**Assignment:**

**WEEK 4**

1. Create a Spring Web Project using Maven
2. 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-country</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<java.version>17</java.version>

<spring.version>6.1.5</spring.version>

<jakarta.servlet.version>6.0.0</jakarta.servlet.version>

</properties>

<dependencies>

<!-- Spring Core + MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- JSP / Servlet API -->

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<version>${jakarta.servlet.version}</version>

<scope>provided</scope>

</dependency>

<!-- JSTL (optional but handy for JSP) -->

<dependency>

<groupId>jakarta.servlet.jsp.jstl</groupId>

<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>

<version>3.0.0</version>

</dependency>

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>jakarta.servlet.jsp.jstl</artifactId>

<version>3.0.1</version>

</dependency>

</dependencies>

<!-- Use Jetty for quick runs: mvn jetty:run -->

<build>

<plugins>

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>12.0.11</version>

<configuration>

<webApp>

<contextPath>/spring-country</contextPath>

</webApp>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Spring XML configuration**

<?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

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

<!-- A very simple bean -->

<bean id="country" class="com.example.country.Country">

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

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

<property name="population" value="1393409038"/>

</bean>

</beans>

**Model (Country.java)**

package com.example.country;

public class Country {

private String name;

private String capital;

private long population;

// getters & setters

public String getName() { return name; }

public void setName(String n) { this.name = n; }

public String getCapital() { return capital; }

public void setCapital(String c) { this.capital = c; }

public long getPopulation() { return population; }

public void setPopulation(long p) { this.population = p; }

}

**Controller (CountryController.java)**

package com.example.country;

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

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

@Controller

public class CountryController {

private final ApplicationContext ctx;

@Autowired

public CountryController(ApplicationContext ctx) {

this.ctx = ctx; // root context loaded from spring-config.xml

}

@GetMapping("/country")

public String showCountry(Model model) {

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

model.addAttribute("country", country);

return "country"; // /WEB-INF/views/country.jsp

}

}

**Web deployment descriptor (web.xml)**

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

<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"

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

xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee

https://jakarta.ee/xml/ns/jakartaee/web-app\_6\_0.xsd"

version="6.0">

<display-name>spring-country</display-name>

<!-- Spring MVC dispatcher -->

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- Load our bean XML on startup -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-config.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

**Spring MVC servlet context (spring-servlet.xml)**

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

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

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

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

xsi:schemaLocation="

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

http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd

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

<!-- Scan for @Controller classes -->

<context:component-scan base-package="com.example.country"/>

<!-- JSP view resolver: /WEB-INF/views/\*.jsp -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/views/"/>

<property name="suffix" value=".jsp"/>

</bean>

</beans>

**View (/WEB-INF/views/country.jsp)**

<%@ page contentType="text/html; charset=UTF-8" language="java" %>

<%@ taglib uri="jakarta.tags.core" prefix="c" %>

<!DOCTYPE html>

<html>

<head>

<title>Country Details</title>

</head>

<body>

<h2>Country Details</h2>

<ul>

<li><strong>Name:</strong> ${country.name}</li>

<li><strong>Capital:</strong> ${country.capital}</li>

<li><strong>Population:</strong> ${country.population}</li>

</ul>

</body>

</html>

**Expected browser output**

Country Details

Name: India

Capital: New Delhi

Population: 1393409038

**Questions:**

1. Hello World RESTful Web Service
2. REST - Country Web Service
3. 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</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<java.version>17</java.version>

<spring.boot.version>3.3.1</spring.boot.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

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

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

</dependency>

<!-- Optional: dev‑time auto‑restart & test libs -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Application bootstrap**

package com.example.restcountry;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestCountryApplication {

public static void main(String[] args) {

SpringApplication.run(RestCountryApplication.class, args);

}

}

**Domain model**

package com.example.restcountry.model;

public record Country(String code, String name, String capital, long population) { }

**In‑memory repository**

package com.example.restcountry.repo;

import com.example.restcountry.model.Country;

import org.springframework.stereotype.Component;

import java.util.Collection;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

@Component

public class CountryRepository {

private final Map<String, Country> store = new ConcurrentHashMap<>();

public CountryRepository() {

// Seed a tiny data set

save(new Country("IN", "India", "New Delhi", 1\_393\_409\_038L));

save(new Country("US", "United States", "Washington, D.C.", 331\_002\_651L));

save(new Country("JP", "Japan", "Tokyo", 125\_836\_021L));

}

public Collection<Country> findAll() { return store.values(); }

public Country findByCode(String code) { return store.get(code.toUpperCase()); }

public void save(Country c) { store.put(c.code().toUpperCase(), c); }

}

**REST controllers**

**HelloController.java**

package com.example.restcountry.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String hello() {

return "Hello World";

}

}

**CountryController.java**

package com.example.restcountry.controller;

import com.example.restcountry.model.Country;

import com.example.restcountry.repo.CountryRepository;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.Collection;

