Technologie Web

XML et DTD

Alexandre Pauchet

INSA Rouen - Département ASI

BO.B.RC.18, pauchet@insa-rouen.fr

Plan

Introduction (1/4)
Origines : SGML et HTML

- HTML
  - HTML = *HyperText Markup Language*
  - Modèle de représentation d'hyper-documents
  - Très utilisé par les serveurs et les clients WEB
  - Défini selon le standard SGML
- SGML
  - *Standard Generalized Markup Language*
  - Représentation de données et de documents structurés
  - Méta-langage de balisage de documents
- XML
  - *eXtensible Markup Language*
  - Standard développé par le W3C
  - Forme étendue de HTML : **définition de balises**
  - Sous ensemble restreint de SGML : **applications WEB**
  - Peut être utilisé avec les protocoles (HTTP, MIME) et les mécanismes (URL) du WEB

Introduction (2/4)
Comparaison HTML/XML

```
<p>Alexandre Pauchet</p>
<p>
   INSA de Rouen<br/>
   BP08<br/>
   Avenue de l'Université<br/>
   76800 Saint-Étienne du
         Rouvray
</p>
```

- Balises et sémantiques associées sont prédéfinies
- Mélange de structurations logique et physique
- Affichage déléguée (CSS)
- Perte du sens

```
<enseignant corps="maître de conférences">
  <prenom>Alexandre</prenom>
  <nom>Pauchet</nom>
  <adresse>
    <structure>INSA de Rouen</structure>
    <bp>BP08</bp>
    <rue>Avenue de l'Université</rue>
    <cp>76800</cp>
    <ville>Saint Étienne du Rouvray</ville>
  </adresse>
</enseignant>
```

- Extensibilité du langage
- Structuration logique
- Affichage déléguée (CSS, XSL)
- Modularité et réutilisation des structures
- Facilite l'accès à des sources de données hétérogènes

Introduction (3/4)
Structure, contenu et présentation

- 1 document = 3 aspects différents
  - Le contenu (le fond)
  - La structure logique
  - La présentation (la forme)

- XML : séparation **contenu textuel/structure logique**
  - Autres contenus = ressources externes (photos, vidéos, sons, ...)
  - Présentation décrite par moyens complémentaires (CSS, XSL, ...)
  - Présentation indépendante des contenus et de la structure

Introduction (4/4)

Exemple : Description d'un article en XML

Exemple

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE album [
  <!ELEMENT album   (titre, auteur, contenu)>
  <!ELEMENT titre   (#PCDATA)>
  <!ELEMENT auteur  (#PCDATA)>
  <!ELEMENT contenu (piste+)>
  <!ELEMENT piste   (#PCDATA)>
]>
<?xml-stylesheet type="text/css" href="styleAlbum.css"?>
<album>
  <titre>Le monde de la chatouille</titre>
  <auteur>Sacha Touille</auteur>
  <contenu>
    <piste>01 - Guiliguili</piste>
    <piste>02 - Gratouilli</piste>
  </contenu>
</album>
```

Document XML (1/10)
Structure d'un document XML

- Prologue
  - Déclaration XML
  - Déclaration de type
  - Instructions de traitement

- Arbre des éléments
  - Éléments (balises)
  - Attributs
  - Entités
  - ...

Document XML (2/10)
Prologue : déclaration XML

Déclaration XML

```
<?xml version="1.0" [encoding="encodage"] [standalone="yes|no"]?>
```

La déclaration XML indique :

- la conformité du document à une version de la norme XML,
- le jeu de caractères utilisé dans le document,
- la présence ou non de références externes.

Cette déclaration est facultative mais fortement conseillée notamment pour le problème d'encodage des caractères

Document XML (3/10)
Prologue : déclaration de type

Déclaration de type

<!DOCTYPE elt−racine Info−DTD [déclarations internes] >

La DTD (*Document Type Definition*) d'un document XML définit la grammaire d'un document XML :

- Elle facilite l'échange de fichiers (fournit une description)
- Elle facilite la validation de document (impose une grammaire)

Elle peut être :

- Interne/externe : un document ou plusieurs documents
- System/public : privée ou publiée officiellement

Document XML (4/10)

Prologue : instructions de traitement

- Instructions “facultatives”
  Leur contenu est transmis à une application pour traitement

