# Apache Solr
# Out Of The Box (OOTB)

Chris Hostetter

hossman - apache - org

2007-11-16

`http://people.apache.org/~hossman/apachecon2007us/`

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

# Why Are We Here?

- Learn What Solr Is
- Opening the Box
- 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
- Lucene Sub-Project Since January 2007

# 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

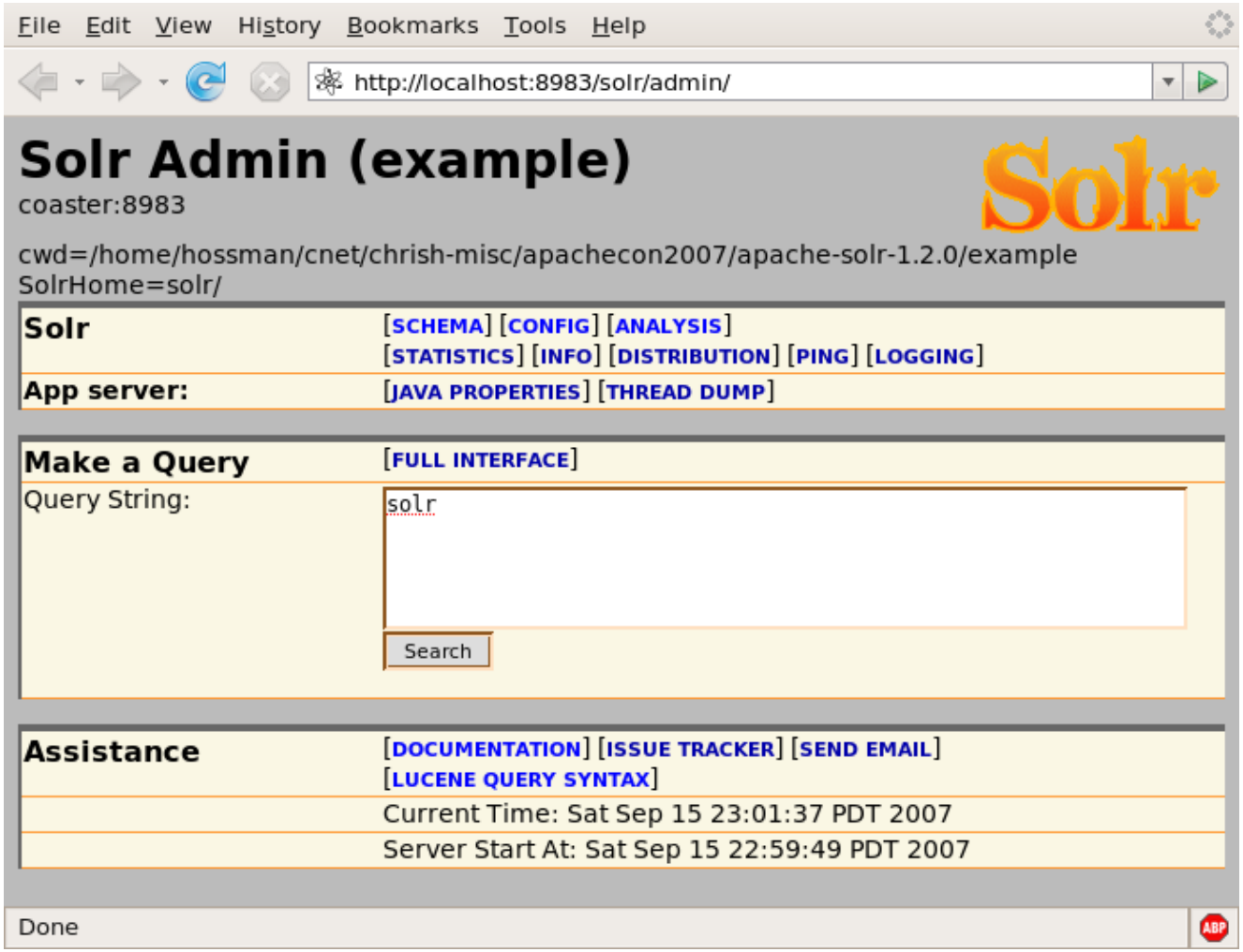
# The Solr Tutorial

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
```

# The Admin Console



# 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: Response

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>
      ...
```

