DB Owner

WP8 DBOwnerSDK 使用文档

V1.0.1

更多资源请访问 DBOwner 知识库

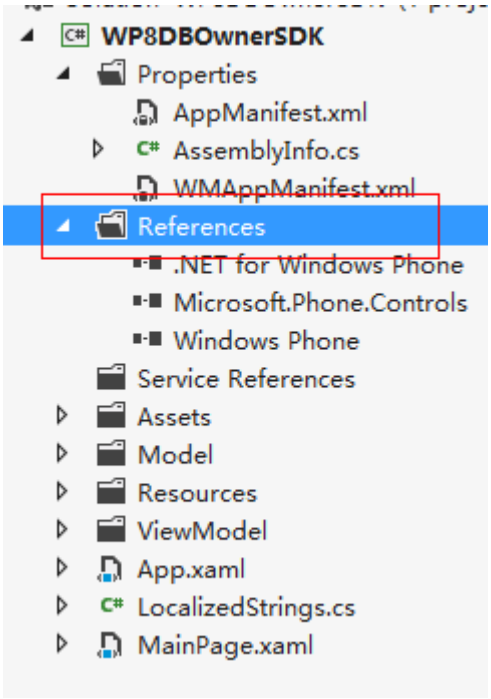
http://wiki.dbowner.com

一、 环境设置

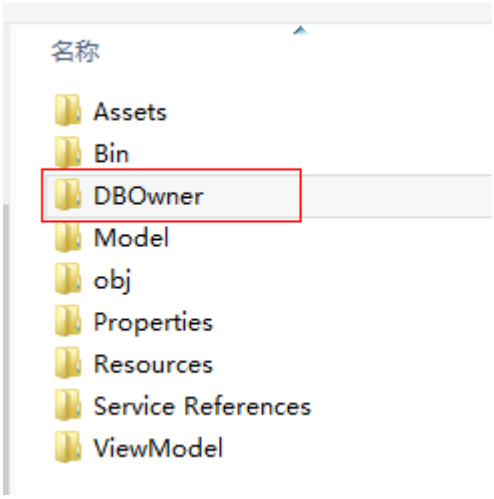
1、 添加 DBOwnerLib 类库

（1）首先下载 Demo 样例，然后新建项目工程文件

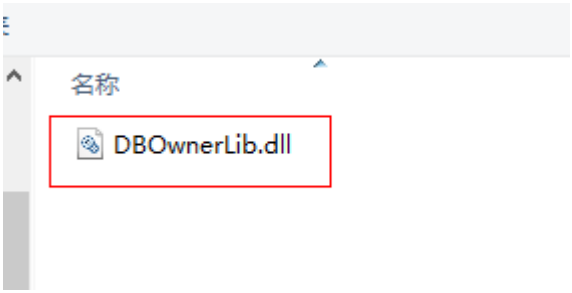
（2）将 DBOwnerLib 引入项目中，右击项目的 References->Add Reference...->Brower...



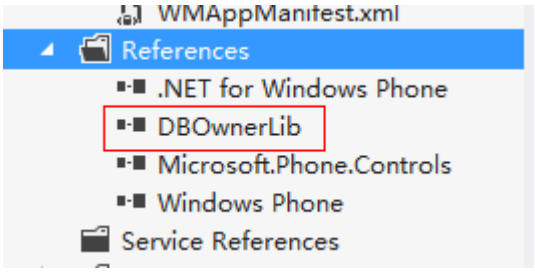
找到下载的文件 DBOwner



加载打开文件夹里面的 DBOwnerLib.dll



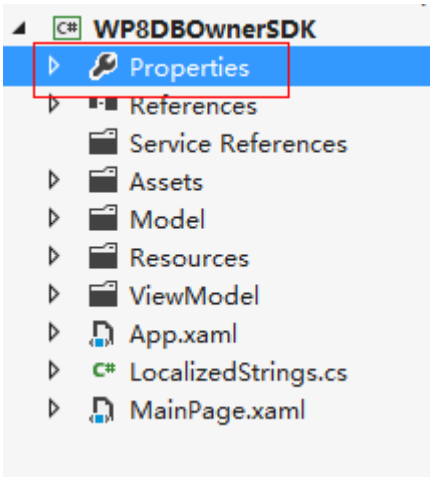
引入文件之后，如下图所示



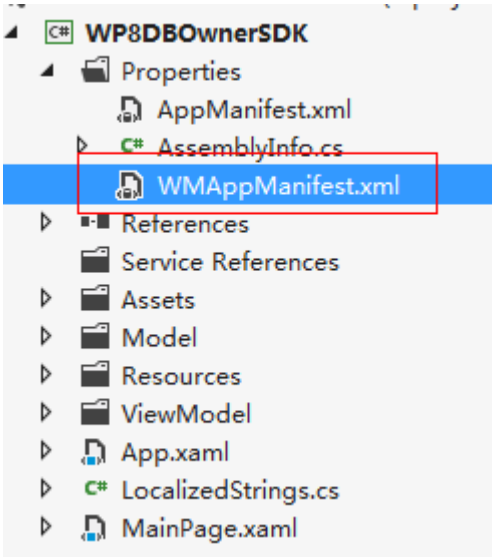
2、 设置 Capabilities

添加项目所需要的能力，以便使用 DBOwnerSDK

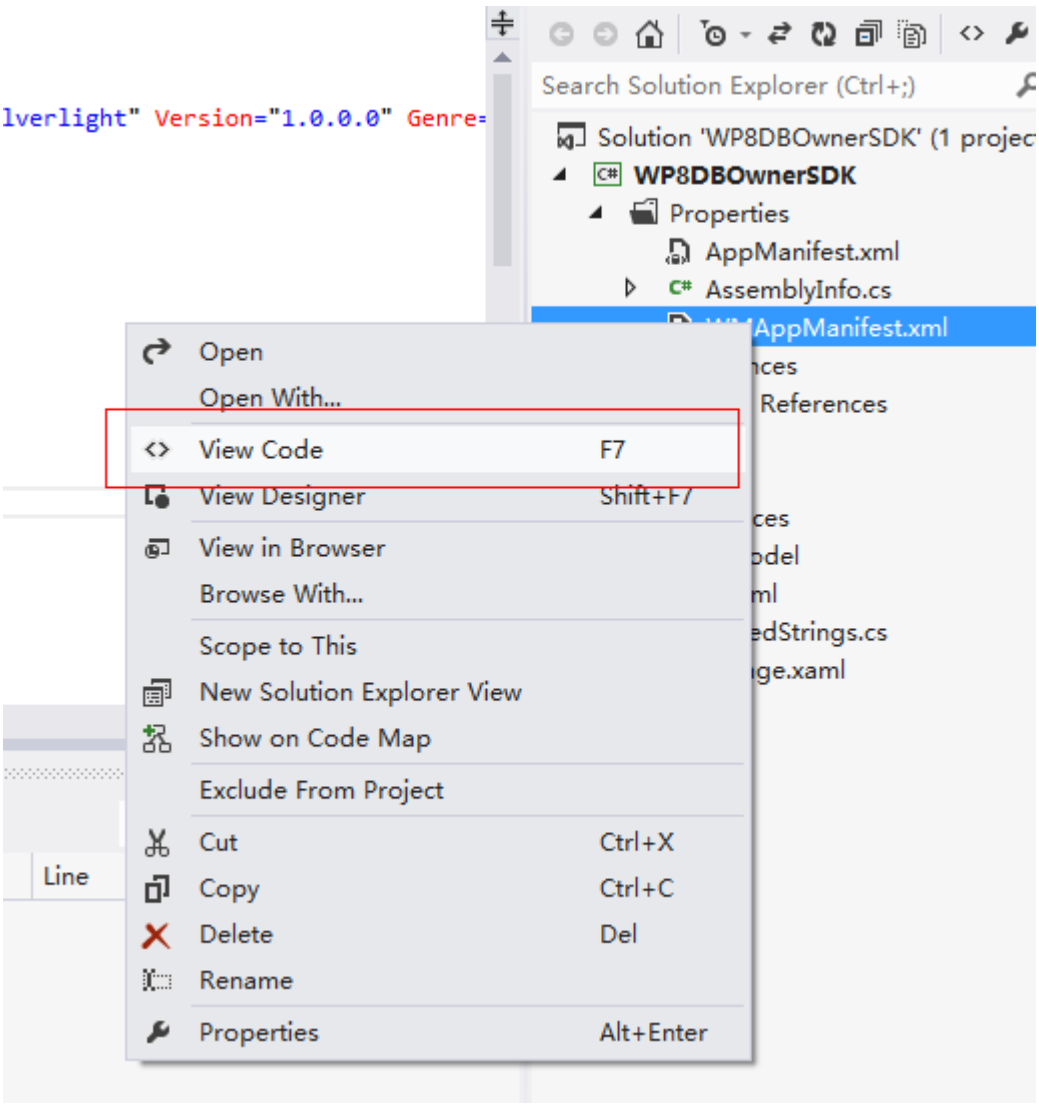
首先找到项目文件 Properites



找到 WMAppManifest.xml



右击 WMAppManifest.xml，然后选择 View Code



打开文件，找到 Capabilities，添加

```
<Capability Name="ID_CAP_IDENTITY_DEVICE" />
<Capability Name="ID_CAP_IDENTITY_USER" />
<Capability Name="ID_CAP_LOCATION" />
<Capability Name="ID_CAP_PUSH_NOTIFICATION"/>
<Capability Name="ID_CAP_PHONEDIALER"/>
```

五个能力

App.xaml.cs WMAppManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Deployment xmlns="http://schemas.microsoft.com/windowsphone/2012/deploy
  <DefaultLanguage xmlns="" code="en-US" />
  <App xmlns="" ProductID="{91151110-5cfd-417e-8034-381214475c4e}" Title
    <IconPath IsRelative="true" IsResource="false">Assets\ApplicationIcc
    <Capabilities>
      <Capability Name="ID_CAP_NETWORKING" />
      <Capability Name="ID_CAP_MEDIALIB_AUDIO" />
      <Capability Name="ID_CAP_MEDIALIB_PLAYBACK" />
      <Capability Name="ID_CAP_SENSORS" />
      <Capability Name="ID_CAP_WEBBROWSERCOMPONENT" />
