

# TECHNOLOGIES

## XML

### EXTENSIBLE MARKUP LANGUAGE

#### PARITE1

# Objectifs

2

- Comprendre l'utilisation XML dans le contexte d'une application Web.
- Apprendre la syntaxe XML et les modèles sous-jacents.
- Etudier et mettre en œuvre quelques outils (langages) pour la manipulation de XML.

# Plan

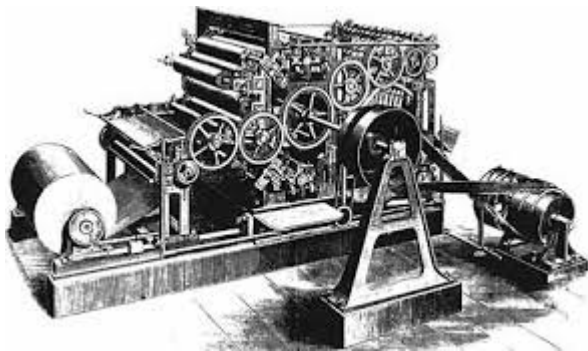
3

- Introduction
- Définition de Document Type(DTD)
- Schéma XML
- XPATH
- Le langage de style XSLT
- Programmation XML(DOM & SAX)
- XQUERY

# eXtensible Markup Language

4

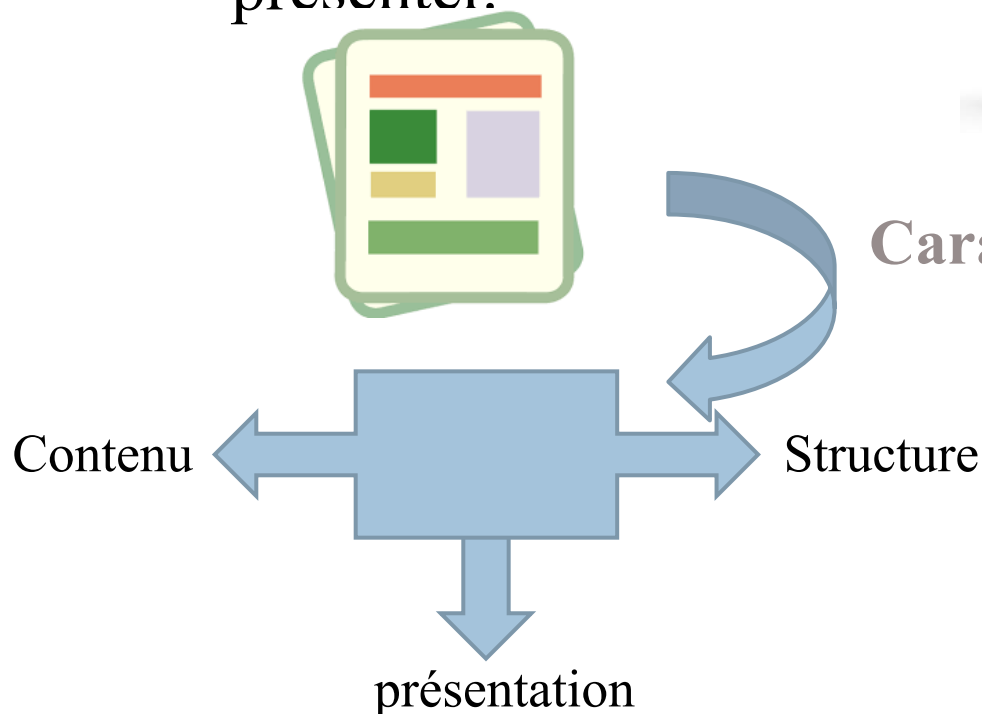
- *Markup* signifie à l'origine les symboles que l'imprimeur utilise pour les corrections de mise en page et d'affichage:
  - ▣ marques, annotations manuscrites placées sur un document pour préciser à l'imprimeur comment il doit être présenté





Les documents électroniques prenaient de plus en plus de l'ampleur.



- ▣ Usage des macros spécifiques pour pouvoir les présenter.



Markup

# Historique

6

- en 1967 William Tunncliffe a proposé pour la première fois une approche pour séparer le fond du document par rapport à sa forme **GenCode**
  - ▣ propres notations de contrôle, souvent spécifiques à l'appareil cible
- en 1969 des chercheurs de IBM(C.Goldfarb, E.Mosher et R.Lorie) ont présenté **GML** (*Generalized Markup Language*) basé sur le Gencode
  - ▣ faciliter l'échange d'informations entre les outils d'édition, de formatage et de recherche de documents
    - Le GML possédait déjà un concept de « type de documents » similaire aux documents DTD (Data Type Definition)

# Historique(suite)

7

- en 1974 Goldfarb a présenté SGML (Standard Generalized Markup Language)
  - ▣ publié en 1986 comme norme ISO (ISO 8879:1986).
  - ▣ est un langage de description de données qui divise un document en deux parties : la DTD (Data Type Definition) et les données elles-mêmes.
  - ▣ c'est un standard ouvert indépendant de toute plateforme ou constructeur.
    - Les fichiers SGML sont stockés sous forme de textes ASCII et peuvent donc être utilisés par n'importe quelle machine.

# SGML

8

- SGML par sa performance a été utilisé par les militaires américains , le secteur aéronautique et par d'autres domaines nécessitant la gestion de documentations
- SGML sépare le contenu, la présentation et la structure d'un document
  - ▣ gère les présentations avec possibilité de multiples présentations et plusieurs formats , l'échange des documents et la garanti de la pérennité



# SGML critiques

9

- ❑ Très lourd et complexe
- ❑ Une grande rigueur est demandée à l'entrée des documents
- ❑ Liens hypertextes possibles mais complexes
  - Réservé aux professionnels de la documentation
  - Difficilement extensible au Web



# HTML

- Avec le développement de l'Internet et pour la présentation des pages Web une application très légère du SGML a vu le jour appelée HTML (développé par Tim Berners-Lee) en 1990.

