O'REILLY®

OSCON data

open source convention

Solr Application Development Tutorial

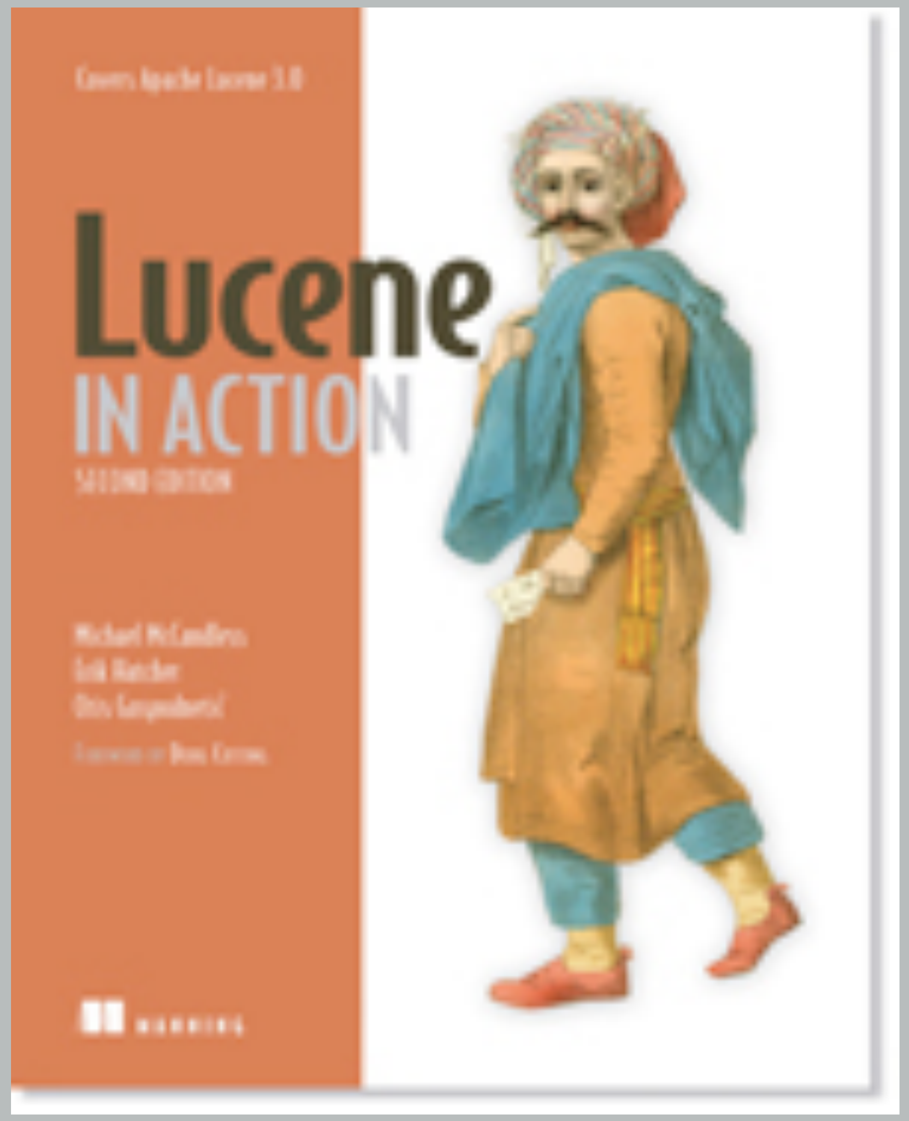
Presented by Erik Hatcher, Lucid Imagination
erik.hatcher@lucidimagination.com
http://www.lucidimagination.com

Abstract

- This fast-paced tutorial is targeted at developers who want to build applications with Solr, the Apache Lucene search server. You will learn how to set up and use Solr to index and search, how to analyze and solve common problems, and how to use many of Solr’s features such as faceting, spell checking, and highlighting.
- Topics covered include: how to make content searchable; basics and best practices for indexing and searching using Solr; how to integrate Solr into your solutions; techniques to analyze and resolve common search issues.

About me...

- Co-author, “Lucene in Action”
- Commiter, Lucene and Solr
- Lucene PMC and ASF member
- Member of Technical Staff / co-founder, Lucid Imagination



About Lucid Imagination...

- Lucid Imagination provides commercial-grade support, training, high-level consulting and value-added software for Lucene and Solr.
- We make Lucene ‘enterprise-ready’ by offering:
  - Free, certified, distributions and downloads
  - Support, training, and consulting
  - LucidWorks Enterprise, a commercial search platform built on top of Solr
  - Cloud offering in private beta - ask for more details!

What is Lucene?

- An open source Java-based IR library with best practice indexing and query capabilities, fast and lightweight search and indexing.
- 100% Java (.NET, Perl and other versions too).
- Stable, mature API.
- Continuously improved and tuned over more than 10 years.
- Cleanly implemented, easy to embed in an application.
- Compact, portable index representation.
- Programmable text analyzers, spell checking and highlighting.
- Not a crawler or a text extraction tool.

Lucene's History

- Created by Doug Cutting in 1999
  - built on ideas from search projects Doug created at Xerox PARC and Apple.
- Donated to the Apache Software Foundation (ASF) in 2001.
- Became an Apache top-level project in 2005.
- Has grown and morphed through the years and is now both:
  - A search library.
  - An ASF Top-Level Project (TLP) encompassing several sub-projects.
- Lucene and Solr "merged" development in early 2010.

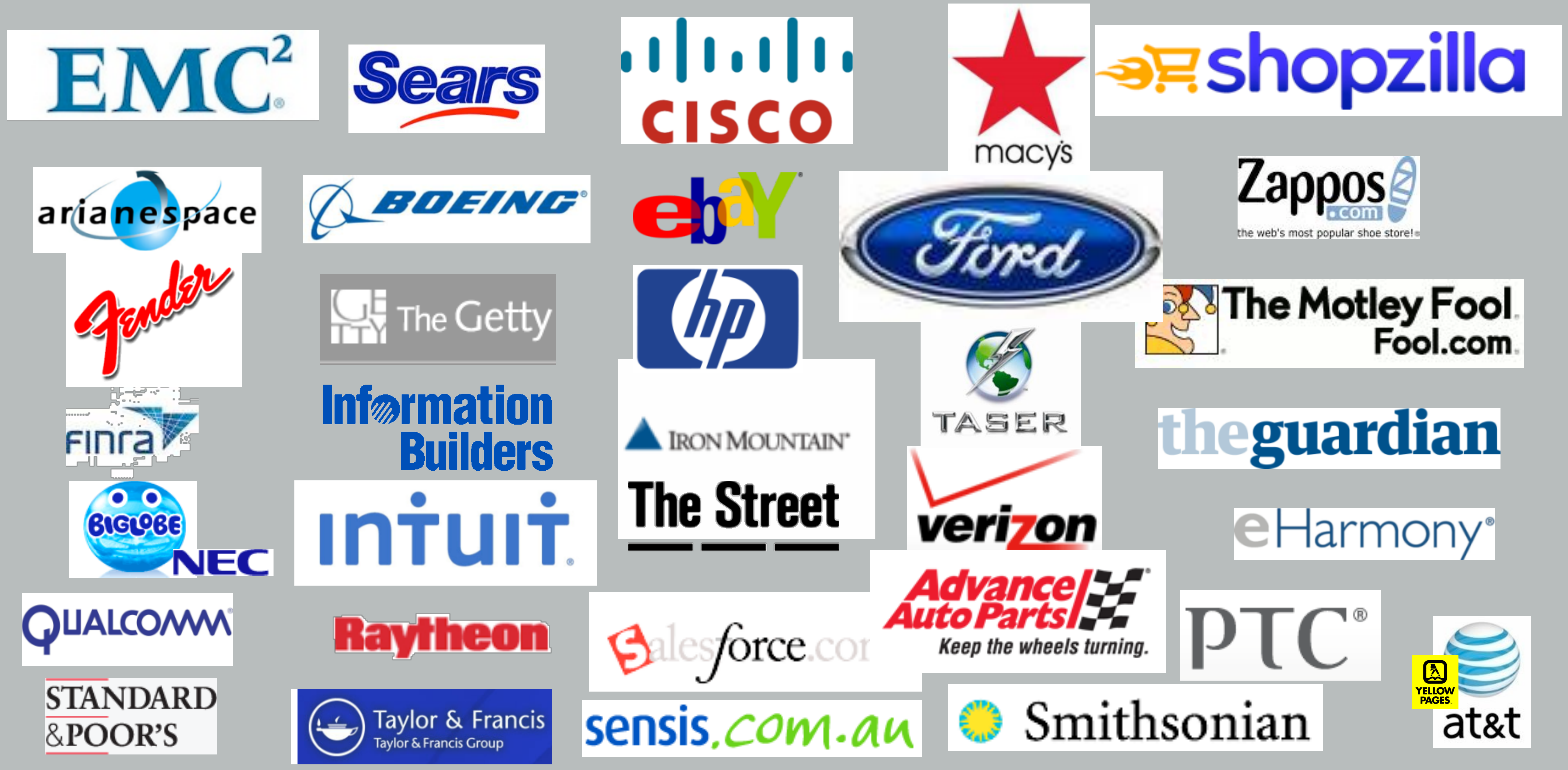
What is Solr?

- An open source search engine.
- Indexes content sources, processes query requests, returns search results.
- Uses Lucene as the "engine", but adds full enterprise search server features and capabilities.
- A web-based application that processes HTTP requests and returns HTTP responses.
- Initially started in 2004 and developed by CNET as an in-house project to add search capability for the company website.
- Donated to ASF in 2006.

What Version of Solr?

- There’s more than one answer!
- The current, released, stable version is 3.3
- The development release is referred to as “trunk”.
  - This is where the new, less tested work goes on
  - Also referred to as 4.0
- LucidWorks Enterprise is built on a trunk snapshot + additional features.

Who uses Lucene/Solr?

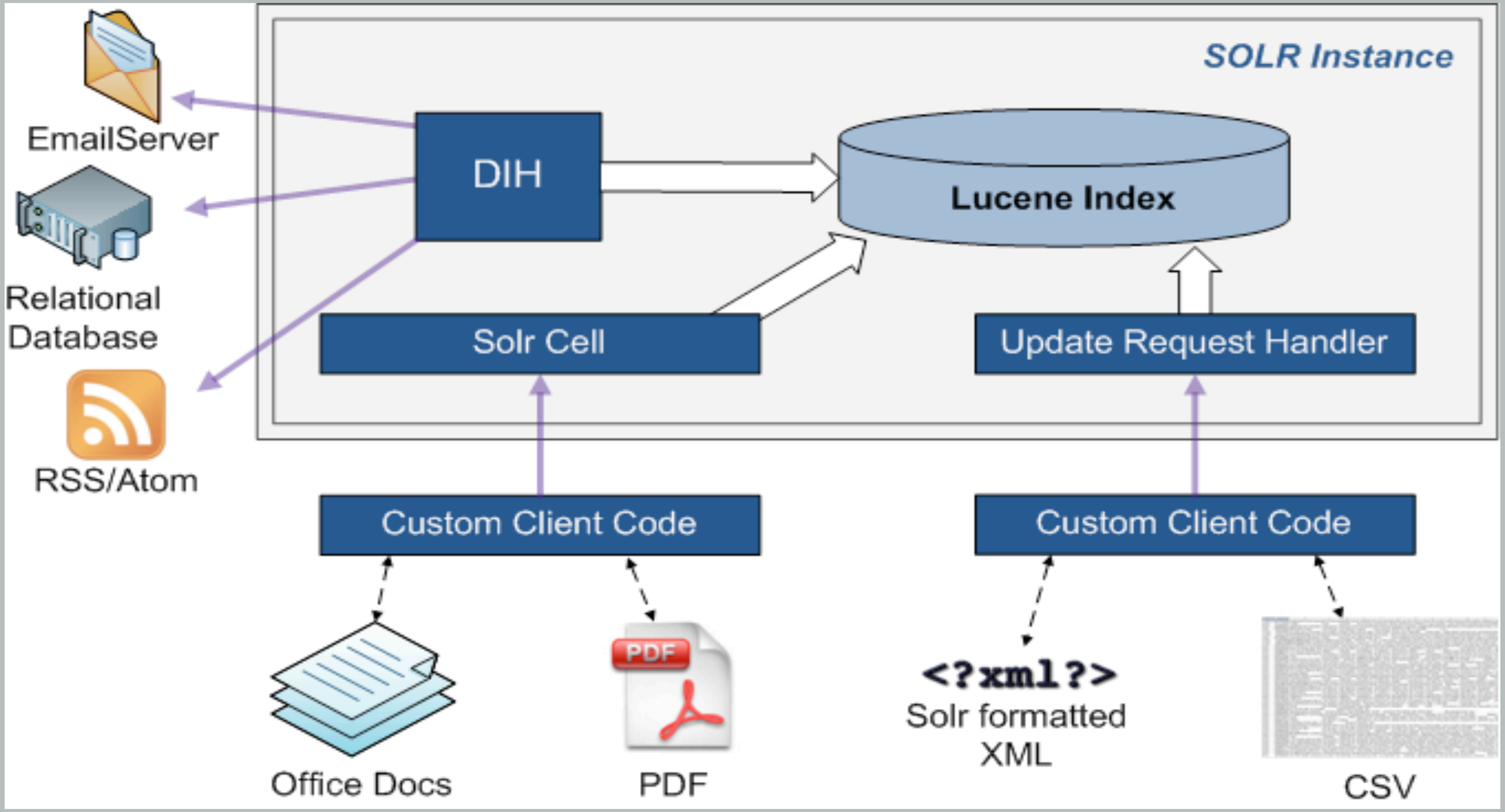


And many many many many more...!

