

Backend การตรวจสอบสิทธิ์

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

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

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

```
<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.eyJwcm90b3RpdmlkeSI6ImN5Y21wYXN0IiwiaWF0IjE1NjYzODgxfQ.3rb2yJcrfJrg-nfcFzusDJ-bhnsIsZLu07IaeiTaznI"
```

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("principal").asLong();
String role = decoded.getClaim("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 TokenFilterConfigurer(tokenService, jwtblacklistRepository));
```

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

อีกครั้ง

```
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

กรณีตัวอย่างการใช้งาน

1. การเข้าสู่ระบบ

```
{
  "email": "cfanint222test@gmail.com",
  "password": "cfanint222test"
}
```

POST `{{(base_url)}}general/login`

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

| KEY | VALUE | DESCRIPTION |
|--|-------|------------------------|
| Body Cookies Headers (14) Test Results Status: 200 OK Time: 998 ms Size: 889 B | | |
| Pretty | Raw | Preview Visualize JSON |

```

1 {
2   "user": {
3     "userId": 1,
4     "username": "cfanint222test",
5     "email": "cfanint222test@gmail.com",
6     "firstname": "cfanint222",
7     "lastname": "test",
8     "dob": "2000-01-01",
9     "gender": "M",
10    "weight": 49.0,
11    "height": 175.0,
12    "status": "ADMIN",
13    "image": "UP-1-0bc4443445444c3aa273d0151bad38c9.jpg"
14  },
15  "success": true,
16  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcmVuY2lwYWwiojEsInZjbGUoIjBREiJTIiIsImZybyI6ImNmYW5CYWNRZW5kIiwiaXNjaXNjcycMDK5fQ.oqkiH22NGPvd-t7qMXQS_ZWI1NZRhtYP2E-bfN6fv4"

```

2. การเรียกข้อมูล User คนนั้น ๆ

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{base_url}}user
- Authorization:** Bearer Token
- Token:** eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...
- Status:** 200 OK
- Time:** 1111 ms
- Size:** 683 B
- Body (JSON):**

```
1 {
2   "userid": 1,
3   "username": "cfaint222test",
4   "email": "cfaint222test@gmail.com",
5   "firstname": "cfaint222",
6   "lastname": "test",
7   "dob": "2000-01-01",
8   "gender": "M",
9   "weight": 49.0,
10  "height": 175.0,
11  "status": "ADMIN",
12  "image": "UP-1-0bc4443445444c3aa273d0151bad38c9.jpg"
13 }
```

กรณีไม่ได้ส่ง Header

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{base_url}}user
- Authorization:** No Auth
- Status:** 403 Forbidden
- Time:** 20 ms
- Size:** 565 B
- Body (JSON):**

```
1 {
2   "timestamp": "2021-12-04T02:44:31.108+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Access Denied",
6   "path": "/api/user"
7 }
```

3. การออกจากระบบ

DELETE

▼

{{base_url}}user/logout

Params

Authorization ●

Headers (7)

Body

Pre-request Script

Tests

Settings

Type

Bearer Token ▼

ⓘ

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment we recommend using variables. [Learn more about variables](#) ➤

The authorization header will be automatically generated when you send the request.
[Learn more about authorization](#) ➤

Token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...

Body

Cookies

Headers (14)

Test Results

🌐

Status: 200 OK

Time: 1329 ms

Size: 451 B

S

Pretty

Raw

Preview

Visualize

JSON ▼

≡

1

2

3

```
{
  "success": true
}
```

กรณี JWT ที่ส่งมาถูกใช้ออกจากระบบไปแล้ว

GET

▼

{{base_url}}user

Params

Authorization ●

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Type

Bearer Token ▼

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#) ➔

Token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...

The authorization header will be automatically generated when you send the request.
[Learn more about authorization](#) ➔

Body

Cookies

Headers (14)

Test Results

⚙ Status: 403 Forbidden Time: 65 ms Size: 565 B S

Pretty

Raw

Preview

Visualize

JSON ▼

≡

1

2

3

4

5

6

7

⌕

```
"timestamp": "2021-12-04T02:51:25.585+00:00",
"status": 403,
"error": "Forbidden",
"message": "Access Denied",
"path": "/api/user"
```

⌕

4. เมื่อ User จะเข้า Endpoint ที่ไม่มีสิทธิ์

POST

{{base_url}}general/login

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE | DESCRIPTION |
|---|---------|--------------|--------------|
| Body | Cookies | Headers (14) | Test Results |
| Status: 200 OK Time: 116 ms Size: 891 B | | | |
| S | | | |
| PrettyRawPreviewVisualizeJSON | | | |
| <pre>1 { 2 "user": { 3 "userid": 2, 4 "username": "int222testuser", 5 "email": "int222testuser@gmail.com", 6 "firstname": "int222", 7 "lastname": "testuser", 8 "dob": "2000-01-01", 9 "gender": "M", 10 "weight": 50.0, 11 "height": 180.0, 12 "status": "NORMAL", 13 "image": "UP-2-9f2b3fdc450d40829162484be5b6a586.png" 14 }, 15 "success": true, 16 "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcmVudWV2Y2lwYyYwIjIsInJvbGU0IjOT1JNQWwiLCJpc3M1IjJjZmFuQmFja2VudVZCIsImV4cCI6MTYzODY3MjkzMn0.aKNlVMWoC5rC101inPGMKtRjkcI4JJxIZSPAsZgibUg" 17 }</pre> | | | |

GET

{{base_url}}admin/foodmenu

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcmVudWV2Y2lwYyYwIjIsInJvbGU0IjOT1JNQWwiLCJpc3M1IjJjZmFuQmFja2VudVZCIsImV4cCI6MTYzODY3MjkzMn0.aKNlVMWoC5rC101inPGMKtRjkcI4JJxIZSPAsZgibUg

BodyCookiesHeaders (14)Test Results

Status: 403 Forbidden Time: 175 ms Size: 571 B

S

PrettyRawPreviewVisualizeJSON

```
1  {
2    "timestamp": "2021-12-04T02:58:44.700+00:00",
3    "status": 403,
4    "error": "Forbidden",
5    "message": "Forbidden",
6    "path": "/api/admin/foodmenu"
7  }
```