Developer Quick Start into Power BI Embedding

The first chapter was dedicated to exploring the theory of Power BI embedding. This brings up an old adage. In theory, there is no difference between theory and practice. But, in practice, there is. To that end, this chapter has been written to get you up and running with Power BI embedding in Visual Studio as quickly as possible. For many developers, there is simply no substitute for the experience of pressing the {F5} key in Visual Studio and seeing how all their theoretical knowledge has been translated into a running, functional application.

This chapter provides lab exercises with step-by-step instructions to create a new Office 365 trial tenant to serve as a Power BI development environment. After that, you will create and test your first Visual Studio project that embeds Power BI resources including reports, dashboards and the Q&A experience. Along the way, you will leverage the Power BI Embedding Onboarding Experience which assists you by registering an Azure application in your Office 365 tenant and by creating an app workspace in Power BI and populating it with content. Once you have prepared your environment with the Power BI Embedding Onboarding Experience, you will then be ready to create a new ASP.NET web application in Visual Studio and to write all the code required to embed reports, dashboard and the Q&A experience on a custom web page.

Exercise 1 - Create an Office 365 Trial Tenant

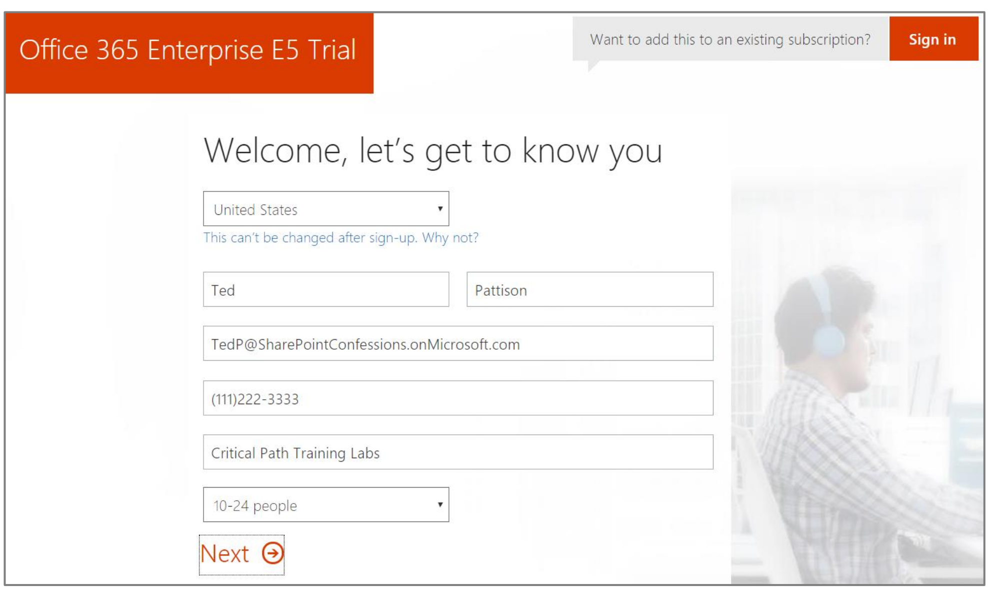
In the first exercise you will create a new Office 365 trial tenant. As you work through the sign up process for this free trial, you will be asked to provide a user name and a password for an Azure AD user account that will be configured as the tenant Global administrator. You will log in with this account when developing and testing applications that use Power BI embedding. However, it's a good practice that you also test your applications with standard user Azure AD accounts that have no administrative permissions. The trial tenant that you are going to create will allow you to create up to 25 user accounts with Office 365 E5 subscriptions. Remember that any user with an Office 365 E5 subscription is automatically assigned a Power BI Pro license as well.

1. Navigate to the Office 365 trial sign up page using an Incognito browser window.
   a) Launch the Chrome browser.
   b) Using the dropdown menu in the upper right, select the command to open a **New incognito window**.
   c) Copy and paste the following URL into the address bar of the incognito window to navigate to the sign up page.

```
https://go.microsoft.com/fwlink/p/?LinkID=698279&culture=en-US&country=US
```

It's not always necessary to sign up for an Office 365 trial account using an incognito window. However, most errors that occur when attempting to sign up for a new trial account are caused by user-specific browser settings such as cached credentials from another Office 365 account. The solution to overcoming errors when signing up for an Office 365 trial account is using an incognito window.

2. Fill out the form with your personal information and click **Next**.

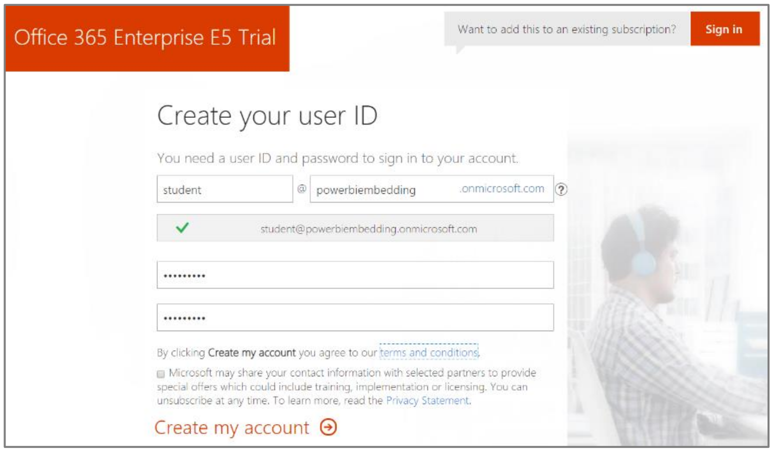


The information you provide on the next page of the signup process will be used to name your new Office 365 tenant.

© Critical Path Training. 2018. All Rights Reserved

www.CriticalPathTraining.com

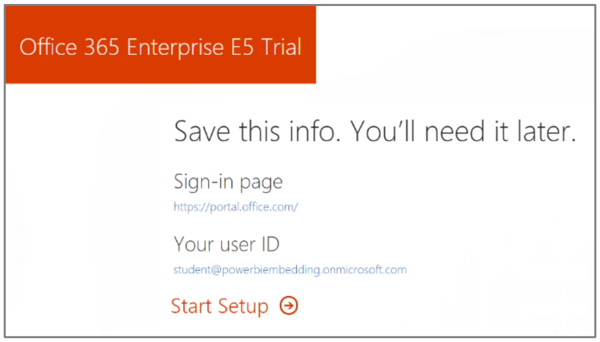
3. On the **Create your user ID** page…
   a) Enter a user name
   b) Enter a unique company name *(you might have to try a few before you get one that's unique)*
   c) Enter a password that you will remember.



Note that the company name you enter on this page will be used to create the domain name for your new Office 365 trial tenant. For example, if you were to enter a company name of **powerbiembedding**, it would result in the creation of a new Office 365 tenant within a domain of **powerbiembedding.onMicrosoft.com**. The user name you enter will be used to create the first user account which will be given administrative rights within the Office trial tenant. If you enter a user name of **Student**, then the email address as well as user principal name for this account will be **student@powerbiembedding.onMicrosoft.com**.

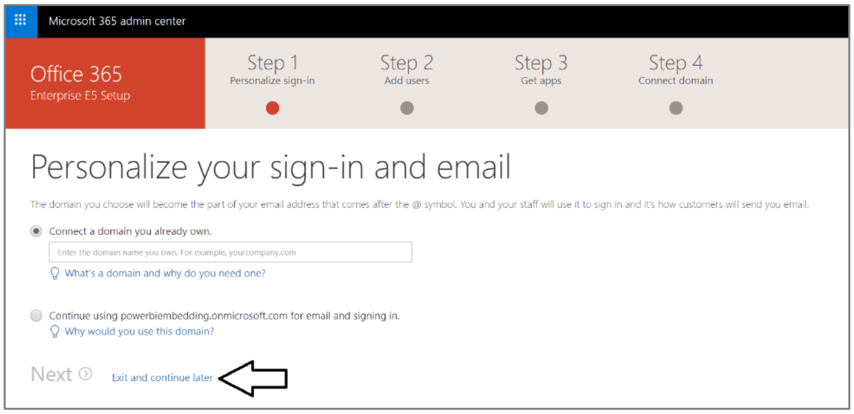
4. Click **Next** to continue to step 3.
5. Complete the validation form in step 3 by proving you are not a robot.
   a) Select the **Text me** option and provide the number of your mobile phone.
   b) When you go through this process, a Microsoft service will send you a text message that contains an access code.
   c) You retrieve the access code form your mobile device and use it to complete the validation process.

6. Once you have completed the validation process, click the **You're ready to go…** link to navigate to the portal welcome page for your new Office 365 trial tenant. Note that you should already be logged on using the user account that was created during the signup process.



© Critical Path Training. 2018. All Rights Reserved

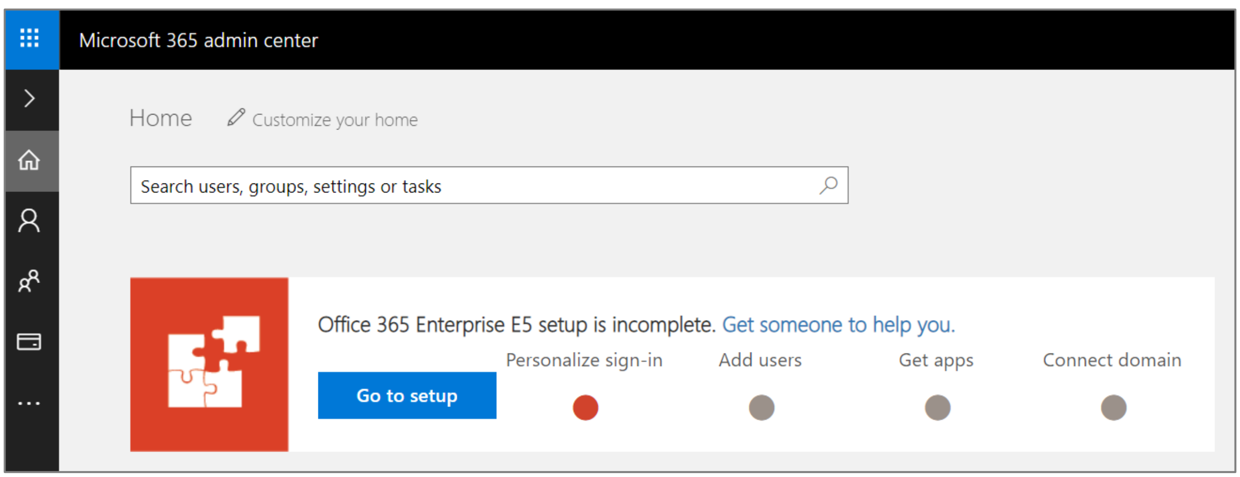
7. If you are prompted with the **Personalize your sign-in and email**, click the **Exit and continue later** link at the bottom of the page.



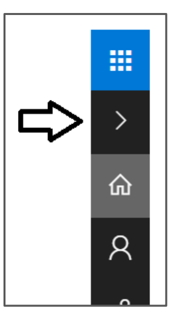
At this point, you have already created your new Office 365 tenant which can support creating up to 25 user accounts with Office 365 Enterprise E5 trial licenses. Note that some Office 365 services within your new Office 365 tenant such as the Microsoft 365 admin center, PowerApps, Flow and Power BI can be accessed immediately. Other services in your Office 365 tenant such as SharePoint Online, OneDrive for Business and Outlook will not be ready immediately and can take some time to provision.

There is no more need to run the browser in incognito mode anymore because it's only required to get through the signup process. You can now return to using a standard browser window. However, it's always a good thing to check to see who you are logged in as because sometimes the browser may log you on using a different Office 365 account you have instead of your new trial account.

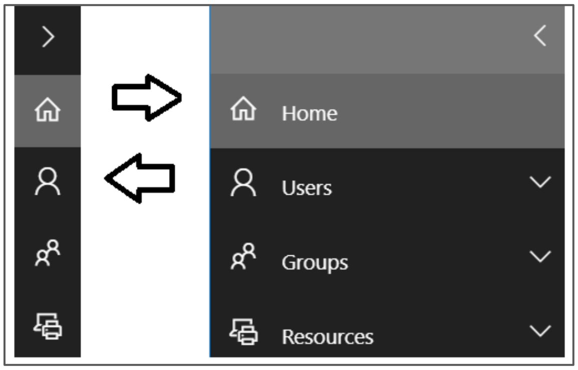
8. At this point, you should be located on the home page of the **Microsoft 365 admin center**.



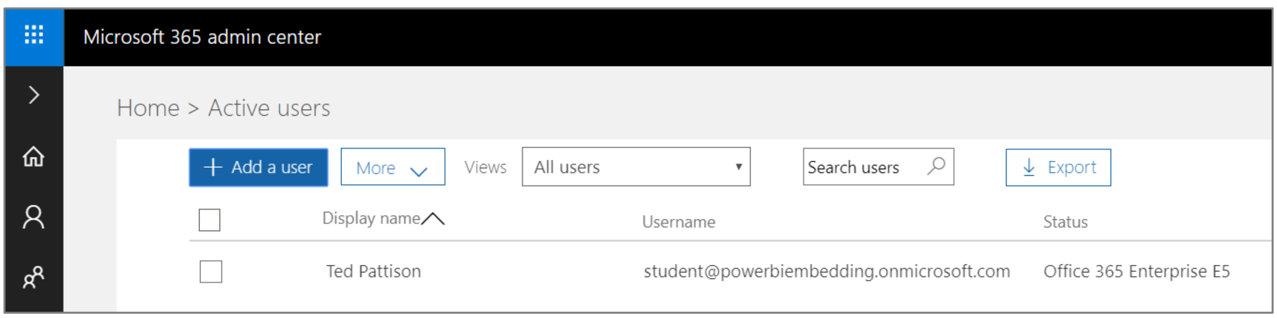
9. Inspect the set of Active Users in the current tenancy.
   a) Locate the top **Menu** button for the left navigation menu. It's the second button from the top with the arrow icon which sits just beneath the Office 365 App Launcher menu button.



b) Click the top **Menu** button several times and see how it toggles the left navigation between a collapsed and expanded mode.

