**Week-4 Handson**

**Spring Rest using Springboot3**

***Exercise1:***

**Creation of a Spring Web Project using Maven**

Initialize Spring Boot Project

* Opened Spring Initializer
* Updated project details:
  + **Group**: com.cognizant
  + **Artifact**: spring-learn
* Selected Dependencies:
  + Spring Web
  + Spring Boot DevTools
* Downloaded the generated zip file.

**Setup in Eclipse Workspace**

* Extracted the downloaded zip to the root Eclipse workspace folder.
* Opened Command Prompt and navigated to the extracted folder.
* Executed Maven build using the following command (with proxy)

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

**Import Project into Eclipse**

**Add Logs in main() Method  
SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

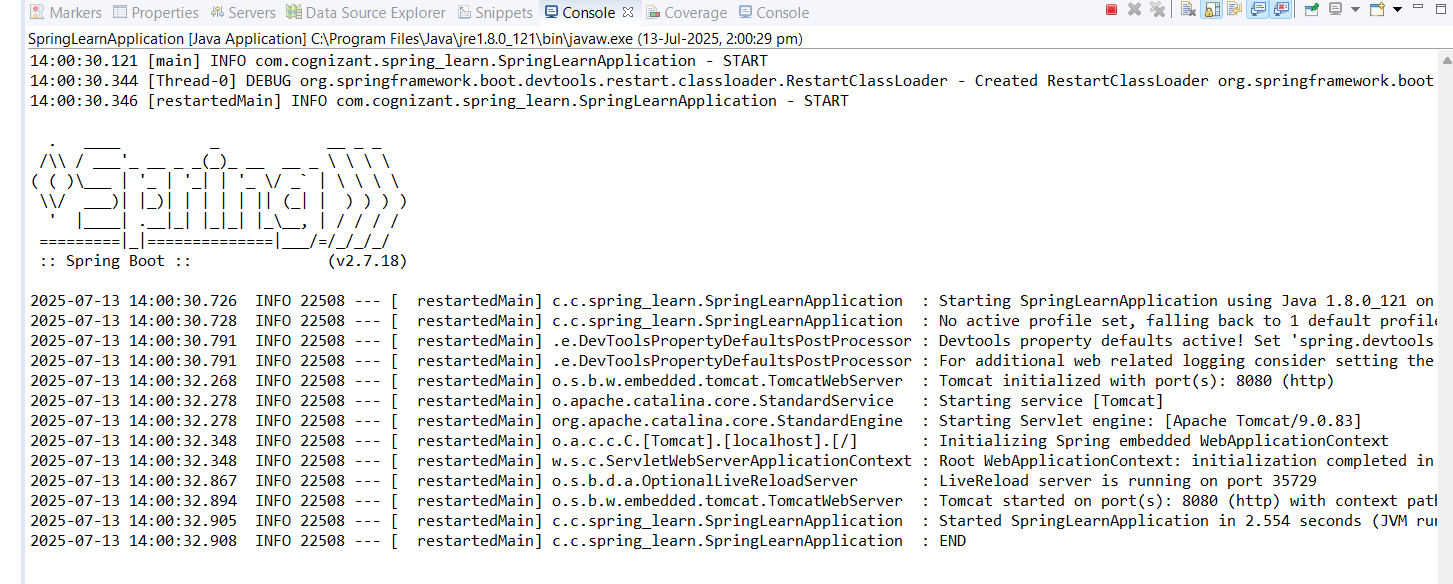
private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

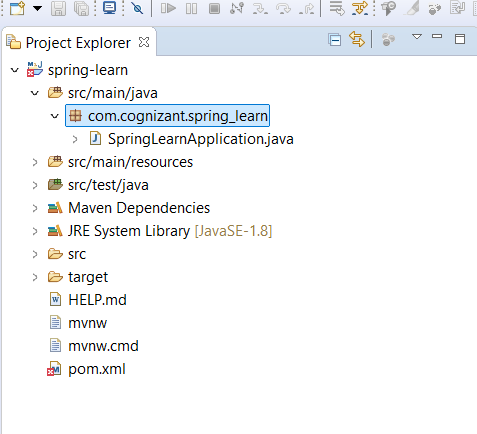
public static void main(String[] args) {

LOGGER.info("START");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");}}

**Run Application:**

**Exercise2:**

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

Inorder to create this, we need some of the files to be created in the project folder.

