Azure AD Authentication with ASP.NET Core

Contents

Calling a web API in an ASP.NET Core web application using Azure AD

URL: https://azure.microsoft.com/en-in/resources/samples/active-directory-dotnet-webapp-webapi-openidconnect-aspnetcore/

Folder: D:\Users\AjayS\Tryouts\dotNET_Core\active-directory-dotnet-webapp-webapi-openidconnect-aspnetcore

Repo:

Step 1: Clone or download this repository

From your shell or command line:

```
git clone https://github.com/Azure-Samples/active-directory-dotnet-webapp-webapi-openidconnect-aspnetcore.git
```

Step 2: Register the sample application with your Azure Active Directory tenant

If you want to use this automation:

1. On Windows, run PowerShell and navigate to the root of the cloned directory
2. In PowerShell run:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process -Force
```

3. Run the script to create your Azure AD application and configure the code of the sample application accordingly.
4. In PowerShell run:

```
.\AppCreationScripts\Configure.ps1
```

Other ways of running the scripts are described in App Creation Scripts

1. Open the Visual Studio solution and click start to run the code.

If you don't want to use this automation, follow the steps below.

Choose the Azure AD tenant where you want to create your applications

As a first step you'll need to:

1. Sign in to the Azure portal using either a work or school account or a personal Microsoft account.
2. If your account is present in more than one Azure AD tenant, select your profile at the top right corner in the menu on top of the page, and then **switch directory**. Change your portal session to the desired Azure AD tenant.

Register the service app (TodoListService-aspnetcore)

1. Navigate to the Microsoft identity platform for developers App registrations (https://go.microsoft.com/fwlink/?linkid=2083908&clcid=0x409) page.
2. Select **New registration**.
3. When the **Register an application page** appears, enter your application's registration information:

- In the **Name** section, enter a meaningful application name that will be displayed to users of the app, for example `TodoListService-aspnetcore`.
- Change **Supported account types** to **Accounts in any organizational directory**.
- In the Redirect URI (optional) section, select **Web** in the combo-box and enter the following redirect URIs: `https://localhost:44351/` (*This uri may be different for you. Check the relevant uri for your application and use that*).

4. Select **Register** to create the application.
5. On the app **Overview** page, find the **Application (client) ID** value and record it for later. You'll need it to configure the Visual Studio configuration file for this project.
6. Select the **API permissions** section
   - Click the **Add a permission** button and then,
   - Ensure that the **Microsoft APIs** tab is selected
   - In the *Commonly used Microsoft APIs* section, click on **Microsoft Graph**
   - In the **Delegated permissions** section, ensure that the right permissions are checked: **User.Read**. Use the search box if necessary.
   - Select the **Add permissions** button
7. Select the **Expose an API** section, and:
   - Select **Add a scope**
   - Accept the proposed Application ID URI (api://{clientId}) by selecting **Save and Continue**
   - Enter the following parameters
   - for **Scope name** use `user_impersonation`
   - Keep `Admins and users` for **Who can consent**
   - in **Admin consent display name** type `Access TodoListService-aspnetcore as a user`
   - in **Admin consent description** type `Accesses the TodoListService-aspnetcore Web API as a user`
   - in **User consent display name** type `Access TodoListService-aspnetcore as a user`
   - in **User consent description** type `Accesses the TodoListService-aspnetcore Web API as a user`
   - Keep **State** as `Enabled`
   - Select **Add scope**

Register the client app (TodoListWebApp-aspnetcore)

1. Navigate to the Microsoft identity platform for developers App registrations (https://go.microsoft.com/fwlink/?linkid=2083908&clcid=0x409) page.
2. Select **New registration**.
3. When the **Register an application page** appears, enter your application's registration information:
   - In the **Name** section, enter a meaningful application name that will be displayed to users of the app, for example `TodoListWebApp-aspnetcore`.
   - Change **Supported account types** to **Accounts in any organizational directory**. > Note that there are more than one redirect URIs. You'll need to add them from the **Authentication** tab later after the app has been created successfully.

4. Select **Register** to create the application.
5. On the app **Overview** page, find the **Application (client) ID** value and record it for later. You'll need it to configure the Visual Studio configuration file for this project.
6. From the app's Overview page, select the **Authentication** section.
   - In the Redirect URIs section, select **Web** in the combo-box and enter the following redirect URIs (*Please check the uri for your application and use that instead of localhost:44377*).
     - `https://localhost:44377/`
     - `https://localhost:44377/signin-oidc`
   - In the **Advanced settings** section set **Logout URL** to `https://localhost:44377/Account/EndSession`