Ingestion

- CSV
- Relational databases
- File system
- Web crawl
- API / Solr XML, JSON, and javabin/SolrJ
- Others - XML feeds (e.g. RSS/Atom), e-mail

Solr indexing options



Solr XML

- Techniques
  - /update

POST to /update

```
<add>
  <doc>
    <field name="id">rawxml1</field>
    <field name="content_type">text/xml</field>
    <field name="category">index example</field>
    <field name="title">Simple Example</field>
    <field name="filename">addExample.xml</field>
    <field name="text">A very simple example of
         adding a document to the index.</field>
  </doc>
</add>
```

CSV indexing

- http://localhost:8983/solr/update/csv
- Files can be sent over HTTP:

  -
    ```
    curl http://localhost:8983/solr/update/csv --data-binary @data.csv -H
    'Content-type:text/plain; charset=utf-8'
    ```
- or streamed from the file system:

  -
    ```
    curl http://localhost:8983/solr/update/csv?stream.file=exampledocs/
    data.csv&stream.contentType=text/plain;charset=utf-8
    ```

CSV indexing params

- header: Default is true. Indicates that the first line of the file contains field names.
- fieldnames: A comma-separated list of fieldnames which will override the header parameter.
- separator: Defaults to a comma, but other characters can be used.
- trim: If true, remove whitespace before and after separator.
- Full parameter list: http://wiki.apache.org/solr/UpdateCSV
- Yes, tab-delimited files are supported too:
  - &separator=%09

Rich documents

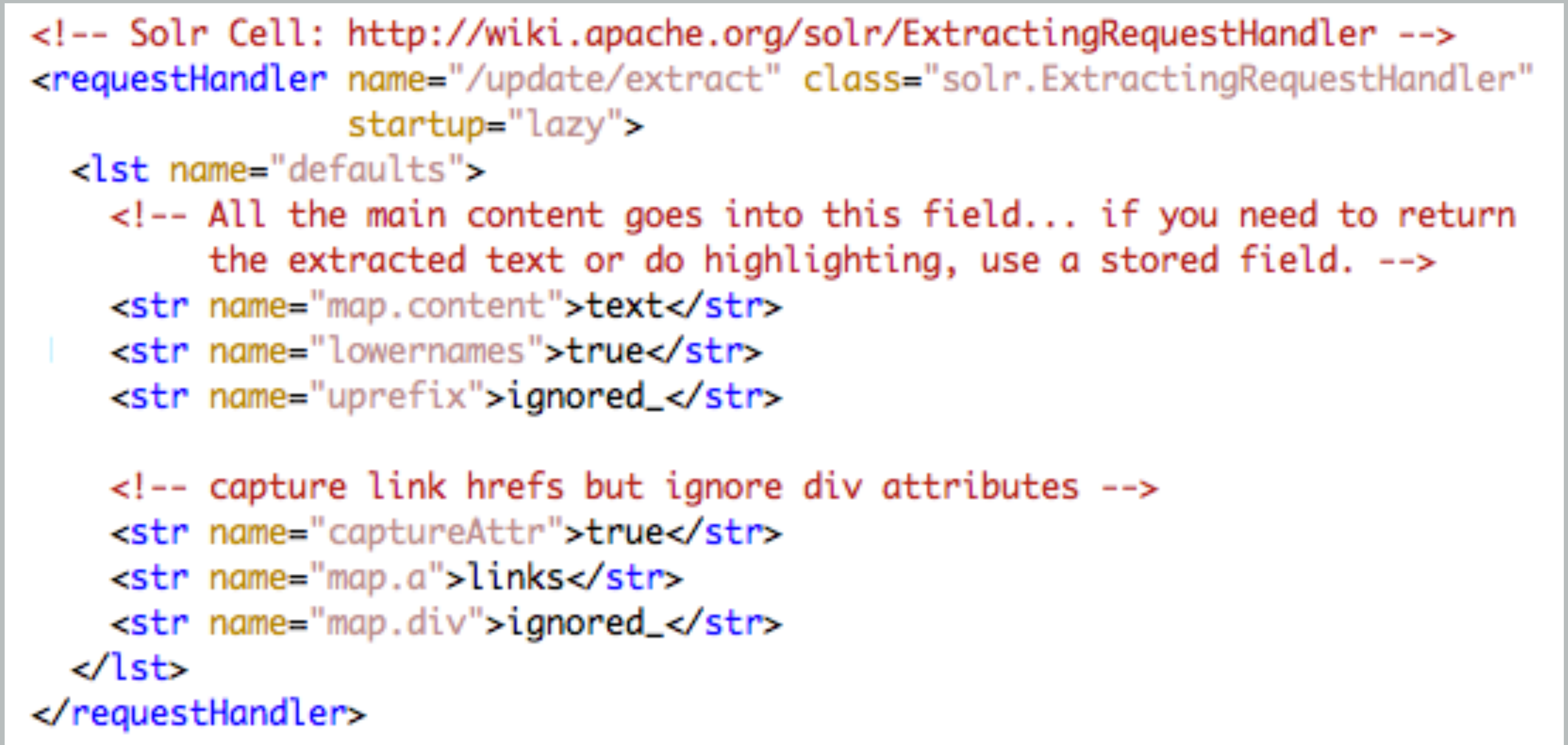
- Tika is a toolkit for detecting and extracting metadata and structured text content from various document formats using existing parser libraries.
- Tika identifies MIME types and then uses the appropriate parser to extract text.
- The ExtractingRequestHandler uses Tika to identify types and extract text, and then indexes the extracted text.
- The ExtractingRequestHandler is sometimes called "Solr Cell", which stands for Content Extraction Library.
- File formats include MS Office, Adobe PDF, XML, HTML, MPEG and many more.

Solr Cell basics

- defaultField: If uprefix is not set, and a field cannot be determined, the default field is used.
- Full list of parameters: http://wiki.apache.org/solr/ExtractingRequestHandler

Solr Cell config

- Indexing Rich Content with Solr
- The ExtractingRequestHandler is configured in solrconfig.xml



```
<!-- Solr Cell: http://wiki.apache.org/solr/ExtractingRequestHandler -->
<requestHandler name="/update/extract" class="solr.ExtractingRequestHandler"
                startup="lazy">
  <lst name="defaults">
    <!-- All the main content goes into this field... if you need to return
         the extracted text or do highlighting, use a stored field. -->
    <str name="map.content">text</str>
    <str name="lowernames">true</str>
    <str name="uprefix">ignored_</str>

    <!-- capture link hrefs but ignore div attributes -->
    <str name="captureAttr">true</str>
    <str name="map.a">links</str>
    <str name="map.div">ignored_</str>
  </lst>
</requestHandler>
```

Solr Cell parameters

- The literal parameter is very important.
  - A way to add other fields not indexed using Tika to documents.
  - &literal.id=12345
  - &literal.category=sports
- Using curl to index a file on the file system:

  -
    ```
    curl 'http://localhost:8983/solr/update/extract?
    literal.id=doc1&commit=true' -F myfile=@tutorial.html
    ```
- Streaming a file from the file system:

  -
    ```
    curl "http://localhost:8983/solr/update/extract?stream.file=/some/path/
    news.doc&stream.contentType=application/msword&literal.id=12345"
    ```

Streaming remote docs

- Streaming a file from a URL:
  - `curl http://localhost:8983/solr/update/extract?literal.id=123&stream.url=http://www.solr.com/content/goodContent.pdf -H 'Content-type:application/pdf’`

Indexing from RDBMS

- Techniques
  - DIH
  - Custom script
- Hurdles
  - Flattening
  - Performance

DataImportHandler

- The DataImportHandler (DIH):
  - An "in-process" module that can be used to index data directly from relational databases and other data sources.
  - Configuration driven.
  - A tool that can aggregate data from multiple database tables, or even multiple data sources to be indexed as a single Solr document.
  - Provides powerful and customizable data transformation tools.
  - Can do full import or delta import.
  - Pluggable to allow indexing of any type of data source.

Using Solr’s Data Import Handler

- One or more configuration files can be created for DIH instances.
- All DIH configuration files must be declared in a request handler in solrconfig.xml and given a unique name:

```
<requestHandler
  class="org.apache.solr.handler.dataimport.DataImportHandler"
  name="/dataimport">
    <lst name="defaults">
      <str name="config">db-config.xml</str>
    </lst>
</requestHandler>
```

Enabling the DataImportHandler

- The jar files for DIH must be included on the path
  - apache-solr-dataimporthandler-3.3.0.jar
  - apache-solr-dataimporthandler-extras-3.3.0.jar
  - This is a change since version 1.4.1
- The example solrconfig file include a “lib” command to include these files.

DIH Components

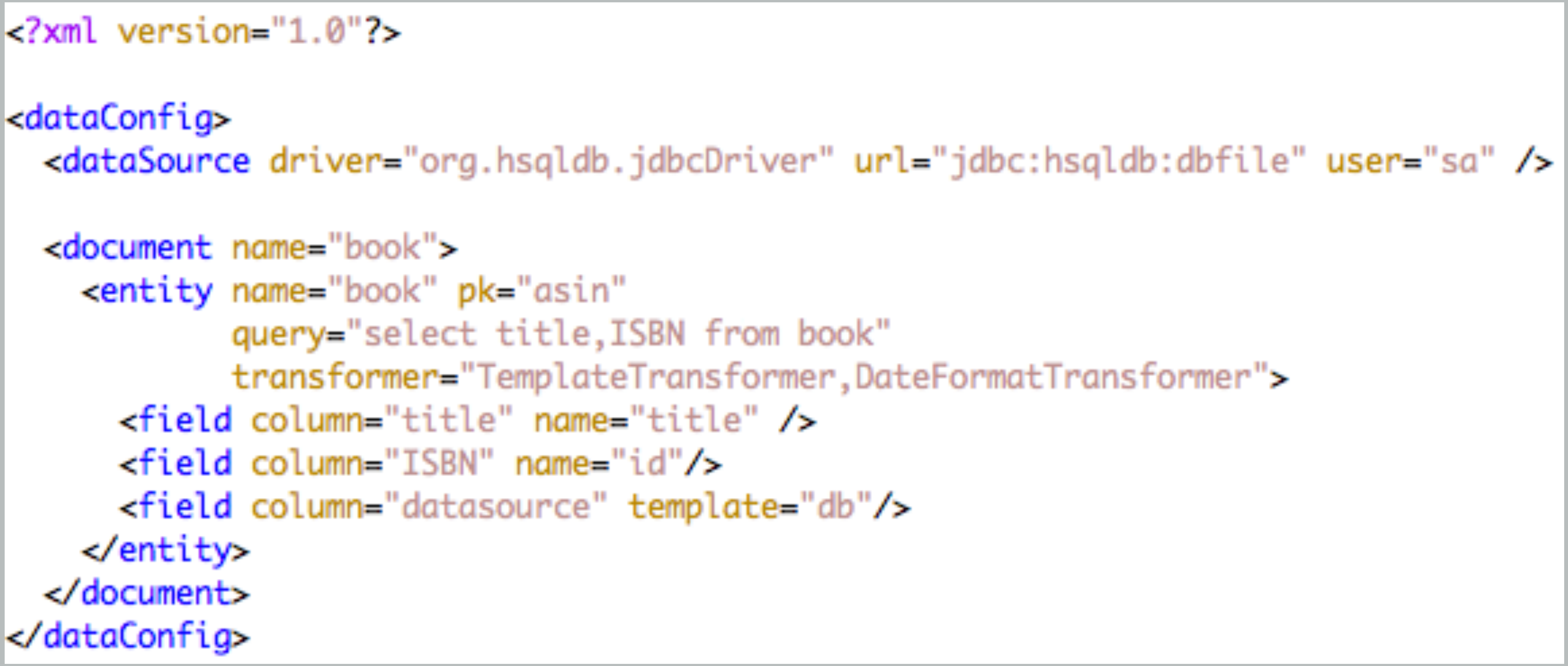
- Data sources:
  - Provide access to relational databases, plain files, XML, etc.
- Entity processors:
  - Configuration elements that work with data sources and do most of the work.
  - Can be nested.
- Transformers:
  - Modify data in many different ways.
  - Can be chained together.
  - Custom transformers can be written.

DIH Capabilities

- From a database. (SQL entity processor, Jdbc datasource)
- An RSS or Atom feed. (XPath entity processor, URL datasource)
- XML files. (Xpath entity processor, File datasource)
- Plain text files. (Plaintext entity processor, File datasource)
- From a mail server. (Mail entity processor)

DIH with database

- Create a document with database fields title, ISBN mapped to Solr fields title, id
- No EntityProcessor is included so the default SqlEntityProcessor is used.



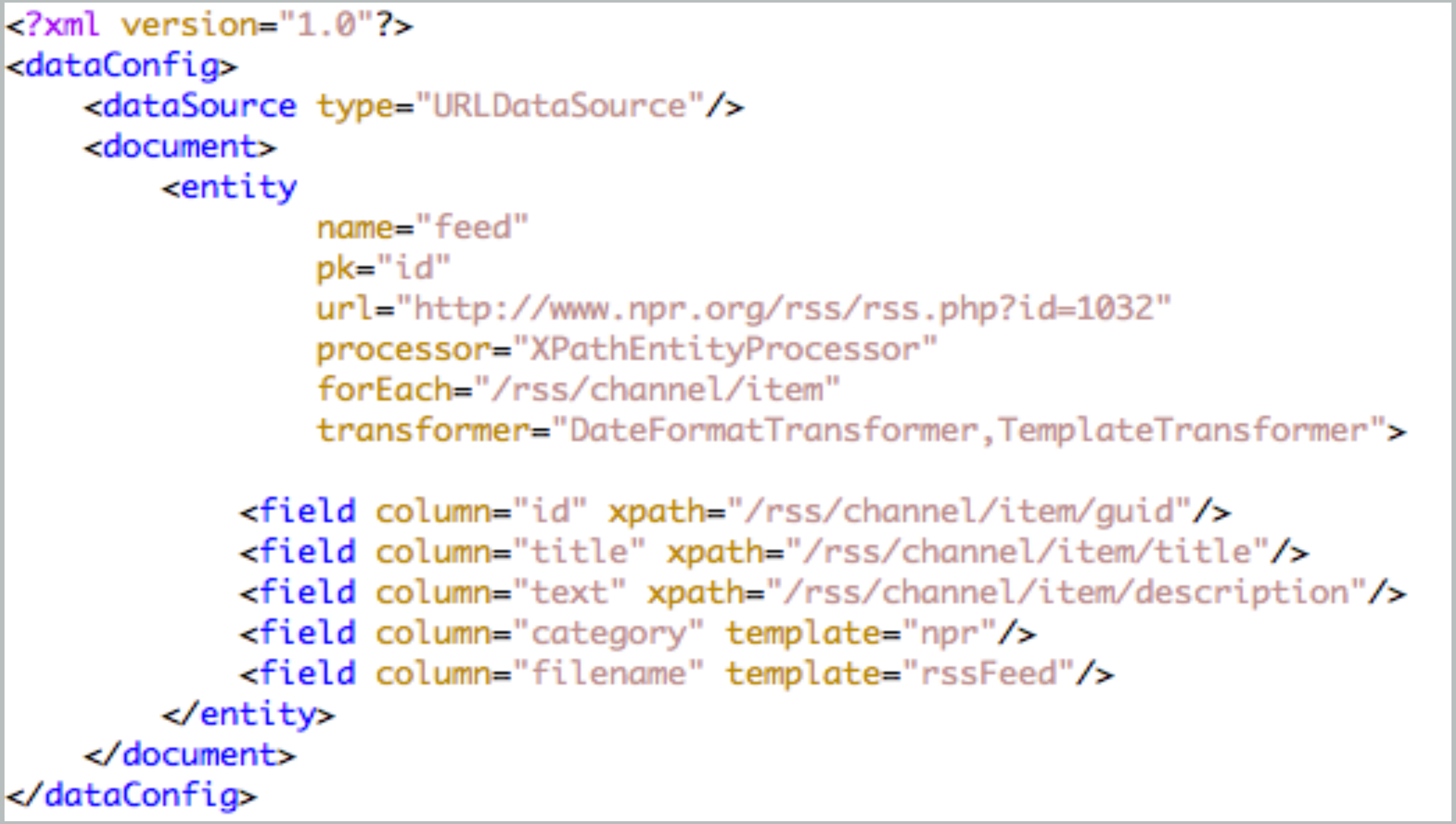
```xml
<?xml version="1.0"?>

<dataConfig>
  <dataSource driver="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:dbfile" user="sa" />

  <document name="book">
    <entity name="book" pk="asin"
            query="select title,ISBN from book"
            transformer="TemplateTransformer,DateFormatTransformer">
      <field column="title" name="title" />
      <field column="ISBN" name="id"/>
      <field column="datasource" template="db"/>
    </entity>
  </document>
</dataConfig>
```

DIH with RSS

- Using the XPathEntityProcessor to map data to index fields.

