**1. spring-rest-handson**

**Hands on 1**

**Create a Spring Web Project using Maven** 

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);

}

}

<?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>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration 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="dateFormat" class="java.text.SimpleDateFormat">

        <constructor-arg value="dd/MM/yyyy" />

    </bean>

</beans>

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.text.SimpleDateFormat;

import java.util.Date;

*@SpringBootApplication*

public class SpringLearnApplication {

    public static void main(String[] args) {

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

*displayDate*();

    }

    public static void displayDate() {

        ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

        SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

        try {

            Date date = format.parse("31/12/2018");

            System.***out***.println("Parsed Date: " + date);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Hands on 3**

**Spring Core - Incorporate Logging**

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

server.port=8081

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@SpringBootApplication*

public class SpringLearnApplication {

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

    public static void main(String[] args) {

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

***LOGGER***.info("Inside main()");

*displayDate*();

    }

    public static void displayDate() {

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

        ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

        SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

        try {

            Date date = format.parse("31/12/2018");

***LOGGER***.debug("Parsed Date: {}", date);

        } catch (Exception e) {

***LOGGER***.error("Error parsing date", e);

        }

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

    }

}

**Hands on 4**

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

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 + "]";

    }

}

<?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.springlearn.Country">

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

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

    </bean>

</beans>

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@SpringBootApplication*

public class SpringLearnApplication {

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

    public static void main(String[] args) {

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

***LOGGER***.info("Inside main()");

*displayCountry*();

    }

    public static void displayCountry() {

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

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

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

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

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

    }

**Hands on 5**

**Spring Core – Demonstration of Singleton Scope and Prototype Scope**

<?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.springlearn.Country">

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

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

</bean>

</beans>

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@SpringBootApplication*

public class SpringLearnApplication {

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

    public static void main(String[] args) {

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

***LOGGER***.info("Inside main()");

*displayCountry*();

    }

    public static void displayCountry() {

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

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

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

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

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

***LOGGER***.debug("Another Country : {}", anotherCountry);

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

    }

}

<?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.springlearn.Country" scope="prototype">

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

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

</bean>

</beans>

**Hands on 6**

**Spring Core – Load list of countries from Spring Configuration 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.cognizant.springlearn.Country">

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

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

    </bean>

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

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

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

    </bean>

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

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

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

    </bean>

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

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

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

    </bean>

    <!-- List of countries -->

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

        <constructor-arg>

            <list>

                <ref bean="in" />

                <ref bean="us" />

                <ref bean="de" />

                <ref bean="jp" />

            </list>

        </constructor-arg>

    </bean>

</beans>

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@SpringBootApplication*

public class SpringLearnApplication {

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

    public static void main(String[] args) {

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

***LOGGER***.info("Inside main()");

        //displayCountry();

*displayCountries*();

    }

    public static void displayCountry() {

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

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

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

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

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

***LOGGER***.debug("Another Country : {}", anotherCountry);

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

    }

*@SuppressWarnings*("unchecked")

    public static void displayCountries() {

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

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

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

        for (Country country : countries) {

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

        }

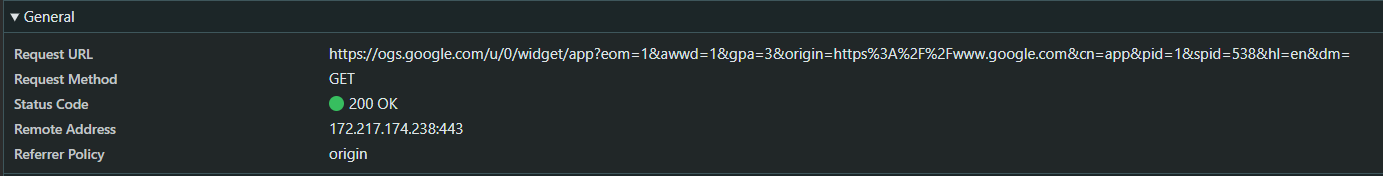
***LOGGER***.info("END");

    }

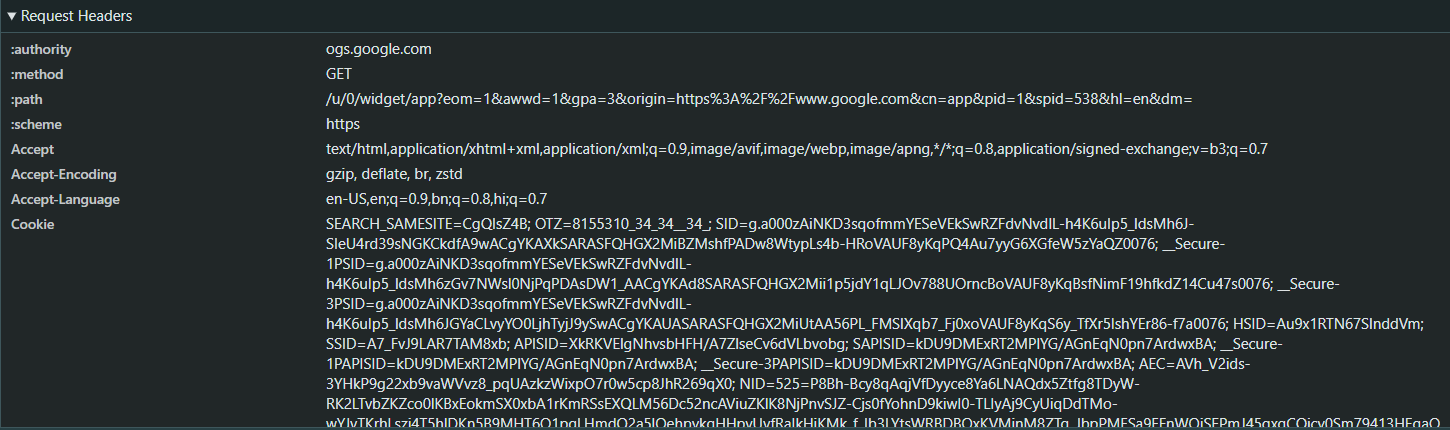
}

**2. spring-rest-handson**

**HTTP Request Response**







**Hello World RESTful Web Service**

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 - sayHello()");

       String response = "Hello World!!";

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

       return response;

