

# Cours Données post-relationnelles

**Sana Sellami**

sana.sellami@univ-amu.fr

**Master 1 Informatique**

2022-2023

# Objectifs et organisation du cours

- Objectifs:
  - Apprendre à construire des documents XML
  - Définir une grammaire pour valider les documents
  - Apprendre à transformer des documents XML
  - Apprendre à analyser des documents XML

## **Modalité de contrôle des connaissances**

**Session 1:  $NF = 3/4 \text{ note Projet} + 1/4 \text{ Examen Terminal (ET)}$**

**Session 2:  $NF = \text{Max} (ET, 0.25 * ET + 0.75 * \text{Projet})$**

# PLAN DU COURS

- I. Introduction au langage XML
- II. XML (eXtensible Markup Language)
- III. DTD (Definition Type Document)
- IV. Localisation des composants avec XPath
- V. Transformation de documents XML avec XSLT
- VI. Modélisation des documents avec les Schémas XML

# I. Introduction au langage XML

...

## De SGML à XML en passant par HTML

	1986 : SGML	1991 : HTML	1998 : XML
Objectifs	<ul style="list-style-type: none"><li>▪ adaptabilité,</li><li>▪ intelligence,</li><li>▪ gestion des liens</li></ul>	<ul style="list-style-type: none"><li>▪ simple,</li><li>▪ portable,</li><li>▪ gestion de liens</li></ul>	<ul style="list-style-type: none"><li>▪ puissance de SGML,</li><li>▪ simplicité du HTML</li></ul>
Inconvénients	<ul style="list-style-type: none"><li>▪ complexe,</li><li>▪ difficilement portable</li></ul>	<ul style="list-style-type: none"><li>▪ non adaptable,</li><li>▪ non intelligent</li></ul>	

# HTML ?

- Langage de balisage issu du SGML(Standard Generalized Markup Language)
- Naissance du World Wide Web
- Partage d'informations sur Internet de façon efficace et relativement structurée.

☹ Demandes continuelles de modifications et d'extensions (versions 1.0, 2.0, 3.0, 4.01, 5, etc.).

☹ Incompatibilité des extensions proposées (mise en forme, formule mathématique, police de caractères, ...).

☹Mélange de balises sémantiques (<ADDRESS>, <TITLE>, <H1>) et de balises de mise en forme (<B>, <I>, <FONT>).

☹ Difficulté d'analyser un document HTML.

# XML??

- Langage de balisage pour la description de documents structurés (eXtensible Markup Language)
- Est une recommandation W3C ([www.w3c.org/XML](http://www.w3c.org/XML))
- XML est un langage pour représenter les données du Web
- XML standardise la manière dont l'information est :
  - Représentée
  - Échangée
  - Retrouvée
  - Transformée
  - ...

# Exemple

- Document HTML

```
<dd> Ma chanson </dd>
<ul>
<li> par l'auteur</li>
<li> Producteur : Dupond</li>
<li> Editeur : Maison edition</li>
<li> Duree : 6:20</li>
<li> Date : 1978</li>
<li> Artiste : Toto</li>
</ul>

</body>
</html>
```

- Document XML

```
<SONG>
...
<TITLE> Ma chanson </TITLE>

<COMPOSER> par l'auteur</COMPOSER>

<PRODUCER> Dupond</PRODUCER>

<EDITOR> Maison edition</EDITOR>

<DURATION> 6:20</DURATION>

<DATE> 1978</DATE>

<ARTIST> Toto</ARTIST>

</SONG>
```

