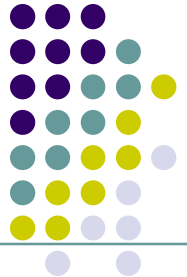
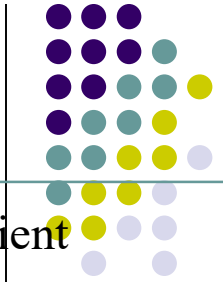
Université Gaston Berger de Saint-Louis

Année académique 2016-2017



XSLT

EXtensible Stylesheet Language Transformations



Références

1. XML Programming Success in a Day: Beginner's Guide to Fast, Easy, and Efficient Learning of XML Programming. Sam Key, 2015.

2. XML Programming: The Ultimate Guide to Fast, Easy, and Efficient Learning of XML Programming. Christopher Right, 2015

3. Beginning XML. Joe Fawcett and Danny Ayers, 2012.

4. XML, Cours et exercices. Modélisation, Schémas et DTD, design patterns, XSLT, DOM, Relax NG, XPath, SOAP, XQuery, XSL-FO, SVG, eXist. *Alexandre Brillant*, Édition : Eyrolles 2eédition, 2010

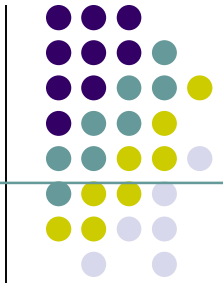
5. Schémas XML, Jean-Jacques Thomasson, Edition Eyrolles 2002

6. World Wide Web Consortium (W3C) : w3.org 00336241511074

7. Tutoriel XSLT de World Wide Web School : w3schools.com/xsl

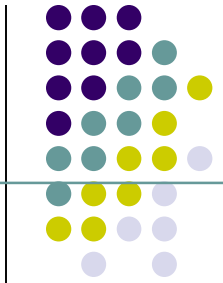
8. xsl.developpez.com

9. Youtube.com



Sommaire

- Introduction
- La syntaxe XSLT
- Les éléments XSLT
- Transformation côté client
- Transformation côté serveur
- Edition d'un document XML

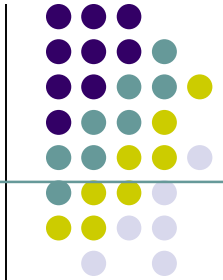


Introduction

Qu'est-ce que XSL?

- **XSL** (E**X**tensible **S**tylesheet **L**anguage) est un langage de feuille de style pour documents XML. Il est composé des langages:
  - XSLT - pour transformer des documents XML
  - XPath - pour naviguer dans des documents XML
  - XSL-FO - pour formater des docs XML (interrompu en 2013)
  - XQuery - d'interrogation de documents XML

- Avec le **module CSS3 Paged Media** , W3C a livré un nouveau standard pour le formatage de document XML. Ainsi, depuis 2013, CSS3 est proposé pour remplacer XSL-FO.

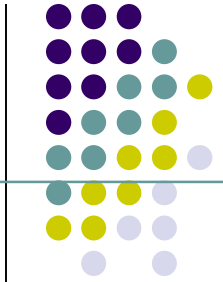


Introduction

Qu'est-ce que XSLT?

XSLT (**XSL T**ransformations) est la partie la plus importante de XSL :

- Il permet de transformer un document XML en un autre document XML ou un autre type de document reconnu par les navigateurs.
- Il est supporté par les principaux navigateurs
- Il utilise XPath pour naviguer dans un document XML
- Il est une recommandation du W3C depuis le 16 novembre 1999
- Prérequis: HTML et XML



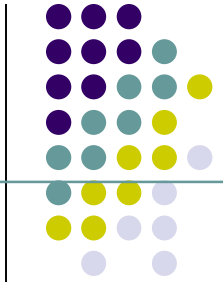
Introduction

Le processus de transformation

• XSLT est utilisé pour transformer un document XML en un autre document XML ou un autre type de document reconnu par un navigateur, comme HTML et XHTML.

• Avec XSLT, on peut ajouter / supprimer des éléments et des attributs dans ou à partir du fichier de sortie. On peut aussi réorganiser et trier les éléments, effectuer des tests et prendre des décisions sur les éléments à masquer et à afficher, etc.

