

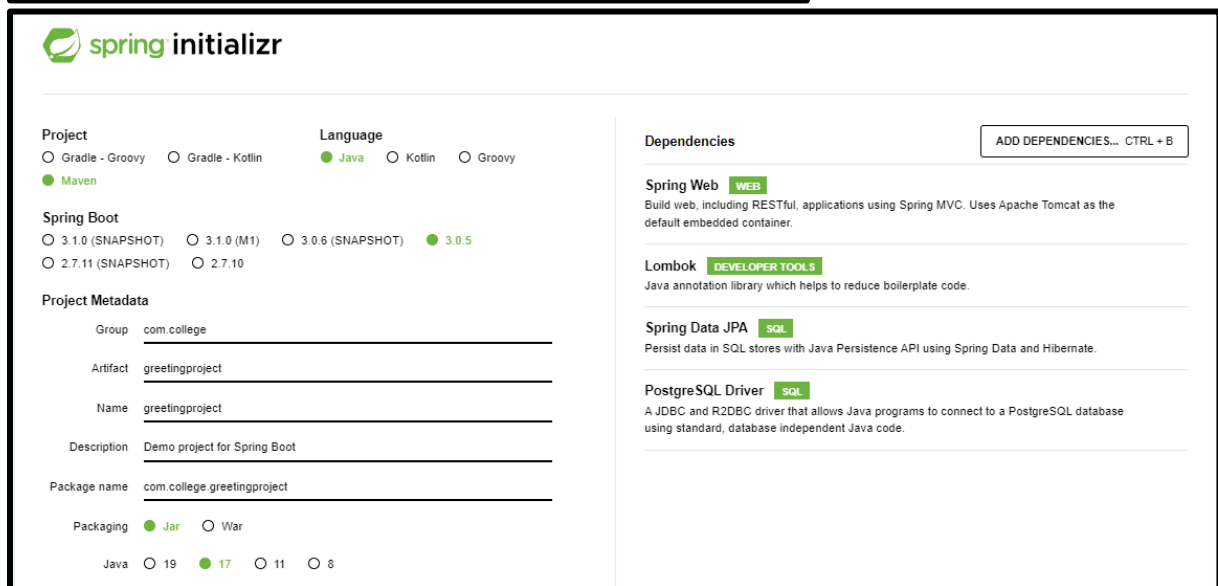
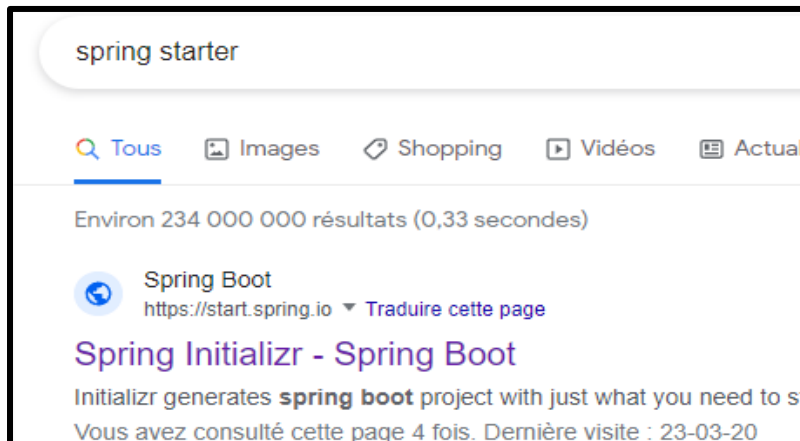
Projet 01 - « Greeting »

TABLE DES MATIÈRES

1 – Créer et importer le projet	2
1.1- Starter ou Spring Initializr	2
1.2. Ajout des dépendances	3
1.3. Ouvrir à STS, effectuer un import de type « Maven Projects », et sélection de pom.xml	4
1.4. Créez la classe Greeting.java	6
1.5. Créez la classe GreetingController.java	7
1.6. Observez et testez	7
1.7. Résolution d’erreurs	9
1.7.1. Résolution de problèmes #1	9
1.7.2. Résolution de problèmes #2	12

1 – Créer et importer le projet

1.1- Starter ou Spring Initializr



A screenshot of the Spring Initializr web application. The page has a green header with the "spring initializr" logo. The main content area is divided into two columns. The left column contains settings for the project, and the right column lists dependencies.

