Enterprise Application Integration 2009/2010
Project #1 – XML and XML Manipulation

Objectives

- To gain familiarity with XML, and XML data processing
- To learn (or remember) how to use regular expressions
- Gain familiarity with the concept of screen scrapping

Due Date

- **2009/10/02**

Scheduled Effort

- **20 hours per student**, according to the following plan:
  - 4 hours → Reading and learning technologies (e.g., doing tutorials)
  - 12 hours → Coding and testing
  - 4 hours → Writing the report

PLEASE WRITE DOWN THE EFFECTIVE NUMBER OF HOURS SPENT IN EACH TASK. THE REPORTED EFFORT SHOULD BE PER STUDENT!

ONLY ONE MEMBER OF THE GROUP SHOULD SUBMIT THE ASSIGMENT

Final Delivery

- You must submit your project using Moodle (http://classes.dei.uc.pt). The enrollment key for the "Enterprise Application Integration" course is: "***eai2009***".
- The submission contents are:
  - Source code of the project ready to compile and execute. If you use Java, the compilation process must be automated by using an ***ant script*** (http://ant.apache.org/). If you don't know ant check this online tutorial: http://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html. Make sure you have the "compile" and "run" targets. If you decide not to use Java for the project, make sure to include a script that allows compiling and executing the project in an automated way.
  - A small report (5 pages max) about the implementation of the project. The report must specify the number of hours spent per student while working on the assignment. Please use PDF. We will not open any Microsoft Word documents.

Bibliography

Regular Expressions

- Scott A. Hommel, "Regular Expressions", in Java Tutorial http://java.sun.com/docs/books/tutorial/extra/regex/index.html
- David Mertz, "Learning to Use Regular Expressions Tutorial" http://gnosis.cx/publish/programming/regular_expressions.html
- "Class java.util.regex.Pattern", in J2SE 6.0 Documentation http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

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://java.sun.com/j2ee/1.4/docs/tutorial/doc/
- Keld H. Hansen, "Java and JDOM: the perfect couple", in JavaBoutique
  http://javaboutique.internet.com/tutorials/JDOM/

Processing XML/XSLT in Java

- Keld H. Hansen, "Working with JDOM, XPath and XSLT", in JavaBoutique
  http://javaboutique.internet.com/tutorials/jdom&/
- "Chapter 7: Extensible Stylesheet Language Transformations", in J2EE 1.4 Tutorial
  (Especially, the part "How XPath Works" and "Transforming XML Data with XSLT")
  http://java.sun.com/j2ee/1.4/docs/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

Ant

- Ant Manual
  http://ant.apache.org/manual/

Advice: Skim all the links above before starting to read anything in detail.

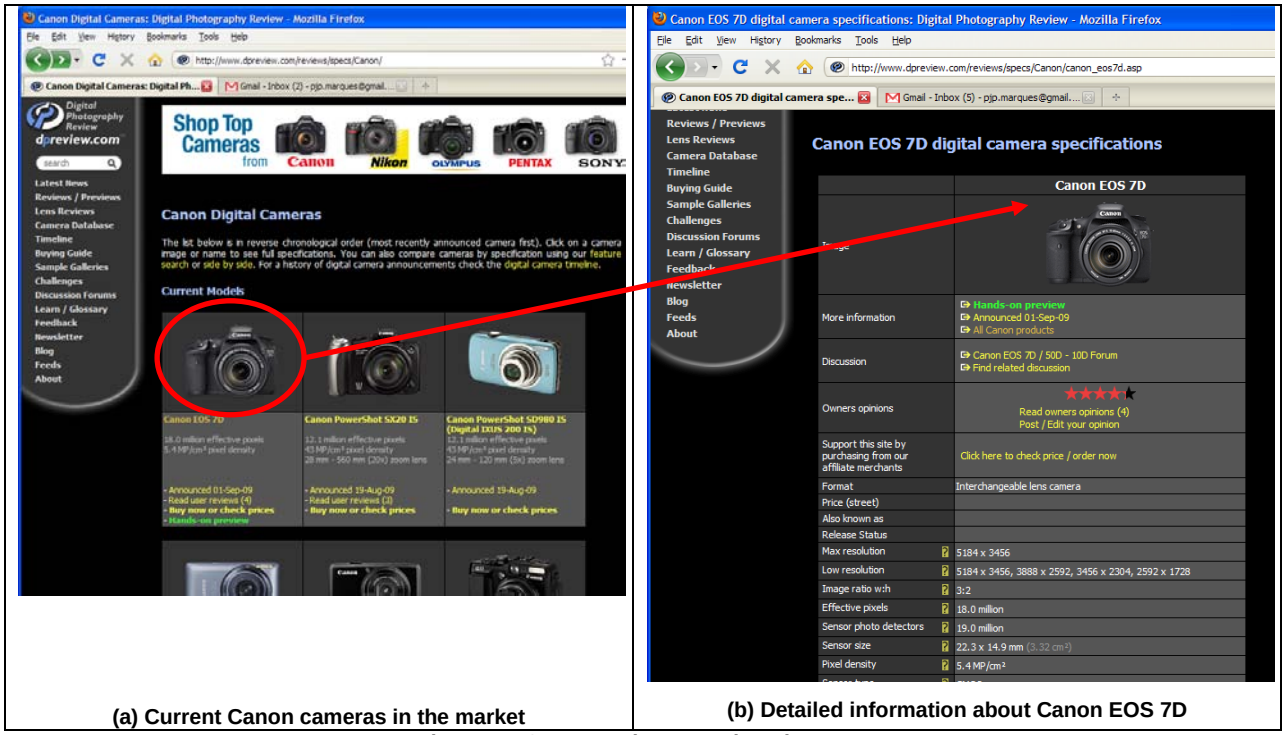
Introduction

XML plays a fundamental role in enterprise application integration. In fact, it is nowadays the *de facto* standard for data interchange between heterogeneous systems. It is also the core technology behind the so hyped web services and Service Oriented Architectures.

This first project aims at giving your exposure to this important technology. In particular, you will learn how to develop software to read, process and write XML documents. This will be done both in terms of ordinary programs and XST/XSLT templates.

The Assignment

Dpreview.com (http://www.dpreview.com/) is one of the most well-known and respected sites for camera reviews. Its core business is to provide in-depth reviews of the most important cameras in the market. As such, they have a very large database containing all the details regarding cameras from different manufacturers. Figure 1a) shows a Dpreview page where current Canon models are listed. If the user clicks on one of them (e.g., Canon EOS 7D), a new page is shown with the details about the camera (Figure 1b)).



(a) Current Canon cameras in the market

(b) Detailed information about Canon EOS 7D

Figure 1 – Cameras shown on dpreview.com

As it can be seen, information about many cameras is available. In this project you will develop three different applications which collect information about cameras preparing it to be processed by other systems which understand XML. These applications will allow you to:

- Understand the technique of "**Screen Scraping**". Screen scrapping consists in parsing the information shown in a terminal so that it can be used on a different system. It's 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 scrapping 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 regular expressions**. Regular expressions are an extremely powerful mechanism for cleaning, gathering and processing information which is embedded in text files. In this project you will use regular expressions extensively.
- Learn **XML technologies**. In particular, you will learn XML, XSD, XSL, XSLT and XPATH. This project is mostly about XML processing.

First Application – CameraSearchXML

CameraSearchXML is a command line application that receives a brand name (e.g., "Canon") as parameter and generates an XML file containing the details of all cameras of that brand currently available in the market. Note that:

- The list of brands available in Dpreview is listed at http://www.dpreview.com/reviews/specs.asp
- For obtaining the list of cameras of a certain BRAND you just need to access the corresponding URL: http://www.dpreview.com/reviews/specs/BRAND/. Note that "BRAND" is to be substituted by the actual brand (e.g., "Canon", "Nikon", etc.) For instance, Figure 1a) is the result of accessing the URL http://www.dpreview.com/reviews/specs/Canon/.

The tricky part is that for obtaining the details of each camera you need to visit the links associated to them. If you refer back to Figure 1a), if you access the link associated to "Canon EOS 7D" you get the details shown on Figure 1b).

Your application should capture all important information associated to each camera. You don't need to capture all information but, at least, you must capture:

- Camera Name
- Description
- Data it was announced to the market
- Max resolution
- Lower resolutions
- Image ratio
- Effective pixels
- Sensor size
- ISO rating
- Min shutter speed
- Max shutter speed
- Link to the "in depth review"
- Link to the camera picture

Note that some of this information may not be available. That will be something you need to decide how to handle.

