XSLT
eXtensible Stylesheet Language Transformations

Sébastien Laborie
Sebastien.Laborie@iutbayonne.univ-pau.fr

Christian Sallaberry
Christian.Sallaberry@univ-pau.fr

Motivations

- On voudrait afficher des données contenues au sein de documents XML dans des pages Web.
- On voudrait appliquer différentes transformations sur un même document XML en récupérant des informations via l'utilisation de technologies XML existantes, telles que XPath.
- On voudrait transformer/adapter une structure de données décrites en XML vers une autre structure de données (décrite en XML ou un autre format).
  - Echanger des informations entre plusieurs applications ou services différents.

XSLT

- Partant d'un document XML, le rôle du langage XSLT consiste à définir des **règles de transformations** qui vont produire un nouveau type de document (XML ou non).

- Un document XSLT est un document XML.
- On appelle souvent un document XSLT **une feuille de style XSLT**.

3

Affecter une feuille de style XSLT à un document XML

- Pour affecter une feuille XSLT à un document XML, il faut la renseigner dans le prologue de ce document. (cf., Cours 1)
- Exemple :

```
<?xml version="1.0"?>
<?xml-stylesheet  type="text/xsl" href="recettes.xsl" ?>
<!DOCTYPE collection SYSTEM "recettes.dtd »>
<collection>
  <recette categorie="plat">
    . . .
  </recette>
 . . .
</collection>
```

4

Structure de base

- Comme tout document XML, une feuille de style XSLT contient un prologue ainsi qu'un élément racine.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <!-- règles de transformation ici -->
</xsl:stylesheet>
```

- L'élément racine est xsl:stylesheet.
- Tout élément du langage XSLT que vous souhaitez utiliser doit commencer par xsl:.
- À ce stade, la feuille XSLT est fonctionnelle. Tous les éléments sont parcourus et les éléments textuels sont écrits.

Contrôler le type de sortie

- L'élément **xsl:output** permet de préciser les caractéristiques de sortie du document à produire.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml | html | text"
               version=""
               encoding=""
               standalone=""
               omit-xml-declaration="yes | no"
               doctype-system=""
               indent="yes | no" />
   ...
</xsl:stylesheet>
```

- Exemple :

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" standalone="no"/>
</xsl:stylesheet>
```

Modèle de transformation

- L'élément xsl:template définit un modèle de transformation à appliquer à un ensemble de nœuds.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="collection">
    . . .
  </xsl:template>
</xsl:stylesheet>
```

- Chaque modèle dispose d'un nom ainsi que d'un jeu de données auquel s'applique le modèle.
- Le modèle de transformation s'applique à l'emplacement du ou des nœuds sélectionnés.

7

Exemple

- Cette feuille de style XSLT :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="collection">
    <html>
      <head></head>
      <body>
        <h1>Mes recettes</h1>
        <h2>Titre</h2>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

- Produit une page Web avec deux titres :
  - Un titre de niveau 1 : « Mes recettes ».
  - Un titre de niveau 2 : « Titre ».
- Dans l'attribut *match*, vous pouvez renseigner une expression XPath.

8

Modularité des modèles

- Il est possible de définir plusieurs modèles de transformation.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="collection"> . . . </xsl:template>
  <xsl:template match="recette"> . . . </xsl:template>
  <xsl:template match="etape"> . . . </xsl:template>
</xsl:stylesheet>
```

- Ceci permet de structurer les différentes transformations à appliquer au sein d'un élément particulier.
- Un modèle peut faire appel à d'autres modèles.

```
<xsl:template match="collection">
    Mes recettes : <xsl:apply-templates select="recette" />
</xsl:template>
```


Récupérer des données

- Il est possible de récupérer des valeurs du document XML source à l'aide de l'élément **xsl:value-of**.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="collection">
     <html>
       <body>
          <h1>Mes recettes</h1>
          <h2><xsl:value-of select="recette/titre" /></h2>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

- Dans notre exemple, on récupère seulement le titre de la première recette ☹

Parcourir un ensemble de données

- Il est possible de parcourir un ensemble de valeurs à l'aide de l'élément **xsl:for-each**.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="collection">
     <html>
       <body>
          <h1>Mes recettes</h1>
          <xsl:for-each select="recette">
              <h2><xsl:value-of select="titre" /></h2>
          </xsl:for-each>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

- Remarquez, que les expressions XPath tiennent compte d'un contexte relatif aux balises englobantes.

Trier un ensemble de données

- Il est possible de trier un ensemble de valeurs à l'aide de l'élément **xsl:sort**.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="/">
     <html>
       <body>
          <h1>Mes recettes</h1>
          <xsl:for-each select="collection/recette">
              <xsl:sort select="titre" order="descending" />
              <h2><xsl:value-of select="titre" /></h2>
          </xsl:for-each>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

- L'élément xsl:sort dispose d'attributs qui permettent de paramétriser le tri.

Structures conditionnelles

- Il est possible d'appliquer différentes transformations en fonction de **conditions**.
- Pour ce faire, il est possible d'utiliser l'élément **xsl:if**.
- Exemple :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="init" match="/">
     <html>
       <body>
          <h1>Mes recettes</h1>
          <xsl:for-each select="collection/recette">
              <xsl:if test="@categorie = 'plat'">
                 <h2><xsl:value-of select="titre" /> (Plat)</h2>
              </xsl:if>
          </xsl:for-each>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

13

Structures conditionnelles

- Des structures conditionnelles plus complètes peuvent aussi être spécifiées.
- Pour ce faire, il est possible d'utiliser l'élément **xsl:choose**. Différentes situations sont ensuite testées.
- Exemple :

```
<xsl:for-each select="collection/recette">
    <xsl:choose>
        <xsl:when test="@categorie = 'plat'">
            <h2><xsl:value-of select="titre" /> (Plat)</h2>
        </xsl:when>
        <xsl:when test="@categorie = 'entree'"> . . . </xsl:when>
        <xsl:otherwise> . . . </xsl:otherwise>
    </xsl:choose>
</xsl:for-each>
```

14

Les variables

- Il est possible de stocker des valeurs au sein de variables. Pour ce faire, il est nécessaire d'utiliser l'élément **xsl:variable** en spécifiant un nom de variable et une donnée associée à cette variable.
- Exemple :

```
<xsl:variable name="var1" select="/collection/recette[2]" />
```

- Vous pouvez faire référence à votre variable dans la feuille XSLT.

```
<xsl:value-of select="$var1/titre" />
```

- ou

```
<xsl:variable name="var2" select="/collection/recette[2]/@categorie" />
…
<xsl:value-of select="/collection/categories/categorie[@id = $var2]" />
```

15

Question

Afficher les détails des recettes

16

Exercices !

Créez vos feuilles XSLT (afficher les infos de l'exercice 5 du cours 1)