LINQ TO SQL

一）表与字段的映射：

[System.Data.Linq.Mapping 命名空间](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/9f8e8440-a491-7dba-93bd-170406129fde.htm)

[Table] , [Column] 属性来完成对象到数据模型的映射。

[Table]:

Name: 对应数据模型的表或者视图的名称

[Column]:

**属性**

|  |  |  |
| --- | --- | --- |
|  | **名称** | **说明** |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [AutoSync](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/0dd2a686-c6fc-eb9c-0952-9475da86ff88.htm) | 获取或设置 [AutoSync](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/e657e4a3-81e1-bd5e-aa45-935c4fda8811.htm) 枚举。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [CanBeNull](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/c096af53-d831-c7c0-259a-ca710115290a.htm) | 获取或设置一个值，该值指示列是否可包含 null 值。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [DbType](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/0fd4876c-63d3-a896-65f6-7febd9f335a7.htm) | 获取或设置数据库列的类型。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [Expression](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/e0cfcb0a-8666-17db-7059-38ae20b321f9.htm) | 获取或设置一个值，该值指示列是否为数据库中的计算列。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [IsDbGenerated](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/a66e3e13-7909-ffe0-8cd5-c82eefd1c5d9.htm) | 获取或设置一个值，该值指示列是否包含数据库自动生成的值。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [IsDiscriminator](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/2299b7e0-0998-8d9d-8826-6e06776bee02.htm) | 获取或设置一个值，该值指示列是否包含 LINQ to SQL 继承层次结构的鉴别器值。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [IsPrimaryKey](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/b5c43b2e-0878-83f9-bf1d-7a77aa79ebf5.htm) | 获取或设置一个值，该值指示该类成员是否表示作为表的整个主键或部分主键的列。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [IsVersion](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/1de9a92d-1c26-0acd-ec75-16c128ee6c44.htm) | 获取或设置一个值，该值指示成员的列类型是否为数据库时间戳或版本号。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [Name](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/8b65b951-4dd9-fffd-5da9-4c5aa34a253e.htm) | 获取或设置列名称。 （继承自 [DataAttribute](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/e9df0774-c645-882e-3f11-780ffc071591.htm)。） |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [Storage](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/e26b555e-f6fe-dbda-2bb1-09bf708a0409.htm) | 获取或设置私有存储字段以保存列中的值。 （继承自 [DataAttribute](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/e9df0774-c645-882e-3f11-780ffc071591.htm)。） |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | TypeId | 当在派生类中实现时，获取该 Attribute 的唯一标识符。 （继承自 Attribute。） |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/icons/pubproperty.gif | [UpdateCheck](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq.mapping/html/7697e385-ff84-a87c-8ca6-a146ab63fc93.htm) | 获取或设置一个值，该值指示 LINQ to SQL 如何进行开放式并发冲突的检测。 |

[DataContext 类](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm): 与数据源的主要入口点

**重载列表**

|  |  |  |
| --- | --- | --- |
|  | **名称** | **说明** |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DataContext(IDbConnection)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/09ef4227-935c-7693-8315-37d9bb8f6c5a.htm) | 通过引用由 .NET Framework 使用的连接来初始化 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm) 类的新实例。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DataContext(String)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/27ae7300-9f77-dd7c-2f98-c21f81752064.htm) | 通过引用文件源来初始化 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm) 类的新实例。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DataContext(IDbConnection, MappingSource)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/4cce17cc-fdcd-60a7-5eab-f9b374d4e2be.htm) | 通过引用连接和映射源初始化 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm) 类的新实例。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DataContext(String, MappingSource)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/a5de68ab-8937-3559-3bcc-9a9625c266b5.htm) | 通过引用文件源和映射源初始化 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm) 类的新实例。 |

[Table<TEntity> 类](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm)

