# WEEK 4 – SPRING REST HANDSON

**1.Create a Spring Web Project using Maven**

SpringLearnApplication Code:

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) {

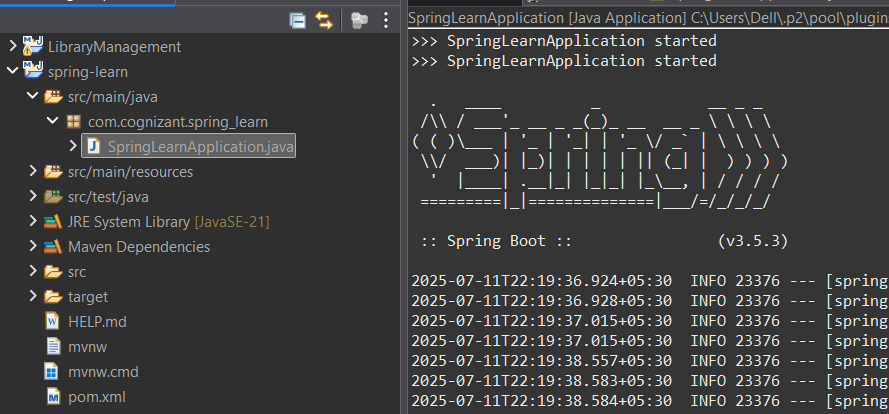
System.***out***.println(">>> SpringLearnApplication started");

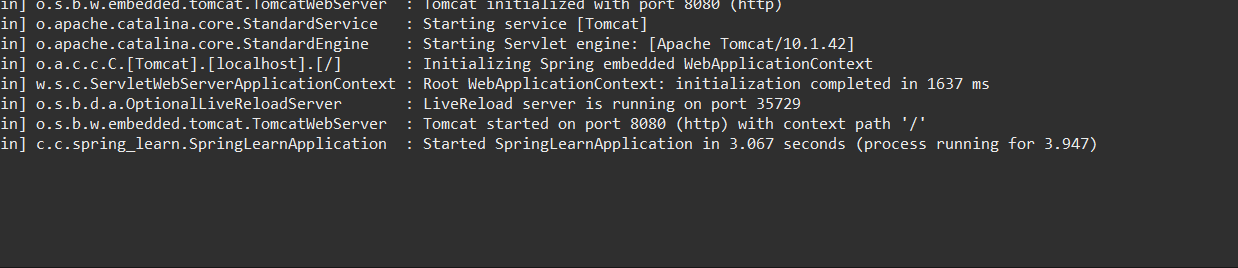
SpringApplication.*run*(SpringLearnApplication.class, args);

}

}

**OUTPUT:**





**Spring Core – Load Country from Spring Configuration XML**

**SpringLearnApplication Code:**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication implements CommandLineRunner {

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

public static void main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.class, args);

}

*@Override*

public void run(String... args) {

***LOGGER***.debug("START");

*displayCountry*();

***LOGGER***.debug("END");

}

public static void displayCountry() {

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

Country country = context.getBean("country", Country.class);

***LOGGER***.debug("Country : {}", country.toString());

}

}

**Country.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

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

private String code;

private String name;

public Country() {

***LOGGER***.debug("Inside Country Constructor.");

}

public String getCode() {

***LOGGER***.debug("getCode() called");

return code;

}

public void setCode(String code) {

***LOGGER***.debug("setCode() called with: {}", code);

this.code = code;

}

public String getName() {

***LOGGER***.debug("getName() called");

return name;

}

public void setName(String name) {

***LOGGER***.debug("setName() called with: {}", name);

this.name = name;

