TP 5 : La publication des documents XML-XSL

Lab. 1: XSL:

1. La publication des documents XML :

L'intranet/Internet a pris une place importante dans les entreprises. Les navigateurs remplacent de plus en plus les applications riches et donnent un point d'accès unique.

Comme nous l'avons vu, le format XML se charge d'agglomérer des données, ce qui facilite leur stockage ainsi que leur circulation. On s'attend donc nécessairement à ce que ces données soient visibles à un moment ou à un autre pour les utilisateurs, que ce soit à titre de contrôle, pour les personnels de l'entreprise, ou bien pour des clients (cadre B2C : Business to consumer). Cette visibilité doit éliminer tout sens technique. Un document XML, même structuré simplement, doit être présentable, c'est-à-dire être qu'il doit être lié à une charte graphique (couleurs, polices de caractères…), contenir des tableaux, des images… En un mot, tout un ensemble qui n'existe pas dans le format initial. Les langages qui autorisent cela restent essentiellement ceux du Web, soit XHTML (Extensible HyperText Markup Language)/HTML et CSS. Mais il existe aussi d'autres formats employés dans l'entreprise comme PDF (Portable Document Format) ou RTF (Rich Text Format)… Par exemple, nous pourrions imaginer que des fiches produit décrites en XML soient disponibles à la fois sur le Web sous forme de page HTML mais également sous un format plus imprimable, comme PDF.

2. L'application des CSS à un document XML

L'application directe des styles CSS à un document XML est possible mais limitée.

A chaque balise "inventée" dans le fichier XML, on va définir un élément de style que le navigateur pourra alors afficher.

Soit l'extrait suivant d'un document XML :

```
<carnet>
<personne>
<nom>Dupont</nom>
<age>44</age>
</personne>
<personne>
<nom>Dupond</nom>
<age>43</age>
</personne>
</carnet>
```

Si nous souhaitons définir un fichier CSS pour présenter ce document de manière rudimentaire, nous pouvons écrire le fichier carnet.css suivant :

```
nom {
display :block ;
color :blue ;
}
age {
color :red ;
}
```

Le lien vers le fichier CSS sera présent via l'instruction de traitement suivante, placée avant la racine du document XML :

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

Mais il y a encore un autre moyen, plus performant et aux possibilités plus étendues : afficher du XML avec le XSL soit le langage de feuilles de style eXtensible. Le pendant du XML au CSS.

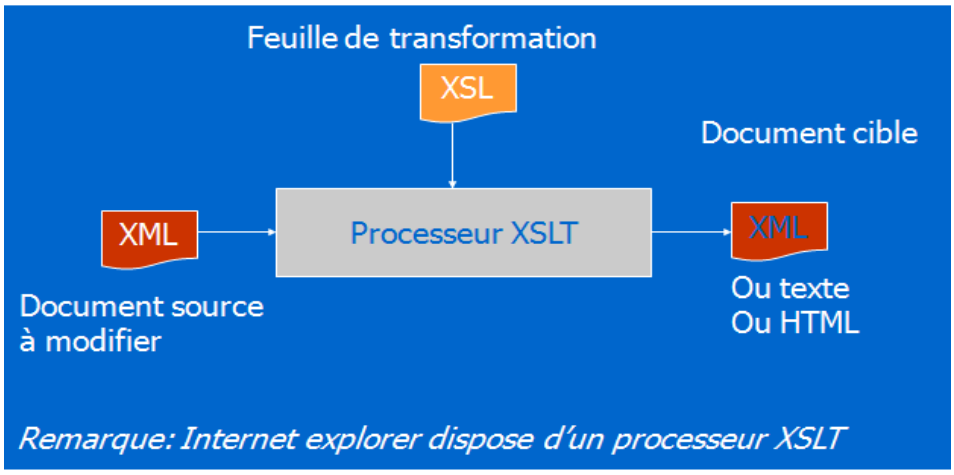
3. L'application de XSL à un document XML

a. XSL :

Comme le XML n'utilise pas des balises prédéfinies (car on peut inventer ses propres balises), le navigateur ne "comprend" pas les balises du XML et ne sais pas trop comment afficher un document XML.

