Web Services Assignment 3 Documentation

DESIGN OF DATA STRUCTURE

Two collections in the test database have been created-

1. Collection name: apiData (to store api data)
2. Collection name: mashupData (to store mashup data)

Collection: mashupData

It consists of 7392 documents and the documents consist of 18 fields. Given below are the fields:

Id: string

Title: string

Summary: string

Rating: double

Name: string

Label: string

Author: string

Description: string

Type: string

Downloads: int

UseCount: int

SampleUrl: string

DateModified: ISO date format

NumComments: int

CommentsUrl: string

Tags: Array of strings

APIs: Array of JSON objects → APIs: [{ApiName: "abc", ApiUrl: "xyz"}]

Update: ISO date format

The fields may not have any values. "(null)" has been inserted for such fields.

Screenshots of the data in Collection: mashupData


```
> db.mashupData.find({}).pretty()
{
        "_id" : ObjectId("57194a7b9ebe2bc589d8c19e"),
        "id" : "http://www.programmableweb.com/mashup/-22",
        "title" : "(null)",
        "summary" : "(null)",
        "rating" : 3.8,
        "name" : "(null)",
        "label" : "(null)",
        "author" : "Unknown",
        "description" : "(null)",
        "type" : "(null)",
        "downloads" : 0,
        "useCount" : 0,
        "sampleUrl" : "http://www.easypeasyphotos.net",
        "dateModified" : ISODate("2011-10-09T18:35:06Z"),
        "numComments" : 0,
        "commentsUrl" : "http://api.programmableweb.com/mashups/-22/comments",
        "tags" : [ ],
        "APIs" : [
                {
                        "apiName" : "Flickr",
                        "apiUrl" : "http://www.programmableweb.com/api/flickr"
                },
                {
                        "apiName" : "Google Maps",
                        "apiUrl" : "http://www.programmableweb.com/api/google-maps"
                }
        ],
        "updated" : ISODate("2011-10-09T18:35:06Z")
}
```


The above image shows a document structure in the mashupData collection

Another screenshot given below:


```
{
        "_id" : ObjectId("57194a7b9ebe2bc589d8c19f"),
        "id" : "http://www.programmableweb.com/mashup/compare-prices.info",
        "title" : "Compare-Prices.info",
        "summary" : "This site is a demo to show the functionality of the Shopzilla.com API. Supports the US and UK  API versions.",
        "rating" : 4.2,
        "name" : "Compare-Prices.info",
        "label" : "Compare-Prices.info",
        "author" : "Unknown",
        "description" : "This site is a demo to show the functionality of the Shopzilla.com API. Supports the US and UK  API versions.",
        "type" : "(null)",
        "downloads" : 0,
        "useCount" : 2170,
        "sampleUrl" : "http://www.compare-prices.info/",
        "dateModified" : ISODate("2009-02-10T05:35:01Z"),
        "numComments" : 2,
        "commentsUrl" : "http://api.programmableweb.com/mashups/compare-prices.info/comments",
        "tags" : [
                "affiliate",
                "eBay",
                "money",
                "Program",
                "shopping",
                "Shopzilla"
        ],
        "APIs" : [
                {
                        "apiName" : "Shopzilla",
                        "apiUrl" : "http://www.programmableweb.com/api/shopzilla"
                }
        ],
        "updated" : ISODate("2009-02-10T05:35:01Z")
}
{
        "_id" : ObjectId("57194a7b9ebe2bc589d8c1a0"),
        "id" : "http://www.programmableweb.com/mashup/mobile-emulator",
```

Collection: apiData

It consists of 11199 documents and the documents consist of 46 fields. Given below are the fields:

Id: string

Title: string

Summary: string

Rating: double

Name: string

Label: string

Author: string

Description: string

Type: string

Downloads: int

UseCount: int

SampleUrl: string

DownloadUrl: string

DateModified: ISO date format

RemoteFeed: string

NumComments: int

CommentsUrl: string

Tags: Array of strings

Category: string

Protocols: string

ServiceEndPoint: string

Version: string

Wsdl: string

DataFormats: string

Groups: string

Examples: Array of strings

ClientInstall: string

Authentication: string

Ssl: string

Readonly: string

VendorApiKits: string

CommunityApiKits: string

Blog: string

Forum: string

Support: string

AccountReq: string

Commercial: string

Provider: string

ManagedBy: string

NonCommercial: string

DataLicencing: string

Fees: string

Limits: string

Terms: string

Company: string

Update: ISO date format

Screenshots of the data in Collection: apiData

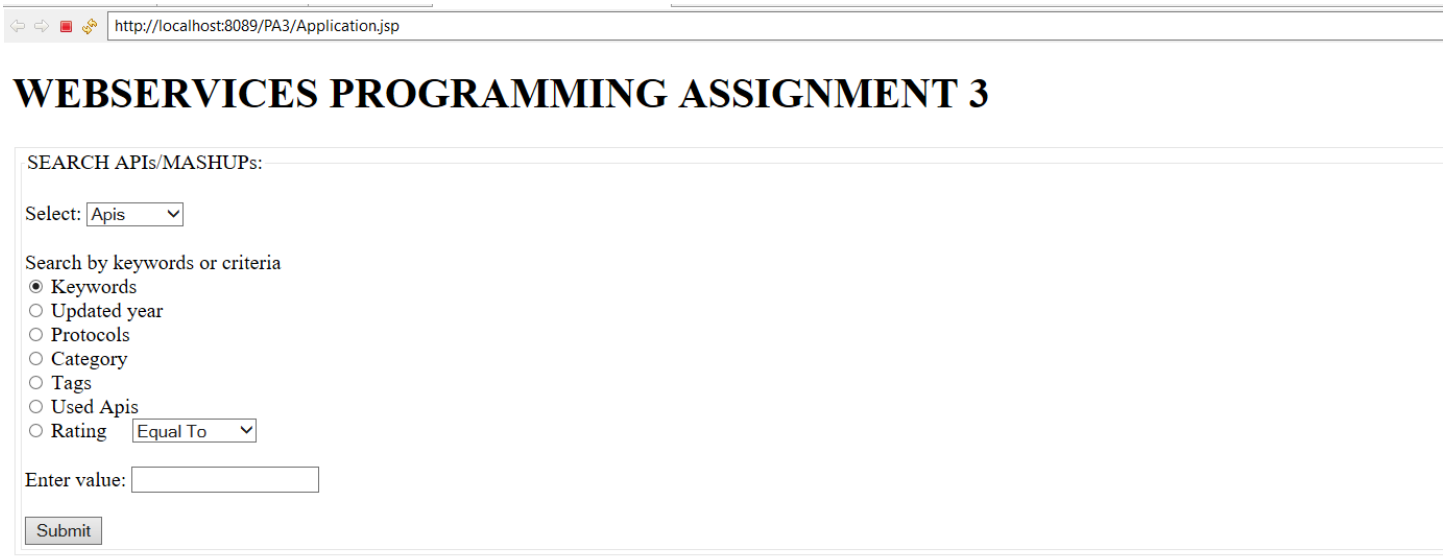

```
> db.apiData.find({}).pretty()
{
        "_id" : ObjectId("57194a719ebe2bc589d895df"),
        "id" : "http://www.programmableweb.com/api/the-global-proteome-machine",
        "title" : "The Global Proteome Machine",
        "summary" : "Proteome data for biomedical research",
        "rating" : 4.4,
        "name" : "The Global Proteome Machine",
        "label" : "The Global Proteome Machine",
        "author" : "(null)",
        "description" : "The Global Proteome Machine is an attempt to create knowledge from proteomics data and reuse it to solve biomedical research proble
s. The Global Proteome Machine Database was built to use GPM data to help validate peptide MS/MS spectra and protein coverage patterns. The Global Proteome
achine Database API provides RESTful access to commonly required information based on data from the GPM Database. Responses are JSON formatted.",
        "type" : "1",
        "downloads" : "(null)",
        "useCount" : "(null)",
        "sampleUrl" : "http://wiki.thegpm.org/wiki/GPMDB_REST",
        "downloadUrl" : "(null)",
        "dateModified" : "2012-12-17T09:51:40Z",
        "remoteFeed" : "(null)",
        "numComments" : "(null)",
        "commentsUrl" : "http://api.programmableweb.com/apis/the-global-proteome-machine/comments",
        "tags" : [
                "database",
                "science"
        ],
        "category" : "Science",
        "protocols" : "REST",
        "serviceEndpoint" : "http://gpmdb.thegpm.org/",
        "version" : "(null)",
        "wsdl" : "(null)",
        "dataFormats" : "JSON",
        "groups" : "(null)",
        "examples" : [ ],
        "clientInstall" : "(null)",
        "authentication" : "(null)",
```

```
        "authentication" : "(null)",
        "ssl" : "(null)",
        "readonly" : "(null)",
        "VendorApiKits" : "(null)",
        "CommunityApiKits" : "(null)",
        "blog" : "(null)",
        "forum" : "(null)",
        "support" : "(null)",
        "accountReq" : "No",
        "commercial" : "(null)",
        "provider" : "http://www.thegpm.org/",
        "managedBy" : "(null)",
        "nonCommercial" : "(null)",
        "dataLicensing" : "(null)",
        "fees" : "(null)",
        "limits" : "(null)",
        "terms" : "(null)",
        "company" : "(null)",
        "updated" : ISODate("2012-12-17T14:51:40Z")
}
```


