

Microsoft Power BI

1

Power BI Dev Camp – Session 1 Developing for Power BI with .NET Core

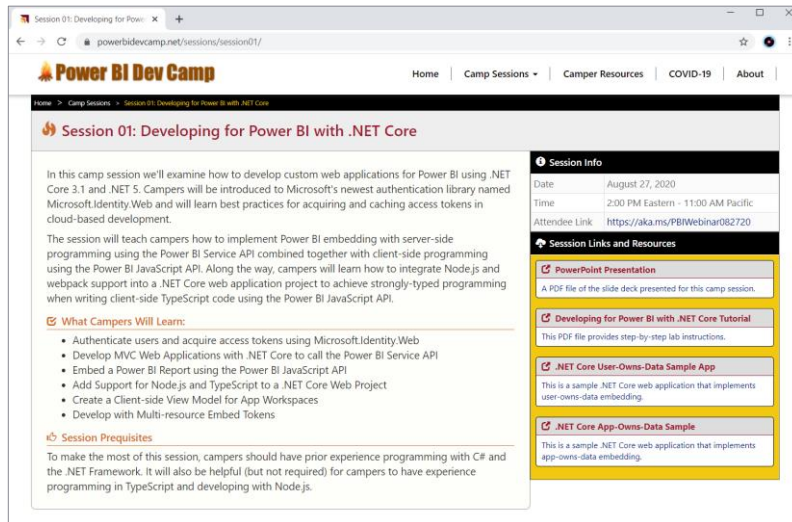
Ted Pattison

Principal Program Manager
Customer Advisory Team (CAT) at Microsoft

2

Welcome to Power BI Dev Camp

- Power BI Dev Camp Portal - <https://powerbidevcamp.net>



3

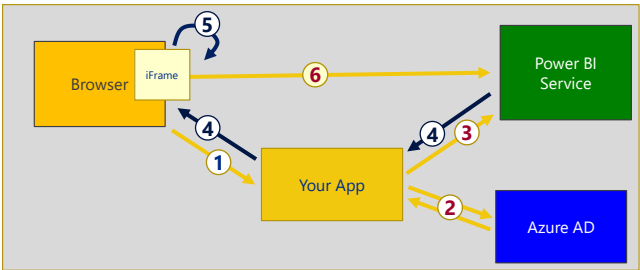
Agenda

- Tutorial Introduction
 - Developing with .NET Core
 - Introducing Microsoft.Identity.Web
 - Calling the Power BI Service API
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

4

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

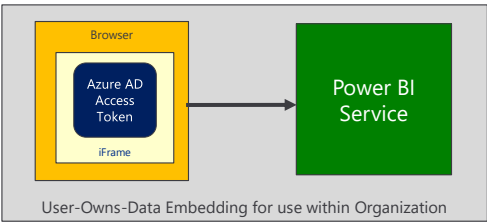


5

Choosing the Correct Embedding Model

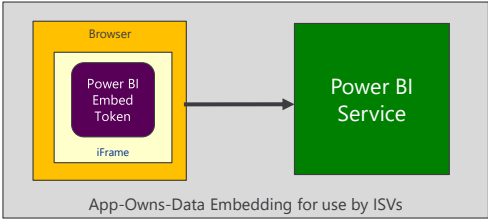
User-Owns-Data Embedding

- 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



App-Owns-Data Embedding

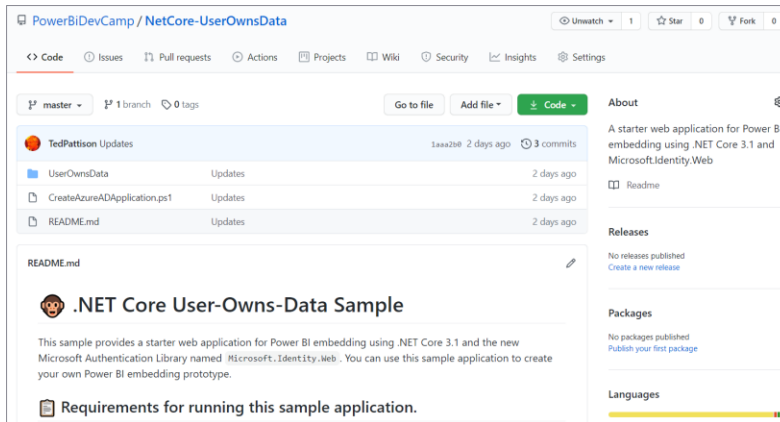
- No users require Power BI license
- Useful in commercial applications
- App authenticates with app-only identity
- Your code runs with admin permissions
- Embed token passed to browser



6

User-Owns-Data Sample Application

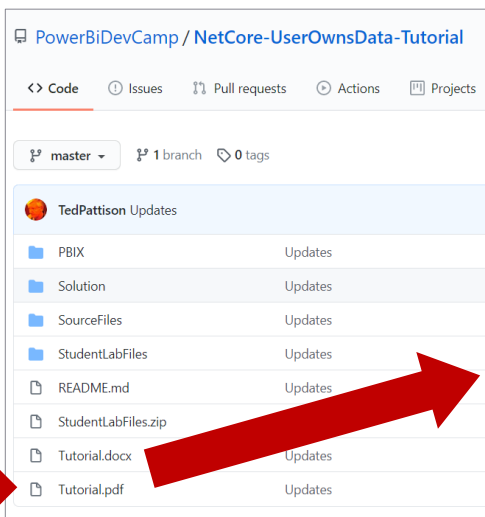
- Stored in a GitHub repository for easy download
- <https://github.com/PowerBIDevCamp/NetCore-UserOwnsData>



7

Tutorial: Developing for Power BI using .NET Core

- <https://github.com/PowerBIDevCamp/NetCore-UserOwnsData-Tutorial>



Developing for Power BI using .NET Core

In this lab, you will create a new .NET Core project for a custom web application and then you will go through the steps required to implement Power BI embedding. You will use the new Microsoft authentication library named *Microsoft.Identity.Web* to provide an interactive login experience and to acquire access tokens which you will need to call the Power BI Service API. After that, you will write the server-side C# code and the client-side JavaScript code to embed a simple Power BI report on a custom Web page. In the later exercises of this lab, you will add project-level support for Node.js, TypeScript and webpack so that you can migrate the client-side code from JavaScript to TypeScript so that your code receives the benefits of strong typing, IntelliSense and compile-time type checks.

To complete this lab, your developer workstation must configure to allow the execution of PowerShell scripts. Your developer workstation must also have the following software and developer tools installed.

- 1) PowerShell cmdlet library for AzureAD - [download](#)
- 2) DOTNET Core SDK 3.1 or later - [download](#)
- 3) Node.js - [download](#)
- 4) Visual Studio Code - [download](#)
- 5) Visual Studio 2019 (optional) - [download](#)

Please refer to this [setup document](#) if you need more detail on how to configure your developer workstation to work on this tutorial.

