Java Spring Boot avec H2

Définition

Java Spring Framework (Spring Framework) est une infrastructure open souce d'entreprise couramment utilisée qui permet de créer des applications autonomes de production qui fonctionnent sur la machine virtuelle Java (JVM).

Java Spring Boot (Spring Boot) est un outil qui accélère et simplifie le développement d'applications Web et de microservices avec Spring Framework grâce à trois fonctionnalités principales :

* Configuration automatique
* Approche directive de la configuration
* Possibilité de créer des applications autonomes

Ces fonctionnalités fonctionnent ensemble pour fournir un outil qui permet de configurer une application Spring avec une configuration et une installation minimale.

1 – Configuration minimale et autonome

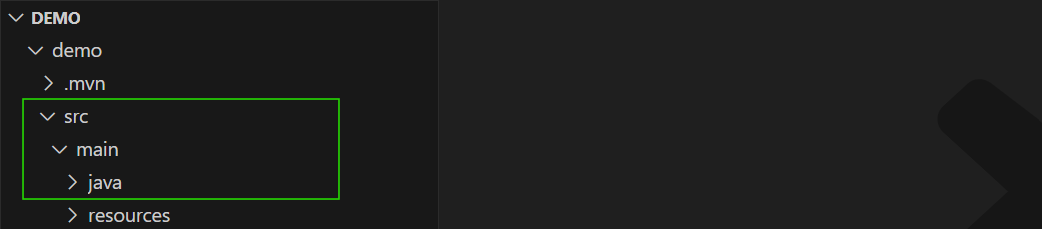
Sprint Boot Starter : Fournit des dépendances préconfigurées pour différents cas d’utilisation (spring-boot-starter-web), sprint-boot-starter-data-jpa)

Spring Boot Auto-Configuration : Configure automatiquement les beans selon des dépendances détectées.

Spring Boot Embedded Server : Tomcat, Jetty ou Undertow sont intégrés par défaut, évitant la configuration manuelle.

2 – Structure d’un projet Spring Boot

src/main/java : chemin du code source principal.

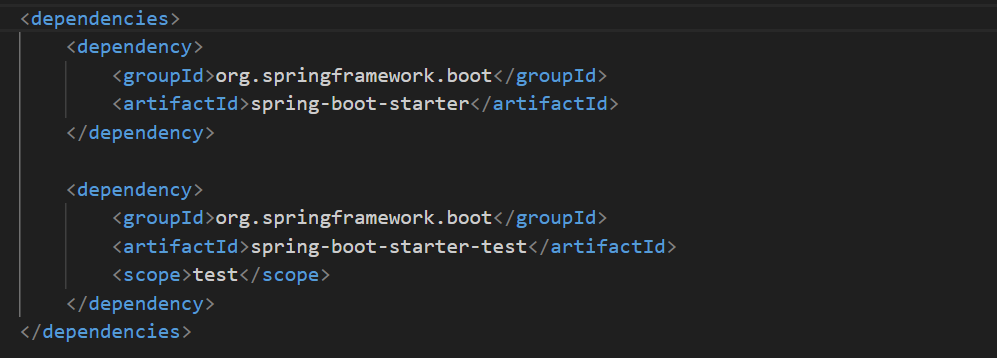


src/main/resources : chemin qui contient les fichiers de configuration (application.properties ou application.yml) dont le paramétrage de la base de données.



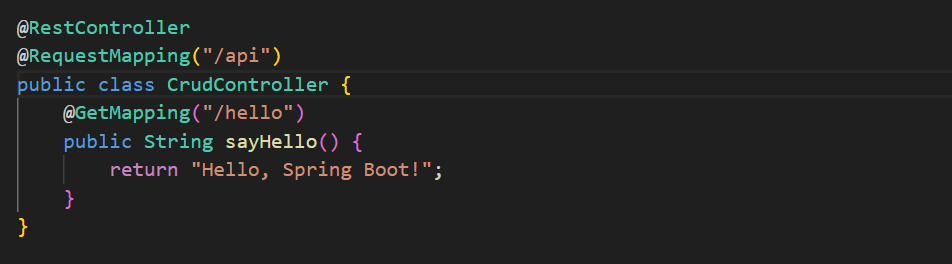
3 – Gestion des dépendances avec Maven ou Gradle

Maven (pom.xml) :



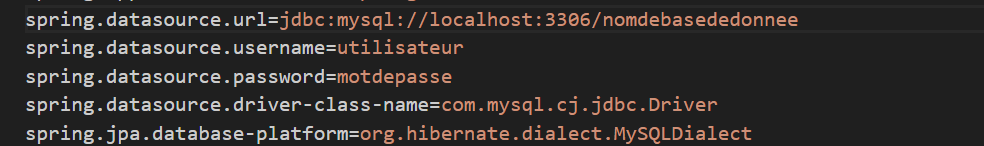
4 – Contrôleurs REST avec Spring Boot

Spring Boot permet de créer facilement des API REST

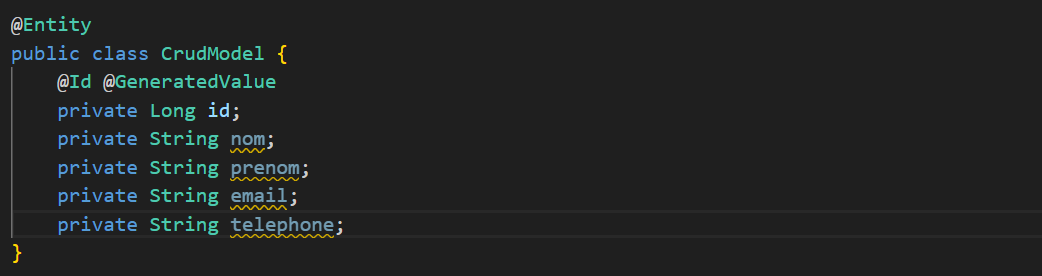


5 – Gestion des bases de données avec Spring Boot

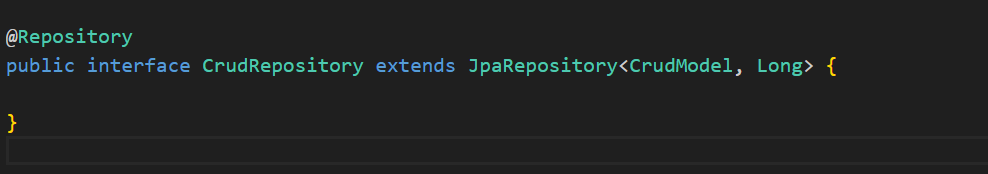
Spring Data JPA simplifie la communication avec une base de données relationnelle, utilise Hibernate par défaut et il nécessite une configuration dans application.properties :



Entité JPA :

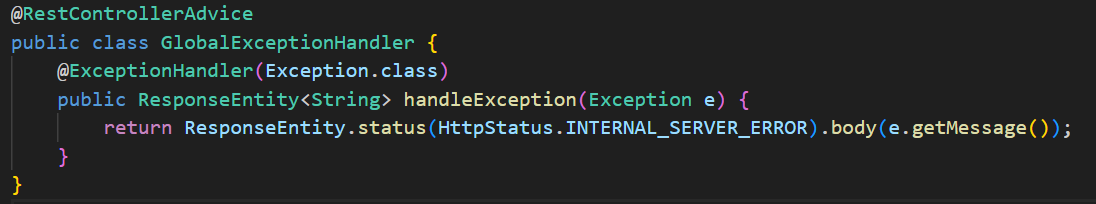


Interface Repository :

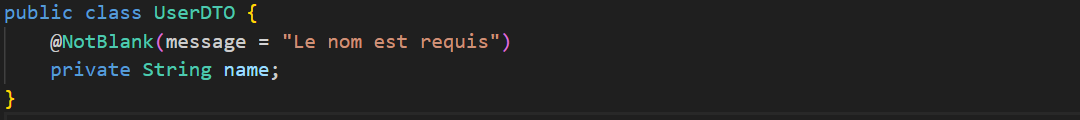


6 – Gestion des erreurs et validations

Spring boot permet une gestion simplifiée des erreurs avec @ExceptionHandler :



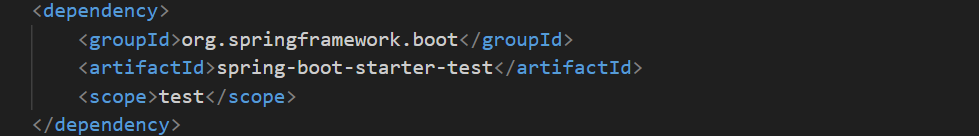
La validation des entrées avec @Valid :



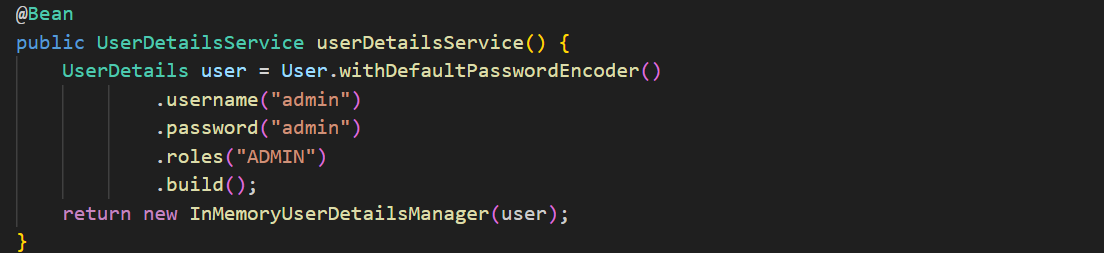
7 – Sécurité avec Spring Security

Spring boot intègre Spring Security pour l’authentification et l’autorisation.

