and connect
Object input_option = new Integer(DK_SS_CONFIG);
dsDL.setOption(DK_OPT_DL_ACCESS, input_option);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");

// ----- Create the CM workflow
DKWorkflowDL newwf = new DKWorkflowDL(wfDL);
newwf.setName("Process claim");
newwf.setWorkBasketSequence((dkCollection *)NULL);
newwf.setAccessList("All Privileges");
newwf.setHistoryDisposition(DK_WF_SAVE_HISTORY);
newwf.add();
... // Processing as appropriate
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TCreateDelWorkflow.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKAny input_option = DK_SS_CONFIG;
DKWorkflowServiceDL wfDL(&dsDL);
dsDL.setOption(DK_DL_OPT_ACCESS, input_option);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKWorkflowDL * newwf = new DKWorkflowDL(&wfDL);
newwf->setName("GOLF");
newwf->setWorkBasketSequence((dkCollection *)NULL);
newwf->setAccessList("All Privileges");
newwf->setHistoryDisposition(DK_WF_SAVE_HISTORY);
newwf->add();
... // do some work
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TCreateDelWorkflowWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Importante: Se estabelecer ligação ao servidor de conteúdos como um utilizador normal (DK_SS_NORMAL), não obtém o fluxo de trabalho definido após estabelecer ligação. Deste modo, este exemplo utiliza DK_SS_CONFIG.

Enumerar fluxos de trabalho

O DKWorkflowServiceDL fornece uma função de listagem dos fluxos de trabalho no sistema, como está demonstrado no exemplo seguinte: A lista é devolvida numa recolha sequencial de objectos de DKWorkflowDL.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
// ----- Get a list of the CM workflows
DKSequentialCollection wfList=(DKSequentialCollection)wfDL.listWorkFlows();
if (wfList != null)
{
    dkIterator pIter = wfList.createIterator();
    DKWorkflowDL pwf1;
    while (pIter.more())
    {
        pwf1 = (DKWorkflowDL)pIter.next();
        pwf1->retrieve();
        ...                // Process as appropriate
    }
}
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListWorkFlowWFS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKWorkflowServiceDL wfDL(&dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKSequentialCollection * wfList1 =
    (DKSequentialCollection *)wfDL.listWorkFlows();
if (wfList1 != NULL)
{
    dkIterator * pIter1 = wfList1->createIterator();
    DKWorkflowDL * pwf1;
    while (pIter->more())
    {
        pwf1 = (DKWorkflowDL *)((void*)(*pIter1->next()));
        pwf1->retrieve();
        ...                // do some work
        delete pwf1;
    }
}
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListWorkFlowWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Criar um cesto de trabalho do Content Manager

Utilize o DKWorkflowServiceDL para criar um cesto de trabalho. Para o fazer, conclua normalmente as etapas seguintes:

1. Crie uma instância de DKWorkBasketDL.
2. Defina o nome do cesto de trabalho (Hot Items).
3. Defina o privilégio (All Privileges).
4. Chame a função de adição.

O exemplo seguinte utiliza estas etapas para criar um cesto de trabalho. Se estabelecer ligação ao servidor de conteúdos como um utilizador normal (DK_SS_NORMAL), não obtém o cesto de trabalho definido após estabelecer ligação. Deste modo, este exemplo utiliza DK_SS_CONFIG.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
Object input_option = new Integer(DK_SS_CONFIG);
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
dsDL.setOption(DK_OPT_DL_ACCESS, input_option);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
// ----- Create the CM workbasket and set properties
DKWorkBasketDL newwb = new DKWorkBasketDL(wfDL);
newwb.setName("Hot Items");
newwb.setAccessList("All Privileges");
newwb.add();
... // Process as appropriate
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TCreateDelWorkBasket.java) está disponível no directório CMBROOT\Samples\java\dl.

C++

```
DKDatastoreDL dsDL;
DKAny input_option = DK_SS_CONFIG;
DKWorkflowServiceDL wfDL(&dsDL);
dsDL.setOption(DK_DL_OPT_ACCESS, input_option);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKWorkBasketDL * newwb = new DKWorkBasketDL(&wfDL);
newwb->setName("Hot Items");
newwb->setAccessList("All Privileges");
newwb->add();
... // do some work
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TCreateDelWorkBasket.cpp) está disponível no directório CMBROOT\Samples\cpp\dl.

Enumerar cestos de trabalho

O DKWorkflowServiceDL fornece uma função de listagem de cestos de trabalho no sistema, como está demonstrado no exemplo seguinte. A lista é devolvida numa recolha sequencial de objectos de DKWorkBasketDL.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKSequentialCollection wbList=(DKSequentialCollection)wfDL.listWorkBaskets();
if (wbList != null)
{
    dkIterator pIter = wbList.createIterator();
    DKWorkBasketDL pwbl;
    while (pIter.hasMore())
    {
        pwbl = (DKWorkBasketDL)pIter.next();
        pwbl->retrieve();
        ... // do some work
    }
}
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListWorkBasketWFS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKWorkflowServiceDL wfDL(&dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKSequentialCollection * wfList1 =
    (DKSequentialCollection *)wfDL.listWorkBaskets();
if (wfList1 != NULL)
{
    dkIterator * pIter1 = wfList1->createIterator();
    DKWorkBasketDL * pwbl;
    while (pIter1->more())
    {
        pwbl = (DKWorkBasketDL *)((void*)(*pIter1->next()));
        pwbl->retrieve();
        ... // do some work
        delete pwbl;
    }
}
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListWorkBasketWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Enumerar artigos num fluxo de trabalho do Content Manager anterior

O DKWorkflowServiceDL fornece uma função de listagem dos IDs dos artigos num fluxo de trabalho, como está demonstrado no exemplo seguinte. A lista é devolvida numa recolha sequencial de objectos de DKString.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
// ----- Get the list of CM workflows
KSequentialCollection wfList = (DKSequentialCollection)wfDL.listWorkFlows();
if (wfList != null)
{
    dkIterator pIter = wfList.createIterator();
    while (pIter.more())
    {
        DKWorkflowDL pwf1 = (DKWorkflowDL)pIter.next();
        // ----- Get the list of items in the CM workflow
        KSequentialCollection itemList=(DKSequentialCollection)pwf1.listItemIDs();
        if (itemList != null)
        {
            dkIterator iter1 = itemList.createIterator();
            String itemid;
            while (iter1.more())
            {
                itemid = (String)iter1.next();
                // ----- Process the items using the item ID
            }
        }
    }
}
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListItemsWFS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKWorkflowServiceDL wfDL(&dsDL);
DKString itemIDWF = DKString("HI7MOPALUPFQ1U47");
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
DKWorkflowDL * wf = new DKWorkflowDL(&wfDL, (char *)itemIDWF);
wf->retrieve;
DKSequentialCollection * pColDoc1 =
    (DKSequentialCollection *)wf->listItemIDs();
if (pColDoc1 != NULL)
{
    dkIterator* pIterDoc1 = pColDoc1->createIterator();
    DKString DocID1;
    while (pIterDoc1->more() == TRUE)
    {
        DocID1 = (DKString)(*pIterDoc1->next());
        ... // do some work
    }
}
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListItemsWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Executar um fluxo de trabalho do Content Manager anterior

O DKWorkflowServiceDL fornece funções de execução de um fluxo de trabalho. O exemplo seguinte demonstra como iniciar um artigo num fluxo de trabalho, como

encaminhar um artigo para um cesto de trabalho e como completar um artigo num fluxo de trabalho. Para utilizar este exemplo tem de o modificar para:

- Utilizar um ID de artigo válido em vez de EP8L80R9MHH##QES.
- utilizar um ID de fluxo de trabalho válido em vez de HI7MOPALUPFQ1U47.
- Utilizar um ID de cesto de trabalho válido em vez de E3PP1UZ0ZUFQ1U3M.

Java

```
DKDatastoreDL dsDL = new DKDatastoreDL();
DKWorkflowServiceDL wfDL = new DKWorkflowServiceDL(dsDL);
DKString itemID = new String("EP8L80R9MHH##QES");
DKString itemIDWF = new String("HI7MOPALUPFQ1U47");
DKString itemIDWB = new String("E3PP1UZ0ZUFQ1U3M");
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
wfDL.startWorkflowItem(itemID,           // itemID
                      itemIDWF,         // itemIDWB
                      NULL,              // default (1st workbasket)
                      TRUE,              // overload
                      DK_WIP_DEFAULT_PRIORITY // initial_priority
                      );
...
wfDL.routeWipItem(itemID,           // itemID
                  itemIDWF,         // itemIDWB
                  TRUE,              // overload
                  DK_NO_PRIORITY_CHANGE // initial_priority
                  );
...
wfDL.completeWorkflowItem(itemID);
dsDL.disconnect();
dsDL.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TProcessWFS.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDL dsDL;
DKWorkflowServiceDL wfDL(&dsDL);
DKString itemID = DKString("EP8L80R9MHH##QES");
DKString itemIDWF = DKString("HI7MOPALUPFQ1U47");
DKString itemIDWB = DKString("E3PP1UZ0ZUFQ1U3M");
dsDL.connect("LIBSRVRN", "FRNADMIN", "PASSWORD");
wfDL.startWorkflowItem(itemID,           // itemID
                      itemIDWF,         // itemIDWB
                      NULL,              // default(1st workbasket)
                      TRUE,              // overload
                      DK_WIP_DEFAULT_PRIORITY // initial_priority
                      );
...
wfDL.routeWipItem(itemID,           // itemID
                  itemIDWF,         // itemIDWB
                  TRUE,              // overload
                  DK_NO_PRIORITY_CHANGE // initial_priority
                  );
...
wfDL.completeWorkflowItem(itemID);
dsDL.disconnect();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TProcessWFS.cpp) está disponível no directório Cmbroot/Samples/cpp/d1.

Trabalhar com OnDemand

O Enterprise Information Portal fornece um conector e classes relacionadas para aceder ao conteúdo nos servidores Content Manager OnDemand. As classes e as APIs OnDemand permitem-lhe:

- Ligar a e desligar de servidores de OnDemand.
- Enumerar grupos de aplicações e campos de grupos de aplicações.
- Enumerar pastas de OnDemand.
- Consultar um grupo de aplicações.
- Pesquisar e obter documentos utilizando a pasta ou a interface de grupo de aplicações.
- Tratar pastas de OnDemand como entidades nativas em pesquisas associadas.
- Pesquisar síncronamente e assincronamente num grupo de aplicações ou em modo de pasta.
- Obter documentos completos ou segmentos de documentos de OnDemand.
- Obter os dados de vista lógica de um determinado documento de OnDemand.
- Obter o grupo de recursos para um determinado documento de OnDemand.
- Obter dados de anotação para um determinado documento de OnDemand.
- Criar e modificar anotações

Restrição: O OnDemand não suporta Motor de Pesquisa de Texto e pesquisa QBIC ou Consulta combinada.

Representar servidores e documentos OnDemand

O utilizador irá representar um servidor de conteúdos do Content Manager OnDemand através de DKDatastoreOD numa aplicação do Enterprise Information Portal e irá representar um documento OnDemand como um DDO que utiliza um DKDDO. Os DDOs OnDemand contêm as seguintes informações:

- Nomes dos atributos dos documentos e os seus valores
- Dados e anotações de documentos (representados como DKParts)
- Recolha de vistas lógicas de um documento
- Dados de grupo de recursos

Os atributos de um documento OnDemand são armazenados num DKDDO como propriedades. Os segmentos e notas de um documento OnDemand são armazenados como DKParts.

Todos os outros dados de documentos (grupo e vistas de recursos, tanto fixos como lógicos), são armazenados como propriedades especiais num DDO OnDemand, em que os seguintes nomes de propriedades estão reservados para o OnDemand:

DKViews

Uma recolha de vistas lógicas

DKFixedView

Contém informações de vista fixa

DKResource

Contém dados do grupo de recursos

Notas:

1. Um administrador do Enterprise Information Portal tem de definir devidamente o conector OnDemand, especificando a cadeia no campo **Parâmetros Adicionais** na página **Parâmetros de Iniciação**. A cadeia deverá ter o aspecto seguinte: ENTITY_TYPE=TEMPLATES;; Certifique-se de que coloca ponto e vírgula duas vezes.
2. No Enterprise Information Portal Versão 8.2, DKViews, DKFixedView e DKResource substituem DKViewDataOD, DKFixedViewDataOD e DKResouceGrpOD, respectivamente.

Estabelecer e anular ligações ao servidor de OnDemand

Para iniciar uma sessão no servidor de conteúdos de OnDemand, transmita o nome do servidor (por exemplo, ODServer.mycompany.com), o ID de utilizador e a palavra-passe através do método de ligação.

Java

```
DKDatastoreOD dsOD = new DKDatastoreOD();
    System.out.println("a ligar ao armazenamento de dados...");
dsOD.connect(ODServer, UserID, Password, "");
```

C++

```
DKDatastoreOD* dsOD = new DKDatastoreOD();
cout << "connecting to datastore ..." << endl;
dsOD->connect(ODServer, UserID, Password, "");
```

Utilize o método de anulação de ligação para terminar uma sessão no servidor de OnDemand.

Java

```
System.out.println("disconnecting from the datastore ...");
dsOD.disconnect();
dsOD.destroy(); // Finished with the datastore
```

C++

```
cout << "disconnecting from the datastore ..." << endl;
dsOD->disconnect();
delete dsOD;
```

Enumerar informações em OnDemand

O utilizador poderá enumerar grupos e pastas de aplicações para servidores OnDemand.

Enumerar grupos de aplicações

O utilizador pode enumerar grupos de aplicações em OnDemand através do método listEntities() de DKDatastoreOD. O exemplo seguinte ilustra como utilizar este método.

Java

```
...
pCol = (DKSequentialCollection) dsOD.listEntities(); //get application groups
pIter = pCol.createIterator();
i = 0;
while (pIter.more() == true)
{
    i++;
    agDef = (DKAppGrpDefOD)pIter.next();
    strAppGrp = agDef.getName();
    System.out.println("  app grp name[" + i + "]: " + strAppGrp);
    System.out.println("  show attributes for " + strAppGrp + " app grp - ");
}
...
```

C++

```
// ----- Show the application groups
// ----- First get the groups
pCol = (DKSequentialCollection*)dsOD.listEntities();
pIter = pCol->createIterator();
int i = 0;
// ---- Process the list
while (pIter->more() == TRUE)
{
    i++;
    agDef = (DKAppGrpDefOD*)((void*)(*pIter->next()));
    strAppGrp = agDef->getName();
    cout << "  app group name[" << i << "]: ==>" << strAppGrp << endl;
}
...

```

O exemplo seguinte ilustra a obtenção de informações de atributos para cada grupo de aplicações:

Java

```
...
pCol2 = (DKSequentialCollection) dsOD.listEntityAttrs(strAppGrp);
pIter2 = pCol2.createIterator();
j = 0;

while (pIter2.more() == true)
{
    j++;
    attrDef = (DKFieldDefOD)pIter2.next();
    System.out.println("    Attribute name[" + j + "]: " + attrDef.getName());
    System.out.println("        datastoreType: " + attrDef.datastoreType());
    System.out.println("        attributeOf: " + attrDef.getEntityName());
    System.out.println("        type: " + attrDef.getType());
    System.out.println("        size: " + attrDef.getSize());
    System.out.println("        id: " + attrDef.getId());
    System.out.println("        nullable: " + attrDef.isNullable());
    System.out.println("        precision: " + attrDef.getPrecision());
    System.out.println("        scale: " + attrDef.getScale());
    System.out.println("        stringType: " + attrDef.getStringType());
}

System.out.println(" " + j + " attribute(s) listed for " +
                    strAppGrp + " app grp\n");
...
```

C++

```
// ----- Get the attributes for each of the entities(application groups)
pCol2 = (DKSequentialCollection*)dsOD.listEntityAttrs(strAppGrp);
pIter2 = pCol2->createIterator();
int j = 0;
// ----- List the attributes
while (pIter2->more() == TRUE)
{
    j++;
    attrDef = (DKFieldDefOD*)(void*)(*pIter2->next());
    cout << "attribute name[" << j << "]: ==>" << attrDef->getName() << endl;
    cout << "        datastore type: " << attrDef->datastoreType() << endl;
    cout << "        attribute of: " << attrDef->getEntityName() << endl;
    cout << "        type: " << attrDef->getType() << endl;
    cout << "        size: " << attrDef->getSize() << endl;
    cout << "        ID: " << attrDef->getId() << endl;
    cout << "        precision: " << attrDef->getPrecision() << endl;
    cout << "        scale: " << attrDef->getScale() << endl;
    cout << "        stringType: " << attrDef->getStringType() << endl;
    cout << "        nULlable: " << attrDef->isNullable() << endl;
    cout << "        queryable: " << attrDef->isQueryable() << endl;
    cout << "        updatable: " << attrDef->isUpdatable() << endl;
    // ----- Clean up the attribute
    delete attrDef;
}
cout << " " << j << " attribute(s) listed for the " << strAppGrp
      << " app group\n" << endl;
// ----- Clean up the iterators and collections
if ( pIter2 )
    delete pIter2;
if ( pCol2 )
    delete pCol2;
...
```

Enumerar pastas de OnDemand

Para obter uma lista de pastas num servidor de conteúdos de OnDemand, utiliza a função `listSearchTemplates()`.

Java

```
...
dsDef = (DKDatastoreDefOD)dsOD.datastoreDef();
pCol = (DKSequentialCollection) dsDef.listSearchTemplates();
pIter = pCol.createIterator();
i = 0;
while (pIter.more() == true)
{
    i++;
    folderName = (String)pIter.next();
    .... // Process the folder as appropriate
}
dsOD.disconnect();
dsOD.destroy();
```

C++

```
. . .
// ----- List the folders
dsDef = (DKDatastoreDefOD*)dsOD.datastoreDef();
pCol = (DKSequentialCollection*)dsDef->listSearchTemplates();
pIter = pCol->createIterator();
i = 0;
// ----- Process the list of folders
while (pIter->more() == TRUE)
{
    i++;
    folderName = (DKString)(*pIter->next());
    cout << "folder name [" << i << "] - " << folderName << endl;
}
// ----- Disconnect
dsOD.disconnect();
. . .
```

Obter um documento OnDemand

Num servidor de OnDemand, pode obter documentos. Também pode apresentar documentos com as suas partes e atributos.

Pesquisar um documento específico

O exemplo seguinte pesquisa um grupo de aplicações (OnDemand Publications) no servidor de conteúdos de OnDemand.

Java

```
DKDatastoreOD dsOD = new DKDatastoreOD();
String appgrp = "OnDemand Publications";
String SQLcmd = "where bookname LIKE 'A%'";

DKNVPair[] parms = new DKNVPair[3];
parms[0] = new DKNVPair("APPL_GROUP", appgrp);
parms[1] = new DKNVPair("MAX_RESULTS", new String(Integer.toString(5)));
parms[2] = new DKNVPair("CONTENT", new String("ATTRONLY"));

System.out.println("executing query");
dkResultSetCursor pCur = dsOD.execute(SQLcmd,DK_CM_SQL_QL_TYPE,parms);
System.out.println("datastore executed query");
```

C++

```
DKDatastoreOD* dsOD = new DKDatastoreOD();

cout << "connecting to datastore ..." << endl;
dsOD->connect(ODServer, UserID, Password, "");

DKString appgrp = "OnDemand Publications";
DKString SQLcmd = "where bookname LIKE 'A%'";

DKNVPair parms[4];
parms[0] = DKNVPair("APPL_GROUP", appgrp);
parms[1] = DKNVPair("MAX_RESULTS", DKString(5));
parms[2] = DKNVPair("CONTENT", DKString("ATTRONLY"));
parms[3] = DKNVPair( DK_CM_PARM_END, DKAny((long)0) );

cout << "executing query" << endl;
dkResultSetCursor* pCur = dsOD->execute(SQLcmd,DK_CM_SQL_QL_TYPE,parms);
cout << "datastore executed query" << endl;
if (pCur != 0)
    delete pCur;
```

O exemplo seguinte pesquisa um grupo de aplicações (OnDemand Publications) e obtém os documentos devolvidos.

Java

```
DKDatastoreOD dsOD = new DKDatastoreOD();
String appgrp = "OnDemand Publications";
String SQLcmd = "where bookname LIKE 'A%'";

DKNVPair[] parms = new DKNVPair[3];
parms[0] = new DKNVPair("APPL_GROUP", appgrp);
parms[1] = new DKNVPair("MAX_RESULTS", new String(Integer.toString(5)));
parms[2] = new DKNVPair("CONTENT", new String("ATTRONLY"));

System.out.println("executing query");
dkResultSetCursor pCur = dsOD.execute(SQLcmd,DK_CM_SQL_QL_TYPE,parms);
System.out.println("datastore executed query");

while (pCur.isValid()) {
    DKDDO p = pCur.fetchNext();
    if (p != null)
    {
        String idstr = ((DKPid)p.getPidObject()).pidString();
        System.out.println(" pidString : " + idstr);
        DKPid pid = new DKPid (idstr);
        DKDDO ddoold = p;
        short id, docType = 0;
        if ((id = ddoold.propertyId(DK_CM_PROPERTY_ITEM_TYPE)) > 0)
            docType = ((Short)ddoold.getProperty(id)).shortValue();
        if (docType == DK_CM_DOCUMENT)
        {
            System.out.println("create a new DDO with a cloned pid to retrieve!");
            p = dsOD.createDDO(ddoold.getObjectType(), DK_CM_DOCUMENT);
            p.setPidObject(pid);
            try
            {
                dsOD.retrieveObject((dkDataObject)p);
            }
            catch (DKException exc)
            {
                System.out.println("Exception name " + exc.name());
                System.out.println("Exception message " + exc.getMessage());
                System.out.println("Exception error code " + exc.errorCode());
                System.out.println("Exception error state " + exc.errorState());
                exc.printStackTrace();
            }
        }
    }
}
pCur.destroy(); // Finished with the cursor
```

C++

```
DKDatastoreOD* dsOD = new DKDatastoreOD();
DKString appgrp = "OnDemand Publications";
DKString SQLcmd = "where bookname LIKE 'A%'";

DKNVPair parms[4];
parms[0] = DKNVPair("APPL_GROUP", appgrp);
parms[1] = DKNVPair("MAX_RESULTS", DKString(5));
parms[2] = DKNVPair("CONTENT", DKString("ATTRONLY"));
parms[3] = DKNVPair( DK_CM_PARM_END, DKAny((long)0) );

cout << "executing query" << endl;
dkResultSetCursor* pCur = dsOD->execute(SQLcmd,DK_CM_SQL_QL_TYPE,parms);
cout << "datastore executed query" << endl;

if (pCur != 0)
{
    while (pCur->isValid())
    {
        DKDDO* p = pCur->fetchNext();
        DKDDO* ddoold = p;
        DKString pidStr = ((DKPid*)ddoold->getPidObject())->pidString();
        DKPid* pid = new DKPid(pidStr);
        short id, docType = 0;
        DKAny a;
        if ((id = ddoold->propertyId(DK_CM_PROPERTY_ITEM_TYPE)) > 0)
        {
            a = ddoold->getProperty(id);
            if (a.typeCode() == DKAny::tc_ushort)
                docType = (short)(USHORT)a;
        }
        else
            docType = a;
        if (docType == DK_CM_DOCUMENT)
        {
            cout << "create the DDO from the pidstring..." << endl;
            p = dsOD->createDDO(ddoold->getObjectType(), DK_CM_DOCUMENT);
            p->setPidObject(pid);

            dsOD->retrieveObject((dkDataObject*)p);
        }
        delete pid;
        delete ddoold;
    }
    delete pCur;
}
```

Apresentar documentos e as suas partes e atributos

O exemplo seguinte apresenta os documentos localizados pela consulta com as suas partes e atributos.

Java

```
//-----For each data item, get the attributes
//-----numDataItems is the number of data items
for (j = 1; j <= numDataItems; j++)
{
    a = p.getData(j)
    strDataName = p.getDataName(j);
    System.out.println("    " + j + ". Attribute Name: " + strDataName );
    System.out.println("        type: " + p.getDataPropertyByName
        (j,DK_PROPERTY_TYPE));

    System.out.println("        nullable: " +
        p.getDataPropertyByName (j,DK_PROPERTY_NULLABLE));
    if (strDataName.equals(DKPARTS) == false &&
        strDataName.equals("DKResource") == false &&
        strDataName.equals("DKViews") == false &&
        strDataName.equals("DKLargeObject") == false &&
        strDataName.equals("DKFixedView") == false &&
        strDataName.equals("DKAnnotations") == false)
    {
        System.out.println("        attribute id: " +
            p.getDataPropertyByName(j,DK_PROPERTY_ATTRIBUTE_ID));
    }
    //-----Check for the type of the attribute
    if (a instanceof String)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof Integer)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof Short)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof DKDate)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof DKTime)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof DKTimestamp)
    {
        System.out.println("        Attribute Value: " + a);
    }
    else if (a instanceof dkCollection)
    {
        System.out.println("        Attribute Value is collection");
        pCol = (dkCollection)a;
        pIter = pCol.createIterator();
        i = 0;
        while (pIter.more() == true)
        {
            i++;
            a = pIter.next();
            pDO = (dkDataObjectBase)a;
        }
    }
}

// continued...
```

Java (continuação)

```
if (pD0.protocol() == DK_XD0)
{
    System.out.println("      dkXD0 object " + i + " in collection");
    pXD0 = (dkXD0)pD0;
    DKPidXD0 pid2 = pXD0.getPidObject();
    System.out.println("          XD0 pid string: " +
                      pid2.pidString());
    //----- Retrieve and open instance handler for an XD0
    pXD0.retrieve();
    // pXD0.open();
}
}
}
else if (a != null)
{
    System.out.println("      Attribute Value: " + a.toString());
    if (strDataName.equals("DKResource") ||
        strDataName.equals("DKFixedView") ||
        strDataName.equals("DKLargeObject"))
    {
        pD0 = (dkDataObjectBase)a;

        if (pD0.protocol() == DK_XD0)
        {
            System.out.println("          dkXD0 object ");
            pXD0 = (dkXD0)pD0;
            DKPidXD0 pid2 = pXD0.getPidObject();
            System.out.println("          XD0 pid string: " +
                              pid2.pidString());
            // Retrieve and open instance handler for an XD0
            pXD0.retrieve();
            // pXD0.open();
        }
    }
}
```

C++

```
DKDDO *p = 0;
DKAny a;

for (j = 1; j <= numDataItems; j++)
{
    a = p->getData(j);
    strDataName = p->getDataName(j);

    cout << " " << j << ". Attribute Name: " << strDataName << endl;
    cout<<"type: "<< p->getDataPropertyByName(j,DK_PROPERTY_TYPE)<<endl;
    cout << "nullable: "
        << p->getDataPropertyByName(j,DK_PROPERTY_NULLABLE) << endl;

    if (strDataName != DK_CM_DKPARTS    &&
        strDataName != "DKResource"    &&
        strDataName != "DKViews"       &&
        strDataName != "DKLargeObject" &&
        strDataName != "DKPermissions" &&
        strDataName != "DKFixedView"   &&
        strDataName != "DKAnnotations")
    {
        cout << "    attribute ID: "
            << p->getDataPropertyByName(j,DK_PROPERTY_ATTRIBUTE_ID) << endl;
    }

    if (a.typeCode() == DKAny::tc_string)
    {
        DKString astring = a;
        cout << "    attribute Value (string): " << astring << endl;
    }
    else if . . .
    {
        // ----- Handle each of the other types
    }
    else if (a.typeCode() != DKAny::tc_null)
    {
        cout << "    Attribute Value (non NULL): " << a << endl;
        if (strDataName == "DKResource" ||
            strDataName == "DKFixedView" ||
            strDataName == "DKLargeObject")
        {
            pDO = (dkDataObjectBase*)a;
            if (pDO->protocol() == DK_XDO)
            {
                cout << "        dkXDO object " << endl;
                pXDO = (dkXDO*)pDO;
                pidXDO = (DKPidXDOOD*)pXDO->getPid();
                cout << "        XDO PID string: " << pidXDO->pidString() << endl;
                // ----- Retrieve and open instance handler for an XDO
                pXDO->retrieve();
            }
        }
    }
    else cout << " Attribute Value is NULL" << endl;
}
```

Para obter a aplicação completa, consulte TRetrieveOD.cpp no directório CMBROOT\samples\cpp\od.

Activar o modo de pastas OnDemand

Para activar o modo de pastas OnDemand, a cadeia
ENTITY_TYPE=TEMPLATES

tem de ser transmitida para o conector OnDemand como parte da cadeia de ligação ou da cadeia de configuração. Uma cadeia de configuração exemplo teria o seguinte aspecto:

Java

```
DKDatastoreOD dsOD = new DKDatastoreOD("ENTITY_TYPE=TEMPLATES");
```

C++

```
DKDatastoreOD* dsOD = new DKDatastoreOD("ENTITY_TYPE=TEMPLATES");
```

Uma cadeia de ligação exemplo teria o seguinte aspecto:

Java

```
DKDatastoreOD dsOD = new DKDatastoreOD();  
dsOD.connect(hostname, userid, password, "ENTITY_TYPE=TEMPLATES");
```

C++

```
DKDatastoreOD* dsOD = new DKDatastoreOD("ENTITY_TYPE=TEMPLATES");  
dsOD->connect(hostname, userid, password, "ENTITY_TYPE=TEMPLATES");
```

Pesquisa assíncrona

O conector OnDemand suporta pesquisas associadas e assíncronas directas. Uma pesquisa assíncrona não obstrui o módulo principal e permite o cancelamento da pesquisa em qualquer altura; prima o botão **Parar Pesquisa** no Visualizador do Modelo da Pesquisa para concluir a pesquisa. A propriedade max hits no Visualizador do Modelo da Pesquisa limita o número máximo de resultados devolvidos.

O conector OnDemand também suporta pesquisas síncronas e assíncronas no modo do grupo de aplicações a partir do cliente de AIX.

Se utilizar um critério de pesquisa como o WHERE userid LIKE '%', o número resultante de documentos devolvidos ao cliente pode consumir toda a memória disponível na máquina cliente'. Ao emitir uma pesquisa assíncrona através do método executeWithCallback(), poderá definir o valor do número máximo de documentos devolvidos e cancelar a pesquisa a qualquer momento.

Provavelmente também terá de aumentar o tamanho da pilha predefinido para a Java Virtual Machine (JVM) se o seu conjunto de resultados for demasiado extenso. O tamanho de pilha predefinido para cada módulo de Java é de 400k, que permite 3920 artigos devolvidos antes que a pilha ultrapasse a sua capacidade. Aumentar o tamanho da pilha da JVM para 800k duplica a capacidade para 7840 artigos. Caso seja necessário, poderá depois aumentar o tamanho da pilha da JVM.

Para aumentar o tamanho da pilha da JVM, utilize a opção de linha de comandos de Java `-oss` seguida de `nnnK` ou `nnM`, onde K significa Kilobytes e M Megabytes.

Para exemplos de utilização da pesquisa assíncrona, consulte os programas exemplo `TRetrieveWithCallbackOD`, `TRetrieveFolderWithCallbackOD` e `TCallbackOD`.

Pastas OnDemand como modelos de pesquisa

Três dos JavaBeans visuais do EIP, `CMBSearchTemplateList`, `CMBSearchTemplateViewer` e `CMBSearchResultsView` utilizam modelos de pesquisa do EIP. Poderá utilizar estes beans para pesquisas associadas através da definição de `CMBConnection` `dsType` como `Assoc`.

Poderá utilizar estes beans também para pesquisas directas em servidores OnDemand. Defina as propriedades seguintes antes de iniciar sessão:

```
connection.setDsType("OD");  
connection.setServerName(<odserver>);  
connection.setConnectionString("ENTITY_TYPE=TEMPLATES");
```

Pastas OnDemand como entidades nativas

O conector de OnDemand também pode definir correspondências de pastas OnDemand como entidades nativas especificando a cadeia de ligação `"ENTITY_TYPE=TEMPLATES"`. Utilizar as pastas OnDemand como entidades pode ser útil em pesquisas associadas, onde as definições da pasta podem ser mais fáceis de trabalhar do que em grupos de aplicações OnDemand, a entidade nativa predefinida para OnDemand.

Para pesquisas associadas, especifique a definição do servidor na administração do Enterprise Information Portal. Depois já pode definir e definir as correspondências das entidades associadas para as pastas no servidor OnDemand.

Criar e modificar anotações

Quando utilizar o visualizador OnDemand, que é iniciado pelo bean de `CMBDocumentViewer`, poderá criar, modificar e eliminar anotações para documentos OnDemand, utilizando o bean `CMBDataManagement` e a classe `CMBAnnotation` associada.

Rastreio

Pode rastrear eventos com o conector de OnDemand. Para activar o rastreio de API de java do conector, coloque o ficheiro INI de rastreio (`cmbodtrace.ini` para Java ou `cmbodctrace.inifor C++`), na raiz da unidade C (`C:\`) ou no directório especificado na variável `CMBROOT`. Em AIX, coloque este ficheiro em `/usr/lpp/cmb/cmgt`. Em Solaris, coloque este ficheiro em `/opt/IBMcmb/cmgt`.

O directório predefinido de saída para ficheiros de rastreio é `C:\Ctrace`. Para escrever informações de rastreio noutra local, edite o ficheiro INI de rastreio. Em AIX, tenha em atenção que os nomes dos ficheiros devem estar em letra minúscula.

Certifique-se de que o nome de caminho especificado no ficheiro de rastreio é válido e de que a linha que contém `CMBODTRACEDIR` não é antecedita por um sinal de `#`. Os ficheiros INI de rastreio exemplo são:

Java

```
#=====
# This is a java property file - not a real INI file!
# *****#
# For windows systems, make sure to use TWO BACK SLASH CHARACTERS (\\)
# to separate the directory names!!! =====
# *****#
#
# ***** On windows systems, this file must be located in c:\ *****
#
# OD Trace File Directory Name property - CMBODTRACEDIR
#
# The CMBODTRACEDIR property defines the directory where the trace files
# will be written to. If the directory name does not exist, it will be
# created.
#
# Please make sure that the directory names are separated by two back slash
# characters to avoid undesirable results.
#
# Please make sure the path name does not point to an existing file name.
# Otherwise, no trace files will be created.
#
# The trace output directory name can be changed to point to a drive
# where more space is available. But it is recommended not to change the
# trace output directory name in the middle of an active trace session.
#
# CMBODTRACESCOPE controls how much trace information to generate.
#
# CMBODTRACESCOPE=ENTRY_EXIT_JNI_ONLY
# Trace the entry & exit points in JNI only. Produce the least amount
# of trace.
#
# CMBODTRACESCOPE=ENTRY_EXIT_ONLY
# Trace the entry and exit points in Java methods and JNI functions.
#
# CMBODTRACESCOPE=JNI_ONLY
# Full trace for the JNI functions only.
#
# If CMBODTRACESCOPE is missing, or set to anything else,
# a full trace will be taken.
#
# To disable the trace, add a leading # character in column 1.
#
# AIX: change the following line to CMBODTRACEDIR=/usr/lpp/cmb/cmgmt/trace
# Sun: change the following line to CMBODTRACEDIR=/opt/IBMcmb/cmgmt/trace
CMBODTRACEDIR=c:\\trace
```


C++

```
#=====
# Ficheiro INI de Rastreio de OnDemand
#
# Chave de Nome do Directório do Ficheiro de Rastreio de OnDemand
# - CMBODTRACEDIR
#
# A chave CMBODTRACEDIR define o directório onde os ficheiros de rastreio serão
# gravados. Se o nome do directório não existir, este será criado.
#
# Certifique-se de que o nome de caminho não indica um nome de ficheiro
# existente. Caso contrário, não serão criados ficheiros de rastreio.
#
# O nome do directório de saída de rastreio pode ser alterado para indicar
# uma unidade em que exista mais espaço disponível. Mas não se recomenda a alteração
# do nome do directório de saída de rastreio a meio de uma sessão activa
# de rastreio.
#
# CMBODTRACESCOPE controla a quantidade de informações de rastreio a gerar.
#
# CMBODTRACESCOPE=ENTRY_EXIT_ONLY
# Efectue apenas o rastreio da entrada e saída de todos os métodos e funções
# de C++.
#
# Se CMBODTRACESCOPE estiver em falta, ou estiver definido doutra forma,
# é realizado um rastreio completo.
#
# Para desactivar o rastreio, adicione um carácter # inicial à coluna 1 da
# linha CMBODTRACEDIR.
#
[ODTRACE]
# For AIX: change next line to CMBODTRACEDIR=/usr/lpp/cmb/cmgmt/ctrace
CMBODTRACEDIR=D:\Ctrace
#CMBODTRACESCOPE=ENTRY_EXIT_ONLY
```

Trabalhar com Content Manager ImagePlus for OS/390

As APIs do Enterprise Information Portal suportam as seguintes funções ao trabalhar com os servidores de conteúdos do Content Manager ImagePlus for OS/390:

- Estabelecer e anular ligações a um ou mais do que um servidor de ImagePlus.
- Obter categorias.
- Obter campos de atributo.
- Obter pastas.
- Obter documentos.

Represente um servidor de conteúdos do ImagePlus for OS/390 através do DKDatastoreIP na sua aplicação.

Restrição: O ImagePlus para OS/390 não suporta:

- Pesquisa do Motor de Pesquisa de Texto e de QBIC
- Consulta combinada
- Cesto de trabalho e fluxo de trabalho

Enumerar entidades e atributos

Após criar um servidor de conteúdos para o servidor de conteúdos de ImagePlus for OS/390 como um objecto de DKDatastoreIP e de estabelecer ligação, pode verificar as entidades e atributos do servidor de conteúdos. O exemplo seguinte enumera todas as entidades de um servidor de conteúdos de ImagePlus for OS/390:

Java

```
// ----- After creating a datastore and connecting
//          dsIP is a DKDatastoreIP object
DKEntityDefIP entDef = null;
DKAttrDefIP attrDef = null;

DKSequentialCollection pCol = (DKSequentialCollection)dsIP.listEntities();
dkIterator pIter = null;

if ( pCol == null )
{
    // ----- Handle if the collection of entities is null
}
else
{
    ... // ----- Process as appropriate
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogIP.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
// List entities...
DKEntityDefIP* docDef = 0;
DKAttrDefIP* attrDef = 0;

cout << "---List entities---" << endl;
DKSequentialCollection* pCol = (DKSequentialCollection*)(dsIP.listEntities());
dkIterator* pIter = 0;

if ( pCol == 0 )
{
    cout << "collection of entities is null!" << endl;
}
else
{
    ...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogIP.cpp) está disponível no directório Cmbroot/Samples/cpp/ip.

O exemplo seguinte enumera todos os atributos associados a cada entidade, utilizando as funções getAttr e listAttrNames de DKEntityDefIP.

Java

```
// ----- List attributes using listAttrNames and getAttr methods

pIter = pCol.createIterator();
while (pIter.more())
{
    // ----- Iterate over the each entity
    entDef = (DKEntityDefIP)pIter.next();
    System.out.println(" Entity type      : " + entDef.getType() );
    System.out.println(" Entity type name: " + entDef.getName() );

    // ----- Get a list of attributes for the entity
    String[] attrNames = entDef.listAttrNames();
    int count = attrNames.length;
    for (int i = 0; i < count; i++)
    {
        attrDef = (DKAttrDefIP)entDef.getAttr( attrNames[i] );
        System.out.println("   Attr name      : " + attrDef.getName() );
        System.out.println("   Attr id        : " + attrDef.getId() );
        System.out.println("   Entity name     : " + attrDef.getEntityName() );
        System.out.println("   Datastore name: " + attrDef.datastoreName() );
        System.out.println("   Attr type       : " + attrDef.getType() );
        System.out.println("   Attr restrict  : " + attrDef.getStringType() );
        System.out.println("   Attr min val   : " + attrDef.getMin() );
        System.out.println("   Attr max val   : " + attrDef.getMax() );
        System.out.println("   Attr display   : " + attrDef.getSize() );
        System.out.println("   Attr precision: " + attrDef.getPrecision() );
        System.out.println("   Attr scale     : " + attrDef.getScale() );
        System.out.println("   Attr update    ? " + attrDef.isUpdatable() );
        System.out.println("   Attr nullable ? " + attrDef.isNullable() );
        System.out.println("   Attr queryable? " + attrDef.isQueryable() );
        System.out.println("   ");
    }
}
```

C++

```
// Method 1:
cout << "List attributes using listAttrNames and getAttr functions" << endl;

pIter = pCol->createIterator();
while (pIter->more())
{
    docDef = (DKEntityDefIP*)(pIter->next()->value());
    cout << " Document type      : " << docDef->getType() << endl;
    cout << " Document type name: " << docDef->getName() << endl;

    long tmpCount;
    DKString* attrNames;

    // Upon return, tmpCount contains the number of elements in the list.
    attrNames = docDef->listAttrNames(tmpCount);
    for (int i=0; i<tmpCount; i++)
    {
        cout << "  Attr name before lookup " << attrNames[i] << endl;
        attrDef = (DKAttrDefIP*)(docDef->getAttr(attrNames[i]));
        cout << "  Attr name [" << i << "] : " << attrDef->getName() << endl;
        cout << "  Attr id      : " << attrDef->getId() << endl;
        cout << "  Entity name  : " << attrDef->getEntityName() << endl;
        cout << "  Datastore name: " << attrDef->datastoreName() << endl;
        cout << "  Attr type    : " << attrDef->getType() << endl;
        cout << "  Attr restrict : " << attrDef->getStringType() << endl;
        cout << "  Attr min val  : " << attrDef->getMin() << endl;
        cout << "  Attr max val  : " << attrDef->getMax() << endl;
        cout << "  Attr display  : " << attrDef->getSize() << endl;
        cout << "  Attr precision: " << attrDef->getPrecision() << endl;
        cout << "  Attr scale    : " << attrDef->getScale() << endl;
        cout << "  Attr update   ? " << attrDef->isUpdatable() << endl;
        cout << "  Attr nullable ? " << attrDef->isNullable() << endl;
        cout << "  Attr queryable? " << attrDef->isQueryable() << endl;
        cout << "" << endl;
        delete attrDef;
    } // end for

    delete [] attrNames;

} // end while
delete pIter;
```

O exemplo seguinte demonstra uma forma alternativa de enumerar os atributos associados a cada entidade, utilizando o método `listEntityAttrs` de `DKDatastoreIP`.

Java

```
// --- List attributes using listEntityAttrs method
pIter = pCol.createIterator();
while (pIter.more())
{
    entDef = (DKEntityDefIP)pIter.next();
    System.out.println(" Entity type      : " + entDef.getType() );
    System.out.println(" Entity type name: " + entDef.getName() );

    DKSequentialCollection pAttrCol =
        (DKSequentialCollection)dsIP.listEntityAttrs(entDef.getName());
    if ( pAttrCol == null )
    {
        // ----- Handle if the collection of attributes is null
    }
    else
    {
        dkIterator pAttrIter = pAttrCol.createIterator();
        while (pAttrIter.more())
        {
            attrDef = (DKAttrDefIP)pAttrIter.next();
            System.out.println("  Attr name      : " + attrDef.getName() );
            System.out.println("  Attr id       : " + attrDef.getId() );
            System.out.println("  Entity name    : " + attrDef.getEntityName() );
            System.out.println("  Datastore name: " + attrDef.datastoreName() );
            System.out.println("  Attr type      : " + attrDef.getType() );
            System.out.println("  Attr restrict  : " + attrDef.getStringType() );
            System.out.println("  Attr min val   : " + attrDef.getMin() );
            System.out.println("  Attr max val   : " + attrDef.getMax() );
            System.out.println("  Attr display   : " + attrDef.getSize() );
            System.out.println("  Attr precision: " + attrDef.getPrecision() );
            System.out.println("  Attr scale     : " + attrDef.getScale() );
            System.out.println("  Attr update    ? " + attrDef.isUpdatable() );
            System.out.println("  Attr nullable ? " + attrDef.isNullable() );
            System.out.println("  Attr queryable? " + attrDef.isQueryable() );
            System.out.println(" ");
        }
    }
}
```

C++

```
// Method 2:
cout << "----List attributes using listEntityAttrs function----" << endl;

pIter = pCol->createIterator();
while (pIter->more())
{
    docDef=(DKEntityDefIP*)(pIter->next()->value()); //iterator returns DKAny*
    cout << " Document type      : " << docDef->getType() << endl;
    cout << " Document type name: " << docDef->getName() << endl;
    DKSequentialCollection* pAttrCol = (DKSequentialCollection*)
                                         (dsIP.listEntityAttrs(docDef->getName()));

    if ( pAttrCol == 0 )
    {
        cout << "collection of entity attrs is null for entity "
              << docDef->getName()
              << endl;
    }
    else
    {
        int i=0;
        dkIterator* pAttrIter = pAttrCol->createIterator();
        while (pAttrIter->more())
        {
            i++;
            // ----- The iterator returns a pointer to DKAny
            attrDef = (DKAttrDefIP*)(pAttrIter->next()->value());
            cout << "  Attr name [" << i << "] : " << attrDef->getName() << endl;
            cout << "  Attr id      : " << attrDef->getId() << endl;
            cout << "  Entity name  : " << attrDef->getEntityName() << endl;
            cout << "  Datastore name: " << attrDef->datastoreName() << endl;
            cout << "  Attr type    : " << attrDef->getType() << endl;
            cout << "  Attr restrict : " << attrDef->getStringType() << endl;
            cout << "  Attr min val  : " << attrDef->getMin() << endl;
            cout << "  Attr max val  : " << attrDef->getMax() << endl;
            cout << "  Attr display  : " << attrDef->getSize() << endl;
            cout << "  Attr precision: " << attrDef->getPrecision() << endl;
            cout << "  Attr scale    : " << attrDef->getScale() << endl;
            cout << "  Attr update   ? " << attrDef->isUpdatable() << endl;
            cout << "  Attr nullable ? " << attrDef->isNullable() << endl;
            cout << "  Attr queryable? " << attrDef->isQueryable() << endl;
            cout << "" << endl;
            delete attrDef;
        } // end while
        delete pAttrIter;
    }
    delete pAttrCol;
    delete docDef;
} // end while
delete pIter;
}
```

```
delete pCol;
```

Sintaxe da consulta de ImagePlus para OS/390

O exemplo seguinte apresenta a sintaxe da consulta de ImagePlus para OS/390:

Java

```
SEARCH = (COND=(search_expression), ENTITY={entity_name | mapped_entity_name}  
[,MAX_RESULTS=maximum_results]);  
[OPTION=([CONTENT={YES | ATTRONLY | NO};][PENDING={YES | NO};])]
```

C++

```
SEARCH=(COND=(search_expression),ENTITY={entity_name | mapped_entity_name}  
[,MAX_RESULTS=maximum_results]);  
[OPTION=([CONTENT={YES | ATTRONLY | NO};][PENDING={YES | NO};])]
```

A consulta utiliza os seguintes parâmetros:

search_expression

Cada expressão de pesquisa consiste num ou mais critérios de pesquisa. Só pode utilizar o operador booleano AND entre critérios de pesquisa.

Os critérios de pesquisa possuem a seguinte forma:

{attr_name | mapped_attr_name} operator literal

em que:

attr_name

É o nome do atributo da entidade no qual se deve basear a pesquisa.

mapped_attr_name

É o nome de atributo com correspondências definidas ao atributo no qual se deve basear a pesquisa.

operador

Todos os atributos suportam sinais de igual (==). Para atributos de tipo DATE, pode utilizar os seguintes operadores adicionais:

- > maior que
- < menor que
- >= maior que ou igual a
- <= menor que ou igual a

literal

Um literal. Em atributos numéricos, não utilize aspas ("), por exemplo:

FolderType == 9

Para atributos de data, hora e marca da hora, não são necessários aspas ou apóstrofos ('), mas são tolerados, por exemplo:

ReceiveDate == 1999-03-08
ReceiveDate == '1999-03-08'

Para atributos de cadeia, não são necessários aspas ou apóstrofos ('), mas são tolerados. Se a cadeia contiver um apóstrofo ('), a cadeia terá de ser especificada através de dois apóstrofos, por exemplo para um valor de Folder'1:

FolderId == 'Folder''1'

entity_name

Nome da entidade a ser pesquisada.

mapped_entity_name

Nome da entidade com correspondências definidas à entidade a ser pesquisada.

maximum_results

Número máximo de resultados a devolver.

As palavras chave opcionais são:

| | |
|---------------------------|---|
| CONTENT | Controla a quantidade de informação devolvida nos resultados |
| YES (predefinição) | Define os PIDs, os atributos e os seus valores de um documento ou pasta. São definidos os PIDs de XDO, caso existam partes num documento. Se existirem documentos numa pasta, são definidos os PIDs dos documentos. |
| NO | Apenas define os PIDs do documento ou da pasta. |
| ATTRONLY | Apenas define os PIDs, os atributos e os seus valores de um documento ou pasta. |
| PENDING | Controla se deve ou não incluir documentos pendentes que não tem quaisquer partes. Esta opção aplica-se apenas quando ENTITY é definido como DOCUMENT ou como uma entidade com correspondência definida para DOCUMENT . |
| YES | Inclui documentos pendentes nos resultados. |
| NO (predefinição) | Não inclui documentos pendentes nos resultados. |

Trabalhar com o Content Manager for AS/400

As classes de API facultadas para o Content Manager for AS/400 (VisualInfo for AS/400) são idênticas às facultadas para o Content Manager.

Restrição: O Content Manager para AS/400 não suporta:

- Pesquisa do Motor de Pesquisa de Texto e de QBIC
- Consulta combinada
- Cesto de trabalho e fluxo de trabalho

Enumerar entidades (classes de índices) e atributos

O utilizador irá representar um servidor de conteúdos do Content Manager for AS/400 como um DKDatastoreV4. Após criar o servidor de conteúdos e de lhe estabelecer ligação, pode enumerar as entidades (classes de índice) e atributos do servidor Content Manager for AS/400 (consultar o exemplo).

Java

```
// ----- After creating a datastore (dsV4) and connecting, get index classes
pCol = (DKSequentialCollection) dsV4.listEntities();
pIter = pCol.createIterator();
i = 0;
while (pIter.more() == true)
{
    i++;
    icDef = (DKIndexClassDefV4)pIter.next();
    strIndexClass = icDef.getName();
    ... // ---- Process the index classes as appropriate
    // ----- Get the attributes
    pCol2 = (DKSequentialCollection) dsV4.listEntityAttrs(strIndexClass);
    pIter2 = pCol2.createIterator();
    j = 0;

    while (pIter2.more() == true)
    {
        j++;
        attrDef = (DKAttrDefV4)pIter2.next();
        ... // ----- Process the attributes
    }
}

dsV4.disconnect();
dsV4.destroy();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogV4.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
cout << "list index class(es)..." << endl;
pCol = (DKSequentialCollection*)((dkCollection*)dsV4.listSchema());
pIter = pCol->createIterator();
i = 0;

while (pIter->more() == TRUE)
{
    i++;
    a = (*pIter->next());
    strIndexClass = a;
    cout << "index class name [" << i << "] - " << strIndexClass << endl;
    cout << " list attribute(s) for " << strIndexClass << " index class:" << endl;
    pCol2 =

(DKSequentialCollection*)((dkCollection*)dsV4.listSchemaAttributes(strIndexClass));
    pIter2 = pCol2->createIterator();
    j = 0;

    while (pIter2->more() == TRUE)
    {
        j++;
        pA = pIter2->next();
        pDef = (DKAttributeDef*) pA->value();
        cout << "    Attribute name [" << j << "] - " << pDef->name << endl;
        cout << "    datastoreType - " << pDef->datastoreType << endl;
        cout << "    attributeOf - " << pDef->attributeOf << endl;
        cout << "    type - " << pDef->type << endl;
        cout << "    size - " << pDef->size << endl;
        cout << "    id - " << pDef->id << endl;
        cout << "    nullable - " << pDef->nullable << endl;
        cout << "    precision - " << pDef->precision << endl;
        cout << "    scale - " << pDef->scale << endl;
        cout << "    string type - " << pDef->stringType << endl;
    }

    cout << " " << j << " attribute(s) listed for "
        << strIndexClass << " index class" << endl;
    pCol2->apply(deleteDKAttributeDef);
    delete pIter2;
    delete pCol2;
}

delete pIter;
delete pCol;
cout << i << " index class(es) listed" << endl;
dsV4.disconnect();
cout << "datastore disconnected" << endl;
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogV4.cpp) está disponível no directório Cmbroot/Samples/cpp/v4.

Executar uma consulta

O exemplo seguinte executa uma consulta em Content Manager for AS/400 e processa os resultados.

Java

```
// ----- After creating a datastore (dsV4) and connecting, build the
//          query and parameters and execute it
pCur = dsV4.execute(cmd,DK_CM_PARAMETRIC_QL_TYPE,parms);
...

if (pCur == null)
{
    // ---- Handle if the cursor is null
}

while (pCur.isValid())
{
    p = pCur.fetchNext();
    if (p != null)
    {
        cnt++;
        i = pCur.getPosition();
        System.out.println("\n====> Item " + i + " <=====");
        numDataItems = p.dataCount();
        DKPid pid = p.getPid();
        System.out.println("  pid string: " + pid.pidString());
        k = p.propertyId(DK_CM_PROPERTY_ITEM_TYPE);

        if (k > 0)
        {
            Short sVal = (Short)p.getProperty(k);
            j = sVal.shortValue();
            switch (j)
            {
                case DK_CM_DOCUMENT :
                {
                    ... // Handle if the item is a document ";
                    break;
                }
                case DK_CM_FOLDER :
                {
                    ... // Handle if the item is a folder
                    break;
                }
            }
        }
    }
}

for (j = 1; j <= numDataItems; j++)
{
    a = p.getData(j);
    strDataName = p.getDataName(j);
    ... // Process the attributes as appropriate
    if (strDataName.equals(DKPARTS) == false
        && strDataName.equals(DKFOLDER) == false)
    {
        System.out.println("      attribute id: "
            + p.getDataPropertyByName(j,DK_CM_PROPERTY_ATTRIBUTE_ID));
    }
}
// continued...
```

Java (continuação)

```
if (a instanceof String)
{
    System.out.println("    Attribute Value: " + a);
}
else if (a instanceof Integer)
    ... // ---- Handle each type for attribute {
else if (a instanceof dkCollection)
{
    // ---- Handle if attribute value is a collection
    pCol = (dkCollection)a;
    pIter = pCol.createIterator();
    i = 0;
    while (pIter.more() == true)
    {
        i++;
        a = pIter.next();
        pDO = (dkDataObjectBase)a;

        if (pDO.protocol() == DK_CM_PDD0)
        {
            // Process a DDO
            pDD0 = (DKDD0)pDO;
            ...
        }
        else if (pDO.protocol() == DK_CM_XD0)
        {
            // Process an XD0
            pXD0 = (dkXD0)pDO;
            DKPidXD0 pid2 = pXD0.getPid();
            ...
        }
    }
}
else if (a != null)
{
    // Process the attribute
}
else ... // Handle if the attribute is null
}
}
pCur.destroy(); // Delete the cursor when you're done
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteV4.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
cout << "executing query..." << endl;
...
pCur = dsV4.execute(cmd);
cout << " query executed" << endl;
...
cout << "\n..... Displaying query results ..... \n\n";

...
while (pCur->isValid())
{
    p = pCur->fetchNext();

    if (p != 0)
    {
        cout << "=====" << "Item " << cnt << " <=====" << endl;
        numDataItems = p->dataCount();
        pid = p->getPid();
        cout << " Pid String: " << pid.pidString() << endl;
        k = p->propertyId(DK_CM_PROPERTY_ITEM_TYPE);

        if (k > 0)
        {
            a = p->getProperty(k);
            val = a;
            cout << " *****" << endl;

            switch (val)
            {
                case DK_CM_DOCUMENT :
                {
                    cout << " Item is a document " << endl;
                    break;
                }
                case DK_CM_FOLDER :
                {
                    cout << " Item is a folder " << endl;
                    break;
                }
            }

            cout << " *****" << endl;
        }

        cout << " Number of Data Items: " << numDataItems << endl;

        for (j = 1; j <= numDataItems; j++)
        {
            a = p->getData(j);
            strDataName = p->getDataName(j);

            switch (a.typeCode())
            {
                case DKAny::tc_string :
                {
                    strData = a;
                    cout << " attribute name: " << strDataName
                        << ", value: " << strData << endl;
                    break;
                }
            }
        }
    }
}

// continued...
```

C++ (continuação)

```
case DKAny::tc_long :
{
    lVal = a;
    cout << " attribute name: " << strDataName
        << ", value: " << lVal << endl;
    break;
}

case DKAny::tc_null :
{
    cout<<" attribute name: "<<strDataName<<", value: NULL "<< endl;
    break;
}

case DKAny::tc_collection :
{
    pdCol = a;
    cout<<strDataName<<" collection name: "<<strDataName << endl;
    cout<<"-----"<<endl;
    pdIter = pdCol->createIterator();
    ushort b = 0;

    while (pdIter->more() == TRUE)
    {
        b++;
        cout << " -----" << endl;
        a = *(pdIter->next());
        pDOBase = a;

        if (pDOBase->protocol() == DK_PDDO)
        {
            pDDO = (DKDDO*)pDOBase;
            cout << " DKDDO object " << b << " in " << strDataName
                << " collection " << endl;
            k = pDDO->propertyId(DK_CM_PROPERTY_ITEM_TYPE);

            if (k > 0)
            {
                a = pDDO->getProperty(k);
                val = a;
                cout << " *****" << endl;

                switch (val)
                {
                    case DK_CM_DOCUMENT :
                    {
                        cout << " Item is a document " << endl;
                        break;
                    }
                    case DK_CM_FOLDER :
                    {
                        cout << " Item is a folder " << endl;
                        break;
                    }
                }
                cout << " *****" << endl;
            }
        }
    }
}

// continued...
```

C++ (continuação)

```
        else if (pD0Base->protocol() == DK_XD0)
        {
            pXD0 = (dkXD0*)pD0Base;
            cout << "  dkXD0 object " << b << " in " << strDataName
                << " collection " << endl;

            }
        }

        if (pdIter != 0)
        {
            delete pdIter;
        }

        if (b == 0)
        {
            cout << strDataName << " collection has no elements " << endl;
        }

        cout << "  -----" << endl;
        break;
    }

    default:
        cout << "Type is not supported\n";
    }

    cout<<"type: "<< p->getDataPropertyByName(j,DK_CM_PROPERTY_TYPE)<<endl;
    cout<<"nullable: "<< p->getDataPropertyByName(j,DK_CM_PROPERTY_NULLABLE)
        << endl;

    if (strDataName != DKPARTS && strDataName != DKFOLDER)
    {
        cout << "    attribute id: "
            << p->getDataPropertyByName(j,DK_PROPERTY_ATTRIBUTE_ID) << endl;
    }
}
cnt++;
delete p;
}
cout << "Total Item count is " << cnt-1 << endl;

if (pCur != 0)
    delete pCur;
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteV4.cpp) está disponível no directório Cmbroot/Samples/cpp/v4.

Executar uma consulta paramétrica

O exemplo seguinte executa uma consulta paramétrica.

Java

```
// ----- Create the query string and the query object
String cmd = "SEARCH=(INDEX_CLASS=V4DEMO)";
pQry = dsV4.createQuery(cmd, DK_CM_PARAMETRIC_QL_TYPE, parms);
// ----- Run the query
pQry.execute(parms);

System.out.println("number of query results = " + pQry.numberOfResults());

// ----- Processing the query results
pResults = (DKResults)pQry.result();
processResults((dkCollection)pResults);
...
```

A aplicação exemplo completa da qual este exemplo foi retirado(TSamplePQryV4.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
cout << "query string: " << cmd << endl;
cout << "creating query..." << endl;
pQry = dsV4.createQuery(cmd);
cout << "executing query..." << endl;
pQry->execute();
cout << "query executed" << endl;
cout << "getting query results..." << endl;
any = pQry->result();
pResults = (DKResults*)((dkCollection*) any);

processResults(pResults);

dsV4.disconnect();
cout << "datastore disconnected" << endl;
delete pQry;
delete pResults;
```

A aplicação exemplo completa da qual este exemplo foi retirado (TSamplePQryV4.cpp) está disponível no directório Cmbroot/Samples/cpp/v4.

Trabalhar com Domino.Doc

O Domino.Doc é a solução de Lotus Domino para organização, gestão e armazenamento de documentos empresariais, bem como para os tornar acessíveis dentro e fora de uma empresa.

O Domino.Doc suporta a API de gestão de documentos abertos (ODMA), de forma a poder criar, guardar e obter documentos através de aplicações activadas para ODMA. A ODMA liga-se a um servidor de Domino.Doc através de HTTP ou do protocolo de Lotus Notes.

O Domino.Doc inclui as seguintes funções:

- Estabelecer e anular ligações a um ou mais do que um servidor de Domino.Doc.
- Capacidade de pesquisar processadores de enlace.
- Capacidade de obter documentos.

- Conformidade com ODMA, de modo a que os utilizadores possam trabalhar em aplicações familiares.

Restrição: O Domino.Doc não suporta:

- Adicionar, actualizar e eliminar métodos de documentos.
- Pesquisa do Motor de Pesquisa de Texto e QBIC.
- Consulta combinada.
- Cesto de trabalho e fluxo de trabalho.

Ao utilizar a API para trabalhar com um objecto de Domino.Doc, terá de construir uma expressão para devolver os objectos. Esta secção descreve a concepção da API de Domino.Doc, o modo como os objectos se encaixam na hierarquia e como construir a expressão. A Figura 19 apresenta a relação entre o modelo de objectos de Domino.Doc e os seus componentes.

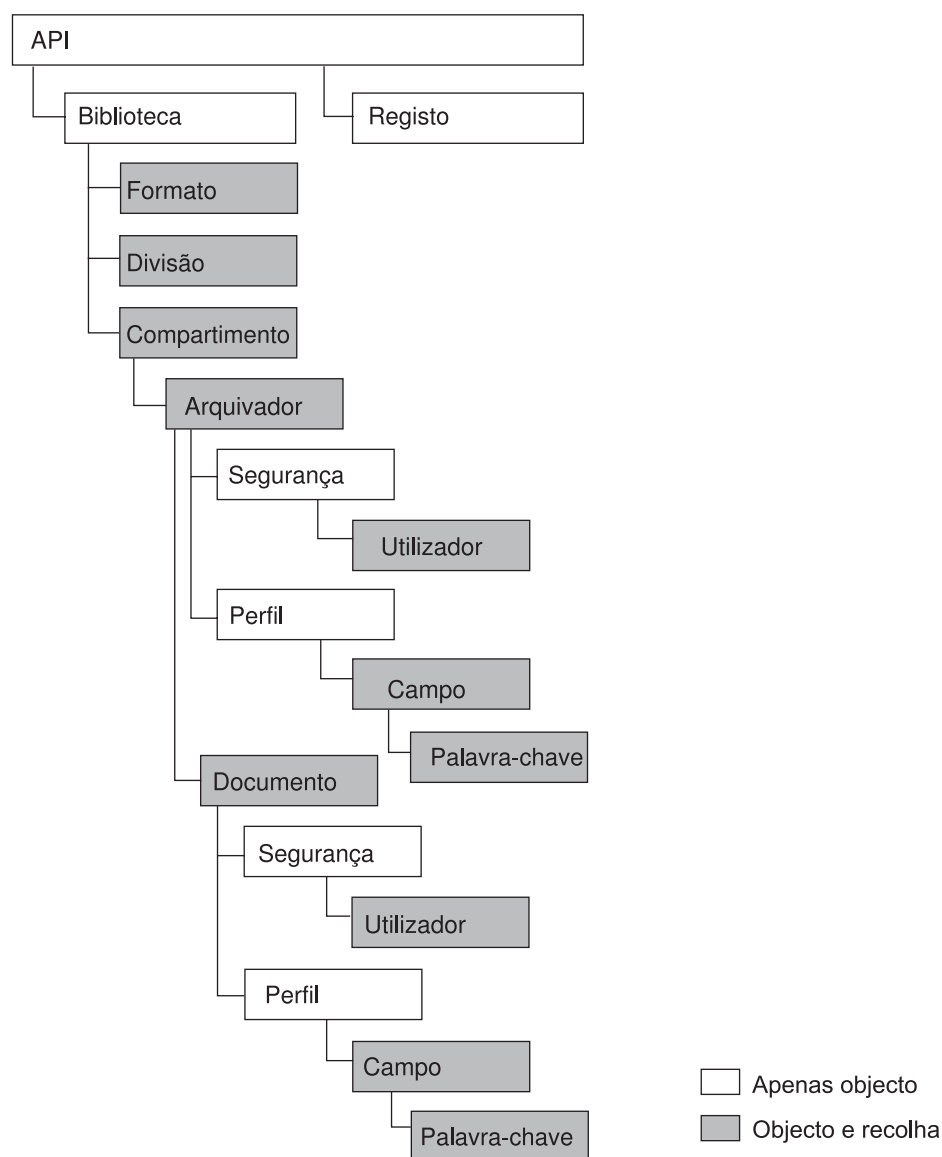


Figura 19. Modelo de objectos do Domino.Doc

Os elementos contidos no Domino.Doc (e as APIs que os representam) estão organizados de forma a:

- A biblioteca contém divisões (DKRoomDefDD objects) e armários(objectos de DKCabinetDefDD).
- Cada armário contém processadores de enlace (objecto de DKBinderDefDD).
- Cada processador de enlaces contém um perfil (objecto de DKAttrProfileDefDD) e segurança.
- Cada processador de enlaces contém documentos (objectos de DKDocumentDefDD).
- Cada documento contém um perfil (objecto de DKAttrProfileDefDD) e segurança.
- Cada perfil contém campos (objectos de DKAttrFieldDefDD).
- Cada campo contém palavras-chave (objectos de DKAttrKeywordDefDD).

Enumerar entidades e sub-entidades

O exemplo seguinte enumera as entidades (que neste exemplo são divisões) do Domino.Doc e as suas sub-entidades (que neste exemplo são armários, processadores de enlace e documentos).

Java

```
...
// ----- Get a list of rooms
dkCollection rooms = dsDD.listEntities();
// ----- Iterate thru the rooms and their subEntities
dkIterator itRooms = rooms.createIterator();
itRooms.reset();
while( itRooms.more() ) {
    ... // Process the rooms and entities
}
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListSubEntitiesDD.java) está disponível no directório CMBROOT\Samples\java\dd.

C++

```
dkCollection* pColl = domDoc.listEntities();

long nbrEnts = pColl->cardinality();

dkIterator* itEnts = pColl-> createIterator();
while( itEnts->more() )
{ // For each returned dkEntityDef...
  DKRoomDefDD* pEnt = (DKRoomDefDD*)itEnts->next()->value();
  cout << "Room title: " << pEnt->getName() << endl;
  cout << "  Has SubEntities: " << pEnt->hasSubEntities() << endl;

  // print subEntities (Cabinets->Binders->Documents)
  printSubEnts(pEnt, domDoc, 1);

  delete pEnt;
}
delete itEnts;
delete pColl;
```

A aplicação exemplo completa da qual este exemplo foi retirado(TListEntitiesDD.cpp) está disponível no directório Cmbroot/Samples/cpp/dd.

O exemplo seguinte lista atributos e palavras chave de documentos:

Java

```
...
DKAttrProfileDefDD profile = aDocument.getProfile();
dkCollection fields = profile.getFields();

if((fields != null) &&( fields.cardinality() > 0 ))
{
  dkIterator itFields = fields.createIterator();
  while( itFields.more() )
  {
    DKAttrDefDD aField = (DKAttrDefDD)itFields.next();
    // ---- get the keywords
    dkCollection keywords = ((DKAttrFieldDefDD)aField).getKeywords();
    if( keywords != null )
    {
      if( keywords.cardinality() > 0 )
      {
        dkIterator itKeywords = keywords.createIterator();
        while( itKeywords.more() )
        {
          DKAttrDefDD aKeyword = (DKAttrDefDD)itKeywords.next();
          // ----- Process the keyword
        }
      }
    }
  }
}
...
```

O exemplo seguinte enumera as sub-entidades (armários, processadores de enlaces e documentos) associadas a uma entidade (neste caso uma sala).

C++

```
void printSubEnts( DKEntityDefDD* pEnt, DKDatastoreDD& domDoc, int indents )
{
    // indents: 1=Cabinets; 2=Binders; 3=Documents
    DKString indentation = "";

    for(int i = 0; i < indents; i++)
    {
        indentation += "  ";
    }

    if( pEnt->hasSubEntities() )
    {
        dkCollection* pColl = pEnt->listSubEntities();
        long nbrEnts = pColl->cardinality();
        dkIterator* itEnts = pColl-> createIterator();
        while( itEnts->more() )
        {
            DKEntityDefDD* pEnt = (DKEntityDefDD*)itEnts->next()->value();
            cout<< indentation << "SubEntity title: " << pEnt->getName() << endl;
            printSubEnts(pEnt, domDoc, indents+1);
            delete pEnt;
        }
        delete itEnts;
        delete pColl;
    }
    return;
}
```

Enumerar atributos de armário

Os armários são os únicos artigos que contêm atributos úteis. Se o utilizador tentar enumerar atributos de entidades para salas, não irá surgir nada na recolha. Consequentemente, quando DKDatastoreDD enumera entidades pesquisáveis, apenas enumera armários.

O exemplo seguinte enumera armários e os seus atributos.

Java

```
...
dkCollection cabinets = dsDD.listSearchableEntities();
dkIterator itCabinets = cabinets.createIterator();
while( itCabinets.more() )
{
    // ----- For each cabinet, list it's attributes.
    dkEntityDef aCabinet = (dkEntityDef)itCabinets.next();
    cabinetName = aCabinet.getName();
    // ----- List Document Profiles without sub-attributes
    System.out.println("\n" + Me + ": calling listAttrs for" + cabinetName );
    DKSequentialCollection coll=(DKSequentialCollection) aCabinet.listAttrs();
    ...
}
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListAttributes.java) está disponível no directório CMBROOT\Samples\java\dd.

