Embedding Power BI Reports and Dashboards

Setup Time: 60 minutes

Lab Folder: C:\Student\Modules\08_PBIEmbedding\Lab

Overview: In this lab you will work through the steps to develop a custom web application that embeds Power BI reports and dashboards. You will begin by creating an app workspace and populating it embeddable Power BI resources including a dataset, a report and a dashboard. Next, you will register a new application with Azure AD in the Azure portal and configure this Azure AD application with the proper permissions to program using the Power BI Service API. After that, you will create a new web application using Visual Studio 2017 and ASP.NET MVC. As you move through the exercises of this lab, you will program against both the Power BI Service API and the Power BI JavaScript API to implement the required steps to embed Power BI reports and dashboards into your custom web application.

Exercise 1: Embed a Power BI 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.

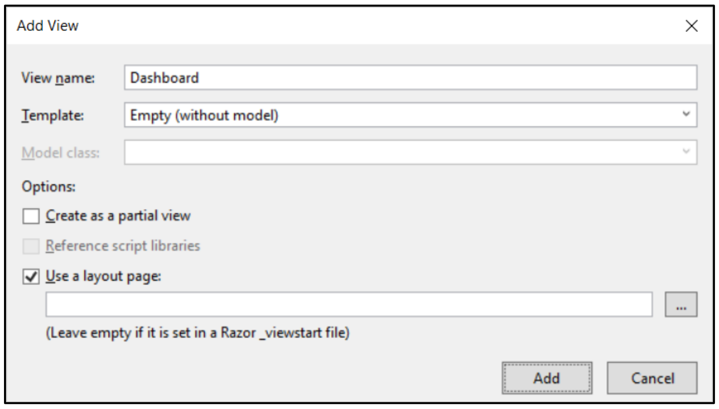
1. Add a new method to the **PbiEmbeddingManger** class named **GetDashboardEmbeddingData**.
   a) Open **PbiEmbeddedManager.cs** in an editor window if it's not already open.
   b) Navigate to the bottom of the class definition just beneath the **GetReportEmbeddingData** method
   c) Paste in the definition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


Add View

View name: Dashboard

Template: Empty (without model)

Model class:

Options:

- [ ] Create as a partial view
- [ ] Reference script libraries
- [x] Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

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.

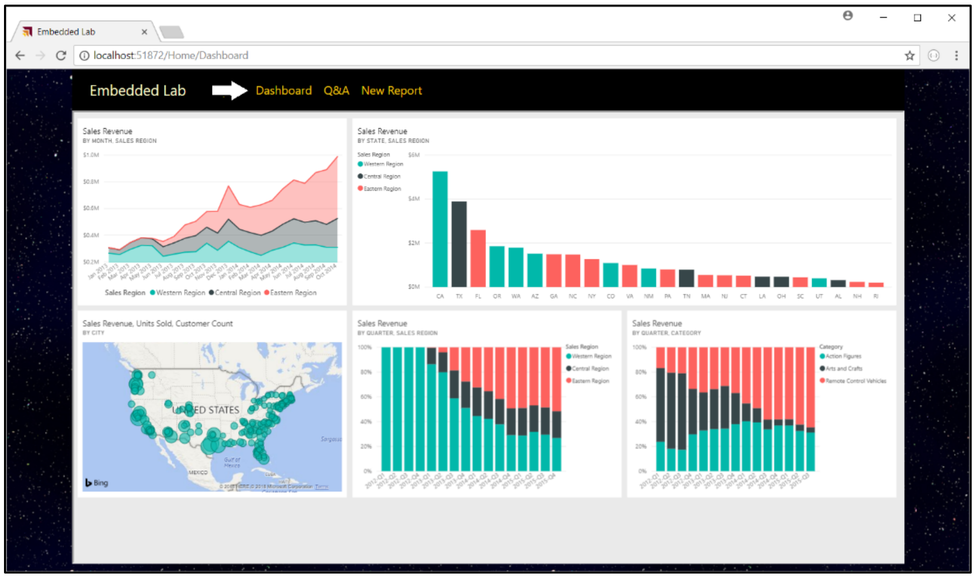
```
@model EmbeddedLab.Models.DashboardEmbeddingData

<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.

Exercise 2: Embed Power BI Dashboard Tiles

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**.

Exercise 3: 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.

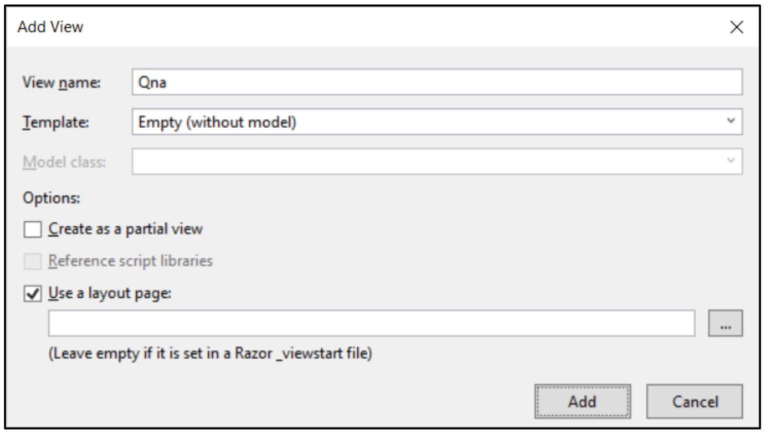
b) Add a new action method named **Qna** just beneath the **Dashboard** method using the following code.

