Relevantes schneller finden – mit Lucene und Solr

Markus Klose

SHI Elektronische Medien GmbH

Markus Klose

- Projektmanagement
- Requirements Engineering
- Trainer für Solr

Wer ist SHI?

Seit mehr als 15 Jahren auf dem Markt mit Enterprise Solutions

Focus: Infrastruktursoftware (ECM, Enterprise Search, BPM)

ECM: Alfresco Authorized Partner seit 2008

Enterprise Search: LucidImagination Authorized Partner

Services: Requirements Engineering, Individualsoftware, Training, etc.

Überblick

- Klassische Suche
- Funktionen einer Solr Suche
- Begriffserklärungen
- Datenstruktur
- Was ist Lucene und was ist Solr ?
- Solr Konfiguration
- Solr Cache
- Solr Datenimport
- Solr Architekturen
- Solr Berechtigungen

Klassische Suche

Die klassischen Suchen:

-Datenbanksuche einer Applikation

-Volltextsuche über den Explorer

-Suche in speziellen Anwendungen wie Outlook, CRM, …

-Per Hand über mehrere Anwendungen

Bei den meisten Suchen gibt es dabei ein einfaches Suchfeld und eine sehr einfache Trefferliste.

Bei Datenbanksuchen oder Suchen in speziellen Anwendungen sind auch erweiterte Suchmasken verbreitet, bei denen in speziellen Feldern gesucht werden kann.

Funktionen einer Solr Suche

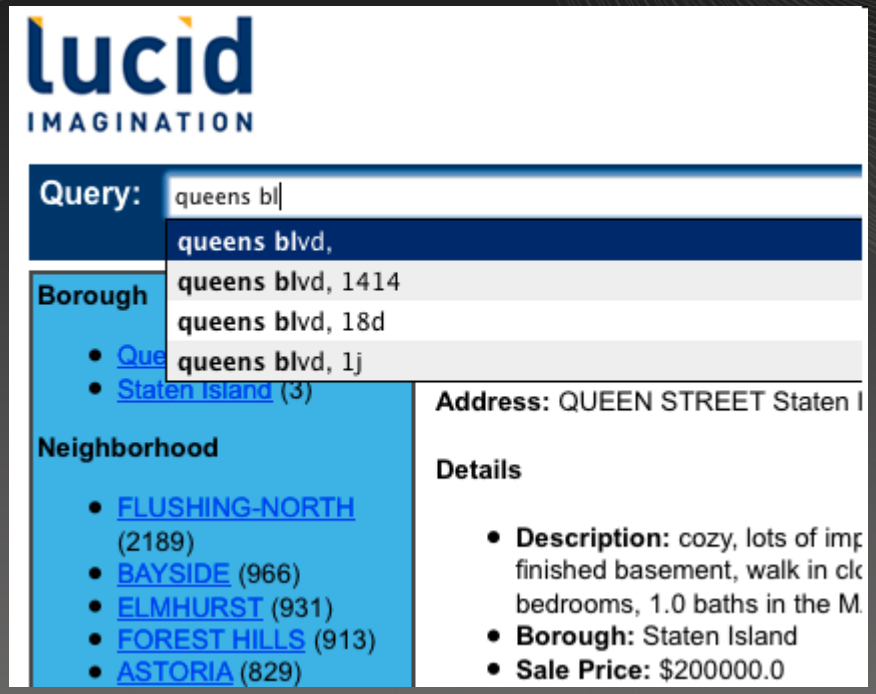
Ein moderner Solr Search Server bietet Suchfunktionen wie:

- Auto Suggest
- Did you mean…
- Faceten
- Field Collapsing
- More like this…
- Fast Highlighting (mit Term Vektoren)
- Deduplication
- Terms
- Clustering
- …

Funktionen einer Solr Suche

Auto Suggest

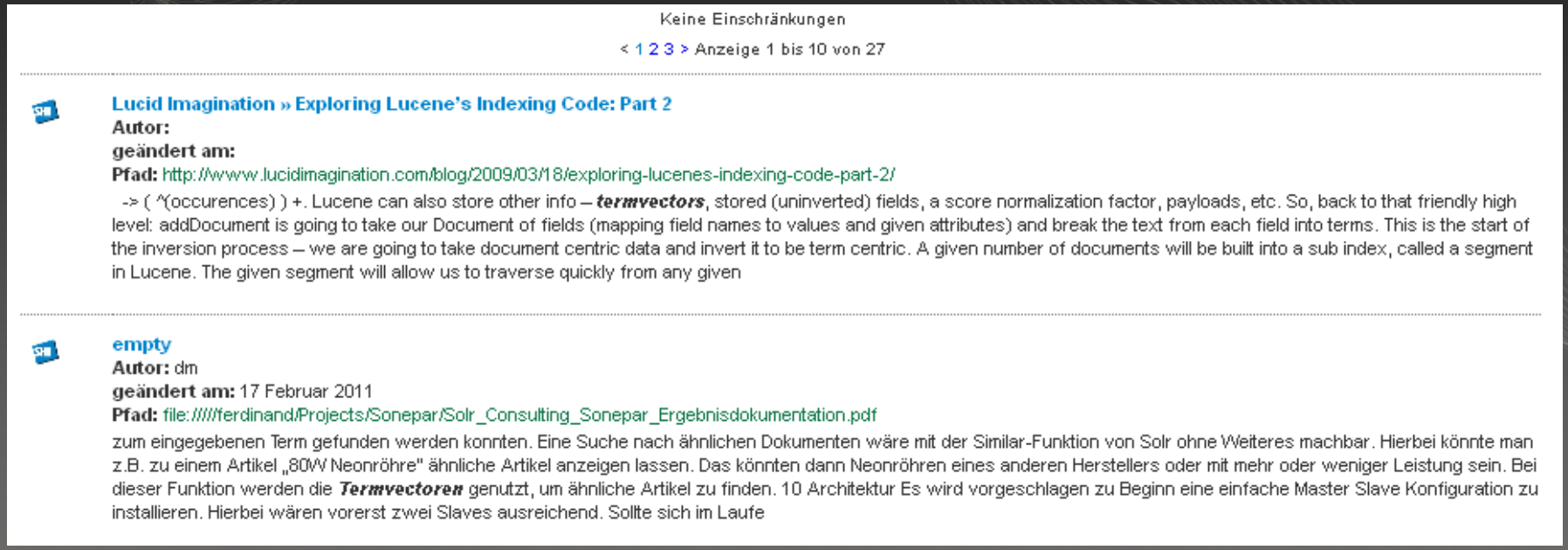
Liefert eine Vorschlagsliste für die eingegebenen Terme



Funktionen einer Solr Suche

Highlighting

- Liefert einen Textausschnitt eines festgelegten Metafelds des jeweiligen Trefferdokuments
- Markiert die gesuchten Worter oder dessen Synonyme im Textauschnitt



Funktionen einer Solr Suche

Did you mean

- Gibt Termvorschläge für andere Schreibweisen eines Terms an
- Es können auch mehrere Vorschläge generiert werden

Funktionen einer Solr Suche

Faceten

- Faceten zeigen Inhalte der aktuellen Treffer eines bestimmten Metafelds.
- Es wird auch angezeigt, wie viele Treffer den entsprechenden Term beinhalten.
- Über Filterqueries kann dann auf das entsprechende Metafeld eingeschränkt werden.