Construir consultas em Domino.Doc

ENTITY= tem de ser a primeira palavra na cadeia de consultas caso pretenda limitar a consulta a um armário. Caso esteja em falta o parâmetro ENTITY e o seu valor, então é pesquisada toda a biblioteca. Para além disso, o valor deve estar entre aspas ("). Por exemplo, "Diane Cabinet".

QUERY= é um parâmetro necessário.

No Domino.Doc, uma cadeia de consultas surge da seguinte forma:

```
"ENTITY=<"títuloArmário"> QUERY=<"lotusQueryString">"
```

Utilize a função FTSearch para pesquisar o servidor de conteúdos do Domino.Doc. O servidor de conteúdos do Domino.Doc tem de estar totalmente indexado de texto para que esta função funcione de forma eficaz. Para testar um índice, utilize a propriedade IsFTIndexed. Para criar um índice, utilize a função UpdateFTIndex.

A função FTSearch pesquisa todos os documentos num servidor de conteúdos. Para pesquisar documentos numa vista específica, utilize a função FTSearch em NotesView. Para pesquisar documentos dentro de uma recolha de documentos específica, utilize a função FTSearch em NotesDocumentCollection.

Se não especificar uma opção de ordenação, os documentos são ordenados por relevância. Caso pretenda ordenar através da data, não irá obter resultados de relevância com os resultados ordenados. Se passar a DocumentCollection resultante para uma ocorrência de NotesNewsletter, os resultados são ordenados ou por data de criação do documento ou por resultados relevantes, dependendo das opções de ordenação que utilizar.

Utilizar sintaxe de consulta

As regras da sintaxe para uma consulta estão na lista seguinte. Utilize os parênteses para substituir a precedência e para agrupar operações.

Texto simples

Utilize texto simples para pesquisar a palavra ou expressão tal como está. Coloque as palavras-chave e símbolos de pesquisa entre apóstrofos ('). Não se esqueça de utilizar aspas (") sempre que se encontrar num literal de LotusScript.

Caracteres globais

Utilize o ponto de interrogação (?) para fazer corresponder um carácter numa posição dentro de uma palavra. Utilize o asterisco (*) para fazer corresponder zero a caracteres *n* (onde *n* é um número qualquer) em qualquer posição numa frase.

Operadores lógicos

Utilize operadores lógicos para expandir ou restringir a sua pesquisa. Os operadores e os seus precedentes são:

1. ! (não)
2. & (e)
3. , (acrécimo)
4. | (ou)

Poderá utilizar a palavra chave ou o símbolo.

Operadores de proximidade

Utilize operadores de proximidade para pesquisar palavras que estejam

próximas umas das outras. Estes operadores necessitam de interrupção de palavra, frase e parágrafo num índice de texto completo. Os operadores são:

- próximo
- frase
- parágrafo

Operador de campo

Utilize o operador de campo para restringir a sua pesquisa a um campo especificado. A sintaxe é `FIELD operador nome-campo`, onde *operador* é CONTAINS para os campos de texto e de rich text e é um dos símbolos seguintes para os campos de número e data: =, >, >=, <, <=

Operador Exactcase

Utilize o operador Exactcase para restringir a pesquisa para a expressão seguinte ao tipo de letra especificado.

Operador Termweight

Utilize o operador Termweight *n* para ajustar a ordenação por relevância da expressão que se segue, onde *n* é 0-100.

Trabalhar com Extended Search (ES)

O Enterprise Information Portal suporta IBM Extended Search 7.1, mas desactualiza o suporte para ES 3. A Extended Search permite-lhe consultar e obter documentos de:

- Bases de dados de Lotus Notes.
- Bases de dados de NotesPump.
- Sistemas de ficheiros.
- Motores de pesquisa da Web.

As classes e APIs do Enterprise Information Portal suportam as seguintes funções de Extended Search:

- Estabelecer e anular ligações a um ou mais do que um servidor de ES.
- Enumerar servidores de ES.
- Enumerar bases de dados e campos.
- Utilizar Generalized Query Language (GQL) para efectuar pesquisas.
- Obter documentos.
- Utilizar as classes e APIs de ES C++ em AIX.
- Executar pesquisas paramétricas de texto a partir do nível associado e através de uma ligação directa a ES.
- Utilizar os operadores de pesquisa de texto CONTAINS_TEXT e CONTAINS_TEXT_IN_CONTENT a partir do nível associado.
- Utilizar JavaBeans de Enterprise Information Portal.

Restrição: A ES não suporta:

- Adicionar, actualizar e eliminar documentos.
- Pesquisa do Motor de Pesquisa de Texto e QBIC.
- Consulta combinada.
- Cesto de trabalho e fluxo de trabalho.

Todas as funções da ES são acedidas e controladas pela base de dados de configuração da ES. Utilize a base de dados de configuração para atribuir definições de bases de dados para pesquisa de origens de dados, endereços de rede, informações de controlo do acesso e outras informações relacionadas com estes temas.

Enumerar servidores de Extended Search

Para fornecer acesso a vários servidores de ES, poderá criar um ficheiro intitulado `cmbdes.ini` que contém as informações do servidor. Armazene este ficheiro em `C:\CMBROOT` (em que `C` corresponde à letra da unidade). O ficheiro `cmbdes.ini` terá de conter uma linha por cada servidor, no seguinte formato:

```
DATASOURCE=TCP/IP endereço;PORT=número porta
```

em que *TCP/IP* corresponde ao endereço de TCP/IP do servidor de ES e *número porta* corresponde ao número de porta para aceder ao servidor (por exemplo (PORT=80)).

Enumerar entidades (bases de dados) e atributos (campos)

Quando construir uma consulta para pesquisar um servidor de ES, o utilizador tem de conhecer os nomes disponíveis da base de dados e do campo. O objecto de `DKDatastoreDES` fornece a função `listEntities` para enumerar as bases de dados e a função `listEntityAttrs` para enumerar os campos para cada base de dados. O exemplo seguinte demonstra como obter uma lista de bases de dados e os seus campos.

Java

```
try {
    DKSequentialCollection pCol = null;
    dkIterator pIter = null;
    DKSequentialCollection pCol2 = null;
    dkIterator pIter2 = null;
    String strDatabase = null;
    DKDatabaseDefDES dbDef = null;
    DKFieldDefDES attrDef = null;
    int i = 0;
    int j = 0;
    DKDatastoreDES dsDES = new DKDatastoreDES();
    System.out.println("connecting to datastore");
    dsDES.connect(libSrv,userid,pw,connect_string);
    System.out.println("datastore connected libSrv " + libSrv + " userid " +
        userid );
    System.out.println("list DES databases");
    pCol = (DKSequentialCollection) dsDES.listEntities();
    pIter = pCol.createIterator();
    i = 0;
    while (pIter.more() == true)
    {
        i++;
        dbDef = (DKDatabaseDefDES)pIter.next();
        strDatabase = dbDef.getName();
        System.out.println("database name [" + i + "] - " + strDatabase);
        System.out.println("list attributes for " + strDatabase + " database");
        pCol2 = (DKSequentialCollection) dsDES.listEntityAttrs(strDatabase);
        pIter2 = pCol2.createIterator();
        j = 0;
        while (pIter2.more() == true)
        {
            j++;
            attrDef = (DKFieldDefDES)pIter2.next();
            System.out.println("Attr name [" + j + "] - " + attrDef.getName());
            System.out.println("    datastoreType " + attrDef.datastoreType());
            System.out.println("    attributeOf " + attrDef.getEntityName());
            System.out.println("    type " + attrDef.getType());
            System.out.println("    size " + attrDef.getSize());
            System.out.println("    id " + attrDef.getId());
            System.out.println("    nullable " + attrDef.isNullable());
            System.out.println("    precision " + attrDef.getPrecision());
            System.out.println("    scale " + attrDef.getScale());
            System.out.println("    stringType " + attrDef.getStringType());
            System.out.println("    queryable " + attrDef.isQueryable());
            System.out.println("    displayName " + attrDef.getDisplayName());
            System.out.println("    helpText " + attrDef.getHelpText());
            System.out.println("    language " + attrDef.getLanguage());
            System.out.println("    valueCount " + attrDef.getNumVals());
        }
        System.out.println("    " + j + " attributes listed for
            " + strDatabase + " database");
    }
    System.out.println(i + " databases listed");
    dsDES.disconnect();
    System.out.println("datastore disconnected");
}
// continued...
```


Java (continuação)

```
catch(DKException exc)
{
    System.out.println("Exception name " + exc.name());
    System.out.println("Exception message " + exc.getMessage());
    System.out.println("Exception error code " + exc.errorCode());
    System.out.println("Exception error state " + exc.errorState());
    exc.printStackTrace();
}
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogDES.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
...
cout << "list entities" << endl;
pCol = (DKSequentialCollection*)((dkCollection*)dsDES.listEntities());
pIter = pCol->createIterator();
i = 0;
while (pIter->more() == TRUE)
{
    i++;
    pEnt = (DKDatabaseDefDES*)((void*)(*pIter->next()));
    strDBName = pEnt->getName();
    cout << "\ndatabase name [" << i << "] - " << strDBName << endl;
    cout << "dispname: " << pEnt->getDisplayName() << endl;
    cout << "helptext: " << pEnt->getHelpText() << endl;
    cout << "lang: " << pEnt->getLanguage() << endl;
    int iVCount = pEnt->getNumVals();
    cout << "NumValus: " << iVCount << endl;
    cout << "datatype: " << pEnt->getDataType() << endl;
    cout << "searchable:" << pEnt->isSearchable() << endl;
    cout << "retrievable" << pEnt->isRetrievable() << endl;
    cout<<"\n list attributes for "<<strDBName<<" database name"<< endl;
    pCol2 =
    (DKSequentialCollection*)((dkCollection*)dsDES.listEntityAttrs(strDBName));
    pIter2 = pCol2->createIterator();
    j = 0;
    while (pIter2->more() == TRUE)
    {
        j++;
        pA = pIter2->next();
        pAttr = (DKFieldDefDES*) pA->value();
        cout << "Attr name [" << j << "] - " << pAttr->getName() << endl;
        cout << "    datastoreName " << pAttr->datastoreName() << endl;
        cout << "    datastoreType " << pAttr->datastoreType() << endl;
        cout << "    attributeOf " << pAttr->getEntityName() << endl;
        cout << "    tipo " << pAttr->getType() << endl;
        cout << "    size " << pAttr->getSize() << endl;
        cout << "    id " << pAttr->getId() << endl;
        cout << "    nullable " << pAttr->isNullable() << endl;
        cout << "    precision " << pAttr->getPrecision() << endl;
        cout << "    scale " << pAttr->getScale() << endl;
        cout << "    string type " << pAttr->getStringType() << endl;
        cout << "    display name " << pAttr->getDisplayName() << endl;
        cout << "    help text " << pAttr->getHelpText() << endl;
        cout << "    language " << pAttr->getLanguage() << endl;
        cout << "    isQueryable " << pAttr->isQueryable() << endl;
        cout << "    isRetrievable " << pAttr->isRetrievable() << endl;
        delete pAttr;
    }
    cout << " " << j << " attributes listed for "
        << strDBName << " database name" << endl;
    delete pIter2;
    delete pCol2;
    delete pEnt;
}
delete pIter;
delete pCol;
cout << i << " entities listed\n" << endl;
...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TListCatalogDES.cpp) está disponível no directório Cmbroot/Samples/cpp/ES.

Utilizar a Generalized Query Language (GQL)

A Extended Search utiliza a Generalized Query Language (GQL) para executar pesquisas. Tabela 20 contém exemplos de expressões de GQL válidas.

Tabela 20. Expressões de GQL

| Expressão de GQL | Descrição |
|-----------------------------|--|
| "software" | Pesquise documentos que contenham a palavra software. |
| (TOKEN:WILD "exec*") | Pesquise documentos que contenham qualquer palavra que comece por exec. |
| (AND "software" "IBM") | Pesquise documentos que contenham as palavras software e IBM. |
| (START "View" "How") | Pesquise documentos nos quais o campo Vista inicie com a palavra How. |
| (EQ "View" "How Do I?") | Pesquise documentos nos quais o campo Vista contenha a cadeia exacta How Do I?. |
| (GT "BIRTHDATE" "19330804") | Pesquise documentos nos quais o campo BIRTHDATE seja superior a 4 de Agosto de 1933. |

A ES utiliza o tipo de consulta DK_DES_GQL_QL_TYPE. Este tipo de consulta tem a sintaxe seguinte:

```
SEARCH=(DATABASE=(db_name | db_name_list | ALL);  
           COND=(GQL_expression));  
[OPTION=([SEARCHABLE_FIELD=(fd_name, ...);]  
          [RETRIEVABLE_FIELD=(fd_name, ...);]  
          [MAX_RESULTS=maximum_results];  
          [TIME_LIMIT=time])]
```

em que db_name_list corresponde a uma lista de nomes de bases de dados (db_name) separados por vírgulas e ALL corresponde a todas as bases de dados disponíveis. O tempo limite predefinido para uma pesquisa é de 30 segundos.

Este exemplo utiliza a cadeia de consultas para pesquisar documentos na base de dados Ajuda de Notas, em que o campo Vista é How Do I? e o máximo de resultados esperados é cinco.

```
String cmd = "SEARCH=(DATABASE=(Notes Help);" +  
             "COND=(EQ \"View\" \"How Do I?\");" +  
             "OPTION=(MAX_RESULTS=5)"
```

Este exemplo executa uma consulta de GQL em ES. Após a execução da consulta, os resultados são devolvidos num objecto de dkResultSetCursor.

Java

```
DKDatastoreDES dsDES = new DKDatastoreDES();
dkResultSetCursor pCur = null;
DKNameValuePair parms[] = null;
System.out.println("connecting to datastore");
dsDES.connect(libSrv,userid,pw,connect_string);
System.out.println("datastore connected libSrv "+libSrv+" userid "+userid);
String cmd = "SEARCH=(DATABASE=(Notes Help);" +
             "COND=(EQ \"View\" \"How Do I?\");" +
             "OPTION=(MAX_RESULTS=5)";
DKDDO ddo = null;
System.out.println("query string " + cmd);
System.out.println("executing query");
pCur = dsDES.execute(cmd,DK_DES_GQL_QL_TYPE,parms);

System.out.println("cardinality " + pCur.cardinality());
System.out.println("datastore executed query");
System.out.println("process query results");
...
pCur.destroy(); // Finished with the cursor
System.out.println("query results processed");
dsDES.disconnect();
System.out.println("datastore disconnected");
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDES.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDES dsDES;
dkResultSetCursor* pCur = 0;
cout << "Datastore ES created" << endl;
cout << "connecting to datastore" << endl;
dsDES.connect(libsrv,userid,pw,str);
cout << "datastore connected " << libsrv << " userid - " << userid << endl;

DKString cmd = "SEARCH=(DATABASE=(Notes Help));";
cmd += "COND=((IN \"Subject\" \"your\"));";
cmd += "OPTION=(MAX_RESULTS=2;TIME_LIMIT=10);";

cout << "query string " << cmd << endl;
cout << "executing query" << endl;
pCur = dsDES.execute(cmd);
cout << "query executed" << endl;
...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDES.cpp) está disponível no directório Cmbroot/Samples/cpp/ES.

Identificar o artigo de tipo de DDO em Extended Search

Um DDO na ES tem sempre o tipo DK_CM_DOCUMENT. Para obter o tipo de artigo do DDO, deverá chamar:

Java

```
Object obj = ddo.getPropertyByName(DK_CM_PROPERTY_ITEM_TYPE);  
short type = ((Short) obj).shortValue();
```

C++

```
DKDDO *p = 0;  
ushort k = p->propertyId(DK_CM_PROPERTY_ITEM_TYPE);  
if (k > 0)  
{  
    DKAny a = p->getProperty(k);  
    ushort val = a; // val = DK_CM_DOCUMENT  
}
```

Criar PIDs na Extended Search

O identificador permanente (PID) contém informações específicas sobre um documento. O tipo de objecto identifica a base de dados em que se localizou o documento. É criado um PID através da utilização no nome da base de dados, seguido do carácter | e do ID do documento, por exemplo:

database name|documentId ()

Para obter mais informações sobre PIDs consulte “Compreender identificadores permanentes (PID)” na página 14 e “Criar um identificador permanente (PID)” na página 44.

Processar os conteúdos de um documento de Extended Search

Cada artigo no DDO representa um campo, uma recolha ou um objecto de DKParts.

Campo

O nome do campo para um único campo está dentro do nome do artigo. O valor do campo também está dentro do valor do artigo. A propriedade do campo pode ser:

- DK_CM_VSTRING
- DK_CM_FLOAT
- DK_CM_XDOOBJECT
- DK_CM_DATE
- DK_CM_SHORT

Recolha

Quando um campo tem vários valores, o nome do campo está no nome do artigo. O valor do artigo é um objecto de DKSequentialCollection. A propriedade pode ser DK_CM_COLLECTION ou DK_CM_COLLECTION_XDO caso o campo seja um BLOB.

DKParts

Um DDO de documento possui um atributo específico com um nome reservado DKPARTS. O seu valor é um objecto de DKParts. O objecto de DKPARTS pode armazenar as informações de endereço da Web (URL) relativas a um documento. DKPARTS também pode conter um XDO cuja conteúdo é uma cadeia que representa o URL de um documento.

Este exemplo processa o conteúdo de um DDO:

Java

```
public static void displayDDO(DKDDO ddo)
    throws DKException, Exception
{
    dkXDO pXDO = null;
    int i = 0;
    int numDataItems = 0;
    short k = 0;
    short j = 0;
    Integer valueCount = null;
    Object value = null;
    String dataName = null;
    dkCollection pCol = null;
    dkIterator pIter = null;
    Object anObject = null;
    numDataItems = ddo.dataCount();
    DKPid pid = ddo.getPidObject();
    System.out.println("pid string " + pid.pidString());

    System.out.println("Number of Attributes " + numDataItems);
    for (j = 1; j <= numDataItems; j++)
    {
        anObject = ddo.getData(j);
        dataName = ddo.getDataName(j);
        System.out.println(j + ": Name " + dataName );
        // determine if data item has a single value or multiple values
        Short type = (Short)ddo.getDataPropertyByName(j, DK_CM_PROPERTY_TYPE);
        k = type.shortValue();
        if (k == DK_CM_COLLECTION)
        {
            {
                pCol = (dkCollection)anObject;
                pIter = pCol.createIterator();
                i = 0;
                while (pIter.more() == true)
                {
                    i++;
                    value = pIter.next();
                    System.out.println("    Value" + i + " " + value);
                }
            }
        }
        else if (k == DK_CM_COLLECTION_XDO)
        {
            {
                pCol = (dkCollection)anObject;
                pIter = pCol.createIterator();
                i = 0;
                while (pIter.more() == true)
                {
                    i++;
                    blob = (DKBlobDES)pIter.next();
                    System.out.println("    Value" + i + " " + blob.getContent());
                }
            }
        }
        else
        {
            System.out.println("    Value " + anObject);
        }
    }
}
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDES.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDDO *p = 0;
dkDataObjectBase *pDOBase = 0;
DKDDO *pDDO = 0;
dkXDO *pXDO = 0;
DKAny a;
ushort j = 0;
ushort k = 0;
ushort val = 0;
ushort cnt = 1;
DKString strData = "";
DKString strDataName = "";
dkCollection* pdCol = 0;
dkIterator* pdIter = 0;
ushort numDataItems = 0;
DKPidXDOES *pidXDO = 0;
DKPid *pid = 0;
DKString strPid;
long pidIdCnt = 0;
long pidIndex = 0;
while (pCur->isValid())
{
    p = pCur->fetchNext();
    if (p != 0)
    {
        cout << "=====" << "Item " << cnt << " <=====" << endl;
        numDataItems = p->dataCount();
        pid = (DKPid*)p->getPidObject();
        strPid = pid->pidString();
        cout << "pid string " << strPid << endl;
        cout << "pid id string " << pid->getId() << endl;
        strPid = pid->getIdString();
        cout << "pid idString " << strPid << endl;
        pidIdCnt = pid->getIdStringCount();
        cout << "pid idString cnt " << pidIdCnt << endl;
        strPid = pid->getPrimaryId();
        cout << "pid primary id " << strPid << endl;
        pidIndex = 0;
        strPid = pid->getIdString(pidIndex);
        cout << "pid item id " << strPid << endl;
        k = p->propertyId(DK_CM_PROPERTY_ITEM_TYPE);
        if (k > 0)
        {
            a = p->getProperty(k);
            val = a;
            cout << "*****" << endl;
            switch (val)
            {
                case DK_CM_DOCUMENT :
                    cout << "Item is a document " << endl;
                    break;
            default:
                cout << " Item is not recognized " << endl;
            break;
            }
            cout << "*****" << endl;
        }
        cout << "Number of Data Items " << numDataItems << endl;
        for (j = 1; j <= numDataItems; j++)
        {
            a = p->getData(j);
            strDataName = p->getDataName(j);
        }
    }
}
// continued...
```

C++ (continuação)

```
        switch (a.typeCode())
        {
        case DKAny::tc_string :
        {
            strData = a;
            cout << "attribute name : " << strDataName << " value : "
                  << strData << endl;
        }
            break;
        case DKAny::tc_long :
        {
            long l = a;
            cout << "attribute name : " << strDataName << " value : " << l << endl;
        }
            break;
        case DKAny::tc_double :
        {
            double db = a;
            cout << "attribute name : " << strDataName << " value : " << db << endl;
        }
            break;
        case DKAny::tc_timestamp :
        {
            DKTimestamp tt = a;
            cout << "attribute name : " << strDataName << " value : "
                  << tt.getMonth() << "/" << tt.getDay() << "/" << tt.getYear() << " "
                  << tt.getHours() << ":" << tt.getMinutes() << ":" << tt.getSeconds() << endl;
        }
            break;
        case DKAny::tc_dobase :
        {
            pDOBase = a;
            pXDO = (dkXDO*)pDOBase;
            cout << "attribute name : " << strDataName << " value : " << endl;
            pidXDO = (DKPidXDOES*)pXDO->getPid();
            cout << "XDO pid database name " << pidXDO->getDatabaseName() << endl;
            cout << "XDO pid docId " << pidXDO->getDocId() << endl;
            cout << "XDO mimetype " << pXDO->getMimeType() << endl;
            ((DKBlobDES*)pXDO)->getContentToClientFile("c:\\temp\\temp.html", 1);
        }
            break;
        case DKAny::tc_collection :
        {
            pdCol = a;
            cout << strDataName << " collection name : " << strDataName << endl;
            cout << "-----" << endl;
            pdIter = pdCol->createIterator();
            ushort b = 0;
            while (pdIter->more() == TRUE)
            {
                b++;
                cout << "-----" << endl;
                a = *(pdIter->next());
                switch (a.typeCode())
                {
                case DKAny::tc_string :
                {
                    strData = a;
                    cout << "attribute name : " << strDataName << " value : " << strData << endl;
                }
                    break;
                }
            }
        }
        // continued...
```


C++ (continuação)

```
        case DKAny::tc_long :
        {
            long l = a;
            cout << "attribute name : " << strDataName << " value : " << l << endl;
        }
        break;
        case DKAny::tc_double :
        {
            double db = a;
            cout << "attribute name : " << strDataName << " value : " << db << endl;
        }
        break;
        case DKAny::tc_timestamp :
        {
            DKTimestamp tt = a;
            cout << "nome atributo : " << strDataName << " valor : "
            << tt.getMonth() << "/" << tt.getDay() << "/" << tt.getYear() << " "
            << tt.getHours() << ":" << tt.getMinutes() << ":" << tt.getSeconds() << endl;
        }
        break;
        case DKAny::tc_dobase :
        {
            pDOBase = a;
            pXDO = (dkXDO*)pDOBase;
            cout << "attribute name : " << strDataName << " value : " << endl;
            pidXDO = (DKPidXDOES*)pXDO->getPid();
            cout << "XDO pid database name " << pidXDO->getDatabaseName() << endl;
            cout << "XDO pid docId " << pidXDO->getDocId() << endl;
            cout << "XDO mimetype " << pXDO->getMimeType() << endl;
            DKString str = "c:\\temp\\temp";
            DKString strT = b;
            str = str + strT + ".html";
            ((DKBlobDES*)pXDO)->getContentToClientFile(str, 1);
        }
        break;
    }
    ushort usCount = p->dataPropertyCount(j);
    for (ushort k = 1; k <= usCount; k++)
    {
        a = p->getDataProperty(j, k);
        cout << " property " << k << " " << a << endl;
    }
    if (b == 0)
    {
        cout << strDataName << " collection has no elements " << endl;
    }
    cout << " -----" << endl;
    break;
}
}
ushort usCount = p->dataPropertyCount(j);
for (ushort k = 1; k <= usCount; k++)
{
    a = p->getDataProperty(j, k);
    cout << " property " << k << " " << a << endl;
}
}
cnt++;
delete p;
```

A aplicação exemplo completa da qual este exemplo foi retirado (TExecuteDES.cpp) está disponível no directório Cmbroot/Samples/cpp/ES.

Obter um documento

Para obter um documento de um objecto de DKDatastoreDES, o utilizador terá de saber o nome da base de dados que contém o documento e o ID do documento. Também terá de associar o DDO a um servidor de conteúdos e estabelecer uma ligação. Este exemplo obtém um documento:

Java

```
DKDatastoreDES dsDES = new DKDatastoreDES();
dsDES.connect(libSrv, userid, pw, connect_string);
DKPid pid = new DKPid();
// ----- Primary ID is 'database name' followed by the
//          character '|' followed by the document ID
pid.setPrimaryId("Notes Help" + DK_DES_ITEMID_SEPARATOR + "215e");
pid.setObjectType("Notes Help");
DKDDO ddo = new DKDDO(dsDES, pid);
ddo.retrieve();
```

A aplicação exemplo completa da qual este exemplo foi retirado (TRetrieveDDODES.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDES dsDES;
dkResultSetCursor* pCur = 0;
cout << "Datastore ES created" << endl;
cout << "connecting to datastore" << endl;
dsDES.connect(libsrv,userid,pw,str);
cout << "datastore connected " << libsrv << " userid - " << userid << endl;
...
p = new DKDDO(&dsDES, "");
DKPid pid2;
pid2.setDatastoreType(dsDES.datastoreType());
pid2.setDatastoreName(dsDES.datastoreName());
pid2.setId("Notes Help|215e");
pid2.setObjectType("");
p->setPidObject((DKPid*)&pid2);
p->retrieve();
...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TRetrieveDDODES.cpp) está disponível no directório Cmbroot/Samples/cpp/ES.

Obter um objecto binário grande (BLOB)

Para obter um BLOB de um objecto deDKDatastoreDES, deve saber o nome da base de dados na qual o documento reside, o ID de documento que contém o BLOB e o nome de campo que contém o BLOB. Também terá de associar o DDO a um servidor de conteúdos e estabelecer uma ligação.

No exemplo que se segue, a base de dados do sistema de ficheiros intitulada ficheiros de ES contém um ficheiro de HTML intitulado D:\desdoc\README.html. O campo que contém o ficheiro HTML é intitulado Doc\$Content. O exemplo seguinte obtém o ficheiro HTML e guarda-o como D:\DESReadme.html .

Java

```
DKDatastoreDES dsDES = new DKDatastoreDES();
dsDES.connect(libSrv, userid, pw, connect_string);
DKBlobDES xdo = new DKBlobDES(dsDES);
DKPidXDOES pid = new DKPidXDOES();
pid.setDocumentId("D:\\desdoc\\README.html");
pid.setDatabaseName("DES files");
pid.setFieldName("Doc$Content");
xdo.setPidObject(pid);
xdo.retrieve("c:\\DESReadme.html");
```

A aplicação exemplo completa da qual este exemplo foi retirado (TRetrieveXDOES.java) está disponível no directório CMBROOT\Samples\java\d1.

C++

```
DKDatastoreDES dsDES;
dkResultSetCursor* pCur = 0;
cout << "Datastore ES created" << endl;
cout << "connecting to datastore" << endl;
dsDES.connect(libsrv,userid,pw,str);
    cout << "datastore connected " << libsrv << " userid - " << userid << endl;
...
cout << "executing retrieve a XDO" << endl;

DKBlobDES* p = new DKBlobDES(&dsDES);
DKPidXDOES pid;
pid.setDocId("D:\\desdoc\\README.html");
pid.setDatabaseName("ES files");
pid.setFieldName("Doc$Content");
pid.setPrimaryId("ES files|D:\\desdoc\\README.html");
p->setPidObject((DKPidXDO*)&pid);

p->retrieve("c:\\temp\\DESReadme.html");

cout << "retrieve executed" << endl;
...
```

A aplicação exemplo completa da qual este exemplo foi retirado (TRetrieveXDOES.cpp) está disponível no directório Cmbroot/Samples/cpp/ES.

Associar tipos de MIME com documentos

ES não suporta directamente a identificação de tipos de Multipurpose Internet Mail Extension (MIME). No entanto, o utilizador terá de saber o tipo de MIME de um XDO que pretenda apresentar dentro de um browser da Web.

O ficheiro CMBCC2MIME.INI é utilizado para determinar o tipo de MIME de um documento. Quando uma consulta de ES de bases de dados NotesPump ou FileSystem devolve um BLOB, é pesquisado o ficheiro CMBCC2MIME.INI para determinar se um tipo de MIME pode ser atribuído ao BLOB. O tipo de MIME predefinido é text/html. Está disponível um ficheiro exemplo cmbcc2mime.ini.samp no directório de exemplos.

Utilizar pesquisa associada em Extended Search

Quando criar consultas associadas, a sintaxe utilizada na ES é igual à sintaxe de SQL. As expressões da consulta associada são convertidas para sintaxe de GQL antes de serem submetidas à ES. Visto que, apesar de tudo, a gramática de SQL e de GQL têm diferenças, só é suportado pelo Enterprise Information Portal um subconjunto da gramática de SQL.

O Tabela 21 resume a conversão de SQL para GQL da comparação suportada e dos operadores lógicos.

Tabela 21. Operadores de SQL e GQL

| Operador de SQL | Operador de GQL |
|-----------------|-----------------|
| AND | AND |
| OR | OR |
| NOT | não suportado |
| IN | não suportado |
| BETWEEN | BETWEEN |
| EQ | EQ |
| NEQ | não suportado |
| GT | GT |
| LT | LT |
| LIKE | não suportado |
| GEQ | GTE |
| LEQ | LTE |
| NOTLIKE | não suportado |
| NOTIN | não suportado |
| NOTBETWEEN | não suportado |

Trabalhar com Panagon Image Services (Apenas para Java)

Pode utilizar as classes de API de Enterprise Information Portal facultadas para Panagon Image Services (FileNET) para aceder a conteúdos em servidores de Panagon Image Services. O suporte para Panagon Image Services só se encontra disponível como uma função especial do EIP.

Restrição: O Panagon Image Services não suporta Motor de Pesquisa de Texto e pesquisa QBIC ou consultas combinadas.

Modelar dados

O Enterprise Information Portal possui alguns conceitos de modelação de dados que devem ser correlacionados aos seus objectos correspondentes no servidor de conteúdos que um conector suporta. A tabela seguinte mostra os modelos de dados de EIP no ambiente de FileNET.

| Conceito de EIP | FileNET |
|-----------------------|--|
| servidor de conteúdos | servidor do Panagon Image Services (FileNET) |
| servidor | servidor |
| entidade | classe de documento |

| Conceito de EIP | FileNET |
|------------------------------------|--|
| atributo | índice |
| DDO | documento (pasta) As pastas em FileNET são utilizadas para organizar temporariamente documentos; não estão actualmente modeladas no conector. |
| XDO | conteúdo de página, anotação |
| PID para um DDO | document_id, como o ID primário |
| PID para XDO de conteúdo de página | System_serial_num + document_id + page# |
| PID para XDO de anotação | System_serial_num + document_id + page# + annotation_id |

O conector de FileNET só devolve índices ou atributos definidos pelo utilizador em FileNET, que podem ser utilizado em correlações de entidade e atributo federados e em listas de visualização de atributos do resultado de pesquisa. A seguinte tabela mostra a correlação de tipos de dados de índice e atributo suportados com os tipos de dados de atributo de EIP correspondentes no Conector de FileNET:

| Tipo de dados de índice de FileNET | Tipo de dados de atributo de EIP |
|------------------------------------|---|
| numérico | double |
| data | varchar com max_length definido para o comprimento máximo definido para o índice de FileNET |
| menu | varchar com max_length = 14 (máximo para nomes de menus) |

Cada XDO em EIP possui um tipo MIME. Os clientes de EIP utilizam o tipo MIME para determinar como visualizar o documento. O Conector de FileNET parte do princípio que todos os XDOs do mesmo documento pertencem ao mesmo tipo MIME. O tipo MIME do XDO é definido de acordo com o valor de atributo F_DOCFORMAT do sistema de FileNET do documento correspondente. Se o valor for nulo, o tipo MIME é então definido para image/tiff se o atributo F_DOCTYPE de sistema FileNET indicar o tipo de imagem; caso contrário, é definido para o tipo MIME predefinido de application/octet-stream.

Documentos e páginas em Panagon Image Services

Para utilizar um DDO que represente um documento em Panagon Image Services, devem ser definidos correctamente quatro campos no PID do DDO:

1. Tipo de Armazenamento de Dados – esta é a constante DK_FN_DSTYPE definida em DKConstantFN. Pode obter este valor do método dataStoreType() de DKDataStoreFN.
2. Nome de Armazenamento de Dados – este é o nome de organização de domínios de FileNET. Pode obter este valor do método dataStoreName() de DKDataStoreFN.
3. Tipo de Objecto – este é o nome de classe de documento de FileNET (nome de entidade nativa em EIP) ao qual o documento pertence.
4. ID Primário – este é o número de documento de FileNET, atribuído pelo FileNET.

Após o PID ser definido, os programas de aplicação podem obter atributos ou conteúdos do DDO chamando o método `retrieve()` na classe `DKDatastoreFN` ou `DKDDO`.

Um DDO de documento em EIP tem um atributo específico com um nome reservado `DKPARTS`, cujo valor é um objecto de `DKParts`. Um objecto de `DKParts` é uma recolha de objectos volumosos binários (BLOBs). As partes de FileNET (páginas/anotações) dentro de um documento são representadas como objectos de `DKBlobFN` (uma sub-classe de `DKXDO`), que são elementos da recolha `DKParts`.

Uma página de FileNET não é uma página convencional; um documento de uma única página (ou única parte) em FileNET pode conter várias páginas físicas, mas estas são armazenadas numa única página dentro do documento de FileNET. Um documento de várias páginas em FileNet faz referência um documento que contém várias partes ou vários ficheiros que são criados utilizando o Panagon Capture Software ou o Batch Entry System.

Enumerar entidades e atributos

O utilizador irá representar um servidor Panagon Image Services como um `DKDatastoreFN`. Após criar o servidor de conteúdos e de lhe estabelecer ligação, pode enumerar as entidades (documentos x classes) e atributos do servidor de Panagon Image Services. O exemplo seguinte ilustra como se executa esta acção:

Java

```
DKDatastoreFN dsFN = null;
try {
    DKSequentialCollection pCol = null;
    dkIterator pIter = null;
    DKSequentialCollection pCol2 = null;
    dkIterator pIter2 = null;
    DKServerDefFN pSV = null;
    String strServerName = null;
    String strServerType = null;
    String strEntity = null;
    DKEntityDefFN entityDef = null;
    DKAttrDefFN attrDef = null;
    int i = 0;
    int j = 0;
    dsFN = new DKDatastoreFN();
    // ----- Connect to the server using the server name, user ID , and password
    dsFN.connect(libSrv, userid, pw, "");
    System.out.println("Datastore connected libSrv " + libSrv + " userid "
        + userid );
    System.out.println("List entities in server " + libSrv);
    pCol = (DKSequentialCollection) dsFN.listEntities();
    pIter = pCol.createIterator();
    i = 0;
    while (pIter.more() == true)
    {
        i++;
        entityDef = (DKEntityDefFN)pIter.next();
        strEntity = entityDef.getName();
        System.out.println("\nEntity name [" + i + "] - " + strEntity +
            ", id = " + entityDef.getId() + ", type = " + entityDef.getType());
        System.out.println(" List attr for the " + strEntity + " entity");
        pCol2 = (DKSequentialCollection) dsFN.listEntityAttrs(strEntity);
        pIter2 = pCol2.createIterator();
        j = 0;
        while (pIter2.more() == true)
        {
            j++;
            attrDef = (DKAttrDefFN)pIter2.next();
            System.out.println("    Attribute name [" + j + "] - " +
                attrDef.getName());
            System.out.println("    datastoreType = " + attrDef.datastoreType());
            System.out.println("    attributeOf = " + attrDef.getEntityName());
            System.out.println("    type = " + attrDef.getType());
            System.out.println("    size = " + attrDef.getSize());
            System.out.println("    id = " + attrDef.getId());
            System.out.println("    nullable = " + attrDef.isNullable());
            System.out.println("    precision = " + attrDef.getPrecision());
            System.out.println("    scale = " + attrDef.getScale());
            System.out.println("    stringType = " + attrDef.getStringType());
        }
        System.out.println(" Total of " + j + " attributes listed for the "
            + strEntity + " entity");
    }
    System.out.println("\nTotal of " + i + " entities listed for server "
        + libSrv);
    // ----- Disconnect and clean up
    dsFN.disconnect();
    dsFN.destroy();
}
catch(DKException exc)
```

Consulte o directório CMBROOT\Samples\java\fn para obter aplicações exemplo.

Consultar

O EIP suporta três formatos básicos de consulta para Panagon Image Services:

- objecto de consulta
- cadeia de comando
- DKCQExpr

Tanto o formato de objecto de consulta como o formato de cadeia de comando da consulta são normalmente utilizados em aplicações que estão a aceder directamente ao servidor de Panagon Image Services. O servidor de conteúdos e a consulta associados de EIP utilizam apenas a forma DKCQExpr da consulta para servir de interface entre os servidores de conteúdo individuais.

As aplicações exemplo para consultar servidores de Panagon Image Services encontram-se disponíveis no directório CMBROOT\Samples\java\fn.

Sintaxe e Parâmetros da Cadeia de Consulta

O Conector de FileNET suporta a utilização de DKParametricQuery para submeter e executar uma consulta. O DKParametricQuery pode ser construído com uma cadeia de comandos. A cadeia de comandos representa a especificação de consulta para a condição de filtro de FileNET. Outras opções de consulta (ENTITY_NAME, MAX_RESULTS, CONTENT) e condição de chave FileNET (KEY) serão aceites como parâmetros para os métodos evaluate() ou execute() em pares (nome, valor) (e.g. DKNVPair).

Tenha em atenção que os valores são sempre de tipo String e os nomes dos parâmetros são definidos em DKConstant e DKConstantFN. Os resultados de retorno são uma recolha de DDOs de documento para evaluate() e o cursor de conjunto de resultados indicando os resultados de execute().

A sintaxe de expressão condicional do comando e para o parâmetro KEY segue a sintaxe de consulta de FileNET suportada pela função FileNET WAL PRS_Parse(). A sintaxe aceita os seguintes operadores:

| Operador | Utilizar |
|----------|----------------------|
| AND | and lógico |
| OR | or lógico |
| NOT | not lógico |
| < | menor que |
| <= | menor que ou igual a |
| = | igual a |
| > | maior que |
| >= | maior que ou igual a |
| != | não igual a |
| + | soma |
| - | subtração |
| * | multiplicação |
| / | divisão |

| Operador | Utilizar |
|---------------------|--|
| IN RANGE | é especificado um intervalo com a sintaxe IN RANGE <i>opconstantopconstant</i> , em que <i>op</i> é <, <=, >, ou >= e <i>constant</i> é uma constante numérica ou de cadeia |
| LIKE | Idêntico a. O padrão do lado direito do operador LIKE deve ser uma cadeia de caracteres entre plicas que pode conter ? como um carácter global (para indicar correspondência em qualquer carácter) e * como carácter global. |
| DEFINED(<i>x</i>) | A função definida devolve um valor diferenrte de zero se o nome de coluna <i>x</i> não for nulo e zero se o valor for nulo. |

Os operandos são constantes numéricas, constantes de cadeia e nomes de atributo. Visto que a cadeia de comandos é utilizada para pesquisar directamente o servidor de conteúdos de FileNET, os nomes dos atributos são todos nomes de atributo/índice FileNET locais. Não seriam suportados, nem necessário, quaisquer nomes de atributos associados ou correlacionamento de esquemas

Os números encontram-se no formato *ddd.dddE+eee* em que *ddd* representa 0 ou mais dígitos numéricos, + representa + ou - (+ é facultativo), *eee* representa 0 para três dígitos do expoente, e . é um ponto decimal (facultativo) e E é o início do expoente (facultativo).

As constantes de cadeia são colocadas entre plicas e a plica incorporada dentro da cadeia é representada por duas plicas seguidas.

A expressão condicional para a cadeia de comandos não possui um número definido de operandos e operadores que pode conter. Também podem ser utilizados parêntesis no filtro para sobrepor a precedência. Os exemplos de condições de filtro são os seguintes:

```
PROD_PRICE > 100 AND (PROD_DATE > '10/15/92' OR PROD_NAME LIKE 'comp*')
PROD_PRICE >= 100 AND (NOT (PROD_ID < 30))
```

A expressão condicional do parâmetro KEY deve conter apenas um nome de atributo, que é definido como uma chave de obtenção em FileNET, e uma condição. Tenha em atenção que os nomes de chave definidos em FileNET devem ser utilizados e não pode ser utilizado um nome de coluna não invertida. Os exemplos de cadeias de chave são:

```
PROD_PRICE = 100000
PROD_PRICE IN RANGE >= 100000 <= 200000
```

Opções de pesquisa adicionais

Os restantes parâmetros opcionais suportados para evaluate() e execute() são:

(ENTITY_NAME, entity_Name):

Especifique o nome da entidade (nome de classe de documento FileNET) como âmbito da consulta. Exemplo:

```
parms[1] = new DKNVPair (DK_FN_PARM_ENTITY_NAME, entity_Name);
```

A cadeia de consulta que é submetida para o servidor de Panagon Image Services é anexada a AND (F_DOCCLASSNUMBER = doc_cls#) , em que = doc_cls# é o número de classe de documento correspondente do nome de entidade especificado. Se não for especificado, todos os documentos em todas as classes de documento de FileNET.

(MAX_RESULTS, Max_results)

Apenas o número máximo especificado de resultados é devolvido na recolha de resultados. Se o valor for zero ou não especificado, é devolvido o número máximo de resultados permitido por FileNET. Exemplo:

```
parms[0] = new DKNVPair(DK_CM_PARM_MAX_RESULTS, Max_results);
```

(CONTENT, YES / NO / ATTRONLY)

Este parâmetro irá sobrepor a opção DK_CM_OPT_CONTENT definida ao nível do servidor de conteúdos.

- YES- serão obtidos os conteúdos dos documentos resultantes (predefinição)
- NO - apenas os IDs de documento são definidos nos DDOs do resultado
- ATTRONLY - os IDs de documento, os atributos de dados e os seus valores são definidos nos DDOs do resultado

Exemplo:

```
dsFN.setOption(DK_CM_OPT_CONTENT, DK_CM_CONTENT_YES);
```

Por exemplo, o valor de conteúdo YES levaria a que os DDOs de documento resultantes tivessem definido o seu PID, tipo de objecto, propriedades e todos os atributos. Esta opção também gera a definição do atributo DKPARTS para uma recolha de partes de conteúdo (XDOs para partes e suas anotações) com informações de PID definidas, mas sem conteúdo. Se o valor CONTENT for ATTRONLY, são definidos o PID, tipo de objecto, propriedades e todos os atributos dos DDOs de documento resultantes. Se o valor CONTENT for NO, são definidos o PID, tipo de objecto e propriedades dos DDOs de documento resultantes. A parte do documento ou conteúdo de XDO pode ser obtido explicitamente quando necessário, utilizando o método retrieve() da classe DKBlobFN.

Manuseamento de excepções e mensagens

O conector de Panagon Image Services emprega o mesmo conjunto de classes de excepção Java definidas em EIP. A mensagem de texto incluída numa excepção também utiliza, sempre que possível, uma das mensagens normalmente definidas em DKMessageID. Por exemplo, a excepção DKUsageError inclui uma mensagem como (DKMessage.getMessage(DK_CM_MSG_NOTIMP) + this.getClass().getName() + XXXX , DK_CM_MSG_NOTIMP); em que XXX é o nome do método que o programa está a tentar invocar.

A excepção DKDatastoreAccessError irá excluir o texto de mensagem específico de FileNET no seguinte formato:

```
WAL_function_name [IMS_SESSION= sss, ERR_CAT=ccc, ERR_FUN=fff, ERR_NUM=nnn]
```

Pode utilizar o comando msg de FileNET (acessível através do directório c:\fns\client\bin) para visualizar o texto de mensagem de erro nativo de FileNET utilizando os três números *ccc*, *fff* e *nnn* (e.g., msg *ccc,fff,nnn*).

Os IDs de mensagem de EIP utilizados unicamente para os servidores de Panagon Image Services podem ir de 3401 a 3600.

Trabalhar com bases de dados relacionais

As classes de API do Enterprise Information Portal suportam IBM DB2 Universal Database e outras bases de dados relacionais que utilizam Java Database Connectivity (JDBC), Open Database Connectivity (ODBC) for C++, ou IBM DB2 DataJoiner.

Ligação a bases de dados relacionais

Para representar uma base de dados relacional, crie um objecto de `DKDatastorexx`, em que `xx` corresponde a DB2 para uma base de dados DB2, JDBC for Java Database Connectivity ou ODBC for Open Database Connectivity. O exemplo seguinte estabelece ligação à base de dados exemplo de DB2:

Java

```
dsDB2 = new DKDatastoreDB2();
dsDB2.connect("sample","db2admin","password","");
.....
dsDB2.disconnect();
dsDB2.destroy();
```

C++

```
try {      DKDatastoreDB2 dsDB2;
          dsDB2.connect("sample", userid, pw);

          . . .
          dsDB2.disconnect();
        }
        catch(DKException &exc) . . .
```

Utilize o nome da base de dados ao estabelecer ligação.

Cadeias de ligação

Ao estabelecer ligação a uma base de dados relacional, pode especificar uma cadeia de ligação e transmiti-la como um parâmetro. Se especificar várias cadeias de ligações, separe-as com ponto e vírgula (;). As cadeias de ligação podem assumir os seguintes formatos:

Cadeias de ligação para DB2, DataJoiner, e ODBC:

`NATIVECONNECTSTRING=(cadeia de ligação nativa)`

Especifica uma cadeia de ligação nativa que vai ser transmitida à base de dados aquando da ligação. Verifique as informações do servidor de conteúdos para determinar as cadeias de ligações nativas válidas.

`SCHEMA=nome do esquema`

Especifica o nome do esquema da base de dados que vai ser utilizado ao executar os métodos `listEntities`, `listEntityAttrs`, `listPrimaryKeyNames`, `listForeignKeyNames`.

Cadeias de ligação para JDBC:

`SCHEMA=nome do esquema`

Especifica o nome do esquema da base de dados que vai ser utilizado ao executar os métodos `listEntities`, `listEntityAttrs`, `listPrimaryKeyNames`, `listForeignKeyNames`.

Cadeias de configuração

Pode especificar uma cadeia de configuração e transmiti-la como um parâmetro ao método de configuração `of`. Se especificar várias cadeias de configuração, separe-as com ponto e vírgula (;). As cadeias de configuração podem assumir os seguintes formatos:

As cadeias de configuração para DB2, DataJoiner, e ODBC:

CC2MIMEURL=(URL)

Especifica o ficheiro cmbcc2mime.ini como um endereço de localizador de serviços uniformes. Utilize este formato da cadeia de configuração ou CC2MIMEFILE, dependendo da localização do ficheiro.

CC2MIMEFILE=(nomeficheiro)

Especifica o ficheiro cmbcc2mime.ini pelo nome.

DSNAME=(nome do servidor de conteúdos)

Especifica o nome do servidor de conteúdos. Para consultas associadas e outras funções associadas, o Enterprise Information Portal define esta acção automaticamente.

AUTO COMMIT=ON | OFF

Liga ou desliga a consolidação automática. A predefinição é desligada. Quando este servidor de conteúdos é utilizado para consultas associadas e outras funções associadas, a autoconsolidação é activada por predefinição.

Cadeias de configuração para JDBC:

CC2MIMEURL=(URL)

Especifica o ficheiro cmbcc2mime.ini como um endereço de localizador de serviços uniformes. Utilize este formato da cadeia de configuração ou CC2MIMEFILE, dependendo da localização do ficheiro.

Especifica o ficheiro cmbcc2mime.ini pelo nome.

JDBC SERVER URL=(URL)

Especifica o ficheiro cmbjdbcsvrs.ini num endereço de localizador de serviços uniformes. Este ficheiro contém a lista de servidores de JDBC.

JDBC SERVER FILE=(nomeficheiro)

Especifica o ficheiro cmbjdbcsvrs.ini que contém a lista de servidores de JDBC como um nomeficheiro.

JDBC DRIVER=(controlador de JDBC)

Especifica o controlador de JDBC que pretende utilizar. Esta acção é definida automaticamente quando utilizar o programa cliente de cliente de administração de sistemas.

DSNAME=(nome do servidor de conteúdos)

Especifica o nome do servidor de conteúdos. Para consultas associadas e outras funções associadas, o Enterprise Information Portal define esta acção automaticamente.

AUTO COMMIT=ON | OFF

Liga ou desliga a consolidação automática. A predefinição é desligada. Quando este servidor de conteúdos é utilizado para consultas associadas e outras funções associadas, a autoconsolidação é activada por predefinição.

Enumerar entidade e atributos de entidades

Após criar o servidor de conteúdos para a base de dados relacional e de estabelecer uma ligação ao mesmo, pode enumerar a entidade e os atributos da entidade. O exemplo seguinte apresenta os passos para obter a lista:

Java

```
// ----- After creating a datastore and connecting, get index classes
pCol = (DKSequentialCollection)dsDB2.listDataSources();
pIter = pCol.createIterator();
while (pIter.more() == true)
{
    i++;
    pSV = (DKServerDefDB2)pIter.next();
    strServerName = pSV.getName();
    .... // Use the server name as appropriate
}
// ----- Connect to datastore
dsDB2.connect(db, userid, pw, "");
if (!schema.equals(""))
{
    dsDefDB2 = (DKDatastoreDefDB2)dsDB2.datastoreDef();
    dsDefDB2.setSchemaName(schema);
    schema = dsDefDB2.getSchemaName();
    System.out.println(" New Schema Name = [" + schema + "]");
}
// ----- List the tables
pCol = (DKSequentialCollection) dsDB2.listEntities();
pIter = pCol.createIterator();
i = 0;
while (pIter.more() == true)
{
    i++;
    tableDef = (DKTableDefDB2)pIter.next();
    strTable = tableDef.getName();
    // ----- List attributes (columns for the table)
    pCol2 = (DKSequentialCollection) dsDB2.listEntityAttrs(strTable);
    pIter2 = pCol2.createIterator();
    j = 0;
    while (pIter2.more() == true)
    {
        j++;
        colDef = (DKColumnDefDB2)pIter2.next();
        .... // Process the information as appropriate
    }
}
// ----- Commit and disconnect
dsDB2.commit();
dsDB2.disconnect();
dsDB2.destroy();
```

Consulte TListCatalogDB2.java, TListCatalogJDBC.java e TListCatalogDJ.java no directórioCMBROOT\Samples\java\d1 para obter exemplos completos.

C++

```
try {
    DKDatastoreDB2 dsDB2;
    DKString schema;
    DKSequentialCollection *pCol2 = 0;
    dkIterator *pIter = 0;
    dkIterator *pIter2 = 0;
    DKTableDefDB2 *pEnt = 0;
    DKString strServerName;
    DKString strTable;
    DKColumnDefDB2 *pAttr = 0;
    DKDatastoreDefDB2 *dsDefDB2 = 0;
    DKAny a;
    DKAny *pA = 0;
    long i = 0;
    long j = 0;

    // ----- Connect to datastore and set value for schema name
    . . .
    // ----- Create a datastore definition and set the schema name
    dsDefDB2 = (DKDatastoreDefDB2 *) dsDB2.datastoreDef();
    if (schema != "")
    {
        dsDefDB2->setSchemaName(schema);
    }

    // ----- Get a list of the entities (tables)
    pCol = (DKSequentialCollection*)((dkCollection*)dsDB2.listEntities());
    pIter = pCol->createIterator();
    i = 0;
    // ----- List the attributes (columns) for each entity (table)
    while (pIter->more() == TRUE)
    {
        i++;
        pEnt = (DKTableDefDB2*)((void*)(*pIter->next()));
        strTable = pEnt->getName();
        cout << "table name [" << i << "] - " << strTable << endl;
        cout << " list columns for " << strTable << " table" << endl;
        pCol2 =
        (DKSequentialCollection*)((dkCollection*)dsDB2.listEntityAttrs(strTable));
        pIter2 = pCol2->createIterator();
        j = 0;
        while (pIter2->more() == TRUE)
        {
            j++;
            pA = pIter2->next();
            pAttr = (DKColumnDefDB2*) pA->value();
            cout << "    Attr name [" << j << "] - " << pAttr->getName() << endl;
            cout << "        datastoreName " << pAttr->datastoreName() << endl;
            cout << "        datastoreType " << pAttr->datastoreType() << endl;
            cout << "        attributeOf " << pAttr->getEntityName() << endl;
            cout << "        type " << pAttr->getType() << endl;
            cout << "        size " << pAttr->getSize() << endl;
            cout << "        id " << pAttr->getId() << endl;
            cout << "        nullable " << pAttr->isNullable() << endl;
            cout << "        precision " << pAttr->getPrecision() << endl;
            cout << "        scale " << pAttr->getScale() << endl;
            cout << "        string type " << pAttr->getStringType() << endl;
            cout << "        primary key " << pAttr->isPrimaryKey() << endl;
            cout << "        foreign key " << pAttr->isForeignKey() << endl;
            delete pAttr;
        }
    }
    // continued...
```

C++ (continuação)

```
        delete pIter2;
        delete pCol2;
        delete pEnt;
    }
    delete pIter;
    delete pCol;
    dsDB2.disconnect();
}
catch(DKException &exc)
{
    . . .
}
```

Consulte TListCatalogDB2.cpp, TListCatalogODBC.cpp e TListCatalogDJ.cpp nos directórios CMBROOT\Samples\cpp\db2, odbc e dj para obter exemplos completos.

Executar uma consulta

Para executar uma consulta, primeiro deve criar a cadeia de consulta e, de seguida, executar a consulta. O exemplo seguinte executa uma consulta e processa os resultados.

Java

```
// ----- After creating a datastore and connecting, build the
//          query and parameters and execute it
sDB2 = new DKDatastoreDB2();
dkResultSetCursor pCur = null;
DKNameValuePair parms[] = new DKNameValuePair[2];
String strMax = "5";
parms[0] = new DKNameValuePair(DK_CM_PARM_MAX_RESULTS, strMax);
parms[1] = new DKNameValuePair(DK_CM_PARM_END, null);
// ----- Connect to datastore
dsDB2.connect(database, userid, pw, "");
// --- Create the query string
String cmd = "";
cmd = "SELECT * FROM EMPLOYEE";

DKDDO p = null;
DKDDO pDDO = null;
dkXDO pXDO = null;
DKPidXDO pidXDO = null;
int i = 0;
int numDataItems = 0;
short k = 0;
short j = 0;
String strDataName;
dkCollection pCol = null;
dkIterator pIter = null;
Object a = null;
dkDataObjectBase pDO = null;
int cnt = 0;

// continued...
```

Java (continuação)

```
// ----- Execute the query
pCur = dsDB2.execute(cmd,DK_CM_SQL_QL_TYPE,parms);
if (pCur == null)
{
    // Handle if the cursor is null
}
while (pCur.isValid())
{
    p = pCur.fetchNext();
    if (p != null)
    {
        cnt++;
        i = pCur.getPosition();
        // Get item information
        numDataItems = p.dataCount();
        DKPid pid = p.getPidObject();
        System.out.println("pid string " + pid.pidString());
        System.out.println("Number of Data Items " + numDataItems);
        for (j = 1; j <= numDataItems; j++)
        {
            a = p.getData(j);
            strDataName = p.getDataName(j);
            // Handle the attributes ;
            if (a instanceof String)
            {
                System.out.println("    Attribute Value " + a);
            }
            ..... // Handle for various types )
            else if (a instanceof dkDataObjectBase)
            {
                pDO = (dkDataObjectBase)a;
                if (pDO.protocol() == DK_PDDO)
                {
                    System.out.println("    DKDDO object ");
                    pid = ((DKDDO)pDO).getPidObject();
                }
                else if (pDO.protocol() == DK_XDO)
                {
                    // dkXDO object
                    pXDO = (dkXDO)pDO;
                    pidXDO = pXDO.getPidObject();
                }
            }
            ..... // Handle for various types
        }
    }
}
// Delete the cursor when you're done, commit and disconnect
pCur.destroy(); // Finished with the cursor
dsDB2.commit();
dsDB2.disconnect();
dsDB2.destroy();
```

Consulte TExecutedB2.java, TExecuteJDBC.java, e TExecuteDJ.java no directório CMBROOT\Samples\java\d1 para obter exemplos completos.

C++

```
try {
    DKDatastoreDB2 dsDB2;
dkResultSetCursor* pCur = 0;
    DKNVPair par[2];
    DKAny anyValue;
    DKString strMax = "5";
    anyValue = strMax;
    par[0].set(DK_CM_PARM_MAX_RESULTS, anyValue);
    par[1].setName(DK_CM_PARM_END);
    // ---- Create a datastore and connect
    . . .
    // ---- Create a query string containing the select
    DKString qstrng = "SELECT * FROM EMPLOYEE";
    // ---- Execute the query
    pCur = dsDB2.execute(qstrng, DK_CM_SQL_QL_TYPE, par);
    // ---- Declarations
    DKDDO *p = 0;
    dkDataObjectBase *pDOBase = 0;
    DKDDO *pDDO = 0;
    dkXDO *pXDO = 0;
    DKAny a;
    ushort j = 0;
    ushort k = 0;
    ushort val = 0;
    ushort cnt = 1;
    DKString strData = "";
    DKString strDataName = "";
    dkCollection* pdCol = 0;
    dkIterator* pdIter = 0;
    ushort numDataItems = 0;
    DKString strPid;
    DKPid* pid = 0;
    short sVal = 0;
    long lVal = 0;
    while (pCur->isValid())
    {
        p = pCur->fetchNext();
        if (p != 0)
        {
            cout << "======" << "Item " << cnt << " <======" << endl;
            numDataItems = p->dataCount();
            pid = (DKPid*)p->getPidObject();
            strPid = pid->pidString();
            cout << "pid string " << strPid << endl;
            k = p->propertyId(DK_CM_PROPERTY_ITEM_TYPE);
            if (k > 0)
            {
                a = p->getProperty(k);
                val = a;
                switch (val)
                {
                    case DK_CM_DOCUMENT :
                    {
                        cout << "Item is document " << endl;
                        break;
                    }
                }
            }
        }
    }
    cout << "Number of Data Items " << numDataItems << endl;
```

C++ (continued)

```
for (j = 1; j <= numDataItems; j++)
{
    a = p->getData(j);
    strDataName = p->getDataName(j);
    switch (a.typeCode())
    {
        case DKAny::tc_string :
        {
            strData = a;
            cout << "attribute name : " << strDataName << " value : "
<< strData << endl;
            break;
        }
        // ---- Handle each type in a similar fashion
        . . .
    }
}
// ----- Delete the cursor and disconnect
if (pCur != 0)
    delete pCur;
dsDB2.disconnect();
}
catch(DKException &exc)
. . .
```

Consulte TExecuteDB2.cpp, TExecuteODBC.cpp, e TExecuteDJ.cpp nos directórios CMBROOT\Samples\cpp para obter exemplos completos.

Criar conectores de servidores de conteúdos personalizados

O utilizador pode criar as suas próprias definições de servidor para servidores de conteúdos personalizados (que actualmente não estão incluídos no Enterprise Information Portal). Se integrar um servidor personalizado em EIP, deve facultar as suas próprias classes de Java ou C++ para suportar a definição.

Desenvolver conectores de servidores de conteúdos personalizados

A estrutura da API orientada para objectos está concebida com os seguintes objectivos:

- Podem ser adicionados sistemas de armazenamento de dados ou servidores de conteúdo adicionais à estrutura.
- Capacidade de definir correspondências a qualquer tipo complexo de dados de servidor de conteúdos.
- Um modelo de objectos comum para todo o acesso a dados de servidor de conteúdos.
- Um mecanismo flexível para utilizar uma combinação de diferentes tipos de motores de busca como, por exemplo, Motor de Pesquisa de Texto, pesquisa de imagens(QBIC), etc.
- Implementação cliente/servidor para utilizadores de aplicações de Java.

Para obter informações sobre APIs específicas orientadas para objectos, consulte o referência de API online.

Se estiver a integrar um servidor de conteúdos personalizado num Enterprise Information Portal terá de:

- Importe o pacote com.ibm.mm.sdk.common.
- **Apenas para Java:** Estabeleça ligação ao ficheiro (Java) cmbcm81.jar de modo a aceder à estrutura comum.
- **Apenas para C++:** Estabeleça ligação aos ficheiros cmbcm817.dll, versão não depurada, e cmbcm8167.dll, versão depurada, de modo a aceder à estrutura comum.

Infra-estrutura da base de dados do Enterprise Information Portal

As classes de dkDatastore funcionam como a interface primária entre o Enterprise Information Portal e os servidores de conteúdos. Cada servidor de conteúdos possui uma classe diferente que implementa a classe de dkDatastore para facultar informações de implementação a um servidor de conteúdos específico. Cada tipo de servidor de conteúdos é representado por uma classe denominada DKDatastore *xx*, onde *xx* identifica o nome ou tipo do servidor de conteúdos especificado. A Tabela 7 na página 36 enumera os servidores de conteúdo actuais facultados no Enterprise Information Portal.

O utilizador deve especificar a classe de DKDatastore que criar para o servidor de conteúdos no cliente de administração do sistema de EIP, quando cria a definição do servidor.

Classes comuns no Enterprise Information Portal

dkDDO

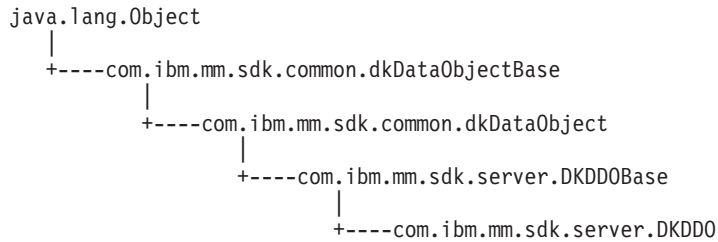
A classe de dkDDO fornece uma representação e um protocolo para definir e aceder aos dados de um objecto, independentemente do tipo do objecto. O protocolo de DDO está implementado como um conjunto de funções para definir, adicionar e aceder a cada artigo de dados de um objecto. Poderá utilizar este protocolo para criar de forma dinâmica um objecto e obtê-lo do servidor de conteúdos, sem ter em conta o tipo do servidor de conteúdos.

Ao implementar um servidor de conteúdos, poderá utilizar informações de definição de correspondências de esquemas, registadas na classe do servidor de conteúdos. O esquema define correspondências de cada artigo de dados permanentes individualmente para a representação subjacente no servidor de conteúdos.

Um DDO tem um conjunto de atributos; cada atributo tem um tipo, valor e propriedades associadas a ele. O próprio DDO pode ter propriedades que pertencem ao DDO como um todo. Por exemplo, pode definir correspondências entre uma classe e um artigo no servid or de conteúdos do Content Manager ou a um documento em OnDemand.

Java

Este diagrama representa a hierarquia da classe de dkDDO:



dkXDO

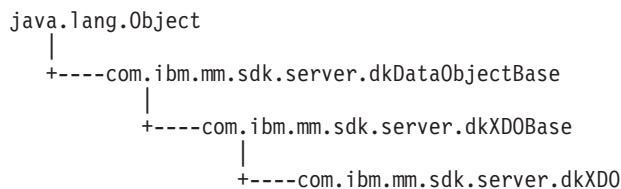
A classe de dkXDO representa tipos complexos de grandes objectos(LOBs) definidos pelo utilizador, que podem existir de forma autónoma ou como fazendo parte do DDO. Deste modo, tem um identificador permanente (PID) e cria, obtém, actualiza e elimina funções.

A classe de dkXDO expande a interface pública de dkXDObase através da definição nos servidores de conteúdos de funções independentes de aceder, criar, obter, actualizar e eliminar. Estas funções permitem a uma aplicação armazenar e obter os dados do objecto de um servidor de conteúdos sem um objecto de classe de DDO ou um XDO autónomo.

O utilizador tem de definir o PID para um XDO autónomo para localizar a sua posição no servidor de conteúdos. Caso esteja a utilizar o XDO com um DDO, o PID é definido automaticamente. Por exemplo, pode definir correspondências entre uma classe e um artigo no servidor de conteúdos do Content Manager ou entre um documento e notas de servidores de conteúdos do OnDemand.

Java

Segue-se a hierarquia de classes de uma classe de dkXDO:



dkCollection

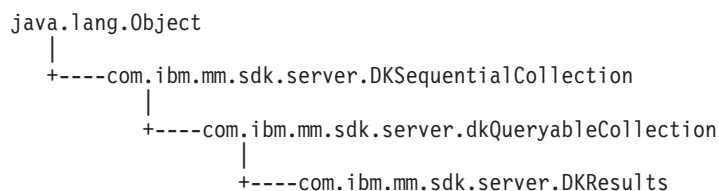
A classe dkCollection é uma recolha de objectos. dkCollection não pode avaliar uma consulta. Uma recolha pode ter um nome (o nome predefinido é uma cadeia vazia). Por exemplo, DKParts é uma subclasse de DKSequentialCollection, que por sua vez é uma subclasse de dkCollection.

DKResults

DKResults é uma subclasse de dkQueryableCollection, logo suporta os iteradores de ordenação e bidireccionais, sendo também passível de consulta. Os membros do elemento de uma classe de DKResults são objectos, ocorrências da classe de dkDDO que representam os resultados da consulta. O iterador criado por esta classe é dkSequentialIterator.

Java

Segue-se a hierarquia de classes da classe de DKResults:



dkQuery

dkQuery é uma interface de um objecto de consulta associada a um servidor de conteúdos específico. Os objectos que implementam esta interface são criados pelas classes de servidores de conteúdos. O resultado de uma consulta é normalmente um objecto de DKResults. Os exemplos de uma implementação concreta da interface dkQuery são DKParametricQuery, DKTextQuery e DKImageQuery, que são criados pelos seus servidores de conteúdos associados.

dkCQExpr

A classe de dkCQExpr representa uma expressão de consulta composta ou combinada. Poderá conter uma árvore de expressões de dkQExpr, que poderá conter uma combinação de consultas paramétricas, de texto e imagem. Se pretende que cada servidor de conteúdos permita uma pesquisa associada, o servidor de conteúdos terá de poder processar este objecto de dkCQExpr.

dkSchemaMapping

dkSchemaMapping é uma interface que define uma definição de correspondências associativa entre uma entidade associada e uma entidade nativa no servidor de conteúdos. O servidor de conteúdos tem de entender esta classe de definição de correspondências para anular e voltar a activar a definição de correspondências nas entidades e atributos associados a entidades e atributos nativos, numa consulta e devolver resultados.

dkDatastore e classes relacionadas

Terá de implementar uma classe concreta para cada uma das classes ou interfaces seguintes para o servidor de conteúdos. Por exemplo, num servidor OnDemand, a classe concreta que implementa a interface de dkDatastore é DKDatastoreOD.

dkDatastore

dkDatastore representa e gere uma ligação ao servidor de conteúdos, suas transacções e comandos. Este suporta a função avaliar, de forma a que possa ser considerado uma subclasse do gestor da consulta.

