

Apache Solr Out Of The Box (OOTB)

Chris Hostetter

2008-11-05

<http://people.apache.org/~hossman/apachecon2008us/>

<http://lucene.apache.org/solr/>



Why Are We Here?

- Learn What Solr Is
- Opening the Box – aka: Getting Started
- Digging Deeper
 - schema.xml
 - solrconfig.xml
- Trial By Fire: Using Solr from Scratch
- But Wait! There's More!

What Is Solr?

Elevator Pitch

"Solr is an open source enterprise search server based on the Lucene Java search library, with XML/HTTP APIs, caching, replication, and a web administration interface."

What Does That Mean?

- Information Retrieval Application
- Java5 WebApp (WAR) With A Web Services-ish API
- Uses the Java Lucene Search Library
- Initially built At CNET
- 1 Year In The Apache Incubator
 - Solr 1.1 - December 22, 2006
- Lucene Sub-Project Since January 2007
 - Solr 1.2 - June 6, 2007
 - Solr 1.3 – September 16, 2008

Solr In A Nutshell

- Index/Query Via HTTP
- Comprehensive HTML Administration Interfaces
- Scalability - Efficient Replication To Other Solr Search Servers
- Extensible Plugin Architecture
- Highly Configurable And User Extensible Caching
- Flexible And Adaptable With XML Configuration
 - Customizable Request Handlers And Response Writers
 - Data Schema With Dynamic Fields And Unique Keys
 - Analyzers Created At Runtime From Tokenizers And TokenFilters

Getting Started

The Solr Tutorial

<http://lucene.apache.org/solr/tutorial.html>

- OOTB Quick Tour Of Solr Basics Using Jetty
- Comes With Example Config, Schema, And Data
- Trivial To Follow Along...

```
cd example
```

```
java -jar start.jar
```

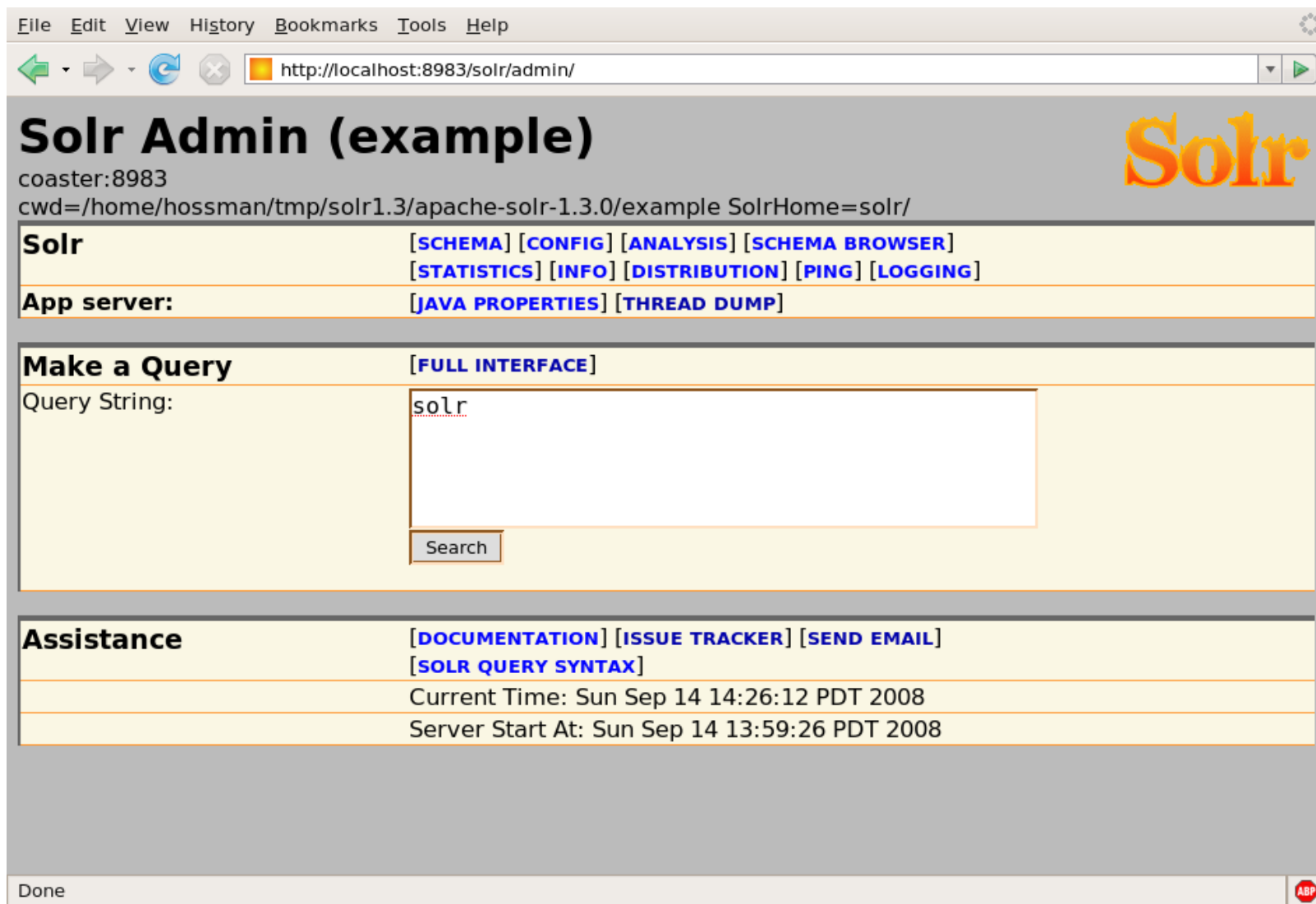
```
http://localhost:8983/solr/
```

```
cd example/exampledocs
```

```
java -jar post.jar *.xml
```



The Admin Console



The screenshot shows a web browser window with the address bar containing `http://localhost:8983/solr/admin/`. The page title is "Solr Admin (example)". The user is logged in as "coaster:8983" and the current working directory is `/home/hossman/tmp/solr1.3/apache-solr-1.3.0/example`. The Solr logo is visible in the top right corner.

The main content area is divided into several sections:

- Solr**: Contains links for [\[SCHEMA\]](#), [\[CONFIG\]](#), [\[ANALYSIS\]](#), [\[SCHEMA BROWSER\]](#), [\[STATISTICS\]](#), [\[INFO\]](#), [\[DISTRIBUTION\]](#), [\[PING\]](#), and [\[LOGGING\]](#).
- App server:**: Contains links for [\[JAVA PROPERTIES\]](#) and [\[THREAD DUMP\]](#).
- Make a Query**: Includes a link for [\[FULL INTERFACE\]](#). Below this is a "Query String:" label, a text input field containing "solr", and a "Search" button.
- Assistance**: Contains links for [\[DOCUMENTATION\]](#), [\[ISSUE TRACKER\]](#), and [\[SEND EMAIL\]](#), along with a link for [\[SOLR QUERY SYNTAX\]](#).

At the bottom of the page, the current time is "Sun Sep 14 14:26:12 PDT 2008" and the server start time is "Sun Sep 14 13:59:26 PDT 2008". The browser status bar shows "Done" and an "ABP" icon.

Configuration

schema.xml

- Where You Describe Your Data

solrconfig.xml

- Where You Describe How People Can Interact With Your Data

Loading Data

- Documents Can Be Added, Deleted, Or Replaced
- Canonical Message Transport: HTTP POST
- Canonical Message Format: XML...

```
<add><doc>
```

```
  <field name="id">SOLR</field>
```

```
  <field name="name">Apache Solr</field>
```

```
</doc></add>
```

```
<delete><id>SP2514N</id></delete>
```

```
<delete><query>name:DDR</query></delete>
```

Querying Data

HTTP GET or POST, params specifying query options...

```
http://solr/select?q=electronics
```

```
http://solr/select?q=electronics&sort=price+desc
```

```
http://solr/select?q=electronics&rows=50&start=50
```

```
http://solr/select?q=electronics&fl=name+price
```

```
http://solr/select?q=electronics&fq=inStock:true
```

Querying Data: Results

Canonical response format is XML...

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">1</int>
  </lst>
  <result name="response" numFound="14" start="0">
    <doc>
      <arr name="cat">
        <str>electronics</str>
        <str>connector</str>
      </arr>
      <arr name="features">
        <str>car power adapter, white</str>
      </arr>
      <str name="id">F8V7067-APL-KIT</str>
      ...
    </doc>
  </result>
</response>
```

Querying Data: Facet Counts

The screenshot shows a web browser window displaying search results for "japanese art" on the Smithsonian Institution's Cross Catalog Searching Center. The browser's address bar shows the URL: <http://siris-collections.si.edu/search/results.jsp?fq=type%3A%22Works+of+art%22&q=japanese+art&view>. The search bar contains "japanese art" and a search button. Below the search bar, there are options to "Only return results with online media" and a "home" link.

The search results are displayed in a grid view. The top navigation bar includes "list view", "grid view", "search history", and "help". Below this, it shows "Search Results (308 documents - page 1 of 16)" and "sort order: title | relevancy slideshow".

The left sidebar contains several facets for filtering the results:

- search terms:** query:japanese art, type: "Works of art"
- browse options:** browse & select
- online media:** Images (118), Online collections (2), Finding aids (1)
- type:** Works of art, Archival materials (216), Books (35), Exhibition catalogs (10), Photographs (10), Exhibitions (events) (9), Pictorial works (6), Diaries (3), Biographies (2), Catalogs (2)
- topic:** Art, Buddhist (14), Sculpture, Buddhist (14), Painting, Buddhist (9)

The main content area displays a grid of search results. The first row includes:

- Japanese Frets and Diapers n.d. Photo-Li...** (Image: A patterned textile design)
- Japanese Designs n.d. Photo-Lithograph** (Image: A patterned textile design)
- Japanese Medallions n.d. Photo-Lithograph...** (Image: Several circular medallions)
- Ethnographie n.d. Engraving** (Image: A detailed engraving of a scene with people)

The second row includes:

- Two Scenes with Japanese Women Eating an...** (Image: A scene with people eating)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)

The third row includes:

- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)
- Carpenter's Tool n.d. Drawing** (Image: A drawing of a carpenter's tool)

The browser's status bar at the bottom shows "Done" and a small "ABP" icon.

Querying Data: Facet Counts

Constraint counts can be computed for the whole result set using field values or explicit queries....

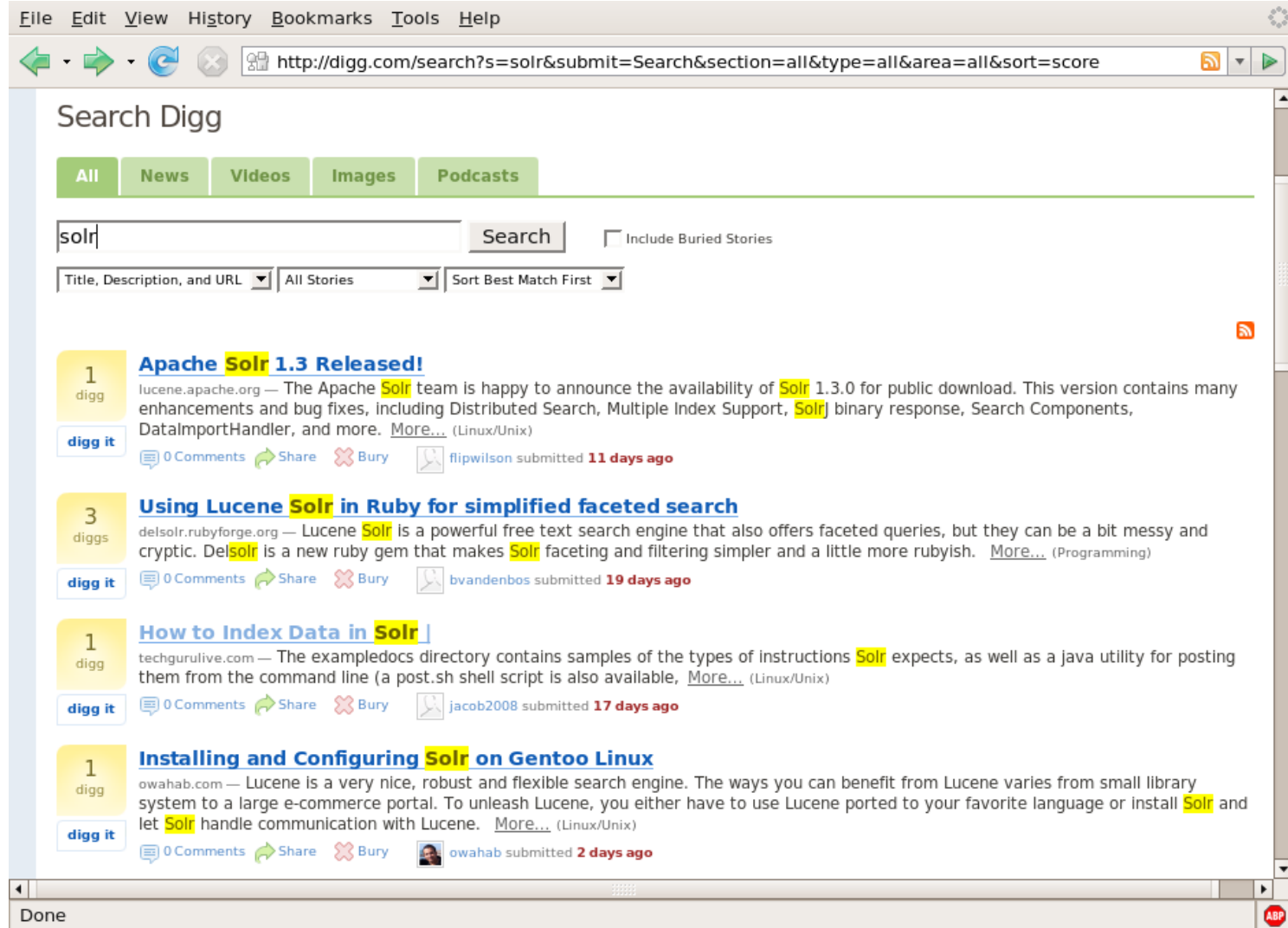
```
&facet=true&facet.field=cat&facet.field=inStock  
&facet.query=price:[0 TO 10]&facet.query=price:[10 TO *]
```

...

