

# YC Tech

## 웹 백엔드 실무 개발 프로젝트



연세대학교 미래교육원  
YC (Yonsei X Codepresso) Tech Academy

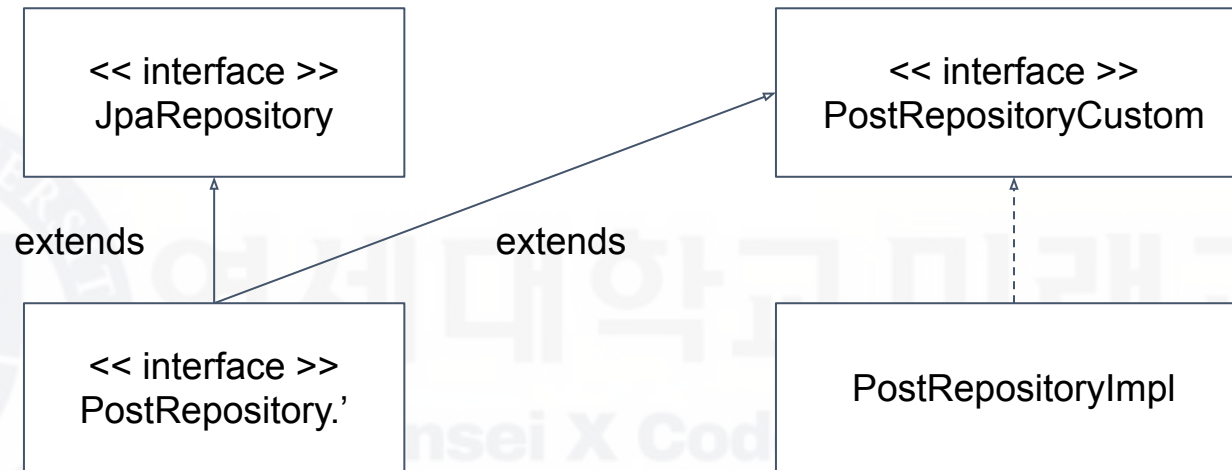
# 3주차 recap



## 복잡한 JPQL 작성 시 이슈

```
@Query("select p from Post p join fetch p.user u "  
      + "where u in "  
      + "(select t from Follow f inner join f.target t on f.source = :user) "  
      + "or u = :user "  
      + "order by p .createdAt desc" )  
List<Post> findAllAssociatedPostsByUser (@Param("user") User user, Pageable pageable);
```

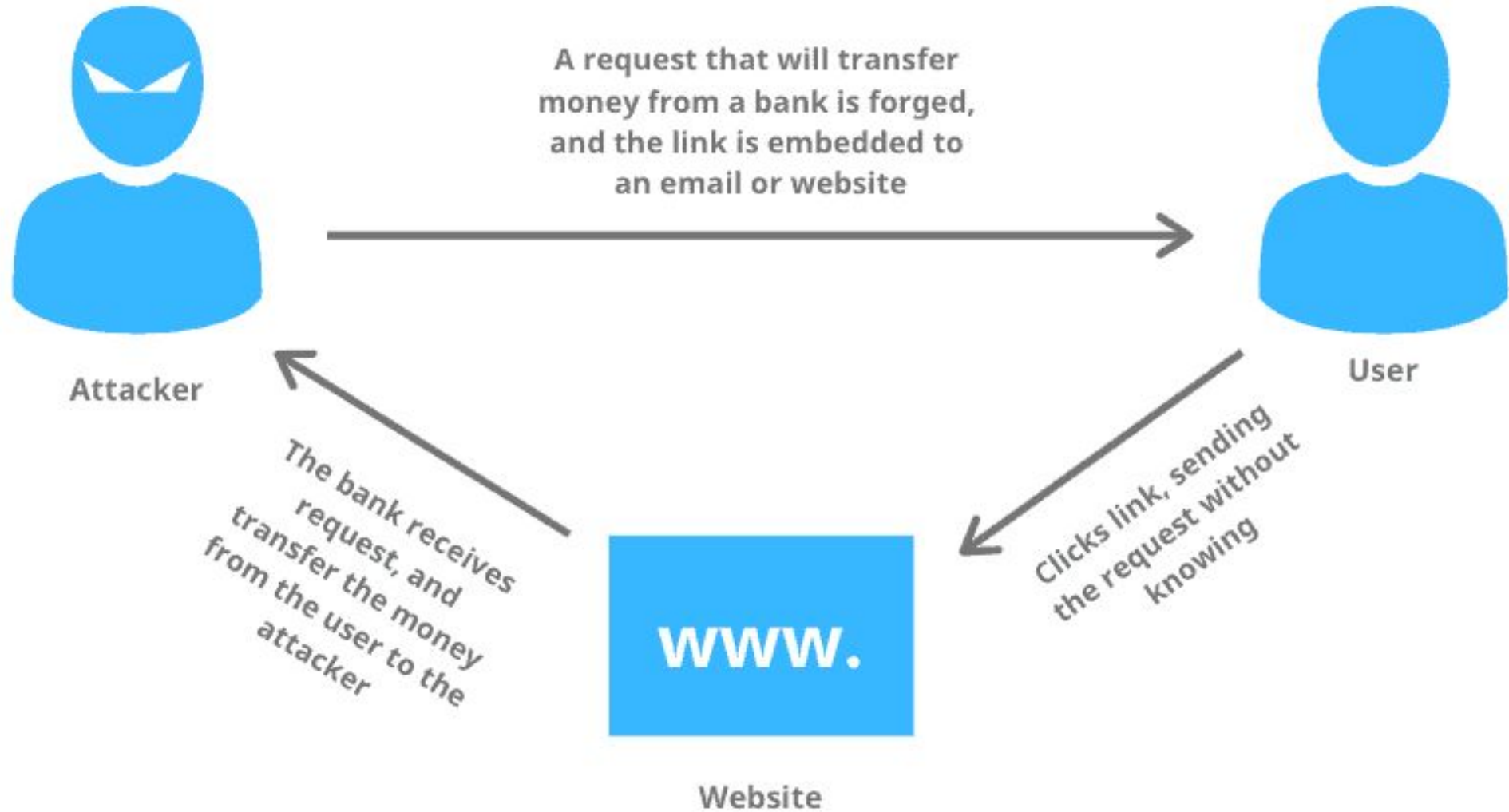
- 간단한 로직을 작성하는데 큰 문제는 없으나, 복잡한 로직의 경우 개행이 포함된 쿼리 문자열이 상당히 길어짐
- JPQL 문자열에 오타 혹은 문법적인 오류가 존재하는 경우, 정적 쿼리라면 어플리케이션 로딩 시점에 이를 발견할 수 있으나 그 외는 런타임 시점에서 에러가 발생
- 동적인 쿼리에 대한 요구사항



