

# Developing with Power BI Embedding



# Agenda

- Power BI Embedding Fundamentals
  - App Workspaces and Dedicated Capacities
  - Authentication with Azure Active Directory
  - Programming with Power BI Service API
  - Working with Embeddable Resources
  - Embedding with Power BI JavaScript API



# The Power BI Service

- Provides cloud-based foundation for Power BI platform
  - Accessible with browser through <https://app.powerbi.com>
  - Accessible through Power BI mobile apps
  - Accessible to developers through Power BI Service API



# The Power BI Service API

- The Power BI Service API goes by other names
  - The Power BI REST API
  - The Power BI API

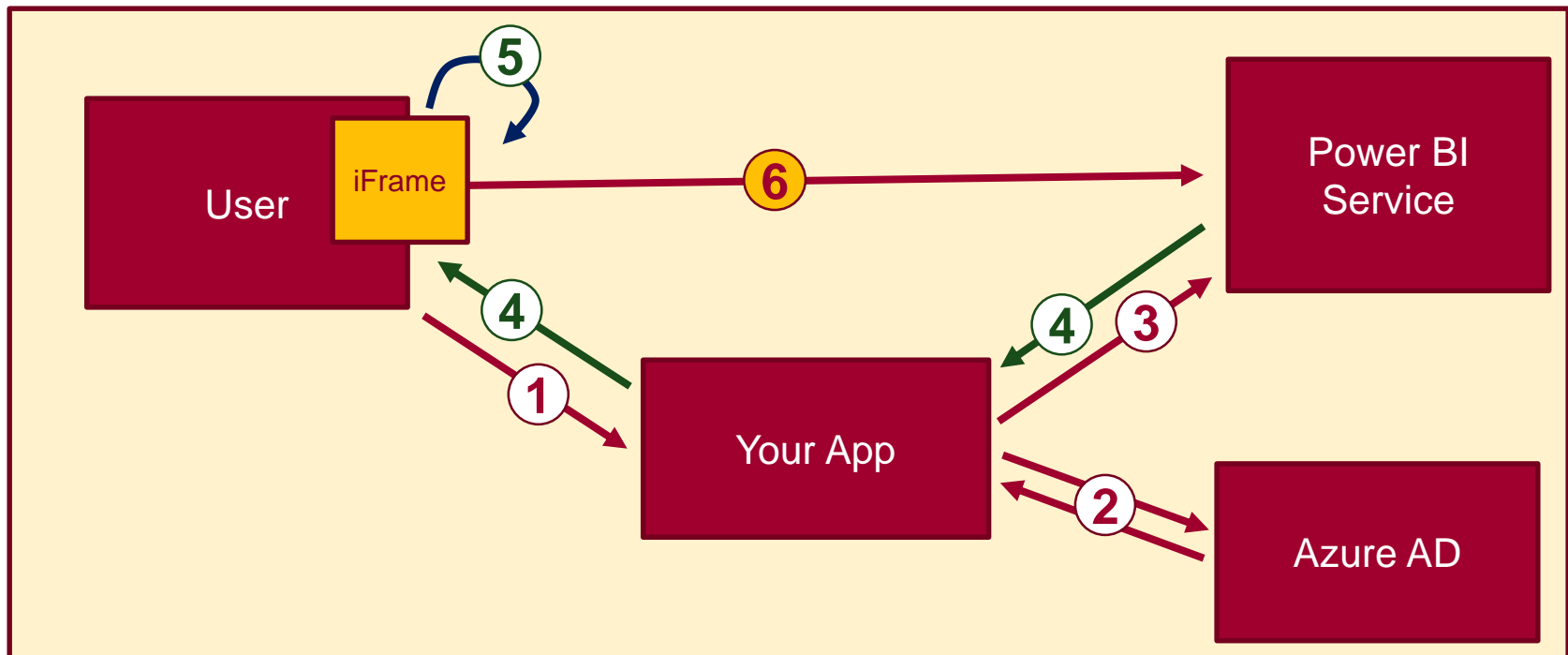


- Using the Power BI Service API
  - Accessible by making direct REST calls against service
  - Accessible by using Assembly DLL that abstracts away REST calls
  - Assembly DLL is named **Microsoft.PowerBI.Api.dll**
  - Assembly DLL part of NuGet package (**Microsoft.PowerBI.Api**)
  - Calling service requires authentication with Azure Active Directory



# Power BI Embedding – The Big Picture

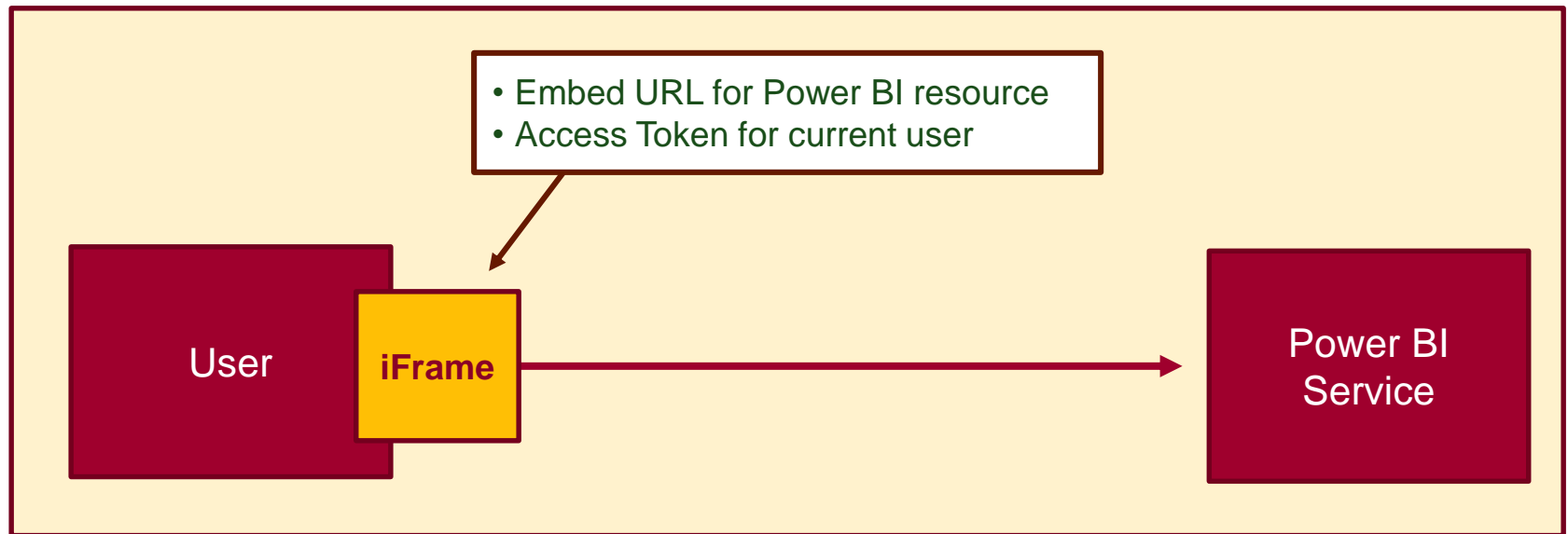
1. User launches your app using a browser
2. App authenticates with Azure Active Directory and obtains access token
3. App uses access token to call to Power BI Service API
4. App retrieves data for embedded resource and passes it to browser.
5. Client-side code uses Power BI JavaScript API to create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