Os métodos principais na interface de dkDatastore são:

ligar() Liga ao servidor de conteúdos.

desligar()

Anula a ligação ao servidor de conteúdos.

avaliar(), executar(), executarComChamada()

Consulta o servidor de conteúdos.

commit(), rollback()

Executa transacções no servidor de conteúdos.

Restrição: Alguns servidores de conteúdos não suportam estas funções.

registerServices(), unregisterServices()

regista motores de pesquisa.

alterarPalavrapasse (idutilizador, antigaPalavrapasse, novaPalavrapasse)

Altera a palavra-passe de início de sessão para o actual ID de utilizador de início de sessão do servidor de conteúdos.

listDataSources()

Devolve uma recolha de objectos de ID de utilizador do servidor de conteúdos para utilizar no início de sessão. O utilizador não tem de estar ligado ao servidor de conteúdos para utilizar esta função.

listDataSourceNames()

Devolve uma matriz de nomes do servidor de conteúdos.

getExtension(String)

Obtém o objecto de dkExtension do servidor de conteúdos. Se a extensão apresentada ainda não existir mas for suportada pelo servidor de conteúdos, é devolvido um objecto recém-criado, caso contrário é devolvido um valor nulo.

addExtension(String, dkExtension)

Adiciona um novo objecto de expansão (XDO) a este servidor de conteúdos.

criarDDO(Cadeia,int)

Cria um objecto de dados baseado no tipo e no sinalizador do objecto apresentado. Criar DDO devolve um novo objecto de DKDDO com todas as propriedades e atributos definidos. O programa de chamada tem de fornecer os valores do atributo para este tipo de dados.

Os métodos de manipulação do objecto de dados na interface de dkDatastore são:

addObject(dkDataObject)

Adiciona um novo documento ou pasta ao servidor de conteúdos.

retrieveObject(dkDataObject)

Obtém um documento ou pasta no servidor de conteúdos.

deleteObject(dkDataObject)

Elimina um documento ou uma pasta do servidor de conteúdos.

updateObject(dkDataObject)

Actualiza um documento ou pasta no servidor de conteúdos.

moveObject(dkDataObject, String)

Move uma pasta ou documento de uma entidade para outra.

Os métodos relacionados com a definição de correspondências do esquema na interface de dkDatastore são:

registerMapping(DKNVPair)

Regista as informações de definição de correspondências neste servidor de conteúdos.

unRegisterMapping(String)

Remove as informações de definição de correspondências deste servidor de conteúdos.

listMappingNames()

Devolve uma tabela de nomes de definição de correspondências deste servidor de conteúdos.

getMapping(String)

Devolve um objecto dkSchemaMapping.

dkDatastoreDef

A interface de dkDatastoreDef define funções para aceder às informações do servidor de conteúdos e para criar, enumerar e eliminar as suas entidades. Mantém uma recolha de objectos de dkEntityDef.

A Tabela 22 contém exemplos de classes concretas para a interface de dkDatastoreDef.

Tabela 22. Classes concretas para dkDatastoreDef

| Tipo de servidor | Nome da classe |
|----------------------------|-------------------|
| Content Manager | DKDatastoreDefICM |
| OnDemand | DKDatastoreDefOD |
| Content Manager for AS/400 | DKDatastoreDefV4 |
| ImagePlus para OS/390 | DKDatastoreDefIP |
| Domino.Doc | DKDatastoreDefDD |
| Extended Search | DKDatastoreDefDES |
| IBM DB2 Universal Database | DKTableDefDB2 |
| JDBC | DKTableDefJDBC |
| ODBC | DKTableDefODBC |
| Content Manager anterior | DKDatastoreDefDL |

Os métodos principais na interface de dkDatastoreDef são:

listEntities()

Enumera entidades.

listEntityAttrs()

Enumera atributos de entidades.

addEntity()

Adiciona uma entidade.

getEntity(name)

Obtém uma entidade.

Cada classe concreta também pode ter as suas funções específicas de servidor de conteúdos com nomes que sejam conhecidos por esse servidor de conteúdos. Por exemplo, a classe de DKDatastoreDefDL contém as seguintes funções específicas:

- listIndexClassNames()
- listIndexClasses()

A classe de DKDatastoreDefOD contém estas funções específicas:

- listAppGrps()

- `listAppGrpNames()`

dkEntityDef

A classe de `dkEntityDef` define funções para:

- Aceder a informações da entidade.
- Criar e eliminar atributos.
- Criar e eliminar a entidade.

Na classe de `dkEntityDef`, todas as funções que estejam relacionadas a sub-entidades geram um `DKUsageError` a informar que o servidor de conteúdos predefinido não suporta sub-entidades. Contudo, caso o servidor de conteúdos não suporte este tipo de entidade de nível múltiplo, como suporta por exemplo o `Domino.Doc`, a subclasse para este servidor de conteúdos terá de implementar as funções adequadas para substituir as excepções.

A Tabela 23 contém exemplos de classes concretas para a classe de `dkEntityDef`.

Tabela 23. Classes concretas para dkEntityDef

| Tipo de servidor | Nome da classe |
|----------------------------|--------------------------------|
| Content Manager | <code>DKItemTypeDefDL</code> |
| OnDemand | <code>DKAppGrpDefOD</code> |
| Content Manager for AS/400 | <code>DKIndexClassDefV4</code> |
| ImagePlus para OS/390 | <code>DKEntityDefIP</code> |
| Domino.Doc | <code>DKCabinetDefDD</code> |
| Extended Search | <code>DKDatabaseDefDES</code> |
| DB2 Universal Database | <code>DKTableDefDB2</code> |
| JDBC | <code>DKTableDefJDBC</code> |
| ODBC | <code>DKTableDefODBC</code> |
| Content Manager anterior | <code>DKIndexClassDefDL</code> |

As funções principais na classe de `dkEntityDef` são:

listAttrs()

Lista os atributos da entidade.

getAttr(String attrName)

Obtém um atributo especificado da entidade.

addAttr(DKAttrDef)

Adiciona um atributo a uma entidade.

getName()

Obtém o nome da entidade.

setName(String)

Define o nome da entidade.

hasSubEntities()

Determina se a entidade contém subentidades.

getSubEntity(String)

Obtém a sub-entidade.

addSubEntity(*dkEntityDef*)

Adiciona uma sub-entidade à entidade.

listSubEntities()

Enumera as sub-entidades da entidade.

removeAttr(*String*)

Remove a sub-entidade da entidade.

add() Adiciona a entidade ao servidor de conteúdos.

update()

Actualiza a entidade no servidor de conteúdos.

retrieve()

Obtém os valores da entidade a partir do servidor de conteúdos.

del() Elimina a entidade do servidor de conteúdos.

dkAttrDef

A classe de dkAttrDef define funções para aceder a informações de atributos e criar e eliminar atributos. A Tabela 24 contém exemplos de classes concretas para a classe de dkAttrDef.

Tabela 24. Classes concretas para dkAttrDef

| Tipo de servidor | Nome da classe |
|----------------------------|-----------------|
| Content Manager | DKAttrDefDICM |
| OnDemand | DKFieldDefOD |
| Content Manager for AS/400 | DKAttrDefV4 |
| ImagePlus para OS/390 | DKAttrDefIP |
| Domino.Doc | DKAttrDefDD |
| Extended Search | DKFieldDefDES |
| DB2 Universal Database | DKColumnDefDB2 |
| JDBC | DKColumnDefJDBC |
| ODBC | DKColumnDefODBC |
| Content Manager anterior | DKAttrDefDL |

Os métodos principais na classe de dkAttrDef são:

listAttrs()

Enumera os atributos.

getAttr(*String attrName*)

Obtém um atributo especificado.

getName()

Obtém o nome do atributo.

getDescription()

Obtém a descrição do atributo.

add() Adiciona a entidade ao servidor de conteúdos.

dkServerDef

A classe de dkServerDef fornece as informações de definição do servidor para cada servidor de conteúdos. A Tabela 25 na página 386 contém exemplos de classes concretas para a classe de dkServerDef.

Tabela 25. Classes concretas para *dkServerDef*

| Tipo de servidor | Nome da classe |
|----------------------------|-----------------|
| Content Manager | DKServerDefICM |
| OnDemand | DKServerDefOD |
| Content Manager for AS/400 | DKServerDefV4 |
| Domino.Doc | DKServerDefDD |
| Extended Search | DKServerDefDES |
| DB2 Universal Database | DKServerDefDB2 |
| JDBC | DKServerDefJDBC |
| ODBC | DKServerDefODBC |
| Content Manager anterior | DKServerDefDL |

As funções principais na classe de *dkServerDef* são:

setDatastore(dkDatastore ds)

Define a referência ao objecto do servidor de conteúdos.

getDatastore()

Obtém a referência ao objecto do servidor de conteúdos.

getName()

Obtém o nome do servidor de conteúdos.

setName(String name)

Define o nome do servidor de conteúdos.

datastoreType()

Obtém o tipo de servidor de conteúdos.

dkResultSetCursor

dkResultSetCursor é um cursor do servidor de conteúdos no conjunto de resultados da consulta, de forma a que o utilizador o possa utilizar na gestão de uma recolha virtual de objectos de DDO. A recolha é um conjunto de resultados da consulta. Cada elemento da recolha só é criado quando o servidor de conteúdos obtém o elemento.

As funções principais na classe de *dkResultSetCursor* são:

isScrollable()

Devolve TRUE caso o cursor possa ser utilizado para avançar e recuar.

isUpdatable()

Devolve TRUE caso o cursor possa ser actualizado.

isValid()

Devolve TRUE caso o cursor seja válido.

isBegin()

Devolve TRUE caso o cursor esteja posicionado no início do conjunto de resultados.

isEnd()

Devolve TRUE caso o cursor esteja posicionado no final do conjunto de resultados.

isInBetween()

Devolve TRUE caso o cursor esteja posicionado entre elementos de dados no conjunto de resultados.

getPosition()

Obtém a posição actual do cursor.

definirPosição(posição int, Valor objecto)

Define a posição especificada do cursor.

setToNext()

Define o cursor para indicar o elemento seguinte no conjunto de resultados.

fetchObject()

Obtém o elemento actual do conjunto de resultados e devolve-o como um DDO.

fetchNext()

Obtém o elemento seguinte do conjunto de resultados e devolve-o como um DDO.

findObject(int position, String predicate)

Localiza o objecto de dados que preenche o predicado especificado, move o cursor para essa posição e, de seguida, obtém o objecto.

addObject(DKDDO ddo)

Adiciona um novo elemento do mesmo tipo, representado pelo DDO específico, ao servidor de conteúdos.

deleteObject()

Elimina o elemento actual do servidor de conteúdos.

updateObject(DKDDO ddo)

Actualiza o elemento actual na posição actual no servidor de conteúdos, utilizando o DDO especificado.

newObject()

Cria um elemento do mesmo tipo e devolve-o como um DDO.

open() Abre o cursor e, caso seja necessário, executa a consulta para criar o conjunto de resultados.

close() Fecha o cursor e o conjunto de resultados.

isOpen()

Devolve TRUE caso o cursor esteja aberto.

destroy()

Elimina o cursor; esta acção permite uma limpeza antes de o cursor ser recolhido como lixo.

datastoreName()

Obtém o nome do servidor de conteúdos ao qual o cursor pertence.

datastoreType()

Obtém o tipo de servidor de conteúdos ao qual o cursor pertence.

handle(int type)

Obtém o parâmetro identificador do conjunto de resultados associado ao cursor de conjunto de resultados, por tipo.

Requisito: para utilizar as funções `addObject` , `deleteObject` e `updateObject`, deve definir a opção de servidor de conteúdos `DK_DL_OPT_ACCESS_MODE` como `DK_READWRITE`.

dkBlob

`dkBlob` é uma classe abstracta que declara uma interface pública comum para tipos de dados de objectos binários de grandes dimensões (BLOB) básicos. As classes concretas derivadas da classe de `dkBlob` partilham esta interface pública comum, que permite o processamento polimórfico de recolhas de BLOBs, provenientes de servidores de conteúdo heterogéneos. Também existe uma classe de `dkClob` e de `dkDBClob` que podem ter classes concretas.

A Tabela 26 contém exemplos de classes concretas para a classe de `dkBlob`.

Tabela 26. Classes concretas para dkBlob

| Tipo de servidor | Nome da classe |
|----------------------------|------------------------|
| Content Manager | DKLobICM |
| OnDemand | DKBlobOD |
| Content Manager for AS/400 | DKBlobV4 |
| ImagePlus para OS/390 | DKBlobIP |
| Domino.Doc | DKBlobDD |
| Extended Search | DKBlobDES |
| DB2 Universal Database | DBBlobDB2, DKBlobDB2 |
| JDBC | DKBlobJDBC, DKBlobJDBC |
| ODBC | DKBlobODBC, DKBlobODBC |
| Content Manager anterior | DKBlobDL |

Os métodos principais na classe de `dkBlob` são:

getContent()

Devolve uma matriz de bytes contendo dados de BLOB do objecto.

getContentToClientFile(String afileName, int fileOption)

Copia os dados de BLOB do objecto para o ficheiro especificado.

setContent(byte[] aByteArr)

Define os dados de LOB do objecto com os conteúdos da matriz de bytes.

setContentFromClientFile(String afileName)

Substitui os dados de LOB do objecto pelos conteúdos do ficheiro *afileName*.

add(String afileName)

Adiciona o conteúdo do ficheiro especificado ao servidor de conteúdos.

retrieve(String afileName)

Obtém o conteúdo do servidor de conteúdos ao ficheiro especificado.

update(String afileName)

Actualiza o objecto e o servidor de conteúdos com o conteúdo do ficheiro especificado

del(boolean flush)

Elimina os dados do objecto do servidor de conteúdos, se *limpar* for TRUE; caso contrário mantém-se o conteúdo actual.

concatReplace(dkBlob aBlob), concatReplace(byte[] aByteArr)

Concatena este objecto com outro objecto de dkBlob ou de matriz de bytes.

length()

Devolve o comprimento do conteúdo de LOB do objecto.

indexOf(String aString, int startPos), indexOf(dkBlob aBlob, int startPos)

Iniciando a pesquisa em posições iniciais, devolve a posição de byte da primeira ocorrência do argumento de pesquisa neste objecto.

substring(int startPos, int length)

Devolve um objecto de cadeia que contém a sub-cadeia dos dados de LOB deste objecto.

remove(int startPos, int aLength)

Ao iniciar nos bytes *startPos* para *aLength*, irá eliminar uma porção de dados de LOB deste objecto.

insert(String aString, int startPos), insert(dkBlob aBlob, int startPos)

Insere os dados do argumento, seguindo a posição *startPos* nos dados de LOB do objecto

open(String afileName)

Descarrega os conteúdos do objecto no ficheiro *afileName* e, de seguida, devolve o parâmetro identificador de um ficheiro predefinido.

setClassOpenHandler(String aHandler, boolean newSynchronousFlag)

Identifica, através do nome do programa executável, a rotina de tratamento de ficheiros para uma classe inteira. Esta função também indica se deve executar a rotina de tratamento de forma síncrona ou assíncrona para o objecto do ficheiro.

setInstanceOpenHandler(String aHandler, boolean newSynchronousFlag)

Identifica, pelo nome do programa executável, o parâmetro identificador do ficheiro e indica se deve ser executado síncrona ou assincronamente no caso deste objecto.

getOpenHandler()

Obtém o nome do programa executável do parâmetro identificador do ficheiro para uma classe inteira.

isOpenSynchronous()

Devolve a actual definição de sincronização do parâmetro identificador.

dkClob

dkClob é uma classe abstracta que declara uma interface pública para armazenar tipos de dados de grandes objectos de caracteres (CLOB), tais como documentos.

A Tabela 27 na página 390 contém exemplos de classes concretas para a classe de dkClob.

Tabela 27. Classes concretas para dkClob

| Tipo de servidor | Nome da classe |
|------------------------|----------------|
| DB2 Universal Database | DKClobDB2 |
| ODBC | DKClobODBC |

As funções principais na classe de dkClob são:

open() Open() é um membro herdado de dkXDOBase. Open() será implementado ou substituído por subclasses concretas de dkClob.

Membros de dkXDO: dkXDO& add(), dkXDO retrieve(), dkXDO update(), dkXDO del()

Herdados como membros seguros de dkXDO. Onde se verifique ser necessário, estes membros seguros serão implementados ou substituídos através de subclasses concretas de dkClob.

A lista seguinte contém membros definidos por dkClob:

add(String afileName)

Adiciona o conteúdo do ficheiro específico ao servidor de conteúdos.

retrieve(String afileName)

Obtém o conteúdo do servidor de conteúdos para o ficheiro especificado.

update(String afileName)

Actualiza o objecto e o servidor de conteúdos com o conteúdo do ficheiro especificado.

del(DKBoolean flush)

Elimina os dados do objecto do servidor de conteúdos, se *limpar* for TRUE; caso contrário mantém-se o conteúdo actual.

getContentToClientFile(String afileName, int fileOption)

Copia os dados de CLOB do objecto para o ficheiro especificado.

setContentFromClientFile(String afileName)

Substitui os dados de LOB do objecto com os conteúdos do ficheiro *afileName*.

indexOf(String& aString, long startPos=1), indexOf(dkClob& adkClob, long startPos=1)

Iniciando a pesquisa em posições iniciais, devolve a posição de byte da primeira ocorrência do argumento de pesquisa neste objecto.

substring(long startPos, long length)

Devolve um objecto de cadeia que contém a sub-cadeia dos dados de LOB deste objecto.

remove(long startPos, long aLength)

Ao iniciar nos bytes *startPos* para *aLength*, irá eliminar uma porção de dados de LOB deste objecto.

insert(DKString aString, long startPos), insert(dkClob& adkClob, long startPos)

Insere os dados do argumento seguindo a posição *startPos* nos dados de CLOB do objecto

open(String afileName)

Descarrega os conteúdos do objecto no ficheiro *afileName* e, de seguida, executa o parâmetro identificador de um ficheiro predefinido.

setInstanceOpenHandler(String ahandler, DKBoolean newSynchronousFlag)

Identifica, pelo nome do programa executável, o parâmetro identificador do ficheiro e indica se deve ser executado síncrona ou assincronamente no caso deste objecto.

setClassOpenHandler(String ahandler, DKBoolean newSynchronousFlag)

Identifica, através do nome do programa executável, a rotina de tratamento de ficheiros para uma classe inteira. Esta função também indica se deve executar a rotina de tratamento de forma síncrona ou assíncrona para o objecto do ficheiro.

getOpenHandler()

Obtém o nome do programa executável do parâmetro identificador do ficheiro para uma classe inteira.

isOpenSynchronous()

Devolve a actual definição de sincronização do parâmetro identificador.

dkAnnotationExt

dkAnnotationExt é a classe de interface para todos os objectos de anotações. Caso o seu servidor de conteúdos suporte dados de anotações, o utilizador terá de implementar esta interface. Este objecto de anotações é uma extensão da sua classe de DKBlobxx, onde o objecto de dkBlob é a representação dos dados de anotação binária e a recolha de DKParts.

dkDatastoreExt

A classe de dkDatastoreExt define as classes de extensão do servidor de conteúdos padrão.

A Tabela 28 contém exemplos de classes concretas para a classe de dkDatastoreExt.

Tabela 28. Classes concretas para dkDatastoreExt

| Tipo de servidor | Nome da classe |
|----------------------------|--------------------|
| Content Manager | DKDatastoreExtICM |
| OnDemand | DKDatastoreExtOD |
| Content Manager for AS/400 | DKDatastoreExtV4 |
| ImagePlus para OS/390 | DKDatastoreExtIP |
| Domino.Doc | DKDatastoreExtDD |
| Extended Search | DKDatastoreExtDES |
| DB2 Universal Database | DKDatastoreExtDB2 |
| JDBC | DKDatastoreExtJDBC |
| Content Manager anterior | DKDatastoreExtDL |

As funções principais na classe de dkDatastoreExt são:

getDatastore()

Obtém a referência ao objecto proprietário do servidor de conteúdos.

setDatastore(dkDatastore ds)

Define a referência do objecto proprietário do servidor de conteúdos.

isSupported(String functionName)

Determina se o nome de função especificado é suportado por esta extensão.

listFunctions()

Enumera todos os nomes de funções suportados desta extensão.

addToFolder(dkDataObject folder, dkDataObject member)

Adiciona um membro a esta pasta e ao servidor de conteúdos.

removeFromFolder(dkDataObject folder, dkDataObject member)

Remove um membro desta pasta e do servidor de conteúdos.

checkout(artigo dkDataObject)

Retira um artigo de documento ou pasta do servidor de conteúdos. Enquanto se retira o artigo, o utilizador tem privilégios exclusivos de actualização do artigo, enquanto que outros apenas têm acesso a leitura.

checkin(artigo dkDataObject)

Dá entrada de um artigo de documento ou pasta que já tinha sido retirado do servidor de conteúdos. Ao dar entrada do ficheiro, o utilizador irá perder todos os privilégios de escrita com esta função.

getCommonPrivilege()

Obtém o privilégio comum de um servidor de conteúdos específico.

isCheckedOut(dkDataObject item)

Determina se foi dada saída de um artigo de documento ou pasta do servidor de conteúdos.

checkedOutUserId(dkDataObject item)

Obtém o ID de utilizador que deu saída do artigo do servidor de conteúdos.

unlockCheckedOut(dkDataObject item)

Desbloqueia o artigo do servidor de conteúdos.

changePassword (String userId, String oldPwd, String newPwd)

Altera a palavra-passe no servidor de conteúdos para o ID de utilizador especificado.

moveObject (dkDataObject dataObj, String entityName)

Move o objecto de *entityName* de uma entidade para outra.

retrieveFormOverlay(String id)

Obtém o objecto de sobreposição de form.

DKPidXDO

A classe de DKPidXDO representa a identificação permanente dos dados de BLOB no servidor de conteúdos.

A Tabela 29 na página 393 contém exemplos de classes concretas para a classe de DKPidXDO.

Tabela 29. Classes concretas para DKPidXDO

| Tipo de servidor | Nome da classe |
|----------------------------|----------------|
| Content Manager anterior | DKPidXDODL |
| OnDemand | DKPidXDOOD |
| Content Manager for AS/400 | DKPidXDOV4 |
| ImagePlus para OS/390 | DKPidXDOIP |
| Domino.Doc | DKPidXDODD |
| Extended Search | DKPidXDODES |
| DB2 Universal Database | DKPidXDODB2 |
| JDBC | DKPidXDOJDBC |
| ODBC | DKPidXDOODBC |

dkUserManagement

A classe de dkUserManagement representa e processa todas as funções de gestão do utilizador do servidor de conteúdos.

A Tabela 30 contém exemplos de classes concretas para a classe de dkUserManagement.

Tabela 30. Classes concretas para dkUserManagement

| Tipo de servidor | Nome da classe |
|----------------------------|----------------|
| Content Manager | DKUserMgmtICM |
| Content Manager for AS/400 | DKUserMgmtV4 |
| ImagePlus para OS/390 | DKUserMgmtIP |
| Content Manager anterior | DKUserMgmtDL |

DKConstant

Todas as constantes comuns estão definidas na classe de DKConstant. Cada servidor de conteúdos tem a sua classe de DKConstantxx para definir constantes específicas para esse servidor de conteúdos.

Recomendação: Todos os servidores de conteúdos utilizam as mensagens comuns sempre que possível.

DKMessageId

Todos os IDs de mensagens comuns estão definidos nesta classe. Cada servidor de conteúdos tem a sua classe de DKMessageIdxx para definir os seus IDs de mensagens.

Recomendação: Todos os servidores de conteúdos devem utilizar as mensagens comuns sempre que possível.

Os ficheiros de propriedades seguintes contêm mensagens comuns de aviso e de erro:

Para Java:

- DKMessage_en.properties
- DKMessage_es.properties

Para C++:

- DKMessage_en_US.properties
- DKMessage_es_ES.properties

Cada servidor de conteúdos tem os seus ficheiros
DKMessagexx_yy_zz.properties para as mensagens de aviso e de erro.

Utilizar a classe FeServerDefBase (apenas Java)

A classe deFeServerDefBase é a classe abstracta que tem de expandir de forma a criar uma definição do servidor personalizado. A classe de Java que expande esta classe de base tem de ter um construtor que aceite os parâmetros seguintes e os transmita à classe superior:

String connectString

A sequência de ligação do servidor.

String[] serverList

A lista de servidores definidos.

String[] associatedServerList

A lista de servidores associados a este servidor (se não houver nenhum, será nula)

String[] serverTypes

A lista de IDs de tipo de servidor definido.

String[] serverTypeDescriptions

A lista de descrições de tipos de servidores definidos.

Quando criar a classe de Java que expande a classe de FeServerDefBase, o utilizador tem de determinar a forma de manusear os dados para o novo diálogo do servidor. Pode utilizar a mesma classe ou uma classe modelo à parte. Se o servidor de conteúdos exigir mais do que campos para a cadeia de ligação, terá de utilizar a base de dados do Enterprise Information Portal e as APIs de Java como modelo para que as funções adicionais sejam devidamente executadas.

Quando os servidores de conteúdos são seleccionados no programa de Administração do Enterprise Information Portal, o menu Novo irá conter a lista de tipos de servidores armazenados na tabela FASERVERTYPES na base de dados do Enterprise Information Portal. Esta tabela contém o nome da classe de Java para ser replicado quando for seleccionado o artigo do menu.

Se o utilizador suportar a verificação da palavra-passe, terá de colocar a sua classe de Java no mesmo directório que o ficheiro do Enterprise Information Portal Administration.jar, poderá replicar de forma dinâmica essa classe de Java e invocar o método verificar com a palavra-passe de entrada do utilizador como parâmetro. O método verificar irá devolver nulo para uma palavra-passe válida ou irá devolver uma matriz de cadeias com as informações de uma palavra-passe não válida.

Construir aplicações de fluxo de trabalho de EIP

Ao utilizar classes de APIs de EIP, pode criar ou expandir as suas próprias aplicações para utilizarem o suporte de fluxo de trabalho de EIP. Normalmente, é efectuada uma pesquisa associada e o fluxo de trabalho é iniciado com o resultado da pesquisa (um artigo de conteúdo ou uma pasta de vários artigos de conteúdo). Irá utilizar as APIs para aceder a uma lista de trabalho e depois apresentar os conteúdos da lista de trabalho. À medida que cada actividade é concluída, o fluxo de trabalho avança para a actividade seguinte do fluxo de trabalho.

Ligação aos serviços do fluxo de trabalho

Para utilizar o fluxo de trabalho de Enterprise Information Portal nas suas aplicações, comece por criar uma instância de DKWorkflowServicesFed. De seguida, estabeleça-lhe uma ligação. O exemplo seguinte inicia os serviços do fluxo de trabalho:

Java

```
// ----- Create the strings for the name of the
//service, user ID and Password
String wfsrv = "icmnlbdb";
String userid = "icmadmin";
String pw = "password";
// ----- Create a federated datastore
DKDatastoreFed dsFed = new DKDatastoreFed();
dsFed.connect(wfsrv, userid, pw,"");
//----- Create the workflow service
DKWorkflowServiceFed svWF =new DKWorkflowServiceFed ();
// ----- Set the datastore in the workflow service
svWF.setDatastore(dsFed);
// ----- Connect to the service
svWF.connect (wfsrv, userid, pw,"");
```

C++

```
// ----- Criar as cadeias para o nome de serviço, PID de utilizador
// ----- and Password
DKString wfsrv = "icmnlbdb";
DKString userid = "icmadmin";
DKString pw = "password";
// ----- Create a federated datastore
DKDatastoreFed* dsFed = new DKDatastoreFed();
dsFed->connect(wfsrv, userid, pw,"");
//----- Create the workflow service
DKWorkflowServiceFed* svWF =new DKWorkflowServiceFed ();
// ----- Set the datastore in the workflow service
svWF->setDatastore(dsFed);
// ----- Connect to the service
svWF->connect (wfsrv, userid, pw,"");
```

Quando tiver terminado de utilizar o serviço do fluxo de trabalho, terá de desligar através da chamada das funções disconnect() e delete().

Java

```
svWF.disconnect();
dsFed.disconnect();
svWF.destroy();
dsFed.destroy();
```

C++

```
svWF->disconnect();
dsFed->disconnect();
delete svWF;
delete dsFed;
```

Iniciar um fluxo de trabalho

Após criar o fluxo de trabalho, deve iniciá-lo. Para iniciar um fluxo de trabalho, conclua os seguintes passos:

1. Crie um objecto de DKWorkflowFed e defina o nome do fluxo de trabalho
2. Crie uma ocorrência de fluxo de trabalho através de um modelo de fluxo de trabalho válido, que é uma definição do fluxo de trabalho concebida no construtor do fluxo de trabalho do Enterprise Information Portal.
3. Defina o PID e a prioridade no contentor.
4. Inicie o fluxo de trabalho.

O exemplo seguinte utiliza estes passos para iniciar um fluxo de trabalho:

Java

```
// ----- Create the DKWorkflowFed object and set the name
DKWorkflowFed WF = new DKWorkflowFed(svWF);
WF.setName("wf1");
// ----- Create an instance of a workflow with the workflow template name
WF.add("WD1");
// ----- Refresh the workflow object
WF.retrieve();
// ----- Construct the container object for the workflow
DKWorkflowContainerFed con = WF.inContainer();
// ----- Retrieve the container data
con.retrieve();
// ----- Add a PID string referring to an Extended Search document
con.setPersistentID("45 3 DES4ross10 Notes Help18 15 Help|23fa");
con.setPriority(100);
// ----- Update the container
con.update();
// ----- Start the workflow
WF.start(con);
```

C++

```
// - Create the DKWorkflowFed object and set the name
DKWorkflowFed* WF = new DKWorkflowFed(svWF);
WF->setName("wfl");
//Create an instance of a workflow with the workflow template name
WF->add("WD1");
// ----- Refresh the workflow object
WF->retrieve();
// ----- Construct the container object for the workflow
DKWorkflowContainerFed* con = WF.inContainer();
// ----- Retrieve the container data
con->retrieve();
// Add a PID string referring to a content item from Extended Search
con->setPersistentID("45 3 DES4ross10 Notes Help18 15 Help|23fa");
// ----- Assign a priority of 100
con->setPriority(100);
// ----- Update the container
con->update();
// ----- Start the workflow
WF->start(con);
. . .
// When you are done, clean up by deleting the container and workflow
delete con;
delete WF;
```

Terminar um fluxo de trabalho

O utilizador pode terminar um fluxo de trabalho chamando a função `terminate()` ou `del()`, tal como é demonstrado no seguinte exemplo:

Java

```
//-----Retrieve the status of the workflow named WF
WF.retrieve();
int state = WF.state();
//-----Check the status and either terminate or delete
if (state ==DKConstantFed.DK_FED_FMC_PS_RUNNING ||
    state ==DKConstantFed.DK_FED_FMC_PS_SUSPENDED ||
    state ==DKConstantFed.DK_FED_FMC_PS_SUSPENDING)
{
    WF.terminate();
}
if (state == DKConstantFed.DK_FED_FMC_PS_READY ||
    state ==DKConstantFed.DK_FED_FMC_PS_FINISHED||
    state ==DKConstantFed.DK_FED_FMC_PS_TERMINATED)
{
    WF.del();
}
```

C++

```
/--Construct a DKWorkflowFed instance
DKWorkflowFed* WF = new DKWorkflowFed(svWF, "Test");
//-----Retrieve the status of the workflow named WF
WF->retrieve();
int state = WF->state();
/--Check the status and either terminate or delete
if (state == DK_FED_FMC_PS_RUNNING ||
state == DK_FED_FMC_PS_SUSPENDED ||
state == DK_FED_FMC_PS_SUSPENDING)
{
WF->terminate();
}
if (state == DK_FED_FMC_PS_READY ||
state == DK_FED_FMC_PS_FINISHED ||
state == DK_FED_FMC_PS_TERMINATED)
{
WF->del();
}
delete WF;
```

Enumerar todos os fluxos de trabalho

O utilizador pode enumerar todos os fluxos de trabalho num serviço de fluxo de trabalho utilizando a função `listWorkFlows()`. O exemplo seguinte enumera o nome e descrição de todos os fluxos de trabalho de um serviço de fluxo de trabalho, referidos pelo objecto `DKWorkflowServiceFed svWF`.

Java

```
// ----- Call the listWorkFlows method
DKSequentialCollection collWF = (DKSequentialCollection)svWF.listWorkFlows();
DKWorkflowFed WF = null;
if (collWF != null)
{
    dkIterator iterWF = collWF.createIterator();
    while (iterWF.more() == true)
    {
        WF = (DKWorkflowFed)iterWF.next();
        WF.retrieve();
        System.out.println("name = " + WF.getName() + " description = "
                           + WF.getDescription());
    }
    iterWF = null;
}
```

C++

```
// ----- Call the listWorkFlows function
DKSequentialCollection *collWF =
    (DKSequentialCollection*)svWF.listWorkFlows();
DKWorkflowFed *WF = NULL;
if (collWF != NULL)
{
    dkIterator *iterWF = collWF->createIterator();
    while (iterWF->more())
    {
        WF = (DKWorkflowFed*)(void*)(*iterWF->next());
        WF->retrieve();
        cout << "name = " + WF->getName()
        << " description = " << WF->getDescription() << endl;
        delete WF;
    }
    delete iterWF;
}
elimine collWF;
```

Suspender um fluxo de trabalho

Pode suspender um fluxo de trabalho em execução com uma hora específica ou indefinidamente. O exemplo seguinte demonstra como suspender um fluxo de trabalho até determinada hora. Se facultar um DKTimestamp nulo, o EIP suspende indefinidamente o fluxo de trabalho.

Java

```
// ----- Construct a DKWorkflowFed object
DKWorkflowFed WF = new DKWorkflowFed(svWF, "Test");
WF.retrieve();
// ----- Call the suspend method if the workflow is in the running state
if (WF.state() == DKConstantFed.DK_FED_FMC_PS_RUNNING)
{
    // ----- Suspended until 2000-07-27-16.30.00.000000
    // ----- The timestamp uses the base year 1900; months are
    // ----- numbered 0 to 11
    DKTimestamp suspension = new DKTimestamp(100, 6, 27, 16, 30, 0, 0);
    WF.suspend(suspension);
}
```

C++

```
// ----- Construct a DKWorkflowFed instance
DKWorkflowFed* WF = new DKWorkflowFed(svWF, "Test");
WF->retrieve();
// ----- Call the suspend function if the workflow is in
the running state
if (WF->state() == DK_FED_FMC_PS_RUNNING)
{
    // ----- Suspended until 2000-07-27-16.30.00.000000
    DKTimestamp* suspension = new DKTimestamp(2000, 7,
    27, 16, 30, 0, 0);
    WF->suspend(suspension);
    delete suspension;
}
delete WF;
```

Retomar um fluxo de trabalho

O utilizador pode retomar um fluxo de trabalho suspenso chamando a função `resume()`. O exemplo seguinte retoma um fluxo de trabalho suspenso.

Java

```
// ----- Construct a DKWorkflowFed object
DKWorkflowFed WF = new DKWorkflowFed(svWF, "Test");
WF.retrieve();
// ---- Call resume() if the workflow is in the suspended state
if (WF.state() == DKConstantFed.DK_FED_FMC_PS_SUSPENDED)
{
    WF.resume();
}
```

C++

```
// ----- Construct a DKWorkflowFed instance
DKWorkflowFed* WF = new DKWorkflowFed(svWF, "Test");
WF->retrieve();
// ----- Check whether the workflow is suspended and call resume
if (WF->state() == DK_FED_FMC_PS_SUSPENDED)
{
    WF->resume();
}
delete WF;
```

Enumerar todas as listas de trabalhos

O utilizador pode enumerar todas as listas de trabalhos num serviço de fluxo de trabalho utilizando a função `listWorkLists()` no serviço de fluxo de trabalho. O exemplo seguinte enumera o nome e descrição de todas as listas de trabalho num serviço de fluxo de trabalho referido pela instância `DKWorkflowServiceFed svWF`.

Java

```
// ----- Call the listWorkLists method
DKSequentialCollection collWL = (DKSequentialCollection)svWF.listWorkLists();
DKWorkListFed WL = null;
if (collWL != null)
{
    dkIterator iterWL = collWL.createIterator();
    while (iterWL.more() == true)
    {
        WL = (DKWorkListFed)iterWL.next();
        WL.retrieve();
        System.out.println("name = " + WL.getName() + " description = "
            + WL.getDescription());
    }
    iterWL = null;
}
```


C++

```
// ----- Call the listWorkLists function
DKSequentialCollection *collWL =
    (DKSequentialCollection*)svWF.listWorkLists();
DKWorkListFed *WL = NULL;
if (collWL != NULL)
{
    dkIterator *iterWL = collWL->createIterator();
    while (iterWL->more())
    {
        WL = (DKWorkListFed*)(void*)(*iterWL->next());
        WL->retrieve();
        cout << "name = " << WL->getName() << " description = "
            << WL->getDescription() << endl;
        cout << "Threshold = " << WL->getThreshold() << endl;
        delete WL;
    }
    delete iterWL;
}
delete collWL;
```

Aceder a uma lista de trabalhos

Pode aceder a uma lista de trabalhos criando uma instância de DKWorkListFed que faz referência à lista de trabalhos que criou utilizando o cliente de administração de sistemas. O exemplo seguinte acede a uma lista de trabalhos denominada WL0712 e apresenta as informações contidas nessa lista de trabalhos.

Java

```
// ----- Create the DKWorkListFed
DKWorkListFed WL = new DKWorkListFed(svWF, "WL0712");
WL.retrieve();
// ----- Display information about the worklist
System.out.println ("worklist name = " + WL.getName());
System.out.println ("description = " + WL.getDescription() +
    " owner = " + WL.getOwner() +
    " filter = " + WL.getFilter() +
    " threshold = " + WL.getThreshold() +
    " sort criteria = " + WL.getSortCriteria());
```

C++

```
// ----- Create the DKWorkListFed
DKWorkListFed* WL = new DKWorkListFed(svWF, "WL0712");
WL->retrieve();
// ----- Display information about the worklist
cout << "worklist name = " << WL->getName() << endl;
cout << "description = " << WL->getDescription() <<
    " owner = " << WL->getOwner() <<
    " filter = " << WL->getFilter() <<
    " threshold = " << WL->getThreshold() <<
    " sort criteria = " << WL->getSortCriteria() << endl;
// ----- Delete the worklist when you are done
delete WL;
```

Aceder a artigos de trabalho

Após criar o DKWorkListFed, pode obter os artigos de trabalho como uma recolha. O exemplo seguinte obtém os artigos de trabalho.

Java

```
// ----- Create a collection and an iterator
DKSequentialCollection coll = (DKSequentialCollection)WL.listWorkItems();
dkIterator iter = (DKSequentialIterator) coll.createIterator ();
Object a;
DKWorkItemFed item;
String nodename;
String workflowname;

// ----- Step through the collections
while (iter.more ())
{
    a = iter.next ();
    item = (DKWorkItemFed) a;
    if (item != null)
    {
        item.retrieve ();
        nodename = item.name ();
        workflowname = item.workflowName ();
        System.out.println ("workitem node = " + nodename +
                             " workflow name = " + workflowname);
    }
}
iter = null;
```

C++

```
DKSequentialCollection *coll;
dkIterator *iter;
DKWorkItemFed* item;
DKString nodename;
DKString workflowname;
// ----- Create a collection and an iterator
coll = (DKSequentialCollection*)WL->listWorkItems();

if (coll != NULL)
{
    iter = coll->createIterator();
    cout << "listWorkItems" << endl;
    // ----- Step through the collections
    while (iter->more ())
    {
        item = (DKWorkItemFed*)((void*)(*iter->next()));

        if (item != NULL)
        {
            //item.retrieve ();
            nodename = item->name();
            workflowname = item->workFlowName();
            cout << "workitem node = " << nodename
                << " workflow name = " << workflowname << endl;
            delete item;
        }
    }
    delete iter;
    delete coll;
}
```

Mover artigos no fluxo de trabalho

À medida que o fluxo de trabalho avança, o utilizador move artigos de trabalho de uma actividade para a actividade seguinte utilizando as funções `checkOut()` e `checkIn()`. O exemplo seguinte demonstra como mover os artigos de trabalho. Tenha em atenção que apenas o utilizador do fluxo de trabalho actualmente nomeado para executar o artigo de trabalho pode dar entrada e dar saída do artigo de trabalho.

Java

```
DKWorkItemFed item =new DKWorkItemFed(svWF, "wf1", "node1", wfuser);
item.retrieve();
// ----- Call the checkOut method to lock the workitem
item.checkOut();
// ----- Call the checkIn method
item.checkIn(null);
```

C++

```
DKWorkItemFed* item =new DKWorkItemFed(svWF, "wf1", "node1", wfuser);
item->retrieve();
// ----- Call the checkOut method to lock the workitem
item->checkOut();
// ----- Call the checkIn method
item->checkIn(NULL);
delete item;
```

Enumerar todos os modelos de fluxo de trabalho

O utilizador pode enumerar todos os modelos de fluxo de trabalho num serviço de fluxo de trabalho chamando a função `listWorkFlowTemplates()`. O exemplo seguinte enumera o nome e descrição de todos os modelos de fluxo de trabalho de um serviço de fluxo de trabalho, referidos pelo objecto `DKWorkFlowServiceFed svWF`.

Java

```
// ----- Call the listWorkFlowTemplates method
DKSequentialCollection collWT =
    (DKSequentialCollection)svWF.listWorkFlowTemplates();
DKWorkFlowTemplateFed WT = null;
if (collWT != null)
{
    dkIterator iterWT = collWT.createIterator();
    while (iterWT.more() == true)
    {
        WT = (DKWorkFlowTemplateFed)iterWT.next();
        WT.retrieve();
        System.out.println("name = " + WT.name() + " description = "
            + WT.description());
    }
    iterWT = null;
}
```

C++

```
// ----- Call the listWorkFlowTemplates function
DKSequentialCollection *collWT =
    (DKSequentialCollection*)svWF.listWorkFlowTemplates();
DKWorkFlowTemplateFed *WT = NULL;
if (collWT != NULL)
{
    dkIterator* iterWT = collWT->createIterator();
    while (iterWT->more())
    {
        WT = (DKWorkFlowTemplateFed*)(void*)(*iterWT->next());
        WT->retrieve();
        cout << "name = " << WT->name() << " description = "
            << WT->description() << endl;
        delete WT;
    }
    delete iterWT;
}
delete collWT;
```

Criar as suas próprias acções (Apenas para Java)

O utilizador poderá criar as suas próprias acções para utilizar num fluxo de trabalho. O utilizador define as acções e adiciona-as a listas de acções na Administração de Enterprise Information Portal. Pode criar acções utilizando objectos de acção (objectos de `DKWorkflowActionFed`). Um objecto de acção é um contentor de metadados que regista instruções detalhadas relativas à forma como se pretende que uma tarefa específica seja executada no nó cliente. Os objectos de acção (contentores de metadados) apenas registam instruções; não iniciam a invocação das tarefas descritas nos metadados da acção.

As acções podem ser agrupadas numa lista de acções (`DKWorkflowActionListFed`). Um contentor de fluxo de trabalho contém o nome da lista de acções (não contém os conteúdos da lista de acções) na qual um conjunto de acções relacionadas com o artigo de trabalho está associado. Um cliente deve obter a lista de acções e, de seguida, iterar através das entradas (acções) da lista e reagir de acordo com as mesmas. O exemplo seguinte mostra como trabalhar com acções e listas de acções. O exemplo completa as seguintes tarefas:

1. Obtém o artigo de trabalho.
2. Obtém o contentor que é encaminhado juntamente com o artigo de trabalho.
3. Obtém a lista de acções do contentor.
4. Obtém a lista de acções da lista de acções.
5. Inicia as acções de acordo com as mesmas.

```
wit.retrieve(); // wit is a DKWorkItemFed object
DKWorkflowContainerFed wcn = wit.inContainer();
wcn.retrieve();
String alName = wcn.getActionList();
DKWorkflowActionListFed wal = new DKWorkflowActionListFed(dsFed);
wal.setName(alName);
wal.retrieve();
dkIterator iter = null;
if ((coll!=null) && (coll.cardinality()>0))
{
    iter = coll.createIterator();
    while (iter.more ())
    {
        DKWorkflowActionFed act = (DKWorkflowActionFed) iter.next();
        System.out.println("ACTION = " + act.getCommand());
        Runtime.getRuntime().exec(act.getCommand());
    }
}
else
    System.out.println("NO ACTION DEFINED");
```

Construir aplicações com JavaBeans visuais e não visuais

Este capítulo descreve JavaBeans visuais e não visuais fornecidos pelo Enterprise Information Portal.

Os JavaBeans de EIP podem ser divididos nas seguintes categorias:

Bens não visuais O utilizador pode utilizar os beans não visuais para construir aplicações de clientes de Java e da Web que necessitam de uma interface de utilizador personalizada. Os beans não visuais suportam o modelo de programação de bean padrão facultando construtores, propriedades, eventos e uma interface serializável predefinidos. Pode utilizar os beans não visuais em ferramentas de construtor que suportam análise introspectiva.

Beans visuais Os beans visuais são componentes de interface gráfica de utilizador com base em Swing, que podem ser personalizados. Utilize os beans visuais para construir aplicações Java para Windows. Pode colocá-los em janelas e caixas de diálogo de aplicações com base em Java. Devido ao facto dos beans visuais serem construídos recorrendo à utilização de beans não visuais (como, por exemplo, um modelo de dados) o utilizador deve utilizá-los juntamente com os beans não visuais ao construir uma aplicação.

beans relacionados com Java Viewer Os beans do Java Viewer são um conjunto de componentes visuais e não visuais. Pode utilizá-los para construir visualizadores de documentos e para converter documentos. Os beans de Java Viewer são utilizados na applet de visualizador do eClient e no escalão intermédio do eClient.

Compreender conceitos básicos de beans

JavaBeans (que a partir deste momento serão designados como Beans) são componentes de software reutilizáveis que são escritos na linguagem de programação de Java e que podem ser manipulados utilizando ferramentas de construtor com conhecimento de Beans. Devido ao facto dos Beans poderem ser reutilizados, poderá utilizá-los para construir mais componentes complexos, construir novas aplicações ou adicionar funcionalidade às aplicações existentes. Poderá efectuar todos estes procedimentos visualmente, utilizando um construtor, ou manualmente, chamando os métodos de beans de um programa.

Os Beans são, essencialmente, classes de Java. Poderá transformar quase todos os componentes de programação e classes de java existentes num Bean. Em geral, toda a classe de Java que adira a convenções específicas relacionadas com definições de propriedades e de interface de eventos pode ser considerada um Bean.

Os Beans definem uma interface na altura da concepção que permite a ferramentas de autor, ou de construtor, de aplicações, consultar componentes para determinar os tipos de propriedades que estes componentes definem e os tipos de eventos que geram ou aos quais respondem. Não tem de utilizar a análise introspectiva e ferramentas de construção especiais ao trabalhar com Beans. As assinaturas padrão estão bem definidas e podem ser facilmente reconhecidas e compreendidas através de inspecção visual.

Os Beans possuem as seguintes características:

Análise introspectiva

A análise introspectiva é o processo através do qual uma ferramenta de construtor determina e analisa como um Bean funciona na altura da concepção e da execução. Devido ao facto dos Beans serem codificados com padrões predefinidos para as suas assinaturas de métodos e definições de classe, as ferramentas que reconhecem estes padrões têm a capacidade de analisar um Bean e determinar as suas propriedades e comportamento. Cada Bean possui uma classe de informação relacionada de Bean, que faculta informações relativas a propriedades, métodos e eventos sobre o Bean. Cada classe de informação de Bean implementa uma interface BeanInfo, que enumera explicitamente as características de Bean que serão expostas a ferramentas de construtor de aplicação.

Propriedades

As propriedades controlam a aparência e comportamento de um Bean. As ferramentas de construtor analisam introspectivamente um Bean de forma a descobrir as suas propriedades e a expor essas propriedades para manipulação. Isto permite ao utilizador alterar a propriedade de um bean na altura da concepção.

Personalização

As propriedades expostas de um bean podem ser personalizadas na altura da concepção. A personalização permite-lhe alterar a aparência e/ou comportamento de um Bean. Os Beans suportam a personalização através da utilização de editores de propriedades e da utilização de personalizadores especiais e sofisticados de Beans.

Eventos

Os Beans utilizam eventos para comunicar com outros Beans. Os Beans podem transmitir eventos, o que significa que o Bean envia um evento a outro Bean. Quando um Bean transmite um evento este é considerado um Bean origem. Um Bean pode também receber um evento, sendo neste caso considerado um Bean receptor. Um Bean receptor regista o seu interesse no evento junto do Bean origem. As ferramentas de construtor utilizam a análise introspectiva para determinar os eventos que um bean envia e os eventos que recebe.

Permanência

Os Beans utilizam a serialização de objectos Java implementando a interface `java.io.Serializable` para guardar e repor estados que poderão ter sido alterados devido a uma personalização. Por exemplo, o estado é guardado quando uma aplicação personaliza um Bean num construtor de aplicações, de modo a que as propriedades alteradas possam ser repostas numa altura posterior.

Métodos

Todos os métodos de Bean são idênticos a métodos de outras classes de Java. Os métodos de Bean podem ser chamados por outros Beans ou através de linguagens de realização de script. Por predefinição, todos os métodos públicos de um Bean são exportados.

Utilizar JavaBeans em construtores

Esta secção explica como utilizar JavaBeans no IBM Websphere Studio Application Developer e noutros construtores.

Para utilizar outros construtores que não o IBM Websphere Studio Application Developer, certifique-se de que o construtor suporta Java 2. Siga as instruções do

construtor para adicionar novos ficheiros jar aos jars especificados nas instruções que se seguem. Depois, siga as instruções do construtor para adicionar beans de um jar, por forma a adicionar beans de EIP, que estão em cmb81.jar.

Importante: Para utilizar os beans, deve possuir, pelo menos, o Java JDK 1.3.1 ou superior.

O directório CMBROOT\Samples\java contém exemplos de código de beans não visuais.

Utilizando o IBM Websphere Studio Application Developer

O utilizador pode utilizar beans não visuais para construir servlets e páginas de JSP no Webphere Studio Application Developer concluindo os passos seguintes:

1. Criar um projecto na Web para a aplicação da Web.
2. Nas propriedades do projecto, em Java Build Path | Libraries, especifique os seguintes ficheiros JAR:
 - \CMBROOT\lib\cmb81.jar
 - \CMBROOT\lib\cmbview81.jar
 - \CMBROOT\lib\cmbsdk81.jar
 - \CMBROOT\lib\esclisrv.jar
 - \SQLLIB\java\db2java.zip
3. Se planeia utilizar os servlets do EIP e o taglib de JSP, deve especificar também os seguintes ficheiros:
 - \CMBROOT\lib\cmbervlet81.jar
 - \CMBROOT\lib\cmbtag81.jar

Para a biblioteca de identificadores, também terá de importar o ficheiro taglib.tld para o taglib de JSP do EIP para a aplicação da web:

- Copiar \CMBROOT\lib>taglib.tld para o directório webApplication\WEB-INF na aplicação da web.
- Configurar o taglib existente no ficheiro webApplication\WEB-INF\web.xml na aplicação da web adicionando o seguinte:

```
<taglib>
    <taglib-uri>cmb</taglib-uri>
    <taglib-location>/WEB-INF/taglib.tld</taglib-location>
</taglib>
```

4. Uma vez que os JARs acima enumerados contêm classes de J2EE, o utilizador terá de incluir o JAR de J2EE, que normalmente está localizado no directório:
Program Files\IBM\Application
Developer\plugins\com.ibm.etools.websphere.runtime\lib\j2ee.jar

Invocar os beans de Java do EIP

Os Beans no nível do EIP podem ser chamados de duas formas. O utilizador pode chamá-los directamente utilizando as suas interfaces públicas (métodos públicos). Neste caso, as excepções explícitas de Java são devolvidas para indicar eventos de erro.

Outro método através do qual pode chamar a funcionalidade nos beans a nível da sessão é ligar quaisquer instâncias deste tipo de bean a outros Beans de EIP utilizando eventos de pedido e resposta. Ao utilizar este método, tenha em consideração o seguinte:

- O bean CMBConnection recebe eventos de pedido de ligação e responde transmitindo eventos de resposta de ligação.
- O bean CMBDataManagement recebe eventos de pedido de dados e transmite de volta eventos de resposta de dados.
- O bean CMBSchemaManagement recebe eventos de pedido de esquema e transmite de volta eventos de resposta de esquema.
- O bean CMBQueryService recebe eventos de pedido de pesquisa e transmite de volta eventos de resposta de pesquisa.
- O bean CMBWorkflowDataManagement recebe eventos de pedido de dados de fluxo de trabalho e transmite de volta eventos de resposta de dados de fluxo de trabalho.
- O bean CMBWorkflowQueryService recebe eventos de pedido de lista de trabalhos e transmite de volta eventos de resposta de lista de trabalhos.

Beans não visuais

O EIP faculta um conjunto de JavaBeans não visuais que pode utilizar para construir aplicações Java. Os beans não visuais são um conjunto de classes de Java que seguem as convenções de Beans. São construídos utilizando as classes de conector de Java do EIP e do CM. Uma utilização normal será a construção de Servlets ou Java Server Pages (JSPs), apesar de também poderem ser utilizados numa linha de comandos ou em aplicações de Windows.

Os benefícios da utilização de beans não visuais são as seguintes:

- Facultam um mecanismo de acesso associado e um modelo de programação comum para os vários e diferentes conectores que são facultados juntamente com o EIP.
- Permitem ao utilizador programar a um nível superior de abstracção.
- Ocultam a complexidade e detalhes de conectores individuais.
- Permitem-lhe potenciar o suporte de Beans incorporado na maioria dos ambientes de desenvolvimento comercial.

Utilizar os beans facilita a construção de aplicações básicas; no entanto, existem algumas limitações de que deve estar consciente antes de iniciar a utilização de um bean. Os Beans não:

- Facultam todas as funções disponíveis no nível de API de Java como, por exemplo, a funcionalidade administrativa ou de configuração.
- Suportam a importação de lotes. Os beans apenas suportam capacidades de importação de tipo de artigo único e casual. Os processos de lote para importar e exportar grandes quantidades de dados devem ser escritos utilizando as interfaces do conector.
- Facultam toda a funcionalidade suportada por todos os servidores. Determinada funcionalidade é específica do servidor. Por exemplo, a funcionalidade relacionada com sub-artigos dentro de artigos está ligada à funcionalidade disponível apenas no conector do Content Manager Versão 8.

As limitações acima enumeradas podem, em alguns casos, ser ultrapassadas utilizando métodos de acesso que permitem o acesso às APIs de Java de base.

Importante: Os beans do EIP não são Enterprise Java Beans (EJB). Consequentemente, não podem ser directamente detidos no ambiente gerido,

facultado por contentores como o IBM Websphere. No entanto, podem ser utilizados a partir do EJBs como o mecanismo de ligação de base de repositórios de dados não estruturados.

Configurações de beans não visuais

Os beans não visuais têm configurações locais, remotas e dinâmicas.

locais Liga directamente ao servidor de conteúdos.

remotas

Liga directamente ao servidor de conteúdos utilizando um servidor RMI.

dinâmicas

Activa uma aplicação que executa uma comutação dinâmica entre local e remoto com base no ficheiro `cmbcs.ini`. O ficheiro `cmbcs.ini` especifica se o servidor de conteúdos é local ou remoto.

Compreender as funções dos beans não visuais

Pode utilizar os Beans de EIP em JSPs pois as suas propriedades são, normalmente, tipos simples como, por exemplo, sequências e tabelas. No fundo, funcionam como o componente modelo de aplicações da Web, pois são modelados utilizando o padrão de concepção do Model View Controller (MVC). Tenha em atenção que o componente de visualização normalmente é formado por JSPs e pelo componente controlador de servlets (como os do conjunto de servlet de EJB). De seguida é apresentada uma lista das características dos beans não visuais do EIP:

- Facultam acesso às definições do esquema no servidor de bibliotecas.
- Facultam métodos CRUD (criar, obter, actualizar, eliminar) para documentos, artigos simples (que não são de recurso) e artigos de recurso em todos os repositórios suportados pelo EIP.
- Facultam funcionalidade para pesquisar e obter documentos, artigos simples e artigos de recurso em todos os repositórios suportados pelo EIP.
- Suportam a conversão de tipos de dados em formatos visualizáveis.
- Funcionam como um nível associado que aplica um conjunto coerente de semânticas em todos os repositórios de gestão de conteúdos suportados pelo EIP.
- Integra e expõe a funcionalidade facultada nos serviços de fluxo de trabalho e de extracção de informações do EIP.
- Facultam serviços de extracção e conversão de documentos, bem como suporte para a gestão de anotações de documentos.
- Facultam a funcionalidade de ordenação e conversão.
- Facultam eventos que são transmitidos para acções chave nos beans membros como, por exemplo, eventos de ligação e de anulação de ligação, eventos de notificação de resultados de pesquisa e eventos de notificação de alteração de conteúdo.

Categorias de beans não visuais

Os JavaBeans não visuais podem ser divididos nas seguintes categorias:

Beans de armazenamento de dados

Estes beans existem em sessões de utilizador normais e facultam ao utilizador serviços especializados. Os beans a nível da sessão incluem:

- **CMBCConnection** Este bean mantém a ligação a um servidor de fundo que pode ser um servidor de conteúdos nativo ou um servidor associado. Este bean é necessário para utilizar um JavaBean.

- **CMBSchemaManagement** Utilizado para trabalhar com metadados de repositórios.
- **CMBDataManagement** Utilizado para trabalhar com dados de repositórios.
- **CMBQueryService and CMBSearchResults** Utilizado para executar consultas e trabalhar com os resultados de consultas.
- **CMBWorkflowDataManagement and CMBWorkflowQueryService** Utilizado para trabalhar com processos de fluxo de trabalho avançados do EIP.
- **CMBDocRoutingDataManagement and CMBDocRoutingQueryService** Utilizado para trabalhar com processos de encaminhamento de documentos do CM v8.
- **CMBDocumentServices** Utilizado para os serviços de emissão e anotação de documentos.

Beans auxiliares

Os beans auxiliares existem no contexto de um ou mais do que um bean a nível da sessão e são primariamente utilizados para a encapsulação de valores de dados e para facultar serviços aos beans a nível da sessão. Os beans auxiliares disponíveis incluem:

- **CMBEntity** Representa definições de artigos de dados disponíveis nos repositórios de gestão de conteúdos. Por exemplo, em relação a um repositório de CM v8, uma CMBEntity representa as definições de tipos de artigo e de componentes descendentes, enquanto que em relação a um repositório de CM v7, uma CMBEntity representa uma classe de índice. A CMBEntity é uma classe auxiliar de CMBSchemaManagement.
- **CMBAttribute** Representa definições de atributos nos repositórios. A CMBAttribute é uma classe auxiliar de CMBSchemaManagement.
- **CMBSearchTemplate** Representa um modelo de pesquisa associada. A CMBSearchTemplate é uma classe auxiliar de CMBSchemaManagement.
- **CMBSTCriterion** Representa um critério de dados que faz parte de um modelo de pesquisa associada. A CMBSTCriterion é uma classe auxiliar de CMBSchemaManagement.
- **CMBItem** Representa instâncias de documentos, artigos de recurso e artigos que não são de recurso. A CMBItem é uma classe auxiliar de CMBDataManagement.
- **CMBObject** Representa instâncias de artigos de recurso, partes de base e partes de diário de notas. A CMBObject também é utilizada para representar atributos de BLOB. A CMBObject é uma classe auxiliar de CMBDataManagement.
- **CMBAnnotation** Representa instâncias de partes de anotação para repositórios CM v8 e notas para repositórios OnDemand.
- **CMBPrivilege** Faculta a funcionalidade necessária para obter informações relacionadas com privilégios de um repositório suportado por CM ou EIP.

Beans Acessórios

Os Beans acessórios não são essenciais nas aplicações, mas podem ser úteis para realçar a funcionalidade. De seguida é apresentada uma lista de beans acessórios.

- **CMBConnectionPool** Utilizado para facultar serviços de agrupamento a beans CMBConnection. A classe de API de Java DKDatastorePool é utilizada para manter as instâncias DKDatastore que são utilizadas por

instâncias **CMBCConnection**. Ao mudar o agrupamento para o nível de API de Java, os servidores de conteúdos podem ser geridos mais eficazmente, pois podem ser agrupados de modo mais racional, com base no tipo de servidor.

- **CMBUserManagement** Utilizado em ligações a repositórios associados para gerir as definições de correspondências de utilizadores associados a utilizadores de servidores de nativos.
- **CMBExceptionSupport** Faculta uma estrutura para o tratamento comum de eventos de excepção.
- **CMBTraceLog** Faculta uma estrutura para o tratamento comum de eventos de rastreio e faculta capacidades de recepção de eventos de rastreio transmitidos por outros beans.

Beans de fluxo de trabalho

Os beans de fluxo de trabalho facultam serviços de fluxo de trabalho. Os serviços de fluxo de trabalho facultados pelo nível de Beans de EIP suportam dois tipos de sistemas de fluxo de trabalho: o fluxo de trabalho avançado e o encaminhamento de documentos. A funcionalidade Fluxo de Trabalho Avançado é construído recorrendo ao Fluxo de Trabalho MQSeries, enquanto o encaminhamento de documentos consiste num sistema de fluxo de trabalho integrado no produto CM V8 e no conjunto de APIs. O nível de beans faculta o conjunto completo de objectos necessários à criação de definições de fluxo de trabalho, à execução de instâncias de fluxo de trabalho com base nas definições criadas e à gestão de instâncias de fluxos de trabalhos em execução. Os beans seguintes constituem os principais componentes do suporte de fluxo de trabalho avançado:

- **CMBWorkflowDataManagement** Este bean é utilizado para criar e trabalhar com instâncias de fluxo de trabalho avançado. Uma instância deste bean pode ser obtido a partir do objecto **CMBCConnection**. Este bean faculta suporte a:
 - Início, paragem, suspensão e retoma de uma instância de fluxo de trabalho.
 - Transferência de artigos e notificações de trabalho de um utilizador para outro.
 - Cancelamento de notificações de trabalho.
- **CMBWorkflowQueryService** O **CMBWorkflowQueryService** faculta uma interface para consultar informações relacionadas com fluxos de trabalho avançados. Uma instância deste bean pode ser obtido a partir do objecto **CMBCConnection**. Este bean faculta suporte para a obtenção das seguintes informações:
 - Informações relativas a fluxos de trabalho no sistema.
 - Informações relativas a artigos de trabalho que se deslocam no sistema como parte de fluxos de trabalho activos.
 - Informações relacionadas com listas de trabalhos.
 - Informações relativas a todas as notificações de trabalho registadas.

Os beans seguintes constituem os principais componentes do suporte de encaminhamento de documentos.

- **CMBDocRoutingManagementICM** Este bean é utilizado para criar e gerir processos de encaminhamento de documentos. Pode obter uma instância deste bean a partir de uma instância do objecto **CMBCConnection**. Este bean faculta suporte para as seguintes funções:

- Início, paragem, suspensão e retoma de uma instância de encaminhamento de documentos.
- Dar saída de um artigo de CM contido num pacote de trabalhos.
- Definição de propriedades de pacotes de trabalho a serem enviados por uma instância de encaminhamento de documentos.,
- **CMBDocRoutingQueryServiceICM** Este bean faculta uma interface para consultar informações facultadas por processos de encaminhamento de documentos. Pode obter uma instância deste bean a partir de uma instância do objecto CMBConnection. Este bean faculta suporte para a obtenção das seguintes informações:
 - Informações relacionadas com os pacotes de trabalho a serem encaminhados através do sistema de encaminhamento de documentos.
 - Informações relacionadas com os processos que actualmente se encontram activos no sistema.
 - Informações relativas a todas as listas de trabalhos que se encontram no sistema.
 - Todos os nós de trabalho que fazem parte do sistema.

Beans de Information Mining

Os beans de Information Mining do EIP permitem a uma aplicação incorporar análise de texto e tecnologia de extracção. Os beans de extracção de informações facultam as seguintes funcionalidades:

- Resumo e categorização de texto, criação de um resumo de documento.
- Categorização, atribuição de uma categoria a um documento.
- Suporte para a extracção de informações relevantes de um documento.
- Suporte para identificar o idioma em que um documento está escrito.
- Suporte para agrupar documentos semelhantes num conjunto de documentos.
- Pesquisa de texto de documentos no catálogo ou restrita a documentos de uma determinada categoria.
- Suporte para aceder a documentos que são continuamente obtidos a partir da Web utilizando o IBM Web Crawler.

Os beans de extracção de informações incluem:

- **CMBCatalogService** Faculta uma interface para obter informações relacionadas com um catálogo. Faculta também suporte para importar para o servidor de conteúdos artigos criados a partir de operações de extracção de informações. Todas as operações de extracção de informações são limitadas por um catálogo. Cada catálogo encontra-se associado a uma taxonomia que, por sua vez, consiste numa hierarquia de categorias.
- **CMBAdvancedSearchService** Faculta suporte para concluir pesquisas de texto em informações contidas no catálogo. Os métodos neste bean permitem-lhe controlar o catálogo no qual a pesquisa irá ser realizada, bem como o conteúdo e a quantidade de resultados gerados pela pesquisa de texto.
- **CMBCategorizationService** Este bean é utilizado para determinar categorias de documentos com base num catálogo especificado.
- **CMBSummarizationService** Faculta a funcionalidade necessária para gerar resumos de documentos.

- **CMBClusteringService** Utilizado para agrupar documentos em conjuntos com base na semelhança dos seus conteúdos.
- **CMBWebCrawlerService** Faculta uma interface para gerir os resultados de acções do Web Crawler. Permite ao utilizador iniciar pedidos de web crawling em espaços da web específicos e gerir resultados. Os resultados criados podem ser categorizados, resumidos e importados para um servidor de conteúdos de fundo.

Beans de Serviços de Documento

Os beans de serviços de documento facultam serviços de emissão e anotação de documentos. Os seguintes beans fazem parte da sub-secção de serviços de documento:

- **CMBDocumentServices** - O bean CMBDocumentServices faculta serviços necessários para trabalhar com documentos, incluindo a funcionalidade necessária para produzir, converter e reconstituir as páginas de um ou mais documentos.
- **CMBDocument** - Este bean representa a entidade criada ao carregar um documento utilizando CMBDocumentServices. Essencialmente, o CMBDocument é um contentor das páginas do documento. O CMBDocument também permite ao utilizador consultar e definir propriedades que controlam as características de um documento.
- **CMBPage** - Este bean faculta uma representação de uma página específica num documento. A funcionalidade desta classe permite ao utilizador especificar e controlar as propriedades de um conjunto de imagens passíveis de serem transmitidas que podem ser geradas para a página.
- **CMBPageAnnotation** Este bean modela uma anotação que pode ser associada a uma página de um documento. Todas as notações suportadas são modeladas por sub-classes deste bean. Para além disso, o próprio CMBPageAnnotation contém propriedades como, por exemplo, a página em que se encontra a anotação e o tipo de anotação. As sub-classes incluem:
 - CMBArrowAnnotation
 - CMBCircleAnnotation
 - CMBHighlightAnnotation
 - CMBLineAnnotation
 - CMBNoteAnnotation
 - CMBPenAnnotation
 - CMBRectAnnotation
 - CMBStampAnnotation
 - CMBTextAnnotation

Outras classes de Beans

As classes de Beans enumeradas nesta secção facultam uma variedade de funcionalidades, que são descritas de seguida.

- **Classes de BeanInfo** Permite a exposição explícita das funções dos beans de EIP numa classe diferente e associada que implementa a interface BeanInfo.
- **Classes de excepção** Utilizadas para encapsular excepções no nível de Beans. Todas as classes de excepção herdam da classe de base CMBException. Cada uma das sub-classes de CMBException indica uma condição de erro específica. Em cada caso, as propriedades do objecto de excepção podem ser utilizadas para obter informações de erro

detalhadas relativas à condição de erro que conduziu à «» da excepção. Quando os beans são utilizados de modo direccionado para os eventos, as excepções são devolvidas em eventos.

- **Classes de evento e de receptor** Implementam o modelo standard de receptor de eventos de Beans. As classes têm de pedir explicitamente um evento, implementando a interface receptora associada ao evento e registando essa interface receptora junto do objecto que gera um evento. O nível de Beans de EIP faculta pares de eventos e de receptores para operações de acesso a esquema, operações de acesso a dados, operações de fluxo de trabalho e operações de pesquisa.
- **Receptores de sessão** Estas são as classes de receptores que existem a nível da sessão. Nos beans de EIP, existem receptores de sessão actuais que identificam pedidos de ligação e eventos de resposta de ligação.

Considerações a ter aquando da utilização de beans não visuais

Pode utilizar os beans não visuais para activar aplicações com fins gerais com a funcionalidade necessária para aceder a repositórios de gestão de conteúdos suportados pelo EIP. Esta secção contém algumas sugestões sobre os padrões específicos de utilização nos beans.