   }

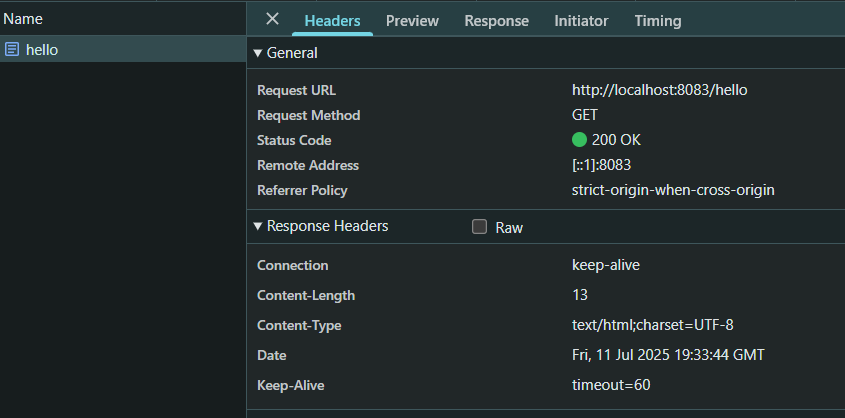
}

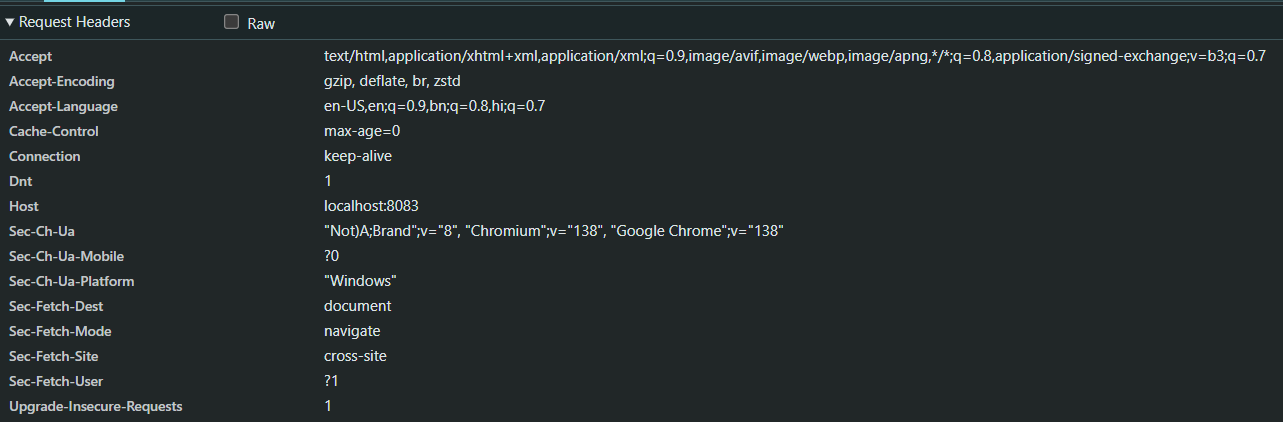
logging.level.org.springframework=info

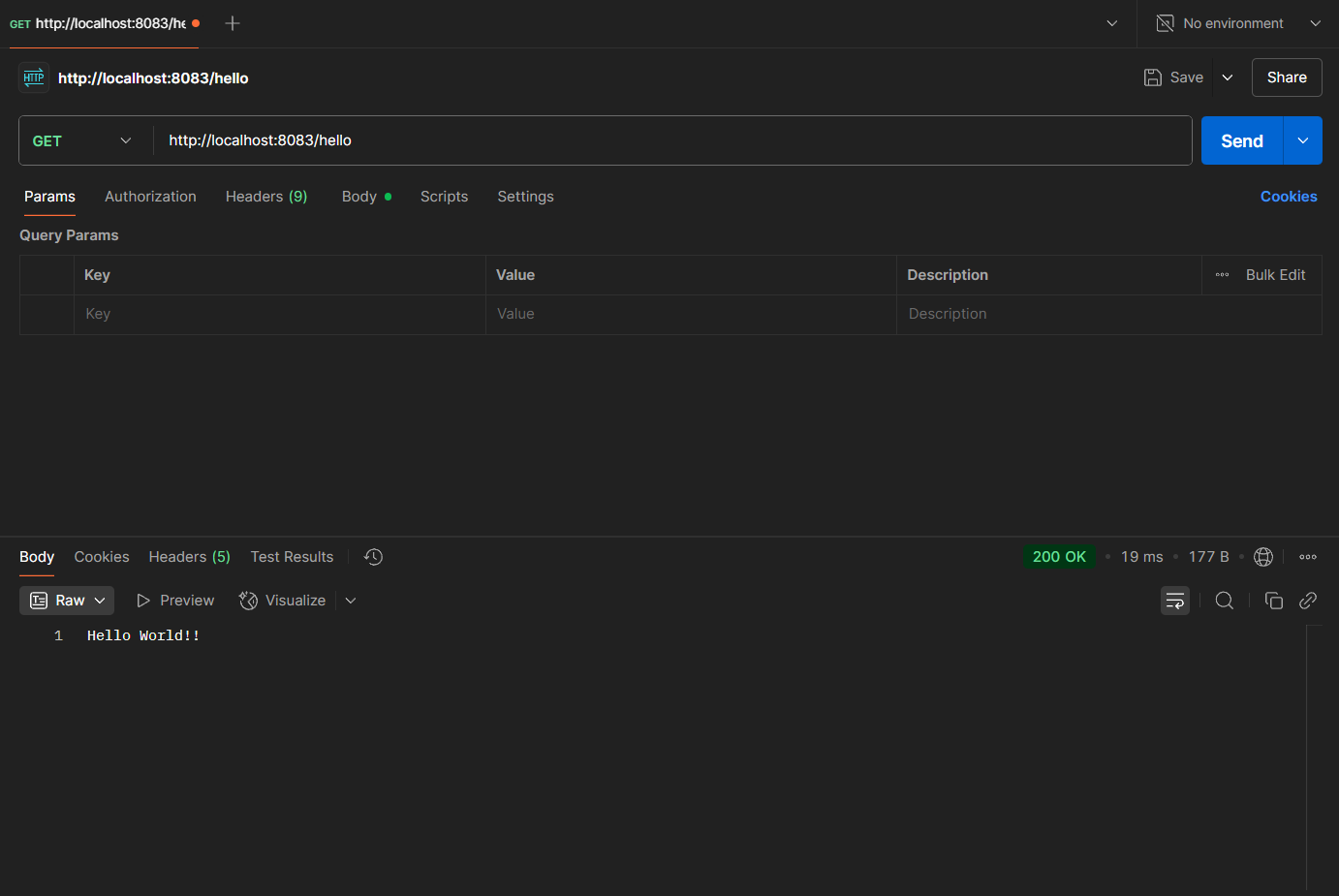
logging.level.com.cognizant.springlearn=debug

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

server.port=8083







**REST - Country Web Service**

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;

   }

}

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

<beans xmlns="http://www.springframework.org/schema/beans"

      xmlns:context="http://www.springframework.org/schema/context"

      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

                          http://www.springframework.org/schema/context

                          http://www.springframework.org/schema/context/spring-context.xsd">

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

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

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

   </bean>

</beans>

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ImportResource;

*@SpringBootApplication*

*@ImportResource*("classpath:country.xml")  // Load XML beans

public class SpringLearnApplication {

   public static void main(String[] args) {

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

   }

}

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.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***.info("START - getCountryIndia()");

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

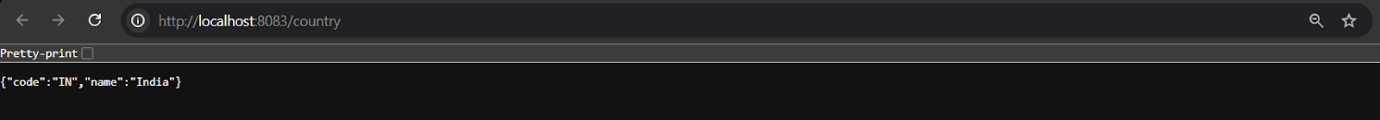
       Country country = (Country) context.getBean("in");

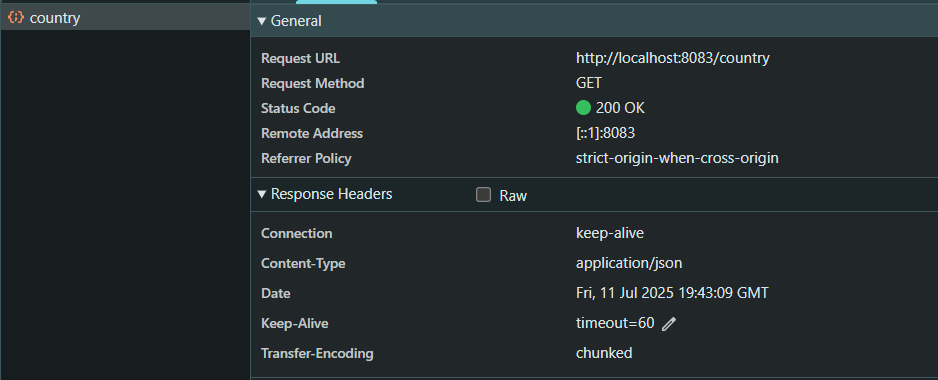
***LOGGER***.info("END - getCountryIndia()");

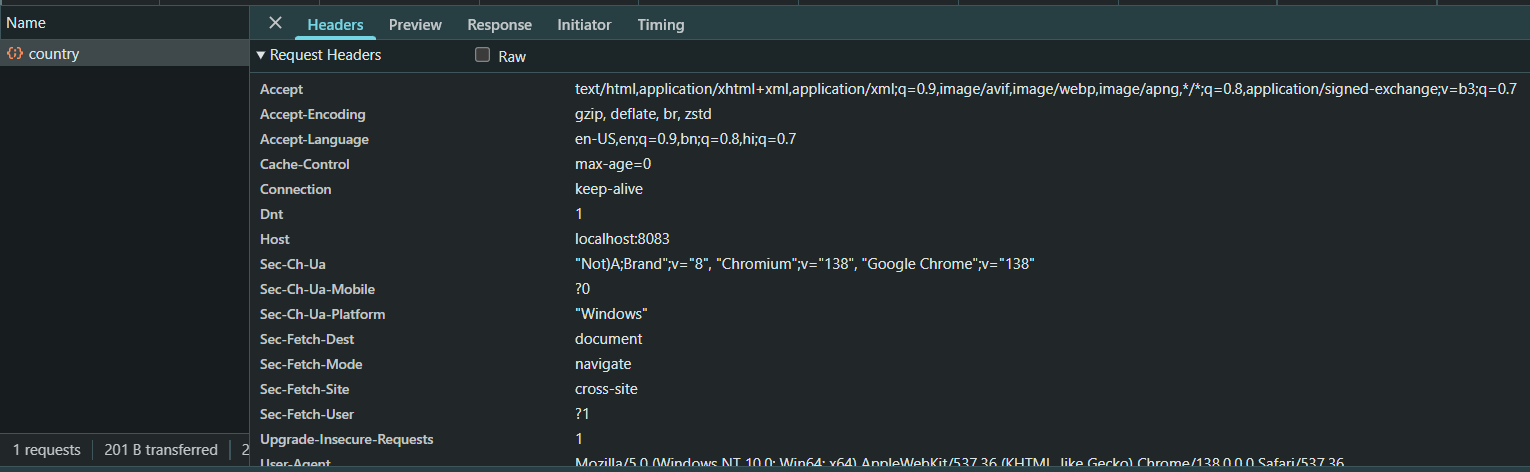
       return country;

