Une image contenant texte

Description générée automatiquement133 – Réaliser des applications web en session-handling

Rapport personnel

Version 1 du 27.04.2025

Créé le 17.03.2025

Schmitt Maxime

Module du 17.03.2025 au 15.04.2025

Table des matières

[1 Introduction 4](#_Toc193190487)

[2 Tests technologiques selon les exercices 5](#_Toc193190488)

[2.1 Installation et Hello World 5](#_Toc193190489)

[2.2 Conteneurisation 5](#_Toc193190490)

[2.3 Création d'un projet Spring Boot 5](#_Toc193190491)

[2.4 Connexion à la DB JDBC 6](#_Toc193190492)

[2.5 Connexion à la DB JPA 6](#_Toc193190493)

[2.6 Connexion à la DB JPA avec DTO 6](#_Toc193190494)

[2.7 Gestion des sessions 6](#_Toc193190495)

[2.8 Documentation API avec Swagger 6](#_Toc193190496)

[2.9 Hébergement 6](#_Toc193190497)

[3 Auto-évaluations et conclusions 7](#_Toc193190498)

# Introduction

* Analyser la donnée, projeter la fonctionnalité et déterminer le concept de la réalisation.
* Réaliser une fonctionnalité spécifique d’une application Web par Session-Handling, authentification et vérification de formulaire.
* Programmer une application Web à l’aide d’un langage de programmation compte tenu des exigences liées à la sécurité.
* Vérifier la fonctionnalité et la sécurité de l’application Web à l’aide du plan tests, verbaliser les résultats et, le cas échéant, corriger les erreurs.

# Tests technologiques selon les exercices

## Installation et Hello World

### Points importants

A l’aide du service WSL2 sur Ubuntu, nous avons mis en place un container accessible en local, à l’adresse « localhost/8080 ».

### Questions

* Observez la console pour comprendre comment le projet est lancé et comment il tourne ?

Il est lancé sur une API REST via le service WSL2.

* C'est quoi le build et le run de Java ? Quel outil a-t-on utilisé pour build le projet ?

Le buid sert à compiler le projet, et le run à l’exécuter. Maven

* Y a-t-il un serveur web ?

Oui, Tomcat.

* Quelle version de java est utilisée ?

17.0.4

* S’il y a un serveur web, quelle version utilise-t-il ?

10.1.24

## Conteneurisation

* Pourquoi faire un container pour une application Java ?

Pour travailler dans un environnement isolé et sécurisé.

* Y a-t-il un serveur web ? Ou se trouve-t-il ?

Oui, dans le container.

* A quoi faut-il faire attention (pensez aux versions !) ?

A la version de Java utilisée. Par exemple, dans le Dockerfile de base, le jdk par défaut est le 8, alors que nous utilisons le 17. Il faut également vérifier le chemin d’accès des différents fichiers dans le Dockerfile.

## Création d'un projet Spring Boot

### Questions

* Quelles sont les annotations utilisées dans votre controller ?
  + @RestController

Définit les contrôleurs REST. Se trouve juste avant la déclaration de la classe :

@RestController

public class Controller {

//...

}

* + @GetMapping

Définit une méthode gérant les requêtes http de type GET :

@GetMapping("/getExample")

public String getExample(@RequestParam(value = "name", defaultValue = "World") String name) {

return String.format("Hello, %s!", name);

}

* + @PostMapping

Définit une méthode gérant les requêtes http de type POST :

@PostMapping("/postExample")

public String postExample(@RequestBody String body) {

return "POST : " + body;

}

* + @PutMapping

Définit une méthode gérant les requêtes http de type PUT :

@PutMapping("/putExample")

public String putExample(@RequestBody String body) {

return "PUT : " + body;

}

* + @RequestParam

Définit la prise en compte et la gestion d’un paramètre de la requête :

public String getExample(@RequestParam(value = "name", defaultValue = "World") String name)

* + @RequestBody

Définit la prise en compte et la gestion du body de la requête :

public String putExample(@RequestBody String body) {

## Connexion à la DB JDBC

### Points importants

Afin de pouvoir connecter notre programme à une base de données, il nous faut tout d’abord créer un conteneur, que l’on va créer avec la commande suivante :

docker run --name mysql -d -p 3308:3306 -e MYSQL\_ROOT\_HOST=% -e MYSQL\_ROOT\_PASSWORD=emf123 -v /opt/mysql:/var/lib/mysql mysql/mysql-server:8.0

Le paramètre « -p 3308 :3306 » mappe le port 3308 sur el port 3306. Ainsi, on peut utiliser le port 3308 à la place du 3306, qui est déjà utilisé.

