

Embedding Power BI Content

Overview

The estimated time to complete the lab is 45 minutes

*You must have completed **Lab 01A** before commencing this lab.*

In this lab, you will commence by registering a client app, and setting delegated permissions. You will then create an ASP.NET web application, and develop a web form for enabling the report selection of Power BI reports. This form will then be extended with logic to embed the selected report. Having completed report embedding, you will clone the web form and adapt it to embed dashboards. Lastly, you will finalize the web application by creating a default page to navigate to either of the embed pages.

This lab focuses on development practices for embedding Power BI reports and dashboards. It does not intend to convey good web application design practices, including—but not limited to—secure storage and retrieval of sensitive data, exception handling, store and reuse of access tokens, and web site styling.

Registering a Client App

In this exercise, you will register an app and configure delegated permissions.

Registering a Client App

In this task, you will register a Power BI client app.

1. Within the existing Google Chrome window, add a new tab, and then navigate to <https://dev.powerbi.com/apps>.
2. At Step 1, click **Sign In with Your Existing Account**.

A yellow rectangular button with the text "Sign in with your existing account" in black.

3. Verify that you have been signed in with the account provided to you.

A screenshot of the Power BI developer portal login page. It shows the heading "Step 1 Login to your Power BI account" and a welcome message "Welcome, [redacted]!". Below the welcome message is a link to "logout" and a note "(Wrong account? No problem, logout and try again.)".

Step 1 Login to your Power BI account

Welcome, [redacted]! (Wrong account? No problem, [logout](#) and try again.)

4. At Step 2, configure the following properties.


For convenience, the **Redirect URL** can be copied from the **<CourseFolder>\PowerBIDev\AD\Lab02A\Assets\Snippets.txt** file.

Property	Value
App Name	Enter a unique app name that will be easy to recognize
App Type	Native app (select from dropdown list)
Redirect URL	https://lab.powerbi.com/gettoken

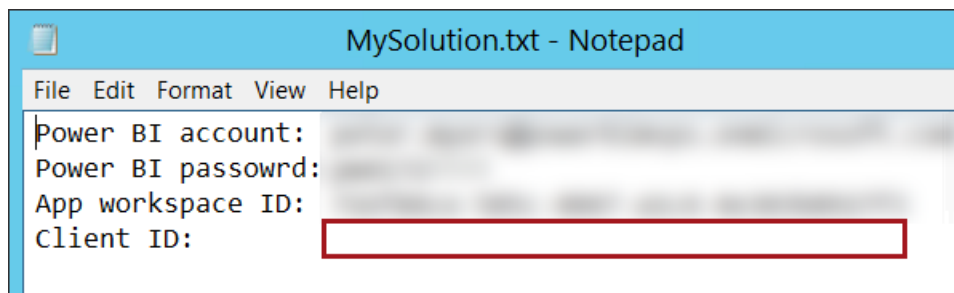
5. At Step 3, to grant access to all APIs, check all checkboxes.

There are additional permissions required which are not exposed in the registration tool. You will use the Azure Portal to set these permissions in the next task.

6. At Step 4, click **Register App**.

A yellow rectangular button with the text "Register App" in black.

7. When the app has registered, copy the **Client ID** value to the clipboard.
8. For future reference, enter your client ID into the **MySolution.txt** file.



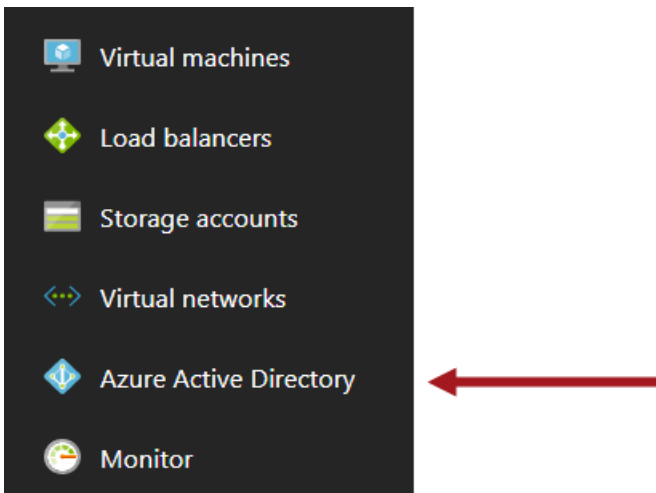
You will require the client ID when embedding the app workspace Power BI content.

9. Save the **MySolution.txt** file.

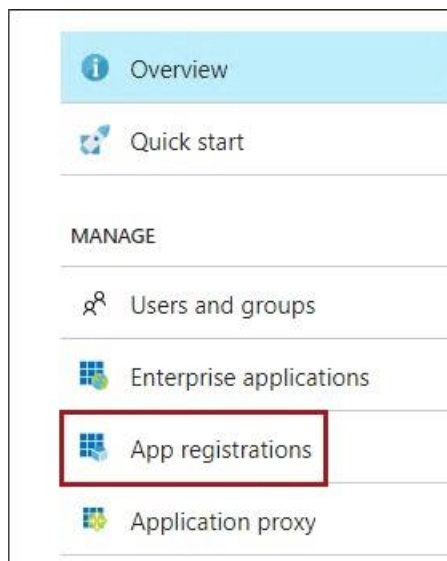
Delegating Additional Permissions

In this task, you will sign in to the Azure Portal, and delegate additional client app permissions.

