S4 : Module OMGL1
BD pour les applications supportant les données et les traitements complexes.

Partie : XML
eXtended
Markup
Language

M. Boughanem

Déroulement de l'enseignement

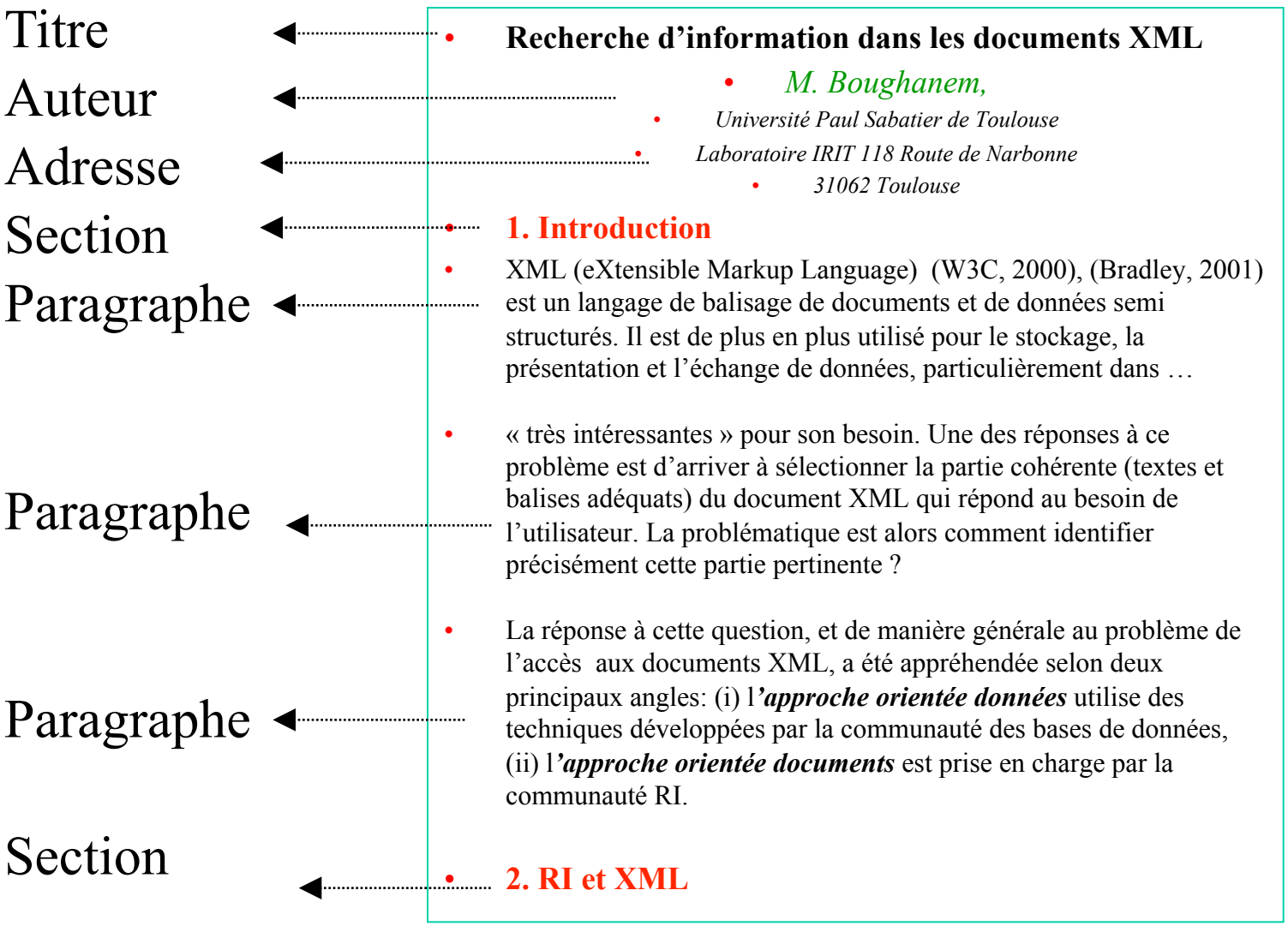
- COURS
  - Enseignant :
    - M. Boughanem
- TD et TP
  - Outil CookTop
  - Enseignants :
    - M. Boughanem et G. Cabanac

Plan

- Chapitre 1 : Introduction à XML
- Chapitre 2 : Structure d'un document XML
- Chapitre 2 : Déclarations de Type de Documents (DTD)
- Chapitre 3 : XPATH

Chapitre 1
Introduction à XML

Présentation et Structuration



Pour construire un document

- Logiciel Word de Microsoft exemple.doc
- Lecture avec Bloc Notes de Microsoft (ou un autre logiciel) exemple.txt

- … ?
  - Impossible d'échanger les documents
  - …

Introduction de la notion de balises (langage de balisage) pour séparer le contenu de la structure et de la présentation

