christophe.ollivier@virtuoworks.com
VirtuoWorks

XML

Table des matières

1 XML

1.1 Introduction

XML (**Extensible Markup Language**) est un langage de traitement de documents proposé par le World Wide Web Consortium (W3C).

Ce langage est dit extensible car, contrairement à HTML, il ne s'agit pas d'un format fixe. Son but est de permettre d'utiliser SGML sur le World Wide Web. C'est un méta-langage, un langage qui permet de définir d'autres langages -qui vous permet de concevoir votre propre balisage.

Le langage XML est un vecteur privilégié d'échange de données entre applications différentes.

SGML signifie Standard Generalized Markup Language (ISO 8879), langage normalisé de balisage généralisé ; c'est la norme internationale pour décrire la structure et le contenu de différents types de documents électroniques.

XML est une version abrégée de SGML, qui permet de définir ses propres types de documents plus facilement, et qui permet aux programmeurs de développer plus facilement des programmes permettant de traiter ces documents.

Bien que nombre de spécifications associées à XML soient encore en développement, nombreux sont ceux qui espèrent que XML et ses technologies parentes remplacent HTML comme langage de marquage par excellence pour les contenus web générés dynamiquement, y compris les pages web non statiques.

XML est un **méta-langage** permettant de créer et de formater les balises de vos propres documents. Avec HTML, le marquage existant est figé : *<head>* et *<body>*, par exemple, sont des marqueurs étroitement liés à HTML, qui ne peuvent être modifiés ou étendus.

XML, à l'inverse, permet de créer ses propres marqueurs et de les configurer individuellement en fonction de ses besoins : par exemple, *<titreA>*, *<ascenseur>*, *<auteurLivre>*.

Ainsi, il est important de réaliser qu'il n'y a pas de marqueurs "corrects" pour un document XML, excepté ceux que vous avez définis vous-même.

Voici les principaux atouts de XML :

- **Lisibilité** : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML
- **Autodescriptif et extensible**
- **Structure arborescente** : permettant de modéliser la majorité des problèmes informatiques
- **Universalité et portabilité** : les différents jeux de caractères sont pris en compte
- **Déployable** : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP
- **Intégrabilité** : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML)
- **Extensibilité** : un document XML doit pouvoir être utilisable dans tous les domaines d'applications

Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

L'intérêt de disposer d'un format commun d'échange d'information dépend du contexte professionnel dans lequel les utilisateurs interviennent. C'est pourquoi, de nombreux formats de données **issus de XML** apparaissent (il en existe plus d'une centaine) :

OFX : Open Financial eXchange pour les échanges d'informations dans le monde financier

MathML : Mathematical Markup Language permet de représenter des formules mathématique

CML : Chemical Markup Language permet de décrire des composés chimiques

SMIL : Synchronized Multimedia Integration Language permet de créer des présentations multimédia en synchronisant diverses sources : audio, vidéo, texte,...

SVG : Scalable Vector Graphics

1.2 TERMINOLOGIE

Avant de nous engager plus en avant, nous devons standardiser la terminologie. Un document XML contient un ou plusieurs éléments. Un élément est marqué de la manière suivante:

```
<livre>
Ceci est le contenu de l'élément livre
</livre>
```

Cet élément comprend deux marqueurs, un marqueur d'ouverture qui place le nom de l'élément entre un signe plus petit que (<) et un signe plus grand que (>).

Chaque fichier XML doit avoir un **élément racine** englobant l'ensemble des éléments sous-jacents.

Les éléments peuvent avoir des attributs comme dans l'expression suivante:

```
<prix monnaie="euro" > 27,52 </prix>
```

Ici, l'**attribut** est spécifié à l'intérieur du marqueur d'ouverture et est appelé "monnaie". Il reçoit une valeur "euro", exprimé entre guillemets. Les attributs sont souvent utilisés pour affiner ou modifier le comportement d'un élément.

En dehors des éléments standard, XML supporte également des éléments vides. Un élément vide ne comporte pas de texte entre les marqueurs d'ouverture et de fermeture. En conséquence, les deux marqueurs peuvent être fusionnés (optionnel), un slash apparaissant alors devant le marqueur de fermeture. Par exemple, les éléments suivant sont identiques :

```
<photo src="/img/yeah.gif"> </photo>
<photo src="/img/yeah.gif" />
```

Les éléments vides sont souvent utilisés pour ajouter un contenu non littéral à un document, ou pour fournir une information supplémentaire à l'application qui analyse le code XML.

Il est à noter que si le slash de fermeture peut être ignoré dans les éléments HTML ouverts (à un seul marqueur), il est obligatoire pour les éléments XML vides.

1.3 TROIS ERREURS À ÉVITER

Les valeurs d'attributs doivent être placées entre guillemets

On ne peut pas spécifier en XML une valeur d'attribut telle que :

```
<photo src=/img/kebab.jpg> <!-- incorrect ! -->
<photo src="/img/kebab.jpg"> <!-- CORRECT ! -->
```

Or c'est une erreur sur laquelle les navigateurs HTML ferment généralement les yeux. Une valeur d'attribut doit toujours être placée entre guillemets, simples ou doubles.

Un élément non vide doit avoir un marqueur d'ouverture et de fermeture

Chaque élément matérialisé par un marqueur d'ouverture doit posséder le marqueur de fermeture qui lui correspond. Si ce n'est pas le cas, et s'il ne s'agit pas d'un élément vide, l'analyseur XML génère une erreur.

Vous ne pouvez pas écrire ce qui suit :

```
<paragraphe>
      Voici un paragraphe
</paragraphe>
Voici un autre paragraphe
```

Les marqueurs doivent être emboîtés correctement

Il est illégal d'écrire ce qui suit :

```
<italic><bold>ceci est incorrect </italic></bold>
```

Le marqueur de fermeture de l'élément « **bold** » devrait être placé avant le marqueur de fermeture de l'élément « **italic** » pour préserver l'emboîtement correct des éléments.

