Enterprise Systems & Architecture

Lab 8 (Week 9): XML / XSD / XSL & Java

Exercise: Java – Loading, Validating and Transforming an XML file

Note: You can do this over the next two lab sessions.

Setup

- Start Eclipse (JEE Installation)
- Create a new eclipse project: *File -> New -> Java Project*
- Give the project a name XSLT_Transformation and click *Finish*
- Copy the following files from this week's lab sheet folder on Brightspace to your project:
  - *shipment.xml*
  - *shipment.xsd*
  - *shipment.xsl*
- Right-Click on the project name:
  - *New -> Class*
  - leave package blank
  - name of class: XsltTransformation
- Ensure the class has an empty *main* method

Load the XML file

- Copy the following code into the main method:

```
//Load the xml file...
File file = new File("shipment.xml");
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(file);

System.out.println(doc.getDocumentElement().getTextContent());
```

- Fix any import and exception handling errors (hint `org.w3c.dom.Document`, most other classes are in the javax.xml package)
- Right-Click on java class -> *Run As -> Java Application*

Create a Schema and Validator object and Execute the Validation

- Copy the following code to your main method just after the code above

```
//Load the xml schema and create a validator for it...
SchemaFactory factory =
        SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema schema = factory.newSchema(new File("shipment.xsd"));
Validator validator = schema.newValidator();

//Validate the xml file against the schema...
validator.validate(new DOMSource(doc));
System.out.println("Success...");
```

- Fix any import and exception handling errors
- Right-Click on java class -> *Run As -> Java Application*

Transform the XML to HTML

- Copy the following code to your class

```
//Transform the xml to html...
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer( new StreamSource("shipment.xsl"));
transformer.transform( new StreamSource("shipment.xml"),
                       new StreamResult(new FileOutputStream("shipment.html")) );

System.out.println("** The output is written to shipment.html **");
```

- Run your class and refresh your project to see the html file generated
- Open the html file with the eclipse browser or another browser.

Test out the Validator

- Modify the XML so that it does not conform with its schema (e.g. add an extra well-formed xml element).
- Run your class to see the validator exceptions

Exercise

To do something a bit more useful, write a class called *XmlHandler*. It should have the following two methods implemented:

1. *public boolean validate(String xml, String xsd)*
2. *public String transform(String xml, String xsl)*

The first method should validate the XML String against the schema also provided as a String and return a success / failure boolean.

The second method should perform an XSL transformation on the XML provided as a String using the XSL also provided as a String.

Write some test code to test out your implementations. Note you can load a file in to a String using

Files.readAllBytes(Paths.get("shipment.xml"))

Notes:

You will need to utilise the following jdk classes in order to convert the xml/xsd/xsl between the various formats (e.g. Files / Strings / Sources / Results). Everything you need is in JDK javadoc.

```
java.io.StringReader;
java.io.StringWriter;

import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import org.xml.sax.InputSource;
```