PXML Master MIAGE M1 2011-2012

Cours 2 : Typer les données XML avec des DTD

Pourquoi définir des DTD ?

c.f. l'exercice de la semaine dernière sur le fichier `maisons.xml`

DTD ?

DTD

= grammaire pour la structure des documents

= un ensemble de règles,

chacune d'entre-elles décrivant le contenu autorisé d'un élément ou l'ensemble des attributs existant pour un élément.

Comment lier une DTD à un document XML

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

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

DTD externe

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

DTD externe

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

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>

...
```

DTD interne

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

DTD mixte

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

Validation d'un fichier XML

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*, *Editix*, *Eclipse*, . . .)
- par programme en utilisant les bibliothèques de traitement de XML disponibles dans beaucoup de langages (*java*, *php* , *perl*, ...)

Le fichier `XMLParser.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 nouveauté
      //saxReader.setContentHandler(new MonHandlerAMoi());// si on veut
      saxReader.parse(args[0]);
      } catch (Exception t) { t.printStackTrace();
   }
}
```

Structure d'une DTD

Une DTD contient :

- des déclarations d'éléments,
- des déclarations d'attributs,
- des déclarations d'entités,
- des commentaires

<!-- comme dans les documents XML -->

Déclaration d'élément

```
<!ELEMENT nom modèle>
```

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le modèle de contenu de l'élément.
  - vide l'élément n'a pas de contenu (mais peut avoir des attributs)
  - libre le contenu de l'élément est un contenu quelconque bien formé
  - données l'élément contient du texte
  - éléments l'élément est composé d'autre éléments (ses fils)
  - mixte l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

```
<!ELEMENT nom EMPTY>
```

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le modèle de contenu de l'élément.
  - vide l'élément n'a pas de contenu (mais peut avoir des attributs)
  - libre le contenu de l'élément est un contenu quelconque bien formé
  - données l'élément contient du texte
  - éléments l'élément est composé d'autre éléments (ses fils)
  - mixte l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

```
<!ELEMENT nom ANY>
```

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le modèle de contenu de l'élément.
  - vide l'élément n'a pas de contenu (mais peut avoir des attributs)
  - libre le contenu de l'élément est un contenu quelconque bien formé
  - données l'élément contient du texte
  - éléments l'élément est composé d'autre éléments (ses fils)
  - mixte l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

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

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le modèle de contenu de l'élément.
  - vide l'élément n'a pas de contenu (mais peut avoir des attributs)
  - libre le contenu de l'élément est un contenu quelconque bien formé
  - données l'élément contient du texte
  - éléments l'élément est composé d'autre éléments (ses fils)
  - mixte l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

```
<!ELEMENT nom modèle>
```

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le modèle de contenu de l'élément.
  - vide l'élément n'a pas de contenu (mais peut avoir des attributs)
  - libre le contenu de l'élément est un contenu quelconque bien formé
  - données l'élément contient du texte
  - éléments l'élément est composé d'autre éléments (ses fils)
  - mixte l'élément contient un mélange de texte et de sous-éléments

Modèle de contenu d'élément

On définit le contenu à l'aide d'une expression régulière de sous-éléments :

- séquence

```
<!ELEMENT chapitre (titre,intro,section)>
```

- choix

```
<!ELEMENT chapitre (titre,intro,(section|sections))>
```

- indicateurs d'occurrence `*` (0-n) `+` (1-n) `?` (0-1)

```
<!ELEMENT chapitre (titre,intro?,section+)>
<!ELEMENT section (titre-section,texte-section)>
<!ELEMENT texte-section (p|f)*>
```

Modèle de contenu d'élément

La syntaxe précise des expressions régulières de sous-éléments est :

- `cp ::= ( Name | choice | seq) ('?' | '*' | '+')?`
- `seq ::= '(' cp ( ',' cp )* ')'`
- `choice ::= '(' cp ( '|' cp )+ ')'`

Contenu mixte

Une seule façon de mélanger texte `#PCDATA` et des sous-éléments est acceptée : `#PCDATA` doit être le premier membre d'un choix placé sous une étoile.

```
<!ELEMENT p (#PCDATA | em | exposant | indice | renvoi ) *>
```

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 données</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
      données
    </prerequis>
  </stage>
</catalogue>
```

Exemple

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

Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de données</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
      données
    </prerequis>
  </stage>
</catalogue>
```

Exemple

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

Exemple

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

La mystérieuse propriété UPA

Unique Particle Attribution

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.

La mystérieuse propriété UPA

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.

La mystérieuse propriété UPA

Définition du W3C

Un modèle de contenu doit être formé de telle sorte que lors de la validation d'un élément séquence élément d'information, les composants des particules contenues directement, indirectement ou implicitement y avec pour tenter de valider chaque élément de la séquence à son tour peut être déterminée de façon unique sans examiner le contenu ou les attributs de cet élément, et sans aucune information sur la articles dans le reste de la séquence.