**Create:**

**Country.java:**

package com.cognizant.springlearn;

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("Inside getCode()");

        return code;

    }

    public void setCode(String code) {

        LOGGER.debug("Inside setCode()");

        this.code = code;

    }

    public String getName() {

        LOGGER.debug("Inside getName()");

        return name;

    }

    public void setName(String name) {

        LOGGER.debug("Inside setName()");

        this.name = name;

    }

    @Override

    public String toString() {

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

    }

}

**SpringLearnApplication.java:**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

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

    public static void main(String[] args) {

        LOGGER.debug("START");

        displayCountry();

        LOGGER.debug("END");

    }

    public static void displayCountry() {

        try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

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

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

    }

    }

}

**In main/resources/**

**country.xml**

**We have used “IN” for India here:**

<?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="country" class="com.cognizant.springlearn.Country">

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

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

    </bean>

</beans>

**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/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.cognizant.springlearn</groupId>

  <artifactId>spring-core-xml</artifactId>

  <packaging>jar</packaging>

  <version>1.0-SNAPSHOT</version>

  <name>spring-core-xml</name>

  <url>http://maven.apache.org</url>

  <dependencies>

    <!-- Spring Core and Context -->

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-context</artifactId>

        <version>5.3.36</version>

    </dependency>

    <!-- SLF4J API for logging -->

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-api</artifactId>

        <version>1.7.36</version>

    </dependency>

    <!-- Logback for SLF4J implementation -->

    <dependency>

        <groupId>ch.qos.logback</groupId>

        <artifactId>logback-classic</artifactId>

        <version>1.2.11</version>

    </dependency>

    <!-- JUnit (Optional) -->

    <dependency>

        <groupId>junit</groupId>

        <artifactId>junit</artifactId>

        <version>4.13.2</version>

        <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <plugin>

        <groupId>org.codehaus.mojo</groupId>

        <artifactId>exec-maven-plugin</artifactId>

        <version>3.1.0</version>

        <configuration>

          <mainClass>com.cognizant.springlearn.SpringLearnApplication</mainClass>

        </configuration>

      </plugin>

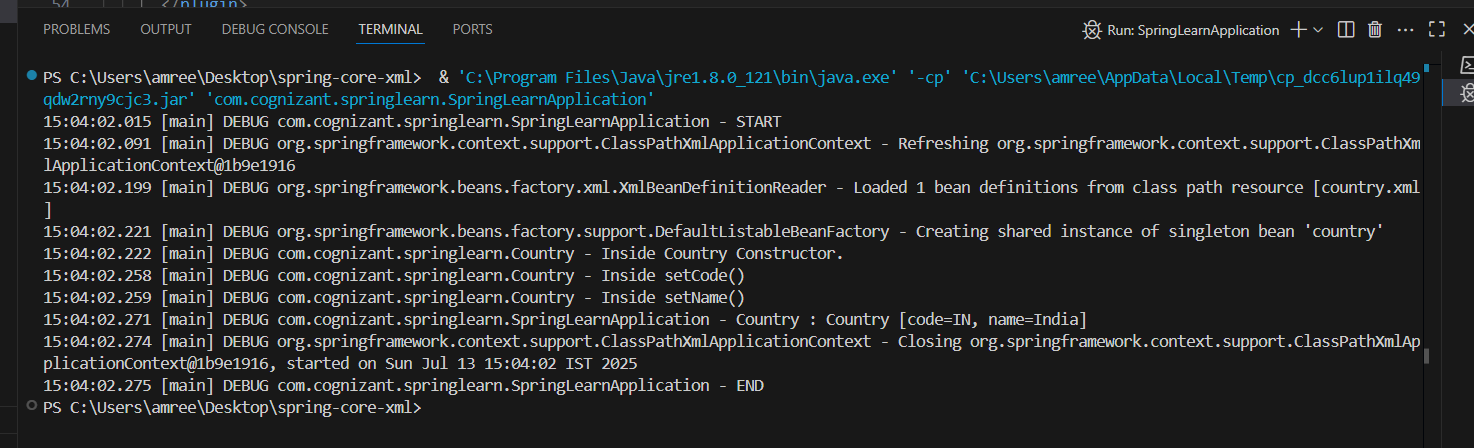
    </plugins>

  </build>

</project>

When we run this java file,we get:

Output:



**Exercise3:**

**Hello World RESTful Web Service**To write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.

**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!!

**Step 1: Add Spring Web Dependency**

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/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.cognizant.springlearn</groupId>