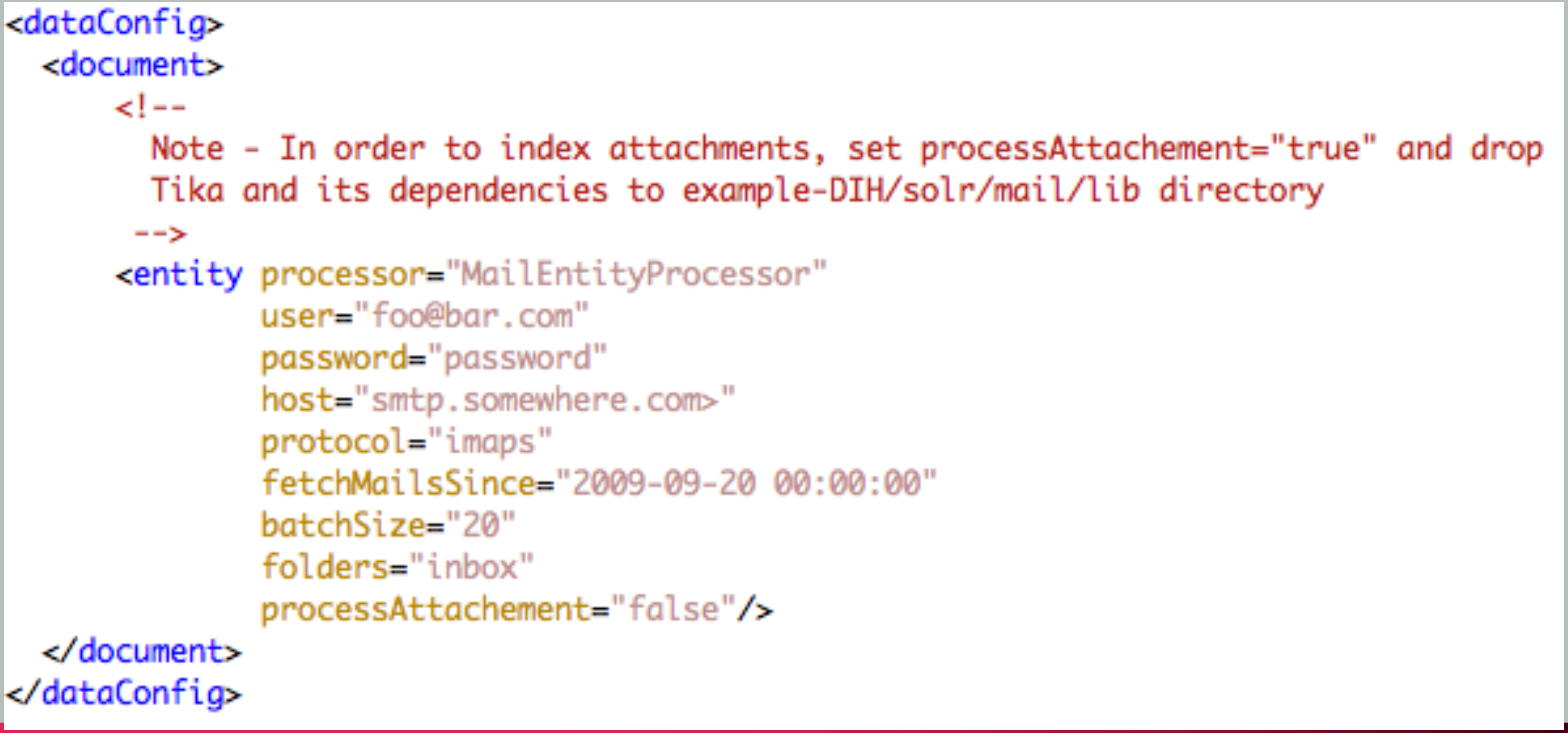


```
<?xml version="1.0"?>
<dataConfig>
    <dataSource type="URLDataSource"/>
    <document>
        <entity
                name="feed"
                pk="id"
                url="http://www.npr.org/rss/rss.php?id=1032"
                processor="XPathEntityProcessor"
                forEach="/rss/channel/item"
                transformer="DateFormatTransformer,TemplateTransformer">

            <field column="id" xpath="/rss/channel/item/guid"/>
            <field column="title" xpath="/rss/channel/item/title"/>
            <field column="text" xpath="/rss/channel/item/description"/>
            <field column="category" template="npr"/>
            <field column="filename" template="rssFeed"/>
        </entity>
    </document>
</dataConfig>
```

DIH with email

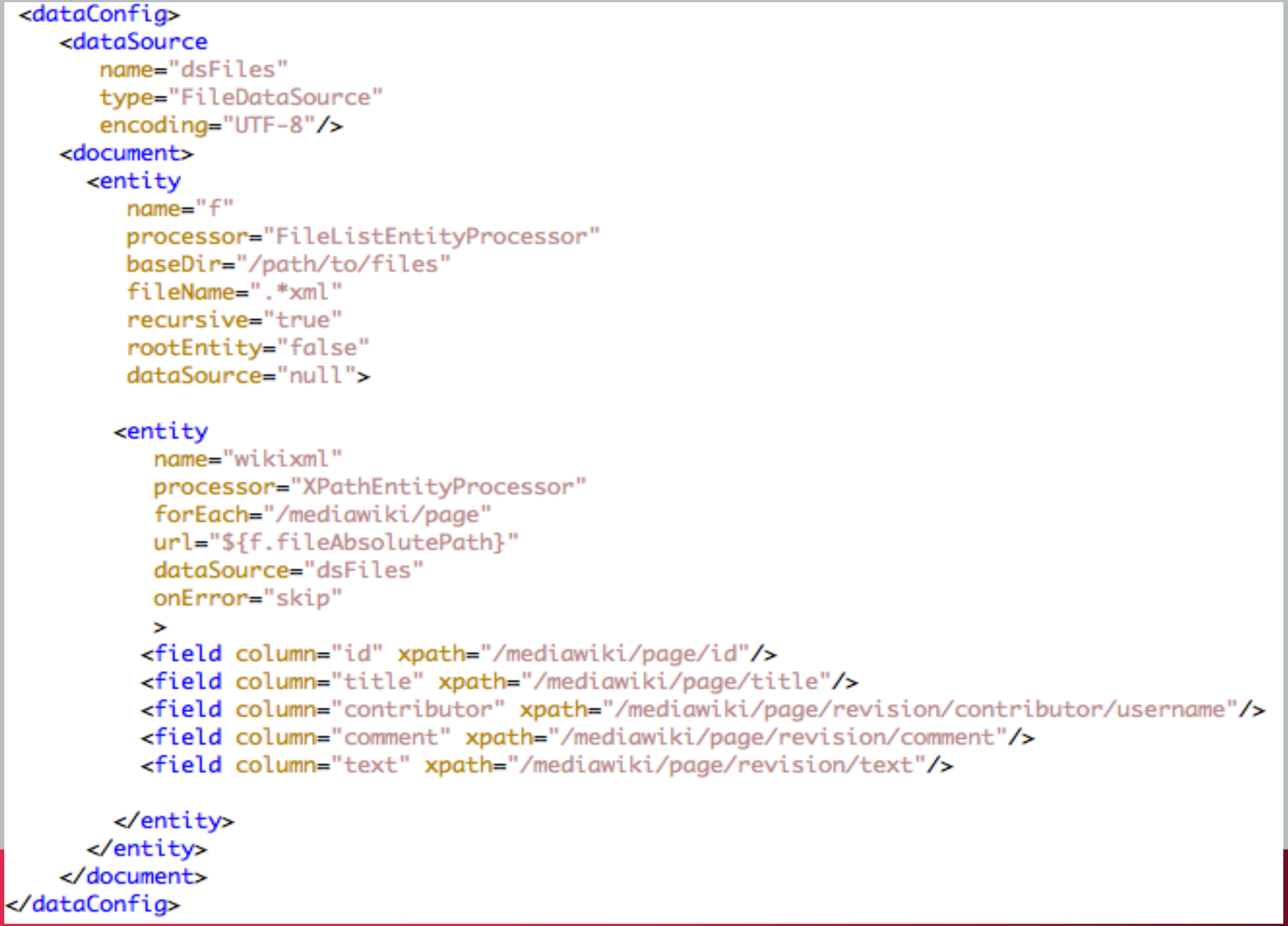
- Mail Input: Example DIH Configuration File
- Using the MailEntityProcessor to index email data.



```
<dataConfig>
  <document>
      <!--
        Note - In order to index attachments, set processAttachement="true" and drop
        Tika and its dependencies to example-DIH/solr/mail/lib directory
       -->
      <entity processor="MailEntityProcessor"
              user="foo@bar.com"
              password="password"
              host="smtp.somewhere.com>"
              protocol="imaps"
              fetchMailsSince="2009-09-20 00:00:00"
              batchSize="20"
              folders="inbox"
              processAttachement="false"/>
  </document>
</dataConfig>
```

DIH - richer example

- XML Files Input: Example DIH Configuration File



```
 <dataConfig>
   <dataSource
      name="dsFiles"
      type="FileDataSource"
      encoding="UTF-8"/>
   <document>
     <entity
        name="f"
        processor="FileListEntityProcessor"
        baseDir="/path/to/files"
        fileName=".*xml"
        recursive="true"
        rootEntity="false"
        dataSource="null">

       <entity
          name="wikixml"
          processor="XPathEntityProcessor"
          forEach="/mediawiki/page"
          url="${f.fileAbsolutePath}"
          dataSource="dsFiles"
          onError="skip"
          >
         <field column="id" xpath="/mediawiki/page/id"/>
         <field column="title" xpath="/mediawiki/page/title"/>
         <field column="contributor" xpath="/mediawiki/page/revision/contributor/username"/>
         <field column="comment" xpath="/mediawiki/page/revision/comment"/>
         <field column="text" xpath="/mediawiki/page/revision/text"/>

       </entity>
     </entity>
   </document>
</dataConfig>
```

DIH DataSources

- JdbcDataSource: Default if none is specified. Iterates rows of a database one by one.
- URLDataSource: Used to fetch content from file:// or http:// locations.
- FileDataSource: Similar to URLDataSource, but locations are specified with a "basePath" parameter.
- The class org.apache.solr.handler.dataimport.DataSource can be extended to create custom data sources.

DIH EntityProcessors

- SqlEntityProcessor: Default if none is specified. works with a JdbcDataSource to index database tables.
- XPathEntityProcessor: Implements a streaming parser which supports a subset of xpath syntax. Complete xpath syntax is not yet supported.
- FileListEntityProcessor: Does not use a DataSource. Enumerates a list of files. Typically used as an "outer" entity.
- CachedSqlEntityProcessor: An extension of the SqlEntityProcessor reduces the number of queries executed by caching rows. (Only for inner nested entities.)
- PlainTextEntityProcessor: Reads text into a "plainText" field.

DIH transforming

- Fields that are processed can either be indexed directly or transformed and modified.
- New fields can be created.
- Transformers can be chained.

DIH Transformers

- RegexTransformer: Manipulates field values using regular expressions.
- DateFormatTransformer: Parses date/time strings into java.util.Date instances.
- NumberFormatTransformer: Parses numbers from a string.
- TemplateTransformer: Explicitly sets a text value. Optionally can use variable names.
- HTMLStringTransformer: Removes HTML markup.
- ClobTransformer: Creates a string from a CLOB data type.
- ScriptTransformer: Write custom transformers in JavaScript or other scripting languages.

DIH Full and Delta Imports

- The DIH can be used for both full imports and delta imports.
- The query element is used for a full import.
- The deltaQuery element gives the primary keys of the current entity which have changes since the last index time. These primary keys will be used by the deltaImportQuery.
- The deltaImportQuery element gives the data needed to populate fields when running a delta-import .

