**Spring REST using Spring Boot 3**

**spring-rest-handson**

**Create a Spring Web Project using Maven**

**//SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("LOGGING");

System.out.println("Application Starting...");

System.out.println();

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("LOGGING");

System.out.println("Application Started Successfully.");

System.out.println();

}

}

**//SpringLearnApplicationTest.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("LOGGING");

System.out.println("Application Starting...");

System.out.println();

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("LOGGING");

System.out.println("Application Started Successfully.");

System.out.println();

}

}

**//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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</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>

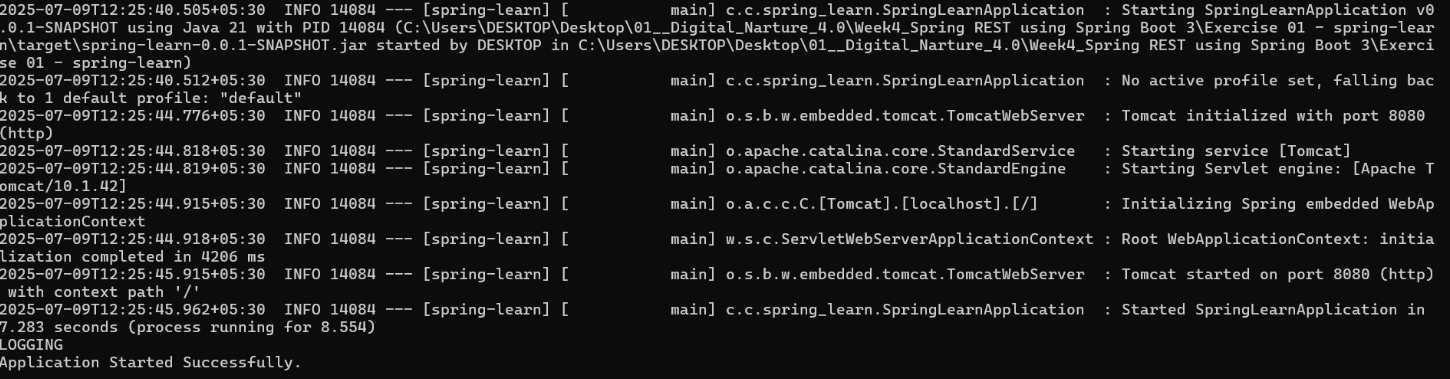
</plugin>

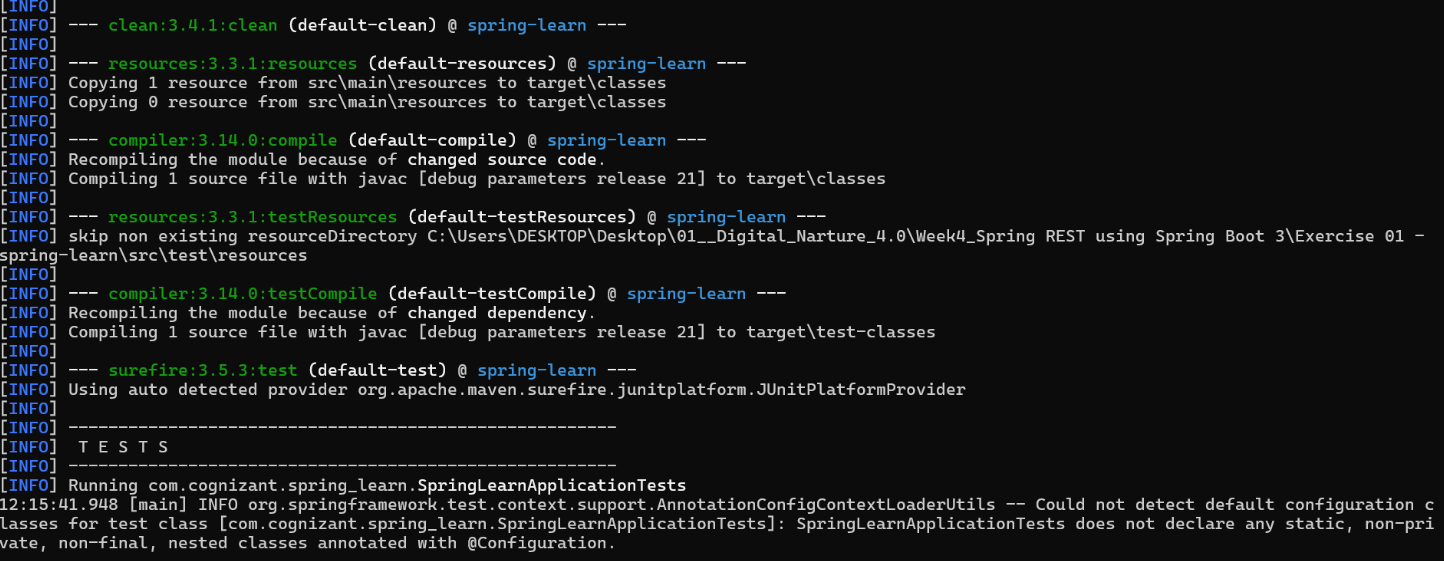
</plugins>

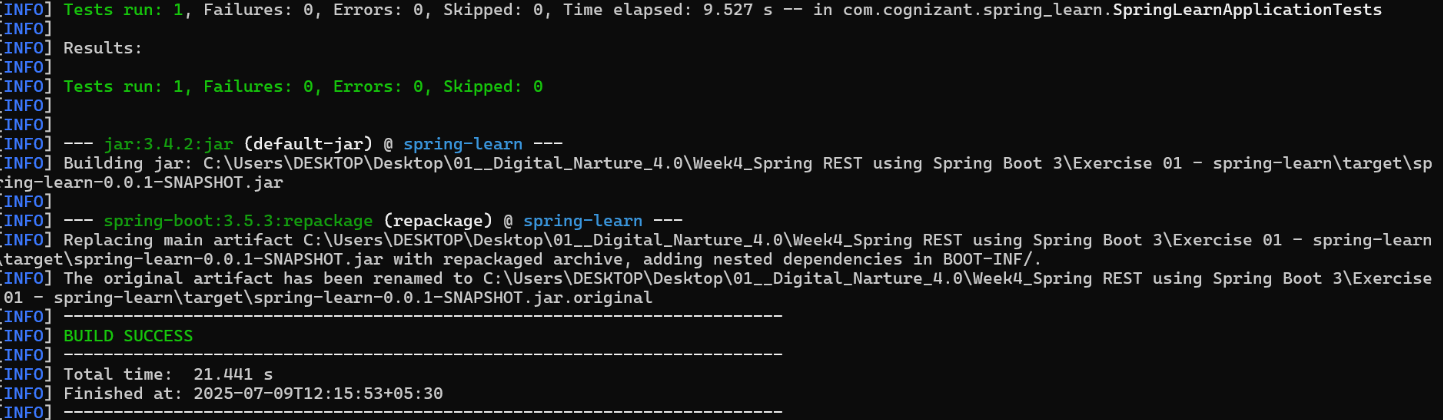
</build>

</project>

**OUTPUT:**







**Spring Core – Load Country from Spring Configuration XML**

**//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("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 + '\'' +

'}';

}

}

**//SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

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

public static void main(String[] args) {

displayCountry();

}

public static void displayCountry() {

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

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

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

}

}

**//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

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>

