Le langage de Transformation XSL

- Formatage d'un document XML,
- Le langage XSLT:
  - Définition,
  - Principe de fonctionnement,
  - Structure d'une feuille de style,
  - Format d'une règle de transformation,
- Tri, Choix et Filtre avec XSLT

Formatage d'un document XML

- **CSS:**

  •CSS1 et CSS2 (*Cascading Style Sheets)* sont des feuilles de style dédiées à la présentation visuelle de documents XML.

- **XSL:**

  •Le langage XSL *eXtensible Stylesheet Language*), permet d'effectuer des transformations sur un document XML.

- **API:**

- Une autre façon de formater un document consiste à modifier son contenu par un programme (Java, C++, PERL,…), pour aboutir à un document HTML ou un format de sortie donnée.
- Les deux principales catégories d'API sont (**SAX** pour ***S**imple **A**PI **X**ML*
- et **DOM** pour ***D**ocument **O**bject **M**odel*)

Le langage XSLT

Le langage **XSL** ***(eXtensible Style Language)*** a été conçu pour transformer des documents XML en d'autres formats comme **PDF** ou des pages **HTML**.

Il est constitué de 2 langages:

- **XSLT** (pour ***XML Stylesheet Language Transformation***) est un langage de transformation de documents XML.

- **XSL-FO** (**pour** ***XML Stylesheet Language - Formatting Objets***) est un langage de mise en page de document.

Ces 2 langages sont des applications XML.

Définition

XSLT est un langage qui permet d'appliquer des règles de transformation sur un document XML donné en entrée, et de générer en sortie un document XML, HTML ou une autre forme.

Le processus de transformation d'un document XML en un document imprimable, au format PDF par exemple, est donc découpé en deux phases:

1 le document XML est transformé en un document XSL-FO à l'aide de feuilles de style XSLT.

2 Dans la seconde phase, le document FO obtenu à la première phase est converti par un processeur FO en un document imprimable.

Principe de fonctionnement

- Une transformation XSLT- appelée **une feuille de style** – consiste en une série de règles pour transformer un **arbre source** *(source tree)* XML en un **arbre résultant** *(result tree)*.

