HANDSON

**HELLO WORLD RESTFUL WEB SERVICE**

# **WRITE A REST SERVICE IN THE SPRING LEARN APPLICATION CREATED EARLIER, THAT RETURNS THE TEXT "HELLO WORLD!!" USING SPRING WEB FRAMEWORK. REFER DETAILS BELOW:**

|  |
| --- |
| Method: GET |
| URL: /hello |
| Controller: com.cognizant.spring-learn.controller.HelloController |
| Method Signature: public String sayHello() |
| Method Implementation: return hard coded string "Hello World!!" |
| Sample Request: http://localhost:8083/hello |
| Sample Response: Hello World!! |
| IMPORTANT NOTE: Don't forget to include start and end log in the sayHello() method. |
| Try the URL http://localhost:8083/hello in both chrome browser and postman. |

SME to explain the following aspects:

* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

# **CODE:**

## **POM.XML**

<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.example</groupId>

    <artifactId>spring-rest-demo</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>spring-rest-demo</name>

    <properties>

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

        <spring.boot.version>3.2.4</spring.boot.version>

    </properties>

    <dependencies>

*<!-- ✅ Spring Boot Web Starter -->*

        <dependency>

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

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

            <version>${spring.boot.version}</version>

        </dependency>

*<!-- ✅ Spring Boot Starter Test (optional) -->*

        <dependency>

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

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

            <version>${spring.boot.version}</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

*<!-- ✅ Spring Boot Maven Plugin -->*

            <plugin>

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

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

                <version>${spring.boot.version}</version>

            </plugin>

        </plugins>

    </build>

</project>

## **SPRINGRESTDEMOAPLLICATION.JAVA**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringRestDemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringRestDemoApplication.class, args);

    }

}

## **HELLOCONTROLLER.JAVA**

package com.example.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*;

@RestController

public class HelloController {

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

    @GetMapping("/hello")

    public String sayHello() {

        LOGGER.info("START sayHello()");

        String message = "Hello World!!";

        LOGGER.info("END sayHello()");

        return message;