**//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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant.springlearn</groupId>

<artifactId>SpringCore\_LoadCountry\_from\_SpringConfigurationXML</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>21</maven.compiler.source>

<maven.compiler.target>21</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- SLF4J and Logback for Logging -->

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.1.0-alpha1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/ch.qos.logback/logback-classic -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.5.18</version>

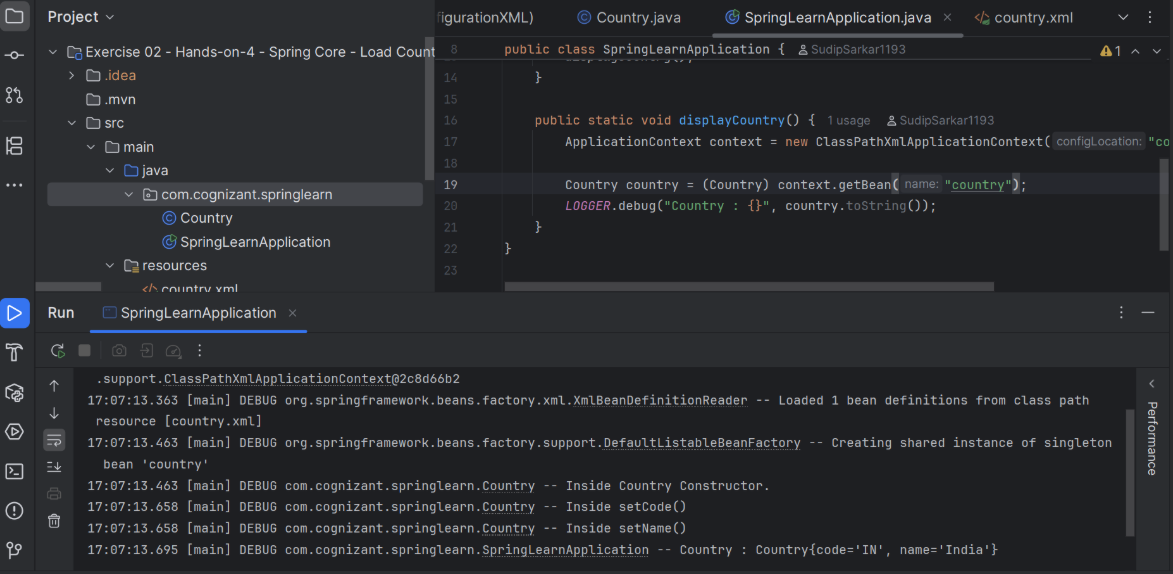
<scope>compile</scope>

</dependency>

</dependencies>

</project>

**OUTPUT:**



**Hello World RESTful Web Service**

**//HelloController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/hello")

public class HelloController {

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

@GetMapping

public String sayHello() {

LOGGER.info("Start of sayHello method");

String response = "Hello World!!";

LOGGER.info("End of sayHello method");

return response;

}

}

**//HelloWorldResTfulWebServiceApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HelloWorldResTfulWebServiceApplication {

public static void main(String[] args) {

SpringApplication.run(HelloWorldResTfulWebServiceApplication.class, args);

}

}

**//HelloWorldResTfulWebServiceApplicationTests.java**

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class HelloWorldResTfulWebServiceApplicationTests {

@Test

void contextLoads() {

}

}

**//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.cognizant.springlearn</groupId>

<artifactId>HelloWorld\_RESTfulWebService</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Hello World RESTful Web Service</name>

<description>Demo project for Spring REST</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</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-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