Singletons nos Beans O CMBConnection possui métodos para obter acesso a instâncias dos outros Beans de EIP a nível da sessão. Quando se obtém desta forma os beans a nível da sessão como, por exemplo, CMBSchemaManagement e CMBDataManagement, estes já se encontram ligados ao bean CMBConnection (a partir do qual foram obtidos) para serem informados de uma ligação ou anulação de ligação e para partilhar operadores de eventos de rastreio e excepção. É criada apenas uma instância de cada um dos beans a nível de sessão. Se estes métodos forem repetidamente chamados, é devolvida a mesma instância (padrão de concepção singleton). Se forem criados beans a nível da sessão na aplicação, e não por parte do bean CMBConnection, estes devem ser ligados ao bean CMBConnection para serem utilizados.

Considerações relativas à modulação nos Beans

Uma instância única do bean CMBConnection só pode ser utilizada num módulo único em qualquer altura. Esta restrição estende-se a todos os outros beans associados ao bean CMBConnection (através da propriedade de ligação de um bean associado). Isso significa que deve criar ligações diferentes para cada módulo. Opcionalmente, vários módulos podem obter e libertar ligações utilizando o bean CMBConnectionPool. Consequentemente, cada módulo deve obter, utilizar e libertar uma ligação.

Todos os beans a nível de sessão possuem afinidades com uma instância de CMBConnection a partir da qual foram obtidos ou aos quais foram associados após a criação. Isto implica que uma instância dos beans a nível de sessão como, por exemplo, o CMBSchemaManagement, só pode ser utilizada por um módulo em qualquer altura. Se a instância do bean a nível de sessão for utilizada por vários módulos, deve realizar uma sincronização explícita na sua aplicação, de modo a garantir que apenas um módulo utiliza activamente a instância de bean a nível de sessão em qualquer altura.

Todos os beans a nível de sessão também recebem eventos de resposta de ligação gerados pelos beans CMBConnection. Isto permite-lhes reconhecer que o repositório de conteúdos de base ao qual a instância do bean CMBConnection está associada mudou, de modo a que o bean possa aplicar a acção adequada.

Ao contrário de CMBConnection, o bean CMBConnectionPool destina-se a ser utilizado por vários módulos. Vários módulos podem chamar simultaneamente

os métodos relacionados com a obtenção e libertação de objectos de ligação. Todas as ligações obtidas a partir do conjunto são instâncias de `CMBCConnection` e só podem ser acedidas por um único módulo. Todas as ligações obtidas a partir do bean de conjunto de ligações devem ser devolvidas ao conjunto o mais depressa possível após a sua utilização, de modo a que sejam disponibilizadas a outros módulos que possam estar a pedir ligações do conjunto.

Rastrear e iniciar sessão nos Beans

Pode activar o rastreio em todos os beans a nível de sessão no nível de Beans de EIP. Activar o rastreio numa instância do bean `CMBCConnection` também permite o rastreio em qualquer Bean de EIP obtido a partir deste bean de ligação, incluindo beans de gestão de esquemas, de gestão de dados, de serviço de consulta e de fluxo de trabalho.

Após a activação do rastreio, são transmitidos eventos de rastreio. O `CMBTraceLog`, um bean de utilitário, pode receber eventos de rastreio e escrever registos de rastreio num registo, `stdout`, `stderr` ou janela definidos (quando utilizado com os beans visuais).

Todos os beans a nível de sessão também recebem eventos de rastreio. A funcionalidade de rastreio nos beans escreve informações de registo no mesmo ficheiro de registo da API de Java, caso `log4j` seja utilizado para iniciar a sessão.

Compreender propriedades e eventos de beans não visuais

Cada bean não visual fornece o seguinte

- Propriedades importadas, proibidas ou não
O valor da propriedade é determinado por outros beans na altura do arranque por eventos `PropertyChange` ou `VetoableChange`. Os beans que têm propriedades de importação têm de detectar eventos de `PropertyChange` ou `VetoableChange`.
- Propriedades exportadas, proibidas ou não
Um bean não visual terá de possuir uma propriedade reduzida e outros beans poderão ter interesse no seu valor. Sempre que o seu valor é alterado, o bean é responsável pela geração de um evento `PropertyChange` ou `VetoableChange`.
- Propriedades autónomas
Estes beans são os únicos com interesse no seu valor de propriedade.
- Eventos gerados por este bean
- Eventos em que este bean está interessado

Construir uma aplicação através de beans não visuais

Exemplo de aplicação que não é de Interface Gráfica de Utilizador (GUI)

O exemplo nesta secção utiliza beans não visuais para criar um exemplo de aplicação que não é GUI. A aplicação exemplificativa inclui todos os beans, excepto o bean `CMBUserManagement`. A aplicação exemplo completa da qual foi retirada este exemplo (`DemoSimpleApp1.java`) está disponível no directório `Cmbroot/Samples/java/beans`. A aplicação exemplificativa demonstra como:

1. Ligar ao servidor (federado) do Enterprise Information Portal
2. Obter uma lista dos nomes de modelos de pesquisas

3. Utilizar o nome dos modelos de pesquisas para obter uma lista de nomes de critérios de pesquisa
4. Seleccionar um modelo de pesquisa e obter os critérios de pesquisa
5. Concluir os valores de pesquisa e submeter uma consulta
6. Imprimir o resultado através do bean de resultados da pesquisa
7. Seleccionar uma fila de resultados e apresentá-la
8. Desligar do servidor

Trabalhar com beans visuais

Os beans visuais permitem ao utilizador integrar a funcionalidade do Enterprise Information Portal ou outros servidores de conteúdos nas aplicações Java baseadas no Swing. Os beans visuais executam tarefas básicas que são comuns a muitas aplicações, tais como iniciar sessão, pesquisar, apresentar e visualizar resultados, actualizar documentos e visualizar informações sobre a versão.

Cada bean visual tem uma propriedade Ligação. Esta propriedade tem de remeter para uma ocorrência de CMBConnection, o bean não visual que mantém a ligação ao servidor de conteúdos. Uma aplicação construída com os beans visuais do EIP também terá de conter uma ocorrência do bean não visual CMBConnection.

CMBLogonPanel

Este bean apresenta um painel para iniciar uma sessão em Enterprise Information Portal ou em servidores de conteúdo como, por exemplo, Content ManagerVersion 8.2 (CM 8.2). Também fornece a janela onde os utilizadores federados podem alterar os IDs de Utilizador e palavras-passe nos servidores de conteúdos.

CMBSearchResultsViewer

Este bean apresenta os resultados da pesquisa. Quando o resultado da pesquisa devolver pastas, utilize o bean CMBSearchResultsViewer para "penetrar" na pasta e averiguar o seu conteúdo. Os artigos nos resultados da pesquisa ou nas pastas podem ser seleccionados e abertos para visualização numa janela do tipo do Windows Explorer

CMBSearchTemplateList

Para servidores que suportem modelos de pesquisa, este bean apresenta uma lista dos modelos de pesquisa disponíveis e permite a selecção de um modelo.

CMBSearchTemplateViewer

Para servidores que suportam esses modelos de pesquisa, este bean apresenta um modelo de pesquisa e fornece campos para os utilizadores inserirem os critérios de pesquisa. Executa uma pesquisa baseada nesses critérios.

CMBSearchPanel

Para todos os servidores, o painel de pesquisa apresenta uma lista das entidades disponíveis e fornece campos para os utilizadores inserirem os critérios de pesquisa. Executa uma pesquisa baseada nesses critérios. O CMBSearchPanel é útil para executar pesquisas nos servidores de conteúdos que não suportem modelos de pesquisa.

CMBFolderViewer

Apresenta os conteúdos de uma ou mais pastas numa janela de estilo do Windows Explorer

CMBItemAttributesEditor

Apresenta uma janela onde os utilizadores podem actualizar a classe de índices e atributos de indexação para um artigo

CMBDocumentViewer

Apresenta um ou mais documentos através da iniciação do visualizador apropriado

CMBVersionsViewer

Apresenta as informações sobre a versão para um documento, se estiver activada a atribuição de versão.

Bean CMBLogonPanel

O bean do CMBLogonPanel (consultar Figura 20) apresenta uma janela que permite aos utilizadores iniciar sessão num servidor de conteúdos, actualizar correlações de utilizadores e alterar uma palavra-passe.

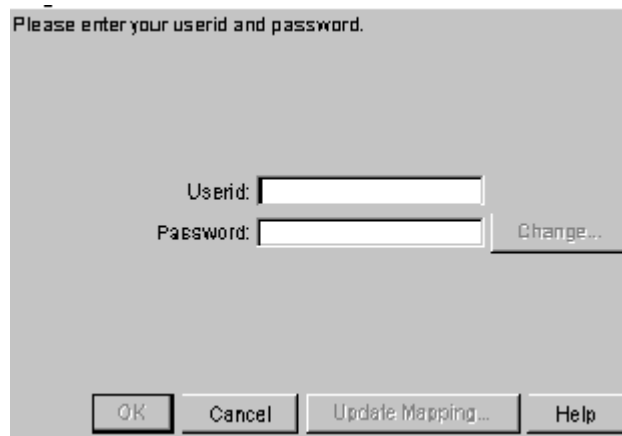


Figura 20. Janela do bean CMBLogonPanel

No bean CMBLogonPanel, quando um utilizador faz clique sobre **Alterar**, surge a janela **Alterar palavra-Passe** (consultar Figura 21.) O utilizador insere a palavra-passe antiga e insere duas vezes a nova palavra-passe.



Figura 21. Janela Alterar Palavra Passe

No bean CMBLogonPanel, quando um utilizador faz clique sobre **Actualizar Definição de Correspondências** na janela de Início de Sessão, é apresentada a

janela **Actualizar Definição de Correspondências de ID de Utilizador** (consultar Figura 22). Quando Actualizar DefCorrespondências, estará a actualizar o ID de utilizador e a palavra-passe especificados para um servidor. Esta função está disponível apenas quando inicia sessão na base de dados associada do Enterprise Information Portal.

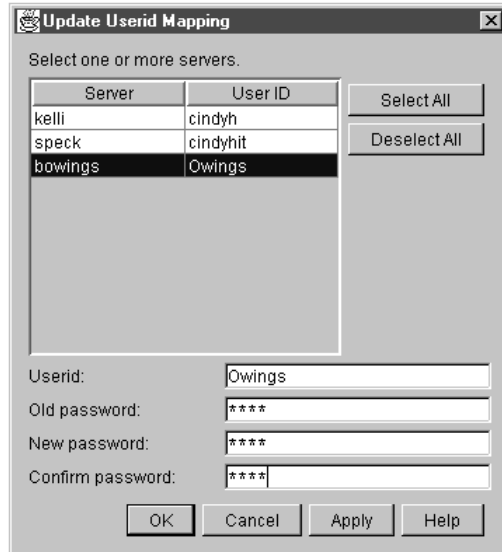


Figura 22. Janela de Definição de Correspondências de ID de Utilizador

No topo da janela está uma lista de todos os servidores e um ID de utilizador correspondente. Os utilizadores podem seleccionar um ou mais servidores da lista. Faça clique sobre **Seleccionar Todos** para seleccionar todos os servidores. Os utilizadores podem especificar um novo ID de utilizador e (opcionalmente) uma nova palavra-passe após a selecção de um ou mais servidores. Caso seleccione um servidor, o ID de utilizador surge no campo **ID de Utilizador**. Se os utilizadores seleccionarem mais do que um ID de utilizador, o campo **ID de Utilizador** fica em branco.

Anular todas as selecções

Remove todas as selecções do servidor.

Aplicar

Faça clique para aplicar as alterações de definição de correspondências e de palavra-passe sem fechar a janela.

OK Faça clique para aceitar as alterações e fechar a janela.

Cancelar

Faça clique para fechar a janela sem efectuar alterações.

Bean CMBSearchTemplateList

O bean CMBSearchTemplateList tem três estilos. O estilo da imagem, demonstrado na Figura 23 na página 421, utiliza uma imagem para os segundos planos dos artigos seleccionados e outra para os artigos não seleccionados. A Figura 24 na página 421 demonstra o estilo da lista de modelos simples. A Figura 25 na página 421 demonstra o estilo dos modelos da lista pendente.

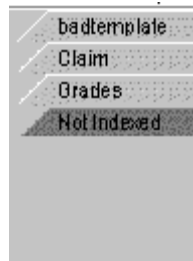


Figura 23. Estilo da lista dos modelos de imagens

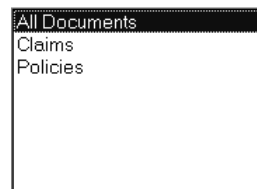


Figura 24. Estilo da lista dos modelos simples



Figura 25. Estilo dos modelos da lista pendente

Bean CMBSearchTemplateViewer

O bean CMBSearchTemplateViewer (consultar a Figura 26) apresenta uma janela onde os utilizadores podem especificar critérios de pesquisa segundo o modelo de pesquisa definido pelo administrador do sistema. O bean CMBSearchTemplateViewer inicia uma pesquisa e gera o evento CMBSearchResults para devolver os resultados da pesquisa.

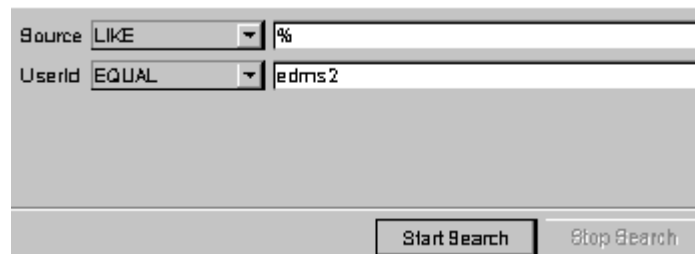


Figura 26. Bean CMBSearchTemplateViewer

O bean CMBSearchTemplateViewer enumera critérios de pesquisa como a Fonte ou o Idutilizador. Cada critério de pesquisa tem um identificador, uma caixa pendente do operador e um campo de texto. O ecrã do operador BETWEEN ou NOTBETWEEN apresenta dois campos de texto. Os operadores IN ou NOTIN têm de possuir uma área de texto multilinha. Cada valor deve ser inserido numa linha em separado.

Áreas de Pesquisa de Texto

O bean CMBSearchTemplateViewer também pode conter áreas que permitem aos utilizadores executar uma pesquisa em atributos totais de texto ou de índice. Uma área de pesquisa de texto total no modelo pode ser tão simples como um campo de texto com uma etiqueta.

Os utilizadores têm de fazer corresponder a sintaxe da consulta para um pesquisa de texto livre ou booleano, quando inserirem a cadeia de pesquisa no campo de texto (consultar a classe DKDatastoreTS). Consulte a referência de API online para obter mais pormenores.

Validar ou editar campos de CMBSearchTemplateViewer

O utilizador poderá fornecer lógica de validação ao bean de CMBSearchTemplateViewer para modificar o critério de pesquisa inserido pelo utilizador. Execute esta acção fornecendo um operador ao CMBTemplateFieldChangedEvent. Os valores actuais dos critérios de pesquisa estão armazenados no CMBTemplate devolvido pelo método getTemplate anterior à chamada deste evento. O utilizador pode examinar e alterar os critérios. Após a conclusão da manipulação do evento, surgem os novos valores.

Bean CMBSearchPanel

O bean CMBSearchPanel apresenta uma janela onde os utilizadores podem especificar os critérios de pesquisa de acordo com as entidades disponíveis no servidor de conteúdos actual. o bean CMBSearchPanel lança uma pesquisa e gera o CMBSearchResultsEvent para devolver os resultados da pesquisa. O CMBSearchPanel enumera todas as entidades disponíveis na lista pendente na parte superior da janela. Quando uma entidade é seleccionada, o CMBSearchPanel apresenta os atributos da entidade. Cada atributo tem um identificador, uma caixa pendente do operador e um campo de texto. Uma visualização do operador de intervalo, tal como BETWEEN ou NOTBETWEEN, tem dois campos de texto. Um operador que permita valores múltiplos, tal como o operador IN ou NOTIN, tem uma área de texto de multi-linhas. Cada valor deve ser inserido numa linha em separado na área de texto de multi-linhas.

Bean CMBSearchResultsViewer

O bean CMBSearchResultsViewer apresenta resultados de pesquisa numa janela que tem uma área de janela em árvore e uma área de janela pormenorizada. Os utilizadores podem redimensionar a janela fazendo clique sobre e arrastando a linha que separa as áreas de janela.

A Figura 27 mostra o bean CMBSearchResultsViewer com a pasta **Resultados da Pesquisa** seleccionada.

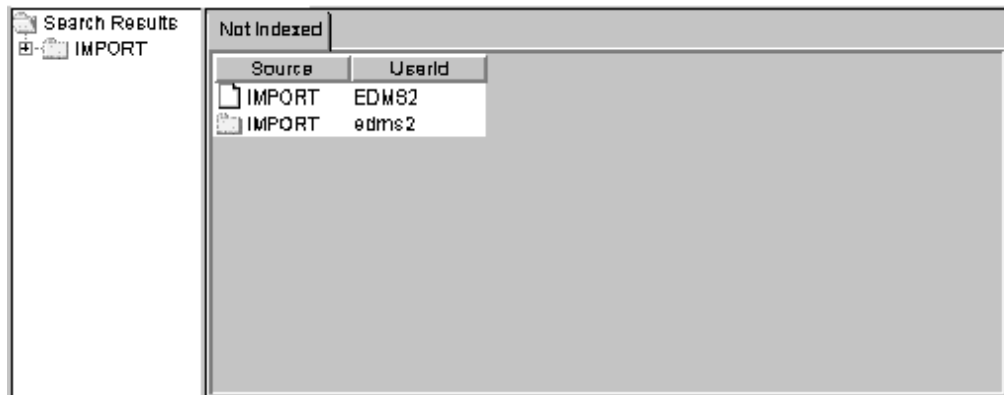


Figura 27. Bean CMBSearchResultsViewer

Área de janela em árvore de CMBSearchResultsViewer

A área de janela (à esquerda) contém uma pasta principal com a etiqueta

Resultados da Pesquisa. Por baixo dessa pasta estão todas as pastas localizadas na pesquisa. A área de janela em árvore é opcional. Remova-a definindo a propriedade `TreePaneVisible`: `setTreePaneVisible(falso)`.

Área de janela Pormenores de `CMBSearchResultsViewer`

A área de janela de pormenores apresenta o conteúdo de cada pasta seleccionada na área de janela em árvore. Quando os utilizadores seleccionam a pasta **Resultados da Pesquisa**, surge um separador no bloco de notas que contém o nome do modelo da pesquisa. Quando os utilizadores seleccionam uma pasta dentro de **Resultados da Pesquisa**, surge um ou mais separadores: um por cada classe de índices na pasta. Os nomes dos separadores têm o formato:

classe de índices @ servidor

onde *classe de índice* é a classe de índice ou nome do tipo de artigo e *servidor* é o nome do servidor de conteúdos. As colunas da tabela são alteradas para visualizar os atributos de acordo com a classe de índice ou o tipo de artigo. É suportada selecção múltipla na área de janela de pormenores. Desactive a Selecção múltipla definindo a propriedade `MultiSelectEnabled`: `setMultiSelectEnabled(falso)`. Se um tipo de artigo é hierárquico, os valores do atributo dos descendentes são visualizados na tabela com os cabeçalhos de coluna do formulário: o nome do componente descendente/nome do atributo, onde o nome do componente descendente é o nome do componente descendente e o nome do atributo é o nome do atributo do componente descendente. Por exemplo, se um tipo de artigo chamado `Journal` tiver um componente descendente chamado `Author` e o componente descendente `Author` tiver um atributo chamado `Last Name`, o cabeçalho da coluna será: `Author/Last Name`.

Menus emergentes

Surge um menu emergente a disponibilizar opções para Ordenar quando o utilizador faz clique com o botão direito do rato num título da coluna da tabela. Os utilizadores fazem clique sobre **Ordem Ascendente** para ordenar os artigos na tabela por ordem ascendente. Os utilizadores fazem clique sobre **Ordem Descendente** para ordenar os artigos por ordem descendente. Surge outro menu emergente quando um utilizador faz clique com o botão direito do rato sobre uma pasta que não seja a pasta **Resultados da Pesquisa** na área de janela em árvore, ou faz clique com o botão direito do rato sobre um documento ou pasta na área de janela de pormenores. O menu emergente permite aos utilizadores Visualizar detalhes das pastas na área de janela em árvore ou Editar Atributos de pastas.

Opcional: Utilize o `CMBViewFolderEvent` em vez de revelar os pormenores da pasta dentro do bean `CMBSearchResultsViewer`. Utilize o evento para fazer o bean `CMBFolderViewer` apresentar os conteúdos da pasta seleccionada.

Acção de duplo clique

Fazer duplo clique sobre uma pasta numa área de janela em árvore ou num artigo na área de janela de pormenores executa a mesma acção que fazer clique sobre o artigo do menu emergente **Ver**. Se suprimir o menu emergente do artigo predefinido, irá ocorrer um `CMBItemActionEvent`.

Substituir menus emergentes

Pode substituir os menus emergentes em `CMBSearchResultsViewer` e em `CMBFolderViewer` com um menu por um menu emergente diferente ou por nenhum outro. Para desactivar os menus predefinidos, utilize `setDefaultPopupMenu(falso)`.

Ao fazer clique com o botão direito do rato sobre uma pasta na área de janela em árvore, gera-se um `CMBFolderPopupEvent`. Ao fazer clique com o botão direito do rato sobre um artigo na área de janela de pormenores, é gerado um `CMBItemPopupEvent`. Poderá utilizar uma rotina de tratamento para fornecer um menu emergente diferente.

Bean `CMBFolderViewer`

O bean `CMBFolderViewer` apresenta uma área de janela em árvore que se assemelha ao bean `CMBSearchResultsViewer`. Não existe uma pasta principal de **Resultados da Pesquisa**. A Figura 28 mostra as áreas de janela em árvore e de pormenores do bean `CMBFolderViewer`.

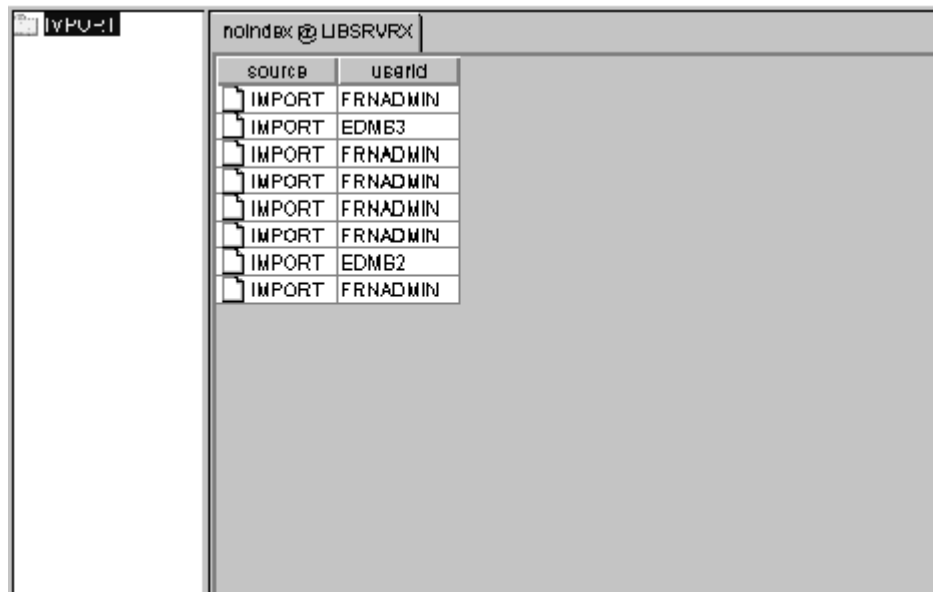


Figura 28. Bean `CMBFolderViewer`

O bean `CMBFolderViewer` apresenta uma árvore de pastas na área de janela à esquerda. A área de janela à direita apresenta um bloco de notas de tabelas dos documentos que se encontram na pasta seleccionada na área de janela em árvore. Um repartidor redimensionável separa as áreas de janela em árvore e de bloco de notas.

Área de janela em árvore de `CMBFolderViewer`

A área de janela em árvore contém pastas. Surgem pastas imbricadas por baixo de cada pasta.

Área de janela Pormenores de `CMBFolderViewer`

A área de janela de pormenores apresenta o conteúdo da pasta que é seleccionada na área de janela em árvore. Os conteúdos são apresentados num bloco de notas com um separador para cada entidade (classe de índice, tipo de artigo ou outro) e servidor sob os quais os artigos da tabela estão indexados. Os nomes dos separadores possuem a seguinte forma:

classe de índice @ servidor em que *classe de índice* corresponde ao nome da classe de índice e *servidor* corresponde ao nome do servidor. Em cada página do bloco de notas está uma tabela que apresenta os documentos e pastas que integram a pasta seleccionada. As colunas da tabela mudam de forma a apresentar os atributos segundo a classe de índices.

Menus emergentes

O comportamento dos menus emergentes no visualizador de pastas é idêntico ao do visualizador de resultados da pesquisa.

Acção de duplo clique

Fazer duplo clique sobre o visualizador da pasta é idêntico a executar a mesma acção no visualizador de resultados da pesquisa.

Bean CMBDocumentViewer

O bean CMBDocumentViewer fornece capacidades de visualização de documentos através da iniciação ou incorporação de visualizadores de documentos específicos de tipo de conteúdo. Existem dois tipos de visualizadores suportados:

1. Visualizadores com base em Java. Estes visualizadores têm de expandir o CMBJavaDocumentViewer da classe.
2. Visualizadores que não sejam de Java. Pode ser iniciado um qualquer executável como visualizador para um tipo de conteúdo em particular.

Caso a propriedade Visível esteja definida como falsa, o visualizador é sempre apresentado numa janela à parte. Se a propriedade Visível for verdadeira, o visualizador, se possível, será apresentado dentro da área de apresentação do bean CMBDocumentViewer. (Actualmente, esta acção é apenas possível para visualizadores com base em Java.)

CMBJavaDocumentViewer é uma classe abstracta expandida por fornecedores de visualizadores de documentos com base em Java que se ligam ao bean CMBDocumentViewer. Estes visualizadores podem apresentar os documentos no espaço visível do bean CMBDocumentViewer ou em janelas à parte no ecrã.

Uma chamada para CMBDocumentViewer terminate() aguarda até que todos os eventos encerrados do documento sejam processados. Se chamar terminate() a partir do operador de eventos encerrado do documento, pode ocorrer um bloqueio e o programa pára. Para evitar este problema, quando chamar terminate() a partir do operador de eventos onDocumentClosed(CMBDocumentClosedEvent), chame o método CMBDocumentViewer.terminate() utilizando SwingUtilities.invokeLater(Runnable). Isto adiciona a chamada de terminate() ao fim da fila de eventos e prossegue com os outros eventos da fila (como, por exemplo, o processamento de outros eventos encerrados do documento) antes e chamar o método de terminação.

Especificações do visualizador

Existem duas formas de especificar visualizadores:

1. Na Administração do EIP, especificar os visualizadores que utilizam o Tipo MIME para o Editor de Associação da Aplicação. Este é seleccionado ao escolher **MIME para Editor Apl.** no menu **Ferramentas**. Para visualizadores

com base em Java, o nome da aplicação deverá ser o nome da classe de Java, incluindo o sufixo `.class`. Para executáveis, o nome da aplicação deverá ser o nome do executável.

2. Utilizar a propriedade `Mime2App` em `CMBDocumentViewer`. Esta propriedade pode ser definida como uma ocorrência de um objecto de Propriedades que define correspondências dos tipos de MIME para nomes de aplicações.

Nos casos em que é especificado um visualizador para um tipo de MIME na Administração do EIP e através da utilização da propriedade de `Mime2App`, terá prioridade a especificação que utilizar `Mime2App`.

Visualizadores predefinidos

Se não for especificado nenhum visualizador para um tipo de conteúdo em particular, será iniciado um visualizador predefinido. Para documentos de `OnDemand`, é iniciado o cliente de `OnDemand` (em modo apenas de visualização). Os documentos de todos os outros servidores de conteúdos serão visualizados através do visualizador do Content Manager. Para editar anotações, seleccione "Editar Documento" no menu "Ficheiro" do visualizador.

Iniciar visualizadores externos

Utilize a propriedade `Mime2App` do `CMBDocumentViewer` para especificar que as aplicações se devem iniciar como visualizadores de documentos para documentos de certos tipos de MIME. Utilize o `setMime2App` com um objecto de propriedades como argumento que tenha nomes de tipos de MIME que definam correspondências de valores que são nomes executáveis.

Bean CMBItemAttributesEditor

O bean `CMBItemAttributesEditor` (consultar Figura 29) apresenta uma janela para visualização e modificação da classe de índices e dos atributos de indexação de uma pasta ou documento.

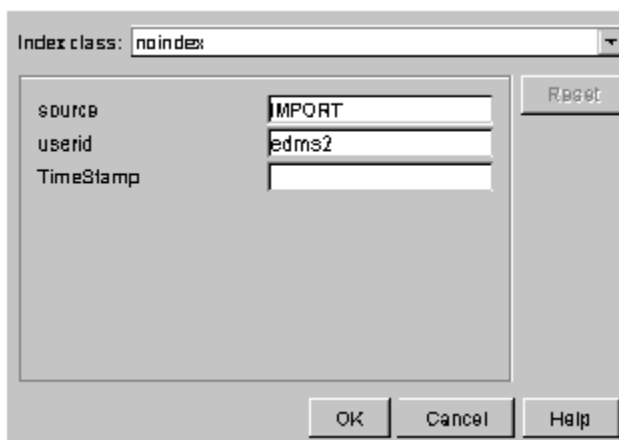


Figura 29. Bean `CMBItemAttributesEditor`

No topo da janela surge uma lista que contém todas as entidades disponíveis. A entidade actual é seleccionada por predefinição. Sob a entidade surge uma lista de atributos para essa entidade. Os campos de texto (nome próprio, apelido, e assim por diante) contém inicialmente os valores actuais para o artigo.

Caso os utilizadores seleccionem uma nova entidade, quaisquer atributos com o mesmo nomes que a entidade seleccionada previamente terão os seus valores propagados para os atributos com nomes semelhantes na nova entidade.

Se fizer clique sobre **Reset** irá devolver os valores originais da entidade e dos atributos.

Ao fazer clique sobre **OK** irá actualizar a entidade e os atributos, activando eventos antes e depois da actualização. Poderá utilizar o evento antes da actualização para validar campos ou completar campos em falta antes da execução da actualização. Este evento pode impedir a actualização especificada.

Proibir alterações no CMBItemAttributesEditor

O utilizador pode fornecer lógica de validação ao CMBItemAttributesEditor, que verifica valores de atributos inseridos pelo utilizador e modifica-os, ou rejeita uma actualização, caso os valores não sejam válidos. Execute esta acção fornecendo um operador ao CMBEditRequestedEvent.

Bean CMBVersionsViewer

O bean CMBVersionsViewer apresenta uma tabela de atributos de atribuição de versão para um documento ou artigo único. Os atributos de elaboração de versões apresentados são: o número da versão, o ID de utilizador do criador, a marca de hora de quando a versão foi criada, o ID de utilizador do maior actualizador e a marca de hora da última actualização. A partir do visualizador das versões, o utilizador pode visualizar as diferentes versões de um artigo ou pode actualizar os atributos de um artigo.

Comportamentos gerais de beans visuais

As secções seguintes descrevem propriedades e comportamentos que são comuns nos beans visuais.

Propriedades

Esta secção descreve três propriedades partilhadas pelos beans visuais.

Ligação

Cada bean tem uma propriedade de ligação, que remete para uma ocorrência do bean não visual CMBConnection. O utilizador terá de definir a propriedade Ligação de forma a que o bean visual opere correctamente.

EficáciaOrdenação

Todos os beans que executam ordenação têm uma propriedade de CollationStrength. Os valores definidos na propriedade de CollationStrength são os mesmos valores definidos para a classe java.text.Collator de Java.

Ocultar/Mostrar botões

Pode ocultar ou apresentar os botões que são apresentados em todos os beans visuais. Utilize a propriedade *setNameButtonVisible*, em que *Nome* corresponde ao nome do botão.

Guardar/restaurar configuração

O CMBSearchTemplateViewer, o CMBSearchResultsViewer e o CMBFolderViewer têm dois métodos - *loadConfiguration* e *saveConfiguration* - que podem ser utilizados para guardar e restaurar valores de campos e tamanhos de colunas entre sessões da aplicação. Um objecto de propriedades é um argumento para todos

estes métodos. Poderá utilizar o mesmo objecto de propriedades para os três beans. Os nomes das propriedades guardadas são únicos em todos os beans.

Eventos de Ajuda

Cada bean visual gera um evento CMBHelp quando o utilizador pede ajuda, quer fazendo clique sobre o botão **Ajuda** ou premindo F1. Alguns beans geram os seguintes eventos relacionados com a ajuda, assim que os utilizadores primem F1 ou Ajuda em janelas secundárias:

CMBChangePasswordHelpEvent

Quando se faz clique sobre **Ajuda** na janela Alterar Palavra Passe

CMBUpdateMappingHelpEvent

Quando se faz clique sobre **Ajuda** na janela Actualizar Definição de Correspondências

CMBLoginFailedHelpEvent

Quando se faz clique sobre **Ajuda** na janela Falhou Início de Sessão do Servidor

CMBServerUnavailableHelpEvent

Quando se faz clique sobre **Ajuda** na janela Servidor Indisponível

Sugestão: Um método possível para funcionar com a ajuda a partir de todas estas origens é criar uma única classe que implemente os receptores de todos estes eventos. No método `onHelp`, poderá ser necessária lógica adicional para determinar qual o bean que originou o evento e apresentar o texto de ajuda apropriado para esse bean.

Substituir um bean visual

É possível substituir um dos beans visuais por outro bean ou pelo componente Swing. Para executar esta acção, o novo bean deverá implementar os operadores para os eventos do bean visual que está a substituir. Também deverá gerar pelo menos os eventos chave do bean que está a substituir. Os eventos-chave estão descritos na Tabela 31.

Tabela 31. Beans visuais e eventos chave

| Bean visual | Eventos chave |
|-----------------------------------|--|
| CMBSearchTemplateList | Evento de CMBTemplateSelectedEvent |
| Visualizador de CMBSearchTemplate | CMBSearchStartedEventCMBSearchResults Event |
| CMBSearchResultsViewer | CMBViewDocumentEvent CMBViewFolderEvent-CMBEditItemAttributesEvent |
| CMBFolderViewer | CMBViewDocumentEvent CMBEditItem AttributesEvent |
| CMBDocumentViewer | CMBDocumentOpenedEvent CMBDocument Closed Event |
| CMBItemAttributesEditor | nenhum |

Todos os dados necessários à implementação da função do bean estão disponíveis nos eventos que o bean está a operar ou no bean não visual CMBConnection.

Construir uma aplicação através de beans visuais

É facultado ao utilizador uma aplicação cliente de exemplo que foi escrita utilizando os beans visuais. Os ficheiros fonte do exemplo encontram-se em:

<cmbrout>/samples/java/beans/gui. O utilizador também deve ler o ficheiro readme.html neste directório p ara obter detalhes sobre os requisitos de cliente exemplo e de configuração.

As secções seguintes demonstram como os beans visuais se ajustam entre si quando o utilizador conceber uma aplicação.

Ligar os beans visuais

Esta secção demonstra uma situação exemplificativa para a ligação de beans visuais, de forma a criar uma aplicação simples. Exceptuando o botão **Pesquisar**, todos os beans são ligados adicionando o bean destino como receptor do evento indicado do bean origem. Por exemplo, para ligar SearchTemplateList a SearchTemplateViewer, só é necessária uma linha de código. Para adicionar um botão para iniciar pesquisas, utilize um JButton padrão. Crie uma classe interna para que o evento de acções do botão invoque o método adequado.

Na Figura 30, as linhas que vão de cada um dos beans para o bean de ligação indicam que o bean contém uma referência ao bean de ligação. Esta situação é criada através da definição da propriedade de ligação para cada bean. Por exemplo, para criar uma referência a partir do bean de painel de início de sessão para o bean de ligação, é necessária uma linha de código.

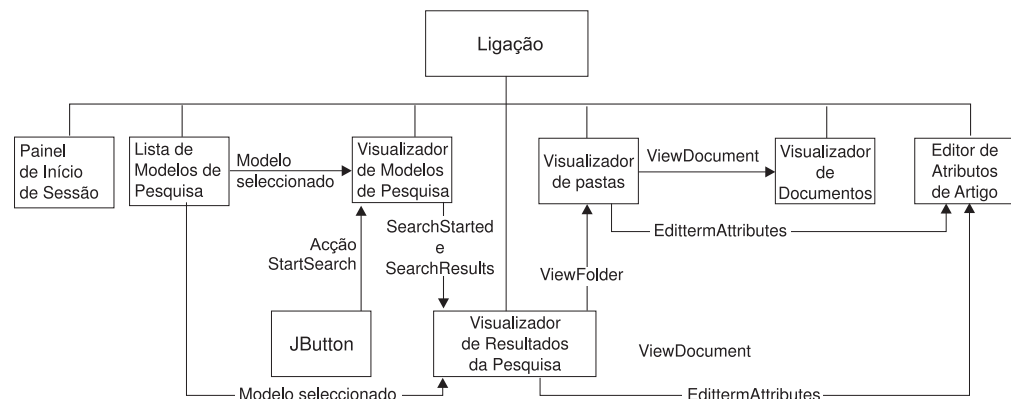


Figura 30. Ligações de bean visual

A Figura 30 mostra nove beans. Um JFrame ou outro bean de conteúdo seria o principal de todos estes beans. Uma ordem possível de eventos durante o tempo de execução poderá ser:

1. O utilizador insere um ID de utilizador e palavra-passe na janela de início de sessão e faz clique sobre **OK**. O bean CMBLogonPanel invoca o método ligar do bean CMBCConnection para estabelecer a ligação ao servidor.
2. O bean de ligação estabelece a ligação. O bean CMBSearchTemplateList obtém e apresenta a lista de modelos de pesquisa para esse ID de utilizador. (Não é necessário invocar métodos para originar esta operação. O bean CMBSearchTemplateList está a receber os eventos apropriados do bean CMBCConnection. CMBSearchTemplateList configura os receptores quando um bean CMBCConnection se associou a ele através do método setConnection.)
3. O utilizador selecciona um modelo de pesquisa da lista. O bean CMBSearchTemplateList gera um CMBTemplateSelectedEvent. Tanto o CMBSearchTemplateViewer como o CMBSearchResultsViewer estão a aguardar o evento. O CMBSearchTemplateViewer apresenta o modelo apropriado. O CMBSearchResultsViewer limpa e apresenta colunas na área de janela de pormenores junto ao modelo.

4. O utilizador conclui o modelo e prime Enter ou faz clique sobre **Pesquisar**. Se o utilizador fizer clique sobre **Pesquisar**, o operador do evento da acção invoca o método `iniciarPesquisa`. Se o utilizador premir Enter, o método `startSearch` é implicitamente invocado.
5. O bean `CMBSearchTemplateViewer` valida os campos modelo para determinar se se pode dar início a uma pesquisa. Caso se possa iniciar a pesquisa, é gerado um `CMBSearchStartedEvent`. O `CMBSearchResultsViewer` aguarda um `CMBSearchStartedEvent` e limpa os resultados como preparação para novos resultados de pesquisa.
6. Enquanto progride a pesquisa, são gerados `CMBSearchResultsEvents` para fornecer resultados de pesquisa parciais ao `CMBSearchResultsViewer`. (Quando a pesquisa estiver concluída é gerado o evento `CMBSearchCompleted`. Este evento pode ser utilizado para activar novamente o botão **Pesquisa**, caso este tenha sido desactivado no início da pesquisa.)
7. O utilizador pode expandir pastas na janela Resultados da pesquisa e em seguida seleccionar um documento ou pasta para visualização. Tendo tomado esta acção, é gerado um `CMBViewFolderEvent` ou um `CMBViewDocumentEvent`. Os beans `CMBFolderViewer` e `CMBDocumentViewer` estão a aguardar os respectivos eventos e apresentam a pasta ou documento.
8. A partir do `CMBFolderViewer`, os utilizadores podem seleccionar um documento para visualização. Ao seleccionar um documento para visualização, irá gerar um `CMBViewDocumentEvent`. O `CMBDocumentViewer` aguarda por este evento e apresenta o documento no visualizador apropriado.
9. Os utilizadores podem seleccionar os atributos de um documento ou de uma pasta para actualização a partir do `CMBSearchResultsViewer` ou do `CMBFolderViewer`. Ao seleccionar um documento irá gerar um `CMBEditItemAttributesEvent`.
10. O bean `CMBItemAttributesEditor` aguarda por um `CMBEditItemAttributesEvent`. Este apresenta a entidade e os atributos para o artigo. O utilizador pode depois alterar a entidade e os atributos, fazendo em seguida clique sobre **OK** para aplicar as alterações.

Utilizar beans em mais do que uma janela ou diálogo

O utilizador terá de fornecer códigos adicionais para passar um evento de um bean numa janela para um bean noutra janela. Normalmente, a constatação de que foi enviado um evento é o motivo para surgir uma janela. A janela `EditAttributesDialog` contém o `ItemAttributesEditor`. `SearchFrame` cria a janela quando é iniciado um `CMBEditItemAttributesEvent`:

```
// Invocar um diálogo secundário para editar atributos
searchResultsViewer.addEditItemAttributesListener(new
CMBEditItemAttributesListener() {
    public void onEditItemAttributes(CMBEditItemAttributesEvent event) {
        EditAttributesDialog editAttributesDialog = new
        EditAttributesDialog(SearchFrame.this,connection,event.getItem());
        editAttributesDialog.setVisible(true);
    }
});
```

As informações que normalmente passam para o bean `CMBItemAttributesEditor`, passam em vez disso como argumentos para o construtor da janela. No construtor, as informações passam para o bean `CMBItemAttributesEditor` através da:

```
itemAttributesEditor.setConnection(connection);

itemAttributesEditor.setItem(item);
```

Trabalhar com o conjunto de ferramentas do visualizador de documentos de Java

Pode utilizar um visualizador de documento para aceder e anotar documentos contidos nos servidores de conteúdos do utilizador. Pode criar um visualizador de documentos personalizado utilizando o conjunto de ferramentas de visualizador de EIP Java. Pode também criar applets e aplicações personalizadas de visualizador para integrar em EIP ou em aplicações autónomas.

Importante: As opções de *dados de visualização* não são suportadas pelo programa emissor de OnDemand.

As classes de conjunto de ferramentas de visualizador Java incluem objectos de acção que facultam as seguintes funções:

- Opções de visualização de página
 - Rotação de documentos: 90 graus no sentido dos ponteiros do relógio, 90 graus no sentido contrário aos ponteiros do relógio e 180 graus
 - Dimensionar: ampliar e reduzir
 - Escala: 25%, 50%, 100%, 150%, 200% e 400%
- Inverter
- Aumentar
- Imprimir
- Fechar o documento actual
- Fechar todos os documentos
- Criar e editar anotações
 - Escrever
 - Realçar
 - Desenhar um quadrado
 - Desenhar um círculo
 - Desenhar uma linha
 - Desenhar uma seta
 - Adicionar texto
 - Senha
 - Adicionar uma nota
 - Apagar
 - Ocultar ou apresentar
 - Modo de curso predefinido
 - Colocar anotação à frente
 - Colocar anotação atrás
 - Alterar propriedades da anotação
 - Anular e voltar a executar operações de anotação
 - Cortar, copiar e colar e eliminar anotações
 - Guardar o documento (apenas as anotações são guardadas)
- Navegar documentos ou páginas dentro de um documento

Navegar páginas: primeira página, página anterior, ir para uma página, página seguinte e última página

Navegar documentos: primeiro documento, documento anterior, ir para um documento, documento seguinte e último documento

- Miniaturas
 - Ocultar ou apresentar miniaturas
 - Navegação em páginas utilizando miniaturas
 - Visualização panorâmica e focagem

O conjunto de ferramentas do visualizador de Java faculta várias classes de GUI que pode utilizar para construir aplicações com base em Swing. Também faculta classes não GUI que pode utilizar em aplicações de visualização de documentos com base em Swing.

Arquitectura do visualizador

O conjunto de ferramentas do visualizador de Java contém um bean de Visualizador de Documentos e de Serviços de Documentos. Os beans facultam um mecanismo para integrar o visualizador em aplicações com base em EIP.

O conjunto de ferramentas também contém as classes Visualizador de Documentos Genérico, Serviços de Documentos de Sequenciação e Serviços de Anotação. As classes permitem-lhe utilizar o visualizador em aplicações em que não pode utilizar os beans como, por exemplo, numa visualização autónoma ou em situações de processo distribuído, em que a ligação a armazenamentos de conteúdos não é local.

Streaming Doc Services gere um conjunto de motores de processamento de documentos que interpretam documentos, transmitem páginas e facultam ao utilizador a capacidade de manipular páginas de documentos. Os motores facultam imagens de documento identificado em vários formatos como, por exemplo, TIFF e IOCA, bem como documentos text e rich text e formatos de office. O utilizador pode também gravar motores de documentos adicionais e ligá-los à arquitectura de conjunto de ferramentas do visualizador para suportar formatos adicionais ou a apresentação alternativa de formatos de documentos.

A classe de Serviços de Anotação permite ao utilizador manipular anotações. O motor de anotações identifica formatos de anotação específicos de Content Manager. Pode gravar motores de anotações adicionais.

Os motores de documentos

São facultados quatro motores de documentos com o EIP:

- Motor de Documentos MS-Tech: Este motor processa tipos de conteúdos normalmente localizados no IBM Content Manager, apresentando páginas como imagens. Os tipos de documentos suportados incluem TIFF, MO:DCA. Este motor também suporta GIF, JPEG e texto normal.
- Motor de Documentos INSO: Este motor suporta Microsoft Office, Lotus SmartSuite e outros formatos de documentos de office.
- Motor de Documentos AFP2Web: Este motor compreende AFP e converte os documentos de AFP para HTML ou PDF.
- Motor de Documentos Java: Este motor converte documentos que são URLs de HTML com uma ligação de reenvio ao URL. Este motor também converte XML em HTML invocando XSLT.

Os motores são interfaces públicas, mas o utilizador não deve efectuar programações directamente nas interfaces. Deve utilizar as interfaces facultadas pelo bean Serviços de Documentos ou a classe Serviços de Documentos de Sequenciação.

Alguns destes motores não são apenas de Java e possuem limitações de portabilidade. Isto poderá restringir a utilização do conjunto de ferramentas em algumas plataformas. Os motores MS-Tech e Java são apenas de Java. Os outros motores contêm uma lógica específica de plataforma que restringe a sua utilização à plataforma Windows.

O motor de anotações

EIP faculta o motor de anotação MS-Tech para operar anotações de Content Manager. O motor MS-Tech suporta anotações do Content Manager Versão 8.2, Content Manager Versão 8.1, Content Manager Versão 7 e VI/400.

Criar um visualizador de documentos genérico

Para criar um visualizador de documentos genérico, trabalha principalmente com a classe CMBGenericDocViewer, que utiliza a interface CMBStreamingDocServices. CMBStreamingDocServices carrega e transmite documentos. CMBStreamingDocServices utiliza um conjunto de motores de documento para converter diferentes formatos de documentos, tais como TIFF e MO:DCA. Para carregar, editar e guardar anotações em documentos, utilize a interface CMBAnnotationServices.

Personalizar o visualizador genérico de documentos

Pode personalizar o ficheiro de configuração predefinido, CMBViewerConfiguration.properties, localizado no ficheiro cmbview81.jar, ou pode criar um novo ficheiro de configuração. Quer crie um ficheiro de configuração ou personalize o ficheiro CMBViewerConfiguration.properties, deve manter o mesmo nome de ficheiro e colocá-lo antes de cmbview81.jar no caminho de classe.

Para criar um ficheiro de configuração, conclua os seguintes passos:

1. Deve especificar a seguinte entrada para cada barra de ferramentas especificada pelo nome de barra de ferramentas. Este passo especifica a posição da barra de ferramentas no sistema principal. A posição predefinida é NORTH. Especifique a posição de cada barra de ferramentas que for enumerada.

```
Toolbars=[<toolbar_name>[,<toolbar2_name>][,<toolbar3_name>]...]
<toolbar_name>.position={NORTH|SOUTH|EAST|WEST}
```

2. Especifique as acções que serão adicionadas à barra de ferramentas especificada. Utilize a palavra 'separador' para inserir um separador entre as acções na barra de ferramentas.

```
<toolbar_name>.tools=[<action_name>[,<action2_name>]
[,<action3_name>]...]
```

```
<action_name>.label=<action_label>
<action_name>.tooltip=<action_tooltip>
<action_name>.icon=<icon_file_name>
<action_name>.key=<key_code>
<action_name>.cursor=<cursor_file>
<action_name>.hotspot=<x,y>
```

3. Não pode adicionar novos menus emergentes. No entanto, pode adicionar sub-menus e artigos de menu aos três menus emergentes predefinidos.

```

<popup_menu_name>.items=[<menuitem_name>[,<menuitem2_name>]
[,<menuitem3_name>]...]
<popup_menu_name>.submenu=[<submenu_name>[,<submenu2_name>]
[,<submenu3_name>]...]
<submenu_name>.label=<submenu_label>

```

Caso tenha especificado o ficheiro de configuração adequado, está pronto a construir uma aplicação visualizadora ou applet autónomas. Consulte *Consulte Online de API* à medida que conclui os seguintes passos:

1. Crie uma classe privada que implemente `CMBStreamingDocServicesCallbacks`.
2. Crie `CMBStreamingDocServices` com as marcações após verificação do ID chamador implementadas no primeiro passo.
3. Crie uma classe privada que implemente `CMBAnnotationServicesCallbacks`.
4. Crie `CMBAnnotationServices` com as marcações após verificação do ID chamador implementadas.
5. Crie uma ocorrência de `CMBGenericDocViewer` e inicialize com `CMBStreamingDocServices`, `CMBAnnotationServices` e o ficheiro de propriedades de configuração. Transmita nulo para o ficheiro de propriedades, caso pretenda utilizar o ficheiro de configuração predefinido.
6. Chame `loadDocument()` em `CMBGenericDocViewer` para carregar um documento. Esta acção devolve uma ocorrência de `CMBDocument`.
7. Chame `loadAnnotationSet()` em `CMBGenericDocViewer` para carregar anotações. É devolvido um objecto de `CMBAnnotationSet`. Utilize o método `setItemHandle(CMBAnnotationSet, CMBItem)` em `CMBAnnotationServices`.
8. Personalize a aparência e comportamento do visualizador utilizando diferentes métodos facultados pelo visualizador para a posição, tamanho das descrições resumidas, vista MDI ou SDI, etc.
9. Adicione o visualizador à estrutura principal da aplicação ou da applet.
10. Prepare a barra do menu obtendo as acções do visualizador de documentos genérico e adicione o menu à estrutura principal das aplicações.
11. Chame `showDocument()` no visualizador de documentos genérico para visualizar o documento.
12. Chame `saveAnnotations(CMBDocument)` em `CMBGenericDocViewer` para guardar anotações no documento.
13. Chame `closeDocument(CMBDocument)` para fechar um documento ou chame `closeAllDocuments()` para fechar todos os documentos.

A Figura 31 na página 435 é um exemplo de um visualizador de documentos genérico que utiliza todas as definições predefinidas.



Figura 31. Visualizador de documentos genérico

Aplicações exemplo

Para o ajudar a compreender o conjunto de ferramentas do Visualizador de Documentos Java, esta secção faculta cinco aplicações exemplo que pode criar. Os exemplos seguintes não constituem as únicas formas de criação do conjunto de ferramentas.

Visualizador autónomo

Pode utilizar o Visualizador de Documentos Genérico para implementar um visualizador autónomo. Pode utilizar um visualizador autónomo para visualizar ficheiros ou documentos que obtiver de URLs. Pode utilizar os visualizadores autónomos como paletes em páginas da web ou, por exemplo, para visualizar documentos obtidos através de correio electrónico. A Figura 32 na página 436 ilustra a arquitectura de um visualizador autónomo.

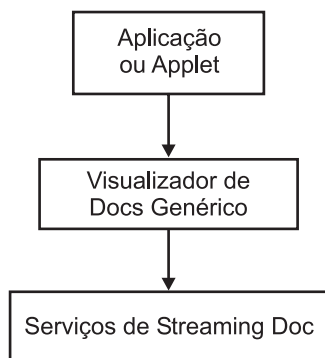


Figura 32. Visualizador autônomo

Aplicação Java

Para criar uma aplicação Java de produção, utilize os beans visuais de EIP, que utilizam o bean visual de CMBDocumentViewer. Este bean utiliza o Visualizador de Documentos Genérico internamente para visualizar documentos. O bean de CMBDocumentViewer pode também iniciar outros visualizadores para ver documentos. No entanto, o utilizador deve ter em atenção que estes visualizadores poderão ter dependências de plataformas. A Figura 33 ilustra a arquitectura que pode utilizar para construir uma aplicação Java.

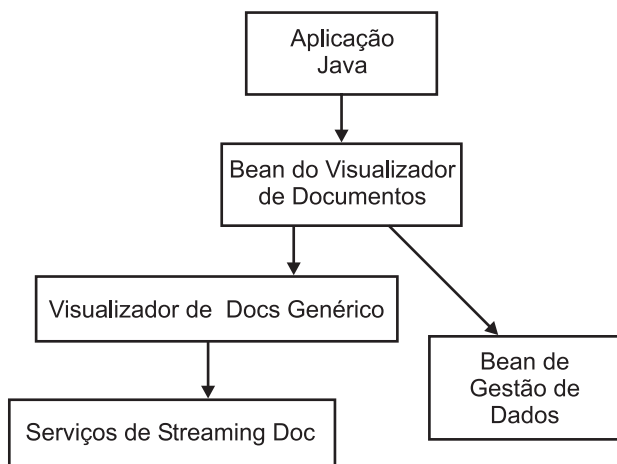


Figura 33. Aplicação Java

Cliente com poucos recursos

Pode utilizar o bean CMBDocumentServices para executar conversões de documentos do lado do servidor numa aplicação baseada na web. Pode converter documentos de tipos de conteúdo que não são tratados pelo browser (documentos que necessitam de um plugin ou de um início de aplicação nativa) para tipos de conteúdo tratados nativamente pelo browser como, por exemplo, HTML, GIF, JPEG ou para os quais os plugins estão prontamente disponíveis como, por exemplo, PDF. A Figura 34 na página 437 ilustra a arquitectura de um cliente de poucos recursos.

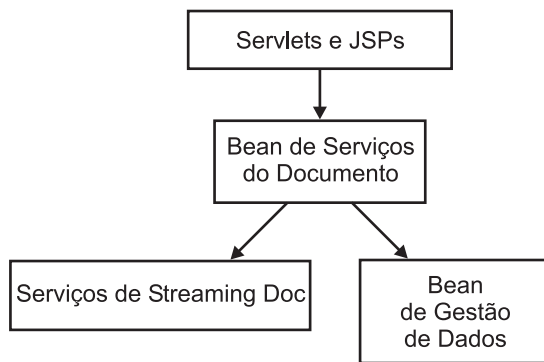


Figura 34. Cliente com poucos recursos

Applet ou servlet

Uma aplicação baseada na web também pode facultar capacidades de visualização de documentos e de edição de anotação utilizando uma applet ou uma abordagem de servlet. A applet do visualizador eClient utiliza esta arquitectura. A applet pode utilizar o Visualizador de Documentos Genérico para visualizar o documento. Alguns tipos de documentos são armazenados em várias partes do servidor de conteúdos para tornar mais eficaz a partilha da maioria das informações, tais como formulários de segundo plano. As partes adicionais são solicitadas pelo Visualizador de Documentos Genérico quando for necessário. A applet que contém o visualizador satisfaz os pedidos enviando pedidos de HTTP para o servlet. No que diz respeito ao servlet, o servidor de conteúdos que utiliza o bean de CMBDataManagement obtém as informações solicitadas. A Figura 35 ilustra uma arquitectura de applet ou servlet.

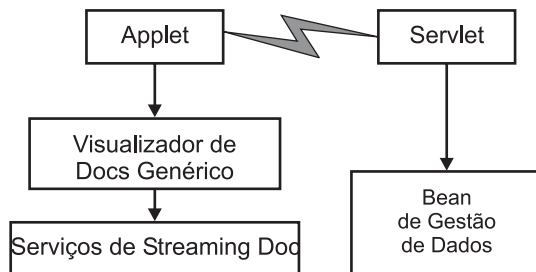


Figura 35. Applet ou servlet

Modo dual e applet ou servlet

Pode utilizar uma variação dos exemplos facultados para aplicações de visualização com base na web. Utilize CMBDocumentServices no servidor e na applet. Esta abordagem é útil quando os documentos não são apresentados na applet, mas são convertidos no servidor. Isto poderá suceder quando os motores de documentos subjacentes da applet e do servidor possuem capacidades diferentes.

Por exemplo, se o servidor for Windows NT ou 2000 e a applet estiver em execução em OS/2, a applet poderá não conseguir apresentar todos os tipos de documentos. A applet apresenta formatos de documentos que suporta, tais como TIFF. Para tipos que a applet não consegue apresentar, tais como formatos de Office, esta solicita conversão do servlet. Segundo a perspectiva do utilizador da aplicação, a mesma interface é apresentada com a mesma funcionalidade. No

entanto, o rendimento do servidor em relação a documentos convertidos compatíveis com o servidor poderá ser mais lento do que para documentos apresentados localmente.

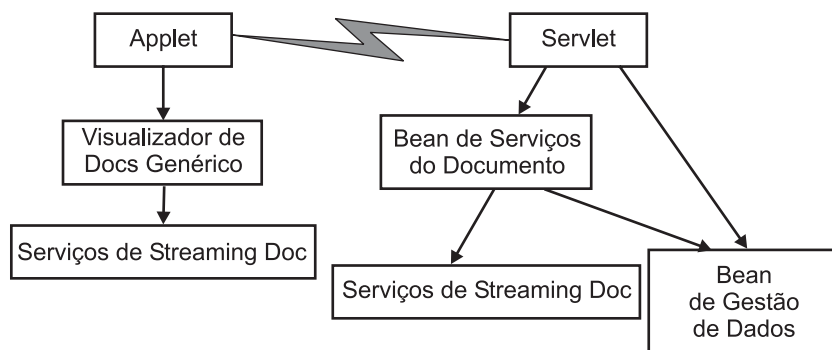


Figura 36. Visualizador de modo dual

Trabalhar com serviços de anotação

O conjunto de ferramentas do visualizador Enterprise Information Portal 8.1 Java faculta capacidades para produzir e converter anotações de documentos. Semelhantes aos serviços de documentos, os motores de anotação ligáveis facultam funções adicionais que pode utilizar nas suas aplicações para interpretar diferentes tipos de anotações. A Figura 37 na página 439 ilustra como o visualizador genérico de documentos e os serviços de documentos e anotação se encaixam.

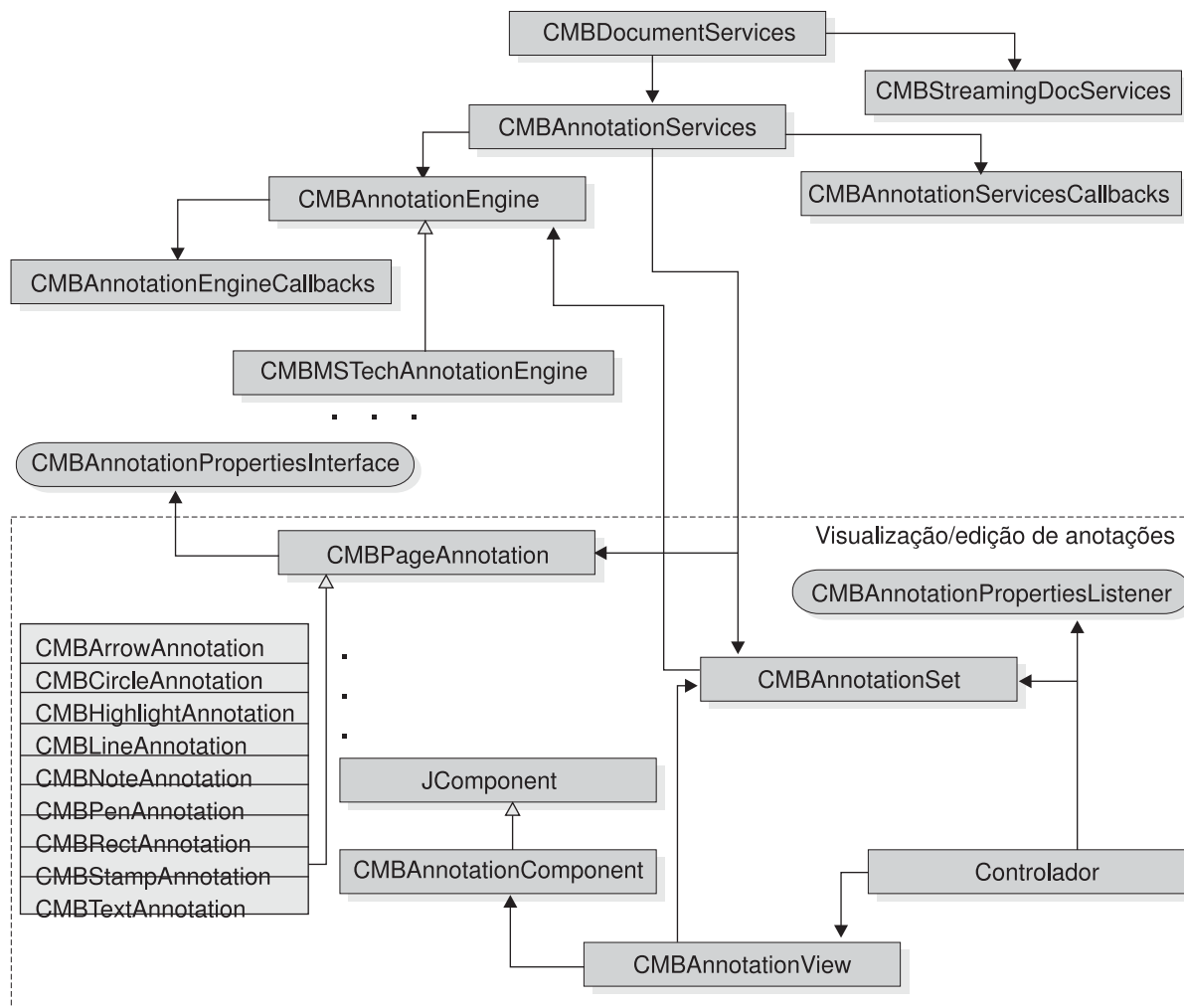


Figura 37. Associação entre os serviços de anotação e o visualizador genérico de documentos

Utilizar interfaces de serviços de anotação

CMBAnnotationServices faculta as principais interfaces utilizadas no conjunto de ferramentas de Java. Os serviços de anotação permitem ao utilizador carregar, manipular e guardar objectos de anotação utilizando os serviços de anotação, independentemente dos programas emissores. Para trabalhar com anotações, o utilizador deve transmitir os dados de anotação como uma sequência e ligar um motor de anotação adequado, que converte os objectos de anotação a ocorrências CMBPageAnnotation. Então, o utilizador poderá manipular e editar as anotações e guardá-las no programa emissor original, no seu formato original.

A Figura 38 na página 440 mostra o diagrama de classe de serviços de anotação.

Para implementar o motor de anotação, o utilizador deve expandir a classe abstracta CMBAnnotationEngine. O motor de anotação utiliza as interfaces CMBAnnotationServicesCallbacks e CMBAnnotationEngineCallbacks para comunicar com uma aplicação e com serviços de anotação. O motor de anotação em EIP 8.1 só reconhece o formato de anotação do Content Manager. Este formato

de anotação é utilizado pelos programas emissores do Content Manager Versão 7, Content Manager Versão 8. 1 e VI/400.

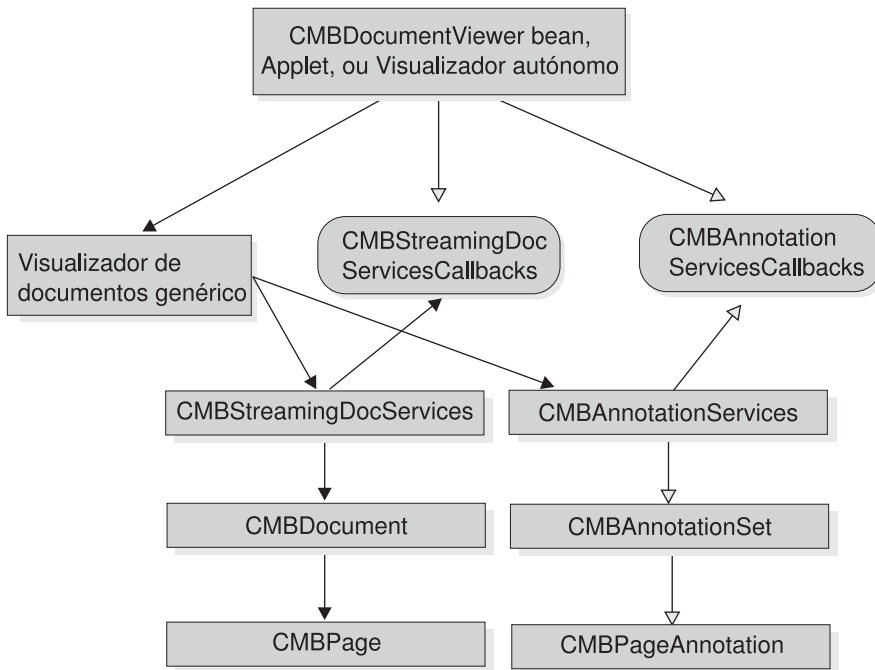


Figura 38. Diagrama de classe de serviços de anotação

Compreender o suporte de edição de anotações

O padrão de concepção do Model View Controller (MVC) é utilizado para implementar a funcionalidade de edição das anotações. MVC funciona como o modelo que representa os dados de anotação. CMBAnnotationSet possui métodos que funcionam sobre os dados, mas não possui interfaces do utilizador. A classe CMBAnnotationSet mantém a lista de objectos CMBPageAnnotation. Cada documento é associado a um objecto CMBAnnotationSet que representa as suas anotações. CMBAnnotationView funciona como a vista que apresenta os dados do modelo para o utilizador. CMBAnnotation processa toda a elaboração de anotações no componente de vista (um JComponent). CMBAnnotationComponent é uma classe auxiliar que pode ser utilizada como componente de vista no qual as anotações são elaboradas. O controlador faz parte do conjunto de ferramentas do visualizador e opera eventos do rato e do teclado para criação e edição de anotações.

CMBPageAnnotation é a classe de base que descreve uma única anotação numa página de um documento. Se necessitar de definir tipos adicionais de anotações gráficas, deve expandir a classe CMBPageAnnotation. Existem nove tipos de anotações de Content Manager que pode criar: CMBArrowAnnotation, CMBCircleAnnotation, CMBHighlightAnnotation, CMBLineAnnotation, CMBNoteAnnotation, CMBPenAnnotation, CMBRectAnnotation, CMBStampAnnotation e CMBTextAnnotation.

Nota: Pode utilizar serviços de anotação em aplicações que não são GUI.

Construir uma aplicação utilizando os serviços de anotação

Esta secção inclui os passos a seguir e as APIs a utilizar para construir uma aplicação de serviços de anotação. Recorra à consulta online de API para obter detalhes relativos à utilização da API.

1. Crie uma sub-classe de `CMBAnnotationServicesCallbacks` para implementar os métodos abstractos de processamento de chamadas de anotações.
2. Crie uma ocorrência de `CMBAnnotationServices`.

```
CMBAnnotationServices annoServices = new  
CMBAnnotationServices(annoServicesCallbacks);
```
3. Obtenha uma ocorrência de `CMBAnnotationSet` carregando uma sequência de anotação.

```
CMBAnnotationSet annotationSet = annoServices.loadAnnotationSet(annoStream,  
format, documentResolution, annotationPartNumber );
```
4. Prepare a vista da anotação.

```
CMBAnnotationComponent annoComponent = new CMBAnnotationComponent();  
CMBAnnotationViewer annoView = annoServices.prepareAnnotationView(  
annoComponent, annotationSet );  
annoView.refreshEntireDrawingArea();
```
5. Adicione o componente de anotação a um contentor Swing como, por exemplo, `Jframe` ou `Jpanel`, da aplicação.
6. Agora poderá utilizar a API de serviços de anotação para adicionar novas anotações, editar ou eliminar anotações já existentes.

```
annoServices.prepareToAddAnnotation();  
annoServices.addAnnotation()  
annoServices.removeAnnotation();  
annoServices.reorderAnnotation();  
...
```
7. Guarde as anotações modificadas.

```
annoServices.saveAnnotationset( annotationSet );
```

Um exemplo de Serviços de Anotação é facultado no directório
<CMBROOT>\Samples\java\viewer directory(TAnnotationEditor.java).