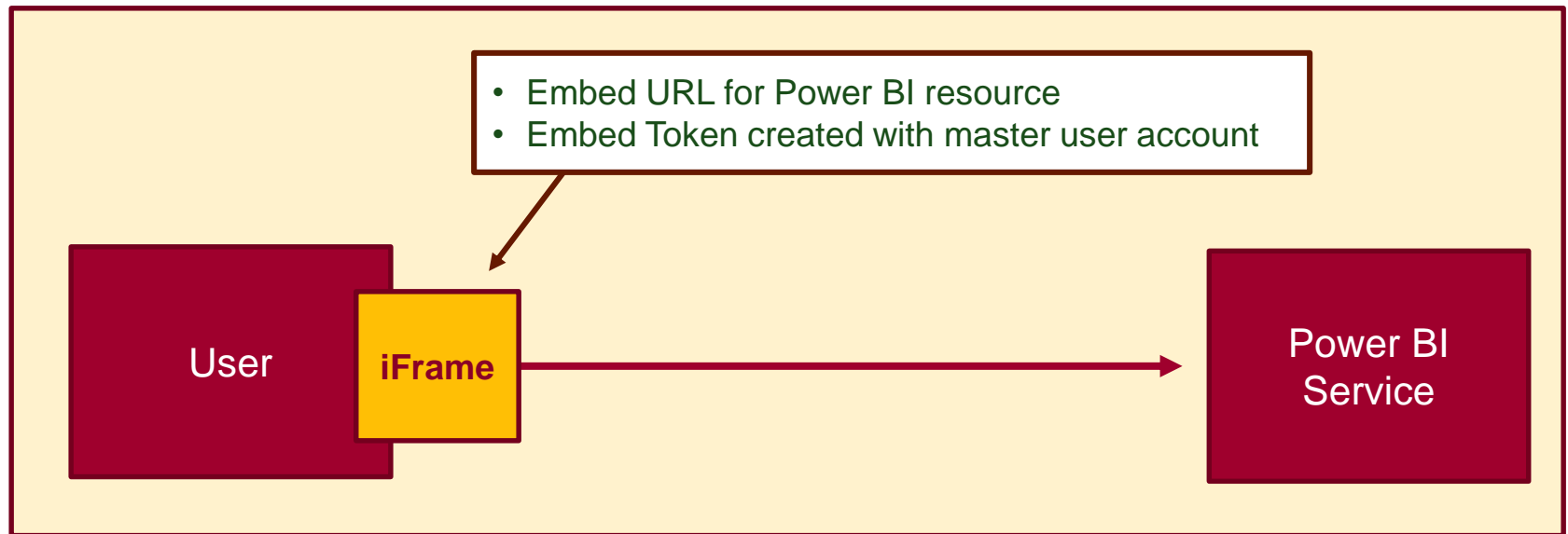
# First Party Embedding

- App authenticates current user with Azure AD
  - Your code accesses Power BI Service as current user
  - Embedding requires Azure AD access token for user
  - User requires Azure AD account and Power BI license
  - Your code has access to whatever user has access to



# Third Party Embedding

- App authenticates using Master User Account
  - Your code accesses Power BI Service as master user
  - Embedding uses embed token instead of access token
  - Users don't need AAD accounts and Power BI licenses
  - Your code has access to whatever master has access to



# First Party vs Third Party Embedding

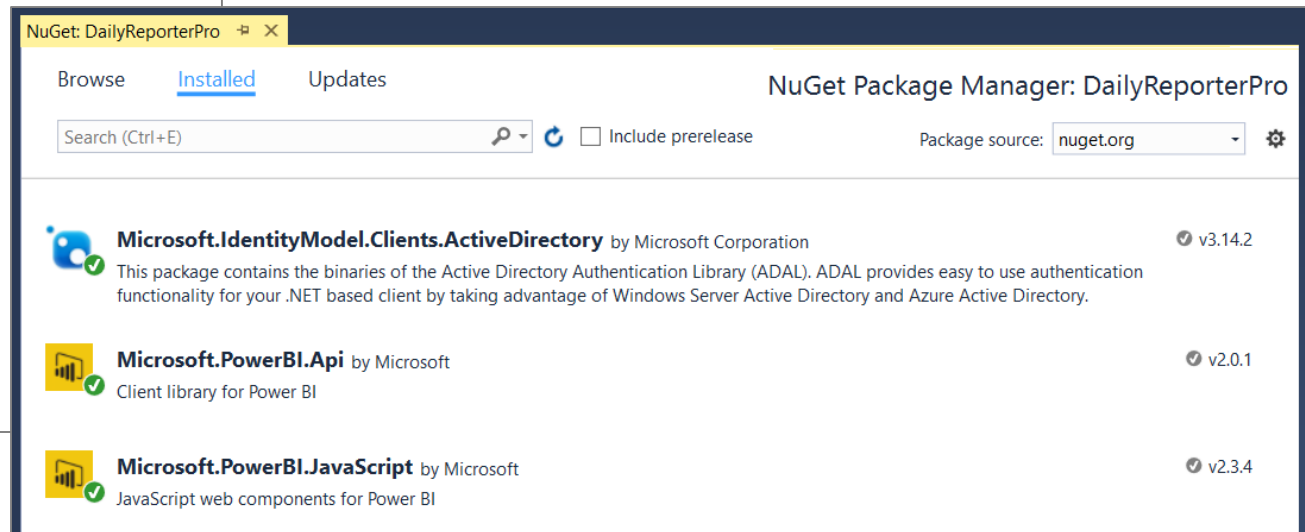
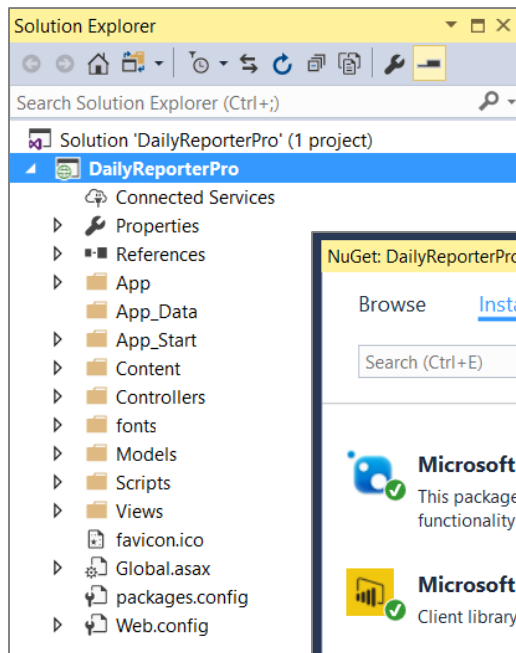
- What scenarios use first party embedding?
  - Organizations where users have Power BI licenses
  - Embedding Power BI reports in SharePoint and Teams
  - Development should go beyond out-of-box experience
- What scenarios use third party embedding?
  - Scenarios where users don't have Power BI licenses
  - Applications which have custom identity providers
  - Applications which use identity provider other than AAD





# NuGet Packages Required in MVC Project

- NuGet Packages used in DailyReporterPro sample app
  - Azure Active Directory Library (ADAL) for .NET
  - Power BI Service API
  - Power BI JavaScript API





**DEMO**

# **The Daily Reporter Pro Sample App**

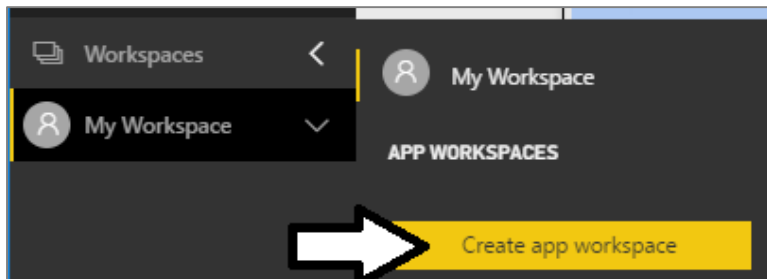
# Agenda

- ✓ Power BI Embedding Fundamentals
- App Workspaces and Dedicated Capacities
  - Authentication with Azure Active Directory
  - Programming with Power BI Service API
  - Working with Embeddable Resources
  - Embedding with Power BI JavaScript API



# Understanding App Workspaces

- App workspaces required for 3<sup>rd</sup> party embedding
  - You access working under identity of master user account
  - Master user account must be configured as app workspace admin
  - App workspace must be associated with dedicated capacity



A screenshot of the 'Edit workspace' configuration page. It includes fields for 'Name' (Sales Team), 'Privacy' (Private - Only approved members can see what's inside), and 'Members can edit Power BI content'. Below these are 'Workspace members' with an 'Add' button and a list of users. At the bottom, there is a section for 'Advanced' settings, including 'Dedicated capacity' (On) and a dropdown to 'Choose an available dedicated capacity for this workspace' (capacity0012).

