IBM Cúram Social Program Management

???Cúram-Server für generische Suche

Version 6.0.5

IBM Cúram Social Program Management

???Cúram-Server für generische Suche

Version 6.0.5

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen in „Bemerkungen" auf Seite 53 gelesen werden.

Überarbeitung: Mai 2013

Diese Ausgabe bezieht sich auf IBM Cúram Social Program Management v6.0.5 und alle nachfolgenden Releases, sofern nicht anderweitig in neuen Auusgaben angegeben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Kapitel 1. Einführung

1.1 Handbuch ???Cúram-Server für generische Suche

Das Handbuch ???Cúram-Server für generische Suche ist ein von der IBM Corporation bereitgestelltes Tool, mit dem Sie leistungsstarke und skalierbare Suchabfragen für Ihre Anwendungslösung entwickeln können.

Das vorliegende Dokument enthält eine Beschreibung des Cúram-Servers für generische Suche und eine Übersicht über dessen Architektur. Außerdem dient es als Referenz für die Konfiguration des Servers für generische Suche und dessen Datenbanktabellen. Schließlich stellt es ein umfassendes Beispiel für die Vorgehensweise zum Implementieren einer Suche mit dem Cúram-Server für generische Suche bereit.

1.2 Voraussetzungen

Die Leser des Handbuchs ???Cúram-Server für generische Suche sollten sowohl mit der Cúram-Architektur als auch mit den Anweisungen und Prozessen der Cúram-Modellierung und -Entwicklung vertraut sein.

1.3 Zielgruppe

Das vorliegende Dokument richtet sich an Architekten, Designer und Entwickler, die Suchseiten mithilfe des Cúram-Servers für generische Suche implementieren wollen.

Kapitel 2. Konzepte und Definitionen

2.1 Einführung

In diesem Kapitel werden neben den zum Cúram-Server für generische Suche gehörigen Definitionen, die in dem gesamten vorliegenden Dokument verwendet werden, einige wichtige Such- und Indexierungskonzepte eingeführt.

2.2 Server für generische Suche

Der Cúram-Server für generische Suche ist eine eigenständige Anwendung, die über eine Reihe von APIs die leistungsstarke Suche von Anwendungsdaten unterstützt. Im Hintergrund wird der Server für generische Suche mithilfe der Apache Lucene-API implementiert. Wer Suchabfragen implementiert, sollte ausschließlich die vom GSS verfügbar gemachten APIs verwenden.

Der Server für generische Suche kann als einfache Java™-Anwendung (zur Vereinfachung der Tests bei der Entwicklung) und auch als J2EE-Anwendung bereitgestellt werden.

2.3 Indizes

Das Herzstück des Servers für generische Suche ist das Konzept der Suche in einem Index. Dies ist eine leistungsstarke, datenbankunabhängige Darstellung einer Gruppe zusammengehöriger durchsuchbarer Daten. Ein Index eines Servers für generische Suche ist ein „invertierter Index", der Wörter den Datenbanksätzen zuordnet, in denen sie angezeigt werden.

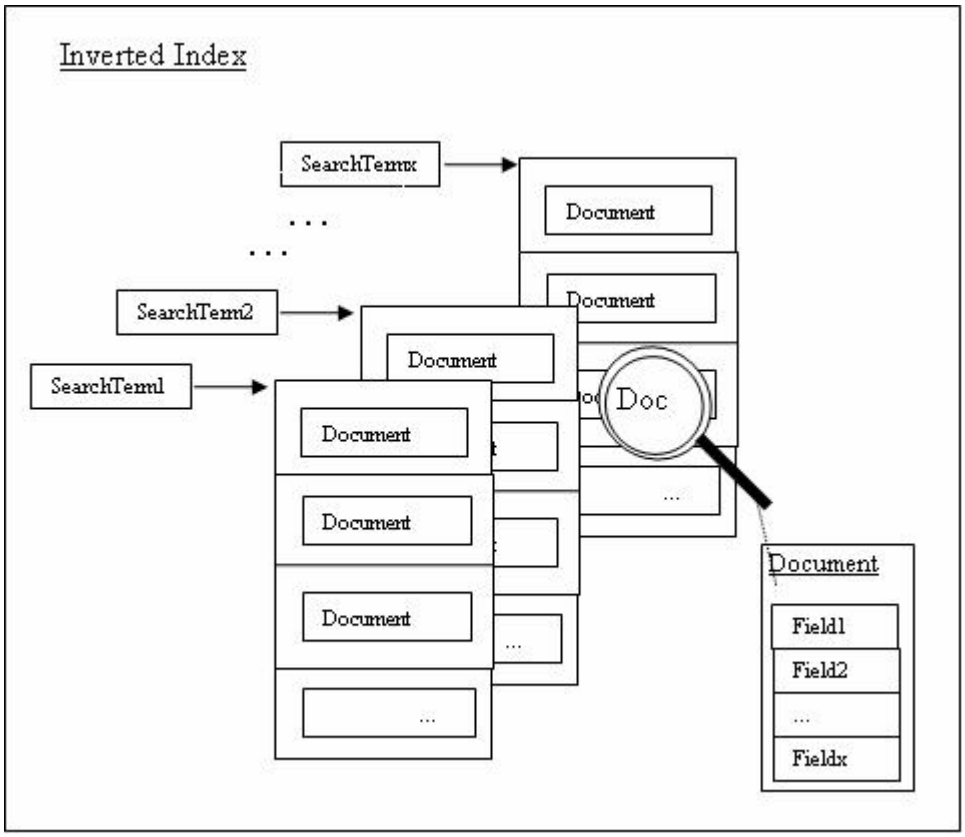

Abbildung 1. Invertierter Index - Beschreibung

Wenn ein Index nach einem Wort durchsucht wird, werden alle übereinstimmenden Sätze abgerufen, ohne dass umfangreiche Datensätze durchsucht werden müssen. Folglich lassen sich solche Indizes gut skalieren und auf großen Systemen können mehrere Indizes parallel ausgeführt werden, was eine ausgezeichnete Suchleistung ermöglicht, sofern die richtige Implementierungskonfiguration und die richtigen Indexoptimierungsparameter gewählt werden.

Entwickler, die Suchabfragen für Anwendungen erstellen, bearbeiten oder verwalten Indizes nicht direkt. Dies alles wird für sie im Hintergrund vom Server für generische Suche erledigt.

2.4 Suchservice

Ein Suchservice beschreibt Folgendes:

1. Informationen zu den durchsuchten Feldern
2. Für die jeweiligen Felder und Felddatentypen verwendete Analyseprogramme
3. Entitätsinformationen zum Füllen eines Laufzeitindex
4. Status des Suchservice (ob aktuell oder ob Synchronisation erforderlich)

Aus dieser Perspektive handelt es sich bei einem Suchservice einfach um Metadaten. Im vorliegenden Dokument wird der Begriff jedoch auch verwendet, um den zur Laufzeit gefüllten Index zu beschreiben.

Es sollte jeweils ein Suchservice für jede konkrete Datengruppe definiert werden, die durchsucht werden soll (z. B. Personensuche, Zahlungssuche etc.). Bei jedem Suchvorgang muss angegeben werden, auf welchem Suchservice er ausgeführt soll.

2.5 Feld

Wie oben erwähnt, bestehen die Suchservices aus Gruppen von Feldern. Diese kann man sich ähnlich vorstellen wie Spaltendefinitionen in Datenbanktabellen. Ein Feld hat einen Namen und einen Typ und wenn es bei einer Suche zurückgegeben wird, hat es außerdem einen Wert, nämlich das Ergebnis.

Felder können als 'Stored' (gespeichert) markiert werden. Felder mit einer solchen Markierung bewirken, dass der Index physisch relevante Werte enthält, die aus der Datenbank extrahiert wurden (siehe 2.13, „Extraktor", auf Seite 6). Das bedeutet, dass ihre Werte nach einer Suche direkt aus dem Index abgerufen und an den Anrufer zurückgegeben werden können, ohne auf den zugehörigen Eintrag in der Tabelle der Anwendungsdatenbank zugreifen zu müssen. Beachten Sie jedoch, dass dies den Index vergrößert und die Leistung des Suchvorgangs beeinträchtigen kann.

Felder können auch als 'Indexiert' markiert werden. Felder mit einer solchen Markierung sind durchsuchbar, Felder ohne eine solche Markierung nicht. Diese Funktion ist für Felder wie eindeutige IDs nützlich, die im Index speicherbar sein sollen, jedoch ohne dass man sie durchsuchen kann.

Beachten Sie, dass Felder nicht als 'Stored' (gespeichert) markiert sein müssen, um durchsuchbar zu sein.

2.6 Dokument

Ein Dokument ist ein Eintrag in einem Index. Ein Dokument wiederum besteht aus einer Gruppe von Feldern. Suchergebnisse werden vom Server für generische Suche als Gruppe von Dokumenten zurückgegeben, die anschließend in Cúram-Strukturobjekte konvertiert werden können. Ein Dokument für die Personensuche könnte beispielsweise die Felder 'Vorname', 'Nachname', 'Adresse', 'Geschlecht' etc. enthalten und bei der Durchführung einer Personensuche/-abfrage (siehe 2.9, „Abfrage", auf Seite 6) auf der Grundlage einiger Eingabekriterien werden null oder mehrere solcher Dokumente zurückgegeben.

2.7 Lucene

Lucene ist ein von der Apache Software Foundation erstelltes Open-Source-Projekt. Der Cúram-Server für generische Suche verwendet Lucene im Hintergrund für seine Indexierungs- und Suchfunktion.

Anmerkung: Beachten Sie, dass die Angaben zur Indexierung und zu Lucene reine Hintergrundinformationen sind. Entwickler, die Suchabfragen mithilfe des Servers für generische Suche erstellen, brauchen Indizes oder Lucene nicht direkt zu bearbeiten. Diese sind alle in der API des Servers für generische Suche eingeschlossen.

2.8 Staging-Datenbank

Die Staging-Datenbank des Servers für generische Suche besteht aus einer Gruppe von Datenbanktabellen, die für folgende Aufgaben verwendet werden:

- Speichern von Suchservicedefinitionen, also von Informationen dazu, welche Suchservices zusammen mit ihrer Struktur verfügbar sind
- Speichern von Werten, die aus der Betriebsdatenbank extrahiert wurden, mit deren Hilfe die Indizes gefüllt werden, die den Suchservicedefinitionen entsprechen

Für die Verwendung von Datenbanktabellen als Vermittler gibt es im Wesentlichen folgende designbezogene Begründungen:

- Sie lagern die Suchabfragen aus der Hauptdatenbank aus. Das bedeutet, dass Suchabfragen die aktuelle Systemleistung nicht beeinträchtigen.
- Sie bleiben in einer für den Suchservice geeigneten Weise erhalten. Die Daten bleiben in einem Format erhalten, das für die Erstellung der Suchindizes geeignet ist. Die Anwendungsdaten werden umgewan-

delt, bereinigt und konsolidiert, bevor sie in der Staging-Datenbank gespeichert werden. Daher brauchen Batch-Jobs nicht ständig wiederholt zu werden, um die Daten bei jedem Start einer Suchserverinstanz erneut zu extrahieren.

2.9 Abfrage

Eine Abfrage ist ein Objekt (genauer gesagt, eine Struktur), das an den Server für generische Suche übergeben wird, wenn eine Suche ausgeführt wird.

2.10 Begriff

Ein Begriff ist ein Teil eines Abfrageobjekts. Derzeit gibt es drei verschiedene Arten von Begriffen: Standardbegriffe für die Suche in normalen Textfeldern, Datumsbegriffe für die Suche in Datumsfeldern und Datumsbereichsbegriffe für die Angabe von zu durchsuchenden Datumsbereichen.

2.11 Analyseprogramm

Ein Analyseprogramm ist ein Lucene-Konzept, das eine Klasse darstellt, die die abstrakte Lucene-Klasse `org.apache.lucene.analysis.Analyzer` implementiert.

Analyseprogramme bereiten Text für die Indexierung und für Suchvorgänge vor. Es ist beispielsweise nicht sinnvoll, jedes Wort in einem Textfeld zu indexieren. Stoppwörter wie „und", „von" und „ein" sind bei einer Suche möglicherweise irrelevant. Wenn diese Wörter bei einer Feldsuche ignoriert werden sollen, wird das Feld in Tokens zerlegt, d. h. es durchläuft ein Analyseprogramm, bevor das Feld zum Index geschrieben wird. In gleicher Weise wird mit einem Begriffswert verfahren, der durchsucht wird.

Analyseprogramme sind sprachspezifisch. Die Definition eines Wortes ist nicht in allen Sprachen gleich. Einige können so konfiguriert werden, dass allgemeine Stoppwörter (ein, der, wenn usw.), Zahlen etc. ignoriert werden. Die vom Server für generische Suche verwendeten Analyseprogramme können auf der Grundlage des jeweiligen Suchservice konfiguriert werden.

2.12 Zuordnungsfunktion

Eine Zuordnungsfunktion ist eine Klasse, die von Entwicklern von Anwendungssuchabfragen für jeden Suchservice geschrieben werden muss. Sie hat die Aufgabe, Daten aus der Anwendung in ein Format umzuwandeln, das in die Staging-Datenbank geschrieben und in einen Index importiert werden kann. Die Umwandlung umfasst die Ermittlung relevanter Entitätseigenschaften, die für den Suchservice von Belang sind, die Erstellung einer Liste mit diesen Werten und deren Zuordnung zu einem einzigen konsolidierten Textwert. Dieser in der Staging-Datenbank gespeicherte Wert wird später bei der Erstellung eines Dokuments mit einem einzigen Suchindex verwendet. Jeder Suchservice, der geschrieben wird, muss eine eigene Implementierung der Zuordnungsfunktion bereitstellen.

2.13 Extraktor

Der Extraktor ruft mithilfe der Metadaten des Suchservice die relevanten Anwendungsdaten ab, die zum Füllen der Suchindizes notwendig sind. Der Extraktor fragt die relevanten Anwendungsentitäten ab, die mithilfe der Metadaten ermittelt wurden, und die erforderlichen Entitätseigenschaften werden der Staging-Datenbank zugeordnet (mit der Zuordnungsfunktion), um beim Start des Suchservice die Indexierung durchzuführen.

Kapitel 3. Server für generische Suche - Übersicht

3.1 Server für generische Suche und Lucene

Die Konzepte der Indexierung und der Lucene-API wurden bereits vorgestellt. Warum also sollte man Lucene nicht direkt in einer Cúram-Anwendung verwenden?

Lucene ist zwar eine hervorragende API für die Indexierung und für Suchvorgänge, erfüllt jedoch nicht alle Voraussetzungen für ein Cúram-Suchprodukt:

- Es bietet keine Lösung für Bereitstellungsfragen, z. B. wie mehrere Suchserver ausgeführt werden sollen oder wie die Anwendung mit den Suchservern kommunizieren soll.
- Es bietet keine Lösung für die Frage, wie Daten in Indizes importiert werden sollen.
- Es bietet keine Lösung für die Frage, wie die Synchronisierung von Indexdaten mit Quellendaten in der aktiven Anwendung erhalten bleiben soll.
- Es bietet keine Lösung für die Frage der Interpretation von Daten, die bei einer Indexsuche als Cúram-Datentypen und -Strukturen zurückgegeben werden.
- Es bietet keine Lösung für die umfassendere Anwendungsvoraussetzung, den Anwendungsentwickler vor detailliertem Wissen über Produkte eines Drittherstellers zu schützen. Da Lucene nur eine von mehreren möglichen Suchlösungen darstellt, würde es sinnvoller erscheinen, eine allgemeinere Such-API bereitzustellen.

Der Cúram-Server für generische Suche wurde entwickelt, um diese Voraussetzungen zu schaffen.

3.2 Daten aus Cúram importieren

Bei Verwendung einer Indexierungstechnologie ist zu beachten, dass ein Index zuerst erstellt werden muss, damit er durchsucht werden kann. Da ein Großteil der schwierigen Sucharbeit eigentlich im Voraus bei der Indexerstellung erledigt wird, laufen Suchabfragen zur Laufzeit schnell ab. Dennoch lohnt es sich, darauf hinzuweisen, dass der Indexierungsprozess selbst eine gewisse Zeit in Anspruch nehmen kann. Diese Zeit nimmt proportional zu dem zu indexierenden Datenvolumen zu.

Die Initialisierung des Servers für generische Suche erfolgt in zwei Phasen.

