2016

SER – Laboratoire 3

HEIG-VD

ADRIANO RUBERTO & MATTHIEU VILLARD

Table des matières

1 INTRODUCTION

Dans les précédents laboratoires, nous avons élaboré une structure xml pour échanger des données et avons conçu l'application permettant de les générer.

Ce laboratoire se base sur le travail précédemment effectué pour générer des pages web consultables par un client. Ainsi, le but est d'exercer l'utilisation de feuilles de styles XSL pour formater des données provenant des documents xml. En effet, nous récupérons les données stockées dans ces documents, et les formatons avec une feuille de style XSL pour les transformer en document html affichable dans un navigateur web.

2 GÉNÉRATION DES DOCUMENTS HTML

Les documents html sont générés à l'aide d'une simple feuille de style XSL, à savoir le fichier *projections.xsl*.

Puisque les documents que nous générons sont de simples fichiers html, ceux-ci sont donc générés de manière statique. De ce fait, il n'est pas possible d'en modifier le contenu après leur création. Et donc, pour permettre à l'utilisateur d'afficher les projections triées selon plusieurs critères variables, il nous incombait de créer plusieurs fichiers, à savoir les fichiers *index.html*, *projections_par_date.html* et *projections_par_classement.html*. Il en est de même pour chaque fiche détaillée d'un acteur ou d'un film.

Nous devons relever que puisque nous travaillons avec un seul fichier xml contenant toutes les données groupées par projection, il y a un risque de doublons. Ainsi, nous avons dû éliminer ces doublons. Bien que notre solution soit fonctionnelle, nous tenons à préciser que si deux films différents venaient à porter le même nom (ou deux acteurs), cela créerait une confusion. En effet, nous créons un fichier pour chaque film (ou acteur) en nous basant sur leurs noms pour les identifier et les relier à leur fichier. Ainsi, la situation où deux films avec un nom identique seraient présents, provoquerait sans nul doute la perte des données de l'un d'eux. Dans ce cas, il serait préférable d'utiliser des identifiants.

Pour commencer, nous allons nous intéresser aux instructions exécutées en tout début, celles qui lancent la génération de chaque type de fichier. Ces instruction se trouvent dans le bloc <xsl:template match="/">. Ce bloc est exécuté en premier puisqu'il est lié à l'élément racine.

Les fichiers générés sont les suivants :

- Index.html : contient la liste des projections triées par titre de film
- projections_par_date.html : contient la liste des projections triées par date
- projections_par_classement.html : contient la liste des projections triées par la note moyenne
- liste_des_films.html : contient la liste de tous les films actuellement à l'affiche
- film_{*titre_du_film*}.html : contient la fiche détaillée du film ayant pour titre *titre_du_film*
- liste_des_acteurs.html : liste de tous les acteurs ayant joué dans un des films à l'affiche
- acteur_{*nom_acteur*}.html : contient la fiche détaillée de l'acteur portant le nom *nom_acteur*

2.1 TEMPLATES GÉNÉRIQUES

Plusieurs templates sont utilisé pour la génération de chaque document html. En tout, il y en a trois.

```
<!-- Entête html -->
<xsl:template name="entete_html">
    <title>Listes des projections</title>
    <style type="text/css">
        @import url("./css/style.css");
    </style>
</xsl:template>
```

Le template ci-dessus se charge d'inclure la description de l'application ainsi que la feuille de style css qui permet d'ajouter un peu de style à notre affichage.

```
<!-- Menus principaux -->
<xsl:template name="menu">
    <a href="index.html">Projections par titre de film</a>
    <a href="projections_par_date.html">Projections par date</a>
    <a href="projections_par_classement.html">Projections par classement</a>
</xsl:template>
```

Le template ci-dessus ajoute les menus principaux qui permettent à l'utilisateur d'accéder rapidement aux différentes listes de projections, chacune ordonnée selon un critère différent.

```
<!-- Pied de page du document -->
<xsl:template name="pied_de_page">
    <br/>
    <br/>
    <a href="liste_des_films.html">Liste des films</a>
    <xsl:text> </xsl:text>
    <a href="liste_des_acteurs.html">Liste des acteurs</a>
</xsl:template>
```

Ce template inclut un pied de page qui comprend deux liens permettant d'afficher la liste des films ou la liste des acteurs.

2.2 PROJECTIONS

2.2.1 Templates communs

Etant donné que les listes de projections affichent les même données mais dans un ordre différents, il est normal de pouvoir factoriser une partie de l'affichage. Les templates suivant remplissent ce rôle.

```
<!-- Entete du tableau contenant les projections -->
<xsl:template name="entete_tableau">
    <li class="listItemLabel">
        <span class="listItemElement titre">Film</span>
        <span class="listItemElement date">Date</span>
        <span class="listItemElement salle">Salle</span>
        <span class="listItemElement grade">Classement</span>
    </li>
</xsl:template>
```

Ce template affiche l'entête du tableau contenant les projections.

```
<!-- Ligne du tableau des projections -->
<xsl:template match="projection" mode="tableau">
  <xsl:variable name="titre_du_film" select="translate(film/titre, translate(film/titre, $allowedChars, ''), '')" />
  <li class="listItem">
    <span class="listItemElement titre">
      <a href="film_{$titre_du_film}.html">
        <xsl:if test="film/photo">
          <xsl:variable name="photo_du_film" select="film/photo" />
          <img alt="photo_du_film" src="resources/{$photo_du_film}"/>
        </xsl:if>
        <xsl:value-of select="film/titre"/>
      </a>
    </span>
    <span class="listItemElement date">
      <xsl:value-of select="substring(date,9,2)"/>.<xsl:value-of select="substring(date,6,2)"/>.<xsl:value-of select="substring(date,1,4)"/><xsl:text> </xsl:text>
      <xsl:value-of select="substring(date,12,2)"/>:<xsl:copy-of select="substring(date,15,2)"/>
    </span>
    <span class="listItemElement salle">
      <xsl:value-of select="noSalle"/>
    </span>
    <span class="listItemElement grade">
      <xsl:if test=".//note"><xsl:value-of select="avg(.//note)"/> / 5</xsl:if>
      <xsl:if test="not(.//note)"> - </xsl:if>
    </span>
  </li>
</xsl:template>
```

Ce template se charge d'afficher les données pour une projection. On constate que la photo n'est affichée que si elle existe et que la date est transformée dans un format plus convivial. On notera également la présence d'une fonction d'agrégat *avg* qui se charge de calculer la moyenne des notes attribuées au film.

Dans ce template on peut observer la présence de l'instruction

```
select="translate(film/titre, translate(film/titre, $allowedChars, ''), '')"
```

Celle-ci permet de supprimer les caractères spéciaux, ce qui est utile pour les url. Cette technique sera réutilisée à chaque écriture d'une url.

2.2.2 Index.html

```
<!-- Generation du fichier index.html - Projections triées par titre de film -->
<xsl:result-document method="html" href="index.html" >
  <html>
    <head>
      <xsl:call-template name="entete_html"/>
    </head>
    <body>
      <xsl:call-template name="menu" />
      <xsl:apply-templates select="projections" mode="organisee_par_titre"/>
      <xsl:call-template name="pied_de_page" />
    </body>
  </html>
</xsl:result-document>
```

Ce template initialize la génération du document *index.html*, en construisant la structure de base et en appelant le template *organisee_par_titre.* Il écrit ensuite le résultat dans le document *index.html*.

```
<!-- Liste des projections triée par titre de film -->
<xsl:template match="projections" mode="organisee_par_titre">
  <h1>Projections par titre de film</h1>
  <ul class="list">
    <xsl:call-template name="entete_tableau"/>
    <xsl:for-each select="projection">
      <xsl:sort order="ascending" select="film/titre"/>
      <xsl:apply-templates select="." mode="tableau"/>
    </xsl:for-each>
  </ul>
</xsl:template>
```

Ce template affiche les projections en les triant par titre de film. A chaque projection, on applique le template déjà décrit précédement, dans la section templates communs.

2.2.3 projections_par_date.html

```
<!-- Generation du fichier projections_par_date.html - Projections triées par date -->
<xsl:result-document method="html" href="projections_par_date.html" >
  <html>
    <head>
      <xsl:call-template name="entete_html"/>
    </head>
    <body>
      <xsl:call-template name="menu" />
      <xsl:apply-templates select="projections" mode="organisee_par_date"/>
      <xsl:call-template name="pied_de_page" />
    </body>
  </html>
</xsl:result-document>
```

Ce template initialize la génération du document *projections_par_date.html*, en construisant la structure de base et en appelant le template *organisee_par_date.* Il écrit ensuite le résultat dans le document *projections_par_date.html*.

```
<!-- Liste des projections triée par date de projection -->
<xsl:template match="projections" mode="organisee_par_date">
  <h1>Projections par date</h1>
  <ul class="list">
    <xsl:call-template name="entete_tableau"/>
    <xsl:for-each select="projection">
      <xsl:sort order="ascending" select="date"/>
      <xsl:sort order="ascending" select="film/titre"/>
      <xsl:apply-templates select="." mode="tableau"/>
    </xsl:for-each>
  </ul>
</xsl:template>
```

Ce template affiche les projections en les triant par date puis par titre de film. A chaque projection, on applique le template chargé d'afficher les information de la projection.

2.2.4 projections_par_classement.html

```
<!-- Generation du fichier projections_par_classement.html - Projections triées par classement -->
<xsl:result-document method="html" href="projections_par_classement.html" >
  <html>
    <head>
      <xsl:call-template name="entete_html"/>
    </head>
    <body>
      <xsl:call-template name="menu" />
      <xsl:apply-templates select="projections" mode="organisee_par_classement"/>
      <xsl:call-template name="pied_de_page" />
    </body>
  </html>
</xsl:result-document>
```

Ce template initialize la génération du document *projections_par_classement.html*, en construisant la structure de base et en appelant le template *organisee_par_ classement.* Il écrit ensuite le résultat dans le document *projections_par_ classement.html*.

```
<!-- Liste des projections triée par classement -->
<xsl:template match="projections" mode="organisee_par_classement">
    <h1>Projections par classement</h1>
    <ul class="list">
        <xsl:call-template name="entete_tableau"/>
        <xsl:for-each select="projection">
            <xsl:sort order="descending" select="avg(.//note)"/>
            <xsl:sort order="ascending" select="film/titre"/>
            <xsl:apply-templates select="." mode="tableau"/>
        </xsl:for-each>
    </ul>
</xsl:template>
```

Ce template affiche les projections en les triant par la moyenne des notes qui leur ont été attribuées puis par titre de film. A chaque projection, on applique le template chargé d'afficher les information de la projection.

2.3 FILMS

2.3.1 liste_des_films.html

```
<!-- Generation du fichier liste_des_films.html -->
<xsl:result-document method="html" href="liste_des_films.html" >
    <html>
        <head>
            <xsl:call-template name="entete_html"/>
        </head>
        <body>
            <xsl:call-template name="menu" />
            <xsl:call-template name="liste_des_films"/>
            <xsl:call-template name="pied_de_page" />
        </body>
    </html>
</xsl:result-document>
```

Ce template initialize la génération du document *liste_des_films.html*, en construisant la structure de base et en appelant le template *liste_des_films.* Il écrit ensuite le résultat dans le document *liste_des_films.html*.

```
<!-- Liste de tous les films des projections -->
<xsl:template name="liste_des_films">
    <h1>Liste des films</h1>
    <xsl:for-each select="//film[not( preceding::film/titre = titre)]">
        <xsl:sort order="ascending" select="titre"/>
        <xsl:apply-templates select="." mode="liste"/>
    </xsl:for-each>
</xsl:template>
```

Ce template affiche les films en les triant par titre. A chaque film, on applique le template chargé d'afficher son nom et sa photo, décrit ci-dessous.

On notera la présence de l'instruction de sélection //film[not(preceding::film/titre = titre)]. Celle-ci permet de supprimer les doublons.

```
<!-- Affichage d'une ligne de la liste des films -->
<xsl:template match="film" mode="liste">
  <xsl:variable name="titre_du_film" select="translate(titre, translate(titre, $allowedChars, ''), '')" />
  <li>
    <b>
      <a href="film_{$titre_du_film}.html">
        <span>
          <xsl:if test="photo">
            <xsl:variable name="photo_du_film" select="photo" />
            <img alt="photo_du_film" src="resources/{$photo_du_film}"/>
            <br/>
          </xsl:if>
          <xsl:value-of select="titre"/>
        </span>
      </a>
    </b>
    <br/>
    <br/>
  </li>
</xsl:template>
```

Ce template se charge d'afficher un aperçu pour un film. On constate que la photo n'est affichée que si elle. Les seules informations affichées sont le titre du film et sa photo.

2.3.2 film_{titre_du_film}.html

```
<!-- Generation des fichiers de film -->
<xsl:for-each select="//film[not( preceding::film/titre = titre)]">
  <xsl:variable name="titre_du_film" select="translate(titre, translate(titre, $allowedChars, ''), '')" />
  <xsl:result-document method="html" href="film_{$titre_du_film}.html" >
    <html>
      <head>
        <xsl:call-template name="entete_html"/>
      </head>
      <body>
        <xsl:call-template name="menu" />
        <xsl:apply-templates select="." mode="detail"/>
        <xsl:call-template name="pied_de_page" />
      </body>
    </html>
  </xsl:result-document>
</xsl:for-each>
```

Ce template initialize la génération du document *film_{titre_du_film}.html* pour chaque film, en construisant la structure de base et en appelant le template d'affichage détaillé qui correspond aux films. Il écrit ensuite le résultat dans le document *film_{titre_du_film}.html*.

On notera que nous nous sommes assuré que chaque film ne donne lieu qu'à un seul fichier.

Le template d'affichage détaillé d'un film se décompose en plusieurs partie :

Affichage de la photo

```
<xsl:if test="photo">
  <span>
    <xsl:variable name="photo_du_film" select="photo" />
    <img alt="photo_du_film" src="resources/{$photo_du_film}"/>
  </span>
</xsl:if>
```

Affichage des informtions générales

```
<span>
  <xsl:if test="duree">
    <p>
      Duree
      <strong>
        <xsl:if test="not(duree &lt; 60)"><xsl:value-of select="floor(duree div 60)"/>h <xsl:value-of select="duree mod 60"/>min</xsl:if>
        <xsl:if test="duree &lt; 60"><xsl:value-of select="duree"/>min</xsl:if>
        </strong>
      </p>
  </xsl:if>
  <xsl:if test=".//acteur">
    <p>
      Avec
      <strong>
        <xsl:apply-templates select=".//role">
          <xsl:sort order="ascending" select="place" data-type="number" />
          <xsl:sort order="ascending" select=".//nom" />
        </xsl:apply-templates>
        <xsl:if test="count(.//acteur) &gt; 3"> <a href="#acteurs">Plus</a></xsl:if>
      </strong>
    </p>
  </xsl:if>
  <xsl:if test=".//genre">
    <p>
      Genres
      <strong>
        <xsl:for-each select=".//genre">
          <xsl:value-of select="."/>
          <xsl:if test="position() &lt; count(..//genre)">, </xsl:if>
        </xsl:for-each>
      </strong>
    </p>
  </xsl:if>
  <xsl:if test=".//langage">
    <p>
      Langues
      <strong>
        <xsl:for-each select=".//langage">
          <xsl:value-of select="."/>
          <xsl:if test="position() &lt; count(..//langage)">, </xsl:if>
        </xsl:for-each>
      </strong>
    </p>
  </xsl:if>
</span>
```

On notera que l'on teste la présence des champs avant de les afficher. La durée est transformée sous la forme hh:mm. De plus, on affiche les acteurs des trois premiers rôles. Pour ce faire, on appelle le template ci-dessous qui n'affiche que les trois premiers rôles.

```
<!-- Affichage des 3 premiers rôles du film -->
<xsl:template match="role">
  <xsl:if test="not(position() > 3)">
    <xsl:value-of select="replace(.//nom, ',', '')"/>
    <xsl:if test="position() &lt; count(..//acteur)">, </xsl:if>
  </xsl:if>
</xsl:template>
```

Affichage du synopsis

```
<xsl:if test="synopsis">
  <div>
    <h2>Synopsis</h2>
    <p><xsl:value-of select="synopsis"/></p>
  </div>
</xsl:if>
```

Affichage des projections liées à ce film

```
<!-- Projections du film -->
<h2>Projections</h2>
<ul>
  <xsl:variable name="titre_du_film" select="titre" />
  <xsl:for-each select="//projection[film[titre = $titre_du_film]]">
    <xsl:sort order="ascending" select="date"/>
    <li>
      <b>
        <strong><xsl:value-of select="substring(date,9,2)"/>.<xsl:value-of select="substring(date,6,2)"/>.<xsl:value-of select="substring(date,1,4)"/>
        <xsl:text> </xsl:text><xsl:value-of select="substring(date,12,2)"/>:<xsl:copy-of select="substring(date,15,2)"/></strong>,
        <xsl:value-of select="noSalle"/>
      </b>
      <br/>
      <br/>
    </li>
  </xsl:for-each>
</ul>
```

Affichage de tous les acteurs ayant joué dans ce film

```
<!-- acteurs du film -->
<xsl:if test=".//acteur">
  <a name="acteurs"/>
  <h2>Acteurs</h2>
  <ul class="list">
    <li class="listItemLabel">
      <span class="listItemElement name">Nom</span>
      <span class="listItemElement birth">Date de naissance</span>
      <span class="listItemElement sex">Sexe</span>
      <span class="listItemElement play">Personnage</span>
    </li>
    <xsl:for-each select=".//acteur">
      <xsl:sort order="ascending" select="../place" data-type="number" />
      <xsl:sort order="ascending" select="nom" data-type="number" />
      <li class="listItem">
        <xsl:apply-templates select="." mode="tableau"/>
        <span class="listItemElement play">
          <xsl:if test="../personnage"><xsl:value-of select="../personnage"/></xsl:if>
          <xsl:if test="not(../personnage)">-</xsl:if>
        </span>
      </li>
    </xsl:for-each>
  </ul>
</xsl:if>
```

Ce template affiche les acteurs avec le personnage qu'ils ont joué. Les acteurs sont triés par numéro de rôle.

Affichage des critiques du film

```
<!-- critiques du film -->
<xsl:if test=".//critiques">
  <div>
    <h2>Critiques</h2>
    <xsl:if test=".//note">
      <h3>Note moyenne <strong><xsl:value-of select="avg(.//note)" /> / 5</strong></h3>
    </xsl:if>
    <xsl:for-each select=".//critique/texte">
      <xsl:if test="../note">
        <h4>Note <strong><xsl:value-of select="../note" /></strong></h4>
      </xsl:if>
      <p><xsl:value-of select="." /></p>
    </xsl:for-each>
  </div>
</xsl:if>
```

Affichage des mots-clés liés au film

```
<!-- Mots clés -->
<xsl:if test=".//motCle">
  <div>
    <h2>Mots-clés</h2>
    <p class="keywords">
      <xsl:for-each select=".//motCle">
        <xsl:value-of select="."/>
        <xsl:if test="position() &lt; count(..//motCle)">, </xsl:if>
      </xsl:for-each>
    </p>
  </div>
</xsl:if>
```

2.4 ACTEURS

2.4.1 liste_des_acteurs.html

```
<!-- Generation du fichier liste_des_acteurs.html -->
<xsl:result-document method="html" href="liste_des_acteurs.html" >
  <html>
    <head>
      <xsl:call-template name="entete_html"/>
    </head>
    <body>
      <xsl:call-template name="menu" />
      <xsl:call-template name="liste_des_acteurs"/>
      <xsl:call-template name="pied_de_page" />
    </body>
  </html>
</xsl:result-document>
```

Ce template initialize la génération du document *liste_des_acteurs.html*, en construisant la structure de base et en appelant le template *liste_des_acteurs*. Il écrit ensuite le résultat dans le document *liste_des_acteurs.html*.

```
<!-- Liste de tous les acteurs des films projetés -->
<xsl:template name="liste_des_acteurs">
  <h1>Liste des acteurs</h1>
  <ul class="list">
    <li class="listItemLabel">
      <span class="listItemElement name">Nom</span>
      <span class="listItemElement birth">Date de naissance</span>
      <span class="listItemElement sex">Sexe</span>
    </li>
    <xsl:for-each select=".//acteur[not( preceding::acteur/nom = nom)]">
      <xsl:sort order="ascending" select="nom"/>
      <li class="listItem">
        <xsl:apply-templates select="." mode="tableau"/>
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>
```

Ce template affiche les acteurs en les triant par nom. A chaque acteur, on applique le template chargé d'afficher son nom et sa date de naissance et son sexe, décrit ci-dessous.

On notera la présence de l'instruction de sélection //acteur[not(preceding::acteur/nom = nom)]. Celle-ci permet de supprimer les doublons.

```
<!-- Ligne du tableau des acteurs -->
<xsl:template match="acteur" mode="tableau">
  <xsl:variable name="nom_acteur" select="translate(nom, translate(nom, $allowedChars, ''), '')" />
  <span class="listItemElement name">
    <a href="acteur_{$nom_acteur}.html">
    <xsl:value-of select="nom"/>
  </a>
  </span>
  <span class="listItemElement birth">
    <xsl:if test="dateNaissance">
      <xsl:value-of select="substring(dateNaissance,9,2)"/>.<xsl:value-of select="substring(dateNaissance,6,2)"/>.<xsl:value-of select="substring(dateNaissance,1,4)"/>
    </xsl:if>
    <xsl:if test="not(dateNaissance)">-</xsl:if>
  </span>
  <span class="listItemElement sex">
    <xsl:if test="sexe"><xsl:value-of select="sexe/@valeur"/></xsl:if>
    <xsl:if test="not(sexe)">-</xsl:if>
  </span>
</xsl:template>
```

Ce template se charge d'afficher un résumé de l'acteur. On constate que la valeur de l'attribut du sexe est bien récupérée.

2.4.2 acteur_{*nom_acteur*}.html

```
<!-- Generation des fichiers des acteurs -->
<xsl:for-each select="//acteur[not( preceding::acteur/nom = nom)]">
  <xsl:variable name="nom_acteur" select="translate(nom, translate(nom, $allowedChars, ''), '')" />
  <xsl:result-document method="html" href="acteur_{$nom_acteur}.html" >
    <html>
      <head>
        <xsl:call-template name="entete_html"/>
      </head>
      <body>
        <xsl:call-template name="menu" />
        <xsl:apply-templates select="." mode="detail"/>
        <xsl:call-template name="pied_de_page" />
      </body>
    </html>
  </xsl:result-document>
</xsl:for-each>
```

Ce template initialize la génération du document *acteur_{nom_acteur}.html* pour chaque acteur, en construisant la structure de base et en appelant le template d'affichage détaillé qui correspond aux acteurs. Il écrit ensuite le résultat dans le document *acteur_{nom_acteur}.html*.

On notera que nous nous sommes assuré que chaque acteur ne donne lieu qu'à un seul fichier.

Affichage des informations générales de l'acteur

```
<h1><xsl:value-of select="nom"/></h1>
<table cellspacing="10">
  <tbody>
    <xsl:if test="nomNaissance">
      <tr>
        <td>Nom de naissance</td><td><strong><xsl:value-of select="nomNaissance"/></strong></td>
      </tr>
    </xsl:if>
    <xsl:if test="sexe">
      <tr>
      <td>Sexe</td><td><strong><xsl:value-of select="sexe/@valeur"/></strong></td>
      </tr>
    </xsl:if>
    <xsl:if test="dateNaissance">
      <tr>
        <td>Date de naissance</td>
        <td>
          <strong>
            <xsl:value-of select="substring(dateNaissance,9,2)"/>.<xsl:value-of select="substring(dateNaissance,6,2)"/>.<xsl:value-of select="substring(dateNaissance,1,4)"/>
          </strong>
        </td>
      </tr>
    </xsl:if>
    <xsl:if test="dateDeces">
      <tr>
        <td>Date de décès</td>
        <td>
          <strong>
            <xsl:value-of select="substring(dateDeces,9,2)"/>.<xsl:value-of select="substring(dateDeces,6,2)"/>.<xsl:value-of select="substring(dateDeces,1,4)"/>
          </strong>
        </td>
      </tr>
    </xsl:if>
  </tbody>
</table>
<xsl:if test="biographie">
  <div>
    <h2>Biographie</h2>
    <p><xsl:value-of select="biographie"/></p>
  </div>
</xsl:if>
```

Ici seules les informations disponibles sont affichées. Ainis, s'il n'y a pas de date de décès, elle ne sera pas affichée.

Affichage des films dans lesquels l'acteur a joué

```
<!-- Films de l'acteur à l'affiche -->
<h2>Films à l'affiche</h2>
<ul>
  <xsl:variable name="nom_acteur" select="nom" />
  <xsl:for-each select="//film[not( preceding::film/titre = titre)][.//acteur[nom = $nom_acteur]]">
    <xsl:sort order="ascending" select="titre"/>
    <xsl:variable name="titre_du_film" select="titre" />
    <li>
      <b>
        <a href="film_{translate(titre, translate(titre, $allowedChars, ''), '')}.html">
          <xsl:value-of select="titre"/>
        </a>
        <xsl:if test="//film[not( preceding::film/titre = titre)][titre = $titre_du_film]//role[.//acteur[nom = $nom_acteur]]/personnage">
          <xsl:text>, </xsl:text><xsl:value-of select="//film[not( preceding::film/titre = titre)][titre = $titre_du_film]//role[.//acteur[nom = $nom_acteur]]/personnage"/>
        </xsl:if>
      </b>
      <br/>
      <br/>
    </li>
  </xsl:for-each>
</ul>
```

Ici, de nouveau, nous avons tenu à supprimer les doublons.

3 RÉSULTAT

3.1 PROJECTIONS

3.1.1 Par titre de film

Projections par titre de film Projections par date Projections par classement

Projections par titre de film

Film	Date	Salle	Classement
Casino Royale	01.06.2016 08:20	Flon 1	5 / 5
Django Unchained	27.04.2016 11:52	Flon 1	-
GoldenEye	27.06.2016 10:15	Flon 3	1 / 5
 Ocean's Eleven	10.05.2016 03:30	Flon 2	5 / 5
 Ocean's Eleven	25.06.2016 04:45	Flon 3	5 / 5
Ocean's Twelve	18.06.2016 01:30	Flon 3	-

Liste des films Liste des acteurs

3.1.2 Par date

Projections par titre de film Projections par date Projections par classement

Projections par date

Film	Date	Salle	Classement
Django Unchained	27.04.2016 11:52	Flon 1	-
 Ocean's Eleven	10.05.2016 03:30	Flon 2	5 / 5
Casino Royale	01.06.2016 08:20	Flon 1	5 / 5
Ocean's Twelve	18.06.2016 01:30	Flon 3	-
 Ocean's Eleven	25.06.2016 04:45	Flon 3	5 / 5
GoldenEye	27.06.2016 10:15	Flon 3	1 / 5

Liste des films Liste des acteurs

3.1.3 Par classement

Projections par titre de film Projections par date Projections par classement

Projections par classement

Film	Date	Salle	Classement
Casino Royale	01.06.2016 08:20	Flon 1	5 / 5
 Ocean's Eleven	10.05.2016 03:30	Flon 2	5 / 5
 Ocean's Eleven	25.06.2016 04:45	Flon 3	5 / 5
GoldenEye	27.06.2016 10:15	Flon 3	1 / 5
Django Unchained	27.04.2016 11:52	Flon 1	-
Ocean's Twelve	18.06.2016 01:30	Flon 3	-

Liste des films Liste des acteurs

3.2 FILMS

3.2.1 Liste des films

Liste des films

- Casino Royale
- Django Unchained
- GoldenEye
- 
  Ocean's Eleven
- Ocean's Twelve

Liste des films Liste des acteurs

3.2.2 Fiche détaillée d'un film

Projections par titre de film Projections par date Projections par classement

Ocean's Eleven

Duree **1h 56min**

Avec **Clooney George, Birt Cecelia Ann, Nolan Paul L., Plus**

Genres **Thriller, Crime**

Langues **Italian, English, Cantonese**

Synopsis

When Daniel Ocean is released from prison in New Jersey, his next heist is already planned. Danny's target are three Las Vegas casinos: The Bellagio, the Mirage and the MGM Grand. They all belong to ruthless entrepreneur Terry Benedict, who, by the way, also shows a certain interest in Danny's beautiful ex-wife Tess. During a much-anticipated boxing event (Lennox Lewis vs. Wladimir Klitschko), there will be $150 million in the safe, 70 yards below the strip. So, Danny starts to hire professionals from all over the country: There's the card magician Rusty Ryan, the perfect pickpocket Linus Caldwell and the ingenious pyrotechnician Basher Tarr. Reuben Tishkoff, who lost a casino to Benedict, provides funding, the brothers Virgil and Turk Malloy will drive and help, and Frank Catton, a professional card dealer, gets a job at the casino to watch the routines. Saul Bloom, already retired, will play the rich heavy weaponry dealer and live in the hotel, while Livingston Dell bugs the place to have a look over the shoulders of the security personnel. Finally, the chinese acrobat artist Yen will be the one to move inside the safe before the motion detectors are turned off. There are three rules to be followed: First: no blood. Second: Rob only who deserves it. Third: Do it as if you have nothing to lose. When the day of the boxing event finally draws near, all is set, and Benedict doesn't have a clue - or does he?

Projections

- **10.05.2016 03:30, Flon 2**
- **25.06.2016 04:45, Flon 3**

Acteurs

Nom	Date de naissance	Sexe	Personnage
Clooney, George	06.05.1961	MASCULIN	Danny Ocean
Birt, Cecelia Ann	-	FEMININ	Board Member #1
Nolan, Paul L.	09.06.1954	MASCULIN	Board Member #2
Florence, Carol	-	FEMININ	Board Member #3
Galinski, Lori	-	FEMININ	Blackjack Dealer
Mac, Bernie	05.10.1957	MASCULIN	Frank Catton
Pitt, Brad	18.12.1963	MASCULIN	Rusty Ryan
Gantt, Mark	10.12.1968	MASCULIN	Bartender
Perez, Timothy Paul	-	MASCULIN	Security Guard
Gould, Elliott	29.08.1938	MASCULIN	Reuben Tishkoff
Patton, Frank	-	MASCULIN	Lockbox Carrier
Affleck, Casey	12.08.1975	MASCULIN	Virgil Malloy
Caan, Scott	23.08.1976	MASCULIN	Turk Malloy
Jemison, Eddie	-	MASCULIN	Livingston Dell
Hernandez, Jorge R.	-	MASCULIN	FBI Man #1
Snay, Tim	-	MASCULIN	FBI Man #2
Pérez, Miguel	07.09.1957	MASCULIN	Explosives Cop
Qin, Shaobo	01.01.1982	MASCULIN	Yen
Reiner, Carl	20.03.1922	MASCULIN	Saul Bloom
Lewis, Lennox	02.09.1965	MASCULIN	Boxing Opponent
Klitschko, Wladimir	25.03.1976	MASCULIN	Boxing Opponent
Damon, Matt	08.10.1970	MASCULIN	Linus Caldwell

Brandt, Barry	-	MASCULIN	Technician #1
Johnson, William	-	MASCULIN	Technician #2
Peters, Robert	20.07.1961	MASCULIN	Eye-in-the-Sky Technician #1
Jensen, David	23.09.1952	MASCULIN	Eye-in-the-Sky Technician #2
Adkins, Kelly	-	FEMININ	Dancer
Stenson, Gregory	-	MASCULIN	Sentry
La Due, Joe	-	MASCULIN	Billy Tim Denham
Garcia, Andy	12.04.1956	MASCULIN	Terry Benedict
Fiore, John C.	-	MASCULIN	Hotel Security
Kordick, Tommy	-	MASCULIN	Hotel Security
DeLano, Michael	26.11.1940	MASCULIN	Casino Manager
La Russa, Charles	-	MASCULIN	Italian High Roller
Allison, Anthony	-	MASCULIN	French High Roller
Soeda, Ronn	-	MASCULIN	Japanese High Roller
Roberts, Julia	28.10.1967	FEMININ	Tess Ocean
Sachs, Robin	05.02.1951	MASCULIN	Seller
Manoux, J.P.	08.06.1969	MASCULIN	Aide-de-Camp
Weintraub, Jerry	26.09.1937	MASCULIN	High Roller
Allison, Frankie J.	-	MASCULIN	High Roller Pit Boss
Curatola, James	-	MASCULIN	Baccarat Dealer
Silva, Henry	15.09.1928	MASCULIN	Boxing Spectator
Gormé, Eydie	16.08.1928	FEMININ	Boxing Spectator
Dickinson, Angie	30.09.1931	FEMININ	Boxing Spectator
Lawrence, Steve	08.07.1935	MASCULIN	Boxing Spectator
Newton, Wayne	03.04.1942	MASCULIN	Boxing Spectator
Newton, Wayne	03.04.1942	MASCULIN	Boxing Spectator
Fischbacher, Siegfried	13.06.1939	MASCULIN	Boxing Spectator
Horn, Roy	03.10.1944	MASCULIN	Boxing Spectator
Lampley, Jim	08.04.1949	MASCULIN	Boxing Spectator
Merchant, Larry	11.02.1931	MASCULIN	Boxing Spectator
Reed, Richard	-	MASCULIN	Bucky Buchanan
Sontag, David	-	MASCULIN	Plainclothes Goon #1
Sontag, Larry	-	MASCULIN	Plainclothes Goon #2
Allison, Bill	-	MASCULIN	Guard
Meyers, Rusty	01.10.1957	MASCULIN	Security Officer #1
Coyle, Joe	-	MASCULIN	Security Officer #2
Schwartz, Scott L.	01.01.0016	MASCULIN	Bulldog, the Bruiser
Steele, Richard	-	MASCULIN	Boxing Referee
Robotham, John	-	MASCULIN	Uzi-Carrying Guard #1
Ward, Vincent M.	-	MASCULIN	Uzi-Carrying Guard #2
Beringer, Scott	-	MASCULIN	Head Goon
Alfonso, Jim	-	MASCULIN	Police Officer
Bom, Bonnie	-	MASCULIN	Dog Track Gambler

Critiques

Note moyenne 5 / 5

Note 5

Un de mes films préférés. "Ocean's Eleven", c'est d'abord un casting comme on en voit rarement au cinéma, c'est aussi une intrigue qui tient en haleine pendant tout le film et une réalisation impeccable de Steven Soderbergh. On ne s'ennuie pas une seule seconde, ce film est une perle, une oeuvre comme on en voit rarement au cinéma. Il y a aussi la musique qui colle parfaitement au cadre pendant tout le long du film. Je conseil vraiment ce film, que ce soit aux amateurs ou aux vrais cinéphiles. "Ocean's Eleven" est un film avec, un scénario parfait, qui est très drôle et très rafraîchissant, un film que l'on peut regarder à n'importe quel moment de la journée, que ce soit le soir ou en plein après-midi. Franchement, c'est un film qu'il faut voir au moins une fois.

Mots-clés

shot-to-death, confidence-man, money-falling-through-the-air, salt-lake-city-utah, writing-on-hand, gambling, split-screen, train, foiled-robbery, stripper, soft-focus, forced-perspective, arrest, brother-brother-relationship, 2000s, punched-in-the-stomach, altered-version-of-studio-logo, poker-game, actor-playing-himself, airport, release-from-prison, false-name, robbery, career-criminal, car-dealer, fountain, movie-actor, raised-middle-finger, reference-to-vermeer, cameo-appearance, heist, reference-to-ted-nugent, pickpocket, master-thief, cigar-smoking, pistol, remote-control-car, flashback, cockney-accent, reference-to-evel-knievel, con-artist, bomb, strip-club, tuxedo, chicago, exploding-car, voice-over, los-angeles-california, ulcer, criminal-mastermind, crossword-puzzle, male-bonding, first-of-trilogy, ex-con, explosive, explosion, gentleman-crooks, boxing-match, shot-in-the-back, caper, prison, explosives-expert, el-train, casino-owner, deception, covered-in-feces, no-opening-credits, reference-to-claude-monet, faked-death, security-guard, ex-husband-ex-wife-relationship, vault, cockney-rhyming-slang, director-also-cinematographer, chicago-illinois, acrobat, beating, poker, racism, freeze-frame, blockbuster, recruiting, fbi-agent, crushed-hand, van, sneaking-something-into-someone's-pocket, briefcase-chained-to-wrist, divorce, las-vegas-nevada, surveillance-camera, art-dealer, security-camera, number-in-title, video-surveillance, impersonating-a-police-officer, sewer, lying, cell-phone, blueprint, chinese-american, ex-wife, handshake, con-man, exploding-building, long-con, mormon, character-repeating-someone-else's-dialogue, st.-petersburg-florida, reference-to-Édouard-manet, bandaged-hand, house-of-cards, black-and-white-scene, pretending-to-be-rich, power-failure, swat-team, reference-to-leon-spinks, nightclub, unsubtitled-foreign-language, electro-magnetic-pulse, ex-convict, fake-accent, punched-in-the-face, elevator-shaft, san-diego-california, wisecrack-humor, character-name-in-title, black-out, reference-to-jim-brown, atlantic-city-new-jersey, gentleman-thief, ensemble-cast, balloon, remake, caper-comedy, swish-pan, reference-to-ella-fitzgerald, bare-chested-male, hollywood-california, 911, british, thief, financier, posing-as-a-doctor, circus-performer, surprise-ending, casino, curator, dog-track, wedding-ring, convicted-felon, planning, card-dealer, motion-sensor, blackout, convertible

3.3 ACTEURS

3.3.1 Liste des acteurs

Projections par titre de film Projections par date Projections par classement

Liste des acteurs

Nom	Date de naissance	Sexe
Abkarian, Simon	05.03.1962	MASCULIN
Ade	-	MASCULIN
Adkins, Kelly	-	FEMININ
Affleck, Casey	12.08.1975	MASCULIN
Alfonso, Jim	-	MASCULIN
Allen, Todd	09.12.1960	MASCULIN
Allison, Anthony	-	MASCULIN
Allison, Bill	-	MASCULIN
Allison, Frankie J.	-	MASCULIN
Ambrosio, Alessandra	11.04.1981	FEMININ
Anden, Mini	07.06.1978	FEMININ

3.3.2 Fiche détaillée d'un acteur

Projections par titre de film Projections par date Projections par classement

Mac, Bernie

Nom de naissance	McCullough, Bernard Jeffrey
Sexe	MASCULIN
Date de naissance	05.10.1957
Date de décès	09.08.2008

Biographie

Bernard Jeffrey McCollough was born in Chicago in 1957. He grew up in Chicago, in a rougher neighborhood than most others, with a large family living under one roof. This situation provided him with a great insight into his comedy, as his family, and the situations surrounding them would be what dominated his comedy. Mac worked in the Regal Theater, and performed in Chicago parks in his younger days. He became a professional comedian in 1977, at the age of 19. He refused to change his image for television and films, and therefore was not very well known for most of the eighties. In 1992 he made his film debut with a small part with _Mo' Money (1992)_ (qv). This started a plethora of small parts in a string of movies, mostly comedies, including _Who's the Man? (1993)_ (qv), _House Party 3 (1994)_ (qv) and _The Walking Dead (1995)_ (qv). 1995 proved to be a turning point in his career. He did an HBO Special called _"Midnight Mac" (1995)_ (qv), and took a part as Pastor Clever in the Chris Tucker comedy _Friday (1995)_ (qv). Bernie Mac developed a cult following due to the movie and had many small parts since. In 1996 he starred in the memorable 'Spike Lee' (qv) movie _Get on the Bus (1996)_ (qv), and was very funny in _Don't Be a Menace to South Central While Drinking Your Juice in the Hood (1996)_ (qv). About this time he had a recurring role in the TV series _"Moesha" (1996)_ (qv). Bernie Mac's star was slowly rising from this point. His next couple of movie parts were more substantial, including _How to Be a Player (1997)_ (qv) and _The Players Club (1998)_ (qv). In 1999 Bernie Mac got his most high profile part up to that point in the film _Life (1999/I)_ (qv) starring 'Eddie Murphy (I)' (qv). The new century started a new era for the brash Chicago comedian. He was a featured comedian in _The Original Kings of Comedy (2000)_ (qv). This performance made him more of a household name, and led to many more major parts. In 2001 he played Martin Lawrence's uncle in _What's the Worst That Could Happen? (2001)_ (qv) and later that year, was in the star studded remake of _Ocean's Eleven (2001)_ (qv). However his biggest success was _"The Bernie Mac Show" (2001)_ (qv), which debuted in 2001 to instant acclaim. However, soon after the series ended, Mac's health took a turn for the worse. He developed sarcoidosis, an autoimmune disease which causes inflammation in the lungs. On August 9, 2008, after weeks of unsuccessful treatments, Bernie Mac died at Northwestern Memorial Hospital in Chicago. He was 50. Bernie Mac was a comedian that refused to change his image for Hollywood and said that his life in Chicago was who he was and there was nothing that could change that. He was a mature comedian who was very intelligent and very engaging in his TV, movie and stand-up appearances.

Films à l'affiche

- Ocean's Eleven, Frank Catton
- Ocean's Twelve, Frank Catton

Liste des films Liste des acteurs

4 MODE D'EMPLOI

Pour généré les documents html à partir du document xml en utilisant une feuille de style xsl, il est possible d'utiliser Java Saxon.

Pour ce faire, en utilisant le terminal depuis le dossier contenant les fichier sources, lancer la commande :

Java -cp saxon9he.jar net.sf.saxon.Transform -s:projections.xml -xsl:projections.xsl -o:projections.html

Il existe également des logiciels tels qu'*Altova XML Spy* qui permettent d'effctuer la génération des dichiers en toute simplicité.

5 CONCLUSION

D'après le travail que nous avons réussi à réaliser, nous pouvons dire que l'utilisation du XSL permet effectivement de transformer des données stockées dans un document XML en un réel site web. Nous avons pu parcourir le document xml de différentes manières et il nous a même été possible de supprimer les éventuels doublons. Nous noterons toutefois un désavantage de l'utilisation du XSL pour la création d'un site web. En effet, les fichiers sont créés statiquement et donc chaque modification nécessiterait de régénérer la totalité des fichiers. En réalité, nous trouvons qu'il serait peu imaginable de devoir effectuer cette action sur un site qui doit régulièrement effectuer des mises à jour de données, tel que c'est le cas pour ce laboratoire.

Ce laboratoire nous a permis de nous familiariser avec l'utilisation de feuilles de style XSL pour la transformation de données. Plus précisément, nous avons pu améliorer notre maîtrise de ces feuilles en matière d'utilisation de templates. Nous avons ainsi pu factoriser certaines parties de code. Nous avons en outre pu remarquer la puissance du XSL, notamment au niveau de la génération dynamique de fichiers.