The first step you have to make in order to get the information is to submit an HTTP GET request to Dpreview, parsing its result. The request will obtain the list of cameras of a certain brand.

After this step, your program will have to parse the resulting web page. One of the easiest ways to do that is by using regular expressions, isolating and breaking the pieces of the page corresponding to each book. If you carefully study the HTML corresponding to the result of a query, you will notice that there are repeatable patterns that can be isolated and automatically processed. For instance, "*<td valign=top class=tdcontentsm …>*", in the source of Figure 1a), is something that appears throughout the page and may be used as a marker to distinguish between cameras. We highly recommend that you use several regular expressions for doing successive partial parses of the text. Trying to write a single all encompassing expression is in many cases highly complex, error prone and difficult to maintain. Try not to do it that way.

The information that you need to save is present on the links you retrieve by parsing the first page. Again, you should use regular expressions to obtain the relevant information for the page source (Figure 1b)).

While gathering (or having gathered) all the information, your application must generate a corresponding XML file. The information that must be saved is, at least, the one mentioned above (camera name, description, etc.). The responsibility of the data model to use is yours. It should be carefully thought and appropriate for the task at hand. You must include an XML schema file (XSD) as part of your final submission and be ready to explain it.

Also note that your program must explicitly write the XML file to disk by using an XML library. It's not acceptable that the XML is generated programmatically as a "hand-crafted" text file.

Second Application – CamaraSummaryXML

The objective of this application is to process one or more XML files generated by the CameraSearchXML application. The name of the files is passed as parameter.

For each one of the camera manufactures present in the input files (e.g., Canon, Nikon, etc.), it should provide the following information:

- Number of cameras that are present in the input files
- Date of announcement of the most recent camera and corresponding model
- Date of announcement of the oldest camera and corresponding model
- Maximum resolution of a camera and corresponding model
- Minimum resolution of a camera and corresponding model
- A list containing all model names of that manufacturer

Overall, since information is per brand, you should have one "top-level tag" for each camera manufacturer.

The result should be written to disk in XML.

The responsibility of the data model to use for the resulting file is yours. It should be carefully thought and appropriate for the task at hand. You must include an XML schema file (XSD) as part of your final submission and be ready to explain it.

Note:

- Your program must explicitly read the XML files using an XML processing library. Also, the generated XML file must be produced using a library.
- You don't have to validate the XML file generated by the CameraSearchXML application against its XSD when reading it.

Third Application – CameraListBeautifier

The objective of this application is to create a nice HTML file starting from the CameraSearchXML application output. For that, you should use the XSLT technology, creating an XSL template.

Basically, you must create a template file with all the necessary rules for transforming the original XML file into HTML. These rules can be applied using: a) a stand-alone XSLT engine (e.g., Apache XALAN); a program written by you using, for instance, Java's internal XSLT libraries; a simple web browser with a built-in XSLT engine (e.g., Firefox and IE). The choice is yours!

Technology

For this project you can use any technology/programming language able to process XML. We suggest that the implementation is done in Java, using the JDOM XML processing library (http://www.jdom.org/). We are suggesting JDOM because of its easy of use compared with the "normal" SAX and DOM libraries. Do note that in Java there are several libraries that can process XML and some are even distributed off-the-shelf in Sun's JDK. The choice is yours!

For the XSLT processing, given the simplicity of the assignment, we suggest that you use a web browser like Firefox or IE.

For completing this project you can use any platform, even if it's not Java (e.g., .NET, Python, Ruby). Nevertheless, do note that future projects may imply reusing part of this project and that some upcoming projects will be in Java. This means that if you develop in a platform other than Java, in one of the next assignments, you may have to wrap your code in Java. That's perfectly acceptable.

Grading

Graded is done according to:

- The quality of the data model used for representing the information (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*)
- Final presentation of the work

The project should be made in groups of 3 students. One 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!

Appendix – Technologies

JDOM

Having Java installed, for compiling a program that uses JDOM (http://www.jdom.org/) you only need to have the corresponding library on the CLASSPATH. Note that the library is the "**build/jdom.jar**" file inside of "jdom-1.0.jar".

