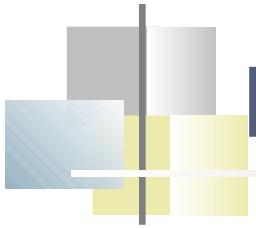


# Programme

- Introduction
  - Documents structurés, XML, Bases de données
- Documents XML : description et manipulation
  - Description de documents
  - Modélisation – *XDM*
  - Localisation de composants XML -- *XPath*
  - **Transformation de documents XML – *XSLT***
- Plus tard…
  - Typage -- *XSchema*
  - Interrogation de documents XML -- *XQuery*

# Transformer des données XML XSLT

- Objectifs
- Description d'une feuille de style
  - Règles
  - Processus de transformation
  - Exemples

- XSLT : fonctions étendues

- Version 1.0 W3C Recommendation 16 November 1999
- Version 2.0 Recommendation 23 January 2007

# Bibliographie

XSLT : XSL Transformation
eXtensible Stylesheet Langage Transformations
http://www.w3.org/TR/xslt

Comprendre XSLT
Bernd Amann et Philippe Rigaux, paru aux Editions O'Reilly

Différents tutoriaux (dont ceux d’Emmanuel Bruno, de Jacques Le Maitre)

# Un exemple : publication de données

- Extraire un contenu d'un document XML pour le mettre dans un format reconnu par une application de publication particulière
- Trois versions différentes du document *guide*
  - – un site web, affichant les informations sur le guide, ses itinéraires et ses cotations
  - – un site WAP permettant de consulter sur un téléphone mobile les mêmes informations
  - – un document PDF donnant l'ébauche de ce qui pourrait être un « Officiel des ballades» imprimé

# Solution : Transformer le document

- Extraire des fragments d'un document et les assembler différemment dans une structure nouvelle
  - À l'aide d'un langage de requêtes (cf. XQuery)
  - À l'aide de feuilles de style
    - associer à certain type d’élément des directives de transformation
    - pour *par exemple*
      - Conserver tel quel le résultat de cette transformation
      - Indiquer des directives de présentation de leur contenu

# Exemple (XML→ HTML)

```
<?xml version="1.0"?>
<exemple>
  <titre>ceci est un titre</titre>
  <auteur>ceci est un auteur</auteur>
</exemple>
```

```
<html>
  <head>
        <title> transformer le document exemple </titre>
  </head>
  <body>
    <h1>ceci est un titre</h1>>
    <h1>ceci est un auteur</h1>
 </body>
</html>
```

# Transformer un document : XSLT

- Le langage XSLT permet de construire un nouveau document XML à partir d'un document XML existant en le transformant

- La transformation s'exprime dans une feuille de style
  - Une feuille de style est **un document XML** qui contient un ensemble de règles (*template*)
  - Chaque règle décrit une transformation à appliquer à certains composants du document de départ

- La feuille de style s’applique à un document XML
- XSLT opère sur l'arbre (ordonné) du document source
- Elle produit du XML ou du texte

# Une feuille de style très simple

- Feuille de style vide
  - Attribut `version` obligatoire
  - `xsl` est le préfixe usuellement utilisé  et qui correspond à l'espace de noms du langage XSLT
    - L'espace de noms permet au processeur XSLT de reconnaitre les « instructions » du langage XSLT

```
<xsl:stylesheet version='1.0'
      xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
     <!- ICI mettre un ensemble de règles (templates) -->
</xsl:stylesheet>
```

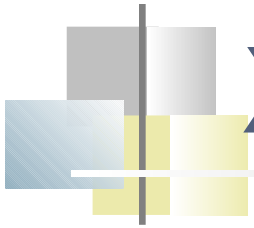
```
<exemple>
  <titre>ceci est un titre</titre>
  <auteur>ceci est un auteur</auteur>
</exemple>
```

→

```
<html>
  <head>
        <title> transformer le document exemple </titre>
  </head>
  <body>
     <h1>ceci est un titre</h1>>
     <h1>ceci est un auteur</h1>
  </body>
</html>
```