Vue balisée du document

- <document>
- <titre> <centré> Recherche d'information dans des documents XML </centré> </titre>
- <auteur> <vert> M. Boughanem </vert> </auteur>
- <adresse> Université Paul Sabatier de Toulouse Laboratoire IRIT, 118 Route de Narbonne
- <codepostal> 31062 </codePostal> <Ville> Toulouse </Ville> </adresse>
- <section titre= "Introduction" >
- <par> XML (eXtensible Markup Language) (W3C, 2000), (Bradley, 2001) est un langage de balisage de documents et de données semi structurés. … qui retournent le document entier, en réponse à une requête utilisateur, ne sont plus adéquates. </par>
- <par> En effet, dans le cas particulier d'un document long, la réponse recherchée par l'utilisateur pourrait être " noyée " au milieu d'autres sujets. … identifier précisément cette partie pertinente ? </par>
- <par> La réponse à cette question, et de manière générale au problème de l'accès aux documents XML, a été appréhendée selon deux principaux angles: (i) l'approche orientée données utilise des techniques développées par la communauté des bases de données, (ii) l'approche orientée documents est prise en charge par la communauté RI. <par>
- </section>
- <section titre= "RI et XML" > … </section>
- </document>

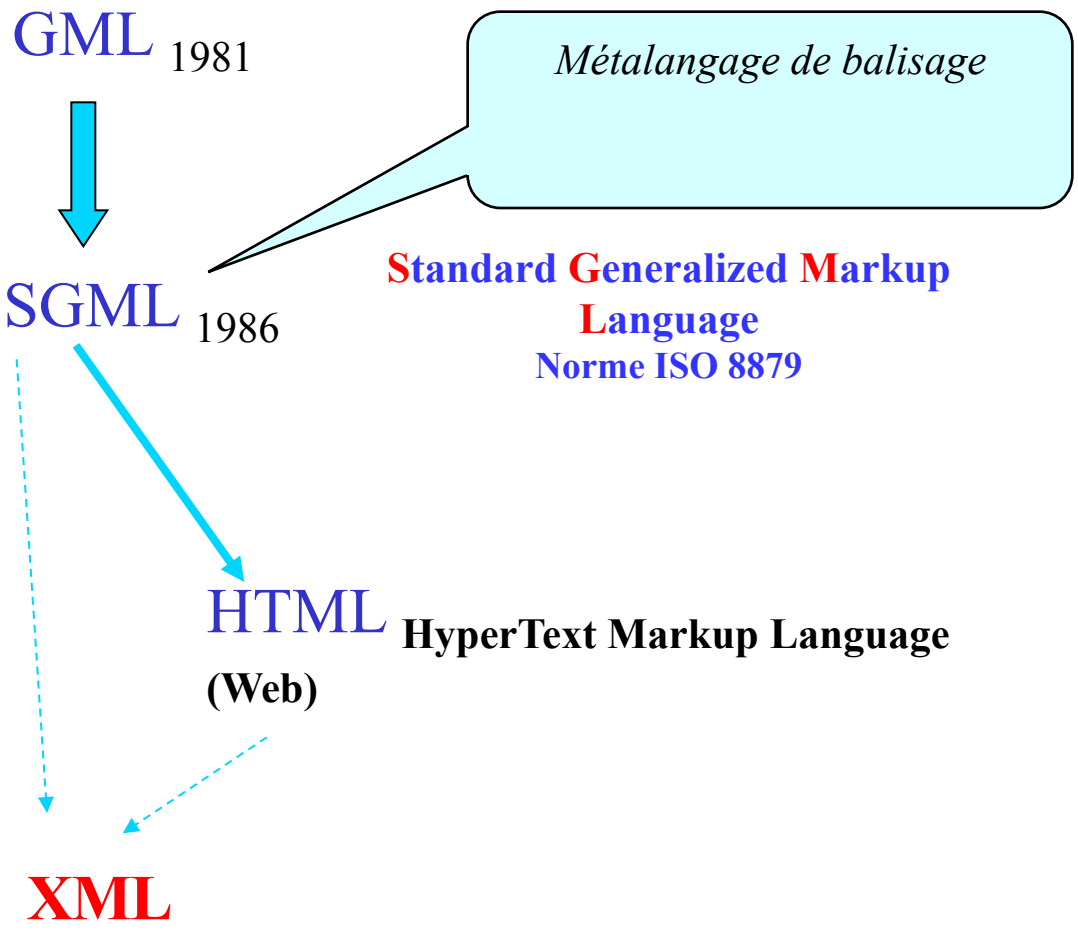
Tag, balise ou label ….. déjà vus

- Pages HTML
  - Lecture avec Browser HTML (Netscape, Internet Explorer)
  - Lecture avec Bloc Notes de Microsoft exemple_balisé.txt

