**1.CREATE A SPRING WEB PROJECT USING MAVEN**

CONTEXT OF THE EXAMPLE:

In this exercise, I created a basic Spring Boot web project using Maven. I added essential dependencies (Spring Web, DevTools), configured the main application class, and verified successful setup using logging. The output confirms that the application runs as expected.

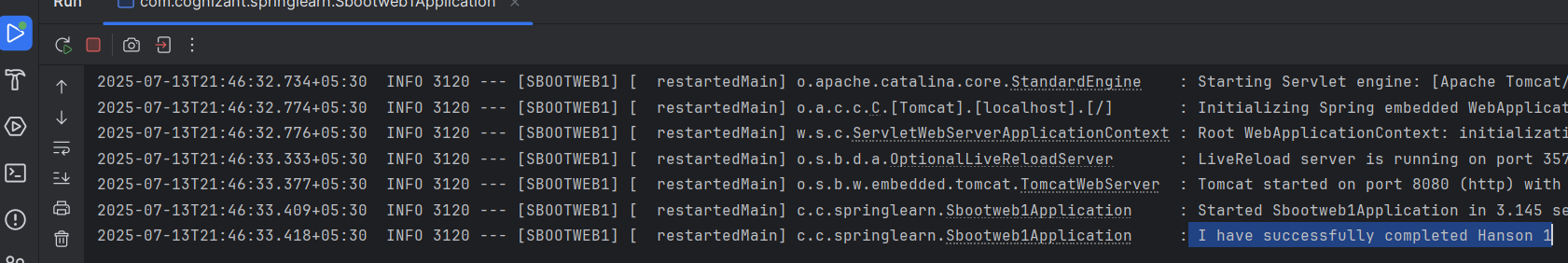
pom.xml

The pom.xml file includes dependencies for **Spring Boot Web** and **DevTools**. These enable auto-configuration for web applications and allow hot reloading during development. No additional manual configurations are required in this case.

Sbootweb1Application.java

package com.dinesh.sbootweb1; // Or your actual package name  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Sbootweb1Application {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(Sbootweb1Application.class);  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Sbootweb1Application.class, args);  
 *LOGGER*.info("I have successfully completed Hanson 1");  
 }  
}

OUTPUT:



**2. SPRING CORE – LOAD SIMPLEDATEFORMAT FROM SPRING CONFIGURATION XML**

CONTEXT OF THE EXAMPLE:

This example demonstrates loading a SimpleDateFormat bean from Spring's XML configuration and parsing a date string into a Date object using dependency injection.

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Sboot2</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Sboot2</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

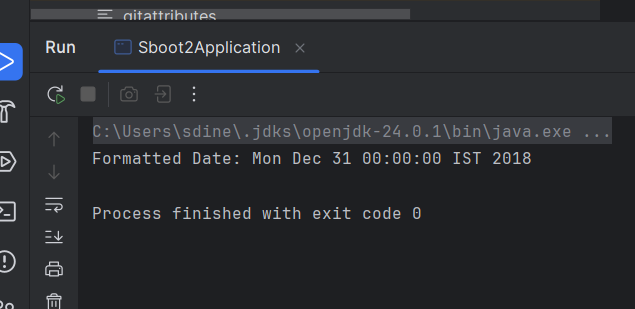
dateformat.xml

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

Sboot2Application.java

package com.dinesh.Sboot2;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Sboot2Application {  
  
 public static void main(String[] args) throws Exception {  
 *displayDate*();  
 }  
  
 private static void displayDate() throws Exception {  
 ApplicationContext context = new ClassPathXmlApplicationContext("dateformat.xml");  
  
 SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  
 Date date = format.parse("31/12/2018");  
  
 System.*out*.println("Formatted Date: " + date);  
 }  
}

OUTPUT:



**3.SPRING CORE – LOAD COUNTRY FROM SPRING CONFIGURATION XML**

CONTEXT OF THE EXAMPLE:

This exercise demonstrates how to use Spring Core to define and load a simple object (Country) from an external XML configuration file. It helps eliminate hardcoded values in Java code by externalizing configuration, enabling better reusability and maintainability.

pom.xml:

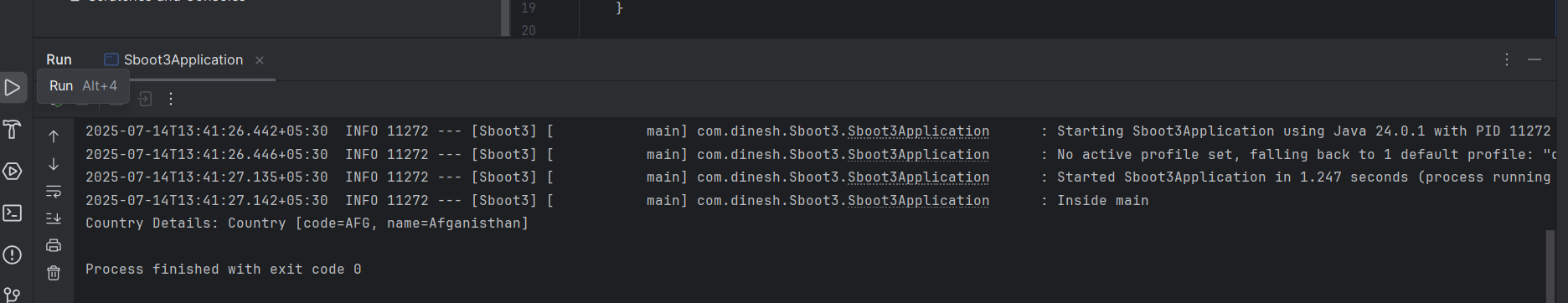
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Sboot3</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Sboot3</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

country.xml:

<?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.dinesh.Sboot3.Country">  
 <property name="code" value="AFG"/>  
 <property name="name" value="Afganisthan"/>  
 </bean>  
  
</beans>

Country.java:

package com.dinesh.Sboot3;  
  
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 + "]";  
 }  
}

OUTPUT:

**4.Hello World-RESTful Web Service**

CONTEXT OF THE EXAMPLE:

This example demonstrates a simple RESTful service using Spring Boot, which handles an HTTP GET request and returns a plain text response. It illustrates how to define a controller, map URLs, and return data using the Spring Web framework.

pom.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Sboot4</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Sboot4</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <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>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

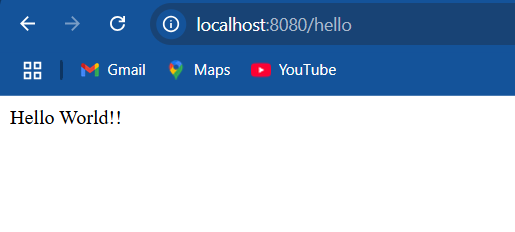
hellocontroller.java:

package com.dinesh.Sboot4;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class hellocontroller {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(hellocontroller.class);  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 *LOGGER*.info("Inside sayHello()");  
 return "Hello World!!";  
 }  
}

Sboot4Application.java:

package com.dinesh.Sboot4;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class Sboot4Application {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(Sboot4Application.class);  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Sboot4Application.class, args);  
 *LOGGER*.info("I have successfully completed Handson 4");  
 }  
}

OUTPUT:



**5. REST - COUNTRY WEB SERVICE**

CONTEXT OF THE EXAMPLE:

pom.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Sboot5</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Sboot5</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <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>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Country.java:

package com.dinesh.Sboot5;  
  
public class Country {  
 private String code;  
 private String name;  
  
 public Country() {  
 System.*out*.println("Inside Country Constructor");  
 }  
  
 public String getCode() {  
 System.*out*.println("Inside getCode()");  
 return code;  
 }  
  
 public void setCode(String code) {  
 System.*out*.println("Inside setCode()");  
 this.code = code;  
 }  
  
 public String getName() {  
 System.*out*.println("Inside getName()");  
 return name;  
 }  
  
 public void setName(String name) {  
 System.*out*.println("Inside setName()");  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

country.xml

package com.dinesh.Sboot5;  
  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Sboot5Application {  
 public static void main(String[] args) {  
 SpringApplication.*run*(Sboot5Application.class, args);  
 }  
}