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Java** ☐ Kotlin ☐ Groovy

☒ **Maven**

Spring Boot

☐ 3.1.0 (SNAPSHOT) ☐ 3.1.0 (M1) ☐ 3.0.6 (SNAPSHOT) ☒ **3.0.5**

☐ 2.7.11 (SNAPSHOT) ☐ 2.7.10

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ **Jar** ☐ War

Java ☐ 19 ☒ **17** ☐ 11 ☐ 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

PostgreSQL Driver SQL
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

1.2. Ajout des dépendances

Dependencies **ADD DEPENDENCIES... CTRL + B**

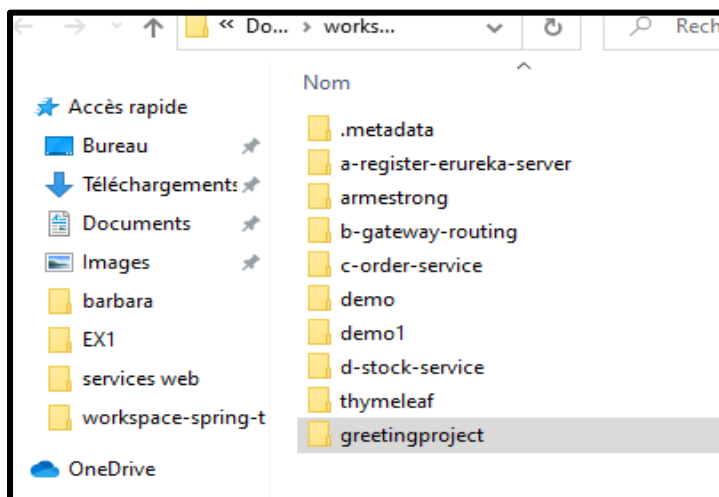
Spring Web **WEB**
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Lombok **DEVELOPER TOOLS**
Java annotation library which helps to reduce boilerplate code.