Exercise 1: Create a New .NET Core MVC Web Application Project

In this exercise, you will begin by copy the student files into a local folder on your student workstation. After that, you will use the .NET Core CLI to create a new .NET Core project for an MVC web application.

1. Download the student lab files to a local folder on your developer workstation.
 - a) Create a new top-level folder on your workstation named **DevCamp** at a location such as **c:\DevCamp**.
 - b) Download the ZIP archive with the student lab files from GitHub by clicking the following link.
<https://github.com/PowerBIDevCamp/NetCore-UserOwnsData-Tutorial/raw/master/StudentLabFiles.zip>
 - c) Extract the **StudentLabFiles** folder from **StudentLabFiles.zip** into a to a local folder such as **c:\DevCamp\StudentLabFiles**.
 - d) The **StudentLabFiles** folder should contain the set of files shown in the following screenshot.

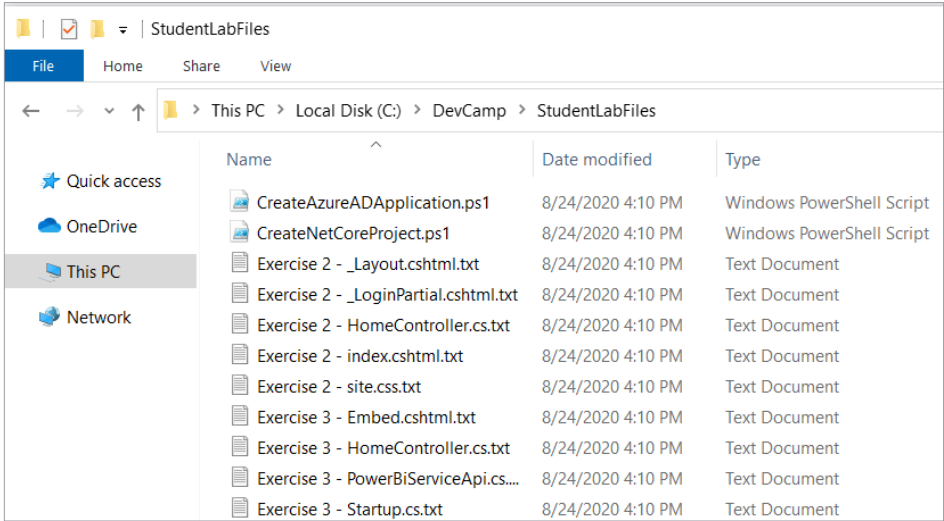
8

Tutorial Exercise Flow

- Exercise 1: Create a New .NET Core MVC Web Application Project
- Exercise 2: Implement User Login using Microsoft.Identity.Web
- Exercise 3: Call the Power BI Service API
- Exercise 4: Embedding a Report using powerbi.js
- Exercise 5: Adding TypeScript Support to a .NET Core Project
- Exercise 6: Creating a View Model for App Workspaces

9

Copying-and-Pasting Code



10

Agenda

- ✓ Tutorial Introduction
- Developing with .NET Core
 - Introducing `Microsoft.Identity.Web`
 - Calling the Power BI Service API
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

11

Install .NET Core SDK

<https://dotnet.microsoft.com/download>

The image is a side-by-side comparison of .NET Core and .NET Framework. It features a light blue background with a white grid. At the top, there are four tabs: 'Windows' (highlighted with a blue underline), 'Linux', 'macOS', and 'Docker'. The left column is for .NET Core, showing the '.NET Core' logo, the title '.NET Core 3.1', a description that it is a cross-platform version for building websites, services, and console apps, and three download links: 'Run Apps', 'Build Apps', and 'Advanced'. The right column is for .NET Framework, showing the '.NET Framework' logo, the title '.NET Framework 4.8', a description that it is a Windows-only version for building any type of app that runs on Windows, and three download links: 'Run Apps', 'Build Apps', and 'Advanced'.

12

.NET Core CLI

- The .NET Core command-line interface (CLI)
 - Cross-platform toolchain for creating, debugging and publishing applications
 - Create new applications using **dotnet new** command
 - Add NuGet packages using **dotnet add package** command

```
CreateNetCoreProject.ps1 X
dotnet new mvc --auth SingleOrg --framework netcoreapp3.1
dotnet remove package Microsoft.AspNetCore.Authentication.AzureAD.UI

# update to latest available version of Microsoft.Identity.Web
dotnet add package Microsoft.Identity.Web -v 0.3.0-preview
dotnet add package Microsoft.Identity.Web.UI -v 0.3.0-preview
dotnet add package Microsoft.PowerBi.Api
```

13

Understanding Dependency Injection is Essential

- Services are registered on application start up
- Services injected into classes using parameterized constructors

```
appsettings.json X
{
  "MySettings": {
    "FavoriteColor": "Red, no Blue"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Dependency injection

```
namespace UserOwnsData {
    public class Startup {
        public IConfiguration Configuration { get; }
        public string FavoriteColor;

        // pass Configuration service to constructor using dependency injection
        public Startup(IConfiguration configuration) {
            // make Configuration service available to all methods in this class
            Configuration = configuration;

            // use Configuration service to retrieve app setting
            this.FavoriteColor = configuration["MySettings:FavoriteColor"];
        }
    }
}
```

14

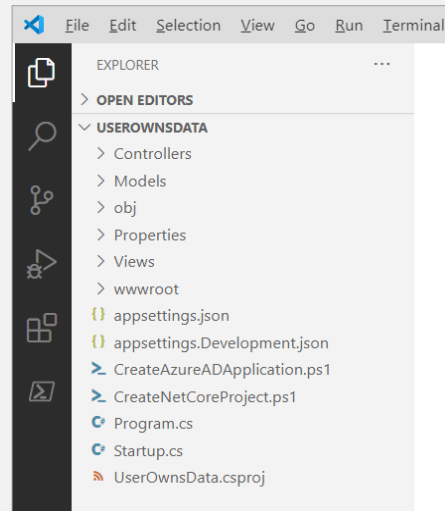
Exercise 1

Create a New .NET Core MVC Web Application Project

```
CreateNetCoreProject.ps1 X
dotnet new mvc --auth SingleOrg --framework netcoreapp3.1
dotnet remove package Microsoft.AspNetCore.Authentication.AzureAD.UI

