XML et DTD

XML, un langage d'arbres

Master Recherche SIA - Master Pro ILI

Année 2012-13

Un rapide historique

Les langages de balises : *SGML* (*Standard Generalized Markup Language*) date du début des années 70

Le web : création au CERN en 1989 par Tim Berners-Lee.

Objectif : mettre à disposition facilement des documents.
À l'origine, des travaux de recherche en physique qui intéressaient une communauté internationalement dispersée.

Nécessité d'établir des normes

Le W3C (World Wide Web Consortium) `www.w3.org` a été fondé en 1994 par Tim Berners-Lee.

Son objectif est de rédiger des recommandations pour la spécification des langages, des services, des protocoles liés au Web.

Introduction

Qu'est-ce que XML ?

- eXtensible Mark-up Language
- défini par le W3C
- permet la définition de familles de langages de balises
- fait aussi référence à une famille de technologies

```
<fiche-identite>
  <nom>Parrain</nom>
  <prenom>Anne</prenom>
</fiche-identite>
```

Introduction

A quoi sert XML ?

Représenter des données pour les manipuler, les échanger, les interroger.

Exemples

- Documents de bureautique : OpenOffice
- Documents texte : DocBook,. . .
- Données informatiques : configurations. . .
- Données échangées : XHTML, jabber, web services,. . .
- Données stockées : bases de données XML
- beaucoup d'autres choses nouvelles, chaque jour ou presque !

Exemple de document XML

```
<?xml version="1.0" encoding="iso-88-59-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
                      "http//www.w3.org/TR/xhtml11/DTD/xhtml11.dtd
<html>
 <head><title>Master Pro ILI</title></head>
 <body>
  <div class="entete">
   <div class="titre">
    <h1>Master Professionnalisé ILI</h1>
    <h1>Ingéniérie Logicielle pour l'Internet</h1>
    <h2>Master Sciences : Mention
        Mathématiques-Informatique</h2>
    <h2><a href="http://www.univ-artois.fr">Université
        d'Artois à l'UFR des Sciences (Lens)</a></h2>
   </div></div></body></html>
```

Introduction

Avantages de XML :

- Favoriser l'interopérabilité, l'échange.
- Rendre pérennes les données.
- Les rendre manipulables à la fois par les hommes et les machines :
  - transformations de documents (fusion, réorganisation, ...)
  - extraction d'informations
  - consultation, modification par programmes ad hoc

Quelles applications ? – Exemple

```
<agenda>
  <evt>
    <date>07/10/2008</date>
    <concerne>parrain@cril.univ-artois.fr</concerne>
    <heure-debut>13h</heure-debut>
    <heure-fin>16h</heure-fin>
    <objet>Licence Pro</objet> </evt>
  <regulier>
    <date-debut>08/09/2008</date-debut>
    <date-fin>15/12/2008</date-fin>
    <periode unite="semaine">12</periode>
    <concerne>parrain@cril.univ-artois.fr</concerne>
    <heure-debut>8h15</heure-debut>
    <heure-fin>12h15</heure-fin>
    <objet>Cours XML</objet></regulier></agenda>
```

Quelles applications ?

- afficher un emploi du temps hebdomadaire ;
- envoyer un mail aux personnes concernées ;
- calculer des heures d'enseignement
- ...

XML : concepts fondamentaux

Éléments fondamentaux de XML :

- éléments
- attributs
- entités

XML : concepts fondamentaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<message priorité="important">
 <destinataire>M. Dupont</destinataire>
 <expediteur>Agence Matous Heureux</expediteur>
 <objet>alimentation du chat</objet>
 <corps>
  <para>Conformément à vos instructions, je donne
   <emphase>trois</emphase> rations de croquettes
   par jour à <chat>Mistigri</chat>.
  </para>
  <para><chat>Mistigri</chat> a cependant
   pris <emphase>deux</emphase> kilos pendant
   les vacances.</para>
 </corps>
 <formulepolitesse style="simple"/>
 <signature>Melle. Dumoulin</signature>
</message>
```

XML : prologue de document

Le prologue du document contient :

- la déclaration XML
- la déclaration du type de document
- le codage des caractères utilisé

Le prologue est facultatif, mais important car il précise des informations importantes pour les processeurs XML.

XML : prologue de document

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="macss.css" type="text/css"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http//www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

XML : prologue de document

La déclaration XML

```
<?xml attribut="valeur" [attribut="valeur"] ?>
```

avec `attribut`

- `version`
- `encoding` (par défaut, c'est unicode)
- `standalone`

XML : prologue de document

Une Définition de Type de Document (DTD)

- sous-langage XML (ex : XHTML, MathML, MusicXML, DocBook ...)
- précise la grammaire que doit suivre le document.

XML : prologue de document

Une DTD peut-être :

- publique : diffusée par une institution, accessible via le web par un identifiant public ;
- spécifique à une application : accessible via une URL
- décrite directement dans le document
- un peu de tout !

XML : prologue de document

La déclaration de type de document :

```
<!DOCTYPE nom_racine PUBLIC identifiant_public [uri]
                            [sous-ensemble interne]>
```

ou

```
<!DOCTYPE nom_racine SYSTEM uri
                            [sous-ensemble interne]>
```

Exemple :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
          "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

XML : Éléments

Les éléments sont les balises qui peuvent apparaître dans un document XML. Ils peuvent être qualifiés par des attributs.

