

Embedding Power BI Reports using SPFX



Agenda

- Power BI Embedding Fundamentals
 - App Workspaces and Dedicated Capacities
 - Programming with Power BI Service API
 - Embedding with Power BI JavaScript API
 - Calling the Power BI Service using AadHttpClient



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

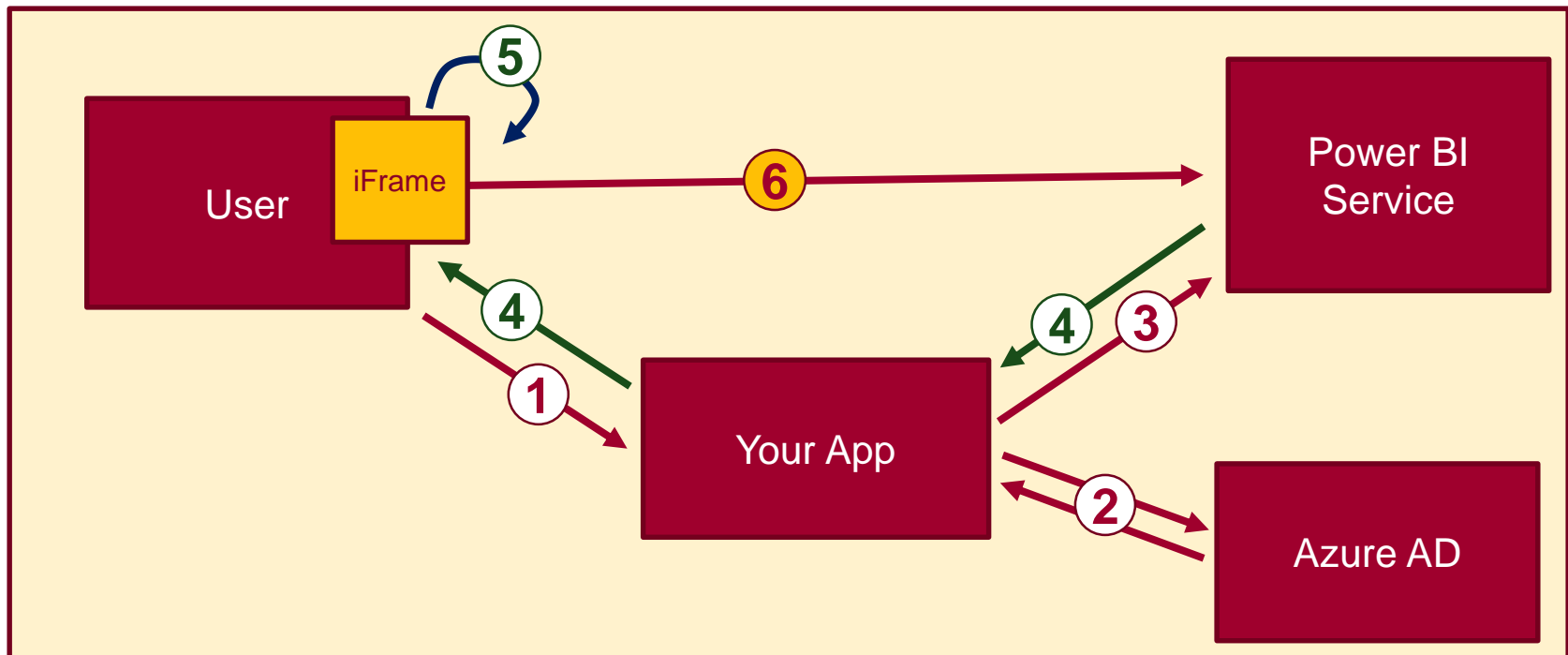


- Using the Power BI Service API
 - Accessible by making direct REST calls against service
 - Accessible by using Assembly DLL that abstracts away REST calls
 - Assembly DLL is named **Microsoft.PowerBI.Api.dll**
 - Assembly DLL part of NuGet package (**Microsoft.PowerBI.Api**)
 - Calling service requires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