- HTML (Hypertext Markup Language)
  - Langage de balisage basé sur un ensemble prédéfini et limité de balises surtout de présentation, défini par une norme (HTML 2.0, 3.2, 4.0).
  - Sémantiques des balises :
    - `h1`,..,`h6`, `title`, `address`, … donnent des indications structurelles
    - `center`,`hr`,`b`,`i`,`big`,`small`,… ne servent qu'à décrire une mise en page

XML= structurer un document avec ses propres balises

Historique



XML 1.0 -10/02/1998

XML – eXtensible Markup Language

- XML est un Langage de marquage (balisage) extensible
  - «Langage», « format » universel pour la description et la structuration des documents
  - Une version simplifiée de SGML (ISO 8879)

- Pas de collections de « balises prédéfinies ». XML permet aux concepteurs de documents de définir
  - leurs propres marqueurs (balises)
  - la structure de ces balises à travers la notion de DTD

- Un modèle de données fondé sur des arbres

Remarques

- XML fournit une *syntaxe*, pas de *sémantique* « a priori »
- Les balises n'ont pas de présentation ou de signification définie par le langage mais elles peuvent bien sûr avoir un sens pour les applications
  - <nom> Georges </nom>
  - <matière> Georges </matière>
- XML ne définit que la structure et le contenu d'un document, pas son comportement

Remarques

- Développement et promotion par W3C
  - Industriels: tous les poids lourds, notamment Oracle, IBM, Compaq, Xerox, Microsoft, etc..
  - Laboratoires de recherche: MIT (représentant les US), INRIA (Europe), université Keio au Japon (Asie)

    - Tout savoir sur XML World Wide Web Consortium.htm

- XML a pour objectif de standardiser la manière dont l'information est :
  - Échangée (XML)
  - Personnalisée (XSL)
  - Retrouvée (XQuery)
  - Sécurisée (Encryption, Signature)
  - Liée (XLink)
  - ...

Avantages : Echange et partage d'information

- En XML, une communauté d'auteurs invente librement les balises qui lui paraissent utiles pour représenter les informations qu'ils comptent échanger ou partager

- Exemple: diverses façons de représenter une date
  - `<date> 5 Janvier 2000 </date>`
  -
```
<date>
   <a>2000</a><m>01</m><j>05</j>
</date>
```
  - `<date format='ISO-8601'> 2000-01-05 </date>`

Avantages : Modularité et réutilisation

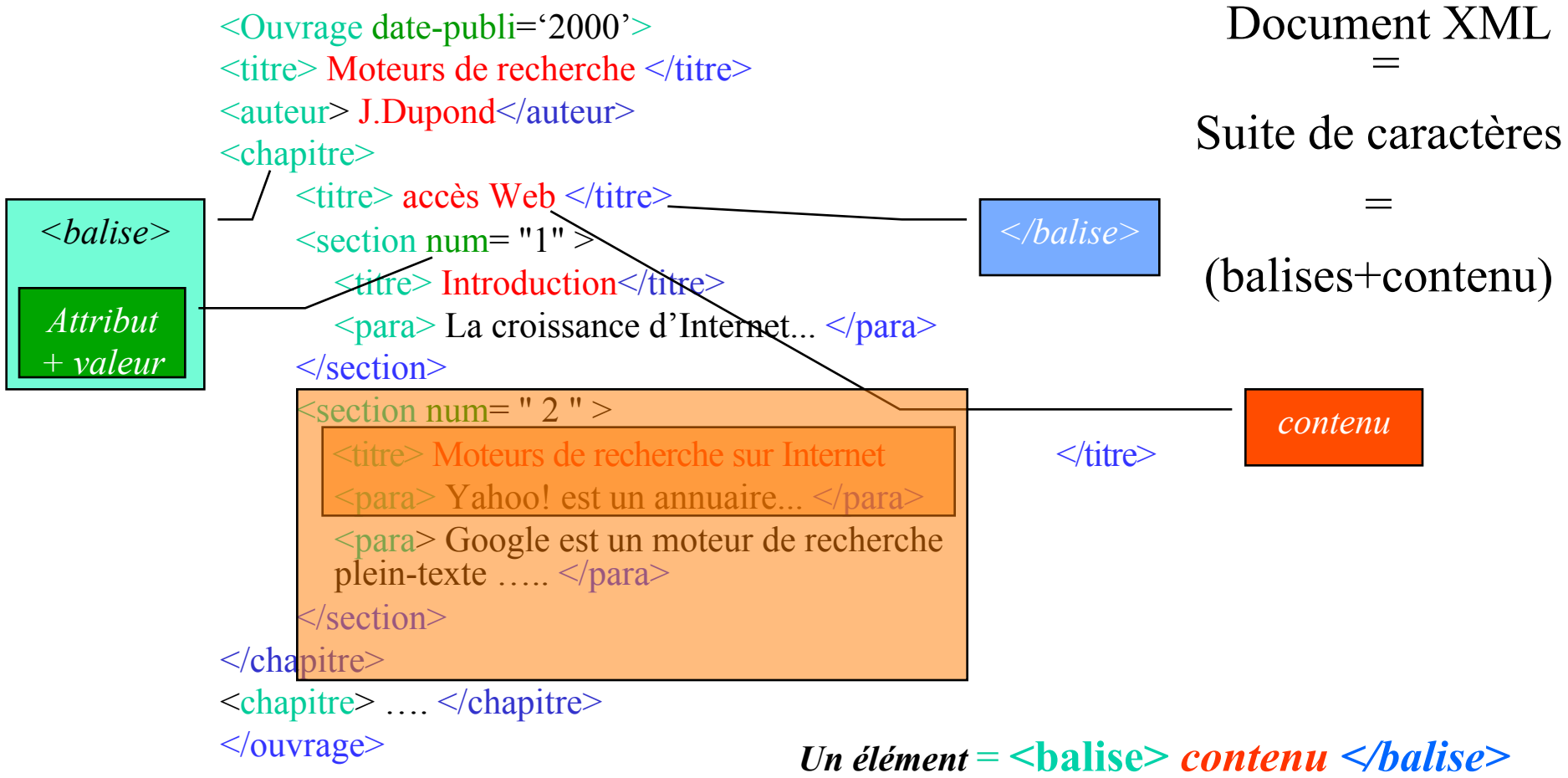
- Chaque utilisateur est libre de définir ses propres structures de document
- Il peut aussi se conformer à des structures types, appelées DTD
- Chaque communauté peut ainsi proposer des structures normalisées
- La conformité à une DTD permet l'automatisation des traitements et assure une possibilité de contrôle de validité

