133 – Réaliser des applications Web en Session-Handling

Rapport personnel

Pochon Marius

Date de création : 18.03.2025  
Version 1 du 19.05.2021



Module du 17.03.2025 au 15.04.2025

Table des matières

[1 Introduction 2](#_Toc194311052)

[1.1 Description 2](#_Toc194311053)

[1.2 Objectifs 2](#_Toc194311054)

[1.3 Setup 2](#_Toc194311055)

[2 Tests technologiques selon les exercices 4](#_Toc194311056)

[2.1 Installation et Hello World 4](#_Toc194311057)

[2.1.1 Questions 4](#_Toc194311058)

[2.1.2 Résultat 4](#_Toc194311059)

[2.2 Conteneurisation 5](#_Toc194311060)

[2.2.1 Questions 5](#_Toc194311061)

[2.2.2 Résultat 6](#_Toc194311062)

[2.3 Création d'un projet Spring Boot 6](#_Toc194311063)

[2.3.1 Questions 6](#_Toc194311064)

[2.3.2 Résultat 7](#_Toc194311065)

[2.4 Connexion à la DB JDBC 10](#_Toc194311066)

[2.4.1 Résultat 10](#_Toc194311067)

[2.5 Connexion à la DB JPA 11](#_Toc194311068)

[2.5.1 Questions 11](#_Toc194311069)

[2.5.2 Résultat 11](#_Toc194311070)

[2.6 Connexion à la DB JPA avec DTO 14](#_Toc194311071)

[2.6.1 Questions 14](#_Toc194311072)

[2.6.2 Résultat 15](#_Toc194311073)

[2.7 Gestion des sessions 18](#_Toc194311074)

[2.7.1 Questions 18](#_Toc194311075)

[2.7.2 Résultat 20](#_Toc194311076)

[3 Auto-évaluations et conclusions 22](#_Toc194311077)

[3.1 Auto-évaluation 22](#_Toc194311078)

[3.2 Conclusion 22](#_Toc194311079)

# Introduction

## Description

Une étude de cas décrit les exigences à réaliser sur une application web comprenant des lots de données. Les attentes ainsi que les représentations sur l'application web à réaliser sont énoncées. L'application web n'est pas seulement constituée du code-programme, mais aussi des autres ressources telles que fichiers HTML, CSS, images, extraits de bases de données, etc.

## Objectifs

1. Analyser la donnée, projeter la fonctionnalité et déterminer le concept de la réalisation.
2. Réaliser une fonctionnalité spécifique d’une application Web par Session-Handling, authentification et vérification de formulaire.
3. Programmer une application Web à l’aide d’un langage de programmation compte tenu des exigences liées à la sécurité.
4. 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.

## Setup

* VSCode
  + Installation de base
  + Multi Extension Java
  + Multi Extension Spring Boot
  + Extension WSL
  + Extension Docker
  + Extension secondaire
* WSL
  + Installation de base
* Docker
  + Installation de base
  + Une image contenant texte, Police, capture d’écran

    Le contenu généré par l’IA peut être incorrect.Activation d’Ubuntu

# **Tests technologiques selon les exercices**

## Installation et Hello World

### Questions

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

Le projet se lance sur WSL :ubuntu et tourne avec Spring Boot (v3.2.3).

C'est quoi le build et le run de Java ?

* Build : La compilation du code source en .class et la création d’un fichier exécutable (.jar)
* Run : Exécution du fichier exécutable (.jar)

Quel outil a-t-on utiliser pour build le projet ?

Nous avons utilisé Maven.

Y a-t-il un serveur web ?

Notre projet utilise Spring Boot, il intègre Tomcat par défaut.

Tomcat initialized with port 8080 (http)

Quelle version de java est utilisée ?

Nous avons utilisé le JDK 17

Starting Ex6Application using Java 17.0.14 with PID 12052 (/home/adm-poc/ex7/ex6/target/classes start

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

Comme dis plus haut, notre projet utilise Tomcat. Voici la version que nous utilisons :

[Apache Tomcat/10.1.19]

### Résultat

Un exemple d’utilisation de l’exercice

Au lancement de l’application, nous avons une erreur car rien n’est affiché :

Une image contenant texte, capture d’écran, Police, affichage

Le contenu généré par l’IA peut être incorrect.