## Avantages :

- Simple
- Intègre les liens hypertexte d'une manière facile
- facile d'intégrer la DTD HTML dans les navigateurs (Netscape, Explorer...)

## Limites:

- Langage non extensible
    - figé (avec un nombre limité de balise)
  - La sémantique du contenu est perdue. On ne sait pas interpréter des données fournies en HTML
- ➔ C'est un langage de balisage de présentation et non de contenu

# Nouvelles utilisation du Web

11

## □ Quelques applications :

- ▣ **Commerce électronique et échange de données informatisées (EDI):** les entreprises veulent échanger des informations et les intégrer dans leurs propres systèmes d'information
- ▣ **Moteur de recherche:** besoin d'interprétation des données transmises pour les indexer efficacement
- ▣ **Gestion électronique de documents (GED):** besoin d'indexer, d'organiser et de structurer le contenu des documents accessibles

# *Nouveaux besoins*

12

- les données doivent être représentées indépendamment d'une machine donnée ou d'une application
- Besoin de transformer facilement les données d'un format à un autre
- ...



- HTML est un langage pour présenter des informations à l'écran. il ne permet pas l'échange et le traitement (autre que l'affichage) des données.
- SGML est très complexe pour l'utiliser sur internet



XML

XML n'est pas le remplaçant de HTML:

- XML sert à structurer l'information
- HTML a pour but de la présenter

# Pourquoi XML?

14

- **Un document XML est un format orienté texte** : il circule facilement
- **XML n'est pas lié à un mode d'utilisation** : C'est un métalangage et chacun peut définir son propre langage
  - ▣ SMIL(Synchronized Multimedia Integration Language): Permet l'intégration de données multimédias (image, vidéo, audio,...) en décrivant le déroulement temporel et spatial des différents composants intégrés
  - ▣ SVG (Scalable Vector Graphic) :utilisé pour décrire des graphiques en 2 dimensions
  - ▣ WML (Wireless Markup Language ):conçu spécifiquement pour le WAP (protocole pour accès à Internet via un appareil de transmission sans fil) afin de pouvoir afficher sur un écran de téléphone mobile
- **Beaucoup d'outils de manipulation** : DOM, SAX, XSLT, XPath, XQuery, etc

