# ****WEEK4 – Spring REST Using Spring Boot****

## ****Hands-on 1: Create a Spring Web Project using Maven****

### Scenario:

Create a Spring Boot project using Maven with basic Web support.

### Steps:

1. Visit [https://start.spring.io/](https://start.spring.io/" \t "_new)
2. Set **Group** as com.cognizant and **Artifact** as spring-learn.
3. Add dependencies: **Spring Boot DevTools**, **Spring Web**.
4. Download and extract the ZIP into Eclipse workspace.
5. Build using Maven with proxy settings (if required).
6. Import into Eclipse as an existing Maven project.
7. Add log statements in SpringLearnApplication.java to confirm main() runs.
8. Run the application from Eclipse.

### SME Notes:

### Explain src/main/java, src/test/java, resources folders.

### 2. Walkthrough SpringLearnApplication.java and @SpringBootApplication.

3. Show pom.xml dependencies and hierarchy view.

**Code: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>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Boot Learning Project</description>

<parent>

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

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

<version>3.1.0</version>

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

</parent>

<properties>

<java.version>21</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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Code: SpringLearnApplication.java**

### package com.cognizant.springlearn;

### 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("START");

### SpringApplication.run(SpringLearnApplication.class, args);

### LOGGER.info("END"); } }}

**Code:CountryNotFoundException.java**

### package com.cognizant.springlearn.exception;

### public class CountryNotFoundException extends Exception {

### public CountryNotFoundException(String message) {

### super(message); } }

### Code:application.properties

### # Server configuration

### server.port=8083

### # Logging level

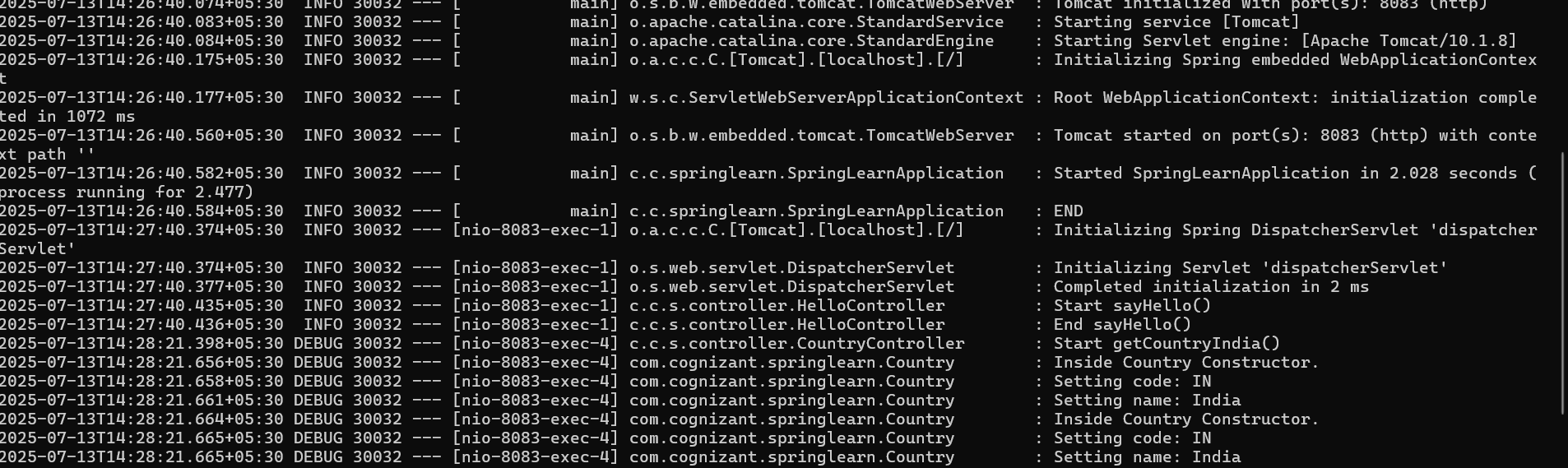
### logging.level.root=INFO

### logging.level.com.cognizant=DEBUG

### # Optional banner off

### spring.main.banner-mode=off

### Output:



## ****Hands-on 4: Spring Core – Load Country from Spring XML Configuration****

### Scenario:

Load and display a country object configured via Spring XML.

