WEEK-4

**2.spring-rest-handson**

**Hello World RESTful Web Service**

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<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

                             https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

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

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

        <version>3.5.3</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>spring-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>spring-learn</name>

    <description>Demo project for Spring Boot</description>

    <properties>

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

    </properties>

    <dependencies>

        <!-- Spring Boot Web Starter: REST + Tomcat -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Boot DevTools for auto-restart (Optional) -->

        <dependency>

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

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <!-- SLF4J API and Logging -->

        <dependency>

            <groupId>org.slf4j</groupId>

            <artifactId>slf4j-api</artifactId>

        </dependency>

        <!-- Required for XML-based Spring context loading -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

        </dependency>

        <!-- Spring Boot Test -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Plugin to run the project via mvn exec:java (optional) -->

            <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>

            <!-- Spring Boot plugin: required for packaging/running -->

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**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);  // Bootstraps the web server

        LOGGER.info("END");

    }

}

**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.info("START");

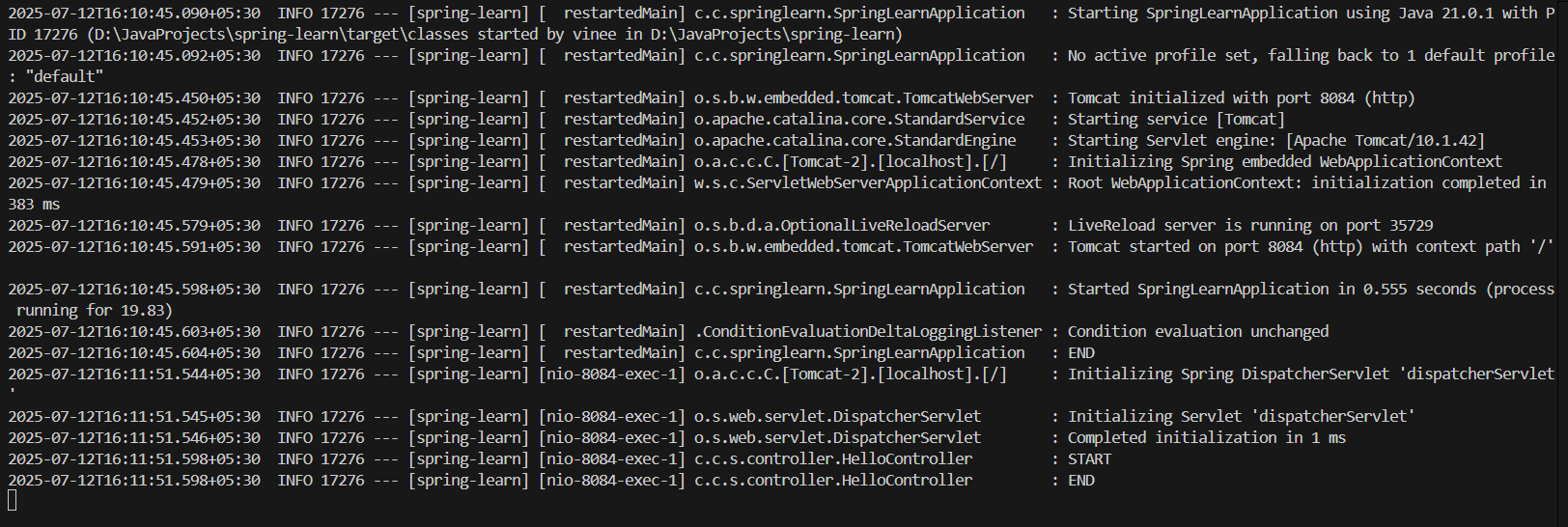
        LOGGER.info("END");

        return "Hello World!!";