# Exemples de domaine d'utilisation

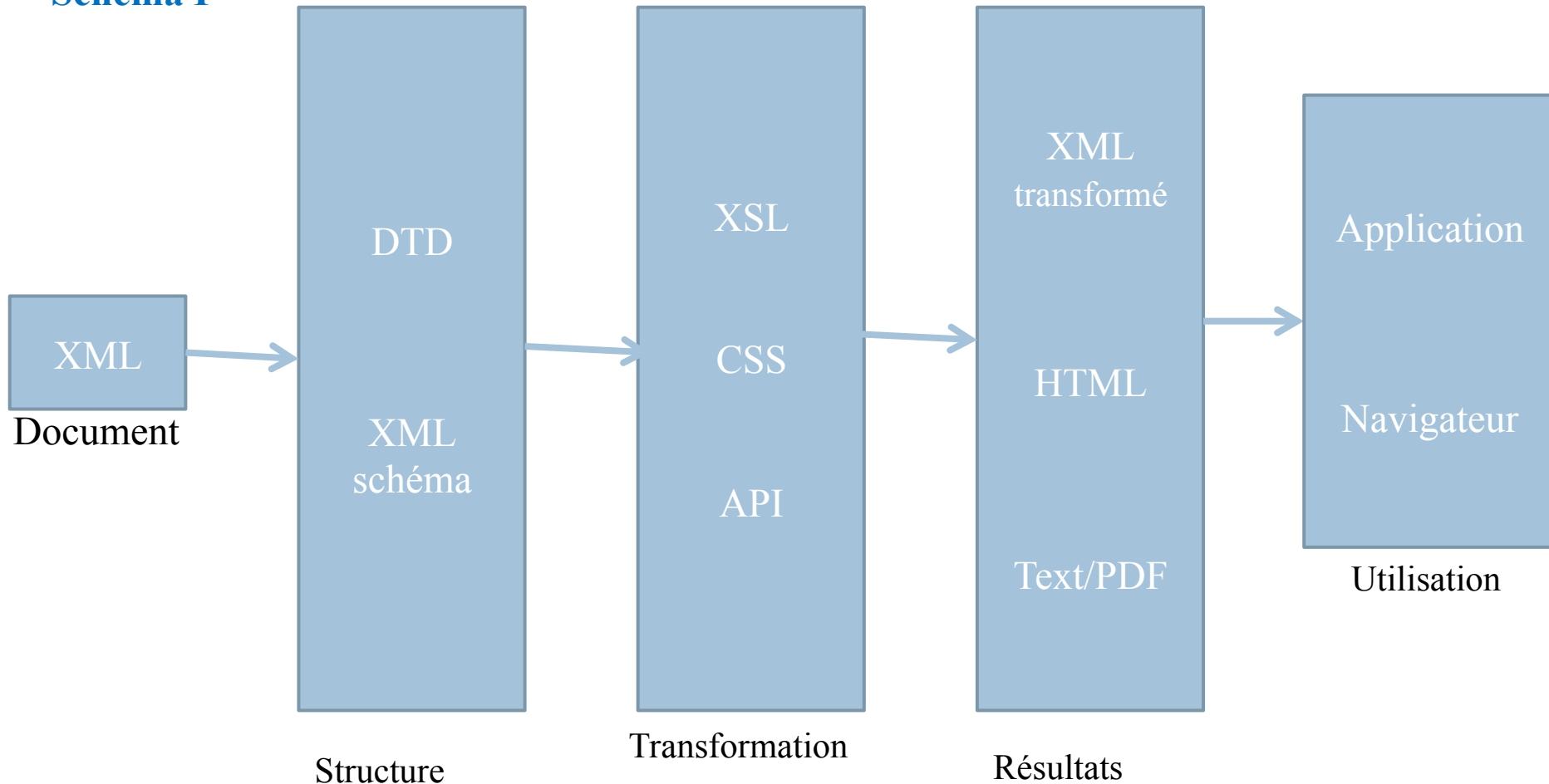
15

- Multimédia (MPEG 7 est basée sur XML)
- Intégration d'applications d'entreprise (EAI) avec la transformation des données vers XML (adaptateurs XML)
- Web Services : ensemble de protocoles et de normes informatiques utilisés pour échanger des données entre les applications
  - ▣ Le langage WSDL : description des services en XML
  - ▣ Le protocole SOAP : format des messages échangés en XML
- Web Sémantique: ensemble de technologies et de normes utilisés pour rendre le contenu Web accessible et utilisable par des logiciels
  - ▣ RDF (Resource Description Framework) qui est un modèle de description des ressources Web et leurs métadonnées afin de permettre le traitement automatique de telles descriptions
  - ▣ OWL (Web Ontology Language) : C'est un formalisme basé sur RDF qui fournit des moyens pour définir des ontologies Web structurés

# Cycle de vie d'un document XML

16

Schéma I





# Structure d'un document XML

# Structure d'un document XML

18

<code>&lt;?xml ... ?&gt;</code>	<code> </code>	Prologue
<code>...</code>	<code>]</code>	
<code>&lt;root-element&gt;</code>	<code> </code>	
<code>...</code>	<code> </code>	Corps
<code>&lt;/root-element&gt;</code>	<code>]</code>	

# Exemple XML

19

```
<?xml version="1.0" encoding="iso-8859-1"?>  
<!-- "Ceci est un commentaire" -->  
<!DOCTYPE bibliographie SYSTEM "bibliography.dtd" >  
<?xml-stylesheet type="text/xml" href="Biblio.xsl"?>  
  
<bibliographie>  
  <livre key="Michard01" lang="fr">  
    .....  
  </livre>  
  <livre key="Zeldman03" lang="en">  
    .....  
  </livre>  
  ...  
</bibliographie>
```

**Commentaire**

**Déclaration de DTD**

**l'élément racine**

**Instructions de traitements**

**Attribut d'un élément**

**Prologue**

**Corps du document**

# Le prologue: entête

20

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

- Version="1.0" : version de la spécification XML utilisée.  
Il existe aussi la version 1.1, mais la version 1.0 est la plus utilisée;
- Le jeu de caractères (*encoding*)  
EX:
  - ▣ ISO-8859-1: convient pour la plupart des langues latines ou occidentales (anglais, français, allemand, espagnol...)
  - ▣ UTF-8 : indispensable pour les autres langues (japonais...)
- **S**andalone: faire savoir à l'avance à l'analyseur syntaxique XML si le document est autonome ou pas.
  - ▣ c'est-à-dire s'il existe des déclarations externes qui affectent le document (DTD externe, Entité paramétré...)
    - Standalone= "yes" indique au parseur que le document est indépendant
    - Standalone= " no " indique qu'il y a des déclarations de balisage externes.
- Le prologue peut contenir d'autre éléments: les commentaires, la déclaration de la DTD et les instructions de traitement

# Le prologue : Commentaire

21

## □ Commentaires

- ▣ Ils se positionnent n'importe où dans le corps du document XML où dans le prologue.

**<!-- Ceci est un commentaire -->**

- ▣ Un commentaire ne doit pas précéder le prologue
- ▣ Un commentaire ne peut être placé à l'intérieur d'une balise
- ▣ Il ne doit pas avoir de « -- » dans le texte du commentaire
- ▣ Un commentaire ne doit pas apparaître dans une balise CDATA

# Le prologue: Instruction de traitement

22

- Les instructions de traitement (PI: processing instructions) permettent de transmettre des instructions aux applications qui traitent les documents XML.
- Ils sont délimités par les chaînes de caractères ‘<?’ ‘ et ‘?>’

**<?xml-stylesheet type="text/css" href=" myCss.css ?>**

Nom de l'IT

- Le nom de l'IT permet à l'application de déterminer si l'instruction lui est destinée

**<?robots index="yes|no" follow= ="yes|no"?>**

# Prologue: Déclaration de DTD externe

23

Déclaration de type de document (à l'aide d'un fichier annexe appelé DTD - *Document Type Definition*)

Ex:

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

# Les éléments

24

## Composants de base d'un document XML

`<title>XML langage et applications</title>`

↑  
balise ouvrante

↑  
Contenu

↑  
balise fermante

- Chaque élément peut contenir du texte simple (comme un fichier), d'autres éléments (comme un répertoire), ou un mélange des deux.
- Élément sans contenu (vide) :  
`<identification isbn="0-7357-1201-8" />`
- Toute balise rencontrée est analysée



# Attributs

25

- Contient des informations sur l'élément

`<livre key="Michard01" lang="fr">`

nom                      valeur

- Un élément peut avoir plusieurs attributs
- Les valeurs d'attributs:
  - ▣ Un nombre, une chaîne de caractères, une adresse,...

# Attributs Prédéfinis

26

Il existe en XML quatre attributs prédéfinis:

- **xml:id**: permet d'associer un identificateur à tout élément indépendamment de toute DTD ou de tout schéma.
- **xml:lang**: spécifie la langue utilisée par les données de l'élément
  - Ex: `<p xml:lang="fr">Bonjour</p>`
  - `< p xml:lang="en-US">Hi</p>`
- **xml:space** (default|preserve) : spécifie si les espaces doivent être préservés dans le contenu de l'élément

# Attributs Prédéfinis (Suite)

27

## ■ **xml:base**

- L'attribut de xml:base permet de préciser l'URI de base d'un élément
- Par défaut, l'URI de base d'un élément est hérité de son parent.
- xml:base est indispensable pour réaliser des inclusions de fichiers externes

