FirstSpirit™
Unlock Your Content

FirstSpirit™ EnterpriseSearch

FirstSpirit Version 5.0

Version	**2.3**
Status	**RELEASED**
Datum	**2014-05-14**
Abteilung	Produktmanagement
Autor/ Autoren	Donato Marro, Andreas Knoor
Copyright	2011 e-Spirit AG

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-spirit.com
www.e-spirit.com

e-Spirit

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die unterschiedlichen Integrationsmöglichkeiten zwischen dem Content Management System FirstSpirit und der Enterprise Suchmaschinentechnologie EXALEAD CLOUDVIEW™ (im Folgenden kurz "Exalead" genannt).

Generell sind beide Produkte unabhängig voneinander einsetzbar. In Kombination können allerdings beide modernen Technologien so miteinander verbunden werden, dass sich ein hoher funktionaler Nutzwert für den Website-Benutzer ergibt, welcher auf einfachste Art und Weise bestimmte Information innerhalb von FirstSpirit-Inhalten finden möchte.

Insgesamt werden folgende sechs Aspekte beschrieben:

- Kapitel 2: Architektur der Integrationslösung zwischen FirstSpirit und Exalead
- Kapitel 3: Beschreibung der FirstSpirit Laufzeitkomponente zur Darstellung von Exalead-Suchergebnissen auf dem Webserver (Modul "Exalead Live")
- Kapitel 4: Erläuterung der Möglichkeiten, Dokumente mit Rechten zu versehen und damit eine personalisierte Suche anzubieten
- Kapitel 5: Inkrementelle Indexaktualisierung mithilfe des FirstSpirit-Moduls "Exalead Content-Update"
- Kapitel 6: Verwendung der Autovervollständigung
- Kapitel 7: Migrationshinweise für die Aktualisierung von Version 1.x auf Version 2.0

Technisch teilt sich die FirstSpirit Exalead Integration somit in zwei Sub-Module auf: "Exalead Live" zur Suche und Ergebnisdarstellung, sowie "Exalead Content-Update" zur inkrementellen Indexaktualisierung.

Diese Dokumentation bezieht sich auf Version 2.3 der Module FirstSpirit Exalead Live und FirstSpirit Exalead Content-Update!

2 Architektur und Komponenten

Insgesamt sind folgende Komponenten im aufgeführten Integrationsszenario relevant:

- FirstSpirit-Server (Version >= 4.2.438), inkl. der Module
  - FirstSpirit Exalead Live
  - FirstSpirit Personalisation (optional)
  - FirstSpirit Exalead Content-Update (optional)
- Exalead CloudView Server (Version 5.1)
- Java Application Server (Servlet-Spec. >= 2.4, empfohlen: Apache Tomcat)
- Java Runtime Environment 1.5 (für die Nutzung der Push-API wird Version 1.6 vorausgesetzt)

Die eigentliche Installation von FirstSpirit, Exalead und dem Application-Server wird hier nicht weiter beschrieben. Bitte greifen Sie dafür auf die jeweiligen Installationsdokumente zurück.

Einige der aufgeführten Module sind optional und daher nicht in allen Anwendungsfällen notwendig bzw. sinnvoll. Daher werden in den weiteren Kapiteln unterschiedliche Ausbaustufen der Integration beleuchtet. Die Varianten werden von Stufe zu Stufe komplexer, bauen funktional jedoch aufeinander auf.

2.1 Basisarchitektur

Die hier erläuterte Basisarchitektur ist beispielsweise für Internet-Auftritte geeignet, die primär öffentliche, nicht personalisierte Informationen durchsuchbar machen wollen.

Das Zusammenspiel der einzelnen Komponenten sowie der Datenfluss untereinander wird in Abbildung 2-1 schematisch dargestellt:

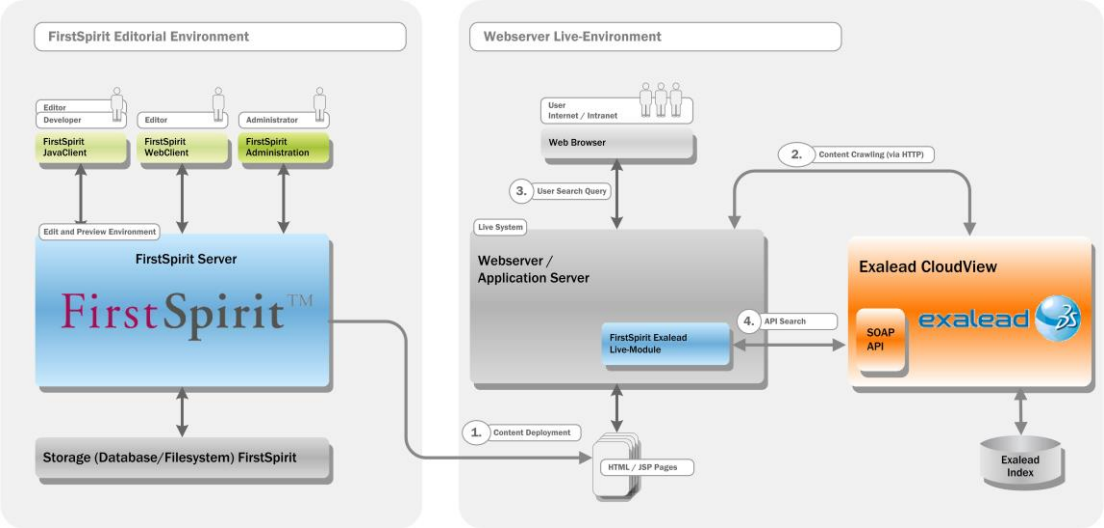

Abbildung 2-1: Basisarchitektur

Ablaufbeschreibung:

- Mithilfe von FirstSpirit werden die Inhalte des Internet-Auftritts gepflegt und in passende HTML- oder JSP-Seiten transformiert.
- Im Rahmen eines Deployment-Prozesses werden die erzeugten Dateien anschließend an den Application-Server übertragen [Schritt 1].
- Sobald alle Seiten auf den Webserver übertragen sind, kann die Exalead Suchmaschine die Dateien im Rahmen eines "Crawling"-Vorgangs einlesen, analysieren und in den Index aufnehmen. Dieser Vorgang geschieht zyklisch, z. B. nach jedem Deployment neuer bzw. geänderter Dateien [Schritt 2]. Der Such-Index ist jetzt gefüllt und kann Suchanfragen beantworten.
- Der Besucher der Internet-Seite hat nun über ein Suchformular die Möglichkeit, die indizierten Inhalte zu durchsuchen [Schritt 3]. Das Formular übergibt die Formularanfrage an das "FirstSpirit Exalead Live Modul", welches die Anfrage wiederum an den Exalead-Server übergibt. Technisch enthält das "FirstSpirit Exalead Live Modul" ein Servlet, welches die SOAP-API (Simple-Object-Access-Protocol) des Exalead-Servers verwendet.
- Nachdem die Suchanfrage vom Exalead-Server verarbeitet und beantwortet worden ist, stellt das "FirstSpirit Exalead Live Modul" die Suchergebnisse auf einer entsprechenden Seite übersichtlich dar. Technisch erfolgt die Parametrisierung des Designs der Ausgabe über eine JSP-Tag-Library.

Diese Basisarchitektur ermöglicht die Anwendung der kompletten Exalead-Suchmöglichkeiten auf FirstSpirit-Inhalte, wie z. B.

- ✓ Indizierung von über 300 Formaten, inkl. Word, Excel, Powerpoint, PDF, RTF, OpenOffice, HTML, XML, Bilder, Audio, Video
- ✓ Mehrsprachen-Support (Unicode)
- ✓ Natürliche Sprachverarbeitung (Rechtschreibvorschläge, phonetischer Abgleich)
- ✓ Automatische Klassifizierung von Suchanfragen und automatische Keyword-Erzeugung
- ✓ Vorschau-Thumbnails der Treffer
- ✓ Geführte Navigation durch "Drill-Down" innerhalb der Ergebnislisten

Einschränkungen der Basisarchitektur:

- keine personalisierte Auslieferung von HTML-Seiten auf Basis von Berechtigungen
- keine personalisierte Auslieferung von Binärdokumenten auf Basis von Berechtigungen
- keine Ad-hoc-Aktualisierung des Index bei neuen Inhalten

2.2 Architektur mit Seitenpersonalisierung

Für Web-Szenarien, bei denen auch geschützte Seiteninhalte relevant sind (z. B. im Falle von Intranets), kann der Basisansatz aus dem vorherigen Kapitel um den Aspekt der Personalisierung ergänzt werden.

Es wird im Folgenden davon ausgegangen, dass zur Personalisierung der Inhalte auf Seiten des Application-Servers das Modul "FirstSpirit Personalisation" verwendet wird.

Architektonisch ergibt sich damit der Aufbau wie in Abbildung 2-2:

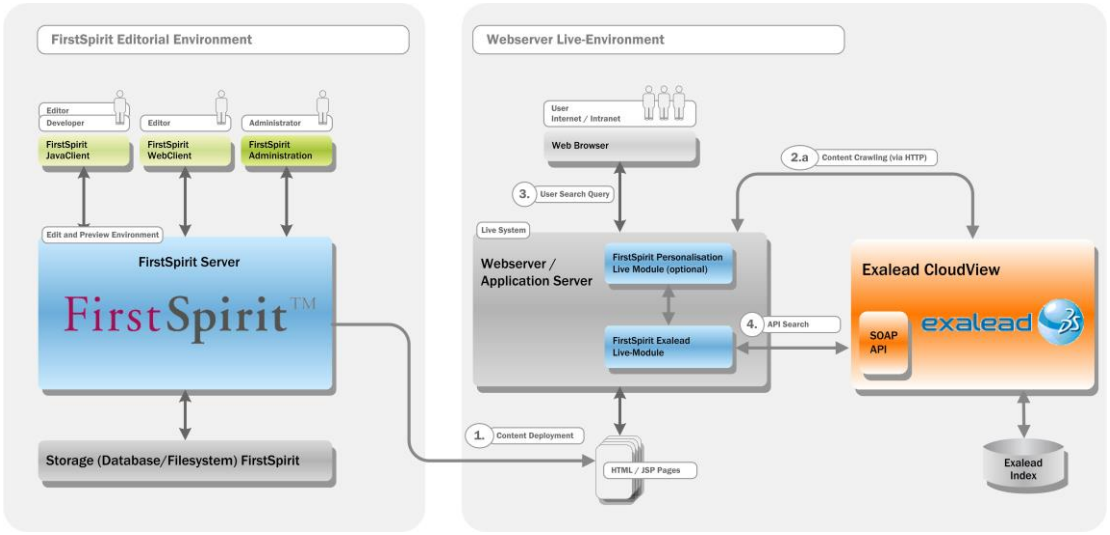

Abbildung 2-2: Personalisierte Suche

Der Aufbau entspricht weitgehend der Basisarchitektur, es kommt jedoch eine Kopplung des "FirstSpirit Exalead Live Moduls" und des "FirstSpirit Personalisation Moduls" hinzu.

Technisch interagieren die Module in der Art, dass alle Suchanfragen, die an die SOAP-API von Exalead übergeben werden, zusätzlich noch mit den relevanten Personalisierungsinformationen des aktuell angemeldeten Besuchers ergänzt werden. Konkret sind dies:

- Login-Name des Benutzers
- Liste der Gruppen des Benutzers

Diese Informationen werden vom "FirstSpirit Exalead Live Modul" transparent aus dem Personalisierungsmodul übernommen.

Der Aufbau des Index erfolgt weiterhin komplett über einen Crawling-Ansatz.

Um eine korrekte Filterung der HTML-Seiten durchführen zu können, müssen innerhalb der HTML-Seiten passende Meta-Tags hinterlegt werden, die die Personalisierungsinformationen pro Seite definieren. Details dazu sind im Kapitel 4.2 aufgeführt. An dieser Stelle ist nur festzuhalten, dass durch passende FirstSpirit-Vorlagen die Berechtigungsinformationen in die HTML injiziert werden.

Damit ergeben sich als Erweiterung der Basisarchitektur folgende Möglichkeiten:

- ✓ sehr einfach zu realisierende Unterstützung von Seitenpersonalisierung durch simple FirstSpirit-Vorlagen-Konstrukte

- ✓ Definition von Berechtigung auf HTML-Seiten direkt durch den FirstSpirit-Redakteur
- ✓ Personalisierte, benutzerspezifisch gefilterte Suchergebnisse durch Kopplung der Benutzerverwaltungen von FirstSpirit und Exalead

Einschränkungen dieser Architektur:

- keine personalisierte Auslieferung von Binärdokumenten auf Basis von Berechtigungen (Personalisierungsinformationen können dort nicht injiziert werden)
- keine Ad-hoc-Aktualisierung des Index bei neuen Inhalten (generell beim Crawling-Ansatz nicht möglich)

2.3 Architektur mit Dokumentenpersonalisierung (Push-API)

Um die beiden Aspekte

- Übergabe von Personalisierungsinformationen an Binärdokumente
- Ad-hoc-Aktualisierung des Index

realisieren zu können, muss man den bisherigen Crawling-Ansatz durch eine zweite Strategie ersetzen: die Exalead Push-API.

Bei der Verwendung der Push-API werden aktiv Dokumente von einer Quelle (FirstSpirit) an die Suchmaschine übergeben (per "Push-Befehl"). Neben dem reinen Dokument können dabei auch Meta-Attribute wie Personalisierungsinformationen mitgegeben werden.

Der Aufbau ist im Schaubild in Abbildung 2-3 zu sehen:

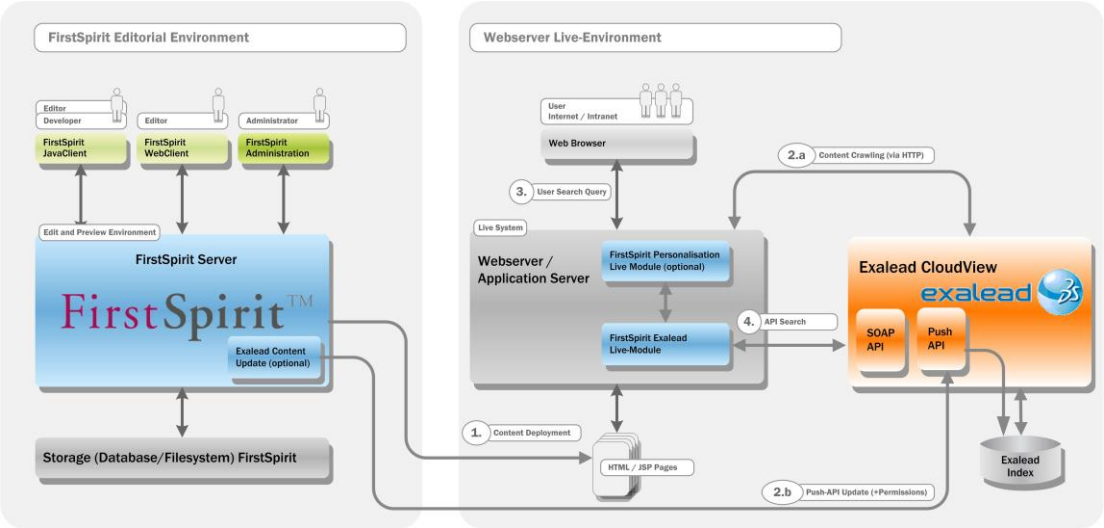

Abbildung 2-3: Push-API-Architektur

Auf Seiten des FirstSpirit-Servers ist das Modul "Exalead Content Update" hinzugekommen. Dieses ist in der Lage, Binärdokumente und HTML-Seiten direkt per Push-API an den Exalead-Server zu übergeben.

Auf dem FirstSpirit-Server werden dazu die lokal erzeugten Dateien (Seiten und Dokumente) eingelesen und an die Exalead Push-API übergeben [Schritt 2.b]. Details zur Verwendung der Push-API von FirstSpirit heraus sind in Kapitel 5 aufgeführt.

2.3.1 Crawling vs. Push-API

Die beiden Varianten der Indizierung "Crawling" [Schritt 2.a] bzw. Push-API [Schritt 2.b] schließen sich nicht zwangsläufig aus, sondern können auch in Kombination eingesetzt werden. In welchen Fällen dies Sinn macht, soll nun näher beleuchtet werden.

Auf den ersten Blick hat der ausschließliche Einsatz der Push-API nur Vorteile:

a) zeitnahe Index-Aktualisierung im Rahmen eines FirstSpirit-Deployments

b) Übergabe von Metadaten für die Personalisierung für Seiten und Dokumente

Es existiert jedoch eine Kondition, unter der eine Kombination von Crawling (nur für HTML-Seiten) und Push-API (nur für Dokumente) notwendig ist.

Immer dann, wenn die HTML-Seiten "indizierungsrelevante" Laufzeitlogik enthalten,

z. B. in Form von JSP, PHP oder .NET Code, die ausschließlich innerhalb des Ziel-Application-Servers lauffähig ist, kann auf ein Crawling über den Application-Server nicht verzichtet werden.

Beispiel:

Eine von FirstSpirit erzeugte JSP-Seite enthält neben redaktionellen FirstSpirit-Inhalten auch eine Inkludierungslogik, welche Nachrichtenbeiträge von einer externen Quelle zur Laufzeit einbindet und anzeigt. Der Inhalt der Nachrichten soll bei der Indizierung der Seiten mitberücksichtigt werden, damit auch diese Schlagwörter bei der Suche zu einem Treffer führen.

In solchen Situationen ist der kombinierte Einsatz von Crawling und Push-API notwendig.

Die Integrationsarchitektur ist in jedem Fall hinreichend flexibel, um auch solche komplexeren Szenarien adäquat abzubilden.

3 FirstSpirit Exalead Live-Modul

3.1 Allgemeine Hinweise

Dieses Kapitel beschreibt die Installation und Konfiguration des Moduls "FirstSpirit Exalead Live". Es wird davon ausgegangen, dass sowohl FirstSpirit als auch der Exalead-Server sowie ein passender Java Application-Server bereits aufgesetzt und fertig konfiguriert sind.

3.1.1 Zeichenkodierung

Das FirstSpirit Exalead Live-Modul verwendet durchgehend UTF-8 als Zeichenkodierung. Um die einheitliche Verwendung dieser Zeichenkodierung zu gewährleisten, ist es notwendig, einige spezifische Angaben dazu zu machen.

Jede HTML-Seite sollte mit einem entsprechenden Meta-Tag versehen sein:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
```

In JSP-Seiten ist zusätzlich auch noch das Page-Encoding anzugeben:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

Bei der Verwendung von Formularen ist darauf zu achten, dass auch hier die zu verwendende Zeichenkodierung angegeben wird:

```
<form ... method="post" accept-charset="UTF-8">
```

Letztlich ist es noch notwendig, den Apache Tomcat Server so zu konfigurieren, dass auch beim Aufruf von Hyperlinks die richtige Zeichenkodierung verwendet wird. Dazu ist in der `server.xml` des Apache Tomcat (im Verzeichnis `conf`) folgender Abschnitt um den Parameter `URIEncoding="UTF-8"` zu ergänzen:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" ... URIEncoding="UTF-8"/>
```

3.1.2 Sonderzeichen im URL-Pfad

Trotz durchgehender Verwendung von UTF-8 müssen Sonderzeichen innerhalb des URL-Pfades gesondert behandelt werden. Dies betrifft zum einen die Angabe der

RedirectUrl und ErrorUrl des SearchServlets (Kapitel 3.7.1 SearchServlet) und zum anderen die RedirectUrl und ErrorUrl des ContainerTags (Kapitel 3.10.1 <search:container>).

Im Falle des SearchServlets müssen die Urls bei der Angabe in den Formularfeldern 1-fach UTF-8 encodiert werden.

