Retour rapide sur les Servlets, JSP, MVC

1 Mise en route

Nous allons rapidement revoir les technologies présentées lors du cours de M1. Pour ce faire, suivez les étapes ci-dessous :

- **Java** : Vérifiez la disponibilité de Java en version 11 ou 14 sur votre poste (`java -version` et `ls /usr/lib/jvm`).
- **Eclipse pour JEE** : Pour facilter l'utilisation de la dernière version d'**Eclipse** et pour ne pas emcombrer votre compte utilisateur, j'ai prévu un script qui charge l'IDE dans `/tmp` et le lance automatiquement. Suivez les étapes ci-dessous pour récupérer le script et l'utiliser :

  Vérifiez la disponibilité de **Eclipse pour JEE**. Dans le cas contraire, téléchargez [1] l'archive.

  ```
  cd
  wget http://tinyurl.com/jlmassat/ens/arch-app/ress/eclipse-jee-wrapper
  chmod u+x ./eclipse-jee-wrapper
  ./eclipse-jee-wrapper
  ```

  Si vous travaillez sur votre prore machine, il est plus simple de télécharger l'archive [2].
- **TomEE (extension de Tomcat)** : Cette année nous allons utiliser **TomEE** à la place de **Tomcat**. C'est une version qui complète Tomcat par d'autres outils que nous utiliserons dans les TP suivants. Téléchargez [3] l'archive en version 9 et décompressez TomEE (le conteneur WEB qui va abriter nos applications WEB).
- **Environnement** : Dans Eclipse, (menu *Préférences/Server/Runtime environments*), créez un environnement d'exécution de type Tomcat 9 mais pour lequel vous allez utiliser TomEE 8 et Java 11/14 (en fait TomEE 8 intègre Tomcat 9).
- **Projet** : Dans Eclipse, créez un projet *Dynamic Web Application* basé sur l'environnement précédent et la version 4.0 de l'API (première fenêtre).
- **Maven** : Convertissez ce projet à Maven (menu contextuel/Configure).
- **JUnit** : Ajoutez au *buildpath* la librairie JUnit 5.
- **Première page JSP** : Dans votre projet, utilisez les aides pour créer une nouvelle page JSP (de nom `index.jsp`).
- **Serveur** : Dans l'onglet *Server*, créez un nouveau serveur basé sur l'environnement précédent et associez votre application à ce serveur (elle doit apparaître sous le serveur).
- **Première exécution** : Tentez d'exécuter votre projet (menu contextuel *Run as.../Run on server*).
- **Modification** : Ajoutez un paragraphe à votre page JSP et vérifiez qu'il est bien visible dans le navigateur.

2 Application exemple

- Arrêtez le serveur.
- Récupérez l'application exemple [4] et décompressez l'archive dans `$HOME` .

1. ress/
2. ress/
3. ress/
4. sample-application.zip

- Recopiez à partir de l'exemple et dans votre projet les parties intéressantes (code source Java et code source JSP) :

```
rsync -av $HOME/sample-application/ $HOME/votre_workspace/votre_projet/
```

- Effectuez une mise-à-jour MAVEN (menu contextuel *Maven*/*Update project*).
- Tentez une nouvelle exécution et explorez cette petite application.
- Testez la requête `/hello` et vérifiez avec la page `WebContent/swa/hello.jsp`.
- Explorez l'architecture des classes java.
- Lancez un test unitaire avec Junit 5 (les classes sont déjà présentes).

3 Améliorations

- Etudiez les classes `JpaMoviesDao` et `AutoCloseableEntityManager`. Simplifiez la méthode `findAllMovies` en utilisant le schéma des deux autres méthodes.
- Ajoutez une fonction de suppression (nouveau bouton, nouvelle méthode de l'interface, nouvelle méthode de la Dao, nouvelle méthode de test, nouvelle entrée danss la servlet et affichage).
- Ajoutez une fonction de création d'un nouveau film en suivant le modèle ci-dessus.
- Nous pouvons facilement détourner l'édition d'un film pour en modifier un autre : comment faire ? Comment corriger ce problème (en placant en session l'ID du film à modifier) ?
- Dans l'édition, le traitement d'une année syntaxiquement incorrecte n'est pas très satisfaisante (par exemple `2000Z`). Proposez une solution technique plus élégante pour traiter les problèmes de validation syntaxe et sémantique en même temps (il faut enrichir la classe de validation).

4 Utiliser CDI

- Dans notre exemple, l'instance de la Dao est injectée manuellement dans la servlet. Commençons à utiliser **CDI** pour gérer cette injection :
  - ▷ créez un fichier **complètement** vide `src/META-INF/beans.xml` (activation de CDI),
  - ▷ ajoutez l'annotation `@Dependent` à la classe `JpaMoviesDao`,
  - ▷ ajoutez l'annotation `@Inject` à la propriété `dao` de la servlet et supprimez l'instanciation manuelle de la méthode `init`,
  - ▷ testez la nouvelle version,
  - ▷ ajoutez à votre Dao une méthode `init` annotée `@PostConstruct`,
  - ▷ ajoutez une annotation `@PreDestroy` à votre méthode `close`,
  - ▷ supprimez l'appel à `close` dans la servlet et vérifiez le bon fonctionnement.
- Créez une nouvelle implantation de l'interface Dao qui ne fait presque rien (un bouchon). Vous devez obtenir une erreur (ambiguïté CDI sur le choix de la bonne classe d'implantation). Donnez des noms à vos implantations (`@Named` dans le cours) pour régler ce problème.