Nous ouvrons un chemin différent qui nous affiche un JSON :

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

Le contenu généré par l’IA peut être incorrect.

## Conteneurisation

### Questions

Pourquoi faire un container pour une application Java ?

Docker est devenu un outil essentiel en programmation, il nous fait gagner beaucoup de temps. Un container permet d’exécuter un application Java de manière isolée et portable. Voici quelques avantages :

* Portable
* Facile à déployer
* Intégration facile d’une base de données

Y a-t-il un serveur web ? Ou se trouve t'il ?

Sur une application Spring Boot, comme nous avons, le serveur Web (Tomcat) est directement intégré à celle-ci

Il tourne sur le port 8080 comme l’indique le DockerFile (EXPOSE 8080)

FROM openjdk:17-jdk-alpine

VOLUME /tmp

ARG JAVA\_OPTS

ENV JAVA\_OPTS=$JAVA\_OPTS

COPY target/ex6-0.0.1-SNAPSHOT.jar ex6.jar

EXPOSE 8080

ENTRYPOINT exec java $JAVA\_OPTS -jar ex6.jar

ENV DATABASE\_URL=jdbc:mysql://host.docker.internal:3308/133ex6

# For Spring-Boot project, use the entrypoint below to reduce Tomcat startup time.

#ENTRYPOINT exec java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar ex6.jar

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

Il faut faire attention aux versions des applications, que Spring boot et Tomcat supporte la version 17 de Java

### Résultat

Un exemple d’utilisation de l’exercice :

Ici nous avons créé un DockerFile puis nous l’avons RUN

FROM openjdk:17-jdk-alpine

VOLUME /tmp

ARG JAVA\_OPTS

ENV JAVA\_OPTS=$JAVA\_OPTS

COPY complete/target/rest-service-complete-0.0.1-SNAPSHOT.jar gsrestservice.jar

EXPOSE 8080

ENTRYPOINT exec java $JAVA\_OPTS -jar gsrestservice.jar

# For Spring-Boot project, use the entrypoint below to reduce Tomcat startup time.

#ENTRYPOINT exec java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar gsrestservice.jar



## Création d'un projet Spring Boot

### Questions

**Quelles sont les annotations utilisée (commencent par @) dans votre controller ?**

**@RestController**

* Combine @Controller et @ResponseBody.
* Indique que ce composant Spring sert des **réponses JSON** directement.

Exemple :

@RestController

public class MyController {

@GetMapping("/hello")

public String sayHello() {

return "Hello, World!";

}

}

**@GetMapping**

* Gère les requêtes **HTTP GET** (lecture de données).
* Ex : Récupérer une liste d’utilisateurs.

@GetMapping("/users")

public List<User> getUsers() { ... }

**@PostMapping**

* Gère les requêtes **HTTP POST** (ajout de données).
* Ex : Ajouter un nouvel utilisateur.

@PostMapping("/users")

public User createUser(@RequestBody User user) { ... }

**@PutMapping**

* Gère les requêtes **HTTP PUT** (mise à jour complète d’une ressource).
* Ex : Modifier un utilisateur existant.

@PutMapping("/users/{id}")

public User updateUser(@PathVariable Long id, @RequestBody User user) { ... }

### Résultat

Un exemple d’utilisation de l’exercice :

Un exemple d’une requête POST

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

Le contenu généré par l’IA peut être incorrect.

Un exemple d’une requête GET

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

Le contenu généré par l’IA peut être incorrect.

Un exemple d’une requête PUT

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

Le contenu généré par l’IA peut être incorrect.

## Connexion à la DB JDBC

### Résultat

Un exemple d’utilisation de l’exercice

Ici, nous avons fait un appel à une base de donnée en local avec JDBC :

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

## Connexion à la DB JPA

### Questions

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

@Autowired permet à Spring d’injecter automatiquement une instance de SkieurRepository, qui est un Spring Data JPA Repository.

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

L’annotation @ManyToOne est utilisée pour définir une relation de type "plusieurs vers un" dans une entité JPA.

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

FechType Charge automatiquement l’objet Pays dès qu’on récupère l’entité principale.

