XML et Programmation Internet

Cours 1

kn@lri.fr

Contenu du cours

1. XML pour la représentation des données
   - Motivation, présentation d'XML sérialisé (cours 1)
   - Notions de schémas et de validation (cours 1)
2. Interroger des documents XML
   - XPath (cours 2-3)
   - XSLT (cours 3-4)
3. Programmer avec XML
   - Les modèles DOM et SAX (cours 5)
   - XML Avancé: encodage relationnel (cours 6)

Modalités de Contrôle des Connaissances (MCC)

2 sessions:

- 1ère session
  - Contrôle continu (33%): Projet (soutenances fin janvier)
  - Examen écrit (67%)
- 2ème session (examen 100%)

Organisation:

- 8 séances de cours
- 8 séances de TD (sur machine)

Plan

Qu'est-ce qu'XML ?

XML (eXtensible Markup Language) est un standard de représentation de données

- Conçu pour être lisible par un humain, mais traîtable simplement par les machines
- Permet la représentation de données structurées
- Est normalisé (par le W3C)
- Typé (notion de schéma très fine)

En a-t-on besoin ?

Quels sont les autres moyen de représenter les données ?

- Table relationnelle
- Fichiers textes non structurés
- Format binaires *ad-hoc*

Quels sont les désavantages des représentations ci-dessus ?

- Non échangeable (il faut faire un *dump* de la base), contraintes simples
- Lisible, mais sans structure donc difficilement utilisable par un programme, pas de schéma
- Lié à une utilisation particulière

Historique

- Fin des années 1980: SGML adopté pour la publication de média
- Milieu des années 1990: un groupe de travail commence à étudier l'utilisation de SGML pour le Web (qui débutait)
- 1998 : XML 1.0 devient une recommandation du W3C

Exemples d'utilisation

- Page Web (XHTML)
- Flux RSS (atom)
- Voix sur IP/Messagerie instantanée (SMIL/Jabber)
- Images vectorielles (SVG)
- Données biologiques (séquençages)
- Données financières et bancaires (XBRL)
- Document bureautiques (OpenDocument Format, Office 2010)
- Données linguistiques (TreeBank)

Plan

Représentation des textes

Avant de représenter des documents complexes, on s'intéresse aux textes (sans structure particulière)

Problématique: comment représenter du texte réaliste ?

Exemple de texte réaliste:
" مرحبا و عليكم السلام,Здравей,¡Hola!, 你好,Góðan daginn,... "

Historiquement...

Encodage 1 caractère = 1 octet (8 bits) :

- Encodage ASCII sur 7 bits (128 caractères)
- ASCII étendu 8 bits (256 caractères, dont 128 de « symboles »
- Latin 1 : ASCII 7 bits + 128 caractères « ouest-européens » (lettres accentuées française, italienne, ...)
- Latin 2 : ASCII 7 bits + 128 caractères « est-européens » (Serbe, Hongrois, Croate, Tchèque, ...)
- Latin 3 : ASCII 7 bits + 128 caractères turques, maltais, esperanto,
- Latin 4 : ASCII 7 bits + 128 caractères islandais, lituanien, ...
- ...
- Latin 15 : Latin 1 avec 4 caractères « inutiles » remplacés (par exemple pour « € » à la place de « ¤ »)

... et pendant ce temps là, ailleur dans le monde

Encodage multi-octets:

- Encodages spécifiques pour le Chinois (Big5, GB, ...)
- Encodages spécifiques pour le Japonais (Shift-JIS, EUC, ...)

Impossibilité de mettre plusieurs « alphabets » dans un même texte

Chaque logiciel « interprétait » les séquences d'octet de manière prédéfinie

UTF-8

Universal (Character Set) Transformation Format 8 bit

- Encodage à taille variable « universel » (contient tous les alphabets connus)
- Un organisme (ISO) donne un code à chaque symbole
- Compatible avec ASCII 7 bits

Encodage

Nombre d'octets	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6
1	0xxxxxxx					
2	110xxxxx	10xxxxxx				
3	1110xxxx	10xxxxxx	10xxxxxx			
4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Exemples

A $\longrightarrow 65_{10} \longrightarrow 0100\ 1010_2$ (représenté sur un seul octet)

