Project Report

Professor: Nicoleta Preda
Student: Carlos Pérez

Parsing

The main class for the project is Called Engine.java. This class receives in the parameters. the calls of the distinct web services in the order specified in the project requirements. This engine is specific for the three web services described in the instructions. Below is an example of the call that the engine receives as argument.

"getArtistInfoByNameiooo(\"Aerosmith\", ?id, ?b, ?e)#getAlbumByArtistIdioo(?id, ?aid, ?albumName)#getSongByAlbumIdiooo(?aid,?arid,?rec, ?date)"

The parser creates an object of type WebService1 for each call and stores them in a list. Each of these objects contains. If the parameter contains "" the parser will assume is a constant value and will store it in the constant inputs list. Otherwise it will store it in the inputs string. Depending on the i's and the o's before the parenthesis the arguments will be stored as input or output.

Execution

To execute the code you have to modify the folder settings of the project with the local folders for the evaluation and definitions folder. Call the Engine class with one parameter containing the call for example.

"getArtistInfoByNameiooo(\"Aerosmith\", ?id, ?b, ?e)#getAlbumByArtistIdioo(?id, ?aid, ?albumName)#getSongByAlbumIdiooo(?aid,?arid,?rec, ?date)"

This will parse the call and call the web services in the sequence of the call. This engine makes use of the existing classes in the project to output the results to folders with the name of the id's. For each call the call results are stored in the call folders and the transformations to the web service are stored in the transformations folder.

To launch the appropriate web service the engine compares the name of the web service with the tree possibilities and chooses the right one. After each call and transformation, the web service verifies if there is another web service in the list that needs to be executed. If there is, it compares its output with the next web service's inputs and filters the fields accordingly. Then it stores this information in a list for the next web service to use.

Output

The output of the web services are stored in the folders transf_results and call_results under the name of each web service. If there are no results for an intermediate web service the next web service in line won't be called. Additionally if there are results for the final web service in the call these will be outputed in the command line as in the corresponding transfer a file.

Shortcomings

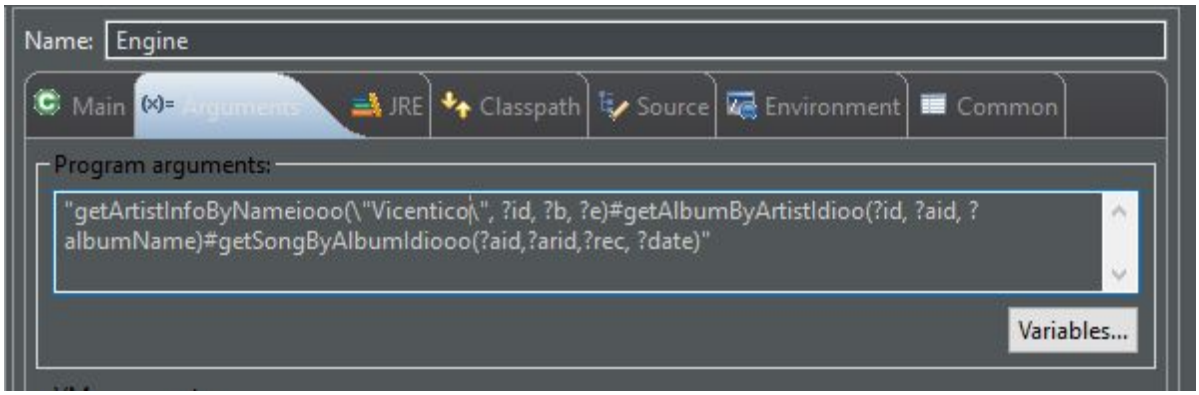
The code is too specific because it only takes into account the three web services described before.

```
switch (name) {
case "getArtistInfoByName":
    ws = ws1;
    break;
case "getAlbumByArtistId":
    ws = ws2;
    break;
case "getSongByAlbumId":
    ws = ws3;
    break;
default:
    break;
}
```

A better approach would be to create the definition files dynamically so the user could call more web services. Also it would be interesting to modify the files dynamically so the user shouldn't be constrained to the parameters already defined in the web service definition.

Running example

Let's make from eclipse the following web service call by passing the following query to the Engine class.



The first part will be the parsing the Parser will split the call in the following strings according to the hashtag.

getArtistInfoByNameiooo(\"Vicentico\", ?id, ?b, ?e)
getAlbumByArtistIdioo(?id, ?aid, ?albumName)
getSongByAlbumIdiooo(?aid,?arid,?rec, ?date)

Then the program will parse the substring and classify the arguments in inputs and outputs. The name of the web service is stored in a variable. And the inputs will be classified in two list either constants or others.

getArtistInfoByNameiooo(\"Vicentico\", ?id, ?b, ?e)
getAlbumByArtistIdioo(?id, ?aid, ?albumName)
getSongByAlbumIdiooo(?aid,?arid,?rec, ?date)

Once this classifications are made they are stored in an object and these objects in a list. Now at runtime the program will call the first web service and according to the web service name it will load the correct one. Using the inputs that in the case of the first web service is the constant "Vicentico" it makes the call to the web service using the xml file getArtistInfoByName.xml and the code given to work on the project. This call will retrieve the result of calling the web service using the given input. Then by using the xsl we retrieve the outputs of the web service that we are interested in saving. These are the artist name the artist id the begin date and the end date.

