Module M2105
Web dynamique
Partie x : XPATH - XSLT

IUT de Béziers, dépt. R&T © 2014-2021
http://www.borelly.net/
Christophe.BORELLY@umontpellier.fr

Généralités

- XSLT (eXtensible Stylesheet Language Transformations) permet de transformer un document XML

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="xxx.xsl"?>
...
```

- Formats de sortie possibles : XML, HTML, XHTML, PDF…
- Fonctionnalités qui se recoupent avec **XQuery** (plutôt orienté bases de données)
- XSLT se sert de **XPath** pour récupérer les données et faire des opérations.
- XSLT utilise le NameSpace : **xsl**

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

XPath

- Permet de sélectionner des éléments ou attributs de l'arbre XML
- Notation semblable à un chemin dans un système de fichiers
  - ***/*** est la racine, **.** le nœud courant, **..** le nœud parent
  - ***//*** pour les descendants, @ pour les attributs
- Chemins absolus ou relatifs

Appellations des nœuds XPath

- Racine
- Parent
- Enfant (Children)
- Enfants de mêmes parents (Siblings)
- Ancêtres (Ancestors)
- Descendants

- Exemple :

```
<?xml version="1.0"
      encoding="utf-8"?>
<bookstore>
 <book>
  <title lang="fr">Fichiers XML</title>
  <title lang="en">XML files</title>
  <price>20</price>
 </book>
 <book>
  <title lang="en">XSLT</title>
  <price>25</price>
 </book>
</bookstore>
```

Exemples de chemins

- `/bookstore/book` : Chemin absolu des éléments **book**
- `/bookstore/book/title/@lang` : Chemin absolu des attributs **lang**
- `//book` : tous les éléments **book** (où qu'ils soient dans le document)
- `//@lang` : tous les attributs **lang** (où qu'ils soient dans le document)
- `/bookstore//price` : les éléments **price** descendant de bookstore

Prédicats

- Recherches conditionnelles
- Notation entre crochets
- Opérateurs : +, -, *, div, mod, =, !=, <, <=, >, >=, or, and
- `/bookstore/book[1]` : le premier élément **book**
- `/bookstore/book[last()]` : le dernier élément **book**
- `/bookstore/book[position()<3]` : les 2 premiers éléments **book**
- `//title[@lang]` : les éléments **titre** possédant l'attribut **lang**
- `//title[@lang='fr']` : les éléments **titre** possédant l'attribut **lang** ayant pour valeur « fr »
- `/bookstore/book[price>20.00]/title` : les éléments **titre** dont les parents ont un enfant **price** supérieur à 20.

Autres syntaxes

- ***** : correspond à n'importe quel élément
- **@*** : correspond à n'importe quel attribut
- **node()** : correspond à n'importe quel nœud
- **|** : permet de sélectionner plusieurs chemins
- **{ }** : permet d'évaluer une expression

Les axes

- Permet de sélectionner plusieurs nœuds par rapport au nœud courant

- ancestor
- attribute
- child
- descendant
- following
- following-sibling
- namespace
- parent
- preceding
- preceding-sibling
- self

Exemples

- `attribute::lang` : l'attribut **lang** du nœud courant
- `child::*` : les éléments enfants du nœud courant
- `child::text()` : les éléments enfant de type texte du nœud courant
- `descendant::book` : les descendants **book** du nœud courant
- `ancestor::book` : les ancêtres **book** du nœud courant
- `child::*/child::price` : les grand enfants **price** du nœud courant

Fonctions XPath 1.0

- last(), position(), count(), id(), name(), local-name(), namespace-uri()
- string(), concat(), starts-with(), contains(), substring(), substring-before(), substring-after(), string-length(), translate()
- boolean(), true(), false(), not()
- number(), sum(), ceiling(), floor(), round()

Fonctions XPath 2.0

- Pas encore supportées nativement par les navigateurs (IE, Firefox, Chrome, ...)
- Numériques :

  `abs``(num)`, **`avg`**`(num,num,...)`, **`max`**`(num,num,...)`, **`min`**`(num,num,...)`, ...
- Chaînes :

  `compare``(comp1,comp2)`, **`normalize-space`**`(string)`,

  `upper-case``(string)`, **`lower-case`**`(string)`,

  `replace``(string,pattern,replace)`, ...
- Autres (dates/erreurs/séquences/...) : voir documentation

Principales directives XSLT

- <xsl:value-of>
- <xsl:template>
- <xsl:apply-templates>
- <xsl:for-each>
- <xsl:sort>
- <xsl:variable>
- <xsl:copy>
- <xsl:copy-of>
- <xsl:if>
- <xsl:choose>
- <xsl:element>
- <xsl:attribute>
- <xsl:text>
- <xsl:comment>
- <xsl:processing-instruction>
- ...

<xsl:value-of>

- Permet de sélectionner une valeur (la première) à partir d'un chemin XPath

```
<xsl:value-of
select="/bookstore/book[1]/title"/>
```

```
<xsl:value-of
select="/bookstore/book[2]/title/@lang"/>
```

- Valeur du nœud courant

```
<xsl:value-of select="."/>
```

Les règles

- **`<xsl:template>`** :
  - Règle s’appliquant à un motif XPath donné
  - Le motif est indiqué dans l'attribut **match**
- **`<xsl:apply-templates>`** :
  - Applique une règle aux enfants du nœud courant ou aux enfants sélectionnés dans l'attribut **select**

Exemple de règles

```
<xsl:template match="/">
 <html>
 <body>
  <h2>Liste de livres</h2>
  <xsl:apply-templates/>
 </body>
 </html>
</xsl:template>

<!-- Si on a un élément de
type texte, on ne fait
rien -->
<xsl:template
     match="text()">
</xsl:template>
```

```
<xsl:template match="book">
 <p>
   <xsl:apply-templates
        select="title"/>
 </p>
</xsl:template>

<xsl:template match="title">
 Titre : <xsl:value-of
              select="."/>
</xsl:template>
```

Les boucles

- On peut faire une boucle sur plusieurs valeurs avec la directive **<xsl:for-each>**
- On peut trier les valeurs en ajoutant la balise **<xsl:sort>** et en option les attributs order (ascending, descending) et data-type (text, number, qname)

```
<xsl:for-each select="//book">
 <xsl:sort select="title"
           order="ascending"/>
 …
</xsl:for-each>
```

Les variables

- On peut définir des variables globales ou locales (dans des règles)
- Une fois définie, on ne peut plus modifier une variable

```
<xsl:variable name="unite" select="' euros'"/>
<xsl:variable name="p1"
              select="/bookstore/book[1]/price"/>
<xsl:variable name="prix1" select="concat($p1,$unite)"/>
<xsl:variable name="header">
  <tr>
  <th>Titre</th>
  <th>prix</th>
  </tr>
</xsl:variable>
```

Les paramètres

- Permet de créer des paramètres globaux (idem variables) ou bien applicables à certaines règles :

```
<xsl:call-template name="show_title">
 <xsl:with-param name="title"
                 select="title"/>
</xsl:call-template>
...
<xsl:template name="show_title" match="/">
  <xsl:param name="title" />
  ... select="$title" ...
</xsl:template>
```

Copie de données

- **<xsl:copy>** permet de copier l'élément courant sans les enfants et attributs (cependant l'attribut **use-attribute-sets** permet, s'il est utilisé, de copier certains attributs du nœud de départ)
- **<xsl:copy-of>** permet de copier intégralement l'élément courant

<xsl:element>

- Permet de créer statiquement ou dynamiquement un élément
  - Attributs possibles : **namespace**, **use-attribute-sets**

```
<xsl:element name="h1">
...
</xsl:element>
<xsl:element name="{$var}">
...
</xsl:element>
```

<xsl:attribute>

- Permet de créer statiquement ou dynamiquement un attribut
  - Attributs possibles : **namespace**

```
<xsl:attribute name="style">
...
</xsl:attribute>
<xsl:attribute name="{$var}">
...
</xsl:attribute>
```

<xsl:attribute-set>

- Permet de créer une liste d'attributs que l'on pourra utiliser avec l'attribut **use-attribute-sets** (voir xsl:copy ou xsl:element)

```
<xsl:attribute-set name="style1">
  <xsl:attribute name="font-size">12pt</xsl:attribute>
  <xsl:attribute name="font-color">red</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

Instructions conditionnelles

- Condition simple :

```
<xsl:if test="price>20">
…
</xsl:if>
```

- Conditions multiples :

```
<xsl:choose>
  <xsl:when test="expression1">...</xsl:when>
  <xsl:when test="expression2">...</xsl:when>
  …
  <xsl:otherwise>...</xsl:otherwise>
</xsl:choose>
```

Texte et commentaires

```
<!-- Conversion par défaut de > en &gt; -->
<xsl:text>40 > 30</xsl:text>
<!-- Pas de conversion dans ce cas -->
<xsl:text disable-output-escaping="yes">
 40 > 30
</xsl:text>

<xsl:comment>
  Ceci est un commentaire
</xsl:comment>
```

Processing instruction

- La directive suivante :

```
<xsl:processing-instruction
      name="xml-stylesheet">
href="style.css" type="text/css"
</xsl:processing-instruction>
```

- Produit le tag :

```
<?xml-stylesheet href="style.css"
type="text/css"?>
```

Références

- http://www.w3.org/TR/xslt-30/
- http://www.w3schools.com/xsl/
- http://www.w3schools.com/xpath/