133 – Réaliser des applications Web en Session-Handling

Rapport personnel

Version 1 du 15 avril 2025

Crée le 17.03.25

|  |  |
| --- | --- |
|  | Module du 17.03.25 au 15.04.25 |

Table des matières

[1 Introduction 3](#_Toc195608035)

[2 Tests technologiques selon les exercices 4](#_Toc195608036)

[2.1 Installation et Hello World 4](#_Toc195608037)

[2.2 Conteneurisation 4](#_Toc195608038)

[2.3 Création d'un projet Spring Boot 4](#_Toc195608039)

[2.4 Connexion à la DB JDBC 5](#_Toc195608040)

[2.5 Connexion à la DB JPA 5](#_Toc195608041)

[2.6 Connexion à la DB JPA avec DTO 5](#_Toc195608042)

[2.7 Gestion des sessions 6](#_Toc195608043)

[2.8 Documentation API avec Swagger 7](#_Toc195608044)

[2.9 Hébergement 8](#_Toc195608045)

[3 Auto-évaluations et conclusions 9](#_Toc195608046)

[3.1 Ce que j’ai appris, ce qui est important 9](#_Toc195608047)

[3.2 Ce que j’ai apprécié / pas apprécié 9](#_Toc195608048)

[3.3 Conclusion personnelle 9](#_Toc195608049)

[3.3.1 Points forts 9](#_Toc195608050)

[3.3.2 Points faibles 9](#_Toc195608051)

# Introduction

Dans ce module nous allons voir comment réaliser et tester une application Web par Session-Handling selon cahier des charges avec un langage de programmation. L’application Web sera faite sous forme de projet qui durera deux à trois semaines.

Cette application devra être sécurisée de diverses manières.

Voici les objectifs opérationnels de module :

* 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

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

Voici ce qu’il se passe dans les différentes étapes de l'exécution :

1. Initialisation du projet Spring Boot
2. Démarrage du serveur Tomcat intégré
3. Initialisation du contexte de l'application Spring
4. **Chargement des beans Spring**

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

**Build :** C’est la phase où le code source est compilé et assemblé en une application exécutable.

**Run :** Cela consiste à exécuter le fichier exécutable généré après le build.

* Y a-t-il un serveur web ?

Oui il y a un serveur web. Il est disponible sur le port 8080.

* Quelle version de java est utilisée ?

La version de java qui est utilisée est la 17.0.14.



* Si il y a un serveur web, quelle version utilise-t-il ?

Le serveur web utilisé est Apache Tomcat et sa version est 10.1.24.



## Conteneurisation

* Pourquoi faire un container pour une application Java ?

Un container permet de rendre l'application Java portable, isolée et facile à déployer sur différents environnements.

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

Oui, il y a un serveur Tomcat embarqué, disponible sur le port 8080.

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

Il faut s'assurer que les versions de Java, des dépendances, et de Tomcat sont compatibles entre elles et avec l'application.

## Création d'un projet Spring Boot

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

Il y a les annotations @RestController, @GetMapping, @RequestParam, @PostMapping, @RequestBody, @PutMapping, @RequestBody

## Connexion à la DB JDBC

* Trouvez le moyen de retourner l'ArrayList en format JSON.

Il n’y a pas de code à modifier pour retourner l’ArrayList au format JSON car c’est fait automatiquement par SpringBoot.

## Connexion à la DB JPA

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

L'annotation « @Autowired » permet à Spring d'injecter automatiquement une instance du repository dans le contrôleur.

Cela permet de faciliter la gestion des dépendances et de suivre le principe de l'injection de dépendances.

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

L'annotation « @ManyToOne » dans l'entité Skieur est utilisée pour définir une relation plusieurs-à-un entre l'entité Skieur et Pays.

Ce qui signifie qu’un pays peut avoir plusieurs skieurs, mais un skieur peut venir que d’un seul pays.

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

C’est le FetchType EAGER qui est utiliser car cela évite une requête supplémentaire, mais peut entraîner un surcoût en termes de performance, surtout si les entités associées sont nombreuses.

## Connexion à la DB JPA avec DTO

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

On retrouve un SkierDTO et pas un PaysDTO car il n'est pas nécessaire de récupérer les pays directement. Ils sont utilisés uniquement comme des attributs au sein des skieurs.

* A quoi servent les model, les repository, les dto, les services et les controlleurs ?

**Modèle :** représente les données (ex : Skieur, Pays)

**Repository :** accès à la base de données.

**DTO :** objet simplifié pour transférer des données.

**Service :** logique complexe (ex : création d’un utilisateur + envoi d’email). Il utilise les repositories et modèles.

**Controller :** gère les requêtes HTTP (GET, POST, etc). Appelle les services et renvoie des réponses (en JSON, HTML, etc).

## Gestion des sessions

Dans cet exercice, HttpSession est utilisé pour stocker les informations d’un utilisateur, comme son nom d'utilisateur et un compteur de visites.

Ces informations seront accessibles pendant toute la durée de sa session sur l'application. Ces données sont accessibles et modifiables à travers les différentes requêtes HTTP.

Pour tester le fonctionnement des sessions, il faut tester les méthodes à l’aide de Postman. Le fonctionnement espéré est que lors du login, une session, contenant deux attributs, un avec le nom d’utilisateur et un autre avec le nombre de visites, soit créée. Un message en cas de réussite est affiché.

La deuxième méthode permet de se déconnecter. Cette méthode supprime les attributs de la session actuelle et affiche un message en cas de réussite.

La troisième et dernière méthode permet d’afficher le nom de visites. À chaque fois que cette méthode est effectuée, le nombre de visites est augmenté.

Voici le résultat obtenu pour les trois méthodes sur Postman :

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

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

## Documentation API avec Swagger

La première chose à faire pour avoir accès à la documentation Swagger est d’ajouter ce code dans le « pom.xml ».

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.3.0</version>

</dependency>

Une fois que ceci est fait, il suffit de relancer l’application et de se rendre sur cette URL : <http://localhost:8080/swagger-ui/index.html>

Une fois sur cette page il est possible d’essayer les méthodes en cliquant sur « Try it out » et puis sur « Execute ».

## Hébergement

Pour héberger nos applications, il faut :

* Préparer l'image Docker : Utiliser des variables d'environnement dans le Dockerfile afin que le container utilise les bonnes valeurs.

ENV DATABASE\_URL=jdbc:mysql://host.docker.internal:3306/dbprojet

* Pousser l'image sur DockerHub. Pour ce faire il faut suivre ces différents points :
* Créer un compte DockerHub.
* Créer un repository public et générer un token.
* Se connecter sur VSCode avec docker login.

docker login -u username

* Tagger et pousser l'image avec docker tag et docker push.

docker tag nomImageLocale:latest username/nomRepo:latest

docker push username/nomRepo:latest

# Auto-évaluations et conclusions

## Ce que j’ai appris, ce qui est important

Durant ce module, j’ai appris à créer application Web par Session-Handling en utilisant Spring Boot.

## Ce que j’ai apprécié / pas apprécié

Personnellement je n’ai pas aimé le déroulement des premières semaines car ce n’est pas très intéressant d’effectuer des exercices puis répondre à des questions sans avoir de corrigés par la suite.

En revanche j’ai beaucoup aimé faire le projet en binôme.

## Conclusion personnelle

### Points forts

Mon point fort de ce module est principalement la programmation du backend.

### Points faibles

Mes points faibles durant le module 133 ont été Docker, car j’ai de la peine à comprendre le fonctionnement de la conteneurisation et la conception de certains schémas, comme le schéma de navigation.