DTD
&
XML Schema

Sébastien Laborie
Sebastien.Laborie@iutbayonne.univ-pau.fr

Christian Sallaberry
Christian.Sallaberry@univ-pau.fr

DTD

- Le rôle d'une DTD (Document Type Definition) est de **définir la structure** d'un document XML.
- Elle va permettre de répondre aux questions suivantes :
  - Quels sont les noms des éléments que je souhaite exploiter au sein de mon document XML ?
  - Quels sont les attributs que je souhaite associer à un élément ?
  - Un attribut donné est-il obligatoire ou facultatif dans un élément ?
  - Est-ce qu'un élément imbrique d'autres éléments, et si oui lesquels ?
  - Est-ce qu'un élément est toujours vide ou non ?
- Une DTD est une sorte de **grammaire**.
  Tout document XML qui fait référence à cette grammaire doit la respecter.

2

Un exemple de DTD

- Une DTD se décrit de la forme suivante :

Déclaration d'éléments
Déclaration d'attributs
Déclaration d'entités

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ATTLIST body lang CDATA #IMPLIED
               signature CDATA #REQUIRED>
<!ENTITY sl "Sébastien Laborie">
...
```

- Elle peut s'écrire dans un fichier **.dtd** ou bien à l'intérieur d'un fichier XML.

3

Déclaration des éléments

- Tout élément est déclaré :
  - Soit à l'aide d'un **nom** et d'une **catégorie.**
  - Soit à l'aide d'un **nom** et d'un **type de contenu**.

```
<!ELEMENT nom categorie>
<!ELEMENT nom (type-contenu)>
```

① Un élément peut appartenir à une catégorie :
  - **Vide** : il ne contiendra ni de texte, ni d'autres éléments.
  - **Quelconque** : il contiendra n'importe quel type de contenu.

② Un élément peut contenir différents types de contenu :
  - **Texte** : il contiendra une séquence de caractères.
  - **Séquence** d'éléments : il contiendra une suite ordonnée d'éléments.
  - **Alternative** d'éléments : il pourra potentiellement contenir les éléments spécifiés.

- Un élément ne peut être déclaré qu'une seule fois.
- Tout élément utilisé dans la DTD doit être déclaré.

4

Déclaration des éléments

- Déclaration d'un élément de catégorie **vide**.

```
<!ELEMENT elt EMPTY>
```

- L'élément ayant comme nom *elt* sera vide.
  - Exemple : <elt />
- Pour rappel, un élément vide ne peut ni contenir de texte, ni contenir d'autres éléments.
- Par contre, un élément de type vide peut bien sûr avoir des attributs.

5

Déclaration des éléments

- Il est possible de déclarer des éléments **quelconques** :

```
<!ELEMENT elt ANY>
```

- Cet élément peut contenir tout autre élément défini dans la DTD (c-à-d, pas d'éléments non-déclarés).
- Cet élément peut aussi contenir du texte.
- Cet élément est « risqué » car il ne permet pas de contrôler vraiment les éléments qui seront inclus dans *elt*.

6

Déclaration des éléments

- Déclaration d'un élément de type **texte**.

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

- L'élément ayant comme nom *elt* contiendra du texte.
  - Exemple : <elt>Je suis un élément qui contient du texte.</elt>
- Par conséquent, dans le XML, cet élément ne doit pas contenir les caractères suivants : <, >, &, ' et ".
- De plus, l'élément ne peut pas contenir d'autres éléments.

Déclaration des éléments

- Déclaration d'un élément de type **séquence**.

```
<!ELEMENT elt (elt1, elt2, elt3)>
```

- L'élément ayant comme nom *elt* contiendra une liste ordonnée d'éléments (l'ordre doit être respecté).
  - Exemple :

```
<elt>
    <elt1>...</elt1>
    <elt2>...</elt2>
    <elt3>...</elt3>
</elt>
```

Déclaration des éléments

- Déclaration d'un élément de type **alternative**.

