Indexing HTML files in Solr[1]

This tutorial explains how to index html files in Solr using the built-in post tool, which leverages Apache Tika and auto extracts content from html files. You should have already downloaded and installed Solr, see

https://solr.apache.org/guide/7_7/installing-solr.html

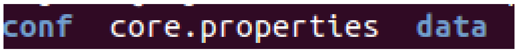
Apache Tika is a library that is used for document type detection and content extraction from various file formats. Internally, Tika uses various existing document parsers and document type detection techniques to detect and extract data. For example, in case of HTML pages, Tika uses its HTMLParser to strip out all html tags and only stores the content from the html pages. Tika is a powerful tool and is very useful when you have crawled various types of documents, e.g. PDF, Images, Videos. Tika is included with the Solr installation.

Using post tool and TIKA

1. Start the Solr server, cd into solr-8.x.x folder and enter the command: ***bin/solr start***

2. Create a new core, in this tutorial the core name is **myexample** using the command: **bin/solr create –c myexample** After the core is created you should be able to see the message – "Created new core 'myexample'". Verify that your new core is created in solr-8.x.x/server/solr/ with the name myexample.

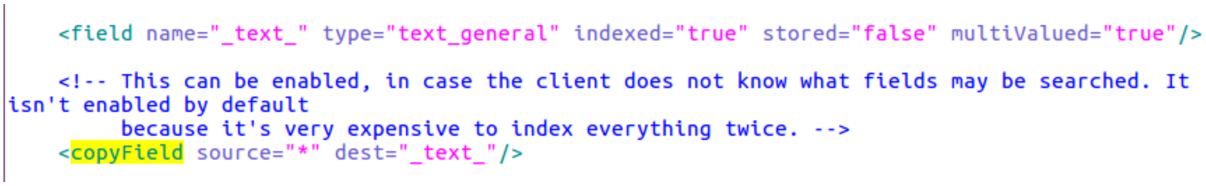
3. Let us examine the directory structure of this folder. cd into the folder myexample in server/solr. There will be a **conf** folder, **data** folder and a **core.properties** file. The conf folder contains the managed-schema file and the solrconfig.xml file, these are the files that have to be modified for specific field level analysis during indexing and querying. For example you can specify that a field from your document has to be tokenized using white space delimiter, whereas another field has to be tokenized into n-grams, depending on how you want to index and later query these fields. You can specify stop words in stopwords.txt file which would enable Solr to eliminate these words during indexing. The data folder contains the indexed data and logs generated by Solr. The core.properties file contains metadata information about the core, e.g, the core name, the directory name where the core data is stored.



```
conf  core.properties  data
```

4. Solr inherently uses Tika for extracting content from the documents that will be indexed. Tika uses the TagSoup library to support virtually any kind of HTML found on the web. The output from the HtmlParser class is used as the streamed content for indexing into Solr. Before indexing html files, we have to edit the schema file to make sure that all the text content from the html pages extracted by Tika are mapped correctly. To do this, go to the conf folder of the core "myexample". You will see a file named – "**managed-schema"**. Now open this xml to edit it. Below is a portion of the managed-schema file.

Uncomment the following line - <!-- <copyField source="*" dest="_text_"/> -->, so it looks like below -



```
    <field name="_text_" type="text_general" indexed="true" stored="false" multiValued="true"/>

    <!-- This can be enabled, in case the client does not know what fields may be searched. It
isn't enabled by default
         because it's very expensive to index everything twice. -->
    <copyField source="*" dest="_text_"/>
```

[1] This tutorial was authored by Majisha Parambath (with a little help from Prof. Horowitz)

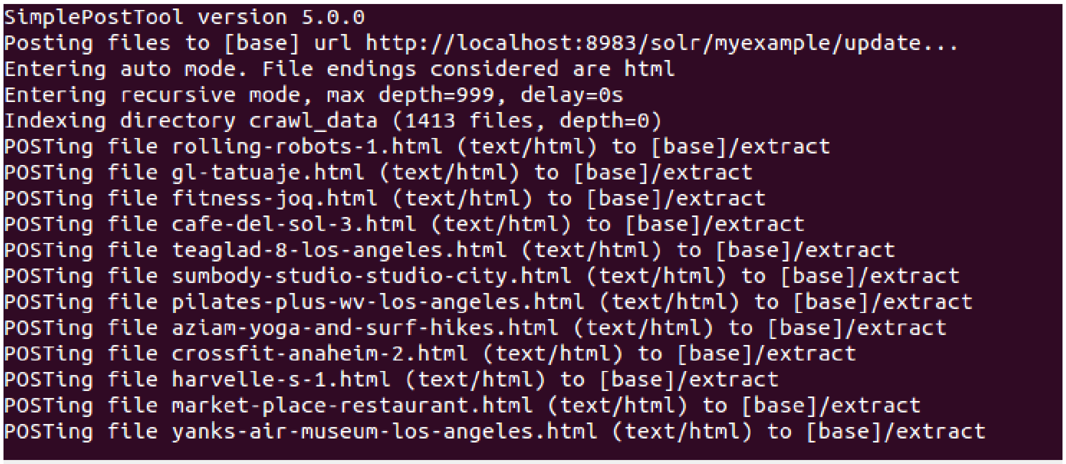
Fields are defined in the field element of managed-schema as shown in the above figure. These are field definitions for the documents that you will be indexing, and each document that gets indexed will have these fields. You can define various properties for the fields, such as “indexed” if set to true, then the content in this field will be indexed, “stored” if set then the content of this field is stored in Solr and we can retrieve this from the query responses, “required” indicates whether this field is mandatory in a document for indexing, multivalued is true meaning it can appear multiple times in a document. In the current managed-schema version, an “id” field, “_version_” and “_root_” fields are auto generated while indexing. These may not necessarily be part of your original documents.

Solr has a mechanism for making copies of fields so that you can apply several distinct field types to a single piece of incoming information. You can see that the copyField element copies all the fields into the destination field “_text_”. The “_text_” field is also only indexed and not stored because there is no need to have redundant data. Basically what we are trying to accomplish here, is to combine all the information from different fields into a single field, “_text_”.

5. Save your changes and go back to the home directory of Solr. Let us try to index some html pages. In this example, I have crawled the webpages from losangeles.eventful.com and stored the crawled pages in a folder named “crawl_data”. You should be using the crawl data folder for the specific news website you are responsible for (based on your USC ID as given in Solr Exercise document). Indexing is performed by using the command:
bin/post –c <core_name> -filetypes html <path_to_crawl_folder>/
The filetypes option specified above implies we are only indexing html files. If we are indexing various document types together, we can ignore the filetypes option, then the command will be simply:
bin/post –c <core_name> <path_to_crawl_folder>/

The crawl folder in this example is “crawl_data”, the core to which I am indexing is “myexample” and I am going to index only html files, therefore the command will be:
bin/post –c myexample –filetypes html crawl_data/

You should see output similar to the one below if the command was successful:



```
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/myexample/update...
Entering auto mode. File endings considered are html
Entering recursive mode, max depth=999, delay=0s
Indexing directory crawl_data (1413 files, depth=0)
POSTing file rolling-robots-1.html (text/html) to [base]/extract
POSTing file gl-tatuaje.html (text/html) to [base]/extract
POSTing file fitness-joq.html (text/html) to [base]/extract
POSTing file cafe-del-sol-3.html (text/html) to [base]/extract
POSTing file teaglad-8-los-angeles.html (text/html) to [base]/extract
POSTing file sumbody-studio-studio-city.html (text/html) to [base]/extract
POSTing file pilates-plus-wv-los-angeles.html (text/html) to [base]/extract
POSTing file aziam-yoga-and-surf-hikes.html (text/html) to [base]/extract
POSTing file crossfit-anaheim-2.html (text/html) to [base]/extract
POSTing file harvelle-s-1.html (text/html) to [base]/extract
POSTing file market-place-restaurant.html (text/html) to [base]/extract
POSTing file yanks-air-museum-los-angeles.html (text/html) to [base]/extract
```

