多奇·教育訓練

ASP.NET Core 5 開發實戰

權限控管篇 (OpenID Connect 與 IdentityServer 4)

多奇數位創意有限公司

技術總監 黃保翕（Will 保哥）

https://blog.miniasp.com

OAuth 2.0: Introduction

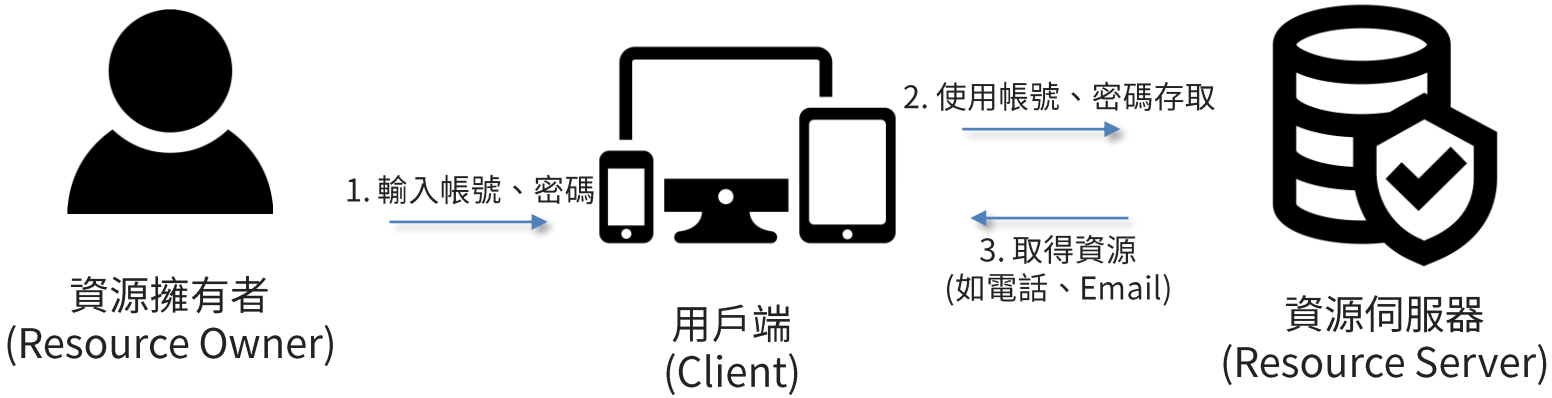
簡介 OAuth 2.0

關於 OAuth 2.0

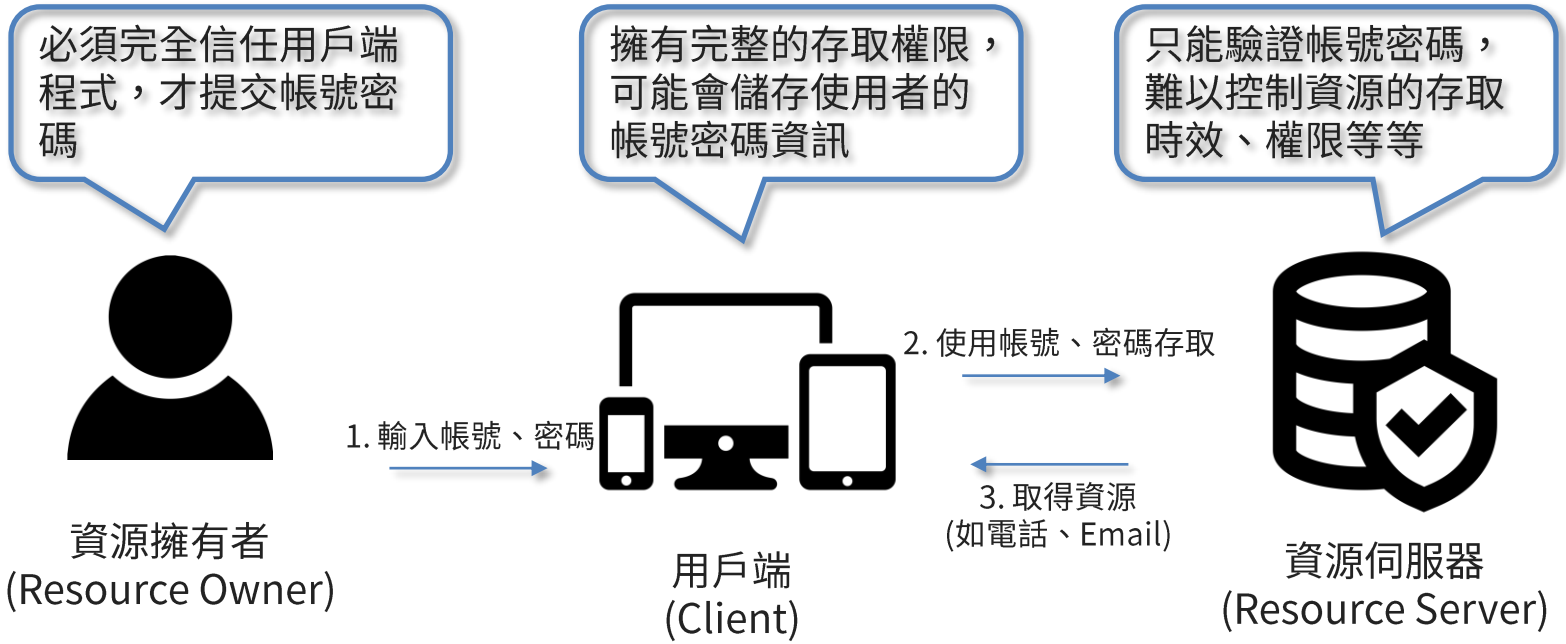
- **OAuth 2.0** 是 RFC 6749 定義的一種**授權規格**，任何程式語言都可以依照此規格實作出自己的授權系統。

- 目的：
  - 控制 **資源擁有者** (Resource Owner) 的**存取權限**
  - 有 **範圍性** (Scope) 的保護 **資源** 的存取

傳統的 Client-Server 架構



傳統的 Client-Server 架構的問題



OAuth 2.0 的四個重要角色

資源擁有者
(Resource Owner)

可以授權別人存取被保護的資源，一般指的是使用者

用戶端
(Client)

負責代表資源擁有者存取資料，通常代表的是「程式」
(ex：瀏覽器或桌面應用程式)

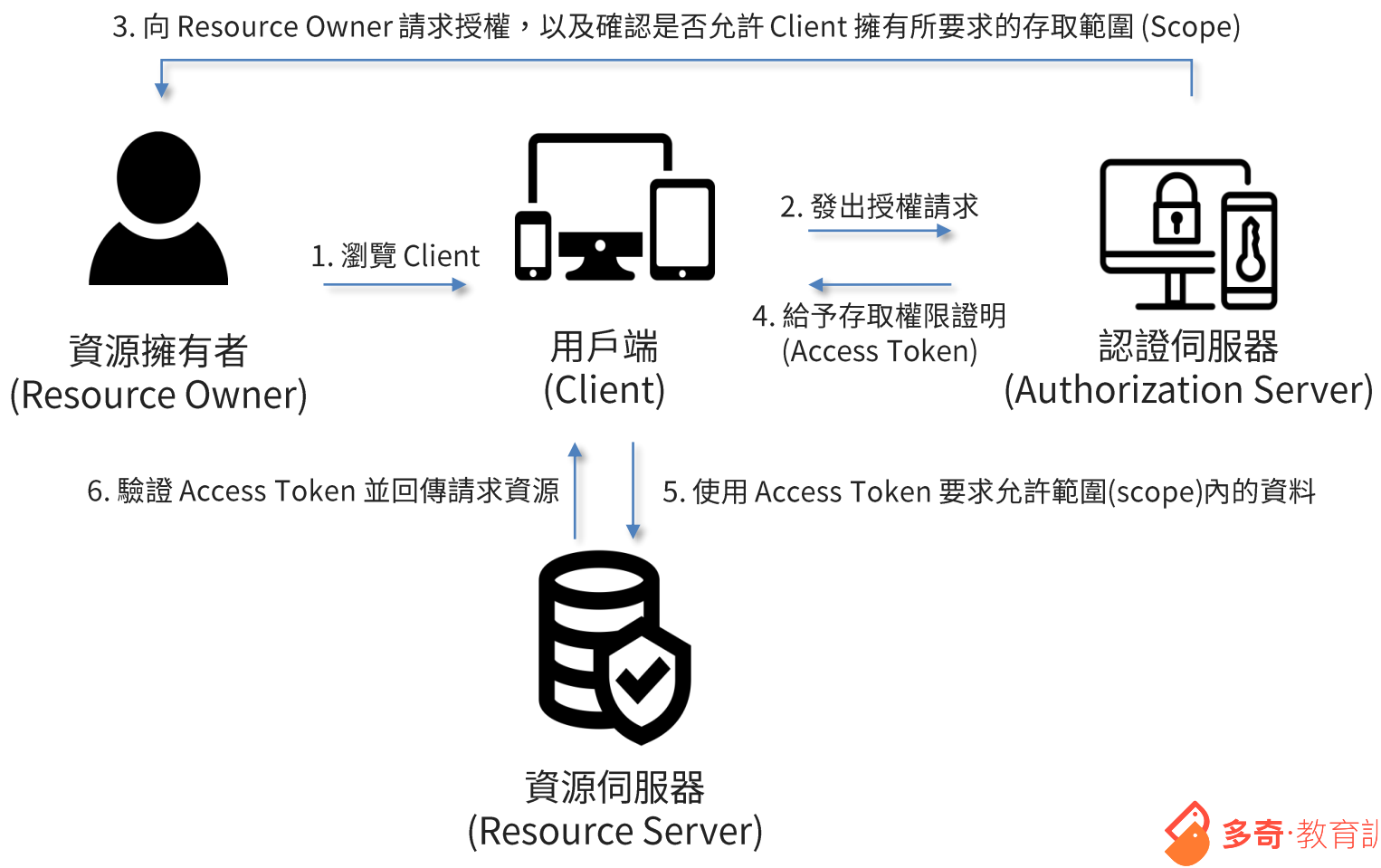
資源伺服器
(Resource Server)

存放受保護資源的伺服器，會依據來源是否有存取權限的證明來給予資料

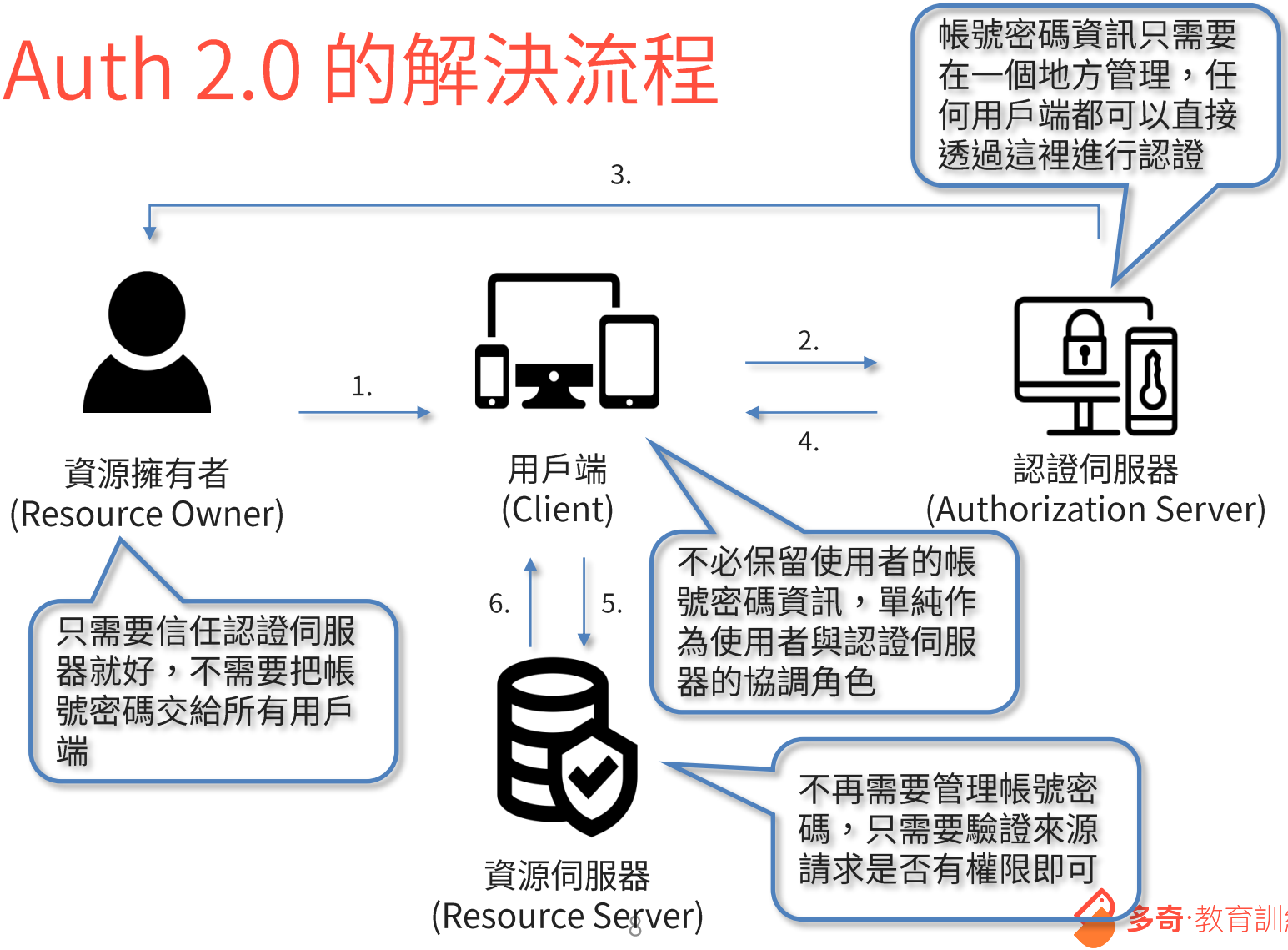
認證伺服器
(Authorization Server)

負責確認是否允許指定的存取範圍，最終核發存取權限的證明(Access Token)

OAuth 2.0 的解決流程



OAuth 2.0 的解決流程



認識 Access Token

- 用來**證明使用者認證資訊**的一組字串，讓 Resource Server 識別使用者，並允許**存取受保護的資源**

- 字串內容可以存放
  - 允許存取的範圍 (Scopes)
  - Access Token 使用期限
  - 使用者識別資訊

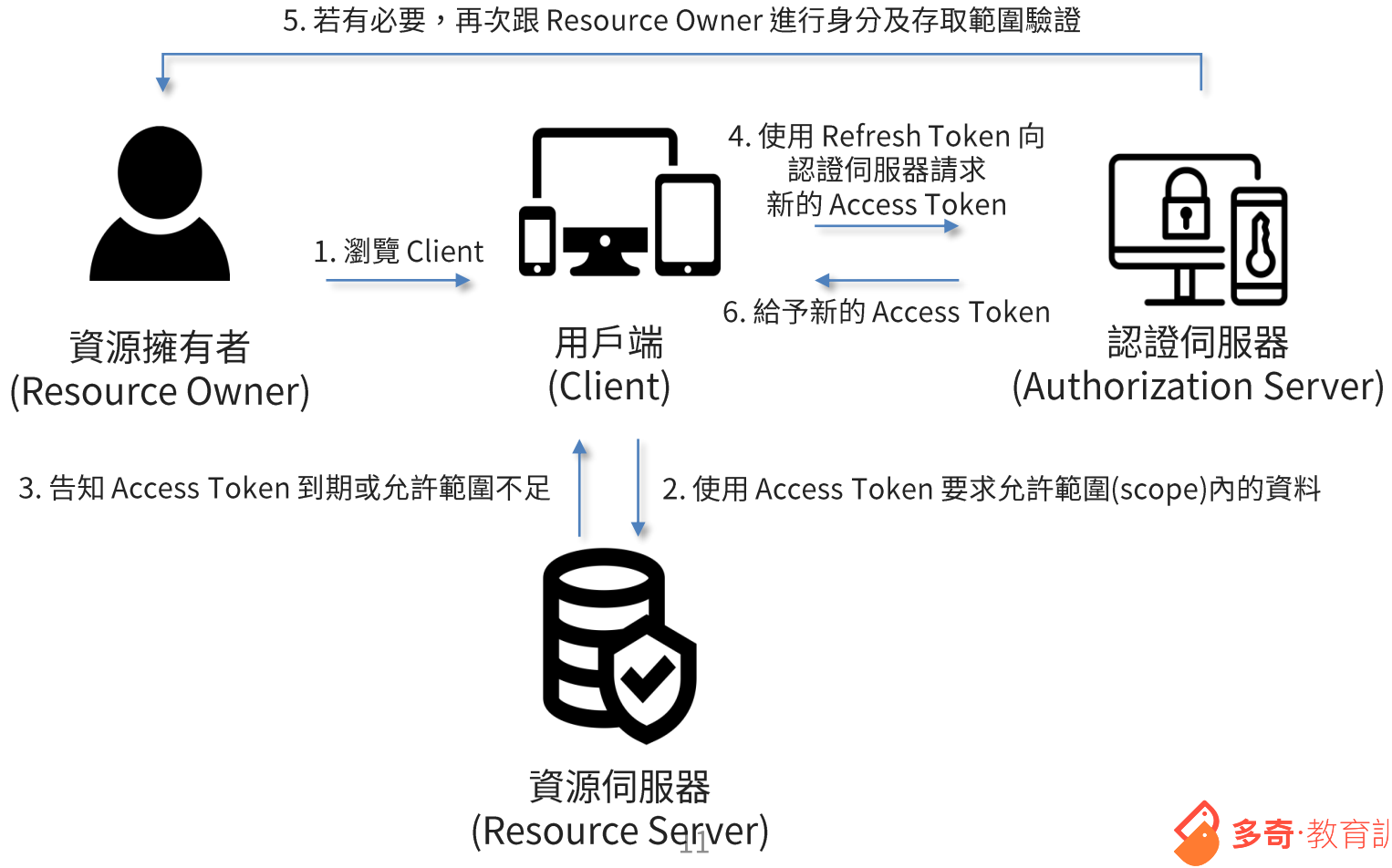
- 不應該存放敏感的使用者資訊，如帳號密碼、信用卡號等等

- 字串可以為任意格式，只須讓 Resource Server 能夠識別即可，例如 Bearer Token (RFC 6750)是一種使用 HTTP 傳輸協定時所規範的資料格式

認識 Refresh Token

- 當 Access Token **到期**或**需要更新允許範圍**時，若希望節省使用者重新請求的次數，會需要使用 Refresh Token 機制
- 使用 Refresh Token 機制時，在核發 Access Token 同時會綁定一組 Refresh Token
- Authorization Server 不一定需要核發 Refresh Token
- Client 可以使用 Refresh Token 協助 Resource Owner 自動完成重新核發的過程
- 重新核發後，**原來的 Refresh Token 會失效**，會隨著新的 Access Token 核發時取得新的 Refresh Token

Refresh Token 流程



OAuth 2.0: Flows

OAuth 2.0 的 4 種授權流程

OAuth 2.0 的 4 種授權流程

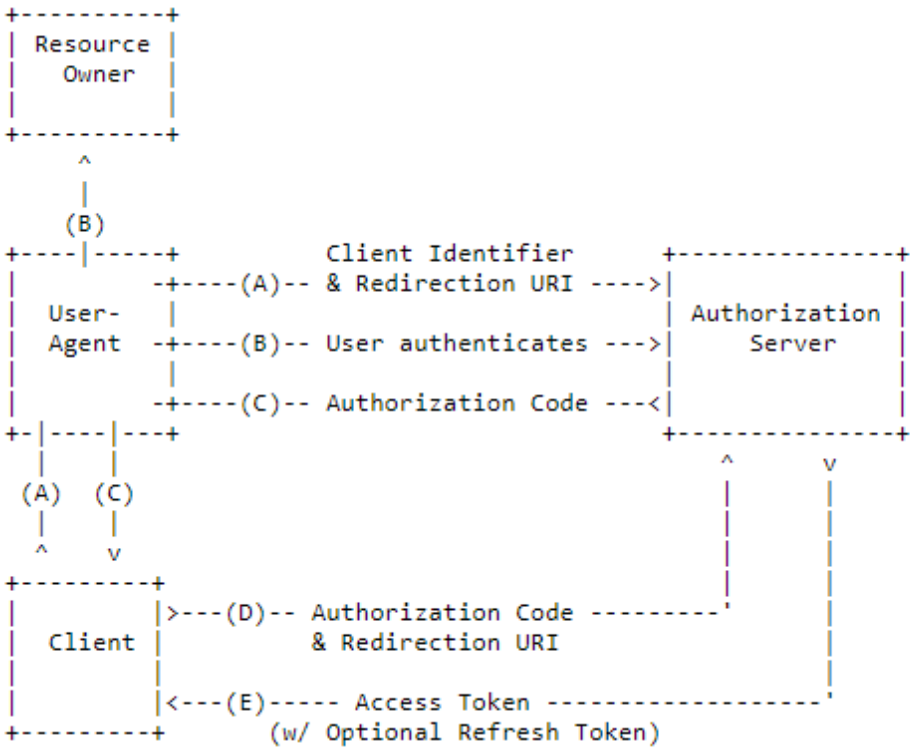
- RFC 6749 中定義了 4 種授權的流程，各種服務提供商必須自行實作，也可以自行擴充新的流程
  - **Authorization Code Grant Type Flow**
    - 授權碼模式
  - **Implicit Grant Type Flow**
    - 簡易模式
  - **Resource Owner Password Credentials Grant Type Flow**
    - 帳號密碼模式
  - **Client Credentials Grant Type Flow**
    - 用戶端認證模式

- 上述 4 種流程，皆可依實際需要進行部分實作。

Authorization Code Grant Type Flow

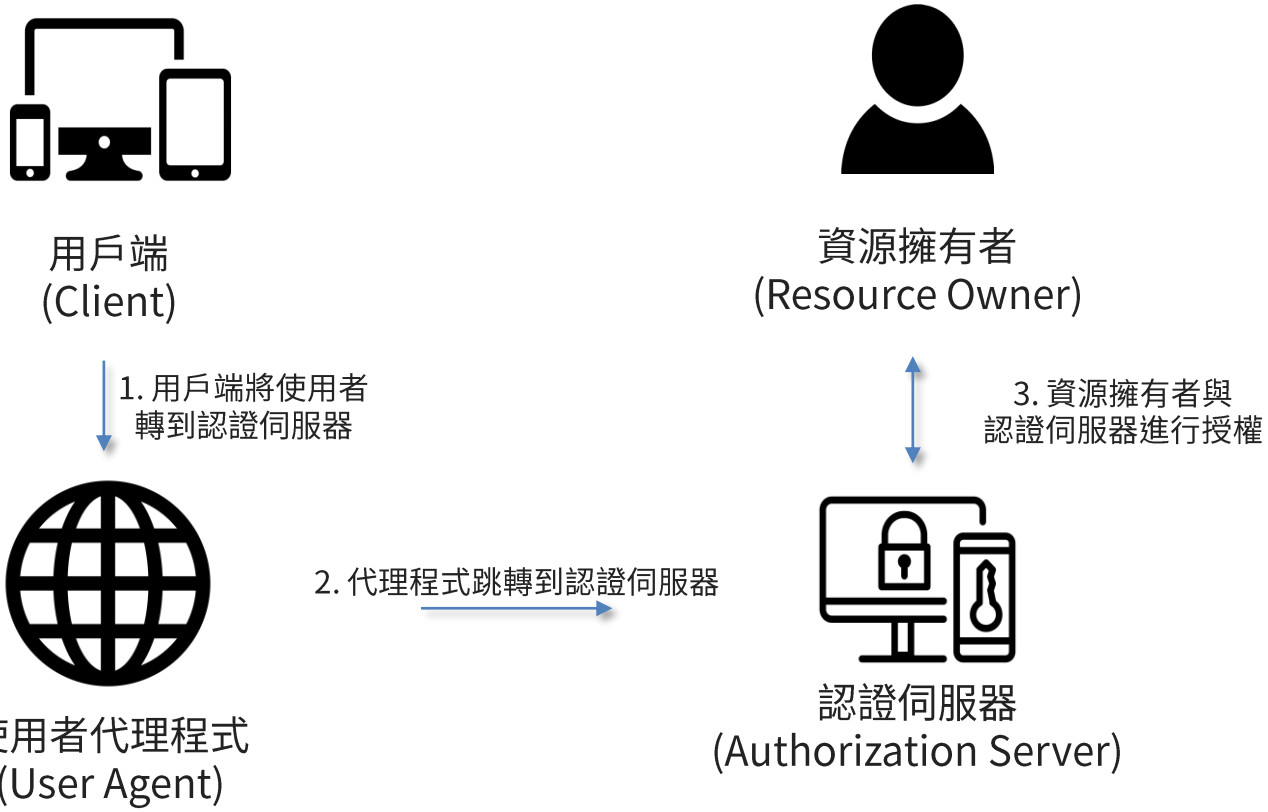
- 向 Authorization Server **取得授權碼(Authorization Code)** 再取得 **Access Token** (兩個步驟)
- 適合 **Client** 位於**伺服器**的環境(Web 應用程式)
  - 用較嚴謹的方式取得權限
- Authorization Server 可以驗證 Client 的身分，進而決定是否給予權限
- 可以核發 Refresh Token
- 需要瀏覽器進行轉址 (User-Agent Redirection)

Authorization Code Grant Type Flow



參考來源

Authorization Code Grant Type Flow



Authorization Code Grant Type Flow

用戶端
(Client)

資源擁有者
(Resource Owner)

5. 代理程式發送
授權碼給用戶端

使用者代理程式
(User Agent)

4. 認證伺服器核發授權碼

認證伺服器
(Authorization Server)

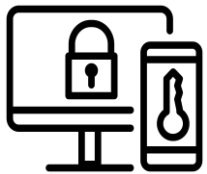
Authorization Code Grant Type Flow

用戶端
(Client)

資源擁有者
(Resource Owner)

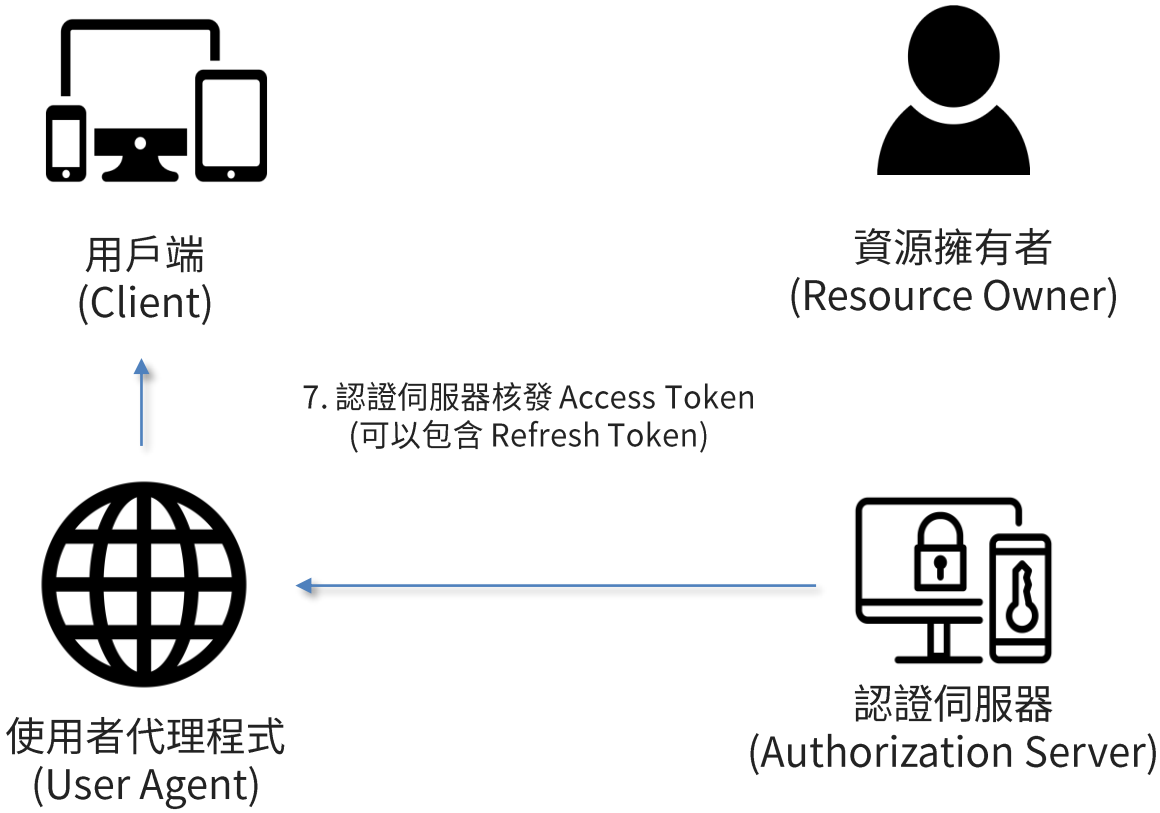
6. 用戶端引導代理程式
將授權碼帶到認證伺服器

使用者代理程式
(User Agent)



認證伺服器
(Authorization Server)

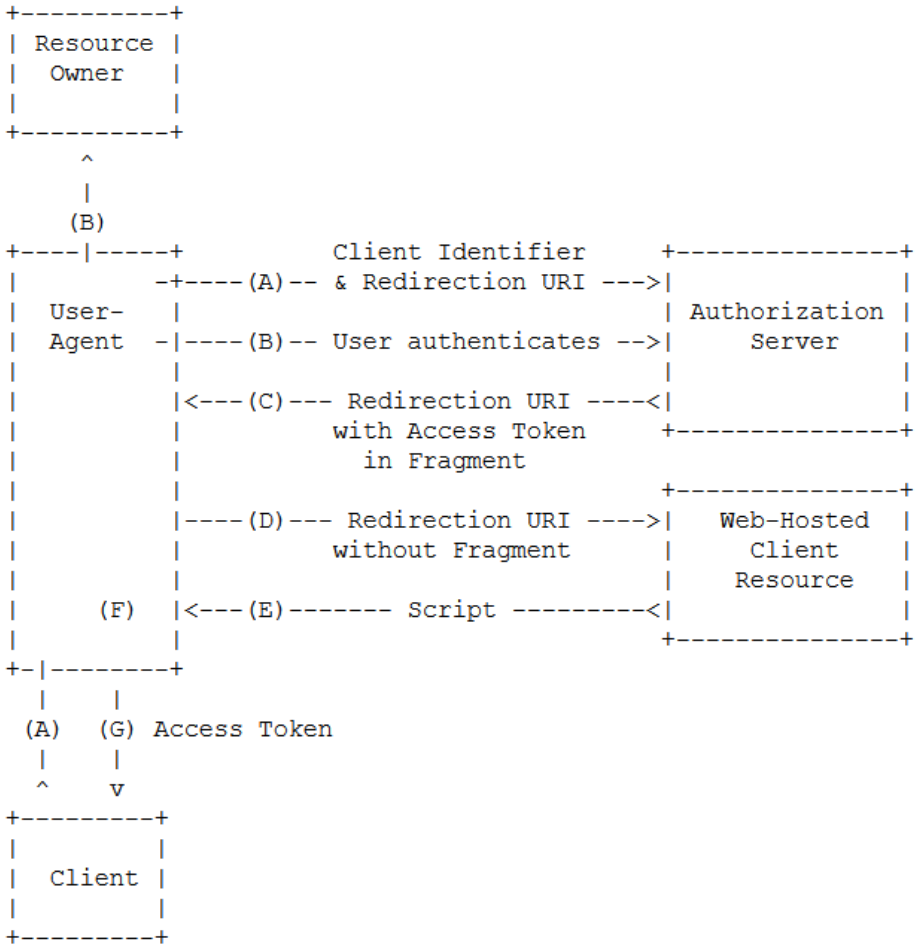
Authorization Code Grant Type Flow



Implicit Grant

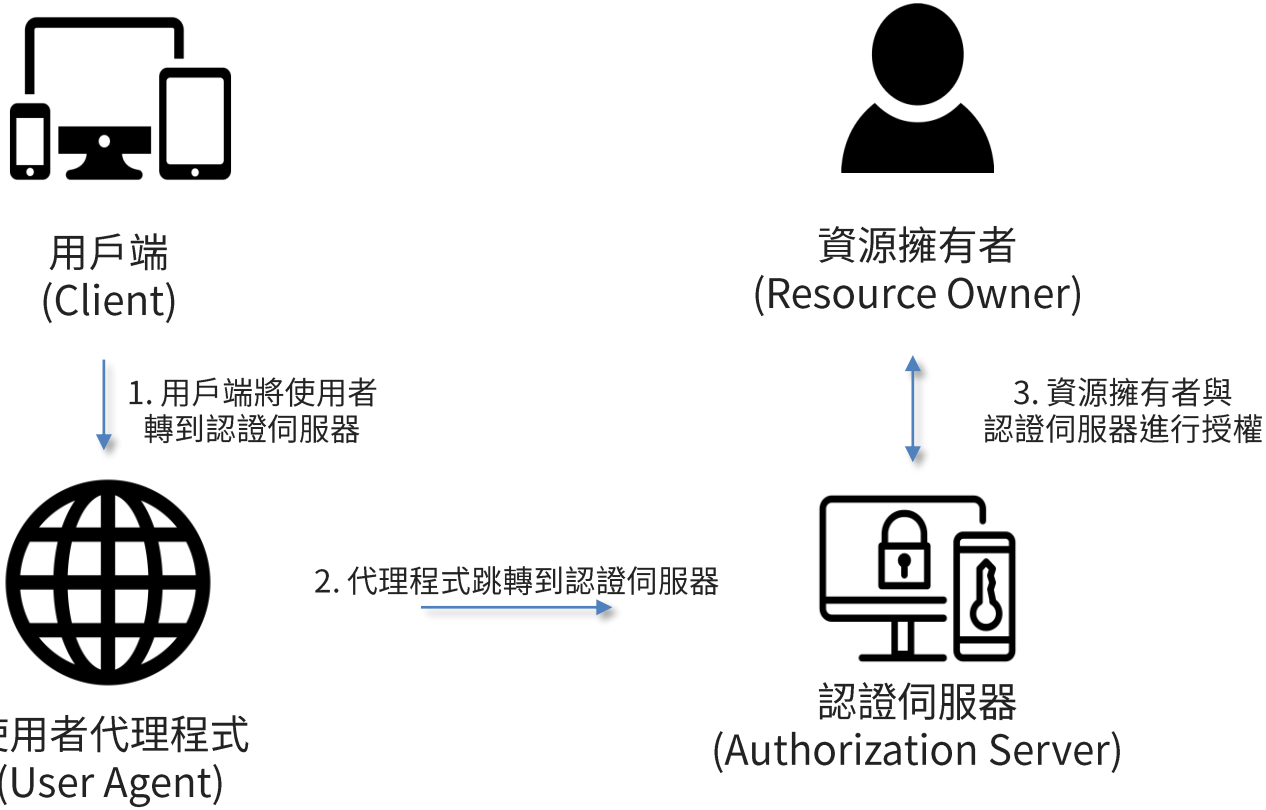
- Authorization Server 直接向 Client 核發 Access Token (一個步驟)
- 適合放在公開環境上的 Client， 如瀏覽器中的應用程式
- 禁止核發 Refresh Token， Client 不可自動幫資源擁有者重新產生新的 Access Token
- 需要瀏覽器進行轉址

Implicit Grant



參考來源

Implicit Grant



Implicit Grant

用戶端
(Client)

資源擁有者
(Resource Owner)

5. 代理程式發送
Access Token
給用戶端

使用者代理程式
(User Agent)

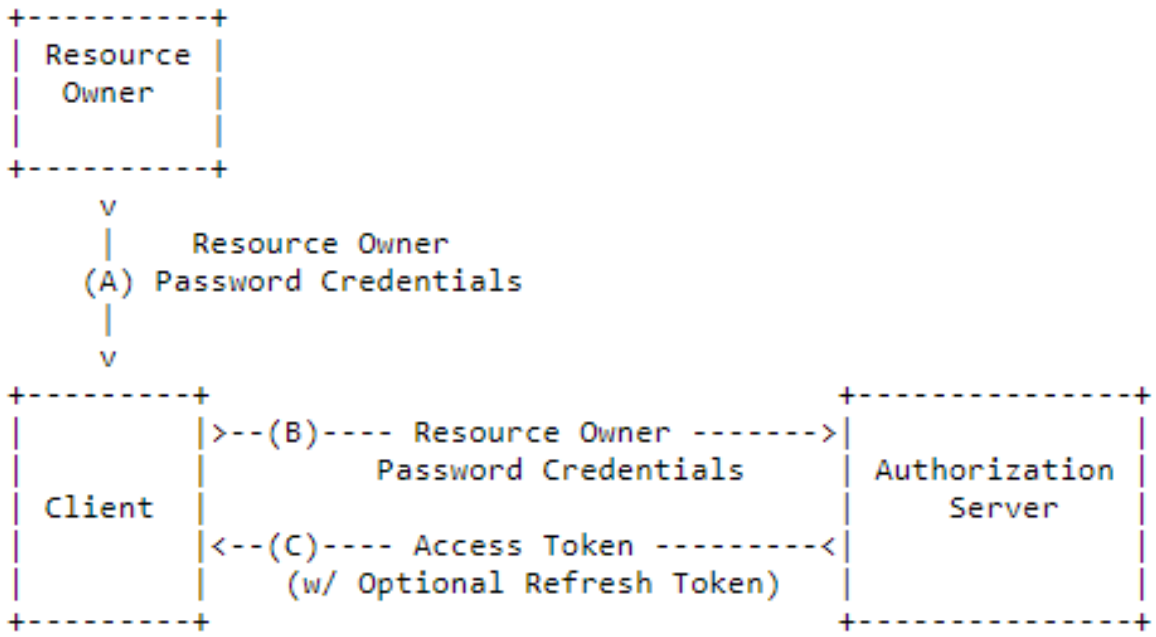
4. 認證伺服器核發 Access Token

認證伺服器
(Authorization Server)

Resource Owner Password Credentials Grant

- Resource Owner 將自己的帳號密碼交給 Client，讓 Client 直接去向 Authorization Server 取得 Access Token
- Client 取得 Access Token 後，不得儲存 Resource Owner 的帳號密碼資訊
- 適用於 Resource Owner 高度信賴的 Client 或是官方的應用程式
- 可以選擇要不要核發 Refresh Token
- 不需要使用瀏覽器轉址

Resource Owner Password Credentials Grant



參考來源

Resource Owner Password Credentials Grant

資源擁有者
(Resource Owner)

1. Resource Owner 直接提供帳號密碼給 Client

2. Client 使用拿到的帳號密碼去要求 Access Token

3. 認證伺服器直接核發 Access Token 給 Client

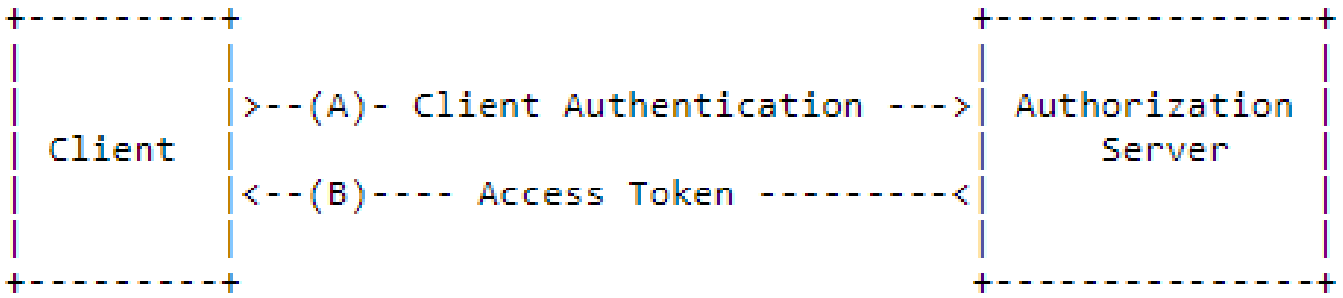
用戶端
(Client)

認證伺服器
(Authorization Server)

Client Credentials Grant

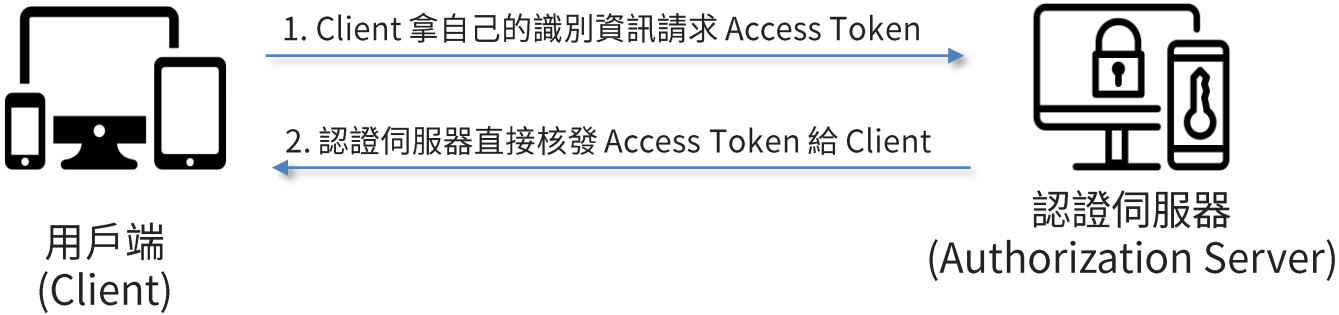
- Client 使用自己的識別資訊 (Client Id、Client Secret) 來向 Authorization Server 取得 Access Token
- 適用於在伺服器上運行的 Client 程式
- Client 本身就是 Resource Owner
- 不建議核發 Refresh Token
- 沒有瀏覽器轉址

Client Credentials Grant



參考來源

Client Credentials Grant



OpenID Connect: Introduction

簡介 OpenID Connect

關於 OpenID Connect

- OpenID Connect 是以 OAuth 2.0 為基礎發展出來的**驗證規範**

- 相較於 OAuth 2.0 的規範著重在整個**「授權 (Authorize)」**的行為，OpenID Connect 的規範則替 OAuth 2.0 的規範加入了**「身分 (Identity)」**的機制
  - 認證：我是誰
  - 授權：我允許別人使用什麼資源

- 允許 Client 在取得授權同時，得知使用者的識別資訊

OpenID Connect 的主要角色

終端使用者
(End User)

需要向網站表明身分的人

識別身分
(Identifier)

終端使用者身分的識別資訊

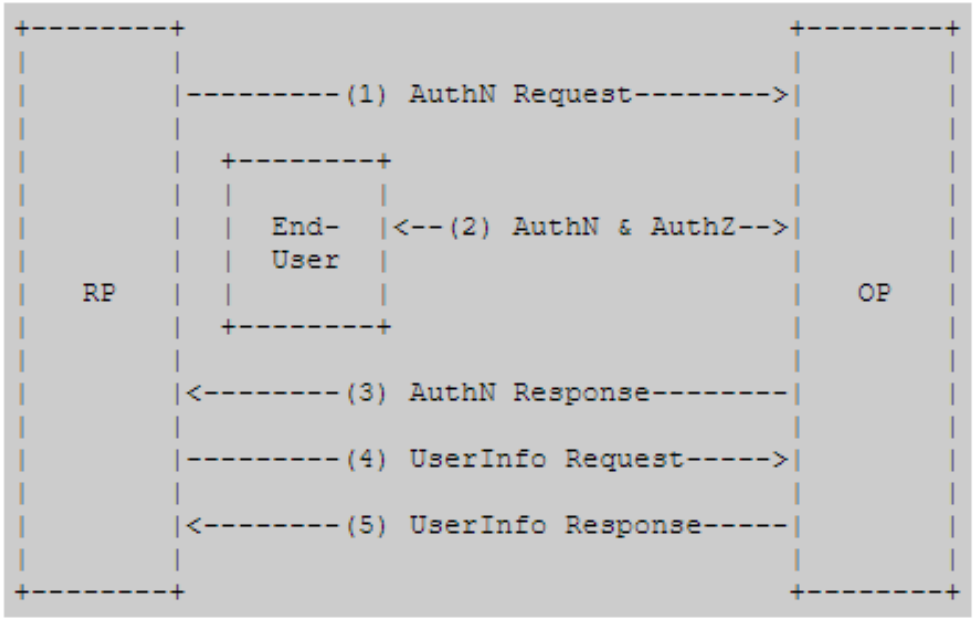
依賴方
(Relying Party)
(RP)

需要依賴終端使用者身分的網站或服務(ex：某服務允許使用 Facebook 作為第三方登入，該服務即為依賴方)

身分提供者
(OpenID Provider)
(OP)

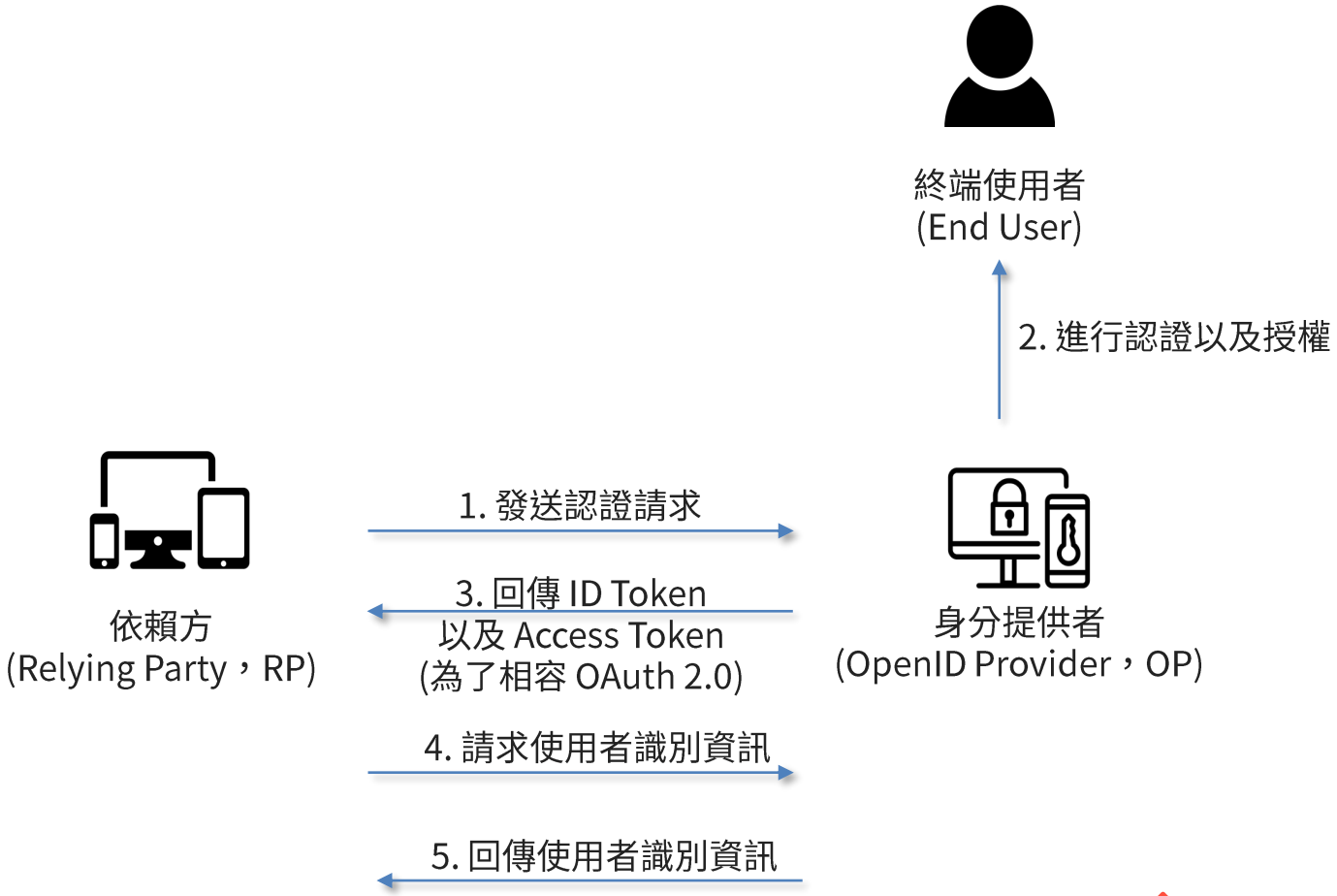
提供驗證及識別身分的服務來源(ex：Facebook 第三方登入)

OpenID Connect 的主要運作流程



參考來源

OpenID Connect 的主要運作流程



關於 ID Token

- 包含使用者基本身分相關識別資訊的字串
- 以 JWT (JSON Web Token) 格式呈現
- 在驗證使用者身分同時，將資訊回傳給依賴方(RP)
- 依賴方以此 Token 向身分提供者(OP) 要求更多的使用者識別資訊 (Claims)

關於 JSON Web Token

- JSON Web Token(JWT)是 RFC 7519 規範的一種資料交換格式
- 以此規範產生的字串可以安全地放在網址列(URL)中傳遞
- JWT 字串分成三個部分
  - **Header**：描述加密的演算法及 token 類型 (JSON 格式)
  - **Payload**：聲明實際上要傳遞的資料 (JSON 格式)
  - **Signature**：將 Header 與 Payload 經過 Base 64 編碼後加上雜湊演算法產生的簽章
- 將三份資料組合起來，即為 JWT 字串
- 只需要使用 Base64 解碼 Payload 即可知道傳遞的資訊
- 只需要知道**產生簽章**的**雜湊演算法**使用的**秘鑰**，即可確認 JWT Token 在傳遞途中是否有被竄改

關於 Claims

- 用來描述使用者(End User)的**識別資訊 (聲明資訊)**
- 在驗證過程中，使用者可以決定允許 Client 能取得那些識別資訊
- OpenID Connect 定義了數種常見的標準身分識別 (Standard Claims)
  - sub　　身分資源的提供者 (Subject) (Identifier for the End-User at the Issuer)
  - name　　使用者的完整名稱 (End-User's full name)
  - email　　使用者的電子郵件地址 (End-User's preferred e-mail address)
  - gender　使用者性別 (End-User's gender) (female, male)
  - locale　　使用者的語言地區別 (End-User's locale) (BCP47 language tag)
  - …

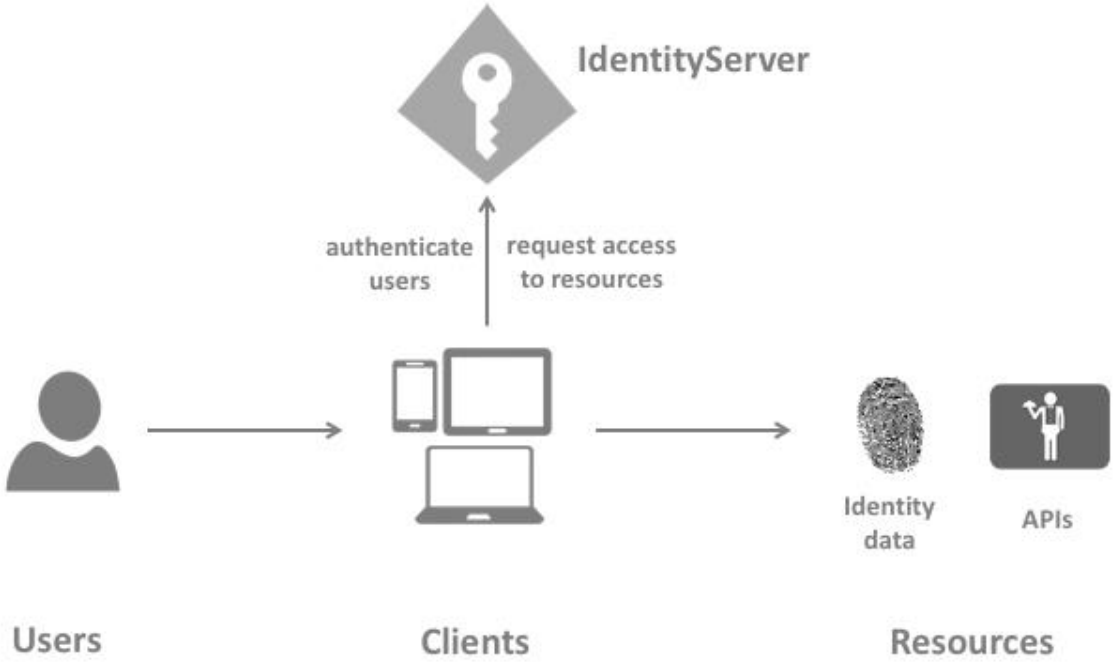
IdentityServer4: Introduction

簡介 IdentityServer4

關於 IdentityServer4

- IdentityServer4 是一套以 **ASP.NET Core 3.x** 實作出來的 **OpenID Connect** 與 **OAuth 2.0** 框架。
- 主要有以下特色
  - **Authentication as a Service** **(認證即服務)**
  - **Single Sign-on / Sign-out** **(單一簽入與簽出)**
  - **Access Control for APIs** **(對 API 進行權限控管)**
  - **Federation Gateway** **(支援邦聯式認證匝道)**
  - **Focus on Customization** **(專注在高度客製化場景)**
  - **Mature Open Source** **(相當成熟的開源專案)**
  - **Free and Commercial Support** **(免費且提供商業支援)**

IdentityServer4 重要名詞



參考來源

IdentityServer4 重要名詞

- **IdentityServer**
  - OpenID Connect 提供者，實作了 OpenID Connect 與 OAuth 2.0 的相關規範
- **User**
  - 透過 Client 存取資源的人
- **Client**
  - 在 IdentityServer 與使用者之間溝通的程式
  - 負責協調**授權**與**身分認證**等工作
  - Client 必須先在 IdentityServer 註冊，才允許請求動作

IdentityServer4 重要名詞

- **Resource**
  - 受 IdentityServer 保護的內容
  - 如使用者識別資料、API存取等等
- **Identity Token**
  - 用來代表整個驗證流程結果的識別文字
  - 包含最基本的使用者識別資訊
- **Access Token**
  - 允許 Client 存取 API 用的識別文字
  - 包含了用戶端與使用者(如果需要)的識別資訊
  - API 利用這些資訊決定是否允許存取

IdentityServer4: Quick Start

快速上手 IdentityServer4

快速建立 IdentityServer 運行環境

- 複製 IdentityServer4 專案
  - git clone https://github.com/IdentityServer/IdentityServer4
  - cd IdentityServer4
  - mkdir nuget
- 範例專案
  - `samples\Quickstarts\3_AspNetCoreAndApis\src`
    - IdentityServer　　http://localhost:5000
    - Api　　http://localhost:5001/identity
    - MvcClient　　http://localhost:5002
    - Client　　純 Console App

認識探索文件 (Discover Document)

- IdentityServer4 執行起來後，會提供一份**探索文件** (Discover Document) 包含整個 Identity Server 的相關資訊 (metadata)
  - http://localhost:5000/.well-known/openid-configuration


```
{
  "issuer": "http://localhost:5000",
  "jwks_uri": "http://localhost:5000/.well-known/openid-configuration/jwks",
  "authorization_endpoint": "http://localhost:5000/connect/authorize",
  "token_endpoint": "http://localhost:5000/connect/token",
  "userinfo_endpoint": "http://localhost:5000/connect/userinfo",
  "end_session_endpoint": "http://localhost:5000/connect/endsession",
  "check_session_iframe": "http://localhost:5000/connect/checksession",
  "revocation_endpoint": "http://localhost:5000/connect/revocation",
  "introspection_endpoint": "http://localhost:5000/connect/introspect",
  "device_authorization_endpoint": "http://localhost:5000/connect/deviceauthorization",
  "frontchannel_logout_supported": true,
```

Config.cs

- Ids
  - 設定內建允許使用的識別資訊
- Apis
  - 設定內建允許使用的 API 資源
- Clients
  - 聲明可以信任的 Relay Party (RP) Clients

Startup.cs

- ConfigureServices()

```
var builder = services.AddIdentityServer()
    .AddInMemoryIdentityResources(Config.Ids)
    .AddInMemoryApiResources(Config.Apis)
    .AddInMemoryClients(Config.Clients)
    .AddTestUsers(TestUsers.Users);
```

使用程式中寫死的資源與 Clients

- Configure()

```
app.UseIdentityServer();
```

相關連結

- RFC 6749 (OAuth 2.0) 規格文件
- OpenID Connect 規格文件
- RFC 7519 (JWT Token) 規格文件
- 支援 JWT 驗證的類別庫
- Identity Server 4 文件

聯絡資訊

The Will Will Web

網路世界的學習心得與技術分享

http://blog.miniasp.com/

Facebook

Will 保哥的技術交流中心

http://www.facebook.com/will.fans

Twitter

https://twitter.com/Will_Huang

多奇·教育訓練

THANK YOU!

Q&A