Processament XML amb Java

Projecte Tecnologies d'Informació

Pau Coma Ninot

Àlex Aguilera Martínez

Grup 20

1. Objectius pràctica

L'objectiu d'aquesta pràctica és la creació d'una aplicació de consola amb el llenguatge de programació Java que mantengui la informació del lloguer de cotxes en un fitxer XML, anomenat carrental.xml. Utilitzarem JDOM, una biblioteca per permetre el desenvolupament ràpid d'aplicacions XML i el fitxer exemple.java proporcionat pel professorat.

Aquesta aplicació, tindrà una serie d'accions que s'especificaran com a paràmetre a l'executar l'aplicació per consola. Les funcions seran les següents:

- Reset → Resetja el fitxer carrental.xml, sense cap rental.
- New → Afegeix un nou rental introduent per consola les dades
- List → Imprimeix el fitxer carrental.xml per consola
- Xslt → Crea un document nou i el transforma a HTML amb el full d'estils XSLT
- Validate → Valida el fitxer carrental.xml amb fitxer carrental.xsd

2. Implementació

1- Instal·lació i configuració

En aquesta pràctica l'instal·lació/configuració és bastant fàcil i ràpida. Descarreguem els fitxers necessaris per la pràctica del gitlab de la fib i creem el nostre directori de treball:

$git clone https://gitlab.fib.upc.edu/pti/pti.git

Seguidament, instal·lem Java, en cas de no tenir-lo, i establim el camí de ruta Java (important per el correcte funcionament de la aplicació CarRental.java) de la següent manera:

$apt-get install default-jdk (instal·lació de Java)

$export CLASSPATH=./xalan.jar:./xercesImpl.jar:./jdom.jar:. (camí de ruta)

Una vegada fet això, creem un nou fitxer anomenat "CarRental.java" i copiem el codi del fitxer "exemple.java" dintre. Ara, ja podem començar a modificar el codi segons les accions que hem de implementar pel lloguer de cotxes.

2- Reset

Per dur a terme aquesta primera acció, hem aprofitat la funció "outputDocumentToFile" modificant el fitxer output per "carrental.xml":

```
public static void outputDocumentToFile(Document myDocument) {
    //setup this like outputDocument
    try {
        // XMLOutputter outputter = new XMLOutputter("  ", true);
        XMLOutputter outputter = new XMLOutputter();

        //output to a file
        FileWriter writer = new FileWriter("carrental.xml");
        outputter.output(myDocument, writer);
        writer.close();

    } catch(java.io.IOException e) {
        e.printStackTrace();
    }
}
```

Al main es crida a la funció:

```
if(command.equals("reset")) outputDocumentToFile(createDocument());
```

On createDocument() és:

```
public static Document createDocument() { //con la estructura indicada; "vacío"
    // Create the root element
    Element carElement = new Element("carrental");
    //create the document
    Document myDocument = new Document(carElement);

    return myDocument;
}
```

(simplement crea l'element carrental al fitxer .xml, però buit)

3- <u>New</u>

Per aquesta segona acció, creem una nova funció anomenada newRental() que es crida al main:

```
else if(command.equals("new")) newRental();
```

L'implementació d'aquesta funció és la següent:

```
public static void newRental() {
    /* Cogemos valores por consola */
    Element rental = askData();

    /* Leemos el documento y escribimos nuevo rental*/
    Document document = readDocument();
    Element root = document.getRootElement();
    root.addContent(rental);
    document.setContent(root);

    /* Escribimos en el fichero "carrental.xml"*/
    writeDoc(document);
}
```

On askData(), és la funció utilitzada per agafar els valors per consola i introduir-los a l'element "rental".

```
public static Element askData() {
    /* Cogemos valores por consola */
    System.out.print("Enter car maker: ");
    String carMaker = System.console().readLine();
    System.out.print("Enter car model: ");
    String carModel =  System.console().readLine();
    System.out.print("Enter days: ");
    String days =  System.console().readLine();
    System.out.print("Enter units:");
    String units =  System.console().readLine();
    System.out.print("Enter discount:");
    String discount =  System.console().readLine();

    /* Creamos el nuevo elemento y le ponemos los parametros introducidos por consola */
    Element rental = new Element("rental");
    rental.setAttribute("id", String.valueOf((int)Math.floor(Math.random()*(20-1+1)+1))); //generamos id random
    rental.addContent(new Element("make").setText(carMaker));
    rental.addContent(new Element("model").setText(carModel));
    rental.addContent(new Element("nofdays").setText(days));
    rental.addContent(new Element("nofunits").setText(units));
    rental.addContent(new Element("discount").setText(discount));

    return rental;
}
```

Després, amb la funció writeDoc() escrivim el nou rental al fitxer carrental.xml. A la següent imatge, s'han introduit dos rentals en mode d'exemple:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<carrental>
  <rental id="8">
    <make>seat</make>
    <model>ibiza</model>
    <nofdays>1</nofdays>
    <nofunits>2</nofunits>
    <discount>15</discount>
  </rental>
  <rental id="3">
    <make>seat</make>
    <model>leon</model>
    <nofdays>2</nofdays>
    <nofunits>3</nofunits>
    <discount>10</discount>
  </rental>
</carrental>
```

4- List

En aquest acció, simplement era llegir el fitxer carrental.xml i imprimir-lo. Per fer-ho, hem utilitzat la funció outputDocument(), ja implementada, fent ús de la funció readDocument() per llegir el fitxer carrental.xml.

```
else if(command.equals("list")) outputDocument(readDocument());
```

```
public static void outputDocument(Document myDocument) {
    try {
        // XMLOutputter outputter = new XMLOutputter("   ", true);
        XMLOutputter outputter = new XMLOutputter(Format.getPrettyFormat());
        outputter.output(myDocument, System.out);
    } catch (java.io.IOException e) {
        e.printStackTrace();
    }
}
```

5- Xslt

Per fer la funció xslt, afegirem l' arxiu carrental.xslt que farem servir per generar l' arxiu html. L' arxiu queda així:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<carrental xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="carrental.xsd">
</carrental>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:template match="/carrental">
      <html>
          <head><title>RENTALS</title></head>
          <body>
              <xsl:apply-templates select="rental"/>
          </body>
      </html>
  </xsl:template>
  <xsl:template match="rental">
      <table border="0">
          <h1>MAKE=<xsl:value-of select="make"/></h1><br/>
          <h1>MODEL=<xsl:value-of select="model"/></h1><br/>
          <h1>NOFDAY=<xsl:value-of select="nofday"/></h1><br/>
          <h1>NOFUNITS=<xsl:value-of select="nofunits"/></h1><br/>
          <h1>DISCOUNT=<xsl:value-of select="discount"/></h1><br/>
      </table>
  </xsl:template>
</xsl:stylesheet>
```

Al main afegim la següent línia:

```java
else if(command.equals("xslt")) executeXSLT(readDocument());
```

La funció readDocument ja la tenim creada, i la funció executeXSLT ens quedaria així:

```java
public static void executeXSLT(Document myDocument) {

    try {
        TransformerFactory tFactory = TransformerFactory.newInstance();
        // Make the input sources for the XML and XSLT documents
        org.jdom2.output.DOMOutputter outputter = new org.jdom2.output.DOMOutputter();
        org.w3c.dom.Document domDocument = outputter.output(myDocument);
        javax.xml.transform.Source xmlSource = new javax.xml.transform.dom.DOMSource(domDocument);
        StreamSource xsltSource = new StreamSource(new FileInputStream("carrental.xslt"));
        //Make the output result for the finished document
        StreamResult xmlResult = new StreamResult(System.out);
        //Get a XSLT transformer
        Transformer transformer = tFactory.newTransformer(xsltSource);
        //do the transform
        transformer.transform(xmlSource, xmlResult);
    } catch(FileNotFoundException e) {
        e.printStackTrace();
    } catch(TransformerConfigurationException e) {
        e.printStackTrace();
    } catch(TransformerException e) {
        e.printStackTrace();
    } catch(org.jdom2.JDOMException e) {
        e.printStackTrace();
    }
}
```

6- Validate

Per la funció validate, afegirem dues coses: la línia de validar al main, i la funció “validar”. El codi quedaria així:

```
else if(command.equals("validate")) validar ();
```

```
public static void validar () {

    try {

        SAXBuilder builder = new SAXBuilder (XMLReaders.XSDVALIDATING);

        Document altre = builder.build (new File ("carrental.xml"));

        System.out.println ("Root: " + altre.getRootElement ().getName ());


    }
    catch (FileNotFoundException e) {

        e.printStackTrace ();

    }

}
```