

XML

eXtensible Markup Language

Laurent Réveillère

Enseirb
Département Télécommunications

`Laurent.Reveillere@enseirb.fr`
`http://www.enseirb.fr/~reveille`

Origines

- ◆ Constat sur HTML
 - très bien adapté à la diffusion de l'information
 - mais peu structuré et pas extensible
- ◆ Nouveaux besoins pour le web
 - commerce électronique
 - fonds documentaires en ligne
 - adaptation du contenu au support (tél., PDA, ...)
 - intégration dans les systèmes d'information des entreprises (bases de données)
 - ...

XML

- ◆ Réponse unique à des besoins variés
- ◆ Langage universel d'échange de documents
 - sur le Web, entre machines, ...
- ◆ Défini par un organisme international
- ◆ Ensemble de technologies en pleine expansion
 - XLink, XPointer, XSchema, DOM, SAX, Xpath, XSL, XQuery, SOAP, WSDL, ...

Objectifs initiaux

- ◆ XML devra être directement utilisable sur Internet
- ◆ XML devra supporter une large variété d'applications
- ◆ XML devra être compatible avec SGML
- ◆ La création de pages XML devra être simple et rapide
- ◆ Les documents XML devront être d'une très grande lisibilité
- ◆ La syntaxe devra être formelle et concise
- ◆ La concision du code est d'une importance minime

XML : eXtensible Markup Language

- ◆ Format universel pour les documents et données structurés sur le Web
 - *successeur* de HTML
 - version simplifiée de SGML
- ◆ Création d'un groupe de travail au sein du W3C en 1996
- ◆ XML 1.0
 - recommandation officielle du W3C depuis 1998

Remarques sur XML

- ◆ Fournit une syntaxe, pas de sémantique
- ◆ Les balises n'ont pas de présentation ou de signification prédéfinie
- ◆ Ne définit que la structure et le contenu d'un document pas son comportement ni son traitement

Structure d'un document XML

- ◆ Prologue, éventuellement vide

```
<?xml version="1.0" standalone="yes" ?>
```

- ◆ Arbre d'éléments

```
<document>  
  <salutation> Bonjour! </salutation>  
</document>
```

- ◆ Commentaires et instructions de traitement
 - facultatifs

Prologue

◆ Déclarations XML (facultative)

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

- indique au processeur qui va traiter le document:
 - » version du langage XML utilisée
 - » codage de caractères utilisé
 - » document autonome ou pas (lien vers d'autres documents)

◆ Déclaration de type de document (facultative)

```
<!DOCTYPE exemple SYSTEM "exemple.dtd" [ déclarations ]>
```

- indique la structure particulière à laquelle doit se conformer un document (grammaire)

Structures (grammaires)

- ◆ Deux langages pour contraindre les structures
 - Définition de Type de Document (DTD)
 - » héritée de SGML
 - Schéma XML
 - » formalisme plus complet et plus rigoureux pour déclarer les contraintes structurales et les types
 - » exprimé en XML
- ◆ DTD et Schémas ne sont pas obligatoires
- ◆ Les document XML doivent au moins être bien formés

Structure d'une DTD

- ◆ Externe
 - définie en dehors du document
 - référencée par une *référence* DTD (URL)
 - » permet d'assurer une réutilisation de DTDs souvent utilisées
- ◆ Interne
 - spécialise une DTD externe générique
 - redéfinie certains types d'éléments

Arbre d'éléments

- ◆ Un et un seul élément (père) contient tous les autres
 - C'est l'élément racine du document.
- ◆ Tout élément distinct de la racine est totalement inclus dans son père.
 - non chevauchement des balises
 - `<nom> ... <prenom> ... </nom> ... </prenom>`

Structure d'un élément

```
<nom attr="valeur"> contenu </nom>
```

◆ Respect de la case des balises

- <NOM> ≠ <nom>

◆ Fermeture d'une balise de début de zone

- <nom> ... </nom>

◆ Raccourci pour les balises autonome

- <nom></nom> ≡ <nom/>

◆ contenu est le contenu d'un élément 😊 :