**方法**

|  |  |  |
| --- | --- | --- |
|  | **名称** | **说明** |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [Attach](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/f6e26b5b-4b63-5108-f52e-7cccc1794e1e.htm) | 已重载。 将实体附加到 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm)。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [AttachAll](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/53c052a8-5ff1-3079-6576-e25ee32d84cc.htm) | 已重载。 以修改或未修改状态将集合的所有实体附加到 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm)。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DeleteAllOnSubmit<(Of <(TSubEntity>)>)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/c8b55eef-d25f-6f25-837d-57d1d56bc115.htm) |  |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [DeleteOnSubmit](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/2c43dd1d-21ba-e053-2a9d-2f6d7de5acfa.htm) | 将此表中的实体置为 **pending delete** 状态。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [InsertAllOnSubmit<(Of <(TSubEntity>)>)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/4674312e-d7e8-9101-5e47-f548f01108b3.htm) | 将集合中所有处于 **pending insert** 状态的实体添加到 [DataContext](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/18dcdfc5-4021-bb96-6e55-86d4512d6552.htm)。 |
| ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/icons/pubmethod.gif | [InsertOnSubmit](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/b7400daf-7bf7-4129-97b9-dd1fee0388fe.htm) | 将处于 **pending insert** 状态的实体添加到此 [Table<(Of <(TEntity>)>)](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/d1d97fb4-a68c-711d-defb-765a3ded6a5b.htm)。 |

简单的例子：

[Table(Name="country")]

public class t\_country

{

[Column(IsPrimaryKey=true, IsDbGenerated=true, Name="id", DbType="Int Not null")]

public int id;

[Column(Name="name", DbType="varchar(50) Not Null")]

public string name;

[Column(Name="iso")]

public string isoname;

}

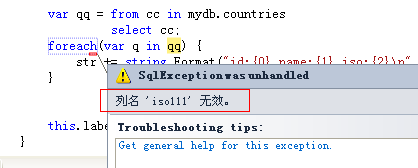
注意：

1. Table 和 Column 的 Name属性是对应数据模型的表名称（或者视图名称）， 和列的名称。 如果数据模型找不到对应的列则抛出错误. 如下：

[Column(Name="iso111")]

public string isoname;

当我们 GetTable<> 时会出错。



1. 如果没有指定： IsPrimaryKey=true

[Column(Name="id", CanBeNull=false)]

public int id;

修改找到的记录, 提交并不会把数据提交给数据库。

t\_country mc = country.SingleOrDefault(p => p.id == 10);

mc.id = 20;

mc.name = "Mex";

mc.isoname = "Mexico";

country.SingleOrDefault(p => p.id == 10).name = "UK";

country.SingleOrDefault(p => p.id == 10).isoname = "Unite Kingdom";

db.SubmitChanges();

str += string.Format("mc:{0}-{1}-{2}\n\n",mc.id, mc.name, mc.isoname);

str += string.Format("country:{0}-{1}-{2}\n\n",

country.SingleOrDefault(p => p.id == 10).id,

country.SingleOrDefault(p => p.id == 10).name,

country.SingleOrDefault(p => p.id == 10).isoname

);

输出



无论什么方法都改不了，集合和数据库的值， 因为没有 PrimaryKey

同样的代码我们添加 PrimaryKey

[Column(Name="id", IsPrimaryKey=true, CanBeNull=false)]

public int id;



我们看见 mc 和 country 集合里的记录是指向同一个对象， 并且更改一并提交给数据库。

关于修改的注意:

1. 定义为 IsPrimaryKey=true ， 不管数据库的设置如何，在客户端是不允许修改的。
2. 关于 IsDbGenerated=true , 不管人工设置何值， 都将以 null 传给数据库。

[Table(Name="country")]

public class t\_country

{

[Column(Name="id", IsPrimaryKey=true, IsDbGenerated=true, CanBeNull=false)]

public int id;

[Column(Name="name", DbType="varchar(50)", CanBeNull=true)]

public string name;

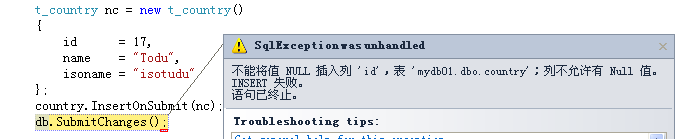
[Column(Name="iso", CanBeNull=false)]

public string isoname;