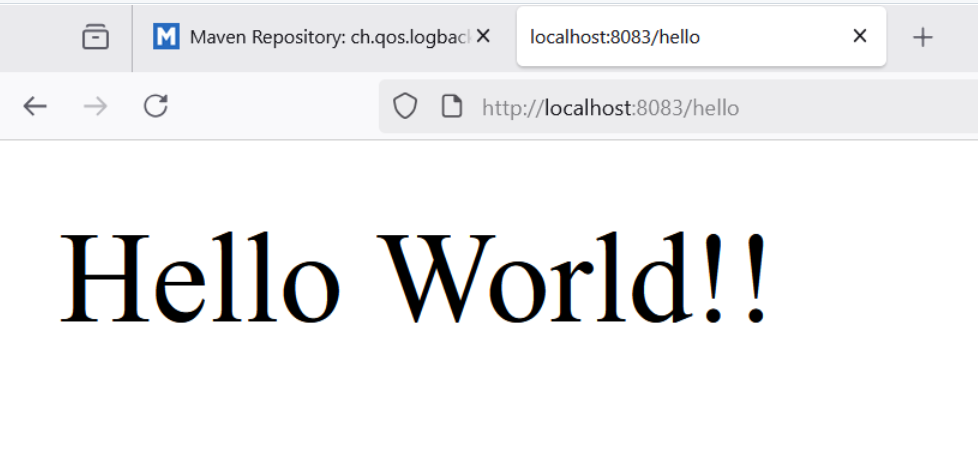
</plugin>

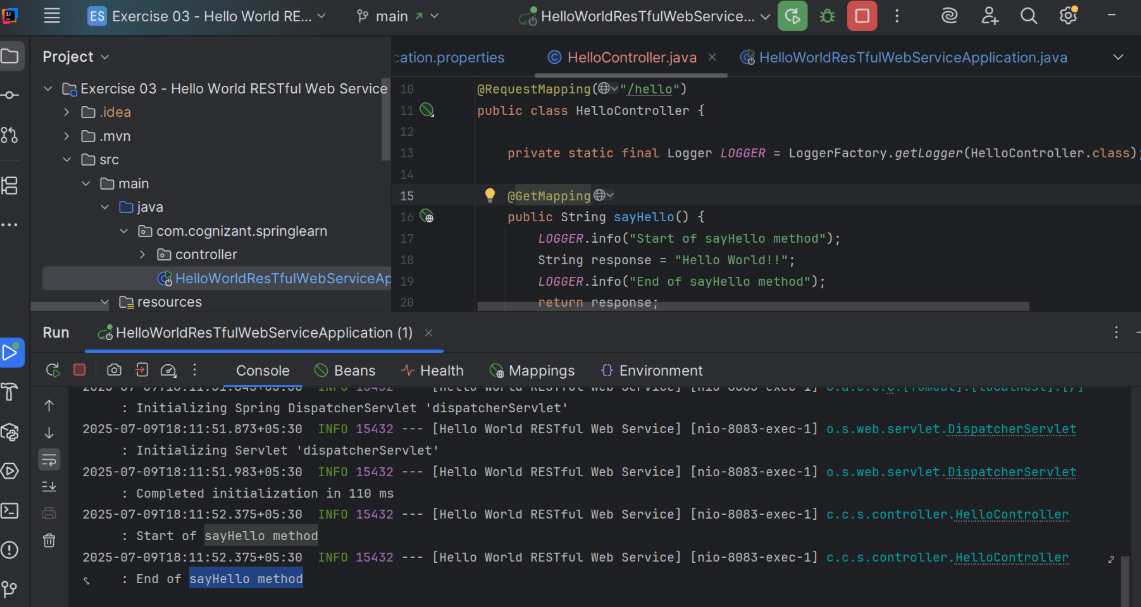
</plugins>

</build>

</project>

**OUTPUT:**





**REST - Country Web Service**

**//CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

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 {

@RequestMapping("/country")

public Country getCountryIndia() {

// Load the Spring XML configuration

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

// Get the India bean

Country country = (Country) context.getBean("india");

return country;

}

}

**//Country.java**

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

// Default constructor

public Country() {

}

// Parameterized constructor

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**//RestCountryWebServiceApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestCountryWebServiceApplication {

public static void main(String[] args) {

SpringApplication.run(RestCountryWebServiceApplication.class, args);

}

}

**//applicationContext.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

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

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

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

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

</bean>

</beans>

**//RestCountryWebServiceApplicationTests.java**

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class RestCountryWebServiceApplicationTests {

@Test

void contextLoads() {

}

}

**//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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>RestCountryWebService</name>

<description>REST Country Web Service</description>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

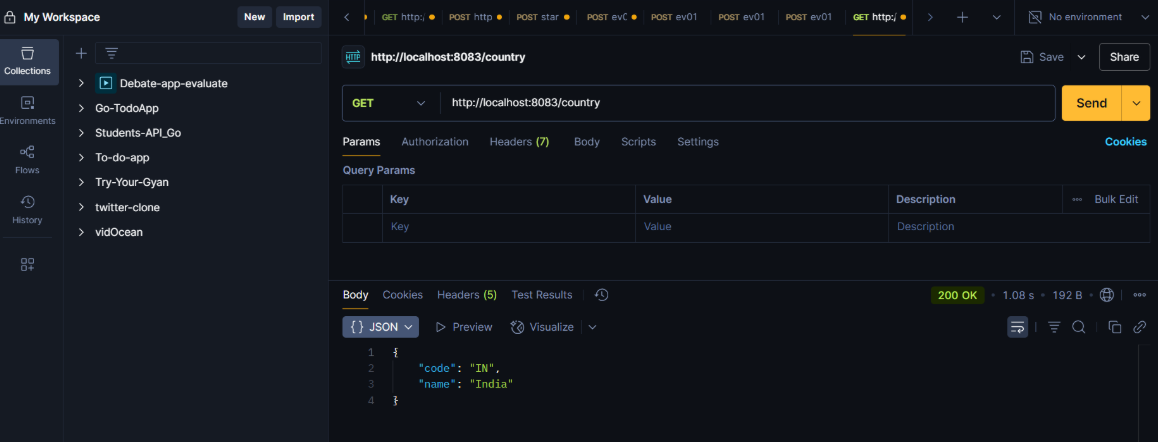
</plugin>

</plugins>

</build>

</project>

**OUTPUT:**



**REST - Get country based on country code**

**//CountryController.java**

package com.cognizant.springlearn.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

/\*\*

\* Country REST Controller

\*

\* Controller: com.cognizant.spring-learn.controller.CountryController

\* Demonstrates RESTful Web Service for getting country by country code

\*/

@RestController

public class CountryController {

private static final Logger logger = LoggerFactory.getLogger(CountryController.class);

