**WEEK-04**

**Date:13-07-2025**

1. **spring-rest-handson:**

**Create a Spring Web Project using Maven:  
  
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) {

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("SpringLearnApplication started");

}

}  
  
**pom.xml:**

<dependencies>

<dependency>

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

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

</dependency>

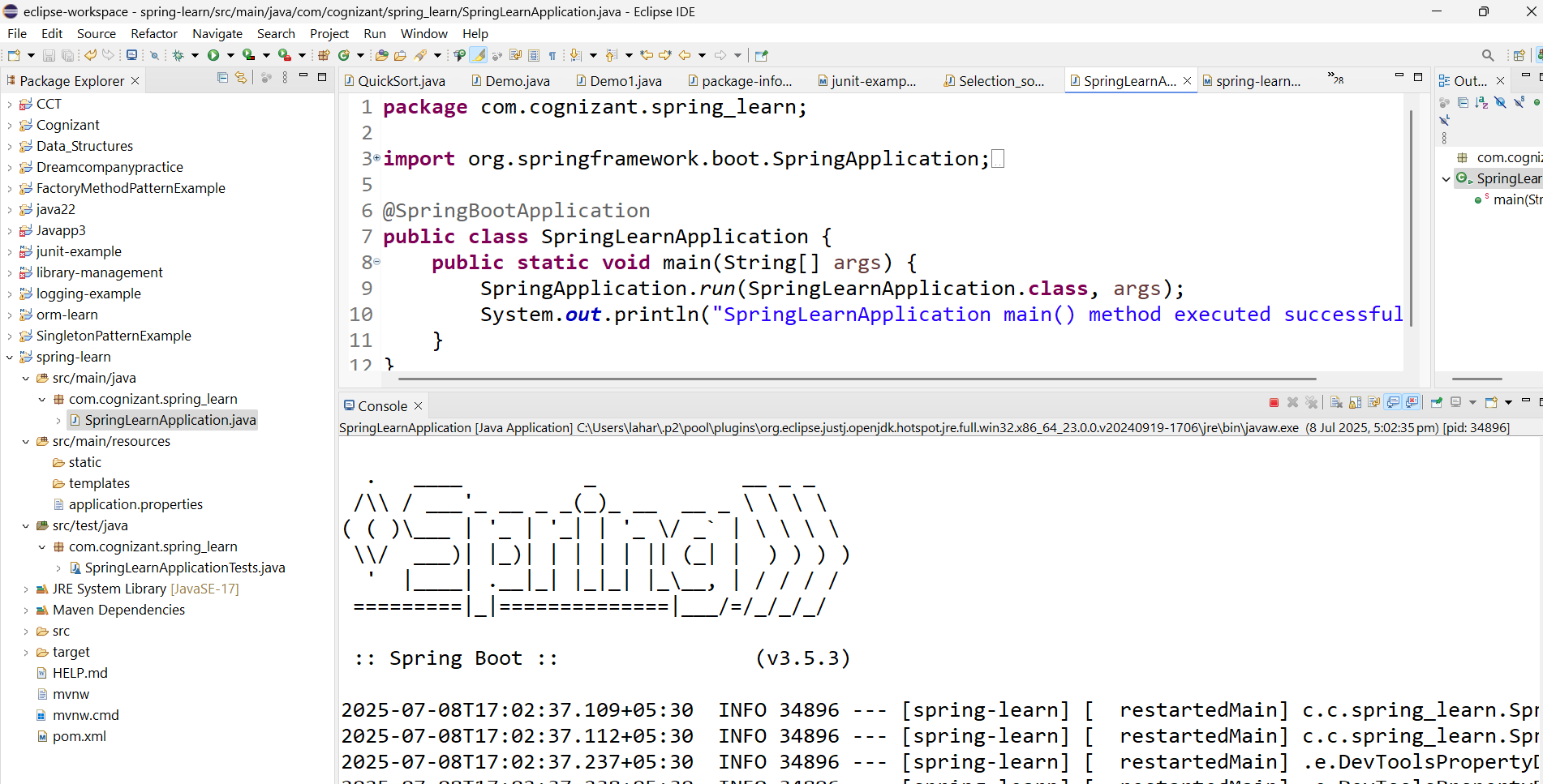
<dependency>

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

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

<scope>runtime</scope>

</dependency>

</dependencies>  
  
**Output:  
**

**Spring Core – Load Country from Spring Configuration XML:  
SpringLearnApplication.java:**

**package** com.cognizant.spring\_learn;

**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** (Exception e) {

System.***out***.println("Error parsing date: " + e.getMessage());

