Schéma XML : introduction

- Les **DTD** ont quelques défauts:

1. **Nouveau format:** DTD n'est pas au format XML. C'est un nouveau langage avec sa syntaxe et ses propres règles.
2. **Typage de données:** DTD ne permettent pas de typer les données. Impossible de préciser pour une balise si on souhaite que ça soit nombre entier, date, ou chaine de caractères

- Les schémas XML permettent de typer les données. On peut même créer nos propres types de données.

Espaces de nom

Lorsque l'on écrit un document XML, on utilise ce que l'on appelle un **vocabulaire**. Par exemple, chaque personne possède :

- Une identité (un nom et un prénom).
- Une adresse.
- Des numéros de téléphone.
- Des adresses e-mail.
- Etc.

A travers cette description, nous avons défini le vocabulaire d'une personne.

Il existe plusieurs vocabulaires qui ont fait leurs preuves et qui ont été mis à dispositions des développeurs afin qu'ils puissent être réutilisés. Nous pouvons en citer quelques uns.

- Le vocabulaire permettant de décrire une **page xHTML**.
- Le vocabulaire permettant de décrire un **Schéma XML**.
- Le vocabulaire permettant de décrire des documents techniques.
- Etc.

Identifier un espace de noms

Un espace de noms est identifié par une **URI** (Uniform Resource Identifier) qui permet de l'identifier de manière unique.

Exemples

- xhtml : http://www.w3.org/1999/xhtml
- Schéma XML : http://www.w3.org/2001/XMLSchema
- DocBook : http://docbook.org/ns/docbook

Exemple d'un document xHTML :

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Titre du document</title>
    </head>
    <body>
        <p>
            <img src="mon_image.png" alt="ma super image" />
            <br/>
            <a href="mon_lien">Mon super lien !</a>
        </p>
    </body>
</html>
```

- Tous les éléments utilisés dans ce document XML comme<head />,<title />,<body />, etc., font partie du vocabulaire d'une page xHTML.
- C'est grâce à la déclaration de l'espace de noms dans l'élément<html />que nous pouvons utiliser les différents éléments du vocabulaire tout en respectant les règles qui régissent leurs imbrications les uns par rapport aux autres.

Les espaces de noms avec préfixe

- Il est impossible d'utiliser au sein d'un même document 2 espaces de noms par défaut qui auraient un mot de vocabulaire en commun.
- La déclaration d'un **espace de noms avec préfixe** se fait dans le premier élément qui utilise le vocabulaire, grâce au mot clef **xmlns:prefixe**.

xmlns:prefixe="mon_uri"

```
<http:html xmlns:http="http://www.w3.org/1999/xhtml">
    <http:head>
        <http:title>Titre du document</http:title>
    </http:head>
    <http:body>
        <http:p>
            <http:img src="mon_image.png" alt="ma super image"
></http:img>
            <http:br></http:br>
            <http:a href="mon_lien">Mon super lien !</http:a>
        </http:p>
    </http:body>
</http:html>
```

Structure d'un Schéma XML:

- L'extension des fichiers (Schémas XML)

Un fichier dans lequel est écrit un Schéma XML porte l'extension "**.xsd**". Ainsi, la première ligne d'un Schéma XML est :

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

- Le corps

L'élément racine a un nom imposé : **xsd:schema**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Elément racine -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  xsd:schema>
```

On remarque la présence de l'attribut **xmlns:xsd**. **xmlns** nous permet de déclarer un **espace de noms**. A travers la déclaration, tous les éléments doivent commencer par **xsd:**

Référencer un schéma XML

Le **référencement d'un schéma XML** se fait au niveau de l'élément racine du **fichier XML** grâce à l'utilisation de 2 attributs.

1) L'espace de noms

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

2) La location

Il nous permet d'indiquer à notre fichier XML où se situe le fichier du Schéma XML.

- 2 possibilités s'offrent alors à nous : les schémas XML qui décrivent un espace de noms et ceux qui ne décrivent pas un espace de noms.

