Un livre de Wikilivres.

Programmation XML

Introduction

XML est l'acronyme d'*eXtended Markup Language*, soit « langage de balise étendu » en français. XML permet la structuration et la manipulation de toutes sortes de données dans un format se voulant le plus indépendant possible des plates-formes et des logiciels.
La technologie XML vise à devenir le standard d'échange principal entre systèmes hétérogènes. XML hérite de SGML et à ce titre est un cousin de HTML. C'est un langage de balises comme HTML mais extensible et strict.
À partir de XML, il est possible de dériver des langages spécialisés comme XHTML ou MathML, ou tout autre langage, et ceci grâce au fait que les noms des balises peuvent être réinventés (étendus) en fonction des besoins.

Histoire de XML

- Juillet 1997 : Création du groupe de travail XML par le W3C
- 10 février 1998 : Publication de la spécification XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210) [archive]
- 4 février 2004 : XML (3eme édition) 1.0 (http://www.w3.org/TR/REC-xml/) [archive]
- 4 février 2004 : spécification XML 1.1 (http://www.yoyodesign.org/doc/w3c/xml11/index.html) [archive] (français)

Structure d'un document XML

Un document XML est fondamentalement de type **texte**, par opposition à d'autres structures informatiques qui peuvent être de type binaire (programmes exécutables, base de données). Ce texte est généralement un texte unicode[1], souvent au format UTF-8. Un simple éditeur de texte compatible Unicode/UTF8, est donc capable de créer, d'ouvrir ou de modifier un document XML stocké sur disque.
Mais un document XML n'existe pas uniquement sous forme de fichier. Il peut exister aussi temporairement en mémoire comme format de transfert de données entre deux applications logicielles (par exemple sous forme de requête html POST pour transmettre un formulaire fraîchement tapé par l'internaute).

Donc, le document XML est un **format** dont nous allons décrire la structure :

Éléments d'un document XML

Balisage

Un document XML est constitué de balises. Une balise est une chaîne de caractères constituant un identificateur et placé entre un caractère < et un caractère >.

Il existe différentes sortes de balises :

- Les balises ouvrantes `<item>`
- Les balises fermantes `</item>`
- Les balises vides `<nop/>`
- Les instructions de traitement `<?xml-stylesheet ... ?>`
- Les commentaires `<!-- J'explique -->`
- Les sections CDATA (Character DATA) `<![CDATA[ Contenu de la section ]]>`

Conventions de nommage des balises

- Les noms ne peuvent pas commencer par la chaîne de caractères « xml », dans n'importe quelle combinaison de casse (ou Xml, ou XML…) ;
- les noms sont sensibles à la casse (différentiation minuscules/majuscules) ;
- les noms ne peuvent pas débuter par un nombre ou un signe de ponctuation ;
- les noms ne peuvent pas contenir d'espaces ;
- éviter les caractères accentués et les opérateurs, virgules, point-virgules…

Les balises ouvrantes

Une balise ouvrante est un élément délimitant une section. La section se termine ensuite par une balise fermante.

Une balise ouvrante est une chaîne composée de chiffres, de lettres et des caractères - et _. Cette chaîne est comprise entre un < et un >.

Exemple :

```
<balise>
```

Les balises fermantes

Une balise fermante reprend l'identificateur de la balise ouvrante, en le précédent d'un caractère « barre de fraction » *(slash)*. Exemple :

```
</balise>
```

Les balises vides

Une balise « vide » est le résultat de la contraction d'une balise ouvrante et d'une balise fermante ne contenant pas de données.

```
<div></div> donne <div />
```

Les attributs

Les balises ouvrantes et les balises vides peuvent être enrichies avec des attributs.

```
<root>
     <item id="67af65c75b" date="22/01/2006">Contenu de l'élément</item>
     <break mode="immediat"/>
</root>
```

Dans l'exemple ci-dessus, la balise ouvrante **item** est munie de deux attributs, **id** et **date**. La balise vide **break** est munie de l'attribut **mode**.

Les instructions de traitement

Une instruction de traitement est destinée à être lue et comprise par un programme spécifique, afin de permettre le traitement du document XML.

Une instruction de traitement commence par un point d'interrogation « ? ».

Exemple :

```
<?php echo date("d/m/Y") ?>
```

Voir aussi : Le point sur les instructions de traitement (http://www.yoyodesign.org/doc/w3c/xml11/index.html#sec-pi) [archive] sur *yoyodesign.org*

Les commentaires

Les commentaires sont des balises qui ne sont pas interprétées par l'application qui traite le document XML.

Un commentaire est une balise commençant par **<!--** et terminant par **-->**.

Exemple :

```
<!-- Ceci est un commentaire utilisé dans les langages web  -->
```

Les entités

XML 1.0 définit un petit nombre d'entités utilisables dans le document sans nécessiter de déclaration dans le doctype.

Ces entités sont :

Entité	Caractère	Description
<	<	Symbole « inférieur »
>	>	Symbole « supérieur »
&	&	Esperluette, « et » commercial
'	'	guillemet simple
"	"	guillemet double

Toujours sans déclaration de doctype spécial, on peut écrire une entité avec « & », « # », une référence unicode [1] et un « ; ».

Entité	Caractère	Description
π	π	pi
β	β	bêta

Exemple

```
<balise>Etudions la balise &amp;lt;p&amp;gt;</balise> Etudions la balise &lt;p&gt;
```

Structure d'un document XML

En-tête

Les trois constituants de l'en-tête que nous allons décrire ne sont pas obligatoires. Ils sont employés en cas de besoin pour fixer l'encodage, définir le doctype et/ou associer une feuille de style.

Version, encodage

Il est souhaitable que le document XML commence par l'indication de la version du langage et le jeu de caractères utilisé. C'est souvent utf-8[1]. Il est également souhaitable, si le document est stocké sur disque, que l'encodage du fichier soit le même que celui mentionné par l'attribut *encoding*…

```
<?xml version="1.0"?> <!-- minimal -->

    <!-- ou -->

<?xml version="1.0" encoding="utf-8"?>
```

Référence au DTD

La deuxième information utile est la mention du doctype. Certains documents XML en ont impérativement besoin, d'autres peuvent s'en passer.

```
<!DOCTYPE root[
<!ELEMENT root (premier,deuxieme+)>
<!ELEMENT premier (#PCDATA)>
<!ELEMENT deuxieme (#PCDATA)>
]>
```

En analysant ce doctype, l'analyseur syntaxique (le parser, p.ex. : via Cooktop (http://www.xmlcooktop.com/) [archive] ou encore XML tools pour Notepad++ (http://sourceforge.net/projects/npp-plugins/files/XML%20Tools/) [archive]) est en mesure de considérer le document XML comme valide.

Mention d'une feuille de style

Il est possible d'associer un document XML à une feuille de style css ou xsl. Si la fonctionnalité fait réfléchir, au début, il y a fort à parier que vous en abandonnerez la pratique par la suite. En effet, la philosophie XML va à l'opposé.

```
<?xml-stylesheet type="text/xsl" href="../../style.xsl"?>
        <!-- ou -->
<?xml-stylesheet type="text/css" href="../../style.css"?>
```

Une seule racine

Une balise a un statut spécial, c'est la racine du document XML. C'est à partir de cette balise que se développe l'arborescence du document.

Règles de mise en œuvre des balises

- Toute balise ouverte doit être fermée.
- Les balises doivent être correctement imbriquées.
- Les balises sont imbriquables hiérarchiquement sans limitation mais il ne doit y avoir qu'un seul élément à la racine.

Le corps d'un document XML est une arborescence d'éléments (balises) imbriqués, avec un élément racine unique.

```
<root>
    <balise1>
         <balise2>
         </balise2>
    </balise1>
</root>
```

Quand toutes ces règles sont respectées, on obtient un document XML « **bien formé** ». Dans ce cas, un navigateur comme Internet explorer ou Firefox peut l'ouvrir. Dans le cas contraire, le navigateur affiche un message d'erreur.

Exemple de structure simple

```
<?xml version="1.0" ?>
<root>
     <items>
         <item no="1">Premier élément</item>
         <item no="2">Autre chose</item>
         <item no="3">Troisième élément</item>
         <item no="4">Quatrième élément</item>
     </items>
</root>
```

Références

1. Unicode est un standard qui permet une représentation abstraite et universelle du texte. Pour découvrir ce sujet, on pourra lire le wikilivre: À la découverte d'Unicode.
2. http://www.w3schools.com/xml/xml_attributes.asp

Liens

Traduction française de la spécification XML1.1 (http://www.yoyodesign.org/doc/w3c/xml11/index.html) [archive]

Espace de nom

Les espaces de noms (ou *namespace*) sont destinés à lever les ambiguïtés éventuelles des intitulés de balise, au moyen d'un identifiant de ressource unique (URI).

Syntaxe

L'exemple ci-dessous contient deux déclarations, la première par défaut et la seconde associant le préfixe « ip » à l'URI des adresses IP, pour les distinguer des adresses postales :

```
<adresse
 xmlns="http://www.example.com/adresses_postales"
 xmlns: ip="http://www.example.com/adresses_ip">
```

Définition de Type Document

Introduction

Le DTD permettent de décrire la forme que doit avoir un document XML pour être valide. Les DTD ont été remplacées par XML Schema qui est bien plus puissant et expressif, aussi nous vous encourageons, pour

tout nouveau projet de favoriser systématiquement l'utilisation de Schema aux DTD.

Nous abordons toutefois les DTD ici, afin que vous puissiez en comprendre une si besoin. Notamment, vous pourriez être amener à transformer une DTD en un Schema pour utiliser cette nouvelle technologie.

Spécification d'une DTD

Contrairement aux Schema, la DTD n'est pas un dialecte XML, c'est un langage spécifique.

La DTD n'est pas obligatoire. Elle peut être **interne** au document XML ou bien **externe** (dans un fichier à l'extension « .dtd »). On utilise une DTD externe lorsque l'on veut la partager entre plusieurs fichiers XML.