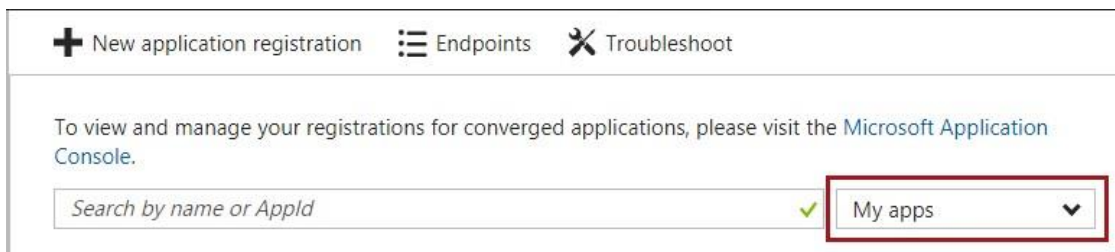
1. Within the same Google Chrome window, add a new tab, and then navigate to <https://portal.azure.com>.
2. In the left pane, select **Azure Active Directory**.



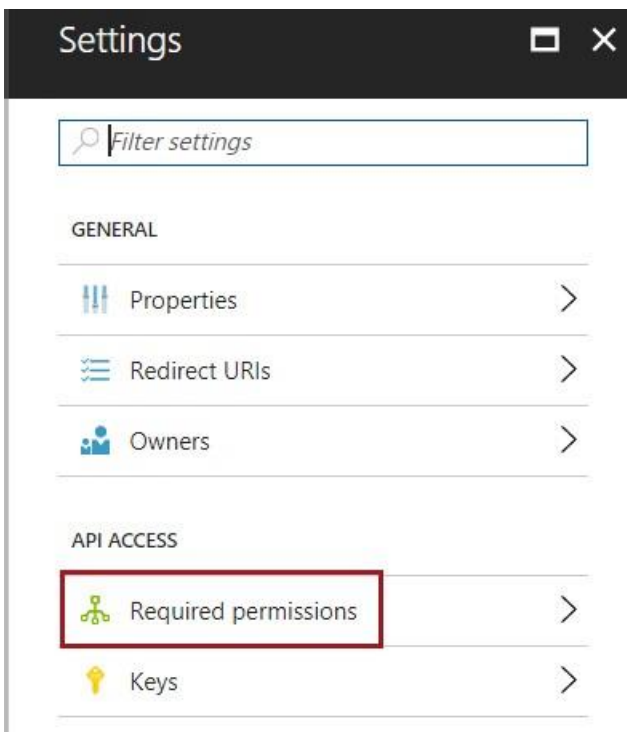
3. In the **Azure Active Directory** blade, from inside the **Manage** group, select **App Registrations**.



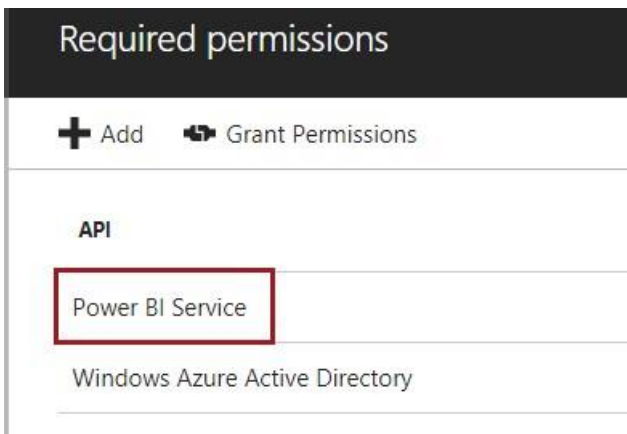
4. In the result pane, in the dropdown list, select **My Apps**.



5. In the search results, select the app that you created in the previous task.
6. In the **Settings** blade, from inside the **API Access** group, select **Required Permissions**.



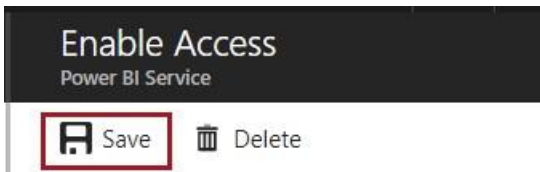
7. In the **Required Permissions** blade, select the **Power BI Service** API.



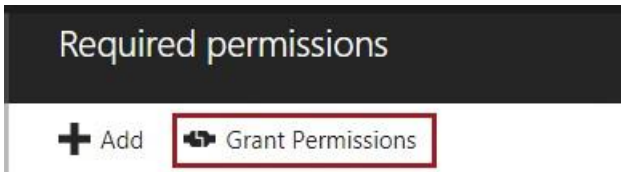
8. In the **Enable Access** blade, to delegate all permissions, check the checkbox the left of **Delegated Permissions**.



9. Click **Save**.



10. When the permissions are saved, in the **Required Permissions** blade, click **Grant Permissions**.



11. When prompted to confirm, click **Yes**.



Developing with the REST API

In this exercise, you will create a web application project, and then use the Power BI REST API to present a dropdown list of reports on a web page.

Creating an ASP.NET Web Application

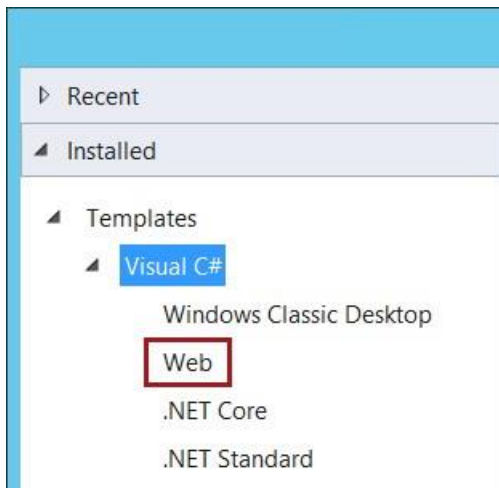
In this task, you will create an ASP.NET web application.

1. Open Visual Studio.

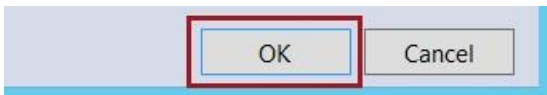
The lab steps have been written and tested by developing a solution with Visual Studio 2017. The labs will also work in Visual Studio 2015 and Visual Studio Community; however, screenshots and lab instructions may differ slightly.



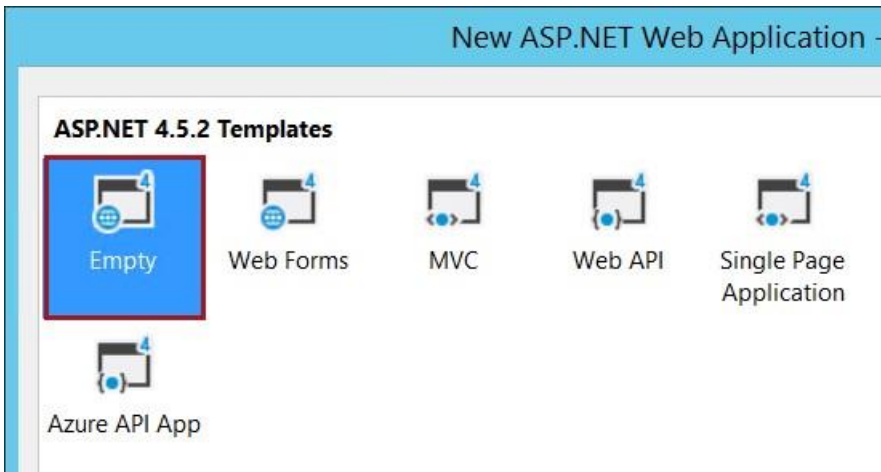
2. To create a new project, on the **File** menu, select **File | New Project**.
3. In the **New Project** window, in the left pane, expand **Visual C#**, and then select **Web**.



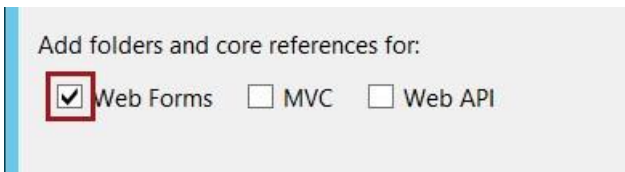
4. In the **Name** box, replace the text with **PowerBIEmbedding**.
5. In the **Solution Name** box, replace the text, also with **PowerBIEmbedding**.
6. Click **OK**.



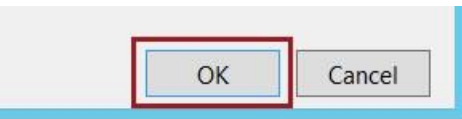
7. In the **New ASP.NET Web Application** window, ensure that the **Empty** template is selected.



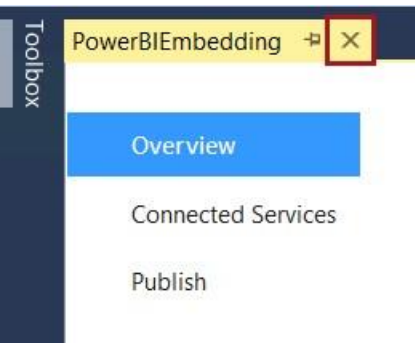
8. Check the **Web Forms** checkbox.



9. Click **OK**.



10. When the project has been created, close the project file.



Installing NuGet Packages

In this task, you will install two NuGet packages to support developing with the Power BI REST API, and AAD authentication.

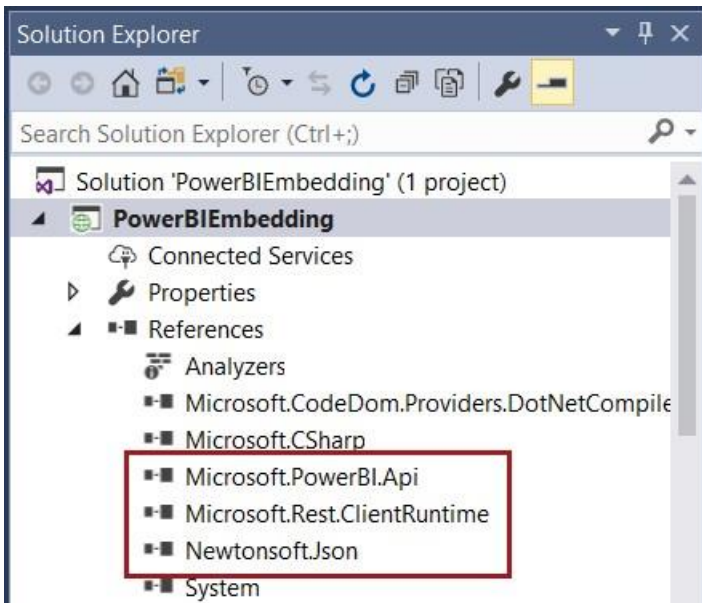
1. On the **Tools** menu, select **NuGet Package Manager | Package Manager Console**.
2. In the **Package Manager Console** pane, enter the following command:

*For convenience, the command can be copied from the
<CourseFolder>\PowerBIDev\AD\Lab02A\Assets\Snippets.txt file.*

NuGet

Install-Package Microsoft.PowerBI.Api

3. Press **Enter**.
4. When the installation has completed, in **Solution Explorer**, expand the **References** folder.
5. Notice the addition of:
 - Microsoft.PowerBI.Api
 - Microsoft.Rest.ClientRuntime
 - Newtonsoft.Json



6. Install a second package:

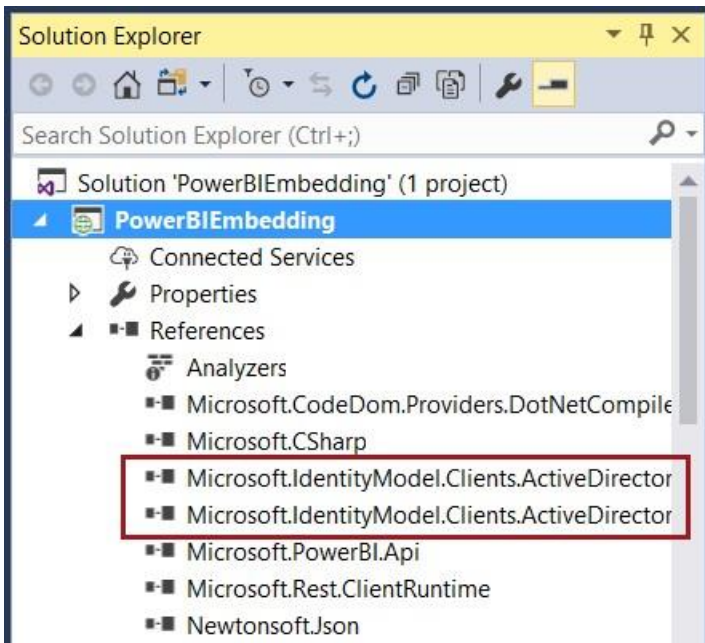
*For convenience, the command can be copied from the **<CourseFolder>\PowerBIDev\AD\Lab02A\Assets\Snippets.txt** file.*

NuGet

Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory

This package is required to authenticate the master app account.

7. In the **References** folder, notice the addition of:
 - Microsoft.IdentityModel.Clients.ActiveDirectory
 - Microsoft.IdentityModel.Clients.ActiveDirectory.Platform



8. Collapse the **References** folder.

Configuring the Web.config File

In this task, you will configure the Web.config file to store service endpoints, and also details of your workspace app id, client id, and master app credentials.

1. In **Solution Explorer**, to open the file, double-click the **Web.config** file.
2. Immediately beneath the **configuration** element, on a new line, paste the **appSettings** element.

For convenience, the element can be copied from the <CourseFolder>\PowerBIDev\AD\Lab02A\Assets\Snippets.txt file.

XML

```
<appSettings>
  <add key="authorityUrl" value="https://login.windows.net/common/oauth2/authorize/" />
  <add key="resourceUrl" value="https://analysis.windows.net/powerbi/api" />
  <add key="apiUrl" value="https://api.powerbi.com/" />
  <add key="embedUrlBase" value="https://app.powerbi.com/" />
  <add key="appWorkspaceId" value="" />
  <add key="clientId" value="" />

  <!-- Note: Do NOT leave your credentials in clear text. Retrieve them from a secure store. -->
  <add key="pbiUsername" value="" />
  <add key="pbiPassword" value="" />
</appSettings>
```

3. Copy the **App Workspace ID** value from **MySolutions.txt** file into the value of the **appWorkspaceId** key.

```

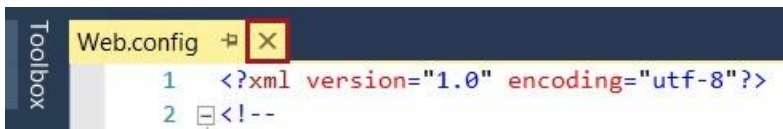
6 <configuration>
7 <appSettings>
8 <add key="authorityUrl" value="https://login.windows.net/common/oauth2/authorize/" />
9 <add key="resourceUrl" value="https://analysis.windows.net/powerbi/api" />
10 <add key="apiUrl" value="https://api.powerbi.com/" />
11 <add key="embedUrlBase" value="https://app.powerbi.com/" />
12 <add key="appWorkspaceId" value="" />
13 <add key="clientId" value="" />

```

4. Copy the **Client ID** value from **MySolutions.txt** file into the value of the **clientId** key.
5. Notice the warning noting that the master app account credentials should not be stored in clear text.

For simplicity in this lab, they will be stored in clear text.

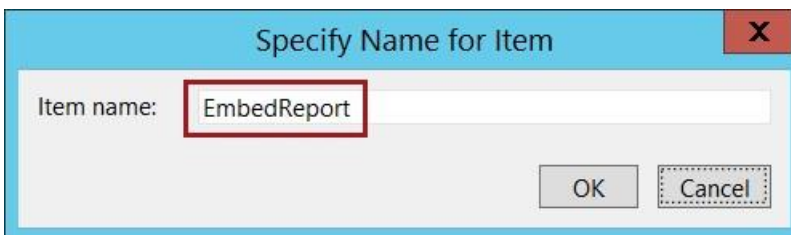
6. Copy the **Power BI Account** value from **MySolutions.txt** file into the value of the **pbiUsername** key.
7. Copy the **Power BI Password** value from **MySolutions.txt** file into the value of the **pbiPassword** key.
8. To save the file, on the **File** menu, select **Save Web.config**.
9. To close the file, close the tab.



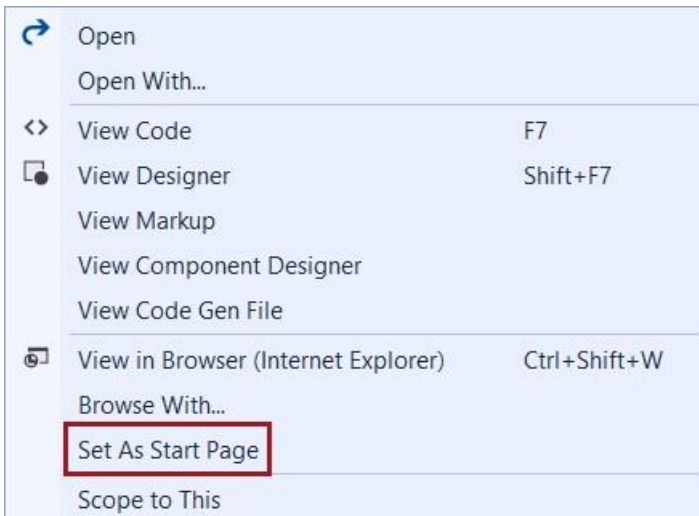
Adding a Web Form

In this task, you will add a web form that will be used to embed a report, and add a server-side dropdown list.

1. In **Solution Explorer**, right-click the **PowerBIEmbedding** project (not the solution), and then select **Add | Web Form**.
2. In the **Specify Name for item** window, in the **Item Name** box, replace the text with **EmbedReport**.




3. Click **OK**.
4. Notice that the web page file (**EmbedReport.aspx**) has automatically opened.
5. In **Solution Explorer**, right-click the **EmbedReport.aspx** item, and then select **Set as Start Page**.



6. To add a dropdown list, beneath the **form** element, add the following element:

```

9  <body>
10     <form id="form1" runat="server">
11         
12         <div>
13             </div>
14     </form>
15 </body>
16 </html>
17

```

For convenience, the element can be copied from the **<CourseFolder>\PowerBIDev\AD\Lab02A\Assets\Snippets.txt** file.

ASP.NET

```

<asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True"
    OnSelectedIndexChanged="Page_Load" />

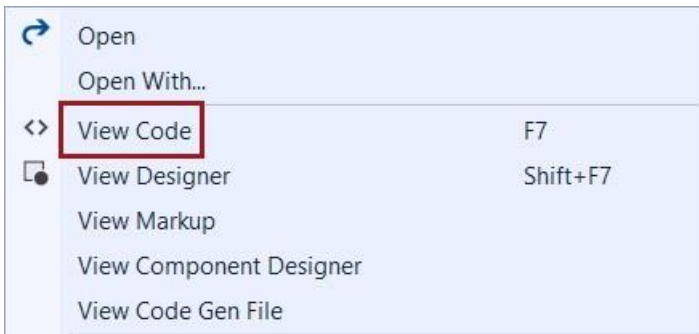
```

This element will add a server-side dropdown list, which will be populated with one item for each report found in the app workspace. It will raise an index changed event when the app user selected a different dropdown item.

Developing the Web Form

In this task, you will develop the web form to retrieve an Access token, and then use the token the retrieve all report definitions from the app workspace.

1. In **Solution Explorer**, right-click the **EmbedReport.aspx** item, and then select **View Code**.



- To develop the app settings retrieval code, add the following namespace directive beneath the last namespace directive (line 6).

For convenience, all remaining code in this lab can be copied from the <CourseFolder>PowerBIDev\AD\Lab02A\Assets\Snippets.txt file.

Visual C#

```
using System.Configuration;
```

- Add the following class-level declarations inside the **EmbedReport** class.

```

9  namespace PowerBIEmbedding
10 {
11     1 reference
12     public partial class EmbedReport : System.Web.UI.Page
13     {
14         0 references
15         protected void Page_Load(object sender, EventArgs e)
16     {

```

Visual C#

```

private static readonly string AuthorityUrl = ConfigurationManager.AppSettings["authorityUrl"];
private static readonly string ResourceUrl = ConfigurationManager.AppSettings["resourceUrl"]; private
static readonly string ApiUrl = ConfigurationManager.AppSettings["apiUrl"];
private static readonly string AppWorkspaceId = ConfigurationManager.AppSettings["appWorkspaceId"]; private
static readonly string ClientId = ConfigurationManager.AppSettings["clientId"];

private static readonly string Username = ConfigurationManager.AppSettings["pbiUsername"]; private
static readonly string Password = ConfigurationManager.AppSettings["pbiPassword"];

```

*Tip: When pasting snippets, they can be easily formatted (correctly indented) by first selecting the inserted lines, and then by using the **Edit** menu, and selecting **Format Selection (Ctrl+K, Ctrl+F)**.*

*This code retrieves all **Web.config** app setting values and stores them in read-only variables.*

- To develop the Access token retrieval code, add the following two namespace directive beneath the last added namespace directive.

Visual C#


```

using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.Rest;

```

- Add the following code inside the **Page_Load** method.

```

26  protected void Page_Load(object sender, EventArgs e)
27  {
28      
29  }
30  }

```

Visual C#

```

var credential = new UserPasswordCredential(Username, Password);
// Authenticate using app settings credentials var authenticationContext = new
AuthenticationContext(AuthorityUrl); var authenticationResult =
authenticationContext.AcquireTokenAsync(ResourceUrl, ClientId, credential).Result;
var tokenCredentials = new TokenCredentials(authenticationResult.AccessToken,
"Bearer");

```

If this code succeeds, authentication has succeeded.

6. To populate the dropdown list with the app workspace reports, add the following two namespace directive beneath the last added namespace directives.

Visual C#

```

using Microsoft.PowerBI.Api.V2; using
Microsoft.PowerBI.Api.V2.Models;

```

Add the following code inside the **Page_Load** method, beneath the last added lines.

Visual C#

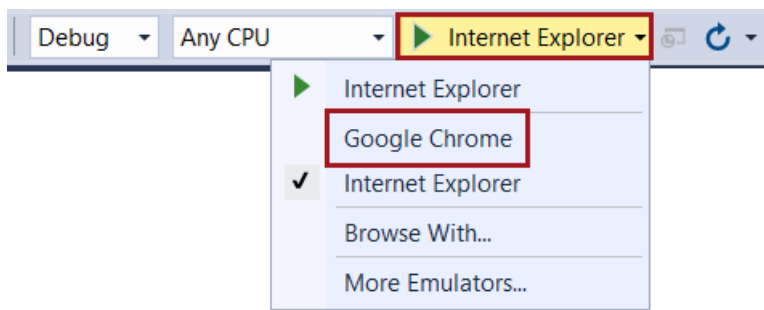
```

// Populate the dropdown list if
(!IsPostBack)
{ using (var client = new PowerBIClient(new Uri(ApiUrl), tokenCredentials))
{
    // Get a list of reports var reports =
client.Reports.GetReportsInGroup(AppWorkspaceId);
    // Populate dropdown list foreach
(Report item in reports.Value)
    {
        ddlReport.Items.Add(new ListItem(item.Name, item.Id));
    }

    // Select first item ddlReport.SelectedIndex = 0;
}
}