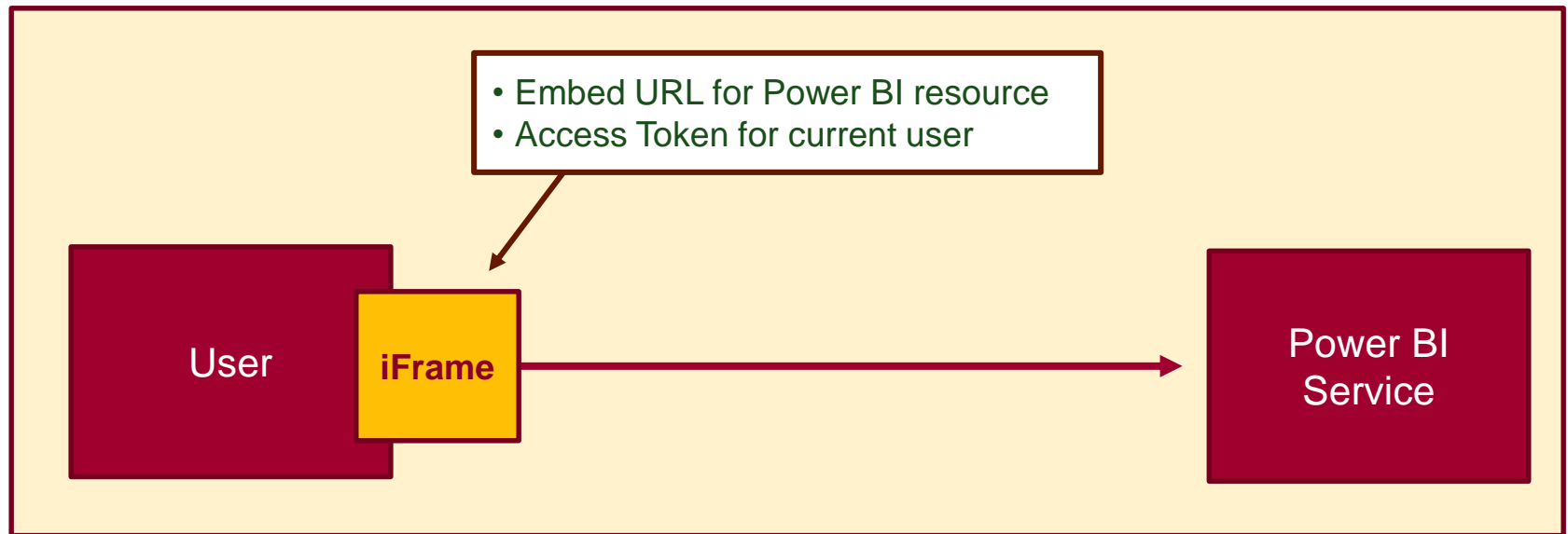
Power BI Embedding – The Big Picture

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



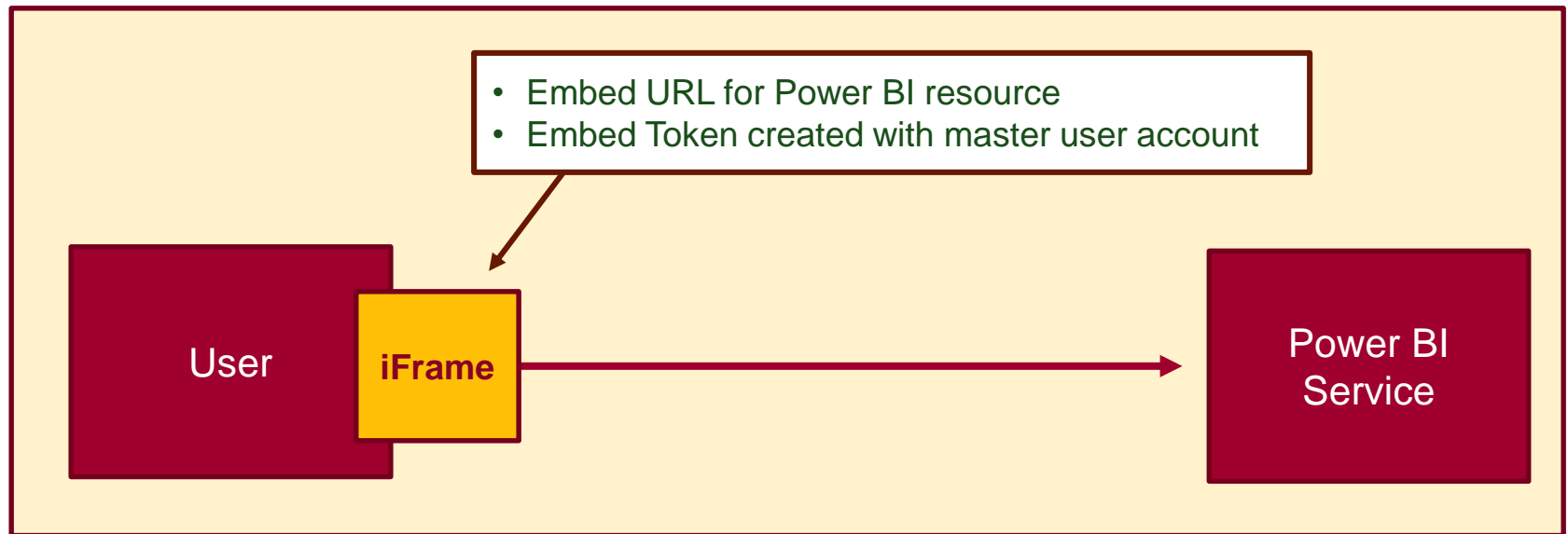
First Party Embedding

- 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



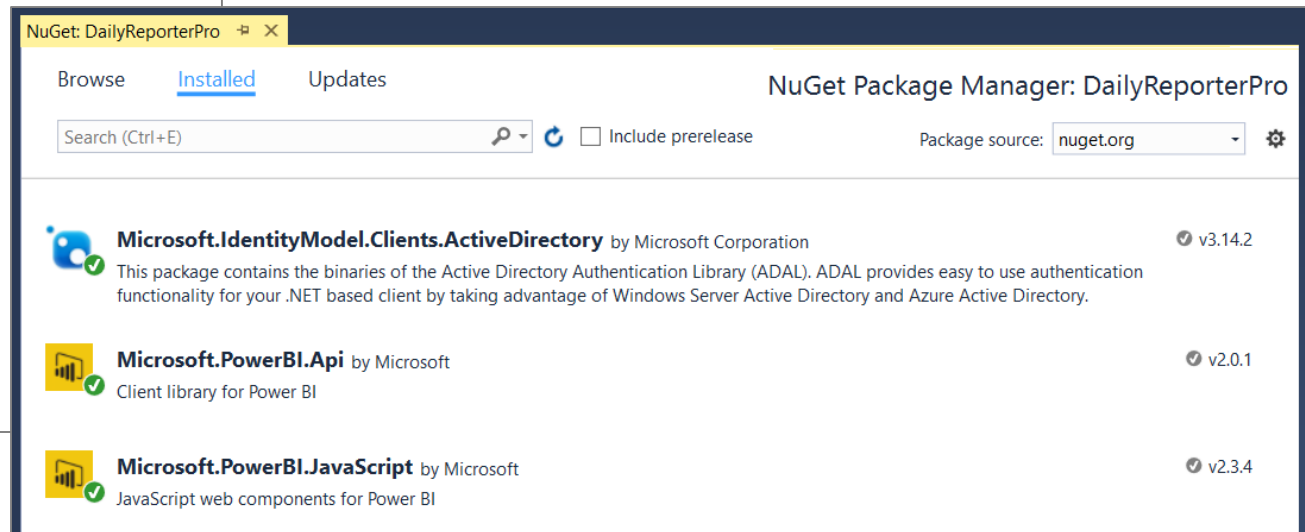
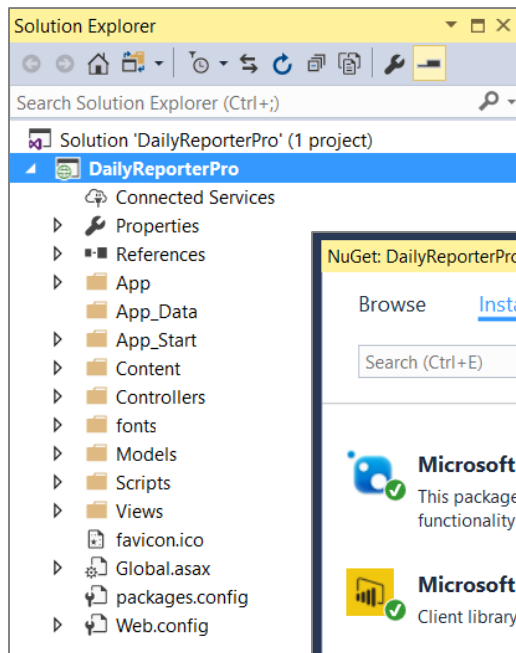
First Party vs Third Party Embedding

- 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

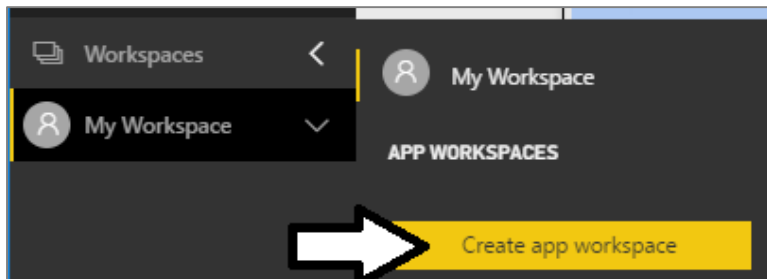
Agenda

- ✓ Power BI Embedding Fundamentals
- App Workspaces and Dedicated Capacities
 - Programming with Power BI Service API
 - Embedding with Power BI JavaScript API
 - Calling the Power BI Service using AadHttpClient



