

## Le XSL

Le XSL, pour *eXtensible Stylesheet Language* (langage extensible de feuilles de style), est une recommandation du consortium W3C datant de novembre 1999. Il s'agit d'un standard dans le domaine de la publication sur le Web. Le XSL est en quelque sorte le langage de feuilles de style du XML. Un fichier de feuilles de style XSL reprend des données XML et produit la présentation ou l'affichage de ce contenu XML selon les souhaits du concepteur de la page web.

Le XSL comporte en fait trois langages :

- Le XSLT (*XSL Transform*), langage qui transforme un document XML en un format, généralement en HTML ou XHTML, reconnu par un navigateur.
- Le Xpath langage qui permet de définir et d'adresser des parties de document XML.
- Le XSL-FO (*eXtensible Stylesheet Language - Formatting Objects*) est le vocabulaire qui décrit les mises en forme de documents XML quel que soit le support : écran, papier, audio, etc.

Pour la suite de ce tutoriel, nous nous limiterons aux langages XSLT et Xpath. Et, comme dans la littérature relative à ce sujet, ils seront identifiés sous le vocable général de XSL.

Le XSL est dérivé du XML, reprenant ainsi toutes les règles de syntaxe de ce dernier (détaillées au chapitre Introduction au XML).

Soit en bref :

- les balises sensibles à la casse, s'écrivent en minuscules ;
- toutes les balises ouvertes doivent être impérativement fermées ;
- les balises vides auront aussi un signe de fermeture soit `<balise/>` ;
- les balises doivent être correctement imbriquées ;
- les valeurs des attributs doivent toujours être mises entre guillemets ;
- le document XSL devra être "bien formé".

Le XSL ne permet pas uniquement l'affichage de XML. Il permet aussi :

- de sélectionner une partie des éléments XML ;
- de trier des éléments XML ;
- de filtrer des éléments XML en fonction de certains critères ;
- de choisir des éléments ;
- de retenir des éléments par des tests conditionnels ;
- de transformer un document.

## Un premier document XSL

Avant de débiter, il est utile de préciser que :

- Le XSL est dérivé du XML. Le document XSL reprend donc la structure et la syntaxe de n'importe quel document XML.
- Le document XSL comporte un document HTML ou XHTML qui sera quant à lui reconnu par le navigateur, et qui servira de support à tout ou partie des données du document XML associé.
- Le XSL fonctionne avec une ou plusieurs **templates**, sortes de gabarit, pour définir comment afficher des éléments du fichier XML. Les éléments concernés du fichier XML sont déterminés par l'attribut `match`.

Voici un premier document XSL. Sa conception est basique mais n'est pas compliquée ; ce document sera étoffé en cours d'étude.

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

Le XSL est dérivé du XML. Aussi, il est normal que le document XSL commence par la déclaration de document XML, soit `<?xml version="1.0"?>`.

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

Cette ligne déclare que le document est au format XSL.

L'attribut `xmlns` fait référence à l'espace de nommage (*namespace*) utilisé. Le *namespace* officiel du W3C est :

```
xmlns:xsl="http://www.w3.org/1999/XSL/ Transform"
<xsl:template match="/">
```

Voici une balise `template` et son attribut `match`.

Cette balise `template` va déterminer un gabarit dans lequel des éléments du fichier XML sont transformés sous une forme affichable par le navigateur.

Les éléments du fichier XML sont déterminés par l'attribut `match="/"`. La barre oblique ("`/`") entre guillemets signale que toutes les balises XML du document associé à partir de la racine sont concernées.

```
<html>
<head>
<title>XSL</title>
</head>
<body>
```

Il s'agit du début de la partie HTML ou XHTML servant de support à l'affichage du document dans le navigateur. Attention, les balises doivent être écrites en minuscules !

Diverses balises Xhtml et XSL...

La partie HTML ou XHTML du document.

```
</body>
</html>
```

Fin de la partie en HTML ou XHTML.

</xsl:template>

La fermeture de la balise de template.

</xsl:stylesheet>

Le document XSL se termine obligatoirement par la fermeture de la balise de déclaration de document XSL.