```
<nom_elt attribut="valeur" [attribut="valeur"]>
</nom_elt>
```

ou bien (élément vide)

```
<nom_elt attribut="valeur" [attribut="valeur"]/>
```

Exemple :

```
<message priorité="important">
<destinataire>M. Dupont</destinataire>
<expediteur>Agence Matous Heureux</expediteur>
...
<formulepolitesse style="simple"/>
<signature>Melle. Dumoulin</signature>
</message>
```

XML : Éléments

Restrictions syntaxiques :

- nom d'élément
  - commence par une lettre ou un _
  - contient des lettres (symboles définis par le codage utilisé), des chiffres, des tirets, des soulignés, des points
- séparateur d'éléments :
  espace, retour à la ligne, tabulation, =, ", '

XML : Éléments

Restrictions syntaxiques :

- une balise de fin arrive après une balise de début
- les balises de début et de fin apparaissent à l'intérieur du même élément parent
- entre la balise de début et celle de fin : données textuelles ou éléments
- chaque document XML a un seul élément racine

XML : Attributs

Les attributs qualifient les éléments sur lesquels ils portent.

```
<nom_elt attribut="valeur" [attribut="valeur"]>
```

Exemple

```
<exception type="msg='erreur'" />
```

Pour un même élément :

- l'ordre n'a pas d'importance
- une seule occurrence d'un même attribut

XML : Attributs

Possibilité de donner plusieurs valeurs à un même attribut :

```
<copains filles="josette ginette colette"
     garçons="alain sylvain govain">
```

Attention ! La bonne solution pourrait être :

```
<copains>
  <fille>josette</fille>
  <fille>ginette</fille>
  <fille>colette</fille>
  <garçon>alain</garçon>
  ...
</copains>
```

XML : Attributs

Attributs particuliers : id et idref pour connecter des ressources

id : une même valeur ne peut être attribuée qu'une seule fois dans le document

idref : il doit exister un élément qui possède un attribut id de même valeur

XML : Attributs

Attributs particuliers :

xml:lang : pour spécifier la langue du document

xml:space : pour spécifier la gestion des espaces (`preserve` ou `default`)

XML : Entités

Les entités sont des réserves de contenu : pour des caractères spéciaux, des éléments textuels, un morceau de balisage XML...
Les entités générales sont déclarées dans le prologue du document, elles sont ensuite appelées dans le corps du document.
Les entités générales sont appelées par `&nom_entite;` dans les données.
Les entités paramètres sont appelées par `%nom_entite;` dans les DTD ou dans le sous-ensemble interne.
Elles sont déclarées dans une DTD ou dans le sous-ensemble interne par

```
<!ENTITY nom_entite "contenu_entite">
```

XML : Entités caractères

Entités caractères prédéfinies

Nom	Valeur
amp	&
apos	'
gt	>
lt	<
quot	"

Caractères réservés

&, < et >

XML : Entités caractères

Entités caractères numériques

Les caractères non accessibles au clavier peuvent être décrits

- par leur code decimal `&#dec;`
- ou par leur code hexadecimal `&#xhex;`

Exemple :

ç peut être appelé par `&#237;` ou par `&#xe7;`

XML : Entités caractères

Entités caractères nommés

Tous les caractères accentués sont nommés dans des modules de DTD utilisés dans la DTD de XHTML 1.0.

Nom	Valeur	Nom	Valeur
agrave	à	Agrave	À
eacute	é	Eacute	É
egrave	è	Egrave	È
ecirc	ê	Ecirc	Ê
ugrave	ù	icirc	î
...	...	...	...

XML : Entités internes-externes

Le contenu d'une entité interne

- se trouve dans le document courant ;
- peut être
  - un caractère
  - une chaîne de caractères
  - un morceau de texte avec des balises XML

Une entité externe est un fragment XML qui se trouve dans un autre fichier :

```
<!ENTITY nom_entite SYSTEM "url_entite">
```

XML : Entités non parsées

Une entité non parsée : pour les entités qui contiennent autre chose que du texte.

