

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Institut Spécialisé de Technologie Appliquée de Gestion Bab Tizimi - Meknès**

**Les langages**

**XML – DTD**

* **XSD - XSL**

**Année de Formation : 2020/2021**

Contenu

[Le langage XML (Extensible marckup language ) 3](#_Toc64270458)

[1) Définition : 3](#_Toc64270459)

[2) Création d’un document XML 3](#_Toc64270460)

[3) La structure d’un document XML. 4](#_Toc64270461)

[4) Les règles de nommage des balises 5](#_Toc64270462)

[5) Les attributs dans un document XML. 6](#_Toc64270463)

[6) Validation d’un document XML : Un document bien formé. 7](#_Toc64270464)

[Le langage DTD (Document Type Description) 10](#_Toc64270465)

[1) Définition : 10](#_Toc64270466)

[2) Création d’un script DTD : 12](#_Toc64270467)

[3) Optionalité d’un élément dans un document XML. 15](#_Toc64270468)

[4) Déclaration des attributs dans un document DTD 18](#_Toc64270469)

[5) Optionalité des attributs : 20](#_Toc64270470)

[XSD (XML Schema Definition) 21](#_Toc64270471)

[1) Définition : 21](#_Toc64270472)

[2) Création d’un fichier XSD : 21](#_Toc64270473)

[3) Déclaration des éléments XSD : 22](#_Toc64270474)

[Le langage XSLT(Extensible Style Language Transformation 26](#_Toc64270475)

[Ou Les feuilles de style du XML 26](#_Toc64270476)

[1) Définition : 26](#_Toc64270477)

[2) Création d’un fichier XSL : 27](#_Toc64270478)

[5) Pourquoi le xsl : 27](#_Toc64270479)

[6) Les instructions XSL 28](#_Toc64270480)

[7) Les variables en XSL : 32](#_Toc64270481)

[8) Les fonctions XSL : 32](#_Toc64270482)

[Les expressions XPATH 35](#_Toc64270483)

[1) Définition 35](#_Toc64270484)

[2) La syntaxe de Xpath: 35](#_Toc64270485)

# Le langage XML (Extensible marckup language )

### Définition :

**QU'EST-CE QUE LE XML ?**

XML est un langage informatique de balisage générique qui permet de définir un document XML (\*.xml) où on peut stocker des données sous la forme de balises.

**<infos\_PC>**

**<modele>**

**<nom>PC Home</nom>**

**<prix\_unitaire>5999</prix\_unitaire>**

**<fournisseur>Sulyo</fournisseur>**

**<particularites>Base + lecteur DVD</particularites>**

**</modele>**

**<modele>**

**<nom>PC Multimedia</nom>**

**<prix\_unitaire>7999</prix\_unitaire>**

**<fournisseur>Fukoji</fournisseur>**

**<particularites>PC HOME + ecran 19 + carte son</particularites>**

**</modele>**

**</infos\_PC>**

En comparaison avec le HTML qui définit un document HTML, en utilisant des balises définies par l’organisation W3C, le XML décrit la structure de données en utilisant des balises génériques (crées par l’utilisateur).

### Création d’un document XML

Un fichier XML est un fichier d’extension (\*.xml) ; il est constitué par une arborescence de balises empilées les unes dans les autres.

**<?xml version="1.0" encoding="utf-8" ?>**

**<BIBLIOTHEQUE>**

**<LIVRE>**

**<TITRE>titre livre 1</TITRE>**

**<AUTEUR>auteur 1</AUTEUR>**

**<EDITEUR>editeur 1</EDITEUR>**

**</LIVRE>**

**<LIVRE>**

**<TITRE>titre livre 2</TITRE>**

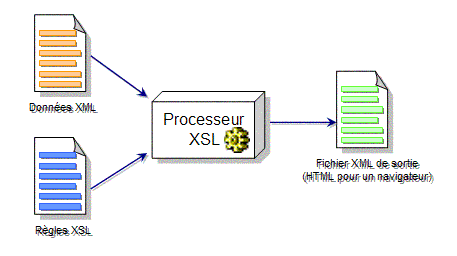
**<AUTEUR>auteur 2</AUTEUR>**

**<EDITEUR>editeur 2</EDITEUR>**

**</LIVRE>**

**</BIBLIOTHEQUE>**

**Pourquoi XML? :**

****

Le fichier XML peut être utilisé comme une base de données qui facilite les échanges de données entre la machine et l’application ou le site web.

Pour créer un fichier XML, il suffit d’utiliser un éditeur de texte ; le fichier est d’extension (\*.xml).

**Exemples :**

* + - 1. **Rapid PHP**
      2. **Visual studio ;**
      3. **Word pad……**

Dans notre cours on va utiliser **Rapid PHP**.

* Démarrer Rapid PHP.
* Cliquer sur file/New et choisir xml file
* Enregistrer Votre fichier sous un nom et d’extension xml.( Dans notre cas le nom du fichier est **BIBLIOTHEQUE.xml. (du document ci-dessus).**

### La structure d’un document XML.

Généralement un document xml est constitué par :

* **Le prologue :**

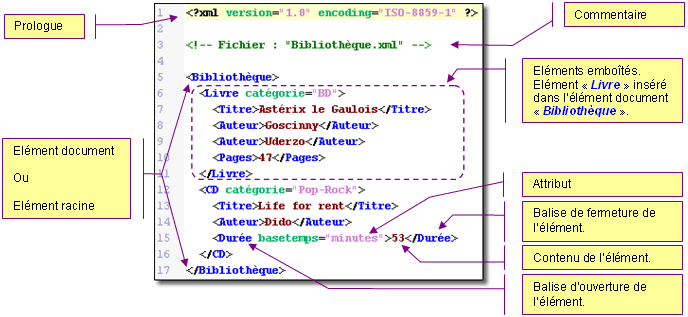
**<? xml version="1.0" encoding="utf-8" ?>**