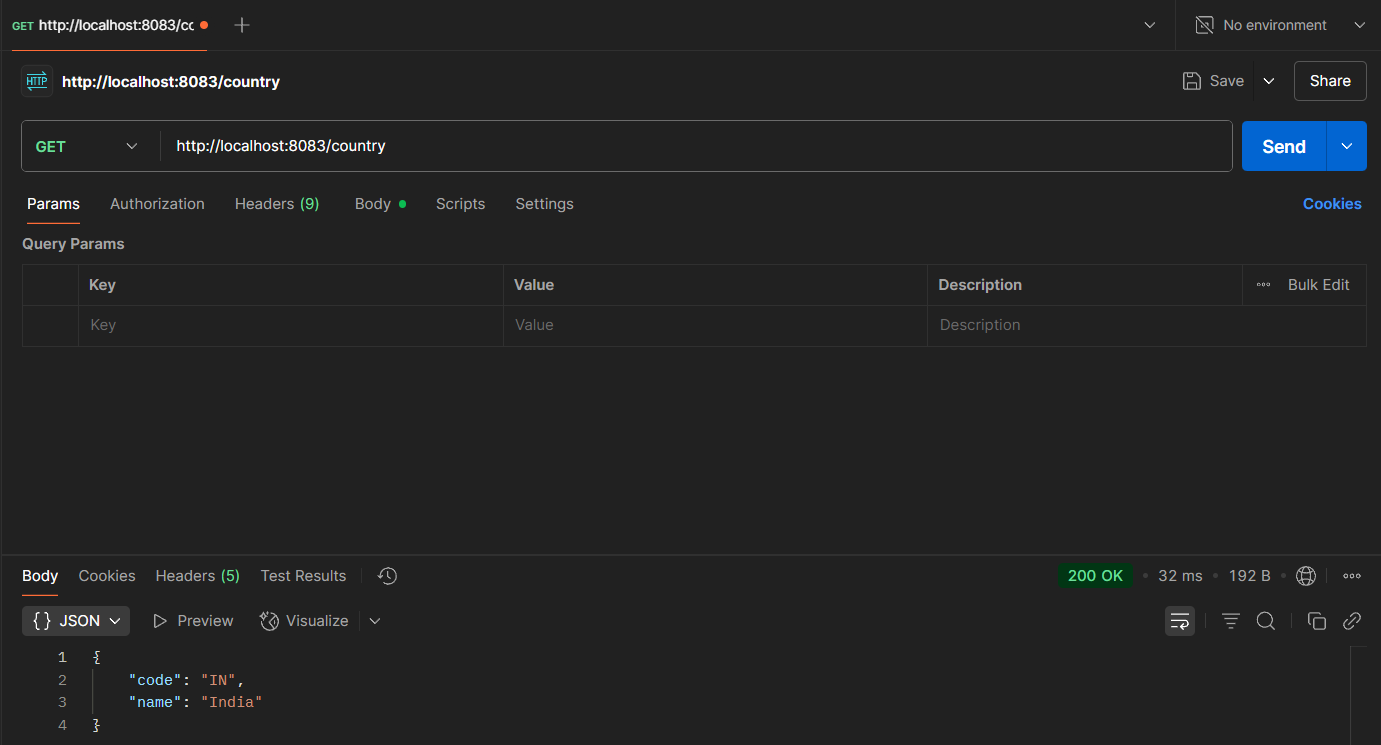
   }

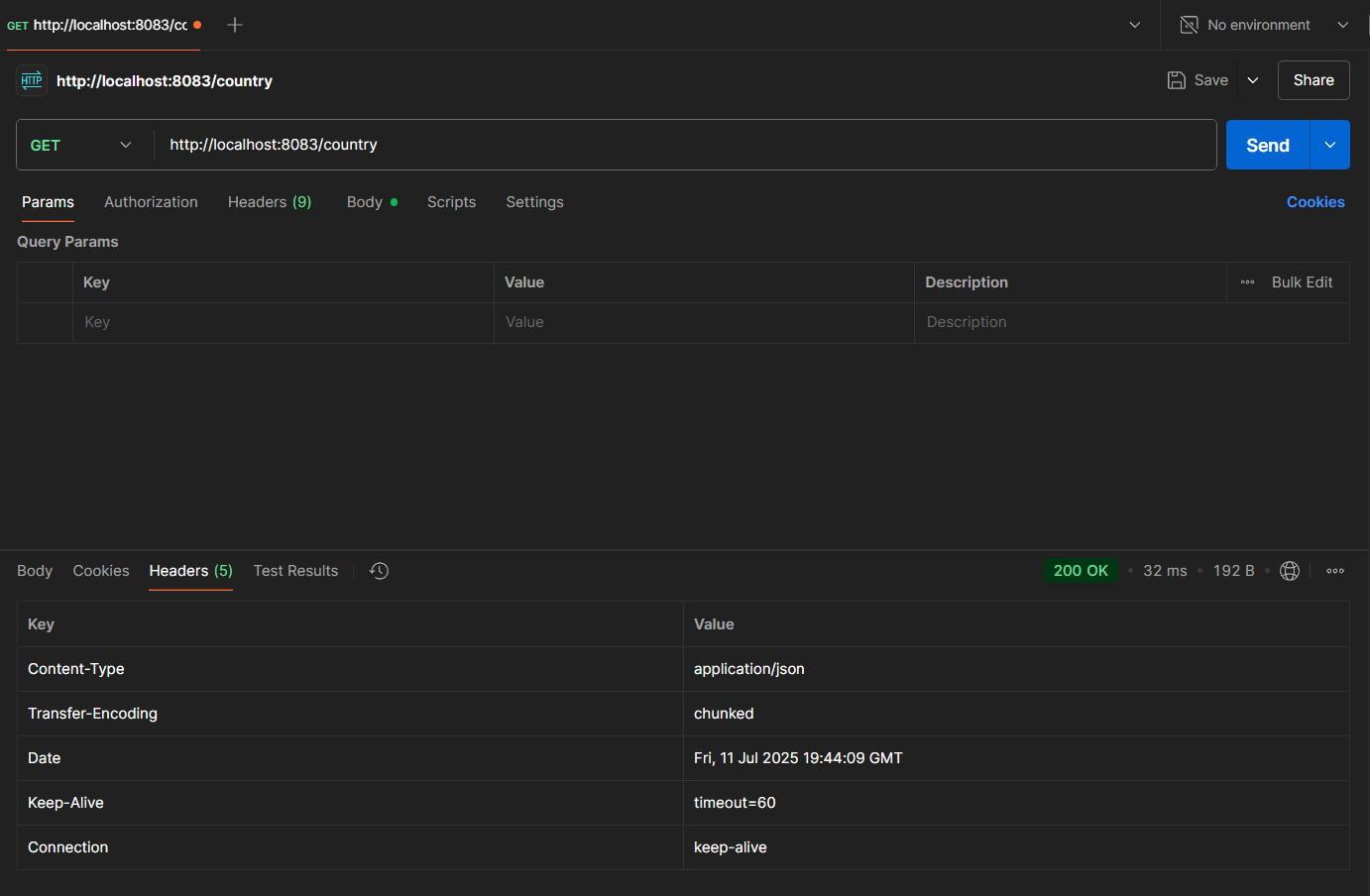
}











**REST - Get all countries**

<?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>

   <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="JP" />

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

               </bean>

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

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

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

               </bean>

           </list>

       </constructor-arg>

   </bean>

</beans>

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.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.GetMapping;

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

import java.util.List;

*@RestController*

public class CountryController {

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

*@GetMapping*("/country")

   public Country getCountryIndia() {

***LOGGER***.info("START - getCountryIndia()");

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

       Country country = (Country) context.getBean("in");

***LOGGER***.info("END - getCountryIndia()");

       return country;

   }

*@GetMapping*("/countries")

   public List<Country> getAllCountries() {

***LOGGER***.info("START - getAllCountries()");

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

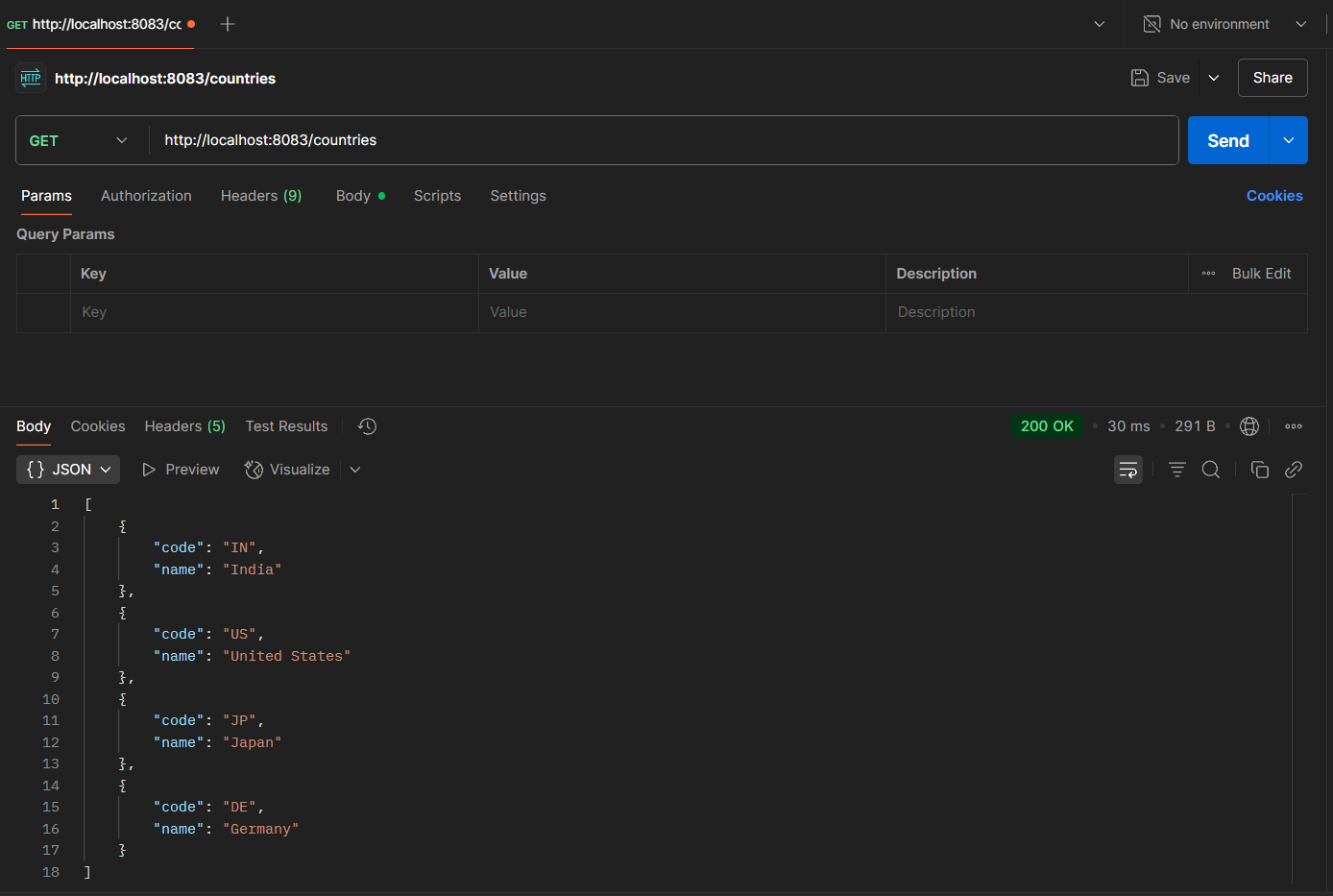
       List<Country> countries = (List<Country>) context.getBean("countryList");

