

# Introduction à XML et aux DTD

L3Pro BDISE – XML

Mathieu Sassolas

IUT de Sénart Fontainebleau  
Département Informatique

Année 2016-2017  
Cours 1



Intro XML &  
DTD

M. Sassolas

L3Pro

Cours 1

Programme

XHTML &  
XML

DTD

TD/TP

- ▶ XHTML et XML.
- ▶ DTD (« Doctypes »).
- ▶ CSS pour (X)HTML et XML.
- ▶ Transformations XSLT (en se servant de XPath).
- ▶ Requêtes XQuery.
- ▶ Application du XML : le SVG.

[www.lacl.fr/~msassolas/enseignement/XML\\_Licence/](http://www.lacl.fr/~msassolas/enseignement/XML_Licence/)

- ▶ Cours
- ▶ Sujets de TD/TP.
- ▶ Parfois également des corrigés.
- ▶ Archives de contrôles.
- ▶ Sujets de TP notés le moment venu.

À garder dans ses favoris dès le premier TP.

## TP noté n° 1

- ▶ Vendredi 2 décembre, 14h.
- ▶ Coefficient 2,5.
- ▶ Programme : XML de base, DTD, CSS, un peu de XSLT.

## TP noté n° 2

- ▶ Semaine du 16 janvier.
- ▶ Coefficient 2,5.
- ▶ Programme : XSLT, XPath, XQuery, SVG.

Susceptible de varier en fonction de l'état d'avancement du cours, des disponibilités de l'emploi du temps. . .

- 1 Programme global
- 2 Introduction : XHTML et XML
  - XHTML
  - XML en 5 secondes
  - Tout de même quelques détails supplémentaires sur XML
- 3 Document Type Definition (DTD)
  - Introduction : pourquoi utiliser les DTD ?
  - Comment invoquer une DTD
  - Construction d'une DTD
  - Les entités
  - Les notations
- 4 Mise en application

- 1 Programme global
- 2 Introduction : XHTML et XML
  - XHTML
  - XML en 5 secondes
  - Tout de même quelques détails supplémentaires sur XML
- 3 Document Type Definition (DTD)
  - Introduction : pourquoi utiliser les DTD ?
  - Comment invoquer une DTD
  - Construction d'une DTD
  - Les entités
  - Les notations
- 4 Mise en application

▶ Tout dans la page est un **élément** encadré par des **balises**, qui peuvent avoir des **attributs**.

▶ `<balise attribut="valeur">Contenu</balise>`

~> `<div id="aveu">J'aime le XHTML!</div>`

▶ **Élément sans contenu** : `<balise/>`

~> ``

Vous connaissez déjà toutes les balises HTML nécessaires à l'édition d'une page.

- ▶ Les balises sont en minuscule.
- ▶ Les attributs sont en minuscule.
- ▶ Les guillemets sont obligatoires : `attribut="valeur"`.
- ▶ Tous les attributs ont une valeur :  
`<option selected="selected">`.

Toutes les balises doivent être fermée (même celles sans contenu) :

- ▶ ~~`<br>`~~ → `<br/>`
- ▶ ~~`<div><p>Bla bla bla</div>`~~ →  
`<div><p>Bla bla bla</p></div>`

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

- ▶ Est utilisée par les **validateurs**.
- ▶ Il y a en fait plusieurs versions du XHTML :
  - *strict* : conseillée car plus propre, et refuse les archaïsmes ;
  - *transitional* : autorise des archaïsmes (éléments et attributs « deprecated ») ;
  - *frameset* : pour des page en tant qu'ensemble de trames (utilisé dans les années 1990...).

- ▶ Écrivez des pages **valides** et faites-les valider :  
<http://validator.w3.org/> (validateur du W3C).
- ▶ Dissociez le **fond** de la **forme** (CSS, à venir).

## Remarque

*De nos jours, il est plus probable que vous écriviez du code qui **génère** de l'HTML plutôt que vous ne l'écriviez directement. Gardez en tête l'objectif de la validité de l'HTML généré.*

Intro XML &  
DTD

M. Sassolas  
L3Pro  
Cours 1

Programme

XHTML &  
XML

XHTML  
XML en 5"

Détails bonus

DTD

TD/TP

↪ Vous savez écrire du XHTML...

