13590--北极燕鸥

博采众长，信誉卓著

posts - 58, comments - 137, trackbacks - 3, articles - 9

导航

首页
新随笔
联系
订阅
管理

[原]iBatis.Net（C#）系列二：SQL数据映射
Posted on 2013-03-01 09:54 北极燕鸥 阅读(5121) 评论(5) 编辑 收藏

转载请注明 http://www.cnblogs.com/13590/archive/2013/03/01/2938126.html

摘要：本文探讨了iBatis.Net框架的XML数据映射文件各配置节点的含义，并通过CRUD四种对数据库的操作讲解了如何配置数据映射文件和调用方法。

关键词：iBatis.Net；XML；SQL Maps；数据映射

上一节介绍了iBatis.Net的基本情况和运行原理，运行环境中各参数的配置情况。并通过一个实例项目进行了说明。

1 数据映射基础

SQL Maps是这个iBatis.Net框架中最重要的部分，而SQL Maps的核心就在于XML数据映射文件(Data Map XML File)。在XML数据映射文件里可以定义包括要执行各种SQL语句、存储过程、输入参数映射、返回结果映射、缓存机制，并且能通过几种相对比较复杂的配置实现对象之间的关联关系和延迟加载。这也是iBatis.Net区别其它ORM框架而具备更灵活性，更高性能的关键所在。

配置文件可以写得很简单，也可以很复杂。复杂配置文件往往出于更好的设计，更好性能，更好扩展性方面的目的。一个XML数据映射文件主要分为两个部分：模块配置和语句配置。

2 模块配置

2.1 Type Alias节点

定义一个XML数据映射文件中的别名，其好处就是避免过长变量值的反复书写，比如通过typeAlias节点为类"iBatisTest.Domain.Sysuser"定义了一个别名"Sysuser",这样在这个数据映射文件中的其他部分，需要引用"iBatisTest.Domain.Sysuser"类时，只需以其别名替代即可，和SqlMap.config配置文件里面的Alias节点配置一样。定义如：

```
<alias><typeAlias alias="Sysuser" type="iBatisTest.Domain.Sysuser,iBatisTest" /></alias>
```

2.2 cacheModel节点

缓存机制是程序开发中经常讨论的一个话题，在实际的系统开发的过程中，总会存在着这样一类数据，它们更新频率很低，然而使用的频率却非常之高。为了提高系统性能，通常将此类数据装入缓存。iBatis.Net也有自己的缓存系机制，它通过cacheModel节点来配置，具体的定义如下：

```
<cacheModel type="LRU" id="Sysuser-cache" readOnly="true" serialize="false">
<flushInterval hours="24"/>
<flushOnExecute statement="SysuserMap. UpdateSysuser "/>
<property name="CacheSize" value="100"/>
</cacheModel>
```

type：缓存的类型，iBatis.Net中有4种类型，分别为MEMORY、LRU、FIFO、OSCACHE。

其中MEMORY是内存缓存，LRU是使用最近最少使用策略，FIFO是使用先进先出策略，OSCACHE是通过第三方的缓存插件来实现。

id：是cacheModel的一个标识，标识该缓存的名字，供后面设置使用。

readOnly：指缓存的数据对象是只读还是可读写，默认为"true"，即只读，这里的只读并不是意味着数据对象一旦放入缓存中就无法再对数据进行修改。而是当数据对象发生变化的时候，如数据对象的某个属性发生了变化，则此数据对象就将被从缓存中废除，下次需要需重新从数据库读

取数据，构造新的数据对象。而readOnly="false"则意味着缓存中的数据对象可更新。

serialize：是否从缓存中读取同一个对象，还是对象的副本。该参数只有在readOnly为false的情况下才有效， 因为缓存是只读的，那么为不同会话返回的对象肯定是一个，只有在缓存是可读写的时候，才需要为每个会话返回对象的副本。

flushInterval：指定缓存自动刷新的时间，可以为hours、minutes、seconds和milliseconds。需要注意的是，这个间隔时间不是时间到了，在缓存里的信息会自动刷新,而是在间隔时间过后，下次查询将不会从缓存中去取值，而会用SQL去查询，同时将查询的结果更新缓存的值。

flushOnExecute：指定在发生哪些操作时，更新缓存。

property：针对cacheModel的额外的一些属性配置，不同type的cacheModel将会有自己专有的一些property配置，如：

FIFO: <property name="CacheSize" value="100" />

LRU: <property name="CacheSize" value="100" />

MEMORY: <property name="Type" value="WEAK" />

当配置好cacheModel之后就可以在后面的语句中使用了，如：

< statement id="SelectSysuser" resultClass="Sysuser" cacheModel="Sysuser-cache" >