Beispiel: Die Suchergebnisseite befindet sich unter http://www.mydomain.com/suche/presseerklärungen/suche.jsp

Die Angabe der RedirectUrl im SearchServlet muss somit folgende Form besitzen:

```
<input type="hidden" name="redirectUrl"
value="/suche/presseerkl%C3%A4rungen/suche.jsp"/>
```

Im Falle des ContainerTags ist es notwendig, die Urls 2-fach UTF-8 zu encodieren:

```
<search:container hasResults="true"
redirectUrl="/suche/presseerkl%25C3%25A4rungen/suche.jsp" …
```

3.2 Quick-Install

Die einfachste und schnellste Art, das Modul "FirstSpirit Exalead Live" zu installieren, besteht darin, das Archiv "search.war" ins Webapps-Verzeichnis des Servers zu kopieren, wo es bei laufendem Server automatisch oder aber spätestens nach einem Neustart des Servers entpackt und als Webapplikation mit dem Namen "search" angelegt wird.

Nun muss in der Datei `webapps/search/WEB-INF/web.xml` noch der Servername `myserver` durch den Namen des Exalead-Servers ersetzt werden (bei Bedarf auch den Port anpassen).

Im Anschluss ist die Suche über einen Aufruf von `/search/index.jsp` erreichbar:

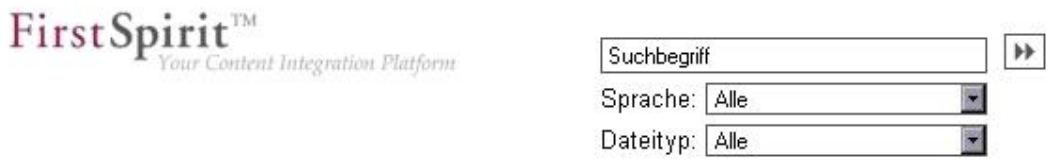

Abbildung 3-1: Suchmaske

3.3 Einbindung in bestehende Webapplikation

Um das FirstSpirit Exalead Live-Modul zu einer bestehenden Webanwendung hinzuzufügen, müssen folgende Dateien aus der in Kapitel 3.2 angelegten Webanwendung in die bestehende Webanwendung kopiert werden:

```
/WEB-INF
- exalead.tld
- web.xml (nur noch nicht vorhandene Einträge übernehmen)
- firstpersonalisation.tld (falls noch nicht vorhanden)
- firstpersonalisation.xml (falls noch nicht vorhanden)

/WEB-INF/lib
- kompletter Inhalt

/WEB-INF/classes
- log4j.properties (falls Log4J nicht anderweitig konfiguriert)
```

Bei gleichzeitigem Einsatz von FirstSpirit™ Dynamic Personalization muss darauf geachtet werden, dass das dazugehörige personalisation.jar im lib-Verzeichnis nicht durch das personalisation.jar aus dem EnterpriseSearch-Modul überschrieben wird.

Im Folgenden werden die notwendigen Einträge in der `web.xml` und die entsprechenden Servlets genauer beschrieben. Außerdem werden die im Zusammenhang mit der Einbindung in eine bestehende Webanwendung nötigen Änderungen an den Einträgen in der `web.xml` erläutert.

In Kapitel 3.8 Seite 25 folgt die Beschreibung der Tag Library, mit deren Hilfe man die Suchoberfläche- und ergebnisse nach eigenen Wünschen gestalten kann.

3.4 web.xml – Servlets

Die Exalead-Webapplikation stellt vier Servlets zur Verfügung:

- *SearchServlet*
  Dieses Servlet leitet die Suchanfragen an den Exalead-Server weiter (zwingend notwendig).

- *SearchDownload*
  Dieses Servlet ist dafür zuständig, Ergebnisdokumente, die nicht direkt vom Browser eines Benutzers erreichbar sind (z. B. Suchtreffer im Dateisystem), beim Exalead-Server abzurufen und über die Webanwendung an den Browser

auszuliefern.

- *ThumbnailServlet*
  Dieses Servlet gibt die Vorschaubilder der Suchtreffer aus, falls solche vorhanden sind.

- *AutocompleteServlet*
  Dieses Servlet bildet die Schnittstelle zum SuggestService von Exalead, der die Autovervollständigung der eingegebenen Suchbegriffe vornimmt.

Detaillierte Informationen zu den Servlets sind im Kapitel 3.7 ab Seite 20 zu finden.

3.5 web.xml – Context-Parameter

`exalead.server.host`

Dieser Parameter dient zur Angabe des Servers, auf dem die Exalead installiert ist.

`exalead.server.baseport`

Über diesen Parameter konfiguriert man den Basisport, den man bei der Exalead-Installation angegeben hat.

`getDocumentServlet`

Dieser Parameter wird nur in Verbindung mit dem `SearchDownload-Servlet` benötigt und enthält den Pfad zu diesem Servlet innerhalb der Webapplikation (Name der Webapplikation im Beispiel: "`/search`").

3.6 Beispielhafte web.xml

Die notwendigen Parameter in der web.xml-Datei sollen hier noch einmal mit sinnvollen Werten zusammengefasst werden. Eine web.xml-Datei, die um die Exalead-Integrations-Elemente ergänzt wurde, könnte damit folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
...
<context-param>
          <param-name>exalead.server.host</param-name>
          <param-value>localhost</param-value>
</context-param>
```

```
<context-param>
          <param-name>exalead.server.baseport</param-name>
          <param-value>10000</param-value>
</context-param>
<context-param>
          <param-name>getDocumentServlet</param-name>
          <param-value>/search/do.searchDownload</param-value>
</context-param>

<servlet>
          <servlet-name>SearchServlet</servlet-name>
          <servlet-class>
                    de.espirit.ps.exalead.servlets.SearchServlet
          </servlet-class>
          <init-param>
                    <param-name>userPrefix</param-name>
                    <param-value>user:</param-value>
          </init-param>
          <init-param>
                    <param-name>groupsPrefix</param-name>
                    <param-value>groups:</param-value>
          </init-param>
          <init-param>
                    <param-name>sendSecurityToken</param-name>
                    <param-value>false</param-value>
          </init-param>
</servlet>

<servlet>
          <servlet-name>SearchDownload</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.SearchDownload
          </servlet-class>
</servlet>


<servlet>
          <servlet-name>ThumbnailServlet</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.ThumbnailServlet
          </servlet-class>
</servlet>

<servlet>
          <servlet-name>AutocompleteServlet</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.AutocompleteServlet
          </servlet-class>
          <init-param>
                    <param-name>suggestServiceName</param-name>
                    <param-value>mysuggest</param-value>
          </init-param>
</servlet>

<servlet-mapping>
          <servlet-name>SearchServlet</servlet-name>
          <url-pattern>*.search</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
          <servlet-name>SearchDownload</servlet-name>
          <url-pattern>*.searchDownload</url-pattern>
</servlet-mapping>

<servlet-mapping>
          <servlet-name>ThumbnailServlet</servlet-name>
          <url-pattern>*.searchThumbnail</url-pattern>
</servlet-mapping>

<servlet-mapping>
          <servlet-name>AutocompleteServlet</servlet-name>
          <url-pattern>*.autocomplete</url-pattern>
</servlet-mapping>
...
</web-app>
```

3.7 Beschreibung der Servlets

Wie bereits im vorherigen Kapitel erwähnt, liefert das FirstSpirit Exalead Live-Modul insgesamt vier Servlets mit, deren Konfiguration im Folgenden beschrieben wird.

3.7.1 SearchServlet

Das `SearchServlet` nimmt alle Suchanfragen entgegen, leitet sie passend an den Exalead-Server weiter, erhält im Anschluss von diesem die Suchergebnisse und stellt diese dann den für die Anzeige zuständigen JSP-Tags zur Verfügung.

Beispielkonfiguration:

```
<servlet>
          <servlet-name>SearchServlet</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.SearchServlet
          </servlet-class>
          <init-param>
                      <param-name>userPrefix</param-name>
                      <param-value>fs:user:</param-value>
          </init-param>
          <init-param>
                      <param-name>groupsPrefix</param-name>
                      <param-value>fs:group:</param-value>
          </init-param>
          <init-param>
                      <param-name>sendSecurityToken</param-name>
                      <param-value>false</param-value>
          </init-param>
          <init-param>
                      <param-name>connectTimeout</param-name>
                      <param-value>2000</param-value>
          </init-param>
          <init-param>
                      <param-name>readTimeout</param-name>
                      <param-value>20000</param-value>
          </init-param>
</servlet>

<servlet-mapping>
          <servlet-name>SearchServlet</servlet-name>
          <url-pattern>*.search</url-pattern>
</servlet-mapping>
```

Die Parameter `userPrefix` und `groupsPrefix` geben die Prefixe der Benutzer- und Gruppenrechte für Dokumente im Suchindex an und spielen somit nur dann eine Rolle, wenn Suchergebnisse personalisiert dargestellt werden sollen. Diese Parameter werden nur ausgewertet, wenn über den dritten Parameter

`sendSecurityToken` das Verwenden von SecurityToken bei der Suchanfrage aktiviert wurde.

Werden für Dokumente manuell Rechte vergeben (siehe auch Kapitel 4.2, 5.3.2.13 und 5.3.2.14), so müssen die dort verwendeten Prefixe den Werten der Parameter `userPrefix` und `groupsPrefix` entsprechen. Werden diese Parameter im `SearchServlet` nicht konfiguriert, so wird auf die Default-Werte `exalead:user:` bzw. `exalead:group:` zurückgegriffen.

Standardmäßig sind für Verbindungen vom SearchServlet zum Exalead-Server folgende Timeout-Werte konfiguriert: `connectTimeout = 2000 ms; readTimeout = 20000 ms.` Diese Werte können über die beiden gleichnamigen Parameter des Servlets überschrieben werden.

Innerhalb der Suchseite wird das Servlet anhand eines Formulars angesprochen.

Ein exemplarischer Aufruf des `SearchServlets` sieht wie folgt aus:

```
<form action="do.search">
<input type="text" name="q" value=""/>
<input type="hidden" name="b" value="0"/>
<input type="hidden" name="l" value="de"/>
<input type="hidden" name="hf" value="10"/>
<input type=”hidden” name=”r” value=”Top/source/intranet”/>
<input type=”hidden” name=”r” value=”Top/mime/text#html”/>
<input type="hidden" name="redirectUrl" value="search.jsp"/>
<input type="hidden" name="errorUrl" value="error.jsp"/>
<input type="submit" value="Suchen"/>
</form>
```

Dabei haben die Parameter folgende Bedeutung:

Parameter	Erwarteter Wert
q	Eingabefeld für das Suchwort
b	Index des ersten anzuzeigenden Treffers der Ergebnisliste
l	Sprache, die zur Interpretation des Suchwortes herangezogen werden soll
hf	Anzahl der anzuzeigenden Treffer pro Ergebnisseite
r	Feld für Suchverfeinerungen (optional), kann mehrfach angegeben werden
redirectUrl	Ergebnisseite (nur lokale URLs erlaubt)

errorUrl	Fehlerseite (nur lokale URLs erlaubt)

Sollte die Suche nicht erfolgreich durchgeführt werden können z.B. wegen Nichterreichbarkeit des Suchservers, so schreibt das `SearchServlet` den aufgetretenen Fehler in Form einer Exception in die Session-Variable `exception`. Diese kann somit in der Suchergebnisseite ausgelesen und dort eine passende Meldung ausgegeben werden.

3.7.2 SearchDownload

Das `SearchDownload`-Servlet dient dazu, Suchtreffer in den Suchergebnislisten über die Webapplikation auszuliefern, da auf diese nicht direkt vom Browser aus zugegriffen werden kann, weil sie zum Beispiel auf einem Netzwerk-Dateisystem liegen. Dazu fragt dieses Servlet die Datei beim Exalead-Server an und reicht sie dann an den Browser weiter.

Beispielkonfiguration:

```
<context-param>
          <param-name>getDocumentServlet</param-name>
          <param-value>/search/do.searchDownload</param-value>
</context-param>

<servlet>
          <servlet-name>SearchDownload</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.SearchDownload
          </servlet-class>
</servlet>

<servlet-mapping>
          <servlet-name>SearchDownload</servlet-name>
          <url-pattern>*.searchDownload</url-pattern>
</servlet-mapping>
```

Für die automatische Generierung der Links zum `SearchDownload`-Servlet muss der Applikation über den Context-Parameter `getDocumentServlet` die Adresse bekanntgemacht werden, unter der das `SearchDownload`-Servlet angesprochen werden kann.

Wie die Ausgabe der automatisch generierten Download-Links innerhalb der Suchergebnisseite gesteuert werden kann, wird in Kapitel 3.12.3 ab Seite 43 beschrieben.

3.7.3 ThumbnailServlet

Das `ThumbnailServlet` liefert die Thumbnails zu den Suchergebnistreffern aus, falls solche vorhanden sind. Im Zusammenspiel mit den entsprechenden JSP-Tags kann im Falle eines nicht vorhandenen Thumbnails auch ein Ersatzbild angegeben werden.

```
<servlet>
          <servlet-name>ThumbnailServlet</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.ThumbnailServlet
          </servlet-class>
</servlet>

<servlet-mapping>
          <servlet-name>ThumbnailServlet</servlet-name>
          <url-pattern>*.searchThumbnail</url-pattern>
</servlet-mapping>
```

3.7.4 AutocompleteServlet

Das `AutocompleteServlet` wird benötigt, um die Anbindung an den SuggestService von Exalead zu ermöglichen, der eine Autovervollständigung der eingegebenen Suchbegriffe über JavaScript erlaubt.

Über den Parameter `suggestservicename` wird der Name des SuggestServices angegeben, den man für die Autovervollständigung verwenden möchte. Wie man solch einen SuggestService in Exalead anlegt und konfiguriert, wird Kapitel 6 Schritt für Schritt erläutert.

```
<servlet>
          <servlet-name>AutocompleteServlet</servlet-name>
          <servlet-class>
          de.espirit.ps.exalead.servlets.AutocompleteServlet
          </servlet-class>
           <init-param>
                      <param-name>suggestServiceName</param-name>
                      <param-value>mysuggestservice</param-value>
          </init-param>
</servlet>

<servlet-mapping>
          <servlet-name>AutocompleteServlet</servlet-name>
          <url-pattern>*.autocomplete</url-pattern>
</servlet-mapping>
```

3.8 JSP-Tags

Mit den FirstSpirit Exalead JSP-Tags ist es möglich, die Suchergebnisse in nahezu jedem beliebigen Layout anzuzeigen. So kann z. B. für jeden Treffer ein Vorschaubild angezeigt werden, Treffer bestimmter Kategorien können hervorgehoben werden, es können Verfeinerungen (Drill-Down) für die Suchergebnisse angegeben werden u.v.m.

Die folgenden Exalead-Funktionalitäten werden zum aktuellen Zeitpunkt noch nicht unterstützt:

- Anzeige von verbundenen Begriffen
- Suche innerhalb von Suchergebnissen

Die dafür zur Verfügung stehenden JSP-Tags werden im Folgenden kurz beschrieben und anschließend im Detail behandelt.

Globale Tags:

- <search:container>
  Bereitstellung aller die Suchergebnisse betreffender Daten
  (siehe Kapitel 3.10.1 Seite 32).
- <search:query>
  Ausgabe des Suchbegriffs
  (siehe Kapitel 3.10.2 Seite 33).
- <search:nhits>
  Anzahl der Suchergebnisse
  (siehe Kapitel 3.10.3 Seite 33).
- <search:time_overall>
  Ausgabe der Dauer der Suchanfrage
  (siehe Kapitel 3.10.4 Seite 34).
- <search:sort_link> und <search:sort_linkparameter>
  Ausgabe eines Links zur Sortierung der Suchergebnisse nach Relevanz, Datum oder Größe
  (siehe Kapitel 3.10.5 Seite 34).
- <search:suggestions_loop>
  Iteration über Suchwortvorschläge
  (siehe Kapitel 3.10.7 Seite 36).
- <search:suggestions_link> und <search:suggestions_linkparameter>
  Ausgabe des Links zur Suche mit dem vorgeschlagenen Suchwort
  (siehe Kapitel 3.10.8 Seite 37).

- <search:suggestions_choice>
  Ausgabe des vorgeschlagenen Suchworts
  (siehe Kapitel 3.10.10 Seite 37).

Tags zur Verfeinerung der Suchergebnisse:

- <search:groups>
  Bereitstellung aller die Verfeinerung der Suchergebnisse betreffender Daten
  (siehe Kapitel 3.11.1 Seite 37).
- <search:group>
  Bereitstellung der Informationen einer bestimmten Suchkategorie
  (siehe Kapitel 3.11.2 Seite 37).
- <search:groups_categories_loop>
  Iteration über die Einträge einer bestimmten Suchkategorie
  (siehe Kapitel 3.11.3 Seite 38).
- <search:groups_category_title>
  Ausgabe des Titels des Eintrags
  (siehe Kapitel 3.11.4 Seite 38).
- <search:groups_category_count>
  Ausgabe der Trefferanzahl innerhalb dieses Eintrags
  (siehe Kapitel 3.11.5 Seite 39).
- <search:groups_category_link> und <search:groups_category_linkparameter>
  Ausgabe des Links zur Einschränkung der Suchergebnisse auf diesen Eintrag
  (siehe Kapitel 3.11.6 Seite 39).
- <search:reset_refinement>
  Bereitstellung der Informationen über getätigte Einschränkungen der Suchergebnisse
  (siehe Kapitel 3.11.8 Seite 40).
- <search:hasRefinements>
  Überprüft, ob Einschränkungen zu einer bestimmten Kategorie existieren
  (siehe Kapitel 3.11.9 Seite 41).
- <search:refinements>
  Bereitstellung der Informationen zu den Einschränkungen einer bestimmten Kategorie
  (siehe Kapitel 3.11.10 Seite 41).
- <search:reset_link> und <search:reset_linkparameter>
  Link zum Zurücksetzen der Einschränkung einer bestimmten Kategorie
  (siehe Kapitel 3.11.11 Seite 42).

Tags zur Anzeige der Suchergebnisse:

- <search:loop_hits>
  Iteriert über die Suchergebnisse
  (siehe Kapitel 3.12.1 Seite 43).
- <search:hits_id>
  Ausgabe der laufenden Nummer des Treffers (beginnend bei 0)
  (siehe Kapitel 3.12.2 Seite 43).
- <search:hits_url>
  Ausgabe der URL des Treffers
  (siehe Kapitel 3.12.3 Seite 43).
- <search:hits_title>
  Ausgabe der Überschrift des Treffers
  (siehe Kapitel 3.12.4 Seite 44).
- <search:hits_doctype>
  Ausgabe des Dokumententyps des Treffers
  (siehe Kapitel 3.12.5 Seite 45).
- <search:hits_score>
  Ausgabe des Rankings des Treffers
  (siehe Kapitel 3.12.6 Seite 46).
- <search:hits_summary>
  Ausgabe des Textauszuges des Treffers
  (siehe Kapitel 3.12.7 Seite 46).
- <search:hits_field>
  Ausgabe weiterer Indexfelder eines Treffers
  (siehe Kapitel 3.12.8 Seite 47).
- <search:hits_filesize>
  Ausgabe der Größe eines Treffers
  (siehe Kapitel 3.12.9 Seite 47).
- <search:hits_date>
  Ausgabe des Datums eines Treffers
  (siehe Kapitel 3.12.10 Seite 48).
- <search:hits_thumbnail>
  Umschliessendes Tag für den Bereich, in dem Thumbnails angezeigt werden sollen
  (siehe Kapitel 3.12.11 Seite 48).
- <search:hits_hasThumbnail> / <search:hits_hasNoThumbnail>
  Überprüft, ob ein Vorschaubild zum Treffer existiert
  (siehe Kapitel 3.12.12 Seite 48).
- <search:is_in_category> / <search:is_not_in_category>
  Überprüft, ob das Suchergebnis zu einer bestimmten Kategorie gehört
  (siehe Kapitel 3.12.14 Seite 49).

Tags für die Navigation:

- <search:navigation>
  Bereitstellung der Informationen für die Navigation
  (siehe Kapitel 3.13.1 Seite 50).
- <search:navigation_page_first_link> und
  <search:navigation_page_first_linkparameter>
  Ausgabe des Links zur ersten Ergebnisseite
  (siehe Kapitel 3.13.2 Seite 50).
- <search:navigation_page_previous_container>
  Überprüft, ob der Link zur vorherigen Ergebnisseite dargestellt werden soll
  (siehe Kapitel 3.13.4 Seite 51).
- <search:navigation_page_previous_link> und
  <search:navigation_page_previous_linkparameter>
  Ausgabe des Links zur vorherigen Ergebnisseite
  (siehe Kapitel 3.13.5 Seite 51).
- <search:navigation_loop_pages>
  Iteriert über die anzuzeigenden Seitenzahlen der Navigation
  (siehe Kapitel 3.13.7 Seite 51).
- <search:navigation_is_current_page>
  Überprüft, ob die angezeigte Seitenzahl der aktuell angezeigten Ergebnisseite entspricht
  (siehe Kapitel 3.13.8 Seite 52).
- <search:navigation_is_not_current_page>
  Überprüft, ob die angezeigte Seitenzahl nicht der aktuell angezeigten Ergebnisseite entspricht
  (siehe Kapitel 3.13.9 Seite 52).
- <search:navigation_page>
  Ausgabe der Seitenzahl
  (siehe Kapitel 3.13.10 Seite 52).
- <search:navigation_page_link> und
  <search:navigation_page_linkparameter>
  Ausgabe des Links der zur angezeigten Seitenzahl entsprechenden Ergebnisseite
  (siehe Kapitel 3.13.11 Seite 52).
- <search:navigation_page_next_container>
  Überprüft, ob der Link zur nächsten Ergebnisseite dargestellt werden soll
  (siehe Kapitel 3.13.13 Seite 53).
- <search:navigation_page_next_link> und
  <search:navigation_page_next_linkparameter>
  Ausgabe des Links zur nächsten Ergebnisseite

(siehe Kapitel 3.13.14 Seite 53).

- <search:navigation_page_last_container>
  Überprüft, ob der Link zur letzten Ergebnisseite dargestellt werden kann
  (siehe Kapitel 3.13.16 Seite 53).
- <search:navigation_page_last>
  Ausgabe der Seitenzahl der letzten Ergebnisseite
  (siehe Kapitel 3.13.17 Seite 54).
- <search:navigation_page_last_link> und
  <search:navigation_page_last_linkparameter>
  Ausgabe des Links zur letzten Ergebnisseite
  (siehe Kapitel 3.13.18 Seite 54).

Eine minimale Suchergebnisseite könnte damit z. B. folgendermaßen aussehen:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib prefix="search" uri="/WEB-INF/exalead.tld" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">



<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Suchergebnisse</title>

</head>

<body>

<form action="do.search">

<input type="text" name="q" value="<%= session.getAttribute("searchString") != null
? session.getAttribute("searchString") : "" %>"/>

<input type="hidden" name="b" value="0"/>

<input type="hidden" name="l" value="de"/>

<input type="hidden" name="hf" value="10"/>

<input type="hidden" name="redirectUrl" value="search.jsp"/>

<input type="hidden" name="errorUrl" value="search.jsp"/>

<input type="submit" value="Suchen"/>

</form>
```

```
<hr/>

<p>Suchergebnisse darstellen</p>

<search:container hasResults="true" redirectUrl="search.jsp" errorUrl="search.jsp"
hitsPerPage="10">

<table border="1">

<tr><th>Index</th><th>Title</th><th>URL</th><th>Summary</th></tr>

<search:loop_hits>

            <tr>

            <td><search:hits_id/></td>

            <td><a href="<search:hits_url completeDocUrl="true"/>">

                        <search:hits_title/>

            </a></td>

            <td><search:hits_url/></td>

            <td><search:hits_summary isLongForm="true" endline=" ...<br/>"
highlightStart="<strong>" highlightEnd="</strong>" /></td>

            </tr>

</search:loop_hits>

</table>

<br/>

<table border="1">

            <tr>

            <search:navigation surroundingPagesRadius="5">

            <td><a href="<search:navigation_page_first_link/>">

                        Erste Seite

            </a></td>

            <search:navigation_page_previous_container>

            <td><a href="<search:navigation_page_previous_link/>">

                        Vorherige Seite

            </a></td>

            </search:navigation_page_previous_container>

            <search:navigation_loop_pages>

            <td>
```

```
            <search:navigation_is_current_page>

                        <strong><search:navigation_page/></strong>

            </search:navigation_is_current_page>

            <search:navigation_is_not_current_page>

            <a href="<search:navigation_page_link/>">

            <search:navigation_page/></a>

            </search:navigation_is_not_current_page>

            </td>

            </search:navigation_loop_pages>

            <search:navigation_page_next_container>

            <td><a href="<search:navigation_page_next_link/>">

                        Nächste Seite

            </a></td>

            </search:navigation_page_next_container>

            <search:navigation_page_last_container>

            <td><a href="<search:navigation_page_last_link/>">

                        Letzte Seite (<search:navigation_page_last/>)

            </a></td>

            </search:navigation_page_last_container>

            </search:navigation>

            </tr>

</table>

</search:container>

<search:container hasResults="false" redirectUrl="index.jsp" errorUrl="index.jsp"
hitsPerPage="10">

<strong>Leider nichts gefunden...</strong><br/>

</search:container>

</body>

</html>
```

