

Developing with the Power BI REST API



Agenda

- Getting Started with the Power BI REST API
- Creating a Real-time Dashboard
- Creating an SPA for Power BI using Angular.js
- Embedding Reports and Dashboard Tiles



The Power BI REST API

- What is Power BI?
 - Collection of cloud services & tools for building BI solutions
 - Power BI service available to licensed subscribers
 - Power BI model based on datasets, reports and dashboards
- What is the Power BI REST API?
 - API built on OAuth2, OpenID Connect, REST and ODATA
 - Allows developers to program datasets, reports and dashboards
- What can you do with the Power BI REST API?
 - Create and update datasets
 - Build real-time dashboards
 - Embed Power BI reports and dashboards tiles in web pages



Getting Started

- What you need to get started?
 - Organizational account in an Office 365 tenancy
 - License for Power BI
 - Visual Studio 2017 and Visual Studio 2015
 - Azure subscription for application registration



What Operations Are Supported in v1.0?

- Workspace Group Operations
 - Get Groups
- Dataset Operations
 - Get Datasets
 - Create Dataset
- Table Operations
 - Get Tables
 - Alter Table Schema
- Table Row Operations
 - Add Rows
 - Delete Rows



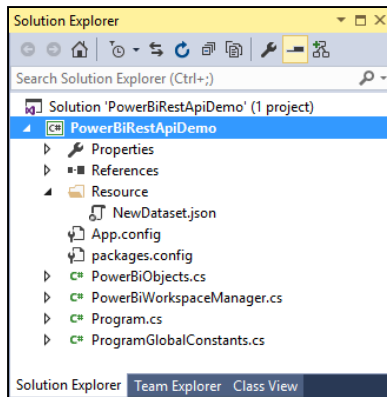
What Operations are Supported in Beta?

- Report Operations
 - Get Reports
- Dashboard Operations
 - Get Dashboards
 - Get Dashboard Tiles
- Import Operations
 - Create Import
 - Get Imports
 - Get Import by GUID
 - Get Import by File Path



The PowerBiRestApiDemo Sample Project

- Console application project in Visual Studio 2015

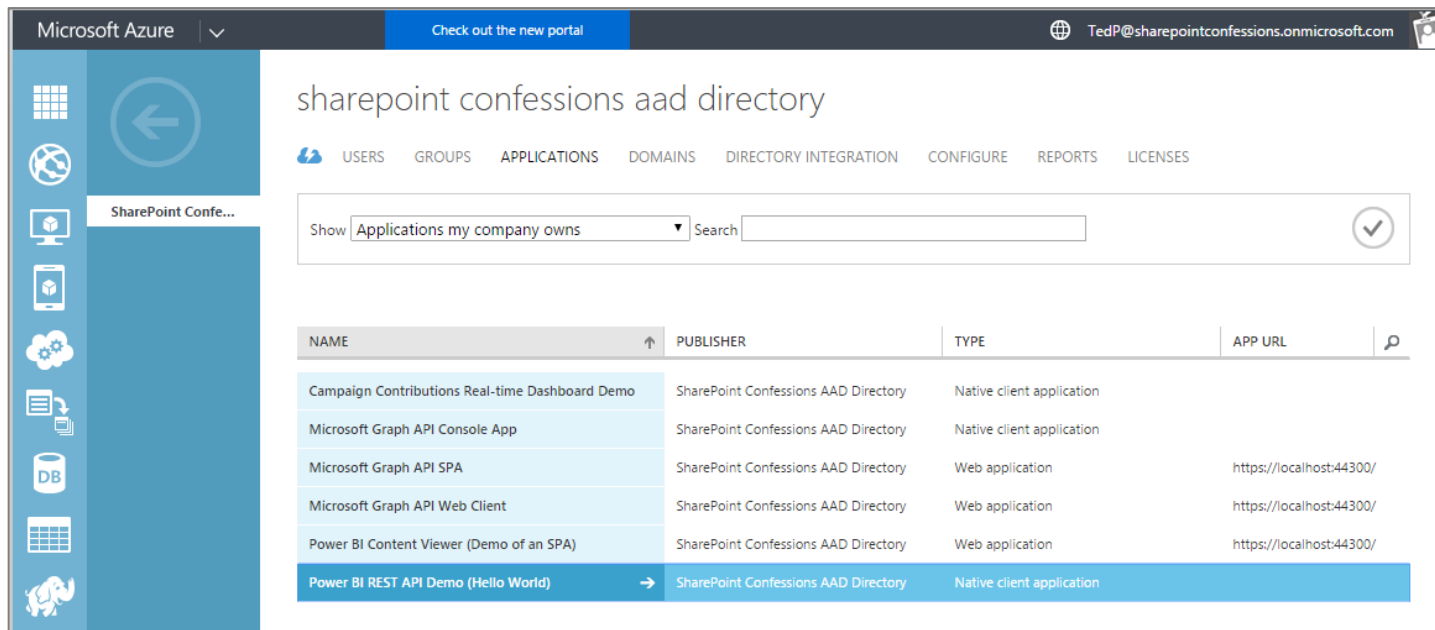


- PowerBiRestApiDemo project contains two NuGet packages
 - Newtonsoft.Json used to convert JSON data to and from and C#
 - Azure ADAL used to get access tokens from Azure Active Directory



