Out Of The Box

Chris Hostetter

2010-11-09

`http://people.apache.org/~hossman/apachecon2011/`
`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
- Use Case: Starting from Scratch
- But Wait! There's More!

What Is Solr?

Elevator Pitch

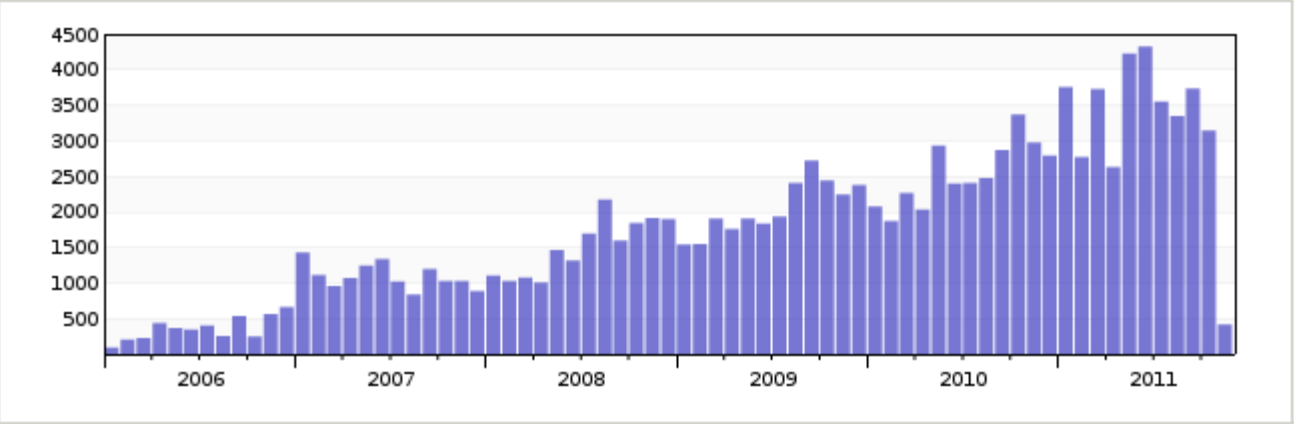
"Solr is the highly scalable open source enterprise search platform from the Apache Lucene project. It supports faceting, highlighting, goruping, distributed search and index replication; and it powers search and navigation on some of the worlds largest websites."

What Does That Mean?

- Information Retrieval Application
- Java WebApp (WAR) With A Web Services-ish API
- Released By The Apache Lucene Project
- Extremely Healthy Developer And User Communities

Searching **3 lists** and **126,281 messages**. First list started in **January 2006**. There are **3 active lists**, recently accumulating **119 messages per day**. You can browse recent emails.

Traffic (messages per month):



Solr In A Nutshell

- Index/Query Via HTTP
- Comprehensive HTML Administration Interfaces
- Scalability - Horizontal and Vertical
- 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`

