Résumé DTD

DTD, Document Type Definition

1-Utilité de la DTD

La DTD permet la définition de la STRUCTURE d'un type de documents. C'est une forme de grammaire qui va servir à spécifier les noms des éléments et la structure d'un document exprimé dans un langage à balise (SMGL, HTML, XML).
Chaque dialecte de XML (SMIL, SVF, XHTML, etc.) doit donc faire référence à une description de document type (DTD) pour permettre la validation de sa structure.

2-Syntaxe DTD

A- Définir des éléments complexes (= éléments possédant des éléments fils)

La description d'un élément complexe nomme cet élément et énumère la liste des éléments qui le composent (ses éléments fils), dans l'ordre où ils doivent apparaître :

```
<!ELEMENT element (composant1, composant2, composant3,... ) >
```

Ici 'composant1', 'composant2' et 'composant3' doivent être présent 1 et 1 seule fois : ils sont uniques et obligatoires dans la composition de 'element' et doivent être présents dans cet ordre.

Associé aux éléments qui le composent, on peut associer le nombre de fois que l'on va trouver chacun de ces éléments grâce à un code :

```
<!ELEMENT element (composant1?, composant2*, composant+,... ) >
```

où :

- composant1**?** : on trouvera composant1 : **0 ou 1 fois**
- composant2***** : on trouvera composant2 : **0 à n fois**
- composant3**+** : on trouvera composant3 : **1 à n fois** (au moins une fois)

On peut également décrire la présence d'éléments alternatifs (soit un élément soit un autre) :

```
<!ELEMENT element (adresse, (departement | etat | ...), pays,... ) >
```

où :

- adresse sera présent 1 fois et une seule (obligatoire)
- ensuite on trouvera soit un élément 'departement' soit un élément 'etat', soit ...
- pays sera présent 1 fois et une seule (obligatoire)

B- Définir des éléments simples (= pas d'élément fils)

La description d'un élément simple nomme cet élément et caractérise son contenu. #PCDATA signifie que le contenu représente une chaîne de caractères quelconque (Parsed Caracter Data) qui sera analysée par le parser :

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

Un élément peut aussi contenir des caractères qu'on ne souhaite pas 'parser' :

```
<!ELEMENT nom_element (#CDATA)>
```

Un élément peut aussi contenir n'importe quelle suite de caractères (chaînes de caractères et autres éléments dont on ne veut pas tenir compte) :

```
<!ELEMENT nom_element ANY >
```

On peut déclarer certains éléments comme étant vides (en HTML, par exemple : br, hr sont des éléments vides, sans contenu) :

```
<!ELEMENT nom_element  EMPTY >
```

Ces éléments peuvent néanmoins contenir des attributs.

C- Définir la liste des attributs associés à un élément

La définition des attributs identifie l'élément auquel ils s'appliquent, puis énumère pour chaque attribut, son nom, son contenu et éventuellement des contraintes qu ilui sont affectées :

```
<!ATTLIST element
     attribut1 contenu1 contrainte1
     ...
     attributN contenuN contrainteN
      >
```

où :

- 'element' : représente le nom de l'élément concerné
- 'attribut1', 'attributN' : les noms des attributs
- 'contenu1', 'contenuN' : le type de contenu :
    - **CDATA** : chaîne de caractère quelconque,
    - **NMTOKEN** : chaîne de caractère sans espace, ni caractères spéciaux (comme un nom d'élément XML)
    - **ID** : attribut dont la valeur sera un identifiant unique
    - **IDREF** : attribut dont la valeur sera l'identifiant d'un autre élément
- 'contrainte1','contrainteN' : optionnel, détermine le type de contrainte associée à cet attribut : soit
    - **#DEFAULT valeur** : définit la valeur par défaut
    - **#REQUIRED** : valeur d'attribut obligatoire dans le document XML,
    - **#IMPLIED** : valeur d'attribut facultative dans le document XML
    - **#FIXED** "valeur" : valeur fixe
    - **(valeur1 | valeur2 | ...) « valeur1 » :** valeur parmi une liste de valeur autorisées
    - "valeur par défaut" : une valeur par défaut si l'attribut n'est pas présent

D- Définir des entités

L'entité définir un contenu qui pourra être inséré dans le document XML :

```
<!ENTITY nom_entité "contenu qui va remplacer nom_entité dans le
document XML " >
```

Pour remplacer une entité par son contenu dans un document XML, on devra l'encadrer par les caractères **&** et **;**. Par exemple :

```
<!ENTITY eau "Bouteille d'eau minérale" >
```

Le contenu de l'entité '**eau**' sera inséré (autant de fois qu'on le souhaite) dans la document XML grâce à **&eau;**.

