Introduction technique à XSLT + XPath

Code: xml-xslt

Originaux

url: http://tecfa.unige.ch/guides/tie/html/xml-xslt/xml-xslt.html
url: http://tecfa.unige.ch/guides/tie/pdf/files/xml-xslt.pdf

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 1.5 (modifé le 27/11/03 par DKS)

Prérequis

- XML de base

Module technique précédent: xml-dom (éléments du XML Framework)
Module technique précédent: xml-tech (arbres XML, DTDs)

Modules suivants

Module technique suivant: xml-ser (server-side XML avec XSLT, à refaire !)

Objectifs

- Avoir une idée de XSLT, savoir écrire des simples feuilles de transformation XML vers (X)HTML

Notes:

- Il reste pas mal d'améliorations à faire !!
- Il reste des lacunes ....

1. Table des matières détaillée

2. Introduction à XSLT

But de XSLT

- XSLT est un langage de transformation d'arbres (fichiers) XML
- XSLT est (bien sûr) écrit en XML
- XSLT permet la génération d'autres contenus à partir d'un fichier XML, par exemple:
  - du HTML (bien formé)
  - du XML plus compliqué (tables de matière + règles de formattage XSL/FO)
  - des extraits en XML ou HTML ....

Utilisation (mécanisme de base):

1. Définir des règles qui disent comment transformer un "noeud" (element, tag) et ses sous-éléments
2. Organiser l'application de ces règles

Compléménts à XSLT

- Xpath (langage pour indiquer un chemin dans un arbre XML, expliqué plus loin)
- XSL/FO (mise en page)

Les feuilles de style XSLT

- Une feuille de style XSLT est un document séparée qui contient des règles de transformation XSLT
- On peut associer une feuille de style XSL(T) à un (ou plusieurs) documents XML
- L'association peut se faire au niveau serveur Web ou au niveau client (IE Explorer 5.5)

Documentation

- La spécification de XSLT est formalisée (W3C Recommendation 16/11/99)
  url: http://www.w3.org/TR/xslt
- A Tecfa: voir la page XML dans la toolbox

2.1 Entêtes des fichiers XSL

A. Définition d'un fichier XSL

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

- Normalement (à TECFA en tout cas) les fichiers XSLT s'appellent *.xsl
- `xmlns:xsl="URL"` définit un "namespace", en gros la définition formelle des balises XSL, qui commencent tous par `"xsl:"`
- Le fait que toutes les balises XSL commencent par `xsl:` empêche toute confusion

B. Association d'un fichier XSL à un fichier XML

L'association peut se faire dans le fichier XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="project.xsl" type="text/xsl"?>
```

- On présuppose ici que soit le serveur (par exemple Cocoon) soit le client (par exemple IE5/6 ou Mozilla 1.x) comprenne cette instruction
- Il existe d'autres méthodes d'association (server-side, comme un "GET Url" avec des paramètres) ou encore un traitement "batch".

2.2 Principe de fonctionnement de XSL

- La transformation du document source (XML) se fait selon des règles (facteurs conditionnels)
- Une feuille de style XSL contient un jeu de règles qui déclarent comment traduire des éléments XML (selon leur contexte).
- Une règle XSL traduit par exemple un "tag" xml en au moins un "tag" html, mais souvent en plusieurs, par exemple:

  ```
  <commentaire> xxxx </commentaire>
  ```

  pourrait donner:

  ```
  <DL>
       <DT>Commentaire  </DT>
       <DD> xxxx </DD>
   </DL>
  ```
- Une feuille de style doit définir une transformation pour chaque tag XML
- L'organisation de la règle qui traite la racine XML est cruciale:
  - Elle définit `<html><head>.....</head><body></body></html>`

2.3 Un simple exemple XSLT

Exemple 2-1: Sensibilisation XML + XSLT

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.dtd **(DTD)**
url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.xml.text **(src XML)**
url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page-html.xsl **(style)**
url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.xml **(XML)**
url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.html **(résultat)**

Le fichier XML (sans association de la feuille de style)

```
<?xml version="1.0"?>
<page>
 <title>Hello Cocoon friend</title>
 <content>Here is some content :)  </content>
 <comment>Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
</comment>
</page>
```

Le "fichier" XHTML résultant que l'on désire obtenir

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-
html40/strict.dtd">
<html><head><title>Hello Cocoon friend</title></head><body bgcolor="#ffffff">
 <h1 align="center">Hello Cocoon friend</h1>
 <p align="center"> Here is some content :) </p>
```

```
<hr> Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
 </body></html>
```

