# **✅ Hands-On 1: Create a Spring Web Project Using Maven**

## **Steps to Create the Project**

1. Open <https://start.spring.io/>
2. Set:
   1. **Group:** com.cognizant
   2. **Artifact:** spring-learn
3. Add **Dependencies**:
   1. Spring Web
   2. Spring Boot DevTools
4. Click **Generate** to download the project as a .zip file.
5. Extract the zip file to your **Eclipse workspace directory**.
6. Open a command line and navigate to the extracted project root.
7. Run the following Maven build command (replace proxy user ID as needed):

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

1. Open Eclipse:
   1. Go to **File > Import > Maven > Existing Maven Projects**
   2. Click **Browse** and select the extracted folder
   3. Click **Finish**
2. Open SpringLearnApplication.java and add logs in the main() method to verify execution:

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);  
  
public static void main(String[] args) {  
 LOGGER.info("START");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 LOGGER.info("END");  
}

1. Right-click on the file and choose **Run As > Java Application**.

## **SME Walkthrough Topics**

### **1. src/main/java**

* Folder containing all **application source code**, including controllers, services, and entity classes.

### **2. src/main/resources**

* Contains **application configuration** files like application.properties or .yml.

### **3. src/test/java**

* Folder dedicated for **unit and integration tests**.

### **4. SpringLearnApplication.java**

* Contains the **entry point** for Spring Boot.
* Uses SpringApplication.run() to bootstrap the application.

### **5. @SpringBootApplication**

* A **meta-annotation** combining:
  + @Configuration: Marks class as a source of bean definitions.
  + @EnableAutoConfiguration: Enables Spring Boot's auto-configuration.
  + @ComponentScan: Scans the package for components.

### **6. pom.xml**

* The **Maven configuration file** that:
  + Declares project metadata
  + Includes dependencies (like Spring Web)
  + Configures build plugins

### **7. Dependency Hierarchy**

* Go to Eclipse: Right-click on project > Maven > Show Dependency Hierarchy.
* This displays the **tree of dependencies**, including transitive ones.

# **✅ Hands-On 4: Spring Core – Load Country from Spring Configuration XML**

## **🎯 Objective**

Create a Spring XML configuration to load a **Country** bean and read it using Spring's ApplicationContext.

## **🔧 Steps to Implement**

### **1. country.xml (in src/main/resources)**

<?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  
 <https://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**

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("Getting Country Code");  
 return code;  
 }  
  
 public void setCode(String code) {  
 LOGGER.debug("Setting Country Code");  
 this.code = code;  
 }  
  
 public String getName() {  
 LOGGER.debug("Getting Country Name");  
 return name;  
 }  
  
 public void setName(String name) {  
 LOGGER.debug("Setting Country Name");  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

### **3. SpringLearnApplication.java – Add displayCountry() Method**

public static void displayCountry() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = context.getBean("country", Country.class);  
 LOGGER.debug("Country : {}", country.toString());  
}

### **4. Call displayCountry() from main()**

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