- vide
- texte
- autres éléments

- imbrication de texte et d'éléments
- instructions de traitement
- commentaires

Structure d'un élément (suite)

- ◆ Nom d'un élément : suite non vide de
 - caractères alphanumériques
 - tiret-souligné (*underscore*)
 - signe moins
 - point
 - caractère deux-points (:)
 - » ce caractère possède un sens particulier
- ◆ Contraintes à satisfaire
 - le premier caractère doit être alphabétique ou un tiret-souligné
 - les trois premiers caractères ne doivent pas former une chaîne dont la représentation en lettres minuscules est "xml".

Exemples de noms d'éléments

◆ Corrects

- `_toto`
- `Nom_société`
- `xsl:rule`
- `X.11`

◆ Incorrects

- `1998-catalogue`
- `XmlSpécification`
- `nom société`

Syntaxe d'un attribut

- ◆ **Paire** `nom="valeur"`
 - Permet de caractériser un élément
 - Un élément peut avoir plusieurs attributs (séparées par un espace)
 - mêmes règles pour le nom d'un attribut et le nom d'un élément
- ◆ **Exemples**
- ◆ **Valeur d'un attribut encadrée par des**
 - guillemets (") ou apostrophes simples (')

```
<cours matiere="prog-web" dern-modif="08/04/03">
```

```
<cours matiere="prog-web" dept="Telecoms" annee="2">
```

Commentaires

- ◆ Commencent par `<!--` et se termine par `-->`

```
<!-- ceci est un commentaire  
sur deux lignes -->
```

- ◆ Chaîne de caractères quelconque qui ne contient pas la chaîne `--`

- ◆ Exemple :

```
<!-- commentaire -->
```

oui

```
<!-- commentaire -- pas bien formé -->
```

non

Document bien formé

- ◆ Pas de déclaration de type de document dans le prologue
- ◆ Présence d'un arbre d'éléments
- ◆ Exemple

```
<?xml version="1.0" standalone="yes" ?>
<document>
  <salutation>
    Bonjour!
  </salutation>
</document>
```

Document valide

- ◆ Déclaration de type de document (DTD) dans le prologue
- ◆ Son arbre d'éléments respecte la structure définie par la déclaration de type
- ◆ Exemple

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE document [
    <!ELEMENT document (salutation)>
    <!ELEMENT salutation (#PCDATA)>
]>

<document>
  <salutation> Bonjour! </salutation>
</document>
```

Exemple de DTD externe

etudiant.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE etudiant SYSTEM "etudiant.dtd">
<etudiant>
  <nom> Toto </nom>
</etudiant>
```

etudiant.dtd

```
<!-- fichier etudiant.dtd. Exemple de DTD simple -->

<!-- la déclaration XML n'est pas obligatoire dans une DTD -->
<!-- permet de s'assurer que les documents qui la référence -->
<!-- utilisent la même version de XML -->
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>

<!-- Définition de l'élément racine -->
<!ELEMENT etudiant (nom)>

<!-- Un élément nom ne contient que du texte -->
<!ELEMENT nom (#PCDATA)>
```

Contenu d'une DTD

- ◆ Déclarations d'éléments
- ◆ Déclarations d'attributs
- ◆ Déclarations d'entités
 - référence à des caractères spéciaux, macros texte, ...
- ◆ Déclarations de notations
 - contenu externe non XML

- ◆ Commentaires

Déclaration d'élément

```
<!ELEMENT nom modèle>
```

- ELEMENT mot-clef impérativement en majuscules
- nom nom valide d'un élément
- modèle modèle de contenu de cet élément.
 - » cinq modèles de contenu
 - ◆ éléments
 - ◆ données
 - ◆ mixte
 - ◆ libre
 - ◆ vide

