Backend การตรวจสอบสิทธิ์

ใช้การตรวจสอบสิทธิ์การเข้าใช้งานด้วย JSON Web Token (JWT) โดย Session Creation Policy จะเป็นแบบ stateless

ไลบรารี่ที่ใช้งาน คือ maven: com.auth0 / java-jwt / 3.18.1

[auth0/java-jwt: Java implementation of JSON Web Token (JWT) (github.com)]

```
<dependency>
    <groupId>com.auth0</groupId>
    <artifactId>java-jwt</artifactId>
    <version>3.18.1</version>
</dependency>
```

ซึ่ง JWT ที่สร้างจะเก็บ

```
{
  "principal": 1,
  "role": "ADMIN",
  "iss": "cfanBackend",
  "exp": 1638663881
}
```

1. principal คือสิ่งไว้ระบุตัวตนผู้ใช้ โดยเก็บเป็น User Id ที่ไม่สามารถเปลี่ยนแปลงได้
2. role คือสิทธิ์ว่าผู้ใช้สามารถเข้าถึง Endpoint ได้ในระดับไหน
3. iss คือการบอกว่า JWT นี้ถูกสร้างมาจาก cfanBackend
4. exp คือวันหมดอายุของ JWT ซึ่งได้กำหนดไว้เป็น 24 ชม.

ขั้นตอนการสร้าง JWT

1. ตรวจสอบสิทธิ์การใช้งานของผู้ใช้ที่เข้าสู่ระบบ

```
public LoginResponseModel login(LoginModel loginModel) {
    User user = userRepository.findByEmail(loginModel.getEmail());
    if (user == null) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_EMAIL_DOES_NOT_EXIST, "User : Email {" + loginModel.getEmail() + "} does not exist !!");
    }
    if (!passwordEncoder.matches(loginModel.getPassword(), user.getPassword())) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_PASSWORD_INCORRECT, "User : password incorrect !!");
    }
    if (user.getStatus().equals(User.Status.TBC)) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_ACCOUNT_NOT_VERIFIED, "User : account not verified !!");
    }
    String token = tokenService.tokenize(user);
    return new LoginResponseModel(user, success: true, token);
}
```

2. สร้าง JWT จากข้อมูลของ User ที่ผ่านการยืนยันแล้ว

```
@Value("${cfan.token.secret}")
private String secret;

@Value("${cfan.token.issuer}")
private String issuer;

private Algorithm algorithm() { return Algorithm.HMAC256(secret); }

public String tokenize(User user) {
    Calendar calendar = Calendar.getInstance(TimeZone.getTimeZone("Asia/Bangkok"));
    calendar.add(Calendar.MINUTE, amount: 60 * 24);
    Date expiresAt = calendar.getTime();

    return JWT.create()
            .withIssuer(issuer)
            .withClaim( name: "principal", user.getUserid())
            .withClaim( name: "role", user.getStatus().toString())
            .withExpiresAt(expiresAt)
            .sign(algorithm());
}
```

3. ส่งคืนข้อมูล JWT ที่สร้าง

```
"success": true,
"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJwcmluY2lwYWwiOjEsInJvbGUiOiJBRE1JTiIsImlzcyI6ImNmYW5CYWNrZW5kIiwiZXhwIjoxNjM4NjYzODgxfQ.
    3rb2yJcrfJrg-nfcFzusDJ-bhnslsZLuO7IaeiTaznI"
```

ขั้นตอนการตรวจสอบสิทธิ์การเข้าถึง

1. ตรวจสอบว่า request ที่เข้ามามี Header “Authorization” และเริ่มต้นด้วย “Bearer ” หรือไม่

```
HttpServletRequest request = (HttpServletRequest) servletRequest;
String authorization = request.getHeader( name: "Authorization");
if (ObjectUtils.isEmpty(authorization)) {
    filterChain.doFilter(servletRequest, servletResponse);
    return;
}

if (!authorization.startsWith("Bearer ")) {
    filterChain.doFilter(servletRequest, servletResponse);
    return;
}
```

