Week 3 – 1. Spring-Rest-Handson

Spring Web Project - Hands-on Summary

Project Setup using Spring Initializr

Group: com.cognizant

Artifact: spring-learn

Dependencies

Selected:Spring Web

Spring Boot DevTools

Packaging: Maven

Java Version: 17

Project generated from https://start.spring.io

Set Project SDK to JDK 17

Code:

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

public static void main(String[] args) { SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("SpringLearnApplication started...");}

SME Walkthrough

1. 📂 src/main/java

Contains all the main application source code

Folder structure:

com.cognizant.springlearn └── SpringLearnApplication.java

2. src/main/resources\

Contains configuration files

application.properties is placed here (empty for now)

3. src/test/java

Contains unit/integration test classes

Auto-generated class: SpringLearnApplicationTests.java

4. SpringLearnApplication.java

Entry point for the Spring Boot application

Uses:

SpringApplication.run(SpringLearnApplication.class, args);

5. Purpose of @SpringBootApplication

@SpringBootApplication is a meta-annotation that combines:

@Configuration – marks this class as source of bean definitions

@EnableAutoConfiguration – enables Spring Boot’s auto-configuration

@ComponentScan – tells Spring to scan current package and sub-packages for components

6. pom.xml ConfigurationIncludes:

Project metadata (groupId, artifactId, version)

Dependencies:

spring-boot-starter-web

spring-boot-devtools

Java version:

<properties>

<java.version>17</java.version>

</properties>

7. Dependency Hierarchy

Viewed using IntelliJ:

Opened Maven tool window

Right-clicked on project > Show Dependency Diagram

Shows transitive dependencies like:

Spring Boot

Jackson

Embedded Tomcat

Logging (SLF4J, Logback)

Final Status

Build: Success

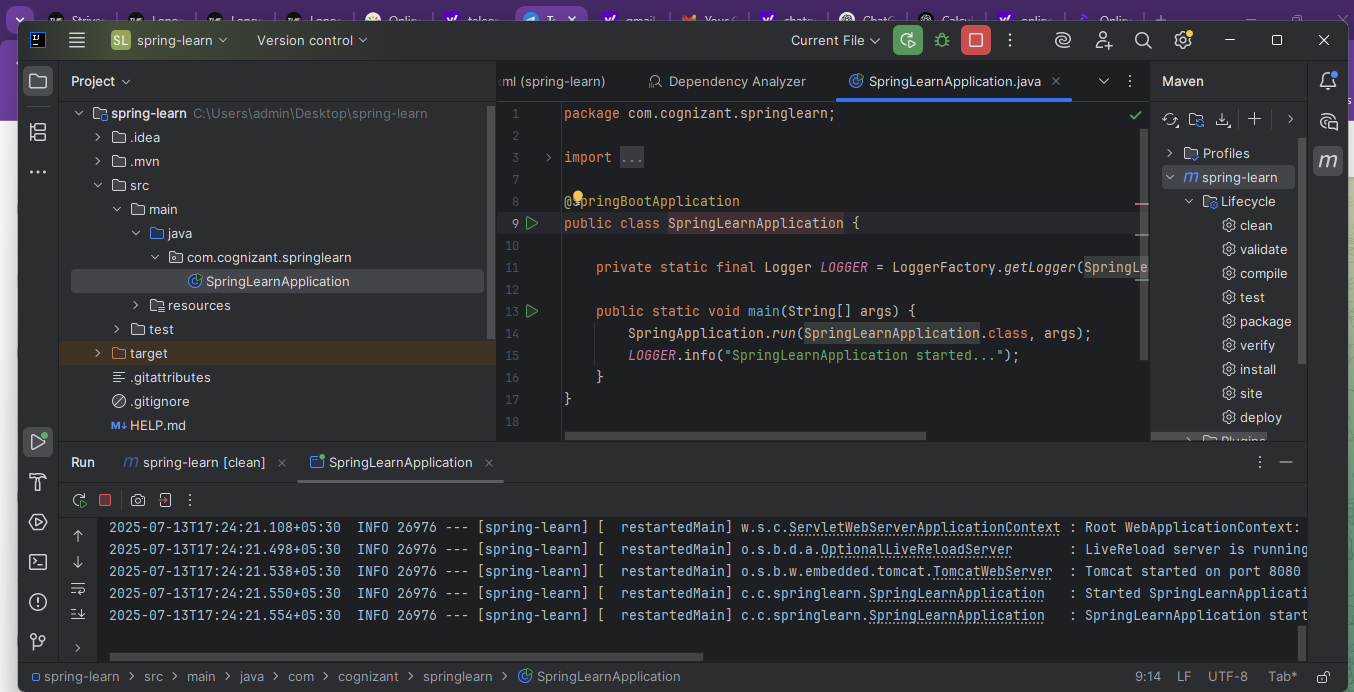
App: Runs Successfully

Logging: Verified

Maven: Dependencies Resolved

IntelliJ: Setup with JDK 17

Output:



Hands-On (additional)

Spring Core – Load SimpleDateFormat from Spring Configuration XML

date-format.xml

<bean id="dateFormat" class="java.text.SimpleDateFormat"> <constructor-arg value="dd/MM/yyyy" />

</bean>

displayDate()

public static void displayDate() throws ParseException { ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

Date date = format.parse("31/12/2018");

System.out.println(date);

}

Called Method from main()java

public static void main(String[] args) throws ParseException {

SpringApplication.run(SpringLearnApplication.class, args);

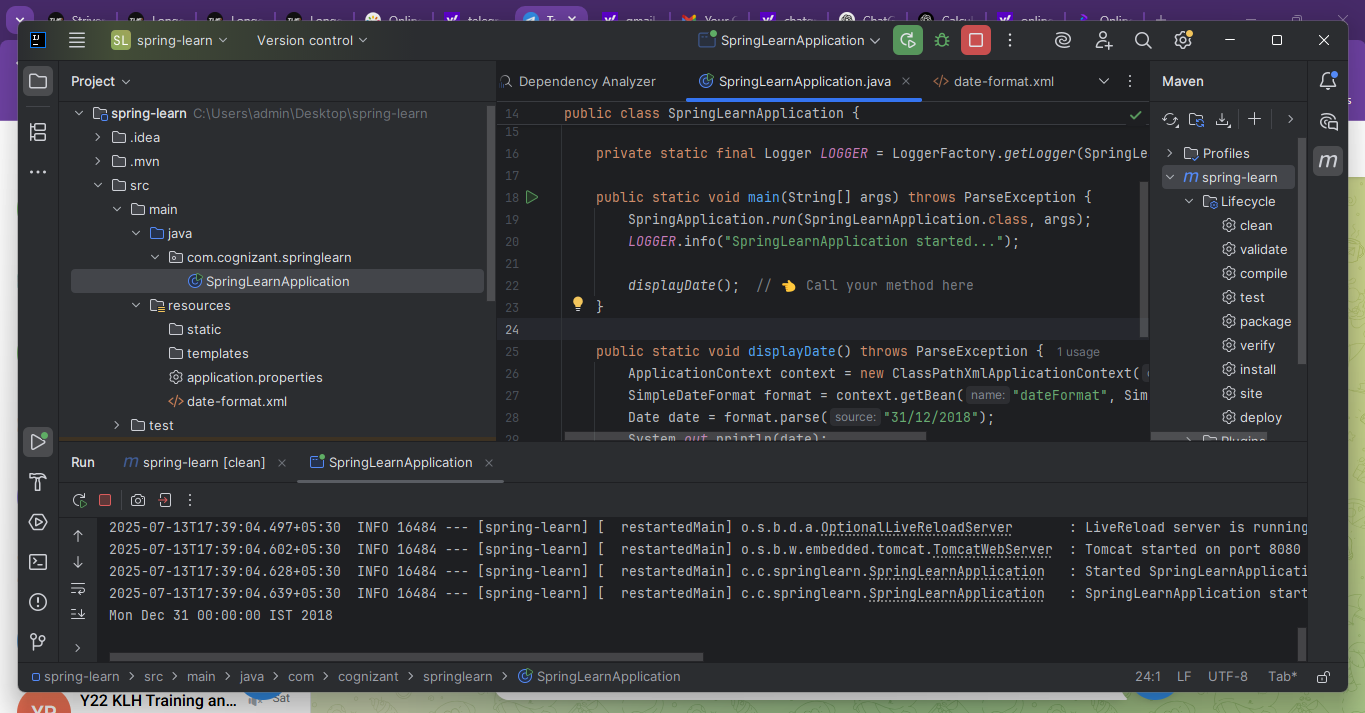
LOGGER.info("SpringLearnApplication started...");

displayDate();

}

Spring XML Bean Configuration

Output:



Hands-On -Spring Core – Load Country from Spring Configuration XML

country.xml

<bean id="country" class="com.cognizant.springlearn.Country"> <property name="code" value="IN" />

<property name="name" value="India" /></bean>

Country.java Class

public Country() {

LOGGER.debug("Inside Country Constructor.");}

public void setCode(String code) {

LOGGER.debug("Inside setCode()"); this.code = code;

}

3 Method displayCountry() in SpringLearnApplication.java

public static void displayCountry() { ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = context.getBean("country", Country.class); LOGGER.debug("Country : {}", country.toString());

}

Called displayCountry() from main() Method

displayDate();displayCountry();

output:

