**SQLite3 Window**

1. 关于此插件

此插件用于转换Excel文件中的数据到数据库中，并自动生成对应的数据类。

1. Tools 菜单介绍
2. Open Folder该按钮内包含两个子菜单
   1. Persistent Data Path 点击该菜单会在系统文件夹中打开Application.PersistentDataPath 路径；
   2. Streaming Assets Path点击该菜单会在系统文件夹中打开Application.persistentDataPath 路径，当该路径不存在时，会自动创建该目录；
3. SQLite3 Window 点击该菜单会打开Excel转换为数据库数据及创建自动化脚本的窗口，该窗口会在下面进行介绍
4. SQLite3 Window 窗口介绍
5. 你需要建立你的Excel文件，该文件需要按照以下格式进行设定：
   1. 文件的第一行为名称、该名称会作为数据库字段名，对应的数据类的变量名称。第一行必须被命名为ID；
   2. 文件的第二行为数据类型，可选的数据类型为

|  |  |  |
| --- | --- | --- |
| C# 类型 | SQLite3 类型 | 备注 |
| int | Integer |  |
| bool | Integer | 在Excel中填入true/1时为真，false/0时为假。在数据库中存入值为1/0 |
| float | Real |  |
| double | Real |  |
| string | string |  |

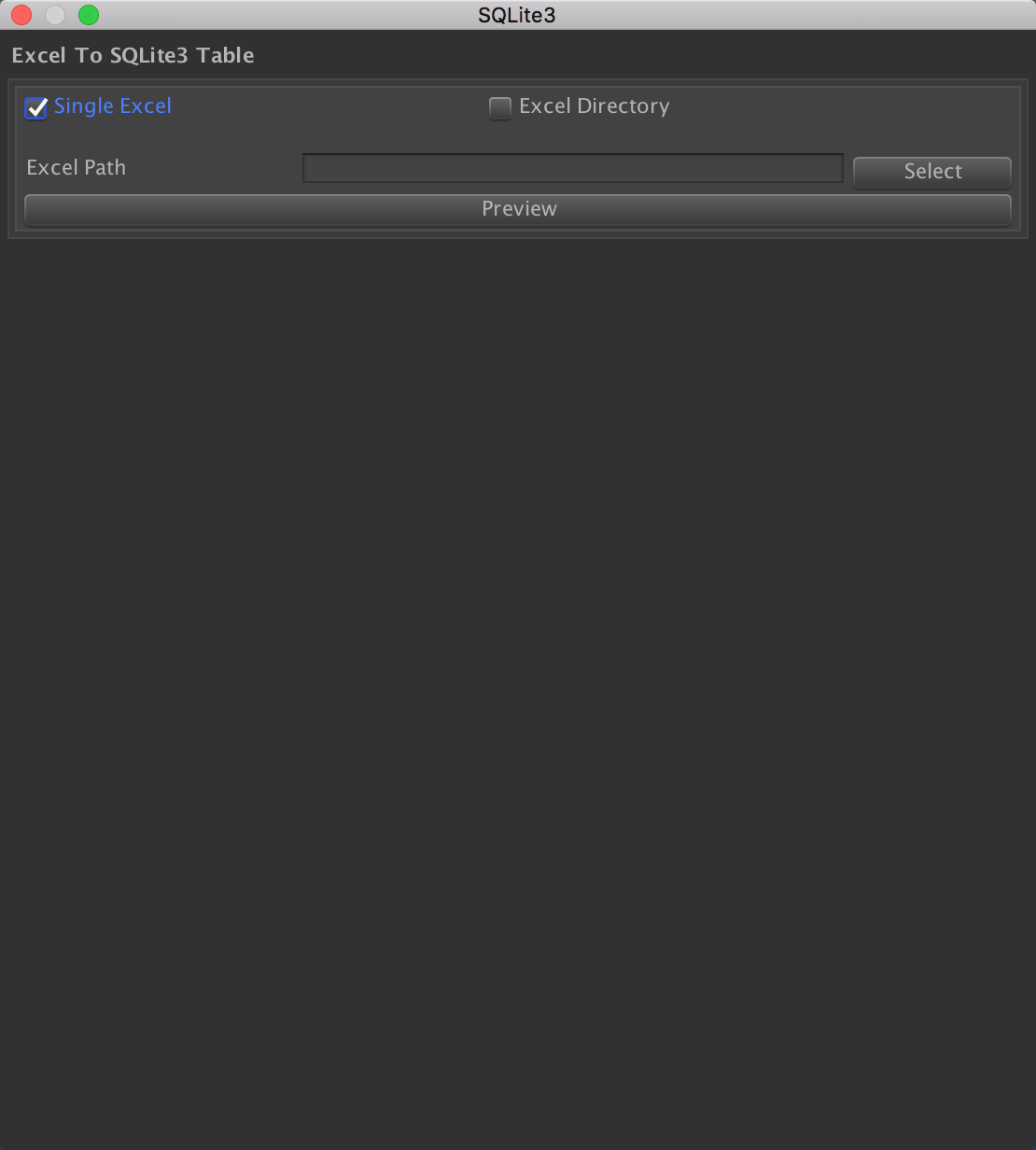
\*当前版本只支持表中类型，下个版本将会加入对数组的支持；

* 1. 第三行为注释行，在将Excel数据导入到数据库时，这一行数据会被忽略，当创建C#数据类的时候该行将会以注释的形式添加到属性之后。如：