@Autowired

private CountryService countryService;

/\*\*

\* REST service that returns a specific country based on country code

\* The country code should be case insensitive

\*

\* Controller: com.cognizant.spring-learn.controller.CountryController

\* Method Annotation: @GetMapping("/countries/{code}")

\* Method Name: getCountry(String code)

\* Method Implementation: Invoke countryService.getCountry(code)

\*

\* Service Method Implementation:

\* • Get the country code using @PathVariable

\* • Get country list from country.xml

\* • Iterate through the country list

\* • Make a case insensitive matching of country code and return the country

\* • Lambda expression is used instead of iterating the country list

\*

\* Sample Request: http://localhost:8083/countries/in

\* Sample Response: { "code": "IN", "name": "India" }

\*

\* @param code Country code (case insensitive)

\* @return Country object if found, 404 if not found

\*/

@GetMapping("/countries/{code}")

public ResponseEntity<Country> getCountry(@PathVariable String code) {

logger.info("START - getCountry() method called with PathVariable code: {}", code);

logger.info("Processing GET request for /countries/{} endpoint", code);

// Invoke countryService.getCountry(code)

Country country = countryService.getCountry(code);

if (country != null) {

logger.info("Country found and returning: {}", country);

logger.info("END - getCountry() method completed - Returning: {}", country);

return ResponseEntity.ok(country);

} else {

logger.warn("Country not found for code: {}", code);

logger.info("END - getCountry() method completed - Returning: 404 Not Found");

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(null);

}

}

/\*\*

\* Get all countries endpoint for testing purposes

\*

\* @return List of all countries

\*/

@GetMapping("/countries")

public ResponseEntity<List<Country>> getAllCountries() {

logger.info("START - getAllCountries() method called");

logger.info("Processing GET request for /countries endpoint");

List<Country> countries = countryService.getAllCountries();

logger.info("Returning {} countries", countries.size());

logger.info("END - getAllCountries() method completed");

return ResponseEntity.ok(countries);

}

/\*\*

\* Simple home endpoint

\*

\* @return Welcome message

\*/

@GetMapping("/")

public String home() {

logger.info("Home endpoint accessed");

return "Welcome to Country Code REST Service! " +

"Use /countries/{code} to get a specific country by code. " +

"Example: /countries/in for India";

}

/\*\*

\* Health check endpoint

\*

\* @return Health status

\*/

@GetMapping("/health")

public String health() {

logger.info("Health check endpoint accessed");

return "Country Code REST Service is UP and running on port 8083!";

}

}

**//Country.java**

package com.cognizant.springlearn.model;

/\*\*

\* Country Model Class

\* Represents a country with code and name

\*/

public class Country {

private String code;

private String name;

// Default constructor

public Country() {

}

// Parameterized constructor

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country{" +

"code='" + code + '\'' +

", name='" + name + '\'' +

'}';

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Country country = (Country) obj;

return code != null ? code.equalsIgnoreCase(country.code) : country.code == null;

}

@Override

public int hashCode() {

return code != null ? code.toUpperCase().hashCode() : 0;

}

}

**//CountryService.java**

package com.cognizant.springlearn.service;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.cognizant.springlearn.model.Country;

/\*\*

\* Country Service Class

\* Service Method: com.cognizant.spring-learn.service.CountryService.getCountry(String code)

\* Handles business logic for country operations

\*/

@Service

public class CountryService {

private static final Logger logger = LoggerFactory.getLogger(CountryService.class);

/\*\*

\* Service Method Implementation:

\* • Get country list from country.xml

\* • Iterate through the country list

\* • Make a case insensitive matching of country code and return the country

\* • Lambda expression is used instead of iterating the country list

\*

\* @param code Country code (case insensitive)

\* @return Country object if found, null otherwise

\*/

@SuppressWarnings("unchecked")

public Country getCountry(String code) {

logger.info("START - getCountry() method called with code: {}", code);

if (code == null || code.trim().isEmpty()) {

logger.warn("Country code is null or empty");

return null;

}

String upperCaseCode = code.toUpperCase().trim();

logger.info("Searching for country with code: {} (original: {})", upperCaseCode, code);

try {

// Get country list from country.xml

logger.info("Loading country list from country.xml");

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

List<Country> countries = (List<Country>) context.getBean("countryList");

logger.info("Successfully loaded {} countries from XML configuration", countries.size());

// Lambda expression for case insensitive matching of country code

Optional<Country> foundCountry = countries.stream()

.filter(country -> country.getCode() != null &&

country.getCode().equalsIgnoreCase(upperCaseCode))

.findFirst();

if (foundCountry.isPresent()) {

Country country = foundCountry.get();

logger.info("Country found: {}", country);

logger.info("END - getCountry() method completed - Returning: {}", country);

return country;

} else {

logger.warn("No country found with code: {}", upperCaseCode);

logger.info("END - getCountry() method completed - Returning: null");

return null;

}

}

} catch (Exception e) {

logger.error("Error in getCountry() method: " + e.getMessage());

logger.info("END - getCountry() method completed with error - Returning: null");

return null;

}

}

/\*\*

\* Get all countries from XML configuration

\* Helper method for demonstration purposes

\*

\* @return List of all countries

\*/

@SuppressWarnings("unchecked")

public List<Country> getAllCountries() {

logger.info("START - getAllCountries() method called");

try {

logger.info("Loading all countries from country.xml");

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

List<Country> countries = (List<Country>) context.getBean("countryList");

logger.info("Successfully loaded {} countries", countries.size());

logger.info("END - getAllCountries() method completed");

return countries;