Attention ! Pour que ce fichier XSL soit d'une quelconque utilité, il ne faut pas oublier de faire référence à celui-ci dans le fichier XML.

Ainsi cette ligne doit être ajoutée dans le fichier XML :

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

Cette balise indique au navigateur qu'une feuille de style (*stylesheet*) de type XSL est associée au fichier XML et qu'il doit aller chercher le fichier à l'adresse indiquée par l'attribut href.

Voici notre premier document XSL :

```
<?xml version="1.0"?encoding="UTF-8">
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
Diverses balises HTML et XSL...
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

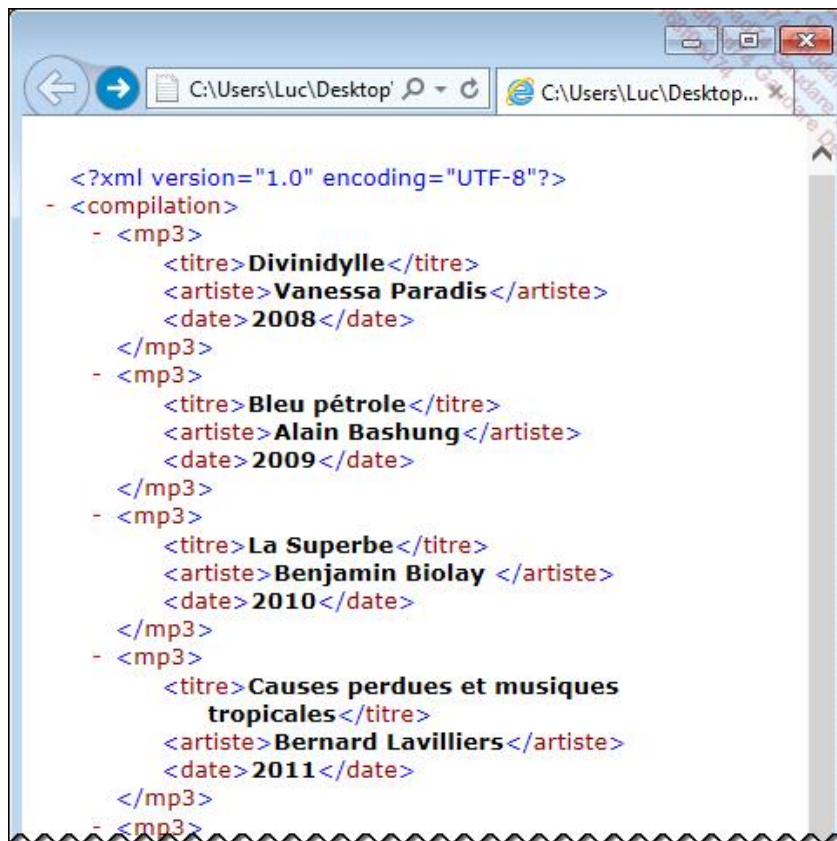
## Un premier exemple XSL

Après cet aperçu théorique, étudions un exemple détaillé : une compilation de compositions musicales MP3.

Voici un fichier XML que nous allons utiliser tout au long de ce chapitre :

```
<?xml version="1.0" encoding="UTF-8"?>
<compilation>
  <mp3>
    <titre>Divinidylle</titre>
    <artiste>Vanessa Paradis</artiste>
    <date>2008</date>
  </mp3>
  <mp3>
    <titre>Bleu pétrole</titre>
    <artiste>Alain Bashung</artiste>
    <date>2009</date>
  </mp3>
  <mp3>
    <titre>La Superbe</titre>
    <artiste>Benjamin Biolay </artiste>
    <date>2010</date>
  </mp3>
  <mp3>
    <titre>Causes perdues et musiques tropicales</titre>
    <artiste>Bernard Lavilliers</artiste>
    <date>2011</date>
  </mp3>
  <mp3>
    <titre>Suppléments de mensonge</titre>
    <artiste>Hubert-Félix Thiéfaine</artiste>
    <date>2012</date>
  </mp3>
  <mp3>
    <titre>La Place du fantôme</titre>
    <artiste>La Grande Sophie</artiste>
    <date>2013</date>
  </mp3>
  <mp3>
    <titre>Racine carrée</titre>
    <artiste>Stromae</artiste>
    <date>2014</date>
  </mp3>
</compilation>
```

On l'enregistre sous le nom xsldemo avec une extension .xml (soit xsldemo.xml), en voici un aperçu dans Internet Explorer :



Passons maintenant au fichier XSL.

Le but de l'exemple est de représenter la compilation sous forme d'un tableau à trois colonnes.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<tr>
<td><xsl:value-of select="compilation/mp3/date" /></td>
<td><xsl:value-of select="compilation/mp3/titre" /></td>
<td><xsl:value-of select="compilation/mp3/artiste" /></td>
</tr>
</table>
</body>
```

```

</html>
</xsl:template>
</xsl:stylesheet>

