

Developing for Power BI with .NET 5

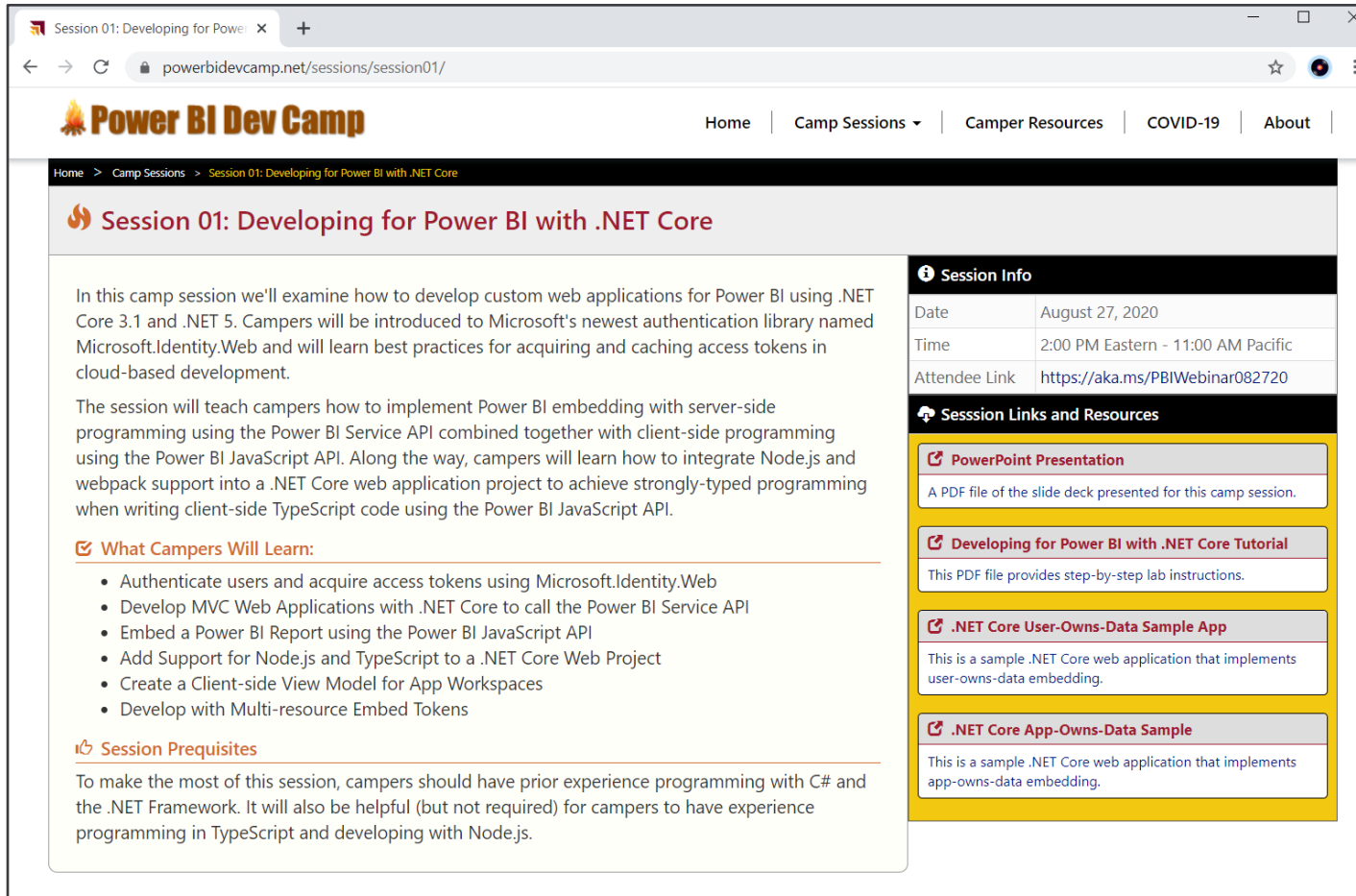
Ted Pattison

Principal Program Manager

Customer Advisory Team (CAT) at Microsoft

Welcome to Power BI Dev Camp

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



The screenshot shows a web browser window with the URL `powerbidevcamp.net/sessions/session01/`. The page features a navigation bar with links to Home, Camp Sessions, Camper Resources, COVID-19, and About. The main content area is titled "Session 01: Developing for Power BI with .NET Core". It includes a description of the session, a list of what campers will learn, and session prerequisites. On the right side, there is a "Session Info" table and a "Session Links and Resources" section with links to a PowerPoint presentation, a tutorial, and two sample applications.

Power BI Dev Camp

Home | Camp Sessions | Camper Resources | COVID-19 | About

Home > Camp Sessions > Session 01: Developing for Power BI with .NET Core

Session 01: Developing for Power BI with .NET Core

In this camp session we'll examine how to develop custom web applications for Power BI using .NET Core 3.1 and .NET 5. Campers will be introduced to Microsoft's newest authentication library named Microsoft.Identity.Web and will learn best practices for acquiring and caching access tokens in cloud-based development.

The session will teach campers how to implement Power BI embedding with server-side programming using the Power BI Service API combined together with client-side programming using the Power BI JavaScript API. Along the way, campers will learn how to integrate Node.js and webpack support into a .NET Core web application project to achieve strongly-typed programming when writing client-side TypeScript code using the Power BI JavaScript API.

What Campers Will Learn:

- Authenticate users and acquire access tokens using Microsoft.Identity.Web
- Develop MVC Web Applications with .NET Core to call the Power BI Service API
- Embed a Power BI Report using the Power BI JavaScript API
- Add Support for Node.js and TypeScript to a .NET Core Web Project
- Create a Client-side View Model for App Workspaces
- Develop with Multi-resource Embed Tokens

Session Prerequisites

To make the most of this session, campers should have prior experience programming with C# and the .NET Framework. It will also be helpful (but not required) for campers to have experience programming in TypeScript and developing with Node.js.

Session Info

Date	August 27, 2020
Time	2:00 PM Eastern - 11:00 AM Pacific
Attendee Link	https://aka.ms/PBIWebinar082720

Session Links and Resources

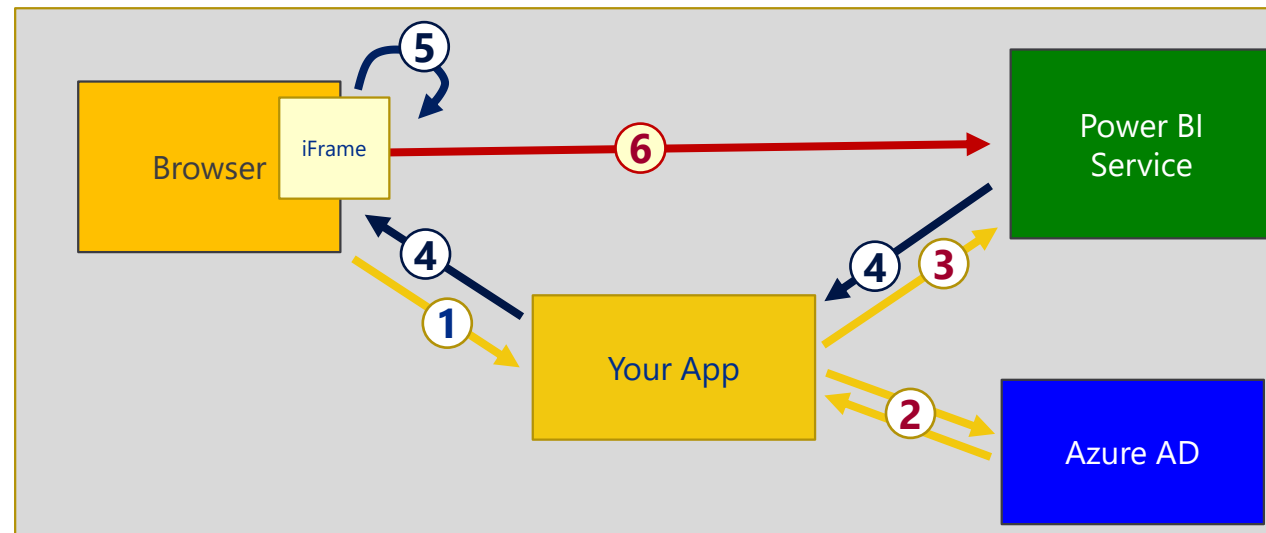
- PowerPoint Presentation**
A PDF file of the slide deck presented for this camp session.
- Developing for Power BI with .NET Core Tutorial**
This PDF file provides step-by-step lab instructions.
- .NET Core User-Owns-Data Sample App**
This is a sample .NET Core web application that implements user-owns-data embedding.
- .NET Core App-Owns-Data Sample**
This is a sample .NET Core web application that implements app-owns-data embedding.