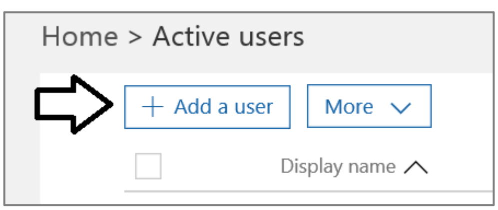


c) Once the **Active Users** page is displayed, you should be able to verify that the user account you are currently logged on as is the only user account that exists in the current tenancy. Remember that this account has been set up as a Global Administrator to the tenant because it is the account that was used when creating the tenant.



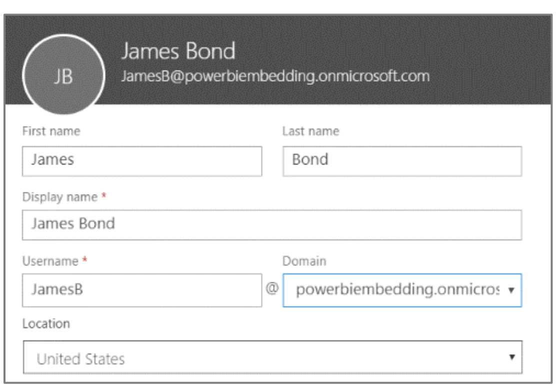
10. Create a new user account.

a) On the **Active Users** page, click the button **Add a user** button to create a new user account



.

b) Fill in the **Create new user account** form with information for a new user account. When creating this account, you can use any name you would like. These lab instructions will demonstrate this by creating a user account for a person named **James Bond** with a user name and email of **JamesB@powerbiembedding.onmicrosoft.com**.



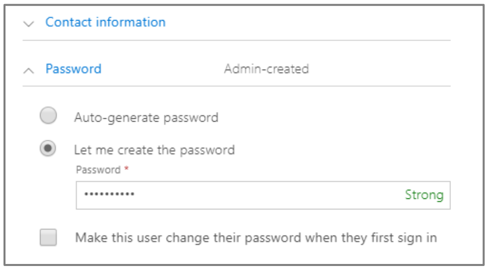
© Critical Path Training. 2018. All Rights Reserved

c) Expand **Password** section under **Contact Information** section.

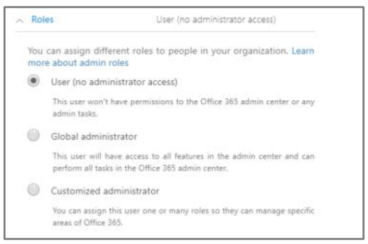
i) Select the option for **Let me create the password**.

ii) Enter a password of **pass@word1** into the textboxes labeled **Password** and **Retype Password**.

iii) Uncheck the checkbox for the option labeled **Make this user change their password when they first sign in**.

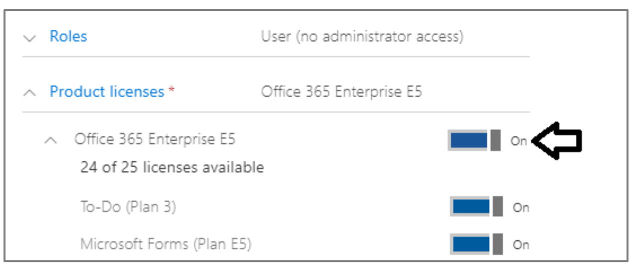


d) Expand the roles section. You do not need to change anything in this section, although you should note that this new user account will be created as a standard user account without any administrator access or privileges.



Note that the new account is usually assigned a trial license for **Office 365 Enterprise E5** plan. However, it's a good practice to check and make sure the new user has been assigned a license for **Office 365 Enterprise E5** which includes the **Power BI Pro** license.

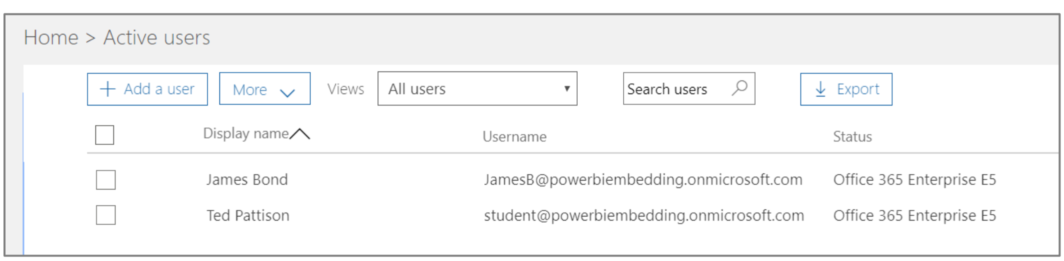
e) In the **Product licenses** section, make sure the **Office 365 Enterprise E5** license is set to **On**..



f) Click the **Save** button at the bottom of the new user form to create the new user account.

g) When you see the **User was added** message, click **Send email and close** to dismiss the **Add new user** task pane.

h) Verify that the new user account has been created and is displayed along with your primary user account.



Now you have a secondary user account that does not have any administrative permissions. It's important that you test applications which use first-party embedding with standard user accounts to ensure your application doesn't require users with special permissions.

© Critical Path Training. 2018. All Rights Reserved

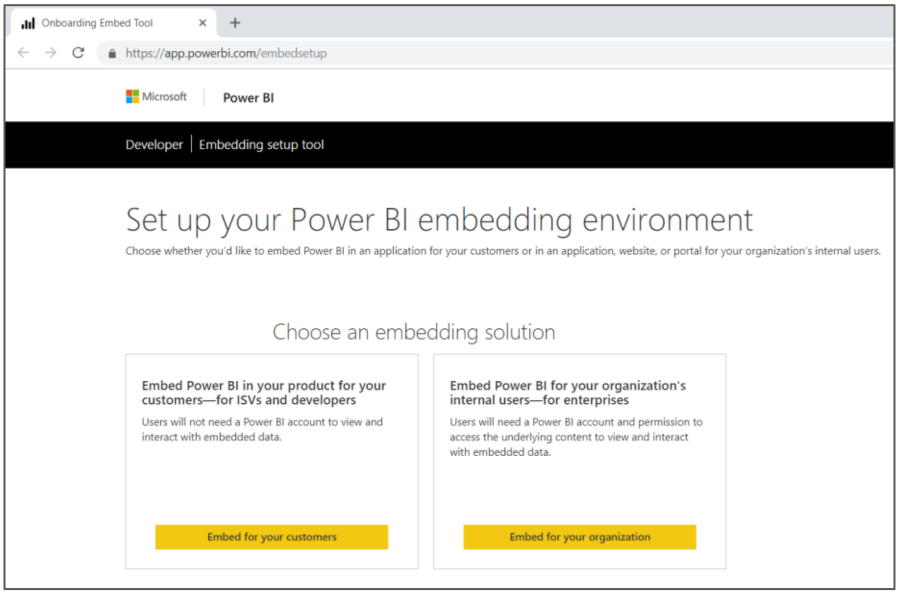
Exercise 2 - Use the Power BI Embedding Onboarding Experience

In this section you will use the Power BI Onboard Experience to get up and running with Power BI embedding as soon as possible.

1. Launch the Power BI Embedding Onboarding Experience to set up for third-party embedding and the app-owns-data model.
   a) Using the browser, navigate to the following URL.

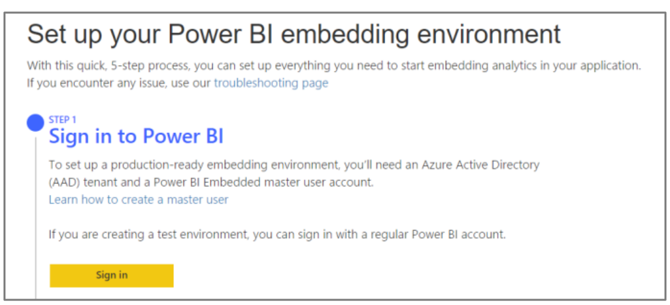
   ```
   https://app.powerbi.com/embedsetup
   ```

   b) You should now see the home page for the Power BI Embedding Onboarding Experience shown in the following screenshot.

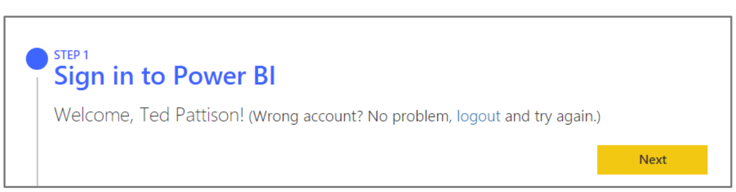


There are two button on the home page for the Power BI Embedding Onboarding Experience. You can click the **Embed for your customers** button to set up for an application using third-party embedding and the app-owns-data model. The other button with the caption **Embed for your organization** is used to set up for an application using first-party embedding and the user-owns-data model.

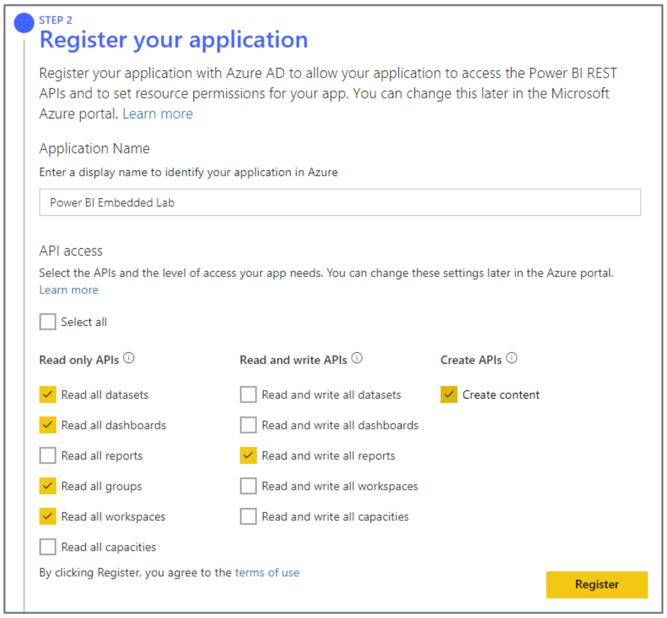
   c) Press the **Embed for your customers** button to begin the set up for an application that uses third-party embedding.
   d) On the **Set up your Power BI embedding environment** page, click the **Sign in** button and sign in with your primary Office 365 user account that is in the role of Global administrator.



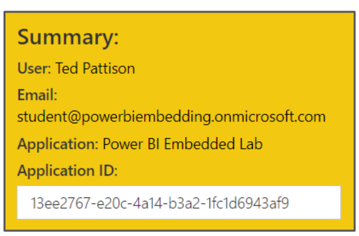
   e) Once you have signed in, click the **Next** button to continue.



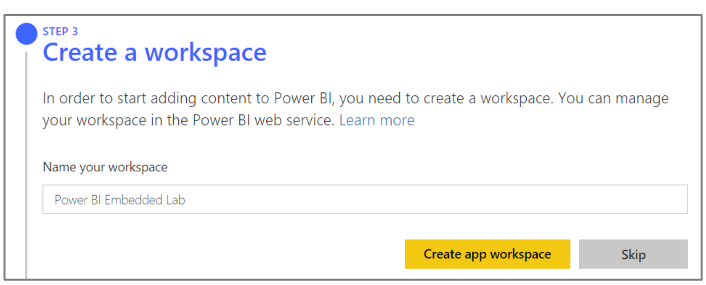
2. In the **STEP 2 - Register your application** section, complete the following steps.
   a) Enter an **Application Name** of **Power BI Embedded Lab**.
   b) Select the **Read all datasets** permission.
   c) Select the **Read all dashboards** permission.
   d) Select the **Read all groups** permission.
   e) Select the **Read all workspaces** permission.
   f) Select the **Read and write all reports** permission.
   g) Select the **Create content** permission.
   h) Click the **Register** button to continue with the registration process.



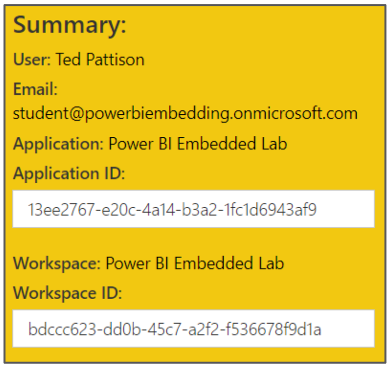
   i) If you examine the **Summary** section to the right, you should see the value for a new **Application ID**.



3. In the **STEP 3 - Create a workspace** section, complete the following steps.
   a) Enter **Power BI Embedded Lab** in the **Name your workspace** textbox.
   b) Click the **Create app workspace** button.



c) Now the **Summary** section should display a second GUID for the new **Workspace ID**.



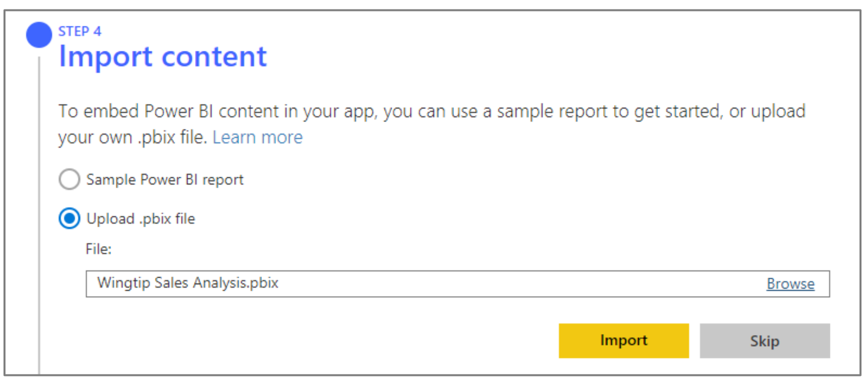
In the next step of the Onboarding Experience, you will be required to upload a Power BI Desktop project file with a **.pbix** extension to populate the new app workspace with a new dataset and report. Before that, you must download the sample PBIX project file named **Wingtip Sales Analysis.pbix** so you have a PBIX file to upload with the Power BI Embedding Onboarding Experience.