Avantages :
Accès à des sources d'information hétérogènes

- L'interrogation et l'échange de données entre systèmes d'information hétérogènes est souvent complexe
- XML contribue à résoudre ce problème
  - format d'échange normalisé indépendant de toute plateforme
- L'indexation et l'interrogation de grosses bases documentaires
  - informations structurelles en plus d'informations textuelles.

Chapitre 2 :
Structure d'un document XML

Document XML : définition



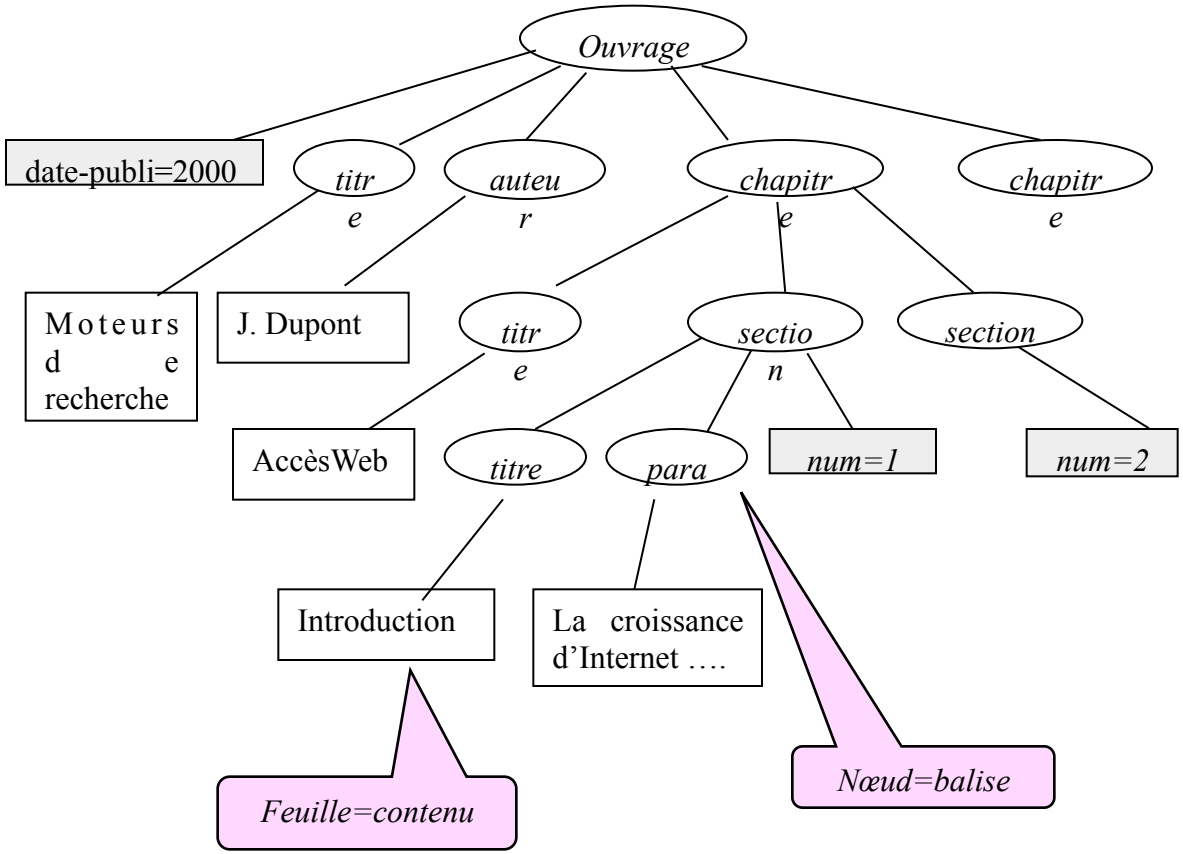
Document XML : définition

- Les documents XML forment un **arbre**
- Chaque nœud de l'arbre possède un nom
- Les règles de description peuvent être définies au sein de **DTD** (Definition de Type de Document)

