XML – présentation

https://www.w3schools.com/xml/default.asp

- XML : eXtensible Markup Language est un langage informatique de balisage générique
- Le XML a été créé pour **faciliter les échanges de données** entre les machines et les logiciels.
- Le XML est un langage qui s'écrit à l'aide de **balises**.
- Le terme **générique** signifie que nous allons pouvoir créer nos propres balises (à la différence du HTML).
- Le XML est une recommandation du **W3C**, il s'agit donc d'une technologie avec des règles strictes à respecter.
- Le XML se veut compréhensible par tous : les hommes comme les machines.
- Le XML est un langage qui permet de décrire des données à l'aide de **balises** et de règles que l'on peut personnaliser.

XML - outils

- Editeur : notepad++, Jedit, SublimeText,
- Editix

XML - balises, attributs, commentaires

- Deux types de balises existent : les **balises par paires** et les **balises uniques** (orphelines).
- Les balises peuvent contenir des **attributs**.
- Un document XML peut contenir des **commentaires**.

Balises

<balise>contenu</balise> : sensible à la casse, toute balise ouverte doit être fermée.
<balise1> <balise2> contenu </balise2> </balise1> : arborescence
<balise /> : balise unique, sans contenu

- 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 les programmes.

Attributs

<prix devise="euro" moyen_paiement="cheque">25.3</prix>

- La valeur d'un attribut doit impérativement être délimitée par des guillemets, simples ou doubles.

Commentaires

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

XML - prologue et corps

- Un document XML est composé de 2 parties : le **prologue** et le **corps**.
- un document XML doit être bien formé pour être exploitable.
- **EditiX** permet de vérifier qu'un document est bien formé en seulement quelques clics.
-

Prologue

<?xml version = "1.0" encoding="UTF-8" standalone="yes" ?>
v1.0 : la plus utilisée. Il existe aussi 1.1
UTF-8 : par défaut
standalone : document autonome (rien de rattaché) : facultatif
Le fichier doit commencer par ce prologue
Il peut y avoir d'autres lignes de prologue : < ?xml ?> : pour lier du CSS ou du XSL

Corps

<racine>
<balise_paire>texte</balise_paire>
<balise_paire2>texte</balise_paire2>
</racine>

Document bien formé : Editix

Certains éditeurs fournissent des outils pour vérifier la syntaxe : Editix par exemple.

XML et Navigateur

On peut afficher un fichier XML dans le navigateur. Avec Firefox, ça présente la hiérarchie des balises qu'on peut ouvrir et fermer. C'est pareil avec les fichiers JSON.

- Exemple

XML et CSS

On peut associer un fichier CSS au fichier XML pour avoir une présentation améliorée.
Il faut rajouter une ligne de prologue :
<?xml-stylesheet type="text/css" href="css.css" ?>

- Exemple

XML et HTML : XSL

https://www.w3schools.com/xml/xsl_intro.asp

- On peut afficher un fichier XML comme un fichier HTML. Pour cela il faut l'associer à un fichier XLS qui est une sorte de fichier HTML qui va faire le lien avec le fichier XML.
- Le XSL (ou XSLT, c'est la même chose) permet par exemple d'afficher une table avec tous les éléments de la table en faisan référence au XML, comme on le fait avec le PHP en utilisant la BD.
- XSL gère donc des boucles, mais aussi des tests.
- Il utilise le langage XPATH pour circuler dans l'arbre des balises : https://www.w3schools.com/xml/xpath_intro.asp
- Les exemples du cours permettent de voir les différentes possibilités.

XML et JavaScript

- Un fichier XML est exploitable en JavaScript comme un fichier HTML : toutes les fonctionnalités du DOM lui sont applicables.

DTD : Document Type Definition : <!ELEMENT : les balises

https://www.w3schools.com/xml/xml_dtd_intro.asp

Définition

Le DTD est un document qui décrit la syntaxe de nos balises dans notre document XML.
Il est constitué de règles : au moins une par balise.
Dans la DTD on doit donner des règles pour toutes les balises qui apparaissent.
L'intérêt du DTD est d'éviter qu'un document XML ne soit pas conforme à ce qu'on attend de lui. C'est une sorte de vérificateur de syntaxe.