Workspace members	
pbimasteruser@sharepointconfessio...	Admin
tedp@sharepointconfessions.onmicro...	Admin

Master User Account

Dedicated Capacity



# Dedicated Capacities

- Power BI workspaces run in two possible environments
  - Shared capacities
  - Dedicated capacities
- Dedicated capacity required for third party embedding
  - Customer pays capacity-based fee for processors cores and RAM
  - No need to pay Microsoft for user licenses
- Dedicated capacities come in two flavors
  - Power BI Premium capacities purchased through Office 365 SKU
  - Power BI Embedded capacities purchased through Azure SKU



# P SKUs (P is for Premium)

- All-in version which allows greatest flexibility
  - Used to provide free users with access to apps and content
  - Used to support 1<sup>st</sup> party and 3<sup>rd</sup> party embedding
  - The more you pay per month, the more resources you get

Name	Virtual cores	Memory (GB)	Peak renders/hr	Cost
P1	8	25	1-300	\$4,995/mo
P2	16	50	301-600	\$9,995/mo
P3	32	100	601-1,200	\$19,995/mo





# Managing Power BI Premium Capacities

The screenshot displays the Power BI Admin portal interface. On the left is a dark sidebar with navigation options: Favorites, Recent, Apps, Shared with me, Workspaces, and My Workspace (selected). The main content area is titled 'Admin portal' and contains a list of settings: Usage metrics, Users, Audit logs, Tenant settings, Capacity settings (highlighted with a yellow bar and an arrow), Embed Codes, and Organization visuals. To the right of the 'Capacity settings' link, there are tabs for 'Power BI Premium' (selected) and 'Power BI Embedded'. Below the 'Power BI Premium' tab, a section titled 'V-CORES' shows a progress bar indicating '8 of 8 used' out of '8 v-cores' available, with '0 available' shown in grey. A link 'Learn more about capacity sizes' and a button 'Set up new capacity' are also present. At the bottom, a table titled 'PREMIUM CAPACITIES' lists existing capacities.

CAPACITY NAME	CAPACITY ADMINS	ACTIONS	SKU	V-CORES
Capacity P1 #1 8GB budget, da... + 3 more			P1	8



# Associating Workspaces with Capacities

## Edit workspace

Name

Privacy

Private - Only approved members can see what's inside

Members can edit Power BI content ▾

Workspace members

Enter email addresses

Add

chass@powerbimvps.onmicrosoft.com	Admin	▾	🗑
criticalpathuser123@powerbimvps.on...	Admin	▾	🗑
tedp@powerbimvps.onmicrosoft.com	Admin	▾	🗑

Advanced ^

Dedicated capacity ⓘ

☒ On

Choose an available dedicated capacity for this workspace

Capacity P1 #1 8GB model ▾



# EM SKUs (EM is for Embedded)

- SKUs for embedding into SaaS Applications
  - SaaS applications include SharePoint and Teams
  - Supports 1<sup>st</sup> party and 3<sup>rd</sup> party embedding
  - Does not provide user with access to PowerBI.com

Name	Virtual cores	Memory (GB)	Peak renders/hr	Cost
EM1	1	3	1-300	\$625/mo
EM2	2	5	301-600	\$1,245/mo
EM3	4	10	601-1,200	\$2,495/mo



# A SKUs (A is for Azure)

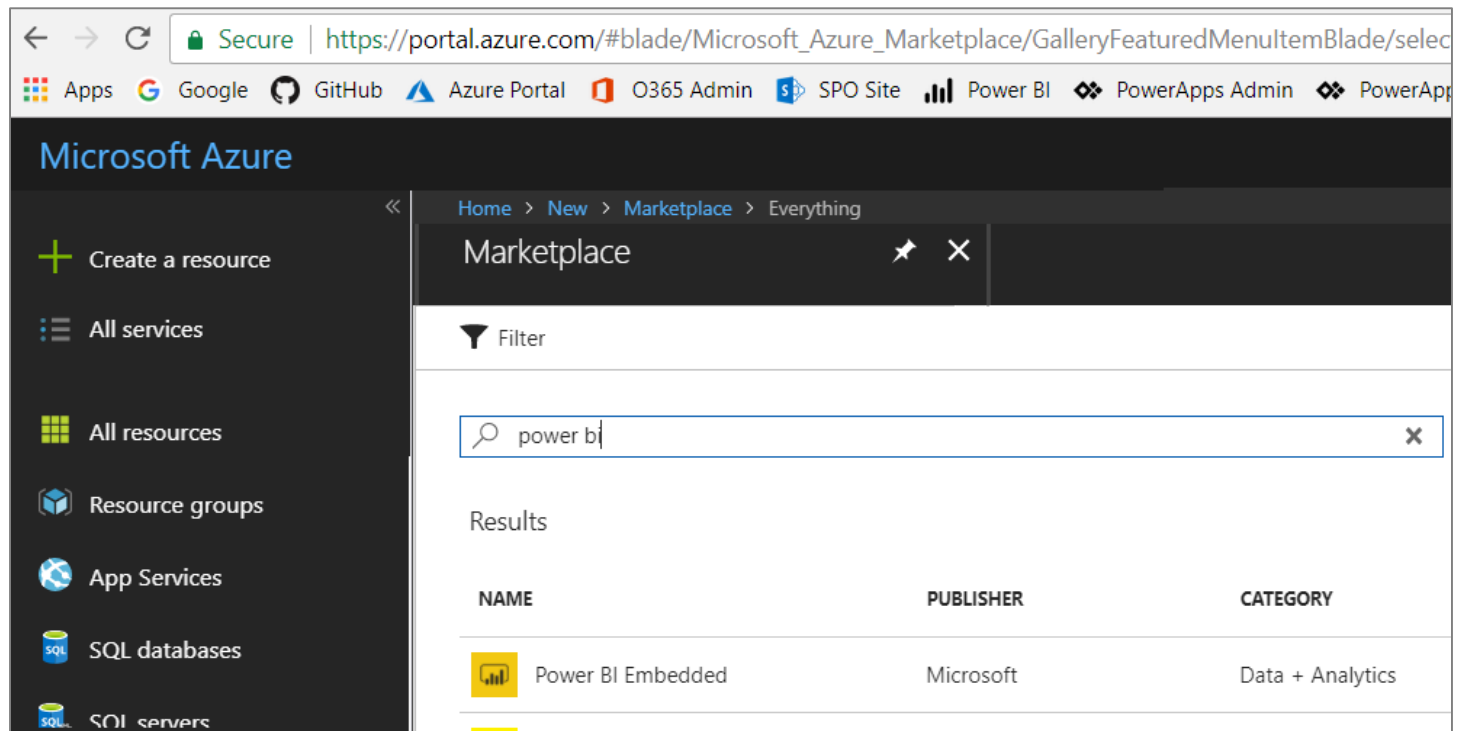
- A SKU is a Platform-as-a-Service
  - Used by ISVs as the data visualization layer
  - Allows for PBI Embedding into Custom Applications
  - Only supports 3<sup>rd</sup> party embedding (i.e. App-Owns-Data model)

Name	Virtual cores	Memory (GB)	Peak renders/hour	Cost/hour
A1	1	3	300	~\$1
A2	2	5	600	~\$2
A3	4	10	1,200	~\$4
A4	8	25	2,400	~\$8
A5	16	50	4,800	~\$16
A6	32	100	9,600	~\$32



# Creating the Power BI Embedded Service

- Power BI Embedded in an Azure on-demand service
  - Can be created manually through the Azure portal
  - Can be created in automated fashion using PowerShell
  - Requires an Azure subscription



