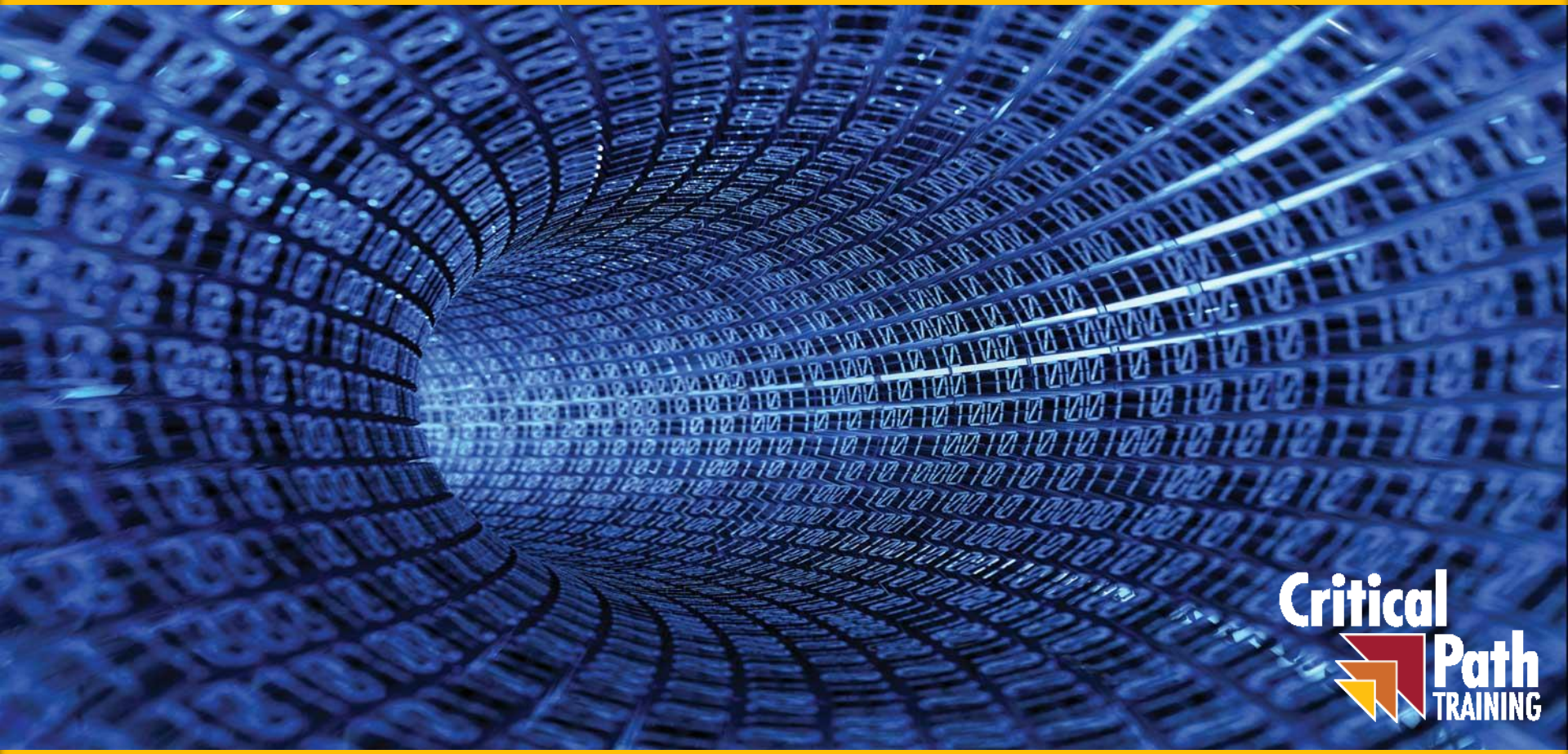
Power BI Embedding

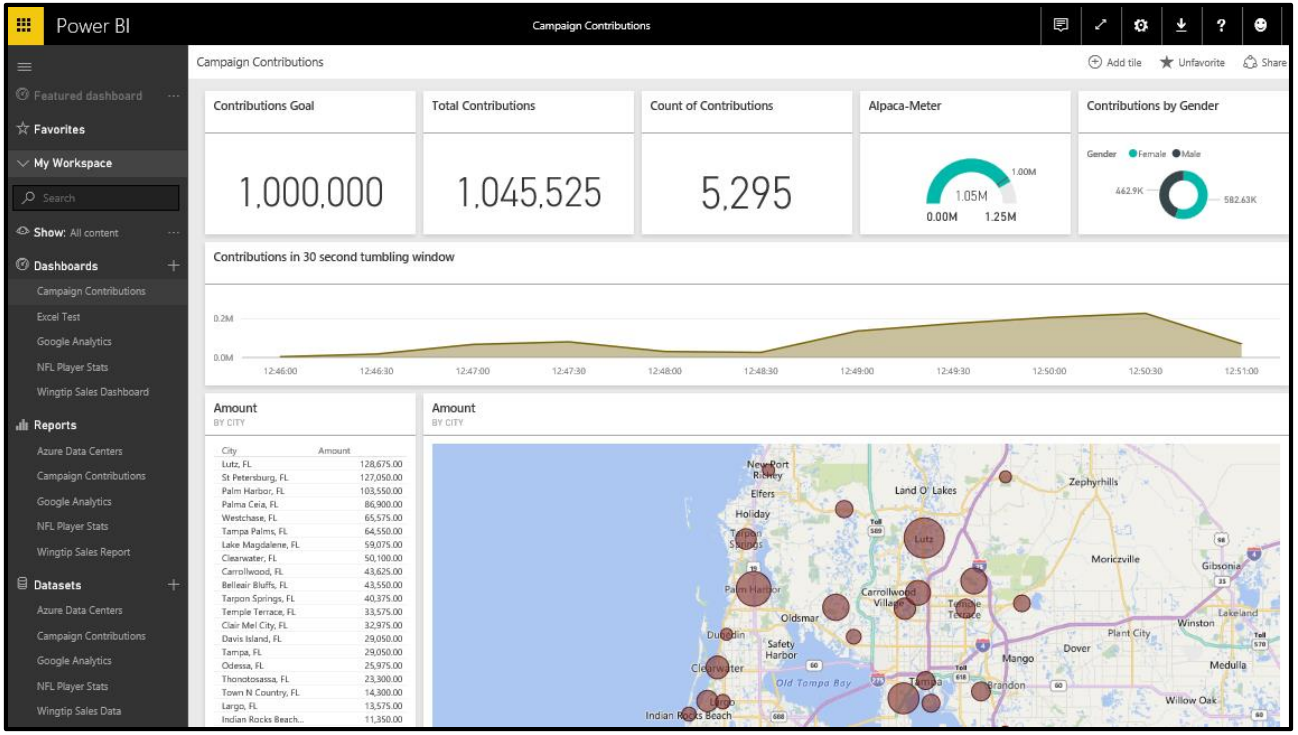


Agenda

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

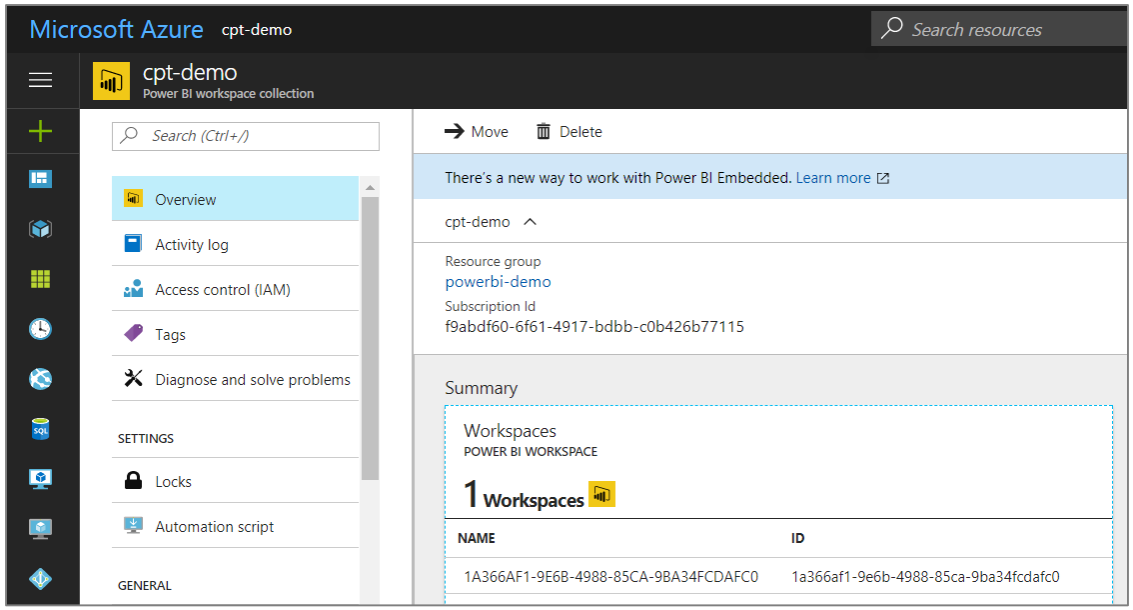
The Power BI Service

- 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



What ~~is~~ was Power BI Embedded V1?

- Power BI Embedded V1 is an Azure Service
  - PBI Embedded service that is provisioned on-demand
  - Service provisioned in terms of **workspace collections**
  - PBI Embedded service required an Azure subscription
  - Pricing model based on number of report sessions



Reflecting on Power BI Embedded V1?

- Good Points about Power BI Embedded V1
  - It eliminates need for Power BI license for each user
  - It decouples user security from app security
  - It opens up PBI platform to commercial applications
- Pain Points with Power BI Embedded V1
  - Requires developers to have Azure subscriptions
  - No out-of-box UX to upload and manage PBIX files
  - It uses separate APIs from Power BI Service API
  - Cannot estimate costs with per-session pricing model
  - It’s deprecated and not available to new customers

Power BI Embedded Version 2

- Power BI Embedded V2 has same good points as V1
  - It eliminates need for Power BI license for each user
  - It decouples user security from app security
  - It opens up PBI platform to commercial applications
- Power BI Embedded V2 significantly improves upon V1
  - Embedding features all available through Power BI Service API
  - Standard PBI UX used to upload and manage PBIX files
  - New pricing models allow for predictable costs per month
  - No need to create, manage and monitor any Azure services
- The term “Power BI Embedded” is now ambiguous
  - Better to refer to the “Embedding features in Power BI”