The two images show a document structure in the apiData collection.

DESIGN OF QUERY SYSTEM

The UI of the application is as shown below:



The user can select whether he wants to search for apis or mashups.

Then he can select whether he wants to query based on some criteria or keywords.

The search vales are entered in the text box labelled by Enter value.

To retrieve apis, the apiData collection is queried and to retrieve mashups, the mashupData collection is queried.

To retrieve Apis

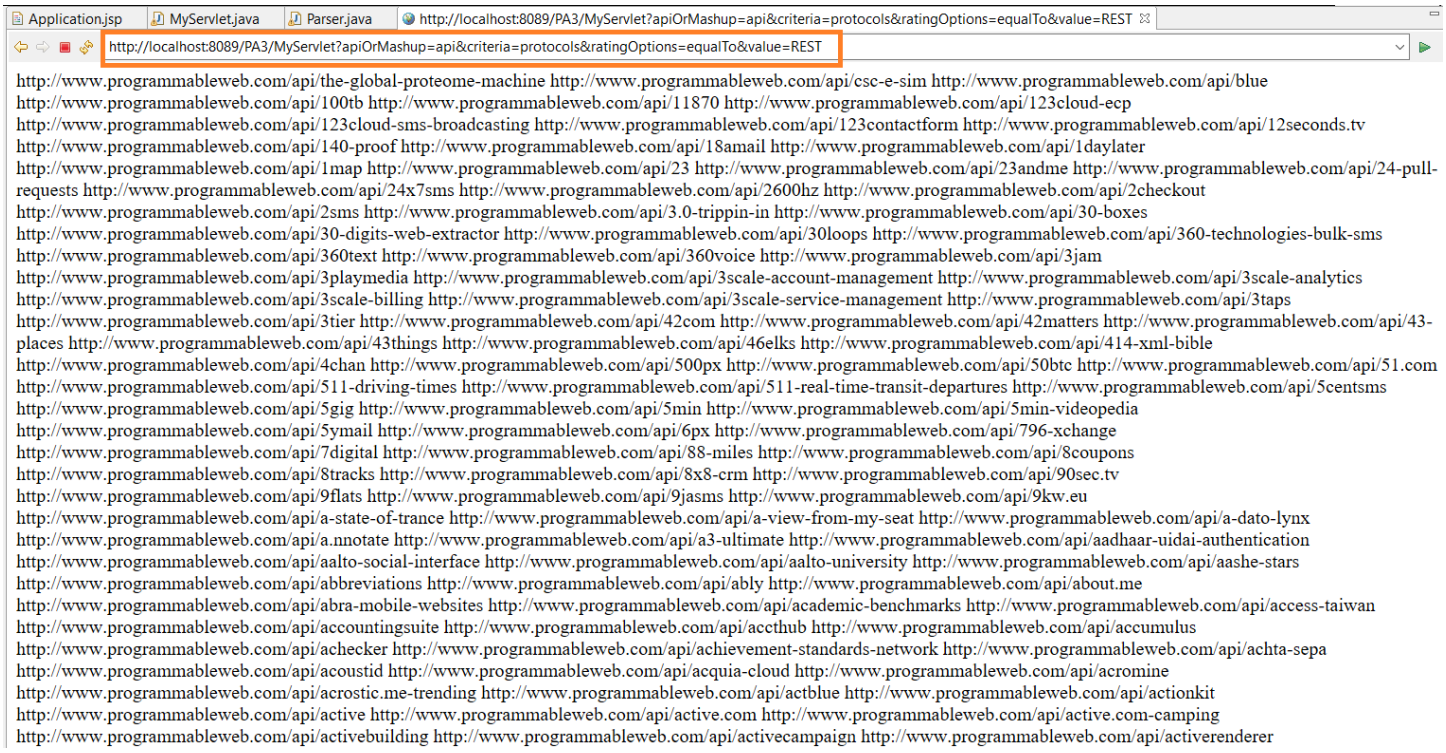
1. Based on Criteria: Protocols

```
BasicDBObject field = new BasicDBObject();
                    field.put("protocols", java.util.regex.Pattern.compile(values,
Pattern.CASE_INSENSITIVE));

                    DBCollection collection = db.getCollection("apiData");
                    DBCursor cursor = collection.find(field);

                    while (cursor.hasNext()) {

                          System.out.println(result.add((String) cursor.next().get("id")));
                          }
```

Result is an arraylist of strings that are api ids that are retrieved based on the input given by the user.

The user selects to retrieve apis. Selects criteria “protocols”. And enters the value of protocol as “REST”.

