

Internationalisation (i18n) en PHP avec XML

par Vincent Flauder ([Internationalisation \(i18n\) en PHP avec XML](#))

Date de publication : 15 Octobre 2007

Dernière mise à jour :

Cet article a pour but de proposer une solution simple pour introduire plusieurs langues dans vos sites web, avec une interface administrateur.

- Niveau : néophyte à intermédiaire/expert
- Connaissances abordées :
- XHTML : Les formulaires
- PHP : PHP5 et les classes
- XML : DTD et XML

- I - Introduction
- II - Le DTD (Document Type Definition)
- III - Le fichier XML
- IV - La base de données SQL
- V - La classe PHP : XMLEngine
- VI - Utilisation de la classe
 - A - Extraction des données
 - B - Ajout d'éléments
 - C - Modification des éléments (traduction)
 - D - Suppression des éléments
 - E - Disponibilité des langues
- VII - Sources


I - Introduction

Le principe du multi-langages est de modifier le contenu textuel de tous les éléments d'une page HTML en fonction de la langue choisie par l'utilisateur. Pour ce faire il faut stocker ces valeurs dans des fichiers externes, ces derniers seront au format XML et leur syntaxe contrôlée avec un DTD (Document Type Definition). Pour pouvoir manipuler ces fichiers XML, PHP nous offre un panel complet d'outils : les fonctions DOM.

L'utilisation d'une base de données de type MySQL ici permet de sauvegarder les langues utilisées ainsi qu'une variable afin de savoir si la langue est ouverte aux utilisateurs ou non.

Pour interagir avec la base de données j'ai utilisé une classe que j'ai moi même crée (`SQLManager.php`) ne sera pas détaillée dans cet article, mais elle est très documentée et vous pourrez apprendre à vous en servir si vous le voulez avec ses commentaires.

En résumé, la classe permet de se connecter au serveur et à une base de données, surtout elle génère les requêtes SQL à notre place et nous renvoie les résultats sous forme de tableau.

 *Ne pas confondre les fonctions DOM et les fonctions DOM XML, les fonctions DOM XML ne sont pas implémentées en PHP 5.*

II - Le DTD (Document Type Definition)

Le DTD sert à créer un modèle pour les documents XML, une hiérarchie stricte à respecter pour l'imbrication des éléments et leurs attributs. Lorsque l'on crée un document XML, on indique l'emplacement du DTD afin de "valider" le document. C'est nécessaire pour utiliser certaines fonctions DOM et permet d'homogénéiser tous les documents.

Le DTD utilise un langage propre à lui, mais simple à comprendre. Dans le fichier seront définis le nom des balises nécessaires ainsi que leurs attributs, en précisant si ces attributs et balises sont obligatoires ou facultatives.

Code DTD (xmldoctype.dtd)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT language (xmldata*) >
<!--
    Définition du premier élément 'language' (l'élément racine du fichier)
    dans lequel on peut placer des éléments (xmldata)
    * : l'étoile indique 0 ou plusieurs éléments
-->

<!ELEMENT xmldata (translation+) >
<!ATTLIST xmldata id ID #REQUIRED >
<!--
    Définition de l'élément 'xmldata', on peut y placer des éléments (translation),
    puis l'attribut 'id' est de type ID et est requis(obligatoire), le '+' signifie
    qu'il faut au moins 1 élément.
-->

<!ELEMENT translation (#PCDATA) >
<!ATTLIST translation lang CDATA #REQUIRED >
<!--
    Pour finir voici la définition d'un élément 'translation' qui contient
    des données (#PCDATA), ce qui signifie que c'est une grande chaîne de
    caractères pour simplifier. C'est cet élément qui contiendra nos textes.
    Il possède un attribut 'lang' qui est aussi requis.
-->
```

III - Le fichier XML

Les fichiers XML seront notre base de données pour les textes du site, en suivant les définitions créées auparavant, il sera alors simple de concevoir la 'base de données' (en mode Design sous Eclipse c'est très pratique et simple).

Code XML (exemple de website.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE language SYSTEM "xmldoctype.dtd" >
<!-- On doit définir l'emplacement du DTD ainsi que l'élément utilisé comme racine-->

<language>

  <xmldata id="webPageTitle">
    <translation lang="fr">Démonstration de XMLEngine</translation>
    <translation lang="en">XMLEngine demonstration</translation>
  </xmldata>

</language>
```


IV - La base de données SQL

Pour notre base de données c'est très simple, on a juste besoin d'une table avec deux champs : l'un contenant le tag des langues utilisées, l'autre étant un booléen qui permet de savoir si la langue est ouverte aux utilisateurs ou non.

