Pràctica 4 - JDOM

Pere Joan Martorell Calaf
Bernat Farrero Badal

En aquesta sessió hem après a gestionar la persistència de les dades per mitjà de fitxers XML. Per fer-ho, hem usat la llibreria JDOM, seguint l'article guia sobre ella de Wes Biggs i Harry Evans publicat el 2001[1].

Talment com en la sessió anterior, en aquesta sessió l'interfície de comunicació amb l'usuari serà altre cop la terminal. Per la terminal es llegiran les dades introduides per l'usuari i s'oferiran les sortides corresponents.

Instal·lació d'eines

Cal descarregar Jdom i Xalan. Jdom, llibreria creada per usar-se exclusivament amb Java, s'integra amb DOM i SAX (Simple API for XML) per tal de convertir un arbre d'objectes de Java en un document XML o al revés. Xalan és una altra llibreria que permet processar XSLT per transformar XMLs.

Després de descarregar i descomprimir les dues llibreries a la carpeta de treball, actualitzem el CLASSPATH de Java per tal que les referencii.

```
$ export CLASSPATH=~/JDOM/xalan-j_2_7_0/xalan.jar:~/JDOM/xalan-
j_2_7_0/xercesImpl.jar:~/JDOM/jdom-1.0/build/jdom.jar:.
```

Aplicació de gestió de Lloguers

Com en les anteriors pràctiques, en aquesta se'ns demana que implementem una aplicació que permeti crear lloguers, que aquests s'emmagatzemin en un fitxer XML situat a xml/llistalloguers.xml i que permeti també llistar-los per pantalla o bé, adicionalment, transformar la llista a un HTML. Les opcions són les següents:

`$ java Lloguers reset` - Esborra els lloguers creats
`$ java Lloguers lloguer` – Crea un nou lloguer introduint la informació
`$ java Lloguers llistar` - Retorna llistat de lloguers existents en XML
`$ java Lloguers  xsl` - Retorna el llistat de lloguers existents en HTML

Per implementar l'aplicació hem pres com a patró l'exemple de l'article mencionat inicialment, Articles.java. Tot seguit detallarem les parts més significatives de la nostra versió, Lloguers.java.

Estructura del fitxer XML que conté els lloguers i que es troba a xml/llistalloguers.xml:

---

[1] http://www.ibm.com/developerworks/java/library/j-jdom/

```
<llistaLloguers>
  <!--Caracteristiques del lloguer-->
  <lloguer>
    <model>Luxe</model>
    <submodel>Gasolina</submodel>
    <dies_lloguer>14</dies_lloguer>
    <num_vehicles>1</num_vehicles>
    <desc>39</desc>
    <total>811.3</total>
  </lloguer>
</llistaLloguers>
```

Per fer el reset cal cridar la funció removeChildrenElement, que conté el següent codi:

```
public static void removeChildrenElement(Document myDocument)
{
    //obtenim l'arrel
    Element llistaLloguersElement = myDocument.getRootElement();
    //eliminem tots els seus lloguers-fills
    boolean eliminat = llistaLloguersElement.removeChildren("lloguer");
    //comprovem si s'han eliminat amb èxit
    if(eliminat)
    {
        System.out.println("S'han eliminat els lloguers satisfactoriament.");
        outputDocumentToFile(myDocument);
    } else {
        System.out.println("Ha estat impossible eliminar els lloguers.");
    }
    outputDocumentToFile(myDocument);
}
```

Com es pot veure, una cop detectat el root, n'hi ha prou amb cridar la funció removeChildrenElement(root) per eliminar cada lloguer que en penja i deixar l'arrel buida. El fitxer xml queda de la següent manera:

```
<llistaLloguers>
  <!--Caracteristiques del lloguer-->
</llistaLloguers>
```

Per crear un lloguer, i després d'haver llegit per pantalla les dades entrades per l'usuari de forma anàloga a la pràctica 3, cal fer el següent:

```
Element rootElement;
Document myDocument = readDocument();

//Si no existeix el fitxer xml creem una nova arrel
if (myDocument==null)
{
    // Create the root element
    System.out.println("S'ha creat un fitxer XML nou!\n");
    rootElement = new Element("llistaLloguers");
    rootElement.addContent(new Comment("Caracteristiques del lloguer"));
    //create the document
    myDocument = new Document(rootElement);
}

//si ja existeix el fitxer xml, llegim l'element arrel
else rootElement = myDocument.getRootElement();

//creem un Element de lloguer
Element lloguer = new Element("lloguer");
//afegim el contingut dins del lloguer (afegim subchilds)
lloguer.addContent(new Element("model").addContent(model));
lloguer.addContent(new Element("submodel").addContent(submodel));
lloguer.addContent(new Element("dies_lloguer").addContent(Integer.toString(dies_lloguer)));
lloguer.addContent(new Element("num_vehicles").addContent(Integer.toString(num_vehicles)));
lloguer.addContent(new Element("desc").addContent(Integer.toString(desc)));
lloguer.addContent(new Element("total").addContent(Double.toString(total)));

//afegim el lloguer com a child de l'arrel
rootElement.addContent(lloguer);

return myDocument;
```

Aquest resultat el guardem a un document mitjançant la funció outputDocumentToFile per tal d'assegurar la persistència i emmagatzemar els lloguer al fitxer xml/llistalloguers.xml. Aquesta funció guarda el fitxer de la següent manera:

```
XMLOutputter outputter = new XMLOutputter(Format.getPrettyFormat());

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

La funció llistar té per objectiu simplement mostrar l'xml amb el registre de lloguers per pantalla. Ho fa cridant la funció outputDocument, que crea un objecte Outputter que s'encarrega de generar el document. Ho fa de la següent manera:

```
XMLOutputter outputter = new XMLOutputter();
outputter.output(myDocument, System.out);
```

Finalment, la funció executeXSL és la que s'encarrega d'aplicar les transformacions corresponents al fitxer XML per convertir-lo a un format llegible, HTML en aquest cas. Per fer-ho hem creat la següent plantilla XSL que es troba a xml/lloguer.xsl:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/llistaLloguers">
        <html>
            <head>
                <title>Llista de Lloguers</title>
            </head>
            <body>
                <br /><h1><i>Llista de Lloguers</i></h1><br />
                <table border='1' style='border-collapse:collapse; text-align:center;'>
                    <tr bgcolor="#9acd32"><td><b>Model de Vehicle</b></td><td><b>Tipus de motor</b></td>
                        <td><b>Dies del lloguer</b></td><td><b>Quantitat</b></td>
                        <td><b>Descompte</b></td><td><b>Import total</b></td>
                    </tr>
                    <xsl:for-each select="lloguer">
                      <tr><td><xsl:value-of select="model"/></td>
                        <td><xsl:value-of select="submodel"/></td>
                        <td><xsl:value-of select="dies_lloguer"/></td>
                        <td><xsl:value-of select="num_vehicles"/></td>
                        <td><xsl:value-of select="desc"/></td>
                        <td><b><xsl:value-of select="total"/></b></td>
                       </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Com es pot comprovar, l'atribut match de la plantilla indica el fitxer a transformar, en aquest cas xml/llistalloguers.xml. El tag `<xsl:for-each select="lloguer">` no és més que una instrucció iterativa que fa recórrer tots els lloguers. El tag `<xsl:value-of select="CAMP">` es tradueix en el valor de l'atribut CAMP de l'objecte lloguer.

La funció transformadora executeXSL, tal com ens mostrava l'exemple de l'Article, fa el següent:

```
TransformerFactory tFactory = TransformerFactory.newInstance();
// Make the input sources for the XML and XSLT documents
org.jdom.output.DOMOutputter outputter = new org.jdom.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("xml/lloguer.xsl"));
StreamResult xmlResult = new StreamResult(System.out);
//Get a XSLT transformer
Transformer transformer = tFactory.newTransformer(xsltSource);
//do the transform
transformer.transform(xmlSource, xmlResult);
```

Com en l'anterior pràctica, la funció main de la classe ofereix el menú amb les opcions, que s'hauran d'escriure per teclat. En cas que no s'introdueixin, es rebrà una ajuda amb les opcions que ofereix la nostra aplicació. Val a dir que, a diferència de la pràctica anterior, aquesta vegada hem usat les llibreries io del sistema per llegir de teclat:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
```