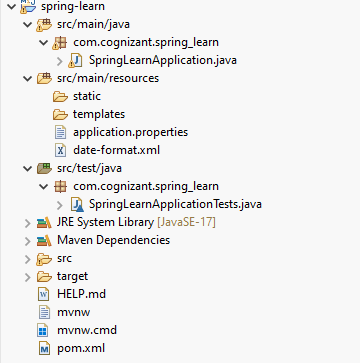
**Docx 1**

**Hands on 1 Create a Spring Web Project using Maven**



**Hands on 2 Spring Core – Load SimpleDateFormat from Spring Configuration XML**

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

**public** **class** SpringLearnApplication {

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

**public** **static** **void** main(String[] args) {

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

SpringLearnApplication app = **new** SpringLearnApplication();

app.displayDate();

}

**public** **void** displayDate() {

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** (ParseException e) {

e.printStackTrace();

}

}

}

**Application.properties**

spring.application.name=spring-learn

logging.level.org.springframework=info

logging.level.com.cognizant.springlearn=debug

**Date-format.xml**

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

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://springframework.org/schema/beans

<http://springframework.org/schema/beans/spring-beans.xsd>">

<beans id="dataFormat" class="java.text.SimpleDataFormat">

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

</beans>

</beans>

**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>

<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 3 Spring Core - Incorporate Logging**

**Application.properties**

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

**SpringLearnApplication.java**

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

**public** **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** (ParseException e) {

e.printStackTrace();

}

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

}

**Date-format.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>">

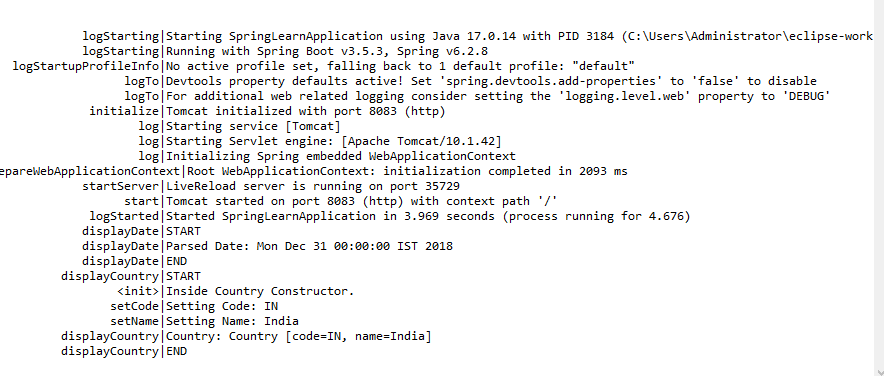
<bean id="dateFormat" class="java.text.SimpleDateFormat">

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

</bean>

</beans>

**Output:**



**Hands on 4 Spring Core – Load Country from Spring Configuration XML**

**Country.java**

**package** com.cognizant.spring\_learn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** Country {

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

**private** String code;

**private** String name;

**public** Country() {

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

}

**public** String getCode() {

***LOGGER***.debug("Getting Code: {}", code);

**return** code;

}

**public** **void** setCode(String code) {

***LOGGER***.debug("Setting Code: {}", code);

**this**.code = code;

}

**public** String getName() {

***LOGGER***.debug("Getting Name: {}", name);

**return** name;

}

**public** **void** setName(String name) {

***LOGGER***.debug("Setting Name: {}", name);

**this**.name = name;

}

@Override

**public** String toString() {

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

}

}

**Country.xml**

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

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<bean id="country" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

</beans>

**SpringLearnApplication.java**

App.displayCountry()

**public** **void** displayCountry() {

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

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

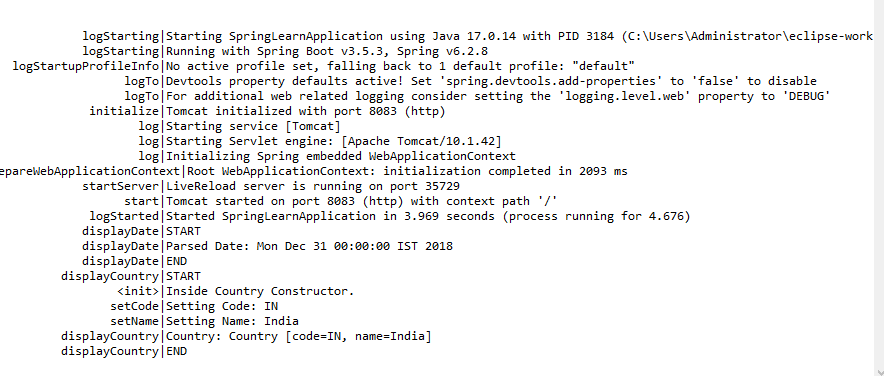
Country country = context.getBean("country", Country.**class**);

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

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

}

**Output**



**Hands on 5 Spring Core – Demonstration of Singleton Scope and Prototype Scope**

**Country.xml**

<bean id="country-prototype" class="com.cognizant.spring\_learn.Country" scope="prototype">

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

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

</bean>

**SpringLearnApplication.java**

App.displayBeanScopes()

**public** **void** displayBeanScopes() {

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

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

Country c1 = context.getBean("country", Country.**class**);

Country c2 = context.getBean("country", Country.**class**);

***LOGGER***.debug("Singleton Scope: c1 == c2? {}", c1 == c2);

Country p1 = context.getBean("country-prototype", Country.**class**);

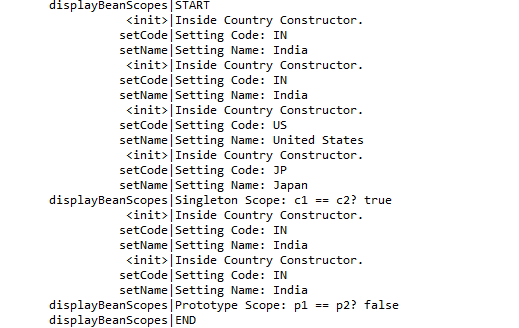
Country p2 = context.getBean("country-prototype", Country.**class**);

***LOGGER***.debug("Prototype Scope: p1 == p2? {}", p1 == p2);

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

}

**Output**



**Hands on 6 Spring Core – Load list of countries from Spring Configuration XML**

**Country.xml**

<bean id="in" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="us" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

</list>

</constructor-arg>

</bean>

**SpringLearnApplication.java**

App.DisplayCountries();

**public** **void** displayCountries() {

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

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

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

**for** (Country c : countries) {

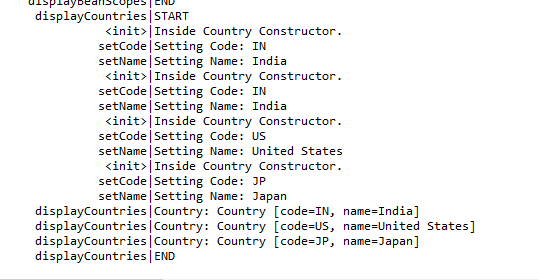
***LOGGER***.debug("Country: {}", c);