Table : Languages		
Nom du champ	Valeur	Description
Language	VARCHAR(2)	Chaîne de caractères contenant le tag de la langue
IsAvailable	BOOL (TinyInt(1))	Variable de contrôle pour l'accessibilité de la langue aux utilisateurs

V - La classe PHP : XMLEngine

Voici la partie centrale de notre solution, c'est cette classe qui nous permettra de manipuler les données des fichiers XML. La classe possède plusieurs fonctions permettant d'aider à l'administration via un panneau administrateur.

 Les types soulignés dans ce tableau ne sont là qu'à titre indicatif. On ne doit pas les recopier dans le fichier source.

Définition des attributs de la classe :


Attributs	
Nom	Définition
<i>private</i> <u>DOMDocument</u> domFile	Attribut dans lequel se trouve l'instance de la classe DOMDocument.
<i>private</i> <u>string</u> currentLanguage	Attribut dans lequel on sauve la langue utilisée lors de la manipulation des fichiers XML.
<i>private</i> <u>string</u> currentFile	Attribut où l'on stocke le chemin du fichier XML ouvert et en cours d'utilisation.
<i>private</i> <u>fileID</u> fileLock	Attribut permettant de bloquer l'accès au fichier lors des modifications.

Définition des méthodes (fonctions) de la classe :

Méthodes	
Nom	Définition
<i>public</i> <u>__construct</u> (<u>string</u> \$filename, <u>string</u> \$language)	Constructeur de la classe, prend en argument le chemin du fichier XML à ouvrir, \$filename, avec la langue \$language à utiliser.
<i>public</i> <u>string</u> getCurrentFile ()	Méthode permettant de récupérer le nom du fichier en cours.
<i>public</i> <u>string</u> getItemValue (<u>string</u> \$itemID)	Méthode permettant d'extraire le contenu textuel d'une balise dont l'identifiant \$itemID est passé en argument.
<i>public</i> <u>string</u> [] getItemNodeList ()	Méthode permettant de récupérer tous les identifiants des balises <i>xml:node</i> du fichier XML.
<i>public</i> changeFile (<u>string</u> \$newFile)	Méthode permettant de charger un nouveau fichier XML.
<i>public</i> changeLanguage (<u>string</u> \$newLanguage)	Méthode permettant de changer la langue utilisée.
<i>public</i> <u>bool</u> addXmlElement (<u>string</u> \$itemID,	Méthode permettant de créer un nouvel élément <i>xml:node</i> dont l'identifiant sera \$itemID avec un enfant <i>translation</i> dans la langue \$lang et avec le texte \$itemDefaultValue.

<code>string \$lang,</code> <code>string \$itemDefaultValue)</code>	
<code>public bool</code> modifyXmlElement(<code>string \$itemID,</code> <code>string \$lang,</code> <code>string \$newContent)</code>	Méthode permettant de modifier le contenu textuel d'un élément dont l'ID est \$itemID, dans la langue \$lang, et le nouveau texte sera \$newContent.
<code>public bool</code> removeXmlElement(<code>string \$itemID,</code> <code>string \$lang)</code>	Méthode permettant de supprimer un élément <i>translation</i> dont la langue est \$lang, ceci dans l'élément <i>xmldata</i> qui a l'ID \$itemID.
<code>public bool</code> completelyRemoveXmlElement(<code>string \$itemID)</code>	Méthode permettant de supprimer un élément <i>xmldata</i> dont l'ID est \$itemID.
<code>public bool</code> isEmptyXml(<code>string \$itemID,</code> <code>string \$lang)</code>	Méthode permettant de vérifier qu'un élément <i>translation</i> est bien rempli dans la langue \$lang.
<code>public</code> addLanguageToFile(<code>string \$newLanguage)</code>	Méthode permettant d'ajouter une nouvelle langue \$newLanguage au fichier. Cette fonction crée un élément <i>translation</i> avec la nouvelle langue dans chaque élément <i>xmldata</i> s'il n'y en a pas.
<code>public bool</code> checkIntegrity(<code>string \$lang)</code>	Méthode permettant de vérifier que tous les éléments <i>xmldata</i> possèdent des éléments <i>translation</i> dans la langue \$lang et qu'ils soient remplis.
<code>public string</code> __get(<code>string \$itemID)</code>	Méthode magique de PHP qui sera exécutée lorsque l'on voudra lire un attribut de la classe. En utilisant les IDs des éléments <i>xmldata</i> comme attributs de classe, cela permet de retourner les contenus textuels des balises plus facilement.
<code>private</code> saveFile()	Méthode permettant de sauvegarder le fichier XML de manière sécurisée.