Funktionen einer Solr Suche

Field Collapsing /Result Grouping

- Gruppiert die Treffer nach einem bestimmten Metafeld
- Zeigt an das es in dieser Kategorie auch noch weitere Treffer gibt

Funktionen einer Solr Suche

More like this

- Kann zu einem bestimmten Treffer ähnliche Dokumente im Index finden
- Wird intern über Termvectoren gelöst

Funktionen einer Solr Suche

Deduplication

- Versucht gleiche Dokumente bei der Indexierung herauszufiltern.
- Wird intern über Termvectoren gelöst.

Funktionen einer Solr Suche

Terms

- Gibt alle Terme zu einem bestimmten Metafeld zurück.

Begriffserklärungen

Dokument

- Ist der wichtigste Bestandteil des Index
- Besteht aus einer Sammlung von Fields mit optionalen Boostfaktor
- Kann ein Word-Dokument, eine Web-Seite, ein Datenbankeintrag, … sein
- Stellt zugleich den Treffer da

Field

- Besteht aus Freitext, Keywords, Datum, Nummern oder ähnlichem
- Es ist auch möglich mehrere Values (Multivalue) in einem Field zu speichern
- Beispiel wäre titel, author, …

Begriffserklärungen

Stored Field

- Die Daten werden genauso wie Sie geliefert werden abgelegt, ohne analyse oder veränderung.
- Ist wichtig für Highlighting und Anzeigen in der Trefferliste

Indexed Fields

- Sind die Felder die später durchsuchbar sein sollen oder in Facetten darstellbar
- Indexed Fields können nur in der Trefferliste angezeigt werden wenn Sie gleichzeitig auch stored sind

Begriffserklärungen

Query

- Stellt die Suchanfrage da
- Kann ein einfacher Term aber auch eine komplexe Anfrage sein

Bsp.: *author:rm*

oder

author:rm AND (doctype:pdf OR doctype:doc) AND text:shi AND text:alfresco

Filter Query

- Eine Möglichkeit den Dokumentenbestand vor der eigentlichen Suche einzuschränken

Begriffserklärungen

Tokenization

- Der Prozess um eine Textsequenz in einzelne Stücke oder Einheiten zu unterteilen

Tokenfilter

- Untersucht Tokens und kann diese nachträglich noch bearbeiten

Term

- Das Ergebnis der Tokenization

CharFilter

- Kann vor der Tokenization durchgeführt werden
- Charbasierte Verarbeitung des Inputs

Begriffserklärungen

Facet

- Dynamische Zählung von in Kategorien eingeteilten Suchergebnissen

Instance

- Eine laufende Solr Applikation

Core

- Eine virtuelle Solr Instanz
- Jede Solr Instance kann ein oder mehrere Cores besitzen
- Ein Core kann eine eigene Konfiguration und eigenen Index haben

Shard

- Ein durchsuchbarer Solr-Subindex
- Eine Query kann über mehrere Shards suchen
- Keine globale Wertung möglich

Begriffserklärungen

Index

- Datenrepository in dem man suchen kann
- Lucene/Solr speichert die Daten als Inverted Index

Segment

- Teile des Gesamtindex
- An sich sind Segmente eigene kleine Indexe

Indexierung

- Prozess des Hinzufügens von Dokumenten in den Index

Datenstruktur

Beispiel:

```
<doc>
        <field name="employeeId">05991</field>
        <field name="office">Augsburg</field>
        <field name="skills">Perl</field>
        <field name="skills">Java</field>
        <field name="skills">C C++ und C#</field>
</doc>
```

Datenstruktur

Was ist ein inverted Index ?

ID	Title
1	Was ist Lucene
2	Was ist Solr
3	Warum Solr und nicht nur Lucene
4	Was ist ein Index
5	Solr Architektur

Term	Freq	Dokument Ids
Was	3	1,2,4
ist	3	1,2,4
Lucene	2	1,3
Solr	3	2,3,4
Warum	1	3
und	1	3
nicht	1	3
nur	1	3
ein	1	4
Index	1	4
Architektur	1	5

Datenstruktur

Was ist ein inverted Index ?

ID	Title
1	Was ist Lucene
2	Was ist Solr
3	Warum Solr und nicht nur Lucene
4	Was ist ein Index
5	Solr Architektur

Term	Freq	Dokument Ids
Was	3	1,2,4
ist	3	1,2,4
Lucene	2	1,3
Solr	3	2,3,4
Warum	1	3
und	1	3
nicht	1	3
nur	1	3
ein	1	4
Index	1	4
Architektur	1	5

ID	Text
1	Eine Java basierende Open Sou...
2	Ein Open Source Suchserver ...
3	Lucene ist lediglich eine Lib...
4	ID Title Term Freq Dokumentids...
5	Replication wird benutzt...

Term	Freq	Dokument Ids
Java	1	1
Open	2	1,2
Lucene	4	1,2,3,4
Term	1	4
Replication	1	5
...		

Wie kann über mehrere Felder gesucht werden ?

Es gibt hierbei mehrer Möglichkeiten:

Query erstellen

title:solr OR text:Solr

Dismax / eDismax Queryparser

Hier können Felder in denen gesucht werden soll explizit gesetzt werden.