Understanding App Workspaces

- App workspaces required for 3rd 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



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 a black 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 tabs, a section titled 'V-CORES' shows a progress bar indicating '8 of 8 used' and '8 v-cores' available. A yellow bar represents the used capacity, and a grey bar below it indicates '0 available'. 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

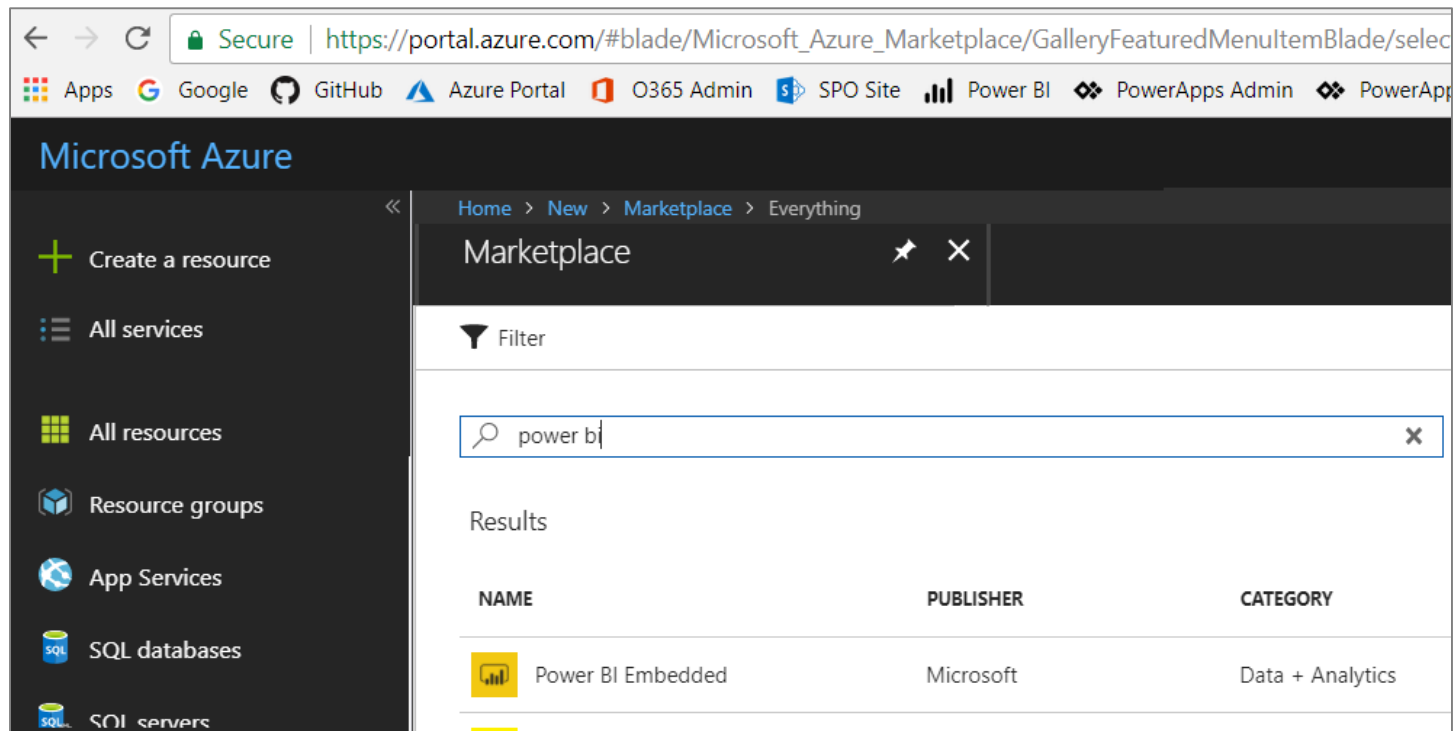
Choose an available dedicated capacity for this workspace

Capacity P1 #1 8GB model ▾

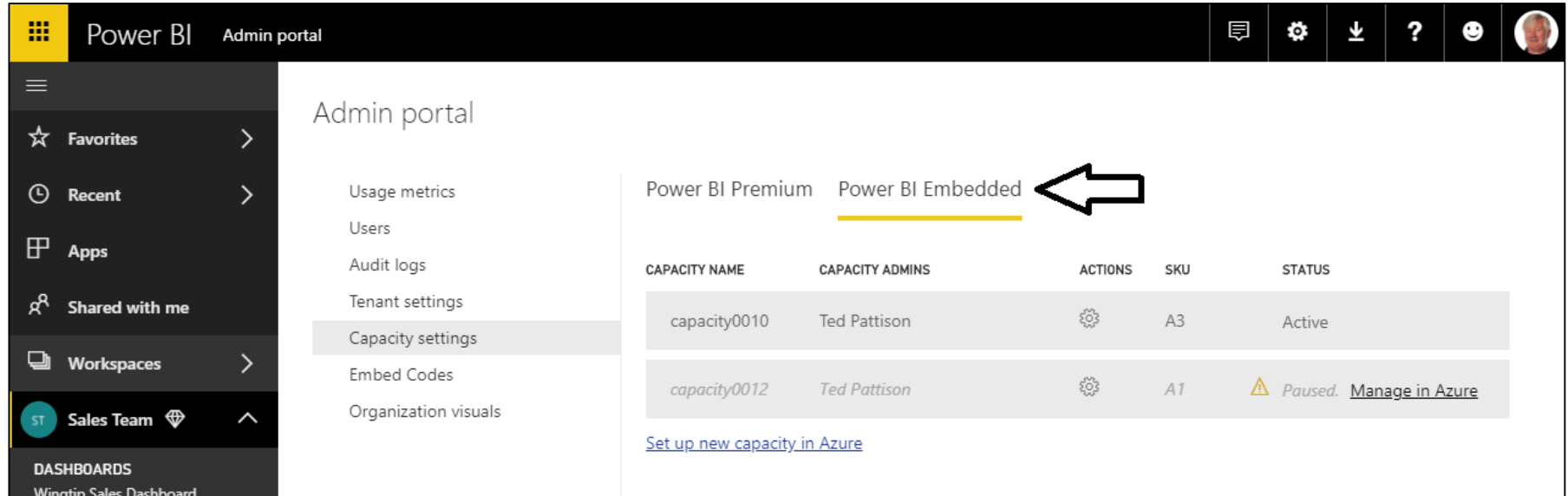


Creating the Power BI Embedded Service

- 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



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). The left sidebar contains a menu with options: Favorites, Recent, Apps, Shared with me, Workspaces, Sales Team (selected), and DASHBOARDS (with a link to 'Winfin Sales Dashboard'). The main content area is titled 'Admin portal' and lists several management options: 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. Manage in Azure

