62-31.3 XML

XML-04-P01-XSLT

Cédric Benoit

Plan du cours

- Présentation et organisation
- Introduction à XML
- Schémas XML
- Types de données
- **Transformation XML**
  - **XSLT**

Points abordés

- Introduction
- CSS ⇔ XSL
- Rappel XML
- DOM et SAX
- Templates et instructions
- Exemples
- Questions

Introduction

Au début il y avait : HTML

- CSS = Feuilles de style pour HTML

Puis est arrivé : XML

- XSL = Feuilles de style pour XML composé de :
  - XSLT : Transformation
  - XPath : Navigation
  - XSL-FO : Formatage

CSS ⇔ XSL (1)

- CSS (Cascading Style Sheets) est souvent utilisée pour la présentation d'une page HTML
- Ce langage est normalisé par W3C et apparaît au milieu des années 1990
- Par rapport à XSL, CSS:
  - ne peut pas changer l'ordre d'apparition des éléments dans un document
  - ne peut pas effectuer de calculs
  - ne peut pas combiner plusieurs documents

CSS ⇔ XSL (2)

- XSLT (eXtensible Stylesheet Language Transformations) est un langage de transforamtion XML et permet de transformer un document XML dans un autre format (HTML, texte,…)
- XSLT:
  - est un document XML
  - basé sur la reconnaissance de motifs (pattern matching)
  - constitué de règles (rules)
  - utilise des itérations et la récursivité pour parcourir les données

Rappel XML

- Balise racine obligatoire
- Imbrication correcte
- Balise avec ou sans attributs
- Balises "Case sensitive"
- Balise de fin obligatoire (sauf pour balise vide)
- Structure peut être décrite par un schéma (DTD, XSD, Relax NG,...)
- Document bien formé : respecte la norme XML
- Document valide : respecte le schéma

DOM et SAX

Un document XML est accessible à différents langages (Java, C#,...), selon deux méthodes :

- DOM (Document Object Model) est une API standardisée du W3C, permettant de constituer un objet en mémoire, de la totalité du document XML
- SAX (Simple API for XML) est une API standardisée (développée à l'origine pour Java), traitant un document XML élément par élément, au fur et à mesure de la lecture document XML

DOM (1)

- DOM (Document Object Model) construit un arbre en mémoire représentant le document XML
- La construction de l'arbre est constituée de différents objets:
  - Node
    - Type de base d'une structure DOM
  - Document
    - Contient la représentation dom d'un document XML

DOM (2)

- Element
  - Element XML
- Attr
  - Attribut XML
- Text
  - Morceau de texte d'un document XML
  - Tout texte xml devient un élément Text
  - Texte d'un élément ou d'un attribut
- Commentaires
  - Commentaires xml <!--... -->

DOM (3)

- ProcessingInstruction
  - Directives de traitement qui peuvent dépendre du processeur utilisé

- Exemple :
  - `<?xml-stylesheet href="case-study.xsl" type="text/xsl"?>`

DOM (4)

- Un "parseur" DOM :
  - Lit complètement le document XML
  - Crée un arbre en mémoire représentant la structure XML
  - Arrange chaque élément sur un arbre
  - Chaque élément a des propriétés et peut avoir des "enfants"
  - Redonne le contrôle uniquement lorsque l'arbre mémoire est complet