• Dans le processus de transformation, XSLT utilise XPath pour définir les parties du document source devant correspondre à un ou plusieurs modèles prédéfinis. Lorsqu'une correspondance est trouvée, XSLT va transformer la partie correspondante du document source dans le document résultat.



La syntaxe

Déclaration de la feuille de style

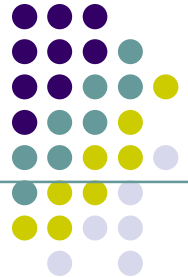
• Elle se fait avec l'une des deux éléments équivalents suivants qui constituent la racine du document XSL :

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

ou

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

- Pour accéder aux éléments, attributs et caractéristiques XSLT, l'espace de noms `xmlns:xsl` doit être déclaré en haut du document. Il pointe sur l'espace de noms XSLT officiel du W3C.
- Si on utilise cet espace de noms, on doit également inclure l'attribut `version="1.0"`.

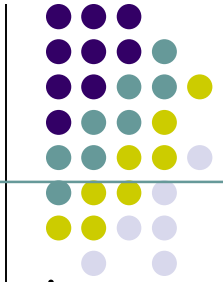


La syntaxe

Exemple : Le document XML à transformer.

Soit le document XML suivant ("cdcatalogue.xml") qu'on veut transformer en XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogue>
  <cd>
    <titre>Jokkoo</titre>
    <artiste>Youssou Ndour</artiste>
    <pays>USA</pays>
    <label>Nonesuch Records</label>
    <prix>3000</prix>
    <annee>2000</annee>
  </cd>
.
.
</catalogue>
```

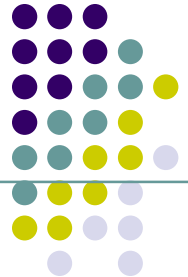


La syntaxe

Exemple : Création de la feuille de style XSL

Soit la feuille de style XSL ("cdcatalogue.xsl") avec un modèle de transformation:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html><body>
  <h2>Ma collection de CD</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Titre</th>
      <th>Artiste</th>
    </tr>
    <xsl:for-each select="catalogue/cd">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <td><xsl:value-of select="artiste"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```



La syntaxe

Exemple : Lier la feuille de style XSL au document XML

Ajouter la référence de la feuille de style XSL au document XML ("cdcatalog.xml"):

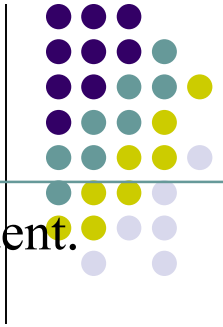
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalogue.xsl"?>
<catalogue>
  <cd>
    <titre>Jokkoo</titre>
    <artiste>Youssou Ndour</artiste>
    <pays>USA</pays>
    <label>Nonesuch Records</label>
    <prix>3000</prix>
    <annee>2000</annee>
  </cd>
.
.
</catalogue>
```

Les éléments

L'élément <xsl: template>

- Une feuille de style XSL est constitué de templates contenant des règles à appliquer quand un nœud spécifié est rencontré.

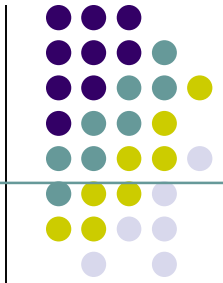
- L'élément <xsl: template> est utilisé pour construire un template. L'attribut **match** est utilisé pour associer un template à un élément XML. Il peut également être utilisé pour définir un template pour le document XML entier. La valeur de l'attribut match est une expression XPath (Ex: match = "/" définit l'ensemble du document).



Les éléments

Exemple : Soit une version simplifiée du fichier XSL de l'exemple précédent.