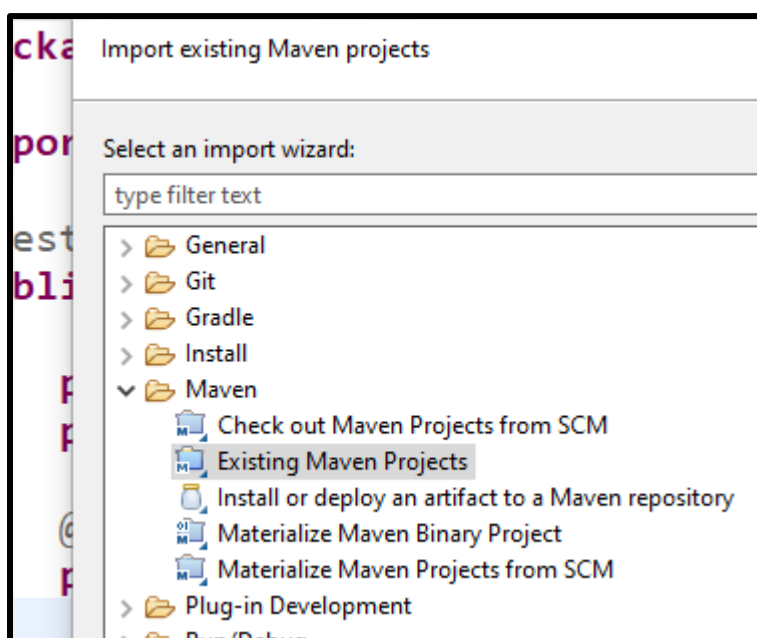
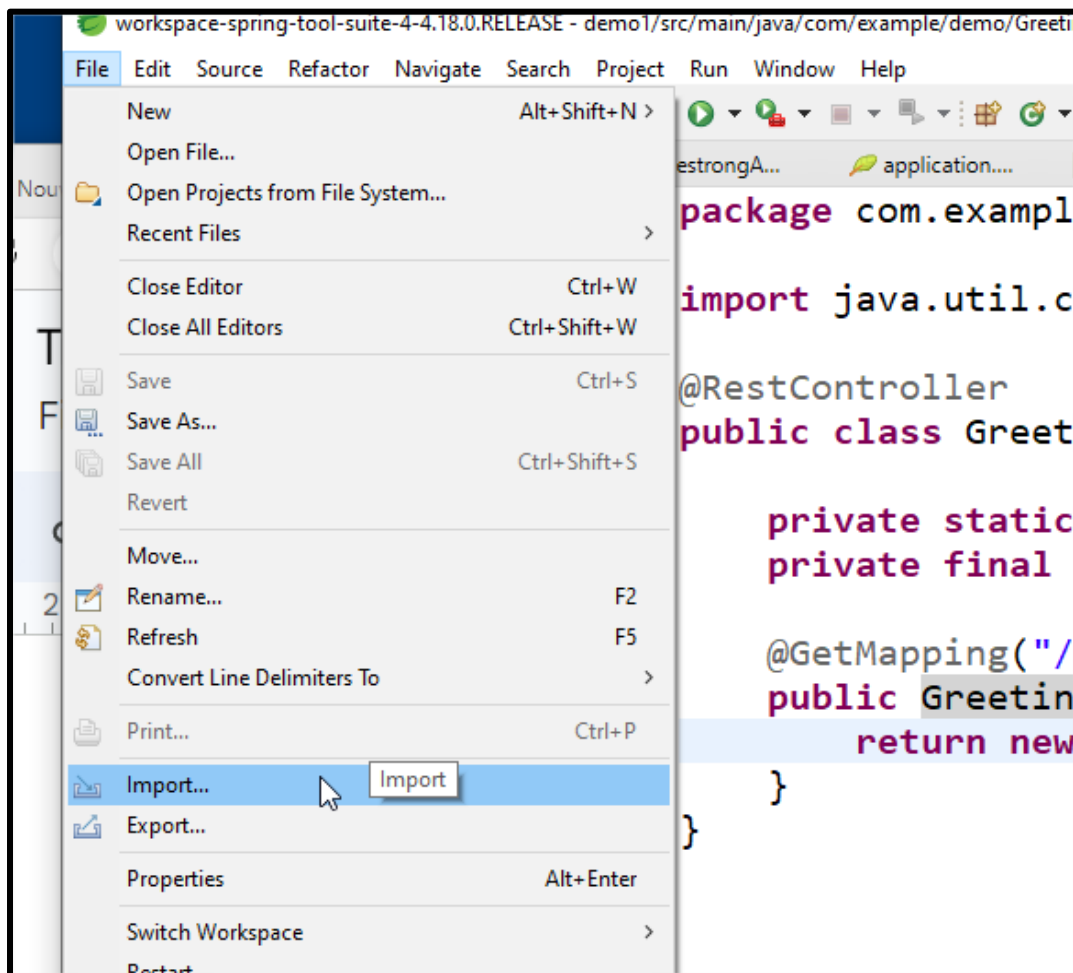
Spring Data JPA **SQL**
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