Quelques entités sont prédéfinies :

<	**<**	tt = less than = plus petit que, inférieur
>	**>**	gt = greater than= plus grand que, supérieur
&	**&**	ampersand = perluète
"	**«**	quotation mark = guillemet

'	‘	apostrophe = apostrophe

3-Déclaration de DTD dans un document XML

Un document XML va soit inclure la DTD, soit faire le lien avec une description externe de la DTD.

A- La DTD au sein du document XML

La **DTD** peut être incluse directement dans le document XML : l’avantage est que l’on n’a pas besoin d’aller voir dans un autre fichier pour obtenir la description.
Le document XML est donc autonome (*anglais : standalone*) :

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE element_racine [
  <!ELEMENT element_racine (cd*)>
...
]>
<element_racine>
…le contenu XML de l’élément racine
</element_racine>
```

B-La DTD dans un fichier séparé

La **DTD** peut être définie dans un fichier séparé : l’avantage est que l’on peut partager un même fichier DTD entre plusieurs documents XML. Un lien est réalisé dans le document XML vers le fichier contenant la DTD.
Le document XML n’est donc plus autonome :

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE element_racine SYSTEM "ma_dtd.dtd">
<element_racine>
…
</element_racine>
```

Le document DTD ‘ma_dtd.dtd’ contient alors :

```
<!ELEMENT element_racine (cd*)>
...
```

6-Validation d’un document XML grâce à sa DTD

Les navigateurs (et les parsers qui y sont intégrés), ne valident pas toujours les documents XML. Ils s’assurent seulement que le document chargé soit ‘bien formé’.

Pour valider un document XML, il faudra faire appel à des applications particulières. Ces services de validation sont offerts, par exemple, au travers d’API *(Application Programme Interface : services offerts pour les applications développées)* mises à disposition dans des framework associés à des langages comme Java ou C# *(le framework, correspond à*

l'ensemble des classes et outils fournis autour de ces langages pour developper des applications).

7-Evolution : de la DTD au XSD (XML Schema Definition)

La syntaxe DTD est simple et rapide à mettre en œuvre. Elle est cependant exprimée dans un langage particulier et des extensions nécessiteraient de définir de nouveaux mots-clefs de syntaxe.
La DTD est encore très utilisée mais elle a tendance à être remplacée par la syntaxe XSD (XML Schema Definition), c'est-à-dire la définition de schéma XML grâce à une syntaxe XML.

La syntaxe XSD, grâce à son expression dans le langage XML, permet l'ajout d'attributs supplémentaires pour contrôler plus précisément les données à transporter.

Exemple de syntaxe DTD :

```
<!ELEMENT compilation (cd*)>
<!ELEMENT cd (titre, auteur*)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ATTLIST cd duree NMTOKEN>
```

Exemple de syntaxe XML Schéma :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

<xsd:element name="titre" type="xsd:string" />
<xsd:element name="auteur" type="xsd:string" />

<xsd:attribute name="duree" type="xsd:integer" />

<xsd:element name="cd">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element ref="titre" />
               <xsd:element ref="auteur" minOccurs="0" maxOccurs="9">
          </xsd:sequence>
          <xsd:attribute ref="duree" />
     </xsd:complexType>
</xsd:element>

<xsd:element name="compilation">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element ref="cd"
                         minOccurs="1" maxOccurs="unbounded">
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>

</xsd:schema>
```

Exemple de document XML (liens vers DTD ou XSD):

Résumé DTD

```
<?xml version="1.0" encoding="iso-8859-1" ?>
si DTD : <!DOCTYPE element_racine SYSTEM "ma_dtd.dtd">
si XSD : <compilation xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance" xsi:noNamespaceSchemaLocation="file:mon_xsd.xsd"/>
Si DTD : <compilation>
<cd duree="4"><titre>Ecoutez moi</titre><auteur>prof</auteur></cd>
</compilation>
```