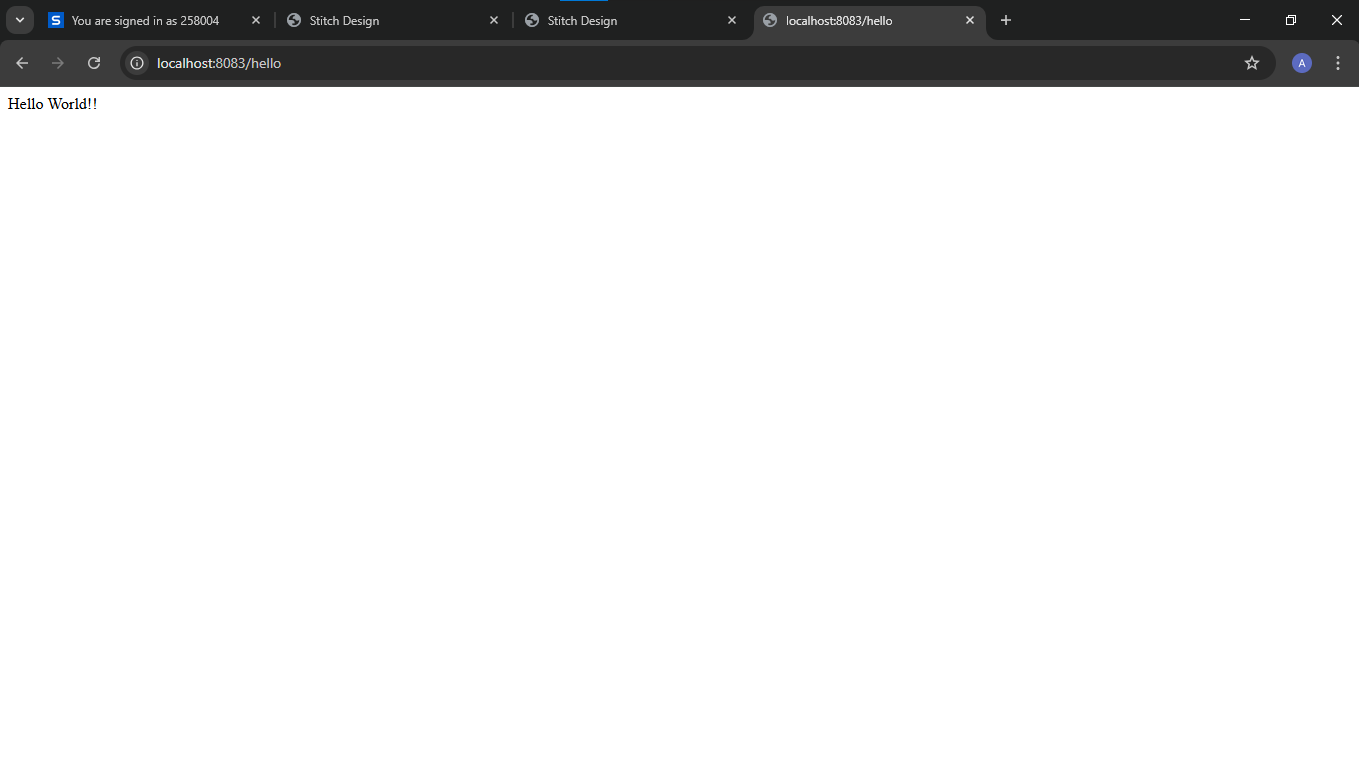
    }

}

# **APPLICATION.PROPERTIES**

server.port=8083

# **OUTPUT:**



**REST - Country Web Service**

# **WRITE A REST SERVICE THAT RETURNS INDIA COUNTRY DETAILS IN THE EARLIER CREATED SPRING LEARN APPLICATION.**

|  |
| --- |
| URL: /country |
| Controller: com.cognizant.spring-learn.controller.CountryController |
| Method Annotation: @RequestMapping |
| Method Name: getCountryIndia() |
| Method Implementation: Load India bean from spring xml configuration and return |
| Sample Request: <http://localhost:8083/country> |
| Sample Response: |
| { |
| "code": "IN", |
| "name": "India" |
| } |

SME to explain the following aspects:

* What happens in the controller method?
* How the bean is converted into JSON reponse?
* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

# **CODE:**

## **COUNTRY.JAVA**

package com.example.model;

public class Country {

    private String code;

    private String 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;

    }

}

## **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*="in" *class*="com.example.model.Country">

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

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

    </bean>

</beans>

## **CONTROLLERCOUNTRY.JAVA**

package com.example.controller;

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;

import com.example.model.Country;

@RestController

public class CountryController {

    @GetMapping("/country")

    public Country getCountryIndia() {

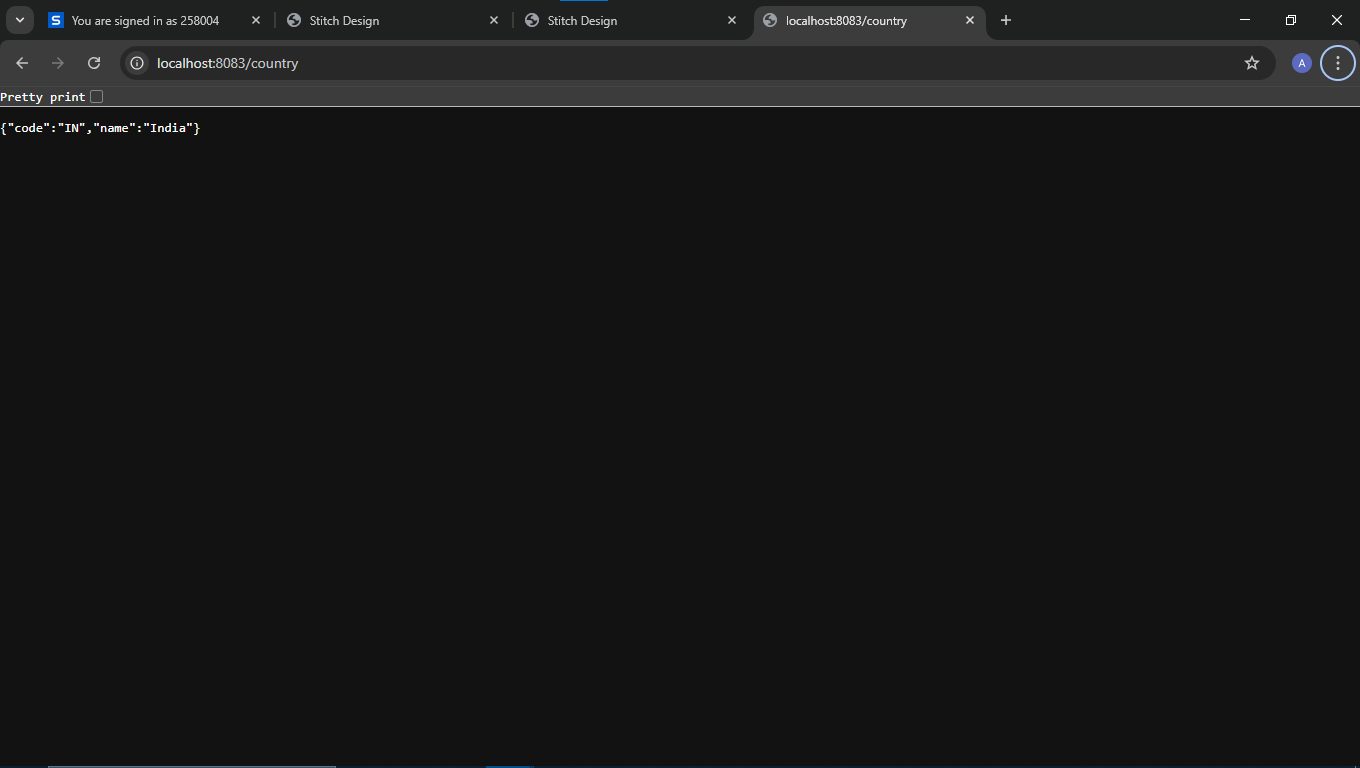
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