}

如果数据库的没有自动号码： Indentity , 而在对象模型里指定为自动赋值。

如果我们手工给ID 赋值， 我们提交修改时。



因为是对象标注为自动赋值， 即使我们给 id 赋值， 但是该值会以 null 传递给数据库, 而忽略该值。

所以就出现了数据库写入时 id 为空的错误， 错误是来自数据库端的写入操作。

现在数据库设定了自动号码：

t\_country nc = new t\_country()

{

id = 17,

name = "Todu",

isoname = "isotudu"

};

country.InsertOnSubmit(nc);

str += string.Format("before id:{0}\n\n", nc.id);

db.SubmitChanges();

str += string.Format("after id:{0}\n\n", nc.id);



**所以如何获取新增记录的 ID : 可以提交完以后， 访问该记录的 id 即可。**

1. CanBeNull=false , 对于对象来说是否是null , 都可以赋值. 关键取决于数据库端是否允许为空。

[Table(Name="country")]

public class t\_country

{

[Column(Name="id", IsPrimaryKey=true, IsDbGenerated=true, CanBeNull=false)]

public int id;

[Column(Name="name", DbType="varchar(50)", CanBeNull=true)]

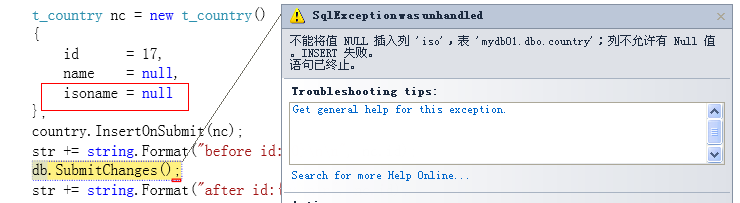
public string name;

[Column(Name="iso", CanBeNull=false)]

public string isoname;

}

插入新记录： 错误来自于数据库写入操作， 而对象是允许赋予空值的。



主体程序：

public DataContext db;

public Table<t\_country> country;

private void Form1\_Load(object sender, EventArgs e)

{

// 可以直接使用连接字符串

string con\_str = @"Server=CANSER\MYSQL2008;Database=mydb01;user id=lwh;pwd=lwh123";

db = new DataContext(con\_str);

// 也可以使用 IDbConnection

SqlConnection scon = new SqlConnection(con\_str);

db = new DataContext(scon);

country = db.GetTable<t\_country>();

var qq = from cc in country

select cc;

string str = null;

foreach(var q in qq) {

str += string.Format("id:{0} name:{1} iso:{2}\n", q.id, q.name, q.isoname);

}

this.label1.Text = str;

}

二) [Column] 可以对应 对象里的 field 或者 property.

[Table(Name="country")]

public class t\_country

{

[Column(Name="id", IsPrimaryKey=true, IsDbGenerated=true, CanBeNull=false)]

public int id;

[Column(Name="name", DbType="varchar(50)", CanBeNull=true)]

public string name;

[Column(Name="iso", CanBeNull=false)]

public string isoname;

}

注意：使用field 在DataGridView 里无法显示数据。

[Table(Name="country")]

public class t\_country

{

private int \_id;

private string kk;

private string \_isoname;

[Column(Name = "id", IsPrimaryKey = true, IsDbGenerated = true, CanBeNull = false)]

public int id

{

get{ return this.\_id; }

set{ this.\_id = value;}

}

[Column(Name = "name",Storage="kk", DbType = "varchar(50)", CanBeNull = true)]

public string name

{

get { return this.kk; }

set { this.kk = value; }

}

[Column(Name = "iso", CanBeNull = false)]

public string isoname

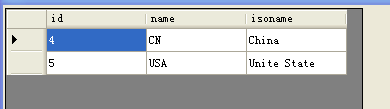
{

get { return this.\_isoname; }

set { this.\_isoname = value; }

}

}



使用属性则输出到 DataGridView 没有问题。

三）[Database(Name="mydbxxx")] 的使用

[Database(Name="mydb01")]

