**SPRING REST USING SPRINGBOOT**

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

package com.cognizat.spring\_learn;

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);

        System.out.println("SpringLearnApplication has started successfully.");

    }

}

package com.cognizat.spring\_learn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

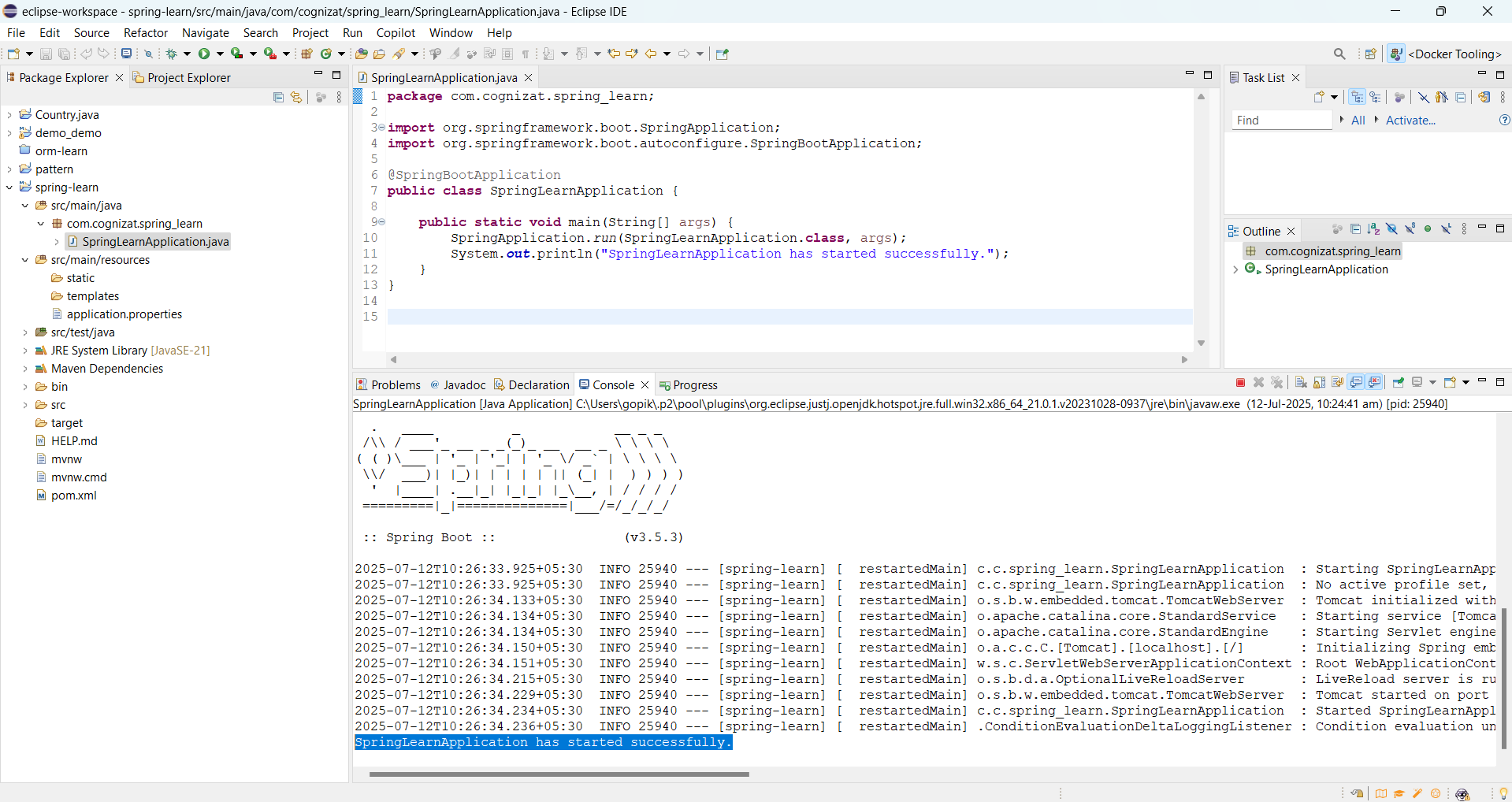
    @Test

    void contextLoads() {

    }

}

**Output:**



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

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.debug("START of main()");

displayCountry();