3.9 Tag-Präfix

Um die FirstSpirit Exalead-Tags verwenden zu können, muss in den JSP-Seiten die entsprechende Tag-Library angegeben werden. Diese Dokumentation verwendet das Präfix "search" für die FirstSpirit Exalead-Tags.

Beispiel für die Einbindung in JSP-Seiten:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/WEB-INF/exalead.tld" prefix="search" %>
```

Wird das Präfix "search" auf einen anderen Wert geändert, so ist dieses Präfix für die einzelnen Tags zu verwenden, d.h. "<**myPrefix**:ref>" anstelle von "<search:ref>".

3.10 Globale Tags

3.10.1 <search:container>

Das Tag <search:container> bereitet die Suchergebnisse auf, so dass sie von den anderen zur Verfügung stehenden Tags (Verfeinerungen, Treffer, Navigation) ausgewertet und ausgegeben werden können.

Attribute:

Attribut	Bedeutung	Pflichtparameter
hasResults	Über dieses Attribut wird gesteuert, ob der Inhalt des Tags bei vorhandenen Suchtreffern angezeigt werden soll (hasResults="true") oder ob der Inhalt im Falle keiner Suchtreffer angezeigt werden soll (hasResults="false").	Ja
redirectUrl	Dieses Attribut gibt die Adresse der Suchergebnisseite an. In der Regel ist dies die Seite, auf der man sich gerade befindet. Die Angabe wird benötigt, damit die automatisch generierten Links (z. B. für die Navigation) auf die korrekte Seite verweisen.	Ja

errorUrl	Dieses Attribut gibt die Adresse der Seite an, die im Fehlerfall angesteuert wird. Auch diese Angabe wird für die automatisch generierten Links benötigt.	Ja
hitsPerPage	Mit diesem Attribut gibt man die Anzahl der anzuzeigenden Treffer pro Seite an. Hat man eine solche Angabe bereits im Formular des SearchServlets gemacht, so muss hier derselbe Wert eingetragen werden.	nein

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp" hitsPerPage="20">
Anzeige der Suchtreffer
</search:container>
<search:container hasResults="false" redirectUrl=" search.jsp"
errorUrl=" error.jsp" hitsPerPage="20">
Keine Treffer gefunden
</search:container>
```

3.10.2 <search:query />

Das Tag <search:query/> gibt die aktuelle Suchanfrage aus.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Ihre Suche nach ‘<search:query />’ ergab <search:nhits /> Treffer.
</search:container>
```

3.10.3 <search:nhits />

Das Tag <search:nhits /> gibt die Anzahl der gefundenen Treffer aus.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Ihre Suche nach ‘<search:query />’ ergab <search:nhits /> Treffer.
```

```
</search:container>
```

3.10.4 <search:time_overall />

Das Tag <search:time_overall /> gibt die Dauer der Suchanfrage aus.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Ihre Suche nach '<search:query />' ergab <search:nhits /> Treffer
in <search:time_overall /> Millisekunden.
</search:container>
```

3.10.5 <search:sort_link />

Das Tag <search:sort_link /> gibt automatisch generierte Links zur Sortierung der Suchergebnisse nach beliebigen Metadaten aus. Dazu gibt man dem Link über ein Attribut das Sortierkriterium an. Das Umschalten zwischen aufsteigender und absteigender Sortierung geschieht automatisch und wird vom Tag übernommen.

Felder, die zur Sortierung genutzt werden sollen, müssen zuvor auf dem Exalead CloudVew Server entsprechend konfiguriert werden. Die Sortierung nach Relevanz (`score`), Datum (`lastmodifieddate`) und Größe (`file_size`) ist nach Installation des Servers bereits vorkonfiguriert. Um weitere Felder anzulegen, die zur Sortierung dienen sollen, kann in der Administrationsoberfläche des Exalead CoudView Servers der „Add a Field“-Wizard verwendet werden. Nach Vergabe eines Namens für das neue Feld muss bei „Create Index Field“ der Wert „Alphanum with sort“ ausgewählt werden. Der Wizard legt anschliessend zwei neue Felder an. Eines mit dem zuvor angegebenen Namen und eines mit dem Suffix `_sort`. Letzteres kann dann als Wert für das sort-Attribut des Tags angegeben werden.

Attribute:

Attribut	Bedeutung	Pflichtparameter
sort	Über dieses Attribut gibt man das Sortierkriterium für den auszugebenden Link an (`score`, `lastmodifieddate`, `file_size` oder beliebiges selbst-definiertes Feld)	Ja

Einfaches Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<a href="<search:sort_link sort="score"/>" >Nach Relevanz
sortieren</a>
<a href="<search:sort_link sort="date"/>" >Nach Datum
sortieren</a>
<a href="<search:sort_link sort="size"/>" >Nach Größe
sortieren</a>
</search:container>
```

Erweitertes Beispiel:

Möchte man den Link zur aktuellen Sortierreihenfolge ausblenden und darüber hinaus auch noch kenntlich machen, ob ein Klick auf einen Link aufsteigend oder absteigend sortieren lässt, so muss man vorher einmal die aktuellen Werte aus der Session holen:

```
<%  final String currentSort = (String)
session.getAttribute(de.espirit.ps.exalead.resources.SearchRequest
Arguments.SESSION_SORT);
final String currentOrder = (String)
session.getAttribute(de.espirit.ps.exalead.resources.SearchRequest
Arguments.SESSION_ORDER); %>
```

Mithilfe dieser Werte kann man dann die Ausgabe der Links präzisieren:

```
<% if ("date".equals(currentSort) || "size".equals(currentSort)) {
%>
<!-- Ausgabe mit Verlinkung -->
<a href="<search:sort_link sort="score"/>" >Nach Relevanz</a>
<% } else { %>
<!-- Ausgabe ohne Verlinkung, da bei der Suche nach Relevanz keine
Umkehrung der Sortierreihenfolge erwünscht ist -->
<b>Nach Relevanz</b>
<% }

 if (!"date".equals(currentSort)) { %>
<!-- Ausgabe ohne Angabe der Sortierreihenfolge, da initial immer
absteigend sortiert wird -->
<a href="<search:sort_link sort="date"/>" >Nach Datum</a>
<% } else { if ("1".equals(currentOrder)) { %>
<!-- Wurde bereits nach Datum sortiert, kehrt ein erneuter Klick
auf den Link die Sortierreihenfolge um -->
<b>Nach Datum</b> <a href="<search:sort_link sort="date"/>"
>abwärts</a>
<% } else { %>
<!-- Wurde bereits nach Datum sortiert, kehrt ein erneuter Klick
auf den Link die Sortierreihenfolge um -->
<b>Nach Datum</b> <a href="<search:sort_link sort="date"/>">
aufwärts</a>
```

```
<% } }
   if (!"size".equals(currentSort)) { %>
<!-- Ausgabe ohne Angabe der Sortierreihenfolge, da initial immer
absteigend sortiert wird -->
<a href="<search:sort_link sort="size"/>" >Nach Größe</a>
<% } else { if ("1".equals(currentOrder)) { %>
<!-- Wurde bereits nach Größe sortiert, kehrt ein erneuter Klick
auf den Link die Sortierreihenfolge um -->
<b>Nach Größe</b> <a href="<search:sort_link sort="size"/>" >
abwärts</a>
<% } else { %>
<!-- Wurde bereits nach Größe sortiert, kehrt ein erneuter Klick
auf den Link die Sortierreihenfolge um -->
<b>Nach Größe</b> <a href="<search:sort_link sort="size"/>" >
aufwärts</a>
<% } } %>
```

3.10.6 <search:sort_linkparameter />

Die Funktionsweise entspricht der des <search:sort_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servletaufruf generiert.

Beispiel:

```
<a href="do.search?<search:sort_linkparameter sort="date"/>" >Nach
Datum sortieren</a>
```

3.10.7 <search:suggestions_loop>

Das <search:suggestions_loop>-Tag iteriert über alle Suchwortvorschläge und stellt den jeweiligen Vorschlag mit dem dazugehörigen Suchlink zur Verfügung.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<% if ("true".equals(pageContext.getAttribute("hasSuggestions")))
{ %>
Meinten Sie
<search:suggestions_loop>
<a href="<search:suggestions_link/>">
<search:suggestions_choice/>
</a>
</search:suggestions_loop> ?
<% } %>
</search:container>
```

3.10.8 <search:suggestions_link />

Das Tag <search:suggestions_link/> blendet den Link zur Suche mit dem vorgeschlagenen Suchwort ein.

3.10.9 <search:suggestions_linkparameter />

Die Funktionsweise entspricht der des <search:suggestions_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-Aufruf generiert.

3.10.10 <search:suggestions_choice />

Das Tag <search:suggestions_choice/> blendet das vorgeschlagene Suchwort ein.

3.11 Tags zur Verfeinerung der Suchergebnisse

3.11.1 <search:groups>

Das <search:groups>-Tag definiert den Bereich für die Anzeige der vorhandenen Suchkategorien und der getätigten Verfeinerungen der Suche.

3.11.2 <search:group>

Das <search:group>-Tag wird innerhalb des <search:groups>-Tags aufgerufen und enthält die vorhandenen Einträge einer bestimmten Suchkategorie, die dem Tag per Attribut mitgeteilt wird. Enthält diese Suchkategorie keine Einträge, so blendet das Tag seinen kompletten Inhalt aus.

Attribute:

Attribut	Bedeutung	Pflichtparameter
groupID	Name der Suchkategorie, deren Einträge angezeigt werden sollen. Sind keine Einträge vorhanden, wird der Inhalt ausgeblendet.	Ja

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<search:groups>
          <search:group groupID="Kategorie1">
                    Anzeige der Einträge in Kategorie1
          </search:group/>
          <search:group groupID="Kategorie2">
                    Anzeige der Einträge in Kategorie2
          </search:group/>
</search:groups>
</search:container>
```

3.11.3 <search:groups_categories_loop>

Das <search:groups_categories_loop>-Tag iteriert über alle Einträge einer Suchkategorie und stellt die anzeigbaren Informationen dieser Einträge zur Verfügung.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<search:groups>
          <search:group groupID="Kategorie1">
                    <search:groups_categories_loop>
<a href="<search:groups_category_link/>">
<search:groups_category_title/>
</a>
(<search:groups_category_count/>)
<a href="<search:groups_category_link action="exclude"/>">
ausschliessen
</a>
                    </search:groups_categories_loop>
          </search:group/>
</search:groups>
</search:container>
```

Innerhalb dieses Tags kann über die PageContext-Variable „level“ der Einrückungsgrad der Kategorie bestimmt werden.

3.11.4 <search:groups_category_title />

Dieses Tag gibt den Titel des Eintrags aus.

Attribute:

Attribut	Bedeutung	Pflichtparameter
trimAt	Zeichen oder Zeichenkette, die die Stelle im Titel angibt, ab welcher dieser angezeigt werden soll. Beispiel: Der ungekürzte Titel lautet text#html. Mit trimAt="#" wird nur html als Titel angegeben.	Nein

Anstelle des Tags kann innerhalb des group_categories_loop-Tags der Titel auch über die PageContext-Variable „title"ausgegeben werden.

Beispiel siehe Kapitel 3.11.3 Seite 38.

3.11.5 <search:groups_category_count />

Dieses Tag gibt die Anzahl der Suchtreffer zu diesem Eintrag aus.

Anstelle des Tags kann innerhalb des group_categories_loop-Tags die Anzahl der Suchtreffer auch über die PageContext-Variable „count" ausgegeben werden.

Beispiel siehe Kapitel 3.11.3 Seite 38.

3.11.6 <search:groups_category_link />

Dieses Tag gibt den Link zum Einschränken der Suchergebnisse auf diesen Eintrag bzw. zum Ausschließen der Suchergebnisse dieses Eintrags aus.

Attribute:

Attribut	Bedeutung	Pflichtparameter
action	Bei action="include" (Default-Wert) wird der Link zum Einschränken der Suchergebnisse auf diesen Eintrag ausgegeben. Bei action="exclude" wird der Link zum Ausschließen der Suchergebnisse dieses Eintrags ausgegeben.	Nein

Beispiel siehe Kapitel 3.11.3 Seite 38.

3.11.7 <search:groups_category_linkparameter />

Die Funktionsweise entspricht der des <search:groups_category_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-Aufruf generiert.

3.11.8 <search:reset_refinement>

Das <search:reset_refinements>-Tag definiert den Bereich für die Anzeige der getätigten Verfeinerungen der Suche. Der Inhalt des Tags wird ausgeblendet, wenn keine Verfeinerungen getätigt wurden.

Wurden Verfeinerungen getätigt und der Inhalt des Tags eingeblendet, so kann man über die Pagecontext-Variable "resetUrl" einen Link zum Entfernen aller getätigter Verfeinerungen bekommen. Alternativ können über die Pagecontext-Variable "resetUrlParameter" auch nur die Parameter des Reset-Links abgefragt werden.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<search:groups>
          <search:reset_refinement>
<a href="<%= resetUrl %>">Reset all refinements</a>
          </search:reset_refinement>
</search:groups>
</search:container>
```

3.11.9 <search:hasRefinements>

Das <search:hasRefinements>-Tag wird innerhalb des <search:reset_refinement>-Tags aufgerufen und enthält die getätigten Verfeinerungen einer bestimmten Suchkategorie, die dem Tag per Attribut mitgeteilt wird. Wurden für diese Suchkategorie keine Verfeinerungen getätigt, so blendet das Tag seinen kompletten Inhalt aus.

Attribute:

Attribut	Bedeutung	Pflichtparameter
groupID	Name der Suchkategorie, deren Verfeinerungen angezeigt werden sollen. Sind keine Verfeinerungen vorhanden, wird der Inhalt ausgeblendet.	Ja

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<search:groups>
          <search:reset_refinement>
          <search:hasRefinements groupID="Kategorie1">
                    Anzeige der Verfeinerungen der Kategorie1
          </search:hasRefinements />
          <search:hasRefinements groupID="Kategorie2">
                    Anzeige der Verfeinerungen der Kategorie2
          </search:hasRefinements />
          </search:reset_refinement>
</search:groups>
</search:container>
```

3.11.10 <search:refinements>

Das <search:refinements>-Tag iteriert über alle getätigten Verfeinerungen einer Suchkategorie.

Attribute:

Attribut	Bedeutung	Pflichtparameter
trimAt	Zeichen oder Zeichenkette, die die Stelle im Titel angibt, ab welcher dieser angezeigt werden soll. Beispiel: Der ungekürzte Titel lautet text#html. Mit trimAt="#" wird nur html als Titel angegeben.	Nein

Innerhalb des Tags kann über die Pagecontext-Variablen „isExcluded" und „isIncluded" überprüft werden, ob es sich bei der Verfeinerung um eine Einschränkung oder um eine Ausschlussregel handelt. Die Variable „title" liefert den Namen der Verfeinerung zurück. Über die PageContext-Variable „level" kann der Einrückungsgrad der Kategorie bestimmt werden.

Beispiel:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<search:groups>
          <search:reset_refinement>
          <search:hasRefinements groupID="Kategorie1">
                    <search:refinements>
<%= isExcluded ? "NICHT " : "" %><%= title %>
<a href="<search:reset_link/>">entfernen</a>
                    </search:refinements>
          </search:hasRefinements />
          </search:reset_refinement>
</search:groups>
</search:container>
```

3.11.11 <search:reset_link />

Das <search:reset_link />-Tag liefert den Link zum Zurücksetzen der Verfeinerung zurück.