public class t\_db : DataContext

{

public static string con\_str = @"Server=CANSER\MYSQL2008;Database=mydb02;user id=sa;pwd=liu011225";

public t\_db() : base(con\_str) { }

public t\_db(string s) : base(s) { }

public t\_db(IDbConnection con) : base(con) { }

public Table<t\_country> countries

{

Get { return this.GetTable<t\_country>(); }

}

}

注意：1） 数据库的 [Database(Name="mydbxxx")] Name 属性并不是用来对应数据库里的数据库名字。

Name 的属性只是 DataContext. Mapping. DatabaseName

而实际的数据库名称，取决于 Connect String 连接字符串里的 Database = mydb02;

String.Format("DB:{0}-{1}\n", mydb.Mapping.DatabaseName, mydb.Connection.Database);

输出： DB: mydbxxx - mydb02

DataContext 的数据库操作：

mydb = new t\_db();

if (mydb.DatabaseExists()) mydb.DeleteDatabase();

mydb.CreateDatabase();

会根据 DataContext 里的 Table 定义自动创建数据库和表格。

[Table(Name="mydb01.dbo.country")] //注意：表可以使用数据库的全路径名称。

public class t\_country

{

……  
 }

虽然数据库的Database=mydb02 , 但是如果用户权限可以跨数据库访问 mydb01.dbo.country 也是允许的。

mydb = new t\_db();

// 新增

t\_country nc = new t\_country()

{

id = 25,

name = "UK",

isoname = "United Kingdom"

};

mydb.countries.InsertOnSubmit(nc);

// 修改

mydb.countries.Single(p => p.id == 14).name = "Moxican";

mydb.countries.Single(p => p.id == 14).isoname = "ISO Moxican";

mydb.SubmitChanges();

// 删除

mydb.countries.DeleteOnSubmit(mydb.countries.First());

mydb.SubmitChanges();

三）[Association] 的使用

[Table(Name="country")]

public class t\_country //: INotifyPropertyChanging, INotifyPropertyChanged

{

//private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_id;

private string kk;

private string \_isoname;

public t\_country()

{

this.\_city = new EntitySet<t\_city>(new Action<t\_city>(this.attach\_cities), new Action<t\_city>(this.detach\_cities));

//this.\_city = new EntitySet<t\_city>();

}

[Column(Name = "id", IsPrimaryKey = true, AutoSync = AutoSync.OnInsert, IsDbGenerated = true, CanBeNull = false, UpdateCheck=UpdateCheck.Always)]

public int id

{

get{ return this.\_id; }

set

{

this.\_id = value;

}

}

[Column(Name = "name",Storage="kk", DbType = "varchar(50)", CanBeNull = true)]

public string name

{

get { return this.kk; }

set { this.kk = value; }

}

[Column(Name = "iso", CanBeNull = false)]

public string isoname

{

get { return this.\_isoname; }

set { this.\_isoname = value; }

}

private EntitySet<t\_city> \_city;

[Association(Name = "ass\_cn\_ct", Storage="\_city", ThisKey="id", OtherKey="ref\_id")]

public EntitySet<t\_city> cities

{

get { return this.\_city; }

set { this.\_city.Assign(value); }

}

// 以下 attach , detach 对于操作子表时会使用。

private void attach\_cities(t\_city entity)

{

entity.pcountry = this;

}

private void detach\_cities(t\_city entity)

{

entity.pcountry = null;

}

}

[Table(Name="city")]

public class t\_city

{

private int \_id;

private int \_ref\_id;

private string \_name;

public t\_city()

{

this.\_pcountry = default(EntityRef<t\_country>);

}

[Column(Name = "id", Storage="\_id", IsPrimaryKey = true, IsDbGenerated = true, CanBeNull = false)]

public int id

{

get { return this.\_id; }

set { this.\_id = value; }

}

[Column(Name = "ref\_id", Storage="\_ref\_id", CanBeNull = true)] // 如果不允许为空；作为子表添加时会出错。

public int ref\_id

{

get { return this.\_ref\_id; }

set { this.\_ref\_id = value; }

}

[Column(Name = "name", Storage="\_name", CanBeNull = false)]

public string name

{

get { return this.\_name; }

set { this.\_name = value; }

}

private EntityRef<t\_country> \_pcountry;

