Microsoft Power BI

1

Power BI Dev Camp – Session 1

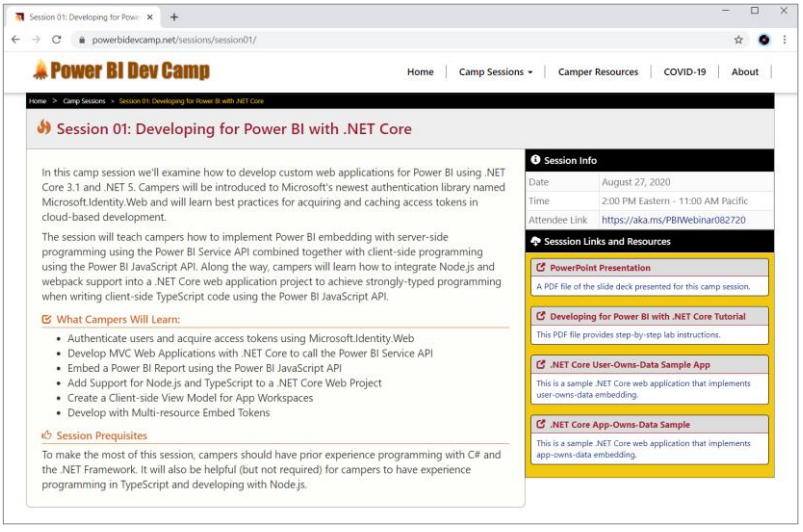
Developing for Power BI with .NET Core

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

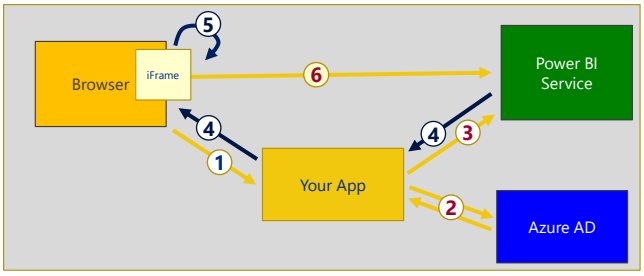
Agenda

- ➢ 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

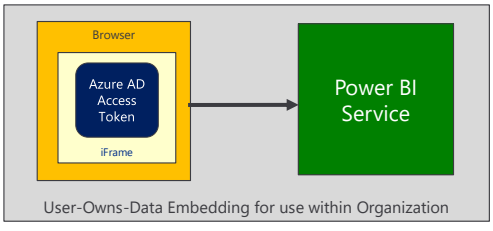


5

Choosing the Correct Embedding Model

User-Owns-Data Embedding

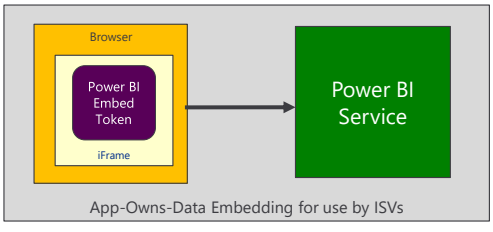
- All users require a Power BI license
- Useful in corporate environments
- App authenticates as current user
- Your code runs with user's permissions
- User's access token passed to browser