Ajout de la dépendance :



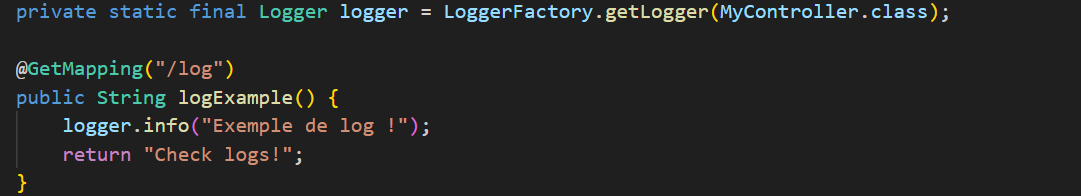
Configuration d'un utilisateur en mémoire :



8 – Gestion des logs

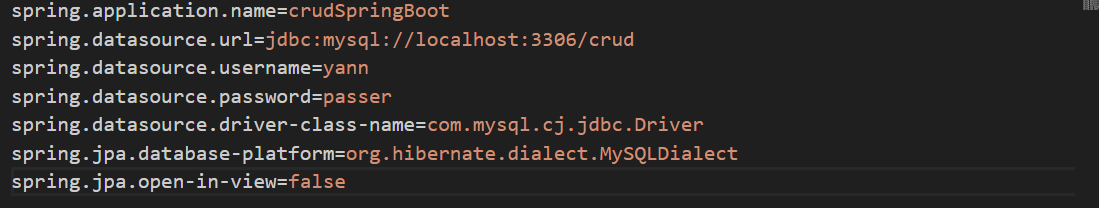
Spring Boot utilise SLF4J + Logback par défaut.

Utilisation :



9 – Fichiers de configuration (application.properties ou application.yml)

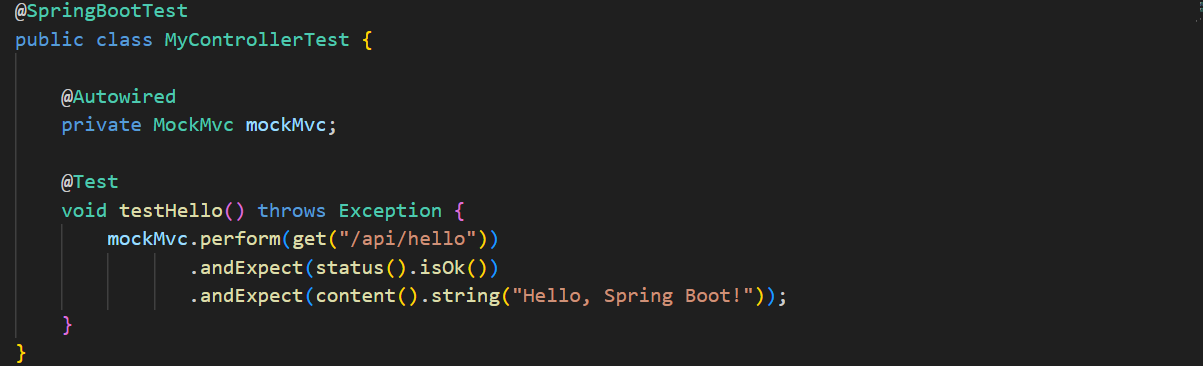
application.properties :



10 – Tests dans Spring Boot

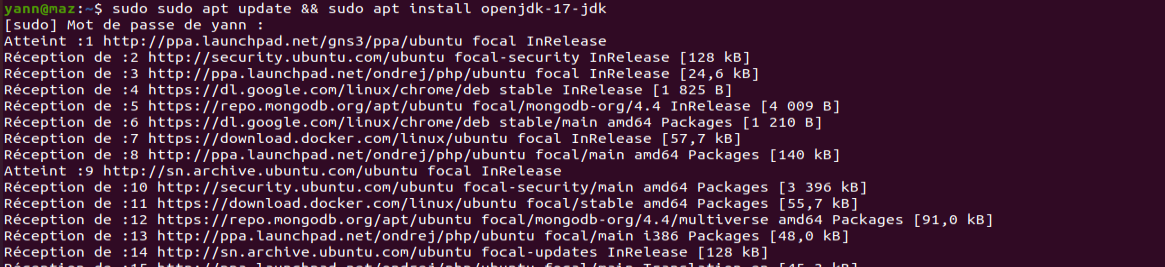
Spring Boot propose des outils de test avec JUnit et Mockito.

Cas d’utilisation :

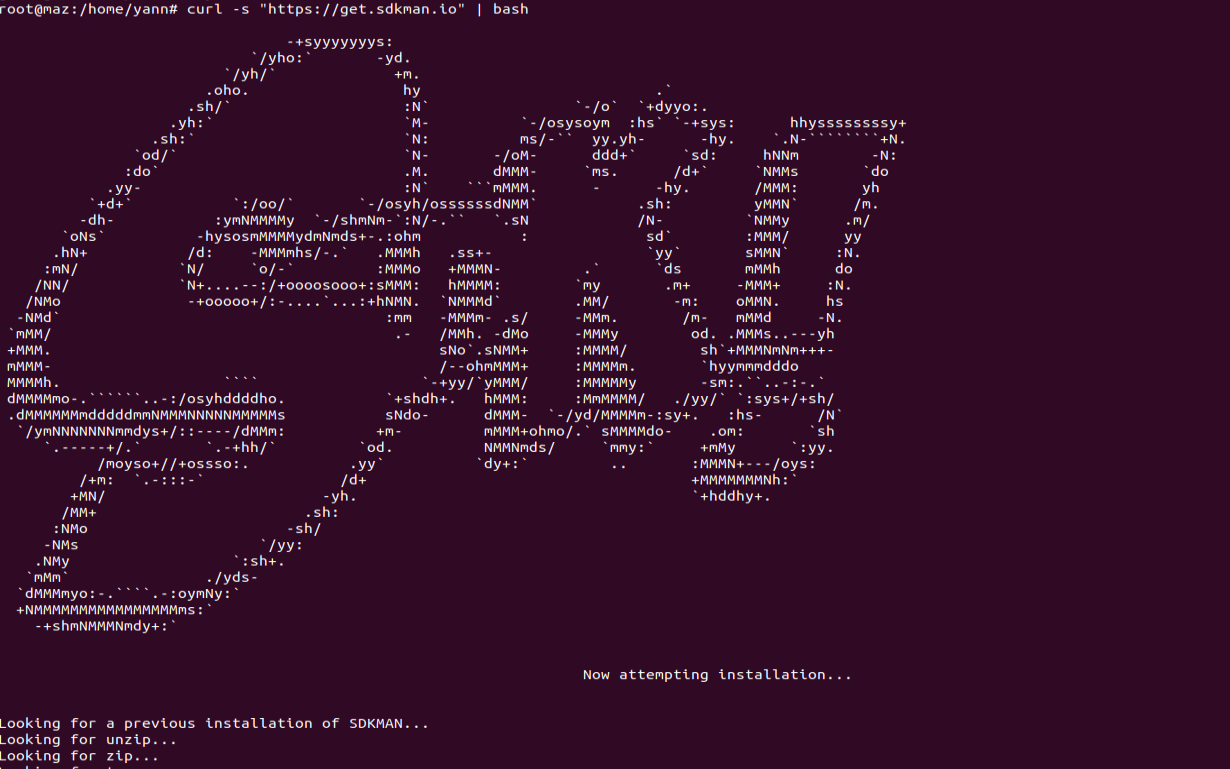


11 – Création de projet Spring Boot en ligne de commande

On fait la mise à jour et on installe les prérequis : apt update && sudo apt install openjdk-17-jdk