Schéma XML décrivant un espace de noms

```
xsi:schemaLocation="chemin_vers_fichier.xsd">
```

Schéma XML ne décrivant pas un espace de noms

On utilisera alors la syntaxe suivante :

```
xsi:noNamespaceSchemaLocation="chemin_vers_fichier.xsd">
```

Résumé

Pour résumer, voici ce à quoi nos fichiers XML ressembleront :

```
<?xml version="1.0" encoding="UTF-8"?>
<racine xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="chemin_vers_fichier.xsd">
</racine>
```

Schéma XML : les éléments simples

Les éléments simples

Définition

Un **élément simple** est un élément qui ne contient qu'une valeur dont le type est dit **simple**. Il ne contient pas d'autres éléments.

Un **élément simple** peut donc être une balise qui ne contient aucun attribut et dans laquelle aucune autre balise n'est imbriquée.

Un attribut d'une balise peut également être considéré comme un élément simple. En effet, la valeur d'un attribut est un type simple.

Un **type simple**, c'est par exemple un chiffre, une date ou encore une chaîne de caractères.

Quelques exemples

Prenons quelques exemples de fichiers XML, et regardons ensemble s'ils peuvent être considérés comme des types simples :

```
<!-- Ne contient ni attribut ni aucun autre élément => élément simple -->
<nom>ROBERT</nom>
<!-- Contient un attribut => n'est pas un élément simple -->
<!-- Cependant l'attribut "sexe" est un élément simple -->
<personne sexe="masculin">Robert DUPONT</personne>
<!-- La balise personne contient d'autres éléments (les balises nom et prénom) => n'est pas un élément simple -->
<personne>
   <!-- Ne contient ni attribut ni aucun autre élément => élément simple -->
   <nom>DUPONT</nom>
    <!-- Ne contient ni attribut ni aucun autre élément => élément simple -->
    <prenom>Robert</prenom>
</personne>
```

Déclarer une balise comme un élément simple

Si vous souhaitez **déclarer une balise en tant qu'élément simple**, c'est le mot clef **element** qu'il faut utiliser. N'oubliez pas de précéder son utilisation par **xsd:**

Cette balise prend 2 attributs : un nom et un type.

```
<xsd:element name="mon_nom" type="xsd:mon_type" />
```

Voyons tout de suite un exemple. Soit les éléments simples suivants :

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

Au sein d'un Schéma XML, les éléments précédents seront déclarés de la sorte :

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

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

<xsd:element name="age" type="xsd:int" ></xsd:element>
```

Que sont ces int et ces strings ?

String est utilisé pour qualifier **une chaîne de caractères** et **int** est utilisé pour qualifier **un nombre entier**.

Valeur par défaut et valeur inchangeable

Valeur par défaut

L'attribut **default** qui est utilisé au niveau de la balise`<element />`du Schéma XML.

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

Voici alors quelques exemples de documents XML possibles :

```
<!-- valide -->
<prenom>Jean</prenom>
<!-- valide -->
<prenom>Marie</prenom>

<!-- valide -->
<!-- la balise prenom vaut "Robert" -->
<prenom ></prenom>
```

Valeur constante

S'il est possible d'indiquer une valeur par défaut, il est également possible d'imposer une valeur. Cette valeur inchangeable est appelée **constante**.

Pour indiquer une **valeur constante**, c'est l'attribut **fixed** qui est utilisé au niveau de la balise`<element />`du Schéma XML. Par exemple, si je souhaite obliger toutes les personnes de mon document XML à porter le prénom Robert, voici la règle à écrire :

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

Voyons alors la validité des lignes XML suivantes :

```
<!-- valide -->
<prenom>Robert</prenom>
```

```
<!-- invalide -->
<prenom>Marie</prenom>

<!-- invalide -->
<prenom></prenom>
```

Les attributs

C'est le mot **attribut** qui est utilisé pour le déclarer.

Cette balise prend 2 attributs : un *nom* et un *type*.

```
<xsd:attribut name="mon_nom" type="xsd:mon_type" />
```

Prenons par exemple la ligne XML suivante :

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

On ne va pas détailler ici comment déclarer la balise. En effet, puisqu'elle contient un attribut, c'est ce qu'on appelle un **élément complexe** et nous verrons comment faire un peu plus tard. Cependant, voici comment déclarer l'attribut dans notre Schéma XML :

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

Valeur par défaut, obligatoire et inchangeable

Pour indiquer une **valeur par défaut**, c'est l'attribut **default** qui est utilisé au niveau de la balise`<attribut />`du Schéma XML.

