UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Introduction XML

Mathieu RAYNAL

mathieu.raynal@irit.fr

http://www.irit.fr/~Mathieu.Raynal

Le langage XML

- XML : eXtensible Markup Language
- Langage balisé
- On peut définir ses propres balises
- Structuration de documents

Petit historique

- Basé sur SGML : Standard Generaliezd Markup Language
  - Défini en 1986
  - Simplifié pour pouvoir être utilisé sur Internet

- 1ère version de XML en 1996
  - Par le XML Working Group
  - Reconnu par le W3C qu'à partir de Février 1998

http://www.w3.org/XML/

Quelques exemples de formats XML

- Quelques formats normalisés
  - XHTML : version XML de HTML
  - MathML : permet de définir des équations mathématiques
  - SVG : codage en XML de dessins vectoriels
  - RSS : gestion simple des news d'un site

Les avantages

- Lisibilité
  - Plus lisible qu'un fichier texte contenant un ensemble de données
- Arborescence
- Universalité / Portabilité
- Intégrabilité
  - utilisable par toute application qui possède un parseur
- Extensibilité
  - utilisable dans chaque domaine

Les différents types de balise

- Balise d'ouverture
  - Est constituée, au minimum, d'un nom entre chevrons
  - Peut contenir en plus des attributs

```
<NomElement …>
```

- Balise de fermeture
  - Est constituée uniquement d'un nom entre chevrons précédé du caractère « / »

```
</NomElement>
```

Attributs d'une balise

- Une balise d'ouverture peut contenir des attributs
  - Ils sont définis après le nom de la balise

- Un attribut est une paire nomAttribut="valeur" ou nomAttribut='valeur'

```
<NomElement attribut1='valeur' … attributN='valeur'/>
```

Un élément

- Un élément est composé
  - Une balise d'ouverture
  - Un contenu
  - Une balise de fermeture (avec le nom identique à la balise d'ouverture)

- Un élément vide
  - est constitué uniquement d'une balise d'ouverture
  - et se termine par « /> »

- Le contenu peut être constitué
  - De texte
  - D'autres éléments

```
<nomElt>
        <nomElt2>Mon texte</nomElt2>
</nomElt>
```

Les codes spéciaux

- Les commentaires
  - Ils débutent par <!--
  - Et finissent par -->
- Les entités prédéfinies

< → <	‘ → '
> → >	“ → "
& → &	

- Sections littérales : insérer une chaîne de caractères avec plusieurs caractères spéciaux

```
< ![CDATA [ Par exemple une <balise>]]
```

Les entités internes

- Les entités
  - Raccourci pour un contenu long
  - Elles sont définies dans la description du langage
  - Utilisées dans le document XML encadrées par un ‘&’ et un ‘;’

Exemple : Entité « m1 » pour remplacer “Master 1 SI”

Dans le fichier XML :
Les cours de &m1; sont très intéressants

Les entités externes

- Permet d'insérer un document (ou partie) XML, bien formé

```
<!DOCTYPE document[
<!ENTITY intro SYSTEM ‘’intro.xml’’>
]>
<document>
 …
 …
 &intro;
…
```

Entités externes et notation

- Permet d'insérer un document non-XML
  - Image
  - Vidéo
  - Fichier de description

```
<!DOCTYPE document[
<!NOTATION vrml SYSTEM ''nom de l'application à utiliser''>
<!ENTITY ev3D SYSTEM ''env.vrml'' NDATA vrml>
]>
<document>
<carte env='ev3D'/>
…
```

Un peu de vocabulaire

- Document bien formé
  - qui répond aux règles syntaxiques du format XML
    - Au moins un élément
    - Un seul élément racine

- Document valide
  - Qui adhère aux spécifications définies pour un type de fichier XML.
  - Ces spécifications sont définies dans une DTD ou un XML Schema

Les normes à respecter (1/2)

- Dans les balises
  - Un espace entre chaque attribut
  - Chaque valeur d'attribut comprise entre " " ou ' '
- Les noms (éléments ou attributs) :
  - 1ère lettre : alphanumérique ou _
  - Les autres : alphanumérique _ - .
  - Pas d'espace ou de retour à la ligne

Les normes à respecter (2/2)

- L'élément racine est unique

- Deux éléments ne peuvent s'entrecouper
  - Soit ils sont complètement disjoints

```
<Bold>Voici un exemple</Bold> <Italic>intéressant</Italic>
```

  - Soit l'un est inclus dans l'autre

```
<Bold>Voici un exemple<Italic>intéressant</Italic></Bold>
```