Intro XML &  
DTD

M. Sassolas  
L3Pro

Cours 1

Programme

XHTML &  
XML

XHTML

XML en 5"

Détails bonus

DTD

TD/TP

⇒ Vous savez écrire du XHTML...  
... donc vous savez écrire du XML

↪ Vous savez écrire du XHTML...

... donc vous savez écrire du XML

- ▶ Le XHTML est un cas particulier du XML.
- ▶ Dans XML, tout n'est qu'une imbrication d'éléments marqués par des balises.

## La seule différence

Vous pouvez choisir n'importe quelle balise (ou presque)

⇒ Vous savez écrire du XHTML...

... donc vous savez écrire du XML

- ▶ Le XHTML est un cas particulier du XML.
- ▶ Dans XML, tout n'est qu'une imbrication d'éléments marqués par des balises.

## La seule différence

Vous pouvez choisir n'importe quelle balise (ou presque)

En revanche...

Contrairement à XHTML, votre navigateur ne sait pas interpréter vos balises « maison ».

- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  
- ▶ CSS pour (X)HTML et XML
  
- ▶ Transformations XSLT (en se servant de XPath)
  
- ▶ Requêtes XQuery
  
- ▶ Application du XML : le SVG



- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  
- ▶ CSS pour (X)HTML et XML
  
- ▶ Transformations XSLT (en se servant de XPath)
  
- ▶ Requêtes XQuery
  
- ▶ Application du XML : le SVG

- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  
- ▶ CSS pour (X)HTML et XML
  
- ▶ Transformations XSLT (en se servant de XPath)
  
- ▶ Requêtes XQuery
  
- ▶ Application du XML : le SVG





- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  
- ▶ CSS pour (X)HTML et XML
  
- ▶ Transformations XSLT (en se servant de XPath)
  
- ▶ Requêtes XQuery
  
- ▶ Application du XML : le SVG

▶ XHTML

▶ XML

▶ DTD (« Doctypes »)

Comment tout de même brider les balises utilisées.

▶ CSS pour (X)HTML et XML

▶ Transformations XSLT (en se servant de XPath)

▶ Requêtes XQuery

▶ Application du XML : le SVG



- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)  
Comment tout de même brider les balises utilisées.
- ▶ CSS pour (X)HTML et XML  
Comment afficher correctement le XML.
- ▶ Transformations XSLT (en se servant de XPath)
- ▶ Requêtes XQuery
- ▶ Application du XML : le SVG



- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  - Comment tout de même brider les balises utilisées.
- ▶ CSS pour (X)HTML et XML
  - Comment afficher correctement le XML.
- ▶ Transformations XSLT (en se servant de XPath)
  - Comment transformer du XML en autre chose.
- ▶ Requêtes XQuery
  
- ▶ Application du XML : le SVG





- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)
  - Comment tout de même brider les balises utilisées.
- ▶ CSS pour (X)HTML et XML
  - Comment afficher correctement le XML.
- ▶ Transformations XSLT (en se servant de XPath)
  - Comment transformer du XML en autre chose.
- ▶ Requêtes XQuery
  - Comment utiliser XML comme base de données.
- ▶ Application du XML : le SVG



- ▶ XHTML
- ▶ XML
- ▶ DTD (« Doctypes »)  
Comment tout de même brider les balises utilisées.
- ▶ CSS pour (X)HTML et XML  
Comment afficher correctement le XML.
- ▶ Transformations XSLT (en se servant de XPath)  
Comment transformer du XML en autre chose.
- ▶ Requêtes XQuery  
Comment utiliser XML comme base de données.
- ▶ Application du XML : le SVG  
Comment utiliser XML pour dessiner.