}

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

}

**Output**



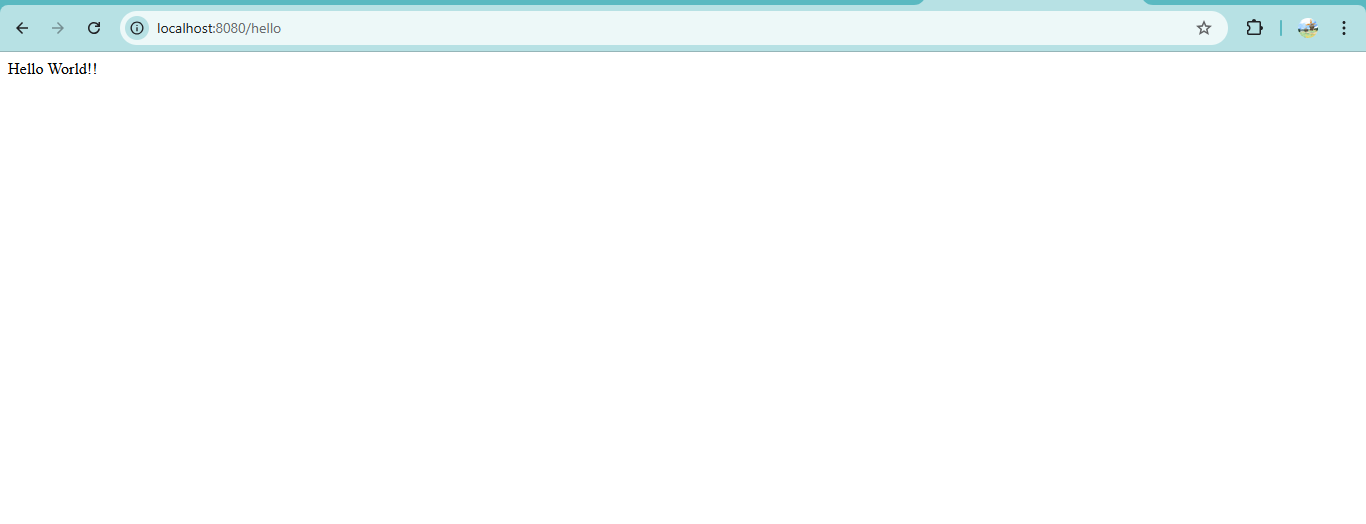
**Docx 2**

**HTTP Request Response**

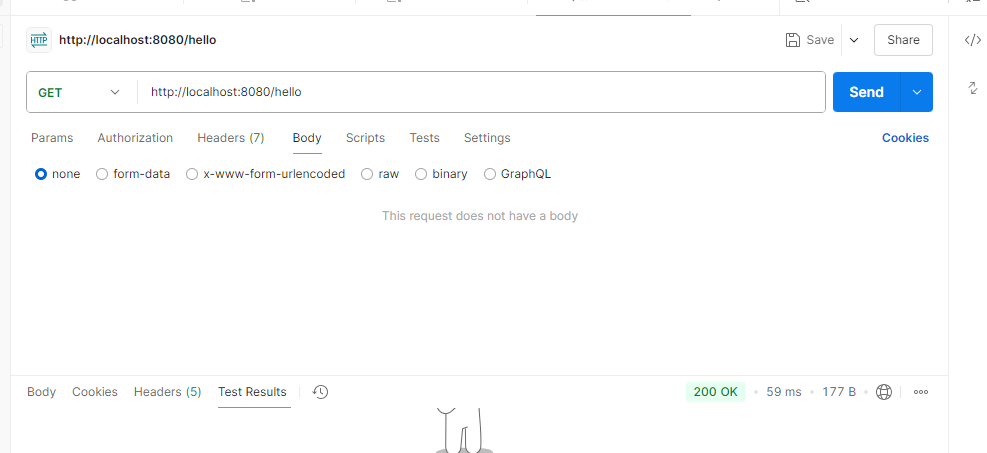
**GET Request**

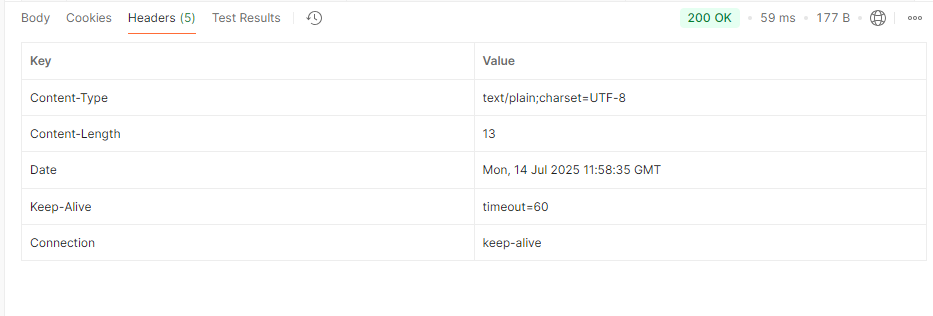
http://localhost:8080/hello

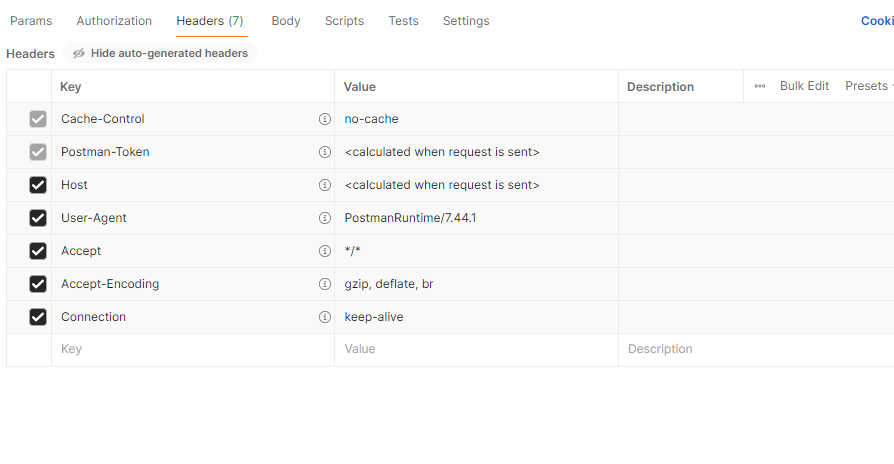
**Output**



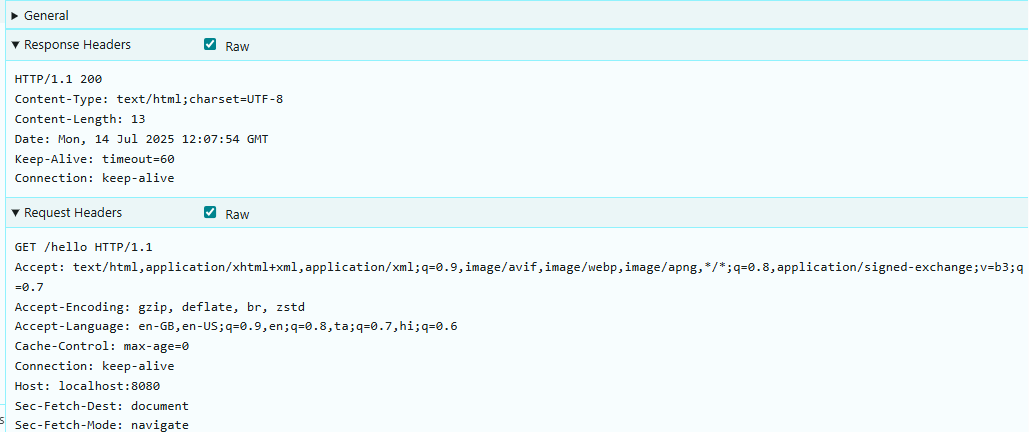
**Postman Output**

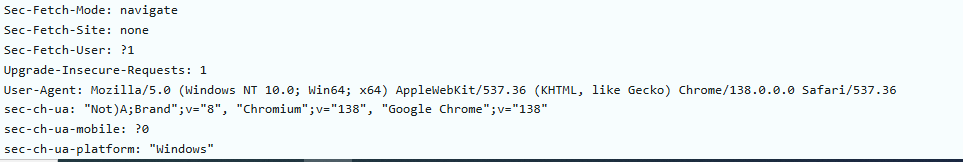


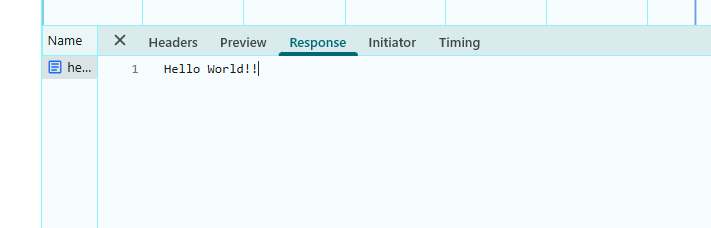




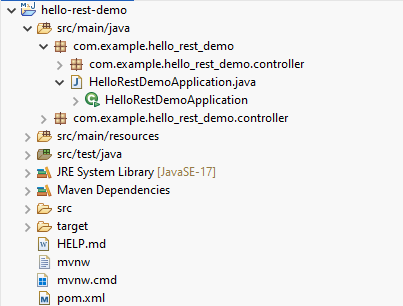
**Chrome Dev Tools – Network Tab**







**Hello World RESTful Web Service**



**HelloCotroller.java**

**package** com.example.hello\_rest\_demo.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()");

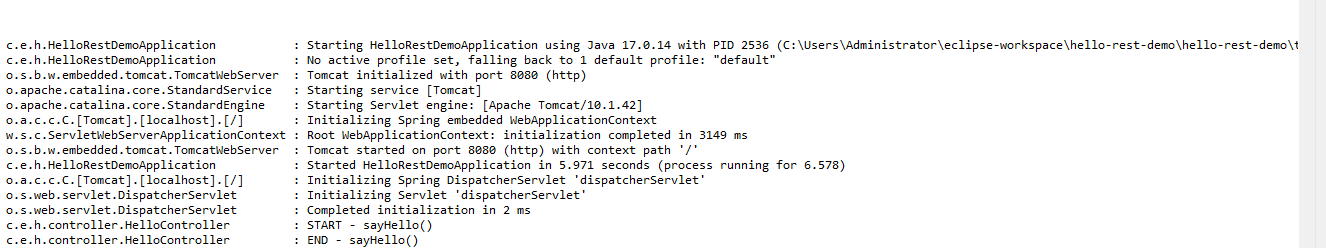
***LOGGER***.info("END - sayHello()");

**return** "Hello World!!";

}

}

**Output**



**REST - Country Web Service**

**Country.java**

**package** com.example.hello\_rest\_demo.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;

}

}

**Country.xml**

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

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<bean id="in" class="com.example.hello\_rest\_demo.model.Country">

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

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

</bean>

</beans>

**HelloRestDemoApplication.java**

**package** com.example.hello\_rest\_demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@ImportResource("classpath:country.xml")

**public** **class** HelloRestDemoApplication {

**public** **static** **void** main(String[] args) {

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

}

}

**CountryController.java**

**package** com.example.hello\_rest\_demo.controller;

**import** com.example.hello\_rest\_demo.model.Country;

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

**import** org.springframework.beans.factory.annotation.Qualifier;

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

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

@RestController

**public** **class** CountryController {

@Autowired

@Qualifier("in")

**private** Country country;

@GetMapping("/country")

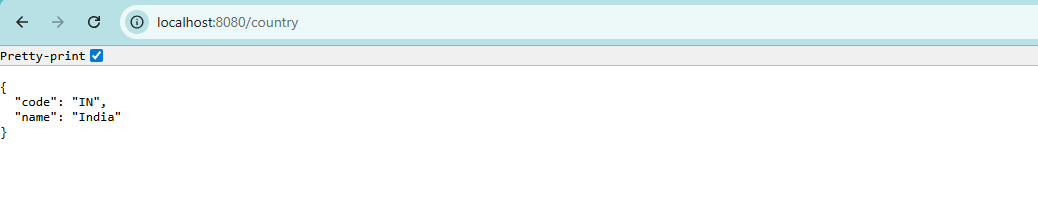
**public** Country getCountryIndia() {

**return** country;

}

}

**Output**



**REST - Get all countries**

**Country.xml**

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

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<bean id="in" class="com.example.hello\_rest\_demo.model.Country">

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

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

</bean>

<bean id="us" class="com.example.hello\_rest\_demo.model.Country">

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

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

</bean>

<bean id="jp" class="com.example.hello\_rest\_demo.model.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

</list>

</constructor-arg>

</bean>

</beans>

**CountryController.java**

@Autowired

@Qualifier("countryList")