In der ersten Phase werden die vorhandenen Anwendungsdaten aus der Anwendung in eine Gruppe von Datenbanktabellen exportiert, die vom Server für generische Suche verwendet werden, die Staging-Tabellen. Dieser Exportvorgang wurde als Batchprozess mit der Bezeichnung 'Extraktor der Datenbanksuche' implementiert und wird im Rahmen des Vertriebs des generische Servers für generische Suche bereitgestellt. Der Export braucht nur ein einziges Mal zu erfolgen, nämlich bei der Erstverwendung des Servers für generische Suche. Für jeden Suchservice werden spezielle Helper-Klassen benötigt, die so genannten Zuordnungsfunktionen. Sie unterstützen den Extraktor bei der Vorbereitung der Daten, die in die Staging-Tabellen importiert werden sollen.

In der zweiten Phase wird für jeden definierten Suchservice ein Index erstellt. Beim Start des Servers für generische Suche wird ein Prozess ausgeführt, um die entsprechenden Daten aus den Staging-Datenbanktabellen zu lesen und die Indizes sowie andere Datenstrukturen zu erstellen, die für Suchabfragen verwendet werden sollen. Sobald die Indizes erstellt sind, hat der Server die Möglichkeit, auf Suchanforderungen zu antworten. Informationen zur Leistungsoptimierung finden Sie in Kapitel 12, „Leistung”, auf Seite 43.

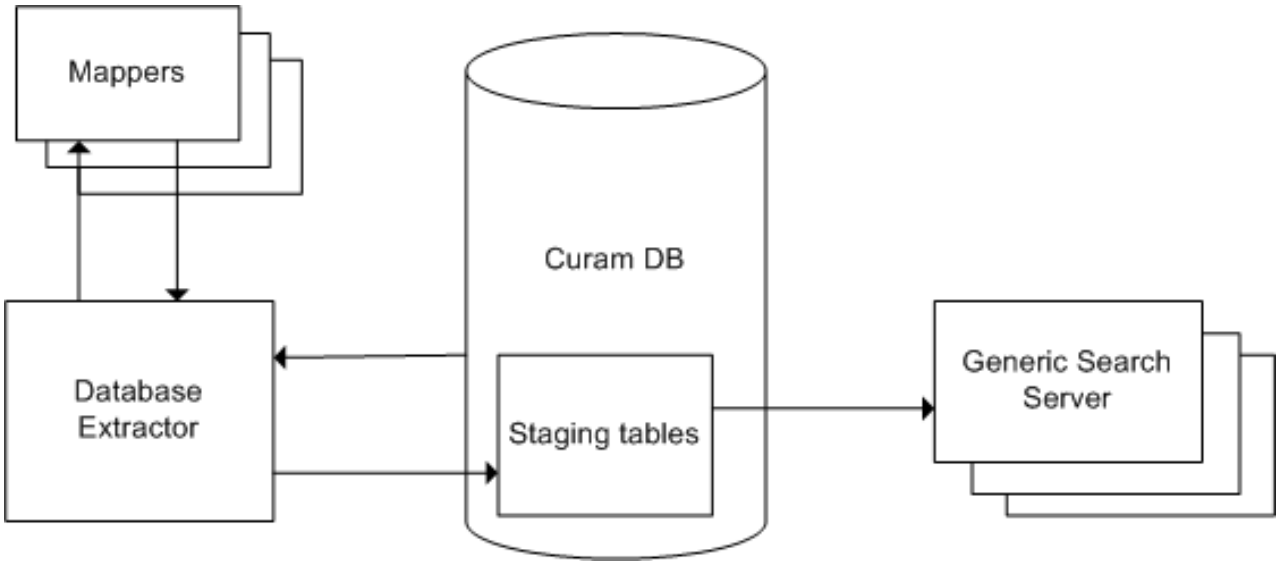

Abbildung 2. Startprozess des Datenbankextraktors und des Servers für generische Suche

3.3 Synchronisation des Suchservers

Da der Server für generische Suche die Suchabfragen nicht in den Livedaten selbst ausführt, sondern in einem Index, der aus diesen Daten erstellt wird, müssen Aktualisierungen der Anwendungsdaten in dem Index repliziert werden. In Cúram-Implementierungen ist es unerlässlich, dass Aktualisierungen durchsuchbarer Daten zeitgerecht und vorhersehbar in die relevanten Indizes übernommen werden. Beim Server für generische Suche ist der zeitliche Abstand kurz (und konfigurierbar).

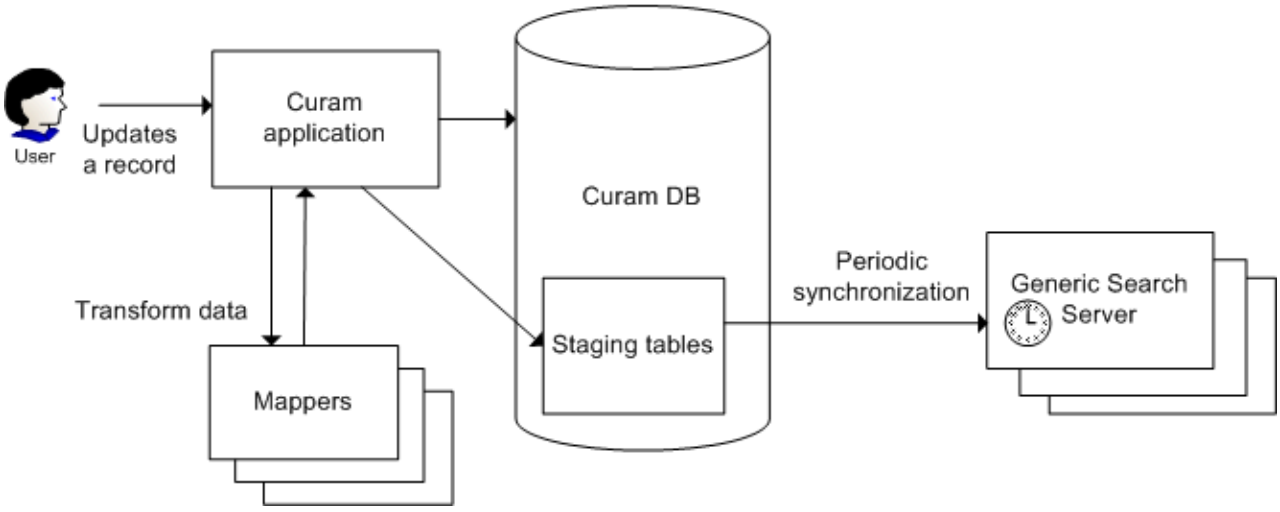

Abbildung 3. Datensynchronisation

Ähnlich wie beim oben beschriebenen Erstimport der Daten gibt es beim Synchronisationsprozess zwei Schritte.

Der erste Schritt in dem Prozess erfolgt, wenn die Anwendungsdaten (die in einem Index verwendet werden) geändert werden, in der Regel aufgrund eines Einfügungs-, Aktualisierungs- oder logischen Lösch-

vorgangs. Wenn dies geschieht, muss die Anwendung Informationen zu dieser Datenänderung in die Staging-Tabellen des Servers für generische Suche schreiben. Alle neuen und aktualisierten Elemente werden mit einer Zeitmarke markiert.

Im zweiten Schritt (der in regelmäßigen Intervallen erfolgt) synchronisiert der Server für generische Suche seine Indizes mit dem aktuellen Inhalt der Staging-Datenbank. Dazu liest er alle Elemente, die nach der Synchronisation geändert wurden, und importiert sie in die Indizes. Konkret wird dies erreicht, indem die Zeitmarke des jeweils geänderten Elements mit der letzten Zeitmarke verglichen wird, die beim letzten Synchronisationsschritt verwendet wurde.

Anmerkung: Beim Schreiben von Komponententests, die Aufrufe von Suchabfragen des Servers für generische Suche beinhalten, ist es wichtig, die Verzögerung beim Synchronisieren der Daten im Auge zu haben. Zudem wird der Server für generische Suche in einem von den Komponententests separaten Prozess ausgeführt, was dazu führt, dass er nicht zu derselben Transaktion gehört. Folglich werden bei Synchronisationen des Servers für generische Suche keine Daten einbezogen, die in der Testtransaktion geändert wurden, sofern sie nicht explizit festgeschrieben werden.

3.4 Suchcontroller

Der Suchcontroller ist eine wichtige Komponente innerhalb des Synchronisierungsmechanismus. Er verwaltet eine Liste aller Entitäten, die den einzelnen Suchservices zugeordnet sind.

Wenn eine Entität geändert wird, kann der Suchcontroller überprüft werden, um zu ermitteln, ob diese Entität von einem oder mehreren Suchservices verwendet wird. Ist dies der Fall, sollten die Daten in der Staging-Datenbank in derselben Transaktion aktualisiert werden wie die Entität. Der Suchcontroller stellt auch eine API zur Aktualisierung der Staging-Datenbank zur Verfügung.

Anmerkung: Eine Reihe von Entitäten der Cúram-Plattform (die bei bestimmten Suchabfragen der Cúram-Plattform angezeigt werden), wurde geändert, um die Implementierung solcher Synchronisierungsaktualisierungen im künftigen Release zu ermöglichen. Die Änderungen haben die Form der Erstellung von Ausstiegspunkten ???vor oder nach Operationen angenommen, die ???abgebrochene Implementierungen enthalten. Diese vor- und nachgeschalteten Ausstiegspunkte sind für die künftige Implementierung reserviert und sollten nicht direkt vom Kunden geändert werden.

3.5 Suchvorgang

Der Suchvorgang lässt sich in drei Phasen unterteilen.

In der ersten Phase erstellt die Cúram-Anwendung eine gültige Abfrage, die dem Server für generische Suche vorgestellt werden soll. Sie füllt diese Abfrage mit Suchkriterien, die vom Benutzer eingegeben werden.

In der zweiten Phase kontaktiert die Cúram-Anwendung eine aktive Instanz des Servers für generische Suche und führt die Suche entsprechend der Definition durch das Abfrageobjekt aus.

In der letzten Phase interpretiert die Cúram-Anwendung die vom Server für generische Suche empfangenen Ergebnisse als Cúram-Datentypen, führt die üblichen Sicherheitsprüfungen bezüglich der Vertraulichkeit der Daten aus und zeigt sie dem Benutzer an.

3.6 Verweise

Lucene-Website: http://lucene.apache.org/.

Kapitel 4. Vom Server für generische Suche aktivierte Suchabfragen

4.1 Einführung

Die IBM Corporation hat den Server für generische Suche (GSS) als optionalen Suchmechanismus für Suchabfragen in Plattform- und Lösungsmodulen eingeführt. Einige Suchabfragen wurden unter Einbeziehung der Suchfunktionen des Cúram-Servers für generische Suche und der Datenbank implementiert, andere dagegen sind nur als GSS-Suchabfragen verfügbar. Bei den Suchabfragen, die per Datenbank oder GSS ausführbar sind, können die Kunden für jeden Suchvorgang die leistungsstarke Suche aktivieren oder inaktivieren, indem sie die Anwendungseigenschaften festlegen.

4.2 Zum Server für generische Suche gehörige Eigenschaften in der Cúram-Anwendung

Bei diesen Eigenschaften handelt es sich um die Anwendungssystemeigenschaften. Diese können in der Anwendung in gewohnter Weise über die Eigenschaftsverwaltung verwaltet werden. Alle relevanten Eigenschaften sind in der Kategorie „Anwendung - erweiterte Lucene-Suchparameter" verfügbar. Eine vollständige Liste dieser Eigenschaften finden Sie unter A.1, „Konfigurationseigenschaften", auf Seite 47.

Tabelle 1. Zum Cúram-Server für generische Suche gehörige Eigenschaften

Name der Eigenschaft	Beschreibung
curam.lucene.luceneEnhancedSearchEnabled	Standardwert: „NEIN". Standardmäßig ist der gesamte Funktionsumfang des Servers für generische Suche inaktiviert. Um die erweiterte Suche zu aktivieren, müssen Sie diese Eigenschaft auf „JA" einstellen. Solange diese Eigenschaft nicht auf „JA" eingestellt wird, sind keine erweiterten Suchabfragen verfügbar.
curam.lucene. luceneOnlineSynchronizationEnabled	Standardwert: „NEIN". Um den Ereignisveröffentlichungsmechanismus zu aktivieren, der Änderungen an den durchsuchbaren Daten für den Suchserver verfügbar macht, müssen Sie diese Eigenschaft auf „JA" einstellen. Solange dies nicht geschieht, werden Einfügungen und Aktualisierungen durchsuchbarer Daten nicht an den Suchserver weitergegeben.
curam.lucene.externalUpdateEventsEnabled	Standardwert: „NEIN". ???To ensure that if any search service related data is updated externally, then the external system receives related update synchronization events to synchronization the searchable data, in case if property "curam.lucene.luceneOnlineSynchronizationEnabled" is not enabled. Die Aktivierung dieser Eigenschaft hat dieselbe Auswirkung wie „curam.lucene.luceneOnlineSynchronizationEnabled" für die Anwendung. Um die Eigenschaft „curam.lucene.externalUpdateEventsEnabled" zu aktivieren, müssen Sie diese Eigenschaft auf „JA" einstellen.

Außerdem besitzt jede Suche, die die erweiterte Suche unterstützt, eine Eigenschaft, die bestimmt, ob sie den Server für generische Suche oder die Datenbank verwendet. Dadurch kann jede Organisation auf der Grundlage jedes einzelnen Suchvorgangs wählen, welche erweiterten Suchabfragen verwendet werden sollen.

4.3 Cúram-Daten und Suchdaten synchronisiert halten

Es ist notwendig, dass die aktuellen Anwendungsdaten und der Suchindex synchronisiert bleiben, um korrekte Suchergebnisse zu gewährleisten. Die Infrastruktur, die der GSS dafür bereitstellt, ist an anderer Stelle beschrieben (siehe 6.3, „Suchcontroller", auf Seite 18).

Anwendungsentwickler sind jedoch auch verpflichtet, Aufrufe des Suchcontrollers hinzuzufügen, wenn in der Anwendung relevante Daten geändert werden. Im diesem Abschnitt wird zu Informationszwecken beschrieben, welches ereignisgesteuerte Verfahren verwendet wird und welches für Kunden empfohlen wird, die eigene GSS-basierte Suchabfragen implementieren.

Neben dem Ereignismechanismus wird auch die Synchronisation der Zuordnungsfunktion für Extrahierung bereitgestellt, die im vorliegenden Handbuch in einem eigenen Kapitel beschrieben wird (siehe Kapitel 8, „Zuordnungsfunktion für Extrahierung", auf Seite 29.

4.3.1 Ereignisgesteuerte Synchronisation

Cúram stellt Ereignisse bereit, mit denen lose verbundene Teile der Anwendung einander über Statusänderungen informieren können. Sie werden im Handbuch *Cúram Server Developer's Guide* dokumentiert.

Für jede Entität, die an einem Suchservice beteiligt ist, sollte ein Ereignis ausgelöst werden, wenn sie erstellt, gelöscht oder geändert wird. Anschließend ruft der Ereignishandler die Klasse `SearchController` auf, um den Suchserver mit der Änderung zu aktualisieren.

Für jede Entität, die an einem Suchservice beteiligt ist, müssen Vorgänge nach Änderungen, Einfügungen und Löschungen hinzugefügt werden, die die Ereignisse auslösen.

Kapitel 5. Staging-Datenbanktabellen

5.1 Einführung

Die Staging-Datenbanktabellen sind Datenbanktabellen in der Betriebsdatenbank, die vom Server für generische Suche verwendet werden. Es gibt vier Tabellen: SearchService, SearchServiceField, SearchServiceRow und SearchSrvcRowExt.

In diesem Kapitel werden der Zweck und die Struktur der Tabellen 'SearchService' und 'SearchServiceField' detailliert beschrieben. Entwickler, die Suchservices erstellen, müssen nicht direkt auf die Tabelle 'SearchServiceRow' oder 'SearchSrvcRowExt' zugreifen und auch keine DMX-Dateien für sie schreiben.

In der Tabelle 'SearchService' werden die Suchservices definiert, die dem Server für generische Suche bekannt sind (eine Einführung in Suchservices finden Sie unter 2.4, „Suchservice", auf Seite 4). Da keine Verwaltungs-API für die Verwaltung von Suchservices bereitgestellt wurde, müssen Suchserviceeinträge derzeit entweder über direkten Zugriff auf die Datenbanktabelle oder durch Bearbeitung der DMX-Dateien und Neuerstellung der Anwendungsdatenbank erstellt und verwaltet werden.