En utilisant LAZY, ça permet charger Pays **seulement quand on en a besoin**

### Résultat

Un exemple d’utilisation de l’exercice

Dans cet exercice, nous pouvons afficher des skieurs et en créer. Nous avons fait un lien à une base de données local.

Affichage d’un skieur

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

Le contenu généré par l’IA peut être incorrect.

Création d’un skieur :

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

Le contenu généré par l’IA peut être incorrect.

Affichage avec le skieur créer :

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

Le contenu généré par l’IA peut être incorrect.

## Connexion à la DB JPA avec DTO

### Questions

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

Car appeler le skieur qui a une FK du pays

Expliquez à quoi servent les model, les repository, les dto, les services et les controlleurs en vous basant sur le code donné.

**Model (Entité JPA)**

* Ce sont les **objets qui représentent les tables de la base de données**.
* Ex : Skieur.java, Pays.java → contiennent les champs et relations @ManyToOne.

**Repository (DAO - Accès aux données)**

* Interfaces qui **permettent de communiquer avec la base**.
* Ex : SkieurRepository extends JpaRepository<Skieur, Long> → permet findAll(), findById(), etc.

**DTO (Data Transfer Object)**

* Objets utilisés pour **transporter uniquement les données utiles** entre le backend et le frontend.
* Ex : SkieurDTO évite d’envoyer toute l’entité Skieur (et donc les données inutiles).

**Service (Logique Métier)**

* Contient la **logique de gestion** des entités (ex : vérifier des règles métier, faire des calculs).
* Ex : SkieurService qui transforme Skieur en SkieurDTO.

**Controller (API REST)**

* Expose les **endpoints** (/api/skieurs) pour que le frontend accède aux données.
* Il utilise **le service** pour appliquer la logique métier.

### Résultat

Un exemple d’utilisation de l’exercice

Ici nous pouvons ajouter un pays puis un skieur avec un pays :

Ajout d’un pays

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

Le contenu généré par l’IA peut être incorrect.

Création d’un skieur avec un pays :

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

Le contenu généré par l’IA peut être incorrect.

Affichage du skieur avec le pays que nous avons créé :

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

Le contenu généré par l’IA peut être incorrect.

## Gestion des sessions

### Questions

Décrivez dans votre rapport l'utilisation de HTTPSession.

HTTP Session est une interface fournie par Java EE (via jakarta.servlet.http.HttpSession) qui permet de stocker des données **spécifiques à un utilisateur** entre plusieurs requêtes HTTP.

En l’occurrence, nous l’utilisons pour :

1. **Login :** en stockant le nom d’utilisateur.
2. **Visites :** en comptant le nombre de visites sur une page.
3. **Logout :** en déconnectant la session.

Illustrez le fonctionnement des sessions avec des printscreen de vos appels Postman à ces 3 méthodes.

La requête POST de login (nous devons nous loguer avec username = username & password = password) :

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

Le contenu généré par l’IA peut être incorrect.

La requête GET pour savoir le nombre de personne qui se sont connecté avec un username « username »

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

Le contenu généré par l’IA peut être incorrect.

La requête POST de logout :

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

Le contenu généré par l’IA peut être incorrect.

### Résultat

Un exemple d’utilisation de l’exercice :

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

Le contenu généré par l’IA peut être incorrect.

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

Le contenu généré par l’IA peut être incorrect.

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

Le contenu généré par l’IA peut être incorrect.

# Auto-évaluations et conclusions

## Auto-évaluation

J’ai pris pas mal de retard sur le début des exercices, j’ai dû rattraper ce retard et cela m’a mis une « pression » en plus sur le projet. Malgré ça, j’ai le sentiment d’avoir compris les exercices et de les avoir réalisés de manière correcte.

## Conclusion

Pour conclure, nous avons appris à mettre en place et utiliser plusieurs technologies dans un projet JAVA :

* Maven
* WSL
* Spring BOOT
* Docker
* POSTMAN
* WorkBench

Pour comprendre ces technologies nous avons mis en place plusieurs concepts :

* Création d’une page simple avec l’affichage d’un JSON
* Création et lancement d’un conteneur DOCKER
* Création d’un projet avec Spring Initializar
* Connexion à une base de données en JDBC
* Connexion à une base de données en JPA
* Connexion à une base de données en JPA avec DTO
* Gestions de sessions
* Documentation avec Swagger
* Hébergement Java