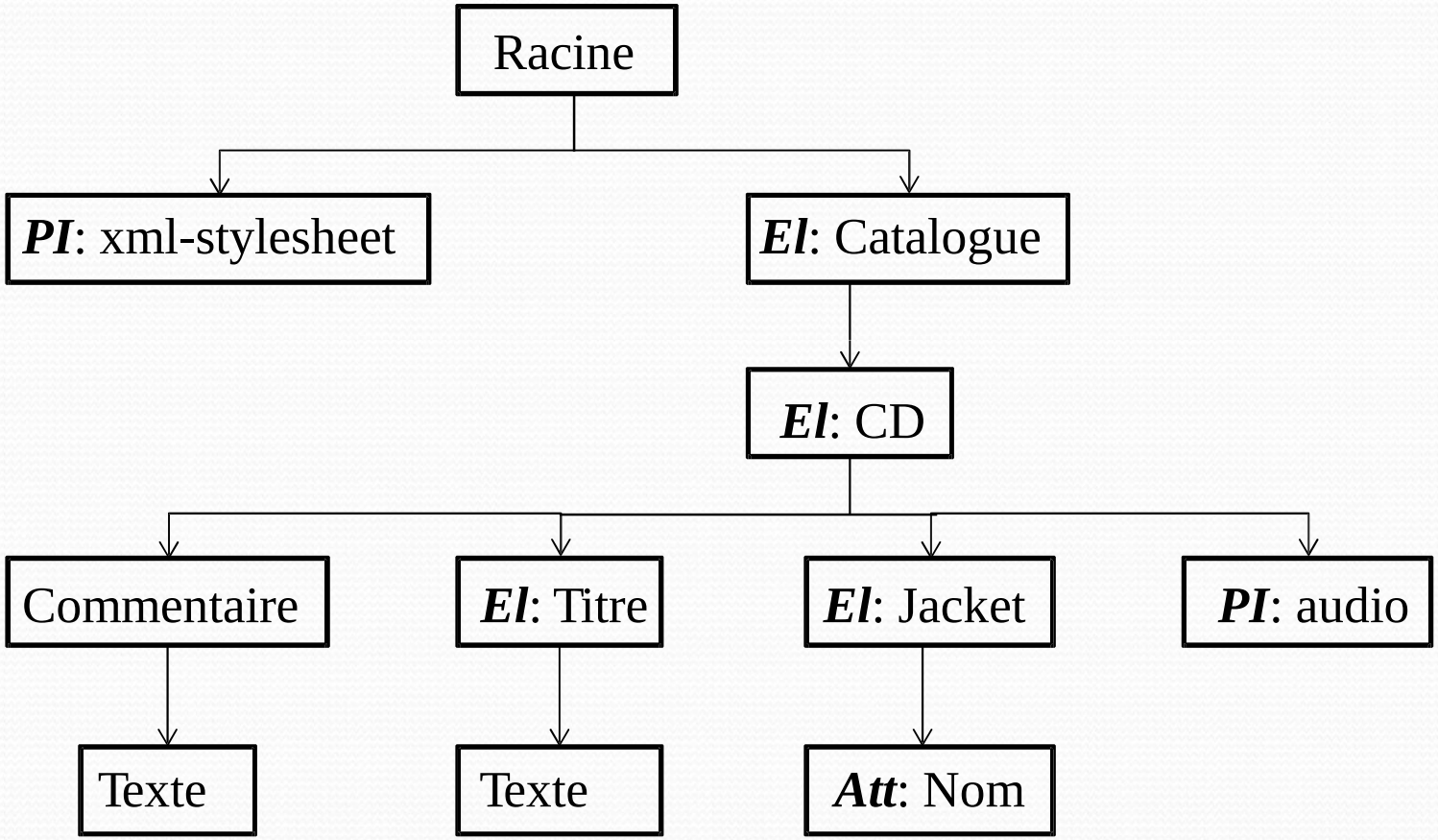
- La transformation se fait en associant des **motifs** *(patterns)* à des **modèles** *(templates)*et en les appliquant aux éléments de l'arbre source.

Exemple

Soit un document XML: catalogue.xml

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="transfo.xsl"?>
<Catalogue>
  <CD>
      <!-- Bon CD -->
      <Titre> Best of Bob Dylan<Titre>
      <Jacket NOM="dylan.gif"/ >
      <?audio dylan.mp3?>
  </CD>
</Catalogue>
```

Exemple: L'arbre associé



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:

Format d'une règle de transformation

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

Une **action** est une opération qui réalise la transformation et/ou spécifie les caractéristiques de la présentation

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

Exemple 1: document **demo.xml**

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
  href="demo.xsl"?>
<demoXML>
  <message>
     Voici du XML
  </message>
</demoXML>
```

Exemple1: le document xsl associé **demo.xsl**

```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:template match="/">
  <html>
    <body>
      <xsl:value-of select="demoXML/message"/>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Remarques:

- 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 **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

Exemple 2: enfants.xml (IE!)

Enfants.xml

```
<?xml version="1.0" encoding="ISO-
   8859-1"?>
<?xml-stylesheet href="enfants.xsl"
   type="text/xsl"?>
<enfants>
  <enfant>
     <nom>Ali</nom>
     <lien>garçon</lien>
     <date>07/11/83</date>
     <data>Une tête brulée!</data>
  </enfant>
  <enfant>
     <nom>Samia</nom>
     <lien>fille</lien>
     <date>20/12/85</date>
     <data>La petite fille chérie à
   son papa.</data>
  </enfant>
</enfants>
```

Enfants.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns:xsl="http://www.w3.org/TR/WD-
xsl">
 <body style="font-family:Arial;
   font- size:12pt;">
   <xsl:for-each select="enfants/enfant">
   <div style="background-
   color:teal; color:white;">
   <span style="font-weight:bold;
   color:white; padding:4px">
   <xsl:value-of select="nom"/>
   <xsl:value-of select="lien"/> </span>
   </div>
   <div style="margin-left:20px;
   font- size:10pt">
   <span> Anniversaire le <xsl:value-
   of select="date"/>
   </span>
   <span style="font-style:italic"> -
   <xsl:value-of select="data"/>
   </span>
   </div>
   </xsl:for-each>
 </body>
</html>
```

