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!

ApacheCon US 2008

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."

ApacheCon US 2008

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

ApacheCon US 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

ApacheCon US 2008

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...

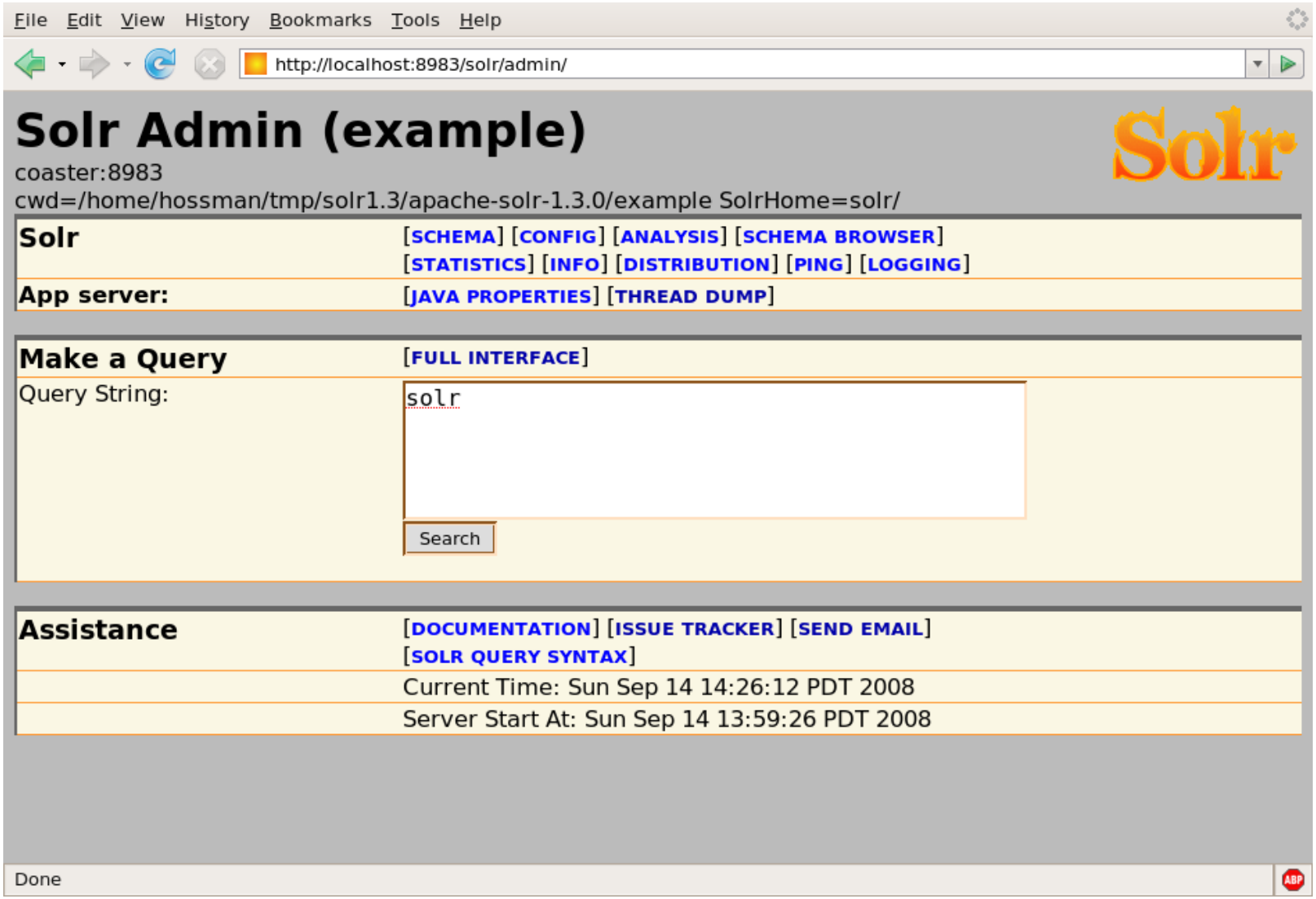
```
cd example
java -jar start.jar

http://localhost:8983/solr/

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


ApacheCon US 2008

The Admin Console



ApacheCon US 2008

Configuration

schema.xml

- Where You Describe Your Data

solrconfig.xml

- Where You Describe How People Can Interact With Your Data

ApacheCon US 2008

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>
```