Exemple de déclaration d'une DTD externe dans un fichier XML :

```
<!DOCTYPE repertoire SYSTEM "repertoire.dtd">
```

La DTD commence par la balise :

```
<!DOCTYPE nom-element-racine [
```

et se termine par :

```
]>
```

La DTD doit être placée après la déclaration initiale XML.

Les commentaires, comme en XML sont utilisés de la manière suivante :

```
<!-- commentaires -->
```

Éléments

Chaque élément de la DTD est défini de la manière suivante[1] :

```
<!ELEMENT nom-element qualificateur>
```

Les qualificateurs possibles sont :

- **ANY** : la structure de l'élément est libre
- **EMPTY** : l'élément est vide
- **#PCDATA(Parsed Character Data)** : donnée textuelle analysable

On peut préciser ces qualificateurs grâce aux éléments suivants :

- un et un seul sous-élément :

(un seul téléphone par personne)

```
<!ELEMENT personne(telephone)>
```

- 0 ou plusieurs sous-éléments :

(une personne peut avoir 0, 1 ou plusieurs téléphones)

```
<!ELEMENT personne(telephone*)>
```

- 1 ou plusieurs éléments :

(une personne doit avoir au moins un téléphone)

```
<!ELEMENT personne(telephone+)>
```

- 0 ou 1 seul élément :

(une personne a au maximum 1 téléphone, elle peut ne pas en avoir)

```
<!ELEMENT personne(telephone?)>
```

- Plusieurs sous-éléments différents :

(une personne a un nom et un téléphone)

```
<!ELEMENT personne(nom, telephone)>
```

- Des sous-éléments identiques ou non :

(une personne a un nom et deux numéros de téléphone)

```
<!ELEMENT personne(nom, telephone, telephone)>
```

- Un sous-élément OU (exclusif) un autre :

(un ou deux téléphone par personne)

```
<!ELEMENT personne(telephone|(telephone, telephone))>
```

Exemple du stockage d'un répertoire de services. Chaque service a un nom et peut avoir un ou plusieurs téléphone(s) :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE repertoire [
        <!ELEMENT repertoire (service)*>
        <!ELEMENT service (nom, tel*)>
```

```
        <!ELEMENT nom (#PCDATA)>
        <!ELEMENT tel (#PCDATA)>
]>
<repertoire>
   <service>
      <nom>pompiers</nom>
      <tel>18</tel>
   </service>
   <service>
      <nom>police</nom>
      <tel>17</tel>
      <tel>22</tel>
   </service>
   <service>
      <nom>samu</nom>
      <tel>15</tel>
   </service>
</repertoire>
```

Attributs

Ils sont déclarés avec `ATTLIST`. Par exemple, on définit un type de paiement par défaut en liquide :

```
<!ATTLIST paiement type CDATA (chèque|liquide) "liquide" >
```

Valeurs des attributs

Les valeurs d'attributs (attribute-value[2]) sont :

Valeur	Explication
value	valeur par défaut de l'attribut
#REQUIRED	attribut requis
#IMPLIED	attribut optionnel
#FIXED value	attribut de valeur fixe

Types des attributs

Type	Description
`CDATA`	valeur de la donnée
`(en1|en2|..)`	valeur dans la liste énumérée
`ID`	identifiant unique
`IDREF`	identifiant d'un autre élément
`IDREFS`	liste d'autres identifiants
`NMTOKEN`	nom d'un XML valide
`NMTOKENS`	liste de noms d'XML
`ENTITY`	entité
`ENTITIES`	liste d'entités
`NOTATION`	nom d'une notation
`xml:`	valeur d'un xml prédéfini

Entités

Déclarer une entité permet de l'intégrer ensuite en appelant son nom précédé d'un et commercial, pourcentage ou guillemet ('&', '%' ou ' " '[3]) et suivi d'un point virgule (';'). Ex :

```
<!ENTITY intro! SYSTEM "https://fr.wikibooks.org/wiki/Programmation_XML/Introduction.xml">

<b>
 &intro;
</b>
```

Exemple

Extrait du cahier des charges XHTML[4] concernant les hyperliens :

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
...
<!ELEMENT link EMPTY>
<!ATTLIST link
  %attrs;
  charset     %Charset;      #IMPLIED
  href        %URI;          #IMPLIED
  hreflang    %LanguageCode; #IMPLIED
  type        %ContentType;  #IMPLIED
  rel         %LinkTypes;    #IMPLIED
  rev         %LinkTypes;    #IMPLIED
  media       %MediaDesc;    #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >
```

Références

1. http://www.w3schools.com/dtd/dtd_elements.asp
2. http://www.w3schools.com/dtd/dtd_attributes.asp

3. http://xmlwriter.net/xml_guide/entity_declaration.shtml
4. http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd

Interprétation du XML

Il existe de nombreuses API permettant d'interpréter un document XML. La plupart sont conçues pour un langage particulier :

- Dom4J (Java)
- JDOM (Java)
- SimpleXML (PHP)
- PyXML (Python)[1]

Tous ces interpréteurs peuvent être de deux types normalisés :

- DOM (créé par le W3C) : un arbre de nœuds complet est construit à partir du document XML.
- SAX (créé par David Megginson) : l'interpréteur appelle certaines fonctions de l'application en fonction de chacun des éléments de base rencontrés dans le document XML (ouvertures de balise, fermetures, données, commentaires, ...).

Références

1. http://pyxml.sourceforge.net/topics/

Voir aussi

- Programmation Python/XML
- Programmation Java/XML

SAX

La méthode SAX (Simple API for XML) analyse les éléments XML au fur et à mesure, par opposition au DOM. Ce qui est plus pratique pour des fichiers de grande taille par exemple.

Voir aussi

- Programmation Python/XML#La méthode SAX
- Programmation Java/XML#La méthode SAX

DOM

La méthode DOM (Document Object Model) analyse les éléments XML en construisant une arborescence de la structure d'un document et de ses éléments. Cela nécessite donc de le lire en entier ce qui peut poser un problème de saturation quand il est de grande taille.

Voir aussi

- Programmation PHP/La librairie DOM

XPath

Syntaxe

Le XPath est un langage de sélection de différents types d'objets XML, appelés « nœuds »[1]. Un ensemble de nœuds est appelé « contexte ».

Le XPath se présente sous la forme de chemins composés de[2] :

Sélecteur	Notes
nom du nœud	Sélectionne ce qui est compris dans le nœud nommé.
/	Sélectionne en partant du nœud racine (chemin absolu).
//	Sélectionne en partant du nœud courant, peu importe le reste de l'emplacement.
.	Sélectionne à partir du nœud courant (chemin relatif). = `self::node()`
..	Sélectionne à partir du parent du nœud courant. = `parent::node()`
@	Sélectionne les attributs. = `attribute::`
|	Opérateur de sélection multiple.

> **Remarque :** Il existe un interpréteur en ligne pour réaliser les exemples décrits ci-dessous : http://www.xpathtester.com/.
> La classe http://www.php.net/manual/fr/domxpath.query.php permet de les programmer.

Ces expressions sont appelées « chemin de localisation », composés d'un ou plusieurs « pas de localisation » (ou « étapes ») séparés par des « / ». Les pas de localisation ont chacun trois composants :

1. Un axe (parent, descendant…).
2. Un test de nœud (nom ou fonction désignant les nœuds).
3. Des prédicats (entre crochets).

Axes

Pour décrire les relations entre les nœuds, XPath utilise le vocabulaire suivant :

Axe	Abréviation	Notes
`ancestor`		ancêtre
`ancestor-or-self`		ancêtre ou soi
`attribute`	`@`	attribut, `@abc` signifie `attribute::abc`
`child`		enfant, `xyz` signifie `child::xyz`
`descendant`		
`descendant-or-self`	`//`	`//` signifie `/descendant-or-self::node()/`
`following`		suivant
`following-sibling`		frère suivant
`namespace`		espace de noms
`parent`	`..`	`..` signifie `parent::node()`
`preceding`		précédent
`preceding-sibling`		
`self`	`.`	soi, `.` signifie `self::node()`

Tests de nœuds

Soit l'espace de nom `ns` :

- `//ns:*` sélectionne tout les éléments du namespace.
- `//ns:nom` récupère tous les éléments du namespace nommés "nom".

Tests	Notes
comment()	trouve tous les commentaires (ex : `<!-- commentaire 1 -->`)
text()	trouve un nœud texte, (ex : `hello world` dans `<k>hello<m> world</m></k>`)
processing-instruction()	trouve les instructions de traitement (ex : `//processing-instruction('php')` `trouve <?php echo $a; ?>`)
node()	trouve tous les nœuds.