[Association(Name="ass\_ct\_cn", IsForeignKey=true, ThisKey="ref\_id", OtherKey="id")]

public t\_country pcountry

{

get { return this.\_pcountry.Entity; }

set

{

this.\_pcountry.Entity = value;

}

}

}

[Database(Name="mydbxxx")]

public class t\_db : DataContext

{

public static string con\_str = @"Server=CANSER\MYSQL2008;Database=mydb02;user id=sa;pwd=liu011225";

public t\_db() : base(con\_str)

{

}

public t\_db(string s)

: base(s)

{

}

public t\_db(IDbConnection con)

: base(con)

{

}

public Table<t\_country> countries

{

get

{

return this.GetTable<t\_country>();

}

}

public Table<t\_city> cities

{

get

{

return this.GetTable<t\_city>();

}

}

}

}

此模型删除子表有问题， 还不清楚问题出在哪里？

**以下是 Linq to SQL Class 自动生成的代码：**

#pragma warning disable 1591

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

// Runtime Version:4.0.30319.225

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace WinLinqSQL1

{

using System.Data.Linq;

using System.Data.Linq.Mapping;

using System.Data;

using System.Collections.Generic;

using System.Reflection;

using System.Linq;

using System.Linq.Expressions;

using System.ComponentModel;

using System;

[global::System.Data.Linq.Mapping.DatabaseAttribute(Name="mydb01")]

public partial class sql1DataContext : System.Data.Linq.DataContext

{

private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();

#region Extensibility Method Definitions

partial void OnCreated();

partial void Insertcountry(country instance);

partial void Updatecountry(country instance);

partial void Deletecountry(country instance);

partial void Insertcity(city instance);

partial void Updatecity(city instance);

partial void Deletecity(city instance);

#endregion

public sql1DataContext() :

base(global::WinLinqSQL1.Properties.Settings.Default.mydb01ConnectionString, mappingSource)

{

OnCreated();

}

public sql1DataContext(string connection) :

base(connection, mappingSource)

{

OnCreated();

}

public sql1DataContext(System.Data.IDbConnection connection) :

base(connection, mappingSource)

{

OnCreated();

}

public sql1DataContext(string connection, System.Data.Linq.Mapping.MappingSource mappingSource) :

base(connection, mappingSource)

{

OnCreated();

}

public sql1DataContext(System.Data.IDbConnection connection, System.Data.Linq.Mapping.MappingSource mappingSource) :

base(connection, mappingSource)

{

OnCreated();

}

public System.Data.Linq.Table<country> countries

{

get

{

return this.GetTable<country>();

}

}

public System.Data.Linq.Table<city> cities

{

get

{

return this.GetTable<city>();

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.country")]

public partial class country : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_id;

private string \_name;

private string \_iso;

private EntitySet<city> \_cities;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnidChanging(int value);

partial void OnidChanged();

partial void OnnameChanging(string value);

partial void OnnameChanged();

partial void OnisoChanging(string value);

partial void OnisoChanged();

#endregion

public country()

{

this.\_cities = new EntitySet<city>(new Action<city>(this.attach\_cities), new Action<city>(this.detach\_cities));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_id", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int id

{

get

{

return this.\_id;

}

set

{

if ((this.\_id != value))

{

this.OnidChanging(value);

this.SendPropertyChanging();

this.\_id = value;

this.SendPropertyChanged("id");

this.OnidChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_name", DbType="VarChar(50)")]

public string name

{

get

{

return this.\_name;

}

set

{

if ((this.\_name != value))

{

this.OnnameChanging(value);

this.SendPropertyChanging();

this.\_name = value;

this.SendPropertyChanged("name");

this.OnnameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_iso", DbType="VarChar(50)")]

public string iso

{

get

{

return this.\_iso;