**private** List<Country> countryList;

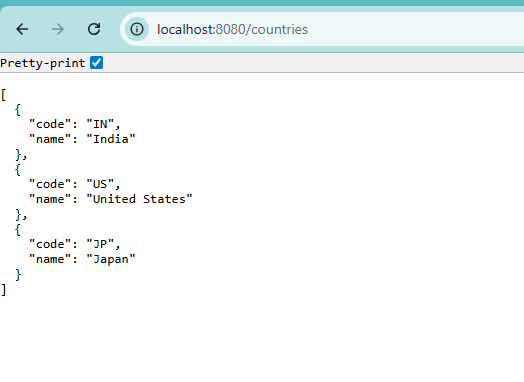
@GetMapping("/countries")

**public** List<Country> getAllCountries() {

**return** countryList;

}

**Output**



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

**CountryService.java**

**package** com.example.hello\_rest\_demo.service;

**import** com.example.hello\_rest\_demo.model.Country;

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

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** CountryService {

@Autowired

@Qualifier("countryList")

**private** List<Country> countryList;

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

**for** (Country country : countryList) {

**if** (country.getCode().equalsIgnoreCase(code)) {

**return** country;

}

}

**return** **null**; // exception

}

}

**CountryController.java**

@Autowired

**private** CountryService countryService;

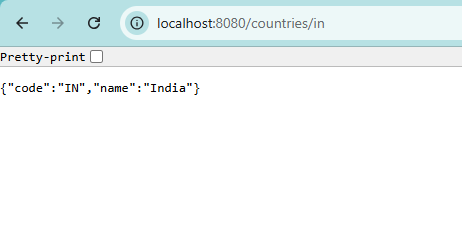
@GetMapping("/countries/{code}")

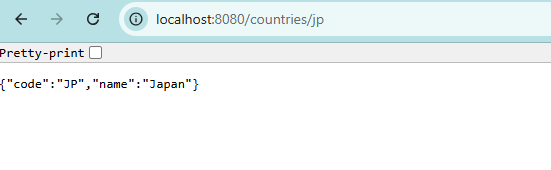
**public** Country getCountryByCode(@PathVariable String code) {

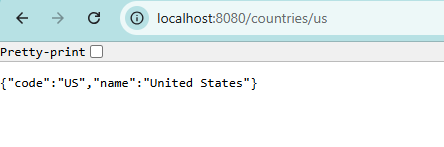
**return** countryService.getCountry(code);

}

**Output**







**REST - Get country exceptional scenari MockMVC - Test get country service**

**CountryNotFoundException.java**

**package** com.example.hello\_rest\_demo.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** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

}

**CountryService.java**

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

**for** (Country country : countryList) {

**if** (country.getCode().equalsIgnoreCase(code)) {

**return** country;

}

}

**throw** **new** CountryNotFoundException();

}

**CountryController.java**

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

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

**return** countryService.getCountry(code);

}

**Output**



**MockMVC - Test get country service**

**Pom.xml**

<dependency>

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

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

<scope>test</scope>

</dependency>

**HelloRestDemoApplicationTests.java**

**package** com.example.hello\_rest\_demo;

**import** com.example.hello\_rest\_demo.controller.CountryController;

**import** org.junit.jupiter.api.Test;

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

**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;

@SpringBootTest

@AutoConfigureMockMvc

**public** **class** HelloRestDemoApplicationTests {

@Autowired

**private** CountryController countryController;

@Autowired

**private** MockMvc mockMvc;

@Test

**void** contextLoads() {

assertNotNull(countryController);

}

@Test

**void** testGetCountryIndia() **throws** Exception {

mockMvc.perform(org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get("/country"))

.andExpect(org.springframework.test.web.servlet.result.MockMvcResultMatchers.status().isOk())

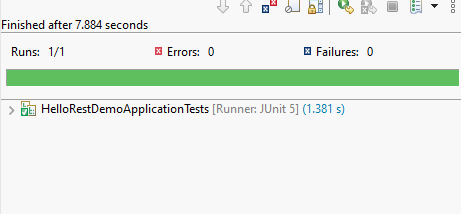
.andExpect(org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath("$.code").value("IN"))

.andExpect(org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath("$.name").value("India"));

}

}

**Output**



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

**HelloRestDemoApplicationTests.java**

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

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

@Test

**void** testGetCountryException() **throws** Exception {

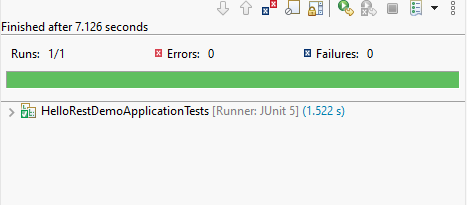
mockMvc.perform(get("/countries/xyz"))

.andExpect(status().isNotFound())

.andExpect(status().reason("Country not found"));

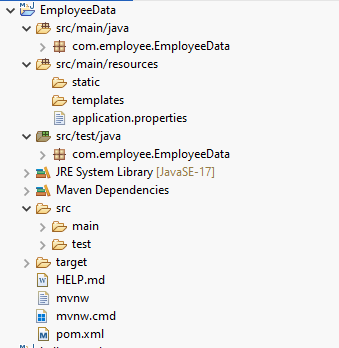
}

**Output**



**Docx 3**

**Problem Statement - Display Employee List and Edit Employee form using 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.employee</groupId>

<artifactId>EmployeeData</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>EmployeeData</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-data-jpa</artifactId>

</dependency>

<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.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Employee.xml**

<?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>

<https://www.springframework.org/schema/beans/spring-beans.xsd>

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

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

<!-- Departments -->

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

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

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

</bean>

<!-- Employees -->

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

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

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

<property name="department" ref="dept1"/>

</bean>

<!-- Employee List -->

<util:list id="employeeList">

<ref bean="emp1"/>

<!-- Add more employees -->

</util:list>

</beans>

**Employee.java**

**package** com.employee.EmployeeData.model;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** Department department;

// getters and setters

}

**Department.java**

**package** com.employee.EmployeeData.model;

**public** **class** Department {

**private** **int** id;

**private** String name;

// getters and setters

}

**EmployeeDataApplication.java**

**package** com.employee.EmployeeData;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@ImportResource("classpath:employee.xml")

**public** **class** EmployeeDataApplication {

**public** **static** **void** main(String[] args) {

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

}

}

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

**Employee.java**

**package** com.employee.EmployeeData.model;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** Department department;

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setDepartment(Department department) {

**this**.department = department;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** Department getDepartment() {

**return** department;

}

}

**Department.java**

**package** com.employee.EmployeeData.dao;

**import** com.employee.EmployeeData.model.Employee;

**import** java.util.List;

**public** **class** EmployeeDao {

**private** List<Employee> employeeList;

**public** EmployeeDao(List<Employee> employeeList) {

**this**.employeeList = employeeList;

}

**public** List<Employee> getAllEmployees() {

**return** employeeList;

}

}

**EmployeeDao.java**

**package** com.employee.EmployeeData.dao;

**import** com.employee.EmployeeData.model.Employee;

**import** java.util.List;

**public** **class** EmployeeDao {

**private** List<Employee> employeeList;

**public** EmployeeDao(List<Employee> employeeList) {

**this**.employeeList = employeeList;

}

**public** List<Employee> getAllEmployees() {

**return** employeeList;

}

}

**EmployeeService.java**

**package** com.employee.EmployeeData.service;

**import** com.employee.EmployeeData.dao.EmployeeDao;

**import** com.employee.EmployeeData.model.Employee;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** java.util.List;

@Service

**public** **class** EmployeeService {

**private** **final** EmployeeDao employeeDao;

**public** EmployeeService(EmployeeDao employeeDao) {

**this**.employeeDao = employeeDao;

}

@Transactional

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAllEmployees();

}

}

**EmployeeController.java**

**package** com.employee.EmployeeData.controller;

**import** com.employee.EmployeeData.model.Employee;

**import** com.employee.EmployeeData.service.EmployeeService;

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

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

**import** java.util.List;

@RestController

**public** **class** EmployeeController {

**private** **final** EmployeeService employeeService;

**public** EmployeeController(EmployeeService employeeService) {

**this**.employeeService = employeeService;

}

@GetMapping("/employees")

**public** List<Employee> getAllEmployees() {

**return** employeeService.getAllEmployees();

}

}

**Employee.xml**

<?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>

<https://www.springframework.org/schema/beans/spring-beans.xsd>

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

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

<!-- Department -->

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

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

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

</bean>

<!-- Employee -->

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

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

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

<property name="department" ref="dept1"/>

</bean>

<!-- Define this list bean -->

<util:list id="employeeList">

<ref bean="emp1"/>

</util:list>

<!-- EmployeeDao uses this list -->

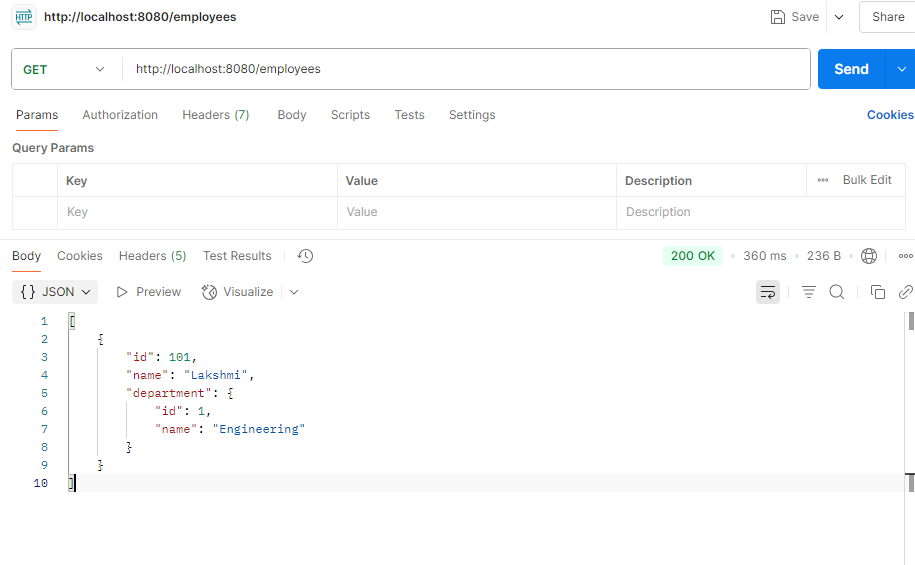
<bean id="employeeDao" class="com.employee.EmployeeData.dao.EmployeeDao">