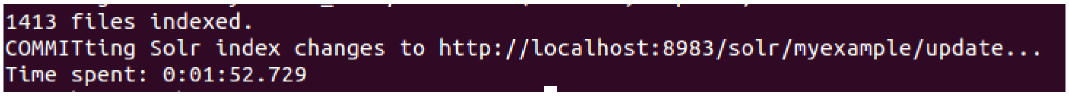
NOTE – If you are getting an error similar to – “SimplePostTool: WARNING: IOException while reading response: java.io.FileNotFoundException: http://localhost:8983/solr/myexample/update/extract?resource.name=%2Fhome%2Fhw4%2Fnytimes%2F39f07542-ee59-4254-ade2-0d393aa7e360.html&literal.id=%2Fhome%2Fhw4%2Fnytimes%2F39f07542-ee59-4254-ade2-0d393aa7e360.html” for almost all the HTML files (if you are getting the error only for a few urls, it’s fine), then you can either downgrade to Solr version 7.7.2, or follow the below steps.

➢ Add the below code to solrconfig.xml –

```
<lib dir="../../extract" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/contrib/extraction/lib" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/contrib/dataimporthandler/lib/" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-dataimporthandler-.*\.jar" />
```

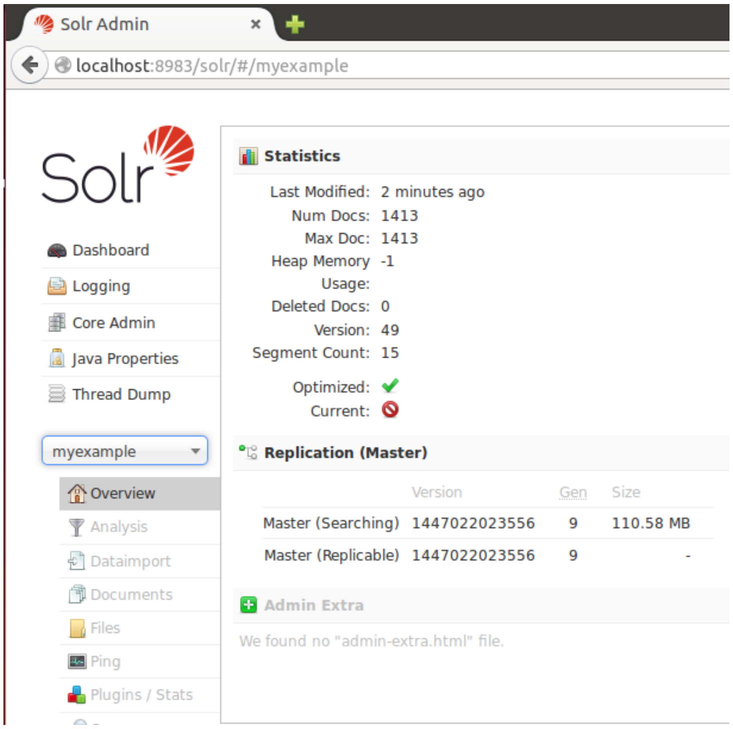
```
<lib dir="${solr.install.dir:../../../..}/contrib/extraction/lib" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-cell-\d.*\.jar" />
<lib dir="${solr.install.dir:../../../..}/contrib/clustering/lib/" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-clustering-\d.*\.jar" />
<lib dir="${solr.install.dir:../../../..}/contrib/langid/lib/" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-langid-\d.*\.jar" />
<lib dir="${solr.install.dir:../../../..}/contrib/velocity/lib" regex=".*\.jar" />
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-velocity-\d.*\.jar" />
<requestHandler name="/update" class="solr.UpdateRequestHandler">
 </requestHandler>
 <requestHandler name="/update/extract"
                startup="lazy"
                class="solr.extraction.ExtractingRequestHandler" >
  <lst name="defaults">
    <str name="lowernames">true</str>
    <str name="uprefix">ignored_</str>
    <!-- capture link hrefs but ignore div attributes -->
    <str name="captureAttr">true</str>
    <str name="fmap.a">links</str>
    <str name="fmap.div">ignored_</str>
  </lst>
 </requestHandler>
```

After making the above changes to the file, save the file. Restart solr with “bin/solr restart” command and rerun the previous command ***bin/post –c myexample –filetypes html crawl_data/***
After all the files are indexed, solr will auto commit the index and you should see the following output.



```
1413 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/myexample/update...
Time spent: 0:01:52.729
```

6. So now we can check the Solr UI and see if the files have gotten indexed. Open the browser and go to http://localhost:8983/solr/. Select the core “myexample” from the dropdown. You should see an output showing the statistics similar to the figure below.



