# DBUtil参考手册

[DBUtil参考手册 1](#_Toc474162851)

[一、概述 2](#_Toc474162852)

[二、快速示例(以SQLSERVER为例) 2](#_Toc474162853)

[三、类说明 4](#_Toc474162854)

[3.1 IDBFactory 4](#_Toc474162855)

[3.2 IDbAccess 4](#_Toc474162856)

[3.3 Result 4](#_Toc474162857)

[3.4 DataBaseType 4](#_Toc474162858)

[3.5 GridProp 4](#_Toc474162859)

[3.6 SearchResult 5](#_Toc474162860)

[3.7 TableStruct 5](#_Toc474162861)

[3.8 SerialChunk 5](#_Toc474162862)

[3.9 IDSNOManager 5](#_Toc474162863)

[3.10 SimpleIDSNOManager 5](#_Toc474162864)

[3.11 SqlServerIDbAccess 5](#_Toc474162865)

[3.12 OracleIDbAccess 5](#_Toc474162866)

[3.13 MySqlIDbAccess 5](#_Toc474162867)

[3.14 SQLiteIDbAccess 5](#_Toc474162868)

[3.15 OleIDbBase 6](#_Toc474162869)

[3.16 AccessIDbAccess 6](#_Toc474162870)

[四、创建iDb对象 6](#_Toc474162871)

[4.1 SQLSERVER 6](#_Toc474162872)

[4.2 ORACLE 6](#_Toc474162873)

[4.3 MYSQL 6](#_Toc474162874)

[4.4 SQLITE 6](#_Toc474162875)

[4.5 ACCESS 7](#_Toc474162876)

[五、其他常见方法 7](#_Toc474162877)

[5.1 测试连接是否成功 7](#_Toc474162878)

[5.2 事务控制 7](#_Toc474162879)

[5.3 控制连接打开和关闭 7](#_Toc474162880)

[5.4 获取数据库参数模型和前缀 7](#_Toc474162881)

[5.5 获取日期型过滤字符串 8](#_Toc474162882)

[5.6 生成分页查询语句 8](#_Toc474162883)

[5.7 判断表、视图、列是否存在 8](#_Toc474162884)

[六、ID和自动编号生成控制 8](#_Toc474162885)

[6.1 说明 8](#_Toc474162886)

[6.2 优点 9](#_Toc474162887)

[(1). ID生成时解放数据库字段设置 9](#_Toc474162888)

[(2). 提升速度 9](#_Toc474162889)

[(3). 生成进度的可视化操作 9](#_Toc474162890)

[6.3 操作方法 9](#_Toc474162891)

[6.3.1 ID操作 9](#_Toc474162892)

[6.3.2 自动编号操作 10](#_Toc474162893)

# 一、概述

数据库操作工具,支持SQLSERVER、oracle、mysql、sqlite、access等oldeb连接。

# 二、快速示例(以SQLSERVER为例)

环境准备：添加对DBUtil.dll的引用

代码操作如下：

DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Data Source=.;Initial Catalog=JACKOA;User ID=sa;Password=sa;", "SQLSERVER");

//1.快速查询

Console.WriteLine(iDb.GetFirstColumnString("select Name from SysUser"));

//2.执行sql语句

Console.WriteLine(iDb.ExecuteSql("update test set CaseNo=CaseNo+'e'"));

//3.执行带参数的SQL语句

Console.WriteLine(iDb.ExecuteSql("update test set CaseNo=@case", new IDataParameter[] { iDb.CreatePara("case", "123") }));

//4.添加数据

Hashtable ht = new System.Collections.Hashtable();

ht.Add("CaseNo", "456");

ht.Add("Name", "ji");

Console.WriteLine(iDb.AddData("test", ht));

//5.修改数据

Hashtable ht2 = new System.Collections.Hashtable();

ht2.Add("Name", "jiko");

Console.WriteLine(iDb.UpdateData("test", ht2," and CaseNo='123'"));

//6.删除数据

Console.WriteLine(iDb.DeleteTableRow("test", " and id=11"));

//7.一般查询

Console.WriteLine(iDb.GetDataSet("select \* from sysuser").Tables[0].Rows.Count);

# 三、类说明

## 3.1 IDBFactory

主要用于创建iDb对象,其IDSNOManage字段是ID和编号生成控制的实现,可以在程序初始化时设置来扩展ID和编号生产控制

## 3.2 IDbAccess

数据库访问对象,常见操作见注释

## 3.3 Result

处理结果模型属性Success代表是否成功,Data代表返回的附加信息

## 3.4 DataBaseType

数据库种类枚举,ORACLE = 0, SQLSERVER = 1, MYSQL = 2, SQLITE = 3, ACCESS = 4

## 3.5 GridProp

表格参数模型

## 3.6 SearchResult

表格查询结果

## 3.7 TableStruct

表结构描述对象（当前只用于SQLSERVER中,在SqlServerIDbAccess中会看到）,另外还有: Proc、Func、TableStruct. Column、TableStruct. Constraint、TableStruct. Trigger、TableStruct. Index

## 3.8 SerialChunk

自动编号的参数模型

## 3.9 IDSNOManager

用于管理自动编号生成的接口

## 3.10 SimpleIDSNOManager

简单实现ID和自动编号生成控制的类

## 3.11 SqlServerIDbAccess

实现IDbAccess,用于操作MS SQLSERVER,额外扩展了IDbAccess的方法

## 3.12 OracleIDbAccess

实现IDbAccess,用于操作ORACLE

## 3.13 MySqlIDbAccess

实现IDbAccess,用于操作MYSQL

## 3.14 SQLiteIDbAccess

实现IDbAccess,用于操作SQLITE

## 3.15 OleIDbBase

实现IDbAccess,代表OLEDB数据连接的操作对象的抽象基类

## 3.16 AccessIDbAccess

继承OleIDbBase,实现ACCESS数据库操作

# 四、创建iDb对象

## 4.1 SQLSERVER

(1).添加对DBUtil.dll引用

(2). DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Data Source=.;Initial Catalog=JACKOA;User ID=sa;Password=sa;", "SQLSERVER");