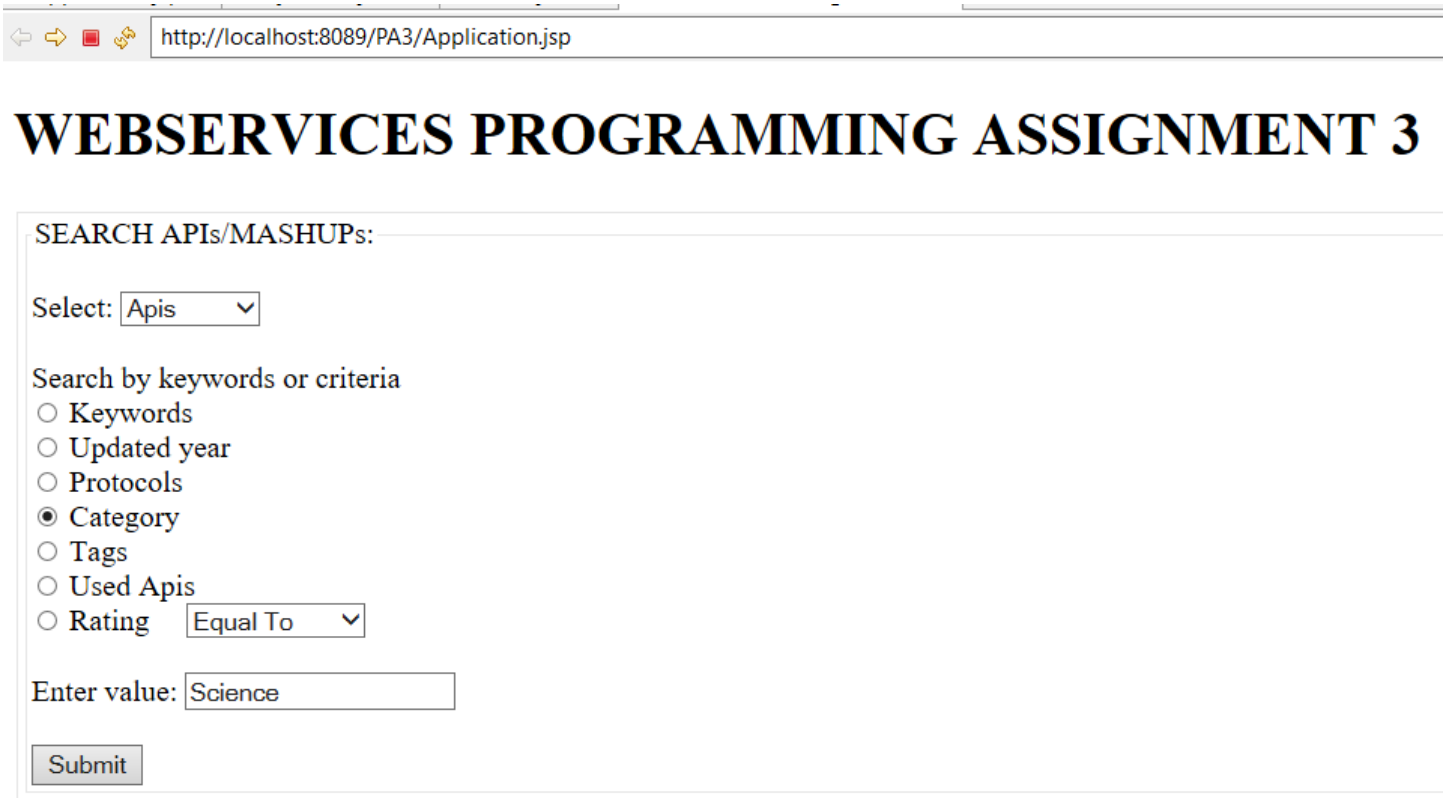
The output of the above query is given below. The screenshot displays a few of them.



2. Based on criteria : category

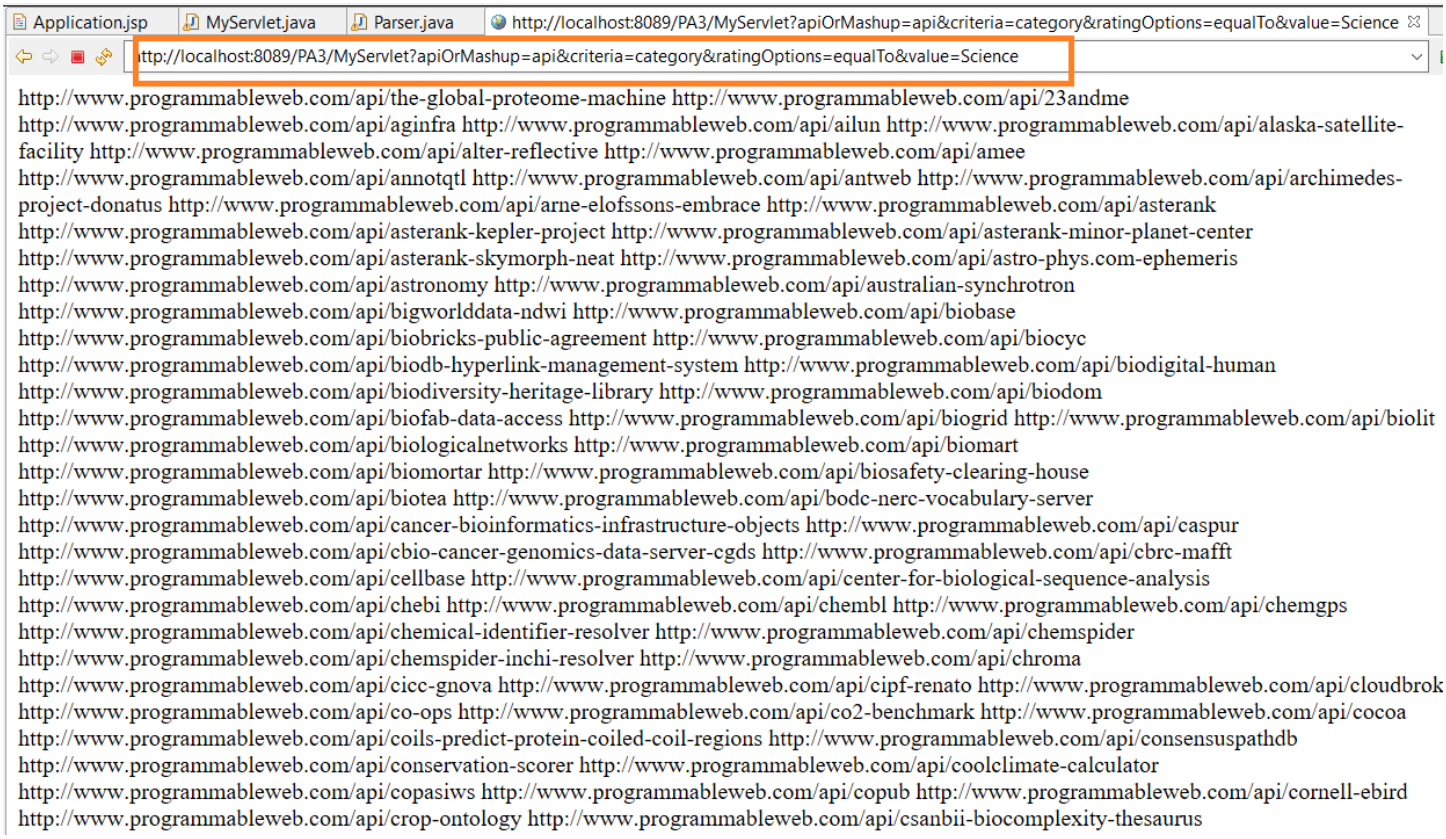
```
BasicDBObject field = new BasicDBObject();
                field.put("category", java.util.regex.Pattern.compile(values,
Pattern.CASE_INSENSITIVE));

                DBCollection collection = db.getCollection("apiData");
                DBCursor cursor = collection.find(field);

                while (cursor.hasNext()) {

                      System.out.println(result.add((String) cursor.next().get("id")));

                }
```

Result is an arraylist of strings that are api ids that are retrieved based on the input given by the user.