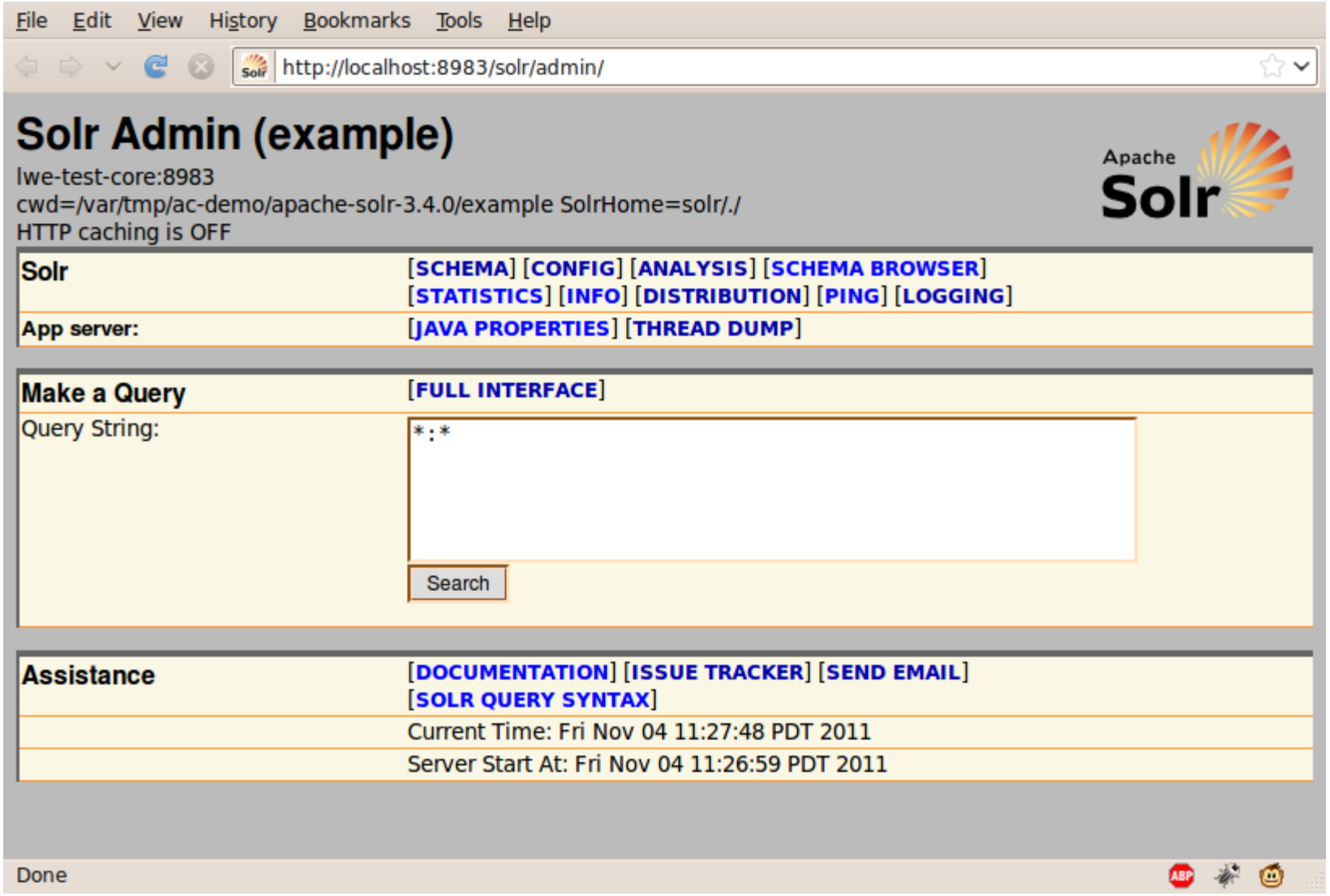
- OOTB Quick Tour Of Solr Basics Using Jetty
- Comes With Example Config, Schema, And Data
- Trivial To Follow Along...

```
cd example
java -jar start.jar

        http://localhost:8983/solr/

cd example/exampledocs
java -jar post.jar *.xml
```

The Admin Console



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...