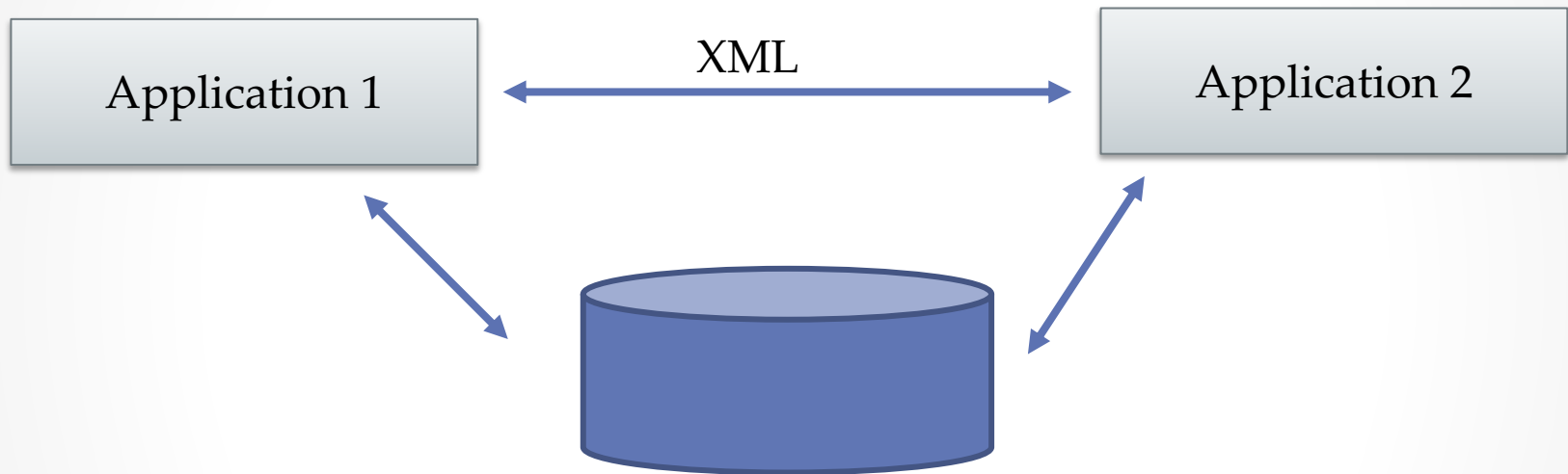


# Différences XML et HTML

- XML et HTML sont 2 langages distincts
- Le XML décrit, **structure**, **échange** des données tandis que le HTML ne fait qu'afficher des données.
- Le XML est extensible et permet de **créer ses propres balises** en fonction des données traitées. En HTML, les balises sont **prédéfinies** et donc figées.
- Outre les PCs, le XML se veut **adapté** aux outils comme les mobiles, etc alors que HTML est surtout conçu pour les ordinateurs de type PC.
- Le XML est un langage **strict** dont l'écriture doit être rigoureuse alors que le HTML est devenu très permissif (à cause des navigateurs récents).

# Pourquoi utiliser XML?

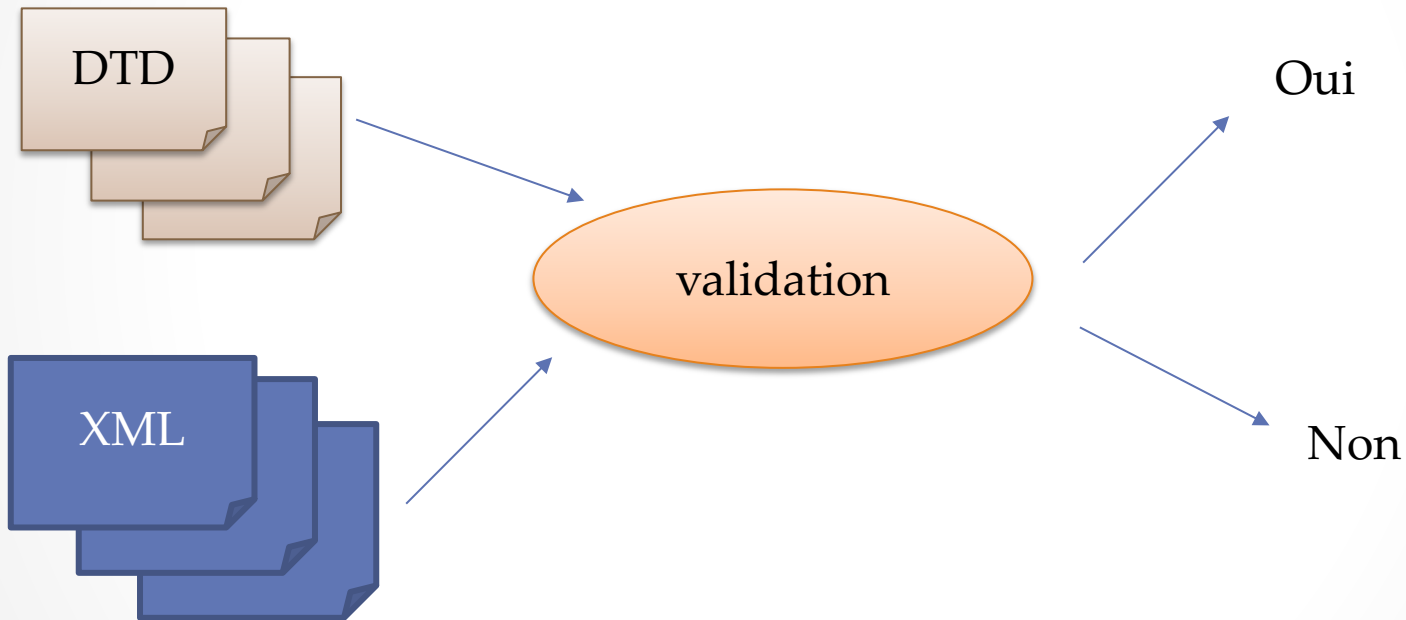
## XML pour communiquer



- Quelques exemples:
  - protocole de transmission (XML-RPC, SOAP),
  - définition de normes (XHTML, XML-FO, SVG),
  - Bases de données semi-structurées,
  - Fichiers de configuration (JSP, EJB, ...),
- Avantages: portabilité, extensibilité, réutilisation d'outils

