Project 1

Enterprise Application Integration (MIEBIOM)
Department of Informatics Engineering
Delivery date: 10 March 2017

Objectives

- Learn XML and how to process XML using JAXB
- Learn XML Schema and XSTL

Final Delivery

- You must submit your project in a zip file using Inforestudante. Do not forget to associate your work colleague during the submission process.
- The submission contents are:
  - Source code of the requested applications ready to compile and execute.
- After submitting, you are required to register the (extra-class) effort spent solving the assignment. This step is mandatory. Please fill the effort form at:
  https://docs.google.com/spreadsheet/viewform?formkey=dHFha3NuWE1pYWJDbTJMTWcxTWRMS1E6MA

Grading

- Quality of the data model used for representing data (XML/XSD);
- Quality of the code (organization, modularity, formatting, code conventions, etc.);
- Simplicity of the solution;
- Final presentation of the work.

Resources

- Java Platform (**JDK**) – http://www.oracle.com/technetwork/java/javase/downloads
- **Eclipse IDE for Java EE Developers** – http://www.eclipse.org/downloads/
- **XML** Technologies – http://www.w3schools.com/xml
  - Check the Schema; XPath; XSLT links
- **JAXB Tutorial** – http://jaxb.java.net/tutorial/index.html
- **Trang** – http://www.thaiopensource.com/relaxng/trang.html
- **HTML** basic information – http://www.w3schools.com/html/

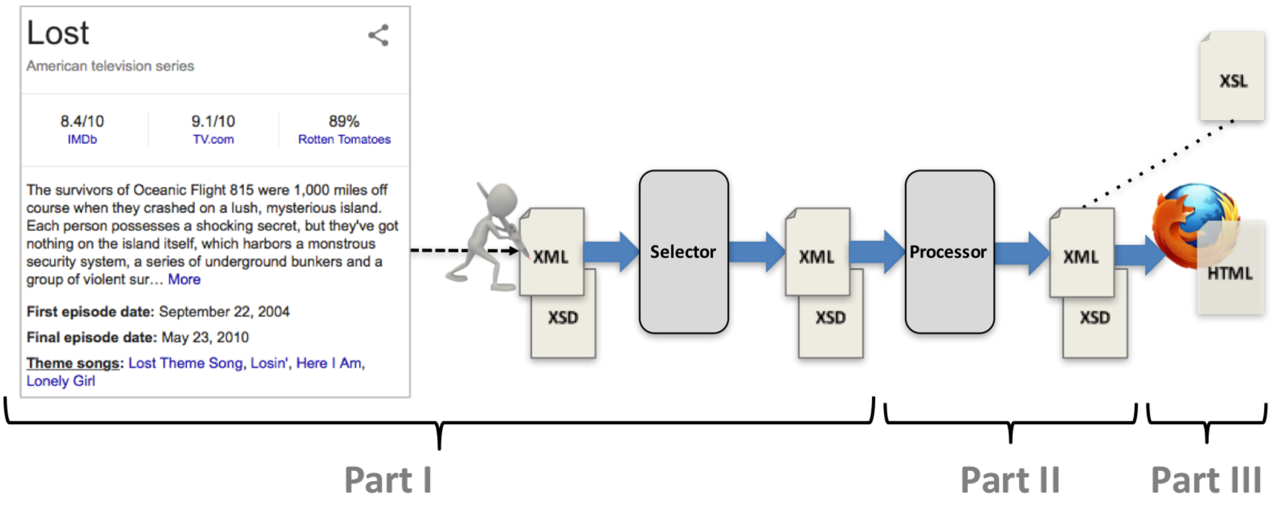
Project Description

In this project, you will be using XML technologies and techniques for reading, validating, processing, and writing XML documents. This will be accomplished using the Java programming language, JAXB, XML Schema, and XSLT.

In this assignment, you will create a set of applications that process and categorize information about television series extracted from an on-line source. The project includes the following parts:

- **Part I – Selector**: Reads an XML document holding information about a set of tv series, including detailed information about each series (e.g., cast, seasons, episodes), validates, processes the available data, and writes a shorter XML document (based on user preferences).
- **Part II – Processor:** Converts the XML document, produced in Part I, to another XML document (which essentially reorganizes the information and adds some statistics).
- **Part III – HTMLViewer:** Converts the XML document, produced in Part II, to HTML using XSLT.

Figure 1 shows the overall scenario of this project. The three parts that compose the project are described in the following paragraphs.


Lost
American television series

8.4/10	9.1/10	89%
IMDb	TV.com	Rotten Tomatoes

The survivors of Oceanic Flight 815 were 1,000 miles off course when they crashed on a lush, mysterious island. Each person possesses a shocking secret, but they've got nothing on the island itself, which harbors a monstrous security system, a series of underground bunkers and a group of violent sur… More

First episode date: September 22, 2004
Final episode date: May 23, 2010
Theme songs: Lost Theme Song, Losin', Here I Am, Lonely Girl

Figure 1 – The information flow

Part I – Selector

The source of information for this project is **google.co.uk**[1]. In Part I, you will create an application, named *Selector*, that reads information in XML format (created by yourself, and based on the information present on the website) and produces a shorter version (also in

[1] **https://www.google.co.uk/#q=game+of+thrones**
Examine a few pages, such as this one, to understand the overall structure and content, in order to be able to define the right data model. You should consider the information that is placed at the right-hand side of the page (a small part of the information to consider is visible in Figure 1).

XML), based on user preferences. Note that the information to be read is not about a single series, but about a set of series.

You should start by analyzing what is the important information to retrieve and create a sample XML file (based on the information provided by the site) to be later processed by the *Selector*. All XML files defined/produced (i.e., the application input and output) should have an associated XML Schema. The XML Schemas allow you to generate Java classes that can be used by an application to hold and manipulate the information.

In practice, to create the *Selector* application, you can do the following:

- Define the XML Schema(s). This may involve the use of *trang* to create XSD from XML. Verify and tune your generated Schema(s).
- For each XML Schema, generate the respective Java classes using the XML binding compiler, *xjc*.
- Once you have the Java classes that can keep the data, you can instantiate and use them normally in your code.

Each time the *Selector* runs, it reads and validates the input XML, filters the data (i.e., discards some items, according to specific user rules) and then produces XML output. The application user should be able to specify one rule that will discard specific series**.** The rule must allow logical 'and' and 'or' combinations and must target the actors, seasons, and ratings (the three items, or less). For example, a user may specify a rule to select series where 'Michael Emerson' plays a role *and* the series has at least 3 seasons *or* the average rating of the series is over 80%, which means that all other cases will be excluded from the application output.

Part II – Processor

This application organizes the information produced by the *Selector* by actors. So, in practice, the output XML will have a list of actors, each actor holding a list of alphabetically sorted series. In addition, in each XML file produced, the *Processor* adds some statistics, which include the total number of actors processed and the name of 3 actors with the highest number of participations in episodes (we will be assuming that an actor participates in all episodes of all seasons of a given series).

Part III – HTMLViewer

In this part of the project you will define an XSL Transformation for converting the XML file produced by the *Processor* to HTML. Use a web browser with a built-in XSLT engine (e.g., Firefox) to apply the transformation and display the resulting HTML page.