Le langage XSLT: exemple

Ali garçon

Anniversaire le 07/11/83 - *Une tête brulée!*

Samia fille

Anniversaire le 20/12/85 - *La petite fille chérie à son papa.*

Exemple 3: compilation.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="compilation.xsl"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

Exemple 3: films.xml

- <?xml version="1.0" encoding="ISO- 8859-1"?>
- **<?xml-stylesheet type="text/xsl" href="films.xsl"?>**
- <films>
- <film>
  - <titre>*Papa est en vayage d'affaires*</titre>
  - <realisateur>*Emir Kusturica*</realisateur>
- </film>
- <film>
  - <titre>*Matador*</titre>
  - <realisateur>*Pedro Almodovar*</realisateur>

<film>
<titre>Le tableau noir</titre>
<realisateur>Samira Makhmalbaf</realisateur>
</film>
<film>
<titre>Chat noir chat blanc</titre>
<realisateur>Emir Kusturica</realisateur>
</film>
<film>
<titre>Amores perros</titre>
<realisateur>Alejandro gonzales</realisateur>
</film>
</films>

Exemple 3: La feuille associée **films.xsl**

```
<?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 bgcolor="#FFFF00">
       <td>Titre</td>
       <td>Réalisateur</td>
      </tr>
      <tr>
       <td>
       <xsl:value-of select="films/film/titre"/>
      </td>
       <td>
       <xsl:value-of select="films/film/realisateur"/>
      </td>
      </tr>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le résultat

Titre	Réalisateur
Papa est en vayage d'affaires	Emir Kusturica

Sélection de tous les titres: filmsTous.xsl

Pour afficher tous les éléments, on ajoute la balise: **xsl:for-each *(pour chaque)*** avec comme attribut **select="films/film"**

```
<?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 bgcolor="#FFFF00">
             <td>Titre</td>
             <td>Réalisateur</td>
           </tr>
         <xsl:for-each select="films/film">
           <tr>
             <td><xsl:value-of select="titre"/></td>
             <td><xsl:value-of select="realisateur"/></td>
           </tr>
         </xsl:for-each>
         </table>
      </body>
   </html>
 </xsl:template>
</xsl:stylesheet>
```

Le résultat

Titre	Réalisateur
Papa est en vayage d'affaires	Emir Kusturica
Matador	Pedro Almodovar
Le tableau noir	Samira Makhmalbaf
Chat noir chat blanc	Emir Kusturica
Amores perros	Alejandro gonzales

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.**

Trier avec XSLT

1 il suffit d'ajouter l'attribut:

- **`order-by="+balise"`** pour trier en ordre croissant et
- **`order-by="-balise"`** pour trier en ordre décroissant.

pour l'espace de nom:

- `xmlns:xsl="http://www.w3.org/TR/WD-xsl`

2 Ou bien, l'attribut:

- **`order="ascending"`** pour trier en ordre croissant et
- **`order="descending"`** pour trier en ordre décroissant.

pour l'espace de nom:

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

Trier avec XSLT: filmsO1.xsl

Trier en ordre alphabétique croissant du nom des réalisateurs

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<!-- 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 bgcolor="#FFFF00">
  <td>Réalisateur</td>
  <td>Titre</td>
  </tr>
  <xsl:for-each select="films/film" order-by="+realisateur">
   <tr>
   <td><xsl:value-of select="realisateur"/> </td>
   <td><xsl:value-of select="titre"/></td>
   </tr>
  </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le résultat

Réalisateur	Titre
Alejandro gonzales	Amores perros
Emir Kusturica	Papa est en vayage d'affaires
Emir Kusturica	Chat noir chat blanc
Pedro Almodovar	Matador
Samira Makhmalbaf	Le tableau noir

Trier avec XSL: filmsO2.xsl

Trier en ordre alphabétique croissant du nom des artistes.

