|  |  |
| --- | --- |
|  | **2011** |
|  | Leonhard hermle  Julien Cellier  Xavier Mourgues |

|  |
| --- |
| **[Rapport Projet tuteurè]** |
| Rapport du projet tuteuré XML exprimant les besoins, les objectifs et les moyens de mise en œuvre du projet ainsi que les problèmes rencontrés et le résultat atteint en fin de projet. |

Université Toulouse I – IUT de Rodez

Année 2010/2011

TABLE DES MATIERES

[REMERCIEMENTS 3](#_Toc287359446)

[Introduction 4](#_Toc287359447)

[I. Présentation du sujet 5](#_Toc287359448)

[1. Les besoins 6](#_Toc287359449)

[2. Les contraintes 6](#_Toc287359450)

[3. L’univers de travail 6](#_Toc287359451)

[4. Répartition des tâches 6](#_Toc287359452)

[II. Java 8](#_Toc287359453)

[Problèmes rencontrés 8](#_Toc287359454)

[III. Base de données 10](#_Toc287359455)

[Problèmes rencontrés 10](#_Toc287359456)

[IV. Restitution des données 11](#_Toc287359457)

[1. Aspect graphique 11](#_Toc287359458)

[Police de caractère 11](#_Toc287359459)

[Les couleurs 11](#_Toc287359460)

[Les liens 11](#_Toc287359461)

[La tailles des éléments 12](#_Toc287359462)

[Le menu 12](#_Toc287359463)

[Le champ de recherche 12](#_Toc287359464)

[La pagination 12](#_Toc287359465)

[2. Aspect technique et structure de l’interface 13](#_Toc287359466)

[Le modèle MVC 13](#_Toc287359467)

[Schéma de navigation 13](#_Toc287359468)

[Le fichier conf.ini 14](#_Toc287359469)

[La pagination 15](#_Toc287359470)

[L’Ajax 16](#_Toc287359471)

[Extraction XML 16](#_Toc287359472)

[Exécution d’une application java via PHP. 16](#_Toc287359473)

[Conclusion 17](#_Toc287359474)

REMERCIEMENTS

Nous souhaitons tout d’abord adresser nos remerciements à ceux qui nous ont permis de réaliser ce projet tuteuré, à ceux qui nous ont encadrés et qui nous ont rendus cette expérience agréable et enrichissante.

Nous remercions PATRICK SABI, notre professeur qui nous à encadré tout au long de notre projet et de nous avoir apporté des conseils.

Merci également à BRUNO BELIERE d’avoir mis à disposition deux bases de données Oracle qui nous ont permis de mener à bien notre projet.

Un grand merci à ALLOCINE qui autorise le parcours des pages HTML de son site afin de permettre la récupération de données textuelles.

Introduction

Dans le cadre de nos projets tuteurés de l’année 2010/2011, notre groupe à choisi de se lancer dans la réalisation d’une application permettant d’exploiter du contenu de pages HTML utilisant du format XML dans une base de donnée afin de pouvoir les stocker et les restituer pour que l’utilisateur puisse jouir des différentes informations dans une même interface graphique. Ainsi nous avons dû faire face à certains choix plus ou moins difficiles à débattre au sain du groupe.

Certaines questions évidentes se posent tel que, quels ont été ces choix ? Pourquoi ces choix ?

Nous établirons une brève présentation de ce projet mettant en avant les choix effectué, nos besoins, les contraintes découlant des décisions prises pour aboutir sur les problèmes rencontrés lors de la période de projet. Enfin, nous conclurons pour faire le bilan de ce dernier.

1. Présentation du sujet

Après mûre réflexion nous nous sommes orientés vers l’exploitation de sites cinématographiques tels qu’Allociné ou IMDB afin de réutiliser des données concernant les films, tel que les avis ou notes attribués par les internautes. D’autre choix auraient été possible comme par exemple l’exploitation de sites météorologiques (restitution de statistique pour une période donnée). Ce dernier choix figurants déjà dans un projet des années précédentes, nous avons préféré partir sur de nouvelles bases en choisissant un nouveau thème, le cinéma.