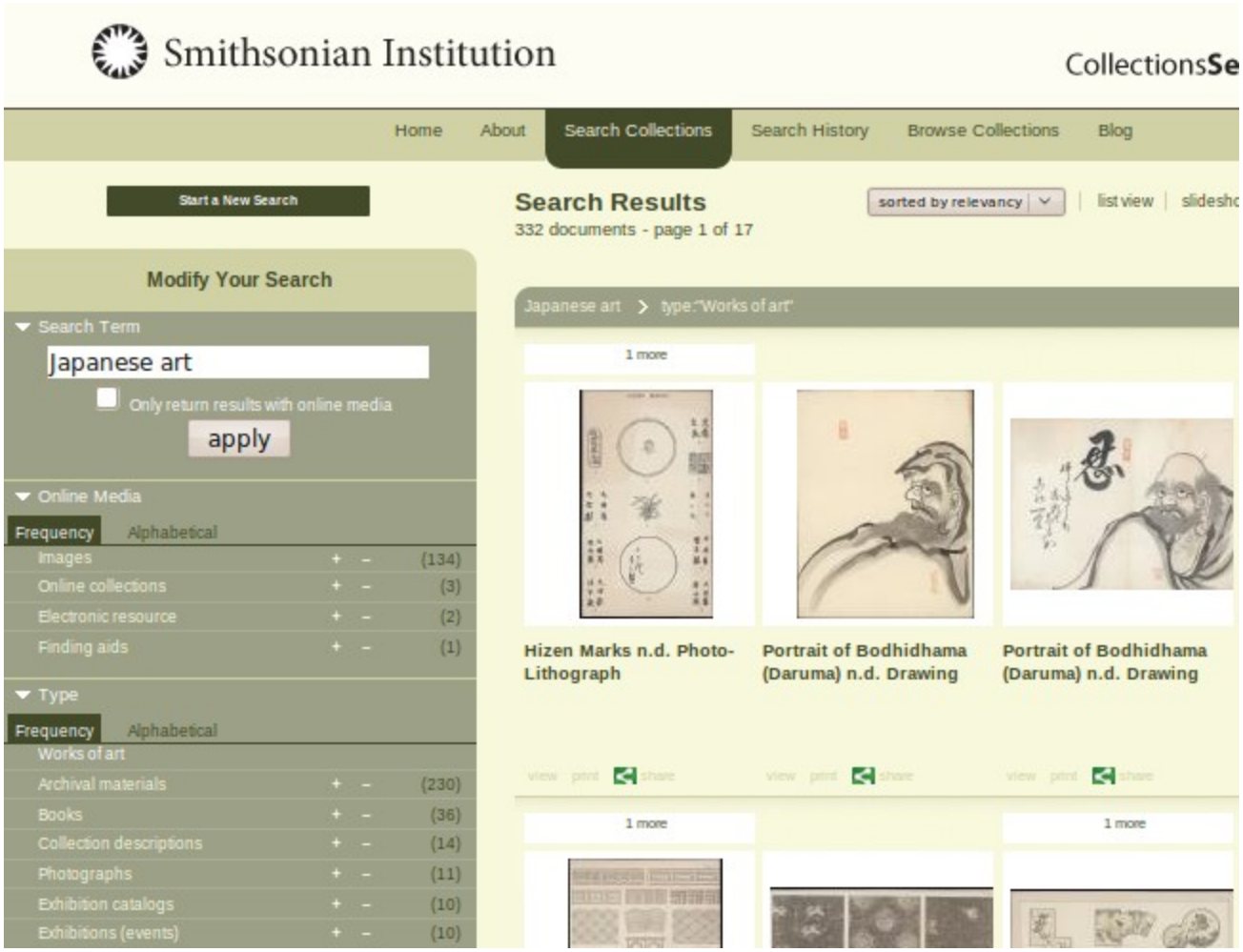
```
http://solr/select?q=electronics
http://solr/select?q=electronics&sort=price+desc
http://solr/select?q=electronics&rows=50&start=50
http://solr/select?q=electronics&fl=name+price
http://solr/select?q=electronics&fq=inStock:true
```

