XML et DTD: rappels

Cours « Document et Web Sémantique »

Nicolas Malandain

INSA ROUEN NORMANDIE

Plan

INSA ROUEN NORMANDIE

Quelques concepts importants

- Méta langage à balises
- Données et balises sont des chaînes de caractères $\Rightarrow$ portable
- La syntaxe est stricte
- Langage à balises structurel et sémantique
- Les marqueurs définissent la sémantique du document
- Ce n'est pas un langage de présentation (HTML)

INSA ROUEN NORMANDIE

Évolution de XML

- 1970 ⇒ 1986 : **S**tandard **G**eneralized **M**arkup **L**anguage
  Langage sémantique et structurel à balises pour documents textuels
- Une application SGML : **H**yper**T**ext **M**arkup **L**anguage
  - restreint à la présentation des pages Web
  - ne peut servir à l'échange de données entre bases hétérogènes
- SGML résoud de nombreux problèmes
  Mais il est trop complexe (les spécifications = 150 pages techniques)
- XML est une application SGML
  - conserve une grande partie de la puissance de SGML
  - élimine les caractéristiques redondantes, compliquées à implanter et qui n'ont montré aucun intérêt après plus de 20 ans

INSA ROUEN NORMANDIE

Une famille de technologies XML 1 / 2

XML 1.0 est un “SGML simplifié”, diverses applications XML ont suivi :

- e**X**tensible **S**tylesheet **L**anguage
  - **X**SLTransformation : transformation d'un document XML en un autre
  - **X**SL-Formatting **O**bject : décrit la composition de pages destinées à l'impression (rival de Postscript)
- **C**ascading **S**tyle**S**heet (non XML) : présentation de documents XML
- e**X**tensible **L**inking **L**anguage
  - XLink : décrit la relation entre documents XML
  - XPointer : identifie une partie de document XML
- XPath (non XML) : cibler un ou des éléments d'un document XML

INSA ROUEN NORMANDIE

Une famille de technologies XML 2 / 2

- XML Schemas : description structurelle et sémantique à laquelle un document XML doit se conformer (*DTD++*)
- **S**imple **A**PI for **X**ML : API que tout parseur XML doit respecter
- **D**ocument **O**bject **M**odel : API permettant de manipuler un document XML
- XML Query Language, XHTML, SMIL, SVG, ...

INSA ROUEN NORMANDIE

HTML / XML

```
<p> Nicolas Malandain </p>
<address>
INSA de Rouen<br>
BP08<br>
Avenue de l'Université<br>
76801 Saint Étienne du Rouvray
</address>
```

- balises et sémantiques associées sont prédéfinies
- mélange de structurations logique et physique
- perte du sens

```
<enseignant corps="maître de conférences">
<prenom>Nicolas</prenom><nom>Malandain </nom>
<adresse>
<structure>INSA de Rouen</structure>
<bp>BP08</bp>
<rue>Avenue de l'Université</rue>
<cp>76801</cp>
<ville>Saint Étienne du Rouvray</ville>
</adresse>
</enseignant>
```

- extensibilité du langage
- structuration logique
- représentation physique déléguée (CSS, XSL)
- modularité et réutilisation des structures
- facilite l'accès à des sources de données hétérogènes

INSA ROUEN NORMANDIE

Document bien formé et valide

XML distingue 2 classes de documents :

1. les documents **bien formés** obéissent à la syntaxe du langage XML,
2. les documents **valides** sont bien formés et obéissent à une structure type définie dans une DTD.

Tout document valide peut être distribué sans sa DTD (ou référence à sa DTD), il apparaîtra alors comme bien formé aux utilisateurs (humains ou électroniques : navigateur).