App Registration with an Azure Tenant

- Application must be registered within an Office 365 tenant
 - Registration can be accomplished using Azure Management portal
 - Registration can be done using Power BI App Registration Page



The screenshot displays the Microsoft Azure portal interface. The top navigation bar includes the 'Microsoft Azure' logo, a 'Check out the new portal' button, and the user's email 'TedP@sharepointconfessions.onmicrosoft.com'. The left sidebar contains various service icons, with 'SharePoint Confessions' highlighted. The main content area is titled 'sharepoint confessions aad directory' and features a navigation menu with 'USERS', 'GROUPS', 'APPLICATIONS', 'DOMAINS', 'DIRECTORY INTEGRATION', 'CONFIGURE', 'REPORTS', and 'LICENSES'. The 'APPLICATIONS' tab is active, showing a list of applications. A search bar at the top of the list is set to 'Applications my company owns'. The table below lists several applications, with 'Power BI REST API Demo (Hello World)' selected.

NAME	PUBLISHER	TYPE	APP URL
Campaign Contributions Real-time Dashboard Demo	SharePoint Confessions AAD Directory	Native client application	
Microsoft Graph API Console App	SharePoint Confessions AAD Directory	Native client application	
Microsoft Graph API SPA	SharePoint Confessions AAD Directory	Web application	https://localhost:44300/
Microsoft Graph API Web Client	SharePoint Confessions AAD Directory	Web application	https://localhost:44300/
Power BI Content Viewer (Demo of an SPA)	SharePoint Confessions AAD Directory	Web application	https://localhost:44300/
Power BI REST API Demo (Hello World)	SharePoint Confessions AAD Directory	Native client application	



Power BI App Registration

Power BI Development Center

https://app.powerbi.com/apps

Power BI for Developers

Register an Application for Power BI

Register a new application that can be used to call Power BI APIs

Step 1: Login to your Power BI account

Welcome, Ted Pattison! (Wrong account? No problem, logout and try again.)

Step 2: Tell us about your app

Let's start with some basic details.

App Name:
Power BI Rest API Demo Console App

App Type:
Specify the type of app. Use "Server-side Web app" for web apps or Web APIs, or "Native app" for apps that run on client devices (Android, iOS, Windows, etc.).
Native app

Redirect URL:
A valid URL.
https://localhost/PowerBIRestApiDemo

Step 3: Choose APIs to access

Select the APIs and the level of access your app needs.

Dataset APIs	Report and Dashboard APIs	Other APIs
<input type="checkbox"/> Read All Datasets	<input type="checkbox"/> Read All Dashboards (preview)	<input type="checkbox"/> Read All Groups
<input checked="" type="checkbox"/> Read and Write All Datasets	<input type="checkbox"/> Read All Reports (preview)	

Step 4: Register your app

Once you've set everything the way you want it, click the button below and we'll register your app. Your client ID and secret (for web apps only) will appear below. Be sure to copy the values into your app. By clicking the Register App button, you have accepted the [terms of use](#).

Register App

Client ID:
58883475-5508-438f-8bd7-9e16a573d70f



Important Application Constants

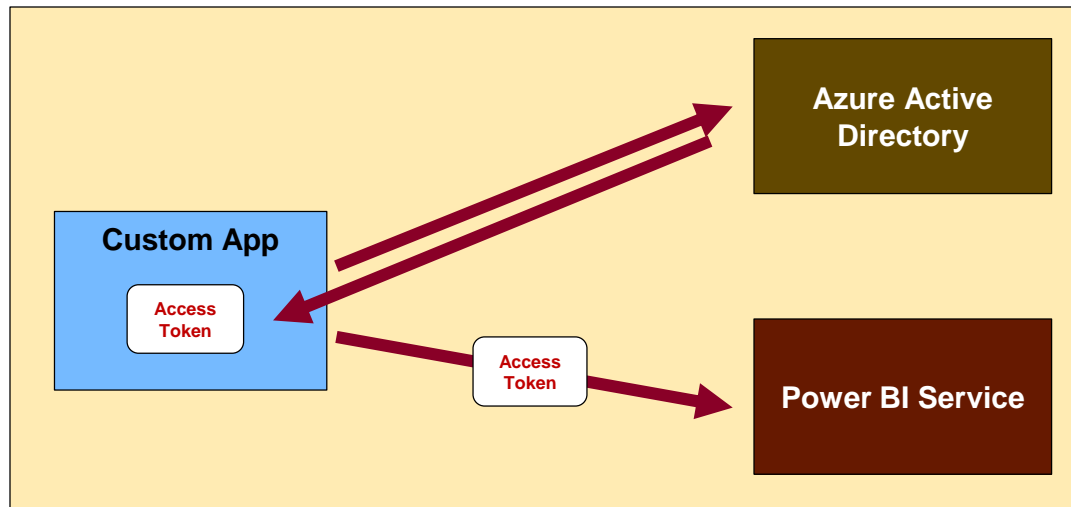
- Power BI REST API requires Client ID and several URLs
 - These values are tracked as application constants

```
class ProgramGlobalConstants {  
  
    public const string AzureAuthorizationEndpoint = "https://login.microsoftonline.com/common";  
    public const string PowerBiServiceResourceUri = "https://analysis.windows.net/powerbi/api";  
    public const string PowerBiServiceRootUrl = "https://api.powerbi.com/v1.0/myorg/";  
  
    public const string ClientID = "bc6b8f66-390b-4ad5-9dc6-9637f7f9841f";  
    public const string RedirectUri = "https://localhost/PowerBiRestApiDemo";  
  
    public const string DatasetName = "My Custom Dataset";  
  
}
```



