TP FRAMEWORK 2

TP sur SPRING, LOG4J, DWR, JUNIT, HTTPUNIT

Exercice 1 : SPRING MVC

Nous allons de nouveau reprendre l'exemple du "HelloWorld". Dans Eclipse, il vous faudra créer un projet Web Dynamique puis importer l'ensemble des fichiers disponibles sur la page Web. Ensuite, il vous faudra réutiliser ces fichiers en les plaçant au bon endroit dans votre projet Eclipse.
Tout d'abord, vous pouvez importer le fichier `web.xml` et remplacer celui qui a normalement été créé en même temps que le projet.
Il faut maintenant importer un fichier XML qui porte le nom `dispatcher-servlet.xml` définit ci dessus suivi du préfixe -servlet. On va donc créer un fichier `dispatcher-servlet.xml` dans notre répertoire WEB-INF.
Puis, on commence par importer **les vues** dans le répertoire `WebContent` : les pages jsp

- une page d'accueil `default.jsp` qui va nous rediriger vers notre formulaire.
- une page avec un formulaire : `form.jsp`, avec des champs pour saisir son nom et un bouton pour valider le formulaire.
- on accèdera à une page `hello.jsp` qui affichera un message de bienvenue avec le nom saisi dans le formulaire de la page `form.jsp`.

.
Passons maintenant à la partie **Modèle**. On importe la classe `MessagePersonneCommand` que l'on place dans le répertoire source du projet.
Ensuite, passons à la partie **contrôleur**. On importe la classe `IndexController` qui héritera de la super classe `SimpleFormController` du Framework Spring. On se contente de redéfinir la méthode `onSubmit()` qui va permettre de rediriger vers la bonne vue lors de la validation du formulaire.
Il est important lorsqu'on produit un formulaire de vérifier le contenu des informations saisies pour s'assurer de leur validité. Pour cela Spring propose une interface `Validator`. C'est pourquoi nous important une classe `MessagePersonneValidatc` qui implémentera cette interface. On associe le validateur au contrôleur dans le fichier `dispatcher-servlet.xml`.
Vous pouvez ensuite exécuter votre exemple au moyen d'un serveur JBoss ou Tomcat.

Exercice 2 : SPRING IoC

Suivant le même principe que l'exercice précédent, il s'agit de construire une application répondant au modèle MVC IoC (Inversion de contrôle). Nous allons prendre l'exemple d'une interface `Logger` qui définit les méthodes proposées par un service de logging. On développera deux implémentations de cette interface `ConsoleLogger` et `MailLogger` qui génère respectivement des logs dans la console et par email, afin de voir comment on peut simplement instancier des implémentations différentes de l'interface `Logger` en utilisant Spring.
L'interface et ses implémentations doivent être chargée dans un projet Java classique. Il s'agit des fichiers `Logger.java`, `ConsoleLogger.java` et `MailLogger.java`.
Il faut ensuite configurer Spring au moyen d'un fichier XML `spring.context.xml` qui sera placé à la racine du projet.
On va ensuite importer une classe de test unitaire (`Junit Test Case`) appelée `TestSpringIoC`.
On peut alors exécuter le test Unitaire. Il suffit de lancer la classe TestSpringIoC, pour cela on sélectionne la classe dans le package explorer, clic droit dessus puis **Run As** et **JUnit Test**. On a alors dans la console les messages envoyés.

Exercice 3 : DWR

Il s'agit tout d'abord de reprendre l'exemple du cours puis de le faire fonctionner.

Exercice 4 : DWR

Il s'agit de créer un formulaire de saisie pour une Personne comportant les éléments suivants :

- Un **nom** qui sera obligatoire pour la saisie.
- Un **prénom** qui sera obligatoire pour la saisie.
- Un **age** qui sera facultatif pour la saisie mais si il est présent il doit être compris entre 0 et 120.
- Un **numéro de rue** qui est facultatif.
- Un **nom de rue** qui est facultatif sauf si le **numéro de rue** est renseigné.
- Un **code postal** qui est facultatif sauf si le **numéro de rue** est renseigné.
- Un **nom de ville** qui est facultatif sauf si le **numéro de rue** est renseigné.

En utilisant le framework DWR mettait en place cette saisie au moyen de pages JSP, de code JavaScript et de code Java.
Cet exercice doit m'être envoyé sous forme d'un projet Eclipse compressé par mail à cette adresse `Claude.Duvallet@gmail.com` avec comme sujet de message `[mtp] Exercice 4 sur DWR`.

Exercice 5 : LOG4J

Ecrire au moyen du framework LOG4J, des messages informatifs et des messages de debug dans les classes Java du projet précédent.

Exercice 6 : JUNIT

Reprendre les exemples du cours et les faire fonctionner sous Eclipse.

Exercice 7 : JUNIT

Écrire une classe ne permettant que de saisir des nombres entiers au clavier. Ecrire ensuite la classe de test permettant de vérifier le fonctionnement de cette classe.

Exercice 8 : HTTPUNIT

Prenez l'exemple disponible sur ma page Web et testez-le vous. Vous devrez utiliser les librairies disponibles sur le site WEB.