

狼人自传

Dapper系列之一：Dapper的入门(多表批量插入)



2017-09-04 20:53 by 码代码_net, 2579 阅读, 0 评论, 收藏, 编辑

Dapper介绍

简介：

不知道博客怎么去写去排版,查了好多相关博客, 也根据自己做过项目总结, 正好最近搭个微服务框架, 顺便把搭建微服务框架所运用的知识都进行博客梳理, 为了以后复习, 就仔细琢磨写一下自己在微服务框架中对Dapepr的理解以及项目中应用。 **dapper** 只是一个代码文件, 完全开源, 你可以在项目里任何位置, 来实现数据到对象ORM操作(当然先引用Dapper文件), 体积小速度快。使用好处增删改查比较快, 不用自己写sql, 因为这都是重复技术含量低的工作, 还有程序中大量的数据库中读取数据然后创建model, 并且为model字段赋值, 这都是很轻松的, 个人认为Dapper可以看做HelpSQL, 甚至比HelperSQL性能高一点。如果你喜欢原生的SQL, 那么有喜欢ORM的简单, 那你一定钟情于Dapper 并且爱上他。

Dapper的优势：

- 
- 1、Dapper是一个轻量级ORM类, 代码就是一个SQLMapper.cs文件, 编译后一般在40k左右的dll;
 - 2、Dapper快, 为啥说快呢? 因为Dapepr速度接近IDataReader, 取列表的数据超过DataTable;
 - 3、Dapper支持什么数据库? 支持Mysql, sqlLite, SQLServer, Oracle等一系列数据库, (备注: 我个人在在做demo中, 就是使用了Mysql, SQLServer, 公司和个电脑装的数据库不一样, 就都测试了);
 - 4、Dapper的R支持多表并联的对象, 支持一对多, 多对多关系, 并且没侵入性, 想用就用;
 - 5、Dapper原理就是通过Emit反射IDataReader的队列, 来快速得到和产生对象; 这也是性能高的原因之一;
 - 6、Dapper语法简单, 快速入手。
- 如果面试, 让你说出Dapper的好处, 为啥用Dapper, 上面回答出来, 杠杠的。。。。。。。。
- 面试官: 我靠, 小伙子懂的挺多。。。。。。。。
- 

在超过500次poco serialization的过程中所表现的性能, 我们发现dapper是第二名, 当然第一名谁也无法超越, 越底层的当然久越快, 同时也就越麻烦。

About

昵称: [码代码_net](#)
园龄: [3年2个月](#)
粉丝: [5](#)
关注: [13](#)
[+加关注](#)

SEARCH

最新评论

日历							随笔档案	
< 2018年7月 >							2017年9月(7)	
日	一	二	三	四	五	六	2017年8月(17)	
24	25	26	27	28	29	30	2016年11月(3)	
1	2	3	4	5	6	7	2015年5月(2)	
8	9	10	11	12	13	14		
15	16	17	18	19	20	21		
22	23	24	25	26	27	28		
29	30	31	1	2	3	4		

随笔分类

- [C#\(4\)](#)
- [Log日志\(3\)](#)
- [ORM系列\(5\)](#)
- [RabbitMQ\(10\)](#)
- [Web Api\(1\)](#)
- [代码托管器\(1\)](#)
- [关键术语\(1\)](#)
- [数据库\(1\)](#)
- [网络编程\(1\)](#)

推荐排行榜

1. [C#string类型总结\(1\)](#)
2. [http协议无状态中的 "状态" 到底指的是什么? ! \(1\)](#)
3. [MySQL-时间\(time、date、datetime、timestamp和year\)\(1\)](#)
4. [Dapper系列之一：Dapper的入门\(多表批量插入\)\(1\)](#)
5. [Log4net系列一：Log4net搭建之文本格式输出\(1\)](#)

阅读排行榜

1. [MySQL-时间\(time、date、](#)

Performance of SELECT mapping over 500 iterations - POCO serialization

Method	Duration	Remarks
Hand coded (using a SqlDataReader)	47ms	
Dapper ExecuteMapperQuery	49ms	
ServiceStack.OrmLite (QueryById)	50ms	
PetaPoco	52ms	
BLToolkit	80ms	Can be faster
SubSonic CodingHorror	107ms	
NHibernate SQL	104ms	
Linq 2 SQL ExecuteQuery	181ms	
Entity framework ExecuteStoreQuery	631ms	

