1 Les principes de bases de XML

2 Typer des données avec des DTD

3 Faire des requêtes avec XPATH

4 Typer des données avec XML-Schema

5 XML-Schema et les espaces de noms

6 Transformer des données avec XSLT

7 Programmer avec XQUERY

## 2 Typer des données avec des DTD

- Pourquoi définir des DTD?
- Comment lier une DTD à un document ?
- Comment valider un document XML?
- Structure d'une DTD

# Motivation

c.f. l'exercice sur le fichier `maisons.xml`

# DTD

## DTD ?

Une DTD est un ensemble de règles, chacune d'entre-elles décrivant le contenu autorisé d'un élément ou l'ensemble des attributs existant pour un élément.

# Lier une DTD à un document

Une DTD peut être associée de 3 façons à un document XML :

1. DTD externe : toutes les règles à respecter sont décrites dans un fichier spécifique
2. DTD interne : toutes les règles sont dans le fichier XML
3. DTD mixte : certaines règles sont décrites dans un fichier spécifique et certaines règles sont dans le fichier XML

# 1 - DTD Externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour SYSTEM "bonjour.dtd">
<bonjour>Hello world!</bonjour>
```

# 1 - DTD Externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour SYSTEM
        "http://www.chez-moi.fr/dtd/bonjour.dtd">
<bonjour>Hello world!</bonjour>
```

# 1 - DTD Externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
 "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd
<html>
  <head>
    <meta   content="text/html; charset=ISO-8859-1"
            http-equiv="content-type" />
    <title>
      Formations en Informatique de Lille 1
    </title>
    ...
```

# 2 - DTD interne

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour [
  <!ELEMENT Bonjour (#PCDATA)>
]>
<bonjour>bonjour tout le monde</bonjour>
```

# 3 - DTD mixte

```
<?xml version="1.0"?>
<!DOCTYPE bonjour SYSTEM "bonjour.dtd" [
  <!ELEMENT bonjour (#PCDATA)>
]>
<bonjour>bonjour tout le monde</bonjour>
```

# Validation

A partir du moment où une DTD est associée au document à valider, on peut valider à l'aide :

- d'un logiciel spécialisé dans le traitement des documents XML (*XMLSpy*, *XML Exchanger Lite*, ...)
- par programme en utilisant les bibliothèques de traitement de XML disponibles dans beaucoup de langages (*java*, *php* , *perl*, ...)

# Le fichier XMLParser.java

```java
import org.xml.sax.XMLReader ;
import org.xml.sax.helpers.DefaultHandler ;
import org.xml.sax.helpers.XMLReaderFactory ;

/**
 *  Analyseur XML pour une validation par rapport a une DTD.
 *  La validation se fait a la volee, en lisant le document.
 *  C'est un analyseur SAX -> On ne construit pas l'arbre DOM du document.
 *
 */
public class XMLParser {

  /**
   *  Methode de validation : Executer "java XMLParser leDocumentAValider.xml"
   *  On vérifie que le document est conforme a la DTD qui lui est liee
   *
   *@param  args            ligne de commande = le nom du fichier XML a valider
   *@exception  Exception  Si probleme lors de la creation des objets.
   */
  public static void main(String[] args) {
    try {
      XMLReader saxReader = XMLReaderFactory.createXMLReader(); //comme dans TP1
      saxReader.setFeature("http://xml.org/sax/features/validation", true); // c'est là la no
      //saxReader.setContentHandler(new MonHandlerAMoi());// si on veut
      saxReader.parse(args[0]);
    } catch (Exception t) {
      t.printStackTrace();
    }
  }
```

Une DTD contient

- des déclarations d'éléments,
- des déclarations d'attributs,
- des déclarations d'entités et de notations
- des commentaires.

selon le schéma générique `<!mot-clé param1 param2 ...>`

# Déclaration d'élément

```
<!ELEMENT nom categorie>
```

ou

```
<!ELEMENT nom (modele)>
```

# Catégories

```
<!ELEMENT nom categorie>
```

avec `categorie` pouvant valoir :

- ANY type d'élément pouvant contenir n'importe quel élément défini dans la DTD
- EMPTY type d'élément vide (on peut avoir des attributs, mais c'est tout)

# Modèles

```
<!ELEMENT nom (modele)>
```

avec différentes possibilités de modèles :