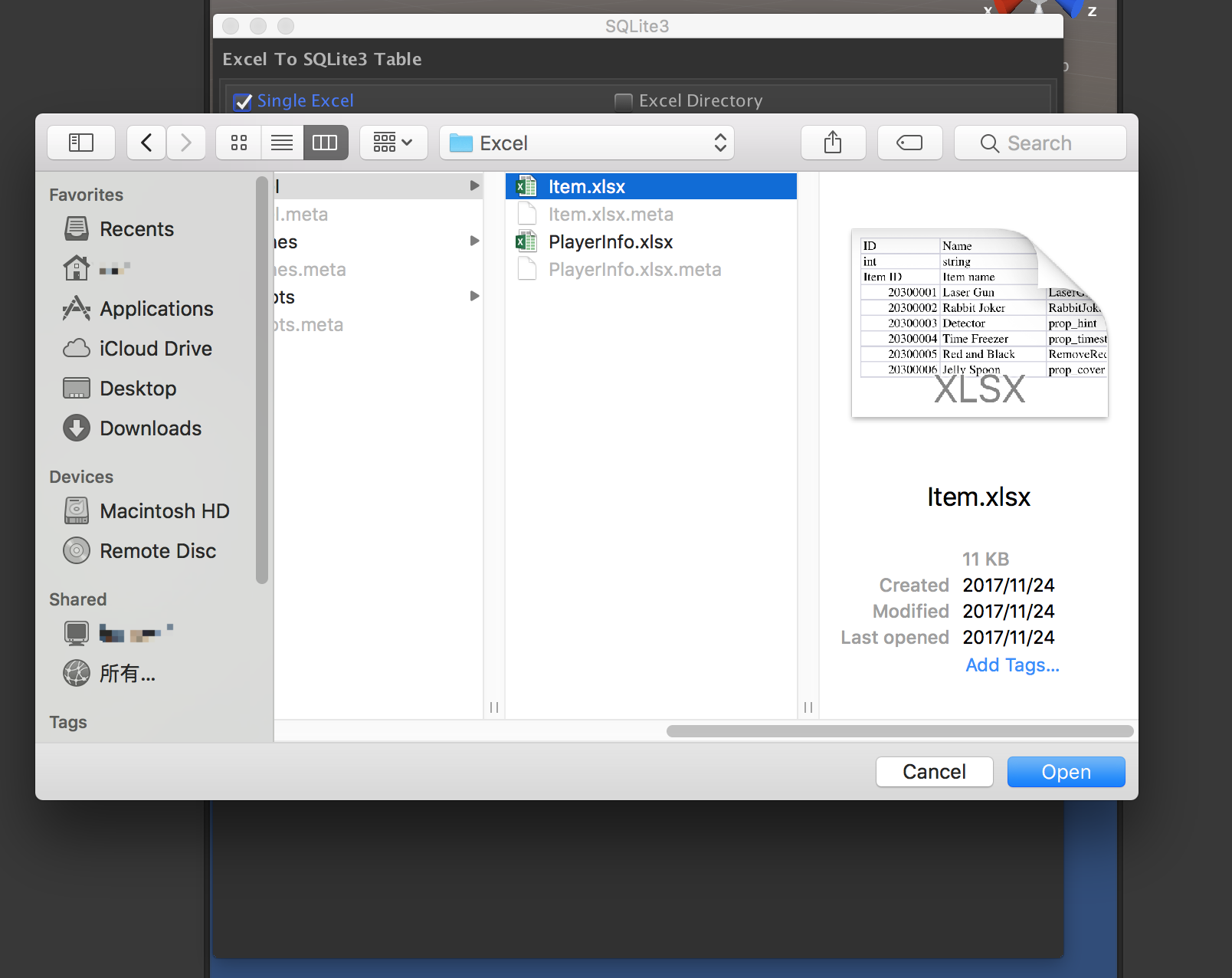
public int ID { get; set;} //唯一标示，序列；

* 1. 从第四行开始是该Excel的真是数据，这些数据将按照SQLite3 类型存入对应的数据库中；

1. 请将你的Excel文件拷贝到Unity项目的Assets目录下，请不要放在特殊文件夹下，比如Resources，StreamingAssets，Plugins，推荐将文件存储在Assets/Editor/下，这样就不用担心这些文件会增加安装包或者运行时内存的大小。Unity不会把Editor下的任何文件放入到安装包，当然Unity也不会将在特殊文件夹以外并且没有引用到的文件放入安装包；
2. 现在你就可以打开Unity菜单栏Tools下的子菜单 SQLite3 Window，你将会看到像下图一样的窗口：

