**WEEK-4**

**SPRING REST USING SPRING BOOT 3**

* **Hello World RESTful Web Service**
* HelloController.java:

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController

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

@GetMapping("/hello")

public String sayHello() {

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

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

return "Hello World!!";

}

}

* Main application (created automatically)
* 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);

}

}

* Application.properties

server.port=8083

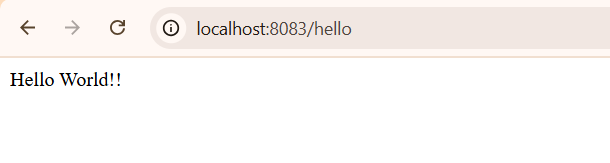
spring.application.name=spring-learn

logging.level.org.springframework=info

logging.level.com.cognizant.springlearn=debug

logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p|%-25.25logger{25}|%25M|%m%n

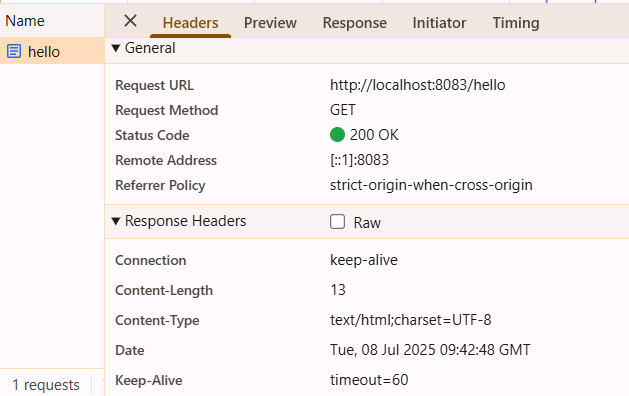
1. SpringLearnApplication -> Run As -> Java Application
2. Testing <http://localhost:8083/hello>



1. Verifying HTTP Headers

* Chrome

Press F12 or right click anywhere on page, choose ‘Inspect’ and click ‘Network’ tab.Network tab must be active. You will see a row named hello



**Explantion:**

* I created a Spring Boot project called spring-learn with basic web dependencies and set the server port to 8083 in application.properties to run a custom REST endpoint.
* I built a HelloController class using @RestController and added a @GetMapping("/hello") method that returns the simple string “Hello World!!”.
* I also configured logging using SLF4J to log the start and end of the sayHello() method for better traceability in the console.
* After running the project, I tested the endpoint at http://localhost:8083/hello in a browser and saw the expected output.
* I verified HTTP headers using Chrome DevTools → Network tab, which confirmed the successful RESTful response including method type and status code.
* **REST - Country Web Service**

1. spring-learn.xml

* Src/main/resources -> New -> File -> springlearn.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.cognizant.springlearn.model.Country">

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

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

</bean>

</beans>

1. Country model class

* Src/main/java -> package (com.cognizant.springlearn.model) -> class (Country)
* Country.java

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

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;

}

}

1. CountryController class

* Src/main/java -> package (com.cognizant.springlearn.controller) -> class (CountryController)
* CountryController.java

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

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;

}

}

1. Src/main/java -> com.cognizant.springlearn -> 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) {

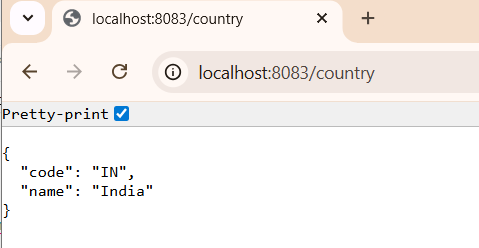
SpringApplication.run(SpringLearnApplication.class, args);

}

}

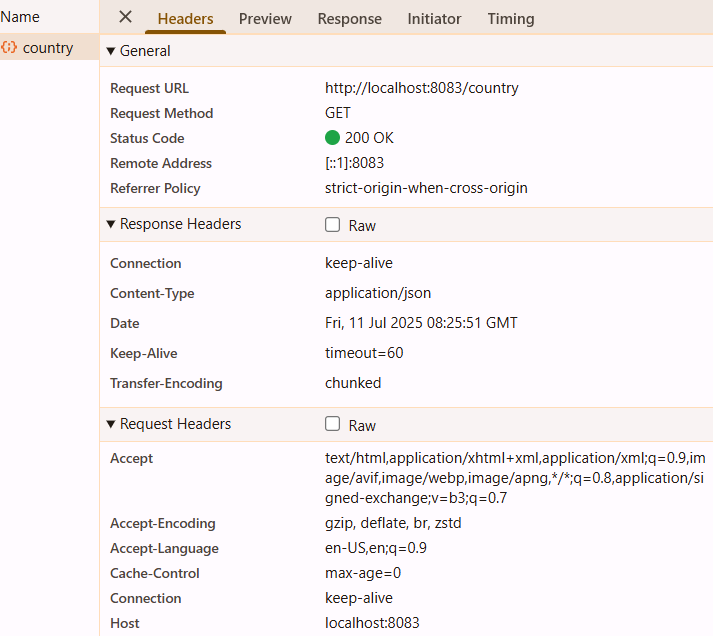
1. SpringLearnApplication -> Run As -> Java Application
2. Test in browser

* <http://localhost:8083/country>
* Browser:



1. HTTP Headers

* Chrome -> Right click -> Inspect -> Network -> Country



**Explanation:**

* I created an XML file springlearn.xml inside the resources folder to define a Spring bean for a Country object with values like code IN and name India.
* I wrote a Country class in the model package with code and name as fields, including standard constructors and getter/setter methods.
* In SpringLearnApplication, I loaded the Spring XML configuration using ClassPathXmlApplicationContext and fetched the bean from the XML.
* I logged the loaded Country object using SLF4J to confirm that the data was retrieved correctly from the XML file.
* Running the application showed the log statements and confirmed that XML-based Spring bean configuration worked correctly in a Spring Boot project.
* **REST - Get country based on country code**

1. Country.xml

* Src/main/resources -> New -> File
* 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="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>

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

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

1. CountryService.java

* Src/main/java -> package (com.cognizant.springlearn.service) -> class (CountryService)
* 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> countryList = (List<Country>) context.getBean("countryList");

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

1. 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);

}

}

1. Country.java

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

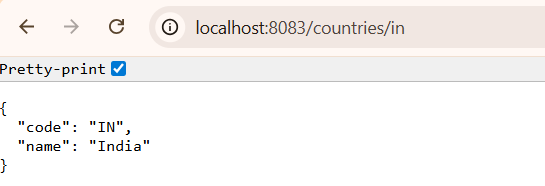
public void setName(String name) {

this.name = name;

}

}

1. SpringLearnApplication -> Run As -> Java Application <http://localhost:8083/countries/in>



**Explanation:**

* I defined a list of Country beans inside country.xml with data for India, USA, and China, and mapped them into a List using the countryList bean.
* I created a CountryService class that loads this bean and filters the country list to find the matching country based on the provided country code.
* I then created a REST controller CountryController and defined a dynamic GET endpoint using @GetMapping("/countries/{code}") to return the selected country.
* I tested this by running the application and opening a browser to http://localhost:8083/countries/in, and the corresponding country object was returned in JSON format.
* This exercise helped me understand how to integrate traditional Spring XML bean configuration with RESTful web services and serve dynamic data through path variables.