IBM DB2 Information Integrator
OmniFind Edition

エンタープライズ・サーチ
プログラミング・ガイドおよび API リファレンス

バージョン 8.2

SD88-6375-00
(英文原典：SC18-9284-00)

IBM DB2 Information Integrator
OmniFind Edition

エンタープライズ・サーチ
プログラミング・ガイドおよび API リファレンス

バージョン 8.2

SD88-6375-00
(英文原典：SC18-9284-00)

お願い

本書および本書で紹介する製品をご使用になる前に、『特記事項』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC18–9284–00
IBM DB2 Information Integrator
OmniFind Edition
Programming Guide and API Reference for Enterprise Search
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.11

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

© Copyright IBM Corp. 2004

第 1 章 本書について

本書では、エンタープライズ・サーチで提供される Java™ アプリケーション・プログラミング・インターフェース (API) の 使用方法について説明します。 API は、検索アプリケーションの作成用ツールを提供します。

本書には、以下の内容が記載されています。

- エンタープライズ・サーチ・アプリケーションの概要
- API コンポーネントの説明
- サンプル・コード

DB2® Information Integrator OmniFind Edition は、エンタープライズ・サーチと呼ばれるテクノロジーを提供します。エンタープライズ・サーチ・コンポーネントは、IBM® DB2 Information Integrator OmniFind Edition (DB2 II OmniFind) のインストール時にインストールされます。CD ラベルや特定の製品コンポーネントに言及する場合を除き、DB2 II OmniFind の資料内では『DB2 II OmniFind』の代わりに『エンタープライズ・サーチ』という用語が使用されています。

本書の対象読者

本書は、カスタマイズされたエンタープライズ・サーチ・アプリケーションを作成したいアプリケーション・プログラマーの方を対象にしています。

本書では、読者が以下のスキルを持っていることを前提としています。

- アプリケーション・プログラミングを熟知している
- Java アプリケーションのコーディング経験がある

本書の追加情報の入手先

詳しくは、以下のリソースを参照してください。

製品情報

DB2 Information Integrator OmniFind Edition に関する情報は、以下から入手できます。

Web で www.ibm.com/software/data/integration/db2ii/support.html にアクセスしてください。このサイトには、以下に関する最新情報が掲載されています。

- テクニカル・ライブラリー
- 資料のオーダー
- クライアントのダウンロード
- ニュースグループ
- フィックスパック
- ニュース

© Copyright IBM Corp. 2004

- Web リソースへのリンク

第 2 章 エンタープライズ・サーチ API

エンタープライズ・サーチ API は、エンタープライズ・サーチ・コレクションにおける文書の追加、文書の除去、および検索を行う一連の Java API です。

API の説明

IBM 検索 および索引 API

IBM 検索および索引 API (SIAPI) は、カスタム検索アプリケーションの作成に使用します。 SIAPI のエンタープライズ・サーチ・インプリメンテーションによって、検索サーバーへのリモート・アクセスが可能になります。これらの API によって、ユーザーは、検索要求のサブミット、検索結果の処理、分類法ツリーのブラウズを行うことができます。

データ・リスナー API

データ・リスナー API は、エンタープライズ・サーチ・コレクションに対する文書の追加または除去に使用します。

データ・リスナーは、外部データ・ソース・クローラーからデータを受け取るエンタープライズ・サーチのコンポーネントです。データ・ソース・クローラーは、データ・リスナーに接続して、ターゲット・コレクションにデータをプッシュします。

また、データ・リスナー・コンポーネントは、エンタープライズ・サーチ・キューに追加され、データがターゲット・コレクションにプッシュされる前に内部でクロールされる文書を受信することができます。

セキュリティー

エンタープライズ・サーチとブラウズ API は、各検索ノードの WebSphere® Application Server にインストールされている ESSearchServer エンタープライズ・アプリケーションとリモートで通信します。

この機密保護方式において、WebSphere Application Server は、すべての HTTP 要求に対して、正当なユーザー名とパスワードを要求します。入力されたユーザー名とパスワードは、 WebSphere 管理コンソールを使用して構成されるアクティブ・ユーザー・レジストリー内で有効でなければなりません。正当なユーザーの信任状が含まれていない要求は、リジェクトされます。

エンタープライズ・サーチ SIAPI インプリメンテーションは、ユーザーに代わって、ユーザー名とパスワードを自動的に設定します。ユーザーは、ユーザー・アプリケーションで ApplicationInfo クラスのインスタンスを作成する際に、ユーザー名とパスワード値を指定することができます。 61 ページの『最低限必要なアプリケーション』の例を参照してください。

検索アプリケーション名およびパスワードは、WebSphere 認証に使用するリポジトリーと同じリポジトリーに保管する必要があります。

制約事項:

© Copyright IBM Corp. 2004

検索 API の制約事項は、次のとおりです。

- HTTP BASIC 認証をサポートする
- HTTPS (SSL v2 または v3) はサポートしない

第 3 章 検索および索引 API (SIAPI)

IBM 検索および索引 API (SIAPI) は、ユーザーが、コレクションおよび分類法を検索あるいはブラウズできるプログラミング・インターフェースです。

SIAPI は、ユーザーが 1 つのプログラムを作成するだけで、さまざまな IBM バックエンド検索プロダクトを検索できるようにする、統一プログラミング・インターフェースを提供します。

SIAPI は、次のようなタスクをサポートします。

- 索引の検索
- 検索結果セットに戻される情報のカスタマイズ
- 分類法の検索およびブラウズ

SIAPI アプリケーションの構造

SIAPI アプリケーションは、以下のタスクで構成されます。

- SIAPI インプリメンテーション・ファクトリー・オブジェクトの取得
- SearchService オブジェクトの取得
- Searchable オブジェクトの取得
- 照会の実行
- 照会結果の処理

SIAPI インプリメンテーション・ファクトリー・オブジェクトの取得

SIAPI ベースの検索アプリケーションでは、まず最初に、インプリメンテーション・ファクトリー・オブジェクトを取得します。

```
SearchFactory factory =
SiapiSearchImpl.createSearchFactory
    ("com.ibm.es.api.search.RemoteSearchFactory");
```

SIAPI は、ファクトリー・ベースの Java API です。検索アプリケーションで使用するすべてのオブジェクトは、 SIAPI オブジェクト・ファクトリー・メソッドを呼び出して作成されるか、またはファクトリー生成オブジェクトのメソッドを呼び出して戻されるかのいずれかです。異なるファクトリーをロードすることによって、SIAPI インプリメンテーション間を容易にスイッチすることができます。

エンタープライズ・サーチ SIAPI インプリメンテーションは、com.ibm.es.api.search.RemoteSearchFactory クラスに提供されています。

SearchService オブジェクトの取得

SearchService オブジェクトを取得するには、ファクトリー・オブジェクトを使用します。ユーザーは、SearchService オブジェクトによって、検索サーバー上の検索可能なコレクションにアクセスすることができます。

© Copyright IBM Corp. 2004

SearchService オブジェクトは、エンタープライズ・サーチ検索サーバーのホスト名とポート、およびエラー・メッセージの受信に必要なロケールを指定して構成する必要があります。 ロケールは、4 文字または 5 文字の Java ストリングです。たとえば、英語の場合は ″enUS″ または ″en_US″ です。

構成パラメーターは java.util.Properties に設定します。パラメーターは、SearchService オブジェクトを生成する getSearchService ファクトリー・メソッドに渡されます。

```
Properties configuration = new Properties();
configuration.setProperty("hostname", "es.mycompany.com");
configuration.setProperty("port", "80");
configuration.setProperty("locale", "en_US");
SearchService searchService =
 factory.getSearchService(configuration);
```

Searchable オブジェクト の取得

Searchable オブジェクトを取得するには、 SearchService オブジェクトを使用します。 Searchable オブジェクトは、検索サーバー上の検索可能なコレクションと関連しています。ユーザーは、Searchable オブジェクトによって、照会を実行し、関連したコレクションに関する情報を取得することができます。各エンタープライズ・サーチ・コレクションには ID があります。

ユーザーが、 Searchable オブジェクトを要求する際には、アプリケーション ID を使用して、アプリケーションを識別する必要があります。適切なアプリケーション ID については、エンタープライズ・サーチ管理者に問い合わせてください。

検索サーバーのグローバル・セキュリティーがオンに構成されている場合は、パスワードを提示する必要があります。パスワードは、アプリケーションの認証に使用されます。詳しくは、 3 ページの『セキュリティー』を参照してください。

```
ApplicationInfo appInfo = factory.createApplicationInfo("my_application_id");
appInfo.setPassword("my_password");
Searchable searchable =
 searchService.getSearchable(appInfo, "some_collection_id");
```

アプリケーションで使用可能なすべての Searchable オブジェクトを取得するには、getAvailableSearchables メソッドを呼び出します。

```
Searchable[] searchables =
 searchService.getAvailableSearchables(appInfo);
```

Searchable オブジェクトについて詳しくは、 25 ページの『Searchable インターフェース』を参照してください。

照会の実行

Searchable オブジェクトを取得したら、その Searchable オブジェクトに対して照会を実行することができます。 Searchable オブジェクトに照会を実行するには、次のようにします。

- Query オブジェクトを作成する。
- Query オブジェクトをカスタマイズする。
- Searchable オブジェクトに対して Query オブジェクトを実行する。
- ResultSet オブジェクトにカプセル化された照会結果を取得する。

```
String queryString = "big apple";
Query query = factory.createQuery(queryString);
query.setRequestedResultRange(0, 10);
ResultSet resultSet = searchable.search(query);
```

詳しくは、『照会動作の制御』および 9 ページの『照会構文』を参照してください。

照会結果の処理

ResultSet および Result インターフェースによって、照会結果にアクセスできます。

```
Result[] results = resultSet.getResults();
for ( int i = 0 ; i < results.length ; i++ ) {
 System.out.println
( "Result " + i + ": " + results[i].getDocumentID()
  + " - " + results[i].getTitle() );
}
```

SIAPI には、 48 ページの『ResultSet インターフェース』、および 45 ページの『Result インターフェース』に記載されている個々のオブジェクトと対話するためのさまざまなメソッドがあります。

照会動作の制御

QUERY インターフェースに属する以下のメソッドによって、照会の処理方法やそれぞれの結果に戻されるメタデータの内容など、あらゆる分野の照会動作を制御することができます。

表 1. 照会動作メソッド

メソッド	説明
setQueryLanguage (String language)	検索サーバーで使用する、コレクション・デフォルト言語以外の言語を指定します。たとえば、英語の場合、照会言語パラメーターは en-US です。中国語では、中国語 (簡体字) の場合は zh-CN、中国語 (繁体字) の場合は zh-TW を使用します。
setRequestedResultRange (int fromResult, int numberOfResults)	戻される結果の範囲を制御します。 **fromResult** 値は、ランク付けされた文書のうち、どの文書から結果セットを開始するかを制御します。たとえば、値 0 は、照会結果の最初の文書を要求していることを意味します (**fromResult** は、 0 または numberOfResults の倍数でなければなりません)。 **numberOfResults** 値は、結果の現行ページにいくつの結果を戻すかを制御します。最大値は 100 です。
setReturnedAttribute (int attributeType, boolean isReturned)	各 Result オブジェクトで戻される定義済み結果属性値を使用可能/使用不可に設定します。 デフォルトでは、エンタープライズ・サーチは、メタデータ・フィールド属性 (RETURN_RESULT_FIELDS) 以外のすべての定義済み結果属性値を戻します。

表 1. 照会動作メソッド (続き)

メソッド	説明
setReturnedFields (java.lang.String[] fieldNames)	Result オブジェクトに戻されるメタデータ・フィールドを制御します。 デフォルトでは、エンタープライズ・サーチはメタデータ・フィールドを戻しません。
setSiteCollapsingEnabled (boolean value)	上位の結果に、同じ Web サイトからの結果を 3 個以上含めるかどうかを指定します。 たとえば、特定の照会で www.ibm.com から 100 個の結果が戻された場合、サイト縮小が使用可能に設定されていると、ResultSet には、 上位の結果に 100 個の結果のうち 2 個の結果しか含まれていません。そのサイトの他の結果は、他のサイトの結果がリストされた後に表示されます。 同じサイトからより多くの結果を検索するには、照会ストリングに www.ibm.com サイトを追加して、同じ照会を再実行します。
setSortKey (String sortKey)	ソート・キーを指定します。以下の事前定義ソート・キー値が、 com.ibm.siapi.search.BaseQuery に定義されています。 • SORT_KEY_NONE • SORT_KEY_DATE • SORT_KEY_RELEVANCE 上記のほかに、検索するコレクションの有効な数値フィールド名をソート・キーとして指定できます。デフォルトのソート・キーは SORT_KEY_RELEVANCE です。
setSortOrder (int sortOrder)	SORT_ORDER_ASCENDING または SORT_ORDER_DESCENDING のいずれかのソート順序を指定します。 ソート・キーが SORT_KEY_RELEVANCE または SORT_KEY_NONE の場合には、ソート順序は無視されます。
setSortPoolSize (int poolSize)	上位関連結果のいくつをソートして、結果セットに戻すかを制御します。値は 1 から 500 までです (デフォルトのソート・プール・サイズは 500 です)。それ以外の値を指定すると、検索サーバーによって SiapiException がスローされます。 sortKey が SORT_KEY_RELEVANCE または SORT_KEY_NONE の場合には、ソート・プール・サイズは無視されます。
setPredefinedResultsEnabled (Boolean value)	照会結果で、通常の結果のほかに定義済みの結果を含めるかどうかを指定します。デフォルトでは、定義済みリンクが使用可能になります。

表 1. 照会動作メソッド (続き)

メソッド	説明
setSpellCorrectionEnabled (Boolean value)	照会結果に照会のスペル修正の提案を含めるかどうかを指定します。デフォルトでは、スペル修正は使用不可になります。
setResultCategoriesDetailLevel (int detailLevel)	照会結果に要求されるカテゴリー詳細レベルを指定します。このメソッドは、カテゴリー属性 (RETURN_RESULT_CATEGORIES) が使用可能な場合に使用されます。デフォルト値は、RESULT_CATEGORIES_ALL です。

照会構文

エンタープライズ・サーチは、以下の構文を使用して、照会を処理します。

検索文字

エンタープライズ・サーチ構文では、検索照会をより詳細に行うために、特殊文字を使用します。

文字	使用法
+	用語の前に正符号 (+) を付けると、その用語に一致する用語が含まれている文書を検索することを示します。正符号は、用語の直前にタイプしてください。 **照会:** `+computer +software` **結果:** この照会は、 `computer` と `software` という用語が含まれている文書を戻します。
-	用語の前に負符号 (-) を付けると、その用語が含まれていない文書を検索することを示します。負符号は、用語の直前にタイプしてください。 **照会:** `computer -hardware` **結果:** この照会は、`computer` という用語が含まれており、 `hardware` という用語が含まれていない文書を戻します。
^	用語の前に曲折アクセント記号 (^) を付けると、(照会で) 曲折アクセント記号 (^) を付けて指定した用語に、曲折アクセント記号を付けずに指定した用語が少なくとも 1 つ以上続いている複合語が含まれている文書を検索することを示します。 **照会:** `^computer science software` **結果:** この照会は、`computer` という用語を含む次のような複合語が含まれている文書を戻します。`computer science`、`computer software`、または `computer science software`。

文字	使用法
( )	括弧 ( ) を使用すると、括弧内に指定した 1 つ以上の用語が含まれている文書を検索することを示します。 括弧内には、正符号 (+) や負符号 (-) を指定しないでください。 括弧内の 用語は、OR や垂直バー (|) で区切ります。 **照会:** (computer OR software) または (computer | software) **結果:** この照会は、 computer か software、または両方の computer software という用語が含まれている文書を検索します。
″ ″	引用符 (″) を使用すると、引用符内に指定したフレーズ (句) と全く同じフレーズが含まれている文書を検索することを示します。 **照会:** "computer software programming" **結果:** この照会は、computer software programming という用語と全く同じ用語が含まれている文書を検索します。
~	用語の前に波形記号 (~) を付けると、言語学上の基本型 (見出し語または語幹ともいいます) が同じである用語が含まれている文書を検索することを示します。 **照会:** "~apples" **結果:** この照会は、apples または apple という用語が含まれている文書を検索します (apple は apples の基本型であるため)。
=	用語の前に等号 (=) を付けると、その用語と全く同じ用語が含まれている文書を検索することを示します。 **照会:** "=apples" **結果:** この照会は、apples という用語が含まれている文書を検索しますが、apple という用語が含まれている文書は検索しません。
site:*text*	Web コンテンツが含まれているコレクションを検索する場合に、 **site** キーワードを使用して、特定のドメインを検索します。たとえば、特定の Web サイトの全ページを戻すことができます。 サイト照会に接頭部として http:// は付けないでください。 **照会:** +laptop site:www.ibm.com **結果:** この照会は、laptop という用語が含まれている、www.ibm.com ドメイン上のすべての文書を検索します。