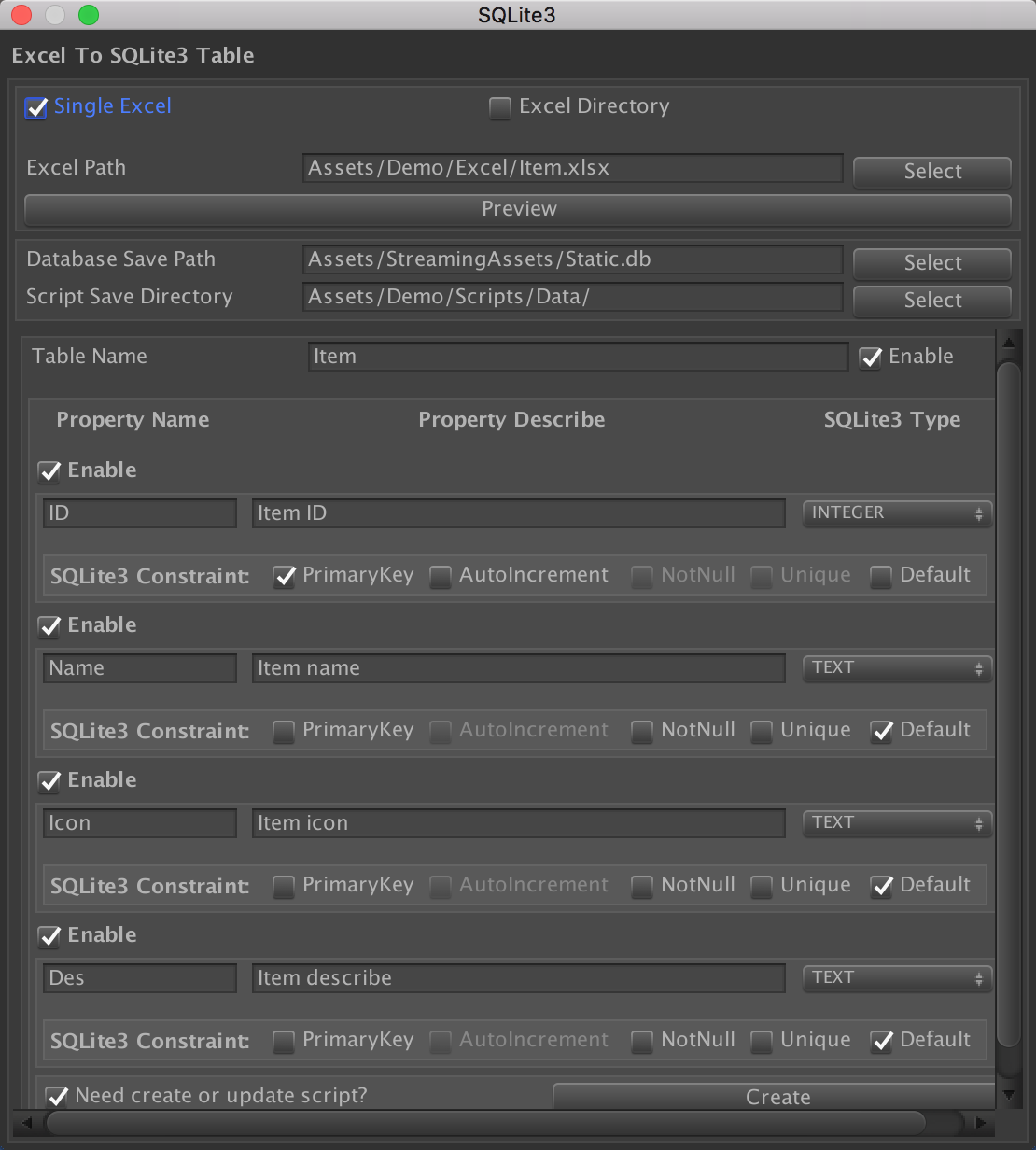


1. 你可以点击‘Single Excel’用来打开一个Excel文件，或者选择‘Excel Directory’用来打开一个目录下的所有Excel文件；
2. 当你选择‘Single Excel’后，你可以在下方‘Excel Path’右侧的输入框中输入Excel文件的路径，或者你也可以点击最右侧的Select按钮，这会弹出一个系统内置的选择文件窗口，你可以像下图这样选择你的Excel文件：



该窗口使用Unity内置的API:EditorUtility.OpenFilePanel()，所以需要你将excel文件拷贝到项目Assets目录下，不然将会在Unity中产生一个Warring，并且产生不可预料的错误，虽然有时会正常运行。该插件会记录你的选择并用EditorPrefs存储在本地，当再次打开该窗口，上次的选择会自动填充到输入框中；

1. 选择一个文件，然后点击Open按钮，你选择的文件路径会被填充到输入框中，点击Preview按钮，你选择的Excel文件中的前三行数据会被展示在窗口中，就像这样：



你会看到Preview下出现了两个编辑框，Database Save Path这个编辑框用来让你输入数据库所在目录，当然你也可以像选择Excel文件那样点击Select按钮选择一个目录进行存储，不同的是点击这个Select按钮弹出的是保存文件窗口，当你选择的文件已经存在的时候会弹出是否覆盖的选项，请按照你的需求进行选择。该弹框使用Unity内置API: EditorUtility.SaveFilePanel()实现，所以你选择的路径也必须是Unity项目Assets目录下，同样的编辑器会使用EditorPrefs记录你的选择，并在下一次进行自动填充。