}

*@Override*

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**Application.properties**

spring.application.name=spring-learn

logging.level.root=DEBUG

**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="country" class="com.cognizant.spring\_learn.Country">

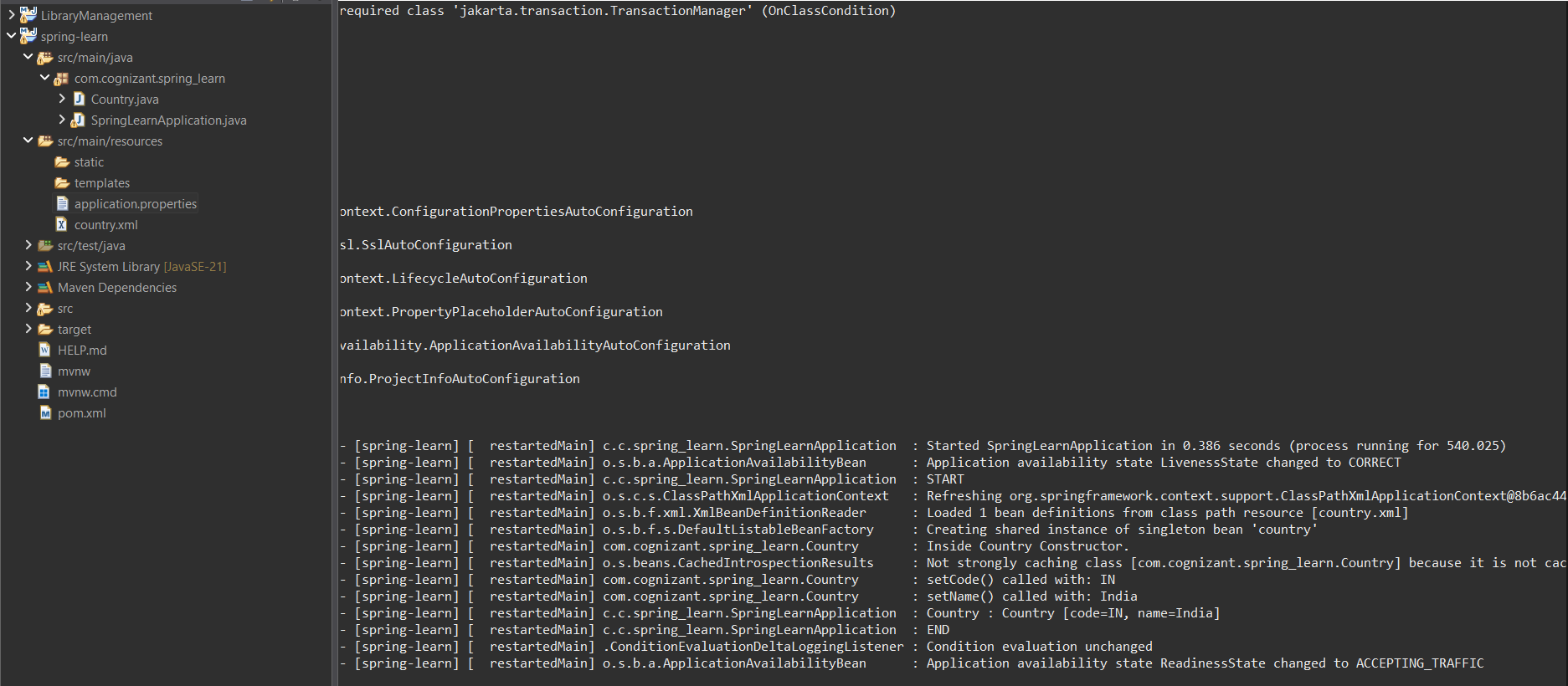
<property name="code" value="IN" />

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

</bean>

</beans>

**OUTPUT:**



**Hello World RESTful Web Service**

STATEMENT:

**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!!

CODE:

**HelloController.java:**

package com.cognizant.spring\_learn.controller;

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.debug("START: sayHello()");

String response = "Hello World!!";

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

return response;

}

}

**SME Explanation**

**Developer Tools (Chrome)**

1. Open Chrome → Visit http://localhost:8083/hello
2. Press F12 → Go to **Network** tab
3. Reload the page
4. Click on the /hello request
5. View these **Response Headers**:
   * Content-Type: text/plain;charset=UTF-8
   * Content-Length: 14
   * Date: ...
   * Connection: keep-alive