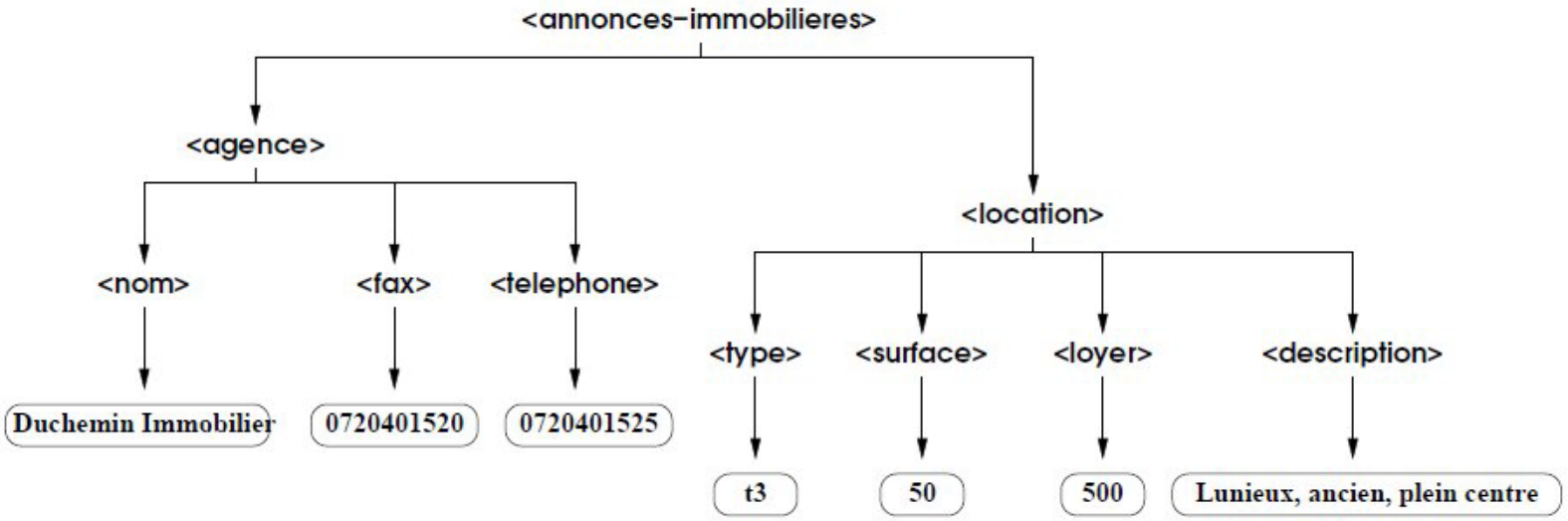
DOM (5)

Exemple de document XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<annonces-immobilieres>
 <agence>
  <nom>Duchemin Immobilier</nom>
  <telephone>0720401520</telephone>
  <fax>0720401525</fax>
 </agence>
 <location type="appartement">
  <quartier>Centre-Ville</quartier>
  <type>T3</type>
  <surface unite="m2">50</surface>
  <loyer type="hc" monnaie="euros">570</loyer>
  <description>
   Lumineux, ancien, plein centre
  </description>
 </location>
</annonces-immobilieres>
```

DOM (6)

Exemple – Structure arborescente



DOM (7) – Exemple DOM avec Java

Exemple de recherche de l'élément racine du document XML

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;

public class GetRootNode{
  public static void main(String[] args) {
    try{
      BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
      System.out.print("Enter xml file name: ");
      String str = bf.readLine();
      File file = new File(str);
      if (file.exists()){
        DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fact.newDocumentBuilder();
        Document doc = builder.parse(str);
        Node node = doc.getDocumentElement();
        String root = node.getNodeName();
        System.out.println("Root Node: " + root);
      }
      else{
        System.out.println("File not found!");
      }
    }
    catch(Exception e){}
  }
}
```

SAX (1)

- SAX Simple Api for Xml
  - Intéractif
  - Ne charge pas tout le document XML en mémoire
  - Analyse et émet des messages au fur et à mesure qu'il trouve des éléments
  - Ne voit pas la structure du document XML

SAX (2) - Exemple SAX avec Java

Exemple de recherche du nombre total de cuillères à Farine que l'on a besoin Dans les recettes