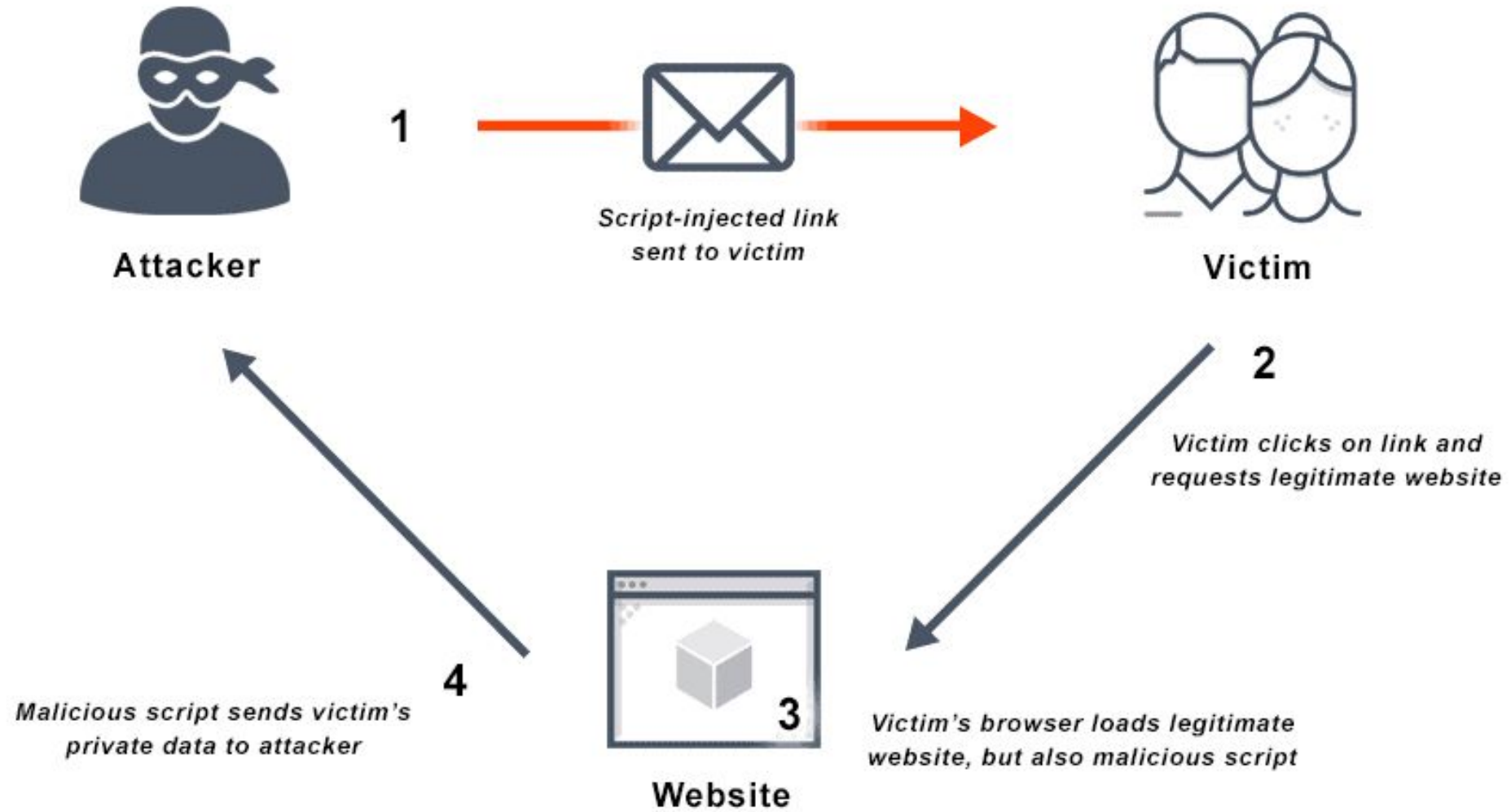
# Security



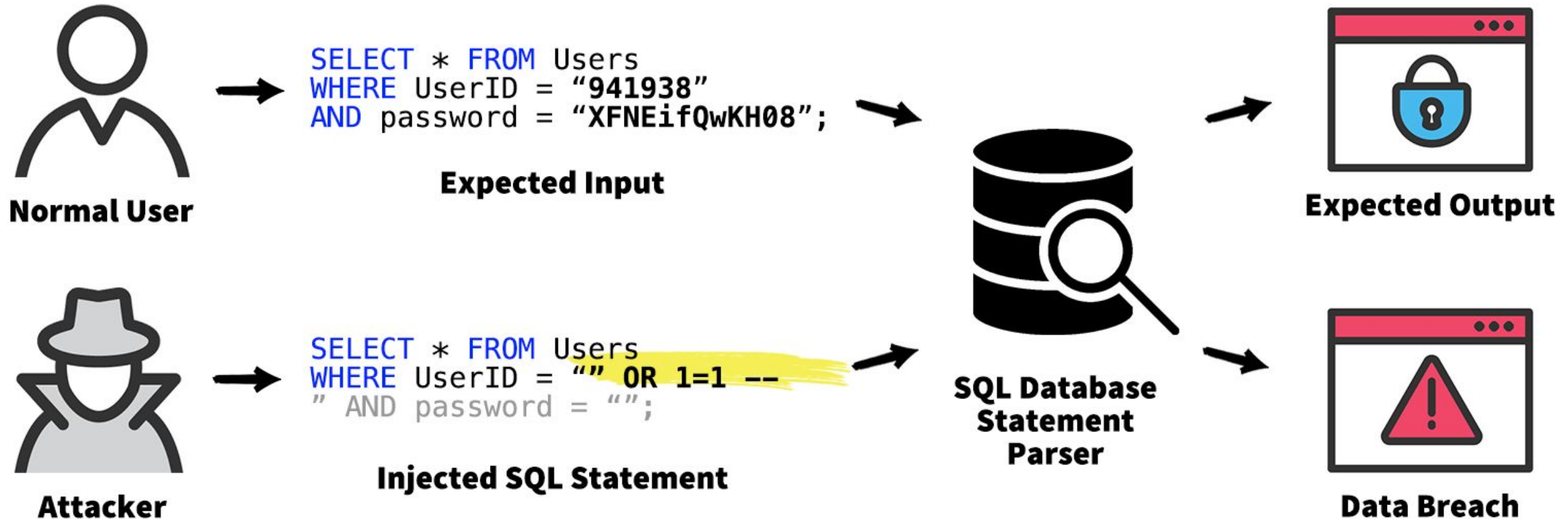
# Cross-Site-Request-Forgery(CSRF)



# XSS



# SQL injection



© TechTerms.com



## 2019

- 1 Broken Object Level Authorization
- 2 Broken User Authentication
- 3 Excessive Data Exposure
- 4 Lack of Resources & Rate Limiting
- 5 Broken Function Level Authorization
- 6 Mass Assignment
- 7 Security Misconfiguration
- 8 Injection
- 9 Improper Assets Management
- 10 Insufficient Logging & Monitoring

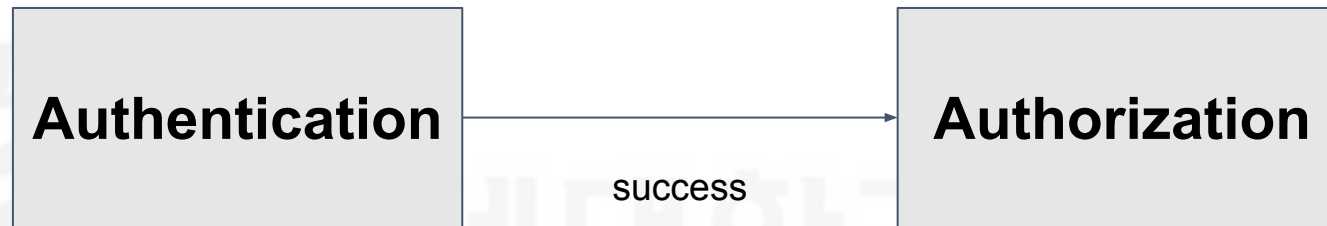
## 2023

- 1 Broken Object Level Authorization
- 2 Broken Authentication
- 3 Broken Object Property Level Authorization
- 4 Unrestricted Resource Consumption
- 5 Broken Function Level Authorization
- 6 Server Side Request Forgery
- 7 Security Misconfiguration
- 8 Lack of Protection from Automated Threats
- 9 Improper Inventory Management
- 10 Unsafe Consumption of APIs