  <artifactId>spring-core-xml</artifactId>

  <packaging>jar</packaging>

  <version>1.0-SNAPSHOT</version>

  <name>spring-core-xml</name>

  <url>http://maven.apache.org</url>

  <dependencies>

    <!-- Spring Core and Context -->

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-context</artifactId>

        <version>5.3.36</version>

    </dependency>

    <!-- SLF4J API for logging -->

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-api</artifactId>

        <version>1.7.36</version>

    </dependency>

    <!-- Logback for SLF4J implementation -->

    <dependency>

        <groupId>ch.qos.logback</groupId>

        <artifactId>logback-classic</artifactId>

        <version>1.2.11</version>

    </dependency>

    <!-- JUnit (Optional) -->

    <dependency>

        <groupId>junit</groupId>

        <artifactId>junit</artifactId>

        <version>4.13.2</version>

        <scope>test</scope>

    </dependency>

    <dependency>

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

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

    <version>2.7.18</version>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <plugin>

        <groupId>org.codehaus.mojo</groupId>

        <artifactId>exec-maven-plugin</artifactId>

        <version>3.1.0</version>

        <configuration>

          <mainClass>com.cognizant.springlearn.SpringLearnApplication</mainClass>

        </configuration>

      </plugin>

      <plugin>

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

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

      </plugin>

    </plugins>

  </build>

</project>

This adds all necessary Spring MVC dependencies.

This also adds Spring Boot Plugin in build section.

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.debug("START");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.debug("END");

}

}

Create HelloController in

src/main/java/com/cognizant/spring-learn/controller

**HelloController.java**

package com.cognizant.springlearn.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 message = "Hello World!!";

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

return message;

}}

To run on port 8083, create this file:

application.properties in resources folder

server.port=8083

Run the Project:

mvn clean install

mvn spring-boot:run

Output:

Now open: <http://localhost:8083/hello>

The output will be like

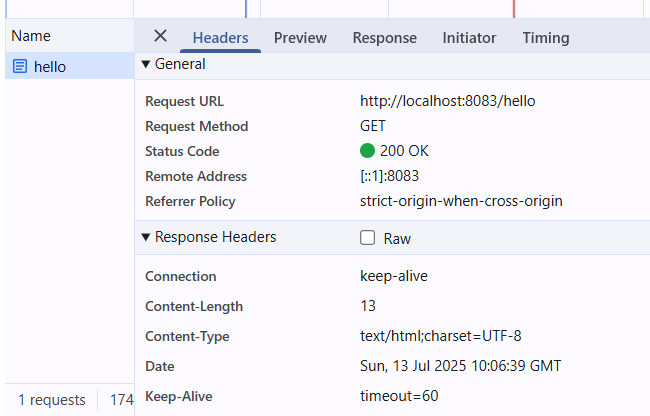
Now, To view Response Headers:

We go to the Network tab in inspect:

In /hello request:

**Headers tab:**

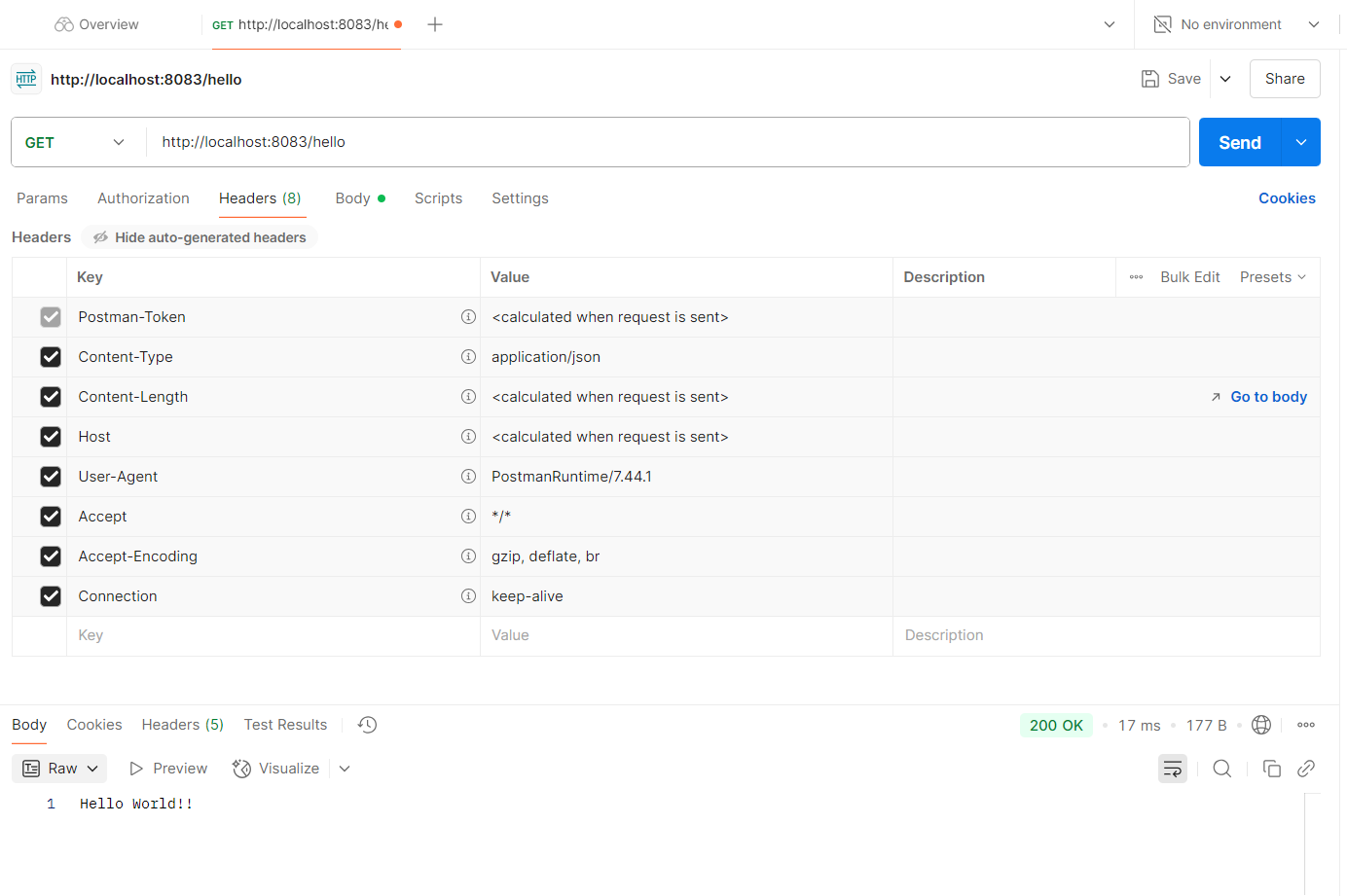
Request URL, Request Method, Status Code

Response Headers like Content-Type, Date, etc.

In Postman,

method: GET

URL: http://localhost:8083/hello

Click send

**Exercise 4:**

**REST - Country Web Service   
Write a service that returns India country details in the earlier created spring learn application.**

URL: /country

Controller Class: CountryController

Return Value: Spring Bean (from country.xml) → auto-converter

**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="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

</beans>

**Country.java**

package com.cognizant.springlearn;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Inside Country Constructor.");

}

public String getCode() {

System.out.println("Getter for code");

return code;

}

public void setCode(String code) {

System.out.println("Setter for code");

this.code = code;

}

public String getName() {

System.out.println("Getter for name");

return name;

}

public void setName(String name) {

System.out.println("Setter for name");

this.name = name;

}

@Override

public String toString() {

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

}

}

In src/main/java/com/cognizant/springlearn/controller:

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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()");

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

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

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

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

return country;

}

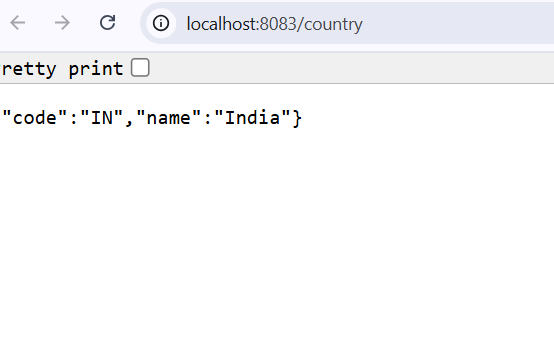
}

}

**Running the Application**

**mvn spring-boot:run**

**Testing in Browser and Postman:** http://localhost:8083/country

**Output: (JSON Response)**

**What happens in the controller method?**

Spring maps the URL /country to getCountryIndia().

country.xml is loaded as a Spring configuration file.

