SME Walkthrough for Spring Web Project (spring-learn)

# 1. src/main/java

This is the main source folder containing your Java source code. It includes your SpringLearnApplication.java (the main class) and other packages like controllers, services, and models as you build the project.

# 2. src/main/resources

This folder contains application configuration files, like:  
- application.properties or application.yml: Used to configure port, database settings, logging, etc.  
- Static files (e.g., HTML, JS, CSS) and templates (e.g., Thymeleaf) also go here.

# 3. src/test/java

Contains test cases written using frameworks like JUnit or Mockito. It's structured similar to src/main/java but used solely for unit/integration testing.

# 4. SpringLearnApplication.java

This is the entry point of the Spring Boot application. It contains the main() method:  
  
public static void main(String[] args) {  
 SpringApplication.run(SpringLearnApplication.class, args);  
}  
  
Running this launches the embedded Tomcat server and initializes the Spring context.

# 5. Purpose of @SpringBootApplication

A meta-annotation that combines:  
- @Configuration: Marks the class as a source of bean definitions.  
- @EnableAutoConfiguration: Enables auto-configuration of Spring Boot.  
- @ComponentScan: Scans the base package for components like controllers, services, etc.  
  
This annotation makes Spring Boot applications quick to configure and launch.

# pom.xml Explanation

## Key Sections:

- <groupId>: Logical grouping — com.cognizant  
- <artifactId>: The actual project name — spring-learn  
- <dependencies>: Includes libraries like spring-boot-starter-web, spring-boot-devtools, etc.  
- <build>: Includes plugin config like spring-boot-maven-plugin for building fat JARs.

## Common 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>  
</dependency>

# Dependency Hierarchy (in Eclipse)

Go to the pom.xml tab.  
Click on 'Dependency Hierarchy' view.  
This view shows the tree structure of all dependencies.  
- Helps in identifying transitive dependencies.  
- Allows you to check for conflicts (e.g., multiple versions of same library).