Modèle de contenu : éléments

- ◆ Séquence d'éléments fils (.., .., ..)
 - `<!ELEMENT chapitre (titre,intro,section)>`
- ◆ Alternative (.. | .. | ..)
 - `<!ELEMENT chapitre (titre,intro,(section|sections))>`
- ◆ Indicateurs d'occurrence *, +, ? d'un élément x
 - x une et une seule fois (défaut)
 - x? 0 ou 1 fois
 - x* 0, 1 ou plus d'une fois
 - x+ au moins 1 fois

Exemple

```
<!-- Un chapitre contient dans l'ordre un titre,
      éventuellement une introduction, puis au moins
      une section -->
<!ELEMENT chapitre (
  titre,
  intro?,
  section+ ) >

<!-- Une section est composée d'un titre et d'un corps -->
<!ELEMENT section (
  titre-section,
  corps-section ) >

<!-- Le corps d'une section est constitué d'une suite
      (possiblement vide) de paragraphes et de figures -->
<!ELEMENT corps-section (paragraphe | figure)* >
```

Modèle de contenu : données

- ◆ Élément ne contenant que des données
 - déclaration avec le mot-clé `#PCDATA`
- ◆ Exemple
 - `<!ELEMENT paragraphe (#PCDATA) >`

Modèle de contenu : mixte

- ◆ Permet de définir des modèles de contenu permettant de mêler données et éléments
- ◆ Forme du modèle
 - `(#PCDATA | nom1 | ... | nomn) *`
- ◆ Exemples de déclarations
 - `<!ELEMENT paragraphe (#PCDATA | em | exposant | indice | renvoi)* >`
 - `<!ELEMENT em (#PCDATA|exposant|indice)* >`
 - `<!ELEMENT exposant (#PCDATA) >`
 - `<!ELEMENT indice (#PCDATA) >`
- ◆ Exemple d'utilisation

`<paragraphe>` un paragraphe peut contenir du texte
`` mis en évidence `` ou en `<exposant>`
`exposant </exposant>` `</paragraphe>`

Modèle de contenu : libre

- ◆ Permet de définir un élément comme ayant un contenu quelconque
 - sous réserve que ce contenu respecte les règles générales du langage XML
- ◆ Exemple
 - `<!ELEMENT foo ANY>`
- ◆ Particulièrement utile lors de la création d'une DTD complexe

```
<!ELEMENT rapport ANY>
<!ELEMENT chapitre ANY>
<!ELEMENT section ANY>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (prenom+,nom)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

Modèle de contenu : vide

◆ Élément obligatoirement vide

- déclaration avec le mot-clé `EMPTY`.

◆ Exemple de déclarations

- `<!ELEMENT par (#PCDATA|bibref)* >`
- `<!ELEMENT bibref EMPTY>`
- `<!ATTLIST bibref ref IDREF #REQUIRED>`

◆ Exemple d'utilisation

- `<par>` pour une définition précise, voir `<bibref ref="reveillere:web-03"/>` `</par>`

◆ Remarques:

- syntaxe `<nom/>` obligatoire
- impossible de mêler `EMPTY` à une autre construction de modèle de contenu

Exemple

```
<?xml version="1.0"?>
<!ELEMENT EMAIL (TO+, FROM, CC*, BCC*, SUBJECT?, BODY?)>
<!ATTLIST EMAIL
  LANGUAGE (Western|Greek|Latin|Universal) "Western"
  ENCRYPTED CDATA #IMPLIED
  PRIORITY (NORMAL|LOW|HIGH) "NORMAL">

<!ELEMENT TO (#PCDATA)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT CC (#PCDATA)>

<!ELEMENT BCC (#PCDATA)>
<!ATTLIST BCC
  HIDDEN CDATA #FIXED "TRUE">

<!ELEMENT SUBJECT (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
```

Déclaration d'attributs

- ◆ Rappel : un attribut est une paire (nom, valeur) associée à un élément
- ◆ Permet de spécifier les attributs qui pourront ou devront être associés à des instances d'éléments
- ◆ Forme de la déclaration

