IBM Cúram Social Program Management

Cúram Generic Search Server

Version 6.0.5

IBM Cúram Social Program Management

Cúram Generic Search Server

Version 6.0.5

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 53

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Cette édition s'applique à IBM Cúram Social Program Management v6.0 5 et à toutes les versions ultérieures, sauf indication contraire dans de nouvelles éditions.

Table des matières

Figures

Tableaux

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Chapitre 1. Introduction

1.1 Manuel Cúram Generic Search Server Guide

Cúram Generic Search Server est un outil fourni par IBM Corporation qui peut être utilisé pour développer des recherches performantes et évolutives pour votre solution d'application.

Ce document décrit Cúram Generic Search Server et fournit une présentation de son architecture. C'est également une référence pour la configuration de Generic Search Server et de ses tables de base de données. Finalement, il fournit un exemple d'implémentation d'une recherche de bout en bout à l'aide de Cúram Generic Search Server.

1.2 Prérequis

Les lecteurs du manuel Cúram Generic Search Server Guide doivent être familiers avec l'architecture Cúram, la modélisation Cúram, les processus et les constructions de développement.

1.3 Public concerné

Ce document est destiné à être lu par les architectes, les concepteurs et les développeurs intéressés par l'utilisation de Cúram Generic Search Server pour l'implémentation de pages de recherche.

Chapitre 2. Concepts et définitions

2.1 Introduction

Ce chapitre présente plusieurs concepts importants de recherche et d'indexation, ainsi que des définitions associées à Cúram Generic Search Server et utilisées dans ce document.

2.2 Generic Search Server

Cúram Generic Search Server est une application autonome qui prend en charge la recherche performante de données d'application grâce à plusieurs interfaces de programme d'application. En outre, Generic Search Server est implémenté avec l'interface de programme d'application Apache Lucene. Ceux qui implémentent les recherches GSS doivent uniquement utiliser les interfaces de programme d'application disponibles pour GSS.

Generic Search Server peut être déployé comme une application Java™ complète (pour faciliter les tests de délai de développement) et comme une application J2EE.

2.3 Index

Le concept d'index de recherche se trouve au coeur de Generic Search Server. Il s'agit d'une représentation performante (pas une base de données) d'un ensemble de données consultables associées. Un index Generic Search Server est un «index inversé» qui mappe des mots à des enregistrements de base de données dans lesquels ces mots apparaissent.

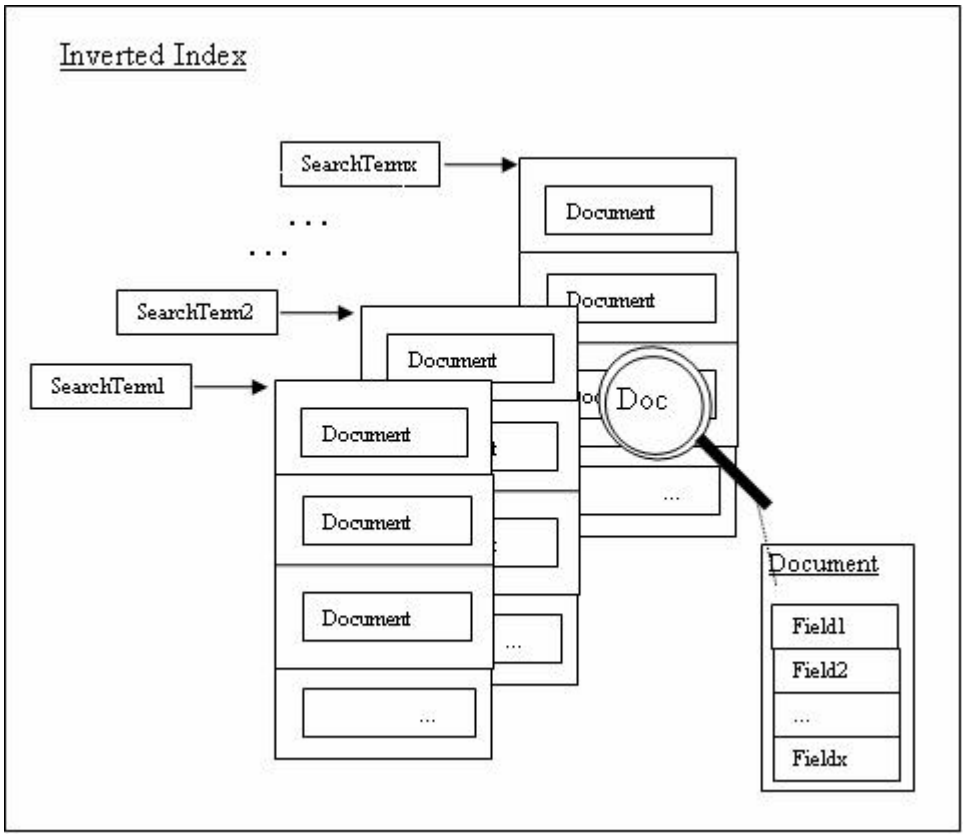

Figure 1. Description d'index inversé

Lors de la recherche d'un mot dans un index, tous les enregistrements correspondants sont récupérés sans avoir à effectuer une recherche de fichiers volumineux. Par conséquent, de tels index s'adaptent bien, et pour les systèmes volumineux, il est possible d'exécuter plusieurs index en parallèle, ce qui permet d'excellentes performances de recherche lorsque les paramètres appropriés de configuration du déploiement et d'optimisation des index sont sélectionnés.

Les développeurs générant des recherches d'application ne manipulent ni ne gèrent directement les index, c'est Generic Search Server qui s'en charge en arrière-plan.

2.4 Service de recherche

Un service de recherche décrit :

1. Les informations associées aux zones recherchées
2. Les analyseurs utilisés dans chaque zone, les types de données de zone
3. Les informations d'entité pour le remplissage d'un index d'exécution
4. Les statuts du service de recherche ("mis à jour" ou "requiert une synchronisation")

Vu sous cet angle, un service de recherche se résume à des métadonnées. Toutefois, ce document utilise également le terme à décrire dans l'index d'exécution rempli.

Un service de recherche doit être défini pour chaque ensemble discret de données à interroger (par exemple, recherche de personne, de paiement, etc.). Chaque recherche effectuée doit spécifier le service de recherche avec lequel elle fonctionne.

2.5 Zone

Comme indiqué plus haut, les services de recherche sont composés d'ensembles de zones. Ces dernières peuvent être considérées comme relativement analogues aux définitions de colonnes dans les tables de base de données. Une zone a un nom et un type. Lorsque elle est renvoyée à partir d'une recherche, elle a également une valeur, qui correspond au résultat.

Les zones peuvent être marquées comme "Stored" (Stockées). Les zones marquées de la sorte permettent à l'index de contenir physiquement des valeurs pertinentes extraites (voir 2.13, «Extracteur», à la page 6) de la base de données. Cela signifie que leurs valeurs peuvent être directement extraites de l'index après une recherche, puis renvoyée au demandeur sans qu'aucun accès à l'enregistrement associé dans la table de base de données d'application ne soit nécessaire. Remarquez toutefois que cela n'augmente pas la taille de l'index et que cela peut avoir une incidence sur les performances de la recherche.

Les zones peuvent également être marquées comme "Indexed" (Indexées) ou non. Les zones marquées de la sorte sont interrogeables et celles qui ne le sont pas ne sont pas interrogeables. Cette fonction est utile pour les zones comme les ID uniques qu'il peut être intéressant de stocker dans l'index mais qui ne sont pas interrogées.

Remarquez que les zones ne doivent pas être marquées comme "Stored" (Stockées) pour être interrogeables.

2.6 Document

Un document est un enregistrement dans un index. Un document est à son tour constitué d'un ensemble de zones. Les résultats de la recherche sont renvoyés de Generic Search Server sous forme d'ensembles de documents, qui peuvent ensuite être convertis en objets de struct Cúram. Par exemple, un document de recherche de personne peut se composer des zones Prénom, Nom de famille, Adresse, Genre, etc. Effectuer une requête/recherche de personne (voir 2.9, «Requête», à la page 6) sur base d'un certain nombre de critères d'entrée renvoie zéro ou plusieurs documents de ce type.

2.7 Lucene

Lucene est un projet open source créé par Apache Software Foundation. En arrière-plan, Cúram Generic Search Server utilise Lucene pour ses fonctions d'indexation et de recherche.

Remarque : Remarquez que les informations relatives à l'indexation et à Lucene sont uniquement fournies à des fins informatives. Les développeurs générant des recherches à l'aide de Generic Search Server n'ont pas besoin de manipuler directement des index ou des objets Lucene. Ces manipulations sont gérées par l'interface de programme d'application de Generic Search Server.

2.8 Base de données de transfert

La base de données de transfert Generic Search Server se compose d'un ensemble de tables de base de données utilisées aux fins suivantes :

- Pour stocker les définitions du service de recherche : les informations relatives aux services de recherche disponibles avec leur structure
- Pour stocker les valeurs extraites de la base de données opérationnelle qui sont utilisées pour remplir les index correspondant aux définitions du service de recherche.

Les justifications de conception fondamentales pour l'utilisation des tables de base de données comme intermédiaire sont les suivantes :

- Elles déchargent les recherches à partir de la base de données principale, ce qui signifie que les recherches n'ont pas d'incidence sur les performances du système opérationnel

- Elles sont conservées de manière appropriée pour le service de recherche. Les données sont conservées dans une forme appropriée pour les objectifs de génération d'index de recherche. Les données d'application sont converties, épurées et consolidées avant d'être stockées dans la base de données de transfert. Par conséquent, les travaux par lots ne doivent pas continuellement ré-extraire les données à chaque démarrage d'une instance de Generic Search Server.

2.9 Requête

Une requête est un objet (une struc, pour être précis) transmis à Generic Search Server lorsqu'une recherche est effectuée.

2.10 Terme

Un terme est une partie d'un objet de requête. Actuellement, il existe trois types différents de terme : les termes Standard, pour la recherche de zones de texte régulière, les termes Date, pour la recherche de zones de date et les termes DateRange, pour la spécification d'une plage de dates à interroger.

2.11 Analyseur

Un analyseur est un concept Lucene, qui représente une classe implémentant la classe abstraite Lucene `org.apache.lucene.analysis.Analyzer`.

Les analyseurs préparent le texte pour l'indexation et la recherche. Par exemple, il n'est pas judicieux d'indexer tous les mots d'une zone de texte, les mots vides comme «et», «de» et «un(e)» peuvent être sans intérêt dans une recherche. S'ils doivent être ignorés lors d'une recherche par zone, alors la zone est segmentée, c'est-à-dire, traitée par l'analyseur avant l'écriture de la zone dans l'index. C'est également le cas pour toute valeur de terme recherchée.

Les analyseurs sont spécifiques pour chaque langue, ce qui définit un mot est différent selon la langue. Certains analyseurs peuvent être configuré pour ignorer les mots vides courants (un(e), le/la/les, si, etc.), les chiffres, etc. Les analyseurs utilisés par Generic Search Server peuvent être configurés pour chaque service de recherche.

2.12 Associateur

Un associateur est une classe écrite par des développeurs de recherches d'application pour chaque service de recherche. Sa fonction est de convertir les données de l'application dans un format qui peut être écrit dans la base de données de transfert, puis de les importer dans un index. La conversion implique l'identification des propriétés d'entité pertinentes intéressantes pour le service de recherche, la génération d'une liste de ces valeurs et leur mappage à une valeur de texte consolidée unique. Cette valeur, stockée dans la base de données de transfert, est utilisée ultérieurement dans la génération d'un document d'index de recherche unique. Chaque service de recherche écrit doit fournir sa propre implémentation d'associateur.

2.13 Extracteur

L'extracteur utilise les métadonnées du service de recherche pour obtenir les données d'application pertinentes nécessaires pour remplir les index de recherche. L'extracteur interroge les entités d'application pertinentes identifiées via les métadonnées et les propriétés d'entité nécessaires sont mappées (avec l'associateur) à la base de données de transfert pour l'indexation au démarrage du service de recherche.

Chapitre 3. Présentation de Generic Search Server

3.1 Generic Search Server et Lucene

Les concepts derrière l'indexation et l'interface de programme d'application de Lucene ont déjà été présentés. Pourquoi ne pas simplement utiliser directement Lucene dans l'application Cúram ?

Bien que Lucene soit une excellente interface de programme d'application pour l'indexation et la recherche, elle ne répond pas à toutes les exigences d'un produit de recherche Cúram :

- Elle ne gère pas les problèmes de déploiement : comment exécuter plusieurs serveurs de recherche, comment l'application doit communiquer avec les serveurs de recherche, etc.
- Elle ne gère pas le problème d'importation des données dans les index.
- Elle ne gère pas le problème de synchronisation des données d'index avec les données source dans l'application en cours d'exécution.
- Elle ne gère pas le problème d'interprétation des données renvoyées à partir d'une recherche d'index comme les structs et les types de données Cúram.
- Elle ne gère par l'exigence d'application plus globale de protection du développeur d'applications à partir de la connaissance approfondie de produits tiers spécifiques. Etant donné que Lucene est seulement une solution de recherche potentielle, il semblerait plus sensé de fournir une interface de programme d'application de recherche plus générique.

Cúram Generic Search Server a été développé pour répondre à ces exigences.

3.2 Importation de données à partir de Cúram

L'utilisation d'une technologie d'index implique qu'avant de pouvoir effectuer une recherche d'index, il faut d'abord créer ce dernier. Etant donné qu'une grande partie du travail de recherche est s'effectue en amont dans la génération d'index, les recherches d'exécution s'accélèrent. Toutefois, il est intéressant de remarquer que le processus d'indexation lui-même peut durer un moment et cette durée augmente proportionnellement à la quantité de données à indexer.

L'initialisation de Generic Search Server s'effectue en deux phases.

Lors de la première phase, les données d'application existantes sont exportées à partir de l'application dans un ensemble de tables de base de données utilisées par Generic Search Server (les tables de transfert). Cette exportation a été implémentée comme un traitement par lots, appelé Database Search Extractor, et elle est fournie dans le cadre de la distribution de Generic Search Server. L'exportation nécessite seulement d'être exécutée une fois, lorsque Generic Search Server est utilisé en premier. Des classes auxiliaires spéciales appelées "Associateurs" sont nécessaires pour chaque service de recherche : celles-ci aident l'extracteur à préparer les données pour l'importation dans les tables de transfert.

Durant la seconde phase, un index est généré pour chaque service de recherche défini. Lorsque Generic Search Server est démarré, un processus s'exécute pour lire les données appropriées à partir des tables de base de données de transfert et générer les index et les autres structures de donnés à utiliser pour effectuer les recherches. Lorsque les index sont générés, le serveur est en mesure de rechercher les requêtes. Des informations relatives à l'optimisation de ces performances sont disponibles dans le Chapitre 12, «Performances», à la page 43

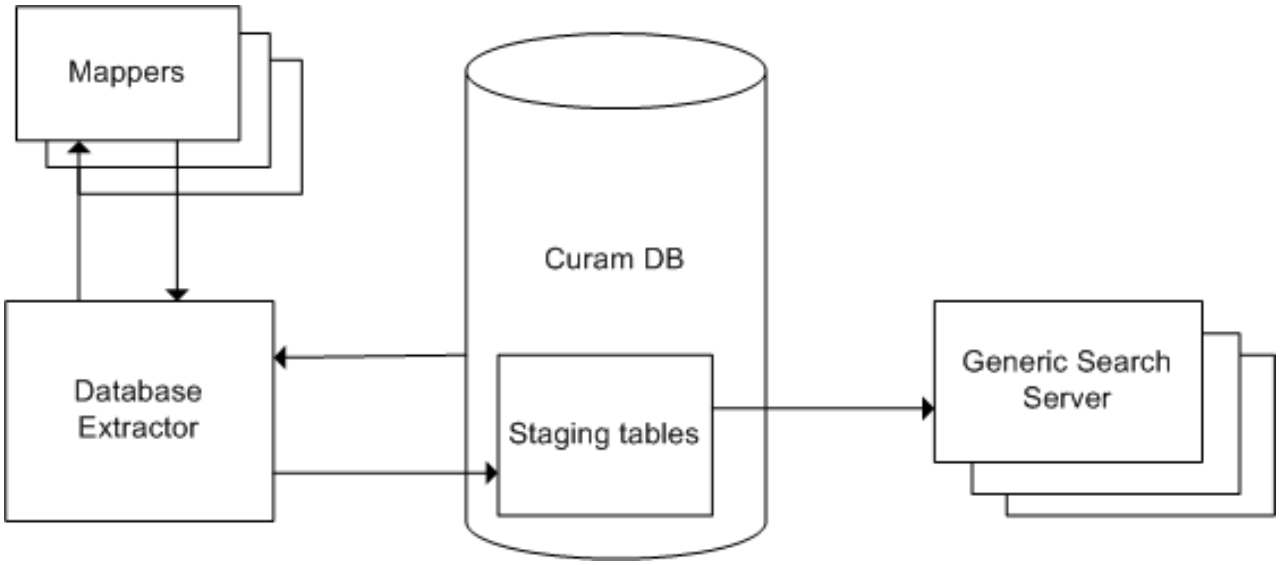

Figure 2. Processus de démarrage de Database Extractor et de Generic Search Server

3.3 Synchronisation du serveur de recherche

Etant donné que Generic Search Server n'effectue pas de recherche sur les données réelles mais sur un index généré à partir de ces données, les mises à jour des données d'application doivent être répliquées dans l'index. Dans les implémentations Cúram, il est essentiel que les mises à jour des données interrogeables se reflètent dans les index concernés de manière prévisible et opportune. Avec Generic Search Server, le délai est court (et configurable).

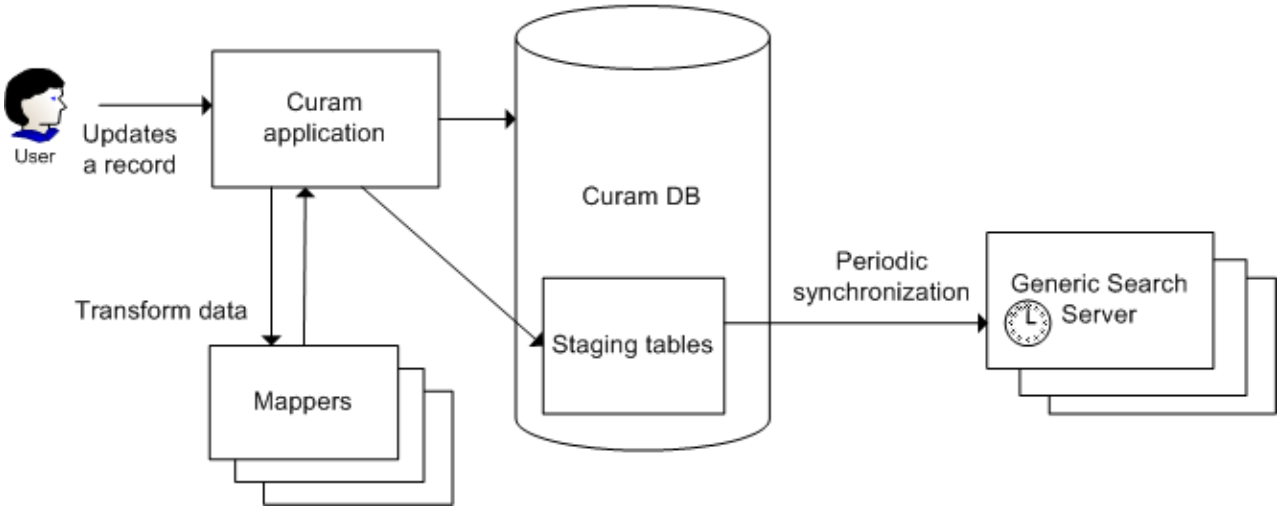

Figure 3. Synchronisation de données

Tout comme l'importation initiale de données décrite ci-avant, il existe deux étapes dans le processus de synchronisation.

La première étape du processus se produit lorsque les données d'application (utilisées dans un index) changent, généralement suite à une insertion, une mise à jour ou une suppression logique. Lorsque cela se produit, l'application doit écrire des informations sur ce changement de données dans les tables de transfert de Generic Search Server. Tous les éléments, nouveaux et mis à jour, sont marqués avec un horodatage.

Durant la seconde étape (qui se produit périodiquement), Generic Search Server synchronise ses index avec le contenu actuel de la base de données de transfert. Pour ce faire, il lit tous les éléments modifiés depuis la dernière synchronisation, puis importe ces éléments dans les index. Cela se réalise en particulier par la comparaison des horodatages associés avec chaque élément modifié avec le dernier horodatage utilisé durant la précédente étape de synchronisation.

Remarque : Lors de l'écriture des tests d'unité qui comprennent des appels de recherches Generic Search Server, il est important de garder à l'esprit le temps d'attente de synchronisation des données. En outre, puisque l'instance de Generic Search Server s'exécute dans un processus distinct des tests d'unité, elle ne fait pas partie de la même transaction. Par conséquent, les synchronisations Generic Search Server ne récupérent pas les données modifiées dans la transaction de test, sauf si cette dernière est explicitement validée.

3.4 Contrôleur de recherche

Le contrôleur de recherche est un composant important du mécanisme de synchronisation. Il conserve une liste de toutes les entités associées avec chaque service de recherche.

Lorsqu'une entité change, le contrôleur de service peut vérifier et déterminer si cette entité est utilisée par un ou plusieurs services de recherche. Si elle est utilisée, les données de la base de données de transfert doivent être mises à jour dans la même transaction que l'entité mise à jour. Le contrôleur de recherche fournit également une interface de programme d'application pour la mise à jour de la base de données de transfert.

Remarque : Plusieurs entités Cúram Platform (qui apparaissent dans certaines recherches Cúram Platform) ont été modifiées de sorte à permettre l'implémentation de telles mises à jour de synchronisation dans la future édition. Ces modifications ont pris la forme de la création de points d'exit pré ou post-opération qui contiennent des implémentations raccordées. Ces points d'exit de pré et post-opération sont réservés pour une implémentation ultérieure et ne doivent pas être modifiés directement par les clients.

3.5 Processus de recherche

Le processus de recherche peut être décomposé en trois phases.

Lors de la première phase, l'application Cúram génère une requête valide à présenter à Generic Search Server. Elle remplit cette requête avec les critères de recherche entrés par l'utilisateur.

Lors de la seconde phase, l'application Cúram contacte une instance Generic Search Server active et effectue la recherche comme défini par l'objet de requête.

Lors de la phase finale, l'application Cúram interprète les résultats qu'elle reçoit de Generic Search Server sous forme de types de données Cúram, effectue ses vérifications de sécurité habituelles relatives à la sensibilité des données, puis les affiche pour l'utilisateur.

3.6 Références

Site Web de Lucene : http://lucene.apache.org/.

Chapitre 4. Recherches prises en charge par Generic Search Server

4.1 Introduction

IBM Corporation a introduit le serveur de recherche générique comme mécanisme de recherche facultatif pour les recherches de plateforme et de module de solution. Plusieurs recherches ont été implémentées grâce à la recherche des bases de données et à Cúram Generic Search Server, et certaines sont disponibles uniquement en tant que recherches GSS. Pour les recherches disponibles en tant que recherches GSS ou de bases de données, les clients peuvent, à chaque utilisation, activer ou désactiver la recherche performante en définissant les propriétés d'application.

4.2 Propriétés associées à Generic Search Server dans l'application Cúram

Ces propriétés sont les propriétés du système d'application et peuvent être gérées normalement l'administration des propriétés dans l'application. Toutes les propriétés concernées sont disponibles sous la catégorie appelée «Application - Lucene enhanced search parameters» (Application - Paramètres de recherche améliorés Lucene). Une liste complète de ces propriétés se trouve dans A.1, «Propriétés de configuration», à la page 47

Tableau 1. Propriétés associées à Cúram Generic Search Server

Nom de la propriété	Description
curam.lucene.luceneEnhancedSearchEnabled	Par défaut : «NO». Par défaut, toutes les fonctionnalités Generic Search Server sont désactivées. Pour les activer, vous devez définir cette propriété sur «YES» pour activer la recherche améliorée. A moins qu'elle soit définie sur «YES», aucune recherche améliorée n'est disponible.
curam.lucene. luceneOnlineSynchronizationEnabled	Par défaut : «NO». Pour activer le mécanisme de publication d'événement qui effectue les changements dans les données interrogeables disponibles dans le serveur de recherche, vous devez définir cette propriété sur «YES». A moins que ce soit fait, les insertions et les mises à jours des données interrogeables ne seront pas propagées dans le serveur de recherche.
curam.lucene.externalUpdateEventsEnabled	Par défaut : «NO». Pour vous assurer que lorsqu'une donnée associée au service de recherche est mise à jour de l'extérieur, alors le système externe reçoit les événements de synchronisation de mise à jour associés pour synchroniser les données interrogeables, au cas où la propriété "curam.lucene.luceneOnlineSynchronizationEnabled" n'est pas activée. L'activation de cette propriété a le même impact que l'activation de «curam.lucene.luceneOnlineSynchronizationEnabled» dans l'application. Pour activer la propriété «curam.lucene.externalUpdateEventsEnabled» définissez cette propriété sur «YES».

Finalement, chaque recherche prenant en charge la recherche avancée dispose d'une propriété qui détermine si elle utilise Generic Search Server ou la base de données. Cela permet à chaque organisation de choisir, à chaque recherche, quelles recherches avancées utiliser.

4.3 Maintenir la synchronisation des données Cúram et des données de recherche

Il n'est pas nécessaire de maintenir la synchronisation entre les données d'application opérationnelles et l'index de recherche lorsque les résultats de la recherche doivent être précis. L'infrastructure fournie par GSS pour ce faire a été décrite auparavant (voir 6.3, «Contrôleur de recherche», à la page 18).

Toutefois, les développeurs d'applications ont également une responsabilité pour l'ajout d'appels au contrôleur de recherche lorsque les données concernées changent dans l'application. Cette section décrit, à titre informatif, l'approche basée sur les événements qui est utilisée et recommandée aux clients pour l'implémentation de leurs propres recherches avec GSS.

Outre le mécanisme d'événement, nous fournissons également la synchronisation de l'associateur d'extraction, décrite dans le chapitre qui lui est consacré de ce manuel, voir Chapitre 8, «Associateur d'extraction», à la page 29.

4.3.1 Synchronisation basée sur les événements

Cúram fournit des événements qui permettent aux parties mal couplées de l'application de se fournir mutuellement des informations relatives aux changements d'état. Ils sont décrits dans le manuel *Cúram Server Developer's Guide.*.

Chaque entité qui contribue à un service de recherche doit avoir des événements lorsqu'elle est créée, supprimée ou modifiée. Le gestionnaire d'événements appelle ensuite la classe `SearchController` pour mettre à jour le serveur de recherche avec le changement.

Toute entité qui contribue au service de recherche doit contenir les opérations postmodify, postinsert et postremove qui génèrent les événements.

Chapitre 5. Tables de base de données de transfert

5.1 Introduction

Les tables de base de données de transfert sont des tables de base de données dans la base de données opérationnelle qui sont utilisées par Generic Search Server. Il existe quatre de ces tables : SearchService, SearchServiceField, SearchServiceRow et SearchSrvcRowExt.

Ce chapitre détaille le but et la structure des tables SearchService et SearchServiceField. Les développeurs générant des services de recherche n'ont pas besoin d'accéder directement aux tables SearchServiceRow ou SearchSrvcRowExt, ni d'écrire des fichiers DMX pour elles.

La table SearchService définit les services de recherche connus de Generic Search Server (voir 2.4, «Service de recherche», à la page 4 pour une présentation des services de recherche). Etant donné qu'aucune API d'administration n'a été fournie pour la gestion des services de recherche, les enregistrements du service de recherche doivent actuellement être créés et gérés en accédant à la table de la base de données ou en éditant les fichiers DMX, puis en régénérant la base de données d'application.

La table SearchServiceField définit une zone unique d'un service de recherche : son nom, son type de données et plusieurs autres attributs expliqués entièrement ci-dessous. Chaque ligne de la base de données SearchServiceField est associée à une ligne SearchService unique. Comme avec le service de recherche, les enregistrements de la zone du service de recherche doivent actuellement être créés et gérés en accédant directement à la table de base de données ou en éditant les fichiers DMX, puis en générant à nouveau la base de données d'application.

SearchServiceRow est une table utilisée pour stocker des données interrogeables à partir de l'application afin de les utiliser pour la génération d'index. Generic Search Server fournit une interface de programme d'application (voir Chapitre 6, «Guide d'initiation à l'interface de programme d'application de Generic Search Server», à la page 17 et Chapitre 7, «Implémentation d'une recherche avec Generic Search Server», à la page 23) utilisée pour manipuler SearchServiceRows : les développeurs doivent interagir avec cette table de base de données uniquement par l'interface de programme d'application plutôt qu'en y accédant directement.

Il existe deux autres tables de base de données GSS : GSSMapperType et GSSEntity. Ces dernières sont uniquement utilisées avec la fonction d'associateur d'extraction, autrement, elles sont ignorées. Ces tables sont décrites dans Chapitre 8, «Associateur d'extraction», à la page 29.

