LANGKAH-LANGKAH MEMBUAT SPRING BOOT + OTENTIKASI JASON WEB TOKEN (JWT) MENGGUNAKAN VS CODE

1. Install setiap tools yang dibutuhkan:

Pastikan tools di bawah ini sudah terinstall:

a. Java Development Kit (JDK)

- Install **JDK 17 or 21**
- Untuk verifikasi bisa melakukan perintah ini dalam cmd/terminal: java -version

b. Visual Studio Code

- Install ekstensi VS Code:
  - **Extension Pack for Java**
  - **Spring Boot Dashboard**
  - **Spring Initializr**
  - **Lombok Annotations Support** (opsional tapi direkomendasikan)

c. MySQL Server + MySQL Workbench

- Buat database baru: CREATE DATABASE dbmarket;

2. Membuat Spring Boot Project (Menggunakan Spring Initializr in VS Code)

1. Buka VS Code
2. Tekan **Ctrl + Shift + P**
3. Pilih **"Spring Initializr: Create a Maven Project"**
4. Pilih **Spring Boot version** (3.x recommended)
5. Pilih **Project language :** Java
6. Isi:
   - Group: `com.rmn`
   - Artifact: `latihan_jwt`
7. Pilih **Packaging Type:** JAR
8. Pilih versi **Java 17** atau **21**
9. Tambahkan dependencies:
   - **Spring Web**
   - **Spring Data JPA**
   - **Spring Boot DevTools**
   - **Spring Security**
   - **MySQL Driver**
   - **Lombok**
10. Generate the project → choose a folder → Finish.

3. Konfigurasi MySQL Connection dalam application.properties

Dalam src/main/resources/application.properties:

```
spring.application.name=latihan_jwt
# --- Konfigurasi Server ---
server.port=8080

# --- Konfigurasi MySQL ---
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/dbmarket?useSSL=false&ser
verTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# --- Konfigurasi JPA/Hibernate ---
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

4. Menambahkan library dalam proyek (pom.xml)

pom.xml digunakan sebagai file konfigurasi utama Maven pada proyek Spring Boot / Java. File ini menentukan bagaimana proyek dibangun, dependensi apa yang dipakai, dan versi library yang digunakan. Karena latihan kali ini menggunakan JWT (Json Web Token) maka tambahkan dependency berikut ke dalam pom.xml:

```
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-api</artifactId>
            <version>0.12.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-impl</artifactId>
            <version>0.12.5</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-jackson</artifactId>
            <version>0.12.5</version>
            <scope>runtime</scope>
        </dependency>
```

Dependency Maven tersebut digunakan untuk mengaktifkan fitur JWT (JSON Web Token) di aplikasi Spring Boot, menggunakan library JJWT (Java JWT). Library ini memungkinkan aplikasi untuk membuat, menandatangani, membaca, dan memvalidasi JWT.

5. Membuat Model (Entity)

Model (Entity): Digunakan untuk merepresentasikan data. Buat folder model dalam package com.rmn dan masukkan kelas baru.

Buatlah kelas baru bernama AuthRequest.java. Kelas ini digunakan untuk mendefinisikan model/DTO (Data Transfer Object) bernama AuthRequest yang berfungsi menampung data autentikasi, yaitu username dan password.

```
package com.rmn.model;

public class AuthRequest {
    private String username;
    private String password;
```

```
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}
```

Buatlah kelas baru bernama AuthResponse.java. Kelas ini digunakan untuk mengirimkan hasil autentikasi (response login) ke client berupa token (JWT – JSON Web Token).

```
package com.rmn.model;

public class AuthResponse {
    private String token;

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

    public String getToken() {
        return token;
    }
}
```

6. Membuat Service

Jika Repository digunakan untuk mengakses data, maka Service (atau *Service Layer*) adalah komponen inti di Spring yang digunakan untuk tujuan utama yaitu mengimplementasikan logika bisnis (Business Logic) dan mengkoordinasikan alur kerja aplikasi. Sekarang buatlah folder baru bernama service dalam package com.rmn dan kelas MyUserDetailsService.java lalu masukkan kode berikut:

```
package com.rmn.service;

import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class MyUserDetailsService implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String username)
            throws UsernameNotFoundException {

        if (!username.equals("admin")) {
            throw new UsernameNotFoundException("User not found");
        }

        return User.builder()
                .username("admin")
.password("$2y$10$AThSYf6otFBLfojORa/dHuscAfuO8QPxcn7E07YW.LepyE2Gm1k4y")
```

```
                .roles("ADMIN")
                .build();
    }
}
```

7. Membuat Konfigurasi

Config (configuration) digunakan untuk mengatur cara aplikasi berjalan, menghubungkan komponen, dan menentukan perilaku sistem tanpa mengubah logic utama. Pertama kita buat terlebih dahulu package (folder) baru bernama config.

Buatlah kelas bernama JwtAuthFilter.java dalam package tersebut. Kelas ini digunakan sebagai JWT Authentication Filter di Spring Boot + Spring Security. Fungsinya adalah memeriksa dan memvalidasi JWT token pada setiap request, lalu mengautentikasi user secara otomatis jika token valid. Peran utama JwtAuthFilter menjaga endpoint yang dilindungi agar hanya bisa diakses user yang sudah login (memiliki JWT valid).