   - In the **Advanced settings** | **Implicit grant** section, check **ID tokens** and **Access Tokens** as this sample requires the Implicit grant flow to be enabled to sign-in the user, and call an API.
7. Select **Save**.
8. From the **Certificates & secrets** page, in the **Client secrets** section, choose **New client secret**:
   - Type a key description (of instance app secret),
   - Select a key duration of either **In 1 year**, **In 2 years**, or **Never Expires**.
   - When you press the **Add** button, the key value will be displayed, copy, and save the value in a safe location.
   - You'll need this key later to configure the project in Visual Studio. This key value will not be displayed again, nor retrievable by any other means, so record it as soon as it is visible from the Azure portal.
9. Select the **API permissions** section
   - Click the **Add a permission** button and then:
   - Ensure that the **My APIs** tab is selected
   - In the list of APIs, select the API TodoListService-aspnetcore.
   - In the **Delegated permissions** section, ensure that the right permissions are checked: **user_impersonation**.
   - Select the **Add permissions** button.

Configure authorized client applications for service (TodoListService-aspnetcore)

For the middle tier web API (`TodoListService-aspnetcore`) to be able to call the downstream web APIs, the user must grant the middle tier permission to do so in the form of consent. However, since the middle tier has no interactive UI of its own, you need to explicitly bind the client app registration in Azure AD, with the registration for the web API. This binding merges the consent required by both the client and middle tier into a single dialog, which will be presented to the user by the client. You can do so by adding the "Client ID" of the client app, to the manifest of the web API in the `knownClientApplications` property. Here's how:

1. In the Azure portal, navigate to your `TodoListService-aspnetcore` app registration, and in the **Expose an API** section, click on **Add a client application**. Input the client ID of the client

application (`TodoListWebApp-aspnetcore`) and check **user_impersonation** for authorized scopes.

2. Click **Add application**

Step 3: Configure the sample to use your Azure AD tenant

In the steps below, "ClientID" is the same as "Application ID" or "AppId".

Open the solution in Visual Studio to configure the projects

Configure the service project

Note: if you used the setup scripts, the changes below will have been applied for you

1. Open the `TodoListService\appsettings.json` file
2. Find the app key `Domain` and replace the existing value with your Azure AD tenant name.
3. Find the app key `TenantId` and replace the existing value with your Azure AD tenant ID.
4. Find the app key `ClientId` and replace the existing value with the application ID (clientId) of the `TodoListService-aspnetcore` application copied from the Azure portal.

Configure the client project

Note: if you used the setup scripts, the changes below will have been applied for you

1. Open the `TodoListWebApp\appsettings.json` file
2. Find the app key `Domain` and replace the existing value with your Azure AD tenant name.
3. Find the app key `TenantId` and replace the existing value with your Azure AD tenant ID.
4. Find the app key `ClientId` and replace the existing value with the application ID (clientId) of the `TodoListWebApp-aspnetcore` application copied from the Azure portal.
5. Find the app key `ClientSecret` and replace the existing value with the key you saved during the creation of the `TodoListWebApp-aspnetcore` app, in the Azure portal.
6. Find the app key `TodoListResourceId` and replace the existing value with `api://{TodoListService_ClientId}`.

Step 4: Run the sample

In the solution properties, set both projects as startup projects. Set **TodoListService** to run first. Clean the solution, rebuild it, and then run it.

On startup, the web API displays an empty web page. This is expected behavior.

Explore the sample by signing in into the web app, clicking on "Todo List", signing again if needed, adding items to the To Do list, signing-out, and starting again. Since the authenticated session is stored in a cookie, the application doesn't require logging in again if the previous session was never signed out.

[!NOTE] The To Do list is stored in memory in this sample. Each time the TodoListService API is stopped, any to-do lists are reset.

How to change the app URL

If you are using Visual Studio 2017

- Edit the TodoListService's properties (right click on TodoListService.csproj, and choose **Properties**)
- In the Debug tab (*Please check the url for your application and use that instead of localhost:44351*):
  - Check the **Launch browser** field to `https://localhost:44351/api/todolist`
  - Change the **App URL** field to be `https://localhost:44351` as this is the URL registered in the Azure AD application representing our Web API
  - Check the **Enable SSL** field

The same kind of modifications can be made on the `TodoListWebApp.csproj` project.

[!WARNING] Ensure that all of the app registration steps reflect any changes made to the URLs, or the sample won't function.

What to change when you deploy the sample to Azure

To deploy this sample to Azure:

- Update the various URLs (reply URLs, Base URL) in the *appsettings.json* files
- Add Reply URLs pointing to the deployed location, for both applications in the Azure portal