```
<lst name="facet_counts">  
  <lst name="facet_queries">  
    <int name="price:[0 TO 10]">0</int>  
    <int name="price:[10 TO *]">13</int>  
  </lst>  
  <lst name="facet_fields">  
    <lst name="inStock">  
      <int name="true">10</int>  
      <int name="false">4</int>  
    </lst>  
  </lst>
```

...

Querying Data: Highlighting



The screenshot shows a web browser window with the address bar containing the URL: `http://digg.com/search?s=solr&submit=Search§ion=all&type=all&area=all&sort=score`. The page title is "Search Digg". Below the title are navigation tabs for "All", "News", "Videos", "Images", and "Podcasts". A search bar contains the text "solr" and a "Search" button. To the right of the search bar is a checkbox labeled "Include Buried Stories". Below the search bar are three dropdown menus: "Title, Description, and URL", "All Stories", and "Sort Best Match First".

The search results are listed as follows:

- 1 digg** **Apache Solr 1.3 Released!**
lucene.apache.org — The Apache Solr team is happy to announce the availability of Solr 1.3.0 for public download. This version contains many enhancements and bug fixes, including Distributed Search, Multiple Index Support, Solr binary response, Search Components, DataImportHandler, and more. [More...](#) (Linux/Unix)
[digg it](#) [0 Comments](#) [Share](#) [Bury](#) [flipwilson submitted 11 days ago](#)
- 3 diggs** **Using Lucene Solr in Ruby for simplified faceted search**
delsolr.rubyforge.org — Lucene Solr is a powerful free text search engine that also offers faceted queries, but they can be a bit messy and cryptic. DelSolr is a new ruby gem that makes Solr faceting and filtering simpler and a little more rubyish. [More...](#) (Programming)
[digg it](#) [0 Comments](#) [Share](#) [Bury](#) [bvandenbos submitted 19 days ago](#)
- 1 digg** **How to Index Data in Solr**
techgurulive.com — The exampledocs directory contains samples of the types of instructions Solr expects, as well as a java utility for posting them from the command line (a post.sh shell script is also available, [More...](#) (Linux/Unix))
[digg it](#) [0 Comments](#) [Share](#) [Bury](#) [jacob2008 submitted 17 days ago](#)
- 1 digg** **Installing and Configuring Solr on Gentoo Linux**
owahab.com — Lucene is a very nice, robust and flexible search engine. The ways you can benefit from Lucene varies from small library system to a large e-commerce portal. To unleash Lucene, you either have to use Lucene ported to your favorite language or install Solr and let Solr handle communication with Lucene. [More...](#) (Linux/Unix)
[digg it](#) [0 Comments](#) [Share](#) [Bury](#) [owahab submitted 2 days ago](#)

Querying Data: Highlighting

Generates summary "fragments" of stored fields showing matches....