```

Après les balises de départ d'un fichier XSL, on élabore un tableau tout à fait classique en HTML. Dans la première cellule, une information de date est renseignée, soit la balise `<xsl:value-of/>` avec l'attribut `select="compilation/mp3/date"`, qui indique comme chemin d'accès dans le fichier XML la balise racine **compilation**, la balise **mp3** et la balise **date**. Dans la deuxième cellule, une information de titre est renseignée, soit la balise `<xsl:value-of/>` avec l'attribut `select="compilation/mp3/titre"`, qui indique comme chemin d'accès dans le fichier XML la balise racine **compilation**, la balise **mp3** et la balise **titre**. Enfin, la dernière cellule est consacrée à l'information relative à l'artiste, avec l'attribut `select="compilation/mp3/artiste"`.

Le fichier est enregistré sous le nom `xsl demo` avec une extension `.xsl` (`xsl demo.xsl`).

Afin d'associer ce fichier XSL au fichier XML précédent, la balise suivante est ajoutée :

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

Le code complet de ce dernier devient :

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl demo1.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...

```

Et une fois affiché dans un navigateur, notre fichier XML est devenu un tableau plus présentable.



Année	Album	Artiste
2008	Divinidylle	Vanessa Paradis

Par contre, une seule référence de la compilation est affichée dans le tableau. Pour afficher toutes les références, procéder comme suit :

Le fichier XML de départ reste inchangé.

Considérons le fichier XSL dans lequel la balise `<xsl:for-each />` (*for-each* signifie pour chaque), avec comme attribut `select="compilation/mp3"`, va être ajoutée. Dans le fichier XML de balises, une ligne de tableau (`<tr>`) est insérée, contenant des cellules `<td>` dont voici le contenu :

- balise **date** soit `<xsl:value-of select="date" />` pour la première cellule.
- balise **titre** soit `<xsl:value-of select="titre" />` pour la deuxième cellule.
- balise **artiste** soit `<xsl:value-of select="artiste" />` pour la troisième cellule.

Ce qui donne :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="compilation/mp3">
<tr>
<td><xsl:value-of select="date" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="artiste" /></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le fichier est enregistré avec l'extension .xsl (xsldemo1.xsl).

Il ne faut pas oublier de modifier le lien dans le fichier XML. Soit :

```
<?xml-stylesheet type="text/xsl" href="xsldemo1.xsl"?>
```

Le code complet du fichier XML devient :

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