```
<?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 bgcolor="#FFFF00">
  <td>Réalisateur</td>
  <td>Titre</td>
  </tr>
```

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

Le résultat

Réalisateur	Titre
Samira Makhmalbaf	Le tableau noir
Pedro Almodovar	Matador
Emir Kusturica	Chat noir chat blanc
Emir Kusturica	Papa est en vayage d'affaires
Alejandro gonzales	Amores perros

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`.

Filtrer avec XSLT: compilationFiltre.xsl

La sélection des titres de l'artiste Alain Souchon.

L'attribut `select` devient:

```
select="films/film[realisateur='Emir Kusturica']"
```

```
<?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 bgcolor="#FFFF00">
         <td>Titre</td>
         <td>Réalisateur</td>
         </tr>
         <xsl:for-each select="films/film[realisateur='Emir Kusturica']">
            <tr>
            <td><xsl:value-of select="titre"/></td>
            <td><xsl:value-of select="realisateur"/></td>
            </tr>
         </xsl:for-each>
         </table>
   </body>
   </html>
</xsl:template>
</xsl:stylesheet>
```

Filtrer avec XSLT:

Titre	Réalisateur
Papa est en vayage d'affaires	Emir Kusturica
Chat noir chat blanc	Emir Kusturica

Le Choix avec XSLT

La balise `<xsl:if> ... </xsl:if>` permet d'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']">
    balises Html
</xsl:if>
```

Le choix avec XSLT: filmsChoix.xsl

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<!--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 bgcolor="#FFFF00">
            <td>Titre</td>
            <td>Réalisateur</td>
         </tr>
         <xsl:for-each select="films/film">
           <xsl:if match=".[realisateur='Samira Makhmalbaf']">
             <tr>
                  <td><xsl:value-of select="titre"/></td>
                  <td><xsl:value-of select="realisateur"/></td>
             </tr>
           </xsl:if>
         </xsl:for-each>
      </table>
     </body>
   </html>
 </xsl:template>
</xsl:stylesheet>
```

Le choix avec XSLT: résultat

Titre	Réalisateur
Le tableau noir	Samira Makhmalbaf

Le choix conditionnel: compilationCond.xsl

Le choix conditionnel se fait avec:

```
<xsl:choose>.
   <xsl:when une  action xsl:otherwise une autre action
</xsl:choose>

<xsl:choose>
   <xsl:when test=".[artiste='Emir Kusturica']">
     <tr bgcolor="#00FF00">
        <td><xsl:value-of select="titre"/></td>
        <td><xsl:value-of select="realisateur"/></td>
     </tr>
   </xsl:when>
   <xsl:otherwise>
    <tr>
        <td><xsl:value-of select="titre"/></td>
        <td><xsl:value-of select="realisateur"/></td>
    </tr>
   </xsl:otherwise>
</xsl:choose>
```

XSLT: Choix conditionnel

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
  <table border="1" cellspacing="0" cellpadding="3">
  <tr bgcolor="#FFFF00">
  <td>Réalisateur</td>
  <td>Titre</td>
  </tr>
  <xsl:for-each select="films/film">
  <xsl:choose>
  <xsl:when test=".[realisateur='Emir Kusturica']">
  <tr bgcolor="#00FF00">
  <td><xsl:value-of select="titre"/></td>
  <td><xsl:value-of select="realisateur"/></td>
  </tr>
  </xsl:when>
  <xsl:otherwise>
  <tr>
  <td><xsl:value-of select="titre"/></td>
  <td><xsl:value-of select="realisateur"/></td>
  </tr>
  </xsl:otherwise>
  </xsl:choose>
  </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le choix conditionnel: résultat

Tous les titres d'Alain Souchon seront affichés dans une colonne verte, les autres seront affichés normalement.

Réalisateur	Titre
Papa est en vayage d'affaires	Emir Kusturica
Matador	Pedro Almodovar
Le tableau noir	Samira Makhmalbaf
Chat noir chat blanc	Emir Kusturica
Amores perros	Alejandro gonzales

Plusieurs choix sur la sélection

```
<xsl:choose>
  <xsl:when test="quelque-chose">
     [action]
  </xsl:when>
  <xsl:when test="autre-chose">
     [action]
  </xsl:when>
     ...
  <xsl:otherwise>
     [action]
  </xsl:otherwise>
</xsl:choose>
```