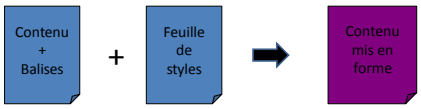
XML Stylesheet Language Transformation

INTRODUCTION À XSLT

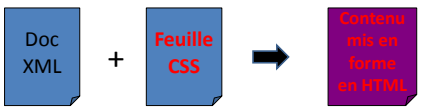
Séparation du fond et de la forme

- Le fond : document XML : contenu + balisage
- La forme de présentation : Feuille de style



Séparation du fond et de la forme

- Le fond : document XML : contenu + balisage
- La forme de présentation : Feuille de style CSS



CSS et XML

- Pas de display par défaut associés aux éléments d'un document XML
- Déclaration du display des éléments XML *inline* ou *block*
- Association d'une feuille de style CSS à un document XML:
  <?xml-stylesheet type="text/css" href="mystyle.css"?>

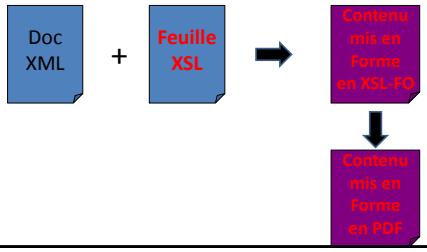
http://www.w3schools.com/xml/xml_display.asp

Séparation du fond et de la forme

- Le fond : document XML : contenu + balisage
- La forme de présentation : Feuille de style XSL

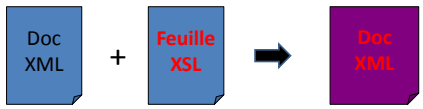
Séparation du fond et de la forme

- Le fond : document XML : contenu + balisage
- La forme de présentation : Feuille de style XSL



Restructuration du fond

- Le fond: document XML : contenu + balisage
- Sélection de contenu et restructuration : Feuille de style XSL (forme ignorée)



Transformations XSL

- Conserver un contenu dans un format et l'afficher dans un autre
  en particulier, engendrer du (X)HTML
- Restructurer un contenu: convertir le format du contenu dans un autre format
- Extraire un sous-ensemble utile du contenu
- Combiner des contenus de provenances différentes

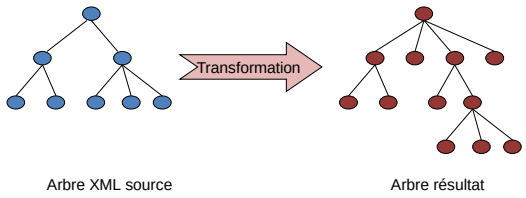
La famille XSL

- XSL est une famille de recommandations pour définir des transformations et des présentations de documents XML
- XSL Transformations: XSLT
  Un langage pour transformer des documents XML en d'autres documents XML
- XML Path Langage: XPath
  Un langage utilisé par XSLT pour accéder ou faire référence aux parties d'un document XML
- XSL Formatting Objects: XSL-FO
  Un vocabulaire XML pour spécifier des sémantiques de formatage

XSLT in a nutshell

- Langage à base de règles
  - Une règle s'applique à un type d'élément XML
  - Une règle décrit la sortie produite en fonction des données d'entrée
- Transformation d'un arbre XML par récursion
  - Application d'une règle à la racine de l'arbre
  - Transformation des sous-arbres de l'arbre

XSLT in a nutshell



XSLT in a nutshell

Règles dites « modèles » :
- élément xsl:template
- décrivant la structure, le modèle de l'arbre résultat

Sélection du nœud auquel appliquer le template

```
<xsl:template match="/">
  <html>
    <head><title>Ma première règle modèle</title></head>
    <body>
      <h1>Bonjour!</h1>
    </body>
  </html>
</xsl:template>
```

Template

Feuille de style XSLT

Document XML

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:template match="/">
        <html>
          <head><title>Ma première règle modèle</title></head>
          <body><h1>Bonjour!</h1></body>
        </html>
     </xsl:template>
</xsl:stylesheet>
```

Application d'une feuille de style XSLT

- Instruction de traitement :
  <?xml-stylesheet type="text/xml" href="mytransfo.xsl"?>
- http://www.w3.org/TR/xml-stylesheet/
  Associating Style Sheets with XML documents Version 1.0

Application récursive des règles XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/..." version="1.0">
     <xsl:template match="/">
        <html>
          <head><title>Ma première règle modèle</title></head>
          <body>
            <h1>Bonjour!</h1>
            <xsl:apply-templates/>
          </body>
        </html>
     </xsl:template>
     …
</xsl:stylesheet>
```

