# Moon.Orm 技术文档

版本号:9.0
发布时间:2016-8-31

# Moon.Orm 旗舰版技术文档导航

# 1. Moon. Orm 概述

## 1.1 Moon. Orm 简述

Moon 意思是明月的意思,大家不要理解为 mono. 它是一个 orm 框架. 关于它和其他 orm 的对比,大家可以阅读:www. cnblogs. com/humble/p/3426888. html

大道至简,是其着力点. 其特色是以数据库为根基. 所以灵活便捷易用是主要特色.

EF 优雅,性能及其坑多是其致命伤; NH 历史悠久,然使用和配置复杂; 传统代码生成器三层的生成方案失去了编码的便捷和灵活性; 其他的商业化的 Orm 不予评述。

Moon. Orm,意在打造高性能、易用、便捷、易于维护、多数据库数据源支持的 Orm 框架。

当然实际开发中没有银弹,只有平衡点。

1. 性能:测试报告 http://www. cnblogs. com/humble/p/3472764. html

2. 易用性:实体层一键生成→配置文件→智能感知化地编程。

3. 多数库多数据源支持:Moon. Orm 在一同一个项目中,支持多数据库(种类)、多数据源(连接字符串). Moon. Orm 目前支持的数据库类型有 sqlserver、 sqlite、 oracle、 mysql 等,只要该数据库拥有对应 ADO. NET 库,那么 Moon. Orm 就可以支持.

4. 可维护性:

a) 如果您需要换数据库,直接修改配置文件然后一键生成实体层即可,逻辑代码不动。

b) 如果您需要多数据库,直接添加配置节点即可。

c) 如果您的数据库表结构或字段发生变动,一键重新生成实体层代码即可。

## 1.2 Moon. Orm 追求的方向

### 1.2.1 高性能

这也是架构创建的目的之一,已经将它的性能提升到了极致. 大家可以自己测试. 简单给大家一个和 ADO. NET 的性能对比测试.

更多有关内容请看:http://www.cnblogs.com/humble/p/3472764.html

| | Moon | ADO.Net | Moon | ADO.Net | Moon | ADO.Net | Moon | ADO.Net |
|---|---|---|---|---|---|---|---|---|
| | **Create** | **Create** | **GetModel** | **GetModel** | **GetList** | **GetList** | **事务Create** | **事务Create** |
| 1 | 5907 | 4552 | 1101 | 895 | 4 | 4 | 1112 | 698 |
| 2 | 5774 | 4272 | 1124 | 918 | 8 | 7 | 1238 | 602 |
| 3 | 5132 | 4169 | 778 | 1195 | 9 | 10 | 1239 | 625 |
| 4 | 6057 | 4280 | 1226 | 834 | 3 | 4 | 1113 | 610 |
| 5 | 5513 | 4577 | 1515 | 1147 | 7 | 8 | 1242 | 609 |
| 6 | 5438 | 4350 | 1112 | 1095 | 7 | 6 | 1132 | 692 |
| 7 | 5314 | 4098 | 837 | 1459 | 9 | 11 | 1246 | 612 |
| 8 | 4968 | 4080 | 1278 | 821 | 5 | 5 | 1076 | 595 |
| 9 | 6025 | 4808 | 1120 | 947 | 7 | 9 | 1072 | 636 |
| 10 | 5132 | 4169 | 778 | 1195 | 5 | 6 | 1341 | 590 |
| 平均: | 5526 | 4335.5 | 1086.9 | 1050.6 | 6.4 | 7 | 1181.1 | 626.9 |

(说明：同时请求 10000 条数据，此图为一网友公司对 moon.orm 的测评)

### 1.2.2 易用性强

用过 Moon.Orm 的用户应该可以知道这点.配置简单,智能感知,代码生成器的辅助,会 sql 就可使用之.

后续篇幅介绍了关于如何配置的问题,用户可以自行了解.

### 1.2.3 多数据库多数据源同时支持

在同一个项目中我们常常需要处理这些情况时.目前 moon 的目标,支持 sqlserver 、sqlite、oracle、mysql 等各类关系型数据库库.

1.如果您需要换数据库:直接修改配置文件然后一键生成实体层即可;实体层的代码新建一个项目装起来,在你需要开发的系统中引用这个实体层项目即可.如果要换其他的数据库库(如 sqlserver 换 mysql),重新生成一份 mysql 的实体层代码即可(覆盖原来的),因为实体层中的 model 对外被调用都是一致的,所以根本就不用改代码就能换数据库的目的;

2.如果您需要同时使用多数个据库:直接添加配置文件即可.详情:Moon 使用配置说明

3.如果您的数据库表结构发生变动:一键重新生成实体层代码即可.

### 1.2.4 智能感知

这个不用讲了,值得一提的是 MQL(moon query language,类似于 linq:非 linq),她能够为你提供强大的智能感知功能,并且任何数据库类型差异.

### 1.2.5 NET 2.0 原生支持.

有人问:为什么没有 LINQ、lambda,因为设计理念不同.觉得 MQL 复杂的人们,有了智能感知,你们就适应适应,因为这是萝卜白菜的问题.你不可能在低版本系统中使用 Lamda 等高级功能,当然你会说这样的系统是不是应该淘汰了,当你遇到一些老系统中你就明白了.一句话萝卜白菜各有所爱,设计理念不同.

### 1.2.6 使用便捷.

这个上面的链接也谈到,详情见:http://www.cnblogs.com/humble/p/3293500.html

### 1.2.7 灵活的事务支持

## 1.3 Moon.Orm 学习建议

### 1.3.1 学习建议

首先,给读者最重要的一个建议是配合 API 文档看看 Db 类各个 API 是如何使用的.如果读者能够使用 Db 中各个 API,那么应该使用起来就不是什么大问题了,其实各个方法见名思意,很容易理解,何况还有注释.

其次,配合博文(http://www.cnblogs.com/humble)以及此技术文档作为参考;

再次,结合下面给的 Demo 自己动手试试.当然你可以进入技术群:

群 2:225656797

群 1:216965349

### 1.3.2 随时更新

moon.orm 的更新维护地址:

http://lko2o.com/moon/article/3

代码生成器的更新地址:

http://lko2o.com/moon/article/9

本文档最新地址:

http://files.cnblogs.com/humble/d.pdf

建议用户将生成的 model 等生成到一个文件中,不要自己轻易去动这个文件.当数据库中有调整时,一键生成就可以了.便捷易用.

# 2.Moon.Orm 相关说明

## 2.1 使用授权说明

### 2.1.1 Moon.Orm 任何组织和个人都可以免费使用;

### 2.1.2 Moon.Orm 使用之前必须按下面的规范授权注册(免费);

如不注册,功能会只限于 sqlserver.免费快速免费.当然你可以选中不使用旗舰版,你就不用注册了.之所以要注册是为了规范.

### 2.1.3 未注册即使用的企业,我们有权追究责任;

很多人在项目中得了便利,但从来不愿参与反馈和建议,我们颇感心寒.何况注册免费且快速.所以希望得到大家支持.注册方式如下 2.2 所述。

## 2.2 授权申请规范

### 2.2.1 第一步 填写表格.

| | |
|---|---|
| 申请类型 | 个人或企业 |
| 联系人姓名 | 你可以乱填,但我们有权终止你的授权(博客园可以不用真实姓名,邮件中需真实) |
| 联系邮箱 | |
| 联系 QQ | |

| | |
|---|---|
| 通信地址及邮编 | (博客园中可以不填写地址) |
| 备注(可选) | |

### 2.2.2 第二步 将上面表格填完后截图上传.

将表格截图另存为文件,然后上传到博客园的评论中.

博客地址:http://www.cnblogs.com/humble/p/3323161.html

### 2.2.3 第三步 邮箱联系我们.

同时将表格以邮件的形式发送给我们,发送邮件则需要联系人等所有信息必须用真实信息,不要向发送到博客园那样了.

(注:不要截图,要表格.我们的邮箱是随身的,所以能立即回复你)

邮箱地址:qsmy_qin@163.com

邮件标题:授权申请

### 2.2.4 第四步 我们立刻返回授权文件

我们以邮件的形式,如反馈给你授权码.你建立一个文件名为 moon.license,然后将收到的授权码写入此文件中保存即可.