L’objectif du projet est donc de restituer à partir de sites web existants des données de films (critiques, commentaires, notes, bande annonce, informations diverses…). Ainsi, les sites exploités seront ceux comportant des informations diverses pour un même élément recherché (un film en particulier). Les commentaires et les notes d’utilisateurs et des sites seront rassemblés permettant une vue d’ensemble sur le film recherché, tout ceci pouvant servir à comparer et fusionner les avis d’un film donné.

Plus simplement, il sera possible de regrouper les commentaires et notes restitués via une interface graphique développé en langage web, notamment en PHP. Cette partie du projet sera expliquée plus en détails dans le chapitre « Restitution de données ». Le parcours (parsing) des pages des sites web se fera en java.

Le projet se présente donc en 3 étapes :

* La **récupération de données** à travers divers sites web évolutif et dynamique (plus en détails dans le chapitre « JAVA »).
* Le **stockage de ces données** dans une base de données Oracle v10.2 (grâce à JAVA et plus en détails dans le chapitre «Base de données »).
* La **restitution visuelle des données** pour les utilisateurs (plus en détails dans le chapitre «Restitution de données »).

A l’origine, les sites web que nous avons sélectionnés étaient Allociné et IMDB. Malheureusement, se posait un souci d’autorisation car le site en question bloque les robots qui parcourent ses pages. Ce problème est mis en avant dans le chapitre des problèmes en fin de rapport sous «IMDB». Par conséquent, le choix d’un second site autre qu’IMDB contenant une quantité importante de données de films, non négligeables. IMDB et l’une plus grande base de données de films mondiale rassemblant des informations sur tous les films disponibles, c’est pourquoi l’impossibilité d’exploitation de leur site internet est très regrettable étant donnée que la comparaison de données entre une sites de référence comme IMDB et un second peu montrer des différences importantes.

Nous aurons une première phase de découverte visant à étudier les différentes techniques de mise en œuvre d’un tel projet.

Plus tard, dans un souci d’évolution, plus d’information pourront être récupérer et d’autres sites dit «source d’information» pourront être rajoutés. En effet, dans la mesure du possible un site pourra être ajouté afin de pouvoir être exploité. La démarche à suivre sera indiquée à l’utilisateur. Cette option étant assez complexe à réaliser, d’autant plus que les délais de finalisation du projet sont serrés, ne pourra peut-être pas être créée.

## Les besoins

Nous avons besoin de récupérer des données de sites web qui ne sont pas structuré en XML mais en HTML ce qui nous posera problème lorsque voudront traiter ces données. En effet, un site étant évolutif il est possible que sa structure HTML change. Pour parer cet effet, nous devrons réussir à soutirer uniquement les informations nécessaires et non les informations inutiles (publicités …). Pour cela l’utilisation de la technologie DOM sera indispensable.

Dans un deuxième temps, nous aurions besoin d’une machine avec un accès à une base de données Oracle ainsi que qu’un accès libre a Internet (sans proxy bloquant les sites tel qu’Allocine).

## Les contraintes

L’une des premières contraintes que nous avons est l’utilisation d’Oracle v10.2.

D’autre part l’extraction des données se fera par des primitives XSLT mais la structure de données sera en HTML ce qui nous forcera à analyser syntaxiquement ces informations afin d’avoir ce qui se rapproche le plus d’une structure XML.

## L’univers de travail

Nous avons décidé que pour la récupération des données : nous utiliserons un univers java afin de profiter de sa souplesse d’utilisation de ressources tiers tel que le SGBD Oracle via JDBC, le logiciel JTIDY et les primitives XSLT.

Le stockage des données sera partiellement inséré sous format XML et stocké dans une base Oracle.

Pour finir la restitution des données se fera via une PUI en PHP/HTML.

## Répartition des tâches

Nous avons décidé de séparer le projet en 3 parties au sein du groupe.