Pour néanmoins afficher des documents XML, il est nécessaire d'avoir un mécanisme pour décrire comment le document pourrait être affiché. Un de ces mécanisme est les feuilles de style classiques du Html (CSS), mais le XSL pour eXtensible Stylesheet Language est de loin un langage de feuille de style plus adapté au XML et donc plus performant.

De façon résumée, le XSL est un langage qui transforme le XML en Html. Mais il fait bien plus !



b. La racine d'un document XSL :

La structure de base d'un document XSL commence par un prologue, puis un élément <xsl:stylesheet pouvant contenir quelques attributs, notamment une déclaration d'espace de noms ainsi que le numéro de version.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> (...)
</xsl:stylesheet>
```

L'élément <xsl:stylesheet> est l'élément racine du document XSL.

Une feuille xslt permet de générer 3 formats : texte, XML ou HTML.

La balise <xsl :output> permet de définir le format de sortie, ainsi il est possible de gérer les caractères accentuées et spéciaux.

Par exemple:

```
<xsl:output method= “html” encoding= “iso-8859-1”>
```

Cette déclaration précise que le document sera au format HTML et que les caractères du fichier XML d’entrée seront encodés selon la norme iso-8859-1

c. Contenu d’un document XSL :

Un document XSL peut être vu comme une succession de template. Un template étant du code XSL qui permet de sélectionner une partie du document XML et de lui apporter une transformation.

L'exemple qui suit contient une portion d'un document XML décrivant un livre.

```
<livre id="03432">
<auteur>Paul Lafargue</auteur>
<titre>Le droit à la paresse</titre>
</livre>
```

Si l'on souhaite présenter ce livre avec le HTML suivant :

```
<html>
<body>
<p><b> Paul Lafargue </b> Le droit à la paresse </p>
</body>
</html>
```

On pourra utiliser le document XSLT suivant :

```
<xsl:stylesheet xmlns:xsl="">
<xsl:template match="livre">
<html>
<body>
<p>
<b>
<xsl:value-of select="auteur"/>
</b>
<xsl:value-of select="titre"/>
</p>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Un template définit les règles de la transformation à réaliser sur un ensemble de nœuds (d'un document XML source à un document résultat). L'attribut match définit le noeud (ou la requête Xpath) à transformer.

d. Les éléments XSL :

Les éléments XSL sont utilisés dans les templates pour réaliser les transformations. Il existe un grand nombre d'éléments, permettant de faire des tests conditionnels (if, switch,etc…), des boucles (for, while, …), etc. Nous présenterons dans cette partie quelques-uns uns des éléments fournis avec XSLT.

Soit le fichier xml suivant :

```
<annuaire type=”pages blanches”>
<entree>
<nom>Paul Lafargue</nom>
<telephone>06 03 02 01 00</telephone>
</entree>
<entree>
<nom>Lorédan Larchey</nom>
<telephone>06 00 01 02 03</telephone>
</entree>
</annuaire>
```

i. Sélection de texte :

<**xsl:value-of** select=”requeteXPath”/>

Cet élément est utilisé pour sélectionner une valeur simple dans un élément du document à transformer et l'écrire dans le document de sortie.

Exemple :

XSLT : <xsl:value-of select=”nom/text()”/>
Contexte : /annuaire/entree[1]
Résultat : Lorédan Larchey

ii. Boucle :

Cet élément permet de parcourir une liste de nœuds et leur appliquer une transformation.

Exemple :

XSLT : <xsl:for-each select=”/annuaire/entree/nom”>
Nom: <xsl:value-of select=”text()”/>
</xsl:for-each>
Contexte : /
Résultat : Nom: Lorédan Larchey
Nom: Paul Lafargue

iii. Test conditionnels :

Cet élément permet d'effectuer un test avant transformation. Le test respecte la syntaxe XPath. Notons qu'il n'existe pas de <xsl:else/>.

Exemple :

XSLT : <xsl:if test="name()='nom'">
<xsl:value-of select="text()"/>
</xsl:if>
Contexte : /annuaire/entree[2]/nom
Résultat : Paul Lafargue

<xsl:choose>, <xsl:when test="expressionXPath"> et <xsl:otherwise>

Ces éléments permettent d'effectuer des tests plus complets qu'avec l'élément <xsl:if>.
C'est l'équivalent d'un switch. <xsl:when> permet de tester si une condition est réalisée.
Dans un élément <xsl:choose>, on a un nombre non limité d'éléments <xsl:when> et un élément <xsl:otherwise>.