Statistics

```
      Last Modified:  2 minutes ago
           Num Docs:  1413
            Max Doc:  1413
        Heap Memory   -1
             Usage:
       Deleted Docs:  0
            Version:  49
      Segment Count:  15

          Optimized:  ✓
            Current:  🛇
```

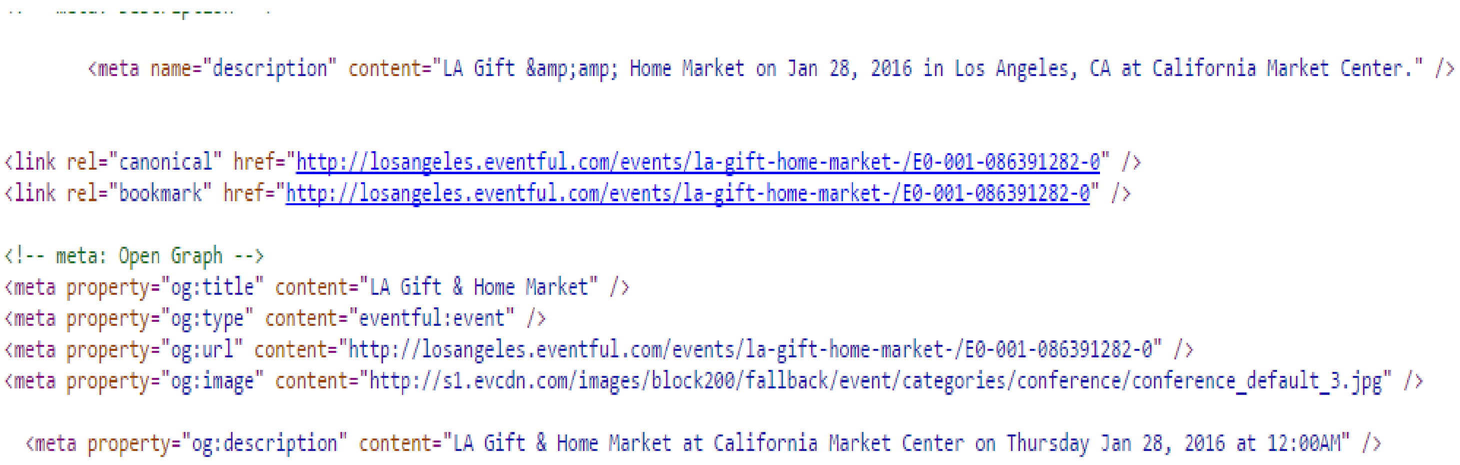
Replication (Master)

	Version	Gen	Size
Master (Searching)	1447022023556	9	110.58 MB
Master (Replicable)	1447022023556	9	-

Admin Extra

We found no "admin-extra.html" file.

7. Let us see how the TIKA parser parsed the html pages and what fields were created by using TIKA. Select the query option, and just submit the default query (*:*). Below is a snapshot of a portion of one of the web pages from losangeles.eventful.com followed by the results produced by the TIKA parser. Some notable fields extracted here are title, latitude, longitude, description etc. We can see the HTML page source from where the fields were extracted below.



