Projet 01 - « Greeting »

TABLE DES MATIÈRES

Contents

[1 - Rappel rapide 3](#_heading=h.gjdgxs)

[1.1- Starter 3](#_heading=h.30j0zll)

[1.2. Dépendances 4](#_heading=h.1fob9te)

[1.3. Importer pom.xml 5](#_heading=h.3znysh7)

[1.4. Créez la classe Greeting.java 8](#_heading=h.2et92p0)

[1.5. Créez la classe GreetingController.java 9](#_heading=h.tyjcwt)

[1.6. Observez et testez 9](#_heading=h.3dy6vkm)

[Résolution de problèmes #1 12](#_heading=h.1t3h5sf)

[Résolution de problèmes #2 13](#_heading=h.4d34og8)

[2 - Ajout de la base de données 15](#_heading=h.2s8eyo1)

[2.1. Ajoutez les dépendances 15](#_heading=h.17dp8vu)

[2.2. Configurez les propriétés (application.properties) 16](#_heading=h.3rdcrjn)

[2.3. annotation du modèle ou du bean Greeting.java 16](#_heading=h.26in1rg)

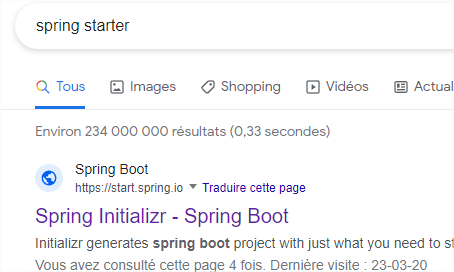
[2.4 (D) s’assurer que vous avez les getters et les setters et les constructeurs dans Greeting.java 17](#_heading=h.lnxbz9)

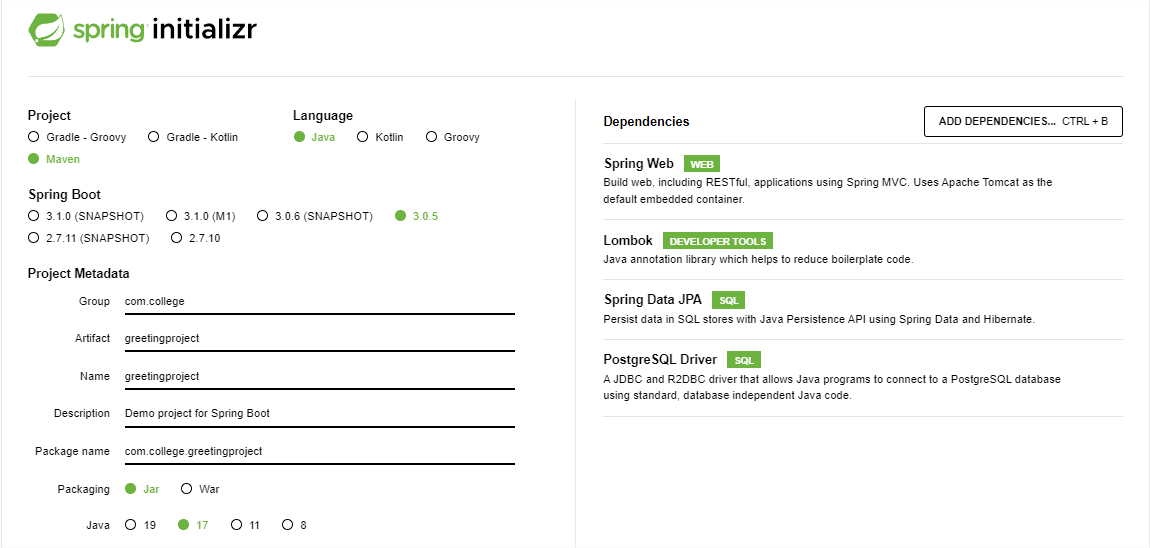
[2.5. Création de la base de données haythem dans PostgreSQL avec les bons privilégies hrgres par exemple 18](#_heading=h.35nkun2)

[2.6. Création de la base de données uniquement avec l’utilisateur hrgres (par exemple) et qui a tous les privilégies 18](#_heading=h.1ksv4uv)