INSA ROUEN NORMANDIE

Structure d'un document XML

- Un prologue (facultatif mais fortement conseillé)
  - une déclaration XML
  - des instructions de traitements (utilisées par les moteurs, les navigateurs)
  - une déclaration de type de document
- Un arbre d'éléments
- Des commentaires (facultatifs)

INSA ROUEN NORMANDIE

Exemple de document XML

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE article SYSTEM "article.dtd" [
  <!ENTITY auteur "Sacha Touille">
]>
<?xml-stylesheet type="text/xsl" href="http://www.gratouille.com/article.xsl" ?>
<article>
  <titre> Le monde de la chatouille </titre>
  <auteur> &auteur; </auteur>
  <texte>
  &auteur; a écrit de nombreux d'ouvrages sur le thème de ...
  <!-- texte à développer -->
  </texte>
</article>
```

Déclaration XML

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes" ?>
```

la version du langage XML utilisé dans le document (1.0)

- La version 1.1 (de 2004) ajoute peu de choses (par exemple : code retour à la ligne IBM)

le codage de caractères (charset) utilisé :
sont autorisés les jeux de caractères définis dans la norme ISO 10646, les parseurs doivent traiter au moins les codages UTF-8 ou UTF-16

- en l'absence de déclaration, ils s'attendent à de l'UTF
- si un autre codage est utilisé il faut le spécifier

l'existence de déclaration extérieure au document :

- si le document est autonome : "yes"
- si le document nécessite la récupération de données externes : "no"

ROUEN NORMANDIE

Cette déclaration est facultative mais fortement conseillée

Instructions de traitements

Instructions facultatives
Leur contenu est transmis à une application qui déclenchera des traitements

<?cible arg_1 arg_2 ... ?>

- `cible` : nom d'une application (`xml` est un mot réservé)
- arguments passés à l'application cible

Exemple

```
<?xml-stylesheet type="text/css"
href="style.css" ?>
```

Déclaration de type de documents

```
<!DOCTYPE elt-racine URI-DTD [ déclarations internes ]>
```

Déclaration :

- de l'élément racine de l'arbre d'éléments (obligatoire si DOCTYPE)
- de la structure à laquelle le document doit se conformer
- d'entités

Exemple

```
<!DOCTYPE lettre SYSTEM "lettre.dtd" [
  <!ENTITY destinataire "Ultasonnul Thérèse">
  <!ENTITY motif "Convocation">
]>
```

L'arbre d'éléments 1 / 2

Un élément est composé :

- d'une balise d'ouverture
- d'un contenu
- d'une balise de fermeture

Le nom d'élément (balise) :

- caractères alpha-numériques, souligné, tiret et point
- pas de caractères d'espacement ou de fin de ligne
- “:” est réservé pour les espaces de noms
- premier caractère alphanum ou “_”
- la casse est importante
- aucune balise ne peut commencer par “xml” (qq soit la casse)

INSA ROUEN NORMANDIE

L'arbre d'éléments 2 / 2

L'élément vide (sans contenu) :

```
<nomelt></nomelt>  ⟺  <nomelt/>
```

- Un document comporte toujours un arbre d'éléments
- Il y a un unique élément racine contenant les autres éléments
- Il n'y a pas de chevauchement d'éléments

Commentaires :

```
<!-- commentaire -->
```

- ne peut contenir la chaîne "--"
- inclusion de commentaires impossible

INSA ROUEN NORMANDIE

Le contenu d'un élément

Un élément peut contenir un mélange :

- d'éléments
- de données : tous les caractères exceptés `&` et `<`
- de références à des entités
- de sections littérales (`CDATA`)
- d'instructions de traitement

INSA ROUEN NORMANDIE

Attributs d'éléments

- les attributs sont placés dans les balises d'ouverture

  `<nomelt att1="val1" att2="val2" ...attn="valn">`

- les noms d'attributs obéissent aux mêmes règles que les noms d'éléments
- les valeurs sont encadrées par `" "` ou `' '`

  ce qui permet : `"aujourd'hui"` ou inversement

- les valeurs ne peuvent contenir les caractères `^, %, &`

Attributs d'éléments réservés

Affectation de propriétés particulières à des éléments

- `xml:lang` permet de spécifier la langue du contenu de l'élément

```
<remerciement xml:lang="fr"> merci </remerciement>
<remerciement xml:lang="en"> thank you </remerciement>
```

- `xml:space` spécifie si les espaces doivent être préservés dans le contenu de l'élément
- `xml:link` spécifie qu'un élément particulier est un lien
- `xml:attribute` gestion des attributs de XLink

INSA ROUEN NORMANDIE

Section littérale

Pas d'interprétation du contenu de la section littérale

Pratique pour mettre du XML dans du XML

```
<![CDATA[ contenu non interprété ]]>
```

Exemple

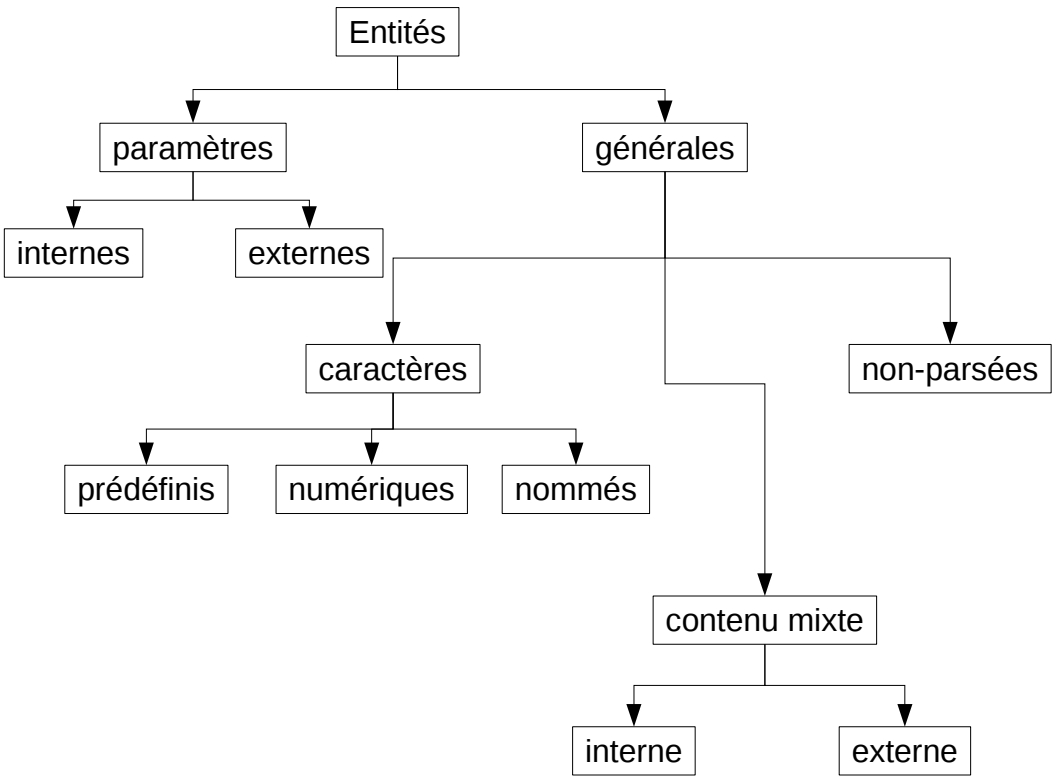
```
<exemple>
  Section littérale:
  <![CDATA[<titre> Un titre littéral </titre>]]>
</exemple>
```

Ce qui donne :

```
Section littérale: <titre> Un titre littéral </titre>
```

Différents types d'entités

Les entités sont comparables à des réserves d'informations



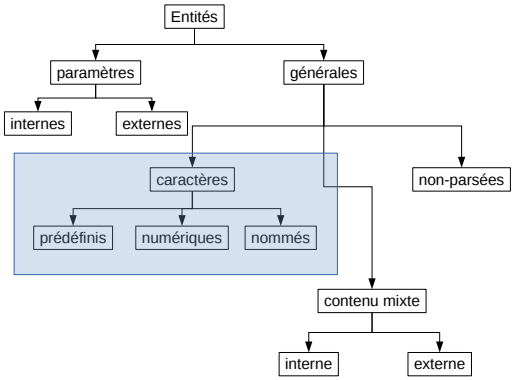

```
entite="valeur"
```

Utilisation :

- entité générale : `&entite;`
- entité paramètre : `%entite;`

Les entités paramètres sont déclarées et utilisées dans les DTD

Entités caractères



Entités ne contenant qu'un seul caractère

- prédéfinies : caractères interdits dans un document XML

`amp` : `&`	`gt` : `<`	`quot` : `"`
`apos` : `'`	`lt` : `>`	

- numériques : caractères saisis sous leur code unicode

ç : `&#231;`$_{(10)}$ ou `&#xE7;`$_{(16)}$

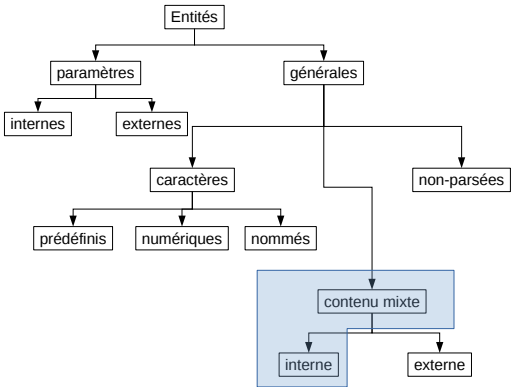
- nommées : noms associés à des caractères (déclaration obligatoire)

