Introduction à XML avec PHP

Code: php-xml

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/php-xml/php-xml.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/php-xml.pdf**

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 0.8 (modifié le 6/3/07 par DKS)

Prérequis

Module technique précédent: xml-dom

Module technique précédent: xml-tech

Module technique précédent: php-intro

Modules suivants

Module technique suivant: php-dom

Module technique suivant: visu-gen

Abstract

- XML avec PHP 5 (la plupart du code ne marchera pas avec PHP 4.x !!!)

  Une petite introduction sur comment utiliser PHP pour extraire ("parser") et traiter des données XML.

- "simple xml" avec le module SimpleXML functions
- "stream-parsing" avec le module "XML parser functions" (librairie expat)
- XSLT et XPath avec le module "XSL functions" (librairie libxslt).
- Note: PHP avec "DOM" se trouve dans les transparents php-dom !

Objectifs

- Savoir lire des données XML dans un programme PHP
- Extraire et manipuler des données xml (visualisations, etc.)

A faire:

- revoir le tout (il s'agit ici d'une première version pour Php 5)
- comment éditer (transformer) des données xml
- un exemple lire du XML avec simple XML pour générer des questionnaires
- XML Reader functions (PHP 5.1)

1. Table des matières

2. Introduction

Objectifs du chapitre:

1. Définitions
2. Références

2.1 Que peut-on faire avec PHP ?

Générer des contenus en XML

- Générer des contenus en XML est presque aussi facile que générer du HTML
- Soit vous utilisez des fonctions comme echo, print, sprint, etc. pour directement créer les balises, soit vous utilisez une librairie qui vous facilite cette tâche.

Analyse (parsing)

- "Parsing" c'est la procedure avec laquelle on extrait des données et on les découpe dans des morceaux d'information pour les traiter dans la suite. Le programme qui fait le "parsing" s'appelle un "parser".

  url: **http://www.parsifalsoft.com/gloss.html (Glossaire des termes de parsing)**

2.2 Références et exemples

Manuels en ligne

- DOM reference

  url: **http://www.w3.org/TR/**
- Fonctions SimpleXML (dans le site PHP):

  url: **http://ch.php.net/manual/en/ref.simplexml.php**
- Fonctions XML "stream-parsing" (sur le site PHP):

  url: **http://www.php.net/manual/en/ref.xml.php**

Répértoires avec des exemples:

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/**

2.3 Les modèles d'analyse

Il existe deux modèles de "parsing" dans la plupart des langages:

1. "***Stream-parsing***" (connu sous le nom de SAX - Simple API for XML - dans le monde "Java":
   - se base sur les événements, ça veut dire que le programmeur doit définir 3 fonctions qui notifient l'application lorsque le parseur trouve le début, le contenu et la fin d'un élément XML
2. ***DOM*** parsing: fait référence au "Document Object Model" du W3C
   - le parseur traverse un fichier xml et il le ré-construit sous forme d'arbre
   - Le DOM est abordé dans une autre série de transparents: php-dom.

   Le "stream-parsing" est plus rapide et surtout moins gourmand en mémoire, mais plus difficile à utiliser dans la plupart des cas.
3. Dans PHP 5 on a "Simple XML"
   - méthode la plus simple à utiliser pour "lire" un fichier XML dans un "array"
   - suffisante pour la plupart des besoins simples
   - Mais attention: ne peut pas toujours remplacer DOM parsing !
4. Dans PhP 5.1 on peut avoir "Reader functions"
   - un "pull" parser qui lit noeud par noeud d'un document et qui permet d'extraire des informations.

   Note: Simple XML et DOM acceptent aussi des requêtes XPath.

3. Générer du XML

3.1 Le Mimetype et les entêtes dans les fichiers

Lorsque vous produisez d'autres contenus que HTML avec PHP, il faut veiller à deux choses:

1. Votre serveur doit indiquer à votre client de quel type de document il s'agit (indiquer le "Mime Type"
2. Le document envoyé par votre programme doit contenir les déclarations nécessaires

A. Définition du mime type

- Cette instruction qui modifie le message HTTP du serveur doit intervenir ***tout au début du fichier !*** (donc éviter toute instruction de type echo, print, etc. avant)
- Attention, certains formats XML ont leur propre mime-type !