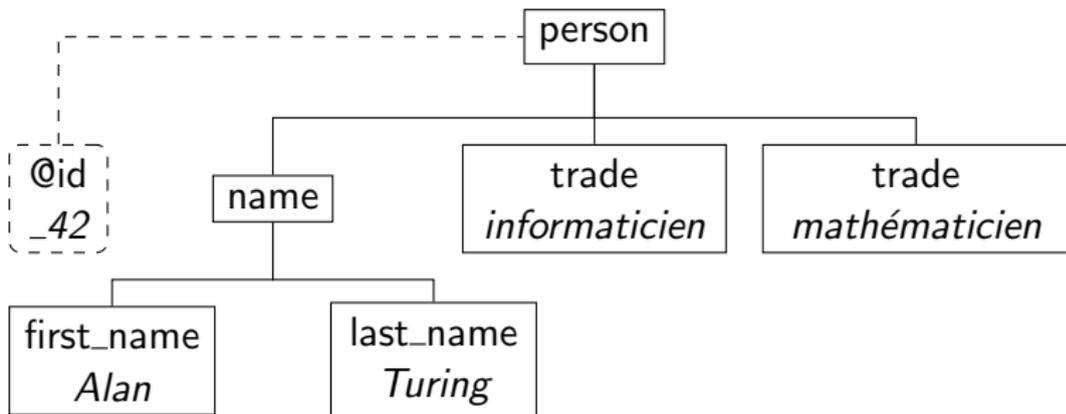
- ▶ Comme le XML est une imbrication d'éléments, on peut le voir comme un arbre.

## Attention !

Un arbre a une seule racine !  
Sinon, cela s'appelle une forêt.

- ▶ Il peut permettre de stocker des données de manière structurée.
- ↪ Souvent le résultat d'une requête sur une base de données sera renvoyée par le serveur sous forme de XML.
- ▶ Chaque élément a des attributs.

```
<person id="_42">
  <name><first_name>Alan</first_name>
    <last_name>Turing</last_name></name>
  <trade>informaticien</trade>
  <trade>mathématicien</trade>
</person>
```



↪ Comme en HTML, il faut échapper les caractères spéciaux.

< &lt;

> &gt;

" &quot;

' &apos;

& &amp;

## Vocabulaire

En XML, ces caractères échappés s'appellent des **entités**.

- ▶ Il faut en début de document préciser qu'on parle en XML :  
`<?xml version="1.0" encoding="UTF-8"?>`
- ▶ On peut utiliser des commentaires de la même manière qu'en (X)HTML :  
`<!-- Ceci est un commentaire. -->`
- ▶ XML (et les DTD, et tout le reste dans ce cours) sont **sensibles à la casse.**

- 1 Programme global
- 2 Introduction : XHTML et XML
  - XHTML
  - XML en 5 secondes
  - Tout de même quelques détails supplémentaires sur XML
- 3 Document Type Definition (DTD)
  - Introduction : pourquoi utiliser les DTD ?
  - Comment invoquer une DTD
  - Construction d'une DTD
  - Les entités
  - Les notations
- 4 Mise en application

Pour spécifier la structure d'un document.

- ▶ Pour avoir des hypothèses sur les éléments rencontrés (potentiellement pour les afficher d'une certaine manière, cf XHTML et – plus tard – CSS).
- ▶ Pour garantir qu'un document suit bien les règles d'une application spécifique (format d'échange de données dans un protocole, par exemple).
- ▶ Pour dire ce qu'il est possible ou impossible de trouver dans un document. Ex : tel élément peut être suivi de tel autre.
- ▶ Pour spécifier que certains éléments doivent avoir des attributs obligatoires. Ex : tout élément `employé` doit avoir un attribut `no_secu`.

↪ Le cas de XHTML :

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

↪ Le cas de XHTML :

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

↪ Dissection :

**DOCTYPE** Mot clef pour dire qu'on déclare une DTD.

↪ Le cas de XHTML :

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

↪ Dissection :

**DOCTYPE** Mot clef pour dire qu'on déclare une DTD.

**html** Nom de l'élément racine.

↪ Le cas de XHTML :

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

↪ Dissection :

**DOCTYPE** Mot clef pour dire qu'on déclare une DTD.

**html** Nom de l'élément racine.

**PUBLIC** La DTD est **publique** : accessible publiquement en ligne. Autre possibilité : **SYSTEM**, la DTD se trouve sur le serveur.

↪ Le cas de XHTML :

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

↪ Dissection :

**DOCTYPE** Mot clef pour dire qu'on déclare une DTD.

**html** Nom de l'élément racine.

**PUBLIC** La DTD est **publique** : accessible publiquement en ligne. Autre possibilité : **SYSTEM**, la DTD se trouve sur le serveur.