La mystérieuse propriété UPA

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)>
```

La mystérieuse propriété UPA

Une vraie définition

Une expression régulière de sous-éléments satisfait la propriété UPA si et seulement si son automate de Glushkov est déterministe.

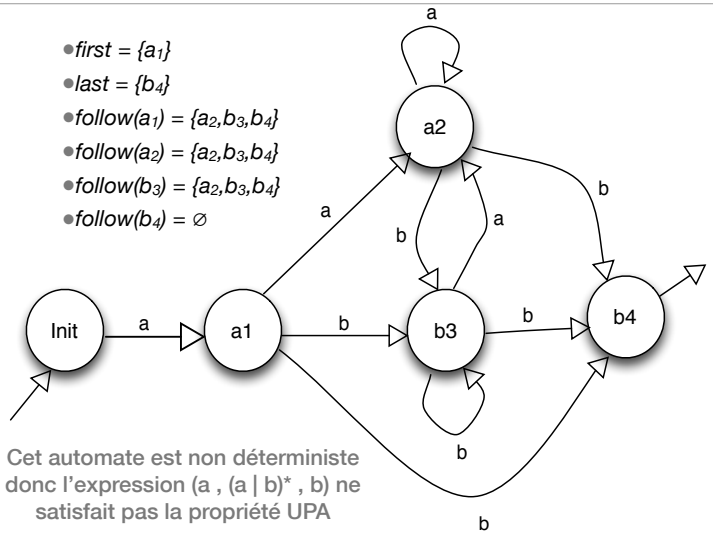
Automate de Glushkov (rappel ?)

Construction de cet automate pour `( a , (a | b )* , b)`

1. distinguer dans l'expression toutes les occurrences des mêmes noms d'élément : `(a1 , (a2 | b3 )* , b4 )`.
2. Calculer les ensembles
   - first = ensemble des lettres qui peuvent apparaître au début des mots qui matchent l'expression *first = {a1}*
   - last = ensemble des lettres qui peuvent apparaître à la fin des mots qui matchent l'expression *last = {b4}*
   - pour chaque lettre *x* calculer son ensemble follow qui contient les lettres qui peuvent apparaître immédiatement après *x* dans au moins un des mots qui matchent l'expression *follow(a1) = {a2,b3,b4}*, *follow(a2) = {a2,b3,b4}* , *follow(b3) = {a2,b3,b4}*, *follow(b4) = ∅*
   - c'est tout, on en déduit alors directement l'automate.

`(a1 , (a2 | b3 )* , b4 )`

- *first = {a1}*
- *last = {b4}*
- *follow(a1) = {a2,b3,b4}*
- *follow(a2) = {a2,b3,b4}*
- *follow(b3) = {a2,b3,b4}*
- *follow(b4) = ∅*



Cet automate est non déterministe donc l'expression (a , (a | b)* , b) ne satisfait pas la propriété UPA

Unique Particle Attribution

En définitive

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

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

Déclarations d'attributs

```
<!ATTLIST element nom-attribut1 type1 defaut1
                  nom-attribut2 type2 defaut2
                                      ...>
```

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.

Déclarations d'attributs

```
<!ATTLIST element nom-attribut1 type1 defaut1
                  nom-attribut2 type2 defaut2
                                      ...>
```

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 l'attribut est optionnel,
- `#FIXED valeur` indique que l'attribut prend toujours la même valeur, dans toute instance de l'élément si elle existe.

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 (donc pas un nombre).
- La valeur de défaut pour un attribut `ID` est obligatoirement `#REQUIRED` ou `#IMPLIED`
- Une valeur utilisée dans un attribut `IDREF` ou `IDREFS` doit obligatoirement correspondre à celle d'un attribut `ID`.

Exemples `ID`, `IDREF`, `IDREFS`

```
<!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>
```

document.dtd

Exemples `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>
```

OK

Exemples `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>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```xml
<?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>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-3">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```xml
<?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>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```
<!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>
```

document.dtd

Exemples ID, IDREF, IDREFS

```xml
<?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>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

OK

Déclarations d'entités

- Les entités internes : «macros exportées» qui sont utilisées dans le document XML validé par la DTD.

  `<!ENTITY euro "&#8364;">`, utilisation `&euro;`

- Les entités paramétriques : «macros non exportées» qui sont utilisées ailleurs dans la DTD.

  `<!ENTITY % editeur "O'Reilly">`, utilisation `%editeur;`

- Les entités externes : «macros importées» définies dans un autre document, utilisables dans la DTD elle-même ou dans tout document XML valide pour la DTD.

```
<!ENTITY % HTMLlat1 PUBLIC
     "-//W3C//ENTITIES Latin 1 for XHTML//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
```

Exemple 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 cette DTD

```
<?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>
```