Generating a Spring Boot Project Using start.spring.io

Estimated time needed: 20 minutes

Overview

In this lab, you will be working on Spring Boot application. Spring Boot helps you to create stand-alone, production-grade Spring-based applications that you can run. You will set up a maven project structure, add dependencies, and build and run it in the IntelliJ environment.

> Please note that copy-paste is not allowed in this environment. You will need to type the code where necessary.

Learning objectives

After completing this lab, you will be able to:

- Use Spring Initializr via web(start.spring.io) to configure the project
- Set up a Spring boot project structure
- Edit the core configuration contained in `pom.xml`
- Build and run the spring boot application

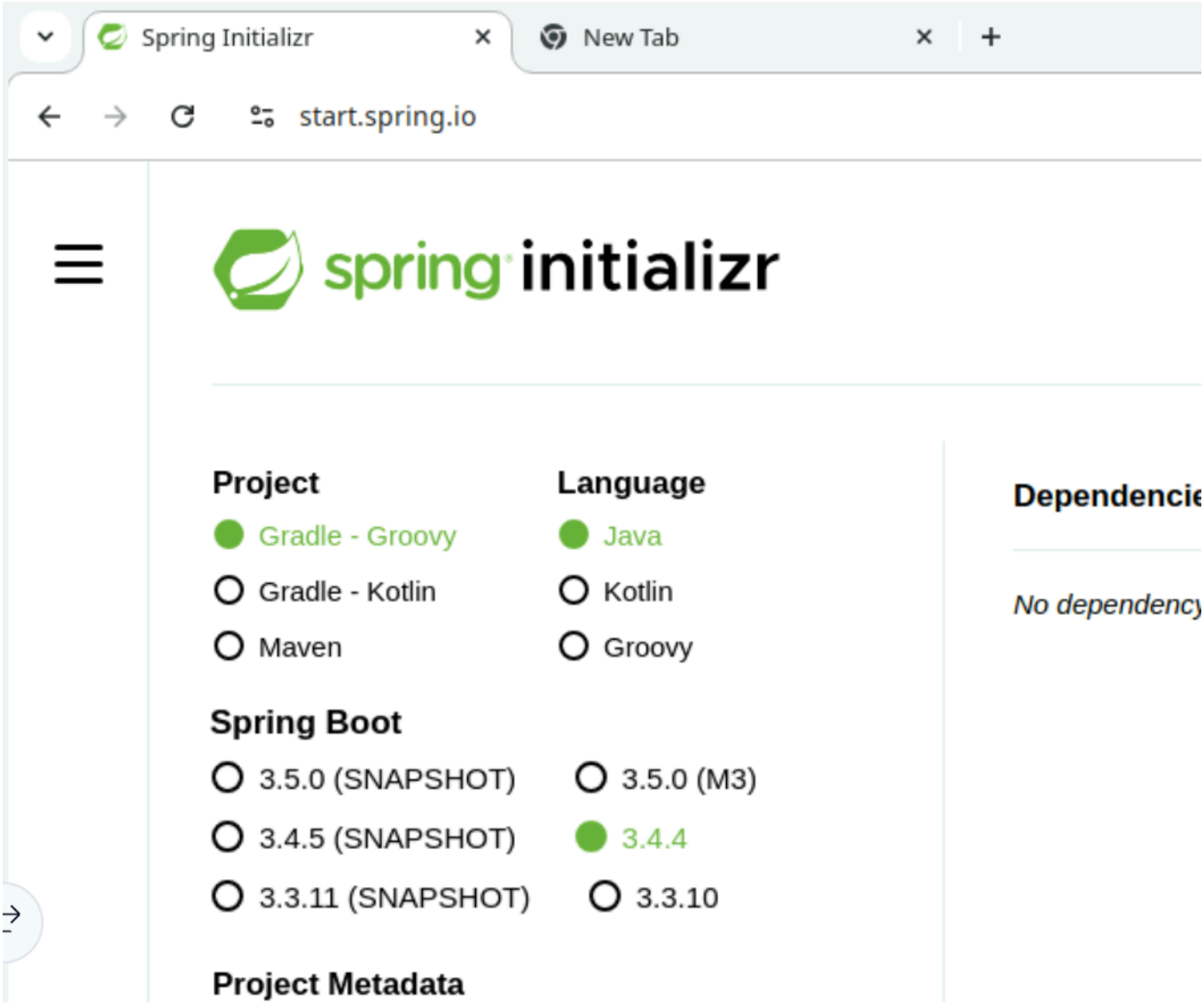
Prerequisites

You should know basic Java programming before you get started with this lab. You should have completed the Spring framework lab.

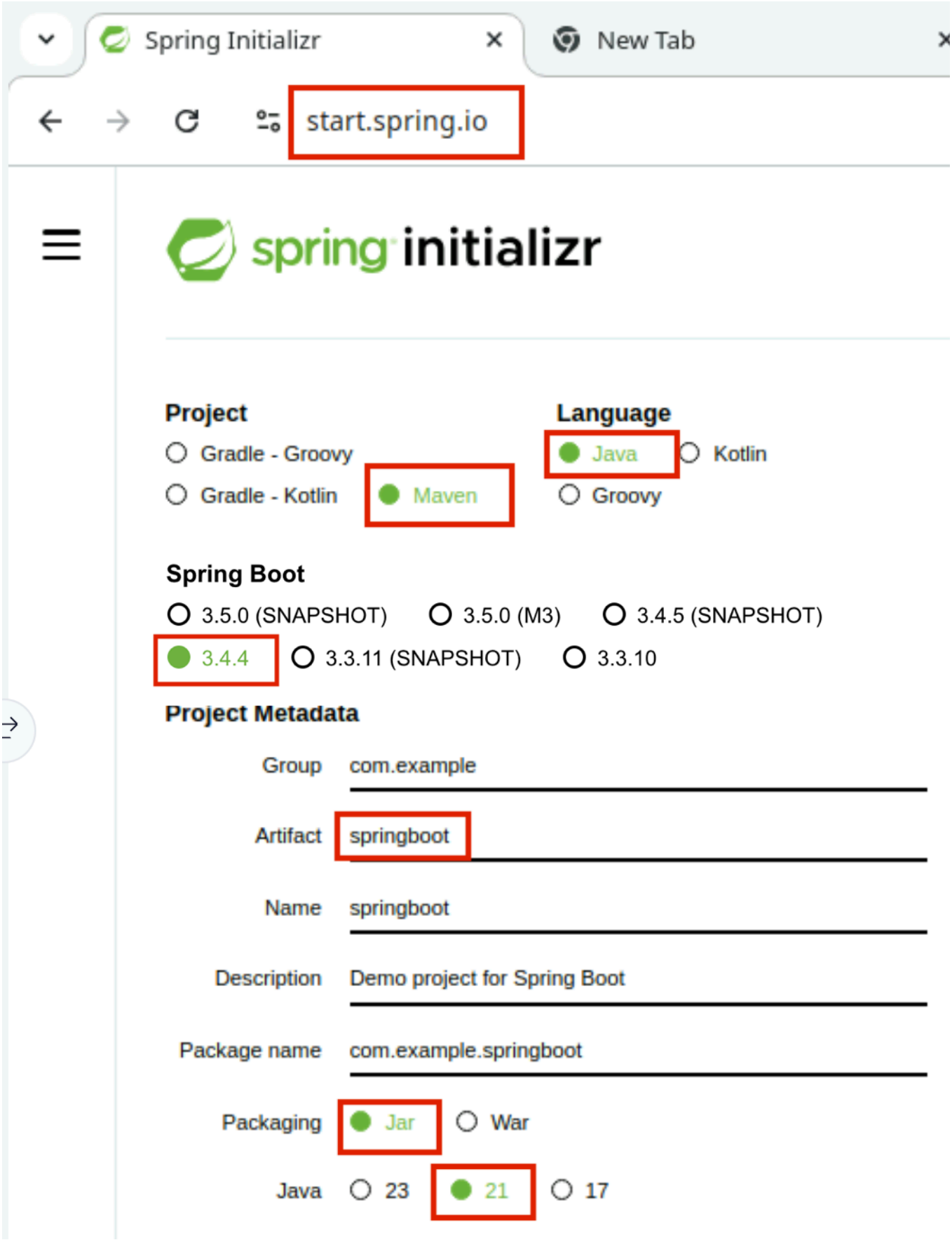
Generate Spring Boot Project with Spring Initializr