Die Tabelle 'SearchServiceField' definiert ein einzelnes Feld eines Suchservice - den Feldnamen, seinen Datentyp und weitere Attribute, die unten ausführlich erläutert werden. Jeder Zeile der Datenbank 'SearchServiceField' ist eine einzelne Zeile von 'SearchService' zugeordnet. Wie bei Suchservices müssen Suchservicefeldeinträge derzeit entweder über direkten Zugriff auf die Datenbanktabelle oder durch Bearbeitung der DMX-Dateien und Neuerstellung der Anwendungsdatenbank erstellt und verwaltet werden.

Die Tabelle 'SearchServiceRow' wird dazu verwendet, durchsuchbare Daten aus der Anwendung für die Erstellung von Indizes zu speichern. Der Server für generische Suche stellt eine API bereit (siehe Kapitel 6, „API des Servers für generische Suche - erste Schritte", auf Seite 17 und Kapitel 7, „Suchvorgang mit dem Server für generische Suche implementieren", auf Seite 23), mit der SearchServiceRow-Tabellen verwaltet werden. Entwickler sollten mit dieser Datenbanktabelle nur über diese API interagieren und nicht direkt darauf zugreifen.

Es gibt zwei weitere GSS-Datenbanktabellen: 'GSSMapperType' und 'GSSEntity'. Diese werden nur mit der Zuordnungsfunktion für Extrahierung verwendet. In allen anderen Fällen können sie ignoriert werden. Diese Tabellen werden in Kapitel 8, „Zuordnungsfunktion für Extrahierung", auf Seite 29 beschrieben.

5.2 Tabelle 'SearchService'

Jeder Suchservice muss einen Eintrag in der Tabelle 'SearchService' enthalten. Gemeinsam mit den ihr untergeordneten SearchServiceField-Zeilen definiert die Tabelle 'SearchService' das Schema für die einzelnen Suchservices. Nachstehend finden Sie eine Beschreibung der einzelnen Spalten der Tabelle 'SearchService':

5.2.1 searchServiceId

Die Suchservice-ID. Eine Zeichenfolge für die eindeutige Identifizierung eines Suchservice.

5.2.2 extKeyName

Der Name eines Suchservicefeldes, mit dem jeder Eintrag in einem Index, der aus dieser Suchservicedefinition erstellt wurde, eindeutig bezeichnet wird. Es ist äußerst wichtig, dass die Werte in dem Index, der diesem Suchservicefeld entspricht, eindeutig sind, da bei der Aktualisierung durchsuchbarer Daten in der Anwendungsdatenbank mit dem Wert dieses Feldes das entsprechende Dokument ermittelt wird, das im Index zu aktualisieren ist.

5.2.3 analyzer

Das Analyseprogramm des Suchservice, das für die Konvertierung von Textbegriffen der Anwendungsdatenbank in Indexbegriffe verwendet werden soll. Der Inhalt dieser Spalte sollte einen der vordefinierten Analyseprogrammnamen bezeichnen, die vom Server für generische Suche bereitgestellt werden (siehe nachstehende Liste), oder einen vollständig qualifizierten Java-Klassennamen einer Klasse, die die abstrakte Klasse `org.apache.lucene.analysis.Analyzer` implementiert. Dies kann entweder ein standardmäßiges Lucene-Analyseprogramm oder eine Implementierung eines anderen Anbieters bzw. eine angepasste Implementierung sein. Beachten Sie, dass die Klasse im Klassenpfad des Servers für generische Suche verfügbar sein muss, wenn es sich nicht um ein standardmäßiges Lucene-Analyseprogramm handelt.

Eine Liste der im Lieferumfang des GSS enthaltenen Analyseprogramme und eine detailliertere Beschreibung der Vorgehensweise bei der Wahl eines Analyseprogramms finden Sie unter 9.6, „Analyseprogramme im Detail”, auf Seite 34.

5.2.4 frcdReidxTimeStmp

Wird vom Extraktor verwendet, um nach einer Extraktion die Neuerstellung der Indizes des Servers für generische Suche zu erzwingen. Beim Erstellen von Suchserviceeinträgen sollte dieser Wert anfangs auf Null eingestellt werden.

5.2.5 mapperName

Der Name der Implementierung der Zuordnungsfunktion (siehe 7.4, „Operationen der Zuordnungsfunktion implementieren”, auf Seite 24). Eine Implementierung der Zuordnungsfunktion ist eine Klasse, die eine Gruppe von Anwendungsentitätsdaten in ein für die Indexierung geeignetes Format konvertiert. Der Wert dieser Spalte sollte der vollständig qualifizierte Klassenname der Zuordnungsfunktionsklasse sein. Wie bei der Implementierung des Analyseprogramms sollte dies im Laufzeitklassenpfad des Servers für generische Suche sein (wenn die Zuordnungsfunktion im Rahmen der Anwendung entwickelt wird, ist sie standardmäßig im Klassenpfad).

5.2.6 dbLastWritten

Wird bei der Synchronisation verwendet. Dieses Element sollte nicht von Anwendungscode oder Administratoren initialisiert oder aktualisiert werden.

5.2.7 prstBlobSize

Gibt die Größe der BLOB-Datei an, die der Tabelle zugeordnet ist, mit deren Hilfe dieser Suchserviceindex erhalten wird. Wird kein Wert angegeben, wird standardmäßig der Wert '50 MB' angenommen. Der Eigenschaftstyp ist eine Zeichenfolge und der Wert sollte der Größenkennungssyntax der betroffenen Datenbank entsprechen.

5.3 Tabelle 'SearchServiceField'

Jedes Feld des Suchservice muss einen Eintrag in der Tabelle 'SearchServiceField' enthalten. Jedes Suchservicefeld steht für ein Suchserviceelement, das durchsucht und/oder von einer Suche zurückgegeben werden kann. Suchservicefelder werden im Server für generische Suche an verschiedenen Stellen verwendet: in Begriffen, Abfragen und Dokumenten. Nachstehend finden Sie eine Beschreibung der einzelnen Spalten der Tabelle 'SearchServiceField':

5.3.1 srchServiceFldId

Die eindeutige Kennung des Suchservicefeldes.

5.3.2 searchServiceId

Die Suchservice-ID des übergeordneten Suchserviceeintrags.

5.3.3 name

Der dem Suchservicefeld zugeordnete Name. Mit diesem Namen wird auf das Feld verwiesen, wenn Suchabfragen ausgeführt oder Ergebnisse abgerufen werden. Er muss nicht genau mit den Feldnamen in Cúram-Entitäten und -Strukturen übereinstimmen, auch wenn eine genaue Entsprechung die Entwicklung vereinfacht.

5.3.4 type

Der Cúram-Datentyp dieses Feldes. Die gültigen Werte werden in der folgenden Tabelle beschrieben.

Beim Exportieren und Synchronisieren von Daten für den Suchservice müssen Betriebsdaten in Zeichenfolgen konvertiert werden und umgekehrt. Daher ist es wichtig, für jedes Feld einen korrekten Datentyp zu definieren. Die folgende Tabelle dient Ihnen dazu als Referenz. Wenn dem Server für generische Suche falsche Werte angeboten werden, löst er eine Ausnahme aus.

Tabelle 2. Zuordnungen zwischen grundlegenden Cúram-Domänendefinitionen und GSS-Felddatentypen

Domänendefinition	GSS-Felddatentyp
SRV_BOOLEAN	boolean (boolescher Wert)
SRV_DATE	Date (Datum)
SRV_DATETIME	DateTime (Datum/Uhrzeit)
SRV_INT8	byte (Byte)
SRV_INT16	short (kurze ganze Zahl)
SRV_INT32	int (ganze Zahl)
SRV_INT64	long (lange ganze Zahl)
SRV_FLOAT	float (Gleitkommazahl)
SRV_DOUBLE	double (doppelt)
SRV_MONEY	Money (Geld)
SRV_CHAR	char (Zeichen)
SRV_STRING	String (Zeichenfolge)
SRV_UNBOUNDED_STRING	String (Zeichenfolge)

Anmerkung: Im Feld für den Datentyp muss die Groß-/Kleinschreibung beachtet werden. Verwenden Sie daher den Typnamen genau wie oben angegeben.

5.3.5 indexed

Gibt an, ob dieses Feld durchsuchbar ist. In bestimmten Fällen ist es wünschenswert, einen Wert für einen Eintrag im Suchservice zu speichern, jedoch keine Suche danach auszuführen (beispielsweise die eindeutige ID eines Eintrags oder dessen Vertraulichkeitsstufe). Wenn Werte, für die keine Indexierung erforderlich ist, nicht indiziert werden, wird die Indexgröße minimiert und die Leistung verbessert. Daher empfiehlt es sich, nur diejenigen Felder zu indexieren, die von Ihren Suchabfragen verwendet werden.

5.3.6 stored

Gibt an, ob dieses Feld in einem Suchergebnis zurückgegeben werden kann oder nicht, d. h. ob der Wert selbst im Index gespeichert wird. Beachten Sie, dass gespeicherte Felder weiterhin nur dann zurückgegeben werden, wenn das an den Server für generische Suche übergebene Abfrageobjekt angibt, dass sie zurückgegeben werden sollten. Jedes Feld sollte indexiert und/oder gespeichert werden. Ein Feld, das weder indexiert noch gespeichert ist, besitzt keine Relevanz für den Suchservice. Auch hier gilt: Wenn Werte, die in Ihren Suchabfragen nicht verwendet werden, nicht gespeichert werden, wird die Indexgröße minimiert und die Leistung verbessert. Speichern Sie daher nur diejenigen Felder, die von Ihren Suchabfragen verwendet werden.

5.3.7 entityName

Der Name der Anwendungsentität, die diesem Feld zugeordnet ist. Konkreter: Der Name der Anwendungsentität, die ein diesem Feld entsprechendes Attribut aufweist, mit dem der Index auf der Basis der übergeordneten Suchservicedefinition gefüllt wird. Diese Angabe ist für die Synchronisation von Anwendungsdaten mit dem Server für generische Suche erforderlich. Alle Entitäten, die laut Liste einen Bezug zu den Suchservicefeldern haben, werden beim Suchcontroller registriert (siehe 3.4, „Suchcontroller", auf Seite 9) und auf Einfügungen, Aktualisierungen und Löschungen überwacht. Es ist von zentraler Bedeutung, dass das Attribut 'entityName' mit den passenden Werten gefüllt wird. Fehlende oder ungültige entityName-Attribute können mit der Zeit bewirken, dass ungültige Indexaktualisierungen erfolgen.

5.3.8 untokenized

Die Eigenschaft gibt an, ob ein Feld in Tokens zerlegt werden und das Analyseprogramm durchlaufen soll oder nicht. Dies ist ein boolescher Wert. Wird er auf 'true' eingestellt, wird dieses Feld vor der Indexierung oder während einer Suche weder in Tokens zerlegt noch analysiert.

5.3.9 analyzerName

Diese Eigenschaft gibt an, mit welchem Analyseprogramm dieses Feld in Tokens zerlegt werden soll. Der Inhalt dieses Feldes kann auf folgende Werte eingestellt werden: LUCENESTANDARD, STANDARD, SIMPLE, STOP, WHITESPACE, KEYBOARD. (Siehe das Analyseprogramm unter 5.2, „Tabelle 'SearchService'", auf Seite 13.) Wird dieses Feld nicht festgelegt, wird standardmäßig das Analyseprogramm aus dem Feld 'Analyseprogramm' des zugehörigen Suchservice verwendet.

Kapitel 6. API des Servers für generische Suche - erste Schritte

6.1 Einführung

In diesem Kapitel finden Sie keine umfassende Beschreibung der gesamten API des Servers für generische Suche. Im Rahmen der Installation wird eine ganze Reihe von Javadocs verfügbar gemacht. Dieses Kapitel hat den Zweck, eine kurze Einführung in die wichtigsten Klassen und Operationen in der API bereitzustellen, um die schnelle Entwicklung von Suchabfragen auf der Basis des Servers für generische Suche zu ermöglichen.

6.2 Zuordnungsfunktionen

Zuordnungsfunktionen sind Klassen, die definieren, wie Suchservicedaten aus den Tabellen der Anwendungsdatenbank den Tabellen der Staging-Datenbank zugeordnet werden. Jeder Suchservice hat eine eigene Zuordnungsfunktion. Welche Zuordnungsfunktion verwendet werden soll, wird in der Tabelle der Datenbank 'SearchService' angegeben. Weitere Informationen finden Sie unter 5.2.5, „mapperName", auf Seite 14.

Diese Zuordnungsfunktion wird in zwei Prozessen verwendet:

1. Wird der Datenbankextraktor ausgeführt, wird jedes Suchservicefeld für einen bestimmten Suchservice iteriert. Für jedes Feld werden die entsprechenden Entitätsattributdaten aus der Anwendungsdatenbank abgerufen und in die Tabelle 'SearchServiceRow' der Staging-Datenbank gefüllt.
2. Wenn ein Erstellungs-, Aktualisierungs- oder Entfernungsvorgang für eine in einem Suchservice verwendete Entität aufgerufen wird, werden die relevanten SearchServiceRow-Zeilen mit den zugehörigen Entitätsänderungen aktualisiert.

In diesen beiden Prozessen wird die relevante Zuordnungsfunktion für jeden Suchservice aufgerufen, um Daten aus den Tabellen der Anwendungsdatenbank den Tabellen der Staging-Datenbank zuzuordnen.

Beim Initialisieren des Servers für generische Suche werden die Informationen der Staging-Datenbank gelesen und dazu verwendet, die Indizes aus den Metadaten des Suchservice zu erstellen. Der Suchserver prüft die Staging-Datenbank in regelmäßigen Abständen auf Aktualisierungen und hält die Servicedaten auf dem neuesten Stand.

Die folgenden Methoden der Zuordnungsfunktions-API müssen von den Entwicklern der Suche auf der Grundlage des jeweiligen Suchservice implementiert werden:

```
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);
```

Weitere Informationen finden Sie unter 7.4, „Operationen der Zuordnungsfunktion implementieren", auf Seite 24.

6.3 Suchcontroller

Der Suchcontroller ist ein Singleton-Objekt, das für die Verwendung in der Anwendung verfügbar ist. Seine Aufgabe besteht darin zu überwachen, auf welche Entitäten in welchen Suchservices verwiesen wird. Außerdem stellt er eine API zum Synchronisieren der Änderungen bereit, die auf dem Server für generische Suche an den Anwendungsdaten mit den relevanten Indizes vorgenommen wurden. Beachten Sie, dass sich der Suchcontroller aus einer Client-Server-Perspektive auf dem 'Client' (in diesem Fall der Cúram-Anwendungsserver) befindet, nicht auf dem 'Server' (in diesem Fall der Server für generische Suche).

Die Suchcontroller-API besteht aus drei Methoden, die aufgerufen werden können, wenn eine Entität geändert wird, die am Füllen eines Index beteiligt ist. Der Entwickler der Suche muss sich bewusst darüber sein, welche Vorgänge mit Anwendungsentitäten zu solchen Änderungen führen, und im Suchcontroller die entsprechenden Methoden aufrufen. Folgende Methoden sind in dieser API verfügbar:

```
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);
```

Weitere Informationen finden Sie unter 7.6, „Synchronisation zu jeder Suchentität hinzufügen", auf Seite 27.

6.4 SearchServiceConnector

SearchServiceConnector ist eine Dienstprogrammklasse, mit der Suchabfragen durchgeführt werden können. Die Operation 'search' für diese Klasse ist das einzige unterstützte Verfahren, mit dem Entwickler einer Suche einen Suchvorgang in einem Index des Servers für generische Suche aufrufen können.

Diese Klasse steuert im Hintergrund die Details zur Verbindung zwischen der aktiven Anwendung und einer Instanz des Servers für generische Suche unabhängig davon, wo dieser bereitgestellt wird.

Suchabfragen mit SearchServiceConnector können mit der folgenden Methode ausgeführt werden:

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

Anmerkung: Wenn der Suchindex keine Daten enthält, löst er die Ausnahme 'IndexEmptyException' aus. Entwickler, die Suchabfragen implementieren, sollten diese Ausnahme ???angemessen behandeln.

Zum Herstellen einer Verbindung zum Server für generische Suche sind Benutzerqualifikationen erforderlich. Der Connector nimmt die Details des aktuellen Benutzers auf und verwendet sie für die Kommunikation mit dem Server für generische Suche.

Anmerkung: Versuchen Sie nicht, die Methode 'DoSearch' (oder eine andere Methode des Servers für generische Suche) direkt zu verwenden. Sie funktioniert nicht, weil sie im Kontext der Cúram-Anwendung ausgeführt wird, nicht im Kontext einer aktiven Anwendung des Servers für generische Suche.