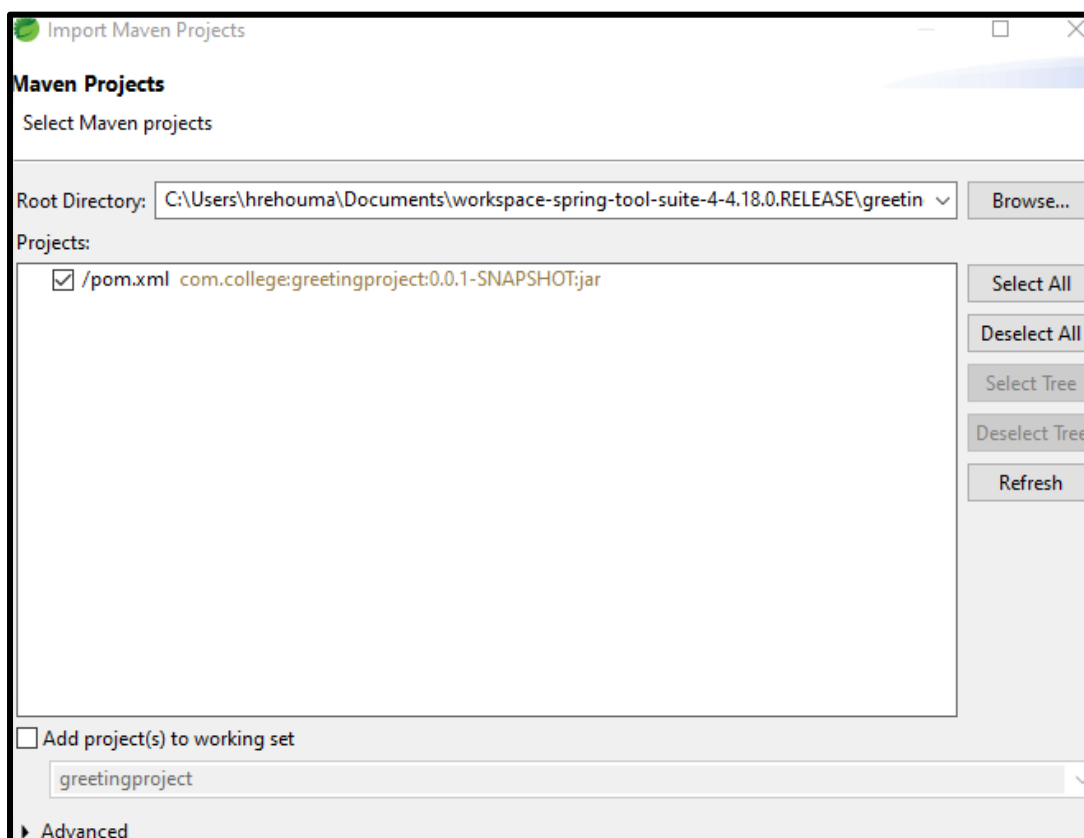
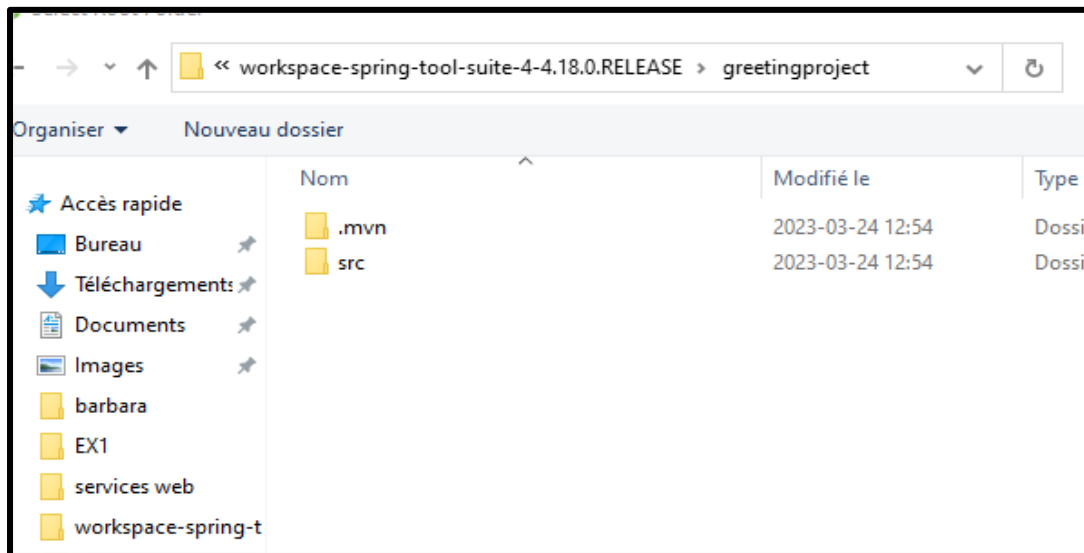
PostgreSQL Driver **SQL**
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