```

<?xml-stylesheet type="text/xsl" href="xsldemo1.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...

```

Pour rappel, le fichier XML sans le fichier XSL prendrait la forme suivante :



Il ne reste plus qu'à vérifier que le fichier affiche bien toutes les références de la compilation.



Les victoires de la musique		
Année	Album	Artiste
2008	Divinidylle	Vanessa Paradis
2009	Bleu pétrole	Alain Bashung
2010	La Superbe	Benjamin Biolay
2011	Causes perdues et musiques tropicales	Bernard Lavilliers
2012	Suppléments de mensonge	Hubert-Félix Thiéfaine
2013	La Place du fantôme	La Grande Sophie
2014	Racine carrée	Stromae

## Trier avec le XSL

Le langage XSL permet de trier des données d'un fichier XML associé, en ordre croissant ou décroissant. Ainsi, il suffit d'ajouter la balise `<xsl:sort select="..." />` et l'attribut `order="ascending"` pour trier des données par ordre croissant, ou `order="descending"` pour trier par ordre décroissant.



Cette concision illustre bien la puissance du langage XSL.

### Exemple

Reprenons notre fichier XML de départ (inchangé).

Dans le fichier XSL, nous allons trier notre compilation de mp3 en XML par ordre alphabétique croissant du nom des artistes. En outre, nous allons permuter les colonnes "Année" et "Artiste" pour bien montrer que le XSL affiche les données du fichier XML selon le fichier HTML ou XHTML qu'il contient.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Artiste</td>
<td>Titre</td>
<td>Année</td>
</tr>
<xsl:for-each select="compilation/mp3">
<xsl:sort select="artiste" order="ascending"/>
<tr>
<td><xsl:value-of select="artiste" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="date" /></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

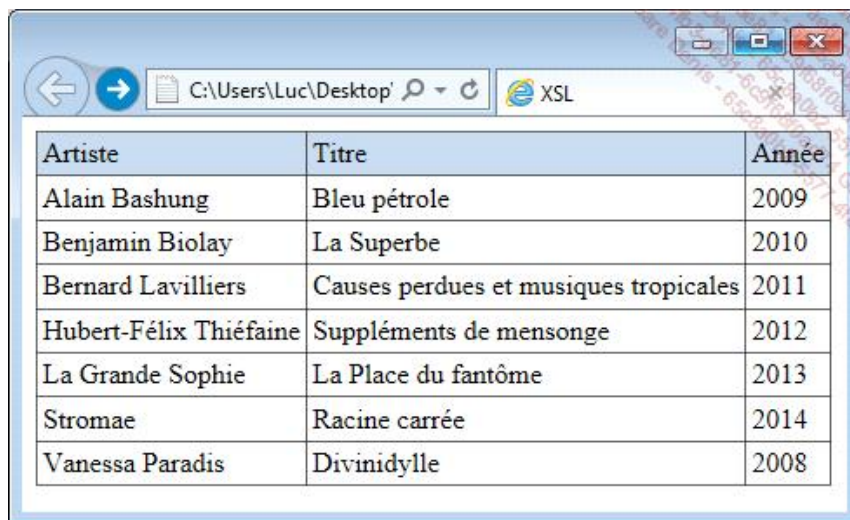
Le fichier est enregistré sous le nom `xsl_order` avec une extension `.xsl` (`xsl_sort.xsl`).

Afin d'associer ce fichier XSL au fichier XML, la balise suivante est ajoutée :

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

Soit :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_sort.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...
```



Artiste	Titre	Année
Alain Bashung	Bleu pétrole	2009
Benjamin Biolay	La Superbe	2010
Bernard Lavilliers	Causes perdues et musiques tropicales	2011
Hubert-Félix Thiéfaine	Suppléments de mensonge	2012
La Grande Sophie	La Place du fantôme	2013
Stromae	Racine carrée	2014
Vanessa Paradis	Divinidylle	2008

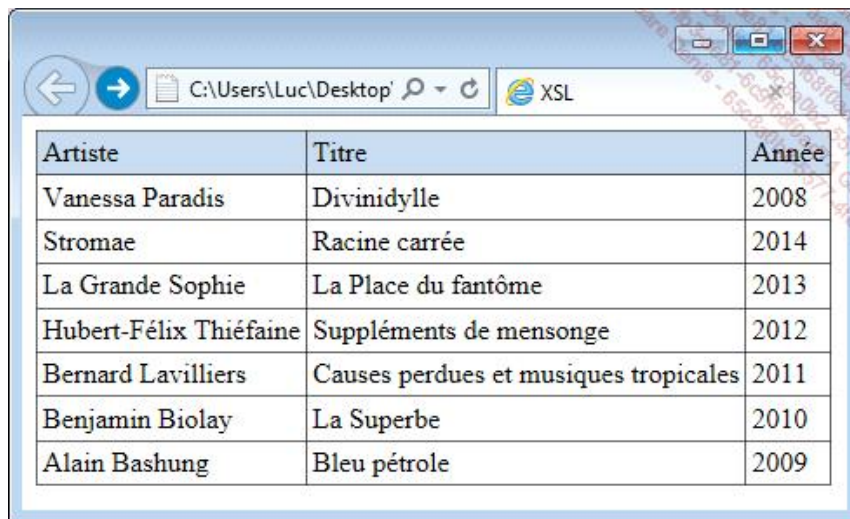
Pour trier la compilation par ordre alphabétique inverse, il suffit de modifier l'attribut `order="descending"`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Artiste</td>
<td>Titre</td>
<td>Année</td>
</tr>
<xsl:for-each select="compilation/mp3">
```

