Técnico en desarrollo de aplicaciones multiplataforma. Acceso a Datos.

Tema 3. Práctica

Profesor: ***Antonio Cuadros.***

MEDAC
Instituto Oficial de Formación Profesional

Consideraciones:

- *Para la relación de esta práctica deberá crear una carpeta llamada "3_1", en dicha carpeta creará un proyecto Java haciendo uso del IDE eclipse.*
- *Al finalizar la práctica deberá entregar en un fichero .zip que contenga el proyecto de Eclipse y este documento relleno.*
- *Se valorará la limpieza, estructuración y claridad del código (tabulaciones, saltos de líneas, nombres de variables significativos, etc.)*
- *Las líneas con este color en los ejercicios deberán ser borradas y sustituidas por lo solicitado.*

Yeray Santiago Santiago 2ºDAM
https://github.com/yeray124/acceso_a_datos.git

Ejercicio 1 guiado. (10 puntos) (Descargar archivo practica3_1.xml)

En este ejercicio se propone aprender paso a paso como leer información de un archivo XML haciendo uso de un parseador de tipo DOM. Durante la realización de este ejercicio deberá construir y entender el programa propuesto.

<u>Parte 1. Crear un parseador DOM:</u>

Tal y como hemos visto en la explicación teórica del tema 3, para crear un parseador de tipo DOM de la librería javax.xml.parsers necesitábamos

1.1 Crear un objeto de tipo DocumentBuilderFactory que nos permitirá crear un objeto de tipo DocumentBuilder (el propio parseador XML de tipo DOM).

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

```
factory.setValidating(true);
factory.setIgnoringElementContentWhitespace(true);
```

1.2 Podemos definir propiedades para los parseadores DOM que se vayan a crear a partir de la clase DocumentBuilderFactory como activar la validación del XML o ignorar los elementos vacíos que no son necesarios para un análisis XML.

1.3 Ahora tal y como hemos mencionado en el punto 1.1, creamos un parseador DOM haciendo uso de la clase DocumentBuilder (Recuerde que debe realizar un control de excepciones):

1.4 El siguiente punto será leer el propio documento XML haciendo uso del parser de tipo DOM, para ello creamos una variable de tipo File apuntando al fichero y almacenaremos en memoria el archivo XML haciendo uso de un dato de tipo Document (Los parseadores de tipo DOM almacenan en memoria todo el XML) (cambie el nombre del File):

```
File file = new File("tema3_3.xml");
Document doc = builder.parse(file);
doc.getDocumentElement().normalize();
```

Inserte a continuación una captura de pantalla del código creado hasta este punto (sin errores, deberá manejar excepciones):

```
        DocumentBuilder builder = factory.newDocumentBuilder();
```

```java
package paquete;

import java.io.File;

import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class VerificadorXML {

    public static void main(String[] args) {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        fabrica.setValidating(true);
        fabrica.setIgnoringElementContentWhitespace(true);

        try {
            DocumentBuilder constructor = fabrica.newDocumentBuilder();
            File xml = new File("tema3_1.xml");
            Document texto = constructor.parse(xml);
            texto.getDocumentElement().normalize();
        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }
    }

}
```

<u>Parte 2. Procesamiento de un fichero XML haciendo uso de un parseador DOM:</u>
Para procesar (en este caso leer) un archivo XML vamos a hacer uso de la clase XPath, la cual es una recomendación del W3C. Esta clase permite navegar y seleccionar nodos dentro de un documento XML.

2.1 Para crear una instancia de la clase XPath lo hacemos de la siguiente forma:

```java
XPath xPath = XPathFactory.newInstance().newXPath();
```

2.2 Lo siguiente que debemos hacer es crear lo que se conoce en XPath como "expresión". Esto es un String que se utiliza para navegar por un documento XML. Por ejemplo, crearemos la siguiente expresión que nos permitirá buscar todos los nodos <student> que son hijos del nodo <class>:

```java
String expression = "/class/student";
```

2.3 A continuación vamos a crear un objeto de tipo NodeList (lista de nodos, almacenará los nodos analizados por nuestro parseador y almacenado en la variable "doc"):

```java
NodeList nodeList = (NodeList) xPath.compile(expression)
                                    .evaluate(doc, XPathConstants.NODESET);
```

Vamos a explicar la línea anterior con más detalle:
2.3.1 XPath.compile(expresion): compila (Significa preparar o transformar la expresión (escrita como un string) en un formato que el motor de XPath pueda interpretar y ejecutar eficientemente.) y almacena la expresión definida anteriormente para su uso.

