

## TP n° 8

**Consignes** les exercices ou questions marqués d'un \* devront être d'abord rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommandé d'être dans les mêmes conditions qu'en examen : pas de document autres que le cours ni de calculatrice. Tous les TPs se font sous Linux.

### 1 Préambule

Le but TP est le suivant :

- avant dernière chance de configurer Eclipse pour un intégrer un serveur Tomcat
- écrire une application Web simple selon le modèle MVC : utilisation de JSP+JSTL pour la vue, JDBC pour le modèle et HttpServlet pour le contrôleur
- **il est interdit de renommer les classes et packages Java**
- **il ne faut pas importer le projet Eclipse du TP avant d'avoir correctement configuré Eclipse!**

### 2 Configuration d'Eclipse

Si vous n'avez pas correctement configuré Eclipse, suivez les instructions du TP 7.

### 3 Un vraie base de données de films et son interface Web

On suppose qu'une base de données est initialisée avec les tables suivantes :

```
CREATE TABLE PEOPLE (pid INTEGER PRIMARY KEY,  
                      firstname VARCHAR(30),  
                      lastname VARCHAR(30));  
  
CREATE TABLE MOVIE (mid INTEGER PRIMARY KEY,  
                    title VARCHAR(90) NOT NULL,  
                    year INTEGER NOT NULL,  
                    runtime INTEGER NOT NULL,  
                    rank INTEGER NOT NULL);  
  
CREATE TABLE ROLE (mid INTEGER REFERENCES MOVIE,  
                    pid INTEGER REFERENCES PEOPLE,  
                    name VARCHAR(70));  
  
CREATE TABLE DIRECTOR (mid INTEGER REFERENCES MOVIE,  
                        pid INTEGER REFERENCES PEOPLE,  
                        CONSTRAINT dir_const UNIQUE(mid,pid));
```

Vous utiliserez pour la suite du TP le compte de base de donnée `knigye10_a`. Vous veillerez à ne faire **que** des SELECT et à surtout ne pas détruire les tables!

Si vous êtes intéressé par créer les tables dans votre propre base, vous pouvez utiliser les instructions en fin de TP.

On souhaite créer une interface permettant d'afficher les films de la base sous forme d'une liste « id : titre ». La liste affiche par défaut les films dix par dix et la page d'affichage contient deux liens permettant de naviguer vers les 10 films suivants ou précédents. Un exemple d'une telle page est indiqué à la figure 1. Le code est architecturé de la manière suivante :

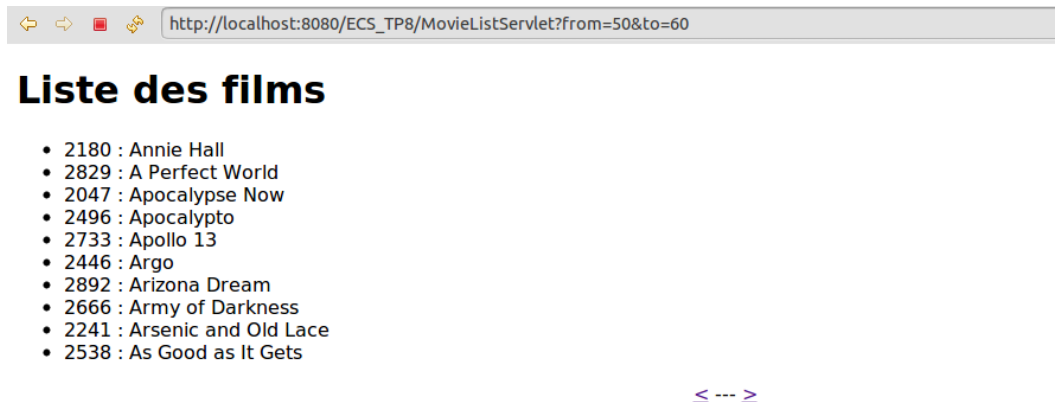


FIGURE 1 – Interface d'affichage de films

- La classe MovieDB encapsule la connection à la base et présente le résultat de requêtes SQL sous forme de collection Java. C'est le modèle.
- La classe MovieListServlet est chargée lorsque l'on navigue dans la liste des films. Elle utilise MovieDB pour récupérer la liste de films souhaités et stocker les résultats trouvés dans des attributs de requêtes. C'est le contrôleur.
- La page movie\_list.jsp qui récupère les données stockées par le contrôleur et les affiche au moyen de tags JSP. C'est la vue.
- La classe auxiliaire Pair<X,Y> qui permet simplement de stocker deux objets en même temps.