1. Open the chrome browser in the lab environment.

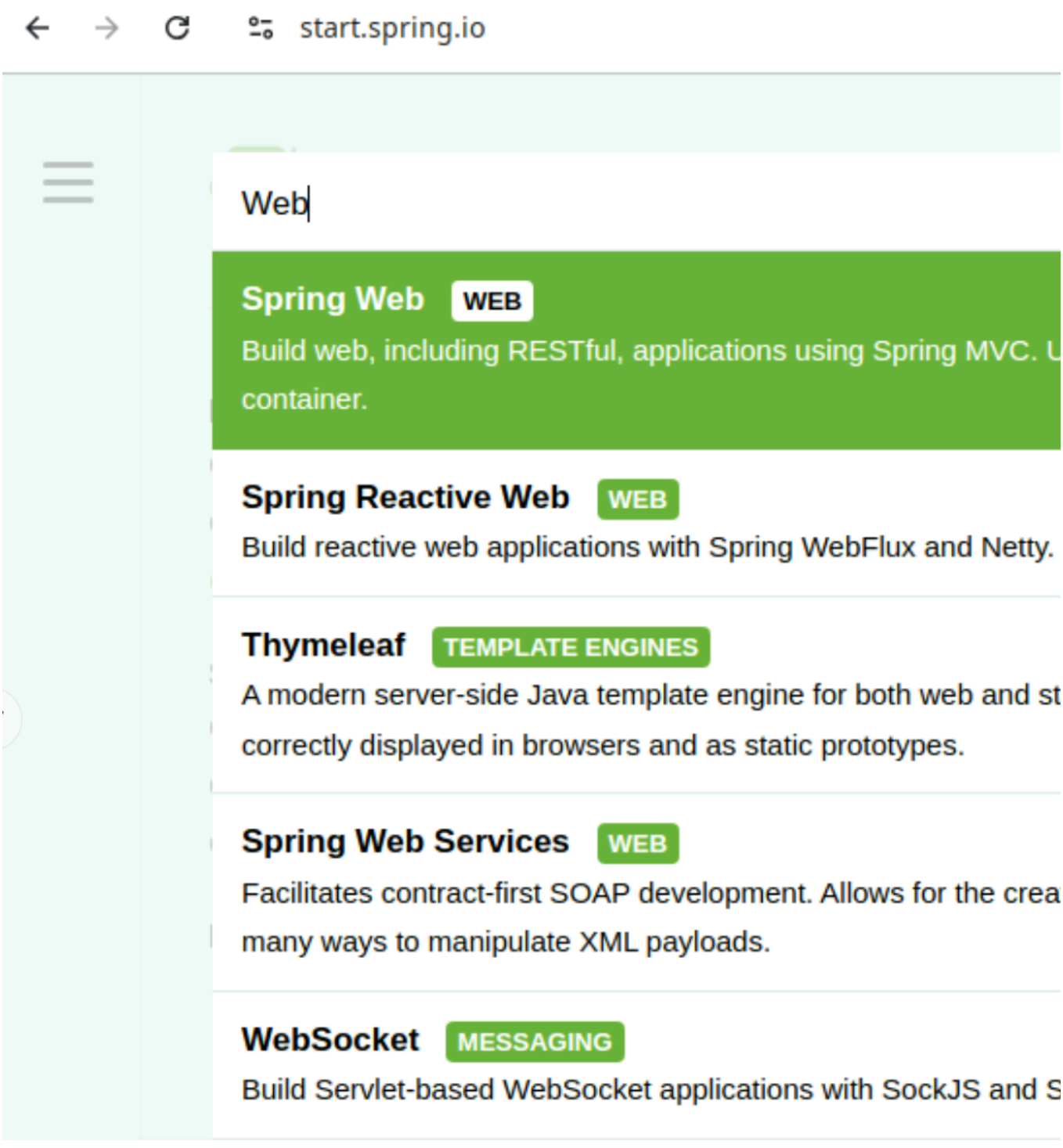
2. In the browser, visit `https://start.spring.io`. This page facilitiates creating Spring Boot projects on IntelliJ Community Edition. The Spring Initializr web page opens up with the default configuration.



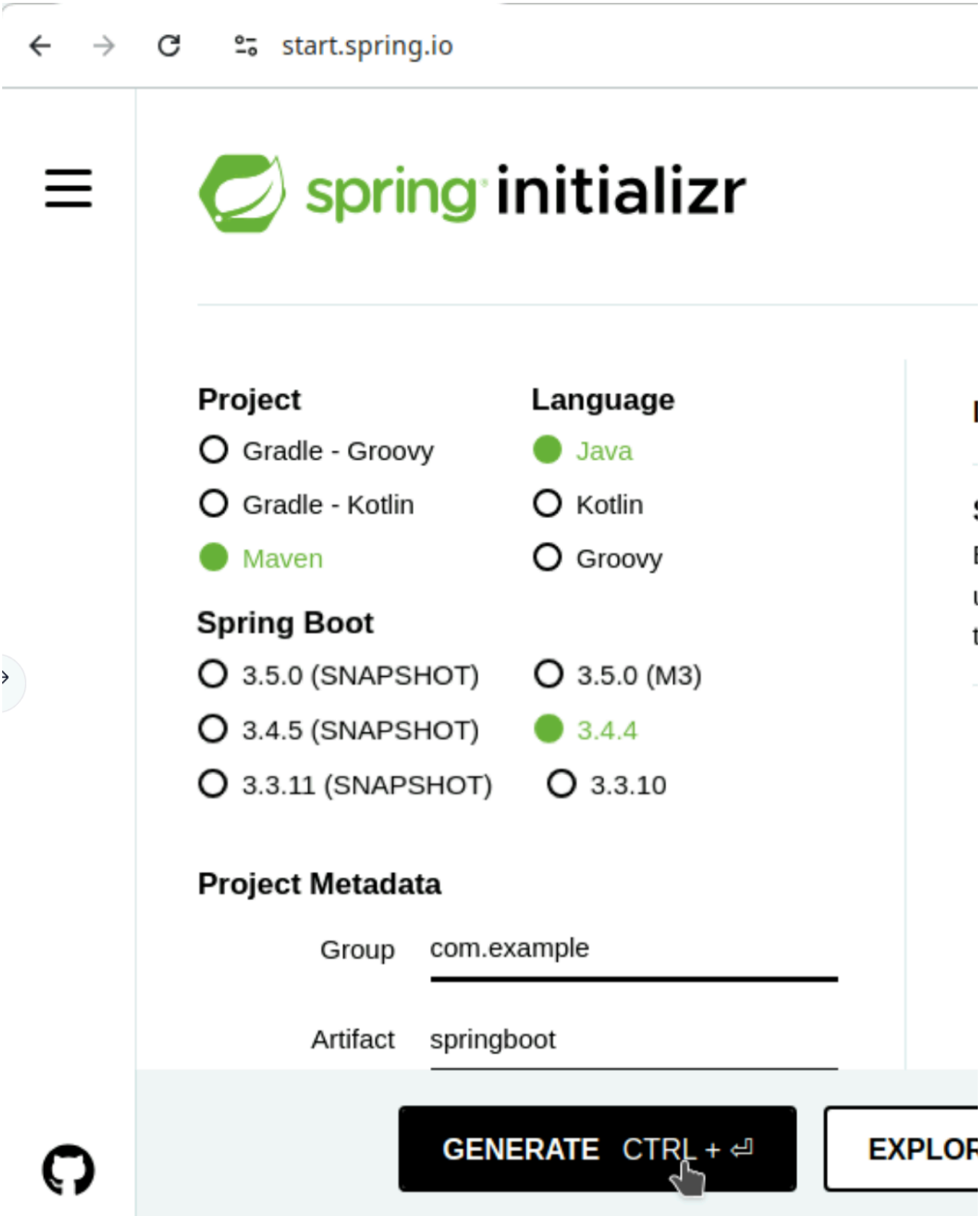
3. Add the values as per the image below, on the page. And then click `ADD DEPENDENCIES`.



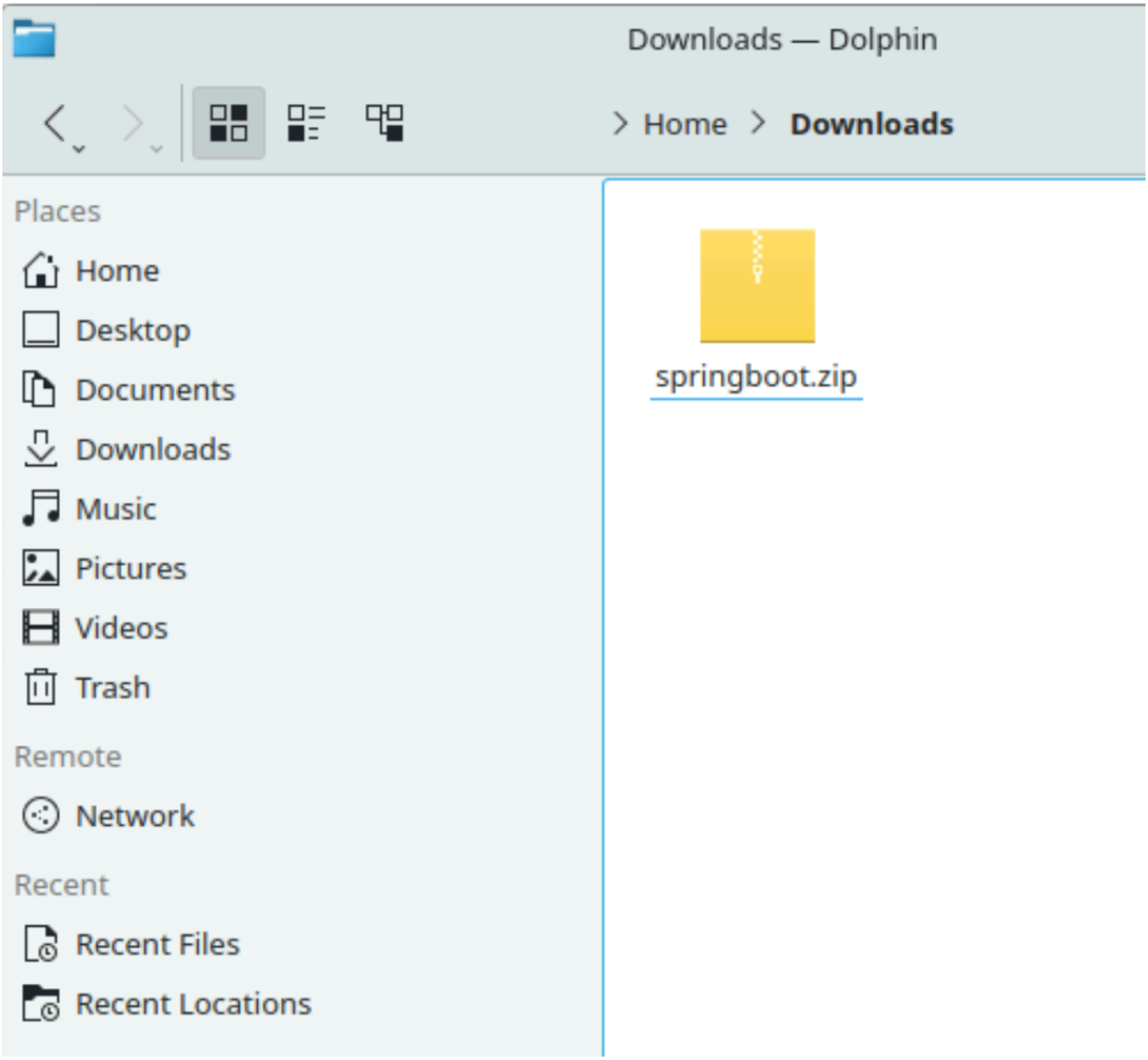
4. Search for Web dependency and add it.



