POO : API & Outils
TD et TP

Anass OUSMOI anass.ousmoi@univ-lemans.fr
Karim RAKHILA karim.rakhila@soprasteria.com

TP 3: DM noté 1/2: Réalisation d'une application web avec Spring Boot

Ce TP est noté !

- Vous pouvez le terminer chez vous ;
- A rendre via : UMTICE (cours : POO API et outillages) ;

Vous allez réaliser une application à l'aide de Java et des technologies suivantes :

- Spring https://spring.io/
- Spring Boot https://spring.io/projects/spring-boot
- JPA https://www.tutorialspoint.com/jpa/index.htm
- Hibernate http://hibernate.org/
- H2 http://www.h2database.com/html/main.html
- Spring Data JPA https://spring.io/projects/spring-data-jpa
- Thymeleaf https://www.thymeleaf.org/

Nous allons utiliser beaucoup d'API, n'hésitez pas à chercher sur internet des indications et de la documentation. Suivez les étapes tranquillement en variant que l'application fonctionne toujours.

Etape 1

Commencez par générer votre application sur le site de Spring, avec Spring Initializer :
https://start.spring.io/

Etape 2

Vous allez générer un projet Maven, avec Java, et Spring Boot 2.7.5. Dans le groupe, ajoutez le nom du groupe Maven du projet que vous souhaitez créer.

Etape 3

Ajoutez les dépendances suivantes :
Web, JPA, Hibernate, H2, DevTools, Thymeleaf

Etape 4

Gênerez votre projet.

Etape 5

Cherchez une description succincte de chaque dépendance ajoutée pour trouver à quoi sert-elle, et ajoutez-y ces informations dans un fichier « README.md » à la racine de votre projet.

Etape 6

Lancez votre IDE, configurez le proxy si nécessaire, et importez le nouveau projet
(Vérifiez les aides plus bas dans le document si besoin).

Etape 7

Pour lancer une application Spring Boot, vous pouvez utiliser la commande maven suivante :

```
mvn spring-boot:run
```

Créez une nouvelle « Run Configuration » de type Maven et dans le goal, utilisez la commande ci-dessus. Lancez la nouvelle configuration et regardez ce qui est inscrit dans le terminal pour voir si tout est OK.

Etape 8

Vous devriez avoir le message suivant dans le terminal :

```
Tomcat started on port (s): 8080 (http) with context path ''
```

Ouvrez la page http://localhost:8080.
Si vous avez un message d'erreur, cela signifie peut-être que le port 8080 est déjà utilisé. Vous pouvez ajouter cette configuration dans le fichier application.properties pour démarrer sur le port 9090 :

```
server.port=9090
```

Etape 9

Nous allons maintenant créer notre première page. Commencez par créer les packages « model », et « controller » (Spring respecte le modelé MVC) dans le package racine, celui contenant la classe avec la méthode main. Attention : le contrôleur a besoin de la vue pour fonctionner (pensez-y lors de vos tests).

Etape 10

Dans le package « controller », créez la classe HelloWorldController.java avec le contenu suivant :

```
@Controller
public class HelloWorldController {

   @GetMapping("/greeting")
   public String greeting(@RequestParam(name="nameGET", required=false, defaultValue="World") String nameGET, Model model) {
      model.addAttribute("nomTemplate", nameGET);
      return "greeting";
   }
}
```

https://e-gitlab.univ-lemans.fr/snippets/2

Etape 11

Dans le dossier « resources », créez un dossier « templates » et dedans, le fichier « greeting.html » :

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Blog</title>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
   <p th:text="'Bonjour ' + ${nomTemplate} + ' !'" />
</body>
</html>
```

https://e-gitlab.univ-lemans.fr/snippets/3

Etape 12

Relancez votre application avec le launcher et allez sur la page http://localhost:8080/greeting. Tentez ensuite l'URL suivante : http://localhost:8080/greeting?name=ENSIM

Etape 13

Relisez le code du contrôleur, aidez-vous de documentation sur internet, et répondez aux questions suivantes :
1. Avec quelle partie du code avons-nous paramétré l'url d'appel /greeting ?
2. Avec quelle partie du code avons-nous choisi le fichier HTML à afficher ?
3. Comment envoyons-nous le nom à qui nous disons bonjour avec le second lien ?
Ajoutez les réponses dans le README.

Etape 14

Nous allons maintenant activer la base de données H2. Dans le fichier « application.properties », ajoutez les lignes suivantes :

