

INTEGRATE POWER BI WITH WPF DESKTOP APPLICATIONS

Maria Dobрева, Nikolay Pavlov, Asen Rahnev

Abstract: Power BI is cloud based service that provides a set of tools for making business analyzes. In this paper we describe how we perform embedding interactive Power BI resources – reports, tiles, dashboards into WPF business applications. This new functionality is implemented as part of Framework for Distributed Business Applications (FDBA). Direct, secure, and quick access to interactive reports and data visualizations is provided to the users of applications developed through FDBA.

Keywords: data analysis, FDBA, Power BI, Power BI Embedded

1. About Power BI

What is Power BI?

Power BI is a suite cloud-based business analytics tools that is used to analyze data and share insights in the form of reports and dashboards. Processes in Power BI starts with connecting and loading data. The service enables to connect to a wide variety of data sources more than 70 data providers. Once the data has been loaded, data transformations are applied according to the needs of the report and dashboard being developed. This step is used to remove errors and redundant data, correct formatting, and prepare data for further analysis by organizing them into suitable normalized forms, make relationships, calculations, measures, hierarchies that can be used to find business insights and so on. Once a data model is ready, interactive reports can be created via whole range of visualizations provided by default from simple bar charts to pie charts to maps, and even more offerings like waterfalls, funnels, gauges, and more. These visuals help present data in a way that provides context and insights. Filters can be applied on the reports so that relevant data is surfaced for users interested in analyzing the data. When report is ready it can be exported or published in Power BI service. Power BI provides numerous options to achieve sharing BI content with someone

or within a group. One can publish the results public to a website or blog, share it with other users within the organization, and share it with user groups. The most common way is publishing the reports in Power BI service from where another users or services can access it. This way users can access it from any device reports via web portal or mobile applications.

Common flow of work in Power BI

Power BI consists of a Windows desktop application called **Power BI Desktop**, an online SaaS (Software as a Service) service called the **Power BI service**, and **Power BI mobile apps** for Windows, iOS, and Android devices [1]. Power BI Desktop is used to connect to data sources and to build the reports. Then these reports are published from Power BI Desktop to the Power BI service from where end users can access it.

2. Problem

Clients wants to make decisions based on operational data produced by the business desktop applications. The Framework for Distributed Business Applications – FDBA is a powerful platform for developing state-of-art enterprise resource planning (ERP) and customer relation management (CRM) software products [4, 6, 7, 8]. No such system can be complete without an adequate tool for business intelligence.

The FDBA framework provide custom reporting engine which is suitable for not very complex reports. The custom engine is integrated with FDBA applications and users are able to produce, store and view these reports within the applications. These reports are generated as Microsoft Word documents and does not support rich graphics or custom visualizations and cannot compute complex calculations.

At some point of the time the need of the more sophisticated and custom BI solutions. We decided to use Power BI as reporting tool. Reports are created via Power BI Desktop, then these reports are published to the Power BI service and users have access to the reports from the web portal. However, users prefer to access these reports from the FDBA application. It would be more convenient for them to access it without explicitly switching from working application to the browser and enter a separate different credentials to login to Power BI Service. Another benefit is making analysis directly on the underlying data from the application.

3. Implementation

For user convenience we decided to embed Power BI content into FDBA business applications. This allows users to see the Power BI reports, tiles, and

dashboard in the FDBA business application with which they work. Users can do this from a separate screen which shows all available Power BI resources developed for the application needs and published in Power BI Service.

The Power BI Service and the Power BI Embedded service in Azure have APIs for embedding analytical content (reports, dashboards, data visualizations) to applications [2]. From an architectural perspective, Power BI embedding involves adding an iframe to a web page and configuring the iframe with a URL and a security token to load a Power BI report or dashboard directly from the Power BI Service.

The process of embedding Power BI in native application involves the following steps:

1. The application is registered in Azure AD
2. Application authenticates against Azure AD to obtain an access token to call the Power BI Service.
3. Call the Power BI Service API to retrieve embedding data about a specific Power BI resource.
4. Pass the embedding data and a security token to client- side code running in the browser

3.1. Register an application in Azure Active Directory

In Azure AD, an application object describes an application as an abstract entity. Developers work with applications. At deployment time, Azure AD uses a given application object as a blueprint to create a service principal, which represents a concrete instance of an application within a directory or tenant. It's the service principal that defines what the app can actually do in a specific target directory, who can use it, what resources it has access to, and so on [3].

There are two ways to register an application in AAD – in Azure Portal or use embedding setup tool available here <https://dev.powerbi.com/apps>.

We registered each FDBA application from the Azure portal. To register application, an appropriate name and application type must be specified. There are two options for “Application type”: Native and Web app/API. FDBA applications run in Windows OS environment so applications are registered as Native one.

Create □ ×

* Name ⓘ

Application type ⓘ

* Redirect URI ⓘ

To allow embedding it is important to setup required permissions for Windows Azure Active Directory API and Power BI Service API.