6.5 Abfragen

Um eine Suche auszuführen, muss ein CuramQuery-Objekt erstellt werden. Die CuramQuery-Klasse umfasst Folgendes:

- Die Suchservice-ID (searchServiceId) des Suchservice, dessen Index Sie durchsuchen wollen. Weitere Informationen zum Konzept von Suchservice finden Sie unter 2.4, „Suchservice", auf Seite 4. Genauere Informationen zum Definieren einer Suchservice-ID (searchServiceId) finden Sie unter 5.2.1, „searchServiceId", auf Seite 13.

- Eine Liste der CuramTerm-Objekte oder ein Attribut 'text', das eine Lucene-Abfragezeichenfolge darstellt. Diese stellen die Suchkriterien dar. Weitere Informationen zu CúramTerms und dem Attribut 'text' finden Sie unten.
- Eine Liste der CuramField-Objekte. Die Werte für diese Felder werden im Rahmen der Suchergebnisse zurückgegeben, jedoch nur, wenn die Felder in der SearchServiceField-Definition als 'Stored' (gespeichert) markiert wurden (siehe 5.3.6, „stored", auf Seite 15).
- Ein Ganzzahlattribut namens 'maxHits', das die maximale Anzahl der für diese Abfrage zurückzugebenden Treffer anzeigt
- Ein boolesches Flag namens 'maxHitsUnbounded', das anzeigt, dass die maximale Trefferanzahl nicht begrenzt ist. Wenn dieses Flag gesetzt ist, wird der Wert des Attributs 'maxHits' ignoriert.

6.6 CuramTerm

???CuramTerm-Objekte (Curam-Begriffe) sind der Teil der CuramQuery-Struktur, der die Suchkriterien darstellt.

Es gibt drei Arten von Begriffen: ???Standardbegriffe, Datumsbegriffe und Datumsbereichsbegriffe. Das CuramTerm-Objekt enthält jeweils einen dieser Typen und gibt mithilfe des Attributs 'termType' an, welcher der Begriffsuntertypen verwendet werden soll. Für jedes CuramTerm-Objekt ist nur ein einziger aggregierter Begriffsuntertyp gültig.

Für alle Begriffstypen wird mit dem Attribut 'field' der Name des Feldes in dem zu durchsuchenden Suchservice angegeben (siehe 2.5, „Feld", auf Seite 5 und 5.3.3, „name", auf Seite 15). Das Attribut 'value' ist als Suchkriterium zu verwenden. Seine Bedeutung richtet sich nach den verschiedenen Begriffstypen und wird nachstehend beschrieben.

6.6.1 Abfragestruktur

Jeder Begriff hat ein Feld mit der Bezeichnung *occurs*. Wie dieses Feld eingestellt wird, bestimmt die Struktur der Abfrage: ob alle Suchbegriffe vorhanden sein müssen, nur ein einziger oder eine andere Kombination. Gültige Werte für *occurs* sind `MUST`, `SHOULD`, `MUST_NOT` und `MUST_FIELD`.

Wird für das Attribut *occurs* für eine Gruppe von Begriffen der Wert `MUST` angegeben, wird nur dann ein Ergebnis zurückgegeben, wenn alle Begriffe gefunden werden. Wird für eine Gruppe von Begriffen der Wert `SHOULD` angegeben, wird ein Ergebnis zurückgegeben, wenn ein oder mehrere Begriffe gefunden werden. Die Mischung dieser Werte in einer Abfrage sollte jedoch vermieden werden, weil sie zu einem nicht definierten Ergebnis führt. Wenn Sie komplexe Abfragen erstellen müssen, die Unterabfragen mit UND und ODER enthalten, müssen Sie das unter 6.6.4, „Text", auf Seite 20 beschriebene Abfrageattribut *text* verwenden.

Wird für das Attribut *occurs* der Wert `MUST_NOT` angegeben, werden nur diejenigen Dokumente zurückgegeben, die nicht mit dem Begriff übereinstimmen. Begriffe, die diesen Wert angeben, können mit Begriffen gemischt werden, die andere Werte für das Attribut *occurs* angeben.

Mit der Option `MUST_FIELD` können Sie eine Unterabfrage erstellen, mit der ein bestimmtes Indexfeld für einen Wert aus einer Wertegruppe getestet wird, d. h. eine ODER-Unterabfrage innerhalb Ihrer Hauptabfrage. Sie sollten diesen Wert für *occurs* für alle Begriffe einstellen, die dieses Feld betreffen, und für jeden gültigen Wert einen Begriff hinzufügen. Begriffe, die `MUST_FIELD` verwenden, können zu einer Gesamtabfrage gehören, in der die Begriffsoption `MUST` oder `SHOULD` verwendet wird.

6.6.2 Standardbegriffe

Ein Standardbegriff wird bei allen Suchabfragen verwendet, die keine Datumsangeben beinhalten. Daher ist dies der Begriffstyp, den Sie am häufigsten verwenden.

Die grundlegendste Verwendung eines Standardbegriffs besteht darin, einfach den Feldnamen und ein einzelnes Token als Wert anzugeben. Der Suchserver gibt Ergebnisse zurück, bei denen der Feldwert exakt mit dem Suchbegriff übereinstimmt.

Eine andere Verwendungsmöglichkeit eines Standardbegriffs besteht darin, einen Wert anzugeben, der mehrere Token enthält, beispielsweise in einer Adresse. Auch dabei gibt der Suchserver Ergebnisse zurück, bei denen der Feldwert exakt mit dem Suchbegriff übereinstimmt.

Wenn der Suchbegriff als einzelnes Token angegeben wird, das ein Platzhalterzeichen enthält, gibt der Suchserver alle übereinstimmenden Ergebnisse zurück. Die unterstützten Platzhalterzeichen sind '*', was einer beliebigen Zeichenfolge entspricht, und '?', was für ein einzelnes Zeichen steht. Beispiel: term = "Dub*"

Ein Standardbegriff kann als Präfixsuche behandelt werden. Das bedeutet, dass nach Suchergebnissen gesucht wird, die die Suchkriterien am Anfang enthalten. Sie geben eine Präfixsuche an, indem Sie das Attribut 'isPrefixSearch' des Standardbegriffs einstellen. Dies hat denselben Effekt wie die Angabe von '*', einem Platzhalter für mehrere Zeichen, am Ende Ihres Suchwerts. Ein Präfixsuchbegriff darf keine anderen Platzhalterzeichen enthalten.

Beispiel 1: Bei einem Standardsuchbegriff mit Token "abc" erfolgt die zugrunde liegende Suche nach dem Begriff "abc*". Bei Suchabfragen mit mehreren Begriffen mit Token und Präfix, z. B. einem Präfixsuchbegriff "abc def", erfolgt die zugrunde liegende Suche nach dem Begriff "abc* def*".

Beispiel 2: Bei einem Standardsuchvorgang mit Token, jedoch ohne Präfix, der mit "abc" anfängt, muss für den Wert des Begriffs "abc*" angegeben werden. Bei einem Standardsuchvorgang mit Token und mehreren Begriffen, jedoch ohne Präfix, der mit "abc" und "def" anfängt, sollte der Wert "abc* def*" angegeben werden.

6.6.3 Datums- und Datumsbereichsbegriffe

Ein Datumsbegriff ähnelt einem Standardbegriff, mit dem Unterschied, dass er für die Suche in Feldern vom Typ 'Date' oder 'DateTime' verwendet wird.

Eine Datumsbereichsbegriff kann für die Suche nach Werten verwendet werden, die zwischen einem Mindestdatum (beginDate) und einem Höchstdatum (endDate) liegen. Mit dem booleschen Attribut 'isExclusive' wird bestimmt, ob das Anfangs- und das Enddatum in die Suchkriterien eingeschlossen ist. Wenn 'isExclusive' auf 'true' (wahr) eingestellt ist, wird die Suche ohne das Anfangs- und Enddatum ausgeführt. Wenn 'isExclusive' auf 'false' (falsch) eingestellt ist, wird die Suche mit dem Anfangs- und Enddatum ausgeführt.

Anmerkung: Wenn eine Abfrage mehrere Begriffe enthält, werden die Ergebnisse zurückgegeben, die mit allen Suchbegriffen übereinstimmen. Derzeit gibt es in der API des Servers für generische Suche kein logisches ODER oder NICHT.

Anmerkung: Beachten Sie, dass Sie bei der Verwendung von Datumsangaben in einer Suche sicherstellen müssen, dass sich das Datum in der Suche auf dieselbe Zeitzone bezieht, die beim Exportieren der Daten in den Suchservice verwendet wurde.

6.6.4 Text

Das Attribut 'text' der CuramQuery-Klasse ist eine Alternative zu einer Gruppe von Begriffen und bietet größere Flexibilität bei der Angabe der Suchkriterien. Es sollte nur verwendet werden, wenn erforderlich, da bei diesem Verfahren auch leichter Fehler in die Suchabfragen eingebaut werden können. Das Format für die Angabe von Suchkriterien mit diesem Attribut wird in der Dokumentation von Lucene beschrieben. Diese finden Sie unter http://lucene.apache.org/java/2_2_0/queryparsersyntax.html.

Sie können Begriffe nicht mit der Zeichenfolge zur Textabfrage kombinieren. Wenn die Zeichenfolge zur Textabfrage vorhanden ist, werden alle CuramTerm-Objekte in der Abfrage ignoriert.

6.7 Abfragen generieren

Die API des Servers für generische Suche enthält eine Dienstprogrammklasse, die Ihnen erlaubt, Curam-Query-Objekte ohne großen Aufwand zu erstellen. Diese Klasse ist `curam.core.impl.util.QueryBuilder`.

6.7.1 Abfragebuilder (QueryBuilder) erstellen

`QueryBuilder` ist keine statische Klasse. Sie müssen für jede von Ihnen generierte Abfrage eine `QueryBuilder`-Instanz erstellen.

Geben Sie mithilfe der Methoden `setUnbounded(boolean unbounded)` und `setMaxHits(long maxHits)` die Anzahl der Treffer an, die die von Ihnen generierte Abfrage zurückgeben sollte.

6.7.2 Suchkriterien hinzufügen

`QueryBuilder` bietet eine Reihe von Methoden im Format `addXXTerm(...parameters...)`, mit denen Sie bequem verschiedene Typen von Suchbegriffen zu Ihrer generierten Abfrage erstellen können. Diese Begriffe werden durch ein logisches UND miteinander verbunden und bilden eine komplexe Abfrage. Diese Methoden werden hier nicht ausführlich beschrieben. Die vollständigen Details sind jedoch im Javadoc zum GSS verfügbar.

6.7.3 Abfragen aus einer Struktur generieren

Wenn Sie eine Cúram-Struktur haben, mit der Sie eine Abfrage generieren wollen, können Sie dazu die folgende Methode verwenden: `setTerms(final Object key)`.

Diese erwartet eine Struktur, in der jedem Attribut *XX* ein boolesches Attribut namens *searchByXX* entspricht, mit dem angegeben wird, ob die Suche mit diesem Attribut durchgeführt werden soll. Es wird angenommen, dass jedes Attribut *XX* einem SearchServiceField-Objekt in Ihrem Suchservice entspricht.

Wenn die Namen der Attribute Ihrer Struktur nicht den Namen entsprechen, die Sie für Ihren Suchservice definiert haben (siehe 2.5, „Feld", auf Seite 5 und 5.3.3, „name", auf Seite 15), können Sie mithilfe einer HashMap namens 'dictionary' eine Zuordnung zwischen ihnen definieren. Die Zuordnung erfolgt von den Attributnamen in der Struktur zu den SearchServiceField-Namen. Fügen Sie die Zeichenfolgepaare einfach zu der HashMap hinzu, wobei der Name des Strukturattributs als Schlüssel und der Feldname als Wert dient. 'dictionary' kann im Konstruktor angegeben werden, wenn Sie Ihr QueryBuilder-Objekt erstellen, oder später mit der Methode `setDictionary(HashMap<String, String>)`.

6.7.4 Zurückzugebende Suchservicefelder angeben

In Ihrer Abfrage können Sie angeben, welches Subset der Felder des Suchservice als Ergebnisse zurückgegeben werden sollen. Häufig ist es zweckmäßig, dass alle Felder zurückgegeben werden. In diesem Fall können Sie folgende Methoden zur Vereinfachung verwenden:

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

6.7.5 Abfrageobjekt abrufen

Verwenden Sie die Methode `getQuery()`, um das generierte CuramQuery-Objekt abzurufen.

6.8 Suchergebnisse bearbeiten

Ähnlich wie die Cúram-Schlüsselstrukturen in `CuramQuery`-Objekte konvertiert werden müssen, sind auch die bei Suchabfragen zurückgegebenen `CuramDocument`-Objekte für die Verwendung in der Anwendung in Cúram-Strukturen zu konvertieren.

Bei der Suchmethode `SearchServiceConnector` werden die Ergebnisse in Form eines `SearchServerResults`-Objekts zurückgegeben. Dieses besteht aus einer Liste von `CuramDocument`-Objekten und jedes `CuramDocument`-Objekt besteht aus einer Liste von `CuramField`-Objekten. Für die Konvertierung zwischen `CuramDocument`-Objekten und Cúram-Strukturen wird eine Dienstprogrammklasse namens `curam.core.impl.util.CuramDocToResultStruct` bereitgestellt.

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

Diese Methode verwendet ein `CuramDocument`-Objekt und eine Strukturinstanz (über den Parameter 'structObj'). Die Methode versucht, für jedes Feld in dem `CuramDocument`-Objekt ein Attribut in der Struktur zu finden, das denselben Namen und Datentyp aufweist. Es wird eine Struktur zurückgegeben, die alle zugeordneten Werte enthält. Diese sollte in eine Struktur des korrekten Typs umgesetzt werden.

Wenn die Namen der Attribute Ihrer Struktur nicht den Namen entsprechen, die Sie für Ihren Suchservice definiert haben (siehe 2.5, „Feld", auf Seite 5 und 5.3.3, „name", auf Seite 15), können Sie mithilfe des Parameters 'dictionary' eine Zuordnung zwischen ihnen definieren. Die Zuordnung erfolgt von den Feldnamen in dem Suchservice zu den Attributnamen in der Struktur. Fügen Sie die Zeichenfolgepaare einfach zu der HashMap hinzu, wobei der Feldname als Schlüssel und der Name des Strukturattributs als Wert dient. Die Konvertierungsfunktion gleicht dann mit dieser HashMap die Feldnamen mit den Attributnamen ab.

Anmerkung: Beachten Sie, dass die Attribute in Ihrer Ergebnisstruktur, deren Namen mit Feldern in Ihrem Dokument übereinstimmen, einfache Cúram-Typen haben müssen, keine aggregierten Strukturen.

6.9 Datentypen und Zeichenfolgekonvertierung

Der Server für generische Suche enthält eine API, mit der durchsuchbare Cúram-Datentypen in Zeichenfolgen konvertiert werden können und umgekehrt. ???Diese müssen möglicherweise manchmal in angepassten Zuordnungsfunktionen oder direkt in Syntaxanalyseergebnissen verwendet werden, anstatt die bereitgestellte Dienstprogrammklasse `curam.core.impl.util.CuramDocToResultStruct` zu verwenden.

Die Umsetzerklasse ist `curam.core.impl.search.datatypes.DataTypeConverter`. Diese Klasse enthält Methoden zur Konvertierung von Cúram-Datentypen in Zeichenfolgen und zur Rückkonvertierung von Zeichenfolgen in Cúram-Datentypen (durch die Übergabe in eine Struktur und die Angabe, welches Attribut in der Struktur eingestellt werden soll).

Kapitel 7. Suchvorgang mit dem Server für generische Suche implementieren

7.1 Übersicht

Dieses Kapitel enthält ein Beispiel für die schrittweise Implementierung einer Suche in der Cúram-Anwendung auf der Grundlage eines Servers für generische Suche. Das hier aufgeführte Beispiel bezieht sich auf eine Personensuche.

Die Implementierung umfasst folgende Schritte:

- DMX-Dateien für Suchservices (SearchService) und Suchservicefelder (SearchServiceField) schreiben
- Zuordnungsfunktionsschnittstelle implementieren
- Suchrouting und Aufruffunktion implementieren
- Synchronisation von Anwendungsoperationen zur Suche von Entitäten hinzufügen (oder die Zuordnungsfunktion für Extrahierung verwenden, siehe Kapitel 8, „Zuordnungsfunktion für Extrahierung", auf Seite 29)
- Benutzerschnittstelle und Fassade für die Suche erstellen. Dies ist die normale Anwendungsentwicklung.