```
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.


Add View
View name: Qna
Template: Empty (without model)
Model class:
Options:
Create as a partial view
Reference script libraries
Use a layout page:
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

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

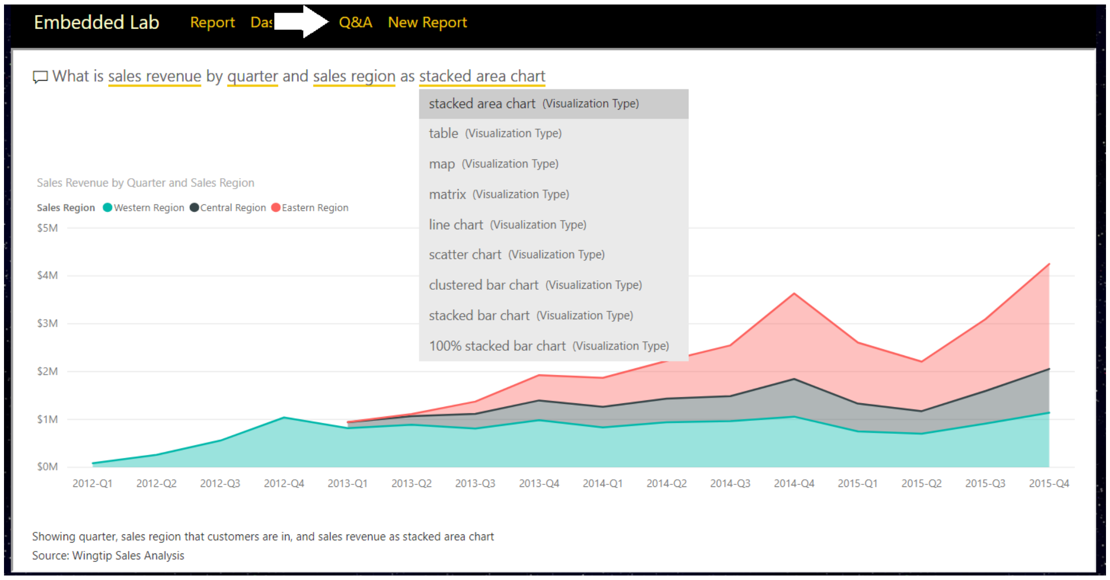
d) Delete any existing code inside **Qna.cshtml** and replace it with the following HTML code.

```
@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.


Embedded Lab
Report
Das
Q&A
New Report
What is sales revenue by quarter and sales region as stacked area chart
stacked area chart (Visualization Type)
table (Visualization Type)
map (Visualization Type)
matrix (Visualization Type)
line chart (Visualization Type)
scatter chart (Visualization Type)
clustered bar chart (Visualization Type)
stacked bar chart (Visualization Type)
100% stacked bar chart (Visualization Type)
Sales Revenue by Quarter and Sales Region
Sales Region ● Western Region ● Central Region ● Eastern Region
$5M
$4M
$3M
$2M
$1M
$0M
2012-Q1 2012-Q2 2012-Q3 2012-Q4 2013-Q1 2013-Q2 2013-Q3 2013-Q4 2014-Q1 2014-Q2 2014-Q3 2014-Q4 2015-Q1 2015-Q2 2015-Q3 2015-Q4
Showing quarter, sales region that customers are in, and sales revenue as stacked area chart
Source: Wingtip Sales Analysis

   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.

Exercise 4: Embed a New Report with Save-as Redirect Logic

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.

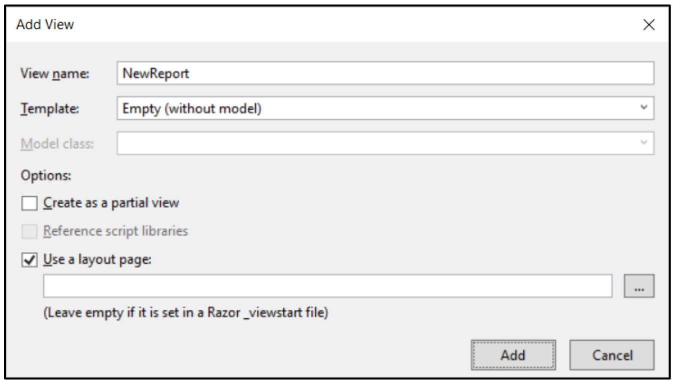
```
public static async Task<NewReportEmbeddingData> GetNewReportEmbeddingData() {

