doi:10.3969/j.issn.1672-4348.2018.04.016

一种优化的 ADO.NET 通用数据库操纵引擎

王晨阳

(福建工程学院 信息科学与工程学院,福建 福州 350118)

摘要:提出一种优化的 ADO.NET 通用数据库操纵引擎 OptimizedDbEngine。该引擎结合多层模式系统的优点,向应用程序提供一致的数据访问接口,同时能自动构造底层繁琐的 SQL 语句,大大简化了应用程序对数据库的操纵。在数据的一致性和安全性方面,支持事务处理,集成数据操纵的日志记录且能有效防止 SQL 注入式攻击。

关键词:ADO.NET;数据库操纵引擎;多层模式;数据操纵日志

中图分类号:TP311 **文献标志码**:A **文章编号**:1672-4348(2018)04-0386-05

An optimized general database manipulation engine based on ADO.NET

WANG Chenyang

(School of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China)

Abstract: An optimized ADO.NET general database manipulation engine OptimizedDbEngine was proposed. This engine combines the advantages of multi-layer model systems, provides a consistent data access interface to the application program, and can automatically construct the underlying complex SQL statements, which greatly simplifies database manipulation in application. In terms of data consistency and security, it supports transaction processing, integrates data manipulation logging and effectively prevents SQL injection attacks.

Keywords: ADO.NET; database manipulation engine; multi-level model; data manipulation logging

在开发一款企业级应用程序,不管是桌面应用程序、WEB 应用程序还是移动端 APP 应用程序,对数据库的封装访问都是其底层的核心模块之一。随着 C#语言的流行,越来越多的应用程序在与数据库的通信上采用 ADO.NET 技术。ADO.NET 是一组允许和不同类型的数据源以及数据库进行交互的面向对象类库,采用 XML 作为通用的数据传送格式,在跨异种环境通信情况下具有极好的互操作性。其次,ADO.NET 具有十分强大的可伸缩性,它采用非连接数据集,数据在本地缓存,等要更新的时候再回传数据库,这使得 WEB 应用程序可以为更多的用户同时提供服务。本文在 ADO.NET 的基础上,实现了一种通用的数据库操纵引擎。

1 ADO.NET 的体系结构

ADO.NET 是微软公司新一代.NET 数据库的访问架构,它是数据库应用程序和数据源之间沟通的桥梁,主要提供一个面向对象的数据访问架构[1]。ADO.NET 主要由 Data Provider 和 DataSet 两大组件组成[2]。.NET Data Provider 针对不同类型的数据源,细分为 SQL Server .NET Data Provider,用于 Microsoft SQL Server 数据源;OLE DB .NET Data Provider 用于 OLE DB 公开的数据源;ODBC .NET Data Provider 用于 ODBC 公开的数据源;Oracle .NET Data Provider 用于 Oracle 数据源。

收稿日期:2018-07-10
基金项目:福建省教育厅项目(JA15342)
作者简介:王晨阳(1984-),男,福建莆田人,讲师,硕士,研究方向:大数据技术、电商推荐系统、搜索引擎技术。

.NET Data Provider 包含 4 个核心对象，Connection 对象用于建立与数据源的连接；Command 对象用于执行数据源操纵命令；DataReader 对象用于从数据源返回一个仅向前的只读数据流；DataAdapter 对象用于充当 DataSet 对象与数据源之间的桥梁。DataSet 对象是创建在内存中的集合对象，不依赖于数据库的独立数据集，即使数据库连接断开，它依然可用。ADO.NET 的体系架构如图 1 所示。

