1.CREATE A SPRING WEB PROJECT USING MAVEN

SpringLearnApplication.java

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) {

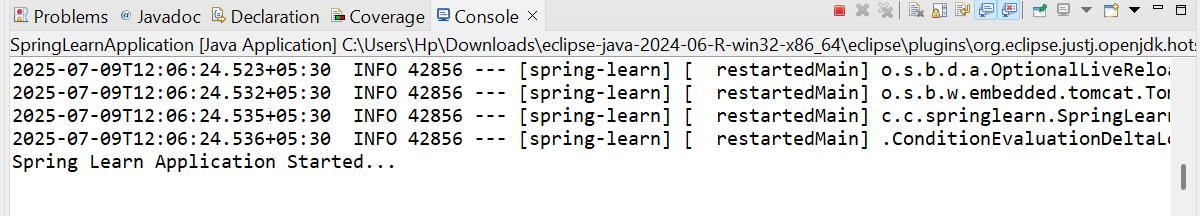
SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("Spring Learn Application Started...");

}

}

OUTPUT:



2.SPRING CORE – LOAD COUNTRY FROM SPRING CONFIGURATION XML

Country.java

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private String code;

private String name;

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

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 + "]";

}

}

SpringLearnApplication.java

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("Inside main");

*displayCountry*();

}

public static void displayCountry() {

@SuppressWarnings("resource")

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

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

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

}

}

Country.xml

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

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

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

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

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

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

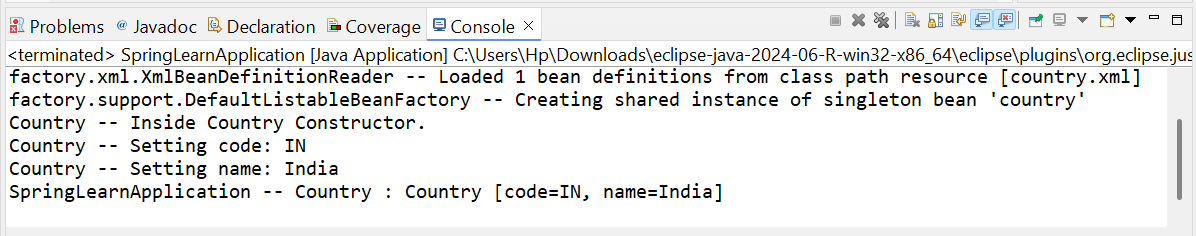
<property name="code" value="IN" />

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

</bean>

</beans>

OUTPUT:



3.HELLO WORLD RESTful WEB SERVICE

HelloController.java

package com.springlearn;

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() method called");

String response = "Hello World!!";

*LOGGER*.info("END: sayHello() method completed");

return response;

}

}

Springlearnapplication.java

package com.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);

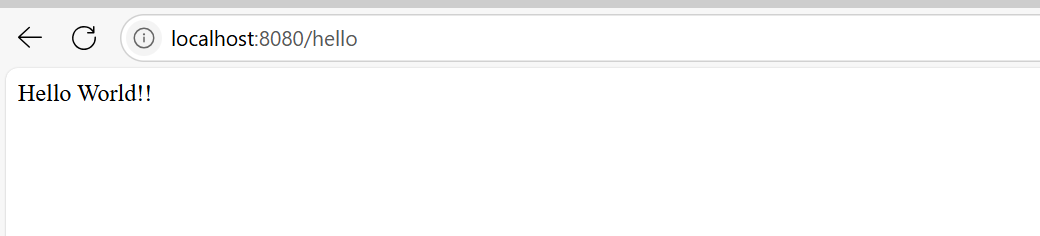
}

}

Application.properties

server.port=8083

OUTPUT:



4.REST COUNTRY WEB SERVICE

Country.java

package com.cognizant.springlearn;

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;

}

}

SpringLearnApplication.java

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan(basePackages = {"com.cognizant.springlearn", "com.cognizant.spring\_learn.controller"})

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

CountryController.java

package com.cognizant.spring\_learn.controller;

import com.cognizant.springlearn.Country;

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 {

@SuppressWarnings("resource")

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return india;

}

}

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.cognizant.springlearn.Country">

