REST Service to Get Country by Code

1. CountryService Implementation

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import java.util.ArrayList;

import java.util.List;

@Service

public class CountryService {

private List<Country> countries = new ArrayList<>();

@PostConstruct

public void init() {

// Initialize with sample data (could be loaded from XML)

countries.add(new Country("IN", "India"));

countries.add(new Country("US", "United States"));

countries.add(new Country("DE", "Germany"));

countries.add(new Country("JP", "Japan"));

}

public Country getCountry(String code) {

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElseThrow(() -> new CountryNotFoundException("Country not found"));

}

}

2. Custom Exception Class

package com.cognizant.springlearn.exception;

public class CountryNotFoundException extends RuntimeException {

public CountryNotFoundException(String message) {

super(message);

}

}

3. Enhanced CountryController

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.GetMapping;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

4. Global Exception Handler

package com.cognizant.springlearn.exception;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(CountryNotFoundException.class)

public ResponseEntity<String> handleCountryNotFound(CountryNotFoundException ex) {

return ResponseEntity

.status(HttpStatus.NOT\_FOUND)

.body(ex.getMessage());

}

}

5. Testing the Service

Successful Request:

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

Response:

{

"code": "IN",

"name": "India"

}

Case Insensitive Test:

URL: GET http://localhost:8083/countries/In

Response: Same as above (works with any case combination)

Country Not Found:

URL: GET http://localhost:8083/countries/XX

Response:

404 Not Found

Country not found

Key Implementation Details:

Case Insensitive Matching:

Uses equalsIgnoreCase() for code comparison

Works with any case combination (IN, in, In, iN)

Service Layer:

Contains business logic for country retrieval

Uses Java Streams with lambda for clean code

Throws custom exception when country not found

Error Handling:

Custom CountryNotFoundException

Global exception handler returns proper 404 response

REST Best Practices:

Proper use of HTTP verbs (GET)

Correct status codes (200 for success, 404 for not found)

Clean URL structure (/countries/{code})

Alternative: Loading from XML

If you need to load countries from XML instead of hardcoding:

Create countries.xml:

<countries>

<country>

<code>IN</code>

<name>India</name>

</country>

<country>

<code>US</code>

<name>United States</name>

</country>

</countries>

Modify CountryService to load from XML:

@PostConstruct

public void init() throws JAXBException {

JAXBContext context = JAXBContext.newInstance(CountryList.class);

Unmarshaller unmarshaller = context.createUnmarshaller();

CountryList countryList = (CountryList) unmarshaller.unmarshal(

new ClassPathResource("countries.xml").getInputStream());

this.countries = countryList.getCountries();

}