



Séance 2

Contraindre le XML

La Document Type Definition

Bien formé ?

```
<paragraphe>du texte</paragraphe>
```

```
<paragraphe><article>du</article><nom>texte</nom></paragraphe>
```

```
<paragraphe><article>du <nom></article>texte</nom></paragraphe>
```

```
<paragraphe><article>du <nom>texte</nom></article></paragraphe>
```

```
<paragraphe type="texte">du texte</paragraphe>
```

```
<paragraphe type=texte>du texte</paragraphe>
```

```
<paragraphe type="texte">du texte<paragraphe/>
```

```
<paragraphe type="texte">du texte<nom>nom</paragraphe>
```

```
<paragraphe type="texte">du texte</Paragraphe>
```

```
<segment type="texte" type="nombre">du texte</paragraphe>
```

Spécifications XML

Les spécifications XML définissent un modèle pour les documents XML qui réglemente la manière dont les données sont encodées

Pourquoi contraindre ?

Uniformité

Dans un livre, il faut que les chapitres soient structurés similairement

Exploitation

Automatisation du traitement des données seulement si elles suivent le même format

Ajout

Plus grande facilité à ajouter de nouvelle donnée si celle-ci suit un modèle

Collaboration

Modèle indispensable dans un projet où différentes personnes encodent dans un but commun

```
<texte>
  <chap>
    <titre><nom>Blanche-Neige</nom> mange la pomme</titre>
    <p>Il était une fois <nom>Blanche-Neige</nom>
      <note n="1">son visage est blanc comme la neige</note>
      que sa belle-mère détestait. [...]</p>
  </chap>
  <div1>
    <h1>Blanche-Neige attend le prince Charmant</h1>
    <div2>Un jour <perso>Charmant</perso><add ref="#1"/>
      arriva sur son cheval blanc et la sauva. [...]</div2>
  </div1>
  <footnote id="1">Nom du prince qui sauve Blanche-Neige</footnote>
</texte>
```

Principe de validation

- Respect de la syntaxe XML : document **bien formé**
- Respect de la structure définie dans une DTD ou un **schéma**
- Toutes les **références** à des entités sont résolues

= tout ce qui n'est pas spécifié est **interdit**



Bien formé / valide

Bien formé

Respect de la
syntaxe XML

Valide

Respect de la syntaxe
XML et conformité à un
schéma

Les différents modèles

DTD

Pas du XML

`.dtd`

XMLSchema

En XML

`.xsd`

Pas d'entités

Domaine de validité
pour la valeur d'un
champ

RelaxNG

En XML

`.rng`

Définit la structure du
document XML

La *Document Type Definition*

Une DTD se présente sous la forme d'une **liste de déclarations** qui définissent la structure que devront suivre les documents qui s'y réfèrent.

La DTD qui sert de modèle à un document doit être déclarée au début de celui-ci, **avant l'ouverture de l'élément racine**.

Contenu d'une DTD

Une DTD contient, dans un ordre indifférent des déclarations concernant :

Éléments

Définit les éléments qui peuvent être utilisés, leur nombre, leur imbrication, leur ordre, etc.

Attributs

Définit les attributs autorisés pour un élément précis, leur valeurs et le type de valeur autorisé

Entités

Définit les entités qui pourront être utilisées dans le document XML

A decorative graphic on the left side of the slide consisting of two blue squares. The top square is light blue and the bottom square is a darker blue, stacked vertically.