1.4 EXEMPLE DE DOCUMENT XML

```
<!-- Prologue -->
<?xml version="1.0" encoding="ISO-8859-15"?>
<!-- Élément racine -->
<biblio>
  <!-- Premier enfant -->
  <livre>
    <!-- Élément enfant titre -->
    <titre>Les Misérables</titre>
    <auteur>Victor Hugo</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
  <livre>
```

```
    <titre>L'Assomoir</titre>
    <auteur>Émile Zola</auteur>
  </livre>
  <livre lang="en">
    <titre>David Copperfield</titre>
    <auteur>Charles Dickens</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
</biblio>
```

Déclaration XML

Cette déclaration fait partie des "instructions de traitement". Exemple de déclaration XML :

```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>
```

On distingue trois informations fournies dans cette déclaration :

- **version** : version du XML utilisée dans le document, 1.0 en ce qui nous concerne (la dernière version du langage, 1.1, date de février 2004 mais ne change rien quant à ses bases)
- **encoding** : le jeu de codage de caractères utilisé. Le jeu de caractère standard pour la France est le ISO-8859-1. il a tendance à être remplacé par l'ISO-8859-15 en attendant la généralisation de l'Unicode. Par **défaut**, l'attribut encoding a la valeur **UTF-8**. Cela permet à l'ordinateur de "savoir" quel caractère il doit afficher en réponse aux combinaisons de 1 et de 0 que contient le fichier sur le disque dur. Voir le chapitre suivant.
- **standalone** : dépendance du document par rapport à une déclaration de type de document (DTD : définitions de Type de données). XML a été conçu pour être utilisé avec ou sans DTD (SGML utilise les DTD). Opérer sans DTD permet d'inventer un balisage sans avoir à le définir de façon formelle, avec le risque de perdre le contrôle automatique de la structuration de documents additionnels de même type. Standalone= ‘yes‘ pour une utilisation sans DTD.

Par exemple, l’élément <html> ne peut avoir comme éléments fils que les éléments <head> et <body> et comme attributs lang et dir.

Cette déclaration est facultative, mais il est préférable de l'utiliser, auquel cas les attributs version, encoding et standalone doivent être placés dans cet ordre. Si elle est utilisée, elle doit être placée en toute première ligne du document XML.

La déclaration simplifiée est la suivante :

```
<?xml version="1.0" ?>
```

Instructions de traitement

Une instruction de traitement est une instruction interprétée par l'application servant à traiter le document XML. Elle ne fait pas totalement partie du document. Les instructions de traitement qui servent le plus souvent sont la déclaration XML ainsi que la déclaration de feuille de style. Exemple d'instruction de traitement :

```
<?xml-stylesheet type="text/xsl" href="biblio.xsl"?>
```

Dans cet exemple, l'application est xml-stylesheet, le processeur de feuille de style du XML. Deux feuilles de style différentes peuvent être utilisées, les XSL (propres au XML) ainsi que les CSS (feuilles de style apparues avec le HTML). L'attribut type indique de quel type de fichier il s'agit (text/css pour les feuilles de style CSS, par exemple) et l'attribut href indique l'URL du fichier. Cette instruction de traitement est notamment utilisée par les navigateurs Internet pour la mise en forme du document. Nous verrons dans un chapitre suivant la mise en forme d'un fichier XML via une feuille de style au format CSS.

Les commentaires

En XML, les commentaires se déclarent de la même façon qu'en HTML. Ils **commencent donc par <!-- et se terminent par -->**. Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise.

Exemples de commentaires valides :

```
<!-- ceci est correct -->
<elt> <!-- ceci est correct aussi -->
Un peu de texte </elt>
```

1.5 CHARACTER-ENCODING

A l'instar d'une base de données relationnelle comme MySql, votre fichier XML doit contenir des informations sur la manière dont sont codées les informations. C'est ce que l'on nomme l'encodage de caractère. Le character-encoding en V.O.

Remontons à la base

Comme toute donnée, une chaîne de caractères est "dans l'ordinateur" une suite d'octets de différentes valeurs. Dans son expression la plus simple, chaque valeur d'octet correspond à une lettre (ex: un octet de valeur 65 correspond à un "A"). Comme un octet peut prendre une valeur comprise entre 0 et 255, on a suffisamment de valeurs pour décrire l'alphabet (majuscules + minuscules + chiffres + ponctuation).

La correspondance (ou table ou **charset**) "standard" entre octet et caractère correspond à l'encodage ASCII, qui suffit à couvrir l'alphabet anglo-saxon avec des valeurs comprises entre 0 et 127.

En France, on a aussi besoin des caractères accentués (éèê etc...); il existe donc un encodage normé avec le même principe 1 octet = 1 caractère qui décrit aussi les caractères français. C'est le codage ISO-8859-1 ou Latin-1, qui étend la table ASCII. Il existe aussi ANSI et **ISO-8859-15** qui étendent ISO-8859-1 de quelques caractères comme €.

D'autres langues ne vont pas avoir besoin des caractères accentués et vont utiliser les mêmes valeurs d'octet pour décrire d'autres caractères spécifiques (ex. ISO 8859-2 pour l'Europe centrale, ISO 8859-9 pour le turc, etc...). Si on utilise la table turque pour lire un texte français Latin-1, les accents vont être remplacés par des caractères turcs. Les problèmes sont encore pires avec les tables asiatiques...

Unicode encodé en UTF-8

On a donc créé une table "**universelle**" où tous les caractères existants, ou ayant existé, ont une valeur correspondante. C'est **Unicode**. Mais dans ce cas, 1 octet ne suffit plus à décrire l'éventail des valeurs : on doit alors utiliser plusieurs octets consécutifs pour décrire cette valeur. C'est ce à quoi sert l'**encodage UTF-8**.