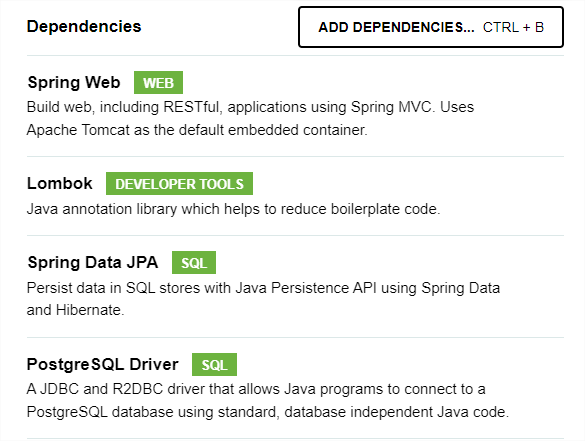
# 1 – Créer et importer le projet

## 1.1- Starter ou Spring Initializr

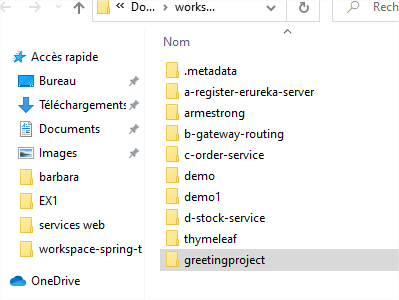




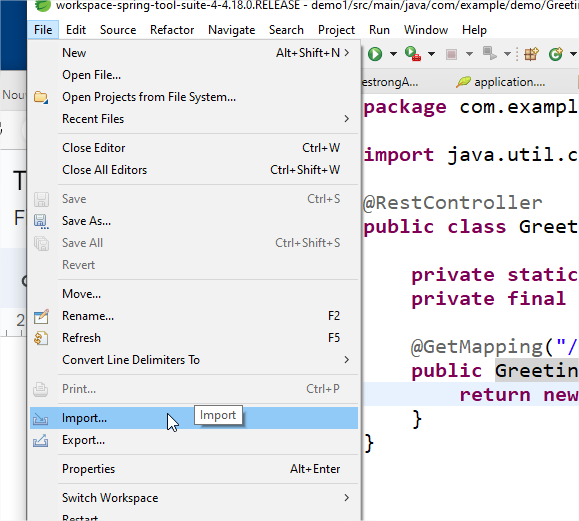
## 1.2. Ajout des dépendances

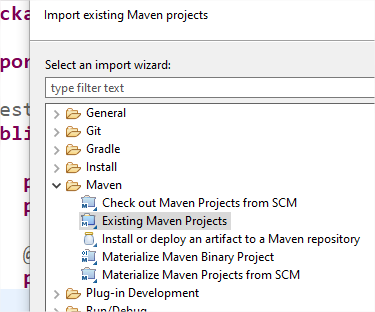


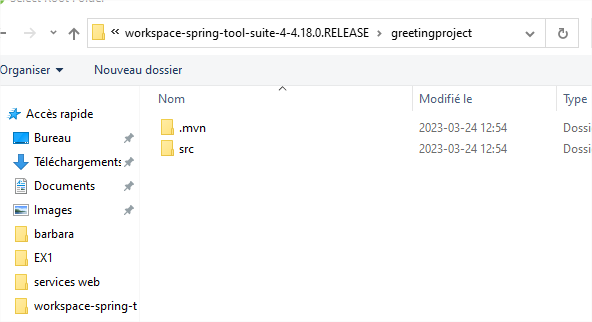
1.3. Le déplacer dans le Workspace

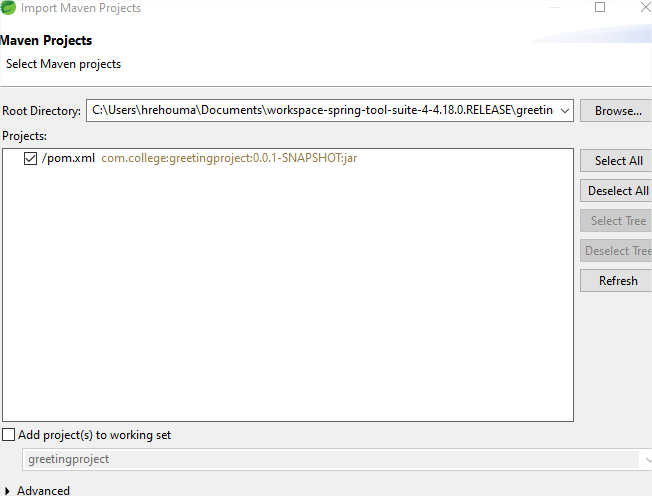


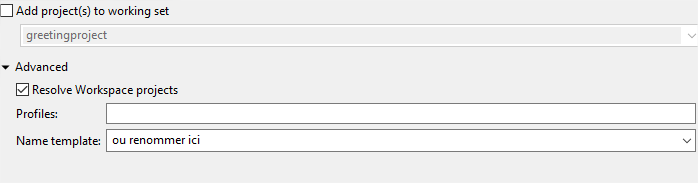
## 1.3. Ouvrir à STS, effectuer un import de type « Maven Projects », et sélection de pom.xml