Structure d'un document XML (1/2)

- Un document XML est composé de :
  - Un prologue
  - Un arbre d'éléments (avec un élément racine unique)
  - Des commentaires

- Le prologue se compose de :
  - Une déclaration XML
  - Une déclaration du type de document

Structure d'un document XML (2/2)

- Déclaration XML

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

  - version : version du langage XML
  - encoding : codage de caractères utilisé
  - standalone : besoin ou non de déclarations extérieurs
    - Yes : toutes les déclarations nécessaires au traitement du document sont à l'intérieur du document
    - No : des déclarations sont dans d'autres fichiers

Les domaines nominaux

- Pour éviter les conflits de nom entre différentes définitions

- On associe à la racine du document
  - xmlns:prefixe=URI
    - URI du document dans lequel sont définis les éléments

```
<racine xmlns:math=‘’http://www.w3.org/TR/1998/REC-MathML-19980407.html>
…
<math:fn>
  <math:apply>
    <math:int/>
…
```

Exercice 1 : Ecrire le fichier XML

OLYMPIQUE LYONNAIS

Ligue 1 2018-2019 - 6e journée

DROIT AU BUT

Lyon	Marseille

Lyon - Groupama Stadium - 57287 spect. | dimanche 23 septembre 2018 - 21h00 - Canal + | arbitre : Clément TURPIN

Lyon		Marseille
H. Aouar 28'		39' F. Thauvin
B. Traoré 51'		82' C. Njie
B. Traoré 60'		
(sp) N. Fekir 74'		
P. Diop 29'		35' K. Strootman
		66' Luiz Gustavo
		83' D. Caleta-car

Lyon	Marseille
76' P. Diop ↓	↓ L. Ocampos 64'
L. Tousart ↑	↑ B. Kamara
88' N. Fekir ↓	↓ K. Strootman 76'
M. Terrier ↑	↑ C. Njie
59' M. Depay ↓	↓ K. Mitroglou 64'
M. Dembélé ↑	↑ V. Germain

entraîneur : Bruno Genesio

1 A. Lopes

14 L. Dubois | 6 Marcelo | 5 J. Denayer | 22 F. Mendy

28 T. Ndombele | 24 P. Diop ↓

10 B. Traoré 2⚽ | 18 N. Fekir C ↓ ⚽ | 8 H. Aouar ⚽

11 M. Depay ↓

11 K. Mitroglou ↓

5 L. Ocampos ↓ | 10 D. Payet | 26 F. Thauvin ⚽

12 K. Strootman ↓ 🟨 | 8 M. Sanson

18 J. Amavi | 19 Luiz Gustavo 🟨 | 15 D. Caleta-car 🟥 | 2 H. Sakai

16 Y. Pelé

entraîneur : Rudi Garcia

UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Comment construire son propre format XML ?

- DTD : Document Type Definition
- XML Schema

Document Type Definition (DTD)

- Permet de définir
  - les balises que l'on souhaite utiliser
  - Les relations entre les différentes balises

- Elle se trouve
  - Soit dans le prologue, en en-tête du fichier XML (DTD **Interne**)
  - Soit dans un fichier à part (DTD **Externe**)
    - Elle peut être partagée par plusieurs documents XML
    - Dans ce cas, elle est
      - **public** : La DTD est accessible via une URI
      - **privée** : La DTD se trouve sur le disque dur

Déclaration dans le document XML

- Commence par <!DOCTYPE et le nom de l'élément racine
  - Contient ensuite le type de DTD : public ou prive (SYSTEM)
  - Puis le chemin vers le fichier DTD

```
<!DOCTYPE elt_racine SYSTEM "be.dtd">
```

- Dans le cas d'une DTD interne, tout est situé à l'interieur de cette déclaration

```
<!DOCTYPE elt_racine [
Declaration
…
]>
```

Déclarer un élément

```
<!ELEMENT nom contenu>
```

- Nom : nom de l'élément
- Contenu : ce qui peut être placé entre les balises d'ouverture et de fermeture
  - Du texte : (#PCDATA)
  - Un élément vide : EMPTY
  - Une séquence d'éléments
  - Un choix entre plusieurs éléments
  - Tout type de contenu : ANY

Déclarer une séquence d'éléments

```
<!ELEMENT nom (elt1,elt2,…,eltN)>
```

- Tous les éléments déclarés devront
  - être présents dans le contenu de l'élément
  - Et dans l'ordre défini dans la DTD