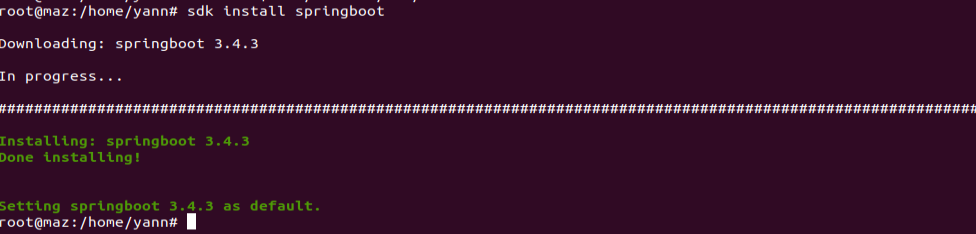
On télécharge le paquet sdkman : curl -s “https://get.sdkman.io’’ | bash



On initialise le sdkman en exécutant la commande : source “$HOME/.sdkman/bin/sdkman-init.sh’’



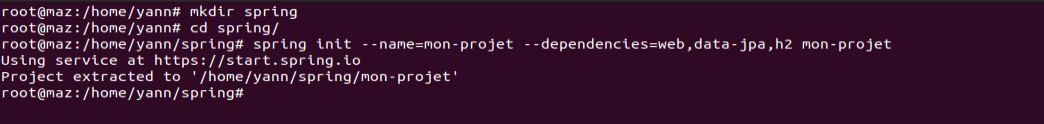
On installe spring boot avec la commande Java sdk : sdk install springboot



On regarde la version de spring boot avec la commande : spring --version



On crée un dossier dans lequel on va créer tous nos projets spring boot



On liste le dossier spring dans lequel on a créé notre projet



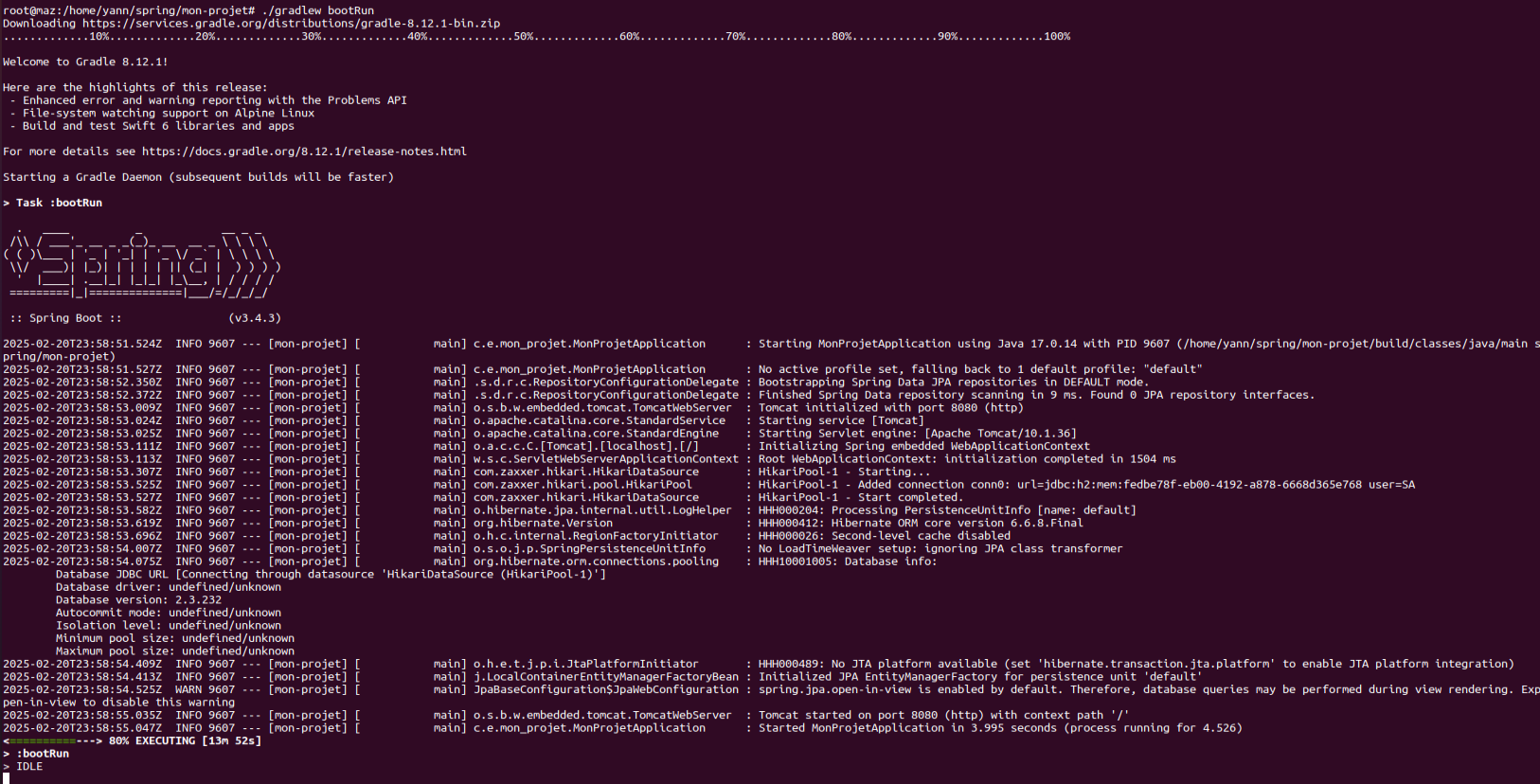
On installe l’outil maven pour avoir les commandes mvn : apt install maven



On liste le projet pour voir le contenu

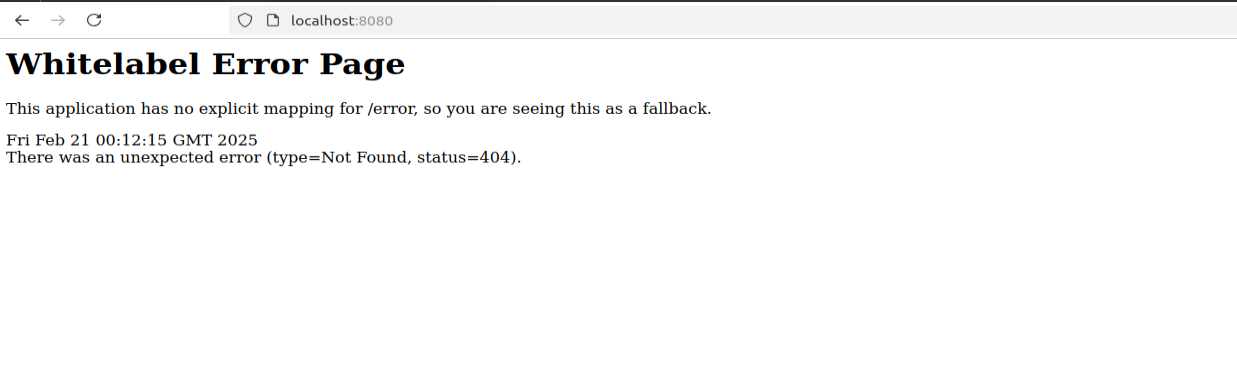


On lance le script pour démarrer le projet : ./gradlew bootRun



On teste sur le navigateur <http://localhost:8080/>

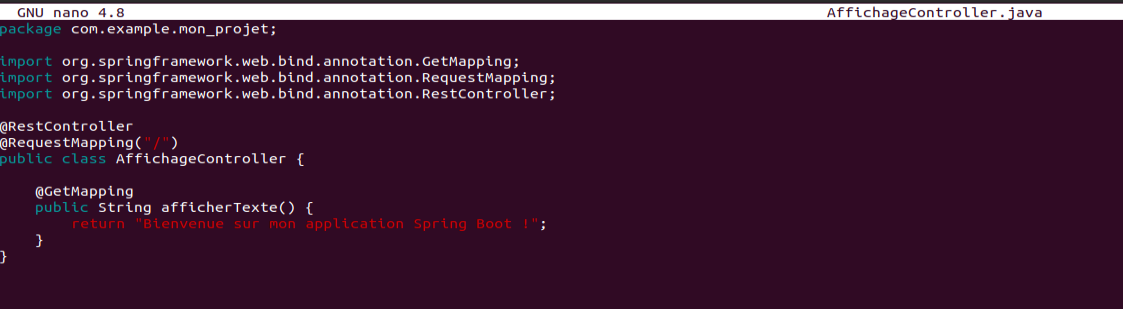
A noter que Spring boot écoute sur le port 8080



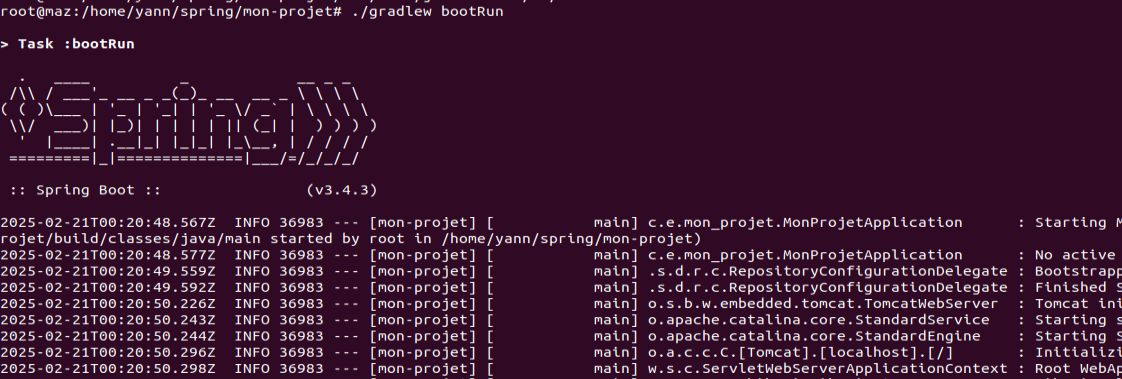
Normal que ce message apparait ci-dessus, c’est juste parce qu’on n’a pas encore créé des routes pour les ressources

Nous allons créer un petit programme qui va afficher un petit texte de bonjour pour voir la différence.