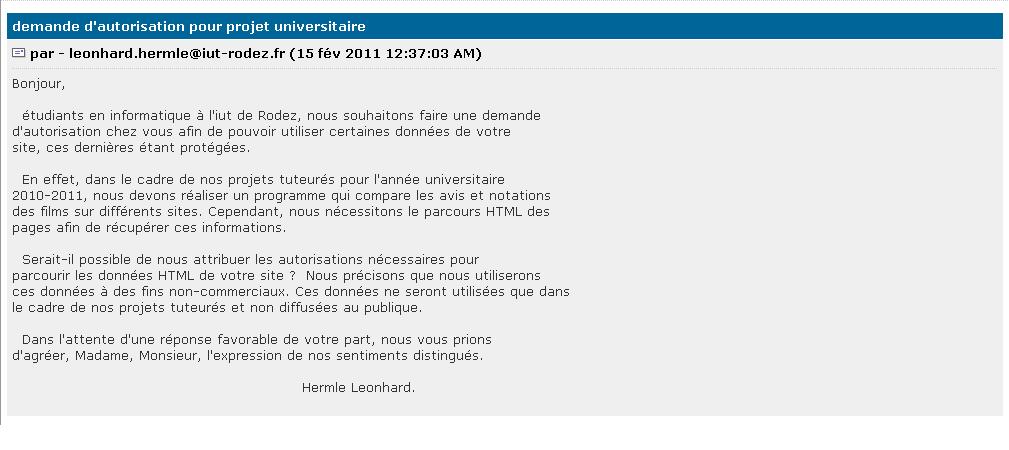
* M. Leonhard Hermle prendra à sa charge la PUI en PHP/HTML
* M. Cellier Julien s’occupera du stockage en base de données et XML avec interfaçage pour le PHP et le Java.
* M. Mourgues Xavier traitera la récupération des données en Java.

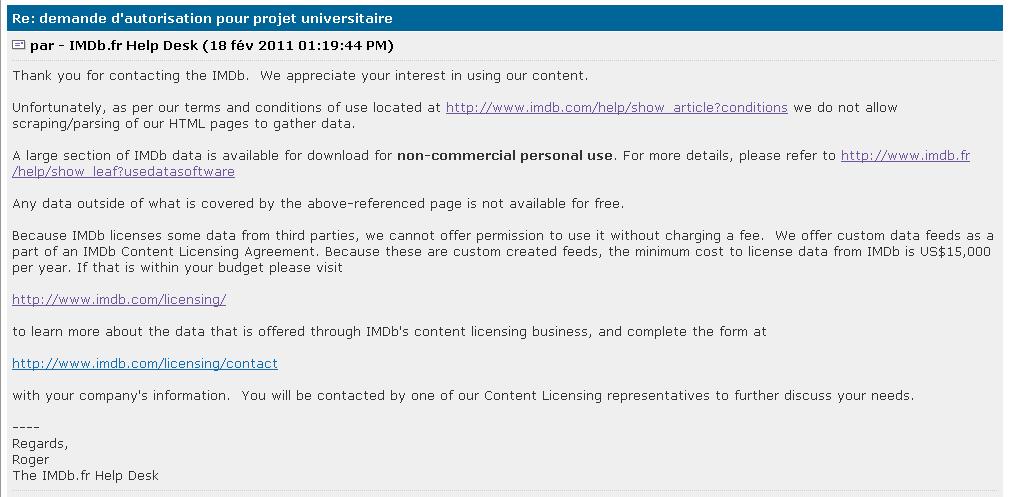
1. Java

Jtidy, aspect technique

## Problèmes rencontrés

**IMDB** : Le site web IMDB refuse strictement le parcours des pages par un robot grâce à la protection par proxy. Cela pose problème pour notre projet car ce site contient l’une des plus grandes bases de données de films, très complet, fiable mais aussi détenteur de nombreux commentaires et critiques de film qui nous auraient été très utiles. Par conséquent, nous avons décidé de prendre contact avec cette société en expliquant les faits, afin de demander une autorisation de parcours des données. Malheureusement, nous avons reçus une réponse négative. En effet, la charte stipule l’interdiction de parsing / scraping de données. Cependant, une sélection de données peut être téléchargée gratuitement où encore une licence permettant l’accès à la totalité des données peut être acheté. Dans les deux cas, cette solution va à l’encontre de notre projet qui consiste à parcourir des pages spécifiques dans le but d’en extraire les données. Voici les messages envoyés à IMDB et leur réponse vis-à-vis de ce dernier :

