

Programmation XSLT

Programmation avec XSLT

Combinaison de deux modes de programmation :

- ⑥ **Déclaratif** : on donne les règles, et le processeur fait le reste.
- ⑥ **Impératif** : on utilise les structures habituelles d'un langage de programmation (tests, boucles, variables)

Pas de variable modifiable $\Rightarrow$ il faut parfois programmer par récursion.

Exemple de référence

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<COURS CODE="TC234">
  <SUJET>Publication XSLT</SUJET>
  <ENSEIGNANTS>
    <!-- Enseignant responsable -->
    <NOM>Amann</NOM>
    <NOM>Rigaux</NOM>
  </ENSEIGNANTS>
  <PROGRAMME>
    <SEANCE ID="1">Documents XML</SEANCE>
    <SEANCE ID="2">Programmation XSLT</SEANCE>
    <ANNEE>2003</ANNEE>
  </PROGRAMME>
</COURS>
```

L'instruction `xsl:call-template`

On peut « nommer » des règles et les appeler par leur nom avec `xsl:call-template`.

- S'apparente à un appel de fonction.
- Contrairement à `xsl:apply-templates`, un appel ne change pas le contexte.

On peut passer des paramètres avec `xsl:param`

Une règle d'affichage de nœud

```
<xsl:template name="Afficher">
  <xsl:value-of select="position()"/> :
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="NOM">
  <xsl:call-template name="Afficher"/>
</xsl:template>

<xsl:template match="text(">
  <xsl:call-template name="Afficher"/>
</xsl:template>

<xsl:template match="comment(">
  <xsl:call-template name="Afficher"/>
</xsl:template>
```

Application à <ENSEIGNANTS>

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--      Application de la règle ENSEIGNANTS
1  :
2  :   Enseignant responsable
3  :
4  :   Amann
5  :
6  :   Rigaux
7  :
```

Passage de paramètres

On peut passer des paramètres à `xsl:call-template` ou `xsl:apply-templates`.

- Avec `xsl:param`, on définit dans la règle les paramètres attendus.
- On associe un ou plusieurs `xsl:with-param` à `xsl:call-template` ou `xsl:apply-templates`.

Exemple de paramètre

La règle Afficher attend une chaîne.

```
<xsl:template name="Afficher">
  <xsl:param name="texte"
              select="string('inconnu')"/>
  <xsl:value-of select="concat(position(),
                              ' : ', $texte)"/>
</xsl:template>
```

- 🔴 texte est le nom du paramètre
- 🔴 select donne la valeur par défaut
- 🔴 \$texte désigne le paramètre

Passage du paramètre

C'est l'instruction `xsl:with-param`, à placer après `xsl:call-template`

```
<xsl:template match="NOM">
  <xsl:call-template name="Afficher">
    <xsl:with-param name="texte" select="."/>
  </xsl:call-template>
</xsl:template>

<xsl:template match="comment(">
  <xsl:call-template name="Afficher">
    <xsl:with-param name="texte"
      select="string('Commentaire')"/>
  </xsl:call-template>
</xsl:template>
```

Niveaux de définition

On peut définir des paramètres à deux niveaux :

- **Dans le corps d'une règle.** Le paramètre est alors local à la règle, et fourni par l'appel de règle (`call` ou `apply`).
- **Élément de premier niveau** (fils de `xsl:stylesheet`). Le paramètre est global, et fourni par le processeur

Exemple : moteur de recherche

```
<?xml:
<xsl:param name="titre"/>
<xsl:param name="seance"/>
<xsl:param name="ville"/>
<xsl:template match="MOTEUR">
  <xsl:for-each select="CINEMA">
    <xsl:if test="
      ($titre = " or TITRE = $titre)
      and ($seance = " or HEURE >= $seance
      and ($ville = " or VILLE = $ville)">
      <xsl:apply-templates select="." /><p/>
    </xsl:if>
  </xsl:for-each>
</xsl:template>
```

Comment faire avec XSQL

On inclut les paramètres HTTP dans le document XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="MoteurCherche.xsl" ty
<page xmlns:xsql="urn:oracle-xsql">
  <xsql:include-request-params/>
  <xsql:include-xml href="Moteur.xml"/>
</page>
```

Instructions de contrôle

Structure de test: `xsl:if`

L'attribut `test` = prédicat XPath

Le contenu s'instancie si `test` s'évalue à `true`.

