# TOPICS IN DATA MANAGEMENT (ASSIGNMENT 3)

**SANDHYA MURALI**
sm2290@g.rit.edu

## ABSTRACT:

This application is about extracting the data from the 2 files (mashup and api) , parsing the data, loading the data into a NoSQL database and firing queries based on user input. The No-SQL database used here is MongoDB. A user friendly interface in JSP is provided for firing and executing the queries. The data from the dataset (mashup and api) is loaded into MongoDB database using Java mongo connection.

## INTRODUCTION:

Web services are a set of application services that are used in order to exchange information between applications. They can be be invoked by programs using HTTP requests from a repository in order to create new products. A number of users can access some of these services through peer-to-peer communication rather than accessing the central server.

In order to store large data from files and for those files whose attributes are not fixed; a No-SQL database is used. In this assignment, No-SQL MongoDB database is used in order to parse and store the data. MongoDB stores the documents in BSON format. When comparing to relational databases, document refers to set of rows and collections refer to as tables.

In this assignment, I have used MongoDB as a No-SQL database where I have parsed the data and loaded into the database. After parsing, based on user input, appropriate queries are executing by considering the criteria parameters needed by the user and appropriate results are generated. The parsing and loading into the database is done using Java Mongo connections and a user friendly interface in JSP is provided which will enable the users to fire the queries.

The implementation details of the application is seen in the design section, screenshots are seen in the results section which is followed by challenges faced and conclusion.

## DESIGN:

The design of the application is as follows:

The design is divided into two parts:
1. Parsing and Loading into the database.
2. Firing Queries.

As a part of setting up, one needs to setup MongoDB in the system and load the required jar files and MongoDB plugins into Netbeans IDE.

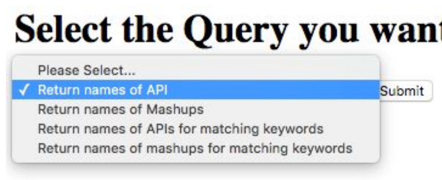
The first part is entirely coded in Java. In this ode, the parsing and separating of multiple attribute values is done using Regex (Regular Expressions ). That is, I have used regex to check if $#$ or ### is present. If $#$ is present, it marks the end of a document and beginning of another document whereas if ### is present, multiple values are present for a particular attribute. The parsed data is loaded to MongoDB database using Java Mongo Connections that help the Java compiler to connect to the mongoDB database. I have created two tables (one for api.txt) and (one for mashup.txt) in order to store these data. The data structure of the database is a key value pair (BSON format) where key is the column name and attribute is a value. In this way the column name and attribute value is specified while loading into the database.

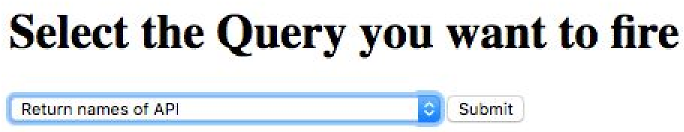
The second part of the assignment is done using JSP wherein a user friendly interface is provided to the users in order to fire the query. The UI takes input as to which query to fire, takes parameters for input criteria and displays the needed result. This process is repeated until the user closes the dialog box.

At the time of querying the database, there are some attributes such as tags that are more than one values. At the time of name retrievals, while considering tag as a criteria; I have used regex in order to check if the tag with that specific user input is present. This regex considers the string as a whole and checks if the user specified tag is present in any of the documents and return those API or Mashup names if satisfied. This querying is done for single as well as multiple criterion based on the requirement of the user.

__SCREENSHOTS:__

The user selects query as follows: In this example, **the user selects API (query1) name as the query he/she wants to fire.**

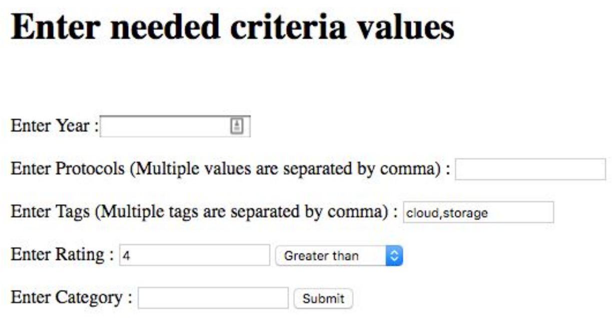

Select the Query you want to fire
Please Select...
✓ Return names of API
Return names of Mashups
Return names of APIs for matching keywords
Return names of mashups for matching keywords
Submit


Select the Query you want to fire

Return names of API

Submit

The user is then asked to select the criteria and value of the criteria.