# Spring security

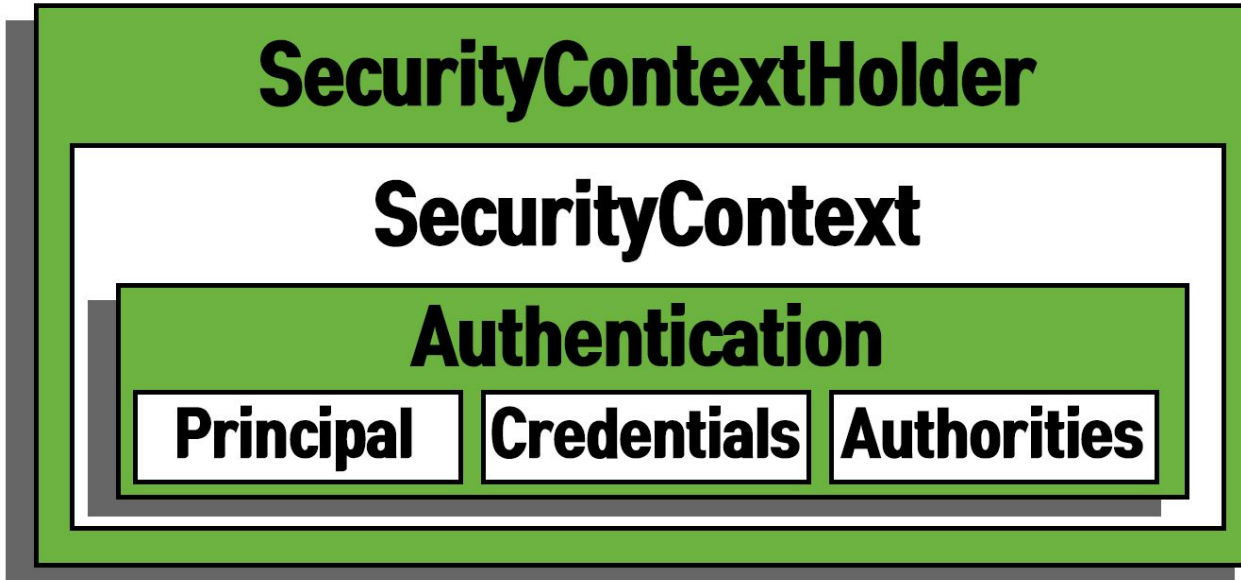


# Authorization, Authentication



- Authentication(인증) : 해당 사용자가 본인이 맞는지 확인하는 절차
- Authorization(인가) : 인증된 사용자가 요청한 자원에 접근 가능한지를 결정하는 절차

# Authentication 관련 module



**SecurityContext** : **Authentication** 을 저장하는 저장소 역할

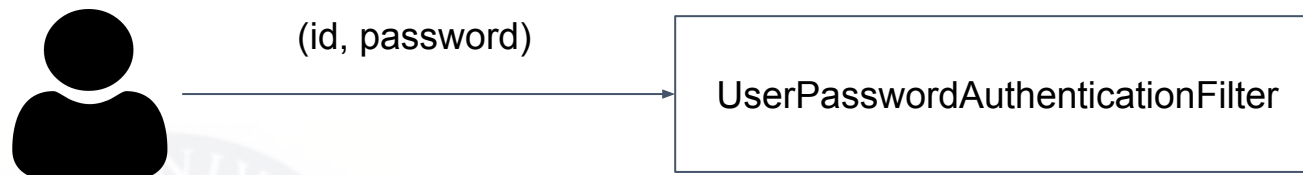
**SecurityContextHolder** : 기본적으로 thread local 로 security context 를 저장

**Authentication** : 접근 주체의 인증 정보와 권한을 다루는 인터페이스

- **Principal** : 접근 주체의 사용자 id 혹은 **User** 객체 저장
- **Credentials** : 비밀번호 정보
- **Authorities** : 인증된 접근 주체자의 권한 목록 저장

# Authentication

```
public interface Authentication extends Principal, Serializable {  
    // 현재 사용자의 권한 목록을 가져옴  
    Collection<? extends GrantedAuthority> getAuthorities();  
  
    // credentials(주로 비밀번호)을 가져옴  
    Object getCredentials();  
  
    Object getDetails();  
  
    // Principal 객체를 가져옴.  
    Object getPrincipal();  
  
    // 인증 여부를 가져옴  
    boolean isAuthenticated();  
  
    // 인증 여부를 설정함  
    void setAuthenticated(boolean isAuthenticated) throws  
    IllegalArgumentException;  
}
```



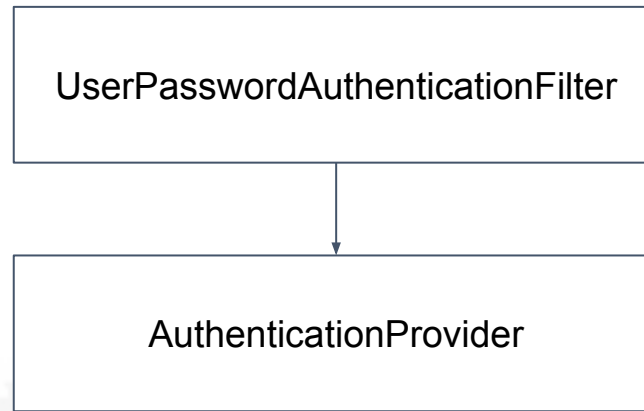
## 1. 로그인 request

- Spring security filter chain 의 UserPasswordAuthenticationFilter 로 id 와 비밀번호 전달



## 2. Authentication 생성

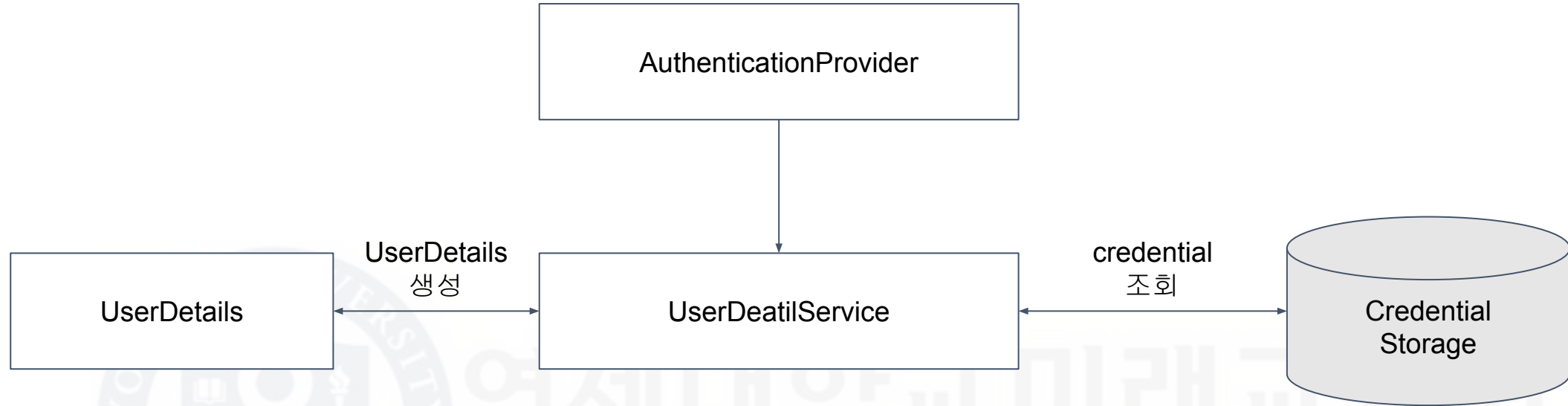
- 요청받은 id 와 비밀번호를 이용하여 UsernamePasswordAuthenticationToken 생성
- UsernamePasswordAuthenticationToken
  - principal 은 Username 등의 신원을 의미
  - credentials 는 Password를 의미