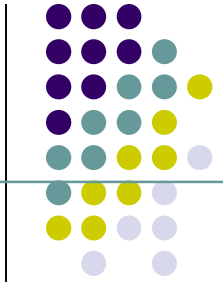
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html><body>
  <h2>Ma collection de CD</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Titre</th>
      <th>Artiste</th>
    </tr>
    <tr>
      <td>-</td>
      <td>-</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```



Les éléments

Explication de l'exemple :

- Puisqu'un fichier XSL est un document XML, il commence toujours par la déclaration XML:<**?xml version="1.0" encoding="UTF-8"?**>.
- L'élément **<xsl: stylesheet>** définit que le document est une feuille de style XSLT (avec le numéro de version et les attributs d'espace de noms XSLT).
- L'élément **<xsl: template>** définit un template. L'attribut **match** = "/" associe le template à la racine du document source XML.
- Le contenu de <xsl: template> définit le code HTML à afficher.
- Les deux dernières lignes définissent la fin du template et de la feuille de style.
- Cet exemple n'affiche aucun résultat. On verra dans la suite comment utiliser l'élément **<xsl: value-of>** pour sélectionner les valeurs des éléments XML.



Les éléments

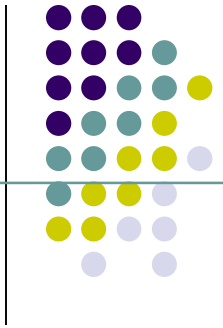
L'élément <xsl: value-of>

- Il est utilisé pour extraire la valeur d'un élément XML et l'ajouter au flux de sortie de la transformation:

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html><body>
  <h2>Ma collection de CD</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Titre</th><th>Artiste</th>
    </tr>
    <tr>
      <td><xsl:value-of select="catalogue/cd/titre"/></td>
      <td><xsl:value-of select="catalogue/cd/artiste"/></td>
    </tr>
  </table>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```



Les éléments

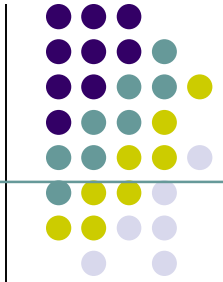
L’élément <xsl: value-of>

Explication de l’exemple :

- L’attribut **select**, dans l'exemple ci-dessus, contient une expression Xpath qui fonctionne comme la navigation d'un système de fichiers; une barre oblique (/) sélectionne les sous-répertoires.

- Le résultat de l'exemple ci-dessus est très simple. Une seule ligne de données est copiée à partir du document XML.

- Dans la suite, l’élément **<xsl:for-each>** sera utilisé pour boucler à travers les éléments XML pour afficher tous les enregistrements.



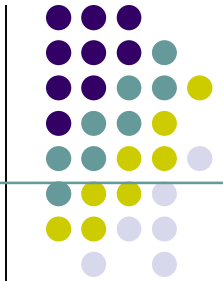
Les éléments

L'élément <xsl: for-each> permet de faire une boucle en XSLT.

- Il est utilisé pour sélectionner tous les éléments d'un ensemble de nœuds spécifié.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html><body>
  <h2>Ma collectionCD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th><th>Artist</th>
    </tr>
    <xsl:for-each select="catalogue/cd">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <td><xsl:value-of select="artiste"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```



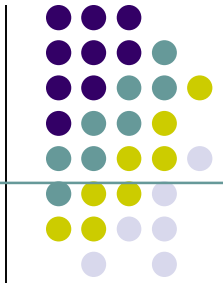
Les éléments

L'élément <xsl:sort> est utilisé pour trier la sortie.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html><body>
  <h2>Ma collection de CD</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Titre</th>
      <th>Artiste</th>
    </tr>
    <xsl:for-each select="catalogue/cd">
      <xsl:sort select="artiste"/>
      <tr>
        <td><xsl:value-of select="titre"/></td>
        <td><xsl:value-of select="artiste"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```

Remarque: L'attribut **select** indique l'élément XML par lequel on trie.



Les éléments

L'élément <xsl:if>

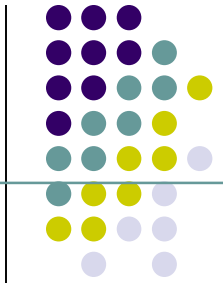
Il permet de mettre un test conditionnel sur le contenu du fichier XML et est placé dans l'élément <xsl: for-each> .