Open a new browser tab and download the Power BI Desktop file named **Wingtip Sales Analysis.pbix** using the following URL:

https://github.com/CriticalPathTraining/DPBIE/raw/master/PBIX/Wingtip%20Sales%20Analysis.pbix

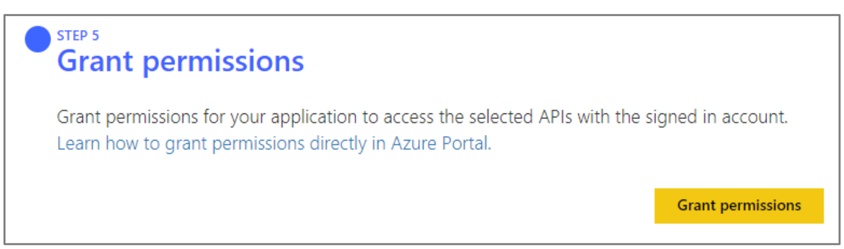
Once you have downloaded a local copy of **Wingtip Sales Analysis.pbix** you can continue to the next step.

4. In the **STEP 4 - Import Content** section, complete the following steps.
   a) Select the option to Upload **.pbix** file.
   b) Click the **Browse** button and select the file named **Wingtip Sales Analysis.pbix** that you just downloaded.
   c) Click the **Import** button to upload **Wingtip Sales Analysis.pbix** to your new app workspace.



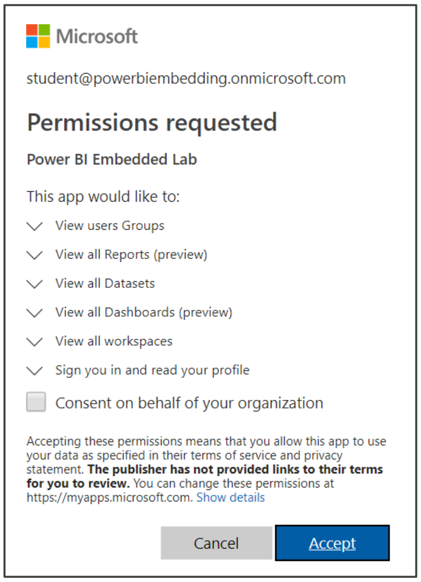
When you upload the Power BI Desktop project named **Wingtip Sales Analysis.pbix**, the Onboard Experience will automatically publish the .pbix project file which will add a new dataset and report to the app workspace. Keep in mind that this process will not automatically create a dashboard. In the next exercise, you will create a dashboard by hand using the Power BI portal.

5. In **STEP 5 - Grant Permissions**, complete the following steps.
   a) Click the **Grant Permissions** button to begin the consent process.



© Critical Path Training. 2018. All Rights Reserved
www.CriticalPathTraining.com

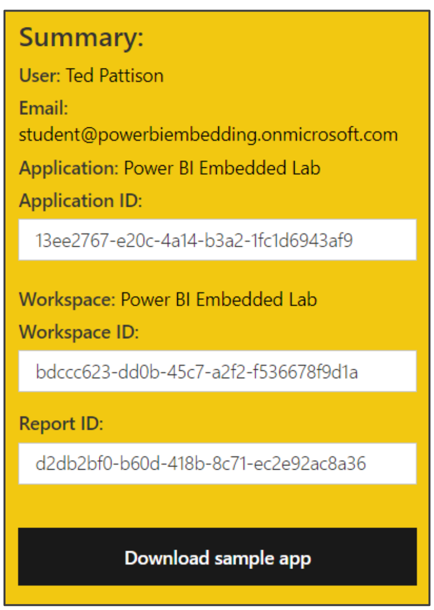
b) When prompted by the **Permissions requested** dialog, click the **Accept** button to consent to the requested permissions.



You can see that the **Permissions requested** dialog contains a **Consent on behalf of your organization** checkbox. Selecting this option would allow you to consent for all the users in your organization at once. This option is unnecessary in this lab because you only need to consent for the current user. You will learn more about how consent works in the next chapter.

c) Once you've consented to the requested permissions, you will see a message indicating success.

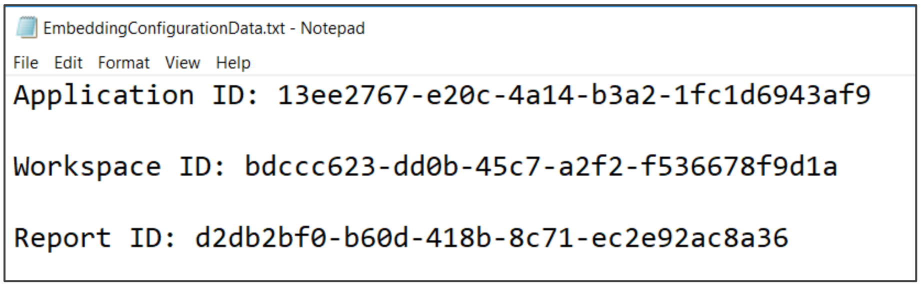
d) Note that the **Summary** section now contains three GUIDs for the **Application ID**, the **Workspace ID** and the **Report ID**.



While you have completed the Onboarding Experience, you still need to record the **Application ID**, **Workspace ID** and **Report ID** for your application. Therefore, you should leave the Onboarding Experience page with the summary data open through the next step.

© Critical Path Training. 2018. All Rights Reserved

6. Record the configuration data for the **Application ID**, **Workspace ID** and **Report ID** in a text file for use later in this lab.
   a) Open Notepad or another text editor of your choice.
   b) Create a simple text file to record the **Application ID**, **Workspace ID** and **Report ID** as shown in the following screenshot.

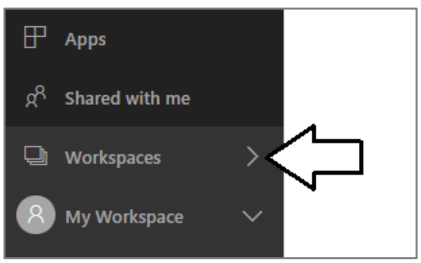

EmbeddingConfigurationData.txt - Notepad
File Edit Format View Help

```
Application ID: 13ee2767-e20c-4a14-b3a2-1fc1d6943af9

Workspace ID: bdccc623-dd0b-45c7-a2f2-f536678f9d1a

Report ID: d2db2bf0-b60d-418b-8c71-ec2e92ac8a36
```

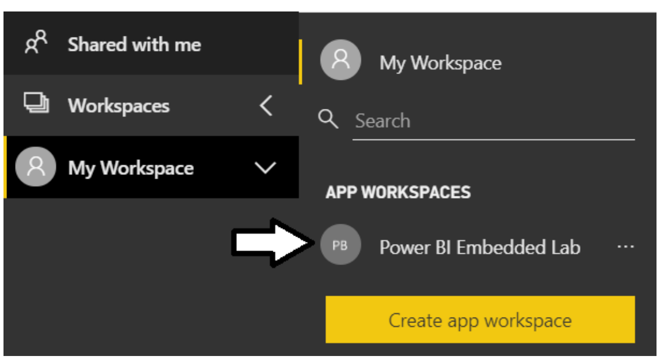
   c) Save the file as **EmbeddingConfigurationData.txt**.

Leave the file **EmbeddingConfigurationData.txt** open because you will be adding more configuration data over the next few steps.

7. Inspect the app workspace named **Power BI Embedded Lab** created by the onboarding experience.
   a) Open a new browser tab and navigate to the Power BI portal at **https://app.powerbi.com**.
   b) Click the **Workspaces** flyout menu in the left navigation.



   c) Using the flyout menu, click **Power BI Embedded Lab** to navigate to the new app workspace which has just been created.

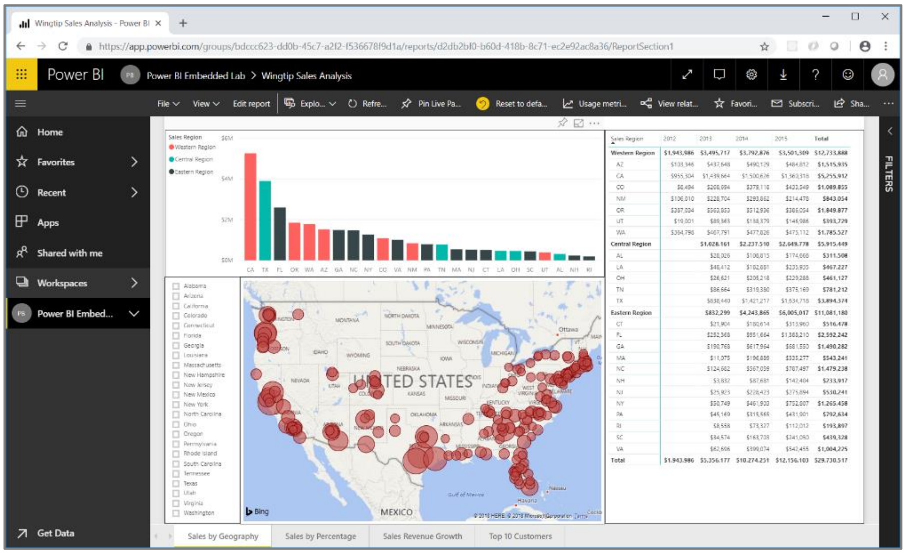


   d) Click on the **Datasets** tab to view the datasets in the app workspace.

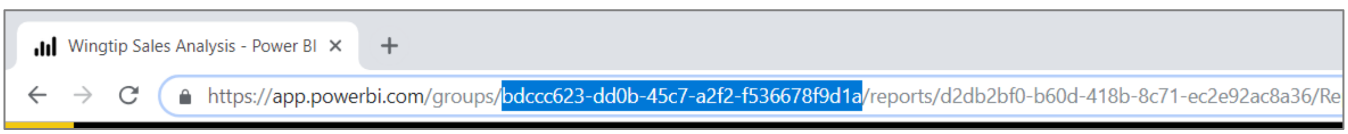
Currently, there should be a single dataset named **Wingtip Sales Analysis**.

e) Click on the **Reports** tab to view the reports in the app workspace.

f) Click on the report named **Wingtip Sales Analysis** to open it and view its contents.



g) Inspect the address bar of the browser to see how the web URL for the report contains the workspace ID right after **groups/**.



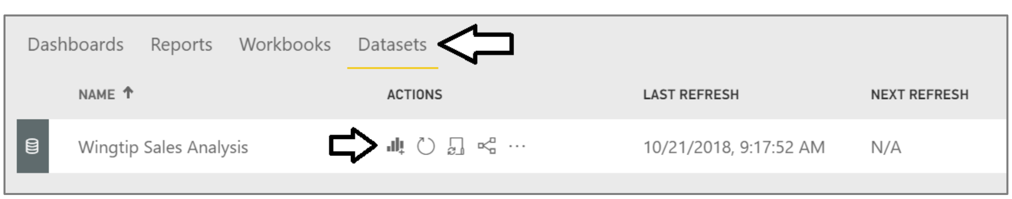
h) You should also be able to see how the web URL for the report contains the report ID right after **reports/**.

You should be able to verify that the workspace ID and the report ID in the address bar match the workspace ID and the report ID that you added to **EmbeddingConfigurationData.txt**. You can also see that the web URL for a Power BI report is built using this format.

`https://app.powerbi.com/groups/{app-workspace-id}/reports/{report-id}/ReportSection1`

8. Determine the dataset ID for the dataset named **Wingtip Sales Analysis**.

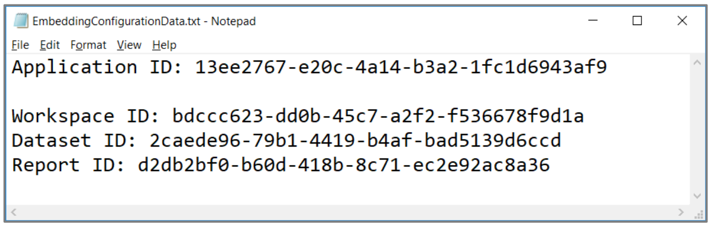
a) Navigate to the **Datasets** tab and then click the **New Report** button for the dataset named **Wingtip Sales Analysis**.



© Critical Path Training. 2018. All Rights Reserved

b) Locate the dataset ID from the URL of the new report that comes after **datasets/**.

c) Copy the dataset ID into **EmbeddingConfigurationData.txt** as shown in the following screenshot.



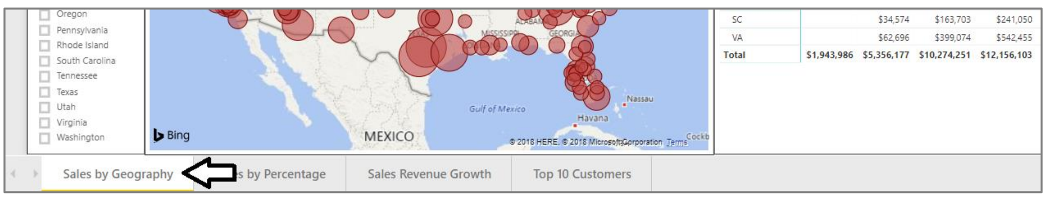

```
Application ID: 13ee2767-e20c-4a14-b3a2-1fc1d6943af9

Workspace ID: bdccc623-dd0b-45c7-a2f2-f536678f9d1a
Dataset ID: 2caede96-79b1-4419-b4af-bad5139d6ccd
Report ID: d2db2bf0-b60d-418b-8c71-ec2e92ac8a36
```

d) Save your changes to **EmbeddingConfigurationData.txt**.

9. Create a new Dashboard named **Wingtip Sales Analysis**.

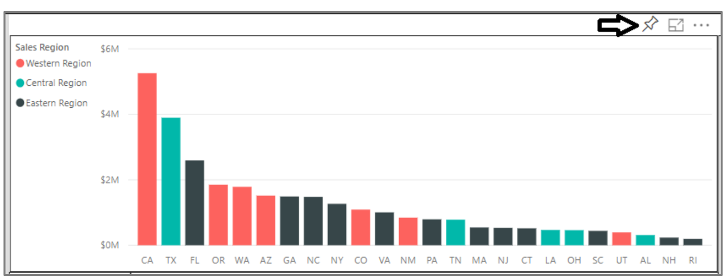
a) Navigate to the **Reports** tab and open the report named **Wingtip Sales Analysis**.

b) Navigate to the **Sales by Geography** page of the **Wingtip Sales Analysis** report.