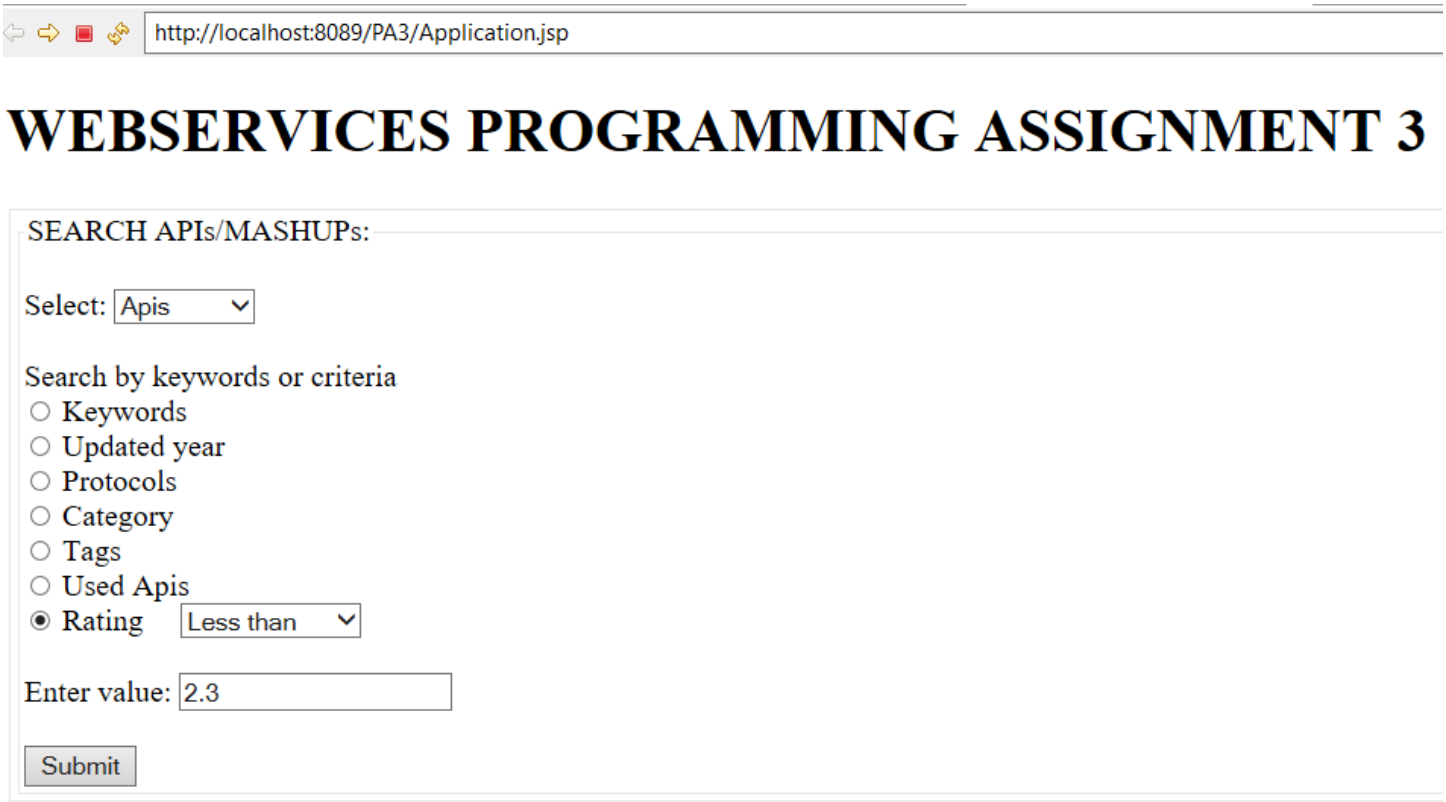
The user selects to retrieve apis. Selects criteria "category". And enters the value of protocol as "Science".

The output of the above query is given below. The screenshot displays a few of them.



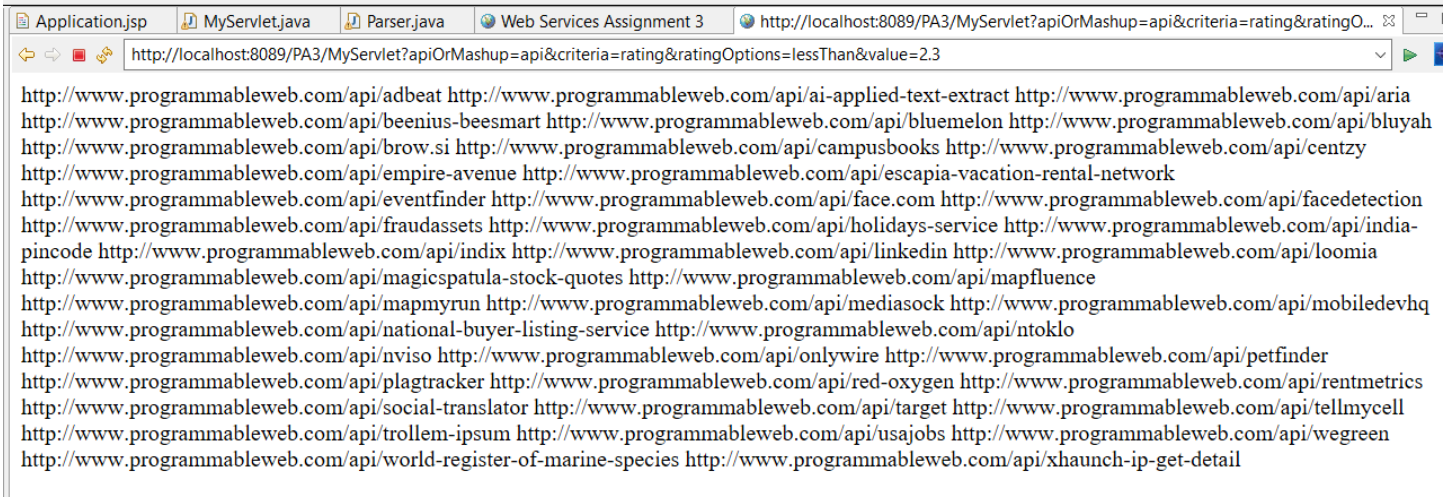
3. **Based on Rating**
    - **Less than a value**

```
BasicDBObject query = new BasicDBObject();
                        query.put("rating", new BasicDBObject("$lt", rating));
                        DBCollection collection = db.getCollection("apiData");
                        DBCursor cursor = collection.find(query);

                        while (cursor.hasNext()) {

                              System.out.println(result.add((String)
cursor.next().get("id")));
                                    }
```



The user selects to retrieve apis. Selects criteria "Rating"→ Less than. And enters the value of 2.3.

