



FIP INF 204 : XML et Web

TP 2 : SAX, DOM, XSLT

1 *But du TP*

Le but de ce TP est de se familiariser avec deux parmi les trois techniques de transformation de document XML les plus importantes : SAX (*Simple API for XML*), DOM (*Document Object Model*). La troisième étant XSLT à laquelle nous consacrerons les TPs 3 et 4.

Pour cela nous allons convertir le même document RSS en XHTML, en utilisant les trois technologies. Le document, que l'on peut récupérer sur moodle (lemonde.xml) est tiré du flux RSS du *Monde* (<http://www.lemonde.fr>) du 5 octobre 2005.

2 *Ressources*

Pour les parties SAX et DOM nous allons utiliser une application Java de la distribution Apache : *Xerces* (analyseur XML, compatible DTD et schémas) (<http://xml.apache.org>). Par contre, pour XSLT nous allons utiliser la version 8 de *Saxon*, une application Java diffusée par la société *Saxonica* (<http://www.saxonica.com>).

Les fichiers `xercesImpl.jar`, `xml-apis.jar` et `saxon8.jar`, que vous trouverez sur moodle : <http://formations.enst-bretagne.fr/fad>

dans «FIP INF 204/Ressources». doivent être mis dans le répertoire courant.

3 *Description du document XML*

Voici une liste des éléments et attributs que l'on trouve dans le document `lemonde.xml` (à récupérer sur moodle) :

```
<!ELEMENT channel (#PCDATA|%textstuff;)*>
<!ELEMENT copyright (#PCDATA|%textstuff;)*>
<!ELEMENT description (#PCDATA|%textstuff;)*>
<!ELEMENT guid (#PCDATA|%textstuff;)*>
<!ATTLIST guid isPermaLink CDATA #REQUIRED
>
```

```

<!ELEMENT image (#PCDATA|%textstuff;)*>
<!ELEMENT item (#PCDATA|%textstuff;)*>
<!ELEMENT link (#PCDATA|%textstuff;)*>
<!ELEMENT pubDate (#PCDATA|%textstuff;)*>
<!ELEMENT rss (#PCDATA|%textstuff;)*>
<!ATTLIST rss version CDATA #REQUIRED
xmlns:rdf CDATA #REQUIRED
>
<!ELEMENT title (#PCDATA|%textstuff;)*>
<!ELEMENT url (#PCDATA|%textstuff;)*>

```

Dans ce document on trouve un préambule :

```

<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<channel>
<title>Le Monde.fr : A la Une</title>
<link>http://www.lemonde.fr</link>
<description>Toute l'actualité au moment de la connexion</description>
<copyright>Copyright Le Monde.fr</copyright>
<image>
  <url>http://medias.lemonde.fr/mmpub/img/lgo/lemondefr_rss.gif</url>
  <title>Le Monde.fr</title><link>http://www.lemonde.fr</link>
</image>
<pubDate>Wed, 05 Oct 2005 22:00:00</pubDate>

```

contenant des éléments concernant le flux RSS en général, et un certain nombre de dépêches. Chaque dépêche d'écrit de la manière suivante :

```

<item>
  <title>le titre...</title>
  <link>l'URL de l'article HTML correspondant</link>
  <description>texte de la dépêche...</description>
  <pubDate>date et heure</pubDate>
  <guid isPermaLink="false">de nouveau une URL</guid>
</item>

```

Notre but sera de convertir le préambule et les dépêches en XHTML, de manière à ce qu'elles soient agréablement présentées (que ce soit par le biais d'un tableau ou d'une liste, à vous de voir). Il faut nécessairement que les informations suivantes y figurent dans chaque dépêche : date, titre, contenu textuel, URL (sous forme de lien hypertexte).

4 SAX

Pour écrire le code Java, on prendra comme modèle le fichier SAXParserDemo.java que l'on récupérera sur moodle. On le compilera et l'on l'exécutera en écrivant (sous Unix) :

```

javac -g -classpath xercesImpl.jar:xml-apis.jar:. SAXParserDemo.java
java -classpath xercesImpl.jar:xml-apis.jar:. SAXParserDemo lemonde.xml > lemonde1.html

```

ou (sous Windows) :

```
javac -g -classpath xercesImpl.jar;xml-apis.jar;. SAXParserDemo.java
java -classpath xercesImpl.jar;xml-apis.jar;. SAXParserDemo lemonde.xml > lemonde1.html
```

en mettant éventuellement le chemin d'accès complet des binaires et des archives JAR.

5 DOM

Pour cela nous allons écrire le code correspondant dans la méthode `transform()` de la classe `DOMParserDemo`, à récupérer sur moodle. On la compilera et l'on l'exécutera en écrivant (Unix) :

```
javac -g -classpath xercesImpl.jar:xml-apis.jar:. DOMParserDemo.java
java -classpath xercesImpl.jar:xml-apis.jar:. DOMParserDemo lemonde.xml > lemonde2.html
```

ou (Windows) :

```
javac -g -classpath xercesImpl.jar;xml-apis.jar;. DOMParserDemo.java
java -classpath xercesImpl.jar;xml-apis.jar;. DOMParserDemo lemonde.xml > lemonde2.html
```

Tuyaux : On «trafiquera» la méthode `printNode` pour obtenir un comportement différent selon le nom de l'élément (on revient ainsi au principe de SAX).

Pour obtenir un `ol` ou un `table` avant le code HTML correspondant à la première dépêche il y a une solution élégante et (au moins) une solution de bricolage.

La solution élégante (pour programmeurs courageux !) consiste à créer un nœud `items` sous `channel` et à lui attacher tous les nœuds `item` en faisant autant de `appendChild()`. Attention : parcourir la liste des nœuds `item` de la fin vers le début pour éviter les effets de bord.

La solution de bricolage consiste à compter les `item` à l'intérieur de la classe `printNode` et à placer la chaîne `` ou `<table>` avant le premier. La chaîne `` (ou `</table>`) peut être déclenchée par la balise `</channel>`.

6 XSLT (à faire en début de TP 3)

Pour cela nous allons écrire une feuille de style XSLT pour la même transformation (un modèle se trouve sur moodle : fichier `lemonde.xsl`). Le processeur *Saxon* est contenu dans l'archive JAR `saxon8.jar`. Pour appliquer la feuille de style `lemonde.xsl` au fichier XML `lemonde.xml`, faire :

```
java -jar saxon8.jar lemonde.xml lemonde.xsl > lemonde3.html
```

Pour afficher les options de *Saxon*, on peut utiliser :

```
java -jar saxon8.jar -?
```

La documentation complète de *Saxon* se trouve à l'adresse suivante :

<http://www.saxonica.com/documentation/documentation.html>