        return context.getBean("in", Country.class);

    }

}

# **OUTPUT:**



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

# **WRITE A 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 Implemetation: Invoke countryService.getCountry(code)

Service Method: com.cognizant.spring-learn.service.CountryService.getCountry(String 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 can also be used instead of iterating the country list

|  |
| --- |
| Sample Request: http://localhost:8083/country/in |
| Sample Response: |
| {    "code": "IN",  "name": "India"  } |

# **CODE:**

# **COUNTRYSERVICE.JAVA**

package com.example.service;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.example.exception.CountryNotFoundException;

import com.example.model.Country;

@Service

public class CountryService {  *// ✅ class wrapper added*

    public Country getCountry(String code) throws CountryNotFoundException {

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

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

        return countries.stream()

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

                .findFirst()

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

    }

    public List<Country> getAllCountries() {

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

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

    }

}

# **COUNTRYCONTROLLER.JAVA**

package com.example.controller;

import com.example.model.Country;

import com.example.service.CountryService;

import com.example.exception.CountryNotFoundException;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

public class CountryController {

    @Autowired

    private CountryService countryService;

    @GetMapping("/country")

    public Country getCountryIndia() {

        return new Country("IN", "India");  *// hardcoded for /country*

    }

    @GetMapping("/countries")

    public List<Country> getAllCountries() {

        return countryService.getAllCountries();  *// if you've added this*

    }

    @GetMapping("/countries/{code}")

    public Country getCountryByCode(@PathVariable String code) throws CountryNotFoundException {

        return countryService.getCountry(code);

    }

}

# **COUNTRYNOTFOUNDEXCEPTION.JAVA**

package com.example.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(value = HttpStatus.NOT\_FOUND, reason = "Country not found")

public class CountryNotFoundException extends Exception {

    public CountryNotFoundException(String message) {

        super(message);

    }

}

# **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*="in" *class*="com.example.model.Country">

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

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

    </bean>

    <bean *id*="countryList" *class*="java.util.ArrayList">

        <constructor-arg>

            <list>

                <ref *bean*="in"/>

                <bean *class*="com.example.model.Country">

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

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

                </bean>

                <bean *class*="com.example.model.Country">

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

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

                </bean>

                <bean *class*="com.example.model.Country">

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

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

                </bean>

            </list>

        </constructor-arg>

    </bean>

</beans>

# **COUNTRY.JAVA**

package com.example.model;

public class Country {

    private String code;

    private String name;

    public Country() {}  *// default constructor*

    public Country(String code, String name) {  *// ✅ fix*

        this.code = code;

        this.name = 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;

    }

}

# **OUTPUT:**