---

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<book xml:base="http://www.somewhere.org/Teaching/index.html">
  <chapter xml:base="XML/chapter.html">
    <section xml:base="XPath/section.html"/>
    <section xml:base="/Course/section.html"/>
    <section xml:base="http://www.elsewhere.org/section.html"/>
  </chapter>
</book>
```

# Choisir élément ou attribut??

28

- ❑ Les éléments sont utilisés pour encapsuler des sous-ensembles de données,
- ❑ Les attributs sont généralement employés pour fournir des informations complémentaires sur un élément et non pour contenir des données brutes.

Utilisez les attributs si des données utilisés sont de type simple et :

- ❑ que vos informations requièrent une valeur fixe ou par défaut ;
- ❑ que vos informations requièrent des informations d'un élément existant;
- ❑ que la taille de votre fichier XML constitue un facteur critique; les attributs ont en effet généralement tendance à occuper quelques octets de moins que les éléments.
- ❑ Si on a pas de souci concernant l'ordre

# Section CDATA: section littérale


29

- CDATA signifie "Character data" = données textuelles
- Intérêt: Insérer du texte non interprété par le processeur XML
- La syntaxe:  
    <![CDATA [  
    **Texte non analysé,**  
    ]]>

NB: on ne peut pas imbriquer deux sections CDATA

# XML: Règles de syntaxe

30

- ❑ L'élément racine doit être unique;
- ❑ Le nom de l'élément commence par une lettre ou par \_
- ❑ Si le nom est composé d'un seul caractère il doit être dans [a-zA-z]
- ❑ Pas d'espaces ni de tabulations;
- ❑ Tous les éléments doivent être fermés dans l'ordre de leur ouverture;
  - ❑ <livre> <auteur> <livre/> <auteur/> : Faux 
- ❑ Les balises XML sont sensibles à la casse;
- ❑ Les valeurs des attributs doivent être entre guillemets;
- ❑ *Un attribut d'un élément ne peut avoir qu'une seule valeur;*
- ❑ Ne pas utiliser de caractères réservés à XML dans le texte du document : <, > et &; ces caractères pourront être respectivement obtenues à l'aide des entités suivants: &lt; et &gt; ; &amp; ;
- ❑ Un document XML a pour extension .xml

# Conventions de nommage

31

- Employer des minuscules pour les attributs et les éléments
- Évitez les accents dans les noms d'attributs et d'éléments
- Séparer les noms composés par `_`, `-`, `..`, majuscule

# Espace de nom: Besoin

32

□ Exemple:

<cours>

<titre> Technologie Web</titre>

<intervenant>

<nom>AZIM</nom>

<prénom>Mohamed</prénom>

<titre>Vacataire</titre>

</intervenant>

.....

</cours>



Comment distinguer  
entre les deux titres ??



# Espace de nom: Besoin

33

Mélanger plusieurs vocabulaires au sein d'un même document xml



Identifier la provenance de chaque élément et de chaque attribut pour le valider correctement



Espace de nom

# Espace de nom

34

- Les espaces de nommage XML représentent une méthode simple pour la qualification des noms des éléments et attributs utilisés dans les documents XML, en les associant à des espaces de nommage identifiés par des *adresses URI [w3C]*

# Espace de nom

35

- Pour identifier les espaces de nom, le *W3C* a utilis<sup>é</sup> les *identificateurs du Web*, c'est à dire les *URI* (*Uniform Resource Identifier*).
- ▣ Ex : URIs qui identifient des vocabulaires:
  - xhtml : <http://www.w3.org/1999/xhtml>
  - Schéma XML : <http://www.w3.org/2001/XMLSchema>
  - DocBook ( langage sémantique pour la documentation technique):  
<http://docbook.org/ns/docbook>
  - Dublin Core(schéma de métadonnées générique qui permet de décrire des ressources numériques ou physiques et d'établir des relations avec d'autres ressources): <http://purl.org/dc/elements/1.1/>

# Espace de nom: Syntaxe

36

- Espaces de noms sans préfixe:

- **xmlns**="mon\_uri"

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ma page</title>
...
</head>
</html>
```

- La déclaration de l'espace de noms par défaut se fait au moyen de l'attribut
  - Elle s'applique à l'élément dans lequel elle est effectuée et à tous ses enfants

# Espace de nom: Syntaxe

37

## □ Les espaces de noms avec préfixe

**xmlns:prefixe**="mon\_uri"

```
<cours xmlns:cours="http://www.ensias.ma/cours">
```

```
  <cours:titre>Thecnomogie Web</cours:titre>
```

```
    <intervenant xmlns:personne="http://www.ensias.ma/personnel">
```

```
      <personne:nom> AZIM </personne:nom>
```

```
      <personne:prenom> Ahmed </personne:prenom>
```

```
      <personne:titre> Vacataire</personne:titre>
```

```
    </intervenant >
```

.....

```
</cours>
```

# Espace de nom (Suite)

38

- Un espace de nom déclaré sans préfixe devient l'espace de nom par défaut de tous les éléments descendants
- Les éléments non qualifiés appartiennent à l'espace de nom le plus interne