7.2 Beispiel für Personensuche - Übersicht

Es ist wichtig, darauf hinzuweisen, dass die Benutzer des Cúram-Servers für generische Suche keinen Funktionsunterschied zwischen ihren Suchabfragen und den mit SQL implementierten Suchabfragen des Servers bemerken sollten. Darüber hinaus können die Anzeigen sowie die allgemeine Benutzererfahrung beibehalten werden. Daher wird bei dem folgenden Beispiel angenommen, dass die Leser eine solche Anwendungsfunktionalität (sowie die entsprechenden Fassadenklassen etc.) in der üblichen Weise entwickeln.

Im vorliegenden Beispiel für Personensuche navigieren die Benutzer zu der relevanten UIM-Seite und führen dort eine Personensuche aus. Auf dieser Seite tragen Sie ein oder mehrere Suchkriterien ein. Wenn sie auf die Schaltfläche 'Suchen' klicken, wird die Suche ausgeführt. Die Ergebnisse umfassen eine Liste mit den Einträgen, die mit den Suchkriterien übereinstimmen.

Bei Anwendungssuchabfragen ist es üblich, dass die Suchkriterien und die Details, die in der Ergebnisliste zurückgegeben werden, aus mehreren zusammengehörigen Entitäten sortiert werden. Bei der Personensuche werden folgende Entitäten und deren Attribut als Suchkriterien verwendet oder als Ergebnisfelder zurückgegeben:

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

Jede dieser Entitäten wird durch einen Fremdschlüssel zugeordnet. Somit ist 'concernRoleID' der externe Schlüssel des Suchserviceattributs für den Suchservice zur Personensuche (siehe 5.2, „Tabelle 'SearchService'", auf Seite 13).

Die folgenden Attribute werden bei der Suche verwendet, entweder im Rahmen der Suchkriterien oder als anzeigbarer Teil der Ergebnisliste:

- referenceNumber
- forename

- surname
- address
- city
- dateOfBirth
- sex
- birthSurname
- motherSurname

Diese Felder werden beim Suchservice zur Personensuche in der Tabelle 'SearchServiceField' gespeichert.

7.3 DMX-Dateien für Suchservices entwickeln

7.3.1 Suchserviceeinträge einrichten

Weitere Informationen finden Sie unter B.1, „Suchserviceeintrag", auf Seite 49 und 5.2, „Tabelle 'SearchService'", auf Seite 13.

7.3.2 Suchservicefeldeinträge einrichten

Weitere Informationen finden Sie unter B.2, „Suchservicefeldeintrag", auf Seite 50 und 5.3, „Tabelle 'SearchServiceField'", auf Seite 14.

7.4 Operationen der Zuordnungsfunktion implementieren

Eine Einführung in Zuordnungsfunktionen finden Sie unter 2.12, „Zuordnungsfunktion", auf Seite 6 und 6.2, „Zuordnungsfunktionen", auf Seite 17.

In den folgenden Abschnitten wird die Implementierung der Methoden der Zuordnungsfunktionsschnittstelle für die einzelnen Suchservices beschrieben. Für jede Schnittstellenmethode wird ein Beispiel für einen Suchservice zur Personensuche (PersonSearch) bereitgestellt. Auch ein umfassendes Javadoc ist für die Zuordnungsfunktionsschnittstelle verfügbar. Dieses sollte von alle Entwicklern gelesen werden, die einen Suchservice implementieren.

7.4.1 Schnittstelle '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;
```

Diese Methode wird während des Batchprozesses der Datenbankextraktion aufgerufen. Für jeden Suchservice wird 'mapToStagingDb' aufgerufen, um Informationen aus den Quellenentitäten abzurufen und zum Batchprozess zurückzugeben.

Es muss eine Cúram-Operation des Typs 'ReadmultiOperation' geschrieben werden, um alle Einträge zu verarbeiten, die für die einzelnen Suchservices in der Staging-Datenbank gespeichert werden sollen. Für jeden dieser Einträge muss eine Operation des Servers für generische Suche mit der Bezeichnung 'ExtractReadMultiOperation' aufgerufen werden. Diese Operation ermittelt intern, welche anderen Entitäten er-

forderlich sind, um eine Tabelle 'SearchServiceRow' auf der Grundlage dieser Daten vollständig zu füllen, und erstellt außerdem ein 'SearchServiceRow'-Objekt.

Das Ergebnis dieses ganzen Prozesses ist einfach eine Liste von Suchservicezeilen (SearchServiceRows), die sämtliche Anfangsdaten darstellen, mit denen die Staging-Datenbank gefüllt werden soll. Dann werden diese Zeilen mithilfe des Batchprozesses der Datenbankextraktion in die Staging-Datenbank eingefügt.

7.4.2 Schnittstelle '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;
```

Wie oben erwähnt, können Daten in einem Suchservice aus einer Reihe verschiedener Entitäten erfasst werden. Außerdem können diese Entitäten durch komplexe Fremdschlüsselbeziehungen zusammengehören (beispielsweise könnte ein Adresseintrag über die Entität 'addressID' mit einem Personeneintrag verbunden sein, die über die Entität 'concernRoleAddressID' verknüpft ist, welche wiederum über die Entität 'concernRoleID' verknüpft ist).

Durch die Aktualisierung einer dieser Entitäten mithilfe der Anwendung werden diese Beziehungen komplexer. Wenn das geschieht, muss der Server für generische Suche ermitteln können, welche Entität gerade betroffen ist, an welchen Suchabfragen sie beteiligt ist und wie sie mit allen anderen in den einzelnen Suchservices enthaltenen Entitäten verbunden ist.

Schließlich müssen ein oder mehrere Dokumente in einem oder mehreren Suchserviceindizes aktualisiert werden und die Angaben in diesen Dokumenten können aus einer Reihe von Entitäten erfasst werden, nicht nur aus der gerade geänderten Entität. Da jeder Suchservice jedoch nur eine einzige Zuordnungsfunktion besitzt, brauchen die einzelnen Implementierungen der Zuordnungsfunktion nur die Angaben zu ihrem eigenen Suchservice zusammenzustellen.

Die Schnittstellenmethode 'getObjectList' befasst sich mit diesem Problem. Bei einem einzelnen aktualisierten Entitätseintrag stellt 'getObjectList' alle anderen ???Dtls-Entitätseinträge zusammen, die für die Aktualisierung des entsprechenden Dokuments im aktuellen Suchserviceindex erforderlich sind. Die Methode 'getObjectList' muss so codiert werden, dass jede der an einem Suchservice beteiligten Entitäten als Startpunkt dieses Prozesses verwendet werden kann. 'getObjectList' hat folgende Aufgaben:

- Die an die Methode übergebene Entität ermitteln
- Alle zugehörigen Entitäten für den betroffenen Suchservice ermitteln
- Alle zugehörigen Entitätseinträge auf der Grundlage der Daten in der Parameterentität lesen und zusammenstellen

Die Methode 'mapper.getobjectList ()' wird bei folgenden Prozessen aufgerufen:

- Database Synchronization insert
- Database Synchronization modify
- Erste Datenbankextraktion

Beachten Sie, dass bei der ersten Datenbankextraktion die Schnittstellenmethode 'getObjectList' bei jedem aus 'ReadmultiOperation' abgerufenen Element aufgerufen wird. In der Regel ist dies die Entität der

obersten Ebene für den vorliegenden Fall (Beispiel: Bei der Extraktion einer Personensuche werden alle Personeneinträge in einem readmulti-Vorgang gelesen. Danach wird 'getObjectList' für jeden Eintrag aufgerufen, um alle übrigen Informationen abzurufen, die für die Erstellung von 'SearchServiceRow' erforderlich sind).

Wenn diese Methode bei einer Eingabe aufgerufen wird, die für diesen Suchservice nicht relevant ist, sollte die Implementierung einfach eine leere Liste zurückgeben.

7.4.3 Schnittstelle '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) ;
```

Die Schnittstellenmethode `getExtKey` gibt eine eindeutige Kennung für den angegebenen Suchservice zurück. Dieser Schlüssel wird als Schlüssel für die einzelnen Zeilen in der Tabelle `SearchServiceRow` der Staging-Datenbank verwendet. Beachten Sie, dass der Parameter 'objList' die Ausgabe der oben beschriebenen Schnittstellenmethode 'getObjectList' ist. Beispiel: Beim Aufruf von `getExtKey` für den Suchservice 'PersonSearch' sollte die ID 'concernRoleID' des fraglichen Eintrags zurückgegeben werden.

Wenn diese Methode bei Daten aufgerufen wird, um die sich der Suchservice nicht kümmert, sollte 'Null' zurückgegeben werden.

7.4.4 Schnittstelle '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;
```

Löscht das angegebene Zeilenobjekt aus der Staging-Datenbank.

7.4.5 Schnittstelle '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);
```

Die Infrastruktur des Servers für generische Suche versucht, mithilfe der aus der Tabelle 'SearchServiceField' abgerufenen Feldmetadaten einen Entitätsattributwert aus einer Objektliste abzurufen. In der Regel

enthalten Objektlisten ???dtls-Entitätsstrukturen. In diesen Fällen verwendet der Server für generische Suche einfach die ???reflection, um das korrekte Attribut zu ermitteln und dessen Wert abzurufen. Genau das erfolgt im Hintergrund.

Wenn die Objektliste jedoch etwas anderes als eine ???dtls-Entitätsstruktur enthält (wie bei einer Personensuche, wo ein Element 'AddressElementDtlsList' vorhanden ist, das seinerseits eine einzelne Address-Element-Struktur enthält), sollten Entwickler der Suche die Schnittstellenmethode 'Mapper.getFieldValue' implementieren.

Die Schnittstellenmethode 'Mapper.getFieldValue' sollte implementiert werden, wenn eine Zuordnungsfunktion einen bestimmten Attributwert nicht automatisch zuordnen kann. Der entsprechende Entitäts- und Feldname wird über den Strukturparameter 'fieldDtls' übergeben und der Attributwert kann mithilfe von ???reflection aus der Objektliste abgerufen werden. Es ist die Entscheidung des Entwicklers der Suche, diese Methodenschnittstelle für den bzw. die betroffenen Typen zu implementieren

Von dieser Methode sollten keine leeren Zeichenfolgen zurückgegeben werden, sondern stets 'Null'.

7.4.6 Zuordnungsfunktion 'newInstance()'

Wenn die Zuordnungsfunktion modelliert wird, sollte die Factory-Klasse für die Eigenschaft 'mapperName' des Suchservice angegeben werden. Wird die Zuordnungsfunktion NICHT modelliert, muss die Implementierung der Zuordnungsfunktion eine Schnittstelle

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

implementieren, die eine neue Instanz der Zuordnungsfunktion des entsprechenden Suchservice zurückgibt. In diesem Fall ist die Eigenschaft 'mapperName' des Suchservice der Klassenname dieser Implementierungsklasse.

7.5 Suchrouter und Implementierung

Wie oben erwähnt, wird derzeit SQL bei Suchabfragen verwendet. In künftigen Versionen wird bei Plattform- und Lösungssuchabfragen wahrscheinlich immer häufiger der Server für generische Suche als Suchverfahren verwendet. Dennoch wird auch die Suche per SQL wahrscheinlich weiter in der derzeitigen Form unterstützt, sowohl mit Blick auf den Upgradeschutz als auch mit Blick auf die Möglichkeit zum Fallback/Failover bei Netzfehlern oder anderen Implementierungsproblemen.

Dazu sollte die Factory-Klasse 'Suchrouter' implementiert werden. Diese sollte auf der Grundlage einer Eigenschafteneinstellung einen Verweis auf die Implementierung der Datenbanksuche oder die auf dem Server für generische Suche basierende Implementierung zurückgeben.

7.6 Synchronisation zu jeder Suchentität hinzufügen

Wie bereits angemerkt, muss die Staging-Datenbank des Servers für generische Suche aktualisiert werden, wenn an Entitäten Änderungen vorgenommen werden, die zum Suchservice gehören. Eine einzelne Entität kann in mehreren Suchservices verwendet werden und jeder dieser Suchservices muss Änderungen an dieser Entität widerspiegeln.

Die Klasse *SearchController* muss sicherstellen, dass alle Informationen der Staging-Datenbank aktuell sind. Wenn der entsprechende Vorgang der Suchserviceentität ausgeführt wird, müssen die Methoden *insert*, *modify* und *remove* der Klasse *SearchController* von der Anwendung aufgerufen werden. Mit den Vorgängen *insert* und *modify* des Suchcontrollers werden die detaillierten Strukturdaten zu der angegebenen Entität in die Tabelle 'SearchServiceRow' eingefügt. Für die Schnittstelle `remove` ist ein Schlüssel erforderlich, mit dem das zu entfernende Entitätsobjekt und der Name der Entität angegeben werden.

