SME Walkthrough: Project Structure and Configuration

(1) src/main/java

- Contains your application's source code.
- Business logic, controllers, services go here.
- Example: You can create REST controllers here to handle HTTP requests using Spring Web.

(2) src/main/resources

- Stores configuration files and static resources.
- static/: Stores static assets like CSS, JavaScript, and images.
- templates/: Used for server-side templates.
- Example: application.properties - spring.application.name=spring-learn

(3) src/test/java

- Contains unit and integration tests.
- JUnit/TestNG tests are placed here.
- Example Code:

```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

- Located at src/main/java/com/cognizant/spring_learn
- Entry point of the Spring Boot application.
- Contains the main() method which boots the app.
- 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

Example code snippet:

```
<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>
</dependencies>
```

(7) Dependency Hierarchy in Eclipse