Prédicats

Les prédicats sont des fonctions filtrant les nœuds évalués à *false*, qui se placent à la fin des sélections[3] :

Par exemple, les quatre requêtes ci-dessous renvoie le même résultat (si la branche 2 est la dernière comme dans l'exemple en bas de cette page) :

```
//branche[2]
//branche[@nom="branche2"]
/tronc/branche[last()]
/tronc/branche[position()=2]
```

Prédicats	Notes
`last()`	renvoie le dernier nœud de la sélection
`position()`	renvoie le nœud situé à la position précisée
`count(contexte)`	renvoie le nombre de nœuds en paramètre
`starts-with(chaine1, sous-chaine2)`	renvoie *true* si le premier argument commence avec le second
`contains(botte_de_foin, aiguille)`	renvoie *true* si le premier argument contient le second
`sum(contexte)`	renvoie la somme des valeurs numériques des nœuds en paramètre
`floor(nombre)`	renvoie le nombre arrondi à l'entier inférieur
`ceiling(nombre)`	renvoie le nombre arrondi à l'entier supérieur
`round(nombre)`	renvoie le nombre arrondi à l'entier le plus proche

Exemples

Soit l'arborescence suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<tronc nom="tronc1">
        <!-- commentaire 1 -->
        <branche nom="branche1" epaisseur="gros">
                <brindille nom="brindille1">
                        <!-- commentaire 2 -->
                        <feuille nom="feuille1" couleur="marron" />
                        <feuille nom="feuille2" poids="50" />
                        <feuille nom="feuille3" />
                </brindille>
                <brindille nom="brindille2">
                        <feuille nom="feuille4" poids="90" />
                        <feuille nom="feuille5" couleur="violet" />
                </brindille>
        </branche>
        <branche nom="branche2">
                <brindille nom="brindille3">
                        <feuille nom="feuille6" />
                </brindille>
                <brindille nom="brindille4">
                        <feuille nom="feuille7" />
                        <feuille nom="feuille8" />
                        <feuille nom="feuille9" couleur="noir" />
                        <feuille nom="feuille10" poids="100" />
                </brindille>
        </branche>
        <branche nom="branche3">
                <brindille nom="brindille5">
                </brindille>
        </branche>
</tronc>
```

Abréviations

- Sélection 1 : toutes les <feuille> d'une <brindille> contenue dans une <branche>, descendant du <tronc>, issu de la racine.

1. Abrégé : `/tronc/branche/brindille/feuille`
2. Non abrégé : `/child::tronc/child::branche/child::brindille/child::feuille`

- Sélection 2 : la <branche> dont l'attribut "nom" est "branche3", enfant du <tronc>, inclue dans la racine.
  1. Abrégé : `/tronc/branche[@nom='branche3']`
  2. Non abrégé : `/child::tronc/child::branche[attribute::nom='branche3']`
- Sélection 3 : toutes les brindilles ont au moins une feuille.
  1. `//brindille[feuille]`
- Sélection 4 : dernière branche du tronc.
  1. `//tronc/branche[last()]`
- Sélection 5 : tous les noms des brindilles qui n'ont pas de feuille.
  1. `//brindille[not(feuille)]/@nom`

PHP

Créer le .php suivant à côté du tronc.xml publié ci-dessus.

```php
<?php
  $file = 'tronc.xml';
  $xpath = "/tronc/branche/brindille/feuille[last()]";
  if(file_exists($file)) {
    $xml = simplexml_load_file($file);
    if($result = $xml->xpath($xpath)) {
                print 'Résultats :';
                var_dump($result);
    } else {
      echo 'Syntaxe invalide.';
        }
  }
  else
    exit("Le fichier $file n'existe pas.");
?>
```

eXist ou BaseX

*Pour plus de détails voir : **XQuery**.*

Références

1. http://www.w3schools.com/dom/dom_nodetype.asp
2. http://www.w3schools.com/xpath/xpath_syntax.asp
3. http://www.w3.org/TR/xpath#corelib

Voir aussi

- http://b3d.bdpedia.fr/query.html#s1-xpath-trouver-son-chemin-xml : explications avec vidéo en français

XSLT

Syntaxe de base

XSLT signifie *Extended Stylesheet Language Transformations*. Il s'agit d'un langage qui permet de transformer un document XML en un autre format en s'appuyant sur XPath. On peut par exemple créer une page HTML, un fichier PDF ou autre à partir d'un fichier XML.

Le langage XSLT se base sur la notion de *règles de transformation*. Pour chaque balise XML, on définit comment son contenu doit apparaitre dans le document produit. Ces règles sont dans des balises <xsl: et on les enregistre dans des documents .xsl.

> **Remarque :** Les exemples ci-dessous sont vérifiables avec l'outil http://www.w3schools.com /xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog

Attributs

Attributs	Note
`match`	désigne le nœud auquel s'applique le code.
`select`	ce qu'il faut trier dans le résultat.
`order`	classe les résultats (`ascending` ou `descending`)
`case-order`	classe les résultats en distinguant les lettres capitales (`upper-first` ou `lower-first`)
`data-types`	convertit le type des données (ex : `string`, `number`, `boolean`)

Exemple

```xml
<?xml version="1.0" ?>
<persons>
  <person username="JS1">
    <name>John</name>
    <family-name>Smith</family-name>
  </person>
  <person username="MI1">
    <name>Morka</name>
    <family-name>Ismincius</family-name>
  </person>
</persons>
```

+

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

  <xsl:template match="/persons">
    <root>
      <xsl:apply-templates select="person"/>
    </root>
  </xsl:template>

  <xsl:template match="person">
    <name username="{@username}">
      <xsl:value-of select="name" />
    </name>
  </xsl:template>
</xsl:stylesheet>
```

=

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name username="JS1">John</name>
  <name username="MI1">Morka</name>
</root>
```

PHP

Depuis PHP5 l'extension est déjà inclue[1] dans le fichier PHP.ini :

```
extension=php_xsl.dll
```

Il n'y a donc rien a installer pour charger les deux fichiers publiés ci-dessus :

```
<?php
$XML = new DOMDocument();
$XML->load('note.xml');

$XSL = new DOMDocument();
$XSL->load('note.xsl', LIBXML_NOCDATA);

$xslt = new XSLTProcessor();
$xslt->importStylesheet($XSL);

print $xslt->transformToXML($XML);
?>
```

Syntaxe avancée

Éléments

Éléments XSL[2] [3]	Description	Catégorie
apply-imports	Applies a template rule from an imported stylesheet	instruction
apply-templates	Applies a template rule to the current element or to the current element's child nodes	instruction
attribute	Adds an attribute	instruction
attribute-set	Defines a named set of attributes	top-level-element
call-template	Calls a named template	instruction
choose	Used in conjunction with <when> and <otherwise> to express multiple conditional tests	instruction
comment	Creates a comment node in the result tree	instruction
copy	Creates a copy of the current node (without child nodes and attributes)	instruction
copy-of	Creates a copy of the current node (with child nodes and attributes)	instruction
decimal-format	Defines the characters and symbols to be used when converting numbers into strings, with the format-number() function	top-level-element
element	Creates an element node in the output document	instruction
fallback	Specifies an alternate code to run if the processor does not support an XSLT element	instruction
for-each	Loops through each node in a specified node set	instruction
if	Contains a template that will be applied only if a specified condition is true	instruction
import	Imports the contents of one stylesheet into another. Note: An imported stylesheet has lower precedence than the importing stylesheet	top-level-element
include	Includes the contents of one stylesheet into another. Note: An included stylesheet has the same precedence as the including stylesheet	top-level-element
key	Declares a named key that can be used in the stylesheet with the key() function	top-level-element
message	Writes a message to the output (used to report errors)	instruction
namespace-alias	Replaces a namespace in the stylesheet to a different namespace in the output	top-level-element
number	Determines the integer position of the current node and formats a number	instruction
otherwise	Specifies a default action for the <choose> element	instruction
output	Defines the format of the output document	top-level-element
param	Declares a local or global parameter	top-level-element

Éléments XSL[2] [3]	Description	Catégorie
preserve-space	Defines the elements for which white space should be preserved	top-level-element
processing-instruction	Writes a processing instruction to the output	instruction
sort	Sorts the output	instruction
strip-space	Defines the elements for which white space should be removed	top-level-element
stylesheet	Defines the root element of a stylesheet	top-level-element
template	Rules to apply when a specified node is matched	top-level-element
text	Writes literal text to the output	instruction
transform	Defines the root element of a stylesheet	top-level-element
value-of	Extracts the value of a selected node	instruction
variable	Declares a local or global variable	top-level-element or instruction
when	Specifies an action for the <choose> element	instruction
with-param	Defines the value of a parameter to be passed into a template	instruction

Fonctions

Nom[4]	Description
current()	Returns the current node
document()	Used to access the nodes in an external XML document
element-available()	Tests whether the element specified is supported by the XSLT processor
format-number()	Converts a number into a string
function-available()	Tests whether the element specified is supported by the XSLT processor
generate-id()	Returns a string value that uniquely identifies a specified node
key()	Returns a node-set using the index specified by an <xsl:key> element
system-property	Returns the value of the system properties
unparsed-entity-uri()	Returns the URI of an unparsed entity

Fonctions héritées de XPath

Nom[5]	Description	Syntaxe
count()	Returns the number of nodes in a node-set	number=count(node-set)
id()	Selects elements by their unique ID	node-set=id(value)
last()	Returns the position number of the last node in the processed node list	number=last()
local-name()	Returns the local part of a node. A node usually consists of a prefix, a colon, followed by the local name	string=local-name(node)
name()	Returns the name of a node	string=name(node)
namespace-uri()	Returns the namespace URI of a specified node	uri=namespace-uri(node)
position()	Returns the position in the node list of the node that is currently being processed	number=position()

Fonctions chaines

Nom	Description	Exemple
Concat()	Returns the concatenation of all its arguments	string=concat(val1, val2, ..) Example: concat('The',' ','XML') Result: 'The XML'
contains()	Returns true if the second string is contained within the first string, otherwise it returns false	bool=contains(val,substr) Example: contains('XML','X') Result: true
normalize-space()	Removes leading and trailing spaces from a string	string=normalize-space(string) Example: normalize-space(' The XML ') Result: 'The XML'
starts-with()	Returns true if the first string starts with the second string, otherwise it returns false	bool=starts-with(string,substr) Example: starts-with('XML','X') Result: true
string()	Converts the value argument to a string	string(value) Example: string(314) Result: '314'
string-length()	Returns the number of characters in a string	number=string-length(string) Example: string-length('Beatles') Result: 7
substring()	Returns a part of the string in the string argument	string=substring(string,start,length) Example: substring('Beatles',1,4) Result: 'Beat'
substring-after()	Returns the part of the string in the string argument that occurs after the substring in the substr argument	string=substring-after(string,substr) Example: substring-after('12/10','/') Result: '10'
substring-before()	Returns the part of the string in the string argument that occurs	string=substring-before(string,substr)

Nom	Description	Exemple
	before the substring in the substr argument	Example: substring-before('12/10','/') Result: '12'
translate()	Takes the value argument and replaces all occurrences of string1 with string2 and returns the modified string	string=translate(value,string1,string2) Example: translate('12:30',':','!') Result: '12!30'

Fonctions nombres

Nom	Description	Exemple
ceiling()	Returns the smallest integer that is not less than the number argument	number=ceiling(number) Example: ceiling(3.14) Result: 4
floor()	Returns the largest integer that is not greater than the number argument	number=floor(number) Example: floor(3.14) Result: 3
number()	Converts the value argument to a number	number=number(value) Example: number('100') Result: 100
round()	Rounds the number argument to the nearest integer	integer=round(number) Example: round(3.14) Result: 3
sum()	Returns the total value of a set of numeric values in a node-set	number=sum(nodeset) Example: sum(/cd/price)

Fonctions booléennes

Nom	Description	Exemple
boolean()	Converts the value argument to Boolean and returns true or false	bool=boolean(value)
false()	Returns false	false() Example: number(false()) Result: 0
lang()	Returns true if the language argument matches the language of the xsl:lang element, otherwise it returns false	bool=lang(language)
not()	Returns true if the condition argument is false, and false if the condition argument is true	bool=not(condition) Example: not(false())
true()	Returns true	true() Example: number(true()) Result: 1

Références

1. http://www.php.net/manual/fr/book.xsl.php
2. http://www.w3schools.com/xsl/xsl_w3celementref.asp
3. http://www.w3.org/TR/xslt#element-syntax-summary)
4. http://www.w3schools.com/xsl/xsl_functions.asp
5. http://www.w3schools.com/xsl/xsl_functions.asp

Voir aussi

- CSS

XML Schema

Syntaxe

XML Schema est un langage de description de format, permettant de définir la structure et le type de contenu d'un document XML (comme pour créer une base de données).

Ces règles sont dans des balises <xs: ou <xsd: et on les enregistre dans des documents .xsd (XML Schema Definition).

Exemple