Authenticating with Azure AD

- User must be authenticated against Azure AD
 - User authentication used to obtain access token
 - Can be accomplished with the Azure AD Authentication Library
 - Access token pass to Power BI REST API in call REST calls



Using ADAL to Retrieve an Access Token

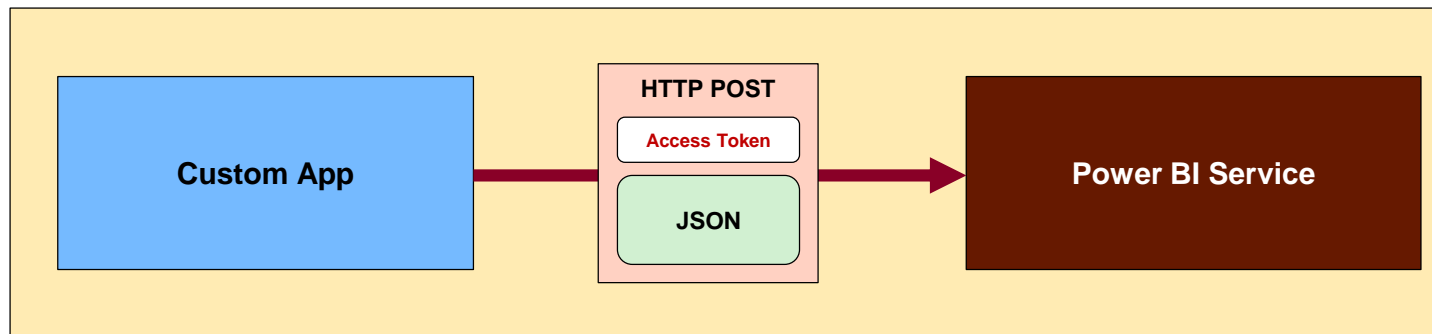
```
class PowerBiWorkspaceManager {  
  
    #region "Authentication Details"  
  
    protected string AccessToken = string.Empty;  
  
    protected void GetAccessToken() {  
  
        1 // create new authentication context  
        var authenticationContext = new AuthenticationContext(ProgramGlobalConstants.AzureAuthorizationEndpoint);  
  
        2 // use authentication context to trigger user sign-in and return access token  
        var userAuthnResult = authenticationContext.AcquireToken(ProgramGlobalConstants.PowerBiServiceResourceUri,  
                                                                ProgramGlobalConstants.ClientID,  
                                                                new Uri(ProgramGlobalConstants.RedirectUri),  
                                                                PromptBehavior.Auto);  
  
        3 // cache access token in AccessToken field  
        AccessToken = userAuthnResult.AccessToken;  
  
    }  
  
    #endregion  
}
```



Executing Power BI REST API Calls

- Generic helper methods designed to execute HTTP operations

```
private void ExecutePostRequest(string restUri, string postBody) {  
    1 // prepare REST call  
    HttpContent body = new StringContent(postBody);  
    body.Headers.ContentType = new MediaTypeWithQualityHeaderValue("application/json");  
    HttpClient client = new HttpClient();  
    client.DefaultRequestHeaders.Add("Accept", "application/json");  
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + AccessToken);  
    2 // execute REST call  
    HttpResponseMessage response = client.PostAsync(restUri, body).Result;  
}
```



Using JSON to Create a Custom Dataset

- Dataset created using JSON-formatted Table Schema

```
{ "name": "My Custom Dataset",  
  "tables": [  
    { "name": "Countries",  
      "columns": [  
        { "name": "Country", "dataType": "string" },  
        { "name": "Population", "dataType": "Int64" },  
        { "name": "Continent", "dataType": "string" }  
      ]  
    },  
    { "name": "States",  
      "columns": [  
        { "name": "State", "dataType": "string" },  
        { "name": "Abbreviation", "dataType": "string" },  
        { "name": "Founded", "dataType": "Int64" },  
        { "name": "SquareMiles", "dataType": "Int64" },  
        { "name": "Population", "dataType": "Int64" },  
        { "name": "PopulationDensity", "dataType": "Double" },  
        { "name": "CapitalCity", "dataType": "string" }  
      ]  
    }  
  ]  
}
```