### 3. Authentication 전달

- AuthenticationManager로부터 인증 처리를 위임 받아 실질적인 인증 수행을 담당하는 컴포넌트
- Username/Password 기반의 인증 처리는 DaoAuthenticationProvider 가 담당하고 있으며, DaoAuthenticationProvider 는 UserDetailsService로부터 전달 받은 UserDetails를 이용해 인증을 처리





#### 4. UserDetails 생성/조회

- 데이터베이스 등의 저장소에 저장된 사용자의 **Username**과 사용자의 자격을 증명해주는 크리덴셜 (**Credential**)인 **Password** 그리고 사용자의 권한 정보를 포함하는 컴포넌트
- **AuthenticationProvider** 는 **UserDetails** 를 이용해 자격 증명을 수행

```
public interface UserDetails extends Serializable {

    Collection<? extends GrantedAuthority> getAuthorities(); // (1) 권한 정보
    String getPassword(); // (2) 비밀번호
    String getUsername(); // (3) Username

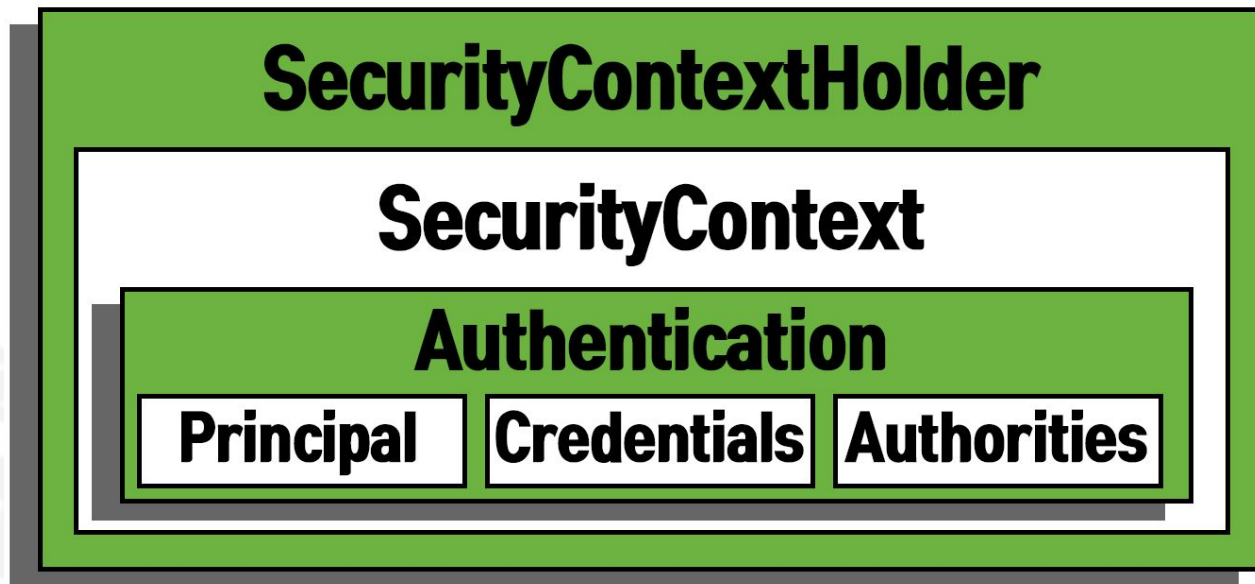
    boolean isAccountNonExpired(); // (4) 사용자 계정의 만료 여부
    boolean isAccountNonLocked(); // (5) 사용자 계정의 Lock 여부
    boolean isCredentialsNonExpired(); // (6) Credentials(Password)의 만료 여부
    boolean isEnabled(); // (7) 사용자의 활성화 여부
}

public interface UserDetailsService {
    UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
}
```



## 4. UserDetails 생성/조회

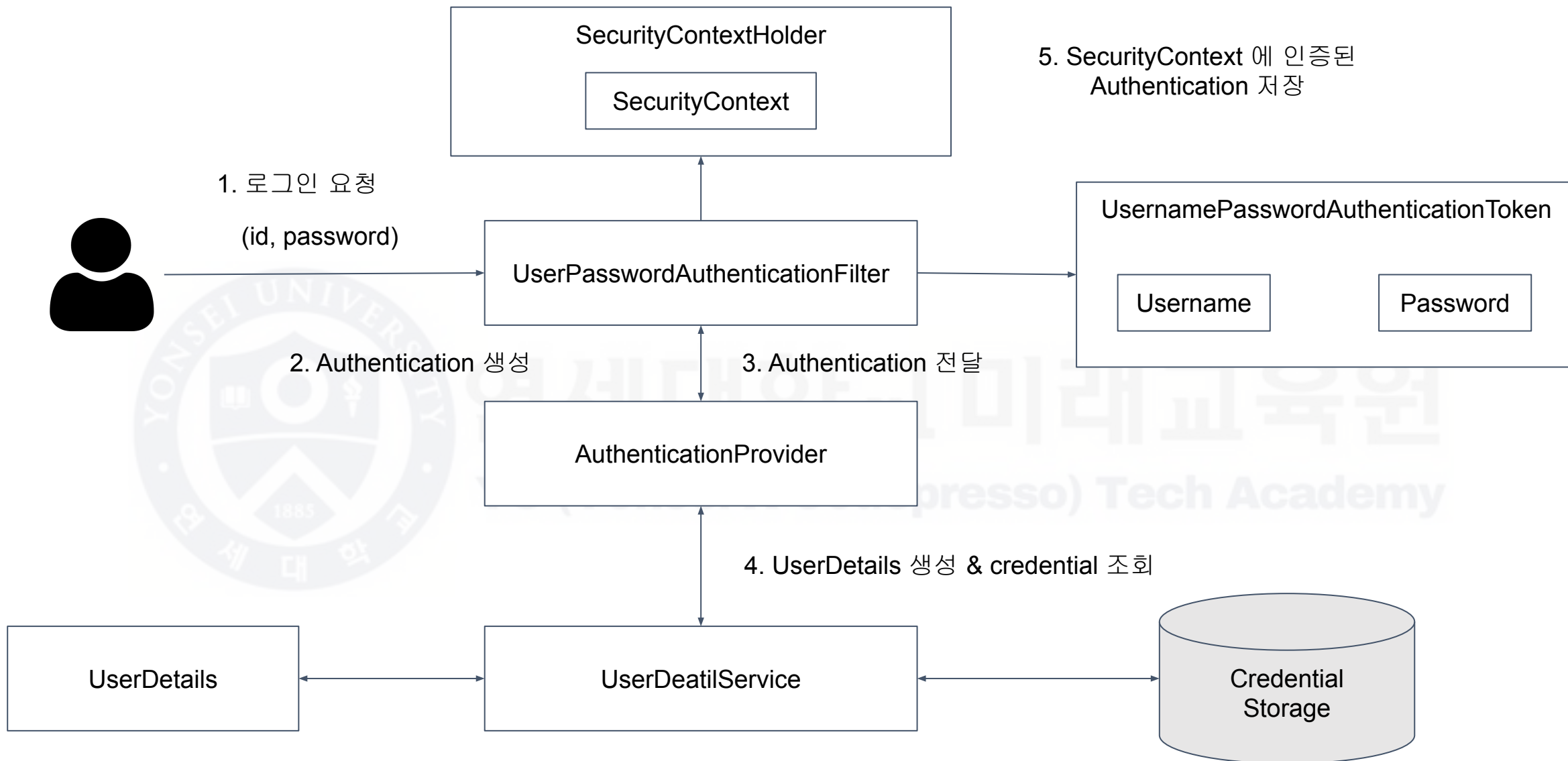
- 데이터베이스 등의 저장소에 저장된 사용자의 **Username**과 사용자의 자격을 증명해주는 크리덴셜 (**Credential**)인 **Password** 그리고 사용자의 권한 정보를 포함하는 컴포넌트
- **AuthenticationProvider** 는 **UserDetails** 를 이용해 자격 증명을 수행
- **UserDetailsService**를 통해 **UserDetails** 로드