```
&hl=true&hl.fl=features&hl.fragsize=30
```

```
...
```

```
<lst name="highlighting">
```

```
  <lst name="F8V7067-APL-KIT">
```

```
    <arr name="features">
```

```
      <str>car power &lt;em>adapter</em>, white</str>
```

```
    </arr>
```

```
  </lst>
```

```
...
```

Digging Deeper

`schema.xml`

Describing Your Data

`schema.xml` is where you configure the options for various fields.

- Is it a number? A string? A date?
- Is there a default value for documents that don't have one?
- Is it created by combining the values of other fields?
- Is it stored for retrieval?
- Is it indexed? If so is it parsed? If so how?
- Is it a unique identifier?

Fields

- `<field>` Describes How You Deal With Specific Named Fields
- `<dynamicField>` Describes How To Deal With Fields That Match A Glob (Unless There Is A Specific `<field>` For Them)
- `<copyField>` Describes How To Construct Fields From Other Fields

```
<field          name="title"    type="text"     stored="false" />  
<dynamicField  name="price*"  type="sfloat"  indexed="true" />  
<copyField     source="*"      dest="catchall" />
```

Field Types

- Every Field Is Based On A `<fieldType>` Which Specifies:
 - The Underlying Storage Class (FieldType)
 - The Analyzer To Use Or Parsing If It Is A Text Field
- OOTB Solr Has 18 FieldType Classes

```
<fieldType name="sfloat" class="solr.SortableFloatField"
  sortMissingLast="true" omitNorms="true" />
```

```
<fieldtype name="string" class="solr.StrField"
  indexed="true" stored="true" />
```

```
<fieldtype name="unstored" class="solr.StrField"
  indexed="true" stored="false" />
```

Analyzers

- 'Analyzer' Is A Core Lucene Class For Parsing Text
- Solr Includes 18 Lucene Analyzers That Can Be Used OOTB If They Meet Your Needs

```
<fieldType name="text_greek" class="solr.TextField">  
  <analyzer class="org.apache.lucene.analysis.el.GreekAnalyzer"/>  
</fieldType>
```

...BUT WAIT!

Tokenizers And TokenFilters

- Analyzers Are Typical Comprised Of Tokenizers And TokenFilters
 - Tokenizer: Controls How Your Text Is Tokenized
 - TokenFilter: Mutates And Manipulates The Stream Of Tokens
- Solr Lets You Mix And Match Tokenizers and TokenFilters In Your `schema.xml` To Define Analyzers On The Fly
- OOTB Solr Has Factories For 12 Tokenizers and 36 TokenFilters
- Many Factories Have Customization Options -- Limitless Combinations

Tokenizers And TokenFilters