**C’est la première ligne du document XML ; ce prologue contient :**

* **Le langage utilisé qui est XML.**
* **La version de xml utilisé : version="1.0" ; version="1.1"**
* **Encodage des caractères appliqué : encoding="utf-8"**
  + - **La Racine :**

**C’est la balise conteneur d’un document Xml ; elle contient des autres balises paires ou orphelines (vides).**

**Dans notre cas c’est la balise BIBLIOTHEQUE**

****

### Les règles de nommage des balises

* Les noms peuvent contenir des lettres, des chiffres ou des caractères spéciaux.
* Les noms ne peuvent pas débuter par un nombre ou un caractère de ponctuation.
* Les noms ne peuvent pas commencer par les lettres XML (quelle que soit la casse).
* Les noms ne peuvent pas contenir d'espaces.
* On évitera les caractères **- , ; . <** et **>** qui peuvent être mal interprétés dans vos programmes.
* La casse des noms doit être respectée

Exemple : Identifier les balises correctes et non correctes et indiquer les erreurs s’ils existent :

Distinguez les noms XML corrects des noms incorrects et corrigez les erreurs.

a. **<Drivers\_License\_Number>98 NY 32</Drivers\_License\_Number>**

**b. <Driver's\_License\_Number>98 NY 32</Driver's\_License\_Number>**

**c. <month-day-year>7/23/2001</month-day-year>**

**d. <first name>Alan</first name>**

**e. <àçttûä>øåú</àçttûä>**

**f. <first\_name>Alan</first\_name>**

**g. <month/day/year>7/23/2001</month/day/year>**

**h. <\_4-lane>I-610</\_4-lane>**

**i. <téléphone>011 33 91 55 27 55 27</téléphone>**

**j. <4\_lane>I-610</4\_lane>**

**Réponses :**

**a. Correct**

**b. Incorrect (apostrophe)**

**c. Correct**

**d. Incorrect (présence d'un espace)**

**e. Correct**

**f. Correct**

**g. Incorrect (à cause des /)**

**h. Correct**

**i. Correct**

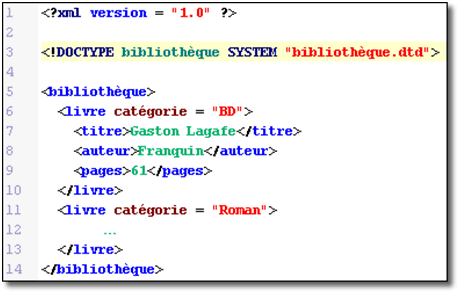
**j. Incorrect (un nom XML ne commence pas par un chiffre)**

### Les attributs dans un document XML.

L’attribut est une paire nom-valeur associée à la balise ouvrante d’un élément XML. En plus des règles de nommage des balises, les attributs doivent être délimités par des guillemets simple ou double.

L’attribut ne peut être présent qu’une seule fois dans une balise.

**Exemple :**



### Validation d’un document XML : Un document bien formé.

* S'il s'agit d'un document utilisant la version 1.0 ou 1.1 du XML, le prologue est bien renseignée.
* Le document XML ne possède qu'une seule balise racine.
* Le nom des balises et des attributs sont conformes aux règles de nommage.
* Toutes les balises en paires sont correctement fermées.
* Toutes les valeurs des attributs sont entre guillemets simples ou doubles.
* Les balises de votre document XML ne se chevauchent pas, il existe une arborescence dans votre document.

**Exemple :**

<?xml version="1.0" encoding="UTF-8" ?>

<repertoire>

<personne sexe="masculin">

<nom>DOE</nom>

<prenom>John</prenom>

<adresse>

<numero>7</numero>

<voie type="impasse">impasse du chemin</voie>

<codePostal>75015</codePostal>

<ville>PARIS</ville>

<pays>FRANCE</pays>

</adresse>

<telephones>

<telephone type="fixe">01 02 03 04 05</telephone>

<telephone type="portable">06 07 08 09 10</telephone>

</telephones>

<emails>

<email type="personnel">john.doe@wanadoo.fr</email>

<email type="professionnel">john.doe@societe.com</email>

</emails>

</personne>

<personne sexe="feminin">

<nom>POPPINS</nom>

<prenom>Marie</prenom>

<adresse>

<numero>28</numero>

<voie type="avenue">avenue de la république</voie>

<codePostal>13005</codePostal>

<ville>MARSEILLE</ville>

<pays>FRANCE</pays>

</adresse>

<telephones>

<telephone type="professionnel">04 05 06 07 08</telephone>

</telephones>

<emails>

<email type="professionnel">contact@poppins.fr</email>

</emails>

</personne>

</repertoire>

Exécution sur le navigateur :

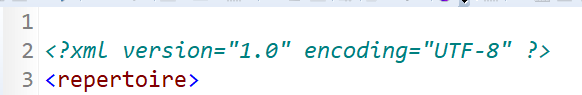
* Exécuter votre fichier sur un navigateur.
* Lire les erreurs détectées par ce navigateur.

Exemple d’erreurs :

1. < ?xml version="1.0" encoding="UTF-8" ?> (espace au début du prologue)



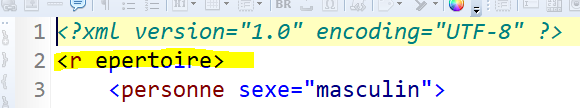
1. Une ligne vide avant le prologue.