# Azure Capacity Pricing Tiers

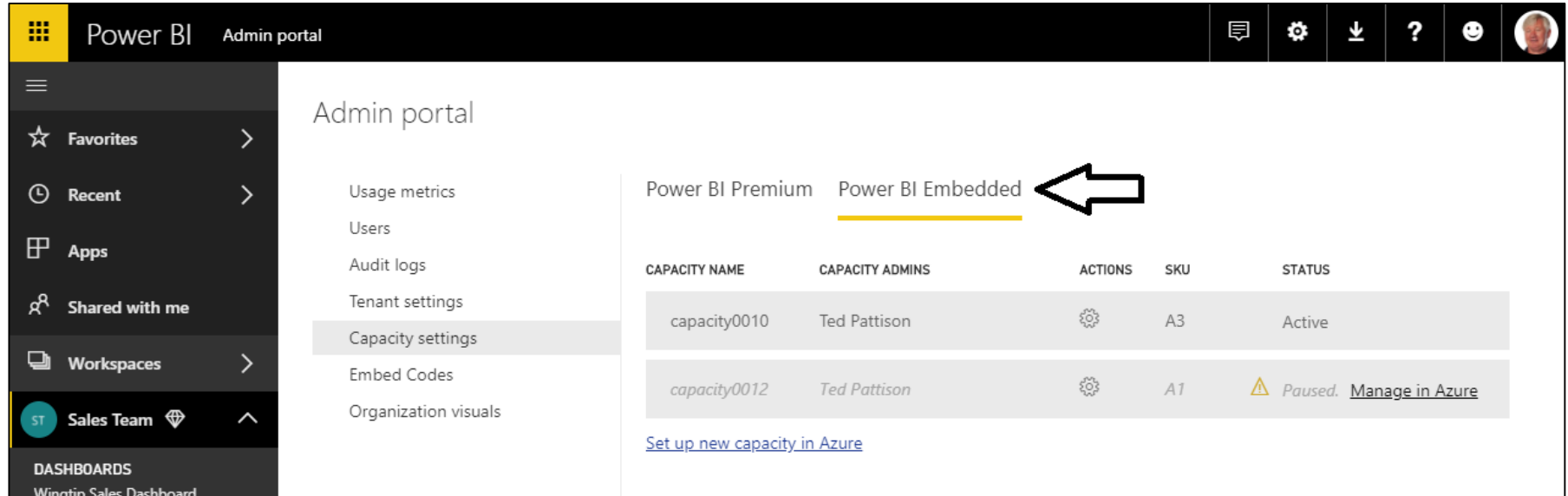
The screenshot shows the 'Choose your pricing tier' page in the Azure portal. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SCALE (Pricing tier), SETTINGS (Quick Start, Power BI capacity administrator, Properties, Locks, Automation script). The main content area displays six pricing tiers (A1 through A6) in a grid. Tier A3 is selected, indicated by a star icon and a blue border. Each tier card shows the number of V-Cores, cache size, infrastructure type, service type, and the estimated monthly cost in USD. A 'Select' button is located at the bottom of the grid.

Tier	V-Cores	Cache	Infrastructure	Service	Estimated Price (USD/MONTH)
A1	1	Up to 3 GB	Shared	Shared	750.03
A2	2	Up to 5 GB	Shared	Shared	1,494.03
A3	4	Up to 10 GB	Dedicated	Dedicated	2,994.00
A4	8	Up to 25 GB	Dedicated	Dedicated	5,994.04
A5	16	Up to 50 GB	Dedicated	Dedicated	11,994.02
A6	32	Up to 100 GB	Dedicated	Dedicated	23,994.45





# Managing Power BI Embedded Capacities



The screenshot displays the Power BI Admin portal interface. The top navigation bar includes the Power BI logo, the text 'Admin portal', and several utility icons (chat, settings, download, help, feedback, and user profile). A left-hand sidebar contains navigation options: Favorites, Recent, Apps, Shared with me, Workspaces, Sales Team, and DASHBOARDS. The main content area is titled 'Admin portal' and features a list of administrative tasks on the left: Usage metrics, Users, Audit logs, Tenant settings, Capacity settings (highlighted), Embed Codes, and Organization visuals. On the right, there are two tabs: 'Power BI Premium' and 'Power BI Embedded', with the latter being selected and underlined. A large black arrow points to the 'Power BI Embedded' tab. Below the tabs is a table listing the embedded capacities.

CAPACITY NAME	CAPACITY ADMINS	ACTIONS	SKU	STATUS
capacity0010	Ted Pattison		A3	Active
capacity0012	Ted Pattison		A1	Paused. <a href="#">Manage in Azure</a>

[Set up new capacity in Azure](#)



# PBI Capacity SKU Decoder Ring

	P SKU	EM SKU	A SKU
Purchased through...	Office 365	Office 365	Azure
Embed content in custom application	Yes	Yes	Yes
Share content with free PBI users outside PowerBI.com	Yes	Yes	No
Share content with free PBI users inside PowerBI.com	Yes	No	No
Billing	Monthly	Monthly	Hourly
Commitment	Monthly	Monthly/Yearly	None
Turn it off when your not using it	No	No	Yes



# Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- Authentication with Azure Active Directory
  - Programming with Power BI Service API
  - Working with Embeddable Resources
  - Embedding with Power BI JavaScript API



# Tenants and Organizational Accounts

- Azure AD used to authenticate users and apps
  - PBI licenses are assigned to Azure AD user accounts
  - Organization owns a tenant (i.e. directory)
  - AAD tenant contains user accounts and groups
  - AAD tenant contains set of registered applications
- You must register your application with Azure AD
  - Requirement of calling to Power BI service API
  - Applications registered as Web app or Native app
  - Registered applications are assigned GUID for client ID
  - Application is configured with permissions



# Creating an Azure AD Application

The image shows a sequence of three overlapping screenshots from the Microsoft Azure portal, illustrating the process of creating and configuring an Azure AD application.

**Top Screenshot: Create Application**  
The "Create" dialog is open in the "premium demo tenant - App registrations" section. The "Name" field is filled with "My Native App" and has a green checkmark. The "Application type" is set to "Native". The "Redirect UR" field is partially visible with the value "https://local". A "Create" button is at the bottom.

**Middle Screenshot: Application Overview**  
The "My Native App" page is shown, titled "Registered app". It has tabs for "Settings", "Manifest", and "Delete". Under the "Essentials" section, the following information is displayed:

Property	Value
Display name	My Native App
Application ID	7802d697-c0d5-480f-8f30-5039226f02a7
Application type	Native
Object ID	e9ee95cc-0e31-4705-a6ac-b2b10053df4b
Home page	Managed application in local directory <a href="#">My Native App</a>

An "All settings" button with a right arrow is located at the bottom right.

**Bottom Screenshot: Settings Page**  
The "Settings" page for the application is shown. It features a search bar labeled "Filter settings". The settings are organized into sections:

- GENERAL**
  - [Properties](#)
  - [Redirect URIs](#)
  - [Owners](#)
- API ACCESS**
  - [Required permissions](#)
  - [Keys](#)



# Application Permissions

- Applications can be granted permissions to other applications
  - Application permissions are app-only permissions
  - Delegated permissions are (app + user) permissions
  - Delegated permissions requires 1-time consent from user

**Required permissions**

**+ Add** **Grant Permissions**

API	APPLICATION PERMI...	DELEGATED PERMIS...
Windows Azure Active Directory	6	
Power BI Service	0	

**DELEGATED PERMISSIONS**

Permission	REQUIRES ADMIN
<input checked="" type="checkbox"/> Add data to a user's dataset (preview)	No
<input checked="" type="checkbox"/> View all Dashboards (preview)	No
<input checked="" type="checkbox"/> View all Datasets	No
<input checked="" type="checkbox"/> Read and Write all Datasets	No
<input checked="" type="checkbox"/> View content properties (preview)	No
<input checked="" type="checkbox"/> Create content (preview)	No
<input checked="" type="checkbox"/> View all Reports (preview)	No
<input checked="" type="checkbox"/> View all Groups	No
<input checked="" type="checkbox"/> View users Groups	No
<input checked="" type="checkbox"/> Read and Write all Reports	No