DIH basic commands

- Full import example:
  - http://localhost:8983/solr/dataimport?command=full-import
- Delta import example:
  - http://localhost:8983/solr/dataimport?command=delta-import
- The "rows" parameter can be used to limit the amount of input:
  - http://localhost:8983/solr/dataimport?command=full-import&rows=10
- The "commit" parameter defaults to "true" if not explicitly set:
  - http://localhost:8983/solr/dataimport?command=full-import&rows=10&commit=false

DIH "clean" param

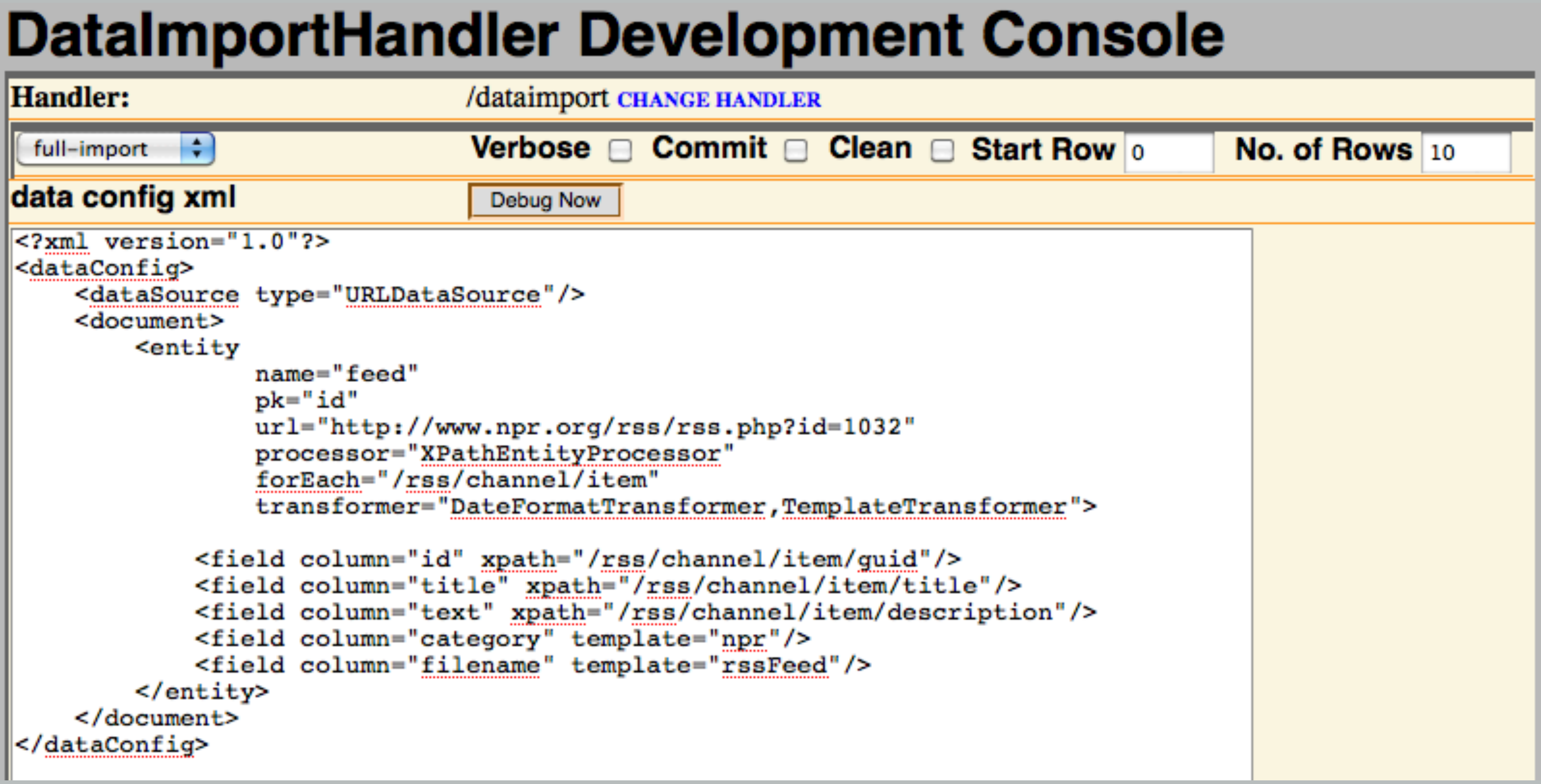
- Be careful with the "clean" parameter.
- clean=true will delete everything from the index – all documents will be deleted.
- clean=true is the default!
- Get in the habit of always setting the clean parameter so you are not surprised with unexpected data loss.
  - http://localhost:8983/solr/dataimport?command=full-import&clean=false

DIH Admin Console

- There is an admin console page for the DIH, but there is no link to it from the main admin page.
  - http://localhost:8983/solr/admin/dataimport.jsp
  - The link brings up a list of all DIH configurations (there can be more than one.)

Using Solr’s Data Import Handler

- DataImportHandler Admin Console
- The main section shows the configuration and a few commands:



DataImportHandler Development Console

Handler: /dataimport CHANGE HANDLER

full-import Verbose □ Commit □ Clean □ Start Row 0 No. of Rows 10

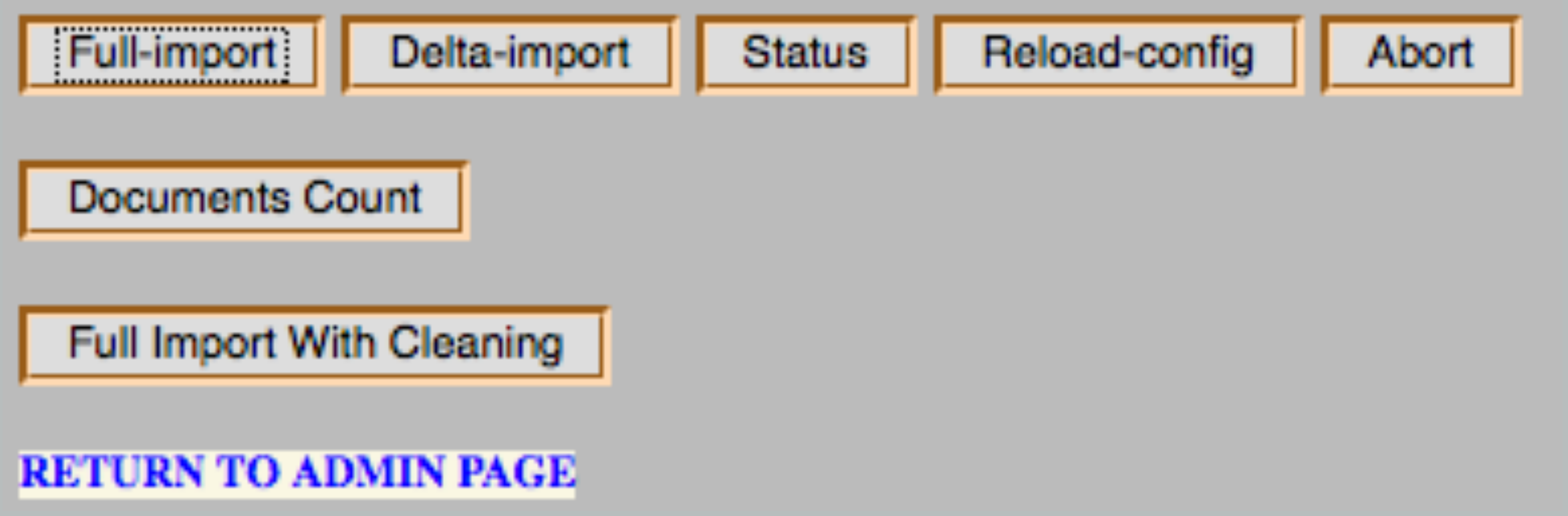
data config xml Debug Now

```
<?xml version="1.0"?>
<dataConfig>
    <dataSource type="URLDataSource"/>
    <document>
        <entity
                name="feed"
                pk="id"
                url="http://www.npr.org/rss/rss.php?id=1032"
                processor="XPathEntityProcessor"
                forEach="/rss/channel/item"
                transformer="DateFormatTransformer,TemplateTransformer">

            <field column="id" xpath="/rss/channel/item/guid"/>
            <field column="title" xpath="/rss/channel/item/title"/>
            <field column="text" xpath="/rss/channel/item/description"/>
            <field column="category" template="npr"/>
            <field column="filename" template="rssFeed"/>
        </entity>
    </document>
</dataConfig>
```

DIH console commands

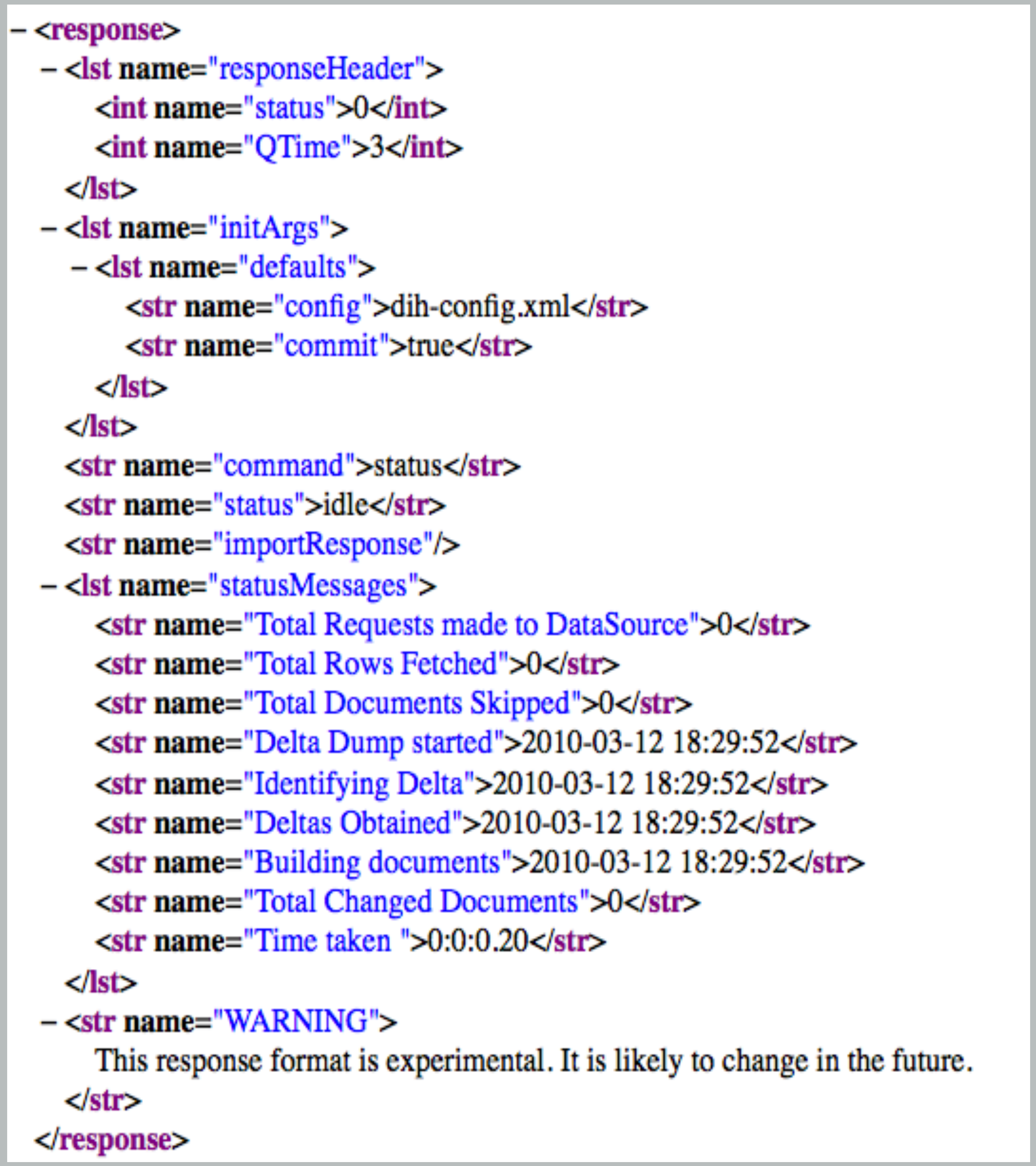
- At the bottom of the page there are options for running various operations such as full-imports and delta-imports:



Note that all of these commands can also be executed from the command line using curl or wget.

DIH status

- The display to the right shows the XML output of commands that are run from the console.
- This example shows the response after a delta-import.



```
- <response>
  - <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">3</int>
    </lst>
  - <lst name="initArgs">
    - <lst name="defaults">
        <str name="config">dih-config.xml</str>
        <str name="commit">true</str>
      </lst>
    </lst>
    <str name="command">status</str>
    <str name="status">idle</str>
    <str name="importResponse"/>
  - <lst name="statusMessages">
      <str name="Total Requests made to DataSource">0</str>
      <str name="Total Rows Fetched">0</str>
      <str name="Total Documents Skipped">0</str>
      <str name="Delta Dump started">2010-03-12 18:29:52</str>
      <str name="Identifying Delta">2010-03-12 18:29:52</str>
      <str name="Deltas Obtained">2010-03-12 18:29:52</str>
      <str name="Building documents">2010-03-12 18:29:52</str>
      <str name="Total Changed Documents">0</str>
      <str name="Time taken ">0:0:0.20</str>
    </lst>
  - <str name="WARNING">
      This response format is experimental. It is likely to change in the future.
    </str>
  </response>
```

DIH experimenting

- You can also view the status of an ongoing process (for example a long import) by going directory to the URL for the handler:
  - http://localhost:8983/solr/dataimport
- curl can also be used with the DIH:
  - `curl 'http://localhost:8983/solr/dataimport?command=full-import&rows=10'`
  - Setting rows=10 is a good way to limit the indexing during development.

