Transformations XSLT en PHP 5

L'extension *xslt* et l'extension *domxml-xslt*, PHP 5 ne fournissent une méthode pour transformer un fichier xml à l'aide d'une feuille xsl basée sur la *libxslt :* la classe *XSLTProcessor*.

Le fichier XML de référence

Ce fichier XML servira pour tous les exemples . Nous le nommerons ***dvd.xml.***

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<dvd>
    <titre>Ma collection de DVD</titre>

    <item date="2004-03-25">
        <titre>Le monde de Nemo</titre>
        <categorie>Animation</categorie>
    </item>
    <item date="2004-06-06">
        <titre>Shreck3D</titre>
        <categorie>Animation</categorie>
    </item>
    <item date="2004-05-20">
        <titre>Kill Bill</titre>
        <categorie>Action</categorie>
    </item>
</dvd>
```

Exemple 1 : affichage simple

Le fichier XSL

Cette template XSLT (que nous nommerons *dvd.xsl* comme dans tous les exemples suivants) récupérera le titre (<titre>) du premier item (<item>) et le placera à l'interieur d'une balise de titre <h3>. Elle récupérera également la catégorie (<categorie>) et la date (attribut du tag <item>) .

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

<xsl:output
    indent="yes"
    method="html"
    omit-xml-declaration="no"
    encoding="iso-8859-1"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd"/>
```

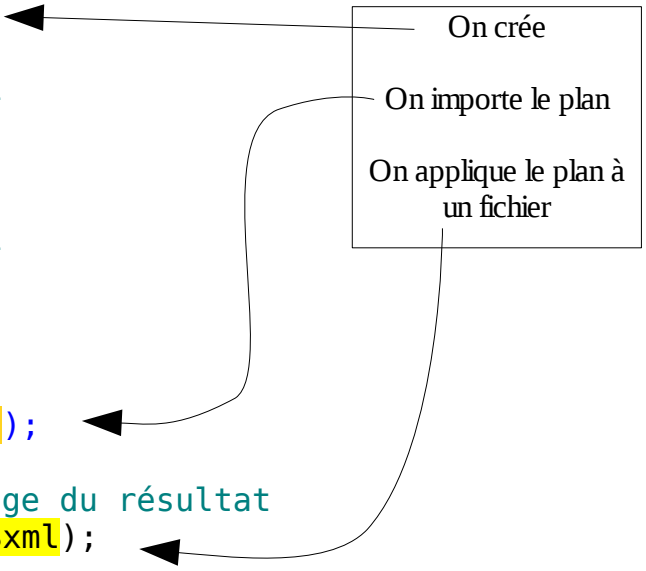
```
<!-- Affichage simple -->
<xsl:template match="/dvd">
<div>
   <h3><xsl:value-of select="item/titre"/></h3>
   <div>
      <em>Catégorie</em> : <xsl:value-of select="item/categorie"/> --
      <em>Date d'achat</em> : <xsl:value-of select="item/@date"/>
   </div>
</div>
</xsl:template>

</xsl:stylesheet>
```

Le code PHP

La transformation s'effectue très simplement. Elle se décompose en **cinq étapes** :

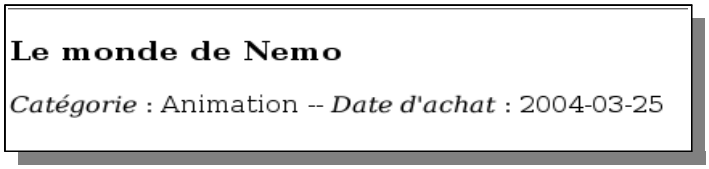
1. Nouvelle instance de la classe *XSLTProcessor*
2. Chargement du code XML grâce à la classe *domDocument* et à la méthode *domDocument :: load()* qui chargera le fichier à partir de son URL. Il est également possible de charger du code XML à partir d'une chaine, grâce à la méthode : *domDocument :: loadXML()*.
3. Chargement du fichier XSL de la même facon que précédamment.
4. Import de l'objet représentant le fichier XSL dans la classe *XSLTProcessor*, grâce à la méthode *XSLTProcessor :: importStylesheet()*.
5. Transformation et affichage du résultat à l'ecran, grâce à la méthode *XSLTProcessor :: TransformToXml()*.

```php
<?php
// Nouvelle instance
$xslt = new XSLTProcessor();

// Chargement du fichier XML
$xml = new domDocument();
$xml->load('dvd.xml');

// Chargement du fichier XSL
$xsl = new domDocument();
$xsl->load('dvd.xsl');

// Import de la feuille XSL
$xslt->importStylesheet($xsl);

// Transformation et affichage du résultat
echo $xslt->transformToXml($xml);

?>
```



Résultat de la transformation

Voila ce que donnera le résultat de la transformation :




```
<div>
    <h3>Le monde de Nemo</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-03-25
    </div>
</div>
```

Exemple 2 : Enregistrement dans un fichier

Le fichier XSL

Cette template XSLT diffère peu de la précédente, hormis le fait qu'on affichera tous les résultats, et non plus seulement le premier. On utilisera pour ce faire, une boucle *<xsl:for-each>*.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:output
    indent="yes"
    method="html"
    omit-xml-declaration="no"
    encoding="iso-8859-1"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>

<xsl:template match="/dvd">
<div>
    <!-- la boucle for-each -->
    <xsl:for-each select="item">
        <h3><xsl:value-of select="titre"/></h3>
        <div>
            <em>Catégorie</em> : <xsl:value-of select="categorie"/> --
            <em>Date d'achat</em> : <xsl:value-of select="@date"/>
        </div>
    </xsl:for-each>
</div>
</xsl:template>

</xsl:stylesheet>
```

Le code PHP XSLTProcessor::transformToURI()

Il est également possible de se servir de l'extension *SimpleXML* ...

- Nouvelle instance de la classe *XSLTProcessor*

- Import de l'objet représentant le fichier XSL dans la classe *XSLTProcessor*, grâce à *SimpleXML*.
- Transformation et enregistrement du résultat dans un fichier *save.xml*, grâce à la méthode *XSLTProcessor :: TransformToUri()*. *Uniform Resource Identifier*

uri est une courte chaîne de caractères iqui vous permet d'identifier une chose (on parle d'une "ressource")

```
<?php
// Nouvelle instance
$xslt = new XSLTProcessor();

// Import de la feuille XSL directement avec simplexml
$xslt->importStylesheet(simplexml_load_file('dvd.xsl'));

// Transformation et enregistrement du résultat dans le fichier
save.xml
// Le fichier XML est également chargé via simplexml
$xslt->transformToUri(simplexml_load_file('dvd.xml'),'save.xml');

?>
```

Résultat de la transformation

Un fichier *save.xml* sera généré et contiendra le code suivant :

```
<div>
    <h3>Le monde de Nemo</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-03-25
    </div>
    <h3>Skreck3D</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-06-06
    </div>
    <h3>Kill Bill</h3>
    <div>
        <em>Catégorie</em> : Action --
        <em>Date d'achat</em> : 2004-05-20
    </div>
</div>
```

Exemple 3 : Passage de paramètres à la template XSLT

Le fichier XSL

Dans cette template XSLT, on récupérera le paramètre *limit* venant de PHP grâce au tag <xsl:param>. Ce paramètre servira a limiter le nombre de résultats affichés par la boucle <xsl:for-each>.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:output
    indent="yes"
    method="html"
    omit-xml-declaration="no"
    encoding="iso-8859-1"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>


<!-- Récuperation du paramètre venant de PHP -->
<xsl:param name="limit"/>

<xsl:template match="/dvd">
<div>
    <!-- Limitation des résutats en  fonction de $limit -->
    <xsl:for-each select="item[ position() &lt;= $limit ]">
        <h3><xsl:value-of select="titre"/></h3>
        <div>
            <em>Catégorie</em> : <xsl:value-of select="categorie"/> --
            <em>Date d'achat</em> : <xsl:value-of select="@date"/>
        </div>
    </xsl:for-each>
</div>
</xsl:template>

</xsl:stylesheet>
```

Le code PHP

```
<?php
// Nouvelle instance
$xslt = new XSLTProcessor();

// Import de la feuille XSL directement avec simplexml
$xslt -> importStylesheet(simplexml_load_file('dvd.xsl'));

// Définition du paramètre limit
// On affichera donc que les deux premiers résultats du fichier XML
$xslt->setParameter(null, 'limit', 2 );

// Transformation et enregistrement du résultat dans le fichier save.xml
// Le fichier XML est également chargé via simplexml
$xslt -> transformToUri(simplexml_load_file('dvd.xml') , 'save.xml');
```

```
?>
```

Résultat de la transformation

Un fichier *save.xml* sera généré et contiendra le code suivant (seuls les deux premiers résultats ont étés pris en compte) :

```
<div>
    <h3>Le monde de Nemo</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-03-25
    </div>
    <h3>Skreck3D</h3>
    <div>
   <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-06-06
    </div>
</div>
```

Exemple 4 : Import du résultat dans DOMXML

Le fichier XSL

Dans cette template XSLT, nous ajouterons juste **un tri par date, grâce à <xsl:sort>.**

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:output
    indent="yes"
    method="html"
    omit-xml-declaration="no"
    encoding="iso-8859-1"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd"/>

<!-- Récuperation du paramètre venant de PHP -->
<xsl:param name="limit"/>

<xsl:template match="/dvd">
<div>
    <!-- Limitation des résutats en  fonction de $limit -->
    <xsl:for-each select="item[ position() &lt;= $limit ]">
    <!-- TRI par date DESC -->
    <xsl:sort select="@date" data-type="text" order="descending"/>
        <h3><xsl:value-of select="titre"/></h3>
        <div>
            <em>Catégorie</em> : <xsl:value-of
select="categorie"/> --
            <em>Date d'achat</em> : <xsl:value-of select="@date"/>
        </div>
    </xsl:for-each>
</div>
</xsl:template>

</xsl:stylesheet>
```

Le code PHP

Dans ce code PHP, nous allons importer le résultat de la transformation XSLT dans DOMXML grâce à la méthode *XSLTProcessor :: transformToDoc()*, rajouter un petit texte de fin (à l'intérieur d'une balise <p>), et afficher le résultat.

```
<?php
// Nouvelle instance
$xslt = new XSLTProcessor();

// Import de la feuille XSL directement avec simplexml
$xslt -> importStylesheet(simplexml_load_file('dvd.xsl'));

// Définition du paramètre limit
// On affichera donc que les deux premiers résultats du fichier XML
$xslt -> setParameter(null, 'limit', 2 );

// Import du résultat de la transformation dans DOMXML
$dom = $xslt -> transformToDoc(simplexml_load_file('dvd.xml'));

// On rajoute un tag <p>
$item = $dom -> createElement('p');
$text = $dom -> createTextNode('Voila le contenu de ma
DVDtheque');
$item -> appendChild($text);
$dom -> documentElement -> appendChild($item);

// Affichage du résultat
echo $dom -> saveXML();
?>
```

Résultat de la transformation

Voilà ce qui sera affiché à l'ecran, un petit texte a été rajouté à la fin, et le résultat est trié par date DESC :

```
<div>
    <h3>Skreck3D</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-06-06
    </div>
    <h3>Le monde de Nemo</h3>
    <div>
        <em>Catégorie</em> : Animation --
        <em>Date d'achat</em> : 2004-03-25
    </div>
    <p>Voila le contenu de ma DVDtheque</p>
</div>
```

Exemple 5 : Utilisation de fonctions PHP

Dans cet exemple, nous allons définir une fonction PHP utilisateur, *dateFr()*, et l'utiliser à l'intérieur de la template XSLT.

Le fichier XSL

Nous utiliserons donc la fonction *dateFr()* pour transformer les dates au format EN en un format FR. ***N'oubliez pas de définir le namespace pour php (xmlns:php="http://php.net/xsl").***

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:php="http://php.net/xsl"
    version="1.0">
<xsl:output
    indent="yes"
    method="html"
    omit-xml-declaration="no"
    encoding="iso-8859-1"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd"/>

<xsl:param name="limit"/>

<xsl:template match="/dvd">
<div>
    <xsl:for-each select="item[ position() &lt;= $limit ]">
    <xsl:sort select="@date" data-type="text" order="descending"/>
        <h3><xsl:value-of select="titre"/></h3>
        <div>
            <em>Catégorie</em> : <xsl:value-of
select="categorie"/> --
  <!-- Utilisation de la fonction utilisateur PHP dateFr() -->
            <em>Date d'achat</em> :
<xsl:value-of select="php:functionString('dateFr', @date)"/>
        </div>
    </xsl:for-each>
</div>
</xsl:template>

</xsl:stylesheet>
```

Le code PHP

Dans ce code PHP nous allons **définir une fonction *dateFr()*** qui transfomera les dates au format YYYY-MM-DD en DD-MM-YYYY, et l'importer dans XSLT.

```
<?php
// Définition de notre fonction dateFr()
function dateFr($dateEn) {
    $tD = explode('-', $dateEn);
    return $tD[2].'-'.$tD[1].'-'.$tD[0];
}


// Nouvelle instance
$xslt = new XSLTProcessor();

// Import de la feuille XSL directement avec simplexml
$xslt -> importStylesheet(simplexml_load_file('dvd.xsl'));

// Définition du paramètre limit
// On affichera donc que les deux premiers résultats du fichier XML
$xslt -> setParameter(null, 'limit', 2 );

// Enregistrement des fonctions PHP
$xslt -> registerPhpFunctions();

// Transformation et Affichage du résultat
echo  $xslt -> transformToXml(simplexml_load_file('dvd.xml') );

?>
```

Résultat de la transformation

Les dates au format EN ont étés transformées au format FR :

```
<div>
   <h3>Skreck3D</h3>
   <div>
      <em>Catégorie</em> : Animation --
      <em>Date d'achat</em> : 06-06-2004
   </div>
   <h3>Le monde de Nemo</h3>
   <div>
      <em>Catégorie</em> : Animation --
      <em>Date d'achat</em> : 25-03-2004
   </div>
</div>
```