    }

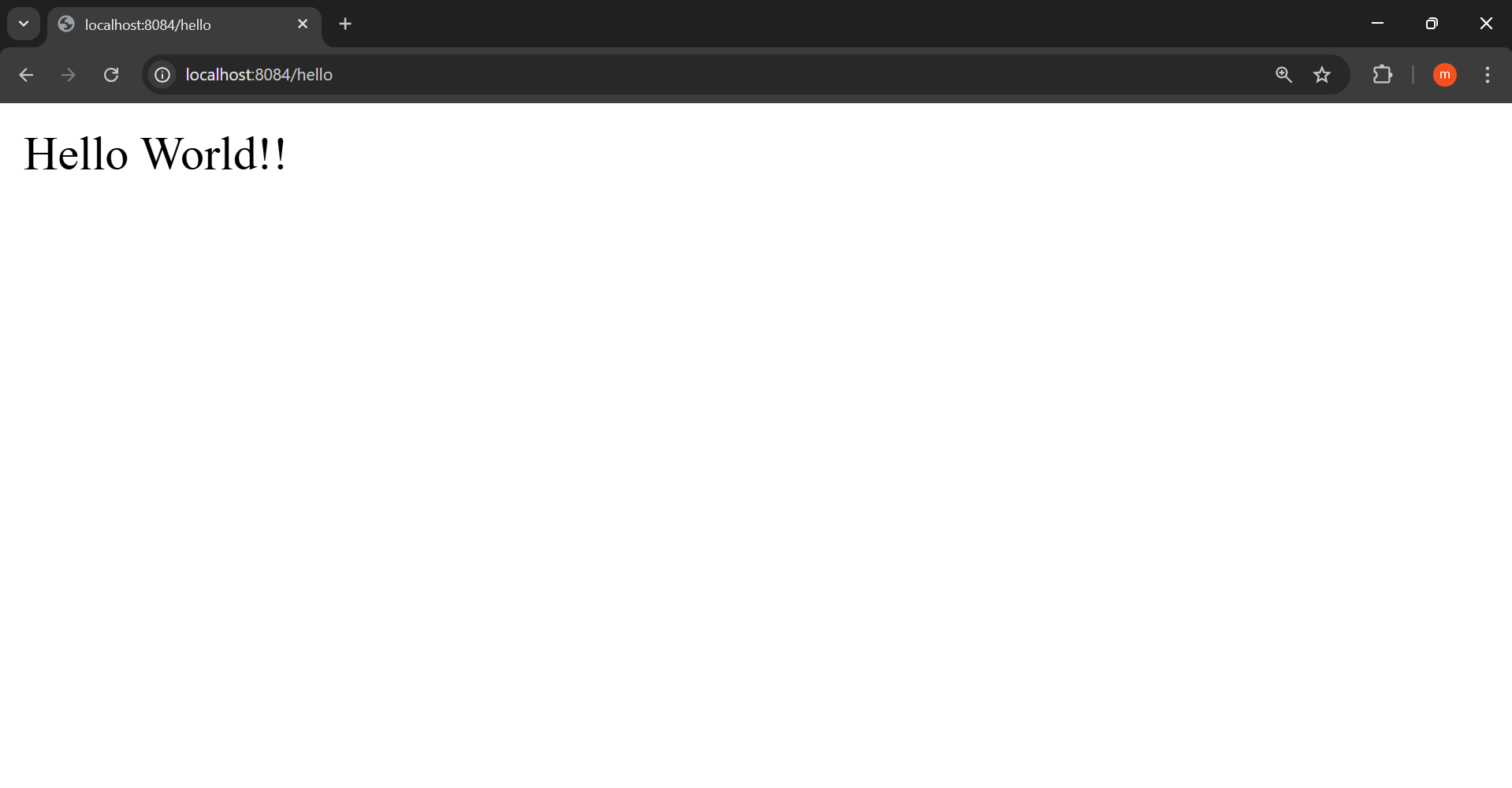
}

**Output:**

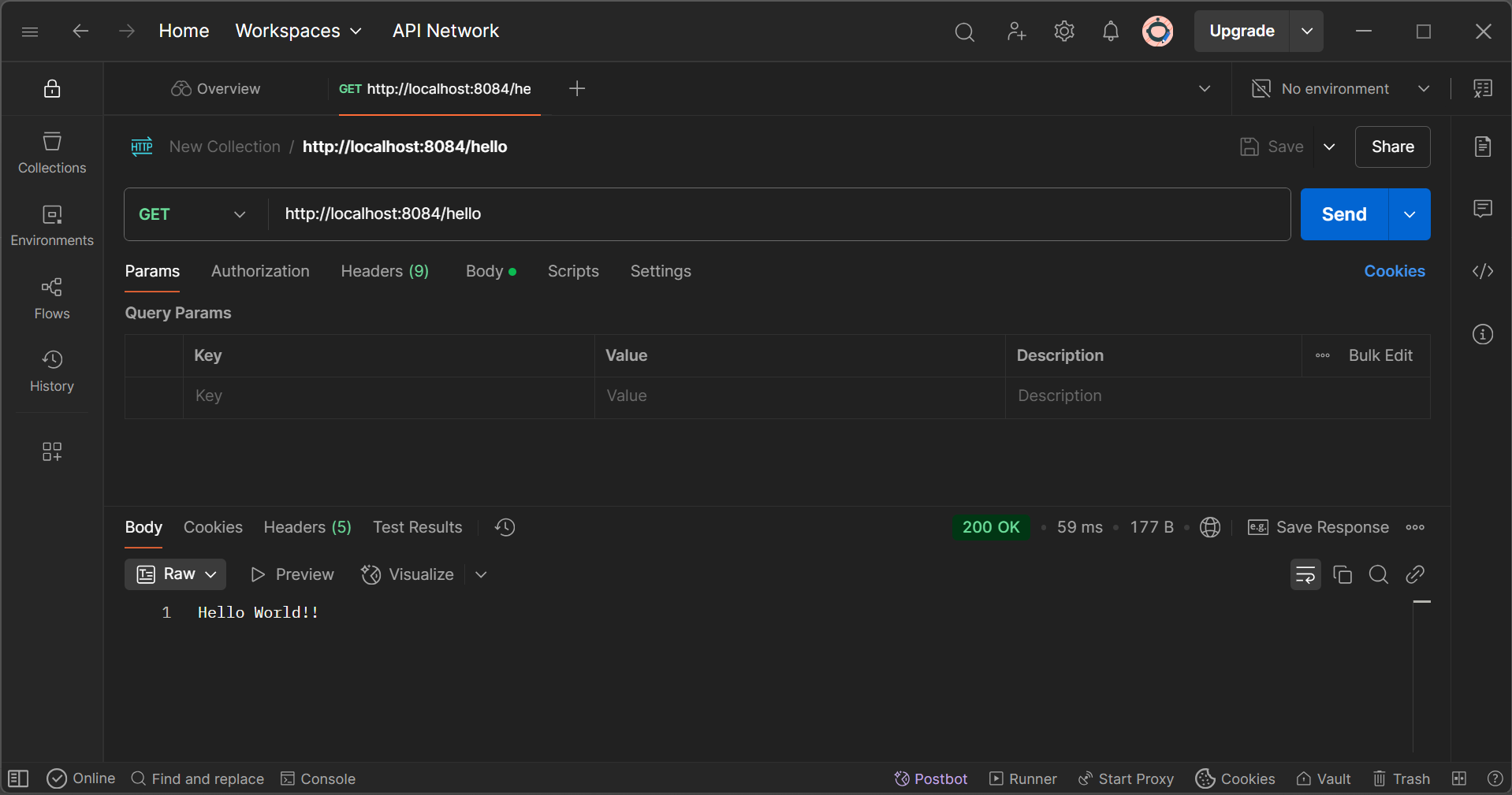


(“ tomcat started on port 8084 “ can be seen above )

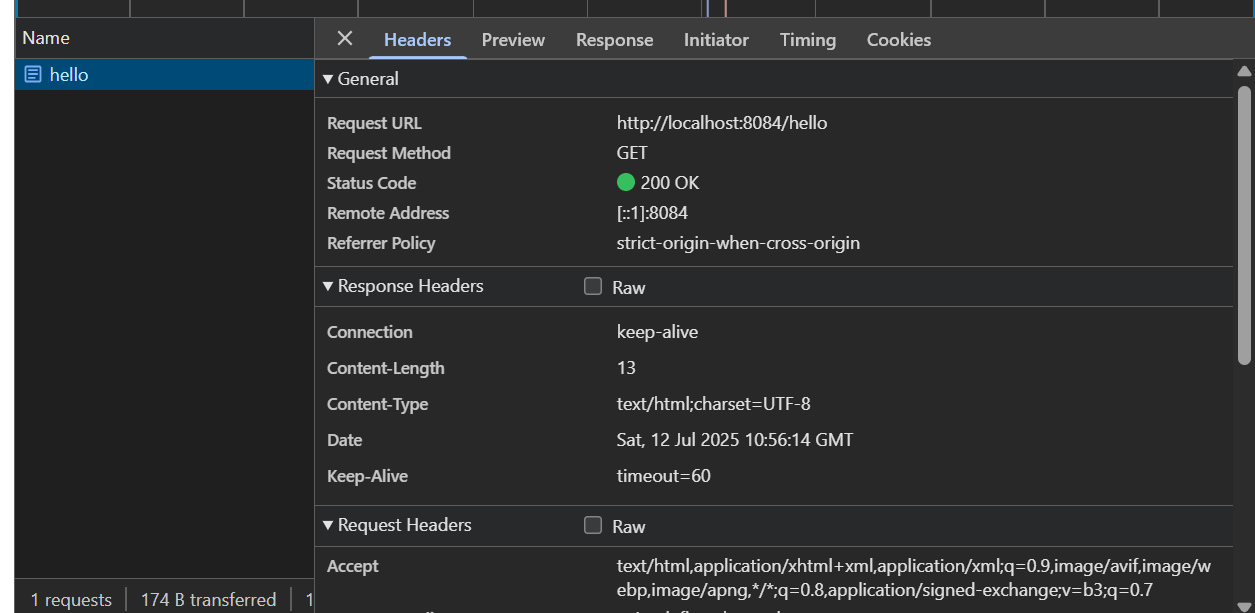
[**http://localhost:8080/hello**](http://localhost:8080/hello)