```
<!ELEMENT elt (elt1 | elt2 | elt3)>
```

- L'élément ayant comme nom *elt* pourra contenir soit *elt1*, soit *elt2* ou soit *elt3*.
  - Exemple :
```
<elt>
  <elt1>...</elt1>
</elt>
<elt>
  <elt3>...</elt3>
</elt>
```

- L'élément ne peut être vide dans notre exemple.
  De plus, il ne peut contenir deux éléments.

Déclaration des éléments

- Des **indicateurs d'occurrence** peuvent s'appliquer à chaque élément, ceci est également utile dans et pour des séquences ou alternatives d'éléments :
  - **?** : symbolise 0 ou une seule occurrence de l'élément.
  - **+** : symbolise une ou plusieurs occurrences de l'élément.
  - ***** : symbolise 0 ou plusieurs occurrences de l'élément.

- Exemples :

```
<!ELEMENT elt (elt1, elt2?, elt3+, elt4*)>
```

≠

```
<!ELEMENT elt (elt1* | elt2* | elt3*)>
<!ELEMENT elt (elt1 | elt2 | elt3)*>
```

Déclaration des éléments

- Il est possible de déclarer des éléments **mixtes** :

```
<!ELEMENT elt (#PCDATA | elt1)*>
```

- Ces éléments peuvent contenir du texte ou bien d'autres éléments (rq., #PCDATA doit être en premier).

- Exemple :

```
<elt>
  <elt1>...</elt1>
   Salut !
  <elt1>...</elt1>
</elt>
<elt />
<elt>
  Salut !
</elt>
```


Questions

- **Déclarer les éléments de la recette de cuisine.**
  (on ne contrôle pas pour le moment les attributs)

- **Que doit-on écrire pour pouvoir associer la DTD au fichier *recettes.xml* ? (cf., cours 1)**

Utilisation de la DTD

- Un document XML peut faire référence à une DTD. (cf., Cours 1).
  - Elle peut être **déclarée en interne** directement dans le prologue.

```
<!DOCTYPE collection [
      <!-- Contenu de la DTD à renseigner ici -->
]>
```

  - Elle peut faire **référence à une DTD externe**.
    - Votre DTD locale (ou mise à disposition sur le Web)

```
<!DOCTYPE collection SYSTEM "maGrammaire.dtd">
```

```
<!DOCTYPE collection SYSTEM "http://www.ex.com/maGrammaire.dtd">
```

    - Référence à une DTD connue avec un FPI (ex., XHTML)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

13

Déclaration des attributs

- Il est possible de déclarer et d'attacher un ensemble d'attributs spécifiques à un élément. Cette déclaration s'effectue par le mot-clé **ATTLIST**.

```
<!ELEMENT elt (…)>
<!ATTLIST elt nom type valeur>
```

- Chaque attribut défini dans la liste possède **un nom**, **un type** et **une valeur par défaut**.

- Exemple :

```
<!ELEMENT elt (…)>
<!ATTLIST elt attr1 CDATA  "0"
              attr2 CDATA  "" >
```

14

Déclaration des attributs

- Il existe différents types d'attributs :
  - **CDATA** : Il s'agit de texte.
  - **NMTOKEN** : Un seul mot sans espace, ni ponctuation.
  - **ID** : Identifiant unique de l'élément.
  - **IDREF** : Une référence vers un identifiant du document.
  - **IDREFS** : Des références vers plusieurs identifiants du document.
- Exemple :

```
<!ELEMENT elt(…)>
<!ATTLIST elt attr1 CDATA  "jus de fruit"
              attr2 NMTOKEN  "euro" >
```

- Attention :
  - Il ne peut y avoir deux ID pour une même liste d'attributs concernant un élément.
  - Les valeurs des ID doivent tous être différents au sein du document XML.

15

Déclaration des attributs

- Un attribut peut faire l'objet de contraintes :
  - **Valeur par défaut**
  - **Requis** (#REQUIRED) : l'attribut est obligatoire.
  - **Optionnel** (#IMPLIED) : l'attribut peut être omis.
  - **Fixe** (#FIXED) : l'attribut contient une valeur fixe (l'utilisateur ne peut la changer).
- Exemple :

```
<!ELEMENT elt(…)>
<!ATTLIST elt attr1 CDATA "0"
              attr2 CDATA #REQUIRED
              attr3 CDATA #IMPLIED
              attr4 CDATA #FIXED "valeur" >
```

16

Déclaration des attributs

- Un attribut peut être de type **énuméré** :
  - La liste des valeurs possibles pour un attribut peut être limitée.
  - À la place du type, il suffit de spécifier toutes les alternatives possibles comme valeur.
- Exemple :

```
<!ELEMENT elt(…)>
<!ATTLIST elt attr1 (val1 | val2) #REQUIRED
              attr2 (val3 | val4)  "val3" >
```


Questions

- **Déclarer les attributs de la recette de cuisine pour les éléments qui les exploitent.**
- **Valider la collection de recettes de cuisine.**

Déclaration d'entités

- Il est possible de définir ses propres **entités** dans une DTD.