### Steps:

1. Create a Spring bean in country.xml for a country like India.
2. Define Country class with fields, constructor, logs in getters/setters.
3. Use ApplicationContext and context.getBean() to load the bean.
4. Add displayCountry() in SpringLearnApplication.java.
5. Invoke it in main() method.

### SME Notes:

1. Explain <bean>, id, class, <property>, name, value.

2. Discuss ApplicationContext, ClassPathXmlApplicationContext.

3. What happens during context.getBean()?

### ****Code: 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">

### <!-- Single country bean -->

### <bean id="country" class="com.cognizant.springlearn.Country">

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

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

### </bean>

### <!-- List of countries -->

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

### <constructor-arg> <list>

### <bean class="com.cognizant.springlearn.Country">

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

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

### <bean class="com.cognizant.springlearn.Country">

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

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

### <bean class="com.cognizant.springlearn.Country">

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

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

### </bean>

### <bean class="com.cognizant.springlearn.Country">

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

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

### </bean>

### </list>

### </constructor-arg>

### </bean>

### </beans>

### ****Code: Country.java****

## package com.cognizant.springlearn;

## import org.slf4j.Logger;

## import org.slf4j.LoggerFactory;

## public class Country {

## private String code;

## private String name;

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

## public Country() {

## LOGGER.debug("Inside Country Constructor.");

## }

## public String getCode() {

## LOGGER.debug("Getting code: {}", code);

## return code; }

## public void setCode(String code) {

## LOGGER.debug("Setting code: {}", code);

## this.code = code; }

## public String getName() {

## LOGGER.debug("Getting name: {}", name);

## return name; }

## public void setName(String name) {

## LOGGER.debug("Setting name: {}", name);

## this.name = name;

## }

## @Override

## public String toString() {

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

## }

## }

## ****Hello World RESTful Web Service****

### Scenario:

Create a simple REST endpoint that returns Hello World!!.

### Steps:

1.Create HelloController with @GetMapping("/hello").

1. Return static message from sayHello() method.
2. Add logs for start and end of method.
3. Test using browser and Postman.

### SME Notes:

1.Show headers in DevTools > Network tab.

2. Inspect headers in Postman.

**Code: HelloController.java**

### package com.cognizant.springlearn.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")

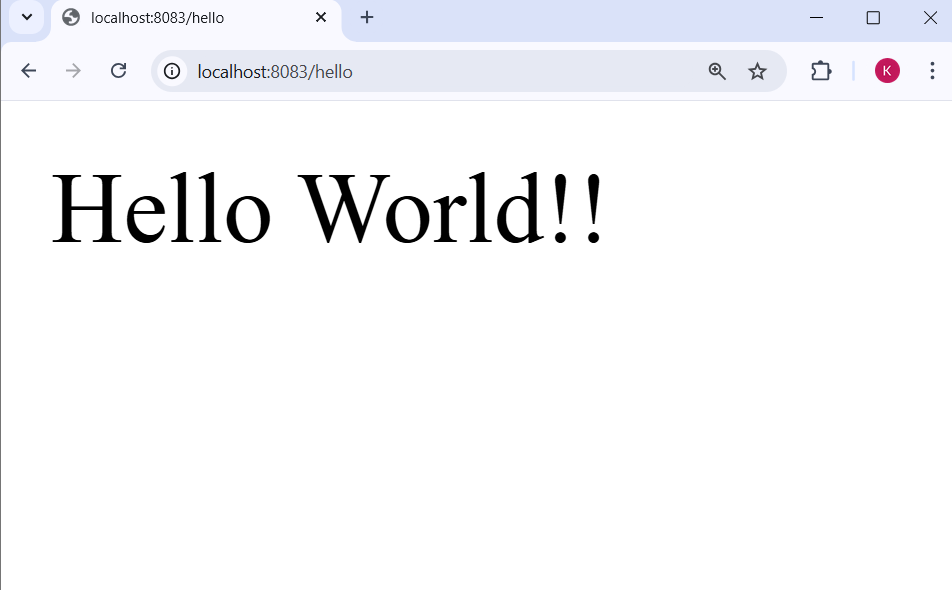
### public String sayHello() {

### LOGGER.info("Start sayHello()");

### String message = "Hello World!!";

### LOGGER.info("End sayHello()"); return message; } }

### Output:

**Request**: http://localhost:8083/hello  


## ****REST - Country Web Service****

### Scenario:

Expose India country details from Spring XML config via REST.

