Introduction à XSLT

L3Pro BDISE – XML

Mathieu Sassolas

IUT de Sénart Fontainebleau
Département Informatique

Année 2016-2017
Cours 3

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL
IUT SENART FONTAINEBLEAU

Plan de la séance

XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Le préambule
TD/TP

1. Faiblesse de CSS et besoin de vraies transformations
2. Sélection et extraction de données avec XSLT
   - Définir des templates
   - Flot de contrôle et parcours de l'arbre
3. Finissons par le préambule : autour des `<xsl:templates/>`
4. Mise en application

Limites de CSS

XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Le préambule
TD/TP

Cadre

On a une liste de `Course`, puis un liste de `Professor` et `Lecturer` mélangés :
`(<!ELEMENT Department (Course+,Instructors))`
`(<!ELEMENT Instructors (Professor|Lecturer)+)`

Quelques choses impossibles à faire en CSS :

- Mettre en gras la liste des `Instructors` seulement si elle contient au moins un `Professor`.
- Mettre la liste des enseignants avant la liste des cours.

Remarque

CSS3 tente déjà d'offrir un peu plus de possibilités à ce niveau là : sélection du premier `Lecturer` dans la liste, du dernier élément de la liste...

Transformations

XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Le préambule
TD/TP

- De XML vers XHTML :
  - Mise en forme de contenu à la structure *ad hoc* (via une DTD privée) vers du contenu en XHTML, lisible par tous les navigateurs (DTD publique).
  - Typiquement : le XML est le résultat d'une requête sur une base, il faut fournir une page présentant ce résultat.
- De XML vers XML :
  - Pour conformer le résultat d'une requête à une autre DTD.
  - Pour affiner le résultat d'une requête.

XSLT, c'est quoi ?

XSLT

eXtensible Stylesheet Language Transformation
Langage extensible de transformation de feuille de style

- XSL est un système pour créer des feuilles de style, mais celles-ci peuvent aussi bien produire du HTML que du PDF.
- XSLT n'est que la partie transformation structurelle de XSL.
- Manipulation de modèles (« templates ») : on remplace un élément par un autre selon ce modèle.

↪ On peut voir ça comme des transformations d'arbres.

Aujourd'hui

Sans ordre particulier

- Réécriture simple d'arbres XML (sélection/extraction de données).
- Création de templates.
- Le parcours de l'arbre par le processeur XSLT.

Templates

- Règles de réécriture de parties de l'arbre.
- On spécifie sur quel types de nœud cette règle s'applique :
  - Exemple `match="Lecturer"`.
  - On peut spécifier le type de nœud par sa position relative dans l'arbre : `match="Instructor/Lecturer"`.
  - On peut spécifier le type de nœud par sa position absolue dans l'arbre : `match="/Department/Instructor/Lecturer"`.
  - On peut faire plus compliqué à l'aide de XPath.

Syntaxe

```
<xsl:template match="type_de_noeud">
    ...
</xsl template>
```


XPath : version simplifiée

XPath sert à répérer les nœuds et attributs dans l'arbre.

- `Crao/Rahan` : nœuds `Rahan` fils de `Crao` (chemin relatif).
- `//neutre` : tous les éléments `neutre`.
- `/monarchie` : élément `monarchie`, référencé par son chemin absolu.
- `@du_sujet` : attribut `du_sujet`.
- `.` : élément (ou attribut) courant.
- `..` : élément père.

Plus de détails sur XPath la séance prochaine.

Conflit dans les templates

En cas de conflit

Si plusieurs règles s'appliquent, il y a un système de priorités.

Les principes généraux en sont :

- Les règles du fichier principal l'emportent sur des fichiers importés.
- Les règles les plus spécifiques l'emportent sur les règles générales :

  `match="Instructor/Lecturer"` > `match="Lecturer"`
- En cas d'égalité de priorité : la dernière règle écrite prévaut.

Attention !

Ces situations sont à éviter autant que possible !

Contenu des templates
Partie I

- Du texte (sorti tel quel).
- Des éléments XML (ou XSL...)

Syntaxe

```
<xsl:element name="nom_element">
    <xsl:attribute name="nom_attribut">
      valeur_attribut
    </xsl:attribute>
    Contenu
</xsl:element>
```

⇝ Peut être mis en texte directement :

```
<nom_element nom_attribut="valeur_attribut">
    Contenu
</nom_element>
```

mais on ne pourrait pas générer du XSLT en XSLT 😄

Contenu des templates
Partie I

- Du texte (sorti tel quel).
- Des éléments XML (ou XSL...)

Syntaxe

```
<xsl:element name="nom_element">
    <xsl:attribute name="nom_attribut">
      valeur_attribut
    </xsl:attribute>
    Contenu
</xsl:element>
```

```
<xsl:element name="xsl:template">
<xsl:attribute name="match">Lecturer</xsl:attribute>
...
</xsl:element>
```

mais on ne pourrait pas générer du XSLT en XSLT 😄

Contenu des templates
Partie II

- La valeur d'éléments ou d'attributs :

Syntaxe

```
<xsl:value-of select="nom_element"/>
<xsl:value-of select="@nom_attribut"/>
```

NB : l'attribut ou l'élément peut être plus complexe. (*Cf* XPath.)

Flot de contrôle (local)
Partie I

- Boucles :

Syntaxe

```
<xsl:for-each select="element">...</xsl:for-each>
```

- Test :

Syntaxe

```
<xsl:if test="expression">...</xsl:if>
```

- Ce n'est pas vraiment une condition : pas de `else`
- Exemple d'`expression` : `@attribut='maValeur'`, `@attribut > 'maValeur'`.
- Remarquer l'utilisation de version échappées des chevrons `>` ⇝ `>`.

Flot de contrôle (local)
Partie II

- Conditions :

Syntaxe

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

- Messages d'avertissement ou d'erreur

Syntaxe

```
<xsl:message terminate="yes|no">
  Message d'erreur
</xsl:message>
```

↪ toujours utile dans un `xsl:otherwise` qui ne devrait jamais être utilisé.

Se promener dans l'arbre XML de départ

- Initialement : on se trouve à la racine.
- On peut demander l'application des templates sur ses fils :`<xsl:apply-templates/>`.

Exemple

```
<div>Avant de traiter le contenu des sous arbres.
<xsl:apply-templates/>
Les sous arbres ont été traités</div>.
```

- Le traitement peut être sélectif :

Exemple

```
<div>Je ne veux que les professeurs:
<xsl:apply-templates select="Professor"/>
Et maintenant les maîtres de conférence:
<xsl:apply-templates select="Lecturer"/></div>
```


Remarques sur `xsl:apply-templates`

- `<xsl:apply-templates/>` change la position courante dans l'arbre au nœud désigné par le modèle : on passe successivement à tous les fils (ou les nœuds spécifiés par le `select`).
- Dans les feuilles XSLT, il y a toujours implicitement

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

- Le processeur XSLT commence par faire comme s'il lisait `<xsl:apply-templates match="/"/>`.
- Un autre moyen de naviguer dans l'arbre : la boucle `for-each`.

Modes de templates

- On peut utiliser des modes pour spécifier de quelle manière traiter un élément.

Exemple

```
<xsl:apply-templates select="elt"
                              mode="block_display"/>
...
<xsl:template match="//elt" mode="block_display">
  <div>...</div>
</xsl:template>
<xsl:template match="//elt" mode="inline_display">
  <span>...</span>
</xsl:template>
```

↪ Les templates sont (par défaut) appliqué aux éléments fils (ici seulement de type elt).

Appel de templates

- On peut appeler un template spécifique s'il est nommé :

Exemple

```
<xsl:call-template name="maTemplate"/>
...
<xsl:template name="maTemplate">...</xsl:template>
```

↪ L'appel ne s'applique que à l'élément courant : pas de changement de nœud !

Exemple de flot de contrôle
Le XSLT

```
...
<xsl:template match="/document">
  <xsl:for-each select="preambule">
    <div>Un nouveau prologue:
    <xsl:value-of select="@longueur"/> caractères.
    <xsl:call-template name="inline_pre"/></div>
  </xsl:for-each>
  <xsl:apply-templates select="contenu"/>
</xsl:template>
<xsl:template match="//contenu">
  <div><h3><xsl:value-of select="."/></h3></div>
</xsl:template>
<xsl:template name="inline_pre">
  <span><xsl:value-of select="."/></span>
</xsl:template>
...
```


Exemple de flot de contrôle
Le XML

Exemple de XML

```
<document>
  <preambule longueur="42">Bla bla bla</preambule>
  <preambule longueur="24">Bli bli bli</preambule>
  <contenu>Bla bli blo</contenu>
  <contenu>Blu ble bly</contenu>
  <contenu>Bly blu bla</contenu>
  <contenu>Blo bli ble</contenu>
</document>
```

Exemple de flot de contrôle
Le résultat

Le HTML obtenu

```
<div>Un nouveau prologue:
   42 caract&egrave;res.
   <span>Bla bla bla</span></div>
<div>Un nouveau prologue:
   24 caract&egrave;res.
   <span>Bli bli bli</span></div>
<div><h3>Bla bli blo</h3></div>
<div><h3>Blu ble bly</h3></div>
<div><h3>Bly blu bla</h3></div>
<div><h3>Blo bli ble</h3></div>
```


IUT SENART FONTAINEBLEAU
XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Templates
Parcours de l'arbre
Le préambule
TD/TP

Exemple de flot de contrôle
Le résultat

Le HTML obtenu

Un nouveau prologue: 42 caractères. Bla bla bla
Un nouveau prologue: 24 caractères. Bli bli bli

Bla bli blo

Blu ble bly

Bly blu bla

Blo bli ble

```
<div><h3>Blo bli ble</h3></div>
```


IUT SENART FONTAINEBLEAU
XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Le préambule
TD/TP

Structure globale

xsl:stylesheet

```
<?xml version="1.0" encoding="utf-8" ?>

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--  Le contenu de la feuille: typiquement
      des <xsl-templates/>                    -->
</xsl:stylesheet>
```

- Un préambule concernant le type de XML et l'encodage
- Racine `xsl:stylesheet`.
- De manière totalement équivalente `xsl:transform`.

IUT SENART FONTAINEBLEAU
XSLT
M. Sassolas
L3Pro
Cours 3
Introduction
Sélection & extraction
Le préambule
TD/TP

Mode de sortie : `xsl:output`

↪ Définit le type du document produit.

Exemple

```
<xsl:output method="html" version="html 4.01"
      encoding="utf-8"
      doctype-public="-//W3C//DTD HTML 4.01//EN"
      doctype-system=
          "http://www.w3.org/TR/html4/strict.dtd"/>
```

⇝ Ajoutera des tags `meta` dans l'entête, veillera à ce que les balises soient refermées à la mode HTML...

Syntaxe (directement sous la racine)

```
<xsl:output method="html|xml|text" version="..."
                encoding="utf-8"
                doctype-public="identifiant_DTD"
                doctype-system="URL_d_une_DTD"  />
```

Compilation

Utilisation de `xsltproc` :

Commande

```
xsltproc -o fichier_de_sortie  \\
    feuille_de_style_XSLT.xsl document_initial.xml
```

Exemple

```
xsltproc -o trains.html trains.xsl trains.xml
```


Fin du cours 3

↳ C'est l'heure du TD/TP ↵