```
<?cible arg1="val1" arg2="val2" ... ?>
```

  - `cible` : nom d'une application (`xml` est un mot réservé)
  - arguments passés à l'application cible

Exemple

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

Document XML (5/10)

Structure et éléments

- Structure d'un document XML
  - Un document XML est une structure logique arborescente
  - *Élément/noeud* = constituant logique du document
  - Éléments non prédéfinis mais choisis en fonction du type de document à représenter
  - L'ensemble de la structure est ordonnée
  - Le contenu est structuré en éléments qualifiés par des attributs valués
  - Un élément est représenté par une paire de balises (tags) et leur contenu
  - Les balises ouvrantes portent les éventuels attributs
  - L'imbrication et l'ordre des éléments reflètent la structure

Document XML (6/10)
Contraintes syntaxiques

- Un document possède une racine et une seule.
- Les éléments :
  - doivent avoir une balise ouvrante et une balise fermante,
  - peuvent être vides (éléments auto-fermants),
  - doivent être imbriquées,
  - peuvent avoir 0,1 ou plusieurs attributs.
- Un nom d'élément :
  - doit commencer par une lettre ou un "_",
  - peut comporter des chiffres, des lettres, "-", "." ou "_",
  - peut posséder un nom de domaine : "`domaineDeNoms:nomElement`",
  - est sensible à la casse.
- Les attributs d'un élément :
  - donnent des précisions sur les éléments et sur leur contenu,
  - doivent avoir un nom et une valeur,
  - ne sont pas sensibles à l'ordre.

Document XML (7/10)
Commentaires, CDATA et Entités prédéfinies

- Caractères spéciaux

  Pour le contenu de tags et les attributs, XML prédéfinit les 5 entités suivantes :

  `&` → `&` `'` → `&apos;` `>` → `>` `<` → `<` `"` → `&quot;`

- Commentaires

  `<!--ceci est un commentaire -->`

  Contraintes sur les commentaires :

  - Tout le contenu sera ignoré par l'analyseur XML
  - Pas de commentaire à l'intérieur d'autres tags

- CDATA

  Indique à l'analyseur de ne pas tenir compte du balisage :

  `<![CDATA[=>texte non regarde par l'analyseur]]>`

  Utile lorsque l'on inclut du Javascript à XML (ex : XHTML)

Document XML (8/10)

Bonne formation et validité

Document XML bien formé

Un document XML est *bien formé* s'il respecte la syntaxe XML :

- il contient 1 ou plusieurs éléments,
- un seul élément contient tous les autres (racine),
- les éléments sont correctement imbriqués,
- les balises de début et fin correspondent (casse comprise),
- les noms d'attributs sont uniques par élément,
- les valeurs d'attributs sont entre " ou ',
- les valeurs d'attributs ne référencent pas d'entités externes,
- les entités sont déclarées avant d'être utilisées.

Document XML valide

Un document XML est *valide* s'il se conforme à sa DTD
$\Rightarrow$ le document est conforme à un modèle de structure

Document XML (9/10)

Conseils pour l'écriture de documents XML

- Choisir des noms d'éléments (balises) qui représentent leur rôle (sémantique), ils doivent être aussi explicites que possible.
  - Le balisage doit être indépendant de la réalisation physique du document (ex : pas de <gras>).
  - Préférer un balisage méta-typographique : (ex : <important>, <ligne>, <cellule>).
- La position d'un élément à l'intérieur d'un autre est importante (l'ordre des balises est préservé).
- Inclure dans le document des métadonnées descriptives afin de décrire le document.
- L'indexation d'un document se fait sur le contenu des balises, pas sur les valeurs des attributs.

Document XML (10/10)
Conseils pour l'écriture de documents XML

- Utilisez un élément (balise) lorsque :
  - le contenu comporte plusieurs mots,
  - l'ordre est important (il n'y a pas d'ordre sur les attributs),
  - l'information fait partie du contenu du document par opposition à un attribut ajustant le comportement d'une balise. Si un analyseur n'est pas capable de traiter un document XML, il affichera le contenu des balises mais pas les attributs.

- Utilisez un attribut lorsque :
  - l'information modifie la balise d'un point de vue du traitement exemple : `<liste type="numérotée">... </liste>`,
  - vous souhaitez contrôler les valeurs,
  - l'information est un identifiant unique ou une référence à l'identifiant d'une autre balise.

Les DTD (1/14)
Un exemple simple de DTD

personne.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT personne (identite, activite*)>
<!ELEMENT identite (prenom, nom)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT activite (#PCDATA)>
```