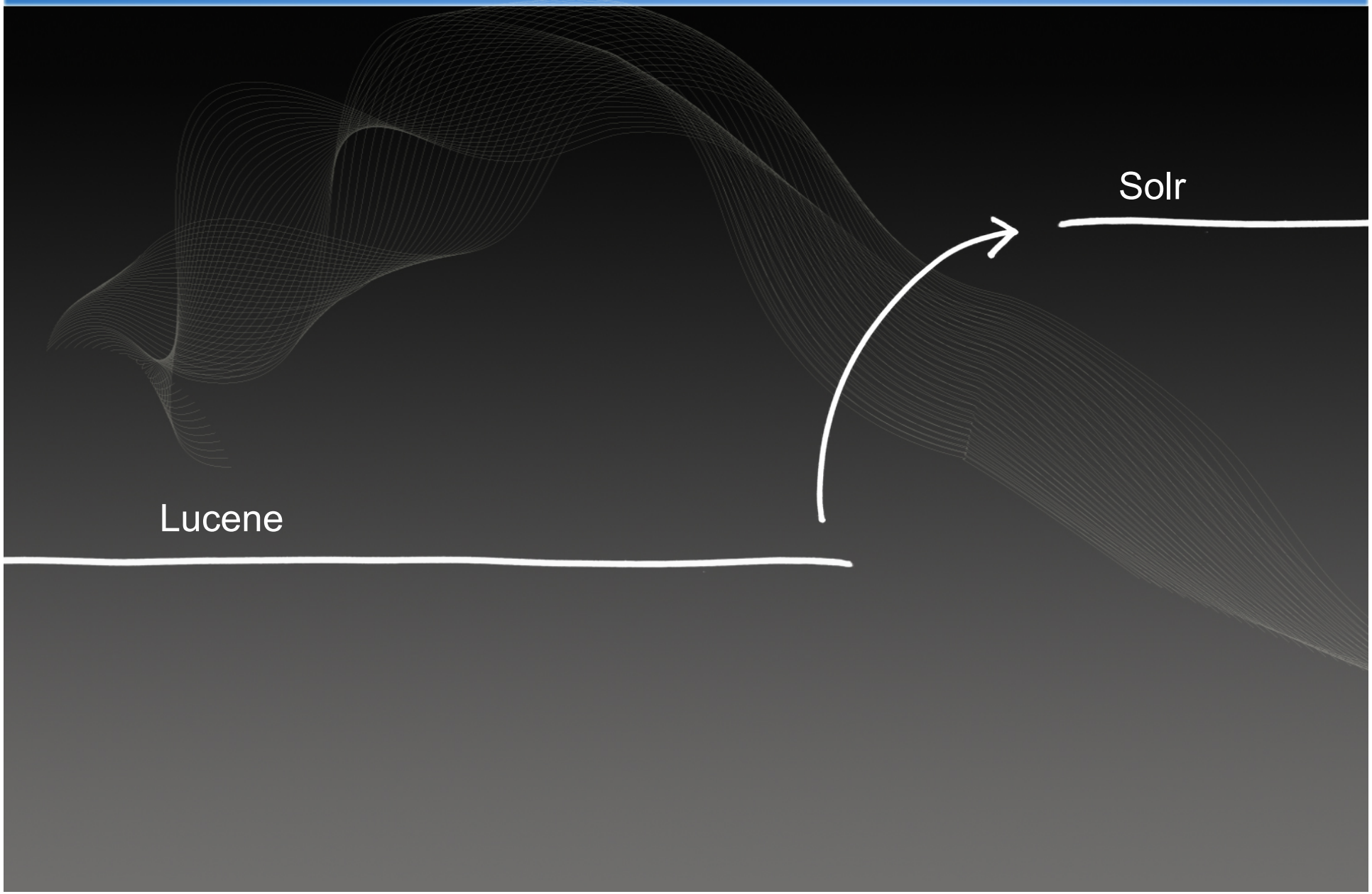
Intern erweiteret der Parser die Query

Suchfeld

Es kann ein defaultSearchfield in der Konfiguration hinterlegt werden.

Beim Indexieren werden dann alle Inhalte die durchsuchbar sein sollen in dieses Feld kopiert.

Was ist Lucene und was ist Solr?



Was ist Lucene und was ist Solr?

Was ist Lucene?

- Eine Java basierende Open Source Library mit guten Indexierungs- und Sucheigenschaften. Lucene ermöglicht so schnelle und Leichtgewichtige Suche und Indexierung
- 100% Java (Es gibt aber auch .Net, Perl und andere Versionen)
- Ausgereifte und stabile API
- Seit über 10 Jahren regelmäßig abgestimmt und verbessert
- Sauber implementiert und leicht in andere Anwendungen einzubinden
- Kompakte und portable Index Darstellung
- Anpassbare Textanalyzers, Spracherkennung und Highlighting
- Es ist kein Crawler und kein Text Extractionstool (Also keine Word Indexierung oder Webseitencrawlen)

Was ist Lucene und was ist Solr?

Was ist Solr?

- Ein Open Source Suchserver
- Solr indexiert verschiedene Contentquellen (DB, Filesystem, …)
- Vollwertiger Suchserver mit Lucene als Engine
- Die Entwicklung startete 2004 bei der Firma CNET
- 2006 wurde Solr an Apache SF übergeben
- Solr ist eine Web basierende Applikation (Zugriff über HTTP)
- Es gibt ein simples Webinterface für alle Operationen
- Solr läuft auf allen gängigen Servlet Containern (Tomcat, JBoss, WebSphere, …)

Was ist Lucene und was ist Solr?

Warum Solr verwenden und nicht nur Lucene?

- Lucene ist lediglich eine Library
- Solr ist eine skalierbare Suchplattform und bietet eine komplette Infrastruktur
- Es gibt ein Admin Interface
- Es gibt ein Cache Management und Tools zur Replikation
- Logging
- Statistiken
- und vieles mehr

Solr Konfiguration

Schema.xml

- Feldtypen
- Felder
- Copyfelder
- Tokenizer für Suche und Indexierung
- Filter für Suche und Indexierung
  - Synonyms
  - Regexp
  - SnowballPorter
  - …

Solr Konfiguration

Schema.xml

Feldtypen

```
<fieldType name="pint" class="solr.IntField" omitNorms="true"/>

<fieldType name="shi_author" class="solr.TextField" multiValued="true" omitNorms="true">
    <analyzer type="index">
        <tokenizer class="solr.KeywordTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter        class="solr.PatternReplaceFilterFactory"
                       pattern="^/o=shigruppe/ou=first\ administrative\ group/cn=recipients/cn="
                       replacement=„ replace="all"/>
        <filter        class="solr.SynonymFilterFactory"
                       synonyms="author_synonyms.txt" ignoreCase="true" expand="true"/>
    </analyzer>
    <analyzer type="query">
        <tokenizer class="solr.KeywordTokenizerFactory"/>
    </analyzer>
</fieldType>
```

Solr Konfiguration

Schema.xml

Fields

<field name="author" type="shi_author" indexed="true" stored="true" multiValued="true"/>

Copyfields

<copyField source="text" dest="spell"/>

Solr Konfiguration

Solrconfig.xml

- Componenten
- RequestHandler
- UpdateHandler
- Caching

…

Solr Konfiguration

Solrconfig.xml

Requesthandler Konfiguration

```
<requestHandler name="dismax" class="solr.SearchHandler">
    <lst name="defaults">
      <str name="defType">dismax</str>
      <str name="echoParams">explicit</str>
      <str name="mm">2<-1 5<-80%</str>
      <str name="qf">text^1.0 title^0.05 author^0.2 shi_quelle^0.4 shi_year adrp_keywords^0.5 shi_path^2.0</str>
      <str name="bf">div(20,log(ms(NOW,last_modified)))</str>
      <str name="fl">*,score</str>
    </lst>
    <lst name="invariants">
      <str name="facet.field">cat</str>
      <str name="facet.field">manu_exact</str>
      <str name="facet.query">price:[* TO 500]</str>
      <str name="facet.query">price:[500 TO *]</str>
    </lst>
</requestHandler>
```

Solr Konfiguration

Solrconfig.xml

SearchComponent Konfiguration

```
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">
    <str name="queryAnalyzerFieldType">textSpell</str>
    <lst name="spellchecker">
        <str name="name">default</str>
        <str name="field">spell</str>
        <str name="distanceMeasure">org.apache.lucene.search.spell.JaroWinklerDistance</str>
        <str name="spellcheckIndexDir">./spellchecker</str>
        <str name="accuracy">0.9</str>
        <str name="buildOnOptimize">true</str>
    </lst>
</searchComponent>
```

Solr Konfiguration

Solrconfig.xml

QueryResponseWriter Konfiguration

```
<queryResponseWriter name="xslt" class="org.apache.solr.request.XSLTResponseWriter">
    <int name="xsltCacheLifetimeSeconds">5</int>
</queryResponseWriter>
```

Solr Cache

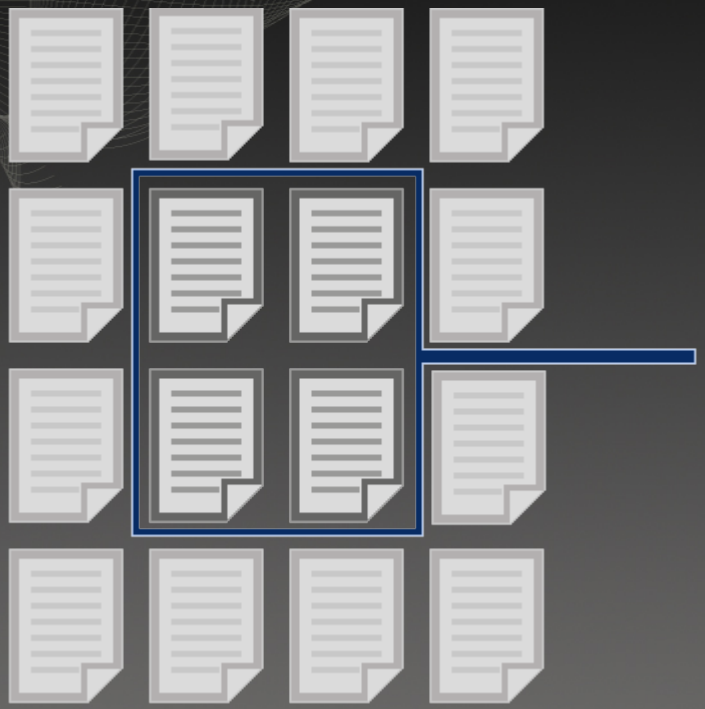
Solr arbeitet mit mehreren Caches, um Ideale Performance zu erreichen