Exemple XML

```
Header("Content-type: text/xml);
```

Exemple SVG

```
Header("Content-type: image/svg+xml");
```

Exemple RDF

```
Header("Content-type: application/rdf+xml");
```

B. Les entêtes de vos fichiers

- Il s'agit des lignes 1 (et plus) dans le document transmis.

XML (simple)

```
print('<?xml version="1.0" encoding="iso-8859-1"?>');
```

Avec SVG:

```
print('<?xml version="1.0" encoding="iso-8859-1"?>' . "\n");
print('<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/
2001/PR-SVG-20010719/DTD/svg10.dtd">' . "\n");
```

Exemple 3-1: Générer et afficher un simple contenu avec XML

url: **http://tecfa.unige.ch/guides/php/examples/simple/simple-calcul-xml.php**

url: **http://tecfa.unige.ch/guides/php/examples/simple/simple-calcul-xml.phps**

url: **http://tecfa.unige.ch/guides/php/examples/simple/simple-calcul-xml.css**

```
<?php
header("Content-type: text/xml");
print('<?xml version="1.0" encoding="iso-8859-1"?>' . "\n");
print('<?xml-stylesheet href="simple-calcul-xml.css" type="text/css" ?>');

$leisure_satisfaction = 5; $work_satisfaction = 7;  $family_satisfaction = 8;
$index = ($leisure_satisfaction + $work_satisfaction + $family_satisfaction) / 3 ;
echo "<resultat> Satisfaction Index = $index </resultat>";
?>
```

Note: pour aller plus loin, voir:

url: **http://tecfa.unige.ch/guides/tie/html/visu-gen/visu-gen.html (Visualisation)**

4. Parsing avec "Simple XML"

Résumé des fonctionalités

- La fonction "simplexml_load_file" permet de parser un document XML dans une structure PHP qui ressemble à des indexed arrays dans des indexed arrays. On peut donc facilement accéder à des éléments (avec des sélecteurs de type "array").
- On peut aussi effectuer des recherches avec une expression XPath.
- Du manuel: The SimpleXML extension provides a very simple and easily usable toolset to convert XML to an object that can be processed with normal property selectors and array iterators
- ***Avantage:*** On peut très facilement intégrer une structure XML complète dans un programme PHP et ensuite la manipuler.
- ***Désavantage:***
  - Solution particulière à PHP, autrement dit ce type d'approche ne se retrouve pas dans d'autres langages de programmation (contrairement à DOM et SAX).
  - Le nom de l'élément racine disparaît dans la nature (!)
- Note: En PHP 4.x une fonctionalité similaire était disponible sous le nom "xmltree")