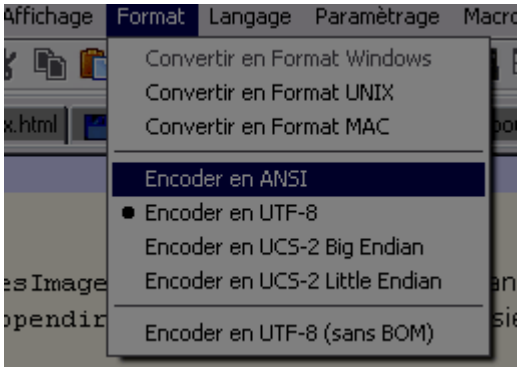
Plutôt que de bêtement utiliser un nombre fixe d'octets pour chaque caractère (ex 4 octets par caractère, ce qui produirait des fichiers 4x plus lourds), en encodage UTF-8, un caractère sera codé sur 1 à 4 octets suivant sa valeur dans la table Unicode. Ainsi un "A" reste la valeur 65 codé sur 1 octet, un "é" sera codé sur 2 octets... C'est pour cela que lorsque vous regardez un "é" encodé en UTF-8 sur 2 octets avec un afficheur qui prend ça pour du Latin-1, vous verrez "Ã©"

Au niveau d'un éditeur de texte

Il faut utiliser un éditeur qui connaît UTF-8. L'éditeur doit être capable de lire un texte UTF-8 et de le comprendre comme tel (c'est à dire afficher des "é" et non des "Ã©"), et il doit être capable d'enregistrer le texte en UTF-8. Heureusement c'est le cas de tous les bons éditeurs. Même Notepad connait UTF-8. Le choix de l'encodage se fait généralement dans les Préférences ou au moment de l'enregistrement du document.

Attention donc à ne pas confondre l'entête que vous rédigez en XML, et le format dans lequel le fichier lui-même (monflux.xml par exemple) est enregistré.

L'écran ci-dessous vous montre comment modifier l'encodage d'un fichier xml dans notepad++.



1.6 SECTION CDATA

Les s**ections CDATA** sont utilisées pour habiller des blocs de texte contenant des caractères qui seraient autrement reconnus comme balisage.

Les crochets servant généralement aux balises, « < » et « > », seront donc reconnus comme élément de balise (contenu entre deux balises)

Un section CDATA commence par **<![CDATA[** et se conclut par **]]>**

```
<![CDATA[  Une balise commence par un < et se termine par un >.  ]]>
```

Sans cette section, les deux crochets seraient pris pour un début de balisage et généreraient donc une erreur.

2 METTRE EN FORME DU XML

XML est un format de description des données et non de leur représentation, comme c'est le cas avec HTML. La mise en page des données est assurée par un langage de mise en page tiers. A l'heure actuelle, il existe trois solutions pour mettre en forme un document XML :

CSS (Cascading StyleSheet), la solution la plus utilisée actuellement, étant donné qu'il s'agit d'un standard qui a déjà fait ses preuves avec HTML

XSL (eXtensible StyleSheet Language), un langage de feuilles de style extensible développé spécialement pour XML

XSLT (eXtensible StyleSheet Language Transformation). Il s'agit d'une recommandation W3C du 16 novembre 1999, permettant de transformer un document XML en document HTML accompagné de feuilles de style

2.1 FEUILLES DE STYLE CSS

Les **feuilles de style en cascade** ou CSS permettent aux auteurs XML de présenter un document de manière attrayante et d'indiquer au navigateur (ou à tout autre agent utilisateur) les propriétés de style à appliquer aux composants du document XML.

Puisque le XML vous offre la possibilité de créer vos propres balises, il devient aisé d'y appliquer une feuille de style.

En effet, de la même manière que vous pouvez redéfinir tous les éléments <p> de votre document HTML, vous allez pouvoir redéfinir tous vos éléments <auteur>,<livre>,<voyage>,<fleurs> etc.

XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="exemple.css"  type="text/css" ?>
<texte>
      Pendant le semestre, nous nous intéresserons à trois auteurs :
      <auteur> Steinbeck </auteur>
      <auteur> Salinger </auteur>
      <auteur> Roth </auteur>
      Nous les présenterons tour à tour.
</texte>
```

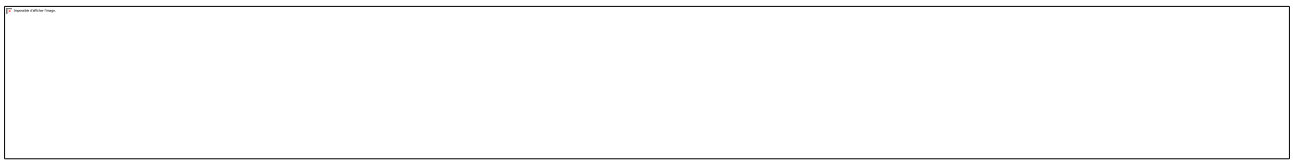
CSS :

```
texte{display:block}
auteur{display:inline; font-weight:bold}
```

La feuille de style est liée avec l'instruction de traitement suivante en début du fichier xml:

```
<?xml-stylesheet href="exemple.css"  type="text/css" ?>
```

Voici le résultat dans Firefox 30 :



2.2 FEUILLES DE STYLE XSL (XSLT)

XSL signifie **eXtensive Stylesheet Langage**, ou langage extensible de feuille de style. XSLT signifie eXtensible Stylesheet Langage Transformation.

Comme son nom l'indique, le XSL ne sert pas uniquement de langage de feuille de style. Il est aussi un très puissant **manipulateur d'éléments**. Il permet de transformer un document XML source en un autre, permettant ainsi, à l'extrême, d'en bouleverser la structure.

Le fonctionnement du XSL est fondé sur les **manipulations de modèles (templates)**. Les éléments du document XML d'origine sont remplacés (ou légèrement modifiés) par ces modèles. Un modèle contient ainsi le texte (éléments, attributs, texte...) de remplacement d'un élément donné.

Structure d'un document XSL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <head>
      <title>Exemple de sortie HTML</title>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    </head>
    <body>
      <h1>Bouteille de marque <xsl:value-of select="bouteille/marque" /></h1>
      <h2>Composition:</h2>
      <p><xsl:value-of select="bouteille/composition" /></p>
      <h2>Lieu d'origine:</h2>
      <p>Ville de <b><xsl:value-of select="bouteille/source/ville" /></b>, dans
le département <b><xsl:value-of select="bouteille/source/departement" /></b></p>
      <h2>Autres informations</h2>
      <ul>
        <li>Contenance: <xsl:value-of select="bouteille/contenance" /></li>
```

```
        <li>pH: <xsl:value-of select="bouteille/ph" /></li>
      </ul>
    </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

L'élément **<xsl:stylesheet>** est l'élément racine du document XSL. C'est lui qui contient tous les modèles, y compris celui qui est associé à la racine du document XML, modèle que l'on note **<xsl:template match="/">**.

L'attribut match="/" indique que ce modèle s'applique à la racine du document XML.

Il est ensuite possible de créer des templates spécifiques pour les balises enfants. Dans notre cas on pourrait créer un template :

