Programmation Web Avancée php

XML

Dom XML en php

Document Object Model

- Voir un document XML comme un arbre
- Parcours des nœuds, détails des attributs, etc.
- Présent dans quasiment tous les langages.

- http://www.w3.org/TR/DOM-Level-2-Core/expanded-toc.html

DTD : (ancienne) façon pour définir un document XML

- DTD (*Document Type Definition*) : vérifier qu'un document XML est conforme à une syntaxe donnée (modèle).
  - une grammaire
  - document valide par rapport à une DTD
- Une DTD définie de 2 façons :
  - sous forme interne (inclure la grammaire dans le document)
  - sous forme externe (fichier local ou URL contenant la grammaire)

DTD

- **Définition d'un élément suivant la syntaxe : <! ELEMENT Nom Modèle >**
- **Modèle**
  - **ANY : L'élément peut contenir tout type de données**
  - **EMPTY : L'élément ne contient pas de données spécifiques**
  - **#PCDATA : L'élément doit contenir une chaîne de caractères**
    - **Le mot clé *#PCDATA* doit nécessairement être écrit entre parenthèses, sinon risque d'obtenir une erreur du parseur.**

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A ou l'élément B peuvent être présents	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
( )	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Exemple de DTD

- <!ELEMENT personne (nom,prenom,telephone),email? >
- <!ELEMENT nom (#PCDATA) >
- <!ELEMENT prenom (#PCDATA) >
- <!ELEMENT telephone (#PCDATA) >
- <!ELEMENT email (#PCDATA) >

- <personne>
  - <nom>Renevier-Gonin</nom>
  - <prenom>Philippe</prenom>
  - <telephone>04….</telephone>
  - <email>Philippe.Renevier@unice.fr</email>
- </personne>

DTD pour un chat

```
<!DOCTYPE chat [

<!ELEMENT chat (timestamp)*>
<!ELEMENT timestamp (number , line*)>
 <!ELEMENT line (user , hour , color? , text)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT hour (#PCDATA)>
<!ELEMENT color (#PCDATA)>
 <!ELEMENT text (#PCDATA)>
<!ATTLIST chat date CDATA #IMPLIED>
 <!ATTLIST timestamp date CDATA #IMPLIED>
 <!ATTLIST line date CDATA #IMPLIED>
]>
```

```
<chat>
<timestamp>
    <number>38</number>
    <line>
        <user><![CDATA[Phil]]></user>
        <hour>11:01</hour>
        <text><![CDATA[bonjour]]></text>
    </line>
    <line>
        <user><![CDATA[Jey]]></user>
        <hour>08:47</hour>
        <text><![CDATA[Yopp]]></text>
    </line>
</timestamp>
</chat>
```

DTD

- Attribut : <! ATTLIST Elément Attribut Type >
- Type représente le type de donnée de l'attribut, il en existe trois:
  - littéral: une chaîne de caractères, mot clé *CDATA*
  - l'énumération: une liste de valeurs possibles pour limiter le choix de l'utilisateur. Syntaxe
    - <! ATTLIST Elément Attribut (Valeur1 | Valeur2 | ...) >
    - <! ATTLIST Elément Attribut (Valeur1 | Valeur2) "valeur par défaut" > (valeur par défaut entre guillemets)
  - atomique: identifiant unique, mot clé *ID*.
- Caractère obligatoire d'un attribut (optionnel) : le faire suivre d'un mot clé particulier :
  - #IMPLIED : optionnel
  - #REQUIRED : obligatoire
  - #FIXED : valeur par défaut (à préciser entre guillemets) sinon défini.

- <! ATTLIST disque IDdisk ID #REQUIRED type (K7|MiniDisc|Vinyl|CD) "CD" >

DTD

- Entités : déclarer un groupe d'éléments sous un nom afin de ne pas avoir à réécrire ces derniers plusieurs fois dans la DTD
  - une meilleure lisibilité
  - un contrôle accru sur le contenu
  - une plus grande facilité de mise à jour
- On distingue plusieurs types d'entités dans XML :
  - les entités générales
    - <!ENTITY nom_de_l_entite "Contenu de l'entite">
    - <!ENTITY site "http://deptinfo.unice.fr/~renevier/L3">
    - usage : <site>&site;</site>
  - les entités paramètres
    - <!ENTITY % nom_de_l_entite definition>
  - les entités caractères
    - & : & < : < > : > ' : ‘ " : «
    - <!ENTITY nom_de_l_entite "&#xCODEHEXA;">
    - <!ENTITY ccedille "ç">

Génération de XML

- Une page PHP peut retourner n'importe quel type de document MIME
  - text/html par défaut
  - Mais aussi des images, du xml, n'importe quel type
- Grâce à la fonction header qui va modifier l'entête de la réponse HTTP
  - header('Content-type: text/xml');
  - Puis il faut faire générer le xml (echo de balises xml, comme pour du html)

- Autre application : les images dans les bd
  - Stockées dans des blob
  - Restituée telles quelles
  - exemple/afficheImage.ph

```
if ( isset($_GET['id']) ) {
   $id = intval ($_GET['id']);
   include ("connexion.inc");
   $req = "SELECT afficheId, image FROM affiche WHERE afficheId = $id";
   $ret = mysql_query ($req) or die (mysql_error ());
   $col = mysql_fetch_row ($ret);
   if ( !$col[0] )
                    {
                    echo "Id d'image inconnu";
                    }
   else
                    {
                    header ("Content-type: img/png");
                    echo $col[1];
                    }
}
```

Structuration DOM : Dom Document

- application de DTD
- Un Document :
  - DOMDocument
  - C'est aussi un nœud (DOMNode) => parcours selon un nœud possible
  - De nombreuses propriétés et de nombreuses méthodes
- création

```
// Create a new DOM Document to hold our document structure
$xml = new DOMDocument();
$xml->load ("fichier.xml");
```

- mixed load (string $filename [, int $options = 0])
- bool loadHTML (string $source)
- bool loadHTMLFile (string $filename)
- mixed loadXML (string $source [, int $options = 0])

Structuration DOM : Dom Document

- Sauvegarde

int save (string $filename [, int $options])
string saveHTML (void)
int saveHTMLFile (string $filename)
string saveXML ([DOMNode $node [, int $options]])

- Création de node

DOMCDATASection createCDATASection (string $data)
DOMElement createElement (string $name [, string $value])
DOMText createTextNode (string $content)

- Importation par copie

DOMNode importNode (DOMNode $importedNode [, bool $deep])

- Sélection

DOMElement getElementById (string $elementId)
DOMNodeList getElementsByTagName (string $name)

Méthodes de DomNode de manipulations des éléments enfants

DOMNode {

// pour tester si le nœud a un/des enfant(s)

bool hasChildNodes (void)

// pour ajouter un nœud / enfant enfin de listes, la valeur retournée est le nœud inséré

// un nœud ne peut être qu'à un seul endroit…

DOMNode appendChild (DOMNode $newnode)

// pour insérer un nœud parmi d'autre (devant un autre) , la valeur retournée est le nœud inséré

DOMNode insertBefore (DOMNode $newnode [, DOMNode $refnode])

// pour remplacer un nœud ($oldnode) par un autre ($newnode) , la valeur retournée est le nœud inséré

DOMNode replaceChild (DOMNode $newnode , DOMNode $oldnode)

// pour supprimer un nœud enfant, , la valeur retournée est le nœud supprimé

DOMNode removeChild (DOMNode $oldnode)

// pour dupliquer (en profondeur si $deep = true) un noeud

DOMNode cloneNode ([bool $deep])

// et encore d'autres méthodes comme bool

isSameNode (DOMNode $node)

// etc.

}

Structuration DOM: DOMNodeList

- Juste une liste de nœuds

DOMNodeList {
/* Propriétés */
readonly public int *$length* ;
/* Méthodes */
DOMNode DOMNodelist::item (int $index)
}

- exemples
  - For($i = 0; $i < $list->length; $i++) { … }
  - If ($list->item(0)->hasChildren()) { … }

Structuration DOM : DOMElement

- C'est aussi un DOMNode
  - Parcours selon un nœud
- Création : plutôt par un objet DOMDocument, sinon avec un nom de tag
- Méthodes liées aux attributs
  - string getAttribute (string $name)
  - bool hasAttribute (string $name)
  - bool removeAttribute (string $name)
  - DOMAttr setAttribute (string $name , string $value)
- sélection
  - DOMNodeList getElementsByTagName (string $name)

Structuration DOM : DOMElement

- Élément de base de l'arbre, avec parent, fratrie, enfants, etc.

// nom du tag
public readonly string *$nodeName* ;
// valeur (attention, pas toujours pertinent, dépend du type de nœud)
public string *$nodeValue* ;
// type : un entier prédéfini : http://php.net/manual/fr/dom.constants.php
// ex: 3 == XML_TEXT_NODE == du texte…
public readonly int *$nodeType* ;

// Cet attribut retourne le contenu texte de ce nœud et de ces descendants.
public string *$textContent* ;

Structuration DOM : DOMElement

- Attributs de structure

// nœud parent
public readonly DOMNode *$parentNode* ;
// listes des nœuds enfants
public readonly DOMNodeList *$childNodes* ;
// premier nœud inclus
public readonly DOMNode *$firstChild* ;
// dernier nœud inclus
public readonly DOMNode *$lastChild* ;
// nœud « frère » (ayant le même parent) précédent
public readonly DOMNode *$previousSibling* ;
// nœud « frère » (ayant le même parent) suivant
public readonly DOMNode *$nextSibling* ;

Exemple : « chat »

- http://deptinfo.unice.fr/~renevier/L3/chat/chat.inc.txt
- Une fonction pour générer la vue
  - generateViewChat
- Une fonction pour générer le formulaire
  - generateFormChat
- Une fonction pour entrer un nouveau message
  - postToChat()

- Encapsulation autre…