Principe illustré avec un exemple:

```
SimpleXMLElement Object
(
 [Titre] => Le garçon webmestre
 [Contexte] => Il était une fois un garçon ni joli ni moche,
ni bête ni intelligent, ni drôle ni ennuyeux.

 [Probleme] => Il était assez heureux dans sa vie de webmaster
pour l'office de la promotion des technologies anciennes.
Toutefois, il lui manquait une amie.

 [But] => Il fallait que cela change !

 [FIL] => SimpleXMLElement Object (
     [EPISODE] => Array (
         [0] => SimpleXMLElement Object (
             [SousBut] => Un jour il s'est dit qu'il doit agir
coûte que coûte.
             [TENTATIVE] => SimpleXMLElement Object (
                 [Action] => Il s'est rendu au pub du coin.
Arrivé au bar il voit un ancien camarade de classe accompagné
de deux filles. Ils commencent à discuter et quand il raconta
qu'il était WebMaster, une des filles lui demande ce qu'il
faisait. Alors il expliqua fièrement qu'il faisait des pages
HTML avec Frontier. Et avec un élan de courage il demanda à
la fille s'il pouvait lui offrir un verre.
                                 )
             [Resultat] => La fille lui répondit: "Non merci".
Un peu désespéré le garçon rentra chez lui.
             )
         [1] => SimpleXMLElement Object(
             [SousBut] => Il sentit qu'il lui tout seul ne pouvait
pas résoudre le problème.
             [TENTATIVE] => SimpleXMLElement Object (
                 [EPISODE] => SimpleXMLElement Object (
                     [SousBut] => Se souvenant qu'il avait une
marraine qui occupait un poste important dans les ressources
humaines, il décida de lui demander conseil.
```

```
)
<RECIT xmlns:xlink="http://www.w3.org/1999/xlink">
 <Titre>Le garçon webmestre</Titre>
 <Contexte>Il était une fois un garçon ni joli ni moche, ni
bête ni intelligent, ni drôle ni ennuyeux.
 </Contexte>
 <Probleme>Il était assez heureux dans sa vie de webmaster
pour l'office de la promotion des technologies anciennes.
Toutefois, il lui manquait une amie.
 </Probleme>
 <But>Il fallait que cela change !
 </But>
 <FIL>
  <EPISODE id="ep1">
   <SousBut>Un jour il s'est dit qu'il doit agir coûte que
coûte.
   </SousBut>
   <TENTATIVE>
    <Action>Il s'est rendu au pub du coin. Arrivé au bar il
voit un ancien camarade de classe accompagné de deux filles.
Ils commencent à discuter et quand il raconta qu'il était
WebMaster, une des filles lui demande ce qu'il faisait. Alors
il expliqua fièrement qu'il faisait des pages HTML avec
Frontier. Et avec un élan de courage il demanda à la fille
s'il pouvait lui offrir un verre.</Action>
   </TENTATIVE>
   <Resultat>La fille lui répondit: "Non merci". Un peu
désespéré le garçon rentra chez lui.
   </Resultat>
  </EPISODE>
  <EPISODE id="ep2">
   <SousBut>Il sentit qu'il lui tout seul ne pouvait pas
résoudre le problème.
   </SousBut>
```

Un élément avec enfants devient un simple XML Object
Ses enfants sont accessibles par une clef (nom de la balise)
La clef retourne soit un contenu, un XML Object, une liste de XML Objects

Exemple 4-1: Lire un fichier XML avec SimpleXML

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/simplexml0.php**

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/ (répertoire)**

```
<?php
  if (file_exists('story.xml')) {
  $xml = simplexml_load_file('story.xml');

  echo "<hr>Here is a dump of the data structure:";

  echo "<pre>";
  print_r($xml);
  echo "</pre>";

} else {
  exit('Failed to open story.xml.');
}
?>
```

- `simplexml_load_file()` charge un fichier XML
- `print_r()` imprime une structure de données de façon "human-readable"
- `var_dump()` imprime plus de détails "techniques".
- Donc faites un "print_r" avant de se lancer dans programmation de l'extration des données qui vous intéressent !!

Exemple 4-2: Extraire des données avec SimpleXML

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/simplexml1.php**

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/ (répertoire)**

```
$xml = simplexml_load_file('story.xml');

echo "<hr>Here we just display some elements (i.e. &lt;SousBut&gt; and &lt;Resultat&gt;
elements found in RECIT->FIL->EPISODE). &lt;FIL&gt; can contain several
&lt;EPISODE&gt;";

  $episodes = ($xml->FIL->EPISODE);
  foreach ($episodes as $episode) {
    echo "<p>Episode:</p> ";
    echo"<pre>";
    printf("Sousbut: %s\n", $episode->SousBut);
    printf("Resultat: %s\n", $episode->Resultat);
    print "----\n";
    echo"</pre>";
```

- `$xml->FIL->EPISODE` collectionne tous les eléments "EPISODE" filles de "FIL".
- foreach (`$episodes as $episode`) ... est une technique standarde pour boucler sur tous les éléments d'un array. `$episode` sera lié à chaque item trouvé lors d'un passage.
- `$episode->SousBut` extrait l'élément SousBut

