ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

Implementation of Highly Optimized Search Engine Using Solr

Vikash Kumar [1], P.N. Barwal[2]

Project Engineer, e-Governance, C-DAC, NOIDA, U.P., India[1]

Joint Director, e-Governance, C-DAC, NOIDA, U.P., India[2]

ABSTRACT: This tool of choice can be used to solve our unanswered questions, finding a product or service through the help of a search engine. This paper provides a detailed description of this tool. It describes the indexing and searching of the data. Whether it is a document or a structured data, anything can be searched using this technology. The functionalities of the proposed framework are exposed using RESTful web services. The outcomes are communicated to Standard Development Organizations C-DAC. Everything that we want to know about, we find now a day's using search engine. That's why now a search engine is the most important part of our daily life. It is either a matter of finding any sci-fi questions answer or anything....anything; the search engine is always helpful. It is now not limited up to an IT professional. Even the non-technical person is also getting in touch with a search engine. That's why it is very important to be the searching to be optimized and fast. I have implemented the search tool using the Apache Solr, with Jquery, in a LINUX environment. This mechanism of searching will help to discover resources and capabilities, once all the indexing of information is done in it.

KEYWORDS*:* Solr, Angular, Linux, Search Engine, Indexing, Full Text Search, JSON, CSV, XML.java

I. INTRODUCTION

The system is to aid the organization in achieving their intent of an efficient, effective, and user friendly search engine. This paper facts about the process in order of objective, technology used, required knowledge, setting up the environment, indexing the existing data, searching based on different example queries, optimization of searching.
The objective of this paper is to endow with the information that we never think about while searching, to the process being performed in the backend. Till now it is limited to the only advanced technical persons only. But there must be some basic know-how to the person from all other categories. Because what I think is, to day in the internet world, nothing is limited to a specific one. It is also useful for them who want to integrate search engine to their application. This has been designed with a motto that it will be used for the application being developed for scientific research board, proposal management system and licence management system. Although it is not limited to these applications, this will be helpful to integrate it in any application. It can be used as an add-on with other applications.

II. RELATED WORK

Searching a word or phrase from any structured database or a file really needs a proper map of related contents.This is where a terms appear indexing. Indexing and crawling is different, that we will see one by one.An index does not contain the documents but a list of word or phrase for each one of them.When we say that the document has been indexed, it means that the word or phrase has been indexed which is related to that document. Ex:If we have a document whis is related to IJIRSET, and it is related with name itself, then indexing that particular document means, whenever any one searches for the IJIRSET, that document will be also part of the result.Crawling is the term used to dedupes the list and then rearranges.Our system aims towards the searching i.e. bot indexing and crawling. And also it is not limited to the urls. It will search in document also.The technique used for this purpose is Apache solr.Solr has been integrated with the post tool, having capability for both indexing and crawling.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