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.apache.xerces.parsers.SAXParser;

public class Flour extends DefaultHandler {

  float amount = 0;

  public void startElement(String namespaceURI, String localName,
                           String qName, Attributes atts) {
    if (namespaceURI.equals("http://recipes.org") && localName.equals("ingredient")) {
       String n = atts.getValue("","name");
       if (n.equals("flour")) {
         String a = atts.getValue("","amount"); // assume 'amount' exists
         amount = amount + Float.valueOf(a).floatValue();
       }
    }
  }

  public static void main(String[] args) {
    Flour f = new Flour();
    SAXParser p = new SAXParser();
    p.setContentHandler(f);
    try { p.parse(args[0]); }
    catch (Exception e) {e.printStackTrace();}
    System.out.println(f.amount);
  }
}
```

Templates (1)

Template

- Règle qui transforme un ensemble de nœuds xml d'un document source dans un autre format

Forme générale :

```
<xsl:template  match="…">
…
</xsl:template>
```

Templates (2)

match = "..."

- ... = Chemin XPath
- Spécifie la liste des nœuds qui seront traités par la règle

Exemple :

- <xsl:template match ="*">
- <xsl:template match ="/">
- <xsl:template match ="/book/price">

Templates (3)

Soit le fichier XML suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- score.xml -->
<scores>
    <score id="1">
    <film>A Little Princess</film>
    <composer>Patrick Doyle</composer>
    <year>1995</year>
    <grade>100</grade>
    </score>
...
```

Templates (4)

```
<score id="2">
<film>Chocolat</film>
<composer>Rachel Portman</composer>
<year>2001</year>
<grade>90</grade>
</score>
<score id="3">
<film>Vertigo</film>
<composer>Bernard Herrmann</composer>
<year>1956</year>
<grade>95</grade>
</score>…
```

Templates (5)

```
    <score id="4">
    <film>Field of Dreams</film>
    <composer>James Horner</composer>
    <year>1989</year>
    <grade>96</grade>
    </score>
    <score id="5">
    <film>Dead Again</film>
    <composer>Patrick Doyle</composer>
    <year>1991</year>
    <grade>97</grade>
    </score>
</scores>
```

Templates (6)

Si on désire effectuer une opération avec tous les éléments film du fichier, il suffit de créer une directive template :

```
<xsl:template match="score">
<!– Traiter les éléments -->
</xsl:template>
```

Templates (7)

Exemple :

```
<xsl:template match="score">
The film: <xsl:value-of select="film"/>
</xsl:template>
```

- Résultat :

The film: A Little Princess

The film: Chocolat

The film: Vertigo

The film: Field of Dreams

The film: Dead Again

Traitement des templates

Lors du traitement d'un template : le processeur XSLT :

- Lit le document XML
- Génère un arbre
- Parcourt complètement l'arbre en recherchant l'élément qui correspond le mieux à la règle définie par le template
- Lorsque il y a correspondance, le template sert de guide pour construire l'ensemble des éléments sélectionnés

Instructions (1)

xsl:apply-templates

- Applique les templates définis pour tous les nœuds en cours ainsi que tous les "enfants "

xsl:apply-templates select ="..."

- Ne traite que les nœuds qui correspondent à la sélection

Instructions (2)

Exemple :

```
<xsl:template match="score">
<xsl:apply-templates/>
</xsl:template>
```

Résultat :

A Little Princess

Patrick Doyle

1995

100

Chocolat

Rachel Portman

…

Instructions (3)

Exemple :

```
<xsl:template match="score">
<xsl:apply-templates select="grade"/>
<xsl:apply-templates select="film"/>
</xsl:template>
```

Résulta t:

100 A Little Princess

90 Chocolat

95 Vertigo

…

Instructions (4)

<xsl-copy>

- Copie le nœud courant
- Garde les balises xml
- Ne copie pas les enfants
- Ne copie pas les attributs
- Ajoute le contenu du texte uniquement si une balise apply-templates est inclue

Instructions (5)

Exemple :

```
<xsl:template match="composer">
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsl:template>
```

Résultat :

```
<composer>Patrick  Doyle</composer>
<composer>Rachel Portman</composer>
…
```

Instructions (6)

xsl:copy-of select="..."