5. Click `Generate` to generate the project as a zipfile. This will be automatically downloaded in the Linux environment.



6. Check the `Download` folders in the Linux lab environment to see that the project zip file is downloaded.



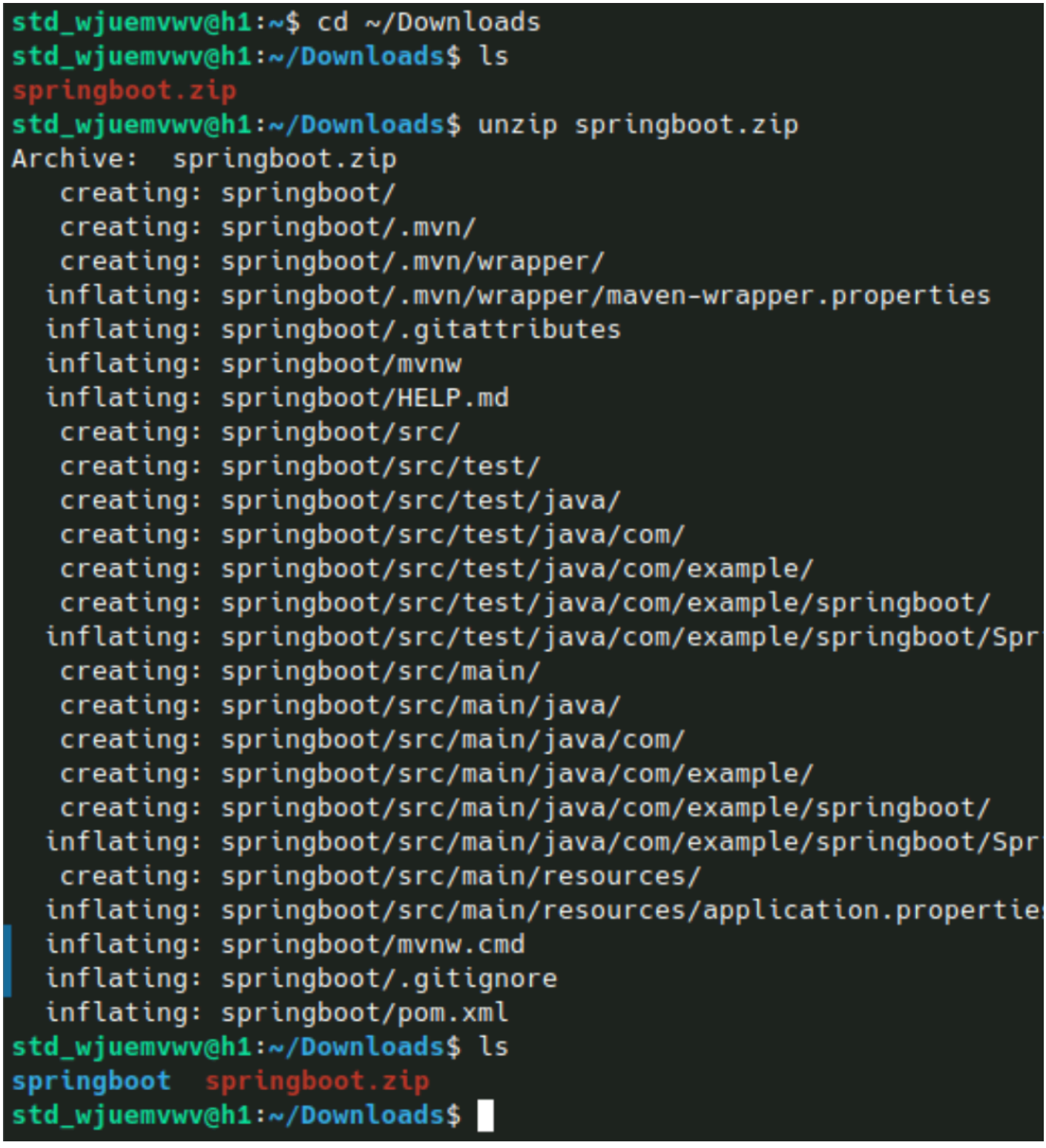
7. Open the `Konsole` app.

8. Navigate to `~/Downloads` directory and list it.

```
cd ~/Downloads
ls
```

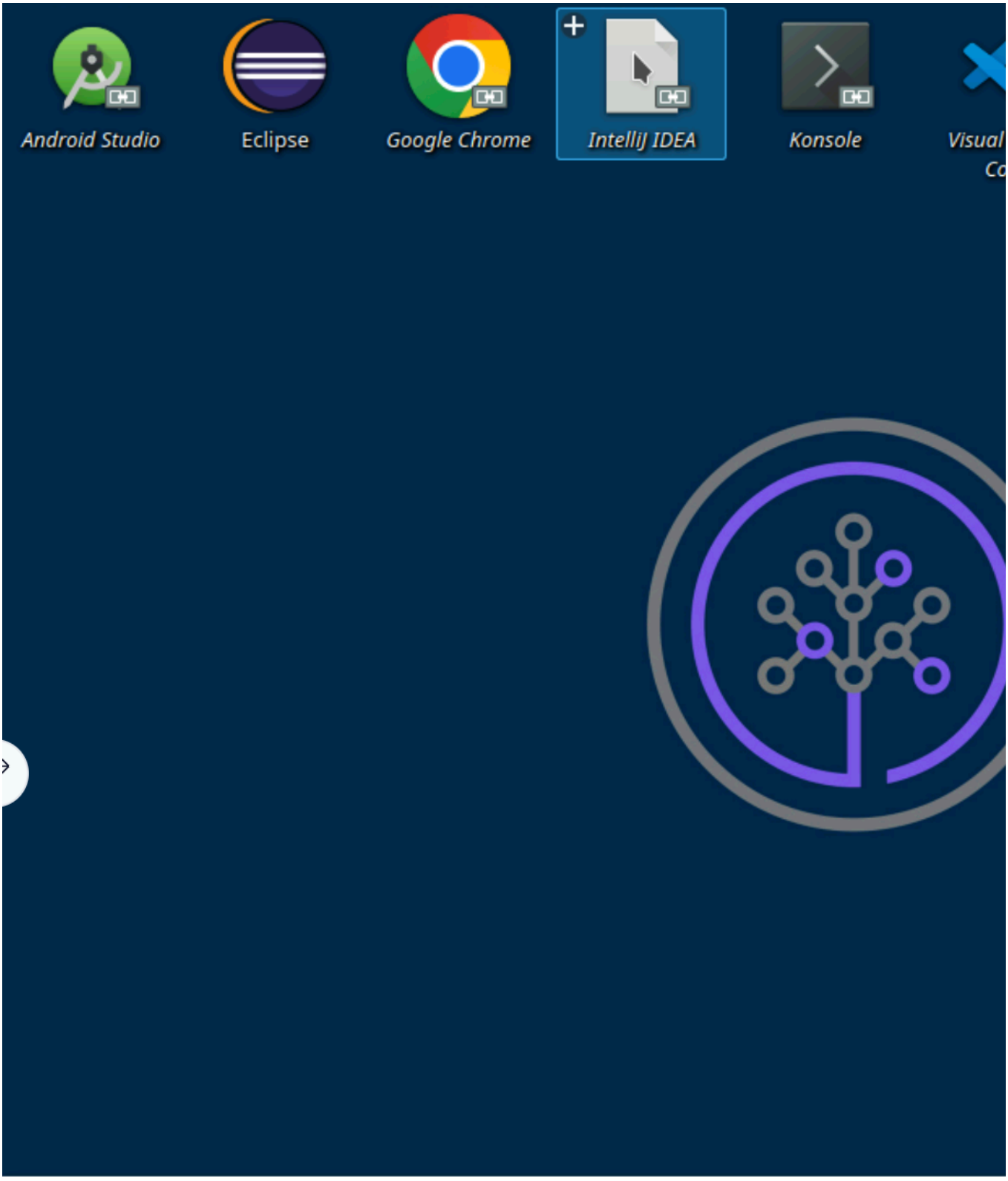
9. Unzip the file on the Konsole. The unzipped folder is the project folder `springboot` that you will load in IntelliJ.

```
unzip springboot.zip
```