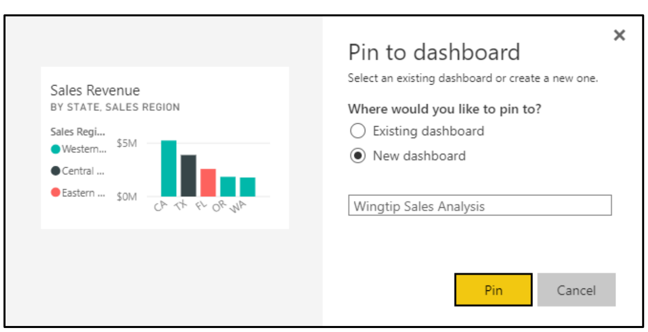
c) Hover the mouse over the column chart visual which displays a sales revenue breakdown across sales regions and states.

d) Locate and click the button with the thumbtack icon to pin this report visual to a new dashboard.



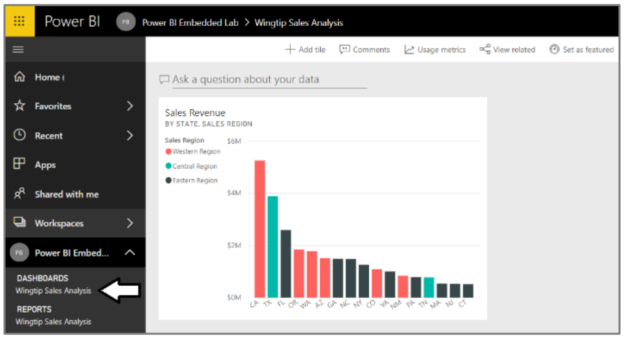
When you click the thumbtack button, you'll be prompted with the **Pin to dashboard** dialog which prompts you to select a dashboard.

e) Select **New Dashboard**, give it a name of **Wingtip Sales Analysis** and click the **Pin** button.

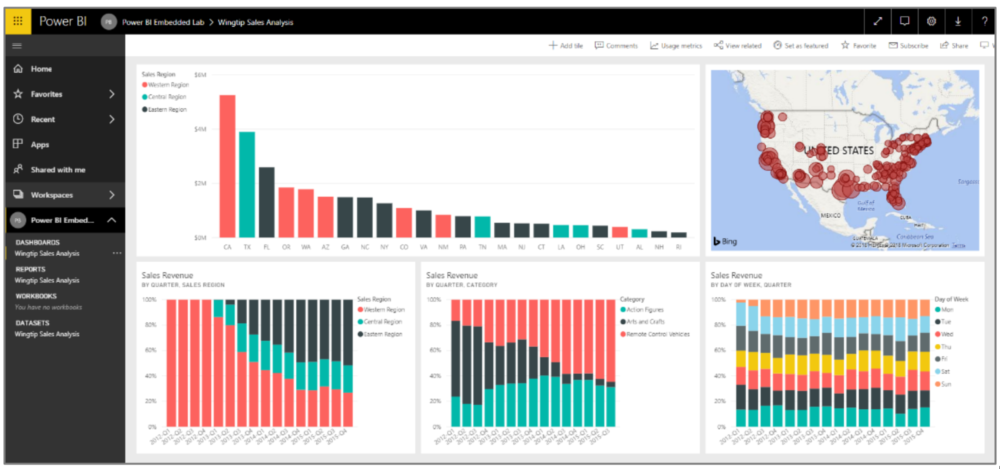


© Critical Path Training. 2018. All Rights Reserved

f) At this point, the **Wingtip Sales Analysis** dashboard should be created and a link to it should appear in the left navigation.



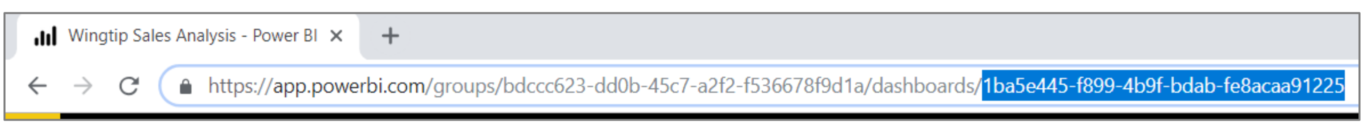
g) Repeat the process of pinning a visual from the report several times to create a few more dashboard tiles. Choose whatever visuals you'd like from the report but make sure your dashboard contains several tiles as shown in the following screenshot.



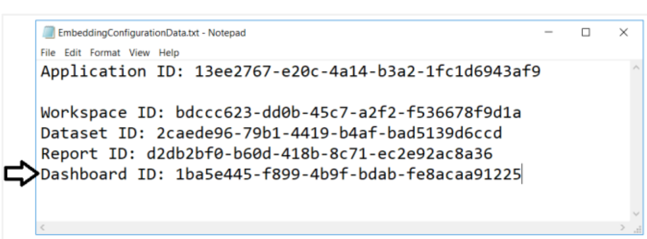
10. Determine the dashboard ID for the new dashboard you just created named **Wingtip Sales Analysis**.

a) With the **Wingtip Sales Analysis** dashboard open, inspect its URL in the address bar of the browser.

b) Locate and copy the dashboard ID from the URL by copy the GUID that comes after **dashboards/**.



c) Copy the dashboard ID into **EmbeddingConfigurationData.txt**.



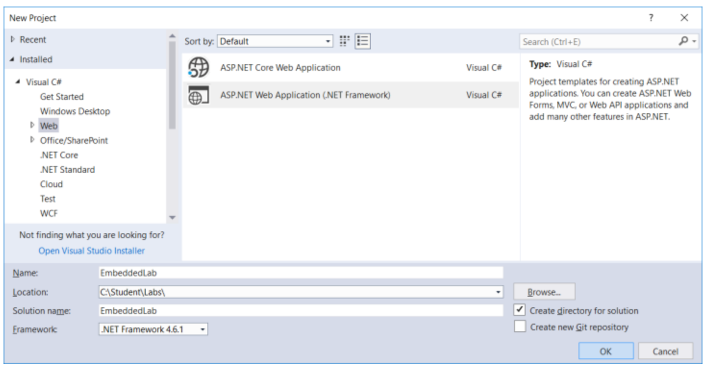
d) Save your changes to **EmbeddingConfigurationData.txt**.

You have populated your app workspace with Power BI resources including a dataset, a report and a dashboard. In the next exercise you will create a new ASP.NET MVC project in Visual Studio to embed these resources on the web pages of a custom application.

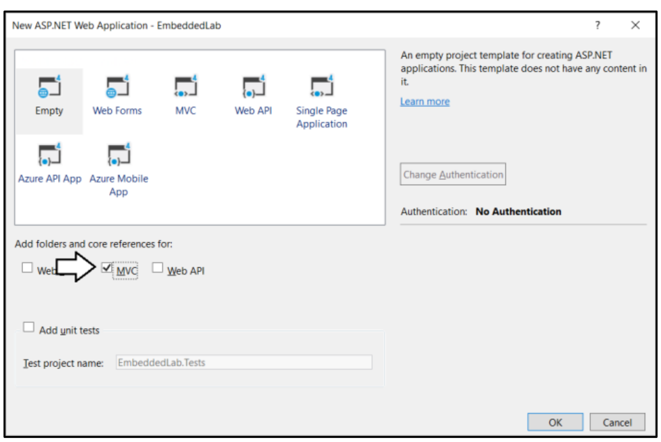
Exercise 3 - Create a ASP.NET MVC Project

In this exercise you will create a new Web Application project using Visual Studio 2017 and the ASP.NET MVC framework.

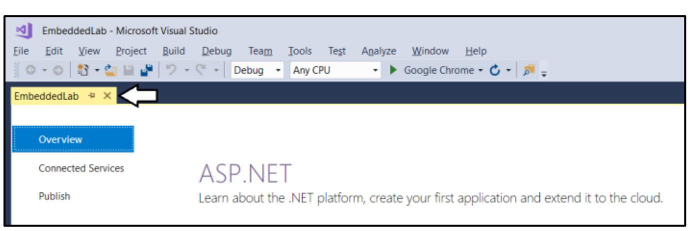
1. Launch **Visual Studio 2017**.
2. Create a new ASP.NET MVC project in Visual Studio 2017.
   a) In Visual Studio select **File > New > Project**.
   b) In the **New Project** dialog:
      i) Select **Installed > Templates > Visual C# > Web**.
      ii) Select the **ASP.NET Web Application** project template.
      iii) Name the new project **EmbeddedLab**.
      iv) Add the new project into a local folder such as **C:\Student\Labs**.
      v) Click **OK** to display the **New ASP.Net Web Application** wizard.



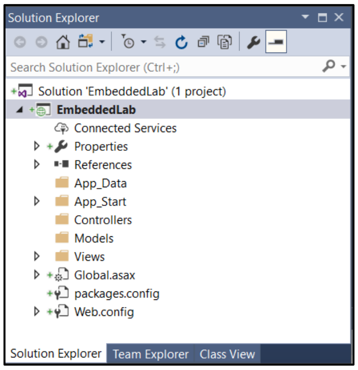
   c) In the **New ASP.Net Web Application** dialog, select the **Empty** template.
   d) In the section with the caption **Add folders and core references**, make sure the **MVC** checkbox is checked.
   e) Click the **OK** button to create the new project.



   f) When Visual Studio finishes creating the project, it displays an information page. Close this page by clicking the **x** in the tab.



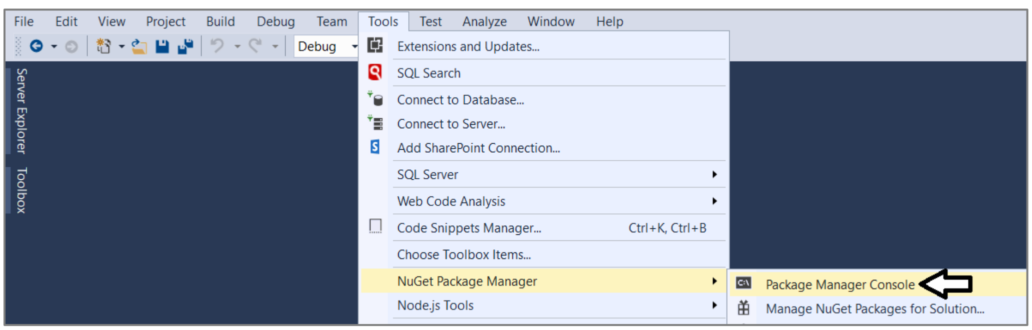
g) Take a minute to familiarize yourself with the structure of the project in the **Solution Explorer**.



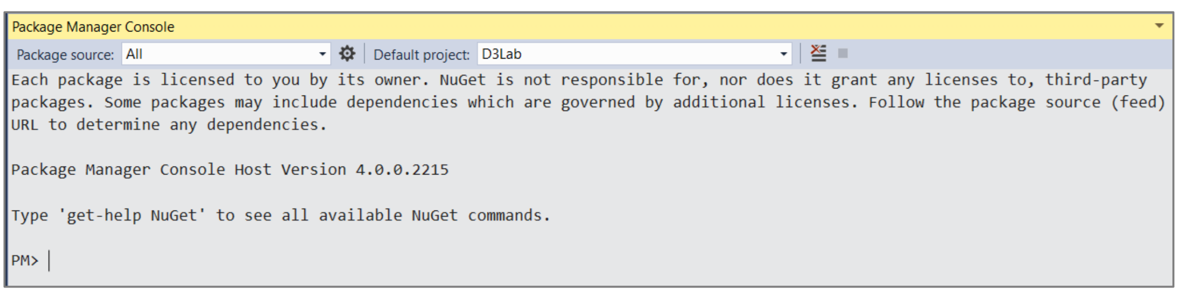
At this point, you have created a new ASP.NET MVC project based on the **Empty** project template. You will need to add an MVC controller and several MVC views before your application provides any type of user interface experience. Before adding a controller or writing any code, you will first update the project's NuGet packages that were automatically included with your new project. You will also prepare for Power BI embedding by adding the NuGet package for the Azure AD Authentication library (ADAL) and the NuGet packages for the Power BI Service API and the Power BI JavaScript API.

3. Configure the **Embedded Lab** project with the required set of NuGet packages

a) From the Visual Studio menu, select the command **Tools > NuGet Package Manager > Package Manager Console**.



b) You should now see the **Package Manage Console** with a **PM>** command prompt as shown in the following screenshot



c) Type in and execute the following command to install the NuGet package for **bootstrap**.

```
Install-Package bootstrap
```

d) Type in and execute the following command to install the NuGet package for **Azure Active Directory Authentication library**.

```
Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory
```

e) Type in and execute the following command to install the NuGet package for the **Power BI Service API**.

```
Install-Package Microsoft.PowerBI.Api
```

f) Type in and execute the following command to install the NuGet package for the **Power BI JavaScript API**.

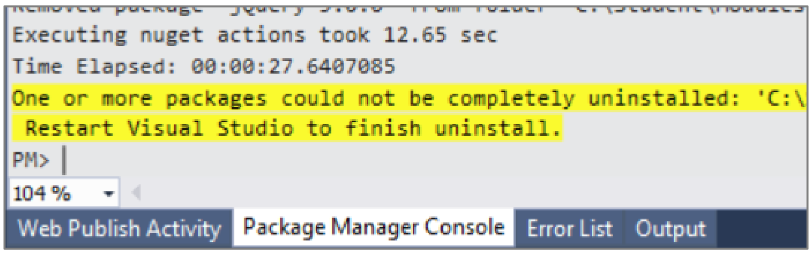
```
Install-Package Microsoft.PowerBI.JavaScript
```

Now that you have installed the required NuGet packages for Power BI embedding, you will now run the **Update-Package** cmdlet to make sure all the packages in your project are updated to the latest versions available in the NuGet repository.

g) Type in and execute the following command to update all NuGet packages in the project to their most current version.

```
Update-Package
```

h) The first time you run the **Update-Package** cmdlet, you will be prompted to restart Visual Studio to complete the update.