- FilterCache
- QueryResultCache
- DocumentCache
- FieldValueCache
- FieldCache (Lucene)

Solr Cache

Filtercache

- Speichert die Ergebnisse der Filterqueries
- Auf den Subbestand wird dann die Query ausgeführt

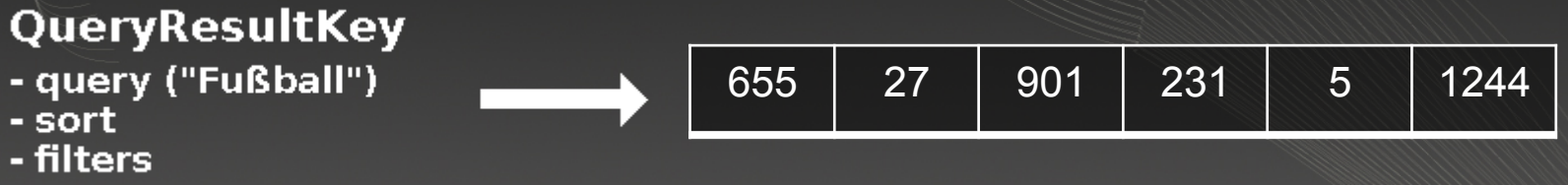


Eine Filter Query beschränkt die Anzahl der Dokumente, die durchsucht werden.

Solr Cache

QueryResultCache

- Der QueryResultCache enthält die DokumentIDs, die zu einer bestimmten Query gefunden wurden (als eine sortierte Liste).
- Kann auf eine maximale Anzahl pro Query beschränkt werden.



Solr Cache

DocumentCache

- Enthält LuceneDocument Objects
- Kann nicht durch auto-warmed gefüllt werden (nur indierekt)
- Kann sehr groß werden insbesondere bei sehr vielen Stored Values

655 →

LuceneDocument object:
- stored field 1
- stored field 2
- stored field 3
- etc.

Solr Cache

FieldCache (Lucene)

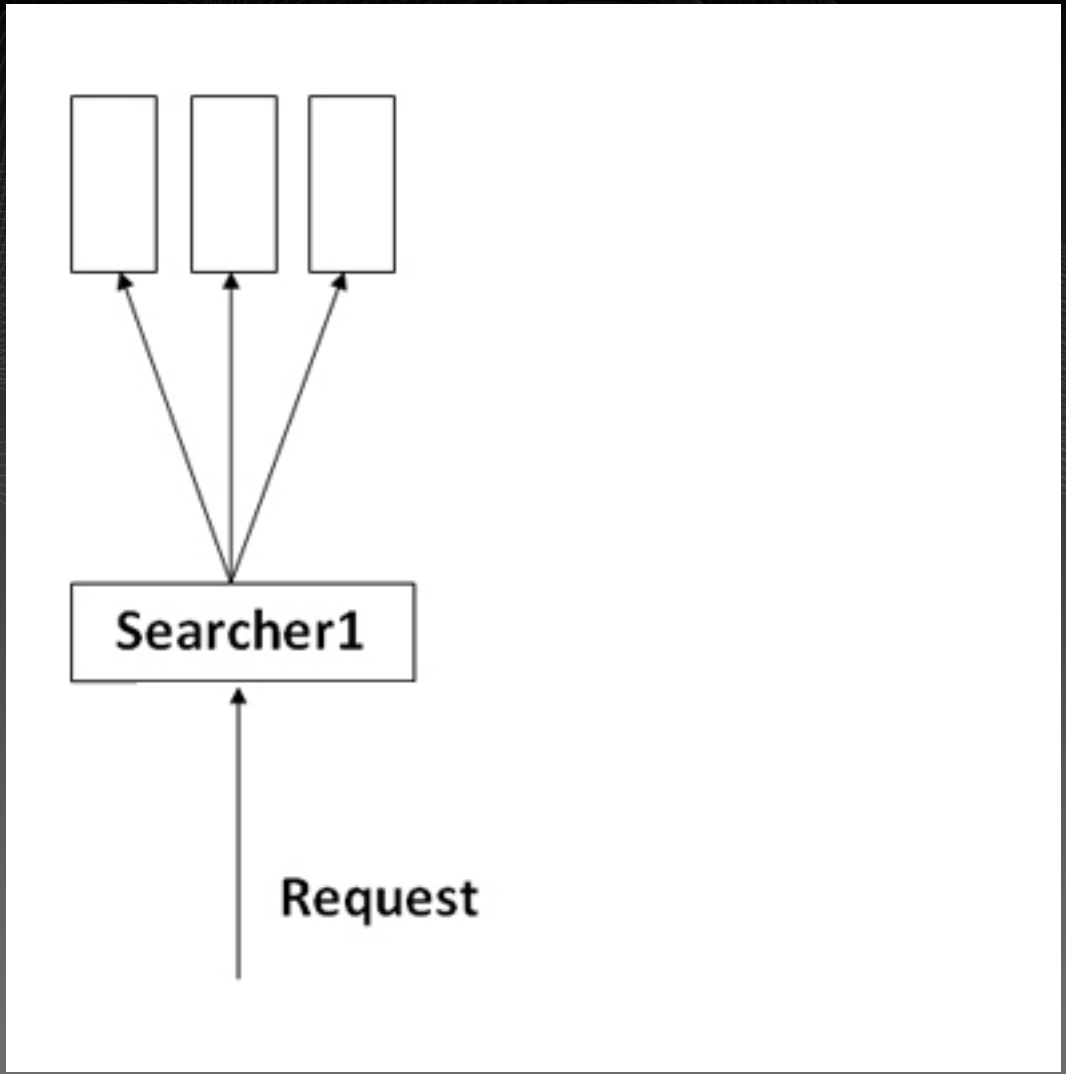
- Spielt eine Role beim Sortieren, Function Queries, Faceten …
- Cached alle Inhalte der entsprechenden Metafelder
- Wird nicht von Solr gemanaged
- Kann nicht durch auto-warmed gefüllt werden (nur indierekt)

FieldValueCache

- Ist dem LuceneFieldCache sehr ähnlich
- Kann mit MultivalueFields arbeiten
- Wird für Faceten und MultivalueFields verwendet
- Kann nicht durch auto-warmed gefüllt werden (nur indierekt)