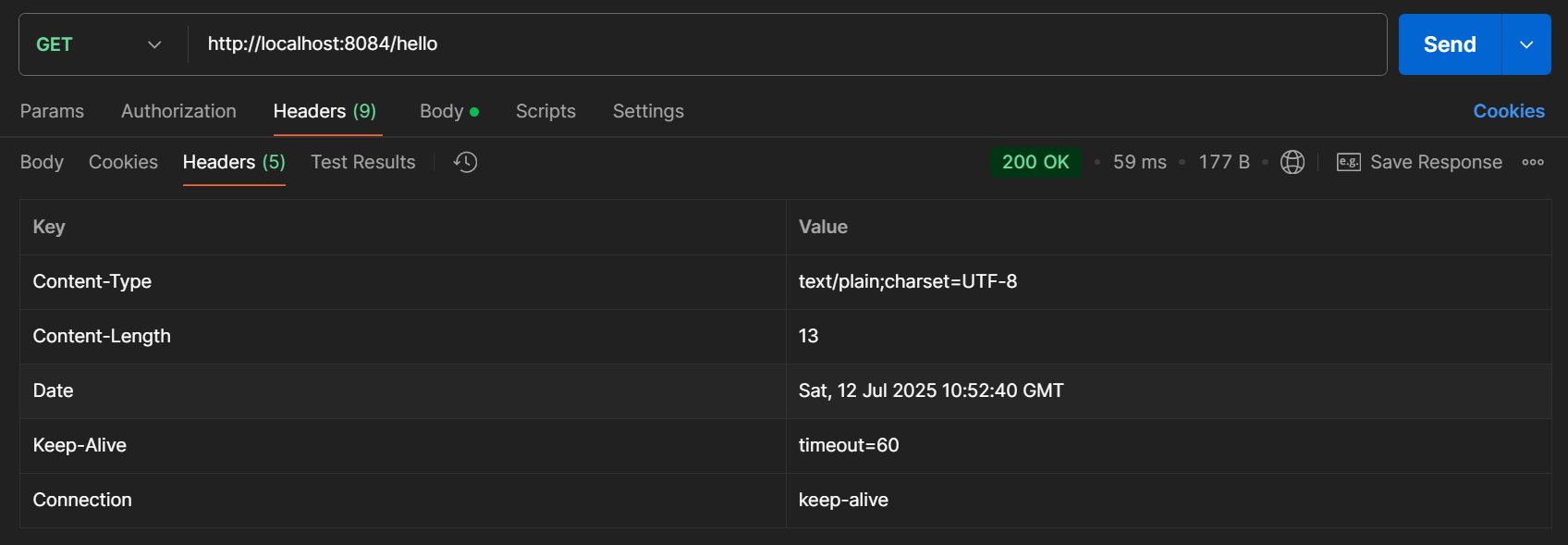
In POSTMAN:



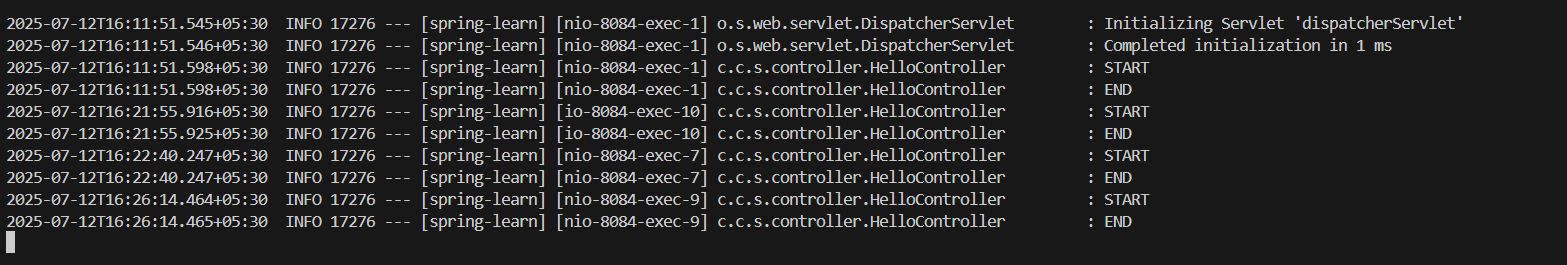
Developer Tools – HTTP Header in Browser



Postman – HTTP Header Details



Log output:



**REST - Country Web Service** 

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<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

                             https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

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

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

        <version>3.5.3</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>spring-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>spring-learn</name>

    <description>Demo project for Spring Boot</description>

    <properties>

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

    </properties>

    <dependencies>

        <!-- Spring Boot Web Starter: REST + Tomcat -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Boot DevTools for auto-restart (Optional) -->

        <dependency>

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

           <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <!-- SLF4J API and Logging -->

        <dependency>

            <groupId>org.slf4j</groupId>

            <artifactId>slf4j-api</artifactId>

        </dependency>

        <!-- Required for XML-based Spring context loading -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

        </dependency>

        <!-- Spring Boot Test -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Plugin to run the project via mvn exec:java (optional) -->

            <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>

            <!-- Spring Boot plugin: required for packaging/running -->

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**country.xml:**

<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() {

        // Logging optional

    }

    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;

    }

    @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.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);  // Bootstraps the web server

        LOGGER.info("END");

    }

}

**CountryController.java:**

package com.cognizant.springlearn.controller;

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;

import com.cognizant.springlearn.Country;

@RestController

public class CountryController {

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

    @RequestMapping("/country")

    public Country getCountryIndia() {

        LOGGER.info("START");

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

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

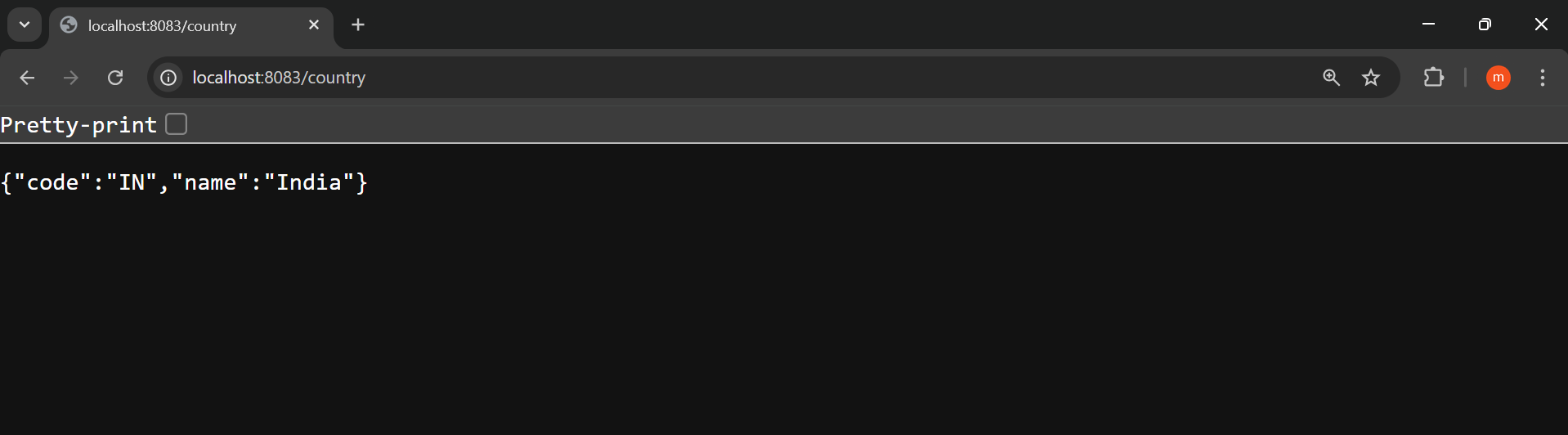
        LOGGER.info("END");

