**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>

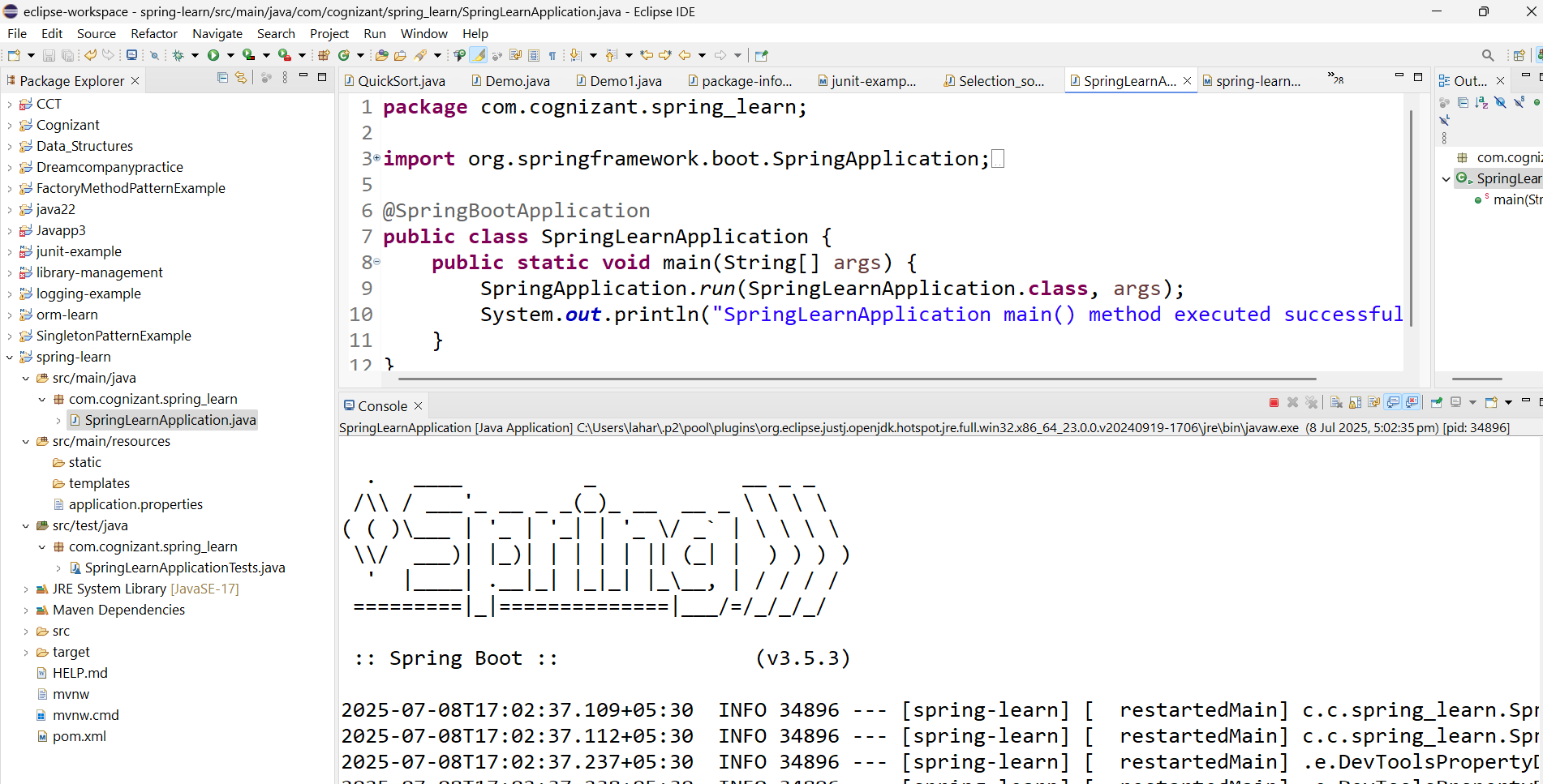
<dependency>

<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"*>

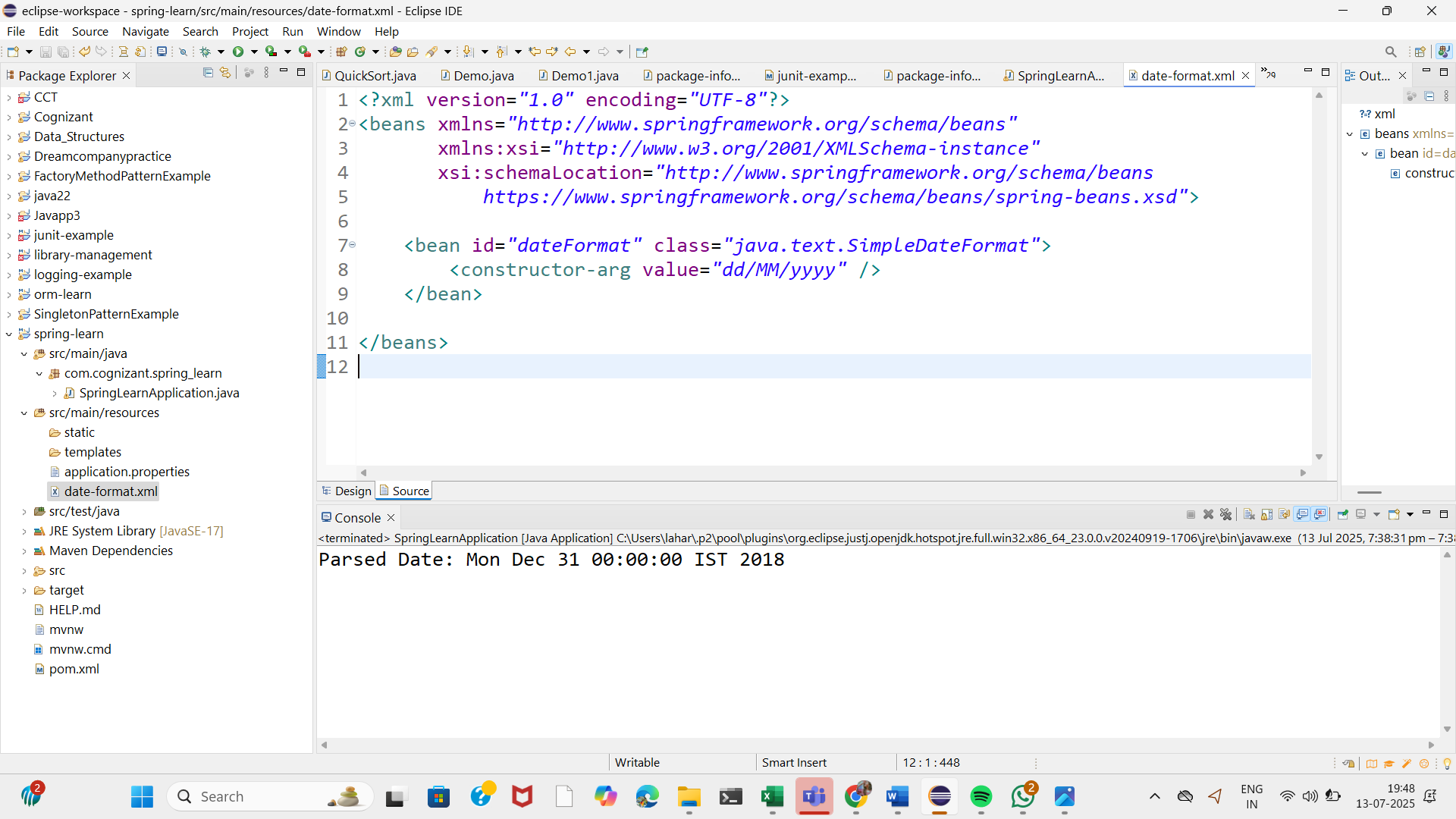
<bean id=*"dateFormat"* class=*"java.text.SimpleDateFormat"*>

<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 code;

private String name;

public Country() {}

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.cognizant.spring\_learn.controller;

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

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return country;

}

}

**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);

}

}

**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.spring\_learn.model.Country">

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

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

</bean>

</beans>

**Output:**

REST - Get country based on country code

A screenshot of a computer

AI-generated content may be incorrect.

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

**Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String 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;

}

}

**CountryService.java**

package com.cognizant.spring\_learn.service;

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

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

import org.springframework.core.io.ClassPathResource;

import org.springframework.oxm.jaxb.Jaxb2Marshaller;

import org.springframework.stereotype.Service;

import javax.xml.transform.stream.StreamSource;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) throws Exception {

Jaxb2Marshaller marshaller = new Jaxb2Marshaller();

marshaller.setClassesToBeBound(CountryList.class); // Wrap class

CountryList countryList = (CountryList) marshaller.unmarshal(

new StreamSource(new ClassPathResource("country.xml").getInputStream()));

return countryList.getCountryList().stream()

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

.findFirst()

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

}

}

**CountryList.java**

package com.cognizant.spring\_learn.model;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

import java.util.List;

@XmlRootElement(name = "countries")

public class CountryList {

private List<Country> countryList;

@XmlElement(name = "country")

public List<Country> getCountryList() {

return countryList;

}

public void setCountryList(List<Country> countryList) {

this.countryList = countryList;

}

}

**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.web.bind.annotation.\*;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) throws Exception {

return countryService.getCountry(code);

}

}

**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>

</countries>

**pom.xml**

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring OXM for XML to Java -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

</dependency>

<!-- JAXB (for Java 11+) -->

<dependency>

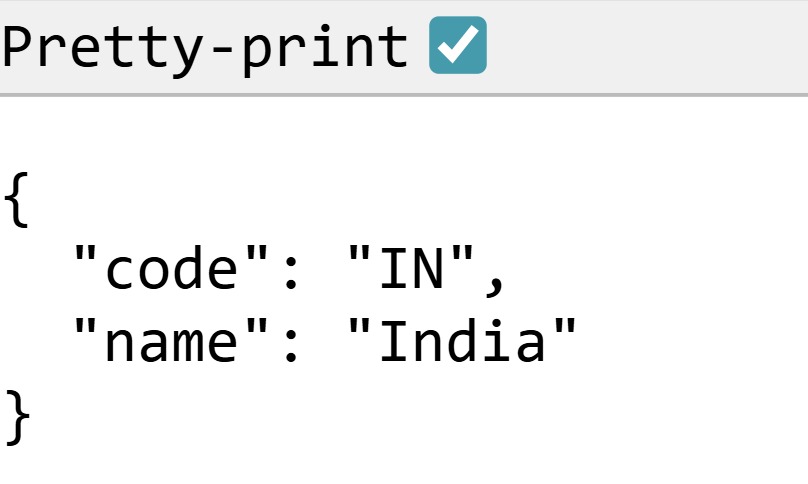
<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

</dependency>

</dependencies>

**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.