***LOGGER***.info("END - getAllCountries()");

       return countries;

   }

}



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

<?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>

   <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="JP" />

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

           </bean>

       </list>

   </constructor-arg>

</bean>

</beans>

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

       // Using lambda for case-insensitive match

       return countries.stream()

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

               .findFirst()

               .orElse(null); // You can throw exception if not found

   }

}

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.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*("/countries/{code}")

   public Country getCountry(*@PathVariable* String code) {

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

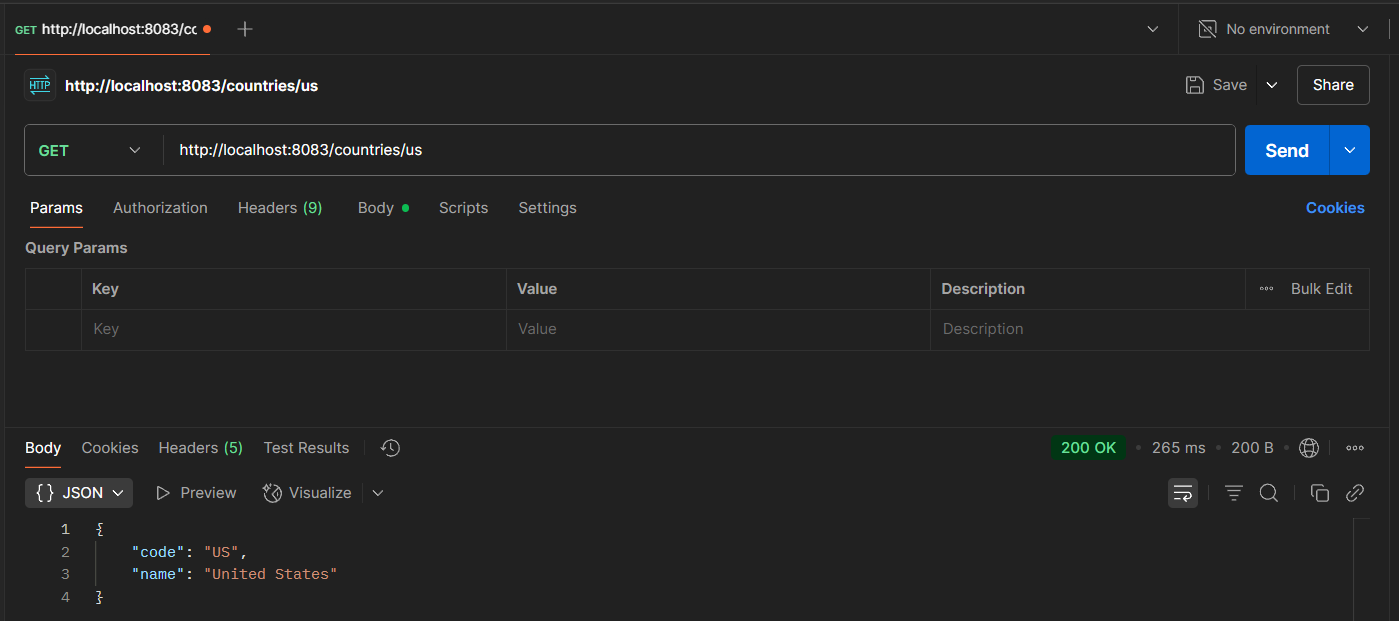
       Country country = countryService.getCountry(code);

***LOGGER***.info("END - getCountry()");

       return country;

   }

}



**REST - Get country exceptional scenario**

package com.cognizant.springlearn.service.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);

   }

}

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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) 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"));

   }

}

package com.cognizant.springlearn.controller;

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

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

*@*GetMapping("/countries/{code}")

public Country getCountry(*@*PathVariable String code) throws CountryNotFoundException {

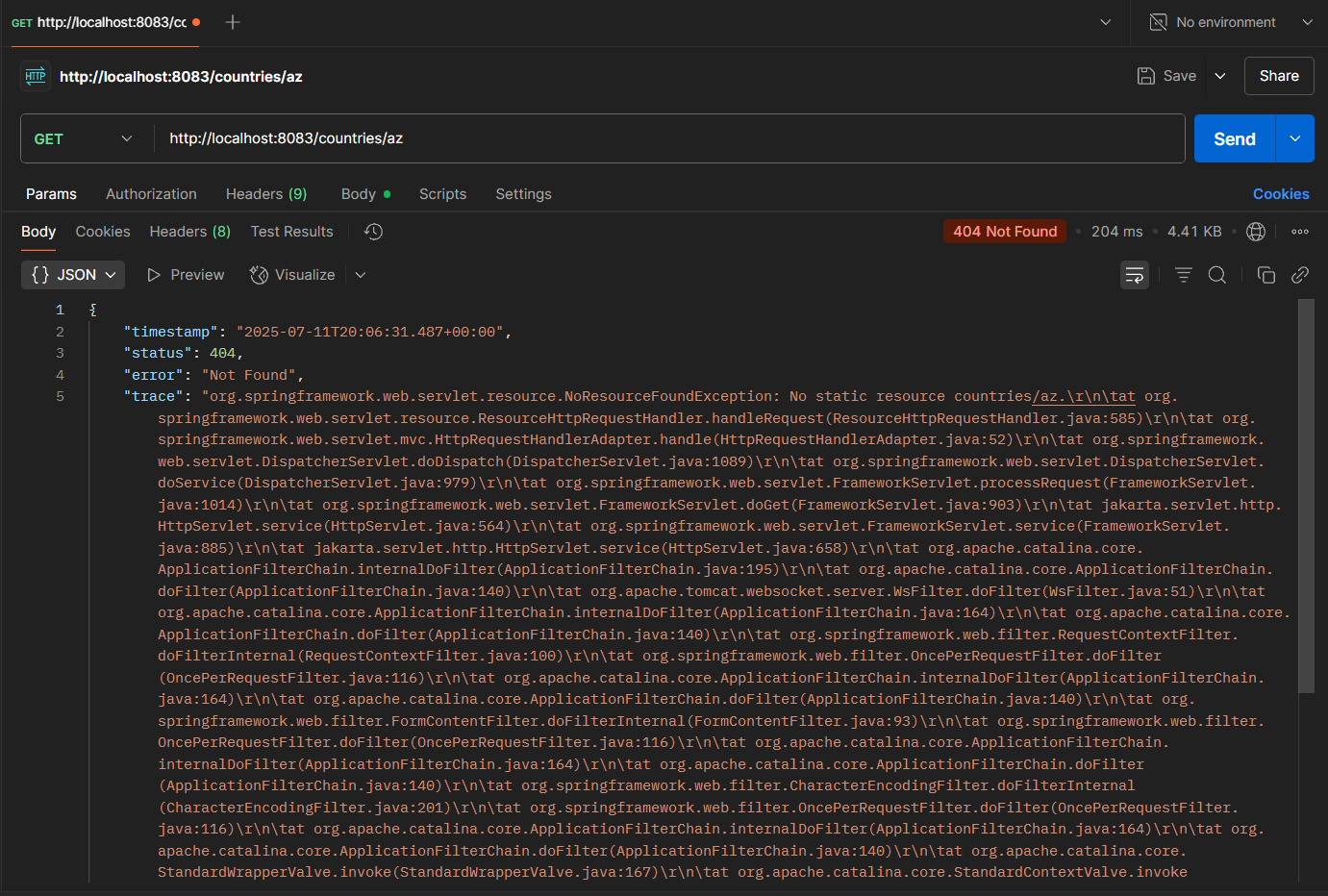
   LOGGER.info("START - getCountry() with code: {}", code);

   Country country = countryService.getCountry(code);

   LOGGER.info("END - getCountry()");

   return country;

}



**MockMVC - Test get country service** 

package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import static org.junit.jupiter.api.Assertions.*assertNotNull*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import org.springframework.test.web.servlet.ResultActions;

*@SpringBootTest*

*@AutoConfigureMockMvc*

public class SpringLearnApplicationTests {

*@Autowired*

   private CountryController countryController;

*@Autowired*

   private MockMvc mvc;

   // ✅ Test if controller is loaded in Spring Context

*@Test*

   public void contextLoads() {

*assertNotNull*(countryController);

   }

   // ✅ Test the /country endpoint

*@Test*

   public void testGetCountry() throws Exception {

       ResultActions actions = mvc.perform(*get*("/country"));

       actions.andExpect(*status*().isOk());

       actions.andExpect(*jsonPath*("$.code").exists());