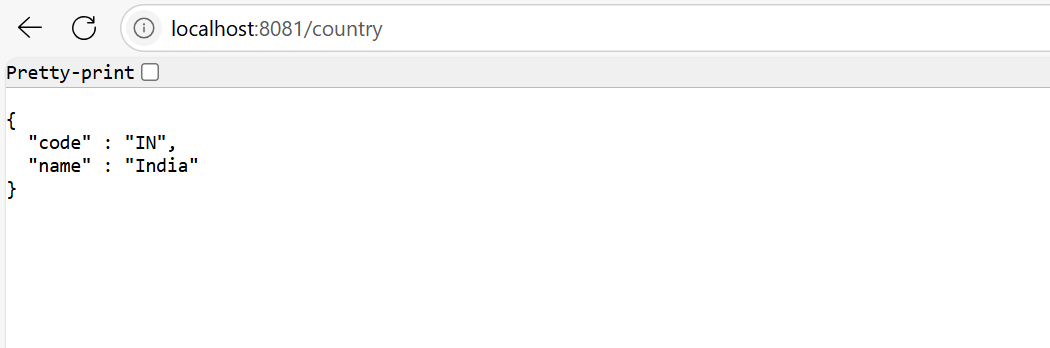
<property name="code" value="IN"/>

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

</bean>

</beans>

OUTPUT:



5.REST – GET COUNTRY BASED ON COUNTRY CODE

Country.java

package com.cognizant.springlearn;

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;

}

}

CountryService.java

package com.cognizant.springlearn;

public interface CountryService {

Country getCountry(String code);

}

CountryServiceImpl.java

package com.cognizant.springlearn;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryServiceImpl implements CountryService {

@Override

public Country getCountry(String code) {

List<Country> countryList = List.*of*(

new Country("IN", "India"),

new Country("US", "United States")

);

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

SpringLearnApplication.java

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan(basePackages = {"com.cognizant.springlearn", "com.cognizant.spring\_learn.controller"})

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

CountryController.java

package com.cognizant.spring\_learn.controller;

import com.cognizant.springlearn.Country;

import com.cognizant.springlearn.CountryService;

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

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

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

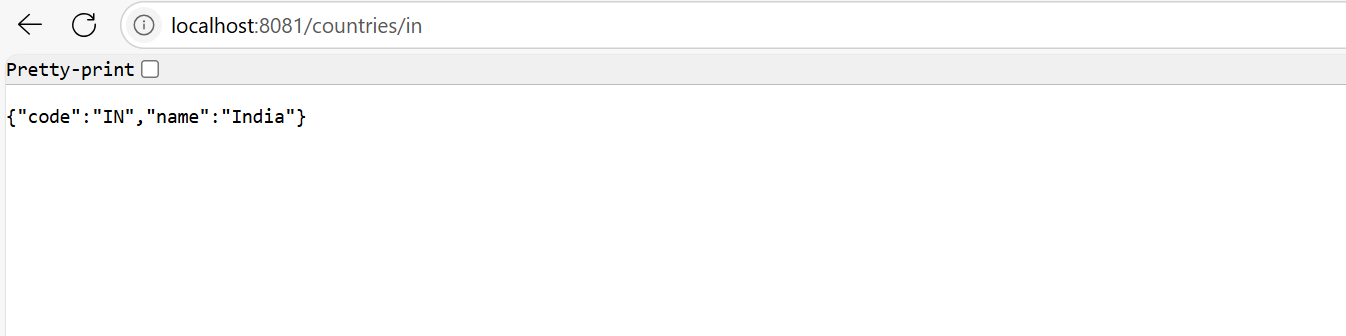
public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

OUTPUT:



6.CREATE AUTHENTICATION SERVICE THAT RETURN JWT

SecurityConfig.java

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and()

.httpBasic();

return http.build();

}

}

JwtUtil.java

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String SECRET\_KEY = "mysecretkey";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

AuthenticationController.java

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

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import jakarta.servlet.http.HttpServletRequest;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

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

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

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

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

String token = jwtUtil.generateToken(username);

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

} else {

return ResponseEntity.status(401).body("Invalid Credentials");

}

}

return ResponseEntity.badRequest().body("Missing Authorization header");

}

}

OUTPUT:

