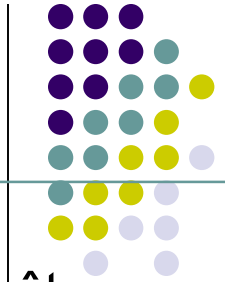
Formatage d'un document XML

Manipulation de données XML

El hadji Mamadou NGUER Enseignant chercheur en Informatique à l'UVS

Séquence 3 : Formatage d'un document XML

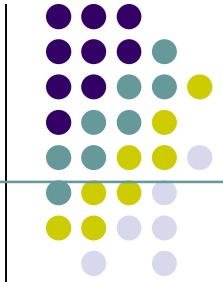


Objectifs spécifiques : A la suite de cette séquence, l'apprenant doit être capable de:

- Décrire le formatage d'un document XML
- Décrire les différentes méthodes de formatage d'un document XML
- Formater un document XML avec le langage CSS
- Formater un document XML avec le langage JavaScript
- Formater un document XML avec langage XSLT
- Formater un document XML avec XSLT via JavaScript
- Formater un document XML avec XSLT via PHP

elhadjimamadou.nguer@uvs.edu.sn

Séquence 3 : Formatage d'un document XML

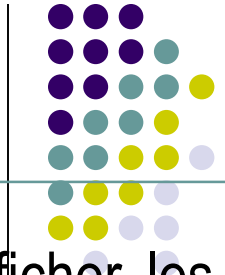


Plan :

- Introduction
- Formatage avec CSS
- Formatage avec JavaScript
- Formatage avec XSLT
- Les éléments XSL
- Transformation côté client
- Transformation côté serveur

elhadjimamadou.nguer@uvs.edu.sn

Introduction



- Un document XML ne porte pas d'information sur la façon d'afficher les données.
- Comme les balises XML sont données par l'auteur du document XML, les navigateurs ne savent pas si une balise comme <table> décrit un tableau HTML ou une table à manger.
- Sans aucune information sur la façon d'afficher les données, le navigateur peut simplement afficher le document XML comme tel.
- Pour régler ce problème, on utilise CSS, JavaScript ou XSLT.
- Cependant CSS n'est pas recommandé pour afficher du XML.

elhadjimamadou.nguer@uvs.edu.sn

Formatage avec CSS

- Formater un document XML avec CSS consiste à lui appliquer une feuille de style CSS externe via la déclaration `<?xml-stylesheet ... ?>`.

- Cette feuille de style externe contient des règles dont les sélecteurs sont les noms des éléments du document XML.

- Syntaxe

cd_catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
  <CD>
    <TITLE>Senegambia</TITLE>
    <ARTIST>Musaa Ngom</ARTIST>
    <COUNTRY>Gambia</COUNTRY>
    <COMPANY>Xippi</COMPANY>
    <PRICE>3000</PRICE>
    <YEAR>1985</YEAR>
  </CD>
```

cd_catalog.css

```
CD {
   background-color: #FFFFFF;
   width: 100%
}
```

Formatage avec CSS

Exemple : Formatage d'un document cd_catalog.xml avec un fichier CSS cd_catalog.css.

cd_catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Senegambia</TITLE>
    <ARTIST>Musaa Ngom</ARTIST>
    <COUNTRY>Gambia</COUNTRY>
    <COMPANY>Xippi</COMPANY>
    <PRICE>3000</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Solda</TITLE>
    <ARTIST>Umar Pen</ARTIST>
    <COUNTRY>Senegal</COUNTRY>
    <COMPANY>Jolloli</COMPANY>
    <PRICE>4000</PRICE>
    <YEAR>1983</YEAR>
  </CD>
  .
  .
  .
</CATALOG>
```

Browser result

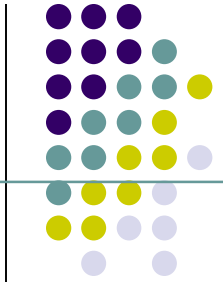
Senegambia
Musaa Ngom
Gambia
Xippi
3000
1985

Solda
Umar Pen
Senegal
Jolloli
4000
1983

cd_catalog.css

```
CATALOG {
    background-color: #ffffff;
    width: 100%;
}
CD {
    display: block;
    margin-bottom: 30pt;
    margin-left: 0;
}
TITLE {
    display: block;
    color: #ff0000;
    font-size: 20pt;
}
ARTIST {
    display: block;
    color: #0000ff;
    font-size: 20pt;
}
COUNTRY, PRICE, YEAR, COMPANY {
    display: block;
    color: #000000;
    margin-left: 20pt;
}
```

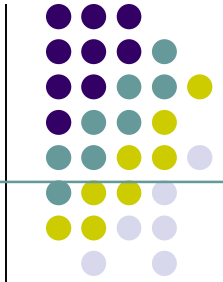
Formatage avec CSS



Les limites du formatage avec CSS

- CSS n'a pas été conçu pour formater du XML mais plutôt du html
- Il ne permet pas de réorganiser et trier les éléments, effectuer des tests et prendre des décisions sur les éléments à masquer et à afficher, et faire beaucoup plus.
- Et enfin, inclure une référence de feuille de style dans un fichier XML n'est pas toujours permis

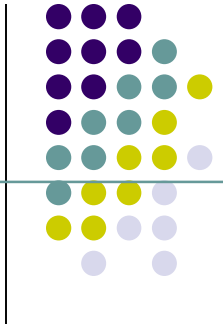
Formatage avec JavaScript



Pour afficher du XML dans un document HTML, JavaScript utilise :

- Le DOM XML (Document Object Model) qui définit les propriétés et les méthodes d'accès et d'édition d'un document XML.
- L'**objet XMLHttpRequest** intégré aux navigateurs pour accéder aux données XML à partir d'un serveur.
- Et le **parseur XML** intégré au navigateur pour convertir du texte en un objet DOM XML, y accéder et le manipuler.

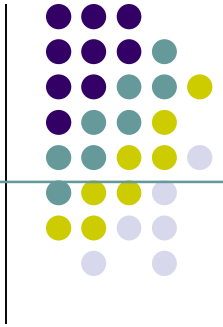
Formatage avec JavaScript



Exemple avec l'objet XMLHttpRequest

```
<p id="demo"></p>
<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
        myFunction(xhttp);
    }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();

function myFunction(xml) {
    var xmlDoc = xml.responseXML;
    document.getElementById("demo").innerHTML =
    xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
}
</script>
```

Formatage avec JavaScript



Exemple avec le parseur XML

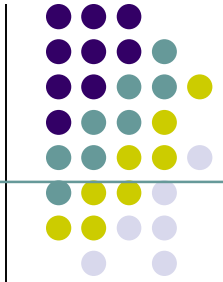
- L'exemple suivant analyse une chaîne de caractères dans un objet DOM XML et extrait les informations avec JavaScript:

Exemple

```
<p id="demo"></p>
<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>JavScript de base</title>" +
"<author>Moussa Lo</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc =
parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML
=
xmlDoc.getElementsByTagName("title")[0].child
Nodes[0].nodeValue;
</script>
```

NB : Le formatage avec JavaScript sera traité en détails dans la suite.

Formatage avec CSS



Les limites du formatage avec JavaScript

- Le complexité du langage JavaScript constitue un obstacle
- JavaScript peut être désactivé dans le navigateur.
- Il ne fonctionnera pas dans un navigateur ne disposant pas d'un parseur XML

Formatage avec XSLT

- XSLT (eXtensible Stylesheet Language Transformations) permet de transformer un document XML en HTML.
- C'est le langage de feuille de style recommandé pour XML et est beaucoup plus sophistiqué que CSS.
- Avec XSLT, on peut ajouter/supprimer des éléments et des attributs à partir du fichier de sortie.
- On peut également réorganiser et trier les éléments, effectuer des tests et prendre des décisions sur les éléments à masquer et à afficher, et faire beaucoup plus.
- XSLT utilise le langage XPath pour trouver des informations dans un document XML.

Formatage avec XSLT

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".

Formatage avec XSLT

La syntaxe : Le document XML à transformer

Exemple : 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>
```

Formatage avec XSLT

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>
```

Formatage avec XSLT

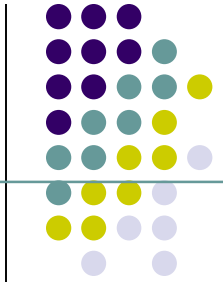
Exemple : Lier la feuille de style XSL au document XML

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 xsl

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 XSL

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 XSL

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 XSL

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 XSL

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 XSL

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 XSL

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 XSL

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 xsl

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 xsl

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 xsl

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.

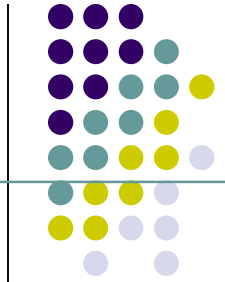
Les éléments xsl

L'élément <xsl: choose>

- **Exemple 2:**

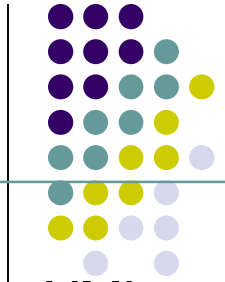
```
<xsl:for-each select="catalogue/cd">
 <tr>
   <td><xsl:value-of select="titre"/></td>
   <xsl:choose>
     <xsl:when test="prix &gt; 1000">
       <td bgcolor="#ff00ff">
       <xsl:value-of select="artiste"/></td>
     </xsl:when>
     <xsl:when test="prix &gt; 900">
       <td bgcolor="#cccccc">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



- XSLT a été conçu pour formater du XML
- Il permet de réorganiser et trier les éléments, effectuer des tests et prendre des décisions sur les éléments à masquer et à afficher, et faire beaucoup plus.
- Mais comme CSS, on inclut une référence du feuille de style XSLT dans le fichier XML. Ce qui n'est pas toujours permis.
- Pour éviter ce problème, la transformation se fera via JavaScript (dans le client) ou via PHP (dans le serveur).

Transformation côté client

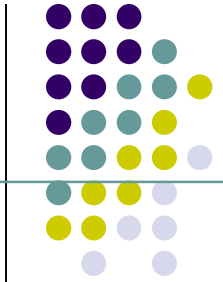


- On a vu comment utiliser XSLT pour transformer un document XML en XHTML en ajoutant une feuille de style XSL dans le fichier XML et en laissant le navigateur faire la transformation.

- 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.

Transformation côté client



Exemple:

- On considère le document XSL cdcatalogue.xsl 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 :

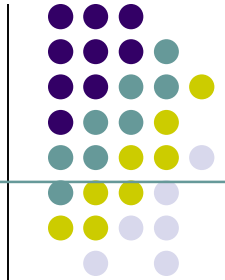
La fonction loadXMLDoc () effectue les opérations suivantes:

- 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);
?>
```

Fin de la séquence 3