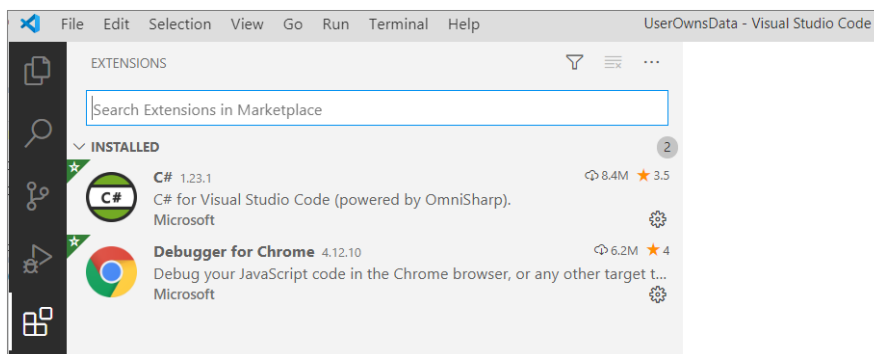
# update to latest available version of Microsoft.Identity.Web
dotnet add package Microsoft.Identity.Web -v 0.3.0-preview
dotnet add package Microsoft.Identity.Web.UI -v 0.3.0-preview
dotnet add package Microsoft.PowerBi.Api
```

```
Windows PowerShell
PS C:\Users\TedP> Set-Location -Path C:\DevCamp\UserOwnsData\
PS C:\DevCamp\UserOwnsData> .\CreateNetCoreProject.ps1
```



15

C# Extension for Visual Studio Code



16

Agenda

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- Introducing `Microsoft.Identity.Web`
 - Calling the Power BI Service API
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

17

Introducing `Microsoft.Identity.Web`

- What is `Microsoft.Identity.Web`
 - Set of components and classes to assist developers
 - Used to perform authentication in Web applications and Web APIs
 - Used to acquire access tokens
 - Used to implement token caching
- When to use `Microsoft.Identity.Web`
 - In Web application and Web APIs built on .NET Core 3.1 and .NET 5
 - `Microsoft.Identity.Web` is currently in preview
 - Scheduled for release with .NET 5 in November 2020
 - More info at <https://github.com/AzureAD/microsoft-identity-web/wiki>

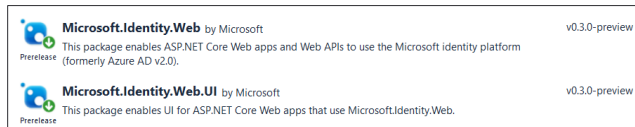
18

Microsoft Authentication Libraries

- What did we do before `Microsoft.Identity.Web`?
 - Use `Microsoft.Identity.Client` (MSAL) to acquire and cache access tokens
 - Use OWIN Middleware components to implement OpenID connect



- `Microsoft.Identity.Web` supports OpenID Connect and token acquisition



19

Exercise 2

Implement User Login using `Microsoft.Identity.Web`

1. Create a Confidential Client Application in Azure AD
2. Implement OpenID Connect using `Microsoft.Identity.Web`
3. Test Secured Routes in an ASP.NET Core Web Application

20

Creating Azure AD Application with PowerShell

```
#authResult = Connect-AzureAD
$UserAccountID = $authResult.Account.Id
$User = Get-AzureADUser -ObjectId $UserAccountID

$appDisplayName = "User-Owns-Data Sample App"
$replyUrl = "https://localhost:44300/signin-oidc"

# create app secret
$NewGuid = New-Guid
$AppSecret = ([System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(($NewGuid))))+"-"
$StartDate = Get-Date
$PasswordCredential = New-Object -Typename Microsoft.Open.AzureAD.Model.PasswordCredential
$PasswordCredential.StartDate = $StartDate
$PasswordCredential.EndDate = $StartDate.AddYears(1)
$PasswordCredential.KeyId = $NewGuid
$PasswordCredential.Value = $AppSecret

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

# assign current user as owner
$appId = $aadApplication.AppId
Add-AzureADApplicationOwner -ObjectId $aadApplication.ObjectId -RefObjectId $User.ObjectId
```

Microsoft Azure

Search resources, services, and docs (G+)

Home > Power BI Dev Camp | App registrations >

User-Owns-Data Sample App

Search (Ctrl+/) << Delete Endpoints

Overview

Quickstart

Integration assistant (preview)

Display name : User-Owns-Data Sample App

Application (client) ID : 4216f5ee-fc2a-4950-b5d9-82a3fb1aaa88

Directory (tenant) ID : e60e3ff3-1c14-4f63-aca4-d30475c0a3d1

Object ID : d3986613-0625-431a-bfb3-c7ba9892c897

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value
App Secret	7/25/2021	952eBrpO4-AZ03huF-14NP_dDMJzmyGuu

21

Azure AD Configuration in appsettings.json

File Edit Selection View Go Run Terminal Help

appsettings.json - UserOwnsData - Visual Studio Code

EXPLORER

USEROWNSDATA

bin

Controllers

Models

obj

Properties

Views

wwwroot

appsettings.json

appsettings.Development.json

CreateAzureADApplication.ps1

CreateNetCoreProject.ps1

Program.cs

Startup.cs

UserOwnsData.csproj

UserOwnsDataSampleApp.txt

1 {

2 "AzureAd": {

3 "Instance": "https://login.microsoftonline.com/",

4 "Domain": "powerbidevcamp.net",

5 "TenantId": "2f23c5ea-5a75-41f6-922e-d3392313e61d",

6 "ClientId": "6d8ab9b5-50fb-4468-a7fd-000000000000",

7 "ClientSecret": "VjEXNjI4NGQ0ZDlmNS00YTZkLTg2MmYyZg2YTRhN2Q0ODAs=",

8 "CallbackPath": "/signin-oidc",

9 "SignedOutCallbackPath": "/signout-callback-oidc"

10 },

11 "PowerBI": {

12 "ServiceRootUrl": "https://api.powerbi.com/"

13 },

14 "Logging": {

15 "LogLevel": {

16 "Default": "Information",

17 "Microsoft": "Warning",

18 "Microsoft.Hosting.Lifetime": "Information"

19 } }

20 },

21 "AllowedHosts": "*"

22 }

23 }

24 }

25 }

26 }

27 }

28 }

29 }

30 }

UserOwnsDataSampleApp.txt - Notepad

File Edit Format View Help

{

"AzureAd": {

"Instance": "https://login.microsoftonline.com/",

"Domain": "powerbidevcamp.net",

"TenantId": "2f23c5ea-5a75-41f6-922e-d3392313e61d",

"ClientId": "6d8ab9b5-50fb-4468-a7fd-000000000000",

"ClientSecret": "VjEXNjI4NGQ0ZDlmNS00YTZkLTg2MmYyZg2YTRhN2Q0ODAs=",

"CallbackPath": "/signin-oidc",

"SignedOutCallbackPath": "/signout-callback-oidc"

},

"PowerBI": {

"ServiceRootUrl": "https://api.powerbi.com/"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "*"

}

22

Enabling Authentication with Microsoft.Identity.Web

```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services) {

    services.AddMicrosoftIdentityWebAppAuthentication(Configuration);

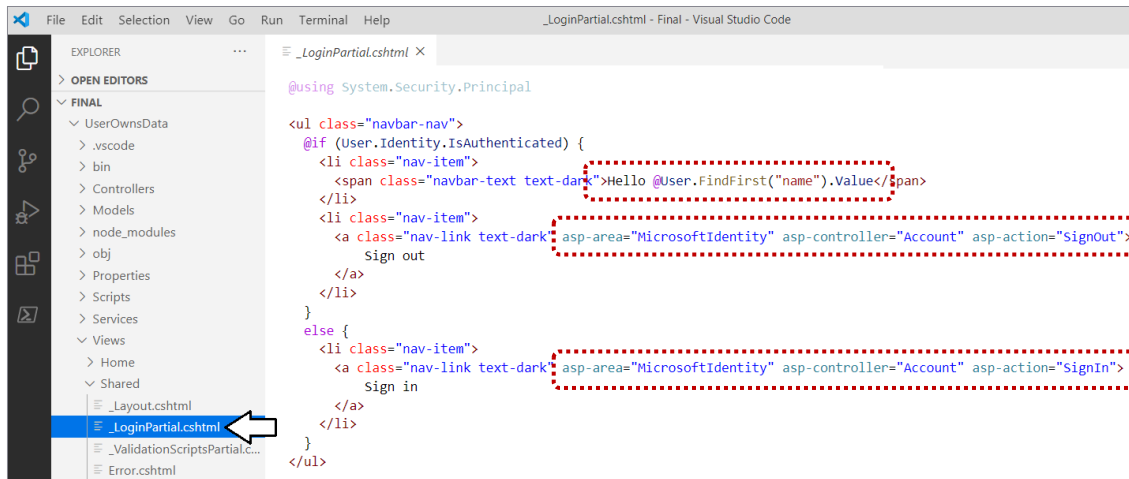
    var mvcBuilder = services.AddControllersWithViews(options => {
        var policy = new AuthorizationPolicyBuilder()
            .RequireAuthenticatedUser()
            .Build();
        options.Filters.Add(new AuthorizeFilter(policy));
    });

    mvcBuilder.AddMicrosoftIdentityUI();

    services.AddRazorPages();
}
```

23

Adding Sign in / Sign Out Links



24

ASP.NET Supports Route Authorization

- Routes with [AllowAnonymous] accessible to anonymous user
- Routes secured with [Authorize] only accessible to authenticated users
- Navigating to secured route automatically prompts for sign in

```
[Authorize]
public class HomeController : Controller {

    public HomeController() {}

    [AllowAnonymous]
    public IActionResult Index() {
        return View();
    }

    public IActionResult Embed() {
        return View();
    }
}
```

Index.cshtml

Embed.cshtml

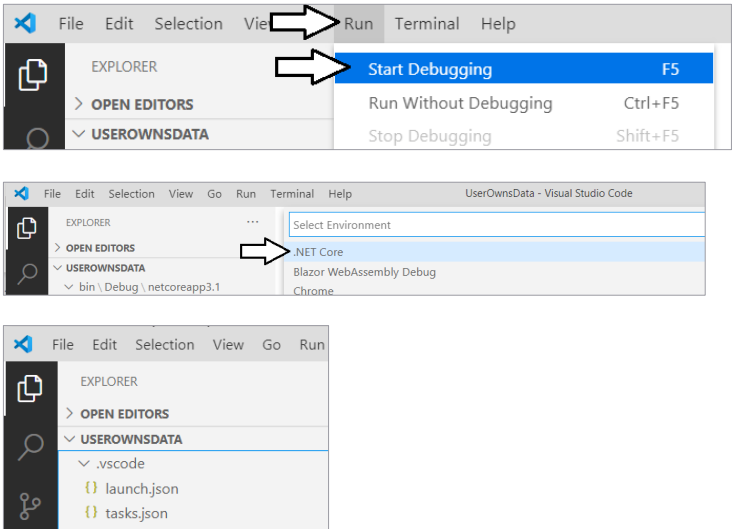
```
Index.cshtml X
@using System.Security.Principal

@if (User.Identity.IsAuthenticated) {
    <div class="jumbotron">
        <h2>Welcome @User.FindFirst("name").Value</h2>
        <p>You have now logged into this application.</p>
    </div>
}
else {
    <div class="jumbotron">
        <h2>Welcome to the User-Owns-Data Tutorial</h2>
        <p>Click the <strong>sign inc/strong> link in the upper
    </div>
}
```

```
Embed.cshtml X

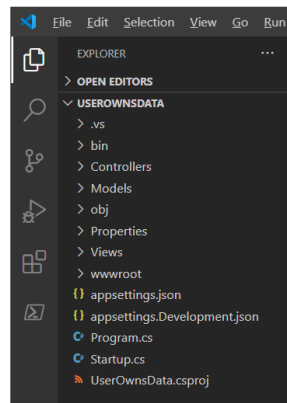
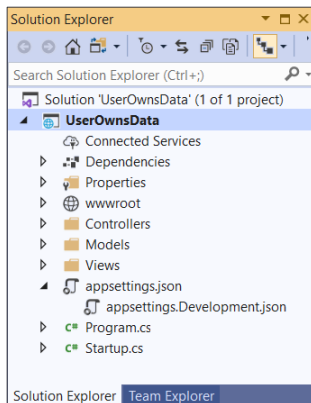
<h2>TODO: Embed Report Here</h2>
```

Starting a .NET Core Debugging Session



Visual Studio 2019 versus Visual Studio Code

- Should you use Visual Studio 2019 versus Visual Studio Code?
 - Yes, either one can be used to develop for .NET Core 3.1 and .NET 5



27

Agenda

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing `Microsoft.Identity.Web`
- Calling the Power BI Service API
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

28

What Is the Power BI Service API?

- **What is the Power BI Service API?**
 - API built on OAuth2, OpenID Connect, REST and ODATA
 - API secured by Azure Active Directory (AAD)
 - API to program with workspaces, datasets, reports & dashboards
 - API also often called "Power BI REST API"
- **What can you do with the Power BI Service API?**
 - Publish PBIX project files
 - Update connection details and datasource credentials
 - Create workspaces and clone content across workspaces
 - Embed Power BI reports and dashboards tiles in web pages
 - Create streaming datasets in order to build real-time dashboards

29

Calling the Power BI Service API

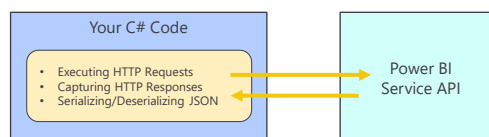
Direct REST calls without using the Power BI .NET SDK

```
static string ExecuteGetRequest(string restUrl) {
    HttpClient client = new HttpClient();
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
    request.Headers.Add("Authorization", "Bearer " + GetAccessToken());
    request.Headers.Add("Accept", "application/json;odata.metadata=minimal");
    HttpResponseMessage response = client.SendAsync(request).Result;
    if (response.StatusCode != HttpStatusCode.OK) {
        throw new ApplicationException("Error occured calling the Power BI Service API");
    }
    return response.Content.ReadAsStringAsync().Result;
}

static void Main() {
    // get report data from app workspace
    string restUrl = "https://api.powerbi.com/v1.0/myorg/groups/" + appWorkspaceId + "/reports/";
    var json = ExecuteGetRequest(restUrl);
    ReportCollection reports = JsonConvert.DeserializeObject<ReportCollection>(json);
    foreach (Report report in reports.value) {
        Console.WriteLine("Report Name: " + report.name);
        Console.WriteLine();
    }
}
```

