# REST – Get Country Based on Country Code

## Introduction

This document describes how to implement a Spring Boot RESTful web service that retrieves a specific country by its code from a list of countries defined in an XML configuration. The search should be case-insensitive, and the result should be returned in JSON format.

## Objective

Create a REST endpoint:

* **URL**: /countries/{code}
* **HTTP Method**: GET
* **Path Variable**: code (country code, case-insensitive)
* **Controller**: com.cognizant.spring-learn.controller.CountryController
* **Service**: com.cognizant.spring-learn.service.CountryService.getCountry(String code)

### Sample Request:

GET http://localhost:8083/countries/in

### Sample Response:

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

## Implementation Details

### XML Configuration File (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="countryList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <bean class="com.cognizant.springlearn.model.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
 <bean class="com.cognizant.springlearn.model.Country">  
 <property name="code" value="US"/>  
 <property name="name" value="United States"/>  
 </bean>  
 <!-- Add more countries as needed -->  
 </list>  
 </constructor-arg>  
 </bean>  
</beans>

### 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; }  
}

### Service Class: CountryService.java

package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.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) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countries = (List<Country>) context.getBean("countryList");  
  
 return countries.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
 }  
}

### Controller Class: CountryController.java

package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.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) {  
 return countryService.getCountry(code);  
 }  
}

### Configuration: application.properties

server.port=8083

## Behind the Scenes

* The @GetMapping("/countries/{code}") maps HTTP GET requests to the method getCountry() in the controller.
* The @PathVariable annotation extracts the country code from the URL.
* The service method retrieves the list of countries from country.xml.
* It performs a case-insensitive match using equalsIgnoreCase() in a stream filter.
* On finding the match, it returns the corresponding Country object, which Spring converts into JSON.

## Sample Test

### Request

GET http://localhost:8083/countries/in

### Response

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

## Conclusion

This RESTful web service demonstrates how to extract a path variable from a URL and use it to perform a case-insensitive search within a list of pre-configured beans. It illustrates key Spring concepts such as @PathVariable, @GetMapping, service layer abstraction, and XML-based bean configuration.

By using lambda expressions and Spring Boot features, the service remains concise, clean, and easy to extend in future applications.