Agenda

- Developing for Power BI with .NET 5
 - Introducing `Microsoft.Identity.Web`
 - Calling to Power BI as Service Principal
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET 5 Project
 - Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `EffectivelyIdentity`

Power BI Embedding – The Big Picture

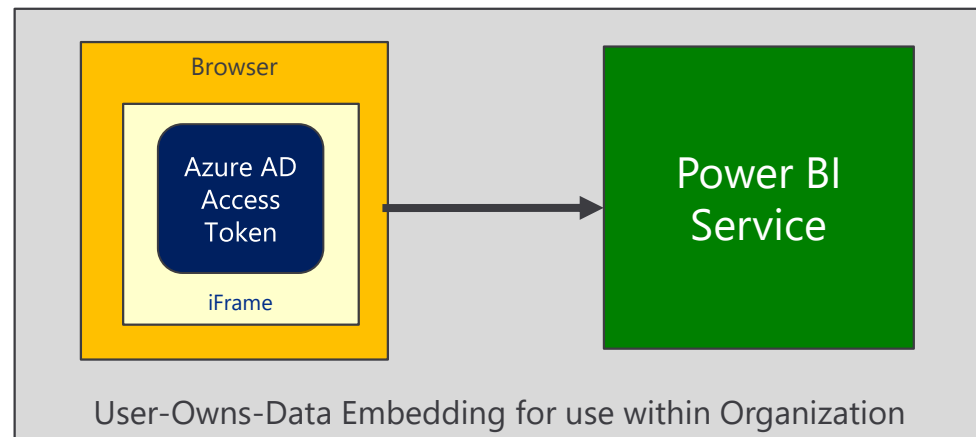
- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded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



Choosing the Correct Embedding Model

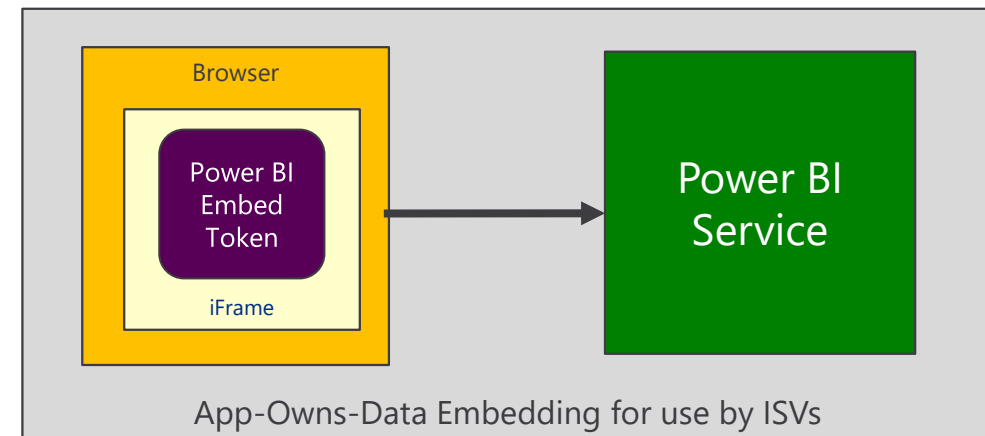
User-Owns-Data Embedding

- 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



App-Owns-Data .NET 5 Tutorial

- Stored in a GitHub repository for easy download
- <https://github.com/PowerBiDevCamp/DOTNET5-AppOwnsData-Tutorial>

The screenshot shows the GitHub repository page for `PowerBiDevCamp / DOTNET5-AppOwnsData-Tutorial`. The repository has 1 star and 0 forks. The main content area displays a list of files and folders, all of which have been updated 9 days ago or yesterday. The files include `Solution`, `StudentLabFiles`, `README.md`, `Tutorial.docx`, and `Tutorial.pdf`. The `README.md` file is selected, showing its content: "App-Owns-Data Embedding Tutorial with .NET 5". The right sidebar contains sections for "About", "Releases", and "Packages".

PowerBiDevCamp / DOTNET5-AppOwnsData-Tutorial

Unwatch 1 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

TedPattison Updates 6a92fa8 yesterday 21 commits

Solution	Updates	9 days ago
StudentLabFiles	Updates	9 days ago
README.md	Updates	9 days ago
Tutorial.docx	Updates	yesterday
Tutorial.pdf	Updates	yesterday

README.md

App-Owns-Data Embedding Tutorial with .NET 5

This tutorial teaches developers how to create a .NET 5 MVC web applicaiton that implements Power BI embedding using the App-Owns-Data embedding model.

About

This tutorial teaches developers how to create a .NET 5 MVC web application that implements Power BI Embedding using the App-Owns-Data development model.

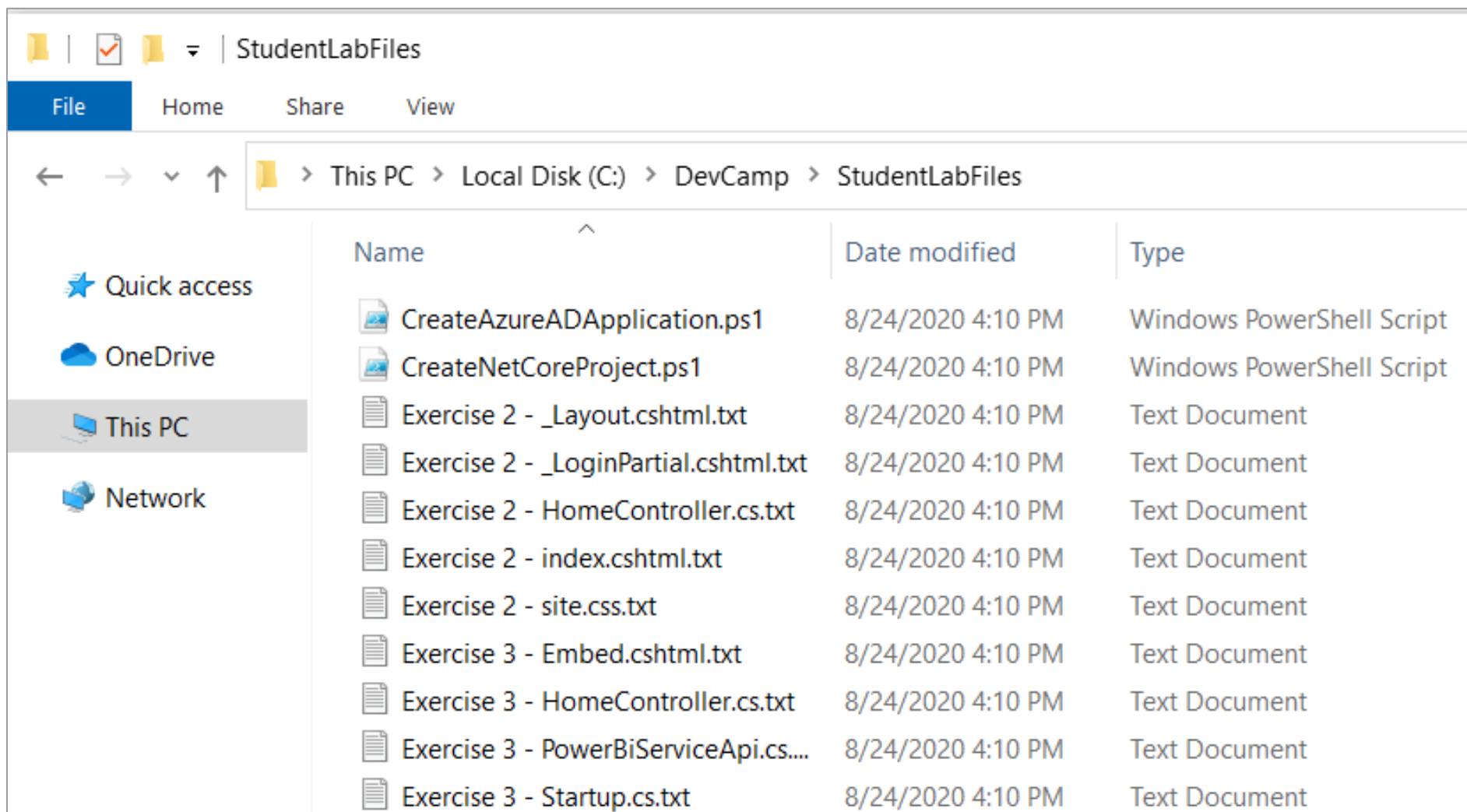
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Copying-and-Pasting Code



Tutorial Exercise Flow

- Exercise 1: Create a New Azure AD Application
- Exercise 2: Create a .NET 5 Project for a Secure Web Application
- Exercise 3: Call the Power BI Service API
- Exercise 4: Embedding a Report using powerbi.js
- Exercise 5: Adding TypeScript Support to a .NET 5 Project
- Exercise 6: Creating a View Model for App Workspaces