```
<!ENTITY sl  "Sébastien Laborie" >
```

- Ceci évite de répéter plusieurs fois le même texte.
- Ainsi, nous pouvons exploiter simplement cette entité dans le document XML, en utilisant le **&** et le **;**.
- Exemple :

```
<elt>&sl; &amp; co.</elt>
```

19

Document XML valide

- Un document XML est dit **« valide »** :
  - **Si le document XML est « bien formé ».**
  - **Si le document XML respecte toutes les règles de la DTD.**
- Un document XML « valide » assure **l'interopérabilité** entre différents systèmes exploitant ce type de document.
  - Par exemple, une page Web « valide » assure qu'elle pourra être correctement interprétée par la majorité des navigateurs.

21

Exercice !

Créez la DTD du Cours 1 Exercice 5

Limites des DTD

- Une DTD n'est pas écrite en XML.
- On ne peut contrôler le nombre d'éléments contenu dans une balise.
- On ne peut contrôler les types des valeurs des attributs.
- On ne peut décrire ses propres types de données avec une DTD.
- On ne peut faire des références vers d'autres DTD.

XML Schema

- Ce langage fournit des nouveautés par rapport au DTD pour **mieux contrôler la structure** des documents XML.
- Les schémas XML sont décrits en XML.
- Le typage des données peut être exploité.
  - Beaucoup de types sont prédéfinis : date, booléen, entier, texte…
- Un nombre minimum et maximum de présence d'un élément peut être renseigné.

Structure de base

- Comme tout document XML, un schéma XML contient un prologue ainsi qu'un élément racine.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001//XMLSchema">
   <!-- déclarations d'éléments, d'attributs et de types ici -->
</xsd:schema>
```

- L'élément racine est xsd:schema.
- Tout élément du langage XML Schema que vous souhaitez utiliser doit commencer par xsd:.

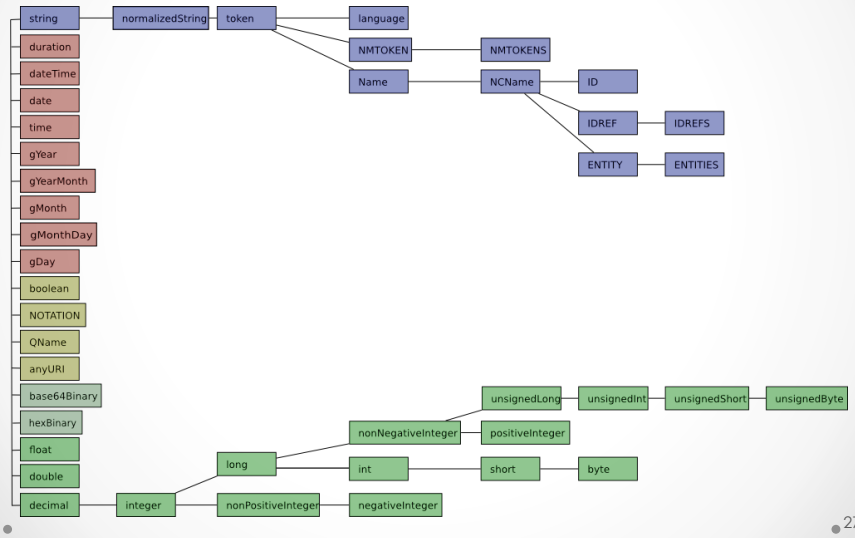
Déclaration d'éléments simples

- Un **élément simple** est un élément qui ne contient que des chaînes de caractères.
  (il ne peut pas contenir d'autres éléments, ni d'attributs)
- Cette chaîne de caractères peut correspondre à :
  - Des types prédéfinis : xsd:string, xsd:decimal, xsd:integer, xsd:boolean, xsd:date…
  - Vos propres types de données.
- Dans un schéma XML, un élément simple se déclare avec la balise xsd:element.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001//XMLSchema">
  <xsd:element name="note" type="xsd:integer" />
</xsd:schema>
```

26

Les types d'éléments simples



27

Définir des restrictions

- Un élément simple peut contenir des **restrictions**.
- Il existe des restrictions sur des plages de données, des séries de valeurs, des longueurs de caractères…
- Exemple :

```
<xs:element name="note">
  <xs:simpleType>
    <xs:restriction base="xsd:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

28

Déclaration d'éléments complexes

- Un **élément complexe** est un élément qui peut contenir d'autres éléments ou bien des attributs.
- Il existe 4 types d'éléments complexes :
  - Les éléments vides.
  - Les éléments qui contiennent d'autres éléments.
  - Les éléments (avec des attributs) qui contiennent uniquement du texte.
  - Les éléments qui contiennent du texte et d'autres éléments.
- Dans un schéma XML, un élément complexe se déclare en utilisant la balise xsd:complexType.

