SME Walkthrough: Project Structure and Configuration

(1) src/main/java

- Contains all your **Java source code** (controllers, services, etc.)
- Starts with the **main application class** (SpringLearnApplication.java)

(2) src/main/resources

Holds **configuration and static resources**:

- application.properties or application.yml
- Templates (Thymeleaf, etc.)
- Static files (CSS, JS, images)

(3) src/test/java

- Contains **unit and integration tests**.
- Follows the same package structure as src/main/java.

Example:

```java
package com.cognizant.spring_learn;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class SpringLearnApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

(4) SpringLearnApplication.java

- Entry point of the Spring Boot app.
- Contains main() method.
- Example Code:

```java
public static void main(String[] args) {
        LOGGER.info("START");
        SpringApplication.run(SpringLearnApplication.class, args);
        LOGGER.info("END");
    }
```

5)Purpose of @SpringBootApplication

It is a combination of 3 annotations:

- @Configuration – Allows registering Spring Beans via Java config
- @EnableAutoConfiguration – Enables Spring Boot's auto configuration
- @ComponentScan – Scans the package for components (beans, controllers, etc.)

6)pom.xml Walkthrough:

Maven configuration file.
Manages:

- Project metadata (groupId, artifactId, version)
- Dependencies (e.g., Spring Web, DevTools)
- Plugins and build instructions

Example code snippet:

```
<groupId>com.cognizant</groupId>
<artifactId>spring-learn</artifactId>
<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>
</dependencies>
```

7)Dependency Hierarchy in Eclipse