5.2 Table SearchService

Chaque service de recherche doit contenir un enregistrement dans la table SearchService. Avec ses lignes SearchServiceField enfant, la table SearchService définit le schéma pour chaque service de recherche. Une description de chaque colonne de la table SearchService est fournie ci-dessous :

5.2.1 searchServiceId

L'identificateur du service de recherche : une chaîne utilisée pour identifier de manière unique un service de recherche.

5.2.2 extKeyName

Le nom d'une zone de service de recherche qui identifie chaque enregistrement de manière unique dans un index créé à partir de cette définition de service de recherche. Il est essentiel que les valeurs de l'index correspondant à cette zone de service de recherche soient uniques, comme lorsque les données

interrogeables sont mises à jour dans la base de données d'application, la valeur de cette zone est utilisée pour identifier le document approprié à mettre à jour dans l'index.

5.2.3 analyseur

L'analyseur du service de recherche à utiliser lors de la conversion à partir des termes de texte de la base de données d'application vers les termes d'index. Le contenu de cette colonne indique un des noms d'analyseur prédéfinis fournis par Generic Search Server (voir la liste ci-dessous) ou un class-name Java entièrement qualifié qui implémente la classe abstraite `org.apache.lucene.analysis.Analyzer`. Il peut s'agir d'un analyseur Lucene standard ou d'une implémentation personnalisée ou tierce. Remarquez que la classe doit être disponible dans le chemin d'accès aux classes Generic Search Server s'il ne s'agit pas d'un analyseur Lucene standard.

Pour obtenir une liste des analyseurs fournis avec GSS et des informations plus précises sur la sélection d'un analyseur, voir 9.6, «Analyseurs en profondeur», à la page 35.

5.2.4 frcdReidxTimeStmp

Utilisée par l'extracteur pour forcer Generic Search Server à régénérer ses index après une extraction. Lors de la création des enregistrements du service de recherche, cette valeur doit initialement être nulle.

5.2.5 mapperName

Le nom de l'implémentation d'associateur (voir 7.4, «Implémentation d'opérations de l'associateur», à la page 24). Une implémentation d'associateur est une classe convertissant un ensemble de données d'entité d'application dans un format approprié pour l'indexation. La valeur de cette colonne doit être le class-name entièrement qualifié de la classe d'associateur, et comme avec l'implémentation d'analyseur, elle doit se trouver dans le chemin d'accès aux classes d'exécution de Generic Search Server (si l'associateur est développé en tant que partie de l'application, il se trouve dans le chemin d'accès aux classes par défaut).

5.2.6 dbLastWritten

Utilisée lors de la synchronisation. Elle ne doit pas être initialisée, ni mise à jour par le code d'application, ni par les administrateurs.

5.2.7 prstBlobSize

Cette propriété indique la taille de l'objet BLOB associé à la table utilisée pour conserver cet index de service de recherche. Si elle n'est pas spécifiée, la taille de l'objet BLOB est définie par défaut sur 50 Mo. Le type de propriété est une chaîne et la valeur doit être conforme à la syntaxe du spécificateur de taille de la base de données concernée.

5.3 Table SearchServiceField

Chaque zone du service de recherche doit contenir un enregistrement de la table SearchServiceField. Chaque service de recherche représente un élément SearchService qui peut être interrogé, renvoyé à partir d'une recherche, ou les deux. Les zones du service de recherche sont utilisées à plusieurs endroits dans Generic Search Server : dans les termes, les requêtes, les documents. Une description de chaque colonne de la table SearchServiceField est fournie ci-dessous :

5.3.1 srchServiceFldId

L'identificateur unique de la zone du service de recherche.

5.3.2 searchServiceId

L'ID du service de recherche de l'enregistrement du service de recherche parent.

5.3.3 nom

Le nom associé à la zone du service de recherche. Il s'agit du nom qui est utilisé pour référencer la zone lors de la recherche ou de l'extraction de résultats. Il ne doit pas correspondre exactement aux noms de zone des structs et des entités Cúram, bien qu'il simplifie le développement quand c'est le cas.

5.3.4 type

Le type de données Cúram de cette zone. L'ensemble de valeurs acceptables est décrit dans le tableau ci-dessous.

Le processus d'exportation et de synchronisation des données vers le service de recherche implique certaines conversions des données opérationnelles en chaîne, et vice-versa. Par conséquent, il est important qu'un type de données précis soit défini pour chaque zone. Reportez-vous à la table suivante pour en savoir plus à ce sujet. Si des valeurs incorrectes lui sont présentées, Generic Search Server génère une exception.

Tableau 2. Mappages des définitions de domaine Cúram de base vers les types de données de zone de GSS

Définition de domaine	Type de données de zone de GSS
SRV_BOOLEAN	boolean
SRV_DATE	Date
SRV_DATETIME	DateTime
SRV_INT8	byte
SRV_INT16	short
SRV_INT32	int
SRV_INT64	long
SRV_FLOAT	float
SRV_DOUBLE	double
SRV_MONEY	Money
SRV_CHAR	char
SRV_STRING	String
SRV_UNBOUNDED_STRING	String

Remarque : La zone de type est sensible à la casse, assurez-vous donc d'utiliser le nom du type exactement tel qu'il apparaît ci-dessus.

5.3.5 indexed

Indique que cette zone est consultable. Vous souhaiterez peut-être stocker la valeur d'un enregistrement dans le service de recherche sans effectuer une recherche sur cette valeur (par exemple, l'ID unique d'un enregistrement ou, peut-être, son niveau de sensibilité). Ne pas indexer les valeurs qui n'ont pas besoin de l'être minimise la taille de l'index et améliore les performances. Ainsi, il est recommandé de n'indexer que les zones que vos recherches utiliseront.

5.3.6 stockées

Indique si cette zone doit être renvoyée dans un résultat de recherche ou pas, c'est-à-dire, si la valeur elle-même est stockée dans l'index. Remarquez que les zones stockées sont uniquement renvoyées lorsque l'objet de requête transmis à Generic Search Server indique qu'elles doivent être renvoyées. Chaque zone doit être indexée, stockée, ou les deux : si une zone n'est ni indexée, ni stockée, alors elle est inutile pour le service de recherche. A nouveau, ne pas stocker les valeurs que vos recherches n'utilisent pas permet de minimiser la taille de l'index et d'améliorer les performances. Par conséquent, stockez uniquement les zones que vos recherches utiliseront.

5.3.7 entityName

Le nom de l'entité d'application associée avec cette zone (ou pour être plus spécifique, le nom de l'entité d'application contenant un attribut correspondant à cette zone) utilisé pour remplir l'index basé sur la définition du service de recherche parent. Ces informations sont requises pour la synchronisation des données d'application avec Generic Search Server. Toutes les entités répertoriées comme associées aux zones du service de recherche sont enregistrées avec le contrôleur de recherche (voir 3.4, «Contrôleur de recherche», à la page 9) et surveillées pour les insertions, les mises à jour et les suppressions. Il est essentiel que l'attribut entityName soit rempli avec les valeurs appropriées. Les attributs entityName omis ou non valides peuvent se traduire par des mises à jour de l'index non valides.

5.3.8 untokenized

Cette propriété indique si une zone doit être segmentée et transmise à l'analyseur ou pas. C'est une valeur booléenne. Si elle est définie sur "true", aucune segmentation n'est réalisée et l'analyse n'est pas effectuée sur cette zone avant l'indexation ou pendant la recherche.

5.3.9 analyzerName

Cette propriété indique l'analyseur à utiliser lors de la segmentation de cette zone. Le contenu de cette zone peut être défini comme LUCENESTANDARD, STANDARD, SIMPLE, STOP, WHITESPACE, KEYBOARD. (Voir analyseur dans 5.2, «Table SearchService», à la page 13) Si cette zone n'est pas définie, l'analyseur utilisé par défaut sera celui qui figure dans la zone "analyseur" du service de recherche associé.

Chapitre 6. Guide d'initiation à l'interface de programme d'application de Generic Search Server

6.1 Introduction

Ce chapitre ne constitue pas une description complète de la totalité de l'interface de programme d'application de Generic Search Server : de nombreux Javadocs sont disponibles avec l'installation. Le but de ce chapitre est de fournir une brève introduction aux classes et aux opérations les plus importantes de l'interface de programme d'application afin de permettre de générer rapidement des recherches avec Generic Search Server.

6.2 Associateurs

Les associateurs sont des classes qui définissent la manière dont les données du service de recherche sont mappées des tables de la base de données d'application aux tables de la base de données de transfert. Chaque service de recherche dispose de son propre associateur (l'associateur à utiliser est spécifié dans la table de base de données du service de recherche). Pour plus de détails, voir 5.2.5, «mapperName», à la page 14.

Deux processus utilise cette fonctionnalité d'associateur :

1. Lorsque l'extracteur de base de données est en cours d'exécution, chaque zone du service de recherche est itérée pour un service de recherche particulier. Pour chaque zone, les données d'attribut d'entité correspondantes sont récupérées dans la base de données d'application, puis utilisées pour remplir la table de base de données de transfert SearchServiceRow.
2. Lorsqu'une opération de création, de mise à jour ou de suppression est appelée pour une entité utilisée dans un service de recherche, les lignes SearchServiceRow concernées sont mises à jour avec les modifications d'entité associées.

Dans ces deux processus, l'associateur approprié pour chaque service de recherche est appelé pour mapper des données à partir des tables de base de données d'application aux tables de base de données de transfert.

Lors de l'initialisation de Generic Search Server, les informations de la base de données de transfert sont lues et utilisées pour générer les index à partir des métadonnées du service de recherche. Le serveur de recherche vérifie périodiquement s'il existe des mises à jour pour la base de données de transfert, et il maintient les données du service à jour.

Les méthodes suivantes de l'interface de programme d'application de l'associateur nécessitent d'être implémentées par les développeurs de recherche sur chaque service de recherche :

```
SearchServiceRowDtlsList mapToStagingDb(
  final SearchServiceKey id) throws AppException,
    InformationalException;

List getObjectList(final SearchServiceKey serviceId,
  final Object obj) throws AppException, InformationalException;

String getExtKey(final SearchServiceKey serviceId, List objList);

void remove(final SearchServiceKey serviceId, final Object objKey)
    throws AppException, InformationalException;

Object getFieldValue(final SearchServiceKey serviceId,
    final List objList, final SearchServiceFieldDtls field);
```

Pour plus de détails, voir 7.4, «Implémentation d'opérations de l'associateur», à la page 24

6.3 Contrôleur de recherche

Le contrôleur de recherche est un objet singleton pouvant être utilisé dans l'application. Il contrôle quelles entités sont référencées dans quels services de recherche. En outre, il fournit une interface de programme d'application pour les changements de synchronisation effectués sur les données d'application avec les index concernés dans Generic Search Server. Remarquez que, d'une perspective client-serveur, le contrôleur de recherche réside sur le "client" (dans ce cas, le serveur d'application Cúram), et pas sur le "Serveur" (dans ce cas, Generic Search Server).

L'interface de programme d'application du contrôleur de recherche se compose de trois méthodes qui peuvent être appelées si une entité impliquée dans le remplissage d'un index est modifiée. Le développeur de recherche doit savoir quelles opérations d'entité résulteront de telles modifications, et il doit appeler les méthodes appropriées dans le contrôleur de recherche. Les méthodes exposées dans cette interface de programme d'application sont :

```
void SearchController.insert(final Object objectDtls,
  String entityName);
void SearchController.modify(final Object objectDtls,
  String entityName)
void SearchController.remove(final Object objKey, final String entityName);
```

Pour plus de détails, voir 7.6, «Ajout de la synchronisation à chaque entité de recherche», à la page 27

6.4 Connecteur du service de recherche

SearchServiceConnector est une classe utilitaire permettant d'effectuer des recherches. L'opération "search" dans cette classe est la seule manière prise en charge permettant aux développeurs de recherche d'appeler une recherche dans un index Generic Search Server.

De plus, cette classe gère les détails de connexion à partir d'une application en cours d'exécution vers une instance de Generic Search Server, où qu'elle soit déployée.

Cette méthode permet d'effectuer des recherches avec SearchServiceConnector :

```
static SearchServerResults search(CuramQuery query)
```

Remarque : Si l'index de recherche ne contient pas de données, il génère une exception IndexEmptyException. Les développeurs mettant en oeuvre des recherches doivent traiter cette exception avec attention.

Les données d'identification de l'utilisateur sont nécessaires pour la connexion à Generic Search Server. Le connecteur sélectionne les détails de l'utilisateur en cours et les utilise pour communiquer avec Generic Search Server.

Remarque : N'essayez pas d'utiliser la méthode DoSearch (ou une autre méthode Generic Search Server) directement, cela ne fonctionnera pas parce qu'elle s'exécute dans le cadre d'une application Cúram et non dans le cadre d'une application Generic Search Server.

6.5 Requêtes

Pour effectuer une recherche, un objet CuramQuery doit être généré. La classe CuramQuery se compose de :

- L'ID du service de recherche dont vous souhaitez utiliser l'index. Voir 2.4, «Service de recherche», à la page 4 pour plus d'informations sur le concept de services de recherche, et 5.2.1, «searchServiceId», à la page 13 pour obtenir plus de détails sur la définition de l'ID du service de recherche.

- Une liste des objets CuramTerm ou un attribut de texte représentant une chaîne de requête Lucene (qui représente les critères de recherche). Voir ci-dessous pour obtenir plus d'informations sur les termes Cúram et sur l'attribut de texte.
- Une liste des objets CuramField. Les valeurs de ces zones sont renvoyées dans les résultats de la recherche, mais uniquement si les zones ont été marquées comme "Stored" (stockées) dans la définition SearchServiceField (voir 5.3.6, «stockées», à la page 15)
- Un attribut entier maxHits indiquant le nombre maximal de résultats à renvoyer pour cette requête.
- Un indicateur booléen maxHitsUnbounded indiquant que le nombre maximal de résultats n'est pas limité. Si cet indicateur est défini, la valeur de l'attribut maxHits est ignorée.

6.6 Objet CuramTerm

Les objets CuramTerms sont la partie de la structure de CuramQuery qui représente les critères de recherche.

Il existe trois types de termes : les termes StandardTerm, DateTerm ou DateRange. Un objet CuramTerm contient une fois chacun de ces types, et dispose d'un attribut TermType qui spécifie les sous-types de terme qui doivent être utilisés. Seul un de ces sous-types de terme agrégé est valide pour chaque objet CuramTerm.

Pour tous les types de terme, l'attribut "field" spécifie le nom de la zone dans le service de recherche où est effectuée la recherche (voir 2.5, «Zone», à la page 5 et 5.3.3, «nom», à la page 15). L'attribut "value" correspond au critère de recherche à utiliser. Sa signification varie pour les différents types de terme. Elle est décrite ci-dessous.

