Web (Persistance)

Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 7

Sérialisation et persistance en XML et JSON

Plan

- Introduction
- XML
- Persistance avec XML
- JSON
- Persistance avec JSON

- Remerciements
  - Georges-André Silber (CRI/ENSMP) pour XML

Introduction

- Dans la Séance 4 nous avons étudié des solutions simples (bas niveau) pour la persistance, basées sur la lecture et écriture de fichiers
- Nous avons discuté des limitations / défauts de cette approche :
  - Manque de sémantique / métadonnées
  - Stockage *ad hoc*
  - Nécessité de « réinventer la roue » à chaque fois
- Nous allons maintenant étudier deux solutions complètement générales pour le problème de la persistance :
  - XML, un langage de balises, généralisation de HTML
  - JSON, un format de sérialisation basé sur JavaScript

XML

- XML : eXtensible Markup Langage
- Langage de balisage extensible
- Norme du W3C depuis 1998
- Version 1.0 (février 1998) ; Version 1.1 (février 2004)
- Origine et buts de XML
  - HTML 1.0, 2.0, 3.0, 4.0, 4.1
  - Volonté de « stabiliser » le langage du Web
  - Comment : en créant un meilleur HTML
  - Inspiration : SGML (Norme ISO 1986)
  - Séparation données / présentation

Qu'est-ce que XML ?

- Un langage de description d'une classe d'objets de données appelés « documents XML »
- La norme décrit partiellement le comportement de programmes les manipulant
- XML est une forme restreinte de SGML (1986)

Exemple de document XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<coordonnees>
  <adresse>
    <lignesAdresse>
      <ligne>Centre de Recherche en Informatique</ligne>
      <ligne>Ecole des mines de Paris</ligne>
      <ligne>35, rue Saint-Honoré</ligne>
    </lignesAdresse>
    <codePostal>77305</codePostal>
    <ville>FONTAINEBLEAU Cedex</ville>
  </adresse>
  <url>http://www.cri.ensmp.fr</url>
  <tel t=”fixe”>01 64 69 47 08</tel>
  <tel t=”fax”>01 64 69 48 47</tel>
</coordonnees>
```

Documents XML

- Composés d'unités de stockage appelées entités, contenant des données analysées syntaxiquement ou non.
- Ces données sont des caractères qui sont soit des données simples soit des données concourant au balisage.
- Le balisage décrit les structures logiques et de stockage du document
- Un document XML est **bien formé** s'il respecte les règles de XML
- XML fournit un mécanisme pour contraindre ces structures, les DTD (Définition de Type de Document)
- Un document XML peut être **valide** par rapport à une ou plusieurs DTD

DTD

- Définition d'un élément : <! 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.
  - Opérateurs :
    - \+ au moins une fois
    - ? optionnel
    - , concaténation
    - * zéro ou plusieurs fois
    - | alternative
    - ( ) regroupement

Exemple de DTD

```
<!ELEMENT    coordonnees (adresse, url?, tel*)>
<!ELEMENT    adresse (lignesAdresse, codePostal, ville)>
<!ELEMENT    lignesAdresse (ligne+)>
<!ELEMENT    ligne (#PCDATA)>
<!ELEMENT    codePostal (#PCDATA)>
<!ELEMENT    ville (#PCDATA)>
<!ELEMENT    url (#PCDATA)>
<!ELEMENT    tel (#PCDATA)>
<!ATTLIST    tel t (fixe|fax|mob) “fixe”>
```

Utilisation d'une DTD locale

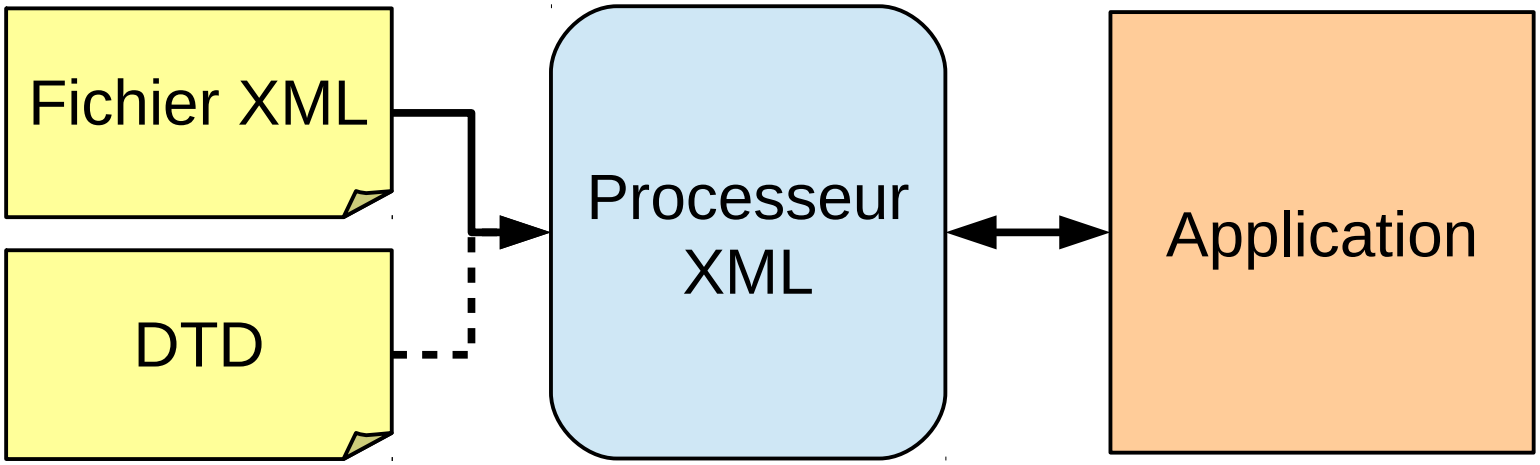
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE coordonnees SYSTEM “adresse.dtd”>
<coordonnees>
  <adresse>
    <lignesAdresse>
      <ligne>Centre de Recherche en Informatique</ligne>
      <ligne>Ecole des mines de Paris</ligne>
      <ligne>35, rue Saint-Honoré</ligne>
    </lignesAdresse>
    <codePostal>77305</codePostal>
    <ville>FONTAINEBLEAU Cedex</ville>
  </adresse>
  <url>http://www.cri.ensmp.fr</url>
  <tel t=”fixe”>01 64 69 47 08</tel>
  <tel t=”fax”>01 64 69 48 47</tel>
</coordonnees>
```