[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



1st Party Embedding vs 3rd 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



Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- Programming with Power BI Service API
 - Embedding with Power BI JavaScript API
 - Calling the Power BI Service using AadHttpClient



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 box 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 URI" is "https://localhost". 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 My Native App

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

Bottom Screenshot: Settings Page
The "Settings" page for the application is shown. It has 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 **REQUIRES ADMIN**

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



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);
}

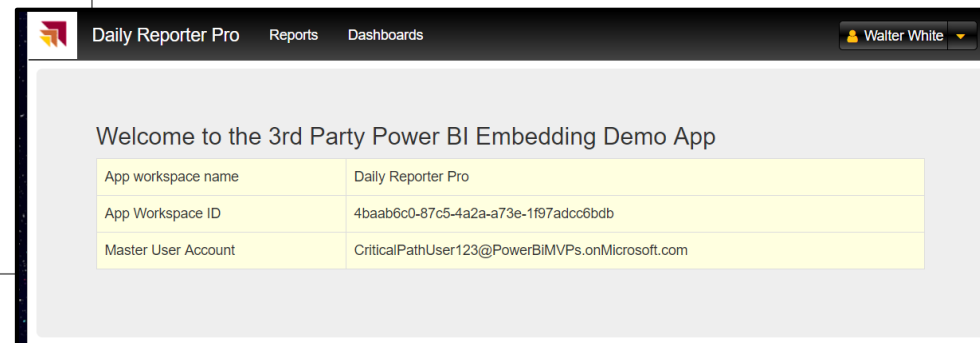
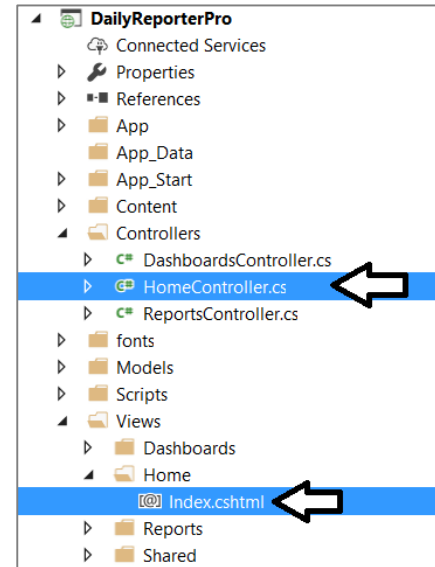
static void Main() {
    PowerBIClient pbiClient = GetPowerBiClient();
    var reports = pbiClient.Reports.GetReports().Value;
    foreach (var report in reports) {
        Console.WriteLine(report.Name);
    }
}
```



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
- ✓ Programming with Power BI Service API
- Embedding with Power BI JavaScript API
 - Calling the Power BI Service using AadHttpClient



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
};
```



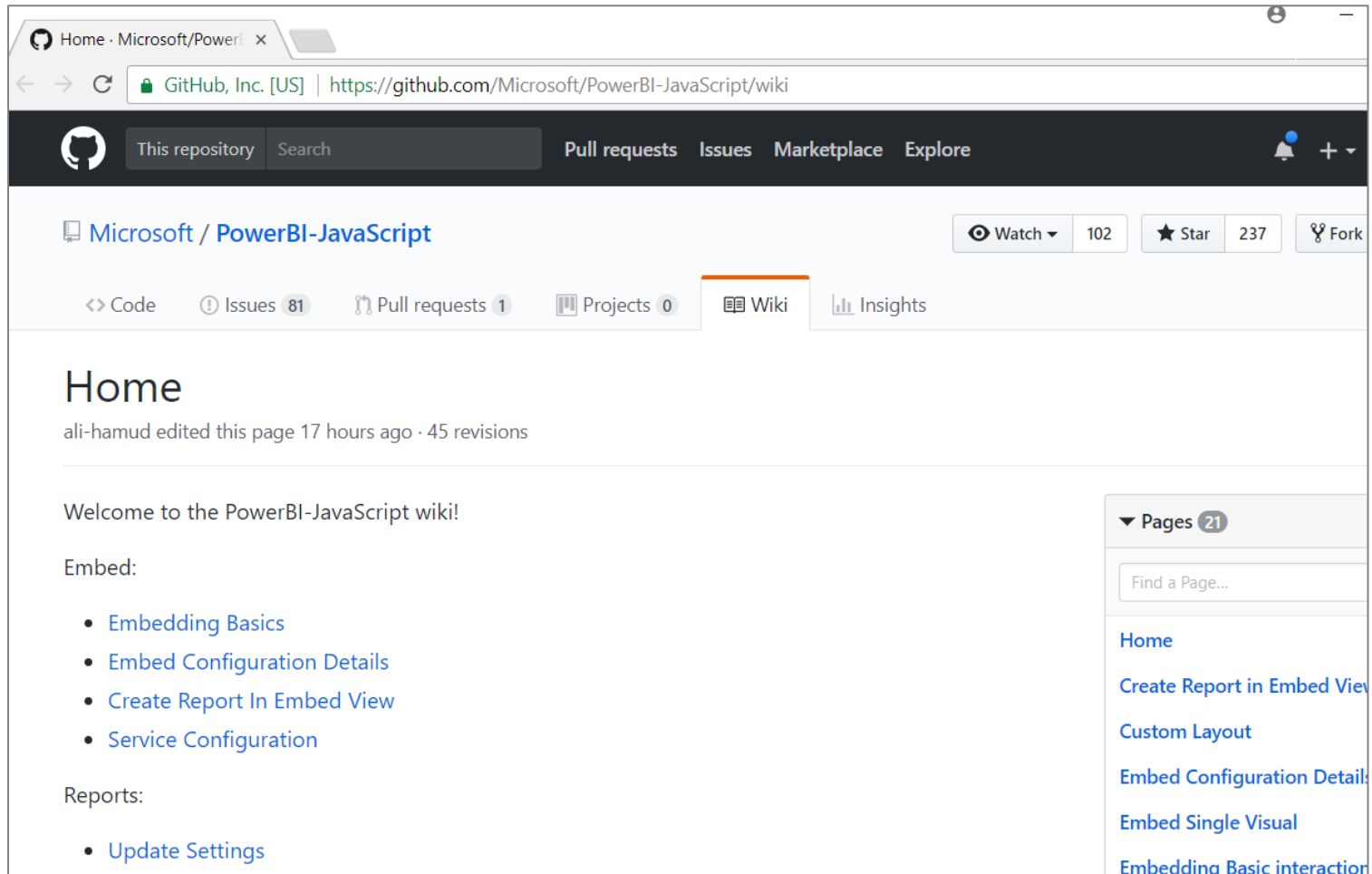
Agenda

- ✓ 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 "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 under "Embed:" (Embedding Basics, Embed Configuration Details, Create Report In Embed View, Service Configuration), and a list under "Reports:" (Update Settings). A sidebar on the right shows a list of 21 pages, including Home, Create Report in Embed View, Custom Layout, Embed Configuration Details, Embed Single Visual, and Embedding Basic interaction.