# 1<sup>st</sup> Party Embedding vs 3<sup>rd</sup> Party Embedding

	1st Part Embedding	3rd Party Embedding
Authentication flow	Authentication Code Grant Flow or Implicit Flow	Direct User Credentials
Identity used to call Power BI	Current User	Master User Account
Access to personal workspace	Yes	No
Access to app workspaces	Yes	Yes
Ability to reach non-licensed users	No	Yes
Supported Power BI Capacity SKUs	P* and EM* SKUs	P*, EM* and A* SKUs





**DEMO**

# Registering an App with Azure AD

# PbiEmbeddingManger Class

- PbiEmbeddingManger Class responsibilities
  - Get access tokens from Azure AD
  - Retrieve embedding data from Power BI service
  - Pass embedding data to browser using MVC view models

```
public class PbiEmbeddingManager {  
    "AAD Authentication Constants"  
  
    static string GetAccessToken() ...  
  
    static PowerBIClient GetPowerBiClient() ...  
  
    public static async Task<HomeViewModel> GetHomeViewModel() ...  
  
    public static async Task<DatasetsViewModel> GetDatasetsViewModel() ...  
  
    public static async Task<ReportsViewModel> GetReportsViewModel(string reportId, string datasetId) ...  
  
    public static async Task<DashboardsViewModel> GetDashboardsViewModel(string dashboardId) ...  
}
```





# Data Required for AAD Authentication

```
<configuration>
  <appSettings>

    <add key="clientId" value="23f6d66f-9a9a-4dba-9b7c-ff8aedadb831" />

    <add key="pbiUserName" value="pbiemasteruser@powerbimvps.onmicrosoft.com" />

    <add key="pbiUserPassword" value="Pa$$word!" />

    <add key="appworkspaceId" value="4baab6c0-87c5-4a2a-a73e-1f97adcc6bdb" />

  </appSettings>
</configuration>
```

```
public class PbiEmbeddingManager {

    #region "AAD Authentication Constants"

    static string aadAuthorizationEndpoint = "https://login.windows.net/common/oauth2/authorize";
    static string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";
    static string urlPowerBiRestApiRoot = "https://api.powerbi.com/";

    static string clientId = ConfigurationManager.AppSettings["clientId"];
    static string appWorkspaceId = ConfigurationManager.AppSettings["appWorkspaceId"];
    static string pbiUserName = ConfigurationManager.AppSettings["pbiUserName"];
    static string pbiUserPassword = ConfigurationManager.AppSettings["pbiUserPassword"];

    #endregion
}
```



# Getting an Access Token for the Master User

```
static string GetAccessToken() {  
    AuthenticationContext authContext = new AuthenticationContext(aadAuthorizationEndpoint);  
    var userCredentials = new UserPasswordCredential(pbiUserName, pbiUserPassword);  
  
    // this call will fail if permission consent has not be granted to master user account  
    string aadAccessToken =  
        authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userCredentials).Result.AccessToken;  
  
    // return Azure AD access token for master user account  
    return aadAccessToken;  
}
```



# Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Authentication with Azure Active Directory
- Programming with Power BI Service API
  - Working with Embeddable Resources
  - Embedding with Power BI JavaScript API





# The Power BI Service API

```
Microsoft.PowerBI.Api
├── Microsoft.PowerBI.Api.V1
├── Microsoft.PowerBI.Api.V1.Models
└── Microsoft.PowerBI.Api.V2
    ├── Dashboards
    ├── DashboardsExtensions
    ├── Datasets
    ├── DatasetsExtensions
    ├── Gateways
    ├── GatewaysExtensions
    ├── Groups
    ├── GroupsExtensions
    ├── IDashboards
    ├── IDatasets
    ├── IGateways
    ├── IGroups
    ├── IImports
    ├── Imports
    ├── ImportsExtensions
    ├── IPowerBIClient
    ├── IReports
    ├── ITiles
    ├── PowerBIClient
    ├── Reports
    ├── ReportsExtensions
    ├── Tiles
    └── TilesExtensions
```

```
Microsoft.PowerBI.Api
├── Microsoft.PowerBI.Api.V1
├── Microsoft.PowerBI.Api.V1.Models
├── Microsoft.PowerBI.Api.V2
└── Microsoft.PowerBI.Api.V2.Models
    ├── BasicCredentials
    ├── BindToGatewayRequest
    ├── CloneReportRequest
    ├── Column
    ├── ConnectionDetails
    ├── CredentialDetails
    ├── Dashboard
    ├── Dataset
    ├── DatasetMode
    ├── Datasource
    ├── EmbedToken
    ├── Gateway
    ├── GatewayDatasource
    ├── GatewayPublicKey
    ├── GenerateTokenRequest
    ├── Group
    ├── GroupCreationRequest
    ├── GroupUser
    ├── GroupUserAccessRight
    ├── Import
    ├── ImportConflictHandlerMode
    ├── ImportInfo
    ├── MemberAdminAccessRight
    ├── ODataResponseListDashboard
    ├── ODataResponseListDataset
    ├── ODataResponseListDatasource
    ├── ODataResponseListGateway
    ├── ODataResponseListGatewayDatasource
    ├── ODataResponseListGroup
    ├── ODataResponseListGroupUserAccessRight
    ├── ODataResponseListImport
    ├── ODataResponseListRefresh
    ├── ODataResponseListReport
    ├── ODataResponseListTable
    ├── ODataResponseListTile
    ├── ODataResponseListUserAccessRight
    ├── PublishDatasourceToGatewayRequest
    ├── RebindReportRequest
    ├── Refresh
    ├── Report
    ├── Row
    ├── Table
    ├── Tile
    ├── TokenAccessLevel
    ├── UpdateDatasourceRequest
    ├── UserAccessRight
    └── UserAccessRightEnum
```



# Initializing an Instance of PowerBIClient

- PowerBIClient object serves as top-level object
  - Used to execute calls against Power BI Service
  - Initialized with function to retrieve AAD access token

```
static string GetAccessToken() ...

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

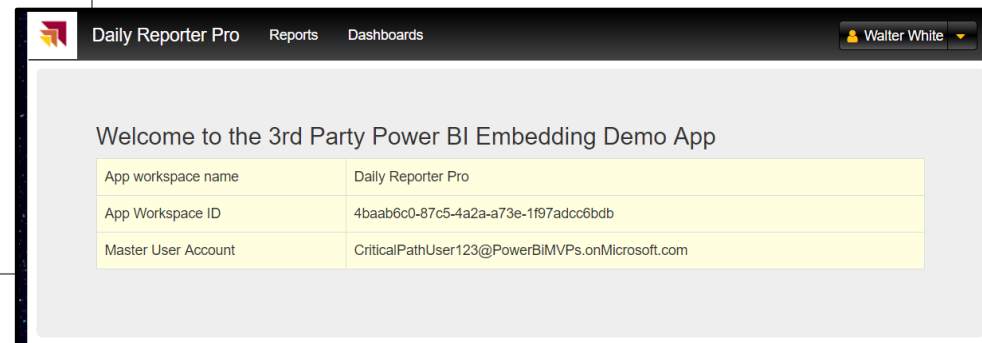
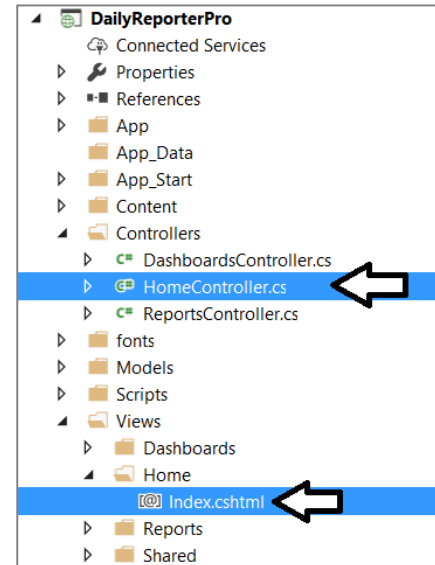
static void Main() {
    PowerBIClient pbiClient = GetPowerBiClient();
    var reports = pbiClient.Reports.GetReports().Value;
    foreach (var report in reports) {
        Console.WriteLine(report.Name);
    }
}
```



