Hands-on 1: Create a Spring Web Project using Maven

# 1. Introduction

This document describes the steps followed to create a Spring Web Project using Maven, as per the given hands-on exercise. The objective is to set up a basic Spring Boot application named 'spring-learn' with essential dependencies and understand the structure and configuration.

# 2. Step-by-Step Execution

1. 1. Visit https://start.spring.io/ to generate a new Spring Boot project.
2. 2. Set Group as 'com.cognizant' and Artifact as 'spring-learn'.
3. 3. Add dependencies: Spring Boot DevTools and Spring Web.
4. 4. Generate and download the project zip file.
5. 5. Extract the zip file to Intellij workspace folder.
6. 6. Build the project using the command:  
   mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456
7. 7. Import the project in Intellij: File > Import > Maven > Existing Maven Projects.
8. 8. Edit SpringLearnApplication.java to add log statements in the main() method.
9. 9. Run SpringLearnApplication.java and verify the logs in the console.

# 3. Project Structure

The following directories are present in the project:

• src/main/java - Contains the main application code including the entry point.

• src/main/resources - Used for application configuration (e.g., application.properties).

• src/test/java - Contains unit and integration test classes.

# 4. SpringLearnApplication.java

This class serves as the entry point for the Spring Boot application. The main() method uses SpringApplication.run() to bootstrap the application. Logging statements were added to verify execution.

# 5. @SpringBootApplication Annotation

@SpringBootApplication is a meta-annotation that includes:  
• @Configuration - Marks the class as a source of bean definitions.  
• @EnableAutoConfiguration - Enables Spring Boot’s auto-configuration mechanism.  
• @ComponentScan - Enables component scanning for current and sub-packages.

# 6. pom.xml Configuration

The pom.xml includes metadata and dependencies such as:

• spring-boot-starter-web - For building web applications.

• spring-boot-devtools - Enables auto-restart during development.

• spring-boot-starter-test - Used for testing purposes.

Dependency hierarchy can be viewed in Eclipse by right-clicking on the project → Maven → Show Dependency Hierarchy.

**7. Output**

When we run the main class the server get started at the port 8080 of localhost and in logs we get –

“Spring Boot Started” which is defined by us in main class as for logging.

# 8. Conclusion

The Spring Web project was successfully created, built, and executed using Maven. The project structure, dependencies, and annotations were explored to understand the functioning of a Spring Boot application.