User-Owns-Data Embedding for use within Organization

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

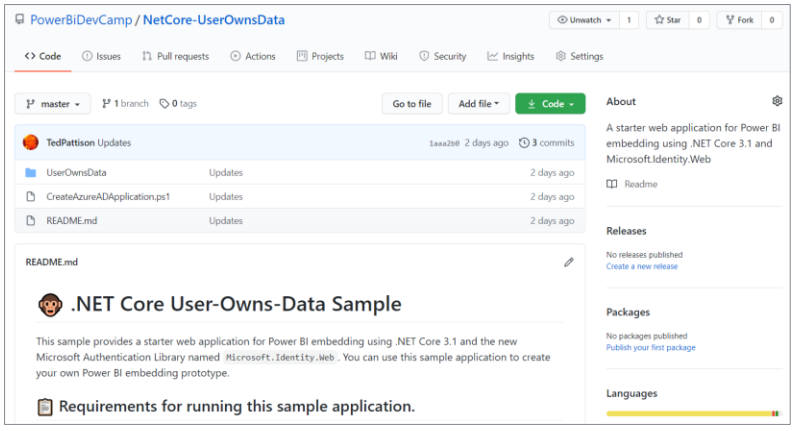


App-Owns-Data Embedding for use by ISVs

6

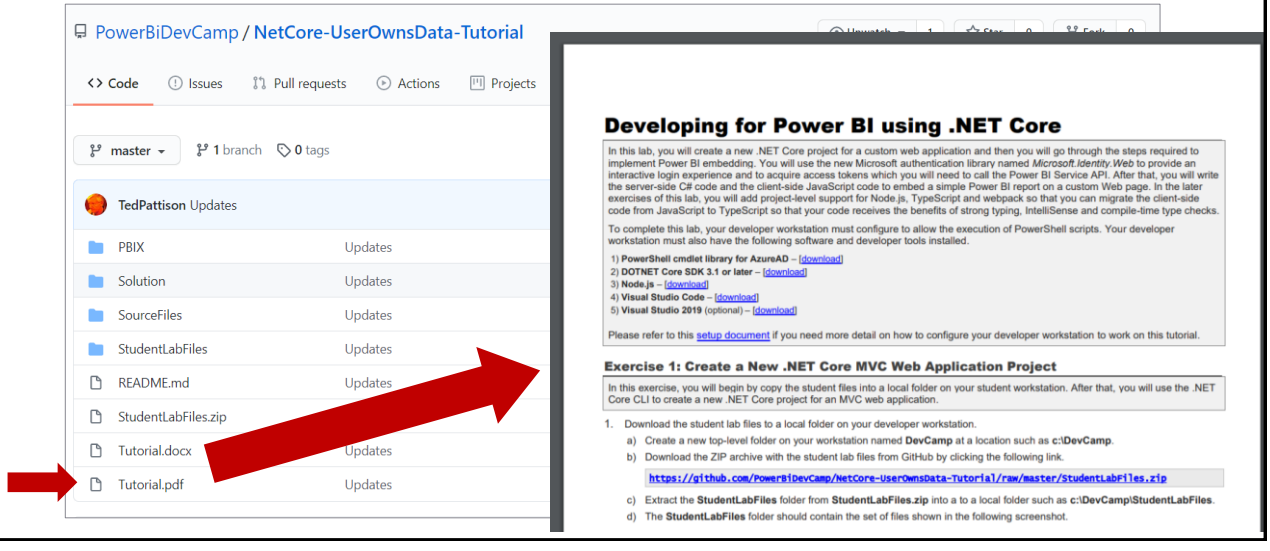
User-Owns-Data Sample Application

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



Tutorial: Developing for Power BI using .NET Core

- https://github.com/PowerBiDevCamp/NetCore-UserOwnsData-Tutorial

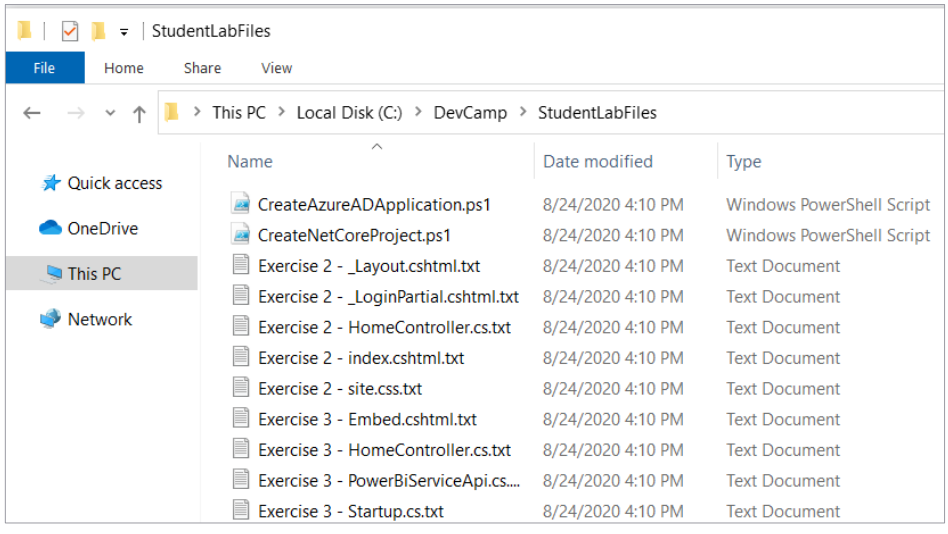


Tutorial Exercise Flow

- 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

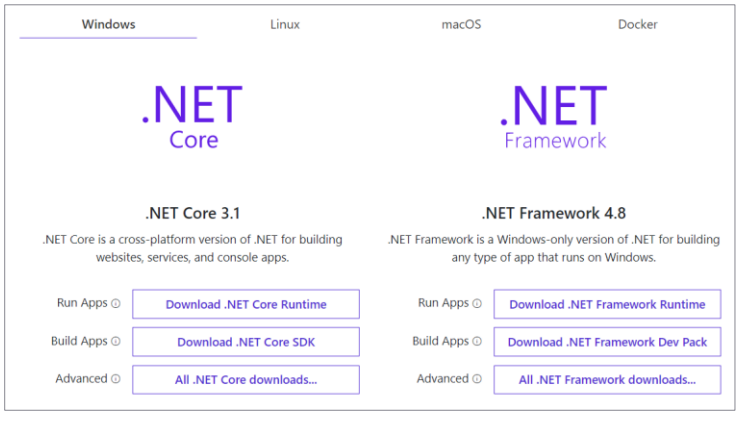
Agenda

- ✓ 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



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

```
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*

```
{
  "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 dependenacy injection
    public Startup(IConfiguration configuration) {