# MVC Controllers and Views

```
public class HomeController : Controller {  
  
    public async Task<ActionResult> Index() {  
        var viewModel = await PbiEmbeddingManager.GetHomeViewModel();  
        return View(viewModel);  
    }  
}
```

```
Index.cshtml  + x  
@model DailyReporterPro.Models.HomeViewModel  
  
<div id="home-view-container">  
    <div class="jumbotron">  
        <h3>Welcome to the 3rd Party Power BI Embedding Demo App</h3>  
  
        <table id="session-info-table" class="table table-bordered">  
            <tr>  
                <td>App workspace name</td>  
                <td>@Model.WorkspaceName</td>  
            </tr>  
            <tr>  
                <td>App Workspace ID</td>  
                <td>@Model.WorkspaceId</td>  
            </tr>  
            <tr>  
                <td>Master User Account</td>  
                <td>@Model.MasterUserAccount</td>  
            </tr>  
        </table>  
    </div>
```



# Back to the DailyReporterPro Application

```
public class HomeViewModel {  
    public string WorkspaceName;  
    public string WorkspaceId;  
    public string MasterUserAccount;  
}
```

```
public static async Task<HomeViewModel> GetHomeViewModel() {  
    var client = GetPowerBiClient();  
    var workspaces = (await client.Groups.GetGroupsAsync()).Value;  
    var workspace = workspaces.Where(ws => ws.Id == appWorkspaceId).FirstOrDefault();  
    var viewModel = new HomeViewModel {  
        WorkspaceName = workspace.Name,  
        WorkspaceId = workspace.Id,  
        MasterUserAccount = pbiUserName  
    };  
    return viewModel;  
}
```



# MVC View Models

```
namespace DailyReporterPro.Models {  
    public class HomeViewModel ...  
    public class DatasetViewModel ...  
    public class DatasetsViewModel ...  
    public class ReportViewModel ...  
    public enum ReportMode ...  
    public class ReportsViewModel ...  
    public class DashboardViewModel ...  
    public class DashboardsViewModel ...  
}
```

```
public static async Task<HomeViewModel> GetHomeViewModel() ...  
public static async Task<DatasetsViewModel> GetDatasetsViewModel() ...  
public static async Task<ReportsViewModel> GetReportsViewModel(string reportId, string datasetId) ...  
public static async Task<DashboardsViewModel> GetDashboardsViewModel(string dashboardId) ...
```



# Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Authentication with Azure Active Directory
- ✓ Programming with Power BI Service API
- Working with Embeddable Resources
  - Embedding with Power BI JavaScript API



# Embeddable Resources

1. Reports
2. Dashboards
3. Dashboard Tiles
4. Q & A Experience
5. Visual \*

\* really just a trick you do when embedding a report



# Report and Dataset Info

- Embed data required for an existing report

```
..... datasetId=9221313d-edc0-4c8a-b70e-ff0ac14f42be  
..... embedUrl=https://app.powerbi.com/reportEmbed?reportId=0dafe667-fd0b-4845-85d8-1  
..... id=0dafe667-fd0b-4845-85d8-136f93cfbde1  
..... isOriginalPbixReport=False  
..... isOwnedByMe=True  
..... modelId=0  
..... name=Northwind Retro  
..... webUrl=https://app.powerbi.com/groups/4baab6c0-87c5-4a2a-a73e-1f97adcc6bdb/repo
```

- Embed data for dataset required to create new

```
..... addRowsAPIEnabled=False  
..... configuredBy=TedP@powerbimvps.onmicrosoft.com  
..... id=9221313d-edc0-4c8a-b70e-ff0ac14f42be  
..... name=Northwind Retro
```





# Embed Tokens

- You can embed reports using master user AAD token, but...
  - You might want embed resource using more restricted tokens
  - You might want stay within the bounds of Power BI licensing terms
- Power BI service supports generating embed tokens
  - Embed token provides restrictions on whether user can view or edit
  - Each embed token created for one specific resource
  - Embed token can only be generated inside Power BI Premium capacity
  - Supports generating tokens using 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;
```



# View Model with Embed Data for Report

```
// create embed info for existing report
var embedConfig = new EmbedConfiguration() {
    EmbedToken = token,
    EmbedUrl = report.EmbedUrl,
    Id = report.Id
};
// add report data to view model
viewModel.CurrentReport = new ReportViewModel {
    Report = report,
    EmbedConfig = embedConfig
};
```



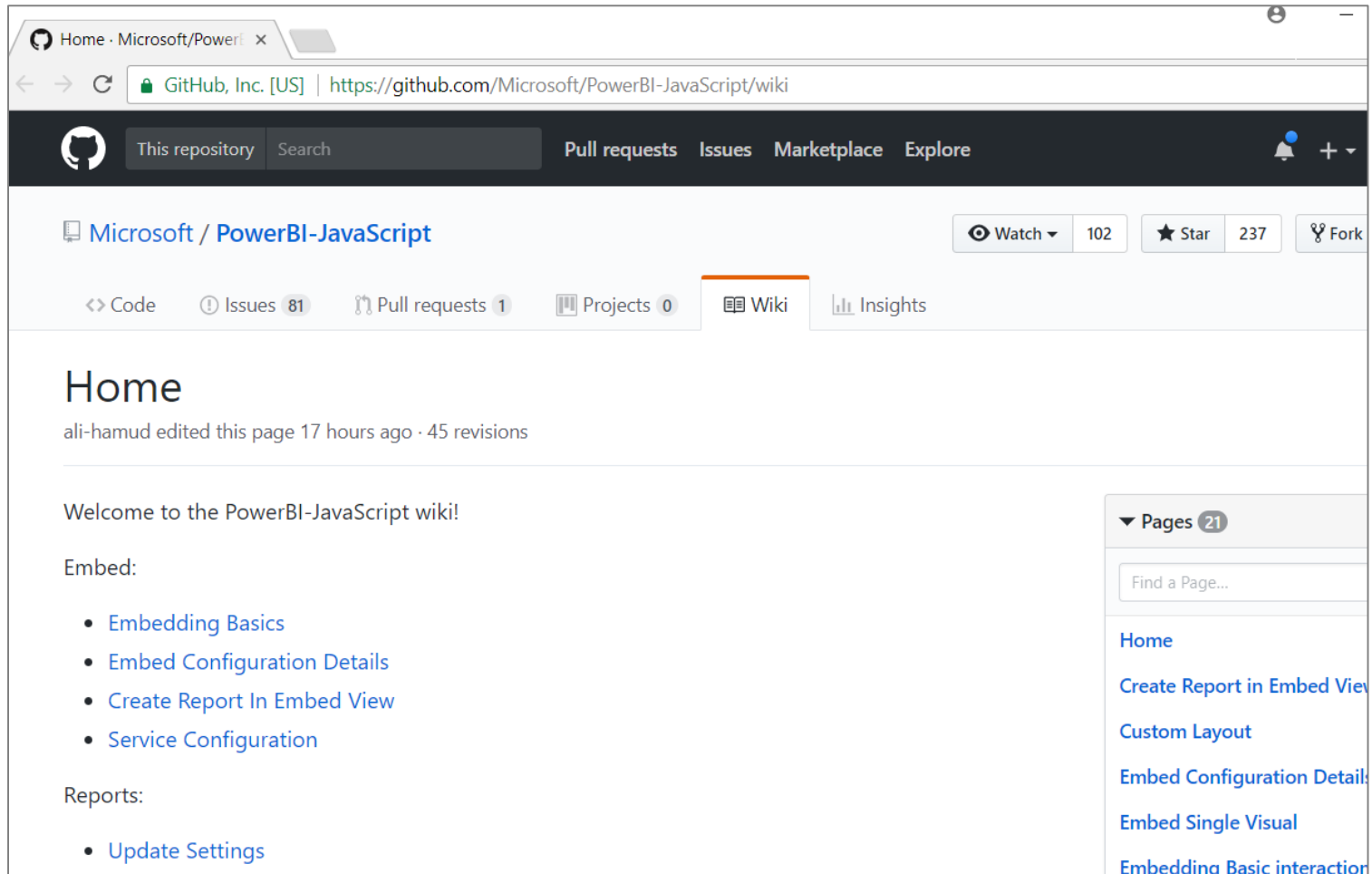
# Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Authentication with Azure Active Directory
- ✓ Programming with Power BI Service API
- ✓ Working with Embeddable Resources
- Embedding with Power BI JavaScript API



# Power BI JavaScript API (PBIJS)

- <https://github.com/Microsoft/PowerBI-JavaScript/wiki>



The screenshot shows a web browser displaying the GitHub repository page for Microsoft/PowerBI-JavaScript. The browser's address bar shows the URL <https://github.com/Microsoft/PowerBI-JavaScript/wiki>. The repository page header includes the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name is Microsoft / PowerBI-JavaScript, with 102 Watchers, 237 Stars, and a Fork button. The Wiki tab is selected, showing a 'Home' page. The page content includes a welcome message, a list of links for embedding Power BI reports, and a sidebar with a list of 21 wiki pages.

Home · Microsoft/PowerBI-JavaScript

GitHub, Inc. [US] | <https://github.com/Microsoft/PowerBI-JavaScript/wiki>

This repository Search Pull requests Issues Marketplace Explore

Microsoft / PowerBI-JavaScript Watch 102 Star 237 Fork

Code Issues 81 Pull requests 1 Projects 0 Wiki Insights

## Home

ali-hamud edited this page 17 hours ago · 45 revisions

Welcome to the PowerBI-JavaScript wiki!

Embed:

- [Embedding Basics](#)
- [Embed Configuration Details](#)
- [Create Report In Embed View](#)
- [Service Configuration](#)

Reports:

- [Update Settings](#)

Pages 21

Find a Page...

- [Home](#)
- [Create Report in Embed View](#)
- [Custom Layout](#)
- [Embed Configuration Details](#)
- [Embed Single Visual](#)
- [Embedding Basic interaction](#)

# Hello World with Power BI Embedding

- PBIJS library provides **powerbi** as top-level service object
  - You create configuration and then call **powerbi.embed** to embed a report
  - You must pass access token as part of the configuration