## 5. 인증된 Authentication 을 SecurityContext 에 저장

- SecurityContext 는 인증된 Authentication 객체를 저장하는 컴포넌트
- SecurityContextHolder 는 SecurityContext를 관리하는 역할

# 인증 flow



# Session, Token



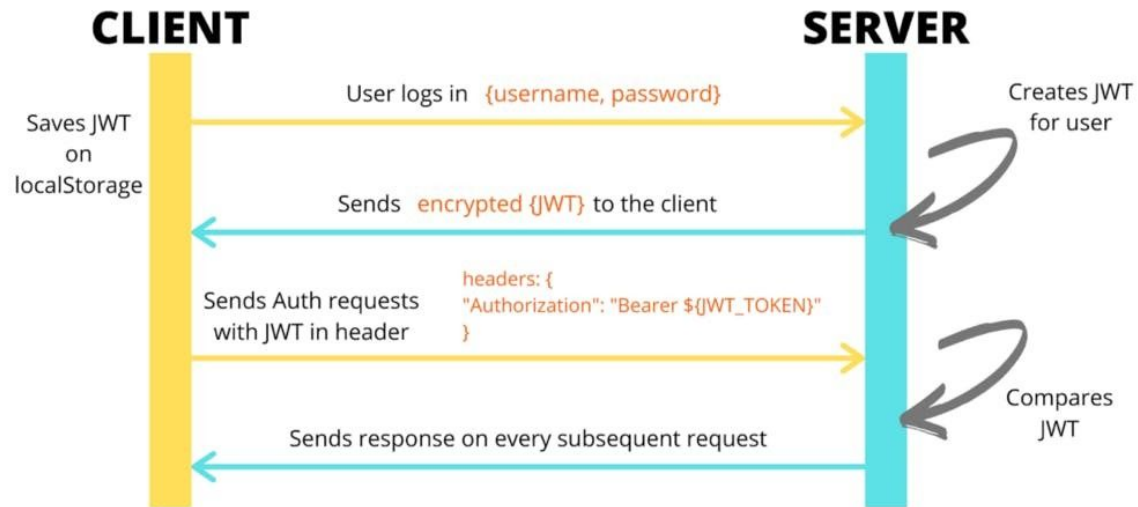
## Session Based Authentication



### Session 기반 인증 방법

- HTTP 프로토콜을 사용하는 인터넷 사용자가 어떤 웹사이트를 방문할 경우, 웹 사이트의 여러 페이지에 걸쳐 사용되는 사용자 정보를 저장하는 방법
- HTTP 프로토콜은 기본적으로 **connectionless, stateless** 하므로 클라이언트에서 로그인 상태를 유지할 방법이 필요함. 이러한 **stateful** 방법이 제공되지 않으면 로그인 정보를 유지할 수 없음
- **session** 정보를 메모리나 디스크, DB 등에 저장

## Token Based Authentication



Token 기반 인증 방법

- Token 기반 인증은 인증 완료 후 클라이언트 측에 저장
- 생성된 토큰을 추적하지 않고 서버가 아는 것은 토큰의 유효성 여부만 관리

# Session vs Token based Authentication

	Session	Token
장점	<ul style="list-style-type: none"><li>+ 서버에서 로그인 상태 확인이 간편</li><li>+ 상대적으로 안전하다. 서버측에서 관리하기 때문에 클라이언트 변조에 영향받거나 데이터의 <b>Stale</b> (손상) 우려가 없다.</li></ul>	<ul style="list-style-type: none"><li>+ 클라이언트에 저장되기 때문에 서버의 메모리에 부담이 되지 않으며 <b>Scale</b> 에 있어 대비책을 고려할 필요가 없다</li><li>+ 멀티 디바이스 환경에 대한 부담이 없다.</li></ul>
단점	<ul style="list-style-type: none"><li>- 유저들의 세션에 대한 정보를 서버 메모리에 들고 있게 된다는 부담이 있다.</li><li>- 서버 메모리에 세션 정보가 저장되기 때문에 <b>Scale Out / Scale In</b> 이 부담이 된다</li><li>- 멀티 디바이스 환경에서 로그인 시 구현이 복잡해진다.</li></ul>	<ul style="list-style-type: none"><li>- 상대적으로 <b>Stale</b> (손상) 의 위험이 크다.</li><li>- 결국 구현을 하다보면 서버측에 <b>token blacklist</b> 를 관리하게 될 가능성이 있고 그렇게 되면 서버측 메모리의 소모가 발생하게 된다</li><li>- <b>XSS</b> 공격에 취약할 수 있어 가능한 민감한 정보는 포함시키지 않을 필요가 있다.</li></ul>



# JWT



## Header.Payload.Signature

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

- JSON 형식의 토큰에 대한 표준 규격인데요. 주로 사용자의 인증(authentication) 또는 인가(authorization) 정보를 서버와 클라이언트 간에 안전하게 주고 받기 위해서 사용
- <https://jwt.io/> 에서 값 확인
- 서버에서는 토큰에 포함되어 있는 서명(signature) 정보를 통해서 위변조 여부를 빠르게 검증

