Retour rapide sur les Servlets, JSP, MVC et Spring-Boot

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 21+ sur votre poste :

```
java -version
ls /usr/lib/jvm
```

- **Eclipse pour JEE** : utilisez la version installée sur les postes DOSI ou téléchargez l'archive.
- Si vous travaillez sur votre **propre machine**, il est plus simple de télécharger l'archive.

2 Application exemple

Le framework **Spring boot** a pour objectif d'accélerer et de simplifier la mise en place d'une application basée sur Spring.

Cet objectif est atteint par l'intégration automatique de nombreux composants et un système d'auto-configuration qui couvre la majorité des cas.

2.1 Mise en place d'un projet

Préparez un environement pour tester une application WEB basée sur Spring-boot :

- Téléchargez le projet Maven déjà préparé.
- Décompressez cette archive.
- Importez, dans Eclipse (ou dans IntelliJ), un projet Maven et choisissez le répertoire précédent.
- Exécutez ce projet (**Run as .. Java application** classe `myboot.Starter`).
- Testez le bon fonctionnement à l'adresse `http://localhost:8081`

> **Note :** Spring boot lance une instance embarquée de Tomcat pour déployer votre application WEB. Elle est donc directement accessible.

Travail à faire :

- Stopper l'exécution.
- Lancer un test unitaire (les classes sont déjà présentes).
- Faites un **Run as ... Maven build... goal : package** dans Eclipse.
- Faites la même chose en ligne de commande :

Build du projet

```
cd repertoire_de_votre_projet
mvn package
```

- Ces deux opérations ont généré une fichier `.war` (ou `.jar`) dans le répertoire `target`.
- Lancez votre application en ligne de commande :

Exécution directe du projet

```
cd repertoire_de_votre_projet
java -jar target/nom_de_votre_fichier.war (ou .jar)
```

- la même chose avec Maven :

Exécution du projet avec Maven

```
cd repertoire_de_votre_projet
mvn spring-boot:run
```

Note : Il est donc très simple de compiler, déplacer et exécuter une application WEB avec ce type d'outil.

2.2 Explorer ce projet

Les projets **Maven** ont une structure particulière :

- `pom.xml` : configuration de Maven
- `src/main/java` : le code source Java
- `src/main/resources` : les ressources (fichiers XML, images, propriétés, etc.)
- `src/main/resources/static` : les ressources statiques de votre application WEB (CSS, images, autres)
- `src/main/resources/application.properties` : le fichier de paramétrage de Spring boot
- `src/test/java` : le code source Java de test
- `src/test/resources` : les ressources pour le test
- `src/main/webapp` : les fichiers de l'application WEB (pages JSP par exemple)
- `src/main/webapp/WEB-INF` : le répertoire `WEB-INF`
- `src/main/webapp/WEB-INF/jsp` : les pages JSP cachées

Note : Pour traiter les questions ci-dessous, vous aurez peut-être besoin d'utiliser le cours Architecture JEE de M1 pour avoir quelques explications.

Travail à faire :

- Faites fonctionner l'application WEB,
- Explorez le(s) contrôleur(s) (classes annotées `@Controller` et les pages JSP. Changez le message dans le fichier `src/main/resources/application.properties` et vérifiez sa prise en compte.
- Explorez les classes de test.
- **Remarque** : Les pages JSP sont cachées dans le répertoire `WEB-INF/jsp`. Elles ne sont donc pas accessibles via une URL construite par le client. Ce dernier doit donc passer obligatoirement par un contrôleur pour envoyer une requête.
- Explorez les interfaces DAO (nommées `Repository`). Elles sont basées sur l'utilisation de Spring Data : vous fournissez la spécification (l'interface) et le framework se charge de l'implémentation.
- Explorez la classe de démarrage : `Starter` et la classe de configuration (annotée `@Configuration`). **Explication** : La méthode `onStartup` est automatiquement appelée par Spring pour initialiser l'application. Cette dernière va créer une servlet générique (`DispatcherServlet`) qui va récupérer toutes les requêtes et les aiguiller vers le bon contrôleur.
- Préparez un onglet vers la documentation.
- Préparez un onglet vers la Javadoc.

3 Améliorations

Travail à faire :

- Modifiez le contrôleur `Hello` (et l'affichage) pour placer un compteur en session qui s'incrémente à chaque appel.
- Ajoutez à cette application la capacité à lister les films anciens (avant 2000). **À faire** : nouveau bouton, nouvelle méthode de l'interface DAO (utilisez l'annotation @Query), nouvelle méthode de test, nouvelle entrée dans le contrôleur et affichage).
- Ajoutez à cette application la capacité à supprimer des films (avec une confirmation préalable). **À faire** : Vous devrez ajouter deux actions dans le contrôleur, une nouvelle page JSP et modifier la liste des films.
- 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) ?

4 Utiliser Spring security

L'application est dotée de deux configurations de Spring security. La première (associée au profil `open` défini dans le fichier de configuration `application.properties`) ne pose aucune contrainte. La seconde (profil `simple`) réclame une authentification.

Travail à faire :

- Comparez les deux classes de configuration.
- Activez la deuxième configuration en modifiant `application.properties`.
- Vérifiez à nouveau votre application. Tentez de violer la protection CSRF qui se trouve dans un champ caché du formulaire.
- Dans l'affichage d'un film, faites en sorte que les administrateurs puissent voir l'ID d'un film (à titre indicatif) (consultez le dernier exemple du dernier TP sur Spring MVC dans le cours de M1).
- Rendez accessible un bouton de déconnexion (action `/logout`).