- **Syntaxe:**

```
<xsl:if test="expression">
  ... Texte de la sortie si l'expression est vraie...
</xsl:if>
```

- **Exemple:**

```
<xsl:for-each select="catalogue/cd">
   <xsl:if test="prix &gt; 1000">
     <tr>
       <td><xsl:value-of select="titre"/></td>
       <td><xsl:value-of select="artiste"/></td>
       <td><xsl:value-of select="prix"/></td>
     </tr>
   </xsl:if>
 </xsl:for-each>
```



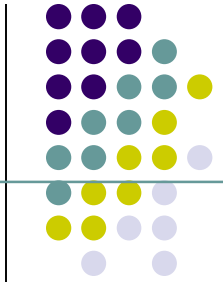
Les éléments

L’élément <xsl:if>

Remarque:

- La valeur de l’attribut obligatoire **test** contient l'expression à évaluer.

- Le code ci-dessus va seulement afficher comme sortie les éléments titre et artiste des CD ayant un prix supérieur à 10.



Les éléments

L'élément <xsl: choose>

- L'élément <xsl:choose> est utilisé en conjonction avec <xsl:when> et <xsl:otherwise> pour exprimer plusieurs tests conditionnels.

- **Syntaxe:**

```
<xsl:choose>
  <xsl:when test="expression">
    ... sortie ...
  </xsl:when>
  <xsl:otherwise>
    ... sortie ....
  </xsl:otherwise>
</xsl:choose>
```

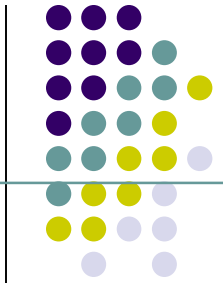
Les éléments

L'élément <xsl: choose>

- **Exemple 1:**

```
<xsl:for-each select="catalogue/cd">
 <tr>
   <td><xsl:value-of select="titre"/></td>
   <xsl:choose>
     <xsl:when test="prix &gt; 1000">
       <td bgcolor="#ff00ff">
       <xsl:value-of select="artiste"/></td>
     </xsl:when>
     <xsl:otherwise>
       <td><xsl:value-of select="artiste"/></td>
     </xsl:otherwise>
   </xsl:choose>
 </tr>
 </xsl:for-each>
```

Le code ci-dessus ajoute un fond de couleur rose à la colonne "Artiste" quand le prix du CD est supérieur à 1000.

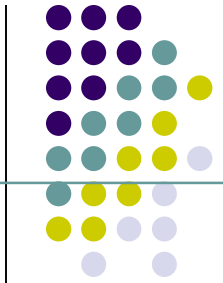


Les éléments

L'élément <xsl: choose>

- **Exemple 2:**

```
<xsl:for-each select="catalogue/cd">
 <tr>
   <td><xsl:value-of select="titre"/></td>
   <xsl:choose>
     <xsl:when test="prix &gt; 1000">
       <td bgcolor="#ff00ff">
       <xsl:value-of select="artiste"/></td>
     </xsl:when>
     <xsl:when test="prix &gt; 900">
       <td bgcolor="#cccccc">
       <xsl:value-of select="artiste"/></td>
     </xsl:when>
     <xsl:otherwise>
       <td><xsl:value-of select="artiste"/></td>
     </xsl:otherwise>
   </xsl:choose>
 </tr>
 </xsl:for-each>
```



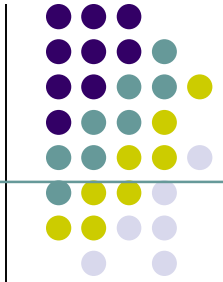
Les éléments

L’élément <xsl:apply-templates>

- Il permet d’appliquer un modèle à l'élément courant ou aux nœuds enfants de l'élément courant.

- Si on ajoute un attribut select à la l’élément <xsl:apply-templates>, il traitera uniquement l'élément enfant correspondant à la valeur de l'attribut.

- On peut utiliser l'attribut select pour spécifier l'ordre dans lequel les nœuds enfants sont traités.



Les éléments

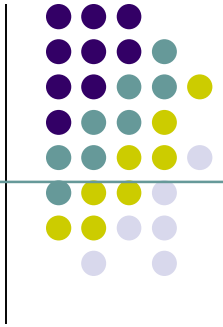
L'élément <xsl:apply-templates>

- **Exemple:**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html><body>
  <h2>Ma collection de CD</h2>
  <xsl:apply-templates/>
  </body></html>
</xsl:template>

<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="titre"/>
  <xsl:apply-templates select="artiste"/>
  </p>
</xsl:template>
```



Les éléments

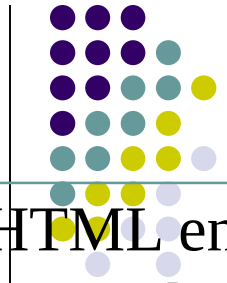
L'élément <xsl:apply-templates>

- **Exemple (suite):**

```
 ...
<xsl:template match="titre">
  Titre: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artiste">
  Artiste: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```



Transformation côté client

- On a utilisé XSLT pour transformer un document XML en XHTML en ajoutant une feuille de style XSL dans le fichier XML et en laissant le navigateur faire la transformation.