6.6.1 Structure de requête

Chaque terme possède une zone appelée *se produit*. La manière dont elle est définie détermine la structure de la requête (si tous les termes de la recherche doivent exister, si un seul suffit ou si une autre combinaison est acceptable). Les valeurs possibles pour *se produit* sont `MUST`, `SHOULD`, `MUST_NOT` et `MUST_FIELD`.

Si `MUST` est spécifié pour l'attribut *se produit* pour une série de termes, alors un résultat sera uniquement renvoyé si tous les termes sont trouvés. Si `SHOULD` est spécifié pour une série de termes, alors un résultat sera renvoyé si un ou plusieurs termes sont trouvés. Toutefois, mélanger ces derniers dans une requête simple produit un résultat non défini et doit être évité. Si vous devez générer des requêtes complexes avec des sous-requêtes AND et OR, utilisez l'attribut de requête *texte* décrit dans 6.6.4, «Texte», à la page 20.

Si `MUST_NOT` est spécifié pour l'attribut *se produit*, alors seuls les documents qui ne correspondent pas au terme seront renvoyés. Les termes spécifiant cette valeur peuvent être mélangés avec des termes spécifiant d'autres valeurs pour l'attribut *se produit*.

L'utilisation de l'option `MUST_FIELD` permet de générer une sous-requête vérifiant une zone d'index spécifique pour une valeur d'une série, c'est-à-dire, une sous-requête OR dans votre requête principale. Vous devez la définir comme la valeur *se produit* pour tous les termes liés à cette zone, et ajouter un terme pour chaque valeur acceptable. Les termes utilisant `MUST_FIELD` peuvent faire partie d'une requête générale utilisant les options de terme `MUST` ou `SHOULD`.

6.6.2 Termes standard

Un terme standard est utilisé pour toutes les recherches qui n'impliquent pas de dates. Par conséquent il s'agit du type de terme le plus utilisé.

La manière la plus simple d'utiliser un terme standard est de spécifier le nom de zone et une chaîne de caractères unique comme valeur. Le serveur de recherche renvoie des résultats où la valeur de zone correspond exactement au terme recherché.

Une autre manière d'utiliser un terme standard est de spécifier une valeur qui contient plusieurs chaînes de caractères, comme une adresse. A nouveau, le serveur de recherche renvoie des résultats où la valeur de zone correspond exactement au terme recherché.

Si le terme recherché est une chaîne de caractères unique contenant un caractère générique, alors le serveur de recherche renvoie tous les résultats correspondants. Les caractères génériques pris en charge sont "*", qui correspond à n'importe quelle chaîne de caractères, et "?", qui correspond à un caractère unique. Par exemple : term = "Dub*"

Un terme standard peut être géré comme une recherche avec préfixe. Cela signifie que nous recherchons des résultats de recherche contenant les critères de recherche du début. Pour spécifier une recherche avec préfixe, définissez l'attribut isPrefixSearch du terme standard. L'effet est identique à la spécification d'un caractère générique "*" à la fin de la valeur de votre recherche. Une recherche avec préfixe ne peut contenir aucun autre caractère générique.

Exemple 1 : pour un terme segmenté standard dont le préfixe est "abc", la recherche sous-jacente s'effectue pour le terme = "abc*", pour les recherches de plusieurs termes préfixés et segmentés, par exemple, un terme segmenté recherché "abc def", la recherche sous-jacente s'effectue pour le terme = "abc* def*"

Exemple 2 : pour un terme segmenté standard sans préfixe et commençant par abc, la valeur du terme = "abc*" doit être spécifiée. Pour les recherches de plusieurs termes segmentés sans préfixe commençant par "abc" et "def", la valeur "abc* def*" doit être spécifiée.

6.6.3 Termes de date et de plage de dates

Un terme de date est semblable à un terme standard, sauf qu'il est utilisé pour la recherche de zones dont le type est "Date" ou "Plage de dates".

Un terme de plage de dates permet de rechercher des valeurs comprises entre une date minimale (date de début) et une date maximale (date de fin). L'attribut booléen "isExclusive" détermine si les dates de début et de fin sont inclues dans les critères de recherche. Si "isExclusive" est défini sur "true", la recherche s'effectue en excluant les dates de début et de fin. Si "isExclusive" est défini sur "false", la recherche s'effectue en incluant les dates de début et de fin.

Remarque : Lorsqu'une requête contient plus d'un terme, les résultats renvoyés sont ceux qui correspondent à tous les termes recherchés. Actuellement, il n'existe pas de concept de "OR", ni de "NOT" dans l'interface de programme d'application de Generic Search Server.

Remarque : Lorsque vous utilisez des dates dans vos recherches, gardez à l'esprit qu'il vous incombe de vérifier que la date dans le terme de votre recherche indique le même fuseau horaire que celui utilisé lors de l'exportation des données dans le service de recherche.

6.6.4 Texte

L'attribut de texte de la classe CuramQuery constitue une alternative à de nombreux termes et permet davantage de flexibilité lors de la spécification de vos critères de recherche. Ne l'utilisez que lorsque c'est nécessaire car cette approche peut également introduire des bogues dans vos recherches. Le format de spécification des critères de recherche avec cet attribut est décrit dans la documentation Lucene. Cette documentation est disponible à l'adresse suivante : http://lucene.apache.org/java/2_2_0/queryparsersyntax.html.

Vous ne pouvez pas combiner les termes et l'utilisation de la chaîne de requête de texte. S'il existe une chaîne de requête de texte, alors tous les objets CuramTerms présents dans la requête sont ignorés.

6.7 Génération de requêtes

L'interface de programme d'application de Generic Search Server contient une classe utilitaire conçue pour vous permettre de générer facilement des objets CuramQuery. Cette classe est : `curam.core.impl.util.QueryBuilder`.

6.7.1 Génération d'une instance de Query Builder

`QueryBuilder` n'est pas une classe statique, vous devez générer une nouvelle instance de `QueryBuilder` pour chaque requête effectuée.

Utilisez les méthodes `setUnbounded(boolean unbounded)` et `setMaxHits(long maxHits)` pour spécifier le nombre de résultats renvoyés par la requête générée.

6.7.2 Ajout de critères de recherche

`QueryBuilder` fournit une sélection de méthodes de forme `addXXTerm(...parameters...)` permettant d'ajouter facilement plusieurs types de terme de recherche à la requête générée. Ces termes sont joints par l'opérateur AND pour former une requête complexe. Ces méthodes ne sont pas intégralement décrites ici mais vous trouverez la totalité des détails dans le Javadoc GSS.

6.7.3 Génération de requêtes à partir d'une struct

Si vous avez une struct Cúram que vous souhaitez utiliser pour générer une requête, vous pouvez ce faire avec la méthode suivante : `setTerms(final Object key)`.

Elle suppose une struct où à chaque attribut *XX* correspond un attribut booléen appelé *searchByXX* qui spécifie si cet attribut doit être utilisé pour la recherche. Chaque attribut *XX* est censé correspondre à une zone de service de recherche dans votre service de recherche.

Si les noms des attributs de votre struct ne correspondent pas aux noms des zones définies pour votre service de recherche (voir 2.5, «Zone», à la page 5 et 5.3.3, «nom», à la page 15), alors vous pouvez définir un mappage entre eux à l'aide d'une mappe de hachage de dictionnaire. Le mappage s'effectue des noms d'attribut dans la struct aux noms SearchServiceField. Ajoutez simplement les paires de chaînes à la mappe de hachage, avec le nom de l'attribut de la struct comme clé et le nom de la zone comme valeur. Le dictionnaire peut être spécifié dans le constructeur lorsque vous créez votre objet QueryBuilder, ou ultérieurement à l'aide de la méthode `setDictionary(HashMap<String, String>)`.

6.7.4 Spécification des zones du service de recherche à renvoyer

Dans votre requête, vous pouvez spécifier le sous-ensemble des zones du service de recherche que vous souhaitez renvoyer comme résultats. Vous voudrez souvent qu'elles soient toutes renvoyées, par conséquent, vous pouvez utiliser les méthodes de simplification suivantes :

- `includeAllFieldsInService()`
- `excludeField(String fieldName)`
- `excludeFields(String[] fieldNames)`

6.7.5 Obtention de l'objet de requête

Utilisez la méthode `getQuery()` pour obtenir l'objet CuramQuery généré.

6.8 Gestion des résultats de recherche

Tout comme il est nécessaire de convertir les structs clés Cúram en objets `CuramQuery`, les documents `CuramDocument` renvoyés par les recherches doivent également être convertis en structs Cúram pour être utilisés dans l'application.

La méthode de recherche `SearchServiceConnector` renvoie les résultats sous forme d'un objet `SearchServerResults`. Il s'agit d'une liste de documents `CuramDocument`, chaque document `CuramDocument` contenant une liste de zones `CuramField`. Une classe utilitaire appelée `curam.core.impl.util.CuramDocToResultStruct` est fournie pour la conversion des documents `CuramDocuments` et des structs Cúram.

```
static java.lang.Object convert(CuramDocument document,
     java.lang.Object structObj,
     java.util.HashMap dictionary)
```

Cette méthode nécessite un document `CuramDocument` et une instance de struct (via le paramètre structObj). Pour chaque zone du document `CuramDocument`, la méthode essaye de trouver un attribut dans la struct dont le nom et le type de données sont les mêmes. Une struct contenant toutes les valeurs mappées est renvoyée, elle doit être transtypée vers une struct du bon type.

Si les noms des attributs de votre struct ne correspondent pas aux noms des zones définis pour votre service de recherche (voir 2.5, «Zone», à la page 5 et 5.3.3, «nom», à la page 15), alors vous pouvez définir un mappage entre eux à l'aide du paramètre de dictionnaire. Le mappage s'effectue des noms de zone dans le service de recherche aux noms d'attribut dans la struct. Ajoutez simplement les paires de chaînes à la mappe de hachage, avec le nom de la zone comme clé et le nom de l'attribut de struct comme valeur. La fonction de conversion fait ensuite correspondre les noms de zone aux noms d'attribut à l'aide de cette mappe de hachage.

Remarque : Remarquez que les attributs de votre struct de résultats dont les noms correspondent aux zones dans votre document doivent contenir des types Cúram simples, et ne pas être des structs agrégées.

6.9 Type de données et conversion de chaîne

Generic Search Server dispose d'une interface de programme d'application permettant de convertir les types de données Cúram interrogeables en chaînes, et vice-versa. Ces derniers peuvent parfois être utilisés dans des associateurs personnalisés ou directement dans des résultats d'analyse syntaxique à la place de la classe utilitaire fournie `curam.core.impl.util.CuramDocToResultStruct`.

La classe de convertisseur est `curam.core.impl.search.datatypes.DataTypeConverter`. Cette classe contient des méthodes permettant de convertir les types de données Cúram en chaînes, et de convertir des chaînes en types de données Cúram (en les transmettant à une struct et en spécifiant l'attribut de la struct à définir).

Chapitre 7. Implémentation d'une recherche avec Generic Search Server

7.1 Présentation

Ce chapitre fournit un exemple concret d'implémentation d'une recherche Generic Search Server dans l'application Cúram. L'exemple montré ici est une recherche de personne.

Les étapes de l'implémentation sont les suivantes :

- Ecrire les fichiers DMX SearchService et SearchServiceField
- Implémenter l'interface de l'associateur
- Implémenter le routage de recherche et la fonctionnalité d'appel
- Ajouter la synchronisation des opérations d'application vers les entités de recherche (ou utiliser l'approche de l'associateur d'extraction, voir Chapitre 8, «Associateur d'extraction», à la page 29
- Créer une interface utilisateur et une façade pour la recherche (développement d'application normal).

7.2 Exemple de recherche de personne : présentation

Il est important de remarquer que les utilisateurs de Cúram Generic Search Server ne devraient constater aucune différence de fonctionnement entre leurs recherches et les recherches de serveur implémentées avec SQL. En outre, les écrans et l'expérience utilisateur globale restent les mêmes. En tant que tel, l'exemple suivant suppose que ses lecteurs développeront une telle fonctionnalité d'application (avec les classes façades appropriées, etc.) normalement.

Dans notre exemple de recherche de personne, les utilisateurs naviguent vers la page du gestionnaire d'interface utilisateur approprié afin d'effectuer une recherche de personne. Sur cette page, ils remplissent un ou plusieurs critères de recherche. Lorsqu'ils appuient sur le bouton "Search" (Rechercher), la recherche est effectuée. Les résultats contiennent une liste des enregistrements correspondant aux critères de recherche.

Dans les recherches d'application, il est courant que les critères de recherche et les détails renvoyés dans la liste des résultats soient assemblés à partir de plusieurs entités associées. Pour la recherche de personne, les entités suivantes et leurs attributs sont utilisés comme critères de recherche ou renvoyés comme zones de résultat :

- Person - primaryAlternateID, personBirthName, motherBirthSurname, dateOfBirth, gender
- ConcernRole - sensitivity, concernRoleID
- AlternateName - firstForeName, surname
- AddressElement - city, address.

Chacune de ces entités est associée par une association de clé étrangère : concernRoleID est donc la clé externe de l'attribut SearchService pour le service de recherche PersonSearch (voir 5.2, «Table SearchService», à la page 13)

Les attributs suivants sont donc utilisés dans la recherche, soit en tant que critères de recherche, soit en tant qu'élément d'affichage de la liste des résultats :

- referenceNumber
- forename
- surname
- address

- city
- dateOfBirth
- sex
- birthSurname
- motherSurname

En tant que tel, il s'agit des zones stockées dans la table SearchServiceField pour le service de recherche PersonSearch.

7.3 Développement de fichiers DMX SearchService

7.3.1 Configuration de l'enregistrement du service de recherche

Reportez-vous à B.1, «Enregistrement de service de recherche», à la page 49 et à 5.2, «Table SearchService», à la page 13

7.3.2 Configuration de l'enregistrement SearchServiceField

Reportez-vous à B.2, «Enregistrement de zone de service de recherche», à la page 50 et à 5.3, «Table SearchServiceField», à la page 14

7.4 Implémentation d'opérations de l'associateur

Voir 2.12, «Associateur», à la page 6 et 6.2, «Associateurs», à la page 17 pour une introduction aux associateurs.

Les sections suivantes décrivent l'implémentation des méthodes de l'interface de l'associateur pour chaque service de recherche. Un exemple de service de recherche PersonSearch est fourni pour chaque méthode de l'interface. Un Javadoc complet est également disponible pour l'interface de l'associateur et devrait être lu par tous les développeurs implémentant un service de recherche.