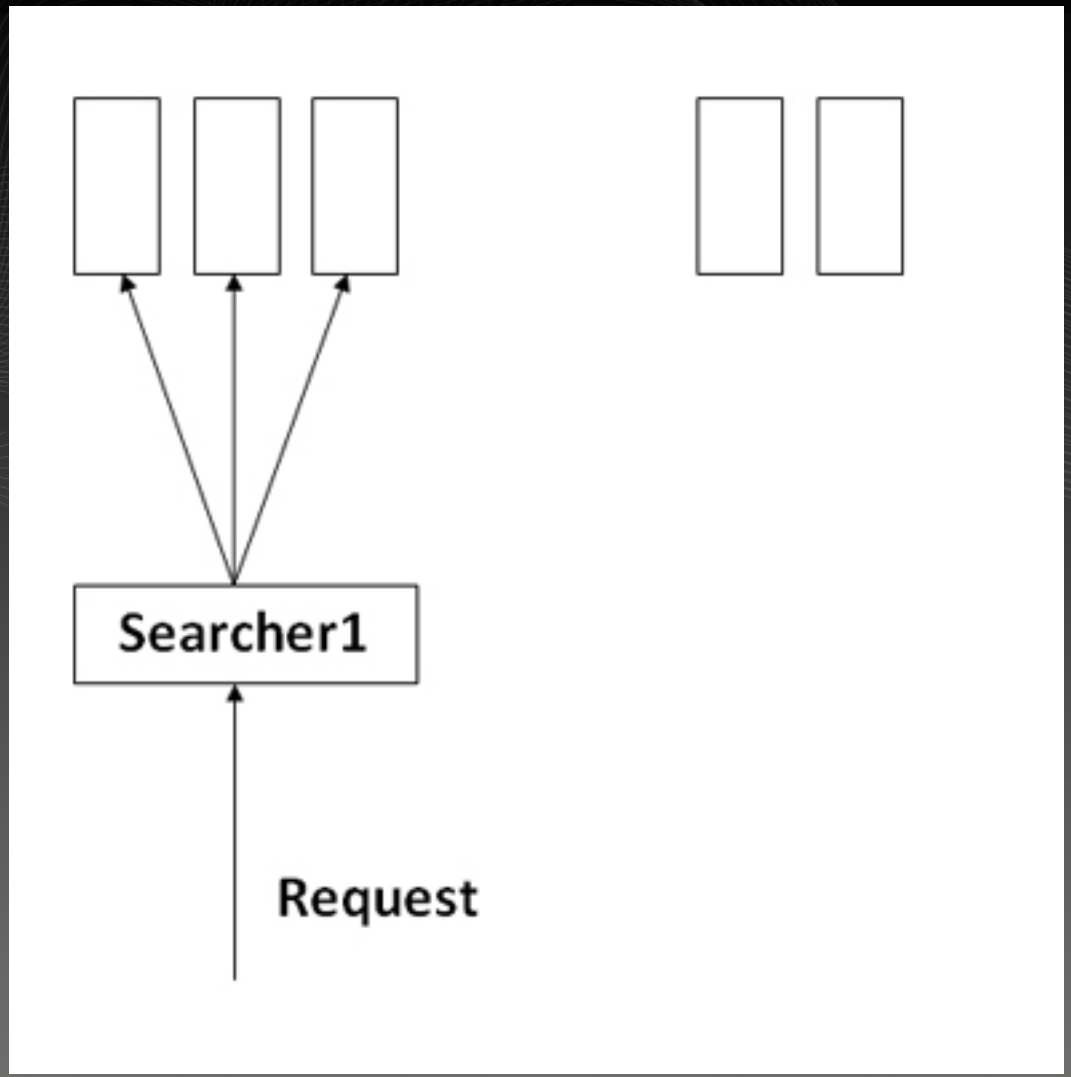
Solr Cache

Autowarm Searcher



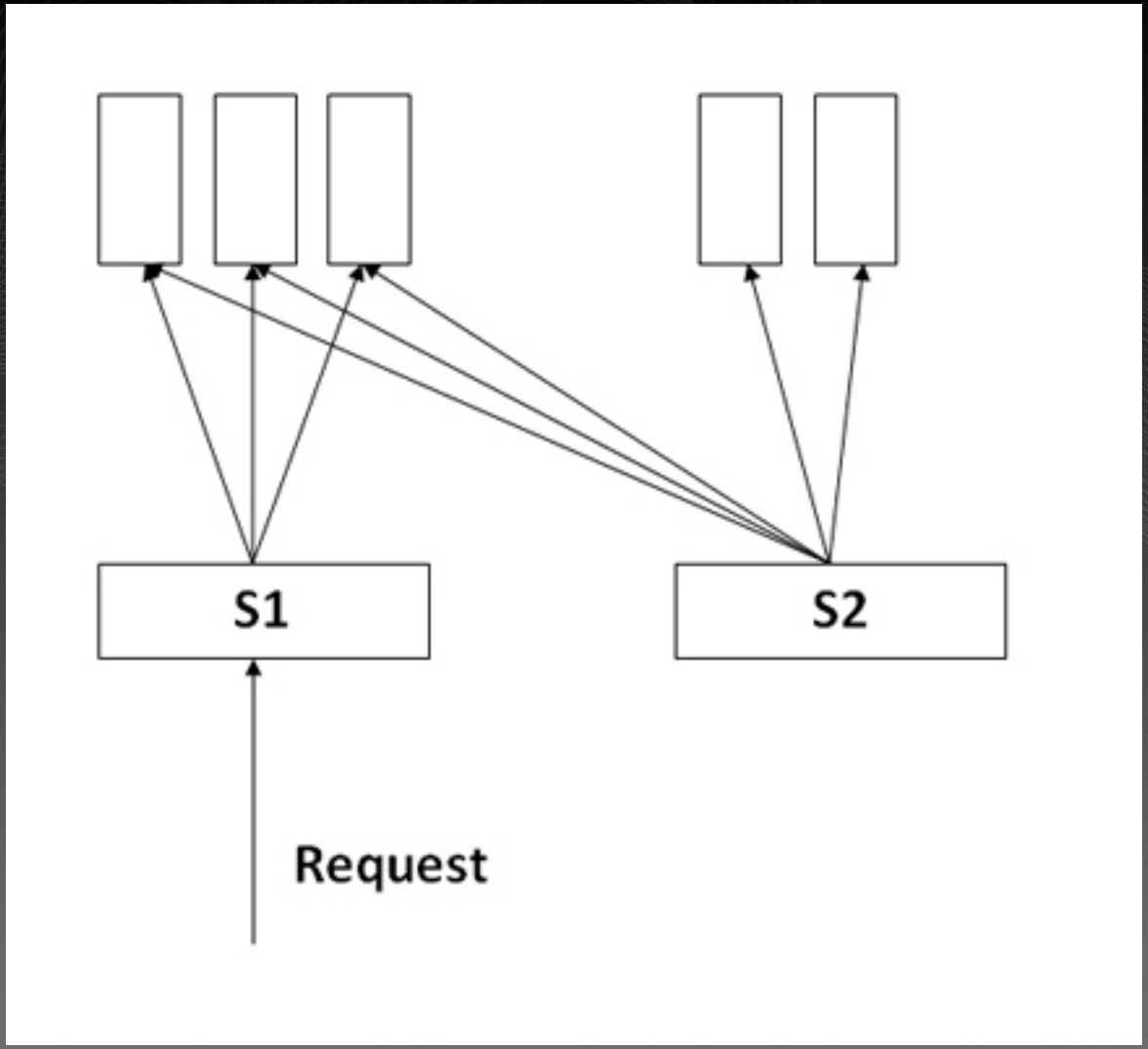
Solr Cache

Autowarm Searcher



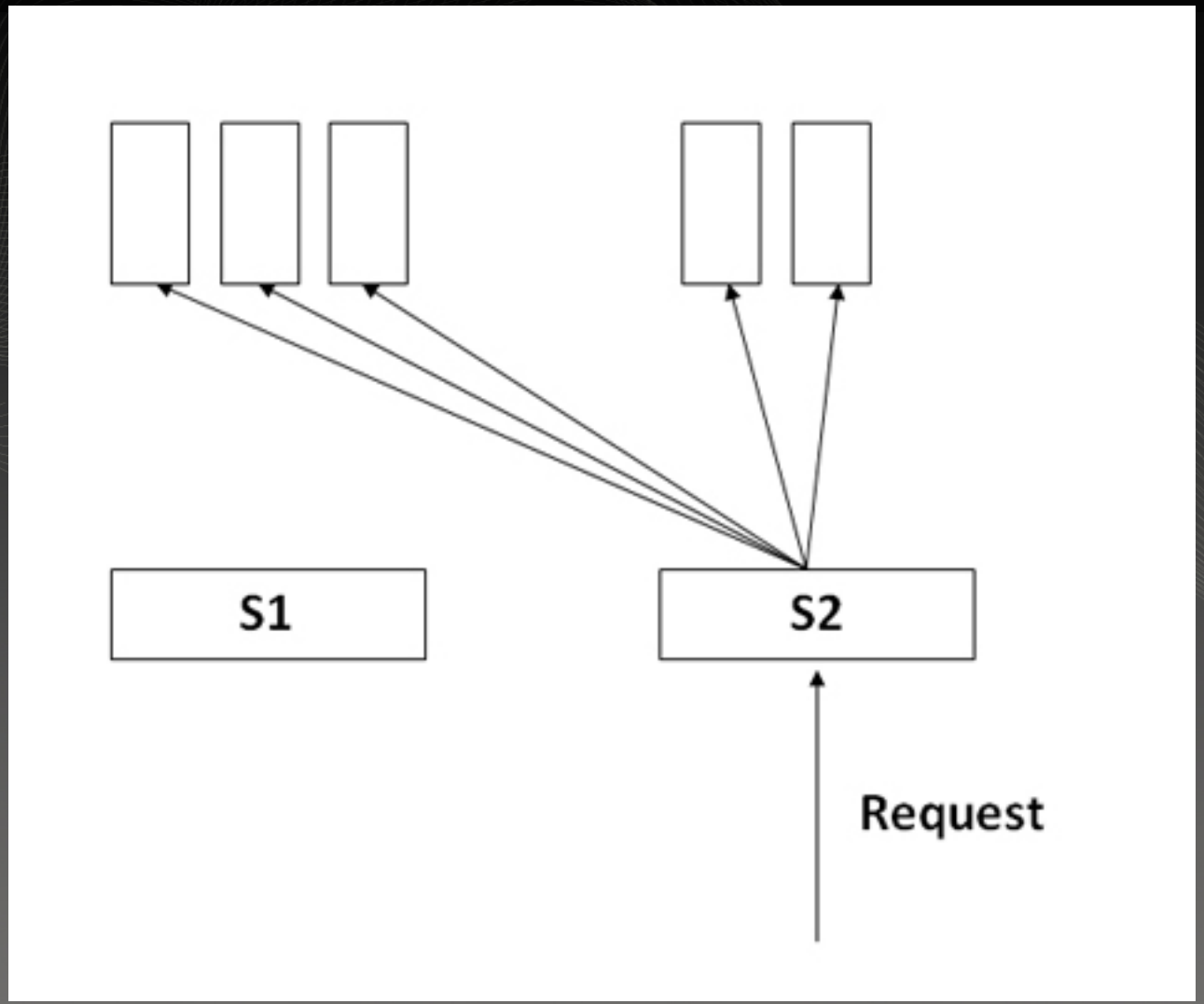
Solr Cache

Autowarm Searcher



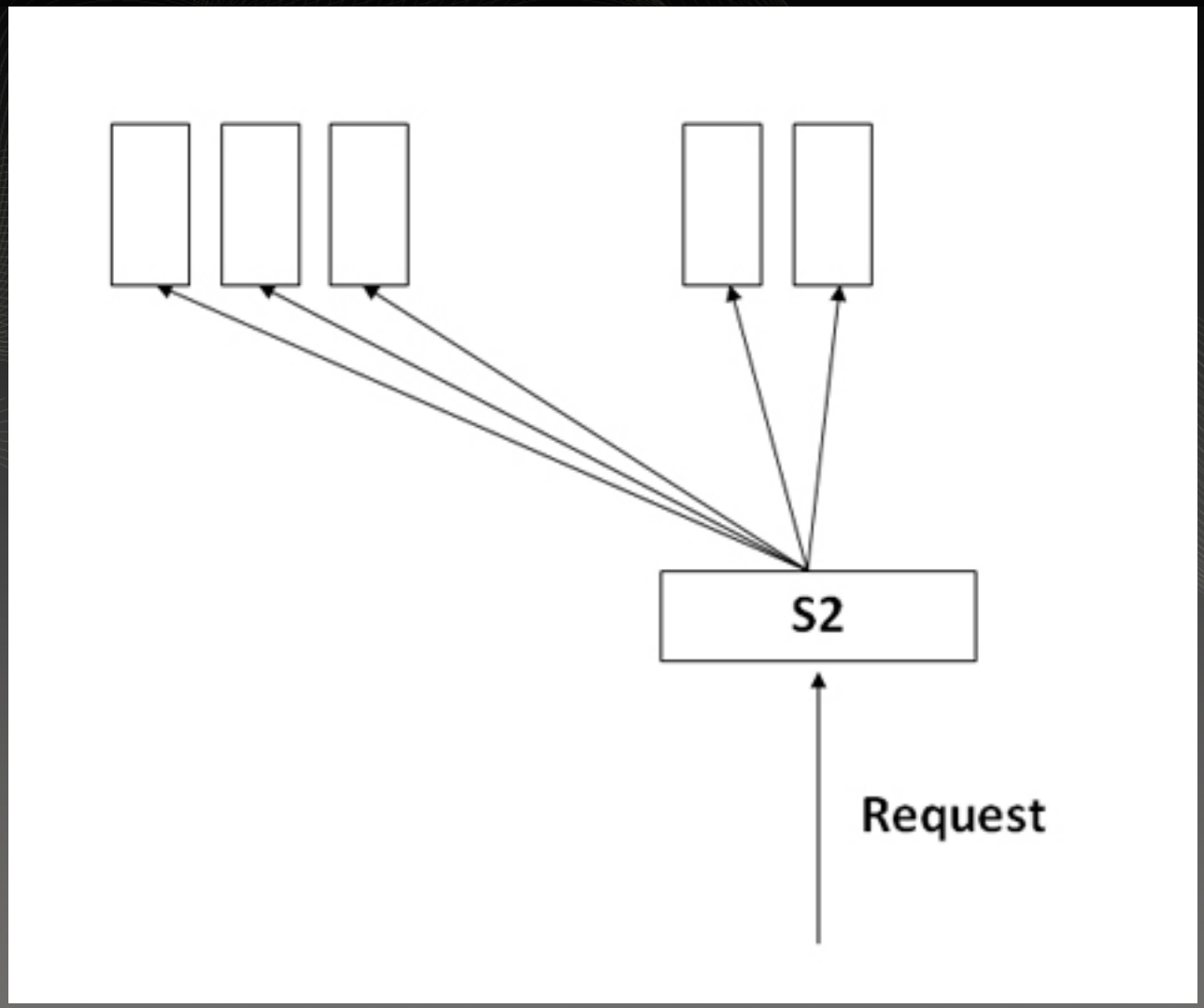
Solr Cache

Autowarm Searcher



Solr Cache

Autowarm Searcher



Solr Cache

Autowarming

- Füllen der Caches
- Wird durchgeführt bevor die Suchanfragen auf den Searcher geleitet werden
- Sollte auf max 1 Autowarmed Searcher begrenzt werden
- Wird immer dann ausgeführt wenn ein neuer Searcher geöffnet wird

Schnittstellen

- HTTP Request / Response
- API für
  - JS
  - PHP
  - Java
  - .Net/C#
  - …

Datenimport

- DIH (Data Import Handler)
  - Datenbanken
  - Filesystem
  - XML
  - Mailserver
- ManifoldCF
  - Datenbanken
  - Filesystem (inkl. Tika)
  - Web
  - AdressPlus CRM (by SHI)
  - Alfresco ECM, SharePoint, etc.
  - …
- Eigene Indexer in Programmiersprache der Wahl

