# XML-DTD-XSD-XPATH-XSLT

**Les Difinitions :**

1) XML : « extensible Mark up language »
2) DTD : « Document Type Définition »
3) XSD : « XML Schéma Définition »
4) XSLT : « eXtensible Stylesheet Language Transformations»
5) Xpath : « est le langage de parcours de l'arbre XML et de désignation des noeuds. »

## *I)DTD :*

**Role** : Le rôle de la DTD est de définir toutes les balises que l'on pourra utiliser pour créer des documents .

**Les Types De DTD :** Il y'a deux Type de DTD sont Interne et externe

| Interne(Syntaxe) | Externe(Syntaxe) |
|---|---|
| < !Doctype **Element_Racine**[<br><-- Declaration des elmts ou bien des att -- ><br>]> | **DTD Prive** (Fichier local) :< !Doctype **Element_Racine** Système «fichier .dtd» ><br><br>**DTD Public** (URL) :< !Doctype **Element_Racine** Public«URL» > |

### Déclaration des éléments :

**Exemple 1** (Sequence) **:**

| **Fichier XML :** | **Fichier DTD :** |
|---|---|
| <Professeur><br><Nom> Ahmed </nom><br><prenom> Ahmed </prenom><br></Professeur> | < !Doctype Professeur[<br>< !Element Professeur(nom,prenom)><br>< !Element nom(#PCDATA)><br>< !Element prenom(#PCDATA)><br>]> |

**Exemple 2** (Choix) **:**

| **Fichier XML :** | **Fichier DTD :** |
|---|---|
| <Etudiant><br><Nom> Ahmed </nom><br></Etudiant> | < !Doctype Etudiant [<br>< !Element Professeur(nom\|prenom)><br>< !Element nom(#PCDATA)><br>< !Element prenom(#PCDATA)><br>]> |

### Déclaration de structure d'élément :

1) **Séquence :** < !Element Etudiant(nom,prenom,age)>
2) **Choix :** < !Element Etudiant(nom|prenom,)>
3) **Indicateur d'occurrences :**





| | 1 | Head |
|---|---|---|
| **?** | **0 ou 1** | **Nom ?** |
| **+** | **1 ou n** | **Nom+** |
| ***** | **0 ou n** | **Nom*** |

**4)Any :** Tu peux faire n'importe quoi (des éléments ou biens des attributs)

**5)EMPtY (**élément vide**) :** Ce Type ne dit pas que il doit Etre vide définitivement ms possible d'ajouter des attributs

### Exemples de type d'attributs:

**Syntaxe :** < !Attliste nom_racine nom_attribut Type Obligation valeur_par_default >

**Spécification** : TYPE OPTIONS VALEUR_PAR_DEFAUT

**Types des Attributs :**

1) **Chaine :** < !Attliste Personne Nom CDATA #IMPLIED >
2) **Énumératif :** < !Attliste Personne Genre (M|F) #REQUIRED >

### Déclaration d'attributs : Types

1) **CDATA :** · Ce type est le plus général. Il n'impose aucune contrainte à la valeur de l'attribut. Celle-ci peut être une chaîne quelconque de caractères.
2) **(value1 | value2 | … | valueN) :** · La valeur de l'attribut doit être un des jetons value1, value2, … valueN. Comme ces valeurs sont des jetons, celles-ci ne sont pas délimitées par des apostrophes ''' ou des guillemets '"'.
3) **ID :** · La valeur de l'attribut est un nom XML. Un élément peut avoir un seul attribut de ce type
4) **IDREF :** La valeur de l'attribut est une référence à un élément identifié par la valeur de son attribut de type ID.
5) **IDREFS :** · La valeur de l'attribut est une liste de références séparées par des espaces

### Déclaration d'attributs : OPTIONS

1) **#IMPLIED** : – L'attribut est optionnel et il n'a pas de valeur par défaut. Si l'attribut est absent, il n'a pas de valeur.
2) **#REQUIRED** – L'attribut est obligatoire et il n'a pas de valeur par défaut
3) **#FIXED "value"** – La valeur value est une chaîne quelconque de caractères délimitée par des apostrophes. La valeur de l'attribut est fixée à la valeur value donnée. – Si l'attribut est absent, sa valeur est implicitement value. – Si l'attribut est présent, sa valeur doit être value pour que le document soit valide.

### EXEMPLE DES DECLARATIONS DES ELEMENTS ET ATTRIBUTS :

**Fichier XML :**

```
< Bibliographie >
  <livres>
    <livre clé="Michard01" langue="Fr" >
        <titre> Titre 1 </titre>
        <auteur>auteur 1</auteur>
        <Anne>2016</Anne>
        <publie par="id2680397"/>
    </livre>
    .
    .
    .
  </livres>
  <éditeurs>
    <éditeur id="id2680397">
        <nom>éditeur 1 </nom>
        <prénom> prénom 1</prénom>
    </éditeur>
    <éditeur id="id2680356">
        <nom>éditeur 2 </nom>
        <prénom> prénom 2</prénom>
    </éditeur>
  <éditeurs>
</Bibliographie>
```

**Fichier DTD :**

```
< !Doctype Bibliographie [
< !Element Bibliographie(Livres,editeurs)>
< !Element Livres(Livre+)>
< !Element éditeurs(éditeur+)>
< !Element Livre(titre,auteur,anne,publie)>
< !Element titre(#PCDATA)>
< !Element auteur(#PCDATA)>
< !Element Anne(#PCDATA)>
< !Element publie EMPTY>
<!Attlist publie par IDREF #Required>
< !Attlist Livre cle  CDATA #Required>
< !Attlist Livre Lan  CDATA #Required>
.
.
.
.
.
.
.
< !Attlist editeur id  IDREF #Required>
 ]>
```

---

## *I)XSD :*

### XSD : Syntaxe

| Syntaxe | EXEMPLE |
|---|---|
| <?xml version="1.0" encoding="iso-8859-1"?><br>< xsd:schema<br><-- Declaration des elmts ou bien des att -- ><br>...<br></xsd:schema> | <xsd :schema><br>< !Element name= 'Professeur'><br>< xsd :complexeType><br>< xsd :sequence><br>< !Element name= 'nom' type='xsd :string'><br>< !Element name= 'age' type='xsd :int'><br></ xsd :sequence><br>< /xsd :complexeType><br></xsd :schema> |

### XSD : Elements

**Syntaxe :** <xsd :element name='nom_element' type='type_element'>

**nom_element** : le nom du nœud

**Type_element**: le type de données de l'élément (chaines de caractéres, entier, date,





**Exemple :**

1) <xsd :element **name**='nom' **type**='xsd :string'>
2) <xsd :element **name**='age' type='xsd :int'>
3) <xsd :element **name**='DN' **type**='xsd :'date'>

### Syntaxe Valeur / Valeur par défaut

1) <xsd :element **name**='nom' **type**='xsd :string' **Default**='Soufiane'>
2) <xsd :element **name**='age' **type**='xsd :int' **Fixed**='20'>

**default**: spécifie (en absence de cet élément) la valeur par défaut

**fixed**:Spécifie une valeur fixe égale à "valeur figée"

### XSD : Eléments / Types des éléments

- Les types simples définissent uniquement des contenus textuels.
- Ils peuvent être utilisé pour les éléments ou les attributs.
- Ils sont introduits par l'élément xsd:simpleType.
- Il peut aussi être construit par union d'autres types simples ou par l'opérateur de listes.
- Les types simples / Déclaration :

```
<xsd:simpleType name="Byte">
</xsd:simpleType>
```

- Au lieu de juste PCDATA et CDATA:

```
<xsd:element name="prix" type="xsd:float"/>
```

- **TYPES COMPLEXES**

#### a. Contenu Mixte

- L'opérateur **xsd:sequence** définit un nouveau type formé d'une suite des éléments en séquence ordonnée.
- – C'est l'équivalent de l'opérateur ',' des DTD.

```
<xsd :element name='Personne'>
    <xsd :ComplexeType mixte='true'>
        <xsd :sequence >
            <xsd :element name='nom' type='xsd :String'>
            <xsd :element name='age' type='xsd :int'>
        </xsd :sequence >
    </xsd :ComplexeType >
</xsd :element >
```

### b. Opérateurs de choix

- L'opérateur **xsd:choice** définit un nouveau type formé d'une suite des éléments énumères de choix
- C'est l'équivalent de l'opérateur '|' des DTD.

```
<xsd :element name='Personne'>
    <xsd :ComplexeType mixte='true'>
        <xsd :Choice >
            <xsd :element name='nom' type='xsd :String'>
            <xsd :element name='age' type='xsd :int'>
        </xsd :Choise >
    </xsd :ComplexeType >
</xsd :element >
```

### c. Opérateurs de all

- L'opérateur **xsd:all** n'a pas d'équivalent dans les DTD. Il définit un nouveau type dont chacun des éléments doit apparaître une fois dans un ordre quelconque.

```
xsd :element name='Personne'>
    <xsd :ComplexeType mixte='true'>
        <xsd :all >
            <xsd :element name='nom' type='xsd :String'>
            <xsd :element name='age' type='xsd :int'>
        </xsd :all >
    </xsd :ComplexeType >
</xsd :element >
```

## XSD : Eléments / Indicateurs d'occurrence

- Les attributs minOccurs et maxOccurs permettent de préciser le nombre minimal ou maximal d'occurrences d'un élément ou d'un groupe
- Ils sont l'équivalent des opérateurs ?, * et + des DTD.
- Ils peuvent apparaître comme attribut des éléments xsd:element, xsd:sequence, xsd:choice et xsd:all.
- L'attribut minOccurs prend un entier comme valeur
- L'attribut maxOccurs prend un entier ou la chaîne unbounded comme valeur pour indiquer qu'il n'y a pas de nombre maximal.

## XSD : Attributs





**Syntaxe :** <xsd :attribute name='nom_ attribute' type='type_ attribute'>

**nom_element** : le nom du attribute

**Type_element**: le type de données de attribute (chaines de caractéres, entier, date, …)

```
xsd :element name='Personne'>
    <xsd :ComplexeType mixte='true'>
        <xsd :all >
            <xsd :element name='nom' type='xsd :String'>
            <xsd :element name='age' type='xsd :int'>
        </xsd :all >
            <xsd :attribute name='genre' type='xsd :string'>
    </xsd :ComplexeType >
</xsd :element >
```

## *XSD : Attributs & Options*

- – optional : si l'attribut est optionnel ( DTD : #IMPLIED )
- required : si l'attrribut est obligatoire (DTD : #REQUIRED )
- – prohibited : L'opérateur xsd:sequence définit un nouveau type formé d'une suite des éléments en séquence ordonnée

```
xsd :element name='Personne'>
            <xsd :attribute name='genre' type='xsd :string'
use='required'>
</xsd :element >
```

## XSD : Restriction de types

- La restriction est la deuxième façon d'obtenir un type dérivé à partir d'un type de base.
- L'idée générale de la restriction est de définir un nouveau type dont les contenus au sens large sont des contenus du type de base.
- La restriction d'un type est introduite par l'élément xsd:restriction dont l'attribut base donne le nom du type de base.

- Par intervalle

```
<xsd:element name="year">
 <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1970"/>
            <xsd:maxInclusive value="2050"/>
    </xsd:restriction>
 </xsd:simpleType>
</xsd:element>

<xsd:attribute name="date">
  <xsd:simpleType>
    <xsd:restriction base="xsd:date">
      <!-- Date après le 1er janvier 2001 exclus -->
      <xsd:minExclusive value="2001-01-01"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

- Par énumération

```
<xsd:element name="language" type="Language"/>
<xsd:simpleType name="Language">
  <xsd:restriction base="xsd:language">
    <xsd:enumeration value="de"/>
    <xsd:enumeration value="en"/>
    <xsd:enumeration value="fr"/>
  </xsd:restriction>
</xsd:simpleType>
```

- Par motif

```
<xsd:simpleType name="ISBN">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d-\d{2}-\d{6}-[\dX]"/>
    <xsd:pattern value="\d-\d{3}-\d{5}-[\dX]"/>
    <xsd:pattern value="\d-\d{4}-\d{4}-[\dX]"/>
    <xsd:pattern value="\d-\d{5}-\d{3}-[\dX]"/>
  </xsd:restriction>
</xsd:simpleType>
```





- . : Tout caractère autre qu'un retour à la ligne ([\n\r])
- \s : caractère d'espacement (espace U+20, tabulation U+09, saut de ligne U+0A our retour chariot U+0D.
- \S : caractère autre qu'un caractère d'espacement
- \d : chiffre
- \D : caractère autre qu'un chiffre

- \w : caractère alphanumérique ou un tiret '-'
- \W : caractère autre qu'un caractère alphanumérique ou un tiret

- \i : caractère commençant un identificateur (lettre, '_' ou ':')
- \I : caractère ne commençant pas un identificateur
- {n} : répétition n fois
- {m,n} : répétition entre m et n fois
- [x-y] : caractère entre x et y

- xsd:minInclusive, xsd:minExclusive, xsd:maxInclusive et xsd:maxExclusive
  - Donnent des valeurs minimale et maximale en incluant ou non la borne donnée.
  - S'appliquent à tous les types numériques et date et d'heure

- xsd:fractionDigits et xsd:totalDigits
  - Fixent respectivement le nombre maximal de chiffres de la partie fractionnaire (à droite de la virgule) et le nombre maximal de chiffres en tout. gule).
  - S'appliquent uniquement aux types numériques dérivés de xsd:decimal. Ceci inclut tous les types entiers mais exclut les types xsd:float et xsd:double.
- xsd:whiteSpace
  - Cette facette est particulière. Elle ne restreint pas les valeurs valides mais elle modifie le traitement des caractères d'espacement à l'analyse lexicale. Cette facette peut prendre les trois valeurs preserve, replace et collapse qui correspondent à trois modes de fonctionnement de l'analyseur lexical.

- **xsd:enumeration**
  - Permet d'énumérer explicitement les valeurs autorisées. Elle s'applique à tous les types simples y compris les types construits avec xsd:union et xsd:list.
- **xsd:pattern**
  - Permet de donner une expression rationnelle pour contraindre les valeurs.
  - Elle ne s'applique pas uniquement aux types dérivés de xsd:string mais à tous les types simples y compris les types numériques et les types contruits avec xsd:union et xsd:list.
  - L'utilisation avec xsd:decimal permet de restreindre, par exemple, aux nombres ayant 4 chiffres pour la partie entière et 2 pour la partie fractionnaire.
- **xsd:length, xsd:minLength et xsd:maxLength**
  - donnent respectivement une longueur fixe ou des longueurs minimale et maximale.

## III)XPATH

### 1) Définition:

Xpath est le langage de parcours de l'arbre XML et de désignation des noeuds.
La désignation peut se faire de plusieurs manières :
1 – Par le nom (par le chemin),
2 – Par une propriété (prédicat le qualifiant),
3 - Alternatives et combinaisons.

### 2) Xpath,Désignation par le Nom :

Le noeud de départ ou racine est donné par : "/" Si le chemin commence par '/', alors il représente un chemin absolu vers l'élément requis.
– Désignation directe : //x

Exemple : **Soit le document xml suivant :**

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<BBB/>
<DDD>
<BBB/>
</DDD>
<CCC/>
</AAA>
```





### 2) Xpath,Désignation par le Prédicat:

Fonctions de sélection:

- produit**[designation]:**

– sélectionne dans le noeud courant, l'élément *produit qui a comme* élément enfant *designation*

- personne**[@sexe]**

– sélectionne dans le noeud courant, l'élément *personne qui possède* un attribut *sexe*

- personne **[@sexe='femme']**

– sélectionne dans le noeud courant, l'élément dont l'attribut *sexe a* une valeur égale à *femme*

- **[i]:**Un nombre entre crochets donne la position d'un élément dans le jeu sélectionné. Ex : /AAA/BBB[1]
- **Position(i):** Retourne la position, ou numéro d'index, du nœud, par rapport à tous les nœuds sélectionnés dans la liste de nœuds. Ex: //B[Position()=2]

**Last():** La fonction last sélectionne le dernier élément du jeu. Ex: /AAA/BBB[last()]

- Les attributs sont spécifiés par le préfixe **@. Ex:**
  - //BBB[@id]: Sélectionne tous les BBB qui ont un attribut *id*
  - //BBB[@*] : Sélectionne tous les BBB qui ont un attribut.
  - //BBB[not(@*)] : Sélectionne tous les BBB qui n'ont pas d'attribut
  - //BBB[@name='HIND']:Sélectionne tous les éléments BBB ayant un attribut name dont la valeur est *HIND*

### 3) Xpath,Alternative et combinaison:

Plusieurs chemins peuvent être combinés avec le séparateur |

Exemple:

- //CCC | //BBB : Sélectionne tous les éléments CCC et BBB
- /AAA/EEE | //BBB: Sélectionne tous leséléments BBB et EEE quisont enfants de l'élément racine AAA

- **Count() :** compte les éléments de la sélection. **Ex:**
  - //*[count(BBB)=2] : Sélectionne les éléments ayant deux enfants BBB
  - //*[count(*)=2] : Sélectionne les éléments ayant deux enfants
- **Name() :** Nom de l'élément . **Ex:**
  - //*[Name() = "personne"]: sélectionne tous les éléments qui s'appelle personne
- **Contains:** condition sur les chaines de caractères. **Ex:**
  - //*[Contains(name()=="H"): tout les éléments qui contient le caractère H
- **Startswith():**condition sur les chaines de caractères. **Ex:**
  - //*[Startswith(name()=="M"): tout les éléments qui commence par le caractère M
- **Text() :** tous les noeuds de type textuels. **Ex:**
  - //question[2]/text() : la 2eme question sous format text

Utilisation d'opérateurs :

| Type | Opérateurs |
|---|---|
| Booléen | and, or |
| Logique | !=, >=, >, <, <= |
| Opérations | +, -, *, div, mod |

**Ex:** //*[@att='en' and name()='B']

Quelques exemples:

| Requête | Résultat |
|---|---|
| B[@att="en"] | sélectionne tous les enfants *B* du noeud contextuel qui ont un attribut *att* ayant la valeur *en* |
| D[@id="10"] [1] | sélectionne le 1er enfant *D* du noeud contextuel qui ont un attribut *id* ayant la valeur *10* |
| D[1][@id="10"] | sélectionne le 1er enfant *D* du noeud contextuel si celui-là a un attribut *id* dont la valeur est *10* |
| //B[C="Hello"] | sélectionne tous les enfants *B* qui ont au moins un enfant *C* dont le contenu textuel est *Hello* |
| B[C] | sélectionne les enfants *B* du noeud contextuel qui ont au moins un enfant *C* |
| B[C and @att] | sélectionne tous les enfants *B* du noeud contextuel qui ont simultanément au moins un enfant *C* et un attribut *att* |





## III)XLST

### 3) Structure d'une feuille de style XSLT:

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

<!-- les règles de transformations -->
  .....
</xsl:stylesheet>
```

**Remarque:**
Toutes les instructions XSL appartiennent à l'espace de noms xsl, elles commencent toutes par **xsl**:

### 4) Liaison d'une feuille de style XSLT avec un document XML :

```
<?xml-stylesheet type="text/xsl" href="annuaire.xsl"?>
```

### 5) Format d'une règle de transformation:

Un **motif** est une expression qui identifie le/les noeud(s) XML du document qui est/sont concerné(s) par la règle et sur le(s) quel(s) il faut appliquer une action.

```
<xsl:template match= 'un motif'>
   [action]
</xsl:template>
```

**Remarque :**

- La balise **template** indique le format de transformation
- L'attribut **match** mentionne les éléments qui sont concernés par la transformation,
- **/** indique l'élément racine et tous les sous éléments

**La balise : <xsl:value-of select='personnes/nom'> ....</xsl:value-of>**

- La balise **<xsl:value-of>** permet de sélectionner un élément du fichier XML pour le traiter dans le fichier XSL.
- Dans l'attribut **select**, on détermine le chemin d'accès vers la balise XML souhaitée.

**La balise : <xsl:foreach select='annuaire/personne'> ... </xsl:foreach >**

xsl:for-each est une boucle traitant tous les noeuds xml qui lui sont soumis, car en XSLT il n'existe pas de boucle indexée(for i=.. to ..) ;
le chemin xpath lui est fourni dans son attribut select . Le code xslt encadré par les balises xsl:for-each sera donc appliqué sur chaque élément "visité" : l'élément courant

**Exemple :**

**Soit le document XML suivant :**

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="annuaire.xsl"?>
 <annuaire>
<personne>
<nom >Hajjar</nom>
 <prenom>Hind</prenom>
<telephone>02 96 45 87 34</telephone>
</personne>
<personne>
<nom >Elouafi</nom>
<prenom>Mouad</prenom>
<telephone>03 45 67 25 99</telephone>
</personne>
```

