

# Embedding Power BI Reports with ASP.NET MVC



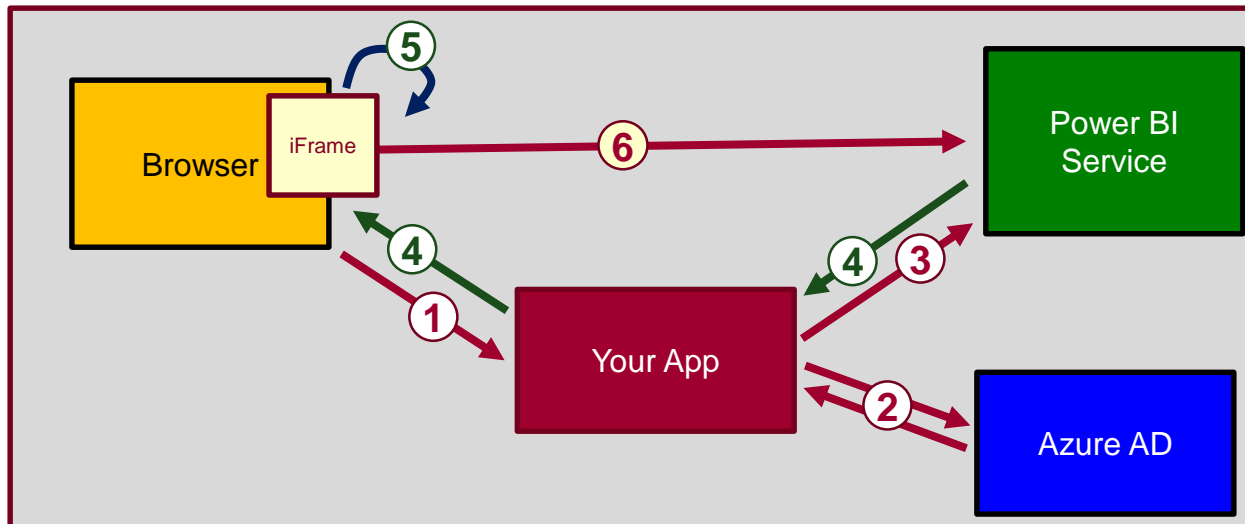
# Agenda

- Power BI Embedding Onboarding Experience
- Creating ASP.NET MVC Project for Power BI Embedding
- Retrieving Power BI Embedding Data
- Generating and Managing Embed Tokens
- Writing Client-side Code to Embed Resources



# Power BI Embedding – The Big Picture

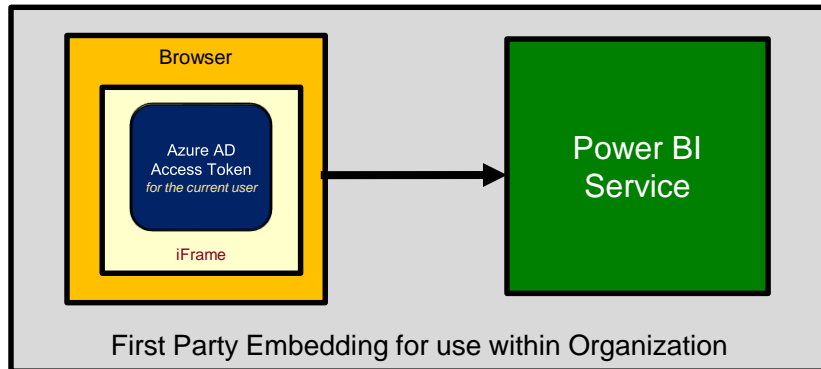
- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded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