- purement textuel : (#PCDATA)
- expression régulière de sous-éléments :
  - cp ::= (Name | choice | seq) ('?' | '*' | '+')?
  - seq ::= '(' cp ( ',' cp )* ')'
  - choice ::= '(' cp ( '|' cp )+ ')'
  - opérateurs unaires : ?(0 ou 1 fois), *(0,1 ou plusieurs fois), +(au moins 1 fois).
- contenu mixte : mélange entre du texte #PCDATA et des sous-éléments

# Exemple

```
<!ELEMENT catalogue  (stage)*>
        <!ELEMENT stage (intitule, prerequis?)>
        <!ELEMENT intitule (#PCDATA)>
        <!ELEMENT prerequis (#PCDATA | xref )*>
        <!ELEMENT xref EMPTY>
```

# Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de donnees</intitule>
    <prerequis>
      connaitre les langages SQL et HTML
    </prerequis>
   </stage>
   <stage>
     <intitule>XML programmation</intitule>
     <prerequis>
       avoir suivi le stage de XML et les bases de donnees
     </prerequis>
  </stage>
</catalogue>
```

# Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de donnees</intitule>
   </stage>
   <stage>
     <intitule>XML programmation</intitule>
     <prerequis>
       avoir suivi le stage de XML et les bases de donnees
     </prerequis>
  </stage>
</catalogue>
```

# Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de donnees</intitule>
    <prerequis>
      connaitre les <xref/> et <xref/>
    </prerequis>
   </stage>
   <stage>
     <intitule>XML programmation</intitule>
     <prerequis>
       avoir suivi le stage de XML et les bases de donnees
     </prerequis>
  </stage>
</catalogue>
```

# Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de donnees</intitule>
    <prerequis>
      connaitre les <xref> langages SQL et HTML</xref>
    </prerequis>
   </stage>
   <stage>
     <jour>lundi</jour>
     <intitule>XML programmation</intitule>
     <prerequis>
       avoir suivi le stage de XML et les bases de donnees
     </prerequis>
  </stage>
</catalogue>
```

# Exemple

```
<catalogue>
</catalogue>
```

# Unique Particle Attribution

## UPA ?

Dans le cas où le contenu d'un élément est défini sous la forme d'une expression régulière, celle-ci doit respecter la règle *UPA* .

# Unique Particle Attribution

## Définition du W3C

*A content model must be formed such that during validation of an element information item sequence, the particle component contained directly, indirectly or implicitly therein with which to attempt to validate each item in the sequence in turn can be uniquely determined without examining the content or attributes of that item, and without any information about the items in the remainder of the sequence.*

# Unique Particle Attribution

## Précision (aveu?) du W3C

*Given the presence of element substitution groups and wildcards, the concise expression of this constraint is difficult, see section* Analysis of the Unique Particle Attribution Constraint (non-normative) (H) *in*

`http://www.w3.org/TR/xmlschema-1/#non-ambig`

*for further discussion.*

# Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE stage SYSTEM "./dtd1.dtd">

<stage>
    <intitule>T1</intitule>
    <prerequis>T2</prerequis>
</stage>

<!--   dtd1.dtd ci-dessous -->

<!ELEMENT stage ((intitule*| prerequis),(intitule*| prerequis)*)>
    <!ELEMENT intitule (#PCDATA)>
    <!ELEMENT prerequis (#PCDATA)>
```

# Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE stage SYSTEM "./dtd2.dtd">

<stage>
    <intitule>T1</intitule>
    <prerequis>T2</prerequis>
</stage>

<!--  dtd2.dtd ci-dessous -->

<!ELEMENT stage (intitule*| prerequis)+>
    <!ELEMENT intitule (#PCDATA)>
    <!ELEMENT prerequis (#PCDATA)>
```

# Unique Particle Attribution

## La réalité

*La très grande majorité des logiciels de validation se moquent complètement de savoir si la DTD satisfait ou non la propriété UPA, mais dans le cas d'une DTD qui ne satisfait pas cette propiété,*

- *certains validateurs fonctionnent parfaitement*
- *d'autres donnent des réponses erronées !*

# Déclaration d'attributs

```
<!ATTLIST nom-element nom-attribut type valeur-defaut>
```

- Le type d'un attribut définit les valeurs qu'il peut prendre.
  - CDATA : valeur chaîne de caractères
  - ID, IDREF, IDREFS permettent de définir des références à l'intérieur du document.
  - Une liste de choix possibles parmi un ensemble de noms symboliques.
- La déclaration par défaut peut prendre quatre formes :
  - la valeur par défaut de l'attribut,
  - #REQUIRED indique que l'attribut est obligatoire
  - #IMPLIED indique que la présence est facultative.
  - #FIXED valeur indique que l'attribut prend toujours la même valeur, dans toute instance de l'élément déclaré **si elle existe**.

