

Apache  
**Solr**  
Out Of The Box



Chris Hostetter

2009-11-05

<http://people.apache.org/~hossman/apachecon2009us/>

<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

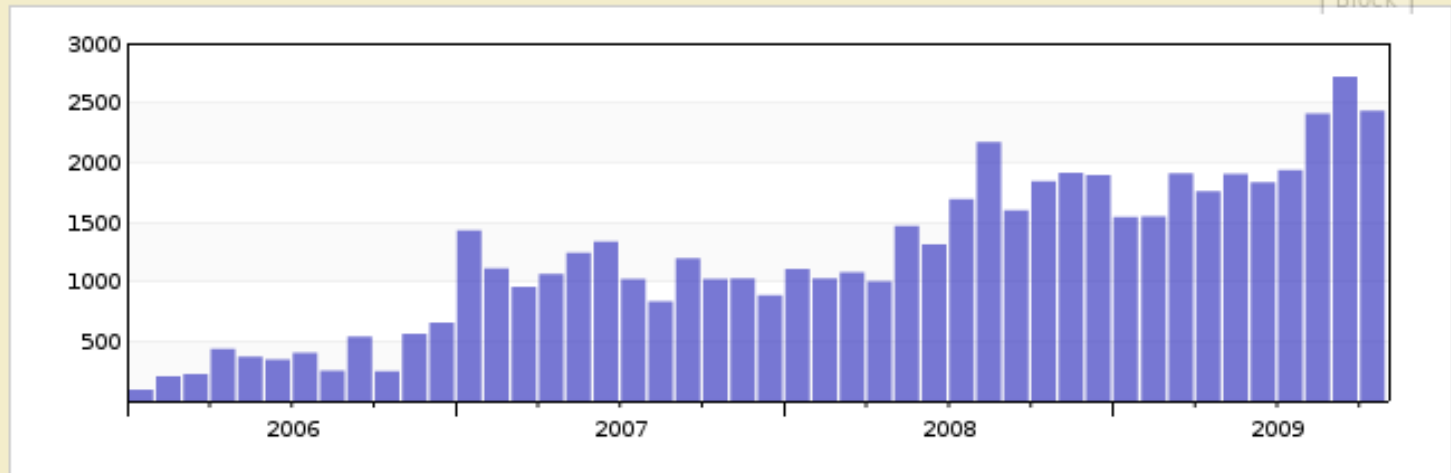
"Solr is an open source enterprise search server based on the Lucene Java search library, with XML/HTTP APIs, caching, replication, and a web administration interface."

# What Does That Mean?

- Information Retrieval Application
- Java5 WebApp (WAR) With A Web Services-ish API
- Uses The Java Lucene Search Library
- Healthy And Growing Lucene Sub-Project

Searching **3 lists** and **55,574 messages**. First list started in **January 2006**. There are **3 active lists**, recently accumulating **56 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

The screenshot shows a web browser window with the address bar containing `http://localhost:8983/solr/admin/`. The page title is "Solr Admin (example)". Below the title, it displays "coaster:8983" and "cwd=/var/tmp/ac-demo/apache-solr-1.4/example SolrHome=solr/". The Apache Solr logo is in the top right corner.

The main content area is divided into several sections:

- Solr**: Includes links for [SCHEMA], [CONFIG], [ANALYSIS], [SCHEMA BROWSER], [STATISTICS], [INFO], [DISTRIBUTION], [PING], and [LOGGING].
- App server:**: Includes links for [JAVA PROPERTIES] and [THREAD DUMP].
- Make a Query**: Features a "Query String:" label, a text input field containing "solr", and a "Search" button.
- Assistance**: Includes links for [DOCUMENTATION], [ISSUE TRACKER], [SEND EMAIL], and [SOLR QUERY SYNTAX].

At the bottom of the assistance section, the following information is displayed:

- Current Time: Sun Oct 25 22:02:06 PST 2009
- Server Start At: Sun Oct 25 21:45:30 PST 2009

The browser's status bar at the bottom left shows "Done", and the bottom right corner has a small "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 Cross Catalog Searching Center. The search results are displayed in a grid view, showing 308 documents on page 1 of 16. The search terms are 'query:japanese art' and 'type: "Works of art"'. The search results are sorted by title, relevancy, or slideshow. The results are categorized by type and topic.