Représentation sous forme d'arbre

```
….
!-- element racine -->
<ouvrage date-publi='2000'>
<!-- enfants -->
<titre> Moteurs de recherche </titre>
<auteur> J.Dupond</auteur>
<chapitre>
        <titre> accès Web </titre>
        <section num= "1" >
            <titre> Introduction </titre>
            <para> La croissance
        d'Internet... </para>
        </section>
<section num= "2" >…
</section>
</chapitre>

<chapitre> …. </chapitre>
</ouvrage>
```



Structure d'un document XML

- Un document XML se compose
  - d'un *prologue,* éventuellement vide

```
<?xml version="1.0" standalone="yes" ?>
```

  - d'un *arbre d'éléments (avec une racine)*

```
<ouvrage> <titre> Moteurs de recherche </titre>
          <auteur> J.Dupond</auteur>
          <chapitre>
          <section>  <para> La croissance d'Internet... </para> </section>
          </chapitre>
</ouvrage>
```

  - de *commentaires* et *d'instructions de traitement,* facultatifs

Exemple XML

```
<!-- Prologue -->
<?xml  version="1.0" encoding="UTF-8" standalone="yes"?>
<!--DOCTYPE ouvrage SYSTEM "doc1.dtd " -->

<!-- element racine -->
<ouvrage date-publi='2000'>

<!-- Premier enfant -->
<titre> Moteurs de recherche </titre>
<auteur> J.Dupond</auteur>
<chapitre>
      <titre> accès Web </titre>
      <section num= "1" >
         <titre> Introduction </titre>
         <para> La croissance d'Internet... </para>
      </section>
      </chapitre>
<chapitre> …. </chapitre>
</ouvrage>
```

Prologue contient (1)

- Une déclaration XML= Spécification (optionnelle)
  - indique la version d'XML utilisée, le type d'encodage utilisé, liens vers d 'éventuelles ressources externe

```
<?xml  version="1.0"?>
<?xml  version="1.0" encoding="UTF-8"?>
<?xml  version="1.0" encoding="UTF-8" standalone="no"?>
```

  - *version* : version XML utilisée dans le document, 1.0 ;
  - *encoding* : le codage utilisé, le jeu de caractère standard pour la France est le *ISO-8859-1*. Par défaut, l'attribut encoding a la valeur UTF-8. ;
  - *standalone* : dépendance du document par rapport à une DTD .
    - Standalone="yes", autonome, pas de DTD externe,
    - Standalone="no", le processeur attend une référence à une DTD.
    - La valeur par défaut est "no".

Prologue contient (2)

- Une *déclaration de type de document,* facultative

```
<!DOCTYPE ouvrage SYSTEM "biblio.dtd" [ déclarations ]>
```

  - indique la DTD à laquelle doit se conformer un document

Les éléments

- Un document XML est structuré selon une **arborescence** d'éléments (ou balises)
- Un document XML possède **un** élément "**racine**" dont lequel sont imbriqués tous les autres éléments
- Un élément est défini par un **nom** et décrit un **contenu**
- Un contenu est délimité par un **élément de début** et un **élément** (optionnel) **de fin**

Syntaxe:

- **<NomElement>** *contenu* **</NomElement>**
- **<NomElement/> (élément vide)**

Exemples:

- **<adresse>** 118, Route de narbonne **</adresse>**
- **<vrai/>**

Contrainte sur les noms des éléments

- Un nom d'élément ou d'attribut est une suite non vide de caractères pris parmi
  - les *caractères alphanumériques;* le tiret-souligné (*underscore*); le signe *moins;* le *point*;
- qui doit satisfaire les contraintes suivantes
  - le premier caractère doit être alphabétique ou un tiret-souligné
  - les trois premiers caractères ne doivent pas former une chaîne dont la représentation en lettres minuscules est "xml".
  - distingue majuscules et minuscules

Exemples de noms d'éléments	
corrects	incorrects
`_toto`	`1998-catalogue`
`Nom_société`	`XmlSpécification`
`xsl:rule`	`nom société`
`X.11`	

Les attributs

- Tous les éléments peuvent contenir un ou plusieurs attributs.
- Un attribut est composé d'un nom et d'une valeur.
  - <section num="1"> Les moteurs de recherche </section>
- Il ne peut être présent **que** dans la balise *ouvrante* de l'élément
  - Exemple :
  - on n'a pas le droit d'écrire </livre lang="en">

Entités

- Entités : alias qui servent à établir un lien entre un nom symbolique et un texte de remplacement ou un pointeur vers une ressource externe. Elles peuvent être:
  - Entités prédéfinies: Les caractères : <, >, &, ‘, “ ne peuvent être utilisés dans le texte, car utilisés dans le balisage

Entité	Valeur	Exemple	Résultat analysé
<	less than (<)	10 < 100	10<100
>	greater than (>)	x > 119	x>119
&	ampersand (&)	AT&T	AT&T
'	apostrophe (‘)	d'autres	d’autres
"	quote(")	"Wow!"	"Wow!"

  - Entités caractères : Il n'existe pas d'entité prédéfinie pour les lettres accentuées. Il faut utiliser à la place les entités numériques du type &#n; (où n est une valeur décimale). La valeur numérique correspond au code ISO 10646 ;
    - exemple *é* est codé par l'entité numérique é.
  - On peut définir ses propres entités. Il est possible d'importer des entités en provenance d'une autre DTD. On parle d’entités générales ou paramètres (voir chapitre DTD)

Commentaires

- En XML, les commentaires se déclarent de la même façon qu'en HTML.
- Ils commencent donc par **<!--** et se terminent par **-->**.
- Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'*extérieur* d'une autre balise.

- Exemples de commentaires valides :
  - <!-- ceci est correct -->
  - <elt> <!-- ceci est correct aussi -->
    Un peu de texte </elt>

Les sections CDATA

- Sections ***CDATA***: “protection” des informations pour un analyseur
  - peut contenir toute sorte de chaîne de caractères.
  - permet de définir un bloc de caractères ne devant pas être analysés par le processeur XML
  - permet entre autres de garder dans un bloc de texte un exemple de code à afficher tel quel

Syntaxe:

```
<![CDATA[ … contenu … ]]>
```

Exemple:

```
<![CDATA[ IF A<B THEN PRINT A+B ]]>
```

<![CDATA[Une balise commence par un < et se termine par un >.]]>

Exercices

- Construire un document XML qui permet de structurer une lettre.
  - Infos destinataire: nom, prénom, adresse (numéro, rue, ville, code postal, pays)
  - Corps du message

Chapitre 3 :
DTD (Document Type Definition)
(Déclaration Type de Document)

DTD
(Déclaration Type de Document)

- Permet de définir le «vocabulaire (les balises, les éléments) » et la structure qui seront utilisés dans le document XML
- Grammaire du langage dont les phrases sont des documents XML (instances)
- Elle peut être absente, ou stockée dans deux endroits différents.
  - incorporée au document XML (elle est alors dite *interne*)
  - ou bien mise dans un fichier à part (on parle alors de DTD *externe*).
- Lorsqu'il existe une DTD interne et une DTD externe: elles se combinent, la DTD externe complète la DTD interne
- La déclaration se place juste après le prologue du document.

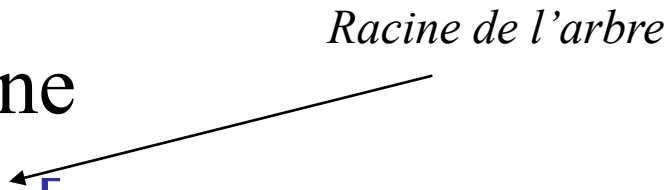
Exemple de DTD

```
<!DOCTYPE ouvrage [
<!-- La DTD est décrite ici ..>

<!ELEMENT ouvrage ( titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ATTLIST ouvrage date-publi CDATA #REQUIRED>
...
]>
```

Déclaration d'un Type DTD interne

- La déclaration est faite à l'intérieur du document XML
- Déclaration DTD interne




```
<!DOCTYPE ouvrage [
Déclarations des éléments
..
]>
<ouvrage> …..</ouvrage>
```

Exemple DTD interne

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE ouvrage [
<!-- La DTD est décrite ici ..>
<!ELEMENT ouvrage (titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
…]>