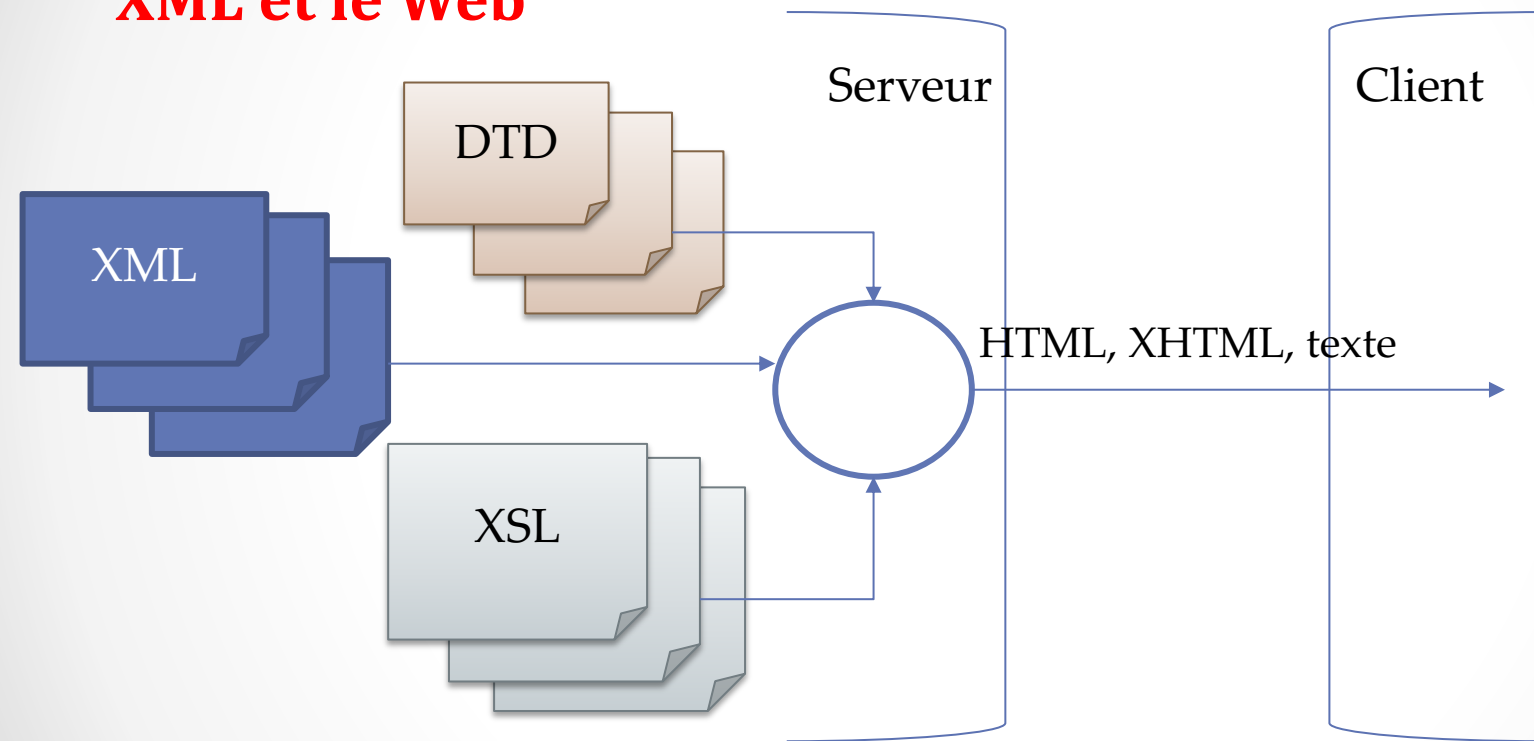
# Pourquoi utiliser XML?