```
<!ATTLIST nom-élément nom-attribut  
                type-attribut déclaration-par-défaut>
```

- ◆ Exemples

```
<!ELEMENT ex1 (#PCDATA)>  
<!ATTLIST ex1 xml:lang NMTOKEN #IMPLIED >  
<!ATTLIST ex1 target ID #IMPLIED >  
<!ATTLIST ex1 nb (1 | 2 | 3) '1'>
```

Type d'attribut

- ◆ CDATA
- ◆ énumération
- ◆ NMTOKEN, NMTOKENS
- ◆ ID, IDREF, IDREFS
- ◆ ENTITY, ENTITIES
- ◆ NOTATION

données textuelles

liste de choix

Jeton de nom

références internes au document

références externes non-XML

association à une application

Exemples

◆ Type CDATA

- `<!ATTLIST Element attr CDATA #REQUIRED>`
 - » `<Element attr=" texte "> ... </Element>`

◆ Type énumération

- `<!ELEMENT date (#PCDATA) >`
- `<!ATTLIST date format (ANSI|ISO|FR) #REQUIRED>`
 - » `<date format='FR'> 24 Mai 1998 </date>`
 - » `<date format='ISO'> 1998-05-24 </date>`

◆ Types NMTOKEN, NMTOKENS

- `<!ATTLIST nom-personne`
 - `titre NMTOKENS #IMPLIED`
 - `suffixe NMTOKEN #IMPLIED >`
- `<nom-personne titre="M. Dr." suffixe="fils"> ... </nom-personne>`

Exemples (2)

◆ Types ID, IDREF, IDREFS

- `<!ELEMENT section (#PCDATA|xref)* >`
- `<!ATTLIST section target ID #IMPLIED >`
- `<!ELEMENT xref EMPTY >`
- `<!ATTLIST xref ref IDREF #REQUIRED>`
 - » `<section target='X321'>` contenu d'une section `</section>`
 - » `<section>` autre section. Fait référence à la section X321 `<xref ref='X321' />` `</section>`

◆ Remarque: on ne référence pas en fait une section, mais un élément du document possédant un attribut de type ID et dont la valeur est X321

Déclaration par défaut

- ◆ #REQUIRED Doit figurer dans chaque instance de l'élément
- ◆ #IMPLIED Facultatif
- ◆ #FIXED valeur Peut figurer ou non dans le document
- ◆ valeur par défaut Peut figurer ou non dans le document

Exemples

◆ #IMPLIED

- `<!ATTLIST Element attr CDATA #IMPLIED>`
 - » `<Element attr=" texte "> ... </Element>`
 - » `<Element> ... </Element>`

◆ #FIXED

- `<!ELEMENT Document (#PCDATA)>`
- `<!ATTLIST Document version CDATA #FIXED "1.0">`
 - » `<Document> ... </Document>`
 - » `<Document version="1.0"> ... </Document>`

◆ valeur par défaut

- `<!ELEMENT Document (#PCDATA)>`
- `<!ATTLIST Document version CDATA "1.0">`
 - » `<Document> ... </Document>`
 - » `<Document version="2.0"> ... </Document>`

Déclarations d'entités

◆ Entités prédéfinies

- certains caractères, comme `<` `>` `&` ne peuvent être utilisés dans le texte d'un document
- ces caractères doivent être représentés par des codes prédéfinis, appelées *entités prédéfinies*

◆ <code><</code>	<code>&lt;</code>	<code>></code>	<code>&gt;</code>
◆ <code>é</code>	<code>&eacute;</code>	<code>è</code>	<code>&egrave;</code>
◆ <code>à</code>	<code>&agrave;</code>	<code>ç</code>	<code>&ccedil;</code>
◆ <code>ê</code>	<code>&ecirc</code>	<code>ô</code>	<code>&ocirc;</code>

Entités internes

- ◆ Entités définies par l'utilisateur