7.4.1 Interface Mapper.mapToStagingDb

```
/**
 * Maps information in the Application database to the search
 * service staging database for the specified search service id.
 *
 * @param id the identifier of the search service.
 * @return the list of all mapped rows for the specified search
 *   service.
 * @throws AppException application exception
 * @throws InformationalException information exception.
 */
SearchServiceRowDtlsList mapToStagingDb(
  final SearchServiceKey id) throws AppException,
    InformationalException;
```

Cette méthode est appelée lors du traitement par lots d'extraction de la base de données : pour chaque service de recherche, mapToStagingDb est appelé pour récupérer des informations à partir des entités source et les renvoyer au traitement par lots.

Une opération Cúram ReadmultiOperation doit être écrite pour traiter tous les enregistrements à stocker dans la base de données de transfert pour chaque service de recherche. Une opération Generic Search Server appelée ExtractReadMultiOperation doit être appelée sur chacun de ces enregistrements. En interne, cette opération détermine quelles autres entités sont nécessaires pour remplir une ligne SearchServiceRow complète basée sur ces données, et génère également un objet SearchServiceRow.

Le résultat de ce processus complet est simplement une liste d'objets SearchServiceRows, constituant toutes les données initiales à renseigner dans la base de données de transfert. Le traitement par lots d'extraction de base de données s'occupe ensuite d'insérer ces lignes dans la base de données de transfert.

7.4.2 Interface Mapper.getObjectList

```
/**
 * Populates the list with all entity objects for the
 * Search Service given any one of the entity objects used.
 * @param searchServiceId. the search service identifier
 * @param obj. The entity object from which all other are
 *   retrieved
 * @return the list of all entity objects for the this search
 *   service given a specified object parameter.
 */
List getObjectList(final SearchServiceKey serviceId,
  final Object obj) throws AppException,
    InformationalException;
```

Comme indiqué précédemment, il est possible que les données d'un service de recherche soient rassemblées à partir de différentes entités. Il est également possible que ces entités soient liées par des relations de clé étrangère complexes (par exemple, un enregistrement d'adresse peut être lié à un enregistrement de personne par une ID addressID, liée par une ID concernRoleAddressID qui est à son tour liée par une ID concernRoleID).

Les choses se compliquent lorsqu'une de ces entités est mise à jour par l'application. Lorsque cela se produit, Generic Search Server doit être en mesure de déterminer les entités affectées, les recherches impliquées et la relation avec les autres entités comprises dans chaque service de recherche.

Finalement, un ou plusieurs documents dans un ou plusieurs index de services de recherche doivent être mis à jour, et les informations contenues dans ces documents peuvent être rassemblées à partir d'une série d'entités, et pas seulement à partir de celle qui vient d'être modifiée. Toutefois, étant donné que les services de recherche disposent d'un seul associateur, chaque implémentation d'associateur nécessite uniquement de se préoccuper des informations d'assemblage pour son propre service de recherche.

La méthode d'interface getObjectList gère ce problème. Pour un enregistrement d'entité unique mis à jour, getObjectList assemble tous les autres enregistrements DTLS d'entité qui sont nécessaires pour mettre à jour le document correspondant dans l'index du service de recherche actuel. La méthode getObjectList doit être codée de telle sorte que les entités impliquées dans le service de recherche peuvent être utilisées comme point de départ de ce processus. getObjectList est responsable de :

- déterminer l'entité transmise
- déterminer toutes les entités associées au service de recherche en question
- lire et assembler tous les enregistrements d'entité associés en fonction des données de l'entité de paramètre

La méthode mapper.getobjectList () est appelée par les processus suivants :

- Database Synchronization insert
- Database Synchronization modify
- Initial Database Extraction

Remarquez que, pour "initial Database Extraction", la méthode d'interface getObjectList est appelée pour chaque élément extrait de ReadmultiOperation : généralement ce sera l'entité de niveau supérieur dans ce cas (par exemple, pour une extraction de recherche de personne, tous les enregistrements de personne sont lus avec readmulti, getObjectList est ensuite appelé pour que chacun récupère toutes les autres informations nécessaires à la création d'une méthode SearchServiceRow).

Si cette méthode est appelée pour une entrée qui ne concerne pas ce service de recherche, alors l'implémentation doit simplement renvoyer une liste vide.

7.4.3 Interface Mapper.getExtKey

```
/**
 * Gets the Row external value for the specified object list.
 * @param searchServiceId. the search service identifier
 * @param objList the list of Search Service related entity
 *   objects.
 * @return the externalKey.
 */
String getExtKey(final SearchServiceKey serviceId, List objList) ;
```

La méthode d'interface `getExtKey` renvoie un identificateur unique pour le service de recherche spécifié. Cette clé est utilisée en tant que clé pour chaque ligne de la table `SearchServiceRow` dans la base de données de transfert. Remarquez que le paramètre objList est le résultat de la méthode d'interface getObjectList décrite ci-dessus. Par exemple, `calling getExtKey` pour le service de recherche PersonSearch devrait renvoyer l'ID concernRoleID de l'enregistrement en question.

Si cette méthode est appelée pour des données dont le service de recherche ne s'occupe pas, alors elle renvoie une valeur nulle.

7.4.4 Interface Mapper.remove

```
/**
 * Deletes the row identified by the specified key from the
 *   staging
 * database.
 * @param serviceId identifier of the service.
 * @param objKey the Key.
 * @throws AppException
 * @throws InformationalException
*/
void remove(SearchServiceKey serviceId, Object objKey)
  throws AppException, InformationalException;
```

Supprime l'objet de ligne spécifié de la base de données de transfert.

7.4.5 Interface Mapper.getFieldValue

```
/**
 * If a specialized field value can't be covered by the
 * <code>SearchServiceMapper.getValue()
 * <code> functionality this method
 * should be overridden in the mapper for the specific search
 *   service.
 * @param objList list of entity objects for this specific
 *   mappers service id.
 * @param field the field whose value is required.
 */
Object getFieldValue(final SearchServiceKey serviceId,
   final List objList, final SearchServiceFieldDtls fieldDtls);
```

L'infrastructure Generic Search Server essaie de récupérer une valeur d'attribut d'entité à partir d'une liste d'objets à l'aide des métadonnées de zone récupérées depuis la table des zones du service de recherche. Généralement, les listes objectLists contiennent des structs DTLS d'entité. Dans ce cas, il est simple pour Generic Search Server d'utiliser le reflet afin d'identifier l'attribut correct et d'obtenir sa valeur (c'est exactement ce qui se passe en arrière-plan).

Toutefois, si la liste objectList contient autre chose que la struct DTLS d'entité (comme c'est le cas d'une recherche de personne, où une liste AddressElementDtlsList est présente, elle-même contenant une struct AddressElement unique), alors la méthode d'interface Mapper.getFieldValue doit être implémentée par les développeurs de la recherche.

La méthode d'interface Mapper.getFieldValue doit être implémentée lorsqu'un associateur ne peut automatiquement mapper une valeur d'attribut spécifique. L'entité concernée et le nom de zone sont transmis par le paramètre de struct fieldDtls, et la valeur d'attribut peut être récupérée à partir de la liste objList à l'aide du reflet. Il incombe au développeur de recherche d'implémenter cette interface de méthode pour le ou les types concernés.

Les chaînes vides ne doivent pas être renvoyées à partir de cette méthode (la valeur nulle doit toujours être renvoyée).

7.4.6 Mapper newInstance()

Si l'associateur est modelé, alors la classe de fabrique doit être spécifiée pour la propriété mapperName du service de recherche. Si l'associateur n'est PAS modelé, alors l'implémentation d'associateur doit implémenter une interface

```
public static Mapper newInstance();
```

interface renvoyant une nouvelle instance de cet associateur de service de recherche. Dans ce cas, la propriété SearchService mapperName correspondra au nom de classe de cette classe d'implémentation.

7.5 Routeur de recherche et implémentation

Comme indiqué précédemment, la recherche utilise actuellement le langage SQL. Dans les versions ultérieures, il est probable que les recherches de platforme et de solution commencent à utiliser Generic Search Server comme méthode de recherche. Toutefois, il est probable que la recherche SQL continuera également à être prise en charge en l'état actuel, à la fois d'un point de vue de la protection des mises à niveau et d'un point de vue de l'option de reprise après incident/de rétromigration en cas de rétromigration du réseau ou d'autres problèmes de déploiement.

Pour faciliter cette opération, une classe de fabrique du routeur de recherche doit être implémentée. Elle doit renvoyer une référence à l'implémentation de recherche de la base de données ou à l'implémentation Generic Search Server en fonction d'un paramètre de propriété.

7.6 Ajout de la synchronisation à chaque entité de recherche

Comme indiqué précédemment, la base de données de transfert de Generic Search Server doit être mise à jour régulièrement lorsque des modifications sont apportées aux entités associées au service de recherche. Une entité unique peut très bien être utilisée dans plus d'un service de recherche, et chacun de ces services de recherche doit refléter les changements apportés à cette entité.

La classe *SearchController* est responsable de vérifier que toutes les informations de la base de données de transfert sont à jour. Les méthodes *SearchController insert*, *modify* et *remove* doivent être appelées à partir de l'application lorsque l'opération d'entité du service de recherche correspondant est exécutée. Les opérations SearchController *insert* et *modify* modifient les informations de la table SearchServiceRow avec les données de struct des détails d'entité spécifiés. L'interface remove nécessite une clé qui identifie l'objet d'entité en cours de suppression, ainsi que le nom de l'entité.

```
/**
 * Insertion générique de mises à jour d'entité dans la base de données.
 *
 * @param details the object details.
 * @param entityName the name of the entity
 * @throws AppException application exception retrieving the
 *   registrar
```

```
 * or during Mapper insert.
 * @throws InformationalException information exception.
 */

public final void insert(final Object details,
  final String entityName)
    throws AppException, InformationalException
/**
 * Generic Modify of entity updates to the database.
 *
 * @param details the object details.
 * @param entityName the name of the entity
 * @throws AppException application exception retrieving the
 *   registrar
 * or during Mapper modify.
 * @throws InformationalException information exception.
 */
public final void modify(final Object details,
  final String entityName)
    throws AppException, InformationalException
/**
 * Generic remove of entity from the database.
 *
 * @param key the object key.
 * @param entityName the name of the entity
 * @throws AppException application exception.
 * @throws InformationalException information exception.
 */
public final void remove(final Object key,
  final String entityName) throws AppException,
    InformationalException
```

Chapitre 8. Associateur d'extraction

8.1 Introduction

Dans le chapitre précédent, nous avons décrit le mécanisme des événements et son utilisation pour la synchronisation de vos données avec le service de recherche. Generic Search Server fournit à présent une autre manière de garder votre service de recherche à jour, il s'agit de l'associateur d'extraction. Ce chapitre décrit le fonctionnement de l'associateur d'extraction et son utilisation avec les nouvelles recherches que vous développez.

8.2 Présentation de l'associateur d'extraction

Le mécanisme d'événement est de loin la méthode la plus efficace pour maintenir vos services de recherche à jour. Toutefois, si vos recherches sont complexes, développer et tester votre service de recherche peut s'avérer fastidieux. C'est le problème que l'associateur d'extraction permet de résoudre.

L'associateur d'extraction utilise les horodatages sur les enregistrements d'application pour rechercher les enregistrements qui ont été créés ou mis à jour depuis la dernière exécution de l'associateur d'extraction ou de l'extracteur. Lorsqu'il identifie de tels enregistrements, il les transmet au contrôleur de recherche pour mettre à jour les services de recherche, ensuite le processus est exactement le même que le mécanisme d'événement standard. Ce processus nécessite que toutes les tables de base de données impliquées dans un service de recherche soient analysées, ce qui nécessite évidemment des ressources de base de données. En substance, l'associateur d'extraction sacrifie des performances d'exécution pour fournir une manière plus rapide et facile de développer des recherches.

8.3 Développement avec l'associateur d'extraction

Cette section vous guide pas à pas dans le processus de développement d'un service de recherche à l'aide de l'associateur d'extraction.

8.3.1 Activation de Last Updated Field (Dernière zone mise à jour) dans vos entités interrogeables

Les horodatages sont requis sur toutes les entités de base de données impliquées dans les services de recherche et utilisant l'associateur d'extraction. Ces colonnes d'horodatage sont automatiquement ajoutées et maintenues à jour par l'infrastructure lorsque vous activez l'option Last Updated Field (Dernière zone mise à jour) pour l'entité dans le modèle. Le processus d'activation de cette fonction est décrit dans le manuel `Server Modelling Guide`.

8.3.2 Modélisation de l'analyse de table

Une autre exigence de modélisation imposée par l'associateur d'extraction pour modeler une opération s'appelle `searchByLastwritten` (utilisez exactement cette casse/orthographe).

Cette opération doit être une opération nsmulti. La valeur pour un code SQL non généré doit être "no". L'opération doit prendre une struct appelée *clé*. Vous devez modeler votre propre struct en tant que paramètre, mais elle doit avoir un attribut appelé *date-heure*, qui doit être un attribut `DateTime`. Plus tard, vous pourrez spécifier le class-name de cette struct dans la table GSSEntity table, comme décrit ci-dessous.

Vous devez fournir une instruction SQL pour l'opération. Voici un simple exemple pour une entité appelée Customer :

```
Select Customer.customer_id, Customer.name,
      recordStatus from Customer
 WHERE Customer.lastwritten >= :datetime
 INTO :customer_id :name :recordStatus
```

Vérifiez que vous sélectionnez toutes les colonnes utilisées par le service de recherche.

En plus de la méthode d'analyse de table, vous devez avoir une méthode de lecture standard dans toutes vos entités interrogeables.

8.3.3 Définition de votre service de recherche

Votre service de recherche doit être défini normalement (voir Chapitre 7, «Implémentation d'une recherche avec Generic Search Server», à la page 23)

En plus des tables SearchService et SearchServiceField, vous devez ajouter des définitions aux tables GSSMapperType et GSSEntity.

8.3.3.1 GSSMapperType

Cette table mappe simplement le nom du service de recherche à une chaîne définissant le type d'associateur. Sa valeur par défaut est l'associateur d'événement standard, qui n'a pas besoin d'être spécifié. Pour utiliser l'associateur d'extraction avec un service de recherche particulier, une ligne doit être ajoutée à cette table et mapper le nom du service de recherche au type d'associateur «PULL».

searchServiceId

L'identificateur du service de recherche : une chaîne utilisée pour identifier un service de recherche de manière unique. Il s'agit d'une clé étrangère de la table SearchService.

mapperType

Définissez cette propriété sur "PULL" (en majuscules) pour activer l'associateur d'extraction pour le service de recherche.