`&Pgr;` pour Π

ISO-8879 contient un ensemble standard d'entités caractères nommées pour les alphabets latin, grec, nordique, cyrillique, ...

Entités de contenu mixte internes

Entités dont la valeur est un mélange de texte et de balisage (=XML)
Texte de remplacement, utile pour la maintenance et éviter la répétition



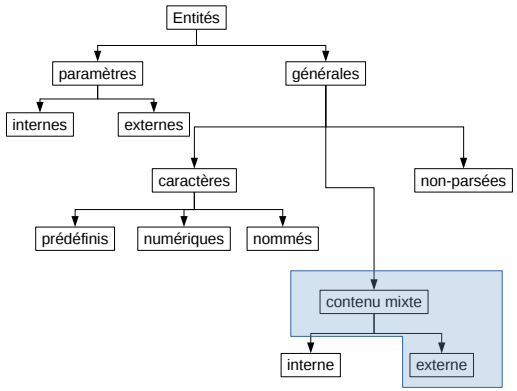

```
<!DOCTYPE eltracine [
  <!ENTITY entite "valeur">
]>
```

Exemple

```
<!ENTITY insa "<ecole>INSA de Rouen</ecole>">
```

Entités de contenu mixte externes

La valeur de l'entité est située à l'extérieur du document



- valeur très grande
- elle peut contenir une déclaration `<?xml ...?>`, dans ce cas seule la déclaration de codage `encoding` est obligatoire (`standalone` est inutile, `version` est optionnelle)
- segmentation du document en plusieurs

Adressage de la valeur par une URL

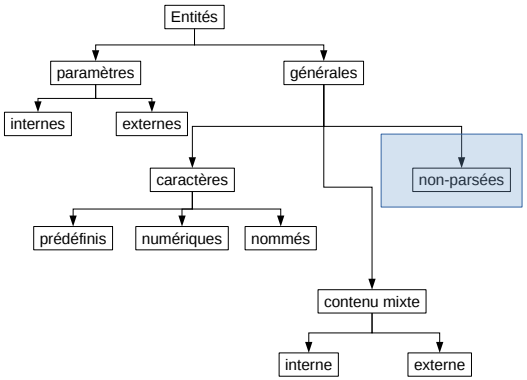
`<!ENTITY` *`entite`* `SYSTEM` *`URL`*`>`

INSA ROUEN NORMANDIE

Entités non XML (non parsées)

L'utilisation d'entité non XML nécessite deux déclarations :

- le format de l'entité (`NDATA`)
- l'application capable de traiter ce format (`NOTATION`)

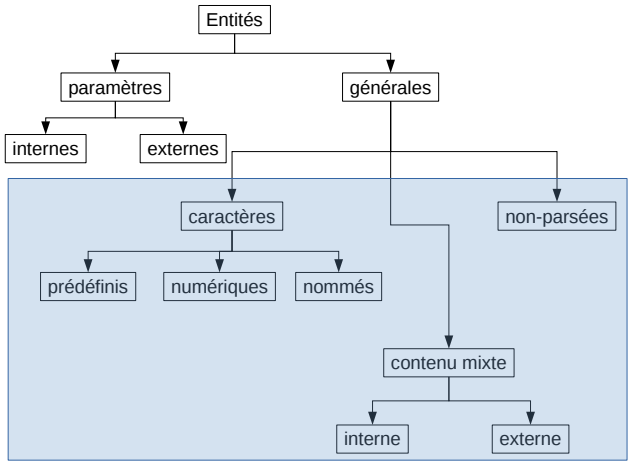


La référence à une entité non XML ne peut se faire que dans un attribut

Exemple

```
<!DOCTYPE visite [
  <!NOTATION vrml SYSTEM "/usr/local/bin/vrml">
  <!ENTITY appartement SYSTEM "./plan/appart.vrml'' NDATA vrml>
]>
<visite>
  <lieu3D plan="appartement"/>
</visite>
```