      // make Configuration service available to all methods in this class
      Configuration = configuration;

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

14

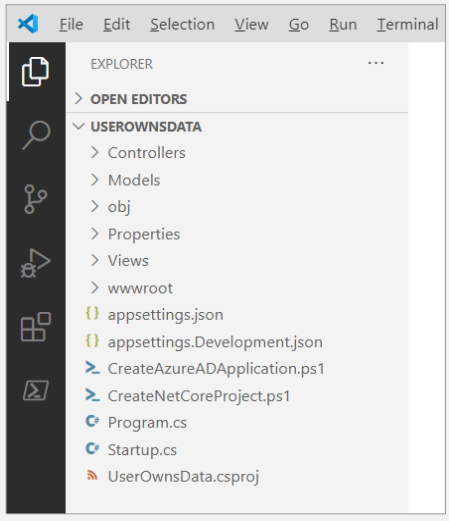
Exercise 1
Create a New .NET Core MVC Web Application Project

CreateNetCoreProject.ps1

```
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
```

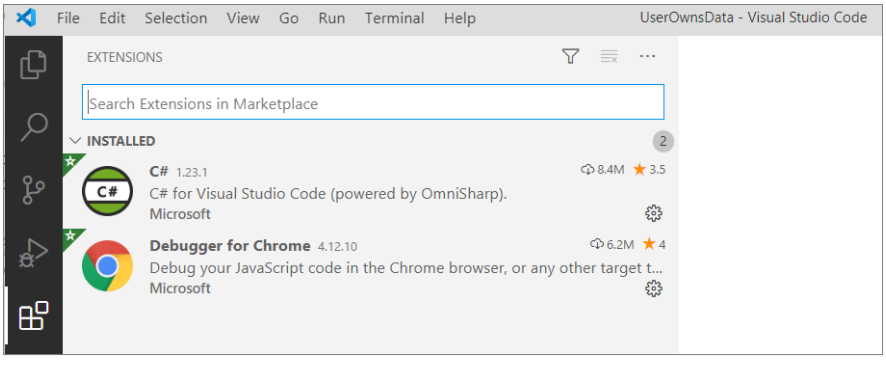
Windows PowerShell

```
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

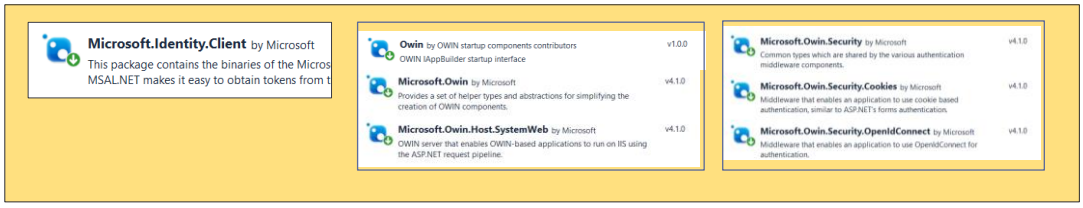
Introducing `Microsoft.Identity.Web`

- 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

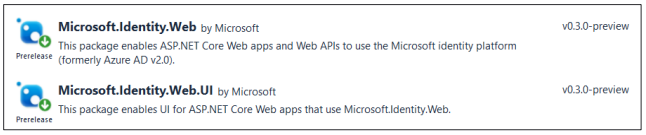
18

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

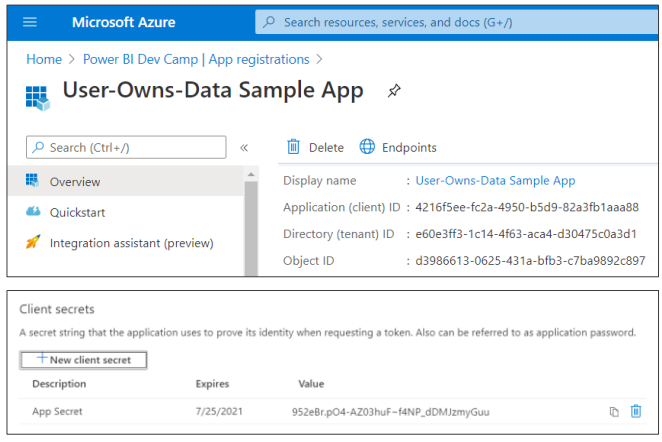
Implement User Login using `Microsoft.Identity.Web`

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

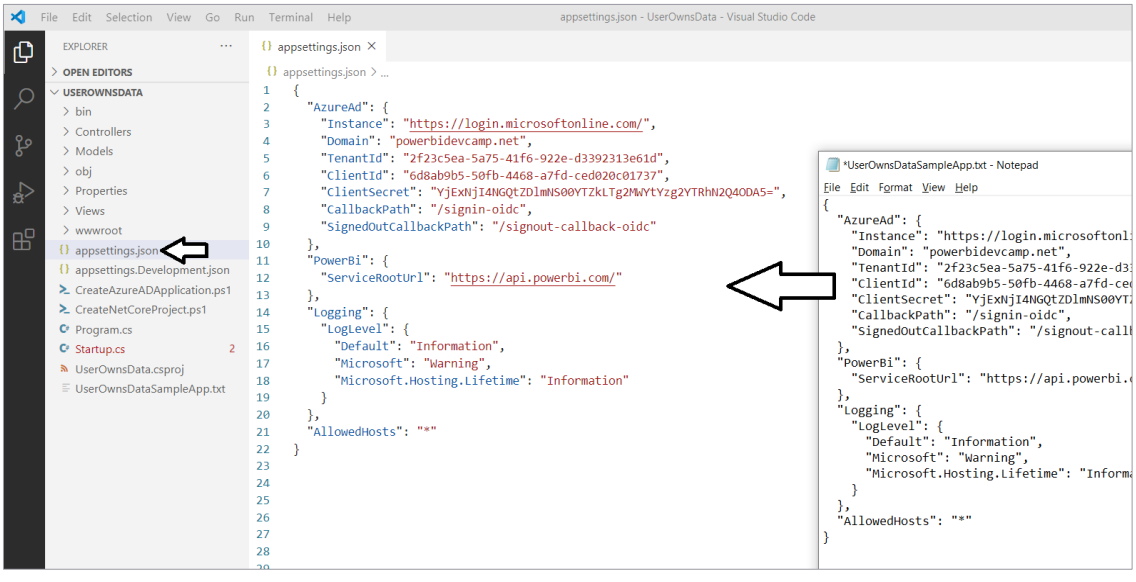
Creating Azure AD Application with PowerShell