```
/**
 * Generic insertion 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 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
```

Kapitel 8. Zuordnungsfunktion für Extrahierung

8.1 Einführung

Im vorigen Kapitel wurde der Ereignismechanismus beschrieben und erläutert, wie Ihre Daten mit seiner Hilfe mit Ihrem Suchservice synchronisiert bleiben können. Jetzt bietet der Server für generische Suche ein anderes Verfahren, mit dem die Aktualität Ihres Suchservice gewährleistet wird, nämlich die Zuordnungsfunktion für Extrahierung. In diesem Kapitel wird beschrieben, wie die Zuordnungsfunktion für Extrahierung funktioniert und wie sie bei von Ihnen entwickelten Suchabfragen eingesetzt werden kann.

8.2 Zuordnungsfunktion für Extrahierung - Übersicht

Der Ereignismechanismus ist bei weitem die effizienteste Methode, um Ihre Suchservices auf dem neuesten Stand zu halten. Wenn Ihre Suchabfragen jedoch komplex sind, kann es sehr umständlich sein, einen Suchservice zu entwickeln und vollständig zu testen. Dieses Problem wird mit der Zuordnungsfunktion für Extrahierung gelöst.

Die Zuordnungsfunktion für Extrahierung sucht mithilfe von Zeitmarken in Anwendungseinträgen nach Einträgen, die seit der letzten Ausführung der Zuordnungsfunktion für Extrahierung erstellt oder aktualisiert wurden. Wenn sie solche Einträge findet, übergibt sie diese an den Suchcontroller, damit die Suchservices aktualisiert werden. Ab diesem Moment ist der Prozess mit dem Standardereignismechanismus identisch. Für diesen Prozess müssen alle an einem Suchservice beteiligten Datenbanktabellen durchsucht werden, was natürlich Datenbankressourcen erfordert. Im Wesentlichen nimmt die Zuordnungsfunktion für Extrahierung gewisse Einbußen bei der Laufzeitleistung in Kauf, um die Entwicklung von Suchabfragen zu beschleunigen und zu vereinfachen.

8.3 Entwicklung mit der Zuordnungsfunktion für Extrahierung

In diesem Abschnitt werden die einzelnen Prozessschritte bei der Entwicklung eines Suchservice mit der Zuordnungsfunktion für Extrahierung erläutert.

8.3.1 'Zuletzt aktualisiert' für Ihre durchsuchbaren Entitäten aktivieren

Für alle Datenbankentitäten, die an Suchservices beteiligt sind und eine Zuordnungsfunktion für Extrahierung verwenden, sind Zeitmarken erforderlich. Die Zeitmarkenspalten werden durch die Infrastruktur automatisch hinzugefügt und aktuell gehalten, wenn Sie für die Entität in dem Modell die Funktion 'Zuletzt aktualisiert' aktivieren. Wie diese Funktion aktiviert wird, ist im Handbuch `Server Modelling Guide` dokumentiert.

8.3.2 Tabellensuche modellieren

Eine andere Anforderung der Zuordnungsfunktion für Extrahierung für die Modellierung einer Operation namens `searchByLastwritten` (Groß-/Kleinschreibung ist zu beachten).

Dies sollte eine nsmulti-Operation sein. Der Wert für nicht generiertes SQL sollte 'Nein' lauten. Die Operation sollte eine Struktur namens *key* verwenden. Sie können Ihre eigene Struktur als Parameter modellieren, sie muss jedoch ein Attribut namens *datetime* besitzen, das ein `DateTime`-Objekt darstellt. Später geben Sie den Klassennamen dieser Struktur in der Tabelle 'GSSEntity' an, wie unten beschrieben.

Sie müssen SQL für die Operation bereitstellen. Hier ein einfaches Beispiel für eine einfache Entität namens 'Customer':

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

Sie müssen darauf achten, alle vom Suchservice verwendeten Spalten auszuwählen.

Neben der Tabellensuchmethode müssen Sie über eine Standardlesemethode für alle durchsuchbaren Entitäten verfügen.

8.3.3 Suchservice definieren

Ihr Suchservice sollte in der üblichen Weise definiert werden (siehe Kapitel 7, „Suchvorgang mit dem Server für generische Suche implementieren", auf Seite 23.

Neben den Tabellen 'SearchService' und 'SearchServiceField' müssen Sie Definitionen zu den Tabellen 'GSSMapperType' und 'GSSEntity' hinzufügen.

8.3.3.1 GSSMapperType

Diese Tabelle ordnet einfach den Namen des Suchservice einer Zeichenfolge zu, die den Typ der Zuordnungsfunktion definiert. Standardmäßig wird die Standardereigniszuordnungsfunktion verwendet, die nicht angegeben werden muss. Um bei einem bestimmten Suchservice die Zuordnungsfunktion für Extrahierung zu verwenden, sollte eine Zeile zu dieser Tabelle hinzugefügt werden, die den Suchservicenamen dem Zuordnungsfunktionstyp „PULL" zuordnet.

searchServiceId

Die Suchservice-ID. Eine Zeichenfolge für die eindeutige Identifizierung eines Suchservice. Dies ist ein Fremdschlüssel der Tabelle 'SearchService'.

mapperType

Stellen Sie hier den Wert 'PULL' (in Großbuchstaben) ein, um für den Suchservice die Zuordnungsfunktion für Extrahierung zu aktivieren.

8.3.3.2 GSSEntity

Wenn die Zuordnungsfunktion für Extrahierung verwendet wird, benötigt der GSS weitere Informationen zu den Entitäten, die gerade in den Suchservices verwendet werden. Für jede eindeutige in den untergeordneten Suchservicefeldeinträgen aufgeführte Entität, die zu den einzelnen Suchservices gehört und die Zuordnungsfunktion für Extrahierung verwendet, muss ein GSSEntity-Eintrag hinzugefügt werden (wenn jedoch mehrere Felder zu derselben Einheit gehören, brauchen Sie die Angaben nicht mehrmals einzugeben).

searchServiceId

Die Suchservice-ID. Eine Zeichenfolge für die eindeutige Identifizierung eines Suchservice. Dies ist ein Fremdschlüssel der Tabelle 'SearchService'.

tblScanKeyStruct

Der vollständige Klassenname der Struktur, die als Parameter für Ihre modellierte Methode `searchByLastwritten` dient, die unter 8.3.2, „Tabellensuche modellieren", auf Seite 29 beschrieben wird.

entityKeyStruct

Der vollständige Klassenname des Parameters 'struct' für das Leseverfahren Ihrer Entität.

EntityFactClass

Der vollständige Klassenname der generierten Factory-Klasse für Ihre Entität.

8.3.4 Zuordnungsfunktionsklasse schreiben

Eine SearchServiceMapper-Implementierung mit der Zuordnungsfunktion für Extrahierung entspricht in vielen Punkten einer standardmäßigen SearchServiceMapper-Implementierung, wie im Kapitel "Suchvorgang mit dem Server für generische Suche implementieren" dieses Handbuchs beschrieben. Es müssen jedoch einige weitere Punkte berücksichtigt werden.

Bei der Verwendung der Zuordnungsfunktion für Extrahierung mit einem komplexen Suchservice, der aus mehreren zusammengehörigen Entitäten besteht, müssen Sie sicherstellen, dass Ihre SearchService-Mapper-Implementierung sich ordnungsgemäß verhält, wenn Sie mit unvollständigen Gruppen von Entitäten zu tun haben. Wenn beispielsweise die Entitäten A, B und C gemeinsam einen Suchservice bilden, könnte Ihre Zuordnungsfunktion einen Aufruf erhalten, wenn nur A und C vorhanden sind. Je nach Ihrem Suchservice könnte das korrekte Verhalten darin bestehen, die unvollständige Datengruppe zum Suchservice hinzuzufügen oder nichts zu tun, bis die Gruppe vollständig ist.

8.4 Löschvorgänge

Die Zuordnungsfunktion für Extrahierung kann keine Standardlöschvorgänge vornehmen. Wenn Sie über eine durchsuchbare Entität verfügen, die gelöscht werden kann, müssen Sie diesen Vorgang mit einem anderen Verfahren ausführen (beispielsweise mit dem in diesem Handbuch beschriebenen ereignisbasierten Verfahren).

Die Zuordnungsfunktion für Extrahierung ist jedoch in der Lage, standardmäßige logische Löschvorgänge vorzunehmen, bei denen beispielsweise eine 'recordStatus-Spalte mit Werten aus der Codetabelle `RecordStatus` eingestellt wird.

Kapitel 9. Suchvorgänge und Abfragen im Detail

9.1 Einführung

Wie jede Software müssen Ihre GSS-basierten Suchabfragen bestimmte konzeptionelle Einschränkungen beachten, damit sie zufriedenstellend ausgeführt werden und entsprechend den Erwartungen der Benutzer funktionieren. In diesem Kapitel werden der Entwurfsprozess einer GSS-Suche und die ordnungsgemäße Verwendung von GSS-Abfragen detailliert beschrieben.

9.2 Suchservice - allgemeine Richtlinien

Als Erstes müssen Sie beim Entwurfsprozess entscheiden, welche Daten durchsuchbar sein sollen. Welche Felder sollen durchsuchbar sein? Welche Daten soll Ihre Suche zurückgeben? Da es dabei mehrere Kompromisse gibt, lohnt es sich, sorgfältig über diese Dinge nachzudenken.

Erstens sollte Ihr Index so wenige Felder wie möglich enthalten. Weniger Felder bedeuten einen kleineren Index zur Laufzeit und weniger belegte Systemressourcen. Binden Sie nichts in den Suchservice ein, was nicht benötigt wird.

Jedes Feld in Ihrem Index kann indexiert und/oder gespeichert werden (d. h. durchsuchbar sein und/oder Sie können seinen Wert abrufen). Die Gründe für die Indexierung eines Feldes liegen auf der Hand: Sie wollen auf seiner Grundlage Suchabfragen durchführen. Allerdings wollen Sie in bestimmten Feldern möglicherweise keine Suchabfragen durchführen, beispielsweise in von Menschen nicht lesbaren IDs. Sie könnten solche Felder als gespeicherte, jedoch nicht indexierte Felder zu Ihrem Suchservice hinzufügen, um Datenbanksuchvorgänge auf der Basis der Ergebnisse Ihrer Suchabfragen ausführen zu können. Wenn Sie ein Feld nicht indexieren müssen, tun Sie es nicht. Dann laufen Ihre Extraktionsprozesse schneller ab und Ihr Index verbraucht weniger Systemressourcen.

Analog dazu könnten Sie entscheiden, ob Feldwerte gespeichert werden sollen oder nicht. Im Allgemeinen speichert der Index nicht den ursprünglichen Wert eines Feldes, sondern behält nur eine durchsuchbare Darstellung bei. Generell gilt: Damit die Suche nützlich ist, muss sie mindestens ein Feld speichern (den entsprechenden Primärschlüssel des Datenbanksatzes).

Danach ist es ein Kompromiss, ob Felder gespeichert werden oder nicht. Sie könnten alle benötigten Felder speichern, um die Suchergebnisse anzuzeigen, oder Sie könnten nur die Datenbank-IDs speichern und mit ihrer Hilfe die anzuzeigenden Daten aus der Datenbank abrufen. Die erste Option führt zu einem viel größeren Index, bewirkt jedoch eine schnellere Anzeige der Suchergebnisse, da die Datenbank nicht erforderlich ist.

9.3 Datenbankstruktur einem Index zuordnen - Denormalisierung

Möglicherweise wollen Sie Daten aus verschiedenen Entitäten in Ihre Suche einbinden. Im Gegensatz zur Datenbanksuche werden Suchvorgänge mit Indizes nicht mithilfe von Joins durchgeführt. Denken Sie daran, dass der Hauptvorzug bei der Verwendung eines Index darin liegt, dass die Sucharbeit im Wesentlichen vorab geschieht, wenn der Index erstellt wird, nicht wenn die Suche aufgerufen wird. Dementsprechend sollten für die Indexierung alle Datenbanktabellen denormalisiert werden. Die Alternative, nämlich separate Indizes zu erstellen, sie separat zu durchsuchen und dann zu versuchen, die Ergebnisse zusammenzuführen, ist weitaus komplexer und ineffizient.

Beispiel: Angenommen, Sie haben folgende Entitäten: Entität 'Person' mit den Attributen 'Name' und 'Geburtsdatum' und mit einem Fremdschlüssel, der auf eine Entität 'Adresse' mit den Attributen 'Hausanschrift', 'Ort' und 'Land' verweist. Sie wollen eine Suchabfrage erstellen, die Ihnen die Suche von Perso-

nen nach Name, Geburtsdatum, Hausanschrift, Ort und Land ermöglicht. Dazu müssten Sie einen durchsuchbaren Index erstellen, der sämtliche Daten aus beiden Tabellen enthält.

Wenn Sie mehrere Entitäten haben, die an einem einzelnen Suchindex beteiligt sind, müssen Sie beachten, dass Aktualisierungen der betroffenen Tabellen bewirken können, dass der Suchindex aktualisiert werden muss.

9.4 Felder mit und ohne Token

Es wurde bereits kurz erwähnt, dass Suchfelder in Tokens zerlegt werden können. Dabei werden im Wesentlichen die indexierten Daten in Einheiten zerlegt, die als Tokens bezeichnet werden. Dies geschieht mithilfe eines Analyseprogramms. Analyseprogramme weisen ein unterschiedliches Verhalten auf, einige zerlegen Tokens bei Leerzeichen, andere bei Interpunktionszeichen etc. Außerdem werden die so generierten Tokens normalerweise in Kleinbuchstaben umgewandelt. Bei in Tokens zerlegten Feldern werden die Abfragezeichenfolgen in derselben Weise zerlegt, sodass bei Suchabfragen - neben anderen Vorteilen - keine Groß-/Kleinschreibung beachtet werden muss.

Bei bestimmten Feldern ist die Zerlegung in Tokens jedoch nicht sinnvoll. Gute Beispiele dafür sind computergenerierte Werte wie Codes von Codetabellen. Im Allgemeinen sollten jedoch die meisten Ihrer Felder in Tokens zerlegt werden. Besonders das Verhalten von Feldern und Suchabfragen ohne Token, die mehrere Wörter enthalten, widerspricht der Intuition. Ziehen Sie diesen Fall in Betracht, wenn Sie den merken, dass Ihre Suchabfragen nicht die erwarteten Daten zurückgeben.

Nehmen Sie beispielsweise ein Adressfeld mit einem Dokument, das "Joyce Way Parkwest Dublin" enthält. Wenn dieses Feld in Tokens zerlegt ist und das standardmäßige Analyseprogramm enthält, enthält der Index vier Begriffe: "joyce", "way", "parkwest" und "dublin". Jede Abfragezeichenfolge, die Begriffe enthält, die mit diesen Begriffen (genau oder über einen Platzhalter) übereinstimmen, finden dieses Dokument. Beispiele: "Dublin", "Joyce Way", "park*" etc.

Wenn dieses Feld dagegen nicht in Tokens zerlegt ist und dasselbe Dokument hinzugefügt wird, enthält der Index nur einen einzigen Begriff: "Joyce Way Parkwest Dublin". Damit stimmen viel weniger Abfragezeichenfolgen überein, eigentlich nur die Zeichenfolge selbst oder der erste Teil der Zeichenfolge als Präfixsuche. Außerdem muss bei der Suche die Groß-/Kleinschreibung beachtet werden.

9.5 Platzhalter

Der GSS unterstützt Platzhalter, die für ein einzelnes oder für mehrere Zeichen stehen. Das Fragezeichen „?” steht für ein beliebiges einzelnes Zeichen. Der Stern „*” steht für eine beliebige Zeichenfolge. Keines dieser beiden Zeichen sollte am Anfang eines Suchbegriffs stehen, da dies eine schlechte Suchleistung zur Folge hätte. Beim Implementieren einer Suche sollte die Entwickler berücksichtigen, ob den Benutzern gestattet werden soll, diese Zeichen bei Suchabfragen einzugeben, und gegebenenfalls eine nützliche Onlinehilfe bereitstellen. Andernfalls können sie durch ein Escapezeichen unwirksam gemacht werden: „\”. Außerdem könnte es nützlich sein sicherzustellen, dass diese Zeichen nicht am Anfang von Suchbegriffen stehen, und dem Benutzer eine konkretere Fehlernachricht zurückzugeben, als dies die GSS-Infrastruktur kann (Es wird eine generische Ausnahme zurückgegeben, die anzeigt, dass die Abfrage ungültig ist, aber der Entwickler, der die Suche implementiert, kann genauere Informationen dazu machen, welches Feld ungültig ist).

9.6 Analyseprogramme im Detail

Die bereits vorgestellten Analyseprogramme bereiten Ihren durchsuchbaren Text für die Indexierung und für Suchvorgänge vor.

Es ist sehr wichtig, welche Analyseprogramme Sie auswählen. Analyseprogramme sind konkrete Klassen, die die Klasse 'org.apache.lucene.analysis.Analyzer' erweitern. Im Lieferumfang des GSS sind mehrere Analyseprogramme enthalten. Sie können aber auch eigene Analyseprogramme erstellen und verwenden.

Manchmal, wenn Sie versucht sind, ein Feld als in Tokens zerlegt zu definieren, sollten Sie stattdessen die Wahl Ihres Analyseprogramms sorgfältiger überdenken.

Jeder Suchservice besitzt ein Standardanalyseprogramm. Jedes Suchservicefeld kann dieses Analyseprogramm jedoch überschreiben und ein bestimmtes Analyseprogramm für die die Verwendung mit diesem Feld definieren (siehe 5.3.9, „analyzerName", auf Seite 16). Der GSS verwendet für die Indexierung und für Suchvorgänge dasselbe Analyseprogramm.

Der Server für generische Suche bietet folgende vordefinierte Analyseprogramme.

LUCENESTANDARD
Teilt Text bei Interpunktionszeichen, wobei die Interpunktion entfernt wird. Allerdings wird ein Punkt, auf den kein Leerzeichen folgt, als Teil eines Tokens betrachtet. Teilt Wörter bei Bindestrichen, sofern in dem Token keine Zahl enthalten ist. In diesem Fall wird das gesamte Token als Produktnummer interpretiert und nicht geteilt. Erkennt E-Mail-Adressen und Internethostnamen als ein Token. Normalisiert Tokentext zu Kleinbuchstaben und entfernt gebräuchliche englische Stoppwörter.

STANDARD