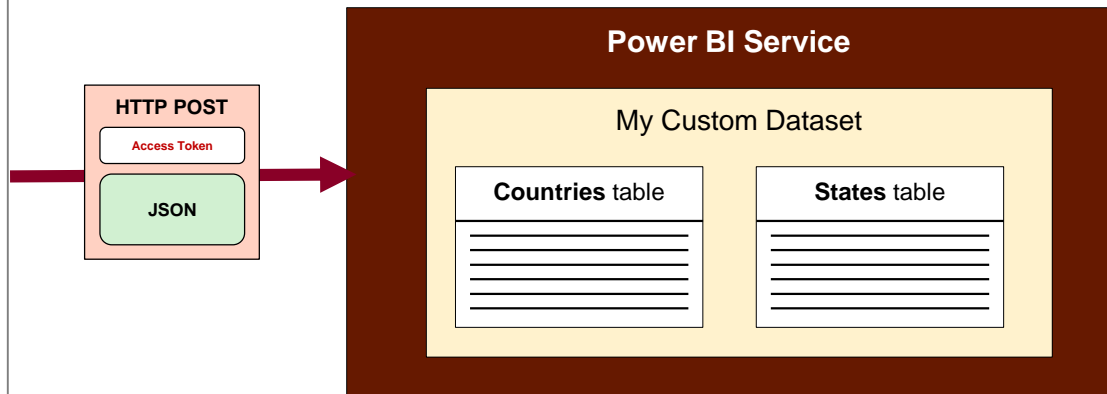


Creating a Custom Dataset

- Dataset created by executing HTTP POST operation
 - One-time operation done as application begins running

```
// prepare call to create new dataset
1 string restUrlDatasets = ProgramGlobalConstants.PowerBiServiceRootUrl + "datasets";
string jsonNewDataset = Properties.Resources.NewDataset_json;
// execute REST call to create new dataset
2 string json = ExecutePostRequest(restUrlDatasets, jsonNewDataset);
// retrieve Guid to track dataset ID
3 Dataset dataset = JsonConvert.DeserializeObject<Dataset>(json);
CustomDatasetId = dataset.id;
```

```
{ "name": "My Custom Dataset",
  "tables": [
    { "name": "Countries",
      "columns": [
        { "name": "Country", "dataType": "string" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "Continent", "dataType": "string" }
      ]
    },
    { "name": "States",
      "columns": [
        { "name": "State", "dataType": "string" },
        { "name": "Abbreviation", "dataType": "string" },
        { "name": "Founded", "dataType": "Int64" },
        { "name": "SquareMiles", "dataType": "Int64" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "PopulationDensity", "dataType": "Double" },
        { "name": "CapitalCity", "dataType": "string" }
      ]
    }
  ]
}
```



Designing C# Classes to Convert to JSON

- C# class can be created to facilitate reading & writing JSON
 - Newtonsoft.Json package contains classes for performing conversions

```
public class CountryRow {  
    public string Country { get; set; }  
    public int Population { get; set; }  
    public string Continent { get; set; }  
}  
  
class CountryTableRows {  
    public CountryRow[] rows { get; set; }  
}
```

```
class SampleData {  
  
    public static CountryTableRows GetCountries() {  
        CountryRow[] Countries = {  
            new CountryRow { Country="China", Population=1385566537, Continent="Asia" },  
            new CountryRow { Country="India", Population=1252139596, Continent="Asia" },  
            new CountryRow { Country="United States", Population=320050716, Continent="North America" },  
            new CountryRow { Country="Indonesia", Population=249865631, Continent="Asia" },  
            new CountryRow { Country="Brazil", Population=200361925, Continent="South America" },  
            new CountryRow { Country="Pakistan", Population=182142594, Continent="Asia" },  
            new CountryRow { Country="Nigeria", Population=173615345, Continent="Africa" },  
            new CountryRow { Country="Bangladesh", Population=156594962, Continent="Asia" },  
            new CountryRow { Country="Russia", Population=142833689, Continent="Asia" },  
            new CountryRow { Country="Japan", Population=127143577, Continent="Asia" },  
            "More countries"  
        };  
        return new CountryTableRows { rows = Countries };  
    }  
  
    public static StateTableRows GetStates() { ...  
}
```

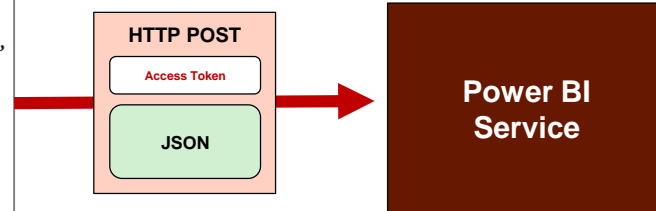


Adding Rows to a Table in a Dataset

- Executing POST to add rows to Countries table

```
public void AddCountryRows() {  
    string restUrlDatasets = ProgramGlobalConstants.PowerBiServiceRootUrl + "datasets/";  
    CountryTableRows countryRows = SampleData.GetCountries();  
    1 string jsonCountryRows = JsonConvert.SerializeObject(countryRows);  
    string restUrlCountryTableRows = string.Format("{0}/{1}/tables/Countries/rows", restUrlDatasets, CustomDatasetId);  
    2 string json = ExecutePostRequest(restUrlCountryTableRows, jsonCountryRows);  
}
```

```
{ "rows": [  
    { "Country": "China", "Population": 1385566537, "Continent": "Asia" },  
    { "Country": "India", "Population": 1252139596, "Continent": "Asia" },  
    { "Country": "United States", "Population": 320050716, "Continent": "North America" },  
    { "Country": "Indonesia", "Population": 249865631, "Continent": "Asia" },  
    { "Country": "Brazil", "Population": 200361925, "Continent": "South America" },  
    { "Country": "Pakistan", "Population": 182142594, "Continent": "Asia" },  
    { "Country": "Nigeria", "Population": 173615345, "Continent": "Africa" },  
    { "Country": "Bangladesh", "Population": 156594962, "Continent": "Asia" },  
    { "Country": "Russia", "Population": 142833689, "Continent": "Asia" },  
    { "Country": "Japan", "Population": 127143577, "Continent": "Asia" }  
]
```