- Copie l'ensemble de nœuds définis par select
- Garde les balises xml
- Copie les enfants, attributs et le contenu
- Garde les commentaires et le PI

Instructions (7)

Exemple:

```
<xsl:template match="score">
<xsl:copy-of select="."/>
</xsl:template>
```

Résultat:

```
<score id="1">
<film>A Little Princess</film>
<composer>Patrick Doyle</composer>
<year>1995</year>
<grade>100</grade>
</score>
…
```

Instructions (8)

xsl:value-of select="..."

- Convertit le contenu d'un élément en texte
- Si le contenu est un ensemble de nœuds, seul le premier nœud est converti
- Ne conserve pas les balises
- Ne convertit pas les attributs
- Pour récupérer un attribut, il faut utiliser la syntaxe select ="@..."

Instructions (9)

Exemple:

```
<xsl:template match="score">
<xsl:value-of select="film"/>
</xsl:template>
```

Résultat:

A Little Princess

Chocolat

…

Instructions (10)

Exemple:

```
<xsl:template match="score">
<xsl:value-of select="@id"/>
</xsl:template>
```

Résultat:

1

2

3

....

Instructions (11)

Template nommé se déclare :

- <xsl:template name ="…">

- Ne contient pas d'attribut match
- Est appelé par xsl:call-template

Instructions (12)

Exemple :

```
<xsl:template name="CreditLine">
<xsl:value-of select="."/> - Brought to you by Tumplates, The Template People.
</xsl:template>

<xsl:template match="composer">
Composer: <xsl:call-template name="CreditLine"/>
</xsl:template>

<xsl:template match="year"/>
<xsl:template match="grade"/>
```

Variables (1)

- XSLT permet de définir des variables de deux manières différentes
- Content variables

<xsl:variable name = "…"> …. </xsl:variable>

- Utilisation : $ devant le nom de la variable

<xsl:value-of select="$mystate"/>

Variables (2)

Select attribute variable

- <xsl:variable name="..." select="Expression"/>
- Expression est évalué avant d'assigner la valeur à la variable

Exemple :

<xsl:variable name="Total" select="round(5.129)"/>

Variables (3)

Exemple :

```
<xsl:variable name="pgFilms" select="//film[mpaa='PG']"/>
<xsl:variable name="totalNumOfPgFilms"
select="count($pgFilms)"/>
<xsl:template match="/">
<p>The total number of PG films:
<xsl:value-of select="$totalNumOfPgFilms"/>
</p>
</xsl:template>
```

Exemples – HTML (1)

Exemple de document XML : contenant "Hello Word"

```
<?xml version="1.0" encoding="UTF-8"?>
<greeting>
    Hello, World!
</greeting>
```

Exemples – HTML (2)

Exemple de fichier XSLT

```
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">
<xsl:output method="html"/>

<xsl:template match="/">
      <xsl:apply-templates select="greeting"/>
</xsl:template>
```

Exemples – HTML (4)

Exemple de fichier XSLT (suite)

```
<xsl:template match="greeting">
    <html>
        <body>
            <h1>
                <xsl:value-of select="."/>
            </h1>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Exemples – HTML (5)

Comment fonctionne la transformation ?

- xsl:output => spécifie le format de sortie (HTML dans notre cas)
- xsl:template => spécifie comment transformer les différentes parties du document XML

Exemples – HTML (6)

Après transformation, on a le résultat suivant:

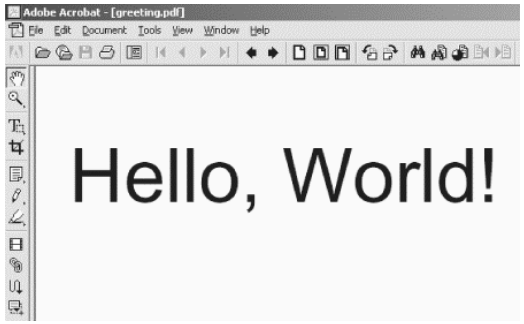
```
<html>
<body>
<h1>
    Hello, World!
</h1>
</body>
</html>
```

Exemples - Autres

L'exemple ci-dessus peut-être transformé dans d'autres format:

- SVG (Scalable Vector Graphics)

- XSL-FO → PDF
- ....



Merci pour votre attention !