Exemple : DTD interne, dans le fichier XML

```
< !DOCTYPE racine [
        <!ELEMENT
        …
]>
```

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

<!DOCTYPE boutique [
  <!ELEMENT boutique (telephone*)>
  <!ELEMENT telephone (marque, modele)>
  <!ELEMENT marque (#PCDATA)>
  <!ELEMENT modele (#PCDATA)>
]>

<boutique>
  <telephone>
    <marque>Samsung</marque>
    <modele>Galaxy S3</modele>
  </telephone>

  <telephone>
    <marque>Apple</marque>
    <modele>iPhone 4</modele>
  </telephone>

  <telephone>
    <marque>Nokia</marque>
    <modele>Lumia 800</modele>
  </telephone>
</boutique>
```

Présence d'une balise

```
<!ELEMENT mabalise >
```

Cette règle signifie que la balise < mabalise /> existe.

Cas d'une balise contenant une autre balise

```
<!ELEMENT personne (nom)>
```

Cette règle signifie que la balise <personne /> contient la balise <nom />.
Le document XML respectant cette règle ressemble donc à cela :

```
<personne>
    <nom>toto</nom>
</personne>
```

Cas où une balise contient une valeur simple
on utilisera la mot clef #PCDATA

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

Cas d'une balise vide

```
<!ELEMENT nom EMPTY>
```

Cas d'une balise pouvant tout contenir

```
<!ELEMENT nom ANY>
```

Cas d'une balise contenant une suite ordonnée de balise : séquence

```
<!ELEMENT balise (balise2, balise3, balise4, balise5, etc.)>
```

exemple :

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

Cas d'une balise optionnelle dans la séquence
on ajoute un ?

```
<!ELEMENT balise (balise2, balise3?, balise4, balise5, etc.)>
```

Cas d'une balise répétée et optionnelle dans la séquence
on ajoute un *

```
<!ELEMENT balise (balise2, balise3*, balise4, balise5, etc.)>
```

Cas d'une balise répétée et non optionnelle dans la séquence
on ajoute un +

```
<!ELEMENT balise (balise2, balise3+, balise4, balise5, etc.)>
```

Cas d'une balise contenant une balise et une seule parmi une liste

```
<!ELEMENT balise (balise2 | balise3 | balise4 | balise5 | etc.)>
```

DTD : Document Type Definition : <!ATTLIST : les attributs

Exemple XML

```
<personne sexe="masculin" />
```

Syntaxe de base des attributs

```
<!ATTLIST balise attribut type mode>
```

on précise le nom de la balise, de l'attribut, son type et son mode

Types possibles
Liste de valeurs

```
<!ATTLIST balise attribut (valeur 1 | valeur 2 | valeur 3 | etc.)
    mode>
```

Texte quelconque (non parsé) : CDTA

```
<!ATTLIST balise attribut CDATA mode>
```

Valeur unique : ID

```
<!ATTLIST balise attribut ID mode>
```

IDREF : faire référence à l'id d'un parent

```
<!ATTLIST balise attribut ID mode baliseParent IDREF mode>
```

par exemple :

```
<father id="PER-1" ></father>
<child id="PER-2" father="PER-1" ></child>
```

Modes possibles

Obligatoire : REQUIRED

```
<!ATTLIST personne sexe (masculin|féminin) #REQUIRED>
```

Optionnel : IMPLIED

```
<!ATTLIST personne sexe CDATA #IMPLIED>
```

Valeur par défaut : en dur, entre guillemets

```
<!ATTLIST objet devise CDATA "Euro">
```

Constante fixé : FIXED + en dur, entre guillemets

```
<!ATTLIST objet devise CDATA #FIXED "Euro">
```

DTD : Document Type Definition : <!ENTITY : les entités

Définition

Les entités sont des alias.

Les entités du XML

Elles peuvent apparaître dans les documents XML.

Exemple :

```
<!ENTITY samsung "Samsung">
<!ENTITY apple "Apple">

<telephone>
    <marque>&samsung;</marque>
    <modele>Galaxy S3</modele>
</telephone>
<telephone>
    <marque>&apple;</marque>
    <modele>iPhone 4</modele>
</telephone>
```

Les entités du DTD

Syntaxe :

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

Exemple :

```
<!ENTITY % listeMarques "marque (Samsung|Apple) #REQUIRED">
<!ATTLIST telephone %listeMarques; >
```

DTD externe

Le DTD peut être dans un fichier séparé.
Le fichier XML y fait référence dans un DOCTYPE.

Document XML + DTD

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

<!DOCTYPE boutique SYSTEM "doc1.dtd">

<boutique>
  <telephone>
  etc.
```

Syntaxe

```
<!DOCTYPE baliseRacine SYSTEM "URI">
```

Schéma XML : XSD

https://www.w3schools.com/xml/schema_intro.asp

Défintion
Les schémas XML sont une autre technique pour écrire des DTD.
Elle offre plus de possibilités que les DTD.

- Les **Schémas XML** offrent plus de possibilités que les DTD : ils permettent entre autre de typer les données.
- Les **Schémas XML** s'écrivent à l'aide d'un langage de type XML : à l'aide de balises.
- Un fichier dans lequel est écrit un **Schéma XML** porte l'extension **".xsd"**.

Intérêt
Un schéma XSD permet avec ses règles de valider le fichier XML.