Déclaration de la DTD

La déclaration de la DTD peut être faite au **début du document XML** dont elle contraint l'encodage, ou dans un **fichier externe** dont il est fait référence dans le document XML

Déclaration interne

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE personne [
    <!--début de la DTD interne -->
    <!ELEMENT personne (prenom, nom)>
    <!ELEMENT prenom (#PCDATA)>
    <!ELEMENT nom (#PCDATA)>
    <!--fin de la DTD interne -->
]>

<!--début du document-->
<personne>
    <prenom>Albert</prenom>
    <nom>Camus</nom>
</personne>
<!--fin du document-->
```

Déclaration externe

Document XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personne SYSTEM "modele.dtd">

<!--début du document-->
<personne>
    <prenom>Albert</prenom>
    <nom>Camus</nom>
</personne>
<!--fin du document-->
```

modele.dtd

```
<!ELEMENT personne (prenom, nom)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
```

Déclaration Web

```
<!DOCTYPE html SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

A decorative graphic on the left side of the slide consisting of two blue squares. The top square is a lighter shade of blue and is positioned above the bottom square, which is a darker shade of blue. They are aligned to the left edge of the slide.

La syntaxe DTD

Comment contraindre la validation du
document XML à son schéma

Éléments

```
<!ELEMENT nom_element modele_contenu>
```

Le modèle de contenu peut contenir des règles sur le **type** et le **nombre** de contenus admis

Le modèle de contenu ne porte que sur les éléments **directement contenus** dans la balise, en non sur les sous-éléments potentiellement contenus

Type de contenu

`<!ELEMENT nom_element (#PCDATA)>`

Expression Signification

EMPTY contenu vide

ANY contenu quelconque

(#PCDATA) contenu textuel

(*elt*) un seul élément

(*elt*₁, *elt*₂, ..., *elt*_{*n*}) séquence d'éléments pris
dans cet ordre

(*elt*₁ | *elt*₂ | ... | *elt*_{*n*}) un des éléments **au choix**

 Pour déclarer des sous-éléments et du texte dans un élément, il faut déclarer (#PCDATA ...)* en premier suivi d'un * après les parenthèses

Règle → application

```
<!ELEMENT sonnet (quatrain, quatrain, sizain)>
```

```
<sonnet>  
  <quatrain>...</quatrain>  
  <quatrain>...</quatrain>  
  <sizain>  
    <tercet>...</tercet>  
    <tercet>...</tercet>  
  </sizain>  
</sonnet>
```

Opérateurs d'occurrence

```
<!ELEMENT el (e11+, e12?)>
```

Opérateur Signification

? 0 ou 1 occurrence

***** 0 ou plusieurs occurrences

+ 1 ou plusieurs occurrences

rien 1 seul occurrence

 Les opérateurs qualifient l'élément juste avant ou le groupe d'éléments contenu dans les parenthèses juste avant

Règle → application

```
<!ELEMENT sonnet (quatrain+, sizain)>
```

```
<sonnet>  
  <quatrain>...</quatrain>  
  <quatrain>...</quatrain>  
  <sizain>  
    <tercet>...</tercet>  
    <tercet>...</tercet>  
  </sizain>  
</sonnet>
```

L'élément **e1** contient

(**e11** OU **e12**) au moins une fois

```
<!ELEMENT e1 (e11 | e12)+>
```

```
<e1>  
  <e11></e11>  
  <e12></e12>  
  <e11></e11>  
</e1>
```

e11 au moins une fois OU **e12** au moins une fois

```
<!ELEMENT e1 (e11+ | e12+)>
```

```
<e1>  
  <e11></e11>  
  <e11></e11>  
</e1>
```

```
<e1>  
  <e12></e12>  
  <e12></e12>  
</e1>
```

Le contenu entre parenthèse doit être pris
comme une **entité autonome**

```
<!ELEMENT e1 ((e11 | e12)+ | e13+)>
```

e11 **OU** **e12** au moins une fois
OU
e13 au moins une fois

```
<e1>  
  <e11></e11>  
  <e12></e12>  
</e1>
```

```
<e1>  
  <e13></e13>  
  <e13></e13>  
</e1>
```

Exercice



**Traduire les expressions
suivantes**

Traduire ces déclaration

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