Datenimport

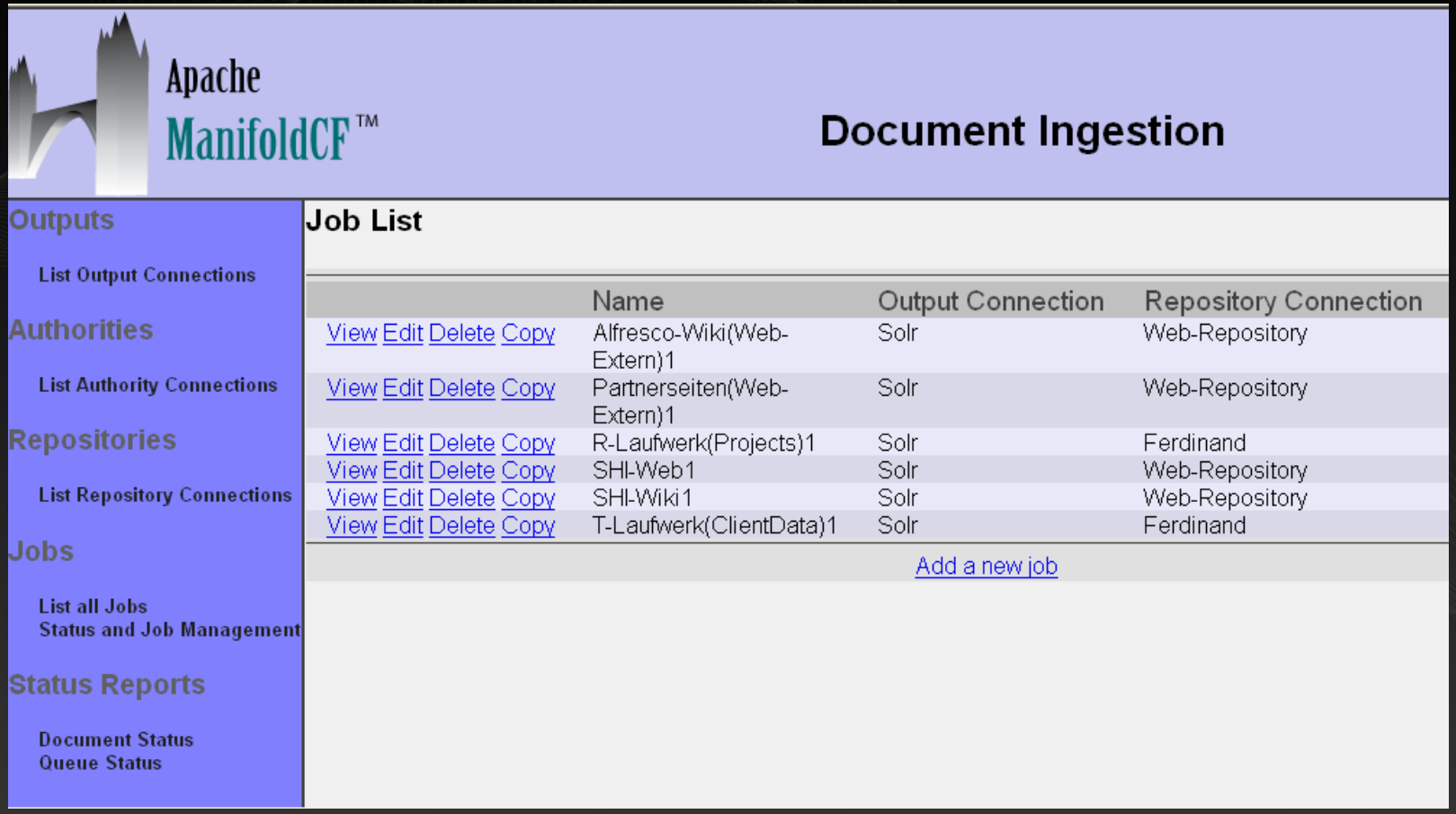
DIH (Data Import Handler)

Wird über Konfigurationsfile definiert und als Requesthandler eingebunden

```
<dataConfig>
<dataSource driver="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:/temp/example/ex" user="sa" />
<document name="products">
<entity name="item" query="select * from item">
        <field column="ID" name="id" />
        <field column="NAME" name="name" />
        <field column="MANU" name="manu" />
        <field column="WEIGHT" name="weight" />
        <field column="PRICE" name="price" />
        <field column="POPULARITY" name="popularity" />
        <entity name="feature" query="select description from feature where item_id='${item.ID}'">
                <field name="features" column="description" /> </entity>
        <entity name="item_category" query="select CATEGORY_ID from item_category where item_id='${item.ID}'">
                <entity name="category" query="select description from category where id = '${item_category.CATEGORY_ID}'">
                        <field column="description" name="cat" />
                </entity>
        </entity>
</entity>
</document>
</dataConfig>
```

Datenimport

ManifoldCF



Solr Architekturen

Basisarchitektur

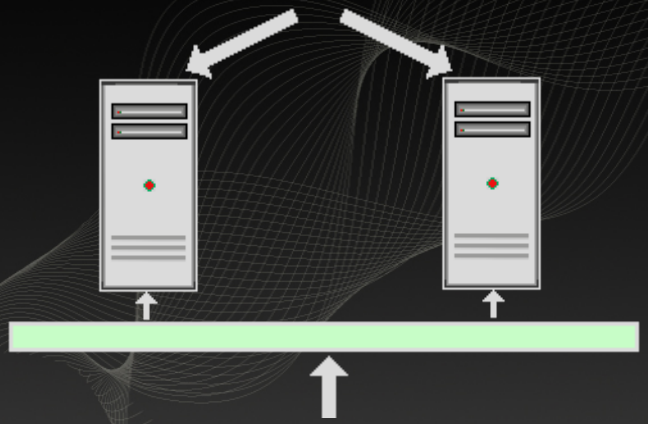
- Eine Instanz übernimmt sowohl die Indexierung, als auch die Suche

Solr Architekturen

Replication

- Wird benutzt, um die Suchanfragen auf mehre Server zu verteilen.
- Es gibt einen Master der die Indexierung vornimmt.
- Die Slaves holen sich den geänderten Index vom Master.
- Die Suchanfragen können mit einem Apache etc. auf die beiden Slaves verteilt werden.

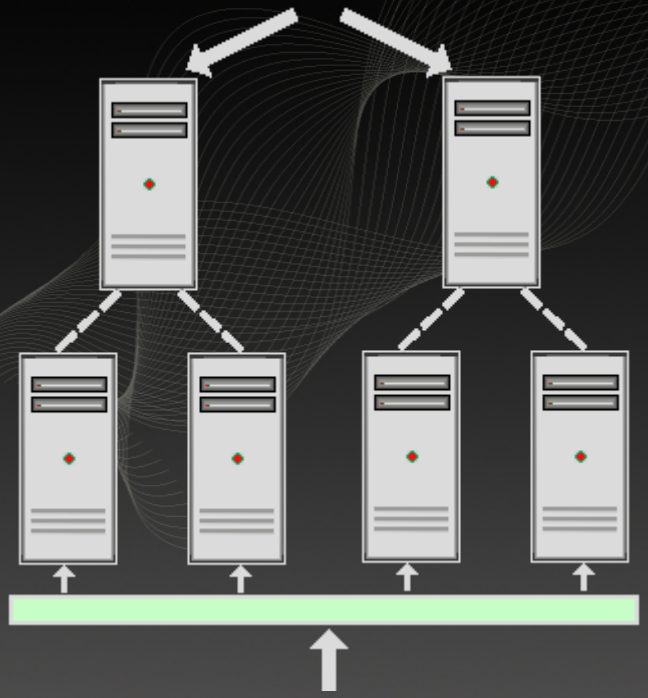
Solr Architekturen



Distributed

- Wird benutzt, um einen großen Index auf mehrere Server zu verteilen.
- Die Dokumente werden gleichmäßig auf die beiden Server verteilt.
- Beide Server können Angefragt werden und liefern ein gesamt Ergebnis zurück.