ApacheCon US 2008

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
```


ApacheCon US 2008

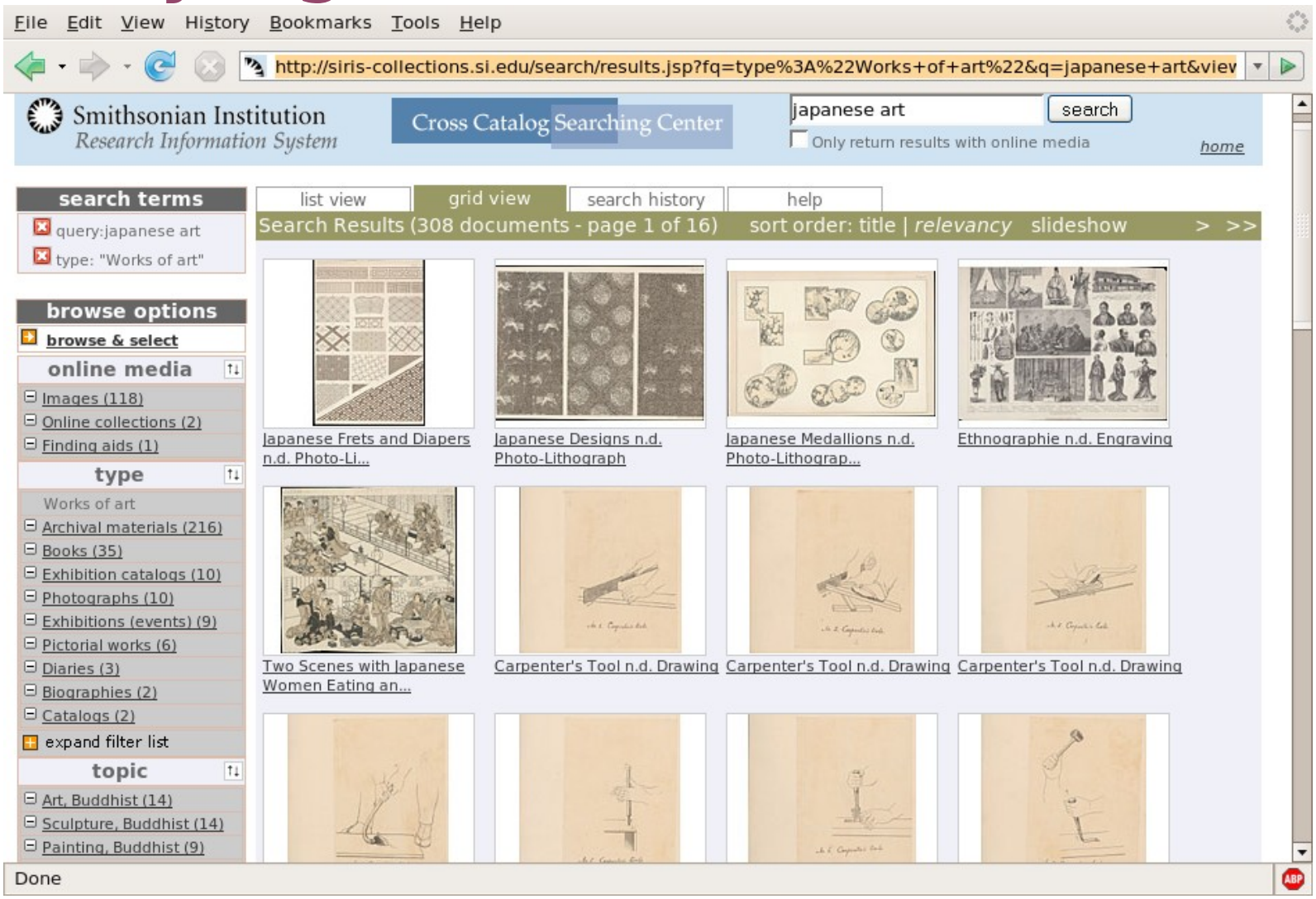
Querying Data: Results

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>
      ...
```