```
<fieldType name="text" class="solr.TextField">
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StopFilterFactory words="stopwords.txt"/>
    <filter class="solr.WordDelimiterFilterFactory"
      generateWordParts="1" generateNumberParts="1"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPorterFilterFactory"
      protected="protwords.txt"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory"
      synonyms="synonyms.txt" expand="true"/>
  </analyzer>
</fieldType>
```


Notable Token(izers|Filters)

- StandardTokenizerFactory
- HTMLStripWhitespaceTokenizerFactory
- KeywordTokenizerFactory
- NGramTokenizerFactory
- PatternTokenizerFactory

- EnglishPorterFilterFactory
- SynonymFilterFactory
- StopFilterFactory
- ISOLatin1AccentFilterFactory
- PatternReplaceFilterFactory

Analysis Tool

- HTML Form Allowing You To Feed In Text And See How It Would Be Analyzed For A Given Field (Or Field Type)
- Displays Step By Step Information For Analyzers Configured Using Solr Factories...
 - Token Stream Produced By The Tokenizer
 - How The Token Stream Is Modified By Each TokenFilter
 - How The Tokens Produced When Indexing Compare With The Tokens Produced When Querying
- Helpful In Deciding Which Tokenizer/TokenFilters You Want To Use For Each Field Based On Your Goals

Analysis Tool: Output

File Edit View History Bookmarks Tools Help

http://localhost:8983/solr/admin/analysis.jsp

Field name text

Field value (Index)
 verbose output
 highlight matches

Field value (Query)
 verbose output

Analyze

Index Analyzer
org.apache.solr.analysis.WhitespaceTokenizerFactory {}

term position	1	2	3	4	5	6	7	8
term text	The	Quick/Brown	Fox	Jumped	Over	The	Lazy	Dog
term type	word	word	word	word	word	word	word	word
source start,end	0,3	4,15	16,19	20,26	27,31	32,35	36,40	41,44
payload								

org.apache.solr.analysis.StopFilterFactory {enablePositionIncrements=true, words=stopwords.txt, ignoreCase=true}

term position	2	3	4	5	7	8
term text	Quick/Brown	Fox	Jumped	Over	Lazy	Dog
term type	word	word	word	word	word	word
source start,end	4,15	16,19	20,26	27,31	36,40	41,44
payload						

org.apache.solr.analysis.WordDelimiterFilterFactory {catenateWords=1, catenateNumbers=1, splitOnCaseChange=1, catenateAll=0, generateNumberParts=1, generateWordParts=1}

term position	2	3	4	5	6	8	9
term text	Quick	Brown	Fox	Jumped	Over	Lazy	Dog
		QuickBrown					

Done

Digging Deeper

`solrconfig.xml`

Interacting With Your Data

`solrconfig.xml` is where you configure options for how this Solr instance should behave.

- Low-Level Index Settings
- Performance Settings (Cache Sizes, etc...)
- Types of Updates Allowed
- Types of Queries Allowed

Note:

- `solrconfig.xml` depends on `schema.xml`.
- `schema.xml` does not depend on `solrconfig.xml`.

Request Handlers

- Type Of Request Handler Determines Options, Syntax, And Logic For Processing Requests
- OOTB Indexing Handlers:
 - XmlUpdateRequestHandler
 - CSVRequestHandler
 - DataImportHandler
- OOTB Searching Handler:
 - SearchHandler + QParsers

Indexing Message Transports

- Request Handlers Deal Abstractly With "Content Streams"
- Several Ways To Feed Data To Solr As A Content Stream...
 - Raw HTTP POST Body
 - HTTP Multipart "File Uploads"
 - Read From Local File
 - Read From Remote URL
 - URL Param String

SearchHandler

- SearchHandler Executes Query With Filtering, Pagination, Return Field List, Highlighting, Faceting, Etc...
- Uses a QParser To Parse Query String
- OOTH Solr Provides Two Great QParsers You Can Use Depending On Your Needs

`&defType=lucene` (Default)

`&defType=dismax`

LuceneQParserPlugin

- Main Query String Expressed In The "Lucene Query Syntax"
- Clients Can Search With Complex "Boolean-ish" Expressions Of Field Specific Queries, Phrase Queries, Range Queries, Wildcard And Prefix Queries, Etc...
- Queries Must Parse Cleanly, Special Characters Must Be Escaped

```
?q=name:solr+%2B(cat:server+cat:search)+popular:[5+T0+*]
```

```
?q=name:solr^2+features:"search+server"~2
```

```
?q=features:scal*
```

LuceneQParserPlugin

```
q = name:solr +(cat:server cat:search) popular:[5 TO *]
```

```
q = name:solr^2 features:"search server"~3
```

```
q = features:scal*
```

Good for situations where you want to give smart users who understand both the syntax and the fields of your index the ability to search for very specific things.

DisMaxQParserPlugin

- Main Query String Expressed As A Simple Collection Of Words, With Optional "Boolean-ish" Modifiers
- Other Params Control Which Fields Are Searched, How Significant Each Field Is, How Many Words Must Match, And Allow For Additional Options To Artificially Influence The Score
- Does Not Support Complex Expressions In The Main Query String

```
?q=%2Bsolr+search+server&qf=features+name^2&bq=popular:[5+T0+*]
```

DisMaxQParserPlugin

```
q = +solr search server  
& qf = features name^2  
& bq = popular:[5 TO *]
```

Good for situations when you want to pass raw input strings from novice users directly to Solr.

QParser For Other Params?

- By Default Other Query Params Use LuceneQParser
- Prefix Notation Exists To Override This, And Customize Behavior

```
&bq={!dismax qf=desc^2,review}cheap
```

```
&fq={!lucene df=keywords}lucene solr java
```

Request Handler Configuration

- Multiple Instances Of Various RequestHandlers, Each With Different Configuration Options, Can Be Specified In Your `solrconfig.xml`
- Any Params That Can Be Specified In A URL, Can Be "Baked" Into Your `solrconfig.xml` For A Particular RequestHandler Instance
- Options Can Be:
 - "defaults" Unless Overridden By Query Params
 - "appended" To (Multi-Valued) Query Params
 - "invariants" That Suppress Query Params

`http://solr/select?q=ipod`

`http://solr/simple?q=ipod`

`http://solr/complex?q=ipod`



Example: Handler Configuration