VI - Utilisation de la classe

 Dans les pages de code on ignorera les standards HTML afin d'aller à l'essentiel et d'éviter de se perdre, seulement les éléments nécessaires au bon fonctionnement de la solution seront présents. Aussi aucun style CSS n'a été défini, les styles sont insérés directement dans l'attribut STYLE pour les démonstrations présentes dans cet article.

A - Extraction des données

Pour commencer nous allons simplement utiliser la classe pour extraire des données :

Code PHP (extract.php)

```
<?php
session_start();

//On ajoute le fichier de classe.
require_once 'include/XMLEngine.php';

/* On va utiliser une variable de session pour la langue de l'utilisateur.
 * Si la variable n'existe pas on la crée. */
if ( !isset( $_SESSION['Language'] ) )
{
    $_SESSION['Language'] = 'fr'; //Français par défaut.
}

//Création de l'instance de notre classe avec le fichier XML à ouvrir.
$xml = new XMLEngine( 'xml/website.xml', $_SESSION['Language'] );

/* Grâce à la fonction __get(), cette ligne suffit à afficher le contenu
 * de l'élément XMLDATA d'identifiant 'webPageName' en langue française.*/

echo $xml->webPageName;
?>
```

Il n'y a que cela pour l'extraction des données, les autres fonctions sont principalement conçues pour créer l'administration au travers d'une interface web.

B - Ajout d'éléments

Dans cette section nous allons voir comment ajouter des éléments, en créant dans un premier temps un formulaire HTML pour entrer les éléments nécessaires, puis la partie PHP qui traitera ce formulaire pour ajouter l'élément et afficher un message javascript, le tout en utilisant à chaque fois notre objet XML pour récupérer les textes du site :

Code PHP (addForm.php)

```
<?php
/*
 * On reprend le même début qu'avant :
 */
session_start();
require_once 'include/XMLEngine.php';
if ( !isset( $_SESSION['Language'] ) )
```

Code PHP (addForm.php)

```

{
    $_SESSION['Language'] = 'fr'; //Français par défaut.
}
$xml = new XMLEngine( 'xml/website.xml', $_SESSION['Language'] );

//On inclut la classe SQLManager et créer un objet.
require_once 'include/SQLManager.php';
$sql = new SQLManager( 'localhost', 'root', '', 'Test' );

/* Création du formulaire de saisie : */ ?>
<form action="addForm.php" method="post">

<!-- On place un tableau dans le formulaire pour le mettre
en forme -->
<table border="1">

<!-- Titre -->
<tr>
<td colspan="2">
<?php echo $xml->webPageName;?>
</td>
</tr>

<!-- Saisie de l ID de la balise -->
<tr>
<td>
<?php echo $xml->inputID;?>
</td>
<td>
<input type="text" name="itemId" />
</td>
</tr>

<!-- Saisie du texte pour le nouveau contenu -->
<tr>
<td>
<?php echo $xml->inputData;?>
</td>
<td>
<textarea cols="20" rows="5" name="itemData"></textarea>
</td>
</tr>

<!-- Création de la liste de choix des langues -->
<tr>
<td>
<?php echo $xml->inputLang;?>
</td>
<td><?php
//On récupère toutes les langues de la base de données SQL
$langs = $sql->selectQuery( '*', 'Languages', 0, '', '', '' );

//Puis on créer la liste ?>
<select name="itemLanguage" style="width:100%;"><?php
foreach ( $langs as $language )
{?>
<option value="<?php echo $language['Language'];?>"><?php echo $xml->{
$language['Language']. 'Lang' };?></option><?php
}?>
</select>
</td>
</tr>

<!-- Bouton d envoi pour créer une balise XML -->
<tr>
<td colspan="2">
<input style="width:100%;text-align:center;" type="submit"
    
```