LOGGER.debug("END of main()");

}

public static void displayCountry() {

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

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

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

}

}

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 + "]";

    }

}

package com.cognizat.spring\_learn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

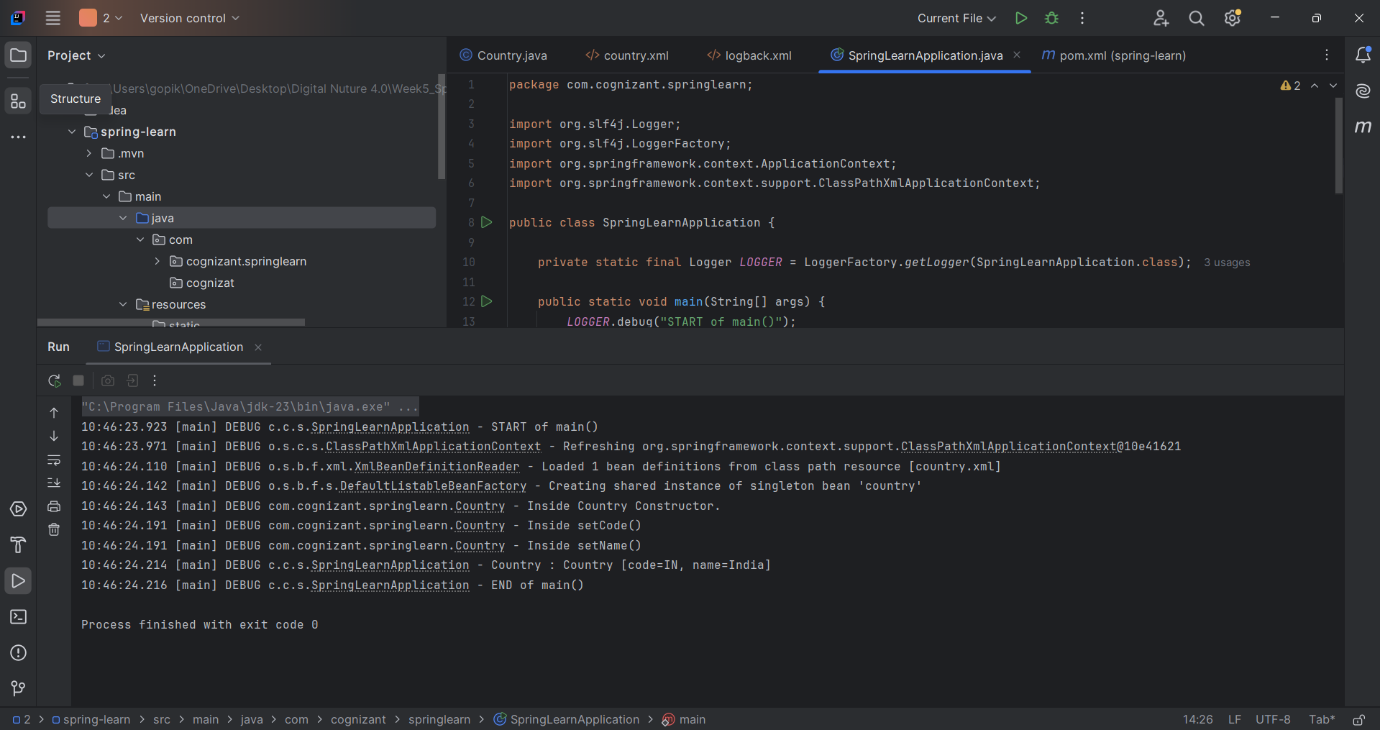
    @Test

    void contextLoads() {

    }

}

**Output:**



1. **Hello World RESTful Web Service**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

package com.cognizant.spring\_learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {

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

    @GetMapping("/hello")

    public String sayHello() {

        LOGGER.debug("START of sayHello()");

        String message = "Hello World!!";

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

        return message;

    }

}

package com.cognizant.spring\_learn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