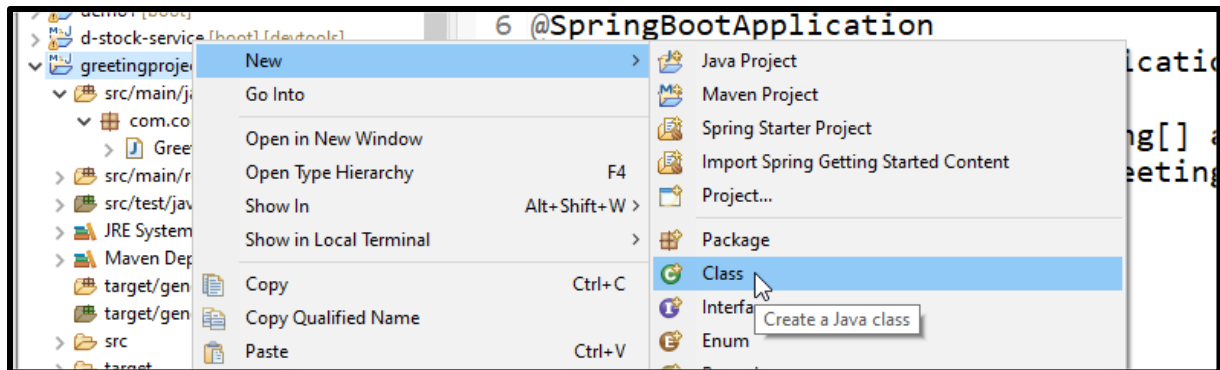
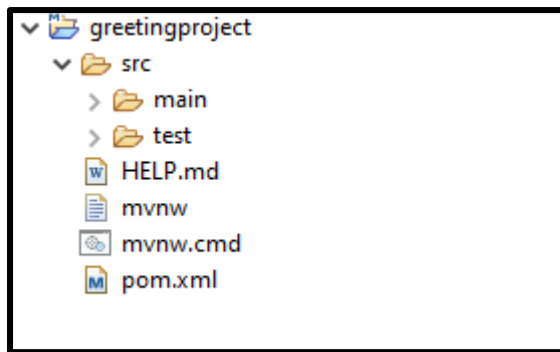
1.3. Le déplacer dans le Workspace