Dupond.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE personne SYSTEM "./personne.dtd" >
<personne>
  <identite>
    <prenom>Eustache</prenom>
    <nom>Dupond</nom>
  </identite>
  <activite>détective</activite>
  <activite>gardien de nuit</activite>
</personne>
```

Les DTD (2/14)
DTD d'un document

DTD

- DTD = *Document Type Definition*
- Elle définit éléments et règles d'utilisation (noms des éléments, attributs possibles pour un élément, imbrications)
  ⇒**Modèle de document XML**
- Si un document n'a pas de DTD et qu'il suit les règles définies par XML, il est **bien formé**
- Si un document est bien formé et qu'il fait référence à une DTD à laquelle il est conforme, il est **valide**
- Ni la DTD, ni le document XML ne contiennent d'information concernant l'affichage, c'est la CSS qui définira la présentation

Les DTD (3/14)
Document sans DTD

- Document sans DTD
  - Le balisage est défini de manière informel
  - Il doit être bien formé pour pouvoir être affiché par un navigateur
  - Le document doit préciser dans sa déclaration XML qu'il est autonome (SDD = *Standalone Document Declaration)*

Remarque

Un document sans DTD peut être affiché mais ne peut pas être interprété

Les DTD (4/14)

A quoi sert la DTD ?

DTD = grammaire du document XML

Elle décrit

- Les éléments types :
  - noms de balises autorisées,
  - ordre et imbrication des balises,
  - caractère optionnel des éléments.
- Les attributs pour chaque élément :
  - noms des attributs autorisés,
  - caractère optionnel/obligatoire des attributs,
  - type,
  - valeur par défaut.

Les DTD (5/14)
Déclaration de DTD

3 types de déclaration

- **DTD interne**

  ```
  <!DOCTYPE racine [declarations]>
  ```

- **DTD externe privée**

  ```
  <!DOCTYPE racine SYSTEM "fichier.dtd">
  ```

  ```
  <!DOCTYPE racine SYSTEM "http://www.adresse/fichier.dtd">
  ```

- **DTD externe publique**

  ```
  <!DOCTYPE racine PUBLIC "nomConnu" "URL">
  ```

Remarque : on peut combiner une partie de DTD interne et une partie de DTD externe (privée ou publique).

Les DTD (6/14)
Prologue d'une DTD

- Prologue
  Le prologue est identique à un document XML, excepté `standalone` et `DOCTYPE` qui n'ont aucun sens dans une DTD.
  Utilité : déclarer l'encodage utilisé dans la DTD

Une DTD n'a pas nécessairement de prologue

Exemple

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

Remarque : l'encodage de la DTD n'est pas propagé aux documents XML utilisant cette DTD

Les DTD (7/14)
Contenu d'une DTD

- Déclarations dans les DTD
  - Déclarations des éléments autorisés :

    `<!ELEMENT nomBalise (contenu)>`
  - Déclaration de l'ordre des éléments dans `(contenu)`
  - Déclarations des attributs des éléments :

    `<!ATTLIST element attribut type>`

Remarque : l'ordre des déclarations n'est pas important.

Exemple

```
<!ELEMENT auteur (nom, prenom, initial)>
```

Les DTD (8/14)
Déclaration d'un élément

Déclaration d'un élément

<!ELEMENT nomElement EMPTY|ANY|(modeleContenu)>

- Un élément peut :
  - être vide,
  - contenir n'importe quel élément déclaré de la DTD,
  - être formaté selon un modèle.
- Les éléments non vides peuvent contenir :
  - uniquement des données,

    <!ELEMENT nomElement (#PCDATA)>
  - uniquement d'autres éléments,

    <!ELEMENT nomElement (element1, element2, ...)>

    <!ELEMENT nomElement (element1+, element2*, element3?)>

    <!ELEMENT nomElement (element1 | element2 |...)>
  - des données et d'autres éléments.

    ex : <!ELEMENT type(#PCDATA | article | livre)

Les DTD (9/14)
Déclaration d'attribut

Déclaration d'attribut

