Module 4 solutions

**Superset ID:6403222**

Spring Boot Project

# 1. Spring Web Project using Maven

This is a basic Spring Boot web application created using Maven.

## pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.example</groupId>  
 <artifactId>webdemo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
 <name>Spring Web Demo</name>  
 <description>Demo project for Spring Boot Web</description>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.1.2</version>  
 <relativePath/>  
 </parent>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 </dependencies>  
</project>

## HomeController.java

package com.example.webdemo.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HomeController {  
  
 @GetMapping("/")  
 public String home() {  
 return "Welcome to Spring Web Project!";  
 }  
}

# 2. Spring Core – Load Country from XML Configuration

This Spring Core example loads a Country object from an XML configuration using Spring's ApplicationContext.

## Country.java

package com.example.core.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; }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

## beans.xml

<beans xmlns="http://www.springframework.org/schema/beans" ... >  
 <bean id="country" class="com.example.core.model.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
</beans>

## MainApp.java

public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");  
 Country country = (Country) context.getBean("country");  
 System.out.println(country);  
 }  
}

# 3. Hello World RESTful Web Service

## HelloController.java

@RestController  
public class HelloController {  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello, World!";  
 }  
}

# 4. REST - Country Web Service

## Country.java

public class Country {  
 private String code;  
 private String name;  
  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = name;  
 }  
 public String getCode() { return code; }  
 public String getName() { return name; }  
}

## CountryController.java

@RestController  
@RequestMapping("/api/countries")  
public class CountryController {  
  
 private Map<String, Country> countries = new HashMap<>();  
  
 public CountryController() {  
 countries.put("IN", new Country("IN", "India"));  
 countries.put("US", new Country("US", "United States"));  
 }  
  
 @GetMapping  
 public Collection<Country> getAllCountries() {  
 return countries.values();  
 }  
  
 @PostMapping  
 public String addCountry(@RequestBody Country country) {  
 countries.put(country.getCode(), country);  
 return "Country added.";  
 }  
}

# 5. REST - Get Country based on country code

## Endpoint Example

@GetMapping("/{code}")  
public Country getCountryByCode(@PathVariable String code) {  
 return countries.getOrDefault(code.toUpperCase(), new Country(code, "Not Found"));  
}

# 6. Authentication Service that returns JWT

This project authenticates users and returns a JWT token using Spring Security.

## AuthController.java

@RestController  
@RequestMapping("/auth")  
public class AuthController {  
  
 @PostMapping("/login")  
 public ResponseEntity<?> authenticate(@RequestBody AuthRequest request) {  
 // Validate credentials, generate JWT  
 String token = jwtService.generateToken(request.getUsername());  
 return ResponseEntity.ok(new AuthResponse(token));  
 }  
}

## JWT Utility

public class JwtService {  
 private final String SECRET = "secret\_key";  
  
 public String generateToken(String username) {  
 return Jwts.builder()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.currentTimeMillis() + 86400000))  
 .signWith(SignatureAlgorithm.HS256, SECRET)  
 .compact();  
 }  
}