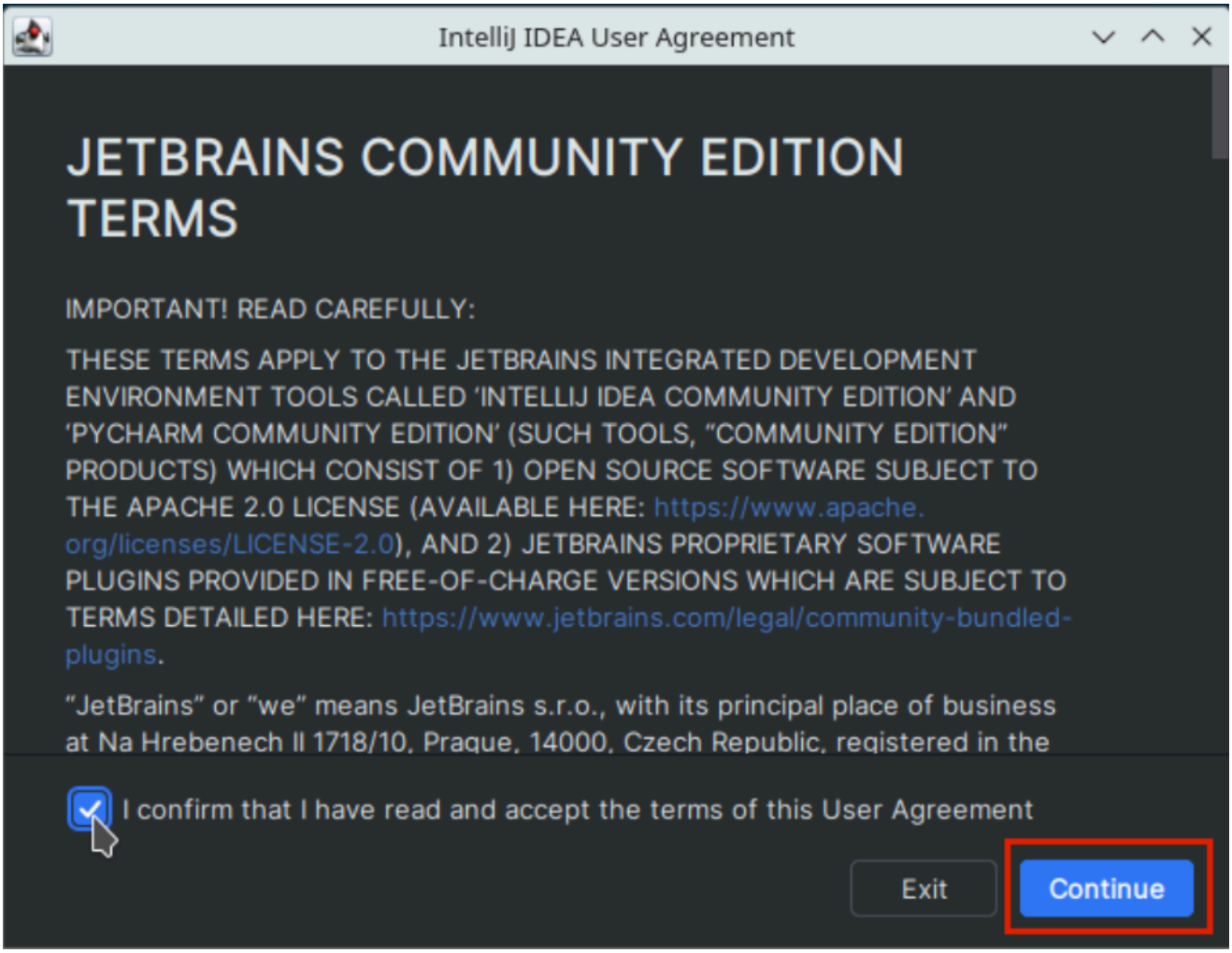


Load project in IntelliJ

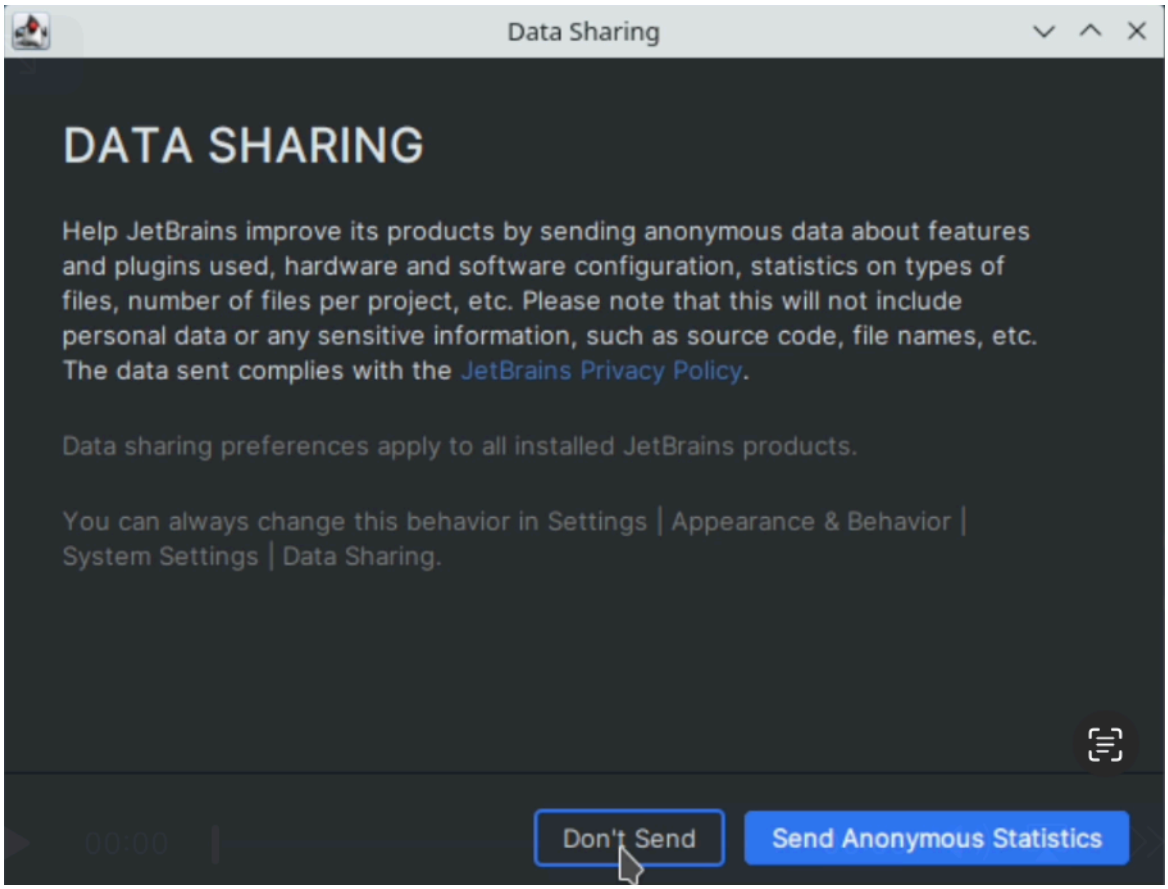
1. Open the IntelliJ IDE in the lab environment provided.



2. A `User Agreement` containing the terms and conditions for using the IntelliJ community edition will be displayed. Agree by selecting the checkbox and clicking `Continue`.

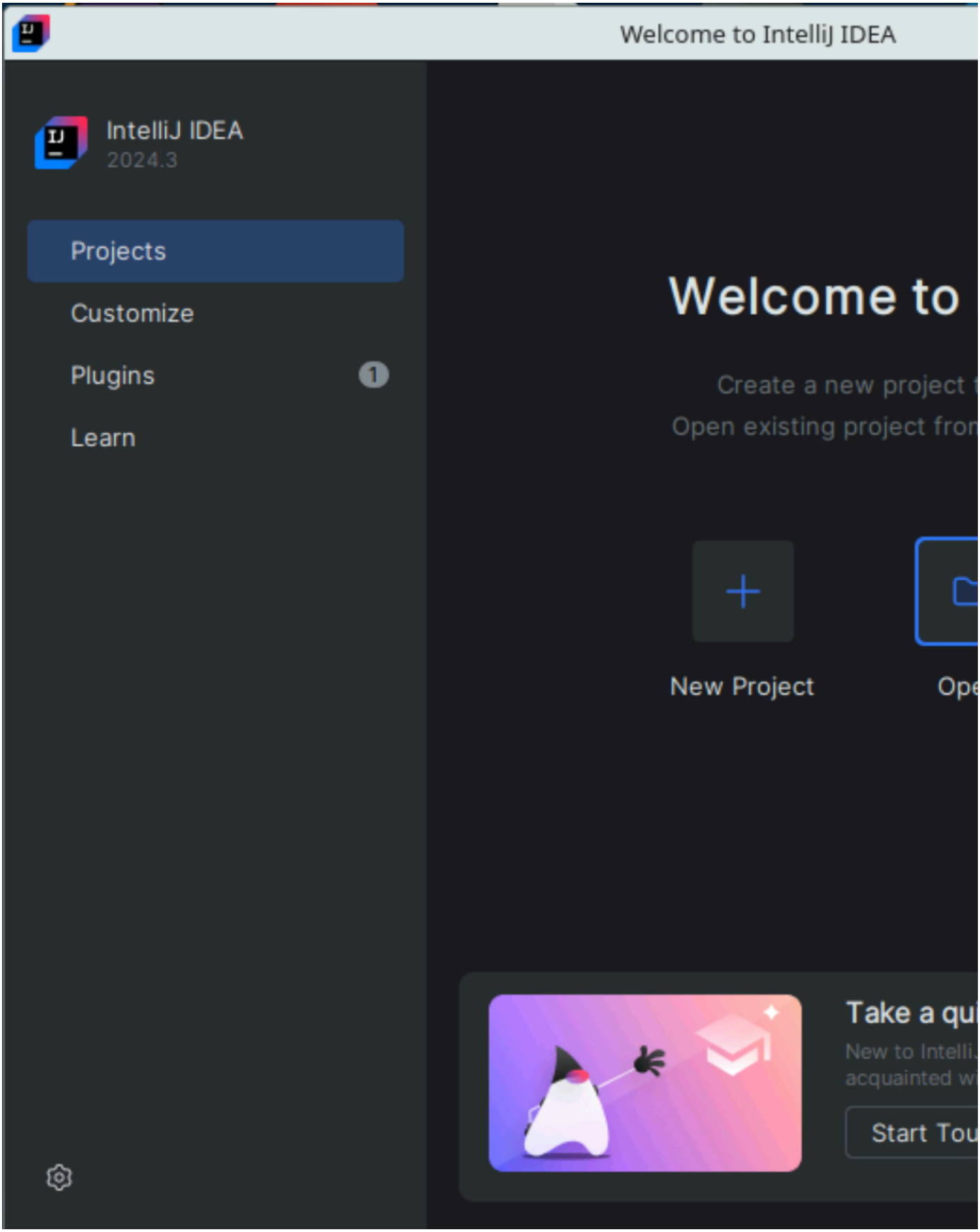


3. A window asking if you want to share your data on usage statistics will be displayed. Click `Don't Send` option.

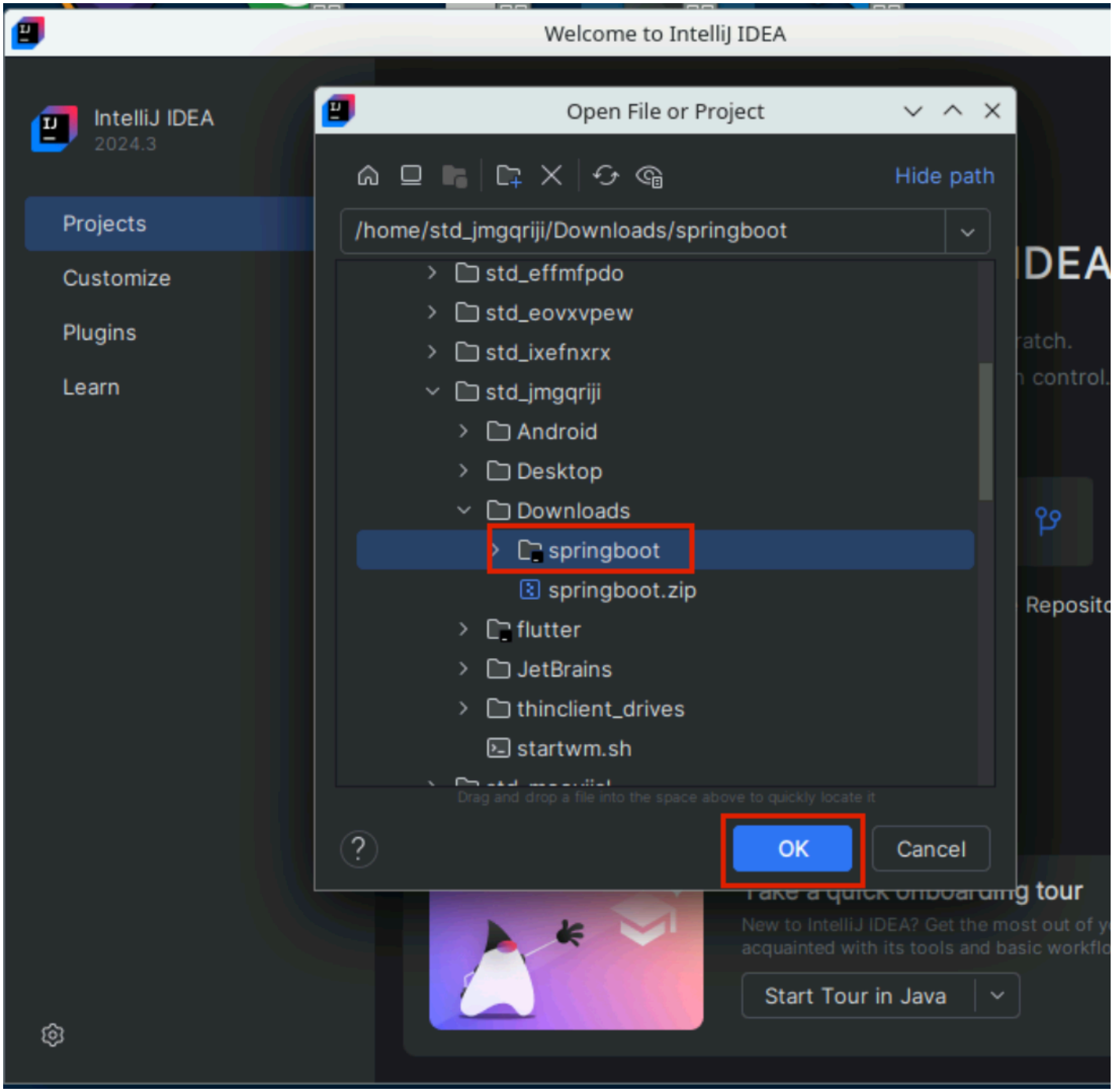


4. IntelliJ will open and provide you the options for creating a new project, opening an existing project from the Linux environment, or cloning from a repository.

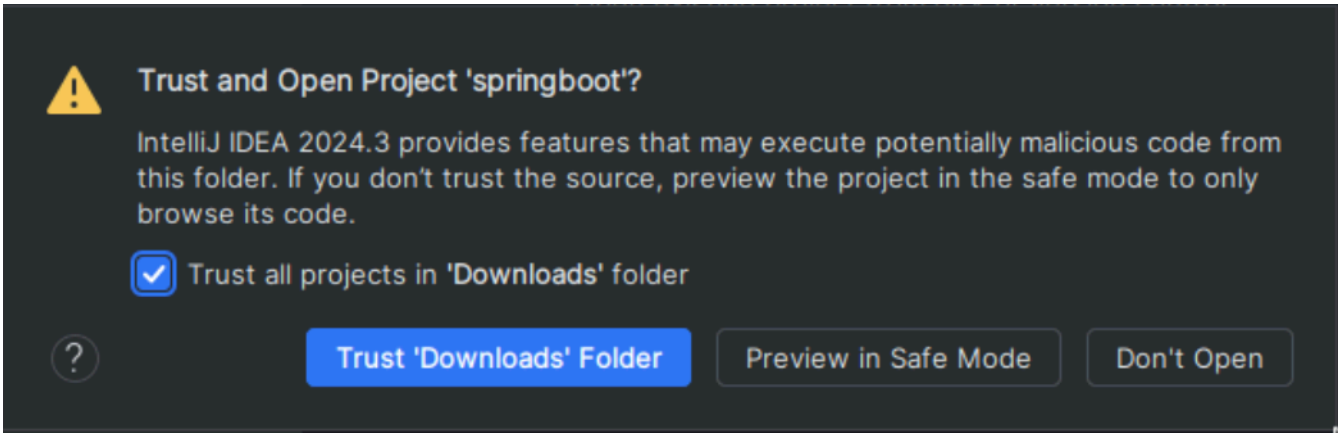
Choose the option to open an existing project.