Etant donné que l’on utilise un conteneur, il nous faut également mettre en place la variable suivante dans notre programme :

final String url = "jdbc:mysql://host.docker.internal:" + port + "/" + dbName

Schéma de connexion JDBC :  
Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

## Connexion à la DB JPA

### Points importants

FetchType.EAGER : Charge l’entité associée en même temps que l’entité principale.

FetchType.LAZY : Charge l’entité associée seulement lorsqu’elle est implicitement appelée

### Questions

* À quoi sert l'annotation @Autowired dans vos controlleur pour les Repository ?

Il sert à indiquer un attribut qui sera géré automatiquement par Spring. Cela permet d’automatiser la détection et l’initialisation du Repository.

* A quoi sert l'annotation @ManyToOne dans l'entité skieur ?

Cela indique une relation « N à 1 » depuis un attribut. On l’utilise ici avec le paramètre « FecthType » et la valeur « EAGER » afin de charger immédiatement l’entité liée, alors que la valeur « LAZY » l’aurait fait uniquement sur demande.

* Sur la même ligne, quel FetchType est utilisé et pourquoi, rééssayer avec le FetchType LAZY et faites un getSkieur.

EAGER, afin de récupérer immédiatement l’entité « Pays » lorsque qu’on appelle l’entité « Skieur ». Changer avec le type LAZY crée une erreur interne dans le serveur, car on lui demande une donnée qu’il n’a pas chargée.

## Connexion à la DB JPA avec DTO

### Points importants

Model : Lie une table de la DB à une classe correspondante

Repository : Implémente un CRUD pour un modèle.

Dto : Classe permettant de limiter les attributs retournés, limitant l’accès et augmentant la sécurité.

Services : Classe contenant des méthodes utilitaires.

Controllers : Receveurs de l’EndPoint qui va gérer les liens entre. fichiers et les services

### Questions

* Pourquoi dans ce cas, on retrouve un SkieurDTO et pas de PaysDTO ?

Parce que le pays est chargé d’office dans SkieurDTO, ce qui évite les problèmes de chargement d’une classe pays.

## Gestion des sessions

Les sessions permettent à chaque utilisateur d’avoir son expérience personnalisée. Il sera reconnu par le site selon les informations données au préalable (pseudonyme, nom, prénom, etc…) et il aura des accès selon les privilèges lui étant accordés (différence entre session user et admin par exemple).

Par exemple dans notre projet, la gestion des visites :

Sans session :



Avec session :



## Documentation API avec Swagger

Swagger est un outil indépendant permettant de générer automatiquement la documentation de notre API, apparaissant sur une page Web. On rajoute simplement la dépendance Swagger dans notre fichier « pom.xml ».

Voici un exemple avec la page locale <http://localhost:8080/swagger-ui/index.html> :

Une image contenant texte, capture d’écran, logiciel, nombre

Le contenu généré par l’IA peut être incorrect.

## Hébergement

Lorsque l’on passe sur l’hébergement, il faut déjà adapter les variables afin de ne pas rester en localhost, mais en accédant bien au site depuis la bonne url.

Le processus d’installation et de paramétrage ne faisant pas partie de ce module, ces informations ne sont pas données ici.

La seule partie à faire de ce côté est de créer 3 éléments dans cet ordre précis : compte🡪repository🡪token.

Puis depuis le terminal, on entre cette commande, après quoi on entrera notre token :

docker login -u username

On effectue ensuite ces deux commandes, en adaptant les informations à notre situation :

docker tag [image locale]:latest [username]/[nom repository]:latest

docker push [username]/[nom repository]:latest

Et voilà, si votre serveur docker est bien configuré, tout devrait rouler !

# Auto-évaluations et conclusions

Ce module, bien que partiellement semblable au 151, a été très enrichissant, montrant une nouvelle face de Java qui m’était encore inconnue.

J’ai également beaucoup aimé la jonction entre les deux modules, malgré les complications de communications entre eux, cela nous a réellement permis de comprendre le processus de développement et de mise en service.

J’ai eu du mal à me lancer dans le module, mais une fois parti tout s’est bien passé. J’ai l’impression très nette d’avoir progressé durant ce module, et j’en suis reconnaissant.