```
public class Report {
    public string id { get; set; }
    public string name { get; set; }
    public string webUrl { get; set; }
    public string embedUrl { get; set; }
    public bool isOwnedByMe { get; set; }
    public string datasetId { get; set; }
}

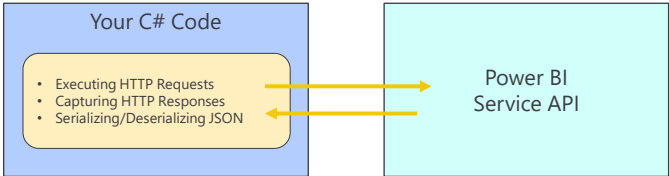
public class ReportCollection {
    public List<Report> value { get; set; }
}
```



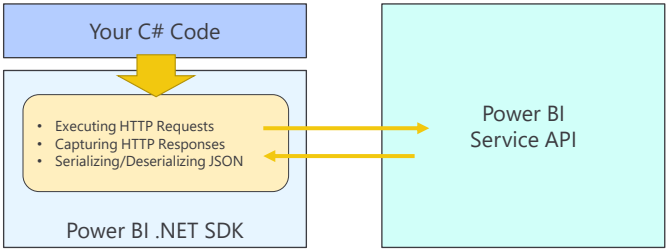
30

Power BI .NET SDK

• Developing without the Power BI .NET SDK



• Developing with the Power BI .NET SDK



31

Migrating to v3 of the Power BI .NET SDK

- You must move to SDK v3 to take advantage of latest API features
 - Automated server-side generation of PDF file from Power BI report
 - Generation of multi-resource embed tokens



- Beware of breaking changes when moving from v2.x to 3.x
 - Namespace `Microsoft.PowerBI.Api.v2` renamed to `Microsoft.PowerBI.Api`
 - Namespace `Microsoft.PowerBI.Api.Models.v2` renamed to `Microsoft.PowerBI.Api.Models`
 - Parameters for Power BI resource IDs now based on GUIDs instead of strings

32

Exercise 3

Call the Power BI Service API

1. Add a new class named `PowerBiServiceApi`
2. Add support to acquire access tokens using `Microsoft.Identity.Web`
3. Call Power BI Service API to get embedding data for a report
4. Pass the embedding data for a report to the browser

33

Adding Support for Token Acquisition

- Modify `ConfigureService` in `Startup.cs`

```
public void ConfigureServices (IServiceCollection services) {  
  
    services  
        .AddMicrosoftIdentityWebAppAuthentication(Configuration)  
        .EnableTokenAcquisitionToCallDownstreamApi(PowerBiServiceApi.RequiredScopes)  
        .AddInMemoryTokenCaches();  
  
    services.AddScoped (typeof (PowerBiServiceApi));  
}
```

- Add a new C# Class named `PowerBiServiceApi`

```
public class PowerBiServiceApi {  
  
    private ITokenAcquisition tokenAcquisition { get; }  
    private string urlPowerBiServiceApiRoot { get; }  
  
    public PowerBiServiceApi(IConfiguration configuration, ITokenAcquisition tokenAcquisition)  
    {  
        this.urlPowerBiServiceApiRoot = configuration["PowerBi:ServiceRootUrl"];  
        this.tokenAcquisition = tokenAcquisition;  
    }  
}
```



Dependency
injection

34

Adding Support for Acquiring Access Tokens

```
public class PowerBIServiceApi {  
  
    private ITokenAcquisition tokenAcquisition { get; }  
    private string urlPowerBIServiceApiRoot { get; }  
  
    public PowerBIServiceApi(IConfiguration configuration, ITokenAcquisition tokenAcquisition) {  
        this.urlPowerBIServiceApiRoot = configuration["PowerBi:ServiceRootUrl"];  
        this.tokenAcquisition = tokenAcquisition;  
    }  
  
    public static readonly string[] RequiredScopes = new string[] {  
        "https://analysis.windows.net/powerbi/api/Group.Read.All",  
        "https://analysis.windows.net/powerbi/api/Report.ReadWrite.All",  
        "https://analysis.windows.net/powerbi/api/Dataset.ReadWrite.All",  
        "https://analysis.windows.net/powerbi/api/Content.Create",  
        "https://analysis.windows.net/powerbi/api/Workspace.ReadWrite.All"  
    };  
  
    public string GetAccessToken() {  
        return this.tokenAcquisition.GetAccessTokenForUserAsync(RequiredScopes).Result;  
    }  
  
    public PowerBIClient GetPowerBIClient() {  
        var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");  
        return new PowerBIClient(new Uri(urlPowerBIServiceApiRoot), tokenCredentials);  
    }  
}
```

35

Call GetReportInGroupAsync to Get Embedding Data

```
public class EmbeddedReportViewModel {  
    public string Id;  
    public string Name;  
    public string EmbedUrl;  
    public string Token;  
}
```

```
public async Task<EmbeddedReportViewModel> GetReport(Guid WorkspaceId, Guid ReportId) {  
  
    PowerBIClient pbiClient = GetPowerBIClient();  
  
    // call to Power BI Service API to get embedding data  
    var report = await pbiClient.Reports.GetReportInGroupAsync(WorkspaceId, ReportId);  
  
    // return report embedding data to caller  
    return new EmbeddedReportViewModel {  
        Id = report.Id.ToString(),  
        EmbedUrl = report.EmbedUrl,  
        Name = report.Name,  
        Token = GetAccessToken()  
    };  
}
```

36

Accessing PowerBiServiceApi from a Controller

```
[Authorize]
public class HomeController : Controller {

    private PowerBiServiceApi powerBiServiceApi;

    public HomeController(PowerBiServiceApi powerBiServiceApi) {
        this.powerBiServiceApi = powerBiServiceApi;
    }

    public async Task<IActionResult> Embed() {
        Guid workspaceId = new Guid("912f2b34-7daa-4589-83df-35c75944d864");
        Guid reportId = new Guid("cd496c1c-8df0-48e7-8b92-e2932298743e");

        // call to PowerBiServiceApi
        var viewModel = await powerBiServiceApi.GetReport(workspaceId, reportId);
        return View(viewModel);
    }
}
```

Dependency
injection



37

Displaying Data from the View Model

The screenshot shows the Visual Studio Code interface with the Explorer pane on the left and the Editor pane on the right. The Explorer pane shows the project structure with the following folders and files:

- USEROWNSDATA
 - .vscode
 - bin
 - Controllers
 - Models
 - obj
 - Properties
 - Services
 - Views
 - Home
 - Embed.cshtml (selected)
 - Index.cshtml
 - Shared
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - wwwroot
- appsettings.json

The Editor pane shows the content of the Embed.cshtml file:

```

1  @model UserOwnsData.Services.EmbeddedReportViewModel;
2
3  <style>
4      table td {
5          min-width: 120px;
6          word-break: break-all;
7          overflow-wrap: break-word;
8          font-size: 0.8em;
9      }
10 </style>
11
12 <h3>Report View Model Data</h3>
13
14 <table class="table table-bordered table-striped table-sm" >
15 <tr><td>Report Name</td><td>@Model.Name</td></tr>
16 <tr><td>Report ID</td><td>@Model.Id</td></tr>
17 <tr><td>Embed Url</td><td>@Model.EmbedUrl</td></tr>
18 <tr><td>Token</td><td>@Model.Token</td></tr>
19 </table>
  
```

38

Exercise 3 Finale

[illegible]

39

Agenda

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling the Power BI Service API
- Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

40

Exercise 4

Embedding a Report using powerbi.js

1. Add a script link to powerbi.js
2. Create view model to pass embedding data to browser
3. Write JavaScript code to embed a report

41

Creating a Simple View To Embed a Report

1. First script link loads Power BI JavaScript API
2. Second script link adds view model to web page with report embedding data
3. Third script link loads a custom Javascript file named embed.js that you will write

```
Embed.cshtml X

@model UserOwnsData.Services.EmbeddedReportViewModel;

<div id="embed-container" style="height:800px;"></div>

@section Scripts {
    <script src="https://cdn.jsdelivr.net/npm/powerbi-client@2.13.3/dist/powerbi.min.js"></script>
    <script>
        var viewModel = {
            reportId: "@Model.Id",
            embedUrl: "@Model.EmbedUrl",
            token: "@Model.Token"
        };
    </script>
    <script src="~/js/embed.js"></script>
}
```

42

Retrieving Data from the Report View Model

wwwroot

> css

> js

js

embed.js

> lib

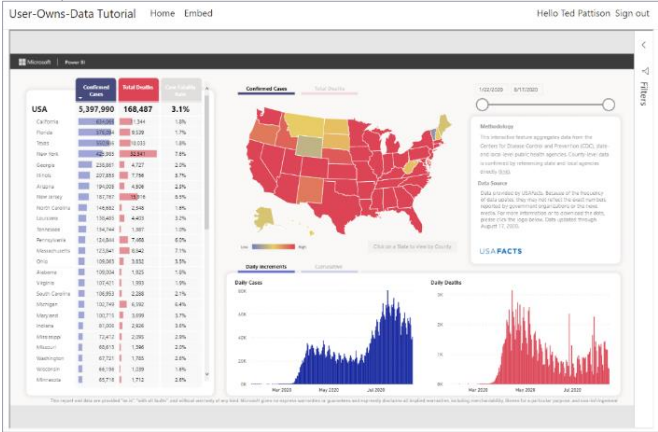
js embed.js

```
$(function () {  
  
    // 1 - get DOM object for div that is report container  
    var reportContainer = document.getElementById("embed-container");  
  
    // 2 - get report embedding data from view model  
    var reportId = window.viewModel.reportId;  
    var embedUrl = window.viewModel.embedUrl;  
    var token = window.viewModel.token  
  
    // 3 - embed report using the Power BI JavaScript API.  
  
    // 4 - add logic to resize embed container on window resize event  
  
});
```

43

Embedding a Report using powerbi.js

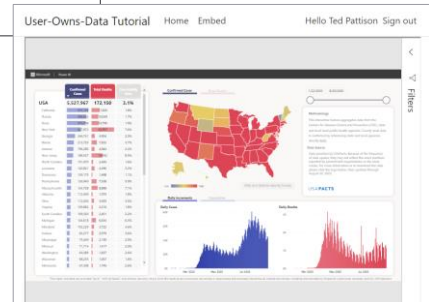
```
// 2 - get report embedding data from view model  
var reportId = window.viewModel.reportId;  
var embedUrl = window.viewModel.embedUrl;  
var token = window.viewModel.token  
  
// 3 - embed report using the Power BI JavaScript API.  
var models = window['powerbi-client'].models;  
  
var config = {  
    type: 'report',  
    id: reportId,  
    embedUrl: embedUrl,  
    accessToken: token,  
    permissions: models.Permissions.All,  
    tokenType: models.TokenType.Aad,  
    viewMode: models.ViewMode.View,  
    settings: {  
        panes: {  
            filters: { expanded: false, visible: true },  
            pageNavigation: { visible: false }  
        }  
    }  
};  
  
// Embed the report and display it within the div container.  
var report = powerbi.embed(reportContainer, config);
```



44

Dynamically Resizing the Embed Container

```
// 4 - add logic to resize embed container on window resize event
var heightBuffer = 12;
var newHeight = $(window).height() - ($("#header").height() + heightBuffer);
$("#embed-container").height(newHeight);
$(window).resize(function () {
    var newHeight = $(window).height() - ($("#header").height() + heightBuffer);
    $("#embed-container").height(newHeight);
});
```



45

Agenda

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing Microsoft.Identity.Web
- ✓ Calling the Power BI Service API
- ✓ Programming the Power BI JavaScript API
- Adding TypeScript Support to a .NET Core Project
 - Programming with Multi-Resource Embed Tokens

46

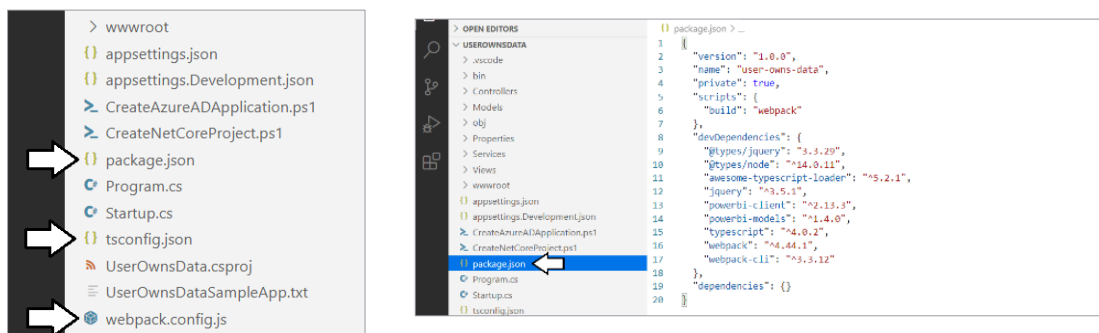
Exercise 5

Adding TypeScript/Webpack Support to a .NET Core Project

1. Copy `package.json` file to `UserOwnsData` project root folder
2. Execute `npm install` to install Node.js packages
3. Copy `tsconfig.json` and `webpack.config.js` to root folder
4. Add new TypeScript file named `embed.ts`
5. Execute `npm run build` to compile `embed.ts` to `embed.js`
6. Update `UserOwnsData.csproj` with `npm run build` command

47

Adding Node.js Support for TypeScript Compilation



48

Running npm install

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

PS C:\DevCamp\UserOwnsData> npm install

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the file structure with 'node_modules' highlighted. The Terminal window on the right shows the output of the 'npm install' command, including warnings about deprecated packages and the installation of 423 packages.

49

Understanding Webpack and Dynamic Module Loading