Adicionar um tipo de anotação personalizado à aplicação do utilizador

Para adicionar um tipo de anotação personalizada ao serviços de anotação, conclua os seguintes passos. Recorra à consulta online de API para obter detalhes relativos à utilização da API.

1. Crie uma sub-classe de `CMBPageAnnotation`. A classe pública `TImageAnnotation` expande `CMBPageAnnotation`
2. Defina uma constante para a anotação personalizada com um valor superior a cem. O limite entre 1 e 99 é reservado.

```
public static final int ANN_IMAGE = 101;
```
3. Sobreponha os seguintes métodos de `CMBPageAnnotation`:

```
public void draw(Graphics2D g2);  
public void drawOutline(Graphics2D g2);  
public CMBPropertiesPanel getAnnotationPropertiesPanel();
```
4. Implemente a interface `CMBAnnotationPropertiesInterface` para criar o painel de propriedades. O painel de propriedades surge quando um utilizador opta por editar as propriedades da anotação personalizada.
5. Faculte os métodos de definição e obtenção específicos às propriedades da anotação personalizada.

6. Adicione o tipo de anotação personalizada.

```
annoServices.prepareToAddAnnotation(TImageAnnotation.ANN_IMAGE,  
"TImageAnnotation",1);
```

O exemplo de tipo de anotação `TImageAnnotation` é incluído no directório `<CMBROOT>\Samples\java\viewer`. O exemplo demonstra como adicionar um tipo de anotação personalizada aos serviços de anotação.

Trabalhar com a biblioteca de identificadores e o servlet controlador do Enterprise Information Portal

O Enterprise Information Portal inclui uma biblioteca de identificadores de Java Server Pages e uma servlet que pode utilizar ao escrever JSPs ou servlets para aplicações da Web. Utilizar a biblioteca de identificadores reduz a necessidade de scriptlets Java em JSP escritas em JavaBeans de EIP.

Esta biblioteca de identificadores funciona como servlet que pode funcionar como o controlador de uma aplicação da Web de design controladora de vista de modelo e executar a inicialização de beans e outras acções.

Configurar a biblioteca do identificador e o servlet

Deve instalar a biblioteca de identificadores e o servlet num servidor da Web com o IBM WebSphere Application Server e configurar o servidor da Web para que este os utilize. Para obter informações relativas à instalação da biblioteca de identificadores e do servlet, da sua configuração e para obter informações relativas à construção de ficheiros WAR/EAR, consulte *Planear e Instalar o Enterprise Information Portal*.

Utilizar a biblioteca de identificadores

O seguinte exemplo de JSP mostra como utilizar o identificador de modelos de pesquisa para obter uma lista de modelos de pesquisa:

```
<%@ taglib uri="cmb" prefix="cmb" %>
<%@ page import="com.ibm.mm.beans.*" %>
<jsp:useBean id="connection" scope="session"
              class="com.ibm.mm.beans.CMBConnection" />

<%
    CMBSchemaManagement schema = connection.getSchemaManagement();
    CMBSearchTemplate[] searchtemplates = schema.getSearchTemplate();
    request.setAttribute("searchtemplates", searchtemplates);
%>
<html>
<head>
<title>Search Templates Tag Test</title>
</head>

<body bgcolor="white">
<table border=2 cellpadding=3 cellspacing=3>
<tr>
<td><b>Modelos de pesquisa Disponíveis</b></td>
</tr>
<tr>
<td><%= searchtemplate.getName() %></td>
</tr>
</table>
</body>
</html>
```

A directiva `taglib` declara que a página utiliza a biblioteca de identificadores de EIP e `lhes` associa o prefixo `cmb`. Em seguida é chamado o identificador `searchtemplates` e o método `getName()` devolve o nome de cada exemplo.

Convenções utilizadas na biblioteca do identificador

Os identificadores JSP têm atributos para especificar os beans que utilizam ou geram. Quando esses atributos não são especificados, são utilizados os valores predefinidos. Estes parâmetros são opcionais. Os seus valores são retirados a partir de variáveis locais e os atributos no intervalo do pedido e da sessão. Quando utilizados em conjunto com o toolkit do servlet, as variáveis locais, os atributos de pedido ou os atributos de sessão contêm as predefinições apropriadas. A Tabela 32 mostra os bens predefinidos e o âmbito no qual são assumidos ou colocados. Estas convenções são também seguidas pelo servlet; siga estas convenções noutros servlets que gravar para trabalhar com a biblioteca de identificadores.

Tabela 32. Convenções da biblioteca de identificadores

| Âmbito | Nome | Tipo | Descrição |
|-----------|------------------|---------------------|---|
| aplicação | conjuntoLigações | CMBConnectionPool | O bean do conjunto de ligações que é partilhado através das sessões |
| sessão | ligações | CMBConnection | A ocorrência de CMBConnection para a sessão |
| sessão | esquema | CMBSchemaManagement | bean da gestão do esquema |
| sessão | dados | CMBDataManagement | Bean da gestão do esquema |
| sessão | utilizador | CMBUserManagement | Bean da gestão do utilizador |
| sessão | consulta | CMBQuesryService | Bean do serviço da consulta |
| sessão | traceLog | CMBTraceLog | bean de registo de rastreio; todos os outros beans enviam o seu rastreio para este bean |
| sessão | docservices | CMBDocumentServices | Bean de serviços de documentos |
| pedido | artigo | CMBItem | O último artigo no qual o utilizador funcionou |
| pedido | artigos | CMBItem[] | Recolha dos artigos nos quais o utilizador funcionou pela última vez |
| pedido | searchTemplate | CMBSearchTemplate | O modelo de pesquisa seleccionado |
| pedido | searchResutls | CMBSearchResults | Os resultados da última pesquisa |

Resumo do identificador

As seguintes secções resumem a biblioteca de identificadores:

Identificadores relacionados com ligações

`<cmb:datasources" connection="connection">datasource ... </cmb:datasources>`

Este identificador itera através das origens de dados disponíveis.

connection

Especificar o nome de uma variável do tipo CMBConnection que contém a ligação.

datasource

Uma variável de cadeia que contém o nome da origem de dados como uma cadeia.

Identificadores relacionados com esquemas

<cmb:searchtemplates searchTemplates="searchTemplate"> ...

</cmb:searchtemplates>

Este identificador itera através dos modelos de pesquisa disponíveis.

searchTemplates

Especifique o nome de um conjunto do tipo CMBSearchTemplate[] para conter os modelos de pesquisa.

searchTemplate

Uma variável do tipo CMBSearchTemplate para conter o modelo de pesquisa.

<cmb:searchcriteria searchTemplate="searchtemplate">criteria ...

</cmb:searchcriteria>

Este identificador itera através do critério de pesquisa de um modelo de pesquisa.

searchTemplate

Especificar o nome do modelo de pesquisa.

criteria

Uma variável de tipo CMBSTCriteria para conter o critério de pesquisa.

<cmb:displaycriteria searchTemplate="searchTemplate">criteria ...

</cmb:displaycriteria>

Este identificador itera através do critério de pesquisa que pode ser apresentado para um modelo de pesquisa.

searchTemplate

Especificar o nome do modelo de pesquisa.

criteria

Uma variável de tipo CMBSTCriteria para conter o critério de pesquisa.

<cmb:allowedoperators criteria="criteria">operator ... </cmb:allowedoperators>

Este identificador itera através dos operadores permitidos para um critério de pesquisa.

criteria

Especificar o nome de uma variável de tipo CMBSTCriteria para conter o critério de pesquisa.

operator

Uma cadeia para conter o valor do operador.

<cmb:predefinedvalues criteria="criteria"> value... </cmb:predefinedvalues>

Este identificador itera através dos valores predefinidos de um critério de pesquisa.

criteria

Especificar o nome de uma variável de tipo CMBSTCriteria para conter o critério de pesquisa.

valor

Uma cadeia para conter o valor predefinido do critério de pesquisa.

<cmb:entities schema="schema">entity ... </cmb:entities>

Este identificador itera através das entidades associadas disponíveis.

schema Especificar o nome de uma variável de tipo
CMBSchemaManagement contendo o esquema.

entity Uma cadeia para conter o nome da entidade.

<cmb:attributes entity="entity" schema="schema">attribute ... </cmb:attributes>

Este identificador itera através dos atributos associados de uma entidade associada.

schema Especificar o nome de uma variável de tipo
CMBSchemaManagement para conter o esquema.

entity Especificar o nome da entidade.

attribute

Uma cadeia para conter o nome de uma variável de tipo CMBAttribute que contém o atributo.

Identificadores relacionados com pesquisa

<cmb:searchresults searchresults="searchResults">item ... </cmb:searchresults>

Este identificador itera através dos resultados da pesquisa.

searchResults

Especificar o nome de uma variável de tipo CMBSearchResults que contém os resultados da pesquisa.

artigo Uma cadeia para conter o nome de uma variável de tipo CMBItem para conter o artigo resultante da pesquisa.

Identificadores relacionados com artigos

<cmb:itemattributes item="item"> attrname ... attrtype... attrvalue... </cmb:itemattributes>

Este identificador itera através dos atributos de um artigo.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

attrname

Uma variável de cadeia para conter o nome do atributo.

attrtype

Uma variável de cadeia para conter o tipo do atributo.

attrvalue

Uma variável de cadeia para conter o valor do atributo.

<cmb:itemcontents " data="data" item="item"> content... </cmb:itemcontents>

Este identificador itera através do conteúdo de um artigo.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

conteúdo

Uma variável de tipo CMBObject para os conteúdos do artigo.

<cmb:itemnotelogs " data="data" item="item">notelog ... </cmb:itemnotelogs>

Este identificador itera através dos registos de nota de um artigo.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

registro de notas

Uma variável de tipo CMBObject para conter o registro de nota do artigo.

<cmb:itemprivileges data="data" item="item">privilege ... </cmb:itemprivileges>
Este identificador itera através dos privilégios de um artigo.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

privilégio

Uma variável de tipo CMBPrivilege para conter o privilégio do artigo.

<cmb:itemresources data="datamanagement" item="item"> resource... </cmb:itemresources>
Este identificador itera através dos recursos de um artigo.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

recurso

Uma variável de tipo CMBResources para conter o recurso do artigo.

<cmb:unmappeditem data="data" item="item">unmappeditem...</cmb:unmappeditem>
Este identificador devolve um artigo sem definição de correspondências de um dado artigo com definição de correspondências.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável de tipo CMBItem que contém o artigo correlacionado.

unmappeditem

Uma variável de tipo CMBItem para conter o artigo não correlacionado.

<cmb:viewdata data="data" item="item">viewdata... </cmb:viewdata>
Este identificador devolve uma vista de um artigo.

dados Especificar o nome de uma variável de tipo CMBDataManagement.

artigo Especificar o nome de uma variável do tipo CMBItem que contém o artigo.

viewdata

Uma variável do tipo CMBViewData para conter dados visualizáveis.

Identificadores relacionados com pastas

<cmb:folderitems folder="folder"> item... </cmb:folderitems>
Este identificador itera através do conteúdo de uma pasta.

folder Especificar o nome de uma variável de tipo CMBItem para conter os conteúdos da pasta.

artigo Uma variável de tipo CMBItem que representa a pasta.

Identificadores relacionados com documentos

<cmb:viewerdocuments docservices="docservices">document ...
</cmb:viewerdocuments>

Este identificador itera através de documentos que estão actualmente a ser carregados.

docservices

Especificar o nome de uma variável de tipo CMBDocumentServices.

document

Uma variável de tipo CMBDocument para conter o documento.

<cmb:documentpages document="document"> docpage... </cmb:documentpages>

Este identificador itera através das páginas de um documento.

document

Especificar o nome de uma variável de tipo CMBDocument para conter o documento.

docPage

Uma variável de tipo CMBPage para conter a página.

Servlet controlador do EIP

O EIP faculta um servlet com acções ligáveis que podem ser utilizadas ao construir aplicações da Web. Este servlet funciona como um controlador de uma aplicação da Web concebida através de um controlador de visualização de modelos, executando acções e iniciando os beans (o modelo) que são depois acedidos em JSPs (as visualizações), directa ou indirectamente, através da utilização de identificadores de JSP.

As acções são fornecidas para tarefas típicas de aplicações:

- Iniciar e terminar uma sessão.
- Pesquisar.
- Criar, obter, modificar e eliminar documentos.
- Criar pastas e adicionar ou remover documentos de pastas.
- Iniciar documentos e páginas de documentos para visualização.

Além disso, o servlet executa tarefas comuns antes e depois da acção, como por exemplo a gestão da ligação ao servidor de conteúdos. Após cada acção, é invocado um JSP para formatar os resultados e enviá-los de volta para o browser.

Pode personalizar o servlet para adicionar novas acções e associar JSPs às acções.

O que o servlet pode fazer

Estes são alguns aspectos do servlet que pode utilizar:

Agrupamento de ligações

O servlet controlador utiliza o agrupamento de ligações de EIP para facultar uma gestão de ligações de elevado rendimento. O período de

atribuição de uma ligação a uma sessão pode ser relativo ao pedido ou ao tempo durante o qual a sessão está iniciada. Actualmente, o agrupamento de ligações encontra-se no âmbito da aplicação.

Início de Sessões de Tempo de Espera Excedido

Se uma sessão exceder o tempo de espera e um pedido chegar ao servlet, é apresentada a JSP de início de sessão, permitindo ao utilizador iniciar novamente uma sessão. O pedido original é executado após um início de sessão bem sucedido.

Limpar um fim de sessão

O servlet limpa totalmente a sessão quando esta é terminada, seja através da conclusão da sessão ou através de um tempo de espera esgotado. Isto significa que a sessão é anulada ou devolvida ao conjunto. Todos os outros beans de EIP criados pelo servlet são terminados e os seus recursos são libertados sem aguardar a execução de um ciclo de recolha de dados desnecessários.

Locale O servlet garante a definição correcta do locale nos beans subjacentes, o que significa que as cadeias de mensagens e caracteres são sensíveis a locale.

Utilizar diferentes definições de JSP

Um ficheiro de propriedades, denominado `cmbervlet.jsp.properties`, por predefinição, descreve as JSPs a utilizar para respostas a acções do servlet. A localização do ficheiro de propriedades é um parâmetro da aplicação. Consequentemente, várias aplicações diferentes da web poderiam ser escritas utilizando conjuntos diferentes de JSPs.

Expandir o servlet

Todas as acções conhecidas do servlet estão definidas num ficheiro de propriedades chamado `cmbervlet.properties` (predefinido). Pode adicionar, modificar ou eliminar acções do servlet alterando este ficheiro. Para adicionar uma nova acção, siga estes passos:

1. Implemente uma classe para executar a acção. A classe deve expandir `com.ibm.mm.servlets.CMBServletAction`.
2. Adicione o nome da classe e o nome da acção ao ficheiro `cmbervlet.properties`. Este possui a seguinte sintaxe:

```
actions = lista das acçõesaction.<action_name>.class = class_name
```

`actions` lista as acções compreendidas pelo servlet. Para cada acção, uma linha dentro do ficheiro de propriedades define a classe da acção. Por exemplo, para adicionar uma acção denominada reprodução, numa classe denominada `ReplayAction`:

```
actions = ... reprodução
action.replay.class = ReplayAction
```

Pode também substituir uma acção ou facultar uma acção do utilizador para preceder ou suceder quaisquer acções predefinidas. Por exemplo, para preceder o início de sessão com uma acção do utilizador e para executar uma validação adicional:

```
action.logon.class = MyLogonAction com.ibm.mm.servlets.CMBLogonAction
```

A convenção de nomenclatura utilizada para todas as acções predefinidas é `com.ibm.mm.servlets.CMBactionAction`, em que *action* é o nome da acção, em que a primeira letra é maiúscula.

Referência de servlet

Um conjunto de parâmetros de aplicação, parâmetros de pedido e um ficheiro de propriedades servem para utilizar o servlet controlador nas aplicações do utilizador.

Convenções

O servlet define a sessão e valores de pedido seguintes, que podem ser utilizados noutras JSPs ou servlets. Estas convenções são seguidas pela biblioteca e identificadores de JSP. Estas convenções são iguais às da biblioteca de identificadores de EIP.

Parâmetros de aplicação

O servlet compreende os seguintes parâmetros de aplicação (uma alternativa será colocar estes parâmetros no ficheiro `cmbervlet.properties`).

| Parâmetro da aplicação | Valores | Descrição |
|-----------------------------------|---|---|
| <code>servletPropertiesURL</code> | URL | A localização do ficheiro <code>cmbervlet.properties</code> |
| <code>defaultServerType</code> | Associado, ICM , OD, DL, DES, V4, IP, DD, ... | Informações de início de sessão predefinido. Estas, juntamente com <code>defaultServer</code> , <code>defaultUserid</code> e <code>defaultPassword</code> podem ser utilizadas em situações de ID de utilizador partilhado. Em vez de pedir informações com uma página de início de sessão, as informações de início de sessão predefinido serão utilizadas para executar o início de sessão. |
| <code>defaultServer</code> | | Informações de início de sessão predefinido. |
| <code>defaultUserid</code> | | Informações de início de sessão predefinido. |
| <code>defaultPassword</code> | | Informações de início de sessão predefinido. |
| <code>connectionpool</code> | Booleano: verdadeiro falso | Para activar o agrupamento de ligações |
| <code>maxfreeconnection</code> | inteiro | Número máximo de ligações disponíveis num conjunto de ligações. |
| <code>minfreeconnection</code> | inteiro | Número mínimo de ligações disponíveis num conjunto de ligações |
| <code>timeout</code> | inteiro | A duração de tempo (em milisegundos) depois do qual uma ligação livre irá ser desligada e destruída. |
| <code>noSessionPage</code> | URL | Esta é uma página que serve para ser apresentada no início de sessão, caso o servlet seja invocado sem uma sessão ou ligação estabelecida. Pode ser utilizada para pedir o início de sessão e efectuar o encadeamento de volta à acção original, permitindo ligações marcadas ao EIP para trabalhar, mesmo que o utilizador tenha de iniciar uma sessão. |

| Parâmetro da aplicação | Valores | Descrição |
|----------------------------|-------------------------------|--|
| timedOutPage | URL | Esta página serve para apresentação, caso a sessão tenha esgotado o tempo de espera devido a inactividade. |
| serverErrorPage | URL | Esta página serve para apresentação caso tenha ocorrido um erro ao aceder a um servidor. |
| connectFailedPage | URL | Esta página serve para apresentação caso tenha ocorrido um erro ao estabelecer ligação a um servidor. Pode ser apresentado um pedido de para introduzir o id/palavra-passe de utilizador correctos para o servidor e pode ser efectuada uma nova tentativa. |
| nível de rastreio | 0, 1, ou 2 | Para indicar o nível de rastreio: <ul style="list-style-type: none"> • 0 - sem registo • 1 - excepções de registo (predefinição) • 2 - excepções de registo, mensagens de alerta, cabeçalhos e atributos de WebSphere Application Server, ficheiros ini de EIP, propriedades de sistema JVM, informações de rastreio interno de EIP |
| tipo de ligação | 0, 1, ou 2 | A localização da base de dados de EIP e dos tempos de execução do servidor de conteúdos: <ul style="list-style-type: none"> • 0 - local (predefinição) • 1 - remoto • 2 - dinâmico |
| cmbclient | URL | Localização do cmbclient.ini |
| cmbcs | URL | Localização de cmbcs.ini |
| tipo de ligação de serviço | 0, 1, ou 2 | Localização dos tempos de execução de serviço <ul style="list-style-type: none"> • 0 - local (predefinição) • 1 - remoto • 2 - dinâmico |
| cmbsvclient | URL | Localização do cmbsvclient.ini |
| cmbsvcs | URL | Localização do cmbsvcs.ini |
| cmbcc2mime | URL | Localização do cmbcc2mime.ini |
| cachedir | nome de um directório | Directório para colocação de documentos em memória cache durante a conversão de documentos |
| jnitrace | nome de um ficheiro | Ficheiro no qual gravar as informações de rastreio JNI para o lógico JNI utilizado na conversão de documentos (para finalidades de diagnóstico IBM) |
| conversão | Booleana: verdadeira ou falsa | Se for verdadeira, os documentos são convertidos para formatos que podem ser apresentados num browser de escalão intermédio, se possível. Se for falso, o documento original, não convertido, é enviado para o browser. |

| Parâmetro da aplicação | Valores | Descrição |
|------------------------|---------------------------|---|
| maxresults | inteiro | Máximo de acertos devolvido; -1 (predefinição) significa todos os acertos. |
| delimitador de valor | carácter | Define o carácter que irá delimitar valores em critérios de pesquisa. O carácter predefinido é dependente de locale e consiste numa vírgula (,) no caso do Inglês dos Estados Unidos. |
| conversion.<mimetype> | <none document page > | Opções de conversão para visualizar documentos de um tipo de mime específico. Isto afecta o comportamento do servlet viewDocument. Paginar significa a tentativa de paginar o documento. Documento significa a conversão do documento para uma forma passível de ser lida num browser. Nenhum significa a execução de nenhuma conversão -- devolver o documento à sua forma nativa. |
| nameseparator | carácter | Define o carácter que irá separar o atributo de componente descendente a partir do atributo de componente ascendente em nomes qualificados. A predefinição é dependente do locale, e é uma barra (/) para o Inglês dos EUA. |

Ficheiro de Propriedades

Este servlet procura um ficheiro de propriedades, `cmbervlet.properties`. Este ficheiro define as acções que o servlet pode utilizar, incluindo as acções aqui definidas. Também define os nomes dos ficheiros de JSP que são utilizados.

O utilizador pode também definir as propriedades do servlet no servidor de aplicação da Web (motor de servlet). A sintaxe é igual à utilizada no ficheiro.

O conteúdo do ficheiro `cmbervlet.properties` é armazenado num objecto `Properties` pelo servlet de controlo. Pode ser acedido através do atributo da aplicação `"cmbServletProperties,"` como se mostra no exemplo seguinte.

```
// check to see if connection pooling is enabled
String name = "connectionpool";
Properties props = (Properties) application.getAttribute
("cmbServletProperties");
String value = props.getProperty(name);
// "true" if enabled, "false" otherwise
```

Parâmetros de pedido

O servlet compreende os seguintes parâmetros de pedido. Podem ser especificados parâmetros adicionais para serem utilizados na JSP de resposta.

Geral

action=action

A acção a ser executada. Os parâmetros adicionais permitidos têm por base a acção e são descritos de seguida.

Este parâmetro é opcional. Se não for especificado, a página de resposta será executada pelo servlet após realizar a configuração standard como, por exemplo, verificar o tempo de espera excedido ou o início de sessão na ligação.

reply=<URL>

Opcional. Envia para a JSP especificada neste parâmetro em vez da JSP definida como a resposta para a acção no ficheiro `cmbServlet.properties`. Se o parâmetro de acção não for especificado, e a resposta for especificada, a página de resposta é executado pelo servlet do controlados depois de executar a configuração padrão, tal como a verificação da ligação para tempo excedido e início de sessão.

Relacionados com a Ligação

action=logon serverType=<> server=<> userid=<> password=<> [connstring=<>] [configstring=<>]

Inicie uma sessão num servidor. Deve especificar o tipo de servidor, o nome do servidor, ID e palavra-passe de utilizador. A cadeia de ligação e a cadeia init são facultativas e diferentes, dependendo do tipo de servidor.

action=logoff [endSession=<true|false>]

Termine a ligação ao servidor. A sessão é também terminada por predefinição.

Relacionados com pesquisa

action=searchTemplate template=<> {<criteriaName>.op=<> <criteriaName>=<>}

Execute uma pesquisa utilizando o modelo de pesquisa e os valores de critérios especificados.

action=searchEntity entity=<> {attribute.<attrName>.op=<> attribute.<attrName>=<>} [conjunction=<and|or>]

Execute uma pesquisa utilizando uma entidade. Os valores e operadores de atributos também podem ser especificados. Vários valores no valor de atributo são separados por um delimitador de valor, tal como está especificado nos parâmetros de aplicação. Os atributos são combinados para formar uma consulta utilizando `e`(predefinição) ou `ou`, tal como está especificado no parâmetro de intersecção.

action=searchQuery queryString=<> {queryParameter.<parameterName>=<>}

Execute uma pesquisa utilizando a cadeia de pesquisa especificada. A sintaxe da consulta depende do servidor que está a ser pesquisado.

Não existe um máximo de parâmetros de consulta adicionais que podem ser especificados. Estes também são dependentes do servidor.

Relacionados com artigos

action=lock itemId=<>

Bloquei um artigo, normalmente para acesso exclusivo ao actualizar.

action=unlock itemId=<>

Desbloqueie um artigobloqueado.

action=createItem type=<document|folder> entity=<> {attribute.<attrName>=<attrValue>}

Crie um artigo. Se este for registado, o conteúdo pode ser facultado.

action=retrieveItem itemId=<>

Obtém os atributos e o conteúdo de um artigo. É útil para garantir que o conteúdo mais recente armazenado no servidor é utilizado.

action=updateItem itemId=<> [entity=<>]

{attribute.<attrname>=<attrvalue>}

Atualize os atributos de um artigo. Se for especificada a entidade, o artigo é novamente indexado. O conteúdo é também actualizado se o servlet for invocado através de um suporte.

action=deleteItem itemId=<>

Elimine um artigo.

action=addContent itemId=<>

Adicione uma parte do conteúdo a um artigo. Os dados do conteúdo são registados.

action=getContent itemId=<> contentIndex=<>

Obtém a parte do conteúdo e devolve-a ao browser.

action=updateContent itemId=<> contentIndex=<>

Atualize uma parte do conteúdo num artigo; os dados do conteúdo são registados. Se não existir um conteúdo, é adicionada uma parte de conteúdo.

action=deleteContent itemId=<> contentIndex=<>

Elimine uma parte do conteúdo para o artigo especificado.

action=addNoteLog itemId=<>

Modifica o registo de nota de um artigo. O texto do registo de nota é registado.

action=updateNoteLog itemId=<> notelogIndex=<>

Modifica o registo de nota de um artigo; o texto do registo de nota é registado. Se não existir um registo de nota, este é adicionado.

action=deleteNoteLog itemId=<> notelogIndex=<>

Elimina o texto do registo de nota de um artigo. O texto do registo de nota é registado.

Relacionados com pastas

action=addItemToFolder itemId=<> folderId=<>

Adiciona o artigo especificado à pasta especificada.

action=removeItemFromFolder itemId=<> folderId=<>

Remove o artigo especificado da pasta especificada.

Relacionados com documentos

action=viewDocument itemId=<>

Obtém o documento e revê-o. Se o documento se encontrar paginado, esta acção remete para uma JSP que gera a estrutura do visualizador. Se o documento não se encontrar paginado, esta acção devolve o conteúdo actual do documento.

action=viewPage itemId=<> page=<> scale=<> rotation=<>

annotations=<yes|no>

Obtém uma página do documento.

Matrix de função de toolkit do servlet

Tabela 33. Matriz de função do servlet

| Ação | CMv8 | CMv7 | VI/400 | IP/390 | OD/Wkstn | OD/390 | DD |
|--------------------------|------|------|--------|--------|----------|--------|-----|
| logon | Y | Y | Y | Y | Y | Y | Y |
| logoff | Y | Y | Y | Y | Y | Y | Y |
| searchTemplate | N/A | N/A | N/A | N/A | Y | Y | N/A |
| searchEntity | Y | Y | Y | Y | Y | Y | N/A |
| searchQuery | Y | Y | Y | Y | Y | Y | Y |
| lock | Y | Y | Y | Y | N/A | N/A | N/A |
| unlock | Y | Y | Y | Y | N/A | N/A | N/A |
| createItem | Y | Y | N/A | N/A | N/A | N/A | N/A |
| retrieveItem | Y | Y | Y | Y | Y | Y | Y |
| updateItem | Y | Y | Y | Y | N/A | N/A | N/A |
| deleteItem | Y | Y | Y | Y | N/A | N/A | N/A |
| addContent | Y | Y | N/A | N/A | N/A | N/A | N/A |
| getContent | Y | Y | Y | Y | Y | Y | Y |
| updateContent | Y | Y | Y | Y | N/A | N/A | N/A |
| deleteContent | Y | Y | Y | Y | N/A | N/A | N/A |
| addNoteLog | Y | Y | Y | N/A | N/A | N/A | N/A |
| updateNoteLog | Y | Y | Y | N/A | N/A | N/A | N/A |
| deleteNoteLog | Y | Y | Y | N/A | N/A | N/A | N/A |
| addItemToFolder | Y | Y | Y | N/A | N/A | N/A | N/A |
| removeItem FromFolder | Y | Y | Y | N/A | N/A | N/A | N/A |
| viewDocument | Y | Y | Y | Y | Y | Y | Y |
| viewPage | Y | Y | Y | Y | Y | Y | Y |

Y -- Função é suportada **N/A** -- Função não é suportada **Nota:** As acções logon, logoff, getContent, retrieveItem, viewDocument e viewPage são suportadas em todos os servidores de conteúdos.

Tabela 34. Matriz de função do servlet Continuação

| Ação | DES | DB2 | JDBC | IC | Fed | FileNet |
|----------------|-----|-----|------|-----|-----|---------|
| logon | Y | Y | Y | Y | Y | Y |
| logoff | Y | Y | Y | Y | Y | Y |
| searchTemplate | N/A | N/A | N/A | N/A | Y | N/A |
| searchEntity | N/A | Y | Y | N/A | Y | N/A |
| searchQuery | Y | Y | Y | Y | Y | Y |
| lock | N/A | N/A | N/A | N/A | Y | N/A |
| unlock | N/A | N/A | N/A | N/A | Y | N/A |
| createItem | N/A | Y | Y | N/A | N/A | N/A |
| retrieveItem | Y | Y | Y | Y | Y | Y |
| updateItem | N/A | Y | Y | N/A | Y | N/A |

Tabela 34. Matrix de função do servlet Continuação (continuação)

| Acção | DES | DB2 | JDBC | IC | Fed | FileNet |
|----------------------|-----|-----|------|-----|-----|---------|
| deleteItem | N/A | Y | Y | N/A | Y | N/A |
| addContent | N/A | N/A | N/A | N/A | Y | N/A |
| getContent | Y | Y | Y | Y | Y | Y |
| updateContent | N/A | N/A | N/A | N/A | Y | N/A |
| deleteContent | N/A | N/A | N/A | N/A | Y | N/A |
| addNoteLog | N/A | N/A | N/A | N/A | Y | N/A |
| updateNoteLog | N/A | N/A | N/A | N/A | Y | N/A |
| deleteNoteLog | N/A | N/A | N/A | N/A | Y | N/A |
| addItemToFolder | N/A | N/A | N/A | N/A | N/A | N/A |
| removeItemFromFolder | N/A | N/A | N/A | N/A | N/A | N/A |
| viewDocument | Y | Y | Y | Y | Y | Y |
| viewPage | Y | Y | Y | Y | Y | Y |

Y -- Função é suportada N/A -- Função não é suportada **Nota:** As acções logon, logoff, getContent, retrieveItem, viewDocument e viewPage são suportadas em todos os servidores de conteúdos.

Trabalhar com extracção de informações

Esta secção começa com uma descrição da forma como se deve construir uma aplicação utilizando beans de Information Mining, tendo, de seguida, outras secções que descrevem a construção de um facultador de conteúdo do utilizador, utilizando a API de serviço de Information Mining e trabalhar com o exemplo de JSP.

Construir uma aplicação de Information Mining (extracção de informações) utilizando os beans

O Information Mining permite-lhe criar aplicações poderosas com base numa tecnologia de análise de texto e de extracção moderna. Os beans de Information Mining facultam:

- Resumo e categorização de texto, ou seja, a atribuição de um resumo a um documento
- Categorização, ou seja, a atribuição de uma categoria a um documento
- Extracção de informações, ou seja, localização e extracção de unidades de informação relevantes de um documento
- Identificação de idioma, ou seja, a determinação do idioma em que um documento está escrito
- Agrupamento, ou seja, agrupamento de documentos semelhantes num conjunto de documentos
- Pesquisa de texto que pode ser restringida a documentos de uma certa categoria
- Acesso a documentos que são continuamente obtidos a partir da Web utilizando o IBM Web Crawler

Nota

Antes de utilizar os bens de extracção de informações, é importante que compreenda as diferenças entre a API de serviço e os JavaBeans.

- Os JavaBeans de Information Mining são componentes de software para um rápido desenvolvimento de aplicações e são adaptáveis a JavaBeans. Determinados elementos do código estão 'unidos' e não podem ser utilizados pelo utilizador.
- A API de serviço de Java contém a funcionalidade completa de Information Mining como blocos de construção individuais para criar aplicações. Para obter mais informações, consulte "Utilizar a API de serviço" na página 500.

Antes de poder utilizar os beans de Information Mining, tem de instalar e configurar o componente Enterprise Information Portal Information Mining na máquina que irá executar os beans.

Os beans de Information Mining são:

- **CMBSummarizationService**

Trata-se de um bean não visual que pode ser utilizado para criar resumos de documentos. Pode ser executado em três modos diferentes e o utilizador pode determinar a extensão do resumo.

- **CMBCategorizationService**

Trata-se de um bean não visual que pode ser utilizado para determinar categorias de documentos. É executado segundo os resultados obtidos através de Ferramenta de Estruturação das Informações, nomeadamente os termos criados e constituintes de taxonomia e metadados, que foram identificados como sendo distintivos da categoria e das regras que descrevem cada categoria. Um documento pode ser atribuído a uma ou mais categorias. O resultado da categorização é formado por uma ou mais categorias e valores de fiabilidade, que mostram até que ponto o documento de encaixa em cada categoria. O utilizador pode definir este valor de fiabilidade, bem como o número máximo de resultados a serem devolvidos.

- **CMBInformationExtractionService**

Este é um bean não visual que pode ser utilizado para extrair nomes, termos e expressões de um documento.

- **CMBLanguageIdentificationService**

Este é um bean não visual que pode ser utilizado para determinar o idioma em que um documento está escrito.

- **CMBClusteringService**

Este é um bean não visual que pode ser utilizado para particionar uma recolha de documentos em conjuntos idênticos de documentos ou conjuntos de unidades.

- **CMBAdvancedSearchService**

O CMBAdvancedSearchService é um bean não visual que executa uma pesquisa. Uma pesquisa avançada apenas pode localizar artigos que foram armazenados no armazenamento de dados do Information Mining.

- **CMBCatalogService**

O CMBCatalogService mantém o armazenamento de dados permanente de Information Mining. Para utilizar este serviço, o utilizador terá de criar um catálogo utilizando Ferramenta de Estruturação das Informações. Pode utilizar o bean para armazenar resultados criados por quaisquer serviços de Information Mining, bem como buscar e eliminar estas informações.

- **CMBWebCrawlerService**

O CMBWebCrawlerService supervisiona o espaço da Web, ou seja, um directório onde o Web Crawler, que está em execução assincronamente, coloca todos os documentos em HTML buscados. São criados CMBItems (uma classe de programa de ajuda de EIP que apresenta um documento) a partir destes ficheiros, que podem ser utilizados com quaisquer serviços de Information Mining ou simplesmente importados para o armazenamento de dados de Information Mining utilizando CMBCatalogService.

- **CMBInfoMiningAdapter**

O CMBInfoMiningAdapter é um comutador entre diferentes eventos do Information Mining. Além disso, suporta o CMBResultEvent, que permite que os beans de Information Mining funcionem em conjunto com outros beans de Enterprise Information Portal. O CMBInfoMiningAdapter permite-lhe combinar beans para construir vários exemplos, como está demonstrado no diagrama que se segue.

Figura 39 na página 459 demonstra a utilização dos beans de Information Mining.

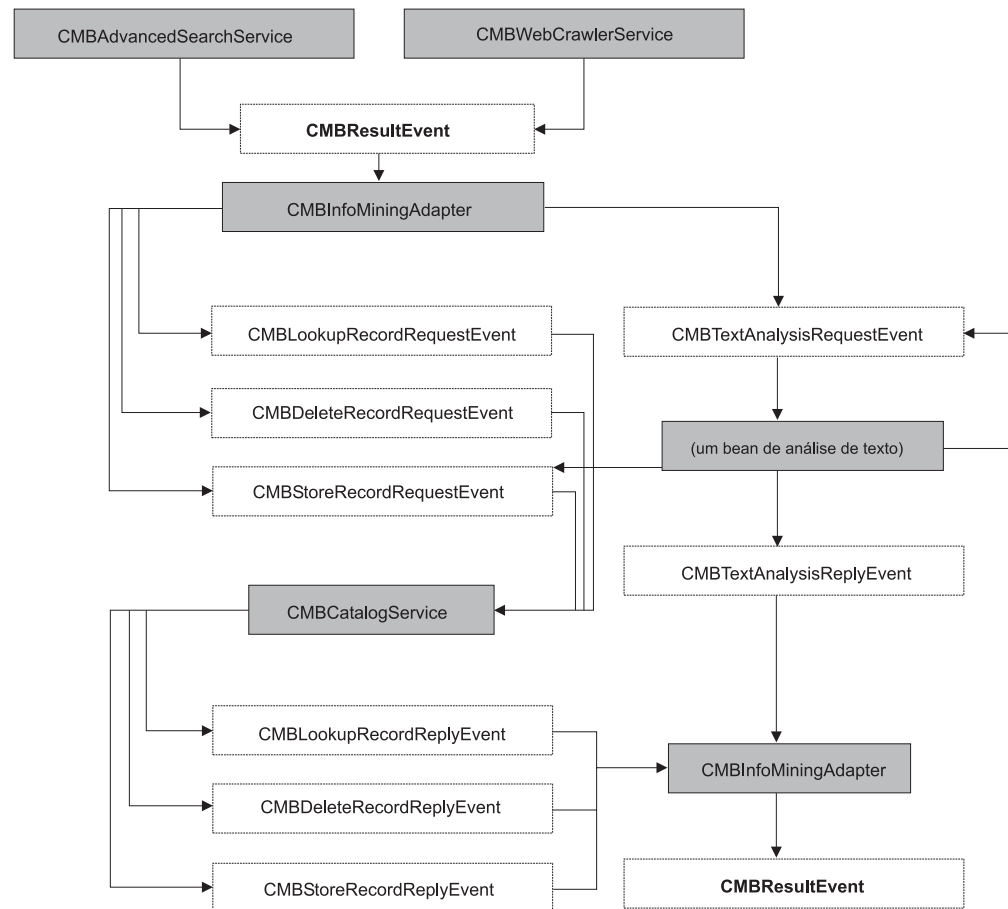


Figura 39. Os beans de Information Mining

Figura 40 enumera os beans de análise de texto.

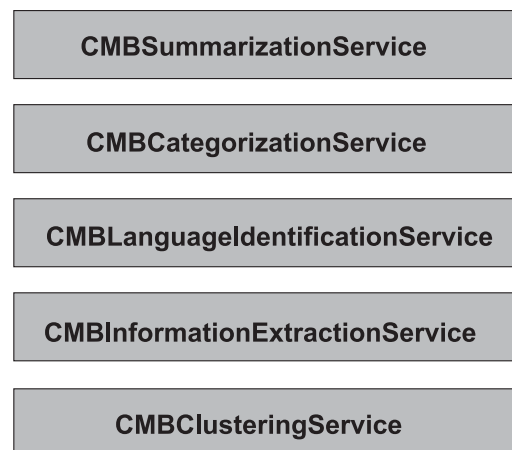


Figura 40. Os beans de análise de texto

Os seguintes exemplos demonstram como pode utilizar os beans de extracção de informações para construir aplicações:

- **Exemplo de categorização:** Reunir informações em preparação para Information Mining é um passo típico para um bibliotecário. Consulte Figura 41 na página 461. Neste exemplo, irá começar por efectuar uma pesquisa de Enterprise

Information Portal (EIP) padrão para localizar documentos no servidor de conteúdos do EIP. (Reunir documentos da Web está no exemplo do Web Crawler.) Seguidamente é determinado o idioma dos documentos e os documentos em Inglês são categorizados. As informações de categoria são armazenadas no armazenamento de metadados.

- **O exemplo de resumo:** Este constitui outro passo típico para um bibliotecário. Tal como no exemplo de categorização, o utilizador começa por realizar uma pesquisa padrão de EIP para localizar documentos no servidor de conteúdos do EIP, identificar documentos em Inglês e, de seguida, resumir estes documentos e armazenar os resumos no armazenamento de metadados. Consulte Figura 43 na página 469.
- **O exemplo de extracção de informações:** Este é um passo de extracção de informações. Neste exemplo, irá começar por efectuar uma pesquisa padrão de EIP para localizar documentos no servidor de conteúdos do EIP. Os nomes e termos extraídos dos documentos em Inglês. As informações extraídas são armazenadas no armazenamento de metadados.

Esta acção está descrita em Figura 45 na página 475.

- **O exemplo de agrupamento:** Este é um passo de extracção de informações. Começa por efectuar uma pesquisa padrão de EIP para localizar documentos no servidor de conteúdos do EIP. De seguida, são determinados os documentos em Inglês e o serviço de agrupamento é manualmente activado. Os resultados são apresentados no monitor. As informações de agrupamento não são armazenadas no armazenamento de metadados.

Esta acção está descrita em “Agrupamento” na página 480.

- **O exemplo de Web Crawler:** Obtenha documentos utilizando o Web Crawler e disponibilize as informações para pesquisa dentro das categorias (o exemplo de pesquisa avançada). Trata-se também de um passo de bibliotecário, semelhante ao exemplo de elaboração de categorias e de resumos, excepto que o utilizador irá reunir os documentos através da Internet ou de uma intranet, em vez de um servidor de conteúdos. Para restaurar as informações de categorias e resumos do catálogo para a pesquisa avançada, irá recuperá-las através do serviço de catálogos.

Consulte Figura 49 na página 485.

- **O exemplo de pesquisa avançada:** Trata-se de um passo da extracção de informações. Efectue uma pesquisa avançada, que lhe irá permitir a pesquisa de documentos através de uma consulta flexível limitada para categorias específicas. Para restaurar as informações de categorias e resumos do catálogo para a pesquisa avançada, irá recuperá-las através do serviço de catálogos.

Localização dos ficheiros exemplo

Os exemplos descritos neste capítulo são fornecidos nos directórios seguintes:

| Exemplo | Localização |
|---------------------------------------|---|
| Aplicação de elaboração de categorias | <CMBROOT>\samples\java\beans\infomining\categorization |
| Aplicação de elaboração de resumos | <CMBROOT>\samples\java\beans\infomining\summarization |
| Aplicação de extracção de informações | <CMBROOT>\samples\java\beans\infomining\informationExtraction |
| Aplicação de agrupamento | <CMBROOT>\samples\java\beans\infomining\clustering |

Aplicação de pesquisa avançada

<CMBROOT>\samples\java\beans\infomining\advancedSearch

Aplicação do Web Crawler

<CMBROOT>\samples\java\beans\infomining\webcrawler

Cada um destes directórios contém um ficheiro compile.bat para permitir ao utilizador a compilação do código de origem. Os directórios de exemplos de aplicações também podem conter um ficheiro run.bat para permitir ao utilizador a execução das aplicações exemplo.

Categorizar documentos

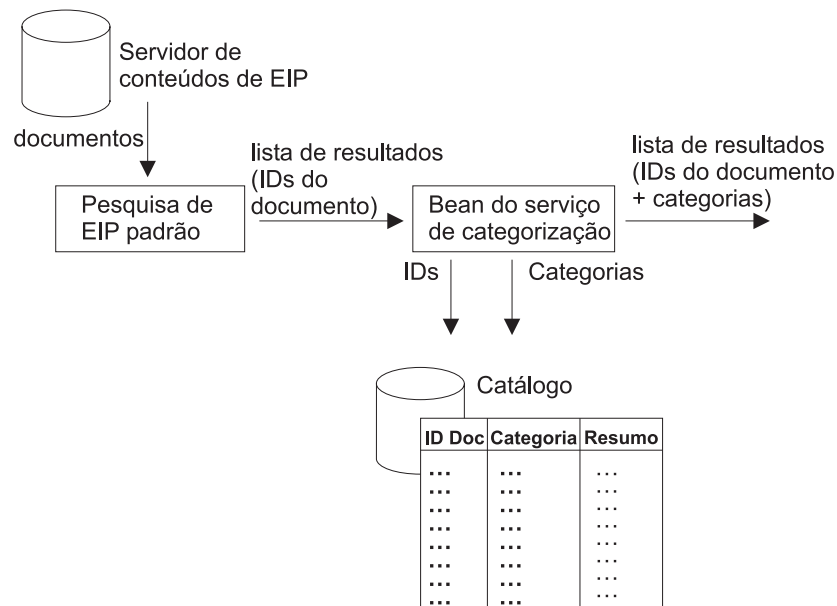


Figura 41. O exemplo de categorização

Este exemplo demonstra como o utilizador pode categorizar documentos que foram obtidos por uma pesquisa do EIP padrão. Os resultados da análise são armazenados e os documentos apropriados são disponibilizados para pesquisa nas categorias.

Na Figura 41, uma pesquisa do EIP padrão é efectuada em documentos retidos no servidor de conteúdos do EIP. O bean de serviço de identificação de idioma determina o idioma no qual os documentos estão escritos. O bean utiliza os IDs de documento para obter o conteúdo do documento e, de seguida, analisa o conteúdo para determinar o idioma. O bean de serviço de categorização categoriza os documentos. Estas informações, os IDs dos documentos e as informações de categorias são depois disponibilizadas para um processamento posterior.

Os beans seguintes são utilizados neste exemplo:

- CMBConnection
- CMBQueryService (para executar a pesquisa do EIP padrão)
- CMBSearchResults (para executar a pesquisa do EIP padrão)
- CMBInfoMiningAdapter
- CMBLanguageIdentificationService

- CMBCategorizationService
- CMBCatalogService

Para este exemplo, a aplicação:

1. Cria os beans.
2. Personaliza os beans.
3. O serviço de categorização analisa os documentos em Inglês. Os resultados são armazenados no catálogo.
4. Executa a consulta do EIP padrão.
5. Apresenta os resultados da pesquisa (listas de documentos) e categorias para verificação.

A fonte Java é a seguinte.

Origem completa para Categorization.java

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBQueryService;
import com.ibm.mm.beans.CMBSearchResults;
import com.ibm.mm.beans.CMBSchemaManagement;
import com.ibm.mm.beans.CMBSearchTemplate;
import com.ibm.mm.beans.CMBItem;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBBaseConstant;
import com.ibm.mm.beans.CMBSearchRequestEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBRecord;
import com.ibm.mm.beans.infomining.CMBDefaultContentProvider;
import com.ibm.mm.beans.infomining.CMBLanguageIdentificationService;
import com.ibm.mm.beans.infomining.CMBCategorizationService;
import com.ibm.mm.beans.infomining.CMBCatalogService;
import com.ibm.mm.beans.infomining.CMBInfoMiningAdapter;
import com.ibm.mm.beans.infomining.CMBNoSuchKeyException;

a classe pública Categorização implementa CMBResultListener, CMBExceptionListener
{

    String DEFAULT_CMDBNAME      = "icmnlbdb";
    String DEFAULT_CMDBUSER      = "icmadmin";
    String DEFAULT_CMDBPASSWORD  = "password";
    String DEFAULT_SAMPDBNAME    = "eipsampl";
    String DEFAULT_SAMPDBUSER    = "icmadmin";
    String DEFAULT_SAMPDBPASSWORD = "password";

    String CATALOG_NAME          = "Sample";
    String SEARCH_TEMPLATE       = "SearchLongBySource";
    String SEARCH_VALUE          = "ibmpress";
    String SEARCH_CRITERION      = "source";

    public Categorization() throws Throwable
    {
        // criar beans
        CMBConnection samplConnection = new CMBConnection();
        CMBConnection eipConnection = new CMBConnection();
        CMBQueryService queryService = new CMBQueryService();
        CMBSearchResults searchResults = new CMBSearchResults();
```

```

CMBLanguageIdentificationService languageIdentificationService =
    new CMBLanguageIdentificationService();
CMBCategorizationService categorizationService =
    new CMBCategorizationService();
CMBCatalogService catalogService = new CMBCatalogService();
CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();

// ler parâmetros
BufferedReader din = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Sample database      [" + DEFAULT_SAMPDBNAME + "] : ");
String sampDbName = din.readLine();
if (sampDbName.trim().equals("")) sampDbName = DEFAULT_SAMPDBNAME;

System.out.print("User ID for " + sampDbName +
    " [" + DEFAULT_SAMPDBUSER + "] : ");
String sampUserName = din.readLine();
if (sampUserName.trim().equals("")) sampUserName = DEFAULT_SAMPDBUSER;

System.out.print("Password for " + sampDbName +
    " [" + DEFAULT_SAMPDBPASSWORD + "] : ");
String sampPasswd = din.readLine();
if (sampPasswd.trim().equals("")) sampPasswd = DEFAULT_SAMPDBPASSWORD;

System.out.print("EIP database      [" + DEFAULT_CMDBNAME + "] : ");
String eipDbName = din.readLine();
if (eipDbName.trim().equals("")) eipDbName = DEFAULT_CMDBNAME;

System.out.print("User ID for
    " + eipDbName + " [" + DEFAULT_CMDBUSER + "] : ");
String eipUserName = din.readLine();
if (eipUserName.trim().equals("")) eipUserName = DEFAULT_CMDBUSER;

System.out.print("Password for " + eipDbName +
    " [" + DEFAULT_CMDBPASSWORD + "] : ");
String eipPasswd = din.readLine();
if (eipPasswd.trim().equals("")) eipPasswd = DEFAULT_CMDBPASSWORD;

System.out.print("Catalog      [" + CATALOG_NAME + "] : ");
String catalog = din.readLine();
if (catalog.trim().equals("")) catalog = CATALOG_NAME;

System.out.println("\n\nRunning Categorization with the
    following settings:");
System.out.println("Sample database      = " + sampDbName);
System.out.println("User ID for " + sampDbName + " = " + sampUserName);
System.out.println("Password for " + sampDbName + " = " + sampPasswd);
System.out.println("EIP Database      = " + eipDbName);
System.out.println("User ID for " + eipDbName + " = " + eipUserName);
System.out.println("Password for " + eipDbName + " = " + eipPasswd);
System.out.println("Catalog      = " + catalog + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == 'n') System.exit(0); }
while (key != 'y');

// personalizar beans
samplConnection.setServerName(sampDbName);
samplConnection.setUserid(sampUserName);
samplConnection.setPassword(sampPasswd);
samplConnection.setConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

```

```

    samplConnection.setServiceConnectionType
        (CMBConnection.CMB_CONNTYPE_DYNAMIC);

    eipConnection.setServerName(eipDbName);
    eipConnection.setUserid(eipUserName);
    eipConnection.setPassword(eipPasswd);
    eipConnection.setConnectToIKF(true);
    eipConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
    eipConnection.setServiceConnectionType
        (CMBConnection.CMB_CONNTYPE_DYNAMIC);

    adapter1.setContentProvider
        (new CMBDefaultContentProvider());
    adapter1.setCatalogName(catalog);
    categorizationService.setCatalogName(catalog);
    catalogService.setCatalogName(catalog);
    adapter2.setCatalogName(catalog);

    // ligar beans
    samplConnection.addCMBConnectionReplyListener(queryService);
    samplConnection.addCMBConnectionReplyListener(searchResults);

    eipConnection.addCMBConnectionReplyListener(adapter1);
    eipConnection.addCMBConnectionReplyListener
        (languageIdentificationService);
    eipConnection.addCMBConnectionReplyListener
        (categorizationService);
    eipConnection.addCMBConnectionReplyListener(catalogService);
    eipConnection.addCMBConnectionReplyListener(adapter2);

    queryService.addCMBSearchReplyListener(searchResults);
    searchResults.addCMBResultListener(adapter1);
    adapter1.addCMBTextAnalysisRequestListener
        (languageIdentificationService);
    languageIdentificationService.addCMBTextAnalysisRequestListener
        (categorizationService);
    categorizationService.addCMBStoreRecordRequestListener
        (catalogService);
    catalogService.addCMBStoreRecordReplyListener(adapter2);
    adapter2.addCMBResultListener(this);

    samplConnection.addCMBExceptionListener(this);
    eipConnection.addCMBExceptionListener(this);
    queryService.addCMBExceptionListener(this);
    searchResults.addCMBExceptionListener(this);
    languageIdentificationService.addCMBExceptionListener(this);
    categorizationService.addCMBExceptionListener(this);
    catalogService.addCMBExceptionListener(this);
    adapter1.addCMBExceptionListener(this);
    adapter2.addCMBExceptionListener(this);

    // executar consulta
    samplConnection.connect();
    eipConnection.connect();
    catalogService.setDefaultCategoryPath
        (catalogService.getTaxonomy().getRootCategory().getPathAsString());
    CMBSchemaManagement schema = samplConnection.getSchemaManagement();
    CMBSearchTemplate searchTemplate = schema.getSearchTemplate(SEARCH_TEMPLATE);
    String[] searchValues = { SEARCH_VALUE };
    searchTemplate.setSearchCriterion
        (SEARCH_CRITERION, CMBBaseConstant.CMB_OP_EQUAL, searchValues);

    CMBSearchRequestEvent searchRequest = new
        CMBSearchRequestEvent(this, CMBSearchRequestEvent.CMB_REQUEST_SEARCH_SYNCH,
            searchTemplate);
    queryService.onCMBSearchRequest(searchRequest);

```



```

        sampleConnection.disconnect();
        eipConnection.disconnect();
    }

    // implementar com.ibm.mm.beans.CMBResultListener
    public void onCMBResult(CMBResultEvent e)
    {
        if(e.getID() == CMBResultEvent.CMB_RESULT_CLEARED)
            return;

        Vector cmbItemVector = (Vector)e.getData();

        if(cmbItemVector == null)
            return;

        try {
            for(int i = 0; i < cmbItemVector.size(); i++)
            {
                CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

                CMBRecord currentRecord = currentItem.getInfoMiningRecord();

                System.out.println("\n\nPID      : " + currentRecord.getPID());
                System.out.println("Categories : " + currentRecord.getValue("IKF_CATEGORIES"));
            } /* for */
        }
        catch (CMBNoSuchKeyException nske)
        {
            nske.printStackTrace();
        }
    }

    //implementar com.ibm.mm.beans.CMBExceptionListener
    public void onCMBException(CMBExceptionEvent e)
    {
        ((Exception)e.getData()).printStackTrace();
    }

    public static void main(String[] args)
    {
        try
        {
            new Categorization();
            System.exit(0);
        }
        catch(Throwable t)
        {
            t.printStackTrace();
        }
    }
}

```

Criar os beans

Necessita de uma ligação a um servidor de conteúdos para executar uma pesquisa. A ligação pode ser estabelecida através do bean CMBConnection. Os beans CMBQueryService e CMBSearchResults têm de executar uma pesquisa do EIP padrão. O idioma dos documentos é determinado utilizando o CMBLanguageIdentificationService e é armazenado no atributo IKF_LANGUAGE do registo correspondente. Os documentos são atribuídos a uma ou mais categorias utilizando CMBCategorizationService, que também armazena as informações da categoria nos registos adequados. O CMBCatalogService é utilizado para armazenar as informações de categorias do artigo no catálogo e disponibilizar os documentos apropriados para pesquisa avançada. Os dois adaptadores são necessários para converter os eventos do resultado da pesquisa em eventos de

pedido de análise de texto, armazenando em seguida os eventos de resposta do artigo de novo nos eventos do resultado da pesquisa.

O código que cria os beans necessários é:

```
CMBConnection samplConnection = new CMBConnection();
CMBConnection eipConnection = new CMBConnection();
CMBQueryService queryService = new CMBQueryService();
CMBSearchResults searchResults = new CMBSearchResults();
CMBLanguageIdentificationService languageIdentificationService =
    new CMBLanguageIdentificationService();
CMBCategorizationService categorizationService =
    new CMBCategorizationService();
CMBCatalogService catalogService = new CMBCatalogService();
CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();
```

Personalizar os beans

O código demonstrado na seção de personalização tem de ser adaptado segundo a sua instalação. No bean de ligação o utilizador terá de especificar o nome do servidor de conteúdos, um ID de utilizador e a palavra-passe adequada.

Antes de poder executar o bean do serviço de elaboração de categorias e o bean do serviço de catálogos, terá de os associar a um catálogo existente. Além disso, o bean do serviço de catálogo necessita de uma categoria predefinida para a qual os artigos são atribuídos, que não contenha nenhuma informação de categoria.

O código que realiza a personalização para o exemplo é:

```
samplConnection.setServerName(sampDbName);
samplConnection.setUserid(sampUserName);
samplConnection.setPassword(sampPasswd);
samplConnection.setConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);
samplConnection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

eipConnection.setServerName(eipDbName);
eipConnection.setUserid(eipUserName);
eipConnection.setPassword(eipPasswd);
eipConnection.setConnectToIKF(true);
eipConnection.setConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);
eipConnection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

adapter1.setContentProvider
    (new CMBDefaultContentProvider());
adapter1.setCatalogName(catalog);
categorizationService.setCatalogName(catalog);
catalogService.setCatalogName(catalog);
adapter2.setCatalogName(catalog);
```

Ligar os beans

A Figura 42 na página 467 ilustra o fluxo de eventos entre os beans.

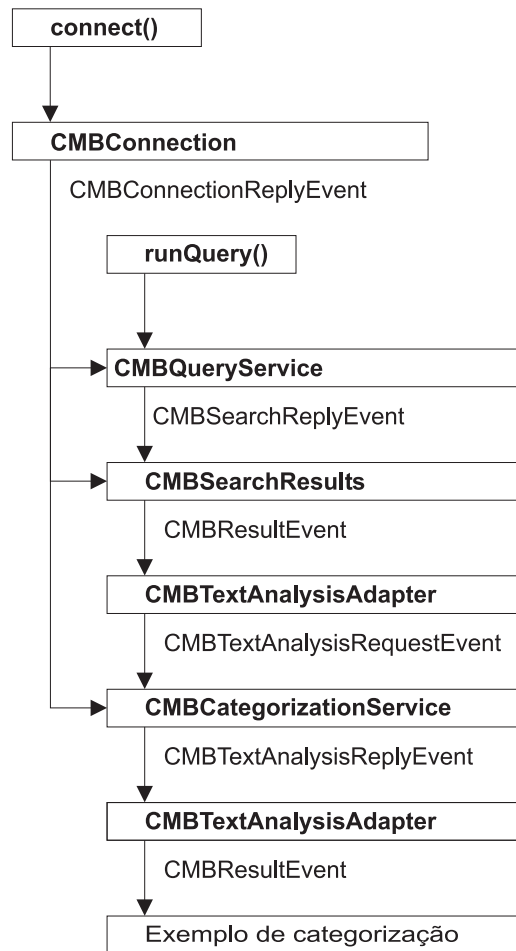


Figura 42. O exemplo de elaboração de categorias: Fluxo de eventos

Cinco dos beans utilizados neste exemplo aguardam o `CMBConnectionReplyEvent` para obter o parâmetro identificador da ligação. O bean `CMBQueryService` inicia uma pesquisa que resulta num evento que depois inicia o fluxo de eventos através dos outros beans.

O código que liga os beans é:

```

sampleConnection.addCMBConnectionReplyListener(queryService);
sampleConnection.addCMBConnectionReplyListener(searchResults);

eipConnection.addCMBConnectionReplyListener(adapter1);
eipConnection.addCMBConnectionReplyListener
    (languageIdentificationService);
eipConnection.addCMBConnectionReplyListener
    (categorizationService);
eipConnection.addCMBConnectionReplyListener(catalogService);
eipConnection.addCMBConnectionReplyListener(adapter2);

queryService.addCMBSearchReplyListener(searchResults);
searchResults.addCMBResultListener(adapter1);
adapter1.addCMBTextAnalysisRequestListener
    (languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener
    (categorizationService);
categorizationService.addCMBStoreRecordRequestListener
    (catalogService);
catalogService.addCMBStoreRecordReplyListener(adapter2);
  
```

```

adapter2.addCMBResultListener(this);

samlConnection.addCMBExceptionListener(this);
eipConnection.addCMBExceptionListener(this);
queryService.addCMBExceptionListener(this);
searchResults.addCMBExceptionListener(this);
languageIdentificationService.
    addCMBExceptionListener(this);
categorizationService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);
adapter1.addCMBExceptionListener(this);
adapter2.addCMBExceptionListener(this);

```

Visto que as excepções são também enviadas como eventos, a classe de Elaboração de Categorias tem de manusear o evento apropriado através da implementação da interface CMBExceptionListener, estando ligada aos beans para ser notificada sobre excepções.

Sugestão

Pode combinar a utilização do bean do serviço de elaboração de categorias e o bean de serviço de elaboração de resumos, um após o outro, em qualquer sequência, para analisar documentos. Consulte “Importar documentos a partir de um espaço da web” na página 485 para obter informações sobre a execução desta acção.

Executar a consulta

Antes de poder executar a consulta de que necessita para estabelecer a ligação ao servidor de conteúdos através da chamada do método de ligação no bean CMBConnection:

```

samlConnection.connect();
eipConnection.connect();

```

Para iniciar uma pesquisa do EIP padrão, terá de especificar um modelo de pesquisa, um critério do modelo, o operador de comparação e, é claro, um valor de pesquisa.

Terá de adaptar o código seguinte segundo a sua configuração:

```

catalogService.setDefaultCategoryPath
    (catalogService.getTaxonomy().getRootCategory().getPathAsString());
CMBSchemaManagement schema = connection.getSchemaManagement();
CMBSearchTemplate searchTemplate = schema.getSearchTemplate(SEARCH_TEMPLATE);
String[] searchValues = { SEARCH_VALUE };
searchTemplate.setSearchCriterion(SEARCH_CRITERION, CMBBaseConstant.CMB_OP_EQUAL,
    searchValues);
CMBSearchRequestEvent searchRequest = new CMBSearchRequestEvent(this,
    CMBSearchRequestEvent.CMB_REQUEST_SEARCH_SYNCH, searchTemplate);
queryService.onCMBSearchRequest(searchRequest);

```

Para fechar a ligação actual, chame os métodos de desligar:

```

samlConnection.disconnect();
eipConnection.disconnect();

```

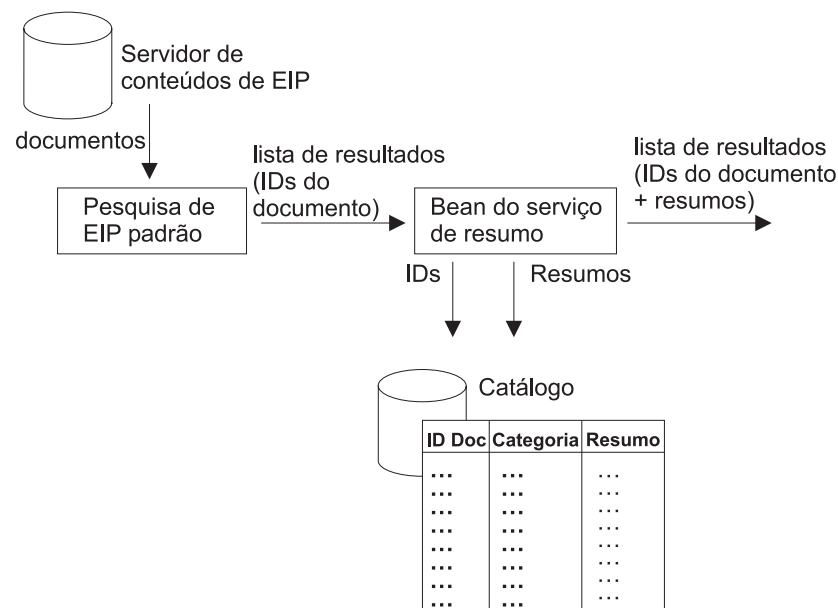
Apresentar resultados da análise de texto

A classe de Elaboração de Categorias implementa a interface CMBResultListener para poder enumerar os documentos que foram localizados durante a pesquisa e para apresentar as informações de categorias criadas para cada documento. O

```
Vector cmbItemVector = (Vector)e.getData();
```

```
CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);
CMBRecord currentRecord = currentItem.getInfoMiningRecord();
System.out.println("\n\nPID          : " + currentRecord.getPID());
```

```
System.out.println("Categories : " + currentRecord.getValue("IKF_CATEGORIES"));
```



- CMBSearchResults (para executar a pesquisa do EIP padrão)
- CMBInfoMiningAdapter
- CMBLanguageIdentificationService
- CMBSummarizationService
- CMBCatalogService

Origem completa para Summarization.java

Segue-se a origem completa do exemplo de elaboração de resumos. Visto que os beans de elaboração de resumos e de categorias operam de forma semelhante, consulte “Categorizar documentos” na página 461 para obter mais detalhes.

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBQueryService;
import com.ibm.mm.beans.CMBSearchResults;
import com.ibm.mm.beans.CMBSchemaManagement;
import com.ibm.mm.beans.CMBSearchTemplate;
import com.ibm.mm.beans.CMBItem;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBBaseConstant;
import com.ibm.mm.beans.CMBSearchRequestEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBRecord;
import com.ibm.mm.beans.infomining.CMBDefaultContentProvider;
import com.ibm.mm.beans.infomining.CMBLanguageIdentificationService;
import com.ibm.mm.beans.infomining.CMBSummarizationService;
import com.ibm.mm.beans.infomining.CMBCatalogService;
import com.ibm.mm.beans.infomining.CMBInfoMiningAdapter;
import com.ibm.mm.beans.infomining.CMBNoSuchKeyException;

public class Summarization implements CMBResultListener, CMBExceptionListener
{
    String DEFAULT_CMDBNAME      = "icmnlbdb";
    String DEFAULT_CMDBUSER      = "icmadmin";
    String DEFAULT_CMDBPASSWORD = "password";
    String DEFAULT_SAMPDBNAME    = "eipsampl";
    String DEFAULT_SAMPDBUSER    = "icmadmin";
    String DEFAULT_SAMPDBPASSWORD = "password";

    String CATALOG_NAME          = "Sample";
    String SEARCH_TEMPLATE       = "SearchLongBySource";
    String SEARCH_VALUE          = "ibmpress";
    String SEARCH_CRITERION      = "source";

    public Summarization() throws Throwable
    {
        // criar beans
        CMBConnection samplConnection = new CMBConnection();
        CMBConnection eipConnection = new CMBConnection();
        CMBQueryService queryService = new CMBQueryService();
        CMBSearchResults searchResults = new CMBSearchResults();
        CMBLanguageIdentificationService languageIdentificationService =
            new CMBLanguageIdentificationService();
        CMBSummarizationService summarizationService = new CMBSummarizationService();
        CMBCatalogService catalogService = new CMBCatalogService();
        CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
        CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();

        // 1er parâmetros
```

```

BufferedReader din = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Sample database      [" + DEFAULT_SAMPDBNAME + "] : ");
String sampDbName = din.readLine();
if (sampDbName.trim().equals("")) sampDbName = DEFAULT_SAMPDBNAME;

System.out.print("User ID for " + sampDbName +
                " [" + DEFAULT_SAMPDBUSER + "] : ");
String sampUserName = din.readLine();
if (sampUserName.trim().equals("")) sampUserName = DEFAULT_SAMPDBUSER;

System.out.print("Password for " + sampDbName +
                " [" + DEFAULT_SAMPDBPASSWORD + "] : ");
String sampPasswd = din.readLine();
if (sampPasswd.trim().equals("")) sampPasswd = DEFAULT_SAMPDBPASSWORD;

System.out.print("EIP database      [" + DEFAULT_CMDBNAME + "] : ");
String eipDbName = din.readLine();
if (eipDbName.trim().equals("")) eipDbName = DEFAULT_CMDBNAME;

System.out.print("User ID for " + eipDbName
                + " [" + DEFAULT_CMDBUSER + "] : ");
String eipUserName = din.readLine();
if (eipUserName.trim().equals("")) eipUserName = DEFAULT_CMDBUSER;

System.out.print("Password for " + eipDbName +
                " [" + DEFAULT_CMDBPASSWORD + "] : ");
String eipPasswd = din.readLine();
if (eipPasswd.trim().equals("")) eipPasswd = DEFAULT_CMDBPASSWORD;

System.out.print("Catalog      [" + CATALOG_NAME + "] : ");
String catalog = din.readLine();
if (catalog.trim().equals("")) catalog = CATALOG_NAME;

System.out.println("\n\nExecutar Resumo com as seguintes definições:");
System.out.println("Sample database      = " + sampDbName);
System.out.println("User ID for " + sampDbName + " = " + sampUserName);
System.out.println("Password for " + sampDbName + " = " + sampPasswd);
System.out.println("EIP Database      = " + eipDbName);
System.out.println("User ID for " + eipDbName + " = " + eipUserName);
System.out.println("Password for " + eipDbName + " = " + eipPasswd);
System.out.println("Catalog      = " + catalog + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == 'n') System.exit(0); }
while (key != 'y');

// personalizar beans
samlConnection.setServerName(sampDbName);
samlConnection.setUserid(sampUserName);
samlConnection.setPassword(sampPasswd);
samlConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
samlConnection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);

eipConnection.setServerName(eipDbName);
eipConnection.setUserid(eipUserName);
eipConnection.setPassword(eipPasswd);
eipConnection.setConnectToIKF(true);
eipConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
eipConnection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);

adapter1.setContentProvider(new CMBDefaultContentProvider());
adapter1.setCatalogName(catalog);

```

```

catalogService.setCatalogName(catalog);
adapter2.setCatalogName(catalog);

// ligar beans
samplConnection.addCMBConnectionReplyListener(queryService);
samplConnection.addCMBConnectionReplyListener(searchResults);

eipConnection.addCMBConnectionReplyListener(adapter1);
eipConnection.addCMBConnectionReplyListener(languageIdentificationService);
eipConnection.addCMBConnectionReplyListener(summarizationService);
eipConnection.addCMBConnectionReplyListener(catalogService);
eipConnection.addCMBConnectionReplyListener(adapter2);
eipConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
eipConnection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);

queryService.addCMBSearchReplyListener(searchResults);
searchResults.addCMBResultListener(adapter1);
adapter1.addCMBTextAnalysisRequestListener(languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener(summarizationService);
summarizationService.addCMBStoreRecordRequestListener(catalogService);
catalogService.addCMBStoreRecordReplyListener(adapter2);

adapter2.addCMBResultListener(this);

samplConnection.addCMBExceptionListener(this);
eipConnection.addCMBExceptionListener(this);
queryService.addCMBExceptionListener(this);
searchResults.addCMBExceptionListener(this);
languageIdentificationService.addCMBExceptionListener(this);
summarizationService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);
adapter1.addCMBExceptionListener(this);
adapter2.addCMBExceptionListener(this);

// executar consulta
samplConnection.connect();
eipConnection.connect();
catalogService.setDefaultCategoryPath(catalogService.getTaxonomy().
    getRootCategory().getPathAsString());

CMBSchemaManagement schema = samplConnection.getSchemaManagement();
CMBSearchTemplate searchTemplate = schema.getSearchTemplate(SEARCH_TEMPLATE);
String[] searchValues = { SEARCH_VALUE };
searchTemplate.setSearchCriterion(SEARCH_CRITERION, CMBBaseConstant.CMB_OP_EQUAL,
    searchValues);

CMBSearchRequestEvent searchRequest = new CMBSearchRequestEvent
    (this, CMBSearchRequestEvent.CMB_REQUEST_SEARCH_SYNCH,
        searchTemplate);
queryService.onCMBSearchRequest(searchRequest);

samplConnection.disconnect();
eipConnection.disconnect();
}

// implementar com.ibm.mm.beans.CMBResultListener
public void onCMBResult(CMBResultEvent e)
{
    if(e.getID() == CMBResultEvent.CMB_RESULT_CLEARED)
        return;

    Vector cmbItemVector = (Vector)e.getData();

    if(cmbItemVector == null)
        return;
}

```



```

try {
    for(int i = 0; i < cmbItemVector.size(); i++)
    {
        CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

        CMBRecord currentRecord = currentItem.getInfoMiningRecord();

        System.out.println("\n\nPID      : " + currentRecord.getPID());
        System.out.println("Summary   : " + currentRecord.getValue("IKF_SUMMARY"));
    } /* for */
}
catch (CMBNoSuchKeyException nske)
{
    nske.printStackTrace();
}

}

//implementar com.ibm.mm.beans.CMBExceptionListener
public void onCMBException(CMBExceptionEvent e)
{
    ((Exception)e.getData()).printStackTrace();
}

public static void main(String[] args)
{
    try
    {
        new Summarization();
        System.exit(0);
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}
}

```

O exemplo seguinte ilustra o fluxo de eventos entre os beans.

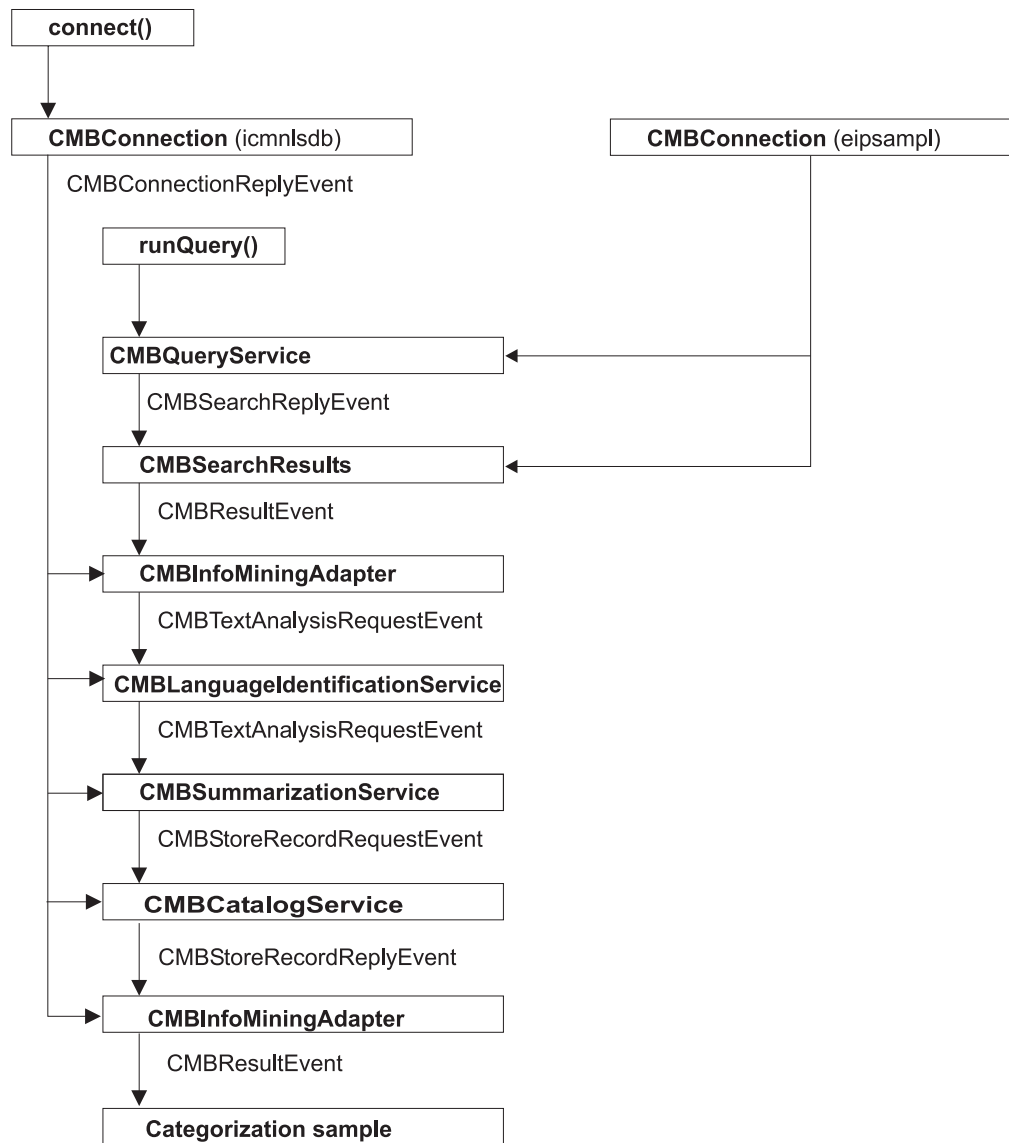


Figura 44. O exemplo de resumo: Fluxo de eventos

Extrair informações

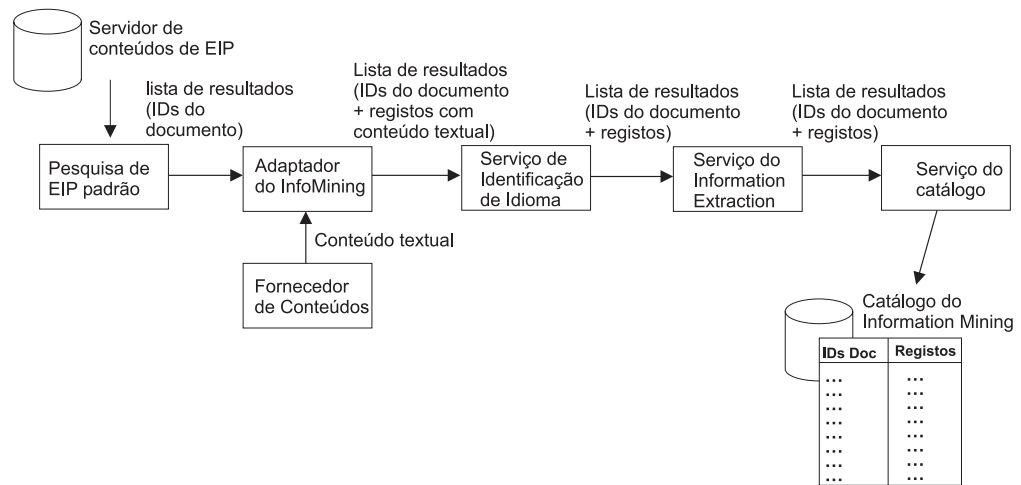


Figura 45. O exemplo de extracção de informações

Este exemplo demonstra como o utilizador pode extrair informações como, por exemplo, nomes, termos e expressões, de documentos que foram obtidos por uma pesquisa padrão de EIP. Os resultados da análise são armazenados e podem ser utilizados para obter documentos relacionados ou extrair informações importantes de documentos individuais.

Na Figura 45, uma pesquisa do EIP padrão é efectuada em documentos retidos no servidor de conteúdos do EIP. O bean de serviço de identificação de idioma determina o idioma no qual os documentos estão escritos. O bean utiliza os IDs de documento para obter o conteúdo do documento e, de seguida, analisa o conteúdo para determinar o idioma. O bean de serviço de extracção de informações recorre aos documentos em Inglês, extrai informações destes documentos e armazena os resultados no armazenamento de metadados. Estas informações, os IDs dos documentos e as informações extraídas são depois disponibilizadas para um processamento posterior.

Os beans seguintes são utilizados neste exemplo:

- CMBConnection
- CMBQueryService (para executar a pesquisa do EIP padrão)
- CMBSearchResults (para executar a pesquisa do EIP padrão)
- CMBInfoMiningAdapter
- CMBLanguageIdentificationService
- CMBInformationExtractionService
- CMBCatalogService

Para este exemplo, a aplicação:

1. Cria os beans.
2. Personaliza os beans.
3. Liga os beans de forma a que o serviço de identificação de idioma possa analisar os documentos. Os resultados são armazenados no catálogo.
4. O serviço de extracção de informações analisa os documentos. Os resultados são armazenados no catálogo.

5. Apresenta os resultados da pesquisa (listas de documentos) e informações extraídas para verificação.

A fonte Java é a seguinte. Visto que os beans de extracção de informações e de categorização funcionam de forma semelhante, consulte “Categorizar documentos” na página 461 para obter mais detalhes.

Fonte completa para InformationExtraction.java

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBQueryService;
import com.ibm.mm.beans.CMBSearchResults;
import com.ibm.mm.beans.CMBSchemaManagement;
import com.ibm.mm.beans.CMBSearchTemplate;
import com.ibm.mm.beans.CMBItem;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBBaseConstant;
import com.ibm.mm.beans.CMBSearchRequestEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBRecord;
import com.ibm.mm.beans.infomining.CMBDefaultContentProvider;
import com.ibm.mm.beans.infomining.CMBLanguageIdentificationService;
import com.ibm.mm.beans.infomining.CMBInformationExtractionService;
import com.ibm.mm.beans.infomining.CMBCatalogService;
import com.ibm.mm.beans.infomining.CMBInfoMiningAdapter;
import com.ibm.mm.beans.infomining.CMBNoSuchKeyException;

public class InformationExtraction implements
    CMBResultListener, CMBExceptionListener
{
    String DEFAULT_CMDBNAME      = "icmnlbdb";
    String DEFAULT_CMDBUSER      = "icmadmin";
    String DEFAULT_CMDBPASSWORD = "password";
    String DEFAULT_SAMPDBNAME    = "eipsampl";
    String DEFAULT_SAMPDBUSER    = "icmadmin";
    String DEFAULT_SAMPDBPASSWORD = "password";

    String CATALOG_NAME          = "Sample";
    String SEARCH_TEMPLATE       = "SearchLongBySource";
    String SEARCH_VALUE          = "ibmpress";
    String SEARCH_CRITERION      = "source";

    public InformationExtraction() throws Throwable
    {
        // criar beans
        CMBConnection samplConnection = new CMBConnection();
        CMBConnection eipConnection = new CMBConnection();
        CMBQueryService queryService = new CMBQueryService();
        CMBSearchResults searchResults = new CMBSearchResults();
        CMBLanguageIdentificationService languageIdentificationService =
            new CMBLanguageIdentificationService();
        CMBInformationExtractionService informationExtractionService =
            new CMBInformationExtractionService();
        CMBCatalogService catalogService = new CMBCatalogService();
        CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
        CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();

        // ler parâmetros
        BufferedReader din = new BufferedReader(new InputStreamReader(System.in));
```

```

System.out.print("Sample database      [" + DEFAULT_SAMPDBNAME + "] : ");
String sampDbName = din.readLine();
if (sampDbName.trim().equals("")) sampDbName = DEFAULT_SAMPDBNAME;

System.out.print("User ID for " + sampDbName +
                " [" + DEFAULT_SAMPDBUSER + "] : ");
String sampUserName = din.readLine();
if (sampUserName.trim().equals("")) sampUserName = DEFAULT_SAMPDBUSER;

System.out.print("Password for " + sampDbName +
                " [" + DEFAULT_SAMPDBPASSWORD + "] : ");
String sampPasswd = din.readLine();
if (sampPasswd.trim().equals("")) sampPasswd = DEFAULT_SAMPDBPASSWORD;

System.out.print("EIP database      [" + DEFAULT_CMDBNAME + "] : ");
String eipDbName = din.readLine();
if (eipDbName.trim().equals("")) eipDbName = DEFAULT_CMDBNAME;

System.out.print("User ID for " + eipDbName +
                " [" + DEFAULT_CMDBUSER + "] : ");
String eipUserName = din.readLine();
if (eipUserName.trim().equals("")) eipUserName = DEFAULT_CMDBUSER;

System.out.print("Password for " + eipDbName +
                " [" + DEFAULT_CMDBPASSWORD + "] : ");
String eipPasswd = din.readLine();
if (eipPasswd.trim().equals("")) eipPasswd = DEFAULT_CMDBPASSWORD;

System.out.print("Catalog      [" + CATALOG_NAME + "] : ");
String catalog = din.readLine();
if (catalog.trim().equals("")) catalog = CATALOG_NAME;

System.out.println("\n\nExecutar InformationExtraction
                    com as seguintes definições:");
System.out.println("Sample database      = " + sampDbName);
System.out.println("User ID for " + sampDbName + " = " + sampUserName);
System.out.println("Password for " + sampDbName + " = " + sampPasswd);
System.out.println("EIP Database      = " + eipDbName);
System.out.println("User ID for " + eipDbName + " = " + eipUserName);
System.out.println("Password for " + eipDbName + " = " + eipPasswd);
System.out.println("Catalog      = " + catalog + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == 'n') System.exit(0); }
while (key != 'y');

// personalizar beans
samp1Connection.setServerName(sampDbName);
samp1Connection.setUserid(sampUserName);
samp1Connection.setPassword(sampPasswd);
samp1Connection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
samp1Connection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

eipConnection.setServerName(eipDbName);
eipConnection.setUserid(eipUserName);
eipConnection.setPassword(eipPasswd);
eipConnection.setConnectToIKF(true);
eipConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
eipConnection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

adapter1.setContentProvider(new CMBDefaultContentProvider());

```

```

adapter1.setCatalogName(catalog);
catalogService.setCatalogName(catalog);
adapter2.setCatalogName(catalog);

// ligar beans
samplConnection.addCMBConnectionReplyListener(queryService);
samplConnection.addCMBConnectionReplyListener(searchResults);

eipConnection.addCMBConnectionReplyListener(adapter1);
eipConnection.addCMBConnectionReplyListener
    (languageIdentificationService);
eipConnection.addCMBConnectionReplyListener
    (informationExtractionService);
eipConnection.addCMBConnectionReplyListener(catalogService);
eipConnection.addCMBConnectionReplyListener(adapter2);

queryService.addCMBSearchReplyListener(searchResults);
searchResults.addCMBResultListener(adapter1);
adapter1.addCMBTextAnalysisRequestListener(languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener
    (informationExtractionService);
informationExtractionService.addCMBStoreRecordRequestListener
    (catalogService);
catalogService.addCMBStoreRecordReplyListener(adapter2);

adapter2.addCMBResultListener(this);

samplConnection.addCMBExceptionListener(this);
eipConnection.addCMBExceptionListener(this);
queryService.addCMBExceptionListener(this);
searchResults.addCMBExceptionListener(this);
languageIdentificationService.addCMBExceptionListener(this);
informationExtractionService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);
adapter1.addCMBExceptionListener(this);
adapter2.addCMBExceptionListener(this);

// executar consulta
samplConnection.connect();
eipConnection.connect();
catalogService.setDefaultCategoryPath(catalogService.getTaxonomy().
    getRootCategory().getPathAsString());

CMBSchemaManagement schema = samplConnection.getSchemaManagement();
CMBSearchTemplate searchTemplate = schema.
    getSearchTemplate(SEARCH_TEMPLATE);
String[] searchValues = { SEARCH_VALUE };
searchTemplate.setSearchCriterion(SEARCH_CRITERION,
    CMBBaseConstant.CMB_OP_EQUAL, searchValues);

CMBSearchRequestEvent searchRequest = new CMBSearchRequestEvent(this,
    CMBSearchRequestEvent.CMB_REQUEST_SEARCH_SYNCH, searchTemplate);
queryService.onCMBSearchRequest(searchRequest);

samplConnection.disconnect();
eipConnection.disconnect();
}

// implementar com.ibm.mm.beans.CMBResultListener
public void onCMBResult(CMBResultEvent e)
{
    if(e.getID() == CMBResultEvent.CMB_RESULT_CLEARED)
        return;

    Vector cmbItemVector = (Vector)e.getData();

    if(cmbItemVector == null)

```

```

        return;

    try {
        for(int i = 0; i < cmbItemVector.size(); i++)
        {
            CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

            CMBRecord currentRecord = currentItem.getInfoMiningRecord();

            System.out.println("\n\nPID          : " + currentRecord.getPID());
            System.out.println("Features      : " + currentRecord.getValue("IKF_FEATURES"));
        } /* for */
    }
    catch (CMBNoSuchKeyException nske)
    { nske.printStackTrace();
    }
}

// implementar com.ibm.mm.beans.CMBExceptionListener
public void onCMBException(CMBExceptionEvent e)
{
    ((Exception)e.getData()).printStackTrace();
}

public static void main(String[] args)
{
    try
    {
        new InformationExtraction();
        System.exit(0);
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}
}

```

A Figura 46 na página 480 ilustra o fluxo de eventos entre os beans.

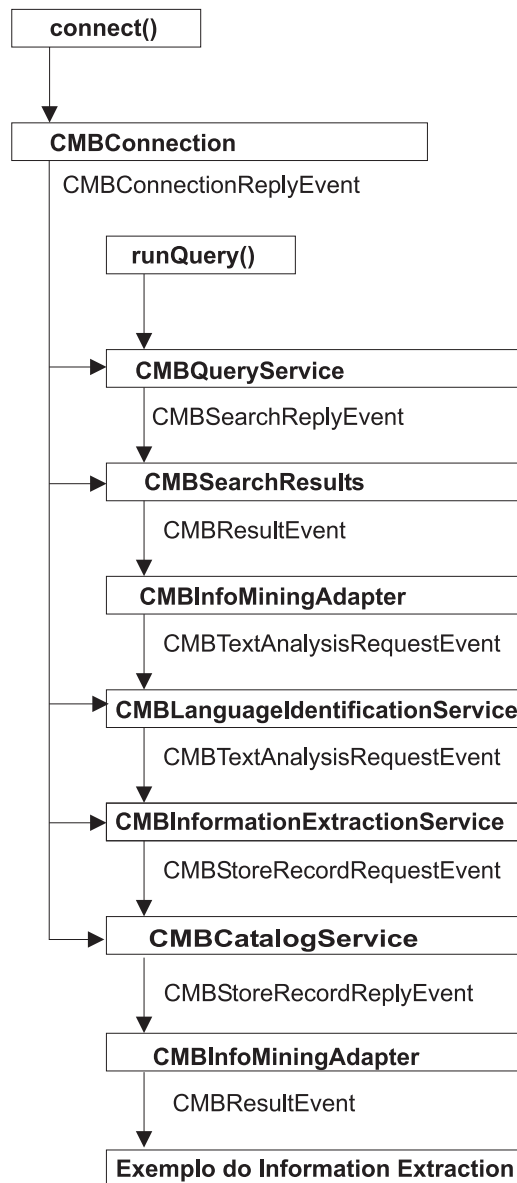


Figura 46. O exemplo de extracção de informações: Fluxo de eventos

Agrupamento

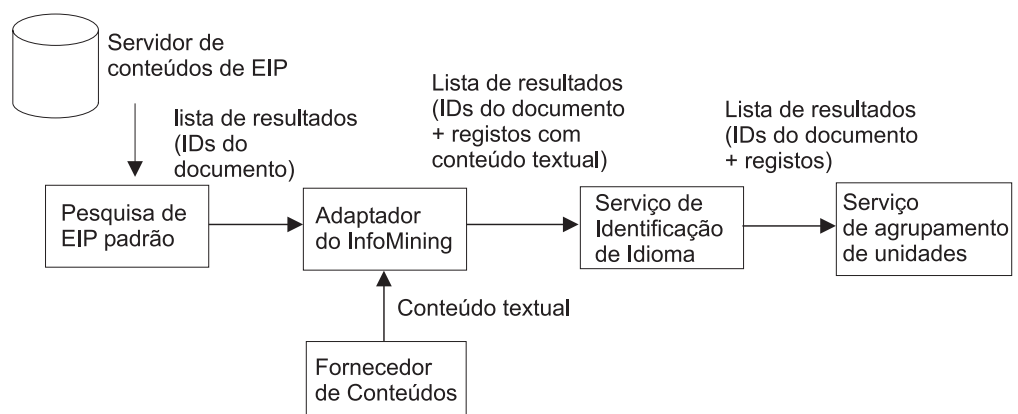


Figura 47. O exemplo de agrupamento

Este exemplo demonstra como o utilizador pode agrupar documentos que foram obtidos por uma pesquisa padrão de EIP. Antes de realizar o agrupamento, é necessário identificar o idioma dos documentos. O serviço de agrupamento só recolhe automaticamente documentos devolvidos por uma pesquisa de EIP. O processo de agrupamento tem de ser activado manualmente chamando o método `cluster()`.

Os beans seguintes são utilizados neste exemplo:

- `CMBConnection`
- `CMBQueryService` (para executar a pesquisa do EIP padrão)
- `CMBSearchResults` (para executar a pesquisa do EIP padrão)
- `CMBInfoMiningAdapter`
- `CMBLanguageIdentificationService`
- `CMBClusteringService`

Para este exemplo, a aplicação:

1. Cria os beans.
2. Personaliza os beans.
3. Liga os beans de forma a que o serviço de Identificação de Idioma possa analisar os documentos e identificar o idioma do documento e para que o serviço de Agrupamento possa recolher os documentos prontos para serem agrupados subsequentemente.
4. Activa o agrupamento chamando o método `cluster()`. O resultado (de classe `CMBClusterResult`) é então apresentado no monitor num formato legível.

Os beans do serviço de Agrupamento são semelhantes aos outros beans de Information Mining, exceptuando que:

- O serviço de agrupamento só recolhe documentos devolvidos por uma pesquisa de EIP. O processo de agrupamento tem de ser activado manualmente.
- O catálogo não é designado para armazenar resultados de agrupamento, o que significa que o serviço de catálogo não é necessário nesta hipótese.

Consulte “Categorizar documentos” na página 461 para obter informações sobre a forma como os beans funcionam.

A fonte Java The Java source é a seguinte.

Fonte completa para Clustering.java

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBQueryService;
import com.ibm.mm.beans.CMBSearchResults;
import com.ibm.mm.beans.CMBSchemaManagement;
import com.ibm.mm.beans.CMBSearchTemplate;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBBaseConstant;
import com.ibm.mm.beans.CMBSearchRequestEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBDefaultContentProvider;
import com.ibm.mm.beans.infomining.CMBLanguageIdentificationService;
```

```

import com.ibm.mm.beans.infomining.CMBClusteringService;
import com.ibm.mm.beans.infomining.CMBInfoMiningAdapter;
import com.ibm.mm.beans.infomining.CMBClusterResult;
import com.ibm.mm.beans.infomining.CMBClusterNode;

public class Clustering implements CMBResultListener,
    CMBExceptionListener
{
    String DEFAULT_CMDBNAME      = "icmnlsdb";
    String DEFAULT_CMDBUSER      = "icmadmin";
    String DEFAULT_CMDBPASSWORD = "password";
    String DEFAULT_SAMPDBNAME    = "eipsamp1";
    String DEFAULT_SAMPDBUSER    = "icmadmin";
    String DEFAULT_SAMPDBPASSWORD = "password";

    String CATALOG_NAME          = "Sample";
    String SEARCH_TEMPLATE       = "SearchLongBySource";
    String SEARCH_VALUE          = "ibmpress";
    String SEARCH_CRITERION      = "source";

    public Clustering() throws Throwable
    {
        // criar beans
        CMBClusteringService clusteringService = new CMBClusteringService();
        CMBInfoMiningAdapter adapter = new CMBInfoMiningAdapter();
        CMBQueryService queryService = new CMBQueryService();
        CMBSearchResults searchResults = new CMBSearchResults();
        CMBLanguageIdentificationService languageIdentificationService =
            new CMBLanguageIdentificationService();

        // ler parâmetros
        BufferedReader din = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Sample database      [" + DEFAULT_SAMPDBNAME + "] : ");
        String sampDbName = din.readLine();
        if (sampDbName.trim().equals("")) sampDbName = DEFAULT_SAMPDBNAME;

        System.out.print("User ID for " + sampDbName +
            " [" + DEFAULT_SAMPDBUSER + "] : ");
        String sampUserName = din.readLine();
        if (sampUserName.trim().equals("")) sampUserName = DEFAULT_SAMPDBUSER;

        System.out.print("Password for " + sampDbName +
            " [" + DEFAULT_SAMPDBPASSWORD + "] : ");
        String sampPasswd = din.readLine();
        if (sampPasswd.trim().equals("")) sampPasswd = DEFAULT_SAMPDBPASSWORD;

        System.out.print("EIP database      [" + DEFAULT_CMDBNAME + "] : ");
        String eipDbName = din.readLine();
        if (eipDbName.trim().equals("")) eipDbName = DEFAULT_CMDBNAME;

        System.out.print("User ID for " + eipDbName +
            " [" + DEFAULT_CMDBUSER + "] : ");
        String eipUserName = din.readLine();
        if (eipUserName.trim().equals("")) eipUserName = DEFAULT_CMDBUSER;

        System.out.print("Password for " + eipDbName +
            " [" + DEFAULT_CMDBPASSWORD + "] : ");
        String eipPasswd = din.readLine();
        if (eipPasswd.trim().equals("")) eipPasswd = DEFAULT_CMDBPASSWORD;

        System.out.print("Catalog      [" + CATALOG_NAME + "] : ");
        String catalog = din.readLine();
        if (catalog.trim().equals("")) catalog = CATALOG_NAME;
    }
}

```

```

System.out.println("\n\nRunning Clustering with the following settings:");
System.out.println("Sample database      = " + sampDbName);
System.out.println("User ID for " + sampDbName + " = " + sampUserName);
System.out.println("Password for " + sampDbName + " = " + sampPasswd);
System.out.println("EIP Database      = " + eipDbName);
System.out.println("User ID for " + eipDbName + " = " + eipUserName);
System.out.println("Password for " + eipDbName + " = " + eipPasswd);
System.out.println("Catalog          = " + catalog + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == 'n') System.exit(0); }
while (key != 'y');

// personalizar beans
samplConnection.setServerName(sampDbName);
samplConnection.setUserid(sampUserName);
samplConnection.setPassword(sampPasswd);
samplConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
samplConnection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);

eipConnection.setServerName(eipDbName);
eipConnection.setUserid(eipUserName);
eipConnection.setPassword(eipPasswd);
eipConnection.setConnectToIKF(true);
eipConnection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
eipConnection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);

adapter.setContentProvider(new CMBDefaultContentProvider());
adapter.setCatalogName(catalog);
clusteringService.setMinClusterCount(2);
clusteringService.setMaxClusterCount(6);
clusteringService.setClusterFeatureCount(5);

// ligar beans
samplConnection.addCMBConnectionReplyListener(queryService);
samplConnection.addCMBConnectionReplyListener(searchResults);

eipConnection.addCMBConnectionReplyListener(adapter);
eipConnection.addCMBConnectionReplyListener(languageIdentificationService);
eipConnection.addCMBConnectionReplyListener(clusteringService);

queryService.addCMBSearchReplyListener(searchResults);
searchResults.addCMBResultListener(adapter);
adapter.addCMBTextAnalysisRequestListener(languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener(clusteringService);

samplConnection.addCMBExceptionListener(this);
eipConnection.addCMBExceptionListener(this);
queryService.addCMBExceptionListener(this);
searchResults.addCMBExceptionListener(this);
languageIdentificationService.addCMBExceptionListener(this);
clusteringService.addCMBExceptionListener(this);
adapter.addCMBExceptionListener(this);

// executar consulta
samplConnection.connect();
eipConnection.connect();

CMBSchemaManagement schema = samplConnection.getSchemaManagement();
CMBSearchTemplate searchTemplate =
    schema.getSearchTemplate(SEARCH_TEMPLATE);
String[] searchValues = { SEARCH_VALUE };
searchTemplate.setSearchCriterion(SEARCH_CRITERION,

```

```

CMBBaseConstant.CMB_OP_EQUAL, searchValues);

CMBSearchRequestEvent searchRequest = new CMBSearchRequestEvent
(this, CMBSearchRequestEvent.CMB_REQUEST_SEARCH_SYNCH, searchTemplate);
queryService.onCMBSearchRequest(searchRequest);

// executar agrupamento e imprimir resultados
CMBClusterResult clusterResult = clusteringService.cluster();

System.out.println(clusterResult.getClusterCount() +
    " clusters found for " +
    clusterResult.getDocumentCount() +
    " documents:");
CMBClusterNode[] clusterNodes = clusterResult.getClusterNodes();
for(int i = 0; i < clusterNodes.length; i++) {
    CMBClusterNode node = clusterNodes[i];
    String[] features = node.getClusterFeatures();
    String[] names = node.getDocumentNames();
    String label = node.getLabel();
    System.out.println("Cluster " + label);
    System.out.print(" Cluster Features : ");
    for(int j = 0; j < features.length; j++) System.out.print(features[j] + " ");
    System.out.println();
    System.out.println(" Documents in cluster :");
    for(int j = 0; j < names.length; j++) System.out.println(" " + names[j]);
}

// desligar
samplConnection.disconnect();
eipConnection.disconnect();
}

// implementar com.ibm.mm.beans.CMBResultListener
public void onCMBResult(CMBResultEvent e)
{
    return;
}

// implementar com.ibm.mm.beans.CMBExceptionListener
public void onCMBException(CMBExceptionEvent e)
{
    ((Exception)e.getData()).printStackTrace();
}

public static void main(String[] args)
{
    try
    {
        new Clustering();
        System.exit(0);
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}
}

```

A Figura 48 na página 485 ilustra o fluxo de eventos entre os beans.

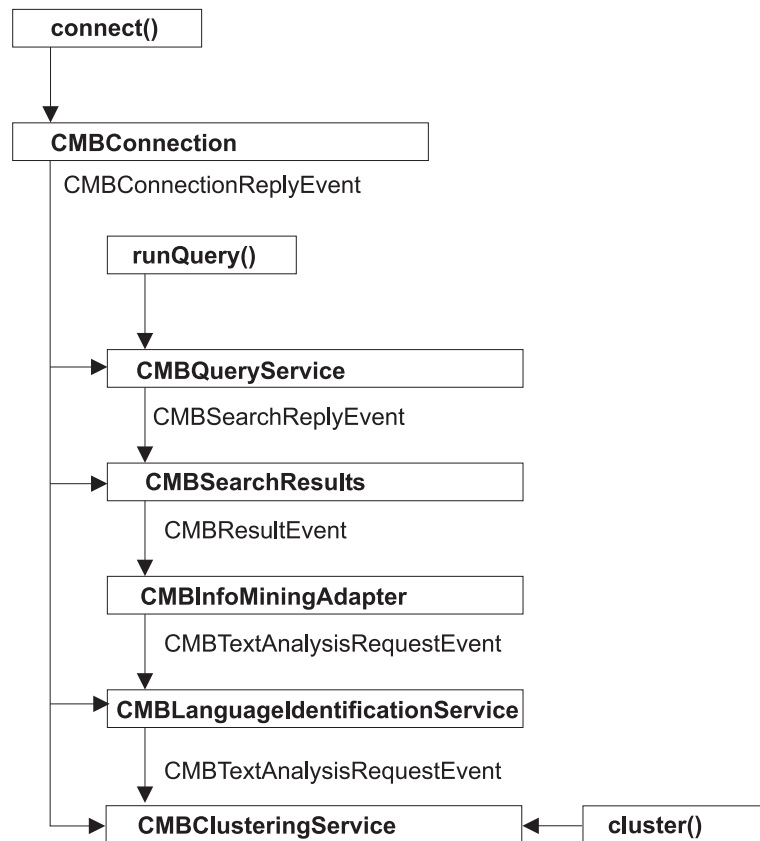


Figura 48. O exemplo de agrupamento: Fluxo de eventos

Importar documentos a partir de um espaço da web

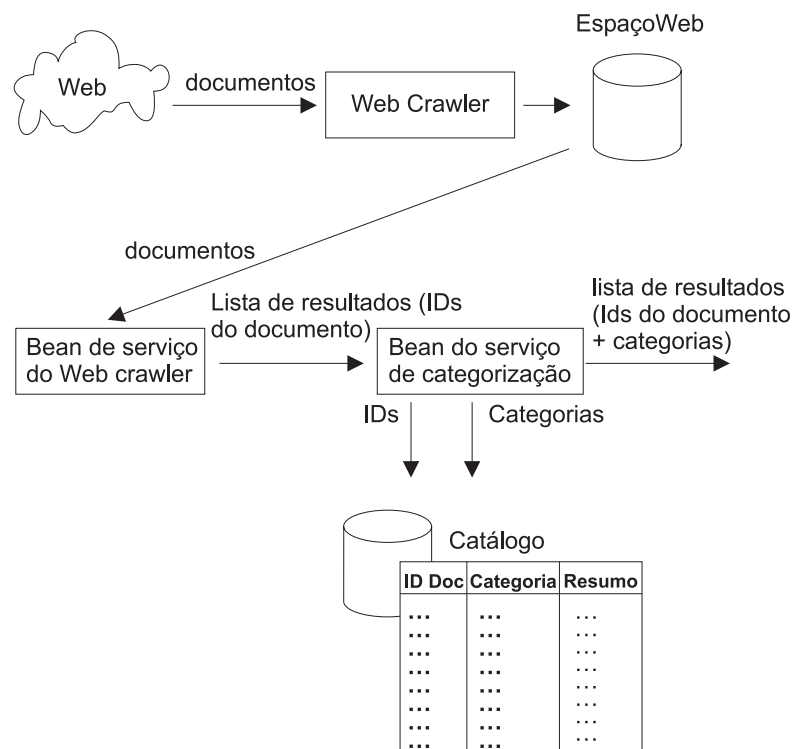


Figura 49. O exemplo de Web Crawler

Este exemplo demonstra como importar documentos buscados para o componente de Information Mining e efectuar análise de texto (identificação de idioma, categorização e resumo) em documentos que foram importados. Esta acção só pode ser realizada se o Web Crawler tiver sido executado ou esteja em execução, e se objectos novos ou alterados forem guardados no directório de espaço da Web especificado, definido pelo atributo `summaries-dir` na configuração do Web Crawler.

Ao utilizar o Web Crawler para extrair informações, certifique-se de que o executa com as definições de configuração de extracção de informações descritas na secção relativa à gestão de extracção de informações em *Gerir o Enterprise Information Portal*. Estas definições fazem parte de um ficheiro de configuração exemplo para o componente de extracção de informações `im-crawler-config-sample.xml` localizado no seguinte directório:

- Em Windows:
`<CMBROOT>\samples\java\beans\infomining\webcrawler\`
- Em UNIX (AIX e Solaris):
`<CMBROOT>/samples/java/beans/infomining/webcrawler/`

Após o Web Crawler ter colocado uma cópia dos documentos de página da web que tem supervisionado no directório de espaço da Web especificado, pode utilizar o bean `CMBWebCrawlerService` para importar os documentos nos quais foi efectuado o `crawl` para Information Mining de Enterprise Information Portal.

O Web Crawler e o bean `CMBWebCrawlerService` podem ser executados como processos permanentes para supervisionar e importar documentos à medida que são alterados. Um `CMBResultEvent` pode ser activado quando o número de documentos importados exceder um determinado valor ou quando todos os ficheiros supervisionados tiverem sido importados. O bean `CMBWebCrawlerService` notifica todos os receptores anexados. O evento contém os ficheiros importados como um vector de `CMBItems`.

Direitos de acesso a `CMBWebCrawlerService` em AIX e Solaris. Ao importar documentos utilizando `CMBWebCrawlerService`, os ficheiros correspondentes são eliminados do sistema de ficheiros ou, caso a opção de arquivamento seja activada utilizando `archiveEnabled`, os ficheiros são movidos do directório de discos para um directório de arquivos. O directório de arquivos localiza-se no mesmo nível do directório de discos (em `.../webspaces/ikf`) e possui a mesma estrutura de sub-directório do directório de discos. Para utilizar `CMBWebCrawlerService`, certifique-se de que o ID de utilizador correspondente possui as seguintes permissões:

- Permissão de escrita no directório de discos e todos os sub-directórios e ficheiros sob este directório. A localização do directório de discos é `.../webspaces/ikf/disks`, tal como está definido no ficheiro de configuração `summaries-dir` do Web Crawler. Esta permissão é necessária para eliminar ou mover ficheiros e documentos nos quais foi efectuado o `crawl`.
- Quando o arquivamento está desactivado, escreva a permissão no directório `.../webspaces/ikf`, de modo a que o directório de arquivos possa ser criado na primeira vez que o arquivamento é utilizado.
- Quando o arquivamento está activado, escreva a permissão no directório de arquivos, nos seus sub-directórios e em todos os ficheiros, de modo a que os ficheiros que são movidos do directório de discos possam ser criados no directório de arquivos.

Modificar a operação de importação. Pode alterar alguns aspectos da operação de importação no bean CMBWebCrawlerService. Inicialmente, o programa procura documentos obtidos através da sequência de hiper-ligações na Web efectuada de 30 em 30 minutos mas pode alterar este valor. Também emite uma notificação CMBResultEvent a todos os receptores a cada 100 documentos importados. Pode alterar este valor para um número de documentos diferente ou fazer com que uma notificação seja emitida quando todos os ficheiros obtidos através da sequência de hiper-ligações na Web tiverem sido importados para o conjunto do Enterprise Information Portal.

Propriedades do bean de Java CMBWebCrawlerService:

Ciclos de chamadas selectivas

Número de vezes a efectuar chamadas selectivas.

Número de minutos para chamadas selectivas

Número de minutos de espera antes da chamada selectiva seguinte. A predefinição é 30.

Directório inicial

Para buscar %IMY_WEBSPACE% no sistema central local.

archiveEnabled

Mantenha os ficheiros nos quais foi efectuado o crawl em *.../webspaces/ikf/archives*. A predefinição é falsa, o que significa que CMBWebCrawlerService elimina os ficheiros nos quais foi efectuado o crawl de *../webspaces/ikf/disks* durante o processamento, sem criar uma cópia de salvaguarda noutra local.

Tamanho da página

Número de artigos importados até o CMBResultEvent ser emitido para os CMBResultListeners.

Espaço da Web

O espaço da Web que é supervisionado pelo Web Crawler.

Tal como os restantes beans de análise de texto, os resultados da pesquisa do Web Crawler podem ser disponibilizados para uma posterior pesquisa avançada.

Os beans seguintes são utilizados neste exemplo:

- CMBConnection
- CMBWebCrawlerService
- CMBInfoMiningAdapter
- CMBLanguageIdentification
- CMBCategorizationService
- CMBSummarizationService
- CMBCatalogService

Para este exemplo a aplicação:

1. Cria os beans
2. Personaliza os beans
3. Liga os beans
4. Inicia o serviço do Web Crawler
5. Apresenta os resultados do crawler e os resultados da análise de texto

Segue-se uma explicação para cada um dos passos anteriores na origem de WebCrawler.java.

Origem completa para WebCrawler.java

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBItem;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBRecord;
import com.ibm.mm.beans.infomining.CMBWebCrawlerService;
import com.ibm.mm.beans.infomining.CMBLanguageIdentificationService;
import com.ibm.mm.beans.infomining.CMBCategorizationService;
import com.ibm.mm.beans.infomining.CMBSummarizationService;
import com.ibm.mm.beans.infomining.CMBCatalogService;
import com.ibm.mm.beans.infomining.CMBInfoMiningAdapter;
import com.ibm.mm.beans.infomining.CMBNoSuchKeyException;

public class WebCrawler implements CMBResultListener, CMBExceptionListener
{
    String CMDBNAME      = "icmnlbdb";
    String CMDBUSER      = "icmadmin";
    String CMDBPASSWORD  = "password";
    String CATALOG       = "Sample";

    String WEBSpace_DIR  = ""; // definir como directório em que residem os espaços da web
    String WEBSpace_NAME = ""; // definir esta para o nome do espaço da web do utilizador

    public WebCrawler() throws Throwable
    {
        // criar beans
        CMBConnection connection = new CMBConnection();
        CMBWebCrawlerService crawlerService = new CMBWebCrawlerService();
        CMBLanguageIdentificationService languageIdentificationService = new
            CMBLanguageIdentificationService();
        CMBCategorizationService categorizationService = new CMBCategorizationService();
        CMBSummarizationService summarizationService = new CMBSummarizationService();
        CMBCatalogService catalogService = new CMBCatalogService();
        CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
        CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();

        // ler parâmetros
        BufferedReader din = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Database [" + CMDBNAME + "]      : ");
        String dbName = din.readLine();
        if (dbName.trim().equals("")) dbName = CMDBNAME;

        System.out.print("User name [" + CMDBUSER + "]      : ");
        String userName = din.readLine();
        if (userName.trim().equals("")) userName = CMDBUSER;

        System.out.print("Password [" + CMDBPASSWORD + "]    : ");
        String passwd = din.readLine();
        if (passwd.trim().equals("")) passwd = CMDBPASSWORD;

        System.out.print("WebSpace directory [" + WEBSpace_DIR + "] : ");
        String webSpaceDir = din.readLine();
        if (webSpaceDir.trim().equals("")) webSpaceDir = WEBSpace_DIR;

        System.out.print("WebSpace name [" + WEBSpace_NAME + "]      : ");
        String webSpaceName = din.readLine();
        if (webSpaceName.trim().equals("")) webSpaceName = WEBSpace_NAME;

        System.out.print("Catalog [" + CATALOG + "]          : ");
```



```

String catalog = din.readLine();
if (catalog.trim().equals("")) catalog = CATALOG;

System.out.println("\n\nExecutar Resumo com as seguintes definições:");
System.out.println("Database      = " + dbName);
System.out.println("User        = " + userName);
System.out.println("Password    = " + passwd);
System.out.println("Webpace directory = " + webpaceDir);
System.out.println("Webpace name   = " + webpaceName);
System.out.println("Catalog       = " + catalog + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == 'n') System.exit(0);
} while (key != 'y');

// personalizar beans
connection.setServerName(dbName);
connection.setUserid(userName);
connection.setPassword(passwd);
connection.setConnectToIKF(true);
connection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
connection.setServiceConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
crawlerService.setRootDirectory(webpaceDir);
crawlerService.setWebSpace(webpaceName);

adapter1.setCatalogName(catalog);
categorizationService.setCatalogName(catalog);
catalogService.setCatalogName(catalog);
adapter2.setCatalogName(catalog);

// ligar beans
connection.addCMBConnectionReplyListener(crawlerService);
connection.addCMBConnectionReplyListener(adapter1);
connection.addCMBConnectionReplyListener(languageIdentificationService);
connection.addCMBConnectionReplyListener(categorizationService);
connection.addCMBConnectionReplyListener(summarizationService);
connection.addCMBConnectionReplyListener(catalogService);
connection.addCMBConnectionReplyListener(adapter2);

crawlerService.addCMBResultListener(adapter1);
adapter1.addCMBTextAnalysisRequestListener(languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener(categorizationService);
categorizationService.addCMBTextAnalysisRequestListener(summarizationService);
summarizationService.addCMBStoreRecordRequestListener(catalogService);
catalogService.addCMBStoreRecordReplyListener(adapter2);
adapter2.addCMBResultListener(this);

connection.addCMBExceptionListener(this);
crawlerService.addCMBExceptionListener(this);
languageIdentificationService.addCMBExceptionListener(this);
categorizationService.addCMBExceptionListener(this);
summarizationService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);
adapter1.addCMBExceptionListener(this);
adapter2.addCMBExceptionListener(this);

// executar consulta
connection.connect();
catalogService.setDefaultCategoryPath
    (catalogService.getTaxonomy().getRootCategory().getPathAsString());

crawlerService.start();
connection.disconnect();
}

// implementar com.ibm.mm.beans.CMBResultListener
public void onCMBResult(CMBResultEvent e)
{

```

```

if(e.getID() == CMBResultEvent.CMB_RESULT_CLEARED)
    return;

Vector cmbItemVector = (Vector)e.getData();

if(cmbItemVector == null)
    return;

try {
    for(int i = 0; i < cmbItemVector.size(); i++)
    {
        CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

        CMBRecord currentRecord = currentItem.getInfoMiningRecord();

        System.out.println("\n\nPID      : " + currentRecord.getPID());
        System.out.println("Categories: " + currentRecord.getValue("IKF_CATEGORIES"));
        System.out.println("Summary   : " + currentRecord.getValue("IKF_SUMMARY"));
    } /* for */
}
catch (CMBNoSuchKeyException nske)
{ nske.printStackTrace();
}

//implementar com.ibm.mm.beans.CMBExceptionListener
public void onCMBException(CMBExceptionEvent e)
{
    ((Exception)e.getData()).printStackTrace();
}

public static void main(String[] args)
{
    try
    {
        new WebCrawler();
        System.exit(0);
    }
    catch(Throwable t)
    {
        t.printStackTrace();
    }
}
}

```

Criar os beans

O utilizador constrói uma ligação a um servidor de conteúdos por razões de segurança; a extracção de informações necessita de saber quem está a trabalhar com o sistema. Isto assegura que apenas os que estão registados no sistema tenham permissão para o utilizar. A ligação pode ser estabelecida através do bean `CMBConnection`. O bean `CMBWebCrawlerService` é utilizado para importar os documentos anteriormente buscados para o componente de extracção de informações e para lançar uma notificação `CMBResultEvent`. As informações de identificação de idioma, resumo e categorização são criadas utilizando os beans `CMBLanguageIdentification`, `CMBSummarizationService` e `CMBCategorizationService`. Os dois adaptadores convertem os eventos do resultado em eventos de pedido de análise de texto e, de seguida, armazenam os eventos de resposta do registo em eventos de resultado da pesquisa.

O código que cria os beans é:

```

CMBConnection connection = new CMBConnection();
CMBWebCrawlerService crawlerService = new CMBWebCrawlerService();
CMBLanguageIdentificationService languageIdentificationService =
    new CMBLanguageIdentificationService();
CMBCategorizationService categorizationService = new CMBCategorizationService();
CMBSummarizationService summarizationService = new CMBSummarizationService();

```

```
CMBCatalogService catalogService = new CMBCatalogService();
CMBInfoMiningAdapter adapter1 = new CMBInfoMiningAdapter();
CMBInfoMiningAdapter adapter2 = new CMBInfoMiningAdapter();
```

Personalizar os beans

O código demonstrado na secção de personalização tem de ser adaptado segundo a sua instalação. Tem de especificar o nome do servidor de conteúdos ao qual ligar, um ID de utilizador e a palavra-passe adequada.

No bean de serviço do Web Crawler, o directório raiz do sistema central local do utilizador, no qual o Web Crawler guardou ou alterou os objectos buscados dentro do espaço da Web, deve ser especificado juntamente com o nome de espaço da Web anteriormente definido. Antes de poder executar os beans do serviço de análise de texto e o bean do serviço de catálogos, terá de os associar a um catálogo existente.

O código que realiza a personalização para o exemplo é:

```
connection.setServerName(dbName);
connection.setUserid(userName);
connection.setPassword(passwd);
connection.setConnectToIKF(true);
connection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
connection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);
crawlerService.setRootDirectory(webSpaceDir);
crawlerService.setWebSpace(webSpaceName);

adapter1.setCatalogName(catalog);
categorizationService.setCatalogName(catalog);
catalogService.setCatalogName(catalog);
adapter2.setCatalogName(catalog);
```

Ligar os beans

Figura 50 na página 492 ilustra o fluxo de eventos entre os beans neste exemplo.

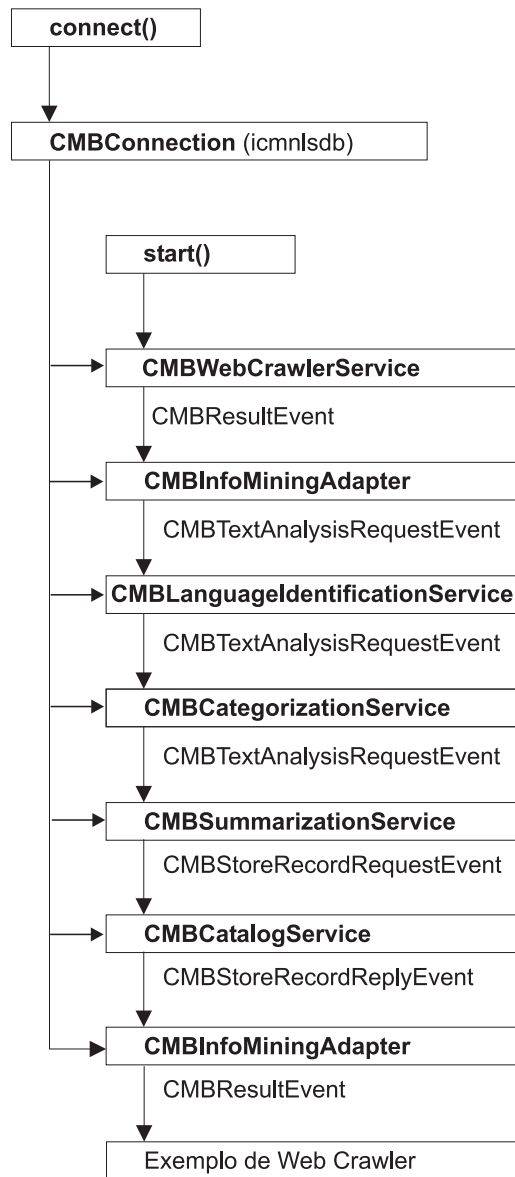


Figura 50. O exemplo de Web Crawler: Fluxo de eventos

Cinco dos beans utilizados neste exemplo aguardam o `CMBConnectionReplyEvent` para obter o parâmetro identificador da ligação. O `CMBWebCrawlerService` inicia o serviço de sequência de hiper-ligações que resulta num evento que depois inicia o fluxo de eventos através dos outros beans.

O código que liga os beans é:

```

connection.addCMBConnectionReplyListener(crawlerService);
connection.addCMBConnectionReplyListener(adapter1);
connection.addCMBConnectionReplyListener(languageIdentificationService);
connection.addCMBConnectionReplyListener(categorizationService);
connection.addCMBConnectionReplyListener(summarizationService);
connection.addCMBConnectionReplyListener(catalogService);
connection.addCMBConnectionReplyListener(adapter2);

crawlerService.addCMBResultListener(adapter1);
adapter1.addCMBTextAnalysisRequestListener(languageIdentificationService);
languageIdentificationService.addCMBTextAnalysisRequestListener(categorizationService);

```

```

categorizationService.
    addCMBTextAnalysisRequestListener(summarizationService);
summarizationService.addCMBStoreRecordRequestListener(catalogService);
catalogService.addCMBStoreRecordReplyListener(adapter2);
adapter2.addCMBResultListener(this);

connection.addCMBExceptionListener(this);
crawlerService.addCMBExceptionListener(this);
languageIdentificationService.addCMBExceptionListener(this);
categorizationService.addCMBExceptionListener(this);
summarizationService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);
adapter1.addCMBExceptionListener(this);
adapter2.addCMBExceptionListener(this);

```

Visto que as excepções são também enviadas como eventos, a classe do WebCrawler tem de manusear o evento apropriado através da implementação da interface CMBExceptionListener, estando ligada aos beans para ser notificada sobre excepções.

Iniciar o serviço do Web Crawler

Antes de poder iniciar o serviço do Web Crawler, terá de estabelecer a ligação ao servidor de conteúdos através da chamada do método de ligação no bean CMBConnection:

```
connection.connect();
```

O código para iniciar o serviço do Web Crawler é:

```

catalogService.setDefaultCategoryPath
    (catalogService.getTaxonomy().getRootCategory().getPathAsString());
crawlerService.start();

```

Para fechar a ligação actual, chame o método desligar():

```
connection.disconnect();
```

Apresentar resultados da análise de texto

A classe do WebCrawler implementa a interface CMBResultListener para poder enumerar os documentos que foram localizados durante a pesquisa e para apresentar as informações de categorias e de resumos criadas para cada documento. O CMBResultEvent, recebido como argumento no método onCMBResult, contém um vector de objectos de CMBItem, onde cada objecto de CMBItem representa um documento:

```
Vector cmbItemVector = (Vector)e.getData();
```

Um objecto de CMBItem encapsula o PID do documento:

```

CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

CMBRecord currentRecord = currentItem.getInfoMiningRecord();

System.out.println("\n\nPID      : " + currentRecord.getPID());

```

bem como os resultados da análise de texto.

Seguidamente obtém as informações relativas ao resumo e categoria:

```

System.out.println("Categories   : " +
    currentRecord.getValue("IKF_CATEGORIES"));
System.out.println("Summary      : " +
    currentRecord.getValue("IKF_SUMMARY"));

```

Pesquisar documentos por categoria

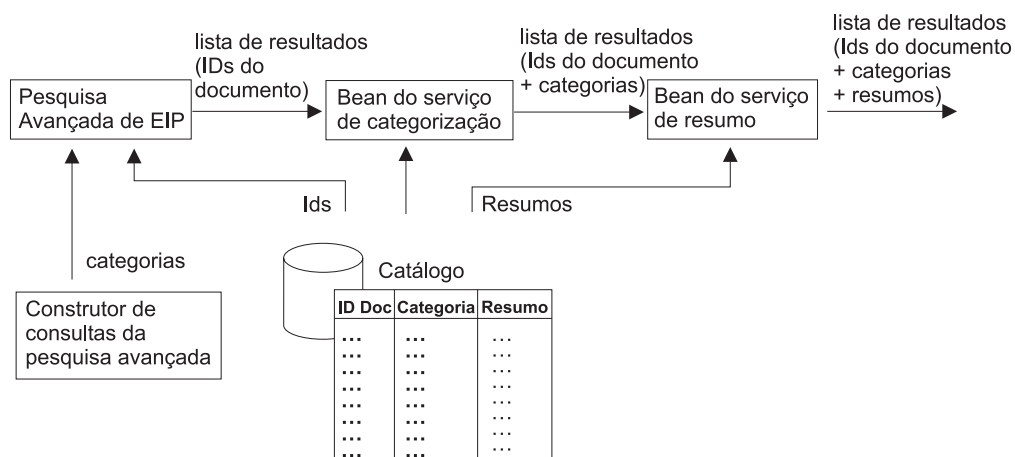


Figura 51. O exemplo de pesquisa avançada

Este exemplo demonstra como executar uma pesquisa avançada e buscar informações relativas à categoria nos documentos que foram localizados. Uma pesquisa avançada só pode localizar documentos que tenham sido disponibilizados para esse tipo de pesquisa, como no exemplo de elaboração de categorias. Consulte “Localização dos ficheiros exemplo” na página 460. Contrastando com os exemplos anteriores, em que é executada uma pesquisa padrão de EIP em todo o servidor de conteúdos de EIP, este exemplo pesquisa metadados no armazenamento de dados.

Se verificar problemas de rendimento ao enviar consultas complexas, consulte “Ajuste de rendimento” na página 509 para obter mais informações.

Quando a pesquisa avançada tiver produzido uma lista de resultados, os IDs dos documentos localizados podem ser utilizados por um bean de serviço de catálogo para obter os metadados anteriormente armazenados no armazenamento de dados.

Os beans seguintes são utilizados neste exemplo:

- CMBCConnection
- CMBAdvancedSearchService
- CMBInfoMiningAdapter
- CMBCatalogService

Para este exemplo a aplicação:

1. Cria os beans
2. Personaliza os beans
3. Liga os beans
4. Executa a pesquisa avançada
5. Apresenta os resultados da análise de texto e da pesquisa para verificação

Segue-se uma explicação de cada um dos passos anteriores depois da origem de AdvancedSearch.java.

Origem completa para AdvancedSearch.java

```
import java.util.Vector;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.ibm.mm.beans.CMBConnection;
import com.ibm.mm.beans.CMBItem;
import com.ibm.mm.beans.CMBException;
import com.ibm.mm.beans.CMBResultEvent;
import com.ibm.mm.beans.CMBExceptionEvent;
import com.ibm.mm.beans.CMBResultListener;
import com.ibm.mm.beans.CMBExceptionListener;

import com.ibm.mm.beans.infomining.CMBAdvancedSearchService;
import com.ibm.mm.beans.infomining.CMBCatalogService;
import com.ibm.mm.beans.infomining.CMBRecord;
import com.ibm.mm.beans.infomining.CMBCategory;

a classe pública AdvancedSearch implementa CMBResultListener,
CMBExceptionListener
{
    String CMDBNAME      = "icmnlbdb";
    String CMDBUSER      = "icmadmin";
    String CMDBPASSWORD = "password";

    String CATALOG      = "Sample";
    String SEARCH_TERM  = "Monitor";

    CMBCatalogService catalogService = null;

    public AdvancedSearch() throws Exception
    {
        // criar beans
        CMBConnection connection = new CMBConnection();
        CMBAdvancedSearchService advancedSearchService = new
            CMBAdvancedSearchService();
        catalogService = new CMBCatalogService();

        // ler parâmetros
        BufferedReader din = new BufferedReader
            (new InputStreamReader(System.in));

        System.out.print("Database [" + CMDBNAME + "] : ");
        String dbName = din.readLine();
        if (dbName.trim().equals("")) dbName = CMDBNAME;

        System.out.print("User name [" + CMDBUSER + "] : ");
        String userName = din.readLine();
        if (userName.trim().equals("")) userName = CMDBUSER;

        System.out.print("Password [" + CMDBPASSWORD + "] : ");
        String passwd = din.readLine();
        if (passwd.trim().equals("")) passwd = CMDBPASSWORD;

        System.out.print("Catalog [" + CATALOG + "] : ");
        String catalog = din.readLine();
        if (catalog.trim().equals("")) catalog = CATALOG;

        System.out.print("Search Term [" + SEARCH_TERM + "] : ");
        String searchTerm = din.readLine();
        if (searchTerm.trim().equals("")) searchTerm = SEARCH_TERM;

        System.out.println("\n\nExecutar Pesquisa Avançada com as seguintes definições:");
        System.out.println("Database      = " + dbName);
        System.out.println("User        = " + userName);
        System.out.println("Password    = " + passwd);
        System.out.println("Catalog     = " + catalog);
    }
}
```

```

System.out.println("Search Term = " + searchTerm + "\n");

int key;
do {
    System.out.print("Continue (y/n)? ");
    InputStreamReader inReader = new InputStreamReader(System.in);
    key = inReader.read();
    if (key == '\n') System.exit(0); }
while (key != 'y');

// personalizar beans
connection.setServerName(dbName);
connection.setUserid(userName);
connection.setPassword(passwd);
connection.setConnectToIKF(true);
connection.setConnectionType(CMBConnection.CMB_CONNTYPE_DYNAMIC);
connection.setServiceConnectionType
    (CMBConnection.CMB_CONNTYPE_DYNAMIC);

catalogService.setCatalogName(catalog);
advancedSearchService.setCatalogName(catalog);
String[] recordKeys = {"IKF_SUMMARY"};
advancedSearchService.setKeysToBeFetched(recordKeys);
advancedSearchService.setQueryString
    ("(\"IKF_CONTENT\" CONTAINS \"" + searchTerm + "\"");

// ligar beans
connection.addCMBConnectionReplyListener(catalogService);
connection.addCMBConnectionReplyListener(advancedSearchService);
advancedSearchService.addCMBResultListener(this);

connection.addCMBExceptionListener(this);
advancedSearchService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);

// executar consulta
connection.connect();
advancedSearchService.runQuery();
connection.disconnect();
}

// implementar com.ibm.mm.beans.CMBResultListener
public void onCMBResult(CMBResultEvent e)
{
    if(e.getID() == CMBResultEvent.CMB_RESULT_CLEARED)
        return;

    Vector cmbItemVector = (Vector)e.getData();

    if(cmbItemVector == null)
        return;

    try {
        for(int i = 0; i < cmbItemVector.size(); i++)
        {
            CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

            CMBRecord currentRecord = currentItem.getInfoMiningRecord();

            String summary = (String)currentRecord.getValue("IKF_SUMMARY");

            CMBCategory[] categories =
                catalogService.getCategoriesForRecord(currentItem);
            String categoryString = categories[0].getPathAsString();
            for(int j = 1; j < categories.length; j++)
            {
                categoryString += ", " + categories[j].getPathAsString();
            }
        }
    }
}

```



```

    }

    System.out.println("\n\nPID      : " + currentRecord.getPID());
    System.out.println("Categories : " + categoryString);
    System.out.println("Summary   : " +
        ((summary == null)? "n/a":summary));
    } /* for */
}
catch (CMBException ex)
{ ex.printStackTrace();
}
}

// implementar com.ibm.mm.beans.CMBExceptionListener
public void onCMBException(CMBExceptionEvent e)
{
    ((Exception)e.getData()).printStackTrace();
}

public static void main(String[] args)
{
    try
    {
        new AdvancedSearch();
        System.exit(0);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
}

```

Criar os beans

Necessita de uma ligação a um servidor de conteúdos para executar uma pesquisa. A ligação pode ser estabelecida através do bean `CMBCConnection`. O bean `CMBCAdvancedSearchService` deve executar uma pesquisa avançada. As informações de categoria podem ser obtidas utilizando o bean `CMBCCatalogService`.

O código que cria os beans é:

```

CMBCConnection connection = new CMBCConnection();
CMBCAdvancedSearchService advancedSearchService = new CMBCAdvancedSearchService();
catalogService = new CMBCCatalogService();

```

Personalizar os beans

O código demonstrado na secção de personalização tem de ser adaptado segundo a sua instalação. Tem de especificar o nome do servidor de conteúdos ao qual ligar, um ID de utilizador e a palavra-passe adequada.

Antes de poder executar os beans do serviço de pesquisa avançada e o bean do serviço de catálogos, terá de os associar a um catálogo existente.

O código que realiza a personalização para o exemplo é:

```

connection.setServerName(dbName);
connection.setUserid(userName);
connection.setPassword(passwd);
connection.setConnectToIKF(true);
connection.setConnectionType
    (CMBCConnection.CMB_CONNTYPE_DYNAMIC);
connection.setServiceConnectionType
    (CMBCConnection.CMB_CONNTYPE_DYNAMIC);

catalogService.setCatalogName(catalog);

```

```

advancedSearchService.setCatalogName(catalog);
String[] recordKeys = {"IKF_SUMMARY"};
advancedSearchService.setKeysToBeFetched(recordKeys);
advancedSearchService.setQueryString
("(\\"IKF_CONTENT\\" CONTAINS \'" + searchTerm + "\'");

```

Ligar os beans

Figura 52 ilustra o fluxo de eventos entre os beans neste exemplo.

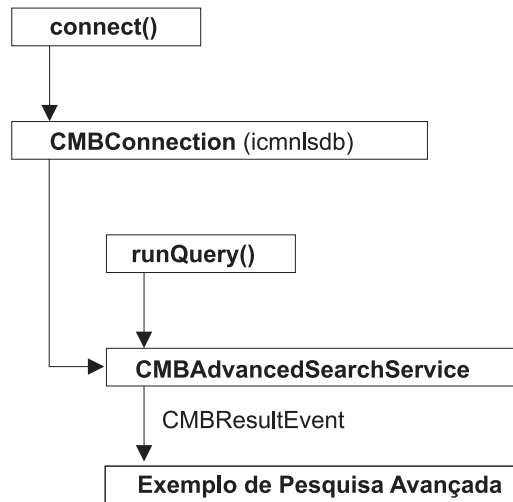


Figura 52. O exemplo de pesquisa avançada: Fluxo de eventos

Dois dos beans utilizados neste exemplo aguardam o `CMBConnectionReplyEvent` para obter o parâmetro identificador da ligação. O bean `CMBAdvancedSearchService` inicia uma pesquisa que resulta num evento que depois inicia o fluxo de eventos através dos outros beans.

O código que liga os beans é:

```

connection.addCMBConnectionReplyListener(catalogService);
connection.addCMBConnectionReplyListener(advancedSearchService);
advancedSearchService.addCMBResultListener(this);

connection.addCMBExceptionListener(this);
advancedSearchService.addCMBExceptionListener(this);
catalogService.addCMBExceptionListener(this);

```

Visto que as excepções são também enviadas como eventos, a classe de `AdvancedSearch` tem de manusear o evento apropriado através da implementação da interface `CMBExceptionListener`, estando ligada aos beans para ser notificada sobre excepções.

Executar a consulta

Antes de poder executar a consulta, terá de estabelecer a ligação ao servidor de conteúdos através da chamada do método de ligação no bean `CMBConnection`:

```

connection.connect();

```

Para iniciar a pesquisa avançada, tem de especificar um catálogo, uma cadeia de consulta de pesquisa e os metadados nos quais efectuar a pesquisa.

O utilizador inicia a pesquisa avançada:

```
advancedSearchService.runQuery();
```

Para fechar a ligação actual, chame o método desligar():

```
connection.disconnect();
```

Apresentar resultados da análise de texto

A classe `AdvancedSearch` implementa a interface `CMBResultListener` para poder enumerar os documentos que foram localizados durante a pesquisa e para apresentar as informações de categorias criadas para cada documento. O `CMBResultEvent`, recebido como argumento no método `onCMBResult`, contém um vector de objectos de `CMBItem` onde cada objecto de `CMBItem` representa um documento:

```
Vector cmbItemVector = (Vector)e.getData();
```

Um objecto de `CMBItem` encapsula o PID do documento:

```
CMBItem currentItem = (CMBItem)cmbItemVector.elementAt(i);

CMBRecord currentRecord = currentItem.getInfoMiningRecord();
String summary = (String)currentRecord.getValue("IKF_SUMMARY");

CMBCategory[] categories =
    catalogService.getCategoriesForRecord(currentItem);
String categoryString = categories[0].getPathAsString();
for(int j = 1; j < categories.length; j++)
{
    categoryString += ", " + categories[j].getPathAsString();
}

System.out.println("\n\nPID          : " + currentRecord.getPID());
```

bem como os resultados dos beans da análise de texto, se existirem alguns no fluxo de eventos.

Seguidamente obtém as informações relativas à categoria:

```
System.out.println("Categories      :
                  " + currentRecord.getValue("IKF_CATEGORIES"));
```

O vector devolvido pelo método `getCategories()` da classe `CMBItem` contém objectos da classe `CMBCategory`, que podem ser utilizados para determinar o caminho absoluto para a categoria actual.

Construir o seu fornecedor de conteúdos

O utilizador pode querer construir o seu fornecedor de conteúdos quando necessitar de aplicar filtros de proprietário, para obter texto da parte binária do formato do proprietário.

Esta secção descreve como pode escrever o seu fornecedor de conteúdos que pode lidar com um modelo de objectos definido pelo utilizador, ou com formatos de proprietário dentro das partes de um `CMBItem`. Para o auxiliar neste trabalho, o `Information Mining` fornece:

- A interface **`CMBCContentProvider`**, que define a interface para classes que sabem como determinar o texto para ser utilizado para análise de texto.
- O método **`setContentProvider(CMBCContentProvider)`** em `CMBInfoMiningUtilities` que define um FornecedorConteúdos.

A interface `CMBCContentProvider` define um método `getContent()` que devolve o texto do artigo especificado para ser utilizado para análise de texto. Por exemplo:

```
public CMBTextAnalysisDocument getContent(CMBConnection connection, CMBItem item )
throws CMBContentProviderException;
```

- O parâmetro ligação: uma ligação aberta ao servidor.
- O parâmetro artigo: o artigo actual que vai ser processado.
- A excepção CMBContentProviderException: caso ocorra um erro durante o processamento do artigo actual.
- Devolve o texto como um objecto de classe CMBTextAnalysisDocument.

Para indicar ao sistema qual o FornecedorConteúdos a utilizar, utilize o método `setContentProvider(CMBContentProvider)` na classe `CMBInfoMiningUtilities` para especificar um objecto que tenha esta interface. Aqui está um exemplo:

```
CMBInfoMiningUtilities.setContentProvider
    (new MyCompaniesLatestGreatestContentProvider());
```

Para desenvolver um fornecedor de conteúdos personalizado, o utilizador pode utilizar o fornecedor de conteúdos exemplo (`SimpleContentProvider`) como ponto de partida. O exemplo de fornecedor de conteúdos encontra-se em:

```
<CMBROOT>\samples\java\beans\infomining\contentprovider
```

É facultado um fornecedor de conteúdos predefinido com Information Mining. Irá expandir a interface `CMBContentProvider` para permitir a selecção de partes individuais para processamento e oferece um mecanismo para evitar o processamento de objectos que sejam demasiado grandes para processamento na memória, tais como sequências de vídeo.

Para registar o fornecedor de conteúdos predefinido, utilize:

```
CMBInfoMiningUtilities.setContentProvider(new CMBDefaultContentProvider());
```

Utilizar a API de serviço

A API de serviço de extracção de informações é uma API de Java que integra a funcionalidade de extracção de informações como um serviço de EIP. Ao utilizar a API de serviço de extracção de informações, o utilizador pode escrever aplicações que:

- Filtrem o conteúdo textual de diferentes formatos de documentos como, por exemplo, pdf, Microsoft Word e HTML
- Executem análise de texto nos documentos de texto para criar metadados como, por exemplo, de resumo e categorização
- Armazenem metadados para um documento em registos permanentes
- Pesquisem registos

Nota

Antes de utilizar os beans de extracção de informações, é importante que compreenda as diferenças entre a API de serviço e os JavaBeans.

- Os JavaBeans facultam uma função para um rápido desenvolvimento de aplicações. Determinados elementos do código estão 'unidos' e não podem ser utilizados pelo utilizador. Para obter mais informações, consulte "Construir uma aplicação de Information Mining (extracção de informações) utilizando os beans" na página 457.
- A API de serviço de extracção de informações faculta mais flexibilidade para a construção de aplicações de extracção de informações. O código pode ser encarado mais como "blocos de construção" individuais que podem ser unidos de modo a preencher os requisitos específicos.

Estabelecer ligação à API de serviço de extracção de informações

Antes de poder utilizar a API de serviço de extracção de informações, o utilizador tem de criar um objecto de serviço utilizando a classe `DKIKFSERVICEFed`.

Dependendo do tipo de aplicação que pretende, o utilizador terá de importar a classe de um dos seguintes três pacotes:

- Se pretende criar uma aplicação num sistema central do servidor, utilize a seguinte instrução de importação: `import com.ibm.mm.sdk.server.DKIKFSERVICEFed;`
- Se pretende criar uma aplicação no sistema central cliente, utilize the seguinte instrução de importação: `import com.ibm.mm.sdk.client.DKIKFSERVICEFed;`
- Se pretende manter a flexibilidade e criar uma aplicação que possa ser executada no servidor, bem como no cliente, utilize a seguinte instrução de importação: `import com.ibm.mm.sdk.cs.DKIKFSERVICEFed;`

Após utilizar a instrução de importação adequada, crie o objecto `ikfService` utilizando:

```
DKIKFSERVICE ikfService = DKIKFSERVICEFed.create();
```

Após criar o objecto de serviço, pode estabelecer ligação ao nome de utilização, ao ID e palavra-passe de utilizador.