  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.


Add View
View name: NewReport
Template: Empty (without model)
Model class:
Options:
☐ Create as a partial view
☐ Reference script libraries
☑ Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)

   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.

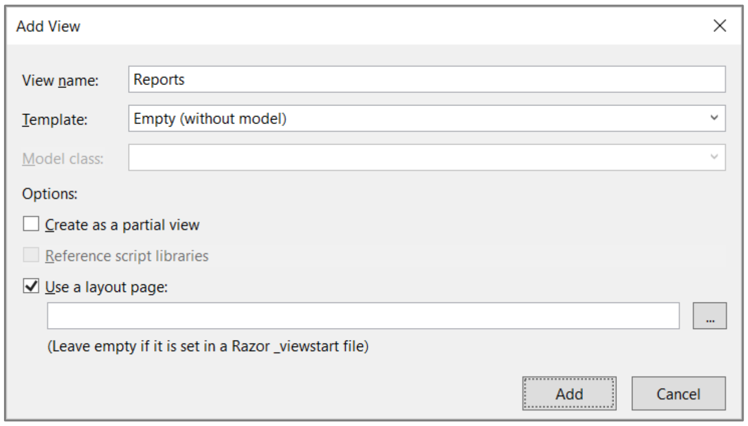
   b) Add a new action method named **Reports** just beneath the **NewReports** method using the following code.

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

  ReportEmbeddingData embeddingData =
      await PbiEmbeddedManager.GetEmbeddingDataForSpecificReport(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.



View name: Reports
Template: Empty (without model)
Model class:
Options:
[ ] Create as a partial view
[ ] Reference script libraries
[✓] Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)

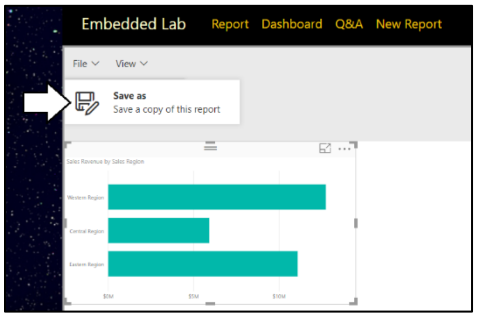
A page break has been inserted here to prevent the following code section from wrapping across pages.