}

} catch (Exception e) {

logger.error("Error in getAllCountries() method: " + e.getMessage());

return List.of(); // Return empty list on error

}

}

}

**//CountryCodeRestServiceApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

/\*\*

\* Main Spring Boot Application Class for Country Code REST Service

\*/

@SpringBootApplication

public class CountryCodeRestServiceApplication {

private static final Logger logger = LoggerFactory.getLogger(CountryCodeRestServiceApplication.class);

public static void main(String[] args) {

logger.info("=================================================================");

logger.info("Starting Country Code RESTful Web Service Application...");

logger.info("=================================================================");

// Start the Spring Boot application

SpringApplication.run(CountryCodeRestServiceApplication.class, args);

logger.info("=================================================================");

logger.info("Country Code REST Service Application started successfully!");

logger.info("Application is running on: http://localhost:8083");

logger.info("Country endpoints:");

logger.info(" - Get Country by Code: http://localhost:8083/countries/{code}");

logger.info(" - Example: http://localhost:8083/countries/in");

logger.info(" - All Countries: http://localhost:8083/countries");

logger.info(" - Health Check: http://localhost:8083/health");

logger.info("=================================================================");

}

}

**//index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Country Code REST Service</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

.container { max-width: 800px; margin: 0 auto; }

.endpoint { background: #f5f5f5; padding: 15px; margin: 10px 0; border-radius: 5px; }

.method { background: #28a745; color: white; padding: 5px 10px; border-radius: 3px; }

.url { font-family: monospace; background: #e9ecef; padding: 5px; border-radius: 3px; }

.example { background: #d1ecf1; padding: 10px; border-radius: 3px; margin: 10px 0; }

.code { background: #f8f9fa; border: 1px solid #dee2e6; padding: 10px; border-radius: 3px; }

</style>

</head>

<body>

<div class="container">

<h1>Country Code REST Service</h1>

<p>Welcome to the Country Code REST Service! This service allows you to get country details by country code with case-insensitive matching.</p>

<h2>Main Endpoint:</h2>

<div class="endpoint">

<h3>Get Country by Code (Case Insensitive)</h3>

<p><span class="method">GET</span> <span class="url">/countries/{code}</span></p>

<p><strong>Description:</strong> Returns a specific country based on country code. The country code matching is case insensitive.</p>

<div class="example">

<h4>Examples:</h4>

<ul>

<li><a href="/countries/in" target="\_blank">http://localhost:8083/countries/in</a> (lowercase)</li>

<li><a href="/countries/IN" target="\_blank">http://localhost:8083/countries/IN</a> (uppercase)</li>

<li><a href="/countries/Us" target="\_blank">http://localhost:8083/countries/Us</a> (mixed case)</li>

<li><a href="/countries/jp" target="\_blank">http://localhost:8083/countries/jp</a> (Japan)</li>

<li><a href="/countries/de" target="\_blank">http://localhost:8083/countries/de</a> (Germany)</li>

<li><a href="/countries/gb" target="\_blank">http://localhost:8083/countries/gb</a> (United Kingdom)</li>

<li><a href="/countries/fr" target="\_blank">http://localhost:8083/countries/fr</a> (France)</li>

<li><a href="/countries/ca" target="\_blank">http://localhost:8083/countries/ca</a> (Canada)</li>

<li><a href="/countries/au" target="\_blank">http://localhost:8083/countries/au</a> (Australia)</li>

</ul>

</div>

<p><strong>Sample Response for India (in/IN/In):</strong></p>

<div class="code">

<pre>{

"code": "IN",

"name": "India"

}</pre>

</div>

</div>

<h2>Additional Endpoints:</h2>

<div class="endpoint">

<h3>Get All Countries</h3>

<p><span class="method">GET</span> <span class="url">/countries</span></p>

<p>Returns all available countries</p>

<p><strong>URL:</strong> <a href="/countries" target="\_blank">http://localhost:8083/countries</a></p>

</div>

<div class="endpoint">

<h3>Health Check</h3>

<p><span class="method">GET</span> <span class="url">/health</span></p>

<p>Service health check endpoint</p>

<p><strong>URL:</strong> <a href="/health" target="\_blank">http://localhost:8083/health</a></p>

</div>

<h2>Implementation Details:</h2>

<ul>

<li><strong>Controller:</strong> com.cognizant.spring-learn.controller.CountryController</li>

<li><strong>Method Annotation:</strong> @GetMapping("/countries/{code}")</li>

<li><strong>Method Name:</strong> getCountry(String code)</li>

<li><strong>Service Method:</strong> com.cognizant.spring-learn.service.CountryService.getCountry(String code)</li>

<li><strong>Path Variable:</strong> Country code obtained using @PathVariable</li>

<li><strong>Data Source:</strong> Countries loaded from country.xml Spring configuration</li>

<li><strong>Matching:</strong> Case insensitive country code matching using Lambda expressions</li>

<li><strong>Response:</strong> Returns 200 OK with country details, or 404 Not Found if country doesn't exist</li>

</ul>

<h2>Available Country Codes:</h2>

<div class="code">

IN - India<br>

US - United States<br>

JP - Japan<br>

DE - Germany<br>

GB - United Kingdom<br>

FR - France<br>

CA - Canada<br>

AU - Australia

</div>

</div>

</body>

</html>

**//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"

xmlns:util="http://www.springframework.org/schema/util"

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

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

http://www.springframework.org/schema/util

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

<!-- Individual Country Beans -->

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

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

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

</bean>

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

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

<property name="name" value="United States"/>

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

<property name="name" value="United Kingdom"/>

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

<!-- Country List Configuration -->

<util:list id="countryList" value-type="com.cognizant.springlearn.model.Country">

<ref bean="india"/>

<ref bean="usa"/>

<ref bean="japan"/>

<ref bean="germany"/>

<ref bean="uk"/>

<ref bean="france"/>

<ref bean="canada"/>

<ref bean="australia"/>

</util:list>

</beans>

**//CountryCodeRestServiceApplicationTests.java**

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class CountryCodeRestServiceApplicationTests {