Script Save Directory这个编辑框用来选择你要把自动生成的脚本存放的目录，使用方式和要求如上。

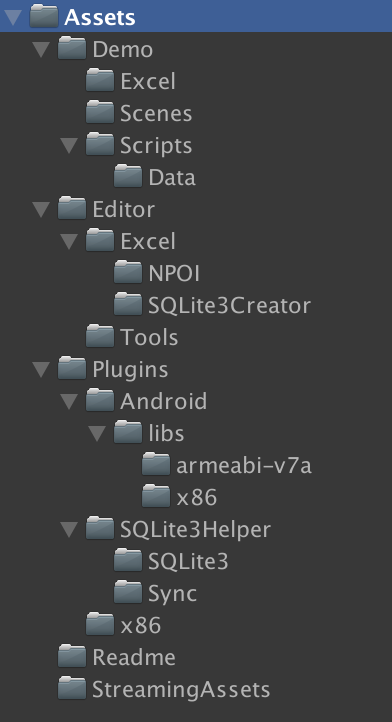
1. 在这两个输入框下方的输入框中是Excel的名称，你可以在此进行编辑，这个编辑框中文本将作为脚本名称以及对应的类名。右侧的Enable选框决定该文档是否会被存储到数据库和是否会被创建脚本。
2. 下方是每一个字段信息：
3. Enable选框标记该字段是否可用，即是否被存储到数据库中，是否被写入到数据类脚本中；
4. 下方的输入框从左到右依次是字段名称，字段注释，字段对应的SQLite3类型，你可以在此做出更改，但是建议你直接更改Excel中的数据，因为在这做的更改只会在此次导出中生效，不会被保存到Excel文件中，下次生成的时候会被Excel中的设置覆盖。最后的SQLite3类型也不建议进行更改，除非你认为有必要进行更改并确保不会引起程序错误；
5. 最后一行是SQLite3的约束条件，该行的选项会被转换成SQLite3字段的约束条件存入数据库，并且会以SQLite3ConstraintAttribute特性标记在数据类属性的上方，用以使用数据类创建相同的表结构。改行会对你做出的选择进行简单的限制，以确保符合SQLite3的语法，限制条件如下：
   1. 当你选择任意一个不是Default的约束时，Default会自动被取消；当你选择Default时，其余选项将会被自动被取消，因为Default和其他约束互斥；
   2. 当你选择该行作为Primary key时，Not null 和 Unique会自动被取消，因为Primary key会自动包含这两个约束；
   3. 当数据的格式不为Integer时，AutoIncrement会自动设置为不可选状态，因为只有Integer字段才可以设置为自增;
6. 最下方‘Need create or update scripts?’左侧的选框标示是否需要创建或者更新脚本，点击右侧的Create按钮将会把Excel文件中的数据写入数据库中，并根据你的选择创建或更新脚本。
7. 当你选择 ‘Excel Directory’时，会出现下方的界面



这个节目和‘Single Excel’存在以下几个区别：

* 1. Excel Directory 右侧的Select按钮将会打开一个目录选择窗口，这个窗口将会让你选择一个目录，而非文件，当然这个目录也必须在项目Assets目录下；
  2. 在Database Save Path和Script Save Path下方会出现一个Create All的按钮，点击此按钮会创建所有下方列出的Excel文件所对应的脚本，并将所有数据写入对应的数据库表中；
  3. 如果当前窗口未显示全部的Excel表格右侧会出现下拉框，只需要下拉滚动条就可以看到所有的文件；

1. 文件目录结构



Demo //此文件夹下为该插件的列

../Excel //此文件夹下为Demo用到的Excel文件

../Scenes //Demo场景，分别演示了只读，创建，写，检查等场景

../Scripts //Demo用到的脚本

../../Data //自动生成的数据类脚本

Editor //插件用到的脚本以及dll

../Excel //读取和转换Excel用到的脚本和dll

../../NPOI //NPOI用于读取Excel文件

../../SQLite3Crator //插件所需的脚本

../Tools //打开文件夹的编辑器脚本

Plugins //SQLite3所用到的dll，请确保该目录不被更改

../Android //该目录下为Android环境下的SQLite3所需的dll

../x86 //该目录为x86环境下SQLite3所需的dll

../SQLite3Helper //该文件夹下为该插件运行时所需的脚本

../../SQLite3 //该文件夹下为运行时访问SQLite3数据库的API

//以及有关SQLite3的定义，特性等

../../Sync //反射相关脚本，用于利用反射获取类名，属性名等信息

ReadMe //该目录存放注释文档

StreamingAssets //请将创建的数据库文件放置在该目录下，在运行时//脚本会自动将该目录下对应的数据库文件拷贝到//PersistentDataPath，并进行访问