<constructor-arg ref="employeeList"/>

</bean>

</beans>

**Postman Output**



**Create REST service to gets all employees**

**EmployeeDataApplication.java**

**package** com.employee.EmployeeData;

**import** com.employee.EmployeeData.dao.EmployeeDao;

**import** com.employee.EmployeeData.model.Employee;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** java.util.List;

**public** **class** EmployeeDataApplication {

**public** **static** **void** main(String[] args) {

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

EmployeeDao employeeDao = (EmployeeDao) context.getBean("employeeDao");

List<Employee> employeeList = employeeDao.getAllEmployees();

**for** (Employee emp : employeeList) {

System.***out***.println("ID: " + emp.getId());

System.***out***.println("Name: " + emp.getName());

System.***out***.println("Department: " + emp.getDepartment().getName());

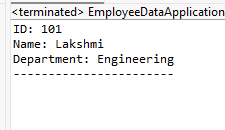
System.***out***.println("-----------------------");

}

}

}

**Output:**



**Create REST service for department**

**Department.java**

**public** **class** Department {

**private** **int** id;

**private** String name;

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

}

**DepartmentDao.java**

**package** com.employee.EmployeeData.dao;

**import** com.employee.EmployeeData.model.Department;

**import** java.util.List;

**public** **class** DepartmentDao {

**private** List<Department> departmentList;

**public** DepartmentDao(List<Department> departmentList) {

**this**.departmentList = departmentList;

}

**public** List<Department> getAllDepartments() {

**return** departmentList;

}

}

**DepartmentService.java**

**package** com.employee.EmployeeData.service;

**import** com.employee.EmployeeData.dao.DepartmentDao;

**import** com.employee.EmployeeData.model.Department;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** java.util.List;

@Service

**public** **class** DepartmentService {

**private** **final** DepartmentDao departmentDao;

**public** DepartmentService(DepartmentDao departmentDao) {

**this**.departmentDao = departmentDao;

}

@Transactional

**public** List<Department> getAllDepartments() {

**return** departmentDao.getAllDepartments();

}

}

**DepartmentController.java**

**package** com.employee.EmployeeData.controller;

**import** com.employee.EmployeeData.model.Department;

**import** com.employee.EmployeeData.service.DepartmentService;

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

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

**import** java.util.List;

@RestController

**public** **class** DepartmentController {

**private** **final** DepartmentService departmentService;

**public** DepartmentController(DepartmentService departmentService) {

**this**.departmentService = departmentService;

}

@GetMapping("/departments")

**public** List<Department> getAllDepartments() {

**return** departmentService.getAllDepartments();

}

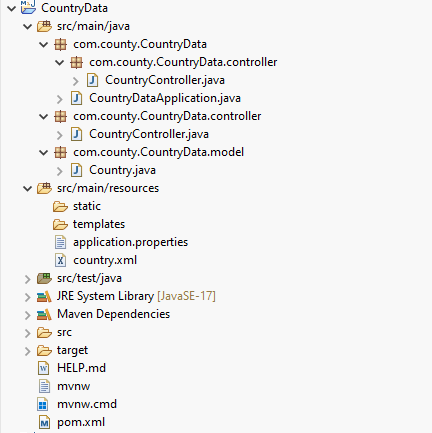
}

**Postman Output**



**Docx 4**

**Significance of HTTP Method Types in RESTful Web Services**



**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.county</groupId>

<artifactId>CountryData</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>CountryData</name>

<description>Demo project for Spring Rest</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-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Country.xml**

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

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<bean id="country" class="com.county.CountryData.model.Country">

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

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

</bean>

</beans>

**Country.java**

**package** com.county.CountryData.model;

**public** **class** Country {

**private** String code;

**private** String name;

**public** Country() {}

**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;

}

}

**CountryDataApplication.java**

**package** com.county.CountryData;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@ImportResource("classpath:country.xml")

**public** **class** CountryDataApplication {

**public** **static** **void** main(String[] args) {

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

}

}

**CountryController.java**

**package** com.county.CountryData.controller;

**import** com.county.CountryData.model.Country;

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

**import** org.springframework.context.ApplicationContext;

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

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

@RestController

**public** **class** CountryController {

@Autowired

**private** ApplicationContext context;

@GetMapping("/country")

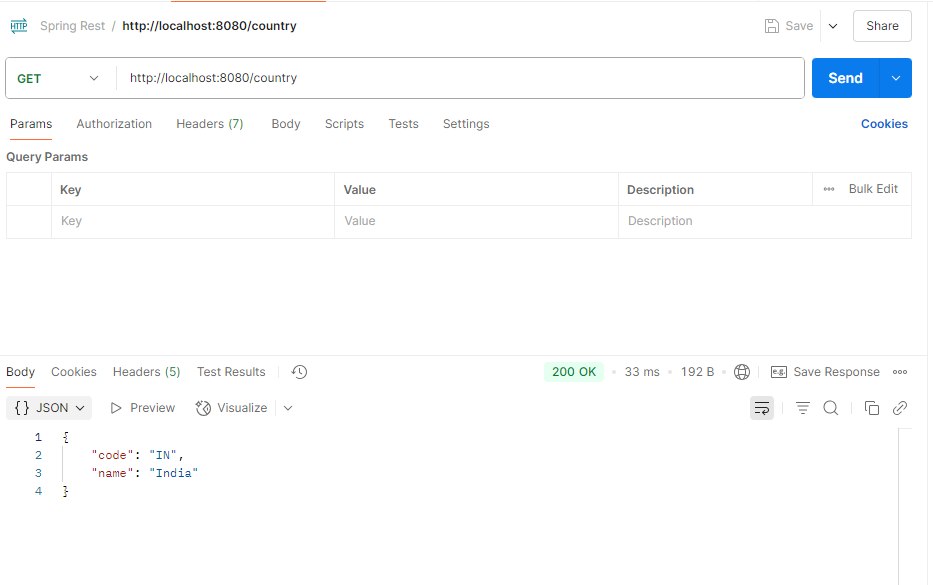
**public** Country getCountryIndia() {

**return** (Country) context.getBean("country");

}

}

**Postman Output**



**Create RESTful Web Service to handle POST request of Country**

**CountryController.java**

**package** com.county.CountryData.controller;

**import** com.county.CountryData.model.Country;

**import** jakarta.validation.Valid;

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

**import** org.springframework.context.ApplicationContext;

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

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

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

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

@RestController

**public** **class** CountryController {

@Autowired

**private** ApplicationContext context;

@GetMapping("/country")

**public** Country getCountryIndia() {

**return** (Country) context.getBean("country");

}

@PostMapping

**public** Country addCountry(@RequestBody @Valid Country country) {

LOGGER.info("Start");

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

**return** country;

}

}

**Country.java**

**import** jakarta.validation.constraints.NotNull;

**import** jakarta.validation.constraints.Size;

@NotNull

@Size(min = 2, max = 2, message = "Country code should be 2 characters")

**private** String code;

@NotNull(message = "Country name is required")

**private** String name;

**GlobalExceptionHandler.java**

**package** com.county.CountryData.controller;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.\*;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

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

**import** org.springframework.web.context.request.WebRequest;

**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

**import** java.util.\*;

**import** java.util.stream.Collectors;

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

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

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatusCode status,

WebRequest request) {

***LOGGER***.info("Validation error handled in GlobalExceptionHandler");

Map<String, Object> body = **new** LinkedHashMap<>();

body.put("timestamp", **new** Date());

body.put("status", status.value());

// error messages

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

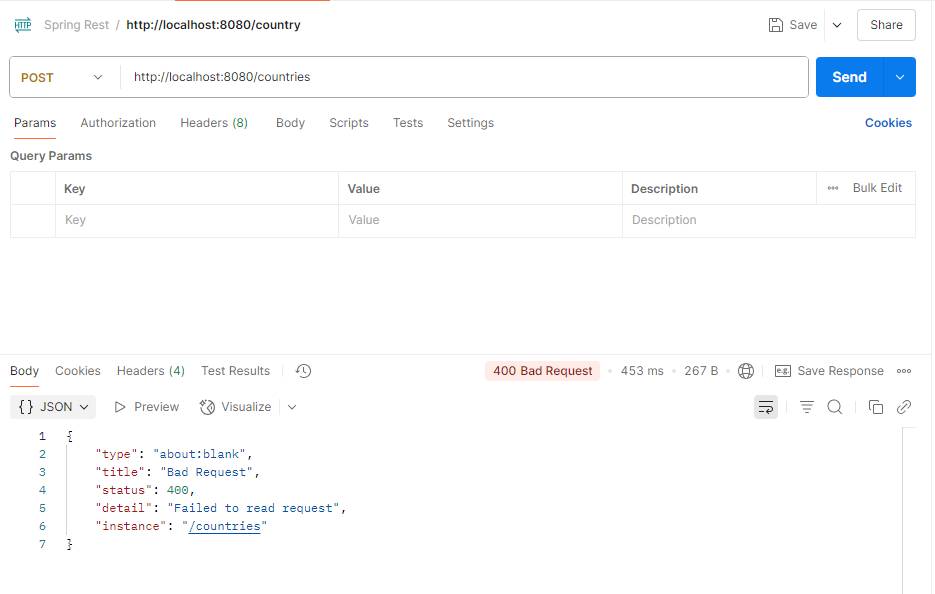
.map(fieldError -> fieldError.getDefaultMessage())

.collect(Collectors.*toList*());

body.put("errors", errors);

***LOGGER***.info("Validation errors: {}", errors);

**return** **new** ResponseEntity<>(body, headers, status);

**Output** 

**Read country data as a bean in RESTful Web Service**

**Country.java**

@NotNull

@Size(min = 2, max = 2, message = "Country code should be 2 characters")

