| | |
|---|---|
| FCTUC<br>UNIVERSIDADE DE COIMBRA<br>FACULDADE DE CIÊNCIAS E TECNOLOGIA<br>***Departamento de Engenharia Informática*** | **Project #1**<br>**Integração de Sistemas/**<br>**Enterprise Application Integration**<br><br>**2012/13 – 1st Semester**<br><br>MEI/MSE<br><br>**Deadline: 2012-10-12** |

**<u>Nota</u>: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.**

# XML and XML Manipulation, Java Message Service and Message Oriented Middleware

## Objectives

- Learn **XML technologies**. In particular, you will learn XML, XSD, XSL, XSLT and XPATH. This project is mostly about XML processing.
- Understand the technique of "**Screen Scraping**". Screen scraping consists in parsing the information shown in a terminal so that it can be used on a different system. It is the technique used for application integration where the only access point to an application is through its user interface (e.g., a venerable VT100 text terminal). Since, nowadays, web systems are ubiquitous, screen scraping is mostly used to gather and process information from web sites that do not expose APIs to the general public (or their business partners).
- Remember (or learn) how to use **HTML parsers**. These parsers can read HTML code and create data structures representing the web page, such as DOM[1] documents. You *may* also need to resort to **regular expressions** to clean data available in the DOM document. Regular expressions are an extremely powerful mechanism for cleaning, gathering and processing data embedded in text files.

[1] Document Object Model.
[2] For example http://www.nextbolsa.com/cotacoes.php?action=psi.

## Final Delivery

- You must submit your project using Inforestudante.
- This assignment contains two parts: one is for training only, and does not count for the evaluation. You should only deliver the other part.
- The submission contents are:
  - Source code of the requested applications ready to compile and execute.
  - A small report (5 pages max) about the implementation of the project. Please use PDF. We will not open any Microsoft Word documents.

---

## Bibliography

jsoup

- jsoup Java HTML Parser, with best of DOM, CSS, and jquery: http://jsoup.org
- Use selector-syntax to find elements: jsoup Java HTML parser: http://jsoup.org/cookbook/extracting-data/selector-syntax

XML, XSD, XSL and XSLT

- XML: http://www.w3schools.com/xml
- XSD: http://www.w3schools.com/schema
- XPATH: http://www.w3schools.com/xpath
- "Chapter 2: Understanding XML", in J2EE 1.4 Tutorial http://download.oracle.com/javaee/1.4/tutorial/doc/
- JAXB Tutorial – Java.net: http://jaxb.java.net/tutorial/index.html

Processing XML/XSLT in Java

- "Chapter 7: Extensible Stylesheet Language Transformations", in J2EE 1.4 Tutorial
  (Especially, the part "How XPath Works" and "Transforming XML Data with XSLT")
  http://download.oracle.com/javaee/1.4/tutorial/doc/
- David Jacobs, "Rescuing XSLT from Niche Status – A Gentle Introduction to XSLT through HTML Templates", http://www.xfront.com/rescuing-xslt.html
- G. Ken Holman, "What is XSLT?", in XML.COM http://www.xml.com/lpt/a/2000/08/holman/index.html
  (Especially, the part "Getting started with XSLT and XPath")
- Paul Grosso and Norman Walsh, "XSL Concepts and Practical Use", in NWalsh.COM
  http://nwalsh.com/docs/tutorials/xsl/xsl/frames.html

Java Message Service

- JMS with JBoss AS 7: http://eai-course.blogspot.pt

Round Robin Database Tool

- RRDTool – About RRDtool:

http://oss.oetiker.ch/rrdtool/

**Advice:** Skim all the links above before starting to read anything in detail.
**Note:** You have short examples of some of the technologies you need in the end of this document.

## Training Part (doesn't count for evaluation)

1. Use JAXB to output the following XML:

a)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<class>
      <student>
            <name>Alberto</name>
            <age>21</age>
      </student>
      <student>
            <name>Patricia</name>
            <age>22</age>
      </student>
    <student>
            <name>Luis</name>
            <age>21</age>
      </student>
</class>
```

b)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<class>
      <student id="201134441110">
            <name>Alberto</name>
            <age>21</age>
      </student>
      <student id="201134441116">
            <name>Patricia</name>
            <age>22</age>
      </student>
    <student id="201134441210">
            <name>Luis</name>
            <age>21</age>
      </student>
</class>
```

c)