Résultat d’erreur sur le navigateur :



1. Le document n’est pas bien mis en forme (valide).



Résultat d’erreur sur le navigateur :



**TP1 Exercice 1 et 2.**

# Le langage DTD (Document Type Description)

### Définition :

Le DTD est une définition d’un document XML; c’est un ensemble de règles que l’on pose au document XML. Ces règles permettent de décrire la façon dont le document XML doit être construit.

Exemples :

Le DTD peut imposer :

* **L’existence des attributs ou des balises.**
* **Imposer l’apparition des balises dans un ordre bien détaillé.**
* **Imposer les types de données du document XML.**

**En général, le DTD est document d’extension DTD qui permet de contrôler la validité d’un document XML.**

Le script DTD peut être intégré dans le document XML (**DTD interne**) ou dans un fichier séparé d’extension (\*.dtd) (**DTD externe**).

* **DTD interne :**

Le script DTD est écrit dans le même fichier XML ; ce script est spécifique pour le document XML en question. Pour se faire ; on ajoute la clause standalone= yes.

Et le code suivant :

|  |
| --- |
| **<?xml version = "1.0" encoding="UTF-8" standalone="yes" ?>**  **<!DOCTYPE repertoire [**  **…….**  **]>**  **<repertoire>**  **…..**  **</repertoire>.** |

**Exemple : copier le code suivant est l’exécuter sous Google chrome.**

**<?xml version = "1.0" encoding="UTF-8" standalone="yes" ?>**

**<!DOCTYPE repertoire [**

**]>**

**<repertoire>**

**<!-- John DOE -->**

**<personne sexe="masculin">**

**<nom>DOE</nom>**

**<prenom>John</prenom>**

**<adresse>**

**<numero>7</numero>**

**<voie type="impasse">impasse du chemin</voie>**

**<codePostal>75015</codePostal>**

**<ville>PARIS</ville>**

**<pays>FRANCE</pays>**

**</adresse>**

**<telephones>**

**<telephone type="fixe">01 02 03 04 05</telephone>**

**<telephone type="portable">06 07 08 09 10</telephone>**

**</telephones>**

**<emails>**

**<email type="personnel">john.doe@wanadoo.fr</email>**

**<email type="professionnel">john.doe@societe.com</email>**

**</emails>**

**</personne>**

**<!-- Marie POPPINS -->**

**<personne sexe="feminin">**

**<nom>POPPINS</nom>**

**<prenom>Marie</prenom>**

**<adresse>**

**<numero>28</numero>**

**<voie type="avenue">avenue de la république</voie>**

**<codePostal>13005</codePostal>**

**<ville>MARSEILLE</ville>**

**<pays>FRANCE</pays>**

**</adresse>**

**<telephones>**

**<telephone type="professionnel">04 05 06 07 08</telephone>**

**</telephones>**

**<emails>**

**<email type="professionnel">contact@poppins.fr</email>**

**</emails>**

**</personne>**

**</repertoire>**

* **DTD externe :**

Le script DTD doit être dans un fichier séparé du fichier XML.

1. En effet créer un fichier dont l’extension est DTD.

* **Dans rapid PHP ; cliquer sur file/new /text document.**
* **Enregistrer ce fichier sous le nom repertoire.dtd.**

1. Modifier la valeur de **standalone en «no ».**
2. Pour relier le fichier repertoire.dtd au fichier repertoire.xml ; on ajoute le code suivant au fichier XML :

**<!DOCTYPE repertoire SYSTEM "repertoires.dtd" >**

**Exemple :**

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

**<!DOCTYPE repertoire SYSTEM "repertoire.dtd" >**

**<repertoire>**

……

Dans cette ligne, on indique la racine du document XML (repertoire) et l’emplacement du fichier DTD :

* **SYSTEM:** dans le même serveur ou ordinateur et en suite le chemin du fichier (\*.dtd).
* **PUBLIC :** indique que le fichier \*.dtd existe dans un autre serveur d’internet ; et en suite en définit le lien de ce fichier.

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

### Création d’un script DTD :

En premier lieu le fichier \*.dtd commence par le prologue suivant:

**<?xml version="1.0" encoding="UTF-8"?>**

**Exemple : (Voir le fichier repertoire.xml).**

**<?xml version="1.0" encoding="UTF-8" standalone=yes?>**

**<!DOCTYPE repertoire[**

**<!ELEMENT repertoire (personne+)>**

**<!ELEMENT personne (nom,prenom,adresse,telephones,emails)>**

**<!ATTLIST personne sexe (masculin|feminin) #REQUIRED>**

**<!ELEMENT nom (#PCDATA) >**

**<!ELEMENT prenom (#PCDATA) >**

**<!ELEMENT adresse (numero,voie,codePostal,ville,pays)>**

**<!ELEMENT numero (#PCDATA) >**

**<!ELEMENT voie (#PCDATA) >**

**<!ATTLIST voie type (CDATA) #IMPLIED>**

**<!ELEMENT codePostal (#PCDATA) >**

**<!ELEMENT ville (#PCDATA) >**

**<!ELEMENT pays (#PCDATA) >**

**<!ELEMENT telephones (telephone+)>**

**<!ELEMENT telephone (#PCDATA)>**

**<!ATTLIST telephone type (fixe|portable) #REQUIRED>**

**<!ELEMENT emails (email+)>**

**<!ELEMENT email (#PCDATA)>**

**<!ATTLIST email type (CDATA) #IMPLIED>**

**]>**

* **ELEMENT :** Ce mot clé permet de définir la règle portant sur une balise (nom\_balise).

**< !ELEMENT nom\_balise (contenu)>**

**Exemple 1:**

**<!ELEMENT repertoire (personne+)>**

Cette ligne signifie que la balise **repertoire** contient une ou plusieurs autres balises **personne**.