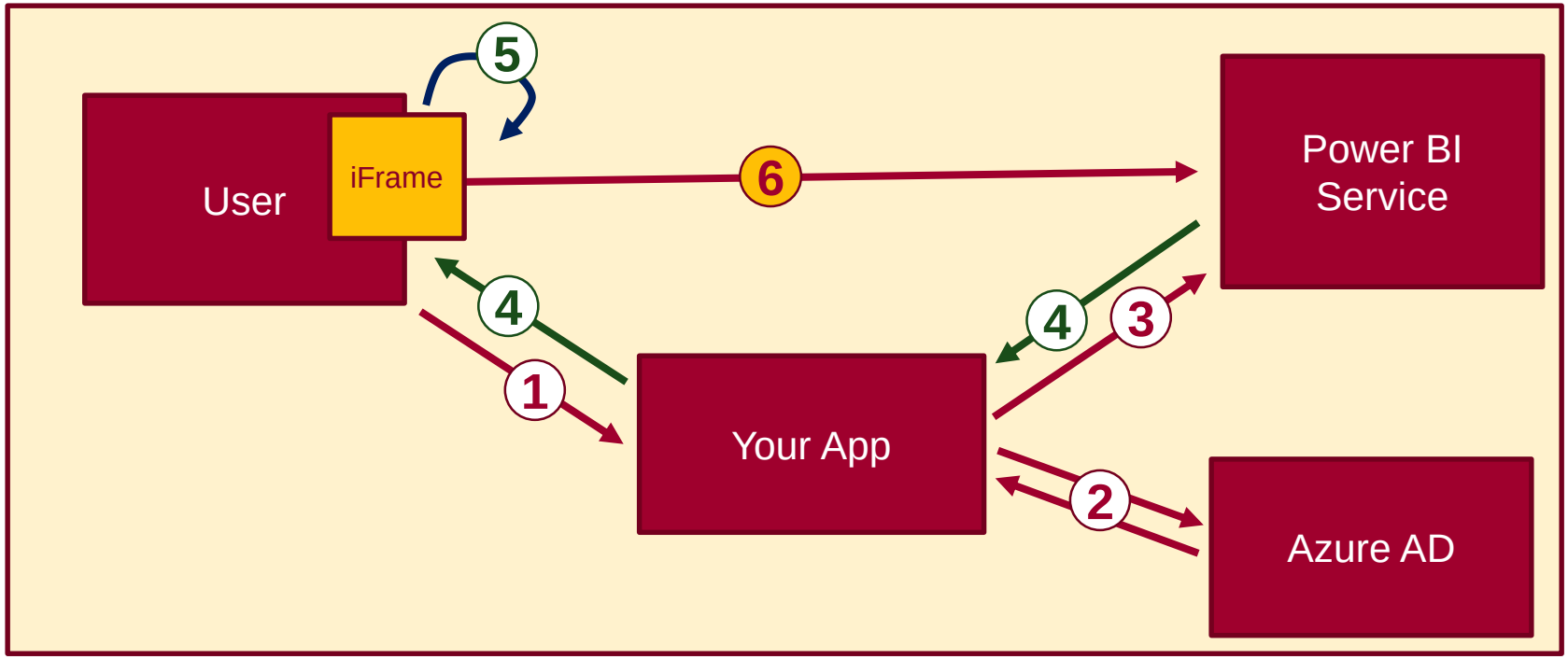
The Power BI Service API

- 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

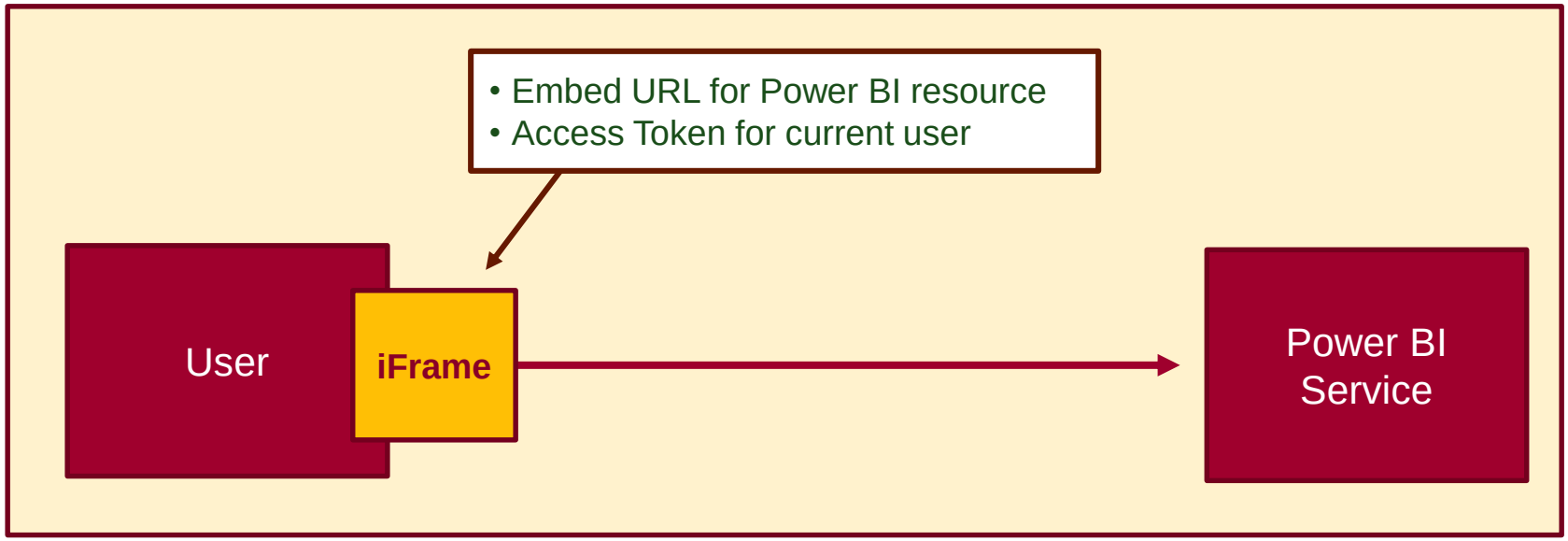
Power BI Embedding – The Big Picture

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



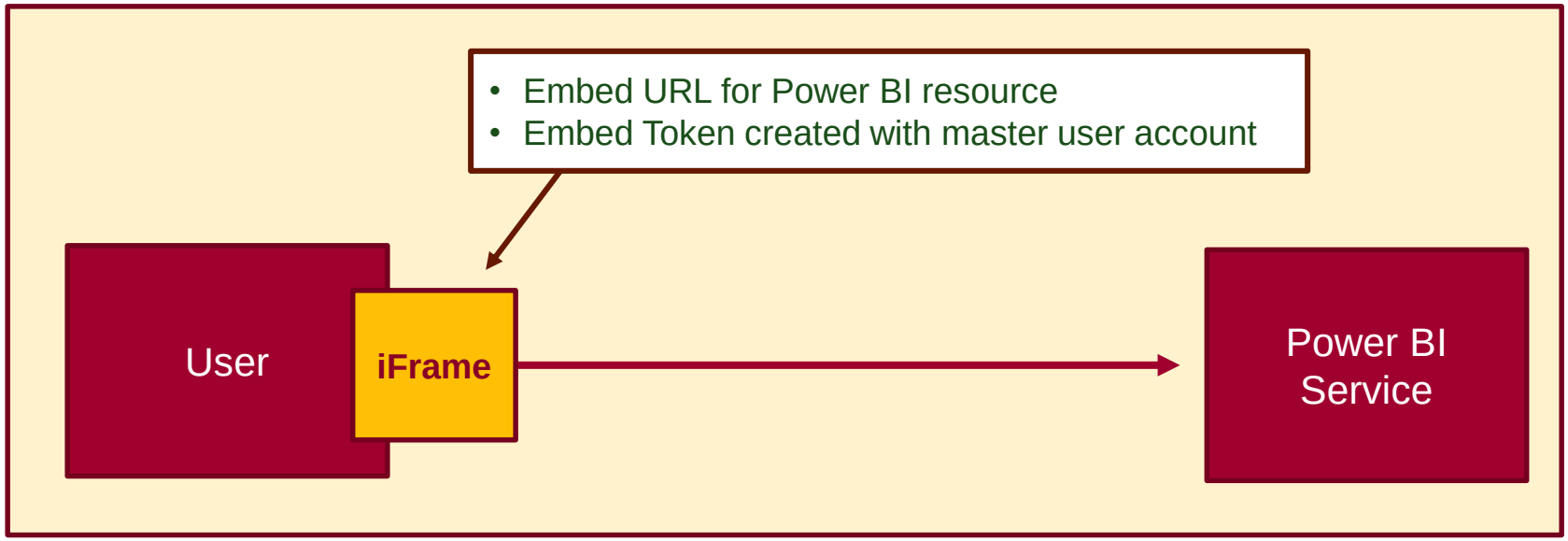
First Party Embedding

- 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



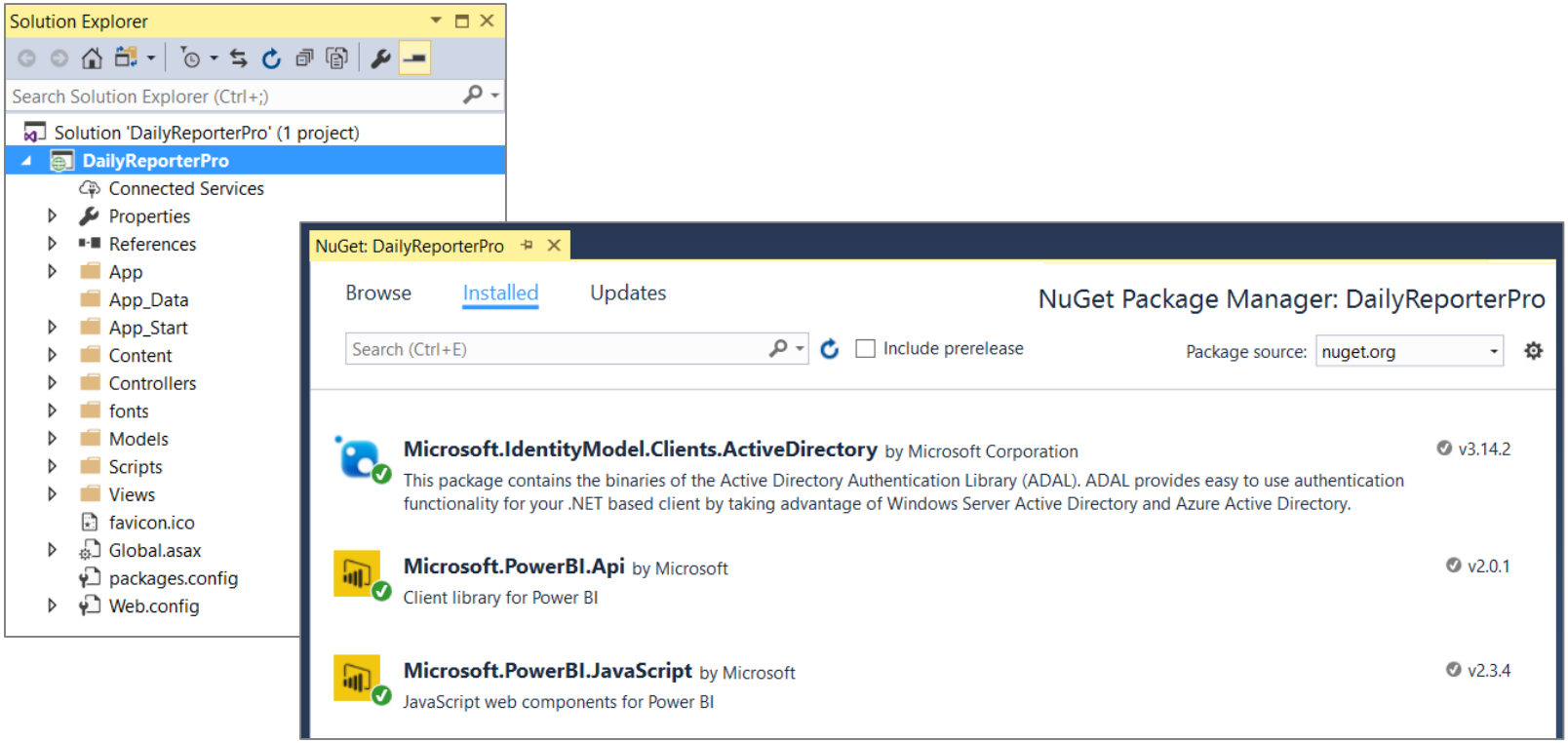
First Party vs Third Party Embedding

- What scenarios use first party embedding?
  - Organizations where users have Power BI licenses
  - Users can already access Power BI with browser
  - 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 Premium Capacities
- Authentication with Azure Active Directory
- Programming with Power BI Service API
- Working with Embeddable Resources
- Embedding with Power BI JavaScript API