```
<xsl:template match="bouteille">
        < !-- Instructions -->
</xsl:template>
```

Pour lier un fichier xsl à un fichier xml, on procède de la même façon qu'avec un fichier css:

```
<?xml-stylesheet href="bouteilles.xsl"  type="text/xsl" ?>
```

3 XML et PHP5

Le but est de montrer que PHP permet d'intégrer du contenu dans un fichier XML, et d'afficher le résultat dans une page avec la mise en forme souhaitée.

L'application principale du XML est le partage d'informations entre applications hétérogènes.

Ainsi, avec l'aide de PHP, nous devons être capables de :

- Créer un fichier XML à partir des données contenus dans une base de données.
- Alimenter une base de données à partir des informations contenues dans un fichier XML

3.1 FONCTION PHP simplexml_load_file()

Cette fonction transfère le contenu du fichier XML dans un objet de type simplexml_element contenant l'arborescence du fichier. Les propriétés de cet objet prennent pour noms ceux de chacun des éléments XML et pour valeurs les contenus des éléments du fichier.

Utilisation : nous avons par exemple un fichier livres.xml :

```
<?php

$ml = simplexml_load_file('livres.xml');

<!-- Instructions →

?>
```

La variable $xml est alors un objet.

Exemple de syntaxe pour afficher le contenu d'un élément nommé <livre> :

```
echo $xml->livre ;
```

Pour afficher le contenu de sous-élément (ie un sous-objet) d'un élément nommé <livre>, nous devons utiliser une syntaxe proche de celle des tableaux (syntaxe utilisée pour les objets) :

```
echo $xml->livre[0]->titre ;
echo $xml->livre[1]->auteur ;
```

Pour afficher le contenu d'un attribut appartenant à un élément nommé <livre>, nous utilisons la syntaxe suivante :

```
echo $xml->livre['nom'] ;
echo $xml->livre->auteur['age'] ;
```

3.2 APPLICATION 1 : Afficher des photos à partir d'un fichier XML

Création d'un fichier XML nommé photos.xml :

```
<?xml version="1.0" ?>
<photos>
	<auteur id='15' name='Jerem' >
		<photo num='1522' cat='china' uri='blablu.jpg' />
		<photo num='1545' cat='china' uri='ventrilok.jpg' />
		<photo num='1111' cat='china' uri='yeah.jpg' />
	</auteur>
	<auteur id='24' name='Abstract Girl' >
		<photo num='152' cat='abstract' uri='abstract05.jpg' />
		<photo num='450' cat='abstract' uri='koya.jpg' />
		<photo num='750' cat='abstract' uri='recv.jpg' />
		<photo num='450121' cat='abstract' uri='abstractt.jpg' />
	</auteur>
	<auteur id='125' name='Lord of destruction' >
		<photo num='1522' cat='norway' uri='norwayy.jpg' />
		<photo num='1545' cat='norway' uri='nornorw.jpg' />
		<photo num='1111' cat='norway' uri='dark_norway.jpg' />
	</auteur>
</photos>
```

Création d'un fichier PHP nommé photos.php :

```
<?php

$xml = simplexml_load_file('photos.xml');


// print_r($xml); //Pour afficher de manière brute la structure de notre XML.
//Attention chacun des élements est considéré comme un objet

// foreach littéralement "pour chaque", c'est-à-dire une exécution de la boucle
pour chaque élément d'un tableau, une boucle spécialement dédiée aux tableaux !

foreach($xml->auteur as $auteur) {
	foreach($auteur->photo as $photo){
	echo '<img src='.$photo[' uri '].' />' ;
	echo '<br />';
	}

}

?>
```

3.3 APPLICATION 2 : Afficher des photos en créant un fichier XML dynamiquement

Application sous forme d'un exercice en s'appuyant sur l'application 1 et les bases Mysql/PHP.

Création d'une table SQL nommée « photos » dont les champs sont : 'id_photos', 'titre', 'description', 'url'.

Création d'un fichier PHP (le fichier XML va être créé automatiquement) nommé photos_sql.php.

```
<?php

// Connexion à la base //
      $host="mysql5-2";
      $user="nomutilisateur";
      $pswd="******";
      $base="********";

      if (!mysql_connect($host,$user,$pswd)){
            exit("Echec de connexion au serveur...");
      }
      if (!mysql_select_db($base)){
            exit("Echec de sélection de la base...");
      }
      mysql_query('SET NAMES utf8');
?>

// Prologue de la page HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">

<head>
      <META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
      <title>LA BOUTIQUE FR : PHOTOS </title>
      <link rel="stylesheet" href="css/photo.css" />
</head>
<body>

<?php
      // Récupération du contenu dans une base de données
      $sql = 'SELECT fichier FROM photos';

      $req = mysql_query($sql) or die('Erreur SQL !<br>'.
      $sql.'<br>'.mysql_error());

      // prologue et nœud racine du fichier XML qui va être crée

      $xml = '<?xml version="1.0" encoding="UTF-8"?>';
      $xml .= '<photos>';

      // Création d'un tableau afin d'ajouter les photos dans le fichier xml

      while($image = mysql_fetch_assoc($req)){

            $fichier=$image['titre'];     // déclaration
            $fichier2=$image['url'];
            $fichier3=$image['description'];
            // Concaténation dans la variabel $xml
            $xml .= '<photo titre=''.$fichier.''
fichier="http://www.cebastien.fr/fotos/'.$fichier2.'">';
```

christophe.ollivier@virtuoworks.com
VirtuoWorks

```
            $xml .= $fichier3;
            $xml .= '</photo>';
      }
      $xml .= '</photos>'; // balise de fin, celle de la racine
      $fp = fopen("photos2.xml", 'w+'); // Ouverture du fichier xml et création
si non fait
      fputs($fp, $xml); // Ecriture
      fclose($fp);  // fermeture du fichier xml

      echo 'Export XML effectue !<br /><a href="photos2.xml">Voir le
fichier</a>';