```
ikfService.connect("databaseName", "userID", "password", null);
```

Já possui uma ligação à API de serviço de extracção de informações e pode utilizar classes dos dois seguintes pacotes:

- **`com.ibm.mm.sdk.common.infomining`** para gerir tarefas de biblioteca, catálogo e registo.
- **`com.ibm.mm.sdk.common.infomining.analysis`** para utilizar as diferentes ferramentas de análise de texto.

Para obter uma explicação relativa a estas funções principais, consulte as seguintes secções.

Para terminar a ligação, utilize:

```
ikfService.disconnect();
```

Gerir a biblioteca, taxonomias e catálogos

As classes para gerir as tarefas de biblioteca, catálogo e taxonomia encontram-se no pacote `com.ibm.mm.sdk.common.infomining`.

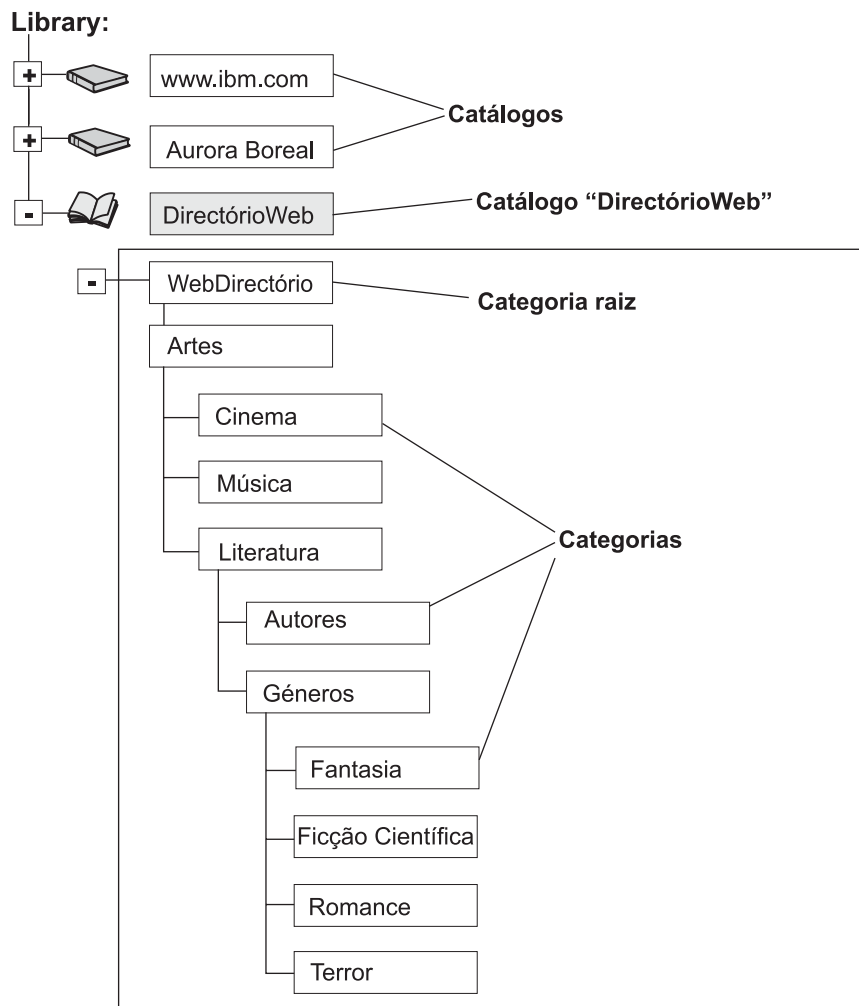


Figura 53. Um exemplo de um catálogo

No exemplo de catálogo, a Biblioteca contém três catálogos, nomeadamente, `www.ibm.com`, `Northern Light`, e `Webdirectory`. `Webdirectory` possui a estrutura em árvore de categorias apresentada no diagrama.

Os catálogos só podem ser tratados por Information Structuring Tool (IST).

Devolver uma biblioteca

Para obter um objecto de biblioteca, utilize:

```
DKIKFLibrary library = ikfService.getLibrary();
```

Enumerar todos os catálogos da biblioteca

Para obter os nomes de todos os catálogos da biblioteca, utilize:

```
String[] catalogNames = library.getCatalogNames();
```

Este irá devolver uma tabela que enumerará todos os nomes de catálogos. O exemplo devolve os seguintes nomes de catálogos, segundo uma ordem não específica: Webdirectory, www.ibm.com e Northern Light.

Devolver um catálogo específico

Para obter um catálogo específico, utilize:

```
DKIKFCatalog catalog = library.getCatalog(catalogNames[0]);
```

Este exemplo devolve o catálogo com o primeiro nome da lista devolvida no passo anterior. Neste caso, é devolvido o catálogo Webdirectory.

Devolver a taxonomia de um catálogo

Para obter a taxonomia de um catálogo, utilize:

```
DKIKFTaxonomy taxonomy = catalog.getTaxonomy();
```

O objecto de taxonomia possui vários métodos para trabalhar com as categorias.

Nota

O objecto de taxonomia é lido a partir da base de dados e todas as restantes acções são executadas nesta cópia.

Enumerar todas as categorias de uma taxonomia

Para obter todas as categorias de uma taxonomia, utilize:

```
DKIKFCategory[] categories = taxonomy.getCategories();
```

Este irá devolver uma tabela que enumerará todas as categorias, incluindo a categoria raiz. O exemplo devolve as seguintes categorias, segundo uma ordem não específica: Terror, Autor, Música, Literatura, Directório da Web, Géneros, Ficção Científica, Fantasia, Filmes, Romance e Artes.

Nota

Os objectos de categoria são lidos a partir da base de dados e todas as restantes acções são executadas na cópia.

Devolver a categoria raiz

Para obter a categoria raiz, utilize:

```
DKIKFCategory rootCategory = taxonomy.getRootCategory();
```

Este exemplo devolve a categoria raiz da lista devolvida no passo anterior. Neste caso, é devolvida a categoria raiz Directório da Web.

Devolver uma categoria específica

Para devolver uma categoria específica, utilize:

```
DKIKFCategory category = taxonomy.getCategory("Webdirectory/Literature/Genres");
```

Ao utilizar o exemplo, é devolvida a categoria Géneros.

Utilize também o mesmo carácter de operador "/" que está actualmente definido no objecto de taxonomia.

Nota

Estão disponíveis três métodos de getCategory no objecto de taxonomia, sendo que cada um necessita de parâmetros diferentes:

- O caminho é uma cadeia com nomes de categorias separadas por um carácter especificado
- O caminho é uma tabela de nomes de categorias
- Outro objecto de categoria

Consulte a *Referência online de API* para obter mais informações.

Verificar se a taxonomia actual é a mais recente

Para verificar se a taxonomia actual é a mais recente, utilize:

```
if(taxonomy.getTimestamp().before(catalog.getTaxonomyLastModified()))
{
    taxonomy = catalog.getTaxonomy();
}
```

Nota

O objecto de taxonomia é lido a partir da base de dados e todas as restantes acções são executadas nesta cópia. Consequentemente, deverá certificar-se de que o objecto de taxonomia é o mais recente.

Devolver os descendentes de uma categoria

Para devolver os descendentes de uma categoria, utilize:

```
DKIKFCategory[] children = category.getChildren();
```

Este irá devolver uma tabela com todos os descendentes da categoria. O exemplo com base na categoria Gêneros devolve os seguintes descendentes, segundo uma ordem não específica: Fantasia, Ficção científica, Romance e Terror.

Nota

Podem existir vários descendentes para uma categoria, mas não devolve mais sub-categorias destes descendentes.

Devolver um ascendente de uma categoria

Para devolver o ascendente de uma categoria:

```
DKIKFCategory parent = category.getParent();
```

O exemplo com base na categoria Gêneros apenas devolve a categoria Literatura.

Devolver um esquema de catálogo e enumerar todos os atributos do esquema

Para obter o esquema de catálogo específico, utilize:

```
DKIKFSchema schema = catalog.getSchema();
Iterator keys = schema.keySet().iterator();

while(keys.hasNext())
{
    String key = (String)keys.next();
    System.out.println("key: " + key + "type: " + schema.getType(key).getName());
}
```


O exemplo devolve o esquema do catálogo Webdirectory.

O utilizador pode devolver as chaves (nomes de atributos) do esquema. Consulte a *Referência online de API* para obter informações adicionais.

Nota

O objecto de esquema é lido a partir da base de dados e todas as restantes acções são executadas nesta cópia.

Utilizar as ferramentas de Information Mining

As classes para utilizar as ferramentas de Information Mining encontram-se no pacote **com.ibm.mm.sdk.common.infomining.analysis**. Utilize as ferramentas para:

- Gerar um resumo de documento
- Determinar a categoria do documento
- Determinar o idioma o documento
- Extrair informações como, por exemplo, nomes, termos e expressões dos documentos
- Agrupar conjuntos de documentos

As ferramentas processam documentos de texto, que são objectos de classe `DKIKFTextDocument`. Existem dois métodos de criação de um objecto de documento de texto, nomeadamente:

- Crie métodos da classe `DKIKFTextDocument` caso o documento já exista como uma cadeia de java
- Classe `DKIKFDocumentFilter`, se o conteúdo do documento se encontrar num formato formatado

Ambas as classes se encontram no pacote **com.ibm.mm.sdk.common.infomining**.

Documento de Texto

Para criar um objecto de documento de texto, caso o conteúdo do documento exista como uma cadeia de Java, utilize:

```
DKIKFTextDocument document = DKIKFTextDocument.create("the document content");
```

Nota

Estão disponíveis três métodos diferentes de criação:

- Criar um documento de texto com conteúdo especificado
- Criar um documento de texto com conteúdo e nome especificados
- Criar um documento de texto com conteúdo, nome e idioma especificados

Consulte a *Referência online de API* para obter mais informações.

Filtrar o documento

Para criar um documento de texto a partir de um documento formatado, utilize:

```
DKIKFDocumentFilter documentFilter = new DKIKFDocumentFilter(ikfService);  
byte[] documentBytes = ... //definir os bytes do documento a partir da fonte de dados  
DKIKFTextDocument document2 = documentFilter.getTextDocument(documentBytes);
```

Tenha em atenção que o documento de texto original pode apresentar-se em qualquer formato suportado como, por exemplo, um documento de pdf ou Microsoft Word.

Utilize o método do objecto de filtro de documento para definir a codificação do filtro. Consulte a *Referência online de API* para obter mais informações.

Determinar o idioma do documento

Para determinar o idioma de um documento, consulte:

```
DKIKFLanguageIdentifier languageIdentifier =  
    new DKIKFLanguageIdentifier(ikfService);  
DKIKFLanguageIdentificationResult[] languageResults =  
    languageIdentifier.analyze(document);  
document.setLanguage(languageResults[0].getLanguage());
```

Este devolve uma tabela que enumera todos os objectos que contêm resultados de idioma e de valor de fiabilidade, em que o primeiro valor de fiabilidade enumerado é o mais elevado.

Nota

Para utilizar as outras ferramentas de Information Mining como, por exemplo, o serviço de resumo e categorização, o idioma deve ser definido no documento de texto.

Utilize os métodos dos objectos de resultado de idioma para devolver valores de fiabilidade e idioma.

```
string language = languageResults[0].getLanguage();  
float confidence = languageResults[0].getConfidence();
```

Para obter mais informações relativas ao serviço de Identificação de Idioma e das predefinições, consulte a *Referência online de API*.

Gerar um resumo de documento

Para criar um resumo de documento, utilize:

```
DKIKFSummarizer summarizer = new DKIKFSummarizer(ikfService);  
DKIKFSummarizationResult summarizationResult = summarizer.analyze(document);
```

Para devolver o resumo, utilize:

```
string summary = summarizationResult.getSummary();
```

Para obter mais informações relativas ao serviço de Resumo e das predefinições, consulte a *Referência online de API*.

Extrair informações relativas ao documento

Para extrair informações relativas a um documento, utilize:

```
DKIKFInformationExtractor extractor = new DKIKFInformationExtractor(ikfService);  
DKIKFInformationExtractionResult information = extractor.analyze(document);
```

Utilize os métodos dos objectos de resultado de extracção de informações para explorar a funcionalidade analítica adicional.

```
DKIKFFeature[] features = information.getFeatures();
```

Para obter mais informações relativas ao serviço de Extracção de Informações e às predefinições, consulte a *Referência online de API*.

Agrupamento

Pode agrupar documentos, de forma a que os documentos semelhantes sejam agrupados utilizando:

```
DKIKFClusterer clusterer = new DKIKFClusterer();
cluster.analyze(doc1);
cluster.analyze(doc2);
cluster.analyze(doc3);
DKIKFClusterResult clusterResult = clusterer.cluster();
```

Utilize os métodos do objecto de resultado de agrupamento para devolver os nós de agrupamento, o número de agrupamentos e o número total de documentos.

```
int clusterCount = clusterResult.getClusterCount();
DKIKFClusterNode[] nodes = clusterResult.getClusterNodes();
int documentCount = clusterResult.getDocumentCount();
```

Para obter mais informações relativas ao serviço de Agrupamento e às predefinições, consulte a *Referência online de API*.

Categorização

Para atribuir categorias a documentos, utilize:

```
DKIKFCategorizer categorizer = new DKIKFCategorizer(catalog);
DKIKFCategorizationResult[] categorizationResults = categorizer.analyze(document);
```

Este devolve uma tabela que enumera todos os objectos de resultado de categorização que contêm categorias e valores de fiabilidade, em que o primeiro valor de fiabilidade enumerado é o mais elevado.

Utilize os métodos dos objectos de resultado de categorizador para devolver valores de fiabilidade e as categorias.

```
DKIKFCategory bestCategory = categorizationResults[0].getCategory();
```

Nota

Para utilizar o serviço de categorização, o utilizador deve criar e adaptar o catálogo utilizando Information Structuring Tool (IST). O serviço de categorização utiliza um catálogo.

Tenha também em atenção que os objectos de categoria devolvidos não pertencem ao objecto de taxonomia. Utilize os métodos de taxonomia para cruzar categorias ascendentes e descendentes.

Para obter mais informações relativas ao serviço de Categorização e às predefinições, consulte a *Referência online de API*.

Criar registos e armazenar metadados em catálogos

As classes para criar registos e armazenar metadados encontram-se no pacote **com.ibm.mm.sdk.common.infomining**. Utilize-as para:

- Criar um novo registo num catálogo
- Obter um registo de um catálogo
- Devolver as categorias para um registo
- Actualizar um valor de registo
- Actualizar a atribuição de categoria de registo
- Eliminar um registo

Tenha em atenção que a API de serviço de extracção de informações apenas pode gerir catálogos criados por Information Structuring Tool (IST). Após serem criados, os registos e metadados podem então ser adicionados.

Criar um novo registo num catálogo

Um registo é criado utilizando um PID e um esquema de catálogo:

```
DKIKFRecord record = DKIKFRecord.create("PID", schema);
record.setValue("IKF_TITLE", "Este é o título");
record.setValue("IKF_SUMMARY", "Este é o resumo do documento");
record.setValue("IKF_CONTENT", "Este é o conteúdo do documento");
DKIKFCategory[] categoriesParam = {bestCategory};
catalog.createRecord(record, categoriesParam);
```

No exemplo, estão definidos os valores de título, resumo e conteúdo do registo. O registo é criado utilizando os valores acima definidos e pode ser atribuído a uma ou mais categorias.

Obter um registo de um catálogo

Para obter um registo de um catálogo utilizando um PID, introduza:

```
DKIKFRecord retrievedRecord = catalog.getRecord("PID");
```

Devolver as categorias de um registo

Para devolver as categorias de um registo, utilize:

```
DKIKFCategory[] recordCategories = catalog.getCategoriesForRecord("PID");
```

Actualizar um valor de registo num catálogo

Para actualizar um valor de registo como, por exemplo, o título de um registo, utilize:

```
record.setValue("IKF_TITLE", "Este é o novo título");
catalog.updateRecord(record);
```

Nota

Estão disponíveis três métodos diferentes de actualização:

- Actualize o registo apenas com valores de registo (consulte o exemplo de título anterior)
- Actualize o registo com valores de registo e categorias atribuídas
- Actualize o registo apenas com categorias atribuídas (consulte o exemplo seguinte em que um registo de uma categoria é adicionado a outra categoria)

Consulte a *Referência online de API* para obter mais informações.

Adicionar um registo de uma categoria a outra categoria

Para adicionar um registo a outra categoria, utilize:

```
DKIKFCategory oldCategories = catalog.getCategoriesForRecord("PID");
DKIKFCategory newCategory = taxonomy.getCategory("Webdirectory/arts/movies");
List categoriesList = Arrays.asList(oldCategories);
categoriesList.add(newCategory);
catalog.updateRecord("PID",
    categoriesList.toArray(new DKIKFCategory[categoriesList.size()]));
```

No exemplo, o registo é adicionado a uma nova categoria Filmes. Se pretende manter as atribuições de categoria originais, deve incluir as categorias originais na chamada de registo de actualização, tal como o exemplo anterior mostra. Uma chamada de registo de actualização numa nova categoria irá remover todas as atribuições de categoria originais.

Existem três métodos para actualizar um registo. Consulte “Actualizar um valor de registo num catálogo” na página 508 para obter mais informações.

O registo pode também ser adicionado à categoria raiz e a mais do que uma categoria. Consulte a *Referência online de API* para obter mais detalhes.

Eliminar um registo armazenado

Para eliminar um registo, utilize:

```
catalog.deleteRecord("PID");
```

Pesquisar documentos

As classes para pesquisar os registos encontram-se no pacote **com.ibm.mm.sdk.common.infomining**.

Localizar registos que contêm uma palavra específica

Por exemplo, para localizar um registo que contém a palavra “Bach” na categoria “Música”, utilize:

```
String queryString = "("IKF_CONTENT\" contains \"Bach\") and  
(DKIKF_CATEGORY = \"Webdirectory/Music\");  
DKIKFSearchConfiguration searchConfiguration = new DKIKFSearchConfiguration();  
searchConfiguration.setTaxonomy(taxonomy);  
DKIKFSearchResult searchResult =  
    catalog.searchRecords(queryString, searchConfiguration);  
Iterator resultPIDs = searchResult.iterator();
```

Para executar o método `searchRecords` no objecto de catálogo, o utilizador necessita de:

- Uma cadeia de consulta e
- Um objecto de configuração de pesquisa

O objecto de configuração de pesquisa especifica propriedades de pesquisa como, por exemplo, o número máximo de resultados da pesquisa. Caso esteja a utilizar categorias na cadeia de consulta, também é necessário o objecto de taxonomia.

Os resultados de pesquisa podem ser cadeias ou registos de PID e podem ser especificados na chamada `DKIKFSearchConfiguration`.

Ajuste de rendimento

Poderá verificar problemas de rendimento ao emitir consultas complexas, especialmente se a cadeia de consulta contiver vários operadores OR. Para aumentar o rendimento, evite consultas complexas que contenham operadores OR, aplicando as regras de Morgan’s para converter expressões OR. Construa consultas de pesquisa de texto complexas utilizando os operadores monádicos + e -. As classes para converter estas expressões para a linguagem de consulta de Information Mining encontram-se em

`com.ibm.mm.sdk.common.infomining.DKIKFWebQueryConverter` para aplicações que utilizam a API de Serviço. Para aplicações que utilizam os beans, utilize o método `CMBAAdvancedSearchService.convertWebQuery`.

Executar uma tarefa de servidor

Se pretende executar uma sequência definida de tarefas da aplicação cliente, pode agrupar estas chamadas numa tarefa de servidor, enviá-las para o servidor uma vez e, de seguida, chamar esta tarefa do servidor sempre que o cliente dela necessitar. Isto mantém o nível de tráfego da rede o mais reduzido possível e melhora significativamente o rendimento.

Uma tarefa de servidor é um objecto que implementa a interface `com.ibm.mm.sdk.common.infomining.DKIKFServerTask`. O objecto é replicado da aplicação cliente, definido no objecto e serviço utilizando o método `setServerTask` e, de seguida, é executado no servidor utilizando o método `runServerTask`.

No exemplo seguinte, um objecto de classe `AnalysisTask`, que implementa a interface de tarefa de servidor, é definido no serviço e, de seguida, é executado:

```
...
ikfService.setServerTask(new AnalysisTask(secondCatalog.getName()));
HashMap documentMap = new HashMap();
documentMap.put("PID1", DKIKFTextDocument.create("content1"));
documentMap.put("PID2", DKIKFTextDocument.create("content2"));
documentMap.put("PID3", DKIKFTextDocument.create("content3"));
Map recordMap = (Map)ikfService.runServerTask(documentMap);

Iterator pids = recordMap.keySet().iterator();
while(pids.hasNext())
{
    DKIKFRecord record = (DKIKFRecord)recordMap.get(pids.next());
    System.out.println(record.getPID());
    System.out.println(record.getValue("IKF_LANGUAGE"));
    System.out.println(record.getValue("IKF_SUMMARY"));
}
...
```

É transmitido um mapa de documentos para a tarefa de servidor para ser processado. A tarefa de servidor devolve um mapa dos registos relativos aos documentos especificados que contém os metadados criados (idioma de documento e resumo, neste caso).

A origem da tarefa de servidor exemplo é a seguinte:

```
public class AnalysisTask implements DKIKFServerTask {
    private String catalogName;
    private DKIKFSchema catalogSchema;

    public AnalysisTask(String catalogName) {
        this.catalogName = catalogName;
    }

    public Serializable runServerTask(DKIKFService ikfService, Serializable argument)
        throws DKIKFServerTaskException {
        try {
            //o esquema é obtido apenas uma vez
            if(catalogSchema == null) {
                catalogSchema = ikfService.getLibrary().getCatalog(catalogName).getSchema();
            }

            //preparar o documento, ferramentas e o mapa a serem devolvidos
            Map documentMap = (Map)argument;
            DKIKFLanguageIdentifier languageIdentifier = new DKIKFLanguageIdentifier(ikfService);
            DKIKFSummarizer summarizer = new DKIKFSummarizer(ikfService);
            HashMap recordMap = new HashMap();
            Iterator pids = documentMap.keySet().iterator();

            //criar um registo para cada pid
            while(pids.hasNext()) {
                String pid = (String)pids.next();
                DKIKFTextDocument document = (DKIKFTextDocument)documentMap.get(pid);
                DKIKFRecord record = DKIKFRecord.create(pid, catalogSchema);
                String language = languageIdentifier.analyze(document)[0].getLanguage();
                document.setLanguage(language);
                record.setValue("IKF_LANGUAGE", language);
                record.setValue("IKF_SUMMARY", summarizer.analyze(document).getSummary());
                recordMap.put(pid, record);
            }