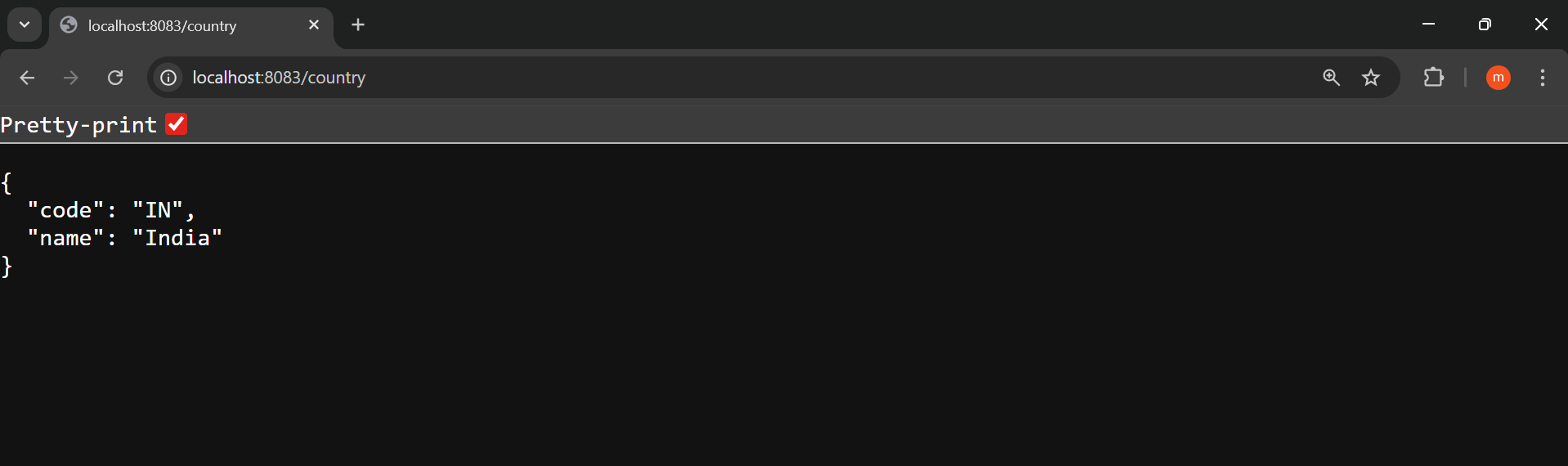
        return country;

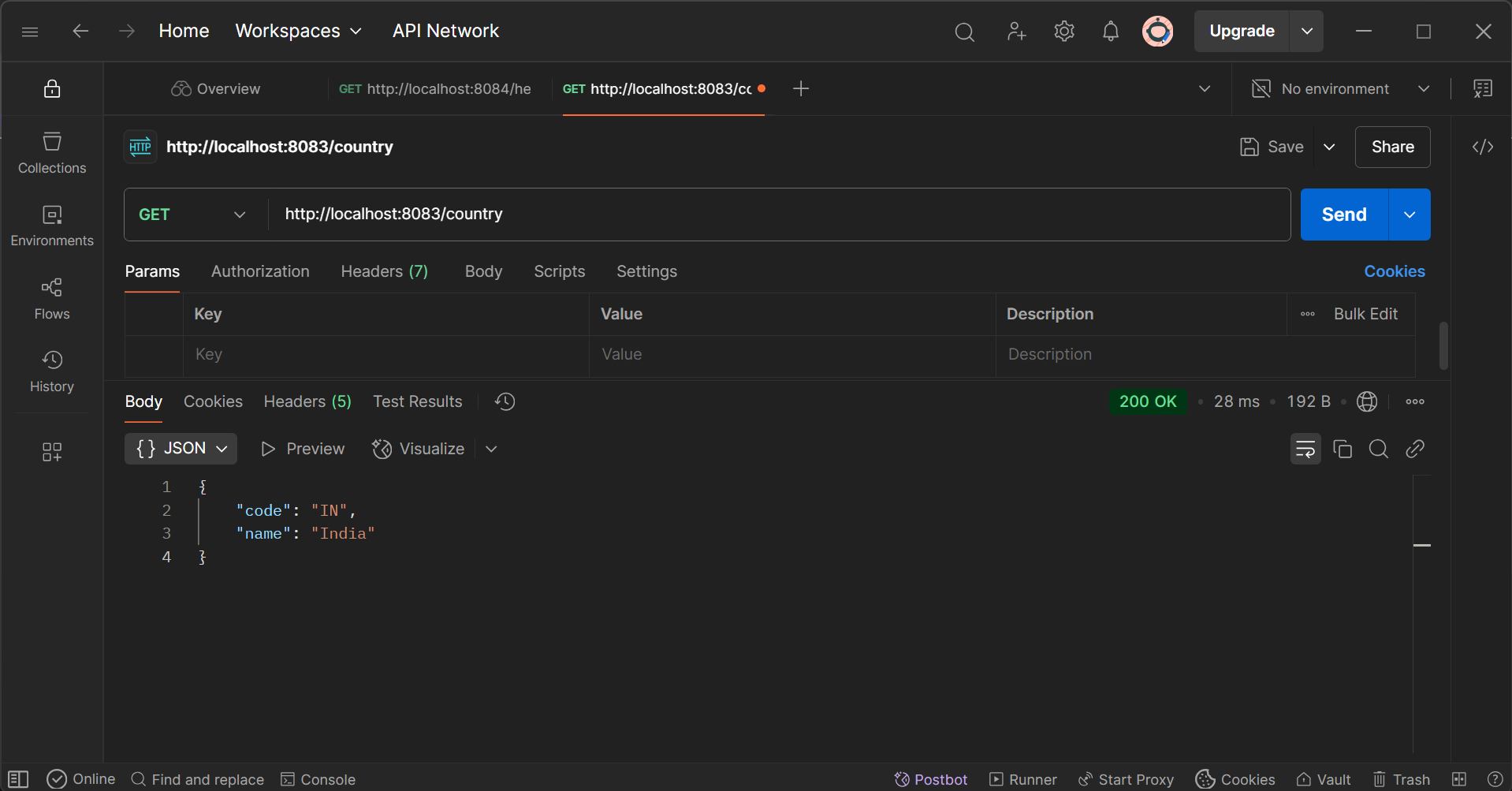
    }

}









**1. What Happens in the Controller Method?**

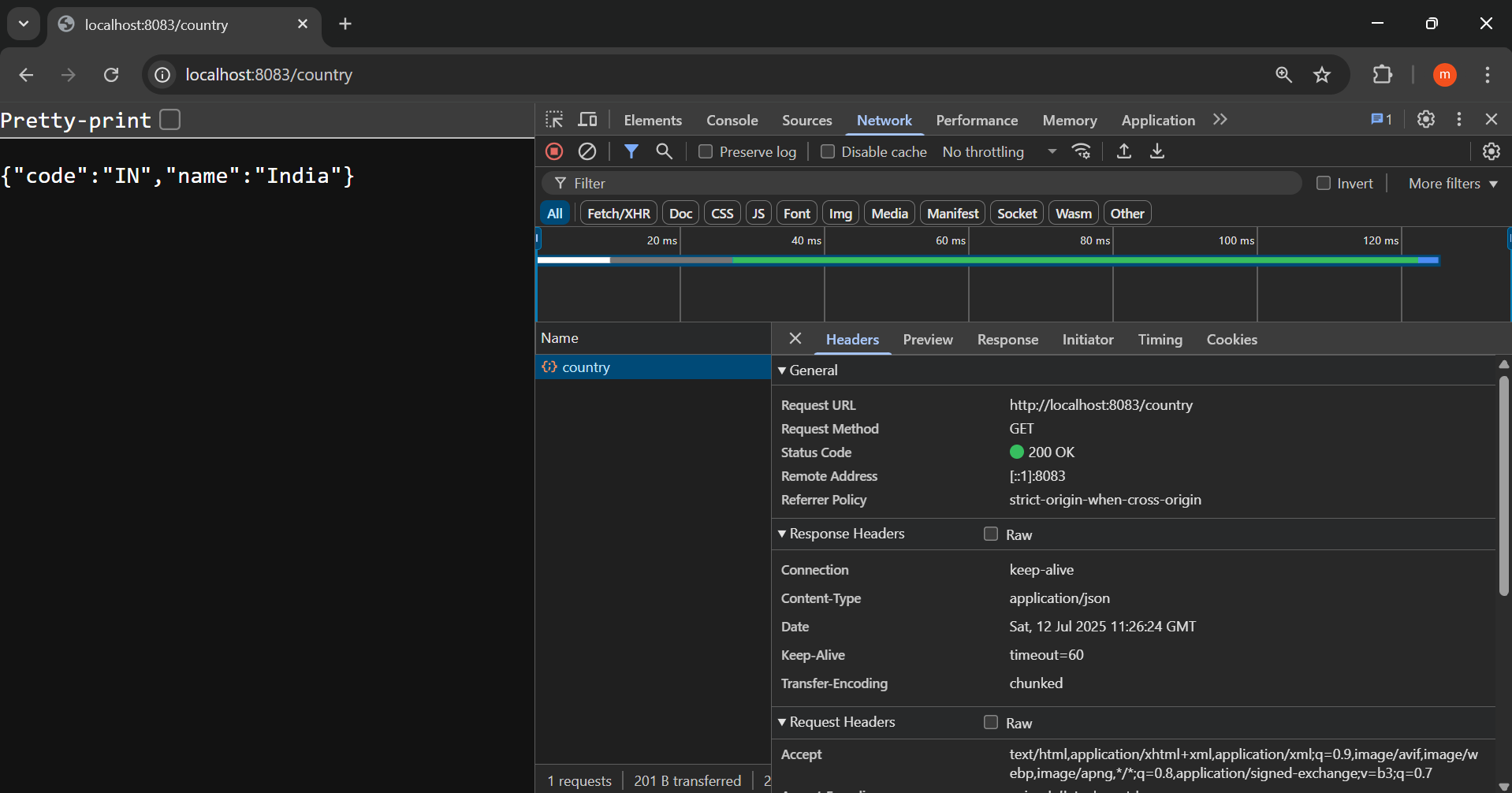
* @RequestMapping("/country") maps the HTTP GET request to the method.
* ClassPathXmlApplicationContext("country.xml") loads the XML configuration.
* context.getBean("country") fetches the Country bean.
* Method returns Country object.

**2. How is the Bean Converted to JSON?**

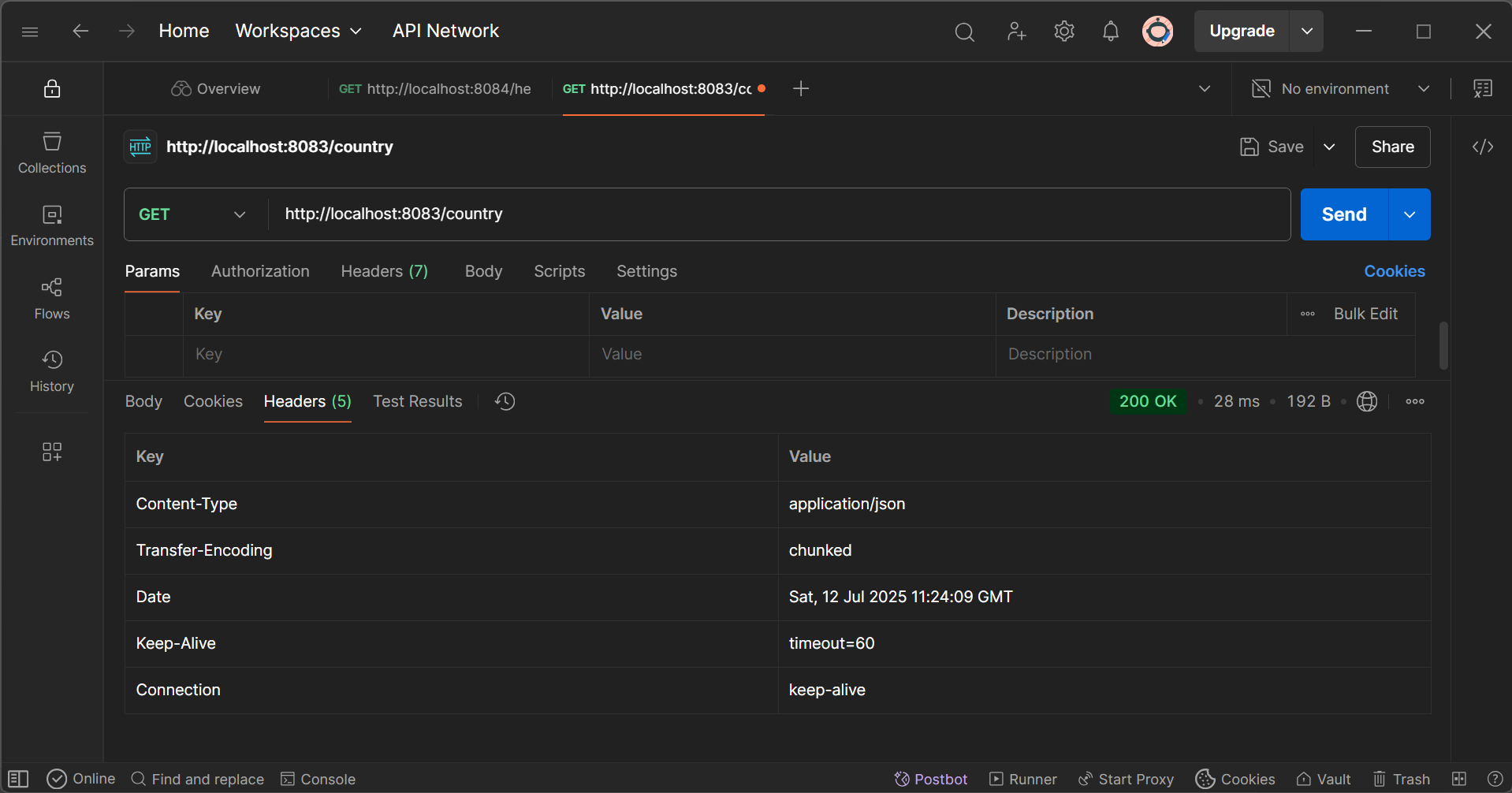
* Spring Boot includes **Jackson** (via spring-boot-starter-web).
* When you return an object (like Country), Spring automatically:
  + Detects the return type
  + Uses Jackson to serialize the object into JSON
  + Sets Content-Type: application/json in response

That’s why returning Country object gives JSON without writing any serialization code.

3. **Network Tab – Chrome DevTools**



4. **Postman – Headers Tab**



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

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<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

                             https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

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

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

        <version>3.5.3</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>spring-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>spring-learn</name>

    <description>Demo project for Spring Boot</description>

    <properties>

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

    </properties>

    <dependencies>

        <!-- Spring Boot Web Starter: REST + Tomcat -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Boot DevTools for auto-restart (Optional) -->

        <dependency>

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

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <!-- SLF4J API and Logging -->

        <dependency>

            <groupId>org.slf4j</groupId>

            <artifactId>slf4j-api</artifactId>

        </dependency>

        <!-- Required for XML-based Spring context loading -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

        </dependency>

        <!-- Spring Boot Test -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Plugin to run the project via mvn exec:java (optional) -->

            <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>

            <!-- Spring Boot plugin: required for packaging/running -->

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**country.xml:**

<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="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="IN"/>

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

                </bean>

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

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

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

                </bean>

            </list>

        </constructor-arg>

    </bean>

</beans>

**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);  // Bootstraps the web server

        LOGGER.info("END");

    }

}

**Country.java:**

package com.cognizant.springlearn;

public class Country {

    private String code;

    private String name;

    public Country() {

        // Logging optional

    }

    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;

    }

    @Override

    public String toString() {

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

    }

}

**CountryService.java:**

package com.cognizant.springlearn.service;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.cognizant.springlearn.Country;

@Service

public class CountryService {

    public Country getCountry(String code) {

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

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

        // Using Java Stream (Lambda)

        return countryList.stream()

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

                .findFirst()

                .orElse(null);

