**Week4\_Creating a Spring Web Project with Maven**

**Follow steps below to create a project:**

**Step 1: Go to [https://start.spring.io/](https://www.google.com/url?sa=E&q=https://start.spring.io/" \t "https://aistudio.google.com/prompts/_blank)**

* The project generation process begins at the official Spring Initializr web portal.

**Step 2: Change Group as “com.cognizant”**

* The project's Group ID, which represents the organization's domain, was set to com.cognizant.

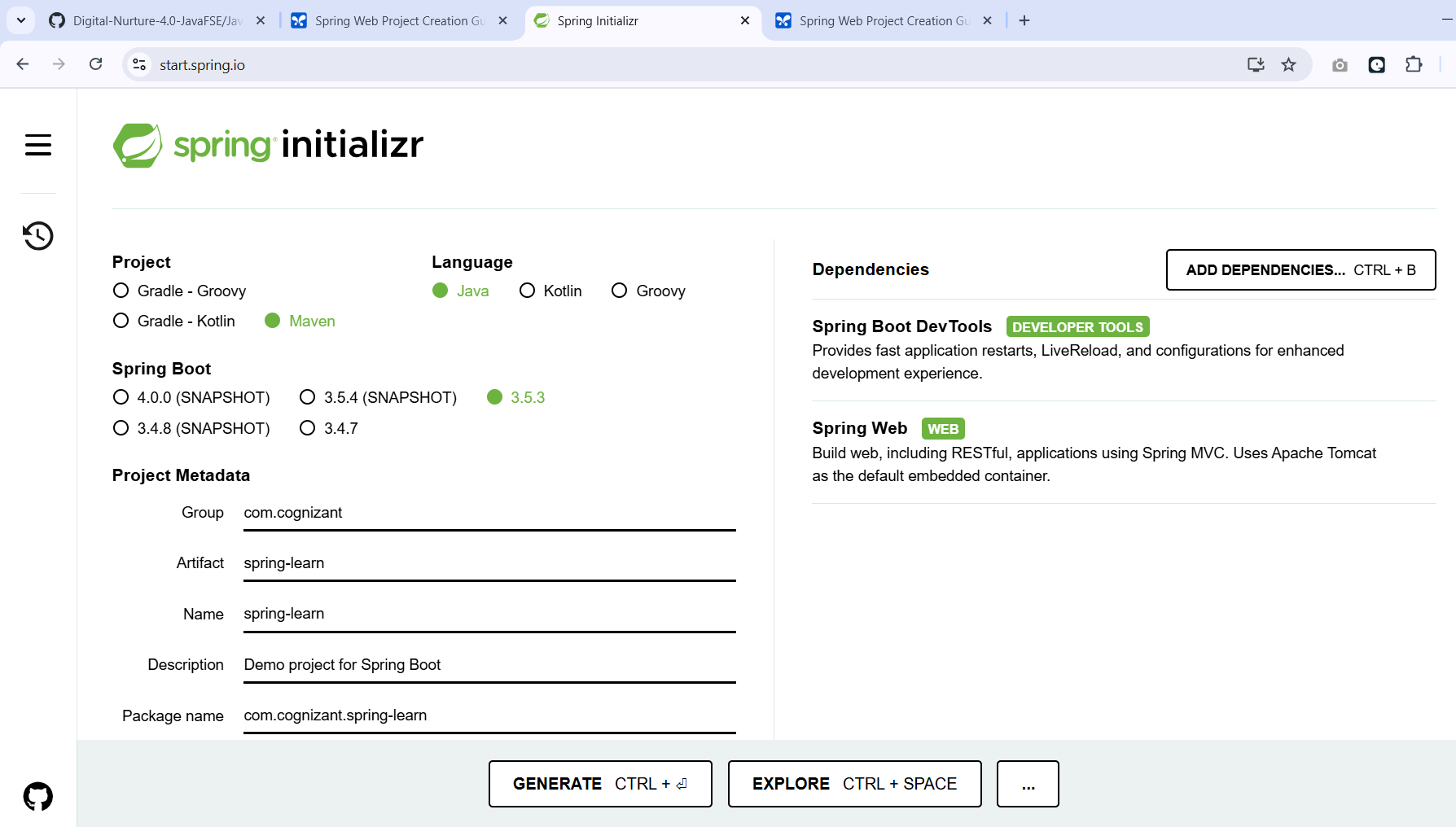
**Step 3: Change Artifact Id as “spring-learn”**

* The project's Artifact ID, which is the unique name of the project, was set to spring-learn.

**Step 4: Select Spring Boot DevTools and Spring Web**

The following starter dependencies were added to the project:

* Spring Boot DevTools: For automatic application restarts and other development-time features.
* Spring Web: To build a web application, including an embedded Tomcat server.



**Step 5: Create and download the project as zip**

* The configured project was generated and downloaded as a spring-learn.zip archive.

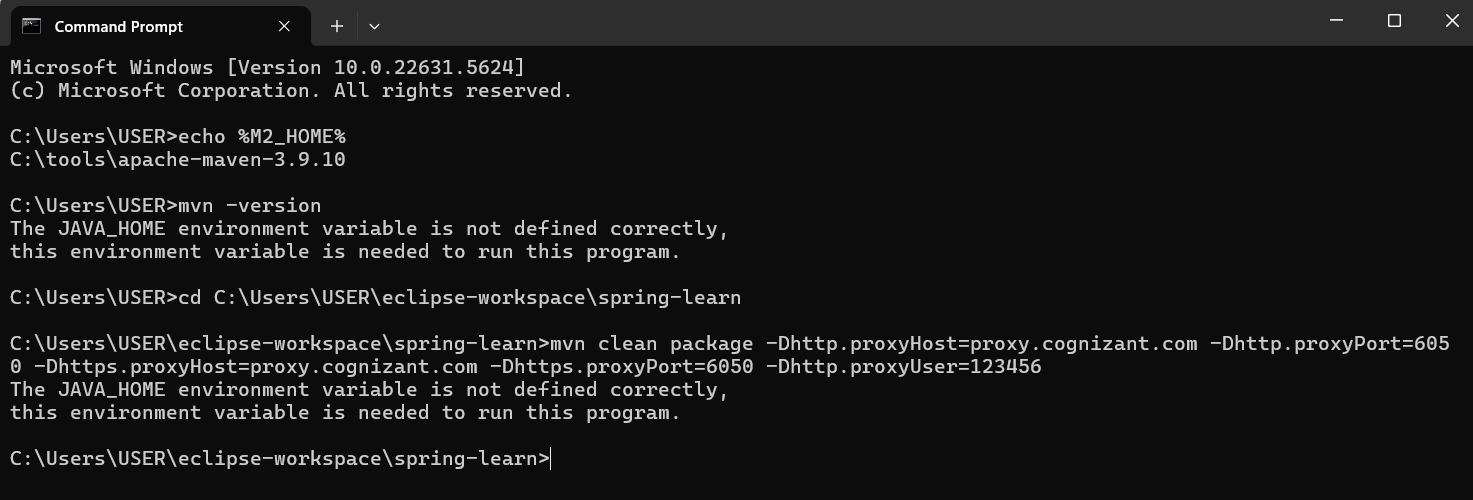
**Step 6: Extract the zip in root folder to Eclipse Workspace**

* The spring-learn.zip file was unzipped directly into the primary Eclipse Workspace directory.

**Step 7: Build the project using Maven Command**

* The project was built from the command line using the following Maven command to compile code, run tests, and package the application.

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com - Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com - Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456



**Step 8: Import the project in Eclipse**

The existing Maven project was imported into the Eclipse IDE using the path: File > Import > Maven > Existing Maven Projects.

**Step 9: Include logs to verify main() method**

Logger statements were added to the main() method within the SpringLearnApplication.java class to confirm its execution start and successful completion.

**Step 10: Run the SpringLearnApplication class**

The application was launched by running the SpringLearnApplication.java class as a Java Application within Eclipse.

**Output**



**SME to walk through the following aspects related to the project created:**

**1. src/main/java**

* This folder contains all the application's Java source code, including the main application class and any controllers, services, or repositories.

**2. src/main/resources**

This folder holds all non-code resources. Key files include:

* application.properties: For all application configuration.
* static: For assets like CSS, JavaScript, and images.
* templates: For view templates (e.g., Thymeleaf or FreeMarker).

**3. src/test/java**

* This folder contains all source code for testing the application, including both unit and integration tests.

**4. SpringLearnApplication.java Walkthrough**

* This class is the entry point for the application.
* The main() method calls SpringApplication.run(...), which bootstraps the entire application, creates the Spring container (ApplicationContext), and starts the embedded web server.

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**5. Purpose of @SpringBootApplication Annotation**

This is a core convenience annotation that combines three other annotations:

* @Configuration: Marks the class as a source of bean definitions.
* @EnableAutoConfiguration: Enables Spring Boot's intelligent auto-configuration mechanism.
* @ComponentScan: Enables component scanning in the current package and its sub-packages.

**6. pom.xml**

* The pom.xml (Project Object Model) is the central configuration file for Maven.

1. Walkthrough all the configuration defined in XML file

* <parent>: Inherits default settings from the spring-boot-starter-parent.
* <groupId>, <artifactId>, <version>: Defines the project's unique Maven coordinates.
* <dependencies>: Lists all required project libraries, such as spring-boot-starter-web.
* <build>: Contains plugins used to build the project, notably the spring-boot-maven-plugin which creates the executable JAR.

2. Open 'Dependency Hierarchy' and show the dependency tree

* This view in Eclipse shows a tree of all project dependencies. It displays both the *direct dependencies* explicitly listed in the pom.xml and the *transitive dependencies* they bring in automatically. This is essential for managing libraries and resolving version conflicts.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>