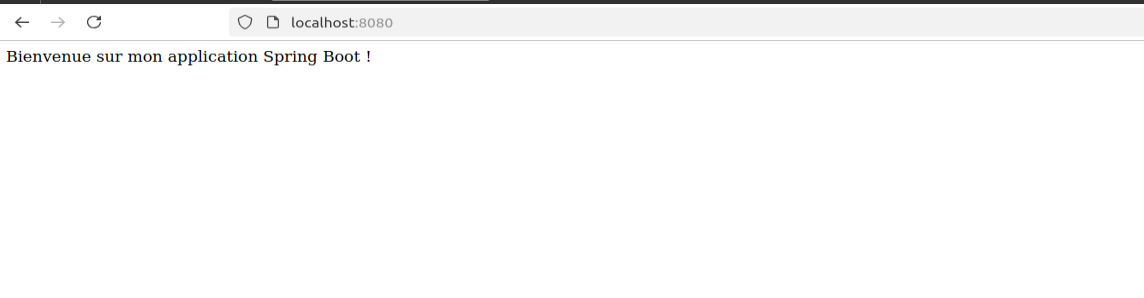
On va donner comme nom AffichageController.java



On lance le projet

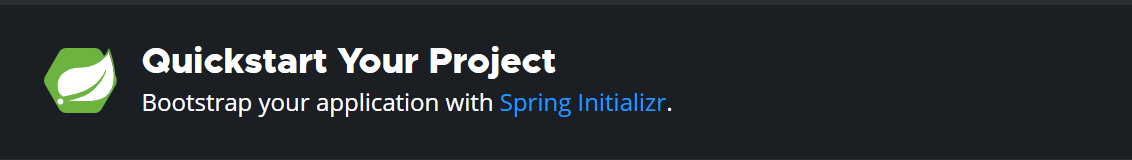


Le résultat sur le navigateur



12 – Création d’un projet Spring boot avec Spring initializr

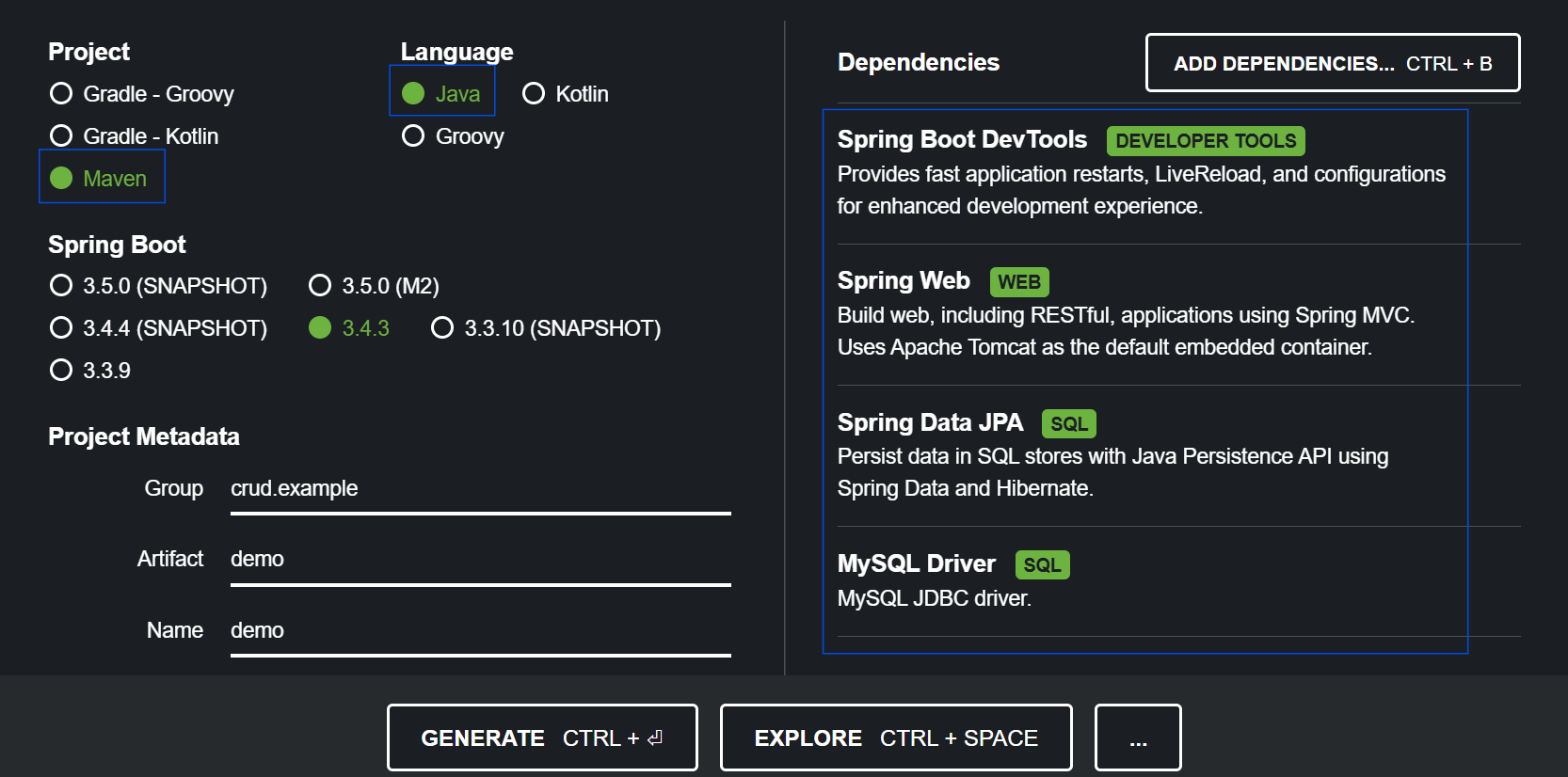
Pour créer un projet avec Spring Initializr il faut se rendre sur le site officiel de spring boot