            //devolver os resultados
            return recordMap;
        }
    }
}
```

```

    }
    catch(Exception e) {
        throw new DKIKFServerTaskException(e);
    }
}

```

A tarefa de servidor é replicada com o nome do catálogo que contém o esquema necessário para a criação do registo. O método `runServerTask` obtém o esquema apenas uma vez. Para cada um dos documentos, cria um registo, executa as ferramentas para analisar o documento e armazena os metadados criados no registo. Por fim, todos os registos criados são devolvidos ao programa de chamada (aplicação cliente).

Nota

Apenas os metadados são devolvidos no código de exemplo acima descrito e não são criados registos no catálogo.

Exemplo de uma aplicação de Information Mining baseada em JSPs

A aplicação de Java Server Page (JSP) de Information Mining pesquisa no componente de Information Mining documentos dentro de categorias. Apresenta os documentos localizados numa estrutura de categorias.

A aplicação exemplificativa de JSP está localizada no seguinte directório:

| | |
|----------------|--------------------------------------|
| Windows | <CMBROOT>\samples\jsp\infomining\ |
| AIX | /usr/lpp/cmb/samples/jsp/infomining/ |
| Solaris | /opt/IBMcmb/samples/jsp/infomining/ |

O directório contém estes ficheiros:

advSearch.jsp O ficheiro de nível mais elevado do exemplo.

Esta parte fornece o código avançado de manuseamento de formato e específico para pesquisa, além de instruções de formatação de formatos (HTML). Também funciona como o controlador para seleccionar vistas dos dados. Este ficheiro contém instruções de iniciação específicas para a pesquisa avançada, em particular a disponibilidade de categorias em que se possa executar uma pesquisa.

catView.jsp Esta parte fornece o código específico para vistas e as instruções de formatação (HTML) para a visualização de categorias dos resultados obtidos. Contém ciclos para iterar através de resultados devolvidos, mas contém principalmente instruções de formatação.

classes.jsp Esta parte fornece o código lógico e de ligação ao bean. Apenas contém o código Java. Existe uma implementação de classes de estrutura de dados simples utilizadas para a visualização de resultados devolvidos. Existe também uma implementação de um operador de eventos para obter e manipular resultados devolvidos. É aqui que é feita grande parte do trabalho após a devolução de uma lista de resultados, a partir da pesquisa avançada.

logon.html A entrada de dados de conta e catálogo necessários para executar o exemplo.

A entrada de dados de conta e catálogo engloba o nome do servidor, o nome do utilizador, a palavra-passe e o nome do catálogo.

O código de origem é um exemplo da utilização dos beans de extracção de informações. Contém uma descrição do modo como o código funciona em casos de replicação de beans, de ligações de beans entre si, de processamento da devolução de documentos utilizando um tratador de eventos, etc.

Para que as JSPs possam ser executadas, o utilizador deve ter instalados o Enterprise Information Portal e o componente Information Mining. Também necessita de um servidor da Web que seja capaz de executar JSPs.

Para obter mais pormenores sobre aplicações de JSP, vá para:
<http://java.sun.com/products/jsp/index.html>

Instalar as JSPs

Antes de implementar as JSPs, certifique-se de que o IBM WebSphere Application Server (WAS) está instalado e em execução.

Os direitos de acesso de utilizador necessários para a implementação das JSPs são:

- Para Windows: Autoridade de administrador
- Para AIX: Privilégios de utilizador raiz
- Para Solaris: Privilégios de utilizador raiz

As JSPs estão implementadas como uma aplicação da Web no directório <WAS_Home>\installedApps\JSP.ear\JSP.war. Caso efectue alterações a um JSP de exemplo, substitua a JSP no directório acima referido pela JSP por si criada. O WAS irá recompilá-lo automaticamente.

Para obter informações relativas à configuração do WebSphere Application Server para as JSPs, consulte *Planning and Installing Enterprise Information Portal*.

Informações

Estas informações foram desenvolvidas para produtos e serviços disponibilizados nos E.U.A.

Os produtos, os serviços ou as funções descritas neste documento poderão não ser disponibilizados pela IBM noutros países. Consulte o seu representante IBM para obter informações sobre os produtos e serviços actualmente disponíveis na sua área. Quaisquer referências, nesta publicação, a programas licenciados IBM ou outros produtos ou serviços IBM, não significam que apenas esses programas licenciados, produtos ou serviços IBM possam ser utilizados. Qualquer outro produto, programa ou serviço, funcionalmente equivalente, poderá ser utilizado em substituição daqueles, desde que não infrinja nenhum dos direitos de propriedade intelectual da IBM. No entanto, é da inteira responsabilidade do utilizador avaliar e verificar o funcionamento de qualquer produto, programa ou serviço não IBM.

Nesta publicação, podem ser feitas referências a patentes ou pedidos de patente pendentes. O facto de este documento lhe ser oferecido não lhe confere quaisquer direitos sobre essas patentes. Todos os pedidos de informação sobre licenças deverão ser endereçados a:

IBM Portugal, SA
Praça de Alvalade, 7
1799 LISBOA CODEX
PORTUGAL

Pode endereçar os seus pedidos de informação sobre licenças relacionados com informação de duplo byte (DBCS) ao Departamento de Propriedade Intelectual IBM no seu país. Também pode enviá-los, por escrito, para:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japão

O parágrafo seguinte não se aplica ao Reino Unido ou a nenhum outro país onde tais cláusulas sejam compatíveis com a lei local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO “TAL COMO ESTÁ” SEM GARANTIA DE QUALQUER ESPÉCIE, QUER EXPLÍCITA QUER IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS DE NÃO INFRACÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO FIM. Alguns Estados não permitem a exclusão de garantias, quer explícitas quer implícitas, em determinadas transacções; esta declaração pode, portanto, não se aplicar ao seu caso.

É possível que estas informações contenham imprecisões técnicas ou erros de tipografia. A IBM permite-se fazer alterações periódicas às informações aqui contidas; essas alterações serão incluídas nas posteriores edições desta publicação. A IBM pode introduzir melhorias e/ou alterações ao(s) produto(s) e/ou programa(s) descrito(s) nesta publicação em qualquer altura sem aviso prévio.

Quaisquer referências, nesta publicação, a sítios da Web não IBM são fornecidas apenas para conveniência e não deverão nunca servir como aprovação desses sítios

da Web. Os materiais existentes nesses sítios da Web não fazem parte dos materiais destinados a este produto IBM e a utilização desses sítios da Web será da exclusiva responsabilidade do utilizador.

A IBM pode utilizar ou distribuir qualquer informação que lhe seja fornecida, de qualquer forma que julgue apropriada, sem incorrer em qualquer obrigação para com o autor dessa informação.

Os possuidores de licenças deste programa que pretendam obter informações sobre o mesmo com o objectivo de permitir: i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização recíproca das informações que tenham sido trocadas, deverão contactar a:

IBM Portugal, SA
J46A/G4
1799 LISBOA CODEX
PORTUGAL

Tais informações poderão estar disponíveis, sujeitas aos termos e às condições adequadas, incluindo, nalguns casos, o pagamento de uma taxa.

O programa licenciado descrito neste documento e todo o material licenciado disponível para o programa são fornecidos pela IBM de acordo com os termos do IBM Customer Agreement, do IBM International Program License Agreement or de qualquer outro acordo equivalente entre ambas as partes.

Quaisquer dados de rendimento aqui contidos foram determinados num ambiente controlado. Consequentemente, os resultados obtidos noutros sistemas operativos poderão variar significativamente. Algumas medições podem ter sido efectuadas em sistemas ao nível do desenvolvimento, pelo que não existem garantias de que estas medições sejam iguais nos sistemas normalmente disponíveis. Para além disso, algumas medições podem ter sido calculadas por extrapolação. Os resultados reais podem variar. Os utilizadores deste documento devem verificar os dados aplicáveis ao seu ambiente específico.

A informação relativa a produtos não IBM foi obtida a partir dos fornecedores desses produtos, dos seus comunicados ou de outras fontes de divulgação ao público. A IBM não testou esses produtos e não pode confirmar a exactidão do rendimento, da compatibilidade ou de quaisquer outras afirmações relacionadas com produtos não IBM. Todas as questões sobre as capacidades dos produtos não IBM deverão ser endereçadas aos fornecedores desses produtos.

Todas as afirmações relativas às directivas ou tendências futuras da IBM estão sujeitas a alterações ou descontinuação sem aviso prévio, representando apenas metas e objectivos.

Esta publicação contém exemplos de dados e relatórios utilizados em operações comerciais diárias. Para os ilustrar o melhor possível, os exemplo incluem nomes de indivíduos, firmas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com nomes e moradas reais é mera coincidência.

LICENÇA DE COPYRIGHT:

Estas informações contém programas de aplicações exemplo no idioma de origem, que ilustram as técnicas de programação em várias plataformas operativas. Pode copiar, modificar e distribuir estes programas exemplo de qualquer forma, sem encargos para com a IBM, com a finalidade de desenvolver, utilizar, comercializar

ou distribuir programas de aplicação conformes à interface de programação de aplicações e destinados à plataforma operativa para a qual os programas exemplo são escritos. Estes exemplos não foram testados exaustivamente sob todas as condições. Deste modo, a IBM não garante nem se responsabiliza pela fiabilidade, assistência ou funcionamento implícito destes programas. Pode copiar, modificar e distribuir estes programas exemplo de qualquer forma, sem encargos para com a IBM, com o objectivo de desenvolver, utilizar, comercializar ou distribuir programas de aplicação em conformidade com as interfaces de programação de aplicações da IBM.

Marcas Comerciais

Os seguintes termos são marcas comerciais da International Business Machines Corporation nos Estados Unidos, noutros países ou em ambos:

| | | |
|--------------------------------|---------------|------------------|
| IBM | DisplayWrite | PowerPC |
| 400 | e-business | PTX |
| Funcionamento Avançado em Rede | HotMedia | QBIC |
| Unidade-a-Unidade | | |
| AIX | Hummingbird | RS/6000 |
| AIXwindows | ImagePlus | SecureWay |
| APPN | IMS | SP |
| AS/400 | Micro Channel | VideoCharger |
| C Set ++ | MQSeries | Visual Warehouse |
| CICS | MVS/ESA | VisualAge |
| DATABASE 2 | NetView | VisualInfo |
| DataJoiner | OS/2 | WebSphere |
| DB2 | OS/390 | |
| DB2 Universal Database | PAL | |

Approach, Domino, Lotus, Lotus 1-2-3, Lotus Notes e SmartSuite são marcas comerciais ou marcas comerciais registadas da Lotus Development Corporation nos Estados Unidos, em outros países ou ambos.

Intel e Pentium são marcas comerciais ou marcas comerciais registadas da Intel Corporation nos Estados Unidos, em outros países ou ambos.

Microsoft, Windows e Windows NT são marcas registadas da Microsoft Corporation nos Estados Unidos, outros países ou ambos.

Java e todas as marcas comerciais e logotipos baseados em Java são marcas comerciais ou marcas registadas da Sun Microsystems, Inc. nos Estados Unidos e/ou noutros países.

UNIX é uma marca registada do The Open Group nos Estados Unidos e em outros países.

Outros nomes de empresas, produtos e serviços podem ser marcas comerciais ou marcas de serviços de outras empresas.

Glossário

Este glossário define termos e abreviaturas específicas para este sistema. Os termos em *itálico* são definidos noutro local deste glossário.

A

ADSM. Consultar *Tivoli Storage Manager*.

agente iterativo. Uma classe ou construção utilizada para se deslocar num conjunto de objectos, um de cada vez.

API. Consulte *interface de programação de aplicações*.

aplicação cliente. Uma aplicação escrita com as APIs de Content Manager para personalizar uma interface de utilizador. Uma aplicação escrita com as APIs orientadas para objectos e de Internet para aceder aos *servidores de conteúdos* a partir do Enterprise Information Portal.

Aplicação de Cliente para Windows. Um sistema completo de gestão de objectos fornecido com o Content Manager e escrito com APIs do Content Manager. Suporta a criação de documentos e de pastas, armazenamento e apresentação, processamento e controlo de acesso. Poderá personalizá-la com as rotinas de saída de utilizador e invocá-la parcialmente.

área de transferência ascendente. A área de armazenamento de trabalho para o *gestor de recursos*. Também denominada como *gestor de recursos cache*.

arquitectura do conteúdo de documentos (DCA). Uma arquitectura que garante a integridade das informações dum documento objecto de intercâmbio numa rede de sistema de escritório. A DCA fornece a regra de especificação da forma e do significado de um documento. Define o texto de formato passível de revisão (alterável) e o texto de formato final (não alterável).

arquivador de suportes. Um dispositivo físico utilizado para armazenar sequências de dados de áudio e vídeo. O VideoCharger é um tipo de arquivador de suportes.

arquivo. Armazenamento persistente utilizado para retenção de informações a longo prazo, armazenadas normalmente a baixos custos por unidade armazenada e de acesso lento, mas numa localização geográfica diferente, de maneira a protegê-las contra falhas de equipamento e desastres naturais.

arquivo de dados. (1) Termo genérico para designar um local (tal como um sistema de bases de dados,

ficheiro ou directório) onde são armazenados dados. (2) Num programa de aplicações, uma representação visual de um *servidor de conteúdos*.

arquivo de dados associado. Representação virtual de qualquer número de *servidores de conteúdos* específicos, tais como Content Manager.

atributo. Uma unidade de dados que descrever uma determinada característica ou propriedade (por exemplo, nome, endereço, idade e por aí adiante) de um item, e que pode ser utilizada para localizar esse item. Um atributo tem um tipo, que indica a âmbito das informações armazenados por esse atributo, e um valor, que está dentro desse âmbito. Por exemplo, as informações sobre um ficheiro num sistema de ficheiros multimédia, tais como, título, tempo de execução ou tipo de codificação ((MPEG1, H.263, e por aí adiante). Para Enterprise Information Portal, consulte também *atributo federado* e *atributo nativo*

atributo federado. Uma categoria de metadados do Enterprise Information Portal definida por correspondência com *atributos nativos* num ou mais *servidores de conteúdos*. Por exemplo, o atributo federado, número de política, pode ser correlacionado para um *atributo*, num política, no Content Manager e para um atributo, policy ID, em Content Manager ImagePlus for OS/390.

atributo nativo. Uma característica de um objecto que é gerido num *servidor de conteúdos* e que é específico desse servidor de conteúdos. Por exemplo, o *campo-chave* número de política pode ser um atributo nativo num servidor de conteúdos do Content Manager enquanto o campo ID de política pode ser um atributo nativo num servidor de conteúdos do Content Manager OnDemand.

atributos base. Um conjunto de índices que é atribuído a cada *objecto*. Todos os objectos do Content Manager têm *atributos base*.

Audio/Video Interleaved (AVI). Uma especificação de ficheiro RIFF (*Resource Interchange File Format*) que permite que dados áudio ou em vídeo sejam imbricados num ficheiro. As pistas separadas podem ser acedidas em porções alternativas para serem reproduzidas ou gravadas enquanto mantêm o acesso sequencial no dispositivo de ficheiros.

AVI. Consulte *Audio/Video Interleaved*.

B

BLOB. Consulte *objecto binário grande*

C

cache. Uma memória tampão com propósitos especiais, mais pequena e mais rápida do que a memória principal, utilizada para manter uma cópia dos dados que pode ser acedida frequentemente. A utilização da memória cache reduz o tempo de aceso, mas pode aumentar os requisitos de memória. Consulte também *gestor de recursos memória cache* e *memória cache de LAN*.

Cache da LAN. Uma área de memória temporária num *gestor de recursos* local que contém uma cópia de objectos armazenados num gestor de recursos remoto.

cache do servidor de objectos. Consulte *gestor de recursos cache*.

cadeia de consulta. Uma cadeia de caracteres que especifica as propriedades e valores de propriedade de uma consulta. Pode criar a cadeia de consulta numa aplicação e transmiti-la à consulta.

campo-chave. Consulte *atributo*.

carácter global. Um carácter especial, tal como, um asterisco (*) ou um ponto de interrogação (?) que pode ser utilizado para representar um ou mais caracteres. Qualquer carácter ou conjunto de caracteres pode substituir um carácter global.

cardinalidade. O número de linhas numa tabela de base de dados.

categoria. Consulte *tipo de item*

CGI. Consulte *Interface de porta de ligação comum*.

CIF. Consulte *ficheiro de intercâmbio comum*.

CIU. Consulte *unidade de intercâmbio comum*.

classe. Na concepção ou programação orientada para objectos, pode ser definido um modelo para criar objectos com uma definição comum e, consequentemente, propriedades, operações e comportamento comuns. Um objecto é uma ocorrência de uma classe.

classe abstracta. Uma *classe* de programação orientada por objecto que representa um conceito; as classes que dela derivam representam implementações do objecto. Não poderá construir um objecto de uma classe abstracta; isto não pode ser transformado em ocorrência.

classe de conectores. *Classe* de programação orientada para objectos que fornece um acesso standard a APIs nativas de *servidores de conteúdos*.

classe de conteúdo. Consulte *tipo de MIME*

classe de gestão. O termo utilizado nas APIs para a *política de migração*.

classe de índice. Consulte *tipo de item*

classe de memória. Identifica o tipo de suporte de dados no qual um determinado objecto está armazenado. Não está directamente associado com uma localização física; no entanto, está directamente associado com o *gestor de dispositivos*. Os tipos de classes de memória incluem:

DASD

Disco Rígido

Óptico

Sequência

Banda

TSM

classificação. (1) (2) Um valor inteiro que representa a importância de uma determinada parte relativamente aos resultados de uma consulta. Um valor mais elevado significa uma correspondência mais exacta.

classificação de tipo de item. Uma categorização dentro de um *tipo de item* que fornece mais identificação aos *itens* desse tipo de item. Todos os itens do mesmo tipo de item têm a mesma classificação de tipo de item.

O Content Manager fornece as seguintes classificações de tipo de item: *pasta*, *documento*, *objecto*, *vídeo*, *imagem* e *texto*; os utilizadores também podem definir as suas próprias classificações de tipo de item.

cliente de biblioteca. O componente de um sistema de Content Manager que fornece uma interface de programação de baixo nível para o sistema de bibliotecas. O cliente da biblioteca inclui API que fazem parte do kit do programador de software.

cliente de poucos recursos. Um cliente tem pouco ou nenhum software instalado mas que tem acesso a software gerido e disponibilizado por servidores de rede a ele ligados. Um cliente de poucos recursos constitui uma alternativa a um cliente de plenas funções, como uma estação de trabalho.

cliente/servidor. Nas comunicações, o modelo de interacção no processamento de dados distribuído no qual um programa num local envia um pedido para um programa para outro local e aguarda uma resposta. O programa que está a fazer o pedido é chamado de cliente; o programa que pede é chamado de servidor.

Common Gateway Interface (CGI). Um standard para o intercâmbio de informações entre um servidor da Web e programas que lhe são externos. Os programas externos podem ser escritos em qualquer linguagem de programação suportada pelo sistema operativo no qual o servidor da Web esteja a ser executado. Consulte *script CGI*.

componente de raiz. O primeiro ou único nível de um *tipo de item* hierárquico, que consiste em *atributos* definidos pelo sistema e pelo utilizador relacionados.

componente descendente. Nível secundário ou inferior opcional de um *tipo de item* hierárquico. Cada componente descendente está directamente associado ao nível que acima dele.

componente. Termo genérico para um *componente raiz* ou um *componente derivado*.

conjunto. Um grupo de objectos com um conjunto de regras de gestão semelhante.

conjunto de privilégios. Um conjunto de *privilégios* para trabalhar com componentes e funções do sistema. O administrador atribui conjuntos de privilégios a utilizadores (IDs de utilizador) e *grupos de utilizadores*.

construtor. Em linguagens de programação, consiste num método com o mesmo nome que a classe e que é utilizado para criar e inicializar objectos dessa classe.

consulta por conteúdo de imagem (QBIC). Uma tecnologia de consultas que permite pesquisas baseadas em conteúdo visual, denominadas funções, em vez de texto simples. Utilizando a QBIC, pode procurar objectos com base nas suas características visuais, como, por exemplo, cor e textura.

contentor. Um elemento de uma interface de utilizador que guarda objectos. No *gestor de pastas*, um *objecto* que pode conter outras pastas ou documentos.

controlo de acesso. O processo que garante que determinadas funções e *objectos* armazenados podem ser acedidos apenas por utilizadores autorizados sob formas autorizadas.

coordenador de fluxo de trabalho. Em fluxos de trabalho de Content Manager anteriores, um utilizador que recebe a notificação de que um *item de trabalho* no *fluxo de trabalho* não foi processado numa hora específica. O utilizador é seleccionado para um *grupo de utilizadores* específico ou após a criação do fluxo de trabalho.

critérios de procura. No Content Manager, os valores dos *atributos* que são utilizados para obter um *item* armazenado. No Enterprise Information Portal, campos específicos que um administrador define para um *modelo de pesquisa* que limita e define as opções que estão disponíveis para os *utilizadores*.

cursor. Uma estrutura de controlo designada utilizada por um programa de aplicação para apontar para uma linha específica num conjunto ordenado de linhas. O cursor é utilizado para obter linhas a partir do conjunto.

D

dados de sequência. Quaisquer dados enviados através de uma ligação a uma rede a uma velocidade especificada. Uma sequência pode ser um tipo de dados ou uma combinação de tipos. As velocidades dos dados, expressas em bits por segundo, variam conforme os diferentes tipos de sequências e redes.

DCA. Consulte *arquitectura do conteúdo de documentos*.

DDO. Consulte *objecto de dados dinâmicos*.

definição de correspondências de utilizadores.

Associar palavras-passe e IDs de utilizador do Enterprise Information Portal às palavras-passe e aos IDs de utilizador correspondentes num ou mais servidores de conteúdos. A correlação de utilizador permite o início de sessão única no Enterprise Information Portal e *servidores de conteúdos* múltiplos.

definição de tipo de documento (DTD). As regras que especificam a estrutura para uma classe específica de documentos XML. A DTD define a estrutura com elementos, atributos e notações, e estabelece as restrições para como cada elemento, atributo e notação podem ser utilizados dentro de uma classe específica de documentos. Uma DTD é análoga a um esquema de base de dados, já que a DTD descrever completamente a estrutura para um idioma de marca específica.

definição do servidor. As características de um *servidor de conteúdos* específico que o identifica, de forma única, no Enterprise Information Portal.

definição do tipo de servidor. A lista de características, de acordo com o que foi definido pelo administrador, necessárias para identificar de modo exclusivo um servidor personalizado de um determinado tipo perante o Enterprise Information Portal.

documento. Um *item* que pode ser armazenado, obtido e trocado entre utilizadores e sistemas do Content Manager como uma unidade independente. Um artigo com o documento de *tipo semântico* deve conter informações que componham um documento, mas isso não significa que constitua uma implementação do modelo de documentos do Content Manager.

Um artigo criado a partir de um tipo de artigo classificado de documento (uma implementação específica do modelo de documentos do Content Manager) deve conter partes de documento. Pode utilizar tipos de artigo classificados de documento para criar artigos com o tipo semântico documento ou pasta.

As partes de documento podem incluir tipos variados de conteúdos, incluindo, por exemplo, texto, imagens e folhas de cálculo.

DTD. Consulte *definição de tipo de documento*

E

elemento. Um *objecto* atribuído pelo *gestor de listas* a uma aplicação.

encaminhamento de documentos processo. No Content Manager, uma sequência de *passos de trabalho*, e as regras que regem esses passos, pelos quais um *documento* ou *pasta* passa enquanto está a ser processado.

entidade associada. Um *objecto* de metadados do Enterprise Information Portal composto por *atributos associados* e opcionalmente associados a um ou mais *índices de texto associado*.

entidade nativa. Um *objecto* que é gerido num *servidor de conteúdos* específico e que é composto por *atributos nativos*. Por exemplo, as *classes de índice* do Content Manager são entidades nativas compostas por *campos-chave* do Content Manager.

estado do fluxo de trabalho. O estado de um *fluxo de trabalho*.

estado do trabalho. O estado de um determinado *item de trabalho*, *documento*, ou *pasta*.

estrutura de dados gerados pela máquina (MGDS).

(1) Um protocolo de formato de dados estruturados IBM para transmitir dados de caracteres entre os diversos programas do Content Manager ImagePlus for OS/390. (2) Dados extraídos de uma imagem e colocados no formato de sequência de dados normal (GDS).

Extensible Markup Language (XML). Uma metalinguagem padrão para definir idiomas de marca que são derivados de, e são um sub-conjunto de, SGML. O XML omite as partes mais complexas e menos utilizadas de SGML e torna mais fácil escrever aplicações para processar tipos de documentos, autor e gerir informações estruturadas, e transmitir e partilhar informações estruturadas através de diversos sistemas. A utilização de XML não requer as aplicações e processamento robusto que são necessários para SGML. XML está a ser desenvolvido sob as normas do World Wide Web Consortium (W3C).

extracção de informações. O processo automático de extracção de informações-chave de texto(resumo), localizando temas predominantes num conjunto de documentos (categorização), e pesquisando documentos relevantes utilizando consultas poderosas e flexíveis.

F

ficheiro de intercâmbio comum (CIF). Um ficheiro que contém uma sequência de dados ImagePlus Interchange Architecture (IPIA).

ficheiro de tabelas de rede. Um ficheiro de texto que contém as informações de configuração específicas do sistema para cada nó num sistema do Content Manager. Cada nó do sistema tem de ter um ficheiro de tabelas de rede que identifique o nó e enuncie os nós com os quais tem de estabelecer ligação.

O nome da tabela de rede é FRNOLINT.TBL.

ficheiro README. Um ficheiro que deve ser visualizado antes do programa a ele associada ser instalado ou executado. Um ficheiro README normalmente contém informações de último minuto sobre o produto, informações sobre a instalação ou sugestões sobre como utilizar o produto.

fluxo de trabalho. Em Content Manager anteriores, uma sequência de *cestos de trabalho* pelos quais um *documento* ou *pasta* passa enquanto está a ser processada. No Enterprise Information Portal, uma sequência de *passos de trabalho*, e as regras que governam esses passos, através dos quais um *pacote de trabalho*, um *documento* ou uma *pasta* progridem enquanto está a ser processada.

Por exemplo, aprovação da reclamação descreve o processo pelo qual uma reclamação de um seguro teria de passar para ser aprovada.

formato de dados. Consulte *tipo de MIME*

função. As informações de conteúdo visual armazenadas no servidor de pesquisa de imagens. Do mesmo modo, as características visuais que as aplicações de pesquisa de imagens utilizam para determinar correspondências. As quatro funções QBIC são: cor média, cor do histograma, cor posicional e textura.

H

HTML. Consulte *Hypertext Markup Language*.

Hypertext Markup Language (HTML). Uma linguagem de marcação que está em conformidade com a norma SGML e que foi concebida principalmente para suportar a apresentação online de informações textuais e gráficas que incluem ligações de hipertexto.

I

identificador permanente. Um identificador que identifica de modo único um *objecto*, independentemente do local onde está armazenado. O PID é composto por um ID do item e por uma localização.

Image Object Content Architecture (IOCA). Uma recolha de construções utilizadas para intercâmbio e apresentação de imagens.

incluído. No Content Manager, um objecto que está online e que está numa unidade, mas que não tem *instalações* activas. Contrasta com *montado*.

índice. Para adicionar ou editar os valores de atributo que identificam um *item* ou um *objecto* específico, de forma a este poder ser posteriormente obtido.

índice de texto federado. Um objecto de metadados do Enterprise Information Portal que é definido para um ou mais *índices de texto nativo* num ou em mais *servidores de conteúdos*.

índice de texto nativo. Um índice dos *itens* de objectos que são geridos num *servidor de conteúdos* específico. Por exemplo, um índice de pesquisa de texto simples num servidor de conteúdos do Content Manager.

índice (TOC). A lista de *documentos* e *pastas* contidas numa pasta ou *repositório de trabalho*. Os resultados da pesquisa são apresentados como um índice numa pasta.

instalado. No Content Manager, um objecto que está online e numa unidade, com *instalações* activas. Contrasta com *incluído*.

instalar. Colocar um suporte de dados numa posição para funcionar.

interface de programação de aplicações (API). Uma interface de software que permite que as aplicações comuniquem entre si. Uma API é o conjunto de instruções em linguagem de programação que podem ser codificadas num programa de aplicação para obter as funções e os serviços específicos fornecidos pelo programa licenciado subjacente.

inventário do servidor. A lista global de *entidades nativas* e *atributos nativos* a partir dos *servidores de conteúdos* especificados.

IOCA. Consulte *Image Object Content Architecture*.

item. No Content Manager, termo genérico para uma ocorrência de um *tipo de item*. Por exemplo, um item pode ser *pasta*, *documento*, vídeo ou imagem. Termo genérico para a unidade mais pequena de informação que o Enterprise Information Portal administra. Cada item tem um identificador. Por exemplo, um item pode ser uma *pasta* ou um *documento*.

item de trabalho. Em fluxos de trabalho de Content Manager e fluxos de trabalho avançados de Enterprise Information Portal anteriores, qualquer actividade de trabalho que se encontre activa num *fluxo de trabalho*.

J

JavaBeans. Uma tecnologia de componente de software, independente de plataformas, para construir componentes de Java reutilizáveis chamados “beans.” Depois de serem construídos, esses beans pode ficar disponíveis para serem utilizados por outros

engenheiros de software ou podem ser utilizados em aplicações de Java. Utilizando o JavaBeans, os engenheiros de software podem manipular e reunir beans num ambiente gráfico de programação de arrastar-e-largar.

Joint Photographic Experts Group (JPEG). (1) Um grupo que trabalho para estabelecer a norma para a compressão de imagens digitalizadas de tom contínuo. (2) A norma para imagens paradas desenvolvidas por este grupo.

JPEG. Consulte *Joint Photographic Experts Group*.

L

LAN. Consulte *rede de área local*.

libertar. Remover critérios de suspensão de um *item*. Um item suspenso é libertado quando os critérios tiverem sido satisfeitos ou quando um utilizador com autoridade adequada substituir os critérios e o libertar manualmente.

ligação. Uma relação direccional entre dois *artigos*: a origem e o destino. Poderá utilizar um conjunto de ligações para modelar demasiadas associações. Contrasta com *referência*

lista de acções. Uma lista aprovada das acções, definidas por um administrador de sistemas ou outro *coordenador de fluxo de trabalho*, que um utilizador possa efectuar num processo de *fluxo de trabalho* ou de encaminhamento de documentos.

lista de controlo de acesso. Uma lista constituída por um ou mais IDs de utilizadores e grupos de utilizadores e os seus *privilégios* associados. Poderá utilizar as listas de controlo de acesso para controlar o acesso do utilizador aos *itens* e *objectos* no sistema do Content Manager. Poderá utilizar as listas de controlo de acesso para controlar o acesso dos utilizadores à *procura de modelos* no sistema do Enterprise Information Portal.

lista de trabalhos. Um conjunto de *itens de trabalho*, *documentos* ou *pastas* atribuídas a um utilizador.

M

memória gerada pelo sistema (SMS). A abordagem do Content Manager à gestão da memória. O sistema determina a colocação dos objectos e gere automaticamente a cópia de segurança, movimentação, espaço e segurança dos objectos.

método. Na concepção ou programação Java, o software que implementa o comportamento especificado por uma operação. Sinónimo de uma função de membro no ++.

MGDS. Consulte *estrutura de dados gerada pela máquina*.

migração. (1) O processo de mover dados e origem de um sistema informático para outro sistema informático sem os converter como, por exemplo, quando são movidos para um novo ambiente operativo. (2) A instalação de uma nova versão ou edição de um programa para substituir uma versão ou edição anterior.

migrador. Uma função do *gestor de recursos* que verifica as *políticas de migração* e move os objectos para a *classe de armazenamento* seguinte quando estão marcadas para serem movidas.

Mixed Object Document Content

Architecture–Presentation (MO:DCA–P). Uma arquitectura de subconjunto do MO:DCA que é utilizada como um envelope para guardar documentos que são enviados para a estação de trabalho do Content Manager ImagePlus for OS/390 para serem apresentados ou impressos.

MO:DCA. *Objecto Misto da Arquitectura do Conteúdo de Documentos*

MO:DCA–P. *Mixed Object Document Content Architecture—Apresentação*

modelo de pesquisa. Um formato, composto por *critérios de pesquisa* concebido por um administrador para um determinado tipo de pesquisa associada. O administrador também identifica os *utilizadores* e os *grupos de utilizador* que podem aceder a cada modelo de pesquisa.

multimédia. Combinar elementos de suporte de dados diferentes (texto, gráficos, áudio, imagem parada, vídeo, animação) para visualizar e controlar a partir de um computador.

Multipurpose Internet Mail Extensions (MIME) . Consulte *tipo de MIME*

O

Object Linking and Embedding (OLE). Uma especificação da Microsoft para ligar e incorporar aplicações para que possam ser activadas a partir de outras aplicações.

objecto. Qualquer conteúdo digital que um utilizador pode armazenar, obter e manipular como uma unidade única, por exemplo, imagens de JPEG, de áudio MP3, vídeo AVI e um bloco de texto a partir de um livro.

objecto binário grande (BLOB). Uma sequência de bytes com um tamanho variável entre 0 bytes e 2 gigabytes. Esta cadeia não tem associada uma página de códigos nem um conjunto de caracteres. As imagens, os objectos de áudio e de vídeo são armazenados em BLOBs.

objecto da biblioteca. Consulte *item*.

objecto de dados dinâmico (DDO). Num programa de aplicações, uma representação genérica de um objecto armazenado utilizado para mover esse objecto para dentro e para fora do armazenamento.

objecto de dados expandido (XDO). Num programa de aplicação, uma representação genérica de um *objecto* de multimédia complexo armazenado que é utilizado para mover esse objecto para dentro e para fora da memória. Os XDOs são, a maior parte das vezes, contidos dentro de DDOs.

Object Misto da Arquitectura do Conteúdo de Documentos (MO:DCA). Uma arquitectura IBM desenvolvida para permitir o intercâmbio de dados de objectos entre aplicações no ambiente de intercâmbio e entre ambientes.

OLE. Consulte *Object Linking and Embedding*.

P

pacote. Um conjunto de *classes* relacionadas e interfaces que fornecem protecção de acesso e gestão de espaço de nome.

pacote de trabalho. No Enterprise Information Portal Versão 7.1, um conjunto de *documentos* encaminhado de uma localização para outra. Os utilizadores acedem e trabalham com pacotes de trabalho através de *listas de trabalho*.

parâmetro identificador. Uma cadeia de caracteres que representa um objecto e é utilizada para obter o objecto.

parte. Consulte *objecto*.

passo do trabalho. Um ponto discreto do *fluxo de trabalho* ou do *processo de encaminhamento de documentos* pelo qual um *item de trabalho*, *documento* ou *pasta* deve passar.

pasta. Um *artigo* de qualquer *tipo de artigo*, independentemente da classificação, com o *tipo semântico* pasta. Qualquer artigo com tipo semântico pasta contém uma funcionalidade de pasta específica facultada pelo Content Manager, para além de todas as capacidades de artigo que não é de recurso e todas as funcionalidades adicionais disponíveis numa classificação de tipo de artigo como, por exemplo *documento* ou artigo de recurso. As pastas podem conter um número indeterminado de artigos de quaisquer tipos, incluindo documentos e sub-pastas. Uma pasta é indexada por *atributos*.

patron (patrono). O termo utilizado nas APIs do Content Manager para *utilizador*.

pesquisa associada. Uma consulta emitida a partir do Enterprise Information Portal que pesquisa

simultaneamente dados num ou mais *servidores de conteúdos* que podem ser heterogéneos.

pesquisa combinada. Uma consulta que combina um ou mais dos seguintes tipos de pesquisas: *paramétrica*, texto ou imagens.

pesquisa paramétrica. Uma consulta de *objectos* baseada nas *propriedades* dos *objectos*.

PID. Consulte *identificador permanente*.

política de migração. Uma marcação definida pelo utilizador para mover *objectos* de uma *classe de armazenamento* para a seguinte. Descreve as características de retenção e de transição de classe para um grupo de *objectos* numa hierarquia de armazenamento.

porta de ligação. Uma unidade funcional que interliga duas redes de computadores com arquitecturas de rede diferentes. Uma porta de ligação liga redes ou sistema de diferentes arquitecturas. Uma ponte interliga redes ou sistema com a mesma arquitectura ou com arquitecturas semelhantes.

privilégio. O direito de aceder a um *objecto* específico de um modo específico. Os privilégios incluem direitos como criação, eliminação e selecção de *objectos* armazenados no sistema. Os privilégios são atribuídos pelo administrador.

propriedade. Uma característica de um *objecto* que o descreve. Uma propriedade pode ser alterada ou modificada. O estilo de escrita é um exemplo de uma propriedade.

Q

QBIC. Consulte *consulta por conteúdo de imagem*.

R

recolha associada. Um agrupamento de *objectos* que resulta de uma *pesquisa associada*.

rede de área local (LAN). Uma rede na qual um conjunto de dispositivos estão ligados uns aos outros para comunicar e que podem ser ligados a uma rede maior.

referência. Associação um-a-um de direcção única entre uma raiz ou um *componente descendente* e outro *componente de raiz*. Contrasta com *ligação*.

registo do histórico. Um ficheiro que mantém um registo das actividades de um *fluxo de trabalho*.

Remote Method Invocation (RMI). Um conjunto de APIs que permite programação distribuída. Um *objecto* de uma Java Virtual Machine (JVM) pode chamar métodos em *objectos* de outras JVMs.

repositório de trabalho. Um conjunto de *documentos* ou *pastas* que estão a ser processados ou que aguardam processamento. A definição de um cesto de trabalhos inclui as regras que governam a apresentação, o estado e a segurança do seu conteúdo.

reproduzir. Tomar dados que não são normalmente orientados para imagens apresentá-los como uma imagem. No Content Manager, os documentos de processamento de texto podem ser convertidos em imagens para fins de apresentação.

Resource Interchange File Format (RIFF). Utilizado para armazenar som e gráficos para serem reproduzidos em diferentes tipos de equipamento informático.

RIFF. Consulte *Resource Interchange File Format*.

rotina de saída de utilizador. Uma rotina escrita pelo utilizador que recebe controlo em *saídas de utilizador* predefinidas.

S

saída de utilizador. Um ponto num programa fornecido pela IBM em que é possível dar o controlo a uma rotina de saída de utilizador.

Script CGI. Um programa de computador executado num servidor da Web, que utiliza a *Common Gateway Interface (CGI)* para efectuar tarefas que normalmente não são efectuadas por um servidor da Web (por exemplo, acesso a bases de dados e processamento de formulários). Um script CGI é um programa CGI escrito numa linguagem de scripts como, por exemplo, Perl.

servidor de bibliotecas. O componente de um sistema do Content Manager que armazena, gere e processa consultas em *ítems*.

servidor de conteúdos. Um sistema de software que armazena dados multimédia e comerciais e os respectivos metadados necessários para os utilizadores trabalharem com esses dados. O Content Manager e o Content Manager ImagePlus for OS/390 são exemplos de servidores de conteúdos.

servidor de objectos. Consulte *gestor de recursos*.

servidor de suportes. Um componente do sistema Content Manager baseado no AIX utilizado para armazenar e aceder a ficheiros de vídeo.

servidor de utilitários. Um componente do Content Manager que é utilizado pelos utilitários da base de dados para fins de marcação. Poderá configurar um servidor de utilitário quando configurar um *gestor de recursos* ou *servidor de biblioteca*. Existe um servidor de utilitários para cada gestor de recursos e cada servidor de bibliotecas.

Servidor RMI. Um servidor que implementa o modelo de objecto distribuído do Java *Remote Method Invocation (RMI)*.

sistema autónomo. Um sistema Content Manager pré-configurado que instala todos os componentes de um sistema Content Manager num único computador pessoal.

sistema de ficheiros. No AIX, o método de criação de partições para memória num disco rígido.

sistema de ficheiros multimédia. Um *sistema de ficheiros* que é optimizado para o armazenamento e entrega de vídeo e áudio.

sistema de memória. Um termo genérico para memória no sistema Content Manager. Consulte *volume TSM, arquivador de suporte de dados e volume*.

SMS. Consulte *memória gerida pelo sistema*.

sobreposição. Um conjunto de dados predefinidos como, por exemplo, linhas, sombreado, texto, caixas ou logótipos que podem ser intercalados com os dados da variável numa página durante a impressão.

subclasse. Uma *classe* derivada de outra classe. Pode existir uma ou mais classes entre a classe e a subclasse.

subconjunto de classe de índices remissivos. No Content Manager anterior, uma vista de uma *classe de índice* que uma aplicação utiliza para armazenar, obter e visualizar pastas e objectos.

superclasse. Uma *classe* a partir da qual uma classe é derivada. Uma ou mais classes podem estar entre a classe e a superclasse.

supressor. Uma função do *gestor de recursos* que remove os *objectos* a partir do sistema.

suspender. Remover um *objecto* do *fluxo de trabalho* e definir os critérios de suspensão necessários para o activar. A activação posterior do objecto permite que este possa continuar a processar.

T

tipo de item. Um modelo para definir e mais tarde localizar como *itens*, consiste num *componente de raiz*, zero ou mais *componentes descendentes*, e uma classificação.

Tipo de MIME. Uma norma da Internet para identificar o tipo de objecto que está a ser transferido através da Internet. Os tipos de MIME incluem várias variantes de áudio, imagem e vídeo. Cada objecto tem um tipo de MIME.

tipo semântico. A utilização ou as regras para um *item*. Base, anotação e nota são tipos semânticos

fornecidos pelo Content Manager; os utilizadores também podem definir os seus próprios tipos semânticos.

Tivoli Storage Manager (TSM). Um produto *cliente/servidor* que fornece serviços de gestão da memória e de acesso de dados num ambiente heterogéneo. Suporta vários métodos de comunicação, faculta funções administrativas para a gestão da cópia de segurança e armazenamento de ficheiros e faculta funções para programação de operações de cópia de segurança.

TOC. Consulte *índice*.

transferência ascendente. O processo de mover um *objecto* armazenado a partir de um dispositivo off-line ou de baixa prioridade de volta para um dispositivo online ou de prioridade mais elevada, normalmente a pedido do sistema ou a pedido de um utilizador. Quando um utilizador pede um objecto armazenado na memória permanente, é guardada uma cópia de trabalho na *área de transferência ascendente*.

transferidor. Uma função do Content Manager *gestor de recursos* que move objectos a partir da *área de transferência ascendente* para o primeiro passo na *política de migração* do objecto.

troca. A capacidade de importar ou exportar uma imagem com o seu índice a partir de um sistema Content Manager ImagePlus for OS/390 para outro sistema ImagePlus utilizando um *ficheiro de intercâmbio comum* ou uma *unidade de intercâmbio comum*.

TSM. Consultar *Tivoli Storage Manager*.

U

unidade de intercâmbio comum (CIU). A unidade de transferência independente de um ficheiro de intercâmbio comum (CIF). Esta é a parte do CIF que identifica a relação para a base de dados de recepção. Um CIF pode incluir várias CIUs.

uniform resource locator (URL). Uma sequência de caracteres que representa os recursos de informações num computador ou numa rede, tais como a Internet. Esta sequência de caracteres inclui o nome abreviado do protocolo utilizado para aceder aos recursos de informações e às informações utilizadas pelo protocolo para localizar o recurso da informação. Por exemplo, no contexto da Internet, estes são nomes abreviados de alguns protocolos utilizados para aceder a vários recursos de informações: http, ftp, gopher, telnet e news.

utilizador. Uma pessoa que requer os serviços do Content Manager. Este termo normalmente refere-se a utilizadores de aplicações de clientes, em vez de aplicações de programadores, que utilizam as APIs do Content Manager. No Enterprise Information Portal,

qualquer utilizador que é identificado no programa de administração do Enterprise Information Portal.

V

vista de classes de índice. No Content Manager anterior, o termo utilizado nas APIS para *sub-conjunto de classes de índices*.

volume. Uma representação de um dispositivo ou unidade de memória física na qual os objectos do sistema são armazenados.

Volume TSM. Uma área de armazenamento local que é gerida pelo *Tivoli Storage Manager*.

X

XDO. Consulte *objecto de dados expandidos*.

XML. Consulte *Extensible Markup Language*.

Índice Remissivo

A

- abrir, BLOB 389
- abrir, CLOB 390
- abrir, cursor de conjunto de resultados 387
- acções
 - aceder a listas 405
 - criar 405
- actualizar, BLOB 388
- actualizar o conteúdo 390
 - XDO 58
- addObject 382, 387
- addToFolder 392
- adicionar, BLOB 388
- administração
 - objecto
 - obter 213
 - privilégios 137
- agrupamento de ligações
 - servlet 448
 - valores de servlet 450
- AIX
 - iniciar o servidor de RMI 28
 - objectos partilhados para C++ 29
- AllPrivSet 137, 139
- alterar palavra-passe 145
- ambiente, definir
 - C++ 28
 - Java 26
- âmbito
 - biblioteca de identificadores 444
 - transacção 220
- anotações
 - adicionar um objecto a um XDO 56
 - beans 412, 415
 - classe de dkAnnotationExt 391
 - classes de serviços 432
 - constante 155
 - documentos 431
 - FileNET 366
 - motor 433
 - OnDemand 327
 - personalizar 441
 - serviços 438
 - suporte de edição 440
 - tipo semântico 155
- anyA, DKAny 95
- API de gestão de documentos abertos (ODMA) 344
- API de serviço
 - agrupamento 506
 - armazenar metadados 507
 - atribuir categorias 507
 - criar registos 507
 - criar um documento de texto 505
 - determinar o idioma do documento 506
 - executar uma tarefa deservidor 509
 - extrair informações de documentos 506
 - filtrar um documento 505
- API de serviço (*continuação*)
 - gerar um resumo 506
 - gerir a biblioteca 502
 - gerir catálogos 502
 - gerir taxonomiasd 502
 - Information Mining 500
 - ligação 501
 - pesquisar 509
 - utilizar as ferramentas de extracção de informações 505
- aplicações JSP em Information Mining
 - implementação de WAS 512
- Aplicações JSP em Information Mining
 - exemplos 511
- armários, Domino.Doc 346
 - enumerar atributos 348
- armazenamento de dados,
 - Consulte servidores de conteúdos
- artigos
 - adicionar a uma pasta 168
 - associada 15
 - beans 412, 444, 446
 - constante em árvore 172
 - Content Manager Versão 8 130
 - criar um tipo de artigo de documento 213
 - dar entrada e saída 164
 - definir e obter atributos no CM 8 156
 - efectuar referências 143
 - elaboração de versões 135, 164
 - estabelecer ligações através de pastas 134
 - gerir com o servlet controlador 453
 - ligar 173
 - modificar atributos 156
 - obter artigos ligados 175
 - obter no CM 8 161
 - pesquisar no CM 8 160
 - remover de pastas 170
 - representação no modelo de dados 187
- artigos de dados, DDO 44
- artigos de recurso
 - constante 145
 - Content Manager Versão 8 131
 - definir um tipo de artigo para 158
 - tipo de artigo 145
- artigos de trabalho
 - aceder 402
- artigos que não são de recurso
 - Content Manager Versão 8 131
- ASCENDING 208
- assistente de configuração cliente 4
- associada 22
 - artigos 15
 - atributos 15
 - beans 446
 - classe de base de definição do servidor 394
 - consulta
 - criar 18
- associada (*continuação*)
 - consulta (*continuação*)
 - exemplos 20
 - ilustração de processamento 19
 - resultados 19
 - sintaxe 19
 - definição de correspondências 16
 - definição de correspondências de esquema 17
 - entidades 15
 - funções de administração de sistemas 22
 - modelo de documento 15
 - nível para beans 411
 - pastas 22
 - pesquisa de imagens 19
 - pesquisar
 - classe de expressão 381
 - Extended Search 364
 - ilustração 17
 - ilustração relacional 19
 - introdução 3, 15
 - processo 18
 - recolha 19
- associado
 - definir correspondências de IDs de utilizador 18
 - registar servidores de conteúdos 18
- associados
 - iteradores 100
 - recolhas 100
- associar, DataJoiner 6
- atributos
 - aceder num DDO 45
 - actualizar grupos no CM 8 150
 - agrupamento 131
 - apresentar em beans 418, 426
 - beans 412, 446
 - classes DK específicas para definição 385
 - comparar com DDOs e XDOs 14
 - consultar 198
 - Content Manager Versão 8 131
 - criar no CM 8 148
 - de vários valores 131
 - definição 131
 - definidos pelo utilizador 187
 - tornar pesquisáveis por texto 190
 - definir 246
 - definir e obter no CM 8 156
 - elaboração de versões 163
 - eliminar para DKAny 99
 - enumerar em bases de dados relacionais 373
 - enumerar em FileNET 366
 - enumerar em IP OS/390 330
 - enumerar no CM for AS/400 336
 - enumerar para OnDemand 317
 - enumerar para um tipo de artigo 151
 - Extended Search 351
 - gerir grupos no CM8 149

- atributos (*continuação*)
 - gerir no servlet controlador 453
 - modificar para um artigo 156
 - obter para um tipo de artigo de documento 213
 - obter propriedades de 47
 - pesquisa de imagens 296
 - referência 188
 - representar através de DDOs 143
- atributos de referência 188
- atributos definidos pelo utilizador
 - tornar pesquisáveis por texto 190
- ATTRONLY, ImagePlus for OS/390 336
- AUTOCOMMIT, bases de dados relacionais 371
- avaliar a pesquisa 33
- avaliar consultas 102

B

- base de dados de administração
 - definição de correspondências de esquema 12
 - descrição 4
 - introdução 4
- bases de dados relacionais
 - cadeias de configuração 371
 - cadeias de ligação 371
 - consultar 375
 - definir correspondências a partir de associada 16
 - enumerar atributos 373
 - enumerar entidades 373
 - introdução 370
 - ligação a 371
- BasicExpression 209
- bean de ligação 444
- bean searchTemplate 445
- BeanInfo 415
- beans 413
 - anotação 412
 - anotações 415
 - artigos 444
 - auxiliar 412
 - auxiliares 412
 - características
 - análise introspectiva 407
 - eventos 408
 - métodos 408
 - permanência 408
 - personalização 408
 - propriedades 408
 - classes de BeanInfo 415
 - classes de eventos e de receptor 416
 - classes de excepção 415
 - CMBSchemaManagement 409
 - compreender conceitos básicos 407
 - conjunto de ligações
 - CMBConnectionPool 444
 - construir WebSphere Studio Application Developer 409
 - critérios de pesquisa 445
 - documentos 448
 - elaboração de versões 419, 427
 - esquema 409
 - ficheiros JAR 409
 - fluxo de trabalho 409, 413
- beans (*continuação*)
 - fonte de dados
 - bean de fonte de dados 444
 - gestão de dados 409, 444
 - gestão de esquemas 444
 - gestão de utilizadores 444
 - Information Mining 414
 - introdução 407
 - invocar 409
 - ligações 409, 444
 - lista de trabalhos 410
 - Model View Controller (MVC) 411
 - modelo de pesquisa 444, 445
 - modulação 416
 - não visuais
 - categorias 411
 - configurações 411
 - considerações 416
 - construir aplicações 417
 - definição 407
 - eventos 417
 - funções 411
 - introdução 410
 - propriedades 417
 - outros construtores 408
 - pastas 447
 - pedido de pesquisa 409
 - pesquisar OnDemand 327
 - rastrear 417
 - receptores de sessão 416
 - registo de rastreio 444
 - relacionado com Java Viewer
 - definição 407
 - requisitos 408
 - resultados da pesquisa 444
 - serviço de consulta 444
 - serviços de documento 415, 444
 - singletons 416
 - suporte de lotes 410
 - utilizar em construtores 408
 - visuais
 - área de janela em árvore 422
 - áreas de pesquisa de texto 421
 - comportamentos
 - especializados 428
 - comportamentos gerais 427
 - construir aplicações 428
 - definição 407
 - editor de atributos 426
 - especificações do
 - visualizador 425
 - eventos chave 428
 - eventos de ajuda 428
 - força de ordenação 427
 - guardar/repor configuração 427
 - introdução 418
 - ligação 429
 - ligações 427
 - lista de modelos de pesquisa 420
 - menus emergentes 423
 - nomes 418
 - ocultar e apresentar botões 427
 - painel de início de sessão 419
 - painel de pesquisa 422
 - substituir 428
 - substituir menus emergentes 424
 - utilizar em janelas 430

- beans (*continuação*)
 - visuais (*continuação*)
 - visualizador de documentos 425
 - visualizador de modelos de pesquisa 421
 - visualizador de pastas 424
 - visualizador de resultados da pesquisa 422
 - visualizador de versões 427
 - visualizadores externos 426
 - visualizadores predefinidos 426
 - web.xml 409
- beans auxiliares 412
- BETWEEN 203, 208
- biblioteca
 - Java 26, 28
 - listagem de C++ 29
 - Microsoft Visual Studio .NET 31
- biblioteca de identificador
 - relacionados com artigos 446
 - relacionados com documentos 448
 - relacionados com esquemas 445
 - relacionados com ligações 444
 - relacionados com pastas 447
 - relacionados com pesquisa 446
- BLOB,
 - Consulte objecto binário grande (BLOB)

C

- C++
 - bibliotecas 29
 - cadeias de configuração de bases de dados relacionais 51
 - classes alteradas 9
 - classes novas 9
 - conectores 5
 - configurar o ambiente 28
 - constantes 35
 - DKAny 92
 - DKConstant.h 25
 - ficheiros de propriedades de mensagens de erro 393
 - ficheiros DLL 29
 - objectos partilhados para AIX 29
 - sequências de escape 207
 - suporte de XML 25
- cadeias de configuração, bases de dados relacionais 51
- campo de parâmetros adicionais, OnDemand 316
- campos, Domino.Doc 346
- campos, Extended Search 357
- caracteres globais
 - consulta 204
 - Domino.Doc 349
 - pesquisa de texto 190
- caracteres múltiplos 265
- caracteres únicos exigidos (SC) 265
- Catálogo de Informações
 - definir correspondências a partir de associada 16
 - restrições da versão 8.2 7

Catálogo de Informações de DB2 Data Warehouse Manager,
Consulte Catálogo de Informações catálogos, pesquisa de imagens 291
 cc2mime.ini 371
 CC2MIMEFILE, bases de dados relacionais 371
 CCSID 265
 cenário, exemplo de seguro para o Content Manager Versão 8 142
 centro de informações 141
 cesto de trabalho
 criar numa versão anterior do CM 310
 definição numa versão anterior do CM 307
 enumerar numa versão anterior do CM 311
 identificar numa versão anterior do CM 313
 regras 307
 changePassword 382
 checkedOutUserId 392
 checkIn, dkDatastoreExt 392
 checkOut, dkDatastoreExt 392
 classe de expressão 381
 classes de artigo,
 Consulte tipos de artigos
 classes de definição de dados 247
 classes de EIP comuns 379
 cliente
 construir para trabalhar com extração de informações 5
 cliente de administração
 descrição 4
 instalar em estações de trabalho adicionais 4
 introdução 4
 cliente de administração de sistemas
 criar listas de trabalho 401
 personalizar 23
 saídas do utilizador 23
 cliente de poucos recursos 436
 cliente exemplo 283
 aplicação 5
 CLOB,
 Consulte objecto grande de carácter (CLOB)
 cmb81.jar 408
 CMBAnnotationPropertiesInterface 442
 CMBAnnotationServices 434
 CMBAnnotationServicesCallbacks 434
 CMBAnnotationSet 434
 cmbcc2mime.ini 451
 CMBCC2MIME.INI 363
 cmbcc2mime.ini.samp 363
 cmbclient.ini 4, 451
 cmbcm81.jar 379
 cmbcm81.lib 29
 cmbcm817.dll 379
 cmbcm817d.dll 379
 cmbcm81d.lib 29
 CMBConnection 327, 409, 444
 cmbcs.ini 411, 451
 CMBDataManagement 409, 444
 cmbdes.ini 351
 CMBDocumentServices 436, 444

CMBDocumentViewer 419, 425
 especificações 425
 terminar 425
 CMBFolderViewer 418, 424
 CMBGenericDocViewre 434
 CMBItem 444
 CMBItemAttributesEditor 426
 CMBLogonPanel 418, 419
 CMBObject 446
 CMBPage 448
 CMBPageAnnotation 440
 CMBQueryService 409, 444
 cmbregist81.bat 28
 cmbregist81.sh 28
 CMBSchemaManagement 444
 CMBSearchPanel 422
 CMBSearchResults 444
 CMBSearchResultsView 327
 CMBSearchResultsViewer 418, 422
 CMBSearchTemplate 444
 CMBSearchTemplateList 327, 418, 420
 CMBSearchTemplateViewer 327, 418, 421
 cmbServlet.properties 449
 CMBServletAction 449
 cmbServlet.jsp.properties 449
 CMBSTCriterion 445
 CMBStreamingDocServices 434
 CMBStreamingDocServicesCallbacks 434
 cmbsvclient.ini 451
 cmbsvcs.ini 451
 CMBTraceLog 444
 CMBUserManagement 444
 cmbview81.jar 433
 CMBViewerConfiguration.properties 433
 CMCOMMON 4
 CMCOMMON_URL 4
 cmvcmenv.properties 4
 COBRA
 Serviço de Dados Permanentes (PDS) 12
 Serviço de Objecto Permanente 12
 colocação em memória cache, servlet controlador 451
 com.ibm.mm.sdk.client 25
 com.ibm.mm.sdk.common package 379
 com.ibm.mm.sdk.server 25
 Comparação 209
 CompareOperator 210
 compilador visual de C++ 29
 componentes descendentes
 Content Manager Versão 8 132
 criar no CM 8 153
 diagrama 132
 efectuar referências 133
 elaboração de versões 135
 representação no modelo de dados 187
 representar como atributos de DDO 143
 componentes raiz
 Content Manager Versão 8 132
 criar no CM 8 153
 efectuar referências 133
 elaboração de versões 135
 ligar 133

componentes raiz (*continuação*)
 representação no modelo de dados 187
 comprimento, BLOB 389
 concatReplace 389
 conector ICM,
 Consulte Content Manager Versão 8
 conectores
 descrição 5
 local versus remoto 5
 personalizar 378
 valores de servlet controlador 450
 conectores personalizados
 administração do sistema 379
 desenvolver 378
 expandir a classe
 FeServerDefBase 394
 configuração dinâmica, beans 411
 conjunto de ferramentas do visualizador de documentos
 Aplicação Java 436
 aplicações exemplo 435
 applet ou servlet 437
 arquitectura 432
 cliente de poucos recursos 436
 criar um visualizador genérico 433
 funções 431
 introdução 431
 menus emergentes 433
 modo dual e applet ou servlet 437
 motores
 Documento AFP2Web 432
 Documento INSO 432
 Documento Java 432
 Documento MS-Tech 432
 personalizar o visualizador genérico de documentos 433
 serviços de anotação 438
 visualizador autónomo 435
 constantes 35
 encaminhar 244
 construir aplicação
 beans não visuais 417
 beans visual 428
 criar no Content Manager Versão 8 142
 planear 140
 utilizar serviços de anotação 441
 Consulta
 Java 101
 dkResultSetCursor vs DKResults 102
 tipo de texto 109
 tipo paramétrico 102
 tipos de objectos de consulta 102
 consulta aritmética
 operações 198
 sintaxe 209
 consulta combinada
 definição 33
 Consulta combinada
 C++
 ordenar 306
 Sugestões de programação 306
 Java 191, 303
 paramétrica com texto 303
 utilizar um âmbito 305

- Consulta de XML W3C 185
- Consultar Segundo Conteúdo de Imagem (QBIC),
 - Consulte* pesquisa de imagens
- consultas
 - associada
 - processar 18
 - avaliar 102
 - bases de dados relacionais 375
 - classe de dkQuery 381
 - CM para AS/400 338
 - compreender a linguagem 185
 - Content Manager Versão 8 186
 - Domino.Doc 349
 - enumerar os resultados 202
 - executar 102
 - exemplo de caracteres globais 201
 - exemplos 194, 197
 - exemplos de encaminhamento 242
 - expressões combinadas 381
 - expressões compostas 381
 - Extended Search 355
 - Generalized Query Language (GQL) 355
 - gramática da linguagem 208
 - linguagem 203
 - modelo de dados 195
 - operações aritméticas 198
 - Panagon Image Services 368
 - pesquisa de cor média 290
 - pesquisa de imagens 287, 297
 - pesquisa de texto 190
 - pesquisa OnDemand 319
 - pesquisar documentos
 - estruturados 273
 - recolha associada 100
 - sequências de escape 203
 - sintaxe 197
 - sintaxe de consulta da pesquisa de imagens 288
 - sintaxe de Domino.Doc 349
 - sintaxe de ImagePlus 334
 - versão 201
- contém pesquisa de texto básica 190
- contém texto, pesquisa de texto 191
- CONTENT
 - FileNET 370
 - ImagePlus for OS/390 336
- Content Manager, versões anteriores
 - actualizar partes 253
 - actualizar pastas 255
 - armazenar XML 83
 - definir correspondências a partir de associada 16
 - eClient 5
 - fluxo de trabalho 307
 - identificador permanente (PID) 248
 - introdução 247
 - manusear objectos grandes 247
 - pesquisa de imagens 6, 283
 - representar documentos 248
 - representar partes 248
 - representar pastas 249
- Content Manager para AS/400
 - classes de índices 336
 - definir correspondências a partir de associada 16
- Content Manager para AS/400 (continuação)
 - enumerar entidades e atributos 336
 - execução da consulta 338
 - introdução 336
- Content Manager Versão 8
 - actualizar documentos 217
 - artigos 130
 - artigos de recurso 131
 - artigos que não são de recurso 131
 - atributos 131
 - coerência de servidor de bibliotecas e do gestor de recursos 220
 - componentes descendentes 132
 - componentes raiz 132
 - compreender a consulta de pesquisa 185
 - compreender a linguagem de consulta 185
 - conceitos básicos 130
 - conjunto de privilégios 137
 - consultar 186
 - controlar o acesso
 - grupos de utilizadores 138
 - listas de acesso 138
 - controlar privilégios de acesso 136
 - controlo de acesso 135
 - criar atributos 148
 - criar tipos de artigo 145
 - criar um documento 215
 - criar um servidor de conteúdos 144
 - criar uma aplicação 142
 - criar uma palavra-passe 145
 - dar entrada e saída de artigos 164
 - definir correspondências a partir de associada 16
 - definir e obter atributos de artigo 156
 - definir um tipo de artigo de recurso 158
 - documentos 131, 134
 - eClient 5
 - elaboração de versões 134
 - eliminar um documento 219
 - encaminhar documentos 224
 - enumerar atributos para um tipo de artigo 151
 - enumerar tipos de artigo 147
 - exemplo de cenário de seguro 142
 - exemplos 141
 - exemplos de consulta 194
 - gerir documentos 211
 - gerir grupos de atributos 149
 - gestor de recursos 129
 - ilustração de componentes 130
 - introdução 129
 - ligação a 144
 - ligações 133
 - ligar artigos 173
 - modificar atributos de artigo 156
 - objectos 133
 - obter artigos 161
 - obter um documento 219
 - partes de documento 131
 - pastas 134
 - pesquisar artigos 160
 - planear uma aplicação 140
- Content Manager Versão 8 (continuação)
 - referências 133
 - servidor de bibliotecas 129
 - Servidor de System Managed Storage (SMS) 129
 - tipos de artigo 131
 - trabalhar com controlo de acesso 175
 - trabalhar com o gestor de recursos 210
 - trabalhar com pastas 167
 - transacções 220
 - XML 80
- contentores
 - constante 155
 - Content Manager Versão 8 133
 - tipo semântico 155
- conteúdo de multimédia 13
- conteúdo do recurso
 - Consulte* XDO
- contraste 286
- controlo de acesso
 - atribuir uma lista a um artigo 184
 - Content Manager Versão 8 135
 - definir listas (ACLs) 180
 - diagrama 136
 - listas (ACL) 136, 138
 - NoAccessACL 139
 - obter e apresentar listas (ACLs) 182
 - PublicReadACL 139
 - regras para listas (ACLs) 139
 - SuperUserACL 139
 - trabalhar com 175
 - trabalhar com listas para o encaminhamento de documentos 244
- conversão de página de códigos
 - definir o subsistema da consola em Windows 32
- cor de histograma
 - definição 285
 - exemplo de consulta 290
 - valores válidos 289
- cor média
 - definição 285
 - exemplo de consulta 290
 - valores válidos 289
- cor posicional
 - definição 286
 - exemplo de consulta 290
 - valores válidos 290
- corresponder destacado,
 - Consulte* destacar, pesquisa de Texto
- createChildDDO 153
- createDDO 42
- criar 154
- critério, pesquisa, bean 445
- cursor, conjunto de resultados
 - abrir 387
 - actualizar 386
 - actualizar elementos
 - updateObject 387
 - adicionar elementos 387
 - deslocar 386
 - destruir 387
 - eliminar elementos
 - deleteObject 387
 - fechar 387

cursor, conjunto de resultados
 (continuação)
 indicar um elemento 387
 obter elementos 387
 obter informações relativas ao
 servidor de conteúdos 387
 obter parâmetro identificador 387
 posicionar 386
 verificar a validade 386
 cursor do conjunto de resultados
 funções 386
 Cursor do conjunto de resultados
 Java 121
 abrir e fechar 122
 criar uma recolha 124
 definir e obter 122

D

dados binários,
 Consulte XDO
 data_id 47
 databaseNameStr 144
 DataJoiner,
 Consulte DB2 DataJoiner
 DATASOURCE 351
 DB2
 assistente de configuração cliente 4,
 28
 cadeias de configuração 51
 suporte cliente 4, 28
 DB2 DataJoiner 370
 cadeias de configuração 51
 restrições da versão 8.2 6
 DB2 Extenders 13
 DB2 Net Search Engine (NSE) 33
 db2cliws_dj.bnd 6
 db2clpr_dj.bnd 6
 ddo.dtd 81
 ddoDocument 56
 DEFINED, FileNET 368
 definição de correspondências de
 esquema
 associar em federada 17
 bases de dados relacionais 371
 classe de dkSchemaMapping 381
 conectores personalizados 379
 introdução 12
 métodos de dkDatastore 383
 definição do tipo de documento (DTD),
 importar XML 81
 definições
 ambiente de C++ 28
 ambiente de Java 26
 Definições
 C++
 Construir em NT 31
 Java
 Cliente/Servidor 25
 Em NT 27, 28, 30
 Ligação de desligação de
 cliente 37
 Sugestões de programação 26
 Utilizar exemplos de applets e servlet
 de Java 443
 Acesso local 444
 Acesso remoto 444

Definições (continuação)
 Utilizar exemplos de applets e servlet
 de Java (continuação)
 Aplicação Java em cliente 443
 Servlet de Obtenção 443
 deleteDKAttributeDef 99
 deleteObject 382
 DemoSimpleAppl.java 417
 DESCENDING 208
 deslocar o cursor do conjunto de
 resultados 386
 destacar, pesquisa de texto 112
 obter informações para cada
 resultado 113
 obter informações para um resultado
 específico 117
 destruir, cursor de conjunto de
 resultados 387
 detecção e correção de problemas
 ficheiros de propriedades de
 mensagens de erro 393
 rastrear 33
 DfltACLCode 139
 DIGIT 208
 direccionalidade 286
 DIV 208
 divisões, Domino.Doc 346
 DK_CM_DOCUMENT 248
 DK_CM_FOLDER 249
 DK_CM_OPT_ACCESS_MODE 246
 DK_CM_OPT_CONTENT 370
 DK_CM_PROPERTY_ITEM_TYPE 248,
 249
 DK_CM_READWRITE 246
 DK_CM_VERSION_LATEST 162
 DK_DES_GQL_QL_TYPE 355
 DK_DL_OPT_ACCESS_MODE 388
 DK_OPT_TS_CCSSID 265
 DK_OPT_TS_LANG 265
 DK_READWRITE 388
 DK_SS_CONFIG 309, 310
 DK_SS_NORMAL 309, 310
 DK_TS_DOCFMT_HTML 273
 dkAbstractWorkFlowUserExit 405
 dkAnnotationExt 391
 DKAny
 alteração à versão 8.2 10
 atribuição a 93
 atribuição de 94
 código de tipo, obter 93
 destroying 95
 ecrã de 94
 eliminar recolhas 99
 gestão de memória 93
 sugestões de programação 95
 Typecode 92
 utilizar construtores do tipo 93
 utilizar dentro de um conjunto 95
 verificar o tipo 94
 dkAttrDef 246
 classes 385
 DKAttrFieldDefDD 346
 DKAttrGroupDefICM 150
 DKAttrKeywordDefDD 346
 DKAttrProfileDefDD 346
 DKBinderDefDD 346
 dkBlob 246

dkBlob (continuação)
 classes 388
 métodos 388
 DKBlobDL
 setToBeIndexed 301
 DKBlobFN 366
 DKCabinetDefDD 346
 dkClob
 classes 389
 dkCollection 380
 DKCollectionResumeListEntryICM 225
 DKConstant
 constantes
 comum 393
 DKConstant.h 25
 DKConstantFN 368
 DKConstants 35
 dkCQExpr 381
 DKCQExpr 368
 dkDatastore
 addObject 382
 avaliar 381
 changePassword 382
 consolidar 381
 deleteObject 382
 desligar 381
 executar 381
 executeWithCallback 381
 introdução 381
 ligar 381
 listDataSourceNames 382
 listDataSources 382
 listMappingNames 383
 moveObject 382
 registerMapping 382
 registerServices 382
 remover alterações 381
 retrieveObject 382
 unRegisterMapping 382
 unregisterServices 382
 updateObject 382
 DKDatastore 245
 conectores personalizados 379
 DKDatastorexx 245
 DKDatastoreAccessError 370
 dkDatastoreDef 245
 classes 383
 DKDatastoreDef
 métodos 383
 DKDatastoreDefDL
 funções adicionais 383
 DKDatastoreDefOD
 funções adicionais 383
 DKDatastoreDES
 consultar 355
 listEntities 351
 listEntityAttrs 351
 DKDatastoreDL
 Java 36
 enumerar esquemas e atributos de
 esquemas 39
 Enumerar servidores 38
 ligação 36
 opções de DKDatastoreDL 37
 dkDatastoreExt
 classes 391
 funções 391

- DKDatastoreFN 364
- DKDatastoreICM 142
- DKDatastoreIP 329
- DKDatastoreOD 315
 - listEntities 316
- DKDatastoreQBIC 284
- DKDatastoreTS
 - atributos 263
 - Java 260
 - enumerar esquemas 267
 - enumerar servidores 265
 - ligação 264
 - Opções de DKDatastoreTS 265
- DKDatastoreV4 336
- DKDatastoreIP
 - listEntityAttrs 332
- DKDatastoreOD
 - listSearchTemplates 319
- dkDDO 379
- DKDDO,
 - Consulte* objecto de dados dinâmico (DDO)
- DKDLITEMID 263
- DKDocRoutingServiceICM 225
- DKDocRoutingServiceMgmtICM 225
- DKDocumentDefDD 346
- DKDSIZE 263
- dkEntityDef 245
 - classes 384
 - funções 384
- DkEntityDefIP
 - getAttr 330
- DKEntityDefIP
 - listAttrNames 330
- DKException 34, 140
- DKFederatedCollection 100
- dkFederatedIterator 100
- dkFederatedQuery 100
- DKFederatedQuery 18
- DKFixedView 315
- DKFixedViewDataOD 316
- DKFolder 249
- dkIterator 100
 - alteração à versão 8.2 9
- DKLink 173
- DKLinkCollection 175
- DKLITEMID 296
- DKMessage_en.properties 393
- DKMessage_en_US.properties 393
- DKMessage_es_ES.propertie 393
- DKMessage_es.properties 393
- DKMessageId 393
- DKMessageID, FileNET 370
- DKParametricQuery 368
- DKPARTNO 263, 296
- DKParts
 - CM anterior 248
 - Extended Search 357
- DKPidXDO 392
- DKPrivilegeSetICM 177
- DKProcessICM 225
- dkQuery 381
- dkQueryableCollection 380
- DKRANK 249, 263, 296, 303
- DKRCNT 263
- DKREPTYPE 263, 296
- DKResource 315
- DKResourceGrpOD 316
- DKResults 380
 - posicionar o iterador 101
- dkResultSetCursor 246
 - funções 386
- DKResumeListEntryICM 225
- DKRMConfiguration 145
- DKRoomDefDD 346
- DKRouteListEntryICM 225
- dkSchemaMapping 383
- DKSequentialCollection 253, 255, 380
- dkSequentialIterator 380
- dkServerDef 246
 - classes 385
 - funções 386
- dkSort 99
- DKStorageManageInfo 78
- DKString
 - alteração à versão 8.2 10
- DKTimestamp
 - suspender um fluxo de trabalho 399
- dkUserManagement 393
- DkViewOD 316
- DKViews 315
- DKWorkBasketDL 307
- DKWorkflowFed
 - retomar 400
 - suspender 399
- DKWorkflowServiceDL 307
- DKWorkflowServiceFed
 - svWf 398
- DKWorkflowServicesFed 395
- dkWorkflowUserExit 405
- DKWorkItemFed
 - dar entrada 403
 - dar saída 403
- DKWorkListFed 401
- DKWorkListtICM 225, 232
- DKWorkNodeICM 225, 227
- DKWorkPackageICM 225, 239
- dkXDO 390
 - alteração à versão 8.2 9
- dkXDOBase 380
 - abrir 390
- DLSEARCH_DocType 103
- doAction 405
- documentos 154
 - atualizar 382
 - atualizar no CM 8 217
 - adicionar 382
 - agrupar com beans 415
 - anotar 431
 - apresentar em beans 419, 425
 - beans 415, 444, 448
 - colocação em memória cache no servlet controlador 451
 - Content Manager Versão 8 131, 134
 - criar no CM 8 215
 - criar o modelo de dados de gestão no CM 8 213
 - criar um tipo de artigo de documento 213
 - dar saída e entrada 392
 - diferenças de gestão em versões anteriores do CM 254
 - Domino.Doc 346
- documentos (*continuação*)
 - elaborar versões de partes no modelo de dados de gestão 219
 - eliminar 382
 - eliminar no CM 8
 - SDocModelItemICM 219
 - encaminhar através de um processo 224
 - fluxo de trabalho da versão anterior do CM 307
 - gerir no CM 8 211
 - iniciar um processo de encaminhamento 236
 - mover 382
 - obter 382
 - obter no CM 8 219
 - pesquisa de texto de 190
 - representar no modelo de dados 188
 - tipo semântico 155
 - visualizar,
 - Consulte* conjunto de ferramentas do visualizador de documentos
- documentos identificados
 - pesquisa de texto 273
- Domino.Doc
 - API de gestão de documentos abertos (ODMA) 344
 - consultar 349
 - consultar objectos de DKResults 102
 - definir correspondências a partir de associada 16
 - enumerar atributos de armários 348
 - enumerar entidades e sub-entidades 346
 - introdução 344
 - modelo de objectos 345
 - sintaxe da consulta 349
- DSNAME, bases de dados relacionais 371, 372
- dsType 327
- dynamic data object (DDO)
 - obterObjecto 163
 - preencher com setData 154

E

- eClient
 - criado por JavaBeans 407
 - introdução 5
- elaboração de versões
 - artigos 162
 - atributos 163
 - beans 419, 427
 - Content Manager Versão 8 134
 - controlada pela aplicação 135
 - definir para um artigo 164
 - identificador permanente (PID) 135
 - partes no modelo de dados de gestão de documentos 219
 - política 135
 - políticas de controlo de versões 165
 - tipos de artigo 165
- elim, BLOB 388
- eliminar 388
- encaminhamento 413
 - beans 412, 413
 - introdução 224

- encaminhamento (*continuação*)
 - nós de trabalho 224
 - pontos de recolha 224
 - encaminhamento ad hoc 241
 - encaminhamento de documentos, *Consulte* encaminhar
 - encaminhar
 - ad hoc 241
 - atribuir privilégios 243
 - configurar 225
 - constantes 244
 - continuar um processo 237
 - criar objectos de serviço 226
 - definir um novo nó de trabalho regular 226
 - definir um novo ponto de recolha 228
 - definir um novo processo e percurso associado 233
 - definir uma lista de trabalhos 231
 - enumerar listas de trabalhos 232
 - enumerar nós de trabalho 227
 - enumerar PIDs de pacotes numa lista de trabalhos 238
 - enumerar processos de encaminhamento de documentos 240
 - exemplos 242
 - iniciar 236
 - lista de continuação 225
 - lista de trabalhos 225
 - listas de controlo de acesso (ACLs) 244
 - obter informações relativas a pacotes de trabalho 239
 - pacote de trabalho 225
 - retomar um processo 238
 - suspender um processo 237
 - terminar um processo 236
 - Enterprise Information Portal
 - arquitectura de escalões múltiplos 2
 - bases de dados 12
 - cenário de seguro 1
 - classes comuns 379
 - classes de conector 5
 - componentes 4
 - conceitos 11
 - definição de correspondências de esquema 12
 - diagrama de organização 11
 - função de fluxo de trabalho 6
 - identificador permanente (PID) 14
 - infra-estrutura da base de dados 379
 - introdução 1
 - migrar 4
 - motores de documentos 432
 - novidades 7
 - objectos de dados dinâmicos (DDO) 12
 - objectos de dados expandidos (XDOs) 13
 - servidores de conteúdos suportados 11
 - Enterprise Java Beans (EJBs) 410
 - entidades
 - armazenar pastas associadas em beans 412, 446
 - entidades (*continuação*)
 - consultar em FileNET 369
 - controladas, Content Manager Versão 8 136
 - definição de correspondências de esquema 381
 - definir correspondências em associada 16
 - enumerar em bases de dados relacionais 373
 - enumerar em FileNET 366
 - enumerar em IP OS/390 330
 - enumerar no CM 8 147
 - enumerar no CM for AS/400 336
 - enumerar no Domino.Doc 346
 - Extended Search 351
 - hierarquia de classes 245
 - identificar 14
 - ImagePlus for OS/390 336
 - mover objectos 392
 - nativa 327, 381
 - nomes de classe DK específicas para definição 384
 - pastas OnDemand como nativas 327
 - pesquisar no Domino.Doc 348
 - privilégios 137
 - entidades controladas 136
 - ENTITY_TYPE 316, 326
 - ErrorCode 141
 - ErrorState 141
 - ESCAPE,KEYWORD, consulta 208
 - ESCAPE_LITERAL 209
 - esquema, beans 444
 - eventos
 - beans 416
 - excepções, tratar 34
 - EXCEPT 202, 208
 - executar a pesquisa 33
 - executar a pesquisa com repetição de marcação 33, 186
 - executar consultas 102
 - executeWithCallback 381
 - exemplo de agrupamento em Information Mining 480
 - exemplo de categorização em Information Mining 461
 - exemplo de extracção de informações em Information Mining 475
 - exemplo de pesquisa avançada em Information Mining 494
 - exemplo de políticas 143
 - exemplo de resumo em Information Mining 469
 - exemplo de Web Crawler em Information Mining 485
 - exemplos de código, actualizações 6
 - exemplos de TxdoAdd 79
 - exponente, consulta 208
 - expressão, consulta 209
 - ExpressionList 210
 - ExpressionWithOptionalSortBy 209
 - expressões, consulta 202
 - expressões condicionais, FileNET 369
 - Extended Search
 - associar tipos de MIME 363
 - consultar com GQL 355
 - criar PIDs 357
 - Extended Search (*continuação*)
 - definir correspondências a partir de associada 16
 - enumerar entidades e atributos 351
 - enumerar servidores 351
 - identificar o DDO 356
 - introdução 350
 - obter um BLOB 362
 - obter um documento 362
 - pesquisa associada 364
 - processar campos 357
 - processar os conteúdos de 357
 - Extenders 13
- ## F
- F_DOCFORMAT 365
 - F_DOCTYPE 365
 - fechar, cursor de conjunto de resultados 387
 - ferramentas
 - Content Manager Versão 8 141
 - FeServerDefBase 394
 - fetchNext 387
 - fetchObject 387
 - ficheiro, adicionar um XDO de 55
 - ficheiros DLL
 - C++ 29
 - ficheiros exemplo
 - Content Manager Versão 8 141
 - ficheiros exemplo, localização
 - Information Mining 460
 - FileNET,
 - Consulte* Panagon Image Services
 - filtro 369
 - findObject 387
 - FLoadSampleTSQBICDL 269
 - FLOAT_LITERAL 208
 - fluxo de trabalho
 - aceder a artigos de trabalho 402
 - aceder a listas de trabalhos 401
 - beans 409
 - criar acções de Java 405
 - criar numa versão anterior do CM 308
 - definição numa versão anterior do CM 307
 - desligar de 395
 - eliminar 397
 - enumerar 398
 - enumerar IDs de artigos numa versão anterior do CM 312
 - enumerar numa versão anterior do CM 309
 - enumerar todas as listas de trabalho 400
 - enumerar todos os modelos 404
 - enviar resultados a partir de uma pasta associada 22
 - executar numa versão anterior do CM 313
 - iniciar 396
 - introdução 6, 395
 - ligação a 395
 - modelo 307
 - mover artigos 403

- fluxo de trabalho (*continuação*)
 - serviço da versão anterior do Content Manager 307
 - suporte de beans 411
 - suspender 399
 - terminar 397
- fornecedor de conteúdos em Information Mining 499
- fromXML 81
- FTSearch 349
- funções
 - nome, pesquisa de imagens 287
 - pesar, pesquisa de imagens 287
 - pesquisa de imagens 284
 - resultados máximos, pesquisa de imagens 288
 - valor, pesquisa de imagens 287
- FunctionName 210

G

- garbage collector, Java 25
- Generalized Query Language (GQL)
 - diferenças de SQL 364
 - expressões 355
- gestão de objectos
 - Java 249
 - actualizar 157, 253
 - criar 152, 249
 - eliminar 163, 256
- Gestão Hierárquica de Armazenamento (HSM) 210
- gestor de recursos, Content Manager
 - Versão 8 129
 - elaboração de versões 135
 - enumerar 145
 - trabalhar com 210
 - trabalhar com objects 211
- getCommonPrivilege 392
- getContent 388
- getContentToClientFile 388, 390
- getDatastore 391
- getOpenHandler 389, 390
- getPosition 386

H

- história
 - constante 155
 - tipo semântico 155

I

- IBM Enterprise Information Portal for Multiplatforms
 - componentes 4
 - aplicação cliente exemplo 5
 - base de dados de administração 4
 - cliente de administração 4
 - conectores 5
- ICMLogon 138
- ID de utilizador
 - definir correspondências em associada 18
- identificador de recursos uniforme (URI) 211
- identificador permanente (PID)
 - elaboração de versões 135
 - encaminhar 238
 - introdução 14
 - objecto de dados dinâmico (DDO) 12
 - objecto de dados expandido (XDO) 13
 - pesquisa de imagens 296
- identificadores
 - pesquisa de imagens 284, 287
- IDENTIFIER 209
- ImagePlus for OS/390
 - definir correspondências a partir de associada 16
- eClient 5
- enumerar atributos 330
- enumerar entidades 330
- introdução 329
- parâmetros da consulta 335
- sintaxe da consulta 334
- imldiag.log 282
- importar XML 81
- IN RANGE 368
- INBOUNDLINK 188
- indexação
 - pesquisa de imagens 301
- indexOf 389, 390
- índices
 - pesquisar texto em 109
- informações de diagnóstico, *Consulte* rastrear
- Information Mining
 - aplicações JSP 511
 - beans 414, 457
 - construir uma aplicação 457
 - descrição 5
 - exemplo de agrupamento 480
 - exemplo de elaboração de categorias 461
 - exemplo de elaboração de resumos 469
 - exemplo de extracção de informações 475
 - exemplo de importação de documentos 485
 - exemplo de pesquisa avançada 494
 - exemplo de Web Crawler 485
 - exemplos 457
 - localização dos ficheiros exemplo 460
 - o seu fornecedor de conteúdos 499
 - pesquisar por categoria 494
 - suporte de beans 411
 - utilizar a API de serviço 500
- inicialização, campo de parâmetros, OnDemand 316
- iniciar sessão, *Consulte* rastrear
- inserir, BLOB 389
- interface gráfica de utilizador (GUI), *Consulte* beans, visuais
- Interfaces de Programação de Aplicações (API)
 - Java 25
 - arquitectura 25
 - diferenças do C++ 25
 - elaboração de pacotes 26
 - pesquisa múltipla 32

- interfaces de programação de aplicações (APIs)
 - capacidades 25
 - componentes de software 142
 - conceitos 11
 - consulta online 141
- INTERGER_LITERAL 208
- INTERSECT 202, 208
- invocação de método remoto (RMI)
 - conectores de servidor de conteúdos 5
 - configurar 4
 - iniciar em Java 28
 - ligar com beans 411
 - utilizar com APIs de Java 28
- IS 208
- isBegin 386
- isCheckedOut 392
- isEnd 386
- IsFTIndexed 349
- isInBetween 386
- isOpen 387
- isOpenSynchronous 389, 391
- isScrollable 386
- isSupported 391
- isUpdatable 386
- isValid 386
- ItemAdd 138
- ItemAdminPrivSet 137
- ItemQuery 138
- ItemReadPrivSet 139
- ItemSetSysAttr 138
- ItemSetUserAttr 138
- ItemSQLSelect 138
- ItemTypeQuery 138
- iteradores
 - associados 100
 - posicionar em DKResults 101

J

- j2ee.jar 409
- Java
 - aumentar o tamanho da pilha de memória de JVM 326
 - classes alteradas 8
 - classes novas 8
 - conectores 5
 - configurar o ambiente 26
 - conjunto de ferramentas do visualizador de documentos 431
 - constantes 35
 - criar acções de fluxo de trabalho 405
 - DKConstant.h 25
 - FeServerDefBase 394
 - ficheiros de propriedades de mensagens de erro 393
 - funções de XML 25
 - garbage collector 25
 - pacotes 25
 - seqüências de escape 207
 - trabalhar com XML 80
- Java Database Connectivity (JDBC) 370
- java.lang.exception 141
- java.lang.objects 156

JavaBeans,
 Consulte beans
JDBC_DRIVER, bases de dados
 relacionais 372
JDBC_SERVER_FILE, bases de dados
 relacionais 372
JDBC_SERVER_URL, bases de dados
 relacionais 372
JSPs,
 Consulte Java Server Pages (JSPs)

K

KEY 368

L

LANG 265
LETTER 209
ligações
 artigo de descrição 173
 constantes de nome de tipo 174
 Content Manager Versão 8 133
 definir entre artigos 173
 destino 173
 internas e externas 174
 nomes de tipos 174
 obter artigos ligados 175
 origem 173
 percorrer em consulta 198
 representação no modelo de
 dados 187
 tipo de atributo 143
ligações externas 172, 174
ligações internas 174
LIKE 204, 208
LIKE, FileNET 368
linguagem de consulta 203
linguagem de marcação expansível
 Consulte XML
lista de continuação 225
lista de trabalhos 225
 definir 231
 enumerar 232
 enumerar PIDs de pacotes 238
listas de trabalhos
 aceder 401
 aceder a artigos de trabalho 402
 beans 410
 enumerar 400
 mover artigos 403
ListConstructor 210
ListContent 210
listDataSourceNames 382
listDataSources 145, 382
listEntityNames 147
listFunctions
 dkDatastoreExt 392
listMappingNames 383
listResourceMgrs 145
listWorkflowTemplates 404
listWorkLists 400
literais
 ImagePlus for OS/390 335
literais, consulta 202
literal, consulta 210

LoadFolderTSQBICDL 269
locale
 servlet 449
LocalPart 210
LogicalOrSetExpression 209
LogicalOrSetPrimitive 209
LogicalOrSetTerm 209

M

MATCH_DICT 112
MATCH_INFO 112
MAX_RESULTS, FileNET 369
membro
 adicionar 254
 remover 254
memória tampão, adicionar um XDO 54
mensagens
 ficheiros de propriedades 393
 Informações de DKException 141
mensagens de erro
 ficheiros de propriedades 393
metadados
 extração de informações 5
Microsoft Visual C++ compiler,
 Consulte Compilador visual de C++
Microsoft Visual Studio .NET 31
MOD 208
Model View Controller (MVC) 411, 440
modelo de dados, consulta
 representação de XML 195
modelo de dados, Content Manager
 Versão 8
 aplicar a linguagem de consulta 187
modelo de documento,
 Consulte partes de documento
modelo de documento associado 15
modelo de pesquisa, OnDemand
 utilizar pastas como 327
 visualizador 326
modelos
 fluxo de trabalho 404
Motor de Pesquisa de Texto
 Java
 Carregar e indexar dados 272
 Consulta booleana 261
 Consulta de proximidade 262
 Consulta de texto livre 261
 Consulta GTR 262
 Consulta híbrida 262
 Definir tamanho da pilha 247
 Sugestão de programação 265
motores de pesquisa, registar 382
moveObject 382
moveObject, dkDatastoreExt 392
msg command, FileNET 370

N

nameOfAttrStr 156
NameText 210
NATIVECONNECTSTRING, bases de
 dados relacionais 371
nó de trabalho
 definir 226
 encaminhamento 224

NoAccessACL 139
NodeGenerator 209
Nome do Tipo de Ligação 173
NONZERO 208
NoPrivSet 137
nós de trabalho
 enumerar 227
NOT 208
nota
 constante 155
 tipo semântico 155
notas, registo de beans 447
novidades na versão 8.2 7
novidades nas APIs da versão 8.2 6
NULL, consulta 208

O

Object Management Group (OMG) 12
objecto binário grande (BLOB) 388
 abrir assincronamente 389
 atualizar 388
 adicionar 388
 classes 246
 classes de DKPidXDO
 específicas 392
 classes DK específicas 388
 concatenar 389
 copiar dados 388
 criar XDOs para 50
 devolver o comprimento 389
 gerir conteúdos 388
 identificar o parâmetro identificador
 do ficheiro 389
 inserir dados de argumento 389
 métodos 388
 obter 388
 obter em Extended Search 362
 pesquisar 389
 testar com uma função de XDO 59
objecto de dados dinâmico (DDO)
 aceder a atributos 45
 aceder a conteúdos de pasta 91
 adicionar propriedades 43
 associada 15
 comparar com atributos 14
 dkDDO 379
 efetuar referências 134
 especificação de COBRA 12
 exportar XML 85
 grupos de atributos no CM 8 149
ID permanente de pesquisa de
 imagens 296
identificador permanente 14
identificar em Extended Search 356
importar XML 84
introdução 12
obter propriedades de atributos 47
obter todas as pastas que contêm 172
ordenar uma recolha 99
propriedades 248
relação com servidores de
 conteúdo 14
representar conteúdo de
 multimédia 13
representar em FileNET 365

- objecto de dados dinâmico (DDO)
 - (*continuação*)
 - representar em pesquisa de imagens 296
 - resultado da pesquisa de imagens 291
 - Objecto de Dados Dinâmicos (DDO)
 - C++
 - eliminar 49
 - Java 41
 - adicionar 44
 - apresentar 48
 - atributo, DKPARTS 86, 89
 - criar 42
 - Informações, Biblioteca Digital 143, 248
 - Informações, Motor de Pesquisa de Texto 263
 - PID 44
 - propriedades 46
 - valores de artigos de dados 45
 - objecto de sobreposição 392
 - objecto de sobreposição de formato 392
 - objecto de suporte de dados
 - adicionar em versões anteriores do CM 64
 - eliminar 69
 - nomes de exemplo de código 79
 - obter 73
 - objecto grande (LOB) 143
 - tratamento em CM anterior 247
 - objecto grande de carácter (CLOB) 390
 - adicionar conteúdo 390
 - classes 389
 - eliminar conteúdo 390
 - eliminar uma porção de 390
 - obter conteúdo 390
 - parâmetro identificador do ficheiro 390
 - objectos
 - actualizar no CM 8 157
 - armazenar valores de atributo 156
 - beans 446
 - Content Manager Versão 8 133
 - gerir no Domino.Doc 346
 - pesquisa de texto de conteúdos 189
 - objectos de dados expandido, *Consulte* XDO
 - objectos de dados expandidos, *Consulte* XDO
 - Obtenção
 - C++
 - partes 258
 - Java 257
 - partes 258
 - pastas 259
 - obter, BLOB 388
 - obter informações 239
 - OnDemand
 - activar o modo de pastas 326
 - anotações 327
 - apresentar atributos 322
 - apresentar documentos 322
 - consultar um grupo de aplicações 320
 - definir correspondências a partir de associada 16
 - OnDemand (*continuação*)
 - desligar de 316
 - enumerar grupos de aplicações 316
 - enumerar informações relativas ao servidor 316
 - enumerar pastas 319
 - introdução 315
 - ligação a 316
 - nomes de propriedades 315
 - obter documentos 319
 - pesquisar assincronamente 326
 - pesquisar um documento 319
 - rastrear 327
 - representar servidores e documentos 315
 - utilizar pastas como modelo de pesquisa 327
 - Open Database Connectivity (ODBC) 370
 - cadeias de configuração 51
 - operadores
 - consulte de ImagePlus for OS/390 335
 - Domino.Doc 349
 - FileNET 368
 - Generalized Query Language (GQL) 364
 - pesquisa paramétrica 188
 - OptionalExpressionList 210
 - OptionalPredicateList 209
 - OR 208
 - output_option 38
- ## P
- pacote cs 26
 - pacote de trabalho 225, 239
 - enumerar cadeias de PID numa lista de trabalhos 238
 - introdução 395
 - pacotes
 - cliente de Java 25
 - cliente e servidor (cs) 26
 - hierarquia em Java 26
 - servidor de Java 25
 - padrão 286
 - páginas
 - beans 448
 - FileNET 365
 - Páginas de Servidor Java (JSPs)
 - servlet 449
 - palavra-passe
 - alterar 382
 - alterar em beans 428
 - criar no Content Manager Versão 8 145
 - definir correspondências em associada 18
 - palavras-chave, Domino.Doc 346
 - Panagon Capture Software 366
 - Panagon Image Services
 - classes 364
 - classes de documentos 366
 - consultar 368
 - detecção e correcção de problemas 370
 - enumerar entidades e atributos 366
 - Panagon Image Services (*continuação*)
 - exemplos 368
 - introdução 364
 - operadores 368
 - parâmetros de pesquisa opcionais 369
 - representar documentos e páginas 365
 - sintaxe da consulta 368
 - tipos de dados suportados 365
 - partes
 - actualizar em versões anteriores do CM 253
 - CM anterior 248
 - elaboração de versões 219
 - Extended Search 357
 - partes de artigo
 - comparar com DDOs e XDOs 14
 - partes de documento
 - constante 145
 - Content Manager Versão 8 131
 - tipo de artigo 145
 - passo, consulta 209
 - pastas
 - aceder 91
 - activar o modo em OnDemand 326
 - actualizar 382
 - actualizar em versões anteriores do CM 255
 - adicionar 382
 - adicionar conteúdos a 168
 - adicionar membros a 392
 - apresentar em beans 418, 424
 - beans 447
 - comportamento do Content Manager 91
 - Content Manager Versão 8 134
 - criar no CM 8 167
 - dar saída e entrada 392
 - eliminar 382
 - enumerar em OnDemand 319
 - fluxo de trabalho da versão anterior do CM 307
 - gerir no servlet controlador 454
 - mover 382
 - obter 382
 - obter conteúdos 172
 - obter todas as pastas com um DDO específico 172
 - pastas OnDemand como entidades nativas 327
 - remover conteúdos de 170
 - remover membros de 392
 - representar no CM anterior 249
 - tipo semântico 155
 - PENDING, ImagePlus for OS/390 336
 - pEnt, C++ 41
 - perfis, Domino.Doc 346
 - permissões, *Consulte* privilégios
 - pesquisa de cor
 - histograma 285, 289, 290
 - média 285, 289, 290
 - posicional 286, 290
 - textura 286, 290
 - pesquisa de imagens 283
 - aplicações 286

pesquisa de imagens (*continuação*)
 associada 19
 atributos 296
 avaliar uma consulta a partir de um servidor de conteúdos 299
 bases de dados 284, 285
 carregar dados para indexação 300
 catálogos 284, 285
 closeCatalog 291
 consultar 297
 cor de histograma 285, 289, 290
 cor média 285, 289, 290
 cor posicional 286, 290
 criar consultas 287
 critérios de pesquisa 287
 definição 33
 desligar de 291
 diagrama 285
 enumerar bases de dados 293
 enumerar catálogos 293
 enumerar funções 293
 enumerar servidores 292
 executar uma consulta a partir de um servidor de conteúdos 298
 funções 284, 287
 identificadores 287
 indexar um XDO 301
 introdução 6
 ligação a 291
 listDatabases 291
 máximo de resultados 288
 openCatalog 291
 representar informações com um DDO 296
 sintaxe da consulta 288
 textura 286, 290

pesquisa de texto
 avaliar uma consulta a partir de um servidor de conteúdos 112
 beans 421
 caracteres globais 190
 compreender 189
 consulta básica 190
 consultar a partir de um servidor de conteúdos 111
 consultar em vários índices 109
 consultar vários índices 109
 criar o modelo de documentos 275
 definição 33
 definir regras de indexação 277
 destacar 112
 enumerar o modelo de documentos 273
 execução da consulta 109
 exemplo de consulta 199
 formular uma cadeia de consulta 109
 funções de gestão 261
 identificador permanente (PID) 263
 iniciar o processo de indexação 280
 OnDemand 315
 operadores 190
 pesquisa avançada 191
 pesquisar conteúdos de objectos 189
 pesquisar documentos 190
 pesquisar documentos estruturados 273, 282
 rastrear 33

pesquisa de texto (*continuação*)
 recolhas consultáveis 126
 resultado básico 191
 sintaxe da consulta 190
 tornar os atributos definidos pelo utilizador pesquisáveis 190

pesquisa de texto de resulatado básico 190

pesquisa paramétrica
 associada 19
 avaliar uma consulta a partir de um servidor de conteúdos 108
 compreender 188
 consultar a partir do servidor de conteúdos 106
 consultar com vários critérios 103
 consultar no CM para AS/400 343
 definição 32
 executar a consulta 104
 formular uma cadeia de consultas 103
 formular vários critérios 103
 operadores 188
 Panagon Image Services 368
 rastrear 34

pesquisar
 beans 412, 444, 446
 classe de DKResults 380
 compreender a linguagem de consulta 185
 módulos de API 143

ponto de recolha
 definir 228
 descrição 224

posicionamento relativo, cursor de conjunto de resultados 124

Predicado 209

privilegio comum, dkDatastoreExt 392

privilégios
 acesso a dados 137
 ACLs pré-configuradas 139
 apresentar propriedades de um conjunto 179
 atribuir para encaminhar 243
 beans 412, 447
 códigos 138
 conjunto 137
 conjuntos predefinidos do Content Manager Versão 137
 Content Manager Versão 8 136
 criar no CM 8 176
 criar um conjunto 177
 definidos pelo utilizador 137
 listas de controlo de acesso (ACLs) 138
 tipos de conjuntos 137
 utilizadores e grupos de utilizadores no CM 8 138

privilégios de utilizador
 Content Manager Versão 8 136

PrivSetCode 137

processadores de enlace, Domino.Doc 346

processar fila de imagens 301

processos
 continuar 237

processos (*continuação*)
 definir para um percurso associado 233
 encaminhar 225
 enumerar 240
 retomar 238
 suspender 237
 terminar 236
 propriedade Mime2App 426
 PublicReadACL 139

Q

QbColor 289
 QbColorFeatureClass 285, 289
 QbColorHistogramFeatureClass 285, 289, 290
 QbDraw 290
 QbDrawFeatureClass 286, 290
 QbHistogram 289
 QbTexture 290
 QbTextureFeatureClass 286, 290
 QName 210

R

rastrear
 beans 417, 444
 contentor de código de retorno 141
 estado de erro 141
 ferramentas exemplo 141
 FileNET 370
 JNI, servlet 451
 OnDemand 327
 pesquisa de texto 33
 pesquisa paramétrica 34
 pilha de memória 141
 servlet controlador 451
 tratar erros com DKException 140

rastreio de pilha de memória 141

rastreio JNI, servlet 451

receptores de sessão, beans 416

recolha de memória
 adicionar um XDO a 78
 alterar para um XDO 80

Recolha passível de consulta
 Java 125
 avaliar 126
 obter resultados 125
 passível de consulta vs aperfeiçoado 127
 Sugestões de programação 127

recolhas
 associada 100
 classe de dkCollection 380
 eliminar 99
 Extended Search 357
 ordenar 99

recolhas e iteradores
 C++
 gestão de memória 98
 Java 95
 iterador sequencial 96
 recolha sequencial 96

REFERENCEDBY 188
 REFERENCER 188

- referência de programação de aplicações (APR) 141
- referências
 - Content Manager Versão 8 133
 - percorrer em consulta 200
 - tipo de atributo 143
- registerMapping 382
- registerServices 382
- removeAllElement 99
- removeFromFolder 392
- removeMember 254
- remover, BLOB
 - objecto binário grande (BLOB)
 - remover uma porção 389
- remover, CLOB 390
- remover alterações 381
- repetição de marcação, executar 33, 186
- replaceElementAt 99
- ReplayAction, servlet 449
- resultado básico, pesquisa de texto
 - pesquisa de texto
 - contém texto 191
- retomar um fluxo de trabalho 400
- retrieveFromOverlay 392
- retrieveObject 382

S

- saídas do utilizador
 - administração do sistema 23
 - fluxo de trabalho 405
- SAttributeDefinitionCreationICM 131
- SConnectDisconnectICM 141
- SDocModelItemICM 134
- SDocRoutingDefinitionCreationICM 226, 227, 230, 232, 236
- SDocRoutingListingICM 228, 233, 239, 241
- SDocRoutingProcessingICM 236, 237, 238, 240
- SequencedValue 209
- sequenciação múltipla 25
- sequências de escape, exemplos de consulta 203
- Serviço de Objecto Permanente, COBRA 12
- Serviço de Objecto Permanente (PDS), COBRA 12
- serviços de fluxo de trabalho de EIP
 - Consulte* fluxo de trabalho
- servidor controlador
 - predefinições 450
- servidor de bibliotecas, Content Manager Versão 8 129
 - elaboração de versões 135
 - enumerar 145
- servidor de RMI,
 - Consulte* invocação do método remoto (RMI)
- servidores de conteúdos
 - aceder a informações em 246
 - actualizar um documento ou pasta 382
 - adicionar membros a pastas 392
 - adicionar um documento ou pasta 382
 - alterar palavra-passe 382

- servidores de conteúdos (*continuação*)
 - associada 15
 - avaliar uma consulta de imagens a partir de 299
 - avaliar uma consulta de texto a partir de 112
 - avaliar uma consulta paramétrica a partir de 108
 - bases de dados relacionais 370
 - beans 411
 - classes de extensão 391
 - classes de gestão do utilizador 393
 - classes de XDO 392
 - classes DK específicas para definição 385
 - conectores suportados 5
 - consultar 381
 - Content Manager, versões anteriores 247
 - Content Manager para AS/400 336
 - Content Manager Versão 8 129
 - criar 245
 - criar um DDO em 382
 - cursor do conjunto de resultados 246
 - dar saída de documentos ou pastas 392
 - definir 245
 - devolver nomes 382
 - devolver os objectos de ID de utilizador 382
 - Domino.Doc 344
 - eliminar um documento ou uma pasta 382
 - enumerar nomes de função 392
 - executar transacções 381
 - executar uma consulta de imagens a partir de 298
 - executar uma consulta de texto a partir de 111
 - executar uma consulta paramétrica a partir de 106
 - expandir objectos 382
 - expandirfeServerDefBase para a personalização de conectores 394
 - Extended Search 350
 - fazer referência a 391
 - gerir conteúdo de CLOB 390
 - gerir dados 11
 - gerir dados binários 388
 - hierarquia de definição de dados 247
 - identificadores permanentes (PIDs) 14
 - identificar onde o cursor de conjunto de resultados pertence 387
 - ImagePlus for OS/390 329
 - introdução 1
 - ligação 381
 - mover um documento ou pasta 382
 - nomes de classe 245
 - objecto de dados dinâmico (DDO) 12
 - objecto de dados expandido (XDO) 13
 - obter o objecto de administração 213
 - obter o objecto de sobreposição de formato 392
 - obter o privilégio comum 392
 - obter um documento ou pasta 382

- servidores de conteúdos (*continuação*)
 - OnDemand 315
 - Panagon Image Services 364
 - personalizar 378
 - pesquisa de imagens 283
 - registar em associada 18
 - registar informações de definição de correspondências 382
 - relação com DDOs 14
 - remover membros de pastas 392
 - servidores suportados pelo EIP 11
 - terminar ligação 381
 - utilizar com RMI 28
 - valores de servlet controlador 450
- servlet,
 - Consulte* servlet controlador
- servlet controlador
 - acções 448, 449
 - agrupamento de ligações 448
 - convenções 450
 - definições de JSP 449
 - expandir 449
 - ficheiro de propriedades 449, 452
 - gestão de sessões 449
 - limpar 449
 - locale 449
 - matriz de função de conjunto de ferramentas 455
 - páginas que apresentam mensagens de erro 450
 - parâmetro de separador de nomes 452
 - parâmetros 450, 452
 - parâmetros de conversão 452
 - parâmetros de pedido
 - acção 453
 - artigos 453
 - geral 452
 - gerir conteúdos 454
 - pastas 454
 - pesquisar 453
 - relacionados com documentos 454
 - relacionados com ligações 453
 - reply 452
 - rastrear 451
 - referência 450
 - replay 449
 - tempos de execução de serviço 451
 - utilizar 448
 - valores de agrupamento de ligações 450
- sessões
 - servlet 449
- sessões expiradas, servlet 449
- setClassOpenHandler 389, 390
- setContent 388
- setContentFromClientFile 388, 390
- setData 154
- setDatastore 391
- setInstanceOpenHandler 389, 390
- setPosition 386
- setToBeIndexed 301
- setToFirstCollection 101
- setToLastCollection 101
- setToNext 387
- setToNextCollection 101

- setToPreviousCollection 101
- SFolderICM 134, 167, 172
- Sistema de Entrada de Lotes, FileNET 366
- SItemCreationICM 134
- SItemDeletionICM 163
- SItemRetrievalICM 163
- SItemTypeCreationICM 131
- SItemTypeRetrievalICM 166
- SItemUpdateICM 160, 164
- SLinksICM 133, 174, 175
- SLinkTypeDefinitionCreationICM 175
- SORTBY 208, 209
- SortSpec 209
- SortSpecList 209
- SOURCEITEMREF 188
- SReferenceAttrDefCreationICM 134, 188
- SResourceItemMimeTypeICM 160
- SSearchICM 161
- streaming doc services 432
- STRING_LITERAL 208
- sub-entidades
 - enumerar no Domino.Doc 346
- subsistema da consola, definir em Windows 32
- subString 389, 390
- SuperUserACL 139
- suspender um fluxo de trabalho 399
- suspender um processo 237
- svWF 398
- SYSREFERENCEATTRS 188
- System Managed Storage (SMS), Content Manager Versão 8 129
- SystemAdmin 138
- SystemAdminPrivSet 137
- SystemDefineItemType 138

T

- tabela FASERVERTYPES 394
- taglib.tld 409
- TARGETITEMREF 198
- TCallbackOD 327
- TCheckStatusTS 282
- TCP/IP
 - pesquisa de texto 264
- terminologia do nome da classe 245
- TExportICM 80
- TExportPackageICM 80
- text information extender (TIE)
 - Net Search Engine (NSE) 33
- textura
 - contraste 286
 - definição 286
 - direccionabilidade 286
 - espessura 286
 - exemplo de consulta 290
 - valores válidos 290
- TImageAnnotation 442
- TImportICM 80
- tipo semântico
 - definição 154
 - definido pelo utilizador 155
 - tipos predefinidos 155
- TipoRep 269
- tipos de artigo
 - classificações 145

- tipos de artigo (*continuação*)
 - componentes, raiz e descendentes 132
 - consultar vários 198
 - Content Manager Versão 8 131
 - criar no Content Manger Versão 8 145
 - criar uma definição 158
 - elaboração de versões 135, 165
 - enumerar 147
 - enumerar atributos para 151
- tipos de MIME
 - associar em Extended Search 363
 - definir para um artigo de recurso 160
- toXML 81
- transacções, Content Manager Versão 8 220
 - considerações 221
 - dar entrada e saída 222
 - lista 223
 - precaução em relação a transacções explícitas 222
 - processar 223
- tratar, cursor de conjunto de resultados 387
- TRetrieveFolderWithCallbackOD 327
- TRetrieveWithCallbackOD 327
- TxdAsyncRetDL 247

U

- UNICODE_CHARACTER 208
- UNION 202, 208
- unlockCheckedOut 392
- unRegisterMapping 382
- unregisterServices 382
- UpdateFTIndex 349
- updateObject 157, 382
- utilizadores
 - controlar acesso no CM 8 138

V

- valueObj 156
- versão, consultar 201
- versão 8.2, novidades 7
- versão 8.2, novidades de APIs 6
- VideoCharger 129
- Visual Studio.NET 31
- visualizador de documentos genéricos
 - Consulte conjunto de ferramentas do visualizador de documentos
- visualizadores, documento de Java
 - Consulte conjunto de ferramentas do visualizador de documentos

W

- wakeUpService 272
- WAL_PRs_Parse 368
- web crawler
 - beans 415
 - introdução 5
- web.xml 409

- WebSphere Studio Application Developer
 - construir beans 408
- Windows
 - definir o subsistema da consola 32
 - ficheiros DLL de C++ 29
 - iniciar o servidor de RMI 28

X

XDO

- actualizar 58
- adicionar a partir da memória tampão 54
- adicionar a um conjunto de memória 78
- adicionar de um ficheiro 55
- adicionar um objecto de anotação a 56
- adicionar um objecto de suporte de dados 64
- alterar o conjunto de armazenamento 80
- classe 380
- classes DK específicas 392
- comparar com atributos 14
- criar um artigo de recurso 159
- definir o tipo de MIME 160
- dkBlob 388
- dkXDO 380
- eliminar 58, 69
- exemplos 58
- exportar XML 85
- FileNET 366
- hierarquia de tipo de classes 159
- indexar em versões anteriores do CM 51
- indexar uma pesquisa de imagens 301
- introdução 13
- invocar uma função 59
- Java 50
 - autónomo 54
 - DDO, parte de 52
 - indexação 268
 - PID 50
 - propriedades de dados 51
 - Sugestões de programação 52
- membros 390
- objecto grande 143
- objectos 133
- obter 58
- obter um objecto de suporte de dados 73
- pesquisar texto 200
- representar conteúdo de multimédia 13

XML

- armazenar no CM 82
- biblioteca de identificadores de beans 409
- consulta de W3C 185
- definição do tipo de documento (DTD) para importar 81
- exemplos 83
- exportar 85
- extrair
 - endereço da Web 84

- XML (*continuação*)
 - extrair (*continuação*)
 - ficheiro 84
 - memória tampão 84
 - ferramentas exemplo 80
 - importar 81
 - importar para o CM 84
 - introdução 80
 - representação do modelo de dados de
exemplos de consulta 195
 - suporte de Java 25
- XQuery Path Expressions (XQPE) 185



Número do Programa: 5724-B43

SC17-5414-01



Spine information:

IBM Content
Manager for Multiplatforms/IBM
Information
Integrator for Content



Manual de Programação de Aplicação de Estações de
Trabalho

Versão 8 Edição 2