```
<!ENTITY nom-entité "valeur-entité">
```

- ◆ Exemple

```
<!DOCTYPE toto [  
  <!ENTITY copyright "&#x00A9; Editions titi">  
>  
<toto> &copyright </toto>
```

© Editions titi

- Remarque

- » 00A9 est le codage Unicode du caractère ©

Entités externes

```
<?xml version='1.0' ?>
<!DOCTYPE bouquin [
  <!ENTITY chapitre1 SYSTEM "chap1.xml">
  <!ENTITY chapitre2 SYSTEM "chap2.xml">
  <!ENTITY auteur "toto">
]>

<bouquin>
  <titre> XML par la joie </titre>
  <auteur> &auteur; </auteur>
  <intro> Il était une fois le Web... </intro>
  &chapitre1;
  &chapitre2;
</bouquin>
```

Entités paramètres

◆ Formes possibles

```
<!ENTITY % nom-entité "valeur-entité">  
<!ENTITY % nom-entité SYSTEM url>
```

◆ Exemples

- `<!ENTITY % IDREF_Req "IDREF #REQUIRED">`
- `<!ENTITY % IDREF_Opt "IDREF #IMPLIED">`
 - » `<!ATTLIST Livre
 IDLivre ID #REQUIRED
 editeur #IDREF_Req;
 publication #IDREF_Opt;
>`

Espaces de noms

Laurent Réveillère

Enseirb
Département Télécommunications

`Laurent.Reveillere@enseirb.fr`
`http://www.enseirb.fr/~reveille`

Espace de nomage

- ◆ Utilisation de balises provenant de plusieurs DTD dans un document XML

- ◆ Déclaration

```
<balise xmlns:nom="URL" ... >
```

- attribut réservé `xmlns` fournissant un nom et URL de la DTD associée
- peut être ajouté à n'importe quelle balise
- peut être utilisé plusieurs fois

```
» <html xmlns:m="http://www.w3.org/1998/Math/MathML"
      xmlns:s="http://www.w3.org/2000/svg" >
```

- l'espace de noms reste valide jusqu'à la balise fermante (ici `</html>`)
- les balises des DTD doivent être préfixées par *nom*:

- ◆ Utilisation

- `<s:svg width="2cm" height="0.6cm">`

Espace de nomage (suite)

- ◆ Possibilité de définir un espace par défaut
 - sans nom associé
- ◆ Exemple
 - Utilisation conjointe de 3 DTD
 - » XHTML reformulation de HTML en XML
 - » SVG figures géométriques
 - » MathML formules mathématiques

Exemple

```
<?xml version="1.0" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:s="http://www.w3.org/2000/svg"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
<head>
  <title>Exemple d'utilisation des espaces de noms</title>
</head>
<body> <h1>Les espaces de noms</h1>
<p>Un rectangle en SVG</p>
<s:svg width="120" height="35">
  <s:rect width="120" height="35" rx="12" fill="blue" stroke="#4C00E5" />
</s:svg>
<p>Une formule de math en MathML</p>
<m:math>
  <m:mroot>x+y</m:mroot>
</m:math>
</body> </html>
```

Exemple


```
<?xml version="1.0" encoding="UTF-8" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:svg="http://www.w3.org/2000/svg"
      xmlns:math="http://www.w3.org/1998/Math/MathML">
  <head>
    <title>Exemple d'utilisation des espaces de noms</title>
  </head>
  <body>
    <h1>Les espaces de noms</h1>
    <p>Un rectangle en SVG</p>
    <s:svg width="100px" height="50px">
      <s:rect width="100px" height="50px" fill="#4C00E5" />
    </s:svg>
    <p>Une formule de math en MathML</p>
    <m:math>
      <m:mroot>
        <m:math>√x+y</m:math>
      </m:mroot>
    </m:math>
  </body> </html>
```

Exemple d'utilisation des espaces de noms

File Edit XHTML XML Links Views Style
Special Attributes Annotations Help

Les espaces de noms

Un rectangle en SVG



Une formule de math en MathML

$$\sqrt{x+y}$$

Text \ p \ body \ html \ Document

= "#4C00E5" />