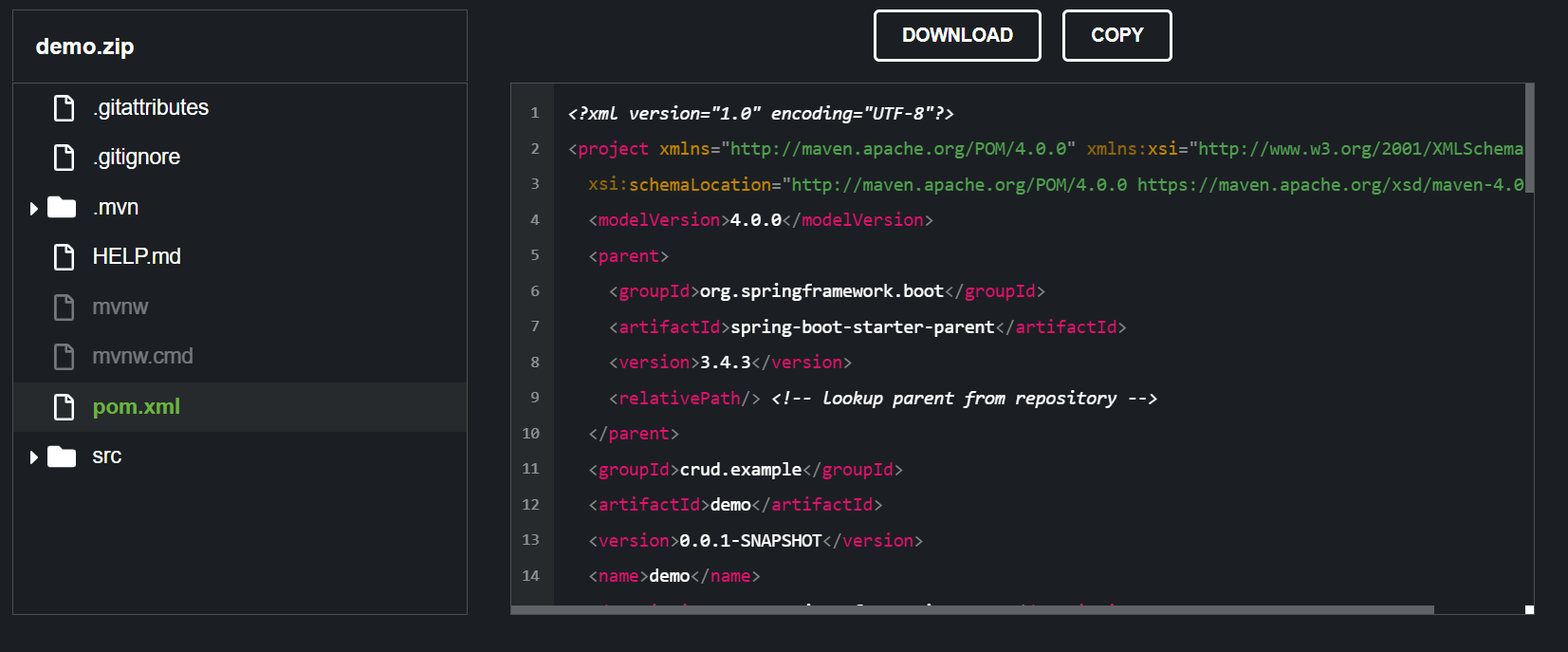
Installation des dépendances, on clique sur le bouton « ADD DEPENDENCIES… CTRL + B »



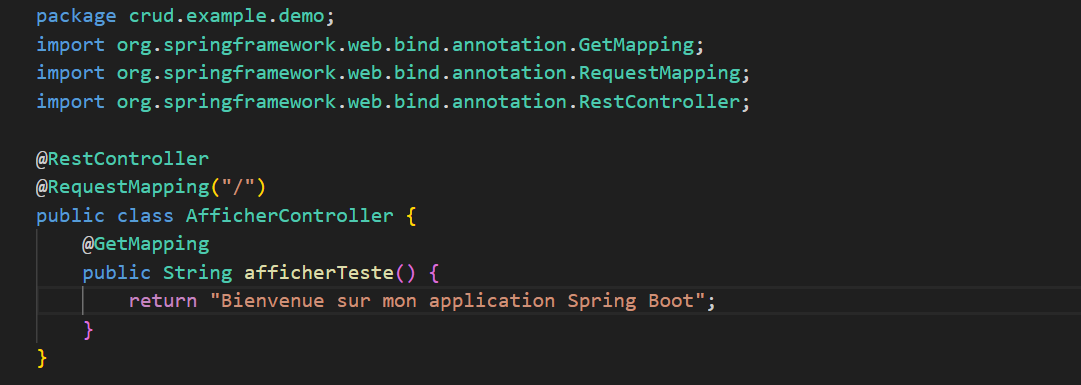
On peut choisir toutefois la version de java (17)

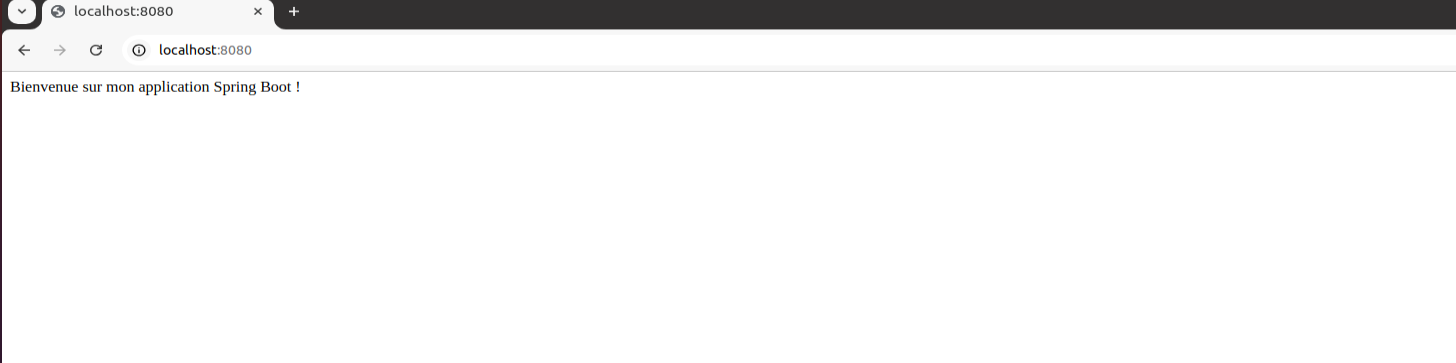


On peut cliquer sur Explore pour voir la démo ensuite soit télécharger soit modifier le projet



On teste le projet





13 – La différence entre Spring boot et un autre framework backend (Flask)

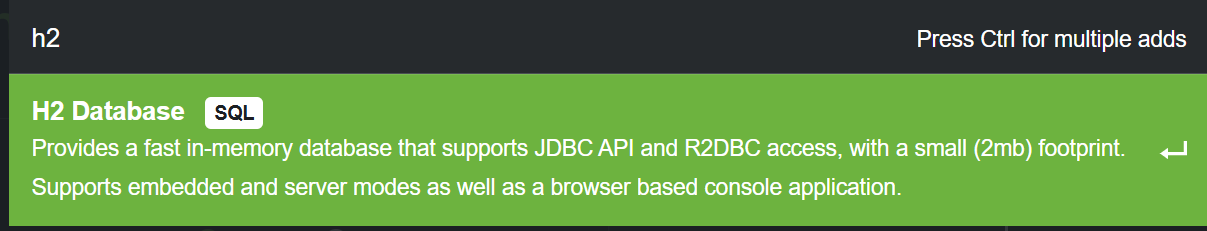
Ici ci-dessous, nous allons faire une comparaison entre un autre framework backend (Flask) et Spring Boot

|  |  |  |
| --- | --- | --- |
| Critères | Spring boot | Flask |
| Langage | |  | | --- | | Java/Kotlin |  |  | | --- | |  | | Python |
| Niveau de complexité | Complexe | Plus léger |
| Performances | Très performant, optimisé pour les grandes applications | Rapide mais moins optimisé pour des applications très lourdes |
| Ecosystème | Spring, Hibernate | Moins vaste mais modulaire |
| Utilisation | Application d’entreprise et microservice | APIs rapides, prototypage |

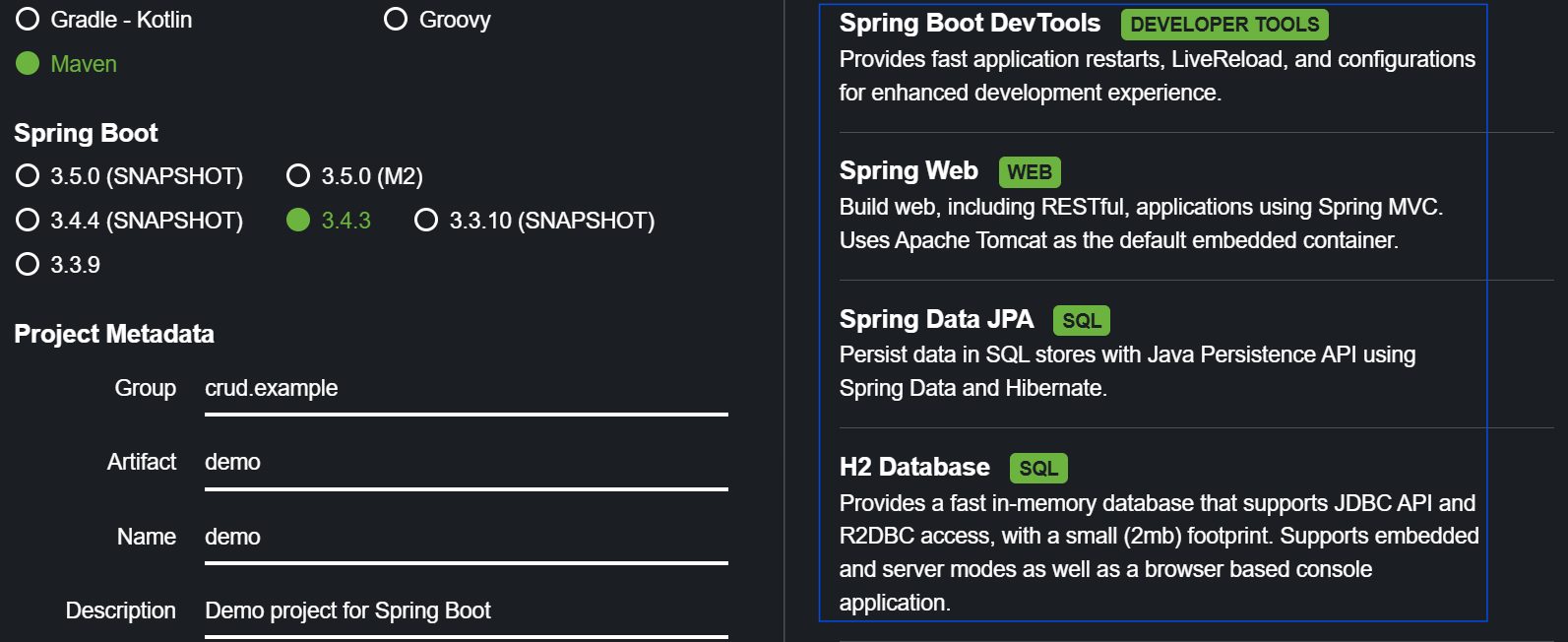
14 – Mise en place d’un projet Spring Boot avec H2