}

}

}

**date-format.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"*>

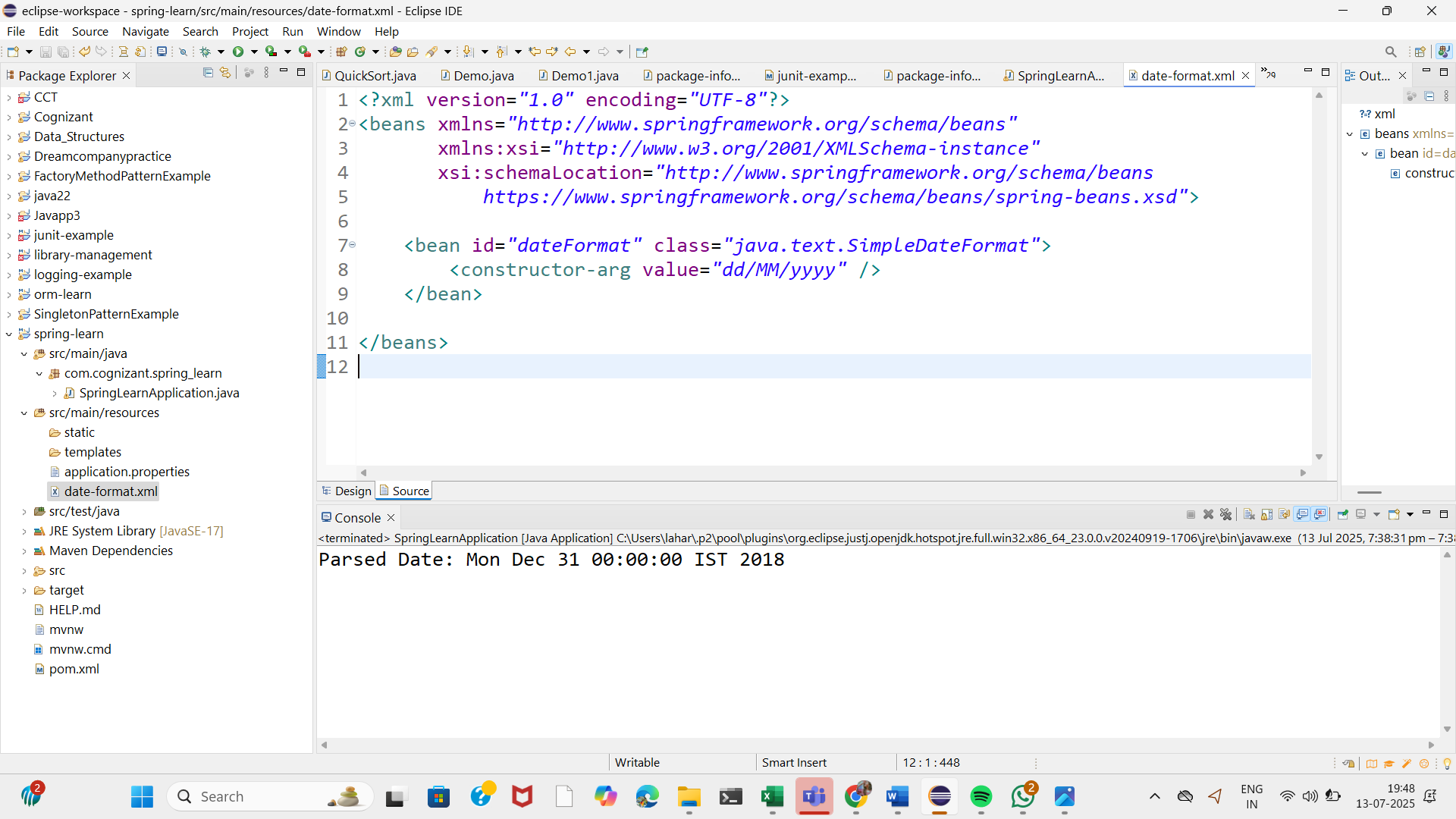
<bean id=*"dateFormat"* class=*"java.text.SimpleDateFormat"*>

<constructor-arg value=*"dd/MM/yyyy"* />

</bean>

</beans>

**Output:**



**2. spring-rest-handson:**

**Hello World RESTful Web Service:**

HelloController

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 response = "Hello World!!";

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

return response;

}

}

SpringLearnApplication.java

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Boot Hello World</description>

<parent>

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

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

<version>3.2.5</version> <!-- or your chosen version -->

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

</parent>

<properties>

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

</properties>

<!-- You MUST put dependencies inside this block -->

<dependencies>

<!-- Your Web dependency -->

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Output:**

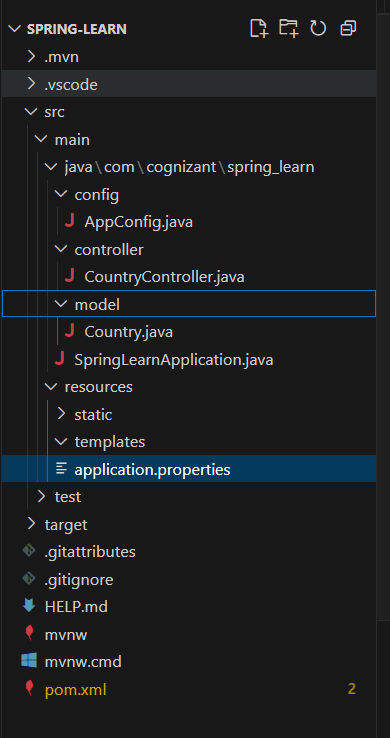
**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**REST - Country Web Service:**

****

**Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

    private String name;

    private String capital;

    private long population;

    private String currency;

    public Country() {

    }

    public Country(String name, String capital, long population, String currency) {

        this.name = name;

        this.capital = capital;

        this.population = population;

        this.currency = currency;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getCapital() {

        return capital;

    }

    public void setCapital(String capital) {

        this.capital = capital;

    }

    public long getPopulation() {

        return population;

    }

    public void setPopulation(long population) {

        this.population = population;

    }

    public String getCurrency() {

        return currency;

    }

    public void setCurrency(String currency) {

        this.currency = currency;

    }

    @Override

    public String toString() {

        return "Country{" +

                "name='" + name + '\'' +

                ", capital='" + capital + '\'' +

                ", population=" + population +

                ", currency='" + currency + '\'' +

                '}';

    }

}

**CountryController.java**

package com.cognizant.spring\_learn.controller;

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

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class CountryController {

    @GetMapping("/country")

    public Map<String, String> getCountry() {

        Map<String, String> country = new HashMap<>();

        country.put("code", "IN");

        country.put("name", "India");

        return country;

    }

    @GetMapping("/")

    public String home() {

        return "Spring Boot is working! Try /country endpoint";

    }

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringLearnApplication.class, args);

    }

}

**AppConfig.java:**

package com.cognizant.spring\_learn.config;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

    @Bean

    public Country indiaCountry() {

        Country country = new Country();

        country.setName("India");

        country.setCapital("New Delhi");

        country.setPopulation(1380000000L);

        country.setCurrency("INR");

        return country;

    }

    @Bean

    public Country usaCountry() {

        Country country = new Country();

        country.setName("United States");

        country.setCapital("Washington D.C.");

        country.setPopulation(331000000L);

        country.setCurrency("USD");

        return country;

    }

}

**country.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>3.2.0</version>

        <relativePath/>

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>spring-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>spring-learn</name>

    <description>Demo project for Spring Boot</description>

    <properties>

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

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

        <!-- This dependency includes spring-context which has ImportResource -->

        <dependency>

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

            <artifactId>spring-boot-starter</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>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**Output:**

REST - Get country based on country code

****

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

**Country.java**

package com.cognizant.spring\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

    @Id

    @Column(name = "code")

    private String code;

    @Column(name = "name")

    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;

    }

    @Override

    public String toString() {

        return "Country{" +

                "code='" + code + '\'' +

                ", name='" + name + '\'' +

                '}';

    }

}

**CountryService.java**

package com.cognizant.spring\_learn.service;

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

import com.cognizant.spring\_learn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service

public class CountryService {

    @Autowired

    private CountryRepository countryRepository;

    public Optional<Country> getIndia() {

        return countryRepository.findByCode("IN");

    }

    public Optional<Country> getIndiaByCustomQuery() {

        return countryRepository.findIndia();

    }

    public void printIndiaDetails() {

        Optional<Country> india = getIndia();

        if (india.isPresent()) {

            Country country = india.get();

            System.out.println("Country Code: " + country.getCode());

            System.out.println("Country Name: " + country.getName());

            System.out.println("Full Details: " + country.toString());

        } else {

            System.out.println("India not found in database");

        }

    }

}

**CountryRepository.java**

package com.cognizant.spring\_learn.repository;

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

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

    Optional<Country> findByCode(String code);

    @Query("SELECT c FROM Country c WHERE c.code = 'IN'")

    Optional<Country> findIndia();

    @Query(value = "SELECT \* FROM country WHERE code = 'IN'", nativeQuery = true)

    Optional<Country> findIndiaByNativeQuery();

}

**CountryController.java**

package com.cognizant.spring\_learn.controller;

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

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

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

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/api/countries")

public class CountryController {

    @Autowired

    private CountryService countryService;

    @GetMapping("/india")

    public ResponseEntity<Country> getIndia() {

        return countryService.getIndia()

                .map(country -> ResponseEntity.ok(country))

                .orElse(ResponseEntity.notFound().build());

    }

    @GetMapping("/india/print")

    public ResponseEntity<String> printIndiaDetails() {

        countryService.printIndiaDetails();

        return ResponseEntity.ok("India details printed to console");

    }

    @GetMapping("/india/info")

    public ResponseEntity<String> getIndiaInfo() {

        return countryService.getIndia()

                .map(country -> ResponseEntity.ok(

                    "Code: " + country.getCode() + ", Name: " + country.getName()))

                .orElse(ResponseEntity.notFound().build());

    }

}

**SpringLearnApplication.java:**

package com.cognizant.spring\_learn;

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

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.core.annotation.Order;

@SpringBootApplication

public class SpringLearnApplication implements CommandLineRunner {

    @Autowired

    private CountryService countryService;

    public static void main(String[] args) {

        SpringApplication.run(SpringLearnApplication.class, args);

    }

    @Override

    @Order(2) // Run after DataInitializer

    public void run(String... args) throws Exception {

        Thread.sleep(1000);

        System.out.println("=== INDIA COUNTRY DETAILS ===");

        countryService.printIndiaDetails();

        System.out.println("==============================");

    }

}

**Data.sql:**

    INSERT INTO country (code, name) VALUES ('IN', 'India');

    INSERT INTO country (code, name) VALUES ('US', 'United States');

    INSERT INTO country (code, name) VALUES ('UK', 'United Kingdom');

**country.xml**

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

<countries>

    <country>

        <code>IN</code>

        <name>India</name>

    </country>

    <country>

        <code>US</code>

        <name>United States</name>

    </country>

    <country>

        <code>UK</code>

        <name>United Kingdom</name>

    </country>

    <country>

        <code>CA</code>

        <name>Canada</name>

    </country>

    <country>

        <code>AU</code>

        <name>Australia</name>

    </country>

    <country>

        <code>DE</code>

        <name>Germany</name>

    </country>

    <country>

        <code>FR</code>

        <name>France</name>

    </country>

    <country>

        <code>JP</code>

        <name>Japan</name>

    </country>

    <country>

        <code>BR</code>

        <name>Brazil</name>

    </country>

    <country>

        <code>CN</code>

        <name>China</name>

    </country>

</countries>

**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>3.1.0</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>REST Country Code Project</description>

    <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-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

        <dependency>

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

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

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**Output:**

****

**5. JWT-handson**

**Create authentication service that returns JWT**

**pom.xml**

<!-- JWT Dependency -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

**AuthenticationController.java**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import javax.servlet.http.HttpServletRequest;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

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

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

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

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

String token = jwtUtil.generateToken(username);

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

}

}

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

}

}

**JwtUtil.java**

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secret = "mySecretKey";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 mins

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

**SecurityConfig.java**

package com.example.jwtauth.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

**Output:**

A black text on a white background

AI-generated content may be incorrect.