Exemple 4-3: Extraire des données avec SimpleXML et Xpath

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/simplexml2.php**

url: **http://tecfa.unige.ch/guides/php/examples/simplexml-functions/ (répertoire)**

- Meme exemple que le précédant sauf qu'on utilise XPath pour extraire la liste des épisodes.

```
$xml = simplexml_load_file('story.xml');

$episodes = $xml->xpath('//EPISODE');
```

- `$xml->xpath('//EPISODE')` collectionne tous les eléments "EPISODE" filles de "FIL".
- foreach (`$episodes as $episode`) ... est une technique standarde pour boucler sur tous les éléments d'un array. `$episode` est un item.

5. "Stream-parsing"

Principe du stream parsing: le parseur lit le fichier XML et "crache" des informations:

- Il faut définit des "handlers" qui qui sont appelés par le parser dès qu'il tombe sur un début ou une fin d'élément.
- Le parseur fait appel à ces handlers pour chaque début ou fin de balise: "tiens voilà le début d'un élément "p" et "voici la list des attributs".
- Lorsqu'il tombe sur des caracter data (du contenu), il fait également appel à un handler.

Identification des "Event Handlers" pour les éléments

Définition:

```
Syntaxe: xml_set_element_handler ($parseur, "start_event_handler",
   "end_event_handler")
```

Définition formelle du manuel:

```
Syntaxe: bool xml_set_element_handler ( resource parser, callback
   start_element_handler, callback end_element_handler )
```

nom_handler_début" et "nom_handler_fin" sont fonctions que vous définissez.

Définition des "Event Handlers

Les fonctions start-event_handler et end_event_handler (vous pouvez choisir d'autres noms) doivent obéir aux définitions suivantes:

```
Syntaxe: start_element_handler ($parser, $nom_element, $attributs)
Syntaxe: end_element_handler ($parser, $nom_element)
```

Exemple:

```
function startElement($parser, $name, $attrs)
{
    print "j'ai trouvé l'éléement: $name <br>";
   }
```

Identification du character data handler

Syntaxe: `xml_set_character_data_handler($parser, "characterData");`

Définition du character data handler

Syntaxe: `characterData($parser, $data)`

Exemple:

```
function characterData($xml_parser, $data)
{
    print $data;
}
```

Exemple 5-1: Structure d'un fichier xml en forme d'une liste des éléments

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/list-xml-elements.php**

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/list-xml-elements.phps**

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/story.xml**

```
<?
$file = "story.xml";
$depth = 0;

# appelé au début d'une balise,l'imprime avec une jolie indentation
function startElement($parser, $name, $attrs)
{
    global $depth;
    for ($i = 0; $i < $depth; $i++) {
        print "   ";
    }
    print "$name\n<br>";
    $depth[$parser]++;
}
# appelé à la fin d'une balise et imprime
function endElement($parser, $name)
{
    global $depth;
    $depth[$parser]--;
    for ($i = 0; $i < $depth; $i++) {
        print "   ";
    }
    print "/$name\n<br>";
}
```

```
# créer un "parser"
$xml_parser = xml_parser_create();

# commence à "parser"
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}

# libérer le "parser"
xml_parser_free($xml_parser);

?>
```