Ähnelt dem Analyseprogramm LUCENESTANDARD. Allerdings werden Stoppwörter aus den in Tokens zerlegten Begriffen entfernt und wenn es sich bei dem in Tokens zu zerlegenden Inhalt um eine einzelne Zahl handelt, wird dieser nicht geändert (so lassen sich generierte Infrastruktur-IDs verarbeiten, die negative Zahlen sein können).

SIMPLE
Teilt Text bei Zeichen, die keine Buchstaben sind, und normalisiert Tokentext zu Kleinbuchstaben.

STOP Teilt Text bei Zeichen, die keine Buchstaben sind, normalisiert Tokentext zu Kleinbuchstaben und entfernt gebräuchliche englische Stoppwörter.

WHITESPACE
Teilt Text bei Leerzeichen. Angrenzende Folgen von Zeichen ohne Leerzeichen bilden Tokens.

KEYWORD
Bildet aus dem gesamten Datenstrom ein einziges Token. Dies ist für Daten wie Postleitzahlen, IDs und bestimmte Produktnamen nützlich.

Beachten Sie, dass die Klasse im Klassenpfad des Servers für generische Suche verfügbar sein muss, wenn Sie ein anderes Analyseprogramm als ein vordefiniertes Analyseprogramm oder als die im Lieferumfang von Lucene enthaltenen Analyseprogramme verwenden.

Kapitel 10. Server für generische Suche in Eclipse ausführen

10.1 Einführung

In diesem Kapitel wird beschrieben, wie die Entwicklungsumgebung zu konfigurieren ist, um den Server für generische Suche zu Entwicklungs- und Testzwecken in der Eclipse-IDE auszuführen.

Der Server für generische Suche kann zu Entwicklungszwecken im RMI-Modus ausgeführt werden, ähnlich wie die Cúram-Anwendung selbst. In diesem Kapitel wird detailliert beschrieben, welche Einstellungen dafür erforderlich sind.

10.2 Bootstrap.properties

Vor Beginn der Entwicklung sollten die relevanten Einstellungen bei Bedarf zu Ihrer Datei `Bootstrap.properties` hinzugefügt werden. Unter A.1, „Konfigurationseigenschaften", auf Seite 47 finden Sie eine Beschreibung der Konfigurationseigenschaften.

10.3 Cúram-Server für generische Suche aus Eclipse starten

Wie für die Cúram-Anwendung ist auch für den Server für generische Suche im Entwicklungsmodus ein Prozess namens 'tnameserv' erforderlich, damit die Ausführung auf Ihrem System erfolgen kann.

In Ihrer Entwicklungsinstallation finden Sie folgende Implementierungs-Java-Klassendateien in Ihrer Datei `/CuramSDEJ/lib/gss.jar` in Eclipse:

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

Sie sollten in der Lage sein, die beiden obigen Klassendateien im Kontext des EJBServer-Projekts in der üblichen Weise als normale Java-Anwendung auszuführen:

- Klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie **???Als Java-Anwendung ausführen** aus.

Führen Sie den Prozess 'DataBaseSearchExtractor' aus, um Ihre Staging-Datenbank vor dem Start des Suchservers zu erstellen. Führen Sie den Prozess 'StartSearchServer' immer dann aus, wenn Sie zum Testen der Suchfunktionalität eine Suchserverinstanz ausführen müssen. Wenn Sie Ihre Anwendungsdatenbank neu erstellt haben, sollten Sie 'DataBaseSearchExtractor' vor dem Start Ihres Suchservers erneut ausführen.

Anmerkung: Wenn einer Ihrer Suchservices angepasste Analyseprogramme oder Analyseprogramme anderer Anbieter verwendet (Analyseprogramme, die nicht im Lieferumfang des Lucene-Produkts enthalten sind), müssen Sie sicherstellen, dass diese zum Klassenpfad des Projekts `EJBServer` hinzugefügt werden.

Kapitel 11. Server für generische Suche bereitstellen

11.1 Einführung

In diesem Kapitel wird beschrieben, wie der Cúram-Server für generische Suche auf Ihrem Anwendungsserver bereitgestellt wird. Dieses Kapitel richtet sich an Administratoren, die für die Bereitstellung des Suchservers und der Cúram-Anwendung zuständig sind und mit dem ???entsprechenden Cúram-Entwicklungshandbuch vertraut sind.

11.2 Bereitstellungsoptionen

Sie können den GSS in seiner eigenen EAR-Datei oder als Teil der EAR-Datei von Cúram bereitstellen. Die Datei EJBServer/project/config/deployment_packaging.xml enthält die Option requireSearchServer, mit der der GSS eingebunden wird. Wenn Sie diese Option einstellen und die Cúram-EAR-Datei erstellen, brauchen Sie den GSS nicht mit einer separaten EAR-Datei bereitzustellen (Ihr Anwendungsserver lässt dies auch nicht zu). Im Allgemeinen empfiehlt sich eine solche Konfiguration nicht, da sie keine besonders leistungsstarke Bereitstellungskonfiguration darstellt. Sie kann jedoch für Testzwecke oder kleine Bereitstellungen nützlich sein.

11.3 Bereitstellungsprozess

Der Bereitstellungsprozess umfasst folgende Schritte:

- Stellen Sie in Ihrer Datei 'Bootstrap.properties' Ihre Konfigurationseigenschaften und die Eigenschaften ein, sich auf Ihren Suchserver beziehen. Eine Beschreibung der Konfigurationseigenschaft finden Sie unter A.1, „Konfigurationseigenschaften", auf Seite 47.
- Erstellen Sie die EAR-Datei Ihrer Cúram-Anwendung in der gewohnten Weise (damit wird auch Ihre GSS-EAR-Datei erstellt).
- Richten Sie Ihre Datenbank in der gewohnten Weise ein.
- Führen Sie den Suchdatenbankextraktor des Cúram-Servers für generische Suche aus.
- Stellen Sie alle EAR-Dateien Ihrer Anwendung bereit, einschließlich 'SearchServer.ear'.
- Melden Sie sich an der Anwendung als Administrator an und richten Sie die Systemeigenschaften so ein, dass die vom GSS unterstützten Suchabfragen, die Sie verwenden wollen, und der Synchronisationsmechanismus aktiviert werden. Weitere Informationen finden Sie in Kapitel 4, „Vom Server für generische Suche aktivierte Suchabfragen", auf Seite 11.
- Führen Sie den Startprozess des Servers für generische Suche aus.

Anschließend sollte der Server für generische Suche für die Beantwortung von Abfragen verfügbar sein.

11.4 Clustering

In einer Clusterumgebung wird die Bereitstellung mehrerer GSS-Instanzen unterstützt. Eine ausführliche Beschreibung der Topologien von erweiterter Clusterbereitstellung geht über Inhalt und Umfang des vorliegenden Handbuchs hinaus. Siehe auch 12.9, „Empfohlene Konfiguration für die Produktionsumgebung", auf Seite 46.

Anmerkung: Es empfiehlt sich, den GSS in einem eigenen Cluster bereitzustellen.

11.5 Erstellungsziele

Die folgenden Erstellungsziele gelten speziell für den Cúram-Server für generische Suche.

11.5.1 weblogicEARGSS

Dieses Ziel erstellt die Datei 'SearchServer.ear' und kopiert sie - zusammen mit Ihrer Cúram-EAR-Datei - in das Verzeichnis EJBServer/build/ear/WLS/. Es wird automatisch als Teil des Ziels **weblogicEAR** ausgeführt. Die EAR-Datei des Suchservers muss nach der Cúram-EAR-Datei erstellt werden. Nach der Erstellung der EAR-Datei des Suchservers ist die Anwendung für die Bereitstellung auf Oracle WebLogic Application Server bereit, wobei dieselben Erstellungsziele oder manuellen Prozesse wie für die Cúram-EAR-Datei verwendet werden.

11.5.2 websphereEARGSS

Dieses Ziel erstellt die Datei 'SearchServer.ear' und kopiert sie - zusammen mit Ihrer Cúram-EAR-Datei - in das Verzeichnis EJBServer/build/ear/WLS/. Es wird automatisch als Teil des Ziels **websphereEAR** ausgeführt. Die EAR-Datei des Suchservers muss nach der Cúram-EAR-Datei erstellt werden. Nach der Erstellung der EAR-Datei des Suchservers ist die Anwendung für die Bereitstellung auf IBM®WebSphere Application Server bereit, wobei dieselben Erstellungsziele oder manuellen Prozesse wie für die Cúram-EAR-Datei verwendet werden.

11.5.3 runExtractor

Dieses Ziel muss ausgeführt werden, nachdem Ihre Anwendungsdatenbank konfiguriert wurde. Standardmäßig extrahiert es alle Daten, die zu den CEF-Suchservices und zu allen anderen von Ihnen definierten Suchservices gehören, aus Ihrer Anwendungsdatenbank und wandelt sie in ein Format um, das sich für die Indexierung eignet. Die Dauer dieses Prozesses steigt mit dem zu extrahierenden Datenvolumen. Dieses Ziel kann bei Bedarf mehrmals wiederholt werden.

Dieses Zeil kann gegen einen einzelnen Suchservice ausgeführt werden, indem die Eigenschaft „SERVICE" angegeben wird. Beispiel: „build runExtractor -DSERVICE=PersonSearch"

11.5.4 runPersist

Wenn Sie einen als persistent definierten Datenbankindex verwenden (siehe 12.3, „Indexpersistenz", auf Seite 43, erstellt dieses Ziel den Index aus den Staging-Datenbanktabellen. Es sollte nur ausgeführt werden, nachdem Ihre Anwendungsdatenbank konfiguriert und das Ziel 'runExtract' ausgeführt wurde. Das Ziel 'runExtract' erstellt Ihren als persistent definierten Index, wenn die Persistenz konfiguriert wurde. Daher muss dieses Ziel nur dann separat ausgeführt werden, wenn Sie Ihre Konfiguration seit der Ausführung des Ziels 'runExtractor' geändert haben.

11.5.5 startupSearchServer

Dieses Ziel ist optional. Wenn es ausgeführt werden soll, muss die Ausführung erfolgen, nachdem der Server für generische Suche auf Ihrem Anwendungsserver bereitgestellt wurde. Es bewirkt, dass der Suchserver seine Indizes so einrichtet, dass sie für Suchvorgänge verfügbar sind. Die Dauer dieses Prozesses steigt mit dem zu indexierenden Datenvolumen. Wenn Sie das Startziel nicht explizit ausführen, initialisiert der Suchserver seine Indizes bei der ersten Suchanforderung. Diese Funktion dient in erster Linie dazu, Tests mit kleinen Datensätzen zu vereinfachen. Bei großen Datensätzen sollte die automatische Startfunktion nicht verwendet werden. Sie können den automatischen Start inaktivieren, indem Sie die Eigenschaft „curam.searchserver.autostartup.disabled" in Ihrer Datei `Bootstrap.properties` auf 'true' einstellen, wenn Sie Ihre EAR-Datei einrichten. Dies wird empfohlen.

11.6 Datenbankleistung

Die Cúram-Anwendung und die Suchserveranwendung nutzen eine Datenbank gemeinsam, haben jedoch unterschiedliche Anforderungen an sie. Die Tabelle 'SearchServiceRow' trägt den Hauptteil der Schreib- und Zugriffsvorhänge und wird sehr groß, da sie eine Version aller durchsuchbaren Daten enthält. Die Cúram-Anwendung schreibt in diese Tabelle in dem Maße, in dem durchsuchbare Entitäten eingefügt oder aktualisiert werden. Immer wenn Ihr Suchserver erneut gestartet oder synchronisiert wird, erfolgen

umfangreiche Lesevorgänge aus dieser Tabelle. Je nach den Ressourcen und Anforderungen Ihrer Organisation kann es sinnvoll sein, die Tabelle 'SearchServiceRow' in einem anderen Tabellenbereich zu speichern als die übrigen Anwendungstabellen.

11.7 Hinweise zur Uhrzeit

Wenn verschiedene Systeme für die Ausführung der Cúram-Anwendung und des Servers für generische Suche verwendet werden, müssen alle Systeme eine synchronisierte Uhrzeit haben und in derselben Zeitzone bleiben. Es wird empfohlen, eine Softwarelösung wie NTP (je nach Implementierungsplattform) zu verwenden, um die Erfüllung dieser Kriterien sicherzustellen. Geschieht dies nicht, kann nicht gewährleistet werden, dass alle Aktualisierungen der Anwendungsdaten vom Server für generische Suche entsprechend dargestellt werden.

Kapitel 12. Leistung

12.1 Einführung

In diesem Kapitel werden die Leistung des Cúram-Suchservers und der Einfluss verschiedener Implementierungsszenarios und Konfigurationseinstellungen auf die Leistung beschrieben.

12.2 Indextypen

Wie unter 2.3, „Indizes", auf Seite 3 beschrieben, ist ein Index die Datenstruktur, die GSS-Suchabfragen ermöglicht. Dies kann eine Datenstruktur mit stark veränderbarer Größe sein (siehe 12.7.1, „Berechnung der Indexgröße", auf Seite 46), was die Frage aufwirkt, wo sie gespeichert werden soll. GSS bietet zwei Optionen: Speicher oder Datei. Informationen zum Konfigurieren dieser Eigenschaften finden Sie unter A.1, „Konfigurationseigenschaften", auf Seite 47.

Arbeitsspeicherverzeichnisse (speicherintern) müssen bei jedem Start eines Anwendungsservers wiederhergestellt werden (außer bei Verwendung von Persistenz, siehe 12.3, „Indexpersistenz". Sie bieten schnellen Zugriff, doch ihr Speicherbedarf überschreitet möglicherweise die verfügbaren Ressourcen. Arbeitsspeicherverzeichnisse können jedoch sehr nützlich für Tests sein, da sie Statuszustände nicht beibehalten.

Dateiindezes verwenden das lokale Dateisystem zum Speichern des Index. Zwar erstreckt sich die J2EE-Spezifikation nicht auf den Zugriff auf das Dateisystem, doch in der Praxis funktioniert dies bei allen unterstützten Versionen, die in einem separaten Dokument namens *Voraussetzungen für Cúram-Unterstützung* dokumentiert sind. Natürlich ist die Leistung des GSS desto besser, je besser die Leistung des zugrunde liegenden Dateisystems ist.

12.3 Indexpersistenz

Jedem Suchservice ist ein Index zugeordnet, der bei jeder Suche abgefragt wird. Dieser Index wird bei der Initialisierung des Suchservers aus den Staging-Datenbanktabellen generiert. Ein erheblicher Zeitraum ist möglicherweise zum Lesen aller Suchservicedaten aus den Staging-Datenbanktabellen und zum anschließenden Generieren der relevanten Indizes für diese Daten erforderlich.

Der Server für generische Suche bietet die Möglichkeit, den aktuellen Index in der Datenbank als persistent zu definieren, um die Startzeit zu verbessern. Wenn die Indexpersistenz aktiviert ist, wird der persistente Index geladen, sofern er verfügbar ist, und zwar vor der Abfrage der Staging-Tabellen. Ist er nicht verfügbar, werden alle Daten aus den Staging-Tabellen gelesen und der Startvorgang verlangsamt sich.

Dem persistenten Index wird eine Zeitmarke zugeordnet, die in der entsprechenden Suchservicetabelle dieses Index gespeichert wird. Diese Zeitmarke zeigt an, wann dieser Arbeitsspeicherindex zum letzten Mal auf Platte gespeichert wurde. Dadurch dass der Server für generische Suche diesen Zeitpunkt kennt, kann er neue oder geänderte Suchservicedaten aus den Staging-Tabellen abrufen. Der persistente Index und die neuen/geänderten Daten aus den Staging-Tabellen stellen einen vollständigen speicherinternen Index dar, in dem gesucht werden kann. Durch eine Verringerung des Zugriffs auf die Staging-Tabellen und die zugehörige Verarbeitung während der Indexerstellung lässt sich Zeit sparen.

Persistente Indexdaten werden im Format BLOB gespeichert. Dies gewährleistet die optimale Leistung beim Schreiben und Lesen eines umfangreichen Index in die bzw. aus der Datenbank.

12.3.1 Aufruf einer Persistenzoperation

Die Batchoperation 'DataBaseIndexPersist.persistIndex()' wird ausgeführt, um eine Sicherung aller Indizes durchzuführen. Um einen Index als persistent zu definieren, müssen Sie folgende Schritte ausführen:

1. Lesen Sie den aktuell als persistent definierten Index.
2. Lesen Sie die neuen oder die geänderten Daten aus den Staging-Tabellen.
3. Generieren Sie mit den obigen Schritten 1 und 2 einen speicherinternen Index.
4. Speichern Sie den neu generierten speicherinternen Index in der Datenbank.
5. Wiederholen Sie die Schritte 1 bis 4 für jeden Suchservice.

12.4 Hinweise zu Test und Betrieb

Indizes, die als persistent definiert wurden, sowie FILE-Indizes haben die Funktion, erstellte Indizes zwischen Rücksetzvorgängen des Servers zu erhalten.

Die Daten bleiben auch zwischen Operationen zur Datenbankwiederherstellung erhalten. Dies macht unter Umständen den Testern Probleme, wenn Indexdaten inkonsistent zur aktuellen Datenbank werden.

Ähnliches könnte im Betrieb passieren, wenn bei einer Datenbankaktualisierung die Aktualisierungen des Suchindex in der Anwendung nicht aktiviert werden (mit der Eigenschaft „curam.lucene.luceneOnlineSynchronizationEnabled"). Dann veralten die Daten im Index und es können Fehler auftreten.

Falls eines der beiden obigen Szenarios auftritt, können die als persistent definierten Daten manuell aus der Datenbank entfernt werden. Löschen Sie dazu alle Datenbanktabellen, die mit „GSS_" anfangen (in jedem Suchservice ist jeweils eine Tabelle vorhanden). Die als persistent definierten Indizes werden auf normale Weise neu erstellt, wenn eine Extraktions- oder Persistenzoperation ausgeführt wird.

Bei einem FILE-Index kann die Datei gelöscht werden. Wenn ein standardmäßiger Arbeitsspeichersuchservice auf solche Probleme stößt, lassen sich diese durch eine erneute Ausführung des Extraktionsprozesses lösen.

12.5 Leistungsoptimierung

In diesem Abschnitt werden die Parameter beschrieben, die sich auf die Lese- und Schreibleistung des Suchindex auswirken. Sie bestimmen darüber, wie der Index erstellt wird und wie neue Indexeinträge zu schreiben sind.

12.5.1 ???Maximale Zusammenführung von Dokumenten

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

