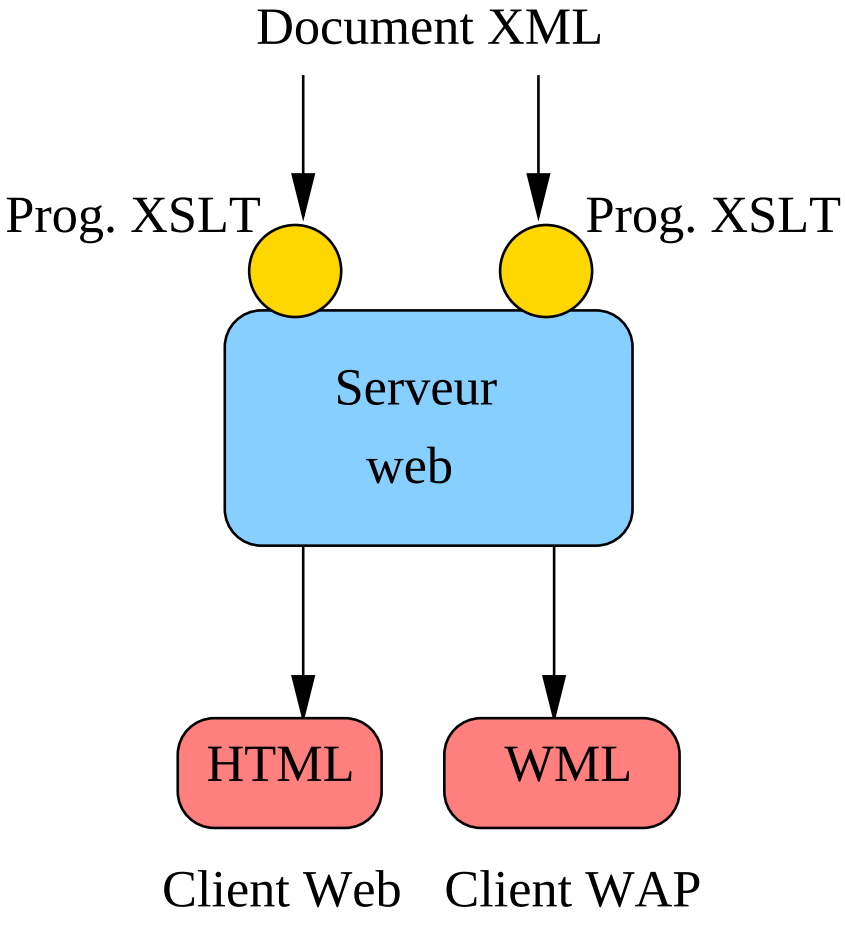
XSLT
Le langage de transformation de documents XML

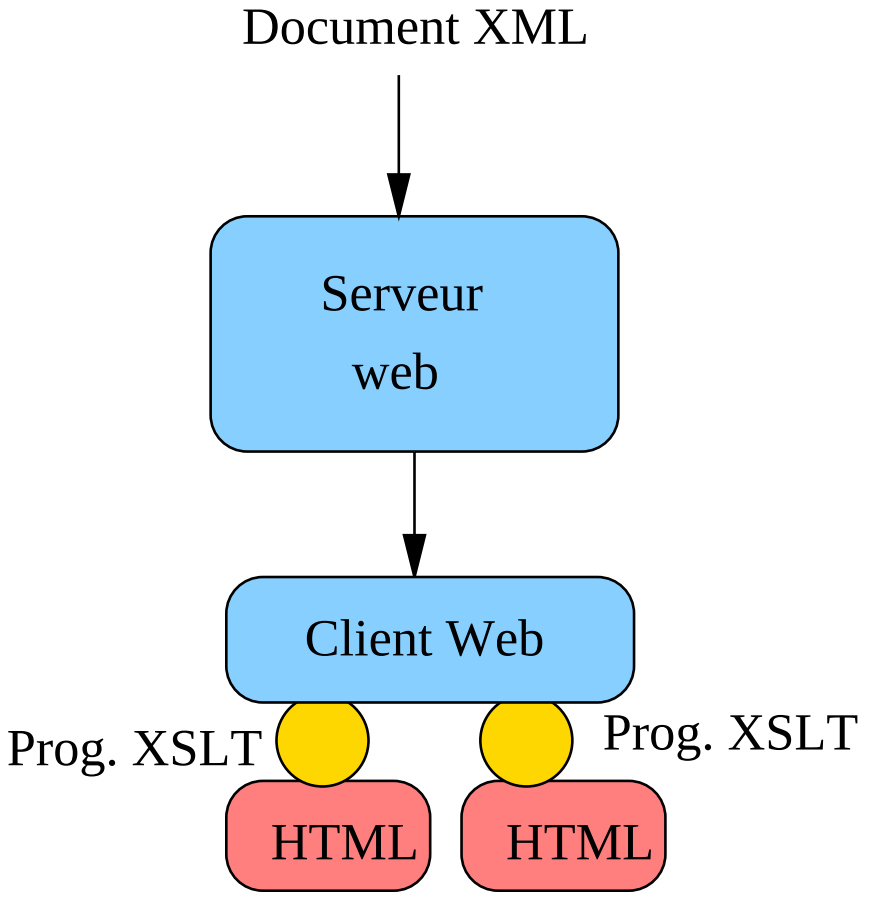
Sommaire

Une introduction à XSLT, destinée à comprendre les **mécanismes** du langage.

- Règles XSLT
- Désignation de fragments XML
- Appels de règles
- Application : XML -> HTML et XML -> WML
- Passage de paramètres

XSLT, où, quand, comment





Structure de base : les règles

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

Un exemple

Produire une phrase quand on rencontre un nœud `FILM` :

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

- **Motif de sélection** : `match="FILM"`
- **Corps de la règle** = contenu de l'élément

Extraction de données

On insére dans le résultat des fragments du document traité

- Exemple : recherche du titre pour le nœud `FILM` :

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

- Plus généralement : on donne un chemin d'accès à un nœud à partir du nœud courant

Illustration

Contexte d'application de la règle XSLT

select="TITRE"

Vertigo	Hitchcock	1958	Drame	USA	Scotty...

Une règle complète

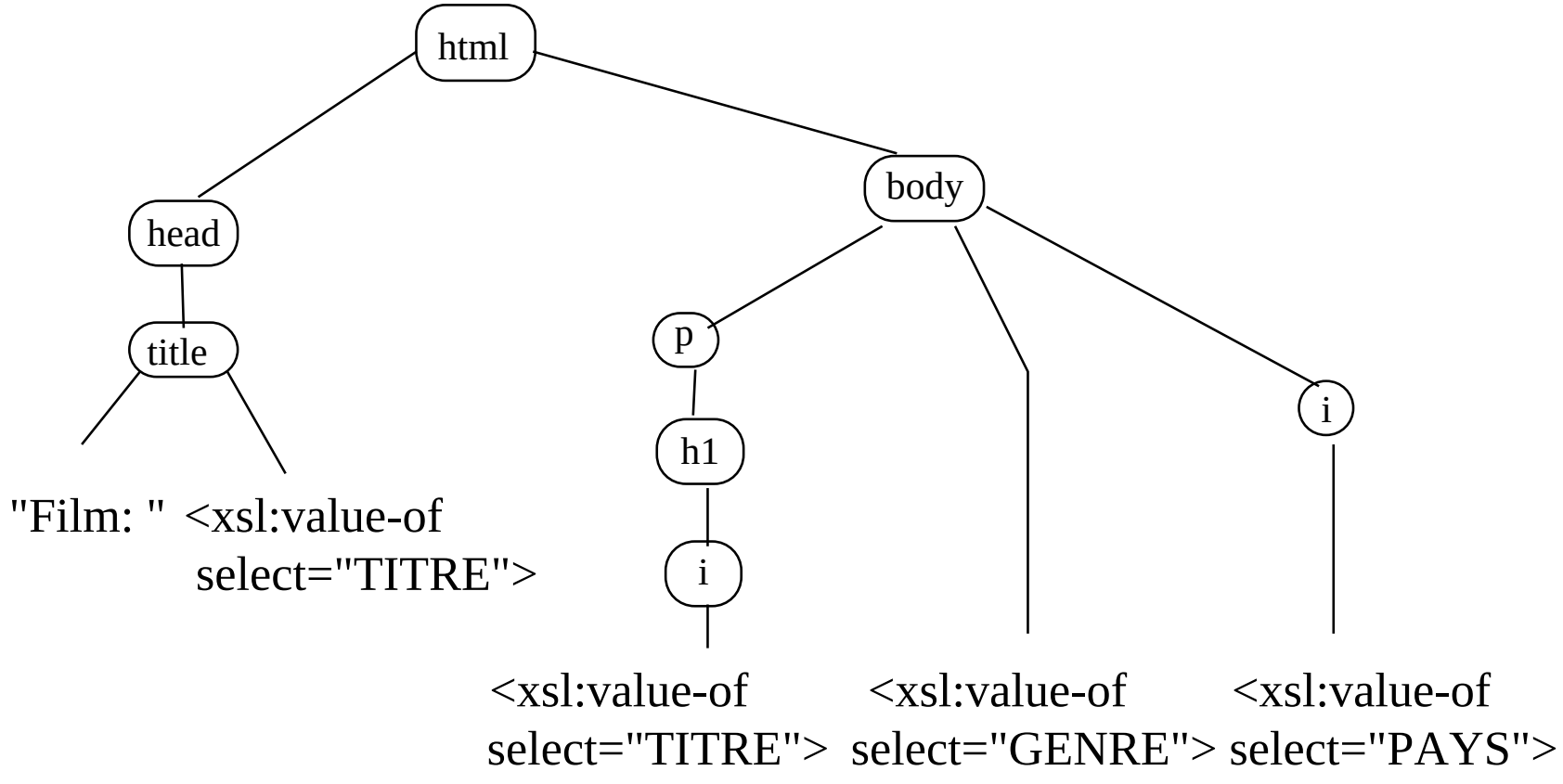
```
<xsl:template match="FILM">
 <html>
   <head>
    <title>Film:
      <xsl:value-of select="TITRE"/>
    </title>
   </head>
   <body>
     Le genre du film, c'est
     <b><xsl:value-of select="GENRE"/></b>
   </body>
 </html>
</xsl:template>
```

Le résultat

Appliquée à *Vertigo.xml*, on obtient :

```
<html>
  <head>
    <title>Film:
        Vertigo
    </title>
  </head>
  <body>
    Le genre du film, c'est
      <b>Suspense</b>
  </body>
</html>
```

Produire un document HTML



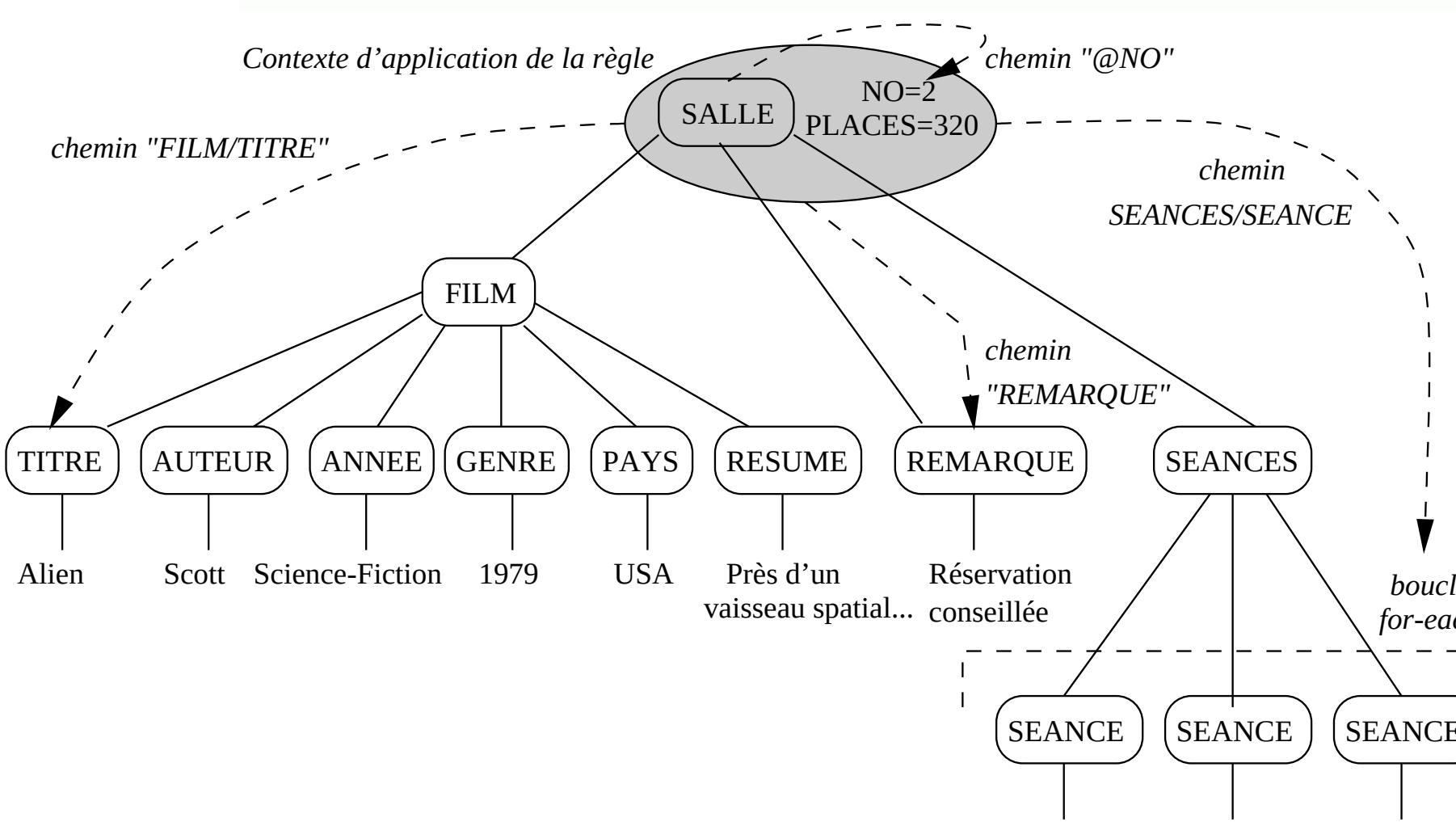
Chemins complexes

Dans l'exemple précédent, on accédait aux fils d'un nœud.
En fait on peut :

- Accéder à tous les descendants
- Accéder aux parents, aux frères, aux neveux...
- Accéder aux attributs
- Effectuer des boucles

Le langage d'expression de chemins : XPath.

Salles et séances



Chemins complexes

- **Descendants** : le titre du film est désigné par `select="FILM/TITRE"` ;
- **Attributs** : le numéro de la salle est désigné par `select="@NO"` ;
- **Boucles** : avec `xsl:for-each` sur l'ensemble des séances désignées par `select="SEANCES/SEANCE"` ;
- **L'élément contexte** : désigné par le symbole '.', comme dans `select="."`.

Application

Quelques exemples de chemins :

```
<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>
```

Le résultat

Appliqué à *Alien.xml* :

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

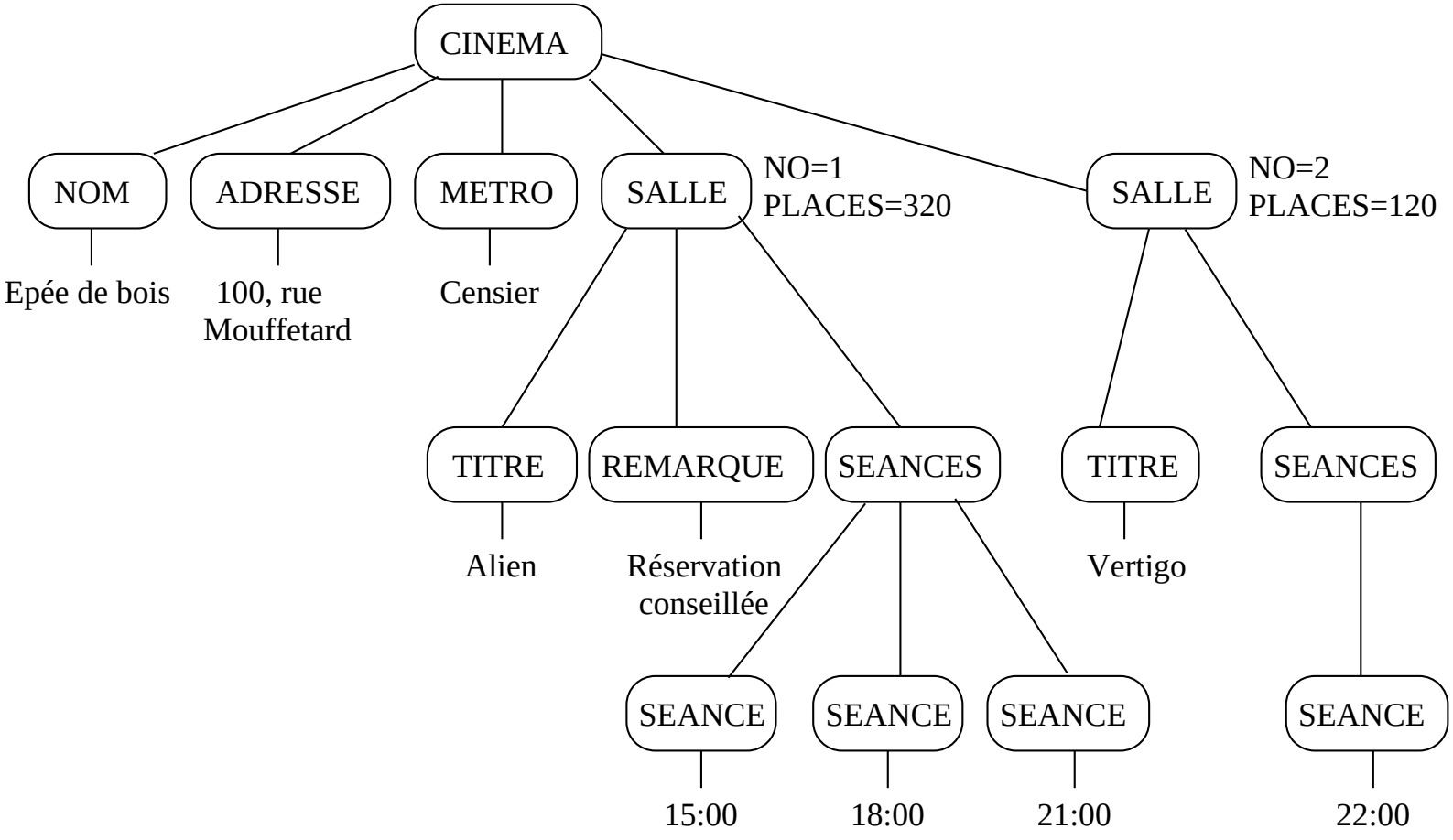
NB : c'est un **fragment HTML**, à intégrer dans un document complet.

Appels de 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

Application au cinéma



Première règle

« Cadre » HTML, puis appel de la règle `CINEMA`

```
<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>
```

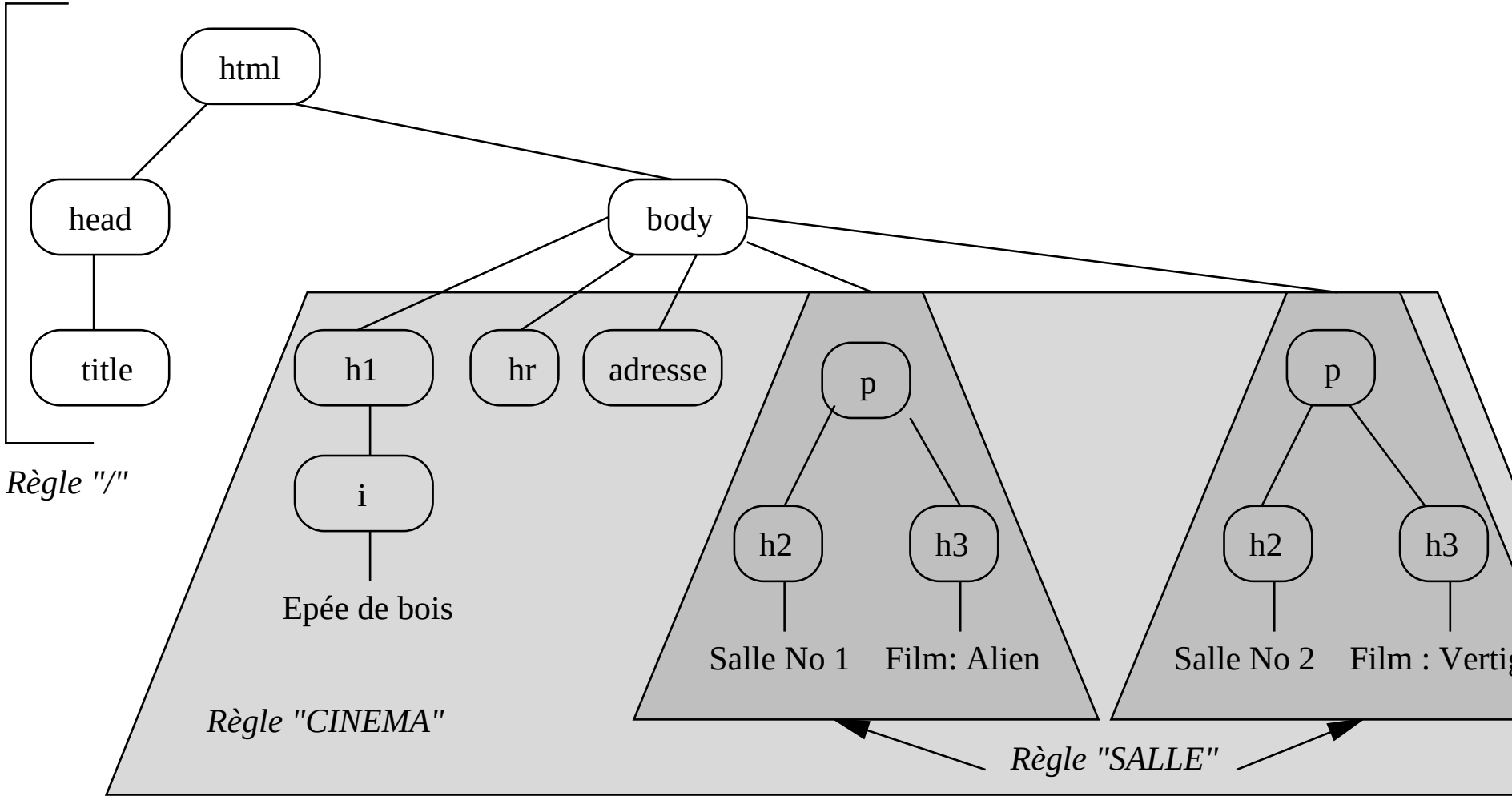
Règle CINEMA

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>
```

Vue d'ensemble Démo



La même chose, en WML

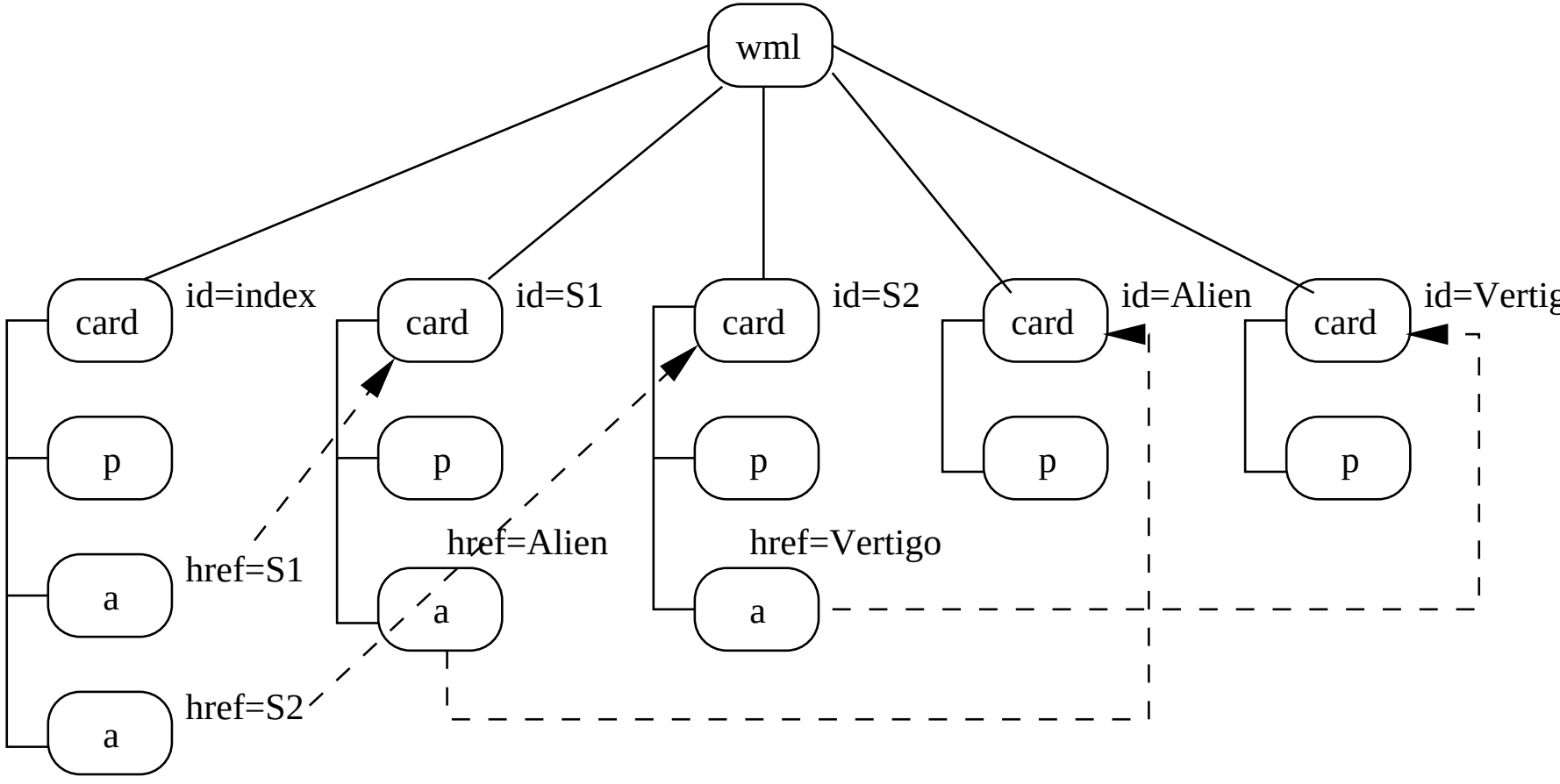
On génère les cartes du site

```
<xsl:template match="/">
 <wml>
   <!-- Carte d'accueil -->
   <xsl:apply-templates select="CINEMA"/>

   <!-- Cartes pour les salles et séances --
   <xsl:apply-templates select="CINEMA/SALLE

   <!-- création des cartes pour les films -
   <xsl:apply-templates select=".//FILM"/>
 </wml>
</xsl:template>
```

Arbre XML du site



Règle CINEMA

```
<xsl:template match="CINEMA">
  <card id="index" title="Programme">
    <p align="center">
     <xsl:value-of select="NOM"/>
    </p>
    <xsl:for-each select="SALLE">
      <p> <a href="#S{@NO}">
            Salle <xsl:value-of select="@NO"/
          </a>
            <xsl:value-of select="FILM/TITRE
      </p>
    </xsl:for-each>
  </card>
</xsl:template>
```

Résultat

```
<card id="index" title="Programme">
  <p align="center">
    Ep&#xE9;e de bois
  </p>
  <p>
    <a href="#S1"> Salle 1: </a>
    Alien
  </p>
  <p>
    <a href="#S2"> Salle 2: </a>
    Vertigo
  </p>
</card>
```

Règle SALLE

```
<xsl:template match="SALLE">
 <card id="S{@NO}">
   <p>Séances salle
       <xsl:value-of select="@NO"/></p>
   <p>
    <a href="#{FILM/TITRE}">
      Film :
       <xsl:value-of select="FILM/TITRE"/>
      </a>
    </p>
  <xsl:apply-templates select="SEANCES"/>
 </card>
</xsl:template>
```

Résultat

```
<card id="S2">
  <p>
    S&#xE9;ances salle 2
  </p>
  <p>
    <a href="#Vertigo">
        Film : Vertigo</a>
  </p>
  <p>
    S&#xE9;ance 22:00
  </p>
</card>
```

*XSLT avec paramètres (*Démo*)*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0/
        "http://www.w3.org/TR/REC-html40/stri
<html>
  <head>
    <title>Formulaire de Recherche</title>
  </head>
  <body bgcolor="white">
    <h1>Formulaire de Recherche</h1>
    <form method='get' action='Moteur.xml' r
      Film:          <INPUT TYPE='TEXT' NAME='
      Séance:        <INPUT TYPE='TEXT' NAME='
      Ville:         <INPUT TYPE='TEXT' NAME='
      <input type='submit' name='chercher' v
    </form>
  </body>
</html>
```

Le document XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MOTEUR [
 <!ENTITY EpeeDeBois
     SYSTEM "http://epee-de-bois.fr/EDB.xml"
 <!ENTITY CineMarseille
     SYSTEM "http://cine-marseille.fr/CM.xml
]>
<MOTEUR>
  <CINEMA>
    &EpeeDeBois;
   </CINEMA>
  <CINEMA>
    &CineMarseille;
   </CINEMA>
</MOTEUR>
```

Traitement des paramètres

```
...
<xsl:param name="titre"/>
<xsl:param name="seance"/>
<xsl:param name="ville"/>

<xsl:template match="MOTEUR">
 <xsl:for-each select="CINEMA">
   <xsl:if test="
         ($titre = ”  or TITRE = $titre)
     and ($seance = ” or HEURE &gt;= $seance
     and ($ville = ”  or VILLE = $ville)">
     <xsl:apply-templates select="." /><p/>
   </xsl:if>
 </xsl:for-each>
</xsl:template>
```

Conclusion sur XSLT

Un langage totalement adapté au traitement de documents XML

- Parcours d'un document, vu comme un arbre
- Déclenchement de règles sur certains nœuds
- Association de plusieurs programmes à un même document