```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
```



21

Azure AD Configuration in appsettings.json



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

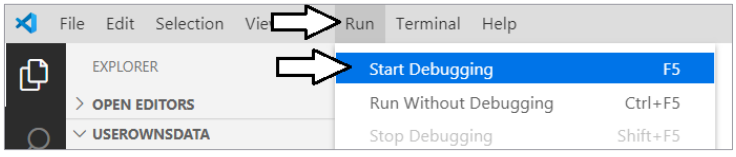
```
@using System.Security.Principal

@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 in</strong> link in the upper
  </div>
}
```

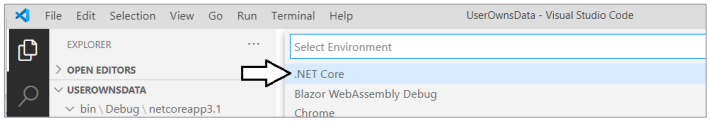
Embed.cshtml

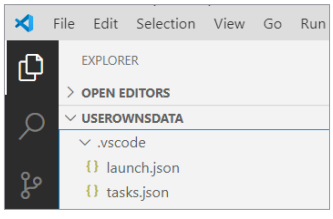
```
<h2>TODO: Embed Report Here</h2>
```

25

Starting a .NET Core Debugging Session



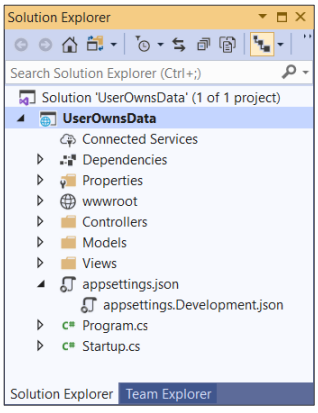


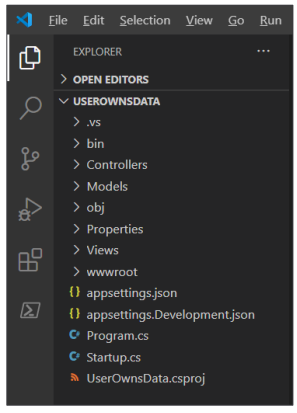


26

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

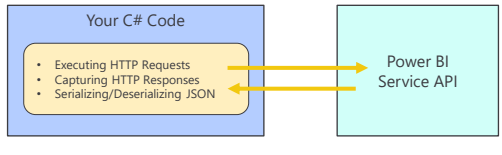
29

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 Servide 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();
  }
}
```