## Questions

1. ★ Donner une requête SQL permettant d'avoir tous les films se trouvant entre les positions  $i$  et  $j$  dans l'ensemble des films triés par titre (on suppose que  $i$  et  $j$  sont des constantes connues).
2. ★ Donner une requête SQL permettant d'avoir pour un film d'id  $i$  donné, la liste des noms et prénoms de ses réalisateurs.
3. Dans la classe MovieDB, compléter la méthode getNumMovies() qui renvoie le nombre de films dans la base. Faites en sorte que ce nombre soit calculé une fois pour toute et stocké afin de ne pas provoquer un appel à la base à chaque appel de méthode.
4. Dans la classe MovieDB, compléter la méthode getMovieList renvoyant la liste des films se trouvant entre les position from (inclus) et to (exclus) dans la liste triée des films. La liste est renvoyée sous forme d'une List de Pair, contenant le mid du film et son titre.
5. On considère la classe MovieListServlet. Compléter la méthode doGet pour :
  - (a) récupérer dans une variable de session "db" un objet MovieDB ou le créer (et le stocker dans la variable de session "db") s'il n'existe pas.
  - (b) récupérer les paramètres from et to de la requête. Ces derniers doivent être convertis en entiers de la manière suivante :
    - si from n'est pas une chaîne de caractères convertible en entier, il est converti en 0.
    - si to n'est pas une chaîne de caractères convertible en entier, il est converti en from+10.
    - après conversion, si from est inférieur à 0 il est mis à 0

— après conversion, si `to` est supérieur au nombre de films dans la base (accessible via `db.getNumMovies()`), il est mis à ce nombre.

- (c) récupérer la liste des films triés entre les positions `from` et `to` et stocker cette liste dans un attribut de requête "movies"
- (d) stocker les entiers `from` et `to` dans deux attributs de requêtes "from" et "to"
- (e) enfin, rediriger la page au moyen d'un `RequestDispatcher` vers la page `movie_list.jsp`.

Testez votre code en « exécutant » depuis eclipse la classe `MovieListServlet`. Celle-ci s'affiche sans erreur et vous redirige immédiatement vers la page `movie_list.jsp` (qui pour l'instant n'affiche rien de spécial).

- 6. Dans le fichier `movie_list.jsp`, effectuer un test `c:choose` et tester que l'attribut de requête "movies" est non-vide. Si c'est le cas, alors le parcourir au moyen d'un `c:forEach` et afficher dans une liste non-énumérée (`ul/li`) les titres des films chacun précédé de leur id. Si "movies" est vide, alors rediriger vers l'URL `MovieListServlet`.
- 7. \* si la page affiche les films entre *i* et *j*, quelle URL permet de naviguer aux dix films précédents? aux dix suivants?
- 8. Afficher en bas de la page des films deux liens pour naviguer vers les films suivants ou précédents.
- 9. Faire en sorte que les liens « arrière » ne s'affiche pas si on est au début de la liste (idem pour le lien « suivant ») si on est à la fin de la liste.

**Utilisation de la base POSTGRESQL au PUIO.** Pour utiliser la base, vous devez en premier lieu déterminer votre login court (généralement première lettre du prénom, puis 5 lettre du nom puis éventuellement des chiffres). Vous pouvez déterminer votre login court de la manière suivante. Dans un terminal, effectuer les commandes :

```
cd
pwd
```

La première s'assure que vous êtes dans votre répertoire personnel, la seconde affiche le chemin de ce dernier. Le nom du dernier répertoire est votre login court. On suppose dans la suite que ce dernier est `monlogin`.

Vous disposez d'un utilisateur `Postgresql monlogin_a` dont le mot de passe est `monlogin_a` (ce n'est **pas** le mot de passe de votre compte) et qui possède les droits sur une base `monlogin_a`.

Vous pouvez vous connecter à la base depuis la console :

```
psql -h tp-postgres -U monlogin_a
```

Une fois connecté, vous pouvez effectuer des requêtes. Les scripts de création de table sont donnés sur la page du cours. Vous pouvez les exécuter dans la console `psql` avec la commande :

```
\i create_movies.pg_sql
\i insert_all_movies.sql
```