```

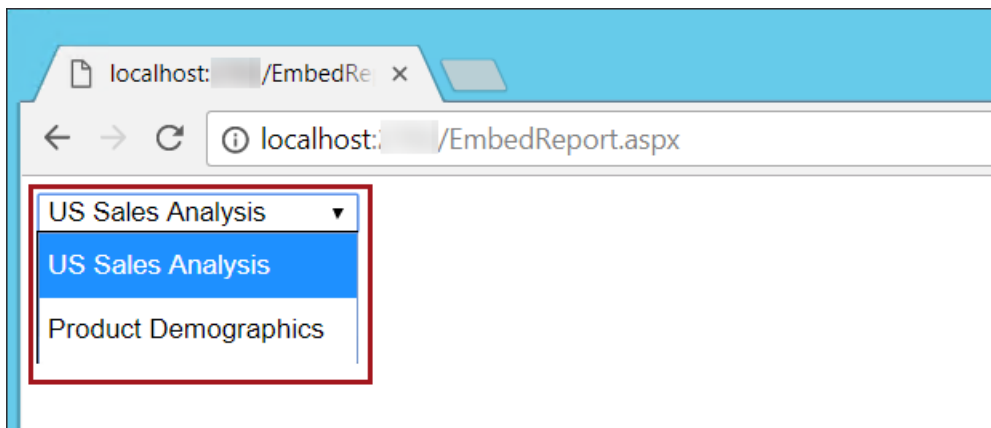
7. To start the web application, on the toolbar, first set the web browser to **Google Chrome**.



8. On the toolbar, start the web application (start debugging).



9. When the web browser opens, verify that a dropdown list appears, and that it contains two items, one for each report.



10. To stop debugging, close the web browser.

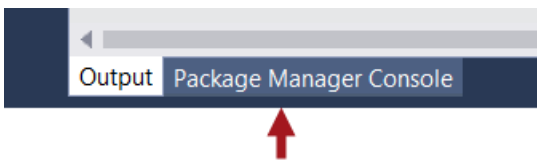
Embedding a Report

In this exercise, you will develop the web form to embed a report.

Installing a NuGet Package

In this task, you will install a NuGet package to embed reports client-side with the Power BI JavaScript API.

1. At the bottom-left corner, select the **Package Manager Console** pane.

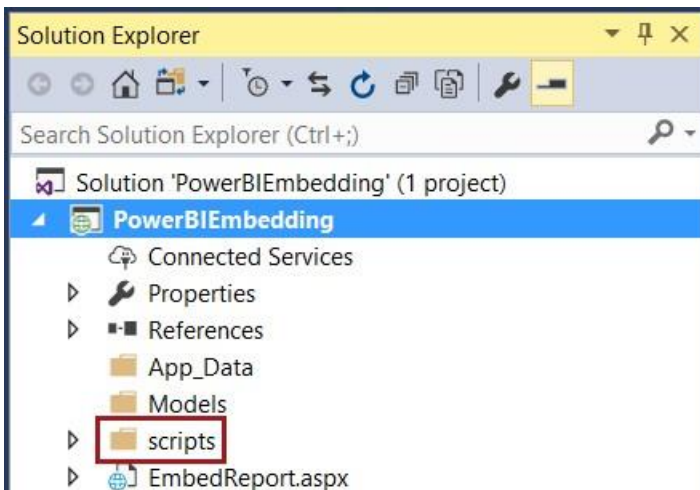


2. In the **Package Manager Console** pane, install the following package:

NuGet

Install-Package Microsoft.PowerBI.JavaScript

3. When the installation has completed, in **Solution Explorer**, notice the addition of the **scripts** folder.



4. Expand the **scripts** folder, and notice the **powerbi.js** file.



*The **powerbi.js** file must be referenced by web pages that will perform embedding.*

Developing the Web Form

In this task, you will continue to develop the web form to embed a report.

1. Switch to the **EmbedReport.aspx** file.
2. Immediately after the **body** element, insert the following **script** element.

```

9   <body>
10  ──┐
11     <form id="form1" runat="server">

```

HTML

```
<script src="/scripts/powerbi.js"></script>
```

3. Immediately after the **asp:DropDownList** element, replace the **div** element (both lines) with the following **div** element.

```

11  <form id="form1" runat="server">
12      <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True"
13          <div>
14          </div>
15  </form>

```

HTML

```
<div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />
```

4. Switch to the **EmbedReport.aspx.cs** file (code-behind).
5. Add the following additional class-level declarations inside the **EmbedReport** class, after the last added declarations.

```

23  private static readonly string Username = ConfigurationManager.AppSettings["pbiUsername"];
24  private static readonly string Password = ConfigurationManager.AppSettings["pbiPassword"];
25  ──┐
26     ──┐

```

Visual C#

```

public string embedToken;
public string embedUrl;
public string reportId;

```

These variables will store values required for client-side embedding.