SELECT * FROM DEAN.SYSUSER

< /statement >

2.3 resultMap节点

resultMap从字面理解就是结果集的映射，它将返回的记录与实体对象进行映射，它属于直接映射。如果查询出来的数据集的字段与属性和实体类一致时，我们常用resultClass来替代，它属于隐身映射。通常resultMap比resultClass性能要高。

在使用resultMap时需要注意，如果在resultMap中给出的配置字段，但是返回的数据集却没有返回这个字段，那程序将出抛出异常。相反的，如果返回了一些字段，却没有在resultMap给出配置定义的话，那么那些字段将不会被处理而不会给出任何的提示，相当没有查询出这些字段。具体的定义例子如下：

```
<resultMaps>
<resultMap id="SysuserResult" class="Sysuser">
    <result property="Userid" column="USERID" />
    <result property="Password" column="PASSWORD" />
    <result property="Loginname" column="LOGINNAME" />
    <result property="Sex" column="SEX"/>
    <result property="Birthday" column="BIRTHDAY"/>
    <result property="Idcard" column="IDCARD" />
    <result property="Officephone" column="OFFICEPHONE" />
    <result property="Familyphone" column="FAMILYPHONE" />
    <result property="Mobilephone" column="MOBILEPHONE"/>
    <result property="Email" column="EMAIL"/>
    <result property="Address" column="ADDRESS"/>
    <result property="Zipcode" column="ZIPCODE"/>
    <result property="Remark" column="REMARK" />
    <result property="Status" column="STATUS" />
    <result property="Registerdate" column="REGISTERDATE" />
</resultMap>
</resultMaps>
```

该参数也是查询语句特有的配置项，因为insert，update，delete三种语句的操作是不会返回数据结果集的，也就没有resultMap和resultClass。如果在一个语句配置中同时指定了resultMap和resultClass属性的话，将会优先使用resultMap。同时result Map的使用也是一个实现对象复杂查询功能的重要手段，如：result map的继承，对象的1..1、1..N关系查询。

2.4 ParameterMap节点

参数映射的配置，它是被用来向一个语句提供所需参数的配置。该参数配置节点和resultMaps节点类似。

3 语句配置

语句配置（Mapped Statements）：顾名思义就是映射的语句声明。它是XML数据映射文件的核心，在iBatis.Net框架中真正和数据库打交道的被执行的SQL语句（或存储过程）都必须在这里被显式声明。语句配置可以包含有：statement、select、insert、update、delete和procedure这6种不同的语句类型。其中statement可以包含所有类型的SQL语句（存储过程），它是一个泛泛的语句配置，没特别明确的职责，相反，其它5种类型的语句配置就是专门负责各种不同的SQL语句。下面这张表列出了各种类型的语句的不同职责和调用方法。

Statement类型	属性	子元素	方法
<statement>	id	所有动态元素	insert
	parameterClass		update
	resultClass		delete
	parameterMap		所有的查询方法
	resultMap		
	cacheModel		
<insert>	id	所有的动态元素	insert
	parameterClass	<selectKey>	update
	parameterMap		delete
<update>	id	所有的动态元素	insert
	parameterClass		update
	parameterMap		delete
<delete>	id	所有的动态元素	insert
	parameterClass		update
	parameterMap		delete
<select>	id	所有的动态元素	所有的查询方法
	parameterClass		
	resultClass		
	parameterMap		
	resultMap		
	cacheModel		
<procedure>	id	所有的动态元素	insert
	parameterClass		update
	resultClass		delete
	parameterMap		所有的查询方法

m.....

--CatcherX

resultMap

应用系统操作数据库数据一般有CRUD四种方式，即增加(Create)、查询(Retrieve，重新得到数据)、更新(Update)和删除(Delete)。下面详细介绍一下这四种情况的SQL语句配置及调用方法。

首先在Oracle数据库中新建表DEAN.SYSUSER。表结构如图1

	Name	Code	Data Type	Length
1	登录ID	USERID	number(10,0)	10
2	登录密码	PASSWORD	VARCHAR2(40)	40
3	登录用户名	LOGINNAME	VARCHAR2(40)	40
4	性别	SEX	VARCHAR2(40)	40
5	出生日期	BIRTHDAY	DATE	
6	身份证号	IDCARD	VARCHAR2(18)	18
7	办公电话	OFFICEPHONE	VARCHAR2(40)	40
8	家庭电话	FAMILYPHONE	VARCHAR2(40)	40
9	手机	MOBILEPHONE	VARCHAR2(40)	40
10	Email	EMAIL	VARCHAR2(100)	100
11	通讯地址	ADDRESS	VARCHAR2(200)	200
12	邮编	ZIPCODE	VARCHAR2(20)	20
13	备注	REMARK	VARCHAR2(200)	200
14	状态	STATUS	VARCHAR2(20)	20
15	注册时间	REGISTERDATE	DATE	

图1:SYSUSER表结构

增加实体类iBatisTest.Domain.Sysuser，该实体类和表DEAN.SYSUSER对应。Sysuser类代码如下：

```
using System;

namespace iBatisTest.Domain

{

        [Serializable]

        public sealed class Sysuser

        {

                #region 私有成员定义

                private Int32 _userid;

                private string _password;

                private string _loginname;

                private string _sex;

                private DateTime _birthday;

                private string _idcard;

                private string _officephone;

                private string _familyphone;

                private string _mobilephone;

                private string _email;

                private string _address;

                private string _zipcode;

                private string _remark;

                private string _status;

                private DateTime _registerdate;

#endregion 私有成员定义
```

```
#region 定义默认类结构函数
public Sysuser()
                {
                        _userid = 0;
                        _password = null;
                        _loginname = null;
                        _sex = null;
                        _birthday = new DateTime();
                        _idcard = null;
                        _officephone = null;
                        _familyphone = null;
                        _mobilephone = null;
                        _email = null;
                        _address = null;
                        _zipcode = null;
                        _remark = null;
                        _status = null;
                        _registerdate = new DateTime();
                }
#endregion 定义默认类结构函数

#region 定义公有属性
/// <summary>
                /// 登录ID
                /// </summary>
                public Int32 Userid
                {
                        get { return _userid; }
                        set { _userid = value; }
                }
                /// <summary>
                /// 登录密码
                /// </summary>
                public string Password
                {
                        get { return _password; }
                        set
                        {
                                if( value!= null && value.Length > 40)
                                        throw new ArgumentOutOfRangeException("密
码值错误", value, value.ToString());
                                 _password = value;
```

```
                }
            }
            /// <summary>
            /// 登录用户名
            /// </summary>
            public string Loginname
            {
                get { return _loginname; }
                set
                {
                    if( value!= null && value.Length > 40)
throw new ArgumentOutOfRangeException("登录用户名值错误", value,
value.ToString());
                    _loginname = value;
                }
            }
            /// <summary>
            /// 性别
            /// </summary>
            public string Sex
            {
                get { return _sex; }
                set
                {
                    if( value!= null && value.Length > 40)
throw new ArgumentOutOfRangeException("性别值错误", value, value.ToString());
                    _sex = value;
                }
            }
            /// <summary>
            /// 出生日期
            /// </summary>
            public DateTime Birthday
            {
                get { return _birthday; }
                set { _birthday = value; }
            }
            /// <summary>
            /// 身份证号
            /// </summary>
            public string Idcard
            {
```

```
                get { return _idcard; }
                set
                {
                        if( value!= null && value.Length > 18)
throw new ArgumentOutOfRangeException("身份证号值错误", value,
value.ToString());
                         _idcard = value;
                }
        }
        /// <summary>
        /// 办公电话
        /// </summary>
        public string Officephone
        {
                get { return _officephone; }
                set
                {
                        if( value!= null && value.Length > 40)
throw new ArgumentOutOfRangeException("办公电话值错误", value,
value.ToString());
                        _officephone = value;
                }
        }
        /// <summary>
        /// 家庭电话
        /// </summary>
        public string Familyphone
        {
                get { return _familyphone; }
                set
                {
                        if( value!= null && value.Length > 40)
throw new ArgumentOutOfRangeException("家庭电话值错误", value,
value.ToString());
                         _familyphone = value;
                }
        }
        /// <summary>
        /// 手机
        /// </summary>
        public string Mobilephone
        {
                get { return _mobilephone; }
```

```
                        set
                        {
                                if( value!= null && value.Length > 40)
throw new ArgumentOutOfRangeException("手机号值错误", value, value.ToString());
                                 _mobilephone = value;
                        }
                }
                /// <summary>
                /// Email
                /// </summary>
                public string Email
                {
                        get { return _email; }
                        set
                        {
                                if( value!= null && value.Length > 100)
throw new ArgumentOutOfRangeException("Email值错误", value, value.ToString());
                                 _email = value;
                        }
                }
                /// <summary>
                /// 通讯地址
                /// </summary>
                public string Address
                {
                        get { return _address; }
                        set
                        {
                                if( value!= null && value.Length > 200)
throw new ArgumentOutOfRangeException("通讯地址值错误", value,
value.ToString());
                                 _address = value;
                        }
                }
                /// <summary>
                /// 邮编
                /// </summary>
                public string Zipcode
                {
                        get { return _zipcode; }
                        set
                        {
```

```
                                if( value!= null && value.Length > 20)
throw new ArgumentOutOfRangeException("邮编值错误", value, value.ToString());
                                 _zipcode = value;
                        }
                }
                /// <summary>
                /// 备注
                /// </summary>
                public string Remark
                {
                        get { return _remark; }
                        set
                        {
                                if( value!= null && value.Length > 200)
throw new ArgumentOutOfRangeException("备注值错误", value, value.ToString());
                                 _remark = value;
                        }
                }
                /// <summary>
                /// 状态
                /// </summary>
                public string Status
                {
                        get { return _status; }
                        set
                        {
                                if( value!= null && value.Length > 20)
throw new ArgumentOutOfRangeException("状态值错误", value, value.ToString());
                                 _status = value;
                        }
                }


                /// <summary>
                /// 注册时间
                /// </summary>
                public DateTime Registerdate
                {
                        get { return _registerdate; }
                        set {_registerdate = value; }
}
#endregion          定义公有属性
}
```

```
}
```

(1)添加

添加也即插入，使用的SQL语句是insert。XML数据映射配置信息如下：

```
<insert id="InsertSysuser" parameterClass="Sysuser">
        INSERT INTO DEAN.SYSUSER
(USERID,PASSWORD,LOGINNAME,SEX,BIRTHDAY,IDCARD,OFFICEPHONE,FAMILYPHONE,MOBILEPHONE,EMAIL,ADDRESS,ZIPCODE,REMARK,STATUS)
        VALUES
(#Userid#,#Password#,#Loginname#,#Sex#,#Birthday#,#Idcard#,#Officephone#,#Familyphone#,#Mobilephone#,#Email#,#Address#,#Zipcode#,#Remark#,#Status#)
</insert>
```

insert标签表面该语句是插入语句，id为该配置语句的标识，在后面的调用中通过该标识引用这个语句。变量用#号分隔，如"#Userid#"在运行期会由传入的Sysuser对象的Userid属性填充，注意变量名需区分大小写。调用代码如下：

```
protected void BtnAddUser_Click(object sender, EventArgs e)
{
try
{
iBatisTest.Domain.Sysuser model = new iBatisTest.Domain.Sysuser();//实例化Sysuser对象
model.Userid = 1;//Userid为主键，多次添加需修改该值
model.Password = "123";
model.Loginname ="dean";
model.Sex = "男";
model.Birthday = Convert.ToDateTime("1980-01-01");
model.Idcard ="510200198001014200";
model.Officephone="86279528";
model.Familyphone ="86279528";
model.Mobilephone = "13880980000";
model.Email ="476408321@qq.com";
model.Address = "四川成都";
model.Zipcode = null;
model.Remark = null;
model.Status = "有效";
ISqlMapper mapper = Mapper.Instance(); //得到ISqlMapper实例
mapper.Insert("SysuserMap.InsertSysuser", model);//调用Insert方法
Label1.Text = "添加用户成功";
}
catch (Exception ex)
{
Label1.Text = ex.Message;
}
}
```

调用程序先实例化Sysuser对象并赋值，再得到ISqlMapper的实例，调用该实例的Insert方法。

Insert方法有两个参数，分别是语句名称和参数对象。语句名称就是数据映射文件里面的配置语句id，参数对象是一个object对象，可以传入类、单个值或者哈希表等。

编译程序，点击"添加用户"按钮运行调用程序，系统会成功的向数据库SYSUSER表新增一条记录。查询数据库结果如图2所示：

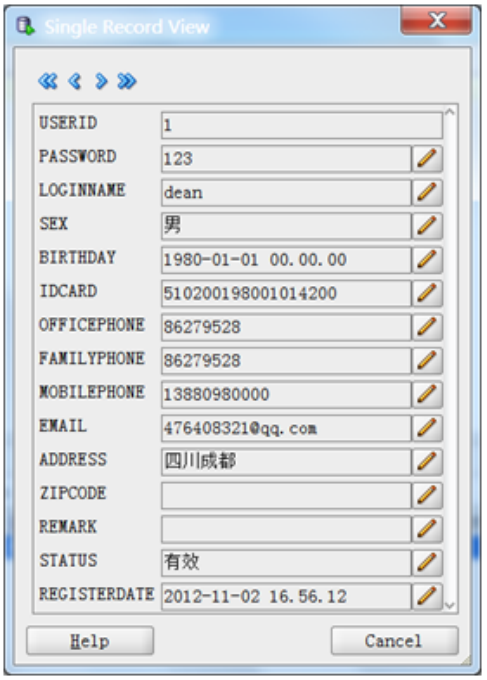

图2：新增数据后数据库结果

(2)更新

更新修改操作使用的SQL语句为update，XML数据映射文件配置信息为：

```
<update id="UpdateSysuser" parameterClass="Sysuser">

UPDATE SYSUSER

SET
PASSWORD=#Password#,LOGINNAME=#Loginname#,SEX=#Sex#,BIRTHDAY=#Birthday#,IDCARD=#Idca
rd#,OFFICEPHONE=#Officephone#,FAMILYPHONE=#Familyphone#,MOBILEPHONE=#Mobilephone#,EM
AIL=#Email#,ADDRESS=#Address#,ZIPCODE=#Zipcode#,REMARK=#Remark#,STATUS=#Status#

WHERE USERID = #Userid#

</update>
```

调用代码为：

```
protected void BtnUpdateUser_Click(object sender, EventArgs e)

{

try

{

iBatisTest.Domain.Sysuser model = new iBatisTest.Domain.Sysuser();

model.Userid = 1;//修改Userid为1的用户信息

model.Password = "1234";

model.Loginname = "deanu";

model.Sex = "男";

model.Birthday = Convert.ToDateTime("1970-01-01");

model.Idcard = "310200198001014200";

model.Officephone = "76279528";

model.Familyphone = "76279528";
```

```
model.Mobilephone = "13880000000";
model.Email = "1@qq.com";
model.Address = null;
model.Zipcode = "610000";
model.Remark = "修改ID为1的用户";
model.Status = "有效";
ISqlMapper mapper = Mapper.Instance(); //得到ISqlMapper实例
mapper.Update("SysuserMap.UpdateSysuser", model);//调用Update方法
Label1.Text = "修改用户成功";
}
catch (Exception ex)
{
Label1.Text = ex.Message;
}
    }
```

(3)删除

删除操作使用的SQL语句为delete，XML数据映射文件配置信息为：

```
<delete id="DeleteSysuser" parameterClass="int">
DELETE FROM SYSUSER WHERE USERID = #Userid#
</delete>
```

由于删除操作是根据主键进行处理的，传入参数为整形值，因此在parameterClass直接为int。调用代码为：

```
protected void BtnDelete_Click(object sender, EventArgs e)
{
//删除用户
try
{
ISqlMapper mapper = Mapper.Instance(); //得到ISqlMapper实例
mapper.Delete("SysuserMap.DeleteSysuser", 1);//调用Delete方法
Label1.Text = "删除用户成功";
}
catch (Exception ex)
{
Label1.Text = ex.Message;
}
    }
```

(4)查询

查询操作使用的语句为select语句，XML数据映射文件配置信息为：

```
<select id="SelectSysuser" parameterClass="int" resultMap="SysuserResult">
SELECT
USERID,PASSWORD,LOGINNAME,SEX,BIRTHDAY,IDCARD,OFFICEPHONE,FAMILYPHONE,MOBILEPHONE,EM
AIL,ADDRESS,ZIPCODE,REMARK,STATUS,REGISTERDATE
FROM SYSUSER WHERE USERID = #Userid#
```

```
</select>
```

调用代码为：

```
protected void BtnQuery_Click(object sender, EventArgs e)
{
//查询用户
try
{
ISqlMapper mapper = Mapper.Instance(); //得到ISqlMapper实例
int Userid = 1;
IList<iBatisTest.Domain.Sysuser> plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("SysuserMap.SelectSysuser",
Userid);//调用QueryForList方法
if (plist != null && plist.Count > 0)
{
gvUserData.DataSource = plist;
gvUserData.DataBind();
}
Label1.Text = "查询用户成功";
}
catch (Exception ex)
{
Label1.Text = ex.Message;
}
}
```

调用ISqlMapper实例的QueryForList方法返回的是一个IList泛型接口，这里指定为iBatisTest.Domain.Sysuser对象。程序运行结果如图3所示。

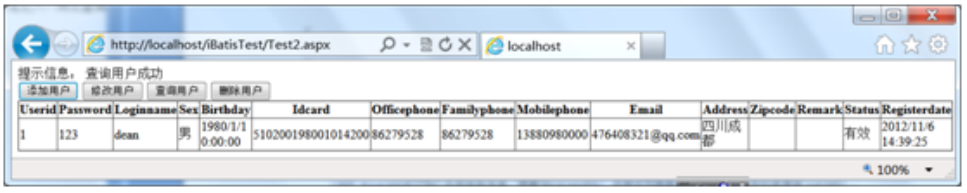

图3：查询结果

以上4种配置语句都可以用statement来代替，其配置信息和调用代码完全一样。但用statement定义所有操作，缺乏直观性。建议在开发中根据实际情况来选用配置，既使得配置文件直观，又可以借助xsd对节点声明进行更有针对性的检查，以避免配置上的失误。

4 特殊配置

4.1 XML转义字符

很多时候在SQL语句中会用到大于或者下于符号（即：><），这个时候就与XML规范相冲突,影响XML映射文件的合法性。通过加入CDATA节点来避免这种情况的发生，如：

```
<statement id="SelectPersonsByAge" parameterClass="int" resultClass="Person">
<![CDATA[
SELECT * FROM PERSON WHERE AGE > #value#
]]>
```

```
</statement>
```

4.2自动生成主键

目前很多数据库都支持为新插入的记录自动生成主键，Oracle也提供这种功能，通过序列加触发器来解决。这当然很方便,但是当希望在插入一条记录后立即获得该记录的主键值，问题就出现了,因为很可能为此不得不新写一条语句去获取该主键值。iBatis.Net提供了一种比较好的解决方式。通过在XML数据映射文件中的<selectKey>元素来获取这些自动生成的主键值并将其保存在对象中。

如上面的"添加"操作配置信息修改为：

```
<insert id="InsertSysuser" parameterClass="Sysuser">

<selectKey resultClass="int32" property="Userid" type="pre">

SELECT DEAN.SEQ_SYSUSER_USERID.NEXTVAL AS Userid FROM DUAL

</selectKey>

INSERT INTO DEAN.SYSUSER
(USERID,PASSWORD,LOGINNAME,SEX,BIRTHDAY,IDCARD,OFFICEPHONE,FAMILYPHONE,MOBILEPHONE,E
MAIL,ADDRESS,ZIPCODE,REMARK,STATUS)

VALUES
(#Userid#,#Password#,#Loginname#,#Sex#,#Birthday#,#Idcard#,#Officephone#,#Familyphon
e#,#Mobilephone#,#Email#,#Address#,#Zipcode#,#Remark#,#Status#)

</insert>
```

粗体部分为新加的selectKey节点。#Userid#参数将使用节点中从序列里选出的值进行填充。修改调用的代码：

```
ISqlMapper mapper = Mapper.Instance(); //得到ISqlMapper实例

Int32 result = (Int32)mapper.Insert("SysuserMap.InsertSysuser", model);//调
用Insert方法

Label1.Text = "添加用户成功,result返回UserID为" + Convert.ToString(result) +
",Sysuser实例model的UserID为" + Convert.ToString(model.Userid);
```

运行程序，这样result就得到想要获取的键值，同时Sysuser的实例化对象model的UserID属性也返回了该条新记录的键值。

4.3 #与**$**的选择

在XML数据映射文件中，参数用#号来表示。如前面提到的#Userid#，同时也可以使用$来标识。两者有什么区别，以及如何选择呢？

（1）#是把传入的数据当作参数，如 order by #field# ,如果#field#传入的值是字符型，传入值为id, 则sql语句会生成order by "id"，很明显生成的sql语句提交给数据库执行会报错。#参数往往用在需传入实际变量值的情况，如where id=#id#，变量#id#传入10，则sql语句会生成where id=10。

（2）$采用拼接方式生成sql语句，即传入的数据直接生成在sql语句里，如 order by $field$，如$field$传入的是id,则sql语句会生成order by id。$参数方式一般用于传入数据库对象。例如传入表名。不过要特别提醒，因为使用该参数是直接组成sql语句，所以需注意防止sql注入。

4.4 LIKE语句处理

在查询处理过程中，模糊查询经常会用到。iBatis如何处理这种模糊查询呢？处理Like模糊查询iBatis提供以下两种方法：

（1）使用#参数： 配置如Where UserName Like #param#||'%'，采用参数传递的方式，如#param#传入字符d，生成的sql语句为Where UserName Like 'd%'，其他模糊情况同理可以相应配置出来。

（2）使用$参数：配置如Where UserName Like '$param$%' ，采用字符串替换，就是用参数的值替换param，如$param$传入字符d，也可以生成相同的语句。

$方式会引起sql注入风险，实际的使用中，建议大家都使用第一种配置方式来处理LIKE模糊查询。

5 结语

以上程序在Windows7(64位)+VS2012+Oracle 11g(64位)上测试通过。

分类：C#编程,Oracle数据库

北极燕鸥
关注 - 1
粉丝 - 39
+加关注

4 0

« 上一篇：[原]iBatis.Net（C#）系列一：简介及运行环境
» 下一篇：[原]iBatis.Net（C#）系列三：数据库查询

Feedback

#1楼
2013-03-01 10:11 by 黑色街角

楼主代码要是能够着色就更棒了！支持一下！

支持(0) 反对(0)

#2楼
2013-03-01 15:24 by xj3614

非常感谢啊,项目正要用这个东东.四处找寻各种资料都是java的,我正愁找到不好打学习资料,楼主真好,看了你写的我就大概明白了很多.

支持(0) 反对(0)

#3楼[楼主]
2013-03-01 15:28 by 北极燕鸥

@ xj3614
呵呵，有空一起研究，随后出第3篇

支持(0) 反对(0)

#4楼
2013-03-01 15:49 by xj3614

@ 北极燕鸥
感谢啊,一定持续关注.

支持(0) 反对(0)

#5楼
2013-05-23 19:04 by 小鸡哥

最近用到了 ibatisnet .
遇到个问题就是sql注入问题:
我用#传参
老是报错:
insert into share_Journal (UserId,CreateDateTime,Title) values (#UserId#,#CreateDateTime#,N#Title#);
UserId 是int，CreateDateTime是DateTime，Title是Varchar。
报错是'@param2' 附近有语法错误。
求解方法啊

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部