ApacheCon US 2008

Querying Data: Facet Counts



ApacheCon US 2008

Querying Data: Facet Counts

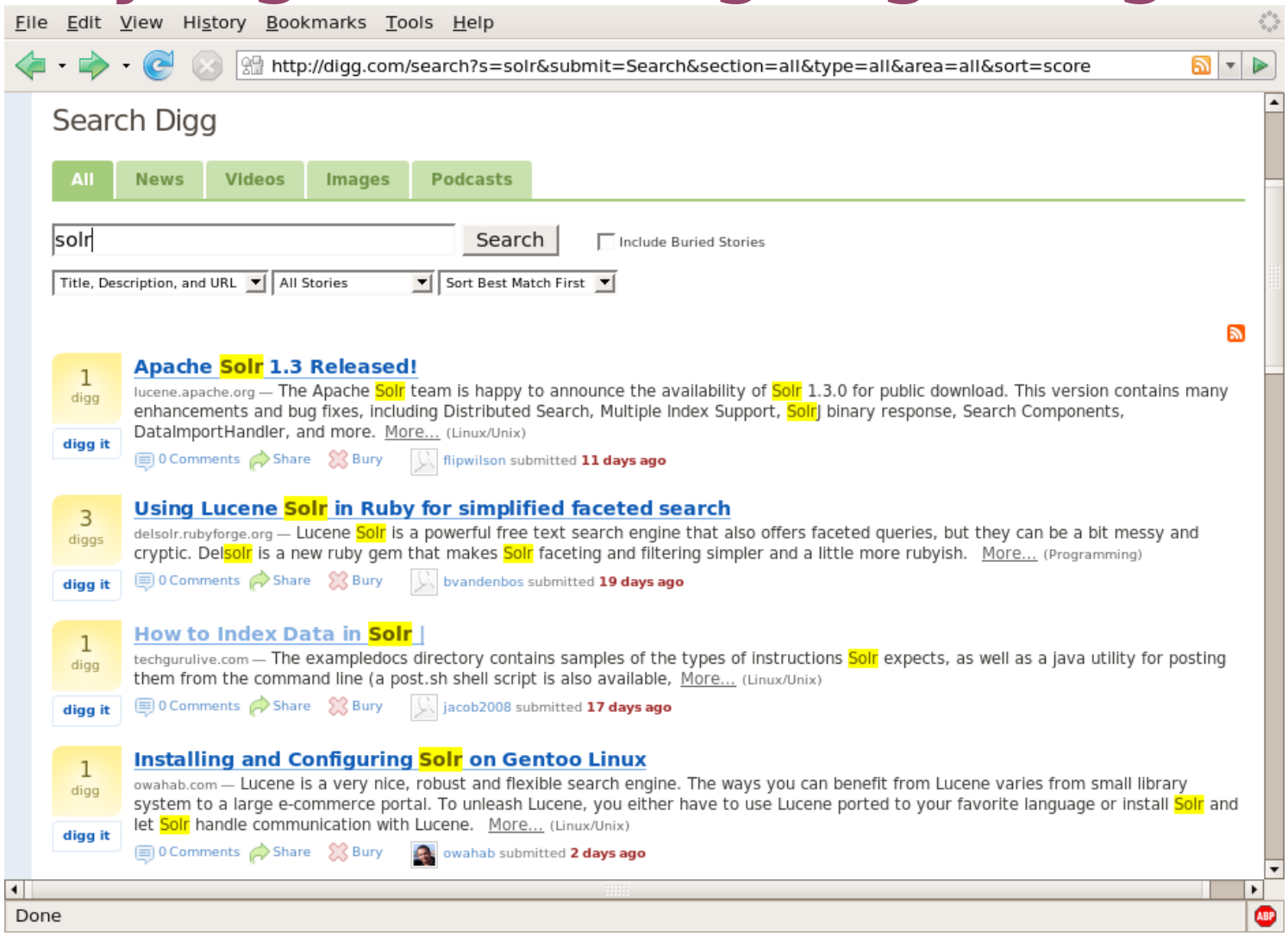
Constraint counts can be computed for the whole result set using field values or explicit queries....

```
  &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>
    ...
```


ApacheCon US 2008

Querying Data: Highlighting



ApacheCon US 2008

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>
  ...
```


ApacheCon US 2008

Digging Deeper

`schema.xml`

ApacheCon US 2008

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?

ApacheCon US 2008

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" />
```


ApacheCon US 2008

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" />
```


ApacheCon US 2008

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!

ApacheCon US 2008

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

ApacheCon US 2008

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"/>
   ...
```


ApacheCon US 2008

Notable Token(izers|Filters)

- StandardTokenizerFactory
- HTMLStripWhitespaceTokenizerFactory
- KeywordTokenizerFactory
- NGramTokenizerFactory
- PatternTokenizerFactory

- EnglishPorterFilterFactory
- SynonymFilterFactory
- StopFilterFactory
- ISOLatin1AccentFilterFactory
- PatternReplaceFilterFactory

ApacheCon US 2008

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

ApacheCon US 2008

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 ☑

The Quick/Brown Fox Jumped Over The Lazy Dog

Field value (Query)
verbose output ☐

brown fox?

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

ApacheCon US 2008

Digging Deeper

`solrconfig.xml`

ApacheCon US 2008

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`.