Demande d’autorisation d’exploitation des données auprès de IMDB

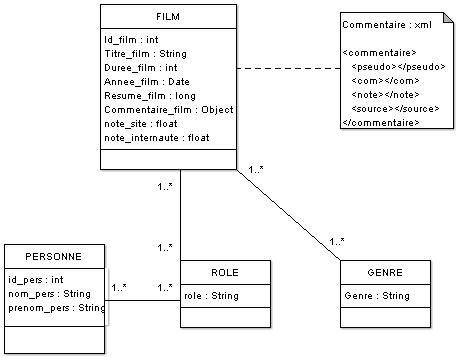
Réponse de IMDB suite à notre demande

**Souci de délais :** Tous les objectifs n’ont pas pu être atteints.

1. Base de données

Au début de notre projet s’est posé le problème du choix du type de base de données que nous allions utiliser. En effet, il nous est mis à disposition deux bases de données Oracle XMLP1 et XMLP2. Il fallait donc choisir entre une base de données relationnelle qui prévoit l’utilisation de SQL2 ou une base de données objet utilisant du SQL3.

Pour une utilisation de ce genre il s’est révélé plus simple d’utiliser une base de données relationnelle.



## Problèmes rencontrés

1. Restitution des données

La restitution des données consiste en la création d’une interface graphique permettant à son utilisateur de naviguer aisément pour trouver les avis et commentaires concernant un film donné. En effet, il s’agit de faciliter l’accessibilité aux données dans le but de cacher les étapes techniques complexes permettant d’aboutir à la recherche. Un utilisateur peu donc être ignorant des problèmes auxquels sont confrontés les développeurs de l’application car aucune manipulation compliquée n’est à executer.

La restitution s’effectue alors principalement sous le langage de programmation PHP. Bien sûr on retrouve aussi Du HTML et du CSS permettant la mise en forme dynamique des pages de l’interface.

L’interface doit être ergonomique et homogène afin de faciliter la navigation mais aussi sobre tout en respectant un choix de couleur épuré. Pour ce faire, Nous avons établis des règles permettant d’atteindre facilement l’interface recherchée. Nous avons aussi établis des aspects fonctionnels et techniques permettant une navigation améliorée sur l’interface.

## Aspect graphique

### Police de caractère

Nous avons choisis la police de caractères « verdana » sans serif où les empattements des caractères sont inexistants et donnent une impression de modernité. Le nombre de police est limité à une seule car trop de diversité donne un aspect lourd à l’interface qui devient désagréable à regarder. Cela porterait atteinte au confort de navigation de l’utilisateur. Aussi l’utilisation d’un nombre de polices de caractères trop élevé sera limitée et nous n’utilisons pas de types de police gras afin de remédier à cet effet.

### Les couleurs

Une interface optimale ne contient pas plus de trois couleurs. Ainsi, nous nous basons sur deux couleurs principales qui sont le Orange et le Kaki.

### Les liens

Afin d’optimiser la souplesse de l’interface nous utilisons au maximum des liens notamment dans la barre de menu qui est le moyen le plus efficace de recherche sur les interfaces.

### La tailles des éléments

Les pages seront assez courtes et le nombre d’éléments affichés sont limités. Cela nécessite le développement d’un système de pagination en PHP permettant de répartir les films sur un nombre de pages nécessaires. Aussi, toute liste de film n’affiche que les informations capitales, utile pour sa recherche. En cliquant sur ce dernier, la fiche entière du film sera affichée, permettant ainsi de voir tous les détails relatifs au film, tels que nom d’auteur, année de production, note attribuée, commentaires, etc.…

### Le menu

Le menu comporte les principaux liens nécessaires à une recherche rapide de données. En effet, ici on retrouve des liens vers différentes pages tels que la recherche par alphabet ou par genres. Aussi on y retrouve le formulaire de recherche simplifié.