- Même si cela fonctionne bien, il n'est pas toujours souhaitable d'inclure une référence de feuille de style dans un fichier XML (ex, il ne fonctionnera pas dans un navigateur courant non XSLT.)

- Une solution plus souple serait d'utiliser JavaScript pour faire la transformation. En utilisant JavaScript, on peut:
  - faire des tests spécifiques au navigateur
  - utiliser différentes feuilles de style en fonction des besoins du navigateur et de l'utilisateur.

Telle est la beauté de XSLT! L'un des objectifs de conception de XSLT était de rendre possible la transformation des données d'un format à un autre, en prenant en charge les différents navigateurs et les besoins différents des utilisateurs.

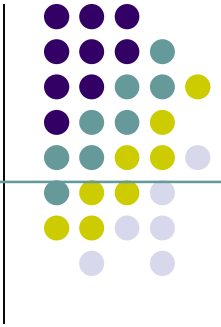
Transformation côté client

Exemple:

- On considère le document XSL cdcatalogue.xsl du diapo 16 et le document XML cdcatalogue.xml auquel on ne met pas de référence au fichier XSL.
- L'exemple ci-après contient le code permettant de transformer le document XML avec le document XSLT en utilisant JavaScript

```
<html><head>
<script>
function loadXMLDoc(filename){
  if WINDOW.ActiveXObject){
    xhttp = new ActiveXObject("Msxml2.XMLHTTP");
  }
  else {xhttp = new XMLHttpRequest();}

  xhttp.open("GET", filename, false);
  try {xhttp.responseType = "msxml-document"} catch(err) {} //IE11
  xhttp.send("");
  return xhttp.responseXML;
}
```



Transformation côté client

```
function displayResult(){
  xml = loadXMLDoc("cdcatalog.xml");
  xsl = loadXMLDoc("cdcatalog.xsl");
  // code for IE
  if WINDOW.ActiveXObject || xhttp.responseType == "msxml-document"){
    ex = xml.transformNode(xsl);
    document.getElementById("example").innerHTML = ex   ;
  }
  // code for Chrome, Firefox, Opera, etc.
  else if (document.implementation &&
  document.implementation.createDocument){
    xsltProcessor = new XSLTProcessor();
    xsltProcessor.importStylesheet(xsl);
    resultDocument = xsltProcessor.transformToFragment(xml, document);
    document.getElementById("example").appendChild(resultDocument);
  }
}
</script></head>
<body onload="displayResult()">
  <div id="example" />
</body></html>
```

Transformation côté client

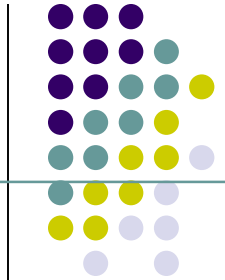
Explication :

La fonction loadXMLDoc () effectue les opérations suivantes:

- Crée un objet XMLHttpRequest
- Utilise les méthodes open() et send() de l'objet XMLHttpRequest pour envoyer une requête au serveur
- Récupère les données de la réponse au format XML

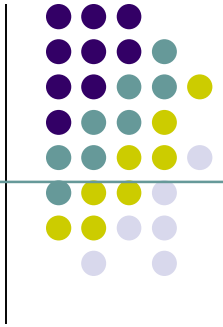
La fonction displayResult () affiche le fichier XML conçu avec le fichier XSL:

- Charge les fichiers XML et XSL
- Teste le type de navigateur de l'utilisateur
- Si c'est Internet Explorer:
  - Utilise la méthode transformNode () pour appliquer le XSL au document xml
  - Met le résultat dans le contenu de l'élément ayant id = "example"
- Si d'autres navigateurs:
  - Crée un nouvel objet XSLTProcessor et y importe le fichier XSL
  - Utilise la méthode transformToFragment () pour appliquer le XSL au doc xml
  - Met le résultat dans le contenu de l'élément ayant id = "example"