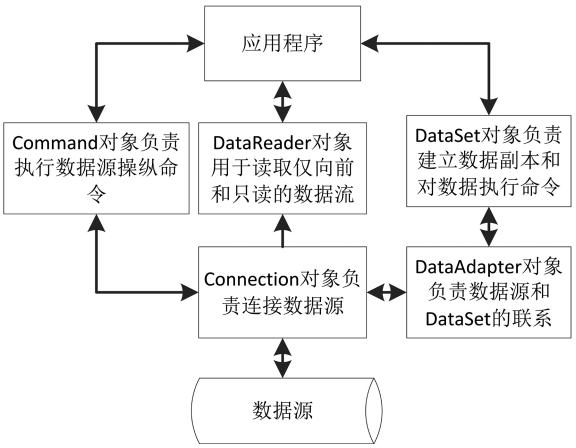

图 1　ADO.NET 体系架构

Fig.1　ADO.NET architecture

2　ADO.NET 的改进研究

虽然 ADO.NET 具有互操作性、可伸缩性、性能高等优点，但是也存在以下不足：

(1) 每次访问数据库 (如：SQL Server 数据库) 都要创建 SqlConnection 对象，创建一个 SqlCommand 对象并编写具体的 SQL 语句，而 SQL 语句是程序员最容易拼错的地方，并且很多程序员会因一时疏忽把表单的内容直接拼接到 SQL 语句，这样会造成系统存在 SQL 注入式攻击漏洞，攻击者可以把恶意的 SQL 命令插入到表单从而让服务器执行恶意的 SQL 语句。

(2) 系统中会到处分散着对 ADO.NET 相关类的调用和数据访问的代码，这对调试代码增加了许多工作量。

(3) 没有提供数据库操纵日志记录功能，要记录一些数据库的重要操作，如 delete 或 update 操作，在每次调用时必须添加日志记录代码。

针对以上不足，对 ADO.NET 二次封装进行改进研究。如文献[3-4]提出了将 ADO.NET 与多层模式相融合的方法，在数据访问层构建独立的 SQL 引擎组件，把对 ADO.NET 相关类的调用集中在数据访问层，方便后期的代码维护。但是它并不能实现自动构造 SQL 语句，也不能有效防止 SQL 注入式攻击。文献[5]提出的 ADO.NET 通用数据库引擎，虽然可以自动构造 SQL 语句，但是并没有提供对事务处理的支持，而且 Update 方法只有一个 Parameters 参数，当要更新的域字段的名称和条件字段域的名称一样的时候，无法实现。

3　优化的 ADO.NET 通用数据库操纵引擎的设计

针对以上不足，本文提出一种优化的 ADO.NET 通用数据库操纵引擎 OptimizedDbEngine，提供以下 6 个方面的功能：(1) 能够自动构造 SQL 语句；(2) 每次执行对数据库的操作，自动创建数据库的连接，执行完后自动释放连接对象；(3) 以统一的方式从数据库获取检索结果，并支持分页；(4) 支持数据库事务处理；(5) 能有效防止 SQL 注入式攻击；(6) 提供数据库操作日志记录。

OptimizedDbEngine 主要有 3 个类：BaseDAO (自动构造 SQL 语句并向业务层提供数据库操纵接口)、DBUtil(向数据库发送操纵命令)、DataSetForPage(支持分页的数据集)。它们的关系如图 2 所示。

3.1　BaseDAO 类

BaseDAO 类向业务层提供 Insert 和 InsertWithTrans、Update 和 UpdateWithTrans、Delete 和 DeleteWithTrans、Select 方法实现对数据库的增、改、删、查操作，其中 InsertWithTrans、UpdateWithTrans 和 DeleteWithTrans 方法是带事务处理。Select 方法用于执行检索操作，是重载方法，第 2 个 Select 方法是带分页的检索。RunProcedure 和 RunProcedureWithTrans 方法用于执行存储过程，后者带事务处理。BeginTransaction、CommitTransaction 和 RollbackTransaction 方法用于控制事务。Log 方法是一个保护方法，用于记录数据库操纵的日志，日志的记录主要用到开源的 log4net.dll 组件。

对数据库的增、改、删、查的具体 SQL 语句是自动构造出来，不需要程序员在业务层中编写。构造 SQL 语句的原理是基于 htFieldValue 和 htParamValue 这两个 Hashtable 类型的字段。ht-

FieldValue 用于存储要更新的字段域的名称和值的键值对;htParamValue 用于存储条件字段域的名称和值的键值对。其中 Update、Select 方法需用到 htFieldValue 和 htParamValue 两个字段,Insert 方法只需用到 htFieldValue 字段,Delete 方法只需用到 htParamValue 字段。因为 SQL 中 update 的语法格式为:update tableName set field_column1 = field_value1,field_column2 = field_value2,… where param_column1 = param_value1 and param_column2 = param_value2 and …;所以构造 update 语句的重点在于自动构造出 set 子句和 where 子句,这两个字符串分别由若干<field_column,field_value>和<param_column,param_value>键值对组成,可从 htFieldValue 和 htParamValue 自动生成。insert 语句的语法格式为:insert into tableName (field_column1,field_column2,…) values (field_value1,field_value2,…);语句里的<field_column,field_value>键值对可从 htFieldValue 自动生成。delete 语句的语法格式为:delete from tableName where param_column1 = param_value1 and param_column2 = param_value2 and …;语句里的<param_column,param_value>键值对可从 htParamValue 自动生成。select 语句的语法格式为:select field_column1,field_column2,… from tableName where param_column1 = param_value1 and param_column2 = param_value2 and …;语句里的<param_column,param_value>键值对可从 htParamValue 自动生成,<field_column>列表可从 htFieldValue 自动生成。

下面给出 Update 方法中自动构造 SQL 语句的具体步骤以及核心代码,在其它方法中构造 SQL 语句类似。

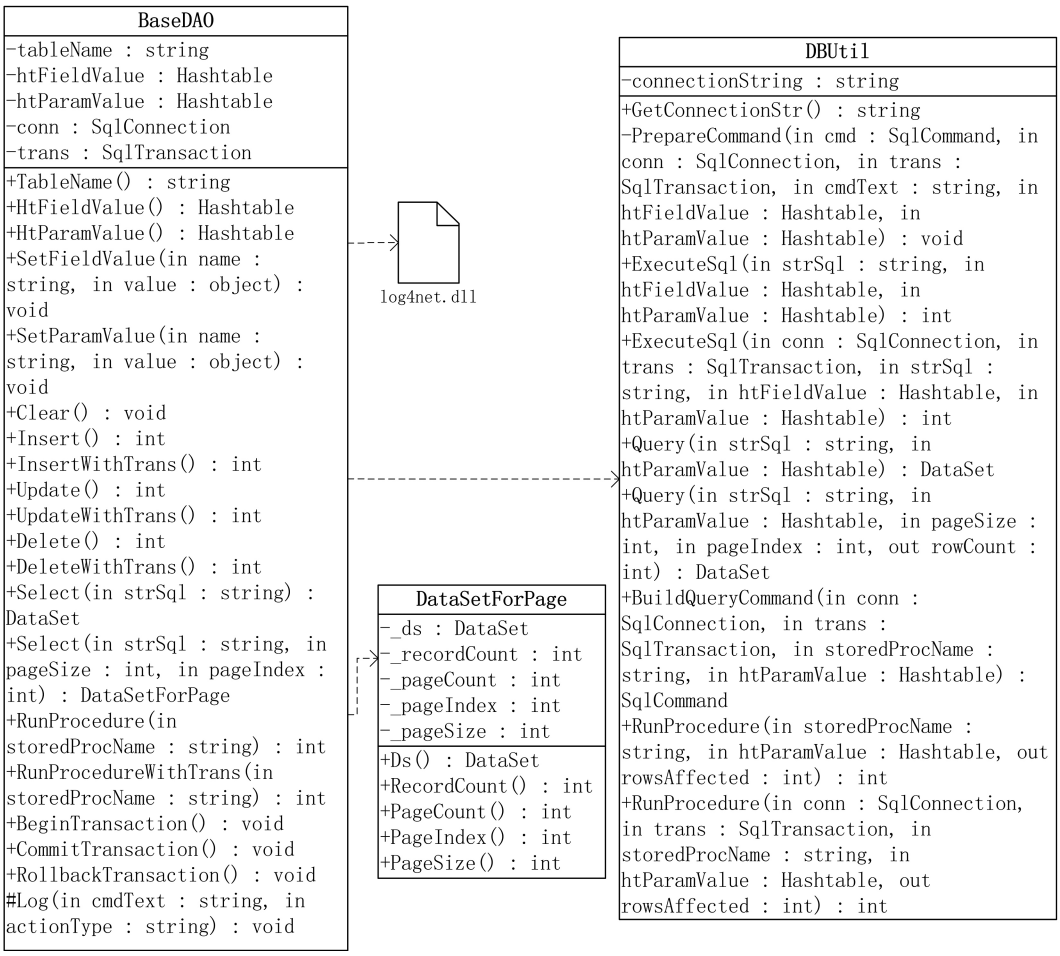

图 2 OptimizedDbEngine 的类图

Fig.2 Class diagram of OptimizedDbEngine

(1)遍历存储在 htFieldValue 中的键集合,构造 set 子句,具体代码如下。

```
String sqlSet = "";
foreach (DictionaryEntry de in htFieldValue)
{
if (! (String.IsNullOrEmpty(sqlSet))) sqlSet = sqlSet + ",";
sqlSet += de.Key + "=" + "@f_" + de.Key;
}
```

这里的参数名称加个前缀“f_”是为了区分 htFieldValue 和 htParamValue 中的参数,因为在 htFieldValue 和 htParamValue 中可能存在相同名称的参数。

(2)遍历存储在 htParamValue 中的键集合,构造 where 子句,具体代码如下。

```
String sqlWhere = "";
foreach (DictionaryEntry de in htParamValue)
{
if (! (String. IsNullOrEmpty (sqlWhere))) sqlWhere = sqlWhere + " and ";
sqlWhere += de. Key + "=" + "@" + de.Key;
}
```

(3)拼接 set 子句和 where 子句,构造出完整的 update 语句,具体代码如下。String sql = "update " + tableName + " set " + sqlSet + " where " + sqlWhere ;

(4)然后调用 DBUtil 类的 ExecuteSql 方法,传递的参数为(sql, htFieldValue, htParamValue),接着调用 Log 方法记录该操作日志,最后调用 Clear 方法清空存储在 htFieldValue 和 htParamValue 的所有元素。

从上面 Update 方法中构造 SQL 语句的具体代码中可以看出,最后传递给 DBUtil 类的 ExecuteSql 方法的 sql 语句是采用参数的形式,并没有把参数具体的值(具体的值往往是接收用户表单提交的数据,存在 SQL 注入式攻击风险)直接拼接在 SQL 语句中,这样可以有效防止 SQL 注入式攻击。并且 Update 方法最后会调用 Log 方法进行日志记录,不需要程序员在业务层显式编写代码进行数据库操纵日志记录。

3.2　DBUtil 类

数据库通信和操纵的具体实现是封装在 DBUtil 类中。DBUtil 类中有一个属性:connectionString,读取存储在配置文件中的数据库连接字符串。并向 BaseDAO 类提供 ExecuteSql、Query、RunProcedure 方法,这 3 个方法执行时会根据属性 connectionString 的值自动创建用于数据库连接的 SqlConnection 对象,并在方法返回前会断开数据库连接。

(1)ExecuteSql 方法,主要用来执行增、删、改操作,是 1 个重载方法。第 1 个 ExecuteSql 方法接受 3 个参数,第 1 个参数 strSql 是要执行数据库操纵的带参数的 SQL 语句;第 2 个参数 htFieldValue 是 1 个 Hashtable 类的对象,用于存储操作字段域的名称和值的键值对;第 3 个参数 htParamValue 也是 1 个 Hashtable 类的对象,用于存储条件字段域的名称和值的键值对。方法内会先创建 1 个 SqlCommand 对象;接着调用 PrepareCommand 私有方法,PrepareCommand 方法主要根据 htFieldValue 和 htParamValue 参数设置 SqlCommand 对象的 Parameters 属性;然后执行 SqlCommand 对象的 ExecuteNonQuery 方法,完成数据库的操纵。

第 2 个 ExecuteSql 方法共接受 5 个参数,多接受 1 个 SqlConnection 对象的参数和 1 个 SqlTransaction 对象的参数,用于实现带事务处理的操作。

(2)Query 方法,主要用来执行查询操作,是 1 个重载方法。第 1 个 Query 方法接受两个参数,第 1 个参数 strSql 是要执行查询操作的 SQL 语句;第 2 个参数 htParamValue 是 1 个 Hashtable 类的对象,用于存储查询条件字段域的名称和值的键值对。方法内会先创建 1 个 SqlDataAdapter 对象,再创建一个 SqlCommand 对象作为 SqlDataAdapter 对象的 SelectCommand 属性;接着调用 PrepareCommand 私有方法,PrepareCommand 方法会根据 htParamValue 参数设置 SqlCommand 对象的 Parameters 属性;然后执行 SqlDataAdapter 对象的 Fill 方法,完成数据库的查询,返回 1 个查询结果集 DataSet 对象。

第 2 个 Query 方法实现分页查询,共接受 5 个参数,多接受了 3 个整型参数 pageSize、pageIndex、rowCount, pageSize 参数用于指定每页记录数,pageIndex 参数用于指定页码,rowCount 是一个输出参数,用于返回总共有多少条记录。分页

检索的原理就是自动构造出如下SQL语句。

select top pageSize * from (select row_number() rowNum,field_column1,field_column2,... from tableName where param_column1 = param_value1, param_column2 = param_value2,...) t1 where rowNum>((pageIndex-1) * pageSize) order by rowNum。

(3)RunProcedure方法,用来执行存储过程,是1个重载方法。第1个RunProcedure方法接受3个参数,第1参数storedProcName用于指定存储过程的名称,第2个参数htParamValue是一个Hashtable类的对象,用于存储参数的名称和值的键值对,第3个参数rowsAffected是1个输出参数,用于返回受影响的行数。

第2个RunProcedure方法共接受5个参数,多接受一个SqlConnection对象的参数和一个SqlTransaction对象的参数,用于实现带事务处理的存储过程。

3.3 DataSetForPage类

DataSetForPage类是对ADO.NET体系架构中的DataSet对象的进一步封装,用于接收带分页查询的结果,有5个可以读写的属性,_ds是1个DataSet对象,用于存储数据结果集;_recordCount是1个整型对象,用于存储总记录数;_pageCount是1个整型对象,用于存储总页数;_pageIndex是1个整型对象,用于存储当前页码;_pageSize是1个整型对象,用于存储每页记录数。

3.4 在业务层使用OptimizedDbEngine

通用数据库引擎以DLL的形式发布,在业务层只要引用OptimizedDbEngine.dll即可。下面给出1个简单的应用示例。如在数据库里有一张表student,它有3个字段num(学号),name(姓名),age(年龄)。现要往student表插入1条记录(1001,"张三",20),具体的代码如下:

```
BaseDAO dao = new BaseDAO();
dao.TableName = "student";
dao.SetFieldValue("num",1001);
dao.SetFieldValue("name","张三");
dao.SetFieldValue("age",20);
dao.Insert();
```

如要把学号为1001的学生记录的年龄修改为21,具体的代码如下:

```
dao.SetFieldValue("age",21);
dao.SetParamValue("num",1001);
dao.Update();
```

从以上示例代码可以看出,使用OptimizedDbEngine.dll操纵数据库,完全屏蔽了底层的数据库访问技术,大大降低了应用程序访问数据库的复杂性。

4 结语

数据库访问在绝大多数应用系统里占据着举足轻重的地位,而编写SQL语句是一项繁琐而又重复的工作。OptimizedDbEngine提供了简单易用的接口供应用程序业务层调用,大大降低了数据库访问的门槛,即使从来没用过ADO.NET的程序员,也可以快速开发出高质量的数据库应用程序。OptimizedDbEngine.dll在笔者的多个实际项目中使用并取得很好的效果。

参考文献:

[1] 胡学钢. C#应用开发与实践[M]. 北京: 人民邮电出版社, 2013: 178-179.
[2] 罗福强, 杨剑, 张敏辉. C#程序设计经典教程[M]. 北京: 清华大学出版社, 2012: 250-267.
[3] 孙仁鹏. ADO.NET在多层模式下应用的研究[J]. 计算机工程与设计, 2010, 31(16): 3621-3624.
[4] 王净, 战凯. 基于ADO.NET的通用数据访问组件的实现[J]. 计算机与数字工程, 2010, 38(12): 162-166.
[5] 高英, 郭荷清. 基于改进的ADO.NET的通用数据库引擎的设计与实现[J]. 计算机应用, 2005, 25(1): 35-38.
[6] 徐宝林. 基于ADO.NET的应用程序访问后台数据的模块划分研究[J]. 计算机应用与软件, 2011, 28(7): 212-215.
[7] 陈日莉. 基于ADO.NET构建数据库访问中间层类库的研究[J]. 现代电子技术, 2012, 35(12): 24-27.
[8] 刘宝娥. 利用ADO.NET技术开发SQL Server数据库的相关研究[J]. 赤峰学院学报(自然科学版), 2012, 28(3): 46-47.
[9] 王彬, 靳大尉, 郝文宁, 等. 设计模式在数据库访问权限系统中的应用[J]. 计算机应用, 2012, 32(S2): 113-115, 130.
[10] ZHOU X, ZHUO Z H. Research and realization of ADO.NET database access technology[J]. Applied Mechanics & Materials, 2014, 496-500: 1748-1751.

(特约编辑: 黄家瑜)