```
package com.rmn.config;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationT
oken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
public class JwtAuthFilter extends OncePerRequestFilter {

    private final JwtUtil jwtUtil;
    private final UserDetailsService userDetailsService;

    private static final String[] WHITELIST = {
        "/api/auth/login",
        "/api/auth/register",
        "/error"
    };

          public    JwtAuthFilter(JwtUtil    jwtUtil,    UserDetailsService
userDetailsService) {
        this.jwtUtil = jwtUtil;
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected boolean shouldNotFilter(HttpServletRequest request) {
```

```
         String path = request.getServletPath();
        for (String w : WHITELIST) {
            if (path.startsWith(w)) return true;
        }
        return false;
    }

    @Override
    protected void doFilterInternal(
            HttpServletRequest request,
            HttpServletResponse response,
            FilterChain filterChain)
            throws ServletException, IOException {

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

        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            String token = authHeader.substring(7);

            try {
                String username = jwtUtil.validateToken(token);

                                                UserDetails    userDetails    =
userDetailsService.loadUserByUsername(username);

                UsernamePasswordAuthenticationToken auth =
                        new UsernamePasswordAuthenticationToken(
                                                            userDetails,   null,
userDetails.getAuthorities());

                SecurityContextHolder.getContext().setAuthentication(auth);

            } catch (Exception e) {
                response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                return;
            }
        }

        filterChain.doFilter(request, response);
    }
}
```

Buatlah kelas baru lagi Bernama JwtUtil.java. Kelas ini digunakan sebagai utility (helper) untuk JWT (JSON Web Token) di Spring Boot, khususnya untuk membuat (generate) dan memvalidasi token JWT yang dipakai pada proses autentikasi. Kelas ini adalah inti dari mekanisme login berbasis JWT. Fungsi utama JwtUtil adalah membuat token saat login berhasil dan memvalidasi token pada setiap request selanjutnya.

```
package com.rmn.config;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.io.Decoders;
```

```
import io.jsonwebtoken.security.Keys;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.Date;

@Component
public class JwtUtil {

    // Base64 secret minimal 256-bit
    private static final String SECRET =
            "uA6ZlDzFf6FvO6YvWZ9yWv7mCS3uT4pQYJx2e7v8xWk=";

    private final SecretKey key = Keys.hmacShaKeyFor(
            Decoders.BASE64.decode(SECRET)
    );

    public String generateToken(String username) {
        long now = System.currentTimeMillis();

        return Jwts.builder()
                .subject(username)
                .issuedAt(new Date(now))
                .expiration(new Date(now + 1000 * 60 * 60)) // 1 jam
                .signWith(key)
                .compact();
    }

    public String validateToken(String token) {
        return Jwts.parser()
                .verifyWith(key)
                .build()
                .parseSignedClaims(token)
                .getPayload()
                .getSubject();
    }
}
```

Buatlah kelas baru lagi Bernama SecurityConfig.java. Kelas ini digunakan untuk mengatur seluruh aturan keamanan (security) aplikasi Spring Boot, khususnya JWT-based authentication dengan Spring Security.

```
package com.rmn.config;

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.EnableWebS
ecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticat
ionFilter;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    private final JwtAuthFilter jwtFilter;
    private final UserDetailsService userDetailsService;

     public SecurityConfig(
            JwtAuthFilter jwtFilter,
            UserDetailsService userDetailsService
    ) {
        this.jwtFilter = jwtFilter;
        this.userDetailsService = userDetailsService;
    }

    @Bean
      public  SecurityFilterChain  filterChain(HttpSecurity  http)  throws
Exception {
        System.out.println(">>> SECURITY CONFIG LOADED <<<");
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                                        .requestMatchers("/api/auth/login",
"/api/auth/register").permitAll()
                .anyRequest().authenticated()
            )
            .sessionManagement(session -> session
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            )
            .userDetailsService(userDetailsService);

                                            http.addFilterBefore(jwtFilter,
UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    public AuthenticationManager authenticationManager(
            AuthenticationConfiguration config) throws Exception {
        return config.getAuthenticationManager();
```

```
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

8. Membuat Controller REST API

Ini adalah komponen yang bertugas menerima permintaan HTTP dari klien dan mengirimkan respons kembali. *Controller* digunakan untuk mengekspos *endpoint* REST.