2.3.2 .evaluate(doc, XPathConstants.NODESET): Ejecuta la expresión definida sobre el documento especificado en el primer parámetro y obtiene todos los nodos que cumplen esa expresión. El segundo argumento especifica que queremos que devuelva un conjunto de nodos.

2.4 En este punto ya tenemos la lista de nodos de nuestro archivo XML que son de tipo student e hijos directos de class. Ahora se propone imprimir de nuestro archivo XML todos los datos de estos estudiantes:

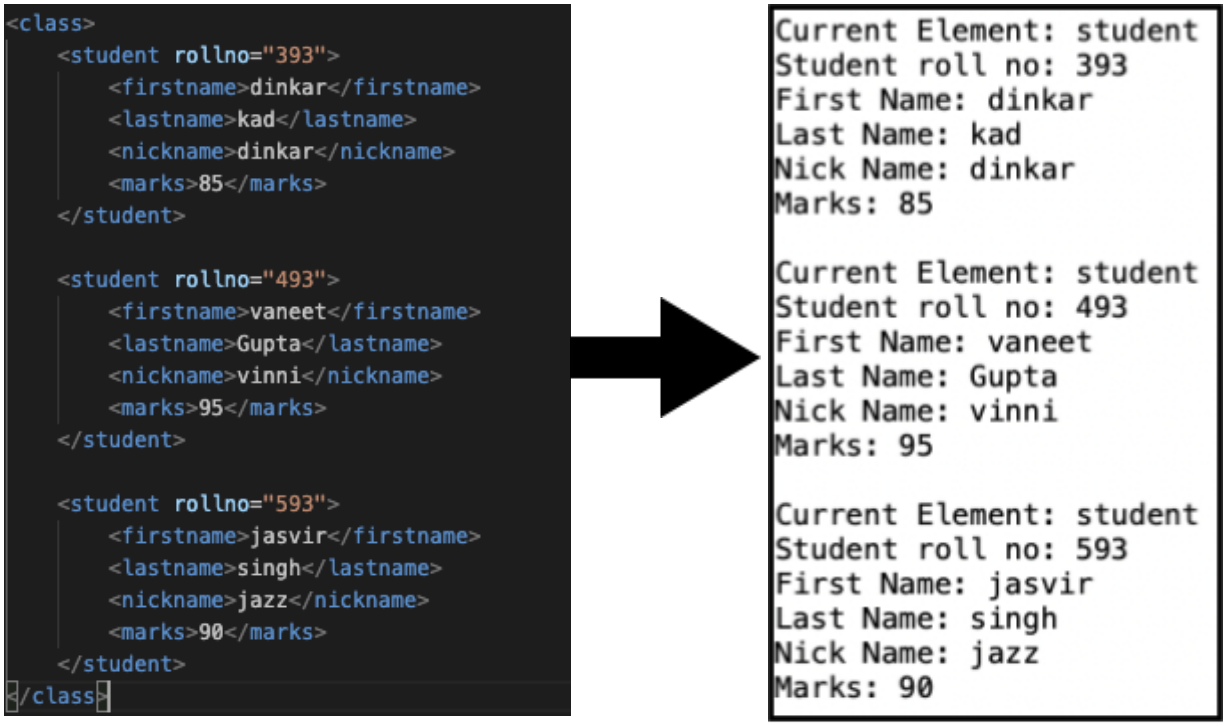

Figura 1. Resultado esperado

Para ello tendremos que recorrer nuestra lista de nodos haciendo uso de un bucle de la siguiente forma, donde en cada iteración obtenemos un determinado nodo. En este caso, cada nodo es un elemento de tipo "student" ya que así lo hemos especificado en la expresión utilizada por XPath):

```
for (int i = 0; i < nodeList.getLength(); i++) {
    Node nNode = nodeList.item(i);

}
```