**private** String code;

**CountryController.java**

@PostMapping

**public** Country addCountry(@RequestBody @Valid Country country) {

***LOGGER***.info("Start - addCountry");

***LOGGER***.info("Country code: " + country.getCode());

***LOGGER***.info("Country name: " + country.getName());

**return** country;

}

**GlobalExceptionHandler.java**

**package** com.county.CountryData.controller;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

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

**import** org.springframework.web.context.request.WebRequest;

**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

**import** java.util.\*;

**import** java.util.stream.Collectors;

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

**private** **static** **final** org.slf4j.Logger ***LOGGER*** = org.slf4j.LoggerFactory.*getLogger*(GlobalExceptionHandler.**class**);

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

***LOGGER***.info("Start - Global Exception Handler");

Map<String, Object> body = **new** LinkedHashMap<>();

body.put("timestamp", **new** Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(x -> x.getDefaultMessage())

.collect(Collectors.*toList*());

body.put("errors", errors);

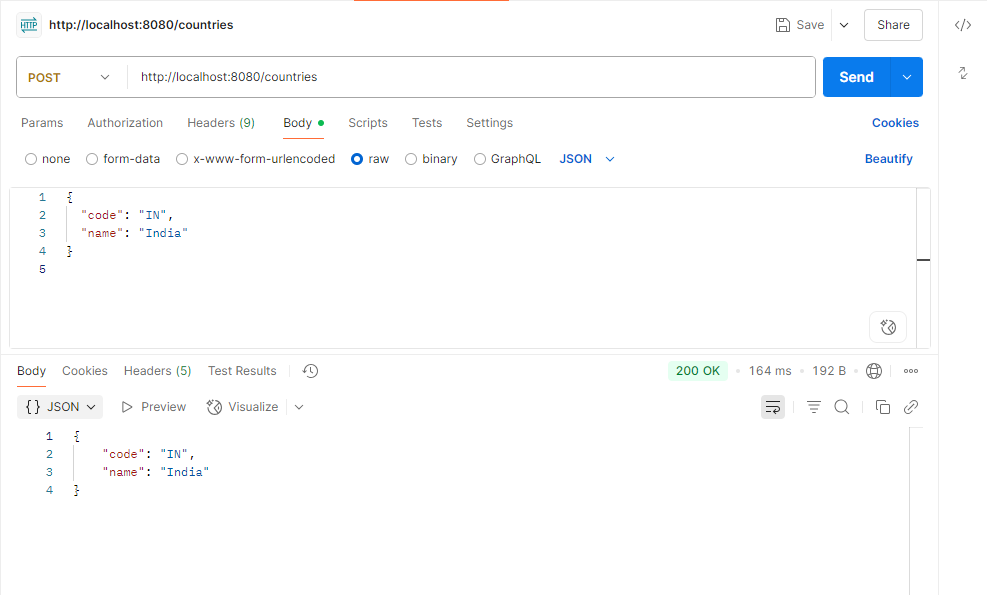
***LOGGER***.info("End - Global Exception Handler");

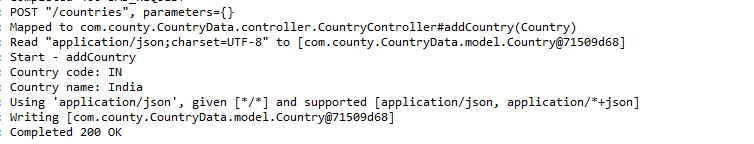
**return** **new** ResponseEntity<>(body, headers, status);

}

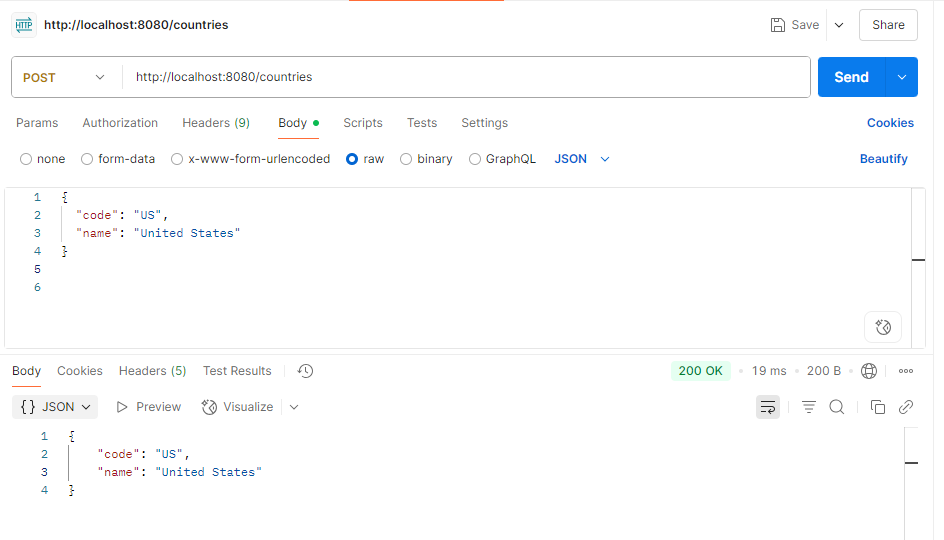
}

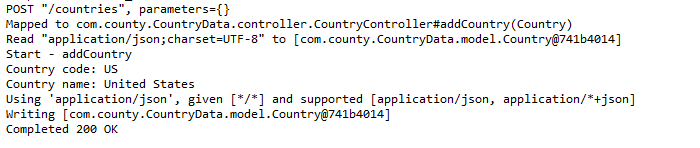
**Output:**



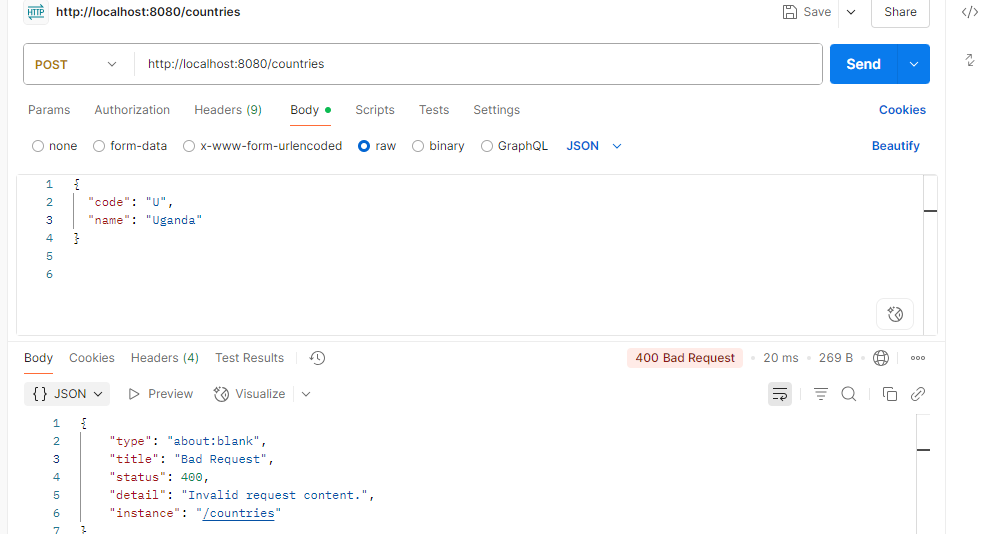


**Validating country code**

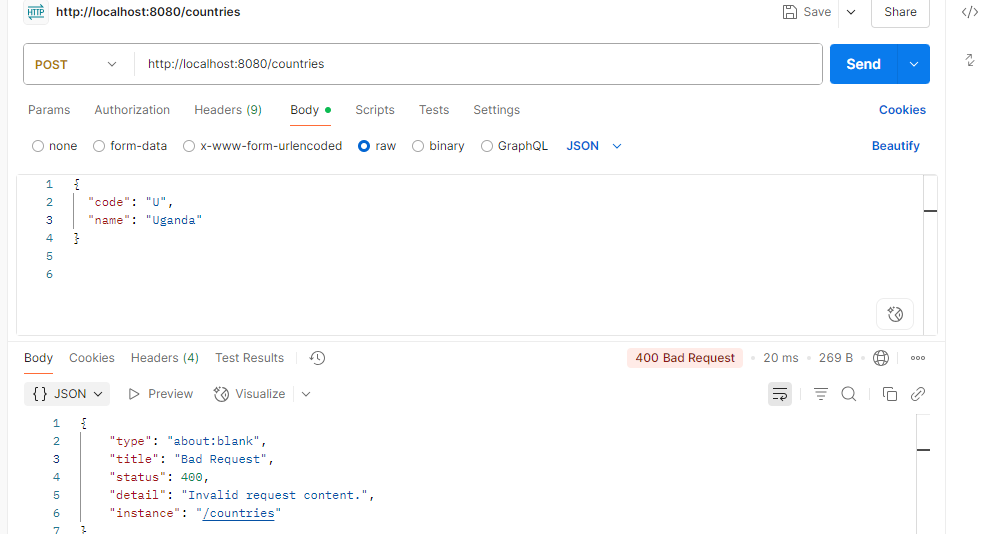




**Error Country Code**



Resolved [org.springframework.web.bind.MethodArgumentNotValidException: Validation failed for argument [0] in public com.county.CountryData.model.Country com.county.CountryData.controller.CountryController.addCountry(com.county.CountryData.model.Country): [Field error in object 'country' on field 'code': rejected value [U]; codes [Size.country.code,Size.code,Size.java.lang.String,Size]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [country.code,code]; arguments []; default message [code],2,2]; default message **[Country code should be 2 characters]] ]**



**Include global exception handler for validation errors**

**GlobalExceptionHandler.java**

**package** com.county.CountryData.controller;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

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

**import** org.springframework.web.context.request.WebRequest;

**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

**import** java.util.\*;

**import** java.util.stream.Collectors;

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

**private** **static** **final** org.slf4j.Logger ***LOGGER*** = org.slf4j.LoggerFactory.*getLogger*(GlobalExceptionHandler.**class**);

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

***LOGGER***.info("Start - Global Exception Handler");

Map<String, Object> body = **new** LinkedHashMap<>();

body.put("timestamp", **new** Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(x -> x.getDefaultMessage())