      <Capability Name="ID_CAP_IDENTITY_DEVICE" />
      <Capability Name="ID_CAP_IDENTITY_USER" />
      <Capability Name="ID_CAP_LOCATION" />
      <Capability Name="ID_CAP_PUSH_NOTIFICATION"/>
      <Capability Name="ID_CAP_PHONEDIALER"/>
    </Capabilities>
    <Tasks>
      <DefaultTask Name="_default" NavigationPage="MainPage.xaml" />
    </Tasks>
```

二、 修改 App.xaml.cs

1、 引入类库 DBOwnerLib

```
using DBOwnerLib;
```

2、 在 App 类中声明类库对象

```
//声明参数
public static DBOwnerSDK dbOwnerSDK;
```

3、 在 InitializePhoneApplication 方法中添加以下方法

```
//引入DBOwnerSDK
dbOwnerSDK = new DBOwnerSDK("client_id", "client_secret", "ADCode", "PublicKey", "PublicIV",
"ChannelID");
```

参数说明：

client_id：应用 ID
client_secret：应用密匙
ADCode：广告 code
PublicKey：预留值
PublicIV：预留值
ChannelID：来源平台

4、 在 Application_Launching、Application_Activated、Application_Deactivated、Application_Closing 四个方法中分别添加 DBOwnerLaunching 、DBOwnerActivated、DBOwnerDeactivated、DBOwnerClosing 四个方法，如下面代码所示

```
// Code to execute when the application is launching (eg, from Start)
// This code will not execute when the application is reactivated
private void Application_Launching(object sender, LaunchingEventArgs e)
{
    dbOwnerSDK.DBOwnerLaunching();
}