}

set

{

if ((this.\_iso != value))

{

this.OnisoChanging(value);

this.SendPropertyChanging();

this.\_iso = value;

this.SendPropertyChanged("iso");

this.OnisoChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="country\_city", Storage="\_cities", ThisKey="id", OtherKey="ref\_id")]

public EntitySet<city> cities

{

get

{

return this.\_cities;

}

set

{

this.\_cities.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_cities(city entity)

{

this.SendPropertyChanging();

entity.country = this;

}

private void detach\_cities(city entity)

{

this.SendPropertyChanging();

entity.country = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="")]

public partial class city : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_id;

private int \_ref\_id;

private string \_name;

private EntityRef<country> \_country;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnidChanging(int value);

partial void OnidChanged();

partial void Onref\_idChanging(int value);

partial void Onref\_idChanged();

partial void OnnameChanging(string value);

partial void OnnameChanged();

#endregion

public city()

{

this.\_country = default(EntityRef<country>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_id", IsPrimaryKey=true, IsDbGenerated=true)]

public int id

{

get

{

return this.\_id;

}

set

{

if ((this.\_id != value))

{

this.OnidChanging(value);

this.SendPropertyChanging();

this.\_id = value;

this.SendPropertyChanged("id");

this.OnidChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ref\_id")]

public int ref\_id

{

get

{

return this.\_ref\_id;

}

set

{

if ((this.\_ref\_id != value))

{

if (this.\_country.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.Onref\_idChanging(value);

this.SendPropertyChanging();

this.\_ref\_id = value;

this.SendPropertyChanged("ref\_id");

this.Onref\_idChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_name", CanBeNull=false)]

public string name

{

get

{

return this.\_name;

}

set

{

if ((this.\_name != value))

{

this.OnnameChanging(value);

this.SendPropertyChanging();

this.\_name = value;

this.SendPropertyChanged("name");

this.OnnameChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="country\_city", Storage="\_country", ThisKey="ref\_id", OtherKey="id", IsForeignKey=true)]

public country country

{

get

{

return this.\_country.Entity;

}

set

{

country previousValue = this.\_country.Entity;

if (((previousValue != value)

|| (this.\_country.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_country.Entity = null;

previousValue.cities.Remove(this);

}

this.\_country.Entity = value;

if ((value != null))

{

value.cities.Add(this);

this.\_ref\_id = value.id;

}

else

{

this.\_ref\_id = default(int);

}

this.SendPropertyChanged("country");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}

#pragma warning restore 1591

并发数据操作异常处理：

try

{

this.mydb.SubmitChanges(ConflictMode.ContinueOnConflict);

}

catch (ChangeConflictException err)

{

string err\_msg = null;

foreach (ObjectChangeConflict oc in mydb.ChangeConflicts)

{

oc.Resolve(RefreshMode.KeepChanges);

foreach (MemberChangeConflict mc in oc.MemberConflicts)

{

err\_msg += String.Format("db\_value:{0} Org\_value:{1} Curr\_value:{2}\n",

mc.DatabaseValue,

mc.OriginalValue,

mc.CurrentValue);

}

}

err\_msg += string.Format("Error:{0}\nMessage:{1}\nTrack:{2}",

err.Message,

err.Data.Count,

err.InnerException);

this.label1.Text = err\_msg;

}

RefreshMode 枚举

# 成员

|  |  |  |
| --- | --- | --- |
|  | **成员名称** | **说明** |
|  | KeepCurrentValues | 强制 [Refresh](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/d07d3b26-f9a1-2701-a0c0-41f912539ac1.htm) 方法使用从数据库检索的值替换原始值。不会修改当前值。 |
|  | KeepChanges | 强制 [Refresh](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/d07d3b26-f9a1-2701-a0c0-41f912539ac1.htm) 方法保留已更改的当前值，但将其他值更新为数据库值。 |
|  | OverwriteCurrentValues | 强制 [Refresh](ms-help://MS.MSDNQTR.v90.chs/fxref_system.data.linq/html/d07d3b26-f9a1-2701-a0c0-41f912539ac1.htm) 方法使用数据库中的值重写所有当前值。 |