Les éléments simples
- Un **élément simple** est un élément qui ne contient qu'une valeur dont le type est dit simple comme par exemple une balise qui ne contient aucun attribut et dans laquelle aucune autre balise n'est imbriquée.
- Pour décrire un élément simple, on utilise la balise<xsd:element />.
- Il est possible de définir une valeur par défaut « default » ou une valeur constante « fixed » à un élément simple.
- XML

```
<nom>DUPONT</nom>
<prenom>Robert</prenom>
<age>38</age>
```

- XSD

```
<xsd:element name="nom" type="xsd:string" defaut="toto" />
<xsd:element name="prenom" type="xsd:string" />
<xsd:element name="age" type="xsd:int" ></xsd:element>
```

Les attributs
- Un **attribut** est également un élément simple.
- Pour décrire un **attribut**, on utilise la balise<xsd:attribut />.
- Il est possible de préciser si l'attribut a une valeur par défaut « default », une valeur constante « fixed » ou est obligatoire « required ».
- XML

```
<personne sexe="masculin">Robert DUPONT</personne>
```

- XSD

```
<xsd:attribut name="sexe" type="xsd:string" use="required" />
```

Les types simples
- Il existe 4 grandes familles de types simples : les chaînes, les dates, les numériques et tous les autres.
- Ces familles permettent d'être très précis quant à la description des éléments simples et des attributs.

```
<xsd:element name="nom" type="xsd:string" />
```

Les éléments complexes
- Un **élément complexe** est un élément qui contient d'autres éléments ou des attributs.

- Un **élément complexe** est décrit grâce à la balise<xsd:complexType />.
- Un **élément complexe** a 3 types de contenus possibles : les **contenus simples**, **"standards"** et **mixtes**.

Autres subtilités
- Le nombre d'occurrences d'un élément s'exprime grâce aux mots clefs **minOccurs** et **maxOccurs**.
- Le mot clef **ref** permet de faire référence à des éléments dans le but de les réutiliser plusieurs fois au sein du Schéma XML.
- L'**héritage** permet de réutiliser des éléments d'un Schéma XML pour en construire de nouveaux.
- Il existe 2 types d'héritages : l'**héritage par restriction** et l'**héritage par extension**.

Exemple : schéma XML du répertoire :
Où l'on comprend pourquoi le XML n'a plus le vent en poupe !!!

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- balises isolées -->
    <xsd:element name="nom" type="xsd:string"/>
    <xsd:element name="prenom" type="xsd:string"/>

    <!-- balises d'une adresse -->
    <xsd:element name="numero" type="xsd:string"/>
    <xsd:element name="voie">
        <xsd:complexType>
       <xsd:simpleContent>
           <xsd:extension base="xsd:string">
            <xsd:attribute name="type">
                <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="impasse"/>
                <xsd:enumeration value="avenue"/>
                <xsd:enumeration value="rue"/>
                            </xsd:restriction>
          </xsd:simpleType>
            </xsd:attribute>
        </xsd:extension>
       </xsd:simpleContent>
   </xsd:complexType>
    </xsd:element>
    <xsd:element name="codePostal">
        <xsd:simpleType>
       <xsd:restriction base="xsd:string">
           <xsd:pattern value="[0-9]{5}"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>
    <xsd:element name="ville" type="xsd:string"/>
    <xsd:element name="pays" type="xsd:string"/>

    <!-- balise adresse -->
    <xsd:element name="adresse">
        <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="numero"/>
           <xsd:element ref="voie"/>
           <xsd:element ref="codePostal"/>
```

```
            <xsd:element ref="ville"/>
            <xsd:element ref="pays"/>
        </xsd:sequence>
    </xsd:complexType>
    </xsd:element>

    <!-- balise telephone -->
    <xsd:element name="telephone">
    <xsd:complexType>
            <xsd:simpleContent>
            <xsd:extension base="xsd:string">
             <xsd:attribute name="type">
                 <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                   <xsd:enumeration value="fixe"/>
                 <xsd:enumeration value="portable"/>
                 <xsd:enumeration value="bureau"/>
               </xsd:restriction>
           </xsd:simpleType>
             </xsd:attribute>
        </xsd:extension>
            </xsd:simpleContent>
    </xsd:complexType>
    </xsd:element>

    <!-- balise telephones -->
    <xsd:element name="telephones">
        <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="telephone" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    </xsd:element>

    <!-- balise email -->
    <xsd:element name="email">
        <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
             <xsd:attribute name="type">
                 <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                   <xsd:enumeration value="personnel"/>
                 <xsd:enumeration value="professionnel"/>
               </xsd:restriction>
           </xsd:simpleType>
             </xsd:attribute>
        </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    </xsd:element>

    <!-- balise emails -->
    <xsd:element name="emails">
    <xsd:complexType>
            <xsd:sequence>
            <xsd:element ref="email" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    </xsd:element>