Buat folder baru (misalnya com.rmn.controller).

Buat kelas bernama AuthController.java dalam package controller. Kelas ini digunakan sebagai controller REST untuk proses login (autentikasi) pada aplikasi Spring Boot dengan JWT. Kelas menerima request login, memverifikasi username & password, lalu mengembalikan JWT token.

```
package com.rmn.controller;

import com.rmn.config.JwtUtil;
import com.rmn.model.AuthRequest;
import com.rmn.model.AuthResponse;

import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    private final JwtUtil jwtUtil;
    private final UserDetailsService userDetailsService;
    private final PasswordEncoder passwordEncoder;


    public AuthController(
        JwtUtil jwtUtil,
        UserDetailsService userDetailsService,
        PasswordEncoder passwordEncoder
    ) {
        this.jwtUtil = jwtUtil;
        this.userDetailsService = userDetailsService;
        this.passwordEncoder = passwordEncoder;
    }

    @PostMapping("/login")
    public AuthResponse login(@RequestBody AuthRequest req) {

        UserDetails user =
```

```
                userDetailsService.loadUserByUsername(req.getUsername());

        if                      (!passwordEncoder.matches(req.getPassword(),
user.getPassword())) {
            throw new RuntimeException("Invalid credentials");
        }

        String token = jwtUtil.generateToken(user.getUsername());
        return new AuthResponse(token);
    }
}
```

Buat kelas bernama HelloController.java dalam package controller. Kode ini digunakan sebagai REST Controller sederhana untuk menguji endpoint API, khususnya endpoint yang dilindungi (protected) oleh Spring Security + JWT.

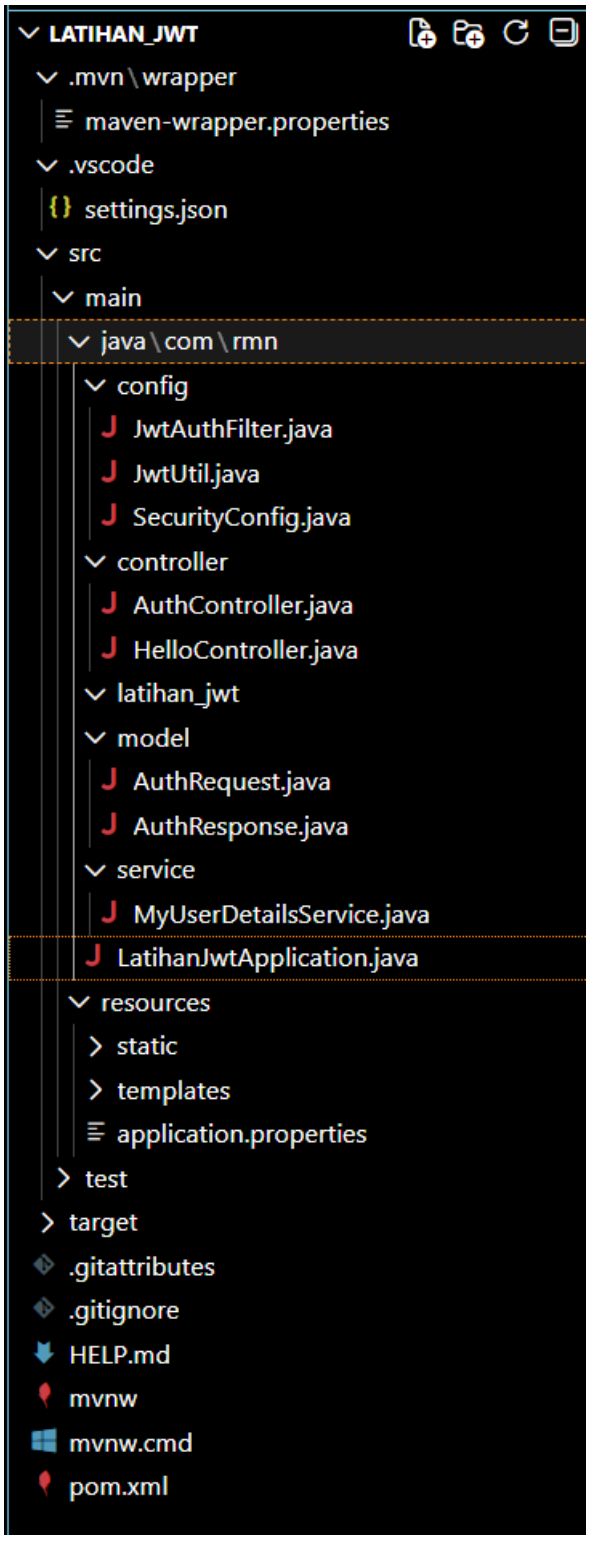
```
package com.rmn.controller;

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

@RestController
public class HelloController {

    @GetMapping("/api/hello")
    public String hello() {
        return "Hello, this is a protected endpoint!";
    }
}
```

