Data Integration

Project Report

Michał Suliborski (a2019156841@isec.pt)

1. ANALYSIS OF THE DATA SOURCE(S)

After analysis of multiple movie's pages on Wikipedia I saw that all necessary data can be found in the table on the right side of the webpage. Not only it contains the title and image but also all needed data. Usually most of this data is repeated in the following parts of the article, yet gathering information from there would be very difficult for a computer program.

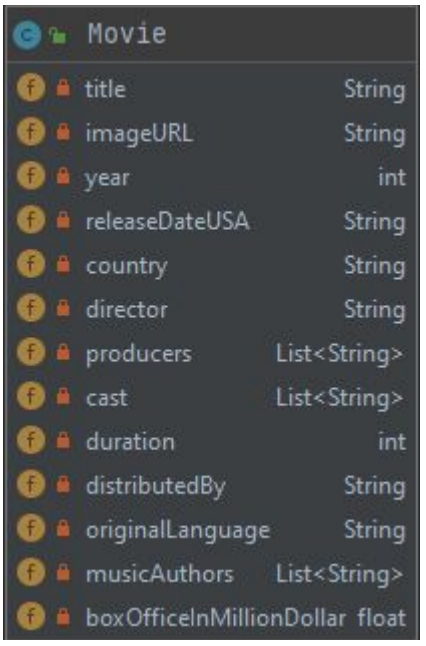
To make sure that the information is there though, one must be sure that they provide a title of the movie that matches the title of the Wikipedia page, for example by adding '(film)' at the end of the title. In case the movie has not been found it will simply not be added and the program will continue to work properly.

2. DEFINITION OF THE GLOBAL SCHEMA

Data scheme in my case is fairly simple, yet provides all necessary fields to contain all of the required data. Proposed schema looks as follows:

```
<movies>
  <movie>
    <title></title>
    <image-URL></image-URL>
    <release-date-USA></release-date-USA>
    <year></year>
    <country></country>
    <director></director>
    <producers>
      <producer></producer>
      ...
    </producers>
    <cast>
      <actor></actor>
      ...
    </cast>
    <duration></duration>
    <distributed-by></distributed-by>
    <original-language></original-language>
    <music>
      <author></author>
      ...
    </music>
    <box-office></box-office>
  </movie>
</movies>
```

3. IMPLEMENTATION OF THE WRAPPERS



After providing the program with a title of the movie, it downloads the source code of the webpage in form of String object with help of slightly modified version of HttpRequest function provided on Moodle. The next task is to convert that source code into an object implemented in the program. Structure of the object is as on the image to the right:

All of that mapping takes place in the `public static Movie sourceCodeToMovie(String sourceCode)` function.

One can distinguish 3 main parts of gathering data: title, image URL and others. In the first phase the program obtains the original (meaning, not modified by Wikipedia) title of the movie. It is accomplished using the following RegEx expression:

```
<h1 id="firstHeading" class="firstHeading"
lang="en"><i>(.*?)</i>
```

The expression is looking for any kind of characters that are in the `<h1>` tag with specific attributes.

Second phase is obtaining the image URL of the cover of the movie. It is done using the following RegEx expression:

```
<table .*?<img alt=".*?" src="(.*?)"
```

The key element of this expression is the '`<table`' part which assures us that the image will be inside of the main table. Second part is finding tag `<img>` and grouping content of `src` attribute.

Last, and probably most important part, is extracting the rest of the data. At first, the program extracts certain data into String list using the following expression:

```
(?<![a-z]=)["|>]([ A-Za-z\p{L}0-9()$&#;,-.]+?)["|<]
```

This RegEx basically finds all the characters between '>' and '<' or in quotes, with the exception of the content of attributes (which was achieved with look-behind `(?<![a-z]=)`). Such a prepared list of strings is then put through a loop. Dependandly of what is the content of each string (or cell) the program decides what to do. If it finds a key word such as 'Director' or 'Country' it switches mode to write, so that the next piece of data will be saved in the Movie object. After it is saved, program mode is switched back to search. Search is finished after obtaining box office data (because it is always last category) or simply at the end of file (because till the end of loop mode is set to search, which will not override or add any more data).

Having the objects ready, the program uses `public static void moviesToNewXMLFile(List<Movie> movies, String fileName)` function to write the list of Movie objects into a new XML file. This is achieved by using JDOM2 API, that makes it easier to create an XML file and populate it with desired data. Along the way various other functions are used, that converts between data types and helps to avoid repentance of the code.

4. GENERATION AND MANIPULATION OF THE XML FILE

As intended in the task description program allows the user to add, delete and edit data. Adding the movie takes the title as parameter (title must be the same as it is in Wikipedia), then the program downloads source code, puts it through the wrappers and adds it into the XML file. In case the XML file does not exist it is created (as it is during generating sample data) or if the given title already exists it is simply skipped.

Program allows the user to delete movies, also by providing the title of the movie that the user wants to delete. If the title does not exist in the XML file the function is skipped.

Finally the program has been implemented with the possibility to edit movies's year, director and country. In order to do so one needs to provide both the title of the movie he wants to alter and a new value of the field.

5. VALIDATION OF THE XML FILE

The XML file can be validated with both DTD and XSD at any time simply by clicking 'Validate with DTD' or 'Validate with XSD' buttons.

6. SEARCH INFO USING XPATH

Users can generate and view outputs of 8 (5 base and 3 additional) preconfigured xPath searches by clicking the proper button in GUI and filling corresponding search parameters. Available searches are:

- Search for a movie title and show relevant information
- Search for films by a given director
- Search for films with the participation of a given actor / actors
- Search for movies with a duration between a given interval
- Search for films from a given country
- Search for movies with a year between a given interval
- Search for movies with a box office between a given interval
- Search for films in a given language

7 GENERATE OUTPUTS WITH XSLT/XQUERY

The user can generate and view both outputs and XSL files themselves (6 XSL files and 6 outputs). Transformations were implemented by using SAXON API and functions provided on Moodle. Available transformations are:

- HTML file of photos of the films
- XML file that shows the listing of the films of a given director
- TXT file that shows the films of a given country.
- TXT file that shows movies with their release's year sorted ascending by year (additional transformation)
- TXT file that shows movies with the amount of actors in it (additional transformation)
- XML file that shows only non-english movies (additional transformation)

8. GUI INTERFACE

Graphical user interface was implemented with the usage of JavaFX library. Half of the application's window is reserved for outputs and viewing content of the files. Below that there are buttons allowing the user to validate XML file with DTD or XSD at any time and also button generating sample data from over 30 movies. On the right hand side there is a control panel allowing the user to view XML, DTD and XSD files. Then, buttons and text fields to manipulate data (add, delete or edit) and below that there are xPath searches with parameters. Finally, at the bottom, there are 12 buttons allowing user to both see the XSL transformation and XSL files.