```
<!ELEMENT b EMPTY>
```

```
<!ELEMENT c ANY>
```

```
<!ELEMENT d (e1, e2, e3, e4, e5)>
```

```
<!ELEMENT e1 (a, b)+>
```

```
<!ELEMENT e2 (a | b)+>
```

```
<!ELEMENT e3 (a+ | b+)>
```

```
<!ELEMENT e4 ((a|b), c)>
```

```
<!ELEMENT e5 ((a|b)+, c)>
```


<d>

<e1> (a, b)+ <!--une séquence de couples (a, b)-->

<a>xxx<a>xxx

</e1>

<e2> (a | b)+ <!--une alternance de a ou de b-->

<a>xxx<a>xxx

</e2>

<e3> (a+ | b+) <!--une séquence de a ou une séquence de b-->

</e3>

<e4> ((a | b), c) <!--a ou b suivi de c-->

<a>xxx<c>zzz</c>

</e4>

<e5> ((a | b)+, c) <!--une alternance de a ou de b, suivie de c-->

<a>xxx<a>xxx<c>zzz</c>

</e5>

</d>

Traduire cette déclaration

```
<!ELEMENT personne (nom, prenom+, telephone*, adresse?)>
```

Traduire cette déclaration

```
<!ELEMENT personne (nom, prenom+, telephone*, adresse?)>
```

L'élément `personne` est composé, **dans cet ordre**, de :

- un élément `nom`
- un ou plusieurs éléments `prenom`
- zéro, un ou plusieurs éléments `telephone`
- zéro ou un élément `adresse`

Traduire cette déclaration

```
<!ELEMENT para (#PCDATA | note | renvoi)*>
```

Traduire cette déclaration

```
<!ELEMENT para (#PCDATA | note | renvoi)*>
```

```
<para>
```

```
    Prendre le chemin de l'Inca <renvoi>...</renvoi>et marcher  
    jusqu'à la porte du Soleil <renvoi>...</renvoi>. Ne pas  
    oublier de prendre un foulard et de bonnes chaussures  
    <note>...</note>.
```

```
</para>
```

Exercice

A decorative graphic on the right side of the slide, consisting of a light gray square positioned above a blue square, both of which are partially cut off by the right edge of the frame.

**Rédiger des
déclarations d'éléments**

Rédiger une DTD

Rédiger les déclarations d'éléments pour décrire l'encodage du poème *Mon rêve familier* de Verlaine

Attributs

```
<!ATTLIST nom_element nom_attribut type_contenu type_attribut>
```

La déclaration d'un attribut définit les attributs autorisés pour un élément en termes de **contenu** et de **type**

Type de contenu

`<!ATTLIST el att CDATA ... >`

Expression Signification

CDATA chaîne de caractère ne
comprenant pas de balises

(val1 | val2 | ...) liste de valeur à utiliser

ENTITY / ENTITIES entité déclarée dans la DTD (ou
liste séparée par des espaces)

ID pour identifier l'élément

IDREF / IDREFS ID d'un autre élément (ou liste
séparée par des espaces)

Type d'attribut

```
<!ATTLIST chap n CDATA #REQUIRED>
```

Opérateur Signification

#REQUIRED valeur requise dans l'élément

#IMPLIED valeur facultative

#FIXED **“valeur”** valeur fixe pour l'attribut

“valeur” valeur par défaut de l'attribut
(on peut la remplacer)

Règle → application

```
<!ATTLIST strophe type (quatrain | tercet) #REQUIRED>
```

```
<strophe type="quatrain">  
  <vers>...</vers>  
  <vers>...</vers>  
  <vers>...</vers>  
  <vers>...</vers>  
</strophe>  
<strophe type="tercet">  
  <vers>...</vers>  
  <vers>...</vers>  
  <vers>...</vers>  
</strophe>
```



Limites

IDREF

La DTD ne permet pas de préciser l'ID de quel type d'élément est autorisé dans un attribut de type IDREF

ID unique

La valeur d'un attribut de type ID doit être unique dans tout le document, même pour des éléments différents

ID → IDREF

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

```
<!ATTLIST pers id ID #REQUIRED  
              lieu_naissance IDREF #REQUIRED>
```