然后你将授权文件 moon.license 放入正在使用 moon.orm 的项目中(moon.orm.dll 所在之处).该授权文件可以重复用于其他项目中.

如果我们给你文件.请将文件的名字改为 moon.license

## 2.3 技术问题咨询规范

### 2.3.1 第一步 描述问题的规范.(直接复制,不用截图)

问题描述:.....
异常错误信息:.....
异常截图:.....
联系邮箱:.....

### 2.3.2 第二步 发送问题到官方博客

请按照上面的格式将内容发送到以下地址评论中.我们会立即作答,如果按照规范.

反馈地址:http://lko2o.com/moon/article/3

(注:评论与我们的邮箱联系在一起的,所以我们能够随时回复你.)

## 2.4 企业服务指南

注:非诚勿扰

### 2.4.1 获取旗舰版源代码

### 2.4.2 定制 API 功能

可以为你的项目量身定制你所需的 API 功能;

### 2.4.3 数据迁移服务

可以为你已有的项目做数据迁移工作.

### 2.4.4 技术问题咨询服务

可以随时为你项目做相应的技术咨询服务.

## 2.4 获取标准版源码

5.0 之前已经全部开源,5.0 标准版本目前对参与者开源(看看下面的获取其实很容易的),当然以后会逐渐开源出来;

### 2.4.1 开源项目获取源代码

免费赠送标准版源代码.

### 2.4.2 帮助 Moon 获取源代码

1.帮助写 Moon 相关博文一篇;

2.帮助完成周边扩展,比如围绕 Moon 做的一些扩展;

3.帮助完成其他数据库的驱动开发;

4.捐助 Moon.Orm,钱多少不分贵贱,在乎支持诚意;

捐助 Moon{支付宝:shichuangege@126.com}

(套用网友的一句话:为您节约更多时间，去陪恋人、家人和朋友.....,留下)

5.加入代码生成器谱写行业,现在你可以写 oracle、db2、postgreSQL 的代码生成功能,这个很简单,围绕着格式写就可以了;

6.现在推荐 15 个人入群,即可获取源码,告诉入群者你推荐的他(记住不要乱推荐人进来)

# 3.技术详解

## 3.1 开发中如何配置

当开发一个项目时,我们首先将 Moon.Orm.dll 引入项目中;(如果是 sqlite 请 SQLite.Interop.dll 手动放入该项目 dll 所在目录中;如果数据库是 mysql 记得项目生成目录中有 mysql.data.dll),如没有注册就想使用 mysql 数据库,请将代码生成器中的 moon.licence 放入你的项目的生成目录中,当你觉得这个框架不错时实战项目开发时,可免费注册获取授权.

或者使用 NuGet 一键引入, Install-Package MoonOrm

然后利用代码生成器,生成你这个项目数据库对应的 Model 层,然后将生成的 Model 层添加到你的项目中.代码生成器在这里:http://lko2o.com/moon/article/9

接下来,设置程序的配置文件,在 connectionStrings 节点下如下格式配置

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="连接字符串"
    providerName="Moon.Orm,Moon.Orm.SqlServer" />
</connectionStrings>
```

### 3.1.1 配置说明

1)name="DefaultConnection",表示默认配置.通过

var db=Db.**CreateDefaultDb**()调用即可,因为是默认的配置连接;

2)connectionString="连接字符串",表示对应的连接字符串.这个 ADO.NET 一致;

3)providerName="Moon.Orm,Moon.Orm.SqlServer",表示对应的驱动

Moon.Orm 表示 Moon.Orm.dll 中的驱动;(也就是说你自己可以开发驱动)

Moon.Orm.SqlServer 表示其中的 SqlServer 类,还可以使用以下类型

### 3.1.2 Moon.Orm.dll 直接支持的数据库类型

Moon.Orm.SqlServer

Moon.Orm.MySql

Moon.Orm.Oracle (微软默认的)

Moon.Orm.Sqlite

### 3.1.3 Moon.Orm.dll 通用数据库方案

不想看此节可跳过,此方案的目的有两个:其一,为用户的数据库提供自己满意的驱动(比如我要用甲骨文自己的数据库驱动);其二,一些关系型数据库没有提供直接支持,可以采用此方案进行支持.配置方式如下:

```
<connectionStrings>
    <add name="configName" connectionString="连接字符串" providerName="官方提供的ADO.NET 驱动文件名(没有.dll)" />
</connectionStrings>
```

注意,这个 providerName 和上面 3.1.1 的配置稍有差别,providerName 的值如:mysql.data

到 MOON_WORK_DIRECTORY_PATH\SharedDbConfig\providerName.config 这个地方进行对应驱动进行简单配置,这个文件及文件内容会在调用你第一次调用 Db.CreateSharedDbByConfigName("configName")**自动生成为你生成**,配置内容如下,你需要进行填写:

```
<?xml version="1.0"?>
<SharedDbConfigs>
    <SharedDbConfig ProviderDllName="MySql.Data">
        <AutoIncreasePKSqlPosition>
            请填写获取自增长主键值的 SQL 语句所在位置,可值
1:表示和插入语句形成整体执行; 2:插入之后执行
        </AutoIncreasePKSqlPosition>
        <AutoIncreasePKSQL><![CDATA[请填写获取自增长主键值的 SQL 语句]]></AutoIncreasePKSQL>
        <PName>请填写参数化查询所用符号,如@</PName>
    </SharedDbConfig>
</SharedDbConfigs>
```

### 3.1.4 MySql 数据库配置另行说明

(注:2015_8_12 版本之后可以跳过以下内容,当然你也可以根据你的需要进行如下

配置,非必须)

MySql 除了上面的配置,根据你的需要加上以下配置

(注意:如果你系统中默认就安装了.net 的 mysql 驱动,可能就不需要了,具体情况,文档下方的 mysql 常见错误 3.6.3.1 中介绍)

注意其中的版本号 Version=5.2.3.0,这是 moon.orm 中默认的.当然你可以用其他版本,这样的话你就需要自己填写您对应的版本号了.

```
<configuration>
  <system.data>
    <DbProviderFactories>
      <add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient"
           description=".Net Framework Data Provider for MySQL"
           type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data,
Version=5.2.3.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d" />
    </DbProviderFactories>
  </system.data>……
```

### 3.1.5 多数据库多数据源支持

```
<connectionStrings>
      <add name="SQLServer"
connectionString="Server=abc;Database=TestContext;uid=sa;Passwor
d=123456;" providerName="Moon.Orm,SqlServer" />
      <add name="DefaultConnection"
connectionString="server=localhost;port=3306;user
id=root;password=;database=ypzdw_busi_v3;Charset=utf8;"
providerName="Moon.Orm,Moon.Orm.MySql" />
      <add name="DefaultConnection2"
connectionString="server=192.76.5.3;user=abc;password=123!;datab
ase=panguerp;port=3306;Charset=utf8;"
providerName="Moon.Orm,Moon.Orm.MySql" />
</connectionStrings>
```

调用方式如下:

```
using (var db=Db.CreateDbByConfigName("DefaultConnection2"))
{
  //----逻辑代码
}
```

## 3.2 全局配置配置说明