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>
      <bool name=”inStock”>true</bool>
      ...
```

Querying Data: Facet Counts



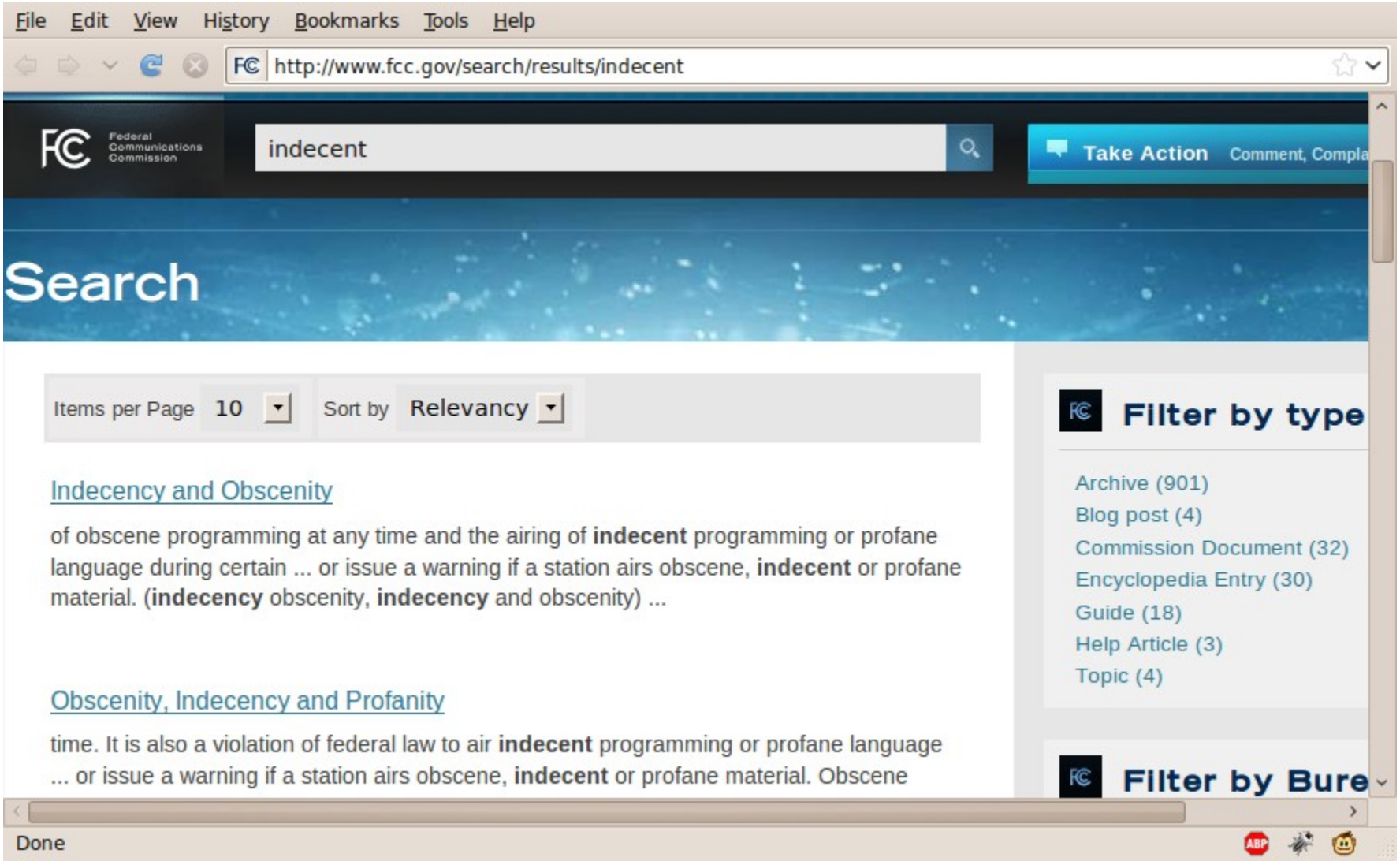
Querying Data: Facet Counts

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>
    ...
```

