## Developing for User-Owns-Data Embedding using .NET Core

In this lab

To complete this lab, you must be able to run PowerShell scripts on your developer workstation. You must also install the following software if it is not already installed.

1) PowerShell cmdlet library for AzureAD

2) DOTNET Core SDK

3) Node.js

4) Visual Studio Code

5) Visual Studio 2019 (optional)

Please refer to the setup document for this lab to ensure you have all the

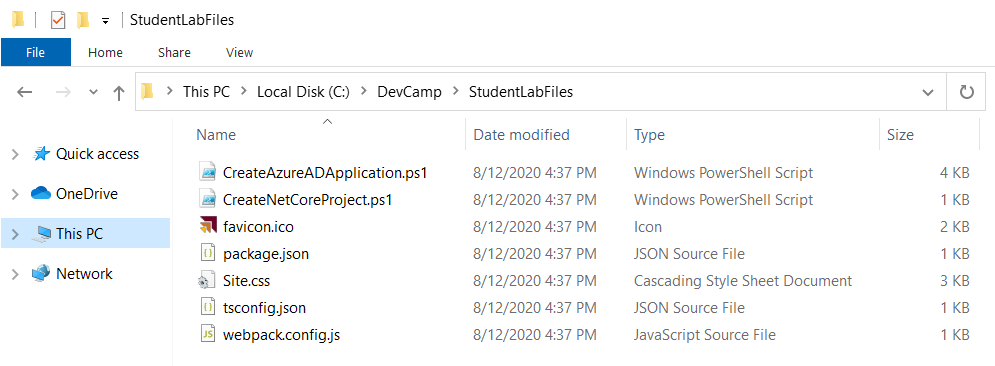
### Exercise 1: Create a New .NET Core MVC Web Application Project

In this exercise, you will create a new .NET Core project for an MVC web application.

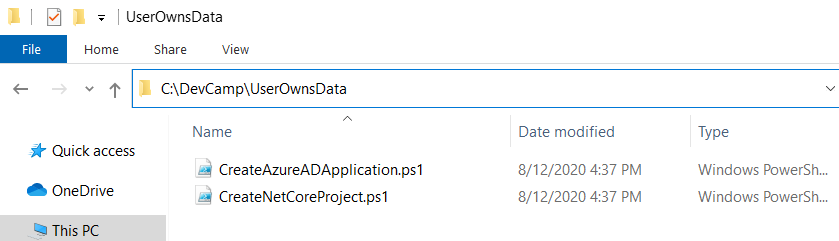
1. Download the student lab files and copy them to a local folder.
   1. Download the student lab files from GitHub

<https://github.com/TedPattison/NetCore-UserOwnsData-Tutorial/raw/master/StudentLabFiles.zip>

* 1. Copy the top-level folder inside the StudentLabFiles.zip to a to a local
  2. The StudentLabFiles folder should contain the set of files shown in the following screenshot.



1. Create a new .NET Core project using the .NET Core CLI and a PowerShell script.
   1. Create a new folder on your local drive named **UserOwnsData**.
   2. In the **StudentLabFiles** folder, locate the scripts named **CreateNetCoreProject.ps1** and **CreateAzureADApplication.ps1**.
   3. Copy **CreateNetCoreProject.ps1** and **CreateAzureADApplication.ps1** into the **UserOwnsData** folder.



1. Review the PowerShell code in **CreateNetCoreProject.ps1**.
   1. Open **CreateNetCoreProject.ps1** in a text editor such asNotepad or the PowerShell Integrated Scripting Environment (ISE).
   2. You can see **CreateNetCoreProject.ps1** includes commands to create a new project and add a few .NuGet packages.

dotnet new mvc --auth SingleOrg --framework netcoreapp3.1

dotnet remove package Microsoft.AspNetCore.Authentication.AzureAD.UI

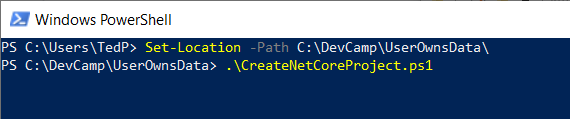
# update to latest available version of Microsoft.Identity.Web

dotnet add package Microsoft.Identity.Web -v 0.2.3-preview

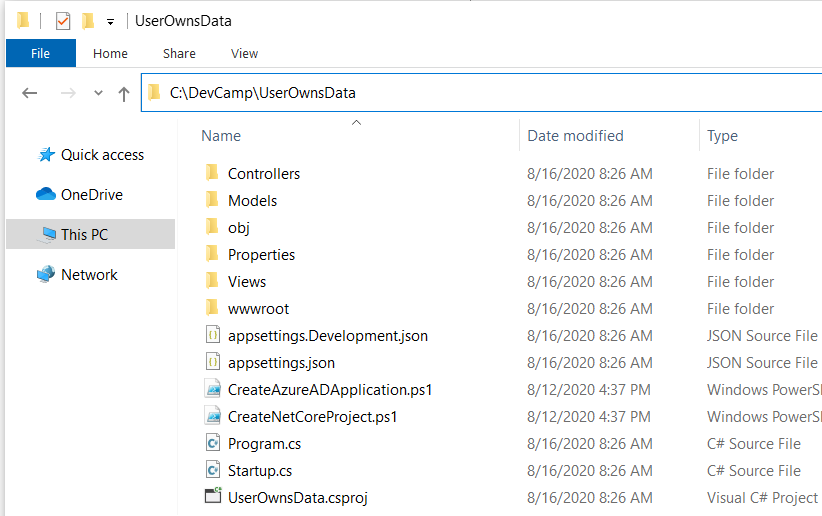
dotnet add package Microsoft.Identity.Web.UI -v 0.2.3-preview

dotnet add package Microsoft.PowerBi.Api

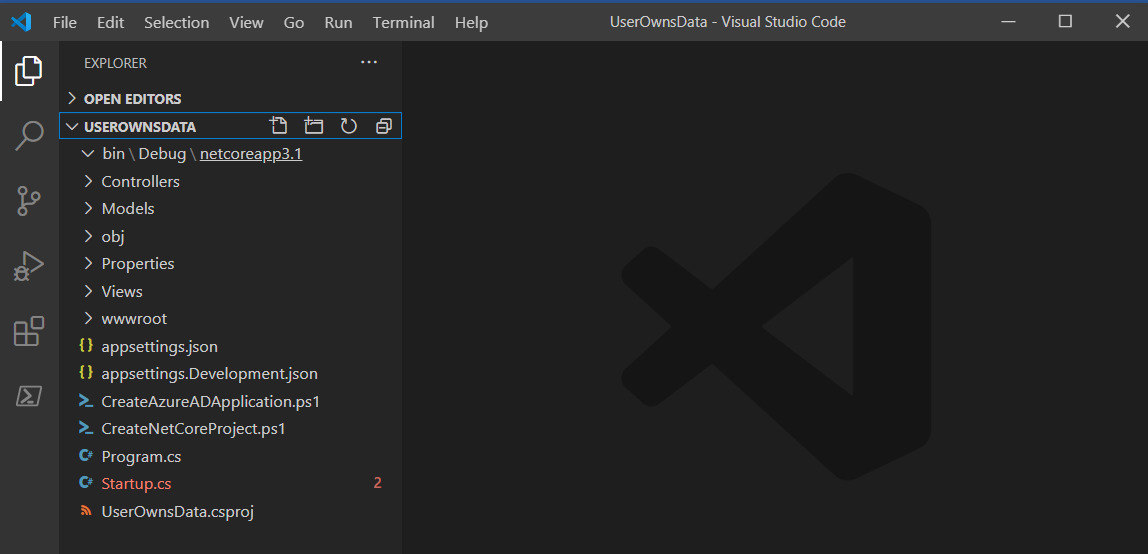
* 1. Open up a PowerShell console and set the location of the command prompt to the **UserOwnsData** folder.
  2. Execute the script **CreateNetCoreProject.ps1** and issuing the command **.\CreateNetCoreProject.ps1**.



* 1. Once the script has completed, you should see that the **UserOwnsData** folder has been populated with project files.



1. Open the **UserOwnsData** folder with Visual Studio Code
   1. Launch Visual Studio Code.
   2. Use the **Open Folder** command in Visual Studio Code to open the **UserOwnsData** folder.



You will not be able to build the project yet. That is because the PowerShell script removed the .NuGet package for Microsoft.AspNetCore.Authentication.AzureAD.UI and added two new packages for the Microsoft.Identity.Web library. You will have to modify the code in the project before it will build.

### Exercise 2: Implement User Login using Microsoft.Identity.Web

In this exercise, you will create a new confidential client application in the Azure portal and you will configure the application’s required permissions to provide the access you need to call into the Power BI Service API.

1. Creating a new Azure AD application by running a PowerShell script
   1. Open CreateAzureADApplication.ps1
   2. Sss

$authResult = Connect-AzureAD

* 1. Xxx

# create app secret

$newGuid = New-Guid

$appSecret = ([System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(($newGuid))))+"="

$startDate = Get-Date

$passwordCredential = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordCredential

$passwordCredential.StartDate = $startDate

$passwordCredential.EndDate = $startDate.AddYears(1)

$passwordCredential.KeyId = $newGuid

$passwordCredential.Value = $appSecret

* 1. Run the script

# create Azure AD Application

$aadApplication = New-AzureADApplication `

-DisplayName $appDisplayName `

-PublicClient $false `

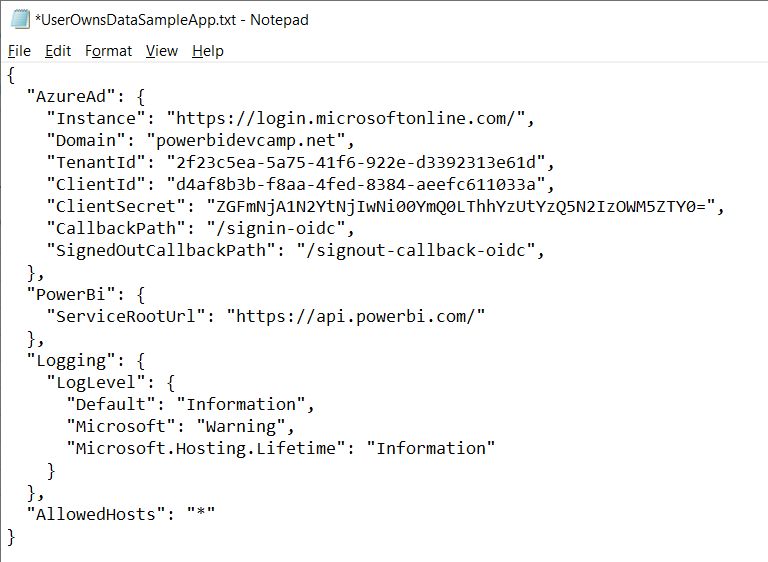
-AvailableToOtherTenants $false `

-ReplyUrls @($replyUrl) `

-Homepage $replyUrl `

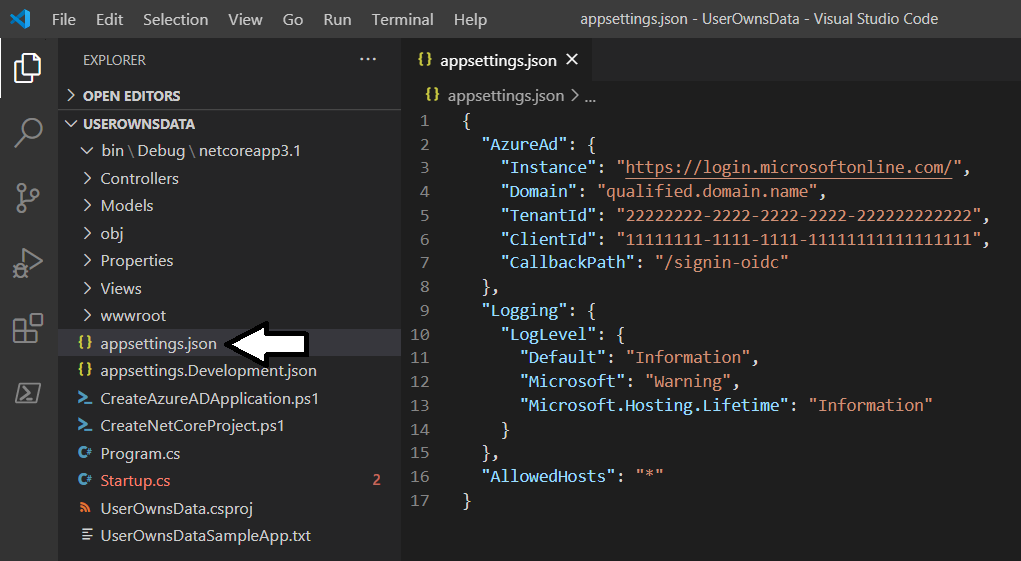
-PasswordCredentials $passwordCredential

* 1. Run the script

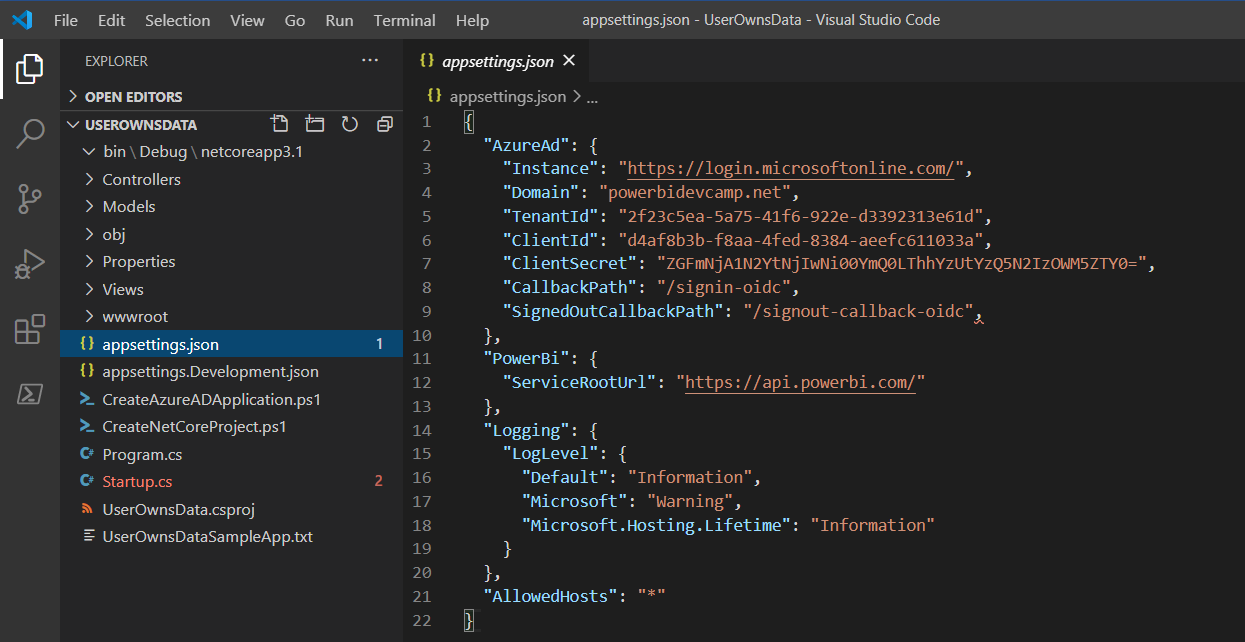


* 1. Cc

1. Modify appSettings.html
   1. Ssss

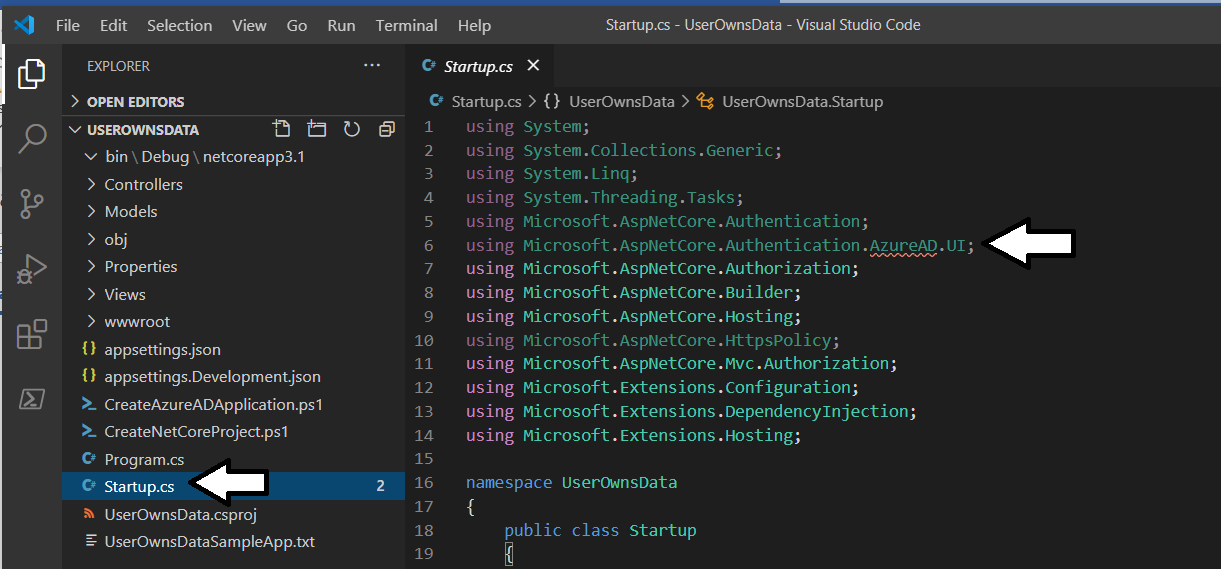


* 1. Zzzzz



* 1. x

1. Modify Startup.cs
   1. Open file in editor
   2. Remove line that imports Microsoft.AspnetCore.Authentication.AzureAD.UI which is causing a build error.



* 1. Below the existing import statements, add the following import statements.

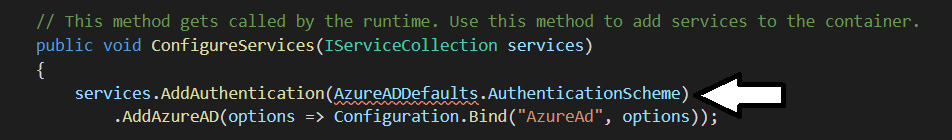
using Microsoft.Identity.Web;

using Microsoft.Identity.Web.TokenCacheProviders;

using Microsoft.Identity.Web.TokenCacheProviders.InMemory;

using Microsoft.Identity.Web.UI;

* 1. Locate ..



* 1. kkkkk

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services) {

services.AddMicrosoftWebAppAuthentication(Configuration);

* 1. Locate this section

services.AddControllersWithViews(options =>

{

var policy = new AuthorizationPolicyBuilder()

.RequireAuthenticatedUser()

.Build();

options.Filters.Add(new AuthorizeFilter(policy));

});

* 1. Ddd

var mvcBuilder = services.AddControllersWithViews(options => {

var policy = new AuthorizationPolicyBuilder()

.RequireAuthenticatedUser()

.Build();

options.Filters.Add(new AuthorizeFilter(policy));

});

mvcBuilder.AddMicrosoftIdentityUI();

* 1. D

public void ConfigureServices(IServiceCollection services) {

services.AddMicrosoftWebAppAuthentication(Configuration);

var mvcBuilder = services.AddControllersWithViews(options => {

var policy = new AuthorizationPolicyBuilder()

.RequireAuthenticatedUser()

.Build();

options.Filters.Add(new AuthorizeFilter(policy));

});

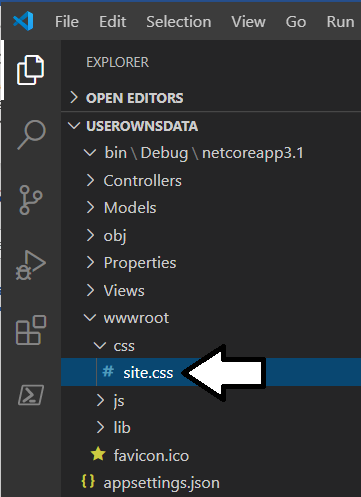
mvcBuilder.AddMicrosoftIdentityUI();

services.AddRazorPages();

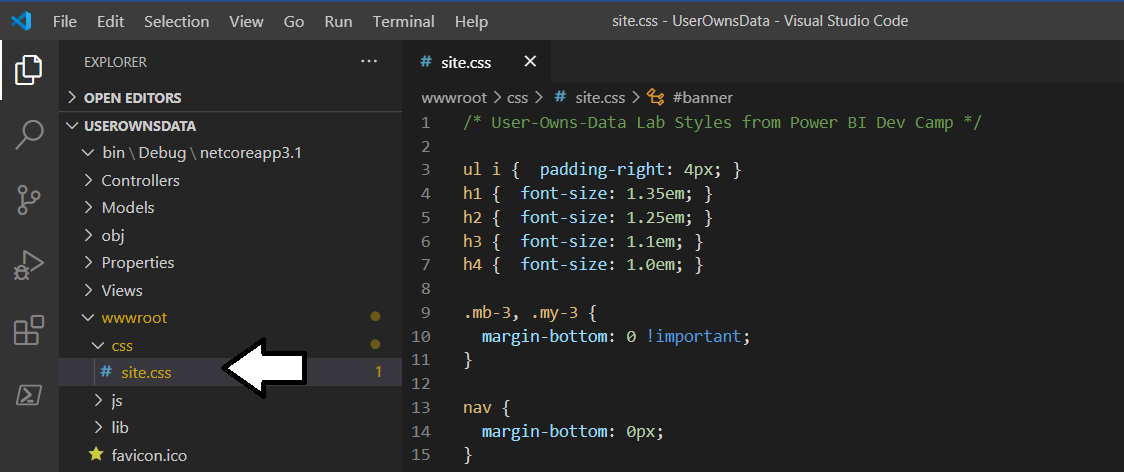
}

* 1. Save and close Startup.cs.

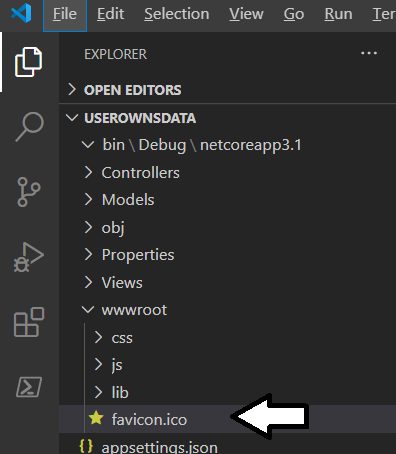
1. Copy a pre-written set of CSS styles into the application's Site.css file in the www/root/css folder
   1. Locate CCS file in StudentLabFiles folder named Site.css



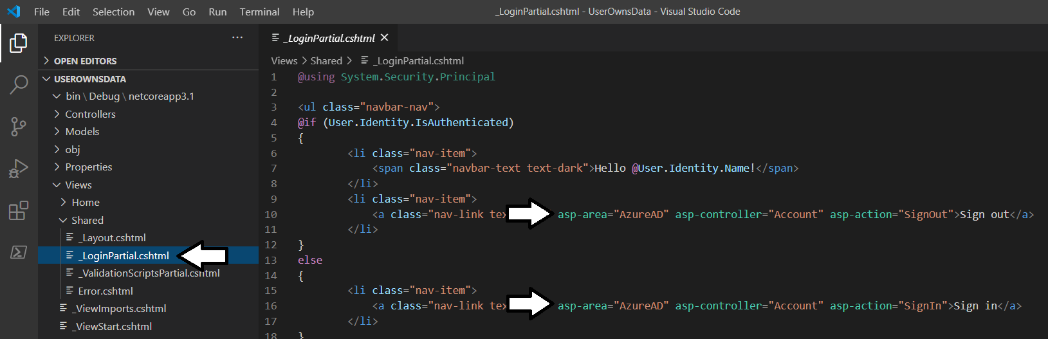
* 1. Copy CCS styles



1. Copy a custom favicon.ico
   1. Copy



1. Modify \_LoginPartial.cshtml.
   1. Sss



* 1. Z

@using System.Security.Principal

<ul class="navbar-nav">

@if (User.Identity.IsAuthenticated) {

<li class="nav-item">

<span class="navbar-text text-dark">Hello @User.FindFirst("name").Value</span>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="MicrosoftIdentity" asp-controller="Account" asp-action="SignOut">

Sign out

</a>

</li>

}

else {

<li class="nav-item">

<a class="nav-link text-dark" asp-area="MicrosoftIdentity" asp-controller="Account" asp-action="SignIn">

Sign in

</a>

</li>

}

</ul>

1. Modify Index.cshtml
   1. Sss

@using System.Security.Principal

@if (User.Identity.IsAuthenticated) {

<div class="jumbotron">

<h2>Welcome @User.FindFirst("name").Value</h2>

<p>You have now logged into this application.</p>

</div>

}

else {

<div class="jumbotron">

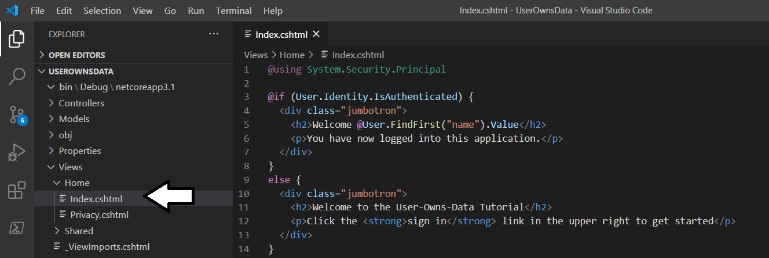
<h2>Welcome to the User-Owns-Data Tutorial</h2>

<p>Click the <strong>sign in</strong> link in the upper right to get started</p>

</div>

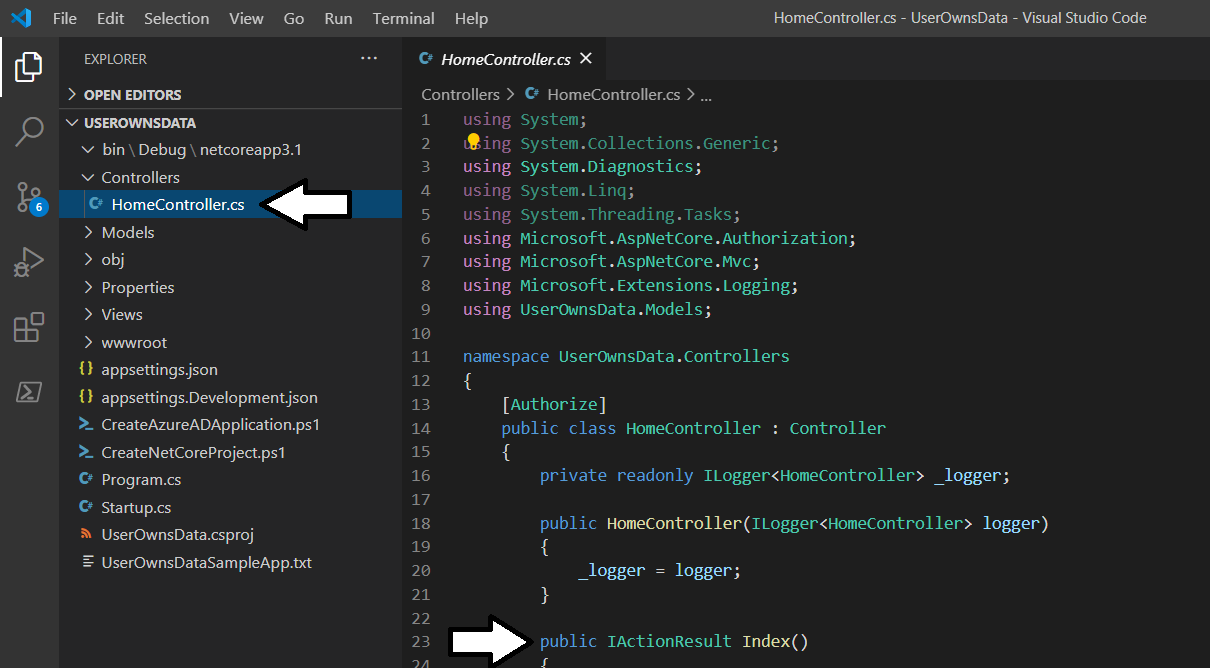
}

* 1. X



* 1. Save and close.

1. Modify HomeController.cs
   1. Dddd



* 1. X

[AllowAnonymous]

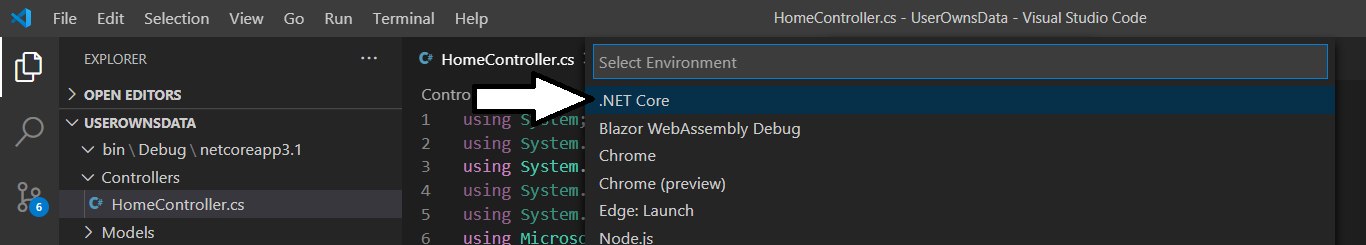
public IActionResult Index()

{

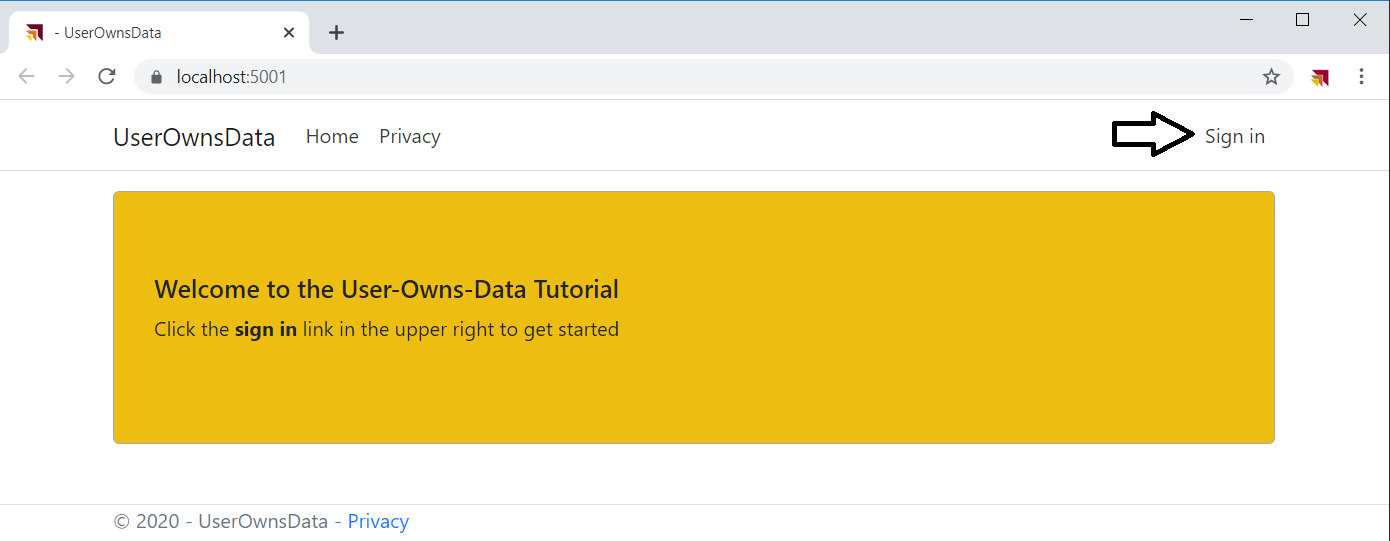
return View();

}

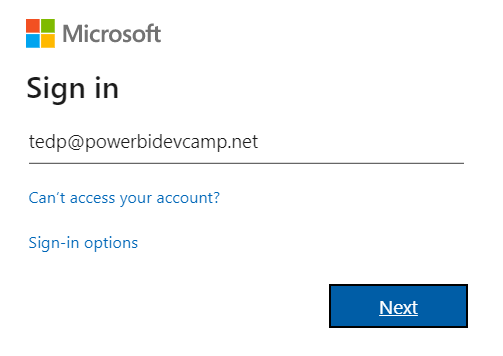
1. Test
   1. Press {F5}



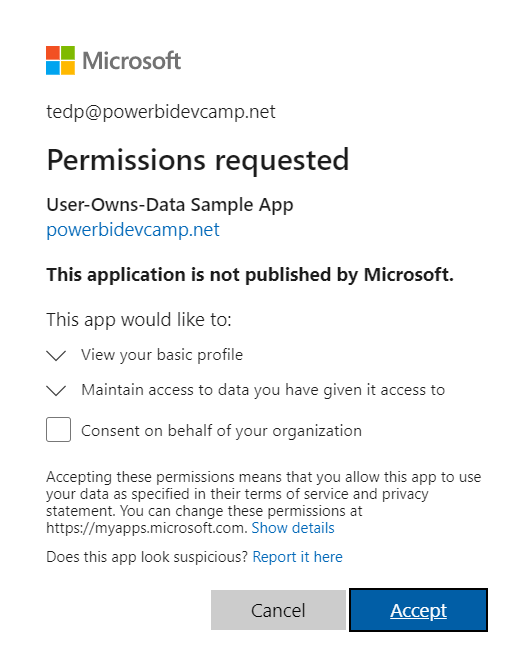
* 1. X



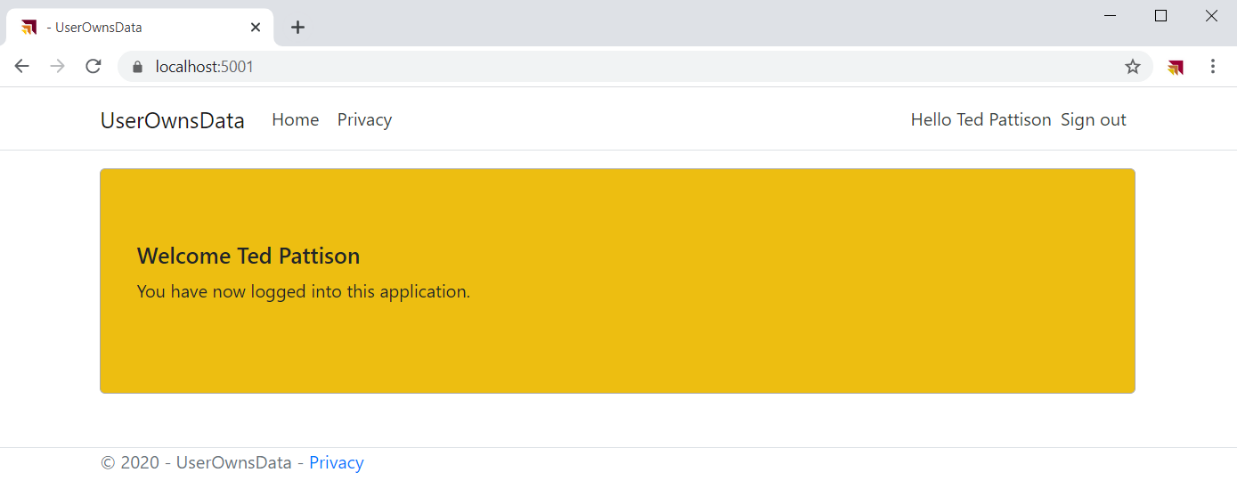
* 1. X



* 1. X



* 1. X



* 1. x

1. Add Embed page
   1. Add Embed method to Home controller.
   2. Add Modify Embed.cshtml
2. Modify Navigation in \_Layout.cshtml
   1. ssss

### Exercise 3: Call the Power BI Service API

In this exercise, you will create a new confidential client application in the Azure portal and you will configure the application’s required permissions to provide the access you need to call into the Power BI Service API.

1. In Power BI Service, create a new workspace and add a report.
   1. Get GUIDs for these
2. Add a new service for PowerBiApi.cs.
   1. sss
3. Modify Startup.cs to register service
4. Modify Home controller to inject PowerBiApi.
   1. Ssss
5. Add HTML to Embed.cshtml to create a table.
   1. Sss
6. Test

### Exercise 4: Embedding a Report using powerbi.js

In this exercise, you will create a new confidential client application in the Azure portal and you will configure the application’s required permissions to provide the access you need to call into the Power BI Service API.

1. Add client-side library support with Libman,json
   1. sss
2. Add view model to Embed.cshtml
   1. Sss
3. Add Embed,js
4. Test

### Exercise 5: Adding TypeScript Support to a .NET Core Project

In this exercise, you will create a new confidential client application in the Azure portal and you will configure the application’s required permissions to provide the access you need to call into the Power BI Service API.

1. Add package.json – npm init
   1. sss
2. Add tsconfig.json
   1. sss
3. Add webpack.config.js
   1. Ssss
4. Add embed.js
   1. Create Script folder
   2. Add embed.js
   3. Compile from command line
5. Update UserOwnsData.csproj
   1. sss

### Exercise 6: Creating a View Model for App Workspaces

In this exercise, you will create a new confidential client application in the Azure portal and you will configure the application’s required permissions to provide the access you need to call into the Power BI Service API.

1. Add new method to PowerBiApi.cs
   1. Sss
2. Modify Embed method in HomeController to call new method.
   1. Pass view model as object not as string
3. Rewrite Embed.cshtml
4. Rewrite Embed.ts.