Countries Table		
Country	Population	Continent
China	1385566537	Asia
India	1252139596	Asia
United States	320050716	North America
Indonesia	249865631	Asia
Brazil	200361925	South America
Pakistan	182142594	Asia
Nigeria	173615345	Africa
Bangladesh	156594962	Asia
Russia	142833689	Asia
Japan	127143577	Asia



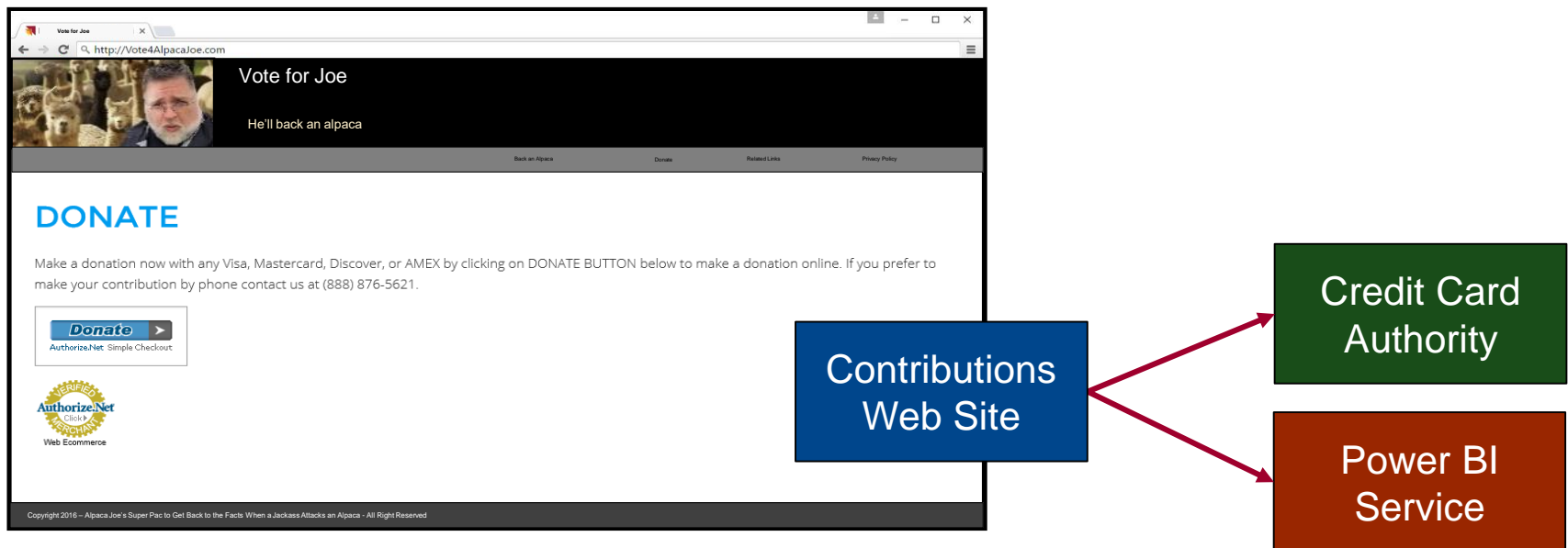
Agenda

- ✓ Getting Started with the Power BI REST API
- Creating a Real-time Dashboard
 - Creating an SPA for Power BI using Angular.js
 - Embedding Reports and Dashboard Tiles



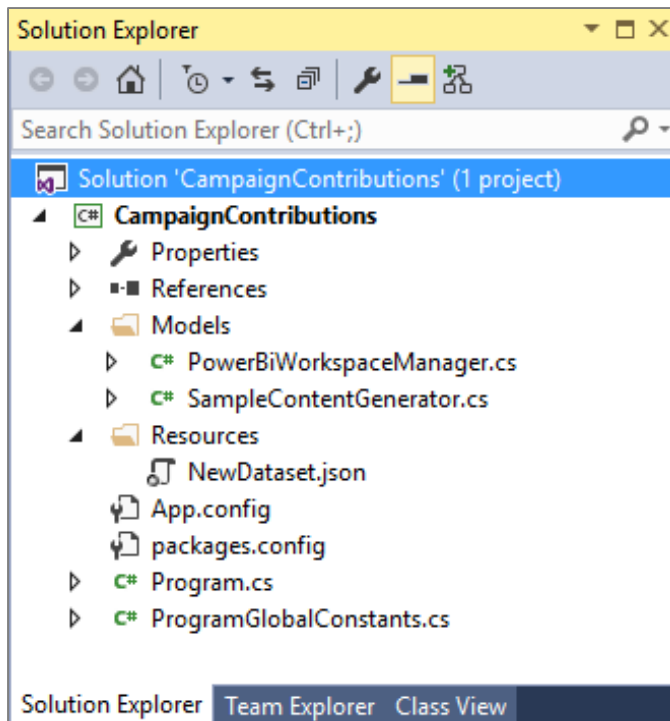
The Scenario Being Simulated

- Contribution website accepts contributions from donors
 - Website calls web service to process credit card transaction
 - Website calls to Power BI REST API to create & update dataset