## 4.2 ORACLE

(1).添加对DBUtil.dll引用,将Oracle.DataAccess.dll拷贝到程序运行目录,如果需要对oracle进行细粒度操作则需要添加对Oracle.DataAccess.dll的引用

(2). DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Data Source=ORCLmyvm2;Password=sys123;User ID=sys;DBA Privilege=SYSDBA;" , "ORACLE");

## 4.3 MYSQL

(1).添加对DBUtil.dll引用,将MySql.Data.dll拷贝到程序运行目录,如果需要对mysql进行细粒度操作则需要添加对MySql.Data.dll的引用

(2). DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Data Source=localhost;Initial Catalog=test;User ID=root;Password=hujie110;" , "MYSQL");

## 4.4 SQLITE

(1).添加对DBUtil.dll引用,将System.Data.SQLite.dll、SQLite.Interop.dll拷贝到程序运行目录,如果需要对SQLITE进行细粒度操作则需要添加对System.Data.SQLite.dll的引用

(2). DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Data Source=f:\demo.db;" , "SQLITE");

## 4.5 ACCESS

(1).添加对DBUtil.dll引用

(2). DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=G:\work\Multiplan.mdb;", "ACCESS");或DBUtil.IDbAccess iDb = DBUtil.IDBFactory.CreateIDB("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\Administrator\Desktop\demo.accdb;", "ACCESS");

# 五、其他常见方法

## 5.1 测试连接是否成功

DBUtil.Result res = iDb.OpenTest();

## 5.2 事务控制

iDb.BeginTrans();//开启事务

IDbTransaction tran= iDb.tran;//获得事务对象

iDb.Commit();//提交事务

iDb.Rollback();//回滚事务

bool b = iDb.IsTran;//是否开启了事务

## 5.3 控制连接打开和关闭

iDb.IsKeepConnect = true;//保持连接的打开状态

## 5.4 获取数据库参数模型和前缀

IDataParameter para= iDb.CreatePara();//获取空参数模型

IDataParameter para2 = iDb.CreatePara("name","jack");//获取设置好的参数模型

string str = iDb.paraPrefix;//获取sql中参数名的前缀

## 5.5 获取日期型过滤字符串

string filter=iDb.GetDateFilter("end", "2016-12-1", "2017-01-01", true, false);//返回 and end>='2016-12-1' and end<'2017-01-01'

## 5.6 生成分页查询语句

string sql = iDb.GetSqlForPageSize("select ID,Name from SysUser where ID>100", " order by ID desc", 10, 1);//MySql返回select ID,Name from SysUser where ID>100 order by ID desc limit 10 offset 10

## 5.7 判断表、视图、列是否存在

bool b=iDb.JudgeTableOrViewExist("test");//判断test表是否存在

bool b2 = iDb.JudgeColumnExist("test", "id");//判断test表中是否有列id

# 六、ID和自动编号生成控制

## 6.1 说明

(1).该库默认使用的生成控制是SimpleIDSNOManager,如果需要扩展可以自定义实现IDSNOManager,并在应用程序启动时书写：IDBFactory. IDSNOManage= new 自定义实现();

(2).默认控制操作将ID和编号的生成控制是将上一次的ID或编号驻留在了内存中,当有生成操作时会先去内存查找是否存在这种类型的上一个记录,如果有则直接递增一个,如果没有就根据参数去数据库里对应生成

## 6.2 优点

### (1). ID生成时解放数据库字段设置

只需将数据库的ID字段设置为数字类型即可,不在需要自增或触发器+序列控制

### (2). 提升速度

除第一次需要从后台数据库中生成ID和编号外,其余全部采用内存记录+程序锁实现

### (3). 生成进度的可视化操作

用户可以在页面上看到当前ID生成的进度和编号生成的进度,并且可以删除或修改上一次生成的记录(删除生效后其后的生成操作将首先从数据库中初始化ID和编号),对于ID可以手动增加控制项(可以新的表名和列名,要保证数据库中真实存在的,如果不存在程序在重启或故障时将不能继续控制这个ID的生成)

## 6.3 操作方法

### 6.3.1 ID操作

//生成下一个ID

int id = iDb.IDSNOManager.NewID(iDb, "test", "id");

//强制从数据库中生成ID

int id2 = iDb.IDSNOManager.NewIDForce(iDb, "test", "id");

//重置ID生成的进度

iDb.IDSNOManager.ResetID("test", "id", 1);

//显示当前控制的所有ID进度

iDb.IDSNOManager.ShowCurrentIDs(null,null);

### 6.3.2 自动编号操作

//生成下一个编号

iDb.IDSNOManager.NewSNO(iDb, "test", "caseno", new List<SerialChunk>(){

new SerialChunk("prefix\_GWFW","Text[GWFW][4]"),

new SerialChunk("RiQiCtr","DateTime[yyyyMMdd][8][incycle]"),

new SerialChunk("SerialNo","SerialNo[1,1,5,,day]")

});

//获取当前编号的生成进度

List<string[]> li = iDb.IDSNOManager.ShowCurrentSNOs(null, null, null);

//重置当前的编号生成

List<SerialChunk> li = new List<SerialChunk>();

li.Add(new SerialChunk(“prefix\_GWFW”,null));

li.Add(new SerialChunk(“RiQiCtr”,null));

li.Add(new SerialChunk(“SerialNo”,null));

iDb.IDSNOManager.ResetSNO(tableName, colName, li, null);