文字	使用法
url:*text*	Web コンテンツが含まれているコレクションを検索する場合に、 **URL** キーワードを使用して、URL の一部に特定のワードが含まれている文書を検索します。 **照会:** `URL:support` **結果:** この照会は、 URL の一部に support という値が含まれている文書 (`http://www.ibm.com/support/fr/` など) を検索します。
link:*text*	Web コンテンツが含まれているコレクションを検索する場合に、 link キーワードを使用して、特定の Web ページへの**リンク** が少なくとも 1 つ以上含まれている文書を検索します。 **照会:** `link:http://www.ibm.com/us` **結果:** この 照会は、ページ `http://www.ibm.com/us` へのリンクが 1 つ以上含まれている文書をすべて検索します。
field:*text*	コレクション内の文書にフィールド (リレーショナル・データベースの列など) が含まれており、コレクション管理者がこれらのフィールドをフィールド名によって検索可能にしている場合に、コレクション内の特定のフィールドを照会できます。 **照会:** `lastname:smith div:software` **結果:** この照会は、 ソフトウェア部門 (`div:software`) で働いており、 ラストネームが Smith (`lastname:smith`) である従業員に関する文書をすべて検索します。
docid:*documentid*	特定の URL または文書 ID の文書を検索するには、docid キーワードを使用します。 **照会:** `docid:http://www.ibm.com/solutions/us/` or `docid:http://www.ibm.com/products/us/` **結果:** この照会は、URL が `docid:http://www.ibm.com/solutions/us/` または `docid:http://www.ibm.com/products/us/` のすべての文書を検索します。
taxonomy_ID::*category_ID*	エンタープライズ・サーチ・カテゴリーのいずれかによって生成されたカテゴリーが含まれているコレクションを検索するには、カテゴリー用語を使用して、特定のカテゴリー化分類法内の特定のカテゴリーに属している文書を検索します。 **照会:** `scopes::research "computer science"` **結果:** この照会は、`research` 有効範囲カテゴリーに属しており、`computer science` というフレーズ (句) が含まれているすべての文書を検索します。

文字	使用法
$source::*source_type*	特定のソース・タイプの文書を検索するには、$source キーワードを使用します。ソース照会は、複数のソースの文書が含まれているコレクションにおいて便利です。 **照会:** `$source::DB2 "computer science"` **結果:** この照会は、`computer science` というフレーズ (句) が含まれている DB2 文書を検索します。
$language::*language_id*	特定の言語で書かれている文書を検索するには、$language キーワードを使用します。 **照会:** `$language::en "computer science"` **結果:** この照会は、`computer science` というフレーズ (句) が含まれている英語の文書を検索します。
$doctype::*document_type*	特定の文書フォーマットの文書を検索するには、$doctype キーワードを使用します。 **照会:** `$doctype:application/pdf "computer science"` **結果:** この照会は、`computer science` というフレーズ (句) が含まれている Adobe Acrobat 文書を検索します。
#field::=value	指定した数値に等しい値が数値フィールドにある文書を検索するには、数値制約構文を使用します。 **照会:** `#price:=1700 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 である文書を検索します。
#field::>value	指定した数値より大きい値が数値フィールドにある文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>1700 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より大きい文書を検索します。
#field::<value	指定した数値より小さい値が数値フィールドにある文書を検索するには、数値制約構文を使用します。 **照会:** `#price:<1700 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より小さい文書を検索します。
#field::>=value	指定した数値より大きいか等しい値が数値フィールドにある文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>=1700 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より大きいか等しい文書を検索します。

文字	使用法
#field::<=value	指定した数値より小さいか等しい値が数値フィールドにある文書を検索するには、数値制約構文を使用します。 **照会:** `#price:<=1700 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より小さいか等しい文書を検索します。
#field::>value1<value2	指定した範囲内の文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>1700<3900 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より大きく 3900 より小さい文書を検索します。
#field::>=value1<=value2	指定した範囲内の文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>=1700<=3900 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 以上 3900 以下の文書を検索します。
#field::>value1<=value2	指定した範囲内の文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>1700<=3900 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 より大きく 3900 以下の文書を検索します。
#field::>=value1<value2	指定した範囲内の文書を検索するには、数値制約構文を使用します。 **照会:** `#price:>=1700<3900 laptop` **結果:** この照会は、価格フィールド `laptop` の値が 1700 以上で 3900 より小さい文書を検索します。
ACL constraints: (ACL tokens)	セキュリティー上の理由から、アクセス制御制約を他の制約のように照会ストリングに指定することはできません。照会のアクセス制御制約を指定するには、QUERY インターフェースの setACLConstraints(String aclConstraints) メソッドを使用します。 ACL 制約を指定する際に、括弧、正符号 (+)、負符号 (-) および曲折アクセント記号 (^) を、通常の照会構文と同じ意味で使用することができます。 **照会:** `thinkpad ACL constraints: (john_m | dev_group)` **結果:** この照会は、 `thinkpad` というワードと `john_m` または `dev_group` という ACL トークンが含まれている文書を検索します。

Java ソース・コードの作成

Java ソース・コードを作成する前に、Java ベースのビルド・ツールである Apache ANT をインストールして、構成する必要があります。Apache ANT のインストールおよび構成方法について詳しくは、 http://ant.apache.org/ を参照してください。

Java ソース・コードを作成するには、

1. コマンド行で、以下のいずれかのディレクトリーに移動します。
   - $install_root/samples/siapi (たとえば、/opt/IBM/es/samples/siapi)
   - $install_root/samples//datalistener (たとえば、/opt/IBM/es/samples//datalistener)

   いずれのディレクトリーにも、ANT がファイルのビルド時に使用する build.xml ファイルが含まれています。
2. ant と入力して Enter キーを押します。

Java ソース・コードのコンパイル後、ADD TEXT HERE メッセージが表示されます。

SearchExample クラス

SearchExample クラスには、エンタープライズ・サーチ検索サーバーに検索をサブミットするのに 最低限必要な簡単な例が用意されています。

例

```
public class SearchExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String queryString;
    private String collectionId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
     // obtain the OmniFind specific SIAPI Search factory implementation
        SearchFactory factory =
            SiapiSearchImpl.createSearchFactory("com.ibm.es.api.search.
     RemoteSearchFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
```

```
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Search Service implementation
        SearchService searchService =
            factory.getSearchService(config);

        // obtain a Searchable object to the specified
        // collection ID
        Searchable searchable = searchService.getSearchable(appinfo, collectionId);
        if(searchable == null) {
            System.out.println("Failed to get a searchable for collection:
    " + collectionId);
            return;
        }

        // create a new Query object using the specified
        // query string
        Query q = factory.createQuery(queryString);
                q.setQueryLanguage("en_US");

 // execute the search by calling the Searchable's
        // search method.  A SIAPI ResultSet object will
        // be returned
 ResultSet rset = searchable.search(q);
 if(rset != null) {
            // get the array of results from the ResultSet
            Result r[] = rset.getResults();
            if(r != null) {
                // walk the results list and print out the
                // document identifier
                for(int k = 0; k < r.length; k++) {
                    System.out.println("Result " + k + ": " + r[k].getDocumentID());
  }
            }
        }
    }

    public static void main(String[] args) throws Exception {
        SearchExample sc = new SearchExample();
        if (args.length < 5) {
            System.out.println("Usage: SearchExample <queryString> <collection id>
     <application name>" +
                               "<application password> <hostname> <port>");
            System.out.println("Example Usage: SearchExample lotus col1 app1
     password localhost 80");
            return;
        }
        sc.queryString = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

AdvancedSearchExample クラス

AdvancedSearchExample クラスは、拡張照会設定および結果処理オプションの使用例です。

例

```
public class AdvancedSearchExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String queryString;
    private String collectionId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
     // obtain the OmniFind specific SIAPI Search factory implementation
        SearchFactory factory =
            SiapiSearchImpl.createSearchFactory("com.ibm.es.api.search.
     RemoteSearchFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Search Service implementation
        SearchService searchService =
            factory.getSearchService(config);

        // obtain a Searchable object to the specified
        // collection ID
        Searchable searchable = searchService.getSearchable(appinfo, collectionId);
        if(searchable == null) {
            System.out.println("Failed to get a searchable for collection:
     " + collectionId);
            return;
        }

        // create a new Query object using the specified
        // query string
        Query q = factory.createQuery(queryString);

        // set the result range we want to access on this query
        q.setRequestedResultRange(0, 20);
                q.setQueryLanguage("en_US");
        // designate that we do not want the Description
        // field returned for each result
        q.setReturnedAttribute(Query.RETURN_RESULT_DESCRIPTION, false);

 // execute the search by calling the Searchable's
        // search method.  A SIAPI ResultSet object will
        // be returned
```

```
ResultSet rset = searchable.search(q);
if(rset != null) {
           System.out.println("");
           System.out.println("Estimated results:
    " + rset.getEstimatedNumberOfResults());
           System.out.println("Available results:
    " + rset.getAvailableNumberOfResults());
           System.out.println("Evaluation time: " + rset.getQueryEvaluationTime());
           System.out.println("");
           Result pr[] = rset.getPredefinedResults();
           if(pr != null) {
               // walk the predefinedResults list and print out the
               // document identifier and document title
               for(int k = 0; k < pr.length; k++) {
                   System.out.println("PredefinedResult " + k + ":
      " + pr[k].getDocumentID());
                   System.out.println("¥tTitle: " + pr[k].getTitle());
                   System.out.println("¥tDescription: " + pr[k].getDescription());
               }
           }
           System.out.println("");
           SpellCorrection sc[] = rset.getSpellCorrections();
           if(sc != null) {
               // walk the list of returned spelling corrections
               // and print out the query sub string and the
               // spelling suggestions
               for(int k = 0; k < sc.length; k++) {
                       System.out.println("SpellCorrection " + k + ": " + sc[k].
        getQuerySubstring());
                       String[] corrections = sc[k].getSuggestions();
                       if(corrections != null) {
                               for(int s = 0; s < corrections.length; s++) {
                                       System.out.println("Suggestion " + s + ":
               " + corrections[s]);
                               }
                       }
               }
           }
           System.out.println("");
           // get the array of results from the ResultSet
           Result r[] = rset.getResults();
           if(r != null) {
               // walk the results list
               for(int k = 0; k < r.length; k++) {
                   // print out all predefined field values.
                   // NOTE:  Description will be "null"
                   // since we modified the Query above
                   // to NOT return the Description field.
                   System.out.println("Result " + k + ": " + r[k].getDocumentID());
    System.out.println("¥tScore: " + r[k].getScore() + "%");
                   System.out.println("¥tTitle: " + r[k].getTitle());
    System.out.println("¥tLanguage: " + r[k].getLanguage());
    System.out.println("¥tType: " + r[k].getDocumentType());
                   System.out.println("¥tSource: " + r[k].getDocumentSource());
                   System.out.println("¥tDate: " + r[k].getDate());
    System.out.println("¥tDescription: " + r[k].getDescription());
                   // walk any categories that this result belongs to
                   ResultCategory[] cats = r[k].getCategories();
                   if(cats != null) {
                    for(int s = 0; s < cats.length; s++) {
                           CategoryInfo info = cats[s].getInfo();
                           // print out the taxonomy id, the category id, and the
        category label
                           // for each ResultCategory returned
                           System.out.println("¥tTaxonomy : " + cats[s].
         getTaxonomyID());
                           if(info != null) {
```

```
                                System.out.println("¥tCategory " + s + ":
            " + info.getID());
                                System.out.println("¥t¥tLabel " + s + ":
            " + info.getLabel());
                            }
                     }
                    }
                    // print out any additional field names and their
                    // values (document metadata)
                    NameValuePair[] fields = r[k].getFields();
                    if(fields != null) {
                        for(int s = 0; s < fields.length; s++) {
                            NameValuePair nvp = fields[s];
                            if(nvp != null) {
                                System.out.print("¥tField " + nvp.getName() + ":
            " + nvp.getValue());
                            }
                        }
                    }
      System.out.println("");
                }
            }
        }
    }

    public static void main(String[] args) throws Exception {
        AdvancedSearchExample sc = new AdvancedSearchExample();
        if (args.length < 5) {
            System.out.println("Usage: AdvancedSearchExample <queryString>
     <collection id> <application name>" +
                               "<application password><hostname><port>");
            System.out.println("Example Usage: AdvancedSearchExample lotus col1 app1
     password localhost 80");
            return;
        }
        sc.queryString = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

BrowseExample クラス

BrowseExample クラスは、コレクションの分類法ツリーにアクセスし、基本的なナビゲーション・プロパティーをいくつか印刷する例を示します。

例

```
public class BrowseExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String collectionId;
    private String taxonomyId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
        // obtain the OmniFind specific SIAPI Browse factory implementation
 BrowseFactory factory =
  SiapiBrowseImpl.createBrowseFactory("com.ibm.es.api.browse.RemoteBrowseFactory");
```

```
        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Browse service implementation
BrowseService browseService = factory.getBrowseService(config);

        // get a TaxonomyBrowser forthe specified taxonomy id and collection id
        TaxonomyBrowser browser = browseService.getTaxonomyBrowser(appinfo,
  collectionId,
  taxonomyId);
        if(browser == null) {
            System.out.println("Failed to get a taxonomy for taxonomy id:
    " + taxonomyId +
                               " from collection: " + collectionId);
            return;
        }

        //Display the Taxonomy Info
        TaxonomyInfo taxonomyInfo = browser.getTaxonomyInfo();
        System.out.println("Taxonomy label: " + taxonomyInfo.getLabel());

        // get the root category
        Category rootCategory = browser.getRootCategory();

        // display the root category
        System.out.println("Root category label: " + rootCategory.getInfo().
   getLabel());
        System.out.println("child categories:");
        CategoryInfo[] childrenInfo  = rootCategory.getChildren();
        for (int i = 0; i < childrenInfo.length; i++) {
            System.out.println("¥t" + childrenInfo[i].getLabel());
        }

        // Now get the root's first child category
        Category childCategory = browser.getCategory(rootCategory.getChildren()
  [0].getID());

        // Display the child category and it's path from root
        System.out.println("Root's first child's label: " + childCategory.getInfo()
  .getLabel());
        System.out.println("It's path from root is : ");
        CategoryInfo[] pathFromRoot = childCategory.getPathFromRoot();
        for (int i = 0; i < pathFromRoot.length; i++) {
            System.out.println("-->" + pathFromRoot[i].getLabel());
        }
   }
```

```
    public static void main(String[] args) throws Exception {
        BrowseExample sc = new BrowseExample();
        if (args.length < 5) {
            System.out.println("Usage: BrowseExample <taxonomy id> <collection id>
     <applicationname>");
System.out.println("Example Usage: BrowseExample tax1 col1 Default password
localhost 80");
            return;
        }
        sc.taxonomyId = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

SIAPI 呼び出しおよびメソッド

SIAPI は、以下のクラスおよびインターフェースを使用します。

SiapiVersion クラス

SiapiVersion クラスは、SIAPI のバージョンを管理します。

getSiapiVersion メソッド

SIAPI のバージョンを戻します。

構文

```
public String getSiapiVersion()
```

SiapiSearchImpl クラス

SiapiSearchImpl クラスは、エンタープライズ・サーチ SIAPI インプリメンテーションの検索ファクトリーを獲得します。

createSearchFactory メソッド

インプリメンテーション・クラス名によって SIAPI インプリメンテーションの検索ファクトリーを作成します。

このメソッドは com.ibm.es.api.search.RemoteSearchFactory を指定して呼び出してください。

構文

```
public static SearchFactory createSearchFactory(String factoryImplClassName)
  throws SiapiException
```

パラメーター

factoryImplClassName
インプリメンテーション・ファクトリー・クラスの絶対パス名。

戻り

SIAPI インプリメンテーションの検索ファクトリー。

SearchFactory インターフェース

SearchFactory インターフェースは、SIAPI 検索インプリメンテーションのエントリー・ポイントです。

getSearchService メソッド

特定のアプリケーションおよび特定の検索サーバーの検索サービス・オブジェクトを取得します。

構文

```
public SearchService getSearchService(Properties config) throws SiapiException
```

パラメーター

config
エンタープライズ・サーチは、以下の 3 つの構成プロパティーをサポートします。

- hostname (必須) - エンタープライズ・サーチ検索サーバーのホスト名。
- port (オプション) - エンタープライズ・サーチ検索サーバーのポート番号 (デフォルト・ポート番号は 80)。
- locale (オプション) - 翻訳されたエラー・メッセージを戻す際に使用されるロケール設定 (デフォルト値は、デフォルト Java ロケール)。

戻り

検索サービス。

スロー

検索サービスを取得できない場合は SiapiException。

createQuery メソッド

指定された照会テキストで、照会オブジェクトを作成します。

照会テキストは、エンタープライズ・サーチでサポートされる照会構文に従っている必要があります。照会の作成について詳しくは、9 ページの『照会構文』を参照してください。

構文

```
public Query createQuery(String queryText) throws SiapiException
```

パラメーター

queryText
照会のテキスト。

戻り

新規照会オブジェクト。

createApplicationInfo メソッド

指定されたアプリケーション ID のアプリケーション情報オブジェクトを作成します。 ユーザーは、ApplicationInfo オブジェクトの取得後、ApplicationInfo.setPassword(String) または ApplicationInfo.setToken(String) を使用して、 アプリケーションのパスワードやセキュリティー・トークンを設定できます。

構文

```
public ApplicationInfo createApplicationInfo(java.lang.String applicationID)
                                      throws SiapiException
```

スロー

applicationID が無効な場合は SiapiException。

getVersion メソッド

特定の SIAPI インプリメンテーションのバージョンを戻します。バージョン・データは、 それぞれの SIAPI インプリメンテーションごとに別々に管理できます。ここに戻される値は、 SIAPI のバージョンと異なる場合があります。

構文

```
public java.lang.String getVersion()
```

CategoryInfo インターフェース

CategoryInfo インターフェースは、カテゴリーの記述情報を提供します。

getID メソッド

カテゴリー ID を戻します。

構文

```
public java.lang.String getID()
```

getLabel メソッド

カテゴリー・ラベルを戻します。

構文

```
public java.lang.String getLabel()
```

CollectionInfo インターフェース

CollectionInfo インターフェースは、コレクションに関する記述情報を提供します。

getCollectionInfo メソッド

検索可能なコレクションの ID およびラベルを戻します。

構文

```
public CollectionInfo getCollectionInfo()
```

戻り

コレクション ID およびラベルが含まれている CollectionInfo インターフェース。

getID メソッド

コレクション ID を戻します。

構文

```
public java.lang.String getID()
```

getLabel メソッド

コレクション・ラベルを戻します。

構文

```
public java.lang.String getLabel()
```

FieldInfo インターフェース

FieldInfo インターフェースは、フィールド定義を表します。

フィールドとは、文書のコンテンツまたはメタデータの名前が付いたテキスト部分です。

getID メソッド

フィールド名を戻します。

構文

```
public java.lang.String getID()
```

getAvailableFields メソッド

このコレクション内の文書に定義されているメタデータ・フィールドに関する情報を戻します。

構文

```
public FieldInfo[] getAvailableFields()
```

戻り

フィールド情報オブジェクトの配列。

isContentSearchable メソッド

このタイプのフィールドが、コンテンツ検索が可能な場合は true を戻します。

コンテンツ検索が可能なフィールドとは、フィールド検索以外の照会を使用する検索可能なテキストが含まれているフィールドです。たとえば、apple というワードの照会では、 apple というワードが含まれている、コンテンツ検索が可能なフィールドがある文書が戻されます。

構文

```
public boolean isContentSearchable()
```

戻り

このタイプのフィールドが、コンテンツ検索が可能な場合は true を、それ以外の場合は false を戻します。

isFieldSearchable メソッド

このタイプのフィールドがフィールド検索可能である場合は true を戻します。

構文

フィールド検索が可能なフィールドとは、フィールド検索照会で検索可能なテキストが含まれているフィールドのことです。たとえば、`keywords:apple` の照会では、`apple` というワードが含まれているフィールド検索が可能なフィールド・キーワードがある文書が戻されます。

```
public boolean isFieldSearchable()
```

戻り

このタイプのフィールドが、フィールド検索が可能な場合は true を、 それ以外の場合は false を戻します。

isReturnable メソッド

このタイプのフィールドの内容が検索結果と共に戻される場合は true を戻します。

たとえば、文書に、戻すことができる `author` メタデータ・フィールドがあって、その文書が照会結果として戻される場合は、author フィールドの内容が結果と共に戻されます。

QUERY インターフェースの setReturnedFields メソッドを使用して、照会結果と共に戻されるフィールドの名前を指定します。結果オブジェクトから戻されたフィールドの内容を取得するには、getFields メソッドを使用します。

構文

```
public boolean isReturnable()
```

戻り

このタイプのフィールドの内容が検索結果と共に戻される場合は true、 それ以外の場合は false を戻します。

isParametric メソッド

このタイプのフィールドがパラメトリックである場合は true を戻します。

パラメトリック・フィールドとは、そのフィールドの値を数値に変換できるフィールドのことです。照会結果を、指定したパラメトリック・フィールド値の範囲内に制限する照会制約を作成する方法については、 9 ページの『照会構文』を参照してください。

構文

```
public boolean isParametric()
```

戻り

このタイプのフィールドがパラメトリックである場合は true を、 それ以外の場合は false を戻します。

Searchable インターフェース

ユーザーは、Searchable インターフェースにより、コレクションの索引を検索して、 コレクションに関する情報を入手できます。

ATTRIBUTE_LANGUAGE 定数

この定数を getAvailableAttributeValues メソッドに渡すと、照会を制約する際にこの属性タイプとして使用できるすべての文書言語の値のリストが得られます。

構文

```
public static final int ATTRIBUTE_LANGUAGE
```

ATTRIBUTE_SOURCE 定数

この定数を getAvailableAttributeValues メソッドに渡すと、照会を制約する際にこの属性タイプとして使用できるすべての文書タイプの値のリストが得られます。

構文

```
public static final int ATTRIBUTE_SOURCE
```

ATTRIBUTE_DOCTYPE 定数

この定数を getAvailableAttributeValues メソッドに渡すと、照会を制約する際にこの属性タイプとして使用できるすべての文書タイプの値のリストが得られます。

構文

```
public static final int ATTRIBUTE_DOCTYPE
```

search メソッド

照会を実行して、結果セットを戻します。

構文

```
public ResultSet search(Query query) throws SiapiException
```

パラメーター

query
検索したい照会。

戻り

検索結果が含まれる ResultSet オブジェクト。

スロー

検索オペレーションを実行できなかった場合は SiapiException。

count メソッド

指定された照会に一致するすべての文書の正確なカウントを戻します。 要求されたすべての照会用語が含まれており、禁止用語が含まれていない文書がカウントされます。

構文

```
public int count(Query query) throws SiapiException
```

パラメーター

query
検索する照会。

戻り

指定された照会に一致する文書の数。

スロー

カウント・オペレーションを実行できなかった場合は SiapiException。

getSpellCorrections メソッド

指定された照会ストリングのスペル修正提案のリストを戻します。 有効な提案が無い場合は、ヌルを戻します。

スペル修正が使用可能な場合は、QUERY インターフェースの 40 ページの『setSpellCorrectionEnabled メソッド』を参照してください。

構文

```
public SpellCorrection[] getSpellCorrections()
```

パラメーター

queryText
検査する照会ストリング。

戻り

指定された照会ストリングのスペル修正提案のリストを戻します。

getDefaultLanguage メソッド

照会の処理で、この Searchable インターフェースが使用するデフォルト言語を検索します。

構文

```
public String getDefaultLanguage()
```

getAvailableAttributeValues メソッド

指定したタイプの属性照会制約の作成に使用する値のリストを戻します。

構文

```
public java.lang.String[] getAvailableAttributeValues(int attributeType)
```

パラメーター

attributeType
属性タイプ。

戻り

指定した属性タイプに有効な値のリストを戻します。

注: 一部の属性タイプでは、戻されるリストが不完全であったり、存在しなかったりする場合があります。このような場合には、NULL 値が戻されます。

有効な属性タイプは、次のとおりです。

- ATTRIBUTE_DOCTYPE
- ATTRIBUTE_LANGUAGE
- ATTRIBUTE_SOURCE

たとえば、getAvailableValues(Searchable.ATTRIBUTE_SOURCE) への呼び出しは、このコレクションの文書ソースのリストを戻します。詳しくは、 9 ページの『照会構文』を参照してください。

getAvailableFields メソッド

このコレクション内の文書に定義されているメタデータ・フィールドに関する情報を戻します。

構文

```
public FieldInfo[] getAvailableFields()
```

戻り

フィールド情報オブジェクトの配列。

setProperty メソッド

検索可能なプロパティーの値を設定します。

これらのプロパティーを使用して、検索可能なプロパティーの動作を変更することができます。サポートされるプロパティーを調べるには、 28 ページの『getProperties メソッド』を呼び出します。

制約事項: エンタープライズ・サーチは、プロパティーをサポートしないため、このメソッドが呼び出されると例外をスローします。

構文

```
public void setProperty(String name,
                        String value) throws SiapiException
```

パラメーター

name
検索可能なプロパティーの名前。

value
検索可能なプロパティーの値。

スロー

この時点でプロパティーを変更できない場合、または値が正しくない場合に SiapiException をスローします。

getProperty メソッド

検索可能なプロパティーの値を戻します。

構文

```
public String getProperty(String name)
```

パラメーター

name
検索する検索可能なプロパティーの名前。

getProperties メソッド

検索可能なすべてのプロパティーを戻します。無い場合は、ヌルを戻します。

構文

```
public Properties getProperties()
```

getCollectionInfo メソッド

検索可能なコレクションの ID およびラベルを戻します。

構文

```
public CollectionInfo getCollectionInfo()
```

戻り

コレクション ID およびラベルが含まれている CollectionInfo インターフェース。

getAvailableSearchables メソッド

コレクションに対して使用可能なすべての参照を取得します。

構文

```
public Searchable[] getAvailableSearchables(ApplicationInfo appInfo)
 throws SiapiException
```

パラメーター

appInfo
要求するアプリケーションの認証および許可情報。

戻り

コレクションの Searchable インターフェース。

スロー

アクションを実行できない場合は SiapiException。

getSearchable メソッド

コレクション ID ごとにコレクションへの参照を取得します。

構文

```
public Searchable getSearchable(ApplicationInfo appInfo,
                                String collectionID) throws SiapiException
```

パラメーター

appInfo
要求するアプリケーションの認証および許可情報。

collectionID
コレクションの ID。

戻り

検索可能な参照。

スロー

検索可能な参照を取得できない場合は SiapiException。

SearchService インターフェース

このインターフェースを使用して、検索可能なオブジェクトを取得します。

getAvailableSearchables メソッド

コレクションに対して使用可能なすべての参照を取得します。

構文

```
public Searchable[] getAvailableSearchables(ApplicationInfo appInfo)
 throws SiapiException
```

パラメーター

appInfo
要求するアプリケーションの認証および許可情報。

戻り

コレクションの Searchable インターフェース。

スロー

アクションを実行できない場合は SiapiException。

getSearchable メソッド

コレクション ID ごとにコレクションへの参照を取得します。

構文

```
public Searchable getSearchable(ApplicationInfo appInfo,
                                String collectionID) throws SiapiException
```

パラメーター

appInfo
要求するアプリケーションの認証および許可情報。

collectionID
コレクションの ID。

戻り

検索可能な参照。

スロー

検索可能な参照を取得できない場合は SiapiException。

ApplicationInfo インターフェース

ApplicationInfo インターフェースは、コレクションへのアクセスを許可する際に使用する情報を制御します。

getID メソッド

アプリケーション ID を戻します。

構文

```
public java.lang.String getId()
```

getPassword メソッド

アプリケーションに関連したパスワードを戻します。

構文

```
public String getPassword()
```

getToken メソッド

アプリケーションに関連したトークン値を戻します。

構文

```
public String getToken()
```

setPassword メソッド

アプリケーションに関連したパスワードを設定します。このメソッドは、 パスワードによる認証が必要な安全保護環境で使用します。

構文

```
public void setPassword(String password) throws SiapiException
```

パラメーター

password
設定するパスワード。

スロー

パスワードが無効な場合は SiapiException。

setToken メソッド

アプリケーションに関連したトークンを設定します。

このメソッドは、シングル・サインオン環境などの、トークンを渡す必要がある安全保護環境で使用します。

構文

```
public void setToken(String token) throws SiapiException
```

パラメーター

token
設定するトークン。

スロー

トークンが無効な場合は SiapiException。

createApplicationInfo メソッド

指定されたアプリケーション ID のアプリケーション情報オブジェクトを作成します。 ユーザーは、ApplicationInfo オブジェクトの取得後、ApplicationInfo.setPassword(String) または ApplicationInfo.setToken(String) を使用して、 アプリケーションのパスワードやセキュリティー・トークンを設定できます。

構文

```
public ApplicationInfo createApplicationInfo(java.lang.String applicationID)
                                      throws SiapiException
```

スロー

applicationID が無効な場合は SiapiException。

createApplicationInfo メソッド

指定されたアプリケーション ID のアプリケーション情報オブジェクトを作成します。

ユーザーは、ApplicationInfo オブジェクトの取得後、 setPassword または setToken を使用して、アプリケーションのパスワードやセキュリティー・トークンを設定できます。

構文

```
public ApplicationInfo createApplicationInfo(String applicationID)
  throws SiapiException
```

パラメーター

ID アプリケーションの ID。

QUERY インターフェース

QUERY インターフェースは、照会動作を制御します。

createQuery メソッド

指定された照会テキストで、照会オブジェクトを作成します。

照会テキストは、エンタープライズ・サーチでサポートされる照会構文に従っている必要があります。照会の作成について詳しくは、9 ページの『照会構文』を参照してください。

構文

```
public Query createQuery(String queryText) throws SiapiException
```

パラメーター

queryText
照会のテキスト。

戻り

新規照会オブジェクト。

SORT_KEY_RELEVANCE 定数

結果を関連性によってソートするには、この定数を setSortKey メソッドに渡します。

構文

```
public static final java.lang.String SORT_KEY_RELEVANCE
```

SORT_KEY_DATE 定数

結果を日付によってソートするには、この定数を setSortKey メソッドに渡します。

構文

```
public static final java.lang.String SORT_KEY_DATE
```

SORT_KEY_NONE 定数

結果をソートしない場合は、この定数を setSortKey メソッドに渡します。

構文

```
public static final java.lang.String SORT_KEY_NONE
```

SORT_ORDER_ASCENDING 定数

結果を昇順にソートするには、この定数を setSortOrder メソッドに渡します。

構文

```
public static final int SORT_ORDER_ASCENDING
```

SORT_ORDER_DESCENDING 定数

結果を降順にソートするには、この定数を setSortOrder メソッドに渡します。

構文

```
public static final int SORT_ORDER_DESCENDING
```

RESULT_CATEGORIES_ALL 定数

完全な結果カテゴリー詳細レベルに設定するには、この定数を setResultCategoriesDetailLevel メソッドに渡します。

構文

```
public static final int RESULT_CATEGORIES_ALL
```

RESULT_CATEGORIES_NO_PATH_TO_ROOT 定数

部分的な結果カテゴリー詳細レベルに設定するには、この定数を setResultCategoriesDetailLevel メソッドに渡します。

構文

```
public static final int RESULT_CATEGORIES_NO_PATH_TO_ROOT
```

RETURN_RESULT_DESCRIPTION 定数

結果記述の戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_DESCRIPTION
```

RETURN_RESULT_TITLE 定数

結果タイトルの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_TITLE
```

RETURN_RESULT_FIELDS 定数

結果保管フィールドの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_FIELDS
```

RETURN_RESULT_CATEGORIES 定数

結果カテゴリーの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_CATEGORIES
```

RETURN_RESULT_TYPE 定数

結果文書タイプの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_TYPE
```

RETURN_RESULT_SOURCE 定数

結果文書ソースの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_SOURCE
```

RETURN_RESULT_LANGUAGE 定数

結果言語の戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_LANGUAGE
```

RETURN_RESULT_DATE 定数

結果日付の戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_DATE
```

RETURN_RESULT_SCORE 定数

結果スコアの戻りを使用可能/使用不可にするには、この定数を setReturnedAttribute メソッドに渡します。

構文

```
public static final int RETURN_RESULT_SCORE
```

getText メソッド

この照会のテキストを戻します。

詳しくは、 9 ページの『照会構文』を参照してください。

構文

```
public String getText()
```

setText メソッド

この照会のテキストを設定します。

詳しくは、 9 ページの『照会構文』を参照してください。

構文

```
public void setText(String text) throws SiapiException
```

パラメーター

text
この照会の新規テキスト。

getQueryLanguage メソッド

setQueryLanguage メソッドを呼び出して以前に指定された、この照会の言語を戻します。 言語が指定されなかった場合には、ヌルを戻します。

構文

```
public String getQueryLanguage()
```

getReturnedFields メソッド

この照会の結果に戻されるフィールドの名前 (setReturnedFields メソッドを呼び出して 事前に指定されている) を戻します。 フィールドが指定されていない場合は、ヌルを戻します。

各文書で、一部のフィールド・エレメントが returnable とマークされている場合があります。これは、そのフィールド・エレメントが、その文書の索引内に保管されていることを意味します。このような文書が検索結果にある場合、ユーザーは、これらのフィールド・エレメントをリストアできます。

戻されるフィールドが指定されていない場合、このメソッドはヌルを戻し、照会結果には何もフィールドが戻されません。

『resetReturnedFields メソッド』を呼び出すことにより、検索するフィールドを正確に指定できます。

構文