Beispiel siehe Kapitel 3.11.10 Seite 41.

3.11.12 <search:reset_linkparameter />

Die Funktionsweise entspricht der des <search:reset_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-

Aufruf generiert.

3.12 Tags zur Anzeige der Suchergebnisse

3.12.1 <search:loop_hits>

Das <search:loop_hits>-Tag iteriert über alle Suchtreffer der aktuell angezeigten Ergebnisseite und stellt die anzeigbaren Informationen dieser Suchtreffer zur Verfügung.

3.12.2 <search:hits_id />

Dieses Tag gibt den laufenden Index des Suchtreffers innerhalb der Trefferliste aus. Der erste Suchtreffer besitzt die ID 0.

Beispiel:

```
<search:loop_hits>
          ID des Suchtreffers: <search:hits_id/>
</search:loop_hits>
```

3.12.3 <search:hits_url />

Dieses Tag gibt die URL des Suchtreffers zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
completeDocUrl	Auf "false" (Standardwert) gesetzt, liefert das Tag die URL des Dokuments so zurück, wie sie im Index gespeichert ist. Diese Links können allerdings bei Dateien, die nicht per http erreichbar sind, nicht ohne Weiteres geöffnet werden. Mit completeDocUrl="true" werden Links zu Dokumenten, die nicht per http erreichbar sind, um den Pfad des DownloadServlets erweitert, so dass diese Dokumente ausgeliefert werden können.	Nein
replaceRule	Ersetzungsregel, um einen Teil der URL durch einen anderen Wert zu ersetzen, bspw. einen anderen Server. Syntax: "<alter String>,<neuer String>"	Nein

Beispiel:

```
<search:loop hits>
URL des Dokuments: <a href=”<search:hits_url completeDocUrl="true"
replaceRule=”myIndexServer,myWebsiteServer”/>”>
<search:hits_url replaceRule=”myIndexServer,myWebsiteServer”/>
</a>
</search:loop_hits>
```

3.12.4 <search:hits_title />

Dieses Tag liefert den Titel des Suchtreffers zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
highlightStart / highlightEnd	Mit diesen Attributen kann man die Art der Hervorhebung des Suchwortes innerhalb des Titels bestimmen.	Nein

Beispiel:

```
<search:loop_hits>
Titel: <search:hits_title highlightStart="<strong>"
highlightEnd="</strong>" />
</search:loop_hits>
```

3.12.5 <search:hit_doctype />

Dieses Tag liefert in der PageContext-Variable „doctype“ den Dokumententyp des Suchtreffers zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
completeDoctype	Bei Angabe von true, wird der komplette Dokumententyp zurückgegeben. Bei Angabe von false, nur der Teil hinter dem ersten ‚/’.	Nein

Beispiel:

```
<search:loop_hits>
Typ:
<search:hits_doctype completeDoctype="false">
          <% if ("html".equals(doctype)) { %>[HTML]<% }
             else if ("pdf".equals(doctype)) { %>[PDF]<% }
             else { %>[unknown]<% } %>
       </search:hits_doctype>
</search:loop_hits>
```

3.12.6 <search:hits_score />

Dieses Tag liefert die Punktzahl des Suchtreffers zurück.

Beispiel:

```
<search:loop_hits>
          Typ: <search:hits_score />
</search:loop_hits>
```

3.12.7 <search:hits_summary />

Dieses Tag liefert einen Auszug des Suchtreffers zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
isLongForm	Über dieses Attribut wird gesteuert, ob nur die erste Zeile des Auszuges (isLongForm = false) oder ob der komplette Auszug (isLongForm = true) dargestellt werden soll. Der Standardwert ist false.	Nein
endline	Mit diesem Attribut kann man einen Textschnipsel definieren, der an das Ende jeder Zeile des Auszuges gehängt wird (z. B. "... ", um jede Zeile mit drei Punkten und einem Zeilenumbruch zu beenden).	Nein
highlightStart / highlightEnd	Mit diesen Attributen kann man die Art der Hervorhebung des Suchwortes innerhalb des Textauszuges bestimmen.	Nein

Beispiel:

```
<search:loop_hits>
<search:hits_summary isLongForm="true" endline=" ...<br/>"
highlightStart="<strong>" highlightEnd="</strong>" />
```

```
</search:loop_hits>
```

3.12.8 <search:hits_field />

Mit diesem Tag können zusätzliche Indexfelder ausgelesen werden (z. B. Metadaten aus FirstSpirit, siehe Kapitel 5.2.3, oder Metadaten, die im HTML-Header einer Seite angegeben wurden). Die auslesbaren Felder müssen zuvor über Exalead definiert werden. Sind einem Feld mehrere Werte zugeordnet, so werden diese als kommaseparierte Liste zurückgegeben.

Attribute:

Attribut	Bedeutung	Pflichtparameter
Key	Name des Indexfeldes, das ausgegeben werden soll.	Ja

Beispiel:

```
<search:loop_hits>
<search:hits_field key="myCustomAttribute">
          <%= value %>
</search:hits_field>
</search:loop_hits>
```

3.12.9 <search:hits_filesize />

Dieses Tag gibt die Größe des Dokuments zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
type	Einheit, in der die Größe ausgegeben werden soll. Mögliche Werte: b, kb, mb, gb. Wird dieses Attribut weggelassen, so wird automatisch eine passende Einheit ausgewählt.	Nein

Beispiel:

```
<search:loop_hits>
Dateigröße: <search:hits_filesize type=”kb” />
</search:loop_hits>
```

3.12.10 <search:hits_date />

Dieses Tag liefert das Erstellungsdatum des Dokuments zurück.

Attribute:

Attribut	Bedeutung	Pflichtparameter
format	Das Ausgabeformat des Datums, z. B. dd.MM.yy	Ja

Beispiel:

```
<search:loop_hits>
Datum: <search:hits_date format="dd.MM.yy" />
</search:loop_hits>
```

3.12.11 <search:hits_thumbnail>

Sollen in einer Webseite Thumbnails für die Suchtreffer dargestellt werden, so muss der entsprechende Bereich von diesem Tag umschlossen werden.

Beispiel:

```
<search:loop_hits>
<search:hits_thumbnail>
          <search:hits_hasThumbnail>
          <img src="/exalead/do.searchThumbnail?index=<%= index
%>" alt="" border="0" />
          </search:hits_hasThumbnail>
          <search:hits_hasNoThumbnail>
          <img src="images/no thumbnail.gif" alt="" border="0" />
          </search:hits_hasNoThumbnail></a>
</search:hits_thumbnail>
</search:loop_hits>
```

3.12.12 <search:hits_hasThumbnail>

Dieses Tag überprüft, ob für einen Suchtreffer ein Vorschaubild vorhanden ist. Ist

dies der Fall, so wird der Inhalt des Tags eingeblendet, ansonsten ausgeblendet.

Beispiel siehe Kapitel 3.12.11 Seite 48.

3.12.13 <search:hits_hasNoThumbnail>

Dieses Tag überprüft, ob für einen Suchtreffer ein Vorschaubild vorhanden ist. Ist dies **nicht** der Fall, so wird der Inhalt des Tags eingeblendet, ansonsten ausgeblendet.

Beispiel siehe Kapitel 3.12.11 Seite 48.

3.12.14 <search:is_in_category>

Mit diesem Tag kann überprüft werden, ob ein Suchtreffer einer bestimmten Kategorie zugehört. Ist dies der Fall, wird der Inhalt des Tags eingeblendet.

Attribute:

Attribut	Bedeutung	Pflichtparameter
fullPath	Der absolute Pfad zur Kategorie.	Ja

Beispiel:

```
<search:loop_hits>
<search:is_in_category fullPath="Top/FirstSpirit/common">
Dieser Treffer gehört zur Kategorie FirstSpirit/common
</search:is_in_category>
<search:is_not_in_category fullPath="Top/FirstSpirit/common">
Dieser Treffer gehört nicht zur Kategorie FirstSpirit/common
</search:is_not_in_category>
</search:loop_hits>
```

3.12.15 <search:is_not_in_category>

Mit diesem Tag kann überprüft werden, ob ein Suchtreffer einer bestimmten Kategorie zugehört. Der Inhalt des Tags wird eingeblendet, wenn dies **nicht** der Fall ist.

Attribute:

Attribut	Bedeutung	Pflichtparameter
fullPath	Der absolute Pfad zur Kategorie.	Ja

Beispiel siehe Kapitel 3.12.14 Seite 49.

3.13 Tags für die Navigation

3.13.1 <search:navigation>

Das <search:navigation>-Tag definiert den Bereich zur Anzeige der Navigation durch die Suchergebnisseiten.

Attribute:

Attribut	Bedeutung	Pflichtparameter
surroundingPagesRadius	Definiert den Seitenradius der Navigation um die aktuelle Seite. Wenn kein Wert oder ein Wert kleiner als 5 angegeben wird, wird ein Radius von 5 verwendet.	Nein

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
... Navigationsleiste ...
</search:navigation>
```

3.13.2 <search:navigation_page_first_link />

Dieses Tag generiert den Link zur ersten Ergebnisseite.

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
<a href="<search:navigation_page_first_link />">Erste Seite</a>
```

```
</search:navigation>
```

3.13.3 <search:navigation_page_first_linkparameter />

Die Funktionsweise entspricht der des <search:navigation_page_first_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-Aufruf generiert.

3.13.4 <search:navigation_page_previous_container>

Dieses Tag überprüft, ob es sich bei der aktuellen Suchergebnisseite um die erste Seite handelt, und gibt den Inhalt des Tags nur dann aus, wenn dies nicht zutrifft.

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
<search:navigation_page_previous_container>
          <a href="<search:navigation_page_previous_link/>">
                    vorherige Seite
          </a>
</search:navigation_page_previous_container>
</search:navigation>
```

3.13.5 <search:navigation_page_previous_link />

Dieses Tag generiert den Link zur vorherigen Seite in der Ergebnisseiten-Navigation.

Beispiel siehe Kapitel 3.13.4 Seite 51.

3.13.6 <search:navigation_page_previous_linkparameter />

Die Funktionsweise entspricht der des <search:navigation_page_previous_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-Aufruf generiert.

3.13.7 <search:navigation_loop_pages>

Dieses Tag iteriert abhängig vom eingestellten Seitenradius über die Suchergebnisseiten der Navigation.

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
```

```
<search:navigation_loop_pages>
          <search:navigation_is_current_page>
                    <strong><search:navigation_page/></strong>
          </search:navigation is current page>
          <search:navigation_is_not_current_page>
                    <a href="<search:navigation_page_link/>">
                    <search:navigation_page/>
                    </a>
          </search:navigation_is_not_current_page>
</search:navigation_loop_pages>
</search:navigation>
```

3.13.8 <search:navigation_is_current_page>

Dieses Tag überprüft, ob es sich bei der Seite aus der Iteration um die aktuell angezeigte Suchergebnisseite handelt. Ist dies der Fall, wird der Inhalt des Tags eingeblendet.

Beispiel siehe Kapitel 3.13.7 Seite 51.

3.13.9 <search:navigation_is_not_current_page>

Dieses Tag überprüft, ob es sich bei der Seite aus der Iteration **nicht** um die aktuell angezeigte Suchergebnisseite handelt. Ist dies der Fall, wird der Inhalt des Tags eingeblendet.

Beispiel siehe Kapitel 3.13.7 Seite 51.

3.13.10 <search:navigation_page />

Dieses Tag gibt die Seitenzahl der Suchergebnisseite aus.

Beispiel siehe Kapitel 3.13.7 Seite 51.

3.13.11 <search:navigation_page_link />

Dieses Tag generiert den Link zur Suchergebnisseite.

Beispiel siehe Kapitel 3.13.7 Seite 51.

3.13.12 <search:navigation_page_linkparameter />

Die Funktionsweise entspricht der des <search:navigation_page_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den

benötigten Servletaufruf generiert.

3.13.13 <search:navigation_page_next_container>

Dieses Tag überprüft, ob es sich bei der aktuellen Suchergebnisseite um die letzte Seite handelt, und gibt den Inhalt des Tags nur dann aus, wenn dies nicht zutrifft.

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
<search:navigation_page_next_container>
          <a href="<search:navigation_page_next_link/>">
                    nächste Seite
          </a>
</search:navigation_page_next_container>
</search:navigation>
```

3.13.14 <search:navigation_page_next_link />

Dieses Tag generiert den Link zur nächsten Seite in der Ergebnisseiten-Navigation.

Beispiel siehe Kapitel 3.13.13 Seite 53.

3.13.15 <search:navigation_page_next_linkparameter />

Die Funktionsweise entspricht der des <search:navigation_page_next_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servletaufruf generiert.

3.13.16 <search:navigation_page_last_container>

Dieses Tag überprüft, ob die Seitenzahl der letzten Seite bekannt ist. Bei einer sehr großen Menge an Treffern kann es vorkommen, dass die Anzahl der Treffer und somit auch die Anzahl der Ergebnisseiten nur geschätzt werden kann. In diesem Fall wird der Inhalt dieses Tags ausgeblendet.

Beispiel:

```
<search:navigation surroundingPagesRadius="5">
<search:navigation_page_last_container>
          <a href="<search:navigation_page_last_link/>">
          Letzte Seite (<search:navigation_page_last/>)
          </a>
</search:navigation_page_last_container>
```

```
</search:navigation>
```

3.13.17 <search:navigation_page_last />

Dieses Tag gibt die Seitenzahl der letzten Suchergebnisseite aus.

Beispiel siehe Kapitel 3.13.16 Seite 53.

3.13.18 <search:navigation_page_last_link />

Dieses Tag generiert den Link zur letzten Suchergebnisseite.

Beispiel siehe Kapitel 3.13.16 Seite 53.

3.13.19 <search:navigation_page_last_linkparameter />

Die Funktionsweise entspricht der des <search:navigation_page_last_link/>-Tags, mit dem einzigen Unterschied, dass dieses Tag nur die URL-Parameter für den benötigten Servlet-Aufruf generiert.

3.14 Implizite Suche

3.14.1 <search:loop_tophits>

Das <search:loop_tophits>-Tag dient der Anzeige der Top N Suchergebnisse zu einem bestimmten vorgegebenen Suchwort oder zur aktuellen Suchanfrage, jedoch unabhängig von etwaigen Suchverfeinerungen.

Attribute:

Attribut	Bedeutung	Pflichtparameter
query	Angabe des Suchwortes. Wenn keine Angabe gemacht wird, wird das aktuelle Suchwort der normalen Suche übernommen.	Nein
filetype	Einschränkung auf Suchergebnisse eines bestimmten Dateityps.	Nein

numberOfHits	Maximale Anzahl der angezeigten Treffer. Default-Wert: 5	Nein

Beispiel:

```
<search:loop_tophits query="internet" filetype="pdf"
numberOfHits="3">
<a href="<search:hits_url completeDocUrl="true"/>">
          <search:hits_title/>
</a><br/>
</search:loop_tophits>
```

Innerhalb des <search:loop_tophits>-Tags sind die gleichen Tags verwendbar wie innerhalb des <search:loop_hits>-Tags, mit Ausnahme der Anzeige von Thumbnails.

3.15 Verwendung von POST-Requests

Die zuvor beschriebenen Tags unterstützen in erster Linie die Verwendung von GET-Requests zur Verfeinerung, Sortierung und Navigation durch die Suchergebnisse. Dazu generiert ein Teil der Tags spezifische Links, die die Suche in gewünschter Form beeinflussen. Sollte die Arbeit mit GET-Requests aus technischen Gründen nicht möglich sein, können diese Tags auch in Kombination mit Formularen genutzt werden. Auf diese Weise erfolgt die Sortierung, Verfeinerung und Navigation ausschließlich über POST-Requests.

Beispiel: Verfeinerung durch GET-Request

```
<search:groups_categories_loop>
   <a href="do.search?<search:groups_category_linkparameter/>">
      <search:groups_category_title/>
   </a>
</search:groups_categories_loop>
```

Beispiel: Verfeinerung durch POST-Request

```
<search:groups_categories_loop>
   <form action="do.search" method="post" accept-charset="UTF-8">
      <input type="hidden" name="params"
      value="<search:groups_category_linkparameter/>"/>
      <input type="hidden" name="redirectUrl" value="search.jsp"/>
      <input type="hidden" name="errorUrl" value="search.jsp"/>
      <input type="submit"
      value="<search:groups_category_title/>"/>
   </form>
</search:groups_categories_loop>
```

4 Personalisierte Suchergebnisse

Um eine personalisierte Anzeige der Suchergebnisse zu erzielen, bietet Exalead die Möglichkeit, Benutzer- und Gruppeninformationen aus Dokumenten auszulesen und diese Dokumente nur berechtigten Benutzern in der Liste der Suchergebnisse anzuzeigen.

Dieses Kapitel befasst sich mit den Möglichkeiten, Dokumente mit Rechten zu versehen. Im Zusammenhang damit wird außerdem die Nutzung der Push-API erläutert, die es erlaubt, Dokumente einzeln in den Index zu übertragen.

Damit personalisierte Suchergebnisse angezeigt werden können, muss das SearchServlet bei jeder Anfrage ein entsprechendes SecurityToken mitschicken. Dies muss in der web.xml aktiviert werden, da die Suchanfragen standardmäßig ohne SecurityToken durchgeführt werden. Siehe dazu Kapitel 3.7.1 Seite 20.

Bei Suchanfragen ohne SecurityToken werden alle gefundenen Dokumente, unabhängig von den vergebenen Rechten, als Suchergebnis zurückgeliefert. Dies ist allerdings nur der Fall, solang die Exalead Search API nicht geschützt und somit ohne SecurityToken verwendbar ist. Um dieses Verhalten zu ändern, kann die Exalead Search API so konfiguriert werden, dass das Verwenden von SecurityTokens erzwungen wird (Abbildung 4-1: Sichern der Search API).

4.1 Personalisierte Suche auf HTML-Seiten

Eine einfache Möglichkeit, HTML-Seiten mit Rechten zu versehen, besteht in der Angabe von Meta-Informationen, die von Exalead ausgelesen und zur Bestimmung der Benutzer- und Gruppenrichtlinien herangezogen werden.

4.2 Vorbereiten der HTML-Dateien / Templates

Die Gruppen bzw. Benutzer werden in der HTML-Datei als MetaTag in folgender Form eingetragen:

```
<meta name="security"
content="exalead:group:BeliebigeGruppe,exalead:user:Jane Doe" />
```

Der Name des MetaTags kann dabei beliebig gewählt werden (im Beispiel: security). Über das Attribut `content` werden die Rechte für dieses Dokument vergeben. Diese sollten immer aus einem Prefix und dem Benutzer- bzw. Gruppennamen

bestehen. **Wichtig:** Der Prefix für Gruppen (hier `exalead:group:`) sollte sich dabei vom Prefix für Benutzernamen (hier `exalead:user:`) unterscheiden.

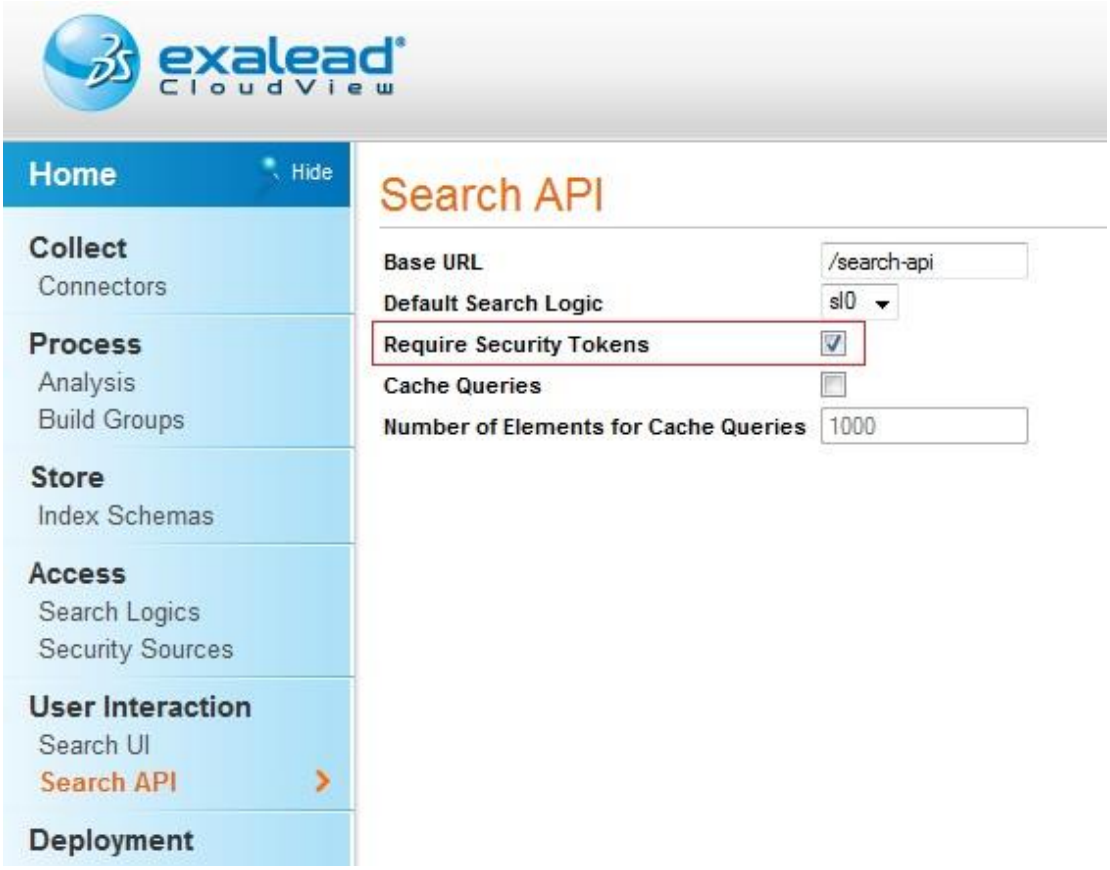

Abbildung 4-1: Sichern der Search API

Die Prefixe können – bis auf eine Ausnahme – beliebig gewählt werden (siehe dazu auch Kapitel 3.7.1).

Das MetaTag darf im Dokument nur einmal vorkommen. Mehrere Gruppen- und Benutzerrechte können innerhalb des MetaTags als kommaseparierte Liste angegeben werden:

```
<meta name="security"
content="exalead:group:Redakteur,exalead:group:Gast,exalead:group:
Admin,exalead:user:John Doe,exalead:user:Jane Doe" />
```

Seiten, die jedem Benutzer in den Suchergebnissen angezeigt werden sollen, müssen explizit zusätzlich der Gruppe `fs_public` zugeordnet werden:

```
<meta name="security" content="
exalead:group:Redakteur,exalead:group:Admin,exalead:group:fs_publi
c" />
```

Zur Erinnerung: Diese Rechte werden nur bei aktivierter Verwendung eines SecurityTokens ausgewertet (zur Konfiguration der Verwendung des SecurityTokens siehe Kapitel 3.7.1 "SearchServlet").

4.3 Einrichten der Document Processoren

Nachdem die HTML-Dateien entsprechend angepasst wurden, folgt nun die Einrichtung zweier Document Processoren in Exalead, die zum einen die einzelnen Werte der im MetaTag angegebenen kommaseparierten Liste auslesen und zum anderen diese Werte dem Dokument als Security-Information zuweisen (siehe Abbildung 4-2: Konfigurieren der Document Processoren).

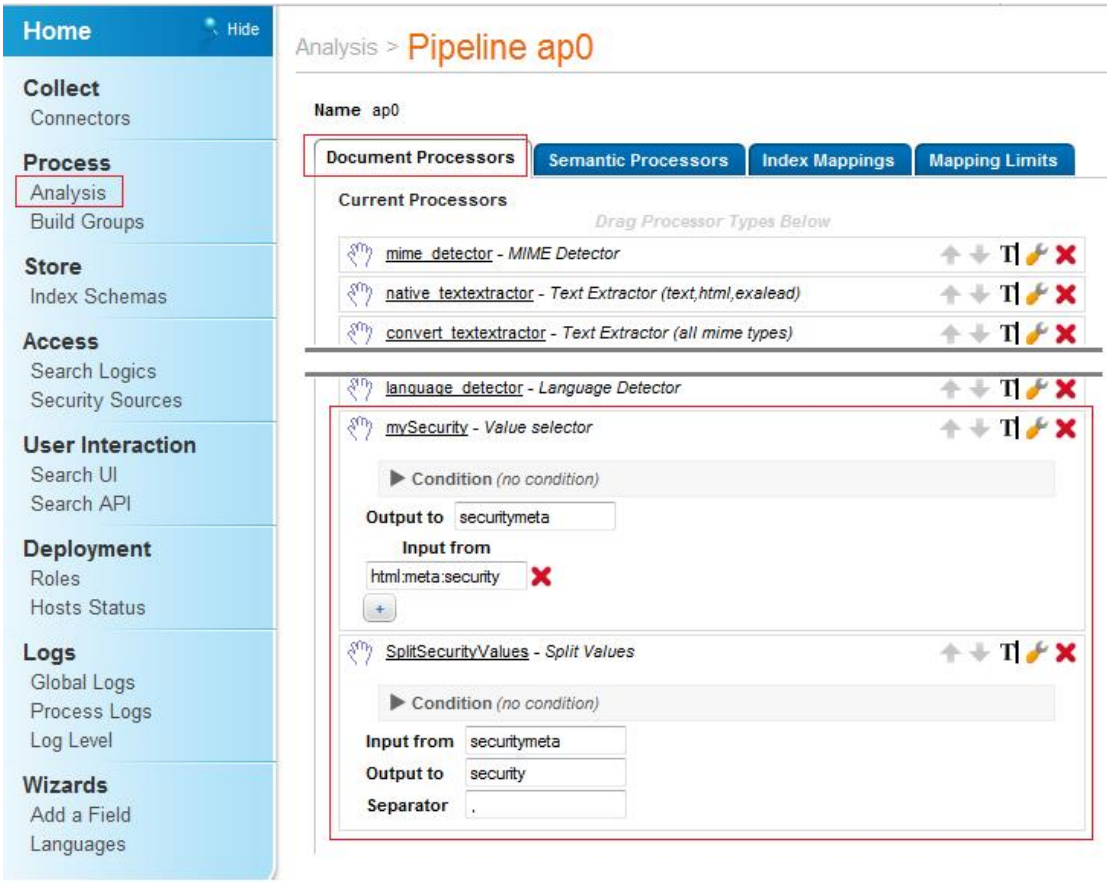

Abbildung 4-2: Konfigurieren der Document Processoren

Folgende Schritte sind dazu vorzunehmen:

1. Anlegen eines neuen Document Processors vom Typ "Value selector" (Kategorie "Chunk Operations"). Der Wert für "Output to" kann beliebig gewählt werden. Der Wert für "Input from" muss `html:meta:` gefolgt vom Namen des MetaTags aus der HTML-Seite lauten.

2. Anlegen eines neuen Document Processors vom Typ "Split Values" (Kategorie "Text Operations"). Der Wert für "Input from" muss hier dem Wert für "Output to" aus dem Value selector entsprechen. Der Wert für "Output to" muss hier `security` lauten. Der Separator muss dem Trennzeichen im MetaTag der HTML-Seite entsprechen.

Nach Abschluss dieser Schritte werden nun beim zukünftigen Indizieren der Dokumente die neu angelegten Document Processoren angewendet, welche die Benutzer und Gruppen aus dem ausgewiesenen Security-MetaTag auslesen und dem Dokument als Benutzer- und Gruppenrichtlinien zuweisen.

5 Inkrementelles Update über Push-API

Die Push-API von Exalead stellt eine Schnittstelle zur Verfügung, mit deren Hilfe es möglich ist, Dokumente direkt in den Index der Suchmaschine zu schreiben ("pushen"). Dadurch können dem Index Änderungen an Dokumenten sofort bekanntgemacht werden, ohne auf den nächsten Crawling-Zeitpunkt warten zu müssen.

Im Folgenden werden die für die Konfiguration der Push-API notwendigen Schritte erläutert. Anschließend wird das auf der Push-API aufbauende FirstSpirit Teilmodul "Exalead Content-Update" beschrieben.

5.1 Einrichten eines Push-Connectors

Zunächst muss ein neuer Push-Connector auf dem Exalead-Server eingerichtet werden (siehe Abbildung 5-1: Einrichten eines Push-Connectors). Der Name des Connectors kann dabei beliebig gewählt werden.

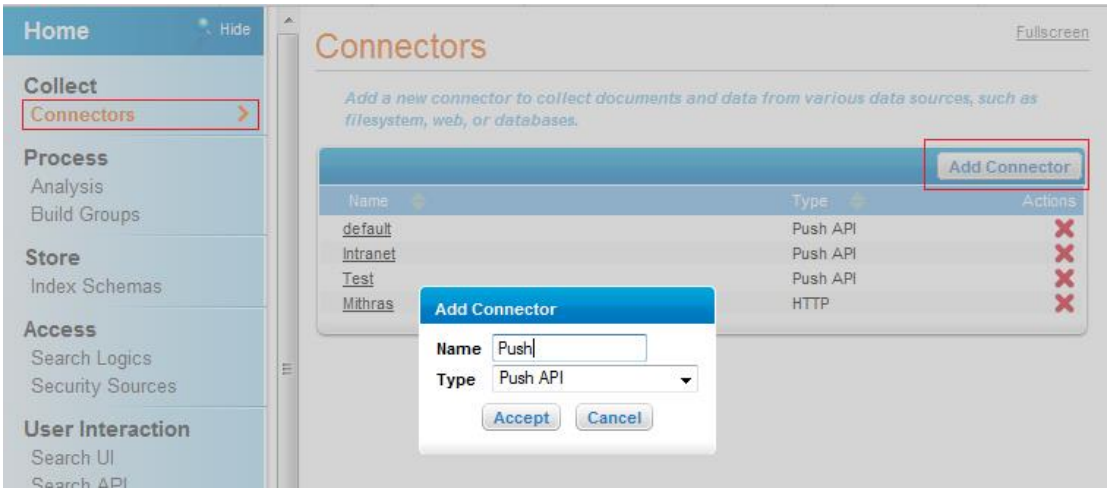

Abbildung 5-1: Einrichten eines Push-Connectors

Nun sollte dieser Connector durch ein Login und Password abgesichert werden. Dafür muss die API Console (erreichbar unter `http://<exalead-server-host>:<baseport>+1`) aufgerufen und darin der Bereich "Manage" angewählt werden:

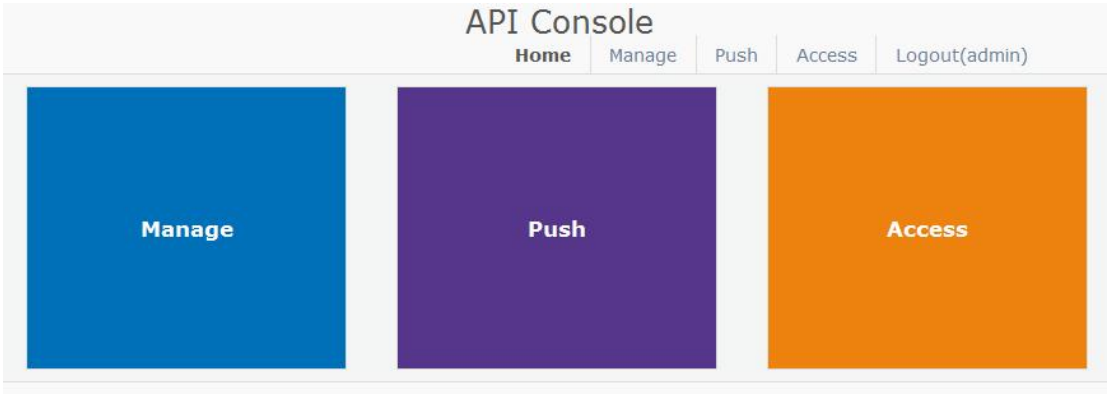

Abbildung 5-2: API Console

In diesem Bereich ist nun der Menüpunkt "connect" anzuwählen und darin der Untermenüpunkt "setConnectorList:

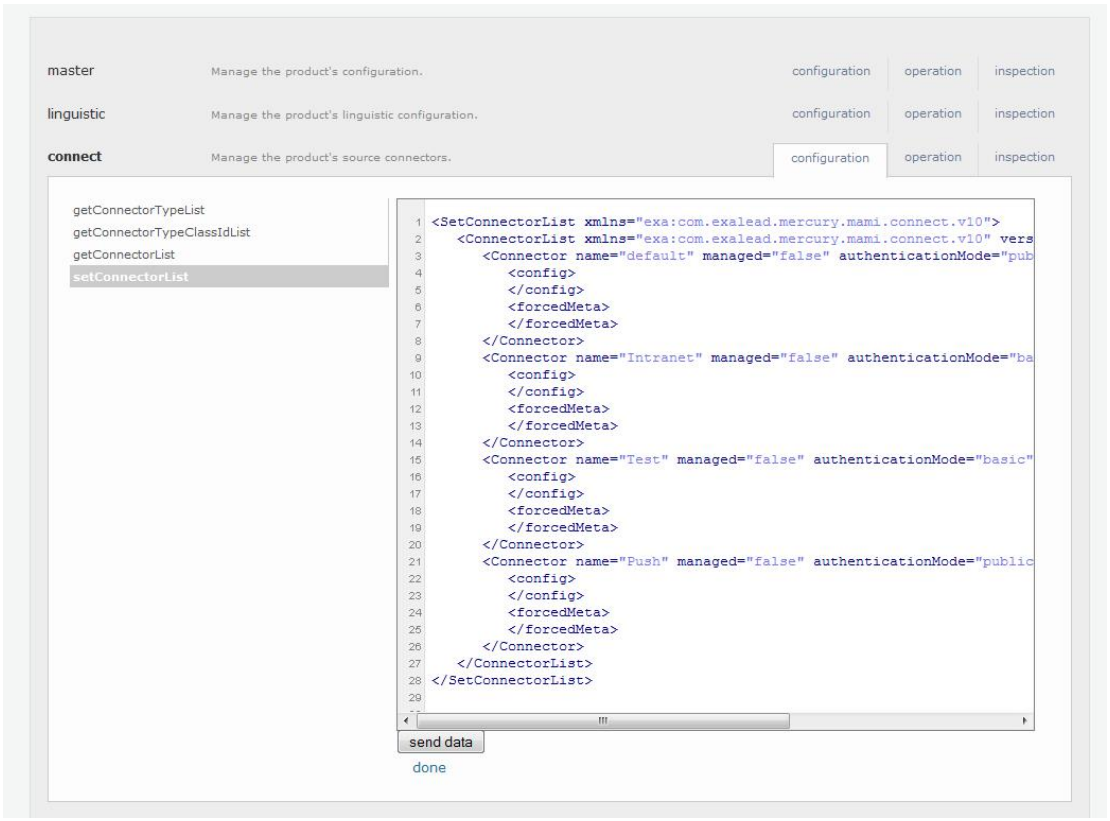

Abbildung 5-3: setConnectorList

Im dort angezeigten XML-Text befindet sich ein Abschnitt, der sich auf den zuvor neu angelegten Push-Connector bezieht (erkennbar am vergebenen Namen).

```
<Connector name=”Push” managed=”false”
authenticationMode="public">
    <config>
    </config>
    <forcedMeta>
```

```
    </forcedMeta>
</Connector>
```

In diesem Abschnitt muss nun der Wert des Attributs `authenticationMode` auf `basic` umgestellt und ein Login und Passwort vergeben werden:

```
<Connector name="Push" managed="false" authenticationMode="basic"
login="test" password="test">
    <config>
    </config>
    <forcedMeta>
    </forcedMeta>
</Connector>
```

Anschließend "send data" wählen, um die Änderungen zu speichern. Um die gespeicherten Änderungen nun noch zu übernehmen, bedarf es noch eines Aufrufs von "applyConfiguration". Dieser Aufruf befindet sich innerhalb des Menüpunkts "master". Auch hier muss erneut der Knopf "send data" betätigt werden.

Abbildung 5-4: applyConfiguration

5.2 FirstSpirit-Modul: Exalead Content-Update

5.2.1 Einsatz

Das FirstSpirit-Modul "Exalead Content-Update" dient dazu, in FirstSpirit generierte Inhalte durch die Nutzung der Push-API direkt nach ihrer Generierung in den Suchindex zu schreiben, so dass diese Inhalte nicht erst beim nächsten Crawler-Durchlauf indiziert werden, sondern ihre Indizierung zeitnah erfolgen kann.

Sind über Metadaten vergebene Rechte vorhanden, so werden diese dabei übernommen und zusammen mit dem jeweiligen Dokument in den Index geschrieben. Voraussetzung hierfür ist die Installation des PermissionServices in FirstSpirit (siehe Modul-Dokumentation zu *FirstSpirit Personalisation*, Kapitel 1.3.2).

Um webserverseitig personalisierte Suchabfragen absetzen zu können, die auf die Rechte aus den Metadaten des PermissionServices zurückgreifen, wird darüber hinaus FirstSpirit Personalisation vorausgesetzt. Dabei ist zu beachten, dass die

Gruppennamen des PermissionServices den Gruppennamen in FirstSpirit Personalisation entsprechen müssen.

5.2.2 Installation

Zur Installation des Exalead Content-Update Moduls ist nur das Einspielen der entsprechenden FSM-Datei auf dem FirstSpirit-Server nötig. Die Installation erfolgt gemäß dem *FirstSpirit Handbuch für Administratoren*.

Bei der Installation des Moduls wird automatisch eine Aktionsvorlage angelegt, die innerhalb eines Auftrages dazu genutzt werden kann, um die Aktion für den Aufruf der Push-API-Helferklasse zu erzeugen.

Die somit erzeugte Aktion enthält ein Skript, das die Übertragung der generierten Inhalte in den Suchindex anstößt. Dieses Skript muss innerhalb eines Auftrages nach einer Generierung aufgerufen werden.

Hinweis: Innerhalb der Generierungsaktion muss darauf geachtet werden, dass dort die Nutzung der ACL-Datenbank aktiviert wurde.

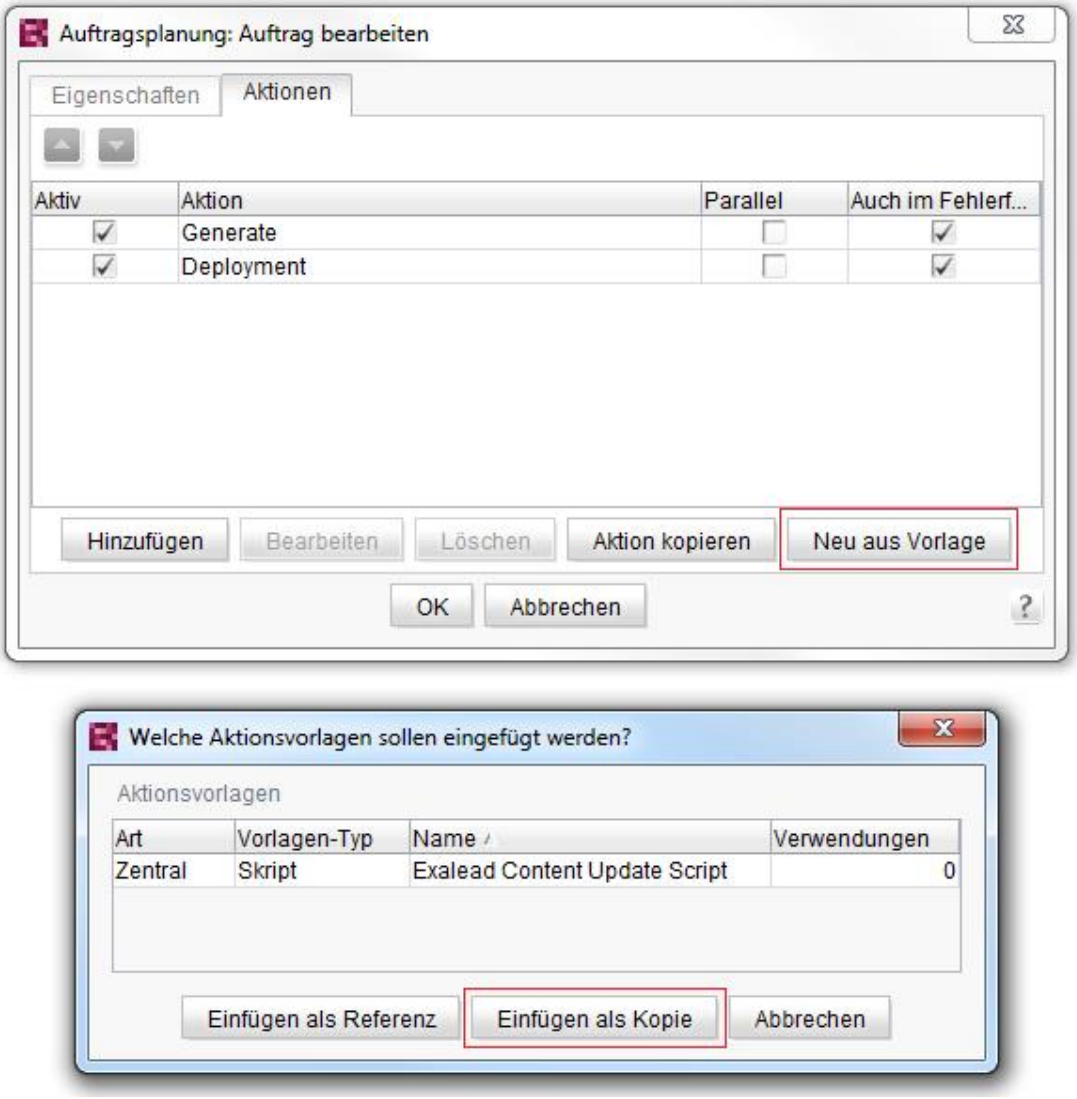

Abbildung 5-5: Einfügen der Aktionsvorlage

5.2.3 Konfiguration

Die Konfiguration des Skripts erfolgt über die Skript-Eigenschaften, wie sie in Abbildung 5-6 zu sehen sind.

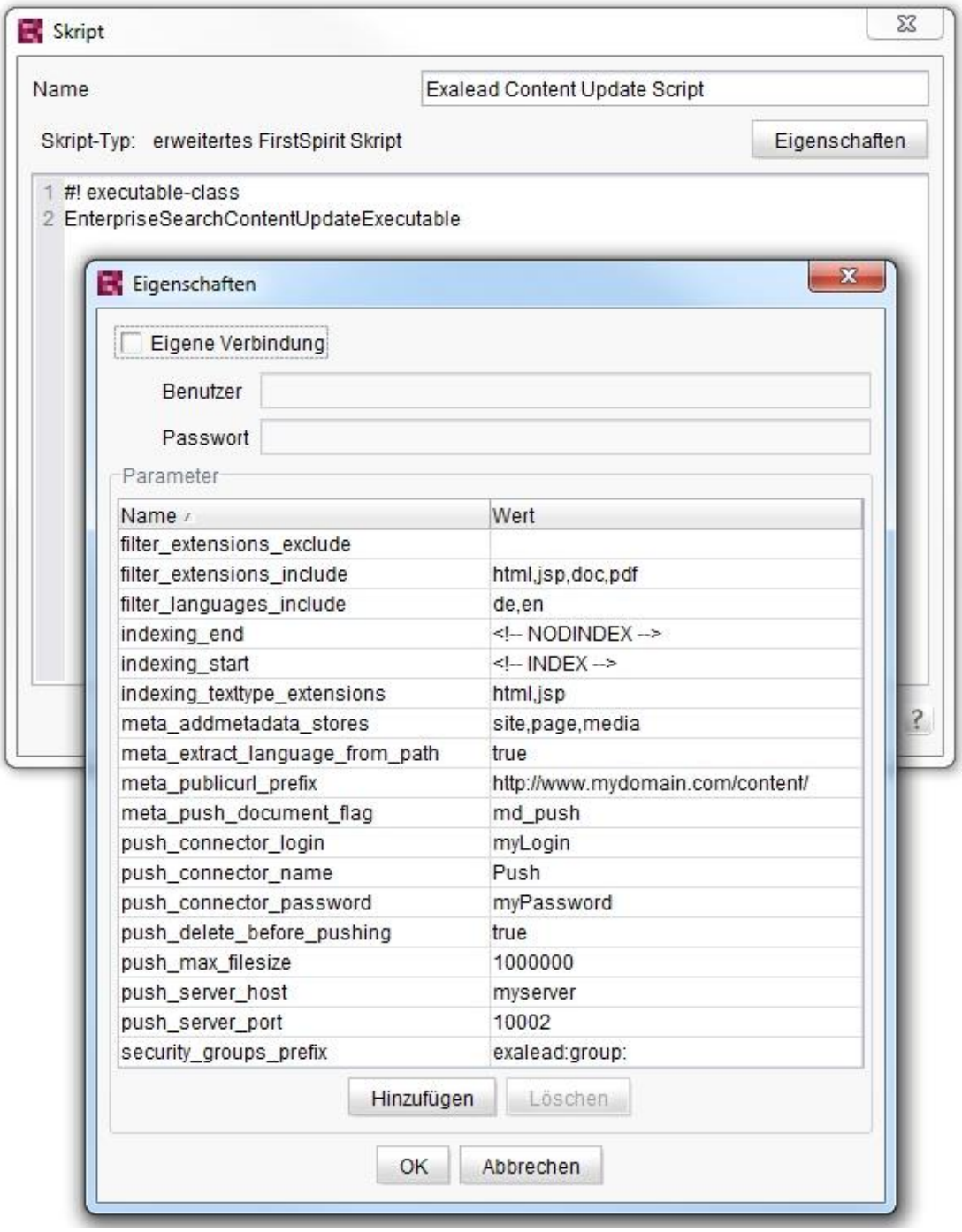

Abbildung 5-6: Skript-Eigenschaften

`filter_extensions_exclude`

Dieser Parameter nimmt eine kommaseparierte Liste von Dateitypen entgegen, die nicht in den Suchindex geschrieben werden sollen. Dieser Parameter wird nur ausgewertet, wenn `filter_extensions_include` leer ist.

Sind sowohl `filter_extensions_include` als auch

`filter_extensions_exclude` leer, so werden alle generierten Dateien (inkl. referenzierter Bilder, Stylesheets etc.) in den Suchindex geschrieben.

`filter_extensions_include`

Dieser Parameter nimmt eine kommaseparierte Liste von Dateitypen entgegen. Ist diese Liste nicht leer, so werden nur Dateien, die dieser Liste entsprechen, in den Suchindex geschrieben. Ist die Liste leer, so wird zusätzlich der Parameter `filter_extensions_exclude` ausgewertet.

`filter_languages_include`

Die Verwendung dieses Parameters ist nur in Verbindung mit der Verwendung von `meta_extract_language_from_path` sinnvoll. Durch die Angabe einer kommaseparierten Liste von Sprachkürzeln werden nur die Seiten und Dokumente in den Index geschrieben, deren Sprache in dieser Liste aufgeführt ist. Dabei werden Dokumente, denen keine Sprache zugeordnet ist, mit dem Kürzel ‚xx' ausgewählt.

Beispiele:

- Keine Angabe von Sprachen: Alle Dokumente werden in den Index geschrieben.
- `de`: Nur Dokumente der Sprache ‚de' werden in den Index geschrieben.
- `de,en`: Dokumente der Sprachen ‚de' und ‚en' werden in den Index geschrieben.
- `xx`: Nur sprachunabhängige Dokumente werden in den Index geschrieben.
- `de,xx`: Dokumente der Sprache ‚de' und sprachunabhängige Dokumente werden in den Index geschrieben.

`indexing_end`

Dieser Parameter setzt die Zeichenkette, die zur Kennzeichnung des Beginns eines Abschnittes dient, der nicht indiziert werden soll. Der Default-Wert lautet `<!-- NOINDEX -->`

`indexing_start`

Dieser Parameter setzt die Zeichenkette, die zur Kennzeichnung des Endes eines Abschnittes dient, der nicht indiziert werden soll. Der Default-Wert lautet `<!-- INDEX -->`

`indexing_texttype_extensions`

Kommaseparierte Liste von Dateierweiterungen von im Textformat gespeicherten Dokumenten. Nur für solche Dokumente können Bereiche definiert werden, die von der Indizierung ausgeschlossen werden. Der Default-Wert lautet: `html,jsp`

`meta_addmetadata_stores`

Über diesen Parameter kann gesteuert werden, ob neben dem Inhalt von Dokumenten auch zusätzliche Metadaten in den Index geschrieben werden sollen. Dabei gibt der Wert des Parameters – eine kommaseparierte Liste der Werte `site` (Struktur), `media` (Medien) und `page` (Inhalte) – die Verwaltung innerhalb von FirstSpirit an, aus der die Metadaten zu dem Dokument bezogen werden sollen.

Beispiele:

- `site` = Es werden die Metadaten aus der Struktur-Verwaltung herangezogen.
- `media,page` = Es werden die Metadaten aus der Medien- und der Inhalte-Verwaltung herangezogen und gegebenenfalls kombiniert.

Zum Auswerten der Metadaten bedarf es noch entsprechender Einstellungen auf Seiten des Exalead-Servers:

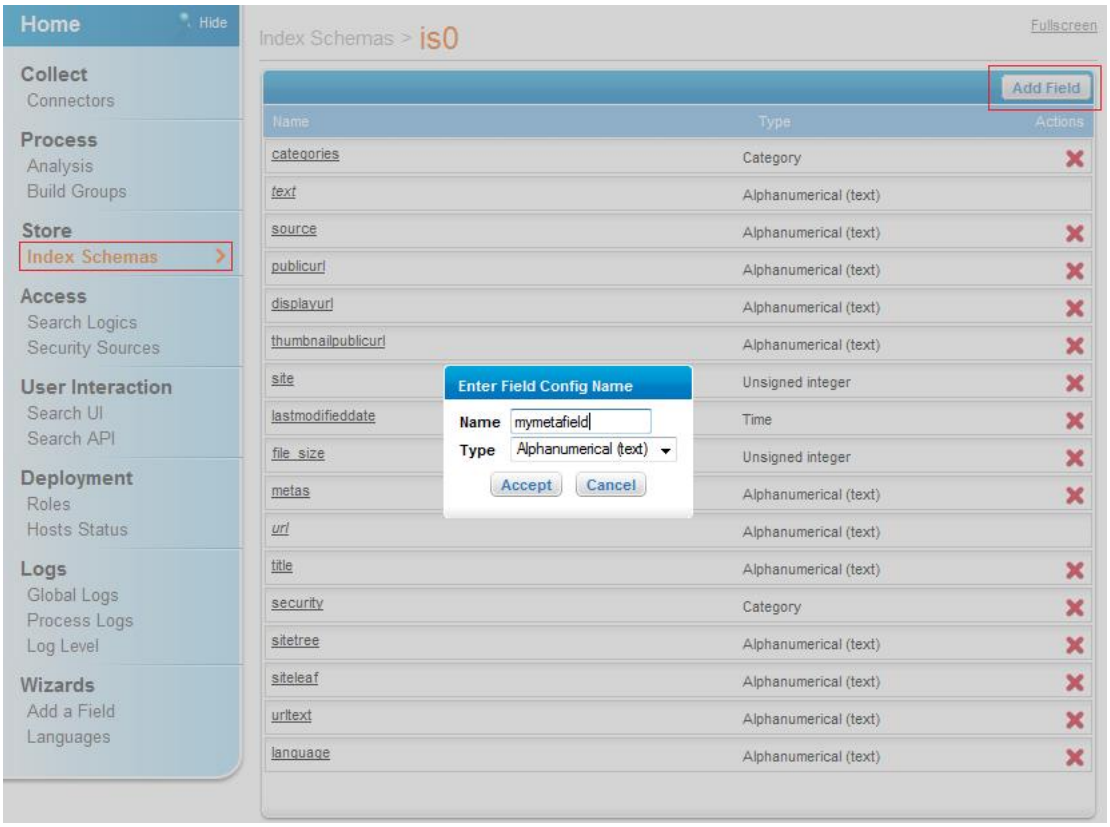

Abbildung 5-7: Anlegen eines neuen Feldes im Index

Zunächst muss für jedes Metadatum, welches ausgewertet werden soll, ein neues Feld im Index erstellt werden (siehe Abbildung 5-7). Der Name des Feldes kann dabei beliebig gewählt werden.

Im nächsten Schritt muss nun ein neues Mapping erstellt werden, welches das FirstSpirit-Metadatum einliest und es auf das soeben angelegte Indexfeld abbildet (siehe Abbildung 5-8: Anlegen eines Mappings).

Die Bezeichnung des Mappings muss exakt dem Namen des Metadatums aus der Metadatenvorlage in FirstSpirit entsprechen.

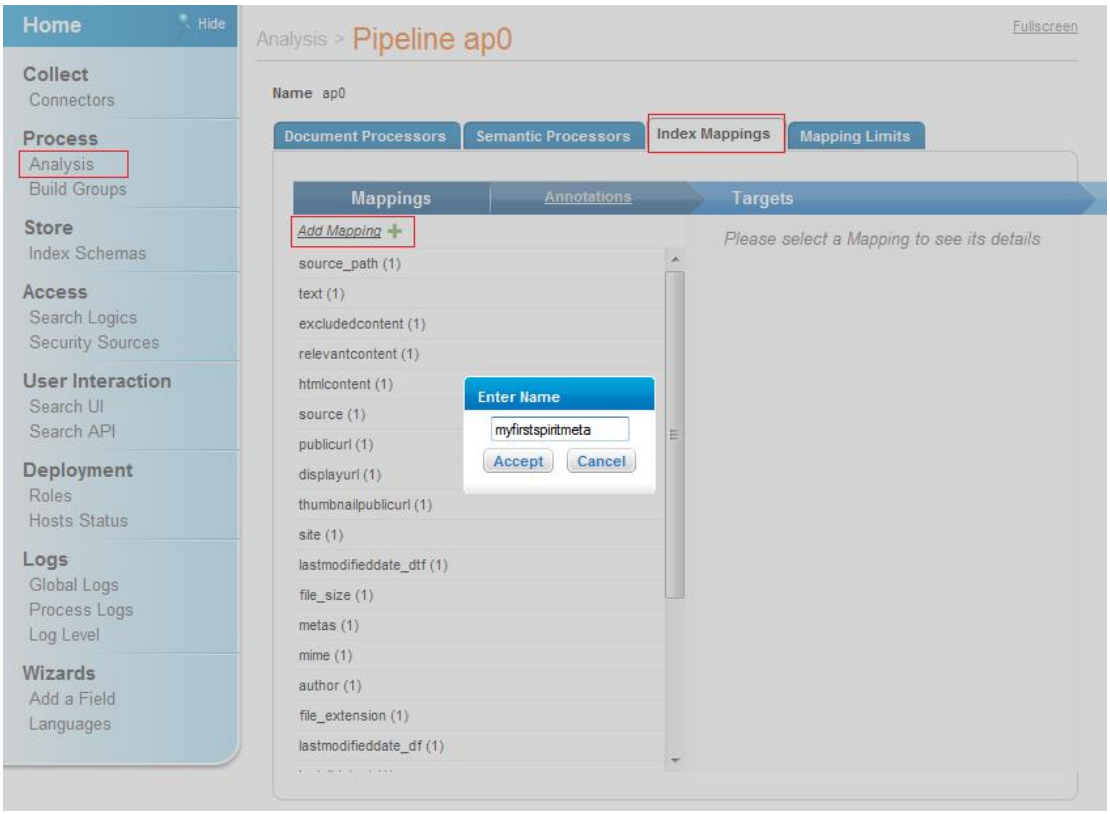

Abbildung 5-8: Anlegen eines Mappings

Nach Anlegen des Mappings muss dem Mapping noch ein Ziel zugewiesen werden (siehe Abbildung 5-9: Hinzufügen eines Mappingsziels). Als Typ des Mappings ist "Index Field" zu wählen und als Name das zuvor angelegte Indexfeld.

Um später bei einer Suchanfrage den Inhalt der neu angelegten Indexfelder ausgeben zu können (siehe dazu auch Kapitel 3.12.8), müssen diese im Bereich "Search Logics" zur Anzeige ausgewählt werden (siehe Abbildung 5-10 Metadatum im Suchergebnis anzeigen).

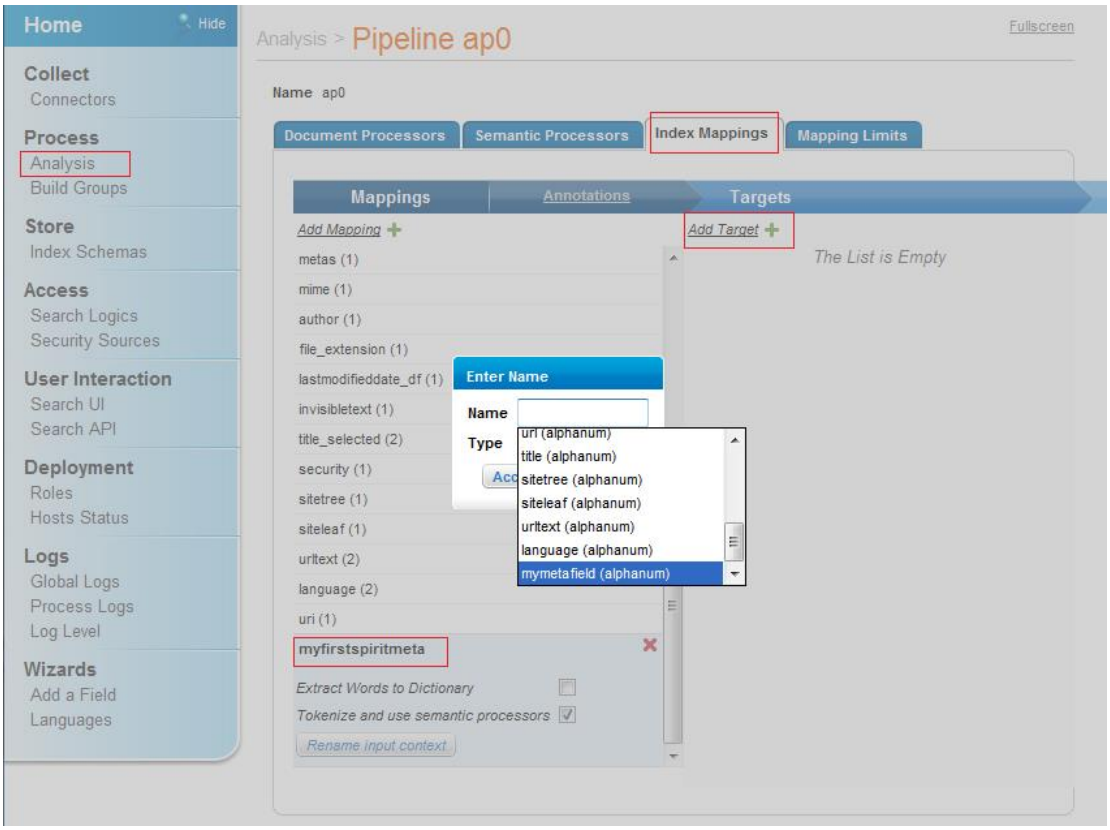

Abbildung 5-9: Hinzufügen eines Mappingsziels

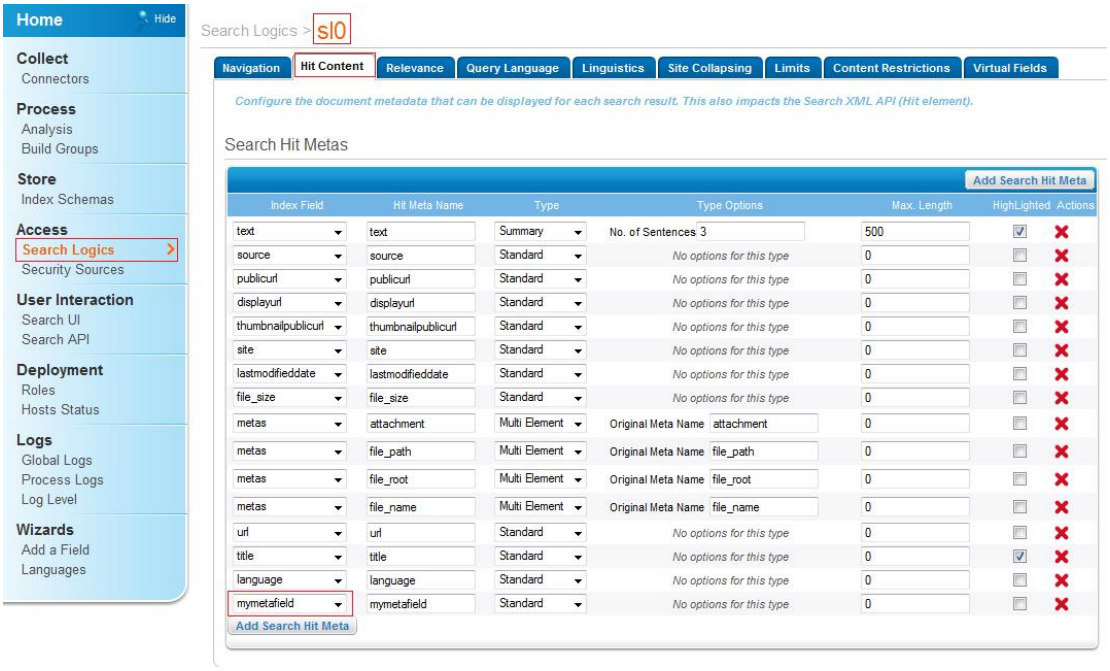

Abbildung 5-10 Metadatum im Suchergebnis anzeigen

Unterstützt werden Metadaten aus folgenden FirstSpirit-Eingabekomponenten:

- `CMS_INPUT_TEXT`
- `CMS_INPUT_DATE`
- `CMS_INPUT_COMBOBOX`
- `CMS_INPUT_CHECKBOX`
- `CMS_INPUT_LIST`
- `CMS_INPUT_TOGGLE`

Für `CMS_INPUT_TEXT` und `CMS_INPUT_DATE` wird der Inhalt der Eingabekomponente als Wert an Exalead übergeben. Für `CMS_INPUT_COMBOBOX`, `CMS_INPUT_LIST` und `CMS_INPUT_CHECKBOX` wird das in der Metadaten-Vorlage definierte Label der entsprechenden Auswahl als Wert an Exalead übergeben. Bei `CMS_INPUT_TOGGLE` wird bei Selektion der Komponente `true` übergeben.

`meta_extract_language_from_path`

Über diesen Parameter kann gesteuert werden, ob die Dokumentensprache anhand des Pfades der generierten Datei ermittelt werden soll. Wird dieser Parameter nicht gesetzt (oder der Wert `false` übergeben), entscheidet der Exalead-Server anhand eines internen Algorithmus, welcher Sprache ein Dokument zugeordnet wird. Durch Setzten dieses Parameters auf den Wert `true` wird die Sprache anhand des im Dateipfad vorgefundenen Sprachkürzels ermittelt:

/de/content/start.html	→	Sprache:deutsch
/en/content/start.html	→	Sprache:englisch
/de_1/content.pdf	→	Sprache:deutsch
/media/en/docs/flyer.pdf	→	Sprache:englisch
/media/docs/flyer.pdf	→	keine zugeordnete Sprache

Bei Verwendung des automatischen Extrahierens der Dokumentensprache anhand des Dateipfades ist allerdings zu beachten, dass der Name jedes direkt unterhalb des Medienverzeichnisses liegenden Verzeichnisses, das aus zwei Buchstaben besteht, als Sprache der darin enthaltenen Dokumente interpretiert wird.

Beispiel:

/media/sg/styleguide.pdf	→	Sprache:sg

Besser:

/media/guides/styleguide.pdf	→	keine zugeordnete Sprache

Bei Dokumenten ohne zugeordnete Sprache übernimmt Exalead die Analyse und

die Zuordnung des Dokuments zu einer Sprache.

> *Das Extrahieren der Sprache aus dem Dateipfad ist nur möglich, wenn zur Pfaderzeugung ‚Default URLs' im Generierungsauftrag ausgewählt wird.*

`meta_publicurl_prefix`

Über diesen Parameter wird der Prefix für die Pfade der deployten Dateien angegeben. Dabei wird im absoluten Pfad der generierten Dateien das Generierungsverzeichnis durch diesen Prefix ersetzt, wenn die Dateien in den Suchindex geschrieben werden. So wird im obigen Beispiel aus dem Pfad

`<Generierungsverzeichnis>/de/unternehmen/kontakt.jsp`

der Pfad

`http://www.mydomain.de/content/de/unternehmen/kontakt.jsp`

`meta_push_document_flag`

Über diesen optionalen Parameter kann man den Namen einer Metadaten-Eingabekomponente in FirstSpirit angeben, über die einzelne Dokumente (Seiten/Medien) von der Indizierung ausgeschlossen werden können.

Die Eingabekomponente muss vom Typ `CMS_INPUT_TOGGLE` sein. Bei Selektion der Checkbox wird das entsprechende Dokument nicht indiziert.

Es wird nur der Wert der Eingabekomponente aus der Struktur- bzw. der Medienverwaltung ausgewertet. Selektierte Checkboxen dieser Eingabekomponente in der Inhalte-Verwaltung werden beim Indizieren nicht berücksichtigt.

Beispielkonfiguration in der Metadaten-Vorlage:

```
<CMS_INPUT_TOGGLE name="md_push" singleLine="no">
    <LANGINFOS>
        <LANGINFO lang="*" label="Disable push"/>
    </LANGINFOS>
</CMS_INPUT_TOGGLE>
```

`push_connector_login`

Login für den Zugriff auf den Push Connector (Einstellung des Exalead-Servers)

`push_connector_name`

Name des Push Connectors (Einstellung des Exalead-Servers)

`push_connector_password`

Passwort für den Zugriff auf den Push Connector (Einstellung des Exalead-Servers)

`push_delete_before_pushing`

Über diesen Parameter kann gesteuert werden, ob alle bisher in den Suchindex geschriebenen Dokumente vor einem erneuten Schreibvorgang entfernt werden. Dies ist für eine Vollgenerierung zu empfehlen; bei einer Teilgenerierung sollte man dies jedoch deaktivieren.

`push_max_filesize`

Über diesen optionalen Parameter kann man eine Maximalgröße (in Bytes) für Dokumente einstellen, die an den Exalead-Server übertragen werden sollen. Wird der Parameter nicht angegeben oder kein Wert für ihn definiert, wird auf den Defaultwert (10485760 Bytes = 10 MB) zurückgegriffen.

`push_server_host`

Name des Exalead-Servers im Netz

`push_server_port`

Port, unter dem der Push Connector erreichbar ist (Einstellung des Exalead-Servers, in der Regel <baseport> + 2, z.B. 10002)

`security_groups_prefix`

Anhand dieses Parameters wird dem Skript mitgeteilt, welchen Prefix es beim Setzen der Gruppenrechte verwenden soll. Hier sollte derselbe Wert wie in der Konfiguration des SearchServlets verwendet werden (siehe Kapitel 3.7.1).

5.2.4 Entfernen einzelner Seiten/Dokumente aus dem Index

Um einzelne, zuvor über das Exalead Content-Update Modul in den Index geschriebene Seiten bzw. Dokumente wieder aus dem Index zu entfernen, stellt das Modul einen Service zur Verfügung, der aus beliebigen Skripten heraus (z. B. innerhalb eines Lösch-Workflows) angesprochen werden kann.

Um den Service verwenden zu können, wird zunächst eine Instanz des Services

benötigt:

```
exaleadService =
context.getConnection().getService("EnterpriseSearchService");
```

Im nächsten Schritt muss der Service mit den Verbindungsdaten des Exalead-Servers initialisiert werden:

```
exaleadService.initPAPI(String exalead_hostname, String port,
String push_connector, String push_login, String push_password);
```

Dabei müssen die Parameter mit folgenden Werten belegt werden:

- `exalead_hostname =` Name des Exalead-Servers
- `port` = Push-API Port des Exalead-Servers (Baseport + 2, z. B.: 10002)
- `push_connector =` Name der verwendeten Push-Connectors
- `push_login =` Login für den Push-Connector (Leerstring, falls der Push-Connector nicht geschützt ist)
- `push_password =` Passwort für den Push-Connector (Leerstring, falls der Push-Connector nicht geschützt ist)

Beispiel:

```
exaleadService.initPAPI(myServer, 10002, Push, myLogin,
myPassword);
```

Nach der Initialisierung des Services kann dieser nun zum Entfernen von Seiten/Dokumenten verwendet werden:

```
exaleadService.removeFromIndex(String document_path, String
projectId);
```

Dabei muss als `document_path` der Generierungspfad des Dokuments und als `projectId` die Projekt-ID des Projekts angegeben werden.

Beispiel:

```
exaleadService.removeFromIndex("de/aboutus/company/company.jsp",
2574);
```

Dieser Aufruf muss für jede Projektsprache und jeden Ausgabekanal wiederholt werden:

```
exaleadService.removeFromIndex("de/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("en/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("de_1/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("en_1/aboutus/company/company.jsp",
2574);
```

Um im Anschluss an das Entfernen der gewünschten Dokumente diese Änderungen in den Live-Index des Exalead-Servers zu übernehmen, ist ein manueller Commit nötig:

```
exaleadService.commitChanges();
```

Ohne manuellen Commit werden die Änderungen am Index beim nächsten automatischen Commit übernommen (das Intervall des automatischen Commits ist auf Seiten des Exalead-Servers konfigurierbar).

5.3 Die Java Push-API-Helferklasse

Die Push-API kann zur Abdeckung spezieller Projektanforderungen auch manuell aufgerufen werden. Eine passende Java-Helferklasse zur Durchführung der notwendigen Operationen befindet sich in der Datei `enterprisesearch-fsm-x.jar`, die Teil des FirstSpirit-Moduls „Exalead Content-Update“ ist.

5.3.1 Initialisierung

Um per Java die Push-API zu verwenden, muss nun zunächst die Helferklasse initialisiert werden:

```
de.espirit.ps.exalead.pushapi.PushAPIHelper pHelper = new
de.espirit.ps.exalead.pushapi.PushAPIHelper("myserver", "10002",
"Push", "test", "test");
```

Der erste Parameter des Konstruktors steht dabei für den Servernamen und der zweite Parameter für den Port (`baseport + 02` ist der Standardport des Push-Services). Der dritte Parameter gibt den Namen des angelegten Push-Connectors an. Der vierte und fünfte Parameter geben die in Kapitel 5.1 vergebenen Login-Daten an.

5.3.2 Öffentliche Methoden

5.3.2.1 sendDocument(PushAPIDocumentValues docValues)

Diese Methode überträgt ein beliebiges, einzelnes Dokument (lokales File-System oder HTTP-Ressource) in den Index. Dazu wird zunächst ein PushAPIDocumentValues-Objekt erzeugt, welches mit entsprechenden Werten befüllt werden muss.

Beispiel: Pushen einer Datei aus dem lokalen File-System

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();


// Notwendige Angabe: URI = Ort der Datei
pav.setUri("E:/PushTest/MyDocument.pdf");
// Notwendige Angabe: PublicURL = Öffentlich zugänglicher Pfad
pav.setPublicURL("http://www.myserver.de/media/pdf/MyDocument.pdf"
);


// Optionale Angaben:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My document");


// Pushen des Dokuments
pHelper.sendDocument(pav);
```

Beispiel: Pushen einer HTTP-Ressource

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();


// Notwendige Angabe: URI = Ort des Seite
pav.setUri("http://www.myserver.de/content/de/index.html");
// Notwendige Angabe: PublicURL = Öffentlich zugänglicher Pfad
pav.setPublicURL("http://www.myserver.de/content/de/index.html");


// Optionale Angaben:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My document");


// Pushen des Dokuments
pHelper.sendDocument(pav);
```

Zusätzlich ist es auch möglich, den Inhalt eines Dokuments direkt anzugeben. Dazu kann man dem PushAPIDocumentValues-Objekt zusätzlich ein Byte-Array übergeben. Beim Pushen wird der Inhalt des Dokuments dann nicht mehr von dem als URI angegebenen Ort gelesen, sondern der Inhalt des Byte-Arrays übernommen.

Beispiel: Pushen eines Byte-Arrays

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();


// Notwendige Angabe: Der Inhalt als Byte-Array
pav.setContent(myByteArray);
// Notwendige Angabe: URI (wird intern trotz Angabe des Byte-
Arrays benötigt
pav.setUri("http://www.myserver.de/myByteArray.html ");
// Notwendige Angabe: PublicURL
pav.setPublicURL("http://www.myserver.de/myByteArray.html");
// Notwendige Angabe: Date
pav.setDate(new Date());
// Notwendige Angabe: Filename = Name des Dokuments
pav.setFilename("myByteArray.html");
// Notwendige Angabe: Stamp = Eindeutige Angabe zur
Identifizierung von Änderungen am Dokument, zum Beispiel
Zeichenkette aus Erstellungsdatum und Länge des Dokuments
pav.setStamp((new Date()).toString + myByteArray.length);


// Optionale Angaben:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My Byte Array Document");


// Pushen des Dokuments
pHelper.sendDocument(pav);
```

Wird kein Sprachkürzel über die Methode setLanguage() gesetzt, so versucht Exalead, anhand einer internen Inhaltsüberprüfung die Sprache des Dokuments herauszufinden. Möchte man dies für Dokumente, die keiner Sprache zugeordnet sind, verhindern, so kann man die Sprache über setLanguage("xx") auf unbekannt setzen. Exalead wird den Inhalt des Dokuments dann nicht mehr auf eine feststellbare Sprache hin überprüfen.

5.3.2.2 sendDirectory(PushAPIDocumentValues docValues, boolean recursive, String excludeExtensions)

Diese Methode überträgt den kompletten Inhalt eines Verzeichnisses in den Index.

Über den Parameter `excludeExtensions` kann eine kommaseparierte Liste an Datei-Erweiterungen angegeben werden, die von der Übertragung in den Index ausgeschlossen werden sollen. Es kann darüber hinaus angegeben werden, ob auch alle Unterverzeichnisse übertragen werden sollen.

Beispiel: Pushen eines Verzeichnisses

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();


// Notwendige Angabe: URI = Das zu pushende Verzeichnis
pav.setUri("E:/PushTest/dir");
// Notwendige Angabe: PublicURL = Öffentlich zugänglicher Pfad
pav.setPublicURL("http://www.myserver.de/content/" +
PushAPIHelper.PUBLIC_URL_PLACEHOLDER);

// Rekursives Pushen des Verzeichnisses
pHelper.sendDirectory(pav, true, "css,js");
```

Da man beim Pushen eines kompletten Verzeichnisses nicht für jedes Dokument einzeln die PublicURL angeben kann, kann man in diesem Fall auch mit einem Platzhalter arbeiten, der beim Pushen der einzelnen Dokumente dann durch den Dateinamen des jeweiligen Dokuments ersetzt wird. Dabei gilt als Dateiname immer der komplette Pfad, der nach der angegebenen URI beginnt.

Beispiel:

Es liegt folgende Dateistruktur vor:

E:\PushTest\dir\a.html
E:\PushTest\dir\b.html
E:\PushTest\dir\subdir1\c.html
E:\PushTest\dir\subdir1\d.html
E:\PushTest\dir\subdir1\subdir2\e.html
E:\PushTest\dir\subdir1\subdir2\f.html

Dann lauten die PublicURLs dieser Dokumente wie folgt:

http://www.myserver.de/content/a.html
http://www.myserver.de/content/b.html
http://www.myserver.de/content/subdir1/c.html
http://www.myserver.de/content/subdir1/d.html
http://www.myserver.de/content/subdir1/subdir2/e.html
http://www.myserver.de/content/subdir1/subdir2/f.html

5.3.2.3 resetAllDocuments()

Mit dieser Methode werden alle gepushten Dokumente aus dem Index entfernt.

5.3.2.4 long getTransferRate()

Mit dieser Methode kann man die Übertragungsrate aller seit Initialisierung der Helferklasse gepushten Dokumente in kb/s abfragen.

5.3.2.5 Date getGlobalStartDate()

Zeitpunkt der Initialisierung der Helferklasse

5.3.2.6 long getGlobalByteCounter()

Seit Initialisierung der Helferklasse übertragene Bytes

5.3.2.7 long getErrorCounter()

Anzahl der aufgetretenen Fehler seit Initialisierung der Helferklasse

5.3.2.8 long getDocCounter()

Anzahl der seit Initialisierung der Helferklasse gepushten Dokumente

5.3.2.9 resetCounters()

Setzt alle Counter zurück und GlobalStartDate auf die aktuelle Zeit.

5.3.2.10 setMaxFileSize(int maxFileSize)

Maximale Größe der zu pushenden Dokumente in byte. Dokumente, die diesen Wert überschreiten, werden nicht in den Index geschrieben.

5.3.2.11 int getMaxFileSize()

Abfrage der eingestellten Maximalgröße

5.3.2.12 setCheckpointAndTriggerIndexing()

Ein Aufruf dieser Methode veranlasst Exalead, die übertragenen Dokumente sofort zu indizieren. Andernfalls wird eine Indizierung gemäß der Build Groups Einstellungen vorgenommen.

5.3.2.13 setUserPrefix(String userPrefix)

Anhand dieser Methode wird der Push-API-Helferklasse mitgeteilt, welchen Prefix sie beim Setzen der Benutzerrechte verwenden soll. Hier sollte derselbe Wert wie in der Konfiguration des SearchServlets verwendet werden (siehe Kapitel 3.7.1).

5.3.2.14 setGroupsPrefix(String groupsPrefix)

Anhand dieser Methode wird der Push-API-Helferklasse mitgeteilt, welchen Prefix sie beim Setzen der Gruppenrechte verwenden soll. Hier sollte derselbe Wert wie in der Konfiguration des SearchServlets verwendet werden (siehe Kapitel 3.7.1).

5.3.2.15 setNoIndexStart(String noIndexStart)

Diese Methode überschreibt den Default-Wert für die Kennzeichnung des Beginns eines nicht zu indizierenden Bereichs innerhalb eines HTML-Dokuments. Der Default-Wert lautet:

```
<!-- NOINDEX -->
```

5.3.2.16 setIndexStart(String indexStart)

Diese Methode überschreibt den Default-Wert für die Kennzeichnung des Endes eines nicht zu indizierenden Bereichs innerhalb eines HTML-Dokuments (= Beginn des wieder zu indizierenden Bereichs). Der Default-Wert lautet:

```
<!-- INDEX -->
```

5.3.2.17 setTextTypeExtensions(String extensions)

Mit dieser Methode gibt man eine kommaseparierte Liste von Dateierweiterungen von im Textformat gespeicherten Dokumenten an. Nur für solche Dokumente können Bereiche definiert werden, die von der Indizierung ausgeschlossen werden. Der Default-Wert lautet:

```
html,jsp
```

5.4 Vergabe von Rechten über die Push-API

Möchte man beim Pushen über die Push-API den Dokumenten Rechte mitgeben, so geschieht dies durch Setzen der Rechte im entsprechenden PushAPIDocumentValues-Objekt:

```
List<String> users = new ArrayList<String>();
users.add("user1");
users.add("user2");
pav.setUsers(users);

List<String> groups = new ArrayList<String>();
groups.add("group3");
groups.add("group4");
pav.setGroups(groups);
```

Hierbei muss im Gegensatz zum Setzen der Rechte über MetaTags (vergleiche Kapitel 4.2) kein Prefix angegeben werden, da dieser von der Push-API-Helferklasse automatisch ergänzt wird (siehe Kapitel 5.3.2.13 und 5.3.2.14).

5.5 Bereiche von der Indizierung ausschließen

Innerhalb von Dokumenten, die über die Push-API in den Index geschrieben werden, gibt es die Möglichkeit, durch spezielle NoIndex-Tags bestimmte Bereiche von der Indizierung auszuschließen. Dies ist z. B. sinnvoll, wenn man innerhalb einer HTML-Seite die Navigation nicht indizieren möchte.

Der Anfang eines Bereichs, der nicht indiziert werden soll, wird durch folgendes Tag gekennzeichnet:

```
<!-- NOINDEX -->
```

Das Ende eines solchen Bereichs kennzeichnet folgendes Tag:

```
<!-- INDEX -->
```

Bemerkung: Eine Schachtelung dieser Tags ist **nicht** möglich, was bei der Erstellung von FirstSpirit-Vorlagen passend berücksichtigt werden muss.

Bemerkung: Diese Tags entsprechen der Default-Einstellung und können über die entsprechenden Methoden der Push-API-Helferklasse überschrieben werden (siehe Kapitel 5.3.2.15 und Kapitel 5.3.2.16).

Bemerkung: Die Erkennung solcher Bereiche erfolgt nur bei Dokumenten, die im Text- und nicht im Binärformat gespeichert werden. Daher ist es notwendig, der Push-API-Helferklasse Dateitypen, die im Textformat vorliegen, über die entsprechende Methode bekannt zu machen (siehe Kapitel 5.3.2.17).

6 Autovervollständigung (SuggestService)

Exalead Cloudview bietet mit dem SuggestService eine Schnittstelle an, die es ermöglicht, dem Benutzer bei der Eingabe eines Suchbegriffs in Echtzeit Vorschläge für eine Autovervollständigung des Suchbegriffs zu machen. Beispiele für die Oberfläche sind in Abbildung 6-1 und Abbildung 6-2 zu sehen.

Abbildung 6-1: Suchvorschläge in der Exalead-Suchoberfläche

Abbildung 6-2: Suchvorschläge im Exalead Live Modul

Das Einrichten eines SuggestServices gliedert sich dabei in folgende vier Schritte:

1. Erstellen einer Wörterbuchdatei
2. Kompilieren der Wörterbuchdatei
3. Einrichten des SuggestServices auf dem Exalead-Server
4. Anpassen der Suchvorlage

Der letzte Schritt ist nur einmal pro SuggestService durchzuführen (es ist möglich, mehrere SuggestServices einzurichten, von denen aber immer nur einer ausgewählt sein kann). Die ersten beiden Schritte hingegen sind bei jeder Aktualisierung des SuggestServices durchzuführen.

6.1 Erstellen einer Wörterbuchdatei

Der SuggestService von Exalead basiert auf einem Vorschlagswörterbuch, das in Form einer speziellen XML-Datei vorliegen muss:

```
<SuggestDictionary xmlns="exa:com.exalead.mot.suggest.v10"
maxEntries="10">
    <SuggestDictEntry entry="Safari" score="1" />
    <SuggestDictEntry entry="SAMBA" score="1" />
    <SuggestDictEntry entry="Sandbox-Effekt" score="1" />
    <SuggestDictEntry entry="SAX" score="1" />
</SuggestDictionary>
```

Diese XML-Datei kann von Hand, von FirstSpirit über eine dafür vorgesehene Vorlage oder von Exalead selbst erstellt werden. Im Falle einer von Exalead selbst erstellten XML-Datei kann diese nach Erzeugung auch immer noch von Hand bearbeitet werden, bevor sie von Exalead als Wörterbuch für den SuggestService verwendet wird (weiterführende Informationen zum Aufbau der XML-Datei sind in der Exalead-Dokumentation zum SuggestService zu finden).

Die Erstellung der XML-Datei durch Exalead wird über folgenden Kommandozeilenbefehl angestossen:

```
cvconsole create-suggest-file-from-index fieldName suggest.xml
```

Dabei steht `fieldName` für ein Indexfeld, welches als Quelle für die vorzuschlagenden Suchbegriffe dienen soll. Dieses Indexfeld sollte optimalerweise ein Feld sein, welches pro Dokument bei der Indizierung mit nur einem einzelnen

Wort befüllt wird.

Der zweite Parameter gibt den Namen der zu erzeugenden XML-Datei an und kann frei gewählt werden.

Beispiel:

```
D:\Programme\Exalead Cloudview\data\bin>cvconsole create-suggest-
file-from-index mymetafield mymetasuggest.xml
```

6.2 Kompilieren der Wörterbuchdatei

Nachdem im ersten Schritt eine Wörterbuchdatei im XML-Format erstellt wurde, muss diese nun im zweiten Schritt in eine Binärdatei überführt und als Quelle für den SuggestService installiert werden. Dies geschieht mit folgendem Kommandozeilenbefehl:

```
cvconsole compile-suggest /path/to/source.xml qshare:///destDir
```

Der Befehl erwartet als ersten Parameter den Pfad zur XML-Datei und als zweiten Parameter das Zielverzeichnis, in welches die Binärdatei geschrieben werden soll. Die Angabe des Zielverzeichnisses erfolgt dabei immer in der Form `qshare:/// + <beliebiger Verzeichnisname>`

Beispiel:

```
D:\Programme\Exalead Cloudview\data\bin>cvconsole compile-suggest
mymetasuggest.xml qshare:///mysuggest
```

Dieser Aufruf hat ein neues Verzeichnis `mysuggest` unterhalb von `D:\Programme\Exalead Cloudview\data\build\qshare\dic0` angelegt, welches das kompilierte Wörterbuch enthält.

Wurde das Verzeichnis bereits durch einen früheren Aufruf dieses Kommandos erstellt und der entsprechende SuggestService in Exalead bereits eingerichtet (Schritt 3), so kann man dem Kommando noch einen dritten Parameter übergeben, der den Namen des SuggestServices angibt, der nach dem Kompilieren der XML-Datei neu gestartet werden soll, damit die Änderungen am Wörterbuch für den Benutzer sofort sichtbar sind.

Beispiel:

```
D:\Programme\Exalead Cloudview\data\bin>cvconsole compile-suggest
mymetasuggest.xml qshare:///mysuggest mysuggestservice
```

6.3 Einrichten des SuggestServices auf dem Exalead-Server

Nach Erstellen, Kompilieren und Installieren des Wörterbuchs muss nun ein entsprechender SuggestServices auf dem Exalead-Server eingerichtet werden. Die dafür notwendigen Operationen werden online im Manage-Bereich der API Console von Exalead durchgeführt (erreichbar unter `http://<exalead-server-host>:<baseport>+1`):

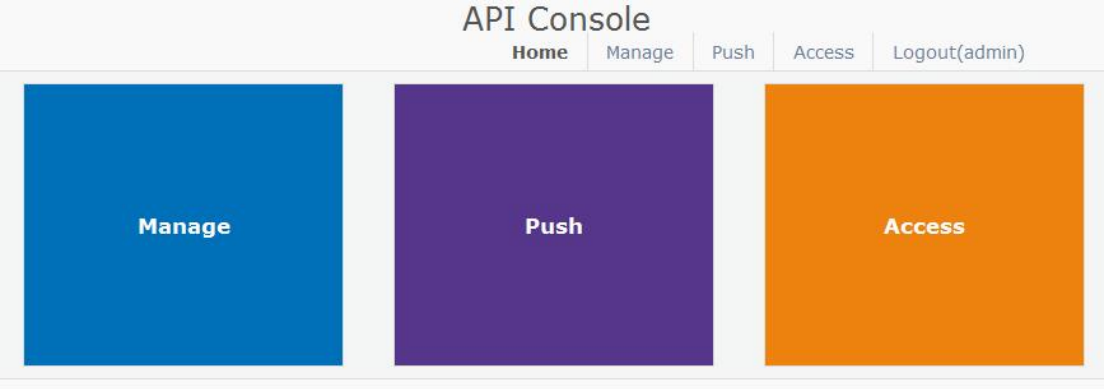

Abbildung 6-3: API Console

Zunächst muss ein neuer SuggestService angelegt werden. Dazu wird im Bereich linguistic > configuration der Befehl setSuggestServiceList aufgerufen:

Abbildung 6-4: Hinzufügen eines SuggestServices

Innerhalb des SuggestServiceList-Tags wird nun ein Eintrag für den neuen SuggestService angelegt, indem folgendes Tag hinzugefügt wird:

```
<SuggestService serviceName="mysuggestservice"
resourceDir="qshare:///mysuggest">
</SuggestService>
```

Der Name des Services kann beliebig gewählt werden. Das ResourceDir muss auf

das Verzeichnis zeigen, in welches das kompilierte Wörterbuch kopiert wurde.

Ein Klick auf den "send data"-Knopf speichert die Änderungen. Damit die Änderungen angewandt werden, muss nun noch im Bereich master > operation der Befehl "applyConfiguration" ausgeführt werden (ebenfalls durch einen Klick auf den entsprechenden "send data"-Knopf).

Um den SuggestService nun zusammen mit dem Exalead Live Modul zu verwenden, muss der Name des angelegten SuggestServices in der Konfiguration des AutocompleteServlets in der web.xml angegeben werden (siehe Kapitel 3.7.4 AutocompleteServlet).

Soll der SuggestService auch in der Standard-Exalead-GUI zur Verfügung stehen, so muss er in der Exalead-Administration dafür erst aktiviert werden (siehe Abbildung 6-5: Aktivieren des SuggestServices). Dies ist nicht notwendig, wenn zur Suche nur das Exalead Live Modul verwendet wird.

In manchen Fällen wird der zuvor angelegte SuggestService in der Exalead-Administration nicht angezeigt. Dies kann durch einen Neustart des Servers behoben werden.

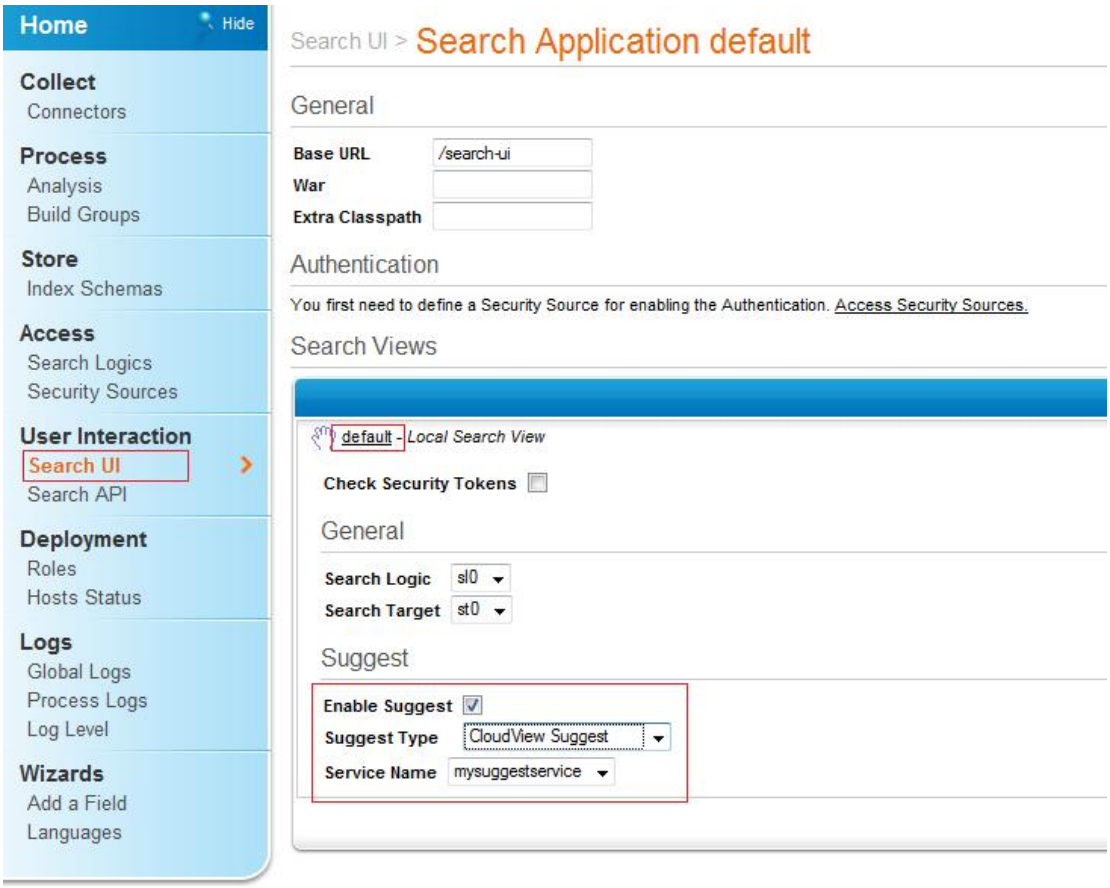

Abbildung 6-5: Aktivieren des SuggestServices

6.4 Anpassen der Suchvorlage

Um die Autovervollständigungsfunktion zusammen mit dem Exalead Live Modul zu verwenden, muss als letzter Schritt noch die Vorlage der Suchseite angepasst werden.

Der HTML-Header der Seite muss um folgende drei Zeilen erweitert werden:

```
<link type="text/css" href="css/redmond/jquery-ui-
1.8.6.custom.css" rel="stylesheet" />
<script type="text/javascript" src="js/jquery-
1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-
1.8.6.custom.min.js"></script>
```

Außerdem muss für das Eingabefeld des Suchformulars noch die Autovervollständigungsfunktion registriert werden:

```
<script>
$(function() {
```

```
        $("#searchquery").autocomplete({
                source: "do.autocomplete"
        });
});
</script>
```

In diesem Beispiel wird das Eingabefeld über die ID `searchquery` angesprochen. Die Angabe für `source` muss dem Mapping des AutocompleteServlets entsprechen, wie es in der web.xml angegeben wurde.

Wurden diese Anpassungen an der Vorlage durchgeführt, so unterstützt das Formular nun die Autovervollständigung der Sucheingaben wie in Abbildung 6-6 zu sehen.

Abbildung 6-6: Autovervollständigung

7 Migrationsleitfaden von Version 1.x auf 2.0

Sollten die Module FirstSpirit Exalead Live und FirstSpirit Exalead Content-Update bereits in einer früheren Version eingesetzt worden sein, so sind bei einem Update auf Version 2.0 folgende Punkte zu beachten:

7.1 web.xml

- Die web.xml ist um globale Parameter zur Angabe des Exalead-Servers zu erweitern (siehe Kapitel 3.5 Seite 17).
- Die Parameter des SearchServlets sind gemäß der neuen Servletbeschreibung anzupassen (siehe Kapitel 3.7.1 Seite 20).

- Die Parameter des SearchDownload-Servlets sind gemäß der neuen Servletbeschreibung anzupassen (siehe Kapitel 0 Seite 22).

7.2 Tag-Library

- Die Anzeige von Thumbnails für die Suchergebnisse wurde geändert (siehe Kapitel 3.12.11 Seite 48).
- Das Tag `has_meta_data` wird nicht mehr unterstützt. Stattdessen soll das Tag `hits_field` verwendet werden (siehe Kapitel 3.12.8 Seite 47).
- Das Tag `hits_field_attributes` wurde ebenfalls durch das Tag `hits_field` ersetzt.
- Die PageContext-Variable „path“ des `refinements`-Tags wurde in „title“ umbenannt.

7.3 Personalisierte Suchergebnisse

Bei der Nutzung von personalisierten Suchergebnissen gab es weitreichende Änderungen, die hauptsächlich die Konfiguration des Exalead-Servers betreffen. Alle wichtigen Informationen dazu sind dem Kapitel 4 auf Seite 56 zu entnehmen.

7.4 Installation des FirstSpirit-Moduls „Exalead Content-Update“

Beim Aktualisieren des Exalead Content-Update Moduls von einer früheren Version (< 2.0) auf die aktuelle Version, ist zu beachten, dass vor der Installation des Moduls die alte Version deinstalliert und die zuvor mit dem alten Modul installierte Exalead-Push Aktionsvorlage manuell entfernt wird (siehe Abbildung 7-1).

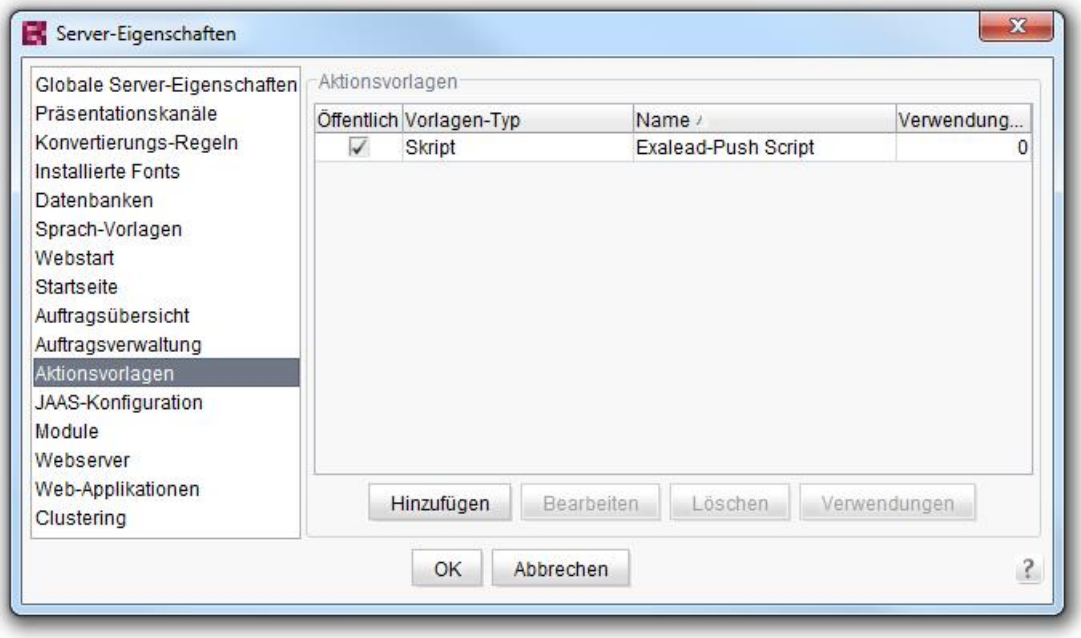

Abbildung 7-1: Aktionsvorlagen

Alle weiteren wichtigen Informationen, die die Installation des Moduls betreffen, können Kapitel 5.1 auf Seite 60 und Kapitel 5.2 auf Seite 62 entnommen werden.