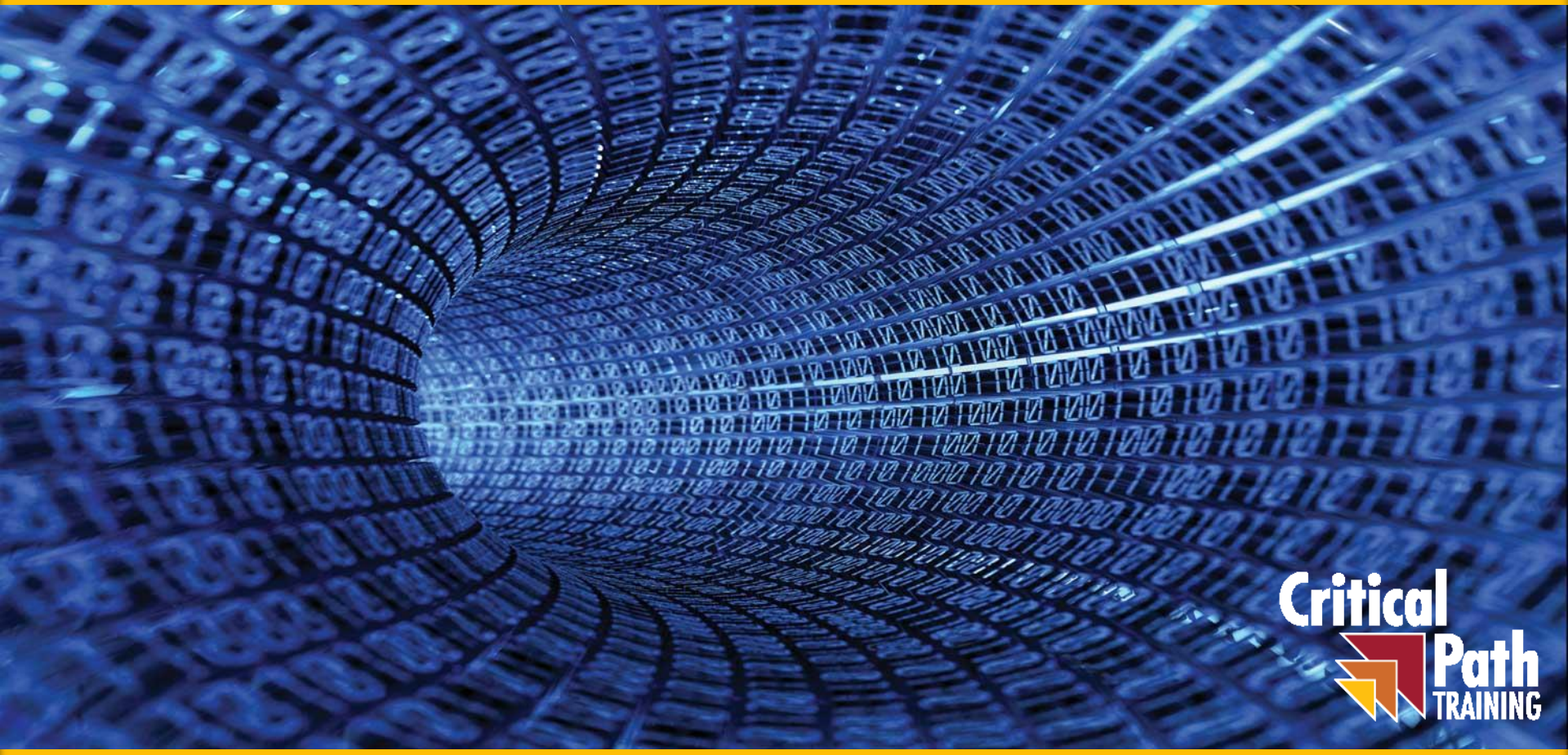
Developing with Power BI Embedding

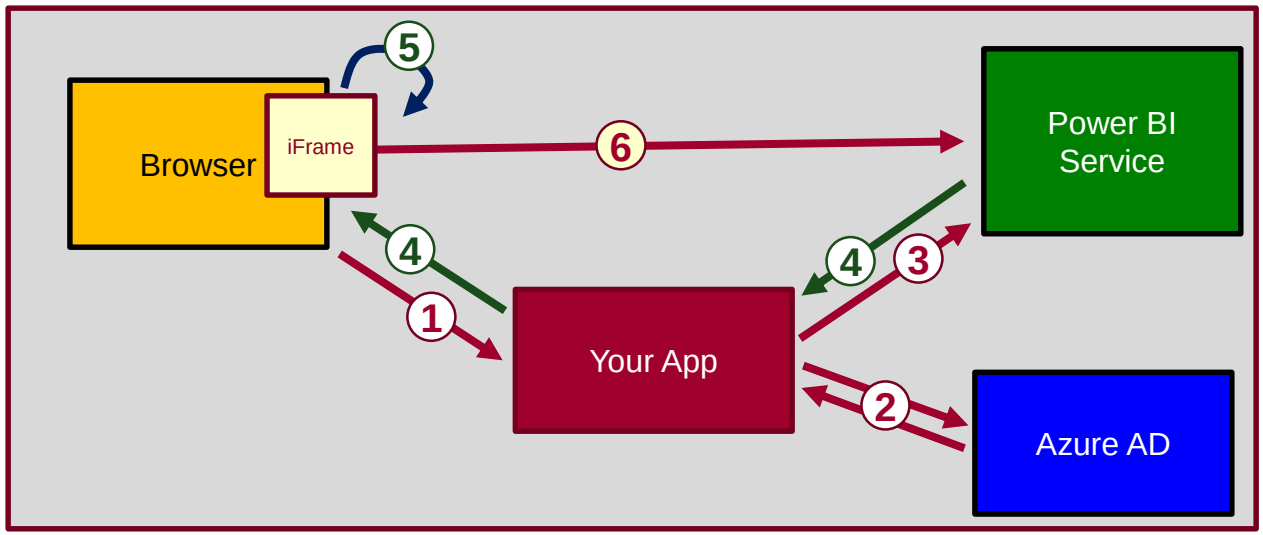


Agenda

- Power BI Embedding Overview
- Embedding with App-Owns-Data Model
- Caching Access Tokens using OWIN Middleware
- Embedding with the User-Owns-Data Model
- Developing with the Power BI JavaScript API

Power BI Embedding – The Big Picture

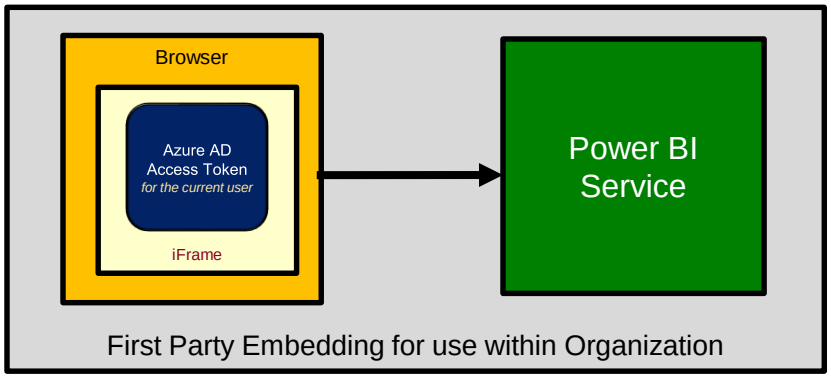
- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded resource
- Embedded resource session created between browser and Power BI service



First Party Embedding vs Third Party Embedding

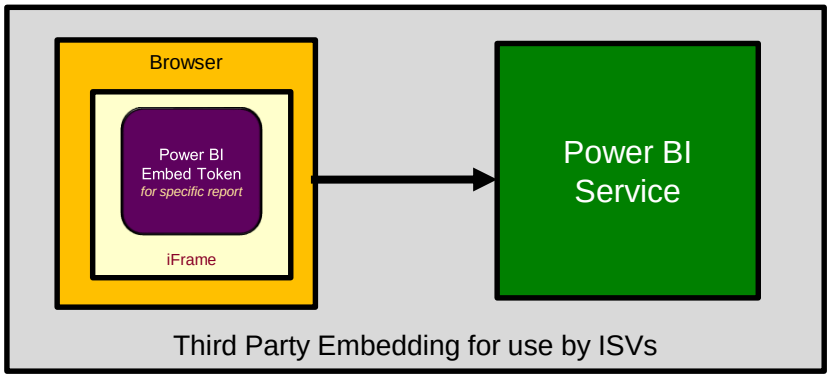
First Party Embedding

- Known as **User-Owns-Data** Model
- All users require a Power BI license
- Useful in corporate environments
- App authenticates as current user
- Your code runs with user's permissions
- User's access token passed to browser



