**HRM CODE TUTORIAL COLLECTION**

**Collection By:** R&D Team

Index

[IdentityServer4 (Latest) 2](#_Toc23858531)

[Features 2](#_Toc23858532)

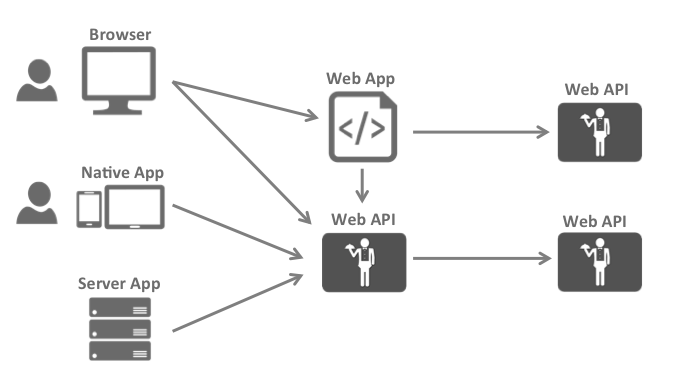
[Update Note 2](#_Toc23858533)

[How to use 3](#_Toc23858536)

# CONFIGURE SWAGGER API

## The Big Picture

Đa phần các Modern Application sẽ có kiểu



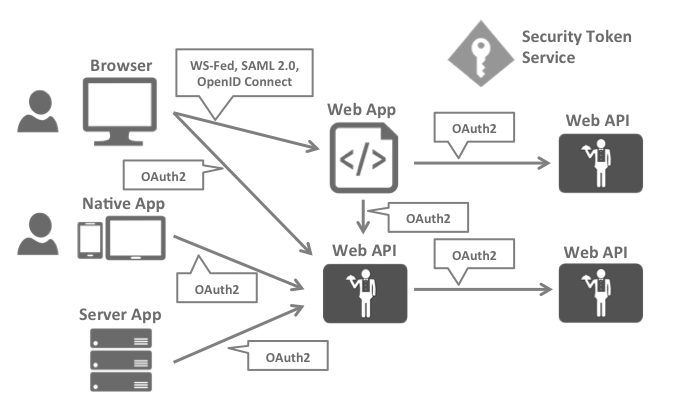
Các tương tác phổ biến nhất thường là

* Browser giao tiếp với Web Applications
* Web Applications giao tiếp với Web APIs
* Browser-Based Applications giao tiếp với Web APIs
* Native Applications giao tiếp với Web APIs
* Server-based Applications giao tiếp với Web APIs
* Web APIs giao tiếp với Web APIs

Thông thường với mỗi Layer (Front-end, Middle-tier và Back-end) sẽ phải bảo vệ Resources và triển khai xác thực và/hoặc phân quyền với cùng User

Việc Outsourcing những chức năng bảo mật cơ bản cho một Security Token Service sẽ ngăn ngừa việc Duplicate những chức năng đó qua những Applications và Endpoint.

Xây dựng lại Application hỗ trợ Security Token sẽ có mô hình như sau



Mô hình mà vấn đề Security được chia làm hai phần

### Authentication

Authentication (xác thực) cần thiết khi một Application cần biết Idenitty (định danh) của User hiện tại. Thông thường những Applications này sẽ quản lý Data thay cho người dùng, và đảm bảo việc người dùng chỉ có thể truy cập vào Data họ được cho phép. Ví dụ phổ biến nhất cho vấn đề này là Web Applications – nhưng Native và JS-based Applications hiện nay cũng cần phải được xác thực.

Các Authentication Protocols phổ biến nhất hiện nay là SAML2p, WS-Federation và OpenId Connect – SAML2p hiện nay đang phổ biến nhất và được xử dụng một cách rộng rãi

### API Access

Application có hai hướng cơ bản để có thể giao tiếp với APIs – sử dụng Application Identity, hoặc ủy quyền User’s Identity. Đôi khi cần kết hợp cả hai phương thức.

#### OAuth2

OAuth2 là một Protocol cho phép Applications yêu cầu Access Tokens từ một Security Token Service và sử dụng chúng để giao tiếp với APIs. Sự ủy quyền này làm giảm sự phức tạp của cả Client Applications và APIs bởi vì Authentication và Authorization có thể được Centralized.

### OpenID Connect và OAuth 2.0 – Better together

OpenID Connect và OAuth 2.0 khá giống nhau – trên thực tế OpenID Connect là một Extension dựa trên OAuth 2.0. Hai vấn đề bảo mật cơ bản cần quan tâm ở đây là Authentication và API Access được kết hợp thành một Protocol.

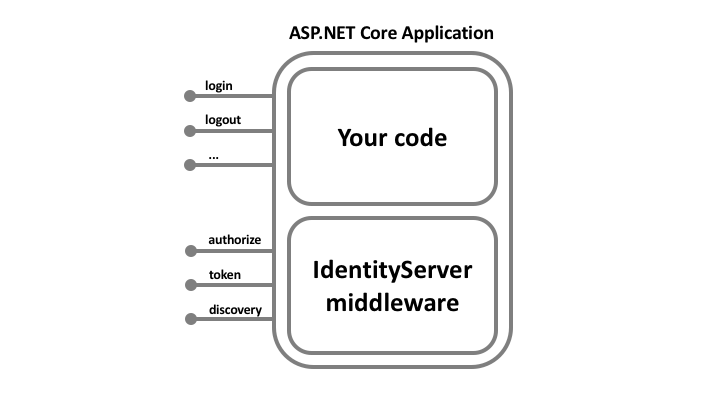
Hiện tại việc kết hợp OpenID Connect và OAuth 2.0 được coi là phương pháp tốt nhất để bảo vệ Applications trong một tương lai có thể thấy trước.

IdentityServer4 là một sự triển khai của hai Protocols này và được tối ưu cao để xử lý các vấn đề bảo mật cơ bản hiện nay trên Mobile, Native và Web Applications.

### How IdentityServer4 can help

IdentityServer là một Middleware tuân theo OpenID Connect và OAuth 2.0 bổ sung Endpoints vào một ASP.NET Core Application.

Thông thường, chúng ta build (hoặc sử dụng lại) một Application chứa các Page Login và Logout (đôi khi bao gồm cả Consent – Consent thể hiện một cách rõ ràng User’s Permission để cho phép một Application truy cập vào Resouces được bảo vệ bởi một Scope cụ thể), và IdentityServer Middleware thêm các Protocol cần thiết để Client Applications có thể giao tiếp với chúng sử dụng Standard Protocols



## Features

## Update Note

## How to use

### Protecting an API using Client Credentials

#### Setup ASP.NET Core Application

##### Identity Server

##### API

##### Client

#### Defining an API Scope

API là một Resouce trong hệ thống của chúng ta và cần được bảo vệ….

public static IEnumerable<ApiScope> ApiScopes =>

new List<ApiScope>

{

new ApiScope("api1", "My API")

};

#### Defining the Client

Define một Client Application sẽ truy cập vào API mới tạo

public static IEnumerable<Client> Clients =>

new List<Client>

{

new Client

{

ClientId = "client",

// no interactive user, use the clientid/secret for authentication

AllowedGrantTypes = GrantTypes.ClientCredentials,

// secret for authentication

ClientSecrets =

{

new Secret("secret".Sha256())

},

// scopes that client has access to

AllowedScopes = { "api1" }

}

};

#### Configure IdentityServer

Loading Resouce và Client Definitions diễn ra tại Startup.cs (IdentityServer)

public void ConfigureServices(IServiceCollection services)

{

var builder = services.AddIdentityServer()

.AddInMemoryApiScopes(Config.ApiScopes)

.AddInMemoryClients(Config.Clients);

builder.AddDeveloperSigningCredential();

}

### Interactive Applications with ASP.NET Core

### ASP.NET Core and API Access