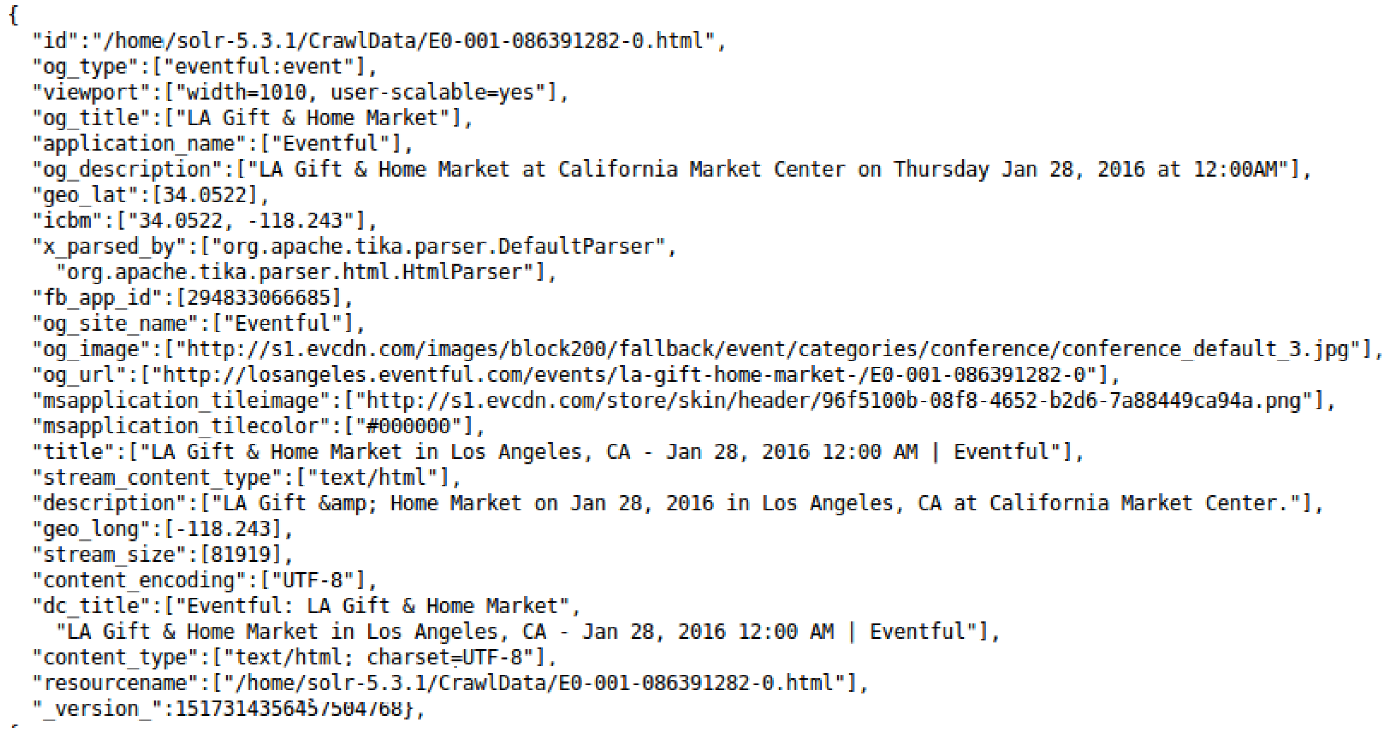
```
    <meta name="description" content="LA Gift &amp;amp; Home Market on Jan 28, 2016 in Los Angeles, CA at California Market Center." />


<link rel="canonical" href="http://losangeles.eventful.com/events/la-gift-home-market-/E0-001-086391282-0" />
<link rel="bookmark" href="http://losangeles.eventful.com/events/la-gift-home-market-/E0-001-086391282-0" />

<!-- meta: Open Graph -->
<meta property="og:title" content="LA Gift & Home Market" />
<meta property="og:type" content="eventful:event" />
<meta property="og:url" content="http://losangeles.eventful.com/events/la-gift-home-market-/E0-001-086391282-0" />
<meta property="og:image" content="http://s1.evcdn.com/images/block200/fallback/event/categories/conference/conference_default_3.jpg" />

  <meta property="og:description" content="LA Gift & Home Market at California Market Center on Thursday Jan 28, 2016 at 12:00AM" />
```

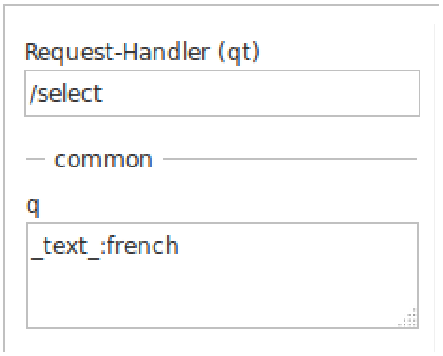
The following figure is snipped from the response of the default query in the SolrUI. Notice that the response uses the JSON format. Also notice that all meta properties in the html are preserved. Notice that some key : value pairs are autogenerated by Tika, e.g. content_encoding, content_type, x_parsed_by, stream_size, and stream_content_type.



```
{
  "id":"/home/solr-5.3.1/CrawlData/E0-001-086391282-0.html",
  "og_type":["eventful:event"],
  "viewport":["width=1010, user-scalable=yes"],
  "og_title":["LA Gift & Home Market"],
  "application_name":["Eventful"],
  "og_description":["LA Gift & Home Market at California Market Center on Thursday Jan 28, 2016 at 12:00AM"],
  "geo_lat":[34.0522],
  "icbm":["34.0522, -118.243"],
  "x_parsed_by":["org.apache.tika.parser.DefaultParser",
    "org.apache.tika.parser.html.HtmlParser"],
  "fb_app_id":[294833066685],
  "og_site_name":["Eventful"],
  "og_image":["http://s1.evcdn.com/images/block200/fallback/event/categories/conference/conference_default_3.jpg"],
  "og_url":["http://losangeles.eventful.com/events/la-gift-home-market-/E0-001-086391282-0"],
  "msapplication_tileimage":["http://s1.evcdn.com/store/skin/header/96f5100b-08f8-4652-b2d6-7a88449ca94a.png"],
  "msapplication_tilecolor":["#000000"],
  "title":["LA Gift & Home Market in Los Angeles, CA - Jan 28, 2016 12:00 AM | Eventful"],
  "stream_content_type":["text/html"],
  "description":["LA Gift &amp; Home Market on Jan 28, 2016 in Los Angeles, CA at California Market Center."],
  "geo_long":[-118.243],
  "stream_size":[81919],
  "content_encoding":["UTF-8"],
  "dc_title":["Eventful: LA Gift & Home Market",
    "LA Gift & Home Market in Los Angeles, CA - Jan 28, 2016 12:00 AM | Eventful"],
  "content_type":["text/html; charset=UTF-8"],
  "resourcename":["/home/solr-5.3.1/CrawlData/E0-001-086391282-0.html"],
  "_version_":1517314356457504768},
```

TIKA extracted some fields automatically from these pages.

8. We can also query using the hidden field "_text_". In the following example I am going to search in the _text_ field for "French".