2. ตรวจสอบว่า JWT ที่เข้ามาไม่ได้อยู่ใน jwtblacklist ซึ่งเก็บข้อมูลของ JWT ที่ได้ logout ไปแล้ว

```
String token = authorization.substring(7);
if (jwtblacklistRepository.existsByToken(token)) {
    filterChain.doFilter(servletRequest, servletResponse);
    return;
}
```

3. Verify JWT โดยจะตรวจสอบว่า JWT ถูกต้อง ไม่ได้ถูกแก้ไขและยังไม่หมดอายุ

```
DecodedJWT decoded = tokenService.verify(token);

if (decoded == null) {
    filterChain.doFilter(servletRequest, servletResponse);
    return;
}
```

```
public DecodedJWT verify(String token) {
    try {
        JWTVerifier verifier = JWT.require(algorithm())
                .withIssuer(issuer)
                .build();

        return verifier.verify(token);

    } catch (Exception e) {
        return null;
    }
}
```

4. นำข้อมูลที่ได้จาก JWT มาเก็บไว้ใน Context โดยเก็บ role เป็น authorities

```
Long principal = decoded.getClaim( name: "principal").asLong();
String role = decoded.getClaim( name: "role").asString();

List<GrantedAuthority> authorities = new ArrayList<>();
authorities.add(new SimpleGrantedAuthority(role));

UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(principal, token, authorities);

SecurityContext context = SecurityContextHolder.getContext();
context.setAuthentication(authentication);

filterChain.doFilter(servletRequest, servletResponse);
```

5. User ที่มี authorities ถูกต้องจะสามารถเข้า Endpoint ได้ตามที่กำหนดไว้

```
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
.and().authorizeRequests().antMatchers( ...antPatterns: "/actuator/**", "/api/general/**").permitAll()
.antMatchers( ...antPatterns: "/api/user/**").hasAnyAuthority( ...authorities: "NORMAL", "ADMIN")
.antMatchers( ...antPatterns: "/api/admin/**").hasAuthority("ADMIN")
.anyRequest().authenticated()
.and().apply(new TokenFilterConfiguerer(tokenService, jwtblacklistRepository));
```

6. ในทุก ๆ Endpoint ของ /user จะมีการดึงข้อมูล User จาก Id ใน Context มาตรวจสอบอีกครั้ง

```
public User getUserById(Long userid) {
    if (userid == null) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_INCORRECT_ID, "User : id null !!");
    }
    User user = userRepository.findById(userid).orElse( other: null);
    if (user == null) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_DOES_NOT_EXIST, "User : id {" + userid + "} does not exist !!");
    }
    return user;
}

public User getUser() {
    Long userid = SecurityUtil.getCurrentUserId();
    if (userid == null) {
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_UNAUTHORIZED, "User : unauthorized !!");
    }
    return getUserById(userid);
}
```