通常并发异常后， 解决冲突以后，再按保存就可以保存新的值了。

事务的使用：

1. 使用 TransactionScope

using (TransactionScope ts = new TransactionScope())

{

……. Update, insert , delete, add operation

mydb.SubmitChanges();

ts.Complete();

ts.Dispose();

}

2） 使用连接的事务：

try

{

mydb.Connection.Open();

mydb.Transaction = mydb.Connection.BeginTransaction();

……. Update, insert , delete, add operation

mydb.SubmitChanges();

mydb.Transaction.Commit();

mydb.Transaction.Dispose();

}

catch

{

mydb.Transaction.Rollback();

}

―――――――――――――――――――――――――――――――――――――――――――――――――――――――

Linq to SQL 映射存储过程或者函数：

首先DataContext有一些直接操作数据库的方法：

[ExecuteCommand](ms-help://MS.MSDNQTR.v90.en/fxref_system.data.linq/html/90af9a6d-827e-1fe8-a7cb-334b51f6c3eb.htm)

[ExecuteQuery](ms-help://MS.MSDNQTR.v90.en/fxref_system.data.linq/html/cb01912b-96ba-abe4-bd1f-27f9cacf0a0b.htm)

[ExecuteMethodCall](ms-help://MS.MSDNQTR.v90.en/fxref_system.data.linq/html/349eee93-5a50-463a-186f-8f846ecc131b.htm)

var cnt = db.ExecuteQuery<int>("select max(id) from country");

this.label1.Text = cnt.First().ToString();

System.Collections.IEnumerable cnt1 = db.ExecuteQuery(typeof(int), "select max(id) from country");

this.label1.Text = cnt1.ToString();

System.Collections.IEnumerator ee = cnt1.GetEnumerator();

ee.MoveNext();

this.label1.Text = ((int)ee.Current).ToString();

var cs = db.ExecuteQuery<country>("select \* from country");

//cs.ToList();

this.dgv1.DataSource = cs.ToList();

db.ExecuteCommand("UPDATE country SET name = 'China PRC1111' WHERE id = {0} AND id = {1}", 10,10);

db.ExecuteCommand("Insert country (name, iso) values({0}, {1})", "good", "luck");

[Table(Name = "depart")]

public class dep

{

private int \_id;

private string \_dd;

[Column(Name = "id", IsDbGenerated = true, IsPrimaryKey = true)]

public int id

{

get { return \_id; }

set { \_id = value; }

}

[Column(Name = "depart", DbType = "varchar(50)")]

public string depp

{

get { return \_dd; }

set { \_dd = value; }

}

}

// DataContext 必须有定义表的类

var cs = db.ExecuteQuery<dep>("select \* from depart");

this.dgv1.DataSource = cs.ToList();

**如何映射到存储过程：**

1)情况一： 不带参数的， 只返回数值的存储过程

Create PROCEDURE [dbo].[max\_id]

AS

BEGIN

declare @mn int;

select @mn = MAX(id) from country;

return @mn

END

―――――――――――――――――――――――――

ALTER PROCEDURE test1  
 @num varchar(100)  
AS  
BEGIN  
 return '222字符333';  
END

―――――――――――――――――――――――――

注意： 存储过程的 return 只能是整型数据， 不能是字符或者其他类型。 否则出错。

// 必须在 DataContext 类里去映射数据库的存储过程或者函数

[Function(Name = "max\_id")]

public int call\_max()

{

IExecuteResult ies = this.ExecuteMethodCall( this, ((MethodInfo)MethodInfo.GetCurrentMethod()) );

return (int)ies.ReturnValue;

}

// 以上是不带任何参数的， 而且返回值是整型数。

2)情况二： 不带参数的， 有返回值，也有返回表数据的存储过程

Create PROCEDURE [dbo].[max\_name]

AS

BEGIN

declare @mn int;

