XML – DTD

(Document Type Definition)

Aziz Bouras

- DTD obligatoire ou non ?
- Déclaration de la DTD
- DTD interne/externe
- DTD public/system
- Déclaration d'éléments
- Exemple DTD interne/externe
- Déclaration d'attributs
- Types de données
- Exemple valide/non valide

DTD obligatoire ou non ?

- La DTD n’est pas obligatoire
- Elle peut être interne ou externe
- il s’agit d’une grammaire qui definit:
  - les tags possibles et leurs attributs
  - quels tags sont autorisés à l’intérieur d’autres tags (leur imbrication)
  - quels tags et attributs sont à option et quels sont obligatoires
- Chaque "tag" XML est définit comme un élément dans le DTD
  - Exemple illustratif: <!ELEMENT title (#PCDATA)>

Déclaration de la DTD dans le fichier XML

- Il existe 4 façons d'utiliser une DTD :
  - On ne déclare pas de DTD (dans ce cas le fichier est juste "bien formé"
  - On déclare la DTD et on y ajoute les défintions dans le fichier (DTD interne)
  - On déclare la DTD en tant que DTD "privée", la DTD se trouve quelque part dans votre système ou sur Internet
  - On déclare une DTD "publique", c.à.d. on utilise un nom officiel pour la DTD. Cela présuppose que votre éditeur et votre client connaissent cette DTD.

- Lieu de la déclaration
  - La DTD est déclarée entre la déclaration de XML et le document lui-même
  - La déclaration de XML et celle de la DTD font partie du prologue (qui peut contenir d'autres éléments comme les processing instructions)

Exemples

- **1. Hello XML sans DTD**

```
<?xml version="1.0" standalone="yes"?>
<hello> Hello XML et hello cher lecteur ! </hello>
```

- **2. Hello XML avec DTD interne**

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE hello [
<!ELEMENT hello (#PCDATA)>
]>
<hello> Hello XML et hello chère lectrice ! </hello>
```

- **3. Hello XML avec DTD externe**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE hello SYSTEM "hello.dtd">
<hello> Hello XèMèLè et hello cher lectrice ! </hello>
```

