Enterprise Application Integration

XML & XML Manipulation

Project #1 Report

Carlos Simões
Miguel Oliveira
Pedro Saraiva

2009/2010

Introduction

During the first project of the course Enterprise Application Integration, named “XML and XML Manipulation”, decisions, from the point of view of design and implementation, had to be made. The goal of this report is to describe these decisions and the reasons why they were made.

In addition, the report presents the instructions for installing and executing both applications and applying the CameraListBeautifier XSL transformation.

Time Spent

Student	Mostly involved in	Time spent
Carlos Simões	• First application • Definition of the XSD for the first application	26:21
Miguel Graça Oliveira	• Third application • XPath for the second application • Report	19:02
Pedro Saraiva	• Second application • Definition of the XSD for the second application	22:25

Installation Instructions

CameraSearchXML

Prerequisites:

- Ant should be installed in the system.
- Java should be installed in the system.

Instructions:

1. Unzip the source files to a directory.
2. Open a shell in that directory and run the command “ant”.
3. Go to the directory “dist” that has just been created.
4. Run the command “java –jar CameraSearchXML.jar <brand>”

CameraSummaryXML

Prerequisites:

- Ant should be installed in the system.
- Java should be installed in the system.

Instructions:

1. Unzip the source files to a directory.
2. Open a shell in that directory and run the command “ant”.
3. Go to the directory “dist” that has just been created.
4. Run the command “java –jar CameraSummaryXML.jar <brand1>.xml <brand2>.xml … <brandN>.xml”

CameraListBeautifier

In order to generate the final html document a XSLT file is provided called CameraListBeautifier.xslt. There are two recommended methods in order to see the result of applying the XSL to the input XML file.

The first one would be to alter the output XML file from CameraSearchXML with the following line:

```
<?xml-stylesheet
      type="text/xsl" href="CameraListBeautifier.xslt"?>
```

This solution assumes that both files share the same filesystem path.

The second one, and perhaps the more elegant one, would be to use the application msxsl, included in the package, to apply the XSL transformation to the XML file. An example of a possible usage would be:

```
msxsl canon.xml CameraListBeautifier.xslt –o output.html
```

Which would generate a file called output.html with the output of the XSL transformation when applied to the file canon.xml.

Implementation and Design decisions

CameraSearchXML

Regarding the resolution, both horizontal and vertical, are separated in different elements. This was decided in order to allow for faster and easier searches for a specific element of the resolution. A possible example would be finding the camera with the largest horizontal resolution.

Also regarding the resolution, an additional element, with information not present on the page and calculated, was added. The amount of pixels a given resolution produces, this means multiplying the horizontal resolution component with the vertical resolution component. This was added in order to allow for easier comparison between resolutions, which means being able to say that resolution from camera X is larger than resolution from camera Y, and for quick referencing when using XPath.

In the case of lists of elements, a parent element is used to group the rest of the elements. A good example may be the child element of the root node: Cameras. A node of this type contains a list of cameras, which means one or more Camera nodes. In the case of an empty list, no child nodes would exist and the parent node (in this example, Cameras) would be empty:

```
<Cameras />
```

Certain elements, such as EffectivePixels and ImageRatio, are stored as strings instead of numbers. This was decided because these elements are used only for presentation and no searches are foreseen on these elements.

The name of the Brand is stored within an element and not as an attribute on the root element. This was done in order to prevent possible errors related to special characters on the name of the Brand. We used this principle in the definition of the CameraSearchXML XML format.

CameraSummaryXML

All files to be summarized are totally loaded to memory when the application starts. This decision was based on the assumption that no brand file would be extremely large and, hence, will never spend an amount of memory that would hinder the normal execution of a regular laptop. An internal data structure based on HashMap is used in order to structure the loaded files and its information.

XPath is used extensively to search for the required nodes in order to summarize the data on each of the brands. This is due to being much faster than browsing the DOM model of the XML searching for the relevant data. The data found is then stored in another structure based on HashMap with a list of all the cameras with the required characteristics.

When dealing with XPath, and calculating maximum values, though, we ran into some issues, which limited our ability to select the desired nodes without necessity of further processing. First, the Java implementation of XPath does not support XPath 2.0, which prevented the team from using several functions existent on the latest specification. In addition, the Java implementation of XPath 1.0 is different from the standard when comparing strings that can be evaluated to numbers. This can be verified because on both XML Spy and XML Copy Editor, XPath 1.0 expressions return the same output in every occasion. On the other hand, Java differs from the standard, at least, in the described situation.

Like in the previous application, the same rules for lists of elements were applied.

CameraListBeautifier

Three templates were used.

The first one, which searches for the root node, has all the HTML and CSS code for the presentation. This template applies the other two templates.

The second template, searches for Name, which contains the brand name. This template is applied in order to specify the title of the page.

The third template creates a table for each of the Cameras and presents in pretty print the list of cameras.