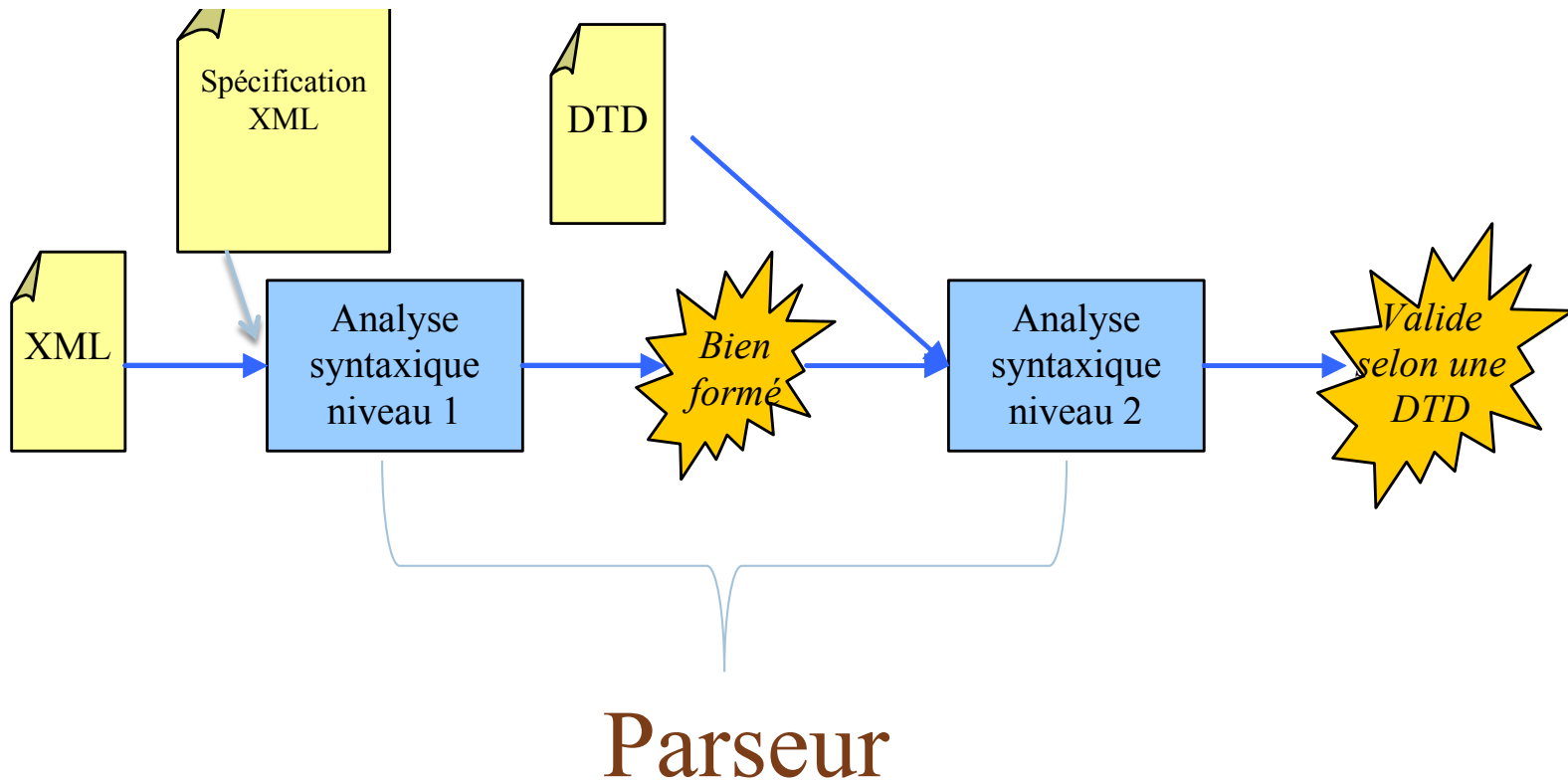
# Document bien formé et valide

39

- Le document est bien formé s'il respecte les règles de base définies dans la spécification XML.
- Le document est valide s'il est bien formé et conforme à une grammaire (DTD ou schéma)

# Validation de document

40





# Parseurs

41

- XML permet de structurer et de représenter les données
  - ▣ Il ne permet pas d'exploiter directement les données
- Un analyseur syntaxique XML (ou parseur), permet de **récupérer dans une structure** hiérarchique XML, **des balises**, leur **contenus**, leurs **attributs** pour les rendre accessibles.
  - ▣ EX: Les navigateurs intègrent un parseur → affichage du fichier xml sous forme d'un arbre
- Deux types d'approches sont possibles:
  - ▣ *Hiérarchique* : manipulation d'un document XML après l'avoir représenté en mémoire sous la forme d'un arbre d'objets (DOM)
  - ▣ *Événementiel*: manipulation d'un document XML au fur et à mesure de sa lecture, sans avoir à le charger en totalité en mémoire (SAX)
- Deux types de parseurs XML:
  - ▣ *les parseurs non validant* qui vérifient que le document XML est bien formé (respecte la syntaxe XML de base)
  - ▣ *les parseurs validant* qui permettent de vérifier la conformité d'un document XML avec la DTD ou schema XML

# Atelier 1

42

Création d'un document XML

# Grammaire XML

43

## □ **Intérêts:**

Décrire la structure d'un document XML



Définir l'ensemble des éléments et attributs à respecter par le document xml:

- Le nom des types d'éléments, leur contenu et dans quel ordre d'autres éléments peuvent y apparaître;
- Les attributs et leur valeurs par défaut;
- Le nom des entités qui peuvent être utilisées.

## □ **Approches:**

- **DTD:** plus simple et moins complète
- **XML-Schema:** plus complexe mais plus complète

# *DOCUMENT TYPE DEFINITION* DTD

# Document type Definition(DTD)

45

- C'est une grammaire qui hérite du SGML. Elle exprime les contraintes sur la structure d'un document
- Une DTD peut être :
  - ▣ une description interne au document lui-même
  - ▣ un document texte séparé, référencé par le document lui-même
- Un fichier DTD doit avoir l'extension ".dtd". Il n'utilise pas la syntaxe XML
- Trois types de déclaration d'une DTD dans le prologue

# DTD

46

## □ DTD interne:

```
<!DOCTYPE Nom_racine [  
  <!-- declarations -->  
]>
```

URI

FPI (Formal Public Identifier)

## □ DTD externe:

```
<!DOCTYPE Nom_racine SYSTEM "maDTD.dtd">
```

```
<!DOCTYPE NomEltRacine PUBLIC "-//W3C//DTD XHTML 4.0.1  
  Transitional//EN">
```