```
<xsd:complexType name="Type_de_genre">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string"/>
    <xsd:element name="description" type="xsd:string"/>
    <xsd:element name="film" type="Type_de_film" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<!-- Type_de_film est un enfant de Type_de_genre qui l'appelle par son nom -->
<xsd:complexType name="Type_de_film">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string"/>
    <xsd:element name="acteur" type="Type_d_acteur" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<!-- Exemple de restriction en regex -->
<xsd:simpleType name="Type_d_adresse_email">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[^@]+@[^\.]+\..+"/>
  </xsd:restriction>
</xsd:simpleType>
```

Voir aussi

- Catégorie:Expressions rationnelles

XLink

XPointer

XForms

XQuery

XQuery est une recommandation W3C de sélection de données depuis des documents et bases de données basée sur XML.

Sommaire

1. HelloWorld
2. Chargement de données
3. Expression FLWOR
4. Séquences
5. Analyse syntaxique CSV
6. Exemples XPath
7. Expressions rationnelles
8. Searching multiple collections
9. Getting URL Parameters
10. Getting POST Data
11. Checking for Required Parameters
12. Displaying Lists
13. Extracting data from XHTML files
14. Displaying data in HTML Tables
15. Limiting Result Sets
16. Filtering Words
17. Saving and Updating Data
18. Quantified Expressions
19. Dates and Time
20. Chaining Web Forms

Exemples intermédiaires

1. Using XQuery Functions
2. Creating XQuery Functions
3. Returning the Longest String
4. Net Working Days
5. Tag Cloud
6. String Analysis
7. Manipulating URIs
8. Parsing Query Strings
9. Splitting Files
10. Filling Portlets
11. Filtering Nodes
12. Limiting Child Trees
13. Higher Order Functions
14. Timing Fibonacci algorithms
15. Using Intermediate Documents
16. Formatting Numbers
17. Uploading Files
18. TEI Concordance
19. Queries on Tables
20. Namespace Constructors

Recherche

1. Introduction to XML Search
2. Basic Search
3. Searching,Paging and Sorting
4. Keyword Search

5. Employee Search
6. Incremental Search of the Chemical Elements
7. Lucene Search
8. Incremental Searching
9. Advanced Search
10. Open Search
11. Auto-generation of Index Config Files

Interaction

1. Adder
2. Simple XForms Examples
3. Navigating Collections
4. Envoyer des e-mails

Vues personnalisées

1. HTML Table View
2. Tree View
3. Grouping Items

Transformation de documents XML complexes

1. Transformation Styles
2. Typeswitch Transformations
3. Transformation idioms
4. Generating Skeleton Typeswitch Transformation Modules
5. Web XML Viewer

Rapports paginés

1. Installing_the_XSL-FO_module
2. Generating PDF from XSL-FO files
3. XSL-FO Tables
4. Converting HTML tables to XSL-FO tables
5. XSL-FO Images

Publication de contenu

1. Publishing Overview
2. Publishing to Subversion

Comparaison et fusion de XML

1. Compare two XML files
2. XML Differences
3. Compare with XQuery
4. Time Comparison with XQuery
5. Synchronizing Remote Collections
6. Finding Duplicate Documents

Requêtes basées sur le temps

1. Time Based Queries
2. Timing a Query

Documents

1. TEI Concordance
2. TEI Document Timeline
3. DocBook to HTML
4. DocBook to PDF
5. DocBook to ePub
6. DocBook to Microsoft Word
7. OpenOffice to HTML
8. Office Open XML

XML Schemas

1. XML Schema to Instance
2. XML Schema to XForms
3. XML Schema to SVG
4. Caractères spéciaux

Liens avec d'autres langages

1. Using intermediate MusicXML documents
2. MusicXML to Arduino
3. XQuery et Python
4. XQuery SQL Module
5. XQuery depuis SQL
6. List OWL Classes
7. Excel et XML
8. AJAX Navigating Collections
9. AJAX Employee Search
10. AJAX Incremental Search of the Chemical Elements
11. AJAX DOJO data
12. XML vers SQL
13. Simple RSS reader
14. XHTML et Voice
15. XQuery et XSLT
16. Basic Authentication
17. Digest Authentication
18. OAuth
19. Wikipedia Page scraping
20. Wikipedia Lookup
21. Wikipedia Events RSS
22. Wiki weapons page
23. Wikibook index page
24. Wikibook list of code links
25. Freebase

26. Google Docs

Virtualisation

1. Graph Visualization
2. Graphing from RDF
3. Dataflow diagrams
4. Sequence Diagrams
5. Example Sequencer

Google

1. Google Charts
2. Google Chart Sparkline
3. Google Chart Bullet Bar
4. Histogram of File Sizes

Page Scraping

1. Overview of Page Scraping Techniques
2. Page scraping and Yahoo Weather
3. UK shipping forecast
4. BBC Weather Forecast
5. Page scraping and Mashup
6. Simple RSS reader
7. Multiple page scraping and Voting behaviour
8. Link gathering
9. REST interface definition
10. Caching and indexes

Mapping

1. Google Geocoding