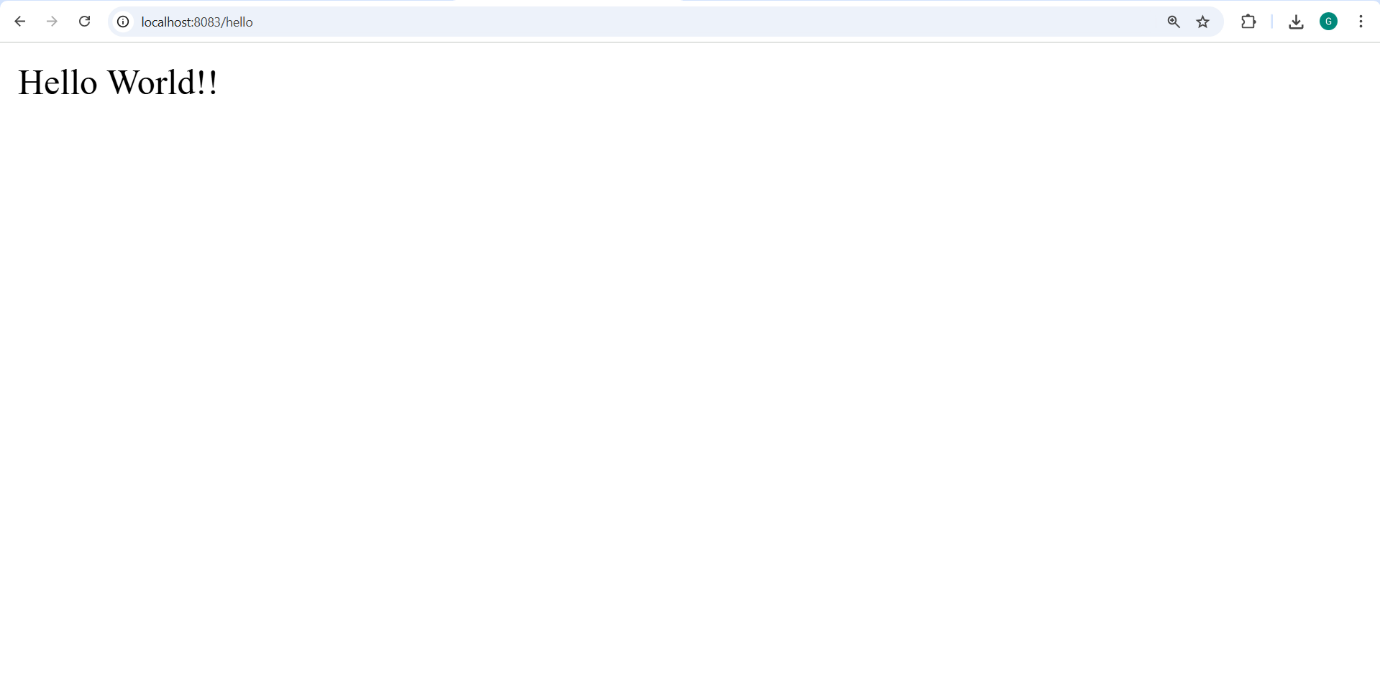
    @Test

    void contextLoads() {

    }

}

**Output:**



1. **REST - Country Web Service**

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 + "]";

}

}

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);

    }

}

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController {

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

    @RequestMapping("/country")

    public Country getCountryIndia() {

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

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

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

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

        return country;

    }

}

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

    @Test

    void contextLoads() {

    }

}

**Output:**



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

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);

}

}

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("/country/{code}")

    public Country getCountry(@PathVariable String code) {

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

        Country country = countryService.getCountry(code);

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

        return country;

    }

}

package com.cognizant.springlearn.model;

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 + "]";

    }

}

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

        return countries.stream()

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

                .findFirst()

