**Module 7 - Spring REST using Spring Boot 3**

**Hands on 1**

**Create a Spring Web Project using Maven**

**Objective:**

To create a Spring Boot web application using Maven, explore the project structure, and implement a simple REST controller.

**Steps Performed:**

**1. Project Creation:**

* Visited: <https://start.spring.io>
* Group: com.cognizant
* Artifact: spring-learn
* Dependencies selected:
  + Spring Web
  + Spring Boot DevTools
* Project downloaded as a .zip and extracted to local workspace.

**2. Import in Eclipse:**

* Eclipse → File → Import → Maven → Existing Maven Projects → selected spring-learn folder → Finish
* Maven dependencies loaded successfully.

**3. Project Structure:**

* src/main/java → Contains main application and controller.
* src/main/resources → Contains application.properties.
* src/test/java → For writing unit tests.
* pom.xml → Maven dependency configuration file.

**4. Application Code:**

**SpringLearnApplication.java**:

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("STARTING THE APPLICATION");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("APPLICATION STARTED SUCCESSFULLY");

}

}

**5. REST Endpoint Implementation:**

**HelloController.java**:

package com.cognizant.spring\_learn;

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 from Cognizant SpringLearn App!";

}

}

**6. Configuration:**

**application.properties**:

server.port=9292

**Build and Execution:**

* Application built and run inside Eclipse.
* Default port 8080 was in use, changed to 9292.
* Browser accessed: http://localhost:9292/hello

**Output:**

* **Console Log**:

... INFO SpringLearnApplication : APPLICATION STARTED SUCCESSFULLY

... INFO Tomcat started on port 9191

**Browser Output**:

Hello from Cognizant SpringLearn App!

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion:**

The Spring Boot web project was successfully created using Maven, imported into Eclipse, and executed on a custom port. The /hello REST endpoint worked as expected, confirmed by both console logs and browser output. This hands-on helped me understand basic Spring Boot setup, project structure, and REST controller implementation.

**Hands-on Task 4:**

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

**Objective:**

To learn how to configure and load a Spring bean from an XML file using Spring Core. This involves defining a Country bean, loading it using ApplicationContext, and logging the details.

**Step-by-Step Implementation:**

**Create the Country class**

**src/main/java/com/cognizant/springlearn/Country.java**

package com.cognizant.springlearn;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Inside Country Constructor.");

}

public String getCode() {

System.out.println("Inside getCode()");

return code;

}

public void setCode(String code) {

System.out.println("Inside setCode()");

this.code = code;

}

public String getName() {

System.out.println("Inside getName()");

return name;

}

public void setName(String name) {

System.out.println("Inside setName()");

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**Create Spring Configuration File: country.xml**

**src/main/resources/country.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.cognizant.springlearn.Country">

<property name="code" value="IN"/>

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

</bean>

</beans>

**Create Main Class: Main.java**

**src/main/java/com/cognizant/springlearn/Main.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

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

public static void main(String[] args) {

LOGGER.info("STARTING THE APPLICATION");

displayCountry();

LOGGER.info("APPLICATION ENDED");

}

public static void displayCountry() {

System.out.println("Inside displayCountry()");

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

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

System.out.println("Country from XML: " + country);