// Code to execute when the application is activated (brought to foreground)
// This code will not execute when the application is first launched
private void Application_Activated(object sender, ActivatedEventArgs e)
{
    dbOwnerSDK.DBOwnerActivated();
}

// Code to execute when the application is deactivated (sent to background)
// This code will not execute when the application is closing
private void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
    dbOwnerSDK.DBOwnerDeactivated();
}

// Code to execute when the application is closing (eg, user hit Back)
// This code will not execute when the application is deactivated
private void Application_Closing(object sender, ClosingEventArgs e)
{
    dbOwnerSDK.DBOwnerClosing();
}
```

5、 在方法 Application_UnhandledException 中添加 DBOwnerListenError 方法，如下面代码

```
        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (Debugger.IsAttached)
            {
                dbOwnerSDK.DBOwnerListenError(e);
                //e.Handled = true;

                // An unhandled exception has occurred; break into the debugger

                Debugger.Break();
            }
        }
```

三、 实现 Auth 登陆验证

Auth 登陆支持新浪微博、腾讯 QQ、等多大 16 个合作网站的账号，使用标准 Auth2.0 协议。DBOwnerAuth 登陆时候需要传入用于登陆的控件显示的父 UIViewController，调用方法如下：

1、 引入类库 DBOwnerLib

```
using DBOwnerLib;
```

2、 在类中声明类库对象

```
//声明参数
DBOwnerSDK dbOwnerSDK = App.dbOwnerSDK;
```

3、 在需要登录的地方添加以下代码

```
dbOwnerSDK.DBOAuthLogin(this, DBOwner_LoginCompletedHandler);
```

其中 DBOwner_LoginCompletedHandler 是登录成功之后的回调函数，可以像下面代码处理

```
/// <summary>
/// 登录成功后处理
/// </summary>
public void DBOwner_LoginCompletedHandler(object sender, AuthLoginCompletedEventArgs e)
{
    if (e.Error == null)
    {
```

```
                NavigationService.Navigate(new Uri("/ViewModel/MainView.xaml",
UriKind.Relative));
            }
            else
            {
                MessageBox.Show(e.Error.Message);
            }
        }
```

4、 DBOwner 中关于登录需要说明的方法

IsAccessTokenValid：是否已经授权过
GetAccessToken：返回 access_token 值
GetRefreshToken：返回 refresh_token 值
GetUserID：返回 user_id 值
其他获取用户相关方法，请参考 demo 中的例子

四、 实现 View 统计

1、 引入 DBOwnerLib

```
using DBOwnerLib;
```

2、 在类中声明类库对象

```
//声明参数
DBOwnerSDK dbOwnerSDK = App.dbOwnerSDK;
```

3、 在类的构造方法中加入统计代码

```
dbOwnerSDK.DBOwnerPushViewPage("ViewCode");
```

其中 ViewCode 即代表当前页面 code 值

五、 实现广告投放

DBOwnerAD 支持全屏、插屏、横条 (Top、Bottom、Left、Right) 多种广告类型，并支持自动转屏重置。

1、 引入 DBOwnerLib

```
using DBOwnerLib;
```

2、 在类中声明类库对象

```
//声明参数
DBOwnerSDK dbOwnerSDK = App.dbOwnerSDK;
```

3、 广告使用方法

```
全屏广告：dbOwnerSDK.addAdFullScreen(this);
插屏广告：dbOwnerSDK.addAdInsertScreen(this);
顶部横幅广告：dbOwnerSDK.addAdBanner(this, "up");
底部横幅广告：dbOwnerSDK.addAdBanner(this, "down");
```

六、 数据 API

DBOwner 提供了数据存储 ,您可以通过 api 来管理您保存在 DBOwner 上面的数据。DBOwner 提供了增删改查多个接口，将按照您在开发者平台上定义的数据接口进行存储数据，并且您完全享有自己所拥有访问 APP 下面的所有数据使用权。

DBOwner 提供的接口将按照 NoSql 的 bjson 格式规则，在此基础上进行整合，请求操作可以参照以下说明

1、 查询数据方法