```
<lieu id="Paris"></lieu>
```

```
<pers id="nom_prenom" lieu_naissance="Paris">  
  Nom Prénom  
</pers>
```

Exercice



**Traduire les expressions
suivantes**

Traduire ces déclaration

```
<!ATTLIST livre gencode ID #REQUIRED>
<!ATTLIST livre auteur CDATA "nom">
<!ATTLIST porte ouvert (true|false) "true">
<!ATTLIST carte couleur (cœur|pique|trefle|carreau) #IMPLIED>
<!ATTLIST eleve surnom CDATA #IMPLIED>
<!ATTLIST lettre destinataire IDREF #REQUIRED>
<!ATTLIST hotel etoile (1 | 2 | 3 | 4 | 5) #IMPLIED
    responsable IDREF #REQUIRED
    code ID #REQUIRED >
```

```
<!ATTLIST livre gencode ID #REQUIRED>
```

L'élément `livre` a un attribut obligatoire `gencode` dont la valeur l'identifie de manière unique

```
<!ATTLIST livre auteur CDATA "nom">
```

L'élément `livre` a un attribut `auteur` qui contient une chaîne de caractère dont la valeur par défaut est `"nom"`

```
<!ATTLIST porte ouvert (true|false) "true">
```

L'élément `porte` a un attribut `ouvert` dont la valeur peut être `true` ou `false`, par défaut fixée à `"true"`

```
<!ATTLIST carte couleur (cœur|pique|trefle|carreau) #IMPLIED>
```

L'élément `carte` a un attribut facultatif `couleur` dont la valeur peut être `cœur`, `pique`, `trefle` ou `carreau`

```
<!ATTLIST eleve surnom CDATA #IMPLIED>
```

L'élément `eleve` a un attribut facultatif `surnom` qui contient une chaîne de caractère

```
<!ATTLIST lettre destinataire IDREF #REQUIRED>
```

L'élément `lettre` a un attribut obligatoire `destinataire` dont la valeur est l'identifiant unique d'un autre élément

Traduire ces déclaration

```
<!ATTLIST hotel etoile (1 | 2 | 3 | 4 | 5) #IMPLIED  
                responsable IDREF #REQUIRED  
                code ID #REQUIRED>
```

L'élément `hotel` a un attribut facultatif `etoile` dont la valeur comprise entre 1 et 5,
un attribut obligatoire `responsable` dont la valeur est l'identifiant unique d'un autre élément
un attribut obligatoire `code` dont la valeur l'identifie de manière unique

Exercice



**Rédiger des
déclarations d'attributs**

Rédiger une DTD

Rédiger les déclarations pour décrire l'encodage du poème
Mon rêve familial de Verlaine

Entités

```
<!ENTITY nom_entite "texte de remplacement">
```

La déclaration d'une entité permet de créer des "abréviations" qu'il sera possible d'utiliser dans le document XML

Utilisation

```
<!ENTITY enc "École des chartes">
```

À chaque fois que `&enc` ; sera présent dans le document XML, l'entité se substitue à la chaîne de caractère `"École des chartes"`

Entités prédéfinies

Entité Caractère

`<` <

`>` >

`'` ‘

`"` “

`&` &

Déclaration externe

DTD

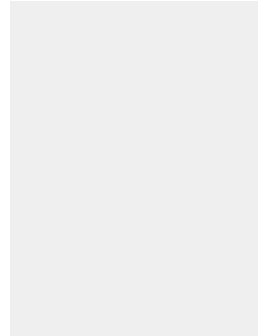
```
<!ENTITY signature SYSTEM  
    "sign.txt">
```

sign.txt

Emmanuelle Bermès
Responsable pédagogique Master TNAH
École des chartes

Exercice

**Encoder une carte
postale avec sa DTD**



Encoder une carte postale

Transcrire et encoder en XML la [carte postale](#) suivante à l'aide de ce [patron](#), puis rédiger la DTD correspondante