Creating Azure AD Application with PowerShell

```
$authResult = Connect-AzureAD
$userId = $authResult.Account.Id
$user = Get-AzureADUser -ObjectId $userId

$appDisplayName = "User-Owns-Data Sample App"
$returnUrl = "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
$adApplication = New-AzureADApplication `
    -DisplayName $appDisplayName `
    -PublicClient $false `
    -AvailableToOtherTenants $false `
    -ReplyUrls @($returnUrl) `
    -Homepage $returnUrl `
    -PasswordCredentials $passwordCredential

# assign current user as owner
$appId = $adApplication.AppId
Add-AzureADApplicationOwner -ObjectId $adApplication.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	952eBr.pO4-AZ03huF~f4NP_dDMJzmyGuu

Install .NET 5 SDK

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

Windows

Linux

macOS

Docker

.NET

.NET 5.0 (recommended)

Current

.NET is a free, cross-platform, open-source developer platform for building many different types of applications.

[Download .NET SDK x64](#)

[Download .NET Runtime](#)

[All .NET downloads](#)

.NET Core

.NET Core 3.1

LTS

.NET Core is a free, cross-platform, open-source developer platform for building many different types of applications.

[Download .NET Core SDK x64](#)

[Download .NET Core Runtime](#)

[All .NET Core downloads](#)

.NET Framework


.NET Framework 4.8

.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.

[Download .NET Framework Dev Pack](#)

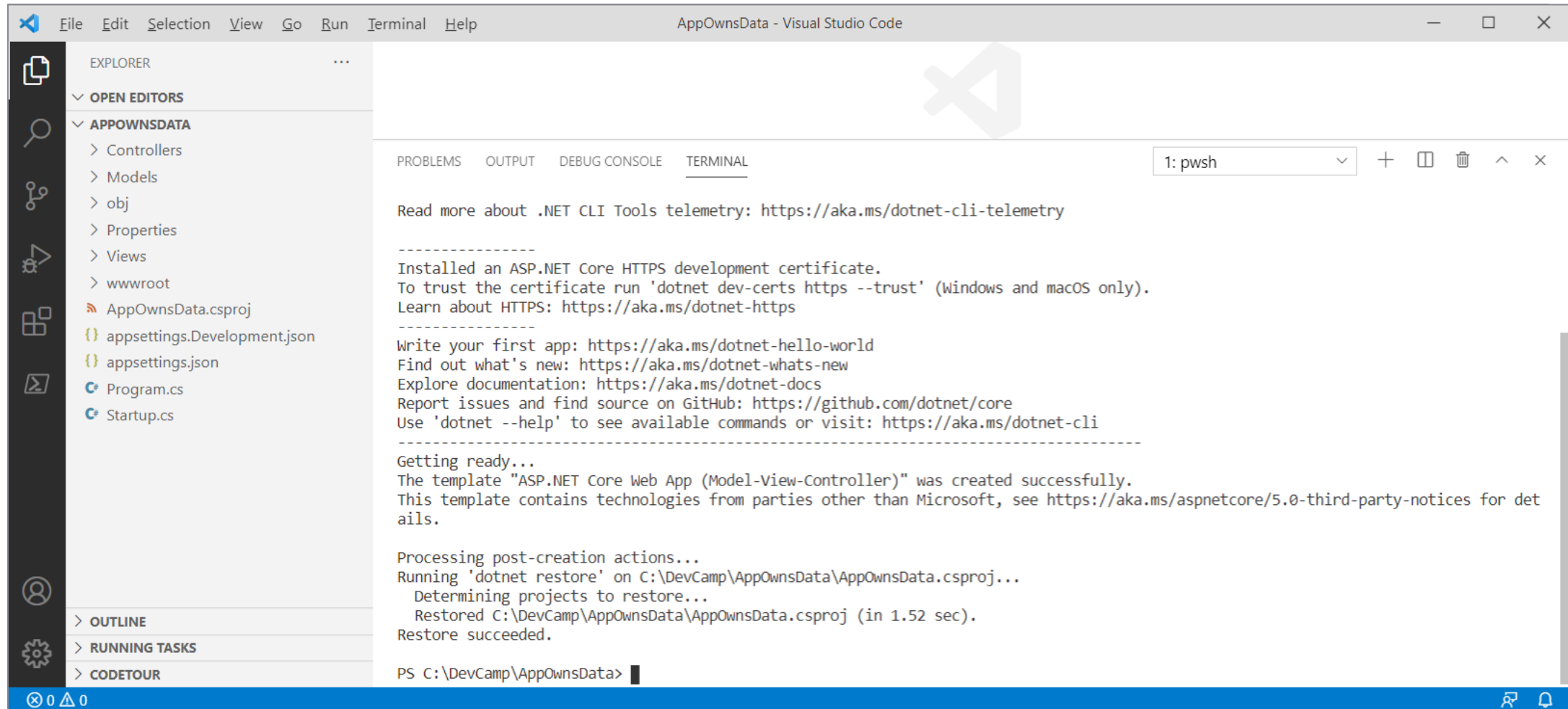
[Download .NET Framework Runtime](#)

[All .NET Framework downloads](#)



Getting Started with the .NET 5 CLI

- Command to create a new project > `dotnet new mvc -auth singleOrg`



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the project structure for 'AppOwnsData'. The main area shows the 'TERMINAL' tab with the following output:

```
Read more about .NET CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

-----
Installed an ASP.NET Core HTTPS development certificate.
To trust the certificate run 'dotnet dev-certs https --trust' (Windows and macOS only).
Learn about HTTPS: https://aka.ms/dotnet-https
-----
Write your first app: https://aka.ms/dotnet-hello-world
Find out what's new: https://aka.ms/dotnet-whats-new
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli
-----
Getting ready...
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/5.0-third-party-notices for details.

Processing post-creation actions...
Running 'dotnet restore' on C:\DevCamp\AppOwnsData\AppOwnsData.csproj...
    Determining projects to restore...
    Restored C:\DevCamp\AppOwnsData\AppOwnsData.csproj (in 1.52 sec).
Restore succeeded.

PS C:\DevCamp\AppOwnsData>
```

Azure AD Configuration in appsettings.json

The image shows a Visual Studio Code editor window with the file 'appsettings.json' open. The Explorer sidebar on the left shows the project structure, with 'appsettings.json' selected. The main editor displays the JSON content of 'appsettings.json'. A Notepad window on the right shows a duplicate of the same configuration, with an arrow pointing from the main editor to it.

Visual Studio Code Explorer:

- EXPLORER
- > OPEN EDITORS
- 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

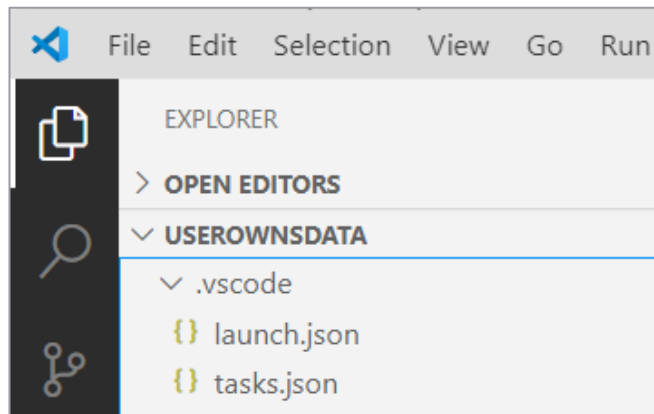
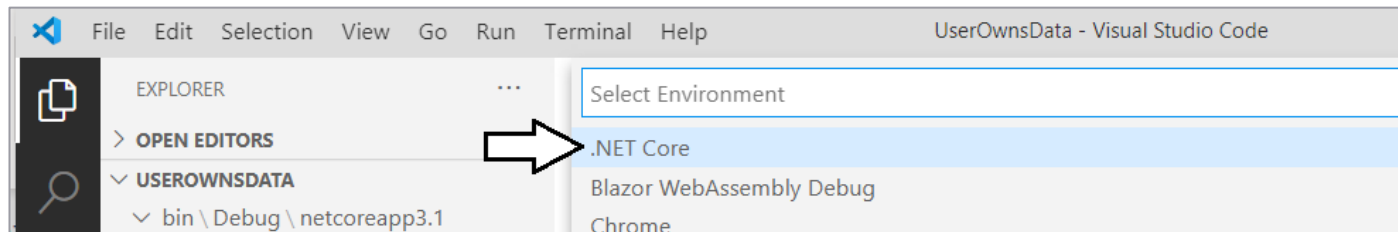
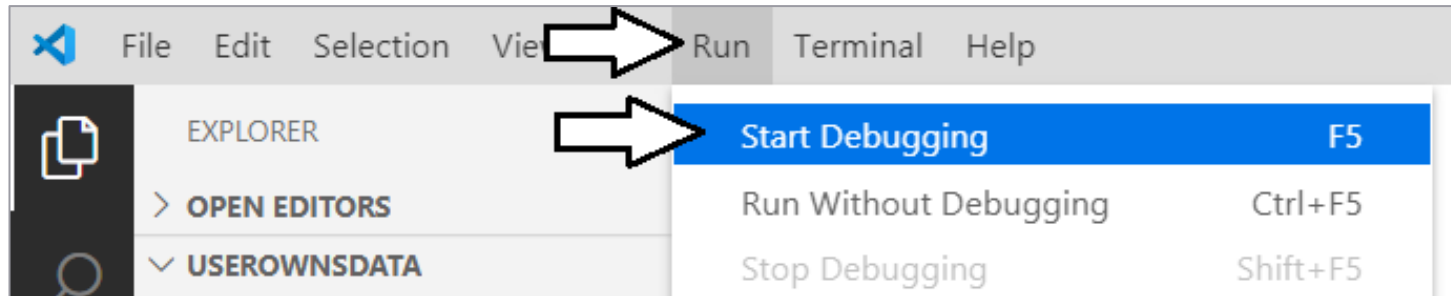
appsettings.json Content:

```
{
  "AzureAd": {
    "Instance": "https://login.microsoftonline.com/",
    "Domain": "powerbidevcamp.net",
    "TenantId": "2f23c5ea-5a75-41f6-922e-d3392313e61d",
    "ClientId": "6d8ab9b5-50fb-4468-a7fd-ced020c01737",
    "ClientSecret": "YjExNjI4NGQtZDlmNS00YTZkLTg2MmWYtYzg2YTRhN2Q4ODA5=",
    "CallbackPath": "/signin-oidc",
    "SignedOutCallbackPath": "/signout-callback-oidc"
  },
  "PowerBi": {
    "ServiceRootUrl": "https://api.powerbi.com/"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