Transformation côté serveur

- On a vu comment XSLT peut être utilisé pour transformer un doc. XML en XHTML dans le navigateur avec JavaScript et un parseur XML. Cependant, cela ne fonctionnera pas dans un navigateur ne disposant pas d'un parseur XML.
- Pour rendre les données XML disponibles à tous les types de navigateurs, on peut transformer le document XML sur le serveur et l’envoyer au navigateur en format XHTML.
- **Voilà une autre beauté de XSLT. L'un des objectifs de conception de XSLT était de permettre la transformation des données d'un format à un autre dans un serveur, et de rendre ces données lisibles par tous les types de navigateurs.**



Transformation côté serveur

Exemple : Transformation avec PHP

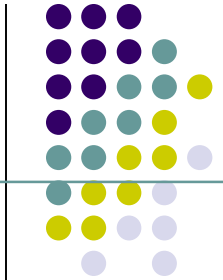
```
<?php
// Load XML file
$xml = new DOMDocument;
$xml->load('cdcatalog.xml');

// Load XSL file
$xsl = new DOMDocument;
$xsl->load('cdcatalog.xsl');

// Configure the transformer
$proc = new XSLTProcessor;

// Attach the xsl rules
$proc->importStyleSheet($xsl);

echo $proc->transformToXML($xml);
?>
```



Edition de document XML

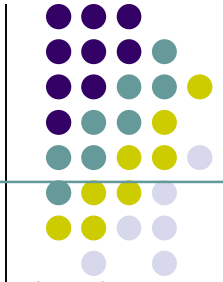
- En utilisant XSL, il est possible à partir d'un navigateur de modifier les données stockées dans un fichier XML.

- Pour cela on transforme le document XML depuis un formulaire HTML . Il s'agit :

  - d'écrire les valeurs des éléments XML dans des champs de saisie de formulaire HTML.

  - puis de modifier les données depuis le formulaire

  - ensuite d'envoyer les données du formulaire au serveur qui se chargera de mettre à jour le document XML.

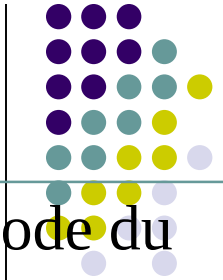


Edition de document XML

Exemple:

On considère le document XML (tool.xml) ci-dessous qu'on voudrait modifier depuis le navigateur.

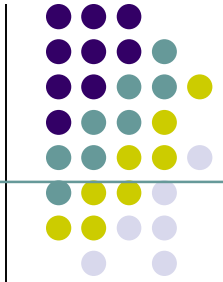
```
<?xml version="1.0" encoding="UTF-8"?>
<tool>
  <field id="prodName">
    <value>HAMMER HG2606</value>
  </field>
  <field id="prodNo">
    <value>32456240</value>
  </field>
  <field id="price">
    <value>$30.00</value>
  </field>
</tool>
```



Edition de document XML

Exemple: Le document XSL (tool.xsl) ci-dessous contenant le code du formulaire de modification de tool.xml.

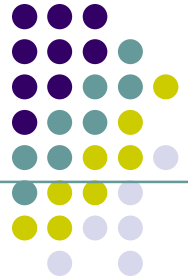
```
<xsl:template match="/">
  <html><body>
  <form method="post" action="edittool.php">
  <h2>Tool Information (edit):</h2>
  <table border="0">
    <xsl:for-each select="tool/field">
    <tr>
      <td><xsl:value-of select="@id"/></td>
      <td><input type="text">
      <xsl:attribute name="id">
         <xsl:value-of select="@id" />
      </xsl:attribute>
      <xsl:attribute name="name">
        <xsl:value-of select="@id" />
      </xsl:attribute>
```



Edition de document XML

Exemple: Suite du document XSL (tool.xsl).

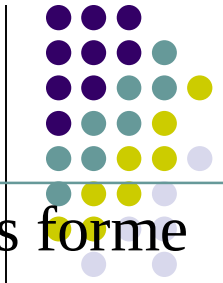
```
      <xsl:attribute name="value">
        <xsl:value-of select="value" />
      </xsl:attribute>
      </input>
      </td>
    </tr>
    </xsl:for-each>
  </table>
  <br />
<input type="submit" id="btn_sub" name="btn_sub" value="Submit"/
>
<input type="reset" id="btn_res" name="btn_res" value="Reset" />
  </form>
  </body>
  </html>
</xsl:template>
```



Edition de document XML

Exemple: Explication du code du document XSL (tool.xsl).

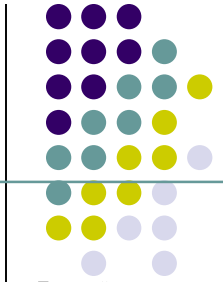
- Le fichier XSL ci-dessus boucle à travers les éléments "field" du fichier XML et crée un champ d'entrée pour chaque élément XML.
- La valeur de l'attribut "id" de l'élément "field" du document XML est ajouté à la fois dans les attributs "id" et "name" de chaque champ de saisie HTML.
- La valeur de chaque élément "value" du document XML est ajouté à l'attribut "value" de chaque champ de saisie HTML.
- Le résultat est un formulaire HTML modifiable contenant les valeurs provenant du fichier XML.
- Ensuite le fichier "tool_updated.xsl" est utilisé pour afficher sous forme de table HTML statique les données XML mises à jour.



Edition de document XML

Exemple: Le fichier "tool_updated.xsl" utilisé pour afficher sous forme de table HTML statique les données XML mises à jour.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html><body>
  <h2>Updated Tool Information:</h2>
  <table border="1">
    <xsl:for-each select="tool/field">
    <tr>
      <td><xsl:value-of select="@id" /></td>
      <td><xsl:value-of select="value" /></td>
    </tr>
    </xsl:for-each>
  </table>
  </body></html>
</xsl:template>
```



Edition de document XML

Exemple: Le fichier "edittool.php"

La page "edittool.php" contient deux fonctions: la fonction loadFile() charge et transforme le fichier XML à afficher et la fonction updateFile() applique les modifications apportées au fichier XML:

```
<?php
function loadFile($xml, $xsl){
$xmlDoc = new DOMDocument();
$xmlDoc->load($xml);

$xslDoc = new DOMDocument();
$xslDoc->load($xsl);

$proc = new XSLTProcessor();
$proc->importStyleSheet($xslDoc);
echo $proc->transformToXML($xmlDoc);
}
...
```

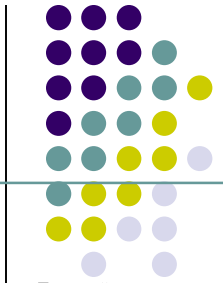
Edition de document XML

Exemple: Le fichier "edittool.php" contient deux fonctions: la fonction loadFile() charge et transforme le fichier XML à afficher et la fonction updateFile() applique les modifications apportées au fichier XML:

```
function updateFile($xml){
$xmlLoad = simplexml_load_file($xml);
$postKeys = array_keys($_POST);

foreach($xmlLoad->children() as $x){
  foreach($_POST as $key=>$value){
    if($key == $x->attributes()){
      $x->value = $value;
    }
  }
}

$xmlLoad->asXML($xml);
loadFile($xml,"tool_updated.xsl");
}
```



Edition de document XML

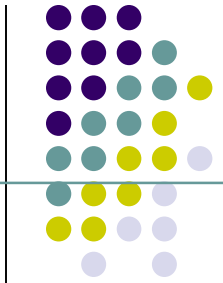
Exemple: Le fichier "edittool.php"

La page "edittool.php" contient deux fonctions: la fonction loadFile() charge et transforme le fichier XML à afficher et la fonction updateFile() applique les modifications apportées au fichier XML:

```
...
if($_POST["btn_sub"] == ""){
  loadFile("tool.xml", "tool.xsl");
}
else{
  updateFile("tool.xml");
}
?>
```

Remarque: La transformation est faite et les modifications apportées au fichier XML sont appliquées au serveur. Ceci est une solution multi-navigateur qui fonctionne dans tous les navigateurs. Le client ne recevra que le code HTML provenant du serveur.

FIN



FIN