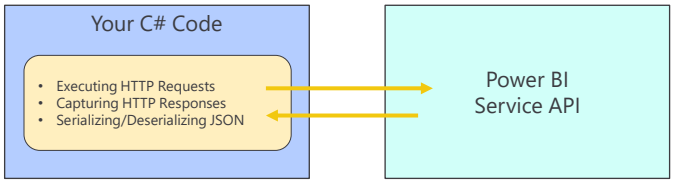
```
public class Report {
  public string id { get; set; }
  public string name { get; set; }
  public string webUrl { get; set; }
  public string embedUrl { get; set; }
  public bool isOwnedByMe { get; set; }
  public string datasetId { get; set; }
}

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



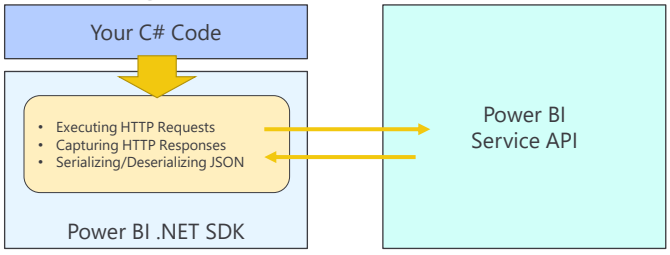
30

Power BI .NET SDK

- Developing without the Power BI .NET SDK



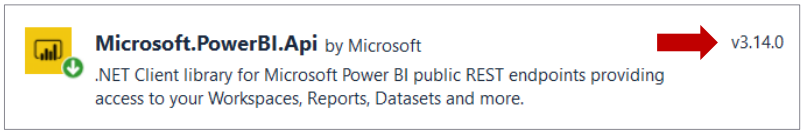
- Developing with the Power BI .NET SDK



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

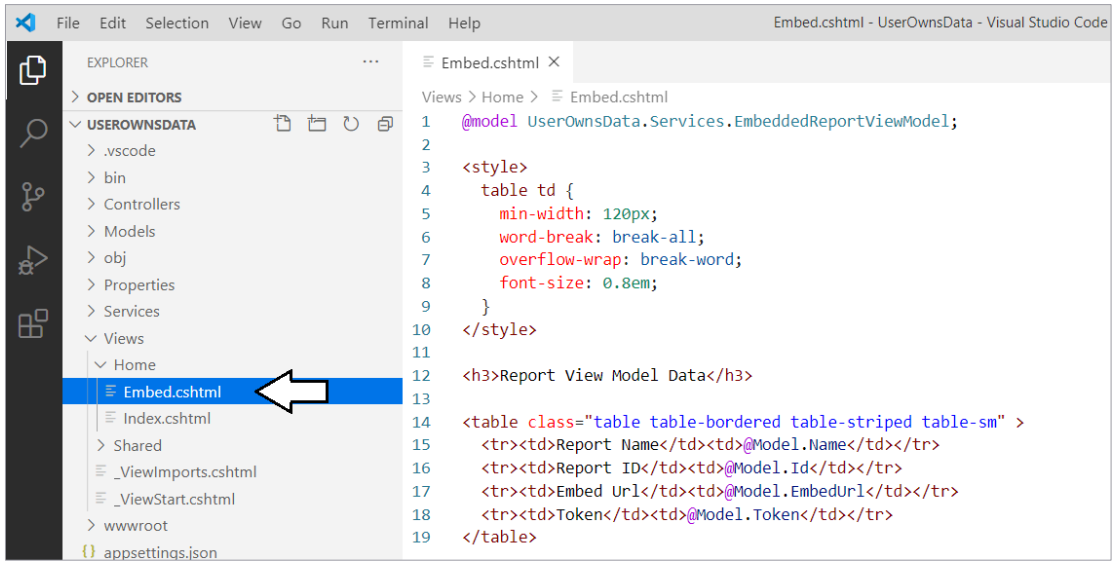
Accessing PowerBiServiceApi from a Controller

```
[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

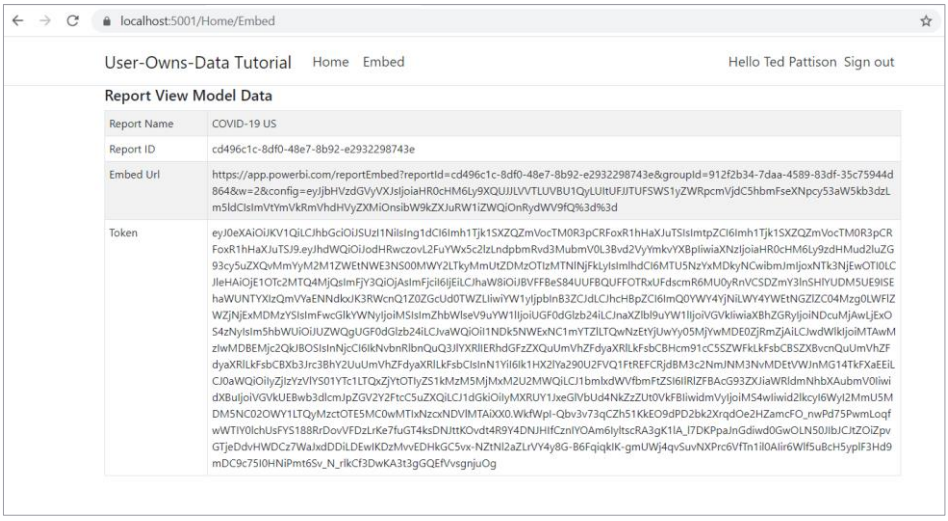
37

Displaying Data from the View Model



38

Exercise 3 Finale



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

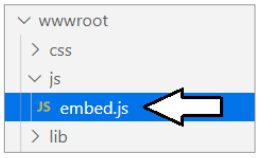
1. First script link loads Power BI JavaScript API
2. Second script link adds view model to web page with report embedding data
3. Third script link loads a custom Javascript file named embed.js that you will write