Ici nous avons à faire à un menu de type classique tel qu’on les retrouves souvent. L’avantage est la pertinence et l’efficacité de recherche car l’utilisation d’un menu est universelle pour toute application. L’utilisateur ne sera donc pas perturbé et trouvera rapidement les données de film recherchés.

### Le champ de recherche

Il permet une recherche rapide d’un fim donné sous condition de connaitre un minimum d’information sur ce dernier (nom…). C’est une recherche approximative et plusieurs résultats peuvent être trouvés et affichés.

Le formulaire sera accompagné d’un script d’**auto-complétion** en ajax. Démuni d’un bouton de validation de la recherche, pour chaque caractère entré dans le champs, les résultats affichés seront réactualisés. Cette méthode apporte un confort conséquent à l’utilisateur mais peut cependant ralentir le système étant donné que la requête est ré-executée pour chaque nouveau caractère ajouté. Dans notre cas, les requêtes utilisées pour la recherche de données dans Oracle sont simples, n’utilisant que peu de jointure empéchant un ralentissement trop important de la recherche. Aussi, le nombre de film ne dépassant pas les quelques milliers, le parcours de la base Oracle se fera assez rapidement.

### La pagination

L’interface de restitution des données doit afficher un nombre de données assez important en l’occurrence d’une liste de films assez nombreux. En effet, ils existent trop de films pour pouvoir les afficher dans une seule page. De ce fait, nous utilisons la solution de pagination pour les films et commentaires dès que le nombre d’éléments donnés est d’ordre trop important. Cette solution consiste en la répartition du nombre de film sur un nombre de pages donnés.

## Aspect technique et structure de l’interface

### Le modèle MVC

Pour bien structurer le site de restitution nous nous baserons sur un modèle MVC qui consiste en la séparation du code produit en 3 parties distinctes que voici :

* Le **modèle** qui représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application et définit les méthodes d'accès.
* La **vue** correspond à l'interface avec laquelle l'utilisateur interagit. Elle est conçue en html, CSS et JavaScript. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle, et de permettre à l'utilisateur d'interagir avec elles.
* Le **contrôleur** synchronise la gestion des évènements pour mettre à jour la vue ou le modèle. Ainsi il sélectionne le modèle et la vue adéquate selon les souhaits de l’utilisateur par interaction avec l’interface graphique.

Ainsi, cette méthode de conception permet de bien ordonner les différents éléments afin de faciliter le travail du développeur qui aura une facilité de navigation au sain de son code. Par conséquent, la mise à jour, elle aussi, par la même personne ou un autre développeur est plus simple et plus rapide à effectuer.

### Schéma de navigation

Page principale (affiche tous les films par défaut).

Menu

Ajouter un film (permet la recherche d’un film grâce à un formulaire via l’application java)

Page recherche simple (fonction Ajax qui fait une recherche approximative d’un film par nom)

Page genres (affichage des genres disponibles dans la base de données Oracle)

Page films (propose un tri par alphabet)

Affichage des films disponibles sur les sites. Permet de choisir le bon résultat.

Page liste film (liste les films disponibles selon les options sélectionnées. C’est un affichage simple)

Affichage détaillé d’un film

### Le fichier conf.ini

Afin de pouvoir configurer facilement l’espace graphique et la connexion à la base de données Oracle, nous utiliserons un fichier texte « conf.ini » qui comporte les éléments nécessaires à la configuration. Tout utilisateur pourra alors modifier ce fichier selon ces besoins en ouvrant ce dernier grâce à un éditeur simple.