Récapitulatif de l'utilisation d'entités



Utilisation	Caractère	XML interne	XML externe	non XML externe
Référence dans le contenu d'un élément	Remplacement	Remplacement	Remplacement immédiat ou non (choix du développeur)	Interdit
Référence dans la valeur d'un attribut	Remplacement	Remplacement [1]	Interdit	Interdit
Nom symbolique comme valeur d'attribut [2]	Non reconnu	Interdit	Interdit	Message à l'application ou appel de l'application externe
Référence dans la valeur d'une autre entité	Remplacement	Transmis tel quel	Transmis tel quel	Interdit

1. Attention : les " ou ' de la valeur doivent être balancés
2. Sans utilisation du `&` et du **;**

INSA ROUEN NORMANDIE

Conseils pour l'écriture de documents XML 1 / 2

- choisir des noms d'éléments qui représentent leur rôle, il doivent être aussi explicites que possible (lisibles par l'homme).
- la position d'un élément à l'intérieur d'un autre est importante (l'ordre des éléments est préservé)
- donner du sens aux balises (ex : acronyme, mots étrangers, etc.)
- le balisage doit être indépendant de la réalisation physique du document (ex : pas de `<gras>`) ; préférer un balisage métatypographique : (ex : `<important>`, `<ligne>`, `<cellule>`)
- inclure dans le document des métadonnées descriptives afin de décrire le document (cf. RDF)
- l'indexation d'un document se fait sur le contenu des éléments, pas sur les valeurs des attributs

INSA ROUEN NORMANDIE

Conseils pour l'écriture de documents XML 2 / 2

- Utilisez un élément lorsque :
  - le contenu comporte plusieurs mots
  - l'ordre est important (il n'y a pas d'ordre sur les attributs)
  - l'information fait partie du contenu du document en opposition à un paramètre ajustant le comportement d'un élément. Si un processeur n'est pas capable de traiter un document XML, il affichera le contenu des éléments mais pas les attributs.
- Utilisez un attribut lorsque :
  - l'information modifie l'élément d'un point de vue du traitement exemple : `<liste type="numero"> ...</liste>`
  - vous souhaitez contrôler les valeurs
  - l'information est un identifiant unique ou une référence à l'identifiant d'un autre élément (cf. `ID et IDREF`)

INSA ROUEN NORMANDIE

Contenu d'une *Document Type Definition*

Une DTD contient :

- la liste des éléments autorisés
- pour chaque élément, le contenu autorisé
- pour chaque élément, la liste des attributs et valeurs autorisés
- les entités autorisés

La DTD ne donne pas d'information sur :

- l'élément racine du document (cf. `DOCTYPE`)
- le nombre d'occurences d'un élément dans le document
- le contenu textuel des éléments
- la syntaxe (type) du contenu d'un élément

INSA ROUEN NORMANDIE

Un exemple simple de DTD

Fichier personne.dtd

```
<?xml version="1.0" encoding="ISO-8859" ?>
<!ELEMENT personne (identite, profession*)>
<!ELEMENT identite (prenom, nom)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT profession (#PCDATA)>
```

Fichier dupond.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE personne SYSTEM "./personne.dtd" >
<personne>
  <identite>
    <prenom>Jean</prenom>
    <nom>Dupondavecund</nom>
  </identite>
  <profession>detective</profession>
  <profession>garde du corps</profession>
</personne>
```

Prologue d'une DTD

Une DTD n'a pas nécessairement de prologue

Le prologue est identique à un document XML, excepté `standalone` et `DOCTYPE` qui n'ont aucun sens dans une DTD.

Utilité : déclarer l'encodage utilisé dans la DTD

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Remarque : la déclaration d'encodage de la DTD n'est pas propagé au document XML

Déclaration d'éléments

Structure d'une déclaration d'élément

<!ELEMENT *nom_element* (*modele_contenu*) >

- *nom_element* sera le nom donné à la balise
- *modele_contenu* désigne les sous éléments autorisés, leur ordre, si du contenu textuel est autorisé, ...

INSA ROUEN NORMANDIE

Modèles de contenu 1 / 3

- Données textuelles parsés : `#PCDATA`

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

- Sous élément