Solr Architekturen

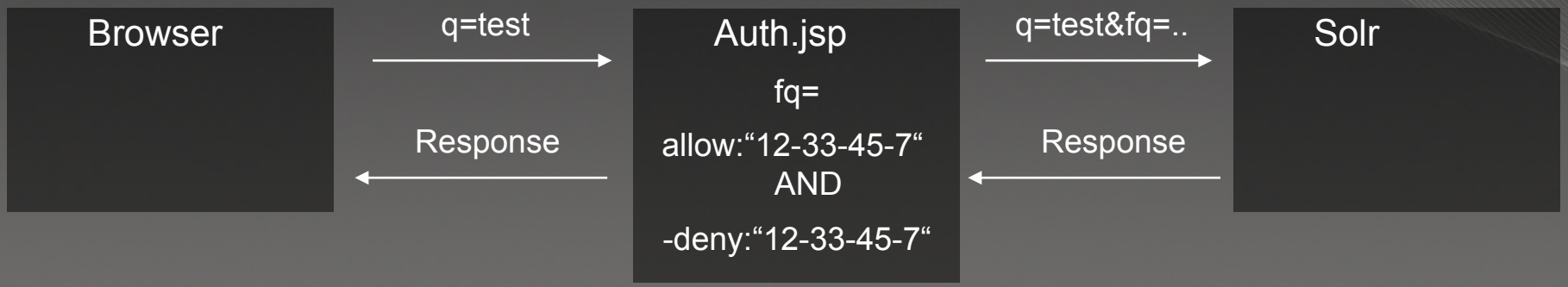


Distributed und Replication

- Wird benutzt, um einen großen Index auf mehrere Server zu verteilen.
- Die Dokumente werden gleichmäßig auf die beide Masterserver verteilt.
- Die Slaves holen sich die Daten von ihrem Master.
- Die Anfrage kann an jeden Slave gestellt werden.
- Der Aufbau eines solchen Systems kann auch auf Pools basieren.

Solr Berechtigungen

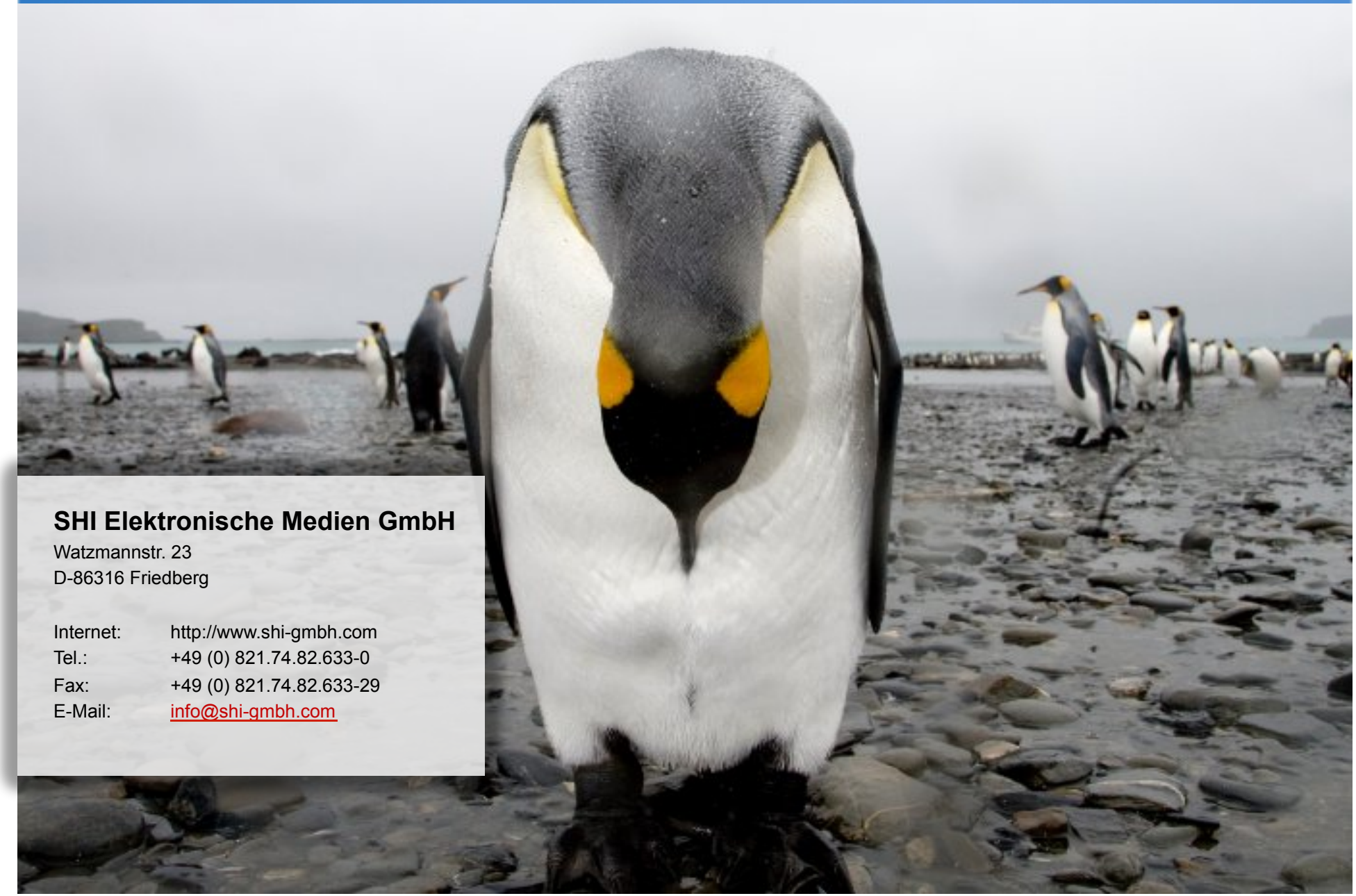
- Keine Standardimplementierung vorhanden
- Kann über Metafelder zur Indexierungszeit gelöst werden
- Bsp. ActiveDirectory bei SHI
  - Die Security Tokens für Allow und Deny werden in getrennten Multivaluefields gespeichert.
  - Eine Zwischenschicht ermittelt alle Security Tokens des aktuellen Benutzers.
  - Suche wird über Filterqueries automatisch eingeschränkt .



Solr Informationsquellen

- Solr Wiki http://wiki.apache.org/solr/
- Solr Jira (Bugtracking System) https://issues.apache.org/jira/browse/SOLR
- Solr Mailinglist http://lucene.apache.org/solr/mailing_lists.html
- SLUG-Deutschland e.V. (Solr Lucene User Group)
- SHI Blog http://www.shi-gmbh.com/blog

DANKE FÜR IHRE AUFMERKSAMKEIT!



SHI Elektronische Medien GmbH
Watzmannstr. 23
D-86316 Friedberg

Internet: http://www.shi-gmbh.com
Tel.: +49 (0) 821.74.82.633-0
Fax: +49 (0) 821.74.82.633-29
E-Mail: info@shi-gmbh.com