Application récursive des règles XSLT

- Par défaut, traitement jusqu'aux feuilles
- Stopper la récursion :
  <xsl:template match="confidentiel"/>

Application récursive des règles XSLT

Règles implicites selon le type de nœud

- 7 types de nœuds
  - Elément
  - Attribut
  - Texte
  - Commentaire
  - Instruction de traitement
  - Racine
    - Contient tous les autres nœuds du document
    - Nœud abstrait, père de l'élément document
  - Espace de noms
    - Une déclaration d'espace de nom n'est pas traitée comme les autres attributs

Application récursive des règles XSLT

Règles implicites

- Le traitement commence à la racine de l'arbre source et celui-ci est parcouru récursivement entièrement:
  <xsl:template match="/"> <xsl:apply-templates/> </xsl:template>
  <xsl:template match="*"> <xsl:apply-templates/> </xsl:template>
- La valeur de chaque attribut est incluse dans l'arbre résultat si le noeud a été atteint:
  <xsl:template match="@*"> <xsl:value-of select="."/> </xsl:template>
- Un nœud texte est inclus dans l'arbre résultat:
  <xsl:template match="text()"> <xsl:value-of select="."/> </xsl:template>

Application récursive des règles XSLT

Règles implicites (suite)

- Les instructions de traitement sont omises dans l'arbre résultat:
  <xsl:template match="processing-instruction()"/>
- Les commentaires sont omis dans l'arbre résultat:
  <xsl:template match="comment()"/>

Sélection d'une règle

- Le critère de sélection d'une règle est basé sur les motifs de correspondance
  - que sont les chemins de localisation **XPath**
  - qui apparaissent en valeur des attributs **match** des éléments xsl:template
  - Seuls les axes descendants et autoréférentiels sont autorisés
  - Les chemins sont évalués de la droite vers la gauche :

    match="chapitre/section/para"
    Le nœud contextuel est-il de type para?
    Si oui, son père est-il de type section?
    Si oui, son grand-père est-il de type chapitre?

Sélection d'une règle

Résolution de conflits entre règles

Une règle plus spécifique est prioritaire
sur une règle plus générale

Règles de précédence qui s'appliquent aux motifs susceptibles de s'appliquer à un même nœud:

- Chaque alternative dans un motif a une importance égale
- Un motif plus spécifique qu'un autre a une importance plus grande:

truc/machin	plus spécifique que	machin
truc/machin	plus spécifique que	truc/*
truc[@machin="bidule"]	plus spécifique que	truc

Exemple : feuille de style

```
<?xml version="1.0">
<xsl:stylesheet  xmlns:xsl="http://www..." version="1.0">
    <xsl:template  match="liste-citations">
      <html>
        <body><h1>Citations</h1><xsl:apply-templates/></body>
      </html>
    </xsl:template>
    <xsl:template  match="aphorisme">
      <blockquote>
        <xsl:apply-templates/>
      </blockquote>
    </xsl:template>
    <xsl:template  match="texte">
      <p><xsl:apply-templates/></p>
    </xsl:template>
</xsl:stylesheet>
```

Exemple : arbre source

```
<?xml version= ="1.0">
<liste-citations>
    <categorie type="humour">
        <citation type="humour" id="1">
          <texte>
              La tentative est la première étape avant l'échec.
          </texte>
          <source type="série-tv"/>
        </citation>
        <aphorisme type="humour" id="2">
          <texte>
              Le travail paie à long terme. La paresse paie à court terme.
          </texte>
        </aphorisme>
        …
    </categorie>
    …
</liste-citations>
```

Exemple : arbre résultat

```
<html>
    <body>
        <h1>
          Citations
        </h1>
        <blockquote>
          <p>
            Le travail paie à long terme. La paresse paie à court terme.
          </p>
        </blockquote>
    </body>
</html>
```

L'élément xsl:value-of

1. Extraction de la valeur des nœuds :
<xsl:value-of select="."/> chemin XPath

- Valeur selon le type de nœud
  - Nœud Racine
    Hérite de la valeur de l'élément document
  - Nœud élément
    Toutes les données textuelles de l'élément ainsi que les valeurs de tous ses descendants
  - Nœud attribut
    La valeur de l'attribut
  - Nœud texte
    Tout le texte du nœud