```
public String[] getReturnedFields()
```

setReturnedFields メソッド

戻されるメタデータ・フィールドの名前を設定します。デフォルトでは、 すべてのフィールドが戻されます。

このメソッドを呼び出す前に、setReturnedAttribute メソッドを使用して、メタデータ・フィールドの検索を (RETURN_RESULT_FIELDS 定数を使用して) 使用可能に設定してください。

構文

```
public void setReturnedFields(String[] fieldNames) throws SiapiException
```

パラメーター

fieldNames
必要フィールドの名前

resetReturnedFields メソッド

この照会で戻されるメタデータ・フィールド・データをリセットします。

構文

```
public void resetReturnedFields()
```

getNumRequestedResults メソッド

この照会で要求される結果数を戻します。

構文

```
public int getNumRequestedResults()
```

getFirstRequestedResult メソッド

この照会で要求されている最初の結果のランク (順位) を戻します。

たとえば、最初の結果が 20 で要求された結果数が 10 の場合は、 20 から 30 にランクされた 10 個の結果が検索されます。ランクは 0 から始まります。

構文

```
public int getFirstRequestedResult()
```

setRequestedResultRange メソッド

この照会で要求される最初の結果のランク (順位) と、要求される結果数を設定します。 最初の結果は、0 または要求される結果数の倍数のいずれかでなければなりません。

ランクは 0 から始まります。

構文

```
public void setRequestedResultRange(int fromResult,
                                    int numberOfResult) throws SiapiException
```

パラメーター

fromResult
最初に要求される結果のランク。

numberOfResults
要求される結果数 (このパラメーターで許可されている最大値は 100 です)。

getProperty メソッド

検索可能なプロパティーの値を戻します。

構文

```
public String getProperty(String name)
```

パラメーター

name
検索する検索可能なプロパティーの名前。

setProperty メソッド

検索可能なプロパティーの値を設定します。

これらのプロパティーを使用して、検索可能なプロパティーの動作を変更することができます。サポートされるプロパティーを調べるには、 28 ページの『getProperties メソッド』を呼び出します。

制約事項: エンタープライズ・サーチは、プロパティーをサポートしないため、このメソッドが呼び出されると例外をスローします。

構文

```
public void setProperty(String name,
                        String value) throws SiapiException
```

パラメーター

name
検索可能なプロパティーの名前。

value
検索可能なプロパティーの値。

スロー

この時点でプロパティーを変更できない場合、または値が正しくない場合に SiapiException をスローします。

getProperties メソッド

検索可能なすべてのプロパティーを戻します。無い場合は、ヌルを戻します。

構文

```
public Properties getProperties()
```

getSortKey メソッド

この照会の結果をソートするキーを戻します。

特別に事前定義されているキーは、次のとおりです。

SORT_KEY_RELEVANCE
結果を関連性によってソートする (デフォルト)。

SORT_KEY_DATE
結果を日付によってソートする。

SORT_KEY_NONE
ソートを行わない。

その他の有効なキー値として、検索エンジンがソートをサポートするフィールドの名前があります。

構文

```
public String getSortKey()
```

getSortPoolSize メソッド

検索エンジンがソート・キーによってソートする、上位関連結果の数を戻します。ソート・プール・サイズのデフォルト設定は 500 です。

構文

```
public int getSortPoolSize()
```

戻り

ソート・プール・サイズ。

setSortPoolSize メソッド

検索エンジンがソート・キーによってソートする、上位関連結果の数を設定します。

このデフォルト設定は、インプリメンテーションに依存しているため、SIAPI インプリメンテーションによって異なる可能性があります。

構文

```
public void setSortPoolSize(int sortPoolSize) throws SiapiException
```

パラメーター

sortPoolSize
ソート・プール・サイズ。

スロー

ソート・プール・サイズが負の数、または大き過ぎる場合は SiapiException。

getSortOrder メソッド

照会結果がソートされる順序を戻します。

有効な値は、次のとおりです。

SORT_ORDER_ASCENDING
ソート順序は昇順。

SORT_ORDER_DESCENDING
ソート順序は降順。

構文

```
public int getSortOrder()
```

戻り

ソート順序。

setSortKey メソッド

この照会の結果をソートするキーを設定します。

特別に事前定義されているキーは、次のとおりです。

SORT_KEY_RELEVANCE
結果を関連性によってソートする (デフォルト)。

SORT_KEY_DOCUMENT_ID
結果を文書 ID によってソートする。

SORT_KEY_DATE
結果を日付によってソートする。

SORT_KEY_NONE
ソートを行わない。

その他の有効な値として、検索エンジンがソートをサポートするフィールドの名前があります。

構文

```
public void setSortKey(String sortKey) throws SiapiException
```

パラメーター

sortKey
ソート基準となるフィールド名またはソート方法

isSiteCollapsingEnabled メソッド

この照会の結果のコンピューティング時に、サイト縮小を適用する必要がある場合は true を戻します。

上位の結果に、同じ Web サイトから 3 個以上の結果が含まれている場合に、結果を縮小します。

構文

```
public boolean isSiteCollapsingEnabled()
```

setSiteCollapsingEnabled メソッド

この照会のサイト縮小動作を設定します。

構文

```
public void setSiteCollapsingEnabled(boolean value) throws SiapiException
```

パラメーター

value
この照会の結果のコンピューティング時に、サイト縮小を適用する必要がある場合は true、 それ以外の場合は false です。

isPredefinedResultsEnabled メソッド

この照会で定義済みリンクが戻される場合は true を戻します。

構文

```
public boolean isPredefinedResultsEnabled()
```

setPredefinedResultsEnabled メソッド

この照会の事前定義リンク動作を設定します。

構文

```
public void setPredefinedResultsEnabled(boolean value) throws SiapiException
```

パラメーター

value
この照会で事前定義リンクを戻す必要がある場合は true、それ以外の場合は false。

setSpellCorrectionEnabled メソッド

この照会で、スペル修正を使用可能にするかどうかを設定します。

構文

```
public void setSpellCorrectionEnabled(boolean enable) throws SiapiException
```

パラメーター

enable
スペル修正を使用可能にする場合は true、それ以外の場合は false。

スロー

スペル修正を使用可能に設定できなかった場合。

isSpellCorrectionEnabled メソッド

この照会でスペル修正が使用可能な場合は true を戻します。それ以外の場合は、false を戻します。

構文

```
public boolean isSpellCorrectionEnabled()
```

setResultCategoriesDetailLevel メソッド

照会結果の必要カテゴリー詳細レベルを設定します。

構文

RETURN_RESULT_CATEGORIES 定数を指定して setReturnedAttribute メソッドを使用し、結果カテゴリーの検索を使用可能/使用不可にします。

有効な detailLevel 値は、次のとおりです。

RESULT_CATEGORIES_ALL
各結果カテゴリーは、完全なルートからのパス情報と共に戻されます (デフォルト値は RESULT_CATEGORIES_ALL)。

RESULT_CATEGORIES_NO_PATH_TO_ROOT
各結果カテゴリーは、ルートからのパス情報なしで戻されます (つまり、getPathFromRoot はヌルを返します)。

```
public void setResultCategoriesDetailLevel(int detailLevel)
```

setReturnedAttribute メソッド

デフォルトでは、RETURN_RESULT_FIELDS 以外のすべての結果属性が戻されます。

有効な attributeType 値は、次のとおりです。

RETURN_RESULT_TITLE
isReturned が false に設定されている場合、getTitle はヌルを戻します。

RETURN_RESULT_DESCRIPTION
isReturned が false に設定されている場合、getDescription はヌルを戻します。

RETURN_RESULT_FIELDS
isReturned が false に設定されている場合、getFields はヌルを戻します。

RETURN_RESULT_CATEGORIES
isReturned が false に設定されている場合、getCategories はヌルを戻します。

RETURN_RESULT_TYPE
isReturned が false に設定されている場合、getDocumentType はヌルを戻します。

RETURN_RESULT_SOURCE
isReturned が false に設定されている場合、getDocumentSource はヌルを戻します。

RETURN_RESULT_LANGUAGE
isReturned が false に設定されている場合、getLanguage はヌルを戻します。

RETURN_RESULT_DATE
isReturned が false に設定されている場合、getDate はヌルを戻します。

RETURN_RESULT_SCORE
isReturned が false に設定されている場合、getScore は 0.0 を戻します。

構文

```
public void setReturnedAttribute(int attributeType,
                                 boolean isReturned)
```

パラメーター

attributeType
結果属性タイプ。

isReturned
指定されたタイプの属性を戻す必要がある場合は true、それ以外の場合は false。

isAttributeReturned メソッド

照会結果オブジェクトに、指定された属性タイプが含まれている場合は true を戻します。 デフォルトでは、すべての結果属性が戻されます。

有効な attributeType 値は、次のとおりです。

RETURN_RESULT_TITLE
isReturned が false に設定されている場合、getTitle はヌルを戻します (デフォルトでは、フィールドを除くすべての結果属性が戻されます)。

RETURN_RESULT_DESCRIPTION
isReturned が false に設定されている場合、getDescription はヌルを戻します。

RETURN_RESULT_FIELDS
isReturned が false に設定されている場合、getFields はヌルを戻します。

RETURN_RESULT_CATEGORIES
isReturned が false に設定されている場合、getCategories はヌルを戻します。

RETURN_RESULT_TYPE
isReturned が false に設定されている場合、getDocumentType はヌルを戻します。

RETURN_RESULT_SOURCE
isReturned が false に設定されている場合、getDocumentSource はヌルを戻します。

RETURN_RESULT_LANGUAGE
isReturned が false に設定されている場合、getLanguage はヌルを戻します。

RETURN_RESULT_DATE
isReturned が false に設定されている場合、getDate はヌルを戻します。

RETURN_RESULT_SCORE
isReturned が false に設定されている場合、getScore は 0.0 を戻します。

構文

```
public boolean isAttributeReturned(int attributeType)
```

パラメーター

attributeType
結果属性タイプ

戻り

指定された属性が戻された場合は true、それ以外の場合は false。

setQueryID メソッド

この照会にアプリケーションで定義済みの ID を割り当てます。この ID は、 検索エンジンがこの照会の実行に関連した情報をログに記録する際に使用します。

構文

```
public void setQueryID(java.lang.String queryID)
```

パラメーター

queryID
アプリケーションで定義済みの照会 ID。

getQueryID メソッド

この照会に設定されている、アプリケーションで定義済みの照会 ID を戻します。 ID が設定されていない場合は、ヌルを戻します。

構文

```
public java.lang.String getQueryID()
```

戻り

アプリケーションで定義済みの照会 ID を戻します。

setACLConstraints メソッド

この照会のアクセス制御に関する制約を設定します。

ACL 制約構文について詳しくは、 9 ページの『照会構文』を参照してください。

構文

```
public void setACLConstraints(java.lang.String aclConstraints)
```

パラメーター

aclConstraints
アクセス制御に関する制約を指定するストリング。

getACLConstraints メソッド

この照会に設定されているアクセス制御制約ストリングを戻します。 アクセス制御制約が設定されていない場合は、ヌルを戻します。

ACL 制約構文について詳しくは、 9 ページの『照会構文』を参照してください。

戻り

この照会に設定されているアクセス制御制約ストリングを戻します。

setQueryLanguage メソッド

検索サーバーにおけるコレクション・デフォルト言語以外の言語を指定します。

構文

```
public void setQueryLanguage(String lang) throws SiapiException
```

パラメーター

lang
この照会の言語。2 文字のコード (たとえば、en や jp) または 4 文字のコード (たとえば、 en-US や fr-FR) を使用して、照会言語を指定することができます。

getResultCategoriesDetailLevel メソッド

照会結果のカテゴリー詳細レベルを戻します。

照会結果のカテゴリー詳細レベル。有効な戻り値は、次のとおりです。

RESULT_CATEGORIES_ALL
戻される各結果カテゴリーに、完全なルートからのパス情報が含まれます。

RESULT_CATEGORIES_NO_PATH_TO_ROOT
戻される各結果カテゴリーに、ルートからのパス情報は含まれません。

構文

```
public int getResultCategoriesDetailLevel()
```

戻り

setText メソッド

この照会のテキストを設定します。

詳しくは、 9 ページの『照会構文』を参照してください。

構文

```
public void setText(String text) throws SiapiException
```

パラメーター

text
この照会の新規テキスト。

NameValuePair インターフェース

NameValuePair インターフェースには、文書の名前および関連した値が含まれます。

各 NameValuePair は、文書フィールドまたはメタタグを表します。 Result インプリメンテーションは、NameValuePair オブジェクトの配列を保管し、特定文書のフィールドの完全なリストを表します。

getName メソッド

この値に関連した名前を戻します。

構文

```
public String getName()
```

戻り

この値に関連した名前。

getValue メソッド

値を戻します。

構文

```
public String getValue()
```

戻り

値を戻します。

getFields メソッド

この結果に戻されたすべてのフィールドの名前と値を戻します。 フィールドが存在しない場合、またはユーザーがフィールドを戻さないように要求している場合は、ヌルを戻します。

構文

```
public NameValuePair[] getFields()
```

Result インターフェース

Result インターフェースには、単一の検索結果が含まれます。

各 result は、この結果に対応する文書の固有の ID に関する情報を保持します。 result には、ユーザーが検索で要求した属性に応じて、文書のタイトル、記述 (ある場合)、スコア、文書タイプ、ソース、言語が含まれます。

また、各 result には、NameValuePairs として表される動的フィールド・プロパティーのセットや、文書が現在割り当てられているカテゴリーのリストを含めることもできます。

getCategories メソッド

この結果の文書が属しているカテゴリーを戻します。 この結果にカテゴリーが無い場合、またはユーザーがカテゴリーを戻さないように要求した場合、 このメソッドは、ヌルを戻します。

構文

```
public ResultCategory[] getCategories()
```

getDate メソッド

日付が存在しない場合、またはユーザーが照会オブジェクトの setReturnedAttribute メソッドを使用して 日付を戻さないように指定した場合は、ヌルを戻します。

構文

```
public Date getDate()
```

getDescription メソッド

この結果の文書の記述 (サマリーと呼ばれます) を戻します。 フィールドが存在しない場合、またはユーザーが照会オブジェクトの setReturnedAttribute メソッドを使用して、 フィールドを戻さないように指定した場合は、ヌルを戻します。

構文

```
public String getDescription()
```

getDocumentID メソッド

この結果の文書の固有の名前を戻します。

構文

```
public String getDocumentID()
```

getDocumentSource メソッド

この結果の文書のソースを戻します。フィールドが存在しない場合、またはユーザーがフィールドを戻さないように要求している場合は、ヌルまたは unknown を戻します。

構文

```
public String getDocumentSource()
```

getDocumentType メソッド

この結果の文書のタイプを戻します。フィールドが存在しない場合、またはユーザーがフィールドを戻さないように要求している場合は、ヌルまたは unknown を戻します。

構文

```
public String getDocumentType()
```

getFields メソッド

この結果に戻されたすべてのフィールドの名前と値を戻します。 フィールドが存在しない場合、またはユーザーがフィールドを戻さないように要求している場合は、ヌルを戻します。

構文

```
public NameValuePair[] getFields()
```

getFields メソッド

この結果に戻されたフィールド (fieldName) の値を戻します。 有効なフィールドが無い場合は、ヌルを戻します。

構文

```
public String[] getFields(String fieldName)
```

getLanguage メソッド

この結果の文書の言語を戻します。不明な場合、またはユーザーが言語を戻さないように要求している場合は、ヌルを戻します。

構文

```
public String getLanguage()
```

getProperties メソッド

検索可能なすべてのプロパティーを戻します。無い場合は、ヌルを戻します。

構文

```
public Properties getProperties()
```

getProperty メソッド

検索可能なプロパティーの値を戻します。

構文

```
public String getProperty(String name)
```

パラメーター

name
検索する検索可能なプロパティーの名前。

getScore メソッド

索引によってこの結果に割り当てられるスコアを戻します。ユーザーがスコアを戻さないように要求している場合は、ヌルを戻します。

構文

```
public double getScore()
```

getTitle メソッド

この結果の文書のタイトルを戻します。有効なタイトルが無い場合、 またはユーザーがタイトルを戻さないように要求している場合は、ヌルを戻します。

構文

```
public String getTitle()
```

isFirstOfASite メソッド

この結果が、同じサイトの一連の結果において、最初の結果である場合は true を戻します。

構文

```
public boolean isFirstOfASite()
```

ResultCategory インターフェース

結果文書のカテゴリー。

構文

```
public interface ResultSet extends Serializable
```

getConfidence メソッド

文書がこのカテゴリーに属していることの信頼度を戻します。 信頼度は 0 から 100 までです。

構文

```
public double getConfidence()
```

getInfo メソッド

このカテゴリーの ID およびラベルを戻します。

構文

```
public CategoryInfo getInfo()
```

getPathFromRoot メソッド

ルート・カテゴリーからこのカテゴリーの直接の親までの、すべての上位カテゴリーの ID とラベルを戻します。

構文

```
public CategoryInfo[] getPathFromRoot()
```

戻り

上位カテゴリー情報。

getTaxonomyID メソッド

このカテゴリーの分類法 ID を戻します。

構文

```
public String getTaxonomyID()
```

戻り

分類法 ID。

MAX_CONFIDENCE 定数

最高信頼度 (MAX_CONFIDENCE の値は 100.0 です)。

構文

```
public static final double MAX_CONFIDENCE
```

MIN_CONFIDENCE 定数

最低信頼度 (MIN_CONFIDENCE の値は 0.0 です)。

構文

```
public static final double MIN_CONFIDENCE
```

ResultSet インターフェース

ResultSet インターフェースには、実行した照会の検索結果が含まれます。

ResultSet には、照会結果のほかに、事前定義リンクを割り当てた管理者、この要求に対する結果の見積もり総数、使用可能な結果数、合計実行時間、およびスペル修正提案が含まれます。 Searchable インターフェースを使用して公開されている search メソッドは、 照会完了時に、呼び出し元に ResultSet インスタンスを戻します。

getAvailableNumberOfResults メソッド

この照会で検索できる結果数を戻します。

この数値は、見積もり結果数と異なる場合があります。たとえば、見積もり結果数が最大の 500 ソートを超える場合、getAvailableNumberOfResults は 500 を戻します。照会結果がソートされていない場合 (ソート・キーが SORT_KEY_NONE に設定されている場合)、このメソッドは、これが結果の最終ページである場合には 0 を、さらに結果が続く場合には 1 を戻します。

構文

```
public int getAvailableNumberOfResults()
```

getEstimatedNumberOfResults メソッド

この照会で一致する結果の総数の見積もりを戻します。 これは、getAvailableNumberOfResults によって戻される値と異なる可能性があります。 この数値が既知のものでない場合は -1 を戻します。

この見積もりは、要求されたすべての照会用語が含まれており、禁止用語が含まれていない文書の数です。エンタープライズ・サーチは、ソートされていない結果に対する結果数の見積もりは提供しません。そのため、ソート・キーが SORT_KEY_NONE の場合には 0 が戻されます。

構文

```
public int getEstimatedNumberOfResults()
```

getPredefinedResults メソッド

定義済み照会結果の配列を戻します。有効な定義済み結果が無い場合、またはユーザーが定義済みリンクを戻さないように要求している場合は、ヌルを戻します。

このメソッドにより、検索エンジンは、照会に対して特別な結果セットを強調して戻すことができます。

構文

```
public Result[] getPredefinedResults()
```

getProperties メソッド

検索可能なすべてのプロパティーを戻します。無い場合は、ヌルを戻します。

構文

```
public Properties getProperties()
```

getProperty メソッド

検索可能なプロパティーの値を戻します。

構文

```
public String getProperty(String name)
```

パラメーター

name
検索する検索可能なプロパティーの名前。

getQueryEvaluationTime メソッド

照会を処理して、この結果セットを取得するのに要する時間 (ミリ秒) を戻します。

構文

```
public long getQueryEvaluationTime()
```

getResults メソッド

有効な結果の配列を戻します。

構文

```
public Result[] getResults()
```

getSpellCorrections メソッド

指定された照会ストリングのスペル修正提案のリストを戻します。 有効な提案が無い場合は、ヌルを戻します。

スペル修正が使用可能な場合は、QUERY インターフェースの 40 ページの『setSpellCorrectionEnabled メソッド』を参照してください。

構文

```
public SpellCorrection[] getSpellCorrections()
```

パラメーター

queryText
検査する照会ストリング。

戻り

指定された照会ストリングのスペル修正提案のリストを戻します。

hasUnconstrainedResults メソッド

この照会で結果セットに結果が戻されなかったのは、検索中に適用された制約によるものかどうかの指標を戻します。 照会から制約が取り外されると、制約されていない結果が戻されます。

以下の制約があります。

- ACL
- カテゴリー
- 範囲制約
- 結果言語
- 文書タイプの制約
- 文書ソースの制約

構文

```
public int hasUnconstrainedResults()
```

戻り

制約されていない結果が存在する場合は正の値、制約されていない結果が存在しない場合は 0 (ゼロ)、制約されていない結果が存在するかどうかが不明な場合は負の値。

isEvaluationTruncated メソッド

この検索結果セットを取得する検索の評価プロセスで、検索プロセス中に少なくとも 1 回は切り捨てる必要があった場合は true を戻します。

構文

```
public boolean isEvaluationTruncated()
```

SpellCorrection インターフェース

SpellCorrection インターフェースには、修正される元の照会サブストリングと、 そのサブストリングの修正に関する提案 (可能性の高い順) が含まれます。

getQuerySubstring メソッド

スペルの修正を提案されている照会のサブストリングを戻します。

構文

```
public String getQuerySubstring()
```

getSuggestions メソッド

スペル修正提案を可能性の高い順に戻します。提案が無い場合は、ヌルを戻します。

構文

```
public String[] getSuggestions()
```

BrowseFactory インターフェース

BrowseFactory インターフェースは、SIAPI 分類法ブラウズ機能のエンタープライズ・サーチ・インプリメンテーションのエントリー・ポイントです。

getBrowseService メソッド

ブラウズ・サービス・オブジェクトを取得します。

構文

```
public BrowseService getBrowseService(java.util.Properties config)
                                throws SiapiException
```

パラメーター

config
検索サーバーの構成情報。

エンタープライズ・サーチは、3 つの構成プロパティーをサポートします (SearchService の取得について詳しくは、5 ページの『SIAPI アプリケーションの構造』を参照してください)。

- **hostname** (必須) - エンタープライズ・サーチ検索サーバーのホスト名。
- **port** (オプション) - エンタープライズ・サーチ検索サーバーのポート番号 (デフォルト・ポート番号は 80)。
- **locale** (オプション) - 翻訳されたエラー・メッセージを戻す際に使用されるロケール設定 (デフォルト値は、デフォルト Java ロケール)。

createApplicationInfo メソッド

指定されたアプリケーション ID のアプリケーション情報オブジェクトを作成します。

ユーザーは、ApplicationInfo オブジェクトの取得後、 setPassword または setToken を使用して、アプリケーションのパスワードやセキュリティー・トークンを設定できます。

構文

```
public ApplicationInfo createApplicationInfo(String applicationID)
  throws SiapiException
```

パラメーター

ID アプリケーションの ID。

getVersion メソッド

特定の SIAPI インプリメンテーションのバージョンを戻します。

バージョン・データは、それぞれの SIAPI インプリメンテーションごとに別々に管理できます。ここに戻される値は、SIAPI API のバージョンと異なる場合があります。

構文

```
public String getVersion()
```

BrowseService インターフェース

BrowseService インターフェースは、コレクションの分類法ブラウズ・サービスを提供します。

getTaxonomyBrowser メソッド

指定したコレクションの分類法をブラウズするインターフェースを戻します。

構文

```
public TaxonomyBrowser getTaxonomyBrowser(ApplicationInfo appInfo,
                                          java.lang.String collectionID,
                                          java.lang.String taxonomyID)
                                   throws SiapiException
```

パラメーター

appInfo
要求しているアプリケーションに関する認証および許可情報。

collectionID
コレクションの ID。

taxonomyID
要求される分類法の ID。

戻り

指定したコレクションの分類法ブラウザー。

getAvailableTaxonomyBrowsers メソッド

指定されたコレクションに有効なすべての TaxonomyBrowsers の配列を戻します。

構文

```
public TaxonomyBrowser[] getAvailableTaxonomyBrowsers(ApplicationInfo appInfo,
                                                      java.lang.String collectionID)
                                               throws SiapiException
```

パラメーター

appInfo
要求しているアプリケーションに関する認証および許可情報。

collectionID
コレクションの ID。

戻り

有効な TaxonomyBrowsers の配列。

Category インターフェース

Category インターフェースは、カテゴリーの分類法データをカプセル化します。

getInfo メソッド

このカテゴリーの ID およびラベルを戻します。

構文

```
public CategoryInfo getInfo()
```

getChildren メソッド

すべての子カテゴリーの ID およびラベルを戻します。

構文

```
public CategoryInfo[] getChildren()
```

戻り

子カテゴリーの ID。

getPathFromRoot メソッド

ルート・カテゴリーからこのカテゴリーの直接の親までの、すべての上位カテゴリーの ID とラベルを戻します。

構文

```
public CategoryInfo[] getPathFromRoot()
```

戻り

上位カテゴリー情報。

getRootCategory メソッド

ツリーのルート・カテゴリーを戻します。

構文

```
public Category getRootCategory()
```

戻り

ルート・カテゴリー。

getCategory メソッド

指定されたカテゴリー ID のカテゴリーを戻します。

構文

```
public Category getCategory(java.lang.String categoryID)
```

パラメーター

categoryID
戻されるカテゴリーの ID。

戻り

指定されたカテゴリー。

TaxonomyBrowser インターフェース

TaxonomyBrowser インターフェースは、分類法ブラウズおよびナビゲーション・インターフェースを提供します。 このインターフェースは、読み取り専用で、非管理分類法ブラウズ・タスクです。

getRootCategory メソッド

ツリーのルート・カテゴリーを戻します。

構文

```
public Category getRootCategory()
```

戻り

ルート・カテゴリー。

getCategory メソッド

指定されたカテゴリー ID のカテゴリーを戻します。

構文

```
public Category getCategory(java.lang.String categoryID)
```

パラメーター

categoryID
戻されるカテゴリーの ID。

戻り

指定されたカテゴリー。

getTaxonomyInfo メソッド

分類法情報 (ID およびラベル) を戻します。

構文

```
public TaxonomyInfo getTaxonomyInfo()
```

SiapiException クラス

汎用 SIAPI 例外。

この例外は、検索および索引付け API メソッドの結果として発生することがあります。これは、唯一の SIAPI 例外で、適切な重大度と適切なタイプを設定することにより、多数のエラーをカバーします。

SiapiException は、それぞれ、重大度、エラー・コード、およびオプションとして組み込まれた例外と関連付けられます。各エラー・コードは、パラメーター化されたメッセージに関連付けられており、そのメッセージを解決する引き数を SIAPI インプリメンテーションにより追加することもできます。

この例外は、例外の元の原因 (つまり、最初の引き金となった例外) が SiapiException 内に組み込み例外として保持されるようにインプリメントされています。そのため、スタック・トレースを印刷すると、元の例外の呼び出しスタックが印刷されます。このインプリメンテーションは、問題の元の原因を識別する際に役立ちます。

SEVERITY_ERROR 定数

非致命的エラー重大度。

構文

```
public static final int SEVERITY_ERROR
```

SEVERITY_FATAL_ERROR 定数

致命的エラー重大度。

構文

```
public static final int SEVERITY_FATAL_ERROR
```

TYPE_UNKNOWN_ERROR 定数

予期しない内部エラー。

メッセージ: 内部エラー: %1
メッセージ・パラメーター:
%1: 汎用メッセージ

構文

```
public static final int TYPE_UNKNOWN_ERROR
```

TYPE_DOC_EXIST_ERROR 定数

すでに存在している文書を作成しようとしました。

メッセージ: 「文書 %1 はすでに存在しています」
メッセージ・パラメーター:
%1: 文書 ID

構文

```
public static final int TYPE_DOC_EXIST_ERROR
```

TYPE_DOC_NOT_FOUND_ERROR 定数

存在しない文書にアクセスしようとしたか、または除去しようとしました。

メッセージ: 「文書 %1 がありません」
メッセージ・パラメーター:
%1: 文書 ID

構文

```
public static final int TYPE_DOC_NOT_FOUND_ERROR
```

TYPE_SEARCH_ENGINE_STATE_ERROR 定数

検索エンジンとの対話で正しくないことが行われました。検索エンジンの状態が、要求されたオペレーションに一致しませんでした。

メッセージ: 「検索エンジン状態エラー: %1」
メッセージ・パラメーター:
%1: 汎用メッセージ

構文

```
public static final int TYPE_SEARCH_ENGINE_STATE_ERROR
```

TYPE_ILLEGAL_VALUE_ERROR 定数

正しくない値を設定しようとしました (たとえば、正しくない値をプロパティーに設定するなど)。

メッセージ: 「正しくない値のエラー: %1」
メッセージ・パラメーター:
%1: 正しくない名前および値

構文

```
public static final int TYPE_ILLEGAL_VALUE_ERROR
```

TYPE_IO_ERROR 定数

検索エンジンとの対話が、入出力エラーにより失敗しました。 たとえば、ディスクがフルでファイルを作成できなかったなどです。

メッセージ: 「入出力エラー」
メッセージ・パラメーター:
なし

構文

```
public static final int TYPE_IO_ERROR
```

TYPE_IMPL_FACTORY_ERROR 定数

SIAPI インプリメンテーションのファクトリー・オブジェクトを作成しようとして、失敗しました。

考えられる原因:

- ファクトリー・インプリメンテーション・クラスを検索およびロードできなかった。

- ファクトリー・インプリメンテーション・オブジェクトをインスタンス化できなかった。

メッセージ: 「ファクトリー・インプリメンテーション・エラー」
メッセージ・パラメーター:
なし

構文

```
public static final int TYPE_IMPL_FACTORY_ERROR
```

TYPE_UNSUPPORTED_OPERATION 定数

SIAPI インプリメンテーションに一部の機能が欠落していることを示します。

メッセージ: 「サポートされないオペレーション: %1」
メッセージ・パラメーター:
%1: サポートされないオペレーションの名前

構文

```
public static final int TYPE_UNSUPPORTED_OPERATION
```

TYPE_INDEX_DOES_NOT_EXIST 定数

指定された名前の索引がありません。

メッセージ: 「コレクション %1 がありません」
メッセージ・パラメーター:
%1: コレクション ID

構文

```
public static final int TYPE_INDEX_DOES_NOT_EXIST
```

TYPE_TOO_MANY_VALUES 定数

指定された多値変数の値数が、許可された値数を超えています。

メッセージ: 「引き数 %1 の値が多過ぎます」
メッセージ・パラメーター:
%1: 引き数名

構文

```
public static final int TYPE_TOO_MANY_VALUES
```

TYPE_TOO_FEW_VALUES 定数

指定された多値変数の値数が、必要な値数に不足しています。

メッセージ: 「引き数 %1 の値が少な過ぎます」
メッセージ・パラメーター:
%1: 引き数名

構文

```
public static final int TYPE_TOO_FEW_VALUES
```

TYPE_ILLEGAL_RESULTS_RANGE 定数

要求された結果範囲が、検索エンジンによって許可されている値を超えています。

メッセージ: 「指定された結果範囲が正しくありません: %1」
メッセージ・パラメーター:
%1: 指定された結果範囲

構文

```
public static final int TYPE_ILLEGAL_RESULTS_RANGE
```

TYPE_INDEX_CORRUPTED 定数

索引が破壊されています。

メッセージ: 「コレクション %1 に破壊された索引があります」
メッセージ・パラメーター:
%1: コレクション ID

構文

```
public static final int TYPE_INDEX_CORRUPTED
```

TYPE_UNKNOWN_ENCODING 定数

文字エンコード方式が不明であるため、一部のデータを使用できません。

メッセージ: 「エンコード方式 %1 が不明であるため、データを使用できません」
メッセージ・パラメーター:
%1: エンコード方式名

構文

```
public static final int TYPE_UNKNOWN_ENCODING
```

TYPE_QUERY_SYNTAX_ERROR 定数

照会構文エラーが発生しました。

メッセージ: 「照会構文エラー: %1」
メッセージ・パラメーター:
%1: 正しくない照会ストリング

構文

```
public static final int TYPE_QUERY_SYNTAX_ERROR
```

getSeverity メソッド

この例外の重大度を戻します (SEVERITY_ERROR または SEVERITY_FATAL_ERROR)。

構文

```
public int getSeverity()
```

戻り

この例外の重大度を戻します。

getSeverityDescription メソッド

この例外の重大度を、人が読み取り可能な名前で戻します。

構文

```
public java.lang.String getSeverityDescription()
```

戻り

この例外の重大度を、人が読み取り可能な名前で戻します。

getType メソッド

この例外のタイプを戻します。

SEVERITY_ERROR または SEVERITY_FATAL_ERROR のいずれか

構文

```
public int getType()
```

戻り

この例外のタイプを戻します。

getTypeDescription メソッド

この例外のタイプを、人が読み取り可能な名前で戻します。

構文

```
public java.lang.String getTypeDescription()
```

戻り

この例外のタイプを、人が読み取り可能な名前で戻します。

printStackTrace メソッド

組み込み例外のスタック・トレースを印刷します。スタック・トレースが無い場合は、 この例外のスタックを印刷します。

構文

```
public void printStackTrace()
```

オーバーライド

java.lang.throwable クラスの printStackTrace をオーバーライドします。

addArgument メソッド

追加する「名前 - 値」ペアをこの例外の引き数として追加します。

構文

```
public void addArgument(NameValuePair arg)
```

パラメーター

arg
追加する「名前 - 値」ペア。

getArguments メソッド

この例外の引き数を戻します。

構文