Querying Data: Highlighting



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&gt;adapter&lt;/em&gt;, 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="body"   type="text"   stored=”false” />
<dynamicField name="price*" type="float"  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 For Parsing If It Is A Text Field
- OOTB Solr Has Dozens of FieldType Classes

```
<fieldType name="float" class="solr.TrieFloatField"
           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 Dozens Of 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!

Configurable Analyzers

- Solr Lets You Mix And Match CharFilters, Tokenizers And TokenFilters In Your `schema.xml` To Define Analyzers On The Fly
  - CharFilter: Mutates And Manipulates The Stream Of Characters
  - Tokenizer: Splits The Characters Into Tokens
  - TokenFilter: Mutates And Manipulates The Stream Of Tokens
- OOTB Solr Has Factories For Several CharFilters, Dozens Of Tokenizers And Nearly 100 TokenFilters
- Many Factories Have Customization Options -- Limitless Combinations

Configurable Analyzers

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

Notable Analysis Factories

CharFilterFactory

- HTMLStripCharFilterFactory
- MappingCharFilterFactory
- PatterReplaceCharFilterFactory

TokenizerFactory

- StandardTokenizerFactory
- WhitespaceTokenizerFactory
- KeywordTokenizerFactory
- NGramTokenizerFactory
- PatternTokenizerFactory
- PathHierarchyTokenizerFactory

TokenFilterFactory

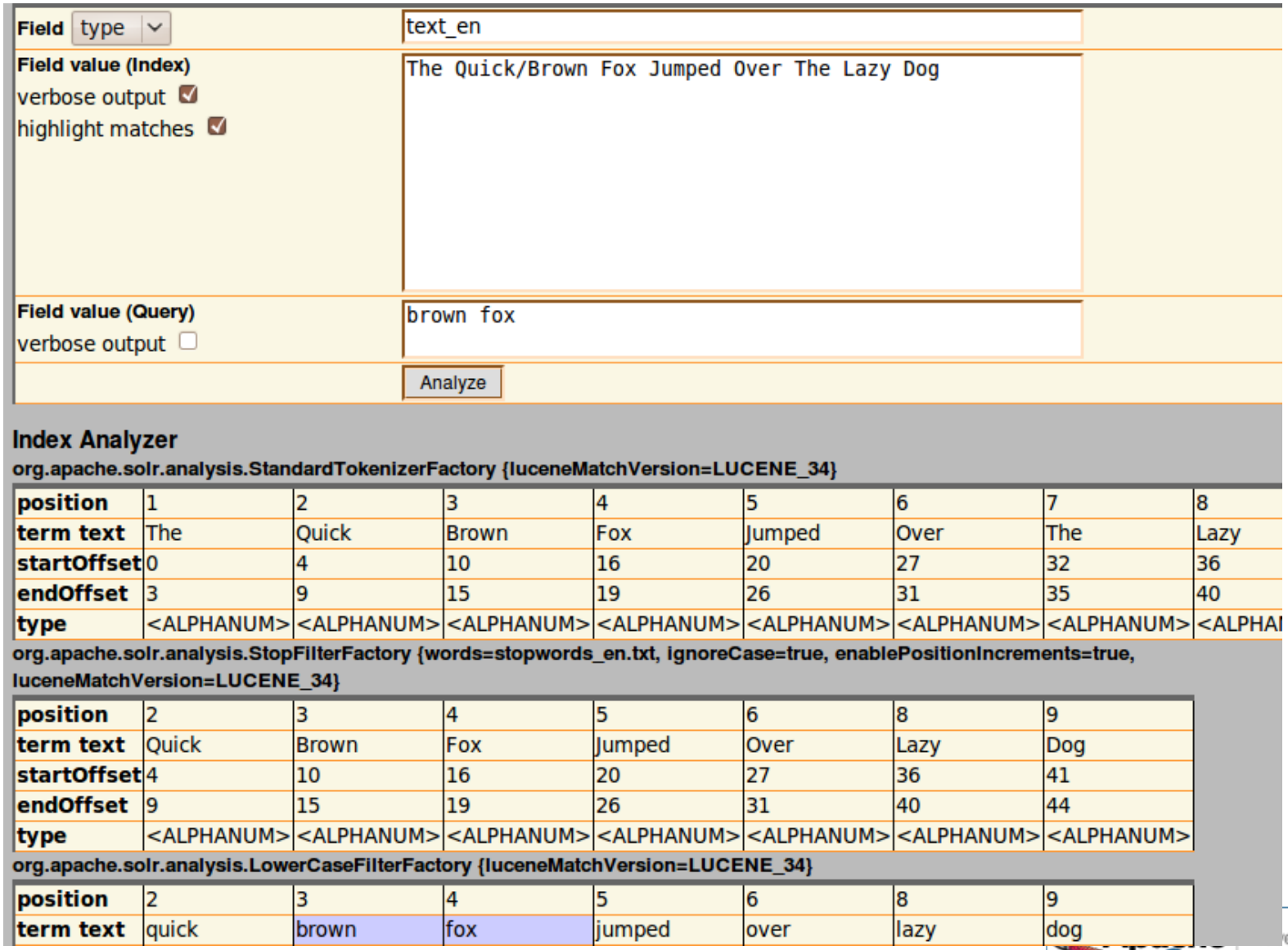
- SynonymFilterFactory
- StopFilterFactory
- ReversedWildcardFilterFactory
- CommonGramsFilterFactory
- ICUCollactionKeyFilterFactory
- PatternReplaceFilterFactory
- LengthFilterFactory

...and loads of Stemmers for various languages.

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...
  - Char Stream Produced By The CharFilter
  - 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 How to Configure Analyzer Factories For Each Field Based On Your Goals

Analysis Tool: Output



Field: type — text_en

Field value (Index): The Quick/Brown Fox Jumped Over The Lazy Dog

verbose output ☑

highlight matches ☑

Field value (Query): brown fox

verbose output ☐

Analyze

Index Analyzer

org.apache.solr.analysis.StandardTokenizerFactory {luceneMatchVersion=LUCENE_34}

position	1	2	3	4	5	6	7	8
term text	The	Quick	Brown	Fox	Jumped	Over	The	Lazy
startOffset	0	4	10	16	20	27	32	36
endOffset	3	9	15	19	26	31	35	40
type	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHA

org.apache.solr.analysis.StopFilterFactory {words=stopwords_en.txt, ignoreCase=true, enablePositionIncrements=true, luceneMatchVersion=LUCENE_34}

position	2	3	4	5	6	8	9
term text	Quick	Brown	Fox	Jumped	Over	Lazy	Dog
startOffset	4	10	16	20	27	36	41
endOffset	9	15	19	26	31	40	44
type	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>

org.apache.solr.analysis.LowerCaseFilterFactory {luceneMatchVersion=LUCENE_34}

position	2	3	4	5	6	8	9
term text	quick	brown	fox	jumped	over	lazy	dog

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
    - XsltUpdateRequestHandler
  - JsonUpdateRequestHandler
  - CSVRequestHandler
  - BinaryUpdateRequestHandler
  - DataImportHandler
  - ExtractingRequestHandler
- OOTB Searching Handler:
  - SearchHandler + SearchComponents + QParsers

SearchHandler

- SearchHandler Executes Query With Filtering, Pagination, Return Field List, Highlighting, Faceting, Etc...
- Uses `defType` To Pick QParser For Query String `q`
- OOTB Solr Provides A Dozen QParsers You Can Use Depending On Your Needs
- Two Key QParsers To Know About:

```
&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+TO+*]
?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+TO+*]
```

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.

Other QParsers

- By Default All Query Params Use LuceneQParser
- “LocalParams” Prefix Notation Exists To Override This, And Customize Behavior
- Even Supports Parameter Dereferencing

```
?q={!boost b=div(popularity,price) v=$qq}
&qq={!dismax qf=desc^2,review}cheap
&bq={!lucene df=keywords}lucene solr java
&fq={!geofilt sfield=location pt=10.312,-20.556 d=3.5}
&fq={!term f=$ff v=$vv}&ff=keywords&vv=solr
&sort=query(keywords:lame) asc, score desc
```

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">
    <bool name="facet">false</bool>
    ...
```

Output: Response Writers

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

```
http://solr/select?q=ipod&wt=xml
http://solr/select?q=ipod&wt=json
http://solr/select?q=ipod&wt=python
http://solr/select?q=ipod&wt=ruby
http://solr/select?q=ipod&wt=php
http://solr/select?q=ipod&wt=csv
http://solr/select?q=ipod&wt=xslt&tr=example.xsl
http://solr/select?q=ipod&wt=velocity&v.template=browse
```

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

Use Case

Starting 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"
         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 neccessary
    <uniqueKey>id</uniqueKey>
    <defaultSearchField>catchall</defaultSearchField>
  -->
</schema>
```

Feeding Data From The Wild

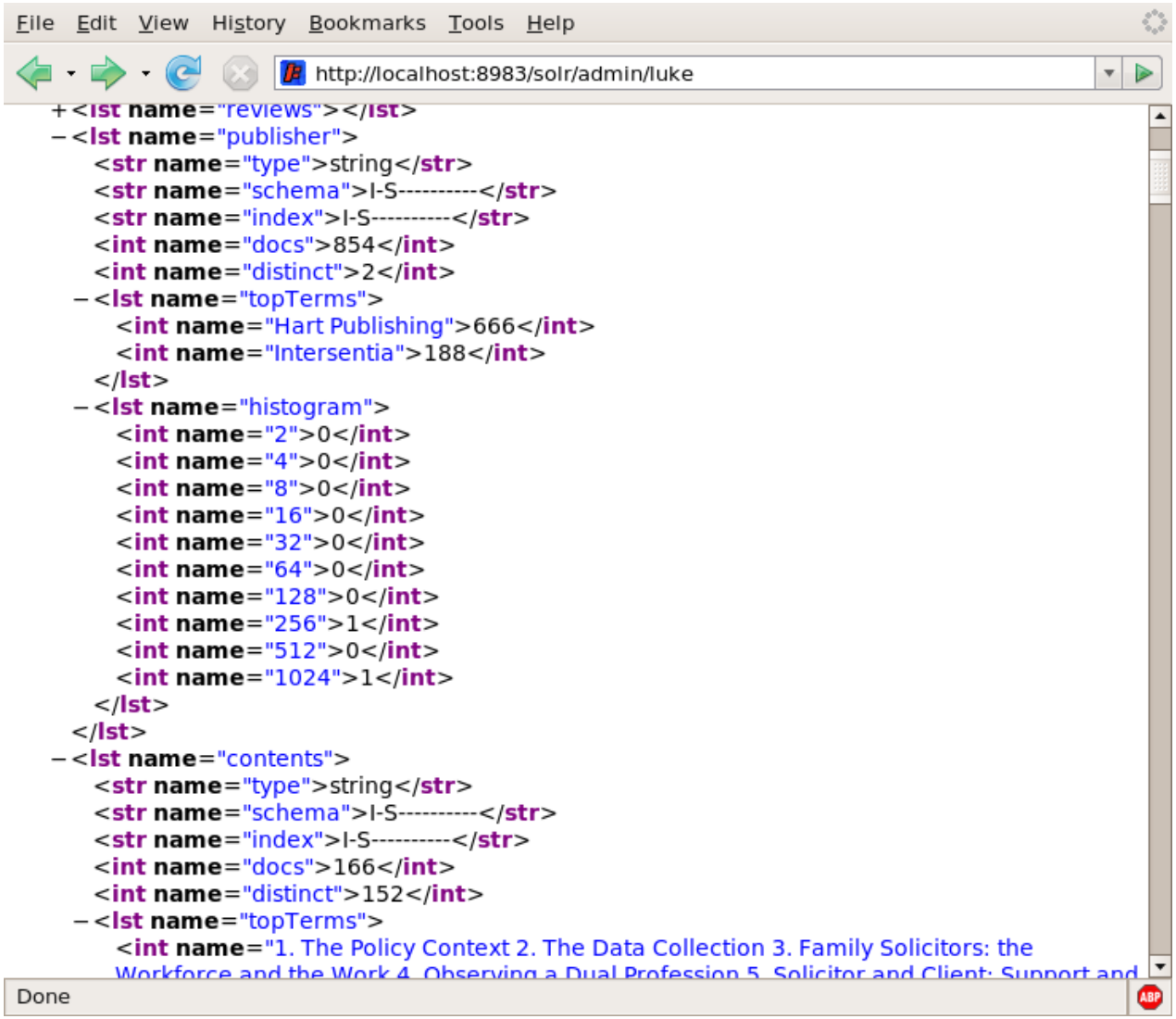
- I Went Online And Found A CSV File Containing Data On Books
- Deleted Some Non UTF-8 Characters
- Made Life Easier For Myself By Renaming The Columns So They Didn't Have Spaces

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

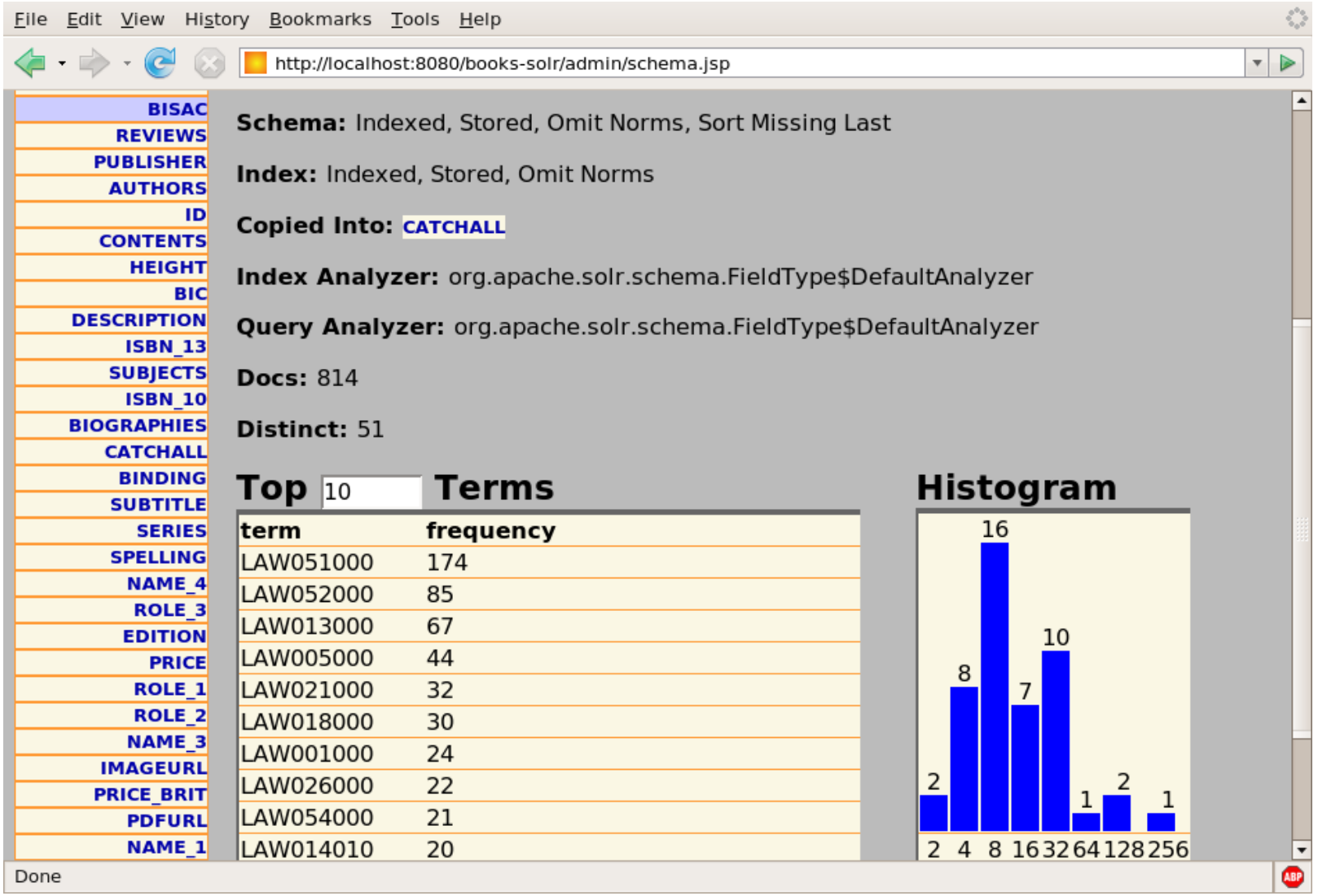
Understanding The Data: Luke

- 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



Example: Schema Browser



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!

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

ExtractingRequestHandler

- Aka: “Solr Cell”
- Uses Tika to Parse Binary & Rich Content Documents
  - HTML
  - PDF
  - MS-Word
  - MP3
- Maps Tika Output Fields To Solr Schema Fields
- Supports XPath Filtering Of The Generated DOM

DataImportHandler

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" />
  ...
```