Code PHP (addForm.php)

```

        name="submitCreateItem" value="<?php echo $xml->submitCreate;?>" />
    </td>
</tr>
</table>
</form> <?php
/*
 * Traitement du formulaire lorsqu'il est soumis
 */
if ( isset( $_POST['submitCreateItem'] ) )
{
    /*
     * On fait alors appel à la fonction addXmlElement avec les
     * variables du formulaire, comme la fonction renvoi un boolean
     * on peut l inclure directement dans la condition du IF :
     */
    if ( $xml->addXmlElement( $_POST['itemId'], $_POST['itemLanguage'], $_POST['itemData'] ) )
    {
        /* Réussite de l'ajout de l'élément
         * On affiche un petit message javascript, toujours avec notre classe.
         */
        <?>
        <script type="text/javascript">
        <!--
            alert("<?php echo $xml->createXmlDone;?>");
        -->
        </script><?php
    }
    else
    {
        <?>
        <script type="text/javascript">
        <!--
            alert("<?php echo $xml->createXmlError;?>");
        -->
        </script><?php
    }
}
?>

```

C - Modification des éléments (traduction)

Avec ceci on peut déjà créer des éléments dans les fichiers XML. Maintenant nous allons modifier les éléments dans le but de traduire ceux qui n'y sont pas. On va pour cela générer un formulaire HTML pour modifier les balises selon la langue de traduction choisie :

Code PHP (traduceForm.php)

```

<?php
/*
 * On fait une nouvelle page,
 * On reprend le même début qu'avant :
 */
session_start();
require_once 'include/XMLEngine.php';

//On inclut la classe SQLManager et créer un objet.
require_once 'include/SQLManager.php';
$sql = new SQLManager( 'localhost', 'root', '', 'Test' );

if ( !isset( $_SESSION['Language'] ) )
{
    $_SESSION['Language'] = 'fr'; //Français par défaut.
}

$xml = new XMLEngine( 'xml/website.xml', $_SESSION['Language'] );

```

Code PHP (traduceForm.php)

```

/*
 * On inclut ici le traitement du formulaire qui sera vu ensuite.
 * Il faut voir le 'require_once' comme un copier/coller, donc le
 * fichier 'traduceTraitement.php' est uniquement à inclure.
 */
require_once 'traduceFormTraitement.php';

/* Début du formulaire d'administration : */ ?>
<div style="height:300px;overflow:scroll;" align="center"><?php
/*
 * Pour commencer il faut créer un formulaire pour choisir la
 * langue dans laquelle traduire, on affichera ce formulaire que
 * si aucun autre n'a été envoyé.
 */
if ( !isset( $_POST['submitLangToTraduce'] ) && !isset( $_POST['submitTraduceFile'] ) )
{?>
<form action="traduceForm.php" method="post">
<table border="1">
<tr>
<td><?php
    echo $xml->foreignLang;?>
</td>
</tr>

<!-- On crée la liste de langues l'ol -->
<tr>
<td align="center"><?php
    //On récupère toutes les langues de la base de données SQL
    $langs = $sql->selectQuery( '*', 'Languages', 0, '', '', '' );

    /* Puis on créer la liste */ ?>
    <select name="langToTraduce" style="width:100%;"><?php
        foreach ( $langs as $language )
        {?>
            <option value="<?php echo $language['Language'];?>"><?php echo $xml->{
                $language['Language'].'Lang' };?></option><?php
            }?>
        </select>
    </td>
</tr>

<tr>
<td colspan="2">
    <input type="submit" name="submitLangToTraduce" value="<?php echo $xml->submitChooseLang;?>" />
</td>
</tr>
</table>
</form><?php
} //Fin du formulaire de choix des langues

/*
 * Une fois la langue choisie ou le traitement effectué on affiche le formulaire
 * avec les éléments du fichier website.xml
 */
if ( isset( $_POST['submitLangToTraduce'] ) || isset( $_POST['submitTraduceFile'] ) )
{
/*On créer un nouvel objet xml pour la langue à traduire
$xmlTraductor = new XMLEngine( 'xml/website.xml', $_POST['langToTraduce'] );*/ ?>

<form action="traduceForm.php" method="post">
<table border="1">

<!-- Titre -->
<tr>
<td colspan="3">
    <?php echo $xml->webPageName;?>
</td>