@Test

void contextLoads() {

}

}

**//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>3.2.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>country-code-rest-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>country-code-rest-service</name>

<description>REST Service to get country by country code using Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Jackson for JSON processing -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

<!-- Spring Boot Test Starter -->

<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>

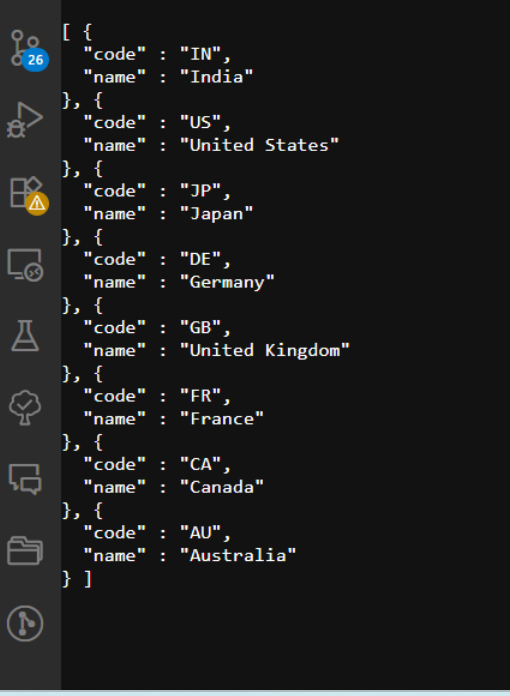
</plugin>

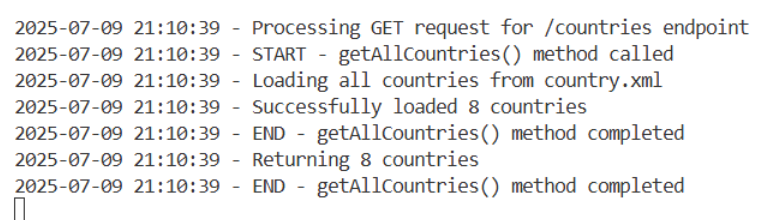
</plugins>

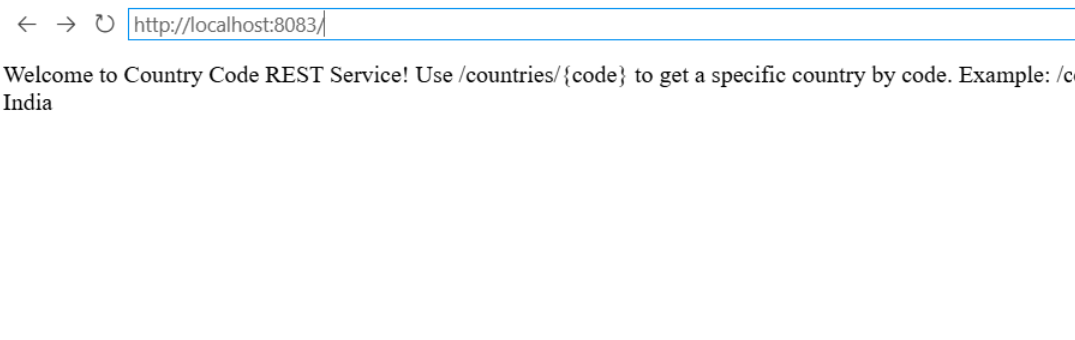
</build>

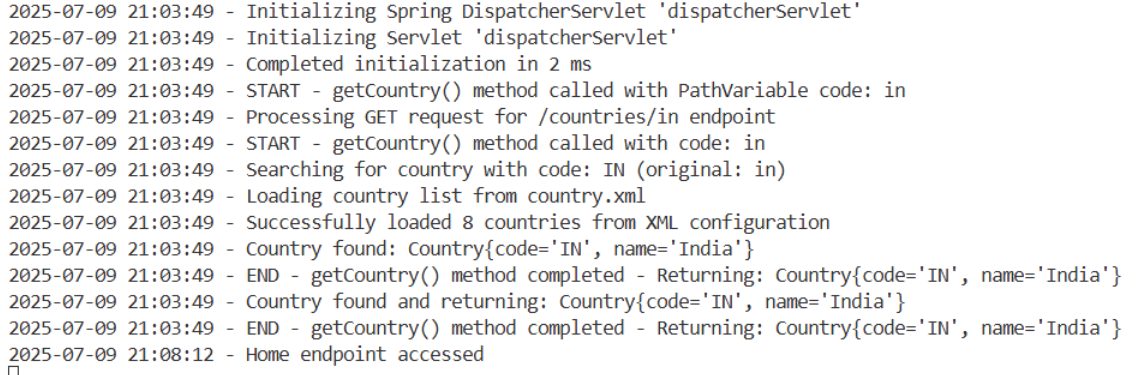
</project>

**OUTPUT:**









**JWT-handson**

**Create authentication service that returns JWT**

**//XmlConfigController.java**

package com.cognizant.springlearn.controller;

import java.text.SimpleDateFormat;

import java.util.Date;

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.GetMapping;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* REST Controller to demonstrate Spring XML Configuration via web endpoints

\*/

@RestController

public class XmlConfigController {

private static final Logger logger = LoggerFactory.getLogger(XmlConfigController.class);

/\*\*

\* Home endpoint showing XML configuration demo results

\*/

@GetMapping("/")

public String home() {

logger.info("Home endpoint accessed - demonstrating XML configuration");

StringBuilder response = new StringBuilder();

response.append("<html><head><title>Spring XML Configuration Demo</title></head><body>");

response.append("<h1>Spring Core - Load SimpleDateFormat from Spring Configuration XML</h1>");

response.append("<h2>Demo Results:</h2>");

try {

// Create ApplicationContext from XML configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Get the dateFormat bean from Spring container

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

// Parse the string '31/12/2018' to Date class

String dateString = "31/12/2018";

