**Spring Web Project Setup Using Maven in Eclipse**

**1. Generate the Project Using Spring Initializr**

1. Go to [Spring Initializr](https://start.spring.io/).
2. Configure the Project:
   * Group: com.cognizant
   * Artifact Id: spring-learn-v2 (or any custom name like spring-learn-1 to avoid conflicts with existing projects)
   * Dependencies: Select Spring Boot DevTools and Spring Web.
3. Click Generate to download the project as a .zip file.

**2. Extract the Project**

1. Extract the .zip file you downloaded to your desired directory on your local machine.

**3. Open the Project in Eclipse**

1. Open Eclipse IDE.
2. Go to File > Import > Maven > Existing Maven Projects.
3. Click Browse, navigate to the folder where you extracted the .zip file, and select the project folder (e.g., spring-learn-v2).
4. Click Finish to import the project.

**4. Build the Project Using Maven**

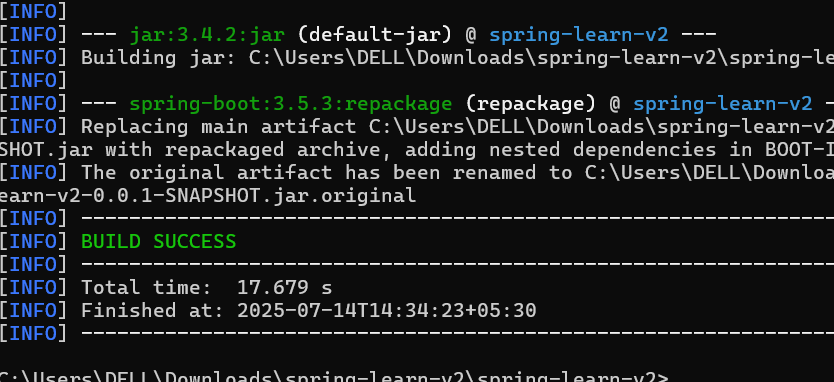
1. Open Command Prompt or Terminal.
2. Navigate to the project folder where the pom.xml file is located. For example:
3. cd C:\Users\DELL\Downloads\spring-learn-v2\spring-learn-v2
4. Run the following Maven build command:

If behind a proxy:

**mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 \**

**-Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456**

Wait for the command to complete and check for a BUILD SUCCESS message in the terminal.



**5. Add Logs to Verify main() Method in SpringLearnApplication.java**

1. Go to src/main/java/com/cognizant/springlearn/SpringLearnApplication.java.
2. Modify the class as follows:
3. package com.cognizant.springlearn;
4. import org.springframework.boot.SpringApplication;
5. import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

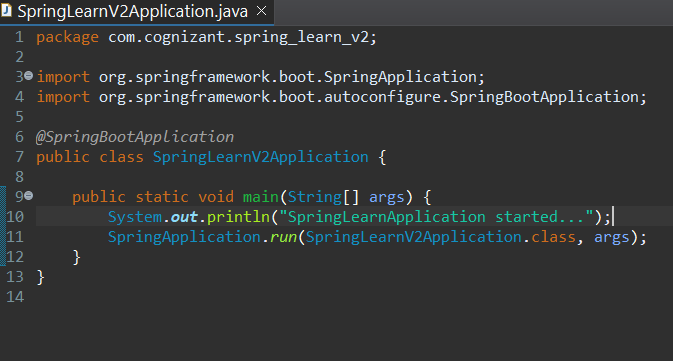
System.out.println("SpringLearnApplication started...");

SpringApplication.run(SpringLearnApplication.class, args);

}

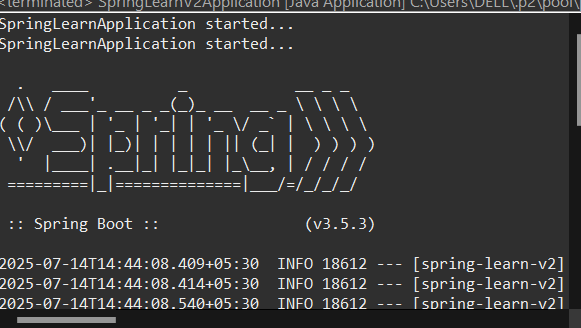
}

Save the file.



**7. Run the Spring Boot Application**

1. Right-click on the SpringLearnApplication.java file.
2. Select Run As > Java Application.
3. The application should start, and you will see the log message SpringLearnApplication started... in the Console of Eclipse.



**SME Walkthrough**

**1. src/main/java - Folder with Application Code**

* Contains all your application logic such as controllers, services, and models.
* For example, SpringLearnApplication.java is the main entry point.

**2. src/main/resources - Folder for Application Configuration**

* This folder holds the configuration files like application.properties or application.yml, static resources (e.g., CSS, JS), and templates (if using Thymeleaf).

**3. src/test/java - Folder for Testing**

* Contains test classes where you can write unit tests to verify your application's functionality using JUnit.

**4. SpringLearnApplication.java - Walkthrough of main() Method**

* The main() method serves as the entry point for your Spring Boot application. The @SpringBootApplication annotation is used to configure and run the application.
* The method SpringApplication.run(SpringLearnApplication.class, args) starts the application and initializes the Spring context.

**5. Purpose of @SpringBootApplication Annotation**

* The @SpringBootApplication annotation is a convenience annotation that includes:
  + @Configuration: Indicates the class is a Spring configuration class.
  + @EnableAutoConfiguration: Enables Spring Boot’s automatic configuration.
  + @ComponentScan: Scans the current package and subpackages for Spring components like @Controller, @Service, etc.

**6. pom.xml Configuration Walkthrough**

* The pom.xml file contains the configuration for Maven, including dependencies and build configurations.

**Key Sections in pom.xml:**

* Dependencies:
  + spring-boot-starter-web: Includes libraries for building web applications (Spring MVC, Tomcat, Jackson for JSON).
  + spring-boot-devtools: Provides development tools like automatic restarts and live reload.
* Maven Plugins:
  + spring-boot-maven-plugin: Packages the application as a runnable JAR.

Example dependency in pom.xml:

<dependency>

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

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

</dependency>

]

**7. Viewing the Dependency Tree in Eclipse**

To view the Maven dependency tree in Eclipse:

1. Open pom.xml in the Project Explorer.
2. Go to the Maven view in Eclipse.
3. Click on Dependency Hierarchy to view a graphical representation of all dependencies.

Alternatively, you can use the following Maven command in the terminal to view the dependency tree:

mvn dependency:tree