- **4. Un fichier RSS (DTD externe public)**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS
  0.91//EN"
"http://my.netscape.com/publish/formats/rss-0.91.dtd">
<rss version="0.91">
<channel> ..... </channel>
</rss>
```

Déclaration de la DTD

- Chaque déclaration du DTD commence par:
  <!DOCTYPE et fini par: >

- La racine de l'arbre XML (ici: <hello>) doit être indiquée après <!DOCTYPE

- Syntaxe pour définir une DTD interne (seulement !)
  - La DTD sera insérée entre [...]

    <!DOCTYPE hello **[**

    <!ELEMENT hello (#PCDATA)>

    ]>
- Syntaxe pour définir une DTD privée externe:
  - La DTD est dans l'URL indiquée après le mot clef "SYSTEM".
    <!DOCTYPE hello SYSTEM "hello.dtd">

DTD interne

Le sous-ensemble interne de la DTD est placé dans le DOCTYPE après la déclaration XML.

- L'attribut *standalone* prendra dans la déclaration XML la valeur "yes" s'il n'y a pas à rechercher de DTD externe, "no" sinon.

- <?xml version="1.0" standalone="yes"> <!DOCTYPE racine [<!-- sous-ensemble de DTD interne -->]> <racine> ... </racine>

DTD externe

- Le sous-ensemble externe est placé dans un fichier séparé.
- L'appel à ce fichier qui doit avoir le même nom que la racine du document (élément qui englobe tous les autres) est effectué dans le DOCTYPE après la déclaration XML.
- L'attribut standalone prendra dans la déclaration XML la valeur "no".
- <!DOCTYPE racine (SYSTEM "url") | (PUBLIC "identifiant_public" "url"?)>

PUBLIC

- PUBLIC : utilisé lorsque la DTD est une norme ou qu'elle est enregistrée sous forme de norme ISO par l'auteur.

- identifiant_public contient les caractéristiques :
  type_enregistrement // proprietaire // DTD description // langue

  - type_enregistrement :
    - un signe + si c'est selon la norme ISO 9070,
    - un signe - sinon ;
  - proprietaire : nom du propriétaire (entreprise ou personne) ;
  - DTD description : une description textuelle, espaces autorisés ;
  - langue : un code de langue ISO 639.

- L'adresse du fichier décrivant la DTD n'est pas obligatoire, le processeur XML peut utiliser les informations de l'identifiant public pour essayer de générer une adresse. Il faut noter cependant qu'il n'est pas toujours possible de trouver l'adresse à partir de l'identifiant, il est donc conseillé de faire suivre l'identifiant par l'adresse du fichier.

- Voici la déclaration de type de document pour un document html qui utilise la DTD XHTML1.0 :

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

SYSTEM

- SYSTEM est utilisé pour donner l'adresse du fichier qui contient la DTD dans le cas où la DTD n'est pas publique.

- <!DOCTYPE racine SYSTEM "http://www.serveur.fr/racine.dtd">

Déclaration d'éléments

- La déclaration d'éléments est de la forme :
  <!ELEMENT nom_element modele_de_contenu>

- **Elément vide**
- Ne contient aucun texte, aucun autre élément
  <!ELEMENT nom_element EMPTY>

- **Elément non vide**
  - Un élément non vide est formé d'une balise ouvrante, d'un contenu et d'une balise fermante.
  - Pour décrire ce contenu dans la DTD on utilise un modèle de contenu dans la déclaration d'éléments.

- **Données**
  Si l'élément contient uniquement des données (autrement dit, s'il n'y a aucun autre élément inclus dans cet élément) on utilise le type #PCDATA qui signifie *Parsed Character DATA* i.e. données caractères analysées :
  <!ELEMENT nom_element (#PCDATA)>

- Exemple, dans la DTD de XHTML 1.0 l'élément title qui permet de donner le titre de la fenêtre est déclaré : <!ELEMENT title (#PCDATA)>

Déclaration d'éléments

- **conteneur d'éléments**
  - Le modèle de contenu définit les éléments pouvant être inclus dans un élément donné et spécifie également les éléments devant apparaître obligatoirement, leur ordre et leur fréquence.
  - Exemple, dans la DTD de XHTML 1.0 l'élément racine html qui contient les éléments head et body est déclaré <!ELEMENT html (head, body)>.

- Les indicateurs d'occurrences sont les suivants :
  - ? = 0 ou 1 fois
  - * = 0 ou n fois
  - + = au moins une fois

  - Exemple, dans la DTD de XHTML 1.0 l'élément ul est déclaré : <!ELEMENT ul li+>

- Pour indiquer des choix, il est possible d'utiliser le |.
  - Dans la DTD XHTML1.0 l'élément dl est déclaré : <!ELEMENT dl (dt|dd)+>

- **mixte**
  - Si l'élément a un contenu mixte (données + éléments) il est possible d'utiliser PCDATA et les éléments imbriqués.
  - Exemple, dans la DTD de XHTML 1.0 l'élément object est déclaré : <!ELEMENT object (#PCDATA | param | %block; | form | %inline; | %misc;)*>

- **Élément libre**

Élément qui peut contenir tout élément déclaré dans la DTD et du texte.
<!ELEMENT nom_element ANY> **Exemple**

Exemple DTD interne

```
<?xml version="1.0"?>
<!DOCTYPE bibliotheque [ <!ELEMENT bibliotheque (livre+)>
<!ELEMENT livre (titre, auteur, ref)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT ref (#PCDATA)> ] >

<bibliotheque>
<livre>
    <titre>N ou M</titre>
    <auteur>Agatha Christie</auteur>
    <ref>Policier-C-15</ref>
</livre>
<livre>
    <titre>Le chien des Baskerville</titre>
    <auteur>Sir Arthur Conan Doyle</auteur>
    <ref>Policier-D-3</ref>
</livre>
<livre>
    <titre>Dune</titre>
    <auteur>Franck Heckbert</auteur>
    <ref>Fiction-H-1</ref>
</livre>
</bibliotheque>
```

DTD Externe

- Fichier externe biblio.dtd :

```
<!ELEMENT bibliotheque (livre+)>
<!ELEMENT livre (titre, auteur, ref)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT ref (#PCDATA)>
```

- Document XML :

