Dynamic Content Management Project:
A Query Engine for Web Service Compositions

Report By: OUMOUSS EL MEHDI (Master 2 Data & Knowledge)

Goal of the Project: Create a framework that enables the composition of several web services.

Methodology: Figure 1. provide an overview of the query engine to build.
This is going to be done in two stages:

1. The Web Service Description and Wrapper (lab 1) :

Description of the web services using XML and transformation into tuples using XSLT files.

2. The Execution Engine (lab 2) :

Parse the query expression, execute necessary call composition plans and output results.

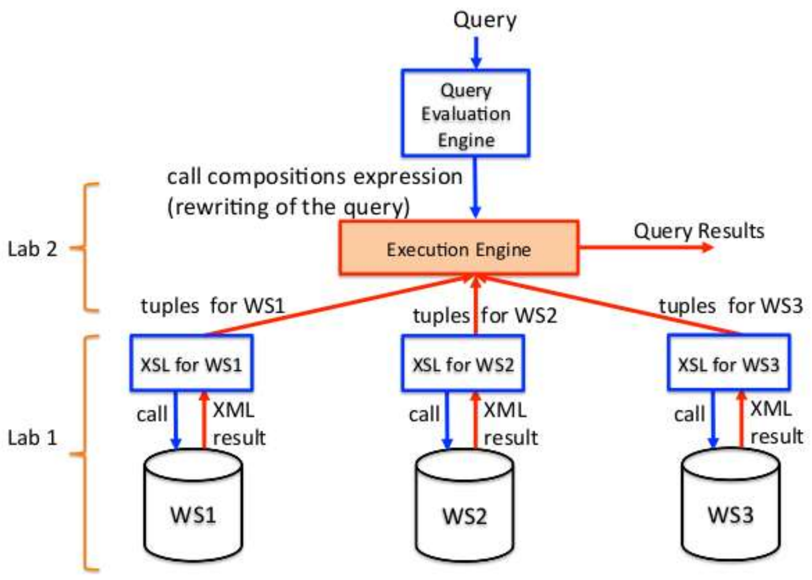


Figure 1. Overview of the Query Engine

I. Web Service Description and Wrapper

The Web Service API that we are going to work with is: the **Music Brainz Web service API** available at: http://musicbrainz.org/ws/2/

We are going to provide a description for 3 web services from this API, namely:

- **mb_getArtistInfoByName:** Given an artist name, it returns information related to this artist and most importantely his id.
- **mb_getAlbumInfoByArtistId:** Given an artist Id, we can have the artist's set of albums.
- **mb_getSongsByAlbumID**: Given an album Id, the WS returns the songs within the album.

These 3 Web services can be composed in order to answer more complex queries.

1. XML Description:

The Web Services are described using an XML document that conforms to the schema of the given web service. This is done, in a first time on the headVariables tag where we explicitly define input and outputs of the web service (signature).

```
<headVariables>
   <variable type="inout" name="?artistId"/>
   <variable type="output" name="?albumId"/>
   <variable type="output" name="?albumName"/>
</headVariables>
```

In a second time, we define also how to compose the web call, in the call tag, where we describe the static (constant) and dynamic (input) part of the call syntax.

```
<call>
<part type="constant" value="http://musicbrainz.org/ws/1/release/?artistid="/>
<part type="input" variable="?artistId" class="singer" separator="+" />
</call>
```

Finally, the description is linked to a XSLT file, for necessary transformation.

```
<transformation file="mb_getAlbumInfoByArtistId.xsl"/>
```

2. XSLT mapping:

Since our application aims at integrating data from a set of Web services, we prefer to reason in matter of tuples instead of the XML document (call results). The XSLT document is responsible of transforming the call results into tuples conforming to the abstract description of the WS.

```
<RESULT>
  <xsl:for-each
select="*[local-name()='metadata']/*[local-name()='release-list']/*[local-
name()='release']">
    <RECORD>
     <ITEM ANGIE-VAR='?artistId'><xsl:value-of select="mb:artist/@id"/></ITEM>
     <ITEM ANGIE-VAR='?albumId'><xsl:value-of select="@id"/></ITEM>
     <ITEM ANGIE-VAR='?albumName'><xsl:value-of select="mb:title"/></ITEM>
    </RECORD>
  </xsl:for-each>
</RESULT>
```

Basically, the XSLT file is some sort of loop function that goes through the XML documents and whenever there is a match, it returns the data in the format that we specified.