# Exemple

```
<xsl:stylesheet version = '1.0'
                xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:template match="/">
    <html>
       <head> <title> transformer le document exemple</title> </head>
       <body> <xsl:apply-templates/> </body>
    </html>
  </xsl:template>
  <xsl:template match='auteur'>
    <h1> <xsl:value-of select="."/> </h1>
  </xsl:template>
  <xsl:template match='titre'>
     <h1> <xsl:value-of select="."/> </h1>
  </xsl:template>
</xsl:stylesheet>
```



# XSLT

- Une feuille de style contient une suite de règles (`xsl:template`)
- L'application de la feuille de style consiste à
  - Parcourir  récursivement le *document source en partant de* la racine
  - pour chaque nœud rencontré dans ce parcours, chercher la règle à appliquer
    - l'application d'une règle produit un fragment du document résultat
    - La règle indique si le parcours récursif du document source continue

```
<xsl:template match='pattern'>
        transformation
</xsl:template>
```

# Règles : deux parties

- *Une partie déclaration*
  - spécification des nœuds sur lesquels la règle doit s'appliquer
- *Une partie exécution*
  - spécification de la transformation (*modèle - template*)
    - définition de la forme du résultat à générer sur le flot de sortie quand la règle sera activée

```
<xsl:template match='pattern'>
        transformation
</xsl:template>
```

# Règles : deux parties

- *Une partie déclaration*
  - spécification des nœuds sur lesquels la règle doit s'appliquer
  - à l'aide d'une expression XPath (*motif - pattern*)
    - les nœuds cibles sont atteints en suivant un chemin conforme à l'expression XPath donné à partir d'un nœud contexte
- *Une partie exécution*
  - spécification de la transformation (*modèle - template*)
    - définition de la forme du résultat à générer sur le flot de sortie quand la règle sera activée
  - à l'aide d'une suite d'instructions
    - qui s'appliquent aux nœuds filtrés en respectant

# Forme d'une règle

```
<xsl:template match='pattern'>
        transformation
</xsl:template>
```

- Le chemin de localisation permet d'atteindre les nœuds cibles de la transformation
- La transformation – corps de la règle est décrite avec un mélange de règles et de syntaxe XML à générer sur le flot de sortie (séquence de caractères et éléments XML, certains peuvent être des instructions XSLT)

- Une règle est évalué dans le contexte d'un nœud donné
- L'ordre des règles n'a pas d'importance

# Extraction de données

- On insère dans le résultat des fragments du document traité

  `<xsl:value-of select='pattern'/>`

- On donne un *chemin* d'accès à un nœud à partir du nœud courant

```
<xsl:template match='auteur'>
     <h1> <xsl:value-of select="."/> </h1>
 </xsl:template>
```

# Exemple

```
<?xml version="1.0"?>
…
<table>
    <description>personnel 4ème étage</description>
    <personne><nom>Bond</nom><bureau>U1</bureau></personne>
    <personne><nom>Lupin</nom><bureau>U2</bureau></personne >
    <personne><nom>Templar</nom><bureau>U3</bureau></personne>
</table>
…
```

```
…
<xsl:template match="personne">
 <LI>
   <xsl:value-of select="nom"/> - <xsl:value-of select="bureau"/>
 </LI>
</xsl:template>
…
```

```
<LI>Bond - U1</LI>
<LI>Lupin - U2</LI>
<LI>Templar - U3</LI>
```

# Les patterns

- On ne peut pas associer n'importe quelle expression XPath à l'attribut `match`
  - Certaines expressions seraient trop complexes à évaluer
  - L'expression doit toujours désigner une séquence de nœuds

```
<xsl:template match="1">
```
NON

```
<xsl:templatematch="preceding::node()[5]">
```
NON

# Axes possibles

- `child` → les nœuds enfants d'un élément
- `Attribute` → les attributs d'un élément
- `//` → les descendants d'un élément ou lui-même (`descendant-or-self::node()/`)