Power BI Premium

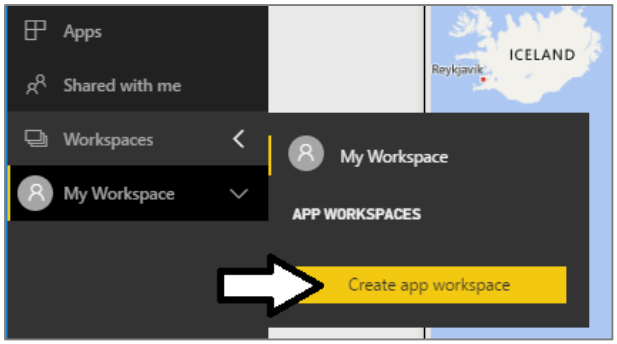
- Microsoft initially offered two Power BI licensing options
  - Power BI Free license
  - Power BI Pro license ($10/month)
  - All Power BI resources and processing runs in shared capacity

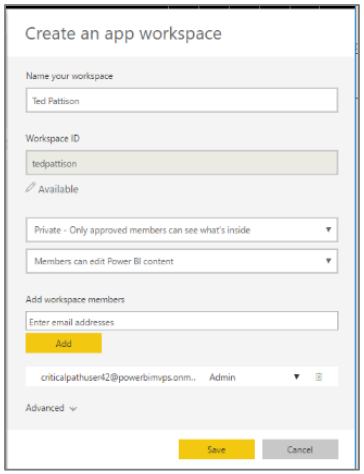
- In May 2017, Microsoft introduced Power BI Premium licensing
  - Power BI Premium customers can create premium capacities
  - Premium capacities useful to organization with many read-only users
  - Premium capacities used by ISVs to reach non-licensed users

- Power BI Premium details and pricing are in flux
  - More info at https://powerbi.microsoft.com/en-us/pricing/

Understanding App Workspaces

- App workspaces used to deploy custom solutions
  - App workspaces required for team-based development
  - App workspace can be secured using private membership
  - App workspace used to publish apps for licensed users
- App workspaces required for 3rd party embedding
  - App workspace must be added to premium capacity
  - Master user account must be configured as app workspace admin





Premium Capacities

- Power BI workspaces run in two possible environments
  - Shared Capacities
  - Premium Capacities *(formerly known as dedicated capacities)*

- Premium capacity acts as dedicated resource
  - Premium capacity only used by single organization
  - PBIX file uploads not limited to 1GB
  - Data refresh frequency can exceed 8 times per day
  - Each premium capacity defines its own set of admins
  - *Premium capacity required to share with users without pro license*

Premium Capacity Nodes

- Power BI Premium Purchased using Nodes
  - Node type defines v-core and RAM capabilities
  - P nodes used for embedded or service deployments
  - EM nodes used only for embedded deployments

Capacity Node	Total cores	Backend Cores	Frontend Cores	Direct Query Limits	Page renders/hour
EM1	1 v-cores	.5 cores, 3GB RAM	.5 cores		1-300
EM2	2 v-cores	1 core, 5GB RAM	1 core		301-600
EM3	4 v-cores	2 cores, 10GB RAM	2 cores		601-1,200
P1	8 v-cores	4 cores, 25GB RAM	4 cores	30 per second	1,201-2,400
P2	16 v-cores	8 cores, 50GB RAM	8 cores	60 per second	2,401-4,800
P3	32 v-cores	16 cores, 100GB RAM	16 cores	120 per second	4,801-9600

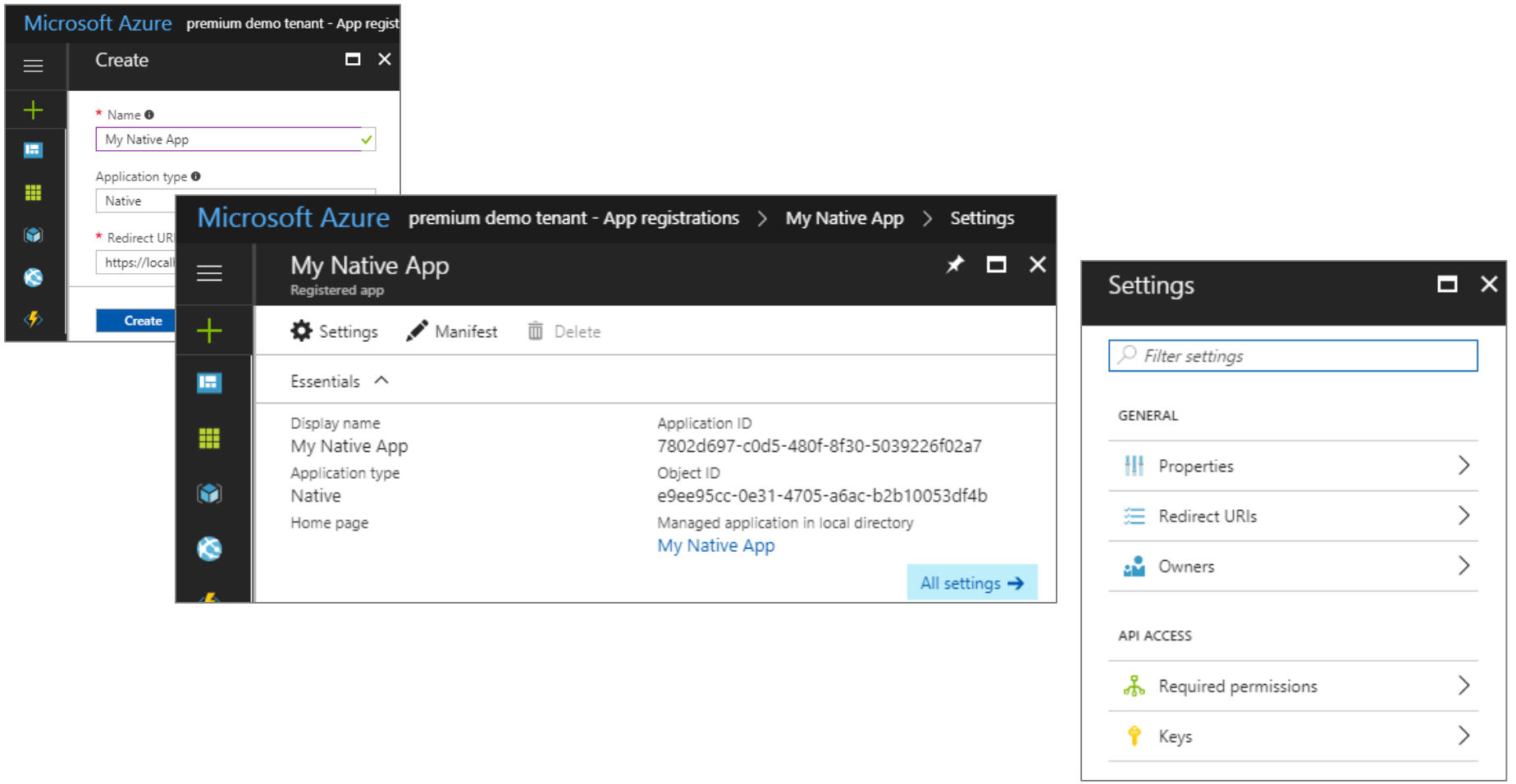
Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ App Workspaces and Premium 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