9. Memindahkan Main Class

Pindahkan main class (Misal: LatihanJwtApplication) ke dalam package com.rmn sehingga kodenya seperti berikut:

```
package com.rmn;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LatihanJwtApplication {

      public static void main(String[] args) {
            SpringApplication.run(LatihanJwtApplication.class, args);
      }

}
```

Struktur Akhir Projek



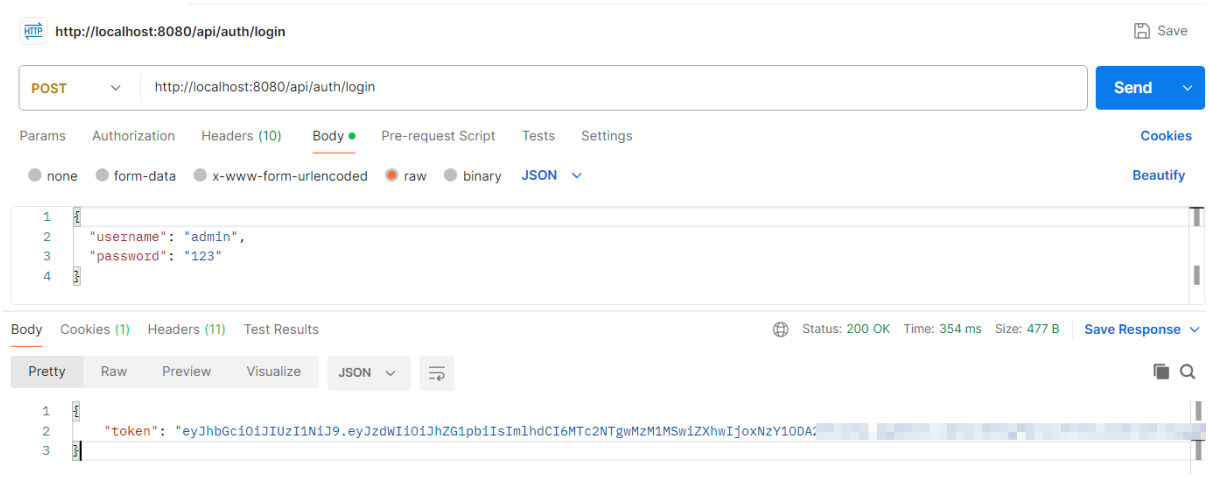
10. Menjalankan Aplikasi

Pada VS Code:

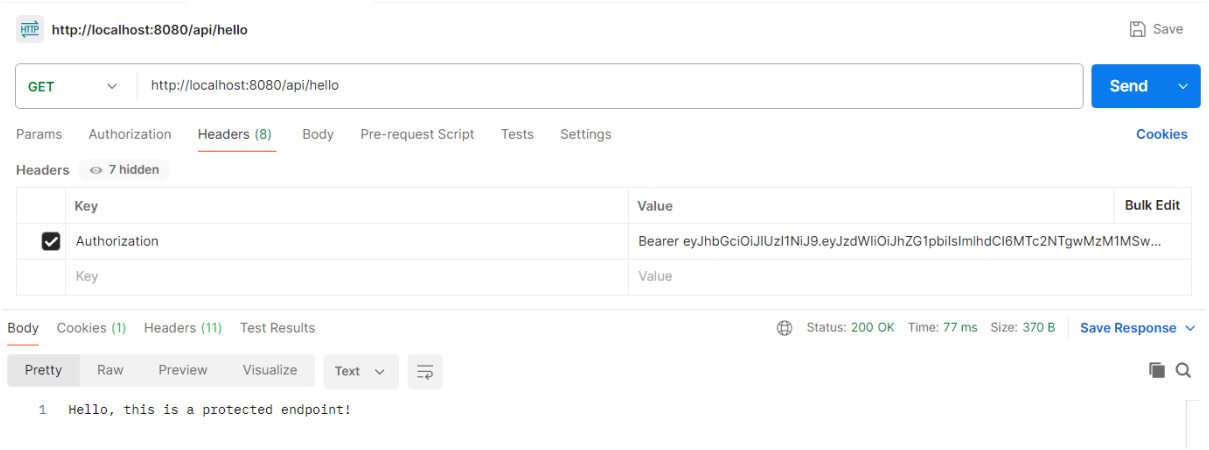
- Gunakan terminal:

  ```
  ./mvnw spring-boot:run
  ```

Check API menggunakan Postman:



Otentikasi API



Menampilkan hello