PROGRAMACIÓN

Unidad 8. Almacenando datos

TRATAMIENTO DE DOCUMENTOS ESTRUCTURADOS XML

José L. Berenguel

Tabla de Contenidos

Introducción

- **XML** (***eXtensible Markup Language***). Mecanismo para estructurar la información de manera sencilla y que permite su intercambio entre diferentes sistemas.
- Consta de un conjunto de estándares elaborados por el W3C.

Elementos de un documento XML.

Elemento	Descripción	Ejemplo
Cabecera o declaración del XML.	Es lo primero que encontramos en el documento XML y define cosas como, por ejemplo, la codificación del documento XML (que suele ser ISO-8859-1 o UTF-8) y la versión del estándar XML que sigue nuestro documento XML.	`<?xml version="1.0" encoding="ISO-8859-1"?>`

Etiquetas.	Una etiqueta es un delimitador de datos, y a su vez, un elemento organizativo. La información va entre las etiquetas de apertura ("`<pedido>`") y cierre ("`</pedido>`"). **Fíjate en el nombre de la etiqueta ("`pedido`"), debe ser el mismo tanto en el cierre como en la apertura, respetando mayúsculas**.	`<pedido>`
		`  información del pedido`
		`</pedido>`

Atributos.	Una etiqueta puede tener asociado uno o más atributos. Siempre deben ir detrás del nombre de la etiqueta, en la etiqueta de apertura, poniendo el nombre del atributo seguido de igual y **el valor encerrado entre comillas**. Siempre debes dejar al menos un espacio entre los atributos.	`<articulo cantidad="20">`
		`  información`
		`</articulo>`

Texto.	Entre el cierre y la apertura de una etiqueta puede haber texto.	`<cliente>`
		`  Muebles Bonitos S.A.`
		`</cliente>`

Etiquetas sin contenido.	Cuando una etiqueta no tiene contenido, no tiene porqué haber una etiqueta de cierre, pero no debes olvidar poner la barra de cierre ("/") al final de la etiqueta para indicar que no tiene contenido.	`<fecha entrega="1/1/2012" />`
Comentario.	Es posible introducir comentarios en XML y estos van dirigidos generalmente a un ser humano que lee directamente el documento XML.	`<!-- comentario -->`

Estructura de un documento XML

- Los elementos XML se estructuran en forma de árbol, con un **nodo raíz**, del que cuelgan todos los demás elementos hasta los **nodos hoja**.
  - El par formado por la **etiqueta de apertura** y **cierre** es un nodo llamado **elemento** (***element***). En su interior puede contener otros elementos.
  - Un **atributo** (***attrib***) es un nodo especial que suele puede estar dentro de una etiqueta de apertura.
  - Un **comentario** (***comment***) es un nodo especial que puede estar en cualquier lugar del documento XML.
  - Un **documento** (***document***) es un nodo que contiene una jerarquía de nodos en su interior.

Estructura de un documento XML

```
<padre att1="valor" att2="valor">
   texto 1
   <ethija> texto 2 </ethija>
</padre>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pedido>
    <cliente> texto </cliente>
    <codCliente> texto </codCliente>
    ...

</pedido>
```

Librerías para procesar documentos XML a DOM

- Vamos a estudiar cómo trabajar solamente con dos tecnologías de XML, de entre las muchas que hay:
  - **Procesadores XML**. Librerías para leer documentos XML y comprobar que están bien formados. En Java disponemos del **API SAX** (***Simple Api for XML***).
  - **XML DOM**. Permite transformar un documento XML en un modelo de objetos accesible desde la aplicación. En Java disponemos del **API JAXP** (***Java API for XML Processing***).

Librerías para procesar documentos XML a DOM

- Para cargar un documento XML necesitamos las siguientes clases:
  - ***javax.xml.parsers.DocumentBuilder***. Será el procesador y transformará el documento XML a árbol DOM. Se le conoce como constructor de documentos.
  - ***javax.xml.parsers.DocumentBuilderFactory***. Permite crear un constructor de documentos.
  - ***org.w3c.dom.Document***. Clase que modela el objeto del árbol DOM. Cuando el parser procesa el documento XML, crea una instancia de ***Document*** con el contenido del mismo.

Librerías para procesar documentos XML a DOM

- 1º Creamos una nueva instancia de una fábrica de constructores de documentos.
- 2º A partir de la instancia anterior, fabricamos un constructor de documentos, que procesará el XML.
- 3º Procesamos el documento XML y lo convertimos en árbol DOM.

```
try {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document documento=db.parse(Ruta_A_Archivo_Xml);

} catch (Exception ex) {
    System.out.println("¡Error! No se ha podido cargar el documento XML.");
}
```

Librerías para procesar documentos DOM a XML

- Es una tarea algo compleja y requiere usar el paquete ***javax.xml.transform***.
  - ***javax.xml.transform.TransformerFactory***: Factoría de transformadores que convierten el árbol DOM a XML.
  - ***javax.xml.transform.Transformer***. Clase que convierte el árbol DOM a XML.
  - ***javax.xml.transform.TransformerException***. Excepción que ocurre cuando se produce un fallo en la transformación.
  - ***javax.xml.transform.OutputKeys***. Contiene opciones de salida para el transformador. Por ejemplo, para indicar la codificación de caracteres (UTF-8).
  - ***javax.xml.transform.dom.DOMSource***. Actúa de intermediario entre el árbol DOM y el transformador, permitiendo al transformador acceder a la información del árbol DOM.
  - ***javax.xml.transform.stream.StreamResult***. Actúa de intermediaria entre el transformador y el archivo o *String* donde se almacena el documento XML.
  - ***java.io.File***. Permite leer y escribir un archivo, que será el documento XML.

Librerías para procesar documentos DOM a XML

- 1º Creamos una instancia de la clase ***File*** para acceder al archivo donde guardaremos el XML.
- 2º Creamos una nueva instancia del transformador a través de la fábrica de transformadores.
- 3º Establecemos algunas opciones de salida, como por ejemplo, la codificación de salida.
- 4º Creamos el ***StreamResult***, que intermediará entre el transformador y el archivo de destino.
- 5º Creamos el ***DOMSource***, que intermediará entre el transformador y el árbol DOM.
- 6º Realizamos la transformación.

```
try {
     File f=new File(Ruta_Al_Archivo_XML);
     Transformer transformer = TransformerFactory.newInstance().newTransformer();

     transformer.setOutputProperty(OutputKeys.INDENT, "yes");
     transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

     StreamResult result = new StreamResult(f);
     DOMSource source = new DOMSource(doc);

     transformer.transform(source, result);
} catch (TransformerException ex) {
     System.out.println("¡Error! No se ha podido llevar a cabo la transformación.");
}
```

Manipulación de documentos XML

- Para manipular el árbol DOM debemos emplear el paquete ***org.w3.dom***.
- Tras cargar el documento XML tendremos una instancia de ***org.w3.dom.Document*** donde encontramos diferentes tipos de clases.
  - ***org.w3.dom.Node***. Todos los objetos contenidos en el árbol DOM son nodos.
  - ***org.w3.dom.Element***. Corresponde a cualquier par de etiquetas (<> </>) y todo su contenido.
  - ***org.w3.dom.Attr***. Corresponde con cualquier atributo.
  - ***org.w3.dom.Comment***. Corresponde con un comentario
  - ***org.w3.dom.Text***. Corresponde con el texto entre etiquetas.

Manipulación de documentos XML

- ***Obtener el elemento raíz del documento***.
  - La variable ***document*** es una instancia de ***Document***.
  - El método ***getDocumentElement()*** devuelve el elemento raíz.

```
Element raiz=documento.getDocumentElement();
```

Manipulación de documentos XML

- **Buscar un elemento en toda la jerarquía del documento**
  - El método ***getElementsByTagName()*** devuelve una lista de nodos (***NodeList***) cuya etiqueta coincide con la buscada.

```
NodeList listaNodos=documento.getElementsByTagName("cliente");

Element cliente;

if (listaNodos.getLength()==1){
     cliente=(Element)listaNodos.item(0);
}
```

Manipulación de documentos XML

- **Obtener la lista de hijos de un elemento y procesarla.**
  - El método ***getChildNodes()*** devuelve la lista de nodos hijos de un nodo cualquiera.
  - El método ***getNodeType()*** permite saber de qué tipo de nodo se trata.

```
NodeList listaNodos=documento.getDocumentElement().getChildNodes();

for (int i=0; i<listaNodos.getLength();i++) {
    Node nodo=listaNodos.item(i);
    switch (nodo.getNodeType()){
        case Node.ELEMENT_NODE:
             Element elemento = (Element) nodo;
             System.out.println("Etiqueta:" + elemento.getTagName());
             break;
        case Node.TEXT_NODE:
             Text texto = (Text) nodo;
             System.out.println("Texto:" + texto.getWholeText());
             break;
    }
}
```

Manipulación de documentos XML

- **Añadir un nuevo elemento hijo a otro elemento**
  - Primero creamos los nodos a insertar.
  - Insertamos con el método ***appendChild()*** que lo añade al final de la lista de hijos. Otros métodos son:
    - ***insertBefore(Node nuevoNodo, Node nodoReferencia).*** Inserta el nuevo nodo antes del nodo de referencia.
    - ***replaceChild(Nodo nuevoNodo, Node nodoAnterior).*** Sustituye el nodo anterior por el nuevo.

```
Element direccionTag=documento.createElement("DIRECCION_ENTREGADA")
Text direccionTxt=documento.createTextNode("C/Perdida S/N");

direccionTag.appendChild(direccionTxt);
documento.getDocumentElement().appendChild(direccionTag);
```

Manipulación de documentos XML

- **Eliminar un elemento hijo de otro elemento.**
  - Primero hay que recurrir al nodo padre del nodo a borrar e invocar el método ***removeChild()*** al que se le pasa el objeto ***Element*** a borrar, lo que implica que hay que buscarlo primero.

```
NodeList listaNodos3=documento.getElementsByTagName("DIRECCION_ENTREGA");

for (int i=0;i<listaNodos3.getLength();i++){
     Element elemento=(Element) listaNodos3.item(i);
     Element padre = (Element)elemento.getParentNode();
     padre.removeChild(elemento);
}
```

Manipulación de documentos XML

- **Cambiar el contenido de un elemento cuando solo es texto**.
  - Los métodos ***getTextContent()*** y ***setTextContent()*** sobre un elemento, permiten consultar y modificar su contenido.
  - El método ***setTextContent()*** elimina cualquier hijo que tuviera previamente.

```
Element
nuevo=documento.createElement(DIRECCION_RECOGIDA").setTextContent("C/Del
Medio S/N");

System.out.println(nuevo.getTextContent());
```

Manipulación de documentos XML

- **Manejar los atributos de un elemento**.
  - ***setAttribute()***. Establece o crea el valor del atributo.
  - ***getAttribute()***. Obtiene el valor del atributo.
  - ***removeAttribute()***. Elimina el valor del atributo.

```
//Se crea el atributo “urgente” y se consulta
documento.getDocumentElement().setAttribute("urgente","no");
System.out.println(documento.getDocumentElement().getAttribute("urgente"));

//Se elimina el atributo “urgente”
documento.getDocumentElement().removeAttribute("urgente");
```

Unidad 8. Almacenando datos

FIN