Contenu du fichier :

* Les identifiants de connexion à la base de données, c'est-à-dire utilisateur, mot de passe et chaine de connexion.
* Le nombre de films par page pour la pagination.
* Le chemin d’accès à l’application java qui sera exécutée à partir du code PHP.

Structure du fichier :

*# Parametres de connection à la base de données Oracle*

*utilisateur=xmlp2*

*mdp=mlp42*

*path=iutdb*

*# pagination*

*# Nombre de films par page*

*nb\_pagination=4*

*#chemin d acces de l'application java*

*java\_path=chemin d’accès*

Méthode d’exploitation :

Le parcours du fichier se fait par le biais de la fonction « file() » qui permet l’ouverture du fichier puis par « foreach » utilisé pour le parcours des lignes. Nous utilisons la fonction « explode() » qui coupe une chaine de caractère afin de retourner uniquement l’élément de la chaine recherché.

EXEMPLE :

*public static function configuration($element){*

*/\*-------------test de lecture du fichier ini----------------\*/*

*$fichier = 'conf.ini';*

*//ouverture du fichier*

*if(file\_exists($fichier) && $fichier\_lecture=file($fichier)){*

*//parcours des lignes pour trouver l’élément recherché*

*foreach($fichier\_lecture as $ligne){*

*//récupération de l’élément*

*$var = explode("=", $ligne);*

*if ($var[0] == $element){*

*return $var[1];*

*}*

*}*

*} else { //après erreur d’ouverture du fichier*

*echo 'Erreur d\'ouverture et/ou de lecture du fichier conf.ini<br/>';*

*}*

*/\*-------------------------------------------------------------\*/*

*}*

Appel de la fonction :

*include\_once 'Mutilitaire.php'; //inclus la fonction*

*Mutilitaire::configuration('utilisateur'); //utilisateur est l’élément recherché*

Remarque 1: le parcours d’un fichier peut être inadapté si ce dernier est trop long nuisant aux performances de l’application. Il faut donc réfléchir à son utilisation. Dans notre cas, c’est une solution adapté car le fichier en question est assez court.

Remarque 2 : La publication des identifiants dans un fichier ini peut causer des soucis de sécurité une fois mis en ligne. En effet, on peut accéder au fichier en entrant son URL dans la barre d’adresse. Par conséquent il est important de prendre des mesures afin de protéger le fichier. L’utilisation d’un fichier HTaccess semble donc judicieuse pour empêcher ce genre de problèmes car il restreint les droits d’accès du fichier à protéger. Par manque de temps nous n’établirons pas cette protection mais il serait très envisageable de la mettre en place dans le futur car cette faille est non-négligeable.

Autre solution : Il est aussi possible d’utiliser la méthode « fgets ()» pour la lecture des lignes d’un fichier. Dans ce cas, il faut utiliser la fonction « fopen » et non pas « file() » pour ouvrir le fichier.

### La pagination

La pagination est un élément principal de l’application car elle garanti un meilleur confort de navigation pour son utilisateur d’autant plus que le nombre de film est très élevé.

Le principe est très simple. Après exécution de la requête le résultat (tableau) comportant tous les films est stocké dans une variable session. Le nombre de films autorisé par page est cherché dans le fichier « conf.ini ». Le numéro de la page recherchée est passé en paramètre dans l’URL. Ensuite ne sont sélectionné que les films de cette page dans le tableau.

Exemple : (nombre total d’élément à afficher / nombre de films par pages)

=> Résultat arrondi à l’unité supérieure

*$elements\_totaux = 9 ;*

*$nombre\_par\_page = 2 ;*

*//ceil() permet d’arrondir à l’unité supérieur ;*

*//ne pas utiliser round() ;*

*$nb\_pages = ceil($elements\_totaux/$nombre\_par\_page) ;*

*//affichage des pages*

*if ($nbpages > 1 ){ for ($i=1; $i<=$nb\_pages; $i++){ echo $i ;*

*}*

*}*

Inconvénients : Cette méthode nécessite la récupération de tous les résultats, non seulement ceux de la page sélectionnée. La meilleure solution serait de faire une requête pour chaque nouvelle page en utilisant la contrainte SQL « LIMIT » où l’on limiterait le nombre de résultats.

