**Create a Spring Web Project using Maven**

SpringLearnApplication.java--package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("START SpringLearnApplication main()");

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("END SpringLearnApplication main()");

}

}

OUTPUT--START SpringLearnApplication main()

:: Spring Boot :: (v3.x.x)

... (Spring Boot logs)

... Tomcat started on port(s): 8080 (http) with context path ''

... Started SpringLearnApplication in X.XXX seconds (JVM running for X.XXX)

END SpringLearnApplication main()

Spring Core - Load SimpleDateFormat from Spring XML

Update SpringLearnApplication.java---public static void displayDate() {

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

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

try {

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

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

} catch (ParseException e) {

e.printStackTrace();

}

}

Main()--public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayDate(); // <-- Call the method

}

OUTPUT 2--START SpringLearnApplication main()

:: Spring Boot :: (v3.x.x)

... (Spring Boot logs)

... Tomcat started on port(s): 8080 (http) with context path ''

... Started SpringLearnApplication in X.XXX seconds (JVM running for X.XXX)

Parsed Date: Mon Dec 31 00:00:00 IST 2018

END SpringLearnApplication main()

Hello World RESTful Web Service—

package com.cognizant.spring\_learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

public class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

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

String message = "Hello World!!";

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

return message;

}

}

OUTPUT-Hello World!!

REST - Country Web Service (Static India)—

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

// Getters and Setters

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; }

}

Controller Class: CountryController-package com.cognizant.spring\_learn.controller;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return country;

}

}

OUTPUT--{

"code": "IN",

"name": "India"

}

3. **REST - Get Country by Code**

Service Class: CountryService-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> countries = context.getBean("countryList", List.class);

return countries.stream()

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

.findFirst()

.orElse(null);

}

}

OUTPUT-{

"code": "IN",

"name": "India"

}

Create Authentication Controller

AuthenticationController.java-package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

@RestController

public class AuthenticationController {

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

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

// Decode base64 encoded username:password

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

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

String credentials = new String(credDecoded);

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

String username = userDetails[0];

String password = userDetails[1];

// Validate hardcoded user

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

String token = JwtUtil.generateToken(username);

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

}

}

return ResponseEntity.status(401).body("{\"error\":\"Invalid Credentials\"}");

}

}

Generate JWT Utility

JwtUtil.java-package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "secret-key";

public static String generateToken(String username) {

long currentTimeMillis = System.currentTimeMillis();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(currentTimeMillis))

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

SecurityConfig.java-package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Configuration;

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

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

@SuppressWarnings("deprecation")

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().disable() // Disable CSRF for testing

.authorizeRequests()

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

.anyRequest().authenticated()

.and()

.httpBasic(); // Enable basic auth

}

}

OUTPUT-{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzAxMjM0NTY3LCJleHAiOjE3MDEyMzUxNjd9.SIGNATURE"}