Date parsedDate = format.parse(dateString);

response.append("<p><strong>Original date string:</strong> ").append(dateString).append("</p>");

response.append("<p><strong>Parsed Date object:</strong> ").append(parsedDate).append("</p>");

response.append("<p><strong>Formatted back to string:</strong> ").append(format.format(parsedDate)).append("</p>");

response.append("<p><strong>Date pattern used:</strong> dd/MM/yyyy</p>");

response.append("<p style='color: green;'><strong>✅ Successfully loaded SimpleDateFormat from Spring XML Configuration!</strong></p>");

response.append("<h3>XML Configuration (date-format.xml):</h3>");

response.append("<pre style='background-color: #f4f4f4; padding: 10px;'>");

response.append("&lt;?xml version=\"1.0\" encoding=\"UTF-8\"?&gt;\n");

response.append("&lt;beans xmlns=\"http://www.springframework.org/schema/beans\"\n");

response.append(" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\n");

response.append(" xsi:schemaLocation=\"http://www.springframework.org/schema/beans\n");

response.append(" https://www.springframework.org/schema/beans/spring-beans.xsd\"&gt;\n\n");

response.append(" &lt;bean id=\"dateFormat\" class=\"java.text.SimpleDateFormat\"&gt;\n");

response.append(" &lt;constructor-arg value=\"dd/MM/yyyy\" /&gt;\n");

response.append(" &lt;/bean&gt;\n\n");

response.append("&lt;/beans&gt;");

response.append("</pre>");

} catch (Exception e) {

response.append("<p style='color: red;'><strong>❌ Error:</strong> ").append(e.getMessage()).append("</p>");

logger.error("Error in XML configuration demo: " + e.getMessage());

}

response.append("<h3>Other Endpoints:</h3>");

response.append("<ul>");

response.append("<li><a href='/health'>Health Check</a></li>");

response.append("<li><a href='/xml-demo'>XML Demo (JSON Response)</a></li>");

response.append("</ul>");

response.append("</body></html>");

return response.toString();

}

/\*\*

\* Health check endpoint

\*/

@GetMapping("/health")

public String health() {

logger.info("Health check endpoint accessed");

return "Spring XML Configuration Demo Application is UP and running on port 8081!";

}

/\*\*

\* XML demo endpoint with JSON response

\*/

@GetMapping("/xml-demo")

public String xmlDemo() {

logger.info("XML demo endpoint accessed");

try {

// Create ApplicationContext from XML configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Get the dateFormat bean from Spring container

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

// Parse the string '31/12/2018' to Date class

String dateString = "31/12/2018";

Date parsedDate = format.parse(dateString);

return "{\n" +

" \"demo\": \"Spring XML Configuration\",\n" +

" \"xmlFile\": \"date-format.xml\",\n" +

" \"beanId\": \"dateFormat\",\n" +

" \"beanClass\": \"java.text.SimpleDateFormat\",\n" +

" \"pattern\": \"dd/MM/yyyy\",\n" +

" \"originalString\": \"" + dateString + "\",\n" +

" \"parsedDate\": \"" + parsedDate + "\",\n" +

" \"formattedString\": \"" + format.format(parsedDate) + "\",\n" +

" \"status\": \"SUCCESS\"\n" +

"}";

} catch (Exception e) {

return "{\n" +

" \"demo\": \"Spring XML Configuration\",\n" +

" \"error\": \"" + e.getMessage() + "\",\n" +

" \"status\": \"ERROR\"\n" +

"}";

}

}

}

**//SpringXmlConfigDemoApplication.java**

package com.cognizant.springlearn;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

/\*\*

\* Main Spring Boot Application Class demonstrating XML Configuration

\*/

@SpringBootApplication

public class SpringXmlConfigDemoApplication {

private static final Logger logger = LoggerFactory.getLogger(SpringXmlConfigDemoApplication.class);

public static void main(String[] args) {

logger.info("=================================================================");

logger.info("Spring Core - Load SimpleDateFormat from Spring Configuration XML");

logger.info("=================================================================");

// Demonstrate Spring XML Configuration with SimpleDateFormat

displayDate();

// Start the Spring Boot application

SpringApplication.run(SpringXmlConfigDemoApplication.class, args);

logger.info("=================================================================");

logger.info("Spring XML Configuration Demo completed successfully!");

logger.info("Application is running on: http://localhost:8081");

logger.info("=================================================================");

}

/\*\*

\* Method to demonstrate Spring XML Configuration for SimpleDateFormat

\* This method loads the date-format.xml configuration and uses the dateFormat bean

\*/

public static void displayDate() {

try {

// Create ApplicationContext from XML configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Get the dateFormat bean from Spring container

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

// Parse the string '31/12/2018' to Date class

String dateString = "31/12/2018";

Date parsedDate = format.parse(dateString);

// Display the result using System.out.println

System.out.println("Original date string: " + dateString);

System.out.println("Parsed Date object: " + parsedDate);

System.out.println("Formatted back to string: " + format.format(parsedDate));

logger.info("Successfully loaded SimpleDateFormat from Spring XML Configuration");

logger.info("Date pattern used: dd/MM/yyyy");

logger.info("Parsed date: " + parsedDate);

} catch (Exception e) {

logger.error("Error in displayDate() method: " + e.getMessage());

e.printStackTrace();

}

}

}

**//index.html**

<!DOCTYPE html>

<html>

<head>

<title>Spring XML Configuration Demo</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

.container { max-width: 800px; margin: 0 auto; }

.success { color: green; }

.code { background-color: #f4f4f4; padding: 10px; border-left: 4px solid #007acc; }

.endpoint { background-color: #e8f4fd; padding: 10px; margin: 10px 0; }

</style>

</head>

<body>

<div class="container">

<h1>Spring Core - Load SimpleDateFormat from Spring Configuration XML</h1>

<h2>Demo Overview</h2>

<p>This application demonstrates how to load SimpleDateFormat from Spring XML Configuration to avoid creating it in multiple places.</p>