(Nota: todo el código mostrado a continuación se sitúa dentro del bucle anteriormente mencionado)

2.5 Si por ejemplo queremos mostrar "Current Element: student" tal y como muestra la Figura 1 podemos utilizar el siguiente método (getNodeName() obtiene el nombre del nodo):

```
System.out.println("\nCurrent Element: " + nNode.getNodeName());
```

2.6 Para mostrar por ejemplo "Student Roll no: …", es decir un atributo de un nodo, podemos utilizar el siguiente método getAttribute(String attribute):

```
Element eElement = (Element) nNode;
System.out.println("Student roll no: " + eElement.getAttribute("rollno"));
```

Se hace un casting a la clase "Element" para poder utilizar más métodos interesantes para este ejercicio guiado.

2.7 Para obtener la siguiente línea de la Figura 1 "First Name: …" podemos utilizar el método getElementsbyTagName("String tagName"):

```
System.out.println("First Name: "
    + eElement
        .getElementsByTagName("firstname")
        .item(0)
        .getTextContent());
```

Inserte a continuación una captura de pantalla del código creado hasta este punto (sin errores, deberá manejar excepciones) y que permita imprimir por pantalla el resto de elementos restantes tal y como muestra Figura 1:

```java
package paquete;

import java.io.File;

public class VerificadorXML {

    public static void main(String[] args) throws XPathExpressionException {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        fabrica.setValidating(true);
        fabrica.setIgnoringElementContentWhitespace(true);

        try {
            DocumentBuilder constructor = fabrica.newDocumentBuilder();
            File xml = new File("tema3_3.xml");
            Document texto = constructor.parse(xml);
            XPath ruta =XPathFactory.newInstance().newXPath();

            String recorrer = "/class/student";
            NodeList nodos = (NodeList) ruta.compile(recorrer).evaluate(texto, XPathConstants.NODESET);
            texto.getDocumentElement().normalize();

            for (int i= 0; i < nodos.getLength(); i++) {
                Node nodos2 = nodos.item(i);

                System.out.println("\nCurrent Element: " + nodos2.getNodeName());
                Element objeto = (Element) nodos2;
                System.out.println("Student roll no: " + objeto .getAttribute("rollno"));
                System.out.println("Primer nombre: " + objeto.getElementsByTagName("firstname").item(0).getTextContent());
                System.out.println("Ultimo nombre: " + objeto.getElementsByTagName("lastname").item(0).getTextContent());
                System.out.println("Apodo: " + objeto.getElementsByTagName("nickname").item(0).getTextContent());
                System.out.println("Puntos: " + objeto.getElementsByTagName("marks").item(0).getTextContent());

            }

        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }
    }

}
```

```
Current Element: student
Student roll no: 393
Primer nombre: dinkar
Ultimo nombre: kad
Apodo: dinkar
Puntos: 85

Current Element: student
Student roll no: 493
Primer nombre: vaneet
Ultimo nombre: Gupta
Apodo: vinni
Puntos: 95

Current Element: student
Student roll no: 593
Primer nombre: jasvir
Ultimo nombre: singh
Apodo: jazz
Puntos: 90
```

Ejercicio 2. (Op) (Descargar archivo practica3_2.xml)

Lea y muestre la información del XML de forma similar al ejercicio guiado haciendo uso del nuevo fichero "practica3_2.xml" disponible en la plataforma. Deberá mostrar por consola el siguiente resultado:

```
Current Element: persona
Nombre: Juan Pérez
Edad: 30
Mascota Nombre: Rex
Mascota Tipo: Perro

Current Element: persona
Nombre: María García
Edad: 25
Mascota Nombre: Whiskers
Mascota Tipo: Gato

Current Element: persona
Nombre: Carlos Sánchez
Edad: 40
Mascota Nombre: Nemo
Mascota Tipo: Pez

Current Element: persona
Nombre: Laura Fernández
Edad: 28
Mascota Nombre: Charlie
Mascota Tipo: Hámster
```

```java

    public static void main(String[] args) throws XPathExpressionException {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance(), fabrica2 = DocumentBuilderFactory.newInstance();
        fabrica.setValidating(true);
        fabrica.setIgnoringElementContentWhitespace(true);
        fabrica2.setValidating(false);
        fabrica2.setIgnoringElementContentWhitespace(true);

        try {
            DocumentBuilder constructor = fabrica.newDocumentBuilder();
            DocumentBuilder constructor2 = fabrica2.newDocumentBuilder();
            File xml = new File("tema3_3.xml"), xml2 = new File("tema3_4.xml");
            Document texto = constructor.parse(xml), texto2 = constructor2.parse(xml2);
            XPath ruta =XPathFactory.newInstance().newXPath(), ruta2 =XPathFactory.newInstance().newXPath();

            String recorrer = "/class/student", recorrer2 = "/personas/persona";
            NodeList nodos = (NodeList) ruta.compile(recorrer).evaluate(texto, XPathConstants.NODESET), nodosotro = (NodeList) ruta2.compile(recorrer2).evaluate(texto2, XPathConstants.NODESET);
            texto.getDocumentElement().normalize();
            texto2.getDocumentElement().normalize();

            for (int i= 0; i < nodos.getLength(); i++) {
                Node nodos2 = nodos.item(i);

                System.out.println("\nCurrent Element: " + nodos2.getNodeName());
                Element objeto = (Element) nodos2;
                System.out.println("Student roll no: " + objeto .getAttribute("rollno"));
                System.out.println("Primer nombre: " + objeto.getElementsByTagName("firstname").item(0).getTextContent());
                System.out.println("Ultimo nombre: " + objeto.getElementsByTagName("lastname").item(0).getTextContent());
                System.out.println("Apodo: " + objeto.getElementsByTagName("nickname").item(0).getTextContent());
                System.out.println("Puntos: " + objeto.getElementsByTagName("marks").item(0).getTextContent());

            }


            for (int z= 0; z < nodosotro.getLength(); z++) {
                    Node nodos3 = nodosotro.item(z);
                    Element objeto2 = (Element) nodos3;

                    System.out.println("\nCurrent Element: " + nodos3.getNodeName());
                    System.out.println("Nombre: " + objeto2.getElementsByTagName("nombre").item(0).getTextContent());
                    System.out.println("Edad: " + objeto2.getElementsByTagName("edad").item(0).getTextContent());

                    Node mascotaNode = objeto2.getElementsByTagName("mascota").item(0);
                        Element mascotaElement = (Element) mascotaNode;

                        System.out.println("Nombre de mascota: " + mascotaElement.getElementsByTagName("nombre").item(0).getTextContent());
                        System.out.println("Tipo de mascota: " + mascotaElement.getElementsByTagName("tipo").item(0).getTextContent());

            }

        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }
```

```
Current Element: student
Student roll no: 393
Primer nombre: dinkar
Ultimo nombre: kad
Apodo: dinkar
Puntos: 85

Current Element: student
Student roll no: 493
Primer nombre: vaneet
Ultimo nombre: Gupta
Apodo: vinni
Puntos: 95

Current Element: student
Student roll no: 593
Primer nombre: jasvir
Ultimo nombre: singh
Apodo: jazz
Puntos: 90

Current Element: persona
Nombre: Juan Pérez
Edad: 30
Nombre de mascota: Rex
Tipo de mascota: Perro

Current Element: persona
Nombre: María García
Edad: 25
Nombre de mascota: Whiskers
Tipo de mascota: Gato

Current Element: persona
Nombre: Carlos Sánchez
Edad: 40
Nombre de mascota: Nemo
Tipo de mascota: Pez

Current Element: persona
Nombre: Laura Fernández
Edad: 28
Nombre de mascota: Charlie
Tipo de mascota: Hámster
```