.collect(Collectors.*toList*());

body.put("errors", errors);

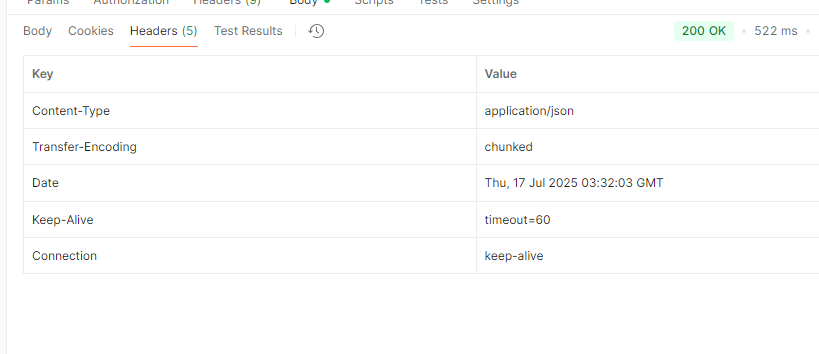
***LOGGER***.info("End - Global Exception Handler");

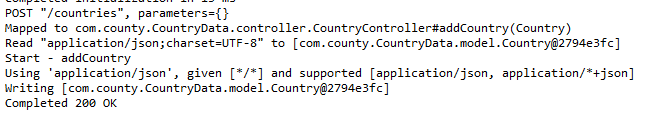
**return** **new** ResponseEntity<>(body, headers, status);

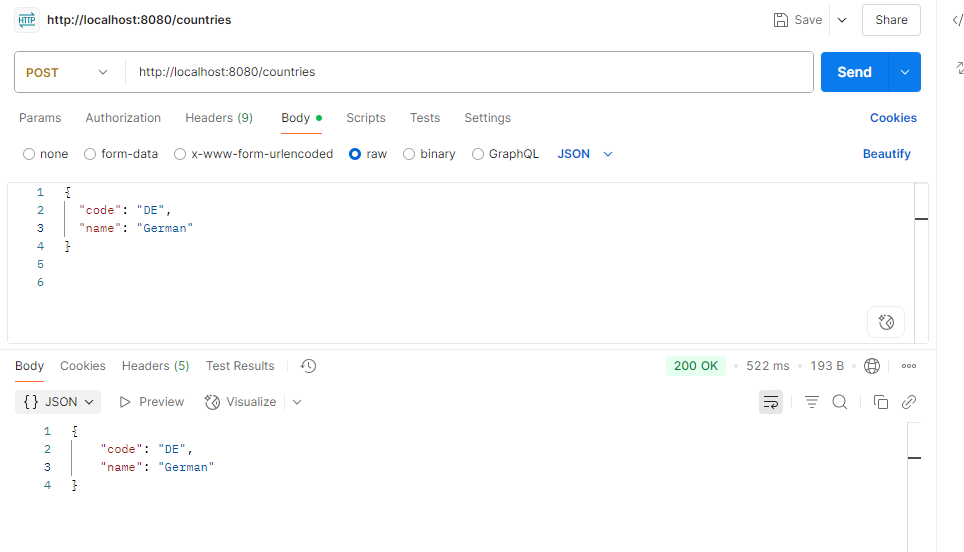
}

}

Country.







**Implement REST service for updating an employee**

**Employee.java**

**package** com.county.CountryData.model;

**import** jakarta.validation.constraints.\*;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Employee {

@NotNull

**private** **int** id;

@NotNull

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

@NotNull

@Min(value = 0)

**private** Double salary;

@NotNull

**private** Boolean permanent;

@JsonFormat(shape = JsonFormat.Shape.***STRING***, pattern = "dd/MM/yyyy")

**private** Date dateOfBirth;

@NotNull

**private** Department department;

@NotNull

**private** List<Skill> skills;

// Getters and setters

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setSalary(Double salary) {

**this**.salary = salary;

}

**public** **void** setPermanent(Boolean permanent) {

**this**.permanent = permanent;

}

**public** **void** setDateOfBirth(Date dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** **void** setDepartment(Department department) {

**this**.department = department;

}

**public** **void** setSkills(List<Skill> skills) {

**this**.skills = skills;

}

**public** **int** getId() {

**return** id;

}

}

**Department.java**

**package** com.county.CountryData;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

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

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

**import** com.county.CountryData.model.Country;

@SpringBootApplication(scanBasePackages = "com.county.CountryData")

@ImportResource("classpath:country.xml")

@RequestMapping("/countries")

**public** **class** CountryDataApplication {

**public** **static** **void** main(String[] args) {

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

}

@PostMapping

**public** Country addCountry(@RequestBody Country country) {

System.***out***.println("Inside addCountry");

**return** country;

}

}

**Skill.java**

**package** com.county.CountryData.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Skill {

@NotNull

**private** **int** id;

@NotNull

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

}

**EmployeeNotFoundException.java**

**package** com.county.CountryData.exception;

**import** org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.***NOT\_FOUND***)

**public** **class** EmployeeNotFoundException **extends** RuntimeException {

**public** EmployeeNotFoundException(String message) {

**super**(message);

}

}

**EmployeeService.java**

**package** com.county.CountryData.service;

**import** java.util.List;

**import** com.county.CountryData.Dao.EmployeeDao;

**import** com.county.CountryData.model.Employee;

**public** **class** EmployeeService {

**private** EmployeeDao employeeDao = **new** EmployeeDao();

**public** **void** updateEmployee(Employee employee) {

employeeDao.updateEmployee(employee);

}

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAll();

}

}

**EmployeeDao.java**

**package** com.county.CountryData.Dao;

**import** java.util.\*;

**import** com.county.CountryData.exception.EmployeeNotFoundException;

**import** com.county.CountryData.model.Department;

**import** com.county.CountryData.model.Employee;

**import** com.county.CountryData.model.Skill;

**public** **class** EmployeeDao {

**private** **static** List<Employee> *EMPLOYEE\_LIST* = **new** ArrayList<>();

**static** {

// Preload 1 employee

Department dept = **new** Department();

dept.setId(1);

dept.setName("HR");

Skill skill = **new** Skill();

skill.setId(1);

skill.setName("Communication");

Employee e = **new** Employee();

e.setId(1);

e.setName("John");

e.setSalary(5000.0);

e.setPermanent(**true**);

e.setDateOfBirth(**new** Date());

e.setDepartment(dept);

e.setSkills(List.*of*(skill));

*EMPLOYEE\_LIST*.add(e);

}

**public** **void** updateEmployee(Employee employee) **throws** EmployeeNotFoundException {

**boolean** found = **false**;

**for** (**int** i = 0; i < *EMPLOYEE\_LIST*.size(); i++) {

**if** (*EMPLOYEE\_LIST*.get(i).getId() == employee.getId()) {

*EMPLOYEE\_LIST*.set(i, employee);

found = **true**;

**break**;

}

}

**if** (!found) **throw** **new** EmployeeNotFoundException("Employee not found with id: " + employee.getId());

}

**public** List<Employee> getAll() {

**return** *EMPLOYEE\_LIST*;

}

}

**EmployeeController.java**

**package** com.county.CountryData;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

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

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

**import** com.county.CountryData.model.Country;

@SpringBootApplication(scanBasePackages = "com.county.CountryData")

@ImportResource("classpath:country.xml")

@RequestMapping("/countries")

**public** **class** CountryDataApplication {

**public** **static** **void** main(String[] args) {

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

}

@PostMapping

**public** Country addCountry(@RequestBody Country country) {

System.***out***.println("Inside addCountry");

**return** country;

}

}

**GlobalExceptionHandler.java**

body.put("error", "Bad Request");

**if** (ex.getCause() **instanceof** InvalidFormatException cause) {

**for** (InvalidFormatException.Reference reference : cause.getPath()) {

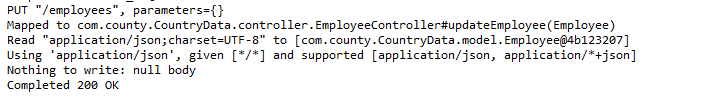
body.put("message", "Incorrect format for field '" + reference.getFieldName() + "'");