3.2. Get Authentication access token

To be able to call Power BI REST API an authentication access token from Azure Active Directory is needed. Our implementation user access token with a master account credentials – using Power BI Pro account credentials (username, password) and Azure app. Application authenticates against Azure AD with those credentials controlled by the FDBA application.

```
private static string resourceUri =
    "https://analysis.windows.net/powerbi/api";
private static string authorityUri =
    "https://login.windows.net/common/oauth2/
    authorize";
private static AuthenticationContext authContext = new
    AuthenticationContext(authorityUri, new TokenCache());
private string accessToken;
var clientId = DecodeValue(ClientId, ClientIdValueType); //
    application Identifier

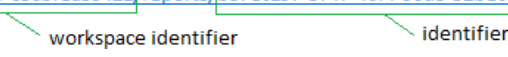
var uc = new UserPasswordCredential(powerBIUserName,
    powerBIPassword);
try
{
    authenticationResult =
    authContext.AcquireTokenAsync(resourceUri, clientId, uc).Result;
}
catch (Exception ex)
{
    ...
    return;
}

if (authenticationResult != null)
    this.accessToken = authenticationResult.AccessToken;
```

3.3. The application calls the Power BI Service API to retrieve embedding data about a specific Power BI resource

To get data for visual we send web request (GET) providing the url on which the Power BI content is available and authorize via generated AD access token. The url contains information about the resource and also for the workspace placed. The example below shows a Power BI report url.

<https://app.powerbi.com/groups/166fe039-7786-4051-ada4-65cbfea59421/reports/b87ec197-3747-46f4-90d3-52be05d7cae3>



Depending on the embedding type the url is composed by the following way:

- Report

`https://api.powerbi.com/v1.0/myorg/groups/{groupId}/reports/{reportId}`

- Dashboard

`https://api.powerbi.com/v1.0/myorg/groups/{groupId}/dashboards/{dashboardId}`

- Tile

`https://api.powerbi.com/v1.0/myorg/groups/{groupId}/dashboards/{tileDashboardId}/tiles/{tileId}`

```
var request = WebRequest.Create(apiUrl) as HttpWebRequest;
request.KeepAlive = true;
request.Method = "GET";
request.ContentLength = 0;
request.ContentType = "application/json";

request.Headers.Add("Authorization", $"Bearer {accessToken}");

using (HttpWebResponse httpResponse = request.GetResponse() as
System.Net.HttpWebResponse)
{
    using (StreamReader reader = new
System.IO.StreamReader(httpResponse.GetResponseStream()))
    {
        string responseContent = reader.ReadToEnd();
        var responseJson =
JsonConvert.DeserializeObject<dynamic>(responseContent);
        displayName = responseJson[nameElement];
        embedUrl = responseJson["embedUrl"];
    }
}
```

On success the response returns an **embedUrl**

3.4. The application passes the embedding data and a security token to client- side code running in the browser

Client-side code in the browser calls the Power BI JavaScript API to embed the Power BI resource.

A WebBrowser user control is used to load and render Power BI service content in WPF application page. This control provides ObjectForScripting property which allows JavaScript functions in an HTML page to call methods and properties of an instance class passed to this property. The Web Browser control

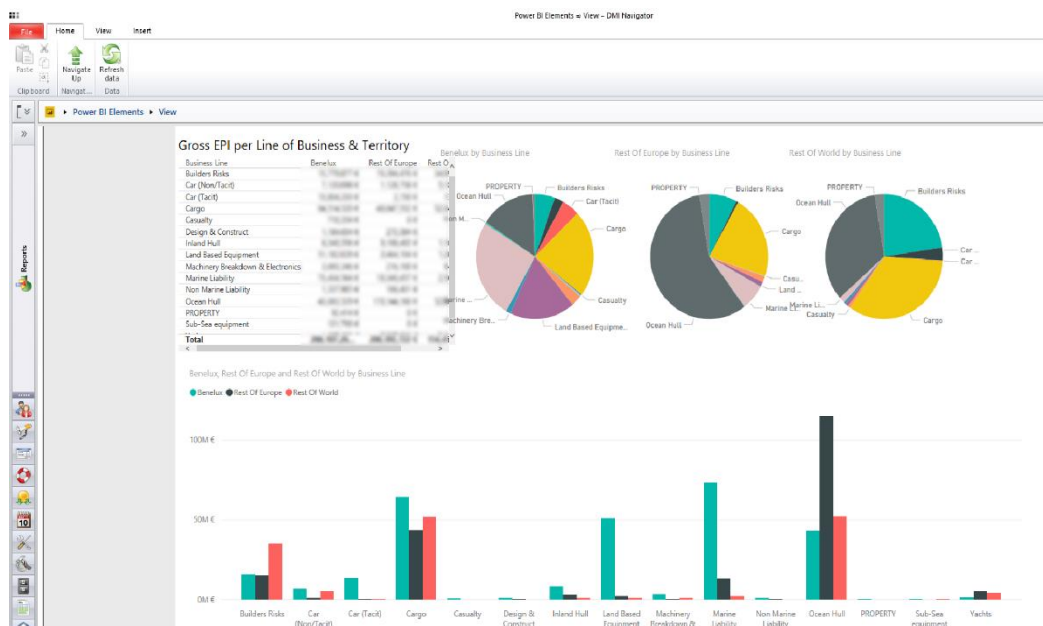
navigates to a locally passed html file that pulls the required JavaScript and has a DIV container to render the Power BI Object into.

```
<WebBrowser x:Name="webBrowser" Grid.Row="0"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" />
public void PBIEmbedded_Invoke()
{
    // Calling JavaScript code to load the report and passing
    parameters from C# code to JavaScriptCode
    var parameters = new object[] { embedUrl, accessToken,
    itemId, embedItemTypeJS, (int)PowerBIAccessTokenType.AD,
    DashboardId };
    webBrowser.InvokeScript("LoadEmbeddedObject", parameters);
}
private void EmbedReport()
{
    webBrowser.ObjectForScripting = new
AddJavascriptObjects(null, PBIEmbedded_Invoke);
    var html = ExtractFiles();
    webBrowser.Navigate(html);
}
}
```