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<!-- Generated automatically. Don't change it. -->
<class xmlns="http://www.dei.uc.pt/EAI">
  <student xmlns="" id="201134441110">
    <name>Alberto</name>
    <age>21</age>
  </student>
  <student xmlns="" id="201134441116">
    <name>Patricia</name>
    <age>21</age>
  </student>
  <student xmlns="" id="201134441210">
    <name>Luis</name>
    <age>21</age>
  </student>
</class>
```

d)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated automatically. Don't change it. -->
<h:class xmlns:h="http://www.dei.uc.pt/EAI">
  <student id="201134441110">
    <name>Alberto</name>
    <age>21</age>
  </student>
  <student id="201134441116">
    <name>Patricia</name>
    <age>21</age>
  </student>
  <student id="201134441210">
    <name>Luis</name>
    <age>21</age>
  </student>
</h:class>
```

e)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="test.xsl"?>
<!-- Generated automatically. Don't change it. -->
<h:class xmlns:h="http://www.dei.uc.pt/EAI">
  <h:student id="201134441110">
    <name>Alberto</name>
    <age>21</age>
  </h:student>
  <h:student id="201134441116">
    <name>Patricia</name>
    <age>21</age>
  </h:student>
  <h:student id="201134441210">
```

```
    <name>Luis</name>
    <age>21</age>
  </h:student>
</h:class>
```

2. Next, you can use a tool like trang to automatically produce the XSD for the following XML. Change the XSD, to ensure that <direction> can only be one of “dgsg|boinc” or “dgsg|xtremweb”, while <timestamp> must be positive.

```
<?xml version="1.0" encoding="UTF-8"?>
<report timestamp="1308046204104" timezone="GMT" version="1.1">
<metric_data>
    <metric_name>cpus_available</metric_name>
    <timestamp>1308046204003</timestamp>
    <value>0.0</value>
    <type>uint32</type>
    <units>cpus</units>
    <spoof>EDGITest|fusion:EDGITest|fusion</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>gflops</metric_name>
    <timestamp>1308046204056</timestamp>
    <value>0.0</value>
    <type>float</type>
    <units>gflops</units>
    <spoof>EDGITest|fusion:EDGITest|fusion</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>past_workunits</metric_name>
    <timestamp>1308046204058</timestamp>
    <value>0.0</value>
    <type>uint32</type>
    <units>wus</units>
    <spoof>EDGITest|fusion:EDGITest|fusion</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>waiting_workunits</metric_name>
    <timestamp>1308046204059</timestamp>
    <value>0.0</value>
    <type>uint32</type>
    <units>wus</units>
    <spoof>EDGITest|dsp:EDGITest|dsp</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
```