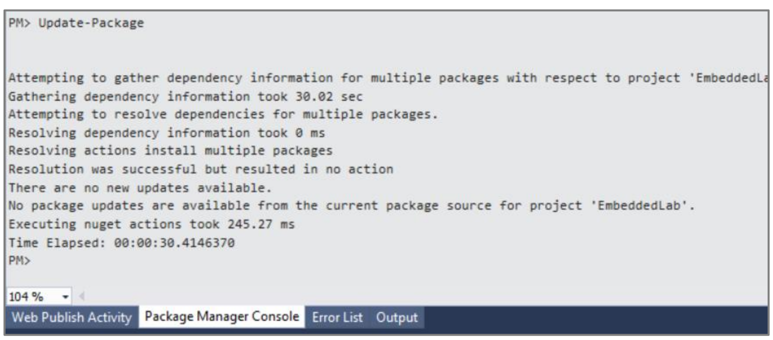
i) Restart Visual Studio and open the **EmbeddedLab** project.

j) Open the Package Manager Console window if it is not already open.

k) Execute the **Update-Package** cmdlet one more time to ensure all NuGet packages are updated to their most current version.

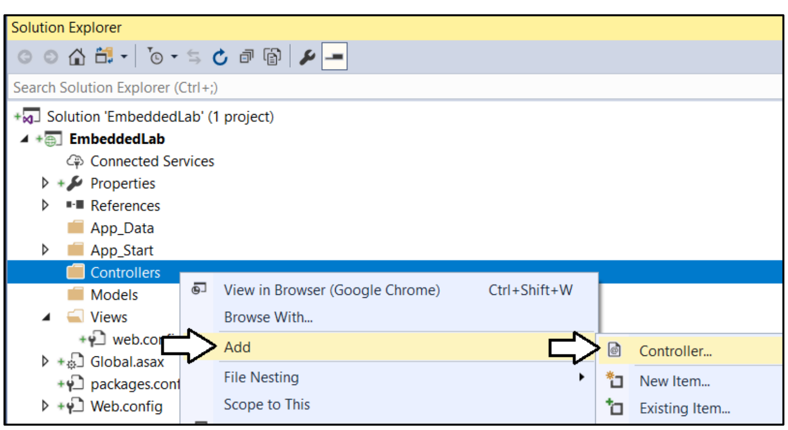
```
Update-Package
```

l) You should now see an output message in the Package Manager Console indicating "There are no new updates available".

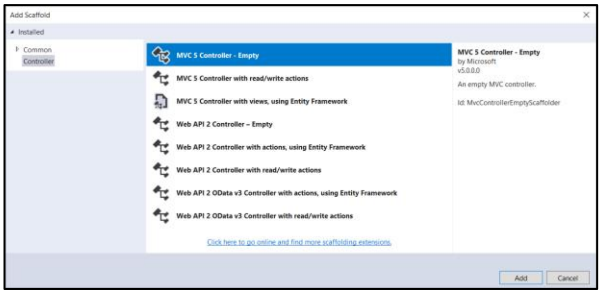


4. Add the **HomeController** class.

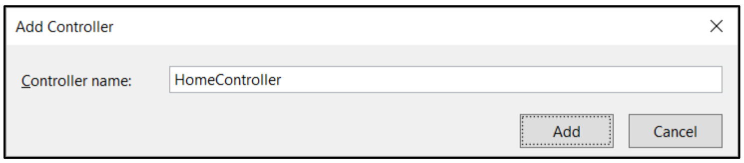
a) In Solution Explorer, right-click on the **Controllers** folder.



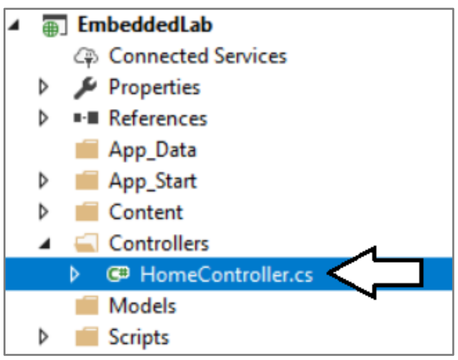
b) In the **Add Scaffold** dialog, select the first option **MVC 5 Controller – Empty** and then click **Add**.



c) In the **Add Controller** dialog, enter a **Controller name** of **HomeController** and then click **Add**.



d) You should now see a new source file in the **Controllers** folder named **HomeController.cs**.

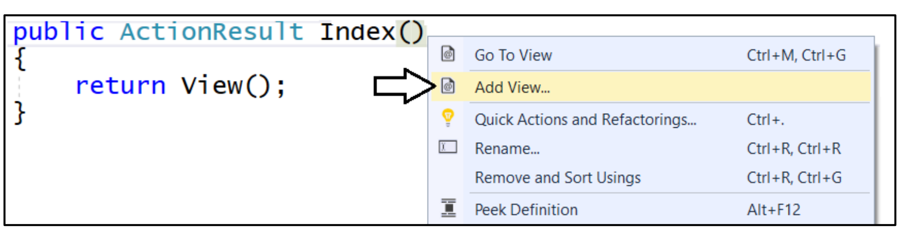


e) Inside **HomeController.cs**, you will find the starting point for the **HomeController** class with a single method named **Index**.

```
namespace EmbeddedLab.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

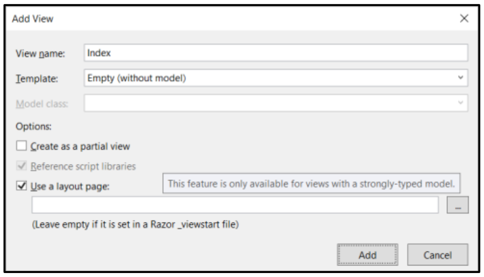
5. Add a view for the **Index** action method of the **Home** controller class.

a) Inside **HomeController.cs**, right-click the **Index** method and select the **Add View…** command.



© Critical Path Training. 2018. All Rights Reserved
www.CriticalPathTraining.com

b) In the **Add View** dialog, accept all the default setting as shown in the following screenshot and click **Add**.

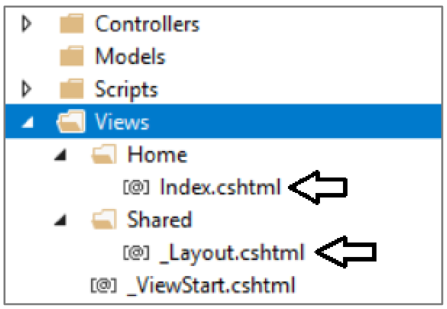


When you create a new view and leave the **Use a layout page** option selected, a new shared layout page named **_Layouts.cshtml** is automatically added to the project in the **Views/Shared** folder.

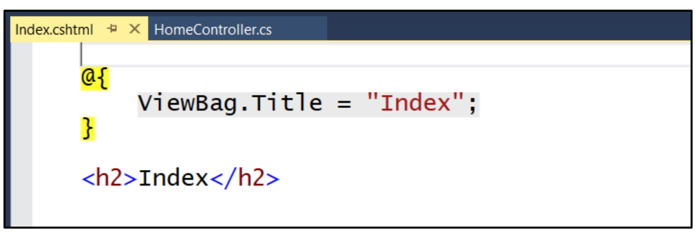
c) In Solution Explorer, you should be able to verify that your project contains two new files.

i) Inside the **Views/Home** folder there is a new razor view file named **Index.cshtml**.

ii) Inside the **Views/Shared** folder there is a new shared layout page named **_Layouts.cshtml**.



d) Examine the code that has been added to **Index.cshtml**.



e) Delete all the code inside **Index.cshtml** and replace it with the following HTML code.

```
<div id="homePageContainer" class="container" >

  <div class="jumbotron">
    <h2>Power BI Embedded Lab</h2>
  </div>

</div>
```

f) Save your changes and close **Index.cshtml**.

Over the next few steps, you will add the HTML code for a shared layouts page into **_Layout.cshtml** in a sequence of several different copy-and-paste operations. If you'd rather copy and paste the all the code for **_Layout.cshtml** at once, you can find the completed HTML code inside a file named **Layout.cshtml.txt** located in the **C:\Student\Modules\08_PBIEmbedded\Lab\Snippets** folder.

6. Modify the shared layouts page named **_Layouts.cshtml**.

a) In Solution Explorer, expand the **Views** folder and then expand the **Shared** folder.

b) Double-click on **_Layouts.cshtml** to open it in an editor window.

© Critical Path Training. 2018. All Rights Reserved

www.CriticalPathTraining.com

c) Delete the entire contents of **_Layouts.cshtml** and replace with the following HTML starter page.

```
<!DOCTYPE html>
<html>

<head>
</head>

<body>
</body>

</html>
```

d) Copy and paste the following HTML code to provide the **head** section

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Embedded Lab</title>
  <link href="~/Content/bootstrap.css" rel="stylesheet" />
  <link href="~/Content/Site.css" rel="stylesheet" />
  <script src="~/Scripts/jquery-3.3.1.js"></script>
</head>
```

e) Make sure your script link to jQuery matches the version number of the jQuery library source file in the **Scripts** folder.

f) Copy and paste the following HTML code to provide the **body** section of the page.

```
<body>

  <!-- Add Banner with TopNav and Toolbar Here -->

  <!-- Add Main Body Content Here -->

  <!-- Add JavaScript Code to Resize Page Elements Here -->

</body>
```

Now you will copy and paste HTML markup code into each of the three sections in the HTML **body** element.

g) Copy and paste the following code into the body just below the ***Add Banner with TopNav and Toolbar Here*** comment.

```
<!-- Add Banner with TopNav and Toolbar Here -->
<div id="banner" class="container">
  <nav id="topnav" class="navbar navbar-expand-sm navbar-dark bg-dark">
    <ul class="navbar-nav">
      <li class="nav-item active">
        @Html.ActionLink("Embedded Lab", "Index", "Home",
                          routeValues: null, htmlAttributes: new { @class = "nav-link navbar-brand" })
      </li>
      <li class="nav-item">
        @Html.ActionLink("Report", "Report", "Home",
                         routeValues: null, htmlAttributes: new { @class = "nav-link" })
      </li>
      <li class="nav-item">
        @Html.ActionLink("Dashboard", "Dashboard", "Home",
                         routeValues: null, htmlAttributes: new { @class = "nav-link" })
      </li>
      <li class="nav-item">
        @Html.ActionLink("Q&A", "Qna", "Home",
                         routeValues: null, htmlAttributes: new { @class = "nav-link" })
      </li>
      <li class="nav-item">
        @Html.ActionLink("New Report", "NewReport", "Home",
                         routeValues: null, htmlAttributes: new { @class = "nav-link" })
      </li>
    </ul>
  </nav>
  @RenderSection("toolbar", required: false)
</div>
```

h) Copy and paste the following code into the body just below the **Add Main Body Content Here** comment

```
<!-- Add Main Body Content Here -->
<div id="content-box" class="container body-content">
  @RenderBody()
</div>
```

i) Copy and paste the following code into the body just below the **Add Main Body Content Here** comment

```
<!-- Add JavaScript Code to Resize Page Elements -->
<script>
  $(function () {
    var heightBuffer = 12;
    var newHeight = $(window).height() - ($("#banner").height() + heightBuffer);
    $("#content-box").height(newHeight);
    $("#embedContainer").height(newHeight);
    $(window).resize(function () {
      var newHeight = $(window).height() - ($("#banner").height() + heightBuffer);
      $("#content-box").height(newHeight);
      $("#embedContainer").height(newHeight);
    });
  });
</script>
```

j) Save your changes and close **_Layouts.cshtml**.

7. Modify the **Content\Sites.css** file with a set of custom CSS styles.

a) In Solution Explorer, expand the **Content** folder and then double-click on **Sites.css** open it in an editor window.

b) Delete all the existing content from **Sites.css**

c) Copy the following code with CCS styles and copy it into **Sites.css**.

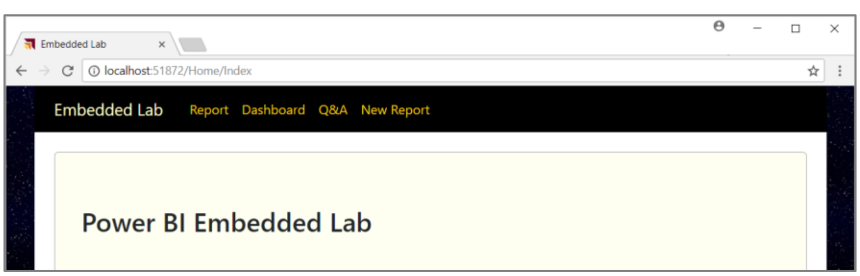
```
body { background-color: black; }

#homePageContainer { padding: 24px; }
#content-box { padding: 0px; background-color: #FFF; }
#banner { padding-left: 0px; padding-right: 0px; }
#topnav { background-color: #000 !important; }
#toolbar { background-color: #666 !important; border-bottom: 1px solid #333; }
#toolbar button { width: 92px; margin: 4px; padding: 2px; font-size: 9px; font-weight: bold;
                  color: #222; background-color: #F2C811; border-color: #111; }

.navbar-dark .navbar-nav .active > .nav-link { color: #FFC; }
.navbar-dark .navbar-nav .nav-link { color: #F2C811; }
.navbar-dark .navbar-nav .nav-link:hover { color: orange; }
.jumbotron { background-color: #FFFFF1; border: 1px solid #AAA; }
```

d) Save your changes and close **Sites.css**.

8. Test out the **EmbeddedLab** project using the Visual Studio Debugger

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

b) When the project starts, the home page should load in the browser and match the following screenshot.



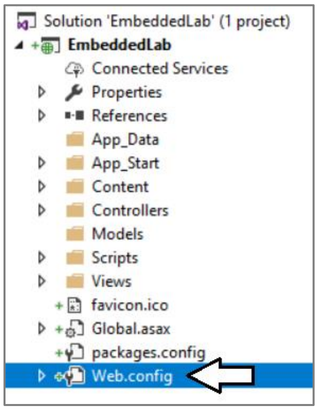
c) Close the browser, return to Visual Studio and stop the debugger.

Note that the navigation links on the top navigation menu are not working yet. Over the next few exercises, you will add MVC action methods and razor views to implement Power BI embedding behavior behind each of these navigation links.

Exercise 4 - Embed a Power BI Report

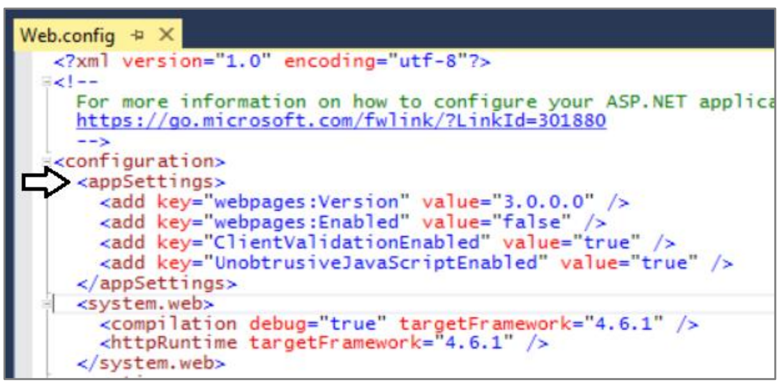
In this exercise, you will continue to modify your ASP.NET MVC project to add support for embedding a Power BI report.

1. Modify the project's **web.config** file to add **appSetting** values for the required configuration data.
   a) Open the **web.config** file located at the root of the **EmbeddedLab** project.

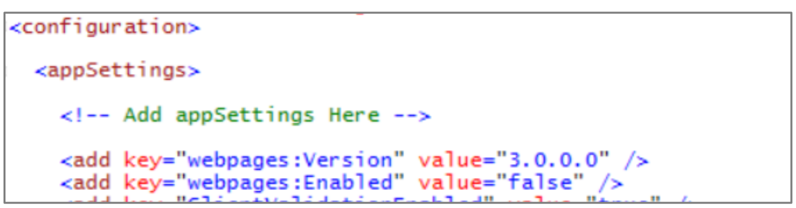


Make sure you open the **web.config** file located at the root of the project and not the **web.config** file inside the **Views** folder.

   b) Locate the **<appSettings>** element at the top of **web.config**.



   c) Add a few blank lines just after the **<appSettings>** element opening tag.



   d) Copy and paste the following XML code into the **web.config** file underneath the **<appSettings>** opening tag.

```
<add key="application-id" value="" />

<add key="app-workspace-id" value="" />
<add key="dataset-id" value="" />
<add key="report-id" value="" />
<add key="dashboard-id" value=" " />

<add key="aad-account-name" value="" />
<add key="aad-account-password" value=" " />
```

© Critical Path Training. 2018. All Rights Reserved

e) Copy configuration values from **EmbeddingConfigurationData.txt** into the new **appSetting** values in **web.config.**

You should be able to supply values for the first 5 **appSettings** values from **EmbeddingConfigurationData.txt**.

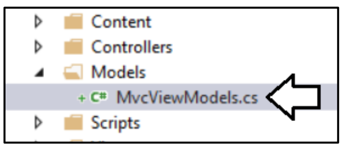
f) For the **appSettings** named **aad-account-name** and **aad-account-password**, enter the user name and password for the primary Office 365 account you have been using to access your Office 365 trial tenant.

```
<add key="aad-account-name" value="student@powerbiembedding.onmicrosoft.com" />
<add key="aad-account-password" value="pass@word1" />
```

g) Save your changes and close **web.config**.

2. Create classes to provide MVC view models for Power BI Embedding data.

a) Add a new C# source file named **MvcViewModels.cs** inside the **Models** folder.



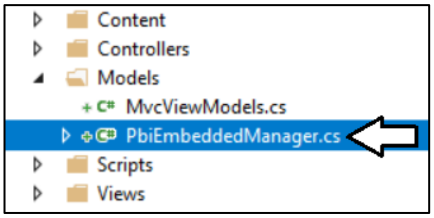
b) If there is any code inside **MvcViewModels.cs**, delete it and replace it with the following code.

```
namespace EmbeddedLab.Models {

  // data required for embedding a report
  public class ReportEmbeddingData {
    public string reportId;
    public string reportName;
    public string embedUrl;
    public string accessToken;
  }

  // data required for embedding a new report
  public class NewReportEmbeddingData {
    public string workspaceId;
    public string datasetId;
    public string embedUrl;
    public string accessToken;
  }

  // data required for embedding a dashboard
  public class DashboardEmbeddingData {
    public string dashboardId;
    public string dashboardName;
    public string embedUrl;
    public string accessToken;
  }

  // data required for embedding a dashboard
  public class QnaEmbeddingData {
    public string datasetId;
    public string embedUrl;.
    public string accessToken;
  }

}
```

c) Save your changes and close **MvcViewModels.cs**.

© Critical Path Training. 2018. All Rights Reserved

3. Create the **PbiEmbeddedManager** class.
   a) Add a new class named **PbiEmbeddedManager** inside the **Models** folder.
   b) The **Models** folder should now contain a C# source file named **PbiEmbeddedManager.cs**.



   c) Delete any code inside **PbiEmbeddedManager.cs** and replace it with the following starter code.

```
using System;
using System.Configuration;
using System.Threading.Tasks;
using Microsoft.Rest;
using Microsoft.PowerBI.Api.V2;
using Microsoft.PowerBI.Api.V2.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;

namespace EmbeddedLab.Models {

  public class PbiEmbeddedManager {
  }

}
```

   d) Modify the **PbiEmbeddedManager** class by adding the following set of static fields.

```
public class PbiEmbeddedManager {

  private static string aadAuthorizationEndpoint = "https://login.microsoftonline.com/common";
  private static string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";
  private static string urlPowerBiRestApiRoot = "https://api.powerbi.com/";

  private static string applicationId = ConfigurationManager.AppSettings["application-id"];

  private static string workspaceId = ConfigurationManager.AppSettings["app-workspace-id"];
  private static string datasetId = ConfigurationManager.AppSettings["dataset-id"];
  private static string reportId = ConfigurationManager.AppSettings["report-id"];
  private static string dashboardId = ConfigurationManager.AppSettings["dashboard-id"];

  private static string userName = ConfigurationManager.AppSettings["aad-account-name"];
  private static string userPassword = ConfigurationManager.AppSettings["aad-account-password"];

}
```

In addition to fields for the seven configuration values, there are other fields named **aadAuthorizationEndpoint**, **resourceUriPowerBi** and **urlPowerBiRestApiRoot** which are used when authenticating with Azure AD and calling to the Power BI Service API.

   e) At the bottom of **PbiEmbeddedManager** class, add a new method named **GetAccessToken** using the following code.

```
private static string GetAccessToken() {

  AuthenticationContext authenticationContext = new AuthenticationContext(aadAuthorizationEndpoint);

  AuthenticationResult userAuthnResult =
    authenticationContext.AcquireTokenAsync(
      resourceUriPowerBi,
      applicationId,
      new UserPasswordCredential(userName, userPassword)).Result;

  return userAuthnResult.AccessToken;
}
```

f) Underneath the **GetAccessToken** method, add a new method named **GetPowerBiClient** using the following code.

```
private static PowerBIClient GetPowerBiClient() {
  var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");
  return new PowerBIClient(new Uri(urlPowerBiRestApiRoot), tokenCredentials);
}
```

You have implemented the essential behavior in the **PbiEmbeddedManager** class to authenticate with Azure AD and to create new **PowerBIClient** objects which represents the top-level entry point into the Power BI Service API. Now you are at a point where you can add methods to the **PbiEmbeddedManager** class which call into the Power BI Service API to retrieve embedding data.

4. Add the **GetReportEmbeddingData** method to the **PbiEmbeddedManager** class.

a) At the bottom of the **PbiEmbeddedManager** class, add a method named **GetReportEmbeddingData** with the following code.

```
public static async Task<ReportEmbeddingData> GetReportEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = await pbiClient.Reports.GetReportInGroupAsync(workspaceId, reportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");
  string embedToken =
        (await pbiClient.Reports.GenerateTokenInGroupAsync(workspaceId,
                                                           report.Id,
                                                           generateTokenRequestParameters)).Token;

  return new ReportEmbeddingData {
    reportId = reportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

b) Save your changes to **PbiEmbeddedManger.cs**.

Now that you have added the **GetReportEmbeddingData** method, you will create a new action method that calls this method.

5. Add the **Report** action method to the **HomeController** class.

a) Inside the **Controllers** folder, open the C# source file named **HomeController.cs**.

b) Update the set of **using** statements at the top of **HomeController.cs** using the following code.

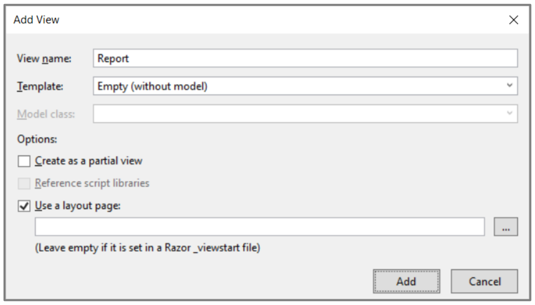
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using EmbeddedLab.Models;
```

c) Underneath the **Index** method, add a new asynchronous action method named **Report** using the following code.

```
public class HomeController : Controller {

  public ActionResult Index() {
    return View();
  }

  public async Task<ActionResult> Report() {
    ReportEmbeddingData embeddingData = await PbiEmbeddedManager.GetReportEmbeddingData();
    return View(embeddingData);
  }

}
```

d) Right-click on the **Report** method and select the **Add View…** command from the context menu.

e) In the **Add View** dialog, accept all the default settings and click the **Add** button.



f) You should be able to verify that a new razor view file named **Report.cshtml** has been created in the **Views/Home** folder.

g) Delete all existing content from **Report.cshtml** and replace it with the following code.

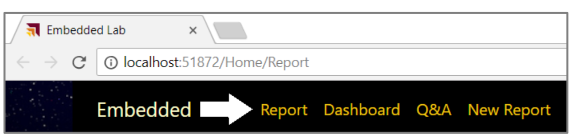
```
@model EmbeddedLab.Models.ReportEmbeddingData

<div id="embedContainer" />

<script src="~/Scripts/powerbi.js"></script>

<script>

  // data required for embedding Power BI report
  var embedReportId = "@Model.reportId";
  var embedUrl = "@Model.embedUrl";
  var accessToken = "@Model.accessToken";

  // Get models object to access enums for embed configuration
  var models = window['powerbi-client'].models;

  var config = {
    type: 'report',
    id: embedReportId,
    embedUrl: embedUrl,
    accessToken: accessToken,
    permissions: models.Permissions.All,
    tokenType: models.TokenType.Embed,
    viewMode: models.ViewMode.View,
    settings: {
      filterPaneEnabled: false,
      navContentPaneEnabled: true,
    }
  };

  // Get a reference to HTML element that will be embed container
  var reportContainer = document.getElementById('embedContainer');

  // Embed the report and display it within the div container.
  var report = powerbi.embed(reportContainer, config);

</script>
```

h) Save your changes and close **Report.cshtml**.

6. Test out the application by running it in the Visual Studio debugger.

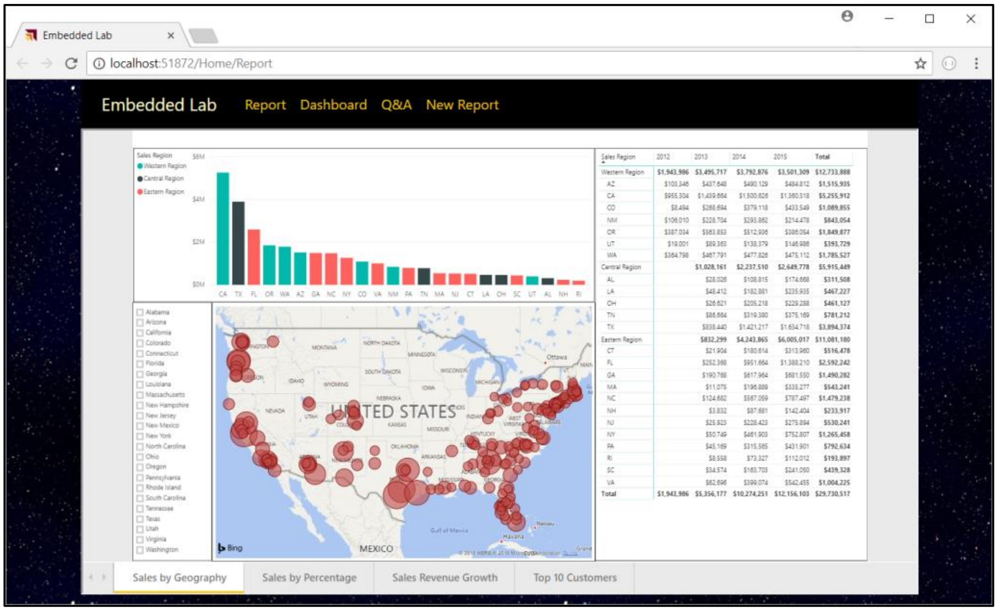
a) Press the **{F5}** key in Visual Studio to begin a new debugging session.

b) Click the **Report** link in the top navigation menu.



If the editor window with a razor view such as **Report.cshtml** is the active window when you press the **{F5}**, the Visual Studio debugger will automatically take you to this view at the start of your debugging session.

c) You should now see the report has been embedded on the web page.



Try resizing the browser window. You will see that your application responds by dynamically changing the size of the HTML element with the ID of **embedContainer** and the embedded report responds automatically by changing its size to fit the new dimensions.

d) Close the browser window and return to Visual Studio and stop the current debugging session.

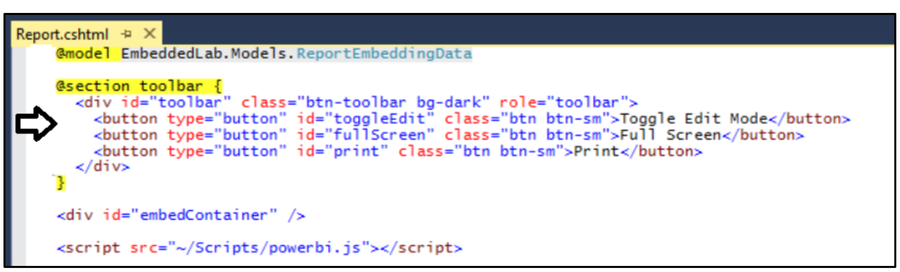
Exercise 5 - Add an Interactive Toolbar for an Embedded Report

In this exercise, you continue to work on **Report.cshtml** by adding a new toolbar with three command buttons. You will also add JavaScript code behind these command buttons to invoke actions on the embedded report.