看看性能对比

[datetime_timestamp和year\)\(4059\)](#)

[2. RabbitMQ五: 生产者--队列--多消费者\(2892\)](#)

[3. Dapper系列之一: Dapper的入门\(多表批量插入\)\(2579\)](#)

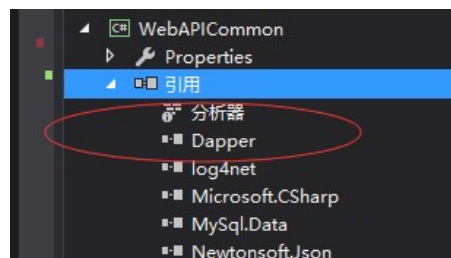
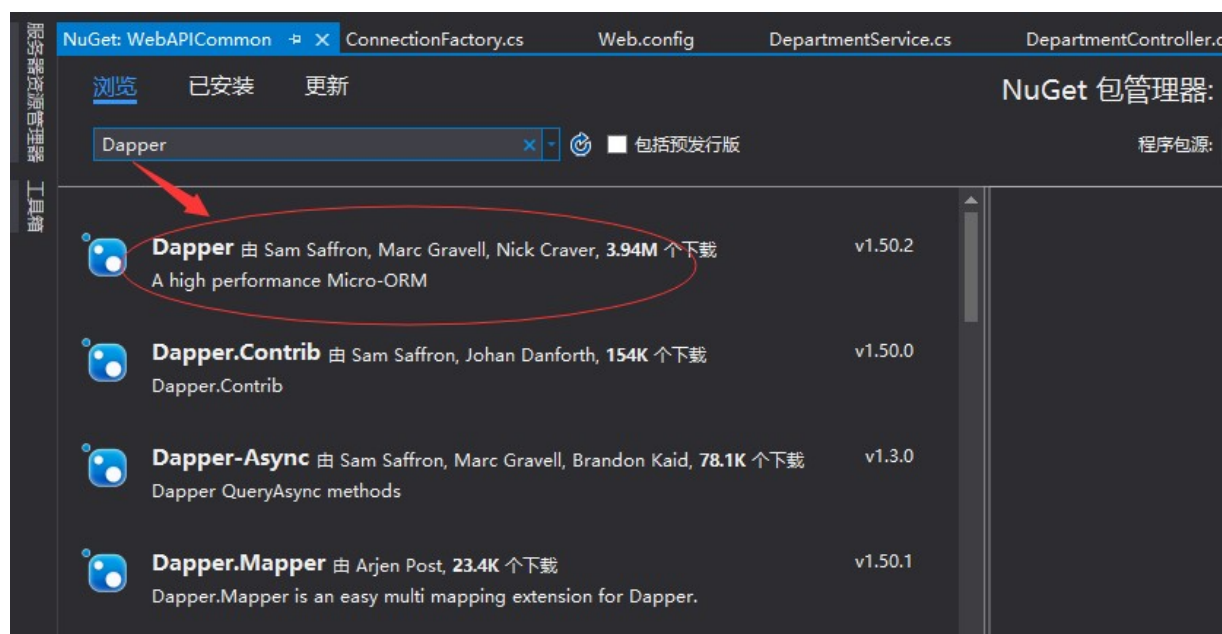
[4. SVN中如何去除版本控制器\(1557\)](#)

[5. C#string类型总结\(1487\)](#)

Dapper代码应用

第一步:

在NuGet中引用Dapper



第二步:

新建一个ConnectionFactory类, 创建链接对象, 这里我们封装两个方法分别获

取SQLServer和MySQL

```
public class ConnectionFactory
{
    //获取web 中的配置文件
    private static readonly string QlwMysqlConnection = ConfigurationManager.AppSettings["sqlconnectionString"];
    /// <summary>
    /// sqlServer 数据库
    /// </summary>
    /// <returns></returns>
    public static IDbConnection SqlServerConnection()
    {
        string sqlconnectionString = QlwMysqlConnection; // ConfigurationManager.ConnectionStrings["sqlconnectionString"].ToString();
        var connection = new SqlConnection(sqlconnectionString);
        if (connection.State == ConnectionState.Closed)
        {
            connection.Open();
        }
        return connection;
    }
    /// <summary>
    /// mySQL 数据库
    /// </summary>
    /// <returns></returns>
    public static IDbConnection MySqlConnection()
    {
        string mysqlconnectionString = QlwMysqlConnection; // ConfigurationManager.ConnectionStrings["mysqlconnectionString"].ToString();
        var connection = new MySqlConnection(mysqlconnectionString);
        if (connection.State == ConnectionState.Closed)
        {
            connection.Open();
        }
        return connection;
    }
}
```