Exemple :

XSLT :
<xsl:choose>
<xsl:when test="@type='pages blanches'">
Annuaire de pages blanches
</xsl:when>
<xsl:when test="@type='pages jaunes'">
Annuaire de pages jaunes
</xsl:when>
<xsl:otherwise>
Type d'annuaire inconnu
</xsl:otherwise>
</xsl:choose>

Contexte : /annuaire
Résultat : Annuaire de pages blanches

iv. Autres :

<xsl:sort select="requeteXPath"
data-type="text | number | Qname"
order="ascending | descending"
case-order="upper-first | lower-first"/>

Cet élément permet d'effectuer des tris sur un ensemble de noeuds résultant d'une requête XPath.

Exemple :

XSLT :
<xsl:for-each select="/annuaire/entree">

<xsl:sort select=”nom
data-type=”text”
order=”ascending”/>
<xsl:value-of select=”nom/text()”/>
</xsl:for-each>
Contexte : /
Résultat : Paul Lafargue
Lorédan Larchey

Lab. 2 : Mise en pratique

Exercice 1 :

1. Créer le fichier XML suivant :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<recette>
        <entete>
                <auteur>ZINEB TAHOR</auteur>
                <titre>Recette du jour</titre>
                <remarque>Pour une personne</remarque>
        </entete>
        <procedure>
Remplir un saladier avec de la confiture de fraises, du chocolat râpé,
des bananes écrasées, de la moutarde forte, des saucisses de Toulouse écrasées tièdes mais crues. Mélanger vigoureusement jusqu'à obtenir une bouillie marron-clair. Il est normal qu'il y ait des grumeaux. Les proportions sont environ égales pour tous les ingrédients, mais il est possible de varier selon les goûts de chacun.
        </procedure>
</recette>
```

2. Créer une feuille de style XSL permettant de produire une page HTML qui :

   - a pour titre le contenu de la balise titre ;
   - commence par un titre <h1> ayant comme contenu le contenu de l'élément titre ;
   - donne ensuite le nom de l'auteur de la recette ;
   - affiche ensuite le mot **Remarque :** puis le contenu de l'élément remarque ;
   - affiche **Procédure** en niveau <h2> ;
   - dans un paragraphe, présente la procédure à suivre.

Exercice 1 :

Cet exercice se propose de transformer des documents XML qui décrivent la répartition en modules de contenus de formation.

1) Soit la DTD (description de documents qui présentent une formation) et le document XML (document qui décrit une formation particulière nommé *cci*).

```
<!ELEMENT formation (titre, niveau, responsable, module+) >
<!ATTLIST formation code ID #REQUIRED >
<!ELEMENT titre (#PCDATA) >
<!ELEMENT niveau (#PCDATA) >
<!ELEMENT responsable (prenom, nom)>
<!ATTLIST responsable tel CDATA #IMPLIED
                                          bureau CDATA #REQUIRED>
 <!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT module (titre, horaire, points, programme) >
<!ATTLIST module semestre (1|2) #REQUIRED
                                  repere ID #REQUIRED>
<!ELEMENT horaire (#PCDATA) >
<!ELEMENT points (#PCDATA) >
<!ELEMENT programme (para*) >
<!ELEMENT para (#PCDATA) >
```

```
<formation code="MCCI">
  <titre>Master CCI</titre>
  <niveau>5</niveau>
  <responsable bureau="224">
    <prenom>Brigitte</prenom>
    <nom>Dillon</nom>
  </responsable>
  <module semestre="1" repere="CCI91">
    <titre>Initiation à l'algorithmique, système et architecture</titre>
    <horaire>50</horaire>
    <points>3</points>
    <programme/>
  </module>
  <module semestre="1" repere="CCI92">
    <titre>Systèmes d'information</titre>
    <horaire>100</horaire>
    <points>6</points>
    <programme>
      <para>Modèles structurés</para>
      <para>Bases de données</para>
    </programme>
  </module>
  <module semestre="1" repere="CCI93">
    <titre>Algorithmique et programmation</titre>
    <horaire>100</horaire>
    <points>6</points>
    <programme>
      <para>Algorithmique et programmation impérative</para>
      <para>Programmation et modélisation par objet</para>
    </programme>
  </module>
  <module semestre="1" repere="CCI94">
```