```

Code PHP (traduceForm.php)

```

</tr>

<!-- Cellules d en-tête -->
<tr>
<td>
    <?php echo $xml->frLang;?>
</td>
<td>
    <?php
    /*
    * Si on utilise toujours la même syntaxe pour les identifiants xml
    * qui représente le nom d'une langue, on peut créer le nom de la variable
    * à l'aide des variables dynamiques sans la connaître, on va ainsi créer
    * le nom de la variable sous forme de chaîne de caractères avant de
    * l'utiliser. Ici on aura frLang ou enLang.
    */
    echo $xml->{ $_POST['langToTraduce'].'Lang' };?>
</td>
<td>
    <?php echo $xml->newContent;?>
</td>
</tr><?php

//On récupère les identifiants des balises :
$idTab = $xml->getItemNodeList();
/*
* On parcourt le tableaux des identifiants en créant à chaque
* fois une ligne qui contient la version française en lecture seule (langue maternelle)
* et à droite la langue à traduire, puis tout à droite, le champs pour modifier
* une valeur si l'on veut, ou remplir un champ vide.
*/
foreach ( $idTab as $cell )
{?>
<tr>
<td>
    <input type="text" readonly="readonly" value="<?php echo $xml->$cell;?>" style="width:250px;"
/>
</td>
<td>
    <input type="text" readonly="readonly" value="<?php echo $xmlTractor->$cell;?>"
style="width:250px;" />
</td>

<!-- Il faut donner un nom au champ que l on remplira, ce nom sera celui de l ID de
la balise XML -->
<td>
    <textarea name="<?php echo $cell;?>" rows="1" cols="20"></textarea>
</td>
</tr><?php
}

/* Puis un bouton d'envoi pour effectuer les modifications : */?>
<tr>
<td colspan="3">
    <input type="hidden" name="langToTraduce" value="<?php echo $_POST['langToTraduce'];?>" />
    <input type="submit" name="submitTraduceFile" value="<?php echo $xml->submitTraduce;?>"
style="width:100%;text-align:center;" />
</td>
</tr>
</table>
</form><?php
} //Fin du formulaire de traduction

//Ici on vérifie que tout s'est bien déroulé pour afficher un message JScript.
if ( isset( $_POST['submitTraduceFile'] ) )
{
    //On vérifie notre variable de contrôle et affiche le bon message.

```

Code PHP (traduceForm.php)

```

if ( $allSaveGood )
{
    /* Succès */ ?>
    <script type="text/javascript">
        alert("<?php echo $xml->modificationDone;?>");
    </script><?php
}
else
{
    /* Echec */ ?>
    ?>
    <script type="text/javascript">
        alert("<?php echo $xml->modificationError;?>");
    </script><?php
}
}??>
</div>
    
```

Ce code va générer un formulaire HTML qui permet de visualiser le contenu des balises dans les deux langues et modifier à souhait, les champs vides ne seront pas traités.



Attention : *il ne faut pas donner un identifiant XML pour le bouton SUBMIT qui soit identique à son attribut NAME, il y aurait alors des conflits dans le tableau \$_POST entre le nom du bouton et le champ TEXTAREA comportant le même nom.*

Une fois ce formulaire soumis, il faut le traiter pour modifier les éléments XML dont les champs du formulaire ne sont pas vide, ce fichier sera inclus au début de *traduceForm.php* :