```
<appSettings>
        <add key="USE_TEMP_DLL" value="true"/>
        <add key="MOON_WORK_DIRECTORY_PATH" value="c:\d\a\"/>
        <add key="CLOSE_LOG" value="true"/>
</appSettings>
```

### 3.2.1 MOON_WORK_DIRECTORY_PATH

表示 Moon.Orm 的工作目录的**路径,,**建议不要自己配置这个路径,系统默认有路径

(例如:c:\a\a\de\ 注意最后有一个分隔符,Linux 则反过来的).

如果为空或者没有此节点,(对于 exe 应用程序,就是 exe 所在的目录下的 MOON_WORK_DIRECTORY_PATH 目录;对于 web 应用程序,就在网站的根目录下的 MOON_WORK_DIRECTORY_PATH 目录.)

若不为空,则为你所设定的路径(路径中的文件夹不存在,系统也会自动创建的).

### 3.2.2 USE_TEMP_DLL

系统每次启动的时候,GetDynamicList 方法是否使用上次动态生成好的 dll 来提高首次性能.在使用 GetDynamicList 方法时,第一次 moon 会自动 编译查询语句为 dll,如果 USE_TEMP_DLL 为 false,那么系统重启后会再次重新生成;如果为 true,就不会删除上次的 dll,不用再次重新编译.

赋值:true 或者 false(默认值)

GetDynamicList 请查看 API.相关使用连接:

注:可以没有此节点,没有此节点则默认为 false

### 3.2.3 DLL_EXE_DIRECTORY_PATH

表示项目的 dll 和 exe 所在目录

### 3.2.4 CLOSE_LOG

是否关闭日志功能,默认 false(即日志可用);

## 3.3 从简单的实例了解项目中的使用过程(强烈推荐)

**详情:**http://www.cnblogs.com/humble/p/3415506.html

## 3.4 增删改查剖析

**详情:**http://www.cnblogs.com/humble/p/3380065.html

**以及:**http://www.cnblogs.com/humble/p/3293500.html

## 3.5 常用方法讲解

### 3.5.1 如何自己写驱动

在 3.3.1 配置说明中讲到,我们是可以直接写数据库驱动的.

书写方式:建一个项目,定义一个类,让他继承 Moon.Orm.Db 抽象类,实现里面的方法即可.

### 3.5.2 缓存机制

#### 1)StartCache(int m),缓存时间单位秒

目前系统采用内存缓存的方式,下面的版本会加入 memcached.

Db 调用该方法,那么就会将会缓存到内存中 m 秒.如果 m 秒内,内存中有数据,则会直接从内存中取出数据.

#### 2)MoonCache<T>

这是一个缓存辅助类(静态类),请查看 API 根据提示进行使用.

### 3.5.3 便捷的分页

Moon.Orm 中支持四种格式存储分页数据

(DataSet,DictionaryList,List<T>,Json:String).当然其分页原理一致,只是存储方式不一样而已.至于 DictionaryList 为什么结构,请看 3.5.7.

分页中需要所涉及的参数说明:

mql/sql 表示你要查询的所有数据.

out int sumPageCount,会返回总页数;

out int sumDataCount,会返回数据总条数;

pageIndex 表示页码;

onePageDataCount,表示每页中的数据数;

oneOrderbyFieldName 是 sqlserver 中会用到的排序字段(查询结果中一个字段),其他类型数据库则填写 null(因为其他数据库可以直接将排序写到中).

友情提示:返回的结果都是指定 pageIndex 页的数据.

1)DataSet GetPagerToDataSet(

```
    MQLBase mql,//或者 string sql
    out int sumPageCount,
    int pageIndex,
    int onePageDataCount,
    string oneOrderbyFieldName
);
```

2)DictionaryList GetPagerToDictionaryList(

```
    MQLBase mql,//或者 string sql
    out int sumPageCount,
    int pageIndex,
    int onePageDataCount,
    string oneOrderbyFieldName
);
```

3)List<T> GetPagerToOwnList<T>(

```
    MQLBase mql,//或者 string sql
    out int sumPageCount,
    int pageIndex,
```

```
    int onePageDataCount,
    string oneOrderbyFieldName
)where T : new()
4)String GetPagerToJson(
    string sql,
    object[] parameters,
    out int sumPageCount,
    out int sumDataCount,
    int pageIndex,
    int onePageDataCount,
    string oneOrderbyFieldName)
```

### 3.5.4 手写 sql 的建议:将 sql 写到配置文件中

有些复杂的 sql 语句,当你无法书写为 mql 时,你可以直接写到配置文件中,Moon.Orm会自动帮你于工作目录的sqls文件夹下创建默认配置的文件sql.config,你在其中按照模板格式进行书写 sql 即可.

开发过程中可能几个人需要同时写 sql.config 文件,我们的建议是重新建一个 sql*.cong 格式的文件(例如 sql_a.config,sql_bbb.config....),几个人各用个的文件,但需要注意,其中的 id 需要自己指定一个前缀防止和其他开发人员重复.

调用方式如下:

```
using (var db = Db.CreateDefaultDb()){
    db.ExecuteSql***(SqlConfigUtil.GetSqlByID(db,"getdemo"),12);
}
```

```
//对应 sql*.config 文件内容
<?xml version="1.0"?>
<sqls>
    <sqlxml id="getdemo">
       <defaultsql><![CDATA[所有通用的查询语句,没有特别指定就使用语句]]></defaultsql>
        <sqlserver>![CDATA[sqlserver 查询,不需要则请置空]]</sqlserver>
        <mysql>![CDATA[mysql 查询,不需要则请置空]]</mysql>
        <sqlite>![CDATA[sqlite 查询,不需要则请置空]]</sqlite>
        <oracle>![CDATA[oracle 查询,不需要则请置空]]</oracle>
        <description>![CDATA[简单描述一下条语句的作用]]</description>
    </sqlxml>
</sqls>
```

### 3.5.5 自定义实体类的开发建议

有时候我们会遇到这样的情况:做了一次查询,需要将结果放到一个具体的实体类中,而这个实体类我们没有定义.这个时候 moon.orm 提供两种方式,帮你自动生成实体类.

方法一:通过代码生成器(还没有下载,就去下载吧,)

方法二:通过 db.GetModelBySQL()方法在 VS 中你加一个断点,然后调试时复制一下就可以了.

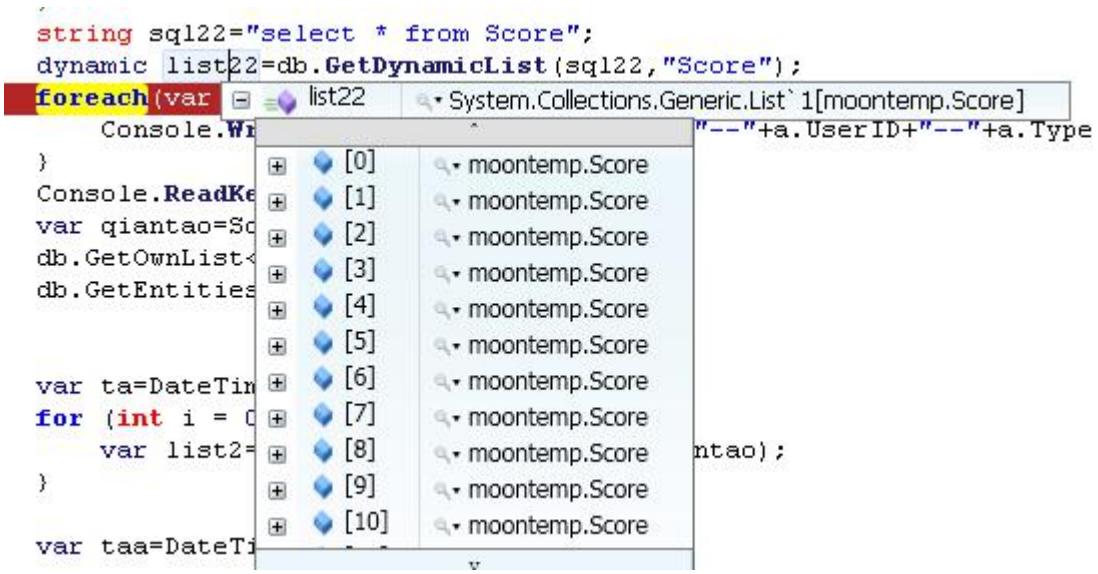
### 3.5.6 sql 之王者归来

使用 db.GetDynamicList ,让你体验另一种自由

```
string sql22="select * from Score";
dynamic list22=db.GetDynamicList(sql22,"Score");//第二参数随便写的(作为类名)
foreach(dynamic a in list22){
Console.WriteLine(a.ID+"--"+a.ScoreM+"--"+a.UserID+"—"+a.TypeName);
```