# Querying Data: Facet Counts

Constraint counts can be computed for any 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

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

# 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 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 (1.2) Has 15 FieldType Classes

# Analyzers

- 'Analyzer' Is A Core Lucene Class For Parsing Text
- Solr (1.2) Includes 18 Lucene Analyzers That Can Be Used OOTB If They Meet Your Needs

...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 (1.2) Has Factories For 9 Tokenizers and 15 TokenFilters
- Many Factories Have Customization Options -- Limitless Combinations

# Notable Token(izers|Filters)

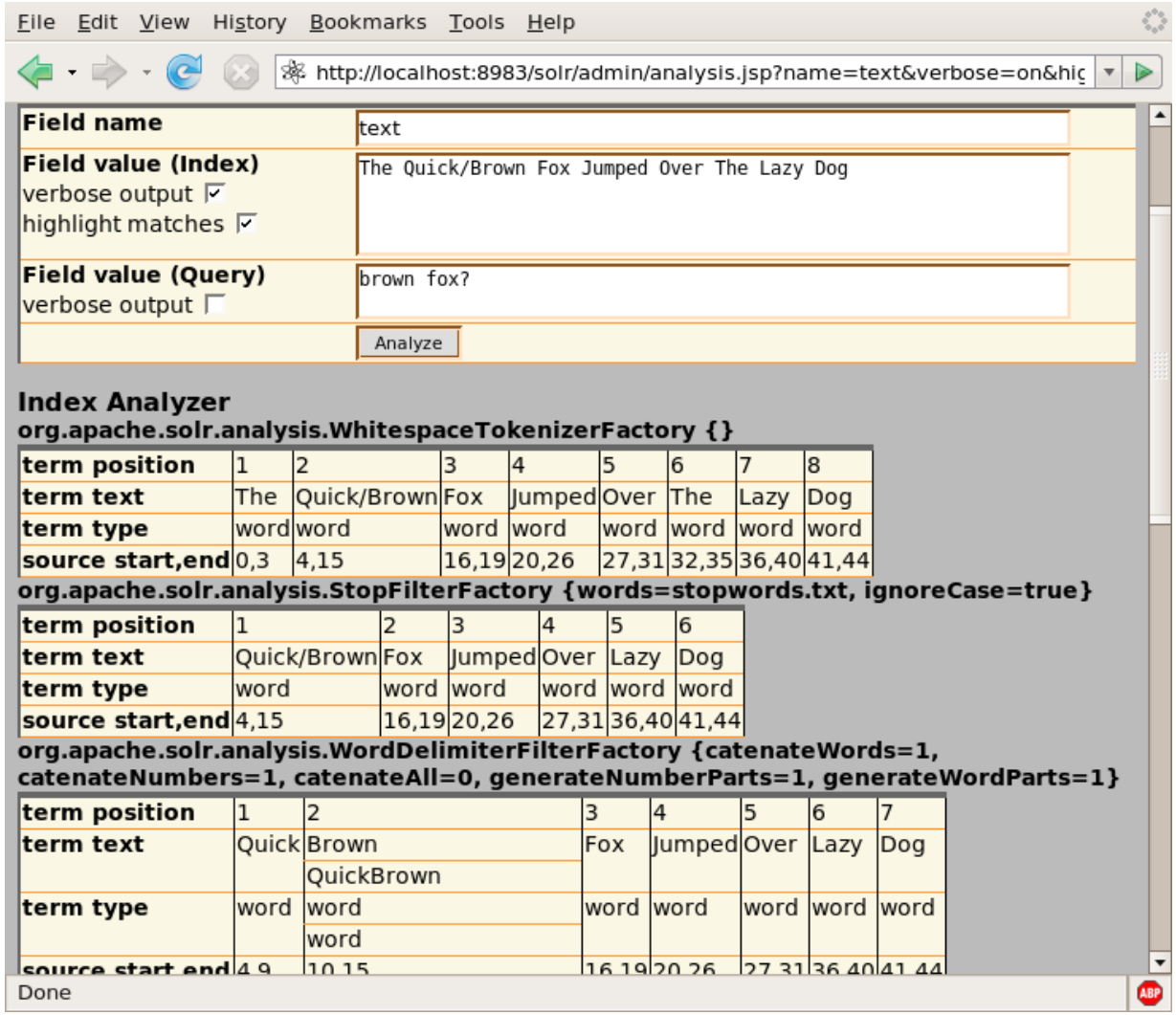
- StandardTokenizerFactory
- HTMLStripWhitespaceTokenizerFactory
- KeywordTokenizerFactory
- NGramTokenizerFactory
- PatternTokenizerFactory (1.3)