```
<!ENTITY nom_entite SYSTEM "url_entite" NDATA id_notation>
```

Exemple

```
<!ENTITY maBinette SYSTEM "maPhoto.gif" NDATA GIF>
```

appel :

```
&maBinette;
```

XML : les sections CDATA

- CDATA : Character Data signifie . . .pas de balises !
- Permet de saisir les caractères de balisage sans précaution.
- Utile pour le code d'un programme informatique.

Exemple :

```
<balise> ici on saisit &amp;
et <![CDATA[ ici on saisit ce qu'on veut & < >!! ]]>
</balise>
```

XML : les instructions de traitement

Les instructions de traitement :

- spécifiques à un processeur XML
- syntaxe :
  `<?nom donnée?>`
- attention à ne pas en abuser

Unité d'enseignement - Définir une structure en XML

RES Administration des systèmes et des réseaux
ECTS : 6 Heures de Cours : 36
Heures de TD : 6 Heures de TP : 12
Enseignant : Helmut Ardelpic

Principe des réseaux numériques (introduction, les types de commutation, modèle OSI, couche physique, ...)

Réseaux locaux (ethernet, token-ring, ...)

Internet et les protocoles TCP/IP (historique, architecture, adressage, couche liaison d'internet, ...)

Programmation réseau (socket)

Nouveaux protocoles (ATM, Ipv6, ...)

Typer les documents

Intérêts

- avoir des documents conformes à un modèle
- pouvoir automatiser des traitements sur les documents

Comment

- avec une définition de type de document DTD
- avec un schéma XML
- avec un schéma RelaxNG

Document Type Definition

Les éléments sont définis à l'aide d'expressions rationnelles :

```
<!ELEMENT nom_elt exp_rat>
```

Opérateurs utilisés :

```
, | * +?
  ( )
#PCDATA
```

mistigri.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<message priorité="important">
<destinataire>M. Dupont</destinataire>
<expediteur>Agence Matous Heureux</expediteur>
<objet>alimentation du chat</objet>
<corps>
<para>Conformément à vos instructions, je donne
<emphase>trois</emphase> rations de croquettes
par jour à <chat>Mistigri</chat>.
</para>
<para><chat>Mistigri</chat> a cependant
pris <emphase>deux</emphase> kilos pendant
les vacances.</para>
</corps>
<formulepolitesse style="simple"/>
<signature>Melle. Dumoulin</signature>
</message>
```

La DTD de mistigri.xml

```
<!ELEMENT message (destinataire,expediteur?,objet?,
                   corps,formule_politesse,
                   (signature|sign_complete))>
<!ELEMENT destinataire (#PCDATA)>
<!ELEMENT expediteur (#PCDATA)>
<!ELEMENT objet (#PCDATA|chat|emphase)*>
<!ELEMENT sign_complete (lieu,date,signature)>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT formule_politesse EMPTY>
<!ELEMENT corps (para+)>
<!ELEMENT para (#PCDATA|chat|emphase)*>
<!ELEMENT chat (#PCDATA)>
<!ELEMENT emphase (#PCDATA)>
```

XML : vers la conception d'une DTD

Concevoir une DTD :

- importance de l'analyse ;
- choisir des noms d'éléments comme un typage
- différentes catégories d'éléments
  - ne contiennent que d'autres éléments, ou que du texte : blocs
  - ont un contenu mixte : leurs éléments fils sont des éléments en ligne
- attention à la position (ordre des éléments significatif)
- attention à la hiérarchie
- éviter les instructions de traitement

XML : vers la conception d'une DTD

Choisir entre un élément et un attribut :

- Utiliser un élément :
  - contenu relativement gros
  - ordre important
  - le contenu fait partie de l'information du document
- Utiliser un attribut :
  - le contenu modifie le traitement de l'information
  - contrôle sur les valeurs
  - le contenu est un identifiant

Définition des attributs

```
<!ATTLIST nom_elt nom_attr1 type_attr1 desc_attr1 ...>
```

- `type_attribut` : type de l'attribut ou liste des valeurs possibles ;
- `desc_attribut` : comportement de l'attribut (obligatoire, optionnel, valeur par défaut, etc . . .)

Comportement des attributs

- spécification d'une valeur par défaut

```
<!ATTLIST message
          priorite (importante|courante|basse)
          "courante">
<!ATTLIST formulepolitesse
          style (simple|appuyee|ceremoniale)
          "appuyee">
```

- attribut obligatoire

```
<!ATTLIST feuTricolore couleur
(rouge|orange|vert) #REQUIRED>
```

- attribut optionnel, sans valeur par défaut

```
<!ATTLIST div class NMTOKEN #IMPLIED>
```

- attribut à valeur fixée (cas rare !) `#FIXED valeur`

Les différents types d'attributs

- CDATA : donnée textuelle. Type le plus permissif

```
<!ATTLIST message  texte CDATA #REQUIRED>
<message texte="Salut machin, truc &amp; Cie!">
... </message>
```