```
// data required for embedding Power BI report
var embedReportId = "f10c9de9-a325-4a43-af9f-0cf35cca2ab7";
var embedUrl = "https://app.powerbi.com/reportEmbed?reportId=f10c9de9-a325-4a43-af9f";
var accessToken = "H4sIAAAAAAEACWwtw6sCBZE_-WlrIR3K02A9x66gQzvVwe0_76tmbySW6pbdF7-Y";

// 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,
};

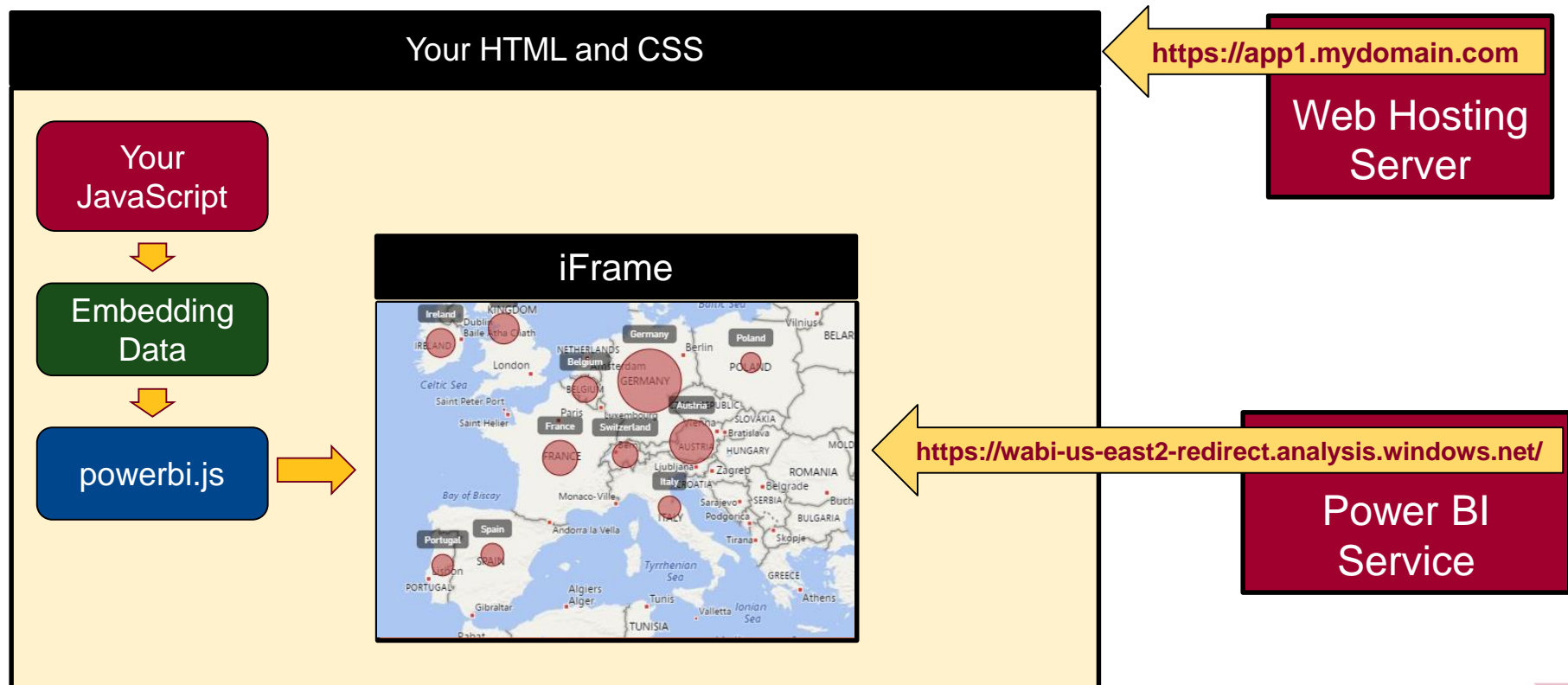
// Get a reference to the embedded report HTML element
var reportContainer = document.getElementById('embedContainer');