```
<requestHandler name="/select" class="solr.SearchHandler" />
<requestHandler name="/simple" class="solr.SearchHandler" >
  <lst name="defaults">
    <str name="defType">dismax</str>
    <str name="qf">catchall</str>
  </lst>
</requestHandler>
<requestHandler name="/complex" class="solr.SearchHandler" >
  <lst name="defaults">
    <str name="defType">dismax</str>
    <str name="qf">features^1 name^2</str>
  </lst>
  <lst name="appends">
    <str name="fq">inStock:true</str>
  </lst>
  <lst name="invariants">
    <str name="facet">>false</str>
    ...
```

Output: Response Writers

- Response Format Can Be Controlled Independently From Request Handler Logic
- Many Useful Response Writers OOTB

```
http://solr/select?q=electronics
```

```
http://solr/select?q=electronics&wt=xml
```

```
http://solr/select?q=electronics&wt=json
```

```
http://solr/select?q=electronics&wt=python
```

```
http://solr/select?q=electronics&wt=ruby
```

```
http://solr/select?q=electronics&wt=php
```

```
http://solr/select?q=electronics&wt=xslt&tr=example.xsl
```

```
<queryResponseWriter name="xml" default="true"  
  class="solr.XMLResponseWriter" />
```


Trial By Fire

Using Solr From Scratch

Installing Solr

- Put The solr.war Where Your Favorite Servlet Container Can Find It
- Create A "Solr Home" Directory
- Steal The Example solr/conf Files
- Point At Your Solr Home Using Either:
 - JNDI
 - System Properties
 - The Current Working Directory

(Or just use the Jetty example setup.)

Example: Tomcat w/JNDI

```
<Context docBase="f:/solr.war"  
  debug="0"  
  crossContext="true" >  
  <Environment name="solr/home"  
    value="f:/my/solr/home"  
    type="java.lang.String"  
    override="true" />  
</Context>
```

Minimalist Schema

```
<schema name="minimal" version="1.1">
  <types>
    <fieldType name="string" class="solr.StrField"/>
  </types>
  <fields>
    <dynamicField name="*"          type="string"
                  indexed="true"  stored="true" />
  </fields>
  <!-- A good idea, but not strictly necessary
    <uniqueKey>id</uniqueKey>
    <defaultSearchField>catchall</defaultSearchField>
  -->
</schema>
```

Feeding Data From The Wild

- I Went Online And Found A CSV File Containing Data On Books
- Deleted Some Non UTF-8 Characters
- Made Life Easier For Myself By Renaming The Columns So They Didn't Have Spaces

```
curl 'http://solr/update/csv?commit=true'  
  -H 'Content-type:text/plain; charset=utf-8'  
  --data-binary @books.csv
```

Understanding The Data: Luke

- The LukeRequestHandler Is Based On A Popular Lucene GUI App For Debugging Indexes (Luke)
- Allows Introspection Of Field Information:
 - Options From The Schema (Either Explicit Or Inherited From Field Type)
 - Statistics On Unique Terms And Terms With High Doc Frequency
 - Histogram Of Terms With Doc Frequency Above Set Thresholds
- Helpful In Understanding The Nature Of Your Data
- Schema Browser: Luke On Steroids

Example: Luke Output

```
File Edit View History Bookmarks Tools Help
http://localhost:8983/solr/admin/luke
+<lst name="reviews"></lst>
- <lst name="publisher">
  <str name="type">string</str>
  <str name="schema">I-S-----</str>
  <str name="index">I-S-----</str>
  <int name="docs">854</int>
  <int name="distinct">2</int>
- <lst name="topTerms">
  <int name="Hart Publishing">666</int>
  <int name="Intersentia">188</int>
</lst>
- <lst name="histogram">
  <int name="2">0</int>
  <int name="4">0</int>
  <int name="8">0</int>
  <int name="16">0</int>
  <int name="32">0</int>
  <int name="64">0</int>
  <int name="128">0</int>
  <int name="256">1</int>
  <int name="512">0</int>
  <int name="1024">1</int>
</lst>
</lst>
- <lst name="contents">
  <str name="type">string</str>
  <str name="schema">I-S-----</str>
  <str name="index">I-S-----</str>
  <int name="docs">166</int>
  <int name="distinct">152</int>
- <lst name="topTerms">
  <int name="1. The Policy Context 2. The Data Collection 3. Family Solicitors: the
  Workforce and the Work 4. Observing a Dual Profession 5. Solicitor and Client: Support and
  Done
```

Example: Schema Browser

Schema: Indexed, Stored, Omit Norms, Sort Missing Last

Index: Indexed, Stored, Omit Norms

Copied Into: **CATCHALL**

Index Analyzer: org.apache.solr.schema.FieldType\$DefaultAnalyzer

Query Analyzer: org.apache.solr.schema.FieldType\$DefaultAnalyzer

Docs: 814

Distinct: 51

Top 10 Terms

term	frequency
LAW051000	174
LAW052000	85
LAW013000	67
LAW005000	44
LAW021000	32
LAW018000	30
LAW001000	24
LAW026000	22
LAW054000	21
LAW014010	20

Histogram

Number of Terms	Frequency
2	2
4	8
8	16
16	7
32	10
64	1
128	2
256	1

Refining Your Schema

- Pick Field Types That Make Sense
- Pick Analyzers That Make Sense
- Use `<copyField>` To Make Multiple Copies Of Fields For Different Purposes:
 - Faceting
 - Sorting
 - Loose Matching
 - Etc...

Example: "BIC" Codes