**Exemple 2:**

**<!ELEMENT nom (#PCDATA) >**

Cela indique que la balise contient un texte et pas de balise.

**Exemple 3:**

**<!ELEMENT personne (nom, prenom, adresse, telephones , emails)>**

Cela indique que la balise personne contient plusieurs balises définies par leurs noms et sans répétition.

L’enchaînement des balises sont imposé par le DTD ; les balises doivent être dans le même ordre décrit dans le DTD.

**Exemple 4:**

**<!ELEMENT livre (intitule, edition, #PCDATA)>**

Cela indique que la balise **livre** contient des balises dans l’ordre indiqué en plus d’un texte qui est à la fin de la balise.comme :

<livre>

<intitule>Victor Hugo </intitule>

<edition>2005</edition>

**histoire**

<livre>

Exemple : soit le script XML suivant ; écrire son code DTD correspondant :

**<personne>**

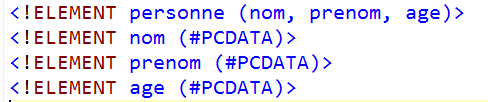
**<nom>DOE</nom>**

**<prenom>John</prenom>**

**<age>24</age>**

**</personne>**

**Réponse :**



A partir du document DTD on vérifie la validité des documents XML ; en effet un document est valide s’il respecte les règles imposé par le DTD correspondant.

**Exemple 5:**

**<!ELEMENT personne (nom|prenom, age)>**

**<!ELEMENT nom (#PCDATA) >**

**<!ELEMENT prenom (#PCDATA) >**

**<!ELEMENT age (#PCDATA) >**

Cela indique que la balise personne peut contenir seulement la balise nom ou la balise prenom.

Exemple :

**<personne>**

**<nom>DOE</nom>**

**<age>24</age>**

**</personne>**

Ou

**<personne>**

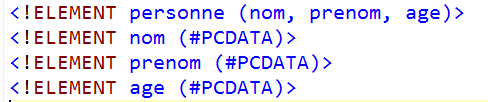
**<prenom>John</prenom>**

**<age>24</age>**

**</personne>**

**Motivation :**

**A partir du document DTD suivant vérifier la validité des documents XML :**

****

|  |  |
| --- | --- |
| **1)**  **<personne>**  **<nom>DOE</nom>**  **<prenom>John</prenom>**  **<age>24</age>**  **</personne>** | **2)**  **<personne>**  **<prenom>John</prenom>**  **<nom>DOE</nom>**  **<age>24</age>**  **</personne>** |
| **3)**  **<personne>**  **<prenom>John</prenom>**  **<age>24</age>**  **</personne>** | **4)**  **<personne>**  **<nom>DOE</nom>**  **<prenom>John</prenom>**  **<age>24</age>**  **<date>12/12/2012</date>**  **</personne>** |

### Optionalité d’un élément dans un document XML.

***?  : autorise zéro ou un élément.***

***\* : autorise zéro ou plusieurs éléments.***

***+  : autorise un ou plusieurs éléments.***

**Exemple 1 : *\* : autorise zéro ou plusieurs éléments.***

**<!ELEMENT repertoire (personne\*)>**

**<!ELEMENT personne (nom, prenom)>**

**<!ELEMENT nom (#PCDATA)>**

**<!ELEMENT prenom (#PCDATA)>**

Cela indique que la balise **repertoire** peut être vide ou peut contenir un ou plusieurs balise **personne**.

Exemple de document XML valides :

**<?xml version="1.0" encoding="UTF-8" standalone='yes'?>**

**<!DOCTYPE repertoire[**

**<!ELEMENT repertoire (personne\*)>**

**<!ELEMENT personne (nom, prenom)>**

**<!ELEMENT nom (#PCDATA)>**

**<!ELEMENT prenom (#PCDATA)>**

**]>**

**<repertoire></repertoire>**

**Exemple 2 :**

**<repertoire>**

**<personne>……</personne>**

**</repertoire>**

**Exemple 3 :**

**<repertoire>**

**<personne>……</personne>**

**<personne>……</personne>**

**<personne>……</personne>**

**</repertoire>**

**Exemple 2: *+  : autorise un ou plusieurs éléments.***

**<!ELEMENT repertoire (personne+)>**

**<!ELEMENT personne (nom, prenom)>**

**<!ELEMENT nom (#PCDATA)>**

**<!ELEMENT prenom (#PCDATA)>**

**Exemple 1 :**

**<repertoire>**

**<personne>……</personne>**

**</repertoire>**

**Exemple 2 :**

**<repertoire>**

**<personne>……</personne>**

**<personne>……</personne>**

**<personne>……</personne>**

**</repertoire>**

**En fin la balise repertoire ne peut en aucun cas être vide.**

**Exemple 3: *?  : autorise zéro ou un élément.***

**<!ELEMENT repertoire (personne ?)>**

**<!ELEMENT personne (nom, prenom)>**

**<!ELEMENT nom (#PCDATA)>**

**<!ELEMENT prenom (#PCDATA)>**

**Exemple 1 :**

**<repertoire>**

**</repertoire>**

**Exemple 2 :**

**<repertoire>**

**<personne>……</personne>**

**</repertoire>**

**Remarque : La balise vide est déclarée dans le DTD comme suit :**

**<!ELEMENT repertoire EMPTY>**

**Exercices d’application :**

**TP2 : EX1-EX2-EX3.**

### Déclaration des attributs dans un document DTD

On déclare un attribut comme suit dans un document DTD :

**< !ATTLIST balise attributs type mode>**

**Exemple 1 : Cas d’un seul attribut :**

<email type="professionnel">contact@poppins.fr</email>

**<!ATTLIST email type(CDATA) #IMPLIED>**

* CDATA : affecter une chaîne de caractères à un attribut.
* Mode : Le niveau de nécessité de l'attribut :
  + **#IMPLIED** signifie que l'attribut est optionnel, c'est-à-dire non obligatoire

**<!ATTLIST email type (CDATA) #IMPLIED>**

* + **#REQUIRED** signifie que l'attribut est obligatoire

**<!ATTLIST email type(CDATA) # REQUIRED** **>**

* + **#FIXED** signifie que l'attribut sera affecté d'une valeur par défaut s'il n'est pas défini. Il doit être immédiatement suivi de la valeur entre guillemets.