The Country bean is fetched from the context.

Spring returns the Country object as the HTTP response.

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

@RestController automatically returns data.

Spring Boot uses Jackson (ObjectMapper) to convert the Java object (Country) into JSON.

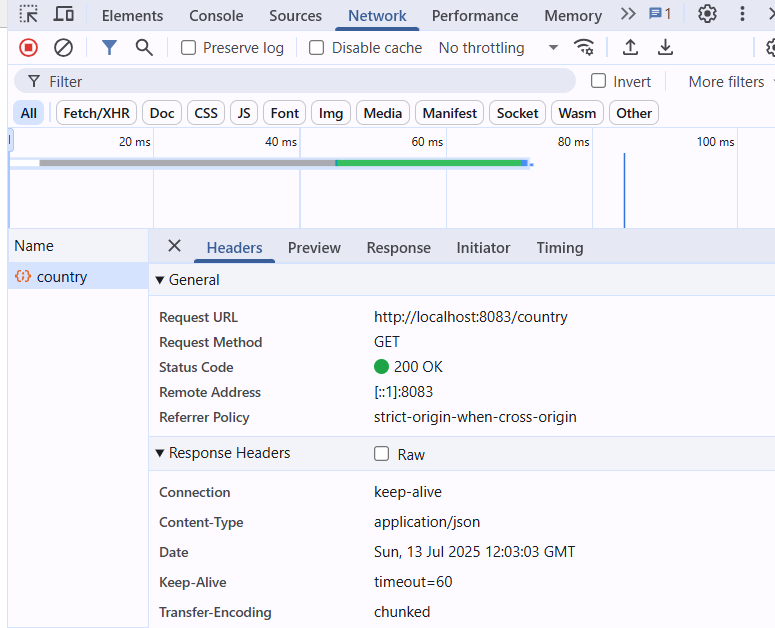
You don’t need to manually serialize it.

View HTTP Headers:

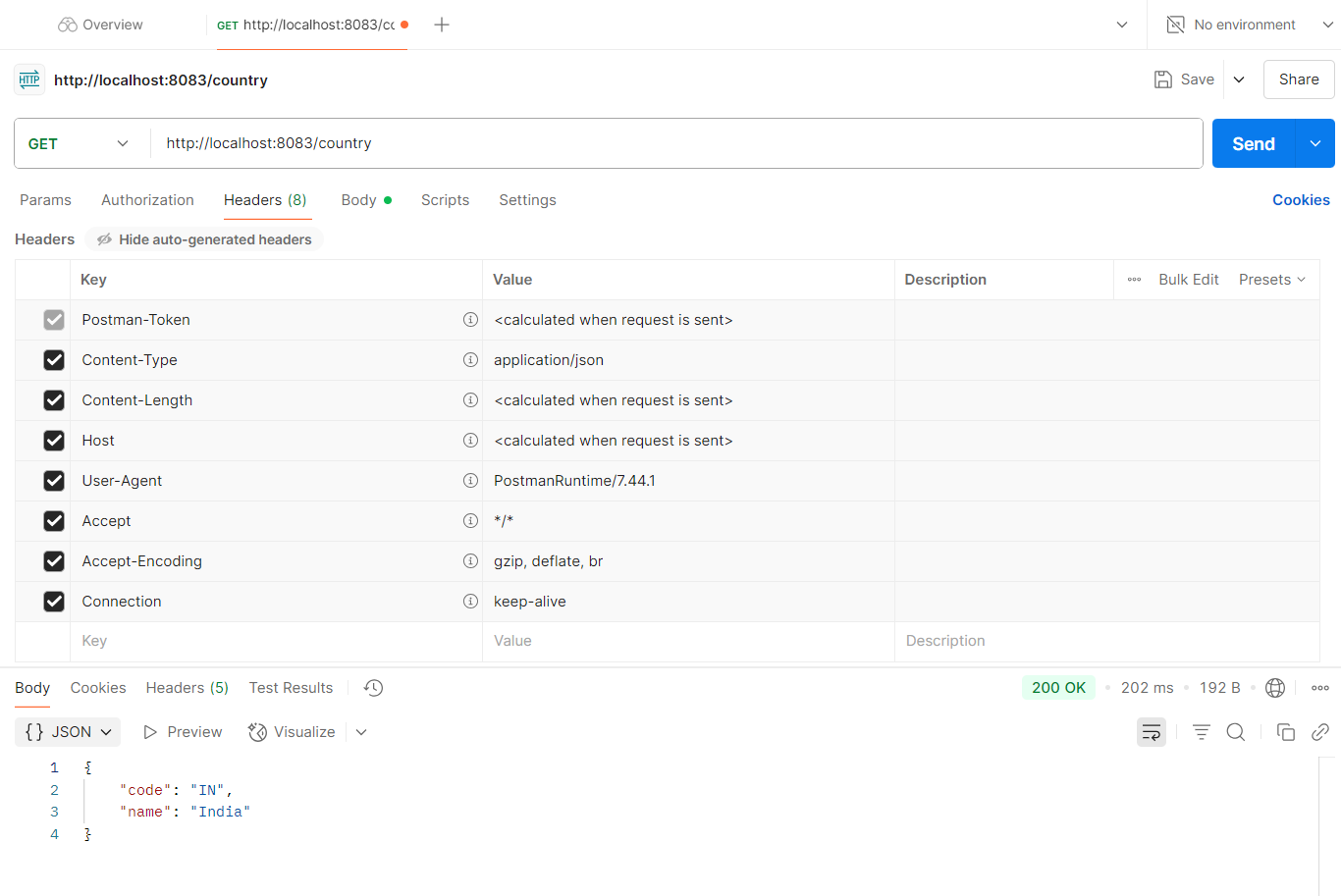
In Chrome Developer Tools:

Visit <http://localhost:8083/country>

In /country request, you will find the Headers present:



**In Postman:**



**Exercise5:**

**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)  
**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="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

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

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

In src/main/java/com/cognizant/springlearn/service:

CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Country;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

try (ClassPathXmlApplicationContext 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);

}

}}

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Country;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

try (ClassPathXmlApplicationContext 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);

}

}

}

**CountryController.java:**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;

import com.cognizant.springlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

LOGGER.debug("START - getCountry() with code: {}", code);

Country country = countryService.getCountry(code);

LOGGER.debug("END - getCountry() result: {}", country);

return country;

}

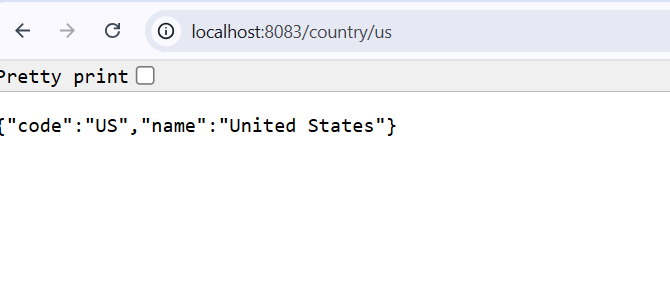
}

**Running the Project:**

mvn spring-boot:run

**Testing the Endpoint:**

[**http://localhost:8083/country/us**](http://localhost:8083/country/us)**:**

****

****

Process:

@PathVariable captures the {code} from the URL.

getCountry() in the controller calls the service method.

The service loads countryList from XML.

It searches (case-insensitively) for a matching code.

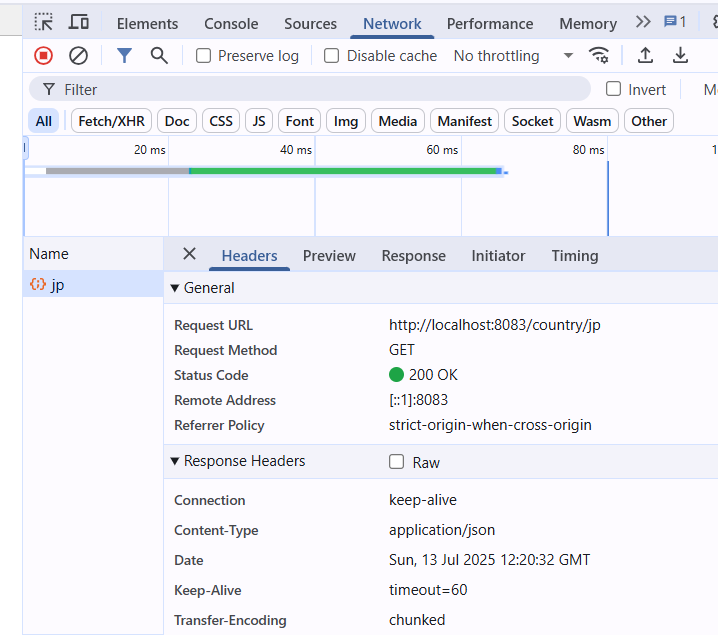
The matching country object is returned as the HTTP response.