6. Add the following additional code inside the **Page_Load** method, after the last added code.

```

54  // Select first item
55  ddlReport.SelectedIndex = 0;
56  }
57  }
58  }
59  ──┐

```

Visual C#

```
// Generate an embed token and populate embed variables using (var client
= new PowerBIClient(new Uri(ApiUrl), tokenCredentials)) {
    // Retrieve the selected report var report =
client.Reports.GetReportInGroup(AppWorkspaceId, ddlReport.SelectedValue);
    // Generate an embed token to view var generateTokenRequestParameters = new
GenerateTokenRequest(accessLevel: "view"); var tokenResponse =
client.Reports.GenerateTokenInGroup(AppWorkspaceId, report.Id,
    generateTokenRequestParameters);

    // Populate embed variables (to be passed client-side)
embedToken = tokenResponse.Token; embedUrl =
report.EmbedUrl; reportId = report.Id;
}
```

7. Switch to the **EmbedReport.aspx** file.

8. Immediately after the **div** element, add the following **script** element.

```
11 <form id="form1" runat="server">
12     <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True" OnSelected
13     <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />
14     </div>
15 </form>
```

HTML

```

<script>
// Read embed token
var embedToken = " ";<% =this.embedToken %>

// Read embed URL
var embedUrl = " ";<% = this.embedUrl %>

// Read report Id
var reportId = " ";<% = this.reportId %>

// Get models (models contains enums) var
models = window['powerbi-client'].models;

// Embed configuration is used to describe what and how to embed
// This object is used when calling powerbi.embed
// It can also includes settings and options such as filters var
config = {
    type: 'report',          tokenType:
models.TokenType.Embed,    accessToken:
embedToken, embedUrl: embedUrl,
    id: reportId,          settings: {
        filterPaneEnabled: true,
        navContentPaneEnabled: true
    }
};

// Embed the report within the div element var
report = powerbi.embed(embedDiv, config);
</script>

```

Reviewing the Web App

In this task, you will start the web app, and then review the **EmbedReport.aspx** web form design.

1. Switch to the **EmbedReport.aspx** file.
2. On the toolbar, start the web application (start debugging).



3. When the web browser opens, wait until the **US Sales Analysis** report is embedded.
4. Interact with the report by:
 - Modifying the **Year** slicer value
 - Multi-selecting **Category** slicer values
 - Highlighting (selecting) month columns, or demographic bars
 - Opening the **Filters** pane, and selecting multiple regions

5. To embed the second report, in the dropdown list, select the **Product Demographics** report.
6. When the report has loaded, to stop debugging, close the web browser.

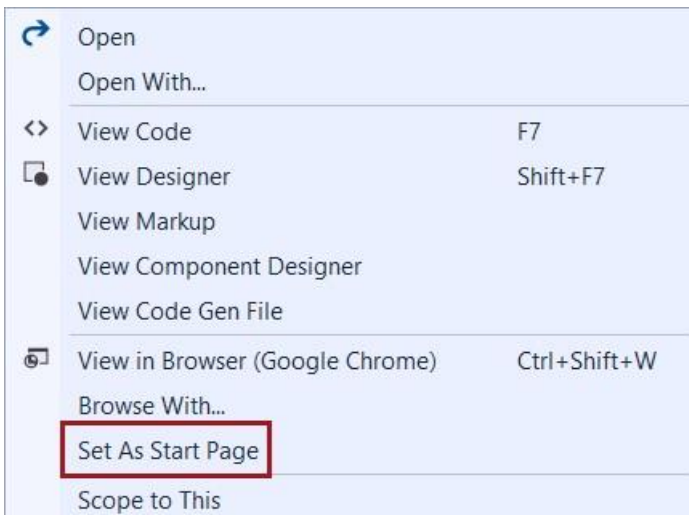
Embedding a Dashboard

In this exercise, you will develop a new web form to embed a dashboard.

Cloning the Web Form

In this task, you will clone (copy) the **EmbedReport.aspx** project item.

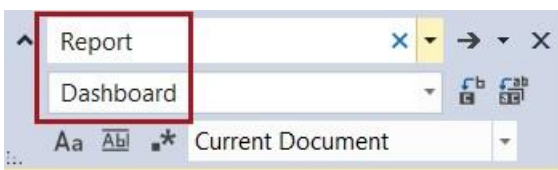
1. Close all open project items.
2. In **Solution Explorer**, right-click the **EmbedReport.aspx** item, and then select **Copy**.
3. In **Solution Explorer**, right-click the **PowerBIEmbedding** project, and then select **Paste**.
4. In **Solution Explorer**, right-click the **EmbedReport - Copy.aspx** item, and then select **Rename**.
5. Rename the item to **EmbedDashboard.aspx**, and then press **Enter**.
6. Set the **EmbedDashboard.aspx** item as the start page.



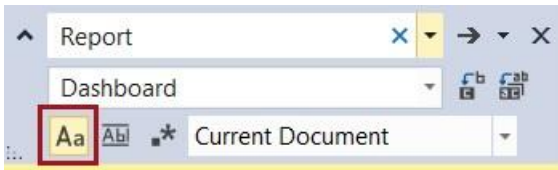
Adapting the Web Form

In this task, you will adapt the new web form by replacing all instances of the word **Report** with **Dashboard**.

1. Open the **EmbedDashboard.aspx** item.
2. To perform search-and-replace, press **Ctrl+H**.
3. In the **Find** box, enter **Report**.
4. In the **Replace** box, enter **Dashboard**.

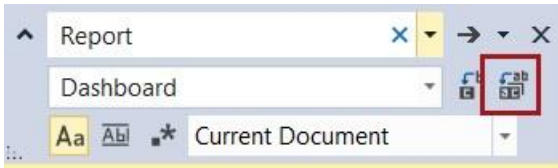


- Click to enable **Match Case**.

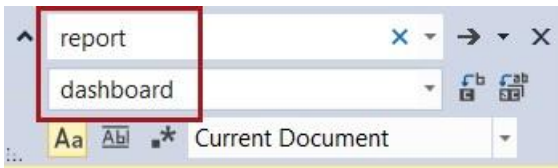


Visual C# is a case-sensitive language, and so you will replace the words twice, once with a capitalized first letter, and once in lower-case.

- Click **Replace All**.




- When prompted with the message, click **OK**.
- Replace all lower-cased instances also.



- In the div element, remove the **max-width** style property.

```
13 | <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />
```



*Unlike reports which are fixed dimension, dashboards can be almost any practical dimensions. It is possible, however, to modify the **pageView** setting to determine how “overflow” content is made visible.*

- In the **script** element, modify the **config** variable declaration by removing the **settings** assignment (leave the comma after the **id** assignment).

```
30 |  
31 |  
32 |  
33 |  
34 |  
35 |  
36 |  
37 |  
38 |  
39 |  
40 |
```

```
var config = {  
  type: 'dashboard',  
  tokenType: models.TokenType.Embed,  
  accessToken: embedToken,  
  embedUrl: embedUrl,  
  id: dashboardId,  
  settings: {  
    filterPaneEnabled: true,  
    viewContentPaneEnabled: true  
  }  
};
```



Settings are specific to report embedding.

11. Add the **pageView** assignment in place of the **settings** assignment.

```
30 var config = {  
31     type: 'dashboard',  
32     tokenType: models.TokenType.Embed,  
33     accessToken: embedToken,  
34     embedUrl: embedUrl,  
35     id: dashboardId,  
36     pageView: "actualSize"  
37 };
```

Adapting the Web Form Code

In this task, you will adapt the new web form code by replacing all instances of the word **Report** with **Dashboard**, and addressing several inconsistencies.

1. Open the **EmbedDashboard.aspx.cs** item.
2. Perform the same two search-and-replace operations applied to the **EmbedDashboard.aspx** item.

This approach almost works... however there are two inconsistencies in the API that will need to be addressed. ☹️

3. In the **Page_Load** method, when populating the dropdown list, modify the **Name** property to **DisplayName**.

```
48 // Populate dropdown list  
49 foreach (Dashboard item in dashboards.Value)  
50 {  
51     ddlDashboard.Items.Add(new ListItem(item.DisplayName, item.Id));  
52 }
```

4. In the **Page_Load** method, when retrieving the dashboard, replace the line of code with the following.

*There is no **GetDashboardInGroup** method.*

Visual C#

```
var dashboard = client.Dashboards.GetDashboardsInGroup(AppWorkspaceId).Value  
.FirstOrDefault(d => d.Id == ddlDashboard.SelectedValue);
```

Reviewing the Web App

In this task, you will start the web application, and then review the **EmbedDashboard.aspx** web form design.

1. On the toolbar, start the web application (start debugging).



2. When the web browser opens, wait until the **US Sales Analysis** dashboard is embedded.

Unlike embedded reports, embedded dashboards do not support any interactivity.

Depending on your screen size, it is possible that the dashboard does not display all tiles, and will require scrolling to see them. You will explore the `pageView` setting in the following steps which provide control over how the embedded dashboard will display its tiles.

3. To stop debugging, close the web browser.
4. In the **EmbedDashboard.aspx** web form, modify the **pageView** assignment to **fitToWidth**.

```
30  var config = {  
31      type: 'dashboard',  
32      tokenType: models.TokenType.Embed,  
33      accessToken: embedToken,  
34      embedUrl: embedUrl,  
35      id: dashboardId,  
36      pageView: "fitToWidth"  
37  };
```

5. Start the application again, and review the embedded dashboard.
6. To stop debugging, close the web browser.
7. In the **EmbedDashboard.aspx** web form, modify the **pageView** assignment to **oneColumn**.

```
30  var config = {  
31      type: 'dashboard',  
32      tokenType: models.TokenType.Embed,  
33      accessToken: embedToken,  
34      embedUrl: embedUrl,  
35      id: dashboardId,  
36      pageView: "oneColumn"  
37  };
```

8. Start the application again, and review the embedded dashboard.
9. To stop debugging, close the web browser.

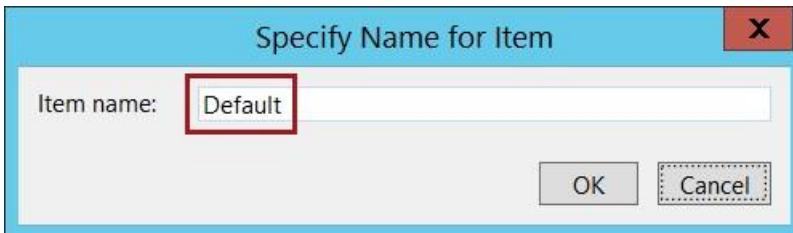
Finalizing the App Design

In this exercise, you will finalize the web application design by adding a default page to enable navigation to either of the embed pages.

Finalizing the App Design

In this task, you will finalize the web application design by adding a default page to enable navigation to either of the embed pages.

1. In **Solution Explorer**, right-click the **PowerBIEmbedding** project, and then select **Add | HTML Page**.
2. In the **Specify Name for Item** window, replace the text with **Default**.



3. Click **OK**.
4. Set the **Default.html** item as the start page.



5. Within the **body** element, insert the following HTML to add hyperlinks to each embed page.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6  </head>
7  <body>
8      <!-- Insert HTML here -->
9  </body>
10 </html>
```

HTML

```
<a href="EmbedReport.aspx">Embed Reports</a><br />
<a href="EmbedDashboard.aspx">Embed Dashboards</a><br />
```

6. Start the application, and test each link.
7. To stop debugging, close the web browser.
8. Close all open project items.
9. On the **File** menu, select **Save All**.
10. Leave the solution open.

*You will enhance the application with client-side logic in **Lab 02B**.*

Summary

In this lab, you commenced by registering a client app, and setting delegated permissions. You then created an ASP.NET web application, and developed a web form for enabling the selection of Power BI reports. This form then was extended with logic to embed the selected report. Having completed report embedding, you cloned the web form and adapted it to embed dashboards. Lastly, you finalized the web application by creating a default page to navigate to either of the embed pages.