    }

}

**CountryCountoller.java:**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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;

import com.cognizant.springlearn.Country;

import com.cognizant.springlearn.service.CountryService;

@RestController

public class CountryController {

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

    @Autowired

    private CountryService countryService;

    @GetMapping("/countries/{code}")

    public Country getCountry(@PathVariable String code) {

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

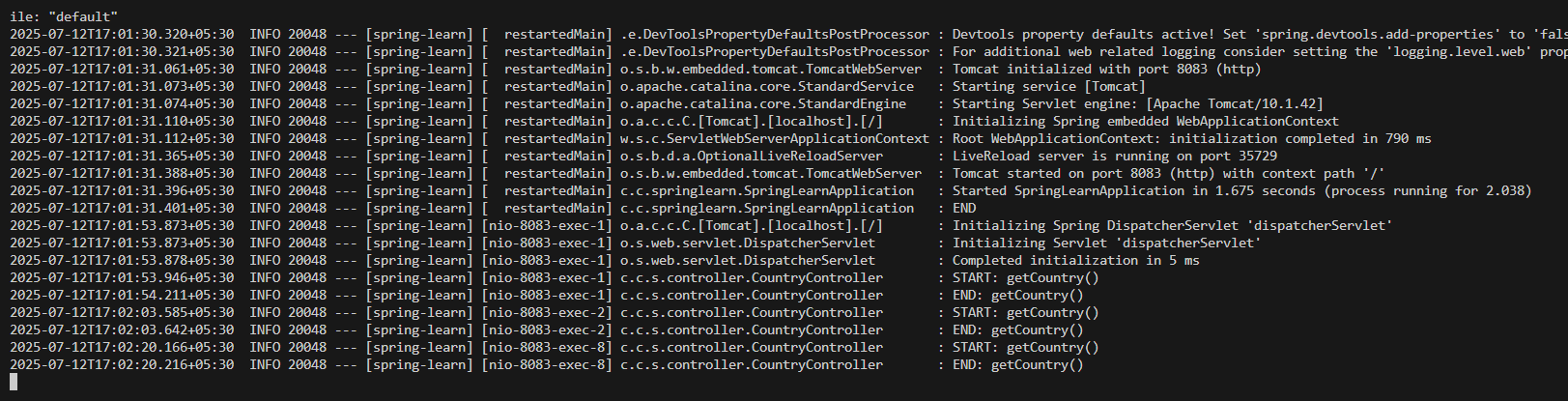
        Country country = countryService.getCountry(code);

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

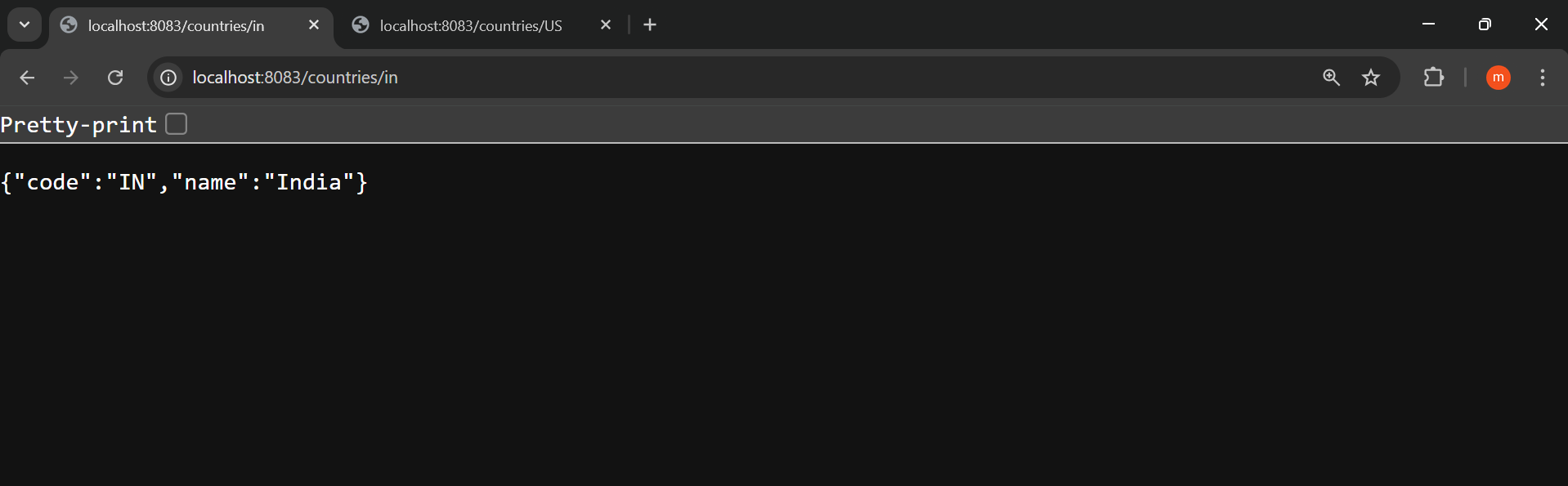
        return country;