ApacheCon US 2008

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

ApacheCon US 2008

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

ApacheCon US 2008

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
```


ApacheCon US 2008

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*
```


ApacheCon US 2008

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.

ApacheCon US 2008

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+*]
```


ApacheCon US 2008

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.

ApacheCon US 2008

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
```


ApacheCon US 2008

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
```


ApacheCon US 2008

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>
    ...
```


ApacheCon US 2008

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"/>
```


ApacheCon US 2008

Trial By Fire

Using Solr From Scratch

ApacheCon US 2008

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.)

ApacheCon US 2008

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>
```


ApacheCon US 2008

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>
```


ApacheCon US 2008

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
```

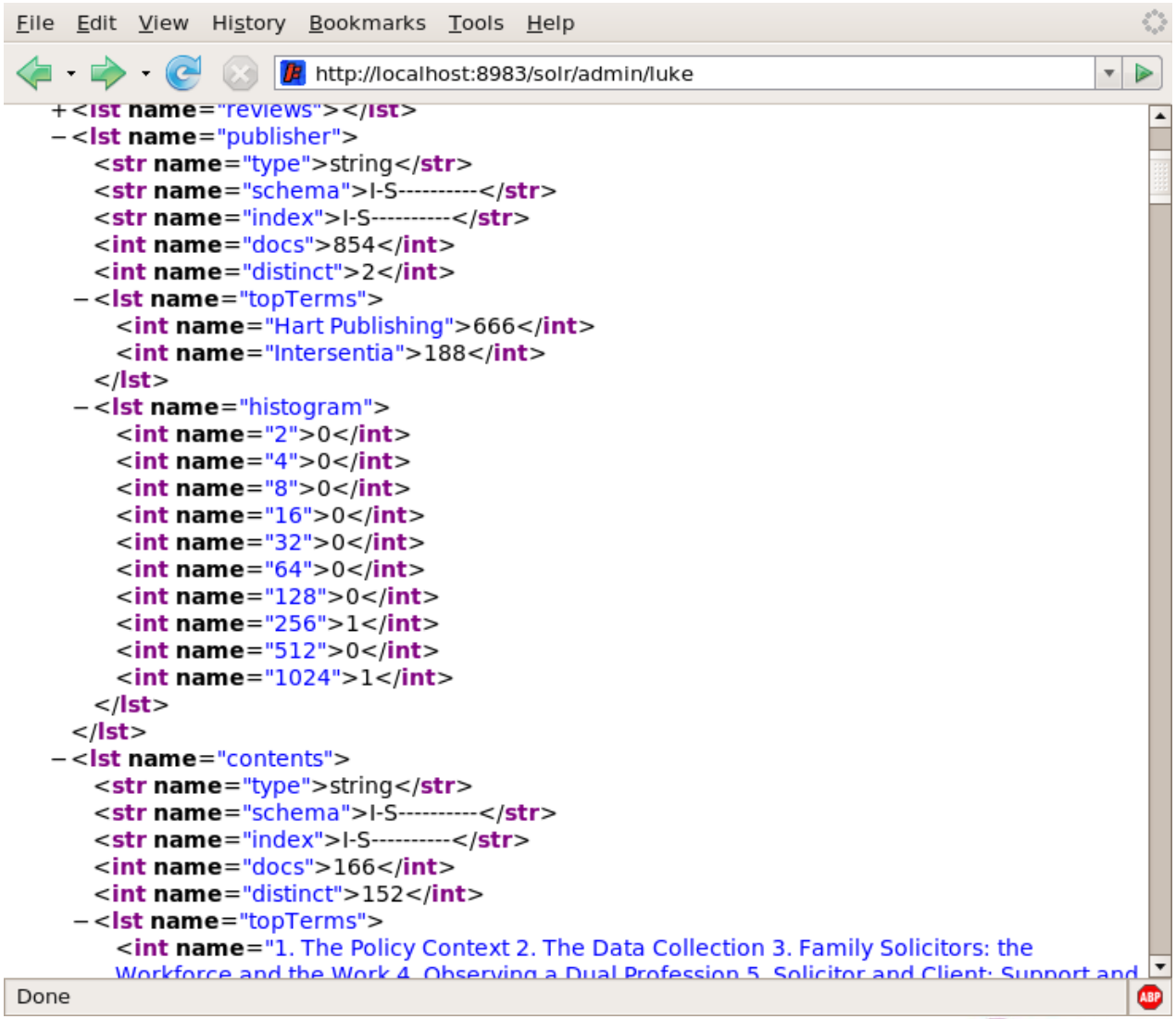

ApacheCon US 2008

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

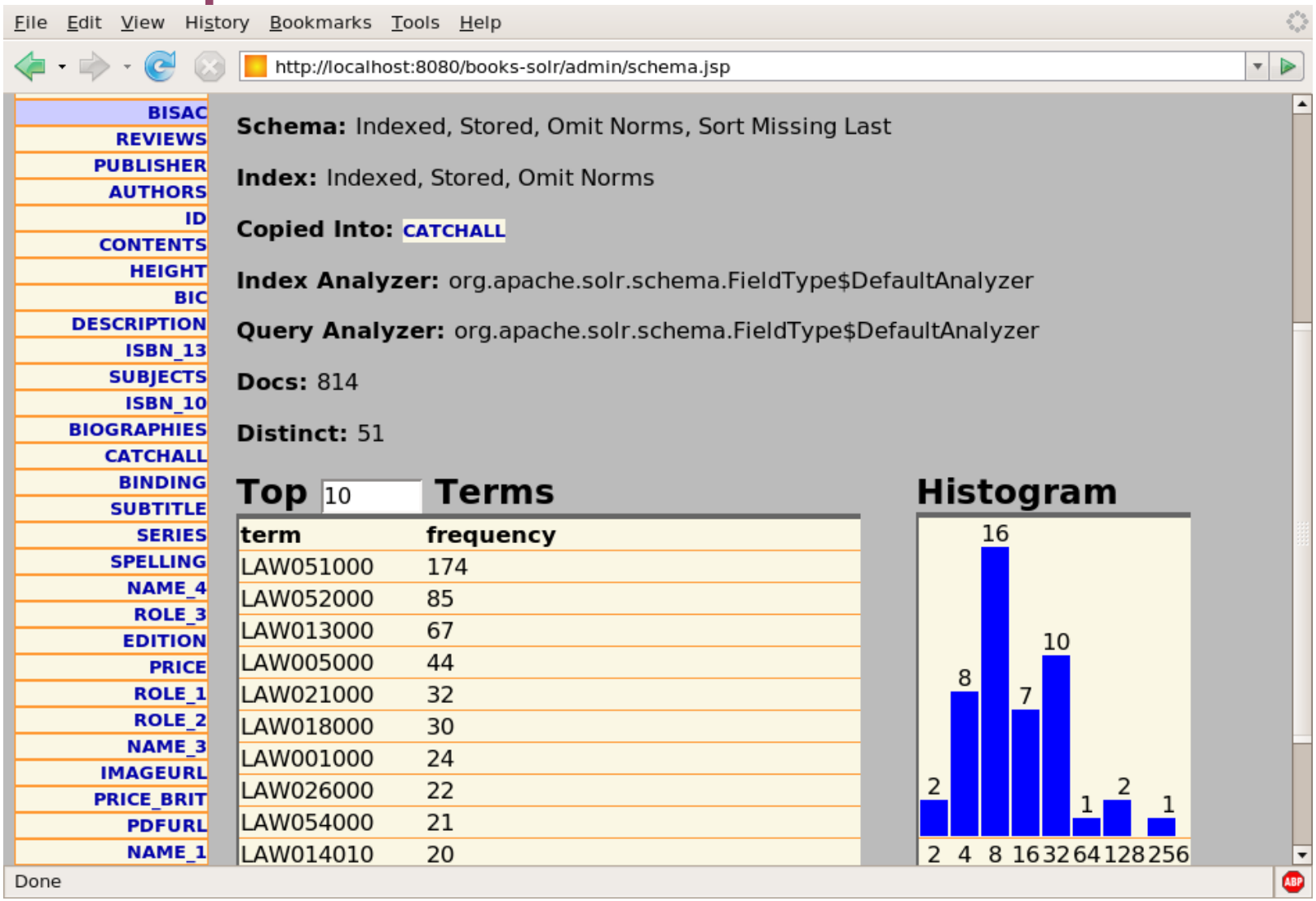
ApacheCon US 2008

Example: Luke Output



ApacheCon US 2008

Example: Schema Browser



ApacheCon US 2008

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...

ApacheCon US 2008

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>
```


ApacheCon US 2008

But Wait!

There's More!

Search Components

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

ApacheCon US 2008

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
```


ApacheCon US 2008

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>
...
```


ApacheCon US 2008

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" />
  ...
```


ApacheCon US 2008

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
```


ApacheCon US 2008

Replication

Replication scripts for efficiently mirroring an index on multiple machines.

- `snapshooter`
- `snappuller`
- `snapinstaller`
- Oh My!

ApacheCon US 2008

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
```


ApacheCon US 2008

Questions?

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

ApacheCon US 2008