As an example, for compiling and executing this simple application:

```
/*
 * Small application that shows how to generate XML using JDOM.
 *
 * (c) Paulo Marques (pmarques@dei.uc.pt), DEI/FCTUC
 * 25-Aug-2005
 */

import java.io.*;
import javax.xml.transform.*;
import org.jdom.*;
import org.jdom.output.*;

public class JDOM_Demo
{
  public static void main(String[] args)
    throws IOException
  {
    // Create a simple XML document
    Element  pubElement = new Element("publications");
    Document myDocument = new Document(pubElement);

    Element autor = new Element("author");
    autor.addContent("Henrique Madeira");
    pubElement.addContent(autor);

    // Write it to disk
    XMLOutputter outputter = new XMLOutputter(Format.getPrettyFormat());
    FileWriter writer = new FileWriter("publications.xml");
    outputter.output(myDocument, writer);
    writer.close();
  }
}
```

Write:

```
$ javac -cp .:jdom.jar JDOM_Demo.java
$ java -cp .:jdom.jar JDOM_Demo
$ cat publications.xml
<?xml version="1.0" encoding="UTF-8"?>
<publications>
  <author>Henrique Madeira</author>
</publications>
```

XSLT

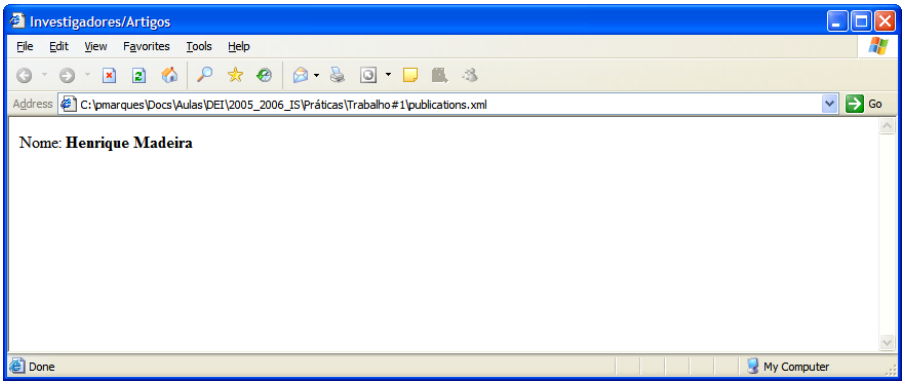
Assuming that you will be using a web browser for showing/displaying the generated HTML you only really need to create the corresponding XSL template file. For instance, given the previous generated XML ("*publications.xml*") you can modify it so that it references a template file named "*ScholarPrettyPrint.xsl*":

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ScholarPrettyPrint.xsl"?>

<publications>
  <author>Henrique Madeira</author>
</publications>
```

which content is:

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

<html xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xsl:version="1.0">

  <head><title> Investigadores/Artigos </title></head>
  <body>

  <xsl:for-each select="//author">
       Nome: <b> <xsl:value-of select="."/> </b> <br/>
  </xsl:for-each>

  </body>

</html>
```

When you ask Internet Explorer to render it, the result is:



Retrieving a web page from a site

The following code shows how to retrieve a web page from a site:

```
/*
 * Small application showing how to retrieve a web page from a site
 *
 * (c) Paulo Marques (pmarques@dei.uc.pt), DEI/FCTUC
 * v1.0 Sep/2008
 */

import java.net.*;
import java.io.*;

public class Flickr
{
  public static void main(String[] args)
    throws IOException
  {
    // Query String
    String query = "Nature";

    // Query Flcikr, allocating a BufferedReader for the resultign page
    URL flickUrl = new URL("http://www.flickr.com/search/?q=" +
               URLEncoder.encode(query, "UTF-8"));
    URLConnection connection = flickUrl.openConnection();
    connection.setRequestProperty("User-Agent", "");
    BufferedReader br =
      new BufferedReader(new InputStreamReader(connection.getInputStream()));


    // Printout the resulting page
    String line = null;
    do
    {
      line = br.readLine();
      if (line != null)
        System.out.println(line);
      } while (line != null);

    br.close();
  }
}
```