**Week 4**

**Create a Spring Web Project using Maven**

**Program:**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("SpringLearnApplication started successfully");

}

}

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

**Program:**

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("Getting Code");

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting Code");

this.code = code;

}

public String getName() {

LOGGER.debug("Getting Name");

return name;

}

public void setName(String name) {

LOGGER.debug("Setting 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.info("START");

displayCountry();

LOGGER.info("END");

}

public static void displayCountry() {

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

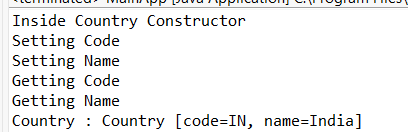
Country country = context.getBean("country", Country.class);

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

}

}

**Output:**



**Hello World RESTful Web Service**

**Program:**

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.info("START: sayHello()");

String response = "Hello World!!";

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

return response;

}

}

**REST - Country Web Service**

**Program:**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String 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;

}

}

**CountryController**

package com.cognizant.spring\_learn.controller;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController {

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

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

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

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

return country;

}

}

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

**Service Class:**

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.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> countryList = context.getBean("countryList", List.class);

return countryList.stream()

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

.findFirst()

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

}

}

**Controller Class**

package com.cognizant.spring\_learn.controller;

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

import com.cognizant.spring\_learn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

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

Country country = countryService.getCountry(code);

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

return country;

}

}

**Create authentication service that returns JWT**

**Program:**

package com.example.demo.config;

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").authenticated()

.anyRequest().permitAll()

.and()

.httpBasic();

return http.build();

}

}

**JwtUtil.java**

package com.example.demo.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secret = "secret123"; // Use env/secure in real apps

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

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

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

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

**AuthController.java**

package com.example.demo.controller;

import com.example.demo.util.JwtUtil;

import jakarta.servlet.http.HttpServletRequest;

import org.apache.tomcat.util.codec.binary.Base64;

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

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

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@RequestMapping("/authenticate")

public TokenResponse authenticate(HttpServletRequest request) {

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

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

throw new RuntimeException("Missing or invalid Authorization header");

}

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

byte[] credDecoded = Base64.decodeBase64(base64Credentials);

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

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

String token = jwtUtil.generateToken(username);

return new TokenResponse(token);

} else {

throw new RuntimeException("Invalid credentials");

}

}

static class TokenResponse {

private String token;

public TokenResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

public void setToken(String token) {

this.token = token;

}

}

}