Tutorial - BaseX and Lucene

XQuery

XQuery is to XML what SQL is to database tables

XQuery is a language for finding and extracting elements and attributes from XML documents

XPath is a language used to succinctly pinpoint exact XML nodes in a DOM
XQuery is a superset of XPath that also provides FLWOR syntax

FLWOR Expressions

for part is like the SELECT part in a SQL query
let lets you create subselections or values in (temporary) variables
where part narrows down the data you want to retrieve
order by part sorts the results
return part can be used to format the XML fragment

Example

Find all movies with rating > 7.5

```
for $x in collection("Movies")/root/movie
where $x/rating/imdbRating > 7.5
order by $x/@title
return <results>{$x/@title}</results>
```

If-Then-Else

```
for $x in collection("Movies")/root/movie
order by $x/@title
return if ($x/rating/imdbRating > 7.5)
then <good><name>{$x/@title}</name></good>
else <bad><name>{$x/@title}</name></bad>
```

For clause

```
for $x in (1 to 5)
return <test>{$x}</test>

for $x at $i in collection("Movies")/root/movie
return <results>{$i}. <name>{$x/@title}</name> </results>
```

Functions

Upper Case

```
for $x in collection("Movies")/root/movie
return <results> <name>{upper-case($x/@title)}</name> </results>
```

Substring

```
for $x in collection("Movies")/root/movie
return <results> <name>{substring($x/@title, 1, 4)}</name> </results>
```

```
declare function local:minPrice($p as xs:decimal?,$d as xs:decimal?)
as xs:decimal?
{
let $disc := ($p * $d) div 100
return ($p - $disc)
};
```

Below is an example of how to call the function above:

```
<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>
```

Why Lucene ?

Easy for searching in documents.
For example consider how search engine works ?.

Apache solr Lucene supports (File endings considered are xml,json,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,html,txt,log)

Start solr Lucene

```
## start solar
$ bin/solr start              # this starts solr
$ bin/solr create -c demo   # this creates a document collection called "demo"
```

1) Web browser - URL http://localhost:8983/solr/ - (just for understanding)
2) Access through terminal using curl or python - we will use python

```
from urllib2 import *
import simplejson
connection = urlopen(url)
response = simplejson.load(connection)
Print response
```

Indexing and Retrieving a Document

```
bin/post -c demo m2016/
```

Dynamic Fields:

Use “schemaless” mode - by default lucene guesses field types.
Use “schema” mode - has specific schema - (i.e) we can add extra fields
Which can be made through schema.xml file or schema API

Querying:

Let’s say our schema has the following fields depending on we decide how to query

id,cat_s,pubyear_i,title_t,author_s,series_s,sequence_i,publisher_s

URL defining and it’s parameters

URL = ‘http://localhost:8983/solr/films/select?indent=on&q=*:*&rows=1100&start=0&wt=json’

Example : http://localhost:8983/solr/demo/query?q=title_t:blackfl=author_s,title_t

What format we can send request parameters ? (JSON)
 q = search value (*:*)
 fl = field list (required fields can be shown) - fl=title_t,pubyear_i'
What format do we get results ? json / xml etc
Sorting and Paging Search Results - rows=3&sort=pubyear_i desc
curl http://localhost:8983/solr/demo/query -d '
q=*:*&fq=publisher_s:Bantam&rows=3&sort=pubyear_i desc&fl=title_t,pubyear_i'

Parameter Explanation

q=*:* – The *:* query matches all documents in the index.

fq=publisher_s:Bantam – “fq” parameters are filter queries

sort=pubyear_i desc – This sorts on the “pubyear_i” field descending.

rows=3 – “rows” specifies the number of results to return.

Queries

Basic Queries - A “term” query is a single word query in a single field that must match exactly.

    q = text:hello

Phrase Query - A phrase query matches multiple terms (words) in sequence.

    q = text:"yonik seeley"

Proximity Query - A proximity query, is like a phrase query with a tilda (~) followed by a slop that specifies the number of term position moves (edits) allowed.

    q = text:"solr analytics"~1

Boolean Query - A boolean query contains multiple clauses.

    q = solr search   (OR condition) (returns values matching atleast one clause)

Boosted Query - Any query clause can be boosted with the ^ operator.

    text:solr^10 text:rocks

Range Query - Range queries work on numeric fields, date fields, and even string and text fields.

    age:[18 TO 30] name:[apple TO banana]

Filter Query , Query Comments

TASK

How do we start using lucene for moviesText folder containing text files ? .

For now we use schemaless fields and post text documents to the core directory and perform boolean query.

Exercise: **Perform Lucene search by using specific schema and run the queries like boolean query ,filter query, Proximity Query, Base Query, Boosted Query, Range Query.**

Reference Links

http://yonik.com/solr-tutorial/

http://yonik.com/solr/query-syntax/

https://cwiki.apache.org/confluence/display/solr/About+This+Guide