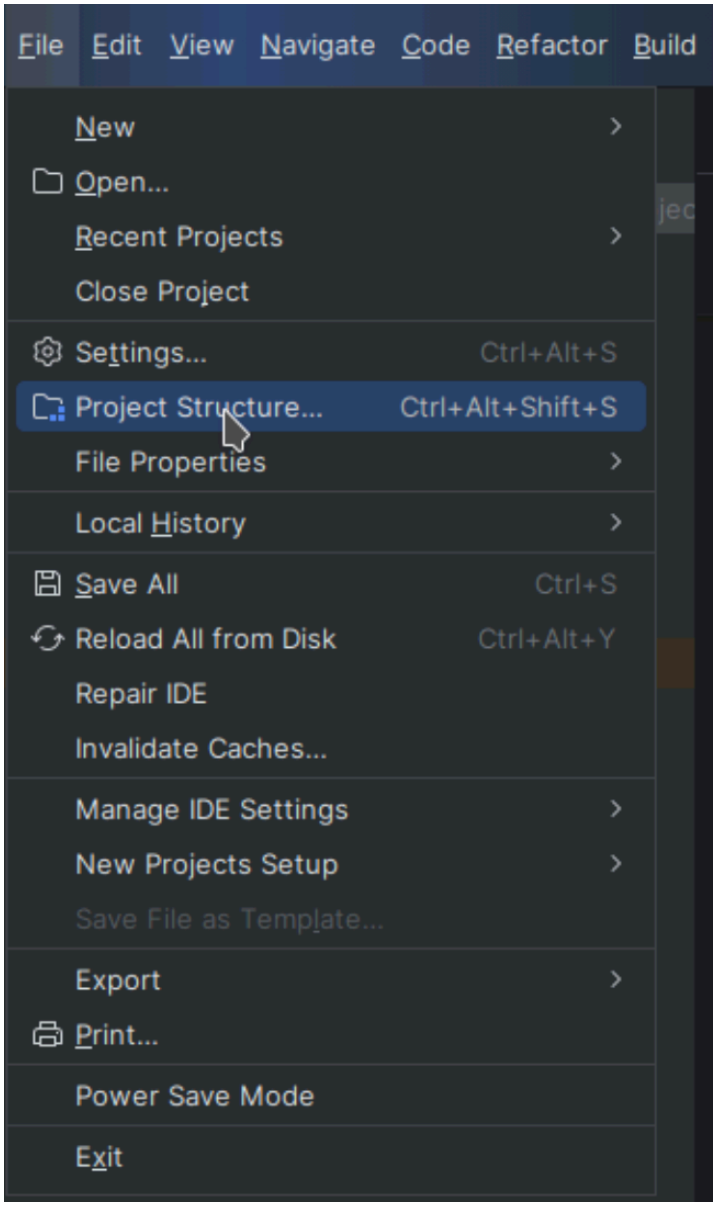
5. Choose the `springboot` folder that was previously unzipped and click `Ok`.



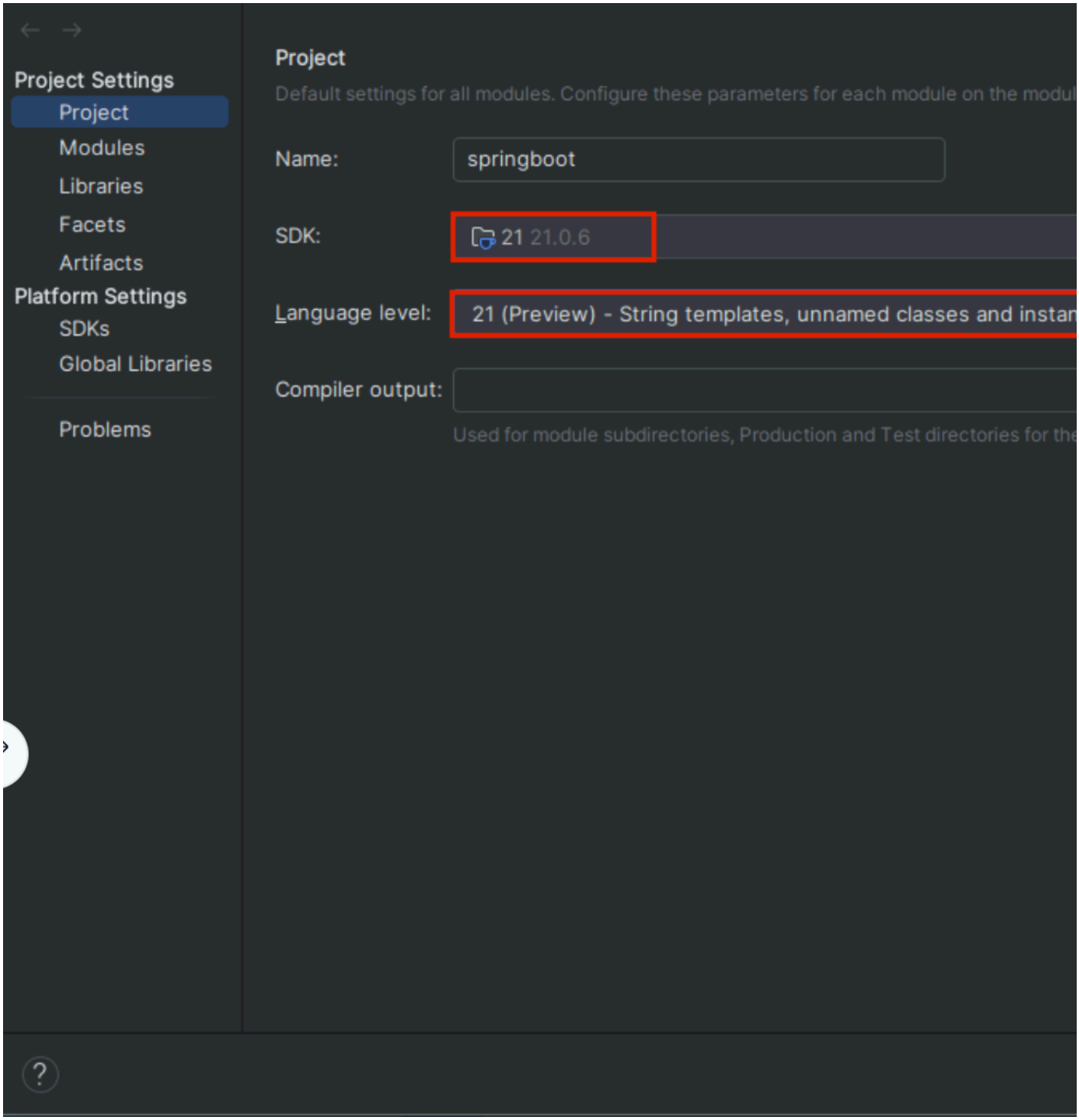
6. As the zipfile for the project was generated and downloaded from an external site, IntelliJ asks for permission to trust the folder. Continue to choose "Trust Downloads folder" option.



7. Click the `File` menu and select `Project Structure` to open the project settings.



7. Set the SDK and Language level to the version of Java installed (21 in this case), as shown in the image.



Configure the project

1. Open the `pom.xml` and check for the following content.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

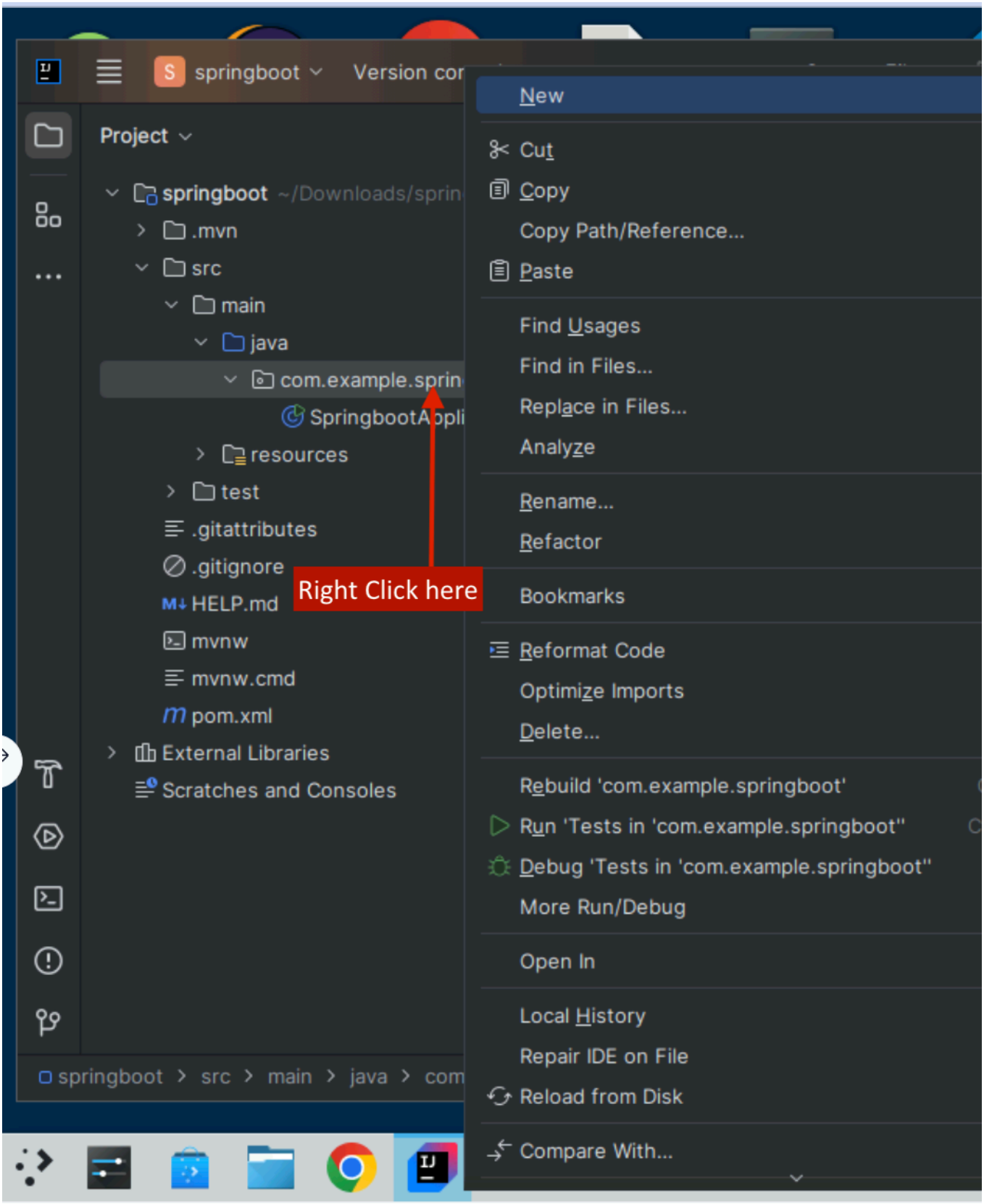
The Spring Boot Web Starter dependency includes a set of dependencies that are commonly used when building web applications with Spring Boot. Some of the key dependencies included in the Spring Boot Web Starter include:

- Spring MVC

- Tomcat (as the default embedded servlet container)
- Jackson (for JSON serialization and deserialization)
- Hibernate Validator (for data validation when data is to be persisted in database)

By including the Spring Boot Web Starter dependency in a project, you can quickly and easily get started with building web applications using Spring Boot.

2. Add a new class named `com.example.springboot.HelloController` under `src` folder.

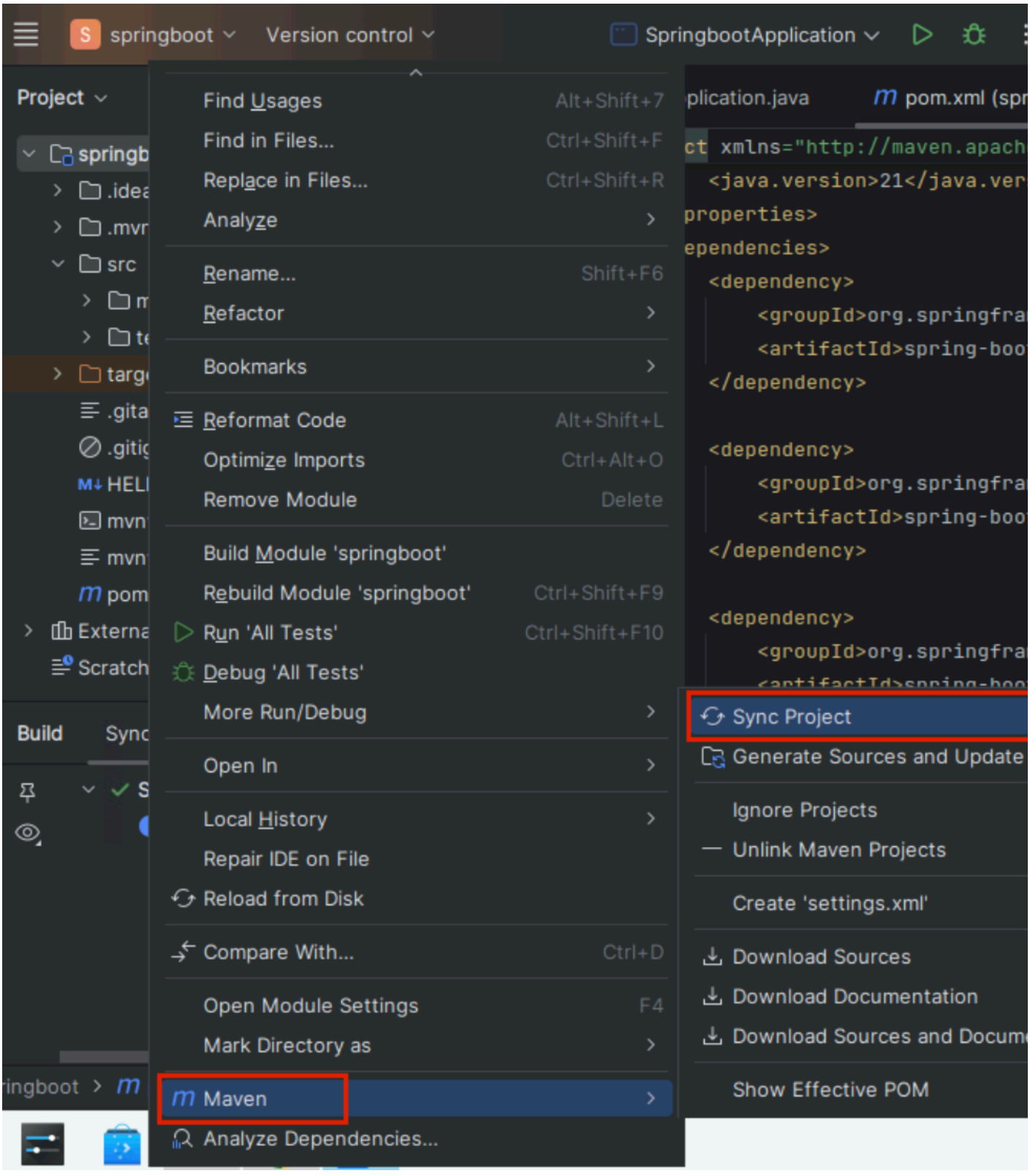


3. Add the following content in `HelloController.java`.

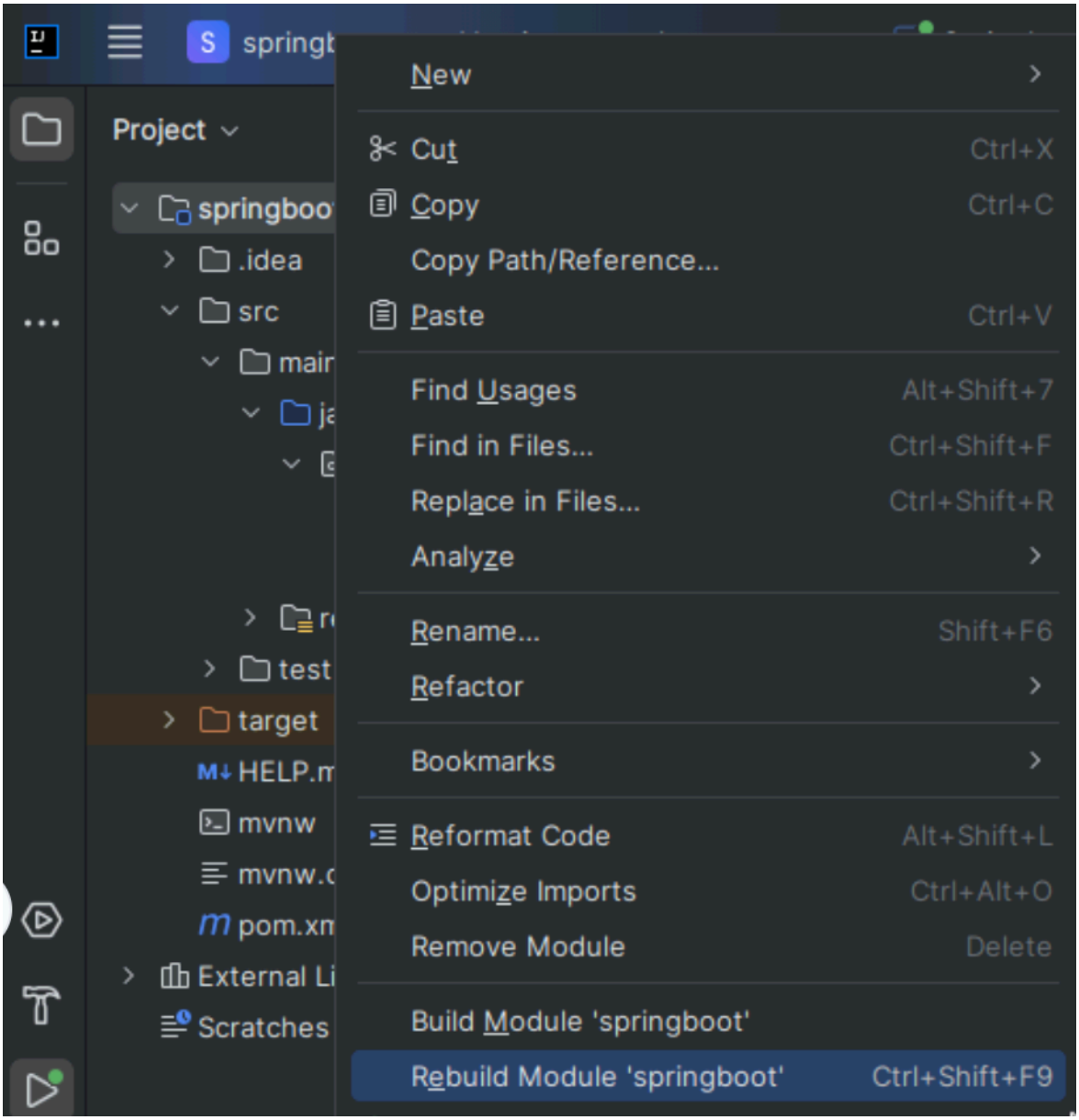
```
package com.example.springboot;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
    @GetMapping("/")
    public String sayHello() {
```

```
        return "Hello, Spring Boot!";
    }
}
```

4. The environment needs to resync for the new configuration to be recognized. Right-click on the project, choose `Maven` from the menu, and click `Sync Project`.