**"-//W3C//DTD XHTML 1.0 Strict//EN"** Identifiant de la DTD ; ne s'applique que pour les DTD publiques.

↪ Le cas de XHTML :

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

↪ Dissection :

**DOCTYPE** Mot clef pour dire qu'on déclare une DTD.

**html** Nom de l'élément racine.

**PUBLIC** La DTD est **publique** : accessible publiquement en ligne. Autre possibilité : **SYSTEM**, la DTD se trouve sur le serveur.

**"-//W3C//DTD XHTML 1.0 Strict//EN"** Identifiant de la DTD ; ne s'applique que pour les DTD publiques.

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"**  
URL de la DTD

# Comment invoquer une DTD ?

## Cas d'une DTD interne

Intro XML &  
DTD

M. Sassolas  
L3Pro  
Cours 1

Programme

XHTML &  
XML

DTD

Intro

Invocation

Dissection I

Les entités

Les notations

TD/TP

```
<!DOCTYPE population[
```

Le contenu de la DTD

```
]>
```

```
</population/>
```

```
<!DOCTYPE population SYSTEM "../DTDs/population.dtd">
```

Intro XML &  
DTD

M. Sassolas

L3Pro

Cours 1

Programme

XHTML &  
XML

DTD

Intro

Invocation

Dissection I

Les entités

Les notations

TD/TP

```
<!DOCTYPE population SYSTEM "../DTDs/population.dtd">
```

## Contenu de population.dtd

```
<!ELEMENT population (person)*>  
<!ELEMENT person (name, trade+)>  
<!ELEMENT name ((first_name,last_name)|  
                (last_name,first_name))>  
<!ELEMENT first_name (#PCDATA)>  
<!ELEMENT last_name (#PCDATA)>  
<!ELEMENT trade (#PCDATA)>  
<!ATTLIST person id ID #REQUIRED>  
<!ATTLIST last_name von_part CDATA #IMPLIED  
                    junior CDATA #IMPLIED>
```

```
<!DOCTYPE population SYSTEM "../DTDs/population.dtd">
```

## Contenu de population.dtd

```
<!ELEMENT population (person)*>  
<!ELEMENT person (name, trade)*>  
<!ELEMENT name ((first_name | last_name) |  
                (first_name last_name))>  
<!ELEMENT first_name (#PCDATA)>  
<!ELEMENT last_name (#PCDATA)>  
<!ELEMENT trade (#PCDATA)>  
<!ATTLIST person id ID #REQUIRED>  
<!ATTLIST last_name von_part CDATA #IMPLIED  
                    junior CDATA #IMPLIED>
```



## Erreur fréquente à éviter

Pas de `<!DOCTYPE racine [...]>` dans une DTD externe.

## Concernant l'invocation

- ▶ Il faut parfois forcer les programmes à aller lire la DTD : un navigateur web s'en fiche ; pour le TP :

Validation avec la DTD incluse ou référencée

```
xmllint --valid monXml.ml
```

- ▶ Il n'est pas obligatoire d'invoquer une DTD dans le XML :

Validation avec une autre DTD

```
xmllint --dtdvalid maDtd.dtd monXml.ml
```

## Exemple

```
<!ELEMENT population (person)*>
```

## Syntaxe

```
<!ELEMENT nom_de_l_element contenu_de_l_element>
```

## Exemple

```
<!ELEMENT population (person)*>
```

## Syntaxe

```
<!ELEMENT nom_de_l_element contenu_de_l_element>
```

Le `nom_de_l_element` :

- ▶ est formé de lettres, chiffres, et de caractères de ponctuation : . (point), - (tiret), : (deux points), \_ (tiret bas);
- ▶ doit commencer par une lettre ou \_;
- ▶ est sensible à la casse (majuscule/minuscule);
- ▶ ne doit pas commencer par `xml` (ni XML, Xml, xML...).

## Exemple

```
<!ELEMENT population (person)*>
```

## Syntaxe

```
<!ELEMENT nom_de_l_element contenu_de_l_element>
```

Dans la mesure du possible, le `nom_de_l_element` :

- ▶ évite d'utiliser `:` (deux points) hormis pour spécifier l'espace de noms ;
- ▶ évite l'utilisation de lettres non ASCII (même lorsque l'encodage est spécifié en entête), certains logiciels le supportent mal.