L'élément xsl:value-of

1. Extraction de la valeur des nœuds :
<xsl:value-of select="."/> chemin XPath

- Valeur selon le type de nœud (suite)
  - Nœud instruction de traitement
    Tout le texte à l'intérieur des délimiteurs sauf le nom de l'instruction
  - Nœud commentaire
    Tout le texte à l'intérieur des délimiteurs
  - Nœud namespace
    L'URI de l'espace de noms

L'élément xsl:value-of

2. Calculs

```
<xsl:template match="liste-citation">
    <p>Total : <xsl:value-of select="count(//citation)"/> citations</p>
</xsl:template>
```

Création d'un nœud « ad hoc »

```
<xsl:template match='lien'>
    <a href='#{id}'>
        <xsl:value-of select='title' />
    </a>
</xsl:template>
```

La valeur de l'attribut et le contenu de l'élément sont calculés à la volée

Une expression entre accolades doit être évaluée

```
<lien>
  <id>abc</id>
  <title>Voir plus loin</title>
</lien>
```

→ `<a href=#abc>Voir plus loin</a>`

Création d'un nœud élément

```
<xsl:template match="chose">
    <xsl:element name="{@type}">
        <xsl:value-of select="."/>
    </xsl:element>
</xsl:template>
```

Le nom et le contenu de l'élément sont calculés à la volée

Une expression entre accolades doit être évaluée

`<chose type="cercle">r = 3</chose>` → `<cercle>r = 3</cercle>`

Création d'un nœud attribut

```
<xsl:template match="a">
    <p> voir ma
      <a>
        <xsl:attribute name="href">
          http://www.essi.fr/~<xsl:call-template name="enseignant"/>
        </xsl:attribute>
        page personnelle
      </a>
    </p>
</xsl:template>
```

- La valeur de l'attribut href est une URI dont le texte est en partie calculé à la volée (par un élément xsl:call-template)

Création d'un nœud attribut

- Affecter un même ensemble d'attributs à plusieurs éléments différents

```
<xsl:attribute-set name="atts-communs">
   <xsl:attribute name="id">
     …
   </xsl:attribute>
   …
</xsl:attribute-set>

<xsl:template match="citation">
   <blockquote xsl:use-attribute-set="att-communs">
     <xsl:value-of select="."/>
   </blockquote>
</xsl:template>
```

Création des autres types de nœuds

- Texte
  - Insertion en le tapant dans le modèle
  - L'élément xsl:text contrôle strictement l'espacement
- Commentaire
  - L'élément xsl:comment

```
<xsl:comment>Ceci est un commentaire</xsl:comment>
<!--Ceci est un commentaire!-->
```

- Instruction
  - L'élément xsl:processing-instruction

```
<xsl:processing-instruction name="xml-stylesheet">
  href="book.css" type="text/css"
</xsl:processing-instruction>
<?xml-stylesheet href="book.css" type="text/css"?>
```

L'élément xsl:copy

- Copie d'un élément

```
<xsl:template match='test'>
   <xsl:copy>
     <xsl:apply-templates select='*'/>
   </xsl:copy>
</xsl:template>
```

- Copie d'un document

```
<xsl:template match='@* | node()'>
   <xsl:copy>
     <xsl:apply-templates select='@* | node()'/>
   </xsl:copy>
</xsl:template>
```

Programmation XSLT

- Structures de contrôle
- Structures itératives
- Templates-fonctions
  - Paramètres
  - Templates récursifs
  - Variables
- Tri

Les éléments xsl:if et xsl:choose

```
<xsl:template match="*">
   <xsl:if test="child::*" > chemin XPath
        J'ai au moins un enfant
   </xsl:if>
</xsl:template>

   <xsl:choose>
     <xsl:when test="child::*" > chemin XPath
       J'ai au moins un enfant
     </xsl:when>
     <xsl:otherwise>
       Je n'ai pas d'enfant
     </xsl:otherwise>
   </xsl:choose>
```