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_-WlrIR3K02A9x66gQzvWw0_76tmbySW6pbd7-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,
};

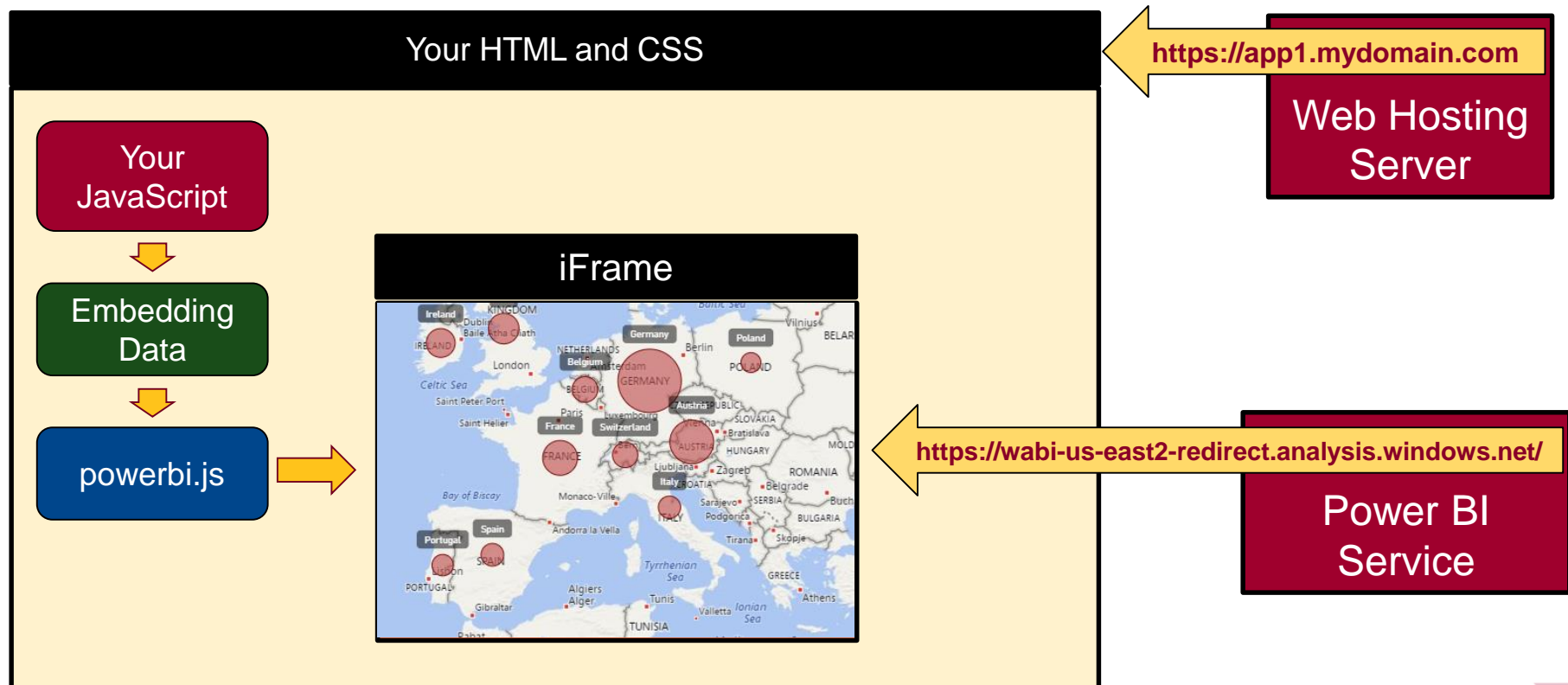
// Get a reference to the embedded report HTML element
var reportContainer = document.getElementById('embedContainer');