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



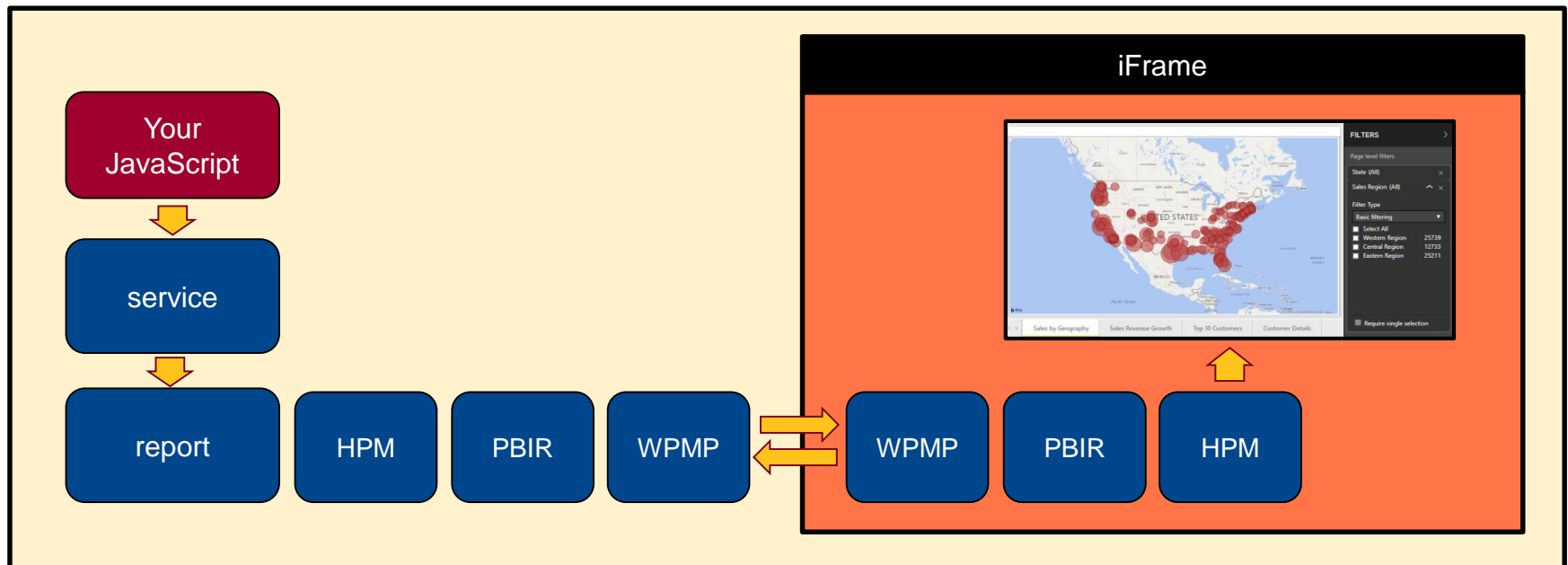
# Report Embedding Architecture

- 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***



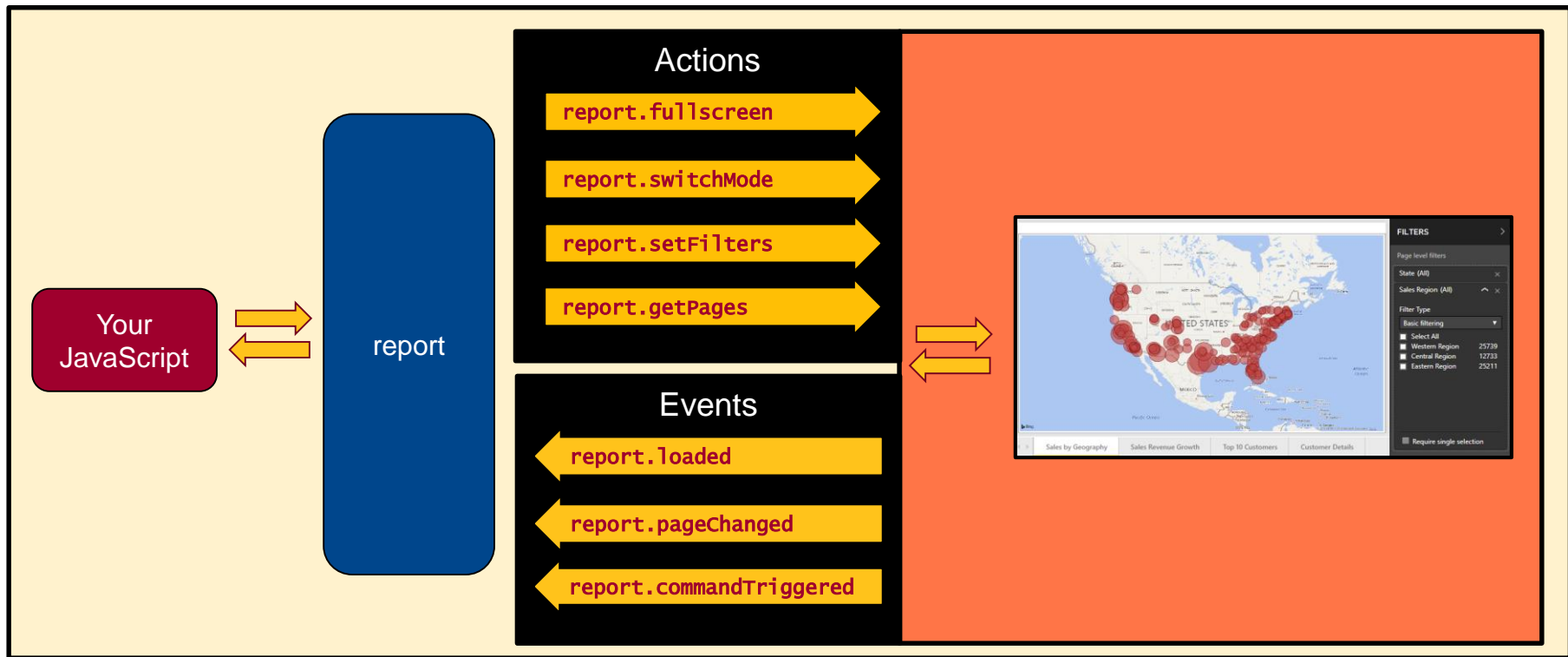
# Post Message Communications Flow

- 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





# Embedding Data in MVC View

```
@if (Model.ReportMode == DailyReporterPro.Models.ReportMode.ExistingReport) {  
<script>  
    var embedReportId = "@Model.CurrentReport.EmbedConfig.Id";  
    var embedUrl = "@Html.Raw(Model.CurrentReport.EmbedConfig.EmbedUrl)";  
    var accessToken = "@Model.CurrentReport.EmbedConfig.EmbedToken.Token";  
    var reportContainer = document.getElementById('reportContainer');  
    // call embedReport utility function defined inside App.ts  
    PowerBIEmbeddingManagerClient.embedReport(embedReportId, embedUrl, accessToken, reportContainer);  
</script>  
}  
@if (Model.ReportMode == DailyReporterPro.Models.ReportMode.NewReport) {  
<script>  
    var embedDatasetId = "@Model.CurrentDataset.EmbedConfig.DatasetId";  
    var embedUrl = "@Html.Raw(Model.CurrentDataset.EmbedConfig.EmbedUrl)";  
    var accessToken = "@Model.CurrentDataset.EmbedConfig.EmbedToken.Token";  
    var reportContainer = document.getElementById('reportContainer');  
    // call embedReport utility function defined inside App.ts  
    PowerBIEmbeddingManagerClient.createReport(embedDatasetId, embedUrl, accessToken, reportContainer);  
</script>  
}
```

```
class PowerBIEmbeddingManagerClient {  
  
    static embedReport = (reportId, embedUrl, accessToken, reportContainer) => {  
        // ...  
    };  
  
    static createReport = (datasetId, embedUrl, accessToken, reportContainer) => {  
        // ...  
    };  
  
    static embedDashboard = (dashboardId, embedUrl, accessToken, reportContainer) => {  
        // ...  
    };  
}
```

# Loading an Embedded Report

```
static embedReport = (reportId, embedUrl, accessToken, reportContainer) => {  
  
    var report: Report;  
    var pages: IPageCollection;  
  
    var currentPage: Page = null;  
    var currentPageIndex: number = 0;  
  
    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  
        }  
    };  
  
    report = <Report>powerbi.embed(reportContainer, config);
```



# Embedded Report Options

```
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>

The screenshot shows the GitHub repository page for **CriticalPathTraining / PowerBiEmbeddedScratchpad**. The repository is described as "A sample application for developer's learning about Power BI Embedding". It has 4 commits, 1 branch, 0 releases, 1 contributor, and is licensed under MIT. The repository is currently on the **master** branch. A table of recent updates is shown below the repository information.

Update	Commit	Time
TedPattison Updates	3d2e699	7 days ago
DemoPagesWithEmbedding		8 days ago
PBIX		7 days ago
PowerBiEmbeddedScratchpad		8 days ago







**DEMO**

## **The Power BI Embedded Scratchpad App**

# Summary

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Authentication with Azure Active Directory
- ✓ Programming with Power BI Service API
- ✓ Working with Embeddable Resources
- ✓ Embedding with Power BI JavaScript API