# First Party Embedding vs Third Party Embedding

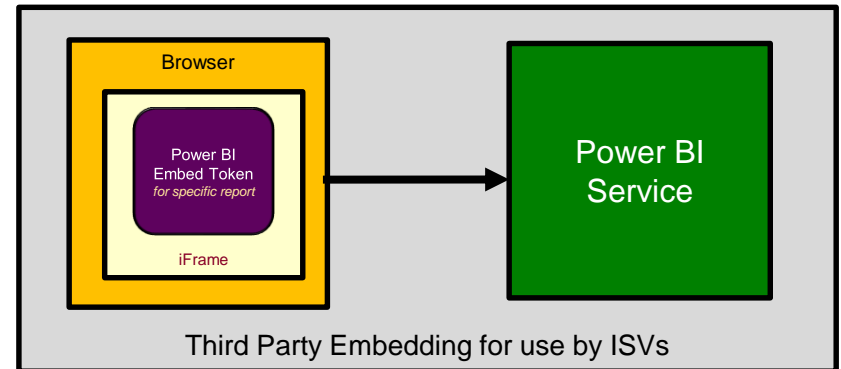
## First Party Embedding

- 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

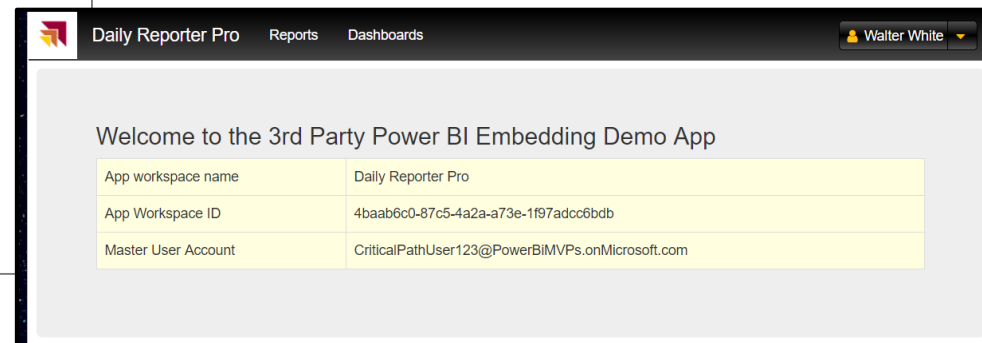
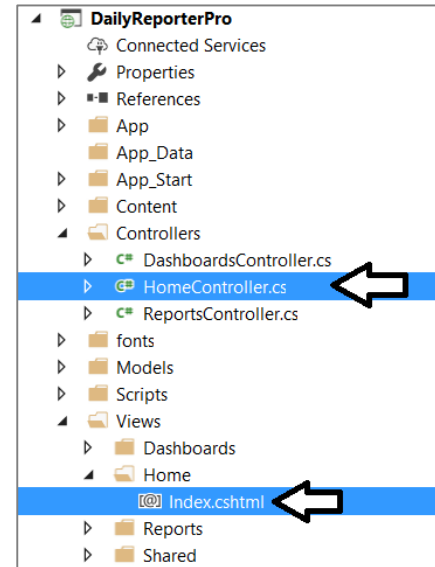
- Known as **App-Owns-Data** Model
- No users require Power BI license
- Useful for commercial applications
- App authenticates with master user account
- Your code runs with admin permissions
- Embed token passed to browser



# 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>
```



# Creating Model Classes for MVC Views

```
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) ...
```



# Report and Dataset Info

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

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

// data required for embedding a dashboard
class DashboardTileEmbeddingData {
    public string dashboardId;
    public string TileId;
    public string TileTitle;
    public string embedUrl;
    public string accessToken;
}
```

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

// data required for embedding QnA experience
class QnaEmbeddingData {
    public string datasetId;
    public string embedUrl;
    public string accessToken;
}
```





# Data for First Party Report Embedding

```
public static ReportEmbeddingData GetReportEmbeddingDataFirstParty() {  
    PowerBIClient pbiclient = GetPowerBIClient();  
  
    var report = pbiclient.Reports.GetReportInGroup(workspaceId, reportId);  
    var embedUrl = report.EmbedUrl;  
    var reportName = report.Name;  
    var accessToken = GetAccessToken();  
  
    return new ReportEmbeddingData {  
        reportId = reportId,  
        reportName = reportName,  
        embedUrl = embedUrl,  
        accessToken = accessToken  
    };  
}
```



# 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
- 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;
```