CampaignContributions Demo App

- Console app similar to PowerBiRestApiDemo
 - Uses Newtonsoft.Json package to convert JSON data
 - Uses Azure ADAL.NET to get access tokens from Azure AD



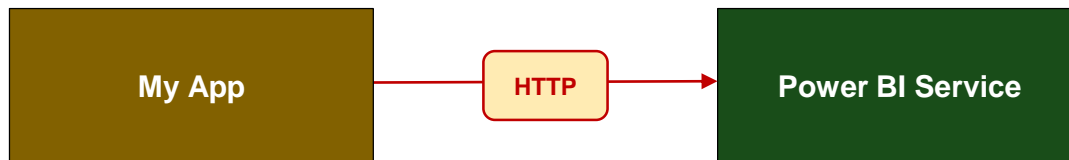
Contribution Row Data

ContributionID	Contributor	City	Zipcode	Gender	Time	Amount
1	Travis Solomon	Lutz, FL	33559	Female	12:03:00	50.00
2	Rudolph Harris	Tarpon Springs, FL	34688	Female	12:03:00	100.00
3	Josh Franco	Davis Island, FL	33606	Male	12:03:00	250.00
4	Tim Duffy	Largo, FL	33774	Female	12:03:00	200.00
5	Napoleon Frye	Lutz, FL	33558	Male	12:03:30	50.00
6	Rodrigo Hart	Largo, FL	33774	Female	12:03:30	100.00
7	Will Levine	Tarpon Springs, FL	34688	Female	12:03:30	75.00
8	Eddie McCullough	Lutz, FL	33558	Male	12:03:30	100.00
9	Kirk Herrera	Lutz, FL	33558	Male	12:03:30	150.00
10	Antonia Bridges	Lutz, FL	33558	Female	12:03:30	100.00
11	Preston Cote	Davis Island, FL	33606	Female	12:03:30	1,000.00
12	Lonnie McCarty	Largo, FL	33774	Female	12:03:30	50.00
13	Humberto Parsons	Lutz, FL	33558	Female	12:03:30	50.00
14	Abel Perkins	Odessa, FL	33556	Female	12:03:30	1,000.00
15	Ernesto McLaughlin	Odessa, FL	33556	Female	12:03:30	150.00
16	Elbert Kinney	Macdill Air Force Base, Port Tampa, FL	33621	Male	12:03:30	50.00
17	Coleman Petersen	Lutz, FL	33558	Male	12:03:30	125.00
18	Nicky Roberson	Odessa, FL	33556	Female	12:03:30	100.00
19	Fritz Parrish	Lutz, FL	33558	Female	12:03:30	50.00
20	Freddie Sparks	Largo, FL	33774	Male	12:03:30	50.00