**Fichier XSLT correspondant:**

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<html>
<body>
<table border="1"cellspacing="0" cellpadding="3">
<tr>
<td>Nom</td>
<td>Prénom</td>
<td>Numéro téléphone</td>
</tr>
<xsl:for-each select="annuaire/personne">
<tr>
<td><xsl:value-of select="nom"/></td>
<td><xsl:value-of select="prenom"/></td>
<td><xsl:value-of select="telephone"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```





**1) Définition:**

Xpath est le langage de parcours de l'arbre XML et de désignation des noeuds.
La désignation peut se faire de plusieurs manières :
1 – Par le nom (par le chemin),
2 – Par une propriété (prédicat le qualifiant),
3 - Alternatives et combinaisons.

**2) Xpath,Désignation par le Nom :**

Le noeud de départ ou racine est donné par : "**/**" Si le chemin commence par '/', alors il représente un chemin absolu vers l'élément requis.
– Désignation directe : **//x**

Exemple : **Soit le document xml suivant :**

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<BBB/>
<DDD>
<BBB/>
</DDD>
<CCC/>
</AAA>
```

**Résultat en HTML :**

| Nom | Prénom | Numéro telephone |
|---|---|---|
| Hajjar | Hind | 02 96 45 87 34 |
| Elouafi | Mouad | 03 45 67 25 99 |

## 6) Les autres possibilités du langage XSLT:

Le langage XSLT permet aussi de :

- **trier** les données XML en ordre croissant ou décroissant.
- **filtrer** des éléments XML en fonction de certains critères.
- **choisir** des éléments.
- retenir des éléments par des **tests conditionnels.**

## 7) Trier avec XSLT:

On utilise l'élément: <xsl:sort> </xsl:sort>
<xsl:sort : est un élément vide qui contient des attributs :

- **Select**="element qu'on veut trier'
- **Order**='**ascending/descending'**
- **Case_order**='**Upper_First / Lower_First**'

Cet élement est utilisé juste après la balise <xsl:foreach> </xsl:foreach>

## 8) Filtrer avec XSLT:

Le langage XSLT permet de filtrer les données du fichier XML
associé selon les critères:
**égal, pas égal, plus grand que, plus petit que**
Pour ce faire, on utilise l'attribut:
**select="chemin_d'accès[balise='xxx']"**
Les opérateurs possibles sont :
**=** pour égal.
**!=** pour différent.
**>** pour plus grand que.
**<** pour plus petit que.

Exemple :

```
xsl:for-each select="annuaire/personne[prénom='Mouad']">
<tr>
<td><xsl:value-of select="telephone"/></td>
</tr>
</xsl:for-each>
```

## 9) Choix avec XSLT:

La balise <xsl:if> ... </xsl:if> permetd'effectuer un choix dans les données du fichier XML.
On ajoute l'attribut match où l'on indique l'élément choisi.

```
<xsl:if match=".[balise='xxx']">
  ……
  ……
</xsl:if>
```





Exemple :

```
<xsl:for-each select="annuaire/personne">
<xsl:if match=".[Nom='Elouafi ']">
<tr>
<td><xsl:value-of select="Telephone"/></td>
<td><xsl:value-of select="prenom"/></td>
</tr>
</xsl:if>
</xsl:for-each>
```

## 10) Le choix conditionnel:

**Syntaxe :**

```
<xsl:choose>.
<xsl:when  test='Condition'
  ..... Action....
</xsl:when>
<xsl:when  test='Condition'
  ..... Action....
</xsl:when>
xsl:otherwise une autre action
</xsl:choose>
```

Exemple :

```
<xsl:for-each select="annuaire/personne">
<xsl:choose>
<xsl:when
test=".[Nom='Hajjar'']">
<tr >
<td><xsl:value-of select="prénom"/></td>
</tr>
</xsl:when>
<xsl:when
test=".[Nom='Elouafi'']">
<tr >
<td><xsl:value-of select="prénom"/></td>
</tr>
</xsl:when>

<xsl:otherwise>
<tr>
<td><xsl:value-of select="telephone"/></td>
</tr>
</xsl:otherwise>
```