## XML pour vérifier un document



# Pourquoi utiliser XML?

## XML et le Web

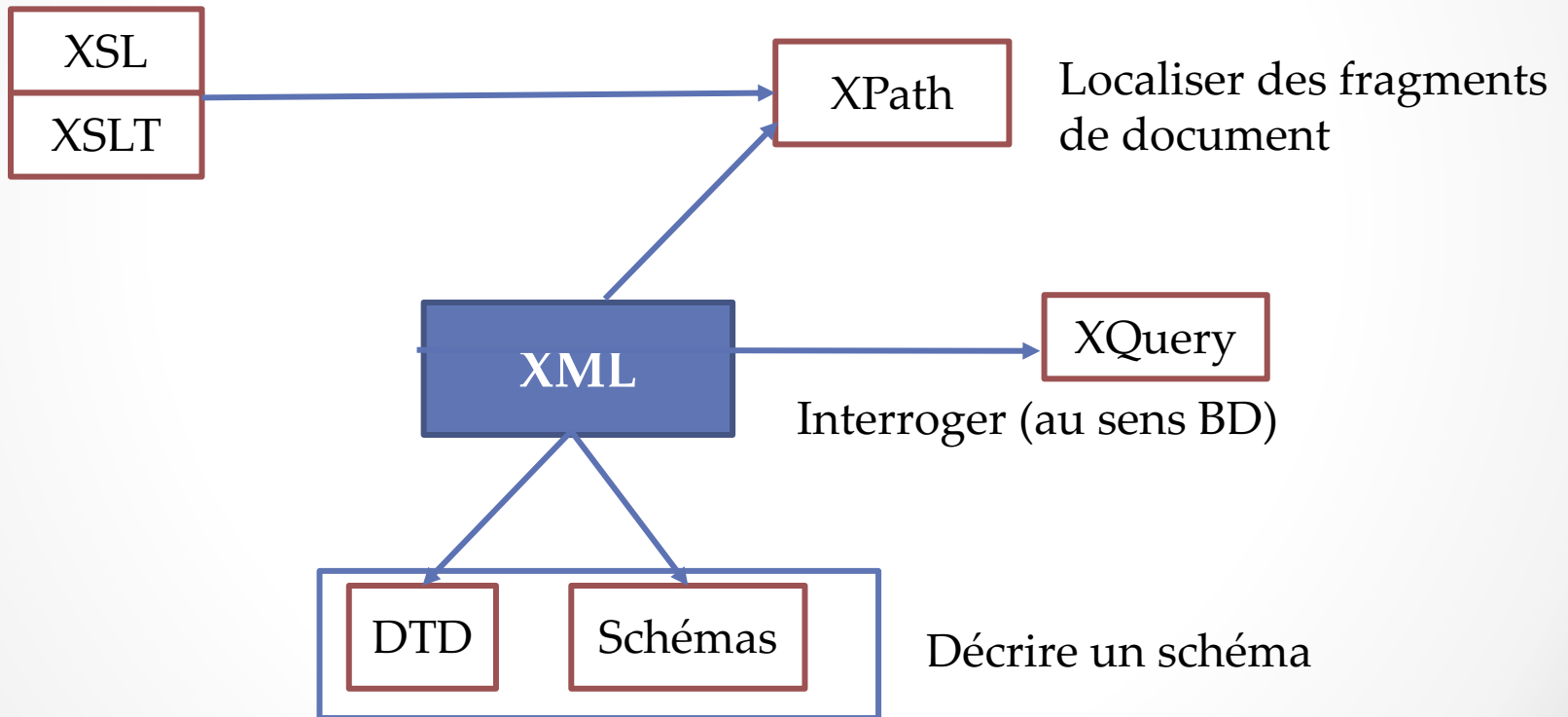


# Caractéristiques

- **Extensibilité:** possibilité de définir ses propres balises
- **Modularité et réutilisation:** mise en œuvre de mécanismes de factorisation de structure et/ou contenu
- **Contrôle de validité:** la possibilité de pouvoir définir des modèles de documents grâce à des langages tels que DTD, XSchema,..., permettant de définir la validité de documents XML par rapport aux contraintes définies dans un modèle.
- **Séparation contenu/contenant:** apport d'un enrichissement structurel grâce aux feuilles de style
- **Représentation arborescente logique**
- **Sensibilité à la casse**

# Outils associés

Présenter  
Transformer



# Les parseurs XML

- **Objectif:** Analyse du document XML.
- **Rôle:** Vérifier la cohérence du document XML et de transmettre à l'application les informations utiles au traitement du document
- Exemples d'API:
  - API SAX (Simple API for XML)
  - API DOM (Document Object Model)
  - JAXP (Java API for XML Processing)
  - JAXB (Java API for XML Binding)
  - Etc.,

# Logiciels

- Pour l'édition et la validation de documents XML:
  - l'IDE eclipse
  - Le logiciel commercial XMLSpy de Altova (<http://www.xmlspy.com>) disponible sous Windows ou sous linux.
  - Le logiciel commercial Oxygen XML Editor (<http://www.oxygenxml.com>)
  - EditiX Editor (<http://www.editix.com/>)
  - Cooktop (<http://www.xmlcooktop.com/>)
  - XML Copy Editor (<http://xml-copy-editor.sourceforge.net/>)
  - Jaxe (<https://sourceforge.net/projects/jaxe/files/>)
- **BaseX**: système de gestion de base de données XML (XPath et XQuery)
- Exist: système de gestion de base de données XML (XPath et XQuery)
- ....



## II. XML: Structure et Manipulation

...

- **Structure d'un document XML**
  - **Documents bien formés**
  - **Les espaces de noms**

- Prologue
- Élément
- Attribut
- Entités prédéfinies
- Commentaire
- Instruction de traitement

# Exemple

```
<?xml version= "1.0 " encoding= "ISO-8859-1 ">
<!-- ceci est un commentaire-->
<cours>
  <titre> XML </titre>
  <intervenant> sana sellami </intervenant>
  <universite> Aix Marseille Universite</universite>
  <ville codepostal="13013">Marseille </ville>
</cours>
```