Le fichier XSL

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="page">
.....
   <html> <head> <title> <xsl:value-of select="title"/> </title> </head>
    <body bgcolor="#ffffff">
     <xsl:apply-templates/>
    </body>
   </html>
  </xsl:template>

  <xsl:template match="title">
   <h1 align="center"> <xsl:apply-templates/> </h1>
  </xsl:template>

  <xsl:template match="content">
   <p align="center"> <xsl:apply-templates/> </p>
  </xsl:template>

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


3. Introduction à XPath

- XPath est un langage qui permet d'identifier un élément dans un arbre (texte) XML selon certains critères, par exemple par son nom.

  url: http://www.w3.org/TR/xpath

Expressions Xpath (simples)

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

- Le principe d'organisation de base ressemble à celui des noms de fichiers (chemins)
- XSLT utilise XPath par ex. pour indiquer à quel élément il faut appliquer une règle
- Les débutants peuvent utiliser des simples noms de balises:

```
<xsl:apply-templates select="title"/>
```

- Pour les experts ces expressions peuvent devenir très compliquées:

```
<xsl:apply-templates
          select="cours/module[position()=1]/section[position()=2]"/>
```


3.1 Xpath patterns de base:

Elément syntaxique		Exemple d'un pattern	Exemple d'un match
tag	nom d'élément	project	<project> ...... </project>
/	sépare enfants direct	project/title	<project><title> ...
		/	(correspond à l'élément racine)
//	descendant	project//title	<project><problem><title>....
*****	"wildcard"	*/title	.<bla><title> *et* <bli><title>
|	opérateur "ou"	title|head	<title>...</title> *ou* <head> ...</head>
		|/|@	(tous les éléments: les enfants, la racine et les attributs de la racine)
../	élément supérieur	../problem	<project>
@	nom d'attribut	@id	id="test"
		project/@id	<project id="test" ...> ... </project>
@attr='type'		list[@type='ol']	<list type="ol"> ...... </list>

4. Elements XSL de base

Opérations de base:

1. Définir des règles qui disent comment transformer un "noeud" (element, tag) et ses sous-éléments
2. Organiser l'application de ces règles

4.1 Définition d'une règle ("template") avec xsl:template

Structure d'une règle (template) XSLT:

Sélecteur d'éléments XML à transformer → **<xsl:template match="*pattern*">**

Ici vont s'insérer les transformations → *pattern d'action*

</xsl:template>

- définit une règle "template" (chablon) qui permet de sélectionner un noeud de l'arbre XML pour un traitement (transformation)
- "match = <expression XPath>" définit le noeud à sélectionner
- Les noeuds peuvent être définis par un chemin complet (pour définir des contextes)

Exemple 4-1: Exemples xsl:template

L'exemple suivant sélectionne tous les noeuds formés par des balises "project":

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

L'exemple suivant sélectionne <title> qui est un enfant de <project>:

```
<xsl:template match="project/title">
  ......
</xsl:template>
```

L'exemple suivant sélectionne <title>, un descendant de <project>:

```
<xsl:template match="project//title">
  ......
</xsl:template>
```


4.2 Anatomie d'un simple style sheet

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

Déclarations XSLT (début de la feuille de style)

```
<xsl:template match="page">
  .......
 <html>
   <head> <title>
        <xsl:value-of select="title"/>
   </title> </head>
   <body bgcolor="#ffffff">
     <xsl:apply-templates/>
   </body>
   </html>
</xsl:template>
```

Règle pour l'élément racine du document

```
<xsl:template match="title">
  <h1 align="center"> <xsl:apply-templates/> </h1>
</xsl:template>
```

Une autre règle

```
...........

</xsl:stylesheet>
```

Fin de la feuille

4.3 Application de templates aux sous-éléments

A. <xsl:apply-templates />

- Un simple apply-templates (sans attributs) examine tous les noeuds enfants
- Dans l'exemple suivant on voit une simple règle pour la racine:

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

B. L'attribut "select" de apply-templates

- permet de spécifier un élément nommé (au lieu de tous les sous-éléments)
- Dans l'exemple ci-dessous une fois un élément <project> identifié, on traite seulement le sous-élément <title>

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

Cette règle pourrait s'appliquer au texte XML suivant:

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/hello-page.sxml.text

4.4 Extraction d'une valeur

A. xsl:value-of

- Sélectionne le contenu d'un élément et le copie vers le document "sortie"
- Autrement dit: extraction d'un seul sous-élément (ou sous-attribut)
- La règle suivante se déclenche dès qu'une balise <projet> est retrouvée et insère dans le document sortie le contenu de l'élément <title> qui se trouve à l'intérieur d'un sous-élément <problem>

```
<xsl:template match="project">
    <P>
      <xsl:value-of select="problem/title"/>
    </P>
  </xsl:template>
```

Syntaxe spéciale pour insérer la valeur d'un objet dans un string d'attribut utilisé dans l'output d'un template: `{....}`

```
<xsl:template match="contact-info">
....
  <a href="mailto:{@email}"><xsl:value-of select="@email"/></a>
...
```

B. xsl:copy

Copie également les "matched tags" (pas juste le contenu)

- Utile pour reproduire l’original (ici un tag <p>...</p>)

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

- Utile pour récupérer tout ce qui n’a pas été défini, mais attention si le tag ne correspond pas à un tag HTML il faut regarder le source HTML produit !

```
<xsl:template match="*">
   <xsl:copy>Garbage: <i> <xsl:apply-templates/> </i> </xsl:copy>
  </xsl:template>
```

- Le fragment suivant copie tous les éléments non traités par les autres templates:
  - utile si vous utilisez des balises XHTML que vous ne désirez pas traiter ou encore si XSLT sert juste à "enrichir" votre code XML (genre table de matières)

```
<xsl:template match="*|@*">
  <xsl:copy>
   <xsl:apply-templates select="@*"/>
   <xsl:apply-templates select="node()"/>
  </xsl:copy>
 </xsl:template>
```


4.5 Boucles et conditions

A. <xsl:if>

- <xsl:if> Permet de changer l’output en fonction d’un test
- Attention:
  - il n’existe pas de "else" (utilisez "choose" à la place)
  - le mot clé "test" doit apparaître tel quel dans une clause if (valable aussi pour choose)

Exemple 4-2: xsl:if exemple pour insérer des virgules dans une liste

```
<xsl:template match="animal">
  Nom:<xsl:value-or select="@name"/>
  <br> Couleurs:
   <xsl:value-of select="couleur"/>
   <xsl:if test="position()!=last()">, </xsl:if>
</xsl:template>
```

XML:

```
<animal name="zebre"> <couleur>noir</couleur> <couleur>blanc</couleur>
<couleur>bleu</couleur> </animal>
```

Résultat:

```
       Nom: zebre
       Couleur: noir, blanc, bleu
```


B. <xsl:choose>

- exemple (à élaborer)

```
<xsl:template match="animal">
  <xsl:choose>
    <xsl:when test="@couleur='noir'">
      <P style="color:black">
        <xsl:value-of select="."/>
      </P>
    </xsl:when>
     <xsl:when test="@couleur='bleu'">
      <P style="color:blue">
        <xsl:value-of select="."/>
      </P>
    </xsl:when>
    <xsl:when test="pattern">
       ...
    </xsl:when>
    <xsl:otherwise>
      <P style="color:green">
        <xsl:value-of select="."/>
      </P>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```


C. <xsl:for-each>

- sert à faire des tables par exemple

Exemple 4-3: Présentation du résultat XML d’une requête SQL avec XSLT

url: http://tecfa.unige.ch/guides/xml/cocoon/mysql/mysql-demo2.sxml
url: http://tecfa.unige.ch/guides/xml/cocoon/mysql/mysql-demo2.xsl

```
<xsl:template match="ROWSET">
    <table border="2" cellspacing="1" cellpadding="6">
     <xsl:for-each select="ROW">  <tr>
      <td><xsl:value-of select="id"/></td>
      <td><xsl:value-of select="login"/></td>
      <td><xsl:value-of select="fullname"/></td>
       <td bgcolor="tan"><a href="{url}"><xsl:value-of select="url"/></a></td>
      <td><xsl:value-of select="food"/></td>
      <td><xsl:value-of select="work"/></td>
      <td><xsl:value-of select="love"/></td>
      <td><xsl:value-of select="leisure"/></td>
      </tr> </xsl:for-each>
    </table>
</xsl:template>
```

4.6 Fabrication de références (liens)

Exemple 4-4: Table de matières pour éléments qui ont un identificateur

url: http://tecfa.unige.ch/guides/xml/examples/recit/

Fragment XSLT (pour faire les liens au départ, voir: mode="toc")