    <!--balise personne -->
    <xsd:element name="personne">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="nom"/>
                <xsd:element ref="prenom"/>
                <xsd:element ref="adresse"/>
                <xsd:element ref="telephones"/>
                <xsd:element ref="emails"/>
            </xsd:sequence>

            <!-- attribut sexe -->
            <xsd:attribute name="sexe">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="masculin"/>
                        <xsd:enumeration value="feminin"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>

    <!-- Schéma XML -->
    <xsd:element name="repertoire">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element  ref="personne"  maxOccurs="unbounded"
    />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

XML – TP

TP-0

Installer Editix

TP-1

Exercice 1 :

Le principe de l'organisation XML, c'est une arborescence.
Créer un document XML structurant les données d'un répertoire.
Votre répertoire doit comprendre au moins 5 personnes.
Pour chaque personne, on souhaite connaître les informations suivantes :

- Son nom
- Son prénom
- Son adresse
- Son sexe
- Un ou plusieurs numéros de téléphone (téléphone portable, fixe, bureau, etc.)
- Une ou plusieurs adresses e-mail (adresse personnelle, professionnelle, etc.)

Exercice 2 :

Faire une version JSON de votre document XML.

Exercice 3 :

Ajouter un fichier CSS.

Exercice 4 :

Faire une version XSL avec un fichier CSS.

TP-2

Choisir l'organisation de données que vous voulez et créer le XML et le JSON

TP-3

Récupérer un fichier XML sur internet à partir d'un flux RSS. Donnez la structure minimum du fichier (la structure minimum, c'est celle avec un seul « objet » et le maximum d'attributs).
Faire une version JSON de la structure minimum.
Ajouter un fichier CSS.

TP-4

Faire le DTD du TP1 (ou d'un autre !)

Exercice 1

Modéliser des articles avec bibliographie

L'objectif de l'exercice est de proposer un format XML permettant de stocker des articles quelconques.

- Un article est constitué d'un titre, d'un texte et d'une bibliographie ;
- le texte lui-même est une succession de paragraphes, chaque paragraphe pouvant contenir :
  - des mots ou expressions importants et devant donc être différenciés du reste du paragraphe ;
  - des références bibliographiques ;
- une entrée dans la bibliographie peut décrire soit un site web, soit un ouvrage ;
- un site web est décrit par un nom et une url ;
- pour un ouvrage, on trouve le titre, les auteurs, la date de parution et l'éditeur.

Questions :

1. Discuter des différentes possibilités de codage en XML.
2. Écrire une DTD et un document respectant cette DTD contenant au moins deux paragraphes et trois entrées bibliographiques (en utilisant les deux types d'entrées possibles).
3. Concevoir une feuille de style CSS permettant de mettre en valeur le document.

Exercice 2

Modéliser un site de brèves

Un site d'actualités veut présenter des nouvelles brèves, regroupées par thème. Quatre thèmes sont possibles : *actualités, sport, bourse* et *média*. Chaque brève correspond à un unique thème.

Les brèves peuvent être rédigées en français ou anglais, chacune est datée et possède un titre. Il est également possible d'illustrer une brève par une photo et de fournir une ou plusieurs urls vers des sites détaillant l'information : chaque url sera agrémentée d'une courte phrase résumant le contenu de la page pointée.

1. Discuter des différentes possibilités de codage en *XML*, en particulier pour la prise en compte de la langue et des thèmes.
2. Écrire une *DTD* et un document respectant cette *DTD* contenant au moins deux brèves.

Exercice 3

Modéliser le championnat de football

Fournir une DTD qui décrit comment stocker les différents éléments d'un championnat de football : l'année, la ligue concernée, les différentes journées avec leurs numéros, les dates et bien sûr les rencontres avec leurs résultats.

Exercice 4

Modéliser des acteurs de cinéma

Fournir une DTD qui décrit comment représenter des acteurs dans un format XML. Pour chaque acteur, on veut pouvoir donner des éléments de son état-civil, sa photo, son site web ainsi qu'une biographie.

La biographie doit autoriser un contenu mixte : des paragraphes de texte pouvant contenir des références à d'autres acteurs, des titres de films, des années, des portions de phrase plus importantes que les autres, etc.

Exercice 5

Modéliser les films de cinéma

Proposer une DTD qui permette de décrire plusieurs films dans un même fichier XML.

Un attribut permettra pour chaque film de préciser sa langue d'origine. En outre, on pourra préciser le titre du film, une photo, le réalisateur, le casting et un synopsis.

Le casting est suite d'acteurs, chacun associé au nom du personnage joué.

Enfin, le synopsis est un texte décrivant brièvement le scénario et faisant éventuellement référence aux personnages décrits dans le casting.