```
    <titre>Systèmes et réseaux</titre>
    <horaire>50</horaire>
    <points>3</points>
    <programme>
      <para>Système Unix</para>
      <para>Architecture des réseaux</para>
    </programme>
  </module>
  <module semestre="2" repere="CCI95">
    <titre>Modèles et documents structurés</titre>
    <horaire>100</horaire>
    <points>6</points>
    <programme>
      <para>Bases de XML</para>
      <para>Applications en XML</para>
    </programme>
  </module>
  <module semestre="2" repere="CCI101">
    <titre>Services du web</titre>
    <horaire>100</horaire>
    <points>6</points>
    <programme>
      <para>Bases des services web</para>
      <para>Applications des services web</para>
    </programme>
  </module>
  <module semestre="2" repere="CCIstage">
    <titre>Stage en entreprise</titre>
    <horaire>560</horaire>
    <points>27</points>
    <programme/>
  </module>

</formation>
```

2) Écrire un programme XSLT formation_lst_a.xsl qui transforme le document cci.xml en une *page HTML* constituée de listes imbriquées, dont l'interprétation par un navigateur donne la présentation suivante : cci.xml.

Master CCI

1. **CCI91**
   - Numéro du semestre : 1
   - Initiation à l'algorithmique, système et architecture
   - 50 heures
   - 3 points ECTS
   - Programme :
2. **CCI92**
   - Numéro du semestre : 1
   - Systèmes d'information
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Modèles structurés
     - Bases de données
3. **CCI93**
   - Numéro du semestre : 1
   - Algorithmique et programmation
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Algorithmique et programmation impérative
     - Programmation et modélisation par objet
4. **CCI94**
   - Numéro du semestre : 1
   - Systèmes et réseaux
   - 50 heures
   - 3 points ECTS
   - Programme :
     - Système Unix
     - Architecture des réseaux
5. **CCI95**
   - Numéro du semestre : 2
   - Modèles et documents structurés
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Bases de XML
     - Applications en XML
6. **CCI101**
   - Numéro du semestre : 2
   - Services du web
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Bases des services web
     - Applications des services web
7. **CCIstage**
   - Numéro du semestre : 2
   - Stage en entreprise
   - 560 heures
   - 27 points ECTS
   - Programme :

4) Écrire une deuxième version du programme XSLT formation_lst_b.xsl pour que le mot *Programme* n'apparaisse que si le module est effectivement accompagné d'un texte décrivant le programme

Master CCI

1. **CCI91**
   - semestre : 1
   - Initiation à l'algorithmique, système et architecture
   - 50 heures
   - 3 points ECTS
2. **CCI92**
   - semestre : 1
   - Systèmes d'information
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Modèles structurés
     - Bases de données
3. **CCI93**
   - semestre : 1
   - Algorithmique et programmation
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Algorithmique et programmation impérative
     - Programmation et modélisation par objet
4. **CCI94**
   - semestre : 1
   - Systèmes et réseaux
   - 50 heures
   - 3 points ECTS
   - Programme :
     - Système Unix
     - Architecture des réseaux
5. **CCI95**
   - semestre : 2
   - Modèles et documents structurés
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Bases de XML
     - Applications en XML
6. **CCI101**
   - semestre : 2
   - Services du web
   - 100 heures
   - 6 points ECTS
   - Programme :
     - Bases des services web
     - Applications des services web
7. **CCIstage**
   - semestre : 2

6) Écrire un programme XSLT formation_tab.xsl qui transforme le document cci.xml en une page HTML constituée d'un tableau qui récapitule les modules.

Master CCI

Repère	Titre	Horaire	ECTS
CCI91	Initiation à l'algorithmique, système et architecture	50	3
CCI92	Systèmes d'information	100	6
CCI93	Algorithmique et programmation	100	6
CCI94	Systèmes et réseaux	50	3
CCI95	Modèles et documents structurés	100	6
CCI101	Services du web	100	6
CCIstage	Stage en entreprise	560	27