Other DIH capabilities

- TikaEntityProcessor
- SolrEntityProcessor (see SOLR-1499)
- Multi-threaded capabilities

Web crawl

- Solr is not a crawler
- Options: Nutch, droids, or LucidWorks Enterprise

Committing

```
curl http://localhost:8983/solr/update
     -H "Content-Type: text/xml"
     --data-binary "<commit/>"
```

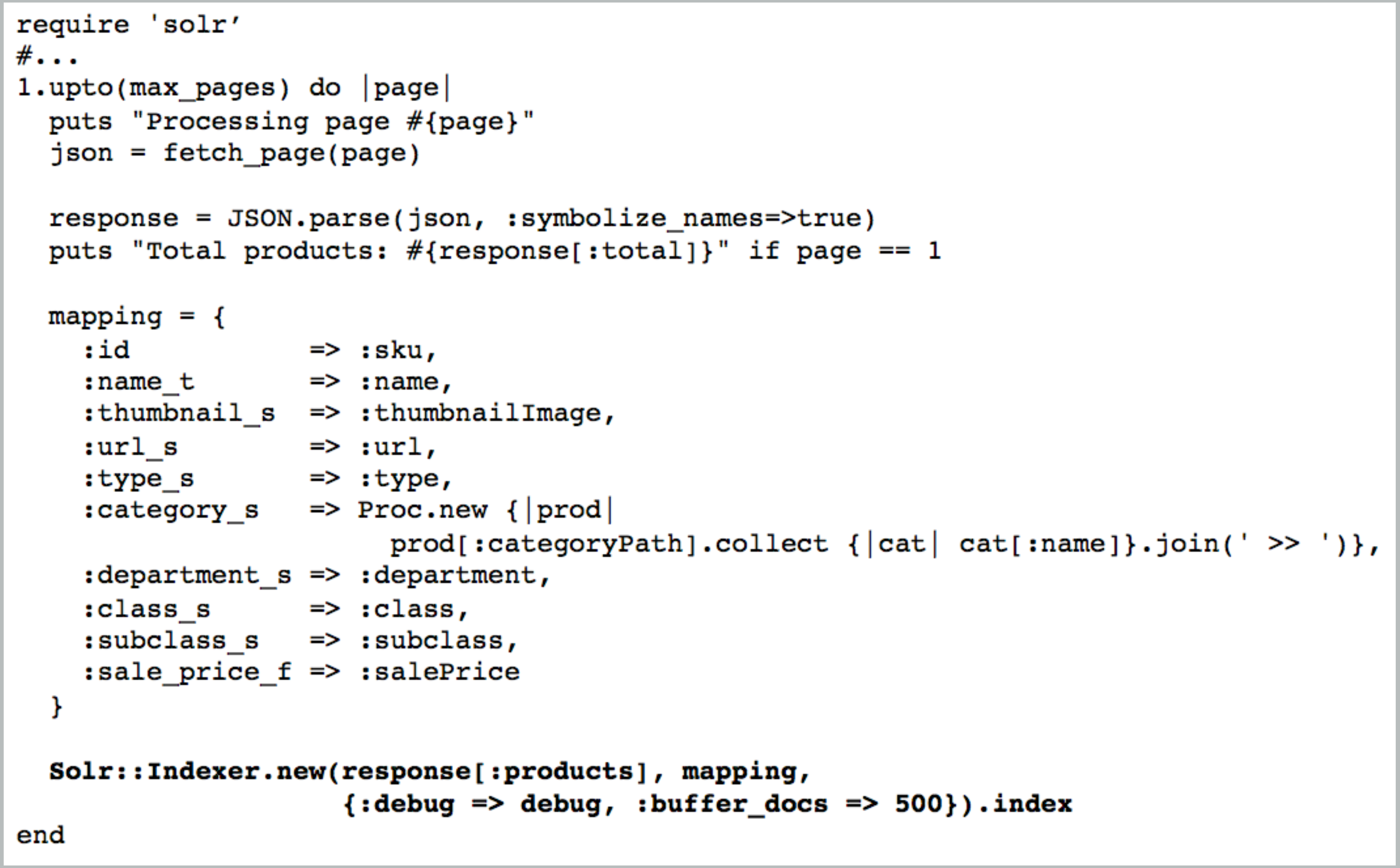
API Indexing

- Solr XML
- Solr JSON
- SolrJ - javabin format, streaming/multithread

Solr APIs

- SolrJ can use an internal javabin format (or XML)
- Most other Solr APIs ride on Solr XML or Solr JSON formats

Ruby indexing example



```
require 'solr’
#...
1.upto(max_pages) do |page|
  puts "Processing page #{page}"
  json = fetch_page(page)

  response = JSON.parse(json, :symbolize_names=>true)
  puts "Total products: #{response[:total]}" if page == 1

  mapping = {
    :id            => :sku,
    :name_t        => :name,
    :thumbnail_s   => :thumbnailImage,
    :url_s         => :url,
    :type_s        => :type,
    :category_s    => Proc.new {|prod|
                        prod[:categoryPath].collect {|cat| cat[:name]}.join(' >> ')},
    :department_s => :department,
    :class_s       => :class,
    :subclass_s    => :subclass,
    :sale_price_f => :salePrice
  }

  Solr::Indexer.new(response[:products], mapping,
                    {:debug => debug, :buffer_docs => 500}).index
end
```

SolrJ searching example

- DEMO: Look at code in IDE

```
SolrServer solrServer = new CommonsHttpSolrServer(
                                  "http://localhost:8983/solr");
SolrQuery query = new SolrQuery();
query.setQuery(userQuery);
query.setFacet(true);
query.setFacetMinCount(1);
query.addFacetField("category");

QueryResponse queryResponse = solrServer.query(query);
```

Updating documents

- Solr uses the "uniqueKey" to determine a the "identity" of a document.
- Adding a document to the index with the same uniqueKey as an existing document means the new document will replace the original.
- An "update" is actually two steps, internally:
  - Delete the document with that id.
  - Add the new document.
  - So documents are, more accurately, "replaced", not deleted.
  - No field-level updating – a whole document has to be replaced

Deleting documents

- Documents can be deleted:
  - Using a delete by id.
    - `<delete><id>05991</id></delete>`
  - Using a delete by query.
    - `<delete><query>category:music</query></delete>`
- When a document is deleted it still exists in an index segment until that segment is merged.
- Rollback: <rollback/>
  - All adds and deletes since the last commit are rolled back.

Fields

- data analysis / exploration
- character mapping
- tokenizing/filtering
- copyField

Field Types: "primitives"

- field types entirely specified in schema.xml, but... keep the standard (non TextField ones) as-is from Solr's provided example schema.
- string, boolean, binary, int, float, long, double, date
- numeric types for faster range queries (and bigger index): tint, tfloat, tlong, tdouble, tdate

TextField

- TextField "analyzes" field value

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
            enablePositionIncrements="true" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
            enablePositionIncrements="true" />
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt"
            ignoreCase="true" expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Character Filters

He went to the café

```
<fieldType name="text_ws" class="solr.TextField">
  <analyzer>
    <charFilter
     class="solr.MappingCharFilterFactory"
         mapping="mapping-ISOLatin1Accent.txt"/>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

he went to the cafe

copyField

- Clones the exact field value to another field
- destination field controls handling of cloned value
- Useful when same field value needs to be indexed/stored in several different ways
  - e.g. sorting

Schema Deep Dive

- DEMO: Let's look at schema.xml interactively, and discuss other features

Searching

- The basic parameters
- Filtering
- Query parsing

Searching Basics

- http://localhost:8983/solr/select?q=*:*

Basic Search Params

- q - main query
- rows - maximum number of "hits" to return
- start - zero-based hit starting point
- fl - comma-separated field list
  - * for all stored fields
  - score for computed Lucene score

Other Common Search Parameters

- sort - specify sort criteria either by field(s) or function(s) in ascending or descending order
- fq - filter queries, multiple values supported
- wt - writer type - format of Solr response
- debugQuery - adds debugging info to response

Filtering results

- Use fq to filter results in addition to main query constraints
- fq results are independently cached in Solr's filterCache
- filter queries do not contribute to ranking scores
- Commonly used for filtering on facets

Richer Solr Request

- http://localhost:8983/solr/select
  ?q=ipod
  &facet=on
  &facet.field=cat
  &fq=cat:electronics

Query Parsing

- String -> org.apache.lucene.search.Query
- Several built-in query parser choices, including
  - "lucene" - Solr-savvy version of Lucene's standard query parser
  - "dismax" - most commonly used, currently
  - "edismax" - extended dismax

Defaults for query "lucene" and "(e)dismax"

- schema.xml
  - <defaultSearchField>field</defaultSearchField>
  - <solrQueryParser defaultOperator="AND|OR"/>

Selecting Query Parser

- defType=lucene|dismax|edismax|....
  - Sets query parser for q parameter only
- Or via {!local_params} syntax
  - q={!dismax}query expression
  - only way to set query parser for fq's

"lucene" query parser

- Solr subclass of Lucene's QueryParser
  - schema-aware, for range queries, etc
  - special handling of wildcarded clauses
  - back-door special "fields"
    - _val_ - for function queries
    - _query_ - for nested queries

"lucene" parser examples

- search AND (lucene OR solr)
- rating:[7 TO 10]
- +required -prohibited
- wild?card OR prefix* AND fuzzy~
- "phrases for proximity"

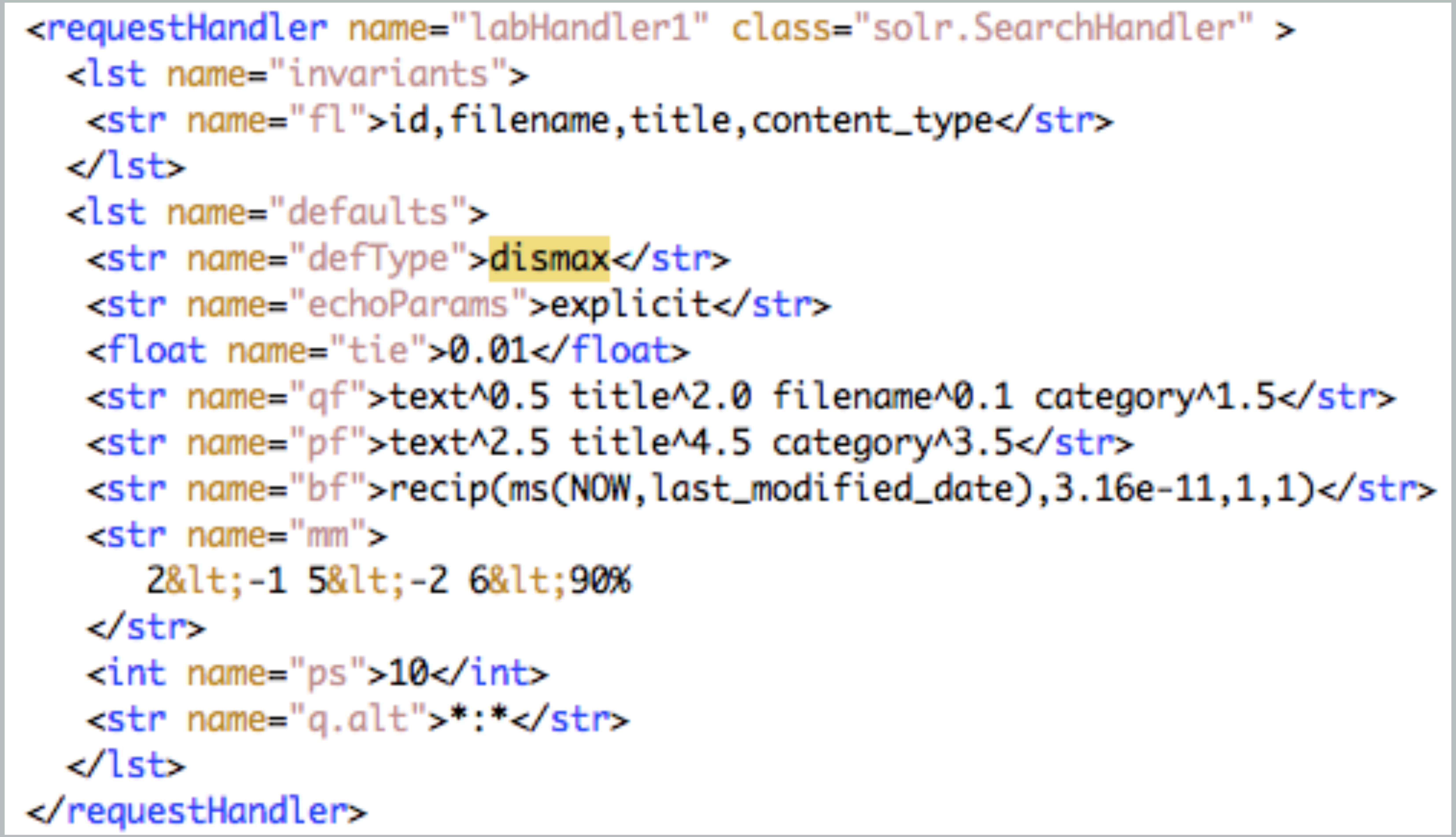
"dismax" query parser

- disjunction-maximum
- enables spreading query terms across multiple fields with individual field-specific boosts
- many parameters

"dismax" params

Parameter	Description
q	Defines the raw input strings for the query.
q.alt	Calls the Lucene query parser and defines query input strings when the q parameter is not used. (useful for getting facet counts when no query specified)
qf	Query Fields: Specifies the fields to be searched.
pf	Phrase Fields: Fields will be queried using the terms entered as a phrase query.
ps	Phrase Slop: How close to one another the terms within a phrase query must be.
mm	Minimum "Should" Match: Number of fields that must match a query
tie	Tie Breaker: A float value to use as a tie breaker
bq	Boost Query: An optional query that can be used to boost and refine results
bf	Boost Functions: Functions that can be used to tune relevance

"dismax" config example