第三步:

- (1) 先看一下后台: SqlMapper, 封装了给我们提供了那些方法:

```

1  程序集 Dapper, Version=1.50.2.0, Culture=neutral, PublicKeyToken=null
4
5  using ...
13
14 namespace Dapper
15 {
16     ...public static class SqlMapper
17     {
20         ...public static Func<Type, ITypeMap> TypeMapProvider;
28
29         ...public static IEqualityComparer<string> ConnectionStringComparer { get; set; }
36
37         ...public static event EventHandler QueryCachePurged;
41
42         ...public static void AddTypeHandler(Type type, ITypeHandler handler);
46         ...public static void AddTypeHandler<T>(TypeHandler<T> handler);
50         ...public static void AddTypeHandlerImpl(Type type, ITypeHandler handler, bool clone);
54         ...public static void AddTypeMap(Type type, DbType dbType);
58         ...public static List<T> AsList<T>(this IEnumerable<T> source);
63         ...public static ICustomQueryParameter AsTableValuedParameter(this IEnumerable<SqlDataRecord> list, string typeName = null);
67         ...public static ICustomQueryParameter AsTableValuedParameter(this DataTable table, string typeName = null);
71         ...public static Action<IDbCommand, object> CreateParamInfoGenerator(Identity identity, bool checkForDuplicates, bool removeUnused);
75         ...public static int Execute(this IDbConnection cnn, CommandDefinition command);
82         ...public static int Execute(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int? commandTimeout =
88         default(int?), CommandType? commandType = default(CommandType?));
89         ...public static Task<int> ExecuteAsync(this IDbConnection cnn, CommandDefinition command);
93         ...public static Task<int> ExecuteAsync(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int?
96         commandTimeout = default(int?), CommandType? commandType = default(CommandType?));
97         ...public static IDataReader ExecuteReader(this IDbConnection cnn, CommandDefinition command);
109         ...public static IDataReader ExecuteReader(this IDbConnection cnn, CommandDefinition command, CommandBehavior commandBehavior);
121         ...public static IDataReader ExecuteReader(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int?
132         commandTimeout = default(int?), CommandType? commandType = default(CommandType?));
133         ...public static Task<IDataReader> ExecuteReaderAsync(this IDbConnection cnn, CommandDefinition command);
145         ...public static Task<IDataReader> ExecuteReaderAsync(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int?
    
```

SqlMapper给我们提供好多方法, 你可以自行研究看看, 基本足够满足了我们的需求。。。

(2) 我们根据上面方法加一层, 简单封装, 为了业务更加方便:

先说说添加Insert操作, 我们对Execute方法进行简单的封装:

SqlMapper提供: 两个封装Execute:

```

13
14 namespace Dapper
15 {
16     ...public static class SqlMapper
17     {
20         ...public static Func<Type, ITypeMap> TypeMapProvider;
28
29         ...public static IEqualityComparer<string> ConnectionStringComparer { get; set; }
36
37         ...public static event EventHandler QueryCachePurged;
41
42         ...public static void AddTypeHandler(Type type, ITypeHandler handler);
46         ...public static void AddTypeHandler<T>(TypeHandler<T> handler);
50         ...public static void AddTypeHandlerImpl(Type type, ITypeHandler handler, bool clone);
54         ...public static void AddTypeMap(Type type, DbType dbType);
58         ...public static List<T> AsList<T>(this IEnumerable<T> source);
63         ...public static ICustomQueryParameter AsTableValuedParameter(this IEnumerable<SqlDataRecord> list, string typeName = null);
67         ...public static ICustomQueryParameter AsTableValuedParameter(this DataTable table, string typeName = null);
71         ...public static Action<IDbCommand, object> CreateParamInfoGenerator(Identity identity, bool checkForDuplicates, bool removeUnused);
75         ...public static int Execute(this IDbConnection cnn, CommandDefinition command);
82         ...public static int Execute(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int? commandTimeout =
88         default(int?), CommandType? commandType = default(CommandType?));
89         ...public static Task<int> ExecuteAsync(this IDbConnection cnn, CommandDefinition command);
93         ...public static Task<int> ExecuteAsync(this IDbConnection cnn, string sql, object param = null, IDbTransaction transaction = null, int?
    
```