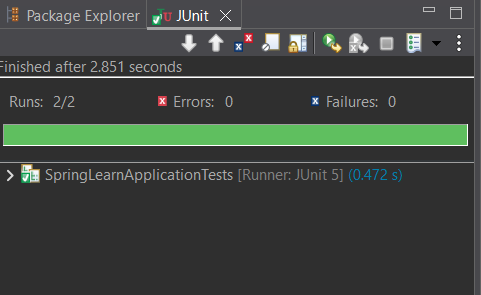
       actions.andExpect(*jsonPath*("$.code").value("IN"));

       actions.andExpect(*jsonPath*("$.name").exists());

       actions.andExpect(*jsonPath*("$.name").value("India"));

   }

}



**MockMVC - Test get country service for exceptional scenario**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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")

   public Country getCountryIndia() {

       // Hardcoded for test purpose

       Country country = new Country();

       country.setCode("IN");

       country.setName("India");

       return country;

   }

*@GetMapping*("/countries/{code}")

   public Country getCountry(*@PathVariable* String code) throws CountryNotFoundException {

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

       Country country = countryService.getCountry(code);

***LOGGER***.info("END - getCountry()");

       return country;

   }

}

package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

import static org.junit.jupiter.api.Assertions.*assertNotNull*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import org.springframework.test.web.servlet.ResultActions;

*@SpringBootTest*

*@AutoConfigureMockMvc*

public class SpringLearnApplicationTests {

*@Autowired*

   private CountryController countryController;

*@Autowired*

   private MockMvc mvc;

   // ✅ Test if the controller is loaded in Spring Context

*@Test*

   public void contextLoads() {

*assertNotNull*(countryController);

   }

   // ✅ Test the /country endpoint (default country)

*@Test*

   public void testGetCountry() throws Exception {

       ResultActions actions = mvc.perform(*get*("/country"));

       actions.andExpect(*status*().isOk());

       actions.andExpect(*jsonPath*("$.code").exists());

       actions.andExpect(*jsonPath*("$.code").value("IN"));

       actions.andExpect(*jsonPath*("$.name").exists());

       actions.andExpect(*jsonPath*("$.name").value("India"));

   }

   // ✅ Test the /countries/{code} endpoint for an invalid country code

*@Test*

   public void testGetCountryException() throws Exception {

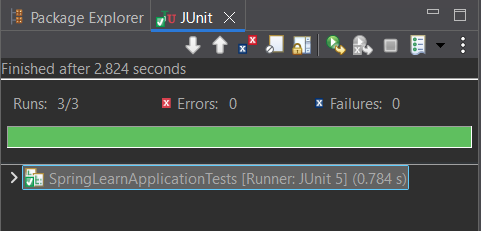
       ResultActions actions = mvc.perform(*get*("/countries/ZZ")); // ZZ = invalid code

       actions.andExpect(*status*().isNotFound()); // 404

       actions.andExpect(*status*().reason("Country not found")); // 👈 match exactly

   }

}



**3. spring-rest-handson**

**Problem Statement - Display Employee List and Edit Employee form using RESTful Web Service** 

<?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="employee1" class="com.example.model.Employee">

       <property name="id" value="1"/>

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

       <property name="designation" value="Developer"/>

   </bean>

   <bean id="employee2" class="com.example.model.Employee">

       <property name="id" value="2"/>

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

       <property name="designation" value="Manager"/>

   </bean>

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

       <constructor-arg>

           <list>

               <ref bean="employee1"/>

               <ref bean="employee2"/>

           </list>

       </constructor-arg>

   </bean>

</beans>

package com.example.model;

public class Employee {

   private int id;

   private String name;

   private String designation;

   public int getId() { return id; }

   public void setId(int id) { this.id = id; }

   public String getName() { return name; }

   public void setName(String name) { this.name = name; }

   public String getDesignation() { return designation; }

   public void setDesignation(String designation) { this.designation = designation; }

}

package com.example.controller;

import com.example.model.Employee;

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

import org.springframework.context.ApplicationContext;

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

import java.util.List;

*@RestController*

*@RequestMapping*("/api/employees")

*@CrossOrigin*(origins = "\*")  // enable CORS for Angular

public class EmployeeController {

*@Autowired*

   private ApplicationContext context;

*@GetMapping*

   public List<Employee> getAllEmployees() {

       return (List<Employee>) context.getBean("employeeList");

   }

*@GetMapping*("/{id}")

   public Employee getEmployeeById(*@PathVariable* int id) {

       List<Employee> employees = (List<Employee>) context.getBean("employeeList");

       return employees.stream()

                       .filter(emp -> emp.getId() == id)

                       .findFirst()

                       .orElse(null);

   }

}

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ImportResource;

*@SpringBootApplication*

*@ImportResource*("classpath:beans.xml")

public class EmployeeApp {

   public static void main(String[] args) {

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

   }

}

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { Employee } from './employee';

@Injectable({

  providedIn: 'root'

})

export class EmployeeService {

  private baseUrl = 'http://localhost:8080/api/employees';

  constructor(private http: HttpClient) {}

  getEmployees(): Observable<Employee[]> {

    return this.http.get<Employee[]>(this.baseUrl);

  }

  getEmployeeById(id: number): Observable<Employee> {

    return this.http.get<Employee>(`${this.baseUrl}/${id}`);

  }

  updateEmployee(employee: Employee): Observable<Employee> {

    return this.http.put<Employee>(`${this.baseUrl}/${employee.id}`, employee);

  }

}

export interface Employee {

  id: number;

  name: string;

  designation: string;

}

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { Employee } from './employee';

@Injectable({

  providedIn: 'root'

})

export class EmployeeService {

  private baseUrl = 'http://localhost:8080/api/employees';

  constructor(private http: HttpClient) {}

  getEmployees(): Observable<Employee[]> {

    return this.http.get<Employee[]>(this.baseUrl);

  }

  getEmployeeById(id: number): Observable<Employee> {

    return this.http.get<Employee>(`${this.baseUrl}/${id}`);

  }

  updateEmployee(employee: Employee): Observable<Employee> {

    return this.http.put<Employee>(`${this.baseUrl}/${employee.id}`, employee);

  }

}

import { Component, OnInit } from '@angular/core';

import { Employee } from '../employee';

import { EmployeeService } from '../employee.service';

import { CommonModule } from '@angular/common';

import { RouterModule } from '@angular/router'; // for routerLink

@Component({

  selector: 'app-employee-list',

  standalone: true,

  imports: [CommonModule,RouterModule],

  templateUrl: './employee-list.component.html',

})

export class EmployeeListComponent implements OnInit {

  employees: Employee[] = [];

  constructor(private employeeService: EmployeeService) {}

  ngOnInit(): void {

    this.employeeService.getEmployees().subscribe(data => {

      this.employees = data;

    });

  }

}

<h2>Employee List</h2>

<table border="1">

  <tr><th>ID</th><th>Name</th><th>Designation</th><th>Action</th></tr>

  <tr \*ngFor="let emp of employees">

    <td>{{ emp.id }}</td>

    <td>{{ emp.name }}</td>

    <td>{{ emp.designation }}</td>

    <td>

      <button [routerLink]="['/edit', emp.id]">Edit</button>

    </td>

  </tr>

</table>

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import { ActivatedRoute } from '@angular/router';

import { EmployeeService } from '../employee.service';

import { Employee } from '../employee';

@Component({

  selector: 'app-employee-edit',

  standalone: true,

  imports: [CommonModule, FormsModule],

  templateUrl: './employee-edit.component.html'

})

export class EmployeeEditComponent implements OnInit {

  employee: Employee = { id: 0, name: '', designation: '' };

  constructor(private route: ActivatedRoute, private service: EmployeeService) {}

  ngOnInit(): void {

    const id = Number(this.route.snapshot.paramMap.get('id'));

    this.service.getEmployeeById(id).subscribe(emp => this.employee = emp);

  }

}

<h2>Edit Employee</h2>

<form \*ngIf="employee">

  <label>ID:</label> {{ employee.id }}<br>

  <label>Name:</label>

  <input [(ngModel)]="employee.name" name="name"><br>

  <label>Designation:</label>

  <input [(ngModel)]="employee.designation" name="designation"><br>

  <button>Save</button>

</form>

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { EmployeeListComponent } from './employee-list/employee-list.component';

import { EmployeeEditComponent } from './employee-edit/employee-edit.component';

const routes: Routes = [

  { path: '', component: EmployeeListComponent },

  { path: 'edit/:id', component: EmployeeEditComponent }

];

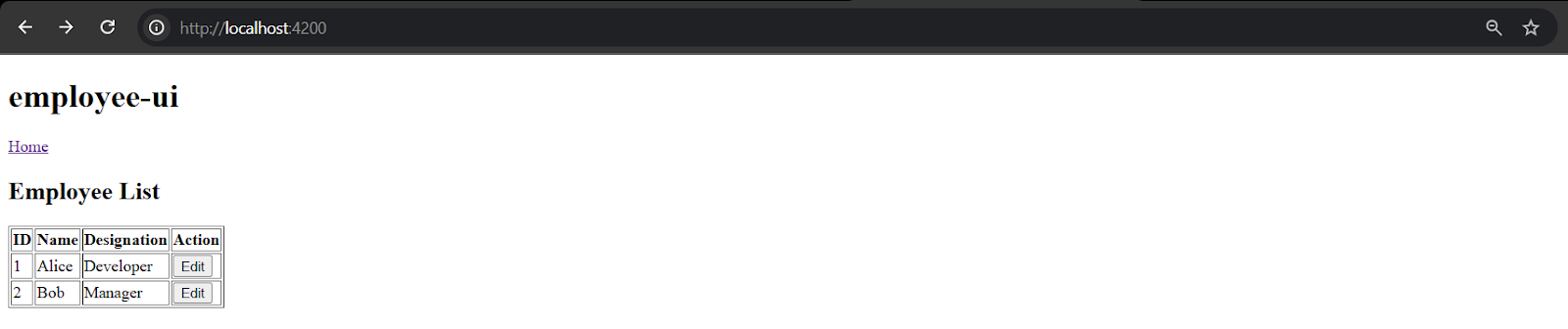
@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule {}