Exemple 5-2: Visualisation des données xml en html

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/map-x-html2.php**

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/map-x-html2.phps**

url: **http://tecfa.unige.ch/guides/php/examples/simple-xml/choco-chip.xml**

```
<?
$file = "choco-chip.xml";

// Put the key in upper case !
$begin_array = array(
            "LIST"        => "<H1>RECIPE LIST</H1>",
            "RECIPE"      => "<HR>",
            "RECIPE_NAME" => "<H2>",
            "AUTHOR"      => "Author: <STRONG>",
            "MEAL"        => "<H3>",
            "COURSE"      => "<I>",
            "INGREDIENTS" => "<H3>Ingrediants</H3><OL>",
            "ITEM"        => "<LI>",
            "DIRECTIONS"  => "<H3>Directions</H3><BLOCKQUOTE>"
            );

$end_array = array(
            "LIST"        => "<BR>",
            "RECIPE"      => "<BR>",
            "RECIPE_NAME" => "</H3>",
            "AUTHOR"      => "</STRONG>",
            "MEAL"        => "</H2>",
            "COURSE"      => "</I>",
            "INGREDIENTS" => "</OL>",
```

```
            "ITEM"        => "</LI>",
            "DIRECTIONS"  => "</BLOCKQUOTE>"
            );

function startElement($parser, $name, $attrs)
{
    global $begin_array;
    // print "DEBUG: $name <br>";
    if ($htmlexpr = $begin_array[$name]) {
        print "$htmlexpr";
    }
}

function endElement($parser, $name)
{
    global $end_array;
    if ($htmlexpr = $end_array[$name]) {
        print "$htmlexpr";
    }
}

function characterData($parser, $data)
{
    print $data;
}

$xml_parser = xml_parser_create();

// use case-folding so we are sure to find the tag in $begin_array
// does this REALLY work ??? / DKS
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
```

```
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");

if (!($fp = fopen($file, "r"))) {
  die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
  if (!xml_parse($xml_parser, $data, feof($fp))) {
    die(sprintf("XML error: %s at line %d",
        xml_error_string(xml_get_error_code($xml_parser)),
        xml_get_current_line_number($xml_parser)));
  }
}

xml_parser_free($xml_parser);

?>
```

6. XSLT avec DOM

- Attention: Dans le manuel PHP 5 il faut consulter la section XSL (pas XSLT !)
- Ici on expliquera pas PhP-DOM, copiez simplement de l'exemple ci-dessous

6.1 Usage simple

Exemple 6-1: Lire un xml et un XSLT et renvoyer un document

url: **http://tecfa.unige.ch/guides/php/examples/xslt/php-xslt-1.php**

url: **http://tecfa.unige.ch/guides/php/examples/xslt/**

```
$xml_file = 'programme.xml';
$xsl_file = 'programme.xsl';

// load the xml file (and test first if it exists)
$dom_object = new DomDocument();
if (!file_exists($xml_file)) exit('Failed to open $xml_file');
$dom_object->load($xml_file);

// create dom object for the XSL stylesheet and configure the transformer
$xsl_obj = new DomDocument();
if (!file_exists($xsl_file)) exit('Failed to open $xsl_file');
$xsl_obj->load($xsl_file);
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl_obj); // attach the xsl rules

$html_fragment = $proc->transformToXML($dom_object);
print ($html_fragment);
```

6.2 Inclure des fonctions PHP dans XSLT

- Attention, les puristes vont vous détester pour cela car le XSLT ne marchera qu'avec PHP
- Donc utiliser avec modération, c.a.d. seulement quand il manque une fonction importante dans XSLT (comme par exemple les fonctions trigonométriques)

Pour que cela marche il faut enregister les fonctions php dans xslt:

```
$proc = new XSLTProcessor;
$proc->registerPHPFunctions();
```

Fonctions PHP dans XSLT:

- Il faut déclarer un namespace php

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

- Syntaxe d'un appel à une fonction PhP:

```
Syntaxe: php:function('nom_de_la fonction', arg, arg, ...)
```

Extrait d'un fichier XSL

```
<xsl:variable name="pos_y" select="$ori_y + round(php:function('sin',
$item_angle) * $radius)"/>T
```

url: **http://tecfa.unige.ch/guides/svg/ex/objects-in-circles/**

url: **http://tecfa.unige.ch/guides/svg/ex/objects-in-circles/elements-on-circle-with-xslt.phps**

url: **http://tecfa.unige.ch/guides/svg/ex/objects-in-circles/elements-on-circle-with-xslt.php**

url: **http://tecfa.unige.ch/guides/svg/ex/objects-in-circles/elements-on-circle-with-xslt.xsl**