```
<?xml version="1.0"?>
<!DOCTYPE bibliotheque SYSTEM "biblio.dtd">
<bibliotheque>
<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>
<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>
<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>
</bibliotheque>
```

Déclarations d'attributs

- Le rôle de la déclaration d'attributs est de donner pour un élément cible :
  - les noms d'attributs permis ;
  - le type de chaque attribut ;
  - la valeur par défaut de l'attribut.
  - <!ATTLIST element_cible nom_attribut type_attribut valeur_par_défaut>

- **Valeur par défaut**
  - Valeur prise par l'attribut si aucune valeur n'est précisée dans l'élément.
  - Cette valeur doit être du même type que celui donné à l'attribut.
  - S'il n'y a pas matière à définir une valeur par défaut on peut remplacer par un mot-clé :

- #REQUIRED = obligatoire, i.e. la valeur d'attribut doit être spécifiée lorsque l'élément est utilisé dans le document (par exemple en HTML il est obligatoire de donner l'attribut src avec l'élément img).
-
- #IMPLIED = facultatif, i.e. la valeur d'attribut peut rester non spécifiée

- #FIXED 'val' = fixée à 'val' i.e. la valeur de l'attribut est fixe et ne peut pas être modifiée par l'utilisateur

Déclaration d'attributs (2)

- Exemple, dans la DTD XHML1.0 on a pour l'élément ***pre*** l'attribut xml:space qui est fixé à ***preserve*** pour signifier que les espaces blancs à l'intérieur de pre sont significatifs et ne doivent pas être modifiés par le processeur XML :

```
<!ATTLIST pre
%attrs;
xml:space (preserve) #FIXED 'preserve'
>
```

- Dans cette même DTD, l'attribut ***method*** de l'élément ***form*** qui ne peut prendre comme valeur que *get* ou *post* prend par défaut la valeur get, d'autre part, le type mime d'encodage prend par défaut la valeur application/x-www-form-urlencoded :

```
<!ATTLIST form
%attrs;
action %URI;                    #REQUIRED
method (get|post)               "get"
enctype %ContentType;           "application/x-www-form-urlencoded"
onsubmit %Script;               #IMPLIED
onreset %Script;                #IMPLIED
accept %ContentTypes;           #IMPLIED
accept-charset %Charsets;       #IMPLIED
>
```

Types de données

- **chaînes de caractères**

```
<!ATTLIST element_cible nom_attribut CDATA valeur_par_défaut>
```

  - Pour XHTML1.0 les attributs http-equiv, name, content, scheme de l'élément meta sont des chaînes de caractères :

```
<!ATTLIST meta
      %i18n;
      http-equiv CDATA #IMPLIED
      name CDATA #IMPLIED
      content CDATA #REQUIRED
      scheme CDATA #IMPLIED
>
```

Types de données (2)

- **Énumérations**

  <!ATTLIST element_cible nom_attribut (val1 | val2 | ... | valN) "valdefaut">

  - L'attribut prend une valeur dans la liste des valeurs possibles définies dans la déclaration d'attribut (chaque valeur doit être un atome nominal ou NMTOKEN valide).

  <!ATTLIST livre type (policier | fiction | aventure) "policier">

- **Identificateurs**
  - <!ATTLIST element_cible nom_attribut ID>
  - <!ATTLIST element_cible nom_attribut IDREF>
  - <!ATTLIST element_cible nom_attribut IDREFS>

  - Un attribut de type ID représente un identificateur unique d'élément, i.e. un attribut dont la valeur distingue l'élément de tous les autres dans le document XML courant (ex. l'attribut id en HTML).

- Les types IDREF et IDREFS sont des références/liste de références séparées par des espaces, à un identifiant unique précédemment déclaré (par exemple, l'attribut header de l'élément td en XHML est de type IDREFS pour indiquer à quelles cellules de l'en-tête du tableau se rapporte la cellule courante).

Types de données (3)

- **Entités**
- **Tokens, nmtokens**
- **Notations**

Exemple valide/non valide ??

- DTD

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name,email*)>
<!ELEMENT name (family,given)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

- Document XML valide

```
<addressBook>
<person>
<name> <family>Wallace</family> <given>Bob</given> </name>
<email>bwallace@megacorp.com</email>
</person>
<person>
<name> <family>Tuttle</family> <given>Claire</given> </name>
<email>ctuttle@megacorp.com</email>
</person>
</addressBook>
```

Exemple valide/non valide ?? (2)

- DTD

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name,email*)>
<!ELEMENT name (family,given)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

- Document XML non valide !!

```
<addressBook>
<address>Derrière le Salève</address>
<person>
<name>
<family>Schneider</family> <firstName>Nina</firstName>
</name>
<email>nina@dks.com</email>
</person>
<name>
<family> Muller </family> </name>
</addressBook>
```