```

<xsl:sort select="artiste" order="descending"/>
<tr>
<td><xsl:value-of select="artiste"/></td>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="date"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```



Artiste	Titre	Année
Vanessa Paradis	Divinidylle	2008
Stromae	Racine carrée	2014
La Grande Sophie	La Place du fantôme	2013
Hubert-Félix Thiéfaine	Suppléments de mensonge	2012
Bernard Lavilliers	Causes perdues et musiques tropicales	2011
Benjamin Biolay	La Superbe	2010
Alain Bashung	Bleu pétrole	2009

Le fichier est enregistré sous le nom xsl\_orderbis avec une extension .xsl (xsl\_orderbis.xsl).

Afin d'associer ce fichier XSL au fichier XML, la balise suivante est ajoutée :

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

Soit :

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_sortbis.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...

```

---

## Filtrer avec le XSL

Le langage XSL permet aussi de filtrer les données du fichier XML associé selon des critères tels que égal, différent de, plus grand que, plus petit que.

Il suffit d'utiliser l'attribut :`select="chemin_d'accès[balise='xxx']"`

Les opérateurs possibles sont :

- = pour égal.
- != pour différent (non égal).
- > pour plus grand que.
- < pour plus petit que.

Ceci vous paraît un peu abstrait ? Voyons un exemple de filtre de données...

Dans la compilation mp3, ne reprenons que le (ou les) titre(s) de l'artiste Stromae.

L'attribut select du fichier XSL devient `select="compilation/mp3[artiste= 'Stromae']"`.

Le fichier XML reste toujours inchangé.

Passons maintenant au fichier XSL.

Nous allons reprendre dans notre compilation de mp3 en XML que le (ou les) titre(s) de Stromae.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="compilation/mp3[artiste='Stromae']">
<tr>
<td><xsl:value-of select="date"/></td>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
```

```
</xsl:template>
</xsl:stylesheet>
```

Le fichier est enregistré sous le nom xsl\_filter avec l'extension .xsl (xsl\_filter.xsl).

Afin d'associer ce fichier XSL au fichier XML, la balise suivante est ajoutée :

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

Soit :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_filter.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...
```

Ce qui nous permet d'afficher uniquement l'album de Stromae.



Année	Album	Artiste
2014	Racine carrée	Stromae

## Choisir avec le XSL

La balise `<xsl:if> ... </xsl:if>` permet d'effectuer un choix dans les données du fichier XML. L'attribut `match` est ajouté pour indiquer l'élément choisi. Ce qui donne :

```
<xsl:if test="balise='xxx'">
  balises Html
</xsl:if>
```

Nous allons illustrer ce choix par un exemple : ne faire apparaître que le(s) titre(s) d'Alain Bashung dans le fichier XML de compilation de mp3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="compilation/mp3">
<xsl:if test="artiste='Alain Bashung'">
<tr>
<td><xsl:value-of select="date"/></td>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le fichier est enregistré sous le nom `xsl_if` avec l'extension `.xsl` (`xsl_if.xsl`).

Afin d'associer ce fichier XSL au fichier XML de départ, la balise suivante est ajoutée :

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

Le fichier XML complet devient :