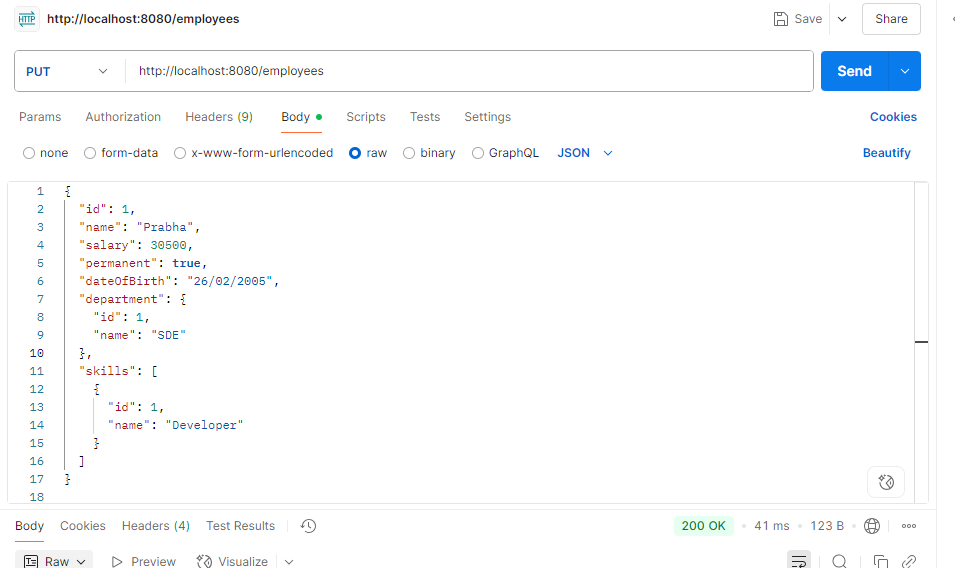
}

}

**Output:**

**Valid**

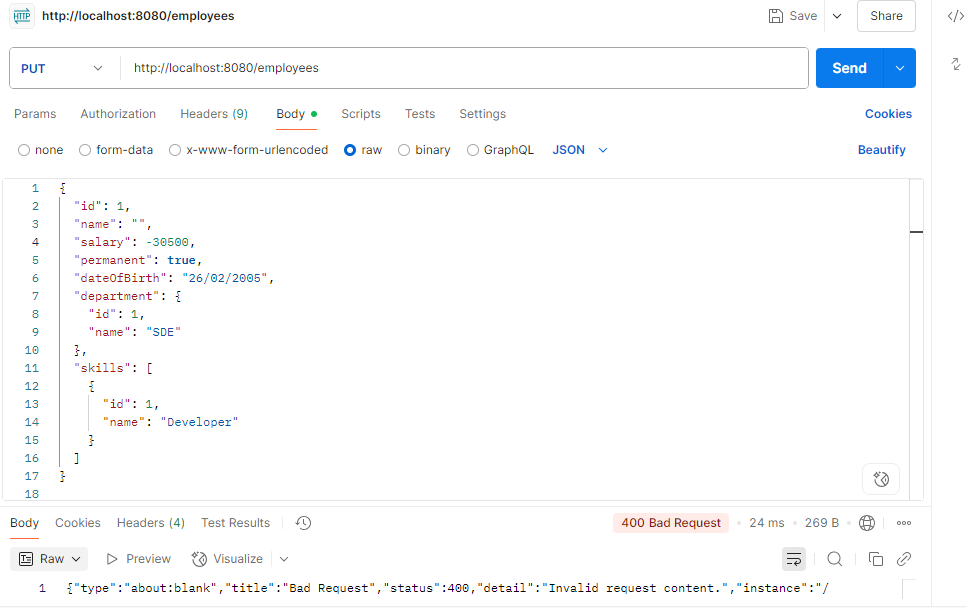




**Invalid**

Writing [ProblemDetail[type='about:blank', **title='Bad Request'**, status=400, detail='Invalid request content.' (truncated)...]

Resolved [org.springframework.web.bind.MethodArgumentNotValidException: Validation failed for argument [0] in public void com.county.CountryData.controller.EmployeeController.updateEmployee(com.county.CountryData.model.Employee) with 3 errors: [Field error in object 'employee' on field 'name': rejected value []; codes [Size.employee.name,Size.name,Size.java.lang.String,Size]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [employee.name,name]; arguments []; default message [name],30,1]; default message [size must be between 1 and 30]] [Field error in object 'employee' on field 'name': rejected value []; codes [NotBlank.employee.name,NotBlank.name,NotBlank.java.lang.String,NotBlank]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [employee.name,name]; arguments []; default message [name]]; default message [must not be blank]] [Field error in object 'employee' on field 'salary': rejected value [-30500.0]; codes [Min.employee.salary,Min.salary,Min.java.lang.Double,Min]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes **[employee.salary,salary];** arguments []; default message [salary],0]; default message **[must be greater than or equal to 0]] ]**



**Implement REST DELETE Service**

**CountryDataApplication.java**

@ComponentScan("com.county.CountryData")

**EmployeeDao.java**

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

**boolean** removed = *EMPLOYEE\_LIST*.removeIf(emp -> emp.getId() == id);

**if** (!removed) {

**throw** **new** EmployeeNotFoundException("Employee not found with id: " + id);

}

}

**EmployeeService.java**

**public** **void** deleteEmployee(**int** id) {

employeeDao.deleteEmployee(id);

}

**EmployeeController.java**

@DeleteMapping("/{id}")

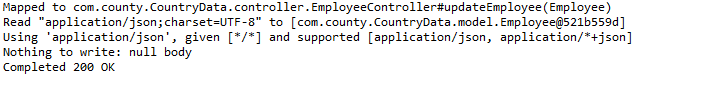
**public** **void** deleteEmployee(@PathVariable **int** id) {

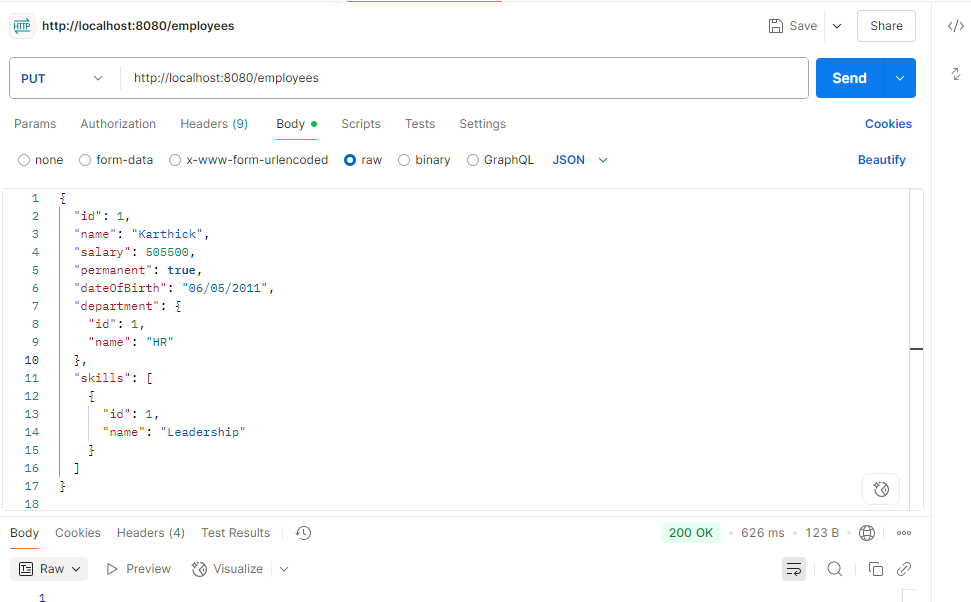
employeeService.deleteEmployee(id);

}

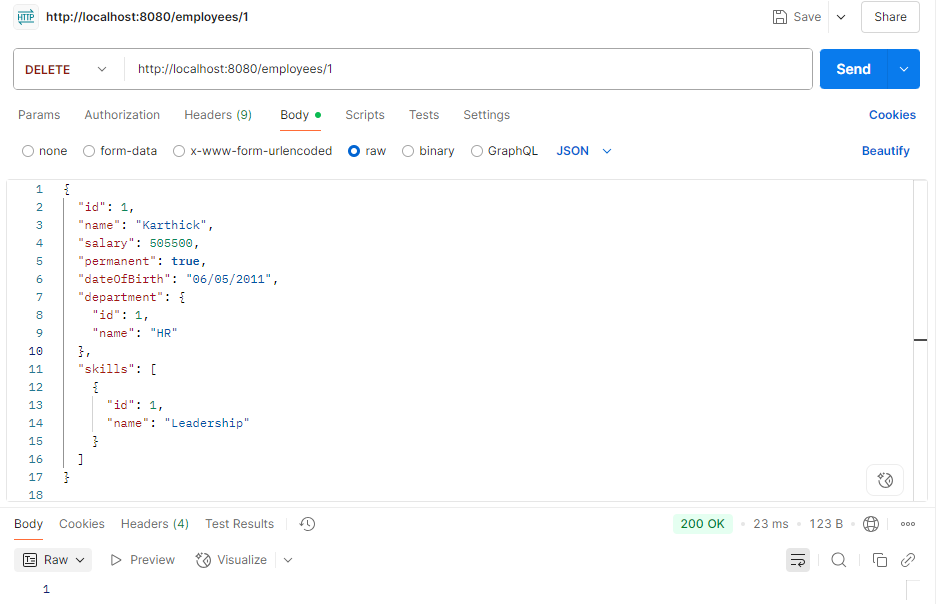
**Output:**

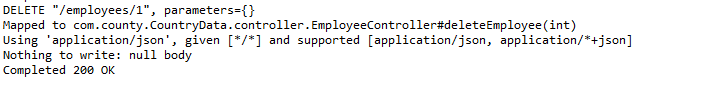
**put**





**Delete**





**Docx 5 JWT hands-on**

**Securing RESTful Web Services with Spring Security**

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

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

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.~~WebSecurityConfigurerAdapter~~;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** ~~WebSecurityConfigurerAdapter~~ {

// configure() methods and PasswordEncoder bean will go here

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable().httpBasic().and()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.addFilter(**new** JwtAuthorizationFilter(~~authenticationManager~~()));

}

}

**AuthenticationController.java**

**package** com.cognizant.spring\_learn.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.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("Inside authenticate");

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

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

**return** map;

}

**private** String getUser(String authHeader) {

String encodedCredentials = authHeader.replace("Basic ", "");

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

String decoded = **new** String(decodedBytes);

**return** decoded.split(":")[0]; // Extracts username

}

**private** String generateJwt(String user) {

**return** Jwts.*builder*()

.setSubject(user)

.setIssuedAt(**new** Date())

.setExpiration(**new** Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 min expiry

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

.compact();

}

}

**JwtAuthenticationFilter.java**

**package** com.cognizant.spring\_learn.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);

}

@Override

**protected** **void** doFilterInternal(HttpServletRequest req, HttpServletResponse res,

FilterChain chain) **throws** IOException, ServletException {

***LOGGER***.info("JWT filter started");

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

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

chain.doFilter(req, res);

**return**;

}

UsernamePasswordAuthenticationToken auth = getAuthentication(req);

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

chain.doFilter(req, res);

}

**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();

**if** (user != **null**) {

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

}

} **catch** (JwtException e) {

***LOGGER***.warn("Invalid JWT token");

}

}

**return** **null**;

}

}

**Creating users and roles in Spring Security**

**CountryController.java**

**package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.Country;

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

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

**import** java.util.\*;

@RestController

**public** **class** CountryController {

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

List<Country> countries = **new** ArrayList<>();

countries.add(createCountry("IN", "India"));

countries.add(createCountry("US", "United States"));

countries.add(createCountry("DE", "Germany"));

countries.add(createCountry("JP", "Japan"));