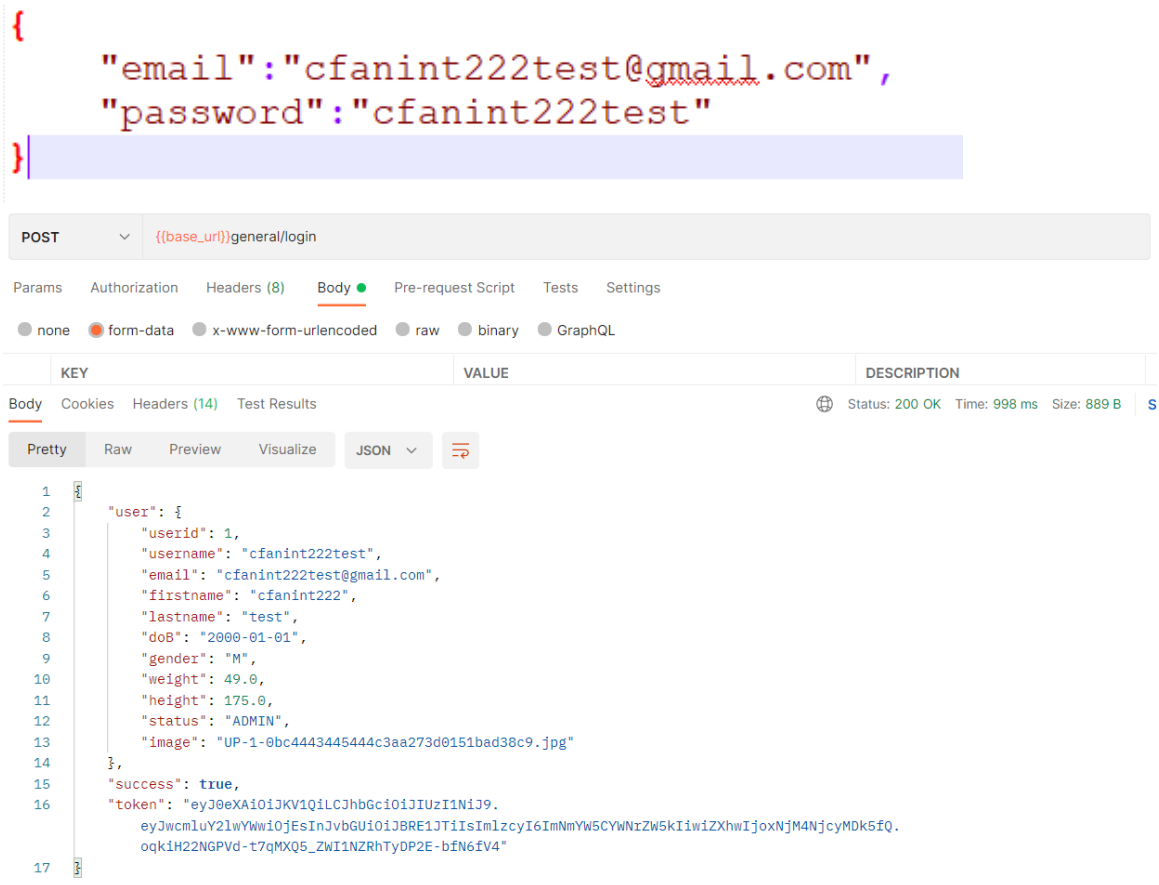
```
public static Long getCurrentUserId() {
    SecurityContext context = SecurityContextHolder.getContext();
    if (context == null) {
        return null;
    }

    Authentication authentication = context.getAuthentication();
    if (authentication == null) {
        return null;
    }

    Object principal = authentication.getPrincipal();
    if (principal == null) {
        return null;