// Chargement du fichier xml

$myXml = simplexml_load_file('photos2.xml');
foreach($myXml->photo as $photo){
      echo '<img src='.$photo[fichier].' />' ;
     echo '<br /> ';
}
?>
</body>
</html> // Fin de la page
```

A NOTER : L'utilisation de la méthode **utf8_encode()** permet d'obtenir une cohérence entre l'encodage de mon fichier PHP et l'entête de mon fichier XML.

La fonction PHP fopen(file, mode) utilise différents modes :

‘r' : Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.

'r+' : Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.

'w' : Ouvre en écriture seule, place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

'w+' : Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

'a' : Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

'a+' : Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

fputs() : permet de mettre du contenu dans un fichier dont l'adresse est connue. fputs() est un Alias de fwrite()

fclose() : ferme le fichier ouvert précédemment avec fopen()

4 LES FLUX DE SYNDICATION

4.1 LES FLUX DE DONNEES

Mettre à disposition un flux de syndication permet une lecture du contenu du flux sans avoir à accéder à l'ensemble du site. La lecture peut se faire depuis un client mail avancé, un agrégateur de flux (logiciel sur ordinateur ou application sur smartphone) ou bien un site web faisant de la syndication.

La syndication entre sites web permet notamment de mettre en ligne sur un site A les manchettes des derniers articles publiés sur le site B, sans que les webmasters des sites A ou de B n'aient de mises à jour à faire.

4.2 RSS

Le format **RSS** est un des formats les plus répandus pour la syndication mais ce n'est pas le seul.

Atom est un autre format de syndication sous forme de XML dont le script diffère quelque peu de celui du format RSS.

Un fichier RSS va donc se présenter sous la même forme qu'un fichier XML. D'ailleurs son extension sera .xml

Exemple simple

```
<?xml version="1.0"?>
<rss version="2.0">
       <channel>
           <title>Mon site</title>
           <description>Ceci est un exemple de flux RSS</description>
           <pubDate>Wed, 27 Jul 2005 00:30:30 -0700</pubDate>
           <link>www.monsite.fr</link>
           <item>
               <title>Actualité N°1</title>
               <description>Ceci est ma première actualité</description>
               <pubDate>Wed, 25 Jul 2005 00:30:30 -0700</pubDate>
               <link>www.monsite.fr/actu1</link>
           </item>
           <item>
               <title>Actualité N°2</title>
               <description>Ceci est ma deuxième actualité</description>
               <pubDate>Tue, 19 Jul 2005 04:32:51 -0700</pubDate>
               <link>www.monsite.fr/actu2</link>
           </item>
       </channel>
   </rss>
```

Explication des principales balises

Le contenu d'un document RSS se situe toujours entre les balises **<rss>**. Elles possèdent obligatoirement un attribut version qui spécifie la version à laquelle le document RSS est conforme.

Au niveau suivant de cette balise se trouve une unique balise **<channel>** qui contiendra les métadonnées du flux RSS, obligatoires ou non, ainsi que la liste des contenus.

Métadonnées : trois éléments sont obligatoires :

<title> : Définit le titre du flux

<link> : Définit l'URL du site correspondant au flux

<description> : Décrit succinctement le flux

D'autre éléments optionnels existent comme :

<pubDate> : Définit la date de publication du flux

<image> : Permet d'insérer une image dans le flux (comprend plusieurs balises)

<language> : Définit la langue du flux

<webmaster> : pour le mail du webmaster

<copyright>

Contenu : Description de chaque article

Pour chaque article, une balise **<item>** est ajoutée dans notre document. Dans cette balise se trouvent les données correspondantes à l'actualité sous forme de balise.

Les balises les plus courantes sont:

<title> : Définit le titre de l'actualité.

<link> : Définit l'URL du flux correspondant à l'actualité.

<pubDate> : Définit la date de l'actualité.

<description> : Définit une description succincte de l'actualité.

D'autres balises existent comme:

<author> : Définit l'adresse électronique (mail) de l'auteur.

<category> : Associe l'item à une catégorie.

<comments> : Définit l'URL d'une page de commentaire en rapport avec l'item.

Valider son flux

Pour garantir la bonne tenue de votre flux, pensez à le faire valider, tout comme vous validez vos pages html et xhtml.

Sur le site du W3C : http://validator.w3.org/feed/

Sur un site dédié feedvalidator : http://feedvalidator.org/

4.3 INTEGRATION

Une fois que votre flux de syndication est prêt, vous devez en indiquer la présence sur votre site en ajoutant le code d'intégration dans chaque page.

Il existe 2 manières simples d'intégrer le flux RSS.

Avec la balise <a>

La première méthode consiste à proposer à l'utilisateur de cliquer tout simplement sur un lien (image, texte) qui pointe vers votre fichier XML. Si son navigateur permet la lecture des flux, il lui proposera de s'abonner à ce flux.

```
<a type="application/rss+xml" title="RSS 2.0" href="monFlux.xml"> S'abonner à
mon blog  </a>
```

Dans l'entête de la page

La deuxième méthode permet d'indiquer au navigateur lui-même que cette page ou ce site propose l'adhésion à un flux. L'internaute doit cliquer sur l'icône « rss » dans la barre d'adresse, suivant le même principe que le lien vu ci-dessus dans la première méthode.

```
<head>
<link rel="alternate" type="application/rss+xml" title="RSS 2.0"
href="http://MonSite/dossier/fluxrss.xml" />
</head>
```

Quelques noms d'agrégateurs :

Agrégation en ligne

Bloglines

del.icio.us

netvibes

NewsRSS

Floobby

FeedBurner

Technocrati

Actuello

My Google

Agrégation en local (lecture)

ziepods

itunes

thunderbird

AlertInfo

4.4 CREER UN FLUX AVEC PHP

Nous l'avons vu, un flux RSS est un fichier XML avec certaines balises prédéfinies.

Pour proposer un flux RSS sur votre site dynamique, il faut que ce fichier XML soit généré par un langage dynamique, PHP dans notre cas.

Il y a deux méthodes pour cela :

- Créer de manière dynamique un contenu XML en utilisant un fichier PHP et en appelant une base de données ;
- Écrire un fichier XML via un fichier PHP qui sera généré à chaque mise à jour.

Créer un flux RSS dans un fichier .PHP

Pour l'exemple suivant, nous considérons une table MySql basique, nommée "films" avec 4 champs : un titre, un lien, une description et un identifiant unique.

Nous allons créer un fichier .PHP qui interroge la table MySql, qui récupère les données récentes et les met en forme.