```
DBOwnerDataFind(string appid, string objName, string findStr, string fieldStr, string sortStr, int limitStr, string pageStr, string userStr, string idStr, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID
objName：数据对象
findStr：查询条件（json 格式）
fieldStr：请求参数（json 格式）
sortStr：排序（json 格式）
limitStr：请求条目（数字：如 10）
pageStr：分页条件（如：0,10）
userStr：用户 id 参数（json 格式）
idStr：记录 id（json 格式）
callback：回调方法

2、 保存数据方法

```
DBOwnerDataSave(string appid, string objName, string saveStr, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID
objName：数据对象
saveStr：存储数据（json 格式）
callback：回调方法

3、 更新数据方法

```
DBOwnerDataUpdate(string appid, string objName, string condStr, string dataStr, string idStr, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

objName：数据对象

condStr：查询条件（json 格式）

dataStr：更新数据（json 格式）

idStr：记录 id（json 格式）

callback：回调方法

4、 删除数据方法

```
DBOwnerDataRemove(string appid, string objName, string condStr, string idStr, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

objName：数据对象

condStr：查询条件（json 格式）

idStr：记录 id（json 格式）

callback：回调方法

5、 数据总数方法

```
DBOwnerDataCount(string appid, string objName, string condStr, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

objName：数据对象

condStr：查询条件（json 格式）

callback：回调方法

6、 指定 APP 详情

```
DBOwnerAppInfoGet(string appid, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

callback：回调方法

7、 指定用户 user_id 的 APP 列表

```
DBOwnerAppInfoList(string user_id, DevRequestCompletedHandler callback)
```

user_id：用户 user_id

callback：回调方法

8、 指定 APP 数据对象

```
DBOwnerAppDataObjList(string appid, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

callback：回调方法

9、 指定 APP 数据对象详情

```
DBOwnerAppDataObjGet(string appid, string AppDataObjID, string AppDataObjCode, string AppDataObjName, DevRequestCompletedHandler callback)
```

user_id：用户 user_id

AppDataObjID：数据对象 ID

AppDataObjCode：数据对象识别码

AppDataObjName：数据对象名

callback：回调方法

10、 指定 APP 数据对象属性

```
DBOwnerAppDataTypeList(string appid, string AppDataObjID, string AppDataObjCode, string AppDataObjName, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

AppDataObjID：数据对象 ID

AppDataObjCode：数据对象识别码

AppDataObjName：数据对象名

callback：回调方法

11、 指定 APP 数据对象属性

```
DBOwnerAppDataTypeGet(string appid, string AppDataObjName, string AppDataObjCode, string AppDataTypeName, string AppDataTypeCode, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID

AppDataObjName：数据对象名
AppDataObjCode：数据对象识别码
AppDataTypeName：数据对象属性名
AppDataTypeCode：数据对象属性识别码
callback：回调方法

12、 数据对象分类信息

```
DBOwnerDataTypeClassList(string DataTypeClassID, DevRequestCompletedHandler callback)
```

DataTypeClassID：分类 ID
callback：回调方法

13、 数据对象分类信息

```
DBOwnerDataTypeClassGet(string Code, DevRequestCompletedHandler callback)
```

Code：分类识别码
callback：回调方法

14、 数据对象分类信息

```
DBOwnerFileUp(string file, DevRequestCompletedHandler callback)
```

file：图片
callback：回调方法

15、 数据对象分类信息

```
DBOwnerFileList(string appid, string FileType, string PageSize, string Page, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID
FileType：文件类型；如 image/jpeg
PageSize：每一页条目
Page：页码
callback：回调方法

16、 数据对象分类信息

```
DBOwnerFileDelete(string appid, string FileCode, DevRequestCompletedHandler callback)
```

appid：所请求数据 AppID
FileCode：文件识别码
callback：回调方法

17、 公共数据

DBOwnerPublicData(DevRequestCompletedHandler callback)
callback：回调方法

18、 公共数据

DBOwnerCloudCode(string ClassName, string Function, string Params, DevRequestCompletedHandler callback)
ClassName：类名
Function：方法名
Params：参数（json 格式）
callback：回调方法