# Data for Third Party Report Embedding

```
public static ReportEmbeddingData GetReportEmbeddingData() {  
    PowerBIClient pbiclient = GetPowerBIClient();  
  
    var report = pbiclient.Reports.GetReportInGroup(workspaceId, reportId);  
    var embedUrl = report.EmbedUrl;  
    var reportName = report.Name;  
  
    // create token request object  
    GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");  
  
    // call to Power BI Service API and pass GenerateTokenRequest object to generate embed token  
    string embedToken = pbiclient.Reports.GenerateTokenInGroup(workspaceId,  
                                                                report.Id,  
                                                                generateTokenRequestParameters).Token;  
  
    return new ReportEmbeddingData {  
        reportId = reportId,  
        reportName = reportName,  
        embedUrl = embedUrl,  
        accessToken = embedToken  
    };  
}
```



# Getting the Data for Dashboard Embedding

```
public static DashboardEmbeddingData GetDashboardEmbeddingData() {  
    PowerBIClient pbiclient = GetPowerBIClient();  
  
    var dashboard = pbiclient.Dashboards.GetDashboardInGroup(workspaceId, dashboardId);  
    var embedUrl = dashboard.EmbedUrl;  
    var dashboardDisplayName = dashboard.DisplayName;  
  
    GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");  
    string embedToken = pbiclient.Dashboards.GenerateTokenInGroup(workspaceId,  
                                                                    dashboardId,  
                                                                    generateTokenRequestParameters).Token;  
  
    return new DashboardEmbeddingData {  
        dashboardId = dashboardId,  
        dashboardName = dashboardDisplayName,  
        embedUrl = embedUrl,  
        accessToken = embedToken  
    };  
}
```



# Data for Dashboard 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  
    };  
}
```



# 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  
    };  
}
```





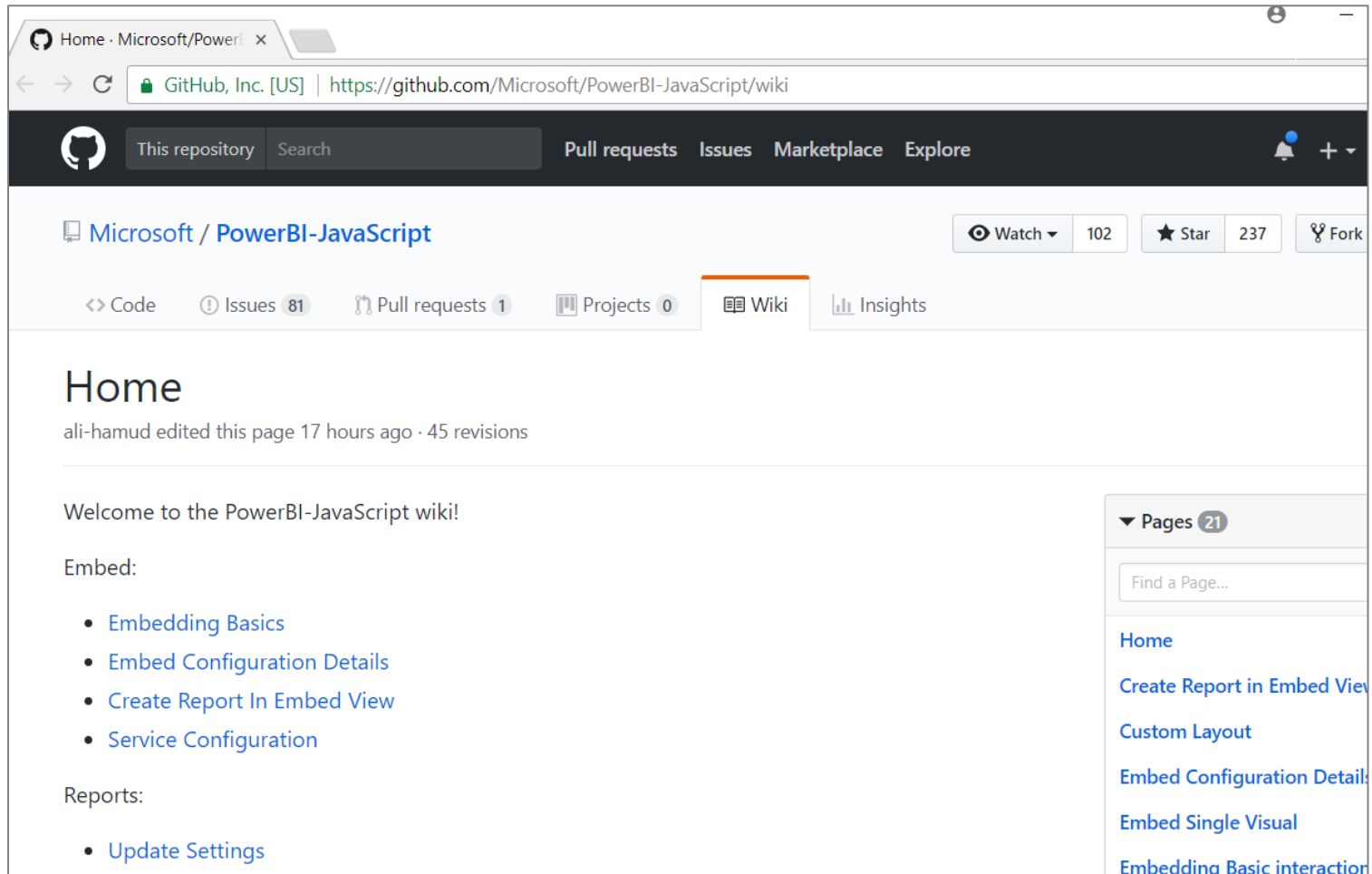
# Data for Q&A Experience Embedding

```
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  
    };  
}
```