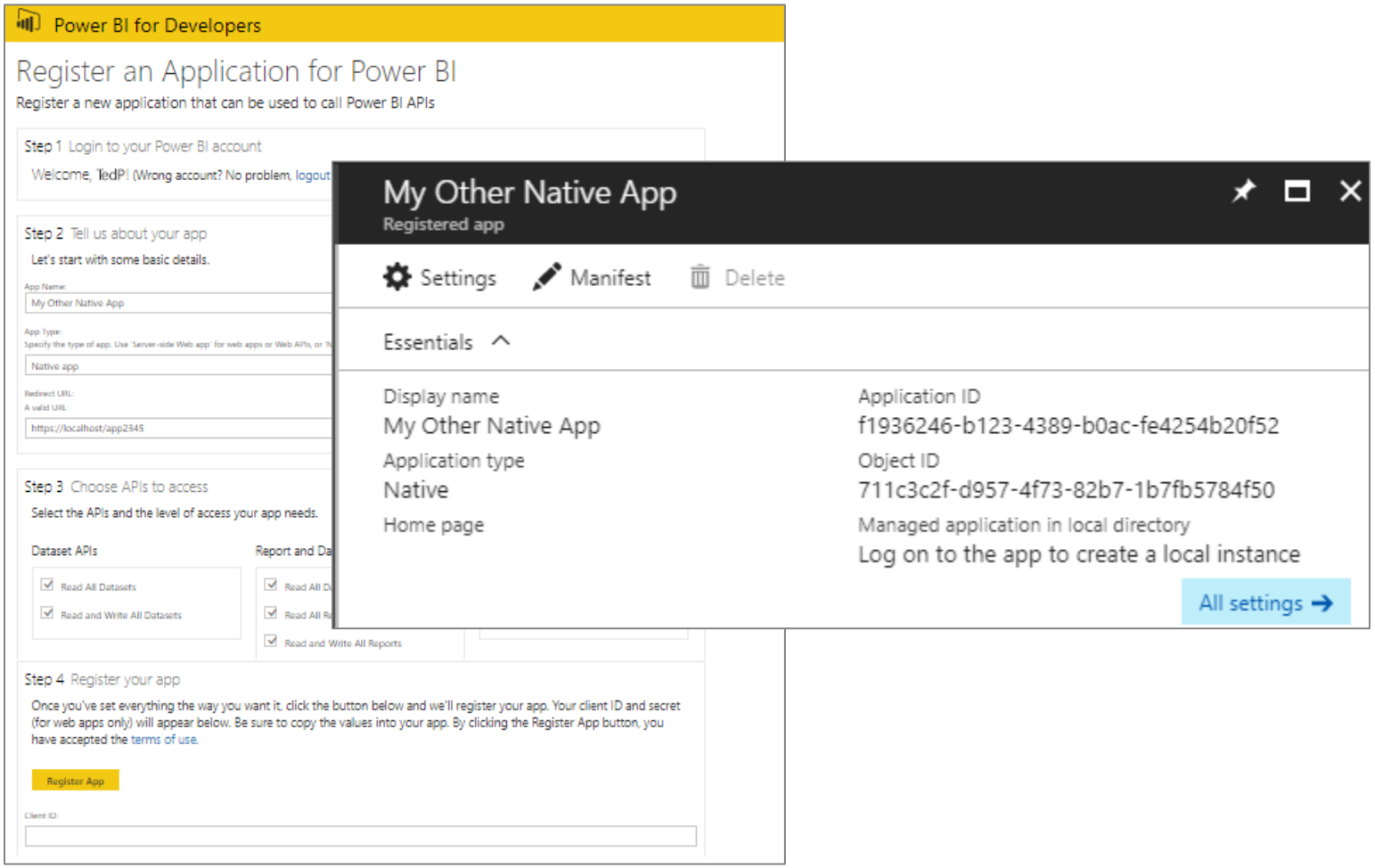
- Azure AD used to authenticate users and apps
  - PBI licenses are assigned to Azure AD user accounts
  - Organization owns a tenant (i.e. directory)
  - AAD tenant contains user accounts and groups
  - AAD tenant contains set of registered applications
- You must register your application with Azure 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



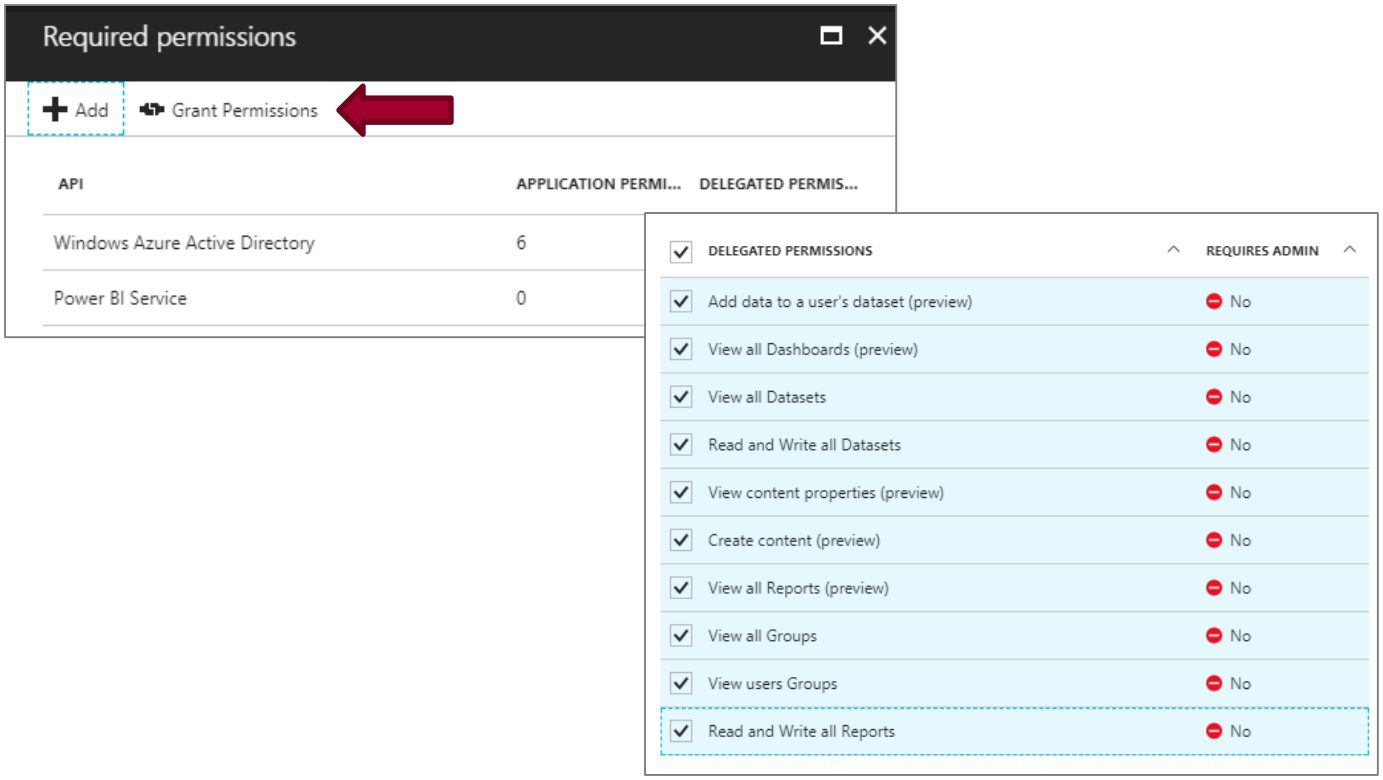
Power BI App Registration Page

- https://app.powerbi.com/apps



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



Authentication Flows

- Client Credentials Grant Flow *(confidential client)*
  - Authentication based on SSL certificate with public-private key pair
  - Used to obtain access token when using app-only permissions
- Authorization Code Grant Flow *(confidential client)*
  - Client first obtains authorization code then access token
  - Server-side application code never sees user's password
- Implicit Grant Flow *(public client)*
  - Used in SPAs built with JavaScript and AngularJS
  - Application obtains access token w/o acquiring authorization code
- User Credentials Flow *(public client)*
  - Used in Native clients to obtain access code
  - Requires passing user name and password


DEMO
Registering an App with Azure AD

1st Party Embedding vs 3rd Party Embedding

	1st Part Embedding	3rd Party Embedding
Authentication flow	Authentication Code 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

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="appWorkspaceId" value="4baab6c0-87c5-4a2a-a73e-1f97adcc6123" />

    <!-- consider a secure approach for password management such as Azure Key Vault  -->
    <add key="pbiUserName" value="MasterUser@YourTenant.onMicrosoft.com" />
    <add key="pbiUserPassword" value="hackMEeyeDairU" />

  </appSettings>
```

```
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

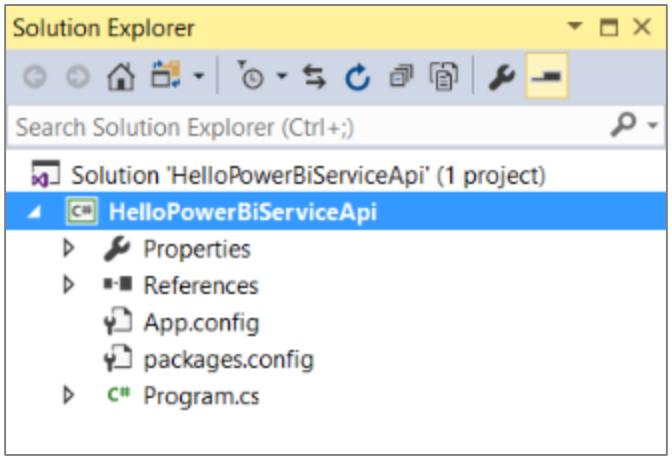
```
static string GetAccessToken() {
  AuthenticationContext authContext = new AuthenticationContext(aadAuthorizationEndpoint);
  var userCredentials = new UserPasswordCredential(pbiUserName, pbiUserPassword);

  // 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;
}
```

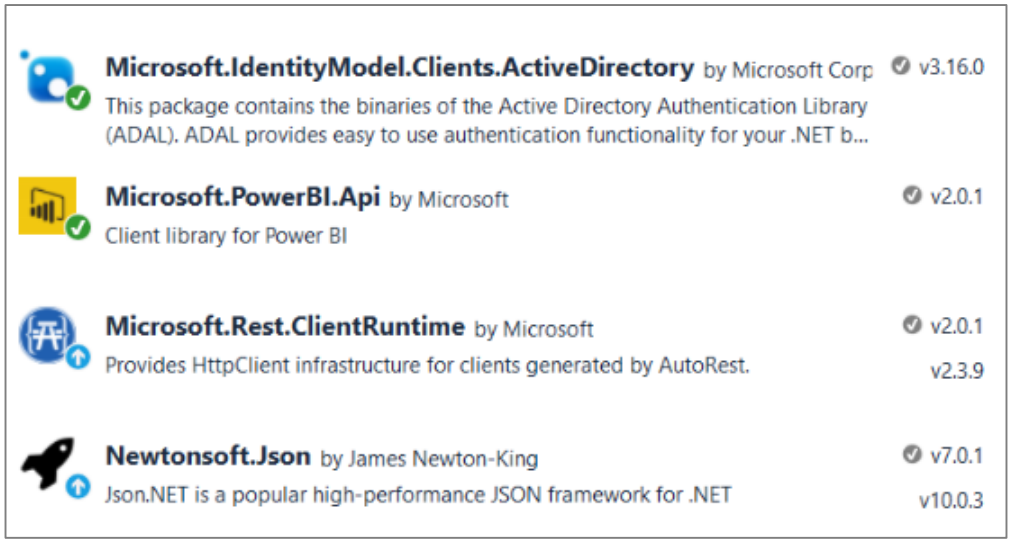
Agenda

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

HelloPowerBiServiceApi Demo

- Let's get started with a simple C# console app
  - NuGet packages added for ADAL and Power BI Service 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 a Instance of PowerBIClient

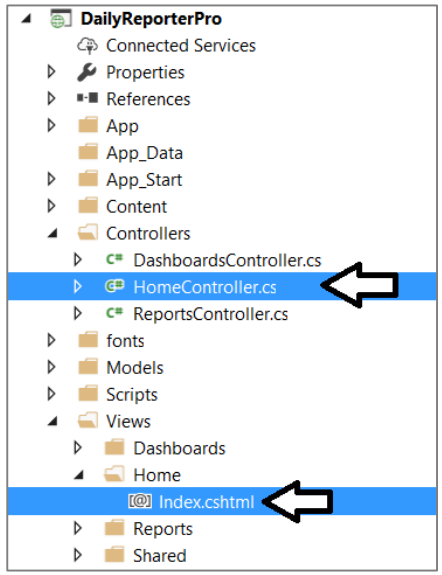
- PowerBIClient object serves as top-level object
  - Used to execute calls against Power BI Service
  - Initialized with function to retrieve AAD access token

```
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);
  }
}
```


DEMO
Programming the Power BI Service API

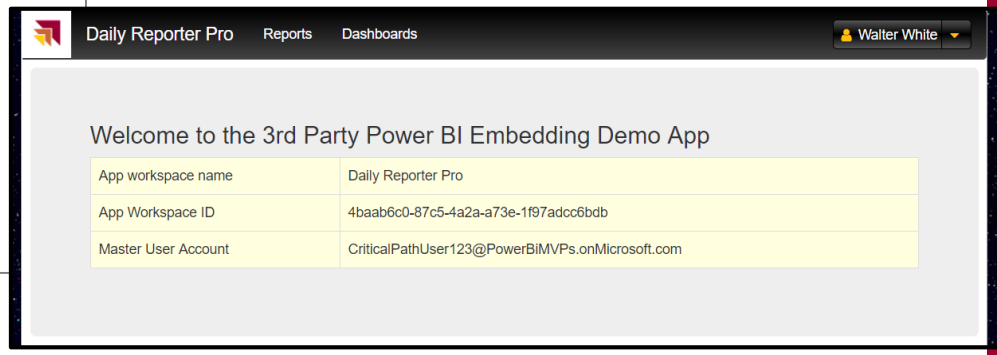
MVC Controllers and Views

```
public class HomeController : Controller {

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

}
```



Index.cshtml

```
@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 Premium 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
   - Provides user with full interactive experience
   - Allows editing existing reports & creating new reports
2. Dashboards
   - Provides user with limited interactive experience
   - Provides support for real-time dashboards
3. Dashboard Tiles
   - Provides flexibility to embed selected tiles
   - No support for tiles which receive real-time updates

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
  - *Support for generating tokens using RLS available any day now*

```
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 Premium Capacities
- ✓ Authentication with Azure Active Directory
- ✓ Programming with Power BI Service API
- ✓ Working with Embeddable Resources
- ➢ Embedding with Power BI JavaScript API

Embedding Data in MVC View

Index.cshtml

```
@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>
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- App_Data
- App_Start
- Content
- Controllers
- fonts
- Models
- Scripts
- Views
  - Dashboards
  - Home
    - Index.cshtml
  - Reports
    - Index.cshtml
  - Shared
  - _ViewStart.cshtml

Properties

App.ts | Index.cshtml

```
class PowerBIEmbeddingManagerClient {

    static embedReport = (reportId, embedUrl, accessTok

    static createReport = (datasetId, embedUrl, accessT

    static embedDashboard = (dashboardId, embedUrl, acc

}
```


DEMO
Programming the Power BI JavaScript API

Summary

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