**<!ATTLIST email type(CDATA) # FIXED "Professionnel">**

**Dans le cas où l’attribut est identifiant on peut utiliser le ID s’il s'agit d'un identifiant unique obligatoire**

**Exemple :**

**<! ATTLIST disque IDdisk ID #REQUIRED >**

**Application:**

<?xml version="1.0" encoding="UTF-8"?>

<compilation>

<mp3 id="1">

<titre>Foule sentimentale</titre>

<artiste>Alain Souchon</artiste>

</mp3>

<mp3 id="2">

<titre>Solaar pleure</titre>

<artiste>MC Solaar</artiste>

</mp3>

<mp3 id="3">

<titre>Le baiser</titre>

<titre>Le baiser</titre>

<artiste>Alain Souchon</artiste>

</mp3>

<mp3 id="4">

<titre>Pourtant</titre>

<artiste>Vanessa Paradis</artiste>

</mp3>

<mp3 id="5">

<titre>Chambre avec vue</titre>

<artiste>Henri Salvador</artiste>

</mp3>

</compilation>

Ecrire le script DTD correspondant pour valider ce document XML.

**Exemple 2 : Cas de plusieurs attributs :**

Dans le cas où la balise a plusieurs attributs ; on les déclare comme suit :

Exemple :

**<img src= "monimage.jpg" alt= "problème de chargement de l’image"/>**

**Le DTD de cette balise est :**

< !ELEMENT img EMPTY>

< !ATTLIST img

Src (CDATA) #REQUIRED

Alt (CDATA) #FIEXED **"problème de chargement de l’image">**

**Pour définir une valeur par défaut il suffit de faire suivre l'énumération par la valeur désirée entre guillemets :**

< !ATTLIST img Alt (CDATA) **"problème de chargement de l’image">**

### 5) Optionalité des attributs :

Cela permet de définir une liste de valeurs possibles pour un attribut donné, afin de limiter le choix de l'utilisateur. La syntaxe de ce type d'attribut est :

**<!ATTLIST telephone type (fixe|portable) #REQUIRED>**

Exercices d’application : TP2- EX4-EX5

# XSD [(XML Schema Definition)](https://searchmicroservices.techtarget.com/definition/XSD-XML-Schema-Definition)

### Définition :

Comme le DTD, le XSD est script qui permet de définir le schéma XML et décrire la structure du document XML :

* Détailler les éléments et les attributs qui peuvent être dans un document XML.
* Le nombre des éléments enfants ou leur séquence.
* Les types de données pour les éléments et les attributs.
* Les valeurs par défaut et fixes pour les éléments et les attributs.

**Exemple :**

**<?xml version= "1.0" encoding="UTF-8"?>**

**<xsd:xmlschema xmlns:xsd="www.3org/2001/XMLschema">**

**<xsd:element name="boutique">**

**<xsd:complexetype>**

**<xsd:sequence minoccurs="1" maxoccurs="1">**

**<xsd:element name="telephone">**

**<xsd:complexetype>**

**<xsd:sequence minoccurs="1" maxoccurs="unbounded">**

**<xsd:element name="marque" type="xsd:string"/>**

**<xsd:element name="modele" type="xsd:string"/>**

**</xsd:sequence>**

**</xsd:complexetype>**

**</xsd:element>**

**</xsd:sequence>**

**</xsd:complexetype>**

**</xsd:element>**

**</xsd:xmlschema>**

### Création d’un fichier XSD :

* Démarrer le Rapid PHP /File/New/Text Document
* Enregistrer le fichier sous le nom boutique.xsd.

Un document XSD est constitué par :

* **Le prologue :**

**<?xml version= "1.0" encoding="UTF-8"?>**

* **La racine de schéma XSD :**

**<xsd:xmlschema xmlns:xsd="www.3org/2001/XMLschema">**

**……………………**

**</xsd:xmlschema>**

Cette racine définit l’espace de nom qui sera utilisé dans le document XSD, dans notre cas l’espace de nom est :**xsd**; autrement tous les éléments xsd doivent commencer par xsd.

Pour associer le fichier \*.xsd au fichier XML en question, on ajoute son lien dans la racine du fichier XML.

**<boutique xmlns:xsd="www.3w.org/2001/XMLchema-instance" xsd:schemaLocation="boutique.xsd">**

**……**

**</boutique> (Voir le fichier boutique.xml)**

### Déclaration des éléments XSD :

#### Déclaration d’un élément simple:

Un élément est dit simple si cet élément ne contient ni attribut ni autres éléments.