# Exemple - instruction *xsl:apply-templates*

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

<xsl:template match="/">

  <html>

    <head> <title> transformation du document exemple</title> </head>

    <body> <xsl:apply-templates/> </body>

  </html>

</xsl:template>

<xsl:template match='auteur'>
        <h1> <xsl:value-of select="."/> </h1>

</xsl:template>

<xsl:template match='titre'>
        <h1> <xsl:value-of select="."/> </h1>

</xsl:template>

</xsl:stylesheet>
```

Applique l'ensemble des règles à tous les fils du nœud contexte

# L'instruction *xsl:apply-templates*

- Elle a la forme suivante

```
<xsl:apply-templates  select='expression'/>
```

- `expression` est un chemin XPath qui désigne les nœuds auxquels la règle doit s'appliquer
- Applique l'ensemble des règles aux nœuds atteints par le chemin XPath

- `apply-templates` permet d'indiquer l'ordre de parcours du document source

# L'instruction *xsl:apply-templates*

- Elle peut aussi avoir la forme suivante

```
<xsl:apply-templates/>
```

- la valeur par défaut de `select` est `child::node()`
  - Application à tous les fils du nœud contexte si l'attribut `select` n'apparaît pas

# Processus de transformation

- Il consiste à instancier des règles
  - le corps de la règle est inséré dans le document résultat
  - les instructions XSLT contenues dans le corps de la règle sont exécutées à leur tour
  - le résultat d’une instruction vient remplacer l’instruction dans le résultat

# Autrement dit…

- Le processeur XSLT
  - analyse l’arbre du document
  - opère sur une liste de nœuds
    - Réduite au nœud racine au départ
  - cherche si une règle vérifie le nœud contexte
    - Si oui, application des instructions contenues dans cette règle
      - Instanciation de la transformation sur le flot de sortie
      - S'il y en a une seule règle, elle est appliquée
    - Si non, application des règles par défaut
      - L’arbre du document source est parcouru à partir de la racine
      - Les nœuds textes sont recopiés dans l’arbre résultat

# Règles prédéfinies – par défaut

- Elles spécifient une transformation par défaut si aucune des règles de la feuille de style n'est applicable
  - Pour les nœuds racine et élément

```
<xsl:template match="*|/">
    <xsl:apply-templates/>
</xsl:template>
```

Si le noeud contexte est le nœud document ou un nœud element, continuer sur les fils de ce noeud

  - Pour les nœuds texte et attribut

```
<xsl:template match="text()|@*">
  <xsl:value-of select="." />
</xsl:template>
```

Recopie leur valeur (le texte ou la valeur de l'attribut) sur le flot de sortie dans l'arbre résultat

# Processus de transformation

- Si plusieurs règles sont applicables ?

- La priorité peut être spécifiée explicitement
  - attribut `priority`
  - un nombre d'autant plus grand que la règle a une priorité forte
- Sinon la plus spécifique est choisie

# Priorité entre règles : exemple

```
<xsl:template match="Personne">
  <regle><xsl:apply-templates/></regle>
</xsl:template>

<xsl:template match="Personne[Bureau]">
   <Autreregle><xsl:apply-templates/></Autreregle>
</xsl:template>
```

```
<?xml version="1.0"?> …
<table><description>personnel 4ème étage</description>
   <personne><nom>Bond</nom><bureau>U1</bureau></personne>
   <personne><nom>Lupin</nom></personne>
   <personne><nom>Templar</nom><bureau>U3</bureau></personne>
</table>…
```

Pour `<personne>…<personne>` → la première règle s'applique

Pour `<personne>…<bureau>…</bureau><personne>` → Seule la seconde s'applique

# Priorité entre règles : autre exemple

```
<xsl:template select="Para">
  <P><xsl:apply-templates</P>
</xsl:template>


<xsl:template
   select='Para[@type="citation"]'>
     <BLOCKQUOTE>
        <xsl:apply-templates/>
     </BLOCKQUOTE>
</xsl:template>
```

Pour `<Para>...</Para>` → La première règle s'applique

Pour `<Para type="citation">...</Para>` → Seule la seconde s'applique

# Un exemple complet

# Document XML