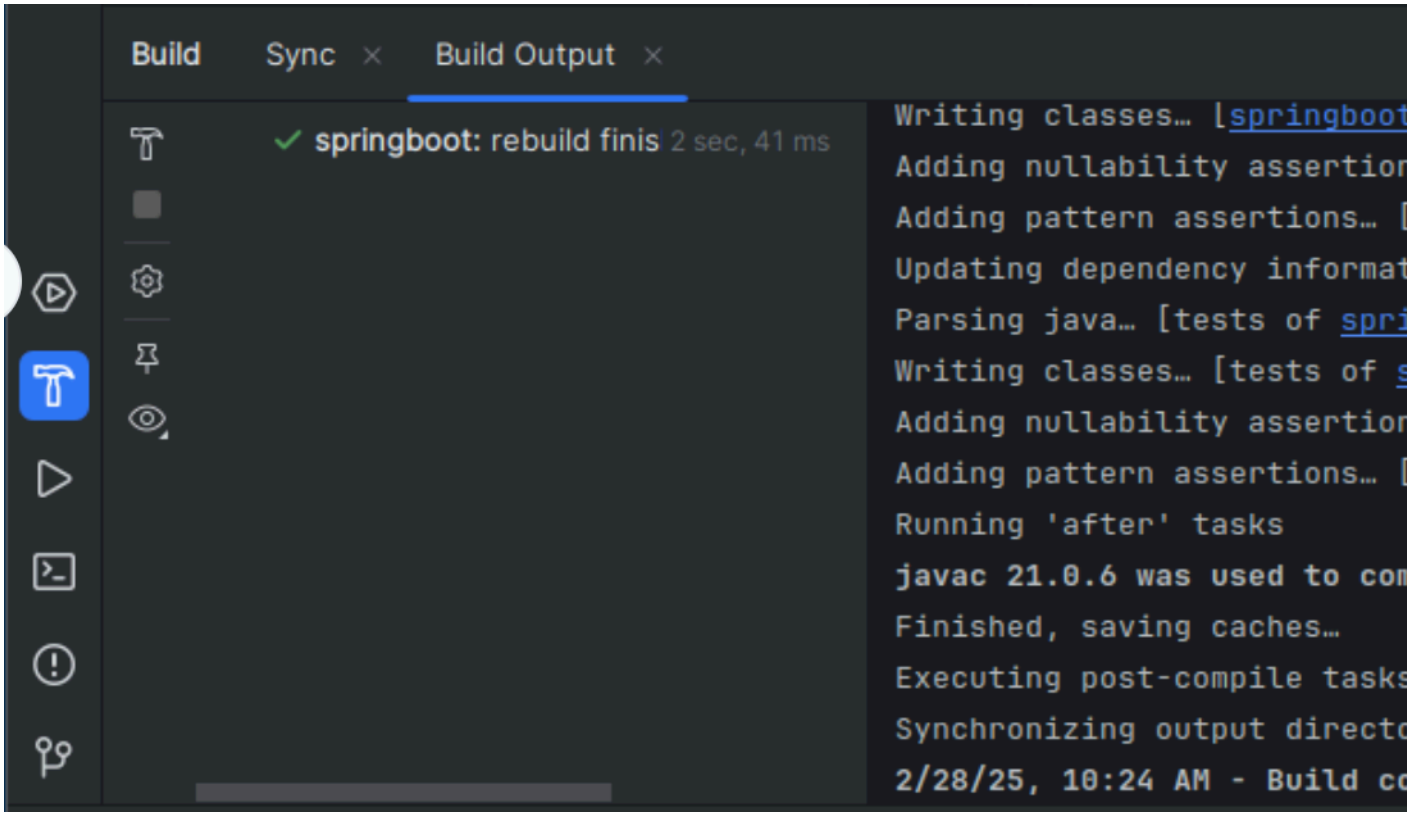


5. Right-click on the project name and click `Rebuild Module`.



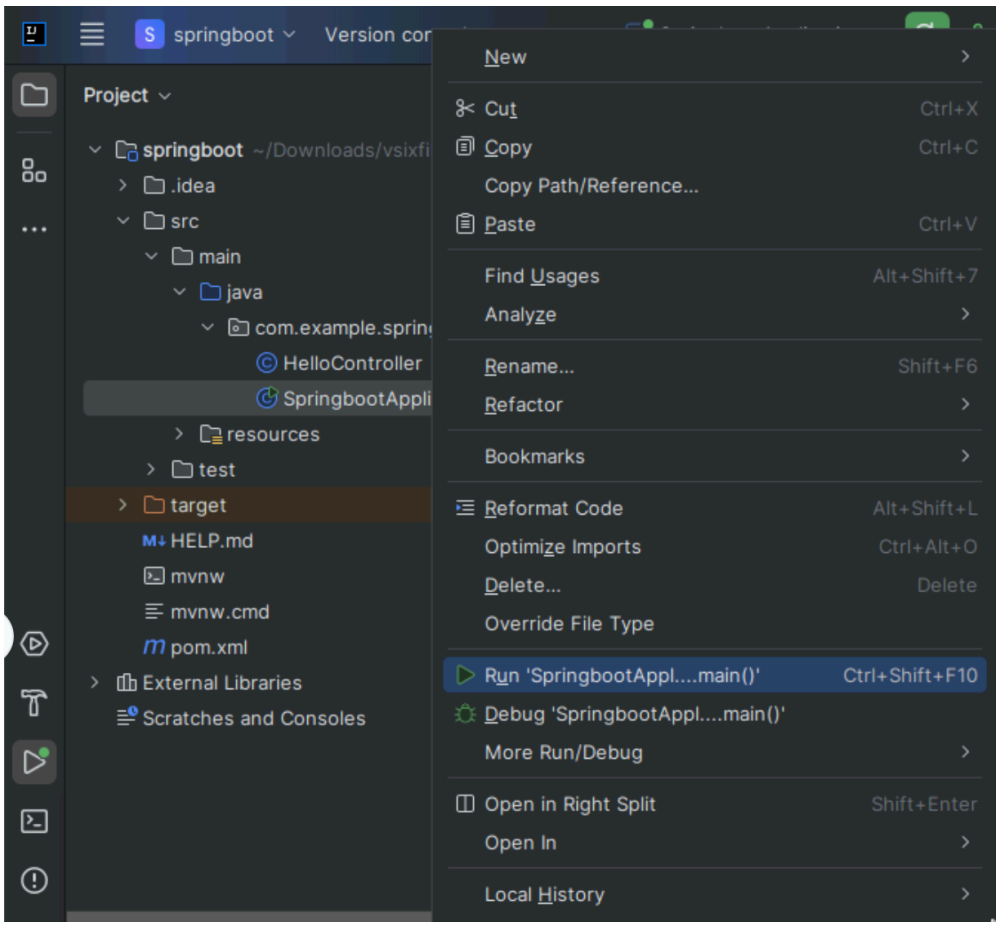
Build and run

1. Click the build icon to build the project.



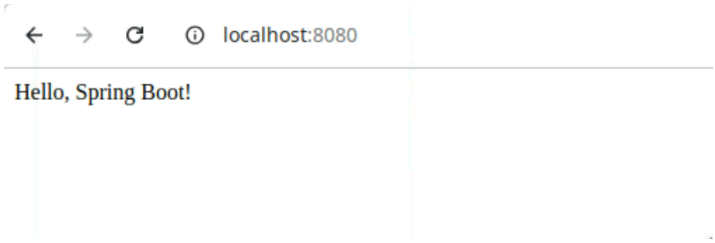
You will see a message confirming that the build is successful.

3. Right-click the `SpringBootApplication` class and click `Run`.



You will see the output in the window. The server application runs on port 8080 and you have a confirmation that the webserver is up and running.

4. On the browser, connect to `localhost:8080` and you should see the Hello Message as seen below.



Practice Exercise

1. In `HelloController.java`, change the message return `Hello` followed by the current date and time . Rebuild the project. Run it once the build is successful and view it in the browser.

Conclusion

In this lab, you have:

- Used Spring Initializr via web to configure the project in IntelliJ Community Edition
- Set up a Spring boot project structure
- Edited the core configuration contained in `pom.xml` to allow web starter pack that includes Tomcat and other server
- Built and run the spring boot application

Author(s)

Lavanya