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  ...
</xsd:complexType>
```

29

Déclaration d'éléments complexes

1. **Les éléments vides** ne contiennent pas de texte, ni d'autres éléments.

- Un élément vide peut bien sûr contenir des attributs.

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:attribute name="nom" type="xsd:string" />
  <xsd:attribute name="prenom" type="xsd:string" />
</xsd:complexType>
```

- Un attribut est optionnel par défaut.
- Pour que l'attribut soit obligatoire :

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:attribute name="nom" type="xsd:string" use="required" />
  <xsd:attribute name="prenom" type="xsd:string" />
</xsd:complexType>
```

30

Déclaration d'éléments complexes

2. **Les éléments qui contiennent d'autres éléments.**
   - Une séquence d'éléments :

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:sequence>
     <xsd:element name="nom" type="xsd:string" />
     <xsd:element name="prenom" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
```

   - Une alternative d'éléments :

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:choice>
     <xsd:element name="nom" type="xsd:string" />
     <xsd:element name="prenom" type="xsd:string" />
  </xsd:choice>
</xsd:complexType>
```

31

Déclaration d'éléments complexes

2. **Les éléments qui contiennent d'autres éléments.**
   - Il est possible de contrôler le nombre minimal et maximal d'occurrences d'un élément.
   - Utilisation de minOccurs et maxOccurs.

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:sequence>
     <xsd:element name="nom" type="xsd:string" />
     <xsd:element name="prenom" type="xsd:string" minOccurs="1" maxOccurs="3"/>
  </xsd:sequence>
</xsd:complexType>
```

   - Les attributs **minOccurs** et **maxOccurs** ont par défaut la valeur 1.
   - La valeur **unbounded** peut être associée à maxOccurs. (infini)

Déclaration d'éléments complexes

3. **Les éléments qui contiennent uniquement du texte.**

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:simpleContent>
     <xsd:extension base="xsd:string" >
        <xsd:attribute name="pays" type="xsd:string" />
     </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

   - Dans cet exemple, l'élément *personne* contient du texte et un attribut ayant pour nom *pays*.

Déclaration d'éléments complexes

4. **Les éléments qui contiennent du texte et d'autres éléments (élément mixte).**

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo" mixed="true">
  <xsd:sequence>
     <xsd:element name="nom" type="xsd:string" />
     <xsd:element name="prenom" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
```

- Exemple correspondant :

```
<personne>
   Je suis <nom>Laborie</nom> <prenom>Sébastien</prenom>.
</personne>
```

34

Déclaration d'éléments quelconques

- Comme pour les DTD, il peut y avoir **des éléments quelconques.**

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:sequence>
     <xsd:element name="nom" type="xsd:string" />
     <xsd:element name="prenom" type="xsd:string" />
     <xsd:any />
  </xsd:sequence>
</xsd:complexType>
```

- Il est possible de faire de même pour les attributs.

```
<xsd:element name="personne" type="personneInfo" />
<xsd:complexType name="personneInfo">
  <xsd:anyAttribute />
</xsd:complexType>
```

35

Utilisation du XML Schema dans un document XML

- Pour valider votre document XML à l'aide d'un schéma XML, vous devez ajouter à votre XML ceci :

```
<collection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="recettes.xsd">
   . . .
</collection>
```

- Bien sûr, votre document XML peut aussi faire référence à une DTD dans le prologue.

```
<?xml version="1.0" ?>
<!DOCTYPE collection SYSTEM "recettes.dtd">
<collection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="recettes.xsd">
   . . .
</collection>
```

36

Résumé

- Une DTD permet assez simplement de contrôler la structure d'un document XML.
- Un schéma XML est plus complexe, il permet de contrôler avec plus de précisions la structure ainsi que les types des éléments ou des attributs.
- Un schéma XML est décrit en XML. Il peut être utilisé par toutes les technologies/applications XML.
  - Il peut être interrogé (XPath, XQuery) ou bien être transformé (XSLT).

37

Exercices !

Créez vos Schéma XML (pour les recettes de cuisine) (pour le Cours 1 Exo 5)