<ouvrage>
<titre> Moteurs de recherche</titre>
<auteur> J.Dupond</auteur>
<chapitre>
    <titre> accès Web </titre>
    <section >
      <titre> Introduction</titre>
      <para> La croissance d'Internet... </para>
    </section>
</ouvrage>
```

Déclaration d'un Type DTD externe

- Peut être privée ou publique.
  - privée est accessible uniquement en local (sur la machine de développement),
    - Exemple déclaration : <!DOCTYPE biblio SYSTEM "biblio.dtd">
  - publique est disponible pour tout le monde
    - Exemple déclaration <!DOCTYPE biblio PUBLIC "http://www.e-xmlmedia.com/doc.dtd">

Élaboration d'une DTD : Déclaration d'un élément

- <!ELEMENT Tag Type_element >
  - Tag : nom élément,
  - Type_element = modèle de contenu
  - modèle de contenu peut être
    - Texte brut (#PCDATA)
    - Texte vide (EMPTY),
    - Séquence
    - Choix d'éléments.

Le modèle de contenu “texte brut”: #PCDATA

- **#PCDATA**: *Parsed Character DATA* (données textuelles analysables), i.e., du texte sans aucun balisage

Syntaxe: **<!ELEMENT** nomElement **(#PCDATA)>**

Exemple: <!ELEMENT titre (#PCDATA)>

Document XML: <titre> accès Web </titre>

Le modèle de contenu : ANY et EMPTY

- **EMPTY**: un élément possédant un tel modèle de contenu ne doit posséder **aucun contenu**

Syntaxe: `<!ELEMENT nomElement EMPTY>`

Exemple:
```
<!ELEMENT null EMPTY>
```

Document XML:
```
<null/>
```

- **ANY**: modèle de contenu pouvant être n'importe quel autre **élément**, ou bien du **texte**, ou même encore **vide**

Syntaxe: `<!ELEMENT nomElement ANY>`

Exemple:
```
<!ELEMENT NimporteQuoi ANY>
```

Document XML:
```
<NimporteQuoi> <b1>du texte ...<b1><b2/> </NimporteQuoi>
```

Les modèles de contenu : Séquences

- Séquence d'éléments : une liste ordonnée des éléments qui doivent apparaître en tant qu'éléments-enfants de l'élément que l'on est en train de définir.

<!ELEMENT ouvrage (titre, auteur, chapitre)>

Exemple d'utilisation valide

```
<ouvrage>
<titre> …</titre>
<auteur> …</auteur>
<chapitre> …</chapitre>
</ouvrage>
```

Exemple d'utilisation non valide

```
<ouvrage>
<titre> …</titre>
<chapitre> …</chapitre>
</ouvrage>
```

Manque auteur

Les modèles de contenu : alternative d'éléments

- Alternative : choix dans une liste de plusieurs éléments possibles

```
<!ELEMENT biblio (ouvrage | article )>
```

Exemple d'utilisation valide

```
<biblio> <ouvrage> …
            </ouvrage>
</biblio>
```

```
<biblio> <article> …
            </article>
</biblio>
```

Exemple d'utilisation non valide

```
<biblio>
<ouvrage> … </ouvrage>
<article> … </article>
</biblio>
```

Les modèles de contenu : Indicateurs d'occurrence

- Lors de la déclaration de séquence ou de choix d'éléments, une indication d'occurrence peut être attribuée à chaque élément enfant
  - aucun indicateur: l'élément doit apparaître une et une seule fois
  - **?**: l'élément (ou groupe d') peut apparaître une fois ou pas du tout
  - *: l'élément (ou groupe d') peut apparaître zéro, une ou plusieurs fois
  - +: l'élément (ou groupe d') peut apparaître une ou plusieurs fois

<!ELEMENT LIVRE (TITRE, AUTEUR+, RESUME?, CHAPITRE*)>

Exemple <!ELEMENT biblio (ouvrage | article)*>

Le document précédent devient valide

<biblio>
<ouvrage> … </ouvrage>
<article> … </article>
</biblio>

Les modèles de contenu : enchaînements d'éléments

- enchaînement d'éléments décrivant une **structure arborescente.** La description d'une telle structure nécessite que le modèle de contenu des éléments soit un (ou plusieurs) autre(s) élément(s).

```
<!ELEMENT ouvrage ( titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT para (#PCDATA)>
```

```
<ouvrage >
<titre> Moteurs de recherche </titre>
<auteur> J.Dupond</auteur>
<chapitre>
<titre> accès Web </titre>
<section >
 <titre> Introduction </titre>
 <para> La croissance d'Internet... </para>
</section>
</chapitre>
</ouvrage>
```

Modèle de contenu : mixte

- Apparaissent comme un choix (“|”) d’éléments uniques incluant des #PCDATA associés à l’indicateur d’occurrences “*”
- Un seul niveau de description
- Dans un document, n’importe quel élément d’un tel modèle de contenu peut apparaître *n* fois dans n’importe quel ordre

Syntaxe déclaration DTD: **<!ELEMENT** nomElement **(#PCDATA |** element | element | **...)*>**

Exemple de déclaration DTD: <!ELEMENT PHRASE (#PCDATA | CITATION | NOM)*>

Document XML:

```
<PHRASE>
   Comme le disait si bien <NOM>Tim Bray</NOM><CITATION>...</CITATION>
</PHRASE>
```

Déclarations des ATTRIBUTS (1/6)

- Permettent d'ajouter de l'information aux éléments (e.g., lier un élément avec un autre élément)

- Déclaration
  - <!ATTLIST nom-element [attribut type #option ["valeur_par_défaut"]]* >

Déclarations des ATTRIBUTS (2/6)

- Type
  - CDATA: chaîne de caractères entre " "
  - Ensemble de valeurs (val1| val2|…)
  - NMTOKEN / NMTOKENS: chaîne respectant les règles de formation des noms d'objets en XML
  - ENTITY / ENTITIES: nom qui doit se retrouver dans une déclaration d'entité <!ENTITY….>
  - ID – IDREF –IDREFS : référencement interne XML

    ID: création IDREF: utilisation
  - NOTATION: nom symbolique faisant référence à une déclaration <! NOTATION….>
  -
- Option
  - **#REQUIRED** Valeur de l'attribut obligatoire
  - **#IMPLIED** Valeur de l'attribut optionnelle
  - **#FIXED** Valeur implicite de l'attribut est fixée *a priori*

Exemples de déclarations (3/6)

- Valeur par défaut
  - Exemple de DTD
    - <ELEMENT editeur #PCDATA>
    - <!ATTLIST editeur ville CDATA "Paris">
  - Exemple XML
    - <editeur ville="Paris"> …</editeur>
- Valeur optionnelle (#IMPLIED)
  - Exemples DTD
    - <!ATTLIST editeur ville CDATA #IMPLIED >
  - Exemple XML
    - <editeur ville="Toulouse"> …</editeur>
    - <editeur > …</editeur>

Exemples de déclarations (4/6)

- Valeur obligatoire (#REQUIRED)
  - Exemples DTD
    - <!ATTLIST editeur ville CDATA #REQUIRED >
  - Exemple XML
    - <editeur ville="Toulouse"> …</editeur>
    - <editeur > …</editeur> écriture incorrecte
- Valeur Fixe (#FIXED)
  - Exemples DTD
    - <!ATTLIST université nom CDATA #FIXED "UPS" >
  - Exemple XML
    - <Université nom="UPS"> …</université>
    - <editeur > …</editeur> écriture incorrecte

Exemples de déclarations (5/6)

- Attributs énumérés(val1 | val2…)
  - Exemples DTD
    - <!ATTLIST paiement type (CB | chèque) "chèque">
  - Exemple XML
    - <paiement type="CB"> …</paiement>

- Autres exemples
  - Exemples DTD
    - <!ATTLIST auteur genre (M | F)
      pays CDATA #IMPLIED > (mettre une val par défaut)
  - Exemple XML
    - <auteur genre="M" pays="USA"> Dan Brown </auteur>

Exemples Déclarations (6/6)

Exemple

<!DOCTYPE **ouvrage** [
<!-- La DTD est décrite ici ..>

<!ELEMENT ouvrage (titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur(#PCDATA)>
<!ELEMENT para(#PCDATA)>
<!ATTLIST ouvrage date-publi CDATA #REQUIRED>
<!ATTLIST section num CDATA #REQUIRED>
]>

Déclarations des entités

- Les entités permettent de référencer :
  - soit un nom de variable utilisable dans le document (**entité générale**)
  - ou un alias utilisable uniquement dans la DTD (**entité paramètre**)
- Les entités peuvent être **internes** déclarées dans le document ou **externe** font référence à une ressource externe
- Il existe aussi des entités prédéfinies et caractères (voir chapitre 2)

Exemple DTD interne

Stocké dans un même fichier

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE ouvrage [
<!-- La DTD est décrite ici ..>
<!ELEMENT ouvrage (date-publi, titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (num, titre, para+)>
…]>

<ouvrage>
<Date-publi="2000">
<titre> Moteurs de recherche</titre>
<auteur> J.Dupond</auteur>
<chapitre>
    <titre> accès Web </titre>
    <section num= "1" >
      <titre> Introduction</titre>
      <para> La croissance d'Internet... </para>
    </section>
</ouvrage>
```

Exemple de document avec DTD externe

```
<?XML version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ouvrage SYSTEM "biblio.dtd">

<ouvrage date-publi='2000'>
<titre> Moteurs de recherche </titre>
<auteur> J.Dupond</auteur>
<chapitre>  <titre> accès Web </titre>
          <section num= "1" >
          <titre> Introduction </titre>
          <para> La croissance d'Internet... </para>
      </section>
      </chapitre>
<chapitre> ….
</chapitre>
</ouvrage>
```

```
<!-- La DTD est décrite ici -->
<!ELEMENT ouvrage ( titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section ( titre, para+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ATTLIST ouvrage date-publi CDATA
     #REQUIRED>
<!ATTLIST section num CDATA #REQUIRED>
```

DTD enregistrée dans le fichier biblio.dtd

Validité des documents

- Document bien formé (Well Formed document)
  - balises correctement imbriquées
  - « parsable » et manipulable
  - pas nécessairement valide par rapport à la DTD

- Document valide (Valid document)
  - bien formé + conforme à la DTD

Analyseurs

- **Analyseur (parser) XML**: outil de lecture et d'analyse de structures de documents XML
  - vérification de la structure et de la validité
  - extraction des données
- Différents types d'analyseurs
  - analyseurs validant/non-validant
  - analyseurs supportant l 'API **DOM**
  - analyseurs supportant l 'API **SAX**
  - autres ...