Code PHP (traduceFormTreatment.php)

```

<?php
/*
 * On détecte le lancement du formulaire de traduction pour modifier
 * le fichier XML avant de le réutiliser.
 */
if ( isset( $_POST['submitTraduceFile'] ) )
{
    /*
     * On supprime la variable du bouton SUBMIT dans le tableau $_POST, et ce pour
     * pouvoir utiliser le tableau $_POST qui ne contiendra QUE les identifiants
     * des éléments XML. Si vous ajoutez d'autres balises dans le formulaire, prenez garde
     * à supprimer ces variables du tableau $_POST ici.
     */
    unset($_POST['submitTraduceFile']);

    //On récupère la langue avant de la supprimer du tableau $_POST.
    $langToTraduce = $_POST['langToTraduce'];
    unset( $_POST['langToTraduce'] );

    /*
     * On lance la fonction ajoutant une langue dans le fichier XML.
     * Cette fonction est intelligente et ne créera pas de doublon, si tous les
     * éléments existent déjà elle ne modifiera pas les données.
     */
    $xml->addLanguageToFile( $langToTraduce );

    //Variable permettant de savoir si tout s'est bien passé.
    $allSaveGood = true;

    //Maintenant on parcourt le tableau et modifie en boucle :
    foreach ( $_POST as $xmlID => $newValue )
    
```

Code PHP (traduceFormTreatment.php)

```
{
/*
 * Si vous avez des difficultés avec le foreach :
 * Ici pour chaque case du tableau, on récupère le nom de la case
 * avec $xmlID (identifiant XML), et sa valeur avec $newValue (le nouveau texte)
 *
 */
//Si le champ n'est pas vide.
if ( $newValue != '' )
{
/*
 * On modifie l'élément XML en utilisant la valeur de retour de la
 * fonction qui est un booléen, si ce dernier est faux alors notre
 * variable de contrôle $allSaveGood sera aussi fausse.
 */
if ( !$xml->modifyXMLElement( $xmlID, $langToTraduce, $newValue ) )
{
    $allSaveGood = false;
}
}
}

/*
 * On doit ré-insérer ces deux valeurs dans le tableau $_POST pour que la suite
 * du fichier se déroule sans problèmes.
 */

$_POST['langToTraduce'] = $langToTraduce;//Langue à traduire.
$_POST['submitTraduceFile'] = true;//Comme quoi le formulaire à été envoyé.

} //Fin du traitement du formulaire de traduction
?>
```

D - Suppression des éléments

Pour supprimer les éléments on a deux fonctions, l'une supprime l'élément `xmldata`, l'autre l'élément `translation` d'un élément `xmldata` dans la langue choisie. On va donc créer un formulaire simple contenant deux listes, l'une avec tous les identifiants des éléments `xmldata`, l'autre les langues utilisées :

Code PHP (removeForm.php)

```
<?php
/*
 * On fait une nouvelle page,
 * On reprend le même début qu'avant :
 */
session_start();
require_once 'include/XMLEngine.php';

//On inclut la classe SQLManager et créer un objet.
require_once 'include/SQLManager.php';
$sql = new SQLManager( 'localhost', 'root', '', 'Test' );

if ( !isset( $_SESSION['Language'] ) )
{
    $_SESSION['Language'] = 'fr'; //Français par défaut.
}

$xml = new XMLEngine( 'xml/website.xml', $_SESSION['Language'] );

/*
 * Inclusion du traitement du formulaire.
 */
require_once 'removeFormTreatment.php';
```