CountryController.java:

package com.dinesh.Sboot5.controller;  
  
  
import com.dinesh.Sboot5.Country;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class CountryController {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryController.class);  
  
 @RequestMapping("/country")  
 public Country getCountryIndia() {  
 *LOGGER*.info("START - getCountryIndia()");  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = (Country) context.getBean("country", Country.class);  
 *LOGGER*.debug("Country: {}", country);  
 *LOGGER*.info("END - getCountryIndia()");  
 return country;  
 }  
}

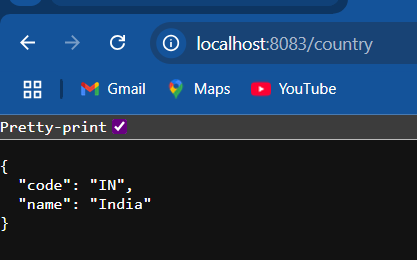
application.properties:

server.port=8083

Sboot5Application.java:

package com.dinesh.Sboot5;  
  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Sboot5Application {  
 public static void main(String[] args) {  
 SpringApplication.*run*(Sboot5Application.class, args);  
 }  
}

OUTPUT:



**6. REST - GET COUNTRY BASED ON COUNTRY CODE**

CONTEXT OF THE EXAMPLE:

In this exercise, we build a **Spring Boot RESTful web service** that accepts a **country code as a path variable** and returns the corresponding **Country object**. The service uses a pre-configured Spring XML file to load country data and fetch the matching country. This helps demonstrate how to combine **Spring REST** and **Spring Core XML-based configuration**.

pom.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Sboot6</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Sboot6</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <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>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

country.xml:

<?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="countryList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <bean class="com.dinesh.Sboot6.model.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="INDIA"/>  
 </bean>  
 <bean class="com.dinesh.Sboot6.model.Country">  
 <property name="code" value="US"/>  
 <property name="name" value="United States"/>  
 </bean>  
 <bean class="com.dinesh.Sboot6.model.Country">  
 <property name="code" value="DE"/>  
 <property name="name" value="Germany"/>  
 </bean>  
 <bean class="com.dinesh.Sboot6.model.Country">  
 <property name="code" value="JP"/>  
 <property name="name" value="Japan"/>  
 </bean>  
 </list>  
 </constructor-arg>  
 </bean>  
  
</beans>

Country.java:

package com.dinesh.Sboot6.model;  
  
public class Country {  
 private String code;  
 private String name;  
  
 public Country() {  
 System.*out*.println("Inside Country Constructor.");  
 }  
  
 public String getCode() {  
 System.*out*.println("Getter for code");  
 return code;  
 }  
  
 public void setCode(String code) {  
 System.*out*.println("Setter for code");  
 this.code = code;  
 }  
  
 public String getName() {  
 System.*out*.println("Getter for name");  
 return name;  
 }  
  
 public void setName(String name) {  
 System.*out*.println("Setter for name");  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

CountryService.java:

package com.dinesh.Sboot6.service;  
  
import com.dinesh.Sboot6.model.Country;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 public Country getCountry(String code) throws Exception {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countryList = context.getBean("countryList", java.util.List.class);  
  
 return countryList.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
 }  
}

CountryController.java:

package com.dinesh.Sboot6.controller;  
  
import com.dinesh.Sboot6.model.Country;  
import com.dinesh.Sboot6.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code) throws Exception {  
 return countryService.getCountry(code);  
 }  
}

Sboot6Application.java:

package com.dinesh.Sboot6;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Sboot6Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Sboot6Application.class, args);  
 System.*out*.println("Sboot6 Application Started...");  
 }  
}

OUTPUT:

