LES MODES

- Objectif
  - ✓ un même nœud peut être traité plusieurs fois
- Exemple
  - ✓ on parcourt tous les chapitres et paragraphes pour produire une table des matières
  - ✓ on les parcourt à nouveau pour publier le contenu
- Il faut donc des règles différentes qui s'appliquent aux mêmes nœuds: c'est le mode qui va permettre la distinction
- Exemple: création de liens HTML
  - ✓ on peut créer des ancres «internes» à un document

    <a name='Alien'/>
  - ✓ on peut ensuite créer un lien vers cette ancre

    Lien vers le film Alien
  - ✓ objectif: une règle pour créer les liens, une autre pour créer les ancres

RÈGLES AVEC MODE

```
<xsl:template match="FILM" mode="Ancres">
   <a href="#{TITRE}"> <xsl:value-of select="TITRE"/> </a>
</xsl:template>

<xsl:template match="FILM">
   <a name="{TITRE}"/>
   <h1><xsl:value-of select="TITRE"/></h1>
   <b><xsl:value-of select="TITRE"/>,</b>
   <xsl:value-of select="GENRE"/> <br/>
   <b>Réalisateur</b>: <xsl:value-of select="MES"/>
</xsl:template>
```

L'APPEL DES RÈGLES

```
<xsl:template match="FILMS">
  <html>
    <head><title>Liste des films</title></head>
    <body bgcolor="white">
      <xsl:apply-templates select="FILM" mode="Ancres"/>
      <xsl:apply-templates select="FILM"/>
    </body>
  </html>
</xsl:template>
```

SYNTHÈSE: SÉLECTION D'UNE RÈGLE

- Soit un **xsl:apply-templates**, et N un des nœuds sélectionné
- On ne prend que les règles avec le même mode que xsl:apply-templates
- On teste le motif XPath pour savoir si le nœud satisfait la règle
- On prend celle qui a la plus grande priorité

ÉLÉMENTS DE PROGRAMMATION

- Traitement conditionnel: **xsl:if**
- Syntaxe

```
<xsl:if test = "boolean-expression">
    <!-- contenu -->
</xsl:if>
```

- Permet de changer l'output en fonction d'un test
- Attention: il n'existe pas de "else" (utilisez "choose" à la place)
- Cas d'utilisation: traitement d'un élément en fonction de sa position, des ses attributs,…

ÉLÉMENTS DE PROGRAMMATION (2)

- Traitement conditionnel: **xsl:choose**
- Syntaxe:

```
<xsl:choose>
    <!-- Content: (xsl:when+, xsl:otherwise?) -->
</xsl:choose>
<xsl:when test = boolean-expression>
    <!-- contenu -->
</xsl:when>
<xsl:otherwise>
    <!-- contenu -->
</xsl:otherwise>
```

- Cette définition dit:
  - ✓ on peut avoir plusieurs clauses avec un test (xsl:when).
  - ✓ la première vraie est utilisée (donc la série des xsl:when correspond à "if (){...}" elseif (){ ...}" elseif (){ ...}")
  - ✓ si aucune clause n'est vraie et s'il existe une clause xsl:otherwise, c'est cette dernière qui est exécutée (il s'agit donc du "else {...}")

ÉLÉMENTS DE PROGRAMMATION (3)

- Itération: **xsl:for-each**
- Syntaxe

```
<xsl:for-each select="motif-XPath">
    <!-- contenu -->
</xsl:for-each>
```

- Permet de parcourir un ensemble de nœuds et d'appliquer un traitement

ÉLÉMENTS DE PROGRAMMATION (4)

- Tri: xsl:sort
- Syntaxe

```
<xsl:sort
  select="motif-XPath"                          <!-- par défaut, . -->
  data-type="text|number"                       <!-- par défaut, text -->
  order="ascending|descending"                  <!-- par défaut, asc -->
  case-order="upper-first|lower-first"          <!-- par défaut, upper -->
  lang="nom_de_langue"                          <!-- par défaut, langue système -->
/>
```

- Associé à un parcours (**xsl:for-each** ou **xsl:apply-templates**)
- Permet de modifier l'ordre des nœuds

PARAMÈTRES

- XSLT ne connait pas de variables au sens des langages procéduraux
- Les paramètres
  - ✓ syntaxe

```
<xsl:param  name = qname
select = expression>
<!-- contenu -->
</xsl:param>
```

- Un paramètre lie un nom à une valeur. La valeur est définie soit dans l'attribut select, soit par son contenu (mais pas les deux !)
- Lorsqu'on définit un paramètre à la racine, il s'applique à tous les templates qui font appel à ce paramètre et qui ne reçoivent pas sa valeur. Cela ressemble à une constante globale

VARIABLES

- Le nom "variable" n’est pas clair. Il s’agit en fait de constantes.
- En règle générale, c’est utile pour faire des calculs "intermédiaires"
- Syntaxe

```
<xsl:variable  name = qname
select = expression>
<!-- contenu -->
</xsl:variable>
```

PARAMÈTRES, VARIABLES: EXEMPLE

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="text"/>
<!-- affichage du resultat -->
<xsl:template match="/factorielle">
    <xsl:variable name="x" select="valeur"/>
    <xsl:text>factorielle(</xsl:text><xsl:value-of select="$x"/><xsl:text>) = </xsl:text>
    <xsl:call-template name="factorielle">
        <xsl:with-param name="n" select="$x"/>
    </xsl:call-template>
</xsl:template>
<!-- fatorielle(n) - calcul de la factorielle -->
<xsl:template name="factorielle">
    <!-- on recupere le parametre, 1 valeur par defaut -->
    <xsl:param name="n" select="1"/>
    <!-- calcul -->
    <xsl:variable name="somme">
        <xsl:if test="$n = 1">1</xsl:if>
        <xsl:if test="$n != 1">
            <xsl:call-template name="factorielle">
                <xsl:with-param name="n" select="$n - 1"/>
            </xsl:call-template>
        </xsl:if>
    </xsl:variable>
    <xsl:value-of select="$somme * $n"/>
</xsl:template>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/xsl" href="fact.xsl"?>

<factorielle>
        <valeur>5</valeur>
</factorielle>
```

INSTRUCTIONS DIVERSES (NON DÉTAILLÉES)

- xsl:copy
- xsl:copy-of
- xsl:number
- xsl:element
- xsl:attribute
- xsl:attribute-set
- xsl:comment
- xsl:processing-instruction
- xsl:key
- xsl:message
- xsl:for-each-group
- xsl:function

XSLT ÉTEND XPATH

- ajout de la fonction generate-id(noeud) qui renvoie un identifiant unique :

```
<xsl:if test="generate-id($n1) = generate-id($n2)">
  ...
</xsl:if>
```

- ajout de la fonction current() qui renvoie le noeud courant

```
<xsl:template ...>
  <xsl:variable name="p" select="//article[@id = current()/@id]"/>
  ...
</xsl:template ...>
```

- ajout de la fonction key(index,clef),
- ajout du **ou** dans les expressions
- ajout de la fonction format-number(nombre,format) qui renvoie une chaîne issue du formatage d'un nombre :

```
<xsl:value-of select='format-number(500100, "###,###.00")' />
```

- ajout de la fonction document(URI) qui renvoie le document XML identifié par l'URI

```
<xsl:value-of select="document('stock.xml')//produit[prix > 10]" />
```