                .orElse(null);  // Optionally throw an exception if not found

    }

}

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

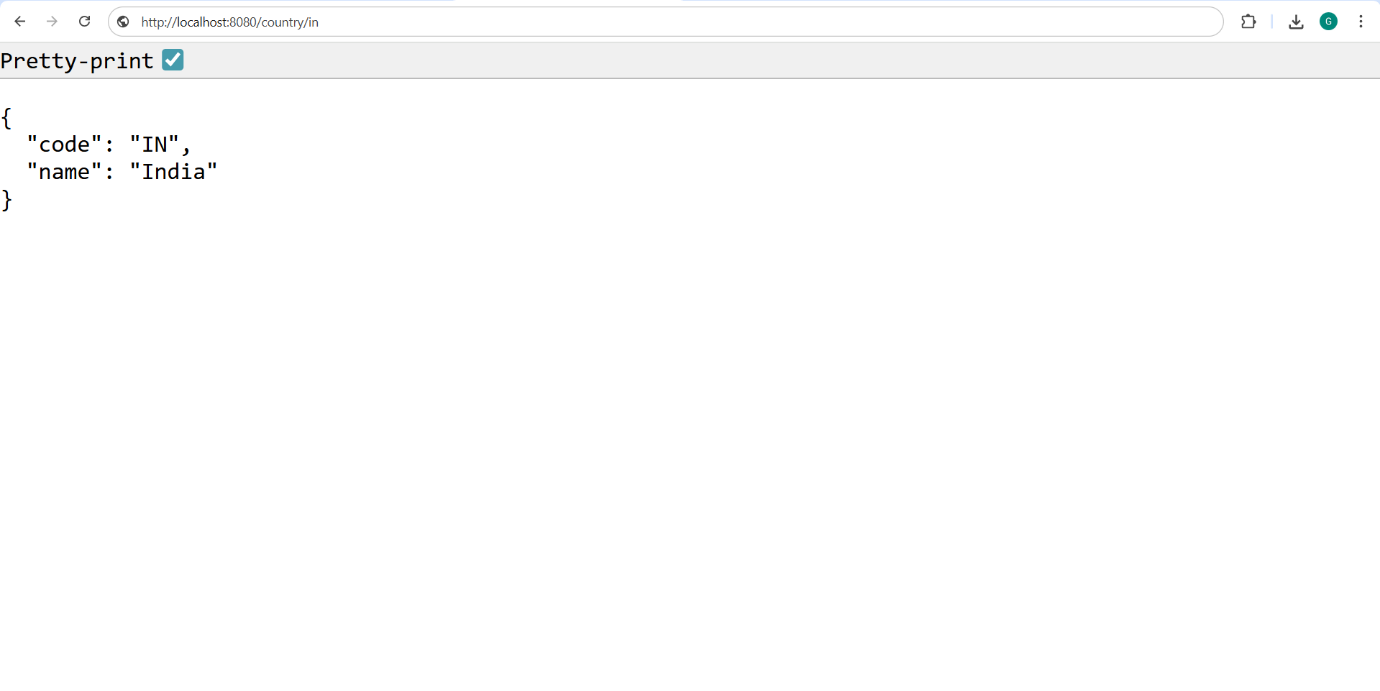
    @Test

    void contextLoads() {

    }

}

**Output:**



1. **Create authentication service that returns JWT**

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);

}

}

package com.cognizant.springlearn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

public class SecurityConfig {

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http

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

                .authorizeHttpRequests(auth -> auth

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

                        .anyRequest().authenticated()

                )

                .httpBasic(Customizer.withDefaults());

        return http.build();

    }

}

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import jakarta.servlet.http.HttpServletRequest;

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

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

import java.util.Base64;

import java.util.Map;

import java.util.HashMap;

@RestController

public class AuthenticationController {

    @Autowired

    private JwtUtil jwtUtil;

    @GetMapping("/authenticate")

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

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

        if (authHeader != null && authHeader.startsWith("Basic ")) {

            // Decode base64 encoded credentials

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

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

            String credentials = new String(credDecoded);

            String[] values = credentials.split(":", 2);

            String username = values[0];

            String password = values[1];

            // Simple hardcoded check

            if ("user".equals(username) && "pwd".equals(password)) {

                String token = jwtUtil.generateToken(username);

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

                response.put("token", token);

                return response;

            }

        }

        throw new RuntimeException("Invalid Credentials");

    }

}

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.util.Date;

import java.security.Key;

@Component

public class JwtUtil {

    private static final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

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

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

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

                .signWith(key)

                .compact();

    }

}

**Output:**