- 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



# Interacting With Your Data

# Query Logic: Request Handlers

- Request Handler Type Determines Options, Syntax, And Logic For Processing Requests (Searches And Updates)
- OOTB Solr Provides Two Great Request Handlers For Searching That You Can Use Depending On Your Needs
- Both Support Common Options For Controlling Pagination, Return Field List, Highlighting, Faceting, Etc...

# StandardRequestHandler

- 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*
```

# StandardRequestHandler

```
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.

# DisMaxRequestHandler

- 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+*]
```

# DisMaxRequestHandler

```
   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.

# Output: Response Writers

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

```
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=xslt&tr=example.xsl
```

# Indexing: Request Handlers

- They Aren't Just For Searching!
- Since Solr 1.2, Data Updating Is Also Controlled By "Request Handlers"
- In Addition To An XmlUpdateRequestHandler For Dealing With The Update Message Format, There Is Also A CSVRequestHandler OOTB

# 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

# 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.Standard..." />
<requestHandler name="/simple" class="solr.DisMax..." >
  <lst name="defaults">
    <str name="qf">catchall</str>
  </lst>
</requestHandler>
<requestHandler name="/complex" class="solr.DisMax..." >
  <lst name="defaults">
    <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>
    ...
```

# 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 neccessary
    <uniqueKey>id</uniqueKey>
    <defaultSearchField>catchall</defaultSearchField>
  -->
</schema>
```