**return** countries;

}

**private** Country createCountry(String code, String name) {

Country country = **new** Country();

country.setCode(code);

country.setName(name);

**return** country;

}

}

**Country.java**

**package** com.cognizant.spring\_learn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** Country {

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

**private** String code;

**private** String name;

**public** Country() {

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

}

**public** String getCode() {

***LOGGER***.debug("Getting Code: {}", code);

**return** code;

}

**public** **void** setCode(String code) {

***LOGGER***.debug("Setting Code: {}", code);

**this**.code = code;

}

**public** String getName() {

***LOGGER***.debug("Getting Name: {}", name);

**return** name;

}

**public** **void** setName(String name) {

***LOGGER***.debug("Setting Name: {}", name);

**this**.name = name;

}

@Override

**public** String toString() {

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

}

}

**SecurityConfig.java**

@Override

**protected** **void** configure(HttpSecurity httpSecurity) **throws** Exception {

httpSecurity.csrf().disable()

.httpBasic()

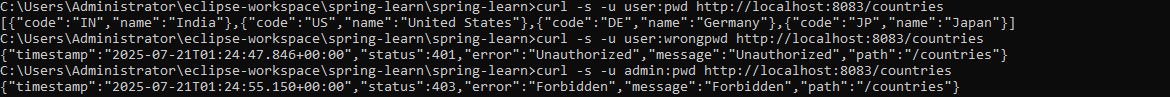
.and()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER");

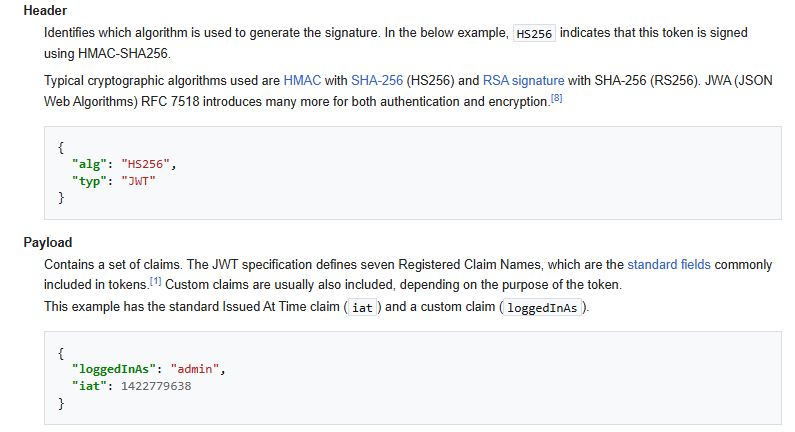
}

**Output**

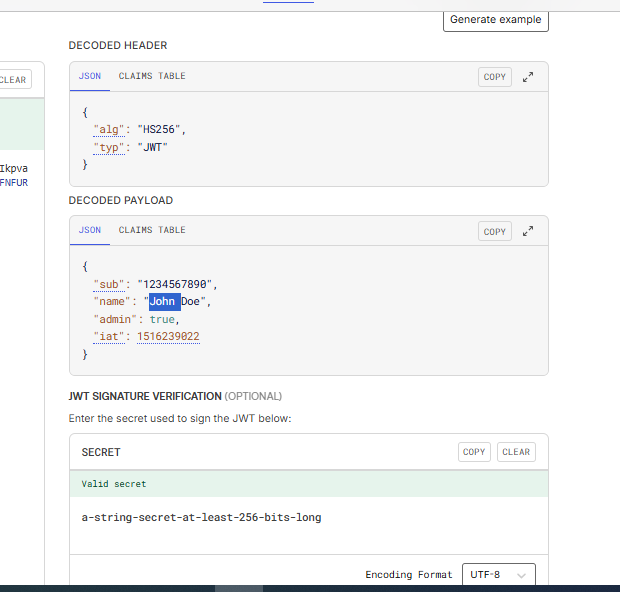


**Understanding JWT - Exercise to check how JWT token is created**

**From JWT wikipedia**



**From the jwt.io**



* This screenshot shows how a JWT is created using jwt.io.
* The header defines the signing algorithm (HS256), the payload contains user claims (sub, name, admin),
* and the signature is verified using a secret key.

**Create authentication service that returns JWT**

**Pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

**AuthenticationController.java**

**private** String generateJwt(String user) {

SecretKey key = Keys.*secretKeyFor*(SignatureAlgorithm.***HS256***);

**return** Jwts.*builder*()

.setSubject(user)

.setIssuedAt(**new** Date())

.setExpiration(**new** Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 minutes

.signWith(key)

.compact();

}

**Country.java**

**package** com.cognizant.spring\_learn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** Country {

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

**private** String code;

**private** String name;

**public** Country() {

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

}

**public** String getCode() {

***LOGGER***.debug("Getting Code: {}", code);

**return** code;

}

**public** **void** setCode(String code) {

***LOGGER***.debug("Setting Code: {}", code);

**this**.code = code;

}

**public** String getName() {

***LOGGER***.debug("Getting Name: {}", name);

**return** name;

}

**public** **void** setName(String name) {

***LOGGER***.debug("Setting Name: {}", name);

**this**.name = name;

}

@Override

**public** String toString() {

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

}

}

**CountryController.java**

**package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.Country;

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

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

**import** java.util.\*;

@RestController

**public** **class** CountryController {

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

List<Country> countries = **new** ArrayList<>();

countries.add(createCountry("IN", "India"));

countries.add(createCountry("US", "United States"));

countries.add(createCountry("DE", "Germany"));

countries.add(createCountry("JP", "Japan"));

**return** countries;

}

**private** Country createCountry(String code, String name) {

Country country = **new** Country();

country.setCode(code);

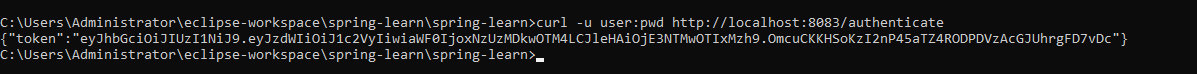
country.setName(name);

**return** country;

}

}

**Output**



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

**AuthenticationController.java**

**package** com.cognizant.spring\_learn.controller;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

**import** java.util.HashMap;

**import** java.util.Map;

@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);

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

map.put("token", ""); // placeholder or your generated token

***LOGGER***.info("End /authenticate");

**return** map;

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.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.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.~~WebSecurityConfigurerAdapter~~;

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

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

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** ~~WebSecurityConfigurerAdapter~~ {

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

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

auth.inMemoryAuthentication()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER")

.and()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN");

}

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable()

.httpBasic()

.and()

.authorizeRequests()

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

.antMatchers("/countries").hasRole("USER");

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

@Bean

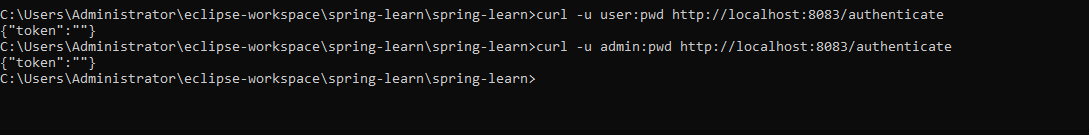
**public** AuthenticationManager authenticationManagerBean() **throws** Exception {

**return** **super**.~~authenticationManagerBean~~();

}

}

**Output:**



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

**AuthenticationController.java**

**package** com.cognizant.spring\_learn.controller;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@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); // ← extract username

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

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

map.put("token", ""); // next hands-on will generate token

**return** map;

}

**private** String getUser(String authHeader) {

// Remove "Basic " prefix

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

// Decode base64 to "user:pwd"

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

String decodedString = **new** String(decodedBytes);

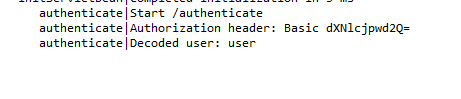
// Extract username before colon

**return** decodedString.split(":")[0];

}

}

**Output:**



**Generate token based on the user**

**AuthenticationController.java**

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

**import** javax.crypto.SecretKey;

@GetMapping("/authenticate")

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

String user = getUser(authHeader); // already decoded earlier

String token = generateJwt(user); // now generate token

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

map.put("token", token);

**return** map;

}

**private** String generateJwt(String user) {

SecretKey key = Keys.*secretKeyFor*(SignatureAlgorithm.***HS256***);

**return** Jwts.*builder*()

.setSubject(user)

.setIssuedAt(**new** Date())

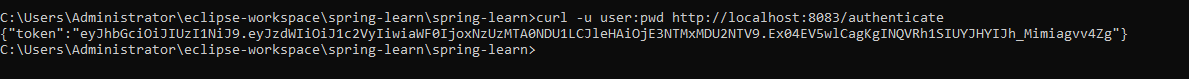
.setExpiration(**new** Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 mins

.signWith(key)

.compact();

}

**Output:**



**Authorize based on JWT**

**JwtAuthenticationFilter.java**

**package** com.cognizant.spring\_learn.security;

**import** io.jsonwebtoken.\*;

**import** io.jsonwebtoken.security.Keys;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

**import** javax.crypto.SecretKey;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** java.io.IOException;

**import** java.util.ArrayList;

**public** **class** JwtAuthorizationFilter **extends** BasicAuthenticationFilter {

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

**private** **final** SecretKey key = Keys.*secretKeyFor*(SignatureAlgorithm.***HS256***); // Same method as token creation

**public** JwtAuthorizationFilter(org.springframework.security.authentication.AuthenticationManager authManager) {

**super**(authManager);

}

@Override

**protected** **void** doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)

**throws** IOException, ServletException {

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

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

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

chain.doFilter(req, res);

**return**;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(header);

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

chain.doFilter(req, res);

}

**private** UsernamePasswordAuthenticationToken getAuthentication(String tokenHeader) {

**try** {

String token = tokenHeader.replace("Bearer ", "");

String user = Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

**if** (user != **null**) {

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

}

} **catch** (JwtException e) {

***LOGGER***.error("Invalid JWT: {}", e.getMessage());

}

**return** **null**;

}

}

**Output:**