```
<?php

$link = mysql_connect("localhost", "root", "")
    or die("Impossible de se connecter : " . mysql_error());
mysql_select_db("xml") or die("Impossible de sélectionner la base :
".mysql_error());
mysql_query("SET NAMES utf8") or die("Impossible d'exécuter la requête set
names".mysql_error());

      $rss = '<?xml version="1.0"?>
            <rss version="2.0">
                  <channel>
                        <title> Voici un flux rss </title>
                        <link>http://localhost</link>
                        <description>Flux rss du site</description>
                  ';
```

```
      $requete = mysql_query('SELECT * FROM `rss` ORDER BY `date`, `id` DESC
LIMIT 0,3');

      while($article = mysql_fetch_assoc($requete)){
            $rss .= '<item>';
                  $rss .= '<title>'.$article['titre'].'</title>';
                  $rss .= '<link>'.$article['lien'].'</link>';
                  $rss .= '<description>'.$article['desc'].'</description>';
                  $rss .= '<pubDate>'.$article['date'].'</pubDate>';
            $rss .= '</item>';
      }

      $rss .= '</channel>';
      $rss .= '</rss>';

      echo $rss ;

?>
```

Ecrire un flux RSS à partir d'un fichier.PHP

Dans l'exemple précédent, à chaque sollicitation du flux (par les abonnés), le fichier PHP est interprété par le serveur, ce qui peut être très handicapant en cas d'affluence sur votre site.

Une méthode très similaire permet d'écrire un fichier XML en passant par du PHP.

Ce fichier sera mis à jour à chaque lancement de votre page PHP et c'est lui qui sera sollicité par vos abonnés.

Il faut pour cela utiliser trois méthodes PHP :

fopen() : permet l'ouverture d'un fichier ou d'une URL.

Cette méthode utilise différents modes :

'r' : Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.

'r+' : Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.

'w' : Ouvre en écriture seule, place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

'w+' : Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

'a' : Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

'a+' : Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

fputs() : permet de mettre du contenu dans un fichier dont l'adresse est connue. fputs() est un Alias de fwrite()

fclose() : ferme le fichier ouvert précédemment avec fopen()

Le reste du code ne change pas, seule la sortie est différente (plus d'écho).

```
<?php

...
      $rss = '<?xml version="1.0"?><rss version="2.0">'
      ...
      $rss .= '</rss>';

      $fichier = fopen("rss.xml", "w+");
      fputs($fichier, $rss);
      fclose($fichier);

?>
```

5 XML et Référencement

Le XML est également utilisé depuis peu par les sites de recherches pour le référencement, à l'aide d'un outil appelé **sitemap**.

La norme sitemap a été développée par Google pour simplifier le référencement des pages internet d'un site. Il s'agit en fait d'un fichier xml qui référence toutes les adresses des pages d'un site, permettant de leur donner un ordre d'importance dans leur référencement et également une fréquence à laquelle les robots doivent référencer ces pages, accélérant notablement le processus de référencement.

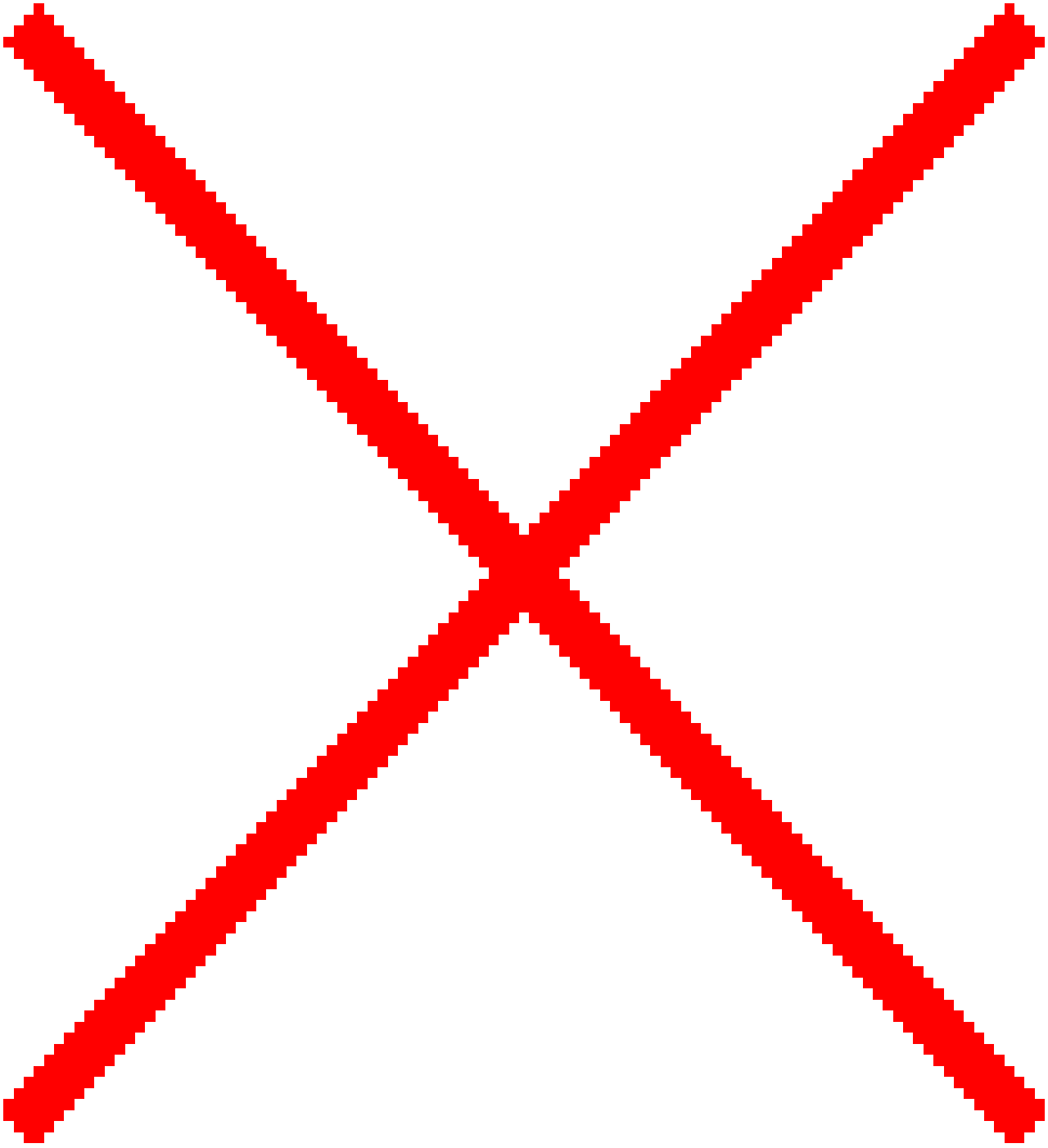
La norme est définie sur le site : http://www.sitemaps.org/

5.1 Syntaxe:

Le format du protocole Sitemap se compose de balises XML. Le fichier doit être enregistré avec un codage UTF-8.

Voici un exemple de plan Sitemap composé d'une seule URL et utilisant toutes les balises facultatives. Ces dernières sont en italique.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.google.com/schemas/sitemap/0.84">
      <url>
            <loc>http://www.example.com/</loc>
            <lastmod>2005-01-01</lastmod>
            <changefreq>monthly</changefreq>
            <priority>0.8</priority>
      </url>
</urlset>
```