    }

    Long userId = (Long) principal;
    return userId;
}
```

7. หากเป็น /Admin ก็จะมีการตรวจสอบว่า User มี Status Admin หรือไม่อีกครั้ง

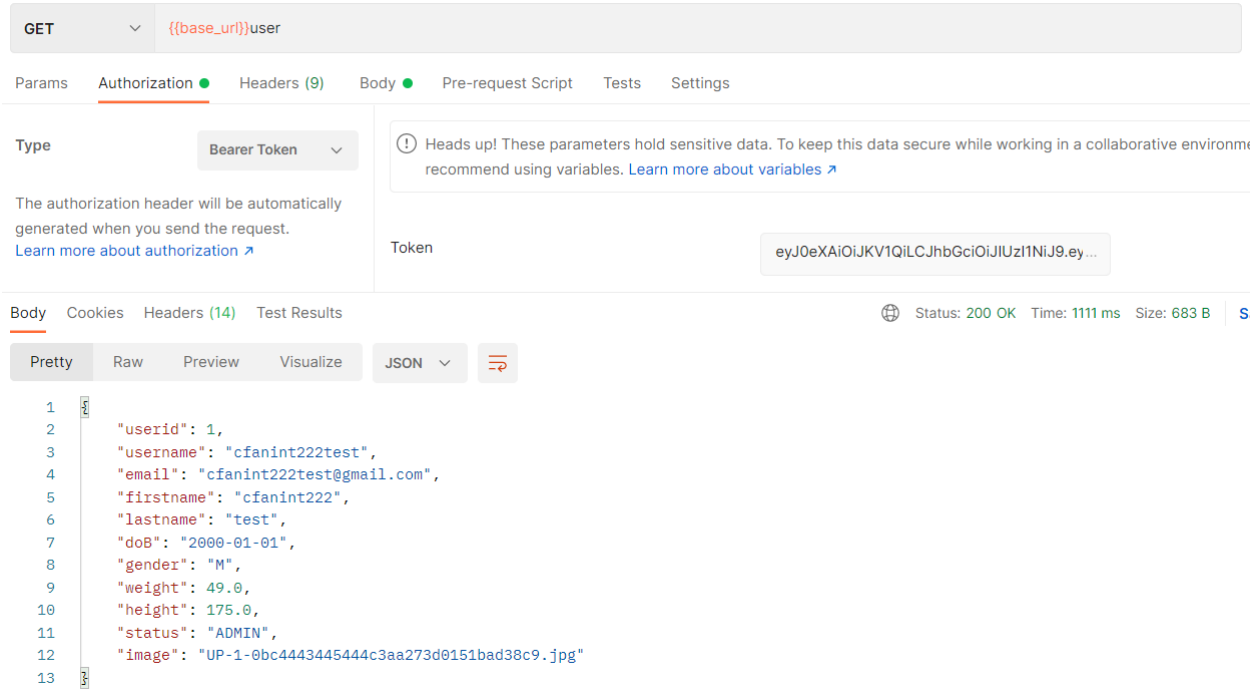
```
public void isADMIN() {
    User user = getUser();
    if (user.getStatus() != User.Status.ADMIN){
        logout();
        throw new BaseException(ExceptionResponse.ERROR_CODE.USER_UNAUTHORIZED, "User : unauthorized !!");
    }
}
```