Enter needed criteria values

Enter Year :

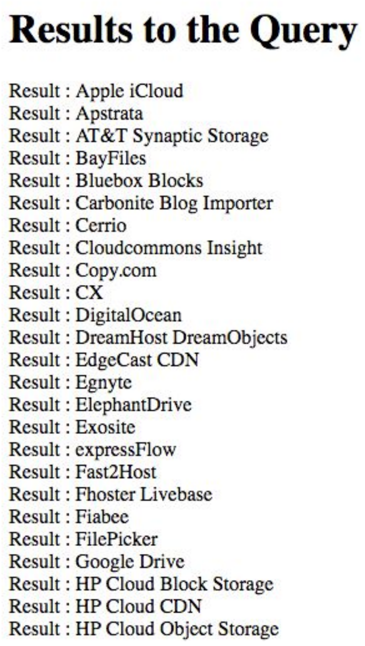
Enter Protocols (Multiple values are separated by comma) :

Enter Tags (Multiple tags are separated by comma) : cloud,storage

Enter Rating : 4 Greater than

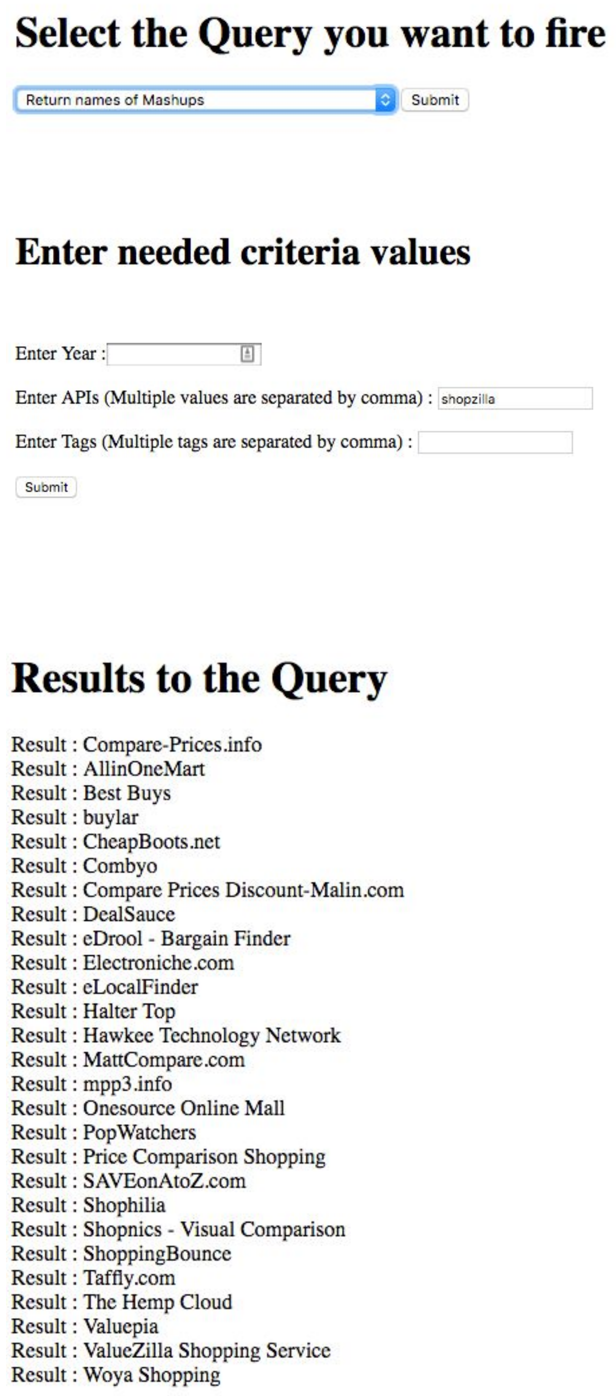
Enter Category : Submit

Based on the above criteria specified and the value of the criteria, a query is fired to the MongoDB database and the below results are fetched.


Results to the Query

```
Result : Apple iCloud
Result : Apstrata
Result : AT&T Synaptic Storage
Result : BayFiles
Result : Bluebox Blocks
Result : Carbonite Blog Importer
Result : Cerrio
Result : Cloudcommons Insight
Result : Copy.com
Result : CX
Result : DigitalOcean
Result : DreamHost DreamObjects
Result : EdgeCast CDN
Result : Egnyte
Result : ElephantDrive
Result : Exosite
Result : expressFlow
Result : Fast2Host
Result : Fhoster Livebase
Result : Fiabee
Result : FilePicker
Result : Google Drive
Result : HP Cloud Block Storage
Result : HP Cloud CDN
Result : HP Cloud Object Storage
```

**Similarly, if user selects query2 (names of Mashup), output will be as follows:**



Select the Query you want to fire

Enter needed criteria values

Enter Year :

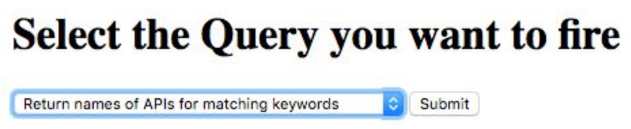
Enter APIs (Multiple values are separated by comma) : shopzilla

Enter Tags (Multiple tags are separated by comma) :

