Project 1

Enterprise Application Integration (MIEBIOM)
Department of Informatics Engineering
Delivery date: see Inforestudante/Submissao de Trabalhos

Objectives

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

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.
  - Report

The REPORT is expected to be complete in the sense that it needs to contain all necessary and sufficient information for the teacher to give the score to the evaluation item, even without having to run the code itself and without having to meet with the group in person for evaluation. For that you can include descriptions, screenshots, code extracts, whatever is needed for a complete and thorough evaluation. If the report is absent the score is 0, and if it is incomplete the score is significantly affected. This is to make sure you do have a complete report.

Before the main body, the REPORT starts with the complete identification of the students and group, then a table of contents, then the following:

a. Lists of what the group succeeded to do and what is missing
b. self-evaluation of the group (0-100%)
c. List of what each student contributed (no repetitions)
d. self-evaluation of each student in the group (0-100%)
e. hours of effort by each student separately

These items a,b,c are important for the teacher to check whether his evaluation coincides more or less with what the group and student thinks.

Grading

- REPORT (confirmed later by defense)
  - Overall 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/
- **JAXB Compiler** (under the bin/ directory)
  - https://repo1.maven.org/maven2/com/sun/xml/bind/jaxb-ri/2.3.0/jaxb-ri-2.3.0.zip

- **Trang** - http://central.maven.org/maven2/org/relaxng/trang/20181222/trang-20181222.jar (from http://www.thaiopensource.com/relaxng)
- **JAXB Libraries**
  - https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api/2.3.0
  - https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime/2.3.0
  - https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-core/2.3.0
  - https://mvnrepository.com/artifact/javax.activation/activation/1.1.1
- **XML** Technologies – http://www.w3schools.com/xml
  - Check the Schema; XPath; XSLT links
- **JAXB Tutorial** – http://jaxb.java.net/tutorial/index.html
- **HTML** basic information – http://www.w3schools.com/html/

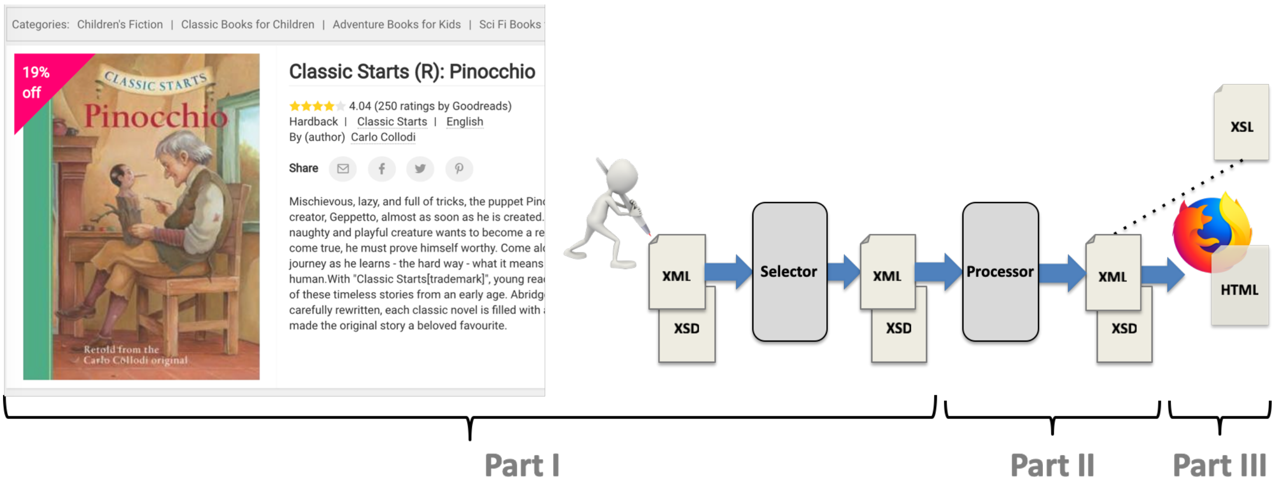
Project Description

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 integrate, process, and categorize information about children's books 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 books retrieved from BookDepository's online Store, including detailed information about each book (e.g., title, editor, book dimensions, price),* 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 information).
- **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.



Categories: Children's Fiction | Classic Books for Children | Adventure Books for Kids | Sci Fi Books

Classic Starts (R): Pinocchio

4.04 (250 ratings by Goodreads)

Hardback | Classic Starts | English

By (author) Carlo Collodi

Share

Mischievous, lazy, and full of tricks, the puppet Pin
creator, Geppetto, almost as soon as he is created.
naughty and playful creature wants to become a re
come true, he must prove himself worthy. Come al
journey as he learns - the hard way - what it means
human.With "Classic Starts[trademark]", young rea
of these timeless stories from an early age. Abridg
carefully rewritten, each classic novel is filled with
made the original story a beloved favourite.

Figure 1 – The information flow

Part I – Selector

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

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:

- Add the JAXB libraries to your project
- 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 rules that will discard specific books. The rules must allow targeting at least the author, title, rating, number of ratings, bestsellers rank, and publication date. It should be possible to place a rule for all criteria (connected with 'AND' logical connectors) or for just a few. For example, a user may specify a rule to select books by 'Antoine de Saint-Exupery' with a review rating over 4 and published after 1995, which means that all other cases will be excluded from the application output.

Part II – Processor

This application reorganizes the information produced by the *Selector* by author. So, in practice, the output XML will have a list of authors, each author holding a list of books sorted by bestsellers rank. In addition, in each XML file produced, the *Processor* adds some statistics, which include the total number of distinct authors processed and the name of N authors that have the books with highest bestsellers rank scores (N should be easily set by the application user).

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. You may use a web browser with a built-in XSLT

[1] **https://www.bookdepository.com/Three-Little-Pigs-Disney-Classic-Random-House-Disney/9780736423120**
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.

engine (e.g., Firefox) to apply the transformation and display the resulting HTML page or use Java code directly in the processor application to generate the HMTL.