마차챠 세미나 2회

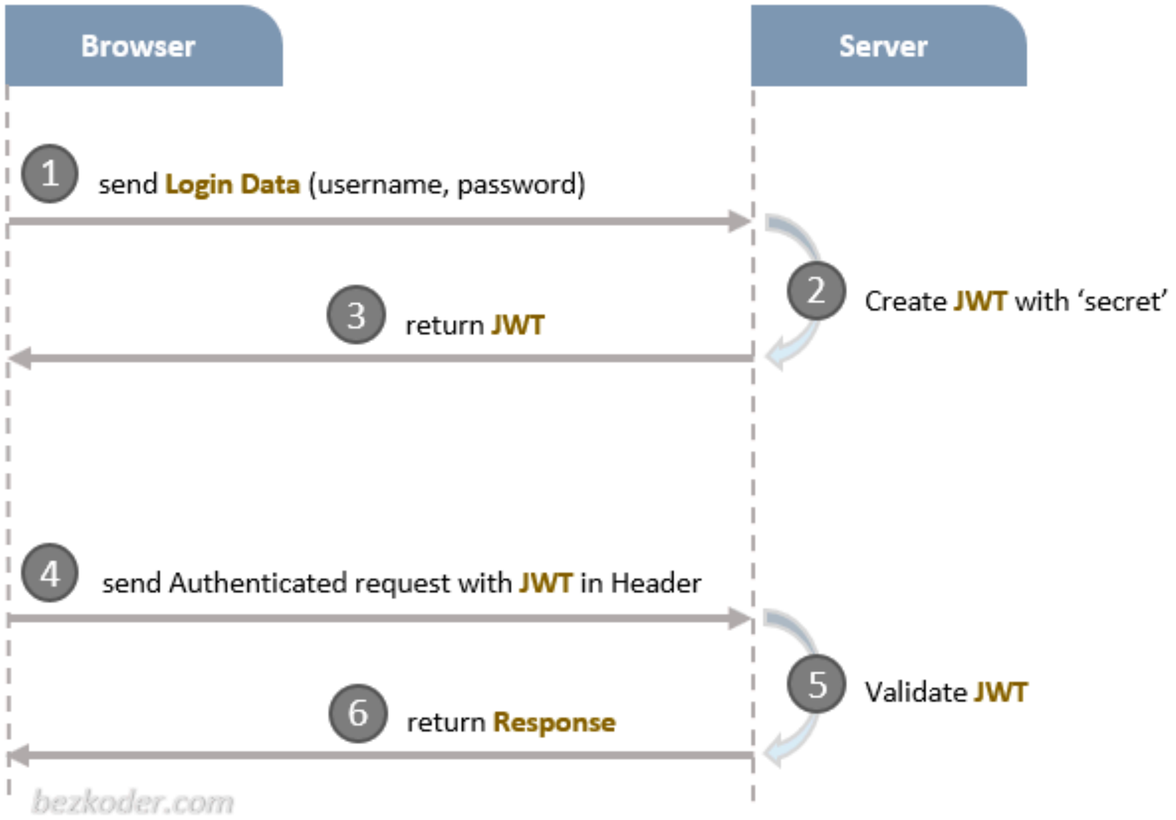
로그인 인증 방식 JWT, Access/Refresh token, Ouath