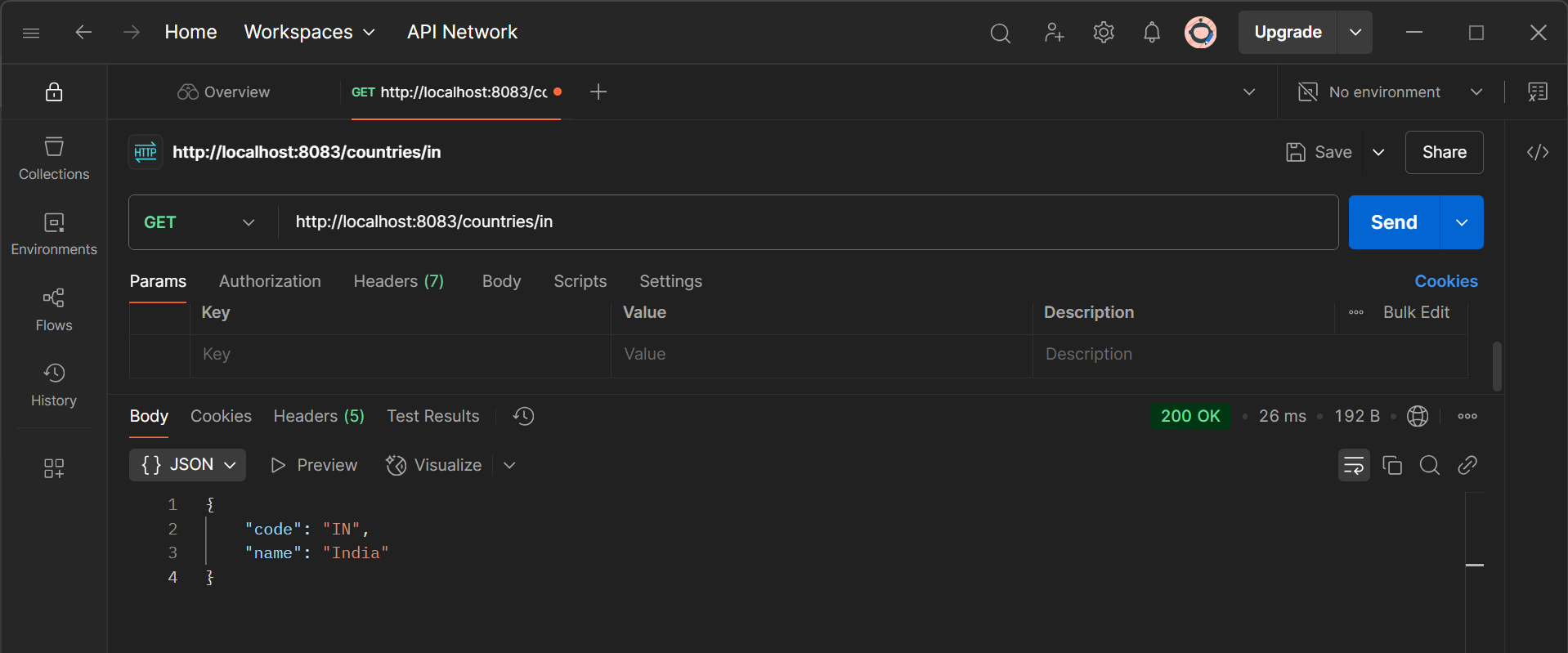
    }

}

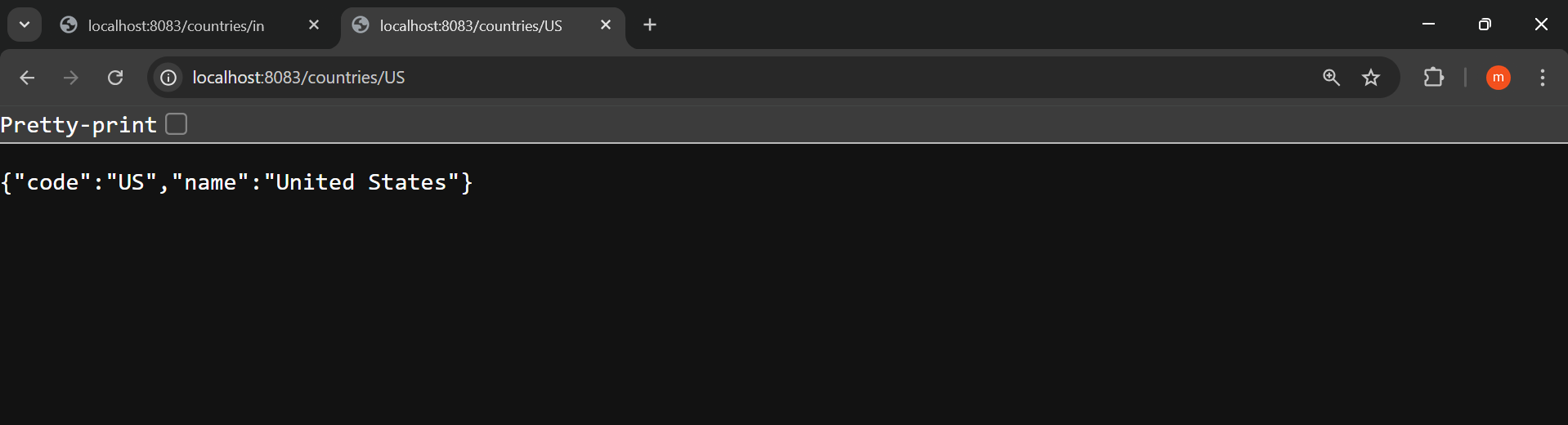


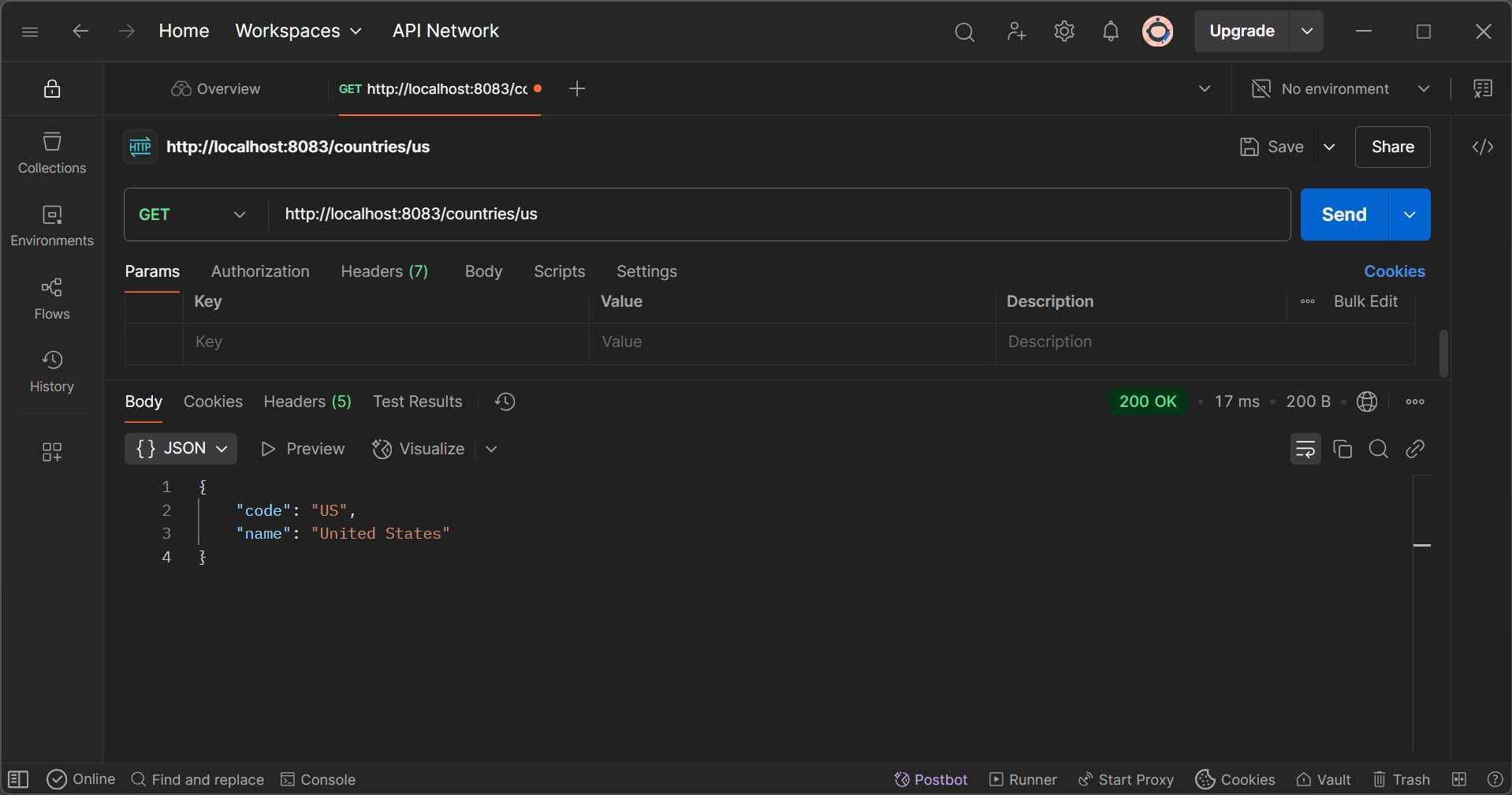
* [**http://localhost:8083/countries/in**](http://localhost:8083/countries/in)





* [**http://localhost:8083/countries/US**](http://localhost:8083/countries/US)