Code PHP (removeForm.php)

```

//Début du formulaire d'administration :
?>
<div style="height:500px;overflow:scroll;" align="center"><?php
/*
 * On prépare une fonction JavaScript pour confirmer la suppression
 * d'un élément, cette fonction renvoi true ou false, permettant d'envoyer
 * ou non le formulaire pour qu'il soit traité.
 */ ?>
<script type="text/javascript">
<!--
//Variable Javascript globale pour savoir quel bouton submit est appuyé.
var button;
function confirmDelete()
{
    if ( button == "xml" )
    {
        return confirm("<?php echo $xml->confirmDelete;?> " + document.removeForm.xmlDataId.value + "
?>");
    }
    else if ( button == "translation" )
    {
        return confirm("<?php echo $xml->confirmDelete;?> 'translation' de langue '"
        + document.removeForm.translationLangId.value + "' de l'element '"
        + document.removeForm.xmlDataId.value + "' ?" );
    }
}
-->
</script>
<?php
/*
 *On ajoute l'attribut 'onsubmit' et 'name' afin d'envoyer le formulaire
 *uniquement après confirmation de la suppression avec la fonction JS.
 */ ?>
<form action="removeForm.php" method="post" name="removeForm" onsubmit="return confirmDelete();">
<table border="1">
<tr>
<td><?php
    echo $xml->element.' <i>xmldata</i>';?>
</td>

<td><?php
    echo $xml->element.' <i>translation</i>';?>
</td>
</tr>
<tr>
<td>
<select name="xmlDataId"><?php
/*
 * On récupère la liste des éléments pour ensuite
 * l'afficher.
 */
$itemList = $xml->getItemNodeList();
foreach ( $itemList as $item )
{?>
    <option value="<?php echo $item;?>"><?php echo $item;?></option><?php
    }?>
</select>
</td>
<td><?php
//On récupère toutes les langues de la base de données SQL
$langs = $sql->selectQuery( '*', 'Langages', 0, '', '', '', '' );

/* Puis on créer la liste */ ?>
<select name="translationLangId" style="width:100%;"><?php
    foreach ( $langs as $language )
    {?>

```

Code PHP (removeForm.php)

```

        <option value="<?php echo $language['Language'];?>"><?php echo $xml->{
    $language['Language'].'Lang' };?></option><?php
        }?>
    </select>
</td>
</tr>
<tr>
<!-- Bouton d envoi pour supprimer un élément xmldata -->
    <td>
        <!-- On utilise l attribut ONCLICK dans lequel on place
        du code JavaScript, on assigne alors à la variable
        bouton (variable JS) le mot XML ou TRANSLATION-->
        <input type="submit" name="submitDeleteXmlDataElement" value="<?php echo
    $xml->deleteElement;?>" onclick="button='xml';" />
    </td>

    <!-- Bouton d envoi pour supprimer un élément translation -->
    <td>
        <!-- Idem pour l attribut ONCLICK que précédement -->
        <input type="submit" name="submitDeleteTranslationElement" value="<?php echo
    $xml->deleteElement;?>" onclick="button='translation';" />
    </td>
</tr>
</table>
</form>
</div>

```

Ensuite le script de traitement du formulaire, inclus juste avant ce dernier pour que la liste soit mise à jour.

Code PHP (removeFormTreatment.php)

```

<?php
/*
 * Fichier uniquement à inclure.
 * Traitement du formulaire de suppression des éléments lorsque celui-ci
 * est envoyé. On doit effectuer deux traitements différents selon le
 * bouton submit utilisé.
 */
if ( isset( $_POST['submitDeleteXmlDataElement'] ) )
{
    /*
     * Pour supprimer un élément xmldata, on utilise la méthode de
     * la classe XMLEngine, qui renvoi true ou false.
     */
    if ( $xml->completelyRemoveXmlElement( $_POST['xmlDataId'] ) )
    {?>
        <script type="text/javascript">
        <!--
            alert("<?php echo $xml->modificationDone;?>");
        -->
        </script><?php
    }
    else
    {?>
        <script type="text/javascript">
        <!--
            alert("<?php echo $xml->modificationError;?>");
        -->
        </script><?php
    }
}
else if ( isset( $_POST['submitDeleteTranslationElement'] ) )
{
    /*
     * Pour supprimer un élément translation, on utilise la méthode de
     * la classe XMLEngine, qui renvoi true ou false.
    */
}
}

```

Code PHP (removeFormTreatment.php)

```

/*
if ( $xml->removeXmlElement( $_POST['xmlDataId'], $_POST['translationLangId'] ) )
{?>
<script type="text/javascript">
<!--
    alert("<?php echo $xml->modificationDone;?>");
-->
</script><?php
}
else
{?>
<script type="text/javascript">
<!--
    alert("<?php echo $xml->modificationError;?>");
-->
</script><?php
}
}??>

```

E - Disponibilité des langues

En dernier on va voir comment vérifier que tous nos fichiers possèdent les éléments nécessaires pour rendre une langue accessible aux utilisateurs. Il nous faut juste un formulaire vérifiant les langues inaccessibles auprès de la base de données, puis ensuite lancer la vérification et modifier la base de données s'il le faut.