H2 est une base données rapide et légère qui peut être persistée dans un fichier ou non. Elle est open source et en mémoire. Il s’agit d’un système de gestion de base de données relationnelle écrit en Java. Il s’agit d’une application client/serveur. Elle stocke les données en mémoire, sans les conserver sur le disque.

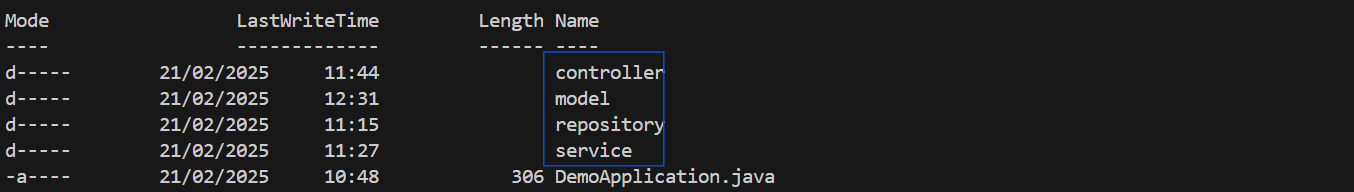
On se rend sur Spring Initialzr pour créer un projet ainsi installer les dépendances.



Installons toutes les dépendances du projet

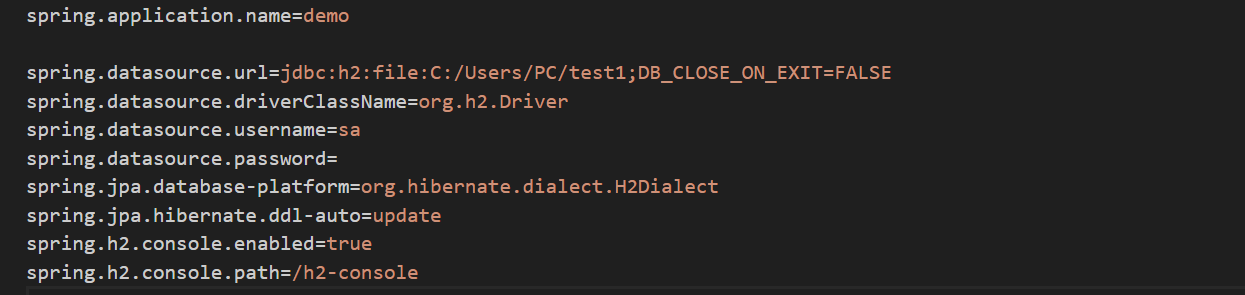


* On crée nos différents programmes avec l’architecture MVC (Model View Controller)
* Le contrôleur récupère la requête et appelle un service.
* Le service applique la logique métier et appelle un repository.
* Le repository interagit avec la base de données via Sequelize.
* Le modèle représente les données stockées et permet l’accès à la base.



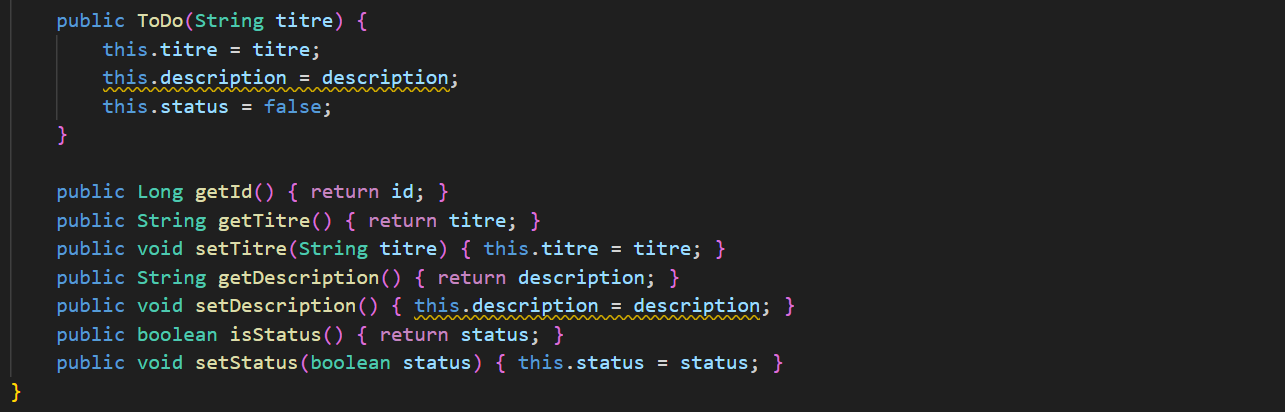
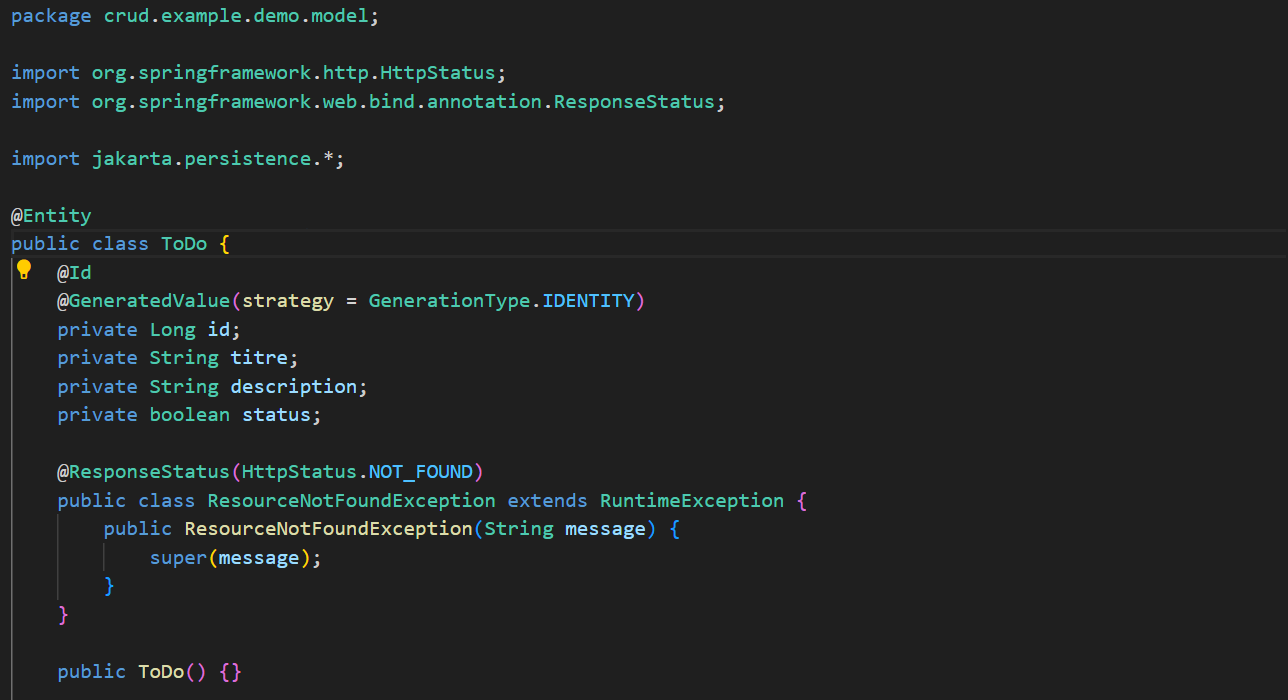
On paramètre la connexion avec notre base de données que ça soit H2, MySQL, PostgreSQL ou mongoDB :