***UserOwnsDataSampleApp.txt - Notepad:**

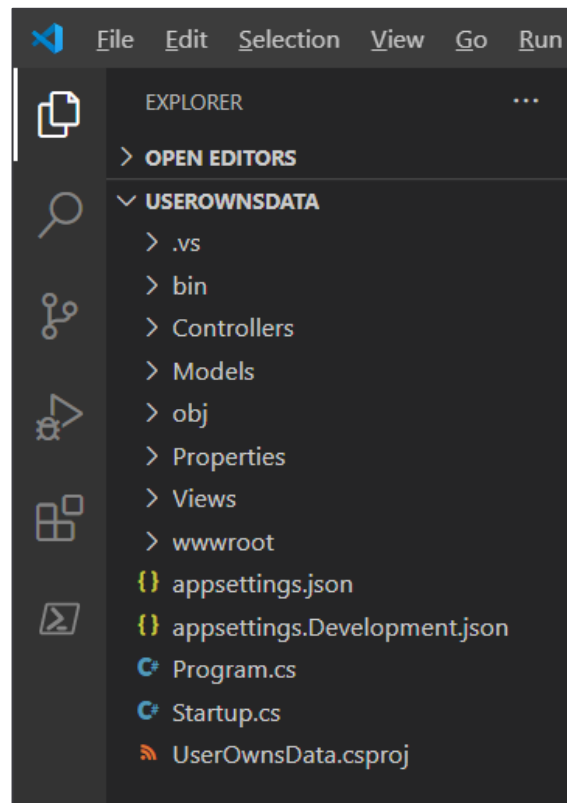
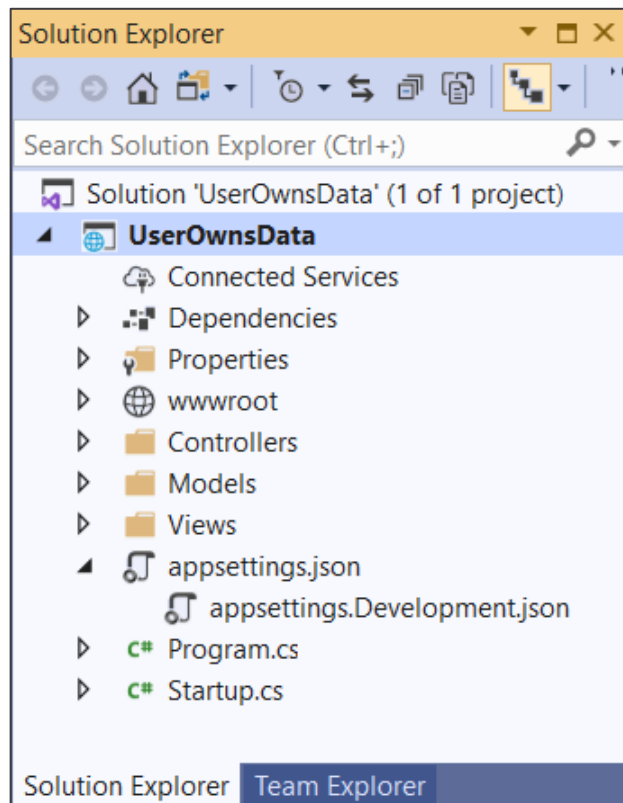
```
{
  "AzureAd": {
    "Instance": "https://login.microsoftonline.com/",
    "Domain": "powerbidevcamp.net",
    "TenantId": "2f23c5ea-5a75-41f6-922e-d3392313e61d",
    "ClientId": "6d8ab9b5-50fb-4468-a7fd-ced020c01737",
    "ClientSecret": "YjExNjI4NGQtZDlmNS00YTZkLTg2MmWYtYzg2YTRhN2Q4ODA5=",
    "CallbackPath": "/signin-oidc",
    "SignedOutCallbackPath": "/signout-callback-oidc"
  },
  "PowerBi": {
    "ServiceRootUrl": "https://api.powerbi.com/"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Starting a .NET 5 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 5 3.1 and .NET 5



Agenda

- ✓ Developing for Power BI with .NET 5
- Introducing `Microsoft.Identity.Web`
 - Calling to Power BI as Service Principal
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET 5 Project
 - Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `EffectivelyIdentity`


Introducing `Microsoft.Identity.Web`

- What is `Microsoft.Identity.Web`
 - Strategic authentication library for Azure AD to assist developers
 - Used to perform authentication in Web applications and Web APIs
 - Used to acquire access tokens
 - Used to implement token caching
- Scenarios where you should use `Microsoft.Identity.Web`
 - When developing Web applications and APIs built on .NET 3.1 and .NET 5
 - When authenticating users and acquiring user tokens
 - When authenticating service principals and acquiring app-only tokens
- More info at <https://github.com/AzureAD/microsoft-identity-web/wiki>

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();  
}
```

Adding Sign in / Sign Out Links



```
File Edit Selection View Go Run Terminal Help _LoginPartial.cshtml - Final - Visual Studio Code

EXPLORER
> OPEN EDITORS
FINAL
  UserOwnsData
    .vscode
    bin
    Controllers
    Models
    node_modules
    obj
    Properties
    Scripts
    Services
    Views
    Home
    Shared
    _Layout.cshtml
    _LoginPartial.cshtml
    _ValidationScriptsPartial.c...
    Error.cshtml

@using System.Security.Principal

<ul class="navbar-nav">
  @if (User.Identity.IsAuthenticated) {
    <li class="nav-item">
      <span class="navbar-text text-dark">Hello @User.FindFirst("name").Value</span>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="MicrosoftIdentity" asp-controller="Account" asp-action="SignOut">
        Sign out
      </a>
    </li>
  }
  else {
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="MicrosoftIdentity" asp-controller="Account" asp-action="SignIn">
        Sign in
      </a>
    </li>
  }
</ul>
```

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

```
Embed.cshtml X
<h2>TODO: Embed Report Here</h2>
```

Agenda

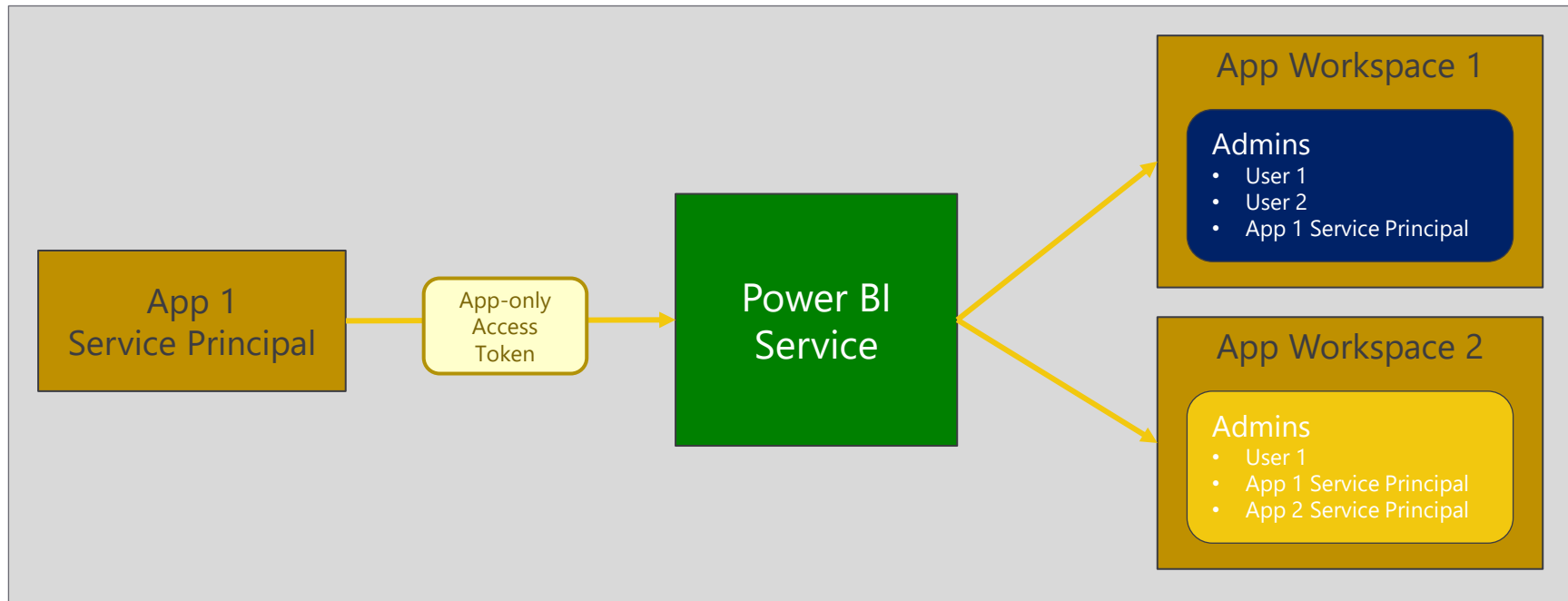
- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- Calling to Power BI as Service Principal
 - Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET 5 Project
 - Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `EffectivelyIdentity`

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

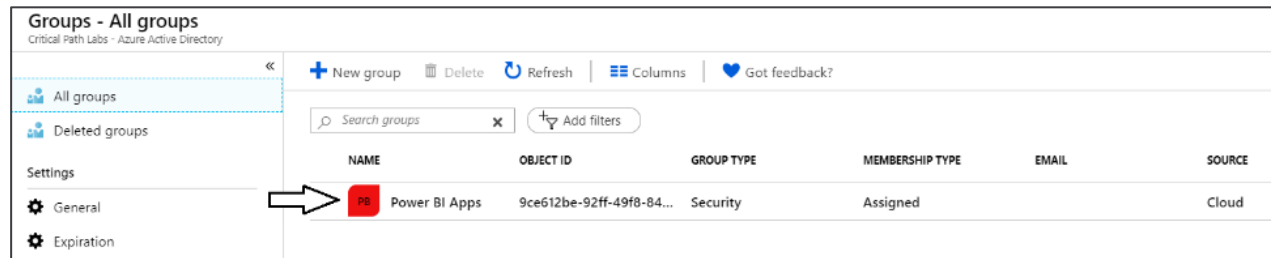
App-only Access Control

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

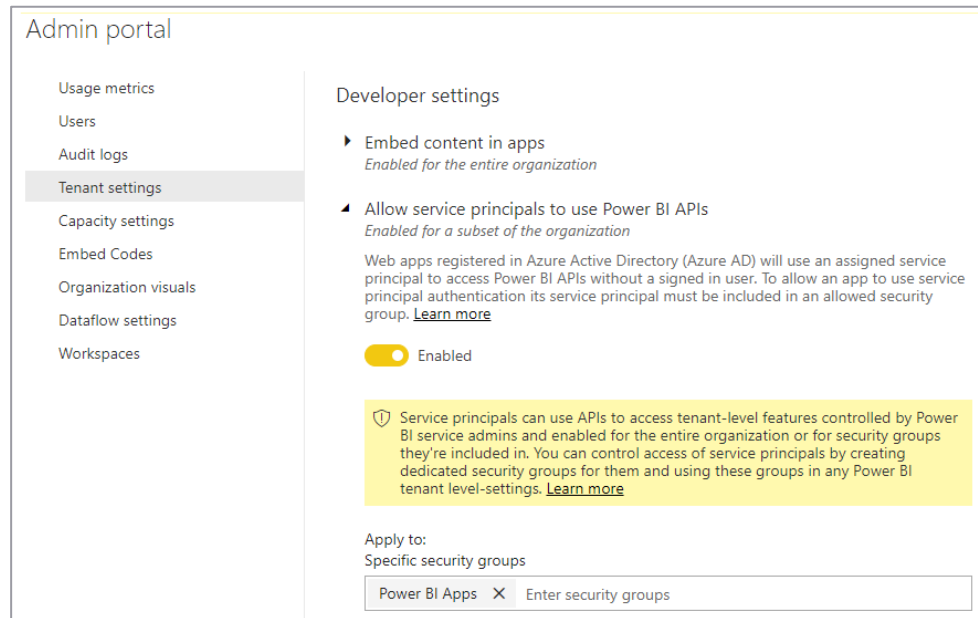


Setting Up for App-Owns-Data – Part 1

- Enable Service Principal Access to Power BI Service API
- Create an Azure AD security group (e.g. Power BI Apps)