```
<!-- used by the bic field, a prefix based code -->
<fieldType name="bicgram" class="solr.TextField" >
  <analyzer type="index">
    <tokenizer class="solr.EdgeNGramTokenizerFactory"
      minGramSize="1"
      maxGramSize="100"
      side="front"          />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.WhitespaceTokenizerFactory" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

But Wait!

There's More!

Search Components

- Default Components That Power SearchHandler
 - QueryComponent
 - HighlightComponent
 - FacetComponent
 - MoreLikeThisComponent
 - DebugComponent
- Additional Components You Can Configure
 - SpellCheckComponent
 - QueryElevationComponent

Score Explanations

- Why Did Document X Score Higher Than Y?
- Why Didn't Document Z Match At All?
- Debugging Options Can Answer Both Questions...
 - idf - How Common A Term Is In The Whole Index
 - tf - How Common A Term Is In This Document
 - fieldNorm - How Significant Is This Field In This Document (Usually Based On Length)
 - boost - How Important The Client Said This Clause Is
 - coordFactor - How Many Clauses Matched

`&debugQuery=true&explainOther=documentId:Z`

Example: Score Explanations

```
<str name="id=9781841135779,internal_docid=111">
```

0.30328625 = (MATCH) fieldWeight(catchall:law in 111),
product of:

3.8729835 = tf(termFreq(catchall:law)=**15**)

1.0023446 = idf(docFreq=851)

0.078125 = fieldNorm(field=catchall, doc=111)

```
</str>
```

...

```
<str name="id=9781841135335,internal_docid=696">
```

0.26578674 = (MATCH) fieldWeight(catchall:law in 696),
product of:

4.2426405 = tf(termFreq(catchall:law)=**18**)

1.0023446 = idf(docFreq=851)

0.0625 = fieldNorm(field=catchall, doc=696)

```
</str>
```

...

DataImporterHandler

Builds and incrementally updates indexes based on configured SQL or XPath queries.

```
<entity name="item" pk="ID" query="select * from ITEM"
  deltaQuery="select ID ... where
    ITEMDATE > '${dataimporter.last_index_time}'">
<field column="NAME" name="name" />
...
<entity name="f" pk="ITEMID"
  query="select DESC from FEATURE where ITEMID='${item.ID}'"
  deltaQuery="select ITEMID from FEATURE where
    UPDATEDATE > '${dataimporter.last_index_time}'"
  parentDeltaQuery="select ID from ITEM where ID=${f.ITEMID}">
<field name="features" column="DESC" />
```

Multiple Indexes

Using a `solr.xml` file, you can configure Solr to manage several different indexes.

```
<solr persistent="true" sharedLib="lib">  
  <cores adminPath="/core-admin/">  
    <core name="books" instanceDir="books" />  
    <core name="games" instanceDir="games" />  
    ...
```

The `CoreAdminHandler` let's you create, reload and swap indexes on the fly.

```
/core-admin?action=RELOAD&core=books
```

```
/core-admin?action=CREATE&name=books2&instanceDir=books2
```

```
/core-admin?action=SWAP&core=books&other=books2
```


Replication

Replication scripts for efficiently mirroring an index on multiple machines.

- `snapshotter`
- `snappuller`
- `snapinstaller`
- Oh My!

Distributed Searching

- Options For Aggregating Results From Multiple Solr “Shards”
- Handy When “Index” Is Too Big For One Machine
- Most Core Features Supported:
 - Basic Queries
 - Highlighting
 - Faceting

?shards=host1:8983/solr,host2:7574/solr&q=ipod

Questions?

<http://lucene.apache.org/solr/>

Your  Logo Here