```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_if.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
EtC...

```

Ce qui nous permet d'afficher uniquement la ligne du tableau correspondant à Alain Bashung.



Année	Album	Artiste
2009	Bleu pétrole	Alain Bashung

Voici une autre illustration de cette possibilité de choix : ne retenir dans notre compilation de mp3 en XML que les informations postérieures à 2011.

Il suffit de modifier la balise `<xsl:if>` du fichier XSL précédent. La condition s'exprime par `test="date > 2011"` (> pour le signe >).

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr style="background: rgb(195,215,235)">
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="compilation/mp3">

```

```

<xsl:if test="date > 2011">
<tr>
<td><xsl:value-of select="date"/></td>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Le fichier est enregistré sous le nom xsl\_ifbis avec l'extension .xsl (xsl\_ifbis.xsl).

Afin d'associer ce fichier XSL au fichier XML de départ, la balise suivante est ajoutée :

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

Soit :

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_ifbis.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...

```

Ce qui nous permet d'afficher uniquement les lignes du tableau pour les années postérieures à 2011.



Année	Album	Artiste
2012	Suppléments de mensonge	Hubert-Félix Thiéfaine
2013	La Place du fantôme	La Grande Sophie
2014	Racine carrée	Stromae



## Conditions et XSL

Le XSL permet aussi de faire un choix conditionnel par la balise `<xml:choose>`. À l'intérieur de cette balise, une action peut être déterminée lorsqu'une condition est vérifiée (balise `<xsl:when>`). Dans le cas contraire, il faut prévoir une autre action (balise `<xsl:otherwise>`).

```
<xsl:choose>
```

Condition vérifiée

```
<xsl:when test="artiste='Bernard Lavilliers'">
<tr style="background: rgb(195,215,235)">
<td><xsl:value-of select="date" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="artiste" /></td>
</tr>
</xsl:when>
```

sinon

```
<xsl:otherwise>
<tr>
<td><xsl:value-of select="date" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="artiste" /></td>
</tr>
</xsl:otherwise>

</xsl:choose>
```

### Exemple

Nous allons reprendre, dans notre compilation de mp3 en XML, tous les titres de Bernard Lavilliers que nous allons afficher avec un arrière-plan de couleur, les autres titres étant affichés normalement.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<table border="1" style="border-collapse: collapse;">
<tr>
<td>Année</td>
<td>Album</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="compilation/mp3">
```

```

<xsl:choose>
<xsl:when test="artiste='Bernard Lavilliers'">
<tr style="background: rgb(195,215,235)">
<td><xsl:value-of select="date" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="artiste" /></td>
</tr>
</xsl:when>
<xsl:otherwise>
<tr>
<td><xsl:value-of select="date" /></td>
<td><xsl:value-of select="titre" /></td>
<td><xsl:value-of select="artiste" /></td>
</tr>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Le fichier est enregistré sous le nom xsl\_choose et l'extension .xsl (xsl\_choose.xsl).

Afin d'associer ce fichier XSL au fichier XML de départ, la balise suivante est ajoutée :

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

Soit :

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_choose.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...

```

Ce qui nous permet de mettre en évidence les lignes concernant Bernard Lavilliers.

Les victoires de la musique		
Année	Album	Artiste
2008	Divinidylle	Vanessa Paradis
2009	Bleu pétrole	Alain Bashung
2010	La Superbe	Benjamin Biolay
2011	Causes perdues et musiques tropicales	Bernard Lavilliers
2012	Suppléments de mensonge	Hubert-Félix Thiéfaine
2013	La Place du fantôme	La Grande Sophie
2014	Racine carrée	Stromae

## Transformer avec le XSL

L'élément `<xsl:apply-templates />`, utilisé au sein de la balise `<xsl:template/>`, permet d'appliquer un modèle (*template*) de transformation ou de présentation sur les éléments du document XML.

Nous allons appliquer à notre fichier XML des présentations de style différentes pour les informations relatives :

- au titre de l'album : `<xsl:template match="titre"> ... </xsl:template>`
- au nom de l'artiste : `<xsl:template match="artiste"> ... </xsl:template>`
- et à la date : `<xsl:template match="date"> ... </xsl:template>`

Les balises concernées ont été préalablement définies par :

```
<xsl:template match="mp3">
<p>
<xsl:apply-templates select="titre"/>
<xsl:apply-templates select="artiste"/>
<xsl:apply-templates select="date"/>
</p>
</xsl:template>
```

Le fichier XSL devient :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>XSL</title>
</head>
<body>
<h3>Les victoires de la musique</h3>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="compilation/mp3"
<p>
<xsl:apply-templates select="titre"/>
<xsl:apply-templates select="artiste"/>
<xsl:apply-templates select="date"/>
</p>
</xsl:template>
<xsl:template match="titre">
Titre : <span style="font-weight:bold;">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
<xsl:template match="artiste">
Artiste : <span style="font-variant:small-caps">
<xsl:value-of select="."/></span>
```

```
<br />
</xsl:template>
<xsl:template match="date">
Date: <span style="font-style:italic;">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
</xsl:stylesheet>
```

Le fichier est enregistré sous le nom `xsl_apply` et l'extension `.xsl` (`xsl_apply.xsl`).

Afin d'associer ce fichier XSL au fichier XML de départ, la balise suivante est ajoutée :

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

Ce dernier devient :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="xsl_apply.xsl"?>
<compilation>
<mp3>
<titre>Divinidylle</titre>
<artiste>Vanessa Paradis</artiste>
<date>2008</date>
</mp3>
<mp3>
<titre>Bleu pétrole</titre>
<artiste>Alain Bashung</artiste>
<date>2009</date>
</mp3>
Etc...
```

Ce qui permet d'afficher notre fichier avec une mise en forme différente pour le titre (en gras), le nom de l'artiste (petites majuscules) et le date (en italique).

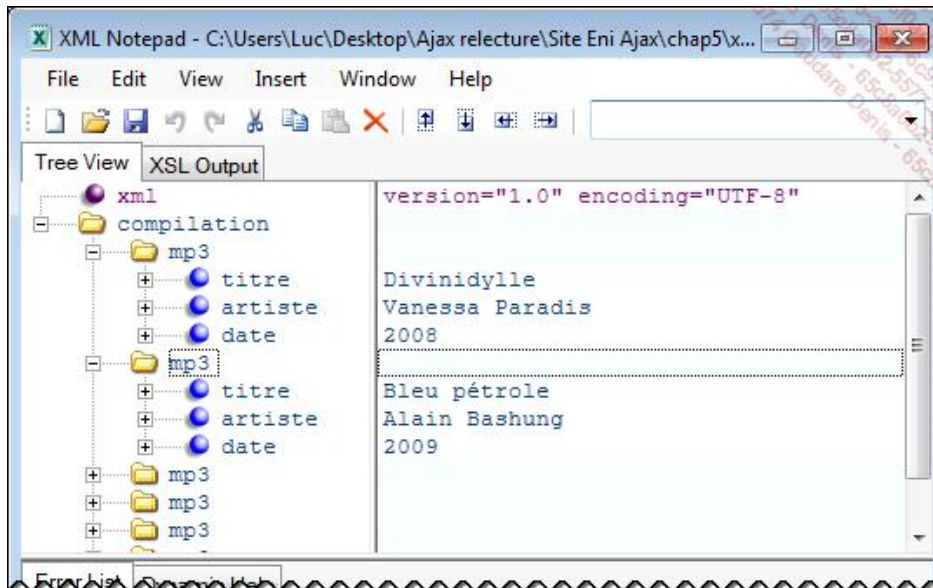




- Pour une étude (beaucoup) plus complète du XSL, vous pouvez vous reporter à l'ouvrage XML et XSL - Les feuilles de style XML de Cyril Vincent de la collection Ressources Informatiques des Éditions ENI.

## Le XSL avec XML Notepad 2007

Si vous utilisez XML Notepad 2007, lorsque vous ouvrez un fichier (**File - Open**), celui-ci s'affiche dans la fenêtre correspondant à l'onglet **Tree View**.



En choisissant l'onglet **XSL Output**, il est possible d'adjoindre au fichier XML un fichier XSL et de visualiser la transformation graphique instantanément, sans avoir à utiliser un navigateur.