- Add group to *Power BI Allow service principals to use Power BI APIs*



Setting Up for App-Owns-Data – Part 2

- Create a confidential client in your Azure AD tenant

Display name	: App-Owns-Data App	Supported account types	: My organization only
Application (client) ID	: af5d13b2-daa3-4b14-ac20-3ee338220630	Redirect URIs	: Add a Redirect URI
Directory (tenant) ID	: 17b8d777-a84a-4b90-b4a3-559ee5cbf07e	Managed application in ...	: App-Owns-Data App
Object ID	: 8f1f9327-f5cc-46b5-babf-6e821a5df079		

- Configured as **TYPE=Web** and no need for a redirect URL

Manage	Redirect URIs
Branding	The URIs that we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users. Also referred to as reply URLs. Learn more about adding support for web, mobile and desktop clients
Authentication	
Certificates & secrets	
API permissions	
Expose an API	

TYPE	REDIRECT URI
Web	e.g. https://myapp.com/auth

- Add a client secret or a client certificate

Manage	Client secrets
Branding	A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.
Authentication	+ New client secret
Certificates & secrets	
API permissions	
Expose an API	

DESCRIPTION	EXPIRES	VALUE
No description	6/3/2020	Hidden

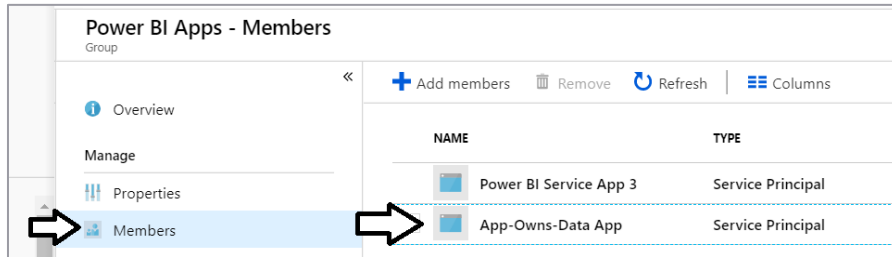
No need to configure any permissions

Manage	API permissions
Branding	Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.
Authentication	+ Add a permission
Certificates & secrets	
API permissions	
Expose an API	
Owners	

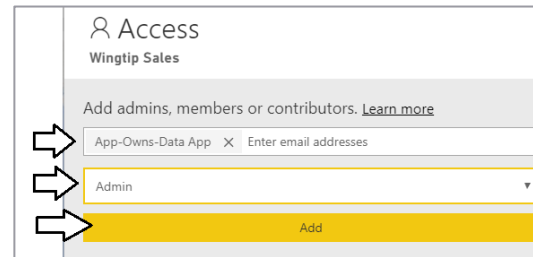
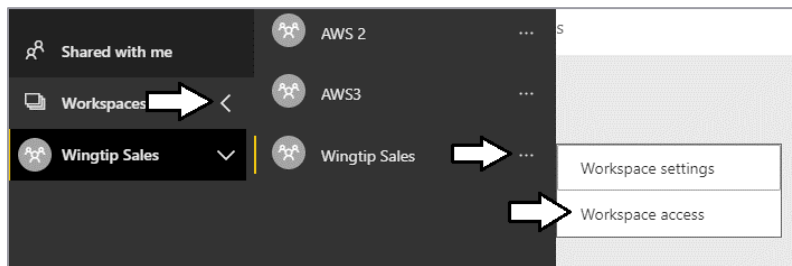
API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
No permissions added			

Setting Up for App-Owns-Data – Part 3

- Add application's service principal in Power BI Apps security group



- Configure application's service principal as workspace admin



- Service principal should now be workspace admin

NAME	PERMISSION	
App-Owns-Data App (Service Principal)	Admin	...
Power BI Service App 3 (Service Princip...	Admin	...
Ted Pattison	Admin	...

Calling Directly to the Power BI Service API

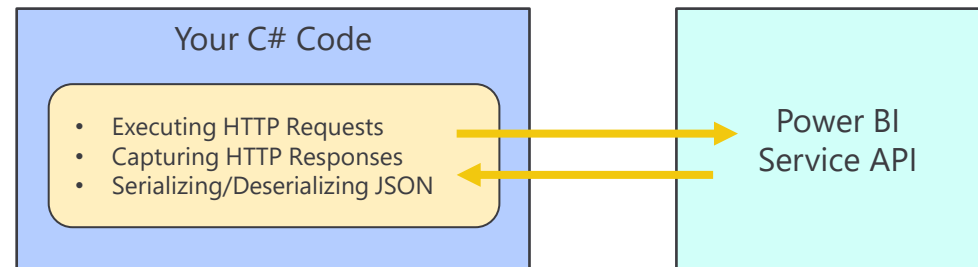
This is a sample of C# code that does not use the Power BI .NET SDK

```
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 occurred 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();
    }
}
```

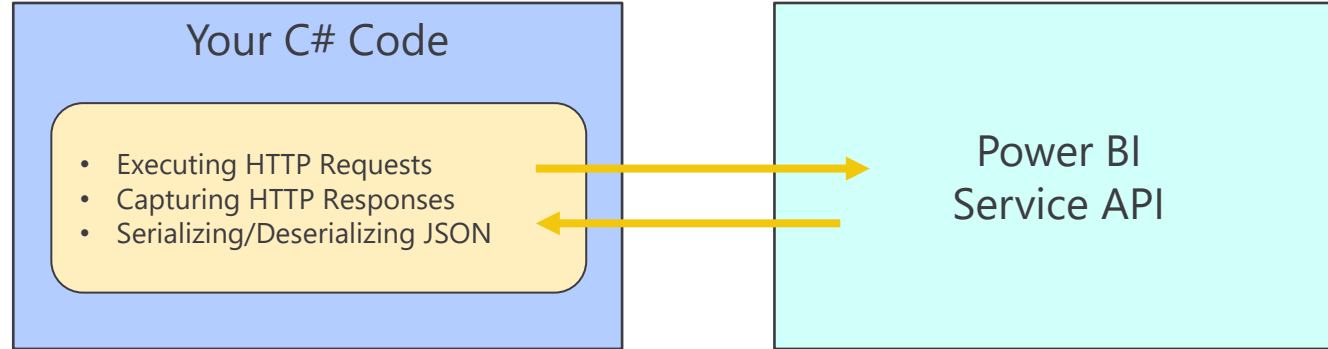
```
public class Report {
    public string id { get; set; }
    public string name { get; set; }
    public string webUrl { get; set; }
    public string embedUrl { get; set; }
    public bool isOwnedByMe { get; set; }
    public string datasetId { get; set; }
}

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



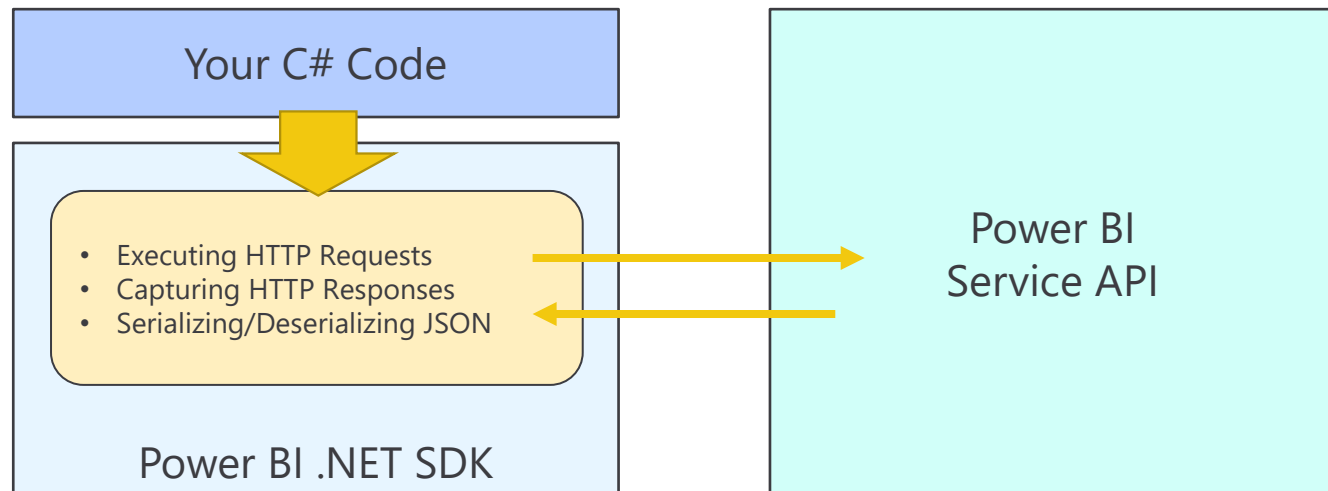
Power BI .NET SDK

• Developing without 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 occurred 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();  
    }  
}
```

• Developing with the Power BI .NET SDK



```
PowerBIClient pbiclient = GetPowerBIClient();  
  
// call to Power BI Service API to get embedding data  
var report = await pbiclient.Reports.GetReportInGroupAsync(workspaceId, ReportId);
```

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

Adding Support for Token Acquisition

- Modify ConfigureServices 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

Authenticating with Azure AD as Service Principal

```
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 const string powerbiApiDefaultScope = "https://analysis.windows.net/powerbi/api/.default";  
  
    public string GetAccessToken() {  
        return this.tokenAcquisition.GetAccessTokenForAppAsync(powerbiApiDefaultScope).Result;  
    }  
  
    public PowerBIClient GetPowerBiClient() {  
        var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");  
        return new PowerBIClient(new Uri(urlPowerBiServiceApiRoot), tokenCredentials);  
    }  
}
```

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

**MSAL Token
Acquisition Service**

Dependency
Injection

PowerBiServiceAPI

Dependency
Injection

HomeController

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()  
    };  
}
```


Generating Embed Tokens

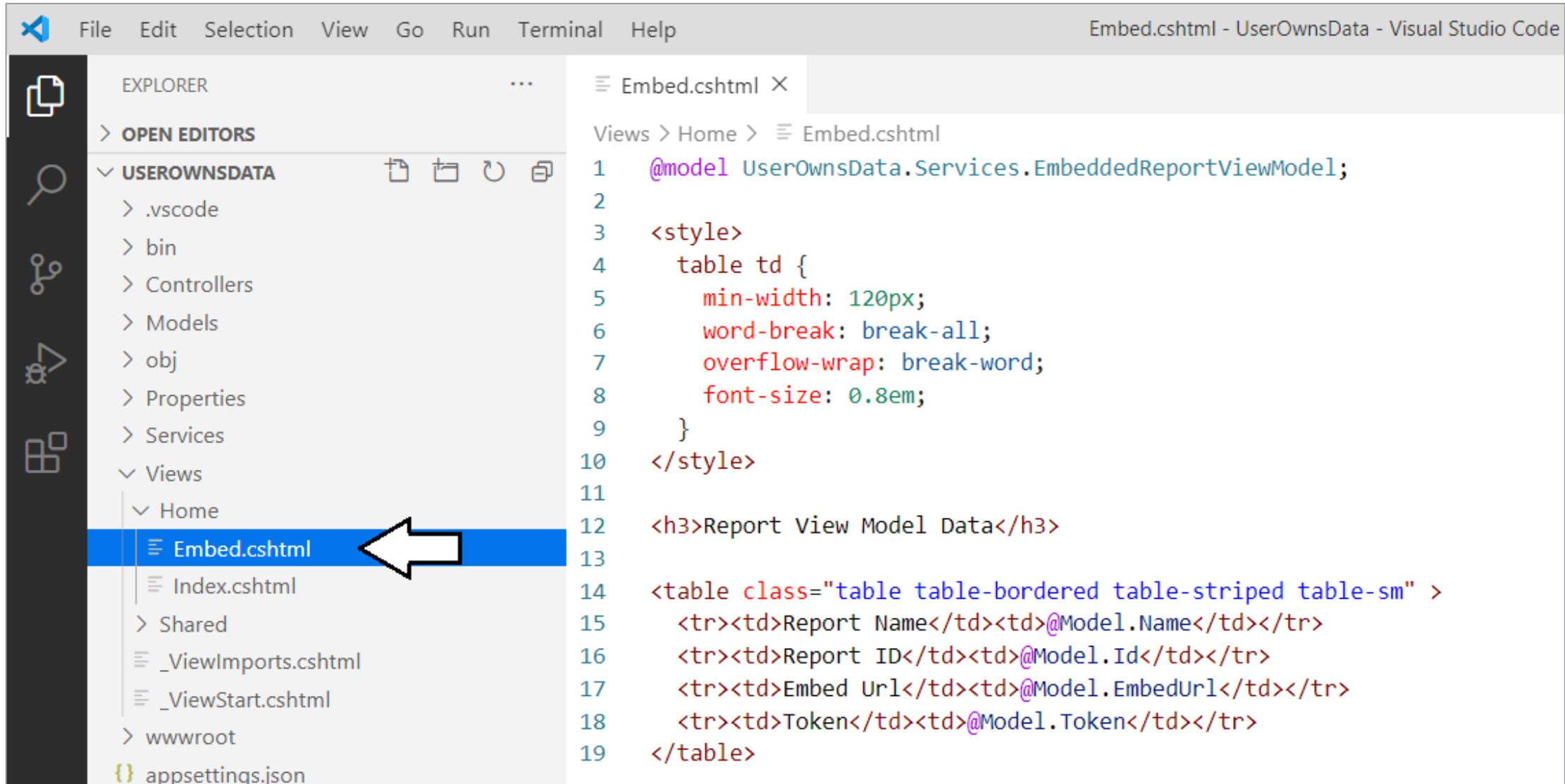
- **App-Owns-Data embedding requires developer to generate embed tokens**
 - Developer controls what permissions are extended to user for embedding purposes
 - Generating embed tokens requires dedicated capacity for any production scenarios
- **You generate embed tokens with the Power BI Service API**
 - Embed Token V1 API used to create single resource embed tokens (see example below)
 - Embed Token V2 API used to create multi-resource embed tokens (see example in later slide)
 - Embed token provides restrictions on whether user can view, edit or create
 - Embed token can be generated to support row-level security (RLS)

```
Report report = reports.Where(r => r.Id == reportId).First();  
var generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");  
var token = client.Reports.GenerateTokenInGroupAsync(appWorkspaceId,  
                                                    report.Id,  
                                                    generateTokenRequestParameters).Result;
```

Getting the Data for Report Embedding

```
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);  
  
    // generate read-only embed token for the report  
    var datasetId = report.DatasetId;  
    var tokenRequest = new GenerateTokenRequest(TokenAccessLevel.View, datasetId);  
    var embedTokenResponse = await pbiClient.Reports.GenerateTokenAsync(WorkspaceId, ReportId, tokenRequest);  
    var embedToken = embedTokenResponse.Token;  
  
    // return report embedding data to caller  
    return new EmbeddedReportViewModel {  
        Id = report.Id.ToString(),  
        EmbedUrl = report.EmbedUrl,  
        Name = report.Name,  
        Token = embedToken  
    };  
}
```

Displaying Data from the View Model



Visual Studio Code interface showing the file Explorer on the left and the code editor on the right.

The Explorer sidebar shows the project structure:

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

The code editor displays the content of `Embed.cshtml`:

```
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>
```

Exercise 3 Solution

[illegible]

Agenda

- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling to Power BI as Service Principal
- Programming the Power BI JavaScript API
 - Adding TypeScript Support to a .NET 5 Project
 - Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `EffectivelyIdentity`

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

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

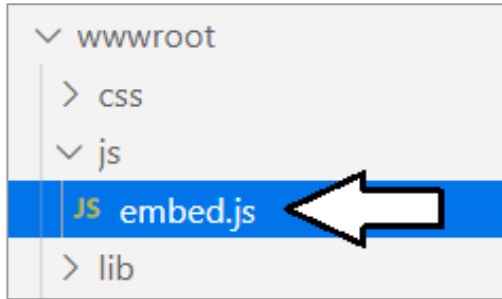
≡ Embed.cshtml ×

```
@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>
}
```

Retrieving Data from the Report View Model



```
JS embed.js X

$(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

});
```

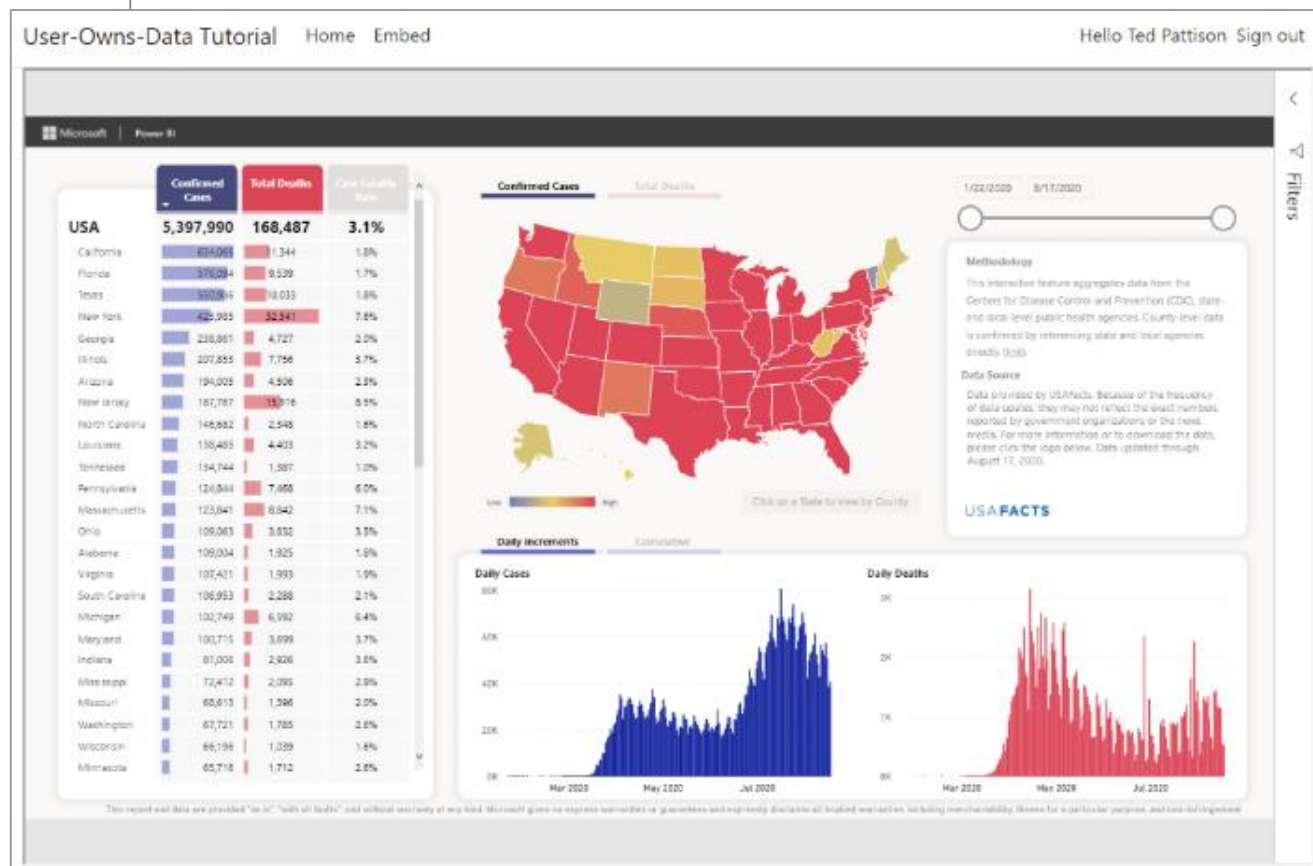

Embedding a Report using powerbi.js