2. String Analysis#Location_Mapping
3. Flickr GoogleEarth
4. Nationalgrid and Google Maps
5. SMS tracker

Timelines

1. Creating a Timeline
2. Timelines of Resource
3. TEI Document Timeline

Web sémantique

1. DBpedia with SPARQL - Football teams
2. DBpedia with SPARQL and Simile Timeline - Album Chronology
3. DBpedia with SPARQL - Stadium locations

4. The Emp-Dept case study
    1. XML to RDF
    2. SPARQL Tutorial
    3. SPARQL interface
5. Graphing Triples
6. SPARQLing Country Calling Codes
7. Southampton Pubs
8. Alphabet Poster
9. Simile Exhibit
10. Latent Semantic Indexing

Outils de développent

1. Sitemap for Content Management System
2. Uptime monitor
3. XQuery IDE
4. Image Library
5. XML Schema to Instance
6. Lorum Ipsum text
7. XQuery and XML Schema
8. Generating XQDocs
9. Call Graphs
10. System Properties
11. Environment Variables

Validation

1. Validating a document
2. Validation using a Catalog
3. Validating a hierarchy
4. Validation with Schematron

Path Analysis

1. All Paths
2. All Leaf Paths

Securité

1. Login and Logout
2. URL Driven Authorization
3. Digital Signatures
4. Changing Permissions on Collections and Resources

Unit Testing Tools

1. XUnit Testing
2. XUnit Annotations

Études de cas

1. Fizzbuzz
2. Project Euler
3. Topological Sort
4. Slideshow
5. Sudoku
6. Pachube feed
7. World Temperature records
8. UWE StudentsOnline

Modules eXistdb

1. Installing the XSL-FO module
2. Setting HTTP Headers
3. Database Utilities
4. Get zipped XML file
5. Unzipping an Office Open XML docx file
6. Installing eXgit
7. File Transfer Client
8. FTP Client
9. Digest Authentication
10. UK shipping forecast
11. Convert XML to JSON
12. Lucene_Search
13. Sending E-mail
14. Basic Feedback Form
15. Using the Math Module
16. Using the Memcached Module
17. Execute External Process
18. Getting URL Parameters
19. Getting POST Data
20. Checking for Required Parameters
21. Manipulating URIs
22. Parsing Query Strings
23. Adder
24. XQuery Batch Jobs
25. Sequences Module
26. Basic Session Management
27. Subversion
28. String_Analysis
29. Registered Modules
30. Registered Functions
31. Dynamic Module Loading
32. Higher Order Functions
33. Timing Fibonacci algorithms
34. XMP data
35. Basic Authentication
36. Validating a document
37. Saving_and_Updating_Data
38. Splitting_Files
39. Generating xqDoc-based XQuery Documentation

40. Installing the XSL-FO module
41. Generating PDF from XSL-FO files
42. XSL-FO Tables
43. XSL-FO Images
44. XSL-FO SVG
45. Using Triggers to Log Events
46. Using Triggers to assign identifiers
47. Sending E-mail
48. Inserting and Updating Attributes
49. Updates and Namespaces
50. URL Rewriting Basics
51. Reindex a Collection
52. eXist Replication
53. eXist Crib sheet

Annexes

1. DataDirect XQuery
2. Microsoft SQL Server 2005
3. NetKernel
4. Oracle Berkeley DB XML
5. Stylus Studio
6. XQilla
7. Gotchas
8. Ah-has

Références

Tout ou partie de cette leçon est issu de la traduction d'une page sous licence GFDL « anglais *Programmation XML* » .

Consultez l'historique de la page originale (https://en.wikibooks.org /wiki/Programmation_XML?action=history) [archive] pour connaître la liste de ses auteurs.

- FunctX XQuery Function Library (http://www.xqueryfunctions.com/xq/) [archive] by Priscilla Walmsley

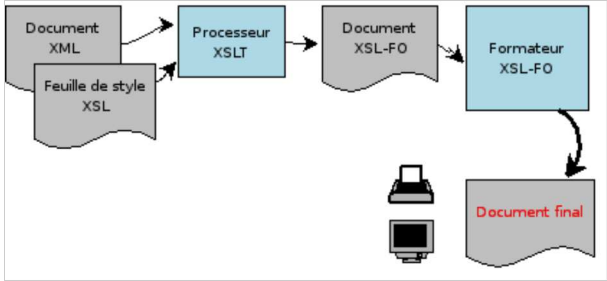
WML

XSL-FO

Introduction à XSL-FO

XSL-FO (eXtensible Stylesheet Language - Formatting Objects) dans la recommandation XSL du W3C est le vocabulaire qui décrit les mises en forme de documents XML quel que soit le support : écran, papier mais aussi dispositifs mobiles ou audio...

XSL-FO est un langage d'une haute technicité qui s'adresse principalement aux typographes afin de fournir avec les outils de gestion de documents, un outil typographique du niveau attendu par les publications imprimées.



Processus de création d'un document de présentation.

L'objectif de XSL-FO est de créer un arbre d'aires où une aire est une zone d'affichage (visuelle ou auditive).

Les aires sont de deux types :

- Les aires de blocs s'empilent les unes sur les autres,
- Les aires en-ligne s'empilent les unes à côté des autres.

XSL-FO fournit l'ensemble des commandes de contrôle de chaque aire : présentation du contenu, direction de l'empilement (écriture de gauche à droite ou inversement, de haut en bas...).

Ces commandes sont dans des balises <fo: et on les enregistre dans des documents .xfo, .fo ou .fob.

Un document "Bonjour le monde"

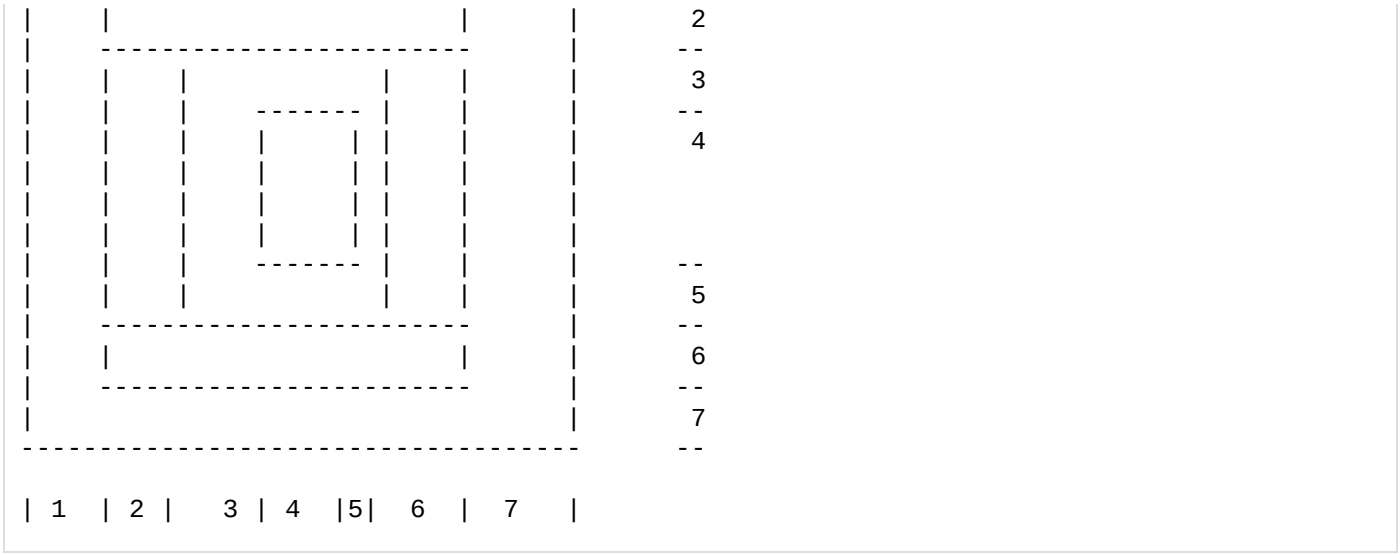
Le résultat du code suivant est *Hello world*[1] :

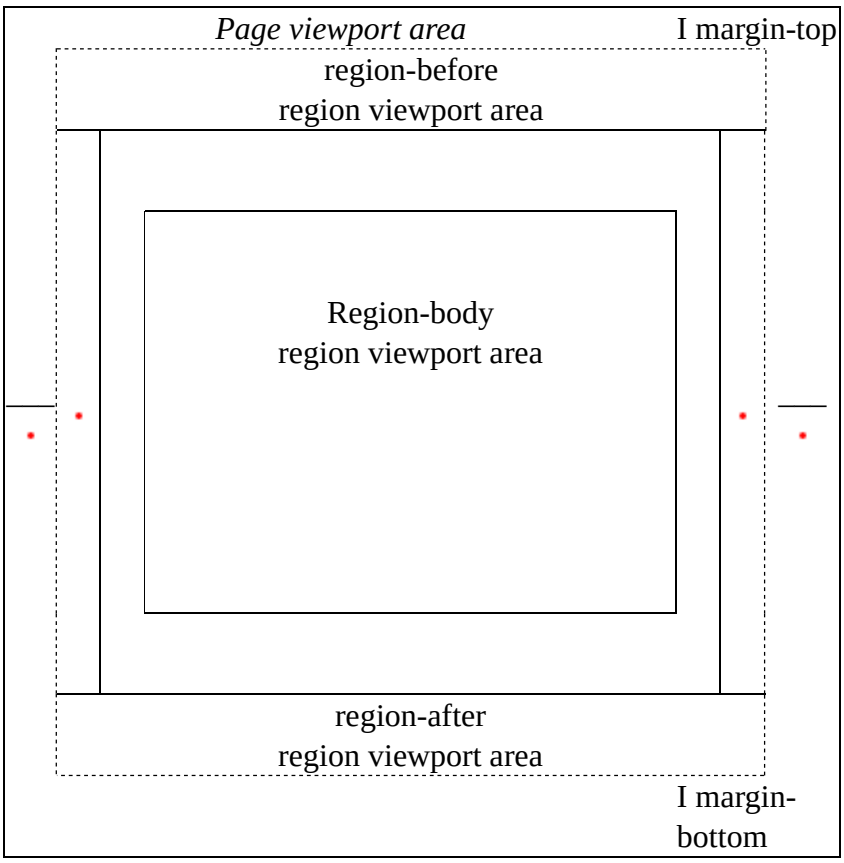
```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

 <fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <fo:region-body />
  </fo:simple-page-master>
 </fo:layout-master-set>

 <fo:page-sequence master-reference="A4">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello world</fo:block>
  </fo:flow>
 </fo:page-sequence>

</fo:root>
```

Le modèle des aires

Modèle d'une page simple

```
Format du tableau :

 ------------------------------------      --
 |                                  |       1
 |    ------------------------      |      --
```





L'affinage et la résolution des propriétés

Cette section est vide, pas assez détaillée ou incomplète.

Les objets de mise en forme

Cette section est vide, pas assez détaillée ou incomplète.

Les propriétés de mise en forme

Cette section est vide, pas assez détaillée ou incomplète.

Références

1. http://www.w3schools.com/xslfo/xslfo_output.asp

Liens externes

La dernière vérification concernant les liens externes a été effectuée le le 25 janvier 2006.

- (anglais) w3.org (http://www.w3.org/TR/xsl/) [archive] Recommandation du W3C.
- (français) yoyodesign.org (http://www.yoyodesign.org/doc/w3c/xsl1/Overview.html) [archive] Traduction de la recommandation du W3C version 1.0 du 15 octobre 2001.
- (anglais) w3schools.com (http://www.w3schools.com/xslfo/) [archive] Apprendre XSL-FO avec w3schools (tutorial, documentation...)
- (anglais) xml.apache.org (http://xml.apache.org/fop/index.html) [archive] FOP, le moteur XSL-FO d'Apache

  FOP est une application Java qui accepte un document FO et génère une page dans différents formats : PDF, PCL, PS, SVG, XML, Print, AWT, MIF and TXT. Le principal format de sortie étant le PDF.

SOAP

XML Signature

SMIL

SVG

MusicXML

SyncML

Objectifs

Après avoir lu ce chapitre, vous serez capable de :

- Comprendre les bases de SyncML et la syntaxe générale ;
- Comprendre comment et pourquoi SyncML est mis en œuvre ;
- Trouver rapidement et utiliser les spécifications techniques de SyncML.

Introduction

Les terminaux mobiles comme les PDA, les pagers, les téléphones mobiles et les ordinateurs portables ne sont par nature pas connectés en permanence à un réseau. Cependant, ces appareils contiennent des applications qui ont besoin d'informations provenant d'un réseau afin d'être utilisables. La plupart des PDA et mobiles possèdent des applications comme des calendriers, des listes de taches, des répertoires pour stocker des informations qui deviennent moins utiles à partir du moment où elles sont statiques, uniquement disponibles sur l'appareil. Par exemple, des copies statiques d'informations seront différentes de l'original dès qu'une modification est apportée d'un côté ou de l'autre. La synchronisation offre la possibilité à un terminal de se connecter à un réseau afin de mettre à jour à la fois l'information de l'appareil et l'information du réseau pour que les deux soient identiques et à jour.

Devant la prolifération d'appareils mobiles et de protocoles propriétaires ainsi que la demande croissante d'accès à de l'information en situation de mobilité, les sociétés leader sur le domaine ont compris l'intérêt de créer un langage standard et universel décrivant les actions de synchronisation entre les terminaux et les applications. Elles ont formé un consortium pour sponsoriser cette initiative et pour créer ce langage.

Actuellement, le consortium SyncML a été adopté et incorporé à l'Open Mobile Alliance (http://www.openmobilealliance.org) [archive], un regroupement de plus de 300 sociétés qui supporte plusieurs projets collaboratifs portant sur les technologies et les protocoles.

Qu'est-ce que SyncML ?

SyncML ou *Synchronization Markup Language* est le protocole standard basé sur XML de synchronisation de données au travers d'une multitude de réseaux, de plates-formes et de terminaux. SyncML a été démarré en tant qu'initiative en 2000 par de grandes sociétés comme Ericsson, IBM, Palm Inc., Lotus, Matsushita Ltd. (Panasonic), Motorola, Nokia, Openwave, Starfish Software, Psion et Symbian. Leur but était la création d'un langage universel à partir de la myriade de protocoles de synchronisation propriétaires utilisés par les appareils mobiles et de fournir un ensemble complet de fonctionnalités de synchronisation pour les futurs terminaux. Le consortium a sorti la version 1.0 en décembre 2000. Ils ont développé des nouvelles fonctionnalités et résolu les problèmes découverts avec cette version, finalisant le protocole avec la version 1.1 en février 2002.

Le protocole SyncML a été conçu en gardant ces objectifs à l'esprit :

- Garder en cohérence deux ensembles de données
- Être indépendant du transport
- Être indépendant des données synchronisées (PIM, email, fichier, ...)

SyncML comprend des commandes client et serveur définies par des DTD...

Principes de SyncML

Vocabulaire

Commençons avec un peu de vocabulaire :

- Client - le terminal mobile, son application et sa base de données locale.
- Serveur - un système distant communiquant avec la base de données du système ou de l'application.
- Modifications - les données dans les champs d'une base de données qui sont modifiées.
- Synchronisation - le client et le serveur échangent des messages SyncML avec des commandes.
- Package - Balises XML conformes à la DTD de SyncML décrivant les requêtes ou actions qui doivent être

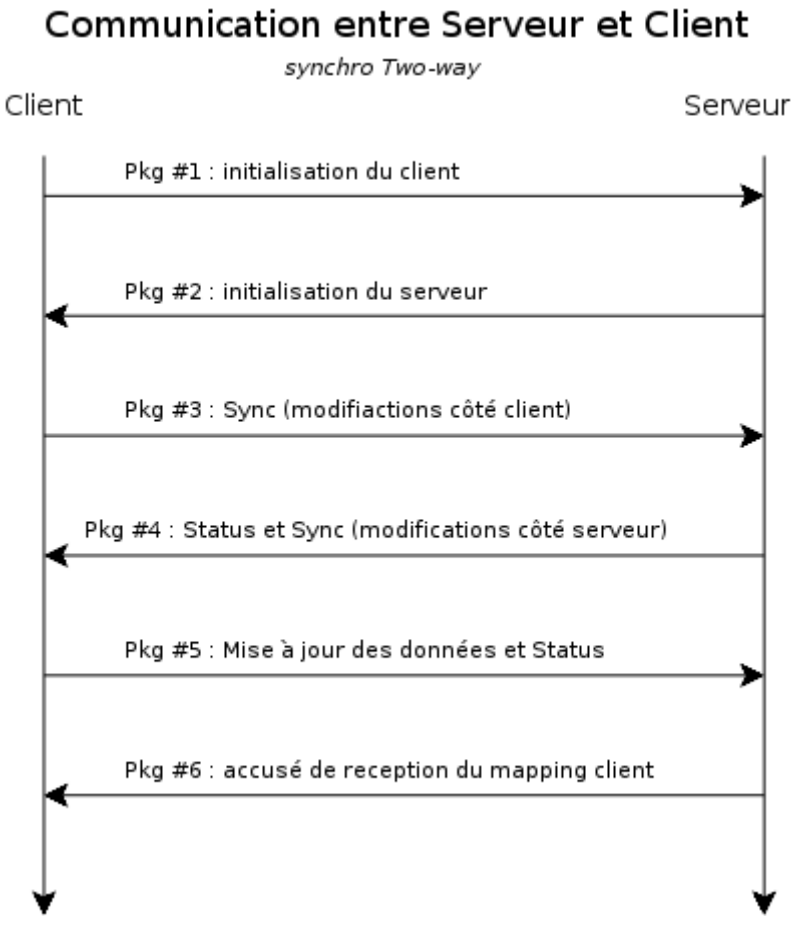
effectuées par un client ou un serveur SyncML. Un package est un ensemble d'actions à effectuer.
- Message - La plus petite unité de balise SyncML. Les grands packages sont découpés en messages séparés.
- Mapping - Utilisation d'un identifiant intermédiaire pour lier deux informations. Exemple : Disons que 'vert' c'est '5', et '5' c'est bien. Qu'est-ce qui est bien ? Si vous répondez 'vert', vous êtes tombé juste. vous avez réalisé un mapping !

Abréviations :

IMEI	International Mobile Equipment Identifier - numéro d'identification des terminaux mobiles
GUID	Global Unique Identifier - identifiant global unique
LUID	Local Unique Identifier - identifiant local unique

Messages et Packages

Un message est un ensemble de commandes SyncML transmises (en une seule fois) vers le client ou le serveur. La taille maximale d'un message est définie par la Meta données MaxMessageSize. Si un message à transmettre dépasse cette taille on le découpe en plusieurs messages. On parle alors de Multiple Message in Package. Un package correspond à un ensemble de messages pour effectuer une des étapes de la synchronisation. Les packages sont les suivants: pkg1 = initialisation du client (authentification, échange des devinf, des informations sur les bases à synchroniser), pkg2 = initialisation du serveur, pkg3 = modification côté client, pkg4 = modification côté serveur, pkg5 = mise à jour des données et statuts, pkg6 = mapping des id client et serveur.



Structure d'un message SyncML

Comme SOAP, il y a deux parties dans un message SyncML, un en-tête <SyncHdr> et un corps

<SyncBody>. L'en-tête contient des meta-informations sur la requête comme la base de données cible <Target> et la base de données source <Source>, les informations d'authentification <Cred>, l'identifiant de session <SessionID>, l'identifiant du message <MsgID>, et la déclaration de la version de SyncML <VerDTD>. Le corps contient les commandes SyncML (les statuts des commandes du message précédent, et toutes les autres commandes prévues par SyncML).

Adressage

L'adressage est réalisé au travers des balises <Source> et <DocURI>. Un serveur aura une URI du genre http://www.chris.syncml.org/sync et le terminal mobile client aura un numéro d'identification IMEA comme ceci 30400495959596904.

Mapping ou Correspondance

SyncML est basé sur l'idée que les clients et les serveurs peuvent avoir leur propre méthode pour faire correspondre les informations dans leur base de données. Aussi, les clients et les serveurs doivent avoir leur propre ensemble d'identifiants uniques.

En effet, par gain de place, certain terminaux mobile ne peuvent accepter des id trop longs, ils vont alors définir leur propres id, et envoyer au serveur le mapping à effectuer à l'aide de la balise <Map>. De cette manière, le mobile ne stocke que l'id qu'il a choisi (généralement assez court) et le serveur, lui, stocke les deux, ce qui lui permet de s'adresser au mobile avec l'id que le mobile connait. Le serveur conserve l'ensemble des id indéfiniment.

Dans les futurs échanges, le mobile utilisera seulement l'id qu'il connait, et le serveur se chargera d'effectuer les mappings correspondants.

- Les identifiants locaux uniques (LUID - Locally Unique Identifiers) sont des nombres assignés par le client à une donnée dans la base de données locale (comme un champ ou une ligne). Ce sont des nombres non réutilisables assignés à ces objets par le client SyncML.
- Les identifiants globaux uniques (GUID - Globally Unique Identifiers) sont des nombres assignés à une donnée utilisés dans une base de données distante. Cet identifiant est assigné par le serveur.

Le serveur va créer une table de correspondance pour lier les LUID et GUID ensemble.

Données côté client

LUID	Data
5	Green

Données côté serveur

GUID	Data
5050505	Green

Correspondance sur le serveur

GUID ---- 5050505	LUID ---- 5

Journaux de modification *(change logs)*

Les change logs sont une manière pour un device (client ou serveur) de déterminer la liste des modifications dans la base depuis la dernière synchro.

Les ancres

Les ancres servent à savoir si la dernière synchro s'est bien passée. Au début de session, le client envoie ses ancres (last et next). Le serveur stock la next du client. À la fin de la session (s'il n'y a pas eu d'interruption), le client met à jour ses ancres (last = next et il incrémente next). Lors de la prochaine session, le client envoie son next et last. Le serveur vérifie que le last du client vaut le next qu'il a stocké précédemment. Si c'est le cas, c'est OK, on continue. Sinon, cela ne va pas du tout et le serveur PEUT forcer une slow sync.

Session 1 : C'est une slow sync : le client envoie juste son next. La synchro se passe nickel

```
Client                      Serveur
 |---------next=1------------>|        next client = 1
 |
 |
 |
 |
```

Fin de synchro : last=1, next=2

Session 2 : Une interruption se produit lors de cette synchro

```
Client                          Serveur
 |--------last=1, next=2------>|          last = next stocké (1). next client devient 2
 |                              |
 |
Interruption
```

Comme la session va pas au bout, le client update pas ses ancres

Session 3 Après l'interruption on essaye de repartir