Embed.cshtml

```
@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

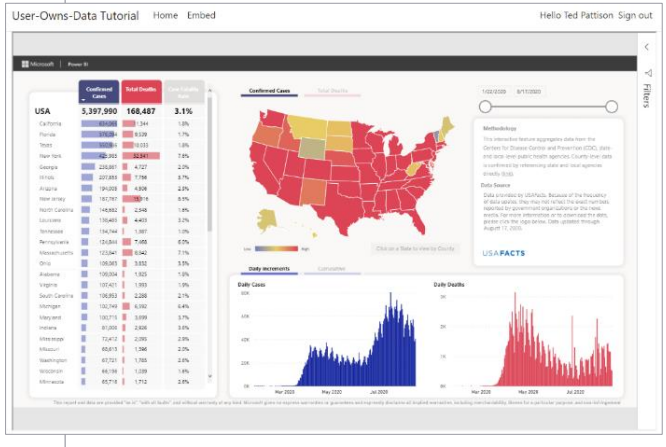


embed.js

```
$(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

});
```

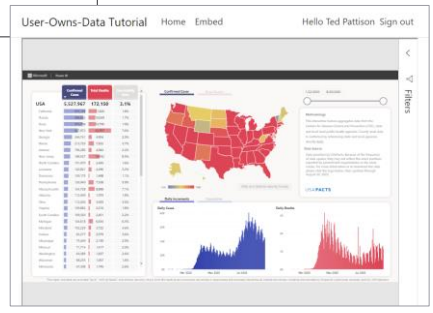

Embedding a Report using powerbi.js

```
// 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);
```



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

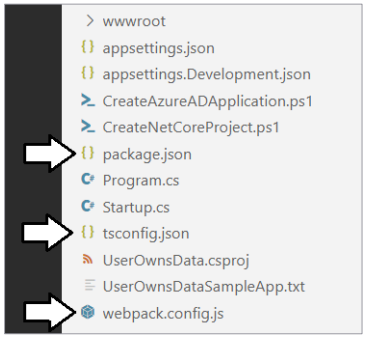
- ✓ Tutorial Introduction
- ✓ Developing with .NET Core
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling the Power BI Service API
- ✓ Programming the Power BI JavaScript API
- ➢ Adding TypeScript Support to a.NET Core Project
- ▪ Programming with Multi-Resource Embed Tokens

46

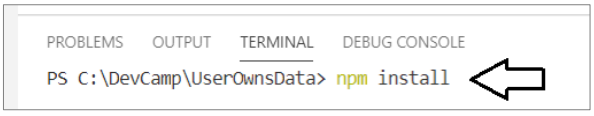
Exercise 5
Adding TypeScript/Webpack Support to a .NET Core Project

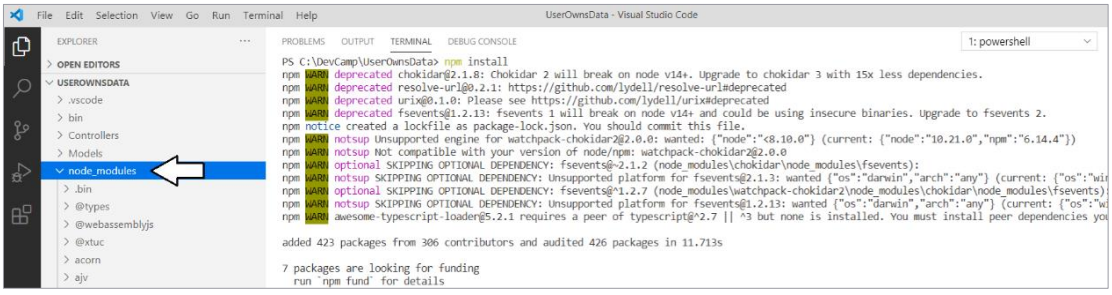
1. Copy `package.json` file to `UserOwnsData` project root folder
2. Execute `npm install` to install Node.js packages
3. Copy `tsconfig.json` and `webpack.config.js` to root folder
4. Add new TypeScript file named `embed.ts`
5. Execute `npm run build` to compile `embed.ts` to `embed.js`
6. Update `UserOwnsData.csproj`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

Adding Node.js Support for TypeScript Compilation



Running `npm install`

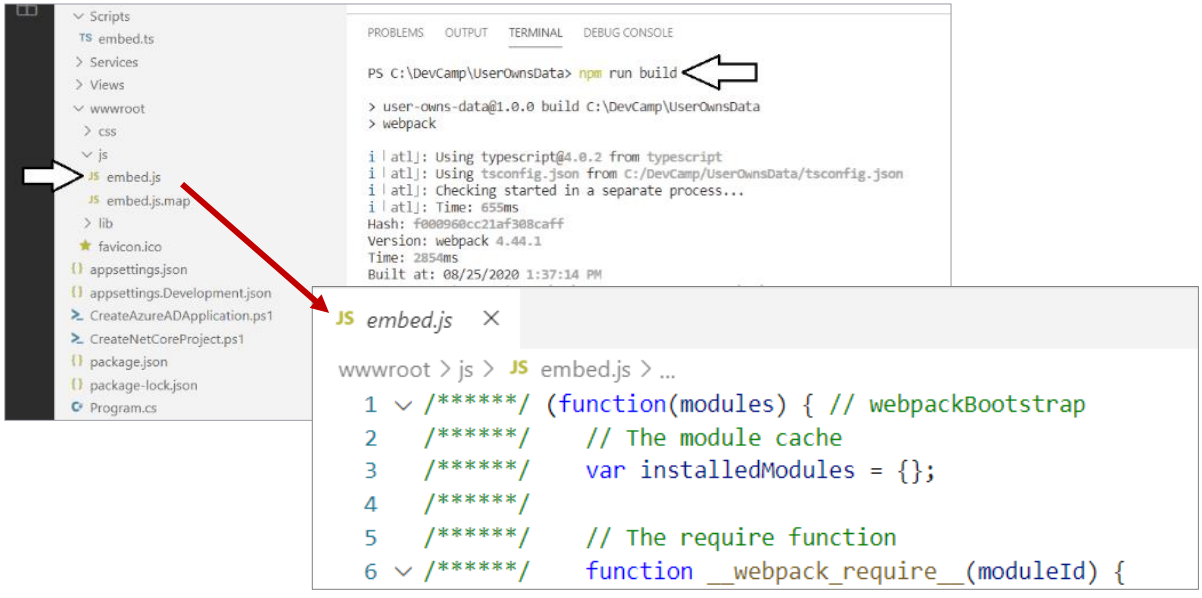




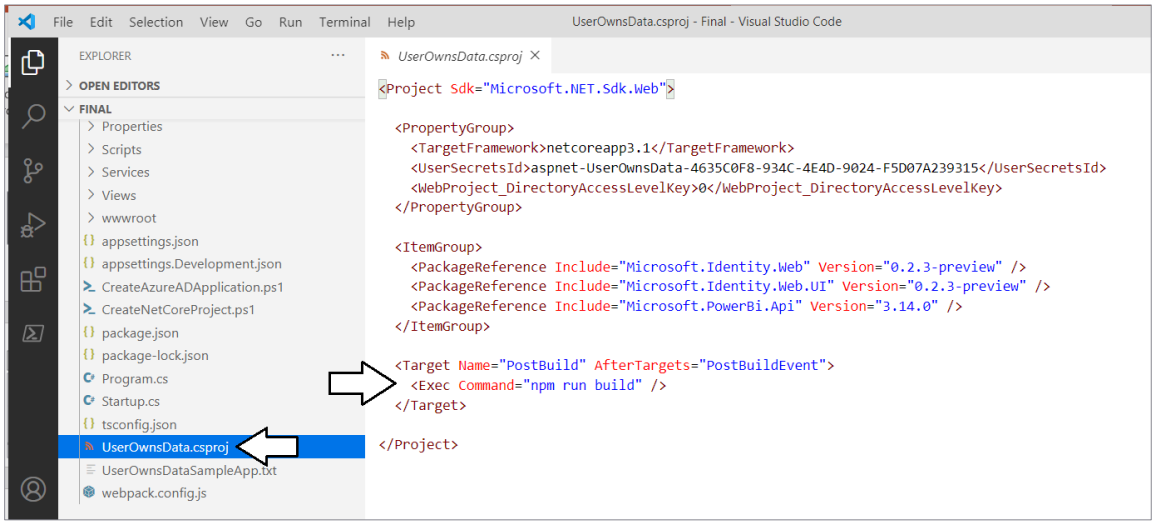
Understanding Webpack and Dynamic Module Loading

```
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');
```

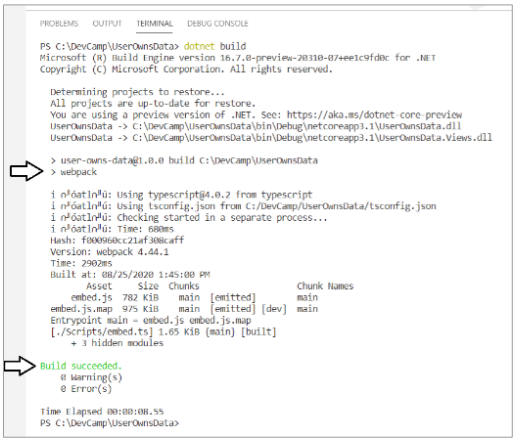
Running `npm build`



Updating UserOwnsData.csproj



Running dotnet build



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

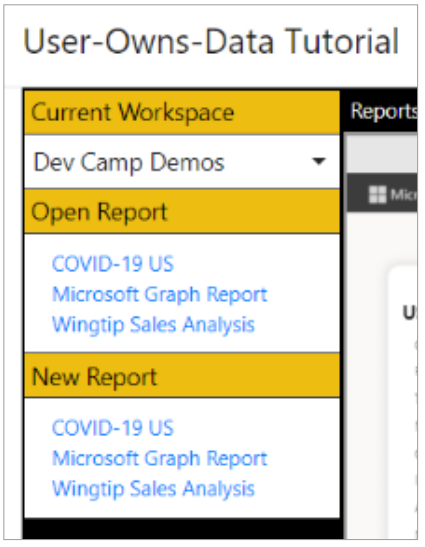
Exercise 6
Creating a View Model for App Workspaces