Results to the Query

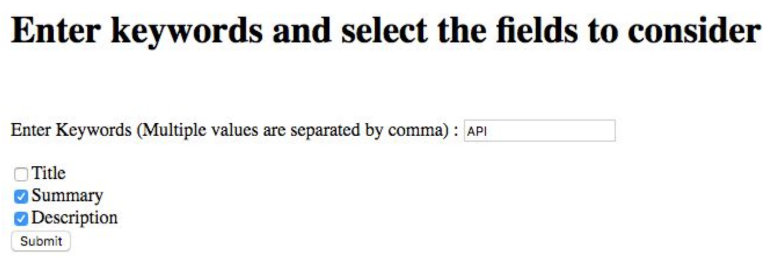
```
Result : Compare-Prices.info
Result : AllinOneMart
Result : Best Buys
Result : buylar
Result : CheapBoots.net
Result : Combyo
Result : Compare Prices Discount-Malin.com
Result : DealSauce
Result : eDrool - Bargain Finder
Result : Electroniche.com
Result : eLocalFinder
Result : Halter Top
Result : Hawkee Technology Network
Result : MattCompare.com
Result : mpp3.info
Result : Onesource Online Mall
Result : PopWatchers
Result : Price Comparison Shopping
Result : SAVEonAtoZ.com
Result : Shophilia
Result : Shopnics - Visual Comparison
Result : ShoppingBounce
Result : Taffly.com
Result : The Hemp Cloud
Result : Valuepia
Result : ValueZilla Shopping Service
Result : Woya Shopping
```

**Consider another example of displaying API names based on keywords (query3):**



Select the Query you want to fire

The user enters a set of keywords separated by comma. These keywords are extracted using regex at the time of querying using regex. The user selects different fields to check if keywords are present in all of the fields.

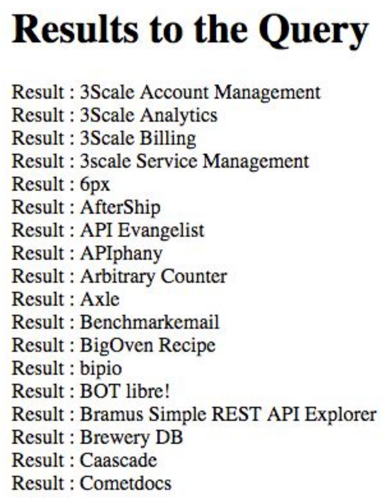


Enter keywords and select the fields to consider

Enter Keywords (Multiple values are separated by comma) : API

[ ] Title
[✓] Summary
[✓] Description

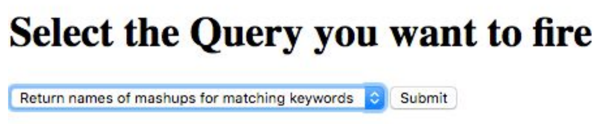
Display the results.



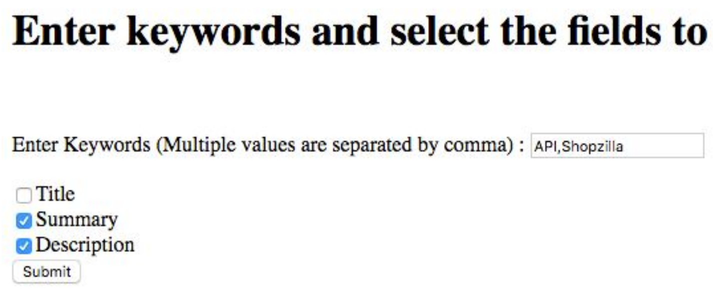
Results to the Query

Result : 3Scale Account Management
Result : 3Scale Analytics
Result : 3Scale Billing
Result : 3scale Service Management
Result : 6px
Result : AfterShip
Result : API Evangelist
Result : APIphany
Result : Arbitrary Counter
Result : Axle
Result : Benchmarkemail
Result : BigOven Recipe
Result : bipio
Result : BOT libre!
Result : Bramus Simple REST API Explorer
Result : Brewery DB
Result : Caascade
Result : Cometdocs

Similarly, if user selects query4 (Mashup names based on keywords), the screenshots are as follows:



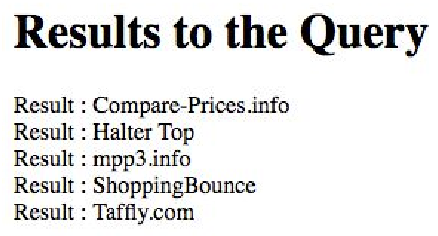
Select the Query you want to fire



## Enter keywords and select the fields to consider

Enter Keywords (Multiple values are separated by comma) : API,Shopzilla

- [ ] Title
- [✓] Summary
- [✓] Description



## Results to the Query

```
Result : Compare-Prices.info
Result : Halter Top
Result : mpp3.info
Result : ShoppingBounce
Result : Taffly.com
```

## <u>CHALLENGES AND LEARNINGS:</u>

From this assignment, I learnt how to use MongoDB for parsing and storing of data. It helped me learn JSP in order to build the UI.

I found some issues initially in order to set up connection using Java with MongoDB as I was not getting the appropriate driver. However once I could understand how connection is established, loading the data as well as querying in Java was easy. However since I was completely new to JSP framework,it took me some time to understand how to use Servlet and JSP with Java Application in order to develop a Web Based System.

## <u>CONCLUSION:</u>

In this way, using MongoDb, parsing,loading and querying was established.