ẽ $\longrightarrow 7877_{10} \longrightarrow 0001\ 1110\ 1100\ 0101_2$ (représenté 3 octets) :

1110 0001 10 11 10 11 10 00 01 01 ≡ 225 187 133

 $\longrightarrow 128053_{10} \longrightarrow \ldots \equiv 240\ 237\ 220\ 181$

Avantages

- compatible ASCII 7 bits (d'anciens documents texte en anglais sont toujours lisibles)
- pas d'espace gaspillé (à l'inverse d'UTF-32 ou tous les caractères font 32 bits)

Inconvénients

- Caractères à taille variable: il faut parcourir le texte pour trouver le $n^{ème}$ caractère

Plan

Qu'est-ce qu'un document XML ? (1)

- C'est un fichier texte, encodé par défaut en UTF-8
- Le texte peut être structuré au moyen de balises
- Une balise est de la forme `<foo>` (ouvrante) ou `</foo>` (fermante)
- Les balises doivent être bien parenthésées
- Il doit y avoir une racine « englobant » tout le contenu du document (texte et autres balises). On appelle cette balise la racine
- Les balises sont sensibles à la casse
- `<foo/>` est un raccourcis pour `<foo></foo>`

Exemple

```
<exemple>
   ceci est la racine <balise>on peut</balise> y mettre des
   balises <level><level>imbriquées</level> </level> comme
   on veut si
   elles sont <FOO>bien</FOO> parenthésées
</exemple>
```

Qu'est-ce qu'un document XML ? (2)

- On peut mettre des espaces et des commentaires en dehors de la racine
- Les commentaires sont délimités par <!-- et -->
- On peut annoter les balises ouvrantes avec des attributs de la forme `att="v"` (on peut aussi utiliser « ' » pour délimiter les chaînes)
- Au sein d'un même élément (ou balise) on ne peut pas avoir deux fois un attribut avec le même nom
- On dispose de séquences d'échappement: < pour ←, > pour →, ' pour ', " pour ", & pour & et &nnnn; pour un caractère UTF-8 dont le code décimal est nnnn.

Exemple

```
<-- commentaire en début de fichier-->
<exemple id="1">
     Un autre <balise id="2" type="texte">exemple</balise>
</exemple>
```


Exemple complet (trouver toutes les erreurs)

```
<exemple id="1">

  On se donne cette fois un <FOO>exemple</Foo> complet, mais

  <balise>incorrect. En effet, il y a

    <note id="32" val='32'>plusieurs</note> erreurs dans

    <note id="42" id="51">dans ce document</note>.

    Il n'est pas simple de toutes les trouver

</exemple>
<exemple>En plus cet exemple est en deux parties</exemple>
```


Utilisations d'XML (1)

- Les noms de balise sont libres. La personne qui conçoit le document choisit les balises
- Un document XML est une manière de représenter un arbre (l'élement racine est la racine de l'arbre)

Un bon exemple est XHTML (XML pour les pages Web)

- La structure des documents (titres, sections, paragraphes, tableaux) est donnée par les balises
- Le rendu graphique ne fait pas partie de la structure du document et est donné par un moyen annexe (feuille de style CSS par exemple)

Autre exemple: les flux RSS de mises à jour d'un blog

- Lien vers le blog wikimedia
- XML est un exellent format d'échange

Utilisation d'XML (2)

- Le format texte est un mauvais format pour l'interrogation
- Il encode un arbre mais ne permet pas de le manipuler directement

En réalité, on ne manipule pratiquement jamais de XML tel que stocké sur le disque

- On peut charger un document XML sous la forme d'un arbre (Objet Java par exemple) dans lequel on peut naviguer
- Certaines bases de données permettent de stocker des fichiers XML dans des colones (comme un VARCHAR)

Une application moderne mélange BD relationnelle et XML (et aussi d'autres choses si besoin: JSON, YAML, ...)

