La transformation XSLT avec PHP

Réalisé par Nabil ADOUI, membre de l'équipe support technique 4D

Sommaire

Résumé :

Dans la présente note technique, nous allons vous présenter l'extension XSL(PHP) qui permet de réaliser des transformations XSLT. Cette extension va remplacer les trois commandes 4D *XSLT APPLIQUER TRANSFORMATION*, *XSLT FIXER PARAMETRE* et *XSLT LIRE ERREUR*.

Introduction

La transformation XSLT est une norme définie par W3C. Ce langage transforme un fichier XML vers un autre document de type XML, HTML ou n'importe quel autre type reconnu par le navigateur pour donner une présentation visuelle plus significative au document original.

Eléments importants :

La transformation XSLT prend en entrée deux éléments :

- Un document XML qu'on veut transformer
- Une feuille de style XSL (XML style sheets). C'est un fichier XML décrivant la présentation du contenu de notre document. Il doit être déclaré de la façon suivante :

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

Ces deux fichiers seront utilisés par un processeur XSLT basé sur la bibliothèque PHP XSLT qui va remplacer le processeur C++ XALAN utilisé par 4D actuellement.

Le résultat est un document XML qui inclut le contenu du document original représenté selon la feuille de style indiquée.

La bibliothèque PHP XSL

PHP XSL est une implémentation de la norme XSLT qui utilise la bibliothèque libxslt (XSLT C library for GNOME).

Elle contient une seule classe XSLTProcessor avec une dizaine de méthodes figurant ci-dessous :

XSLTProcessor — La classe XSLTProcessor

- XSLTProcessor::__construct — Créer un nouveau objet XSLTProcessor
- XSLTProcessor::getParameter — Retourner la valeur du parametre
- XsltProcessor::getSecurityPrefs — Retourner les préférences de sécurité
- XSLTProcessor::hasExsltSupport — Déterminer si PHP supporte XSLT
- XSLTProcessor::importStylesheet — Importer des feuille de style
- XSLTProcessor::registerPHPFunctions — Permettre d'utiliser les fonctions de PHP comme des fonctions de XSLT
- XSLTProcessor::removeParameter — Supprimer parametre
- XSLTProcessor::setParameter — Fixer la valeur d'un parametre
- XSLTProcessor::setProfiling — Fixer un fichier de sortie pour un profile
- XsltProcessor::setSecurityPrefs — Fixer les parametres de sécurité
- XSLTProcessor::transformToDoc — Transformer à un document DOM
- XSLTProcessor::transformToUri — Transformer à un URI
- XSLTProcessor::transformToXML — Transformer à un XML

Dans la suite de cette note technique, nous allons nous concentrer sur 5 méthodes seulement :

- XSLTProcessor::__construct
- XSLTProcessor::setParameter
- XSLTProcessor::importStylesheet
- XSLTProcessor::transformToXML
- XSLTProcessor::transformToURI

L'API de PHP XSL

XSLTProcessor::__construct

Cette méthode est tout simplement le constructeur de la classe XSLTProcessor.
Elle permet de créer un nouveau objet de type XSLTProcess.
Elle n'accepte aucun paramètre et ne retourne aucune valeur.

- Signature :

  XSLTProcessor::__construct (void)

- Exemple d'appel :

  ```
  $xsl = new XSLTProcessor;
  ```

XSLTProcessor::setParameter

Cette méthode stocke un ou plusieurs paramètres à utiliser dans les transformations qui suivent.
Vous pouvez appeler ce paramètre dans votre fichier XSL à l'aide du symbole « $ » suivi par le nom du paramètre.

- Signatures :

  bool XSLTProcessor::setParameter (string $namespace , string $nom , string $valeur)
  bool XSLTProcessor::setParameter (string $namespace , array $options)

  - $namespace : URI de l'espace de nom du paramètre XSLT,
  - $nom : le nom local du paramètre XSLT
  - $valeur : La nouvelle valeur du paramètre XSLT
  - $options : Un tableau de paires nom => valeur

- Exemple d'appel :

  ```
  $xsl = new XSLTProcessor();
  $xsl->setParameter('', 'param1','valeur1');
  ```

  On peut aussi passer un tableau de paramètres :

```
$params['param1'] = 'valeur1' ;
$params['param2'] = 'valeur2' ;
$xsl->etParameter('http://www.w3.org/1999/XSL/Transform', $params);
```

XSLTProcessor::importStylesheet

Cette méthode importe la feuille de style représentée par le fichier XSL qu'on veut appliquer durant la transformation.

- Signature :
  void XSLTProcessor::importStylesheet (object $stylesheet)
- Exemple :

```
$xsl = new DOMDocument;
$xsl->load('xslt2.xsl');
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl);
```

XSLTProcessor::transformToXML

TransformToXML convertit le document source vers un texte en appliquant le fichier XSL importé utilisant la méthode importStylesheet.

- Signature :
  string XSLTProcessor::transformToXML (DOMDocument $doc)

XSLTProcessor::transformToURI

Cette méthode transforme le document source vers un fichier qui sera stocké vers l'URI spécifié comme paramètre de la méthode, après l'application de la feuille de style choisie.

- Signature :
  int XSLTProcessor::transformToURI (DOMDocument $doc , string $uri)

Démonstration

Nous allons prendre comme exemple, le document XML suivant qui contient des titres de chansons et leurs artistes et les années de sortie :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <cd>
        <titre>Empire Burlesque</titre>
        <artiste>Bob Dylan</artiste>
        <annee>1985</annee>
    </cd>
    <cd>
        <titre>Hide your heart</titre>
        <artiste>Bonnie Tyler</artiste>
        <annee>1988</annee>
    </cd>
    <cd>
        <titre>Greatest Hits</titre>
        <artiste>Dolly Parton</artiste>
        <annee>1982</annee>
    </cd>
    <cd>
        <titre>Still got the blues</titre>
        <artiste>Gary Moore</artiste>
        <annee>1990</annee>
    </cd>
    <cd>
        <titre>Eros</titre>
        <artiste>Eros Ramazzotti</artiste>
        <annee>1997</annee>
    </cd>
    <cd>
        <titre>One night only</titre>
        <artiste>Bee Gees</artiste>
        <annee>1998</annee>
    </cd>
    <cd>
        <titre>Sylvias Mother</titre>
        <artiste>Dr.Hook</artiste>
        <annee>1973</annee>
    </cd>
    <cd>
        <titre>Maggie May</titre>
        <artiste>Rod Stewart</artiste>
        <annee>1990</annee>
    </cd>
    <cd>
        <titre>Romanza</titre>
        <artiste>Andrea Bocelli</artiste>
        <annee>1996</annee>
    </cd>
</catalog>
```

Puis nous allons appliquer une transformation XSL avec PHP XSL selon la feuille de style suivante :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" version="4.0"
encoding="UTF-8" indent="yes"/>

<xsl:template match="/">
  <html>
  <body>
    <h2>Ma collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>titre</th>
        <th>artiste</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <xsl:choose>
        <xsl:when test="annee > $param1">
            <td bgcolor="red"><xsl:value-of select="titre" /></td>
            <td bgcolor="red"><xsl:value-of select="artiste" /></td>
        </xsl:when>
        <xsl:otherwise>
            <td><xsl:value-of select="titre" /></td>
            <td><xsl:value-of select="artiste" /></td>
        </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Cette feuille de style va regrouper les données du fichier XML dans un tableau à deux colonnes.
Pour rendre la tâche plus intéressante, nous allons colorer les lignes qui ont une année supérieure à un paramètre (***$param1***) que nous allons fournir au niveau de notre code PHP.

Voici alors notre code PHP :
Nous supposons dans cet exemple que les fichiers XML et XSL se trouvent dans le même répertoire que le fichier source PHP, sinon il faut fournir le chemin complet.

```
<?php
$xml = new DOMDocument;
$xml->load('xml_file.xml'); //chargement du document XML
$xsl = new DOMDocument;
$xsl->load('xsl_file.xsl'); // chargement de la feuille du style
$proc = new XSLTProcessor;  // Création du processeur XSLT
$proc->importStyleSheet($xsl); // attachement des règles XSL
$proc->setParameter('','param1','1996'); //Insertion du paramètre $param1
echo $proc->transformToXML($xml); // Application de la transformation
?>
```

Au niveau de 4D, nous allons exécuter ce code PHP avec les commandes suivantes :

```
$filePath:=Dossier 4D(Dossier Resources courant)+"php.php"
$isOK:=PHP Executer($filePath;"";$res)
PHP LIRE REPONSE
COMPLETE(stdOut;libellesErr;valeursErr;chpsEnteteHttp;valeursEnteteHttp)
Si ($isok)
        vHtml:=$res
Fin de si
```

Le résultat de la transformation sera alors stocké dans la variable vHtml.

Après l'exécution de cette méthode 4D, vHtml va contenir le code HTML suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">\n<html><body>\n<h2>Ma collection</h2>\n<table
border="1">\n<tr bgcolor="#9acd32">\n<th>titre</th>\n<th>artiste</th>\n</tr>\n<tr>\n<td>Empire
Burlesque</td>\n<td>Bob Dylan</td>\n</tr>\n<tr>\n<td>Hide your heart</td>\n<td>Bonnie
Tyler</td>\n</tr>\n<tr>\n<td>Greatest Hits</td>\n<td>Dolly Parton</td>\n</tr>\n<tr>\n<td>Still got
the blues</td>\n<td>Gary Moore</td>\n</tr>\n<tr>\n<td bgcolor="red">Eros</td>\n<td
bgcolor="red">Eros Ramazzotti</td>\n</tr>\n<tr>\n<td bgcolor="red">One night only</td>\n<td
bgcolor="red">Bee Gees</td>\n</tr>\n<tr>\n<td>Sylvias
Mother</td>\n<td>Dr.Hook</td>\n</tr>\n<tr>\n<td>Maggie May</td>\n<td>Rod
Stewart</td>\n</tr>\n<tr>\n<td>Romanza</td>\n<td>Andrea
```

```php
<?php
$xml = new DOMDocument;
$xml->load('xml_file.xml');
$xsl = new DOMDocument;
$xsl->load('xsl_file.xsl');
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl);
$proc->setParameter('','param1','1996');
echo $proc->transformToURI($xml,'out.html');
?>
```

Le résultat alors sera enregistré dans le fichier « out.html » à côté du fichier code PHP, ou si vous voulez, vous pouvez mentionner un autre chemin.

Voici un aperçu du résultat :

Ma collection

titre	artiste
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli

Gestion des erreurs

Au cas où vous rencontrez une erreur pendant la transformation, le message d'erreur sera copié dans la variable valeursErr que vous avez passé dans la commande **PHP LIRE REPONSE COMPLETE**.

Conclusion

La bibliothèque XSL de PHP est un excellent outil qui peut remplacer les commandes *XSLT APPLIQUER TRANSFORMATION*, *XSLT FIXER PARAMETRE* et *XSLT LIRE ERREUR* après leur suppression. Cette bibliothèque offre une API riche qui vous permet de réaliser toutes les opérations voulues pendant vos transformations XSL.