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



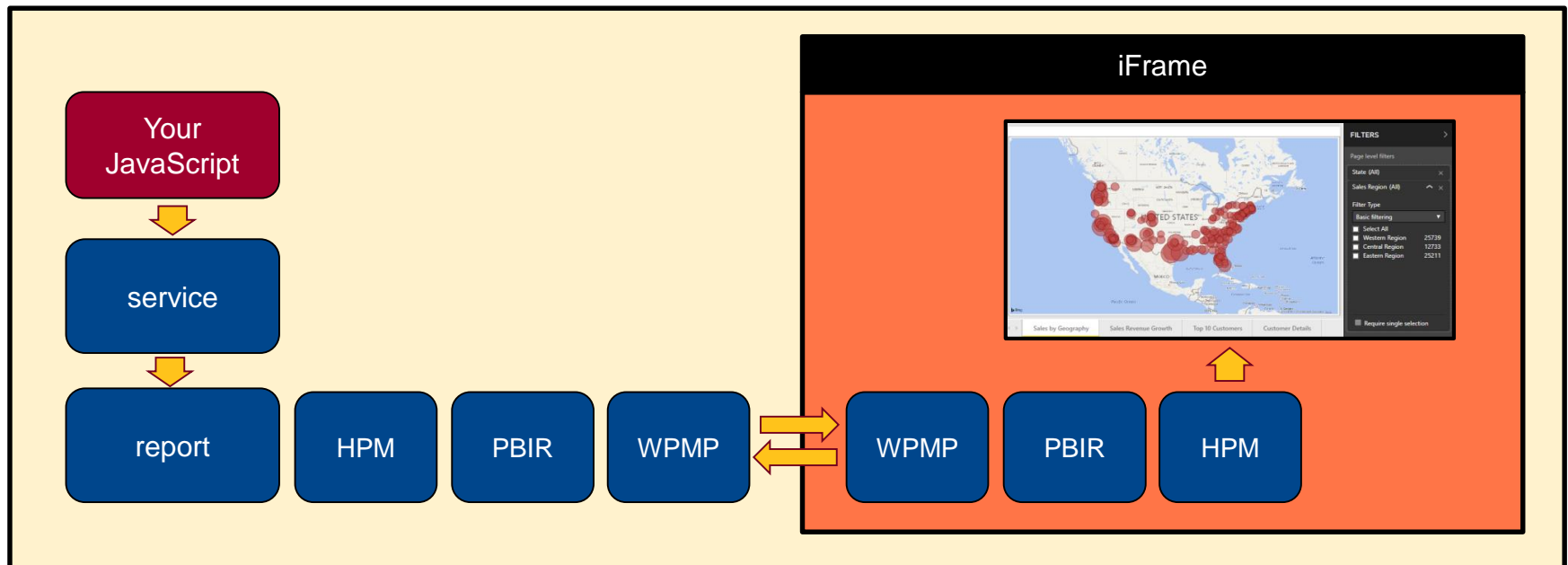
Report Embedding Architecture

- 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***



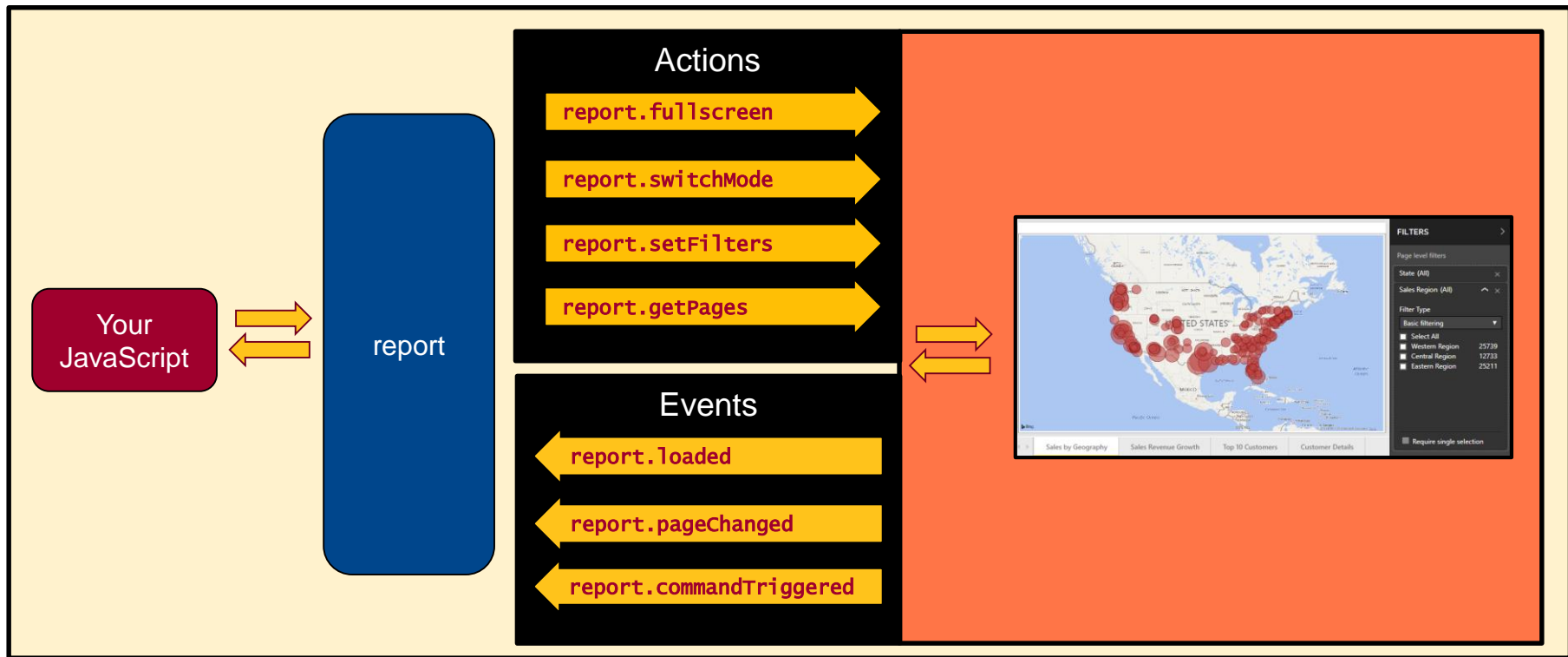
Post Message Communications Flow

- 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 page includes a navigation bar with links to Features, Business, Explore, Marketplace, and Pricing. The repository name is displayed, along with buttons for Watch (3), Star (0), and Fork (0). The main content area shows the repository description: "A sample application for developer's learning about Power BI Embedding". Below this, there are statistics: 4 commits, 1 branch, 0 releases, 1 contributor, and MIT license. A table lists the files and folders in the repository, including DemoPagesWithEmbedding, PBIX, and PowerBiEmbeddedScratchpad, all of which have been updated 8 days ago.

File/Folder	Update Type	Update Time
DemoPagesWithEmbedding	Updates	8 days ago
PBIX	Updates	7 days ago
PowerBiEmbeddedScratchpad	Updates	8 days ago





DEMO

The Power BI Embedded Scratchpad App

Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Programming with Power BI Service API
- ✓ Embedding with Power BI JavaScript API
- Calling the Power BI Service using AadHttpClient