(3)、创建一个DapperDBContext类

```

public static class DapperDBContext
{
    public static List<T> AsList<T>(this IEnumerable<T> source)
    {
        if (source != null && !(source is List<T>))
            return source.ToList();
        return (List<T>)source;
    }
    //参数我们跟后台封装方法保持一致
    public static int Execute(string sql, object param = null,
        IDbTransaction transaction = null, int? commandTimeout = null,
        CommandType? commandType = null, int databaseOption = 1)
    {
        using (var conn = ConnectionFactory.MySqlConnection())
        {
            return conn.Execute(sql, param, transaction, commandTimeout, commandType, databaseOption);
        }
    }
}
    
```

```

        {
            var info = "SQL语句:" + sql + " \n SQL参数: " + J
sonConvert.SerializeObject(param) + " \n";
            // LogHelper.ErrorLog(info); // 可以记录操作
            var sw = new Stopwatch(); sw.Start();
            var result = conn.Execute(sql, param, transactio
n, commandTimeout, CommandType);
            sw.Stop();
            LogHelper.ErrorLog(info + "耗时:" + sw.ElapsedMil
liseconds + (sw.ElapsedMilliseconds > 1000 ? "####" : string.Empty) + "\n"); // 可以记录操作
            return result;
        }
    }

    public static int Execute(CommandDefinition command, int
databaseOption = 1)
    {
        using (var conn = ConnectionFactory.MySqlConnection())
        {
            {
                var info = "SQL语句:" + command.CommandText + "
\n SQL命令类型: " + command.CommandType + " \n";
                // LogHelper.Info(info); // 可以记录操作
                var sw = new Stopwatch(); sw.Start();
                var result = conn.Execute(command);
                sw.Stop();
                // LogHelper.Info(info + "耗时:" + sw.ElapsedMill
iseconds + (sw.ElapsedMilliseconds > 1000 ? "####" : string.Empty) + "\n"); // 可以记录操作
                return result;
            }
        }
    }
}

```

(4.1) 、单条数据插入:

```

public class DepartmentRepository
{
    /// <summary>
    /// 插入单条数据以及多条数据
    /// </summary>
    /// <param name="department"></param>
    /// <returns></returns>
    public bool Add(List<Department> department, AuthResource
s authResources)
    {
        #region 插入单条数据
        string sql = @" INSERT INTO Department (ID,EID,Name,R
emarks,Description,Notice,ParentId,AddTime,IsDel,UpdateTime)
                        VALUES (@ID,@EID,@Name,@Remarks,@Des
cription,@Notice,@ParentId,@AddTime,@IsDel,@UpdateTime) ";
        var result = DapperDBContext.Execute(sql, department[
0]);
        return result >= 1;
        #endregion
    }
}

```

```
}
// ...
```

```
<boolean xmlns="http://schemas.microsoft.com/2003/10/Serialization/">true</boolean>
```

100 %

结果 消息

数据库添加一条, 返回时true

	ID	EID	Name	Remarks	Description
1	9C5D22AA-394C-44A3-AAD8-00369E1F5B86	6C7D87E5-6C3B-471C-9C55-4E615A80748C	部门		

(4.2)、单表批量数据插入:

```
// department是100条数据
public bool Add(List<Department> department, AuthResources authResources)
{
    #region 插入单条数据
    string sql = @" INSERT INTO Department (ID,EID,Name,Remarks,Description,Notice,ParentId,AddTime,IsDel,UpdateTime)
VALUES (@ID,@EID,@Name,@Remarks,@Description,@Notice,
@ParentId,@AddTime,@IsDel,@UpdateTime); ";
    var result = DapperDBContext.Execute(sql, department);
    //直接传送list对象
    return result >= 1;
    #endregion
}
```

THIS XML file does not appear to have any style information associated with it. The document

```
<boolean xmlns="http://schemas.microsoft.com/2003/10/Serialization/">true</boolean>
```