### Steps:

1.Create CountryController with @RequestMapping("/country").

2. Load country bean from XML.

3. Return as JSON.

**Code: CountryController.java**

### package com.cognizant.springlearn.controller;

### import com.cognizant.springlearn.Country;

### import com.cognizant.springlearn.exception.CountryNotFoundException;

### import com.cognizant.springlearn.service.CountryService;

### import org.slf4j.Logger; import org.slf4j.LoggerFactory;

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

### import org.springframework.context.ApplicationContext;

### import org.springframework.context.support.ClassPathXmlApplicationContext;

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

### @RestController

### public class CountryController {

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

### @Autowired

### private CountryService countryService;

### @RequestMapping("/country")

### public Country getCountryIndia() {

### LOGGER.debug("Start getCountryIndia()");

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

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

### LOGGER.debug("Returning country: {}", country); return country; }

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

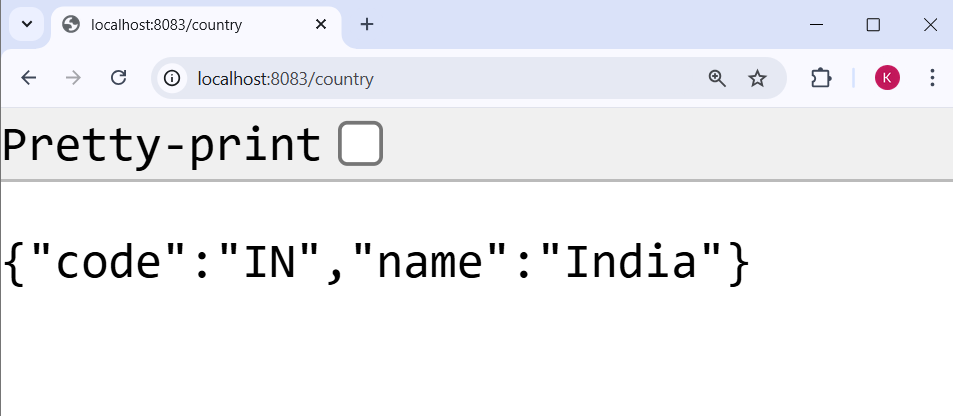
### public Country getCountry(@PathVariable String code) throws CountryNotFoundException {

### LOGGER.debug("Start getCountry() with code: {}", code);

### return countryService.getCountry(code); } }

### Output:

**Request**: http://localhost:8083/country  
**Response**:



**REST - Get Country by Code**

### Scenario: Return a country based on code (case-insensitive).

### Steps:

1. Use @GetMapping("/countries/{code}") in controller.
2. Load country list from XML.
3. Use lambda or loop to match code.
4. Return matched country or throw error.

### ****Code: CountryService.java****

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Country;

import com.cognizant.springlearn.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Service;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

@Service

public class CountryService {

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

public Country getCountry(String code) throws CountryNotFoundException {

LOGGER.debug("Start getCountry() with code: {}", code);

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

List<Country> countryList = context.getBean("countryList", java.util.ArrayList.class);

return countryList.stream()

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

.findFirst()

.orElseThrow(() -> {

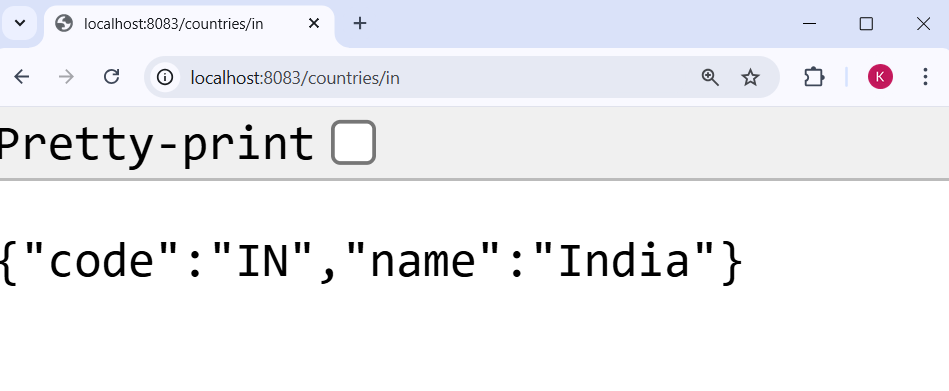
LOGGER.error("Country not found for code: {}", code);

return new CountryNotFoundException("Country not found with code: " + code);

}); } }

**Output:**

#### Request: <http://localhost:8083/countries/in>



## ****JWT Authentication Service****

### Scenario:

Authenticate user via basic auth and return JWT token.

### Steps:

1. Create AuthController in jwt-auth-service.
2. Decode basic auth credentials.
3. Validate user, generate JWT using JwtUtil.
4. Return JWT in JSON response.

**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>auth-jwt</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>auth-jwt</name>

    <description>JWT Auth Service</description>

    <parent>

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

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

        <version>3.1.0</version>

        <relativePath/>

    </parent>

    <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-security</artifactId>

        </dependency>

        <dependency>

            <groupId>io.jsonwebtoken</groupId>

            <artifactId>jjwt</artifactId>

            <version>0.9.1</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

### ****Code: AuthController.java****

package com.cognizant.jwt.controller;

import com.cognizant.jwt.service.JwtService;

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

import org.springframework.security.authentication.AuthenticationManager;

Import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthController {

    @Autowired

    private AuthenticationManager authenticationManager;

    @Autowired

    private JwtService jwtService;

    @PostMapping("/authenticate")

    public Map<String, String> generateToken(@RequestBody Map<String, String> request) {

        try {

            Authentication authenticate = authenticationManager.authenticate(

                    new UsernamePasswordAuthenticationToken(request.get("username"), request.get("password"))

            );

            String token = jwtService.generateToken(authenticate.getName());

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

        } catch (AuthenticationException e) {

            return Map.of("error", "Invalid credentials");

        }

    }

}

**HelloController.java**

package com.cognizant.jwt.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! This is a secured endpoint.";

    }

}

**SecurityConfig.java**

package com.cognizant.jwt.config;

import com.cognizant.jwt.filter.JwtAuthenticationFilter;

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

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.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

    @Autowired

    private JwtAuthenticationFilter jwtFilter;

    @Bean

    public UserDetailsService userDetailsService() {

        return new InMemoryUserDetailsManager(

                User.withUsername("admin").password("{noop}admin").roles("USER").build()

        ); }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

            .authorizeHttpRequests(auth -> auth

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

                .anyRequest().authenticated()

            )

            .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();

    }

    @Bean

    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

        return config.getAuthenticationManager();

    }

}

**WebConfig.java**

package com.cognizant.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig {

    @Bean

    public WebMvcConfigurer corsConfigurer() {

        return new WebMvcConfigurer() {

            @Override

            public void addCorsMappings(CorsRegistry registry) {

                registry.addMapping("/\*\*")

                        .allowedOrigins("http://127.0.0.1:5500")

                        .allowedMethods("\*")

                        .allowedHeaders("\*");

            }

        };

} }

**JwtAuthenticationFilter.java**

package com.cognizant.jwt.filter;

import com.cognizant.jwt.service.JwtService;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

import java.util.Collections;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired

    private JwtService jwtService;

    @Override

    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

            throws ServletException, IOException {

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

        String username = null;

        String jwt = null;

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

            jwt = authHeader.substring(7);

            username = jwtService.validateToken(jwt);

        }

        if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

            UsernamePasswordAuthenticationToken authToken =

                    new UsernamePasswordAuthenticationToken(username, null, Collections.emptyList());

            authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

            SecurityContextHolder.getContext().setAuthentication(authToken);

        }

        filterChain.doFilter(request, response);

    }

}

package com.cognizant.jwt.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class JwtService {

    private static final String SECRET\_KEY = "mysecretkey123";

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

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

                .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

                .compact();

    }

    public String validateToken(String token) {

        return Jwts.parser()

                .setSigningKey(SECRET\_KEY)

                .parseClaimsJws(token)

                .getBody()

                .getSubject();

    }

}

**AuthApplication.java**

package com.cognizant.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AuthApplication {

    public static void main(String[] args) {

        SpringApplication.run(AuthApplication.class, args);

    }

}

**Output:**