```
<?xml version="1.0"?>
<bib>
   <livre>
        <titre>t1</titre>
        <auteur>a1</auteur>
        <auteur>a2</auteur>
    </livre>
    <article>
        <titre>t2</titre>
        <auteur>a3</auteur>
       <auteur>a4</auteur>
    </article>
    <livre>
             <titre>t3</titre>
             <auteur>a5</auteur>
             <auteur>a6</auteur>
             <auteur>a7</auteur>
    </livre>
</bib>
```

# … de XML vers HTML …

```
<HTML>
    <HEAD>
        <TITLE> Entrées bibliographiques</TITLE>
    </HEAD>
    <BODY>
    <TABLE>
        <TBODY>
            <TR><TD>t1</TD><TD>a1</TD><TD>a2</TD></TR>
            <TR><TD>t2</TD><TD>a3</TD><TD>a4</TD></TR>
            <TR>
              <TD>t3</TD><TD>a5</TD><TD>a6</TD><TD>a7</TD>
            </TR>
        </TBODY>
    </TABLE>
    </BODY>
</HTML>
```

# Feuille XSL : style.xsl (1)

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

  <xsl:template match="/">
    <HTML>
       <HEAD><TITLE> Entrées bibliographiques</TITLE>
    </HEAD>
    <BODY>
       <xsl-apply-templates/>
    </BODY>
  </xsl:template>
…
</xsl:transform>
```

# Feuille XSL : style.xsl (2)

```
<xsl:template match="titre">
    <TD><xsl:value-of select="."/></TD>
</xsl:template>

<xsl:template match="auteur">
    <TD><xsl:value-of select="."/></TD>
</xsl:template>

<xsl:template match="livre | article">
    <TR><xsl:apply-templates select="titre"/>
        <xsl:apply-templates select="auteur"/>
    </TR>
</xsl:template>

<xsl:template match="bib">
    <TABLE><TBODY><xsl:apply-templates/></TBODY></TABLE>
</xsl:template>

+ règles par défaut (ou les neutraliser)
```

# Feuille de style : style.xsl (3)

```
<?xml-stylesheet href='style.xsl' type='text/xsl'?>
```

- À insérer après le prologue dans le document à présenter
- Si l'application sait interpréter cette instruction, la transformation et présentation seront effectives

# Résumé

- Déclaration d'un ensemble de *règles* s'appliquant aux noeuds du document source (instruction *xsl:template)*
- Chaque règle produit un *fragment du* document résultat
- Parcours d'un document – arbre du document - avec des instructions *xsl:apply-templates*
- Le processeur choisit la règle s'appliquant à chaque nœud sélectionné

# Instructions

`<xsl:apply-templates/>`

- La liste des nœuds à traiter est constituée des nœuds fils du nœud contexte
- Appliquer les règles

`<xsl:apply-templates select="`*`pattern`*`"/>`

- La liste des nœuds à traiter est constituée des nœuds atteints par le chemin *pattern* depuis le nœud contexte
- Appliquer les règles

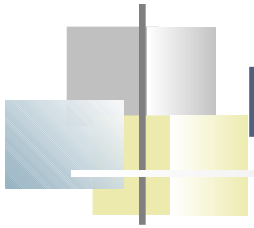
`<xsl:value-of select="`*`pattern`*`"/>`

- Recopier dans le flot de sortie la valeur-chaîne de chaque nœud atteint par le

  le nœud contexte

`<xsl:copy-of select="`*`pattern`*`"/>`

- Recopier dans le flot de sortie le fragment du document dont la racine est le nœud atteint par le chemin *pattern depuis le nœud*



# Bilan

- Langage de règles
  - Transformer pour …
  - Intégration de XPath
  - Instructions …
    - Ce n’est pas un langage de requêtes
    - Une feuille de style pour *un document (ou une classe de document)*