Update Processor Chains

- Configurable Pipelines For Updates
- Abstracts Logic Used Regardless Of Format
- OOTB Support For Computing Signatures & Running UIMA Analysis Engines

```
<updateRequestProcessorChain name="dedupe">
  <processor class="solr.SignatureUpdateProcessorFactory">
    <str name="signatureField">id</str>
    <str name="fields">name,features,cat</str>
  </processor>
  <processor class="solr.RunUpdateProcessorFactory" />
</updateRequestProcessorChain>
```

Search Components

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

Score Explanations

- Why Did Document X Score Higher Then 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>
...
```

Grouping / Field Collapsing

- Group Matching Documents By Common Field Or Arbitrary Query
- Limit Number of Documents Returned In Each Group
- Sort Groups Independently Of Documents In Groups

```
&group=true
&group.field=category
&group.limit=5
&group.sort=popularity desc
&sort=score 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

Use ReplicationHandler to efficiently mirror an index on multiple machines (ie: Scale Horizontally)

```
<requestHandler name="/replication"
                class="solr.ReplicationHandler">
  <lst name="master">
      <str name="replicateAfter">commit</str>
  </lst>
</requestHandler>
...
<requestHandler name="/replication"
                class="solr.ReplicationHandler">
  <lst name="slave">
    <str name="masterUrl">
       http://master:8080/solr/replication
    </str>
    <str name="pollInterval">00:00:60</str>
```

Distributed Searching

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

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

Questions?

http://people.apache.org/~hossman/

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