- NMTOKEN : lexème nominal (commence par une lettre, suivi de lettres, de chiffres ou de .:-...)
- NMTOKENS : suite de lexèmes nominaux séparés par des espaces

```
<!ATTLIST copains filles NMTOKENS #REQUIRED>
<copains filles="Gisèle Marcelle Danièle">
... </copains>
```

Les différents types d'attributs

- ID : identifiant unique. Même syntaxe que NMTOKEN

```
<!ATTLIST garçon nom ID #REQUIRED>

<garçon nom="Paulo">...</garçon>
```

- IDREF : référence d'identifiant

```
<!ATTLIST copains meilleur IDREF #IMPLIED>
<copains meilleur="Paulo"
         filles="Colette Lucette Perette">
...</copains>
```

- IDREFS : liste de références d'identifiants, même syntaxe que NMTOKENS

Les différents types d'attributs

- `ENTITY` : accepte une valeur d'entité générale pour l'attribut

```
<!ATTLIST liste-a-puces icone ENTITY #IMPLIED>
<!ENTITY ptbleu SYSTEM "icones/pointbleu.jpg">
```

  Utilisation :

```
<liste-a-puces icone="ptbleu">
```

- `ENTITIES` : liste de noms d'entités

Les différents types d'attributs

- énumération de valeurs : liste de mots-clés

```
<!ATTLIST message
          priorite (importante|courante|basse)
          "courante">
<!ATTLIST feuTricolore couleur (rouge|orange|vert)
          #REQUIRED>
```

- NOTATION : syntaxe des NMTOKENS : indications sur le traitement de données non XML.

Valider un document XML

- Il existe plusieurs outils
- En TP, nous utiliserons `xmllint` (dans le package `libxml2`).
  `xmllint -valid mon-fichier.xml`
- Ne pas oublier d'inclure le bon doctype dans le fichier XML.

Les entités

Plusieurs types d'entités :

- les entités générales, qui permettent de faire une simple substitution de texte parsé ;
- les entités générales externes : idem, mais source externe au document. On accède à l'entité par un identifiant public ou système.
- les entités externes non parsées pour les données non-XML (NDATA et notation)

Les entités

- entité paramètre : texte de substitution simple, utilisable dans la DTD

```
<!ENTITY % para "(#PCDATA|emphase|chat)*">
```

Utilisation :

```
<!ELEMENT objet %para;>
<!ELEMENT para %para;>
```

- entité paramètre externe : déclarations d'entités, DTD ou fragment de DTD externe. Exemple les entités nommées des caractères iso-latin1.

```
<!DOCTYPE message SYSTEM "mistigri.dtd"
[<!ENTITY % isolatin1 SYSTEM "xhtml-lat1.ent">
%isolatin1; ]>
```

Les DTD modulaires

Première possibilité : en ajoutant les possibilités externes.
La règle est simple pour éviter les conflits :

- si plusieurs déclarations d'un même élément coexistent, seule la première est prise en compte ;
- si plusieurs déclarations d'attributs pour un même élément co-existent, alors on les concatène ;
- si un même attribut pour un même élément est déclaré plusieurs fois, c'est la première qui compte.

Modularisation

Seconde possibilité : utiliser des sections conditionnelles.
Ce sont des balises spécifiques aux DTD, qui permettent de laisser le choix à l'utilisateur de la DTD de valider ou non ces sections.
Syntaxiquement, ces balises ressemblent aux balises `CDATA` :

```
<![test[texte de la DTD ]]>
```

`test` vaut soit `INCLUDE`, soit `IGNORE` :

- S'il vaut `INCLUDE`, le texte de la DTD sera ajouté aux déclarations ;
- s'il vaut `IGNORE`...

Modularisation

- la valeur de `test` n'est pas remplacée manuellement !
- généralement, `test` vaut `%test-struct-cond;`
- et l'entité `test-struct-cond` est déclarée dans le sous-module interne du document : c'est donc au plus bas niveau qu'on décide de ce qui est activé !

Les structures conditionnelles peuvent être imbriquées, mais les plus externes ont la priorité.

Attention

On ne peut pas définir deux fois un même élément dans le même module de DTD.

Mais on peut définir deux fois une entité. Seule la première définition prise en compte.

Pour utiliser les sections conditionnelles pour une définition d'entité, on doit passer par des entités paramètres supplémentaires.

Exemple - la DTD

On propose une nouvelle DTD pour le message :

```
<![%test;[
   <!ENTITY % elt-signature "sign_complete">
]]>

<!ENTITY % elt-signature "signature">

<!ELEMENT message (destinataire,expediteur?,objet?,
                   corps,formulepolitesse,
                   %elt-signature;) >
```

Exemple - le document XML

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE mistigri SYSTEM "mistigri.dtd"
[
<!ENTITY % test "IGNORE">
]>
```