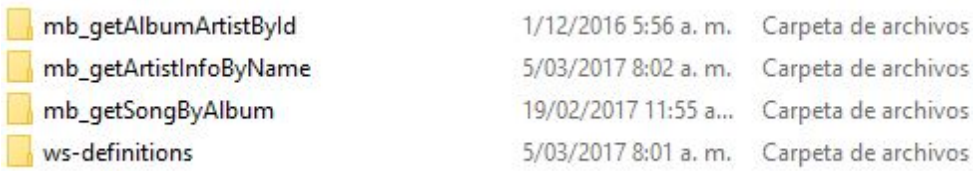
Once we have the transformation data we compare the web service call with that of the next web service and we identity the output arguments of the first that are inputs of the second. Then the next step is to filter the result to get only the needed arguments and we store those elements in a list that the web service getAlbumByArtistId.

getArtistInfoByNameiooo(\"Vicentico\", ?id, ?b, ?e)
getAlbumByArtistIdioo(?id, ?aid, ?albumName)

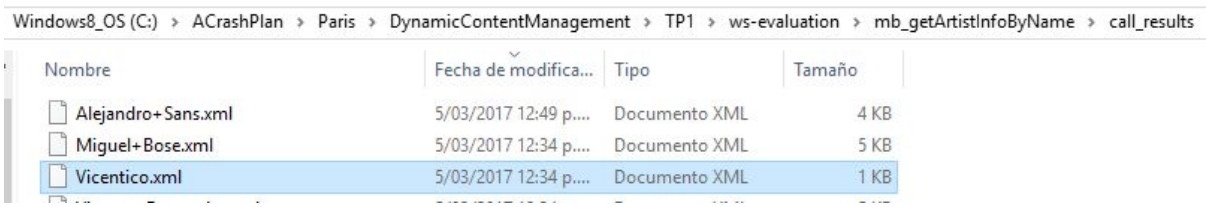
The process of calling the web service starts again and it repeats itself with the couple getAlbumArtistById and getSongByAlbumId. The only difference is that the last web service doesn't compare with anyone as there are no more web services to compare to. In the same way as it is the last web service besides saving the call and transformation results in files this last time the results of the transformation will be printed in the console one line per element.

```
a6e33ed0-796f-440c-aa22-13fe1740fe7c
12d33d7a-91ea-4418-9409-e96736e4d72b
Algo de ti
1991
a6e33ed0-796f-440c-aa22-13fe1740fe7c
12d33d7a-91ea-4418-9409-e96736e4d72b
Te falta corazón
1991
a6e33ed0-796f-440c-aa22-13fe1740fe7c
12d33d7a-91ea-4418-9409-e96736e4d72b
```

Now let's look into the folder s



In each of these we can find the call results and transformation results. Let's verify that the call results for the web services have been made. For the web service getArtistInfoByName we can see the call result has been made.




```
− <metadata>
  − <artist-list count="2" offset="0">
    − <artist id="fcb476f9-d17f-4058-ab30-520e590df076" type="Person" ext:score="100">
        <name>Vicentico</name>
        <sort-name>Vicentico</sort-name>
        <disambiguation>Argentine singer</disambiguation>
        <life-span begin="1964-07-24"/>
      </artist>
    − <artist id="12d33d7a-91ea-4418-9409-e96736e4d72b" type="Person" ext:score="62">
        <name>Vicentico Valdés</name>
        <sort-name>Valdés, Vicentico</sort-name>
        <life-span begin="1921-01-10" end="1995-06-26"/>
      </artist>
    </artist-list>
  </metadata>
```

And also we can examine the transformation file that was created from this one.

```
<RESULT>
 - <RECORD>
      <ITEM ANGIE-VAR="?artistName">Vicentico</ITEM>
      <ITEM ANGIE-VAR="?artistId">fcb476f9-d17f-4058-ab30-520e590df076</ITEM>
      <ITEM ANGIE-VAR="?beginDate">1964-07-24</ITEM>
      <ITEM ANGIE-VAR="?endDate"/>
   </RECORD>
 - <RECORD>
      <ITEM ANGIE-VAR="?artistName">Vicentico Valdés</ITEM>
      <ITEM ANGIE-VAR="?artistId">12d33d7a-91ea-4418-9409-e96736e4d72b</ITEM>
      <ITEM ANGIE-VAR="?beginDate">1921-01-10</ITEM>
      <ITEM ANGIE-VAR="?endDate">1995-06-26</ITEM>
   </RECORD>
</RESULT>
```

Finally we can examine the transformation file for the last web service to verify the whole query was executed.

```
- <RESULT>
  - <RECORD>
      <ITEM ANGIE-VAR="?releaseId">a6e33ed0-796f-440c-aa22-13fe1740fe7c</ITEM>
      <ITEM ANGIE-VAR="?artistId">12d33d7a-91ea-4418-9409-e96736e4d72b</ITEM>
      <ITEM ANGIE-VAR="?recordingTitle">Algo de ti</ITEM>
      <ITEM ANGIE-VAR="?date">1991</ITEM>
    </RECORD>
  - <RECORD>
      <ITEM ANGIE-VAR="?releaseId">a6e33ed0-796f-440c-aa22-13fe1740fe7c</ITEM>
      <ITEM ANGIE-VAR="?artistId">12d33d7a-91ea-4418-9409-e96736e4d72b</ITEM>
      <ITEM ANGIE-VAR="?recordingTitle">Te falta corazón</ITEM>
      <ITEM ANGIE-VAR="?date">1991</ITEM>
    </RECORD>
```

This same results are printed in the console as shown here, one element per line.