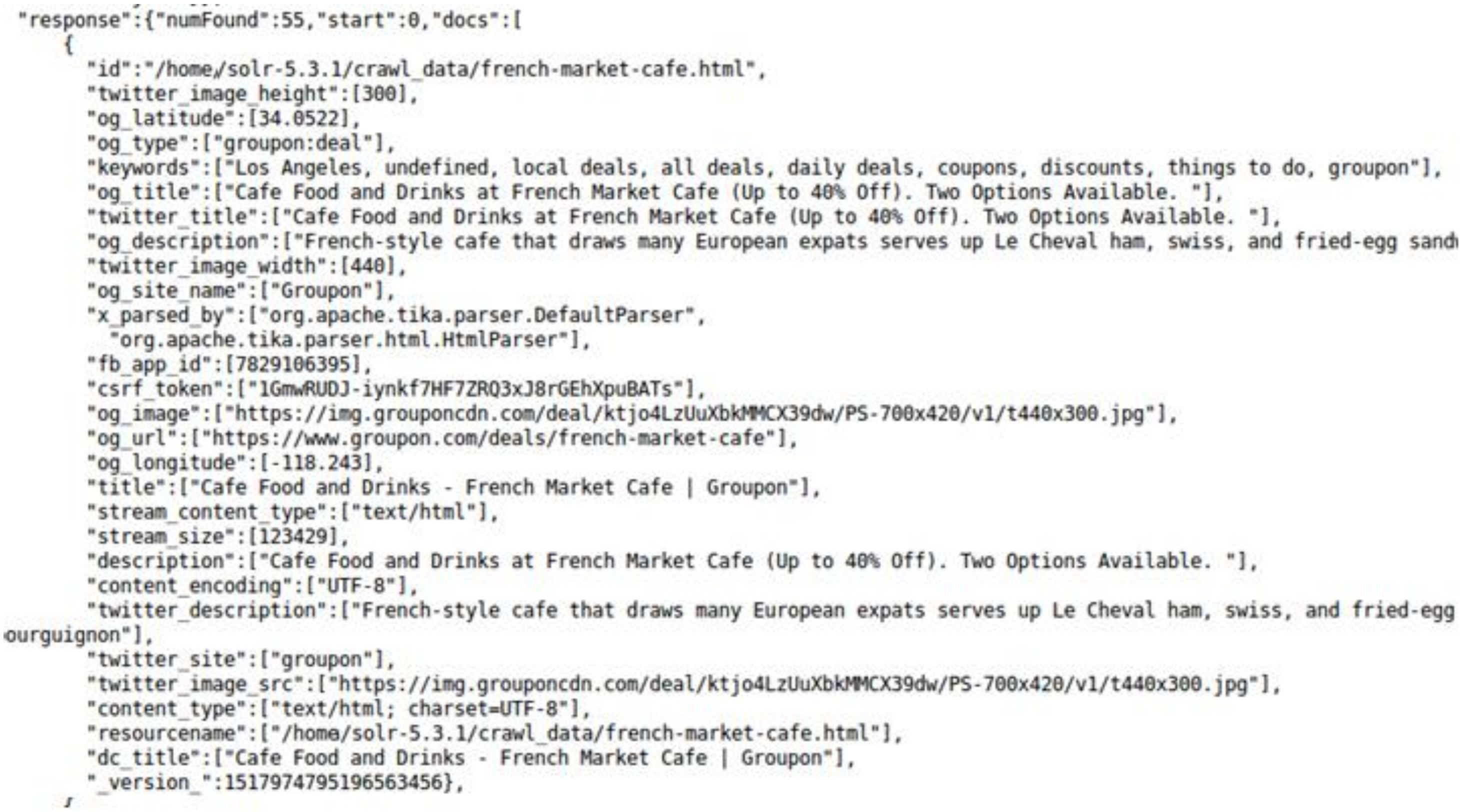


/select

q

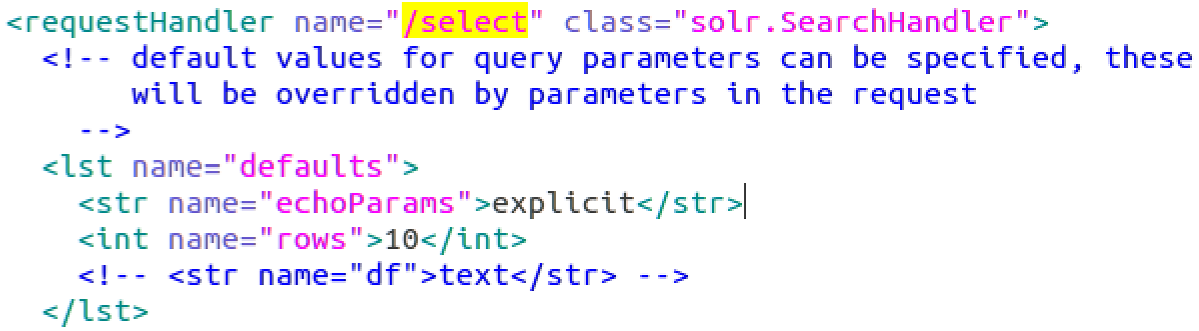
text:french

The query returned the following results. Looking at the JSON we see 55 documents were matched and returned.



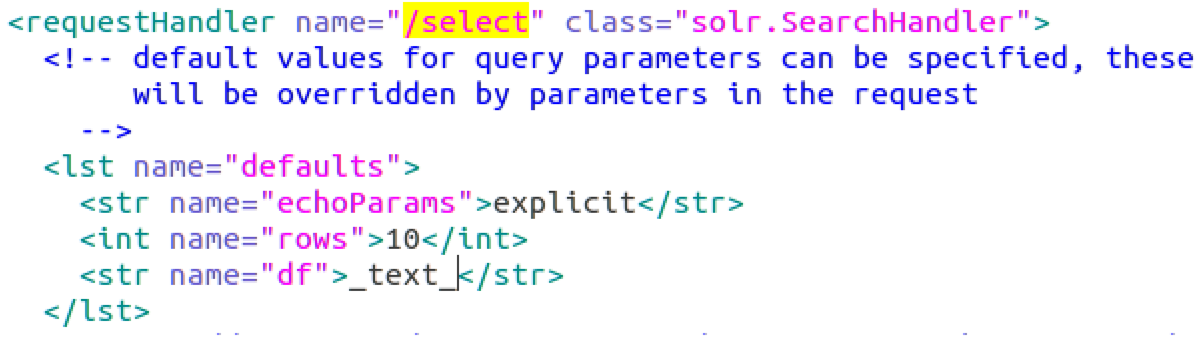
```
"response":{"numFound":55,"start":0,"docs":[
    {
      "id":"/home/solr-5.3.1/crawl_data/french-market-cafe.html",
      "twitter_image_height":[300],
      "og_latitude":[34.0522],
      "og_type":["groupon:deal"],
      "keywords":["Los Angeles, undefined, local deals, all deals, daily deals, coupons, discounts, things to do, groupon"],
      "og_title":["Cafe Food and Drinks at French Market Cafe (Up to 40% Off). Two Options Available. "],
      "twitter_title":["Cafe Food and Drinks at French Market Cafe (Up to 40% Off). Two Options Available. "],
      "og_description":["French-style cafe that draws many European expats serves up Le Cheval ham, swiss, and fried-egg sandi
      "twitter_image_width":[440],
      "og_site_name":["Groupon"],
      "x_parsed_by":["org.apache.tika.parser.DefaultParser",
        "org.apache.tika.parser.html.HtmlParser"],
      "fb_app_id":[7829106395],
      "csrf_token":["1GmwRUDJ-iynkf7HF7ZRQ3xJ8rGEhXpuBATs"],
      "og_image":["https://img.grouponcdn.com/deal/ktjo4LzUuXbkMMCX39dw/PS-700x420/v1/t440x300.jpg"],
      "og_url":["https://www.groupon.com/deals/french-market-cafe"],
      "og_longitude":[-118.243],
      "title":["Cafe Food and Drinks - French Market Cafe | Groupon"],
      "stream_content_type":["text/html"],
      "stream_size":[123429],
      "description":["Cafe Food and Drinks at French Market Cafe (Up to 40% Off). Two Options Available. "],
      "content_encoding":["UTF-8"],
      "twitter_description":["French-style cafe that draws many European expats serves up Le Cheval ham, swiss, and fried-egg
ourguignon"],
      "twitter_site":["groupon"],
      "twitter_image_src":["https://img.grouponcdn.com/deal/ktjo4LzUuXbkMMCX39dw/PS-700x420/v1/t440x300.jpg"],
      "content_type":["text/html; charset=UTF-8"],
      "resourcename":["/home/solr-5.3.1/crawl_data/french-market-cafe.html"],
      "dc_title":["Cafe Food and Drinks - French Market Cafe | Groupon"],
      "_version_":1517974795196563456},
    ,
```

9. We can configure Solr to default querying from the field _text_ by defining the default field in the requestHandler in solrconfig.xml file.



```
<requestHandler name="/select" class="solr.SearchHandler">
  <!-- default values for query parameters can be specified, these
       will be overridden by parameters in the request
    -->
  <lst name="defaults">
    <str name="echoParams">explicit</str>
    <int name="rows">10</int>
    <!-- <str name="df">text</str> -->
  </lst>
```

Uncomment the str element with name “df”, and replace it in the following manner, where we specify that the default query field has to be “_text_”.



```
<requestHandler name="/select" class="solr.SearchHandler">
  <!-- default values for query parameters can be specified, these
       will be overridden by parameters in the request
    -->
  <lst name="defaults">
    <str name="echoParams">explicit</str>
    <int name="rows">10</int>
    <str name="df">_text_</str>
  </lst>
```

You can reload your core and query with the new configuration.

Note: in practice we would want to build a more appropriate user interface, one in which the query results are properly displayed. Our new user interface would make AJAX calls on the Solr index rather than use the SolrUI.