Utilisation d'une DTD distante

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE coordonnees PUBLIC "-//silber//DTD
adresse//FR" "http://www.cri.ensmp.fr/people/silber/xml/
adresse.dtd">
<coordonnees>
  <adresse>
    <lignesAdresse>
      <ligne>Centre de Recherche en Informatique</ligne>
      <ligne>Ecole des mines de Paris</ligne>
      <ligne>35, rue Saint-Honoré</ligne>
    </lignesAdresse>
    <codePostal>77305</codePostal>
    <ville>FONTAINEBLEAU Cedex</ville>
  </adresse>
  <url>http://www.cri.ensmp.fr</url>
  ...
```

Processeur XML

- Un module logiciel appelé processeur XML est utilisé pour lire les documents XML et fournir un accès à son contenu (avec validation éventuelle)
- Un processeur XML effectue le travail pour un autre module, l'application



Objectifs de conception

- Utilisation facile sur Internet
- Permettre de nombreuses applications
- Compatible avec SGML
- Nombre d'options dans XML réduit au minimum, idéalement aucune
- Documents XML lisibles par l'homme et raisonnablement clairs
- Description de XML formelle et concise
- Facilité d'écriture des programmes traitant les documents XML
- Facilité de création de documents XML
- Concision dans le balisage importe peu

Format ouvert

- Pas de droits à payer pour l'utilisation de XML
- La spécification de XML peut être distribuée librement, à condition que tout le texte et les notices juridiques demeurent intacts

Interfaces de programmation XML

- Document Object Model (DOM)
  - Voir un document XML comme un arbre
  - Parcours des nœuds, détails des attributs, etc.
  - Présent dans quasiment tous les langages.
  - Méthodes d'accès et de modification de cet arbre
  - http://www.w3.org/DOM
- Simple API for XML (SAX)
  - Lecture séquentielle, pas de création d'arbre en mémoire
  - Association d'actions (fonctions utilisateur) à la lecture des éléments XML, déclenchées au fur et à mesure de la lecture du document
  - http://www.saxproject.org

Utilisation du DOM en PHP

- Création d'un document DOM :
  - $xml = new DOMDocument();
- Méthodes de DOMDocument :
  - $xml->load ("fichier.xml");
  - int save (string $filename [, int $options])
  - DOMCDATASection createCDATASection (string $data)
  - DOMElement createElement (string $name [, string $value])
  - DOMText createTextNode (string $content)
  - DOMElement getElementById (string $elementId)
  - DOMNodeList getElementsByTagName (string $name)
  - …

Méthodes de DOMNode

- bool hasChildNodes (void)
- DOMNode appendChild (DOMNode $newnode)
- DOMNode insertBefore (DOMNode $newnode [, DOMNode $refnode])
- DOMNode replaceChild (DOMNode $newnode , DOMNode $oldnode)
- DOMNode removeChild (DOMNode $oldnode)
- DOMNode cloneNode ([bool $deep])
- isSameNode (DOMNode $node)
- …

JSON

- Format d'échange
  - Plus léger que XML
  - Basé sur du JavaScript
- Par défaut depuis Php 5.2
  - sur-ensemble de JSON
  - encoder et décoder les données de type scalaire (boolean, integer, float, string) et NULL
  - JSON standard ne les supporte que dans un tableau ou un objet
- Existence de librairies

Format JSON

- Une collection de couples nom/valeur.
  - Réification en objet ou en table de hachage ou en tableau associatif ou en structure, etc.
  - En php : en objet ou en tableau
- Une liste de valeurs ordonnées.
  - Réification en tableau, en liste, etc.
  - En PHP : tableau (séquentiel) ou objet (associatif)
- Les deux comportent des valeurs

Grammaire JSON

object ::= '{}' | '{' *members* '}'

members ::= *pair* | *pair* ',' *members*

pair ::= *string* ':' *value*

array ::= '[]' | '[' *elements* ']'

elements ::= *value* | *value* ',' *elements*

value ::= *string* | *number* | *object* | *array* | 'true' | 'false' | 'null'

Exemple de JSON

```
"coordonnees" : {
  "adresse" : {
    "lignesAdresse" : [
      "Centre de Recherche en Informatique",
      "Ecole des mines de Paris",
      "35, rue Saint-Honoré" ],
    "codePostal" : 77305,
    "ville" : "FONTAINEBLEAU Cedex" },
  "url" : "http://www.cri.ensmp.fr",
  "tel" : { "fixe" : "01 64 69 47 08",
            "fax" : "01 64 69 48 47" }
}
```

Encodage / Décodage

- mixed json_decode (string $json [, bool $assoc = false)
  - // exemple de php.net
    $json = '{"foo-bar": 12345}';
    $obj = json_decode($json);
    print $obj->{'foo-bar'}; // 12345
- string json_encode (mixed $value)
- Fonctions pour connaitre la dernière erreur liée à JSON
  - json_last_error_msg
  - json_last_error

Merci de votre attention