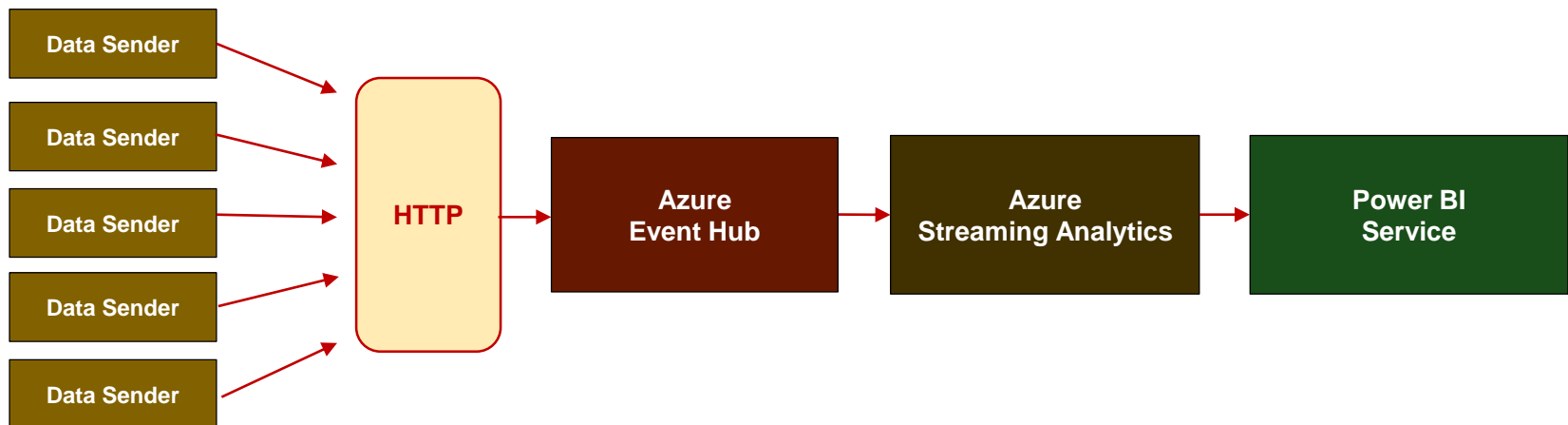


Scaling a Real-time Dashboard

- Low velocity data scenario



- High velocity data scenario



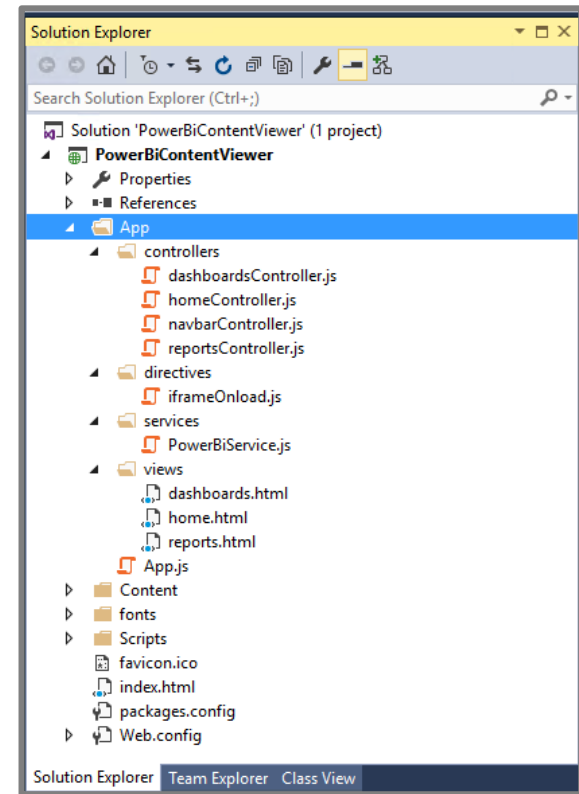
Agenda

- ✓ Getting Started with the Power BI REST API
- ✓ Creating a Real-time Dashboard
- Creating an SPA for Power BI using Angular.js
 - Embedding Reports and Dashboard Tiles



The PowerBiContentViewer Demo

- Demonstrates SPA application
 - Built using client-side code (HTML, CSS and JavaScript)
 - SPA built using Angular-JS
 - Authentication using ADAL-JS



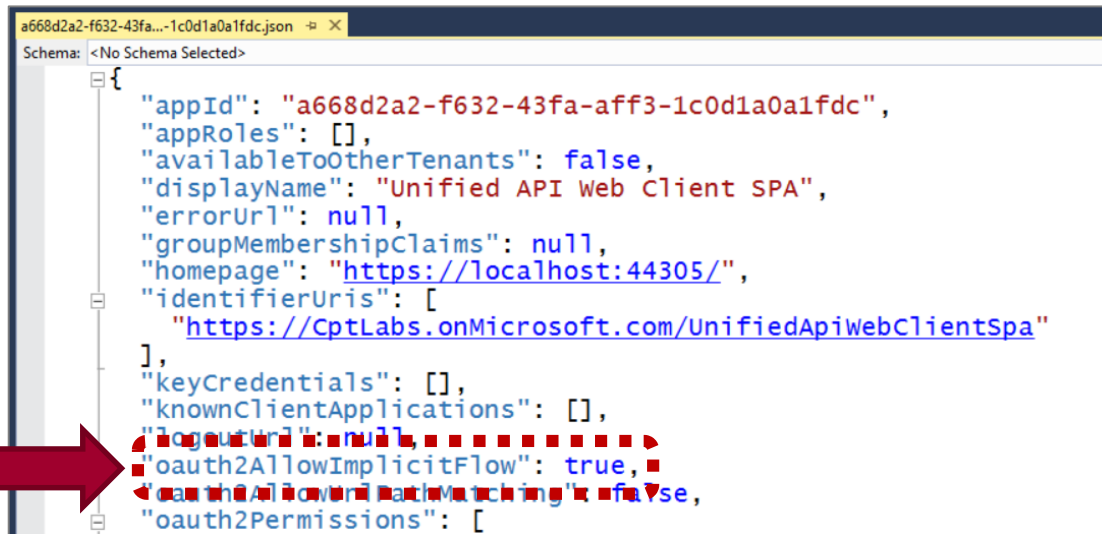
Understanding Implicit Grant Flow

- Implicit Grant Flow
 - Used when client cannot keep secrets (public client)
 - Used with SPAs built using JavaScript and AngularJS
 - Less secure than Authentication Code Grant
- How does it work?
 - Client authorizes user with AD authorization endpoint
 - AD returns access token directly to SPA in browser
 - Authentication flow does not involve authorization code



Configuring Implicit Flow in Azure AD

- Requires configuring AD application in Azure AD
 - Download manifest from Azure AD
 - Update **oauth2AllowImplicitFlow** setting equal to **true**
 - Upload manifest to Azure AD to save changes

