

Configuring Azure AD Applications for Power BI Embedding

Setup Time: 60 minutes

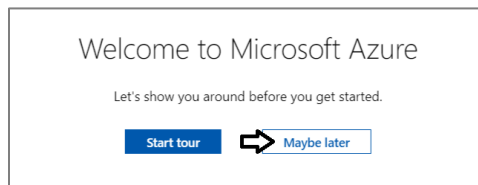
Lab Folder: C:\Student\Modules\02_AzureActiveDirectory\Lab

Overview: In this lab, you will create a new Azure AD application for a native client by hand in the Azure portal and then you use Visual Studio to create a C# console application that uses the Azure AD application to call the Power BI Service API. After that, you will create several more Azure AD applications using PowerShell scripts so you can get a few demo projects that use Power BI embedding up and running.

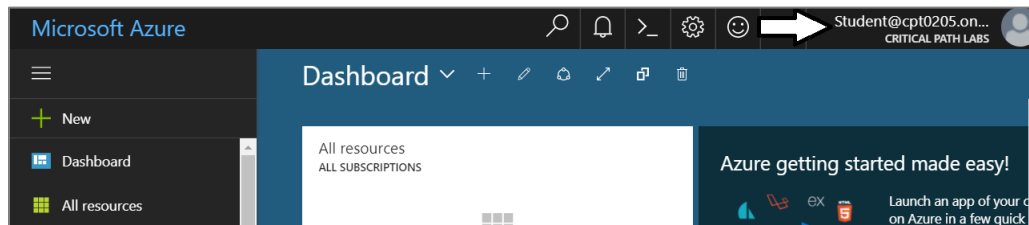
Exercise 1: Register a Native Client Application using the Azure Portal

In this exercise, you will register a new native client application with Azure AD and you will configure the application's required permissions to access the Power BI Service API.

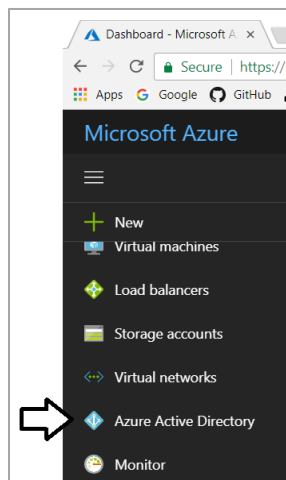
1. Log into the Azure Portal
 - a) In the browser, navigate to the Azure portal at <https://portal.azure.com>.
 - b) When you are prompted to log in, provide the credentials to log in with your Office 365 user account name.
 - c) If you are prompted to start a tour of Microsoft Azure, click **Maybe later**.



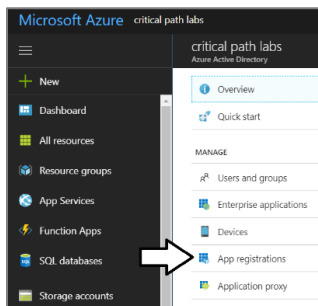
- d) Once you are logged into the Azure portal, check the email address in the login menu in the upper right to make sure you are logged in the Azure portal with the correct identity.



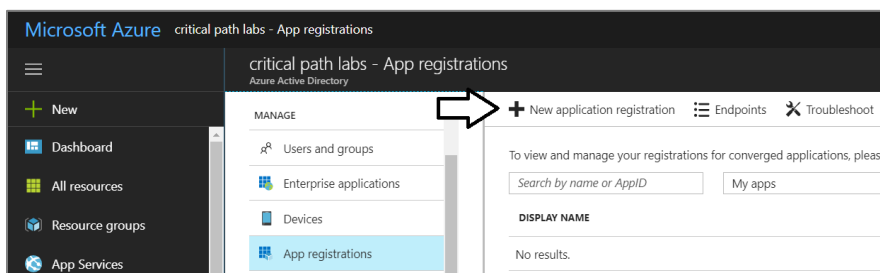
2. Register a new Azure AD application.
 - a) In the left navigation, scroll down and click on the link for **Azure Active Directory**.



- b) Click the link for **App registration**.



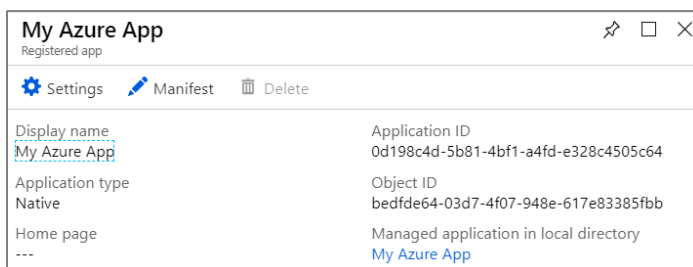
- c) Click **New application registration**.



- d) In the **Create** blade, enter the following information.
- Add a **Name** of **My Azure App**.
 - Set the **Application type** to **Native**.
 - Set the **Redirect URI** to <https://localhost/app1234>.
 - Click the **Create** button to create the new application.

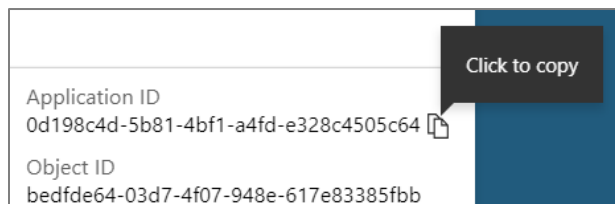
A screenshot of the 'Create' blade in the Azure portal. It contains a form with three fields: 'Name' with the value 'My Azure App', 'Application type' set to 'Native', and 'Redirect URI' with the value 'https://localhost/app1234'. Each field has a green checkmark indicating it is valid. At the bottom of the form is a blue 'Create' button.

- e) You should now see the summary page for the new Azure AD application.

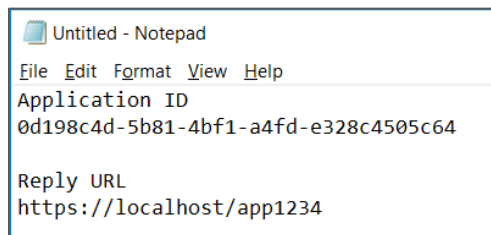


Azure AD will always generate a new GUID for the application ID any time you create a new Azure AD application.

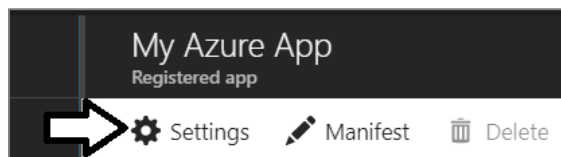
3. Copy the GUID for the Application ID.
 - a) Copy the Application ID to the Windows clipboard.



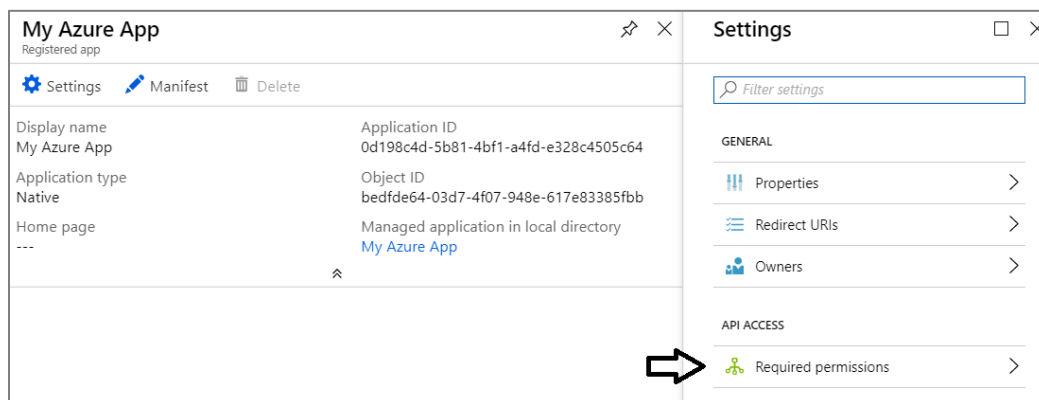
- b) Launch Notepad and paste the Application ID into a new document. Also add the value of the Redirect URI.



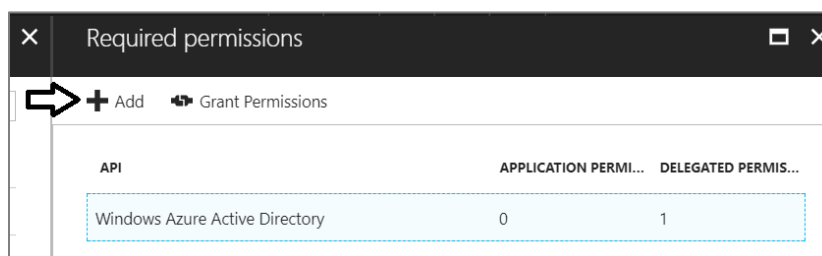
- c) Click on the **Settings** link to configure application settings,



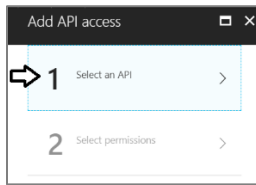
- d) In the **Settings** blade, click **Required permissions**.



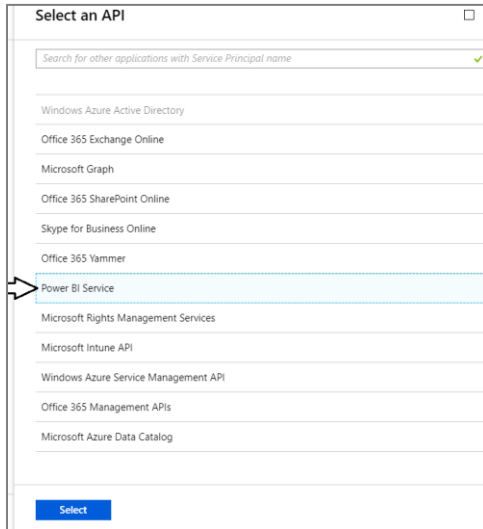
- e) Click the **Add** button on the **Required permissions** blade.



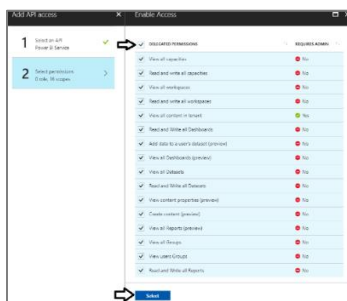
- f) Click the **Select an API** option in the **Add API** access blade.



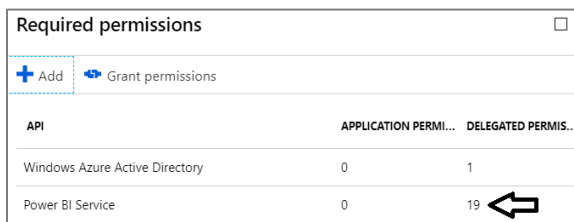
- g) In the **Select an API** blade, click **Power BI Service**.



- h) In the **Enable Access** blade, click the top checkbox for **DELEGATED PERMISSIONS** to select all the permissions.
i) Once you have selected all the permissions, click the **Select** button at the bottom of the blade.



- j) Click the **Done** button at the bottom of the **Add API Access** blade.
k) At this point, you should be able to verify that the Power BI Service has been added to the **Required permissions** list.

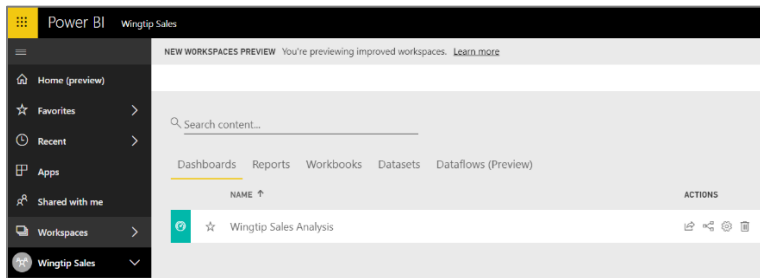


You are now done registering your application with Azure AD.

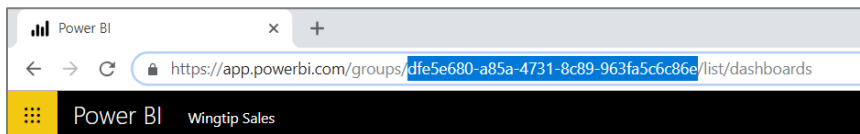
Exercise 2: Call the Power BI Service API from a C# Console Application

In this exercise, you will create a simple C# Console application to call into the Power BI Service API.

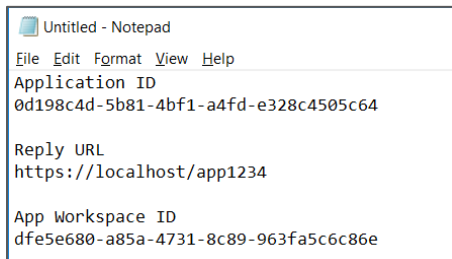
1. Determine the app workspace ID for the **Wingtip Sales** app workspace you created in earlier lab exercise.
 - a) Return to the browser and navigate back to the **Wingtip Sales** app workspace in the Power BI portal.



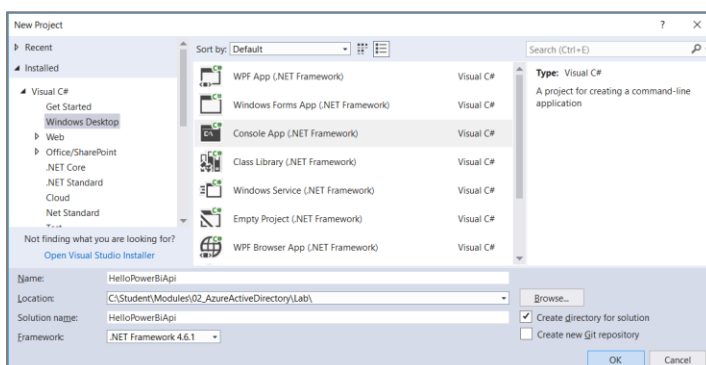
- b) In the browser address bar, select and copy the app workspace ID that appears just after **groups/** in the URL



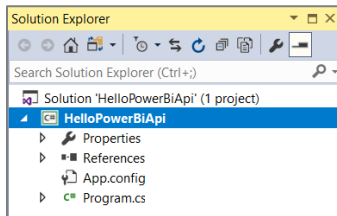
- c) Copy the app workspace ID to the text file you created in notepad as shown in the following screenshot.



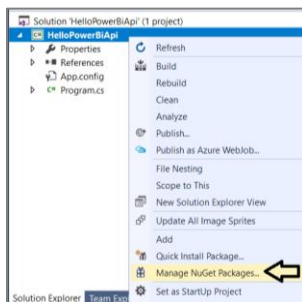
2. Create a new C# Console application in Visual Studio.
 - a) Launch Visual Studio.
 - b) Create a new project by running the **File > New Project** command.
 - c) Navigate to **Installed > Windows Desktop > Visual C#** project templates and select a project type of **Console App**.
 - d) Set the **Location** to **C:\Student\Modules\02_AzureActiveDirectory\Lab**.
 - e) Give the project a name of **HelloPowerBiApi** and click OK.



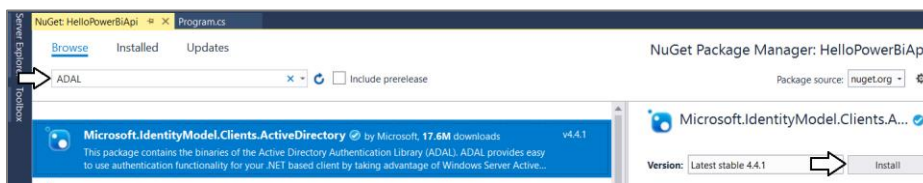
- f) You should now have a new project named **HelloPowerBiApi**.



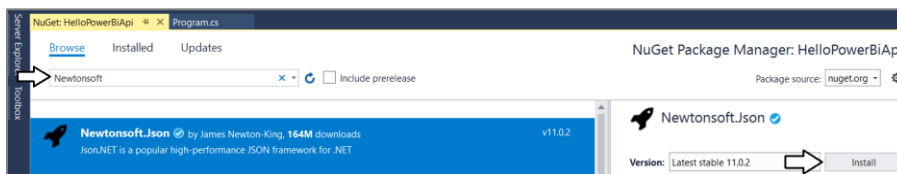
3. Install the NuGet packages for the Azure Active Directory Authentication Library and the Newtonsoft JSON converter.
- a) Right-click the top-level node for the **HelloPowerBiApi** project and select **Manage NuGet Packages...**.



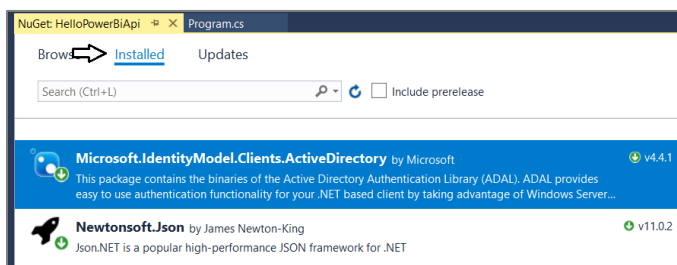
- b) Click the **Browse** tab and type **ADAL** into the search box.
- c) Select and install the Active Directory Authentication library package named **Microsoft.IdentityModel.Clients.ActiveDirectory**.



- d) When prompted about the licensing agreement, click **I Agree**.
- e) Search for **Newtonsoft** and then select and install the **Newtonsoft.Json** package.



- f) At this point, you should have two packages installed in your project as shown in the following screenshot.



- g) Close the window for the NuGet Package Manager.

4. Add starter code to the project by copying and pasting from a pre-provided text file.
 - a) Using Windows Explorer, locate the file named **PowerBiStarter.cs.txt** located in the **Student** folder at the following path.

C:\Student\Modules\02_AzureActiveDirectory\Lab\PowerBiStarter.cs.txt

- b) Open the file named **PowerBiStarter.cs.txt** in Notepad and copy its contents into the Window clipboard.
- c) Return to the **HelloPowerBiApi** project in Visual Studio.
- d) Open the source file named **program.cs**.
- e) Delete all the code inside **program.cs** and replace it with the content you copied into the Windows clipboard.
- f) You should now have the basic code for a simple C# console application which access the Power BI Service API.

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using Microsoft.IdentityModel.Clients.ActiveDirectory;

namespace HelloPowerBiServiceApi {

    class Program {

        const string aadAuthorizationEndpoint = "https://login.windows.net/common";
        const string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";

        static readonly Uri redirectUri = new Uri("https://localhost/app1234");

        const string clientId = "PASTE_YOUR_AZURE_APPLICATION_ID_HERE";
        const string appWorkspaceId = "PASTE_YOUR_POWER_BI_APP_WORKSPACE_ID_HERE";

        static string GetAccessToken() ...

        static string GetAccessTokenUnattended() ...

        static string ExecuteGetRequest(string restUrl) ...

        static void Main() ...

    }

    public class Report ...

    public class ReportCollection ...

}
```

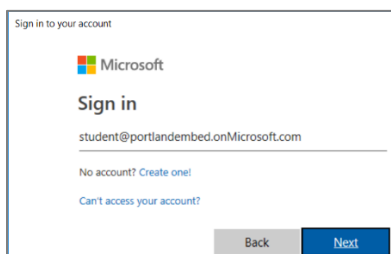
5. Update the code to include your Azure AD application ID and your Power BI app workspace ID.
 - a) Locate the section of the code with the static properties named **clientId** and **appWorkspaceId**.

```
const string clientId = "PASTE_YOUR_AZURE_APPLICATION_ID_HERE";
const string appWorkspaceId = "PASTE_YOUR_POWER_BI_APP_WORKSPACE_ID_HERE";
```

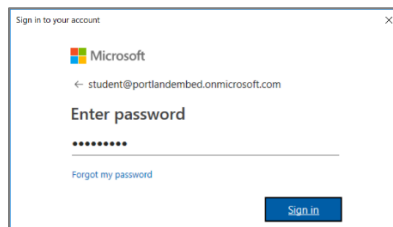
- b) Replace these values with the values you copied into Notepad earlier.

```
const string clientId = "0d198c4d-5b81-4bf1-a4fd-e328c4505c64";
const string appWorkspaceId = "dfe5e680-a85a-4731-8c89-963fa5c6c86e";
```

- c) Save your changes to **program.cs**.
6. Run the application to call to the Power BI Service API.
 - a) Press the **{F5}** key to begin a debugging session.
 - b) When prompted to sign in, enter the name of your Office 365 user account and click **Next**.

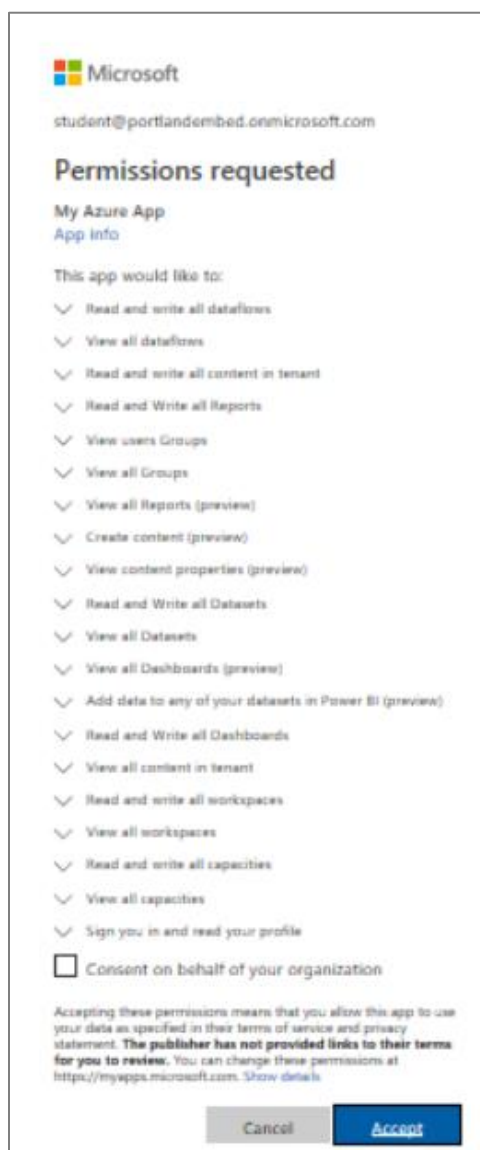


- c) Enter your password and click **Sign in**.



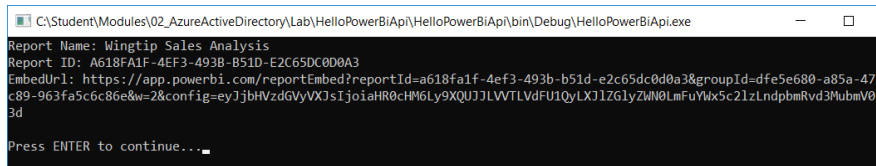
At this point, you have not consented to allow the application to call the Power BI Service API on your behalf. Therefore, Azure AD will prompt you with **the Permissions requested** dialog which will allow you to grant the application the permissions it needs to run.

- d) When prompted with the **Permissions requested** dialog, review the listed permissions and then click **Accept**.



In the simple scenario in this lab, your application will not require all these permissions. However, you have now seen all the possible permissions you can request for the Power BI Service API in the **Permissions requested** dialog.

- e) The application should now run and execute a request into the Power BI Service API to retrieve data the reports in the Wingtip Sales workspace. You should see output in the Console window about the **Wingtip Sales Analysis** report.



```

C:\Student\Modules\02_AzureActiveDirectory\Lab\HelloPowerBiApi\bin\Debug\HelloPowerBiApi.exe
Report Name: Wingtip Sales Analysis
Report ID: A618FA1F-4EF3-493B-B51D-E2C65DC0D0A3
EmbedUrl: https://app.powerbi.com/reportEmbed?reportId=a618fa1f-4ef3-493b-b51d-e2c65dc0d0a3&groupId=dfe5e680-a85a-473c89-963fa5c6c86e&w=2&config=eyJjbHVzdGVyVXJsIjoiaHR0cHM6Ly9XQUJ1LVVTVLdFU1QyLXJ1ZGlyZW90LmFuYmx5c21zLndpbmRvd3MubmV0I
3d
Press ENTER to continue...

```

Congratulations. You have now successfully called the Power BI Service API.

7. Modify the C# console application to acquire an access token using the User Password Credential flow.
- a) Inside the **Program** class in **Program.cs**, locate the method named **GetAccessTokenUnattended**.

```

static string GetAccessTokenUnattended() {
    string userName = "REPLACE_WITH_MASTER_USER_ACCOUNT_NAME";
    string userPassword = "REPLACE_WITH_MASTER_USER_ACCOUNT_PASSWORD";
    var authContext = new AuthenticationContext(aadAuthorizationEndpoint);
    var userPasswordCredential = new UserPasswordCredential(userName, userPassword);
    AuthenticationResult result =
        authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userPasswordCredential).Result;
    return result.AccessToken;
}

```

- b) Update the variables named **username** and **userPassword** to initialize them with your Office 365 user account credentials.

```

static string GetAccessTokenUnattended() {
    string userName = "student@portlandembed.onmicrosoft.com";
    string userPassword = "pass@word1";
    var authContext = new AuthenticationContext(aadAuthorizationEndpoint);
    var userPasswordCredential = new UserPasswordCredential(userName, userPassword);
    AuthenticationResult result =
        authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userPasswordCredential).Result;
    return result.AccessToken;
}

```

- c) Inspect the method named **ExecuteGetRequest** and locate the call to **GetAccessToken**.

```

static string ExecuteGetRequest(string restUrl) {
    HttpClient client = new HttpClient();
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
    request.Headers.Add("Authorization", "Bearer " + GetAccessToken());
    request.Headers.Add("Accept", "application/json;odata.metadata=minimal");
    HttpResponseMessage response = client.SendAsync(request).Result;
    if (response.StatusCode != HttpStatusCode.OK) {
        throw new ApplicationException("Error occurred calling the Power BI Service API");
    }
    return response.Content.ReadAsStringAsync().Result;
}

```

- d) Replace the call to the call to **GetAccessToken** with a call to **GetAccessTokenUnattended**.

```

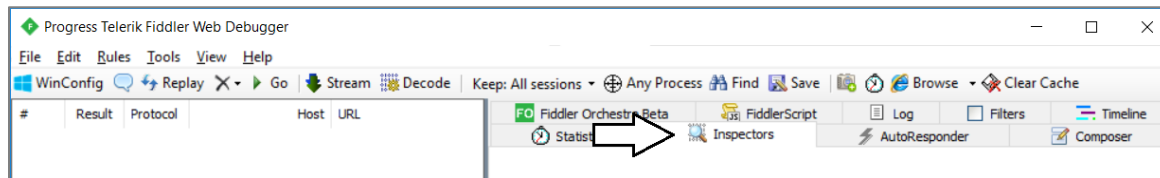
static string ExecuteGetRequest(string restUrl) {
    HttpClient client = new HttpClient();
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
    request.Headers.Add("Authorization", "Bearer " + GetAccessTokenUnattended());
    request.Headers.Add("Accept", "application/json;odata.metadata=minimal");
    HttpResponseMessage response = client.SendAsync(request).Result;
    if (response.StatusCode != HttpStatusCode.OK) {
        throw new ApplicationException("Error occurred calling the Power BI Service API");
    }
    return response.Content.ReadAsStringAsync().Result;
}

```

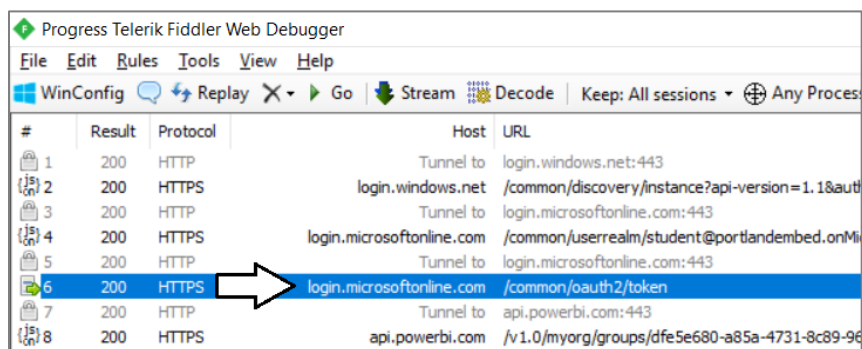
- e) Rerun the program by pressing the **{F5}** key.

The program should run now successfully with requiring an interactive login.

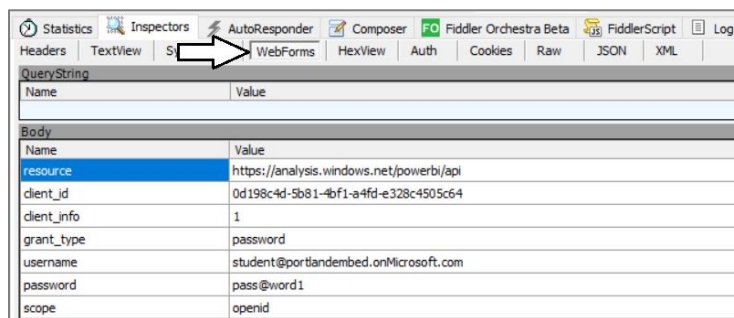
8. Inspect calls from your application to the Power BI Service API using the Fiddler utility.
 - a) Launch Fiddler.
 - b) Click on the **Inspectors** tab on the left so you can see details of the request and response for each HTTP request.



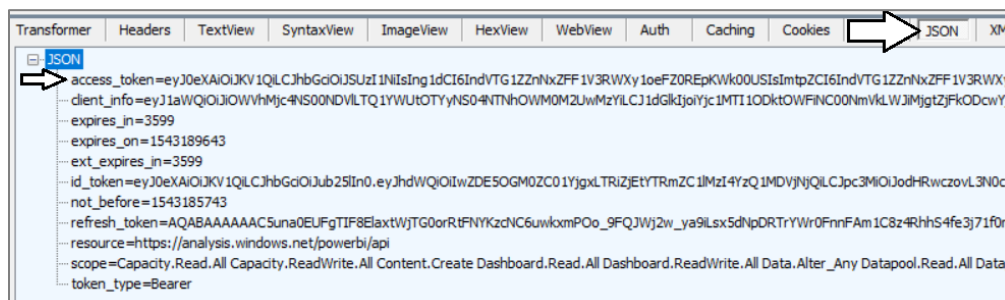
- c) Return to Visual Studio and run the **HelloPowerBiApi** application in the Visual Studio debugger by pressing the **{F5}** key.
- d) Once the application has run, return to Fiddler and examine the HTTP requests.
- e) Look through the HTTP requests in Fiddler and locate the call to **https://login.microsoftonline.com/common/oauth2/token**.



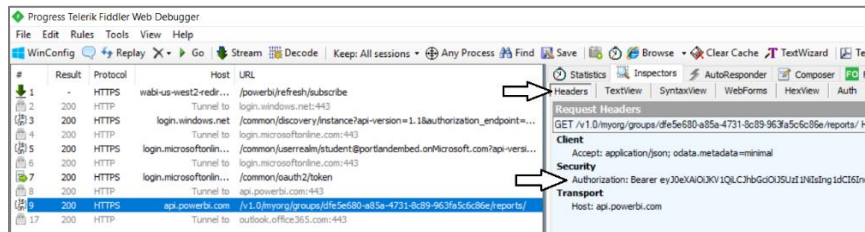
- f) On the right in the request details pane, click the **WebForms** tab so you can see the form variables passed in the request.
- g) You should be able to see form variables for **resource**, **client_id**, **grant_type**, **username**, **password** and **scope**.



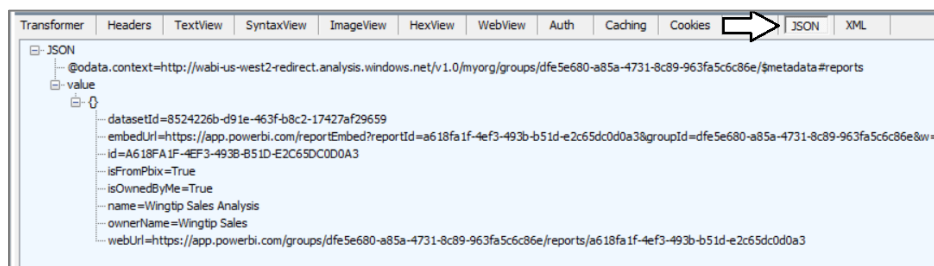
- h) On the right in the response details pane, click the **JSON** tab to see the results formatted as JSON.
- i) You should be able to see the JSON response which includes a value for the **access_token**.



- j) In the request list on the left, locate the call to the Power BI Service which has a **Host** value of **app.powerbi.com**.
- k) On the top right in the request details pane, click the **Headers** tab so you can see the request headers.
- l) You should be able to see the **Authorization** header which contains the text "**Bearer** " and then the access token.



- m) On the bottom right in the response details pane, click the **JSON** tab so you can see the response in a JSON format.
- n) Examine the response and view the values included for each report.



If you have never used Fiddler before, it's a great tool for debugging problems with your authentication code.

9. Close the browser, return to Visual Studio and terminate the current debugging session.

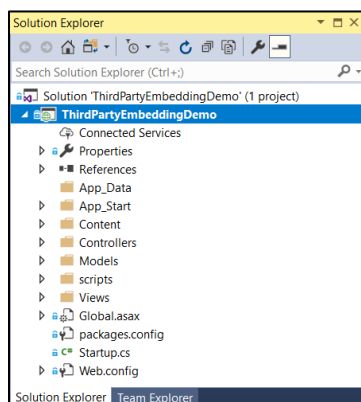
Exercise 3: Configure an Azure AD Application for Third Party Embedding

In this exercise, you will use Visual Studio to open and test out a sample ASP.NET MVC project named **ThirdPartyEmbeddingDemo**. Along the way, you will also create and configure a new Azure AD application for the sample application using a PowerShell script.

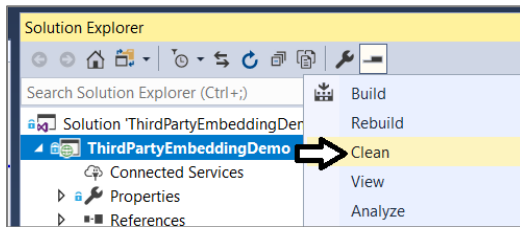
1. Open the Visual Studio demo project named **ThirdPartyEmbeddingDemo**.
 - a) Launch Visual Studio if it's not already running.
 - b) Choose **File > Open / Project/Solution....** and then select the project at the following location.

C:\Student\Demos\ThirdPartyEmbeddingDemo\ThirdPartyEmbeddingDemo.sln

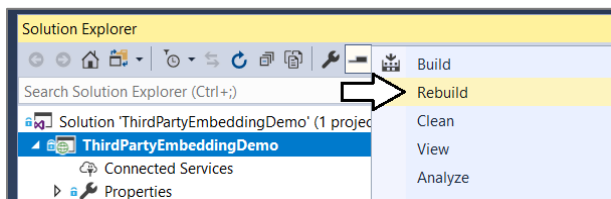
- c) The **ThirdPartyEmbeddingDemo** project should now be open in Visual Studio.



- d) Right-click the project in the Solution Explorer and select **Clean** to prepare for restoring the project's NuGet packages.



- e) Right-click the project in the Solution Explorer and select **Rebuild** to restore the project's NuGet packages.



2. Update **appSettings** in the project's **web.config** file.

- Open the **web.config** file that exists at the root folder of the project.
- Locate the following five **appSetting** values in the **appSettings** section,

```
<appSettings>

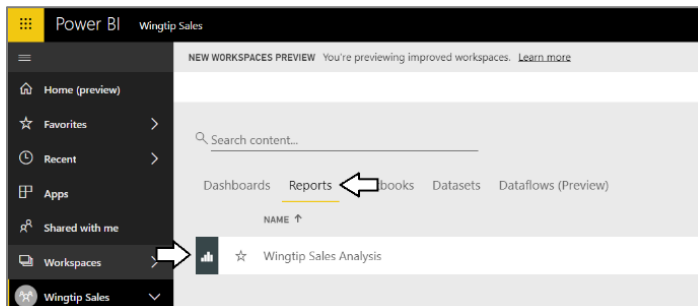
<!-- Update the following 5 app settings for your environment -->
<add key="aad-account-name" value="" />
<add key="aad-account-password" value="" />
<add key="application-id" value="" />
<add key="app-workspace-id" value="" />
<add key="report-id" value="" />
```

- Update the values for **aad-account-name** and **aad-account-password** for using your Office 365 user name and password.
- Update the values for the **application-id** and the **app-workspace-id** using the same values from the previous exercise.

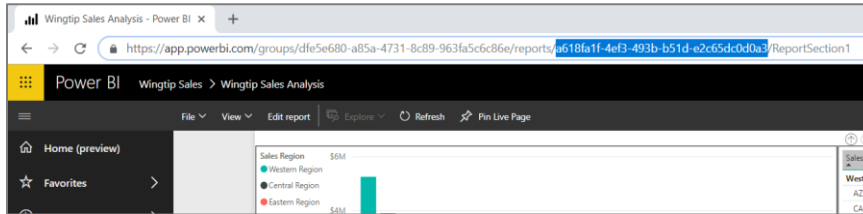
```
<!-- Update the following 5 app settings for your environment -->
<add key="aad-account-name" value="student@portlandembed.onmicrosoft.com" />
<add key="aad-account-password" value="pass@word1" />
<add key="application-id" value="0d198c4d-5b81-4bf1-a4fd-e328c4505c64" />
<add key="app-workspace-id" value="dfe5e680-a85a-4731-8c89-963fa5c6c86e" />
<add key="report-id" value="" />
```

You now need to retrieve the GUID which is the report ID for the report named **Wingtip Sales Analysis**.

- In the browser, navigate the Wingtip Sales app workspace and open the **Wingtip Sales Analysis** report.



- f) Copy the report ID from the browser address bar which appears in the URL after **reports/**.



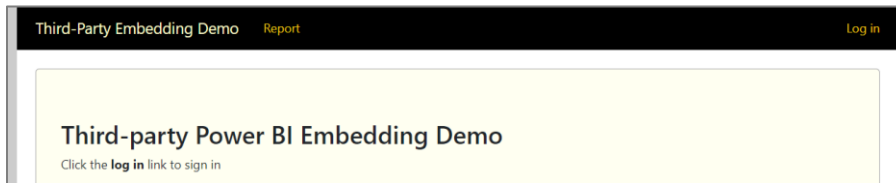
- g) Return to the **web.config** file in Visual Studio and update the **report-id** appSetting with the report ID.

```
<!-- Update the following 5 app settings for your environment -->
<add key="aad-account-name" value="student@portlandembed.onmicrosoft.com" />
<add key="aad-account-password" value="pass@word1" />
<add key="application-id" value="0d198c4d-5b81-4bf1-a4fd-e328c4505c64" />
<add key="app-workspace-id" value="dfe5e680-a85a-4731-8c89-963fa5c6c86e" />
<add key="report-id" value="a618fa1f-4ef3-493b-b51d-e2c65dc0d0a3" />
```

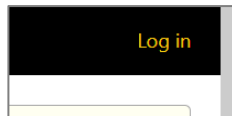
- h) Save and close **web.config**.

3. Run the application.

- a) Press the **{F5}** key to start the project in the Visual Studio debugger.



- b) Click the **Log in** link in the upper right corner of the page.



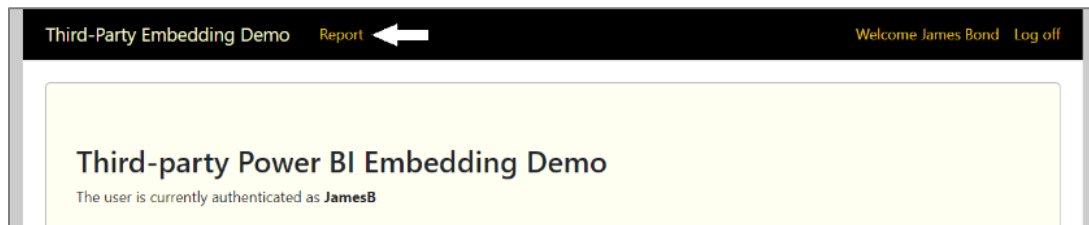
- c) When prompted for credentials, log in as **JamesB** with a password of **pass@word1**.

- d) The login menu should now display **Welcome James Bond**.



The authentication involves a private set of users defined by the application using ASP.NET Core Identity and Entity Framework.

- e) Click the **Report** link to navigate to the page with the embedded report.



- f) The application should display the same Power BI report you have been working with in earlier exercises.



4. Log on as a different user in the role of admin.

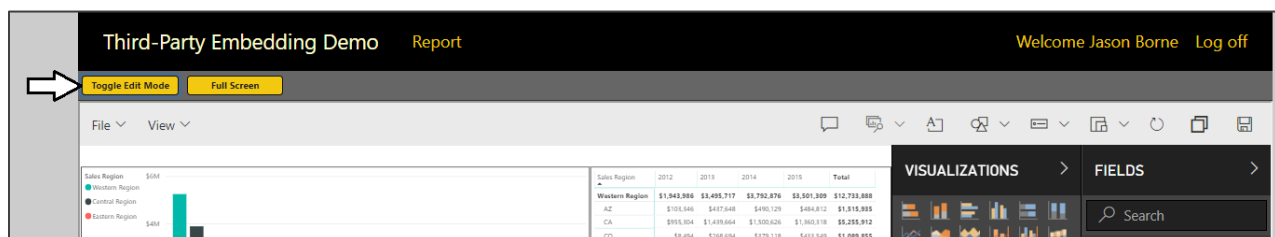
- a) Click the **Log off** link so you are now longer running as **JamesB**.



- b) Click the **Log in** link in the upper right corner of the page.
- c) When prompted for credentials, log in as **JasonB** with a password of **pass@word1**.
- d) The login menu should now display **Welcome Jason Borne**. This is a used in the role of admin.



- e) Click the **Report** link in the topnav bar to navigate to the page with the embedded report. Note that the application now gives the user an option to enter edit mode because the user is in the role of admin.



So far, you got the application up and running using the Azure AD application you created in Exercise 1. Over the next few steps you are going to create a new Azure AD application using a PowerShell script.

5. Use a PowerShell script to create a new Azure AD application.
 - a) Open a PowerShell script editor such as the PowerShell ISE or Visual Studio Code.
 - b) Open the PowerShell script at the following path.

C:\Student\Scripts\RegisterThirdPartyEmbeddingDemo.ps1

- c) Update the variables named **\$userName** and **\$password** with the credentials for your Office 365 user account.

```
RegisterThirdPartyEmbeddingDemo.ps1* X
1  Clear-Host
2
3  $userName = "student@portlandembed.onmicrosoft.com"
4  $password = "pass@word1"
5
```


- d) Save your changes to **RegisterThirdPartyEmbeddingDemo.ps1** and run the script.
 - e) When the script runs, it will create an Azure AD application and display the details in a text file as shown in this screenshot.

```
ThirdPartyEmbeddingDemo.txt - Notepad
File Edit Format View Help
--- Info for Third Party Embedding Demo ---
AppId: a7685382-04fe-4d29-9449-12c222ab4687
ReplyUrl: https://localhost:44300
```

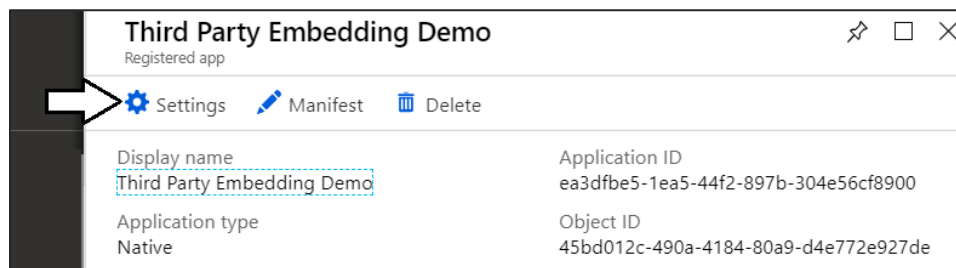
6. Update the value for the **application-id** in the **web.config** file.
 - a) Open the **web.config** file that exists at the root of the project.
 - b) Copy and paste the application ID from the text file into **web.config**.

```
<!-- Update the following 5 app settings for your environment -->
<add key="aad-account-name" value="student@portlandembed.onmicrosoft.com" />
<add key="aad-account-password" value="pass@word1" />
<add key="application-id" value="a7685382-04fe-4d29-9449-12c222ab4687" />
<add key="app-workspace-id" value="dfe5e680-a85a-4731-8c89-963fa5c6c86e" />
<add key="report-id" value="a618fa1f-4ef3-493b-b51d-e2c65dc0d0a3" />
```

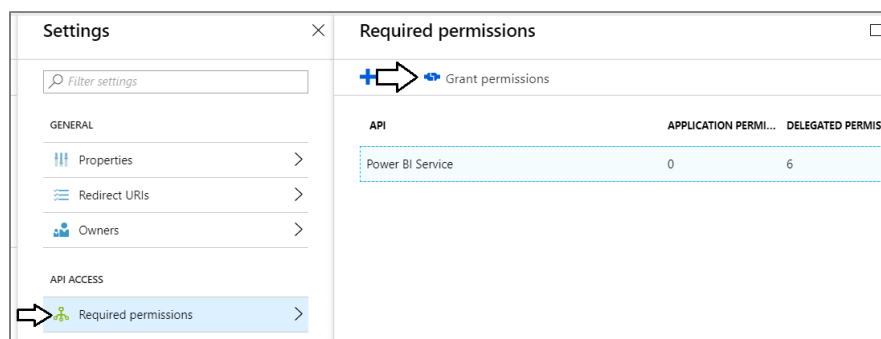
- c) Save your changes and close **web.config**.
7. Grant Permissions to the Azure AD application in the Azure Portal.
 - a) Navigate to the Azure portal.
 - b) In the left navigation, scroll down and click on the link for **Azure Active Directory**.
 - c) Click the link for **App registration**.
 - d) Locate and click on the application named **Third Party Embedding Demo** to see its summary view.

+ New application registration Endpoints Troubleshoot		
The preview experience for App registrations is available. Click this banner to launch the preview experience. →		
Search by name or AppID My apps ▾		
DISPLAY NAME	APPLICATION TYPE	APPLICATION ID
 Third Party Embedding Demo	Native	ea3dfbe5-1ea5-44f2-897b-304e56cf8900

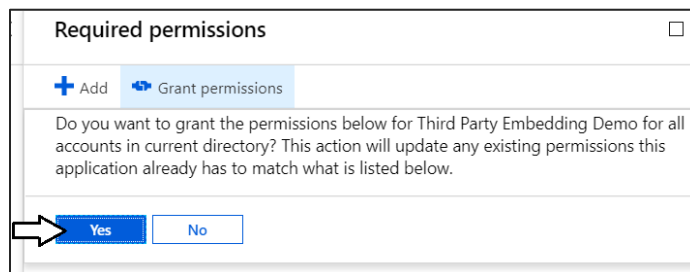
- e) Click the **Settings** button in the **Third Party Embedding Demo** summary view



- f) On the **Settings** blade, click **Required permissions**.
g) On the **Required permissions** blade, click **Grant permissions**.

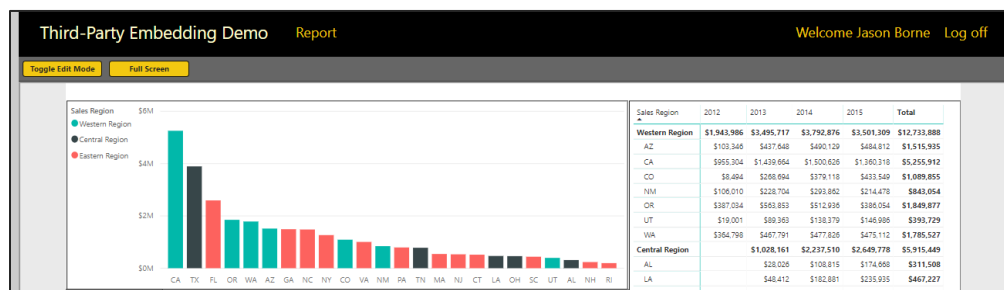


- h) When you are asked to confirm you want to grant permissions, click **Yes**.



Now that you have consented to application's required permissions, you should now be able to use the Third Party Embedding Demo application which acquires access tokens in an unattended fashion using the User Password Credential flow.

8. Run the Third Party Embedding Demo to make sure it works with the new Azure AD application.
a) Press the **{F5}** key to start up the application. You should be able to login and see the embedded report just as you did earlier.



9. Close the browser, return to Visual Studio and terminate the current debugging session.

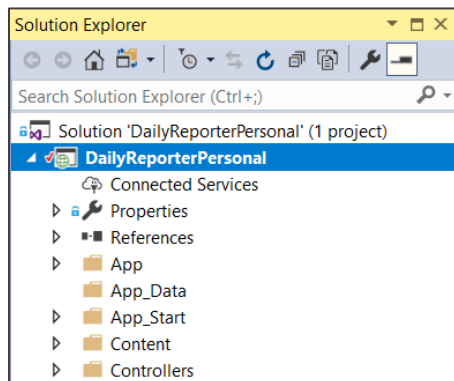
Exercise 4: Configure an Azure AD Application for First Party Embedding

In this exercise, you will use Visual Studio to open and test out a sample ASP.NET MVC project named **DailyReporterPersonal**. Along the way, you will also create and configure a new Azure AD application for the sample application using a PowerShell script.

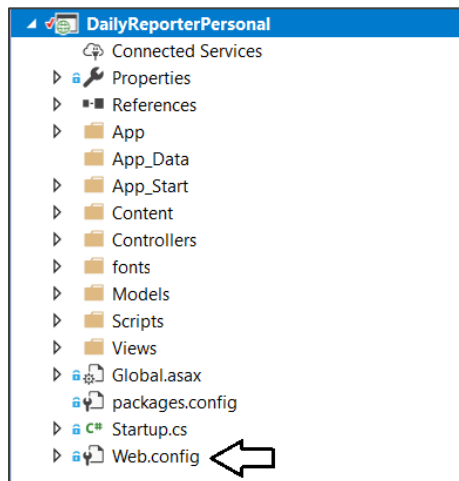
1. Open the Visual Studio demo project named **DailyReporterPersonal**.
 - a) Launch Visual Studio if it's not already running.
 - b) Choose **File > Open / Project/Solution....** and then select the project at the following location.

C:\Student\Demos\DailyReporterPersonal\DailyReporterPersonal.sln

- c) The **DailyReporterPersonal** project should now be open in Visual Studio.



- d) Right-click the project in the Solution Explorer and select **Clean** to prepare for restoring the project's NuGet packages.
 - e) Right-click the project in the Solution Explorer and select **Rebuild** to restore the project's NuGet packages.
2. Update **appSettings** in the project's **web.config** file.
 - a) Open the **web.config** file that exists at the root folder of the project.



- b) Locate the following **appSetting** values in the **appSettings** section.

```
<!-- Azure AD application data -->
<add key="client-id" value="" />
<add key="client-secret" value="" />
<add key="reply-url" value="https://localhost:44300" />
```

3. Use a PowerShell script to create a new Azure AD application.
 - a) Open a PowerShell script editor such as the PowerShell ISE or Visual Studio Code.
 - b) Open the PowerShell script at the following path.

C:\Student\Scripts\DailyReporterPersonal.ps1

- c) Update the variables named **\$userName** and **\$password** with the credentials for your Office 365 user account.

```
RegisterDailyReporterPersonal.ps1* X
1  Clear-Host
2
3  $userName = "student@portlandembed.onmicrosoft.com"
4  $password = "pass@word1"
5
```

- d) Save your changes to **RegisterDailyReporterPersonal.ps1** and run the script.
 - e) When the script runs, it will create an Azure AD application and display the details in a text file as shown in the this screenshot.

```
DailyReporterPersonal.txt - Notepad
File Edit Format View Help
|--- Info for Daily Reporter Personal ---
AppId: 12d191fb-e95a-40df-b2aa-c376888e294f
AppSecret: NWNiNDg00TctMmEwMi00NjkwLWI1ZTctY2UxYzVjOTMyNThl=
ReplyUrl: https://localhost:44300
```

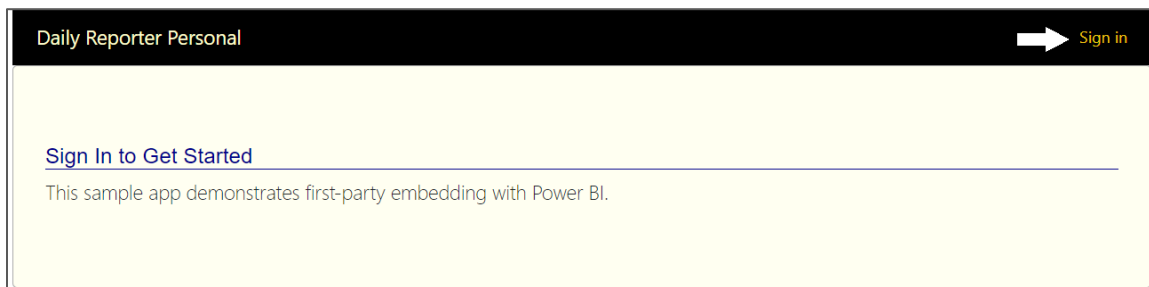
4. Update the value for **client-id** and **client-secret** in the **web.config** file.
 - a) Open the **web.config** file that exists at the root of the project.
 - b) Copy and paste the values for **AppId** and the **AppSecret** in the text file to **client-id** and **client-secret** in **web.config**.

```
<!-- Azure AD application data -->
<add key="client-id" value="12d191fb-e95a-40df-b2aa-c376888e294f" />
<add key="client-secret" value="NWNiNDg00TctMmEwMi00NjkwLWI1ZTctY2UxYzVjOTMyNThl=" />
<add key="reply-url" value="https://localhost:44300" />
```

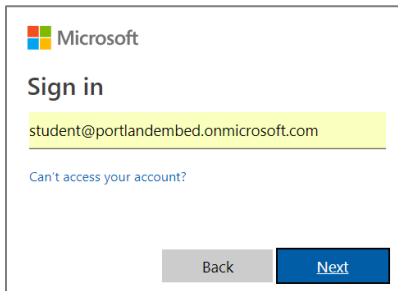
- c) Update the **app-workspace-id** in **web.config** using the same app workspace ID you used in previous lab exercises.

```
<!-- leave blank to use personal workspace -->
<add key="app-workspace-id" value="dfe5e680-a85a-4731-8c89-963fa5c6c86e" />
```

- d) Save your changes and close **web.config**.
5. Run the application.
 - a) Press the **{F5}** key to start the project in the Visual Studio debugger.
 - b) Once the application is running in the browser, click the **Sign in** link.



- c) When prompted to **Sign in**, enter the name of your Office 365 user account and click **Next**.



Microsoft

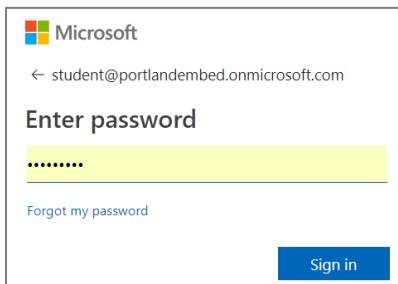
Sign in

student@portlandembed.onmicrosoft.com

[Can't access your account?](#)

[Back](#) [Next](#)

- d) When prompted to **Enter password**, enter your password and click **Sign in**.



Microsoft

← student@portlandembed.onmicrosoft.com

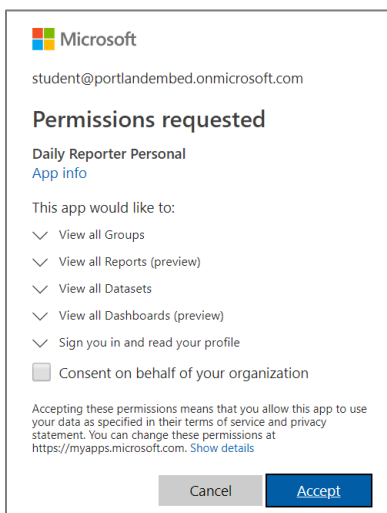
Enter password

.....

[Forgot my password](#)

[Sign in](#)

- e) When prompted with the **Permissions requested** dialog, click **Accept** to continue.



Microsoft

student@portlandembed.onmicrosoft.com

Permissions requested

Daily Reporter Personal
[App info](#)

This app would like to:

- ✓ View all Groups
- ✓ View all Reports (preview)
- ✓ View all Datasets
- ✓ View all Dashboards (preview)
- ✓ Sign you in and read your profile
- ☐ Consent on behalf of your organization

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at <https://myapps.microsoft.com>. [Show details](#)

[Cancel](#) [Accept](#)

- f) Once you have logged in, you should be able to see your name in the upper right corner of the page.



Daily Reporter Personal Hello Ted Pattison! [Sign out](#)

- Dashboards**
 - Wingtip Sales Analysis
- Reports**
 - Wingtip Sales Analysis
- Datasets**
 - Wingtip Sales Analysis

- g) Click the link for the **Wingtip Sales Analysis** in the **Reports** section to see the embedded report.



- h) Click the link for the **Wingtip Sales Analysis** in the **Dashboards** section to see the embedded dashboard.



6. Close the browser, return to Visual Studio and terminate the current debugging session.

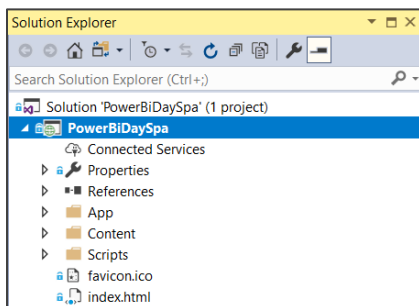
Exercise 5: Configure an Azure AD Application as an SPA with Implicit Flow

In this exercise, you will use Visual Studio to open and test out a sample single page application project named **PowerBiDaySPA**. Along the way, you will also create and configure a new Azure AD application for the sample application using a PowerShell script.

7. Open the Visual Studio demo project named **PowerBiDaySPA**.
- Launch Visual Studio if it's not already running.
 - Choose **File > Open / Project/Solution....** and then select the project at the following location.

C:\Student\Demos\PowerBiDaySPA\PowerBiDaySPA.sln

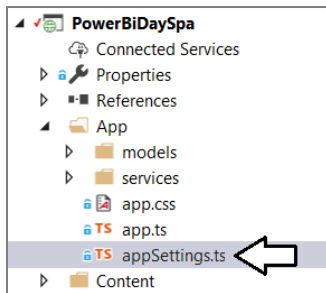
- c) The **PowerBiDaySPA** project should now be open in Visual Studio.



- Right-click the project in the Solution Explorer and select **Clean** to prepare for restoring the project's NuGet packages.
- Right-click the project in the Solution Explorer and select **Rebuild** to restore the project's NuGet packages.

8. Examine the **appSettings.ts** file.

- a) Open the **appSettings.ts** file that exists inside the **App** folder.



- b) Examine the static fields defined inside the **appSetting** class.

```
module myApp {  
  export class appSettings {  
    public static clientId: string = "";  
    public static appworkspaceId: string = "";  
    public static tenant: string = "YOUR_DOMAIN.onMicrosoft.com";  
  }  
}
```

9. Use a PowerShell script to create a new Azure AD application.

- a) Open a PowerShell script editor such as the PowerShell ISE or Visual Studio Code.
b) Open the PowerShell script at the following path.

C:\Student\Scripts\RegisterPowerBiDaySPA.ps1

- c) Update the variables named **\$userName** and **\$password** with the credentials for your Office 365 user account.

```
RegisterPowerBiDaySPA.ps1* X  
1 Clear-Host  
2  
3 $userName = "student@portlandembed.onmicrosoft.com"  
4 $password = "pass@word1"  
5
```

- d) Save your changes to **RegisterPowerBiDaySPA.ps1** and run the script.
e) When the script runs, it will create an Azure AD application and display the details in a text file as shown in this screenshot.

```
PowerBiDaySPA.txt - Notepad  
File Edit Format View Help  
--- Info for Power BI Day SPA ---  
AppId: 83b878c1-621a-4d7e-b5ed-52b26946811c  
ReplyUrl: https://localhost:44300  
Tenant: portlandembed.onmicrosoft.com
```

10. Update the field values for **clientId**, **appWorkspaceId** and **tenant** inside the **appSettings** class.




- a) Copy and paste the values for **AppId** and the **Tenant** to **clientId** and **tenant** in the **appSettings** class.
b) Update the **appWorkspaceId** in the **appSettings** class to use the same app workspace ID used in previous lab exercises.

```
export class appSettings {  
  public static clientId: string = "83b878c1-621a-4d7e-b5ed-52b26946811c";  
  public static appworkspaceId: string = "dfe5e680-a85a-4731-8c89-963fa5c6c86e";  
  public static tenant: string = "portlandembed.onmicrosoft.com";  
}
```

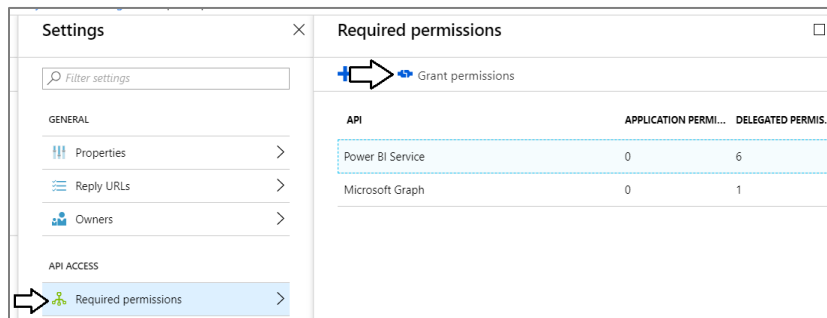
- c) Save your changes and close **appSettings.ts**.

11. Grant Permissions to the Azure AD application in the Azure Portal.

- Navigate to the Azure portal.
- In the left navigation, scroll down and click on the link for **Azure Active Directory**.
- Click the link for **App registration**.
- Locate and click on the application named **Power BI Day SPA** to see its summary view.

DISPLAY NAME	APPLICATION TYPE	APPLICATION ID
 Third Party Embedding Demo	Native	a7685382-04fe-4d29-9449-12c222ab4687
 Daily Reporter Personal	Web app / API	12d191fb-e95a-40df-b2aa-c376888e294f
 Power BI Day SPA	Web app / API	83b878c1-621a-4d7e-b5ed-52b26946811c

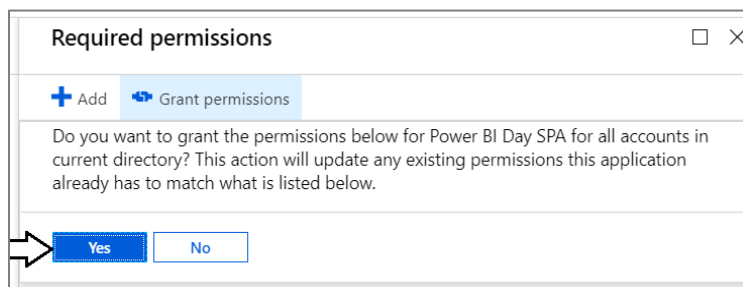
- Click the **Settings** button in the **Power BI Day SPA** summary view
- On the **Settings** blade, click **Required permissions**.
- On the **Required permissions** blade, click **Grant permissions**.



The screenshot shows the 'Required permissions' blade in the Azure Portal. On the left, the 'Settings' pane is open, and 'Required permissions' is selected under 'API ACCESS'. In the main pane, the 'Grant permissions' button is highlighted with a blue arrow. Below it, a table lists the permissions for the application:

API	APPLICATION PERMI...	DELEGATED PERMIS...
Power BI Service	0	6
Microsoft Graph	0	1

- When you are asked to confirm you want to grant permissions, click **Yes**.



The screenshot shows a confirmation dialog box titled 'Required permissions'. It contains the text: 'Do you want to grant the permissions below for Power BI Day SPA for all accounts in current directory? This action will update any existing permissions this application already has to match what is listed below.' At the bottom, there are two buttons: 'Yes' and 'No'. The 'Yes' button is highlighted with a blue arrow.

Now that you have consented to application's required permissions, you should now be able to use the Power BI Day SPA application which uses implicit flow and therefore cannot consent to required permissions when first accessing the application.

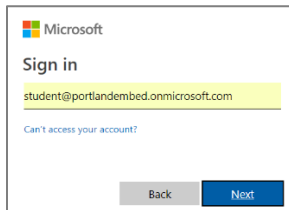
12. Run the application.

- Press the **{F5}** key to start the project in the Visual Studio debugger.
- Once the application is running in the browser, click the **Sign in** link.



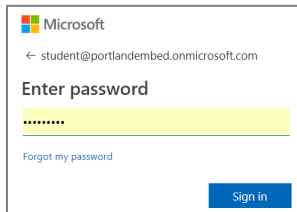
The screenshot shows the Power BI Day SPA application running in a browser. The header bar is dark gray with the text 'Power BI Day SPA' on the left and a 'Login' button on the right. The 'Login' button is highlighted with a blue arrow.

- c) When prompted to **Sign in**, enter the name of your Office 365 user account and click **Next**.



Microsoft
Sign in
student@portlandembed.onmicrosoft.com
Can't access your account?
Back Next

- d) When prompted to **Enter password**, enter your password and click **Sign in**.

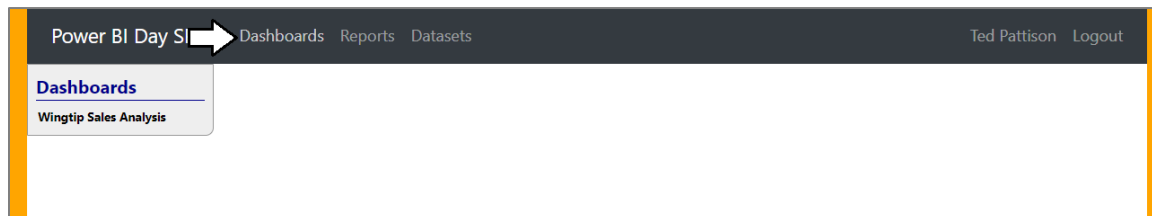


Microsoft
← student@portlandembed.onmicrosoft.com
Enter password
.....
Forgot my password
Sign in

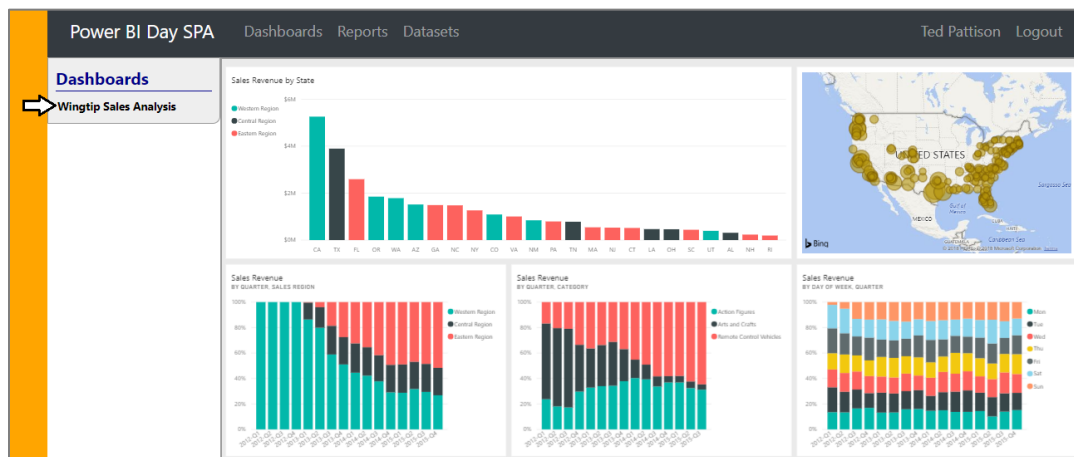
- e) Once you have logged in, you should be able to see your name in the upper right corner of the page.



- f) Click the link for the **Dashboards** link in the top navigation menu.



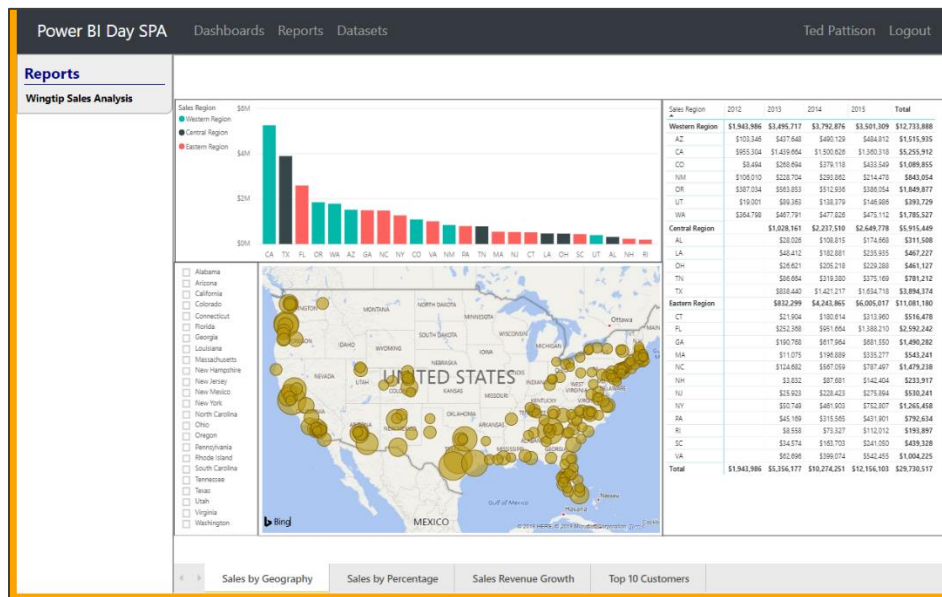
- g) Click the dashboard named **Wingtip Sales Analysis** in the left navigation to display it using Power BI embedding.



- h) Click the link for the **Reports** link in the top navigation menu.



- i) Click the report named **Wingtip Sales Analysis** in the left navigation to display it using Power BI embedding.



13. Close the browser, return to Visual Studio and terminate the current debugging session.

Congratulations. You have now completed this lab.