```
# Enabling H2 Console
spring.h2.console.enabled=true
```

Relancez l'application et allez sur l'URL : http://localhost:8080/h2-console, et appuyez sur « Connect ». Si vous avez une erreur, essayer l'url suivante :

```
jdbc:h2:mem:testdb
```

Etape 15

Vous êtes sur l'interface de votre base de données in-memory !

Etape 16

Nous allons rajouter notre première classe du modèle MVC pour notre site : la classe Address. Dans le package « model » crée tout à l'heure, vous pouvez ajouter la classe Address suivante :

```java
@Entity
public class Address {

    @Id
    @GeneratedValue
    private Long id;
    private Date creation;
    private String content;
}
```

https://e-gitlab.univ-lemans.fr/snippets/4

Rajoutez les méthodes get et set de toutes les propriétés via le bon raccourci de votre IDE.

Etape 17

Relancez-votre application, retournez sur la console de H2 : http://localhost:8080/h2-console. Avez-vous remarqué une différence ? Ajoutez la réponse dans le README.

Etape 18

Expliquez l'apparition de la nouvelle table en vous aidant de vos cours sur Hibernate, et de la dépendance Hibernate de Spring. Ajoutez la réponse dans le README.

Etape 19

Nous allons utiliser Spring pour ajouter des éléments dans la base de données. Pour ce faire, créer un _chier « data.sql » à coté du fichier « application.properties » et ajoutez-y le contenu suivant :

```
INSERT INTO address (id, creation, content) VALUES (1, CURRENT_TIMESTAMP(), '57 boulevard demorieux');
INSERT INTO address (id, creation, content) VALUES (2, CURRENT_TIMESTAMP(), '51 allee du gamay, 34080 montpellier');
```

https://e-gitlab.univ-lemans.fr/snippets/5

Etape 20

Relancez l'application, retournez sur la console H2 : http://localhost:8080/h2-console.
Faites une requête de type SELECT sur la table Adress. Voyez-vous tout le contenu de data.sql ?
Ajoutez la réponse dans le README.

Etape 21

Nous allons maintenant créer une interface de type Repository pour accéder aux adresses.
Une classe Repository permet d'accéder aux données de la base de données. A coté de Adresse, créez une interface nommée « AdresseRepository » avec le code suivant :

```
@Repository
public interface AddressRepository extends CrudRepository<Address, Long> {
}
```

https://e-gitlab.univ-lemans.fr/snippets/6

Etape 22

Nous allons créer un autre contrôleur pour donner un accès aux adresses via l'url http://localhost:8080/adresses. Créez la classe AddressController.java dans le package des contrôleurs.

```
@Controller
public class AddressController {

    @Autowired
    AddressRepository addressRepository;

    @GetMapping("/adresses")
    public String showAddresses(Model model) {
        model.addAttribute("allAddresses", addressRepository.findAll());
        return "addresses";
    }
}
```

https://e-gitlab.univ-lemans.fr/snippets/7

Pouvez-vous trouver à quoi sert l'annotation @Autowired du code précèdent sur internet ? Ajoutez la réponse dans le README.

Etape 24

Créez de plus le fichier « addresses.html » dans le dossier « templates » de tout à l'heure :

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Adresses</title>
<meta http-equiv="Content-Type" content="text/html ; charset=UTF-8" />
</head>
<body>
      <h1>Les differentes adresses</h1>
      <ul th:each="address:${allAddresses}">
            <li th:text="${address.content }" />
      </ul>
      </body>
</html>
```

Etape 25

Vous êtes maintenant prêt et autonome pour la suite !

Etape 26

Et si vous rajoutiez le nom de la personne qui a effectué la recherche d'une adresse ?

1. Ajouter l'auteur dans l'entité « Address.java »,
2. Dans le fichier data.sql, ajoutez la donnée,
3. Coté vue, utilisez la nouvelle donnée et affichez-la.

Etape 27
Pouvez-vous rajouter une navbar pour naviguer entre chaque page ?

Etape 28
Utilisez les Thymeleaf fragments pour pouvoir déplacer la déclaration de la navbar à un seul endroit, et l'inclure dans chaque vue.

Etape 29
Regardez sur internet comment ajouter Bootstrap à votre projet.

Etape 30
Expliquez la méthode que vous avez utilisé pour ajouter Bootstrap dans le README.