```
Client                       Serveur
 |------last=1, next=2------>|          CA MATCH PAS : 1!= 2 La dernière synchro s'est ma
 |
```

En comparant les ancres envoyées et celles stockées avec le type de synchronisation demandé, le serveur peut déterminer quel est l'information la plus récente. Par exemple, il est possible de réécrire une information 'nouvelle'- c'est l'information pour laquelle l'empreinte temporelle est la plus récente dans les logs- avec une information plus ancienne. Cela peut être fait en choisissant une synchronisation dans laquelle le client dit au serveur d'effacer ses informations avec les données client. Cette opération est appelée 'refresh sync from client'. Les différents types de synchronisation sont décris ci-dessous.

Synchronisations (Syncs)

Dans sa version 1.1, le langage SyncML définit 7 types de synchronisation. La section ci-dessous définit ces différents types :

1. **Two-way Sync (Synchronisation bi-directionnelle)** - Le client et le serveur s'échangent des informations relatives aux données modifiées. Le client transmet les modifications en premier.
2. **Slow sync (Synchronisation lente)** - Le client renvoie **l'intégralité de ses données**. Le serveur calcule le delta (avec les siennes) et le renvoie au client. Ce type de synchronisation est généralement utilisé lors d'une première synchro, lors d'une interruption, ou lorsque l'une des deux parties le demande.
3. **One-way sync, client only (Synchronisation uni-directionnelle, Client uniquement)** - Le client transmet ses modifications. Le serveur les accepte puis met à jour les données sans transmettre en retour ses modifications.
4. **Refresh sync from client (Synchronisation de mise à jour avec les donnés du client)** - Le client transmet sa base de données entièrement au serveur. Le serveur remplace la base de données cible par celle transmise par le client.
5. **One-way sync, server only (Synchronisation uni-directionnelle, Serveur uniquement)** - Le serveur transmet ses modifications au client. celui ci les accepte puis met à jour ses données locales sans transmettre en retour ses modifications.
6. **Refresh sync from server (Synchronisation de mise à jour avec les donnée du serveur)** - Le serveur transmet l'intégralité des informations de sa base de données. La base de données du client est entièrement remplacée.
7. **Server alerted sync** - Le serveur télé-commande au client d'initier un des modes de synchronisation présentés ci-dessus. De cette façon, le client est contrôlé à distance par le serveur.

Initialisation de la synchronisation

L'initialisation de la synchronisation est un passage obligatoire pour le client et le serveur avant d'entamer une synchronisation. La première étape est que le client et serveur parlent le même langage, en s'échangeant l'un et l'autre leur capacité (définies par le matériel, comme la quantité de mémoire, et le protocole définit par la DTD). La seconde étape est l'identification des bases de données à synchroniser. Ensuite, les deux doivent décider du type de synchronisation. La troisième et dernière partie est l'authentification. Une fois que cette étape à été complétée avec succès, la synchronisation à proprement parler peut commencer.

Authentification

Le serveur SyncML peut envoyer au client un message contenant la balise <Chal> qui représente une demande d'authentification (**Chal**lenge en anglais) pour les informations auxquelles le client tente d'accéder. Le client doit alors répondre et donner le login et mot de passe dans une balise <Cred> (**Cred**ential en anglais).

SyncML peut utiliser l'accès authentifié par le hachage md5. Le client et le serveur échangent leurs identifiants durant la phase d'authentification, retournant un code d'erreur si le processus s'arrête quelque part. La balise <Cred> est utilisée dans le <SyncHdr> pour fixer le type d'authentification qui sera utilisé dans la phase d'authentification. (Il y a le hashage md5, mais aussi l'encodage base64 et d'autres... Il faut donc que le serveur informe le client du type d'authentification utilisée).

Common SyncML implementations

Nokia a été la première entreprise à commercialiser un téléphone mobile pouvant utiliser SyncML pour la

synchronisation de la base de données du calendrier du téléphone. SyncML permet de synchroniser des listes de choses à faire, des calendriers, des carnets d'adresses, des carnets de numéros de téléphone, bien plus que ce qu'un organiseur peut faire. SyncML est capable de faire beaucoup plus de choses. En fait, il serait même approprié d'utiliser SyncML à chaque fois que deux application distantes différentes partagent les même données.

Syntaxe SyncML

Exemple SyncML

Exemple de SyncML abrégé

```
 1 <SyncML>
 2 <SyncHdr>
 3 <VerDTD>1.1</VerDTD>
 4 <VerProto>SyncML/1.1</VerProto>
 5 <SessionID>104050403</SessionID>
 6 <MsgID>5</MsgID>
 7 <Cred>...</Cred>
 8 </SyncHdr>
 9 <SyncBody>
10 <Status>...</Status>
11 <Sync>
12 <Target>target database URI</Target>
13 <Source>source database URI</Source>
14 <Add>datafield and data</Add>
15 <Replace>an existing data field with some data</Replace>
16 </Sync>
17 </SyncBody>
18 </SyncML>
```

Notez que la ligne {1} et {18} débutent le fichier SyncML par la balise racine. Ensuite, le SyncHdr est définit par les ligne {2} et {8}. Puis les ligne {3,4} qui définissent des informations concernant la version de SyncML utilisée, la ligne {5} définit l'identifiant de session (sessionID) qui permet d'identifier de façon unique le dialogue qui est en cours entre le client et le serveur, la ligne {6} représente l'identifiant du message (MsgID) qui permet d'identifier de façon unique cet ensemble de requêtes (toutes ces balises) qui vont être exécutées par l'application réceptrice. À la ligne {7}, on trouve la balise Cred (demande d'authentification, non détaillée ici) qui fait également partie de l'entête. La ligne {8} est la fermeture du SyncHdr (entête).

Le SyncBody (Corps du message) commence à la ligne {9}. Dans cette partie du message SyncML, on trouve : le status de l'application/l'appareil {10}, la source et cible de la requête (source/target) {12,13}, et les actions demandées comme la synchronisation elle-même entre les ligne {11,16}. Aux lignes {14,15}, on peut voir les commandes Add et Replace qui commandent respectivement l'ajout et le remplacement de données dans la base de donnée cible.

WBXML et SyncML

WAP Binary XML (WBXML) est une forme de XML dans laquelle les tags sont abrégés dans le but de raccourcir les balises pour la transmission vers des périphériques mobiles. En effet, ces périphériques ont en général une bande passante et une mémoire limitées. Les tags XML sont encodés en binaire pour

économiser de la place. Regardons l'exemple suivant, cela aura plus de sens.

Ce qui suit est du code binaire WBXML représentant un message SyncML. Notez que sur la première ligne il y a le définition du type de document, représentée ici en hexadécimal. Regardez ce qui est arrivé à la chaine suivante "//SYNCML//DTD SYNCML 1.1//EN"

Immédiatement après cette chaîne, on trouve les caractères '6D 6C 71'. Chacun d'entre eux représente un tag SyncML

Abréviations wbxml

```
            6D  = "<SyncML>"
            6C  = "<SyncHdr>"
            71  = "<VerDTD>"
```

Abréviations wbxml (cont.)

```
            C3  = représente le début des données opaques
            03  = ceci représente la longueur de ces données opaques
  "1" "." "1"   = le caractère "1" suivit d'un "." et de "1"
            01  = représente "</VerDTD>"
```

le code snippet WBXML *6D6C71C303"1.1"01* représente :

Extrait d'une entête SyncML header snippet

```
1 <SyncML>
2 <SyncHdr>
3 <VerDTD>1.1</VerDTD>
```

Donc on peut voir que la syntaxe WBXML este plus compacte que XML, économisant du réseau pour les appareils mobiles.

Pour plus d'information, voir les articles de Ed Dumbill's sur syncML avec WBXML :

- Have Data, Will Travel (http://www-106.ibm.com/developerworks/xml/library/x-synchml /index.html) [archive]
- WBXML and SyncML (http://www-106.ibm.com/developerworks/xml/library/x-syncml2.html) [archive]

Spécifications de SyncML

La meilleure source d'informations sur SyncML c'est le protocole lui-même. Allez voir le site de l'Open Mobile Alliance pour obtenir les spécifications de SyncML.

Open Mobile Alliance

Téléchargez les Spécifications et White Papers SyncML de l'OMA (http://www.openmobilealliance.org

/syncml/technology.html) [archive] sur le site de l'Open Mobile Alliance (http://www.openmobilealliance.org) [archive]. Ou regardez les articles sur SyncML (http://www.openmobilealliance.org/syncml/downloads.html) [archive] sur le site de l'Open Mobile Alliance (http://www.openmobilealliance.org) [archive].

Mises en œuvre de SyncML

Bien que les spécifications de SyncML soient utiles, vous devez toujours intégrer le protocole dans votre application. Il existe quelques boîtes à outils et transpositions que vous pouvez utiliser pour un démarrage rapide.

Boîte à outils SyncML de référence

l'Open Mobile Alliance a publié une boîte à outils en C pour implémenter SyncML. Vous pouvez l'obtenir ici (http://sourceforge.net/projects/syncml-ctoolkit/) [archive]. Si vous comprenez l'allemand, vous pouvez obtenir un exemple d'application utilisant cette boîte à outils ici (http://www.misterm.de/diplom/index.html) [archive].

Funambol

Si vous êtes intéressé par le développement d'application basée sur SyncML en Java, regardez le projet open source Funambol (https://www.forge.funambol.org/) [archive]. Il propose une bibliothèque de classes Java mettant en œuvre le protocole de synchronisation de données SyncML, un framework Java pour construire des applications serveurs SyncML et un serveur SyncML indépendant.

Conclusion sur SyncML

La technologie des appareils mobiles se développe très rapidement, et la 4G permet de nouveaux appareils puissants sur le marché. Ces derniers proposent du streaming multimédia, et auront pour valeur ajoutée leurs applications personnalisées et services synchronisés en SyncML.

Exercice

> **Exercices**
>
> Visiter le site de l'Open Mobile Alliance[1], et télécharger le PDF du protocole SyncML v. 1.1 pour répondre à ces questions :
>
> 1. Qu'est-ce qu'est le WBXML et où est-il utilisé ?
> 2. Quelles sont les prévisions pour SyncML ?
> 3. Nommer une situation problématique où SyncML est la meilleure solution.
>
> ---
>
> **Solution** (cliquez)

Références

1. http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/syncml-archive

Dumbill, E.(2003, March 1). *XML Watch: WBXML and basic SyncML server requirements.* IBM.com. Retrieved April 6, 2004 from http://www-106.ibm.com/developerworks/xml/library/x-syncml2.html

Open Mobile Alliance (2002, April 2). *SyncML version 1.0, 1.1 specification, white paper, errata.* Retrieved April 6, 2004 from http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html

SyncML Initiative, Ltd.(2000, December 7). *SyncML Specification Protocol version 1.0.* The Open Mobile Alliance. Retrieved April 6, 2004 from http://www.openmobilealliance.org/tech/affiliates/syncml /syncml_represent_v10_20001207.pdf

SyncML Initiative, Ltd.(2002, February 15). *SyncML Device Information DTD version 1.1.* . Retrieved April 6, 2004 from http://www.openmobilealliance.org/tech/affiliates/syncml /syncml_devinf_v11_20020215.pdf

Site de synchronisation (démo gratuite et webmaster très sympathique) http://www.memotoo.com/

MathML

MathML est l'utilisation du formalisme XML pour décrire des formules mathématiques.

Il s'agit bien de description de contenu et non pas de mise en forme ; cela permet de faire varier le rendu selon une feuille de style, et notamment selon des préférences nationales. Par exemple, le vecteur « V » sera noté `<ci type="vector">V</ci>`, et pourra être rendu **V** pour un anglo-saxon et $\vec{V}$ pour un français [1] (http://www.w3.org/TR/2003/REC-MathML2-20031021/chapter5.html#mixing.notsheet).

Enjeu

La notation mathématique est complexe. Il existe déjà des langages permettant de représenter les mathématiques, comme LaTeX. L'enjeu ici est, outre de permettre un rendu correct à l'affichage et à l'impression, de permettre d'utiliser les données par d'autres programmes.

Ainsi, une équation écrite en MathML pourrait être reprise par un solveur qui en donnerait les solutions, ou bien par un traceur qui en dessinerait le graphe, ou encore par un navigateur vocal pour malvoyant.

Format de fichier

Le fichier contenant le code MathML doit contenir la déclaration de type de document (DTD) suivante [2] (http://www.w3.org/TR/2003/REC-MathML2-20031021/appendixa.html) :