## □ DTD Mixte:

```
<!DOCTYPE Nom_racine SYSTEM "maDTD.dtd" [  
  ... déclarations ...  
>
```

# DTD Externe

47

## □ Association XML et DTD

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

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

....

```
<bibliographie>
```

...

```
</bibliographie>
```

# EXAMPLE (DTD Externe)

48

```
<!ELEMENT bibliographie (livre+)>
<!ELEMENT livre (titre, auteur, annee,
editeur, identification, url?) >
<!ELEMENT titre (#PCDATA) >
<!ELEMENT auteur (#PCDATA) >
<!ELEMENT annee (#PCDATA) >
<!ELEMENT editeur (#PCDATA) >
<!ELEMENT identification
(#PCDATA) >
<!ELEMENT url (#PCDATA) >
<!ATTLIST livre key CDATA
#REQUIRED>
<!ATTLIST livre lang CDATA
#REQUIRED>
<!ATTLIST identification isbn CDATA
#REQUIRED>
```

```
<?xml version="1.0" encoding=""iso-8859-1"?>
<!DOCTYPE bibliographie SYSTEM
"bibliography.dtd" >

<bibliographie>
<livre key="Michard01" lang="fr">
<titre>XML </titre>
<auteur>Alain Michard</auteur>
<annee>2001</annee>
<editeur>Eyrolles</editeur>
<identification isbn= "2-212-09206-7" />
<url>http://www.editions-eyrolles/livres/
michard/</url>
</livre>
...
</bibliographie>
```



# DTD interne

49

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

```
<!DOCTYPE bibliographie [
```

```
<!ELEMENT bibliographie (livre+)>
```

```
<!ELEMENT livre (titre, auteur, annee, editeur, identification, url?) >
```

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

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

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

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

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

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

```
<!ATTLIST livre key CDATA #REQUIRED>
```

```
<!ATTLIST livre lang CDATA #REQUIRED>
```

```
<!ATTLIST identification isbn CDATA #REQUIRED>
```

```
<bibliographie>
```

```
<livre key="Michard01" lang="fr">
```

```
<titre>XML langage et applications</titre>
```

```
<auteur>Alain Michard</auteur>
```

```
<annee>2001</annee>
```

```
<editeur>Eyrolles</editeur>
```

```
<identification isbn= "2-212-09206-7" />
```

```
<url>http://www.editions-eyrolles/livres/michard/</url>
```

```
</livre>
```

```
<livre>...</livre>
```

```
</bibliographie>
```

# DTD :Mixte

50

- Les définitions des éléments et des attributs figurant dans les DTD internes et externes se complètent tant qu'il n'y a pas de doublons
  - ▣ s'il y a des doublons dans plusieurs DTD, c'est la DTD interne qui est prioritaire.

# DTD : définition d'éléments

51

`<!ELEMENT nom Type_de_donnée >`

expression définissant le contenu autorisé dans l'élément



Exemple:

`<!ELEMENT livre (auteur+)>`

# DTD : définition d'éléments(suite)

52

- Le *type\_de\_donnée* : représente soit un type de donnée prédéfini, soit un élément de données composé constitué lui même d'éléments
  - ▣ Types prédéfinis :
    - ANY : L'élément peut contenir tout type de donnée
    - EMPTY : L'élément ne contient pas de données spécifiques
    - (#PCDATA) : L'élément doit contenir juste une chaîne de caractère

<!ELEMENT br EMPTY>    <br />

# DTD : définition d'éléments(suite)

53

- Déclaration d'éléments composés :
  - `<!ELEMENT bibliographies (livre+) >`

<u>élément de données composé</u>	Description
(A, B)	Séquencement d'éléments. L'élément B doit suivre A (faire attention à l'ordre)
A*	L'élément A peut apparaître Zéro ou plus
A+	L'élément A doit être présent au moins une fois
A?	L'élément A est optionnel
A B	L'élément A ou L'élément B (pas les deux)

# Example

54

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bibliographie [
  <!ELEMENT bibliographie (livre+)>
  <!ELEMENT livre (titre, auteur, annee, editeur, identification, url?) >
  <!ELEMENT titre (#PCDATA) >
  <!ELEMENT auteur (#PCDATA) >
  <!ELEMENT annee (#PCDATA) >
  <!ELEMENT editeur (#PCDATA) >
  <!ELEMENT identification (#PCDATA) >
  <!ELEMENT url (#PCDATA) >
  <!ATTLIST livre key CDATA #REQUIRED>
  <!ATTLIST livre lang CDATA #REQUIRED>
  <!ATTLIST identification isbn CDATA #REQUIRED>
]>
```

# DTD : définition d'éléments(suite)

55

- On peut avoir comme *type\_de\_donnée du contenu* mixte : mélange de données textuelles et de sous éléments
  - ▣ La seule possibilité est de combiner #PCDATA, | et \*

<!ELEMENT Adresse (#PCDATA, ville) > Faux



<!ELEMENT Adresse (#PCDATA | ville)\* >



PCDATA doit toujours être en tête  
PCDATA veut dire que du texte seul



Limite (on peut avoir + ville!!)

# DTD : Définition d'attributs

56

```
<!ATTLIST nom_element nom_attribut type_attribut  
mode_attribut |valeur_par_defaut>
```

- **nom\_element** : Nom de l'élément concerné
- **nom\_Attribut**: Nom de l'attribut
- **type\_attribut** : Le type auquel appartient l'attribut
- **mode attribut**: informe si l'attribut doit être spécifié (#REQUIRED), s'il peut être omis (#IMPLIED) ou s'il peut être fixe (#FIXED). Pour ce dernier mode on lui associe aussi une constante
- **valeur\_par\_defaut**: une valeur implicite pour l'attribut si aucune valeur ne lui a été affectée.

