# REST – Country Web Service in Spring Boot

## Introduction

This document outlines the implementation of a simple RESTful web service in a Spring Boot application that returns details of a country — specifically India. The service loads a Country bean defined in an XML configuration and responds with the country’s information in JSON format. This serves as a foundational exercise in understanding Spring MVC REST controllers, bean configuration, and JSON serialization.

## Objective

Create a REST endpoint:

* **URL**: /country
* **Controller**: com.cognizant.spring-learn.controller.CountryController
* **Bean Source**: XML file (country.xml)
* **HTTP Method**: GET
* **Expected Response**:

## Project Setup

### Dependencies

* Spring Boot Starter Web
* Spring Context (for XML bean loading)
* Jackson (auto-included with Spring Boot for JSON conversion)

## Component Implementation

### Model Class

// Country.java  
package com.cognizant.springlearn.model;  
  
public class Country {  
 private String code;  
 private String name;  
  
 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; }  
}

### XML Bean Configuration

<!-- country.xml -->  
<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="in" class="com.cognizant.springlearn.model.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
</beans>

### Spring Boot Main Class

// SpringLearnApplication.java  
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) {  
 SpringApplication.run(SpringLearnApplication.class, args);  
 }  
}

### Controller Class

// 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.\*;  
  
@RestController  
public class CountryController {  
  
 @RequestMapping("/country")  
 public Country getCountryIndia() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = (Country) context.getBean("in");  
 return country;  
 }  
}

### Configuration File

# application.properties  
server.port=8083

## Behind the Scenes

### What Happens in the Controller Method?

* The @RequestMapping("/country") maps incoming GET requests to the method getCountryIndia().
* Inside this method:
  + ClassPathXmlApplicationContext loads the country.xml.
  + The Country bean with ID in is retrieved and returned.
* Spring’s @RestController automatically converts the returned Java object to JSON.

### How is the Bean Converted to JSON?

* Spring Boot uses **Jackson** for object-to-JSON conversion.
* Jackson introspects the Country object (via getters/setters) and serializes it to a valid JSON string.
* No need for manual conversion — Spring does it automatically behind the scenes.

## Testing the Service

### Request

GET http://localhost:8083/country

### Response

{  
 "code": "IN",  
 "name": "India"  
}

## Verifying in Tools

### Developer Tools (Browser Network Tab)

* **Request URL**: http://localhost:8083/country
* **Request Method**: GET
* **Status Code**: 200 OK
* **Content-Type**: application/json
* **Response Payload**:

### Postman Headers Tab

When testing in Postman:

| Key | Value |
| --- | --- |
| Content-Type | application/json |
| Content-Length | 31 |
| Date | [Current Timestamp] |
| Server | Apache Tomcat/Embedded |

## Conclusion

This project demonstrates a simple yet effective use of Spring Boot to build a RESTful service that loads a Java bean from an XML configuration and returns it as a JSON response. The use of annotations like @RestController and @RequestMapping, combined with Spring’s integration with Jackson, makes it seamless to build and test REST APIs.