L'élément xsl:for-each

Création d'une liste numérotée de titres de chapitres (avant de traiter chaque chapitre (avec xsl:aply-templates))

```
<xsl:template match="livre">
   <xsl:for-each select="chapitre">
     <xsl:value-of select="position()"/>
     <xsl:value-of select="titre"/>
   </xsl:for-each>
   <xsl:apply-templates/>
</xsl:template>
```

Application d'une règle XSLT nommée

```
<xsl:template name="Afficher">
    <xsl:value-of select="position()"/> : <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="nom">
    <xsl:call-template name="Afficher"/>
</xsl:template>
```

Factorisation d'instructions utilisées en différents endroits

Les éléments xsl:param et xsl:with-param

Valeur par défaut

```
<xsl:template name="Afficher">
    <xsl:param name="texte">inconnu</xsl:param>
    <xsl:value-of select="position()"/> : <xsl:value-of select="$texte"/>
</xsl:template>

<xsl:template match="text()">
    <xsl:call-template name="Afficher">
        <xsl:with-param name="texte">texte vide</xsl:with-param>
    </xsl:call-template>
</xsl:template>
```

L'élément xsl:variable

```
<xsl:variable name="var" select="1"/>
<xsl:variable name="var" select="/etudiants/etudiant"/>
<xsl:variable name="var">ceci est la <v>valeur</v> de la variable<xsl:variable>
```

Variable **var** référencée par **$var**

- Variable globale
  Définie par un élément xsl:variable fils de <xsl:stylesheet>
- Variable locale
  Définie dans un corps de règle; visible dans tous les frères droits de l'élément xsl:variable et leurs descendants

Template récursif

```
<?xml version="1.0"?>
<etudiants>
    <etudiant>
        <nom>Dupont</nom>
        <notes>
            <note>14</note><note>12</note><note>16</note>
        </notes>
        …
    </etudiant>
    …
<etudiants>
```

Question: afficher la moyenne des notes de chaque étudiant

Template récursif

```
<xsl:template name="SommeNotes">
    <xsl:param name="listeNotes"/>
    <xsl:choose>
        <xsl:when test="$listeNotes">
            <xsl:variable name="note" select="$listeNotes[1]"/>
            <xsl:variable name="autresNotes">
                <xsl:call-template name="SommeNotes">
                    <xsl:with-param name="listeNotes"
                                    select="$listeNotes[position() != 1]"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:value-of select="$note + $autresNotes"/>
        </xsl:when>
        <xsl:otherwise>0</otherwise>
    </xsl:choose>
</xsl:template>
```

Template récursif

```
<xsl:template match="etudiant">
    <xsl:variable name="total">
        <xsl:call-template name="SommeNotes">
            <xsl:with-param name="listeNotes" select="notes/note"/>
        </xsl:call-template>
    </xsl:variable>
    <xsl:value-of select="nom"/> :  <xsl:value-of select=$total div count(notes/note)"/>
</xsl:template>
```

L'élément xsl:sort

Ce sont les éléments auxquels on applique un template qui sont triés (et donc les résultatsde l'application du template)

```
<xsl:template match="etudiants">
    <etudiantsTries>
        <xsl:apply-templates select="etudiant">
            <xsl:sort select="nom"/>
            <xsl:sort select="prenom"/>
        </xsl:apply-template>
    </etudiantsTries>
</xsl:template>

<xsl:template match="etudiant">
    <xsl:value-of select="nom"/>, <xsl:value-of select="prenom"/>
</xsl:template>
```

Tri alphabétique sur les noms et prénoms

Traiter plusieurs documents XML

```
<xsl:variable name='doc' select='document("other.xml")'/>
<xsl:value-of select='$doc/book/chapter'/>
```

Le document XML construit à partir du fichier other.xml est traité comme le document courant

Utiliser plusieurs feuilles de style XLST

- Modularité

  Import/inclusion dans une feuille XSLT de templates définis dans une autre feuille XSLT

  <xsl:import href="AutresRegles.xsl"/>

  <xsl:include href="AutresRegles.xsl"/>
- Résolution de conflits
  - Préséance des règles importées avec xsl:import
  - Pas de préséance des règles incluses avec xsl:include

Type de sortie d'une feuille XSLT

- <xsl:output method="xml"/>

  Format par défaut

  Espaces et entités prédéfinies sont gérées comme dans l'arbre XML source
- <xsl:output method="html"/>
- <xsl:output method="text"/>

  Construction d'un arbre résultat mais seules les données textuelles sont produites