@RestController

@RequestMapping("/countries")

public class CountryController {

private final CountryRepository repo;

public CountryController(CountryRepository repo) {

this.repo = repo;

}

// GET /countries → list all

@GetMapping

public Collection<Country> all() {

return repo.findAll();

}

// GET /countries/{code} → lookup by ISO code

@GetMapping("/{code}")

public ResponseEntity<Country> byCode(@PathVariable String code) {

Country c = repo.findByCode(code);

return (c == null)

? ResponseEntity.status(HttpStatus.NOT\_FOUND).build()

: ResponseEntity.ok(c);

}

}

**Authenticates a user with username + password and returns a signed JWT**

**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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>jwt-handson</artifactId>

<version>1.0.0</version>

<properties>

<java.version>17</java.version>

<spring.boot.version>3.3.1</spring.boot.version>

<jjwt.version>0.12.5</jjwt.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<!-- JJWT (sign & parse tokens) -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>${jjwt.version}</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>${jjwt.version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>${jjwt.version}</version>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Application bootstrap**

package com.example.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtHandsonApplication {

public static void main(String[] args) {

SpringApplication.run(JwtHandsonApplication.class, args);

}

}

**JWT utility (JwtUtil.java)**

package com.example.jwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import jakarta.annotation.PostConstruct;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

// 256‑bit secret key

private Key key;

private static final long EXPIRATION\_MS = 60 \* 60 \* 1000; // 1 hour

@PostConstruct

void init() {

key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

}

public String generateToken(String username) {

Date now = new Date();

Date exp = new Date(now.getTime() + EXPIRATION\_MS);

return Jwts.builder()

.setSubject(username)

.setIssuedAt(now)

.setExpiration(exp)

.signWith(key)

.compact();

}

public String extractUsername(String token) {

return Jwts.parserBuilder().setSigningKey(key).build()

.parseClaimsJws(token).getBody().getSubject();

}

}

**Security configuration**

**SecurityConfig.java**

package com.example.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(org.springframework.security.config.annotation.web.builders.HttpSecurity http,

AuthenticationManager authManager,

JwtAuthenticationFilter jwtFilter) throws Exception {

return http

.csrf(cs -> cs.disable())

.sessionManagement(sm -> sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

.authorizeHttpRequests(auth -> auth

.requestMatchers(HttpMethod.POST, "/authenticate").permitAll()

.anyRequest().authenticated())

.addFilter(jwtFilter)

.authenticationManager(authManager)

.build();

}

@Bean

public AuthenticationManager authManager(org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder builder)

throws Exception {

builder.inMemoryAuthentication()

.withUser("alice")

.password(passwordEncoder().encode("password"))

.roles("USER");

return builder.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

**JwtAuthenticationFilter.java**

package com.example.jwt.config;

import com.example.jwt.util.JwtUtil;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.springframework.lang.NonNull;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import org.springframework.security.authentication.AuthenticationManager;

import java.io.IOException;

import java.util.Collections;

public class JwtAuthenticationFilter extends BasicAuthenticationFilter {

private final JwtUtil jwtUtil;

public JwtAuthenticationFilter(AuthenticationManager authManager, JwtUtil jwtUtil) {

super(authManager);

this.jwtUtil = jwtUtil;

}

@Override

protected void doFilterInternal(@NonNull HttpServletRequest request,

@NonNull HttpServletResponse response,

@NonNull FilterChain chain)

throws IOException, ServletException {

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

if (header != null && header.startsWith("Bearer ")) {

try {

String token = header.substring(7);

String username = jwtUtil.extractUsername(token);

var auth = new UsernamePasswordAuthenticationToken(username, null, Collections.emptyList());

SecurityContextHolder.getContext().setAuthentication(auth);

} catch (Exception ignored) { }

}

chain.doFilter(request, response);

}

}

The filter is declared as a bean via the constructor parameter in SecurityConfig.

**Authentication controller**

package com.example.jwt.controller;

import com.example.jwt.util.JwtUtil;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.AuthenticationException;

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

import java.util.Map;

@RestController

public class AuthController {

private final AuthenticationManager authManager;

private final JwtUtil jwtUtil;

public AuthController(AuthenticationManager authManager, JwtUtil jwtUtil) {

this.authManager = authManager;

this.jwtUtil = jwtUtil;

}

@PostMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestBody AuthRequest request) {

try {

var authentication = authManager.authenticate(

new UsernamePasswordAuthenticationToken(request.username(), request.password()));

String token = jwtUtil.generateToken(authentication.getName());

return ResponseEntity.ok(Map.of("token", token));

} catch (AuthenticationException ex) {

return ResponseEntity.status(401).body(Map.of("error", "Invalid credentials"));

}

}

// Simple DTO (Java 17 record)

public record AuthRequest(String username, String password) { }

}