8.3.3.2 GSSEntity

Lorsque l'associateur d'extraction est utilisé, GSS nécessite plus d'informations sur les entités utilisées dans les services de recherche. Pour chaque entité unique répertoriée dans les enregistrements searchServiceField enfants appartenant à chaque service de recherche utilisant l'associateur d'extraction, un enregistrement GSSEntity doit être ajouté (toutefois, si plusieurs zones appartiennent à la même entité, vous n'avez pas besoin de répéter les informations).

searchServiceId

L'identificateur du service de recherche : une chaîne utilisée pour identifier de manière unique un service de recherche. Il s'agit d'une clé étrangère de la table SearchService.

tblScanKeyStruct

Il s'agit du class-name complet de la struct qui est le paramètre de votre méthode `searchByLastwritten` modélisée décrite ici : 8.3.2, «Modélisation de l'analyse de table», à la page 29.

entityKeyStruct

Il s'agit du class-name de la struct de paramètre vers la méthode de lecture de votre entité.

EntityFactClass

Il s'agit du class-name complet de la classe de fabrique générée pour votre entité.

8.3.4 Ecriture de votre classe d'associateur

Une implémentation SearchServiceMapper avec l'associateur d'extraction est très similaire à une implémentation SearchServiceMapper standard, telle que décrite dans le chapitre relatif à l'implémentation d'une recherche avec GSS du présent manuel. Toutefois, d'autres éléments entrent en considération.

Lors de l'utilisation de l'associateur d'extraction avec un service de recherche complexe composé de plusieurs entités associées, vérifiez que votre implémentation SearchServiceMapper se comporte correctement lorsqu'elle traite des ensembles d'entités incomplets, c'est-à-dire si les entités A, B et C ensemble comprennent un service de recherche que votre associateur peut appeler uniquement lorsque A et C existent. Selon votre service de recherche, le comportement approprié peut être d'ajouter l'ensemble de données incomplet au service de recherche, ou de ne rien faire tant que l'ensemble n'est pas complet.

8.4 Opérations de suppression

L'associateur d'extraction ne peut pas traiter les opérations de suppression standard. Si vous avez une entité interrogeable qui peut être supprimée, alors vous devez utiliser un autre mécanisme pour traiter cette opération (par exemple, le mécanisme basé sur les événements décrit dans ce manuel).

Toutefois, l'associateur d'extraction peut s'occuper des opérations de suppression logique standard, c'est-à-dire, où une colonne recordStatus est définie à l'aide des valeurs de table de codes `RecordStatus`.

Chapitre 9. Recherches et requêtes en profondeur

9.1 Introduction

Comme tous les autres logiciels, vos recherches prise en charge par GSS doivent être conformes à certaines contraintes de conception pour qu'elles s'exécutent correctement et que leur fonctionnement convienne aux utilisateurs. Ce chapitre décrit en profondeur le processus de conception d'une recherche GSS et la bonne utilisation des requêtes GSS.

9.2 Service de recherche - instructions générales

Votre première tâche de conception est de déterminer les données que vous souhaitez pouvoir interroger. Quelles zones voulez-vous pouvoir interroger ? Quelles données souhaitez-vous que votre recherche renvoie ? Les implications de ces décisions sont nombreuses, pensez-y donc soigneusement.

D'abord, votre index doit contenir aussi peu de zones que possible. Moins de zones signifie un plus petit index à exécuter et une utilisation réduite des ressources système. N'intégrez une zone à votre service de recherche si vous n'en avez pas besoin.

Chaque zone de votre index peut être indexée (c'est-à-dire, devenir interrogeable), stockée (ce qui signifie que vous pouvez récupérer sa valeur), ou les deux. Les raisons pour lesquelles vous pourriez souhaiter indexer une zone sont évidentes : vous souhaitez que votre recherche puisse l'utiliser. Toutefois, il existe certaines zones que vous ne souhaitez pas interroger, les ID non lisibles. Vous souhaitez peut-être les ajouter à votre service de recherche comme des zones stockées mais non indexées, de la sorte, vous pouvez effectuer des recherches de base de données basées sur les résultats de vos recherches. Si vous n'avez pas besoin d'indexer une zone, ne le faites pas, votre processus d'extraction s'exécutera plus rapidement et votre index utilisera moins de ressources système.

De même, vous pouvez choisir de stocker des valeurs de zone, ou non. En général, l'index ne stocke pas la valeur d'origine d'une zone, mais il conserve uniquement une représentation interrogeable. En général, pour être utile, une recherche doit stocker au moins une zone (la clé primaire correspondant à l'enregistrement de base de données).

Après cela, le choix de stocker des zones ou non vous appartient. Vous pouvez stocker toutes les zones dont vous avez besoin pour afficher vos résultats de recherche, ou stocker uniquement les ID de base de données, puis les utiliser pour récupérer les données à partir de la base de données à afficher. La première option se traduit par un index plus volumineux, mais par un affichage plus rapide des résultats de la recherche parce que la base de données n'est pas nécessaire.

9.3 Mappage de votre structure de base de données à un index - dénormalisation

Vous souhaitez peut être inclure des données de plusieurs entités différentes dans votre recherche. Contrairement aux recherches de base de données, les recherches avec les index ne sont pas effectuées à l'aide de jointures. Souvenez-vous que le principal avantage d'utiliser un index est de permettre au travail de recherche d'être effectué essentiellement en avance, lorsque l'index est créé plutôt que lorsque la recherche est effectuée. Par conséquent, toutes les tables de base de données doivent être dénormalisées pour l'indexation. L'alternative, qui consiste à créer des index distincts, à les interroger séparément, puis à tenter de fusionner les résultats, est beaucoup plus complexe et moins efficace.

Par exemple, si vous disposez des entités suivantes : une entité Personne avec les attributs Nom, date de naissance et la clé étrangère pointant vers une entité Adresse avec les attributs Adresse postale, Ville et Pays. Vous souhaitez créer une recherche qui vous permet de recherche des personnes par nom, date de naisse, adresse postale, ville et pays. Vous pouvez créer un index interrogeable qui contient toutes les données de ces tables.

Lorsque plusieurs entités contribuent à un index de recherche unique, gardez à l'esprit que toute mise à jour d'une des tables concernées peut nécessiter la mise à jour de l'index de recherche.

9.4 Zones segmentées et non segmentées

Nous avons déjà brièvement évoqué la question de la segmentation des zones de recherche. La segmentation en unités implique essentiellement la séparation de données indexées en unités appelées chaînes de caractères. Cette opération est effectuée par un analyseur. Différents analyseurs se comportent différemment, certains séparent les chaînes de caractères en fonction des blancs, d'autres en fonction de la ponctuation, etc. Les chaînes de caractères produites sont également souvent converties en minuscules. Pour les zones segmentées, les chaînes de requête sont segmentées de la même manière. De la sorte, les recherches ne sont, entre autres avantages, pas sensibles à la casse.

Pour certaines zones, la segmentation n'a aucun intérêt. Les valeurs générées par ordinateur, comme les codes des tables de codes, en sont un bon exemple. Toutefois, la plupart des zones doivent généralement être segmentées. Plus spécifiquement, le comportement de zones et de recherches sans jeton à plusieurs mots n'est pas intuitif. Si vous découvrez que vos recherches ne renvoient pas les données attendues, vérifiez que ce ne soit pas le cas.

Par exemple : prenez une zone adresse, avec un document contenant "Joyce Way Parkwest Dublin". S'il s'agissait d'une zone segmentée avec l'analyseur standard, l'index contiendrait quatre termes : joyce, way, parkwest et dublin. Toute chaîne de requête qui contient des termes correspondant à ces termes (exactement ou par le biais d'un caractère générique) trouvera ce document. Par exemple : "Dublin", "Joyce Way", "park*", etc.

Toutefois, si cette zone n'est pas segmentée et que le même document est ajouté, l'index contiendra un seul terme : "Joyce Way Parkwest Dublin". Beaucoup moins de chaînes de requête y correspondront, essentiellement la chaîne elle-même ou la première partie de la chaîne comme préfixe de recherche. La recherche sera également sensible à la casse.

9.5 Caractères génériques

GSS prend en charge les caractères génériques pour plusieurs caractères ou pour un caractère unique. Le point d'interrogation, «?» représente n'importe quel caractère unique. L'astérisque, «*», correspond à toute séquence de caractères. Ni l'un, ni l'autre ne peut être utilisé comme premier caractère dans un terme de recherche parce que les performances en seraient réduites. Lors de l'implémentation d'une recherche, les développeurs doivent déterminer si les utilisateurs sont autorisés à entrer ces caractères dans les recherches et, si c'est le cas, ils doivent proposer une aide en ligne utile. Sinon, ils peuvent être retirés grâce à un caractère d'échappement : «\». Il peut également être utile de vérifier si ces caractères ne se trouvent pas au début des termes de recherche et ne renvoient pas un message d'erreur plus spécifique à l'utilisateur que ne peut le faire l'infrastructure GSS (une exception générique pour indiquer que la requête n'est pas valide sera renvoyé, mais le développeur implémentant la recherche sera capable d'ajouter plus d'informations concernant la zone non valide).

9.6 Analyseurs en profondeur

Comme indiqué précédemment, les analyseurs préparent votre texte interrogeable pour l'indexation et la recherche.

Le choix de vos analyseurs est très important. Les analyseurs sont des classes concrètes qui étendent la classe org.apache.lucene.analysis.Analyzer. GSS est livré avec plusieurs analyseurs, et vous pouvez créer et utiliser les vôtres. Parfois, lorsque vous êtes tentés de définir une zone comme non segmentée, vous préférerez peut-être plus attentivement les choix d'analyseurs dont vous disposez.

Chaque service de recherche dispose d'un analyseur par défaut, et chaque zone de service de recherche peut remplacer cet analyseur pour définir un analyseur spécifique à utiliser avec cette zone (voir 5.3.9, «analyzerName», à la page 16). GSS utilisera le même analyseur pour l'indexation et les recherches.

Generic Search Server fournit les analyseurs prédéfinis suivantes.

LUCENESTANDARD
: Sépare le texte en fonction des caractères de ponctuation, en supprimant la ponctuation. Toutefois, un point qui n'est pas suivi d'un blanc n'est pas considéré comme faisant partie d'une chaîne de caractères. Sépare les mots en fonction des traits d'union, sauf si un nombre se trouve dans la chaîne de caractères, auquel cas la chaîne de caractères entière est interprétée comme un numéro de produit et n'est pas séparée. Reconnaît les adresses de courrier électronique et les noms d'hôte Internet comme une chaîne de caractères. Normalise le texte de la chaîne de caractères en minuscule et retire les mots vides anglais courants.

STANDARD
: Semblable à l'analyseur LUCENESTANDARD mais les mots vides courants sont supprimés des termes segmentés et si le contenu à segmenter est un nombre unique, il ne sera pas modifié (ce qui le rend plus adapté pour les ID d'infrastructure générées par le traitement, qui peuvent être des nombres négatifs).

SIMPLE
: Sépare le texte en fonction des caractères qui ne sont pas des lettres, et normalise le texte de la chaîne de caractères en minuscules.

STOP Sépare le texte en fonction des caractères qui ne sont pas des lettres, normalise le texte de la chaîne de caractères en minuscules, puis supprime les mots vides anglais courants.

WHITESPACE
: Sépare le texte en fonctions des blancs. Les séquences adjacentes des caractères non blancs forment des chaînes de caractères.

KEYWORD
: "Segmente" le flux entier comme une chaîne de caractères unique. Cet analyseur est utile pour les données comme les codes postaux, les ID et les noms de produit.

Remarquez que lorsque vous utilisez un analyseur autre qu'un analyseur GSS prédéfini ou que les analyseurs livrés avec Lucene, la classe doit être disponible dans le chemin d'accès aux classes Generic Search Server.

Chapitre 10. Utilisation de Generic Search Server dans Eclipse

10.1 Introduction

Ce chapitre décrit la configuration de l'environnement de développement pour l'exécution de Generic Search Server dans Eclipse IDE à des fins de test et de développement.

Generic Search Server peut être exécuté en mode RMI à des fins de développement, de manière similaire à l'application Cúram. Ce chapitre détaille la configuration pour ce faire.

10.2 Bootstrap.properties

Avant de commencer le développement, les paramètres pertinents doivent être ajoutés à votre fichier `Bootstrap.properties`, si nécessaire. Voir A.1, «Propriétés de configuration», à la page 47 pour obtenir une description des propriétés de configuration.

10.3 Lancement de Cúram Generic Search Server à partir d'Eclipse

A l'instar de l'application Cúram, en mode développement, Generic Search Server nécessite un processus tnameserv à exécuter sur votre machine.

Dans votre installation en mode développement, les fichiers de classe Java d'implémentation suivants seront accessibles dans votre fichier `/CuramSDEJ/lib/gss.jar` sous Eclipse :

- `curam.core.impl.DataBaseSearchExtractor.class`
- `curam.core.impl.admin.StartSearchServer.class`

Vous devriez être en mesure d'exécuter les deux fichiers de classe ci-dessus comme une application Java normale dans le cadre du projet EJBServer, de la manière habituelle, c'est-à-dire

- Cliquez avec le bouton droit de la souris sur le fichier, puis sélectionnez **Run as Java application** (Exécuter en tant qu'application Java)

Exécutez DataBaseSearchExtractor pour générer votre base de données de transfert avant StartSearchServer. Exécutez ensuite le processus StartSearchServer lorsque vous avez besoin d'exécuter une instance du serveur de recherche pour tester votre fonctionnalité de recherche. Vous devez à nouveau exécuter votre processus DataBaseSearchExtractor avant de démarrer votre serveur de recherche si vous avez généré votre base de données d'application.

Remarque : Si certains de vos services de recherche utilisent des analyseurs personnalisés ou tiers (c'est-à-dire des analyseurs qui ne font pas partie de la distribution Lucene), assurez-vous qu'ils ont été ajoutés au chemin d'accès aux classes du projet `EJBServer`.

Chapitre 11. Déploiement de Generic Search Server

11.1 Introduction

Ce chapitre décrit le processus de déploiement de Cúram Generic Search Server sur votre serveur d'application. Ce chapitre s'adresse aux administrateurs qui vont déployer le serveur de recherche avec l'application Cúram, et qui sont familiers avec le guide de déploiement Cúram approprié.

11.2 Options de déploiement

Vous pouvez déployer GSS dans son propre fichier `ear` avec le fichier `ear` de Cúram. `EJBServer/project/config/deployment_packaging.xml` contient une option permettant d'inclure GSS, appelée `requireSearchServer`. Si vous avez défini et généré votre fichier `ear` Cúram, alors vous n'avez pas besoin de déployer GSS en tant que fichier `ear` distinct (en fait, votre serveur d'application ne le permettra pas). En général, nous ne le recommandons pas, car il s'agit d'une configuration de déploiement moins performante, mais elle peut servir dans le cadre de tests ou de petits déploiements.

11.3 Processus de déploiement

Le processus de déploiement se compose des étapes suivantes :

- Définissez votre propriété Bootstrap.properties avec vos propriétés de configuration, puis toutes les propriétés associées à votre serveur de recherche. Voir A.1, «Propriétés de configuration», à la page 47 pour obtenir une description des propriétés de configuration.
- Générez votre fichier ear d'application Cúram normalement (cela génère également votre fichier ear GSS).
- Configurez votre base de données normalement.
- Exécutez l'extracteur de base de données de recherche Cúram Generic Search Server.
- Déployez tous vos fichiers ear d'application, y compris SearchServer.ear
- Connectez-vous à l'application en tant qu'administrateur, puis configurez les propriétés du système pour activer les recherches prises en charge par GSS que vous souhaitez utiliser, et activez le mécanisme de synchronisation. Voir Chapitre 4, «Recherches prises en charge par Generic Search Server», à la page 11
- Exécutez le processus de démarrage de Generic Search Server.

Generic Search Server est alors disponible pour répondre aux requêtes.

11.4 Mise en cluster

Le déploiement de plusieurs instances de GSS est pris en charge sur un environnement cluster. La discussion étendue des topologies de déploiement en cluster avancé n'entre pas dans le cadre de ce manuel. Voir également 12.9, «Configuration recommandée pour l'environnement de production», à la page 46.

Remarque : Il est recommandé de déployer GSS dans son propre cluster.

11.5 Génération de cibles

Les cibles générées suivantes sont spécifiques à Cúram Generic Search Server.

11.5.1 weblogicEARGSS

Cette cible génère le fichier SearchServer.ear et le copie dans le répertoire EJBServer/build/ear/WLS/, avec votre fichier ear Cúram. Elle s'exécute automatiquement comme une partie de la cible **weblogicEAR**. Le fichier ear SearchServer doit être généré après le fichier ear Cúram. Lorsque le fichier ear SearchServer a été généré, l'application est prête à être déployée dans le serveur d'application Oracle WebLogic avec les mêmes cibles générées ou les mêmes processus manuels que le fichier ear Cúram.

11.5.2 websphereEARGSS

Cette cible génère le fichier SearchServer.ear et le copie dans le répertoire EJBServer/build/ear/WLS/, avec votre fichier ear Cúram. Elle est exécutée automatiquement comme une partie de la cible **websphereEAR**. Le fichier ear SearchServer doit être généré après le fichier ear Cúram. Lorsque le fichier ear SearchServer a été généré, l'application est prête pour être déployée dans le serveur d'application IBM®WebSphere avec les mêmes cibles générées ou les mêmes processus manuels que le fichier ear Cúram.

11.5.3 runExtractor

Cette cible doit être exécutée après la configuration de votre base de données d'application. Par défaut, elle extrait toutes les données associées aux services de recherche CEF ainsi qu'à tous autres services de recherche définis hors de votre base de données d'application. Elle les convertit dans un format approprié à l'indexage. La durée de ce processus augmente avec la quantité de données à extraire. Cette cible peut être exécutée plusieurs fois, si nécessaire.

Cette cible peut être exécutée via un service de recherche unique en spécifiant la propriété «SERVICE». Par exemple, «build runExtractor -DSERVICE=PersonSearch»

11.5.4 runPersist

Si vous utilisez un index de base de données conservé (voir 12.3, «Persistance d'index», à la page 43), cette cible génère l'index à partir des tables de base de données de transfert. Elle doit uniquement être exécutée après la configuration de votre base de données d'application et l'exécution de la cible runExtract. La cible runExtract génère votre index conservé si la persistance est configurée. Par conséquent, cette cible doit uniquement être exécutée séparément si vous avez modifié votre configuration depuis l'exécution de la cible runExtractor.

11.5.5 startupSearchServer

Cette cible est facultative. Si vous devez l'exécuter, exécutez-la après le déploiement de Generic Search Server sur votre serveur d'application. Elle déclenche la configuration des index par le serveur de recherche afin qu'ils soient disponibles pour la recherche. La durée de ce processus augmente avec la quantité de données à indexer. Si vous n'exécutez pas la cible de démarrage explicitement, le serveur de recherche initialise ses index lors de la première demande de recherche. Cette fonction est principalement prévue pour faciliter les tests avec les petits fichiers. Pour les fichiers volumineux, la fonction de démarrage automatique ne doit pas être utilisée. Vous pouvez désactiver le démarrage automatique en définissant la propriété «curam.searchserver.autostartup.disabled» sur "true" dans votre propriété `Bootstrap.properties`. lorsque vous définissez votre fichier ear (recommandé).

11.6 Performances de la base de données

L'application Cúram et l'application du serveur de recherche partagent une base de données commune, mais ont des exigences très différentes à son sujet. La table SearchServiceRow voit l'essentiel des écritures et des accès, puis devient très volumineuse, puisqu'elle contient une version de toutes les données interrogeables. L'application Cúram écrit à cette table à mesure que des entités interrogeables sont insérées ou mises à jour. Régulièrement, si votre serveur de recherche est redémarré ou lorsqu'il se synchronise, de nombreuses lectures proviennent de cette table. Il est judicieux de placer la table SearchServiceRow dans un espace table différent des autres tables de l'application, selon les ressources et les besoins de vos organisations.

11.7 Remarques sur le temps

Si des machines différentes sont utilisées pour exécuter des instances de l'application Curam et du serveur Generic Search Server, tous les systèmes doivent être synchronisés au niveau de leurs horloges et doivent rester sur le même fuseau horaire. Nous recommandons d'utiliser une solution logicielle telle que NTP (selon votre plateforme de déploiement) pour garantir que ce soit toujours le cas. Si ce n'est pas fait, alors rien ne garantit que toutes les mises à jour des données d'application seront fidèlement reflétées par Generic Search Server.

Chapitre 12. Performances

12.1 Introduction

Ce chapitre décrit les performances du serveur de recherche Cúram et la manière dont les différents scénarios de déploiement et les paramètres de configuration peuvent l'influencer.

12.2 Types d'index

Comme décrit dans 2.3, «Index», à la page 3, un index est une structure de données à l'origine des recherches GSS. Cette structure de données peut être relativement volumineuse (voir 12.7.1, «Calcul de la taille d'index», à la page 46), ce qui soulève la question : où la stocker ? GSS propose deux options : en mémoire ou dans un fichier. Pour obtenir plus d'informations sur la configuration de ces propriétés, voir A.1, «Propriétés de configuration», à la page 47

Les répertoires de mémoire RAM (en mémoire) doivent être régénérés à chaque démarrage d'un serveur d'application (sauf si la persistance est utilisée, voir 12.3, «Persistance d'index»). Ils sont rapides d'accès, mais leurs exigences en termes de mémoire peuvent dépasser les ressources disponibles. Toutefois, les répertoires de mémoire RAM peuvent être très utiles pour les tests, puisqu'ils ne conservent pas les états.

Les index de fichier utilisent le système de fichiers local pour stocker l'index. Même si la spécification J2EE ne couvre pas l'accès au système de fichiers, en pratique, cela fonctionne avec toutes les versions prises en charge documentées dans un document distinct, *Curam Supported Prerequisites (Conditions préalables de prise en charge Curam)*. Naturellement, plus les performances du système de fichiers sous-jacent sont bonnes, plus les performances de GSS le seront.

12.3 Persistance d'index

Chaque service de recherche est associé à un index interrogé lors de chaque recherche. Cet index est généré à partir des tables de la base de données de transfert lorsque le serveur de recherche s'initialise. Un long moment peut être nécessaire pour lire toutes les données du service de recherche à partir des tables de la base de données de transfert, puis pour générer les index pertinents pour ces données.

Generic Search Server fournit les moyens de conserver l'index actuel dans la base de données de sorte à améliorer le temps de démarrage. Lorsque la persistance d'index est activée, l'index conservé est chargé (s'il est disponible) avant l'interrogation des tables de transfert. S'il n'est pas disponible, toutes les données sont lues à partir des tables de transfert et le démarrage est plus lent.

L'index conservé est associé à un horodatage, et il est stocké dans la table de service de recherche appropriée pour cet index. Cet horodatage indique l'heure à laquelle l'index RAM a été conservé pour la dernière fois sur le disque. Connaître cette heure permet à Generic Search Server de récupérer toutes les données (nouvelles ou modifiées) du service de recherche à partir des tables de transfert. L'index conservé et les données (nouvelles ou modifiées) des tables de transfert fournissent un index en mémoire complet prêt pour la recherche. L'heure est enregistrée en réduisant l'accès aux tables de transfert et au traitement associé lors de la génération d'index.

Les données de l'index conservé sont stockées au format d'objets BLOB. Par conséquent, les performances de lecture et d'écriture d'un index volumineux, vers et à partir d'une base de données, sont optimales.

12.3.1 Appel d'opération de persistance

L'opération de traitement par lots DataBaseIndexPersist.persistIndex() est exécutée pour effectuer la sauvegarde de tous les index. Le processus de conservation des index consiste à :

1. lire l'index conservé en cours
2. lire les données (nouvelles ou modifiées) à partir des données de la table de transfert
3. générer un index en mémoire avec les étapes 1) + 2) ci-dessus
4. enregistrer l'index en mémoire généré dans la base de données
5. répéter les étapes 1) à 4) pour tous les services de recherche.

12.4 Considérations opérationnelles et de test

Les index conservés, les index FILE, sont conçus pour conserver les index de version entre deux réinitialisations de serveurs.

Les données sont également conservées entre les opérations de régénération de base de données, ce qui peut provoquer des problèmes pour les outils de test si les données de type index ne sont plus cohérentes avec la base de données actuelle.

De même, dans un paramètre opérationnel, si des mises à jour de base de données se produisent sans que les mises à jour des index de recherche soit activées dans l'application (via la propriété «curam.lucene.luceneOnlineSynchronizationEnabled»), les données de l'index deviendront obsolètes et des problèmes peuvent survenir.

Dans le cas des scénarios ci-dessus, les données conservées peuvent être manuellement supprimées à partir de la base de données en supprimant toutes les tables de base de données qui commencent par «GSS_» (il y aura une table pour chaque service de recherche). Les index conservés seront régénérés normalement lors de l'exécution d'une opération d'extraction et de conservation.

Dans le cas d'un index FILE, le fichier peut être supprimé, et dans le cas d'un service de recherche RAM standard rencontrant de tels problèmes, réexécuter le processus d'extraction résoudra le problème.

12.5 Régler les performances

Cette section décrit les paramètres qui influencent les performances de lecture et d'écriture de l'index de recherche. Ces paramètres déterminent comment l'index est généré et comment les nouvelles entrées doivent y être inscrites.

12.5.1 Documents de fusion maximale

`curam.searchserver.luceneadaptor.searcher.index.maxmergedocs`

Cette propriété améliore les temps de recherche pour les valeurs plus hautes et donne de meilleurs résultats pour les plus petites valeurs lorsqu'un index est fréquemment mis à jour. Les petites valeurs (par exemple, moins de 10 000) sont meilleures lorsque l'index est fréquemment mis à jour. Toutefois, les performances des temps de recherche seront affectées. La valeur par défaut est 10 000 000. Lorsque les performances de recherche sont plus importantes, cette valeur doit être grande, par exemple, la valeur par défaut, ou bien lorsque les données de recherche mettant à jour les performances sont plus importantes, alors la valeur doit être une petite valeur, par exemple 10 000.

12.5.2 Facteur de fusion

`curam.searchserver.luceneadaptor.searcher.pool.mergefactor`

Cette propriété a un impact sur la mémoire RAM utilisée lors de la mise à jour d'un index. L'index nécessite une mise à jour suite a une recherche affectant les mise à jours des données d'application. Pour

les petites valeurs (inférieures à 10), les recherches seront plus rapides, toutefois, les mises à jour de l'index de recherche seront plus lentes. Avec de plus grandes valeurs (supérieures à 10), plus de mémoire RAM est utilisée lors de la mise à jour de l'index, et lorsque les recherches sont plus lentes, la mise à jour de l'index est plus rapide. La valeur par défaut est 10. Lorsque les performances de recherche sont plus importantes, cette valeur doit être inférieure à 10, ou bien lorsque les performances de mise à jour des données de recherche sont plus importantes, alors la valeur doit être supérieure à 10.

12.5.3 Activation de la persistance

`curam.searchserver.server.index.persistence.enable`

ajoutez curam.searchserver.server.index.persistence.enable=true à la propriété Bootstrap.properties pour activer la persistance d'index.

Remarque : si cette propriété est activée, les nouveaux index conservés sont également générés lors de l'exécution de l'extraction de base de données.

12.5.4 Références

Pour plus d'informations sur les paramètres détaillés dans cette section, reportez-vous à http://lucene.apache.org/java/2_2_0/api/index.html

12.6 Regroupement en pool de l'outil de recherche

Cette section décrit la configuration des pools de recherche et leur influence sur les performances de la recherche.

12.6.1 Présentation

Lucene dispose d'un mécanisme de mise en cache interne qui rend les recherches effectuées à l'aide des objets IndexSearcher anciens plus rapides que les recherches effectuées avec les nouvelles instances IndexSearcher. Une instance partagée IndexSearcher serait suffisante pour obtenir des recherches rapides dans l'environnement mono-utilisateur, toutefois, dans la plupart des cas d'utilisation d'un environnement de serveur, plusieurs clients interrogent l'index simultanément. Pour éviter le séquencement requis par la recherche dans ce paramètre, ce qui diminuerait les performances de la recherche individuelle, GSS utilise un pool IndexSearcher qui conserve un nombre défini d'instances IndexSearcher qui peuvent être réutilisées par les demandes de recherche simultanées.

Une instance IndexSearcher voit uniquement l'index tel qu'il était au "point dans le temps" où il a été ouvert. Les mises à jour effectuées sur l'index après l'ouverture d'IndexSearcher n'apparaissent pas jusqu'à la réouverture d'IndexSearcher. Chaque instance IndexSearcher peut utiliser une quantité importante de mémoire en fonction de la taille de l'index et du fait que l'index a été mis à jour dans l'intervalle ou non. Le pool IndexSearcher s'occupe de fermer et de rouvrir les instances IndexSearcher lors d'une mise à jour de l'index.

12.6.2 Propriétés de configuration de pool

Le pool IndexSearcher a deux options de base : taille initiale et taille maximale. Le paramètre suivant

`curam.searchserver.luceneadaptor.searcher.pool.initialsize`

spécifie le nombre d'instances IndexSearcher qui seront ouvertes au démarrage et resteront ouvertes en permanence afin d'être utilisées par les clients de recherche. Il s'agit d'une option obligatoire dont les valeurs doivent être un entier positif, y compris 0. Si aucune valeur n'est spécifiée, elle est par défaut "0". Généralement, cette propriété doit être définie sur le nombre maximal anticipé de recherches client simultanées.

`curam.searchserver.luceneadaptor.searcher.pool.maxsize`

spécifie le nombre maximal d'instances IndexSearcher pouvant être ouvertes à un moment donné. Si le nombre de recherches dépasse ce nombre à un moment, une exception est générée et consignée à des fins de diagnostic. Cette option prend des valeurs d'entiers positifs. Lorsqu'elle n'est pas spécifiée, sa valeur par défaut est "100". Il y a également l'option associée

```
curam.searchserver.luceneadaptor.searcher.pool.maxsizeunbounded
```

qui signifie que la taille de pool maximale est illimitée. L'option accepte les valeurs "true" ou "false". Lorsqu'elle n'est pas spécifiée, sa valeur par défaut est "true". Si cette option est définie sur "true", la valeur de l'option curam.searchserver.luceneadaptor.searcher.pool.maxsize est ignorée. L'une de ces deux options associées est nécessaire.

12.7 Limitations de mémoire RAM

Les index de Global Search Server sont stockés en mémoire s'ils sont configurés pour ce faire. Si vous utilisez une machine virtuelle Java 32 bits, la mémoire est limitée à ~3 Go. Toutefois, cette figure ne correspond pas uniquement à la mémoire disponible pour GSS, mais également pour les autres processus système. Il est important de remarquer que des index de service de recherche volumineux peuvent dépasser le maximum de mémoire RAM disponible pour GSS et les autres processus déployés.

12.7.1 Calcul de la taille d'index

La taille de l'index est approximativement de 30 % du texte indexé. Les propriétés stockées et indexées du service de recherche (elles peuvent être obtenues à partir des attributs SearchServiceField où indexed=true et stored=true) sont utilisées pour évaluer la taille de l'index.

- 1 million d'enregistrements de personne, où 1 enregistrement = 1 document d'index.
- 1 document peut contenir les propriétés indexées et stockées suivantes déterminées à partir de la table SearchServiceField pour un service PersonSearch : refnumber(10) forename(20), surname(20), AddressLine1(30), AddressLine2(30), city(20), country(15), gender(10), où (*) = valeur de taille maximale en nombre de caractères pour cette zone.
- 1 document = (155 caractères pour la valeur stockée) + (66 caractères pour chaque nom de zone/terme.) = 221.
- 1 Mo de mémoire de documents de personne et Java utilisant des caractères Unicode de 16 bits. Texte indexé et renvoyé total 442 Mo * 30 % = 132 Mo.

12.8 Configuration recommandée

La configuration recommandée pour le Cúram Generic Search Server est l'utilisation d'un type d'index FILE dont la persistance d'index est par défaut désactivée. Cette configuration devrait générer de bonnes performances sans entraîner de problème de définition de la taille. Le serveur de recherche doit être déployé en tant qu'application distincte et ne doit pas se situer au même endroit que l'application Cúram (voir Chapitre 11, «Déploiement de Generic Search Server», à la page 39.

12.9 Configuration recommandée pour l'environnement de production

Le type d'index FILE est la seule configuration prise en charge dans l'environnement de production.

Annexe A. Propriétés de configuration de Cúram Generic Search Server

A.1 Propriétés de configuration

Avant de commencer le développement ou de déploiement de votre serveur de recherche générique Cúram, les paramètres suivants doivent être ajoutés au fichier `Bootstrap.properties`, si nécessaire.

Tableau 3. Paramètres de configuration de base de Cúram Generic Search Server

Nom de la propriété	Description
curam.searchserver.sync.interval	L'intervalle en millisecondes entre les invocations de synchronisation Generic Search Server. Il s'agit effectivement du temps maximal entre la mise à jour des données et leur disponibilité pour la recherche. Si cette propriété n'est pas définie, sa valeur par défaut est de synchroniser toutes les 3 secondes.
curam.searchserver.sync.username	Le nom d'utilisateur utilisé pour la connexion à l'application afin d'effectuer la synchronisation. L'utilisateur doit être autorisé à exécuter l'identificateur de la fonction DoGSSSync.sync. Uniquement nécessaire lors de l'exécution sous WebSphere Application Server. Oublier de spécifier cette propriété et le mot de passe associé n'empêche pas l'exécution de l'opération de synchronisation, mais cela se traduit par des avertissement de sécurité consignés dans les fichiers journaux lors de chaque synchronisation.
curam.searchserver.sync.password	Les mots de passe associés à la propriété curam.searchserver.sync.username décrite dans l'entrée ci-dessus. Ces mots de passe doivent être chiffrés à l'aide de la cible de version de chiffrement Cúram standard.
curam.searchserver.environment.vendor	Cette propriété doit être définie sur «ITD», «IBM» ou «BEA» selon que vous utilisiez le serveur de recherche en mode développement ou que vous le déployiez sous WebSphere ou WebLogic. Si cette propriété n'est pas définie le serveur de recherche utilisera par défaut la propriété curam.environment.as.vendor.
curam.searchserver.server.host	Le nom de domaine ou l'adresse IP du serveur sur lequel le serveur de recherche s'exécute. Ce doit être défini pour que vous puissiez exécuter le processus de démarrage du serveur à partir de la ligne de commande. Si cette propriété n'est pas définie, sa valeur par défaut est "localhost".
curam.searchserver.server.port	Le port sur lequel le service RMI de votre serveur d'applications est disponible. Il doit être défini pour que vous puissiez exécuter le processus de démarrage du serveur à partir de la ligne de commande.
curam.searchserver.autostartup.disabled	A des fins de test et de développement, le serveur de recherche initialisera ses index lors de la première demande de recherche, sauf s'il a déjà été démarré. Dans un scénario de déploiement, vous souhaitez peut-être désactiver ce comportement et vous assurer que le processus de démarrage s'exécute à partir de la ligne de commande, ce qui vous donne davantage de contrôle sur le processus. La définition de cette propriété sur "true" désactive le comportement de démarrage automatique. Remarquez que le serveur de recherche générera une exception en réponse à toute tentative de recherche survenant avant la fin du démarrage.
curam.searchserver. luceneadaptor.searcher.index.maxmergedocs	Cette propriété est utilisée pour personnaliser les performances de lecture et d'écriture de l'index. Les valeurs supérieures à «1 000 000» sont meilleures pour l'écriture d'index par lots et pour des recherches plus rapides. Les valeurs inférieures à «10 000» sont meilleures pour l'indexation interactive où plusieurs mises à jour d'index individuels se produisent.
curam.searchserver.luceneadaptor.document.flush.count	Indique le nombre de documents à mettre à jour avant le vidage de l'index, lors du traitement de lots volumineux de documents. Si elle n'est pas spécifiée, sa valeur par défaut est 1 000 documents. L'optimisation de cette propriété peut réduire le temps requis pour générer votre index initialement, lors de la conservation d'index ou du démarrage du serveur.
curam.searchserver.term.min.length	Longueur minimale autorisée d'un terme de recherche. Par défaut, cette longueur est de deux caractères. L'utilisation de termes de recherche très courts se traduit par des performances de recherche réduites et généralement par des résultats de recherche de mauvaise qualité.
curam.searchserver.directory.type	Spécifie le type de stockage à utiliser pour les services de recherche (RAM, FILE, etc.). RAM est le type d'index par défaut et est approprié pour les index plus petits qui demandent des performances très rapides. Le paramètre FILE fournit un stockage pour les index volumineux du système de fichiers.
curam.searchserver.file.index.location	Cette propriété indique où stocker l'index de fichier sur le système de fichier si curam.searchserver.directory.type=FILE avec plus de données. Lors d'un déploiement sur plusieurs machines, l'emplacement de fichier doit se trouver sur la machine ciblée.

Tableau 4. Cúram Generic Search Server Searcher Pool Settings

Nom de la propriété	Description
curam.searchserver.luceneadaptor.searcher.pool.initialsize	Cette propriété initialise le nombre d'outils de recherch dans le pool d'outils de recherche au démarrage. Par défaut, la valeur est définie sur 0.
curam.searchserver.luceneadaptor.searcher.pool.maxsize	Cette propriété indique le nombre maximal d'instances IndexSearchers dans le pool d'outils de recherche. Sa valeur par défaut est 100.
curam.searchserver.luceneadaptor .searcher.pool.maxsizeunbounded	Cette propriété définie sur «true» remplace curam.searchserver.luceneadaptor.searcher.pool.maxsize et indique qu'il n'y a pas de nombre maximal d'instances IndexSearchers autorisées dans le pool d'outils de recherche. Sa valeur par défaut est «true».
curam.searchserver.luceneadaptor.searcher.pool.mergefactor	Cette propriété est utilisée pour personnaliser les performances de lecture et d'écriture de l'index. Sa valeur par défaut est «10». La valeur minimale est «2». Des valeurs supérieures se traduisent par plus d'utilisation de la mémoire RAM, des recherches plus lentes, mais une écriture d'index plus rapide.

Tableau 5. Paramètres de conservation de Cúram Generic Search Server

Nom de la propriété	Description
curam.searchserver.server.index.persistence.enable	Cette propriété doit être définie sur «true» pour activer la persistance d'index. Si cette propriété n'est pas définie, sa valeur par défaut est «false».
curam.searchserver.custom.db.init	Cette propriété doit être définie sur «true» lors de la personnalisation des tables de base de données de persistance des index. Elle indique que les tables de persistance d'index par défaut ne sont pas utilisées et que le fichier `CustomDBSearchServices.properties` doit être utilisé pour définir ces tables.

Annexe B. Exemple de liste DMX : PersonSearch

B.1 Enregistrement de service de recherche

```
      <?xml version="1.0" encoding="UTF-8"?>
<table name="SEARCHSERVICE">

  <column name="
      searchServiceId
      " type="text" />
  <column name="
      nom
      " type="text" />
  <column name="
      extKeyName
      " type="text" />
  <column name="
      analyzer
      " type="text" />
  <column name="
      locked
      " type="bool" />
  <column name="
      forcedReindexTimeStamp
      " type="timestamp" />
  <column name="
      mapperName
      " type="text" />
  <column name="
      prstBlobSize
      " type="text" />
  <row>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute>
    <attribute name="name">
      <value>
      PersonSearch
      </value> </attribute>
    <attribute name="extKeyName">
      <value>
      ConcernRoleID
      </value> </attribute>
    <attribute name="analyzer">
      <value>
      STANDARD
      </value>
    </attribute>
    <attribute name="locked">
      <value>
      0
      </value>
    </attribute>
    <attribute name="forcedReindexTimeStamp">
      <value>
      SYSTIME
      </value>
    </attribute>
    <attribute name="mapperName">
      <value>
      curam.core.impl.PersonSearchMapper
```

```
      </value>
    </attribute>
    <attribute name="prstBlobSize">
      <value>
      50M
      </value>
    </attribute>
  </row>
</table>
```

B.2 Enregistrement de zone de service de recherche

```
      <?xml version="1.0" encoding="UTF-8"?>
<table name="SEARCHSERVICEFIELD">

  <column name="
      searchServiceFieldId
      " type="text" />
  <column name="
      searchServiceId
      " type="text" />
  <column name="
      nom
      " type="text" />
  <column name="
      indexed
      " type="bool" />
  <column name="
      type
      " type="text" />
  <column name="
      stored
      " type="bool" />
  <column name="
      entityName
      " type="text" />
  <column name="
      analyzerName
      " type="text" />
  <column name="
      untokenized
      " type="bool" />

  <row>
    <attribute name="searchServiceFieldId">
      <value>
      field0
      </value>
    </attribute>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute><attribute name="name">
      <value>
      primaryAlternateID
      </value>
    </attribute><attribute name="indexed">
      <value>
      1
      </value>
    </attribute><attribute name="type">
      <value>
      String
      </value>
    </attribute><attribute name="stored">
      <value>
```

```
      1
      </value>
    </attribute>
    <attribute name="entityName">
      <value>
      Person
      </value>
    </attribute>
    <attribute name="analyzerName">
      <value></value>
    </attribute>
    <attribute name="untokenized">
      <value>
      1
      </value>
    </attribute>
  </row>

  <row>
    <attribute name="searchServiceFieldId">
      <value>
      field1
      </value>
    </attribute>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute><attribute name="name">
      <value>
      firstForename
      </value>
    </attribute><attribute name="indexed">
      <value>
      1
      </value>
    </attribute><attribute name="type">
      <value>
      String
      </value>
    </attribute>
    <attribute name="stored">
      <value>
      1
      </value>
    </attribute>
    <attribute name="entityName">
      <value>
      AlternateName
      </value>
    </attribute>
    <attribute name="analyzerName">
      <value>
      STANDARD
      </value>
    </attribute>
    <attribute name="untokenized">
      <value>
      0
      </value>
    </attribute>
  </row>

......

</table>
```

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM. IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Licence sur la propriété intellectuelle
Mentions légales et droit de propriété intellectuelle
IBM Japon Ltd
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. INTERNATIONAL BUSINESS MACHINES CORPORATION FOURNIT CETTE PUBLICATION "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, EXPLICITE OU IMPLICITE, Y COMPRIS NOTAMMENT, LES GARANTIES IMPLICITES DE NON-CONTREFACON, DE QUALITE MARCHANDE OU D'ADEQUATION A UN USAGE PARTICULIER. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies. Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
Dept F6, Bldg 1
294 Route 100
Somers NY 10589-3216
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM, conformément aux dispositions du Livret contractuel, des Conditions Internationales d'Utilisation de Logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles.

IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Ces informations sont fournies uniquement à titre de planification. Elles sont susceptibles d'être modifiées avant la mise à disposition des produits décrits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les

exemples de programmes sont fournis "EN L'ETAT", sans garantie d'aucune sorte. IBM décline toute responsabilité relative aux dommages éventuels résultant de l'utilisation de ces exemples de programmes.

Toute copie intégrale ou partielle de ces exemples de programmes et des oeuvres qui en sont dérivées doit inclure une mention de droits d'auteur libellée comme suit :

© (nom de votre société) (année). Des segments de code sont dérivés des exemples de programmes d'IBM Corp.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Marques

IBM, le logo IBM et ibm.com sont des marques ou des marques déposées d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. Une liste des marques commerciales actuelles d'IBM est disponible sur Internet sous "Droits d'auteur et marques" à l'adresse http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache est une marque d'Apache Software Foundation.

Oracle, WebLogic Server, Java et toutes les marques et logos incluant Java sont des marques d'Oracle et/ou de ses affiliés.

D'autres noms peuvent être des marques de leurs propriétaires respectifs. Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.