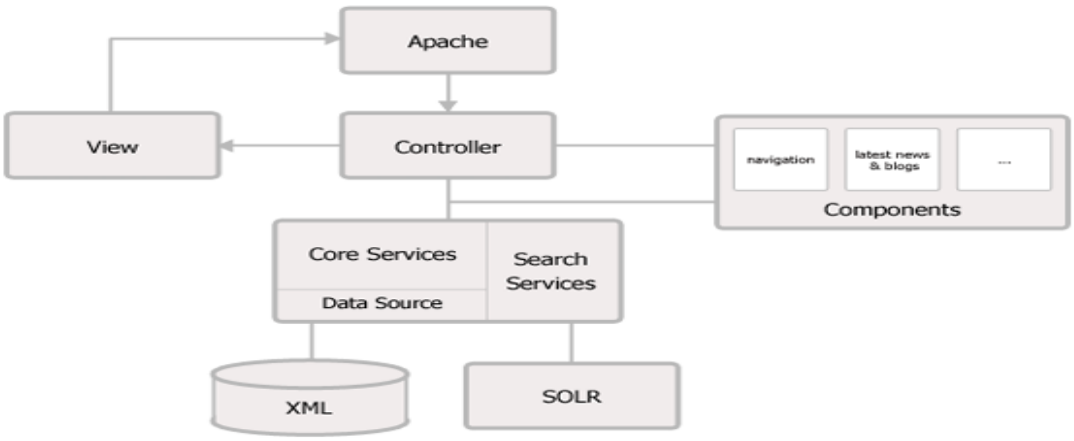
Vol. 5, Issue 3, March 2016

III. TECHNOLOGY USED

The environment used to develop this tool is LINUX. Although it can be done in windows environment also, using minor changes in environment setup/configuration commands. For the indexing purpose, apache Solr has been used. Apache tomcat has been used as web server and of course Java is also part of this. Now a question may arise that if database structured query is there, then why solr? Databases and Solr have complementary strengths and weaknesses. SQL supports very simple wildcard-based text search with some simple normalization like matching upper case to lower case. The problem is that these are full table scans. In Solr all searchable words are stored in an "inverse index", which searches orders of magnitude faster. Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called indexing) via JSON, XML, CSV or binary over HTTP. You query it via HTTP GET and receive JSON, XML, CSV or binary results.

From a database perspective, an index can be thought of as one DB table with very fast lookups and interesting enhancements for text search. This index is relatively expensive in space and creation time. Solr wraps this API with a full-featured front end, providing these additions as: It provides a clean deployment as a web service for indexing and searching, convenient scalability across multiple servers, learning curve & adoption improvement of ~2 orders of magnitude.

Solr enable powerful matching capabilities including phrases, wildcards, joins, grouping and much more across any data type. Solr is proven at extremely large scales the world over.It ships with a built-in, responsive administrative user interface to make it easy to control your Solr instances.)



(a)

IV. ENVIRONMENT SETUP

Now, we will move towards the environment setup. Linux, JDK 7 or higher, apache tomcat can be used. Download the JDK tar file from its official website. Install JDK and set the environment variables.

```
Export JAVA_HOME = /opt/jdk1.7.0.79
EXPORT JRE_HOME = /opt/jdk1.7.0.79/jre
EXPORT PATH=$PATH: /opt/jdk1.7.0.79/bin: /opt/jdk1.7.0.79//jre/bin
```

(a)

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

After installing java and setting the environment variable, whether java has been installed properly or not can be seen using command: java –version. Now download the apache solr and extract it in the required folder.

```
bin/solr stop –all
bin/solr start
bin/solr create –c vikahTest –d basic_configs
```

(b)

The above command can be used to start as well as to stop the solr service. This can be verified from the browser by typing the URL as: http://localhost:8983/solr. Sometimes the port may vary if changed. Otherwise by default the same will be used. The last one is creating a core in solr, which is having the configuration files, required for indexing. Let us assume that we have a CSV file with some field and values. And what we have to do is, we will create the index and get the result by applying the different queries. Below is the screenshot of the CSV file.
When we parse any document for creating index using solr, it save the details of in tokenizes form. These tokens are used as the source of output against any search text.

Id	Cat	Name	inStock	Author	Series_t	Sequence_i	Genre_s
1	Book	A Game of Thrones	20	True	George	1	Fantasy
2	Book	A clash of kings	20	False	Isaac Asimov	1	Fantasy
3	Book	A storm of swords	23	True	Glen Cook	1	Fantasy
4	Book	Foundation	27	True	Orson Scott	1	Fantasy
5	Book	The Black COmpany	29	True	Steven Brust	1	Fantasy

(c)

V. DEVELOPMENT

There is a file named schema.xml, inside the created core folder. The fields available in CSV should also be configured into that file.

```
<uniqueKey>id</uniqueKey>
<field name="cat" type="text_general" indexed="true" stored="true"
multiValued="true"/>
<field name="name" type="text_general" indexed="true" stored="true"
multiValued="true"/>
<field name="price" type="tdouble" indexed="true" stored="true"
multiValued="true"/>
<field name="inStock" type="boolean" indexed="true" stored="true"
multiValued="true"/>
<field name="author" type="text_general" indexed="true" stored="true"
multiValued="true"/>
<field name="authors" type="text_general" indexed="true" stored="true"
multiValued="true"/>
<field name="series_t" type="string" indexed="true" stored="true"
multiValued="true"/>
-----------------------------
```

(a)

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

Now it's the time to index the file. Apache solr provide a tool named Simple Post tool for indexing. For this post.jar must be inside the folder where books.csv is located.

```
java -Dtype=text/csv -Durl=http://localhost:8983/solr/vikashTest/update -jar
post.jar books.csv
```

(a)

