TP 6 Java et XML

I. API JAXP/DOM (Document Object Model)

Ecrire une petite application DOM permettant d'imprimer sur la sortie standard la liste des titres des recettes de cuisine (lecture du document et parcours de l'arbre). Vous pouvez utiliser la JavaDoc du package org.w3c.dom[1] pour vous aider ainsi que l'exemple présenté en cours.

II. L'API TrAX (Transformation API for XML)

1) Construction d'un arbre DOM

Reprenez l'exercice précédent, et construisez un document DOM qui contient le résultat. Imprimez ce document (sous sa forme XML) en utilisant l'API TraX (Transformation API for XML). Pour ce faire, créer un Transformer avec l'aide d'un TransformerFactory. Vous devrez utiliser les packages suivants :

– package javax.xml.parsers[2] : Interface vers les analyseurs JAXP,
– package javax.xml.transform[3] : API TRaX de transformation,
– package javax.xml.transform.dom[4] : API TRaX spécifique à DOM,
– package javax.xml.transform.sax[5] : API TRaX spécifique à SAX,
– package javax.xml.transform.stream[6] : API TRaX spécifique aux streams.

Pour vous aider, voilà un petit exemple de création d'un arbre DOM :

[1] http://www.dil.univ-mrs.fr/docs/java/api/org/w3c/dom/package-summary.html
[2] http ://www.dil.univ-mrs.fr/docs/java/api/javax/xml/parsers/package-summary.html
[3] http://www.dil.univ-mrs.fr/docs/java/api/javax/xml/transform/package-summary.html
[4] http://www.dil.univ-mrs.fr/docs/java/api/javax/xml/transform/dom/package-summary.html
[5] http ://www.dil.univ-mrs.fr/docs/java/api/javax/xml/transform/sax/package-summary.html
[6] http://www.dil.univ-mrs.fr/docs/java/api/javax/xml/transform/stream/package-summary.html

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
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;


public class SampleCreateDom {


      public static void main(String[] args) throws Exception {
// création d'un document vide
Document doc = DocumentBuilderFactory.newInstance()
.newDocumentBuilder().newDocument();
// ajout de noeuds
Element racine = doc.createElement("racine");
racine.appendChild(doc.createTextNode("hello"));
doc.appendChild(racine);
// sérialisation
TransformerFactory myFactory = TransformerFactory.newInstance();
Transformer transformer = myFactory.newTransformer();
transformer.setOutputProperty(OutputKeys.ENCODING, "iso-8859-1");
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.transform(new DOMSource(doc),
new StreamResult(System.out));
}
                                    }
```

2) Application d'une feuille de style XSL

Reprenez l'exercice précédent, et construisez une feuille de style (simple) pour produire du XHTML à partir du document XML résultat. Faites en sorte de lire cette feuille de style sous la forme d'un document DOM (DOMSource) et appliquez cette feuille de style (c'est une nouvelle transformation) sur le document XML résultat.
Votre application doit maintenant imprimer la liste des noms des recettes de cuisine sous la forme d'une page XHTML (en ISO-8859 indentée).

III. Utiliser Java API for XML Binding (JAXB)

En vous appuyant sur l'exemple JAXB vu en cours, vous allez :

- Générer les classes Java qui correspondent à votre schéma des recettes de cuisine. Parcourez les fichiers générés.
- Créer une nouvelle classe dans laquelle vous allez définir une méthode qui permet de récupérer la liste des titres de recettes et leur difficulté. Vous pouvez vous baser sur l'exemple vu en cours pour la désérialisation.

- Vous allez maintenant produire l'opération inverse c'est-à-dire la sérialisation (ou marshalling). L'opération de marshalling[7] (ou de sérialisation) est l'étape de reconstruction du document XML à partir de sa représentation objet. Le code principal du marshalling ressemblera à ceci :

```
JAXBContext contexte = JAXBContext.newInstance( "monpkg" );
…
Marshaller ms = context.createMarshaller();
ms.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE );
ms.marshal( element, new FileWriter( "newfile" ) );
```

- Exécuter votre application
- Vous allez modifier votre application pour qu'elle ajoute une nouvelle recette avant de la sérialiser.

IV. L'API SAX (Simple API for XML)

Ecrire une petite application SAX permettant d'extraire une information particulière d'un fichier XML. Par exemple, le nom des recettes de cuisine. Vous pouvez utiliser la JavaDoc du package org.xml.sax pour vous aider.

Vous pouvez également utiliser JAXP pour vous procurer l'analyseur SAX.

```
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setValidating( true );
SAXParser sp = factory.newSAXParser();
XMLReader reader = sp.getXMLReader();
reader.setContentHandler( ... );

reader.parse( "file" );
```

La classe SAXParserFactory masque en réalité la façon dont le parseur va être construit. L'objet retourné, de type SAXParserFactory, dispose d'une méthode setValidating pour garantir la validation en cas d'usage d'une DTD. La méthode getXMLReader rend le parseur compatible SAX.

[7] http://jmdoudoux.developpez.com/cours/developpons/java/chap-jaxb.php