# Spécifications du langage XML

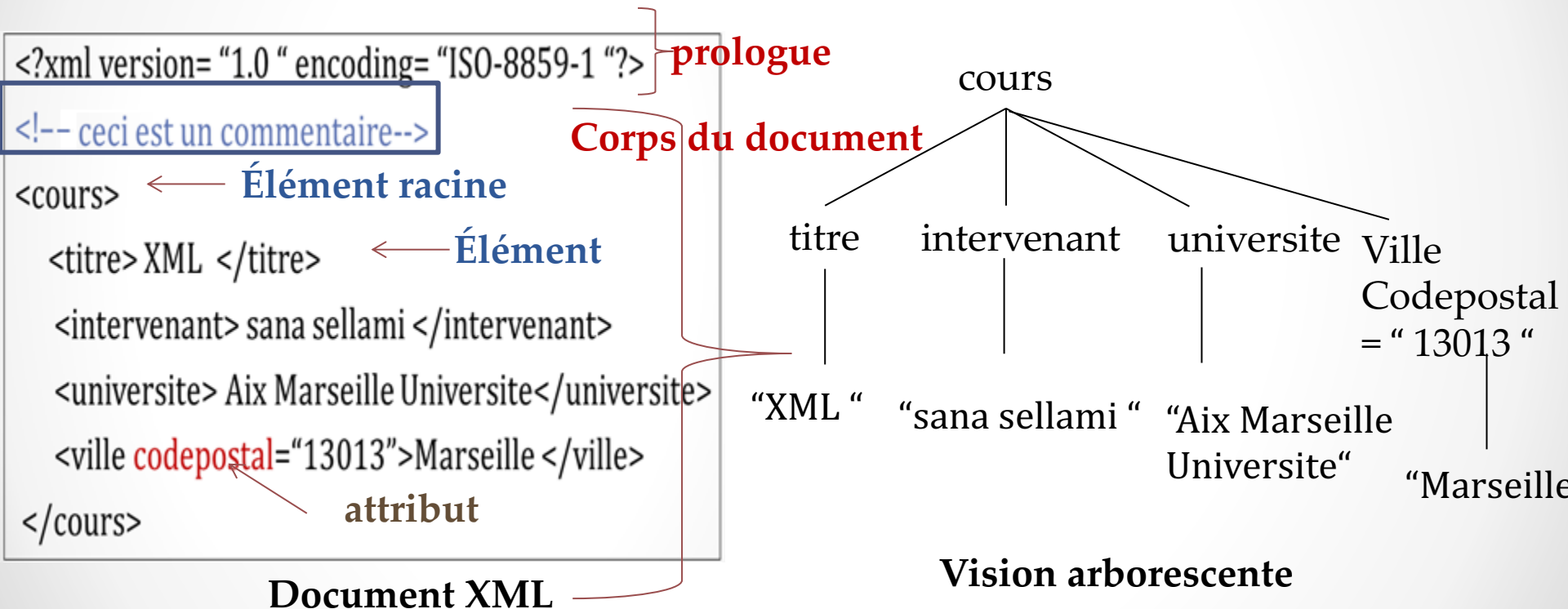
- **Structure** d'un document XML:
  - Un prologue ou en-tête (déclarations)
  - Suivi d'un (seul) arbre d'éléments (balises)
  - Des commentaires
  - Des instructions de traitement
- Un document XML qui respecte les règles syntaxiques est dit **bien formé**. Il est utilisable sans DTD (la grammaire d'un document XML).
- Un document XML bien formé qui respecte sa DTD est dit **valide**.

## Structure d'un document XML

Document Bien formé  
Espaces de noms

- Prologue
- Élément
- Attribut
- Entités prédéfinies
- Commentaire
- Instruction de traitement

# Structure d'un document XML



Traiter un document XML = extraire des informations d'un arbre.

# Prologue

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

- Optionnel mais conseillé
- **Paramètre Version:** Version de XML utilisée pour décrire le document soit 1.0 soit 1.1
- **Paramètre encoding:**
  - Le jeu/codage de caractères utilisé. L'encodage est basé sur la norme ISO 10646 (<http://www.unicode.org>).
  - Les analyseurs XML reconnaissent automatiquement l'encodage UTF-8 et UTF-16.
- **Paramètre standalone:** indique si pour pouvoir être « interprété » et manipulé, le document XML se suffit à lui-même (dans ce cas la valeur du paramètre standalone est yes) ou non (déclarations externes pouvant provenir d'une DTD externe où d'entités paramètres).

# Élément

- **Forme générale:**

`<nom-d'element> contenu de l'element </nom-d'element>`

- Un élément est composé:

- D'une balise de début qui contient le nom de l'élément (dont le premier caractère est soit une lettre, soit un espace souligné soit un deux points). **Les noms** (appelés des **identificateurs** en XML) sont libres mais obéissent à quelques règles :
  - Premier caractère: alphabétique; - (tiret), \_ (souligné)
  - Les autres caractères: alphabétique, chiffre, - , \_ , :
  - Pas d'espace
  - **Interdiction de mettre xml au début**
- Eventuellement des attributs
- D'un contenu
- D'une balise de fin

# Élément

- **Exemple:**

`<ville codepostal= "13013 ">Marseille </ville>`

- Balise de début: `< ville codepostal = "13013 ">`
- Nom: ville
- Attribut: `codepostal= "13013 "`
- Contenu: Marseille
- Balise de fin: `</ville>`

# Élément: Contenu

4 types d'éléments en fonction de leur contenu:

**1. Élément Vide:** ne possède pas de contenu (sauf les attributs)

`<element></element>` ou `<element/>`

**2. Un élément textuel :** qui ne contient que des données textuelles mais aussi:

- des sections CDATA (ce sont aussi des données textuelles)
- des commentaires (ne sont pas pris en compte lors de l'interprétation et la validation des documents)
- des instructions de traitement (mais ce ne sont pas des données)

`<element> ceci est un exemple </element>`



# Élément: Contenu

## 3. Élément mixte:

- Contient à la fois **des données textuelles** et **des éléments imbriqués**.
- **Exemple:**

`<exemple> Le monde <sigle>XML</sigle> est riche (très riche) </exemple>`

L'élément exemple est composé dans l'ordre

- **D'un texte** Le monde
- **D'un élément** `<sigle>XML</sigle>`
- **D'un texte** est riche (très riche)

## Élément: Contenu

**4. Élément Complexe:** est un élément dont le contenu est composé exclusivement d'éléments imbriqués (des sous éléments) qui ne sont pas forcément complexes(vides et/ou textuels et/ou complexes, et/ou mixtes).

```
<element-complexe>
```

```
  <element1> textuel </element1>
```

```
  <element2>
```

```
    <!--exemple d'élément imbriqué complexe-- >
```

```
    <sous-element>encore un élément textuel
```

```
    </sous-element>
```

```
  </element2>
```

```
</ element-complexe >
```

# Attribut

- Un attribut est une paire nom-valeur où :
  - le nom est un nom XML ;
  - la valeur (obligatoire) est une suite de caractères entre guillemets ou entre apostrophes.
  - Attributs séparés par au moins un espace (blanc simple, tabulation retour à la ligne)
  - **Exemple:** codepostal= "13013 "
- Complémentaire de l'élément: ajoute une information à l'élément ou le complète dans sa définition.
- Dans une même balise ouvrante, les attributs doivent porter **des noms différents**
- La valeur des attributs ne peuvent pas contenir directement les caractères %, ^ et &.

## Attributs réservés

- *xml:lang='langue'* permet de définir la langue utilisée dans l'élément et tous les sous-éléments. La langue suit la norme ISO 3166 définie par la RFC 1766 (Request For Comment). Par exemple fr ou en-US ou fr-FR .
- *xml:space='preserve'* ou *xml:space='default'* permet de définir l'interprétation des espaces dans l'élément et tous les sous-éléments.

## Éléments-Attribut

- Les attributs introduisent une ambiguïté dans la représentation de l'information: Élément? Attribut?
- Un attribut **apparaît une unique fois au sein** d' un élément
- **Un même élément peut apparaître plusieurs fois** dans un élément
- Les attributs associés à un élément **ne sont pas ordonnés**
- Les éléments **sont ordonnés** (ordre de lecture du document)
- les éléments décrivent les données alors que les attributs définissent les éléments qui les contiennent.

# Exercice

Création d'un livre en XML (livre1.xml)

- On souhaite écrire un livre en utilisant le formalisme XML. Le livre comprend une section, un auteur (avec nom et prénom), 1 chapitre par section et 2 paragraphes par chapitre.
- Tous les éléments doivent posséder un titre, sauf le paragraphe (qui contient du texte )et l'auteur.

# Exercice

```
<livre titre=" mon livre">  
  <auteur nom="Martin" prenom="Bill" />  
  <section titre=" une section" >  
    <chapitre titre="un chapitre " >  
      <paragraphe>paragraphe 1 </paragraphe>  
      <paragraphe>paragraphe 2 </paragraphe>  
    </chapitre>  
  </section>  
</livre>
```

# Entités prédéfinies

- Certains caractères sont réservés à la syntaxe XML. Il faut être vigilant lors de l'écriture des données.

**Exemple:** `<calcul> if (a<b et b>c) </calcul>`

- Dans cet exemple, `<b et b>` n'est pas une balise mais fait partie des données liées à l'élément calcul.

→ Utilisation d'entités prédéfinies: **&lt;** (less than <); **&gt;** (greater than >), **&amp;** ; (&), **&quot;** ; ("), **&apo;** ; (')

`<calcul> if (a &lt; b et b &gt; c) </calcul>`



## Section CDATA

- Syntaxe: `<![CDATA[ texte ]]>`
- Le texte peut contenir n'importe quels caractères sauf la chaîne `]]`
- **Exemple:**  
`<calcul> <![CDATA[if (a<b et b>c)]] > </calcul>`

# Commentaire

- `<!-- commentaire-->`
- Peut comporter toute suite de caractères sauf --
- Un commentaire peut apparaître partout dans un document **excepté** :
  - dans la balise ouvrante ou fermante d'un élément ;
  - dans une instruction de traitement ;
  - dans un commentaire.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- ceci est un commentaire-->
<cours>
  <titre>XML </titre>
  <intervenant> sana sellami </intervenant>
  <universite> Aix Marseille Universite</universite>
  <ville codepostal="13013">Marseille </ville>
</cours>
```

# Instruction de traitement

- Destinées aux applications qui traitent les documents XML. Elles sont l'analogue des directives #... du langage C qui s'adressent au compilateur
- Commence par <? Suivi du nom de l'instruction de traitement suivi de ses éventuels paramètres suivis de ?>

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

- Bien que le prologue s'écrit avec la même syntaxe qu'une instruction de traitement, ça n'en est pas une.

Un **problème** apparaît si on mélange deux documents XML dont les éléments ont le même nom mais pas la même définition.