1. 脚本用法注释
   * 1. SyncBase
        1. 该类为所有数据类父类，实现了通过反射获取类名，属性名，获取和设置属性值接口
        2. 可以同过继承该类实现

RegisterPropertyChanged //注册属性更改事件

UnRegisterPropertyChange //删除属性更改事件

OnPropertyChanged //调用属性更改事件

用于当属性更改时广播属性更改事件给所有注册了该属性更改事件的对象。

* + - 1. 使用该委托实现属性广播更新请使用OnSyncOne/OnSyncAll更新属性值。如果要使用该功能，建议将属性的set方法设置为private。
    1. SyncAttribute
       1. 该类定义一个特性，用于显式的为每个属性定义一个index，该index对应SyncFactory中PropertyInfo数组的index，用于通过index访问或更改该属性的值及名称，建议使用同名枚举进行定义。



* + - 1. 如果是用枚举进行index定义，请注意使用(int)ItemEnum.ID而不要使用ItemEnum.ID.GetHashCode，因为后者或产生装箱拆箱操作；
    1. SyncProperty
       1. 该类定义了一个类所有需要用到的反射信息，包括类Type，类名，属性列表，属性个数；
    2. SyncFactory
       1. 该类通过反射获取类的SyncProperty属性对象并进行存储，防止频繁使用反射产生不必要的性能损耗，众所周知反射性能一直备受诟病，所以每一个SyncBase子类只会调用一次反射方法；
       2. 该类在SyncBase的构造方法中构建对象，所有SyncBase的子类在创建对象时都会自动调用该类生成自己的SyncFactory对象，所有通过该类生成的对象都被保存在该类的静态字典中，该类提供两个静态方法用于获取已创建的SyncFactory对象。同时也可以通过静态方法获取子类的SyncProperty对象。
    3. SQLite3
       1. 该类定义了从dll中获取的所有外部SQLite3 API。
    4. SQLite3OpenFlags
       1. 该枚举定义了打开数据库的不同方式对应的值，可以混合使用
    5. SQLite3Constraint
       1. 该枚举定义了SQLite3字段约束，可以混合使用
    6. SQLite3Result
       1. 该枚举为数据库操作结果对应的值
    7. SQLite3ConfigOption
       1. 该枚举定义了在多线程还是单线程下使用数据库
    8. SQLite3DataType
       1. 该枚举列举了所有数据库字段类型，用于显示在SQLite3Window上

1. 错误信息
   * + 1. Create table
          1. 默认参数InOverWrite 为true时会默认先删除存在的表
          2. 如果出错则可能是数据库已存在相同名称的表
       2. Drop table
          1. 该方法会在table存在的时候删除，不存在的时候忽略该命令
       3. Insert
          1. 表不存在
          2. 传入的值于表中结构不对应
       4. select
          1. 表不存在
          2. 如果查询结果不存在则返回空值，请在使用返回值之前进行判断
       5. delete
          1. 请参照错误信息
       6. 以上仅列出可能的错误信息，因为每个错误都可能会报相同的信息为 out of memory如果报次信息请跟进以上错误信息进行排查。也可以联系我。
2. 关于作者
   * + 1. 个人主页：shenzhaonan.cn//并没有什么内容，可以给我留言
       2. 个人邮箱：[hellomercury@163.com](mailto:hellomercury@163.com) //每天晚上在线
       3. 腾讯企鹅：282772481