

Create authentication service that returns JWT

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.

Request

```
curl -s -u user:pwd http://localhost:8090/authenticate
```

Response

```
{"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOiE1NzAzODA2NzR9.t3LRv1CV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}
```

This can be incorporated as three major steps:

- Create authentication controller and configure it in SecurityConfig
- Read Authorization header and decode the username and password
- Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

Code:

SecurityConfig.java

```
package com.cognizant.spring_learn.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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().disable()
            .authorizeHttpRequests()
                .requestMatchers("/authenticate").permitAll()
                .anyRequest().authenticated()
            .and()
            .httpBasic();

        return http.build();
    }
}
```

AuthenticationController.java

```
package com.cognizant.spring_learn.controller;

import com.cognizant.spring_learn.util.JwtUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Base64;

import jakarta.servlet.http.HttpServletRequest;

@RestController
public class AuthenticationController {

    @Autowired
    private JwtUtil jwtUtil;

    @GetMapping("/authenticate")
    public TokenResponse authenticate(HttpServletRequest request) {
        String authHeader = request.getHeader("Authorization");
        if (authHeader == null || !authHeader.startsWith("Basic ")) {
            throw new RuntimeException("Missing or invalid Authorization header");
        }

        // Decode base64
        String base64Credentials = authHeader.substring("Basic ".length());
        byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);
        String credentials = new String(credDecoded);
        String[] values = credentials.split(":", 2);

        String username = values[0];
        String password = values[1];

        // Basic check: (replace with DB check or UserDetailsService)
        if (username.equals("user") && password.equals("pwd")) {
            String token = jwtUtil.generateToken(username);
            return new TokenResponse(token);
        } else {
            throw new RuntimeException("Invalid credentials");
        }
    }

    static class TokenResponse {
        private String token;

        public TokenResponse(String token) {
            this.token = token;
        }

        public String getToken() {
            return token;
        }

        public void setToken(String token) {
            this.token = token;
        }
    }
}
```

JwtUtil.java

```
package com.cognizant.spring_learn.util;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class JwtUtil {
    private final String secret = "my_secret_key"; // Use strong secret in prod
    private final long expirationTime = 1000 * 60 * 60; // 1 hour

    public String generateToken(String username) {
        return Jwts.builder()
            .setSubject(username)
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() +
expirationTime))
            .signWith(SignatureAlgorithm.HS256, secret)
            .compact();
    }
}
```

Output:

```

  ____  __  __
 / ___/  / /  /
/ /   /  / /  /
/ /___/  / /  /
\___/___/___/___/

:: Spring Boot ::      (v3.5.3)

2025-07-11T16:01:11.061+05:30 INFO 14236 --- [spring-learn] [ restartedMain] c.c.spring_learn.SpringLearnApplication : Starting SpringLearnApplication using Java 21.0.2 with PID 14236 (C:\User
2025-07-11T16:01:11.064+05:30 INFO 14236 --- [spring-learn] [ restartedMain] c.c.spring_learn.SpringLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-11T16:01:11.104+05:30 INFO 14236 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' t
2025-07-11T16:01:11.105+05:30 INFO 14236 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.we
2025-07-11T16:01:12.055+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8082 (http)
2025-07-11T16:01:12.073+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-11T16:01:12.073+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-11T16:01:12.099+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-11T16:01:12.100+05:30 INFO 14236 --- [spring-learn] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 994 ms
2025-07-11T16:01:12.221+05:30 WARN 14236 --- [spring-learn] [ restartedMain] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: a9abc75b-b26b-4e1f-9498-ac352860025f

This generated password is for development use only. Your security configuration must be updated before running your application in production.

2025-07-11T16:01:12.227+05:30 INFO 14236 --- [spring-learn] [ restartedMain] r$initializeUserDetailsServiceConfigurer : Global AuthenticationManager configured with UserDetailsService bean with
2025-07-11T16:01:12.527+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2025-07-11T16:01:12.568+05:30 INFO 14236 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8082 (http) with context path '/'
2025-07-11T16:01:12.577+05:30 INFO 14236 --- [spring-learn] [ restartedMain] c.c.spring_learn.SpringLearnApplication : Started SpringLearnApplication in 1.933 seconds (process running for 2.32
2025-07-11T16:01:12.706+05:30 INFO 14236 --- [spring-learn] [ restartedMain] c.c.spring_learn.SpringLearnApplication : Country: com.cognizant.spring_learn.Country@465bb390
Country: com.cognizant.spring_learn.Country@465bb390
```