1. Add the HTML layout code for a toolbar into **Report.cshtml**.
   a) Open **Report.cshtml** if it is not already open.
   b) Copy and paste the following HTML code into **Report.cshtml** just below the **@model** directive at the top.

```
@section toolbar {
  <div id="toolbar" class="btn-toolbar bg-dark" role="toolbar" >
    <button type="button" id="toggleEdit" class="btn btn-sm" >Toggle Edit Mode</button>
    <button type="button" id="fullScreen" class="btn btn-sm" >Full Screen</button>
    <button type="button" id="print" class="btn btn-sm" >Print</button>
  </div>
}
```

   c) The top of **Report.cshtml** should match the following screenshot.



© Critical Path Training. 2018. All Rights Reserved

d) Inside **Report.cshtml**, move down inside **<script>** block and add a few new lines after the line which calls **powerbi.embed**.

e) Copy and paste the following JavaScript code at the bottom of the **<script>** block just before the close **</script>** tag.

```
var viewMode = "view";

$("#toggleEdit").click(function () {
  // toggle between view and edit mode
  viewMode = (viewMode == "view") ? "edit" : "view";
  report.switchMode(viewMode);
  // show filter pane when entering edit mode
  var showFilterPane = (viewMode == "edit");
  report.updateSettings({
    "filterPaneEnabled": showFilterPane
  });
});

$("#fullScreen").click(function () {
  report.fullscreen();
});

$("#print").click(function () {
  report.print();
});
```

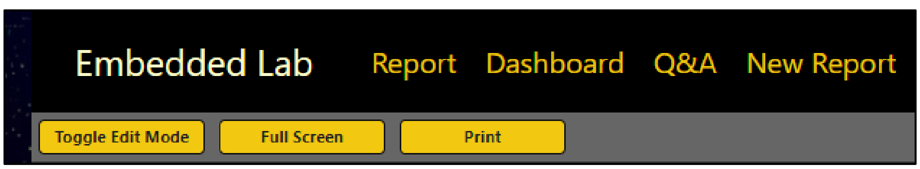
f) Save your changes to **Report.cshtml**.

2. Test out the application by running it in the Visual Studio debugger.

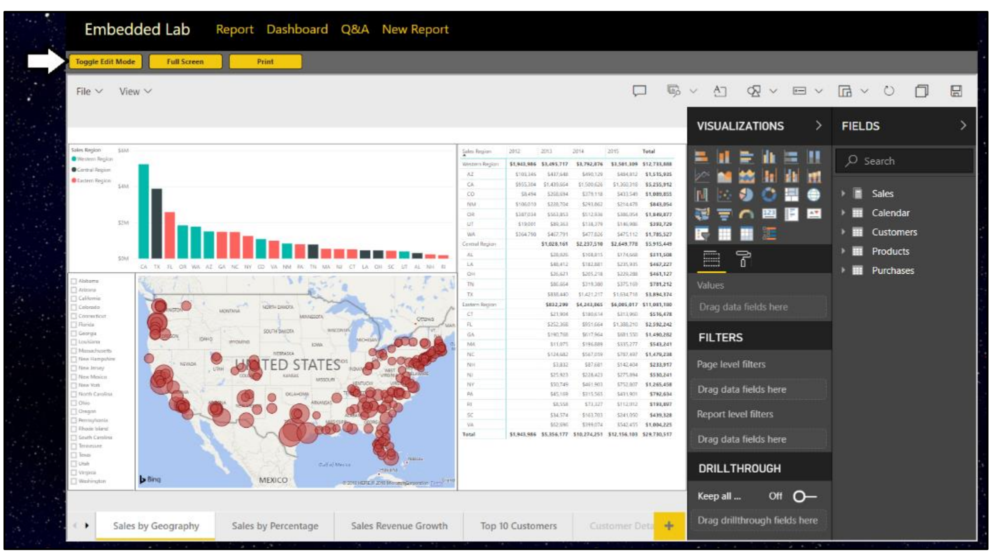
a) Press the **{F5}** key in Visual Studio to begin a new debugging session.

b) Click the **Report** link in the top navigation menu.

c) You should now see three toolbar buttons with the captions **Toggle Edit Mode**, **Full Screen** and **Print**.



d) Click the **Toggle Edit Mode** button several times. The report should toggle back and forth between edit and reader mode.



e) Experiment by clicking the **Full Screen** button.

f) Experiment by clicking the **Print button**.

g) Close the browser window and return to Visual Studio and stop the current debugging session.

© Critical Path Training. 2018. All Rights Reserved

Exercise 6 - Embed a Dashboard

In this exercise you will embed a dashboard. As you will see, it's not very different from the steps you have already implemented to embed a report.

1. Add a new method to the **PbiEmbeddingManger** class named **GetDashboardEmbeddingData**.
   a) Open **PbiEmbeddedManager.cs** in an editor window if it's not already open.
   b) Navigate to the bottom of the class definition just beneath the **GetReportEmbeddingData** method
   c) Paste in the definition for a new method named **GetDashboardEmbeddingData** using the following code.

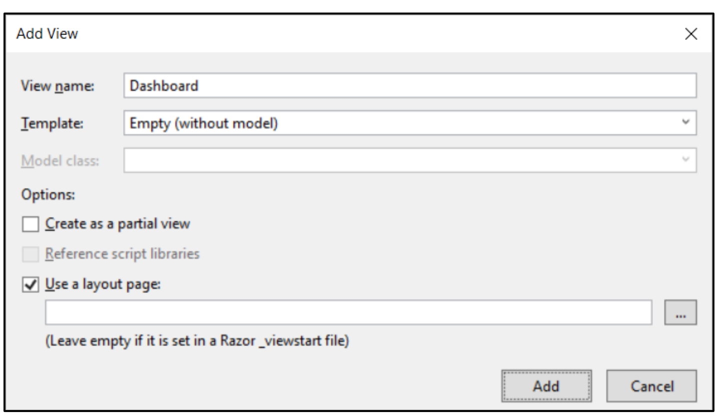
```
public static async Task<DashboardEmbeddingData> GetDashboardEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var dashboard = await pbiClient.Dashboards.GetDashboardInGroupAsync(workspaceId, dashboardId);
  var embedUrl = dashboard.EmbedUrl;
  var dashboardDisplayName = dashboard.DisplayName;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");

  string embedToken =
     (await pbiClient.Dashboards.GenerateTokenInGroupAsync(workspaceId,
                                                           dashboardId,
                                                           generateTokenRequestParameters)).Token;

  return new DashboardEmbeddingData {
    dashboardId = dashboardId,
    dashboardName = dashboardDisplayName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

   d) Save your changes to **PbiEmbeddedManager.cs**.

2. Add a new action method to the **HomeController** class named **Dashboard**.
   a) Open **HomeController.cs** in an editor window if it's not already open.
   b) Add a new action method named **Dashboard** just beneath the **Report** method using the following code.

```
public async Task<ActionResult> Dashboard() {
  DashboardEmbeddingData embeddingData = await PbiEmbeddedManager.GetDashboardEmbeddingData();
  return View(embeddingData);
}
```

3. Create a razor view for the **Dashboard** action method.
   a) Right-click on the **Dashboard** action method and select the **Add View…** command from the context menu.
   b) In the **Add View** dialog, accept all the default settings and click the **Add** button.



c) You should see that a new razor view file named **Dashboard.cshtml** has been created in the **Views/Home** folder.

d) Delete any existing code inside **Dashboard.cshtml** and replace it with the following HTML code.

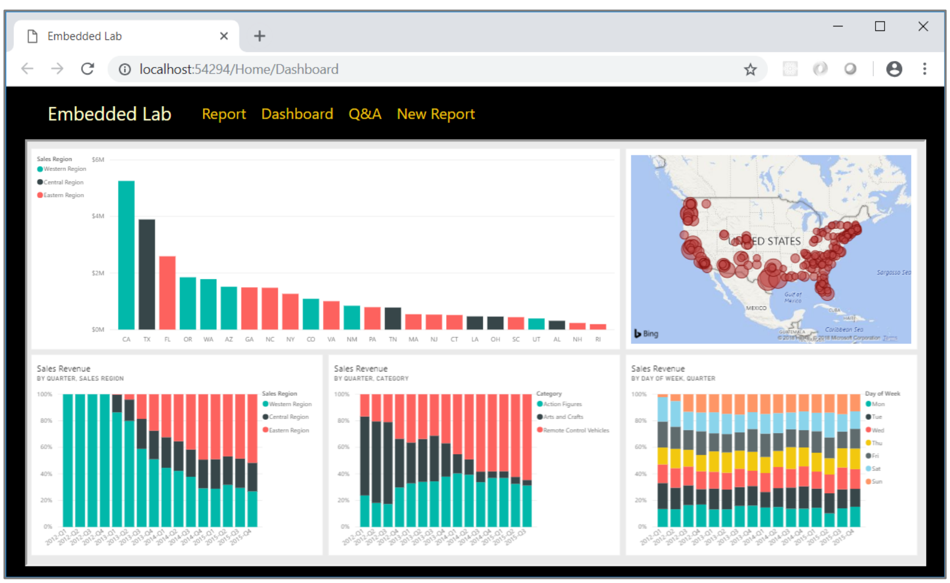
```
@model EmbeddedLab.Models.DashboardEmbeddingData

<div id="embedContainer" />

<script src="~/Scripts/powerbi.js"></script>
<script>

  // data required for embedding Power BI report
  var embedDashboardId = "@Model.dashboardId";
  var embedUrl = "@Model.embedUrl";
  var accessToken = "@Model.accessToken";

  // Get models object to access enums for embed configuration
  var models = window['powerbi-client'].models;

  var config = {
    type: 'dashboard',
    id: embedDashboardId,
    embedUrl: embedUrl,
    accessToken: accessToken,
    tokenType: models.TokenType.Embed,
    pageView: "fitToWidth"
  };

  // Get a reference to the embedded report HTML element
  var embedContainer = document.getElementById('embedContainer');

  // Embed the report and display it within the div container.
  var dashboard = powerbi.embed(embedContainer, config);

</script>
```

e) Save your changes to **Dashboard.cshtml**.

4. Test out the application by running it in the Visual Studio debugger.

a) Press the **{F5}** key in Visual Studio to begin a new debugging session.

b) Click the **Dashboard** link in the top navigation menu and you should see the dashboard embedded in the web page.



c) Try changing the size of the browser window and see how the application responds by adjusting the size of the dashboard.

d) Close the browser window and return to Visual Studio and stop the current debugging session.

© Critical Path Training. 2018. All Rights Reserved

Exercise 7 - Embed the Power BI Q&A Experience

In this exercise you will embed the Power BI Q&A experience. To accomplish this, you will be required to provide the dataset ID associated with the dataset on which you want to execute natural language queries.

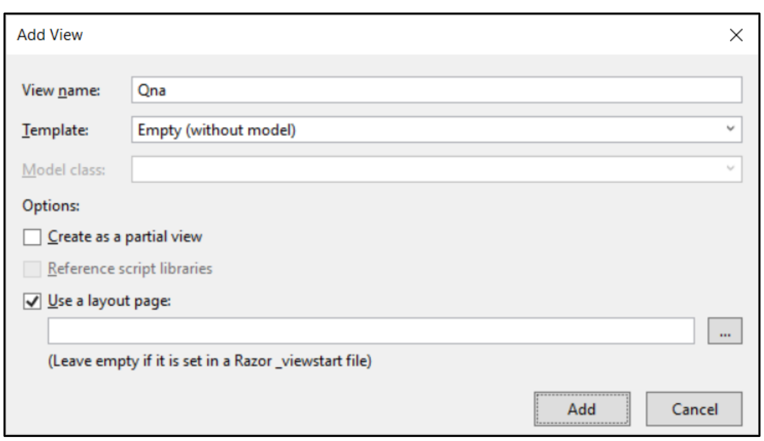
1. Add a new method to the **PbiEmbeddingManger** class named **GetQnaEmbeddingData**.
   a) Open **PbiEmbeddedManager.cs** in an editor if it's not already open.
   b) Navigate to the bottom of the class definition just beneath the **GetDashboardEmbeddingData** method
   c) Add a new method named **GetQnaEmbeddingData** by copying and pasting the following code.

```
public async static Task<QnaEmbeddingData> GetQnaEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var dataset = await pbiClient.Datasets.GetDatasetByIdInGroupAsync(workspaceId, datasetId);

  string embedUrl = "https://app.powerbi.com/qnaEmbed?groupId=" + workspaceId;
  string datasetID = dataset.Id;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");
  string embedToken =
         (await pbiClient.Datasets.GenerateTokenInGroupAsync(workspaceId,
                                                             dataset.Id,
                                                             generateTokenRequestParameters)).Token;

  return new QnaEmbeddingData {
    datasetId = datasetId,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

   d) Save your changes to Open **PbiEmbeddedManager.cs**.
2. Add a new action method to the **HomeController** class named **Qna**.
   a) Open **HomeController.cs** in an editor window if it's not already open.
   b) Add a new action method named **Qna** just beneath the **Dashboard** method using the following code.

```
public async Task<ActionResult> Qna() {
  QnaEmbeddingData embeddingData = await PbiEmbeddedManager.GetQnaEmbeddingData();
  return View(embeddingData);
}
```

3. Create a razor view for the **Qna** action method.
   a) Right-click on the **Qna** action method and select the **Add View…** command from the context menu.
   b) In the **Add View** dialog, accept all the default settings and click the **Add** button.



© Critical Path Training. 2018. All Rights Reserved

c) You should see that a new razor view file named **Qna.cshtml** has been created in the **Views/Home** folder.

d) Delete any existing code inside **Qna.cshtml** and replace it with the following HTML code.