	ID	EID	Name	Remarks	Description	Not
1	9C5D22AA-394C-44A3-AAD8-00369E1F5B86	6C7D87E5-6C3B-471C-9C55-4E615A80748C	部门			
2	ECEDB551-D91D-4079-9E07-03C1452961CF	9A6F99C5-C59E-4295-B6E5-415294E98D95	部门			
3	32DA597A-3D8C-478B-9DC7-05CB52CACFCD	34907A56-A959-4F28-AF9F-5F74D0704D0F	部门			
4	1043FDA7-356A-49B7-B86C-07489682251F	C835671B-FD03-42AD-A11C-E0191DCE324F	部门			
5	4A5496F0-BAA7-4CE2-AD0B-09AD2A638BEE	9E1E2E4B-1824-41AB-94AE-D3911740A7F8	部门			

查询已成功执行。 (local)\SA (12.0 RTM) | zqs (53) | AuthPlatform | 00:00:00 | 101 行

(4.3)、多表多数据批量插入:

这里我们采用事物, 事物本身有两个特有特性: 原子性和统一性, 比如: 向ABC三个表同时插入, 只要有个插入有误都失败, 如果不采用事物, 采用纯sql插入可能出现数据不一致, AB成功, C失败。

那我们在DapperDBContext中继续封装一个事物的方法, 不知道你现在有没有体会到, 我们为啥在中间一层, 为了我们根据业务的扩展而却要。

方法可以自己扩展, 根据自己业务需要去延伸。。。。。

```
// ...
```



```

    /// <summary>
    /// 多表操作--事务
    /// </summary>
    /// <param name="trans"></param>
    /// <param name="databaseOption"></param>
    /// <param name="commandTimeout"></param>
    /// <returns></returns>

    public static Tuple<bool, string> ExecuteTransaction(List
<Tuple<string, object>> trans, int databaseOption = 1, int? comma
ndTimeout = null)
    {
        if (!trans.Any()) return new Tuple<bool, string>(fals
e, "执行事务SQL语句不能为空! ");
        using (var conn = ConnectionFactory.MySqlConnection()
)
        {
            //开启事务
            using (var transaction = conn.BeginTransaction())
            {
                try
                {
                    var sb = new StringBuilder("ExecuteTransa
ction 事务: ");

                    foreach (var tran in trans)
                    {
                        sb.Append("SQL语句:" + tran.Item1 + "
\n SQL参数: " + JsonConvert.SerializeObject(tran.Item2) + " \n");
                        // 根据业务添加纪录日志 LogHelper.InfoLo
g("SQL语句:" + tran.Item1 + " \n SQL参数: " +
                            JsonConvert.SerializeObject(tran.I
tem2) + " \n");

                        //执行事务
                        conn.Execute(tran.Item1, tran.Item2,
transaction, commandTimeout);
                    }
                    var sw = new Stopwatch();
                    sw.Start();
                    //提交事务
                    transaction.Commit();
                    sw.Stop();
                    // 根据业务添加纪录日志 LogHelper.InfoLog(sb
.ToString() + "耗时:" + sw.ElapsedMilliseconds + (sw.ElapsedMilli
seconds > 1000 ?

                    "####" : string.Empty) + "\n");
                    return new Tuple<bool, string>(true, stri
ng.Empty);
                }
                catch (Exception ex)
                {
                    //todo:!!!transaction rollback can not wo
rk.

                    LogHelper.ErrorLog(ex);
                    //回滚事务
                    transaction.Rollback();
                    conn.Close();
                    conn.Dispose();
                    return new Tuple<bool, string>(false, ex.
ToString());
                }
            }
        }
    }
}
finally

```

```

        {
            conn.Close();
            conn.Dispose();
        }
    }
}