```
<xsl:template match="FILM">
  <xsl:if test="ANNEE < 1970">
    <xsl:copy-of select="."/>
  </xsl:if>
</xsl:template>
```

Pas de `else` !

Structure de test : `xsl:choose`

Test à choix multiples : équivalent au case du C

```
<xsl:choose>
  <xsl:when test='expression1'>
    Corps de règle 1
  </xsl:when>
  <xsl:when test='expression2'>
    Corps de règle 2
  </xsl:when>
  <xsl:otherwise>
    Corps de règle par défaut
  </xsl:otherwise>
</xsl:choose>
```

Exemple de `xsl:choose`

```
<xsl:template match="FILM">
  <xsl:choose>
    <xsl:when test="ANNEE < 1960">
      "<xsl:value-of select="TITRE"/>"
      est ancien
    </xsl:when>
    <xsl:when test="ANNEE >= 1960">
      "<xsl:value-of select="TITRE"/>"
      est récent
    </xsl:when>
    <xsl:otherwise>
      De quand date
      "<xsl:value-of select="TITRE"/>" ?
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```


Boucles avec `xsl:for-each`

Boucles avec XSLT :

- S'effectue toujours sur un ensemble de nœuds sélectionnés avec `select`
- Prend les nœuds un par un comme nœud contexte
- Le contenu de `xsl:for-each` est instancié pour chacun des nœuds contexte.

Pas de variable, donc pas d'incrémentement.

Exemple d'un xsl:for-each

```
<xsl:template match="PROGRAMME">
  Programme <xsl:value-of
    select="concat(position(),
                  '/' , last())"/> :
  <SEANCES>
    <xsl:for-each select="SEANCE">
      Séance <xsl:value-of
        select="concat(position(),
                      '/' , last())"/> :
        <xsl:value-of select="." />
    </xsl:for-each>
  </SEANCES>

  Valable pour l'année
  <xsl:value-of select="ANNEE"/>
</xsl:template>
```

Évaluation du `xsl:for-each`

Attention au changement de contexte !

- Dans l'ensemble du corps du programme, le nœud contexte est `<PROGRAMME>`
- Dans le `xsl:for-each`, le contexte comprend :
 - une des séances sélectionnées
 - le nombre de séances (fonction `last()`)
 - la position de la séance courante (fonction `position()`)

Résultat du `xsl:for-each`

Noter le résultat des fonctions *last()* et *position()*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PROGRAMME>
  Programme 1/1 :
  <SEANCES>
    Séance 1/2 : Documents XML
    Séance 2/2 : Programmation XSLT
  </SEANCES>
  Valable pour l'année 2003
</PROGRAMME>
```

Variables XSLT

Variable = nom + valeur. Plusieurs possibilités :

- Attribut `select` avec la valeur (une constante)
- Attribut `select` avec une expression XPath
⇒ arbre temporaire auquel on peut appliquer une expression XPath.
- Le contenu de l'élément `xsl:variable` est la valeur de la variable

Quelques exemples

- `<xsl:variable name='var1' select='12' />`
- `<xsl:variable name='var2'
select='/COURS/ENSEIGNANTS' />`
- Variable définie par son contenu: `<xsl:variable
name='var3'>`
Ceci est un
`<mot-cle>contenu</mot-cle>` de variable
`</xsl:variable>`

Portée d'une variable

Deux types de variables :

- **Variables globales** (éléments de premier niveau). Elles sont visibles dans tout le programme XSLT.
- **Variables locales** (dans les corps de règle). Visibles dans les following-sibling et leurs descendants

Chevauchement des portées interdit, sauf pour une variable globale et une variable locale.

Exemple de variable globale

```
<xsl:variable name="annee" select="1970"/>
<xsl:template match="FILM">
  <xsl:choose>
    <xsl:when test="ANNEE < $annee">
      "<xsl:value-of select="TITRE"/>"
      est ancien
    </xsl:when>
    <xsl:when test="ANNEE >= $annee">
      "<xsl:value-of select="TITRE"/>"
      est récent
    </xsl:when>
  </xsl:choose>
</xsl:template>
```


Exemple de variable locale

```
<xsl:template match="PROGRAMME">
  <SEANCES>
    <xsl:variable name="phrase">
      (valable pour l'an
        <xsl:value-of select="ANNEE"/>)
    </xsl:variable>
    <xsl:for-each select="SEANCE">
      <xsl:value-of select="concat(., $phrase)" />
    </xsl:for-each>
  </SEANCES>
</xsl:template>
```

Tri avec `xsl:sort`

Tri des nœuds traités par `xsl:for-each` ou `xsl:apply-templates` avec `xsl:sort`.

- Un critère de tri est une expression XPath
- Après la balise ouvrante de `xsl:for-each` ou `xsl:apply-templates`
- Il peut avoir plusieurs critères qui définissent les différents niveaux de tri.

Par défaut les nœuds sont triés sur leur position.

Exemple de tri

Tri des films d'abord sur l'année, puis sur le titre.

```
<xsl:template match="FILMS">
  <FILMSTRIES>
    <xsl:apply-templates select="FILM">
      <xsl:sort select="ANNEE"/>
      <xsl:sort select="TITRE"/>
    </xsl:apply-templates>
  </FILMSTRIES>
</xsl:template>
```