application.properties



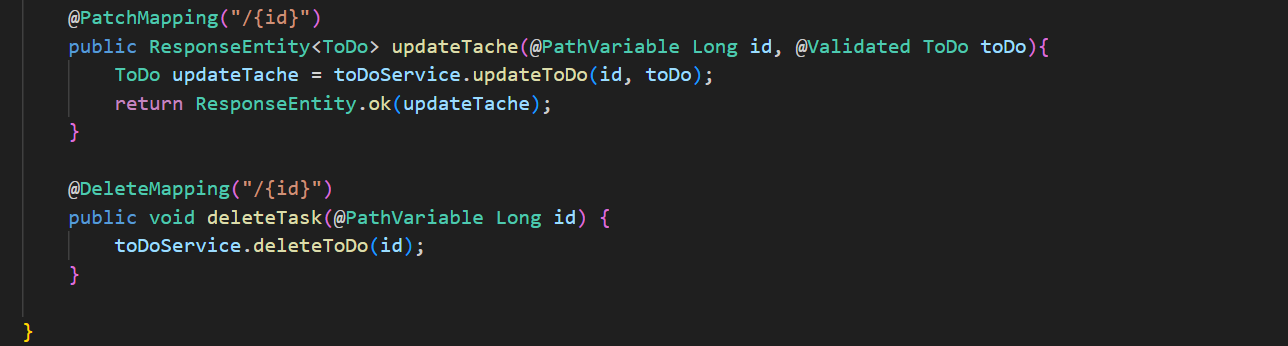
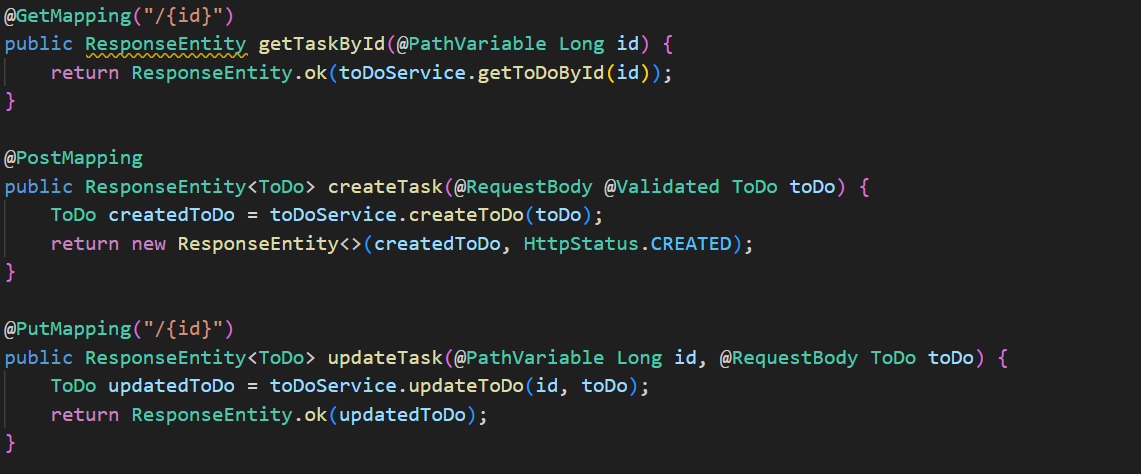
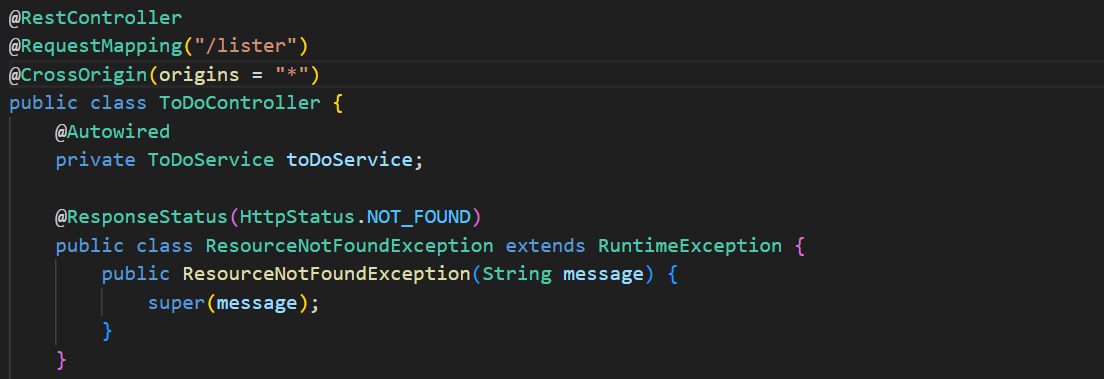
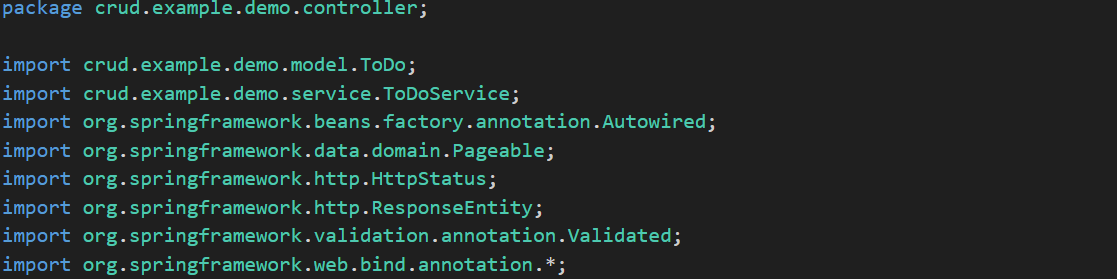
**Le model**

A pour rôle de gérer et d’interagir avec la base de données



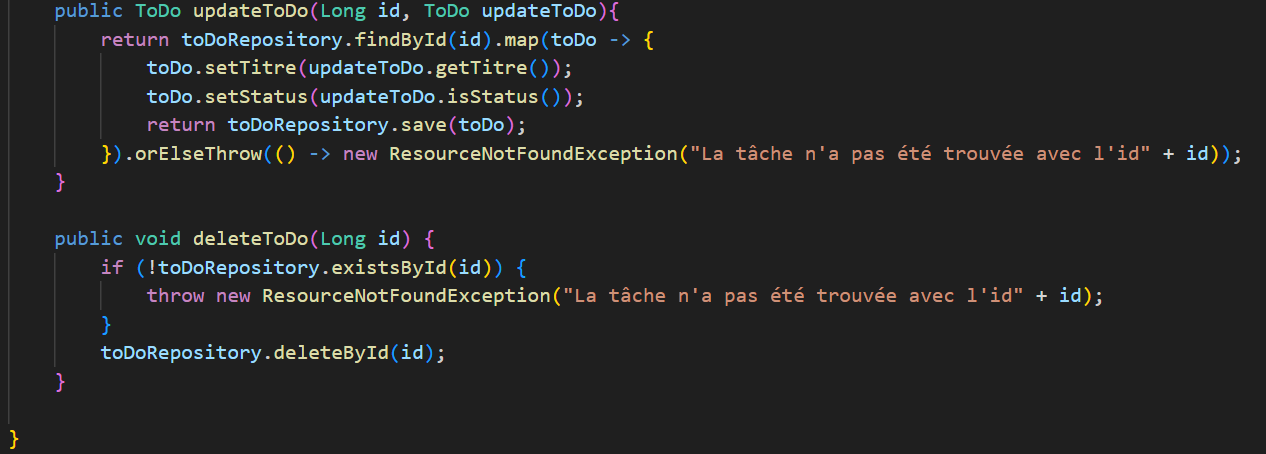
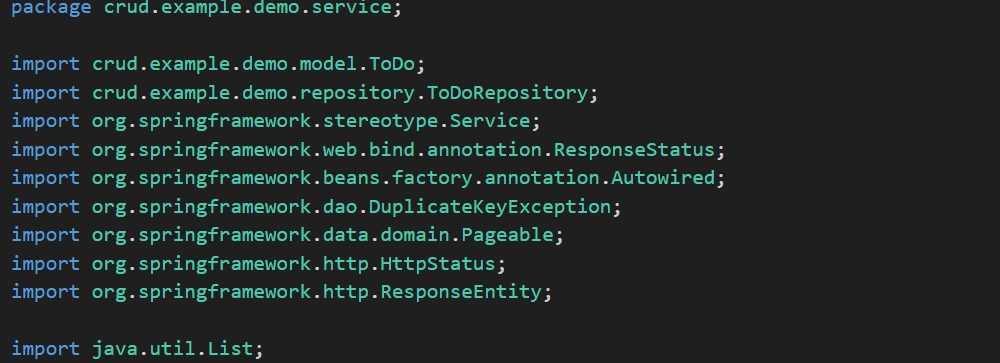
**Le Controller**

A pour rôle de récupérer les données de la requête, d’appeler un service et envoyer la réponse.