```
TS embed.ts ×  
  
import * as $ from 'jquery';  
  
import * as powerbi from "powerbi-client";  
import * as models from "powerbi-models";  
  
// ensure Power BI JavaScript API has loaded  
require('powerbi-models');  
require('powerbi-client');
```



50

Running npm build

Scripts

TS embed.ts

> Services

> Views

> wwwroot

> css

> js

JS embed.js

JS embed.js.map

> lib

★ favicon.ico

() appsettings.json

() appsettings.Development.json

> CreateAzureADApplication.ps1

> CreateNetCoreProject.ps1

() package.json

() package-lock.json

Program.cs

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

PS C:\DevCamp\UserOwnsData> npm run build

> user-owns-data@1.0.0 build C:\DevCamp\UserOwnsData

> webpack

i | atlj: Using typescript@4.0.2 from typescript

i | atlj: Using tsconfig.json from C:\DevCamp\UserOwnsData\tsconfig.json

i | atlj: Checking started in a separate process...

i | atlj: Time: 655ms

Hash: f000950cc21af308caff

Version: webpack 4.44.1

Time: 2854ms

Built at: 08/25/2020 1:37:14 PM

JS embed.js

wwwroot > js > JS embed.js > ...

1 < /*****/ (function(modules) { // webpackBootstrap

2 < /*****/ // The module cache

3 < /*****/ var installedModules = {};

4 < /*****/

5 < /*****/ // The require function

6 < /*****/ function __webpack_require__(moduleId) {

51

Updating UserOwnsData.csproj

EXPLORER

> OPEN EDITORS

> FINAL

> Properties

> Scripts

> Services

> Views

> wwwroot

() appsettings.json

() appsettings.Development.json

> CreateAzureADApplication.ps1

> CreateNetCoreProject.ps1

() package.json

() package-lock.json

Program.cs

Startup.cs

() tsconfig.json

★ UserOwnsData.csproj

UserOwnsDataSampleApp.txt

webpack.config.js

UserOwnsData.csproj

<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>

<TargetFramework>netcoreapp3.1</TargetFramework>

<UserSecretsId>aspnet-UserOwnsData-4635C0F8-934C-4E4D-9024-F5D07A239315</UserSecretsId>

<WebProject_DirectoryAccessLevelKey>0</WebProject_DirectoryAccessLevelKey>

</PropertyGroup>

<ItemGroup>

<PackageReference Include="Microsoft.Identity.Web" Version="0.2.3-preview" />

<PackageReference Include="Microsoft.Identity.Web.UI" Version="0.2.3-preview" />

<PackageReference Include="Microsoft.PowerBi.Api" Version="3.14.0" />

</ItemGroup>

<Target Name="PostBuild" AfterTargets="PostBuildEvent">

<Exec Command="npm run build" />

</Target>

</Project>

52

Running dotnet build

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\DevCamp\UserOwnsData> dotnet build



PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\DevCamp\UserOwnsData> dotnet build
Microsoft (R) Build engine version 16.7.0-preview-20310-07+e1c9f8dc for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
You are using a preview version of .NET. See: <https://aka.ms/dotnet-core-preview>
UserOwnsData -> C:\DevCamp\UserOwnsData\bin\Debug\netcoreapp3.1\UserOwnsData.dll
UserOwnsData -> C:\DevCamp\UserOwnsData\bin\Debug\netcoreapp3.1\UserOwnsData.Views.dll

> user-owns-data@1.0.0 build C:\DevCamp\UserOwnsData

> webpack

1 n/dotIn40: using typescript@4.0.2 from typescript
1 n/dotIn40: Using tsconfig.json from C:\DevCamp\UserOwnsData\tsconfig.json
1 n/dotIn40: Checking started in a separate process...
1 n/dotIn40: Time: 688ms
Hash: f80906cc21af308caff
Version: webpack 4.44.1
Time: 2902ms
Built at: 08/25/2020 1:45:00 PM
Asset Size Chunks Chunk Names
embed.js 782 KiB main [emitted] main
embed.js.map 975 KiB main [emitted] [dev] main
Entrypoint main = embed.js embed.js.map
[./scripts/embed.js] 1.45 KiB [main] [built]
+ 3 hidden modules

Build succeeded.

0 Warning(s)

0 Error(s)

Time Elapsed 00:00:00.55

PS C:\DevCamp\UserOwnsData>

53

Programming the PBI JS API using TypeScript

```
// get DOM object div for report container
var reportContainer: HTMLElement = document.getElementById("embed-container");

var viewModel: ViewModel = window["viewModel"];

var config: powerbi.IEmbedConfiguration = {
  type: "report",
  id: viewModel.reportId,
  embedUrl: viewModel.embedUrl,
  accessToken: viewModel.token,
  permissions: models.Permissions.All,
  tokenType: models.TokenType.Aad,
  viewMode: models.ViewMode.View,
  settings: {
    panes: {
      filters: { expanded: false, visible: true },
      pageNavigation: { visible: true }
    },
    persistentFiltersEnabled: true
  }
};