```
<?xml version="1.0" ?>
<auteur>
  <nom>Poulard </nom>
  <prenom>Philippe</prenom>
  <titre> Baron </titre>
</auteur>
```

```
<?xml version="1.0" ?>
<cours>
  <titre>Fondamentaux XML </titre>
  <contenu> ...
  </contenu>
</cours>
```

Fusion des deux documents

```
<?xml version="1.0" ?>
<cours>
  <titre>Fondamentaux XML </titre>
  <auteur>
    <nom>Poulard </nom>
    <prenom>Philippe</prenom>
    <titre> Baron </titre>
  </auteur>
  <contenu> .../...
  </contenu>
</cours>
```

Confusion sur le sens  
de l'élément titre

# Espaces de noms XML

- Pouvoir mélanger plusieurs vocabulaires au sein d'un même document
- Les espaces de noms **doivent être utilisés** si un document XML est destiné à être mélangé à d'autres sources.
- **Objectifs :**
  - **distinguer** les éléments et les attributs des différentes applications XML qui ont le même nom
  - **grouper** les éléments et les attributs d'une même application XML pour que les logiciels puissent les reconnaître

# Espaces de noms par défaut

- Se déclare dans un élément avec l'attribut **xmlns**
- Identifié par une URI (Uniform Resource Identifier) (très svu une URL)
- **Exemple**

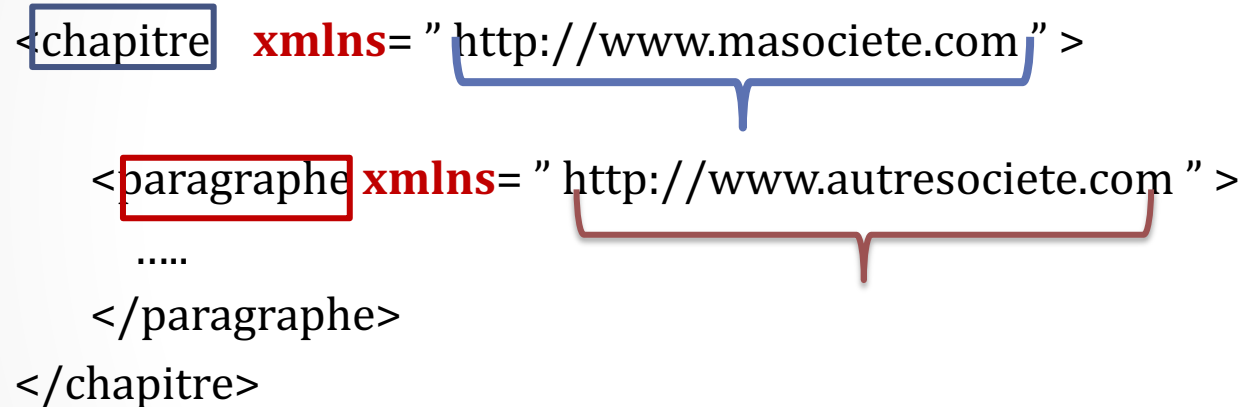
```
<chapitre xmlns="http://www.masociete.com" >  
  <paragraphe>  
    .....  
  </paragraphe>  
</chapitre>
```

- La déclaration d'espace de noms est sensée s'appliquer à l'élément sur lequel elle est spécifiée et à tous les éléments contenus dans celui-ci.

# Espace de noms par défaut

- **Exemple**

```
<chapitre xmlns="http://www.masociete.com">  
  <paragraphe xmlns="http://www.autresociete.com">  
    ....  
  </paragraphe>  
</chapitre>
```



- Un espace de noms **par défaut ne concerne que les éléments**. Les attributs et les textes n'y appartiennent pas. Le texte d'un élément n'est jamais dans un espace de noms puisqu'il représente la donnée.

## Utilisation de plusieurs espaces de noms

- Se déclare dans un élément avec l'attribut **xmlns**
- Identifié par une URI (Uniform Resource Identifier) (très svt une URL)
- A qui on associe un préfixe (il peut être réservé, usuel ou librement choisi)

**xmlns: préfixe**= " uri de l'espace de noms " >

### Exemple

<**p:** resultat **xmlns:p**= " http://www.masociete.com " >

</**p:**resultat>

- L'utilisation du préfixe pour un élément (ou attribut) indique que cet élément (ou attribut) appartient à l'espace de noms associé au préfixe.  
**(nom qualifié)**

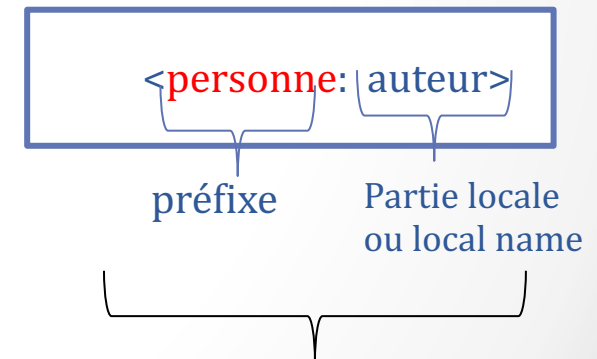


## Utilisation de plusieurs espaces de noms

- Le préfixe est placé avant le nom d'élément ou d'attribut extrait de la ressource associée : `prefixe:nom_local`