In a nutshell, the whole framework at this stages can be described as follow:

1. The XML description is read in order to construct the request expression (URL):
   Ex : http://musicbrainz.org/ws/1/artist/?name=Michael+Jackson

2. Using the previous request, we are able to call the web service.

3. The web service returns results in an XML (or JSON) format:

```
<metadata xmlns:ext=http://musicbrainz.org/ns/ext-1.0#…>
   <artist id="f27ec8…dbecf5e" ext:score="100" type="Person">
      <name>Michael Jackson</name>
      <sort-name>Jackson, Michael</sort-name>
      <disambiguation>"King of Pop"</disambiguation>
      <life-span end="2009-06-25" begin="1958-08-29"/>
   </artist>
</metadata>
```

4. First we store the results then we transform it into tuples using the XSLT file:
   Ex: (Michael Jackson, f27ec8…dbecf5e, …)

II. Execution Engine

As said before, our Execution Engine parse a query expression (see Input example), then executes the different calls in the order specified in the input and do necessary compositions between WS results and finally outputs results.

Input Expression:

> getArtistInfoByName iooo (**"Artist Name"**, **?artistId**, ?b, ?e)
> \# getAlbumByArtistId(**?artistId**, **?albumId**, ?albumName)
> \#getSongByAlbumId(**?albumId**, songTitle, year)

1. Query Parsing:

In this implementation, the query parsing is quite simple. We parse the given expression in input, in order to extract two essential piece of information:

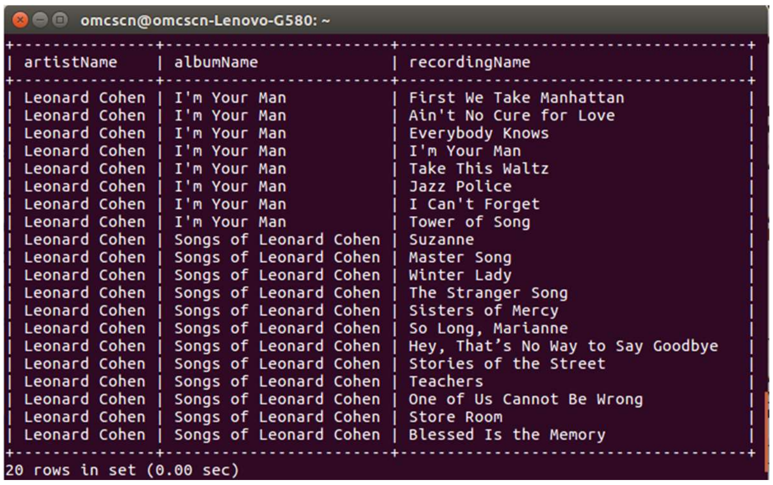
- The web services to use.
- The input parameter (here it is the artist name)

Once this is done, the process of calling web services and composing their results is lunched.

N.B: Since our input has one format, for convenience, we just ask the user for the "Artist Name". However, in the background, the Input Expression is being parsed.

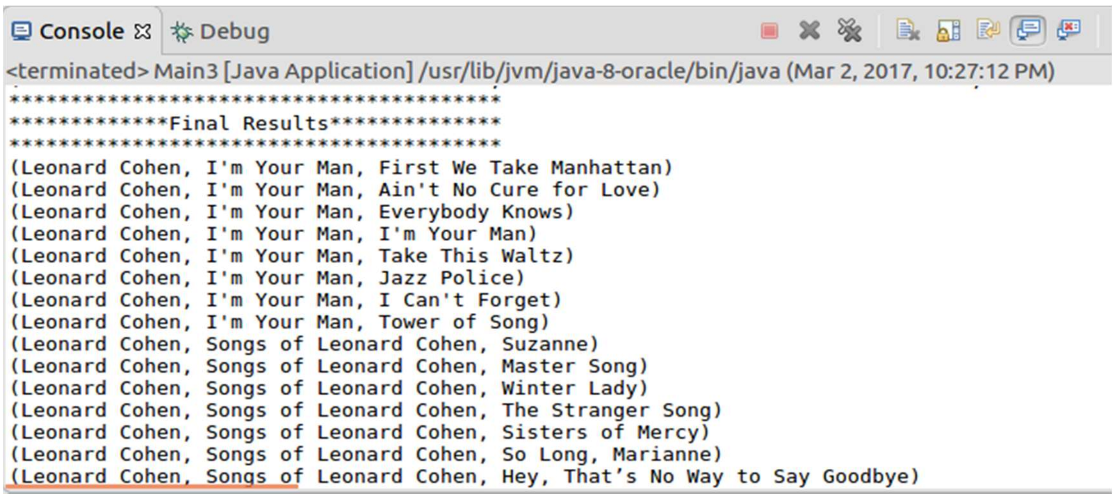
2. Results Compositions:

The composition of the web services is done in an intuitive way. In other words, the second web service is going to be called from the results given by the first WS and the same for the third WS, where the input will depend on the results of the second WS. Results of each web service are being stored in a MySQL database, with a table for each web service. Finally, at the database level, we compose all tables' results through a view.



```
omcscn@omcscn-Lenovo-G580: ~
+---------------+-----------------------------+---------------------------------+
| artistName    | albumName                   | recordingName                   |
+---------------+-----------------------------+---------------------------------+
| Leonard Cohen | I'm Your Man                | First We Take Manhattan         |
| Leonard Cohen | I'm Your Man                | Ain't No Cure for Love          |
| Leonard Cohen | I'm Your Man                | Everybody Knows                 |
| Leonard Cohen | I'm Your Man                | I'm Your Man                    |
| Leonard Cohen | I'm Your Man                | Take This Waltz                 |
| Leonard Cohen | I'm Your Man                | Jazz Police                     |
| Leonard Cohen | I'm Your Man                | I Can't Forget                  |
| Leonard Cohen | I'm Your Man                | Tower of Song                   |
| Leonard Cohen | Songs of Leonard Cohen      | Suzanne                         |
| Leonard Cohen | Songs of Leonard Cohen      | Master Song                     |
| Leonard Cohen | Songs of Leonard Cohen      | Winter Lady                     |
| Leonard Cohen | Songs of Leonard Cohen      | The Stranger Song               |
| Leonard Cohen | Songs of Leonard Cohen      | Sisters of Mercy                |
| Leonard Cohen | Songs of Leonard Cohen      | So Long, Marianne               |
| Leonard Cohen | Songs of Leonard Cohen      | Hey, That's No Way to Say Goodbye |
| Leonard Cohen | Songs of Leonard Cohen      | Stories of the Street           |
| Leonard Cohen | Songs of Leonard Cohen      | Teachers                        |
| Leonard Cohen | Songs of Leonard Cohen      | One of Us Cannot Be Wrong       |
| Leonard Cohen | Songs of Leonard Cohen      | Store Room                      |
| Leonard Cohen | Songs of Leonard Cohen      | Blessed Is the Memory           |
+---------------+-----------------------------+---------------------------------+
20 rows in set (0.00 sec)
```

Figure 2. Screenshot from the DB (The final view)



```
<terminated> Main3 [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Mar 2, 2017, 10:27:12 PM)
****************************************
*************Final Results*************
****************************************
(Leonard Cohen, I'm Your Man, First We Take Manhattan)
(Leonard Cohen, I'm Your Man, Ain't No Cure for Love)
(Leonard Cohen, I'm Your Man, Everybody Knows)
(Leonard Cohen, I'm Your Man, I'm Your Man)
(Leonard Cohen, I'm Your Man, Take This Waltz)
(Leonard Cohen, I'm Your Man, Jazz Police)
(Leonard Cohen, I'm Your Man, I Can't Forget)
(Leonard Cohen, I'm Your Man, Tower of Song)
(Leonard Cohen, Songs of Leonard Cohen, Suzanne)
(Leonard Cohen, Songs of Leonard Cohen, Master Song)
(Leonard Cohen, Songs of Leonard Cohen, Winter Lady)
(Leonard Cohen, Songs of Leonard Cohen, The Stranger Song)
(Leonard Cohen, Songs of Leonard Cohen, Sisters of Mercy)
(Leonard Cohen, Songs of Leonard Cohen, So Long, Marianne)
(Leonard Cohen, Songs of Leonard Cohen, Hey, That's No Way to Say Goodbye)
```

Figure 3. Results of the composition implementation.