Week 4 :

1.Skill : spring REST using spring boot 3

Hands-on Create a spring web project using maven

pom.xml :

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-rest-demo</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<dependencies>

<!-- Spring Web for REST API -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Plugin to build executable jar -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

DemoApplication.java :  
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

HelloController.java :  
package com.example.demo.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String sayHello() {

return "Welcome to Spring Boot 3 REST API!";

}

}

application.properties :  
# Server config (optional)

server.port=8080

OUTPUT :  
Welcome to Spring Boot 3 REST API!

2. Skill : spring REST using spring boot 3

Hands-on : Spring core-load country from spring configuration XML

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-xml-demo</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

</parent>

<dependencies>

<!-- Core Spring dependencies -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

</dependencies>

</project>

Country.java :  
package com.example.demo;

public class Country {

private String name;

private String capital;

public void setName(String name) {

this.name = name;

}

public void setCapital(String capital) {

this.capital = capital;

}

public void displayInfo() {

System.out.println("Country: " + name);

System.out.println("Capital: " + capital);

}

}

country-config.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.example.demo.Country">

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

<property name="capital" value="New Delhi"/>

</bean>

</beans>

DemoApplication.java :  
package com.example.demo;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

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

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

country.displayInfo();

}

}

OUTPUT :  
Country: India

Capital: New Delhi

3. Skill : spring REST using spring boot 3

Hands-on : Hello World RESTful Web Service

pom.xml :  
<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>hello-rest-api</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

</parent>

<dependencies>

<!-- Spring Web for REST API -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

</project>

DemoApplication.java :  
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}  
HelloController.java :  
package com.example.demo.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String sayHello() {

return "Hello, World!";

}

}

OUTPUT :  
Hello, World!

4. Skill : spring REST using spring boot 3

Hands-on : REST - Country web service

pom.xml :  
<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>country-rest-api</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

</parent>

<dependencies>

<!-- Spring Web for REST APIs -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

</project>

Country.java :  
package com.example.demo.model;

public class Country {

private String code;

private String name;

public Country(String code, String name) {

this.code = code;

this.name = 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; }

}

CountryController.java :  
package com.example.demo.controller;

import com.example.demo.model.Country;

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

import java.util.\*;

@RestController

@RequestMapping("/countries")

public class CountryController {

// Simulated data

private static final List<Country> countryList = List.of(

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

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

new Country("UK", "United Kingdom")

);

@GetMapping

public List<Country> getAllCountries() {

return countryList;

}

@GetMapping("/{code}")

public Country getCountryByCode(@PathVariable String code) {

return countryList.stream()

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

.findFirst()

.orElseThrow(() -> new RuntimeException("Country not found"));

}

}

DemoApplication.java :  
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}  
OUTPUT :  
GET <http://localhost:8080/countries>

Response :  
[

{ "code": "IN", "name": "India" },

{ "code": "US", "name": "United States" },

{ "code": "UK", "name": "United Kingdom" }

]

GET <http://localhost:8080/countries/IN>

Response :  
{ "code": "IN", "name": "India" }

5. Skill : spring REST using spring boot 3

Hands-on : REST - Get Country based on country code

pom.xml :  
<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>rest-country-service</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

</parent>

<dependencies>

<!-- Spring Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

</project>  
Country.java :  
package com.example.demo.model;

public class Country {

private String code;

private String name;

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters

public String getCode() {

return code;

}

public String getName() {

return name;

}

}  
CountryController.java :  
package com.example.demo.controller;

import com.example.demo.model.Country;

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

// Hardcoded data

private final List<Country> countryList = List.of(

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

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

new Country("UK", "United Kingdom")

);

// GET country by code

@GetMapping("/{code}")

public Country getCountryByCode(@PathVariable String code) {

return countryList.stream()

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

.findFirst()

.orElseThrow(() -> new RuntimeException("Country code not found: " + code));

}

}

DemoApplication.java :  
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

OUTPUT :  
GET <http://localhost:8080/countries/IN>

Response :  
{

"code": "IN",

"name": "India"

}

GET <http://localhost:8080/countries/XX>

Response :  
{

"timestamp": "2025-07-04T12:00:00",

"status": 500,

"error": "Internal Server Error",

"message": "Country code not found: XX",

...

}

6.Skill : spring REST using spring boot 3

Hands-on : Create authentication service that returns JWT

pom.xml :  
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>jwt-auth-api</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

</project>

AuthRequest.java :  
package com.example.jwt.model;

public class AuthRequest {

private String username;

private String password;

// Getters and Setters

}

AuthResponse.java :  
package com.example.jwt.model;

public class AuthResponse {

private String token;

public AuthResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

JwtService.java :  
package com.example.jwt.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Service;

import java.security.Key;

import java.util.Date;

@Service

public class JwtService {

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

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuer("Harshitha-App")

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 3600000)) // 1 hour

.signWith(key)

.compact();

}

}

AuthController.java :  
package com.example.jwt.controller;

import com.example.jwt.model.AuthRequest;

import com.example.jwt.model.AuthResponse;

import com.example.jwt.service.JwtService;

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

import org.springframework.security.authentication.\*;

import org.springframework.security.core.userdetails.User;

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

@RestController

@RequestMapping("/auth")

public class AuthController {

@Autowired

private AuthenticationManager authManager;

@Autowired

private JwtService jwtService;

@PostMapping("/login")

public AuthResponse login(@RequestBody AuthRequest request) {

Authentication authentication = authManager.authenticate(

new UsernamePasswordAuthenticationToken(request.getUsername(), request.getPassword())

);

User user = (User) authentication.getPrincipal();

String token = jwtService.generateToken(user.getUsername());

return new AuthResponse(token);

}

}

SecurityConfig.java :  
package com.example.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.\*;

import org.springframework.security.config.annotation.authentication.builders.\*;

import org.springframework.security.config.annotation.web.builders.\*;

import org.springframework.security.core.userdetails.\*;

import org.springframework.security.crypto.password.\*;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(

User.withUsername("user").password(passwordEncoder().encode("password")).roles("USER").build()

);

}

@Bean

public AuthenticationManager authenticationManager(HttpSecurity http, UserDetailsService uds) throws Exception {

return http.getSharedObject(AuthenticationManagerBuilder.class)

.userDetailsService(uds)

.passwordEncoder(passwordEncoder())

.and().build();

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

return http.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/auth/\*\*").permitAll()

.anyRequest().authenticated())

.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

DemoApplication.java :  
package com.example.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

Response :  
POST http://localhost:8080/auth/login

Content-Type: application/json

{

"username": "user",

"password": "password"

}

Response :

{

"token": "eyJhbGciOiJIUzI1NiJ9..."

}