Diese Eigenschaft verbessert bei höheren Werten die Suchzeiten und liefert bei niedrigeren Werten bessere Ergebnisse, wenn ein Index häufig aktualisiert wird. Niedrige Werte (z. B. kleiner als 10.000) sind bei einer häufigen Aktualisierung des Index am besten. Allerdings werden dadurch die Leistungswerte der Suchzeiten beeinträchtigt. Der Standardwert ist 10000000. Wenn die Suchleistung wichtig ist, sollte dieser Wert hoch sein, beispielsweise der Standardwert. Wenn dagegen die Aktualisierungsleistung der Suchdaten wichtiger ist, sollte der Wert niedrig sein, beispielsweise 10.000.

12.5.2 Zusammenführungsfaktor

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

Diese Eigenschaft hat Auswirkungen auf den Arbeitsspeicher, der bei der Aktualisierung eines Index verwendet wird. Der Index muss aufgrund von Aktualisierungen von Anwendungsdaten aktualisiert werden, die sich auf die Suche auswirken. Bei niedrigen Werten (kleiner als 10) erfolgen die Suchabfragen schneller, die Indexaktualisierungen jedoch langsamer. Bei höheren Werten (größer als 10) wird bei der Indexaktualisierung mehr Arbeitsspeicher verwendet und die Suchabfragen sind zwar langsamer, die Indexaktualisierung ist jedoch schneller. Der Standardwert ist 10. Wenn die Suchleistung wichtig ist, sollte dieser Wert kleiner als 10 sein. Wenn dagegen die Aktualisierungsleistung der Suchdaten wichtiger ist, sollte der Wert größer als 10 sein.

12.5.3 Persistenz aktivieren

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

Fügen Sie 'curam.searchserver.server.index.persistence.enable=true' zur Datei 'Bootstrap.properties' hinzu, um die Indexpersistenz zu aktivieren.

Anmerkung: Bei Aktivierung dieser Eigenschaft werden bei der Ausführung der Datenbankextraktion auch die neu als persistent definierten Indizes generiert.

12.5.4 Referenzinformationen

Weitere Informationen zu den in diesem Abschnitt behandelten Parametern finden Sie unter http://lucene.apache.org/java/2_2_0/api/index.html.

12.6 Searcher-Pools

In diesem Abschnitt wird die Vorgehensweise beim Konfigurieren von Searcher-Pools und deren Auswirkung auf die Suchleistung beschrieben.

12.6.1 Übersicht

Lucene besitzt einen Caching-Mechanismus, mit dessen Hilfe Suchabfragen, die bestehende IndexSearcher-Objekte verwenden, schneller abgewickelt werden als Suchabfragen mit neu erstellten IndexSearcher-Instanzen. Eine einzige gemeinsam genutzte IndexSearcher-Instanz würde für schnelle Suchabfragen in einer Einzelbenutzerumgebung ausreichen. Der Standardanwendungsfall in einer Serverumgebung ist jedoch, dass mehrere Clients den Index gleichzeitig durchsuchen. Zur Vermeidung einer Reihenfolgeplanung der Suchanforderungen in solchen Situationen, was die individuelle Suchleistung beeinträchtigen würde, verwendet der GSS einen IndexSearcher-Pool, der eine definierte Anzahl von IndexSearcher-Instanzen für die Wiederverwendung durch gleichzeitig ablaufende Suchanforderungen vorhält.

Für ein IndexSearcher-Objekt wird der Index nur in dem Status angezeigt, den er zu seinem 'Öffnungszeitpunkt' hatte. Die Aktualisierungen des Index, die nach dem Öffnen des IndexSearcher-Objekts erfolgen, werden erst sichtbar, wenn das IndexSearcher-Objekt erneut geöffnet wird. Je nachdem, wie groß der Index ist und ob er in der Zwischenzeit aktualisiert wurde, können die einzelnen IndexSearcher-Instanzen eine sehr große Speichermenge belegen. Der IndexSearcher-Pool sorgt dafür, dass die IndexSearcher-Instanzen bei einer Indexaktualisierung geschlossen und erneut geöffnet werden.

12.6.2 Poolkonfigurationseigenschaften

Ein IndexSearcher-Pool besitzt zwei Basisoptionen: Anfangsgröße und Maximalgröße. Der Parameter

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

gibt an, wie viele IndexSearcher-Instanzen beim Start geöffnet sind und die ganze Zeit zur Verwendung durch Suchclients geöffnet bleiben. Dies ist eine erforderliche Option, die positive ganzzahlige Werte einschließlich 0 annimmt. Wenn nichts angegeben wird, lautet der Standardwert "0". In der Regel sollte diese Eigenschaft auf die vermutete maximale Anzahl simultaner Clientsuchen eingestellt werden.

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

gibt die maximale Anzahl der IndexSearcher-Instanzen an, die gleichzeitig geöffnet sein dürfen. Sobald die Suchabfragen diese Anzahl übersteigen, wird eine Ausnahme ausgelöst und zu Diagnosezwecken protokolliert. Diese Option nimmt positive ganze Zahlen an und hat den Standardwert "100", wenn nichts angegeben wird. Darüber hinaus gibt es die folgende zugehörige Option:

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

Sie gibt an, ob die maximale Poolgröße unbegrenzt ist. Die Option nimmt die Werte "true" oder "false" an. Wenn nichts angegeben wird, lautet der Standardwert "true". Wird diese Option auf "true" eingestellt, wird der Wert der Option 'curam.searchserver.luceneadaptor.searcher.pool.maxsize' ignoriert. Eine dieser beiden zugeordneten Optionen ist erforderlich.

12.7 Begrenzungen des Arbeitsspeichers

Die Indizes des Servers für globale Suche werden intern gespeichert, wenn sie entsprechend konfiguriert sind. Bei Verwendung einer 32-Bit-JVM beträgt die Speicherbegrenzung ungefähr 3 GB. Dieser Wert bezieht sich jedoch nicht nur auf den für den GSS, sondern auch für alle anderen Systemprozesse verfügbaren Speicher. Beachten Sie unbedingt, dass sehr große Suchserviceindizes den maximalen Arbeitsspeicher überschreiten können, der für den GSS und für andere bereitgestellte Prozesse verfügbar ist.

12.7.1 Berechnung der Indexgröße

Die Indexgröße beträgt etwa 30 % des indexierten Texts. Die indexierten und gespeicherten Eigenschaften des Suchservice (sie können aus den SearchServiceField-Attributen abgerufen werden; dabei gilt: indexed=true und stored=true) werden zum Schätzen der Indexgröße verwendet.

- 1 Million Personeneinträge. Dabei entspricht ein Eintrag einem Indexdokument.
- Ein Dokument kann folgende indexierte und gespeicherte Eigenschaften enthalten, die für einen Personensuchservice aus der Tabelle 'SearchServiceField' ermittelt wurden: refnumber(10) forename(20), surname(20), AddressLine1(30), AddressLine2(30), city(20), country(15), gender(10). Dabei steht (*) für die maximale Zeichenanzahl in dem entsprechenden Feld.
- 1 Dokument = (155 Zeichen für den gespeicherten Wert) + (66 Zeichen für jeden Feld-/Begriffsnamen) = 221.
- ???Memory 1M Person documents und Java mit 16-Bit-Unicode pro Zeichen. Gesamter indexierter und zurückgegebener Text 442 MB * 30 % = 132 MB.

12.8 Empfohlene Konfiguration

Es wird empfohlen, als Konfiguration für den Cúram-Server für generische Suche standardmäßig den Indextyp FILE mit inaktivierter Indexpersistenz zu verwenden. Dies sollte eine gute Leistung ohne Dimensionierungsprobleme gewährleisten. Der Suchserver sollte als separate Anwendung bereitgestellt werden und nicht der Cúram-Anwendung benachbart sein (siehe Kapitel 11, „Server für generische Suche bereitstellen”, auf Seite 39.

12.9 Empfohlene Konfiguration für die Produktionsumgebung

Der Indextyp FILE ist die einzige Konfiguration, die in einer Produktionsumgebung unterstützt wird.

Anhang A. Konfigurationseigenschaften des Cúram-Servers für generische Suche

A.1 Konfigurationseigenschaften

Vor Beginn der Entwicklung oder der Bereitstellung des Cúram-Servers für generische Suche sollten bei Bedarf die folgenden Einstellungen zu Ihrer Datei `Bootstrap.properties` hinzugefügt werden.

Tabelle 3. Grundlegende Konfigurationseinstellungen des Cúram-Servers für generische Suche

Eigenschaftsname	Beschreibung
curam.searchserver.sync.interval	Das Intervall in Millisekunden zwischen Synchronisationsaufrufen des Servers für generische Suche. Dies ist die maximale Zeit zwischen der Aktualisierung von Daten und deren Verfügbarkeit für Suchvorgänge. Wird diese Eigenschaft nicht festgelegt, wird standardmäßig alle drei Sekunden eine Synchronisation durchgeführt.
curam.searchserver.sync.username	Der Benutzername, mit dem die Anmeldung bei der Anwendung erfolgt, um die Synchronisation durchzuführen. Der Benutzer muss zur Ausführung der Funktionskennung 'DoGSSSync.sync' berechtigt sein. Ist nur bei der Ausführung auf WebSphere Application Server erforderlich. Wenn Sie diese Eigenschaft und das zugehörige Kennwort nicht angeben, wird die Synchronisation dadurch nicht verhindert. Es werden jedoch bei jeder Synchronisation Sicherheitswarnungen in die Protokolldateien geschrieben.
curam.searchserver.sync.password	Das Kennwort, das der oben beschriebenen Eigenschaft 'curam.searchserver.sync.username' zugeordnet ist. Dieses Kennwort sollte mit dem standardmäßigen Verschlüsselelungserstellungsziel von Cúram verschlüsselt werden.
curam.searchserver.environment.vendor	Diese Eigenschaft sollte auf „ITD", „IBM" oder „BEA" festgelegt werden, je nachdem, ob Sie den Suchserver im Entwicklungsmodus verwenden oder eine Bereitstellung für WebSphere oder WebLogic vornehmen. Wird diese Eigenschaft nicht festgelegt, verwendet der Suchserver standardmäßig die Eigenschaft 'curam.environment.as.vendor'.
curam.searchserver.server.host	Der Domänenname oder die IP-Adresse des Servers, auf dem Ihr Suchserver ausgeführt wird. Diese Eigenschaft muss festgelegt werden, damit Sie den Serverstart über die Befehlszeile ausführen können. Wird diese Eigenschaft nicht festgelegt, wird der Standardwert 'localhost' verwendet.
curam.searchserver.server.port	Der Port, an dem der RMI-Service Ihres Anwendungsservers verfügbar ist. Diese Eigenschaft muss festgelegt werden, damit Sie den Serverstart über die Befehlszeile ausführen können.
curam.searchserver.autostartup.disabled	Zu Test- und Entwicklungszwecken initialisiert der Suchserver seine Indizes bei der ersten Suchanforderung, sofern er bereits gestartet wurde. In einem Bereitstellungsszenario empfiehlt es sich möglicherweise, dieses Verhalten zu inaktivieren, um sicherzustellen, dass der Startprozess über die Befehlszeile ausgeführt wird, sodass Sie den Prozess besser steuern können. Wenn Sie diese Eigenschaft auf 'true' festlegen, wird das automatische Startverhalten inaktiviert. Beachten Sie, dass der Suchserver eine Ausnahme auslöst, wenn vor dem Abschluss des Systemstarts versucht wird, Suchvorgänge auszuführen.
curam.searchserver. luceneadaptor.searcher.index.maxmergedocs	Mithilfe dieser Eigenschaft wird die Leistung der Lese- und Schreibvorgänge im Index optimiert. Größere Werte („1.000.000") sind optimal zum Schreiben von Indizes im Batchbetrieb und für schnellere Suchabfragen. Kleinere Werte „10.000" sind optimal für die interaktive Indexierung, bei der zahlreiche einzelne Indexaktualisierungen vorgenommen werden.
curam.searchserver.luceneadaptor.document.flush.count	Gibt die Anzahl der Dokumente an, die vor dem Schreiben in den Index aktualisiert werden sollen, wenn eine große Menge von Dokumenten verarbeitet wird. Wird kein Wert angegeben, werden standardmäßig 1000 Dokumente angenommen. Die Optimierung dieser Eigenschaft kann den Zeitraum verringern, der bei Indexpersistenz oder beim Serverstart für die Indexerstellung erforderlich ist.
curam.searchserver.term.min.length	Die zulässige Mindestlänge eines Suchbegriffs. Der Standardwert sind zwei Zeichen. Sehr kurze Suchbegriffe bewirken eine niedrige Suchleistung und normalerweise eine schlechte Qualität der Suchergebnisse.
curam.searchserver.directory.type	Mit dieser Eigenschaft wird angegeben, welcher Speichertyp für Suchservices verwendet werden soll: RAM oder FILE. RAM ist der Standardindextyp und eignet sich für kleinere Indizes, für die eine schnelle Leistung erforderlich ist. Die Einstellung FILE stellt einen Speicher für große Indizes im Dateisystem bereit.
curam.searchserver.file.index.location	Mit dieser Eigenschaft wird angegeben, wo der Dateiindex im Dateisystem gespeichert werden soll, wenn mehr Daten vorhanden sind und Folgendes gilt: curam.searchserver.directory.type=FILE. Bei einer Bereitstellung auf mehreren System sollte die Dateiposition auf jedem Zielsystem vorhanden sein.

Tabelle 4. Suchpooleinstellungen des Cúram-Servers für generische Suche

Eigenschaftsname	Beschreibung
curam.searchserver.luceneadaptor.searcher.pool.initialsize	Diese Eigenschaft initialisiert beim Systemstart die Anzahl der Searcher-Instanzen im Searcher-Pool. Der Standardwert beträgt 0.
curam.searchserver.luceneadaptor.searcher.pool.maxsize	Diese Eigenschaft gibt die maximale Anzahl der IndexSearcher-Instanzen im Searcher-Pool an. Der Standardwert beträgt 100.
curam.searchserver.luceneadaptor .searcher.pool.maxsizeunbounded	Wird diese Eigenschaft auf „true" festgelegt, wird 'curam.searchserver.luceneadaptor.searcher.pool.maxsize' überschrieben und angegeben, dass im Searcher-Pool keine maximale Anzahl an IndexSearcher-Instanz zulässig ist. Der Standardwert ist „true".
curam.searchserver.luceneadaptor.searcher.pool.mergefactor	Mithilfe dieser Eigenschaft wird die Leistung der Lese- und Schreibvorgänge im Index optimiert. Der Standardwert ist „10". Der Mindestwert ist „2". Höhere Werte erhöhen die Arbeitsspeicherbelegung, verringern die Suchgeschwindigkeit, aber beschleunigen das Schreiben von Indizes.

Tabelle 5. Persistenzseinstellungen des Cúram-Servers für generische Suche

Eigenschaftsname	Beschreibung
curam.searchserver.server.index.persistence.enable	Diese Eigenschaft sollte auf „true" festgelegt werden, um Indexpersistenz zu aktivieren. Wird diese Eigenschaft nicht festgelegt, wird der Standardwert „false" verwendet.
curam.searchserver.custom.db.init	Diese Eigenschaft sollte auf „true" festgelegt werden, wenn Datenbanktabellen mit Indexpersistenz angepasst werden. Sie zeigt an, dass die Standardtabellen mit Indexpersistenz nicht verwendet werden sollen und dass diese Tabellen mithilfe der Datei `CustomDBSearchServices.properties` eingerichtet werden sollen.

Anhang B. DMX-Beispiellisten: Personensuche

B.1 Suchserviceeintrag

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

  <column name="
      searchServiceId
      " type="text" />
  <column name="
      name
      " 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 Suchservicefeldeintrag

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

  <column name="
      searchServiceFieldId
      " type="text" />
  <column name="
      searchServiceId
      " type="text" />
  <column name="
      name
      " 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>
```

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM-Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden. Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing

IBM Europe, Middle East & Africa

Tour Descartes

2, avenue Gambetta

92066 Paris La Defense

France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden.

Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen. Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar.

Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht. Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Bereitstellung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen.

IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können u. U. von den hier genannten Preisen abweichen.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. IBM kann daher die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programm nicht garantieren oder implizieren. Die Musterprogramme werden "WIE BESEHEN", ohne Gewährleistung jeglicher Art bereitgestellt. IBM übernimmt keine Haftung für Schäden, die durch Ihre Verwendung der Musterprogramme entstehen.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihres Unternehmens) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Website "Copyright and trademark information" unter http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache ist eine eingetragene Marke der Apache Software Foundation.

Oracle, WebLogic Server, Java und alle auf Java basierenden Marken und Logos sind eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Andere Namen können Marken der jeweiligen Rechtsinhaber sein. Weitere Firmen-, Produkt- und Servicenamen können Marken oder Servicemarken anderer Unternehmen sein.

Gedruckt in Deutschland