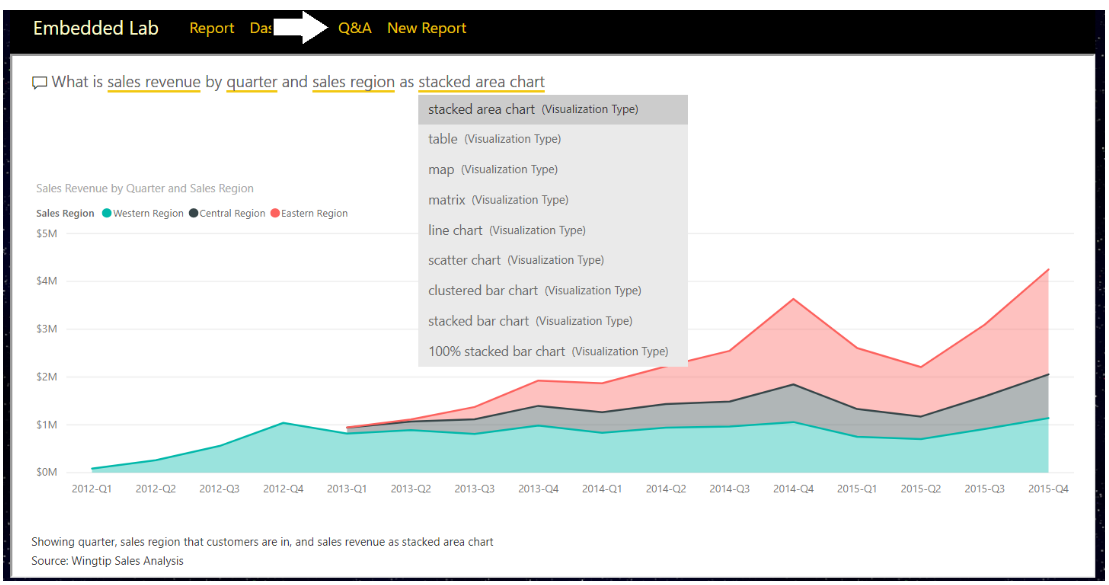
```
@model EmbeddedLab.Models.QnaEmbeddingData

<div id="embedContainer" />

<script src="~/Scripts/powerbi.js"></script>
<script>

  // Get data required for embedding
  var datasetId = "@Model.datasetId";
  var embedUrl = "@Model.embedUrl";
  var accessToken = "@Model.accessToken";

  // Get models object to access enums for embed configuration
  var models = window['powerbi-client'].models;

  var config = {
    type: 'qna',
    tokenType: models.TokenType.Embed,
    accessToken: accessToken,
    embedUrl: embedUrl,
    datasetIds: [datasetId],
    viewMode: models.QnaMode.Interactive,
    question: "What is sales revenue by quarter and sales region as stacked area chart"
  };

  // Get a reference to the embedded report HTML element
  var embedContainer = document.getElementById('embedContainer');

  // Embed the report and display it within the div container.
  var embeddedObject = powerbi.embed(embedContainer, config);

</script>
```

e) Save your changes to **Qna.cshtml**.

4. Test out the application by running it in the Visual Studio debugger.

a) Press the **{F5}** key in Visual Studio to begin a new debugging session.

b) Click the **Q&A** link in the top navigation menu and you should see the Q&A experience embedded in the web page.



c) Experiment by typing questions in English and seeing how the Q&A experience responds with data and visualizations.

d) Close the browser window and return to Visual Studio and stop the current debugging session.

© Critical Path Training. 2018. All Rights Reserved

Exercise 8 - Embed a New Report

In this exercise you will implement the behavior to embed a new report based on a specific dataset. This exercise will be a bit more complicated than the previous exercises because you must implement a client-side event handler to handle the report "Save As" event in which you will redirect the browser to a new action method named **Reports** passing the new report ID in a query string parameter.

1. Add a new method to the **PbiEmbeddingManger** class named **GetNewReportEmbeddingData**.
   a) Open **PbiEmbeddedManager.cs** in an editor window if it is not already open.
   b) Navigate to the bottom of the class definition just beneath the **GetQnaEmbeddingData** method.
   c) Add a new method named **GetNewReportEmbeddingData** by copying and pasting the following code.

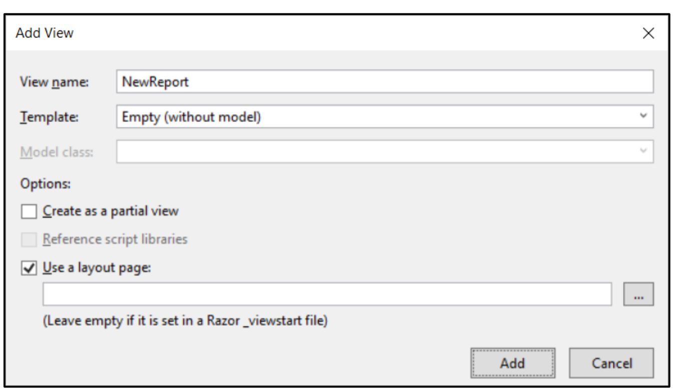
```
public static async Task<NewReportEmbeddingData> GetNewReportEmbeddingData() {

  string embedUrl = "https://app.powerbi.com/reportEmbed?groupId=" + workspaceId;

  PowerBIClient pbiClient = GetPowerBiClient();

  GenerateTokenRequest generateTokenRequestParameters =
                     new GenerateTokenRequest(accessLevel: "create", datasetId: datasetId);
  string embedToken =
    (await pbiClient.Reports.GenerateTokenForCreateInGroupAsync(workspaceId,
                                                                generateTokenRequestParameters)).Token;

  return new NewReportEmbeddingData {
    workspaceId = workspaceId,
    datasetId = datasetId,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Notice that you are required to pass a dataset ID when generating an embed token which will be used to embed a new report.

2. Add a new action method to the **HomeController** class named **NewReport**.
   a) Open **HomeController.cs** in an editor window if it's not already open.
   b) Add a new action method named **NewReport** just beneath the **Qna** method using the following code.

```
public async Task<ActionResult> NewReport() {
  NewReportEmbeddingData embeddingData = await PbiEmbeddedManager.GetNewReportEmbeddingData();
  return View(embeddingData);
}
```

3. Create a razor view for the **NewReport** action method.
   a) Right-click on the **NewReport** action method and select the **Add View…** command from the context menu.
   b) In the **Add View** dialog, accept all the default settings and click the **Add** button.



   c) You should see that a new razor view file named **NewReport.cshtml** has been created in the **Views/Home** folder.

d) Delete any existing code inside **NewReport.cshtml** and replace it with the following HTML code.

```
@model EmbeddedLab.Models.NewReportEmbeddingData

<div id="embedContainer" />

<script src="~/Scripts/powerbi.js"></script>
<script>

  // Get data required for embedding
  var embedWorkspaceId = "@Model.workspaceId";
  var embedDatasetId = "@Model.datasetId";
  var embedUrl = "@Model.embedUrl";
  var accessToken = "@Model.accessToken";

  // Get models object to access enums for embed configuration
  var models = window['powerbi-client'].models;

  var config = {
    datasetId: embedDatasetId,
    embedUrl: embedUrl,
    accessToken: accessToken,
    tokenType: models.TokenType.Embed,
  };

  // Get a reference to the embedded report HTML element
  var embedContainer = document.getElementById('embedContainer');

  // Embed the report and display it within the div container.
  var report = powerbi.createReport(embedContainer, config);

  // add event handler to load existing report afer saving new report
  report.on("saved", function (event) {
    console.log("saved");
    console.log(event.detail);
    window.location.href = "/Home/Reports/?reportId=" + event.detail.reportObjectId;
  });

</script>
```

e) Save your changes to **NewReport.cshtml**.

You should observe how the code in this script block registers a callback function by calling the **report.on("Saved")** method. You should also observe that this event handle is written to redirect the browser to the **Reports** action of the **Home** controller along with a query string parameter named **reportId** which will be used to pass the identifying GUID of the newly created report. Over the next few steps you will create the **Reports** action method in the **Home** controller class to load an existing report that has just been created.

4. Add a new method to the **PbiEmbeddingManger** class named **GetEmbeddingDataForReport**.

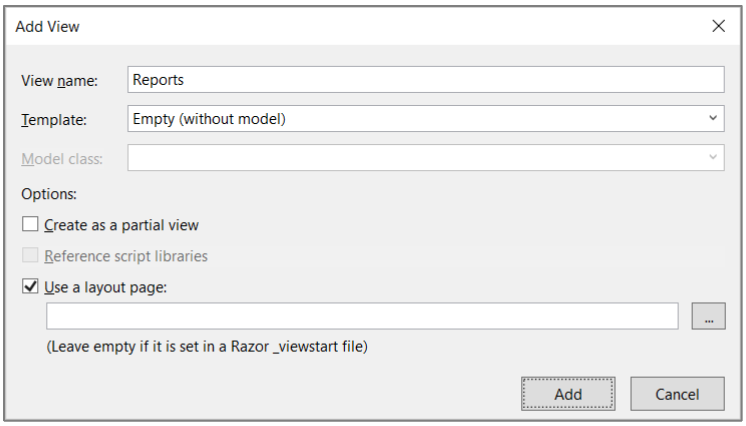
a) In **PbiEmbeddedManager.cs**, add the **GetEmbeddingDataForReport** method by copying and pasting the following code.

```
public static async Task<ReportEmbeddingData> GetEmbeddingDataForReport(string currentReportId) {
  PowerBIClient pbiClient = GetPowerBiClient();
  var report = await pbiClient.Reports.GetReportInGroupAsync(workspaceId, currentReportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");
  string embedToken =
        (await pbiClient.Reports.GenerateTokenInGroupAsync(workspaceId,
                                                           currentReportId,
                                                           generateTokenRequestParameters)).Token;

  return new ReportEmbeddingData {
    reportId = currentReportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

5. Add a new action method to the **HomeController** class named **Reports**.
   a) Open **HomeController.cs** in an editor window if it's not already open.
   b) Add a new action method named **Reports** just beneath the **NewReports** method using the following code.

```
public async Task<ActionResult> Reports(string reportId) {

  ReportEmbeddingData embeddingData =
      await PbiEmbeddedManager.GetEmbeddingDataForReport(reportId);

  return View(embeddingData);
}
```

6. Create a razor view for the **Reports** action method.
   a) Right-click on the R**eports** action method and select the **Add View…** command from the context menu.
   b) In the **Add View** dialog, accept all the default settings and click the **Add** button.



A page break has been inserted here to prevent the following code section from wrapping across pages.

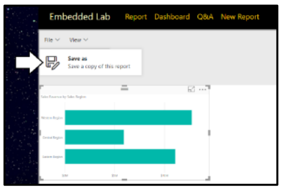
c) You should see that a new razor view file named **Reports.cshtml** has been created in the **Views/Home** folder.

d) Delete any existing code inside **Reports.cshtml** and replace it with the following HTML code.

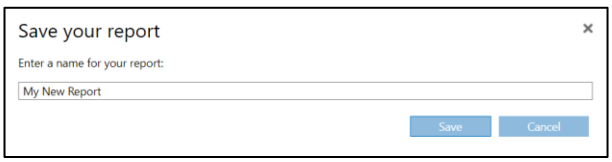
```
@model EmbeddedLab.Models.ReportEmbeddingData

@section toolbar {
  <div id="toolbar" class="btn-toolbar bg-dark" role="toolbar">
    <button type="button" id="toggleEdit" class="btn btn-sm">Toggle Edit Mode</button>
    <button type="button" id="fullScreen" class="btn btn-sm">Full Screen</button>
    <button type="button" id="print" class="btn btn-sm">Print</button>
  </div>
}

<div id="embedContainer" />


<script src="~/Scripts/powerbi.js"></script>

<script>

  // Data required for embedding Power BI report
  var embedReportId = "@Model.reportId";
  var embedUrl = "@Model.embedUrl";
  var accessToken = "@Model.accessToken";

  // Get models object to access enums for embed configuration
  var models = window['powerbi-client'].models;

  var config = {
    type: 'report',
    id: embedReportId,
    embedUrl: embedUrl,
    accessToken: accessToken,
    tokenType: models.TokenType.Embed,
    permissions: models.Permissions.All,
    viewMode: models.ViewMode.Edit,
    settings: {
      filterPaneEnabled: false,
      navContentPaneEnabled: true,
    }
  };

  // Get a reference to HTML element that will be embed container
  var reportContainer = document.getElementById('embedContainer');

  // Embed the report and display it within the div container.
  var report = powerbi.embed(reportContainer, config);

  var viewMode = "edit";

  $("#toggleEdit").click(function () {
    // toggle between view and edit mode
    viewMode = (viewMode == "view") ? "edit" : "view";
    report.switchMode(viewMode);
    // show filter pane when entering edit mode
    var showFilterPane = (viewMode == "edit");
    report.updateSettings({
      "filterPaneEnabled": showFilterPane
    });
  });

  $("#fullScreen").click(function () {
    report.fullscreen();
  });

  $("#print").click(function () {
    report.print();
  });

</script>
```

© Critical Path Training. 2018. All Rights Reserved

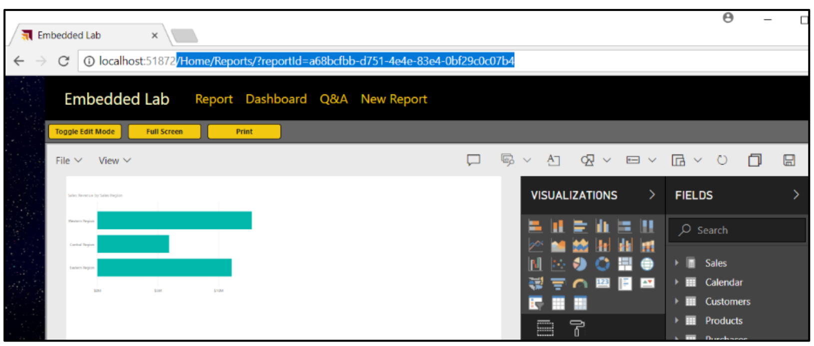
7. Test out the application by running it in the Visual Studio debugger.
   a) Press the **{F5}** key in Visual Studio to begin a new debugging session.
   b) Click the **New Report** link in the top navigation menu and you should see an new empty in design mode.
   c) Add a simple visual to the new report.
   d) Save the new report by dropping down the **File** menu and selecting the **Save as** command.



   e) In the **Save your report** dialog, give the new report a name such as **My New Report** and click the **Save** button.



   f) After the report has been saved, the browser should redirect to the **Home/Reports** action method and your application should be able to load in the newly created report using the GUID for its report ID.



   g) When you are done with your testing, close the browser, return to Visual Studio and stop the current debugging session.

© Critical Path Training. 2018. All Rights Reserved
www.CriticalPathTraining.com