```
//都是强类型

}
```




### 3.5.7 你需要了解 DictionaryList

DictionaryList 继承于 List<Dictionary<string,MObject>>,含有 ToJson,ToString,ShowInConsole 三个方法,大家看看 API 介绍,有助于以后使用于复杂数据结构,因框架中常常会使用它来存放数据.比如一个对象列表就可以用它来存放.

如,取数据的时候,可以用 list[0]["字段名"].

### 3.5.8 一些辅助你开发的帮助类(位于 Moon.Orm.Util)

#### 3.5.8.1 日志功能(LogUtil)

Moon 系统中也会使用此日志工具类.所写的日志位于工作目录的 MoonLogs 文件夹中,系统会每天写一个日志格式:年-月-日.log.可以通过日志查询日常执行的情况;也可以用来作为你自己的日志工具.该辅助类的特色就是短小易用.

Exception(Exception ex)

Warning(string content)

Error(string content)

Debug(string content)

Write(string type,string content)

#### 3.5.8.2 对象克隆(CloneUtil)

1)克隆继承 EntityBase 的类

```
var p2=CloneUtil.CloneEntityBaseObject<Products>(p);
```

2)克隆任意对象(不要较真)

可以克隆绝大多数类型对象(利用 json,如果 json 可以序列化这个对象且能反序列化回来的话,那么一切 OK)

```
CloneUtil.Clone<类>(obj);
```

3)克隆可序列化的类

```
CloneSerializableObject<可序列化的类>(obj);
```

#### 3.5.8.3 Json 处理(JsonUtil)

详情请查看 api 帮助文档,或者直接 VS 提示查看

#### 3.5.8.4 易如反掌的 Ajax 异步分页(PagerUtil)

#### 3.5.8.5 Enum 辅助类(EnumUtil)

1) String GetEnumNameByValue<T>(int value)

通过枚举的值获取对应的字符串形式,如果没有对应就为空

2) int GetValue(object enumObj)

根据枚举获取对应的 int 值

#### 3.5.8.6 ListUtil

List<List<T>> GetCountListList<T>(List<T> list,int count)

该方法可以将一个 list 数据,分成 n 个 list,每个 list 中的数据条数为

count(当然不够时,就让它不够).硬性分页时可以用得上.

## 3.6 使用中常见的问题及注意点

注:所有问题都可以博客园查询.但记住你以后的提交格式(2.3 技术咨询规范)

http://www.cnblogs.com/humble/p/3391005.html

### 3.6.1 常规问题

#### 3.6.1.1 如何对连接字符串加密处理

目前的处理方式是将连接字符串作为一个全局常量放在代码中,然后通过如下这种方式

```
public  static Db GetYourDb(){
    return new SqlServer(全局变量);
}
using ( Db db=XX.GetYourDb()) {
     //---------下方进行对应的操作
}
```

#### 3.6.1.2 如何批量导入

原本这个问题不是一个问题,但偷懒的用户也是值得尊重的,目前我们没有在 moon.orm 中提供方法,关于 sqlbulk 大家应该有所耳闻,大家目前自己勤快一点点吧,有时间之后,我们会给予支持,谢谢.

#### 3.6.1.3 事务操作

我们先看看 Moon.Orm 的原理,采用了 ADO.NET 最传统的事务方式即 DbTransaction

下面的代码 moon.orm 的事务执行过程,读者自行阅读,代码可到这里下载:

http://www.cnblogs.com/humble/p/4470582.html

