TP6 – Implémentation de l'architecture MVC dans un système Client/Serveur

Avant-Propos

Ouvrez deux terminaux et lancez les **deux conteneurs Docker** :

- **`r401-runtime-container`**
- **`r401-tomcat-container`**

conformément au **TP5** en suivant la configuration de votre machine.

> Au cours de ce TP, vous allez utiliser les notions mises en œuvre dans les **TP3, TP4** et **TP5**.

Exercice 1 : MVC Web

Exercice :

Le but est de développer une **application web** à l'aide d'**AJAX côté client** et du **Framework Jakarta côté serveur** en respectant l'**architecture MVC**.

Concrètement, vous devrez finaliser les fichiers :

- **scripts/myview.js** contenant les méthodes de la vue

dans le dossier **/workspace/21-WebMVC/www** du **conteneur Docker `r401-tomcat-container`** (correspondant au **Front-end**).

- **MyController.java** contenant la classe encapsulant le **contrôleur** et qui étendra naturellement la classe **HttpServlet** dans le **Framework Jakarta**
- **MyModel.java** contenant la classe encapsulant le **modèle**

dans le dossier **/workspace/21-WebMVC/src** du **conteneur Docker `r401-tomcat-container`** (correspondant au **Back-end**).

Afin de faciliter le développement, nous fournissons tels quels :

- **styles/reset.css** et **styles/myscreen.css** assurant la mise en forme
- **myaplication.html** contenant l'interface de la fenêtre de connexion

dans le dossier **/workspace/21-WebMVC/www** du **conteneur Docker `r401-tomcat-container`**.

Les méthodes utiles et les composants de votre application portent volontairement **les mêmes noms que lors de l'implémentation de l'architecture MVC en JavaFX**.

En particulier, il identifie les composants graphiques interactifs de l'application :

- la méthode **handleSayHello()** écrite en **JavaScript** servira de point d'entrée lors d'un appui sur le bouton **[Entrer]** ;
- le champ de texte **m_input** permettra à l'utilisateur de saisir son nom
- la division **m_output** réceptionnera le message de bienvenue

Les contraintes suivantes sont imposées :

- Le paramètre d'entrée (**le nom**) sera envoyé dans une requête à l'URL du **contrôleur** avec une méthode de type **GET**
- La donnée de sortie (**le message**) sera retournée à la vue sous la forme d'**un fichier JSON** qui ne contiendra qu'un champ **'value'** qui contiendra le message

Dans la **servlet contrôleur**, vous remplacerez le code de la méthode **doGet()** par celui de la méthode **doPost()** puis vous supprimerez la méthode **doPost()**.

Pour cet exercice vous vous baserez sur :

- l'API **Jakarta HttpServlet (Java)** du **TP5**
- l'API **Jakarta JSon (Java)** du **TP3**
- l'API **XmlHttpRequest (JavaScript)** du **TP5**

Pour compiler et déployer votre code, dans le **terminal Docker de `r401-tomcat-container`**, lancez :

- `deploy.sh 21-WebMVC`

Pour tester votre code, dans un **navigateur**, ouvrez :

- http://localhost:8081/21-WebMVC/controller?name=toto
- http://localhost:8081/21-WebMVC/myapplication.html

Le premier lien devra afficher un **JSON** avec le message « **Bonjour toto** ».
Le second lien simulera la fenêtre de connexion.

Exercice 2 : MVC Remote

Préambule :

Afin de faciliter la vie des utilisateurs (en général pour bénéficier de performances plus poussées), il est souvent proposé une **application native** (c'est le cas pour Facebook ou Instagram par exemple notamment sur mobile) en plus d'une **application web**.

Évidemment, **l'application native** doit réutiliser les composants de l'**application web** (la partie **Back-end**). C'est là où intervient l'intérêt de l'architecture logicielle.

Exercice :

Le but est de réimplémenter la vue de l'**application JavaFX** de sorte à ce qu'elle **communique avec la servlet contrôleur développée dans l'exercice précédent**.

Vous devrez finaliser le fichier :

- **MyView.java** contenant les méthodes de la vue

dans le dossier **/workspace/15-RemoteMVC/src** du **conteneur Docker `r401-runtime-container`**.

Comme on reprend le Front-end de l'exercice précédent, on se base sur les mêmes contraintes (requête à l'URL du contrôleur et retour en JSON).

Pour cet exercice vous vous baserez sur :

- l'API **Jakarta JSon (Java)** du **TP3**
- l'API **HttpUrlConnection (Java)** du **TP5**

Pour compiler et exécuter votre code, dans le **terminal Docker de `r401-runtime-container`**, lancez :

- `run.sh 15-RemoteMVC`