université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

mardi 16 juin 2020

Programmation d'Applications Web Avancées
M2 Fiil

TP n° 3

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 même 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 :

— écrire une application Web plus complexe selon le modèle MVC : utilisation de JSP+JSTL pour la vue, JDBC pour le modèle et HttpServlet pour le contrôleur, utilisation des filtres pour l'authentification et sortie JSON.

— **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 Authentification

Note : Le code fourni part du corrigé du TP2. Une modification apportée est que le fichier `movie_list.jsp` se trouve maintenant dans le répertoire `WEB-INF/jsp` et est donc inaccessible directement depuis l'extérieur (mais est utilisable par JSP pour générer une page Web depuis un Servlet).

On souhaite rajouter un mécanisme d'authentification par mot de passe.

1. Ajouter un formulaire au fichier `login.jsp` permettant de saisir des paramètres *user* et `pass` et accédant à la page `CheckServlet` via la méthode `post`.
2. Ajouter dans `CheckServlet` du code qui vérifie que l'`user` et le `pass` sont corrects (en les comparant à des constantes, comme au TP1). S'ils sont correct, `CheckServlet` effectue une redirection HTTP vers `MovieListServlet` sinon on effectue une redirection vers `login.jsp`. De plus en cas d'authentification réussie, une variable de session `"status"` valant `"connected"` est stockée dans la session.
3. ⋆ On souhaite maintenant que chaque page teste automatiquement que `status` vaut `"connected"`. Implémenter cela au moyen d'un `Filter` (classe `LoginFilter`). Si un accès est invalide, alors on doit être redirigé vers la page `login.jsp`. On prendra un soins particulier à la manière de rediriger la requête selon les cas.
4. ⋆ On suppose qu'un utilisateur navigue sur `login.jsp` sans aucun *cookie* (première visite) et se connecte avec succès. Donner exactement la liste des actions internes au serveur Web et externes pour arriver sur la liste des films (requêtes HTTP effectuées, méthodes exécutées, pages chargées etc...).
5. ⋆ Que peut on dire sur l'exécution de `LoginFilter` ? Modifier le code pour retirer les appels superflus.
6. Ajouter à la page `movie_list.jsp` un bouton de déconnection. Ce dernier accédera à `LogoutServlet` qui redirigera vers `login.jsp` après avoir détruit la session.

3 Web Service

On souhaite implémenter un Web Service permettant de lister les ids de films, puis pour un film donné, son titre. Attention, on fera en sorte que les Servlets de cet exercices ne nécessitent pas d'être authentifié (soit en désactivant le filtre de l'exercice précédant, soit en lui faisant ignorer les servlets créés ici).

— Créer un servlet `MovieWSServlet` similaire à `MovieServlet` qui prend deux paramètres `from=`i et `to=`j en GET et qui renvoie un fichier JSON de la forme :

```
{  "status" : "success",
   "size" : n,
   "ids" : [ "1234", ...,  "4849" ]
}
```

où `n` est la taille du tableau `ids`, ce dernier contenant les mid des films triés par titre entre les positions i et j.

— Étendre le servlet pour qu'il soit associé aux URLs de la forme : `MovieWSServlet/mid1234` et qui affiche en réponse :

```
{  "status" : "success",
   "mid" : "1234",
   "title" : "Le titre du film"
 }
```

— En cas d'erreur (film inexistant, exception) le JSON suivant est renvoyé :

```
{  "status" : "error",
   "message" : "message d'erreur"
 }
```