- Les éléments utilisés dans la séquence doivent être déclarés dans la DTD

Déclarer un choix d'éléments

<!ELEMENT nom (elt1|elt2|…|eltN)>

- Le contenu utilisera un des éléments proposés
- Les éléments utilisés dans la séquence doivent être déclarés dans la DTD

Déclaration de contenu avancé

- Il est possible de combiner différents types de contenu
- Définir une fréquence d'apparition de chaque élément
  - ? : au plus une fois
  - * : de 0 à N fois
  - + : de 1 à N fois

Déclarer un attribut

```
<!ATTLIST  nomElt  nomAtt  typeAtt  [« valeurDef »]>
```

- Nécessite
  - NomElt : nom de l'élément qui va contenir l'attribut
  - NomAtt : nom de l'attribut
  - TypeAtt : donne le type de valeur que peut prendre l'attribut
- Les types possibles :
  - CDATA : toute chaîne de caractères
  - (val1|val2|…|valN) : les différentes valeurs qu'il peut prendre

Définition de la nécessité d'un attribut

- #FIXED : l'attribut aura une valeur par défaut dans le cas où l'attribut ne serait pas déclaré dans la balise d'ouverture.
- #IMPLIED : l'attribut n'est pas obligatoire
- #REQUIRED : l'attribut est obligatoire
- Dans tous les cas, il est possible de mettre une valeur par défaut.

Déclarer une entité

- Raccourci
  - soit pour être utilisé dans la DTD,

```
<!ENTITY % nom “texte à remplacer”>
```

    - Dans ce cas, peut être utilisé dans la DTD sous la forme **%nom;**

  - soit pour être utilisé dans le document XML.

```
<!ENTITY nom “texte à remplacer”>
```

Exemple de DTD

```
<!ELEMENT championnat (journee*)>
<!ATTLIST championnat  nom CDATA # REQUIRED>
<!ATTLIST championnat  saison CDATA>

<!ELEMENT journee (match*)>

<!ELEMENT match (equipe,score,equipe)>

<!ELEMENT equipe (#PCDATA)>

<!ELEMENT score (#PCDATA)>
```

Exercice 2 : Ecrire la DTD

- Ecrire la DTD permettant de définir le format du document réalisé à l'exercice 1

XML Schema

- Même chose qu'une DTD mais au format XML

- Le plus par rapport à une DTD
  - Typage des données
  - Héritage du contenu et des attributs d'un autre
  - Indique le nombre d'occurrences

XML Schema : Structure du fichier

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
        …
        …
</xsd:schema>
```

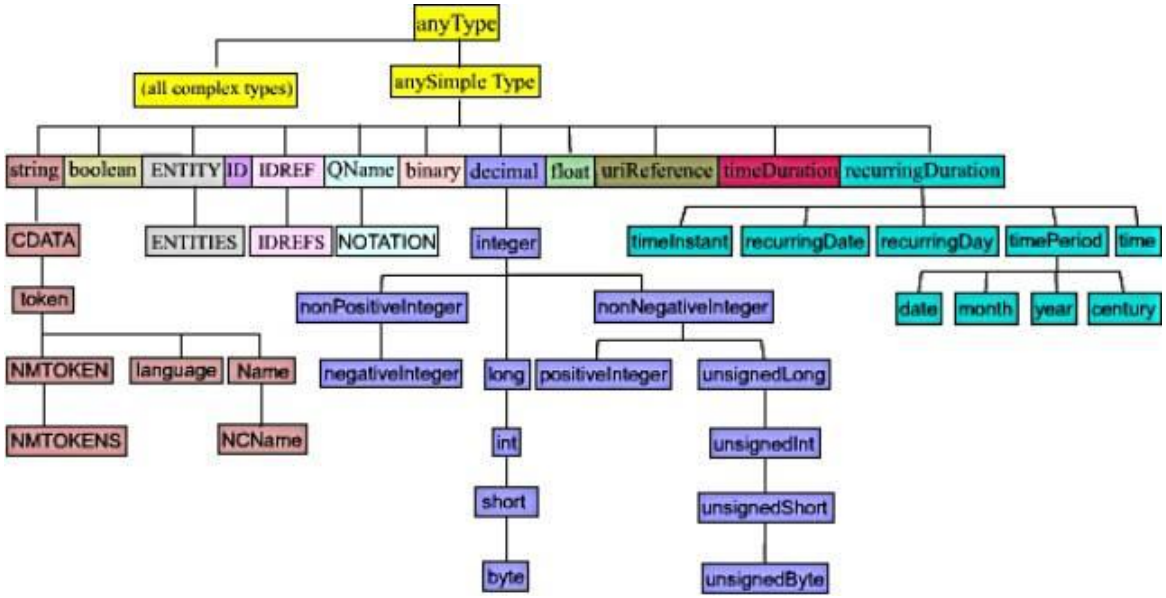
- xsd:schema est l'élément racine

XML Schema : Déclaration d'un élément

- Pour définir seulement un élément personne qui contient juste du texte entre les balises ouvrante et fermante