```
<requestHandler name="labHandler1" class="solr.SearchHandler" >
  <lst name="invariants">
   <str name="fl">id,filename,title,content_type</str>
  </lst>
  <lst name="defaults">
   <str name="defType">dismax</str>
   <str name="echoParams">explicit</str>
   <float name="tie">0.01</float>
   <str name="qf">text^0.5 title^2.0 filename^0.1 category^1.5</str>
   <str name="pf">text^2.5 title^4.5 category^3.5</str>
   <str name="bf">recip(ms(NOW,last_modified_date),3.16e-11,1,1)</str>
   <str name="mm">
      2&lt;-1 5&lt;-2 6&lt;90%
   </str>
   <int name="ps">10</int>
   <str name="q.alt">*:*</str>
  </lst>
</requestHandler>
```

"edismax" query parser

- Extended dismax
- Supports full lucene query syntax in the absence of syntax errors.
- Improved proximity boosting via word bigrams.
- Supports the "boost" parameter like the dismax bf param, but multiplies the function query instead of adding it in for better scoring integration.
- Allows for wildcard searches, which dismax does not do.

"term" query parser

- {!term f=field_name}value
- Very useful for fq parameters where field value may contain Lucene query parser special characters!

Nested queries

- q=_*query*_:"{!dismax qf='author coauthor'}bob" AND _*query*_:"{!dismax qf='title subtitle'}testing"

Faceting

- field
- query
- range, numeric and date
- multi-select
- pivot
- cache impact

Field faceting

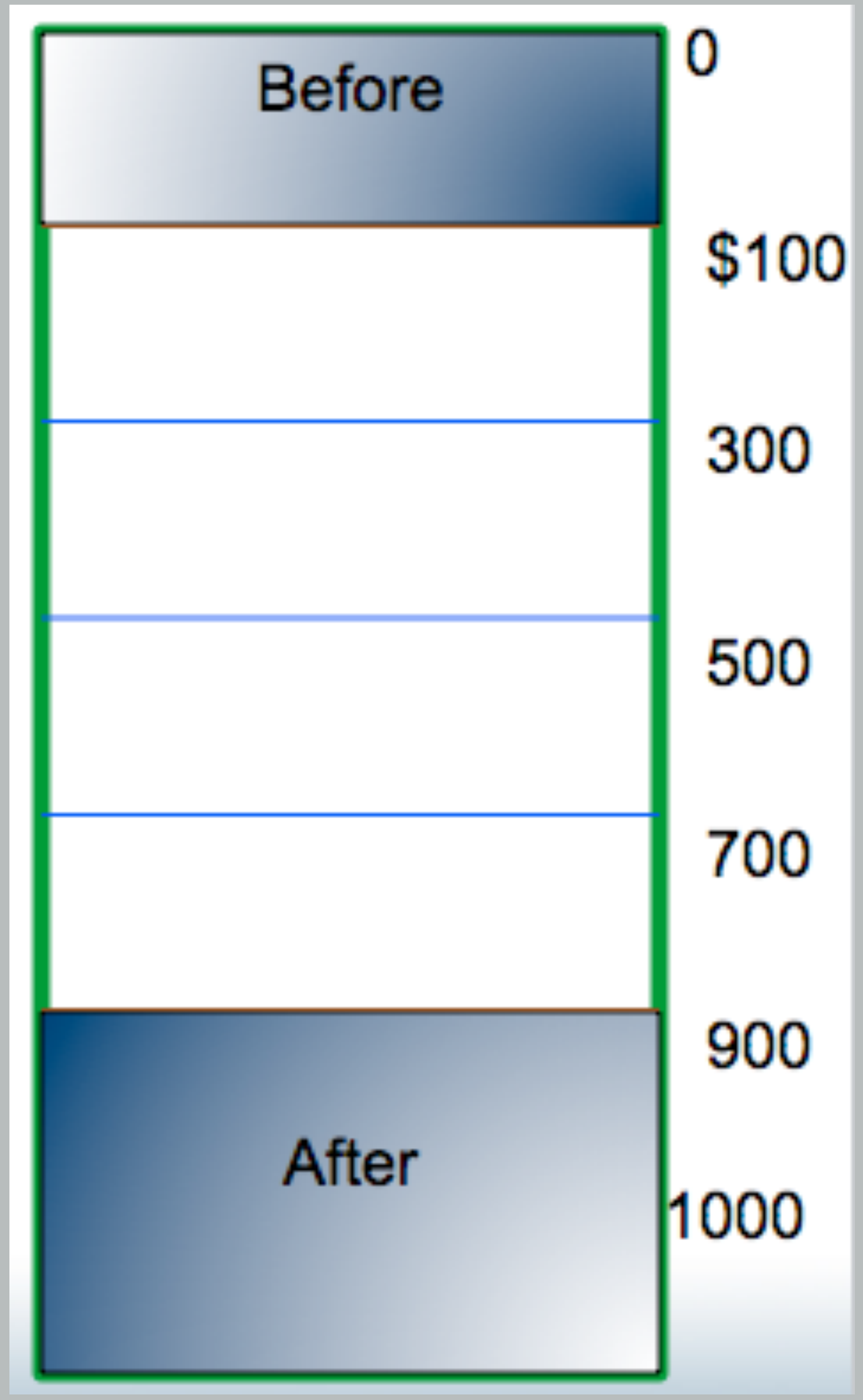
- facet.field=field_name

Query faceting

- facet.query=some query expression
- Default query parser: "lucene"
- Use {!parser ...} syntax to select different parser
- Use {!key=whatever} to have a nicer output key
- Example:
  - `facet.query={!key=cheapest}price:[0 TO 10]`

Range faceting

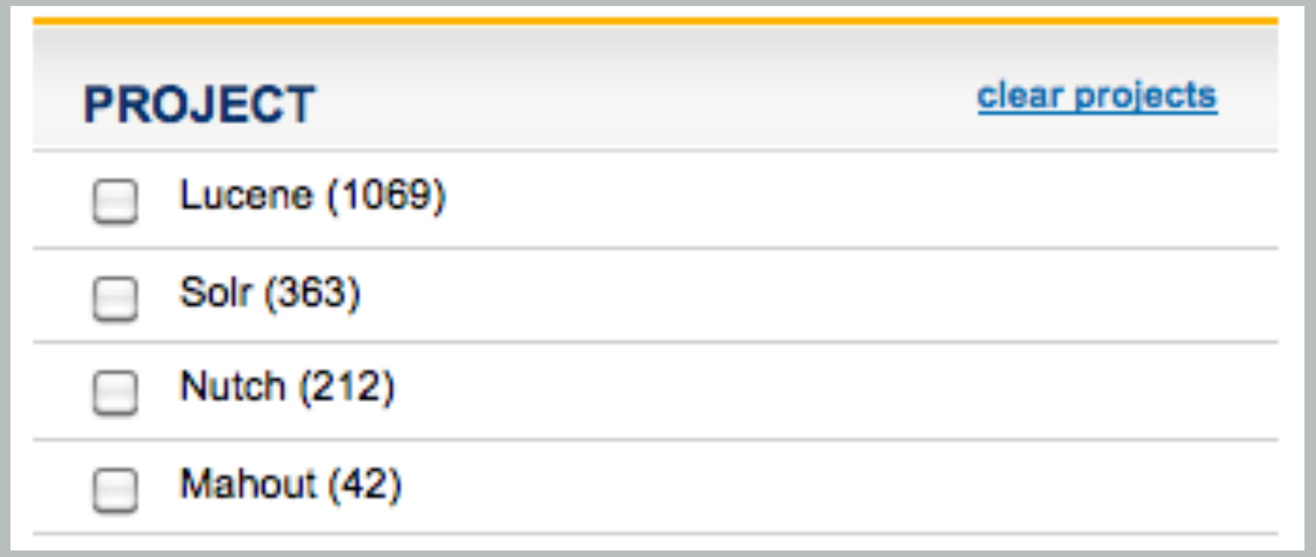
- Works for date and numeric field types
- Range facets divide a range into equal sized buckets.
- facet.range.start=100
- facet.range.end=900
- facet.range.gap=200
- facet.range.other=before
- facet.range.other=after



Multi-select faceting

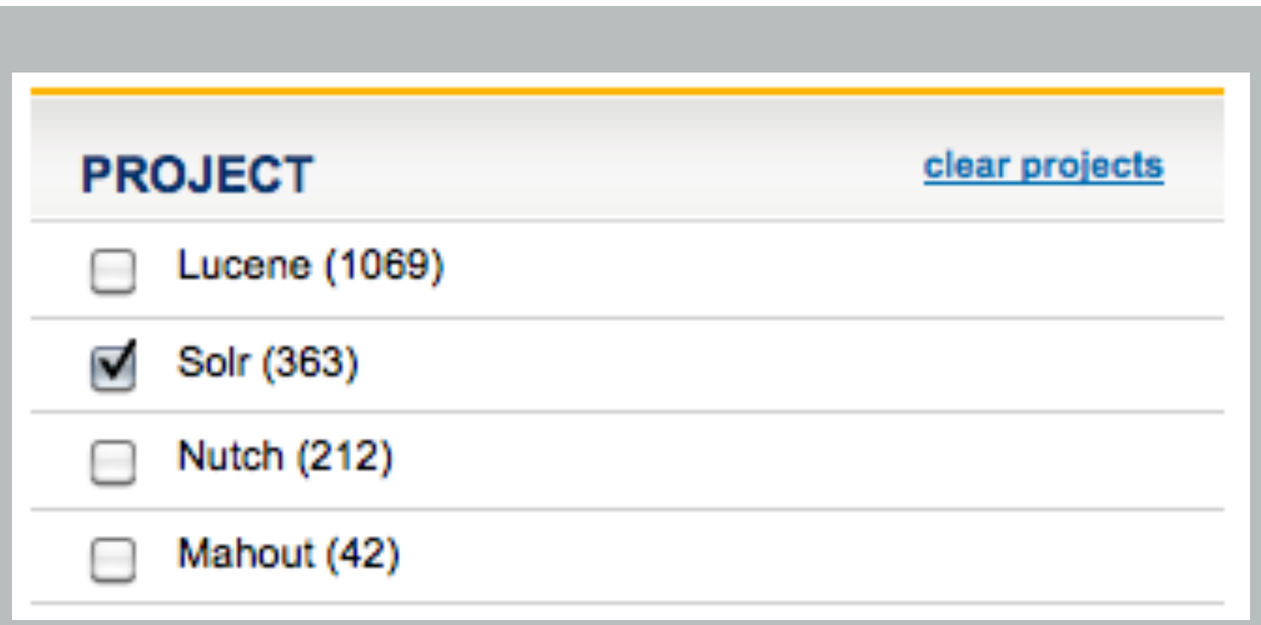
- Traditional faceting/filtering (facet.field=cat&fq=cat:electronics) narrows facet values to only those in result set
- Sometimes you want to allow multiple values and counts across all facet values