```
// 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);
```



Agenda

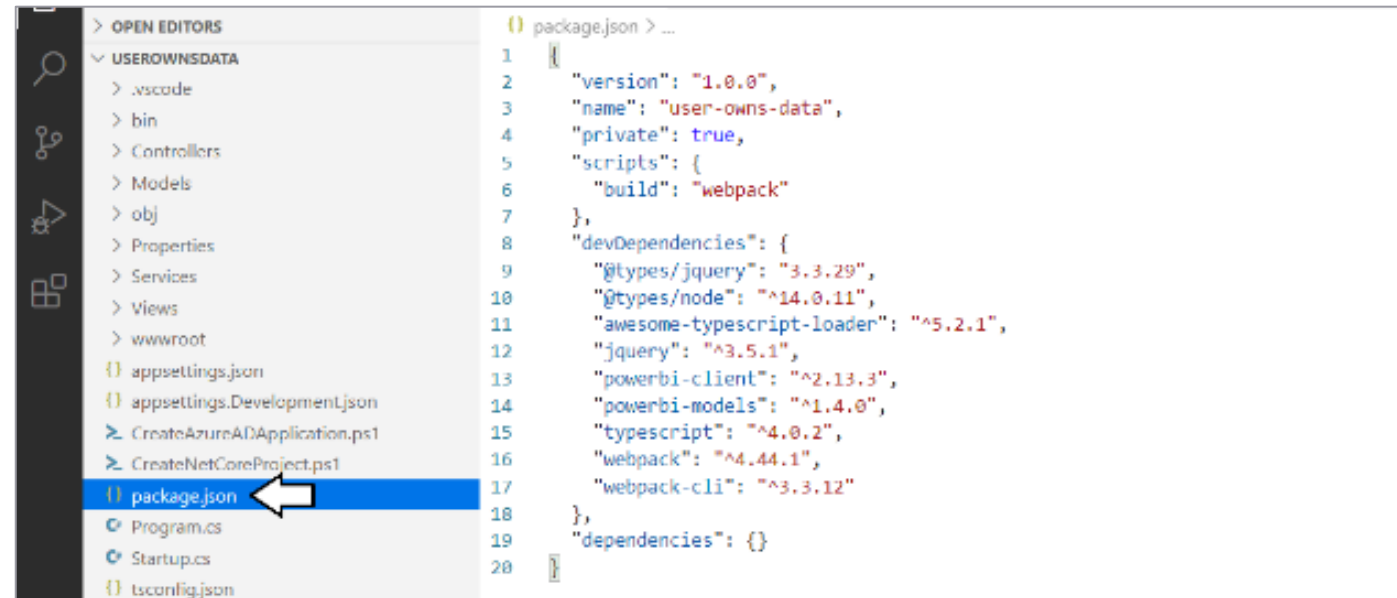
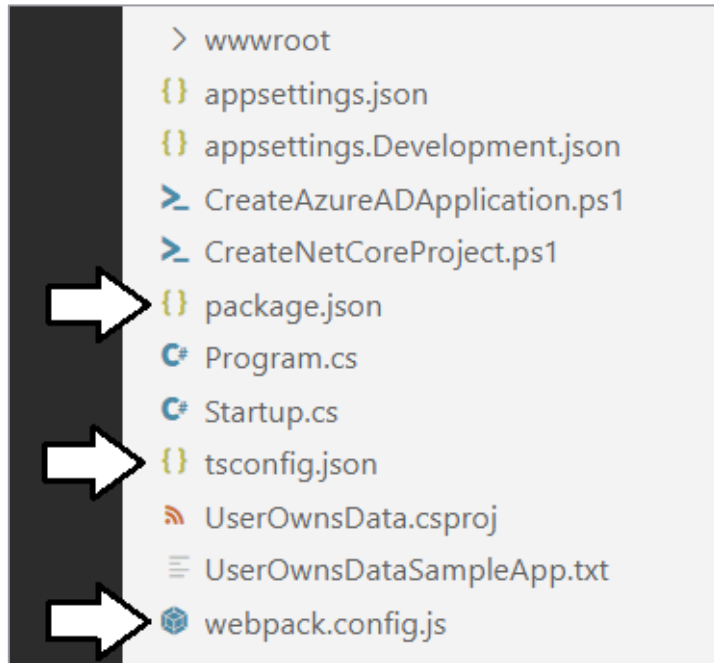
- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling to Power BI as Service Principal
- ✓ Programming the Power BI JavaScript API
- Adding TypeScript Support to a .NET 5 Project
 - Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `Effectiveldentity`

Exercise 5

Adding TypeScript/Webpack Support to a .NET 5 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

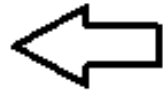
Adding Node.js Support for TypeScript Compilation



Running npm install

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\DevCamp\UserOwnsData> npm install



File Edit Selection View Go Run Terminal Help UserOwnsData - Visual Studio Code

1: powershell

EXPLORER

> OPEN EDITORS

USEROWNSDATA

- > .vscode
- > bin
- > Controllers
- > Models
- node_modules
- > .bin
- > @types
- > @webassemblyjs
- > @xtuc
- > acorn
- > ajv

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\DevCamp\UserOwnsData> npm install
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to fsevents 2.
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN notsup Unsupported engine for watchpack-chokidar2@2.0.0: wanted: {"node": "<8.10.0"} (current: {"node": "10.21.0", "npm": "6.14.4"})
npm WARN notsup Not compatible with your version of node/npm: watchpack-chokidar2@2.0.0
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.1.2 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node_modules\watchpack-chokidar2\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
npm WARN awesome-typescript-loader@5.2.1 requires a peer of typescript@^2.7 || ^3 but none is installed. You must install peer dependencies you
added 423 packages from 306 contributors and audited 426 packages in 11.713s

7 packages are looking for funding
run `npm fund` for details
```

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

The image shows a Visual Studio Code interface with a file explorer on the left and a terminal on the right. A white arrow points to the `embed.js` file in the `js` folder. A red arrow points from the `embed.js` file in the file explorer to the same file in the editor window.

Terminal Output:

```
PS C:\DevCamp\UserOwnsData> npm run build

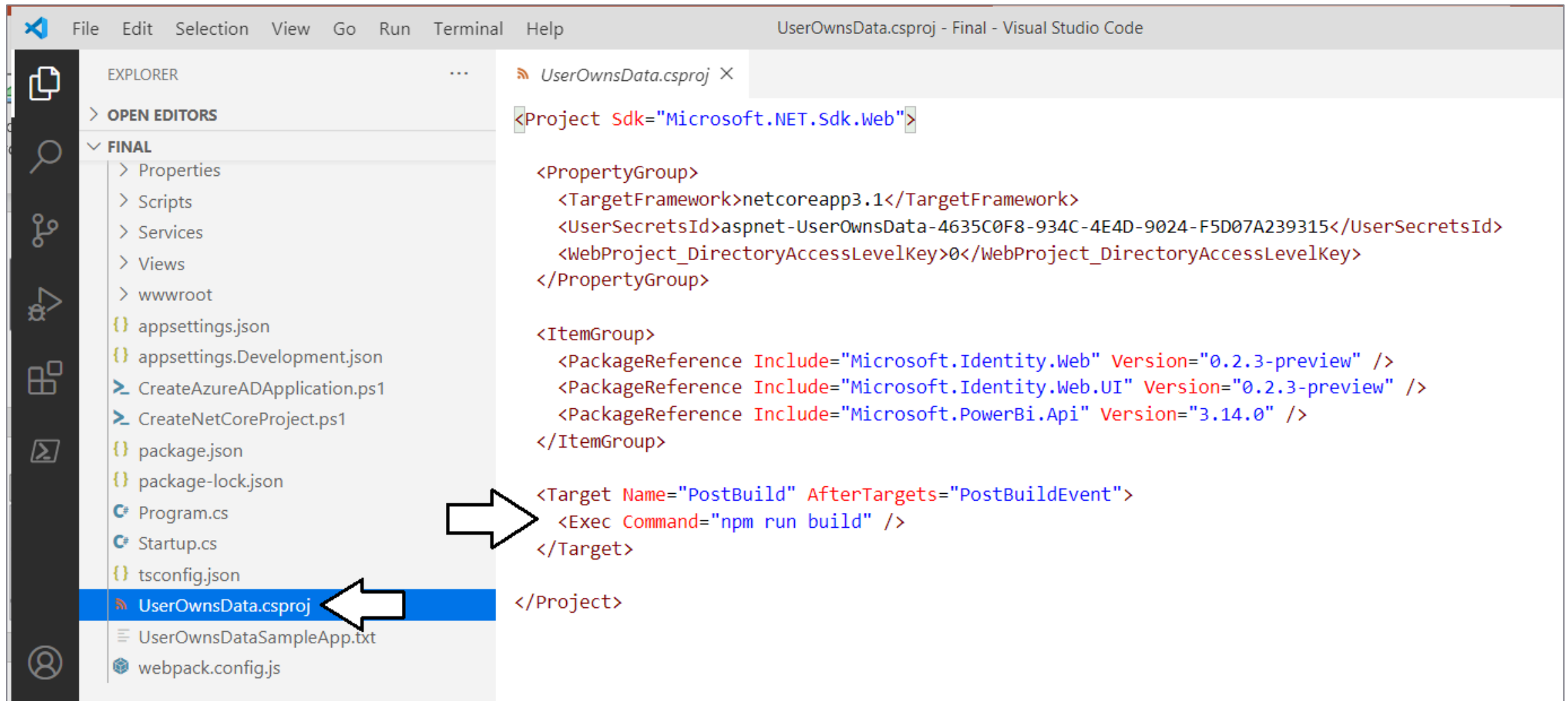
> user-owns-data@1.0.0 build C:\DevCamp\UserOwnsData
> webpack

i | atl|: Using typescript@4.0.2 from typescript
i | atl|: Using tsconfig.json from C:/DevCamp/UserOwnsData/tsconfig.json
i | atl|: Checking started in a separate process...
i | atl|: Time: 655ms
Hash: f000960cc21af308caff
Version: webpack 4.44.1
Time: 2854ms
Built at: 08/25/2020 1:37:14 PM
```