Code PHP (languageAvailability.php)

```

<?php
/*
 * On fait une nouvelle page,
 * On reprend le même début qu'avant :
 */
session_start();
require_once 'include/XMLEngine.php';

//On inclut la classe SQLManager et créer un objet.
require_once 'include/SQLManager.php';
$sql = new SQLManager( 'localhost', 'root', '', 'Test' );

if ( !isset( $_SESSION['Language'] ) )
{
    $_SESSION['Language'] = 'fr'; //Français par défaut.
}

$xml = new XMLEngine( 'xml/website.xml', $_SESSION['Language'] );

/*
 * Traitement du formulaire de vérification
 */
if ( isset( $_POST['submitCheckLanguage'] ) )
{
    /*
     * Pour vérifier les langues de tous les fichiers, il
     * faut auparavant mettre les fichiers XML à part dans un
     * répertoire seul. Puis on ouvre ce répertoire :
     */
    $i=0;
    $dirRes = opendir( 'xml' );

    //Ensuite on lit les noms des fichiers un par un
    while ( false !== ( $file = readdir( $dirRes ) ) )
    {
        //Si on a bien un fichier XML
        if ( strpos( $file, '.xml' ) )

```

Code PHP (languageAvailability.php)

```

{
    //On sauvegarde le nom dans un tableau
    $xmlTab[$i++] = $file;
}
}

//Puis si le tableau existe, on va vérifier tous les fichiers
if ( isset( $xmlTab ) )
{
    //On sauvegarde le fichier utilisé acutellement
    $File = $xml->getCurrentFile();

    //Variable de contrôle sur true par défaut
    $fileChecker = true;

    //On parcourt la liste de fichier pour les vérifier
    foreach ( $xmlTab as $file )
    {
        //On change le fichier dans l'objet XMLEngine
        $xml->changeFile( 'xml/'.$file );

        /*
        * Puis on vérifie son intégrité avec la fonction de la classe
        * qui renvoi true ou false.
        */
        if ( !$xml->checkIntegrity( $_POST['languageToCheck'] ) )
        {
            /* En cas d'échec on affiche une alerte pour chaque fichier */?>
            <script type="text/javascript">
            <!--
            alert("<?php echo $xml->fileCheckError.' : '.$file;?>");
            //-->
            </script><?php
            $fileChecker = false;//Variable de contrôle sur false
            }
        }

        /*
        * Puis si la variable de contrôle l'autorise, on met à jour la base de données
        * pour mettre le champ 'IsAvailable' sur 1 ( ou true )
        */
        if ( $fileChecker )
        {
            $fields[0]='IsAvailable';$values[0]=true;
            $sql->updateQuery( 'Languages', $fields, $values, 'Language', '=', $_POST['languageToCheck'] );

            /* Puis on affiche un message */ ?>
            <script type="text/javascript">
            <!--
            alert("<?php echo $xml->checkLanguageDone;?>");
            //-->
            </script><?php
            }
        }
    }
}

/* Début du formulaire d'administration : */ ?>

<div style="height:500px;overflow:scroll;" align="center">
<form action="languageAvailability.php" method="post">
<table border="1">
<tr>
<td><?php
    echo $xml->languageToCheck;?>
</td>
</tr>
<tr>
<td>

```

Code PHP (languageAvailability.php)

```
<td><?php
//On récupère toutes les langues de la base de données SQL
$langs = $sql->selectQuery( '*', 'Languages', 0, '', '', '', '' );

/* Puis on créer la liste */ ?>
<select name="languageToCheck" style="width:100%;"><?php
foreach ( $langs as $language )
{?>
  <option value="<?php echo $language['Language'];?>"><?php echo $xml->{
$language['Language'].'Lang' };?></option><?php
}??>
</select>
</td>
</tr>
<tr>
<td>
  <input type="submit" name="submitCheckLanguage" value="<?php echo $xml->checkIt;?>" />
</td>
</tr>
</table>
</form>
</div>
```

VII - Sources

Sources au format ZIP

Tutoriel XML

Tutoriel sur les fonctions DOM en PHP (utilisées dans la classe)