Les fichiers sitemap ont une limite de taille: ils ne peuvent contenir que 50000 pages (et peser moins de 10 Mo).

Pour contourner cette limitation (relative en fonction des sites!), il est possible d'avoir recours à un

autre type de fichier: l'index de sitemap.

Il s'agit également d'un fichier xml, qui va répertorier la liste des différents fichiers sitemap du site pour que les robots puissent y accéder.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.google.com/schemas/sitemap/0.84">
      <sitemap>
            <loc>http://www.example.com/sitemap1.xml.gz</loc>
            <lastmod>2004-10-01T18:23:17+00:00</lastmod>
      </sitemap>
      <sitemap>
            <loc>http://www.example.com/sitemap2.xml.gz</loc>
            <lastmod>2005-01-01</lastmod>
      </sitemap>
</sitemapindex>
```

Note: l'extension .gz est un mode de compression utilisé par les systèmes linux, similaire au .zip généralement vu par windows.

6 ANNEXES

6.1 Annexe 1 : Références

Bibliographique

Jean Engels : PHP5 – Cours et exercices

Edition EYROLLES

David Tardiveau : ActionScript 3 – Programmation séquentielle et orientée objet

Edition EYROLLES

Michel Martin : RSS, ATOM. Apprenez à créer des flux de syndication !

Ed:CampusPress, Coll:+ fort en…

Netographie

XML : http://www.w3.org/XML/1999/XML-in-10-points.fr.html

RSS : http://www.xul.fr/xml-rss.html

Character-encoding : http://www.envrac.org/index.php/2006/03/11/58-un-tutoriel-sur-le-character-encoding

http://www.blog-and-blues.org/weblog/2004/08/16/275-encodage-caracteres-xhtml

http://www.unicode.org/standard/translations/french.html

6.2 ANNEXE 2 : XML en 10 points par le W3C

1. XML est une méthode pour structurer des données

On entend par " données structurées " des éléments tels que des feuilles de calcul, des carnets d'adresses, des paramètres de configuration, des transactions financières, des dessins techniques, etc. XML est un ensemble de règles, de lignes directrices, de conventions (quel que soit le nom que vous voulez leur donner) pour la conception de formats texte permettant de structurer des données. XML facilite la réalisation de fichiers qui ne soient pas ambigus, et qui évitent les pièges courants, tels que la non-extensibilité, l'absence de prise en charge de l'internationalisation/localisation et la dépendance par rapport à certaines plates-formes.
XML est conforme à Unicode.

2. XML ressemble un peu à HTML

Comme HTML, XML utilise des balises (des mots encadrés par '<' et '>') et des attributs (de la forme nom="valeur"). Mais alors que HTML définit la signification de chaque balise et de chaque attribut (et souvent la manière dont le texte qu'ils encadrent apparaîtra dans un navigateur), XML utilise les balises seulement pour délimiter les éléments de données et laisse l'entière interprétation des données à l'application qui les lit. En d'autres termes, si vous voyez "<p>" dans un fichier XML, ne supposez pas qu'il s'agit d'un paragraphe. Selon le contexte, cela peut être un prix, un paramètre, une personne, un p... (d'ailleurs, qui a dit qu'il devait s'agir d'un mot commençant par "p" ?).

3. XML est du texte, mais il n'est pas destiné à être lu

Les programmes qui produisent de telles données les stockent souvent sur disque, dans un format binaire ou un format texte. Ce dernier format vous permet, si nécessaire, de consulter les données sans le programme qui les a produites. Les fichiers XML sont des fichiers texte, mais ils sont encore moins destinés à être lus par des individus que les fichiers HTML. Ce sont des fichiers texte, car ils permettent à des experts (tels que les programmeurs) de déboguer plus facilement des applications, et en cas d'urgence, d'utiliser un simple éditeur de texte pour corriger un fichier XML endommagé. Mais les règles des fichiers XML sont beaucoup plus strictes que celles des fichiers HTML. Une balise oubliée ou un attribut sans guillemets rendent le fichier inutilisable, alors qu'avec HTML, de telles pratiques sont souvent explicitement permises, ou au moins tolérées. Les spécifications officielles de XML indiquent que les applications ne sont pas autorisées à essayer de deviner ce qu'a voulu faire le créateur d'un fichier XML endommagé ; si le fichier est endommagé, une application doit s'arrêter immédiatement et émettre une erreur.

4. XML est bavard, mais ce n'est pas un problème

Comme XML est un format texte et qu'il utilise des balises pour délimiter les données, les fichiers XML sont presque toujours d'une taille plus importante que les formats binaires équivalents. Il s'agit là d'une décision prise en toute conscience par les développeurs de XML. Les avantages d'un format texte sont évidents (voir le point 3 ci-dessus), et ses inconvénients Cpeuvent être généralement compensés à un autre niveau. L'espace disque n'est plus aussi coûteux qu'auparavant, et les programmes tels que zip et gzip compressent très bien et très rapidement les fichiers. De plus, les protocoles de communication tels que les protocoles de modem et HTTP/1.1 (le protocole de base du Web) peuvent compresser des données à la volée, ce qui économise de la bande passante aussi efficacement qu'un format binaire.

5. XML est une famille de technologies

Il existe XML 1.0, la spécification qui définit ce que sont les "balises" et les "attributs", mais autour de cette spécification, un nombre de plus en plus important de modules facultatifs fournissant des ensembles de balises et d'attributs ou des lignes directrices pour des tâches particulières ont été définis. C'est, par exemple, le cas de Xlink, qui décrit une méthode standard pour ajouter des liens hypertextes à un fichier XML. XPointer et XFragments sont des syntaxes pour pointer sur des parties d'un document XML. Un XPointer ressemble à un URL, mais au lieu de pointer sur des documents du Web, il pointe sur des éléments de données au sein d'un fichier XML. CSS, le langage des feuilles de style, s'applique à XML de la même façon qu'à HTML. XSL est le langage évolué pour la définition de feuilles de style. Il est basé sur XSLT, un langage de transformation utilisé pour réorganiser, ajouter ou supprimer des balises et des attributs. Le DOM est un ensemble d'appels de fonctions standard pour manipuler des fichiers XML (et HTML) à partir d'un langage de programmation. Les Schémas XML 1 et 2 aident les développeurs à définir précisément leurs propres formats basés sur XML. Plusieurs autres modules et outils sont disponibles ou en cours de développement. Consultez régulièrement la page des rapports techniques du W3C.

6. XML est nouveau, mais pas si nouveau que ça

Le développement de XML a commencé en 1996, et XML est une norme du W3C depuis février 1998, ce qui peut laisser supposer qu'il s'agit d'une technologie plutôt immature. En fait, il ne s'agit pas d'une technologie très nouvelle. Avant XML, il existait SGML, développé au début des années 80, devenu norme ISO depuis 1986 et largement utilisé dans des projets de documentation de taille importante. Et il existait bien sûr HTML, dont le développement a commencé en 1990. Les concepteurs de XML ont simplement pris les meilleures parties de SGML, profité de l'expérience de HTML, et produit une technologie qui n'est pas moins puissante que SGML, mais infiniment plus régulière et plus simple à utiliser. Certaines évolutions, cependant, peuvent être assimilées à des révolutions... Il faut également savoir que SGML est principalement utilisé pour des documentations techniques et beaucoup moins pour d'autres types de données, alors que c'est exactement l'inverse avec XML.

7. XML conduit HTML à XHTML

Une application importante d'XML est le langage XHTML comme successeur de HTML. On retrouve dans XHTML beaucoup d'éléments du langage HTML. La syntaxe a été légèrement modifiée afin de se conformer aux règles d'XML. Plus généralement, un document fondé sur XML hérite sa syntaxe d'XML modulo quelques exceptions : par exemple, XHTML autorise la balise "<p>" mais pas "<r>".

XML ajoute aussi un sens à cette syntaxe : par exemple, XHTML dit que "<p>" signifie "paragraphe" et non "prix" ou "personne" ou etc.

8. XML est modulaire

XML permet de définir un nouveau format de document en associant et en réutilisant d'autres formats. Cependant, deux formats développés indépendamment peuvent avoir des éléments ou attributs de même nom. Il faut alors être très attentif à ne pas confondre les noms : "<p>" signifie-t-il "paragraphe" ou "personne" ? Pour éviter toute confusion lors de l'association de noms identiques, XML fournit un mécanisme d'espaces de nom. XSL et RDF sont de bons exemples de technologies fondées sur XML qui utilisent les espaces de nom. XML Schema a été conçu pour répercuter cette fonctionnalité modulaire pour la définition des structures XML, puisqu'il est facile de combiner deux schémas pour en produire un troisième qui sera associé au document fusionné.

9. XML est le fondement de RDF et du Web Sémantique

RDF, la norme du W3C pour les métadonnées, est un texte au format XML qui autorise la description de ressources et les applications métadonnées, tels que recueils de musiques, albums photos, et bibliographies. Par exemple, RDF pourrait vous laisser identifier des personnes dans un album photo sur le Web en utilisant des informations puisées dans une liste de contacts ; puis, votre messagerie pourrait automatiquement envoyer un message prévenant les personnes que leurs photos sont sur le Web. Tout comme les éléments HTML intégrés, comme les menus et formulaires, RDF intègre les applications et les agents en un Web Sémantique. De la même manière que les humains ont besoin de se mettre d'accord sur les mots qu'ils utilisent en communicant entre eux, les machines ont aussi besoin de mécanismes pour communiquer efficacement. Les descriptions formelles de terminologies dans un domaine particulier (la grande distribution ou l'industrie, par exemple) sont appelées ontologies, et sont une partie nécessaire du Web Sémantique. RDF, ontologies, et la représentation des connaissances sont tous des sujets décrits dans l'activité Web Sémantique.

10. XML est libre de droits, indépendant des plates-formes et correctement pris en charge

En choisissant XML pour un projet, vous bénéficiez d'un ensemble important et sans cesse croissant d'outils (et il est possible que l'un d'eux remplisse déjà la fonction dont vous avez besoin !) et d'ingénieurs expérimentés dans cette technologie. Opter pour XML, c'est un peu comme choisir SQL pour des bases de données : vous devez encore construire votre propre base de données et vos propres programmes ou procédures qui la manipulent, mais un grand nombre d'outils et d'individus peuvent vous y aider. Et comme XML est libre de droits, vous pouvez vous en servir pour construire votre propre logiciel sans avoir à payer quoi que ce soit à qui que ce soit. Sa prise en charge est importante, et ne cesse de croître, ce qui signifie également que vous n'êtes pas lié à un seul fournisseur. XML n'est pas toujours la meilleure solution, mais il vaut toujours la peine d'être pris en considération.