}

}

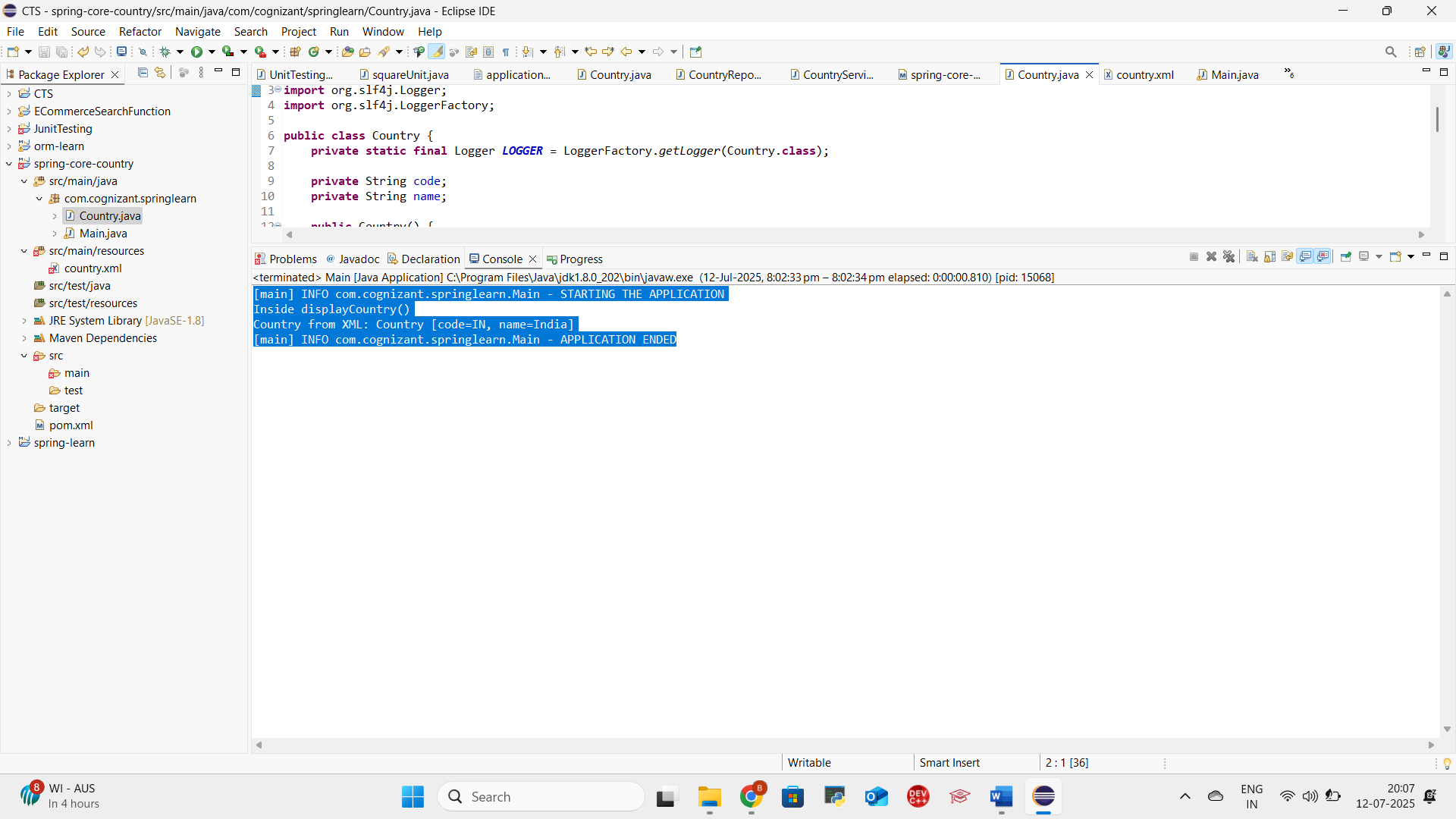
**Output**

[main] INFO com.cognizant.springlearn.Main - STARTING THE APPLICATION

Inside displayCountry()

Country from XML: Country [code=IN, name=India]

[main] INFO com.cognizant.springlearn.Main - APPLICATION ENDED



**Conclusion**

Successfully implemented a Spring Core application using XML-based configuration.  
A Country bean was created and loaded using ApplicationContext from country.xml.  
The bean's data was printed, validating proper Spring XML configuration.

**Hello World RESTful Web Service**

**Objective**

Create a simple RESTful web service using Spring Boot that returns the message Hello World!! at the endpoint /hello.

**Step-by-Step Implementation**

**Step 1: Project Setup**

Used the existing Spring Boot project (spring-learn) created earlier in previous hands-ons.

**Step 2: Create a Controller Package**

* Right-click on src/main/java
* Navigate to your base package: com.cognizant.spring\_learn
* Create a subpackage: controller

**Step 3: Create the HelloController Class**

Create a new Java class inside com.cognizant.spring\_learn.controller named **HelloController.java** and paste the following code:

package com.cognizant.spring\_learn.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;

}

}

**Step 4: Update the Port (Optional)**

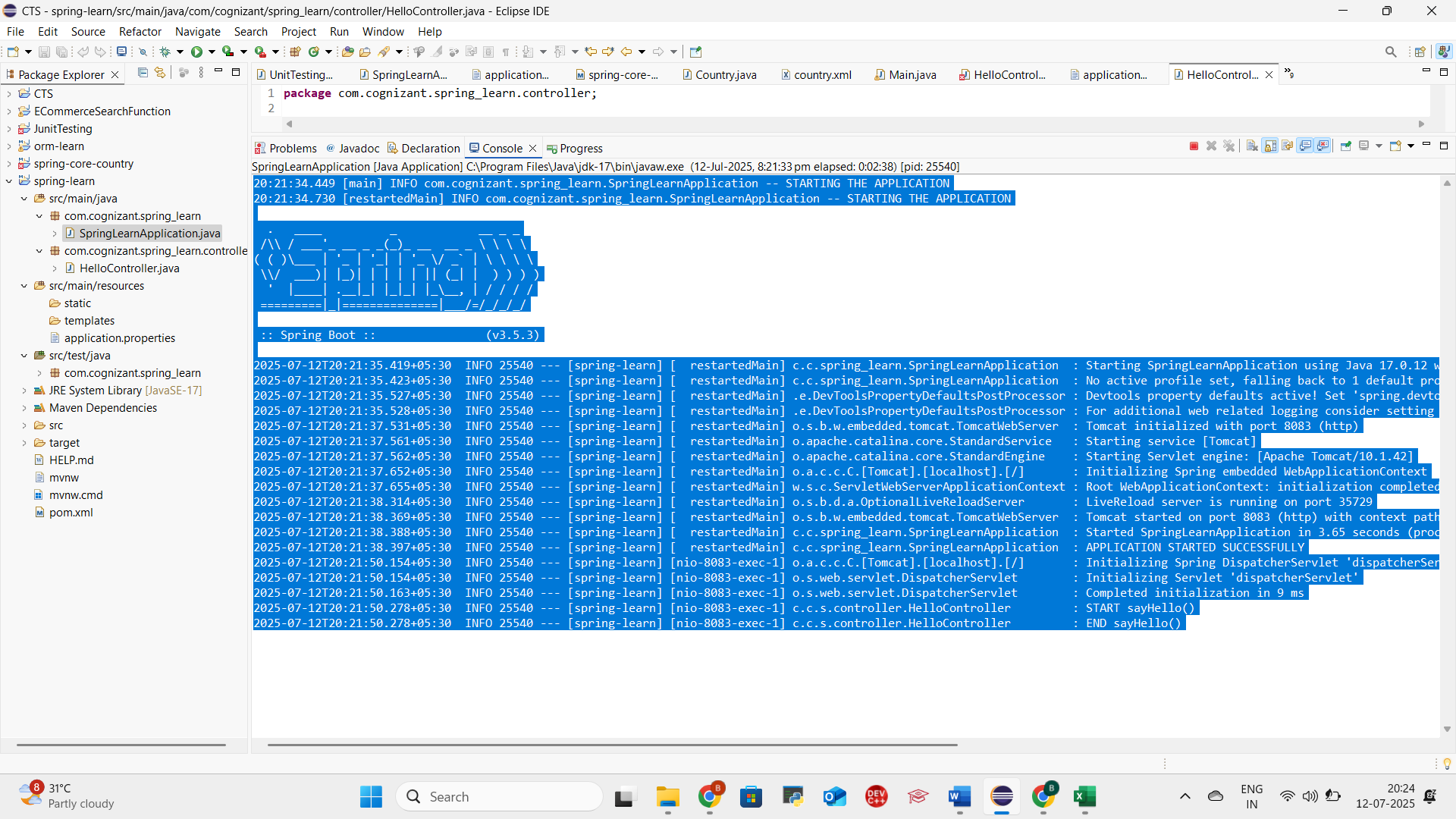
To avoid port conflicts, you can set a custom port like 8083.

In src/main/resources/application.properties

add:

server.port=8083

**OUTPUT:**



A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion**

In this hands-on, we successfully:

* Created a RESTful web service using Spring Boot
* Configured a GET endpoint /hello
* Returned a static string response
* Verified output through both browser and Postman

**REST - Country Web Service**

**Objective**

Create a RESTful web service using Spring Boot that returns the country details (India) configured in an XML file.

**Step-by-Step Implementation**

**CountryController.java (in controller package)**

package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.CountryService;  
import com.cognizant.springlearn.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 // GET: /country → returns India  
 @GetMapping("/country")  
 public Country getCountryIndia() {  
 return new Country("IN", "India");  
 }  
  
 // GET: /countries → returns all countries  
 @GetMapping("/countries")  
 public List<Country> getAllCountries() {  
 return countryService.getAllCountries();  
 }  
  
 // GET: /countries/{code} → returns a country by code  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code) throws CountryNotFoundException {  
 return countryService.getCountry(code);  
 }  
}

**Country.java (in model package)**

package com.cognizant.springlearn.model;  
  
public class Country {  
 private String code;  
 private String name;  
  
 public Country() {}  
  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = 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 (in service package)**

package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.exception.CountryNotFoundException;  
import org.springframework.stereotype.Service;  
  
import java.util.Arrays;  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 private static final List<Country> *countries* = Arrays.*asList*(  
 new Country("IN", "India"),  
 new Country("US", "United States"),  
 new Country("JP", "Japan"),  
 new Country("DE", "Germany")  
 );  
  
 public Country getCountry(String code) throws CountryNotFoundException {  
 return *countries*.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElseThrow(() -> new CountryNotFoundException("Country not found"));  
 }  
  
 public List<Country> getAllCountries() {  
 return *countries*;  
 }  
}

**CountryNotFoundException.java (in service.exception package)**

package com.cognizant.springlearn.service.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value = HttpStatus.*NOT\_FOUND*, reason = "Country not found")  
public class CountryNotFoundException extends Exception {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**Now, run the SPringLearnApplication.java**

**Then, go to any browser and type,**

1. [**http://localhost:8080/countries**](http://localhost:8080/countries) **(to display all countries)**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. [**http://localhost:8080/countries/in**](http://localhost:8080/countries/in) **(to display specific country)**

A black screen with a blue line

AI-generated content may be incorrect.