Create a New SPFX Web Part Project

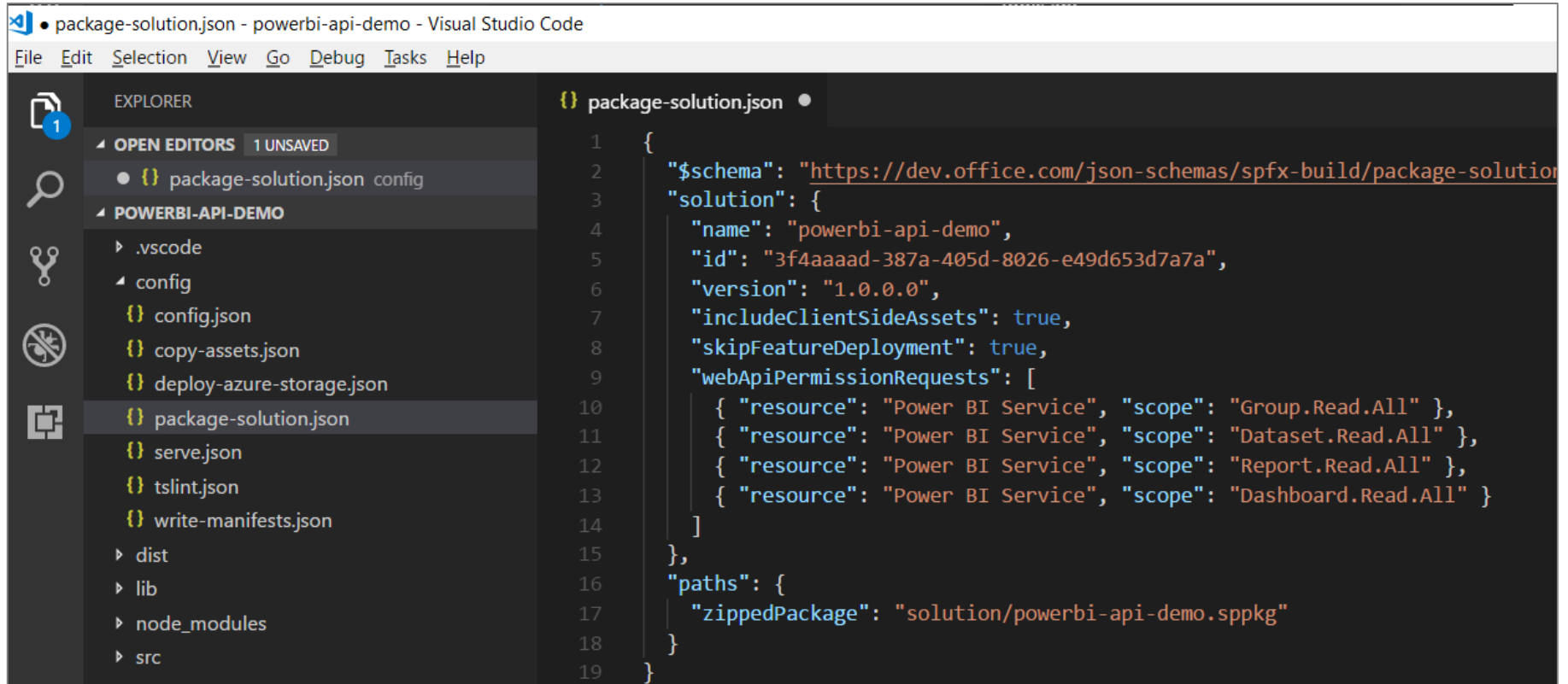
```
c:\Vegas\PBIESPFX\powerbi-api-demo>yo @microsoft/sharepoint

  Welcome to the
  SharePoint Client-side
  Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? powerbi-api-demo
? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)
? Where do you want to place the files? Use the current folder
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without running any feature deployment or adding apps in sites? Yes
? Which type of client-side component to create? WebPart
? What is your Web part name? reportlist
? What is your Web part description? A sample web part which calls the Power BI Service API
? Which framework would you like to use? (Use arrow keys)
> No JavaScript framework
  React
  Knockout
```



Configuring Web API Permissions

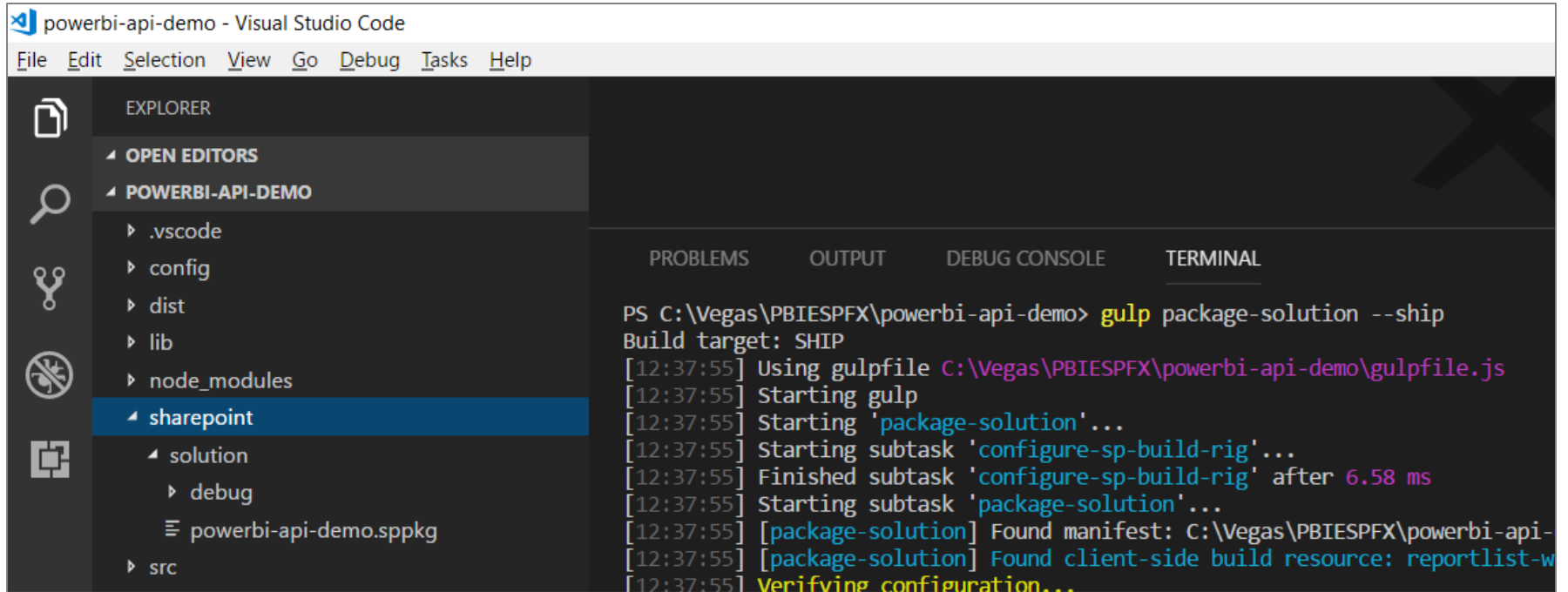


The screenshot shows the Visual Studio Code interface with the file explorer on the left and the editor on the right. The file explorer shows the project structure for 'package-solution.json - powerbi-api-demo'. The editor displays the contents of 'package-solution.json', which is a JSON file defining the solution configuration. The JSON includes a schema URL, solution name, ID, version, and a list of web API permission requests for the Power BI Service. The 'paths' section specifies the location of the zipped package.

```
1 {
2   "$schema": "https://dev.office.com/json-schemas/spfx-build/package-solution",
3   "solution": {
4     "name": "powerbi-api-demo",
5     "id": "3f4aaaad-387a-405d-8026-e49d653d7a7a",
6     "version": "1.0.0.0",
7     "includeClientSideAssets": true,
8     "skipFeatureDeployment": true,
9     "webApiPermissionRequests": [
10      { "resource": "Power BI Service", "scope": "Group.Read.All" },
11      { "resource": "Power BI Service", "scope": "Dataset.Read.All" },
12      { "resource": "Power BI Service", "scope": "Report.Read.All" },
13      { "resource": "Power BI Service", "scope": "Dashboard.Read.All" }
14    ]
15  },
16  "paths": {
17    "zippedPackage": "solution/powerbi-api-demo.sppkg"
18  }
19 }
```



Packaging Your SPFX Solution



powerbi-api-demo - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER

- OPEN EDITORS
- POWERBI-API-DEMO
 - .vscode
 - config
 - dist
 - lib
 - node_modules
 - sharepoint
 - solution
 - debug
 - powerbi-api-demo.sppkg
 - src