Sa déclaration xsd est :

**<xsd:element name="mon\_nom" type="xsd:mon\_type" />**

**Dont :**

* **Name :** contient le nom de la balise,
* **Type :**  est le type de son contenu :

**Les types xsd sont :**

**xs:string**

**xs:decimal**

**xs:integer**

**xs:boolean**

**xs:date**

**xs:time**

**Exemple :**

**<marque>Samsung</marque>**

Le code xsd correspondant est :

**<xsd:element name="marque" type="xsd:string"/>**

**Valeur par défaut et valeur inchangeable**

* **Valeur par défaut :**

<xsd:element name="prenom" type="xsd:string" default="Robert" />

* **Valeur constante :**

<xsd:element name="prenom" type="xsd:string" fixed="Robert" />

Exercice d’application : TP3 Ex1-EX2-**EX3 (les balises simples).**

#### Déclaration d’un élément complexe ou séquence d’éléments:

Un élément complexe est élément XML qui contient des attributs ou d’autres éléments XML.

**Exemple 1:**

**<telephone>**

**<marque>Samsung</marque>**

**<modele>Galaxy S3</modele>**

**</telephone>**

La balise téléphone est un élément complexe, elle contient des autres balises.

Le script XSD corréspendant est :

<xsd:element name="telephone">

<xsd:complexetype>

<xsd:sequence minoccurs="1" maxoccurs="unbounded">

<xsd:element name="marque" type="xsd:string"/>

<xsd:element name="modele" type="xsd:string"/>

</xsd:sequence>

</xsd:complexetype>

</xsd:element>

**Les étapes de déclaration des éléments complexes :**

* **On déclare l’élément : telephone.**

<xsd:element name="telephone">

/xsd:element>

* **Puisque c’est un élément complexe  on lui ajoute la balise :**

**<xsd:complexetype>**

**----------------------------**

**</xsd:complexetype>**

* **Dans cette dernière balise on ajoute la ligne :**

<xsd:sequence minoccurs="1" maxoccurs="unbounded">

Qui signifie : **L’occurrence min et l’occurrence max ; le nombre de répitition des élément XML dans le document XML.**

**Unbounded** : infinit.

**Exemple 2:**

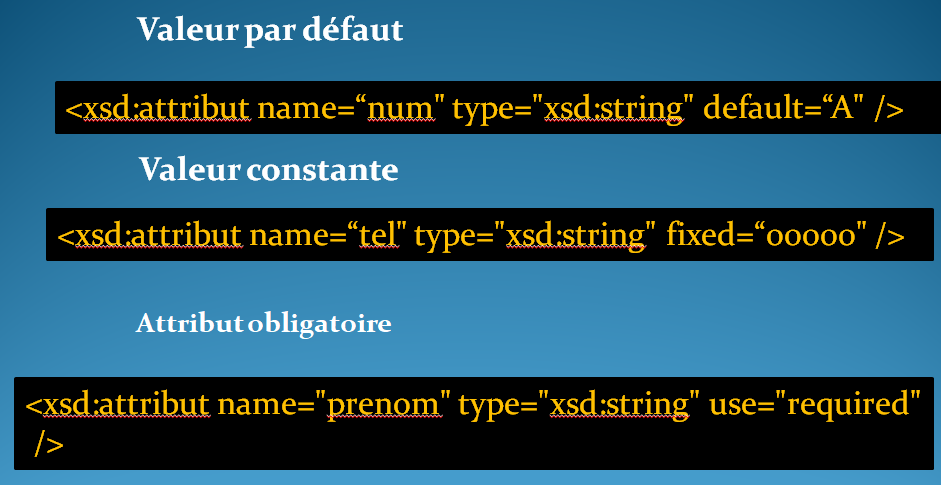
**<nombre valeur="10">dix</nombre>**

La balise nombre est un élément complexe, elle contient des attributs (Valeur).

Dans ce cas la déclaration de l’élément XSD est comme suit :

**<xsd:attribut name="sexe" type="xsd:string" />**

**Remarque :**



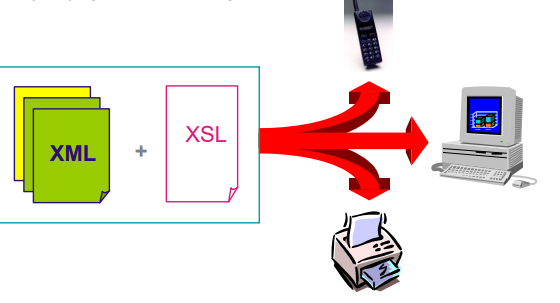
**Exercices d’application TP3.**

# Le langage XSLT(Extensible Style Language Transformation

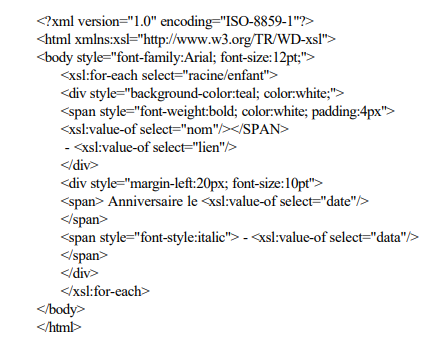
# Ou Les feuilles de style du XML

### Définition :

Le XSL est un langage de feuille de style qui permet d’afficher le contenu d’un fichier XML dans un document HTML ou autres.



Exemple :



**Le corps du document est constitué par des balises html, CSS et des instructions XSL**

### Création d’un fichier XSL :

1. Démarrer le Rapid PHP /File/New/Text Document
2. Enregistrer le fichier sous le nom boutique.xsl.

Un document XSL est constitué par :

* **Le prologue :**

**<?xml version="1.0" encoding="UTF-8"?>**

* **La Racine XSL :**

**<xsl:stylesheet version ="1.0" xmlns:xsl="http://www.3w.org/1999/XSL/Transform">**

**<xsl:template match="/">**

**………………………….**

**</xsl:stylesheet>**

* **La première ligne : définit l’espace de nom utilisé du document XSL.**
* **La deuxième ligne : L'élément <xsl: template> permet de créer un modèle de document. L’attribut match est utilisé pour associer un modèle à un élément XML. La valeur de l'attribut match est une expression Xpath ; autrement il définit la racine de recherche dans le document XML ; match="/" signifie la recherche se fait à partir de la racine.**

Pour associer un document XSL à un document XML on ajoute la ligne suivante au fichier XML.