The output of the above query is given below. The screenshot displays a few of them.



- **Equal To**

```
field.put("rating", rating);
                        DBCollection collection = db.getCollection("apiData");
                        DBCursor cursor = collection.find(field);

                        while (cursor.hasNext()) {

                              System.out.println(result.add((String)
cursor.next().get("id")));
}
```

WEBSERVICES PROGRAMMING ASSIGNMENT 3

SEARCH APIs/MASHUPs:

Select: Apis

Search by keywords or criteria

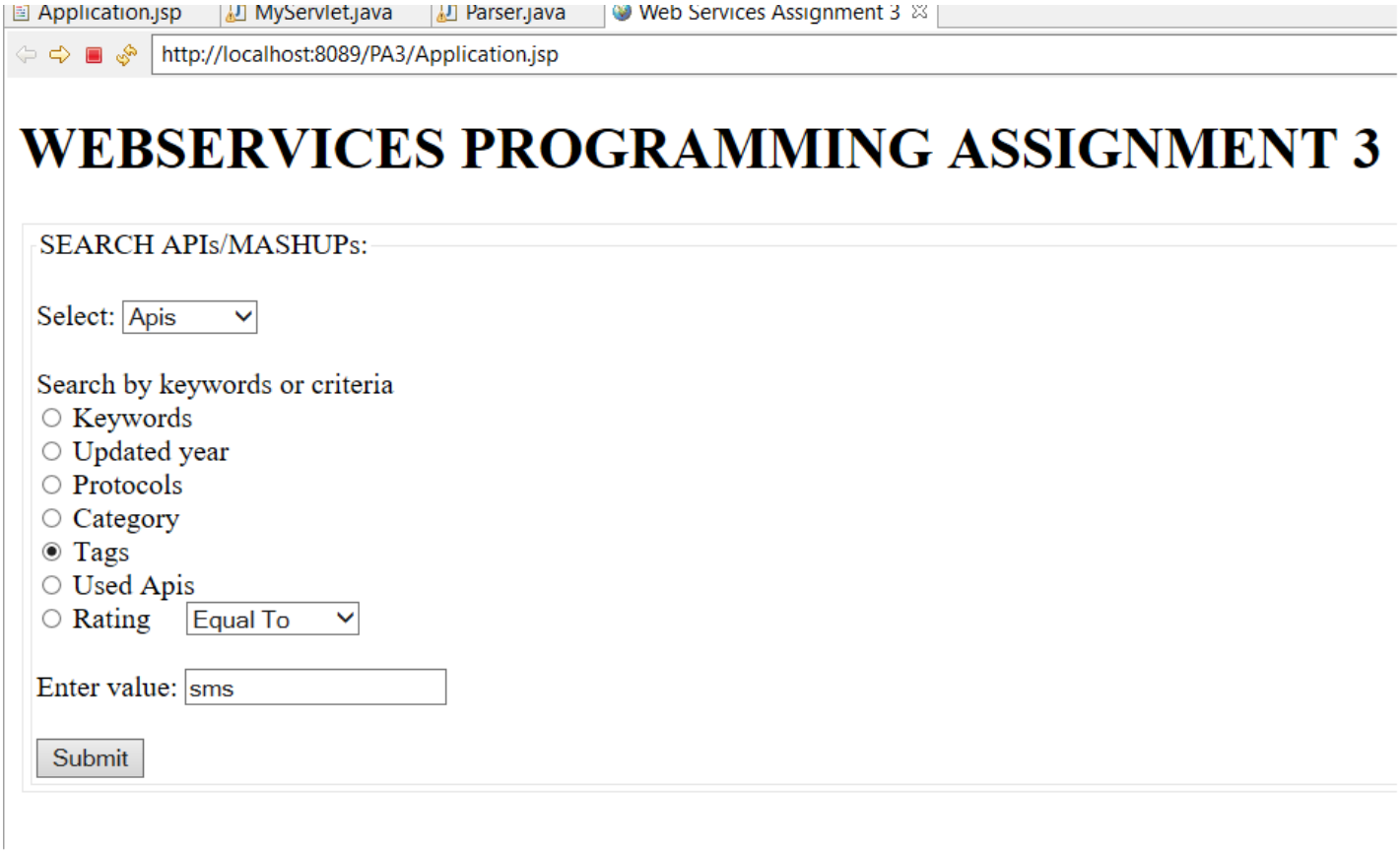
- ○ Keywords
- ○ Updated year
- ○ Protocols
- ○ Category
- ○ Tags
- ○ Used Apis
- ◉ Rating Equal To

Enter value: 2.3

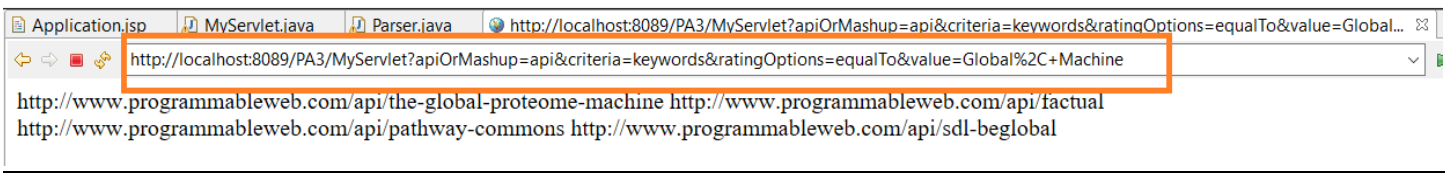
Submit

This gives the following output.