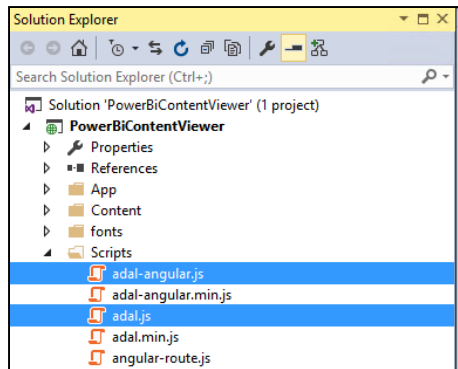


```
a668d2a2-f632-43fa...-1c0d1a0a1fdc.json
Schema: <No Schema Selected>
{
  "appId": "a668d2a2-f632-43fa-aff3-1c0d1a0a1fdc",
  "appRoles": [],
  "availableToOtherTenants": false,
  "displayName": "Unified API Web Client SPA",
  "errorUrl": null,
  "groupMembershipClaims": null,
  "homepage": "https://localhost:44305/",
  "identifierUris": [
    "https://CptLabs.onMicrosoft.com/UnifiedApiWebClientSpa"
  ],
  "keyCredentials": [],
  "knownClientApplications": [],
  "logoutUrl": null,
  "oauth2AllowImplicitFlow": true,
  "oauth2AllowIdTokenMatching": false,
  "oauth2Permissions": [
```



Downloading the ADAL-JS Library

- Developing with ADAL-JS involves to two library files
 - **adal.js** – core ADAL-JS library
 - **adal-angular.js** –integration of ADAL-JS with AngularJS



- Library files downloadable from GitHub Repository
 - <https://github.com/AzureAD/azure-activedirectory-library-for-js>



Initializing ADAL-JS Settings

```
function initializeADALSettings($httpProvider, adalProvider) {  
  
    var endpoints = {  
        "https://graph.windows.net/": "https://graph.windows.net/",  
        "https://api.powerbi.com/v1.0/": "https://analysis.windows.net/powerbi/api",  
        "https://api.powerbi.com/beta/": "https://analysis.windows.net/powerbi/api",  
    };  
  
    var adalProviderSettings = {  
        tenant: 'common',  
        clientId: client_id,  
        extraQueryParameter: 'nux=1',  
        endpoints: endpoints,  
        cacheLocation: 'localStorage' // enable this for IE, as sessionStorage does not work for localhost.  
    };  
  
    adalProvider.init(adalProviderSettings, $httpProvider);  
  
}
```



Making Secure Calls to Custom Web Services

- adal-angular.js adds interceptors to \$http service
 - adal detects when calls are made to secure endpoints
 - adal acquires & caches access tokens behind scenes
 - adal attaches access token to Authorization header

```
var apiRoot = "https://api.powerbi.com/v1.0/";

service.getDatasets = function () {
  var restUrl = apiRoot + "myOrg/Datasets/";
  return $http.get(restUrl);
};
```

```
var apiRootBeta = "https://api.powerbi.com/beta/";

service.getReports = function () {
  var restUrl = apiRootBeta + "myOrg/Reports/";
  return $http.get(restUrl);
};

service.getDashboards = function () {
  var restUrl = apiRootBeta + "myOrg/Dashboards/";
  return $http.get(restUrl);
};

service.getDashboardTiles = function (dashboardId) {
  var restUrl = apiRootBeta + "myOrg/Dashboards/" + dashboardId + "/tiles/";
  return $http.get(restUrl);
};
```



Inspecting Authenticated User Claims

```
var app = angular.module('UnifiedApiSpa')
app.controller('userTokenController', ['$scope', 'adalAuthenticationService', userTokenController]);
function userTokenController($scope, adalAuthenticationService) {
    // create array to track claims for logged-on user
    $scope.claims = [];

    // add claims for id_token into claims array
    for (var property in adalAuthenticationService.userInfo.profile) {
        if (adalAuthenticationService.userInfo.profile.hasOwnProperty(property)) {
            $scope.claims.push({
                key: property,
                value: adalAuthenticationService.userInfo.profile[property],
            });
        }
    }
}
```

OpenId Connect Token Claims

Claim	Value
aud	a668d2a2-f632-43fa-aff3-1c0d1a0a1fdc
iss	https://sts.windows.net/572f112d-3e6c-4151-877c-bc2bcade71b2/
iat	1443647920
nbf	1443647920
exp	1443651820
ver	1.0
tid	572f112d-3e6c-4151-877c-bc2bcade71b2
oid	ff734d8b-eebc-4f9c-8ee5-ba2a588defd9
upn	Student@CptLabs.onmicrosoft.com
sub	WG3k5VRbV9z4ih7wcojYLSj0VFok6VquvDheZUYahEc
given_name	CPT
family_name	Student
name	CPT Student



Agenda

- ✓ Getting Started with the Power BI REST API
- ✓ Creating a Real-time Dashboard
- ✓ Creating an SPA for Power BI using Angular.js
- Embedding Reports and Dashboard Tiles





DEMO

Embedding Power BI Reports

Summary

- ✓ Getting Started with the Power BI REST API
- ✓ Creating a Real-time Dashboard
- ✓ Creating an SPA for Power BI using Angular.js
- ✓ Embedding Reports and Dashboard Tiles