```
&fq={!tag=myTag}project:solr
&facet=on
&facet.field={!ex=myTag}project
```



PROJECT clear projects

□ Lucene (1069)
□ Solr (363)
□ Nutch (212)
□ Mahout (42)



PROJECT clear projects

□ Lucene (1069)
☑ Solr (363)
□ Nutch (212)
□ Mahout (42)

Pivot faceting

- Currently only available on trunk (Solr "4.0")
- facet.pivot=field1,field2,...
- facet counts within results of parent facet

Cache impact of faceting

- Faceting supports different internal algorithms / data structures, controlled through facet.method parameter
  - enum - enumerates all terms in a field, uses Solr's filterCache
  - fc - uses Lucene's FieldCache

Integration

- Prototyping
- Solr from ...
  - PHP
  - Ruby / RoR
  - Java
  - Ajax

Prototyping

- See earlier presentation(s)
- Don't overplan/overthink data ingestion and proving Solr out in your environment
- Just Do It
  - ingest your data in simplest possible way
  - fire up quick and not-so-dirty prototypical UI
  - iterate on config/ingestion/UI tweaks
  - go to scale: collection size, ingestion rate, query load

Solr from PHP

- There are wt=php|phps options
  - phps = serialized PHP structure
- Just use JSON though, why not?

Solr on Rails

- Blacklight - http://www.projectblacklight.org
- Flare
  - old school and dusty, but still got the right idea ;)
- Roll your own using Solr + Ruby APIs
  - solr-ruby
  - RSolr
  - Sunspot
  - and likely others
- Personal pet project: Prism -
  - https://github.com/lucidimagination/Prism - Solr with Sinatra, including JRuby/Velocity support!

SolrJ

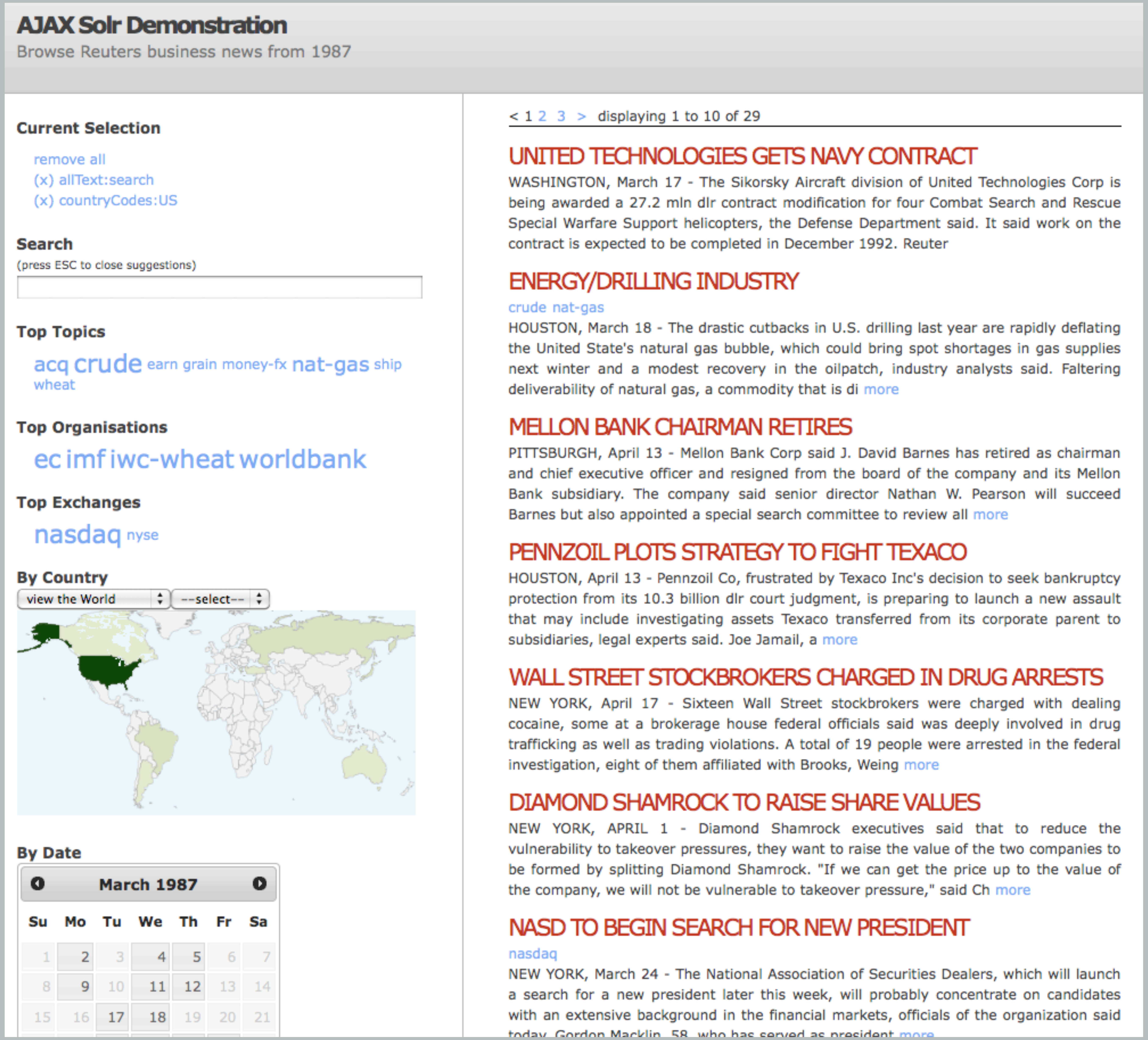
- When on the JVM, use SolrJ
- SolrServer abstraction
  - CommonsHttpSolrServer
  - StreamingUpdateSolrServer
  - LBHttpSolrServer
  - EmbeddedSolrServer

Ajax

- Careful! Generally you don't want Solr exposed to end users (**`<delete><query>*:*</query></delete>`** or worse !!!)
- wt=json
- But also consider remoting in partials generated from Velocity templates - keeps code out of UI

Ajax-Solr

- http://evolvingweb.github.com/ajax-solr/



AJAX Solr Demonstration

Browse Reuters business news from 1987

Current Selection

remove all
(x) allText:search
(x) countryCodes:US

Search

(press ESC to close suggestions)

Top Topics

acq crude earn grain money-fx nat-gas ship wheat

Top Organisations

ec imf iwc-wheat worldbank

Top Exchanges

nasdaq nyse

By Country

By Date

March 1987

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

< 1 2 3 > displaying 1 to 10 of 29

UNITED TECHNOLOGIES GETS NAVY CONTRACT

WASHINGTON, March 17 - The Sikorsky Aircraft division of United Technologies Corp is being awarded a 27.2 mln dlr contract modification for four Combat Search and Rescue Special Warfare Support helicopters, the Defense Department said. It said work on the contract is expected to be completed in December 1992. Reuter

ENERGY/DRILLING INDUSTRY

crude nat-gas

HOUSTON, March 18 - The drastic cutbacks in U.S. drilling last year are rapidly deflating the United State's natural gas bubble, which could bring spot shortages in gas supplies next winter and a modest recovery in the oilpatch, industry analysts said. Faltering deliverability of natural gas, a commodity that is di more

MELLON BANK CHAIRMAN RETIRES

PITTSBURGH, April 13 - Mellon Bank Corp said J. David Barnes has retired as chairman and chief executive officer and resigned from the board of the company and its Mellon Bank subsidiary. The company said senior director Nathan W. Pearson will succeed Barnes but also appointed a special search committee to review all more

PENNZOIL PLOTS STRATEGY TO FIGHT TEXACO

HOUSTON, April 13 - Pennzoil Co, frustrated by Texaco Inc's decision to seek bankruptcy protection from its 10.3 billion dlr court judgment, is preparing to launch a new assault that may include investigating assets Texaco transferred from its corporate parent to subsidiaries, legal experts said. Joe Jamail, a more

WALL STREET STOCKBROKERS CHARGED IN DRUG ARRESTS

NEW YORK, April 17 - Sixteen Wall Street stockbrokers were charged with dealing cocaine, some at a brokerage house federal officials said was deeply involved in drug trafficking as well as trading violations. A total of 19 people were arrested in the federal investigation, eight of them affiliated with Brooks, Weing more

DIAMOND SHAMROCK TO RAISE SHARE VALUES

NEW YORK, APRIL 1 - Diamond Shamrock executives said that to reduce the vulnerability to takeover pressures, they want to raise the value of the two companies to be formed by splitting Diamond Shamrock. "If we can get the price up to the value of the company, we will not be vulnerable to takeover pressure," said Ch more

NASD TO BEGIN SEARCH FOR NEW PRESIDENT

nasdaq

NEW YORK, March 24 - The National Association of Securities Dealers, which will launch a search for a new president later this week, will probably concentrate on candidates with an extensive background in the financial markets, officials of the organization said today. Gordon Macklin, 58, who has served as president more

Extras

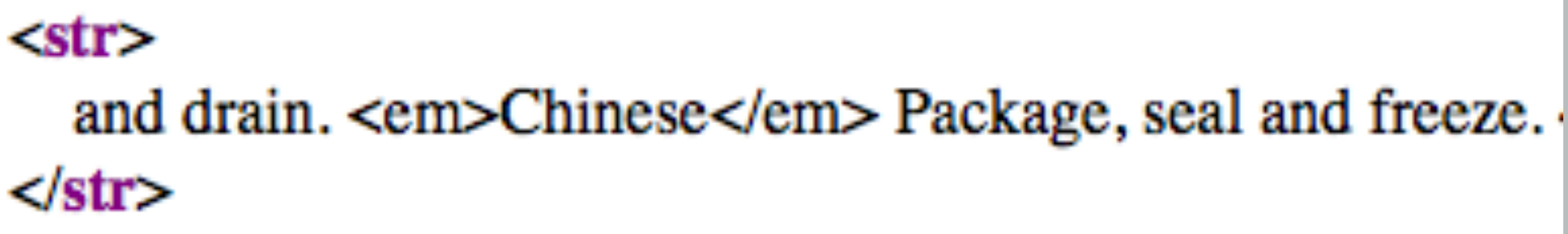
- Highlighting
- More-like-this
- Spell-checking / suggest
- Grouping
- Clustering
- Spatial

Highlighting

- Also known as keyword-in-context (KWIC)
- The highlighting feature adds pre/post highlight tags to the query terms found in stored document fields
- Note: because of stemming & synonyms, the words emphasized may not be what you typed into the search box. ‘change’ and ‘changing’ both stem to ‘chang’. If you type ‘change’ you might find documents with ‘changing’. The word ‘changing’ will be emphasized.

Highlighting example

- http://localhost:8983/solr/select/?q=text:chinese&hl=true&hl.fl=text&fl=id,score



```
<str>
   and drain. <em>Chinese</em> Package, seal and freeze.
</str>
```

MoreLikeThis

- More Like This is used to find similar documents. It might be used for suggestions: "If you liked this, then you may like that".
- Can be configured as either a component, or a request handler.
- Request handler is generally recommended because:
  - You don’t usually want to do this for every query.
  - You have a bit more control.
  - Minimum configuration:
    - <requestHandler name="/mlt" class="solr.MoreLikeThisHandler"/>

MLT params

- &mlt.fl – The field or fields to use for similarity (can’t be *)
- termVectors should be included for this field, but it’s not necessary.
- &mlt.mintf - Minimum Term Frequency - the frequency below which terms will be ignored in the source doc.
- Use mlt.mintf=1 for smaller fields, since the terms may not occur as much.
- &mlt.interestingTerms - will show what "interesting" terms are used for the MLT query.

MLT source terms

- &q=id:1234 – Will build a term list from terms in this document.
- &stream.body=lucene+scoring+algorithms – Will build a term list from the body streamed in.
- &stream.url=http://lucidimagination.com – Will build a term list from the content found at this URL.

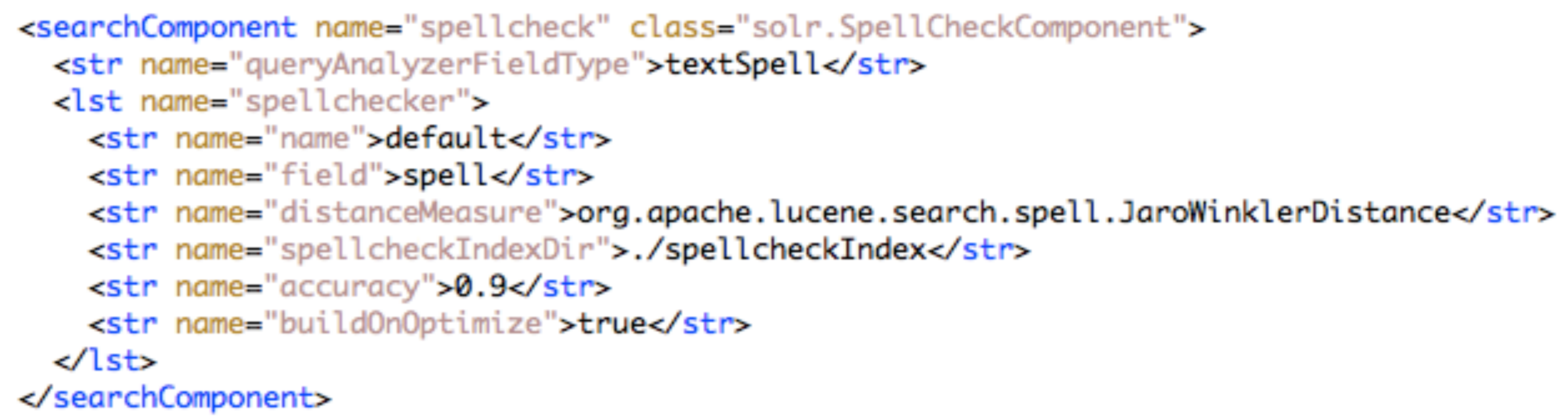
Spell checking

- Common feature often included with search applications.
- Did You Mean ...
- Takes a word and returns a set of similar words from a dictionary, searching for letter rearrangements.
- The N-Gram technique creates a set of (letter sequence -> term) pairs. The term with the most matching letter sequences is the most likely term.
- A separate "spelling dictionary" index must be created.

Spell checking dictionary

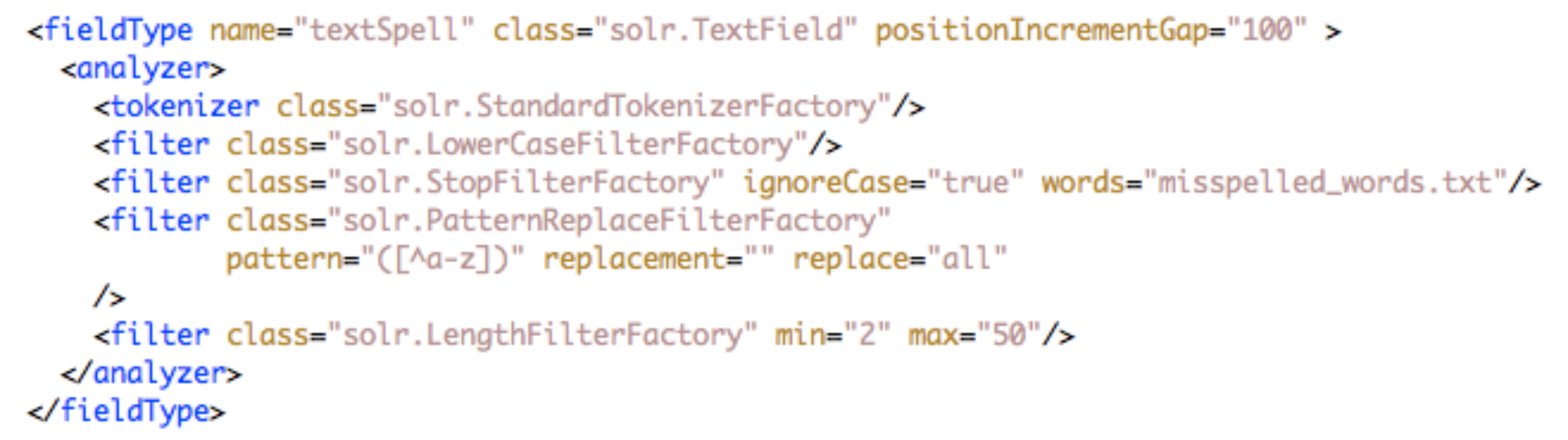
- The tools can use various sources as the spelling dictionary:
- File-based: A standard dictionary text file.
- Indexed data from the main index: A collection of common words harvested from the index via walking the terms for a field.
- The time for this process is linear with the size of the index.
- The terms must not be stemmed.
- The spell checking component must be configured in solrconfig.xml, which is where we specify whether to create the spelling index from a dictionary file or from terms in our main index.

Spell check config

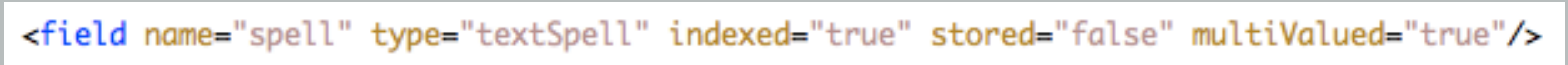


```
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">
  <str name="queryAnalyzerFieldType">textSpell</str>
  <lst name="spellchecker">
    <str name="name">default</str>
    <str name="field">spell</str>
    <str name="distanceMeasure">org.apache.lucene.search.spell.JaroWinklerDistance</str>
    <str name="spellcheckIndexDir">./spellcheckIndex</str>
    <str name="accuracy">0.9</str>
    <str name="buildOnOptimize">true</str>
  </lst>
</searchComponent>
```

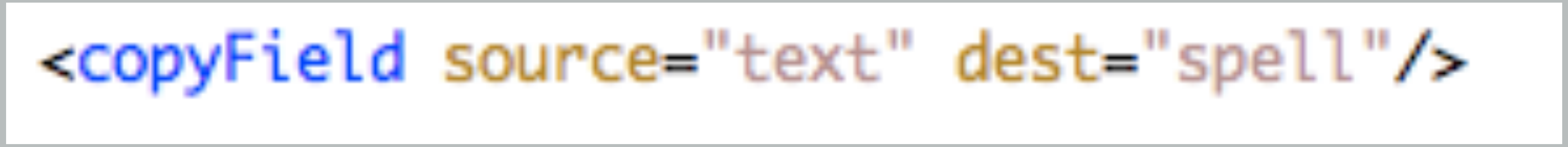
Spell check schema