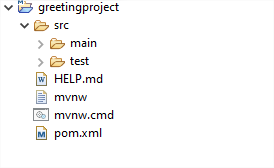


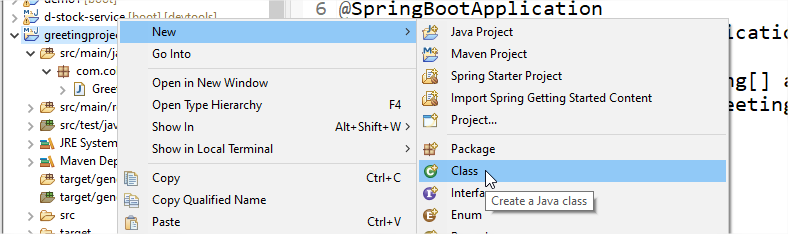












## 1.4. Créez la classe Greeting.java

**package** com.example.demo;

**public** **class** Greeting {

**private** Long id;

**private** String content;

**public** Greeting(Long id, String content) {

**super**();

**this**.id = id;

**this**.content = content;

}

**public** Long getId() {

**return** id;

}

**public** String getContent() {

**return** content;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** **void** setContent(String content) {

**this**.content = content;

}

}

## 1.5. Créez la classe GreetingController.java

package com.example.demo;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class GreetingController {

private static final String template = "Hello, %s!";

private final AtomicLong counter = new AtomicLong();

@GetMapping("/greeting")

public Greeting greeting(@RequestParam(value="name") String name) {

return new Greeting(counter.incrementAndGet(), String.format(template, name));

}

}

## 1.6. Observez et testez

* Observons-les dépendances dans pom.xml (lister les artéfacts)

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**Exercice :**

Listez les dépendances dans le pom.xml et indiquez le rôle de chacune de ces dépendances.

**Réponse :**

1. Starter-data-jpa
2. spring-boot-starter-web
3. org.postgresql
4. org.projectlombok
5. org.springframework.boot

**TESTEZ :**

Toujours, « run as » :

1. Maven clean.
2. Maven install.
3. Spring Boot app

Graphical user interface, application

Description automatically generated

**OUPS !!!! ERREUR**

Graphical user interface, text, application

Description automatically generated

## 1.7. Résolution d’erreurs

* Observons-les dépendances dans pom.xml (lister les artéfacts)
* En effet, nous avons le message suivant :

Graphical user interface, text, application

Description automatically generated

### 1.7.1. Résolution de problèmes #1

La raison est le fait que nous avons ajouté une dépendance que nous n’avons pas utilisé. Observons notre pom.xml !!

|  |  |
| --- | --- |
| **Dépendances à supprimer de votre pom.xml** | **Dépendances à garder dans votre pom.xml** |
| postgresql  Text  Description automatically generated | starter-web |
| JPA  Graphical user interface, text, application  Description automatically generated | starter-test |
| lombok  Graphical user interface, text, application, chat or text message  Description automatically generated |  |

*🡺 Les dépendances supprimées sont les suivantes :*

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

*Les dépendances que nous avons gardées sont les suivantes :*

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

*Supprimez les dépendances inutilisées et retestez !!! ça marche !!!!!!!!!!!!!*

**TESTEZ :**

Toujours, « run as » :

1. Maven clean.
2. Maven install.
3. Spring Boot app
4. Saisir dans le navigateur le lien suivant :

<http://localhost:8080/greeting?name=armestrong>



1. Changez le point de terminaison à ?name=français
2. Changez le point de terminaison à ?name=patrick
3. Rafraîchir la page.
4. Réexécutez le projet sans le fermer !

**OUPS !!!! ERREUR**

### 1.7.2. Résolution de problèmes #2

*Nous pouvons avoir le port 8080 occupé. Dans ce cas, deux solutions se présentent :*

* *Solution 1 : Changer de port dans le fichier application.properties*
* *Solution 2 : Kill the process utilisant le port 8080*

**Solution 1 :** Changer de port dans le fichier *application.properties*

* *Si le format est application.properties, utilisez la ligne suivante :*

server.port=8081

* *Si le format est application.yml, utilisez la ligne suivante :*

server :

Port : 8081

**Source et références :**

Graphical user interface, text, application, email

Description automatically generated

# Server configuration dans yaml

server:

port: 9090

Source : <https://www.baeldung.com/spring-boot-change-port>

**Solution 2 :** Terminer le processus (Kill the process) utilisant le port 8080

**netstat -noa |findstr :8080**

-**a** displays all connections and listening ports.

-**o** displays the owning process ID associated with each connection.

-**n** displays addresses and port numbers in numerical form.

**Arrêter task via son PID:**

**taskkill /F /PID pid\_number**

**demo :**

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

**Si ça ne marche pas, exécutez la ligne de commande «CMD » en tant qu’administrateur**