From the code above to load the Power BI content the following parameters are passed:

- **embedUrl** – this is the embed url (retrieved on step 2)
- **accessToken** – the access token from AAD (generated on step 1)
- **itemId** – the unique identifier of the resource
- **embedItemTypeJS** – one of the following: report, dashboard, tile

The screenshot below shows Power BI report embedded into FDBA application.



4. Conclusion

This paper has described the functional requirements and implementation of embedding Power BI resources in FDBA application as a standard functionality to the Framework for Distributed Business Applications. The approach we have chosen provides users with an easily accessible way to perform and view complex data analysis within the native application.

In future, we will investigate methods to allow users to execute reports under selection of data.

Acknowledgements

This paper “Integrate Power BI with WPF Desktop applications” is partially supported by project FP17-FMI-008 of the Scientific Research Fund of Plovdiv University “Paisii Hilendarski”, Bulgari.

References

- [1] What is Power BI?, <https://docs.microsoft.com/en-us/power-bi/power-bi-overview>, retrieved on January 3, 2019
- [2] Embedded analytics with Power BI, <https://docs.microsoft.com/en-us/power-bi/developer/embedding>, retrieved on January 3, 2019
- [3] What is authentication?, <https://docs.microsoft.com/en-us/azure/active-directory/develop/authentication-scenarios#application-model>, retrieved on January 10, 2019
- [4] Pavlov, N., *Object-Oriented Framework for Development of Distributed Business Applications*, Ph.D. Thesis, Plovdiv University, Plovdiv, Bulgaria, 2011, (in Bulgarian).
- [5] Rob Reilly, Integrate Power BI Dashboards, Reports and Tiles into a WPF Application, <https://devblogs.microsoft.com/premier-developer/integrate-power-bi-dashboards-reports-and-tiles-into-a-wpf-application/> retrieved November 10, 2018
- [6] Dobreva, M., N. Pavlov, A. Rahnev, User Authentication Via Active Directory in FDBA, Scientific Conference „*Innovative Software Tools and Technologies with Applications in Research in Mathematics, Informatics and Pedagogy of Education*”, 23–24 November 2017, Pamporovo, Bulgaria, pp. 65–72, ISBN: 978-619-202-343-0.
- [7] Dobreva, M., Pavlov N., Security Policies in a Framework for Distributed Business Applications, *Doctoral Conference in Mathematics and Informatics MIDOC 2015*, October 15–18, 2015, Sofia, Bulgaria, Proceedings, pp. 36–43, ISBN: 978-954-07-4186-4.

- [8] Pavlov, N., Dobрева M., Pivot Reporting Tool, Сборник с доклади на научна конференция „Иновационни ИКТ: Изследвания, разработка и приложения в бизнеса и обучението“, гр. Хисар, 11–12 ноември 2015 г., pp. 21–30, ISBN: 978-954-8852-56-7.

Faculty of Mathematics and Informatics

Plovdiv University “Paisii Hilendarski”

236 Bulgaria Blvd, Plovdiv 4003, Bulgaria

e-mails: maria.d.dobрева@gmail.com, nikolayp@uni-plovdiv.bg,
asen@uni-plovdiv.bg

ИНТЕГРАЦИЯ НА POWER BI В WPF НАСТОЛНИ ПРИЛОЖЕНИЯ

Мария Добрева, Николай Павлов, Асен Рахнев

Резюме: Power BI е облачна услуга, която предоставя инструменти за изработка на бизнес анализи. В тази статия сме описали как сме реализирали вграждане на предварително изготвени Power BI отчети в настолни бизнес WPF приложения. Добавена е нова функционалност към рамката за разпределени приложения (FDBA) – интеграция с Power BI. На потребителите се предоставя директен, защитен и бърз достъп до интерактивни отчети и визуализации на данни в приложения разработени чрез FDBA.

Ключови думи: анализ на данни, FDBA, Power BI, Power BI Embedded