**<?xml-stylesheet version="1.0" type="text/xsl" href="listes.xsl"?>**

### Pourquoi le xsl :

Avec les scripts XSL on peut :

* Afficher le contenu d’un document XML ou une partie de ce document.
* Trier le contenu du document XML.
* Filtrer les données du document XML selon des critères.
* Retenir des données selon des tests conditionnels.

**Exemple : Document XML demo.xml**

**<?xml version="1.0"?>**

**<?xml-stylesheet href="fichierxsl.xsl"?>**

**<demoXML>**

**<message>Voici du XML</message>**

**</demoXML>**

**Document XSL : demo.xsl**

**<?xml version="1.0"?>**

**<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >**

**<xsl:template match="/">**

**<html>**

**<body>**

**<xsl:value-of select="demoXML/message"/>**

**</body>**

**</html>**

**</xsl:template>**

**</xsl:stylesheet>**

### Les instructions XSL

**<xsl:value-of select="Xpath"/>**

Xpath (Voir plus loin) est une expression qui permet de sélectionner un ou plusieurs éléments dans l’arborescence du document XML.

Cette règle XSL permet de récupérer la valeur d’une balise XML ou plusieurs.

**Exemple 1:**

**<xsl:value-of select="demoXML/message"/> :**

Permet de récupérer la valeur de la balise message de demoXML.

**Exemple 2 :**

**<xsl:value-of select="compilation/mp3/@id"/>**

Permet de récupérer la valeur de l’attribut de la balise mp3.

**Exemple 3 :**

**<xsl:value-of select="count(compilation/mp3)"/>**

Permet de récupérer la nombre de balise mp3dans le document XML.

**<xsl:for-each select=“Xpath">**

La règle xsl **for-each** permet de parcourir une portion d’un nœud sélectionné par l’expression xpath.

**Exemple 1:**

**<xsl:for-each select="//mp3">**

**<xsl:value-of select="titre"/>**

**</xsl:for-each>**

**Cette règle permet d’afficher les titres des mp3.**

**Exemple 2:**

**<xsl:for-each select="//mp3">**

**<xsl:value-of select="@id"/>**

**</xsl:for-each>**

**Cette règle permet d’afficher les valeurs de l’attribut ID des mp3.**

**Exemple 3:**

**<xsl:text>Les Attributs Id de mp3</xsl:text>**

**Cette règle permet d’afficher un texte dans un document HTML.**

**<xsl:text>Les Attributs Id de mp3</xsl:text>**

**<xsl:for-each select="//mp3">**

**<p> <xsl:value-of select="@id"/></p>**

**</xsl:for-each>**

Ou

**<h2>Les Attributs Id de mp3</h2>**

**<xsl:for-each select="//mp3">**

**<p> <xsl:value-of select="@id"/></p>**

**</xsl:for-each>**

**Exemple général : Afficher le contenu du ducument XML demo2.xsl**

<h2>Liste des mp3 </h2>

<table>

<tr>

<th>ID</th>

<th>titre</th>

<th>artiste</th>

</tr>

<xsl:for-each select="//mp3">

<tr>

<td>

<xsl:value-of select="@id"/>

</td>

<td>

<xsl:value-of select="titre"/>

</td>

<td>

<xsl:value-of select="artiste"/>

</td>

</tr>

</xsl:for-each>

</table>

Triage des données d’un document XML :

Pour trier le contenu d’un document XML ; on ajoute à la règle

<xsl:for-each select="//mp3">

La régle :

**<xsl:sort select="@id" order="ascending"/>**

Trier la liste par id en ordre croissant.

**<xsl:sort select="titre" order=" descending"/>**

Trier la liste par titre en ordre décroissant.

**Exemple :**

**<h2>Liste des mp3 </h2>**

**<table>**

**<tr>**

**<th>ID</th>**

**<th>titre</th>**

**<th>artiste</th>**

**</tr>**

**<xsl:for-each select="//mp3">**

**<xsl:sort select="@id" order="descending"/>**

**<tr>**

**<td>**

**<xsl:value-of select="@id"/>**

**</td>**

**<td>**

**<xsl:value-of select="titre"/>**

**</td>**

**<td>**

**<xsl:value-of select="artiste"/>**

**</td>**

**</tr>**

**</xsl:for-each>**

**</table>**

**<xsl:if test="titre='Foule sentimentale'">…</<xsl:if>**

**Cette règle XSL permet de réaliser un Test sur des données XML ; elle ne contient pas de else.**

**Exemple 1:**

<h2>Liste des mp3 </h2>

<table>

<tr>

<th>ID</th>

<th>titre</th>

<th>artiste</th>

</tr>

<xsl:for-each select="//mp3">

<xsl:sort select="@id" order="descending"/>

<xsl:if test="titre='Foule sentimentale'">

<tr>

<td>

<xsl:value-of select="@id"/>

</td>

<td>

<xsl:value-of select="titre"/>

</td>

<td>

<xsl:value-of select="artiste"/>

</td>

</tr>

</xsl:if>

</xsl:for-each>

</table>

**<xsl:choose>**

**Cette règle permet de réaliser plusieurs test sur les données d’un document XML.**

**Exemple 2:**

**<xsl:choose>**

**<xsl:when test="@id='1'">**

**......................**

**</xsl:when>**

**<xsl:when test="@id='2'">**

**......................**

**</xsl:when>**

**<xsl:otherwise ">**

**......................**

**</xsl:otherwise>**

### Les variables en XSL :

En XSL ; une variable est déclaré comme suit :

**<xsl:variable name="x" select="3"/>**

**Ou**

**<xsl:variable name="x"/>**

Comme on peut le déclaré comme un block :

**<xsl:variable name="x">**

**-------------------------**

**</xsl:variable>**

Exemple :

<xsl:variable name="y">

<h1>Je suis un Titre</h1>

<h2>Je suis un sous titre</h2>

</xsl:variable>

On appelle cette variable par: $y.