```
<!ATTLIST identification isbn CDATA #REQUIRED>
```



# DTD: Définition d'attributs

## Type\_Attribut

57

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

Type	Définition
<b>CDATA</b>	Des données textuelles
<b>NMTOKEN</b>	un nom XML valide
<b>NMTOKENS</b>	plusieurs noms XML séparés par des espaces
<b>(val1 val2 ...)</b>	une liste des valeurs possibles pour l'attribut
<b>ID</b>	un identificateur unique d'un élément
<b>IDREF</b>	identificateur d'un autre élément (type ID)
<b>IDREFS</b>	Liste d'identificateurs d'autres éléments
<b>ENTITY</b>	nom d'une entité prédéfinie
<b>ENTITIES</b>	Liste de noms d'entités
<b>NOTATION</b>	Permet de définir des entités externes non XML ainsi que l'application par défaut à lancer pour ouvrir ces documents

# DTD: Définition d'attributs

## Type\_Attribut(suite)

58

### □ Type chaine **CDATA**

- ▣ La valeur de l'attribut peut faire appel à une chaine de caractère.

<!ATTLIST image alt **CDATA** #IMPLIED >



Faire attention aux entités prédéfinies (ex : remplacez le signe < par l'entité &lt; voir section entité prédéfinis)

# DTD: Définition d'attributs

## Type\_Attribut(suite)

59

- Type NMTOKEN et NMTOKENS
  - ▣ La valeur de l'attribut doit être un nom XML  
**<!ATTLIST photo Nom NMTOKEN #REQUIRED>**

**<photo Nom="terre">...</photo>**

- ▣ la valeur d'un attribut de type NMTOKENS est sous forme de liste de valeurs de type NMTOKEN (nom XML) séparées par des espaces.

**<!ATTLIST photo Nom NMTOKENS #REQUIRED>**

**<photo Nom="La terre".>...</photo>**

# DTD: Définition d'attributs

## Type\_Attribut(suite)

60

- Type énuméré
  - ▣ C'est une liste de valeurs possibles pour l'attribut, chaque valeur doit correspondre à un nom XML.

```
<!ATTLIST photo type (GIF|JPEG|PNG) "GIF">
```

```
<photo type= "GIF ">....</photo>
```

# DTD: Définition d'attributs

## Type\_Attribut(suite)

61

### □ Type ID

- ▣ Ce type d'attribut permet de définir de manière unique un élément dans un document XML

Ex: **<!ATTLIST livre id ID #REQUIRED >**

- ▣ Ce type ne permet le mode\_att = #FIXED
  - En général on utilise #REQUIRED
- ▣ La valeur du ID doit être un nom XML

# DTD: Définition d'attributs

## Type\_Attribut(suite)

62

### □ Type IDREF & IDREFS

- ▣ Le type IDREF permet d'établir des liens entre éléments via les ID prédéfinis.

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

```
<!ATTLIST prof  
    id ID #REQUIRED  
    class IDREF #IMPLIED >
```

- ▣ L'attribut du type IDREFS peut prendre comme valeur une liste de valeurs d'attributs IDREF séparées par un espaces.

```
<!ATTLIST prof  
    id ID class IDREFS #IMPLIED
```

```
...
```

```
>
```

# DTD: Définition d'attributs

## Type\_Attribut(suite)

63

- Type ENTITY
  - ▣ l'attribut de ce type peut prendre comme valeur le nom d'une entité générale externe non-analysable (cf section Entité).

```
<!ENTITY Lune SYSTEM "lune.png">
```

```
<!ATTLIST image source ENTITY #REQUIRED>
```

```
<image source="Lune">
```

# DTD: Définition d'attributs

## Type\_Attribut(suite)

64

### □ Type ENTITIES

- ▣ l'attribut peut prendre comme valeur une liste des ENTITY séparées par un espace.

```
<!ENTITY Lune SYSTEM "lune.png">
```

```
<!ENTITY Soleil SYSTEM "Soleil.png">
```

```
<!ATTLIST diaporama source ENTITIES #REQUIRED>
```

```
<diaporama source=" Lune Soleil">
```



# DTD: Définition d'attributs

## Type\_Attribut(suite)

65

### □ Type NOTATION

- ▣ Permettent de spécifier une application (ou plug-in) qui traitera le type de données spécifié dans la déclaration de la Notation.

```
<!NOTATION flash SYSTEM "/usr/bin/flash.exe">
```

//déclaration de notation associant le format nommé flash à l'application spécifiée

```
<!ENTITY animation SYSTEM "../anim fla " NDATA flash >
```

// déclaration de l'entité animation dont le contenu est le fichier anim fla de format flash (NDATA : notation data)

```
<objet type= "animation" />
```

//l'application flash traitera anim fla

# DTD: ENTITE

66

Une **entité** peut-être considérée comme un alias permettant de réutiliser des informations internes<sup>s</sup> ou externes<sup>s</sup> au sein du document XML ou de la définition DTD.

# DTD: ENTITE (suite)

67

- Entités se subdivisent en deux catégories:
  - ▣ Analysables: contiennent un texte XML bien formé
  - ▣ Non-analysables: contiennent du texte non-XML ou des données binaires.
- Ils sont de deux types:
  - ▣ Internes : définies dans l'entité document elle-même
  - ▣ Externes: elles dépendent d'une source de données externe au document XML

# DTD: ENTITE (suite)

68

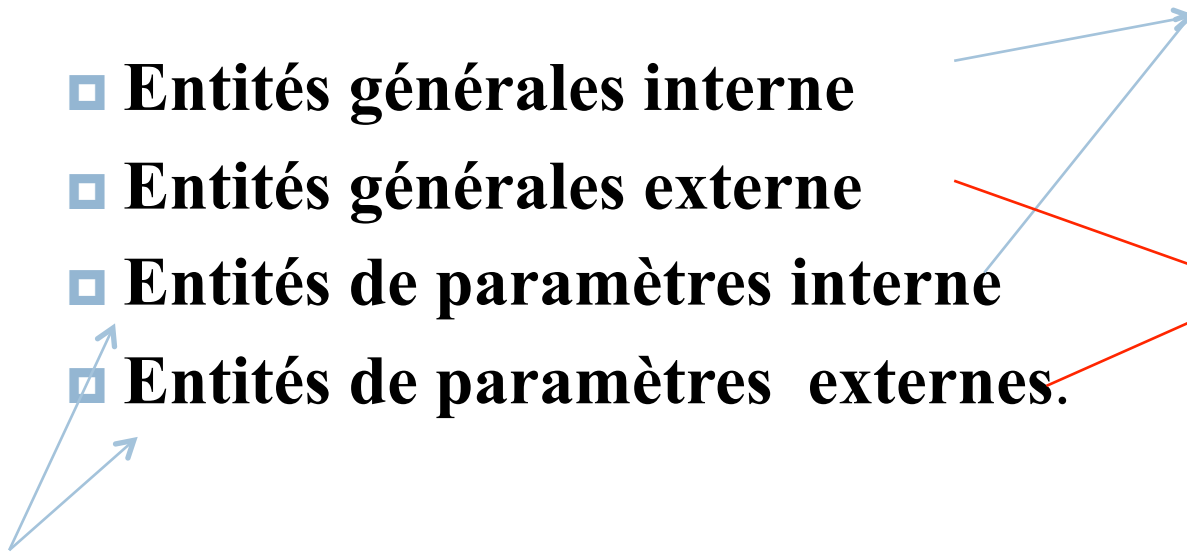
□ On classe les types d'entité comme suit:

- ▣ **Entités générales interne**
- ▣ **Entités générales externe**
- ▣ **Entités de paramètres interne**
- ▣ **Entités de paramètres externes.**

Définies à l'intérieur du document

Identifiées par URL

Utilisées par les DTD



# DTD: Entités générales internes

69

- Les **entités générales internes** permettent d'associer un alias à un texte fréquemment utilisé dans le document XML

**<!ENTITY nom "valeur">**

- Pour utiliser une entité générale dans notre document XML, il suffit d'utiliser la syntaxe suivante : **&nom;**

# DTD: Entités prédéfinies

70

Les entités suivantes sont :

Entité	Caractère
lt	<
gt	>
amp	&
quot	"
apos	'

# DTD Entité interne: Exemple

71

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE livre [
  <!ENTITY auteur "Nom et prénom de l'auteur">
]>
<livre>
  <auteur>
Cours réalisé par &auteur;
  </auteur>
</livre>
```