<div class="endpoint">

<h3>Available Endpoints:</h3>

<ul>

<li><a href="/">Home (Dynamic XML Demo)</a></li>

<li><a href="/health">Health Check</a></li>

<li><a href="/xml-demo">XML Demo (JSON Response)</a></li>

</ul>

</div>

<h3>XML Configuration (date-format.xml):</h3>

<div class="code">

<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;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"&gt;

&lt;bean id="dateFormat" class="java.text.SimpleDateFormat"&gt;

&lt;constructor-arg value="dd/MM/yyyy" /&gt;

&lt;/bean&gt;

&lt;/beans&gt;</pre>

</div>

<h3>Implementation Details:</h3>

<ul>

<li><strong>ApplicationContext:</strong> ClassPathXmlApplicationContext("date-format.xml")</li>

<li><strong>Bean Retrieval:</strong> context.getBean("dateFormat", SimpleDateFormat.class)</li>

<li><strong>Date Parsing:</strong> format.parse("31/12/2018")</li>

<li><strong>Pattern:</strong> dd/MM/yyyy</li>

</ul>

<div class="success">

<p><strong>✅ Benefits:</strong></p>

<ul>

<li>Single SimpleDateFormat instance defined in XML</li>

<li>Reusable across multiple classes</li>

<li>Centralized configuration management</li>

<li>Spring IoC container manages lifecycle</li>

</ul>

</div>

<p><em>Application running on port 8081</em></p>

</div>

</body>

</html>

**//date-format.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>

**//DateFormatDemo.java**

package com.cognizant.springlearn;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

/\*\*

\* Demonstration class showing how to load SimpleDateFormat from Spring XML Configuration

\*/

public class DateFormatDemo {

public static void main(String[] args) {

System.out.println("=================================================================");

System.out.println("Spring Core - Load SimpleDateFormat from Spring Configuration XML");

System.out.println("=================================================================");

displayDate();

System.out.println("=================================================================");

System.out.println("Demo completed successfully!");

System.out.println("=================================================================");

}

/\*\*

\* Method to demonstrate Spring XML Configuration for SimpleDateFormat

\* This method loads the date-format.xml configuration and uses the dateFormat bean

\*/

public static void displayDate() {

try {

// Create ApplicationContext from XML configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Get the dateFormat bean from Spring container

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

// Parse the string '31/12/2018' to Date class

String dateString = "31/12/2018";

Date parsedDate = format.parse(dateString);

// Display the result

System.out.println("Original date string: " + dateString);

System.out.println("Parsed Date object: " + parsedDate);

System.out.println("Formatted back to string: " + format.format(parsedDate));

System.out.println("Successfully loaded SimpleDateFormat from Spring XML Configuration");

System.out.println("Date pattern used: dd/MM/yyyy");

System.out.println("Parsed date: " + parsedDate);

} catch (Exception e) {

System.err.println("Error in displayDate() method: " + e.getMessage());

e.printStackTrace();

}

}

}

**//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>3.2.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-xml-config-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-xml-config-demo</name>

<description>Spring Core - Load SimpleDateFormat from Spring Configuration XML</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Spring Boot Test Starter -->

<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>

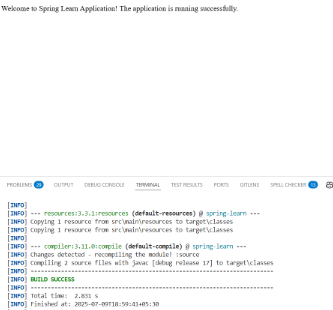
</plugin>

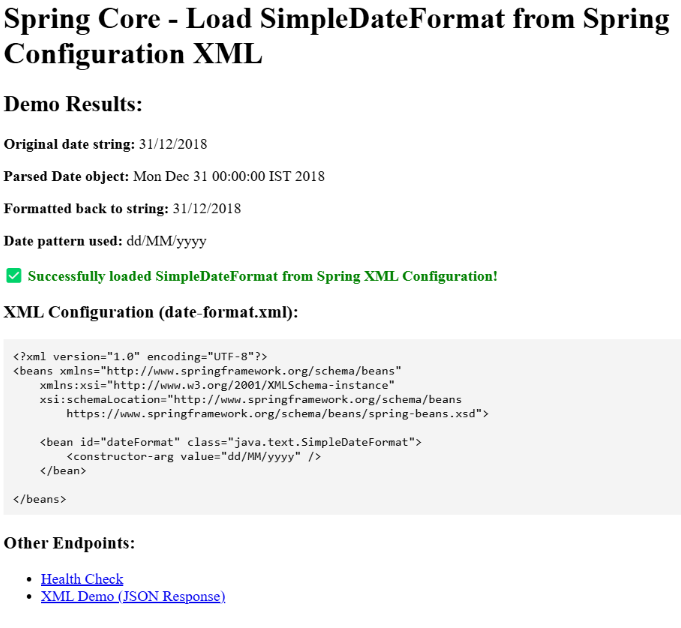
</plugins>

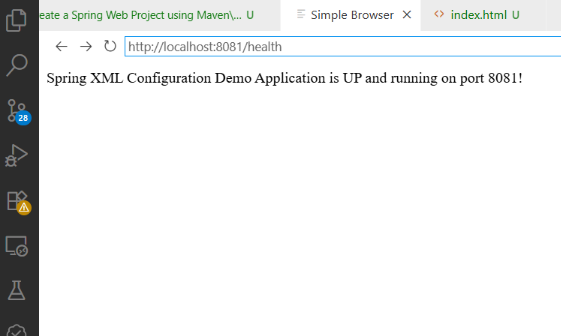
</build>

</project>

**OUTPUT:**

****

****

****

****