**<xsl:copy-of select="$y"/>**

### Les fonctions XSL :

C’est une fonction identifié par **name** (**unique dans le fichier XSL**) et qui permet d’exécuter un bloc d’instructions.

Exemple :

<xsl:template name="hello">

<xsl:text>Bonjour mes amis</xsl:text>

</xsl:template>

On appelle cette template par :

<xsl:call-template name="hello"/>

Exemple complet:

<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/">

<html>

<body>

<h2>Liste des mp3 </h2>

<xsl:variable name="y">

<h1>Je suis un Titre</h1>

<h2>Je suis un sous titre</h2>

</xsl:variable>

<xsl:variable name="x" select="3"/>

<xsl:call-template name="hello"/>

<xsl:for-each select="//mp3">

<xsl:choose>

<xsl:when test="@id=$x">

<xsl:value-of select="titre"/>

<xsl:value-of select="artiste"/>

</xsl:when>

</xsl:choose>

</xsl:for-each>

</body>

</html>

</xsl:template>

<xsl:template name="hello">

<xsl:text>Bonjour mes amis</xsl:text>

</xsl:template>

</xsl:stylesheet>

Pour créer une Template paramétrée, on utilise :

**<xsl:template name="somme">**

**<xsl:param name="x" select="0"/>**

**<xsl:param name="y" select="0"/>**

**<xsl:text>x</xsl:text>**

**<xsl:value-of select="$x"/>**

**<xsl:text>+ y</xsl:text>**

**<xsl:value-of select="$y"/>**

**<xsl:text>=</xsl:text>**

**<xsl:value-of select="$x+$y"/>**

**</xsl:template>**

Pour appeler la Template :

**<xsl:call-template name="somme">**

**<xsl:with-param name="x" select="10"/>**

**<xsl:with-param name="y" select="100"/>**

**</xsl:call-template>**

On peut utiliser aussi des autres formes de Template sur les nœuds.

**<xsl:template match="//mp3">**

**<li>**

**<xsl:value-of select="@id"/>**

**</li>**

**</xsl:template>**

**On appelle la Template par :**

**<ol>**

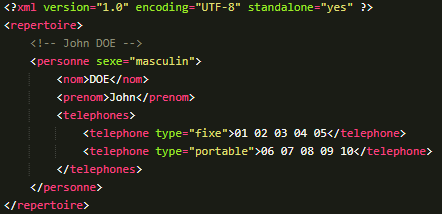
**<xsl:apply-templates select="//mp3"/>**

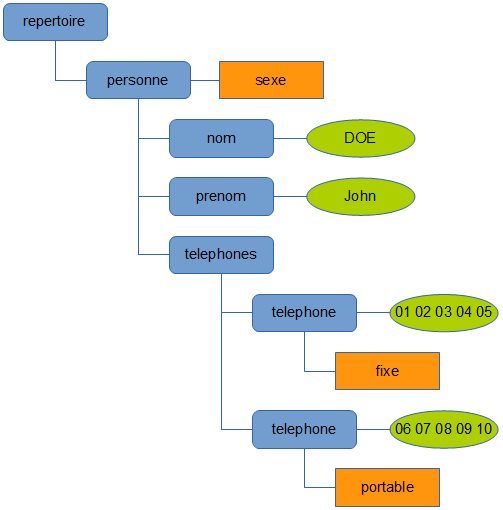
**</ol>**

# Les expressions XPATH

### Définition

Le xpath est un langage qui permet de parcourir les éléments et les attributs d’un document XML à partir de la racine (Racine, Element, Attribut, Text, espace de nom, commentaire..).





### La syntaxe de Xpath:

Le xpath utilise des expressions Xpath pour sélectionner un nœud ou plusieurs dans un document XML.

|  |  |
| --- | --- |
| **Expression** | **Signification** |
| (name) Nom de l’élément | Sélectionner les éléments de même nom (name) |
| / | La sélection commence à partir du nœud racine |
| // | La sélection commence à partir du nœud actuel |
| . | sélectionne le nœud actuel |
| .. | Sélectionne le parent du nœud actuel |
| @ | Sélectionner les attributs |

Exemple :

Utilisons le site : <http://xpather.com/>

Dans l’éditeur copier le document suivant :

<?xml version="1.0" encoding ="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="mp3.xslt"?>

<compilation>

<mp3 id="1">

<titre>Foule sentimentale</titre>

<artiste>Alain Souchon</artiste>

</mp3>

<mp3 id="2">

<titre>Solaar pleure</titre>

<artiste>MC Solaar</artiste>

</mp3>

<mp3 id="3">

<titre>Le baiser</titre>

<artiste>Alain Souchon</artiste>

</mp3>

<mp3 id="4">

<titre>Pourtant</titre>

<artiste>Vanessa Paradis</artiste>

</mp3>

<mp3 id="5">

<titre>Chambre avec vue</titre>

<artiste>Henri Salvador</artiste>

</mp3>

</compilation>

Tester les expressions xpath suivantes :

* //compilation
* //mp3 ou /compilation/mp3
* //mp3[@id='1'].