**Create static employee list data using spring xml configuration**

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

<beans xmlns="http://www.springframework.org/schema/beans"

      xmlns:util="http://www.springframework.org/schema/util"

      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

        http://www.springframework.org/schema/util

        http://www.springframework.org/schema/util/spring-util.xsd">

   <!-- Employee Beans -->

   <bean id="emp1" class="com.example.model.Employee">

       <property name="id" value="1" />

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

       <property name="designation" value="Developer" />

   </bean>

   <bean id="emp2" class="com.example.model.Employee">

       <property name="id" value="2" />

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

       <property name="designation" value="Tester" />

   </bean>

   <bean id="emp3" class="com.example.model.Employee">

       <property name="id" value="3" />

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

       <property name="designation" value="Analyst" />

   </bean>

   <bean id="emp4" class="com.example.model.Employee">

       <property name="id" value="4" />

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

       <property name="designation" value="Support" />

   </bean>

   <!-- Employee List -->

   <util:list id="employeeList" value-type="com.example.model.Employee">

       <ref bean="emp1"/>

       <ref bean="emp2"/>

       <ref bean="emp3"/>

       <ref bean="emp4"/>

   </util:list>

</beans>

package com.example.model;

public class Employee {

   private int id;

   private String name;

   private String designation;

   // Getters & setters

   public int getId() { return id; }

   public void setId(int id) { this.id = id; }

   public String getName() { return name; }

   public void setName(String name) { this.name = name; }

   public String getDesignation() { return designation; }

   public void setDesignation(String designation) { this.designation = designation; }

}

package com.example.model;

public class Department {

   private int id;

   private String name;

   public Department(int id, String name) {

       this.id = id;

       this.name = name;

   }

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

package com.example.dao;

import com.example.model.Employee;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

public class EmployeeDao {

   private static List<Employee> *EMPLOYEE\_LIST*;

   public EmployeeDao() {

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

*EMPLOYEE\_LIST* = (List<Employee>) context.getBean("employeeList");

   }

   public List<Employee> getAllEmployees() {

       return *EMPLOYEE\_LIST*;

   }

   public Employee getEmployeeById(int id) {

       return *EMPLOYEE\_LIST*.stream()

               .filter(emp -> emp.getId() == id)

               .findFirst()

               .orElse(null);

   }

}

package com.example.controller;

import com.example.dao.EmployeeDao;

import com.example.model.Employee;

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

import java.util.List;

*@RestController*

*@RequestMapping*("/api/employees")

*@CrossOrigin*(origins = "\*")

public class EmployeeController {

   private final EmployeeDao employeeDao = new EmployeeDao();

*@GetMapping*

   public List<Employee> getAllEmployees() {

       return employeeDao.getAllEmployees();

   }

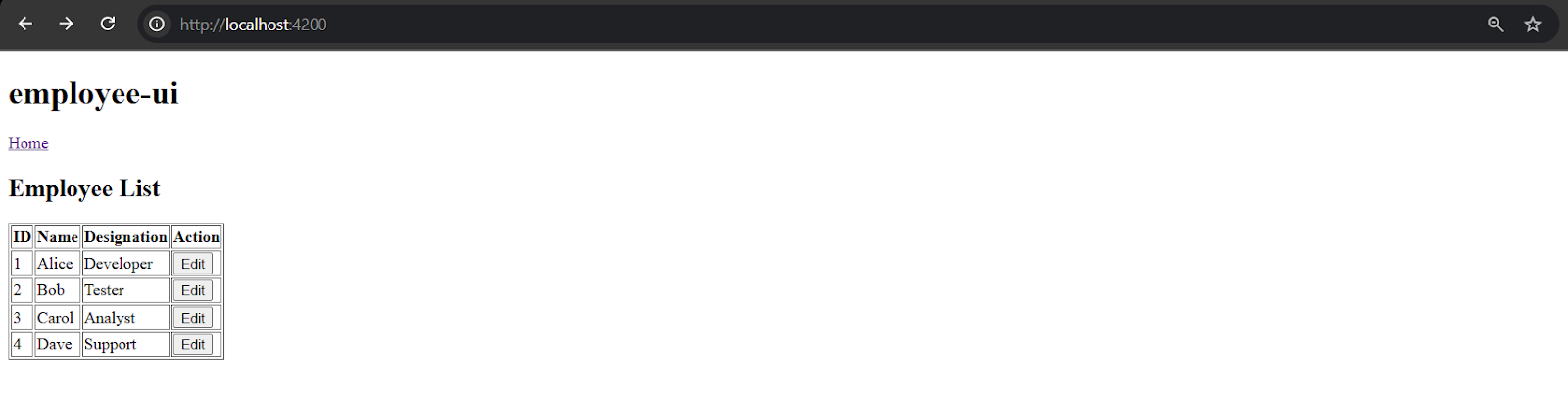
*@GetMapping*("/{id}")

   public Employee getEmployeeById(*@PathVariable* int id) {

       return employeeDao.getEmployeeById(id);

   }

}



**Create REST service to gets all employees** 

package com.example.service;

import com.example.dao.EmployeeDao;

import com.example.model.Employee;

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

import org.springframework.stereotype.Service;

import java.util.List;

*@Service*

public class EmployeeService {

*@Autowired*

   private EmployeeDao employeeDao;

   public List<Employee> getAllEmployees() {

       return employeeDao.getAllEmployees();

   }

}

package com.example.controller;

import com.example.model.Employee;

import com.example.service.EmployeeService;

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

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

import java.util.List;

*@RestController*

*@CrossOrigin*(origins = "\*")

public class EmployeeController {

*@Autowired*

   private EmployeeService employeeService;

*@GetMapping*("/employees")  // <-- Required mapping

   public List<Employee> getAllEmployees() {

       return employeeService.getAllEmployees();

   }

}

package com.example.dao;

import com.example.model.Employee;

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

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public class EmployeeDao {

   private static List<Employee> *EMPLOYEE\_LIST*;

*@Autowired*

   public EmployeeDao(ApplicationContext context) {

*EMPLOYEE\_LIST* = (List<Employee>) context.getBean("employeeList");

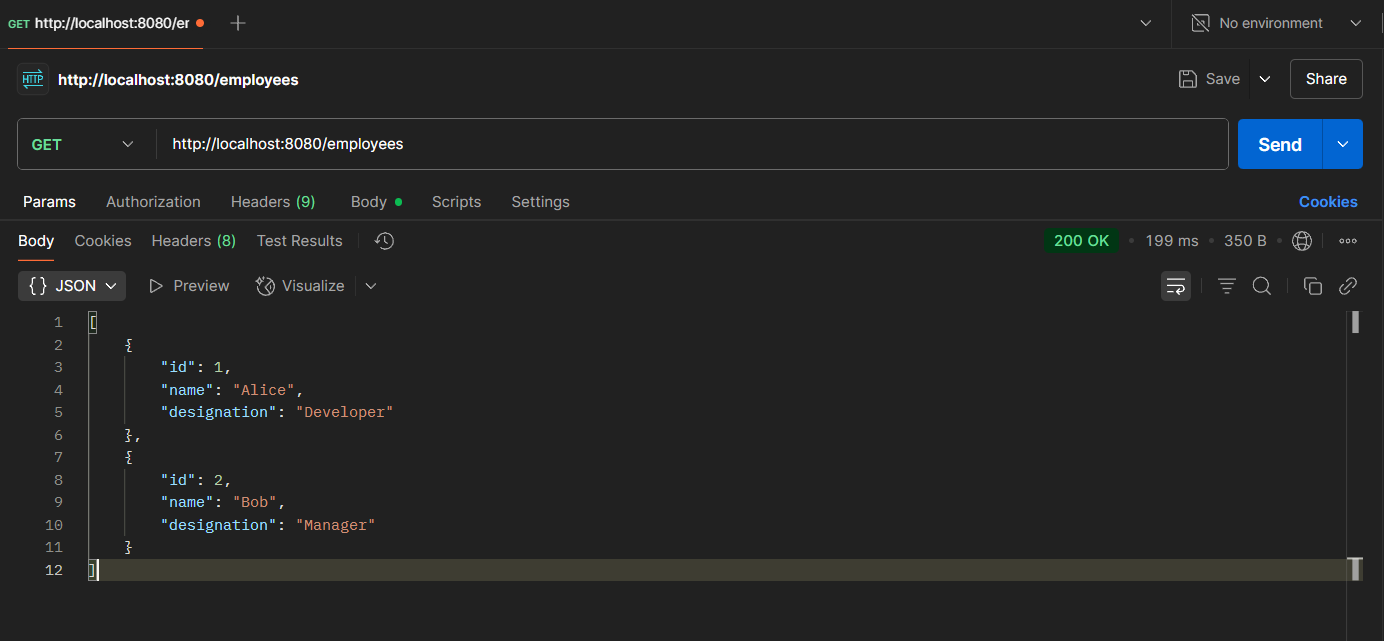
   }

   public List<Employee> getAllEmployees() {

       return *EMPLOYEE\_LIST*;

   }

}



**Create REST service for department**

package com.example.dao;

import com.example.model.Department;