```
public async Task<string> GetEmbeddedViewModel(string appWorkspaceId = "") {

  var accessToken = this.tokenAcquisition.GetAccessTokenForUserAsync(RequiredScopes).Result;
  var tokenCredentials = new TokenCredentials(accessToken, "Bearer");
  PowerBIClient pbiClient = new PowerBIClient(new Uri(urlPowerBiServiceApiRoot), tokenCredentials);

  Object viewModel;
  if (string.IsNullOrEmpty(appWorkspaceId)) {
    viewModel = new {
      currentWorkspace = "My Workspace",
      workspaces = (await pbiClient.Groups.GetGroupsAsync()).Value,
      datasets = (await pbiClient.Datasets.GetDatasetsAsync()).Value,
      reports = (await pbiClient.Reports.GetReportsAsync()).Value,
      token = accessToken
    };
  }
  else {
    Guid workspaceId = new Guid(appWorkspaceId);
    var workspaces = (await pbiClient.Groups.GetGroupsAsync()).Value;
    var currentWorkspace = workspaces.First((workspace) => workspace.Id == workspaceId);
    viewModel = new {
      workspaces = workspaces,
      currentWorkspace = currentWorkspace.Name,
      currentWorkspaceIsReadOnly = currentWorkspace.IsReadOnly,
      datasets = (await pbiClient.Datasets.GetDatasetsInGroupAsync(workspaceId)).Value,
      reports = (await pbiClient.Reports.GetReportsInGroupAsync(workspaceId)).Value,
      token = accessToken
    };
  }
  return JsonConvert.SerializeObject(viewModel);
}
```


```
{
  "workspaces": [
    { "id": "6679bd47-5b5f-4be0-ac6d-7a7ab1ba16f8", "name": "All Company", "isReadOn
    { "id": "912f2b34-7daa-4589-83df-35c75944d864", "name": "Dev Camp Demos", "isRea
  ],
  "currentWorkspace": "Dev Camp Demos",
  "currentWorkspaceIsReadOnly": false,
  "datasets": [
    { "id": "94e863ea-9117-445c-ac9e-db9373bdb8ea", "name": "COVID-19 US" },
    { "id": "89795189-0f43-4057-9af7-ab4838082a3b", "name": "Microsoft Graph Report"
    { "id": "bfa46105-5d6e-4270-88d8-eb6be5bf57ad", "name": "Wingtip Sales Analysis"
  ],
  "reports": [
    { "id": "cd496c1c-8df0-48e7-8b92-e2932298743e", "name": "COVID-19 US", "webUrl":
    { "id": "23d56559-af3c-4c51-9175-2904ee76bbae", "name": "Microsoft Graph Report"
    { "id": "fa89b3ae-c6a6-4e0c-8e4b-be7057cf583c", "name": "Wingtip Sales Analysis"
  ],
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImppYk5iaOZTU2JteFBZck45QOZx
};
```


Creating the User Experience


```
{
  "workspaces": [
    { "id": "6679bd47-5b5f-4be0-ac6d-7a7ab1ba16f8", "name": "All Company", "isReadOn
    { "id": "912f2b34-7daa-4589-83df-35c75944d864", "name": "Dev Camp Demos", "isRea
  ],
  "currentWorkspace": "Dev Camp Demos",
  "currentWorkspaceIsReadOnly": false,
  "datasets": [
    { "id": "94e863ea-9117-445c-ac9e-db9373bdb8ea", "name": "COVID-19 US" },
    { "id": "89795189-0f43-4057-9af7-ab4838082a3b", "name": "Microsoft Graph Report"
    { "id": "bfa46105-5d6e-4270-88d8-eb6be5bf57ad", "name": "Wingtip Sales Analysis"
  ],
  "reports": [
    { "id": "cd496c1c-8df0-48e7-8b92-e2932298743e", "name": "COVID-19 US", "webUrl":
    { "id": "23d56559-af3c-4c51-9175-2904ee76bbae", "name": "Microsoft Graph Report"
    { "id": "fa89b3ae-c6a6-4e0c-8e4b-be7057cf583c", "name": "Wingtip Sales Analysis"
  ],
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImppYk5iaOZTU2JteFBZck45QOZx
};
```

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

Questions