# 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 "Microsoft / PowerBI-JavaScript" is displayed, along with statistics: 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 = "H4sIAAAAAEACWwtw6sCBZE_-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);
```



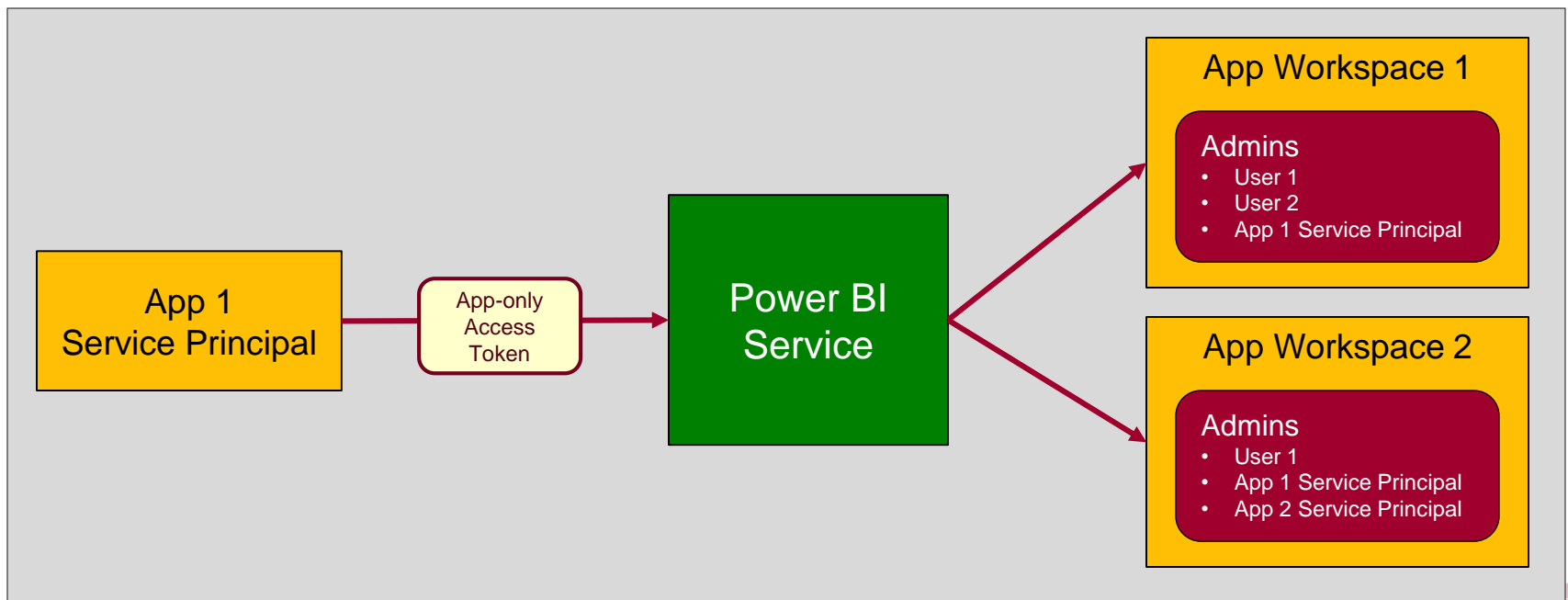
# App-only Auth for Power BI Embedding

- Legacy technique for third-party embedding
  - App authenticates as master user account
  - Master user account added as admin to app workspace
  - Master user account requires Power BI Pro license
  - App must authenticate with user-based identity
- New best practice in third-party embedding
  - App authenticates as itself with app-only identity
  - App service principal added as admin to app workspace
  - No Azure AD application permissions required



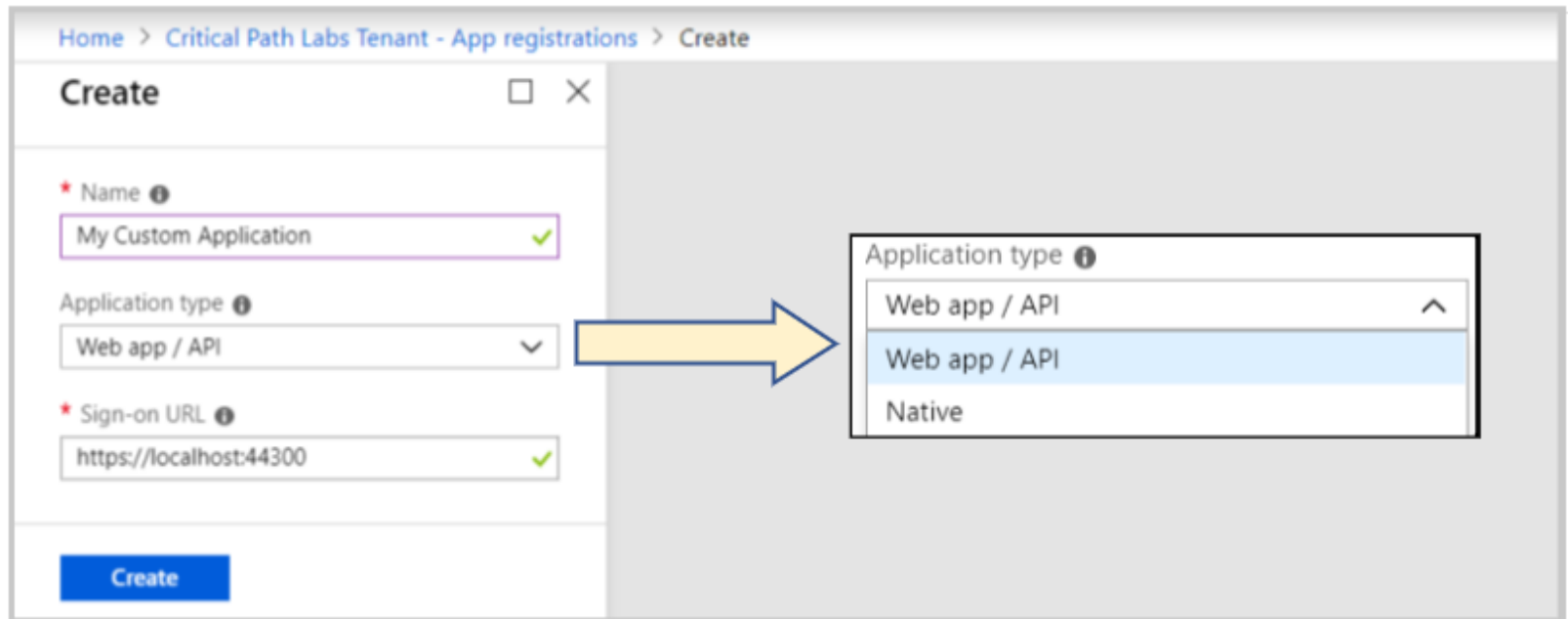
# App-only Access Control

- Service Principal used to configure access control
  - Requires the use of v2 app workspaces
  - Service principal added to app workspaces as admin
  - Access control NOT based on Azure AD permissions



# Application Types

- Azure AD Application Types
  - Native clients
  - Web app / API client



The screenshot shows the 'Create' page for app registrations in the Azure AD portal. The breadcrumb trail at the top reads 'Home > Critical Path Labs Tenant - App registrations > Create'. The form contains three main fields: 'Name' with the value 'My Custom Application', 