```



方法中用到的Tuple（元组）方法我们就不做介绍，后期我会整理一篇专门介绍元组的方法以及一些新的特性。**事物同样可以满足一个表多条数据插入**



```

public bool Add(List<Department> department, AuthResources authResources)
{
    #region 事务：元组形式插入多条数据
    var param = new List<Tuple<string, object>>();
    Tuple<string, object> tuple;
    var sw = new Stopwatch();
    sw.Start();
    for (int i = 0; i < 100; i++)
    {
        tuple = new Tuple<string, object>(@" INSERT INTO
Department (ID,EID,Name,Remarks,Description,Notice,ParentId,AddTime,IsDel,UpdateTime) VALUES (@ID,@EID,@Name,@Remarks,@Description,
@Notice,@ParentId,@AddTime,@IsDel,@UpdateTime) ", new
        {
            ID = Guid.NewGuid(),
            EID = Guid.NewGuid(),
            Name = "部门",
            Remarks = "",
            Description = "",
            AddTime = DateTime.Now,
            IsDel = 0,
            UpdateTime = DateTime.Now,
            ParentId = Guid.NewGuid(),
            Notice = "",
        });
        param.Add(tuple);
    }
    tuple = new Tuple<string, object>(@" INSERT INTO Auth
Resources (ID,EID,AuthId,ResourceId,AddTime,IsDel,UpdateTime) VALUES (@ID,@EID,@AuthId,@ResourceId,@AddTime,@IsDel,@UpdateTime) ",
new
    {
        ID = Guid.NewGuid(),
        EId = Guid.NewGuid(),
        AuthId = Guid.NewGuid(),
        ResourceId = Guid.NewGuid(),
        AddTime = DateTime.Now,
        IsDel = 0,
        UpdateTime = DateTime.Now,
    });
    param.Add(tuple);
    //调用上面我们封装的事物方法：ExecuteTransaction
    var result = DapperDBContext.ExecuteTransaction(param
).Item1;

    sw.Stop();
    return result;
}

```



```

        #endregion
    }
}

```

结果:

.....

```

<boolean xmlns="http://schemas.microsoft.com/2003/10/Serialization/">true</boolean>

```

.....

100 %

结果 消息

ID	EID	Name	Remarks	Description
1	9C5D22AA-394C-44A3-AAD8-00369E1F5B86	6C7D87E5-6C3B-471C-9C55-4E615A80748C	部门	
2	C4263224-22FF-4822-AEBC-028F1D82F4B5	2BF61DF6-3C81-4E9F-B108-0EF52EED3C83	部门	
3	QD8BC750-0D69-4D1D-83D2-034854A95D35	68A42841-E39A-4580-8296-5EEB5C485675	部门	
4	ECFDBF51-041D-4079-9F07-03C1452961CF	9A6F99C5-C59E-4295-B6E5-41F294E98D95	部门	

ID	EId	AuthId
1	EOCDCDB2-E9E1-4F8D-83E6-845F0B95620A	199E3C04-180A-4E54-A3C2-028C0D5BBAC4

查询已成功执行。 (local)\SA (12.0 RTM) zqs (53) AuthPlatform 00:00:00 302 行

总结: (Dapper 还没结束, 下篇正在书写中。.....)

1、插入的三种方式就结束了, 如果你有更好的方法, 欢迎下方留言, 一起讨论, 本文有不对的方法也多多指出;

2、在做的过程中, 还百度相关资料, 无意中发现数据遍历可以不用for和foreach,

可以用 Enumerable.Range, 已测试性能, 很不错, Dapper写完建立新的博客讲解。.....

手动写东西, 你会发现好多问题, 大家一起边学习边总结, 虽然写博客很费事, 但是确实能收入不少东西, 最起码你写的得到别人认可,

看得懂, 如果还能推荐一下, 心理满满的小激动, 哈哈哈哈哈, 好比: 你学外语, 不去跟别人说, 沟通, 怎么知道学的怎么样, 博客园就是这个平台.....

- 博主是利用读书、参考、引用、抄袭、复制和粘贴等多种方式打造成自己的纯镀 24k 文章, 请原谅博主成为一个无耻的文档搬运工!
- 小弟刚迈入博客编写, 文中如有不对, 欢迎用板砖扶正, 希望给你有所帮助。

好文要顶

关注我

收藏该文



代码码_net

关注 - 13

粉丝 - 5

+加关注

1

0

« 上一篇: [Log4net系列二: Log4net邮件日志以及授权码](#)

» 下一篇: [Dapper系列之二: Dapper的事务查询](#)

分类: [ORM系列](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!

【推荐】如何快速搭建人工智能应用?

【大赛】2018首届“顶天立地”AI开发者大赛



最新IT新闻:

- 智利法院裁定银行需要为加密货币交易所开设银行账户
- 倪光南: 投资芯片产业不要期待一两年就取得回报
- 比亚迪: 比亚迪及子公司印章并未出借给李娟或遗失
- 小米即将被纳入“恒生指数”? 错了
- 3D打印技术使移动工厂成为现实

» [更多新闻...](#)



最新知识库文章:

- 危害程序员职业生涯的三大观念
- 断点单步跟踪是一种低效的调试方法
- 测试 | 让每一粒尘埃有的放矢
- 从Excel到微服务
- 如何提升你的能力? 给年轻程序员的几条建议

» [更多知识库文章...](#)