**Spring Rest hands-on**

Hands on 1

Create a Spring Web Project using Maven:

-Created spring web project using start.spring.io

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) {

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

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Hands on 2**

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

SpringLearnApplication.java

package com.cognizant.springlearn;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

public static void main(String[] args) {

displayDate();

}

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();

}

}

}

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

3) Hello World RESTful Web Service Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.

HelloController.java

package com.cognizant.springlearn.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 message = "Hello World!!";

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

return message;

}

}

Output: INFO com.cognizant.springlearn.controller.HelloController - START: sayHello()

INFO com.cognizant.springlearn.controller.HelloController - END: sayHello()

4) REST - Get country based on country code Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

Country.java

package com.cognizant.springlearn.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;

}

}

CountryService.java

package com.cognizant.springlearn.service;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.cognizant.springlearn.model.Country;

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

.findFirst()

.orElse(null);

}

}

CountryController.java

package com.cognizant.springlearn.controller;

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

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

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

Create authentication service that returns JWT As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

AuthController.java

package com.cognizant.springsecurity.controller;

import org.springframework.http.ResponseEntity;

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

@RestController

public class AuthController {

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

public ResponseEntity<?> authenticate() {

return ResponseEntity.ok("Authentication endpoint hit. Token logic will be added.");

}

}

SecurityConfig.java

package com.cognizant.springsecurity.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;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests((authz) -> authz

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

.anyRequest().authenticated()

)

.httpBasic(); // Enables -u user:pwd curl auth

return http.build();

}

}

**JWT Hands-on**

Create authentication service that returns JWT As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT. Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option. Request

AuthController.java

package com.cognizant.springsecurity.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import java.util.Date;

import jakarta.servlet.http.HttpServletRequest;

@RestController

public class AuthController {

private static final String SECRET\_KEY = "mySecretKey";

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

return ResponseEntity.status(401).body("Missing or invalid Authorization header.");

}

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

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

String decodedCredentials = new String(decodedBytes);

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

String username = credentials[0];

String password = credentials[1];

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

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

}

String token = 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();

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

}

}

SecurityConfig.java

package com.cognizant.springsecurity.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.web.SecurityFilterChain;

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

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests((authz) -> authz

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

.anyRequest().authenticated()

)

.httpBasic(); // Allows Basic Auth using -u in curl

return http.build();

}

}