Then,

The controller returns a Country object.

Spring Boot uses Jackson (in spring-boot-starter-web) to serialize the Java object to JSON automatically.

Response content type is application/json.



**Exercise 6:**

**JWT Handson**

**Create authentication service that returns JWT** :

Sample Request:

curl -s -u user:pwd <http://localhost:8090/authenticate>

Sample Response:

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"

}

SecurityConfig.java:

@Configuration

public class SecurityConfig {

@Bean

public AuthenticationManager authenticationManager(

AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults()); // Enable Basic Auth

return http.build();

}

}

MyUserDetailsService.java:

@Service

public class MyUserDetailsService implements UserDetailsService {

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

if ("user".equals(username)) {

return new User("user", "{noop}pwd", Collections.emptyList()); // Plain-text password

}

throw new UsernameNotFoundException("User not found");

}

}

JwtUtil.java:

@Component

public class JwtUtil {

private final String SECRET\_KEY = "secret"; // Keep it secure in real apps

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 10 \* 60 \* 1000)) // 10 min

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

AuthController.java:

@RestController

public class AuthController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@Autowired

private UserDetailsService userDetailsService;

@GetMapping("/authenticate")

public Map<String, String> authenticate(HttpServletRequest request) {

String header = request.getHeader("Authorization");

if (header == null || !header.startsWith("Basic ")) {

throw new RuntimeException("Missing or invalid Authorization header");

}

String[] tokens = extractCredentials(header);

String username = tokens[0];

String password = tokens[1];

authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

String token = jwtUtil.generateToken(userDetails.getUsername());

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

private String[] extractCredentials(String header) {

String base64Credentials = header.substring("Basic ".length());

byte[] decoded = Base64.getDecoder().decode(base64Credentials);

return new String(decoded).split(":", 2); // [username, password]

}

}

application.properties:

server.port=8090

pom.xml:

<!-- JWT support -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Needed for Java 11+ -->

<dependency>

<groupId>javax.xml.bind</groupId>

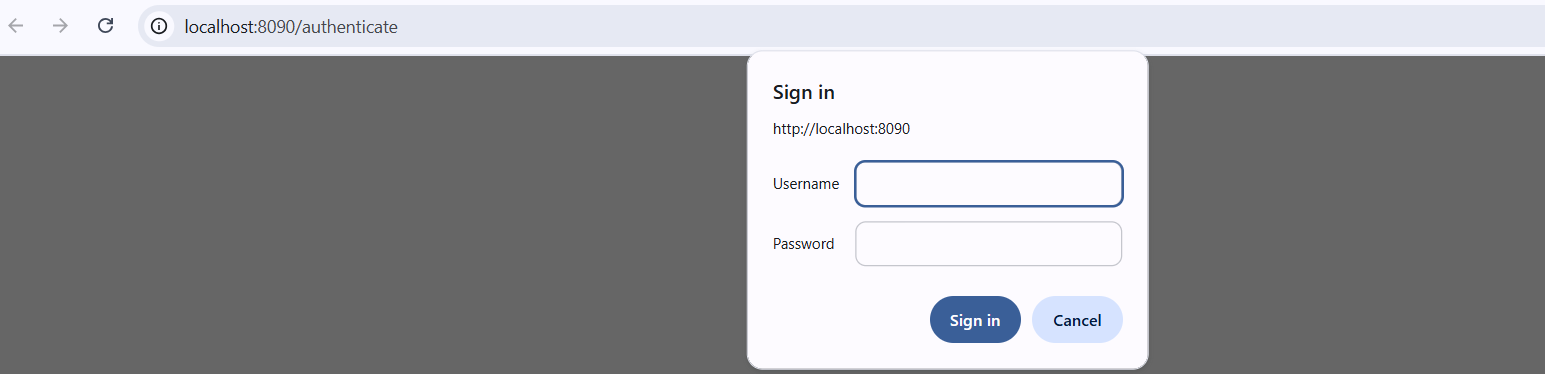
<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

Running the application:

mvn spring-boot:run



Token:

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNjg4NzE1NTk2LCJleHAiOjE2ODg3MTYMTYwfQ.abc123xyz..."

}