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>
    </dependency>

</dependencies>
```

```
<scope>test</scope>
</dependency>

</dependencies>
```

Exercice :

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

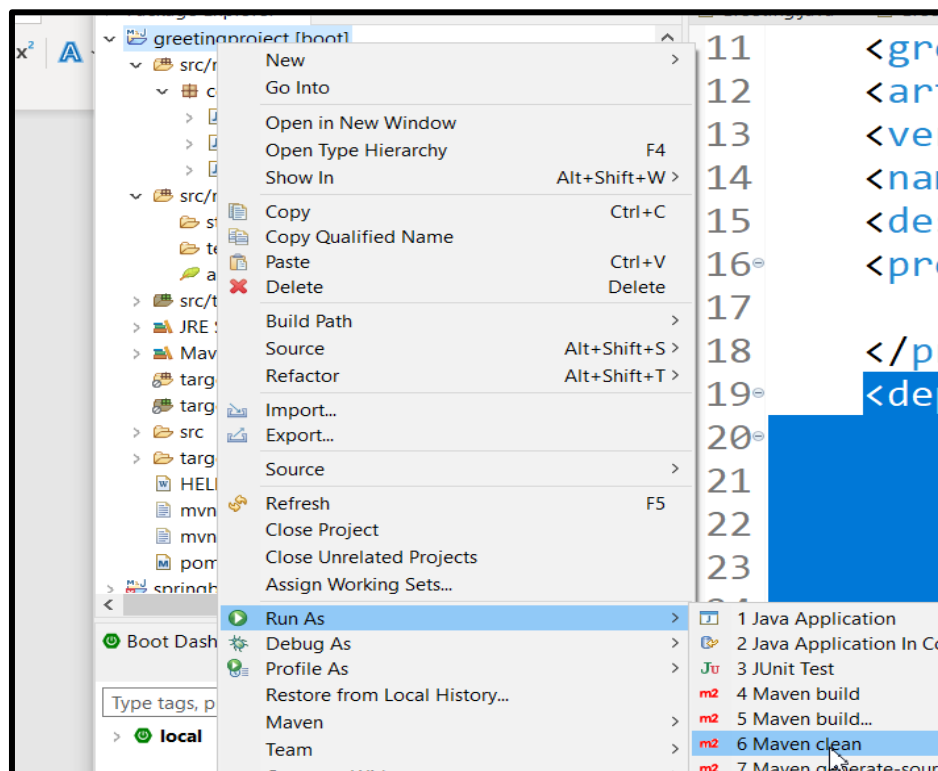
Réponse :

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

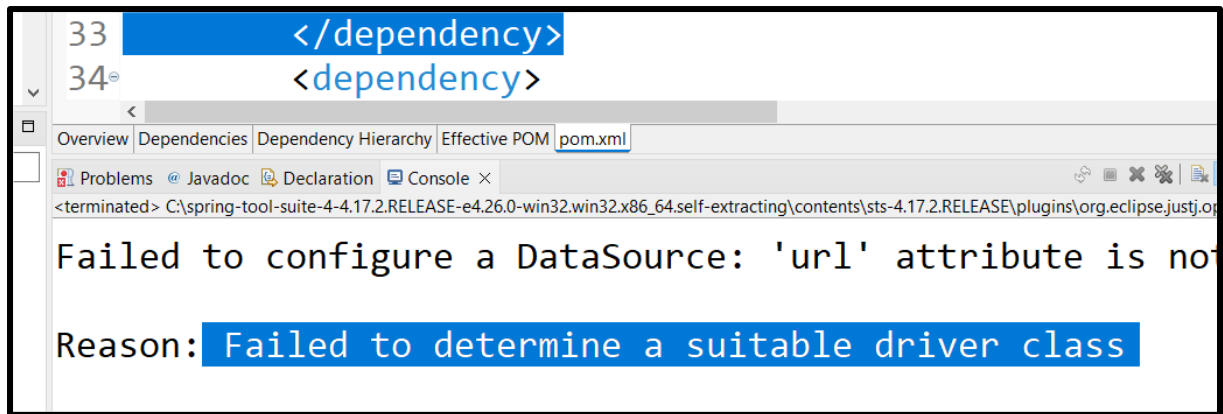
TESTEZ :

Toujours, « run as » :

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

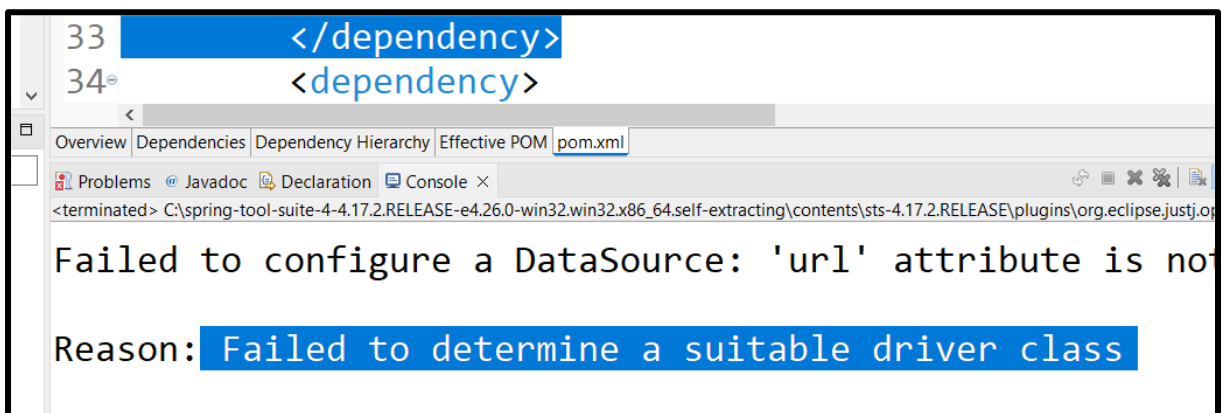


OUPS !!!! ERREUR



1.7. Résolution d'erreurs

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



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
<p>postgresql</p> <pre> 27 </dependency> 28 29 <dependency> 30 <groupId>org.postgresql</groupId> 31 <artifactId>postgresql</artifactId> 32 <scope>runtime</scope> 33 </dependency> </pre>	<p>starter-web</p> <pre> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> </pre>
<p>JPA</p> <pre> 19 </dependencies> 20 <dependencies> 21 <dependency> 22 <groupId>org.springframework.boot</groupId> 23 <artifactId>spring-boot-starter-data-jpa</artifactId> 24 </dependency> </pre>	<p>starter-test</p> <pre> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> </pre>
<p>lombok</p> <pre> 28 29 30 <dependency> 31 <groupId>org.projectlombok</groupId> 32 <artifactId>lombok</artifactId> 33 <optional>true</optional> 34 </dependency> </pre>	

➔ 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>
```