Editor Content (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) {
```

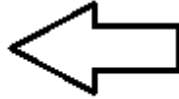

Updating UserOwnsData.csproj



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+ee1c9fd0c 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
```

```
i n!óatln!ú: Using typescript@4.0.2 from typescript
i n!óatln!ú: Using tsconfig.json from C:/DevCamp/UserOwnsData/tsconfig.json
i n!óatln!ú: Checking started in a separate process...
i n!óatln!ú: Time: 680ms
Hash: f000960cc21af308caff
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       0 [emitted]      main
  embed.js.map 975 KiB       0 [emitted] [dev] main
Entrypoint main = embed.js embed.js.map
[./Scripts/embed.ts] 1.65 KiB {main} [built]
+ 3 hidden modules
```

```
Build succeeded.
0 Warning(s)
0 Error(s)
```

```
Time Elapsed 00:00:08.55
PS C:\DevCamp\UserOwnsData>
```

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);
```

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", "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-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": "https://powerbi.com/reports/covid-19-us" },  
    { "id": "23d56559-af3c-4c51-9175-2904ee76bbae", "name": "Microsoft Graph Report" },  
    { "id": "fa89b3ae-c6a6-4e0c-8e4b-be7057cf583c", "name": "Wingtip Sales Analysis" }  
  ],  
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImppYk5ia0ZTU2JteFBZck45Q0Zx" }  
}
```

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-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": "/reports/covid-19-us" },
    { "id": "23d56559-af3c-4c51-9175-2904ee76bbae", "name": "Microsoft Graph Report", "webUrl": "/reports/microsoft-graph-report" },
    { "id": "fa89b3ae-c6a6-4e0c-8e4b-be7057cf583c", "name": "Wingtip Sales Analysis", "webUrl": "/reports/wingtip-sales-analysis" }
  ],
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6ImppYk5ia0ZTU2JteFBZck45Q0Zx"
};
```

User-Owens-Data Tutorial

Current Workspace	Reports
Dev Camp Demos	
Open Report	
COVID-19 US Microsoft Graph Report Wingtip Sales Analysis	
New Report	
COVID-19 US Microsoft Graph Report Wingtip Sales Analysis	

Agenda

- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling to Power BI as Service Principal
- ✓ Programming the Power BI JavaScript API
- ✓ Adding TypeScript Support to a .NET 5 Project
- Programming with Multi-Resource Embed Tokens
 - Integrating RLS using `EffectivelyIdentity`

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;
```

Agenda

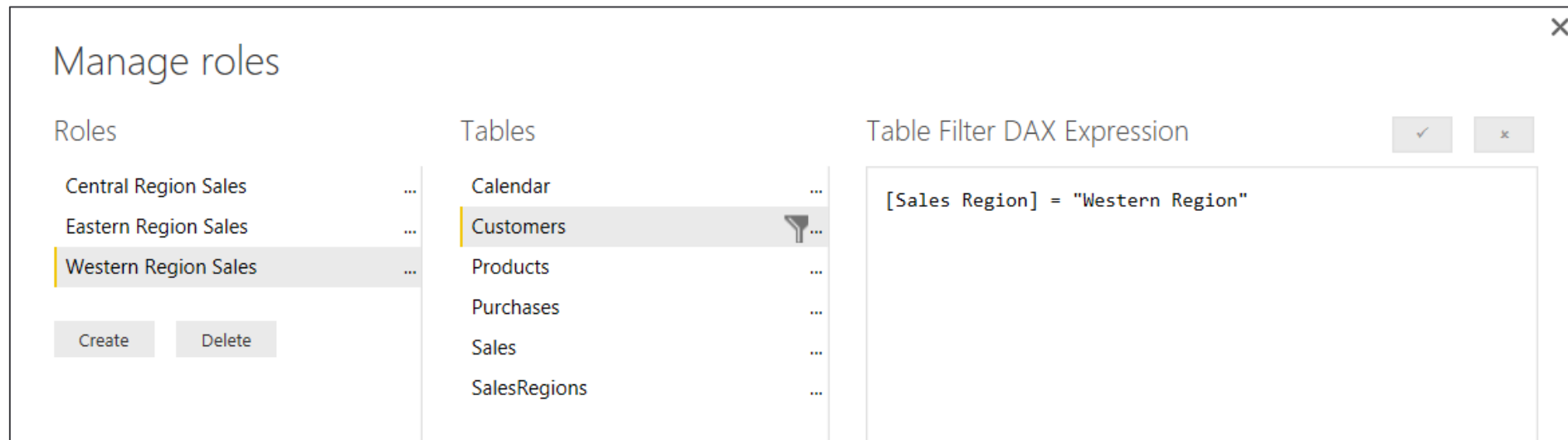
- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling to Power BI as Service Principal
- ✓ Programming the Power BI JavaScript API
- ✓ Adding TypeScript Support to a .NET 5 Project
- ✓ Programming with Multi-Resource Embed Tokens
- Integrating RLS using `EffectivelyIdentity`

App-Owns-Data Embedding with RLS

- What is Row-level Security (RLS)?
 - Ssss
 - Sss
- App-Owns-Data embedding model supports row-level security (RLS)
 - xxx

What Is Row-level Security (RLS)?

- **Security Scheme for Power BI Datasets based on Named Roles**
 - Roles are defined using Power BI Desktop or Tabular Object Model (TOM)
 - Each role is scoped to the dataset
- **Role defined using one or more DAX expressions**
 - DAX expressions restrict which rows are accessible



Embedding RLS-enabled Reports

- Programming with EffectivenessIdentity

- Trying to embed an RLS-enabled report without EffectivenessIdentity will fail
- EffectivenessIdentity containing 1 or more roles must be added to embed token
- Roles added into EffectivenessIdentity as string array

```
GenerateTokenRequest generateTokenRequestParameters =  
    new GenerateTokenRequest(accessLevel: accessLevel,  
                             identities: new List<EffectivenessIdentity> {  
                                 new EffectivenessIdentity(username: currentUser.UserName,  
                                                           datasets: new List<string> { datasetId },  
                                                           roles: roles) });  
  
string embedToken =  
    (await pbiClient.Reports.GenerateTokenInGroupAsync(workspaceId,  
                                                         report.Id,  
                                                         generateTokenRequestParameters)).Token;
```

Generating Embed Tokens with EffectivenessIdentity

```
public async Task<EmbeddedReportViewModel> GetReportWithRls(string UserName, string[] Roles, bool CustomizationEnabled) {  
  
    PowerBIClient pbiClient = GetPowerBiClient();  
  
    // call to Power BI Service API to get embedding data  
    var report = await pbiClient.Reports.GetReportInGroupAsync(WorkspaceId, RlsReportId);  
  
    // generate read-only embed token for the report  
    var datasetId = report.DatasetId;  
    var datasetList = new List<string>() { report.DatasetId };  
  
    // create EffectivenessIdentity object  
    var effectivenessIdentity = new EffectivenessIdentity(UserName, datasetList, Roles);  
  
    // generate embed token  
    TokenAccessLevel tokenAccessLevel = CustomizationEnabled ? TokenAccessLevel.Edit : TokenAccessLevel.View;  
    var tokenRequest = new GenerateTokenRequest(tokenAccessLevel, datasetId, effectivenessIdentity);  
    var embedTokenResponse = await pbiClient.Reports.GenerateTokenAsync(WorkspaceId, RlsReportId, tokenRequest);  
    var embedToken = embedTokenResponse.Token;  
  
    // return report embedding data to caller  
    return new EmbeddedReportViewModel {  
        Id = report.Id.ToString(),  
        EmbedUrl = report.EmbedUrl,  
        Name = report.Name,  
        Token = embedToken,  
        CustomizationEnabled = CustomizationEnabled  
    };  
}
```

Summary

- ✓ Developing for Power BI with .NET 5
- ✓ Introducing `Microsoft.Identity.Web`
- ✓ Calling to Power BI as Service Principal
- ✓ Programming the Power BI JavaScript API
- ✓ Adding TypeScript Support to a .NET 5 Project
- ✓ Programming with Multi-Resource Embed Tokens
- ✓ Integrating RLS using `Effectiveldentity`

Questions