```
<xsd:attribut name="sexe" type="xsd:string" default="masculin" />
```

Valeur constante

S'il est possible d'indiquer une valeur par défaut, il est également possible d'imposer une valeur. Cette valeur inchangeable est appelée **constante**.

Pour indiquer une **valeur constante**, c'est l'attribut **fixed** qui est utilisé au niveau de la balise`<attribut />`du Schéma XML.

```
<xsd:attribut name="sexe" type="xsd:string" fixed="feminin" />
```

Attribut obligatoire

Les attributs sont, par défaut, optionnels.

Pour indiquer qu'un attribut est *obligatoire*, nous devons renseigner la propriété **use** à laquelle nous affectons la valeur **required**.

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

Schéma XML : les types complexes

Définition

Un élément qui possède un attribut n'est plus un élément simple. On parle alors d'**élément complexe**.

Les éléments complexes

Un **élément complexe** est un élément qui contient d'autres éléments ou des attributs. Bien évidemment les éléments contenus dans un élément peuvent également contenir des éléments ou des attributs.

Quelques exemples d'éléments XML qui dans un Schéma XML sont considérés comme complexes :

```
<!-- la balise personne contient d'autres balises => élément complexe -->
<personne>
    <!-- la balise nom est un élément simple -->
    <nom>ROBERT</nom>
    <!-- la balise prenom est un élément simple -->
    <prenom>Axel</prenom>
</personne>
<!-- la balise personne possède un attribut => élément complexe -->
<personne sexe="feminin">Axel ROBERT</personne>
```

Déclarer un élément complexe

Si vous souhaitez déclarer une balise en tant qu'élément complexe, c'est le mot clef **complexType** qu'il faut utiliser associé à celui que nous connaissons déjà : **element**. N'oubliez pas de précéder son utilisation par **xsd:**

```
<xsd:element name="mon_nom">
    <xsd:complexType>
        <!-- contenu ici -->
    <xsd:complexType>
</xsd:element>
```

Nous reviendrons juste après sur la notion de **contenu**,

Reprenons l'un des éléments de type complexe que nous avons vu un peu plus haut:

```
<personne>
    <nom>ROBERT</nom>
    <prenom>Axel</prenom>
</personne>
```

Voici comment le déclarer :

```
<xsd:element name="personne">
    <xsd:complexType>
        <!-- contenu ici -->
    </xsd:complexType>
</xsd:element>
```

Les contenus des types complexes

Concernant les types complexes, il est important de noter qu'il existe 3 types de contenus possibles :

- Les contenus **simples**.
- Les contenus **"standards"**.
- Les contenus **mixtes**.

Les contenus simples

Définition

On appelle **contenu simple**, le contenu d'un élément complexe qui n'est composé que d'attributs et d'un texte de type simple.

Quelques exemples

Je vous propose de voir quelques exemples d'éléments complexes dont le contenu est dit simple.

```
<!-- contient un attribut et du texte -->
<prix devise="euros">35</prix>
<!-- contient un attribut et du texte -->
<voiture marque="Renault">Clio</voiture>
```

Du côté du Schéma XML

La syntaxe

Pour déclarer un élément complexe faisant référence à une balise contenant des attributs et du texte, voici la syntaxe à utiliser :

```
<xsd:element name="mon_nom">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="mon_type">
                <xsd:attribute name="mon_nom" type="mon_type" ></xsd:attribute>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

Un exemple

Reprenons l'exemple d'un prix prenant pour attribut une devise :

```
<prix devise="euros">35</prix>
```

Voici alors le schéma XML associé :

```
<xsd:element name="prix">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:positiveInteger">
                <xsd:attribute name="devise" type="xsd:string" ></xsd:attribute>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

Dans le cas où la balise que l'on cherche à décrire contient plusieurs attributs, il convient de tout simplement les lister entre les balises`<xsd:extension/>`. Par exemple :

```
<voiture marque="Renault" type="essence">Clio</voiture>
<xsd:element name="voiture">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="marque" type="xsd:string" ></xsd:attribute>
                <xsd:attribute name="type" type="xsd:string" ></xsd:attribute>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

Comme vous pouvez le constater, on se contente de mettre à la suite les différents attributs qui composent l'élément. À noter que l'ordre dans lequel les attributs sont déclarés dans le Schéma XML n'a aucune importance.

Les contenus "standards"

Définition

Il est important de noter que cette appellation n'est nullement officielle. C'est une appellation maison car il s'agit du cas de figure qui à tendance à revenir le plus souvent.

Contenu "standard", c'est le contenu d'un élément complexe qui n'est composé que d'autres éléments (simples ou complexes) ou uniquement d'attributs.

Quelques exemples

Comme pour le contenu simple, voyons quelques exemples de contenu "standard" :

```
<!-- contient d'autres éléments -->
<personne>
    <nom>DUPONT</nom>
    <prenom>Robert</prenom>
</personne>
<!-- contient un attribut -->
<voiture marque="Renault" ></voiture>
```

Balise contenant un ou plusieurs attributs

Reprenons l'exemple de notre voiture du dessus :

```
<voiture marque="Renault" ></voiture>
```

Voici alors le Schéma XML associé :

```
<xsd:element name="voiture">
    <xsd:complexType>
        <xsd:attribute name="marque" type="xsd:string" ></xsd:attribute>
    </xsd:complexType>
</xsd:element>
```

Il n'y a, pour le moment, rien de bien compliqué. On se contente d'imbriquer une balise`<xsd:attribute />`dans une balise`<xsd:complexType />`.

```
<voiture marque="Renault" modele="Clio" ></voiture>
```

Regardons alors le Schéma XML :

```
<xsd:element name="voiture">
    <xsd:complexType>
        <xsd:attribute name="marque" type="xsd:string" ></xsd:attribute>
        <xsd:attribute name="modele" type="xsd:string" ></xsd:attribute>
    </xsd:complexType>
</xsd:element>
```

L'ordre dans lequel les balises`<xsd:attribute />`sont placées n'a aucune importance.

Balise contenant d'autres éléments

Il est maintenant temps de passer à la suite et de jeter un coup d'œil aux balises qui contiennent d'autres éléments.

La séquence

Une **séquence** est utilisée lorsque l'on souhaite spécifier que les éléments contenus dans un type complexe doivent apparaître dans un ordre précis.

Voici comment se déclare une **séquence** au niveau d'un Schéma XML :

```
<xsd:element name="mon_nom">
    <xsd:complexType>
        <xsd:sequence>
            <!-- liste des éléments -->
        </xsd:sequence>
```

```
        <!-- listes des attributs -->
    </xsd:complexType>
</xsd:element>
```

Voyons tout de suite un exemple :

```
<xsd:element name="personne">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="nom" type="xsd:string"></xsd:element>
            <xsd:element name="prenom" type="xsd:string"></xsd:element>
        </xsd:sequence>
        <xsd:attribute name="sexe" type="xsd:string" ></xsd:attribute>
    </xsd:complexType>
</xsd:element>
```

Cet extrait signifie que la balise`<personne />`qui possède l'attribut **sexe**, contient les balises`<nom />`et`<prenom />`dans cet ordre.

Illustrons alors cet exemple :

```
<!-- valide -->
<personne sexe="masculin">
    <nom>DUPONT</nom>
    <prenom>Robert</prenom>
</personne>

<!-- invalide => les balises nom et prenom sont inversées -->
<personne sexe="masculin">
    <prenom>Robert</prenom>
    <nom>DUPONT</nom>
</personne>
```

Le type all

Le type **all** est utilisé lorsque l'on veut spécifier que les éléments contenus dans un type complexe peuvent apparaître dans n'importe quel ordre. Ils doivent cependant tous apparaître une et une seule fois.

Voici comment se déclare le type **all** au niveau d'un Schéma XML :

```
<xsd:element name="mon_nom">
    <xsd:complexType>
        <xsd:all>
            <!-- liste des éléments -->
        </xsd:all>
        <!-- listes des attributs -->
    </xsd:complexType>
```

```
</xsd:element>
```

Voyons tout de suite un exemple :

```
<xsd:element name="personne">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="nom" type="xsd:string"></xsd:element>
            <xsd:element name="prenom" type="xsd:string"></xsd:element>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
```

Cet extrait signifie donc que la balise`<personne />`contient les balises`<nom />`et`<prenom />`dans n'importe quel ordre.

Illustrons alors cet exemple :

```
<!-- valide -->
<personne sexe="masculin">
    <nom>DUPONT</nom>
    <prenom>Robert</prenom>
</personne>

<!-- valide -->
<personne sexe="masculin">
    <prenom>Robert</prenom>
    <nom>DUPONT</nom>
</personne>
```

Le choix

Un **choix** est utilisé lorsque l'on veut spécifier qu'un élément contenu dans un type complexe soit choisi dans une liste pré-définie.

Voici comment se déclare un **choix** au niveau d'un Schéma XML :

```
<xsd:element name="mon_nom">
    <xsd:complexType >
        <xsd:choice>
            <!-- liste des éléments -->
        </xsd:choice>
        <!-- listes des attributs -->
    </xsd:complexType>
</xsd:element>
```

Voyons sans plus tarder un exemple :

```
<xsd:element name="personne">
    <xsd:complexType>
        <xsd:choice>
```

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

Cet extrait signifie donc que la balise`<personne />`contient soit la balise`<nom />`, soit`<prenom />`.

Illustrons cet exemple :

```
<!-- valide -->
<personne sexe="masculin">
    <nom>DUPONT</nom>
</personne>
<!-- valide -->
<personne sexe="masculin">
    <prenom>Robert</prenom>
</personne>
<!-- invalide => les 2 balises prenom et nom ne peuvent pas apparaître en même temps --
>
<personne sexe="masculin">
    <prenom>Robert</prenom>
    <nom>DUPONT</nom>
</personne>
```

Cas d'un type complexe encapsulant un type complexe

Prenons par exemple le document XML suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<personne>
    <identite>
        <nom>NORRIS</nom>
        <prenom>Chuck</prenom>
    </identite>
</personne>
```

Ce document XML permet d'identifier une personne via son nom et son prénom. Voyons alors le Schéma XML qui définit notre document XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="personne">
        <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="identite">
            <xsd:complexType>
                <xsd:sequence>
```

```
                <xsd:element name="nom"  type="xsd:string"></xsd:element>
                <xsd:element name="prenom"  type="xsd:string"></xsd:element>
            </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

En soit, il n'y a rien de compliqué. Il convient juste de repérer que lorsque l'on place un élément complexe au sein d'un autre élément complexe, dans notre cas, une **identité** dans une **personne**, il convient d'utiliser une **séquence**, un **choix** ou un type **all**.

Les contenus mixtes

Définition

Il est temps de conclure ce chapitre avec le dernier type de contenu possible : les **contenus mixtes**.

Un **contenu mixte** est le contenu d'un élément complexe qui est composé d'attributs, d'éléments et de texte.

Un exemple

Afin d'illustrer cette définition, je vous propose de nous appuyer sur un exemple :

```
<balise1>
  Ceci est une chaîne de caractères
  <balise2>10</balise2>
  7.5
</balise1>
```

Du côté du Schéma XML

La syntaxe

Pour déclarer un élément complexe au contenu mixte, voici la syntaxe à utiliser :

```
<xsd:element name="mon_nom">
    <xsd:complexType  mixed="true">
        <!-- liste des éléments -->
    </xsd:complexType>
    <!-- liste des attributs -->
</xsd:element>
```

La nouveauté est donc l'utilisation du mot clef **mixed**.

Un exemple

Prenons l'exemple d'une facture fictive dans laquelle on souhaite identifier l'acheteur et la somme qu'il doit payer.

```
<facture><acheteur>Zozor</acheteur>, doit payer <somme>1000</somme>€.</facture>
```

Voici comment le traduire au sein d'un Schéma XML :

```
<xsd:element name="facture">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="acheteur" type="xsd:string" ></xsd:element>
            <xsd:element name="somme" type="xsd:int" ></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Comme vous pouvez le remarquer, j'ai utilisé la balise`<xsd:sequence />`pour encapsuler la liste des balises contenues dans la balise`<facture />`, mais vous pouvez bien évidemment adapter à votre cas de figure et choisir parmi les balises que nous avons vu dans le chapitre précédent, à savoir :

- `<xsd:sequence />`.
- `<xsd:all />`.
- `<xsd:choice />`.

En résumé

- 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**.