**Hands-on 1: Creating a Spring Web Project with Maven**

**1. Project Creation using Spring Initializr**

1. Open your browser and navigate to <https://start.spring.io/>.
2. Set the following fields:
   * **Group:** com.cognizant
   * **Artifact:** spring-learn
3. Add the following dependencies:
   * Spring Web
   * Spring Boot DevTools
4. Click **Generate** to download the starter project as a ZIP file.

**2. Extracting and Preparing the Project**

* Extract the downloaded spring-learn.zip to your Eclipse workspace or any organized project directory for easy management.

**3. Building the Project Using Maven**

Before importing, perform a build to pull dependencies and ensure the project compiles correctly.

Open **Command Prompt / Terminal**, navigate to your extracted project folder, and execute:

go

CopyEdit

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

This command:

* Cleans previous builds.
* Downloads required dependencies through your corporate proxy.
* Packages the project into a JAR.

**4. Importing the Project into Eclipse**

1. Open Eclipse IDE.
2. Navigate to **File > Import > Maven > Existing Maven Projects**.
3. Browse and select your extracted spring-learn folder.
4. Click **Finish** to import the project.

**5. Understanding the Project Structure**

Once imported, you will observe the following structure:

* **src/main/java**: Contains your main application code, including controllers, services, and your main entry point.
* **src/main/resources**: Holds configuration files like application.properties.
* **src/test/java**: Contains test cases and test configurations.
* **pom.xml**: The Maven Project Object Model file that manages dependencies, plugins, and project configurations

**6. Main Application File Walkthrough**

Locate SpringLearnApplication.java under:

src/main/java/com/cognizant/springlearn/

**Code Example:**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("Application is starting...");

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Key Points:**

* This class contains the main() method, which launches the Spring Boot application.
* System.out.println is included to verify when the application starts.

**7. Role of @SpringBootApplication**

The @SpringBootApplication annotation:

* Marks the class as the entry point for the Spring Boot application.
* Enables component scanning, auto-configuration, and Spring configuration in a single annotation.
* Simplifies bootstrapping the application with minimal manual setup.

**8. Exploring pom.xml**

The pom.xml file handles dependency management and project configurations.

**Highlights:**

* Includes dependencies for:
  + spring-boot-starter-web for REST APIs and embedded Tomcat.
  + spring-boot-devtools for hot-reloading during development.
  + spring-boot-starter-test for unit and integration testing.
* Uses spring-boot-starter-parent for version management and build configurations.

**Example snippet:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**9. Viewing Dependency Hierarchy**

To understand which libraries are included:

1. Right-click the spring-learn project in Eclipse.
2. Navigate to **Maven > Show Dependency Hierarchy**.
3. Observe how spring-boot-starter-web brings in dependencies like Spring MVC, Tomcat, and Jackson.

**10. Running the Application**

1. In Eclipse, locate SpringLearnApplication.java.
2. Right-click and select **Run As > Java Application**.
3. Check the console for:

Application is starting...

followed by Spring Boot startup logs indicating the embedded Tomcat server running on port 8080.

**OUTPUT**:  
  


# Hands-on 4: Spring Core – Load Country from Spring Configuration XML

## 1. country.xml (Spring XML Configuration)

**Location:** src/main/resources/country.xml

xml

CopyEdit

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.springlearn.Country">

<property name="code" value="IN" />

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

</bean>

</beans>

## 2. Country.java

**Location:** src/main/java/com/cognizant/springlearn/Country.java

java

CopyEdit

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

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

private String code;

private String name;

public Country() {

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

}

public String getCode() {

LOGGER.debug("Inside getCode()");

return code;

}

public void setCode(String code) {

LOGGER.debug("Inside setCode()");

this.code = code;

}

public String getName() {

LOGGER.debug("Inside getName()");

return name;

}

public void setName(String name) {

LOGGER.debug("Inside setName()");

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

## 3. SpringLearnApplication.java

**Location:** src/main/java/com/cognizant/springlearn/SpringLearnApplication.java

java

CopyEdit

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void displayCountry() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = context.getBean("country", Country.class);

LOGGER.debug("Country: {}", country.toString());

}

public static void main(String[] args) {

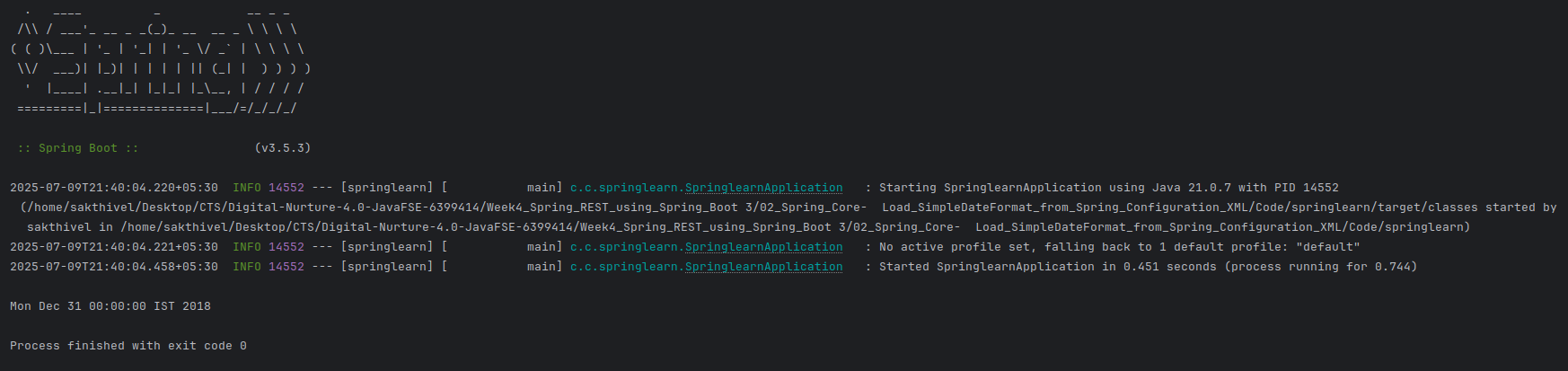
LOGGER.debug("Inside main");

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

}

}

**OUTPUT:  
**