# Exemple de XML accepté

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE population SYSTEM "population.dtd">
<population>
  <person >
    [Redacted]
  </person>
  <person >
    [Redacted]
  </person>
</population>
```

- ▶ Le contenu\_de\_1\_element spécifie quels autres éléments peuvent apparaître en tant que fils, et de quelle manière.
- ▶ Les éléments fils doivent aussi être spécifiés dans la DTD.
- ▶ Les fils peuvent être :

**#PCDATA** « parsed character data » : texte qui sera lu lors du parcours  $\rightsquigarrow$  les entités (&toto;) seront remplacées par leur valeur.

### Exemple

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

(elt1,elt2,...) Éléments elt1, elt2,... dans cet ordre.

### Exemple

```
<!ELEMENT name (first_name,last_name)>
```

(elt1 | elt2 | ...) Éléments elt1 ou elt2,...

### Exemple

```
<!ELEMENT troisieme_plat (fromage|dessert|cafe)>
```

**EMPTY** L'élément doit alors être vide.

### Exemple

```
<!ELEMENT image EMPTY>
```

**ANY** L'élément peut contenir n'importe quoi.

### Exemple

```
<!ELEMENT trucs ANY>
```

- ▶ On peut spécifier le nombre d'éléments :

\* 0 ou plus

? 0 ou 1

+ 1 ou plus

### Exemple

```
<!ELEMENT population (person)*>
```

- ▶ On peut évidemment combiner tout cela :

### Exemple

```
<!ELEMENT person (name, trade+)>
```

```
<!ELEMENT name ((first_name,last_name)|  
                (last_name,first_name))>
```

Intro XML &  
DTD

M. Sassolas

L3Pro

Cours 1

Programme

XHTML &  
XML

DTD

Intro

Invocation

Dissection !

Les entités

Les notations

TD/TP

- ▶ Comment spécifier : « les enfants de pere sont des fils et des filles, **sans ordre particulier** » ?

- Comment spécifier : « les enfants de pere sont des fils et des filles, **sans ordre particulier** » :

```
<!ELEMENT pere (fils|fille)* >  
(au moins 1 : <!ELEMENT pere (fils|fille)+ >)
```

- Comment spécifier : « les enfants de pere sont des fils et des filles, **sans ordre particulier** » :

```
<!ELEMENT pere (fils|fille)* >
(au moins 1 : <!ELEMENT pere (fils|fille)+ >)
```

- Sur le même principe : mélange de texte et d'éléments fils :

```
<!ELEMENT pere (#PCDATA|fils)* >
~> <pere>Blabla <fils/> Bloblu </fils></pere>
```

- **Attention !** Dans le cas d'un mélange entre texte (#PCDATA) et d'autres fils, #PCDATA doit être le premier.
- ↪ On appelle cela du **contenu mixte** (*mixed content*).

# Exemple de XML accepté

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE population SYSTEM "population.dtd">
<population>
  <person >
    <name>
      <last_name>Sassolas</last_name>
      <first_name>Mathieu</first_name>
    </name>
    <trade>informaticien</trade>
  </person>
  <person >
    <name>
      <first_name>Alan</first_name>
      <last_name>Turing</last_name>
    </name>
    <trade>informaticien</trade>
    <trade>mathématicien</trade>
  </person>
</population>
```

## Exemple

```
<!ATTLIST last_name von_part CDATA #IMPLIED  
                junior CDATA #IMPLIED>
```

## Syntaxe

```
<!ATTLIST nom_de_l_element  
          nom_de_l_attribut1 type_de_l_attribut1 options1  
          nom_de_l_attribut1 type_de_l_attribut1 options2  
          ...>
```

Le nom d'un attribut :

- ▶ est formé de lettres, chiffres, et de caractères de ponctuation : ., -, :, \_;
- ▶ doit commencer par une lettre ou \_;
- ▶ est sensible à la casse (majuscule/minuscule).

**CDATA** « character data » : du texte (qui ne sera pas lu).

### Exemple

```
<!ATTLIST last_name von_part CDATA #IMPLIED  
                  junior CDATA #IMPLIED>
```

(**val1|val2|...**) Une valeur parmi val1, val2...

### Exemple

```
<!ATTLIST image type (jpeg|gif|png|svg) #REQUIRED>
```

**NMTOKEN** du texte sans espace.

### Exemple

```
<!ATTLIST person country_code NMTOKEN #IMPLIED>
```

**NMTOKENS** plusieurs morceaux de texte (sans espaces)  
séparés par des espaces.

**ID** identifiant : la valeur doit être unique dans le document XML (et obéir aux mêmes règles que les noms d'attributs).

### Exemple

```
<!ATTLIST person id ID #REQUIRED>
```

**IDREF**, **IDREFS** référence vers un (resp. plusieurs) identifiants.

**ENTITY**, **ENTITIES** une (resp. plusieurs) entités (*cf* suite).

**NOTATION** (**not1|not2...**) une notation parmi not1,  
not2,....

"**valeur\_par\_defaut**" Tous les éléments auront cet attribut, avec cette valeur si elle n'est pas spécifiée explicitement.

### Exemple

```
<!ATTLIST image type (jpeg|gif|png|svg) "jpeg">
```

**#REQUIRED** Attribut obligatoire ; le XML n'est pas valide s'il n'est pas spécifié.

### Exemple

```
<!ATTLIST person id ID #REQUIRED>
```



- ▶ Dans le cas des entités prédéfinies (&, <, >, ", '), pour pouvoir utiliser des caractères réservés.
- ▶ Servent de raccourci pour ne pas avoir à répéter certaines choses.
- ▶ Peuvent n'être utilisables que dans la DTD (entité **paramétrée**) ou également dans le XML (entité **générale**, c'est le cas des entités prédéfinies).
- ▶ Peuvent être externes ou internes (au document).
- ▶ **Attention !** Dans le cas d'entités externes, il faut que le lecteur du XML aille bien lire la DTD pour remplacer les entités. **NB** : les navigateurs web ne le font pas.

## Exemple

```
<!DOCTYPE tract [  
  <!ELEMENT tract (#PCDATA | a)*>  
  <!ELEMENT a (#PCDATA)>  
  <!ATTLIST a href CDATA #IMPLIED>  
  <!ENTITY moi "moi moi moi moi moi!">  
  <!ENTITY vpm "votez pour  
    <a href='http://votepourmoi.com'>&moi;</a">  
  
<tract>Bonjour, &vpm;</tract>
```

## Syntaxe

```
<!ENTITY nom_de_l_entite "valeur_de_l_entite">
```

## Exemple

```
<!DOCTYPE tract [  
  <!ELEMENT tract (#PCDATA | a)*>  
  <!ELEMENT a (#PCDATA)>  
  <!ATTLIST a href CDATA #IMPLIED>  
  <!ENTITY moi "moi moi moi moi moi!">  
  <!ENTITY vpm "votez pour  
    <a href='http://votepourmoi.com'>&moi;</a>">  

```

```
<tract>Bonjour, &vpm;</tract>
```

```
--<tract>  
  Bonjour, votez pour  
  <a href="http://votepourmoi.com">moi moi moi moi moi!</a>  
</tract>
```

### Exemple

```
<!DOCTYPE tract [  
  <!ELEMENT tract (#PCDATA)>  
  <!ENTITY moi SYSTEM "moi.txt">  
  <!ENTITY prog PUBLIC "Mon Programme"  
    "http://votezpourmoi.com/programme.txt">  

```

```
<tract>Bonjour, je suis &moi;;  
  j'ai un beau programme: &prog;</tract>
```

### Syntaxe

```
<!ENTITY nom_de_l_entite SYSTEM "URI_de_l_entite">  
<!ENTITY nom_de_l_entite PUBLIC "identifiant"  
  "URL_de_l_entite">
```

↪ Ces entités ne sont utilisables que dans des attributs (cas `<!ATTLIST e1 att ENTITY #IMPLIED>`).

## Exemple

```
<!DOCTYPE tract [  
  <!ELEMENT tract (#PCDATA | image)*>  
  <!ELEMENT image EMPTY>  
  <!ATTLIST image source ENTITY #REQUIRED>  
  <!ENTITY moi SYSTEM  
    "http://votexpourmoi.com/picture.png" NDATA png>  
  <!NOTATION png SYSTEM "image/png">  

```

```
<tract>Ne suis-je beau?
```

```
  <image source="moi"/></tract>
```

### Syntaxe

```
<!ENTITY nom_de_l_entite SYSTEM "URI_de_l_entite"  
          NDATA notation>
```

```
<!ENTITY nom_de_l_entite PUBLIC "identifiant"  
          "URL_de_l_entite" NDATA notation>
```

- ▶ L'entité contient du code qui ne doit pas être analysé comme du XML.
- ▶ NDATA notation indique le type de données de l'entité, et donc comment elle devra être traitée.
- ▶ La notation devra elle aussi être définie (ce n'est qu'un nom), pour définir vraiment comment traiter ce type de contenu.