```
<!ATTLIST element attribut type value>
```

- Principaux types d'attributs :
  - `CDATA` : données textuelles ne contenant pas de balises XML
  - `(valeur1|...|valeurN)` : liste de valeurs possibles
  - `ENTITY` : entité déclarée dans la DTD
  - `NOTATION` : notation déclarée dans la DTD
  - `ID` : l'attribut possède une valeur unique pour chaque élément
  - `IDREF` : l'attribut se réfère à un ID d'un autre attribut
  - `IDREFS` : l'attribut peut se référer à plusieurs ID, chaque valeur étant séparée par un espace.
- `value` vaut :
  - une valeur par défaut entre guillemets simples ' ou doubles ",
  - `#REQUIRED` : valeur d'attribut obligatoire,
  - `#IMPLIED` : valeur d'attribut optionnelle,
  - `#FIXED 'val'` : définit une valeur fixée pour l'attribut.

Les DTD (10/14)
Exemple

personne.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
  <!ELEMENT arbre (personne+)>
  <!ELEMENT personne (prenom, nom, nom?)>
  <!ATTLIST personne individual_id ID #REQUIRED parent_id IDREFS #IMPLIED>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT nom (#PCDATA)>
  <!ATTLIST nom marital (oui|non) #IMPLIED>
```

arbre.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE arbre SYSTEM "./arbre-genealogique.dtd" >
<arbre>
  <personne individual_id="e10001" parent_id="e10002 e10003">
    <prenom>Bart</prenom><nom>Simpson</nom>
  </personne>
  <personne individual_id="e10002">
    <prenom>Homer</prenom><nom>Simpson</nom>
  </personne>
  <personne individual_id="e10003">
    <prenom>Marge</prenom><nom marital="oui">Simpson</nom><nom>Bouvier</nom>
  </personne>
</arbre>
```

Les DTD (11/14)
Déclaration d'entité

Déclaration d'entité

Entité = alias pour un groupe de données

- *Non analysables* : `<!ENTITY nom SYSTEM "URI" NDATA notation>`
  Elles permettent de déclarer du contenu non-XML dans un document XML (audio, vidéos, images...)
- *Générales internes* : `<!ENTITY nom "chaine">` → `&nom;`
- *Générales externes* : `<!ENTITY nom SYSTEM "URI">` → `&nom;`
- *Paramètres internes* : `<!ENTITY %nom "chaine">` → `%nom;`
- *Paramètres externes* : `<!ENTITY %nom SYSTEM "URI">` → `%nom;`

NB : les entités générales peuvent être substituées dans le corps d'un document XML, les entités paramètres dans la DTD elle-même.

Les DTD (12/14)
Composition de DTD

Inclusion d'une DTD dans une DTD

```
<!ENTITY %secondeDTD SYSTEM "secondeDTD.dtd">
...
%secondeDTD;
```

En cas de conflit de nom :

- Si plusieurs déclarations d'un même élément, la 1ère est prise en compte
- Si plusieurs déclarations d'attributs différents pour un même élément, concaténation
- Si plusieurs déclarations d'un même attribut pour un même élément, la 1ère est prise en compte

Les DTD (13/14)
Déclaration de notation

Notation

```
<!NOTATION nomNotation SYSTEM|PUBLIC "notation">
```

- Elle identifie par un nom le format des entités non analysées par le parseur XML.
- Elle définit le format des données et les applications qui permettent de les traiter.

Exemples

```
<!NOTATION GIF SYSTEM "GIF">
```

```
<!NOTATION GIF89a PUBLIC "-//Compuserve//Notation Graphics Interchange Format 89a//EN">
```

Les DTD (14/14)

Une alternative au DTD : les schémas XML

- Inconvénients des DTD
  - Syntaxe particulière
    $\Rightarrow$ apprentissage plus long, pas d'automatisation possible, ...
  - 1 seul type de données : `PCDATA`
  - Pas de définition précise du nombre d'occurrences
  - Peu de vérifications des données à l'intérieur d'un élément

- Avantages des Schémas XML sur les DTD
  - Typage des données : plusieurs type de données, création de nouveaux types, ...
  - Syntaxe XML
  - Schémas orientés objet : extension/restriction des types, ...