```
<xsl:template match="/">
  <html>
   <body bgcolor="#FFFFFF">
     <h1><xsl:value-of select="/RECIT/Titre"/></h1>
     <p> Highlights de l'histoire:
         <xsl:apply-templates select="//EPISODE" mode="toc"/> </p>
     <xsl:apply-templates/>
   </body>
 </html>
</xsl:template>
```

Fragments XSLT (pour gérér la balise "EPISODE")

```
<xsl:template match="EPISODE" mode="toc">
  <a href="#{@id}"><xsl:value-of select="SousBut"/></a> -
</xsl:template>

<xsl:template match="/RECIT/FIL/EPISODE">
  <a name="{@id}"> <hr /> </a>
  <xsl:apply-templates/>
</xsl:template>
```


Exemple 4-5: Tables de matières pour éléments sans ID

- plus difficile car il faut "fabriquer" des attributs "name" et "href" pour le HTML
  - la solution adopté ici est moche (on aurait pu compter les éléments)
- il s'agit des jours d'un atelier webmaster qu'on peut consulter ici:

  url: http://tecfa.unige.ch/tecfa/teaching/formcont/webmaster2003/programme/
- Pour comprendre cet exemple il faut fouiller dans les fichiers suivants:
  - programme.xml - contient tout le "programme" de l'Atelier
  - programme.xsl - produit le résultat (voir le fichier programme.html)
  - matos.xsl, resume.xsl, animateurs.xsl créent des extraits variés du programme.

Fragments XSLT

```
<!-- Code qui fabrique la table des matières (jours) en HTML -->
Programme: <xsl:apply-templates select="//day" mode="toc" />

<xsl:template match="day" mode="toc">
  <a href="#{@name}{@dayno}{@month}"><xsl:value-of select="@name"/></a> -
</xsl:template>
<!-- Code pour insérer des attributs "name" dans le HTML -->
<xsl:template match="day">
   <a name="{@name}{@dayno}{@month}"><xsl:value-of select="@name"/></a> -
    <xsl:value-of select="@dayno"/>/<xsl:value-of select="@month"/>/
    <xsl:value-of select="@year"/>
</xsl:template>
```


Fragment XML (élément à extraire)

- Faire une table des matières avec les jours de l'Atelier (avec liens)

```
<day name="lundi" year="2003" month="6" dayno="16">
```

Résultat HTML

- Au début du fichier on a un menu qui affiche les jours (liens vers le bas)

```
Programme: <a href="#lundi166">lundi</a> - <a href="#mardi176">mardi</a> - ....
......
```

- Dans le fichier on insère les attributs "name"

```
<a name="lundi166">lundi</a> - 16/6/2003
```

Note:

- Dans ce répertoire il y a aussi un fichier programme-fo.xsl qui génère du code xsl-fo utilisé pour générer la version PDF du programme.

Exemple 4-6: Tables de matières pour éléments sans ID

- Table de travaux pour une "page travaux STAF" (portfolio étudiant)

  url: http://tecfa.unige.ch/guides/xml/examples/xsl-toc/

4.7 Numérotations

Exemple 4-7: Numérotation d'éléments

url: http://tecfa.unige.ch/guides/xml/examples/xsl-numbering/

Variante 1:

```
<xsl:template match="sections">
  <xsl:for-each select="section">
   <p>
    <xsl:number value="position()" format="a. "/>
    (<xsl:value-of select="deposit-date"/>) <a href="{url}"><xsl:value-of
select="title"/></a>
    .........
   </p>
  </xsl:for-each>
 </xsl:template>
```

Variante 2:

```
<xsl:template match="sections">
  <xsl:for-each select="section">
....
   <xsl:number level="multiple"
               count="exercise|section"
               format="1.1 "/>
....
```

5. XSLT en "batch"

Il existe plusieurs processeurs XSLT populaires qu'on peut utiliser pour produire un fichier de sortie à partir d'un xml + xslt

- Xalan du projet Apache:
  url: http://xml.apache.org/xalan/overview.html
- Saxxon:
  url: http://saxon.sourceforge.net/

  Certains outils nécessitent l'installation d'un environnement Java. Certains editeurs XML ont un processeur XSLT intégré, comme Xerlin (selon la version!)

  url: href="http://www.xerlin.org/ (dezipper et clicker sur le *.jar)

5.1 Saxon (conseillé)

- Ce processeur a quelques extensions intéressantes (ex. créer pleins de fichiers)
- Note: Nécessite Java 1.4; la "instant version" est à éviter maintenant.
- Il faut ouvrir une fenêtre de commande et taper 'saxon' pour avoir la syntaxe, utilisations typiques:

```
saxon fichier.xml fichier.xsl
saxon -o output.html fichier.xml fichier.xsl
```


5.2 XSLT avec XALAN

A TECFA on utilise aussi Xalan

Pour l'utiliser :

- il faut installer JDK 1.4 ("JavaTM 2 Platform, Standard Edition ") pour pourvoir utiliser Xalan:
  url: http://java.sun.com/products/
- Télécharger Xalan
  url: http://xml.apache.org/xalan-j/
- créer un répertoire c:\bin et le mettre dans le path du autoexec.bat
  `SET PATH=%PATH%:c:\bin`
- Mettre Xalan, c.a.d. `xalan.jar` et `xml-apis.jar` et `xercesImpl.jar` dans c:\bin
- installer/écrire un fichier c:\bin\xalan.bat pour se facilier la vie:
  Contenu du fichier xalan.bat:

```
set CLASSPATH=c:\bin\xalan.jar;c:\bin\xercesImpl.jar;xml-apis.jar
java org.apache.xalan.xslt.Process %1 %2 %3 %4 %5 %6 %7 %8 %9 | more
```


Options "lignes de commande" pour Xalan

```
-IN inputXMLURL
[-XSL XSLTransformationURL]
[-OUT outputFileName]
[-E (Do not expand entity refs)]
[-QC (Quiet Pattern Conflicts Warnings)]
[-TT (Trace the templates as they are being called.)]
[-TG (Trace each generation event.)]
[-TS (Trace each selection event.)]
[-TTC (Trace the template children as they are being processed.)]
[-TCLASS (TraceListener class for trace extensions.)]
[-EDUMP {optional filename} (Do stackdump on error.)]
[-XML (Use XML formatter and add XML header.)]
[-TEXT (Use simple Text formatter.)]
[-HTML (Use HTML formatter.)]
[-PARAM name expression (Set a stylesheet parameter)]
[-L use line numbers for source document]
[-MEDIA mediaType (use media attribute to find stylesheet associated with a
document.)]
[-FLAVOR flavorName (Explicitly use s2s=SAX or d2d=DOM to do transform.)]
[-DIAG (Print overall milliseconds transform took.)]
[-URIRESOLVER full class name (URIResolver to be used to resolve URIs)]
[-ENTITYRESOLVER full class name (EntityResolver to be used to resolve
entities)]
[-CONTENTHANDLER full class name (ContentHandler to be used to serialize
output)]
```


6. XML + XSL avec Cocoon 1 (usage déconseillé maintenant)

- Cocoon 1 une des solutions "XML server-side" adoptée à TECFA (à remplacer)
- Voir le module xml-ser pour les autres "features" de Cocoon 1 !
- Il existe pleins d'autres alternatives pour servir XML + XSL, par ex. php !

6.1 Cocoon (simple) @ TECFA

url: http://tecfa2.unige.ch/guides/xml/local/cocoon/docs/ **(doc locale)**
url: http://tecfa2.unige.ch/guides/xml/local/cocoon/samples/
url: http://tecfa.unige.ch/guides/xml/cocoon/ **(exmples locaux)**

- Cocoon est accessible de façon transparente depuis nos serveurs Apache tecfa.unige.ch (versions test sur tecfa2.unige.ch)

Voici la procédure:

- Votre fichier XML doit s'appeler *.sxml
- rajouter dans le fichier XML (votre_fichier.sxml) ces 2 processing instructions:

```
<?xml-stylesheet href="votre_fichier.xsl" type="text/xsl" ?>
<?cocoon-process type="xslt"?>
```

- IMPORTANT: Le fichier xsl doit contenir dans le template pour la racine XML:

```
<xsl:processing-instruction name="cocoon-format">type="text/html"
</xsl:processing-instruction>
```

6.2 Exemples Cocoon

Exemple 6-1: Sensibilisation XML + XSLT avec Cocoon

- Il s'agit de nouveau de l'exemple 2-1 "Sensibilisation XML + XSLT" [8]. Cette fois-ci avec les balises spéciales pour Cocoon et un DTD

A. La grammaire (le DTD)

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/hello-page.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT page (title, content, comment?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```


B. Le fichier XML

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/hello-page.sxml

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/hello-page.sxml.text

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="hello-page-html.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>

<page>
 <title>Hello Cocoon friend</title>
 <content>
  Here is some content. Olé !
 </content>
 <comment>
      Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
 </comment>
</page>
```


C. Le fichier XSLT

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/hello-page.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="page">
   <xsl:processing-instruction name="cocoon-format">type="text/html"</xsl:processing-
instruction>
   <html>
    <head>
     <title> <xsl:value-of select="title"/> </title>
    </head>
    <body bgcolor="#ffffff">
     <xsl:apply-templates/>
    </body>
   </html>

  </xsl:template>
  <xsl:template match="title">
   <h1 align="center">
    <xsl:apply-templates/>
   </h1>
  </xsl:template>
  <xsl:template match="content">
   <p align="center">
     <xsl:apply-templates/>
   </p>
  </xsl:template>
  <xsl:template match="comment">
   <hr />
     <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```


(à refaire - maintenant on utilise XML client-side pour cela)

Exemple 6-2: Plans de cours

- Voir les pages formation continue

  url: http://tecfa.unige.ch/tecfa/teaching/formcont/webmaster2001/formcont.sxml

  url: http://tecfa.unige.ch/tecfa/teaching/formcont/webmaster2001/welcome.xsl

  url: http://tecfa.unige.ch/tecfa/teaching/formcont/webmaster2001/module2/

Exemple 6-3: Project Management

- Voir pour le moment les page STAF-18

  url: http://tecfa.unige.ch/tecfa/teaching/staf18/staf18-overview.html

7. Server-side avec PHP

Exemple 7-1: XSLT avec PHP/Sablotron

- nécssite une installation appropriée de PHP (XSLT support = enabled)

  url: http://tecfa.unige.ch/guides/php/examples/xslt-sablotron/

  url: http://tecfa.unige.ch/guides/php/examples/xslt-sablotron/programme.php

```
<?php
error_reporting(63);
$xml = "programme.xml";
$xslt = "programme.xsl";

 // Allocate a new XSLT processor
 $xh = xslt_create();

 // Process the document, returning the result into the $result variable
 $result = xslt_process($xh, $xml, $xslt);
 if ($result) { print $result; }
 else {
     print "Sorry, sample.xml could not be transformed by sample.xsl into";
     print "  the \$result variable the reason is that " . xslt_error($xh) .
     print " and the error code is " . xslt_errno($xh);
     }
 xslt_free($xh);
 ?>
```


8. Client-side XML+XSLT avec Mozilla ou IE6

Exemple 8-1: Un simple exemple

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.dtd **(DTD)**

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.xml.text **(src XML)**

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page-html.xsl **(style)**

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.xml **(XML)**

url: http://tecfa.unige.ch/guides/xml/examples/simple/hello-page.html **(résultat)**

- Il faut ajouter la declaration suivante dans le fichier XSL (il s'agit d'une declaration XSL standarde qui permet d'indiquer la méthode de sérialisation)

```
<xsl:output method="html"/>
```

- Voir les fichiers pour les détails. Si aucun contenu n'est affiché ou s'il est mal affiché, faites "Menu->View->Source"
- Notes:
  - http://www.mozilla.org/releases/ (Mozilla downloads)
  - Evitez de travailler avec IE 5.5. La version "normale" n'est pas conforme au standard XSLT. Installez soit IE 6.x soit MSXML3+,pour IE 5.5.
  - Il faut s'assurer à ce que le serveur Web indique le bon mime type pour xml et xsl (text/xml). Ceci est fait à Tecfa, mais pas forcément chez votre fournisseur.