After this command, re-start the solr server and can type query to select the required data. Below are some example queries that I had used.

```
http://localhost:8983/solr/vikashTest/select?q=sequence_i:2

http://localhost:8983/solr/vikashTest/select?q=pages_i:*

http://localhost:8983/solr/vikashTest/select?q=id:978-0641723445

http://localhost:8983/solr/vikashTest/select?q=cat:hardcover

http://localhost:8983/solr/vikashTest/select?q=cat:hardcover,book

http://localhost:8983/solr/vikashTest/select?q=genre_s:*

http://localhost:8983/solr/vikashTest/select?q=pages_i:*

http://localhost:8983/solr/vikashTest/select?q=pages_i:384
```

(b)

Thus by executing these queries you can get result in xml or json format. Apache solr also support indexing from the JSON, XML, and HTML and from binary data also. We can use these search result into our application.

VI. INTEGRATION WITH TOMCAT

For this it is required to integrate solr into tomcat. There is a folder named dist inside solr home directory. This folder contains solr-x.x.war file. Rename this as solr.war and copy to into the webapps directory of Tomcat. Create a file solr.xml in Tomcat conf\ Catalina\ localhost folder, and add the following in it:

<?xml version="1.0" encoding="UTF-8"?>
<Context crossContext="true" debug="0" docBase="root\Tomcat\webapps\solr.war">
<Environment override="true" value="root\solr" type="java.lang.String" name="solr/home"/>
</Context>

Save it. This sets your solr home.Once you are done with these steps; you need to check if the solr is running in Tomcat. For this purpose, Start Your Tomcat, go to http://localhost:8081/solr/admin , if it works, then your solr has been installed on Tomcat successfully.While selecting the query, we can use facets or conditions through which only the amount of required data will be searched and shown to users. There is a lot to do in solr, which I will gradually implement and share. Below is the code which I will use as a reference.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

```
<!DOCTYPE html>
    <html>
    <head><meta charset="ISO-8859-1"><link
     href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css"
     rel="stylesheet"></link>
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
     $(function() {
     var URL_PREFIX = "http://localhost:8983/solr/jcg/select?q=name:";
     var URL_SUFFIX = "&wt=json";
     $("#searchBox").autocomplete({
     source : function(request, response) {
     var URL = URL_PREFIX + $("#searchBox").val() + URL_SUFFIX;
     $.ajax({
     url : URL,
     success : function(data) {
     var docs = JSON.stringify(data.response.docs);
     var jsonData = JSON.parse(docs);
     response($.map(jsonData, function(value, key) {
     return {
     label : value.name
     } }));
     }, dataType : 'jsonp',
     jsonp : 'json.wrf'
     }); },
     minLength : 1
     })  });
    </script>
    </head>
    <body>
     <div><p>Type The or A</p><label for="searchBox">Tags: </label>
     <input id="searchBox"></input>
     </div>
```

(a)

VII. Future Enhancement

Future Enhancements: There is always scope of enhancements in every work. As I have tried to give a short details but in right sequence for implementation of solr. I have used CSV format for indexing only. In the same way we can use database indexing, different type of documents (PDF, DOX, DOCX...so on) indexing.

VIII. Conclusion

Conclusion Apache solr is proven an agile search engine. It has all the features required to build and integrate into any application where searching is performed from a large amount of data. It supports cloud features as well as scalability. It is free or on a very low cost, and can be created as per the business requirements. It is continuously evolving in real time as developers add to it and modify it to make it a superior quality of search platform. In the coming time, I will share the next stage of the paper.

References

[1] Author:Jenny Halasz, President of an online marketing consulting company offering SEO, PPC, and Web Design services, http://searchengineland.com/how-search-engines-work-really-171556
[2] http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp
[3] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6832331&queryText=indexing&searchWi thin=solr&searchField=Search_All
[4] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6970198&queryText=indexing&searchWi thin=solr&searchField=Search_All
[5] https://wiki.apache.org/solr/WhyUseSolr
[6] http://lucene.apache.org/solr/features.html
[7] http://www.ixxus.com/blog200902ixxus-web-framework/
[8] http://examples.javacodegeeks.com/enterprise-java/apache-solr/solr-autocomplete-example/
[9] http://examples.javacodegeeks.com/enterprise-java/apache-solr/solr-autocomplete-example/
Rajani Maski, www.happiestminds.com