'Application type' set to 'Web app / API', and 'Sign-on URL' with the value 'https://localhost:44300'. A blue 'Create' button is at the bottom left. A yellow arrow points from the 'Application type' dropdown to a detailed view of the dropdown menu on the right. This menu shows 'Web app / API' as the selected option, with 'Native' as another available choice.

Home > Critical Path Labs Tenant - App registrations > Create

Create

\* Name ⓘ  
My Custom Application ✓

Application type ⓘ  
Web app / API ▼

\* Sign-on URL ⓘ  
https://localhost:44300 ✓

Create

Application type ⓘ

- Web app / API ^
- Web app / API
- Native





# Creating a New Azure AD Application

Demo App

Registered app

Settings

Manifest

Delete




Display name	Application ID
Demo App	8d9c62ba-08a4-4b9f-82f4-5c50b4412756
Application type	Object ID
Web app / API	74168095-79c6-4c34-a51a-2d623d36a4bd
Home page	Managed application in local directory
<a href="https://localhost:44300">https://localhost:44300</a>	<a href="#">Demo App</a>


Click to copy



# Creating the App Secret

## Keys


 Save  Discard  Upload Public Key

 Copy the key value. You won't be able to retrieve after you leave this blade.

### Passwords

DESCRIPTION	EXPIRES	VALUE
Key1	2/13/2020	CTFbmw2wjLAWLVZcrEOkKbsegiFnECIFI+MR9FbyR+U=

Key description

Duration 

Value will be displayed on save

Update application keys  
Successfully updated application Dem



# Creating Azure AD Apps with PowerShell