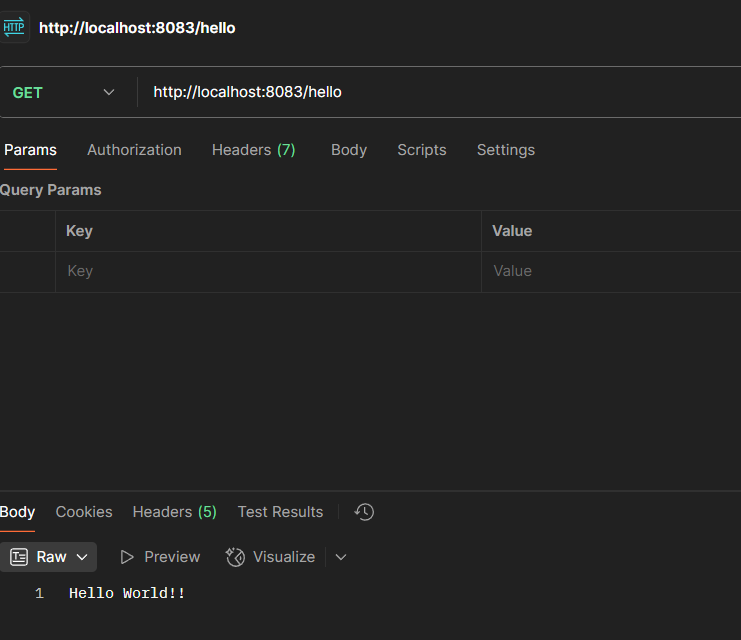
Also view **Request Headers**:

* GET /hello HTTP/1.1
* Host: localhost:8083
* User-Agent: Mozilla/...
* Accept: text/html,...

**Postman → Headers Tab**

1. After sending GET request to /hello, click on the **Headers** tab.
2. You’ll see similar headers:
   * **Response Headers**:
     + Content-Type: text/plain;charset=UTF-8
     + Date, Connection, Content-Length, etc.
   * **Request Headers**:
     + User-Agent, Accept, Host

**OUTPUT:**





**REST - Country Web Service**

**STATEMENTS:**

**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:**

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController {

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

@RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.debug("START: getCountryIndia()");

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

Country country = context.getBean("country", Country.class);

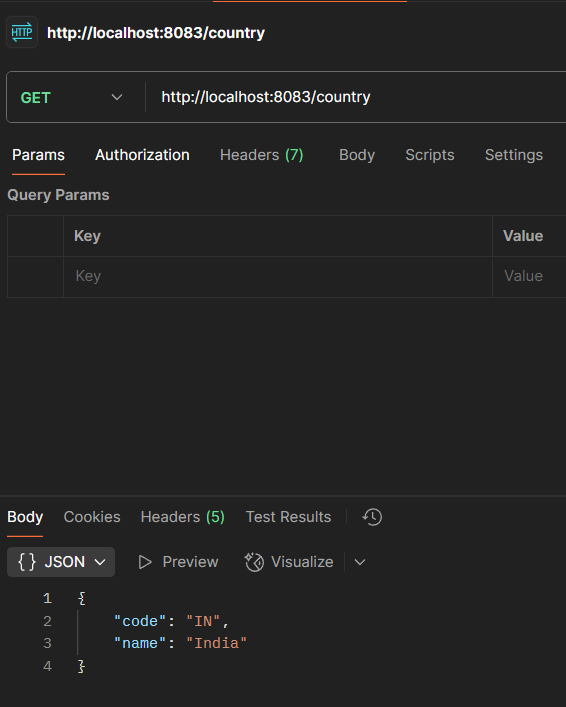
LOGGER.debug("END: getCountryIndia()");

return country;

}

}

**OUTPUT:**



SME:

**What happens in the controller method?**

java

CopyEdit

@RequestMapping("/country")

public Country getCountryIndia() {

...

}

* This method handles HTTP **GET** requests to /country.
* It loads the Spring bean (Country) from country.xml using ApplicationContext.
* Returns a **Java object**, not a JSON string — Spring takes care of conversion.

**How is the bean converted to JSON?**

1. Spring Boot includes **Jackson** (a JSON processor) in its dependencies.
2. When a Java object is returned from a @RestController method, Spring uses Jackson to automatically:
   * Inspect the object's fields via getters
   * Convert it to a JSON string
   * Set Content-Type: application/json in the response headers

**In Chrome Developer Tools**

1. Visit http://localhost:8083/country
2. Press F12 → Go to **Network** tab → Reload
3. Click the /country request

**Request Headers:**

* GET /country
* Host: localhost:8083
* Accept: application/json

**In Postman → Headers Tab**

1. Send a **GET** request to http://localhost:8083/country
2. Click the **Headers** tab