- ▶ Servent de variable dans la DTD.
- ▶ Ne peuvent être utilisées que dans des DTD externes.

### Exemple

```
<!ELEMENT livre (auteur, editeur)>
<!ENTITY % info "(prenom,nom)">
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT auteur %info;>
<!ELEMENT editeur (%info;,adresse)>
<!ELEMENT adresse (#PCDATA)>
```

### Syntaxe

```
<!ENTITY % nom_de_l_entite "contenu">
<!ELEMENT elt %nom_de_l_entite;>
```

```
<!DOCTYPE livre SYSTEM "livre.dtd">
```

```
<livre>
```

```
<auteur>
```

```
<prenom>Victor</prenom>
```

```
<nom>Hugo</nom>
```

```
</auteur>
```

```
<editeur>
```

```
<prenom>François</prenom>
```

```
<nom>Didot</nom>
```

```
<adresse>À l'angle du quai des Grands-Augustins  
et de la rue Séguier, à Paris</adresse>
```

```
</editeur>
```

```
</livre>
```

```
<!-- % nom_de_l_entite "contenu" -->
```

```
<!ELEMENT elt %nom_de_l_entite;>
```

- ▶ Même principe, mais le contenu est dans une DTD

## Exemple

```
<!DOCTYPE bibliotheque [  
  <!ENTITY % livre SYSTEM "livre.dtd">  
  %livre;  
  <!ELEMENT bibliotheque (livre*)>  
>
```

## Syntaxe

```
<!ENTITY % nom_de_l_entite SYSTEM  
                                "URL_de_la_DTD.dtd">  
%nom_de_l_entite;
```

- ▶ On peut utiliser des entités dans la définition des entités.

## Exemple

```
<!ENTITY % iut "IUT de Fontainebleau">
<!ENTITY copyright "Tous droits réservés
                    1993-2013, %iut;">
```

- ▶ Attention à ne pas faire de définition circulaire !

## Exemple

```
<!ENTITY iut "IUT de &bleau;">
<!ENTITY bleau "Fontainebleau, la ville où se
                trouve l'&iut;">
```

- ▶ On peut utiliser des entités dans la définition des entités.

## Exemple

```

<!ENTITY % iut "IUT de Fontainebleau">
<!ENTITY copyright "Tous droits réservés
                                1993-2013, %iut;">
  
```

- ▶ Attention à ne pas faire de définition circulaire !

## Exemple

~~```

<!ENTITY iut "IUT de &bleau;">
<!ENTITY bleau "Fontainebleau est la ville où se
                                trouve %iut;">
  
```~~

- ▶ Les **notations** servent à définir des morceaux de texte/code qui seront lus par une autre application.
- ▶ Cas d'usage majoritaire : définir les **types de données** (*MIME types*).

## Exemple

```
<!NOTATION gif PUBLIC "image/gif">  
<!NOTATION jpeg PUBLIC "image/jpeg">  
<!ATTLIST image type NOTATION (gif|jpeg) #REQUIRED>
```

## Syntaxe

```
<!NOTATION nom_de_la_notation SYSTEM  
                                "contenu_de_la_notation">  
<!NOTATION nom_de_la_notation PUBLIC "identifiant"  
                                "contenu_de_la_notation">
```

- 1 Programme global
- 2 Introduction : XHTML et XML
  - XHTML
  - XML en 5 secondes
  - Tout de même quelques détails supplémentaires sur XML
- 3 Document Type Definition (DTD)
  - Introduction : pourquoi utiliser les DTD ?
  - Comment invoquer une DTD
  - Construction d'une DTD
  - Les entités
  - Les notations
- 4 Mise en application

Intro XML &  
DTD

M. Sassolas

L3Pro

Cours 1

Programme

XHTML &  
XML

DTD

TD/TP

↳ On commence le TD ←