Based on criteria : tags



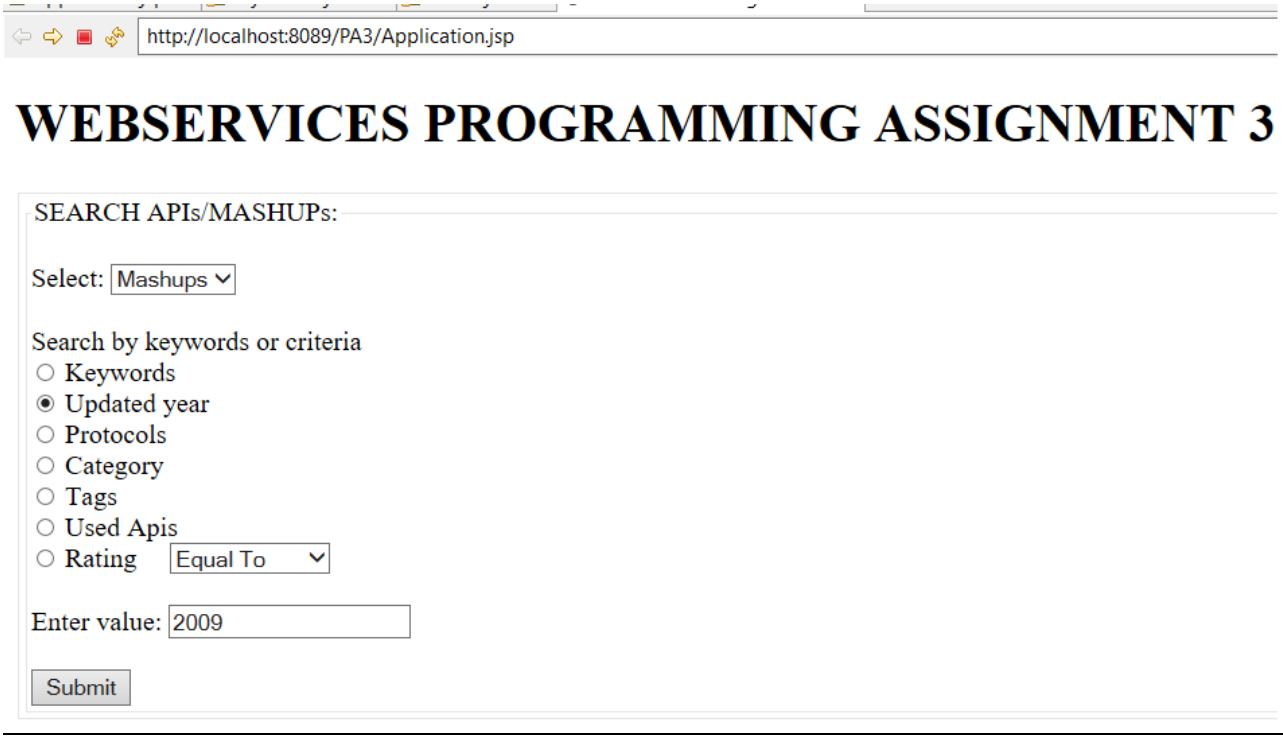
Based on Keywords: api

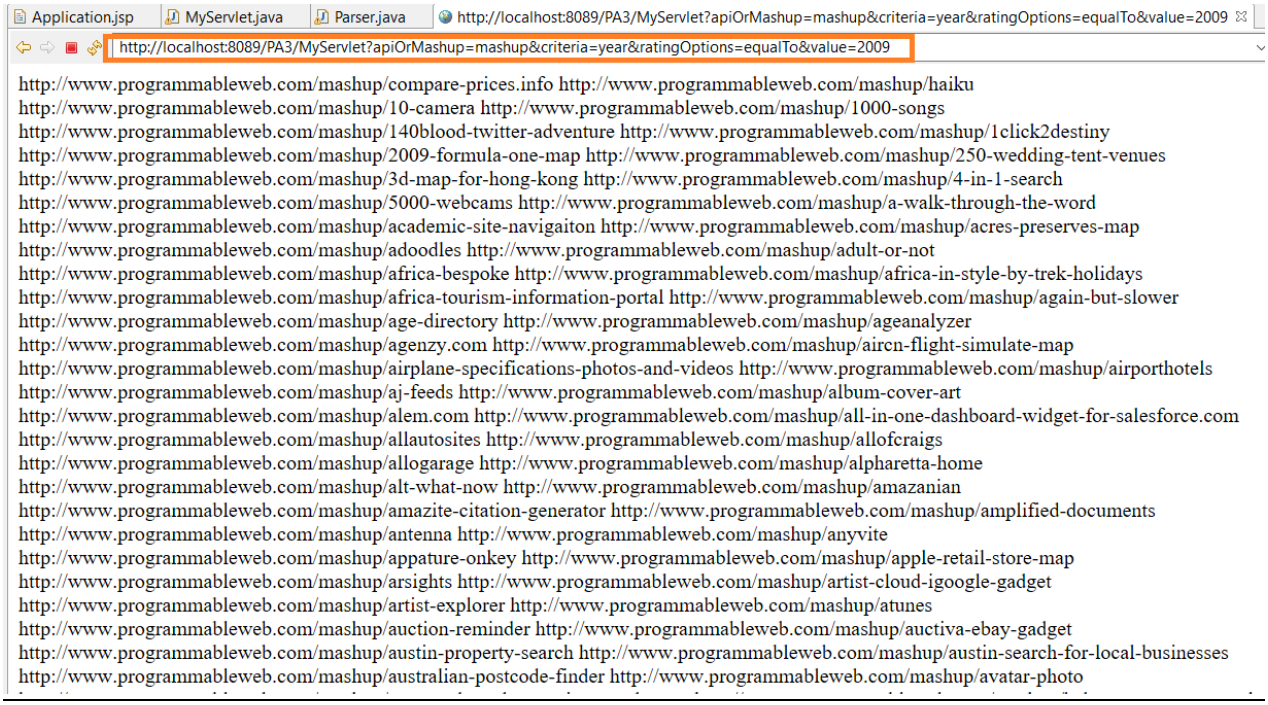


To retrieve Mashups

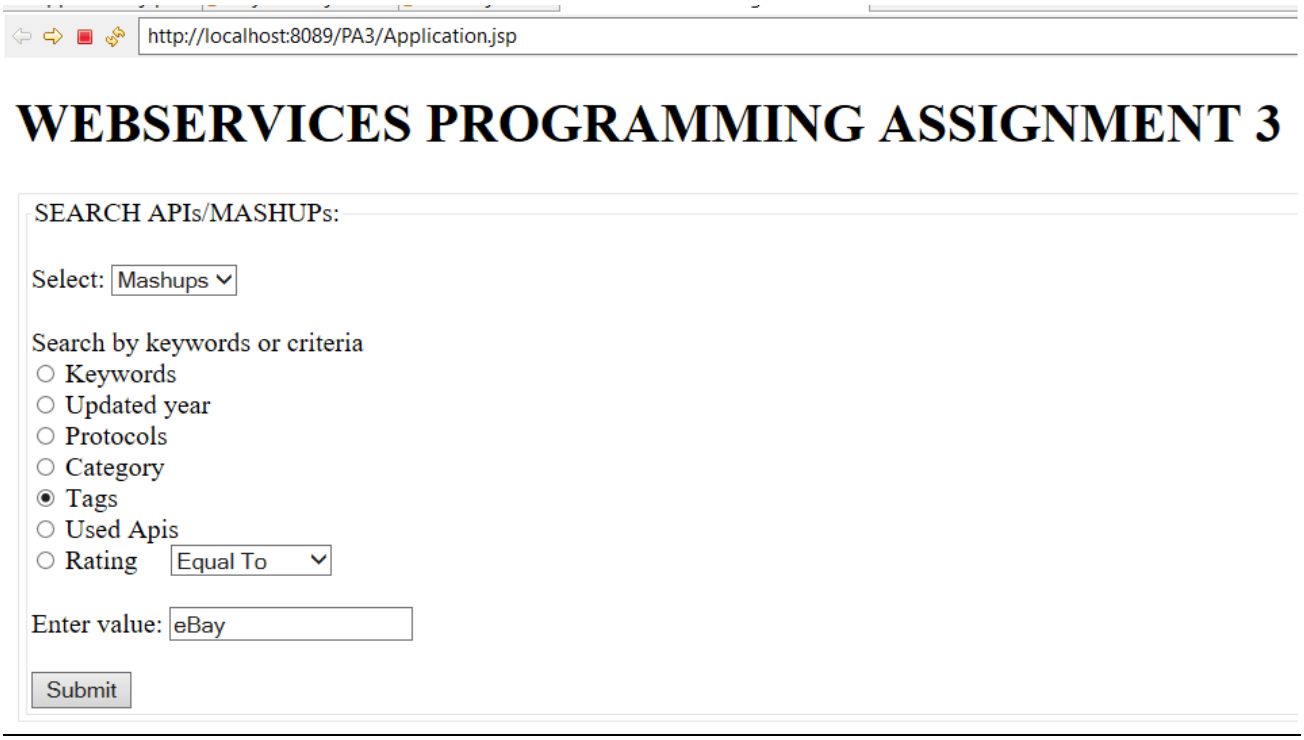
Based on criteria

1. **Year updated**

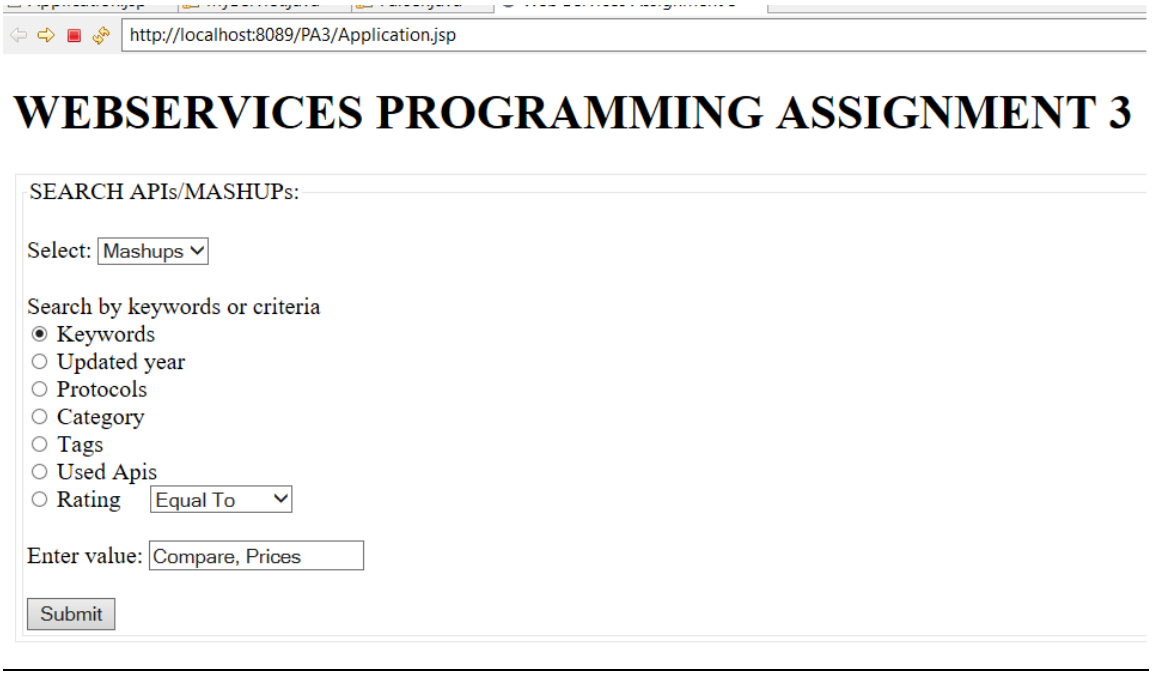


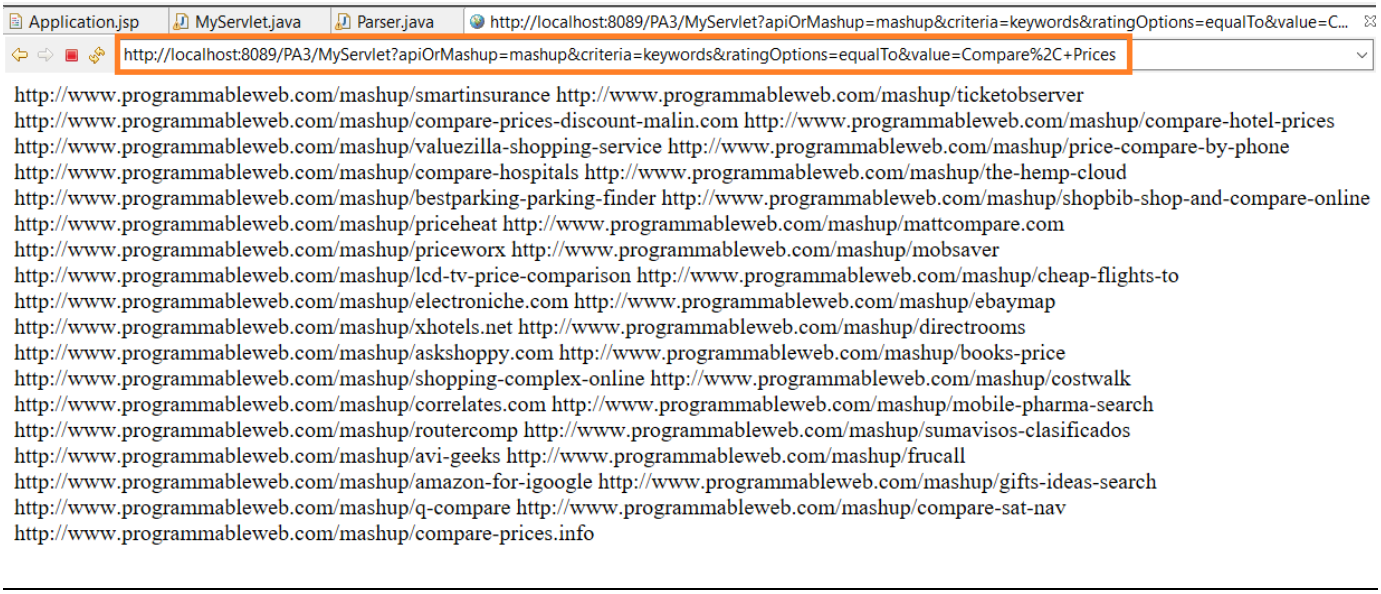


2. Criteria: tags

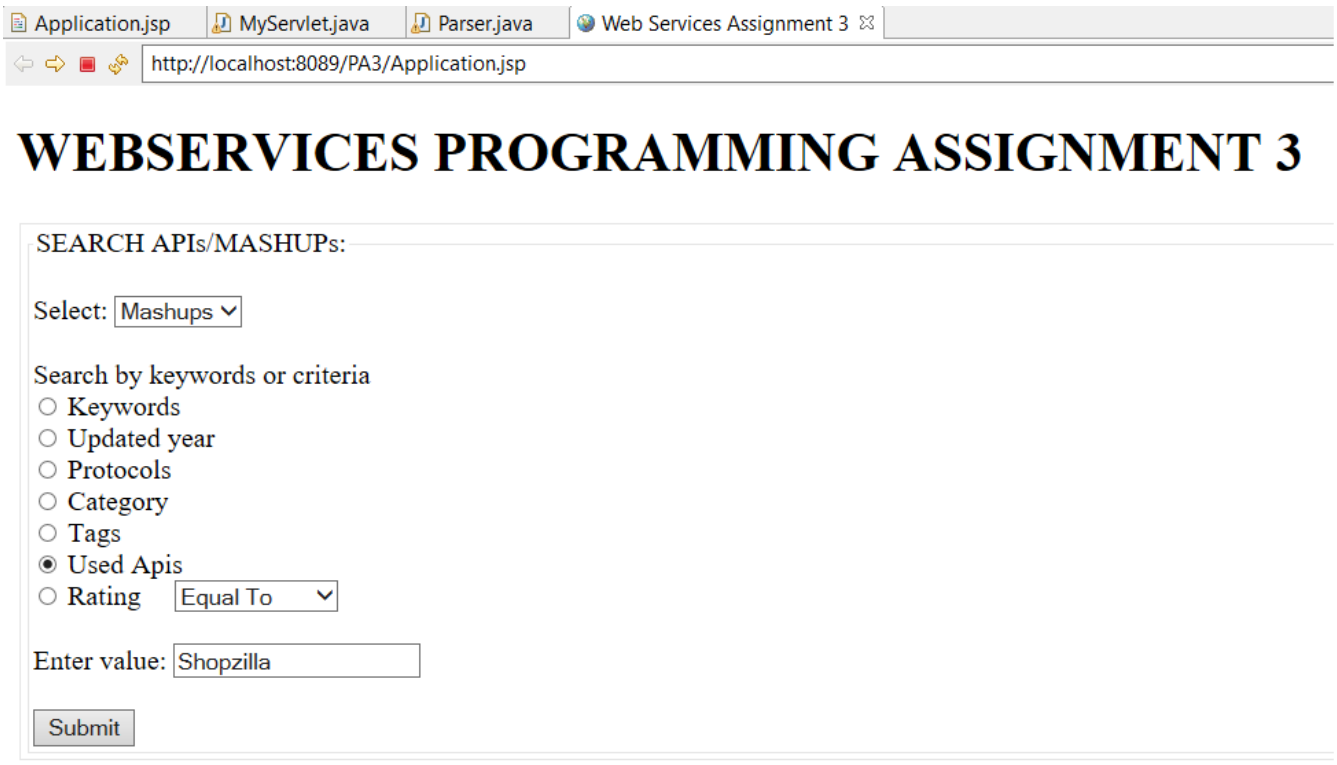


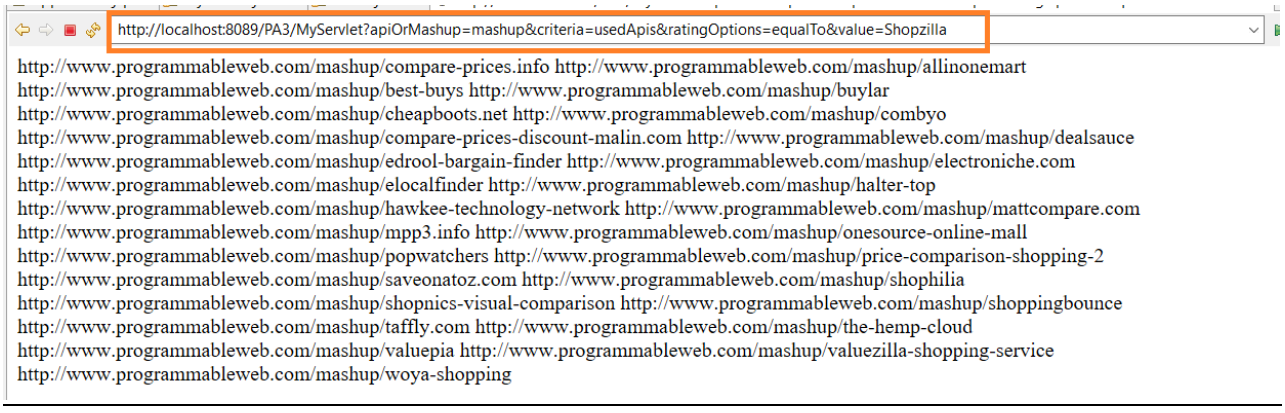
3. Keywords: mashups





Based on criteria: APIs


Application.jsp
MyServlet.java
Parser.java
Web Services Assignment 3
http://localhost:8089/PA3/Application.jsp
WEBSERVICES PROGRAMMING ASSIGNMENT 3
SEARCH APIs/MASHUPs:
Select: Mashups
Search by keywords or criteria
Keywords
Updated year
Protocols
Category
Tags
Used Apis
Rating
Equal To
Enter value: Shopzilla