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

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public class DepartmentDao {

   public static List<Department> *DEPARTMENT\_LIST*;

*@Autowired*

   public DepartmentDao(ApplicationContext context) {

*DEPARTMENT\_LIST* = (List<Department>) context.getBean("departmentList");

   }

   public List<Department> getAllDepartments() {

       return *DEPARTMENT\_LIST*;

   }

}

package com.example.service;

import com.example.dao.DepartmentDao;

import com.example.model.Department;

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

import org.springframework.stereotype.Service;

import java.util.List;

*@Service*

public class DepartmentService {

*@Autowired*

   private DepartmentDao departmentDao;

   public List<Department> getAllDepartments() {

       return departmentDao.getAllDepartments();

   }

}

package com.example.controller;

import com.example.model.Department;

import com.example.service.DepartmentService;

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

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

import java.util.List;

*@RestController*

*@RequestMapping*("/api")

*@CrossOrigin*(origins = "\*")

public class DepartmentController {

*@Autowired*

   private DepartmentService departmentService;

*@GetMapping*("/departments")

   public List<Department> getAllDepartments() {

       System.***out***.println("Fetching all departments...");

       return departmentService.getAllDepartments();

   }

}

package com.example.model;

public class Department {

   private int id;

   private String name;

   // Getters and Setters

   public int getId() { return id; }

   public void setId(int id) { this.id = id; }

   public String getName() { return name; }

   public void setName(String name) { this.name = name; }

}

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

<beans xmlns="http://www.springframework.org/schema/beans"

      xmlns:util="http://www.springframework.org/schema/util"

      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

        http://www.springframework.org/schema/util

        http://www.springframework.org/schema/util/spring-util.xsd">

   <!-- Employee Beans -->

   <bean id="emp1" class="com.example.model.Employee">

       <property name="id" value="1" />

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

       <property name="designation" value="Developer" />

   </bean>

   <bean id="emp2" class="com.example.model.Employee">

       <property name="id" value="2" />

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

       <property name="designation" value="Tester" />

   </bean>

   <bean id="emp3" class="com.example.model.Employee">

       <property name="id" value="3" />

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

       <property name="designation" value="Analyst" />

   </bean>

   <bean id="emp4" class="com.example.model.Employee">

       <property name="id" value="4" />

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

       <property name="designation" value="Support" />

   </bean>

   <!-- Employee List -->

   <util:list id="employeeList" value-type="com.example.model.Employee">

       <ref bean="emp1"/>

       <ref bean="emp2"/>

       <ref bean="emp3"/>

       <ref bean="emp4"/>

   </util:list>

   <!-- Department Beans -->

   <bean id="dept1" class="com.example.model.Department">

       <property name="id" value="101" />

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

   </bean>

   <bean id="dept2" class="com.example.model.Department">

       <property name="id" value="102" />

       <property name="name" value="Human Resources" />

   </bean>

   <!-- Department List -->

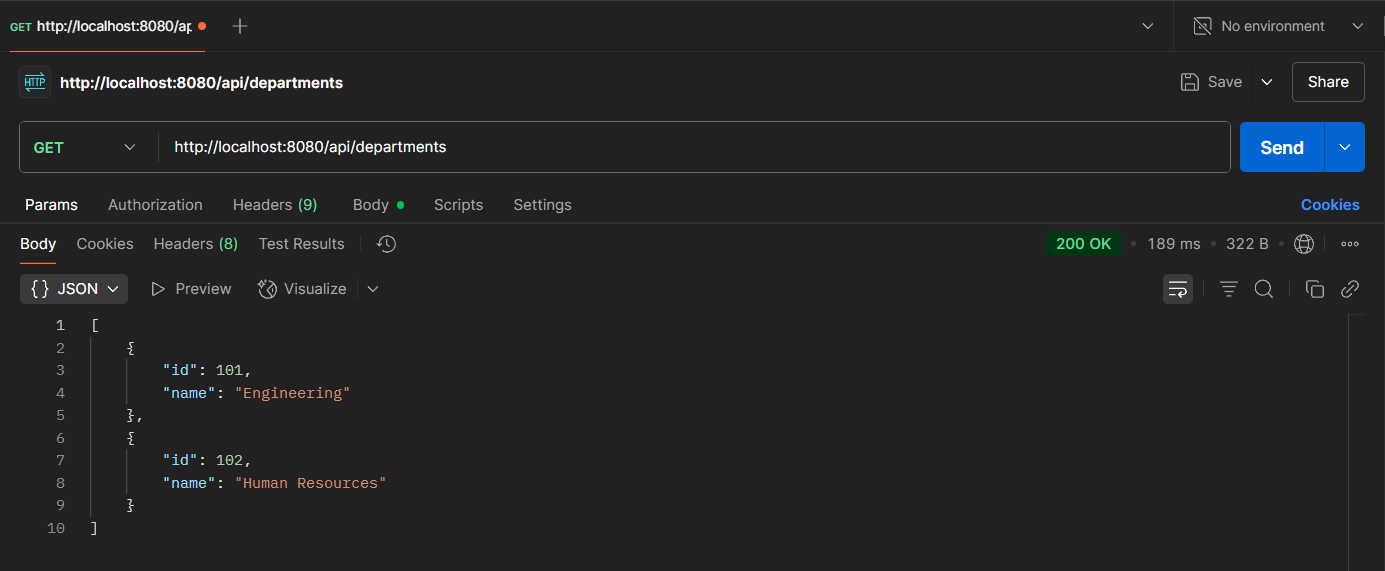
   <util:list id="departmentList" value-type="com.example.model.Department">