Étude de cas: journal en ligne

On se pose dans le cas du site internet d'un journal en ligne

- Le site a des utilisateurs enregistrés (ainsi qu'un accès invité)
- Le site propose des articles en lecture sur le site
- Le site permet d'exporter des articles en PDF pour impression
- Le site permet la lecture sur smartphone/tablette

Une solution possible

- On stocke les articles comme des fichiers XML avec une syntaxe particulière:

```
<article>
  <category>Science</category>
  <title>A new planet discovered</title>
  <date>2014/09/09</date>
  <authors>
    <name><first>Jonh</first><last>Doe</last></name>
    <name><first>Someone</first><last>Else</last></name>
  </authors>
  <content>
    <bold>A new plange</bold> has been discovered …
  </content>
</article>
```

- Une BD relationelle stocke les données dynamiques (utilisateurs, ...)
- Un programme (Java ou PHP) applique une transformation (XSLT) pour transformer les articles XML en pages XHTML
- Le programme utilise la BD pour personaliser l'affichage
- Le programme applique une autre transformation pour transformer les articles en PDF plutot que XHTML
- Le programme transforme l'article en un XML RSS pour affichage dans un lecteur de blog

Plan

Schéma d'un document

Comme tout le monde peut définir son propre format XML, on veut pouvoir être sûr que des données en entrées d'un programme ont un certain format (par exemple, c'est du XHTML valide, sans balise inconnue des navigateurs)

Il existe plusieurs manières de contraindre les balises d'un document XML. On s'intéresse dans le cours à la plus simple.

DTD

Document Type Definitions. Permet de définir le contenu d'un document par des expressions régulières

Syntaxe particulière qui n'est pas du XML

Permet de définir:

- les balises autorisées dans un document
- le contenu des balises au moyen d'expressions régulières
- les attributs d'un élément
- le type des valeurs que peut prendre un attribut

Syntaxe des DTD

Un fichier contenant une suite de directives de la forme suivantes:

- `<!ELEMENT nom_elem regexp_elem>` dit qu'un élement de nom `nom_elem` contient des éléments décrits par l'expression régulière `regexp_elem`
- `<!ATTLIST nom_elem nom_att type_att val_att>` signifie que l'élément (balise) `nom_elem` a un attribut nom_att, dont le type est `type_att` et la valeur est `val_att`
- Les expressions régulières sont formées de `*`, `+`,`?`, mise en séquence `(`,`)`, `EMPTY` (contenu vide), `ANY` (n'importe quel contenu), `#PCDATA` (du texte)
- Les types d'attributs sont `ID` (attribut unique dans tous le document ≡ clé primaire), `IDREF` (fait référence à un ID unique ≡ clé étrangère), `CDATA`(du texte simple), `v1|v2|…|vn` (une liste de valeurs fixées)
- Les valeurs d'attributs sont: `v` (une valeur par défaut si l'attribut est absent), `#REQUIRED` (l'attribut est obligatoire), `#IMPLIED` (l'attribut est optionnel), `#FIXED v` (valeur constante v)

Exemple de DTD

```
<!ELEMENT recette (title,ingredients,duree,etapes)>
<!ATTLIST recette difficulte (facile|normal|difficile) #REQUIRED>
<!ELEMENT title #PCDATA >
<!ELEMENT ingredients (ingredient+) >
<!ELEMENT duree #PCDATA >
<!ELEMENT etapes (e*) >
<!ELEMENT ingredient #PCDATA>
<!ELEMENT e #PCDATA>
<!ATTLIST e num CDATA>
```

Question: quel est la taille minimale d'un document valide ?

Utilisation d'une DTD (1)

Il suffit de référencer la DTD dans un élément spécial
`<!DOCTYPE racine SYSTEM "fichier.dtd" >`
avant la racine du document

```
<!DOCTYPE recette SYSTEM "recette.dtd">
<recette difficulte="facile">
<title>Tiramisú</title>
<ingredients>
  <ingredient>mascarpone</ingredient>
  <ingredient>oeufs</ingredient>
  …
</ingredients>
<duree>2h</duree>
<etapes>
  <e num="1">Séparer les blancs des jaunes</e>
  <e num="2">…</e>
  …
</etapes>
</recette>
```

Utilisation d'une DTD (2)

- Les bibliothèques permettant de charger des fichiers XML vérifient qu'il est bien formé (parenthésages, attributs, ...)
- Elles peuvent aussi valider le document par rapport à une DTD
- Si le fichier est invalide, elles lèvent une erreur
- C'est une forme de vérification de contraintes d'intégrité