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Vegas\PBIESPFX\powerbi-api-demo> gulp package-solution --ship
Build target: SHIP
[12:37:55] Using gulpfile C:\Vegas\PBIESPFX\powerbi-api-demo\gulpfile.js
[12:37:55] Starting gulp
[12:37:55] Starting 'package-solution'...
[12:37:55] Starting subtask 'configure-sp-build-rig'...
[12:37:55] Finished subtask 'configure-sp-build-rig' after 6.58 ms
[12:37:55] Starting subtask 'package-solution'...
[12:37:55] [package-solution] Found manifest: C:\Vegas\PBIESPFX\powerbi-api-
[12:37:55] [package-solution] Found client-side build resource: reportlist-w
[12:37:55] Verifying configuration...
```



Deploy the Web Part to the App Gallery

The screenshot displays the SharePoint 'Apps for SharePoint' interface. At the top, there's a 'Home' link and the title 'Apps for SharePoint'. Below this, a navigation bar includes 'New', 'Upload', 'Sync', 'Share', and a 'More' dropdown. A search bar labeled 'Find a file' is also present. The main content area shows a table with columns: Title, Name, App Version, Edit, Product ID, Metadata Language, Default Metadata Language, Modified, Enabled, Valid App Package, and Deployed. A file explorer window is overlaid on the right, showing the 'solution' folder. Inside the 'solution' folder, there are two items: a 'debug' folder and a file named 'powerbi-api-demo.sppkg'.

Home

Apps for SharePoint ⓘ

+ New **↑** Upload **↻** Sync **↻** Share More ▾

All Apps Featured Apps Unavailable Apps ... Find a file 🔍

✓	📄	Title	Name	App Version	Edit	Product ID	Metadata Language	Default Metadata Language	Modified	Enabled	Valid App Package	Deployed
---	---	-------	------	-------------	------	------------	-------------------	---------------------------	----------	---------	-------------------	----------

File Explorer: solution

- Quick access
 - Desktop
 - Downloads
- debug
- powerbi-api-demo.sppkg



Configuring Trust

Do you trust powerbi-api-demo?



The client-side solution you are about to deploy contains full trust client side code. The components in the solution can, and usually do, run in full trust, and no resource usage restrictions are placed on them.

This client side solution will get content from the following domains:

SharePoint Online

☒ Make this solution available to all sites in the organization

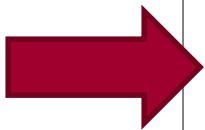
Please go to the Service Principal Permissions Management Page to approve pending permissions.



powerbi-api-demo

Deploy

Cancel



Granting Web API Permissions

API management

Control the third-party APIs that can be called by apps and custom script.

^ API name	Permission	Access
^ Pending approval (0)		
^ Approved (4)		
Power BI Service	Group.Read.All	Approved
Power BI Service	Group.Read.All Dataset.Read.All	Approved
Power BI Service	Group.Read.All Dataset.Read.All Report.	Approved
Power BI Service	Group.Read.All Dataset.Read.All Report.	Approved

[admin center](#)[feedback](#)



Calling Power BI API with AadHttpClient

```
import { AadHttpClient, HttpClientResponse } from '@microsoft/sp-http';

export default class ReportlistWebPart extends BaseClientSideWebPart<any> {

    private powerbiApiResourceId = "https://analysis.windows.net/powerbi/api";
    private pbiClient: AadHttpClient;

    protected onInit(): Promise<void> {
        this.pbiClient = new AadHttpClient(this.context.serviceScope, this.powerbiApiResourceId);
        return Promise.resolve();
    }

    public render(): void {
        var urlReports: string = "https://api.powerbi.com/v1.0/myorg/reports/";
        this.pbiClient.get(urlReports, AadHttpClient.configurations.v1)
            .then((res: HttpClientResponse): Promise<any> => {
                return res.json();
            })
            .then((reports: any): void => {
                this.domElement.innerHTML =
                    `<h2>Power BI Reports</h2>
                    <ul>
                        ${ reports.value.map(r => `<li><a href='${r.webUrl}' target='_blank' >${r.name}</a></li>`).join("") }
                    </ul>`;
            });
    }
}
```

Power BI Reports

- [Northwind Retro](#)
- [Wingtip Sales Analysis](#)

Installing the Power BI JavaScript API

- `npm install powerbi-client --save-dev`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Vegas\PBIESPFX\embed-report-demo> npm install powerbi-client --save
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4
)

+ powerbi-client@2.5.1
added 9 packages in 22.762s
```

```
‣ postcss-value-parser
‣ postcss-zindex
‣ powerbi-client
‣ powerbi-models
‣ powerbi-router
‣ prelude-ls
```

```
tsconfig.json x
1  {
2    "compilerOptions": {
3      "target": "es5",
4      "forceConsistentCasingInFileNames": true,
5      "module": "commonjs",
6      "jsx": "react",
7      "declaration": true,
8      "sourceMap": true,
9      "experimentalDecorators": true,
10     "skipLibCheck": true,
11     "typeRoots": [
12       "./node_modules/@types",
13       "./node_modules/@microsoft",
14       "./node_modules/powerbi-client",
15       "./node_modules/powerbi-models"
16     ],
17     "types": [
18       "es6-promise",
19       "webpack-env"
20     ],
21     "lib": [
22       "es5",
23       "dom",
24       "es2015.collection"
25     ]
26   }
27 }
```



```

public render(): void {
    let hostDiv: JQuery = $(this.domElement);
    let height: string = this.properties.reportHeight + "px";
    hostDiv.empty().css({ "margin": "0", "padding": "0", "height": height });

    var reqHeaders: HeadersInit = new Headers();
    reqHeaders.append("Accept", "*");
    this.pbiclient.get(this.reportUrl, AadHttpClient.configurations.v1, { headers: reqHeaders })
        .then((res: HttpClientResponse): Promise<any> => {
            return res.json();
        })
        .then((report: any): void => {
            console.log("begin embed...");
            var embedReportId: string = report.id;
            var embedUrl: string = report.embedUrl;
            var accessToken: string = window.sessionStorage["adal.access.token.keyhttps://analysis.windows.net/powerbi/api"];
            // Get models object to access enums for embed configuration
            var models = pbimodels;

            var config: any = {
                type: 'report',
                id: embedReportId,
                embedUrl: embedUrl,
                accessToken: accessToken,
                tokenType: models.TokenType.Aad,
                permissions: models.Permissions.All,
                viewMode: models.ViewMode.View,
                settings: {
                    filterPaneEnabled: false,
                    navContentPaneEnabled: this.properties.showPageTabs,
                }
            };
            window.powerbi.reset(this.domElement);
            window.powerbi.embed(this.domElement, config);
        });
}

```

Summary

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Dedicated Capacities
- ✓ Programming with Power BI Service API
- ✓ Embedding with Power BI JavaScript API
- ✓ Calling the Power BI Service using AadHttpClient