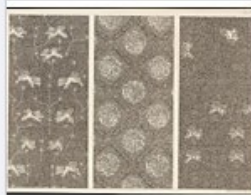






**Search Terms:**  
query:japanese art  
type: "Works of art"


**Browse Options:**  
browse & select  
online media (118)  
Images (118)  
Online collections (2)  
Finding aids (1)  
type (118)  
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)  
expand filter list  
topic (118)  
Art, Buddhist (14)  
Sculpture, Buddhist (14)  
Painting, Buddhist (9)

**Search Results (308 documents - page 1 of 16)**  
sort order: title | relevancy | slideshow > >>

**list view** **grid view** search history help

**Results:**

-   
Japanese Frets and Diapers n.d. Photo-Litho...
-   
Japanese Designs n.d. Photo-Lithograph
-   
Japanese Medallions n.d. Photo-Lithograp...
-   
Ethnographie n.d. Engraving
-   
Two Scenes with Japanese Women Eating an...
-   
Carpenter's Tool n.d. Drawing
-   
Carpenter's Tool n.d. Drawing
-   
Carpenter's Tool n.d. Drawing

Done 

# 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

File Edit View History Bookmarks Tools Help

http://www.whitehouse.gov/search/site/solar%20power?filters=created%3A%5B2009-10-01T

WHITE HOUSE PRESIDENT BARACK OBAMA THE WHITE HOUSE WASHINGTON

BLOG PHOTOS & VIDEO BRIEFING ROOM ISSUES the ADMINISTRATION the WHITE HOUSE

Home • Search • solar power • October 2009

Search WhiteHouse.gov

## Search: solar power

Q solar power Search  Retain current filters

## Search results

Sort by: Relevancy Title Type Author Date

**A Competition Powered by the Sun**  
Posted on October 08, 2009 at 03:45 PM EDT

I just helped kick-off the Department of Energy's 2009 **Solar** Decathlon. This is a unique student competition on the National Mall that showcases the latest energy efficiency and **solar power** ... planning, designing, and building high-efficiency, **solar-powered** homes. I've already met several of them ...

**CURRENT SEARCH**  
Search found 9 items  
solar power  
(-) **October 2009**

**FILTER BY POST DATE**  
(-) **October 2009**  
October 2, 2009 (1)  
October 8, 2009 (2)  
October 12, 2009 (1)  
October 15, 2009 (1)  
October 22, 2009 (1)  
October 23, 2009 (3)

**FILTER BY TYPE**  
Blog Post (4)  
Press Article (4)

Done

# 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 adapter, 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 25 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 17 Tokenizers and 45 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"/>
```

...

# Notable Token(izers|Filters)

- StandardTokenizerFactory
- WhitespaceTokenizerFactory
- KeywordTokenizerFactory
- NGramTokenizerFactory
- PatternTokenizerFactory
  
- EnglishPorterFilterFactory
- SynonymFilterFactory
- StopFilterFactory
- ASCIIFilterFactory
- 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

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+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.

# 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" />
```

# 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"  
  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

The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/books-solr/admin/schema.jsp`. The browser's menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. The main content area is divided into several sections:

- Field List:** A vertical list of fields on the left side, including BISAC, REVIEWS, PUBLISHER, AUTHORS, ID, CONTENTS, HEIGHT, BIC, DESCRIPTION, ISBN\_13, SUBJECTS, ISBN\_10, BIOGRAPHIES, CATCHALL, BINDING, SUBTITLE, SERIES, SPELLING, NAME\_4, ROLE\_3, EDITION, PRICE, ROLE\_1, ROLE\_2, NAME\_3, IMAGEURL, PRICE\_BRIT, PDFURL, and NAME\_1.
- Schema Information:**
  - 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:** A table showing the most frequent terms in the index.
- Histogram:** A bar chart showing the frequency distribution of terms.

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

Frequency	Number of Terms
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!

# 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" />
...
```

# 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

# 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 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>
```

```
...
```

# 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

## ReplicationHandler for efficiently mirroring an index on multiple machines

```
<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>
  </lst>
  ...
</requestHandler>
```

# 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/`