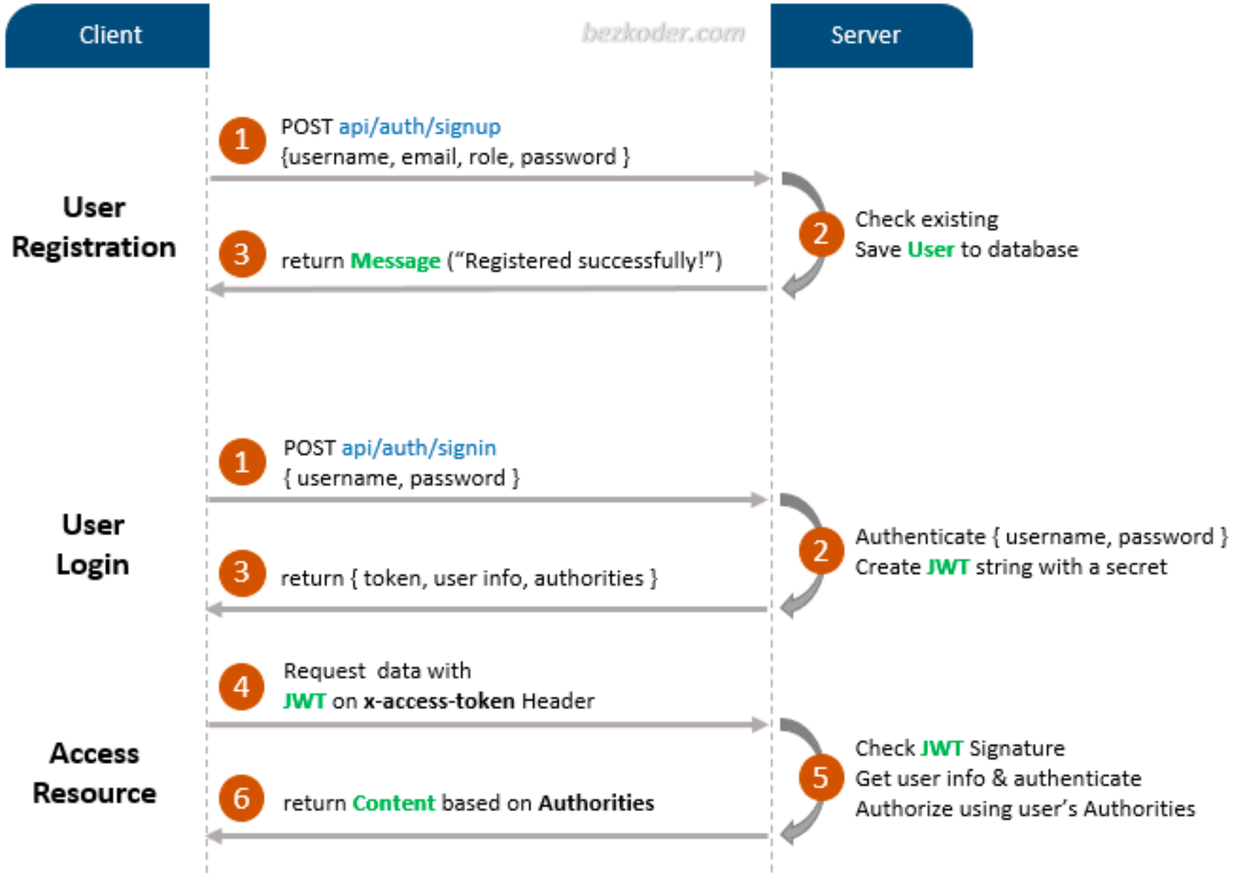
허두영 23/02/02

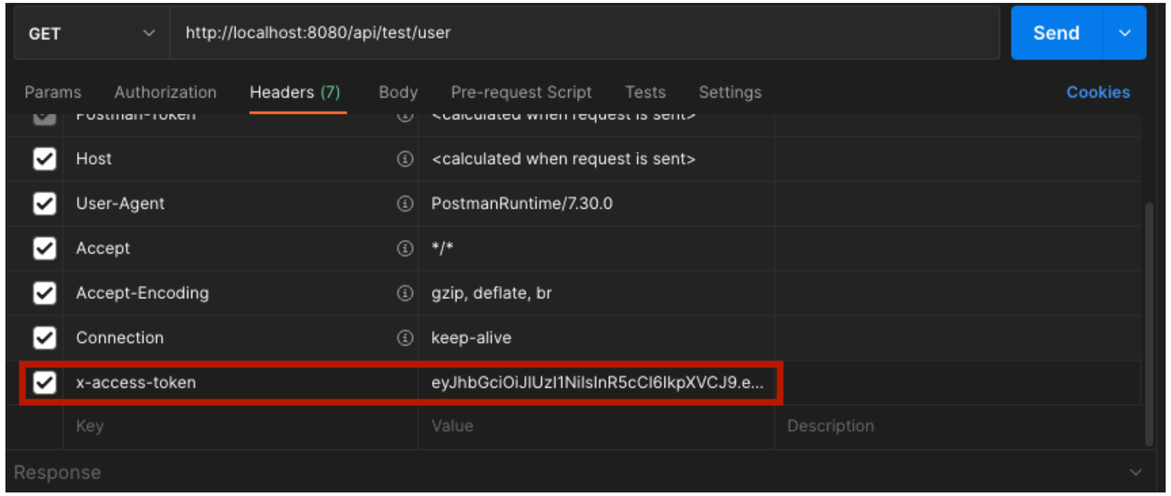
JWT 인증 방식

- JWT = Json Web Token
- JWT 구성요소
  - Header : 암호화알고리즘 + 토큰유형정보
  - Payload : 클레임(정보조각) <- 이메일, 이름 ,토큰 만료기간
  - Signature : 암호화된 구조
- jwt토큰을 발급 받은 후 서버와 클라이언트가 데이터를 주고 받을 때, 별도의 인증과정 없이 HTTP request에 JWT 토큰을 넣은 후 정보를 인증받으면 올바른 HTTP response를 받을 수 있습니다.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyaWQiOiJhYWFAaGlzdC5jby5rciIsInVzZXJubSI6Iuq5gOyGjOumrCIsImlhdCI6MTY0ODYyMjQ3NiwiZXhwIjoxNjQ4NjIyNjU2fQ.iIs572_gMS4o5xWfBRlLZScnDGQgddJwvDOcpXbt-Vc


Browser
Server
1
send Login Data (username, password)
2
Create JWT with 'secret'
3
return JWT
4
send Authenticated request with JWT in Header
5
Validate JWT
6
return Response
bezkoder.com


Client
bezkoder.com
Server
User Registration
1
POST api/auth/signup
{username, email, role, password }
2
Check existing
Save User to database
3
return Message ("Registered successfully!")
User Login
1
POST api/auth/signin
{ username, password }
2
Authenticate { username, password }
Create JWT string with a secret
3
return { token, user info, authorities }
Access Resource
4
Request data with
JWT on x-access-token Header
5
Check JWT Signature
Get user info & authenticate
Authorize using user's Authorities
6
return Content based on Authorities