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

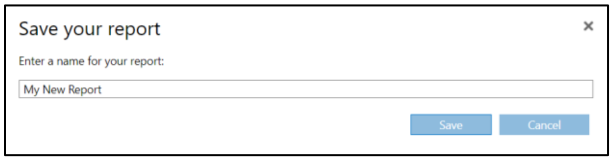
b) Delete any existing code inside **Reports.cshtml** and replace it with the following HTML code.

```
@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>
```

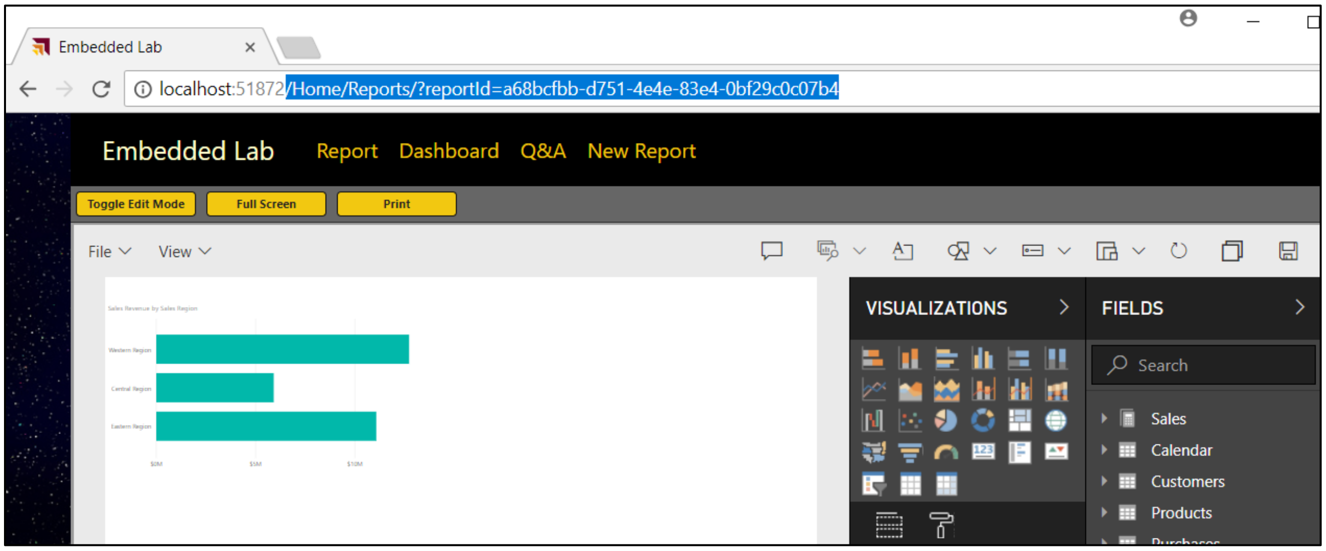
7. Test out the application by running it in the Visual Studio debugger.
   a) Press the **{F5}** key in Visual Studio to begin a new debugging session.
   b) Click the **New Report** link in the top navigation menu and you should see an new empty in design mode.
   c) Add a simple visual to the new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


Save your report
Enter a name for your report:
My New Report
Save Cancel

   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.


localhost:51872/Home/Reports/?reportId=a68bcfbb-d751-4e4e-83e4-0bf29c0c07b4

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

Exercise 5: Add Interactive Support to Set Filters and Apply Bookmarks

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.

8. Add a new method to the **PbiEmbeddingManger** class named **GetNewReportEmbeddingData**.

Exercise 6: Embed a Report using a Custom Layout

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.

9. Add a new method to the **PbiEmbeddingManger** class named **GetNewReportEmbeddingData**.

Congratulations. You have made it to the end of the Power BI Embedded Lab.