```
<!ELEMENT element (souselement)>
<!ELEMENT fax (numero)>
```

- Séquence de sous éléments (ordonnés)

```
<!ELEMENT element (souselement1, souselement2, ...)>
<!ELEMENT identite (prenom, nom)>
```

INSA ROUEN NORMANDIE

Modèles de contenu 2 / 3

- Nombre d'occurence de sous éléments
  - `?` : l'élément ou groupe est optionnel
  - `+` : l'élément ou groupe apparaît au moins 1 fois
  - `*` : l'élément ou groupe apparaît de 0 à N fois

```
<!ELEMENT identite (nom, nomjeunefille?, prenom+, surnom*)>
```

- Choix exclusif entre plusieurs sous éléments possibles

```
<!ELEMENT element (souselement1 | souselement2 | ...)>
<!ELEMENT situation (celibataire | marie | divorce)>
```

- Groupe (parenthèses) : permet de combiner les opérateurs précédents

```
<!ELEMENT cercle (centre, (rayon | diametre))>
<!ELEMENT contact (nom, prenom?, adresse, (telfixe |
                  telmobile | telbureau)*)>
```

INSA ROUEN NORMANDIE

Modèles de contenu 3 / 3

- Contenu mixte : mélange de données textuelles et de sous éléments
  La seule possibilité est de combiner `#PCDATA`, `|` et `*`

```
<!ELEMENT element (#PCDATA | souselement1 | ...)*>
<!ELEMENT paragraphe (#PCDATA | citation)*>
```

  `#PCDATA` doit toujours être en tête

- élément vide : l'élément n'a pas de contenu

```
<!ELEMENT element EMPTY>
```

- contenu indéterminé (à n'utiliser que pour la mise au point de DTD)

```
<!ELEMENT element ANY>
```

INSA ROUEN NORMANDIE

Déclaration d'attributs

Structure d'une déclaration d'attributs

```
<!ATTLIST element   attribut1 type déclaration_par_défaut
                    attribut2 type déclaration_par_défaut
                    ...
                    attributn type déclaration_par_défaut
>
```

Exemple

```
<!ATTLIST image src CDATA #REQUIRED
            width CDATA #REQUIRED
            height CDATA #REQUIRED
            alt  CDATA #IMPLIED
>
```

Types d'attributs 1 / 2

- `CDATA` : n'importe quelle chaîne de caractères
- `NMTOKEN` : unité lexicale nominal = chaine de caractères obéissant aux règles d'un nom XML (élément) excepté qu'il n'y a pas de restriction sur le premier caractère
- `NMTOKENS` : série de `NMTOKEN` séparés par des blancs
- énumération : un des `NMTOKEN` séparé par `|` dans une liste

```
<!ATTLIST element attribut (NOMTOKEN1 | NMTOKEN2 | ...)>
  <!ATTLIST date mois (Janvier | Février | Mars | ...)>
```

- `ID` : nom XML unique dans le document
  un élément ne peut avoir qu'un attribut de ce type

INSA ROUEN NORMANDIE

Types d'attributs 2 / 2

- `IDREF` : fait référence à la valeur d'un attribut de type `ID` permet de créer des relations entre des éléments, exemple : des projets, des personnes ⇒ des personnes qui travaillent dans un ou plusieurs projets
- `IDREFS` : série de références à des `ID` séparées par des blancs
- `ENTITY` : contient le nom d'une entité non parsée
- `ENTITIES` : série de noms d'entités non parsés, séparés par des blancs
- `NOTATION` : fait référence à une `NOTATION` déclarée dans la DTD

```
<!NOTATION png SYSTEM "image/png">
<!NOTATION jpg SYSTEM "image/jpeg">
<!ATTLIST image type NOTATION (png | jpg) #REQUIRED>
```

INSA ROUEN NORMANDIE

Déclaration par défaut

- `#IMPLIED` : l'attribut est optionnel
- `#REQUIRED` : l'attribut est obligatoire
- `#FIXED` : la valeur de l'attribut est fixe et non modifiable
  L'attribut est présent dans l'élément même si il est omis