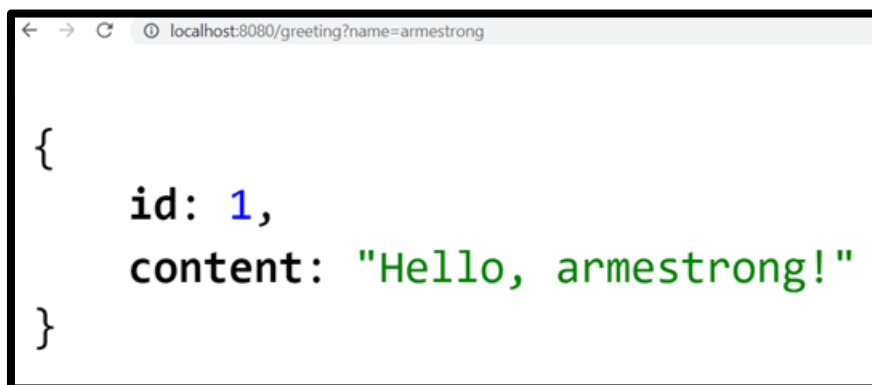
Supprimez les dépendances inutilisées et retestez !!! ça marche !!!!!!!!!!!!!

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=armstrong>



- 5) Changez le point de terminaison à ?name=français
- 6) Changez le point de terminaison à ?name=patrick
- 7) Rafraîchir la page.
- 8) 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`

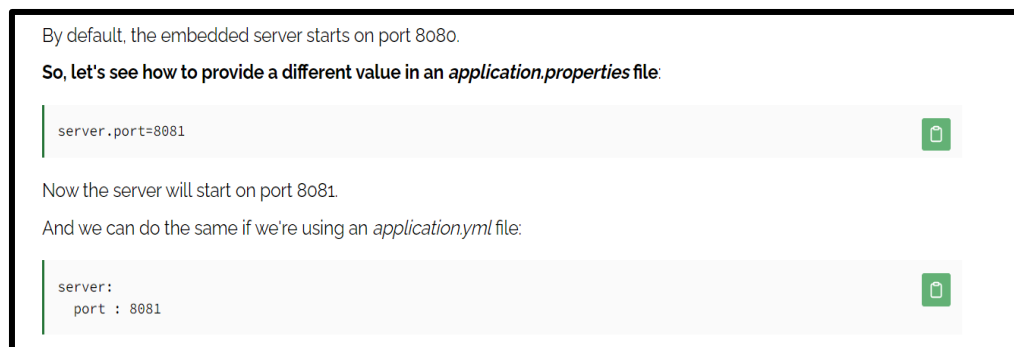
- ⇒ Si le format est `application.properties`, utilisez la ligne suivante :

```
server.port=8081
```

- ⇒ Si le format est `application.yml`, utilisez la ligne suivante :

```
server :  
  Port : 8081
```

Source et références :



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 :

```
C:\Users\Haythem>netstat -noa |findstr :8080
TCP    0.0.0.0:8080      0.0.0.0:0        LISTENING      19528
TCP    [::]:8080       [::]:0           LISTENING      19528
```

```
C:\Users\Haythem>taskkill /F /PID 19528_
```

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