Partie 3 – Mise en place des actions CRUD : échanges entre contrôleurs et vues Thymeleaf

Important ! Nous étudions les différentes étapes du développement d'une application web dynamique avec Spring Boot et IntelliJ. Ces étapes étant dépendantes, **il est indispensable que chaque partie soit entièrement réalisée avant de passer à la suivante**. Une partie pourra occuper plusieurs séances de TP.

Dans cette partie, nous allons mettre en place les actions permettant de réaliser le CRUD de la classe du domaine Utilisateur. Ce sera l'occasion de voir la mise en œuvre du MVC dans Spring MVC ainsi que le fonctionnement du moteur de template Thymeleaf.
Note : on ne vise pas à avoir des pages esthétiquement réussies. On se contentera d'un rendu basique. Il y a aucune balise html à ajouter aux vues fournies.

0. Mise en place d'une feuille de style et de fragments Thymeleaf

1. Ajoutez le fichier de style CSS w3.css de w3schools.com[1] en le téléchargeant à l'adresse suivante :
   https://www.w3schools.com/lib/w3.css
   … et en le plaçant dans le répertoire « resources/static » de votre projet.
2. Dans un répertoire « resources/templates/fragments », ajoutez les deux fichiers suivants dont le contenu est donné dans les liens gist suivants :
   - header.html
     https://gist.github.com/raclet/b0339500a73fbe194e0db01eca05e2ae
   - headerinc.html
     https://gist.github.com/raclet/7e7fb0265ef35964c43cb7088faec23f
3. Mettez à jour le contenu de « index.html » avec le contenu suivant :
   https://gist.github.com/raclet/04602cf41c38fb85c06b3aa83601e79e
4. Mettez à jour le contenu de « utilisateurs.html » avec le contenu suivant :
   https://gist.github.com/raclet/71d5ea25a14c99636b32f5af561c05ca
5. Enfin, ajoutez un fichier « error.html » dans le répertoire « templates » avec le contenu suivant :
   https://gist.github.com/raclet/a04447c050e46dd1bb89b7f6df0eeafb
6. Prenez le temps d'examiner le code des vues ajoutées dans les différents fichiers précédents. Lancez votre application pour constater le changement dans les vues.

1. Action *Read*

1. Ajoutez un fichier « utilisateurShow.html » dans le répertoire « templates » avec le contenu suivant :
   https://gist.github.com/raclet/32cbe09f9df001ccdd67b089b0f34532
2. Modifiez la classe « UtilisateurControllerTest » avec le contenu suivant :
   https://gist.github.com/raclet/86445e72cd1fbb9d65a02ba12ad5bd89
3. Modifiez votre code jusqu'à ce que les tests de la question précédente s'exécutent avec le verdict « succès ». Vous aurez besoin d'ajouter du code Thymeleaf dans

1 Lien vers la documentation du fichier CSS : https://www.w3schools.com/w3css/default.asp

« utilisateurShow.html ».
4. Lancez l'application et effectuez quelques tests manuels.

2. Action *Create*

1. Ajoutez un fichier « utilisateurForm.html » dans le répertoire « templates » avec le contenu suivant :
   https://gist.github.com/raclet/4e763e51b6e598c35c6b5235646a97c2
   Notez la déclaration du formulaire via la balise « form » et, en particulier, la présence de code Thymeleaf. Quel sera la conséquence de l'envoi du formulaire ? Notez aussi la présence d'une balise « input » de type « hidden ». Son rôle sera plus clair quand on implémentera l'action *Update*.
2. Modifiez la classe « UtilisateurControllerTest » avec le contenu suivant :
   https://gist.github.com/raclet/a80210b5e3ae523923ea89a209d67534
3. Modifiez votre code jusqu'à ce que les tests de la question précédente s'exécutent avec le verdict « succès ». Vous utiliserez l'annotation @Valid pour imposer que les valeurs récupérées dans le formulaire satisfassent les contraintes d'attributs d'Utilisateur :
   https://spring.io/guides/gs/validating-form-input/
4. Lancez l'application et effectuez quelques tests manuels, en particulier avec des données non-valides.

3. Action *Update*

On souhaite maintenant offrir la possibilité d'éditer les informations d'un utilisateur. Pour cela, on utilisera « utilisateurForm » dont on initialisera la valeur des champs du formulaire à la valeur courante des attributs.

1. Modifiez la classe « UtilisateurControllerTest » avec le contenu suivant :
   https://gist.github.com/raclet/61b48f8ae8828f586b5ebdcf69bb1c7d
2. Modifiez votre code jusqu'à ce que les tests de la question précédente s'exécutent avec le verdict « succès ».
3. Maintenant que vous avez réalisé l'action *Update*, à quoi sert la balise « input » de type « hidden » dans « utilisateurForm.html » ?
4. Lancez l'application et effectuez quelques tests manuels.

4. Action Delete

Enfin, on souhaite pouvoir supprimer un utilisateur. Attention, cette opération est possible uniquement si l'utilisateur à supprimer n'est responsable d'aucune activité !

1. Modifiez la classe « UtilisateurControllerTest » avec le contenu suivant :
   https://gist.github.com/raclet/8c25c57e173df199b5c8725937668bb6
2. Modifiez votre code jusqu'à ce que les tests de la question précédente s'exécutent avec le verdict « succès ».
3. Lancez l'application et effectuez quelques tests manuels.

Questions de réflexion

Dans le cours, nous avons bien insisté sur le fait qu'il fallait limiter le code java dans les vues et exclure le code métier des contrôleurs. Votre code respecte-t'il ces consignes ?