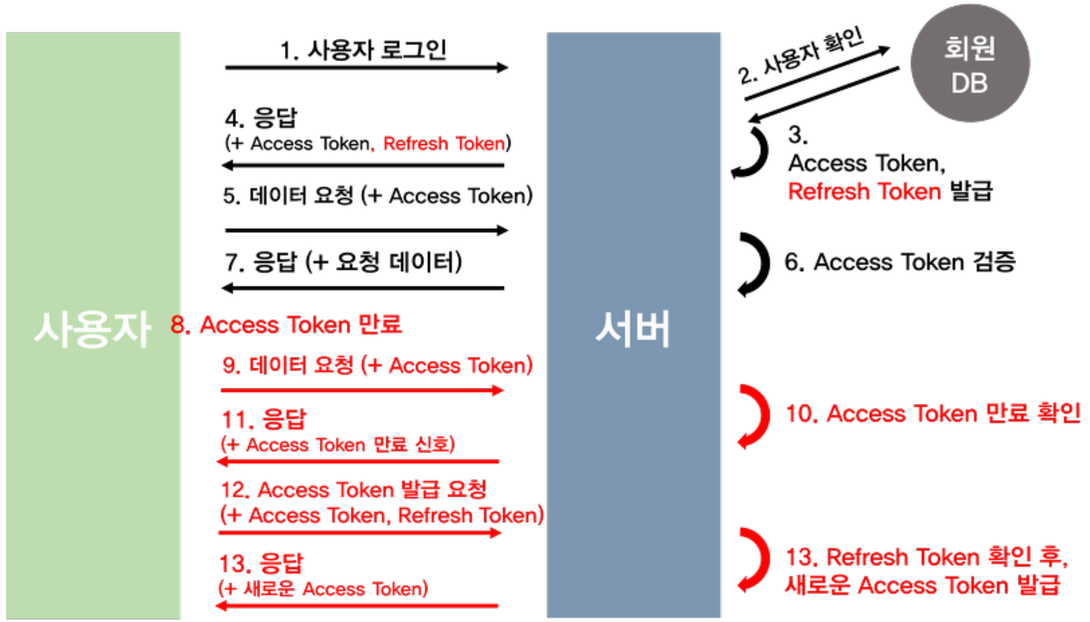

GET
http://localhost:8080/api/test/user
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Host
<calculated when request is sent>
User-Agent
PostmanRuntime/7.30.0
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
x-access-token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e...
Key
Value
Description
Response

JWT의 단점

- JWT decode시 payload에 민감한 정보가 포함됩니다.
- Stateless, 토큰의 상태를 저장하지 않아, 한번 만들어진 토큰을 제어할 수 없습니다.
- 임의로 삭제할 수 없어 만료시간이 중요한데, 만료시간에 대한 정의가 애매합니다.
- 만료시간이 짧으면 잦은 로그인이 필요하고, 만료시간이 길어지면 보안이 취약해집니다.
- 이러한 만료시간 문제를 해결할 방법으로 Access token(jwt)에 refresh token을 추가하는 인증방식이 있습니다

Access token(JWT) + Refresh token 인증 방식

- Refresh token이 추가되어 Jwt보다 더 안전합니다.



토큰

- Refresh token + Access token

액세스 토큰 (Access token)	사용자 인증, 카카오 API 호출 권한 부여	Android, iOS : 12시간 JavaScript: 2 시간 REST API : 6시간
리프레시 토큰 (Refresh token)	액세스 토큰 재발급에 사용 유효한 리프레시 토큰이 있다면 사용자가 매번 카카오계정 정보를 입력하거나 카카오톡으로 로그인하는 인증 절차를 거치지 않아도 액세스 토큰 재발급 가능	2달 만료 시간 1달 남은 시점부터 갱신 가능

- 일반적인 토큰 만료기간
  - access token : 30분 이내
  - refresh token : 2주 정도

- Access token 어디에 저장해야할까? -> 로컬 저장소

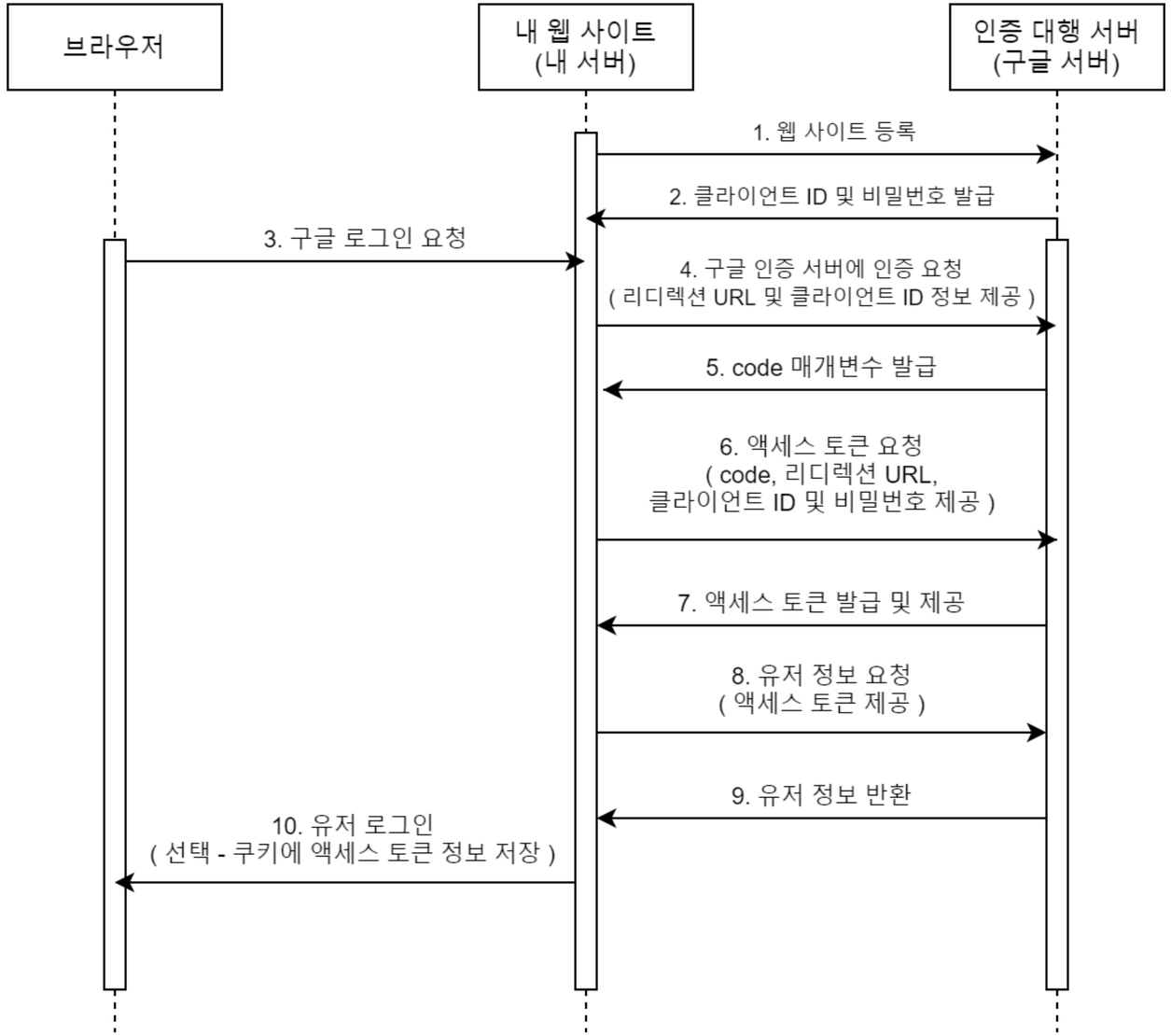
- Refresh token 어디에 저장해야할까? -> 쿠키(HTTP Only), DB
  DB에 실제 토큰을 저장해놓고, DB안 토큰의 위치를 index값이나 해시값으로 로컬스토리지에 저장하는 방식이 있습니다.
  Refresh token은 만료기간이 긴만큼 쉽게 접근할 수 없는 곳에 저장하는 것이 좋다

단점

- 검증 프로세스가 길어 구현이 어렵습니다. (여러번의 콜백함수)
- Access Token이 짧은 만료시간을 가지기 때문에, 만료될때 마다 새롭게 발급이 필요해서 잦은 HTTP Request로 서버의 자원낭비가 심해집니다.
- 만료기간이 긴 Refresh token이 탈취되면 큰 보안문제가 발생할 수 있습니다.

Oauth 란

- 제 3자의 서비스에게 인증과정을 위임하는 방식입니다.
- 앱 서비스 자체에서 id나 pw 입력할 필요가 없습니다.
- Refresh token이 추가되어 Jwt보다 더 안전합니다.

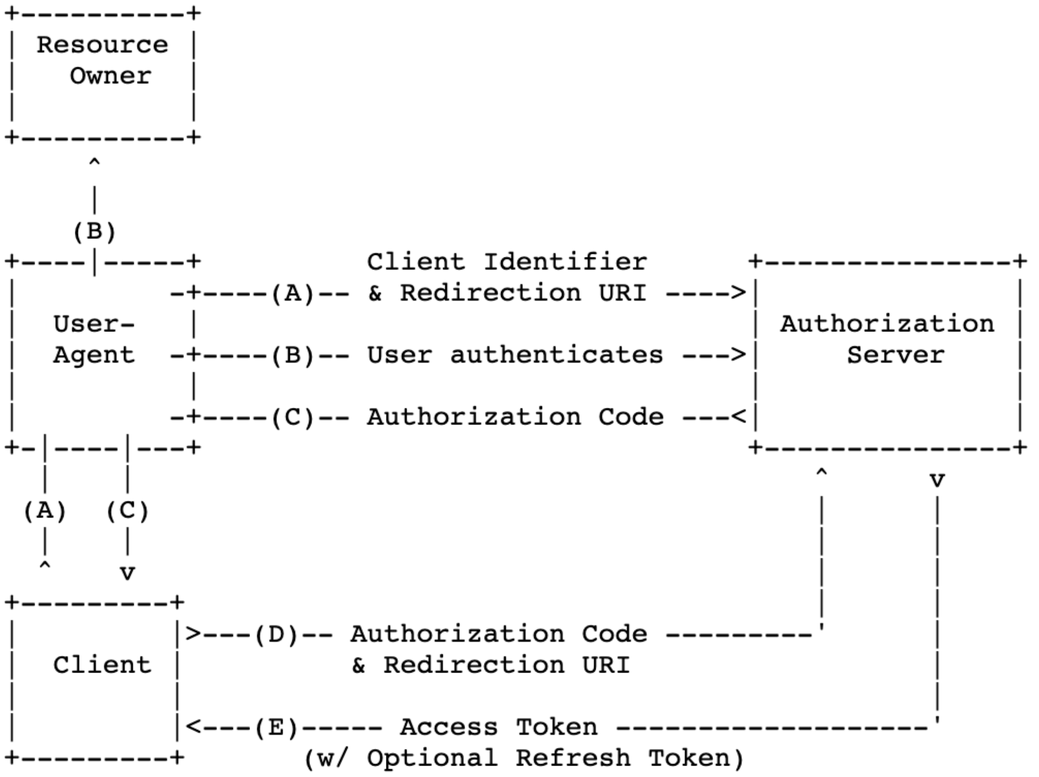


Oauth 인증 방식

- 인증방식
  - GrantImplicit Grant
  - Resource Owner Password Credentials Grant
  - Client Credentials Grant
  - **Authorization Code Grant (가장 많이 사용)**
    - **OAuth 2.0 에서 가장 많이 볼수 있는 유형**

- 구성
  - Client : 앱 서비스
  - Resource Owner : Client를 이용하는 사용자
  - Resource Server : 제 3자의 서비스를 제공해주는 API 서버
  - Authorization Server : 제 3자의 서비스 권한을 부여해주는 인증 서버

- 추가 용어
  - Redirect URI : 제 3자 서비스에 로그인 페이지에서 로그인을 성공하고, Client로 돌아갈 때, 그 돌아갈 주소
  - Authentication Code : Authorization Server 에서 Resource Server를 이용할 수있도록 승인해주는 코드 (토큰 발급받기 전)



- Client ID : 764763960808004
- Client Secret : **********

Continue with Facebook

Client

https://facebook.com?
client_id= 764763960808004&scope=public_profile&
redirect_url=tinderef468.firebaseapp.com/__/auth/handler

Resource Owner

Authorization Server

Resource Server

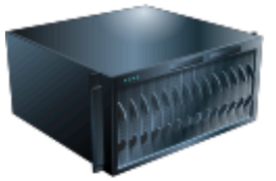
- Client ID : 764763960808004
- Client Secret : **********
- Authorized redirect URI : https://tinder-ef468.firebaseapp.com/__/auth/handler

- Client ID : 764763960808004
- Client Secret : **********

Client

Resource Owner

Authorization Server



Resource Server

- Client ID : 764763960808004
- Client Secret : **********
- Authorized redirect URI : https://tinder-ef468.firebaseapp.com/__/auth/handler
- User_id : 1
- Scope : public_profile

https://tinder-ef468.firebaseapp.com/__/auth/handler&code=3

- Client ID : 764763960808004
- Client Secret : **********
- Authorization_code : 3

Client

Location : https://tinder-ef468.firebaseapp.com/__/auth/handler&code=3

Resource Owner

Authorization Server

Resource Server

- Client ID : 764763960808004
- Client Secret : **********
- Authorized redirect URI : https://tinder-ef468.firebaseapp.com/__/auth/handler
- User_id : 1
- Scope : public_profile
- Authorization_code : 3

- Client ID : 764763960808004
- Client Secret : **********
- Authorization_code : 3

Client

https://facebook.com/token?
grand_type=authorization_code&code=3&
redirect_uri=https://tinder-
ef468.firebaseapp.com/__auth/handler&
client_id=764763960808004&cliend_secret=*****

Resource Owner

Authorization Server

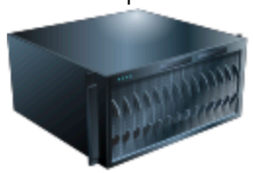
Resource Server

- Client ID : 764763960808004
- Client Secret : **********
- Authorized redirect URI : https://tinder-ef468.firebaseapp.com/__/auth/handler
- User_id : 1
- Scope : public_profile
- Authorization_code : 3

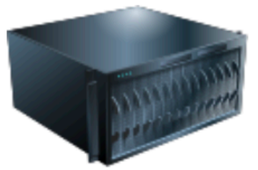
- Client ID : 764763960808004
- Client Secret : **********
- Access Token : 4

Client

Resource Owner

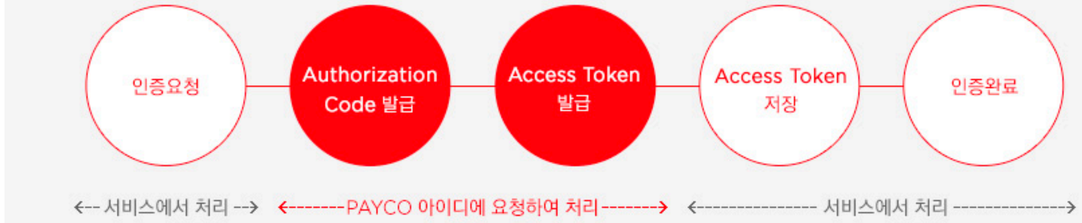


Authorization Server



Resource Server

- Client ID : 764763960808004
- Client Secret : **********
- Authorized redirect URI : https://tinder-ef468.firebaseapp.com/__/auth/handler
- User_id : 1
- Scope : public_profile
- Access Token : 4


인증요청
Authorization Code 발급
Access Token 발급
Access Token 저장
인증완료
<-- 서비스에서 처리 -->
<--------PAYCO 아이디에 요청하여 처리-------->
<---------- 서비스에서 처리 ---------->
사용자
서비스
PAYCO 인증 서비스
PAYCO API 서비스
1. 서비스 접근 및 로그인 요청
2. 로그인 요청
3. 로그인 페이지 제공, ID/PW 요청
4. ID/PW 입력
5. Authorization Code 발급
6. Redirect : Callback URL
7. Access Token 요청
8. Access Token 발급
Token 저장
9. 인증 완료 및 로그인 성공
10. 서비스 요청
11. Access Token으로 API 호출
12. Access Token 검증 및 서비스 제공
13. 서비스 제공

끝..

- Oauth 생활코딩 강의 : https://www.youtube.com/watch?v=hm2r6LtUbk8