Third Party Embedding

- Known as **App-Owns-Data** Model
- No consumers require Power BI license
- Useful for commercial applications
- App authenticates with app-only identity
- Your code runs with admin permissions
- Embed token passed to browser

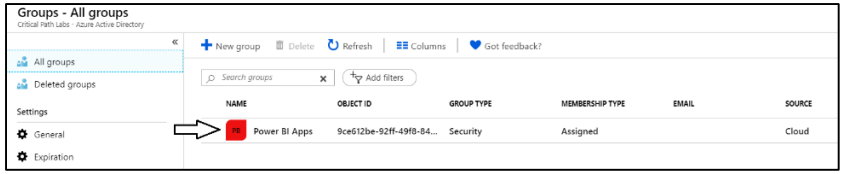


Agenda

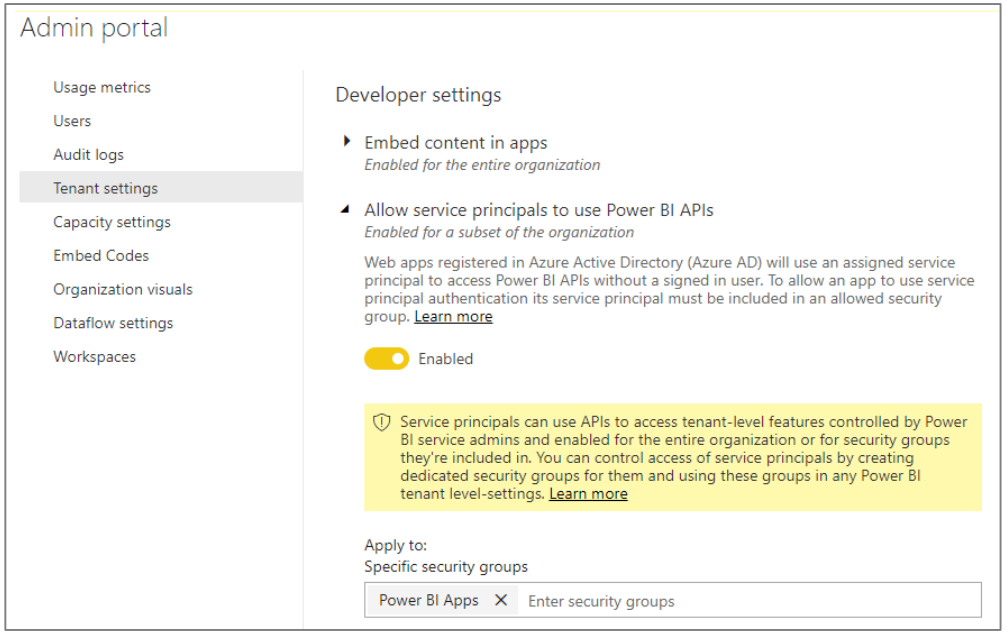
- ✓ Power BI Embedding Overview
- ➢ Embedding with App-Owns-Data Model
- Caching Access Tokens using OWIN Middleware
- Embedding with the User-Owns-Data Model
- Developing with the Power BI JavaScript API

Setting Up for App-Owns-Data – Part 1

- Enable Service Principal Access to Power BI Service API
  - Create an Azure AD security group (e.g. Power BI Apps)

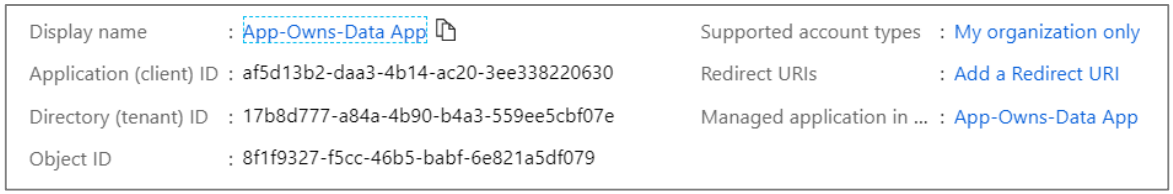


  - Add group to *Power BI Allow service principals to use Power BI APIs*

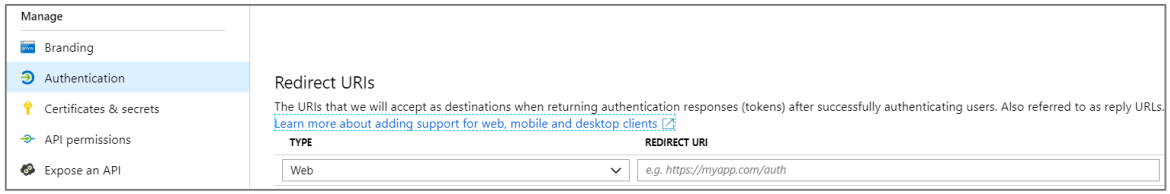


Setting Up for App-Owns-Data – Part 2

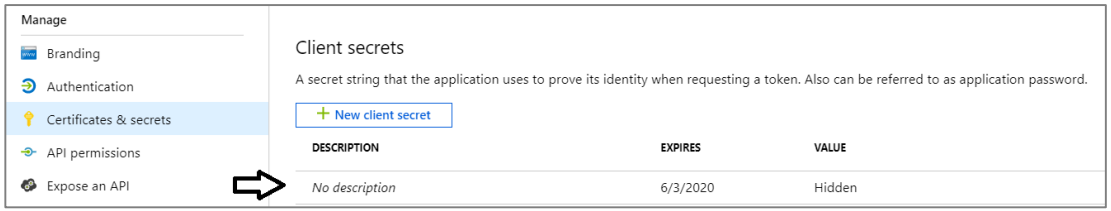
- Create a confidential client in your Azure AD tenant



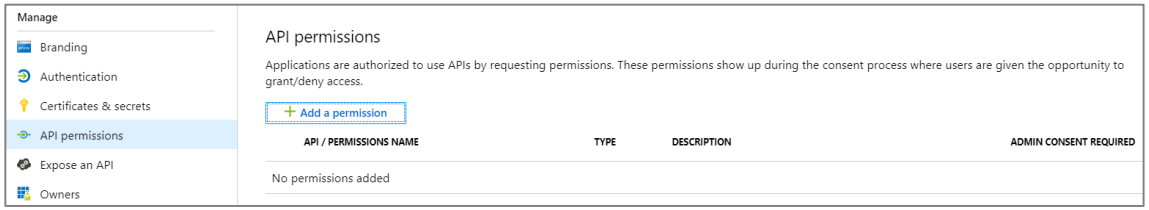
- Configured as `TYPE=Web` and no need for a redirect URL



- Add a client secret or a client certificate

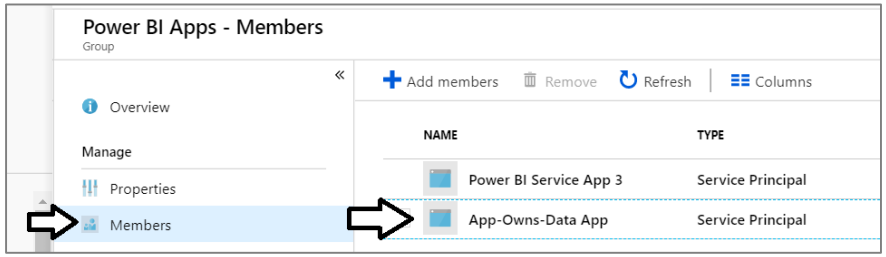


- No need to configure any permissions

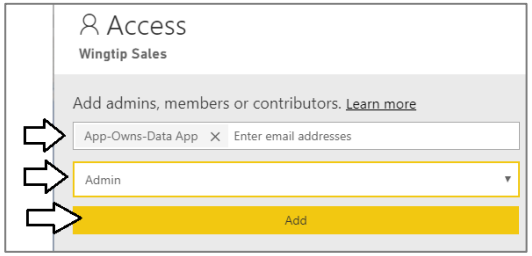


Setting Up for App-Owns-Data – Part 3

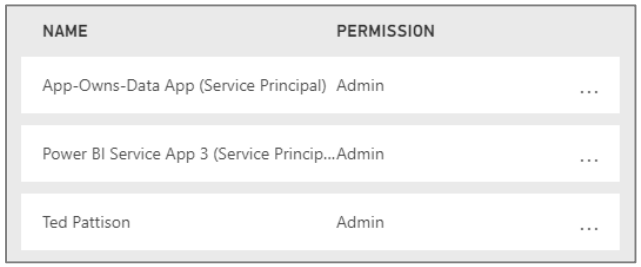
- Add application's service principal in **Power BI Apps** security group



