**Unity版本： 4.6**

**使用语言： C#**

**写在前面**

项目开发中经常会对数据库进行访问和操作，很多人直接建一个脚本，在脚本中直接访问数据库！

是的，对于数据库的操作，我们应该用一种更好的方式，让代码看上去更清晰。（耐心读起来）

**实现功能**

1. 创建数据架构，实现统一接口

2. 简单、清晰的访问数据

3. 跨平台（手机、PC端通用）

**基本架构**

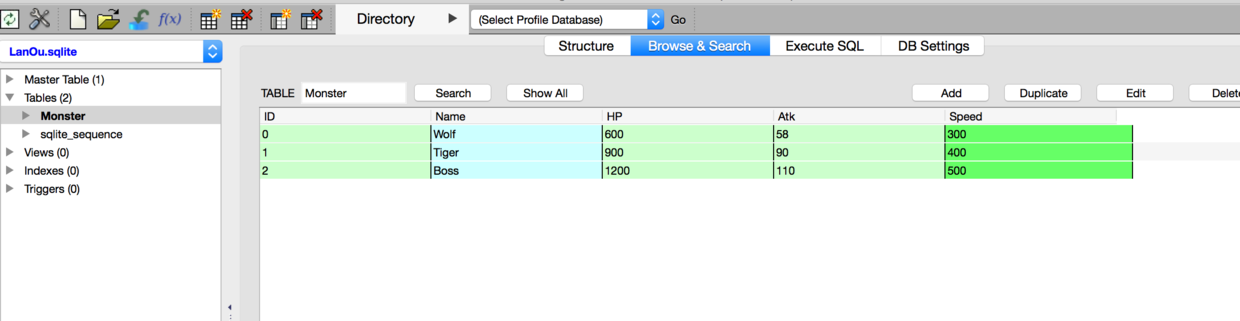
1. 数据库建表

2. 在客户端建立对应的数据结构表（基石）

3. 创建数据库管理类（可以访问数据库）

4. 封装读取数据的接口（核心）

* **1 在数据库建表，添加三条数据**



数据库建表(怪物表 ->主键ID 名字、血量、攻击力、速度）

* **2 在客户端建立对应的数据结构表**

using UnityEngine;  
using System.Collections;  
//**所有数据结构表的基类（核心）**  
//**数据库中每建一个表，就要在客户端新建一个数据结构**

public abstract class DBData{

public int mID; //主键，规定所有的表结构都要有此主键

#//此方法用来解析数据(很重要)

public abstract void Parse(string[] result);

}

//**所有的结构表都要跟数据库的表名相同**  
public class Monster : DBData {

public string mName; //名字

public int mHP; //血量

public int mAtk; //攻击力

public int mSpeed; //移动速度

public override void Parse (string[] result)

{

int index = 0;

mID = System.Convert.ToInt32(result[index++]);

mName = result[index++];

mAtk = System.Convert.ToInt32(result[index++]);

mSpeed = System.Convert.ToInt32(result[index++]);

}

}

* **3 创建数据库管理类**

//数据库访问配置类  
public static class DBConfig  
{

//PC端的数据配置路径

public static string PC\_FilePath = Application.streamingAssetsPath + "/LanOu.sqlite";

public static string PC\_ConnPath = "Data Source = " + Application.streamingAssetsPath + "/LanOu.sqlite";

//移动端的数据配置路径 (persistentDataPath--沙盒)

public static string Mobile\_FilePath = Application.persistentDataPath + "/LanOu.sqlite";

public static string Mobile\_ConnPath = "URI=file: " + Application.persistentDataPath + "/LanOu.sqlite";

public static string Mobile\_StreamingPath = "jar:file://" + Application.dataPath + "!/assets/" + "LanOu.sqlite";

}  
public class DBManager {

#region 单例

private DBManager(){

SetConnStr();

Debug.Log(connPath);

Debug.Log(filePath);

}

private static DBManager instance;

public static DBManager Instance {

get {

if(instance == null)

{

instance = new DBManager();

}

return instance;

}

}

#endregion

#region Params

private SqliteCommand comd;

private SqliteConnection conn;

private SqliteDataReader dataReader;

private static string connPath;

private static string filePath;

#endregion

#region Method

//根据不同平台,配置连接字符串

private void SetConnStr()

{

#if UNITY\_ANDROID

connPath = DBConfig.Mobile\_ConnPath;

filePath = DBConfig.Mobile\_FilePath;

#endif

#if UNITY\_STANDALONE\_WIN

connPath = DBConfig.PC\_ConnPath;

filePath = DBConfig.PC\_FilePath;

#endif

#if UNITY\_STANDALONE\_OSX

connPath = DBConfig.PC\_ConnPath;

filePath = DBConfig.PC\_FilePath;

#endif

}

//开启数据库

private void OpenDB()

{

if(conn == null)

{

//移动端需要我们把数据库复制到沙盒中

if(!File.Exists(filePath))

{

WWW www = new WWW(DBConfig.Mobile\_StreamingPath);

while(!www.isDone){}

File.WriteAllBytes(filePath, www.bytes);

}

conn = new SqliteConnection(connPath);

conn.Open();

}

if(comd == null)

{

comd = conn.CreateCommand();

}

}

//关闭数据库

private void CloseDB()

{

if(conn != null)

{

conn.Close();

}

if(comd != null)

{

comd.Dispose();

}

}

#//封装方法,根据主键 解析数据（核心核心核心核心！）

public void GetData<T>(T data, int id) where T : DBData

{

string sqlStr = "select \* from " + data.GetType().Name + " where id = " + id;

string[] result = Execute(sqlStr);

if(result != null)

{

data.Parse(result);

}

}

//执行查询操作，返回结果形式：字符串数组

public string[] Execute(string sqlStr)

{

try {

OpenDB();

comd.CommandText = sqlStr;

dataReader = comd.ExecuteReader();

string[] result = null;

while(dataReader.Read())

{

result = new string[dataReader.FieldCount];

for(int i = 0; i < dataReader.FieldCount; i++)

{

result[i] = dataReader.GetValue(i).ToString();

}

break;

}

CloseDB();

return result;

} catch (System.Exception ex) {

return null;

Debug.Log(ex.ToString());

}

}

#endregion

}

* **4 在客户端访问数据**

//创建枚举，表示怪物分为三类

public enum MonsterType

{

Wolf = 0,

Tiger = 1,

Boss = 11

}

public class Test : MonoBehaviour {

public Monster wolf; //内存放了怪物的信息

//我们可以在外面动态改变怪物的类型

public MonsterType id = MonsterType.Wolf;

void Start()

{

wolf = new Monster();

#//这里很方便的实现了读取数据

DBManager.Instance.GetData<Monster>(wolf, (int)id);

print (wolf.mName);

}

}