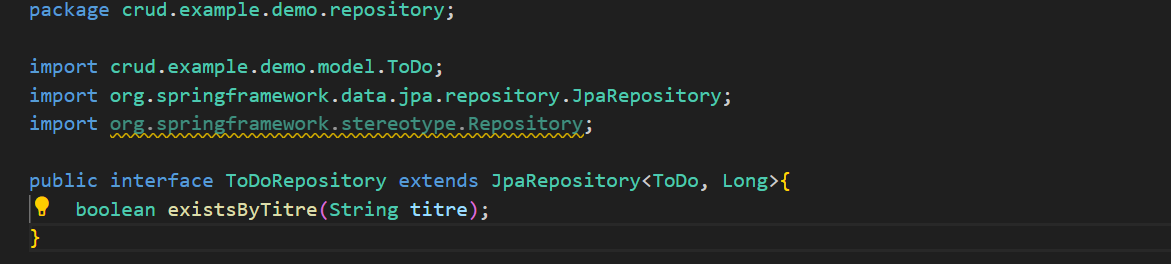
**Le service**

Permet de gérer la logique métier. Il vérifie et traite les données avant l’enregistrement et appelle le repository pour manipuler la base.

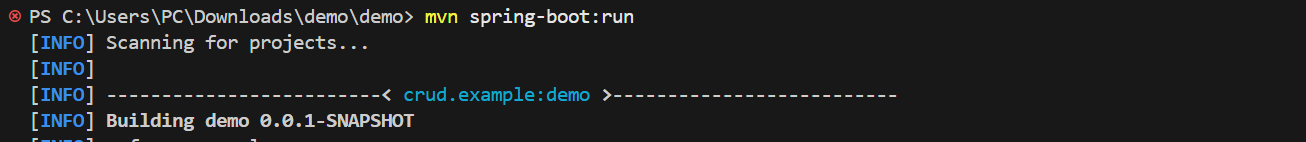


**Le Repository**

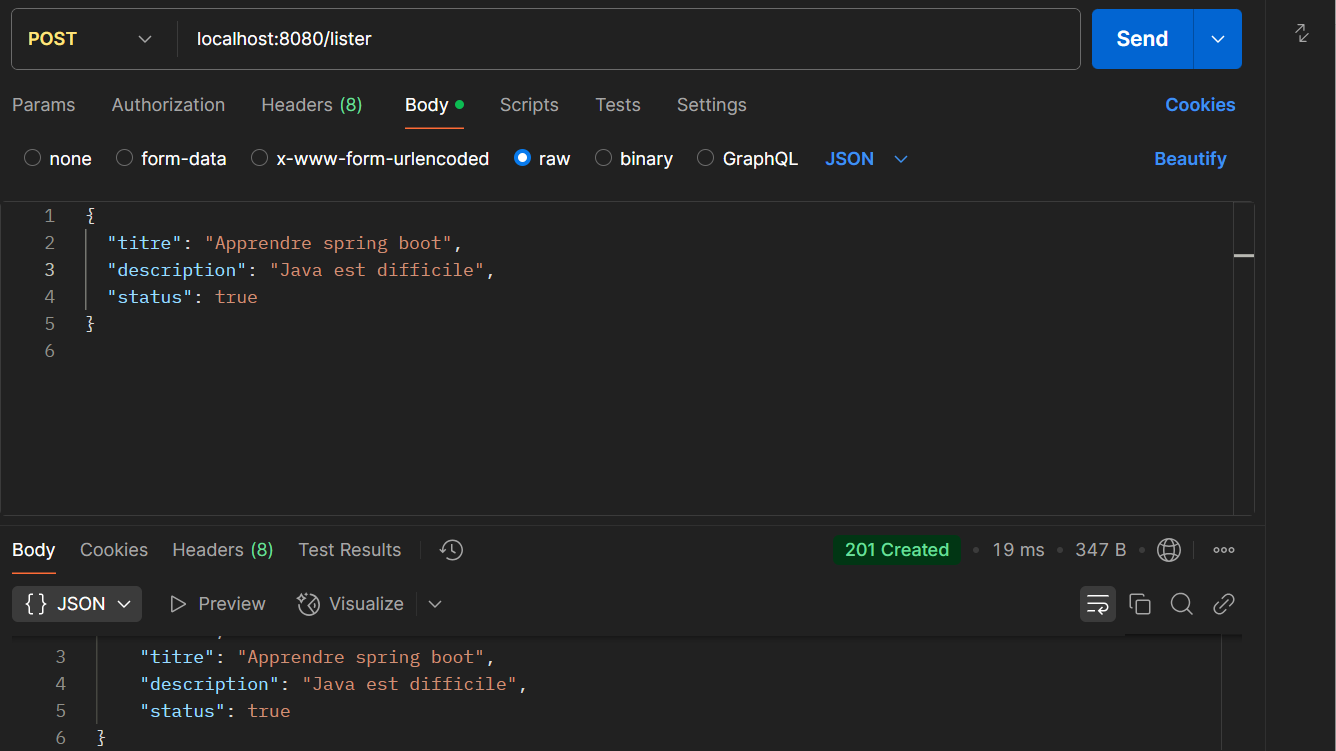
Interagit avec la base de données via Sequelize.



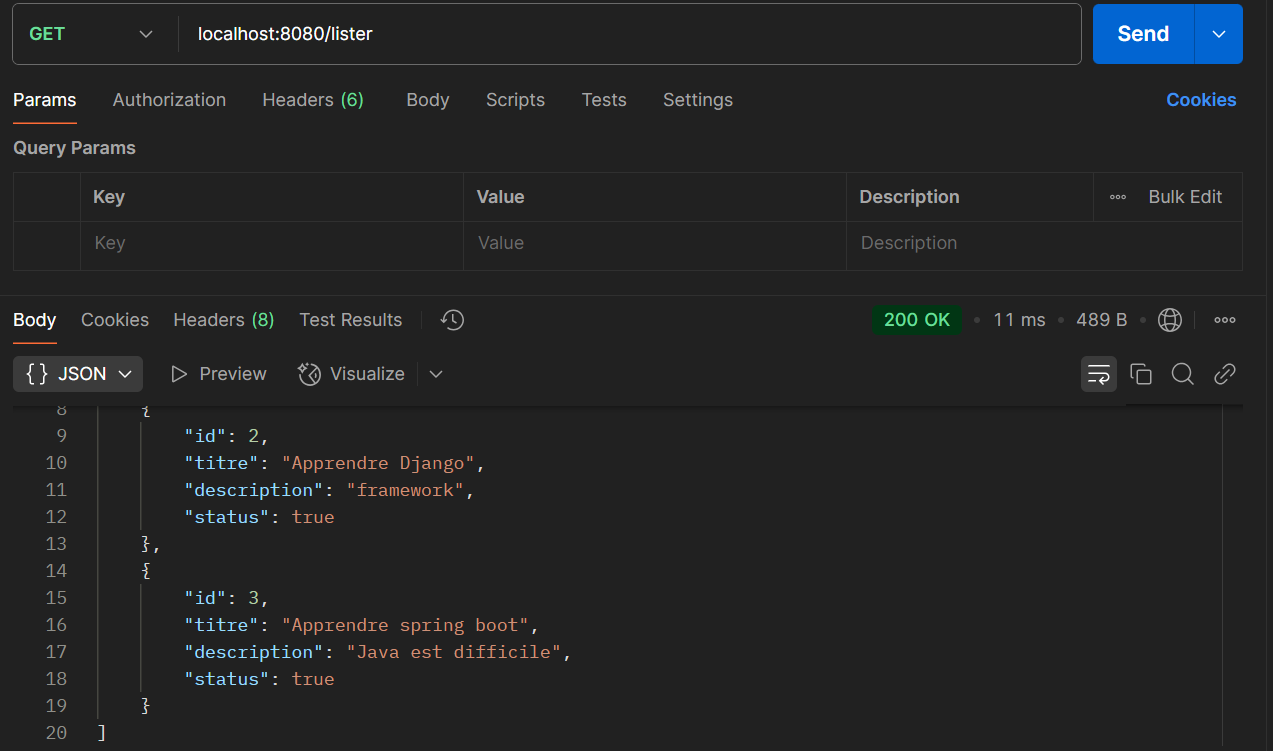
On teste le programme en démarrant l’application : mvn spring-boot:run



On teste le programme avec la méthode POST avec postman



Nous pouvons voir notre api fonctionne



H2-console

Avec spring-boot nous avons la possibilité d’exécuter les requêtes dans une console appélée H2, pour se faire on va activer la console dans application.properties



On tape l’adresse suivante : <http://localhost:8080/h2-console> pour se connecter à la console