# Json Web Token

## Decoded

EDIT THE PAYLOAD AND SECRET

### HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

### PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

### VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

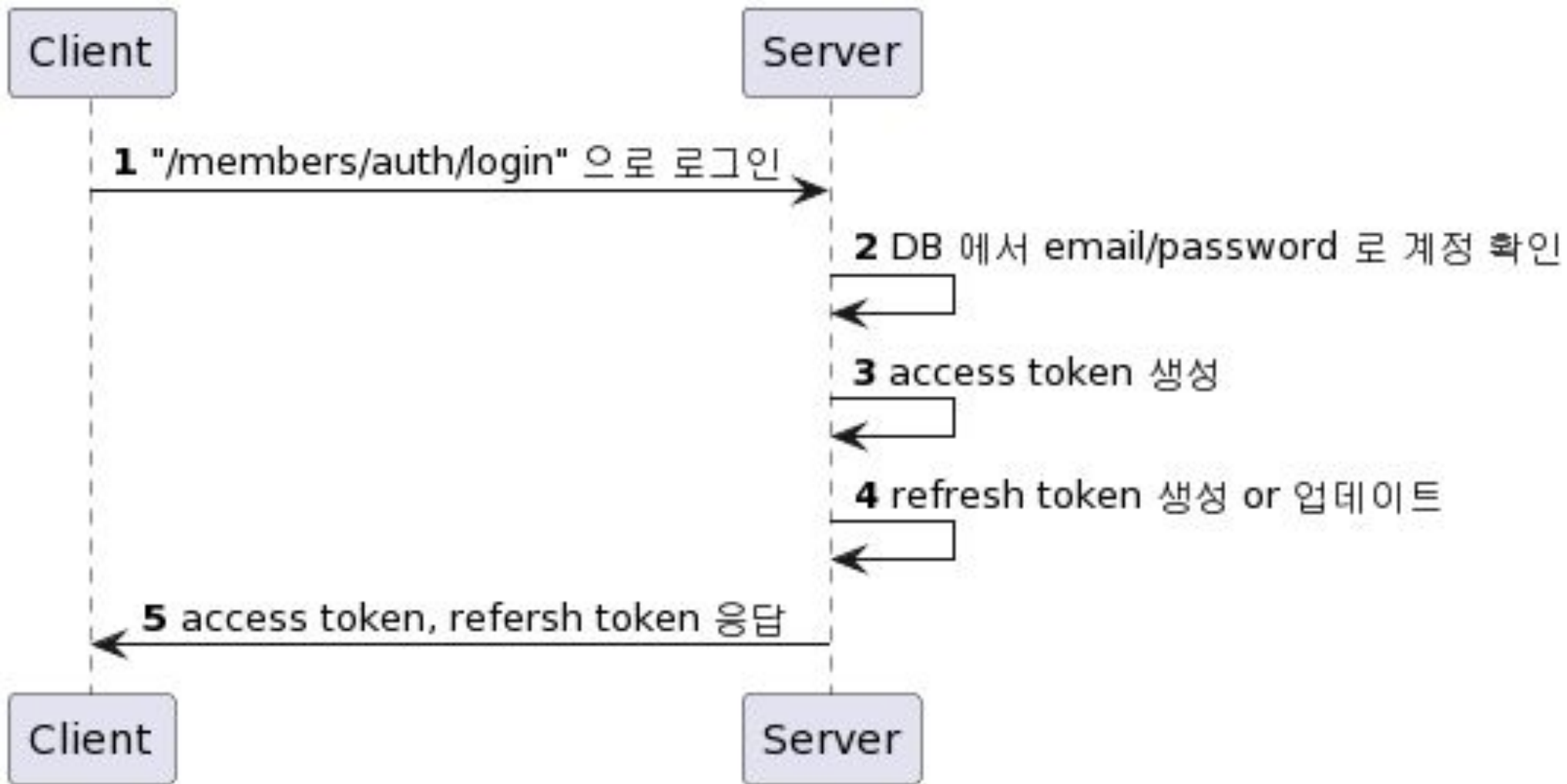


## JWT 키 값

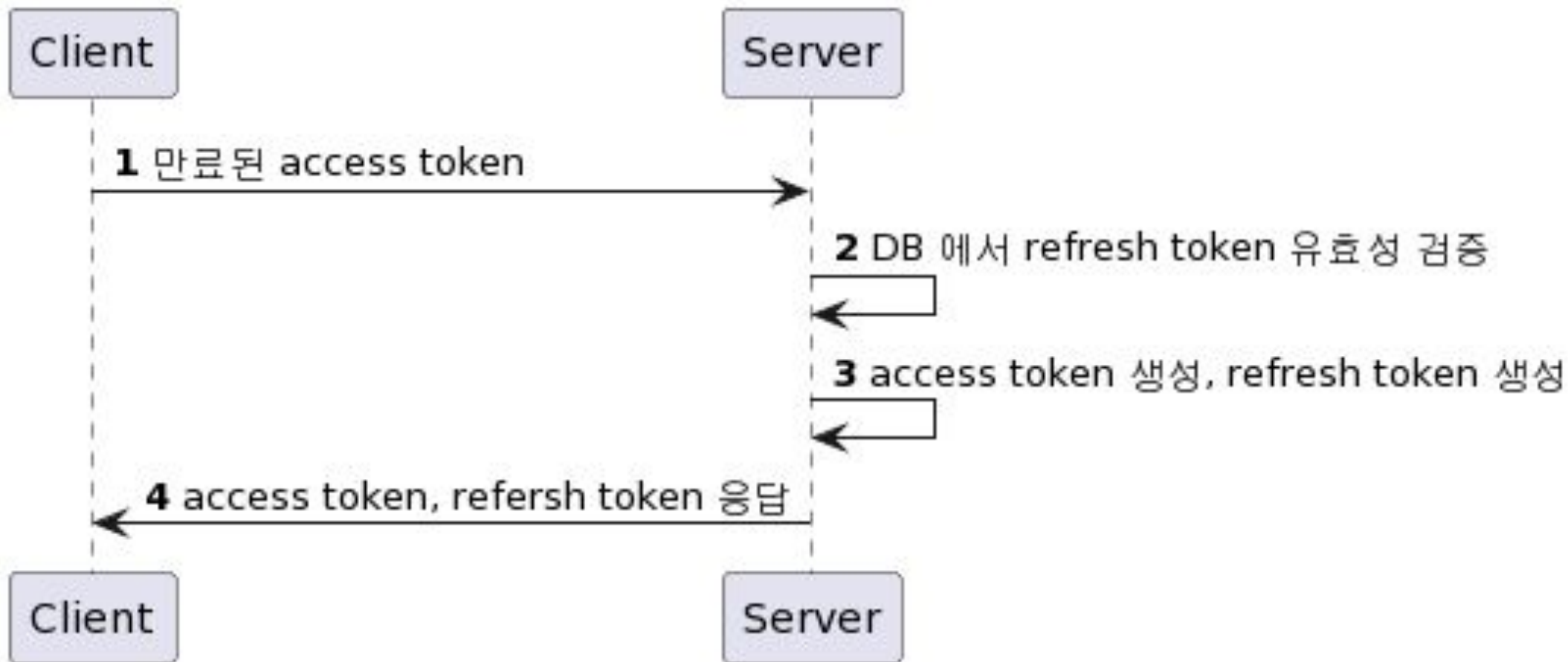
- sub 키: 인증 주체(subject)
- iss 키: 토큰 발급처
- typ 키: 토큰의 유형(type)
- alg 키: 서명 알고리즘(algorithm)
- iat 키: 발급 시각(issued at)
- exp 키: 만료 시작(expiration time)
- aud 키: 클라이언트(audience)