- Configure application's service principal as workspace admin

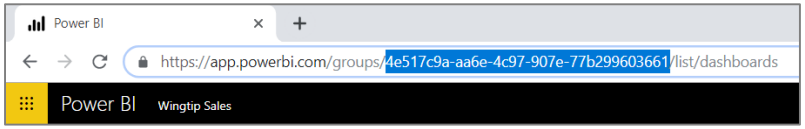


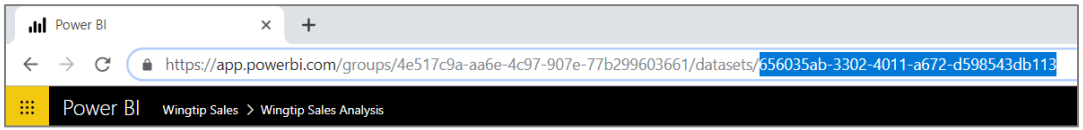
- Service principal should now be workspace admin

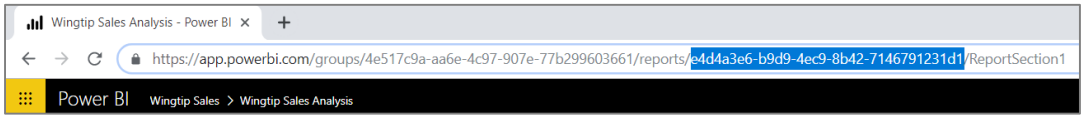


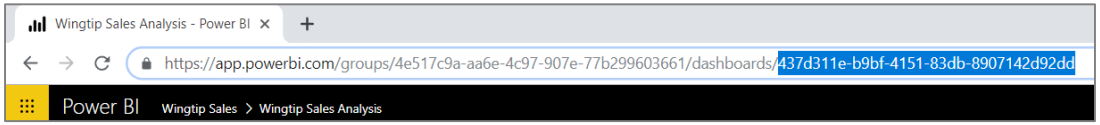
Setting Up for App-Owns-Data – Part 4

- Record the GUIDs for your embeddable resources

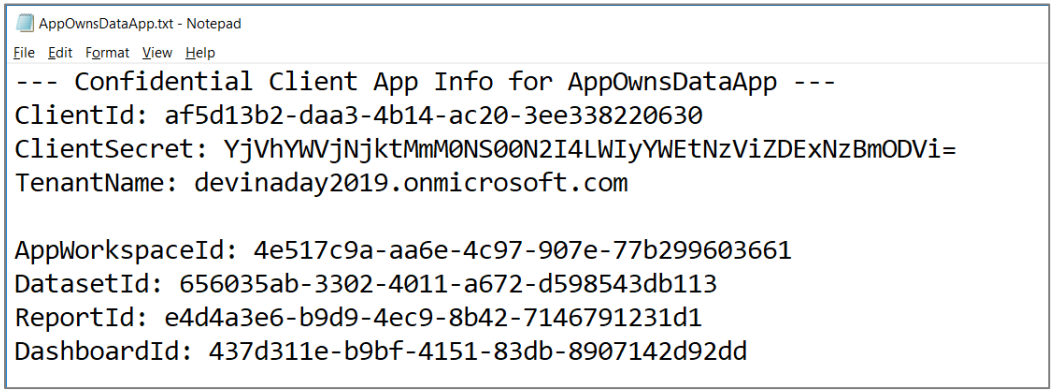








- Gather all configuration data which will be required by application



Generating Embed Tokens

- You can embed reports using an AAD token, but…
  - You might want embed resource using more restricted tokens
  - You might want stay within the bounds of Power BI licensing terms

- You generate embed tokens with the Power BI Service API
  - Each embed token created for one specific resource
  - Embed token provides restrictions on whether user can view or edit
  - Embed token can only be generated in dedicated capacity (semi-enforced)
  - Embed token can be generated to support row-level security (RLS)

```
Report report = reports.Where(r => r.Id == reportId).First();
var generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");
var token = client.Reports.GenerateTokenInGroupAsync(appWorkspaceId,
                                                     report.Id,
                                                     generateTokenRequestParameters).Result;
```

Getting the Data for Report Embedding

```
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
  };

}
```

Getting the Data for Dashboard Embedding

```
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
  };

}
```

Getting the Data for Tile Embedding

```
public static DashboardTileEmbeddingData GetDashboardTileEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var tiles = pbiClient.Dashboards.GetTilesInGroup(workspaceId, dashboardId).Value;

  // retrieve first tile in tiles connection
  var tile = tiles[0];
  var tileId = tile.Id;
  var tileTitle = tile.Title;
  var embedUrl = tile.EmbedUrl;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");
  string embedToken = pbiClient.Tiles.GenerateTokenInGroup(workspaceId,
                                                           dashboardId,
                                                           tileId,
                                                           generateTokenRequestParameters).Token;

  return new DashboardTileEmbeddingData {
    dashboardId = dashboardId,
    TileId = tileId,
    TileTitle = tileTitle,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting Data for New Report Embedding

```
public static NewReportEmbeddingData GetNewReportEmbeddingData() {

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

  PowerBIClient pbiClient = GetPowerBiClient();

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "create",
                                                                                  datasetId: datasetId);

  string embedToken = pbiClient.Reports.GenerateTokenForCreateInGroup(workspaceId,
                                                                      generateTokenRequestParameters).Token;

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

}
```

Getting the Data for Q&A Embedding

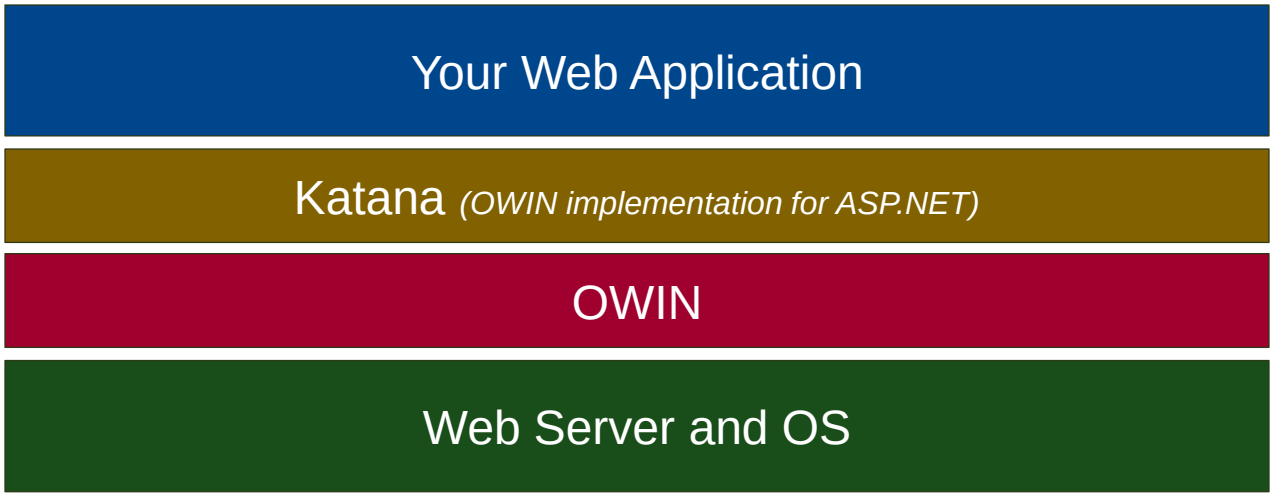
```
public static QnaEmbeddingData GetQnaEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

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

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

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

  string embedToken = pbiClient.Datasets.GenerateTokenInGroup(workspaceId,
                                                              dataset.Id,
                                                              generateTokenRequestParameters).Token;

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

}
```

Agenda

- ✓ Power BI Embedding Overview
- ✓ Embedding with App-Owns-Data Model
- ➢ Caching Access Tokens using OWIN Middleware
- Embedding with the User-Owns-Data Model
- Developing with the Power BI JavaScript API

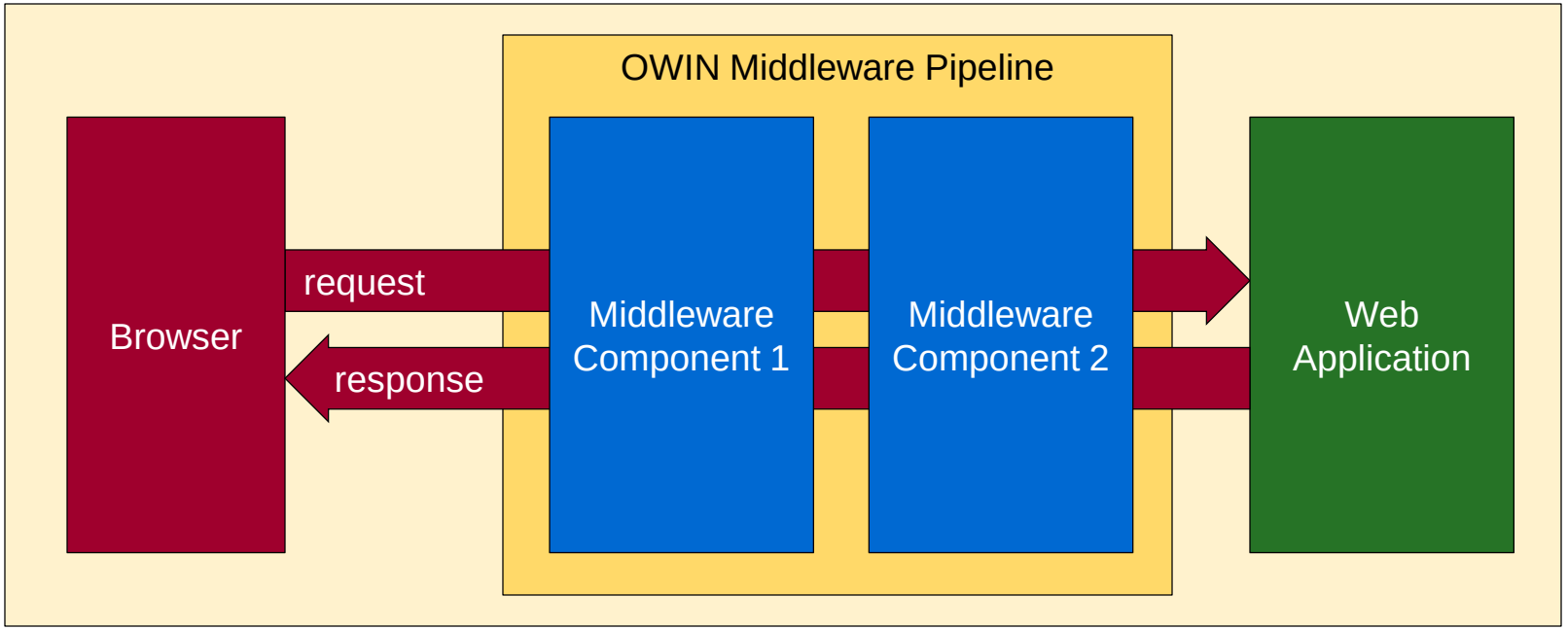
Open Web Interfaces for NET (OWIN)

- OWIN interfaces decouple web server from application
  - OWIN serves to decouple .NET applications from Windows and IIS
  - OWIN promotes the development of smaller modules (middleware)
- Microsoft's Implementation known as Katana
  - Makes it possible to use OWIN with ASP/NET and ASP Core
  - Microsoft provides OWIN-based security middleware



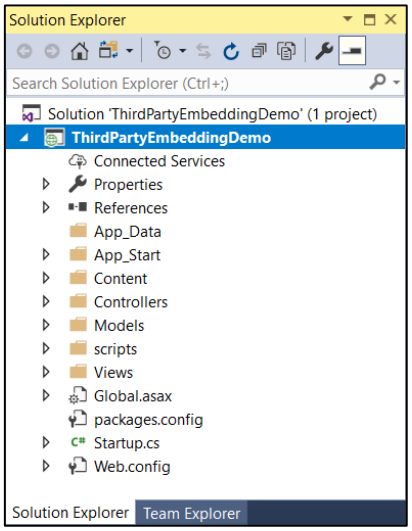
OWIN Middleware Modules

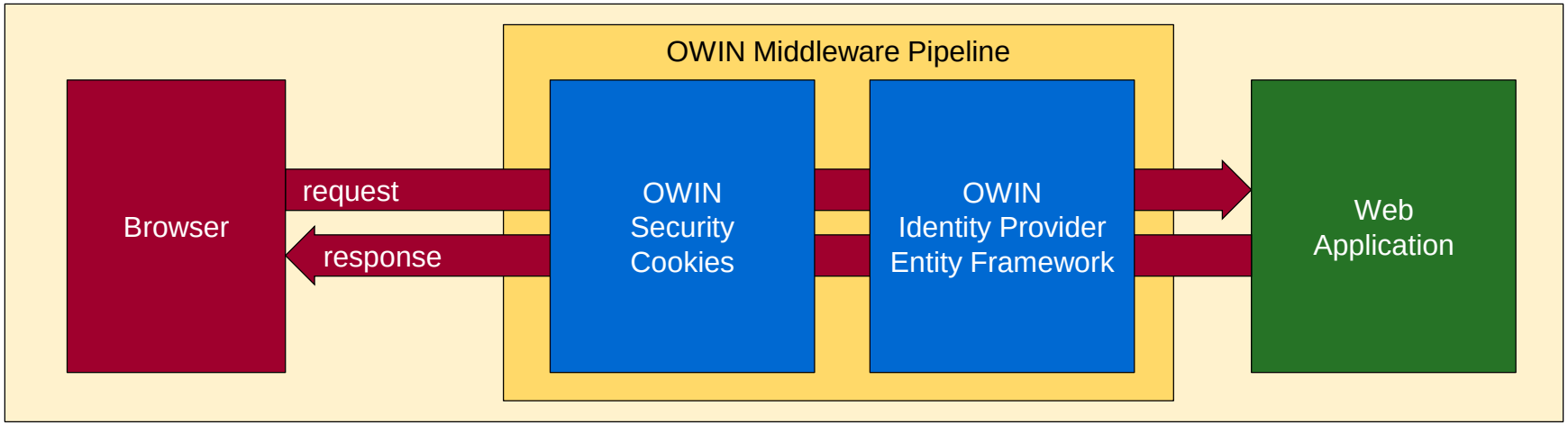
- OWIN create pipeline of middleware components
  - Middleware components added to pipeline on application startup
  - Middleware components pre-process and post process requests
  - Middleware components commonly used to set up authentication



ThirdPartyEmbeddingDemo

- Demonstration key points
  - Entity Framework Identity Provider
  - OWIN Security module



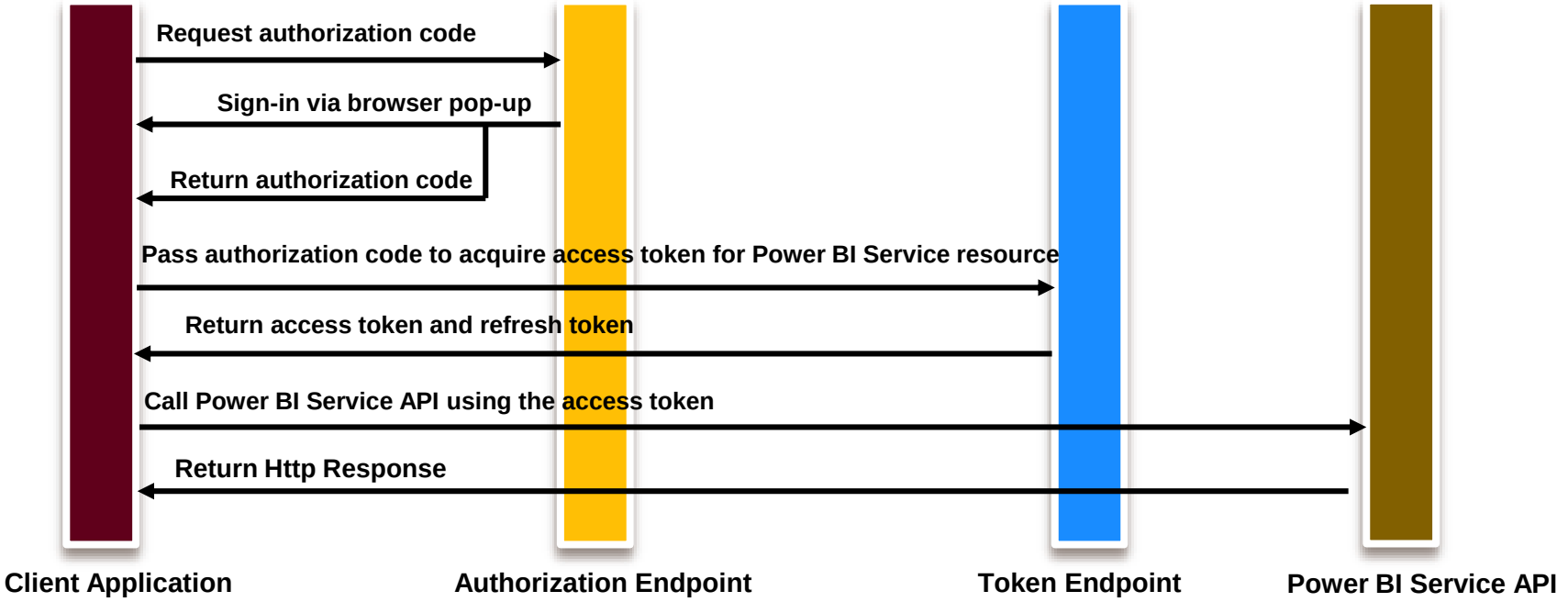


Agenda

- ✓ Power BI Embedding Overview
- ✓ Embedding with App-Owns-Data Model
- ✓ Caching Access Tokens using OWIN Middleware
- ➢ Embedding with the User-Owns-Data Model
- • Developing with the Power BI JavaScript API

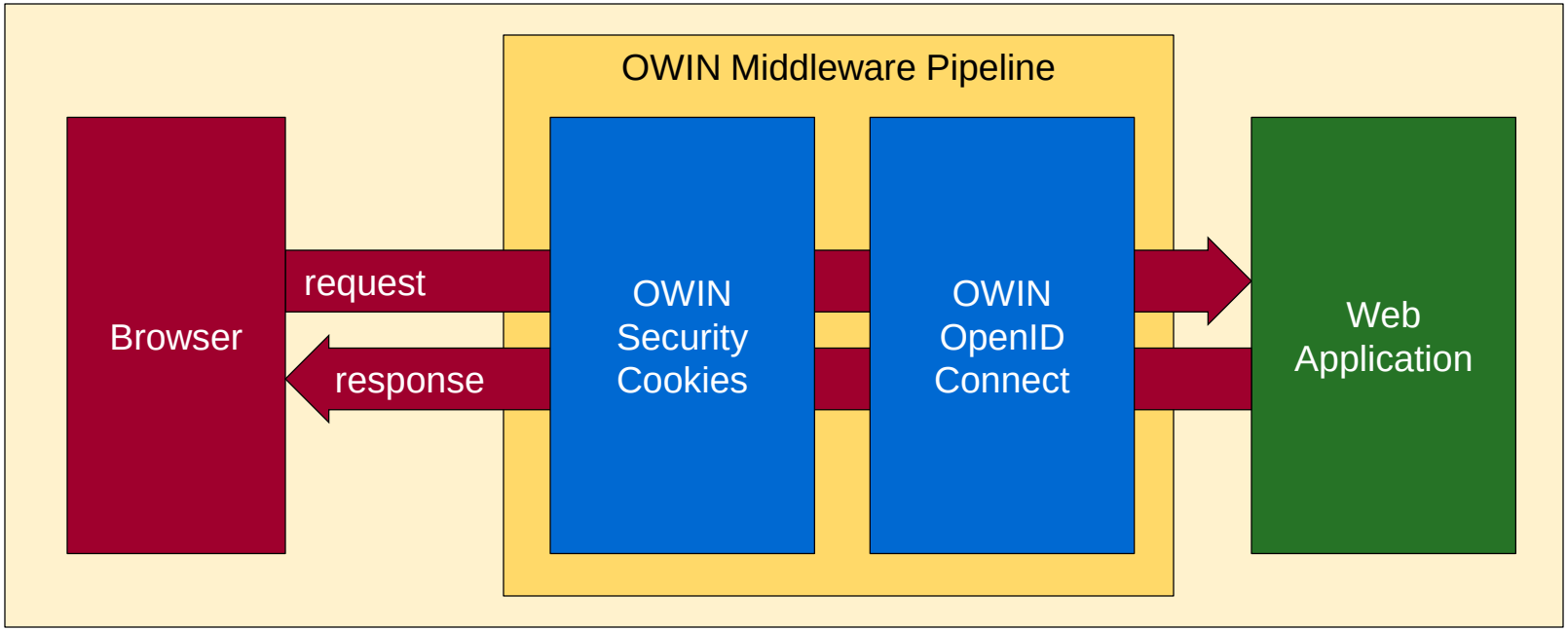
Authorization Code Grant Flow

- Sequence of Requests in Authorization Code Grant Flow
  - Application redirects to AAD authorization endpoint
  - User prompted to sign in using Windows logon page
  - User prompted to consent to permissions (first access)
  - AAD redirects to application with authorization code
  - Application calls to AAD token endpoint to acquire access token



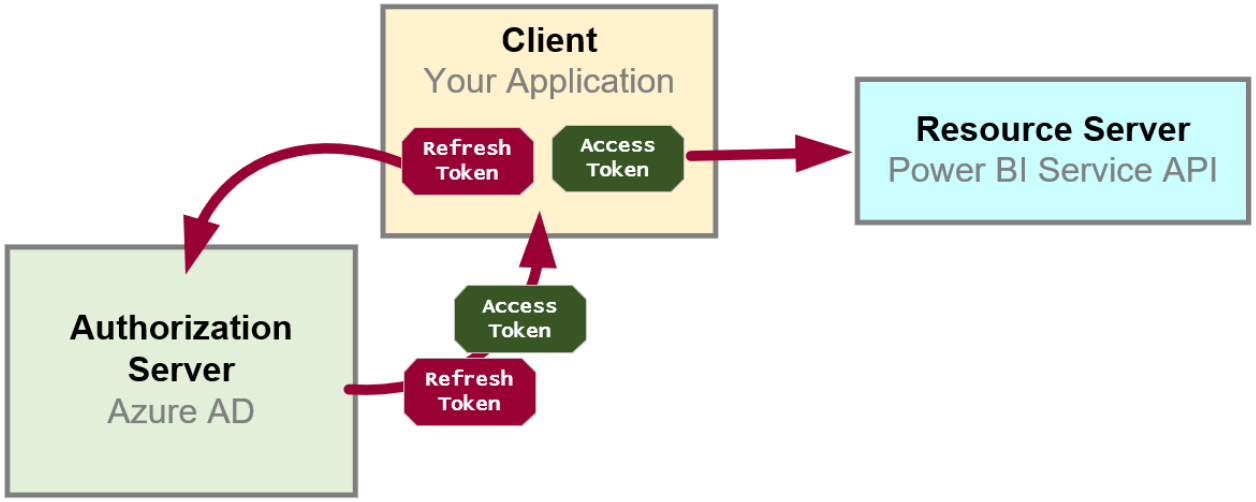
OWIN OpenID Connect Module

- OpenID Connect module used to implement Authorization Code Flow
  - Redirects browsers to authorization endpoint
  - Provides notification when receiving authorization code callback



Token Caching and Refresh Tokens

- OAuth 2.0 provide solution for access token expiration
  - Access tokens have default lifetime of 60 minutes
  - Authorization server passes refresh token along with access token
  - Refresh token used as a credential to redeem new access token
  - Refresh token default lifetime is 14 days (max 90 days)
  - Refresh tokens often persistent in database or browser storage
  - ADAL and MSAL both offer built-in support to mange token caching




DEMO
DailyReporterPersonal

Understanding Implicit Flow

- Single Pages Applications (SPAs) are public clients
  - SPA not able to keep secrets such as application secret
  - No ability to execute server-to-server calls
  - SPAs cannot implement authorization code flow

- Implicit flow requires lowering security bar for SPAs
  - Azure AD application must be configured to allow implicit flow
  - Allows SPAs to retrieve access tokens
  - Access token returned to browser in URL fragment

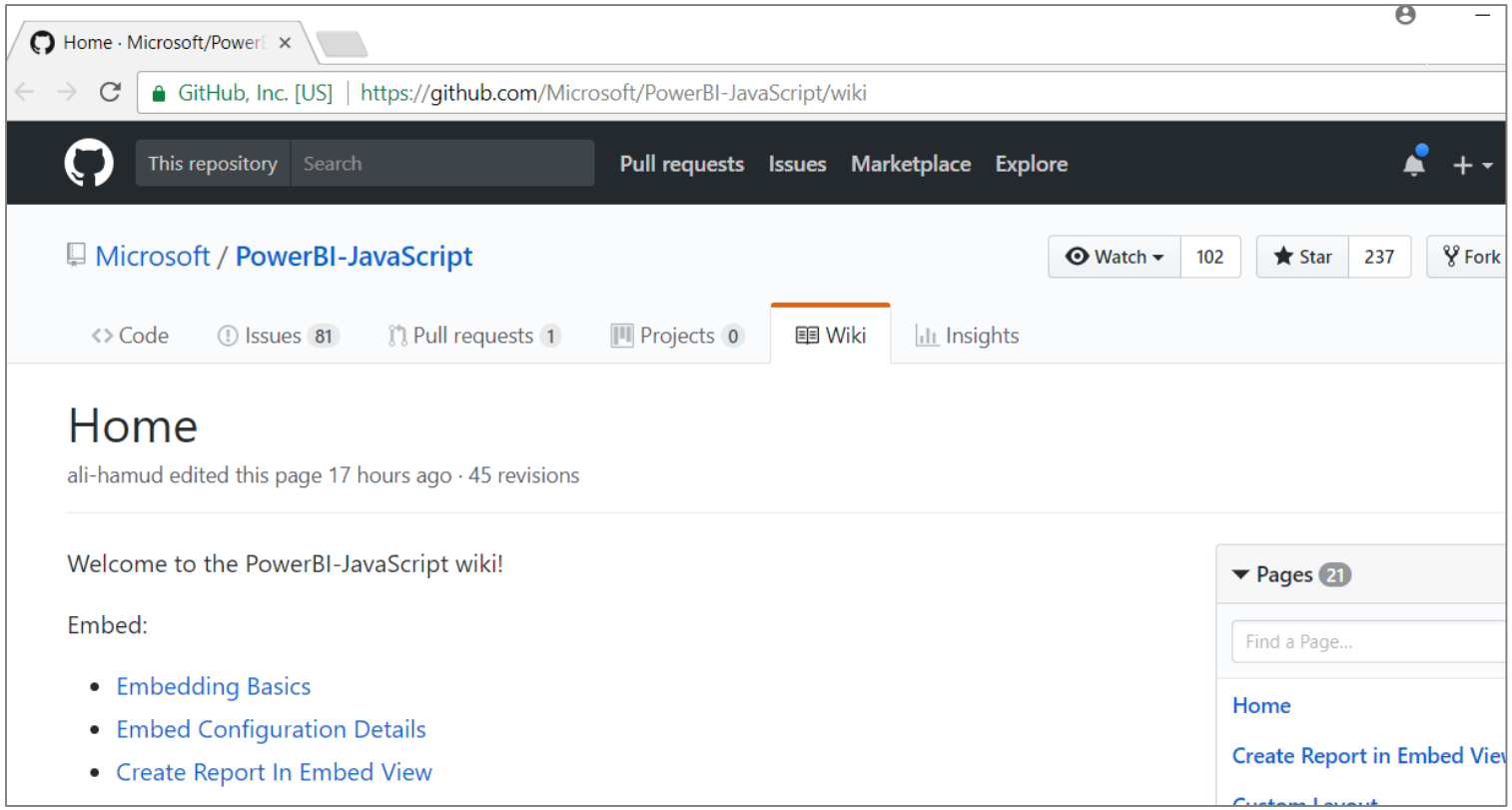

DEMO
PowerBIDaySPA

Agenda

- ✓ Power BI Embedding Overview
- ✓ Embedding with App-Owns-Data Model
- ✓ Caching Access Tokens using OWIN Middleware
- ✓ Embedding with the User-Owns-Data Model
- ➢ Developing with the Power BI JavaScript API

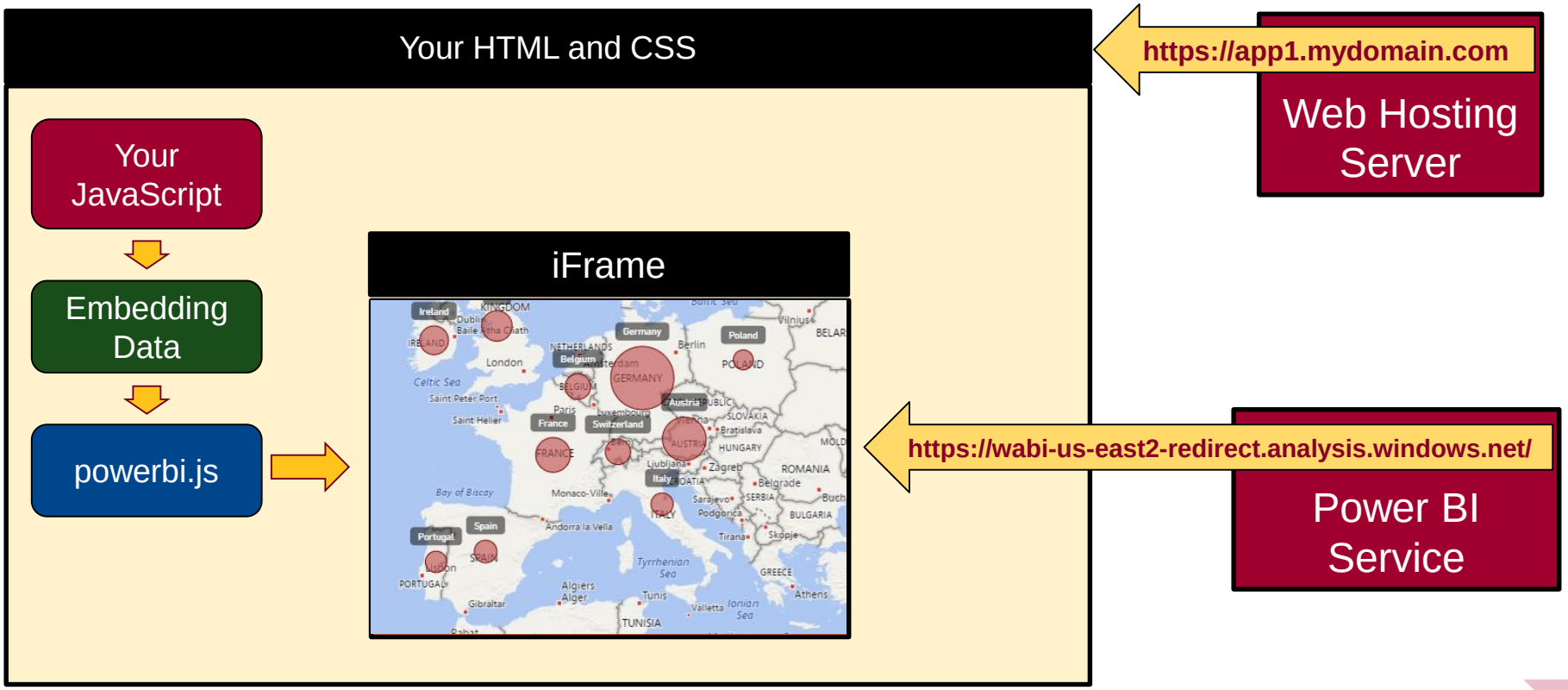
Power BI JavaScript API (powerbi.js)

- Power BI JavaScript API used to embed resources in browser
  - GitHub repo at https://github.com/Microsoft/PowerBI-JavaScript/wiki
  - GitHub repository contains code, docs, wiki and issues list



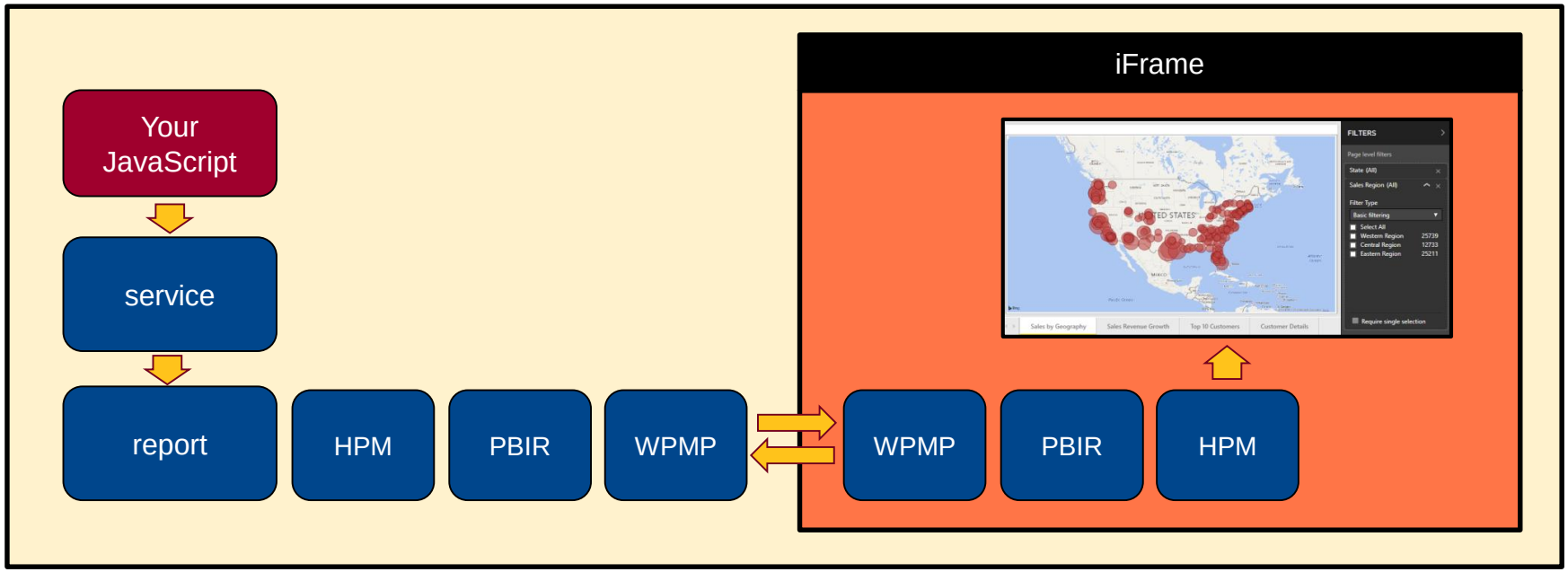
Report Embedding Architecture

- Embedding involves creating an iFrame on the page
  - PBIJS transparently creates iFrame and sets source to Power BI Service
  - ***The iFrame and hosting page originate from different DNS domains***



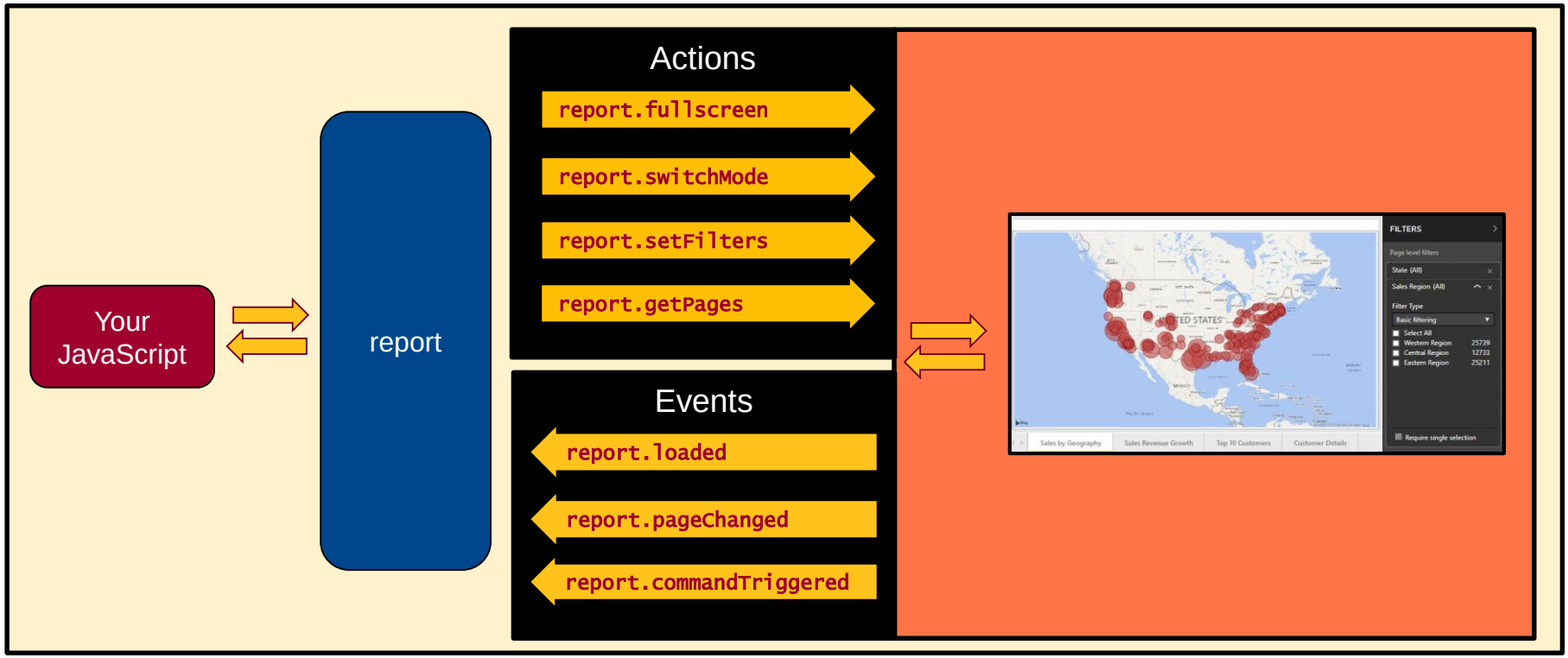
Post Message Communications Flow

- 4 extra libraries used communicate with report in iFrame
  - window-post-message-proxy (WPMP)
  - http-post-message (HPM)
  - powerbi-router (PBIR)
  - powerbi-models (PBIM)



A Promise-based Programming Model

- Design of PBIJS simulates HTTP protocol
  - Creates more intuitive programming model for developers
  - Programming based on asynchronous requests and promises
  - Embedded objects programmed using actions and events

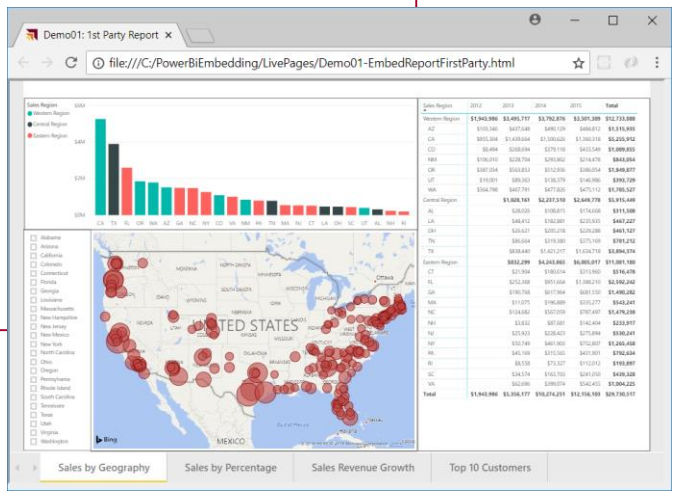


Hello World with Power BI Embedding

- **powerbi.js** library provides **powerbi** as top-level service object
  - You call **powerbi.embed** and pass **configuration** object with access token
  - **models** object available to supply configuration settings
  - **configuration** object sets **tokenType** to either **TokenType.Embed** or **TokenType.Aad**

```
// data required for embedding Power BI report
var embedReportId = "ba274ba0-93be-4e53-af65-fdc8a559c557";
var embedUrl = "https://app.powerbi.com/reportEmbed?reportId=ba274ba0-93be-4e53-af65-fdc8a559c557&groupId=7f4
var accessToken = "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlRpb0d5d3dsaHZkRmJYWjgxM1dwUGF5OUFsVSIsImtpZC

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

// create embed configuration object
var config = {
  type: 'report',
  id: embedReportId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Aad
};

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

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

First party embedding



Embedded Report Configuration Options

- **permissions** determines what permissions are given to user on resource
- **viewMode** determines when report opens in read-only view or edit view
- **pageView** determines how reports scales to fit embed container element

```
var models = window['powerbi-client'].models;

var config: embed.IEmbedConfiguration = {
  type: 'report',
  id: reportId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Embed,
  permissions: models.Permissions.All,
  viewMode: models.ViewMode.Edit,
  pageView: "fitToWidth",
  settings: {
    filterPaneEnabled: false,
    navContentPaneEnabled: false
  }
};
```

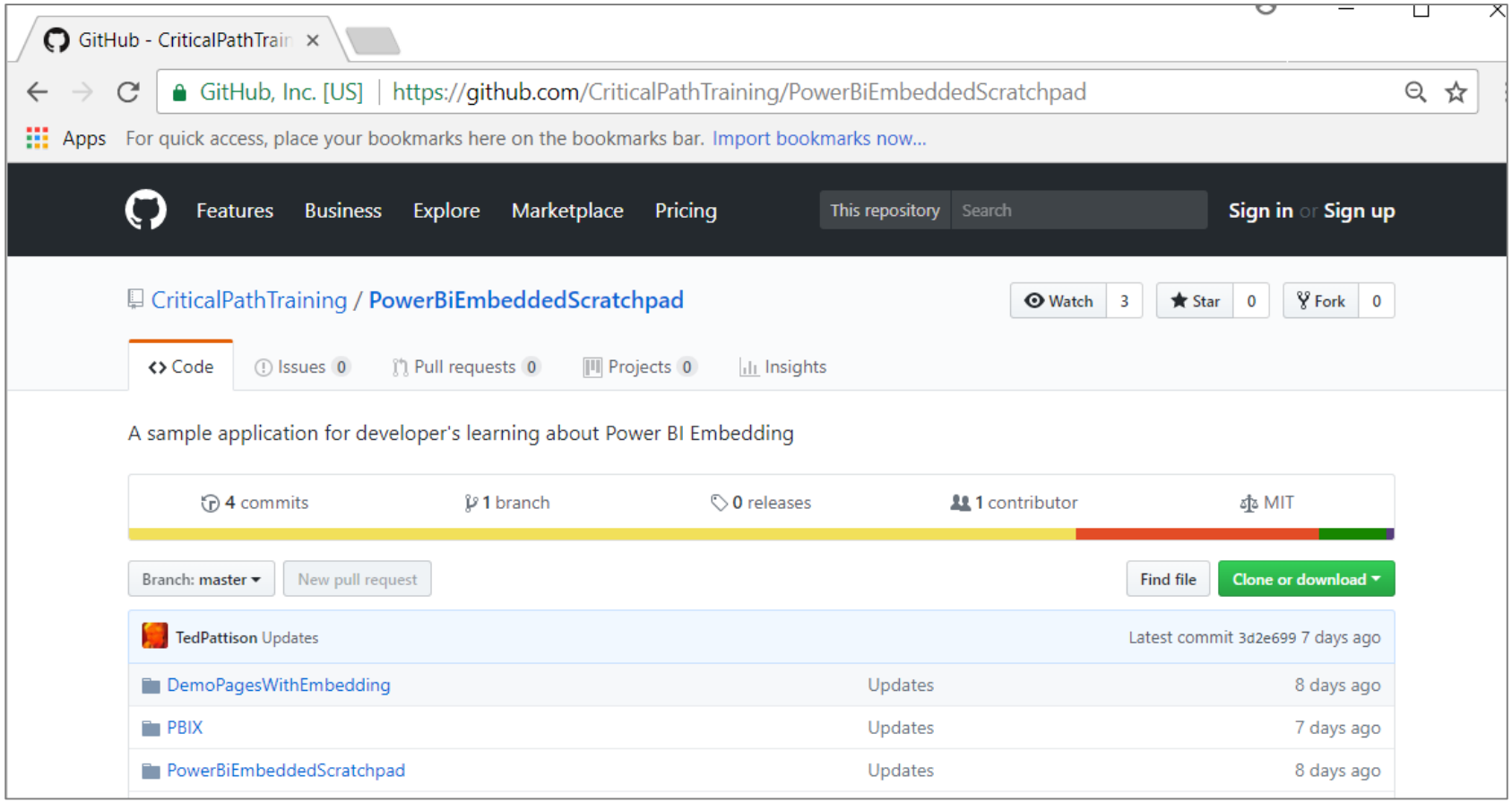
Read: Allows view report only.
ReadWrite: Allows view, Edit and Save report.
Copy: Allows Save a copy of a report using Save As.
Create: Allows creating a new report.
All: Allows everything.

View - Opens report in View mode.
Edit - Opens report in Edit mode.

fitToWidth: Fit to width of host HTML element.
oneColumn: Opens in single column.
actualSize: Actual size as designed in report

PowerBiEmbeddedScratchpad Sample

https://github.com/CriticalPathTraining/PowerBiEmbeddedScratchpad



Handling Report Events

```
var report = powerbi.embed(embedContainer, config);

var pages;

report.on('loaded', function () {
  // call getPages with callback
  report.getPages().then(
    function (reportPages) {
      pages = reportPages;
      // call method to load pages into nav menu
      loadReportPages(pages);
    });
});

var loadReportPages = function (pages) {
  for (var index = 0; index < pages.length; index++) {
    // determine which pages are visible and not hidden
    if (pages[index].visibility == 0) { // 0 means visible and 1 means hidden
      var reportPageDisplayName = pages[index].displayName;
      pageNavigation.append($("<li>")
        .append($('<a href="javascript:;" >')
        .text(pages[index].displayName))
        .click(function (domEvent) {
          var targetPageName = domEvent.target.textContent;
          // get target page from pages collection
          var targetPage = pages.find(function (page) { return page.displayName === targetPageName; });
          // navigate report to target page
          targetPage.setActive();
        }));
    }
  }
}
```

```
▼ Report i
  ▼ allowedEvents: Array(12)
      0: "loaded"
      1: "saved"
      2: "rendered"
      3: "saveAsTriggered"
      4: "error"
      5: "dataSelected"
      6: "filtersApplied"
      7: "pageChanged"
      8: "commandTriggered"
      9: "swipeStart"
      10: "swipeEnd"
      11: "bookmarkApplied"
```

Embedding a New Report

```
// Get data required for embedding
var embedWorkspaceId= "7f4576c7-039a-472f-b998-546a572d5da2";
var embedDatasetId = "b4a48602-71da-42b2-8cf5-44d35b2ac70b";
var embedUrl = "https://app.powerbi.com/reportEmbed?groupId=7f4576c7-039a-472f-b998-546a5
var accessToken = "H4sIAAAAAAAAEAB2Wxw6OCA6E3-W_shIZmpXmQE5NztzIOWdG--7bM3dbsj67qvz3HzN5-i

// 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);
```

New Report with SaveAs Redirect

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

// this event fires whenever user runs save or SaveAs command on a new report
newReport.on("saved", function (event) {

  // get ID and name of new report
  var newReportId = event.detail.reportObjectId;
  var newReportName = event.detail.reportName;

  // set new title for browser window
  document.title = newReportName;

  // reset report container element
  powerbi.reset(embedContainer);

  config = {
    type: 'report',
    id: newReportId,
    embedUrl: "https://app.powerbi.com/reportEmbed?reportId=" + newReportId + "&groupId=" + embedWorkspaceId,
    accessToken: accessToken,
    tokenType: models.TokenType.Aad,
    permissions: models.Permissions.All,
    viewMode: models.ViewMode.Edit,
  };

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

Embedding the Q&A Experience

```
// Get data required for embedding
var datasetId = "b4a48602-71da-42b2-8cf5-44d35b2ac70b";
var embedUrl = "https://app.powerbi.com/qnaEmbed?groupId=7f4576c7-039a-472f-b998-546a57
var accessToken = "H4sIAAAAAAAAEAC2Wx6rFDI6E3-XfeuA4h4FeOOecvXPOObuZd58L3XuBpK8Klf79j5W-

// 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"
};

var embedContainer = document.getElementById('embedContainer');

var embeddedObject = powerbi.embed(embedContainer, config);
```

Configuration Options

```
// 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.View,
  settings: {
    filterPaneEnabled: false,
    navContentPaneEnabled: true,
    localeSettings: { language: "en", formatLocale: "es" },
    background: models.BackgroundType.Transparent
  }
};

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

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


DEMO
The Power BI Embedded Scratchpad App

Summary

- ✓ Power BI Embedding Overview
- ✓ Embedding with App-Owns-Data Model
- ✓ Caching Access Tokens using OWIN Middleware
- ✓ Embedding with the User-Owns-Data Model
- ✓ Developing with the Power BI JavaScript API