กรณีตัวอย่างการใช้งาน

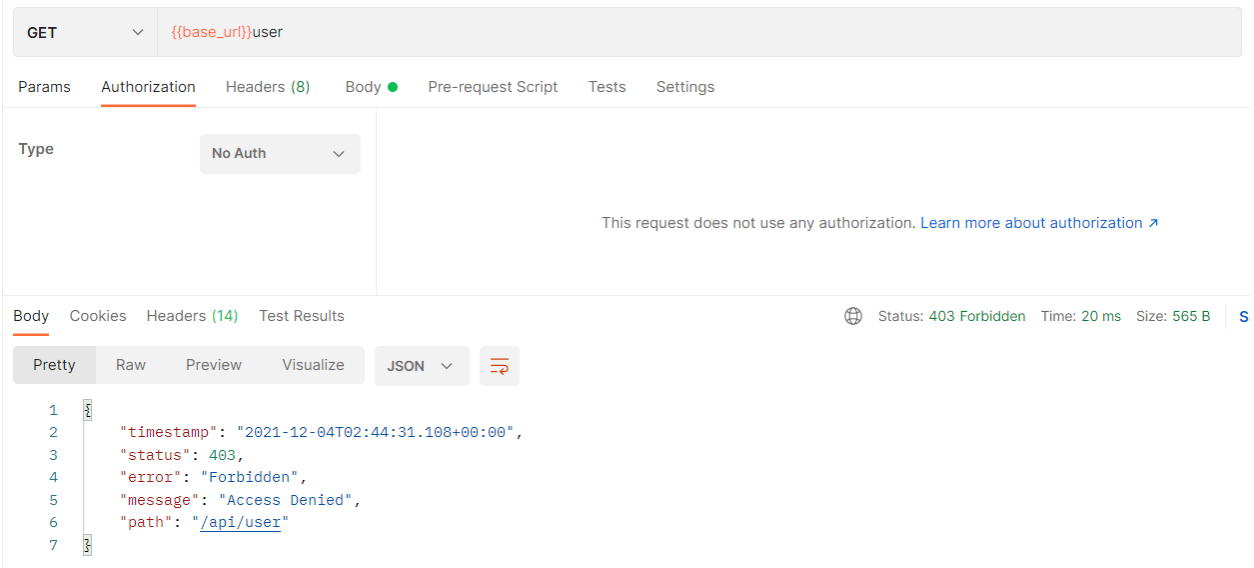
1. การเข้าสู่ระบบ



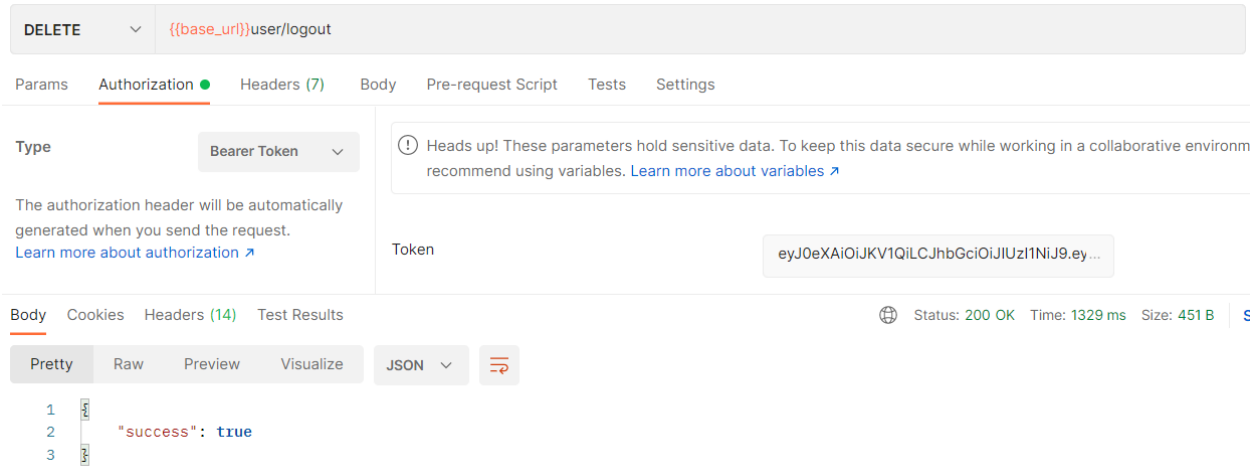
2. การเรียกข้อมูล User คนนั้น ๆ



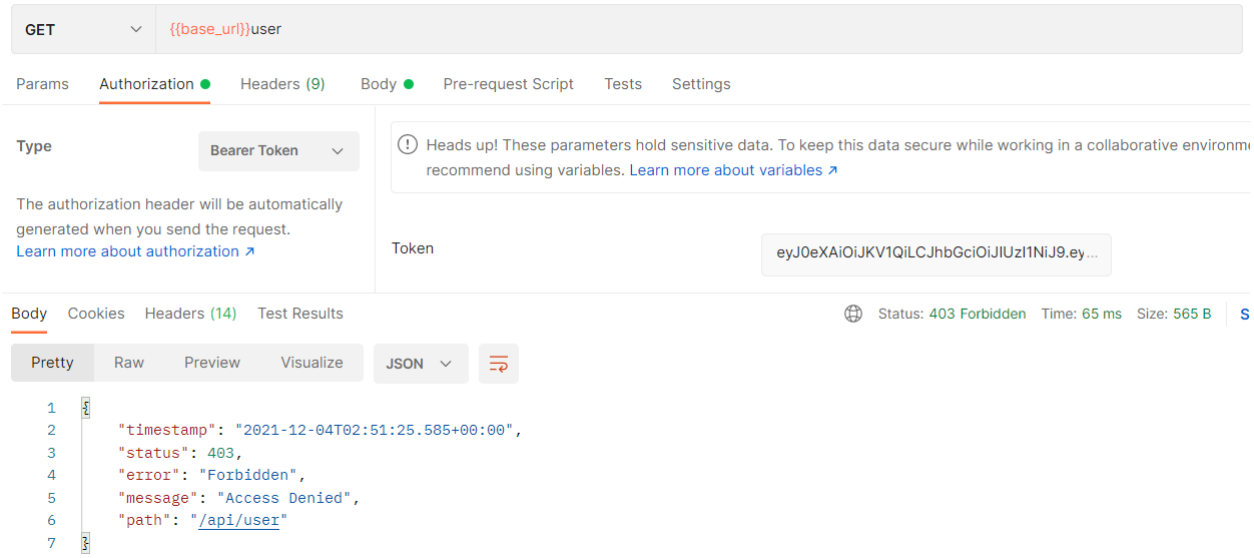
กรณีไม่ได้ส่ง Header



3. การออกจากระบบ



กรณี JWT ที่ส่งมาถูกใช้ออกจากระบบไปแล้ว



4. เมื่อ User จะเข้า Endpoint ที่ไม่มีสิทธิ์