Neel Bhatt

☰ **Thực đơn**

Từng bước thiết lập cho máy chủ Auth và máy khách: IdentityServer4 với .Net Core Part II

Neel .Net Core , .Net Core 2.1 , .Net Core , ASP .Net Core , ASP .Net Core 2.0 , Asp .Net Core 2.1 , asp.netcore2.0 , Xác thực trong .Net Core 2.0 , Core 2.0 , IdentityServer4 , Thiết lập IdentityServer4 , IdentityServer4 với .Net Core , Visual Studio 2017 Ngày 4 tháng 3 năm 2018Ngày 9 tháng 3 năm 2018 5 phút

Lưu ý - Bạn có thể tìm thấy mã nguồn của ứng dụng mẫu của tôi tại đây (https://github.com/NeelBhatt/TeamX.Security.AuthenticationServer-IdentityServerv4-) . (Lưu ý rằng mã có thể chứa mã phụ, tập trung vào máy chủ và máy khách Auth ngay bây giờ)

Bạn có thể tìm thấy tất cả . Net cốt lõi bài viết ở đây (https://neelbhatt.com/category/net-core/) .

Trong bài trước của tôi trên **IdentityServer4** , tôi đã giải thích những **điều cơ bản về IdentityServer4** mà bạn có thể tìm thấy ở đây (https://neelbhatt.com/2018/02/24/identityserver4-in-simple-words-identityserver4-with-net-core-part-i/) .

Trong bài này, chúng tôi sẽ thiết lập một **máy chủ Auth** mẫu cùng với **một máy khách sẽ yêu cầu mã thông báo** .

Chúng ta hãy bắt đầu.

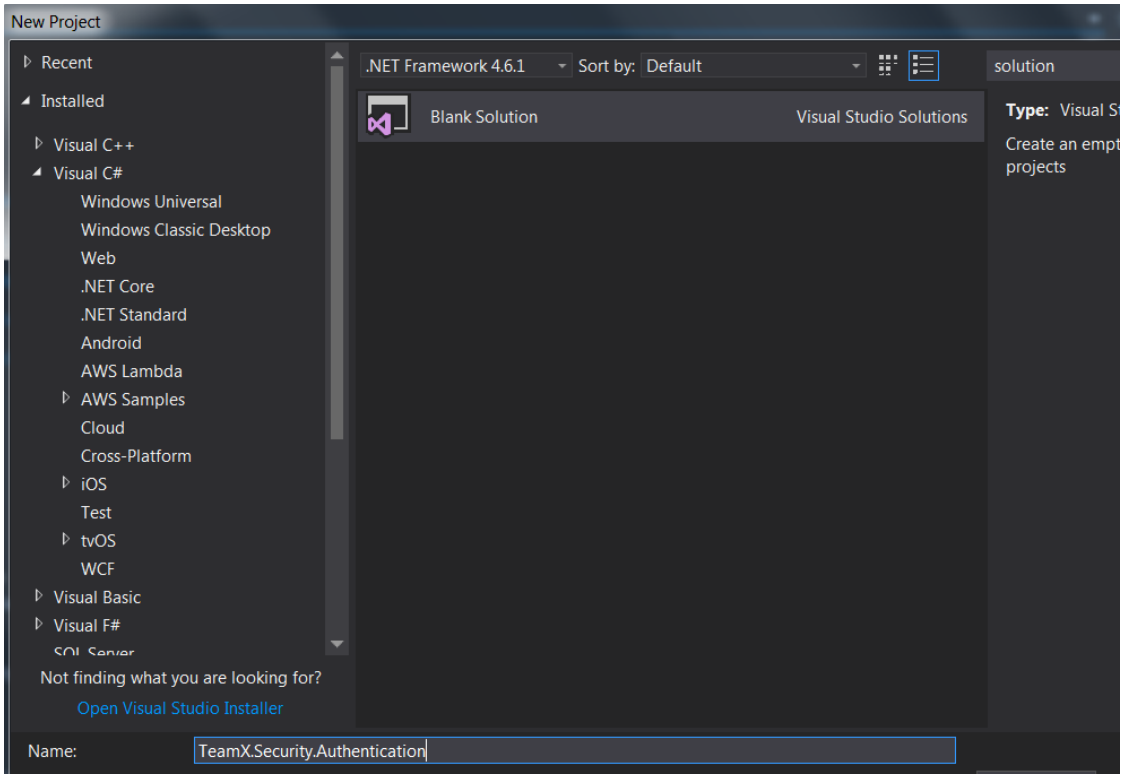
Thiết lập máy chủ xác thực

điều kiện tiên quyết:

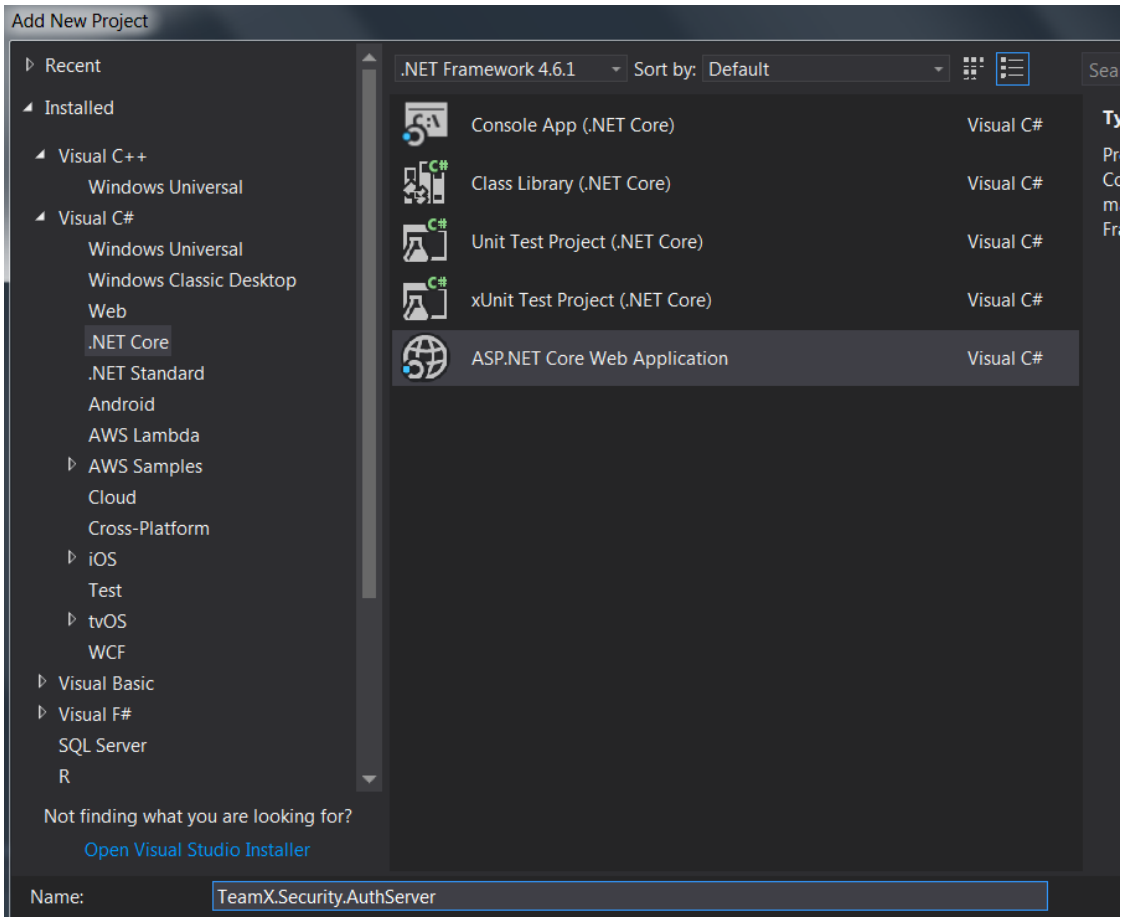
- **Visual studio 2017** phiên bản cộng đồng, tải xuống tại đây (https://www.visualstudio.com/downloads/)
- **.Net Core 2.0 SDK** từ đây (https://www.microsoft.com/net/download/windows) (Tôi đã viết một bài đăng để cài đặt SDK tại đây (https://neelbhatt40.wordpress.com/2017/09/11/net-core-2-0-installation/)) (Bạn có thể tải xuống phiên bản mới nhất, hiện tại tôi đang sử dụng phiên bản 2.0 cho việc này)

Tạo ứng dụng Web bằng mẫu NetNet 2.0 trong VS 2017

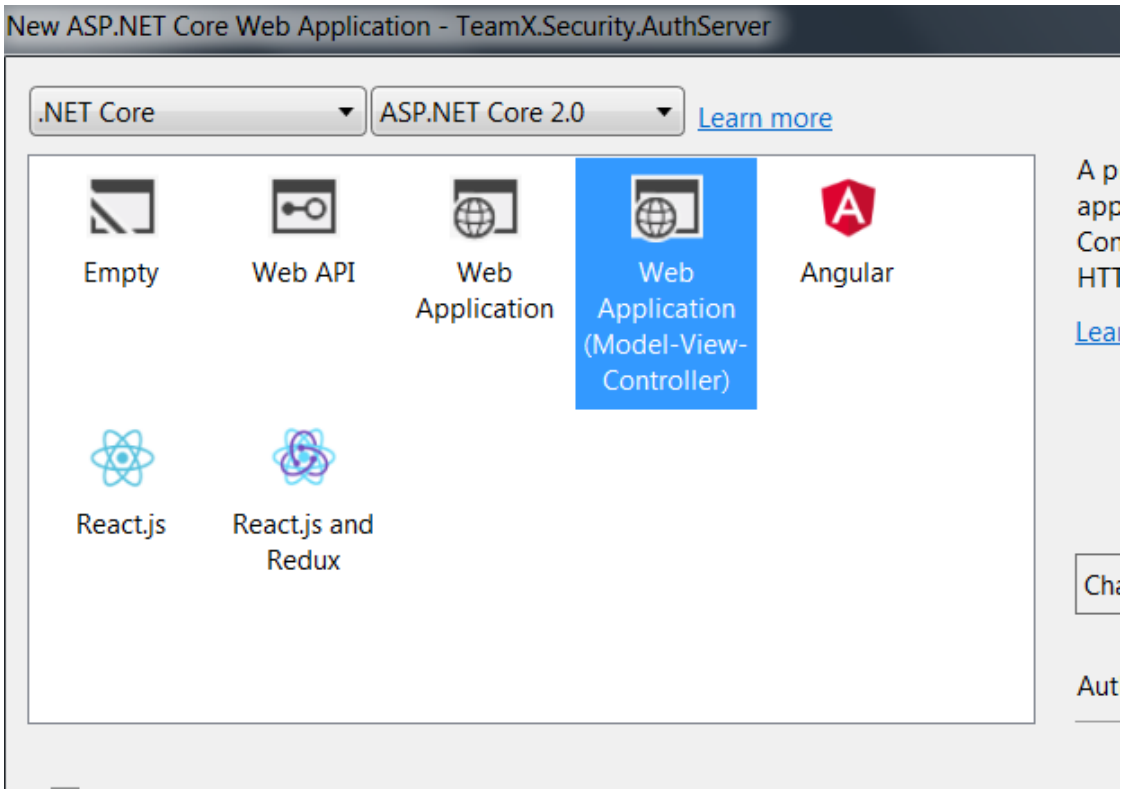
Trước tiên, hãy thêm **giải pháp trống** và trong giải pháp đó, chúng tôi sẽ thêm **các dự án khác nhau** :



Khi giải pháp được thêm vào, nhấp chuột phải vào giải pháp -> Tạo **dự án mới** -> Chọn **ứng dụng Web lõi** :



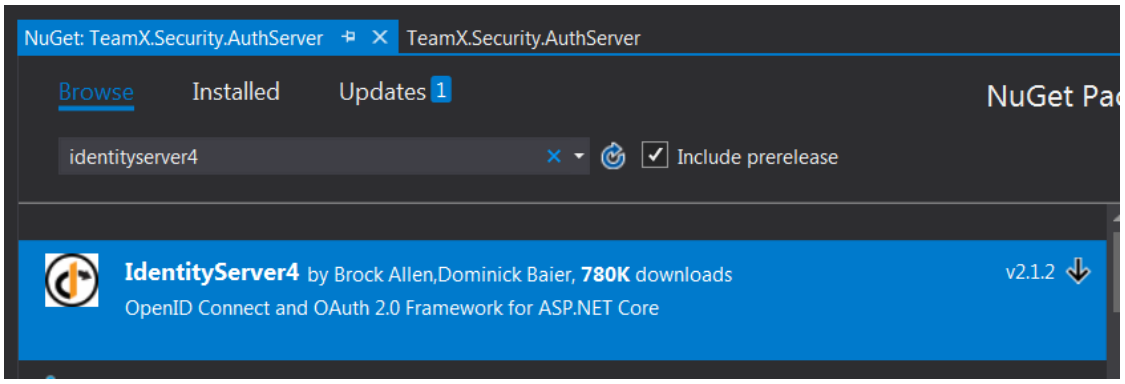
Nhấp vào Ok và trong cửa sổ tiếp theo, chọn Ứng dụng web (MVC) như hình dưới đây:



Visual Studio sẽ tạo ra một ứng dụng có cấu trúc tốt cho bạn.

Bước tiếp theo là cài đặt **IdentityServer4** trong ứng dụng của bạn.

Mở Nuget và tìm kiếm bằng **IdentityServer4** -> nhấp vào **cài đặt** :



Thay đổi trong lớp Startup.cs

Bước tiếp theo là thực hiện thay đổi trong **cấu hình** và **ConfigureService** phương pháp **Startup.cs** lớp.

Định cấu hình thay đổi phương thức

Thêm **app.UseIdentityServer ();** trong phương thức **Cấu hình** sẽ thêm **phần mềm trung gian** vào **đường dẫn HTTP** :

```
1  public void Configure(IApplicationBuilder app, IHostingEnvironment env)
2  {
3
4  app.UseIdentityServer();
5
6  //// Other code
7
số 8  }
```

UseIdentityServer cho phép **IdentityServer** bắt đầu chặn các tuyến đường và xử lý các yêu cầu.

Thay đổi phương thức ConfigureService

Bây giờ, thêm mã cần thiết trong phương thức **ConfigureService để cấu hình và thêm các dịch vụ cần thiết vào hệ thống DI** :.

Bây giờ, chúng tôi sẽ sử dụng **trong bộ nhớ** người dùng, cũng như khách hàng và phạm vi:

```
public void ConfigureServices(IServiceCollection services)
{
services.AddMvc();

services.AddIdentityServer()
.AddDeveloperSigningCredential()
.AddInMemoryApiResources(Config.GetApiResources())
.AddInMemoryClients(Config.GetClients());
}
số
```

Ở đây chúng ta cần thêm các dịch vụ bên dưới cùng với AddIdentityServer:

- **AddInMemoryClients** bao gồm những máy khách nào được phép sử dụng máy chủ Auth này
- **AddInMemoryApiResources** bao gồm các API nào được phép sử dụng máy chủ Auth này
- **AddDeveloperSigningCredential** để ký mã thông báo

Xin lưu ý rằng có **những dịch vụ khác** cũng có thể được bổ sung ở đây nhưng đối với dịch vụ mẫu trên đây là đủ.

Thêm lớp cấu hình

Như bạn có thể thấy ở trên, chúng ta sẽ yêu cầu lớp **Config** sẽ được sử dụng để viết tất cả các phương thức được yêu cầu ở trên.

Hãy để chúng tôi thêm lớp mới **Config.cs** và thêm mã bên dưới vào lớp:

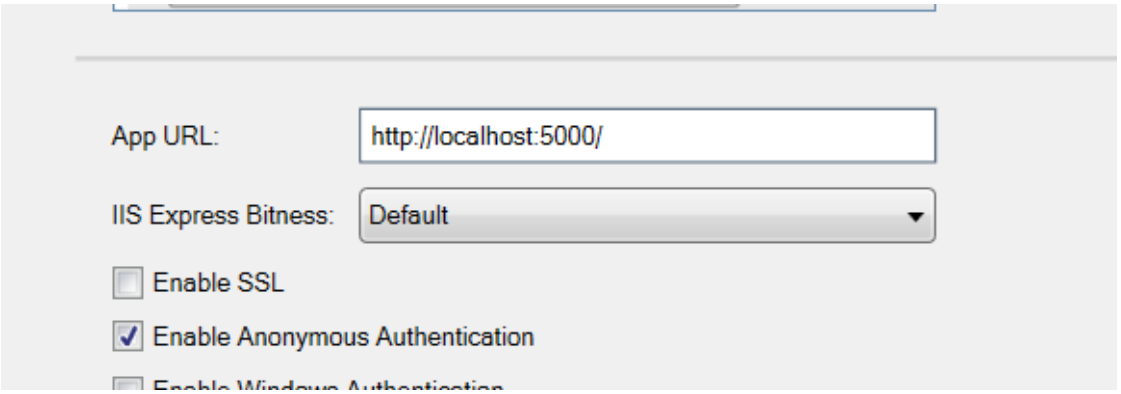
```
using IdentityServer4.Models;
using System.Collections.Generic;

namespace TeamX.Security.AuthServer
{
public class Config
{
// clients that are allowed to access resources from the Auth server
public static IEnumerable<Client> GetClients()
{
// client credentials, list of clients
return new List<Client>
{
new Client
{
ClientId = "client",
AllowedGrantTypes = GrantTypes.ClientCredentials,

// Client secrets
ClientSecrets =
{
new Secret("secret".Sha256())
},
AllowedScopes = { "api1" }
},
};
}

// API that are allowed to access the Auth server
public static IEnumerable<ApiResource> GetApiResources()
{
return new List<ApiResource>
{
new ApiResource("api1", "My API")
};
}
}
}
```

số 8

Ở đây chúng tôi có:

- Chúng tôi đã thêm **một ứng dụng khách có ID "client"** , vì vậy bất cứ khi nào một máy khách có Id "client" gọi đến máy chủ Auth, máy chủ của chúng tôi sẽ trả lời yêu cầu bằng mã thông báo **sha256**
- Hiện tại, chúng tôi **chỉ** thêm **một ứng dụng khách đơn giản** , sau đó chúng tôi sẽ bổ sung thêm các ứng dụng khách khác
- Phương thức **GetClients** nhận danh sách các máy khách được phép sử dụng máy chủ Auth này
- Phương thức **GetApiResources** nhận danh sách các API được phép sử dụng máy chủ Auth này

Chúng ta gần như đã hoàn thành với **AuthServer** , một điều cuối cùng chúng ta cần thay đổi là **số cổng** . Thay đổi nó thành **5000** như hình dưới đây:



Tôi sẽ khuyến khích sử dụng **HTTPS** nhưng đối với dự án mẫu này, chúng tôi đang sử dụng **HTTP** .

Đó là nó. Bạn vừa mới tạo ra máy chủ Auth mới sử dụng IdentityServer4.

Chạy ứng dụng. Bây giờ máy chủ đã sẵn sàng nhận các yêu cầu từ máy khách với id **"client"**

Bạn có thể xem cấu hình OpenId bằng cách chạy dưới URL:

http: // localhost: 5000 / .well-known / openid-configuration (http://localhost:5000/.well-known/openid-configuration)

Nó sẽ trả về một cái gì đó như dưới đây:

Đây được gọi là **Tài liệu Khám phá Kết nối OpenID** .

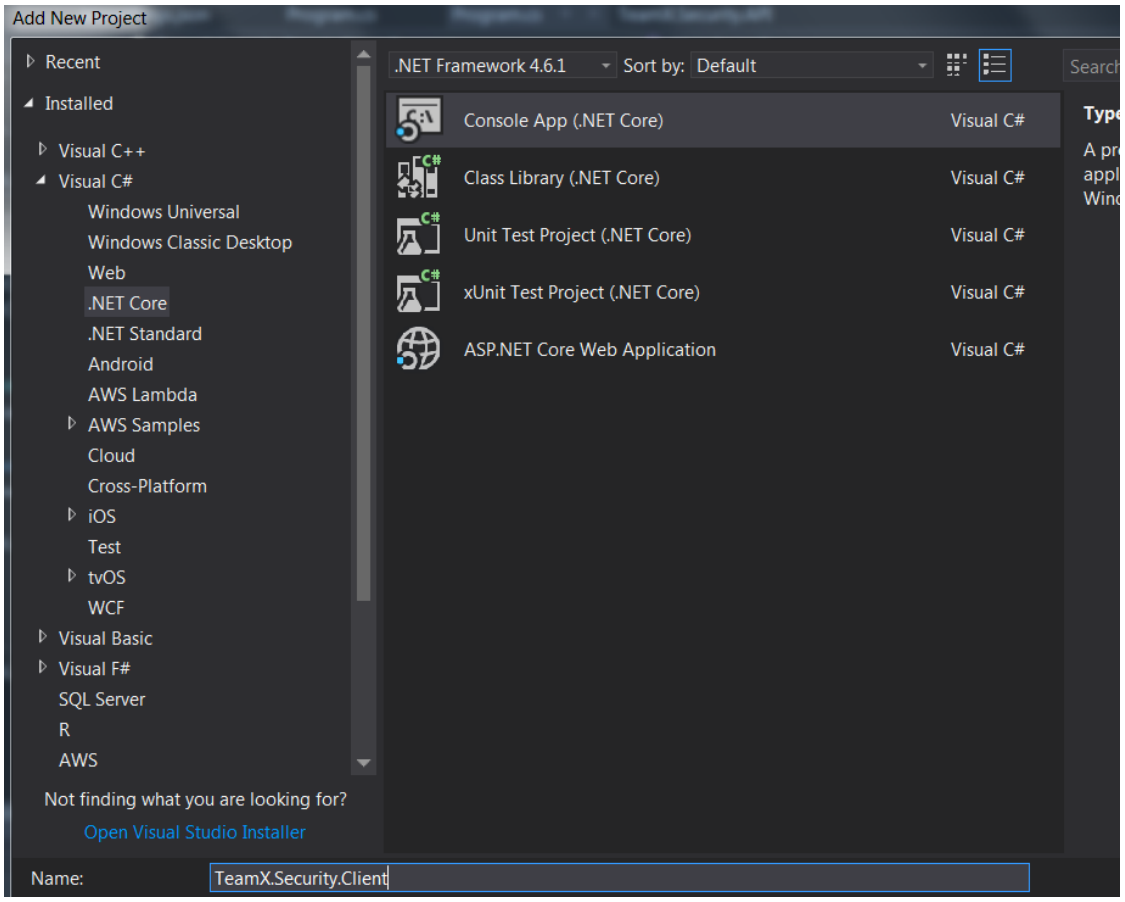
Và chi tiết này chứa:

- Vị trí của các **điểm cuối** khác nhau (Ví dụ: điểm cuối mã thông báo và điểm cuối phiên kết thúc)
- Các loại trợ cấp mà **nhà cung cấp hỗ trợ**
- **Phạm vi** nó có thể cung cấp và một số chi tiết

Chúng ta hãy tạo một **Client** mẫu với Id "client" được cho phép sử dụng máy chủ Auth của chúng ta.

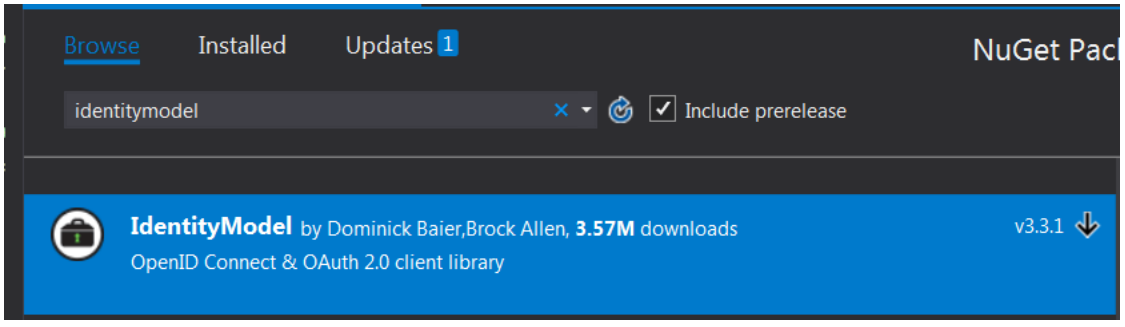
Khách hàng

Nhấp chuột phải vào giải pháp -> Tạo dự án mới -> Chọn **ứng dụng Console (.Net Core)** :



Bước tiếp theo là cài đặt **IdentityModel** trong ứng dụng khách của bạn.

Mở NuGet và tìm kiếm bằng **IdentityModel** -> nhấp vào **cài đặt** :



Thêm mã bên dưới vào lớp Program.cs:

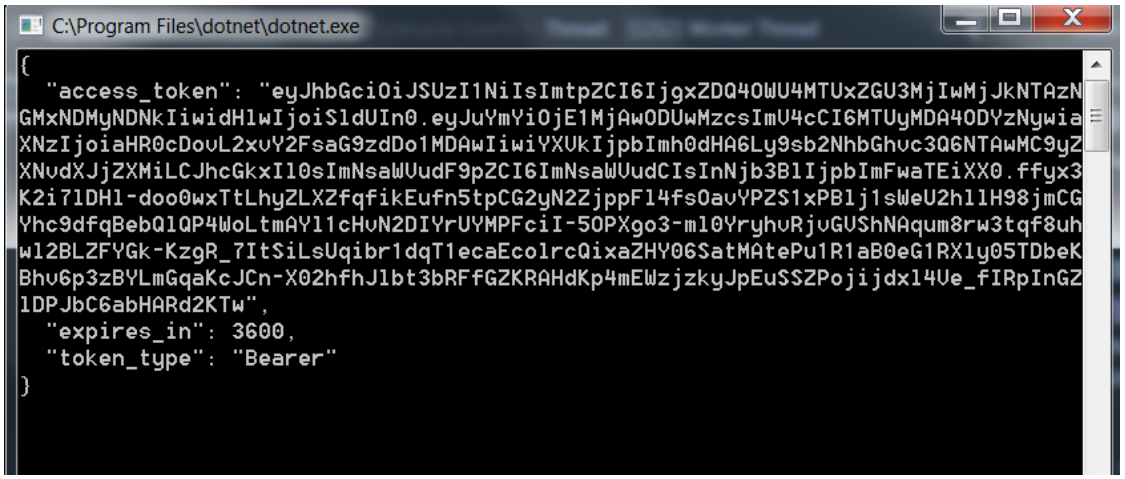
```
sing System.Threading.Tasks;
using IdentityModel.Client;
using System.Net.Http;
using Newtonsoft.Json.Linq;

namespace TeamX.Security.Client
{
public class Program
{
public static void Main(string[] args) =&gt; MainAsync().GetAwaiter().GetRes

private static async Task MainAsync()
{
// discover endpoints from the metadata by calling Auth server hosted on 500
var discoveryClient = await DiscoveryClient.GetAsync("http://localhost:5000
if (discoveryClient.IsError)
{
Console.WriteLine(discoveryClient.Error);
return;
}

// request the token from the Auth server
var tokenClient = new TokenClient(discoveryClient.TokenEndpoint, "client", '
var response = await tokenClient.RequestClientCredentialsAsync("api1");

if (response.IsError)
{
Console.WriteLine(response.Error);
return;
}

Console.WriteLine(response.Json);

}

}

}
```

Ở đây chúng tôi là:

- **Khám phá các điểm cuối** bằng cách gọi máy chủ auth được lưu trữ trên cổng **5000**
- Khi chúng tôi có **điểm cuối** , chúng tôi đang tạo **Mã thông báo khách hàng** có chứa điểm cuối, **ClinetId** (trong trường hợp của chúng tôi là "khách hàng") và **bí mật của khách hàng**
- Khi **TokenClient** được tạo, chúng tôi sẽ yêu cầu mã thông báo từ **máy chủ auth**
- Khi máy chủ gửi mã thông báo, chúng tôi sẽ chuyển đổi thành **json** và sẽ hiển thị trong **bảng điều khiển**

Đảm bảo máy chủ của bạn đã chạy trên cổng **5000** và sau đó chạy ứng dụng khách.

Khi bạn chạy ứng dụng khách, **máy chủ Auth sẽ gửi** lại **mã thông báo** cho máy khách như bạn có thể thấy bên dưới trên màn hình Console:



Đây không phải là **mã thông báo truy cập** mà bạn có thể sử dụng cho bất kỳ thứ gì bạn muốn.

Mã thông báo được trả lại dưới dạng JWT. Nếu chúng tôi muốn giải mã mã thông báo này, hãy truy cập https://jwt.io/ (https://jwt.io/) và dán mã thông báo truy cập tại đó. Nó sẽ trả về một cái gì đó như dưới đây:

```
HEADER:ALGORITHM &amp;amp; TOKEN TYPE

{
"alg": "RS256",
"kid": "81d489e8151de722022d5034c143243d",
"typ": "JWT"
}
```

PAYLOAD: DATA

```
{
"nbf": 1520141076,
"exp": 1520144676,
"iss": "http://localhost:5000 (http://localhost:5000)",
"aud": [
"http://localhost:5000/resources (http://localhost:5000/resources)",
"api1"
],
"client_id": "client",
"scope": [
"api1"
]
}
```

Trong bài viết tiếp theo của **IdentityServer4** , chúng ta sẽ xem cách bảo mật API bằng cách sử dụng mã thông báo mà chúng ta đã tạo bằng cách sử dụng Auth Server.

Lưu ý - Bạn có thể tìm thấy mã nguồn của ứng dụng mẫu của tôi tại đây (https://github.com/NeelBhatt/TeamX.Security.AuthenticationServer-IdentityServerv4-) . (Lưu ý rằng mã có thể chứa mã phụ, tập trung vào máy chủ và máy khách Auth ngay bây giờ)

Hãy theo dõi.

Nhà xuất bản Neel

Xem những bài viết của Neel

4 suy nghĩ về " Thiết lập từng bước cho máy chủ Auth và máy khách: IdentityServer4 với .Net Core Part II "

Pingback: IdentityServer4 trong các từ đơn giản: IdentityServer4 với .Net Core Phần I - Neel Bhatt
Pingback: Dew Drop - Ngày 5 tháng 3 năm 2018 (# 2677) - Morning Dew
Pingback: Bảo mật API Web với IdentityServer4: IdentityServer4 với .Net Core Phần III - Neel Bhatt
Pingback: Từng bước thiết lập cho máy chủ Auth và máy khách: IdentityServer4 với .Net Core Phần II - Cách mã .NET