Submit


http://localhost:8089/PA3/MyServlet?apiOrMashup=mashup&criteria=usedApis&ratingOptions=equalTo&value=Shopzilla
http://www.programmableweb.com/mashup/compare-prices.info http://www.programmableweb.com/mashup/allinonemart
http://www.programmableweb.com/mashup/best-buys http://www.programmableweb.com/mashup/buylar
http://www.programmableweb.com/mashup/cheapboots.net http://www.programmableweb.com/mashup/combyo
http://www.programmableweb.com/mashup/compare-prices-discount-malin.com http://www.programmableweb.com/mashup/dealsauce
http://www.programmableweb.com/mashup/edrool-bargain-finder http://www.programmableweb.com/mashup/electroniche.com
http://www.programmableweb.com/mashup/elocalfinder http://www.programmableweb.com/mashup/halter-top
http://www.programmableweb.com/mashup/hawkee-technology-network http://www.programmableweb.com/mashup/mattcompare.com
http://www.programmableweb.com/mashup/mpp3.info http://www.programmableweb.com/mashup/onesource-online-mall
http://www.programmableweb.com/mashup/popwatchers http://www.programmableweb.com/mashup/price-comparison-shopping-2
http://www.programmableweb.com/mashup/saveonatoz.com http://www.programmableweb.com/mashup/shophilia
http://www.programmableweb.com/mashup/shopnics-visual-comparison http://www.programmableweb.com/mashup/shoppingbounce
http://www.programmableweb.com/mashup/taffly.com http://www.programmableweb.com/mashup/the-hemp-cloud
http://www.programmableweb.com/mashup/valuepia http://www.programmableweb.com/mashup/valuezilla-shopping-service
http://www.programmableweb.com/mashup/woya-shopping