Università degli Studi di Trento

Web Architectures

XSLT demo

Gianvito Taneburgo

November 14, 2016

1 Introduction

The following report describes the procedure that has been followed to develop an application which is able to transform an XML file into a CSV file via XSLT. In particular, the program relies on a XSL stylesheet file to extract some fields of interest from an XML file describing the taxonomy of the courses organization in the University of Trento.
The Implementation paragraph describes the rules contained in the XSL stylesheet and how the transformation is executed in Java by using *Apache Xalan* [1] . Some details about XSLT have been omitted: the reader is supposed to know the benefits of the technology and the basic mechanisms that make it possible.
The Deployment paragraph, on the contrary, provides information on how to compile and run the application.
For the remainder of this report the following versions of specifications and libraries will be considered:

- Java Platform, Standard Edition: *8*
- XSL stylesheet: *1*
- Apache Xalan: *2.7.2*

[1] http://xalan.apache.org/

2 Implementation

The XSL stylesheet file is now described piece-wise.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

These tags define the XML and XSL standard versions used to encode the file itself.

```
<xsl:output method="text" encoding="ISO-8859-1"/>
```

The tag defines the format (plain text) and encoding of the output file.

```
<xsl:strip-space elements="*"/>
```

The tag is used to remove all the blank lines from the output file.

```
<xsl:template match="TAXONOMY/AREA/UNITS/UNIT">
    <xsl:text>"</xsl:text>
    <xsl:value-of select="UNIT_NAME"/>
    <xsl:text>",</xsl:text>
    <xsl:value-of select="@TYPE" />
    <xsl:text>,</xsl:text>
    <xsl:value-of select="TIME"/>
    <xsl:text>&#xa;</xsl:text>
</xsl:template>
```

This template extracts the fields of interest from each *UNIT*. Fields will be separated by a comma (,). The *UNIT-NAME* may contain commas as well, so it is wrapped between double quotes (″) to avoid ambiguity. After each element, a *line feed* character (*&#xa;*) is added to insert a new line.

```
<xsl:template match="TAXONOMY/AREA/AREA_NAME"/>
<xsl:template match="TAXONOMY/AREA/SHORT_NAME"/>
<xsl:template match="TAXONOMY/AREA/DESCRIPTION"/>
```

These template tags filter out useless *AREA* fields contained in the XML.

```
</xsl:stylesheet>
```

The closing stylesheet file tag.

The application code is short and straightforward.
The XSL stylesheet file is first loaded. A DOM representation of the input XML file is built in the traditional way by using a *DocumentBuilderFactory* [2] (*dbf*). DTD-validation is disabled with the following line:

[2]https://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/DocumentBuilderFactory.html

```
dbf.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd", false);
```

An *Apache Xalan Transformer* [3] object is now able to run the XSL transformation on the newly built document, storing the result in an output file.

3 Deployment

Download the *src.zip* file into a folder. All the following commands have to be executed from the download directory.
To unzip the archive type:

```
$ unzip src.zip
```

To compile and package the application type:

```
$ mvn -f src clean install package
```

To run the application and convert the *ACMTrento.xml* file provided type:

```
$ java -jar src/target/xsl-transformer-1.0.jar ACMTrento.xsl
    ACMTrento.xml out.csv ISO-8859-1
```

A new CSV file *out.csv* will be created. It will contain the following lines:

```
"Functions, relations, and sets",CORE,6
"Basic logic",CORE,10
"Proof techniques",CORE,12
"Basics of counting",CORE,5
"Graphs and trees",CORE,4
"Discrete probability",CORE,6
[...]
```

[3] https://docs.oracle.com/javase/8/docs/api/javax/xml/transform/Transformer.html