```
<!DOCTYPE math
  PUBLIC "-//W3C//DTD MathML 2.0//EN"
  "http://www.w3.org/Math/DTD/mathml2/mathml2.dtd" >
```

pour du MathML 2.0. Si le code est inclus dans du HTML, on peut utiliser une DTD HTML étendue au MathML :

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN"
  "http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd" >
```

Le code MathML est introduit par la balise `<math>` et est conclu par la balise `</math>`. Lorsque le code MathML est à l'intérieur d'un document XML contenant d'autres types de données, il faut s'assurer de la bonne gestion de l'espace des nom en utilisant la balise d'ouverture `<math xmlns="http://www.w3.org/1998/Math/MathML">`. Pour éviter les confusions de balises, on peut imposer un préfixe à toutes les balises MathML. Par exemple, si l'on veut imposer le préfixe `m:`, on utilise (exemple de HTML) :

```
<body xmlns:m="http://www.w3.org/1998/Math/MathML">
  <m:math>
    <m:mrow>…</m:mrow>
  </m:math>
</body>
```

ou bien

```
<math xmlns:m="http://www.w3.org/1998/Math/MathML">
  <m:mrow>…</m:mrow>
</math>
```

Exemple élémentaire et balises de base

La formule

$$(a+b)^2$$

s'écrit de deux manières avec MathML :

Description de présentation

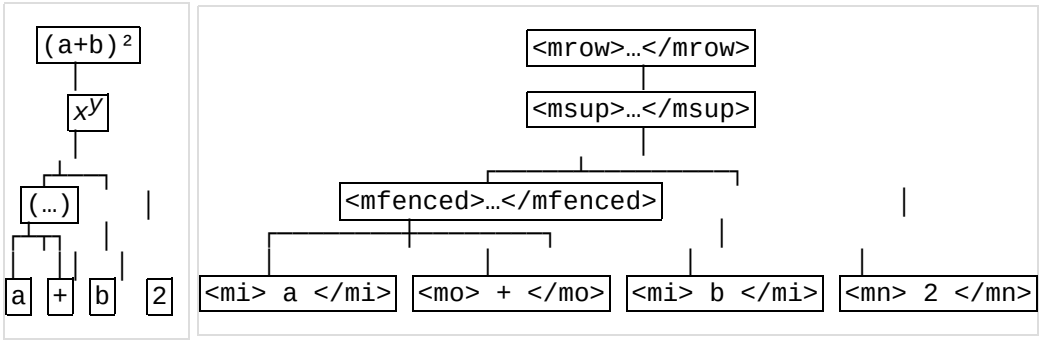
```
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi> a </mi>
        <mo> + </mo>
        <mi> b </mi>
      </mrow>
    </mfenced>
    <mn> 2 </mn>
  </msup>
</mrow>
```

On a donc :

- les identifiants (par exemple les noms de variables, qui sont des textes et symboles devant être affichés tels quels), sont encadrés par les balises `<mi>…</mi>` *(math identifier)* [3] (http://www.w3.org/TR/2003

/REC-MathML2-20031021/chapter3.html#presm.mi) ;
- les opérateurs sont encadrés par les balises `<mo>…</mo>` *(math operator)* [4] (http://www.w3.org/TR/2003/REC-MathML2-20031021/chapter3.html#presm.mo) ;
- les nombres sont encadrés par les balises `<mn>…</mn>` *(math number)* [5] (http://www.w3.org/TR/2003/REC-MathML2-20031021/chapter3.html#presm.mn) ;
- les expressions devant être considérées comme un groupe sont encadrées par `<mrow>…</mrow>` *(math row)* [6] (http://www.w3.org/TR/2003/REC-MathML2-20031021/chapter3.html#presm.mrow) ;
- `mfenced` qui fournit un encadrement, des parenthèses en l'occurrence ;
- `msup` qui accepte contient deux sous-éléments : la base (ici, « (*a* + *b*) ») et l'exposant (ici « 2 »).

On peut représenter la structure de la formule sous la forme d'un arbre :



Description de contenu

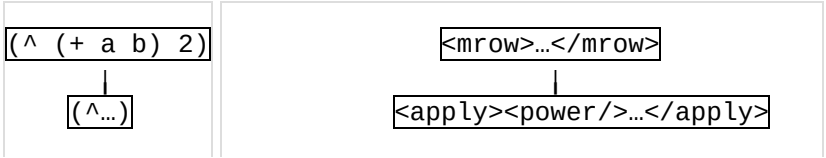
```
<mrow>
  <apply> <power/>
    <apply> <plus/>
      <ci> a </ci>
      <ci> b </ci>
    </apply>
    <cn> 2 </cn>
  </apply>
</mrow>
```

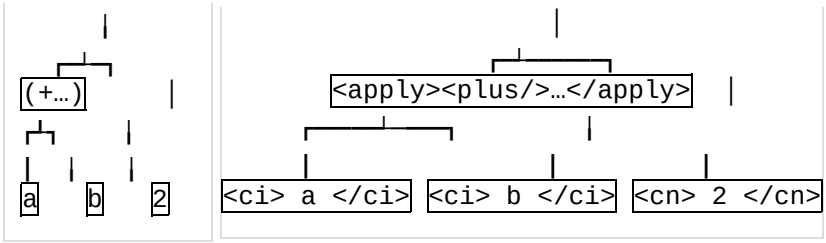
On remarque que l'on utilise ici la notation polonaise (notation préfixée), de type :

```
(^ (+ a b) 2)
```

On a :

- les balises `<apply>…</apply>` qui signifient « applique l'opérateur (le premier élément) aux éléments suivants » ;
- les balises d'opérateur, qui sont des balises sans fermeture : `<power />` pour l'élévation à la puissance et `<plus />` pour l'addition ;
- les balises qui indiquent la fonction d'éléments : `<ci>…</ci>` pour les identifiants et `<cn>…</cn>` pour les nombres.





La description de présentation est plus proche des langages orientés impression comme LaTeX ; cela permet de transposer facilement les formules. La description de contenu permet une interprétation facile de la formule en tant que telle. Si nécessaire, on peut mélanger les deux types de description, à condition d'être rigoureux :

- dans une description de présentation, un élément de contenu devrait être un fragment ayant un sens pris isolément ;
- dans une description de contenu, un élément de présentation devrait être à l'intérieur d'un élément de caractère (un « jeton ») de type variable ou nom de fonction.

Éléments

Comme en HTML, et en général en XML, ce qui est compris entre une balise d'ouverture et une balise de fermeture est appelé un « élément ». On distingue trois types d'éléments :

- les éléments de présentation, comme `mrow`, `msup`, `mi`, `mo` et `mn` ;
- les éléments de contenu, comme `partialdiff`, `leq` et `tan` ;
- les éléments d'interface.

Voir aussi

VoiceXML

X3D

XMI

XUL

XUL, pour **XML-based User interface Language**, est un langage de description d'interfaces graphiques fondé sur XML créé dans le cadre du projet Mozilla.

Exemple Simple : Hello World

Voici une comparaison entre quelques lignes de code XUL et HTML qui affichent toutes deux le traditionnel « Hello World! » :

XUL

```
<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <box>
    <description>Hello world!</description>
  </box>
</window>
```

HTML

```
<html>
  <body>
    <p>
      Hello World!
    </p>
  </body>
</html>
```

Références

- *(anglais)* Page officielle de la doc XUL de Mozilla (http://www.mozilla.org/projects/xul) [archive]
- *(français)* Le wiki xulfr.org comporte des aides sur XUL (http://xulfr.org/wiki/Presentation) [archive]
- *(anglais)* par Alice Corbin, « Xul Periodic Table (http://www.hevanet.com/acorbin/xul/top.xul) [archive] » (NB : ne fonctionne qu'avec un navigateur basé sur Gecko)

==

Récupérée de « https://fr.wikibooks.org/w/index.php?title=Programmation_XML/Version_imprimable&oldid=440121 »