```
<fieldType name="textSpell" class="solr.TextField" positionIncrementGap="100" >
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="misspelled_words.txt"/>
    <filter class="solr.PatternReplaceFilterFactory"
            pattern="([^a-z])" replacement="" replace="all"
    />
    <filter class="solr.LengthFilterFactory" min="2" max="50"/>
  </analyzer>
</fieldType>
```

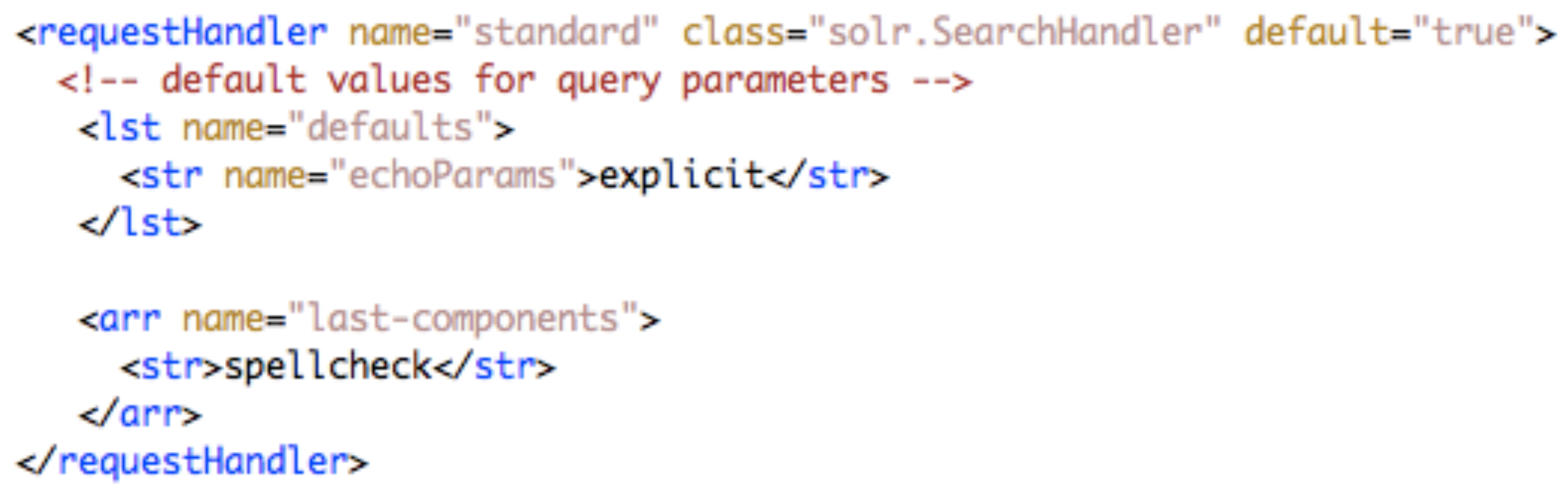


```
<field name="spell" type="textSpell" indexed="true" stored="false" multiValued="true"/>
```



```
<copyField source="text" dest="spell"/>
```

Spell check integration



```
<requestHandler name="standard" class="solr.SearchHandler" default="true">
  <!-- default values for query parameters -->
   <lst name="defaults">
     <str name="echoParams">explicit</str>
   </lst>

   <arr name="last-components">
     <str>spellcheck</str>
   </arr>
</requestHandler>
```

Spell check requests

- Sending requests to the SpellCheckComponent
- Some of the common parameters used for spell checking:
  - &spellcheck=true
  - &spellcheck.count=2
  - &spellcheck.onlyMorePopular=true
  - &spellcheck.collate=true

Suggest

- Various techniques:
  - TermsComponent
    - The following URL will display the top 20 terms for the title field:
    - http://localhost:8983/solr/terms/?terms.fl=title&terms=true&terms.limit=20
    - For auto-suggest add the param: &terms.prefix=at (where "at" are the two characters entered by the user.)
  - Use spell checking or facet.field/facet.prefix

Grouping

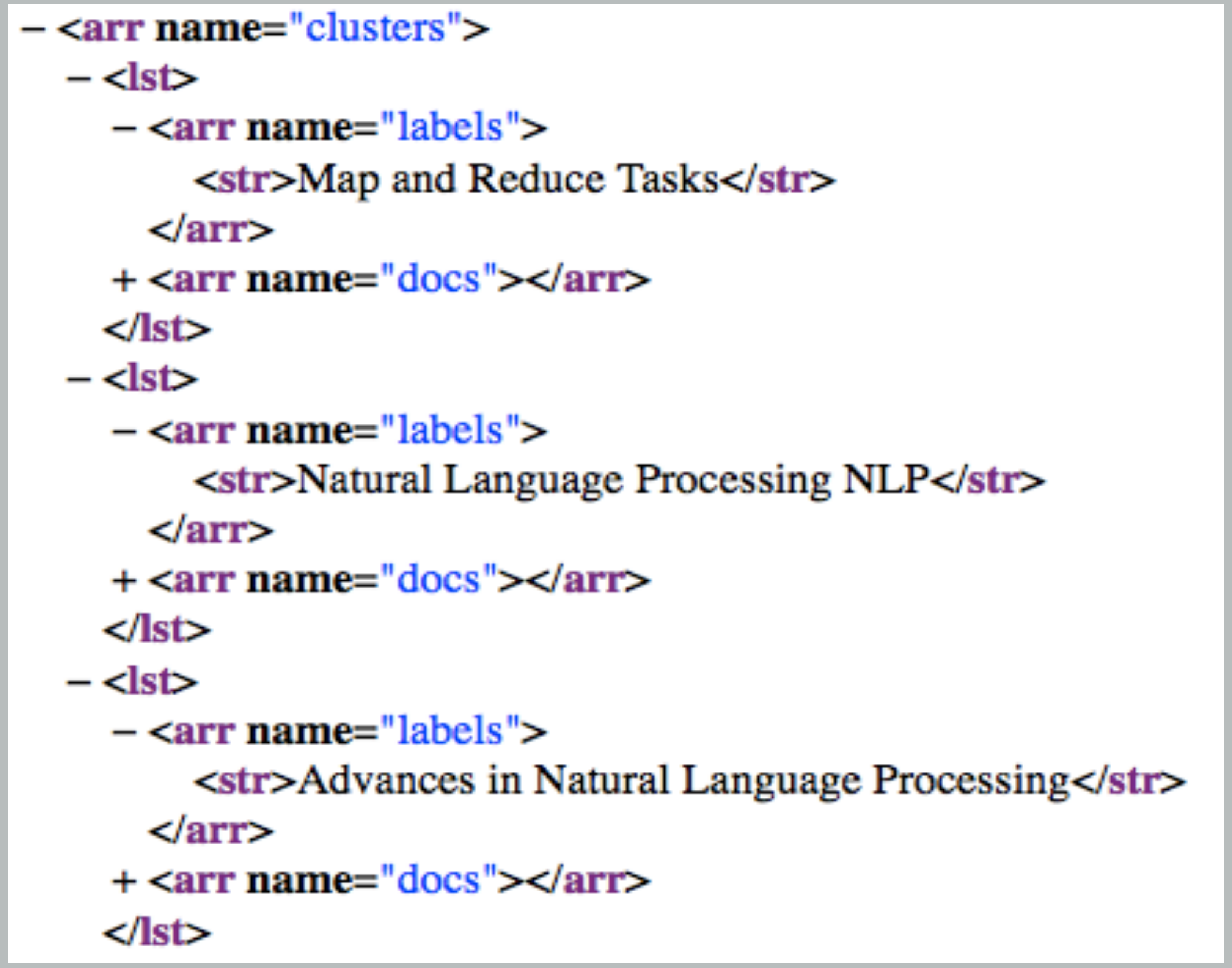
- Field Collapsing collapses a group of results with the same field value down to a single (or fixed number) of entries. For example, most search engines such as Google collapse on site so only one or two entries are shown. Field collapsing can also be used to suppress duplicate documents.
- Result Grouping groups documents with a common field value into groups, returning the top documents per group, and the top groups based on what documents are in the groups. One example is a search at Best Buy for a common term such as DVD, that shows the top 3 results for each category ("TVs & Video","Movies","Computers", etc)

http://wiki.apache.org/solr/FieldCollapsing

Clustering

- A Solr contrib module, the ClusteringComponent can cluster search results or documents in the index.
- Built with code from the Carrot2 open source project
- A way to group together results or documents.

search for "computer"



```
– <arr name="clusters">
  – <lst>
    – <arr name="labels">
        <str>Map and Reduce Tasks</str>
      </arr>
    + <arr name="docs"></arr>
    </lst>
  – <lst>
    – <arr name="labels">
        <str>Natural Language Processing NLP</str>
      </arr>
    + <arr name="docs"></arr>
    </lst>
  – <lst>
    – <arr name="labels">
        <str>Advances in Natural Language Processing</str>
      </arr>
    + <arr name="docs"></arr>
    </lst>
```

Spatial

- http://wiki.apache.org/solr/SpatialSearch
- Represent spatial data in the index
- Filter - bounding box, distance
- Sort by distance
- Score/boost by distance
- LatLongType:
  - <field name="location">45.17614,-93.87341</field>
- geofilt bbox query parsers
- geodist function

Scoring / relevance

- Explaining
- Scoring formula
- Boosting
- Function queries

Tools

- Solr Admin Console
- Luke

Testing

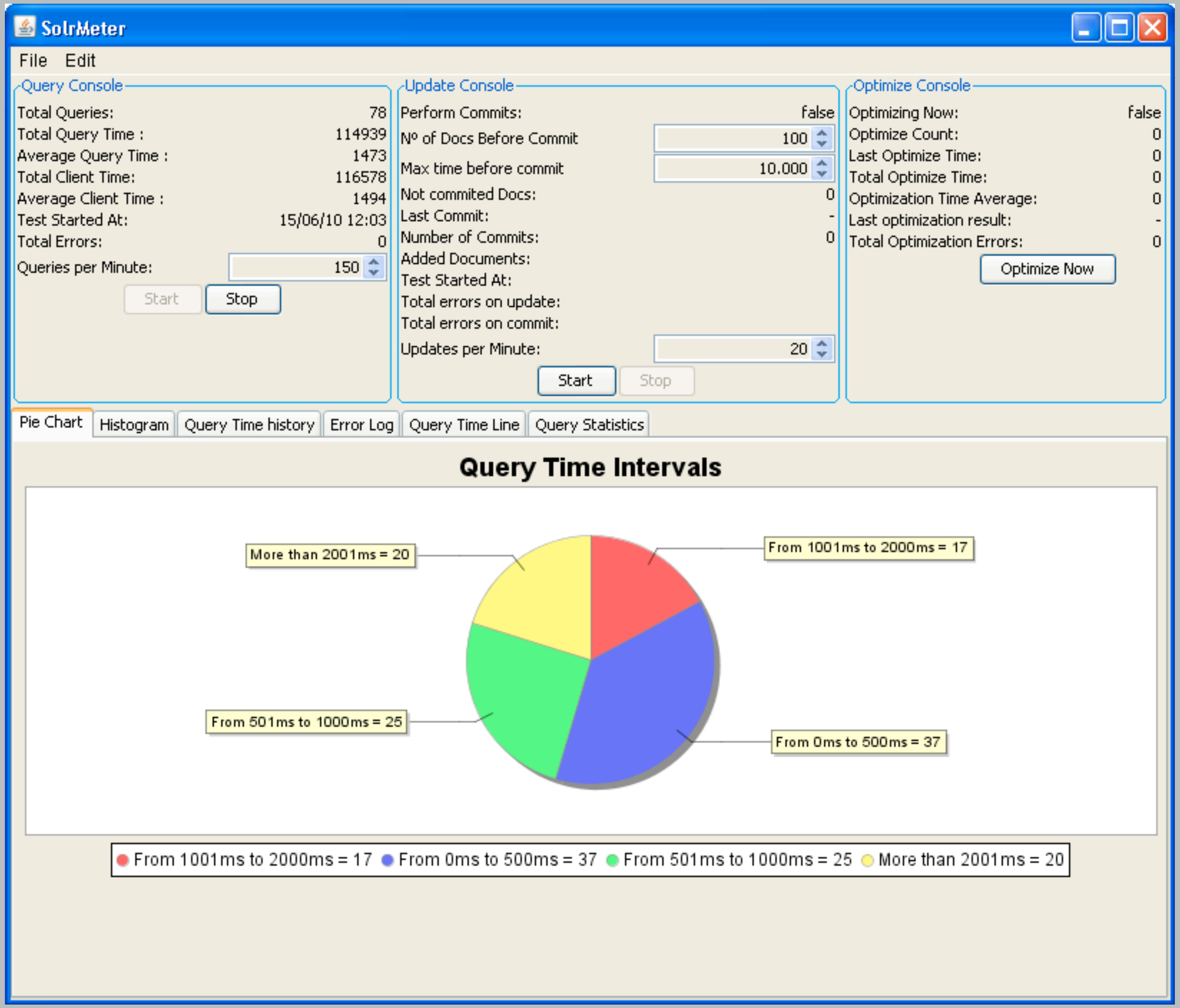
- Solr testing infrastructure
- SolrMeter

Solr's built-in test base

- AbstractSolrTestCase
- SolrTestCaseJ4

SolrMeter

- http://code.google.com/p/solrmeter/



Query Console

Total Queries: 78
Total Query Time : 114939
Average Query Time : 1473
Total Client Time: 116578
Average Client Time : 1494
Test Started At: 15/06/10 12:03
Total Errors: 0
Queries per Minute: 150

Start Stop

Update Console

Perform Commits: false
Nº of Docs Before Commit 100
Max time before commit 10.000
Not commited Docs: 0
Last Commit: -
Number of Commits: 0
Added Documents:
Test Started At:
Total errors on update:
Total errors on commit:
Updates per Minute: 20

Start Stop

Optimize Console

Optimizing Now: false
Optimize Count: 0
Last Optimize Time: 0
Total Optimize Time: 0
Optimization Time Average: 0
Last optimization result: -
Total Optimization Errors: 0

Optimize Now

Query Time Intervals

More than 2001ms = 20
From 1001ms to 2000ms = 17
From 501ms to 1000ms = 25
From 0ms to 500ms = 37
● From 1001ms to 2000ms = 17 ● From 0ms to 500ms = 37 ● From 501ms to 1000ms = 25 ● More than 2001ms = 20

For more information...

- http://www.lucidimagination.com
- LucidFind
  - search Lucene ecosystem: mailing lists, wikis, JIRA, etc
  - http://search.lucidimagination.com
- Getting started with LucidWorks Enterprise:
  - http://www.lucidimagination.com/products/lucidworks-search-platform/enterprise
- http://lucene.apache.org/solr - wiki, e-mail lists

Thank You!

lucid

IMAGINATION