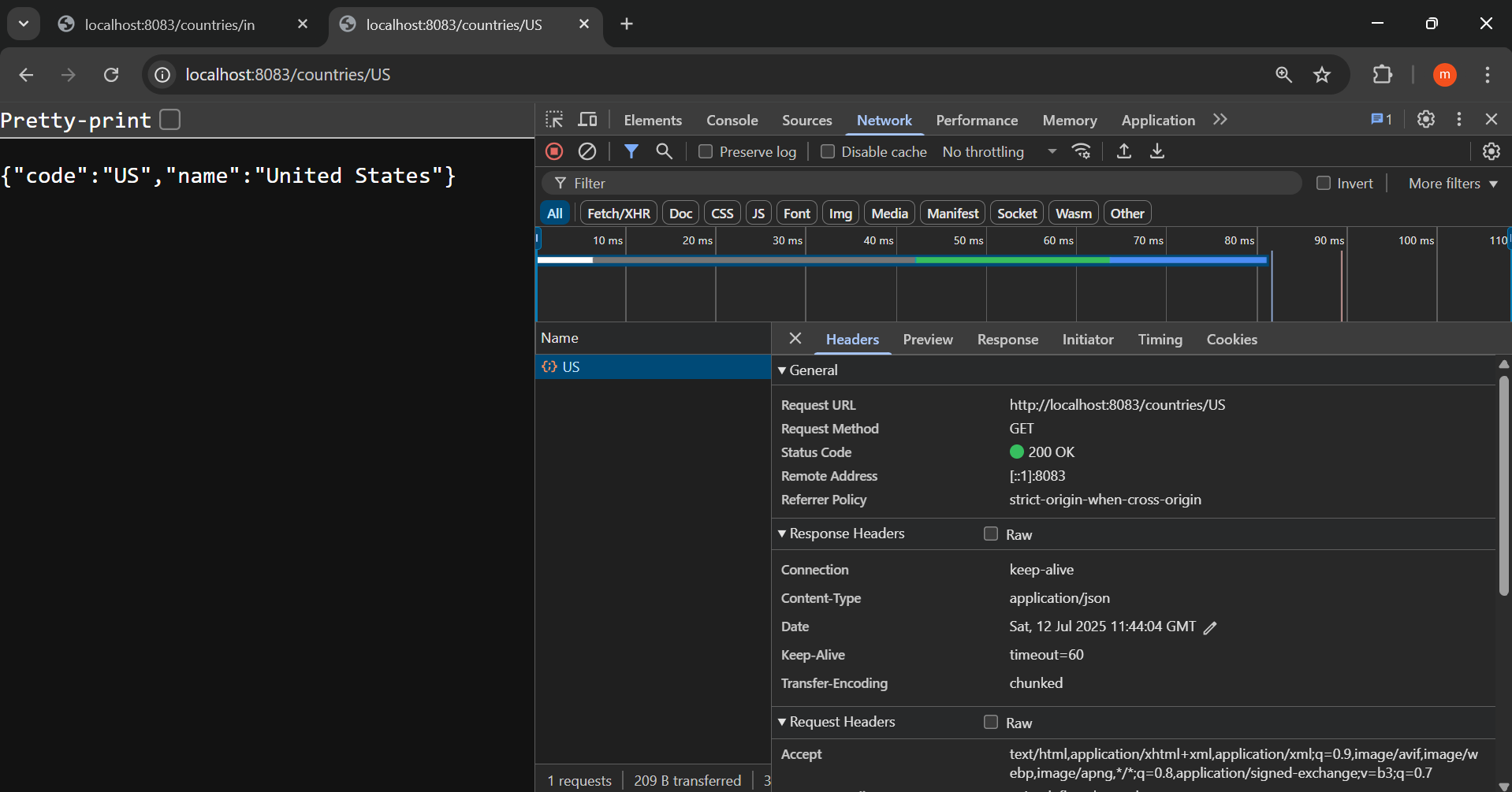
**What Happens in the Controller?**

* @GetMapping("/countries/{code}"): Maps the request to method.
* @PathVariable: Extracts {code} from URL.
* Delegates to countryService.getCountry(code).

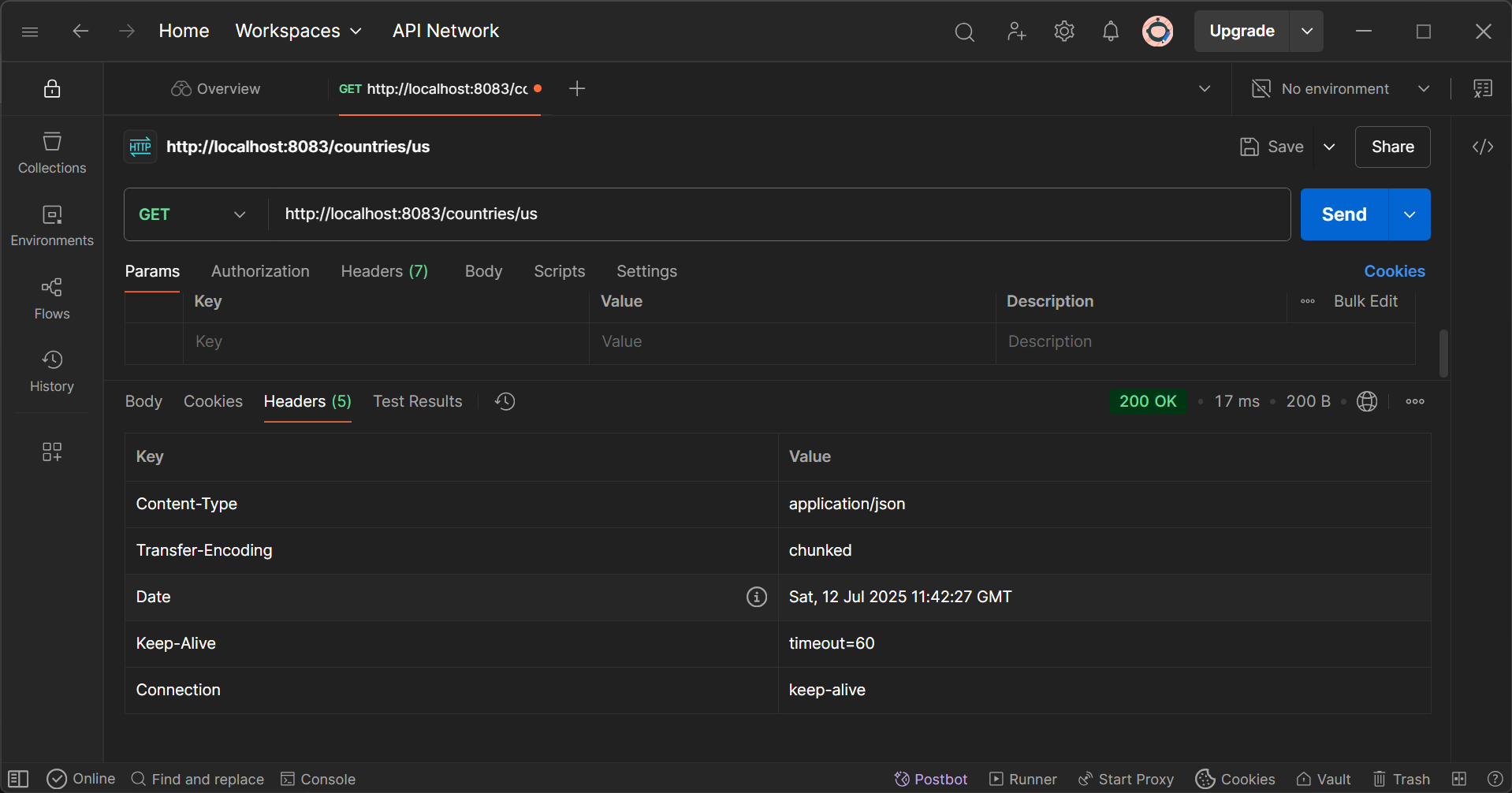
**How Bean Converts to JSON?**

* Return type is Country.
* Spring Boot auto-converts POJO → JSON using **Jackson** (no need to write JSON manually).
* Sets response Content-Type: application/json.

**HTTP Header Info – Developer Tools**

****

**Postman – HTTP Headers**

****