```
public java.util.ArrayList getArguments()
```

戻り

この例外の引き数と引き数の配列リストを戻します。

getMessage メソッド

この例外のエラー・メッセージ・ストリングを戻します。

構文

```
public java.lang.String getMessage()
```

戻り

この例外のエラー・メッセージ・ストリングを戻します。

オーバーライド

java.lang.throwable クラスの getMessage をオーバーライドします。

getLocalizedMessage メソッド

この例外のローカライズ・メッセージを戻します。

構文

```
public java.lang.String getLocalizedMessage()
```

戻り

この例外のローカライズ・メッセージを戻します。

オーバーライド

java.lang.throwable クラスの getLocalizedMessage をオーバーライドします。

TaxonomyInfo インターフェース

TaxonomyInfo インターフェースは、分類法の ID およびラベルを提供します。

getID メソッド

分類法 ID を戻します。

構文

```
public java.lang.String getID()
```

getLabel メソッド

分類法ラベルを戻します。

構文

```
public java.lang.String getLabel()
```

getTaxonomyInfo メソッド

分類法情報 (ID およびラベル) を戻します。

構文

```
public TaxonomyInfo getTaxonomyInfo()
```

サンプル SIAPI アプリケーション

エンタープライズ・サーチ SIAPI には、さまざまなサンプル・アプリケーションが組み込まれています。

サンプル・アプリケーションでは、次のことが示されています。

- 検索サーバーに検索照会をサブミットするのに最低限必要なアプリケーション
- 基本的な検索タスク
- プログラミングに必要な標準的な検索タスク
- 基本的な分類法ブラウズおよびナビゲーション・タスク

最低限必要なアプリケーション

SearchExample クラスには、検索サーバーに検索照会をサブミットするのに最低限必要な、アプリケーションの簡単な例が用意されています。

この例では、以下の方法が示されています。

- サービスへのアクセス
- コレクションの指定
- アプリケーションの指定
- 照会の実行
- 戻される結果の処理

SearchExample クラス

SearchExample クラスは、 すでに esapi.jar ファイル内にコンパイルされています。コマンド行から以下のコマンドを実行することにより、クラスを実行することができます。

UNIX

```
java -classpath <installroot>/lib/esapi.jar:<installroot>>/lib/siapi.jar
SearchExample
```

Windows

```
javac -classpath <installroot>¥lib¥esapi.jar;<installroot>>¥lib¥siapi.jar
SearchExample
```

ホストが指定されていない場合、デフォルトのホストは、ローカル・ホストであると想定されます。ポートが指定されていない場合、デフォルトのポートは 80 であると想定されます。

ファクトリーの取得

SIAPI に対して作成されるプログラムでは、最初の行で、 Factory クラスのインスタンスを作成する必要があります。エンタープライズ・サーチの SearchFactory クラス・インプリメンテーションは、 com.ibm.es.api.search.RemoteSearchFactory です。

エンタープライズ・サーチ固有の SIAPI インプリメンテーションのファクトリーを取得します。

```
SearchFactory factory =
          SiapiSearchImpl.createSearchFactory
  ("com.ibm.es.api.search.RemoteSearchFactory");
```

有効なアプリケーション ID の作成

検索サーバーは、アプリケーション ID を使用して、コレクションへのこのアクセスを許可します。

```
ApplicationInfo appinfo = factory.createApplicationInfo("app1");
appinfo.setPassword("password");
```

新規プロパティー・オブジェクトの作成

```
Properties config = new Properties();
```

ホスト名の設定

SIAPI 検索サーバーのホスト名を設定します。この場合は、エンタープライズ・サーチ検索サーバーです。ホスト名は、エンタープライズ・サーチ WebSphere インストールに割り当てられているホスト名でなければなりません。

```
config.setProperty("hostname", hostname);
```

ポート番号の設定

SIAPI 検索サーバーのポート番号を設定します。デフォルト値は、ポート 80 (Web サーバーのポート) です。

```
config.setProperty("port", portNumber);
```

ロケールの設定

クライアントのロケールを設定します。ベースとなる SIAPI インプリメンテーションは、この値を使用して、適切な言語に翻訳されたエラー・メッセージをクライアントに戻します。

```
config.setProperty("locale", "en_US");
```

SearchService インプリメンテーションの取得

```
SearchService searchService =
    factory.getSearchService(config);
```

指定されたコレクション ID に対する Searchable オブジェクトの取得

```
        Searchable searchable =
            searchService.getSearchable(appinfo, collectionId);
        if (searchable == null) {
            System.out.println("Failed to get a searchable
           for collection: " + collectionId);
            return;
        }
```

指定された照会ストリングを使用した新規照会オブジェクトの作成

```
        Query q = factory.createQuery(queryString);
```

結果範囲の設定

この照会でアクセスする結果範囲を設定します。

```
        q.setRequestedResultRange(0, 10);
```

検索の実行

Searchable インターフェースの検索メソッドを呼び出して、検索を実行します。SIAPI ResultSet オブジェクトが戻されます。

```
        ResultSet rset = searchable.search(q);
        System.out.println("returned from search call");
        if (rset != null) {
```

ResultSet からの結果の取得

```
Result r[] = rset.getResults();
            if (r != null) {
```

結果の印刷

結果リストを走査して、文書 ID と文書タイトルを印刷します。

```
for (int k = 0; k < r.length; k++) {
                  System.out.println
      ("Result " + k + ": " + r[k].getDocumentID() + " - " + r[k].getTitle());
                }
            }
        } else {
            System.out.println("result set was null");
        }
    }
```

テスト目的のみ

```
public static void main(String[] args) throws Exception {
   SearchExample sc = new SearchExample();
   if (args.length < 2) {
      System.out.println
("Usage: SearchExample<queryString><collection ID>optionally:<hostname><port>");
System.out.println("Example Usage: SearchExample lotus col1");
System.out.println("¥tdefault host and port is localhost:80");
System.out.println("Example Usage: SearchExample lotus col1 localhost 80");
      return;
        }
        sc.queryString = args[0];
        sc.collectionId = args[1];
        if (args.length > 2) {
            sc.hostname = args[2];
        } else {
            sc.hostname = "localhost";
```

```
        }
        if (args.length > 3) {
            sc.portNumber = args[3];
        } else {
            sc.portNumber = "80";
        }
        sc.execute();
    }
}
```

ブラウズおよびナビゲーション・アプリケーション

このサンプル・コードは、基本的な分類法ブラウズおよびナビゲーション・タスクの SIAPI サンプルです。

この例では、以下の方法が示されています。

- ブラウズ・ファクトリーの取得
- ブラウズ・サービスの取得
- ブラウザー参照の取得
- ルート・カテゴリーの検出および表示
- ルートの最初の子カテゴリーの検出
- 子カテゴリーおよびそのルートからのパスの表示

ブラウズおよびナビゲーションの例

```
 */
import java.util.Locale;
import java.util.Properties;

import com.ibm.siapi.browse.BrowseFactory;
import com.ibm.siapi.browse.BrowseService;
import com.ibm.siapi.browse.Category;
import com.ibm.siapi.browse.SiapiBrowseImpl;
import com.ibm.siapi.browse.TaxonomyBrowser;
import com.ibm.siapi.common.ApplicationInfo;
import com.ibm.siapi.common.CategoryInfo;
import com.ibm.siapi.common.TaxonomyInfo;

/*
 * The BrowseExample class gives an example of accessing a collection's
 * taxonomy tree and printing out some of the basic navigation properties.
 *
 */
public class BrowseExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String collectionId;
    private String taxonomyId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
        // obtain the OmniFind specific SIAPI Browse factory implementation
 BrowseFactory factory =
  SiapiBrowseImpl.createBrowseFactory("com.ibm.es.api.browse.RemoteBrowseFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);
```

```
        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Browse service implementation
BrowseService browseService = factory.getBrowseService(config);

        // get a TaxonomyBrowser forthe specified taxonomy id and collection id
        TaxonomyBrowser browser = browseService.getTaxonomyBrowser(appinfo,
  collectionId, taxonomyId);
        if(browser == null) {
            System.out.println("Failed to get a taxonomy for taxonomy id: "
    + taxonomyId +
                               " from collection: " + collectionId);
            return;
        }

        //Display the Taxonomy Info
        TaxonomyInfo taxonomyInfo = browser.getTaxonomyInfo();
        System.out.println("Taxonomy label: " + taxonomyInfo.getLabel());

        // get the root category
        Category rootCategory = browser.getRootCategory();

        // display the root category
        System.out.println("Root category label: " + rootCategory.getInfo()
  .getLabel());
        System.out.println("child categories:");
        CategoryInfo[] childrenInfo  = rootCategory.getChildren();
        for (int i = 0; i < childrenInfo.length; i++) {
            System.out.println("¥t" + childrenInfo[i].getLabel());
        }

        // Now get the root's first child category
        Category childCategory = browser.getCategory(rootCategory.getChildren()[0]
  .getID());

        // Display the child category and it's path from root
        System.out.println("Root's first child's label: " + childCategory.getInfo()
  .getLabel());
        System.out.println("It's path from root is : ");
        CategoryInfo[] pathFromRoot = childCategory.getPathFromRoot();
        for (int i = 0; i < pathFromRoot.length; i++) {
            System.out.println("-->" + pathFromRoot[i].getLabel());
        }
    }

    public static void main(String[] args) throws Exception {
        BrowseExample sc = new BrowseExample();
        if (args.length < 5) {
            System.out.println("Usage: BrowseExample <taxonomy id> <collection id>
     <application name>");
```

```
System.out.println("Example Usage: BrowseExample tax1 col1 Default password
localhost 80")
return;
}
sc.taxonomyId = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

すべての検索結果の取得

以下の例は、ソートされていない結果を戻し、照会結果をループする照会の設定方法を示しています。 エンタープライズ・サーチによって、ユーザーは照会で、最大 500 個のソートされた結果を得ることができますが、ソートされていない結果を得ることもできます。

SearchFactory および Searchable オブジェクトの取得

61 ページの『最低限必要なアプリケーション』の例で示されているように、SearchFactory および Searchable オブジェクトを取得します。

```
SearchFactory factory;
Searchable searchable;

... // obtain a SearchFactory and Searchable object
```

新規照会オブジェクトの作成

```
Query q = factory.createQuery("big apple");
```

ソートされていない結果を戻す照会の設定

```
q.setSortKey(Query.SORT_KEY_NONE);
```

検索の実行

照会をループして実行し、一度に 1 ページ分の結果を取得します。 エンタープライズ・サーチで許可される最大結果ページ・サイズは 100 です。

注: 結果ページを受け取る際に、ソートされた照会結果の場合とは異なる方法で、getAvailableNumberOfResults と getEstimatedNumberOfResults をインタープリットする必要があります。

- エンタープライズ・サーチは、ソートされていない結果に対する結果数の見積もりを提供しないため、 getEstimatedNumberOfResults は常に 0 を戻します。
- getAvailableNumberOfResults は、これが最終結果ページである場合には 0、さらに結果が続く場合は 1 を戻します。
- getResults によって戻される配列の長さで、この結果ページ内に実際にいくつの結果があるかがわかります。

```
int fromResult = 0;
int pageSize = 100;
boolean moreResults = true;

// loop over query results, pageSize results at a time
while (moreResults) {

 // set the result range for the next page of results
 q.setRequestedResultRange(fromResult, pageSize);

 // execute the search
 ResultSet resultPage = s.search(q);

 // loop over the results from the ResultSet
 Result[] results = resultPage.getResults();
 for (int i=0;i<results.length;i++) {
... // process result
 }

 // check if there are more available results
 moreResults = (resultPage.getAvailableNumberOfResults() == 1);

 // modify the range for getting the next page of results
 fromResult += pageSize;
}
```

第 4 章 データ・リスナー API

データ・リスナーは、既知のポートからデータを受信し、このデータをコレクションに送信する、エンタープライズ・サーチ・コンポーネントです。ユーザーは、データ・リスナー API を呼び出すことにより、 コレクションにページを追加したり、コレクションから URI を除去したり、URL にアクセスまたは再アクセスするようにコレクションの Web Crawler に指示したりすることができます。

エンタープライズ・サーチの概要

エンタープライズ・サーチ・システムは、さまざまなデータ・ソースからデータを収集し、収集したデータに索引を作成し、検索機能を提供することができます。データ・ソースには、 Web、ニュース・サーバー、データベース表、Lotus Notes® データベース、コンテンツ・マネージメント・システム、ファイル・システムなどがあります。

図 1 に、エンタープライズ・サーチ・システムの検索動作の概要を示します。ユーザーは検索サーバーに照会をサブミットし、索引サーバーは定期的に検索サーバー上のデータをリフレッシュします。

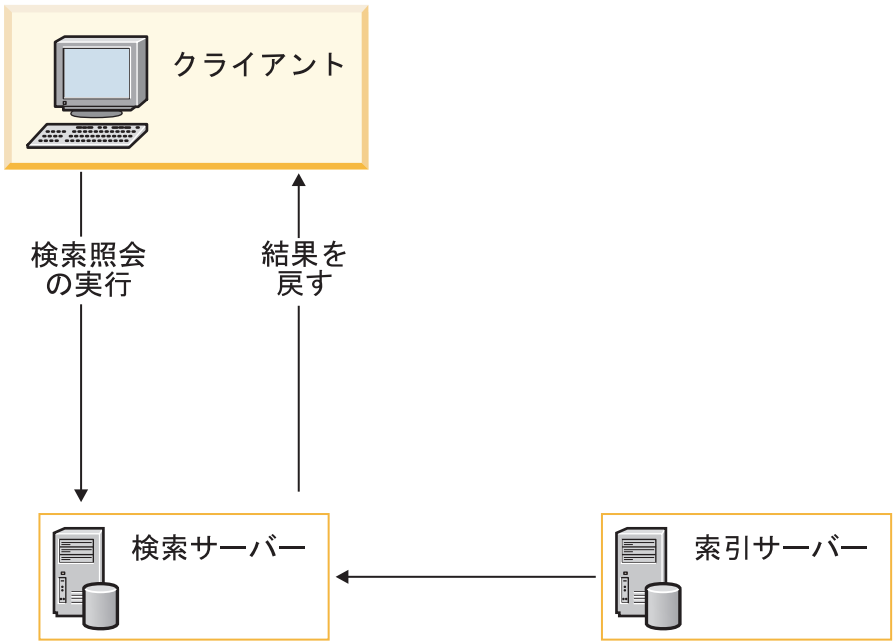

図 1. エンタープライズ・サーチの動作の概要

データ・リスナー API の概要

データ・リスナーは、エンタープライズ・サーチ・システム内の以下のコンポーネントに関与します。

クライアント・マシン
検索照会をサブミットする。

© Copyright IBM Corp. 2004

検索サーバー
照会を受け入れ、結果をクライアントに提供する。

索引サーバー
クローラーによって解析されたデータを保管する。

クローラー
データ・ソースから文書を検索する。

ユーザーは、データ・リスナー API によって、データ要求をエンタープライズ・サーチ・システムにサブミットします。

クライアントは、データ・リスナー・コンポーネントに接続して、適切なサーバーにデータをプッシュします。クライアントがデータ・リスナー・コンポーネントに接続すると、データ・リスナー・コンポーネントは、クライアントID とパスワードを検査して、クライアントが指定されたコレクションにデータをプッシュすることを許可されているかどうか検証します。

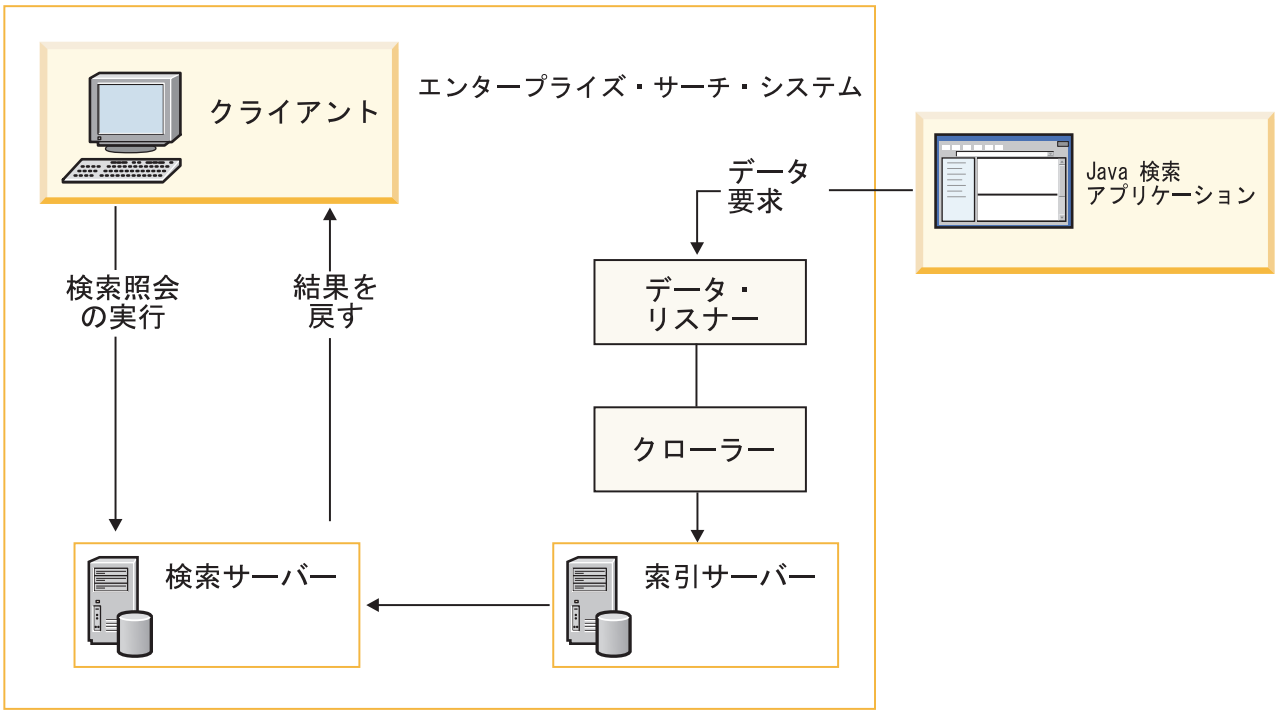

図 2. データ・リスナー API はエンタープライズ・サーチ・システムとどのように連動するか

データ・リスナー API プロパティー

データ・リスナー・コンポーネントにデータをプッシュする場合には、以下の情報を指定する必要があります。

表 2. データ・リスナー API プロパティー

プロパティー	定義
Hostname	データ・リスナー・サーバーのホスト名。

表 2. データ・リスナー *API* プロパティー *(続き)*

プロパティー	定義
Port	サーバーが listen するポート番号。
ID	クライアント ID。
PW	クライアント・パスワード。
Col	ターゲット・コレクションの ID。
URI	データの URI。
Metadata	タイプ DataSourceMetada のデータのメタデータ・オブジェクト。
Content	データのメイン・ボディ。

データ制御

データ・リスナー API は、URI および URL 要求をコレクション・サーバーおよびクローラーにサブミットすることにより、コンテンツを追加および除去します。

汎用リソース ID (URI)
HTML 文書、イメージ、ビデオ・クリップ、プログラムなどの Web 上のリソースを表すエンコード・アドレス。 URI スーパークラスには URL が組み込まれています。

汎用リソース・ロケーター (URL)
Web 上の特定の項目の名前および位置を指定する方法を提供する Web アドレス。

データ・リスナー API によるデータの除去

データ・リスナー API を使用して、検索コレクションからデータを除去するには、次の 2 つの方法があります。

- 検索サーバーおよび索引から URI を除去する
- 索引から URI パターンを除去する

検索サーバーおよび索引から URI を除去する

データ・リスナー API を使用して、 特定の URI を除去することができます。以下の図に示されているように、要求がデータ・リスナーに送信され、URI が索引サーバーおよび検索サーバーから除去されます。

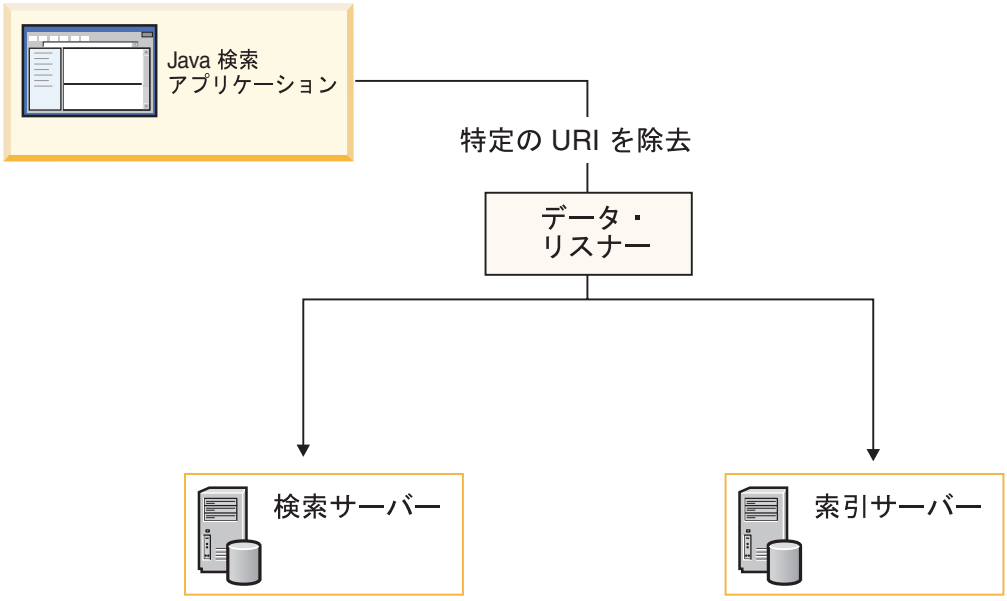

図 3. 特定の URI の除去

removeURI メソッドを使用して、指定されたコレクションからデータを削除します。

索引から URI パターンを除去する

データ・リスナー API を使用して、 URI パターンを除去することができます。たとえば、URI として www.ibm.com/*.html を指定して URI パターンの除去要求をサブミットすると、索引サーバーは以下の URL を除去します。

- www.ibm.com/home.html
- www.ibm.com/family.html
- www.ibm.com/pics.html

注: URI パターンの除去要求は、注意して使用してください。除去されたコンテンツはリカバリーできません。コンテンツを索引に再度、追加する必要があります。

上記の図に示されているように、除去要求がデータ・リスナーに送信され、URI パターンが索引サーバーから除去されます。

removeURI メソッドを使用して、指定されたコレクションからデータを削除します。除去された URI は、索引サーバーが検索サーバー上のデータをリフレッシュするまで、ユーザーへの検索結果に戻されます。

データ・リスナー API によるデータの追加

データ・リスナー API を使用して、検索コレクションにデータをプッシュするには、次の 2 つの方法があります。

- コンテンツと共に URI を追加する
- URL にアクセスまたは再アクセスする

コンテンツと共に URI を追加する

データ・リスナー API を使用して、 pushData 要求に content パラメーターを組み込むことによって、URL をそのコンテンツと共に追加することができます。この方法では、図 4 に示されているように、指定されたデータをデータ・リスナーおよび索引サーバーに送信します。

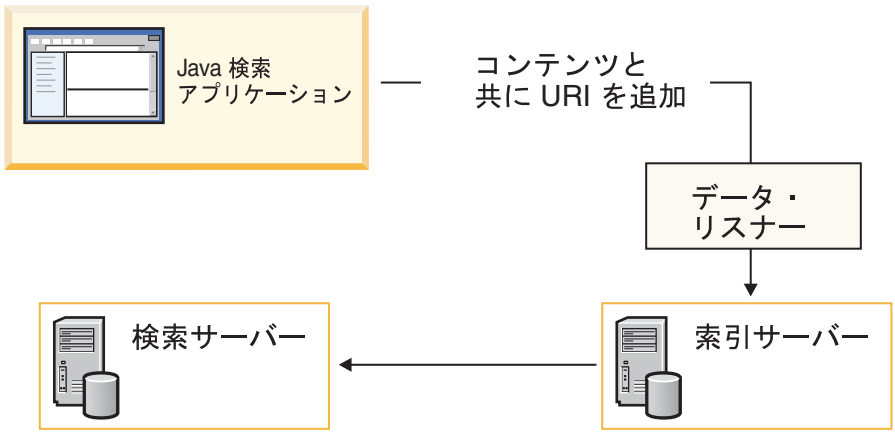

図 4. コンテンツと共に URI を追加する

索引サーバーが検索サーバー上のデータをリフレッシュすると、ユーザーは、追加された URI とそのコンテンツにアクセスできるようになります。

URL にアクセスまたは再アクセスする

データ・リスナー API を使用して、revisitURLs メソッドによって、特定の URL を追加し、そのデータを検索することができます。このメソッドは、Web コンテンツに対してのみ有効です。それ以外のデータ・ソースには、pushData 呼び出しを使用してください。

Visit 新規 URL のデータを検索するように、クローラーに要求します。

Revisit
現在索引にある URL のデータをリフレッシュするように、クローラーに要求します。

関連したコンテンツを指定せずに URL を追加すると、データ・リスナーは要求をクローラーに送信し、クローラーがコンテンツを検索して、それを索引サーバーに送信します。

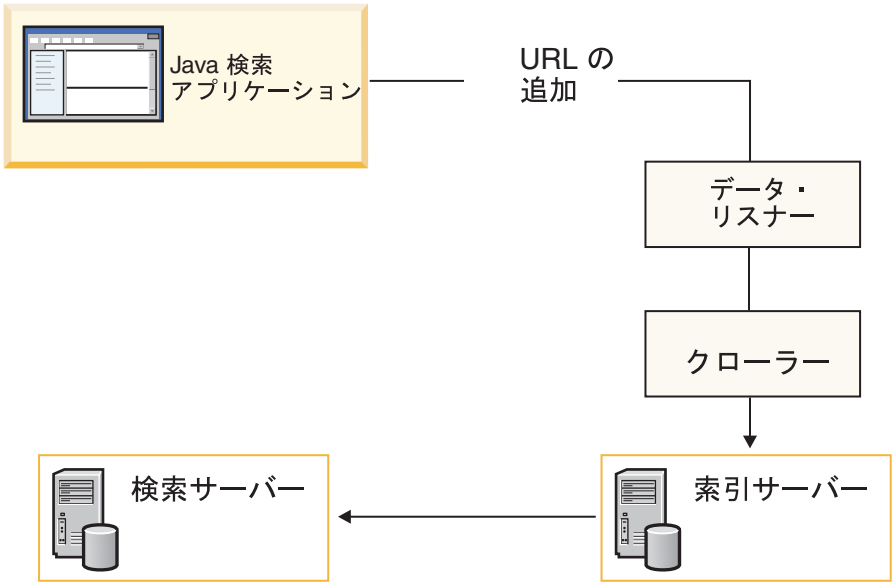

図 5. URL の追加

インデクサーへのデータのプッシュ

データ・リスナー API は、インデクサーに文書をプッシュします。

文書は、次の 3 つのコンポーネントで構成されます。

- URI
- コンテンツ
- メタデータ

メタデータの指定

すべてのデータ追加要求において、データ・プッシュ呼び出しを実行する前に、メタデータ・メソッドを指定する必要があります。メタデータには、文書に関する情報が含まれており、作成者、変更日付、作成日付などの情報も入れられます。

メタデータの作成には、createDataSourceMetadata メソッドを使用します。 各メタデータには、以下の 2 つのセクションがあります。

- CommonMetadata
- DataSourceSpecificMetadata

メタデータの作成

メタデータの作成には、createDataSourceMetadata メソッドを使用します。

CommonMetadata セクションには、全データ・ソースに共通なフィールドが、すべて組み込まれています。これには、次のフィールドがあります。

Datasource

データ・ソースが何であるかを指定します。このフィールドは、常に、フィールド検索が可能で、検索結果に戻されます。

SecurityACL
検索結果としてこの文書が検索されるのにユーザーが必要とする証明書トークンを指定します。複数のトークンを指定する場合は、コンマで区切ります。このフィールドが空であるか、または指定されていない場合は、デフォルト値 PUBLIC が使用されます。これは、すべてのユーザーに使用可能であることを意味します。

Date EPOCH 以降の秒数を表す数値ストリング (負の数値の場合は、EPOCH より前のデータを表します)。日付は、現地時間ではなく、GMT 時間であると見なされます。日付は、文書の静的スコアの計算に使用されます。このフィールドは、日付ベースのランキングや範囲検索のサポートに使用されます。

StaticScoreRef
文書の静的スコアの計算に使用される数値ストリング。このフィールドは、将来の利用に予約されています。

HasSeparateContent
実際のコンテンツが空であるかどうかを示すストリング。コンテンツが空でない場合、属性は、コンテンツのコンテンツ・タイプ、文字セットおよび言語を示します。

以下の例は、共通メタデータ・フィールドを示しています。

```
<Metadata Language="en">
<CommonMetadata
   Datasource = "DB2",
   CrawlerId="MyCrawlerId",
   DatasourceName="My Display Name For My Data Source",
   Date=GMT,
   StaticScoreRef=8>
   <SecurityACLs>Token1,Token2</SecurityACLs>

   <HasSeparateContent ContentType="application/pdf",
      Charset="iso-933",
      Language="gb">
    YES
   </HasSeparateContent>

</CommonMetadata>
</Metadata>
```

DataSourceSpecificMetadata セクションには、そのデータ・ソースに固有のフィールドが組み込まれます。各フィールドの属性を以下に示します。

FieldName
このフィールドのフィールド検索に使用するフィールド名を示します。

Searchable
このフィールドが検索可能であるかどうかを示します。デフォルトは yes です。

FieldSearchable
このフィールドにフィールド検索が必要であるかどうかを示します。デフォルトは yes です。

Metadata
このフィールドの内容を、検索結果の extraFields データに組み込む必要があるかどうかを示します。デフォルトは yes です。

ParametricSearch

このフィールドにパラメトリック検索が必要であるかどうかを示します。デフォルトは no です。yes を指定すると、このフィールドのデータは、構文解析プログラムによって数値に変換できるフォーマットであると見なされます。

ResolveConflict

同じフィールド名のフィールドが複数ある場合に、競合を解決する方法を指定します。たとえば、1 つの文書に 2 つの作成者フィールド (メタデータのものとコンテンツのもの) がある場合などです。有効なオプションには、MetadataPreferred、 ContentPreferred、Coexist (連結) があります。

以下の例は、DataSourceSpecific メタデータ・フィールドを示しています。

```
<Metadata Language="en">
<DatasourceSpecificMetadata>
   <Field FieldName="DatabaseName",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred>
   myDatabase
   </Field>
   Field FieldName="TableName",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred>
   myTable
   </Field>
   <Field FieldName="MYID",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred">
   10
   </Field>
</DataourceSpecificMetadata>
</Metadata>
```

追加メタデータの作成

規定されているフィールドに、さらにメタデータ・フィールドを追加する場合は、addMetaField メソッドを使用します。このメソッドを使用して、既存のメタデータ・オブジェクトにエレメントを追加します。

```
public static void addMetaField(DataSourceMetadata metadata,
                                 String fieldName,
                                 String fieldValue,
                                 boolean searchable,
                                 boolean partOfResult,
                                 boolean fieldSearchable,
                                 boolean parametricSearchable)
```

データ・リスナー API 呼び出しおよびメソッド

データ・リスナーは、以下の API を使用します。

DLDataPusher クラス

データ・リスナーにデータをプッシュして、コレクションから URI を除去し、URL に再アクセスします。

これらの API を呼び出す場合は、データ・リスナーのホスト名とポートを指定し、認証用のクローラー ID とパスワードを指定する必要があります。

pushData メソッド

pushData メソッドは、データをデータ・リスナーにプッシュする API 関数です。

構文

```
public static DLResponse pushData(String hostname,
                                  int port,
                                  String id,
                                  String pw,
                                  String uri,
                                  String col,
                                  String metadata,
                                  byte[] content)
```

パラメーター

hostname
データ・リスナー・サーバーのホスト名。

port
データ・リスナーのポート番号。

ID クローラー ID。

pw クローラー・パスワード。

URI
データの URI。

col ターゲット・コレクションの ID。

metadata
メタデータの XML ストリング・バージョン。

content
データ・リスナーにプッシュされるコンテンツ。

removeURIs メソッド

removeURIs メソッドは、指定されたコレクションから URI を除去する API 関数です。

構文

```
public static DLResponse removeURIs(String hostname,
                                    int port,
                                    String id,
                                    String pw,
                                    String uris,
                                    String col,
```

パラメーター

hostname
データ・リスナー・サーバーのホスト名。

port
データ・リスナーのポート番号。

ID クローラー ID。

pw クローラー・パスワード。

URI
データの URI。

col ターゲット・コレクションの ID。

revisitURLs メソッド

指定されたコレクションの Web Crawler に、URI を追加する、 または URI に再アクセスするように指示します。

構文

```
public static DLResponse removeURIs(String hostname,
                                    int port,
                                    String id,
                                    String pw,
                                    String uris,
                                    String col)
```

パラメーター

hostname
データ・リスナー・サーバーのホスト名。

port
データ・リスナーのポート番号。

ID クローラー ID。

pw クローラー・パスワード。

URI
URI または URI パターン。データを空白文字で区切ります。

col ターゲット・コレクションの ID。

addMetaField メソッド

addMetaField メソッドは、エレメントをメタデータ・オブジェクトに追加します。

構文

```
public static void addMetaField(DataSourceMetadata metadata,
                                String fieldName,
                                String fieldValue,
                                boolean searchable,
                                boolean partOfResult,
                                boolean fieldSearchable,
                                boolean parametricSearchable)
```

createDataSourceMetadata メソッド

createDataSourceMetadata メソッドは、メタデータ・オブジェクトを作成します。

構文

```
public static DataSourceMetadata createDataSourceMetadata(String ds,
                                                           String cid,
                                                           String dsName,
                                                           int score,
                                                           Date dt,
                                                           String language,
                                                           String securityACLs,
                                                           String contentType,
                                                           String charSet,
                                                           byte[] content)
```

データ・リスナー API アプリケーションの例

DLSampleClient クラスに、データ・リスナー API を使用してデータをプッシュする方法を示している簡単な例がいくつか用意されています。

サンプル・アプリケーションでは、次のことが示されています。

- 基本的なデータ・リスナー・タスク
  - 識別と認証
  - データのプッシュ
  - データの除去
- 上級データ・リスナー・タスク
  - メタデータの作成
  - プッシュ・メソッドの呼び出し
  - 結果の検査
  - データ・プッシュの繰り返し

基本的なデータ・リスナー API アプリケーション

DLSampleClient クラスには、基本的なデータ・リスナー API タスクが組み込まれています。

基本的な例には、あるコレクションについて URL を追加したり再アクセスしたりする方法や、別のコレクションから URL を除去する方法が示されています。 次の例が示されています。

- 識別および認証タスク
- データのプッシュ
- データの除去

データ・リスナー・サンプル

```
public class DLSampleClient {
```

識別および認証タスク

データ・リスナーにデータをプッシュするには、サーバーのホスト名とポートを提示する必要があります。また、認証用のクローラー ID とパスワードを指定する必要があります。

```
static void dataPushExample_1(String hostname, int port, String crawlerID,
 String passwd) {
```

プッシュするオブジェクトの作成

```
 DLDataPusher pusher = new DLDataPusher(hostname, port, crawlerID, passwd);
```

追加する URI の指定

クローラーが追加または再アクセスする、 特定の URI または URI パターンのいずれかを指定します。

```
        StringBuffer sb = new StringBuffer();
        sb.append("url1");
        sb.append("¥n");
        sb.append("url*pattern1");
        sb.append("¥n");

        sb.append("url2");
        sb.append("¥n");
        sb.append("url*pattern2");

        String collectionID = "collection2";
        String urls = sb.toString();
```

データの送信

再アクセスする URL のターゲット・コレクションを指定します。

```
pusher.revisitURLs(urls, collectionID);
```

データの除去

URI を除去するターゲット・コレクションを指定します。

```
collectionID = "collection3";
        pusher.removeURIs(urls, collectionID);

    }
```

DB2 Information Integrator の資料

ここでは、DB2 Information Integrator の資料についての情報を提供します。

次のトピックの表は、正式な資料名、資料番号、および PDF 文書の場所を示しています。ハードコピー版の資料を注文するには、正式な資料名または資料番号が必要です。 DB2 Information Integrator のリリース情報とインストール要件の資料名、ファイル名、および場所についても、以下のトピックの中に含まれています。

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料

目的

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料。

表 3. z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

© Copyright IBM Corp. 2004

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料

目的

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料。

表 4. z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
Classic Federation および Classic Event Publishing のクライアント・ガイド	SD88-7512	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の Data Mapper ガイド	SD88-7515	DB2 Information Integrator Support の Web サイト
Classic Event Publishing はじめに	SD88-7516	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のインストール・ガイド	GD88-7517	DB2 Information Integrator Support の Web サイト
Classic Event Publishing オペレーション・ガイド	SD88-7510	DB2 Information Integrator Support の Web サイト
Classic Event Publishing 計画ガイド	SD88-7511	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の管理ガイド	SD88-7509	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のシステム・メッセージ	SD88-7514	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Event Publisher for IMS リリース情報	なし	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Event Publisher for VSAM リリース情報	なし	DB2 Information Integrator Support の Web サイト

Linux、UNIX、および Windows 上のイベント・パブリッシングおよびレプリケーション機能に関する資料

Linux、UNIX、および Windows 上のイベント・パブリッシングおよびレプリケーション機能に関する資料

目的

Linux、UNIX、および Windows 上のイベント・パブリッシングおよびレプリケーション機能に関する資料。

表 5. *Linux、UNIX、および Windows 上のイベント・パブリッシングおよびレプリケーション機能に関する DB2 Information Integrator の資料*

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
インストール・ガイド *(Linux、UNIX、Windows 版)*	GC88-9562	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Migrating to SQL Replication	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
SQL レプリケーション・ガイドおよびリファレンス	SC88-9163	DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Tuning for SQL Replication Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

Linux、UNIX、および Windows 上のフェデレーテッド機能に関する資料

Linux、UNIX、および Windows 上のフェデレーテッド機能に関する資料

目的

Linux、UNIX、および Windows 上のフェデレーテッド機能に関する資料。

表 6. Linux、UNIX、および Windows 上のフェデレーテッド機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
アプリケーション開発者向けガイド	SC88-9609	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発における C++ API リファレンス	SC88-9921	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
データ・ソース構成ガイド	なし	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
フェデレーテッド・システム・ガイド	SC88-9614	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Guide to Configuring the Content Connector for VeniceBridge	なし	DB2 Information Integrator Support の Web サイト
インストール・ガイド (Linux、UNIX、Windows 版)	GC88-9562	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発における Java API リファレンス	SC88-9922	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
マイグレーション・ガイド	SC88-9610	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発者向けガイド	SC88-9923	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト

表 6. *Linux、UNIX、および Windows 上のフェデレーテッド機能に関する DB2 Information Integrator の資料 (続き)*

資料名	資料番号	場所
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

z/OS 上のフェデレーテッド機能に関する資料

z/OS 上のフェデレーテッド機能に関する資料

目的

z/OS 上のフェデレーテッド機能に関する資料。

表 7. *z/OS 上のフェデレーテッド機能に関する DB2 Information Integrator の資料*

資料名	資料番号	場所
Classic Federation および Classic Event Publishing のクライアント・ガイド	SD88-7512	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の Data Mapper ガイド	SD88-7515	DB2 Information Integrator Support の Web サイト
Classic Federation はじめに	GD88-7508	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のインストール・ガイド	GD88-7517	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の管理ガイド	SD88-7509	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のシステム・メッセージ	SD88-7514	DB2 Information Integrator Support の Web サイト
Classic Federation トランザクション・サービス・ガイド	SD88-7513	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Federation for z/OS リリース情報	なし	DB2 Information Integrator Support の Web サイト

z/OS 上のレプリケーション機能に関する資料

z/OS 上のレプリケーション機能に関する資料

目的

z/OS 上のレプリケーション機能に関する資料。

表 8. z/OS 上のレプリケーション機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	DB2 Information Integrator Support の Web サイト
Migrating to SQL Replication	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Replication Installation and Customization Guide for z/OS	SC18-9127	DB2 Information Integrator Support の Web サイト
SQL レプリケーション・ガイドおよびリファレンス	SC88-9163	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Tuning for SQL Replication Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

Linux、UNIX、および Windows 上のエンタープライズ・サーチ機能に関する資料

Linux、UNIX、および Windows 上のエンタープライズ・サーチ機能に関する資料

目的

Linux、UNIX、および Windows 上のエンタープライズ・サーチ機能に関する資料。

表 9. Linux、UNIX、および Windows 上のエンタープライズ・サーチ機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
エンタープライズ・サーチの管理	GD88-6374	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ インストール・ガイド	GD88-6373	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ プログラミング・ガイドおよび API リファレンス	SD88-6375	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ リリース・ノート	なし	DB2 Information Integrator Support の Web サイト

リリース情報とインストール要件

リリース情報には、製品のリリースとフィックスパック・レベルに特有の情報が入っています。 また、それぞれのリリースの資料に対する最新の訂正も含まれています。インストール要件には、 製品のリリースに特有の情報が入っています。

表 10. DB2 Information Integrator のリリース情報とインストール要件

資料名	ファイル名	場所
Installation Requirements for IBM DB2 Information Integrator Event Publishing Edition, Replication Edition, Standard Edition, Advanced Edition, Advanced Edition Unlimited, Developer Edition, and Replication for z/OS	Prereqs	• *DB2 Information Integrator* 製品 CD • DB2 Information Integrator のインストール・ランチパッド
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	ReleaseNotes	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	なし	DB2 Information Integrator Support の Web サイト

表 10. DB2 Information Integrator のリリース情報とインストール要件 (続き)

資料名	ファイル名	場所
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	なし	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ リリース・ノート	なし	DB2 Information Integrator Support の Web サイト

リリース情報とインストール要件の表示

リリース情報とインストール要件の表示

目的

Windows オペレーティング・システム上で 、CD に入っているインストール要件およびリリース情報を表示するには、次のように入力します。

x¥doc¥*%L*

パラメーター

x Windows CD ドライブ名

%L
使用したい資料のロケール。例えば、en_US

目的

UNIX オペレーティング・システム上で 、CD に入っているインストール要件およびリリース情報を表示するには、次のように入力します。

/*cdrom*/doc/*%L*

パラメーター

cdrom
CD の UNIX マウント・ポイント

%L
使用したい資料のロケール。例えば、en_US

PDF 文書の表示と印刷

PDF 文書の表示と印刷

DB2 PDF Documentation CD から DB2 Information Integrator PDF ブックを表示および印刷するには、次のようにします。

1. *DB2 PDF Documentation* CD のルート・ディレクトリーから、index.htm ファイルをオープンします。
2. 使用したい言語をクリックします。
3. 表示したい文書のリンクをクリックします。

DB2 Information Integrator の資料へのアクセス

DB2 Information Integrator の資料へのアクセス

すべての DB2 Information Integrator ブックおよびリリース情報の PDF ファイルは、 www.ibm.com/software/data/integration/db2ii/support.html にある DB2 Information Integrator Support の Web サイトから入手できます。
DB2 Information Integrator Support の Web サイトから、最新の DB2 Information Integrator 製品資料にアクセスするには、図 6 に示すように、「Product Information」リンクをクリックします。

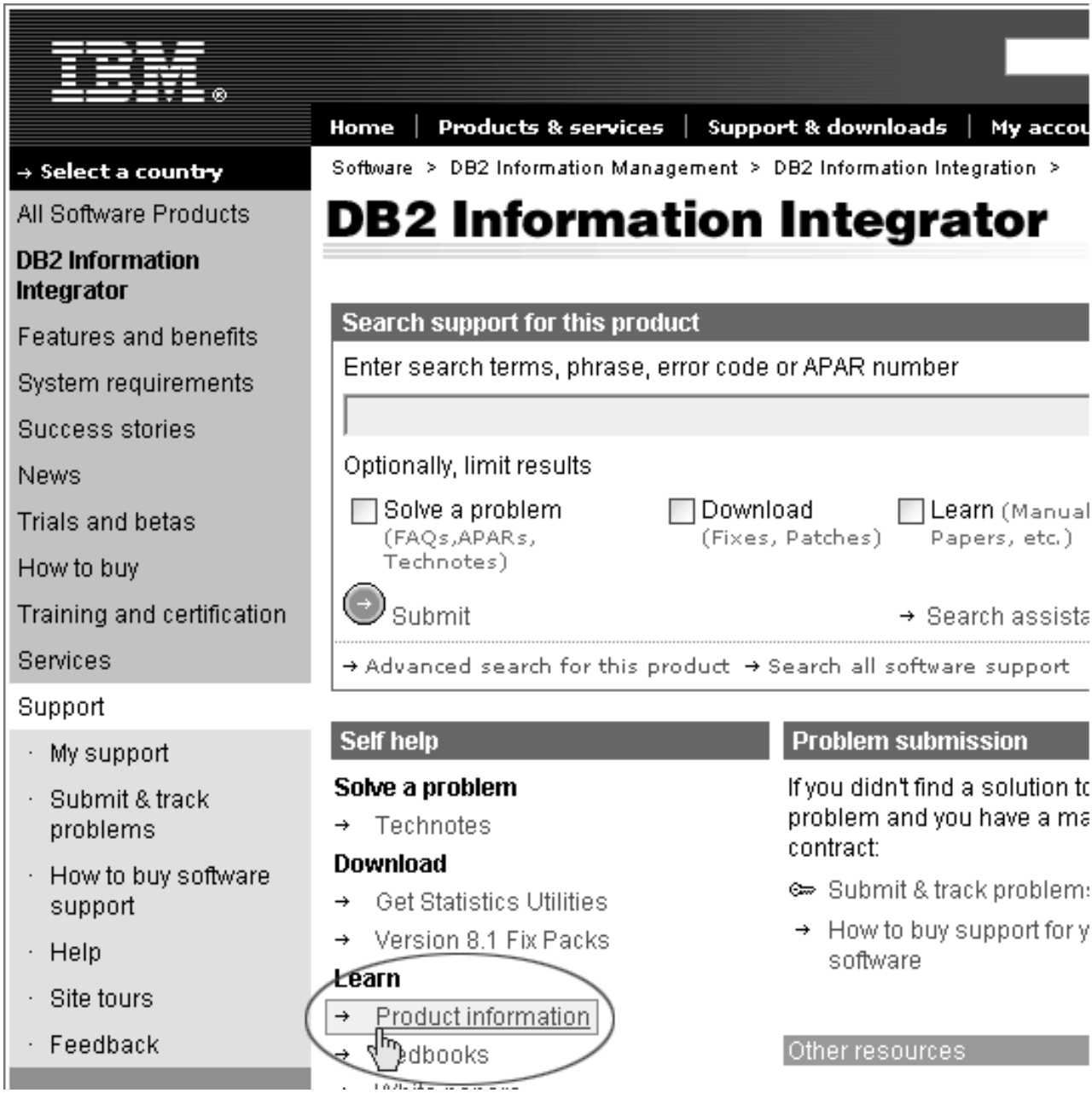

図 6. DB2 Information Integrator Support の Web サイトの「Product Information」リンク

「Product Information」リンクから、サポートされるすべての言語の最新の DB2 Information Integrator の資料にアクセスできます。

- DB2 Information Integrator 製品資料 (PDF ファイル)
- リリース情報も含めた、フィックスパック製品資料
- Linux、UNIX、および Windows の DB2 Information Center のダウンロードとインストールの説明
- DB2 Information Center オンラインへのリンク

DB2 Information Integrator Support の Web サイトは、 サポート資料、IBM Redbooks、白書、製品のダウンロード、ユーザー・グループへのリンク、および、DB2 Information Integrator についてのニュースも提供します。

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 のユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 92 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 92 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 92 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード・フォーカス

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

キーボード入力

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは、「キーボード・ショートカットおよびアクセラレーター: Common GUI help」を参照してください。

キーボード・ナビゲーション

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

© Copyright IBM Corp. 2004

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、「キーボード・ショートカットおよびアクセラレーター: Common GUI help」を参照してください。

アクセスしやすい表示

アクセスしやすい表示

目的

アクセスしやすい表示

フォントの設定

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、「メニューおよびテキストのフォントを変更する: Common GUI help」を参照してください。

色に依存しない

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

アクセスしやすい資料

DB2 の資料は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン資料にアクセスする場合にのみ使用できます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供しない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、製造元によって明示的に指定されたものを除き、他社の製品, プログラムまたはサービスを使用した場合の評価と検証はお客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものでは ありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

© Copyright IBM Corp. 2004

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠し

たアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

Outside In (®) Viewer Technology, ©1992-2004 Stellent, Chicago, IL., Inc. All Rights Reserved.

IBM XSLT Processor Licensed Materials - Property of IBM ©Copyright IBM Corp., 1999-2004. All Rights Reserved.

商標

ここでは、IBM の商標と、特定の IBM 以外の商標をリストします。

以下は、IBM Corporation の商標です。

IBM
AIX
AIX 5L
DB2
DB2 Universal Database
Domino
Informix
Lotus
Lotus Notes
Notes
OmniFind
WebSphere
xSeries
z/OS

以下は、それぞれ各社の商標または登録商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Intel、Intel Inside (ロゴ)、MMX および Pentium は、Intel Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

IBM と連絡を取る

お客様の国または地域で IBM に連絡する方法については、Web の www.ibm.com/planetwide にある「IBM Directory of Worldwide Contacts」にアクセスしてください。

© Copyright IBM Corp. 2004

製品情報

DB2 Information Integrator に関する情報は、Web により入手できます。

Web の www.ibm.com/software/data/integration/db2ii/support.html にアクセスしてください。

1. 製品の注文または一般情報の入手: 1-800-IBM-CALL (1-800-426-2255)
2. 資料の注文: 1-800-879-2755
3. Web の www.ibm.com/software/data/integration/db2ii/support.html に アクセスしてください。

このサイトには、次の最新情報が入っています。

- 技術ライブラリー
- 資料の注文方法
- 製品のダウンロード
- ニュースグループ
- フィックスパック
- ニュース
- Web リソースへのリンク

© Copyright IBM Corp. 2004

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[マ行]

© Copyright IBM Corp. 2004

A

B

C

D

F

G

H

I

J

M

N

P

Q

R

S

T

U

Printed in Japan

SD88-6375-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12