```
# create Azure AD Application
$aadApplication = New-AzureADApplication `
    -DisplayName $appDisplayName `
    -PublicClient $false `
    -AvailableToOtherTenants $false `
    -ReplyUrls @($replyUrl) `
    -Homepage $replyUrl `
    -PasswordCredentials $passwordCredential

# create applicaiton's service principal
$appId = $aadApplication.AppId
$serviceServicePrincipal = New-AzureADServicePrincipal -AppId $appId
$serviceServicePrincipalObjectId = $serviceServicePrincipal.ObjectId

# assign current user as owner
Add-AzureADApplicationOwner -ObjectId $aadApplication.ObjectId -RefObjectId $user.ObjectId
```

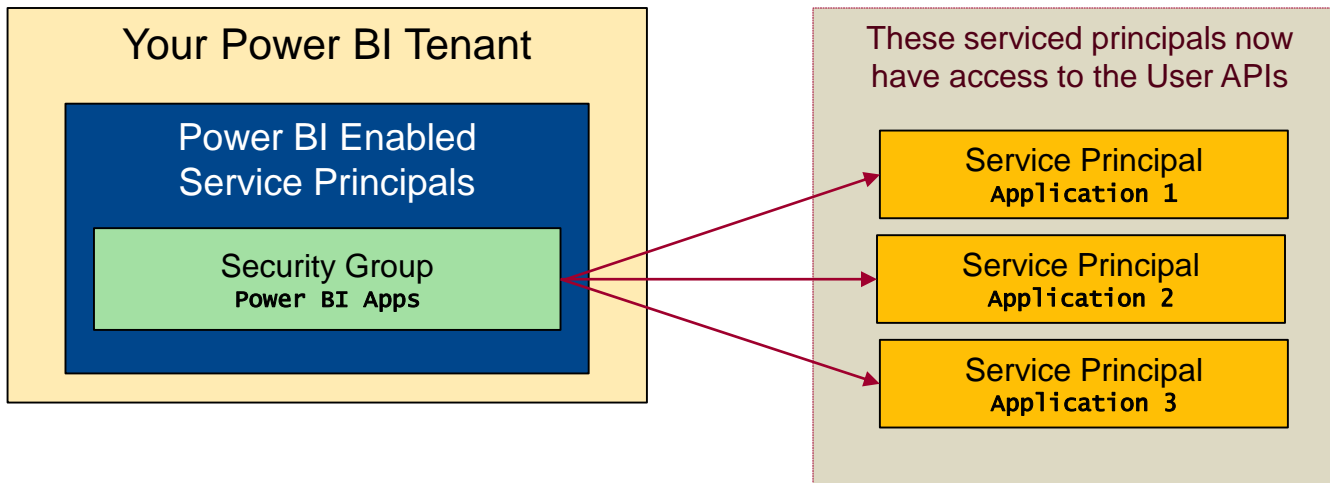
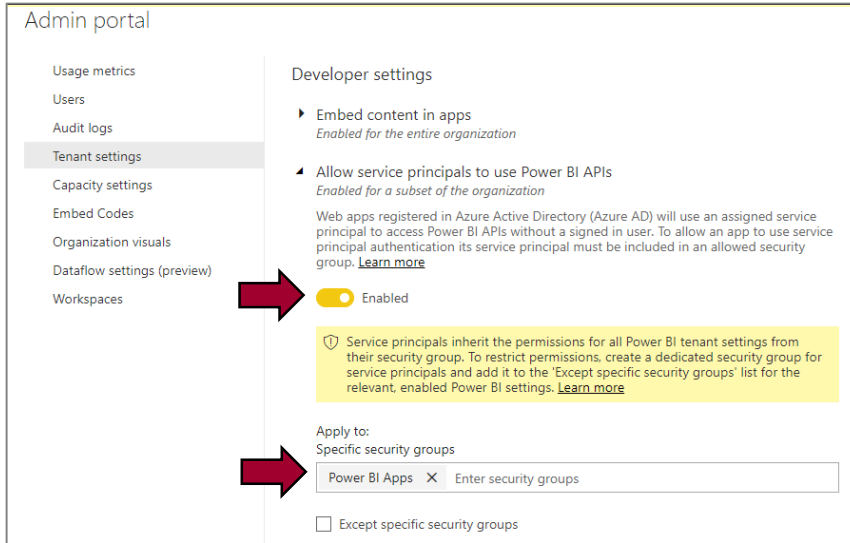


# User APIs versus Admin APIs

- Power BI User APIs (e.g. [GetGroupsAsync](#))
  - provides users with access to personal workspace
  - provides users with access to app workspaces
  - provides service principal (SP) with access to app workspaces
- Power BI Admin APIs (e.g. [GetGroupsAsAdminAsync](#))
  - provides users with tenant-level access to all workspaces
  - does not currently support app-only authentication

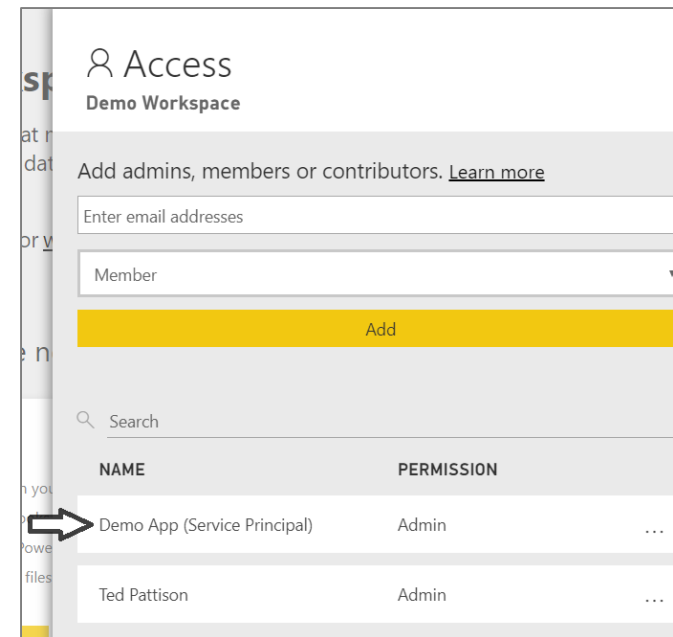
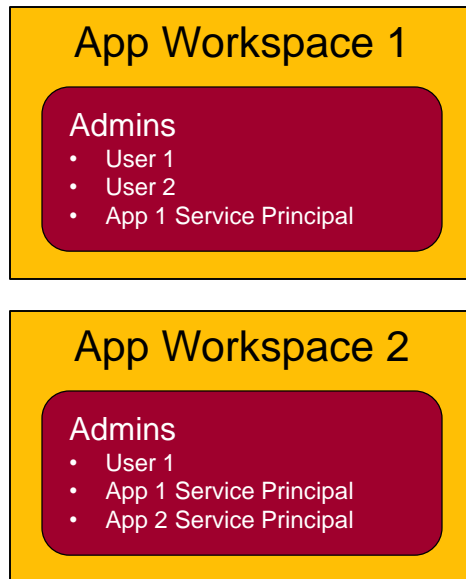


# Tenant Setup



# App-only Access with PBI Service API

- Service Principal added to workspace as admin
  - Only works with v2 app workspaces
  - Provides full workspace access to service principal





# Authentication Flows

- **User Password Credential Flow** (*public client*)
  - Requires registering Azure AD application as native app
  - Requires passing user name and password across network
- **Implicit Flow** (*public client*)
  - Used in SPAs built with JavaScript and AngularJS
- **Authorization Code Flow** (*confidential client*)
  - Client first obtains authorization code for user
  - Access token acquired in server-to-server call
- **Client Credential Flow** (*confidential client*)
  - Used to obtain app-only access token
  - Authentication based on app secret or app certificate



# What Can You Do with an App-only Token?

- Common operations possible through User APIs
  - Create new app workspaces
  - Clone content between app workspaces
  - Enumerate thru list of datasets, reports & dashboards
  - Retrieve embedding data for reports and dashboards
  - Generate embed tokens for App-Owns-Data model
  - Upload PBIX files to create datasets and reports
  - Update connection strings and dataset credentials
  - Create/populate streaming datasets and push datasets



# Agenda

- Power BI Embedding Onboarding Experience
- Creating ASP.NET MVC Project for Power BI Embedding
- Retrieving Power BI Embedding Data
- Generating and Managing Embed Tokens
- Writing Client-side Code to Embed Resources