# Exemples de déclarations d'attributs

```
<!ATTLIST en-tete date CDATA #REQUIRED>
```

Cette règle spécifie que l'élément `en-tete` **doit** posséder un attribut qui a pour nom `date` de type `CDATA` (c'est à dire du texte)

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">
```

```
<document version="1.0">
    ...
</document>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">
```

```
<document version="2.0">
    ...
</document>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">

<document>
    ...
</document>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">
```

```
<document version="1.0">
    ...
</document>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">
```

```
<document version="2.0">
    ...
</document>
```

KO

# Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">

<document>
    ...
</document>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre ( Mlle | Mme | M. ) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="Mme" nom-epouse="Lenoir">Martin</nom>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre ( Mlle | Mme | M. ) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="M." nom-epouse="Lenoir">Martin</nom>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre ( Mlle | Mme | M. ) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="M.">Martin</nom>
```

OK

# Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre ( Mlle | Mme | M. ) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="Madame" nom-epouse="Lenoir">Martin</nom>
```

KO

# Attributs ID, IDREF,IDREFS

Un attribut ID sert à référencer un élément, la valeur de cette référence pouvant être rappelée dans des attributs IDREF ou IDREFS.

Un élément ne peut avoir au plus qu'un attribut ID et la valeur associée doit être unique dans le document XML. Cette valeur doit être un *nom* XML.

Le type par défaut est pour un attribut ID obligatoirement #REQUIRED ou #IMPLIED.

Une valeur utilisée dans un attribut IDREF ou IDREFS doit obligatoirement correspondre à celle d'un attribut ID.

# document.dtd

```
<!ELEMENT  document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
  </personne>
  <personne id="id-2">
    <nom>Durand</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
  </personne>
  <personne id="id-1">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
  </personne>
  <personne id="id-1">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-3">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-3">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

# document.dtd

```
<!ELEMENT  document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREFS #IMPLIED>
```

# Exemple ID, IDREF,IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardelpique</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

# Les entités

- Les entités internes : "macros" qui sont utilisées dans le document XML validé par la DTD.
  `<!ENTITY euro "&#8364;">`, utilisation `&euro;`
- Les entités paramétriques : elles servent à définir des "macros" qui sont utilisées ailleurs **dans** la DTD.
  `<!ENTITY % editeur "O'Reilly">`, utilisation `%editeur;`
- Les entités externes : symboles pouvant être définis dans un autre fichier, utilisable dans un document XML ou dans la DTD elle-même. Il y a plusieurs types d'entités externes (car plusieurs types d'entités !)

# Les entités

De plus les entités peuvent être de 2 types :

- Les entités analysables, internes ou externes
- Les entités non analysables toujours externes. Elles correspondent généralement à des ressources externes comme des fichiers binaires d'images. On ne présenta pas ces notions dans ce cours, au même titre que les *notations* qui sont des types d'attributs assez rares permettant de définir des applications externes pour le traitement de données (qui peuvent être justement des entités non analysables!).

# Exemples d'entités

```
<!-- entite externe pour importer les entites -->
<!--  representant les caracteres accentues  -->
<!ENTITY % HTMLlat1 PUBLIC
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
%HTMLlat1; <!-- c'est comme un import -->
<!-- entite parametrique -->
<!ENTITY % elt "(#PCDATA|elt1)*" >
<!ELEMENT racine %elt;>
<!ELEMENT elt1 (#PCDATA)>
<!--entites interne -->
<!ENTITY euro "&#8364;">
<!ENTITY LILLE1 "Universit&eacute; de Lille 1">
<!-- l'utilisation du &eacute; est possible parce que -->
<!-- c'est une entité externe importée à l'aide de %HTMLlat1 -->
```

# Exemple de document valide pour la DTD précédente

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE racine SYSTEM "./entites.dtd">
<racine>
  blabla
  <elt1>
    Universit&eacute; : &LILLE1;
  </elt1>
  <elt1>
    10000 &euro;
  </elt1>
  c'est fini !
</racine>
```