Point positifs : Une fois le chargement fait, l’affichage est instantané puisqu’aucune autre requête n’est à effectuer

### L’Ajax

Une partie de l’application est réalisée en Ajax afin de garantir plus de fluidité à la navigation. En effet, le champ de recherche muni d’un script Ajax permet d’afficher et de mettre à jour continuellement la liste des films après chaque nouvelle lettre entrée dans le champ. Ainsi, l’utilisateur n’est pas obligé de valider le formulaire par un bouton « valider » et voit les résultats s’afficher en direct.

Le principe de ce script est simple, après chaque nouvelle entrée ce dernier appelle une page PHP à exécuter. Dans notre cas, il effectue une requête et en affiche le résultat grâce à l’insertion d’un modèle et d’une vue adéquate.

Champ de recherche simple

Script AJAX

(S’exécute à chaque nouvelle entrée)

Page PHP

-inclus modèle

-inclus vue

### Extraction XML

Certaines données étant au format XML grâce au type XMLtype dans Oracle, il est nécessaire d’utiliser du Xpath afin de pouvoir extraire chaque commentaire de film stocké dans une hiérarchie prévue à cet effet.

Nous utilisons l’extraction dans une requête qui est la suivante :

*$stmt = ociparse($connect,"*

*SELECT f.TITRE, f.DUREE, f.ANNEE, f.RESUME, f.NOTES\_SITE, f.NOTES\_UTIL, //utilisation du XPATH pour extraire les commentaires stockés sous XML*

*f.COMMENTAIRE.extract('/film/commentaire/com') as COMMENTAIRE, f.COMMENTAIRE.extract('/film/commentaire/pseudo') as PSEUDO, f.COMMENTAIRE.extract('/film/commentaire/note') as NOTE, f.COMMENTAIRE.extract('/film/commentaire/source') as SOURCE FROM FILM f WHERE f.ID\_FILM='".$id."'"); ociexecute($stmt,OCI\_DEFAULT);*

### Exécution d’une application java via PHP.

L’application de restitution est liée à l’application JAVA mais est invisible aux yeux de l’utilisateur. En effet, l’ajout d’un film se fait par le biais d’un formulaire simple dans lequel est entré le nom du film à rechercher. Après validation, le programme java est lancé et effectue la récupération des données.

L’exécution de l’application Java se fait avec une fonction PHP déjà existante « exec() »

Voici le code PHP permettant d’exécuter le programme java :

*<?php*

*class Java {*

*public static function resultats ($data){*

*//récuperation du chemin dans le fichier conf.ini*

*$path = Mutilitaire::configuration('java\_path');*

*echo "<div style='width:400px;'>";*

*//teste si java et installé*

*if (shell\_exec(‘java’)){*

*//vérifie si le formulaire est vide*

*if (empty($data)){*

*echo "< strong>Votre entrée n'est pas valide !!!</strong>";*

*} else { //sinon*

*//exécution de l’application java*

*if ($output = shell\_exec($path.'/'.$data )){*

*echo "<strong>Commande exécutée : ".$data."</strong><br/>";*

*echo $output."<br/><br/><br/><br/></div>";*

*} else {*

*echo "< strong>Erreur, vérifiez le path dans le fichier conf.ini</strong>";*

*}*

*}*

*} else { //Si java n’est pas installé…*

*echo "<strong>Votre ordinateur nécéssite l'installation de l'environnement Java</strong><br/>*

*<span style='font-size:10px;'>(Le téléchargement est disponible <a href='http://www.java.com/fr/download/index.jsp'>>>ici<<</a>)</span>";*

*}*

*}*

*}*

*?>*

Conclusion

Travail d’équipe -> enthousiasme, bonne entente, répartition de travail

Problèmes rencontrés

Objectifs futurs

Ergonomie souplesse fiabilité

A changer : utiliser des vue au lieu du xpath dans le php.