**REST - Get country based on country code**

**Objective**

Create a Spring Boot REST API that returns a specific country by its code, using data loaded from country.xml.

Step-by-Step Implementation

**Step 1: Update country.xml with the list of countries**

**Path: src/main/resources/country.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

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

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<bean class="com.cognizant.spring\_learn.model.Country">

<property name="code" value="IN"/>

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

</bean>

<bean class="com.cognizant.spring\_learn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.spring\_learn.model.Country">

<property name="code" value="DE"/>

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

</bean>

<bean class="com.cognizant.spring\_learn.model.Country">

<property name="code" value="JP"/>

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Step 2: Create the Service Class CountryService**

Path: com.cognizant.spring\_learn.service.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> countryList = context.getBean("countryList", List.class);

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

**Step 3: Update/Create CountryController**

Path: com.cognizant.spring\_learn.controller.CountryController

package com.cognizant.spring\_learn.controller;

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

import com.cognizant.spring\_learn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

LOGGER.info("START getCountry() with code: {}", code);

Country country = countryService.getCountry(code);

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

return country;

}

}

**Testing the API**

**Using Browser or Postman**

**URL:**

http://localhost:8083/countries/in

**Output:**

{

"code": "IN",

"name": "India"

}

<http://localhost:8080/countries/in>

**A black rectangular object with a black stripe

AI-generated content may be incorrect.**

<http://localhost:8080/countries/us>

A black rectangular object with a black border

AI-generated content may be incorrect.

**JWT-handson**

**Create authentication service that returns JWT**

**STEP BY STEP IMPLEMENTATION:**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<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>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring-learn</name>  
 <description>spring-learn</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Spring Security -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <!-- JWT Library -->  
 <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> <!-- or jjwt-gson -->  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**SecurityConfig.java (in package security)**

package com.cognizant.springlearn.security;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
  
@Configuration  
public class SecurityConfig {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SecurityConfig.class);  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public InMemoryUserDetailsManager userDetailsService() {  
 UserDetails admin = User  
 .*withUsername*("admin")  
 .password(passwordEncoder().encode("pwd"))  
 .roles("ADMIN")  
 .build();  
  
 UserDetails user = User  
 .*withUsername*("user")  
 .password(passwordEncoder().encode("pwd"))  
 .roles("USER")  
 .build();  
  
 return new InMemoryUserDetailsManager(admin, user);  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .httpBasic()  
 .and()  
 .authorizeHttpRequests()  
 .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")  
 .anyRequest().authenticated();  
  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

**AuthenticationController.java (in the package controller)**

package com.cognizant.springlearn.controller;  
  
import io.jsonwebtoken.JwtBuilder;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RestController;  
import io.jsonwebtoken.security.Keys;  
  
  
import java.util.Base64;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;  
  
import io.jsonwebtoken.security.Keys;  
import java.security.Key;  
  
@RestController  
public class AuthenticationController {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(AuthenticationController.class);  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 *LOGGER*.info("START - authenticate()");  
 *LOGGER*.debug("Authorization Header: {}", authHeader);  
  
 String user = getUser(authHeader);  
 String token = generateJwt(user);  
  
 Map<String, String> map = new HashMap<>();  
 map.put("token", token);  
  
 *LOGGER*.info("END - authenticate()");  
 return map;  
 }  
  
 private String getUser(String authHeader) {  
 String encodedCredentials = authHeader.replace("Basic ", "");  
 byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);  
 String decodedCredentials = new String(decodedBytes); // "user:pwd"  
 *LOGGER*.debug("Decoded credentials: {}", decodedCredentials);  
 return decodedCredentials.split(":")[0]; // return "user"  
 }  
  
 private String generateJwt(String user) {  
 String secret = "my-secret-key-that-is-long-enough-123456"; // At least 32 chars  
 Key key = Keys.*hmacShaKeyFor*(secret.getBytes());  
  
 return Jwts.*builder*()  
 .setSubject(user)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 minutes  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
}

**Run the SpringLearnApplication.java and go to browser and type,**

[**http://localhost:8080/authenticate**](http://localhost:8080/authenticate)

**You will get a prompt to enter credentials**,

A screenshot of a computer

AI-generated content may be incorrect.

**I have entered “user” for username and “pwd” for password**

A screenshot of a login screen

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**We have fetched the JWT successfully!**