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

54

Exercise 6

Creating a View Model for App Workspaces

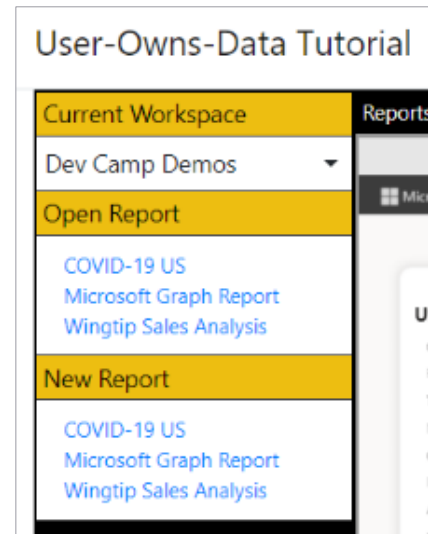
```
public async Task<string> GetEmbeddedViewModel(string appWorkspaceId = "") {  
    var accessToken = this.tokenAcquisition.GetAccessTokenForUserAsync(RequiredScopes).Result;  
    var tokenCredentials = new TokenCredentials(accessToken, "Bearer");  
    PowerBiClient pbClient = new PowerBiClient(new Uri(urlPowerBIServiceApiRoot), tokenCredentials);  
  
    Object viewModel;  
    if (string.IsNullOrEmpty(appWorkspaceId)) {  
        viewModel = new {  
            currentWorkspace = "My workspace",  
            workspaces = (await pbClient.Groups.GetGroupsAsync()).Value,  
            datasets = (await pbClient.Datasets.GetDatasetsAsync()).Value,  
            reports = (await pbClient.Reports.GetReportsAsync()).Value,  
            token = accessToken  
        };  
    }  
    else {  
        Guid workspaceId = new Guid(appWorkspaceId);  
        var workspaces = (await pbClient.Groups.GetGroupsAsync()).Value;  
        var currentWorkspace = workspaces.First(workspace => workspace.Id == workspaceId);  
        viewModel = new {  
            workspaces = workspaces,  
            currentWorkspace = currentWorkspace.Name,  
            currentWorkspaceReadOnly = currentWorkspace.IsReadOnly,  
            datasets = (await pbClient.Datasets.GetDatasetsInGroupAsync(workspaceId)).Value,  
            reports = (await pbClient.Reports.GetReportsInGroupAsync(workspaceId)).Value,  
            token = accessToken  
        };  
    }  
    return JsonConvert.SerializeObject(viewModel);  
}
```

```
{
  "workspaces": [
    {
      "id": "6679bdc47-5b5f-4be0-ac6d-7a7ab1ba16f8",
      "name": "All Company",
      "isReadOnly": true
    },
    {
      "id": "912f72b34-7daa-4589-83df-35c75944d864",
      "name": "Dev Camp Demos",
      "isReadOnly": false
    }
  ],
  "currentWorkspace": "Dev Camp Demos",
  "currentWorkspaceIsReadOnly": false,
  "datasets": [
    {
      "id": "94e863ea-9117-445c-ac9e-db9373bdb8ea",
      "name": "COVID-19 US"
    },
    {
      "id": "89795189-of43-4057-9af7-ab4838082a3b",
      "name": "Microsoft Graph Report"
    },
    {
      "id": "bfa46105-5d6e-4270-88d8-e6b6be5bf5ad",
      "name": "Wingtip Sales Analysis"
    }
  ],
  "reports": [
    {
      "id": "cd4a610c-8df0-48e7-8b92-e2932298743e",
      "name": "COVID-19 US",
      "webUrl": "https://powerbi.com/reports/covid19-us"
    },
    {
      "id": "23d6559a-f3c-4c51-9175-2904ae76bbae",
      "name": "Microsoft Graph Report"
    },
    {
      "id": "fa8b93ae-c6a6-4e0c-8e4b-be7057cf583c",
      "name": "Wingtip Sales Analysis"
    }
  ],
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngidCI6ImppYk5iOiZ0Z2JteFBZc2k4SQ0Z"
}
```

55

Creating the User Experience

```
{
  "workspaces": [
    {
      "id": "6679bd47-5b5f-4be0-ac6d-7a7ab1ba16f8",
      "name": "All Company",
      "isReadOnly": true
    },
    {
      "id": "912f2b34-7daa-4589-83df-35c75944d864",
      "name": "Dev Camp Demos",
      "isReadOnly": false
    }
  ],
  "currentWorkspace": "Dev Camp Demos",
  "currentWorkspaceIsReadOnly": false,
  "datasets": [
    {
      "id": "94e863ea-9117-445c-ac9e-db9373bdb8ea",
      "name": "COVID-19 US"
    },
    {
      "id": "89795189-0f43-4057-9af7-48438082a3b",
      "name": "Microsoft Graph Report"
    },
    {
      "id": "bfa46105-5d6e-4270-88d8-e6b6e5bf57ad",
      "name": "Wingtip Sales Analysis"
    }
  ],
  "reports": [
    {
      "id": "cd496c1c-8df0-48e7-8bd9-e2932298743d",
      "name": "COVID-19 US",
      "webUrl": "https://reports.officeappsclient.com/Reports.aspx?ReportID=cd496c1c-8df0-48e7-8bd9-e2932298743d"
    },
    {
      "id": "23d56559-af3c-4c51-9175-2904ae67bbbae",
      "name": "Microsoft Graph Report"
    },
    {
      "id": "fa89b3ae-c6a6-4e0c-8e4b-be7057cf583c",
      "name": "Wingtip Sales Analysis"
    }
  ],
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngiaXNpdCI6ImppYk5ia0ZUZWJteFZkc45QOXM"
}
```



56

Camper Competition

• Competition Rules

- Build a .NET Core web application that extends User-Owns-Data tutorial
- You must use **Microsoft.Identity.Web** for authentication
- You must use TypeScript instead of JavaScript for client-side programming
- Create a 3-5 minute video where you demo and describe your application

57

Agenda

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing **Microsoft.Identity.Web**
- ✓ Calling the Power BI Service API
- ✓ Programming the Power BI JavaScript API
- ✓ Adding TypeScript Support to a .NET Core Project
- Programming with Multi-Resource Embed Tokens

58

Programming with Multi-Resource Embed Tokens

- Used in App-Owns-Data embedding

```
Guid workspaceId = new Guid(appworkspaceId);
var workspaces = (await pbiclient.Groups.GetGroupsAsync()).Value;
var currentworkspace = workspaces.First((workspace) => workspace.Id == workspaceId);
var datasets = (await pbiclient.Datasets.GetDatasetsInGroupAsync(workspaceId)).Value;
var reports = (await pbiclient.Reports.GetReportsInGroupAsync(workspaceId)).Value;

IList<GenerateTokenRequestV2Dataset> datasetRequests = new List<GenerateTokenRequestV2Dataset>();
foreach (var dataset in datasets) {
    datasetRequests.Add(new GenerateTokenRequestV2Dataset(dataset.Id));
};

IList<GenerateTokenRequestV2Report> reportRequests = new List<GenerateTokenRequestV2Report>();
foreach (var report in reports) {
    reportRequests.Add(new GenerateTokenRequestV2Report(report.Id, allowEdit: true));
};

IList<GenerateTokenRequestV2TargetWorkspace> workspaceRequests =
    new GenerateTokenRequestV2TargetWorkspace[] {
        new GenerateTokenRequestV2TargetWorkspace(workspaceId)
    };

GenerateTokenRequestV2 tokenRequest =
    new GenerateTokenRequestV2(
        datasets: datasetRequests,
        reports: reportRequests,
        targetWorkspaces: workspaceRequests);

// call to Power BI Service API and pass GenerateTokenRequest object to generate embed token
string embedToken = pbiclient.EmbedToken.GenerateToken(tokenRequest).Token;
```

59

Call to Action

1. Complete the Developer for Power BI with .NET Core tutorials
2. Fill out the tutorial survey form
3. Enter the camper competition
4. Come back next month for our Power BI PowerShell sessions

60

Summary

- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling the Power BI Service API
- ✓ Programming the Power BI JavaScript API
- ✓ Adding TypeScript Support to a .NET Core Project
- ✓ Programming with Multi-Resource Embed Tokens

61

Questions

62