```
<!ATTLIST element attribut type #FIXED
"valeur_obligatoire">
```

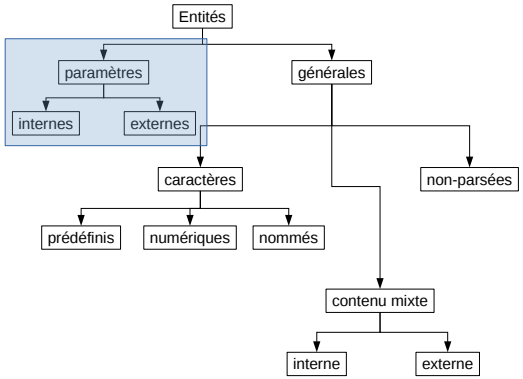
- littéral : la valeur par défaut de l'attribut est spécifiée

```
<!ATTLIST element attribut type "valeur_par_default">
```

INSA ROUEN NORMANDIE

Utilisation et déclaration des entités paramètres

Les entités générales ne peuvent servir de texte de substitution pour un modèle de contenu ou une liste d'attributs dans une DTD
Les entités paramètres autorisent ces substitutions dans une DTD



Déclaration

```
<!ENTITY % entite_parametre "texte_de_substitution">
```

Utilisation

```
%entite_parametre;
```

Utile pour répéter des modèles de contenu ou des listes d'attributs communs à des éléments

Exemple d'entité paramètre

Fichier "location.dtd"

```
<!ELEMENT location (appartement | maison | meuble | chambre)+>
<!ELEMENT appartement (adresse, nb_pieces, loyer, charges)
<!ELEMENT maison (adresse, nb_pieces, loyer)
<!ELEMENT meuble (adresse, nb_pieces, loyer, charges)
<!ELEMENT chambre (adresse, nb_pieces, loyer, charges)
```

Avec une entité paramètre :

```
<!ELEMENT location (appartement | maison | meuble | chambre)+>
<!ENTITY % caracteristiques "adresse, nb_pieces, loyer">
<!ELEMENT appartement (%caracteristiques;, charges)>
<!ELEMENT maison (%caracteristiques;)>
<!ELEMENT meuble (%caracteristiques;, charges)>
<!ELEMENT chambre (%caracteristiques;, charges)>
```

⇒ plus de souplesse dans la gestion de la DTD

Remarque : L'utilisation de l'entité paramètre en tant que partie d'une déclaration d'élément n'est possible qu'en DTD externe

Rédéfinition d'entités paramètres

Il est possible de redéfinir de manière locale une entité paramètre

En cas de conflit de noms d'entités paramètres, la première lue est prioritaire (la DTD interne est lue en premier)

Dans l'exemple précédent "`location.dtd`", il est possible d'ajouter dans le sous ensemble interne de la DTD du document un sous élément :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE location SYSTEM "location.dtd" [
  <!ENTITY % caractéristiques "adresse, nb_pieces, loyer, type_chauffage">
]>
```

Sous ensembles externes de DTD

Les DTD peuvent être très complexes et donc très longues
`XHTML strict` : +1500 lignes, `DocBook` : +11000 lignes

Il est possible de séparer une DTD en plusieurs parties
`DocBook` : 28 parties (tableaux, notations, . . .)

Association des parties : *appels d'entités paramètres externes*

Déclaration :

<!**ENTITY** % entites **SYSTEM** "URI−DTD">

Insertion de la DTD externe :

%entites;

Inclusion conditionnelle

Directive : IGNORE

```
<![IGNORE[
  déclaration à ignorer
]]>
```

Directive : INCLUDE

```
<![INCLUDE[
  déclaration à inclure
]]>
```

Principe : utiliser une entité paramètre pour les directives IGNORE et INCLUDE

Exemple

```
<!ENTITY % definition_prix "INCLUDE">
<![%definition_prix;[
  <!ELEMENT prixHT (#PCDATA)>
  <!ELEMENT prixTTC (#PCDATA)>
]]>
```

En fonction du besoin redéfinir definition_prix dans un sous-ensemble interne de DTD, avec pour valeur INCLUDE ou IGNORE.

Quelques références

- XML in a Nutshell, O'Reilly (Elliotte Rusty Harold & W. Scott Means)
- Introduction à XML, O'Reilly (Erik T. Ray)
- XML langage et applications, Eyrolles (Alain Michard)

INSA ROUEN NORMANDIE