```xml
  <metric_data>
    <metric_name>success_rate</metric_name>
    <timestamp>1308046204061</timestamp>
    <value>1.0</value>
    <type>float</type>
    <units>percentage</units>
    <spoof>EDGITest|dsp:EDGITest|dsp</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>past_workunits_24_hours</metric_name>
    <timestamp>1308046204064</timestamp>
    <value>0.0</value>
    <type>uint32</type>
    <units>wus</units>
    <spoof>EDGITest|fusion:EDGITest|fusion</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>cpus_available</metric_name>
    <timestamp>1308046204066</timestamp>
    <value>0.0</value>
    <type>uint32</type>
    <units>cpus</units>
    <spoof>EDGITest|dsp:EDGITest|dsp</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>success_rate</metric_name>
    <timestamp>1308046204067</timestamp>
    <value>1.0</value>
    <type>float</type>
    <units>percentage</units>
    <spoof>EDGITest|fusion:EDGITest|fusion</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
  <metric_data>
    <metric_name>gflops</metric_name>
    <timestamp>1308046204092</timestamp>
    <value>0.0</value>
    <type>float</type>
    <units>gflops</units>
    <spoof>EDGITest|dsp:EDGITest|dsp</spoof>
    <direction>dgsg|boinc</direction>
  </metric_data>
</report>
```

3. Now, manually write the XML Schema Definition for the XML of exercise 1e).

4. Write an XSL file capable of outputting the XML data into an HTML table. You can use the Xalan java library for this purpose.

---

## Project Part (for evaluation)

In this assignment you should create three applications. The first one is a probe that collects data from a web site with stock price data[2], summarizes these data using XML, and sends XML to a Java Message Service Topic. This topic serves two other applications that store the data for later visualization. One of the applications writes stock values to the Round Robin Database. The other application creates HTML files with the list of stock values that it receives. Refer to Figure 1.

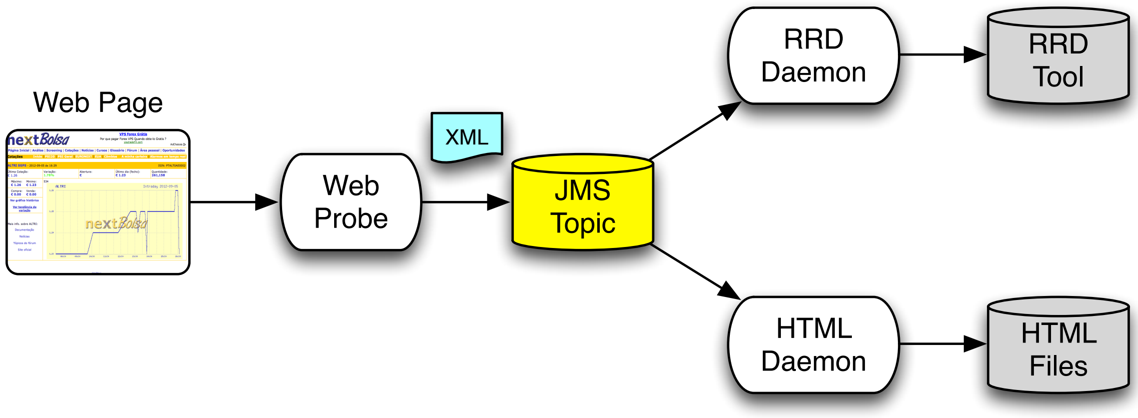

Figure 1 – The Data Flow

### The Probe

The probe is a stand-alone command-line application that reads a web page and sends an XML message to a Java Message Service topic. You should use an HTML parser, such as Jsoup, to get the data from the web page. **You should not parse the web page directly using regular expressions.** Nevertheless, you are allowed to use regular expressions to extract small pieces of data from the results of the HTML parser. For example, you could find a string that looks like "val: 3.11" and use regular expressions to extract the 3.11.

Once you get the DOM document of the web page, you will need to convert it to XML. You should do this as follows:

- Define the XML schema (this may involve the `trang` tool, to create XSD from XML). **You must include an XML schema file (XSD)** as part of your final submission and be ready to explain it;

[2] For example http://www.nextbolsa.com/cotacoes.php?action=psi.

- From the XML schema, generate the 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 in the normal way in the probe source code.

Each time the probe runs, it parses the web page, creates and initializes the Java classes that keep the web site's data, outputs an XML document to a JMS message and publishes this message on a JMS topic. If the topic is down for some reason, you may want to keep a log with the message that the probe was unable to publish, to retry later.

We suggest the following XML format for the message that enters the topic. However, you may find necessary to change it, either because you would like to include extra features, or because the site you use is different:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<cotacoes>
   <cotacao empresa="Am">
      <valor>3.3</valor>
      <subida>1.8</subida>
      <maximo>3.45</maximo>
      <minimo>3</minimo>
      <quantidade>43442</quantidade>
      <ultimodia>3.1</ultimodia>
   </cotacao>
   <cotacao empresa="Sl">
      <valor>4.25</valor>
      <subida>-0.3</subida>
      <maximo>4.45</maximo>
      <minimo>4.25</minimo>
      <quantidade>1044</quantidade>
      <ultimodia>4.47</ultimodia>
   </cotacao>
</cotacoes>
```

## RRD Daemon

This part will be included soon.

## HTML Daemon

This daemon should be permanently running, waiting for XML messages from the JMS topic. You should use durable subscriptions to ensure that even if the daemon fails, the topic will keep the messages for later retrieval. The daemon must create a good-looking HTML file, using the XML files coming from the topic. For that, you should use an XSL template. This template will include all the necessary rules for transforming the original XML file into HTML. These rules should be applied using Java XSLT libraries. You may also want to save the XML with slight modifications to enable a simple web browser with a built-in XSLT engine (e.g., Safari, Firefox, Chrome and IE) to display the HTML pages.

---

## Grading

Graded is done according to:

- The quality of the data model used for representing data (XML/XSD)
- The quality of the overall solution
- The quality of the code (modularity, formatting, comments, code conventions, etc.)
- Complexity/Simplicity of the solution (*simple is beautiful*), including the screen scraping part
- Final presentation of the work

The project should be made in groups of **2** students. On your final report you should mention who was mostly involved in what part. Write it down explicitly. Also, we do expect all the members of the group to be fully aware of all the parts of the code that is submitted. Work together!