       <ref bean="dept1"/>

       <ref bean="dept2"/>

   </util:list>

</beans>



**5. JWT-handson**

**Securing RESTful Web Services with Spring Security**

**<?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>**

**<url/>**

**<licenses>**

**<license/>**

**</licenses>**

**<developers>**

**<developer/>**

**</developers>**

**<scm>**

**<connection/>**

**<developerConnection/>**

**<tag/>**

**<url/>**

**</scm>**

**<properties>**

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

**</properties>**

**<dependencies>**

**<dependency>**

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

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

**</dependency>**

**<dependency>**

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

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

**<scope>runtime</scope>**

**<optional>true</optional>**

**</dependency>**

**<dependency>**

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

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

**<scope>test</scope>**

**</dependency>**

**<dependency>**

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

**<artifactId>spring-boot-starter-security</artifactId>**

**</dependency>**

**</dependencies>**

**<build>**

**<plugins>**

**<plugin>**

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

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

**</plugin>**

**</plugins>**

**</build>**

**</project>**

**package com.cognizant.springlearn.security;**

**import org.springframework.context.annotation.Bean;**

**import org.springframework.context.annotation.Configuration;**

**import org.springframework.security.config.Customizer;**

**import org.springframework.security.config.annotation.web.builders.HttpSecurity;**

**import org.springframework.security.web.SecurityFilterChain;**

***@Configuration***

**public class SecurityConfig {**

***@Bean***

**public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {**

**http**

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

**.authorizeHttpRequests(auth -> auth.anyRequest().authenticated())**

**.httpBasic(Customizer.*withDefaults*());**

**return http.build();**

**}**

**}**

****

**Creating users and roles in Spring Security**

**package com.cognizant.springlearn.security;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

**import org.springframework.context.annotation.Bean;**

**import org.springframework.context.annotation.Configuration;**

**import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;**

**import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;**

**import org.springframework.security.config.annotation.web.builders.HttpSecurity;**

**import org.springframework.security.core.userdetails.User;**

**import org.springframework.security.core.userdetails.UserDetailsService;**

**import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;**

**import org.springframework.security.crypto.password.PasswordEncoder;**

**import org.springframework.security.provisioning.InMemoryUserDetailsManager;**

**import org.springframework.security.web.SecurityFilterChain;**

***@Configuration***

***@EnableMethodSecurity***

**public class SecurityConfig {**

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

***@Bean***

**public PasswordEncoder passwordEncoder() {**

***LOGGER*.info("Start passwordEncoder()");**

**return new BCryptPasswordEncoder();**

**}**

***@Bean***

**public UserDetailsService userDetailsService(PasswordEncoder encoder) {**

***LOGGER*.info("Start userDetailsService()");**

**InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();**

**manager.createUser(**

**User.*withUsername*("admin")**

**.password(encoder.encode("pwd"))**

**.roles("ADMIN")**

**.build()**

**);**

**manager.createUser(**

**User.*withUsername*("user")**

**.password(encoder.encode("pwd"))**

**.roles("USER")**

**.build()**

**);**

**return manager;**

**}**

***@Bean***

**public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {**

**http**

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

**.authorizeHttpRequests(auth ->**

**auth**

**.requestMatchers("/countries").hasRole("USER")**

**.anyRequest().authenticated()**

**)**

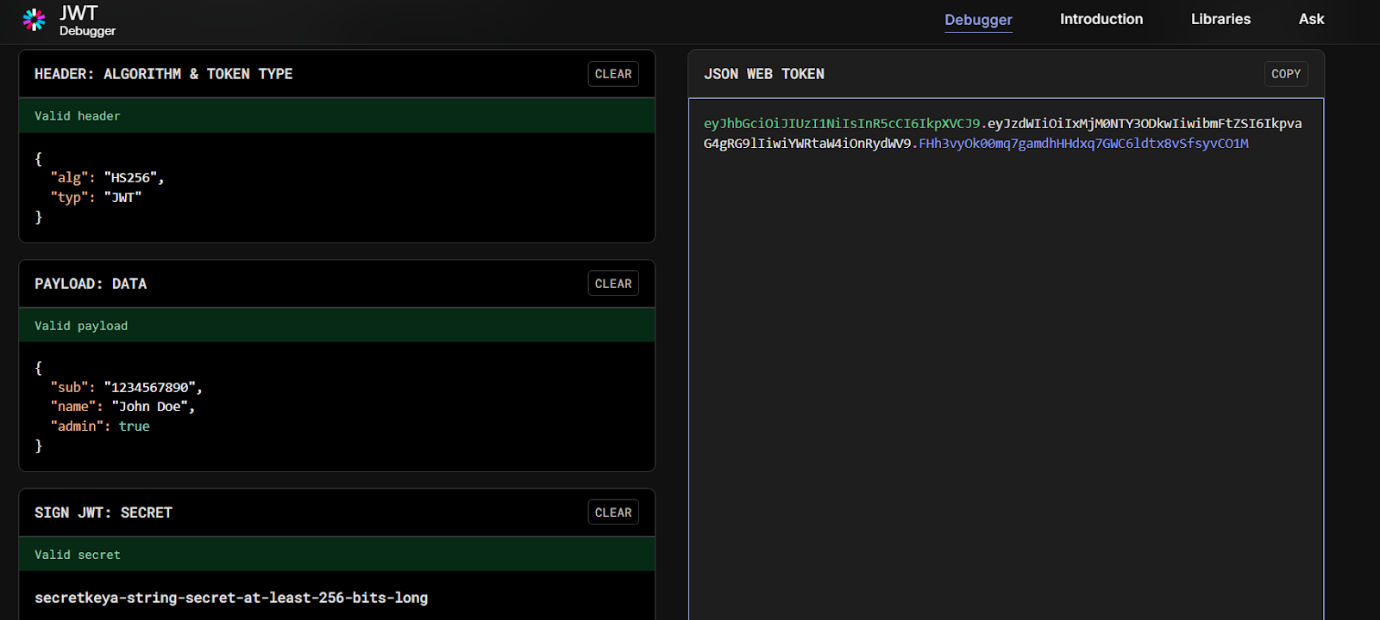
**.~~httpBasic~~();**

**return http.build();**

**}**

**}**

**Understanding JWT**



**Create authentication service that returns JWT**

package com.cognizant.springlearn.controller;

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

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import com.cognizant.springlearn.security.JwtUtil;

*@RestController*

public class AuthController {

*@Autowired*

   private AuthenticationManager authenticationManager;

*@Autowired*

   private JwtUtil jwtUtil;

*@RequestMapping*(value = "/authenticate", method = *RequestMethod*.***GET***)

   public ResponseEntity<?> generateToken(*@RequestHeader*("Authorization") String authHeader) {

       String[] credentials = extractCredentials(authHeader);

       String username = credentials[0];

       String password = credentials[1];

       Authentication authentication = authenticationManager.authenticate(

               new UsernamePasswordAuthenticationToken(username, password));

       String token = jwtUtil.generateToken(authentication);

       return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");

   }

   private String[] extractCredentials(String authHeader) {

       String base64Credentials = authHeader.substring("Basic".length()).trim();

       byte[] credDecoded = java.util.Base64.*getDecoder*().decode(base64Credentials);

       String credentials = new String(credDecoded);

       return credentials.split(":", 2);

   }

}

package com.cognizant.springlearn.security;

import java.util.Date;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.\*;

*@Component*

public class JwtUtil {

   private static final String ***SECRET\_KEY*** = "secretkeya-string-secret-at-least-256-bits-long";

   private static final long ***EXPIRATION\_TIME*** = 1000 \* 60 \* 10; // 10 minutes

   public String generateToken(Authentication authentication) {

       return Jwts.builder()

               .setSubject(authentication.getName())

               .setIssuedAt(new Date())

               .setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

               .signWith(SignatureAlgorithm.HS256, ***SECRET\_KEY***)

               .compact();

   }

}

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableMethodSecurity*

public class SecurityConfig {

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

*@Bean*

   public PasswordEncoder passwordEncoder() {

***LOGGER***.info("Start passwordEncoder()");

       return new BCryptPasswordEncoder();

   }

*@Bean*

   public UserDetailsService userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Start userDetailsService()");

       InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

       manager.createUser(

           User.*withUsername*("admin")

               .password(encoder.encode("pwd"))

               .roles("ADMIN")

               .build()

       );

       manager.createUser(

           User.*withUsername*("user")

               .password(encoder.encode("pwd"))

               .roles("USER")

               .build()

       );

       return manager;

   }

*@Bean*

   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

       http

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

           .authorizeHttpRequests(auth ->

               auth

                   .requestMatchers("/countries").hasRole("USER")

                   .anyRequest().authenticated()

           )

           .~~httpBasic~~();

       return http.build();

   }

*@Bean*

   public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {

       return authConfig.getAuthenticationManager();

   }

}

**Create authentication controller and configure it in SecurityConfig** 

package com.cognizant.springlearn.controller;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

   public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

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

***LOGGER***.debug("Authorization Header: {}", authHeader);

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

       map.put("token", "");  // Placeholder for now

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

       return map;

   }

}

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableMethodSecurity*

public class SecurityConfig {

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

*@Bean*

   public PasswordEncoder passwordEncoder() {

***LOGGER***.info("Start passwordEncoder()");

       return new BCryptPasswordEncoder();

   }

*@Bean*

   public UserDetailsService userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Start userDetailsService()");

       InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

       manager.createUser(

           User.*withUsername*("admin")

               .password(encoder.encode("pwd"))

               .roles("ADMIN")

               .build()

       );

       manager.createUser(

           User.*withUsername*("user")

               .password(encoder.encode("pwd"))

               .roles("USER")

               .build()

       );

       return manager;

   }

*@Bean*

   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

       http

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

           .authorizeHttpRequests(auth ->

               auth

                   .requestMatchers("/countries").hasRole("USER")

                   .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

                   .anyRequest().authenticated()

           )

           .~~httpBasic~~();

       return http.build();

   }

*@Bean*

   public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {

       return authConfig.getAuthenticationManager();

   }

}

**Read Authorization header and decode the username and password**

package com.cognizant.springlearn.controller;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

   public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

***LOGGER***.info("START: /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

       String user = getUser(authHeader);

***LOGGER***.debug("Extracted User: {}", user);

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

       map.put("token", ""); // we’ll add JWT token logic later if needed

***LOGGER***.info("END: /authenticate");

       return map;

   }

   private String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header");

       // Remove "Basic " prefix

       String encodedCredentials = authHeader.substring("Basic ".length());

       // Decode base64 to username:password

       byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);

       String decodedString = new String(decodedBytes);

***LOGGER***.debug("Decoded string: {}", decodedString);

       // Extract username (before ':')

       String username = decodedString.split(":")[0];

       return username;

   }

}

**Generate token based on the user**

<?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>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

   <groupId>io.jsonwebtoken</groupId>

   <artifactId>jjwt</artifactId>

   <version>0.9.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

package com.cognizant.springlearn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

   public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

***LOGGER***.info("START: /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

       String user = getUser(authHeader);

***LOGGER***.debug("Extracted User: {}", user);

       String token = generateJwt(user);

***LOGGER***.debug("Generated Token: {}", token);

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

       map.put("token", token);

***LOGGER***.info("END: /authenticate");

       return map;

   }

   private String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header");

       String encodedCredentials = authHeader.substring("Basic ".length());

       byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);

       String decodedString = new String(decodedBytes);

***LOGGER***.debug("Decoded string: {}", decodedString);

       String username = decodedString.split(":")[0];

       return username;

   }

   private String generateJwt(String user) {

***LOGGER***.debug("Generating JWT token for user: {}", user);

       JwtBuilder builder = Jwts.*builder*();

       builder.setSubject(user);

       builder.setIssuedAt(new Date());

       builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 mins

       builder.signWith(*SignatureAlgorithm*.***HS256***, "secretkey");

       String token = builder.compact();

       return token;

   }

}

**Authorize based on JWT**

package com.cognizant.springlearn.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

   public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

       super(authenticationManager);

***LOGGER***.info("Start JwtAuthorizationFilter");

***LOGGER***.debug("AuthManager: {}", authenticationManager);

   }

*@Override*

   protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,

                                   FilterChain chain) throws IOException, ServletException {

***LOGGER***.info("Start doFilterInternal");

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

***LOGGER***.debug("Authorization Header: {}", header);

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

           chain.doFilter(req, res);

           return;

       }

       UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

       SecurityContextHolder.*getContext*().setAuthentication(authentication);

       chain.doFilter(req, res);

***LOGGER***.info("End doFilterInternal");

   }

   private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

       if (token != null) {

           try {

               Jws<Claims> jws = Jwts.*parser*()

                       .setSigningKey("secretkey")

                       .parseClaimsJws(token.replace("Bearer ", ""));

               String user = jws.getBody().getSubject();

***LOGGER***.debug("User from token: {}", user);

               if (user != null) {

                   return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

               }

           } catch (JwtException ex) {

***LOGGER***.error("Invalid token: {}", ex.getMessage());

           }

       }

       return null;

   }

}

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

*@Bean*

   public SecurityFilterChain filterChain(HttpSecurity http, AuthenticationConfiguration authConfig) throws Exception {

       AuthenticationManager authenticationManager = authConfig.getAuthenticationManager();

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

           .~~httpBasic~~()

           .~~and~~()

           .authorizeHttpRequests(authz -> authz

               .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

               .anyRequest().authenticated()

           )

           .addFilter(new JwtAuthorizationFilter(authenticationManager));

       return http.build();

   }

}