```
public static void TestProc(){
  var addID=-11;
  using (var db=Db.CreateDefaultDb()){
    //开启事务功能
    db.TransactionEnabled=true;
    try {
      Products p=new Products();
      p.ProductName="ProductName_TestProc"+DateTime.Now;
      p.Quantity=100;
      p.Remark="标记";
      p.Unit="单元";
      p.UnitPrice=12m;
      p.CategoryId=3;
      db.Add(p);
       //--------------------
      addID=p.ProductId;
      Console.WriteLine("p.ProductId:"+addID);
      Orders order=new Orders();
      order.Comment="test";
      order.CustomerId=-43;//故意制造错误
      order.Finished=false;
      order.OrderDate=DateTime.Now;
      order.SumMoney=33;
      db.Add(order);
      db.Transaction.Commit();
    } catch (Exception ex) {
      db.Transaction.Rollback();
      var aa=ex.Message;
      Console.WriteLine(aa);
    }
  }
  using (var db=Db.CreateDefaultDb()){
    Console.WriteLine("p.ProductId:"+addID);
    var exist=db.Exist(ProductsSet.ProductId.Equal(addID));
    Console.WriteLine("添加的数据存在吗?"+exist);
  }
}
```

3.6.1.4 网络中的数据传输

在传输时有很多传输格式大家可以利用.但在moon.orm,我们推荐使用json.便捷、短小、方法全面,Db、实体对象、DictionaryList都支持序列化方法(可通过VS智能感知查看),JsonUtil内部还有很多相关功能,可让用户高效使用.

3.6.1.5 distinct查询问题

mql中没有直接提供此关键字.但是对于熟悉sql的读者应该很清楚,通过分组可以替代这个功能,而且性能一致.

我们举例说明:

```
var mql=UserInfoSet.SelectAll().where(
        UserInfoSet.ID.IN(
    UserInfoSet.Select(UserInfoSet.ID.Min())
        .GroupBy(UserInfoSet.Name)
    )
    );
    对应sql:
    select * from userinfo where id in(select min(id) from
    userinfo group by name)
```

3.6.1.6 判断当前的数据库类型

```
using(var db....){
    if(db is Sqlserver)
    if(db is MySql)
}
```

3.6.1.7 如何生成Oracle的实体层

目前代码生成器中没有直接提供对oracle的支持,如果想使用oracle请到群共享中下载codesmith模板,用codesmith生成即可.

codesmith去网上下载吧.对应模板的下载地址如下:

http://pan.baidu.com/s/1o69QAuu

#### 3.6.1.8 MQL 如何拼接

下面以一个示例说明如何拼接,相信大家的智商不低于正常水平.

```
using (var db=Db.CreateDefaultDb()) {
        var mql=StudentSet.SelectAll();
        WhereExpression exprssion=null;
        if(true){
          exprssion=StudentSet.ID.BiggerThan(1);
        }
        exprssion=exprssion.And(StudentSet.Name.NotEqual("a"));
        mql=mql.Where(exprssion);
        mql=mql.Top(10);
```

```
SELECT [Student].* FROM [Student] WHERE [Student].[ID]>@p1 AND
[Student].[Name]<>@p2 LIMIT 0, 10
@p1=1
@p2=a
```

### 3.6.2 Sqlite 的问题及注意点

#### 3.6.2.1 使用 sqlite 时的注意项

Sqlite 的项目使用中,SQLite.Interop.dll(位于 Moon.Orm 目录)需要你手动放入你项目的 dll 目录中,因为这是 c++生成的,VS 不知道它的存在;

### 3.6.3 Mysql 的问题及注意点

#### 3.6.3.1 发现 MySql 一切配置正常,但不能运行

> 连接失败!错误信息:"" 的类型初始值设定项引发异常。

```
<configuration>
  <system.data>
    <DbProviderFactories>
      <add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient"
           description=".Net Framework Data Provider for MySQL"
           type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data,
Version=5.2.3.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d" />
    </DbProviderFactories>
  </system.data>
```

如果你已经进行了如上配置,且项目的 dll 目录中有对应版本的 mysql.data.dll 却不能正常运行,那么有可能因为你的系统中已经安装了 mysql 官方的.net 驱动包,如发现此情况,请删除 DbProviderFactories 节点及其所含内容.

#### 3.6.3.2 不能插入数据(duplicate primary key 类型错误)

原本插入的数据就没有重复,但就是报这个错误,此时表示你的 mysql 自己有问题:数据库表的索引名字一样了,你重新为此表的索引命名.

### 3.6.4 SqlServer 的问题及注意点

#### 3.6.4.1 sqlserver2000 数据库支持问题

分页不支持 2000,同样代码生成器也不支持 2000

#### 3.6.4.2 分页时,使用 GetPagerToOwnList<T>的注意点

因分页的原理采用 ROW_NUMBER,因此 T 对应的类中必须要有这个属性(ROW_NUMBER),可自行添加,请将该属性设置为 Int64;