```
<?xml version="1.0"?>
<cours:cours xmlns:cours="http://www.foo.com/cours">
  <cours:titre>Fondamentaux XML</cours:titre>
  <personne:auteur xmlns:personne="http://www.bar.com/individus">
    <personne:nom>Poulard</personne:nom>
    <personne:prénom>Philippe</personne:prénom>
    <personne:titre>Baron</personne:titre>
  </personne:auteur>
  <cours:contenu>
    ...
  </cours:contenu>
</cours:cours>
```

Élément qualifié:



Nom qualifié ou  
qualified name ou QName

# Utilisation de plusieurs espaces de noms

- **Exemple**

Document 1:

```
<p: res xmlns:p=" http://www.masociete.com " >  
</p:res>
```

Document 2:

```
<zz: res xmlns:zz=" http://www.masociete.com " >  
</zz:res>
```

Que pensez vous de ces deux documents?

Les documents 1 et 2 sont strictement identiques malgré un préfixe différent. Res étant dans tous les cas un élément appartenant à l'espace de noms <http://www.masociete.com>

# Utilisation de plusieurs espaces de noms

- **Exemple**

```
<p: res xmlns:p="http://www.masociete.com" xmlns:p2="http://www.autresociete.com">  
  <p2:res>  
  </p2:res>  
</p:res>
```

À quel espace de noms appartient l'élément res ?

Le premier élément res est dans l'espace de noms `http://www.masociete.com` alors que  
l'élément res à l'intérieur est dans l'espace de noms `"http://www.autresociete.com"`

On ne peut pas utiliser plusieurs préfixes en même temps sur un élément, attribut ou  
valeur d'attribut

# Utilisation de plusieurs espaces de noms

- **Exemple**

```
<p: element xmlns:p="http://www.masociete.com" >  
    <autreelement/>  
</p:element>
```

Déterminer à quel espace de noms appartiennent element et autreelement ?

L'élément element est dans l'espace de noms http://www.masociete.com alors que l'élément autreelement qui n'est pas préfixé n'a pas d'espaces de noms.

- **Exemple**

```
<element xmlns="http://www.masociete.com" >  
    <autreelement xmlns="" />  
    <encoreunelement>  
    </encoreunelement>  
</element>
```

## Même question

l'élément element est dans l'espace de noms http://www.masociete.com alors que l'élément autreelement n'est plus dans un espace de noms. L'élément encoreunelement se trouve également dans l'espace de noms http://www.masociete.com de par l'espace de noms de son parent.

# Espaces de noms et attribut

- Les attributs peuvent également bénéficier des espaces de noms. Pour ce faire, l'utilisation d'un préfixe est obligatoire et l'espace de noms par défaut ne peut être utilisé.
- **Exemple:**

```
<com:produit xmlns:com="http://www.biz.com/commercial" >  
  <com:prix com:monnaie = "Euro" > ... </com:prix>  
</com:produit>
```

## Quelques déclarations connues

Langages	Espaces de noms
XHTML	<code>&lt;xhtml:xhtml xmlns:xhtml= "http://www.w3.org/1999/xhtml"</code>
XML SCHEMA	<code>&lt;xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"</code>
RDF	<code>&lt;rdf:RDF xmlns:rdf= "http://www.w3.org/TR/REC-rdf-syntax#"</code>
SVG	<code>&lt;svg xmlns="http://www.w3.org/2000/svg"</code>
SOAP	<code>&lt;SOAP-ENV xmlns:SOAP- ENV="http://schemas.xmlsoap.org/soap/envelope"</code>

# Document XML bien formé

- Résumé des spécifications :
  - Un document doit commencer par une déclaration XML
  - Toutes les balises avec un contenu doivent être fermées
  - Toutes les balises sans contenu doivent se terminer par les caractères />
  - Le document doit contenir un et un seul élément racine
  - Les balises ne doivent pas se chevaucher
  - Les valeurs d'attributs doivent être entre guillemets ou apostrophes
  - La casse doit être respectée pour toutes les occurrences de noms de balise (MAJUSCULES ou minuscules).
  - Le nom d'un élément ne peut commencer par un chiffre.
  - Si le nom d'un élément est composé d'un seul caractère il doit être dans la plage [a-zA-Z] ou \_ ou :.
  - Avec au moins 2 caractères, le nom d'un élément peut contenir \_ , - , . et : plus les caractères alphanumériques (attention, le caractère : est réservé à un usage avec les espaces de nom).
- Un document respectant ces critères est dit **“bien formé”**

# Conventions de nommage

- Employer des minuscules pour les attributs et les éléments.
- Éviter les accents dans les noms d'attributs et d'éléments.
- Préférer les guillemets délimitant les valeurs d'attribut.
- Séparer les noms composés de plusieurs mots par les caractères -, \_ . ou une majuscule.
- Essayer d'être homogène dans votre document en gardant la même convention.



# Outils pour vérifier la bonne formation

1. Navigateurs web (Firefox, IE)
  - Si XML bien formé → affichage
  - Sinon → première erreur trouvée
2. Éditeurs XMLSpy, Oxygen, etc.