**Java DOM writing example**

In the following example, we create an XML file.

**JavaXmlDomWrite.java**

package com.zetcode;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

public class JavaXmlDomWrite {

public static void main(String[] args) throws ParserConfigurationException,

TransformerException {

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document doc = builder.newDocument();

Element root = doc.createElementNS("zetcode.com", "users");

doc.appendChild(root);

root.appendChild(createUser(doc, "1", "Robert", "Brown", "programmer"));

root.appendChild(createUser(doc, "2", "Pamela", "Kyle", "writer"));

root.appendChild(createUser(doc, "3", "Peter", "Smith", "teacher"));

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transf = transformerFactory.newTransformer();

transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

transf.setOutputProperty(OutputKeys.INDENT, "yes");

transf.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");

DOMSource source = new DOMSource(doc);

File myFile = new File("src/main/resources/users.xml");

StreamResult console = new StreamResult(System.out);

StreamResult file = new StreamResult(myFile);

transf.transform(source, console);

transf.transform(source, file);

}

private static Node createUser(Document doc, String id, String firstName,

String lastName, String occupation) {

Element user = doc.createElement("user");

user.setAttribute("id", id);

user.appendChild(createUserElement(doc, "firstname", firstName));

user.appendChild(createUserElement(doc, "lastname", lastName));

user.appendChild(createUserElement(doc, "occupation", occupation));

return user;

}

private static Node createUserElement(Document doc, String name,

String value) {

Element node = doc.createElement(name);

node.appendChild(doc.createTextNode(value));

return node;

}

}

The example creates a new users.xml file in the src/main/resources directory.

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

A new document builder is created from a document builder factory.

Document doc = builder.newDocument();

From the document builder, we create a new document with newDocument.

Element root = doc.createElementNS("zetcode.com", "users");

doc.appendChild(root);

We create a root element and add it to the document with appendChild.

root.appendChild(createUser(doc, "1", "Robert", "Brown", "programmer"));

root.appendChild(createUser(doc, "2", "Pamela", "Kyle", "writer"));

root.appendChild(createUser(doc, "3", "Peter", "Smith", "teacher"));

We append three child elements to the root element.

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transf = transformerFactory.newTransformer();

Java DOM uses a Transformer to generate the XML file. It is called transformer, because it can also transform the document with XSLT language. In our case, we only write to the XML file.

transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

transf.setOutputProperty(OutputKeys.INDENT, "yes");

transf.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");

We set the encoding and indentation of the document.

DOMSource source = new DOMSource(doc);

The DOMSource holds the DOM tree.

StreamResult console = new StreamResult(System.out);

StreamResult file = new StreamResult(myFile);

We are going to write to a console and to a file. StreamResult is a holder of a transformation result.

transf.transform(source, console);

transf.transform(source, file);

We write the XML sources to the stream results.

private static Node createUser(Document doc, String id, String firstName,

String lastName, String occupation) {

Element user = doc.createElement("user");

user.setAttribute("id", id);

user.appendChild(createUserElement(doc, "firstname", firstName));

user.appendChild(createUserElement(doc, "lastname", lastName));

user.appendChild(createUserElement(doc, "occupation", occupation));

return user;

}

A new user element is created in the createUser method with createElement. An attribute of the element is set with setAttribute.

private static Node createUserElement(Document doc, String name,

String value) {

Element node = doc.createElement(name);

node.appendChild(doc.createTextNode(value));

return node;

}

An element is added to its parent with appendChild and a text node is created with createTextNode.

In this article we have read and written XML files with Java DOM API.

**Referencias**

https://zetcode.com/java/dom/