```
<xsd:element name="personne" type="xsd:string"/>
```

- Il existe plusieurs types prédéfinis



Déclaration de type complexe

- Le type séquence : équivalent à , dans une DTD

```
<xsd:complexType>
 <xsd:sequence>
  <xsd:element name="nom" type="xsd:string" />
  <xsd:element name="prénom" type="xsd:string" />
  <xsd:element name="dateDeNaissance" type="xsd:date" />
  <xsd:element name="adresse" type="xsd:string" />
  <xsd:element name="adresseElectronique" type="xsd:string" />
  <xsd:element name="téléphone" type="numéroDeTéléphone" />
 </xsd:sequence>
</xsd:complexType>
```

Déclaration de type complexe

- Type choice : identique à | dans la DTD

```
<xsd:complexType name="typePersonne">
 <sequence>
  <xsd:element name="nom" type="xsd:string" />
  <xsd:element name="prénom" type="xsd:string" />
  <xsd:element name="dateDeNaissance" type="xsd:date" />
  <xsd:choice>
   <xsd:element name="adresse" type="xsd:string" />
   <xsd:element name="adresseElectronique" type="xsd:string" />
  </xsd:choice>
 </sequence>
 <xsd:element name="téléphone" type="numéroDeTéléphone" />
</xsd:complexType>
```

Déclaration d'un attribut

- La déclaration se trouve dans une balise complexType

<xsd:attribute name="" type="" use="" default="" />

  - use : required | optional
  - default : doit être conforme au type déclaré

```
<xsd:element name="elt">
  <xsd:complexType>
    <xsd:attribute name="attr" type="xsd:string" use="optional" />
  </xsd:complexType>
</xsd:element>
```

Déclaration de type complexe

- Possibilité de rajouter minOccurs et maxOccurs
  - De 0
  - à unbounded

- Possibilité de faire référence à une balise précédemment définie

```
<xsd:element ref="">
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="CHAMPIONNAT">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="JOURNEE" maxOccurs="38">
    <xs:complexType>
            <xs:sequence>
            <xs:element name="RENCONTRE" maxOccurs="10">
                    <xs:complexType>
                        <xs:attribute name="DOMICILE" type="xs:string" />
                        <xs:attribute name="EXTERIEUR" type="xs:string" />
                        <xs:attribute name="SCORED" type="xs:string" />
                        <xs:attribute name="SCOREE" type="xs:string" />
                    </xs:complexType>
             </xs:element>
             </xs:sequence>
             <xs:attribute name="NUMERO" type="xs:integer" />
             <xs:attribute name="DATE"   type="xs:string" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="DIVISION" type="xs:integer" />
  <xs:attribute name="SAISON"   type="xs:string" />
 </xs:complexType>
</xs:element>

</xs:schema>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:attribute name="DOMICILE"  type="xs:string"  />
<xs:attribute name="EXTERIEUR" type="xs:string"  />
<xs:attribute name="SCORED"    type="xs:string"  />
<xs:attribute name="SCOREE"    type="xs:string"  />
<xs:attribute name="NUMERO"    type="xs:integer" />
<xs:attribute name="DATE"      type="xs:string"  />
<xs:attribute name="DIVISION"  type="xs:integer" />
<xs:attribute name="SAISON"    type="xs:string"  />

<xs:element name="RENCONTRE">
 <xs:complexType>
      <xs:attribute ref="DOMICILE"  />
      <xs:attribute ref="EXTERIEUR" />
      <xs:attribute ref="SCORED"    />
      <xs:attribute ref="SCOREE"    />
 </xs:complexType>
</xs:element>

<xs:element name="JOURNEE">
 <xs:complexType>
   <xs:sequence>
     <xs:element ref="RENCONTRE" maxOccurs="10" />
   </xs:sequence>
   <xs:attribute ref="NUMERO" />
   <xs:attribute ref="DATE"   />
 </xs:complexType>
</xs:element>

<xs:element name="CHAMPIONNAT">
```

Exercice 3 : Ecrire le schéma XML

- Ecrire le schéma XML permettant de définir le format du document réalisé à l'exercice 1