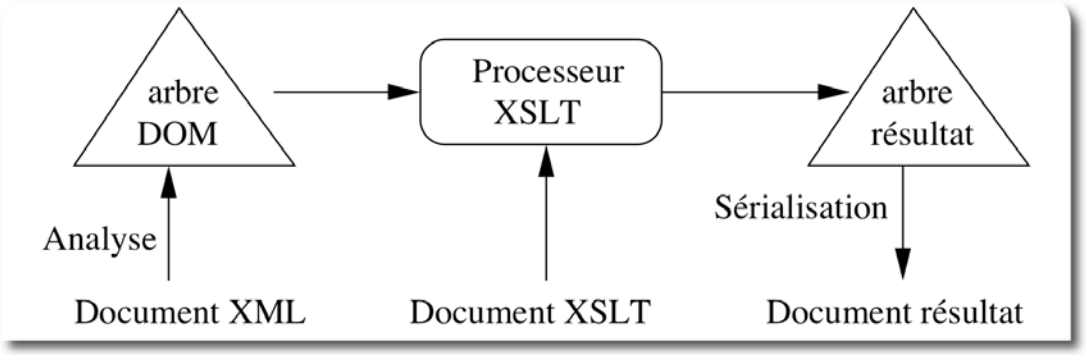
XSLT

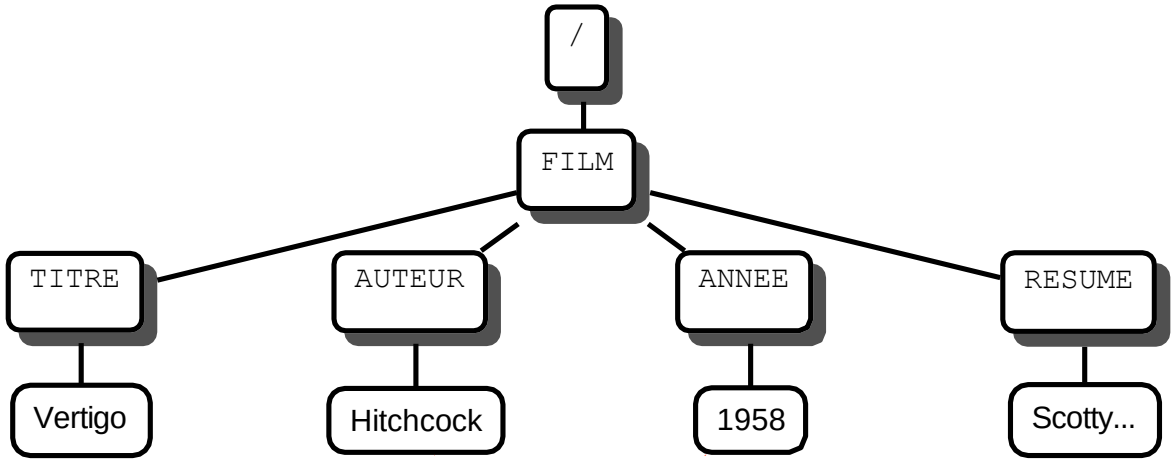
- Objectif: transformer un document XML en
  - ✓ un ou plusieurs documents XML, HTML, WML, SMIL
  - ✓ un document papier: PDF (XSL-FO), LaTeX
  - ✓ un texte simple



FONCTIONS D'UN PROGRAMME XSLT

- Transformation d'arbres XML:
  - ✓ extraction de données
  - ✓ génération de texte
  - ✓ suppression de contenu (nœuds)
  - ✓ déplacer le contenu (nœuds)
  - ✓ dupliquer le contenu (nœuds)
  - ✓ trier

EXEMPLE



- Règles de transformation

```
<xsl:template match="FILM">
    <p>
        <h1> <i> <xsl:value-of select="TITRE"/> </i> </h1>
        <i> <xsl:value-of select="ANNEE"/> </i>
        <p> <xsl:value-of select="AUTEUR"/> </p>
        <h3>Résumé: <xsl:value-of select="RESUME"/> </h3>
    </p>
</xsl:template>
```

Sélecteur d'éléments à transformer

Instructions de transformation

FONCTIONNALITÉS DE XSLT

- Extraction de données (xsl:value-of)

```
<xsl:template match="FILM">
    <xsl:value-of select="TITRE"/>
</xsl:template>
```

- Génération de texte (texte brut)

```
<xsl:template match="FILM">
    Ceci est le texte produit par application de cette règle.
</xsl:template>
```

- Génération d'un arbre XML (fragment XML bien formé)

```
<xsl:template match="FILM">
    <body>
        <p>Un paragraphe</p>
        </body>
</xsl:template>
```

FONCTIONNALITÉS XSLT (2)

- Génération d'arbre avec extraction de valeur

```
<xsl:template match="FILM">
    <body>
    <p>Titre:
        <xsl:value-of select="TITRE"/>
    </p>
    </body>
</xsl:template>
```

LES RÈGLES

- C'est la structure de base
- Règle = *template*: élément de base pour produire le résultat
  - ✓ une règle s'applique dans le contexte d'un nœud de l'arbre
  - ✓ l'application de la règle produit un fragment du résultat
- Programme XSLT = ensemble de règles pour construire un résultat

- Dans une règle, on peut:
  - ✓ accéder aux fils, aux descendants, au parent, aux frères, aux neveux, aux attributs,... du noeud à transformer (grâce à XPath)
  - ✓ effectuer des tests et des boucles,…
  - ✓ "appeler" d'autres règles (récursion)

EXEMPLE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="Salle.xsl" type="text/xsl"?>
<SALLE NO="1" PLACES="320" >
   <FILM>
      <TITRE>Alien</TITRE>
      <AUTEUR>RidleyScott</AUTEUR>
      <ANNEE>1979</ANNEE>
      <GENRE>Science-fiction</GENRE>
      <PAYS>EtatsUnis</PAYS>
      <RESUME>Près d'un vaisseau spatial échoué sur une lointaine planète, des Terriens
      en mission découvrent de bien étranges "oeufs". Ils en ramènent un à bord, ignorant
      qu'ils viennent d'introduire parmi eux un huitième passager particulièrement féroce et
      meurtrier. </RESUME>
   </FILM>
   <REMARQUE>Réservation conseillée</REMARQUE>
   <SEANCES>
      <SEANCE>15:00</SEANCE>
      <SEANCE>18:00</SEANCE>
      <SEANCE>21:00</SEANCE>
   </SEANCES>
</SALLE>
```

EXEMPLE (2)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="Salle.xsl" type="text/xsl"?>
<SALLE NO="1" PLACES="320" >
   <FILM>
      <TITRE>Alien</TITRE>
      <AUTEUR>RidleyScott</AUTEUR>
   <SEANCES>
      <SEANCE>15:00</SEANCE>
      <SEANCE>18:00</SEANCE>
      <SEANCE>21:00</SEANCE>
   </SEANCES>
</SALLE>
```

- Boucle: traduction de l'élément XML <SALLES> en élément HTML

```
<xsl:template match="SALLE">
    <h2>Salle No <xsl:value-of select="@NO"/></h2>  Film:<xsl:value-of select="FILM/TITRE"/>
    de <xsl:value-of select="FILM/AUTEUR"/>
    <ol>
        <xsl:for-each select="SEANCES/SEANCE">
            <li><xsl:value-of select="."/></li>
        </xsl:for-each>
    </ol>
</xsl:template>
```

```
<h2>Salle No 1</h2>
Film: Alien
de Ridley Scott
<ol>
<li>15:00</li>
<li>18:00</li>
<li>21:00</li>
</ol>
```

- Remarque: c'est un fragment HTML, à intégrer dans un document complet

APPEL DES RÈGLES

- En général, on produit un résultat en combinant plusieurs règles:
  - ✓ la règle initiale s'applique à la racine du document traité ('/')
  - ✓ on produit alors le cadre du document HTML
  - ✓ on appelle d'autres règles pour compléter la création du résultat
- Exemple

```
<xsl:template match="/">
<html>
    <head><title>Programme de <xsl:value-of select="CINEMA/NOM"/></title></head>
    <body bgcolor="white">
        <xsl:apply-templates select="CINEMA"/>
    </body>
</html>
</xsl:template>
```

Nouvelle règle

RÈGLE CINEMA

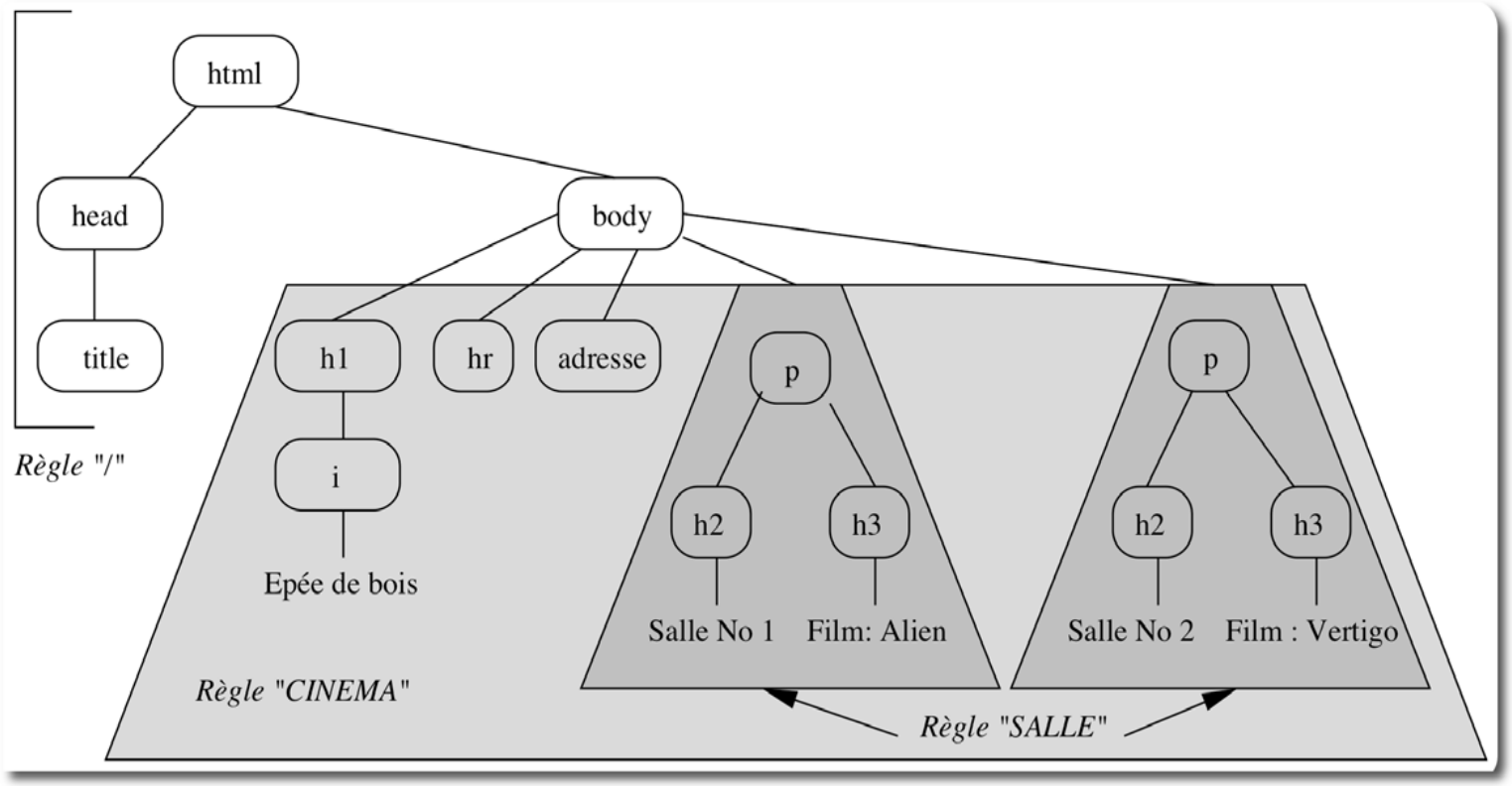
- Exploitation de l'élément CINEMA, puis appel à la règle SALLE

```
<xsl:template match="CINEMA">
    <h1><i><xsl:value-of select="NOM"/></i></h1><hr/>
    <xsl:value-of select="ADRESSE"/>,
    <i>Métro:</i> <xsl:value-of select="METRO"/>
    <hr/>
    <xsl:apply-templates select="SALLE"/>
</xsl:template>
```

Appel de la règle SALLE

```
<xsl:template match="SALLE">
    <h2>Salle No <xsl:value-of select="@NO"/></h2>
    Film:<xsl:value-of select="FILM/TITRE"/>
    de <xsl:value-of select="FILM/AUTEUR"/>
    <ol>
        <xsl:for-each select="SEANCES/SEANCE">
            <li><xsl:value-of select="."/></li>
        </xsl:for-each>
    </ol>
</xsl:template>
```

VUE D'ENSEMBLE



PROGRAMME XSLT

- Un programme XSLT consiste à produire un document résultat à partir d'un document source
- Un programme XSLT est un document XML
- Les éléments XSLT sont différenciés grâce à un espace de noms xsl:

Élément racine du programme

Règle

Corps de la règle: instructions

```
<xsl:template match="COURS">
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL
<html>
    <head><title>Fiche du cours</title></head>
        <body bgcolor="white">
        <p>
        <h1><i><xsl:value-of select="SUJET"/></i></h1>
        <hr/>
        <xsl:apply-templates/>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

ÉLÉMENTS DE PREMIER NIVEAU

- **xsl:import**
  - ✓ pour importer un programme XSLT (doit être avant include, les règles importées sont prioritaires)
- **xsl:include**
  - ✓ pour inclure un programme XSLT (les règles sont au même niveau)
- **xsl:output**
  - ✓ pour définir le format de sortie
- **xsl:param**
  - ✓ pour définir un paramètre
- **xsl:variable**
  - ✓ pour définir une variable
- **xsl:template**
  - ✓ pour définir une règle

RÈGLES

- Définition
  - ✓ une règle est définie par l'élement xsl:template
  - ✓ deux possibilités
    - l'attribut match est une expression XPath définissant les éléments sources de la règle

      <xsl:template match="FILM">
    - l'attribut name donne un nom à la règle

      <xsl:template name="TDM">
- Déclenchement
  - ✓ pour le premier type de règle

    <xsl:apply-templates select="…">
  - ✓ pour le deuxième type de règle

    <xsl:call-template name="…">

SÉLECTION DES RÈGLES

- Problème
  - ✓ étant donné un nœud, comment trouver la règle qui s'applique ?
- Algorithme
  - ✓ soit N le nœud
  - ✓ soit P le motif (pattern) de la règle R
  - ✓ s'il existe quelque part un nœud C tel que l'évaluation de P à partir de C contient N, alors la règle s'applique
- Exemple: la règle pour la racine
  - ✓ le nœud contexte N est la racine du document
  - ✓ il existe une règle R dont le motif est "/"
  - ✓ en prenant n'importe quel nœud, l'évaluation de "/" est N, donc la règle s'applique
- Il est donc préférable (mais pas obligatoire) d'avoir une règle "/"

SÉLECTION DES RÈGLES (2)

- Un motif de sélection est une expression XPath restreinte
  - ✓ les fils d'un élément (axe child)
  - ✓ les attributs d'un élément (axe attribute)
  - ✓ la simplification // (axe /descendant-or-self::node()/)
- Cette restriction garantit que l'on peut savoir si une règle doit être déclenchée pour un nœud N uniquement en analysant les ancêtres de N
- Cela diminue considérablement la complexité de l'algorithme de sélection
- Exemples

  /COURS/ENSEIGNANTS: la règle s'applique à tous les nœuds ENSEIGNANTS fils d'un élément racine COURS

  //SEANCE[@ID=2]: … à tous les nœuds de type SEANCE ayant un attribut ID valant 2

  /descendant::FILM[1]: … au premier élément de type FILM dans le document FILM[1]: … aux premiers fils de type FILM (il peut y en avoir plusieurs!)

  /COURS[@CODE="TC234"]: … aux cours avec le code TC234

RÈGLES PAR DÉFAUT

- Lorsque aucune règle n'est sélectionnée, le moteur XSLT applique des règles par défaut
- La première règle pour les éléments et la racine du document

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

  - ✓ on demande l'application de règles pour les fils du nœud courant
- La deuxième règle insère dans le document résultat la valeur du nœud ou de l'attribut

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

  - ✓ cela suppose (en particulier pour les attributs) d'avoir utilisé un xsl:apply- templates qui ait sélectionné ces nœuds
- La troisième règle concerne les processing-instructions et les commentaires

CONSÉQUENCE

- Si on se contente des règles par défaut, on obtient la concaténation de nœuds de type Text
- Programme minimal:

```
<?xmlversion="1.0"encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

XSL:APPLY-TEMPLATES

- C’est une instruction qui possède 3 attributs
  - select
  - mode
  - priority
- **select** doit sélectionner un ensemble de nœuds
  - ces nœuds consituent le contexte d’utilisation
  - pour chaque nœud, on va rechercher la règle à instancier
- **mode** permet de choisir explicitement une des règles à instancier parmi celles qui sont candidates
- **priority** permet de définir une priorité pour le processeur puisse choisir

SÉLECTION D'UNE RÈGLE

- Comment gérer le fait que plusieurs règles sont éligibles pour un même nœud ?
  - ✓ il existe des priorités implicites qui permettent au processeur de choisir
  - ✓ on peut donner explicitement une priorité
  - ✓ si malgré cela, le choix est impossible, le processeur s'arrête
- Exemple: on souhaite effacer certains nœuds

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<FILM>
    <TITRE>Vertigo</TITRE>
    <ANNEE>1958</ANNEE><GENRE>Drame</GENRE>
    <MES>Alfred Hitchcock</MES>
    <RESUME>Scottie Ferguson, ancien inspecteur de police, est sujet au vertige…</RESUME>
</FILM>
<FILM>
    <TITRE>Alien</TITRE>
    <ANNEE>1979</ANNEE><GENRE>Science-fiction</GENRE>
    <MES>Ridley Scott</MES>
    <RESUME>Près d'un vaisseau spatial échoué sur une lointaine…</RESUME>
</FILM>
```

PROGRAMME XSLT

- Ce programme permet d'effacer les nœuds de type RESUME

```
<!-- on ne recopie pas les nœuds RESUME dans le document résultat -->
<xsl:template match="RESUME"/>
<!-- on recopie les autres nœuds -->
<xsl:template match="@*|node()" priority="-1">
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
                                        </xsl:copy>
                                    </xsl:template>
```

PRIORITÉS IMPLICITES

- Idée: plus c'est «spécifique», plus c'est prioritaire
- Priorité 0: les motifs constitués d'une seule étape XPath, avec un nom d'élément ou d'attribut et sans prédicat
- Priorité -0.5: les filtres autres qu'un nom d'élément ou d'attribut ont une priorité égale à -0,5 (node(),*)
- Tous les autres ont une priorité de 0.5 (prédicats, plusieurs étapes)