# DTD: Entités générales externes

72

- Les entités externes sont classées en deux types:
  - ▣ ***Analysables***: contiennent des données textuelle ou binaire.

<!ENTITY nom SYSTEM "URI">

- Pour utiliser une entité externe dans votre XML, il suffit d'utiliser la syntaxe suivante : **&nom;**
  - ▣ ***Non analysable***: le contenu peut être de toute forme (texte, binaire..). Ces entités sont associées au terme de NOTATION (utilisé au niveau d'attribut)



# DTD Entités générales Analysable

73

- Les entités externes représentent des données contenues dans des fichiers séparés par rapport au document XML.

`<!ENTITY nom SYSTEM "URI">`

- Pour utiliser une entité générale externe dans le document XML, il suffit d'utiliser la syntaxe suivante : `&nom;`

# DTD Entités externes: Exemple

74

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE livre [
  <!ENTITY auteur "Nom et prénom de l'étudiant">
  <!ENTITY chap1 SYSTEM "chapitre1.xml">
]>
<livre>
  <auteur>
Cours réalisé par &auteur;
  </auteur>
  &chap1;
</livre>
```

# *DTD: Entités générale externe non analysable*

75

- Permettent de déclarer un contenu non XML dans le document XML (fichiers binaires images, sons...). Le mot clé NDATA (Notation DATA) précise le type d'entité non analysée que le processeur XML doit traiter.
- Référence se fait dans une valeur d'attribut

Exemple:

```
<!NOTATION jpeg SYSTEM "image/jpeg">
```

```
<!ENTITY maphoto SYSTEM "toto.jpeg" NDATA jpeg>
```

```
<image src= "maphoto " />
```

# DTD: Entité Paramètre interne

76

- Les *entités paramètres* peuvent uniquement être utilisées à l'intérieur de la DTD. Elles permettent d'associer un alias à une partie de la déclaration de la DTD.

**<!ENTITY % nom "valeur">**

Ex:

**<!ENTITY % listeMarques "marque (Samsung|Apple) #REQUIRED">**

**<!ATTLIST telephone %listeMarques; >**

**<telephone marque="Samsung" />**

# DTD: Entité Paramètre externe

77

- Les entités paramètres externes permettent d'inclure des DTD externes.

```
<!ENTITY % nom SYSTEM "URI_dtd_externe">
```

On intègre les composants du dtd externe en appelant  
**%nom;** dans la DTD concernée.

# DTD: Limites

78

La DTD se caractérise par sa simplicité. Néanmoins plusieurs limites se posent:

- Syntaxe spécifique
  - analyseurs XML non appropriés
- Manque de typage de données détaillé
  - pas de possibilité de créer de nouveaux types ou de distinguer des types: nombres entiers, nombres réels et chaînes de caractères
- Pas d'héritage des objets

# DTD: Limites

79

- ❑ Pas de possibilité de contrôler le nombre d'éléments contenu dans une balise;
- ❑ Pas de possibilité de faire des références vers d'autres DTD.
- ❑ Difficulté de définir l'ordre d'un contenu mixte
- ❑ Support des espaces de noms très limité.
- ❑ ...

# Bibliographie

80

- [Livre XML cours et Exercice : https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf](https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf)
- <http://www.telug.ca/inf6450/mod2/chapitre6.xml>
- Espace de nom :  
<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>
- Dtd :  
[http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD\\_0.7.swf](http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf)
- <https://www.lri.fr/~benzaken/documents/sldomsax.pdf>
- <http://www-lium.univ-lemans.fr/~lehuen/master1/xml/cours/xmldiapo3.pdf>



# Bibliographie

81

## □ XPATH:

- <http://www.w3.org/TR/xpath/>

- <http://www.loria.fr/~abelaid/Enseignement/miage-m1/XSL-Xpath.pdf>

## □ XSLT

- [http://miage.univ-nantes.fr/miage/D2X1/chapitre\\_xslt/section\\_regles.htm#22](http://miage.univ-nantes.fr/miage/D2X1/chapitre_xslt/section_regles.htm#22)

# Webographie

82

## □ Livres

Livre XML cours et Exercice :

<https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf>

## □ Espace de nom :

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>

## □ DTD :

[http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD\\_0.7.swf](http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf)