Working with the Power BI JavaScript API

Setup Time: 45 minutes

Lab Folder: C:\Student\Modules\05_PowerBiJavaScriptApi\Lab

Overview: In this lab you will work with a Visual Studio project named **PowerBiEmbeddedScrartchpad** which provides a proof-of-concept C# console application which generates HTML pages with the JavaScript code and security tokens required to implement Power BI embedding. The purpose of this lab exercise is for you to test and experiment with various client-side coding techniques used in Power BI embedding.
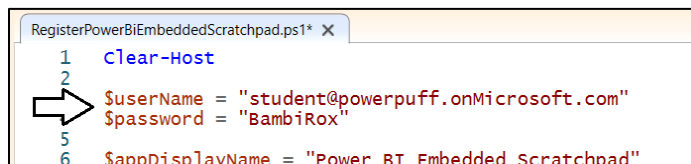
Exercise 1: Create an Azure AD Application for the PowerBI Embedded Scratchpad

In this exercise, you will create a Azure AD application for the **PowerBiEmbeddedScrartchpad** application using a PowerShell script.

1. Use a PowerShell script to create a new Azure AD application for the **Power BI Embedded Scratchpad** application.

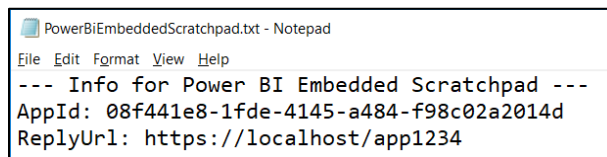   a) Open the PowerShell script at the following path using the Windows PowerShell ISE.

    ```
    C:\Student\Scripts\RegisterPowerBiEmbeddedScratchpad.ps1
    ```

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

    c) Save you changes to **RegisterPowerBiEmbeddedScratchpad.ps1** and run the script.

    d) 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.
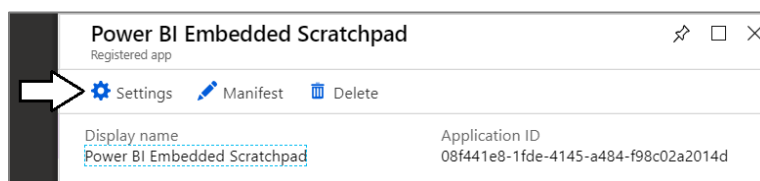

Leave this file open. In the next exercise, you will need to copy and paste the **AppId** value into a configuration file in Visual Studio.

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

    a) Navigate to the Azure portal.

    b) In the left navigation, click **Azure Active Directory** and then click the link for **App registration**.

    c) Locate and click on the application named **Power BI Embedded Scratchpad** to see its summary view.

    d) Click the **Settings** button in the **Power BI Embedded Scratchpad** summary view

e) On the **Settings** blade, click **Required permissions**.

f) On the **Required permissions** blade, click **Grant permissions**.
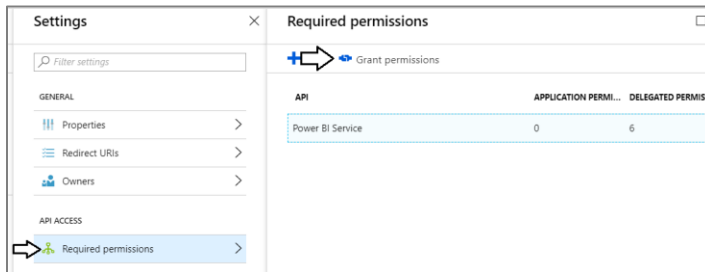
g) When you are asked to confirm you want to grant permissions, click **Yes**.
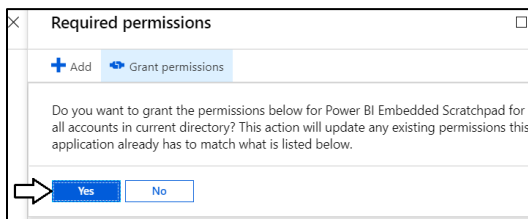
Now that you have consented to application's required permissions, you should now be able to use the Power BI Embedded Scratchpad application which acquires access tokens in an unattended fashion using the User Password Credential flow.

Exercise 2: Configure and Test the PowerBiEmbeddedScratchpad Application

In this exercise, you will configure and run the C# console application named **PowerBiEmbeddedScratchpad** using Visual Studio.

1. Open the Visual Studio demo project named **PowerBiEmbeddedScratchpad**.

   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\PowerBiEmbeddedScratchpad\PowerBiEmbeddedScratchpad.sln`

   c) The **PowerBiEmbeddedScratchpad** project should now be open in Visual Studio.
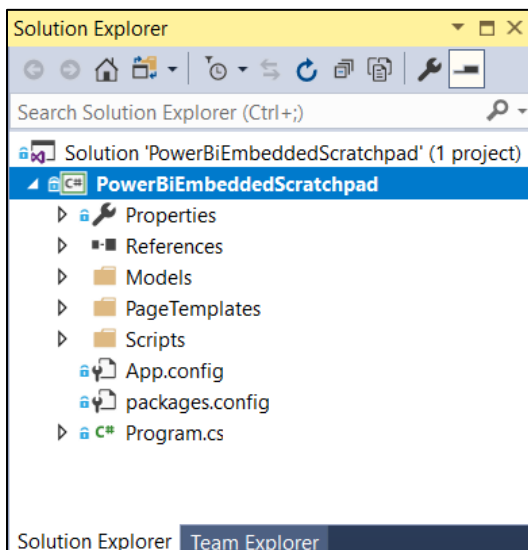
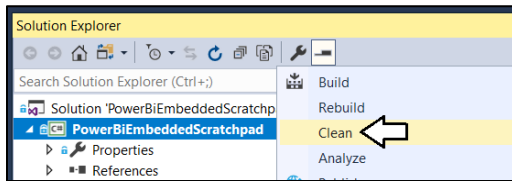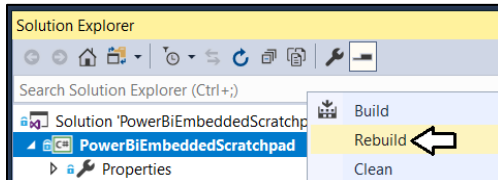d) Right-click the project in the Solution Explorer and select **Clean** to prepare for restoring the project's NuGet packages.

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 **App.config** file.

a) Open the **App.config** file that exists at the root folder of the project.

b) Locate the empty **appSetting** values in the **appSettings** section as shown in the following screenshot

```xml
<appSettings>

  <!-- add AAD account name and password for account with Power BI Pro license -->
  <!-- if you leave these settings blank, user will be presented with interactive login -->
  <add key="aad-account-name" value="" />
  <add key="aad-account-password" value="" />

  <!-- Create a native application in Azure AAD and give it all Power BI Service Permissions -->
  <!-- Add the client id and the reply url for this application here -->
  <add key="client-id" value="" />
  <add key="reply-url" value="https://localhost/app1234" />

  <!-- add the GUID for the workspace, dataset, report and dashboard you want to use for demos -->
  <add key="app-workspace-id" value="" />
  <add key="dataset-id" value="" />
  <add key="report-id" value="" />
  <add key="dashboard-id" value="" />

  <!-- configure where this application generates pages on local hard drive -->
  <!-- make sure your local-pages-folder value ends with a "\" -->
  <add key="local-pages-folder" value="C:\Student\LivePages\" />

</appSettings>
```

c) Update the values for **aad-account-name** and **aad-account-password** for using your Office 365 user name and password.

d) Update the values for the **client-id** and the **reply-url** using the values in the text file **PowerBiEmbeddedScratchpad.txt**.

e) Update the values for **app-workspace-id**, **dataset-id**, **report-id**, **dashboard-id** using the same values as you have used in previous labs to reference an app workspace, a dataset, a report and a dashboard.

```xml
<appSettings>

  <!-- add AAD account name and password for account with Power BI Pro license -->
  <!-- if you leave these settings blank, user will be presented with interactive login -->
  <add key="aad-account-name" value="student@powerpuff.onMicrosoft.com" />
  <add key="aad-account-password" value="BambiRox" />

  <!-- Create a native application in Azure AAD and give it all Power BI Service Permissions -->
  <!-- Add the client id and the reply url for this application here -->
  <add key="client-id" value="08f441e8-1fde-4145-a484-f98c02a2014d" />
  <add key="reply-url" value="https://localhost/app1234" />

  <!-- add the GUID for the workspace, dataset, report and dashboard you want to use for demos -->
  <add key="app-workspace-id" value="7266b3e4-b3b4-46b4-b1c0-9e35ed553b06" />
  <add key="dataset-id" value="02f7e8d5-434c-47e2-aa7e-3268f44a63d9" />
  <add key="report-id" value="484a268e-0c1a-4bdc-b98b-5654188b46d0" />
  <add key="dashboard-id" value="4b293a22-1561-4cab-af1e-a668825df0d2" />

  <!-- configure where this application generates pages on local hard drive -->
  <!-- make sure your local-pages-folder value ends with a "\" -->
  <add key="local-pages-folder" value="C:\Student\LivePages\" />

</appSettings>
```

f) Save your changes to **App.config**.

3. Run the **PowerBiEmbeddedScratchpad** application in the Visual Studio debugger.

    a) Run the application in the Visual Studio debugger by pressing the **{F5}** key.

   b) The application should generate an HTML page and then launch it in the Chrome browser.
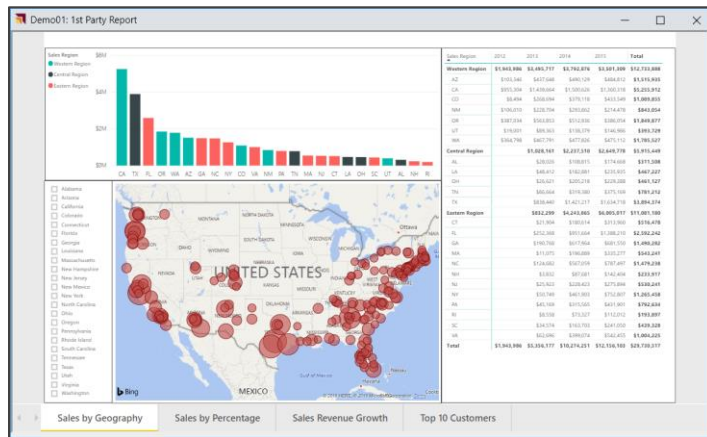
    c) Test out the experience of interacting with the embedded report.

    d) Resize the window and observe how the reports is able to automatically adapt to the new size.

    e) Once you have tested the report, close the browser window.

4. Inspect the HTML files created by the **PowerBiEmbeddedScratchpad** application.

    a) Open the Windows Explorer and navigate to the following folder.

```
C:\Student\LivePages
```

    b) You should see a page has been created named **Demo01-EmbedReportFirstParty.html**.
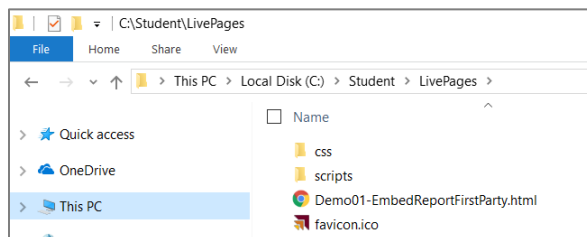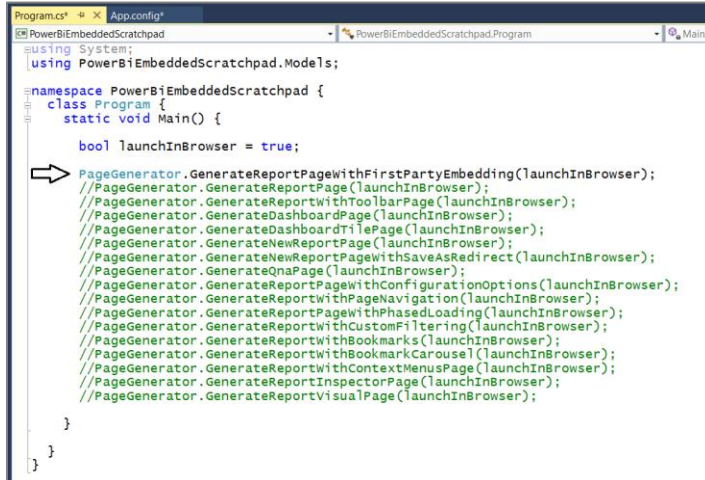

    c) Double-click on **Demo01-EmbedReportFirstParty.html** to open it again in a browser.
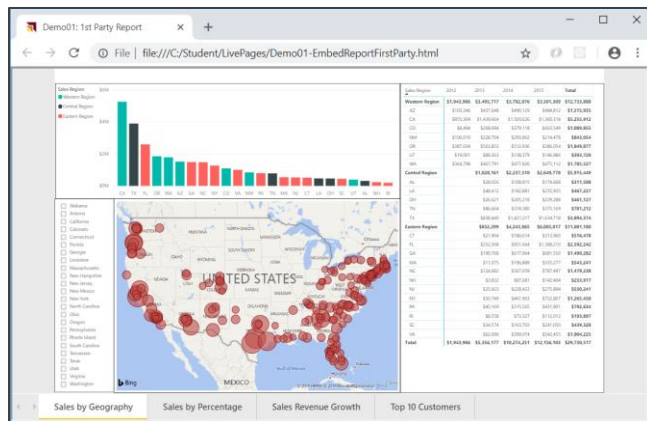
    d) Once you have tested the report, close the browser window.

5. Generate more samples pages using the **PowerBiEmbeddedSratchpad** sample application.

    a) Return to the **PowerBiEmbeddedSratchpad** project in Visual Studio.

    b) Open the C# source file named **Program.cs**.

    c) Currently, the code inside **Program.cs** is only executing one method of the **PageGenerator** class.

```
Program.cs*  ⊕ ×  App.config*
PowerBiEmbeddedScratchpad                              PowerBiEmbeddedScratchpad.Program                    Main()
 using System;
 using PowerBiEmbeddedScratchpad.Models;

 namespace PowerBiEmbeddedScratchpad {
   class Program {
     static void Main() {

       bool launchInBrowser = true;

  ⇨    PageGenerator.GenerateReportPageWithFirstPartyEmbedding(launchInBrowser);
       //PageGenerator.GenerateReportPage(launchInBrowser);
       //PageGenerator.GenerateReportWithToolbarPage(launchInBrowser);
       //PageGenerator.GenerateDashboardPage(launchInBrowser);
       //PageGenerator.GenerateDashboardTilePage(launchInBrowser);
       //PageGenerator.GenerateNewReportPage(launchInBrowser);
       //PageGenerator.GenerateNewReportPageWithSaveAsRedirect(launchInBrowser);
       //PageGenerator.GenerateQnaPage(launchInBrowser);
       //PageGenerator.GenerateReportPageWithConfigurationOptions(launchInBrowser);
       //PageGenerator.GenerateReportWithPageNavigation(launchInBrowser);
       //PageGenerator.GenerateReportPageWithPhasedLoading(launchInBrowser);
       //PageGenerator.GenerateReportWithCustomFiltering(launchInBrowser);
       //PageGenerator.GenerateReportWithBookmarks(launchInBrowser);
       //PageGenerator.GenerateReportWithBookmarkCarousel(launchInBrowser);
       //PageGenerator.GenerateReportWithContextMenusPage(launchInBrowser);
       //PageGenerator.GenerateReportInspectorPage(launchInBrowser);
       //PageGenerator.GenerateReportVisualPage(launchInBrowser);

     }
   }
 }
```

    d) Update the code to initialize the **launchInBrowser** variable with a value of **false**.

```
bool launchInBrowser = false;
```

    e) Uncomment the code that calls the other methods from the **PageGenerator** class as shown in the following screenshot.

```
static void Main() {

  bool launchInBrowser = false;|

  PageGenerator.GenerateReportPageWithFirstPartyEmbedding(launchInBrowser);
  PageGenerator.GenerateReportPage(launchInBrowser);
  PageGenerator.GenerateReportWithToolbarPage(launchInBrowser);
  PageGenerator.GenerateDashboardPage(launchInBrowser);
  PageGenerator.GenerateDashboardTilePage(launchInBrowser);
  PageGenerator.GenerateNewReportPage(launchInBrowser);
  PageGenerator.GenerateNewReportPageWithSaveAsRedirect(launchInBrowser);
  PageGenerator.GenerateQnaPage(launchInBrowser);
  PageGenerator.GenerateReportPageWithConfigurationOptions(launchInBrowser);
  PageGenerator.GenerateReportWithPageNavigation(launchInBrowser);
  PageGenerator.GenerateReportPageWithPhasedLoading(launchInBrowser);
  PageGenerator.GenerateReportWithCustomFiltering(launchInBrowser);
  PageGenerator.GenerateReportWithBookmarks(launchInBrowser);
  PageGenerator.GenerateReportWithBookmarkCarousel(launchInBrowser);
  PageGenerator.GenerateReportWithContextMenusPage(launchInBrowser);
  PageGenerator.GenerateReportInspectorPage(launchInBrowser);
  PageGenerator.GenerateReportVisualPage(launchInBrowser);

}
```
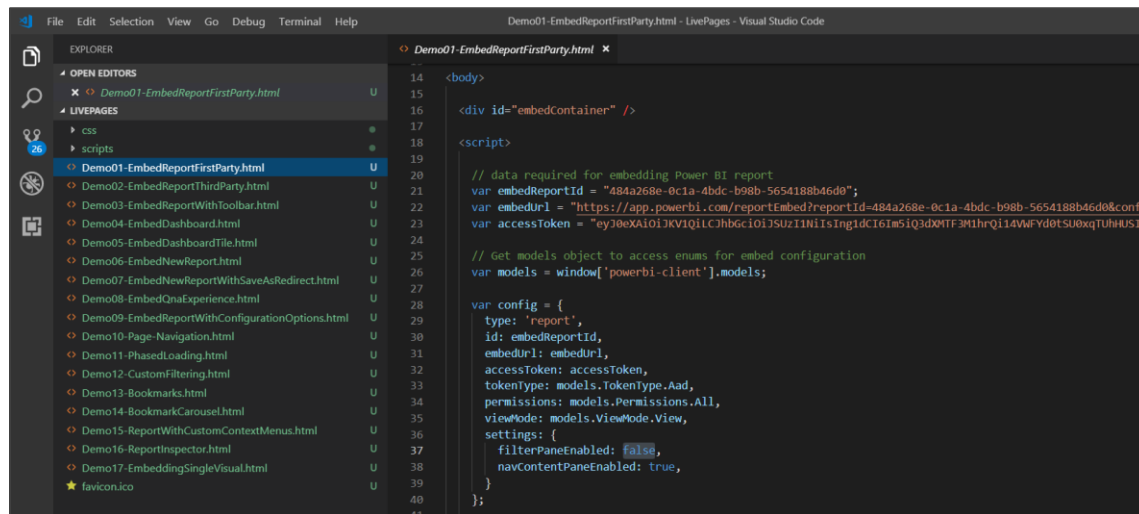
    f) Run the the **PowerBiEmbeddedSratchpad** program once again by pressing the **{F5}** key.

After running the program, you should see there are several more HTML files in the folder at **C:\Student\LivePages**.
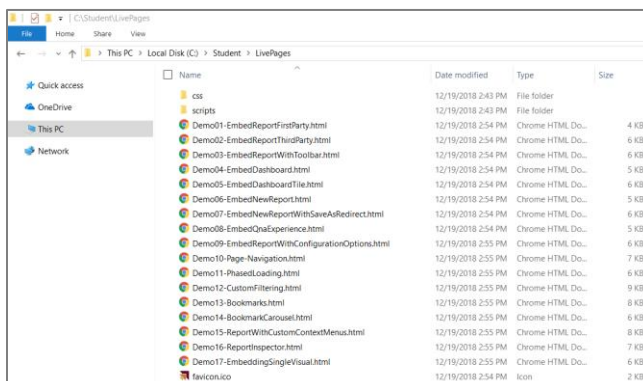
g) Open the folder at **C:\Student\LivePages** using Visual Studio Code so you can open, edit and test the pages inside.

At this point, we will leave it to your imagination to how you experiment and extend the sample JavaScript code in these pages.