학교  
(Codepresso) Tech Academy

# 정상 로그인 Flow



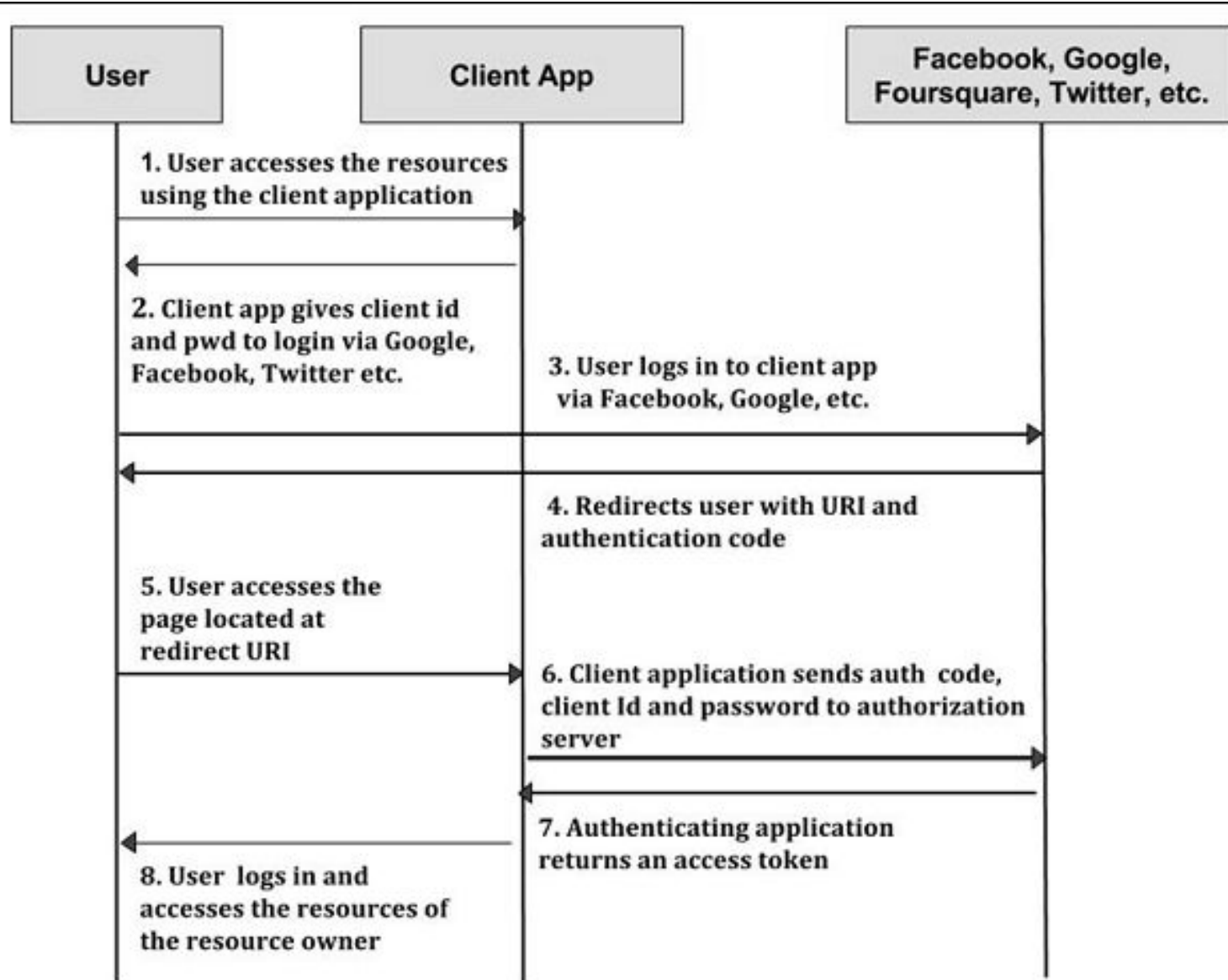
# 만료된 token Flow



# Oauth 2.0



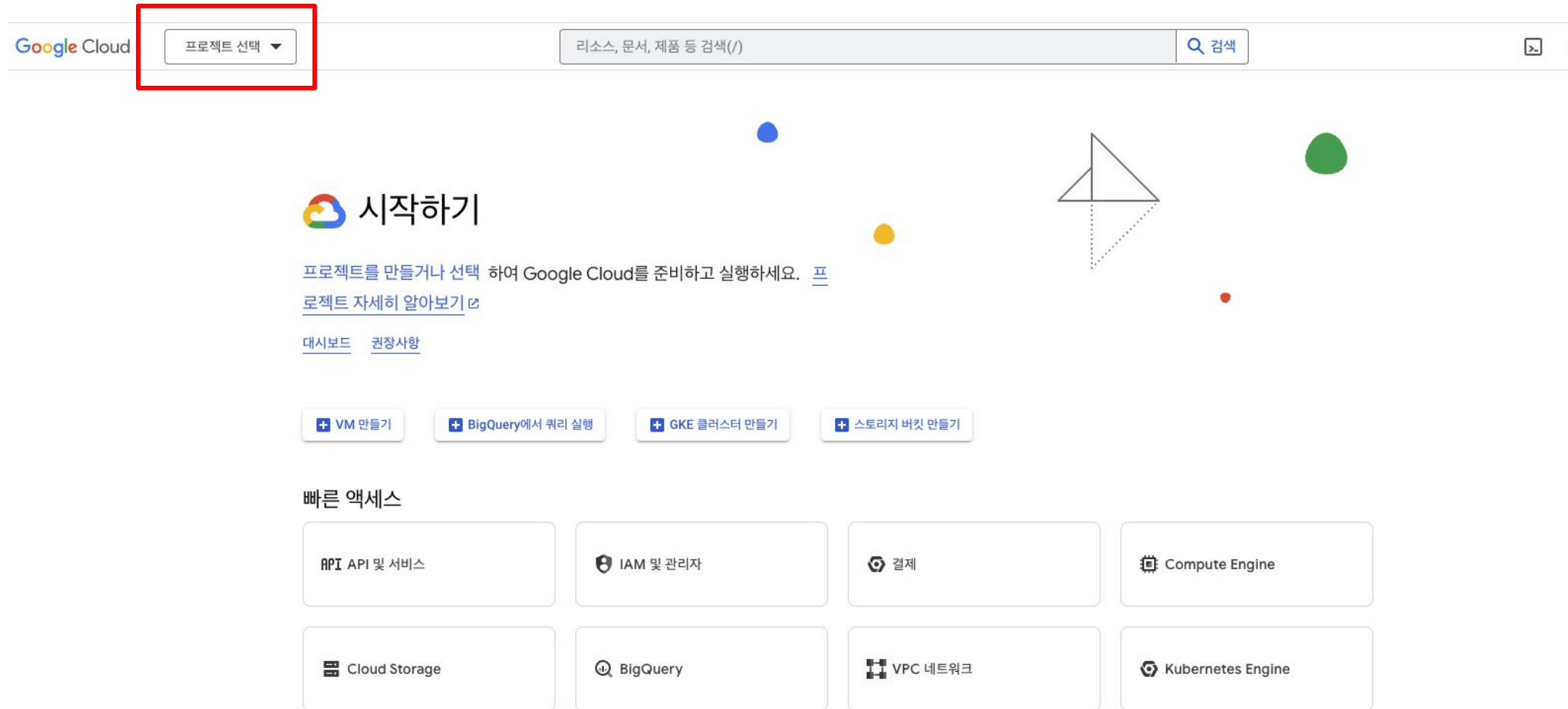
# OAuth2.0 인증 flow



## OAuth2.0

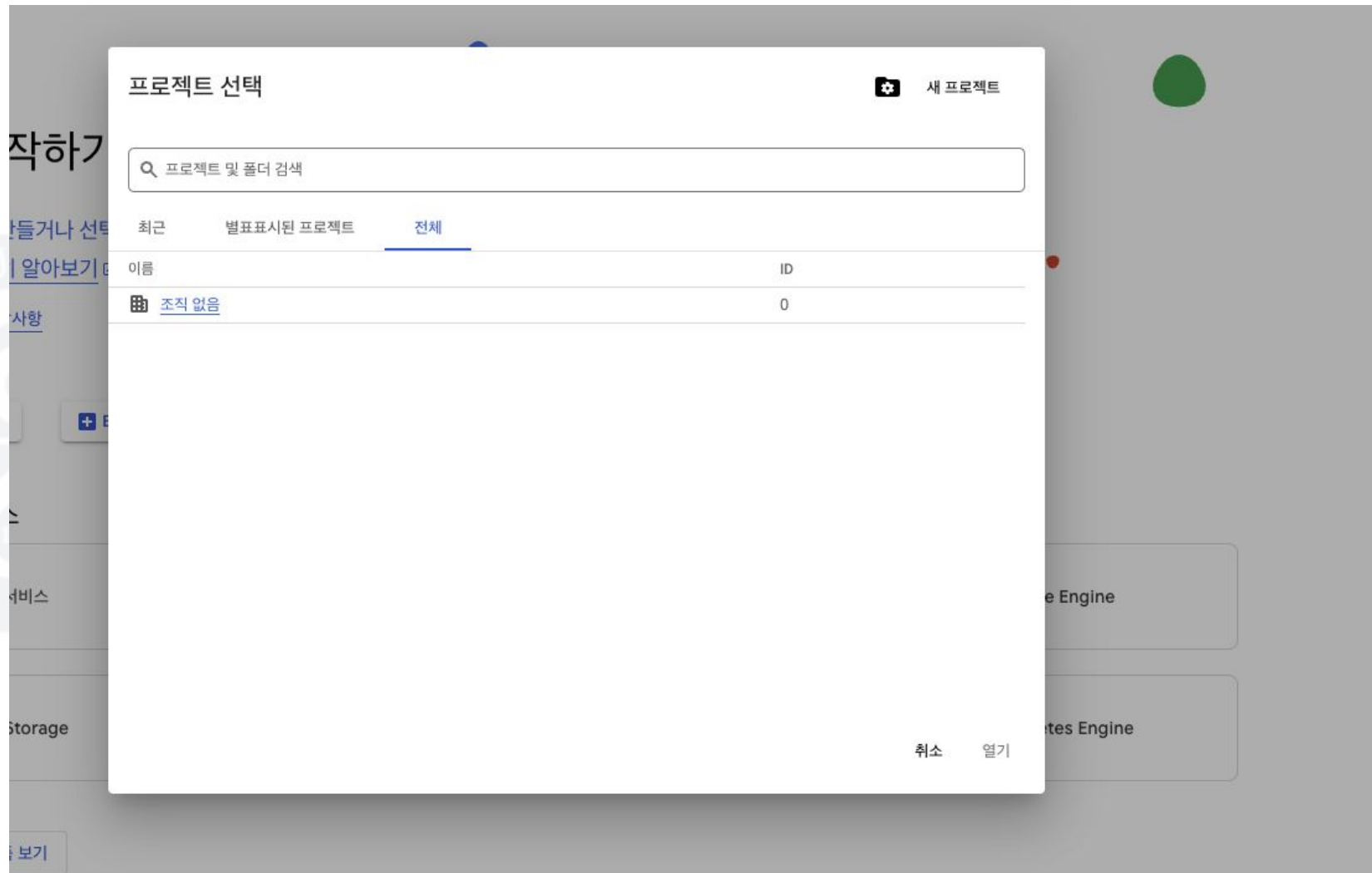
- Third-Party 프로그램에게 리소스 소유자를 대신하여 리소스 서버에서 제공하는 자원에 대한 접근 권한을 위임하는 방식을 제공
- 구글, 페이스북, 카카오, 네이버 등에서 제공하는 간편 로그인 기능도 OAuth2 프로토콜 기반의 사용자 인증 기능을 제공

<https://console.cloud.google.com/>





## ‘새 프로젝트’



## 새 프로젝트



projects 할당량이 12개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 \*

TestApp



프로젝트 ID: testapp-387812입니다. 나중에 변경할 수 없습니다. [수정](#)

위치 \*

조직 없음

[찾아보기](#)

상위 조직 또는 폴더

만들기

취소

육원  
academy

빠른 액세스

**API** API 및 서비스



미래교육원  
YC (Yonsei X Codepresso) Tech Academy

Google Cloud

TestApp

리소스, 문서, 제품 등 검색(/)

API API 및 서비스

사용 설정된 API 및 서비스

라이브러리

사용자 인증 정보

OAuth 동의 화면

페이지 사용 동의

사용자 인증 정보

+ 사용자 인증 정보 만들기

삭제

삭제된 사용자 인증 정보 복원

사용 설정한 API에 액세스하려면 사용자 인증 정보를 만드세요. [자세히 알아보기](#)

⚠

애플리케이션에 대한 정보를 포함하여 OAuth 동의 화면을 구성해야 합니다.

API 키

☐

이름

생성일 ↓

표시할 API 키가 없습니다.

OAuth 2.0 클라이언트 ID

☐

이름

생성일 ↓

표시할 OAuth 클라이언트가 없습니다.

서비스 계정

☐

이메일

표시할 서비스 계정이 없습니다.

원  
emy

## ← OAuth 클라이언트 ID 만들기

클라이언트 ID는 Google OAuth 서버에서 단일 앱을 식별하는 데 사용됩니다. 앱이 여러 플랫폼에서 실행되는 경우 각각 자체 클라이언트 ID가 있어야 합니다. 자세한 내용은 [OAuth 2.0 설정](#)을 참조하세요. OAuth 클라이언트 유형을 [자세히 알아보세요](#).



OAuth 클라이언트 ID를 만들려면 먼저 동의 화면을 구성해야 합니다.

동의 화면 구성

## OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

### User Type

☐ 내부 ?

조직 내 사용자만 사용할 수 있습니다. 인증을 위해 앱을 제출할 필요는 없습니다. [사용자 유형 자세히 알아보기](#)

☒ 외부 ?

Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로덕션에 푸시할 준비가 되면 앱을 인증해야 할 수도 있습니다. [사용자 유형 자세히 알아보기](#)

만들기

OAuth 경험에 대한 [의견 보내기](#)

## 범위 추가 삭제

×

선택한 범위 업데이트

i

아래에는 사용 설정된 API의 범위만 나와 있습니다. 이 화면에 누락된 범위를 추가하려면 [Google API 라이브러리](#)에서 API를 찾아 사용 설정하거나 아래의 '붙여넣은 범위' 텍스트 상자를 사용하세요. 라이브러리에서 사용 설정한 새 API를 확인하려면 페이지를 새로고침하세요.

≡

필터

속성 이름 또는 값 입력

?

<input type="checkbox"/>	API ↑	범위	사용자에게 표시되는 설명
<input checked="" type="checkbox"/>		.../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
<input checked="" type="checkbox"/>		.../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
<input type="checkbox"/>		openid	Google에서 내 개인 정보를 나와 연결
<input type="checkbox"/>	BigQuery API	.../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account
<input type="checkbox"/>	BigQuery	.../auth/cloud-platform	Google Cloud 데이터 확인, 수정, 구성, 삭제 및 Google 계정의 이

호스트/login/oauth2/code/google 로 설정

## 승인된 리디렉션 URI

웹 서버의 요청에 사용

URI 1 \*

http://localhost:8080/login/oauth2/code/google

+ URI 추가



## 클라이언트 ID, 클라이언트 보안 비밀번호 복사

### Additional information

클라이언트 ID

생성일

2023년 10월 6일 PM 8시 37분 18초 GMT+9

### 클라이언트 보안 비밀번호

클라이언트 보안 비밀번호를 변경하는 중인 경우 다운타임 없이 수동으로 순환할 수 있습니다. [자세히 알아보기](#)

클라이언트 보안 비밀번호

생성일

2023년 10월 6일 PM 8시 37분 18초 GMT+9

상태

✓ 사용 설정됨

+ ADD SECRET

원  
my

# Notes

- ❑ Spring security 를 사용하지 않고도 구현 가능
- ❑ 요구사항에 맞춰 security 설정
- ❑ 서비스가 요구하는 보안 수준과 사용자 편의에 맞게 구현

A portrait of Mark Zuckerberg, smiling and looking directly at the camera. He has short brown hair and is wearing a dark-colored t-shirt. The background is a solid dark blue-grey color.

“Done is better  
than perfect.”

- Mark Zuckerberg

# To DO

- ❑ 사용자 Entity & Repository 구현
- ❑ API 보안 구현
- ❑ 팀 프로젝트 인증/회원 기능 구현



The background is a solid blue color with several abstract geometric elements. In the top-left and bottom-right corners, there are circular halftone patterns of varying densities. Diagonal lines in a lighter shade of blue cross the frame. On the right side, there are two thin, light-blue circles. The overall aesthetic is modern and minimalist.

EOD