# Feeding Data From The Wild

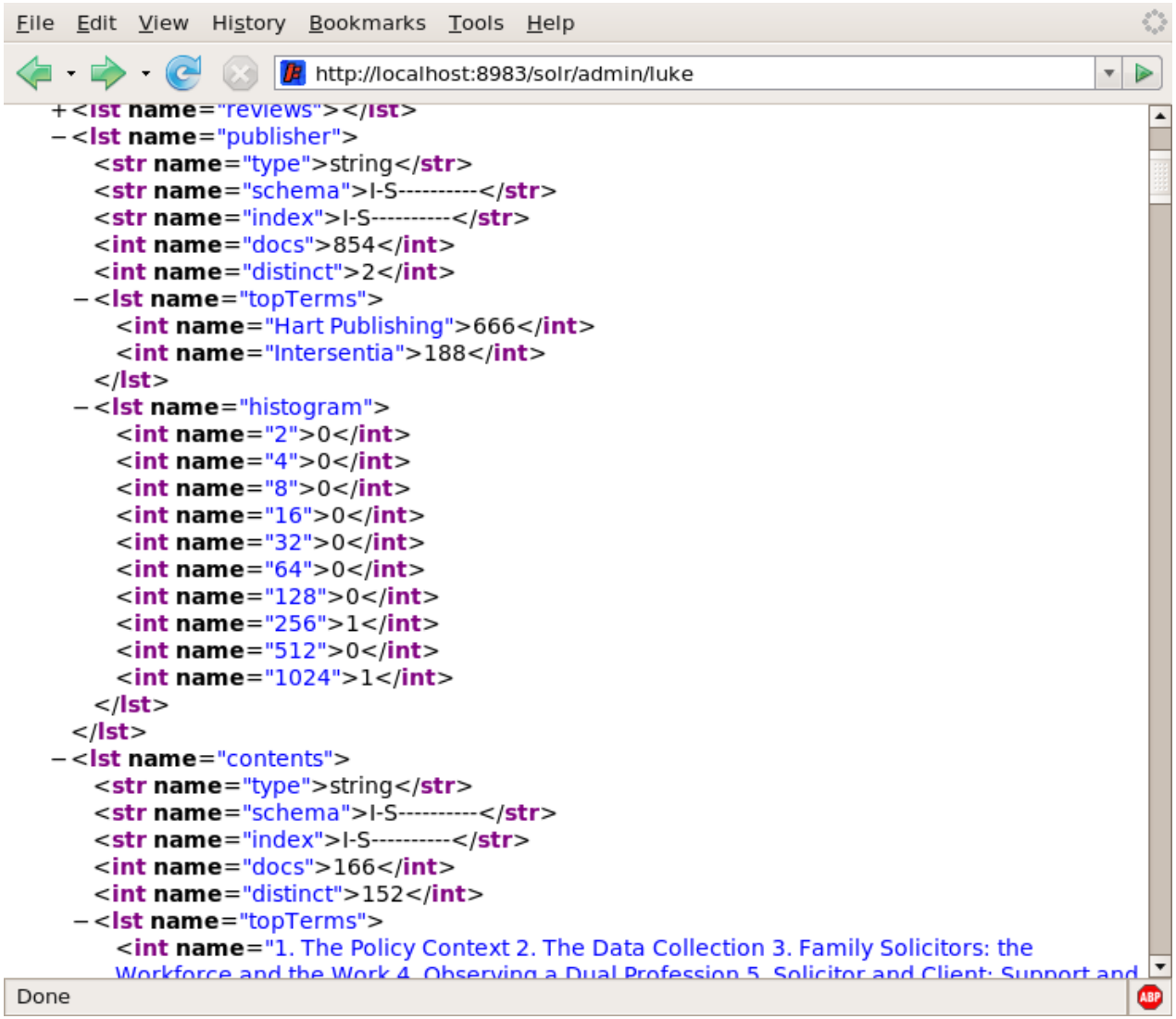
- I Went Online And Found A CSV File Containing Data On Books
- Deleted Some Non UTF-8 Characters
- Made Life Easier For Myself By Renaming The Columns So They Didn't Have Spaces

```
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

# Exmple: Luke Output



# 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!

# Score Explanations

- Why Did Document X Score Higher Then Document Y?
- Why Didn't Document Z Match At All?
- Debugging Options Append Detailed Score Explanations That Can Answer Both Questions...

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

# Explaining Explanations

- Explanations Are Not Easy To Understand
- Look For Key Concepts:
  - 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 (Based On Length And Some Indexing Options)
  - boost - How Important The Client Said This Query Clause Is
  - coordFactor - How Many Clauses Matched

# 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>
...
```

# Replication

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

# SpellcheckerRequestHandler

```
?q=comonallites&suggestionCount=10&accuracy=0.5
```

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">13</int>
  </lst>
  <arr name="suggestions">
    <str>commonalities</str>
    <str>commonality</str>
    <str>communality</str>
    <str>demonstrates</str>
  </arr>
</response>
```

# MoreLikeThisRequestHandler (1.3)

```
?q=id:SP2514N&mlt.fl=manu,cat&fl=id,name
```

```
<result name="response" numFound="13" start="0">
  <doc>
    <str name="id">6H500F0</str>
    <str name="name">
      Maxtor DiamondMax 11 - hard drive - 500 GB - SATA-300
    </str>
   </doc>
   <doc>
    <str name="id">F8V7067-APL-KIT</str>
    <str name="name">
      Belkin Mobile Power Cord for iPod w/ Dock
    </str>
   ...
```

# Questions?

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