SELECT id, name as depart FROM country;

select @mn = MAX(id) from country;

return @mn + 10000

END

[Function(Name = "max\_name")]

public ISingleResult<dep> call\_max()

{

IExecuteResult ies = this.ExecuteMethodCall(this, ((MethodInfo)MethodInfo.GetCurrentMethod()) );

return (ISingleResult<dep>)(ies.ReturnValue);

}

ISingleResult<dep> iee = db.call\_max();

this.dgv1.DataSource = iee.ToList();

返回数据表

this.label1.Text = iee.ReturnValue.ToString(); 返回ID 的最大值 + 10000

**注意：**

**1）** IExecuteResult ies.ReturnValue 和 ISingleResult<dep> iee. ReturnValue 的区别。

**2）** 返回什么类型，只要能有对应的类对应即可，如果类的属性不完全吻合，则使用忽略合并。

3)情况三： 带输入参数，输出参数， 有返回值，也有返回表数据的存储过程

Create PROCEDURE [dbo].[max\_name]

@cid int,

@ma\_name varchar(50) output

AS

BEGIN

declare @mn int;

SELECT id, name as depart FROM country where id = @cid;

select @mn = MAX(id) from country;

select @ma\_name = MAX(name) from country;

return @mn + 10000

END

[Function(Name = "max\_name")]

public ISingleResult<dep> call\_max(

[Parameter(DbType="Int", Name="cid")] int idd,

[Parameter(DbType="varchar(50)", Name="ma\_name")] ref string ma\_name //使用引用

)

{

IExecuteResult ies = this.ExecuteMethodCall(this, ((MethodInfo)MethodInfo.GetCurrentMethod()), idd, ma\_name);

注意 SQL 的output 参数是返回给参数，而不是返回给传入的变量， 所以必须通过获取对应的参数来获取输出的值。

以下是获取输出参数值的方法：

ma\_name = (string)ies.GetParameterValue(1); // 输出参数是第二个参数， 返回是object , 记得强制转化

return (ISingleResult<dep>)(ies.ReturnValue);

}

string tname = string.Empty;

ISingleResult<dep> iee = db.call\_max(10, ref tname); //使用引用

this.dgv1.DataSource = iee.ToList();

this.label1.Text = iee.ReturnValue.ToString() + "\nName:\*" + tname + "\*";

注意： public interface ISingleResult<T> : IEnumerable<T>,

IEnumerable, [IFunctionResult](ms-help://MS.MSDNQTR.v90.en/fxref_system.data.linq/html/2432e6f7-84c7-5898-2c0f-c798d96ab125.htm), IDisposable

是接口： 继承可枚举列表。

4)情况四： 带输入参数，输出参数， 有返回值，并且返回多个表数据的存储过程

ALTER PROCEDURE [dbo].[max\_name]

@cid int,

@ma\_name varchar(50) output

AS

BEGIN

declare @mn int;

SELECT \* FROM country where id = @cid;

SELECT \* FROM city where ref\_id = @cid;

select @mn = MAX(id) from country;

select @mn = @mn + MAX(id) from city;

select @ma\_name = MAX(name) from country;

select @ma\_name = @ma\_name + '::' + MAX(name) from city;

return @mn + 10000 // 26 + 14 + 10000

END

[Function(Name = "max\_name")]

[ResultType(typeof(country))] // 有无都无所谓， 并不限制一定是这些定义的类型

[ResultType(typeof(city))]

public IMultipleResults call\_max(

[Parameter(DbType="Int", Name="cid")] int idd,

[Parameter(DbType="varchar(50)", Name="ma\_name")] ref string ma\_name

)

{

IExecuteResult ies = this.ExecuteMethodCall(this, ((MethodInfo)MethodInfo.GetCurrentMethod()), idd, ma\_name);

ma\_name = (string)ies.GetParameterValue(1);

return (IMultipleResults)(ies.ReturnValue);

}

相对简单： 只使用 IMultipleResults 返回

客户端代码：IMultipleResults GetResult 方法获取不同的结果集合：

ImultipleResults 的方法 GetResult<TElement>()

IEnumerable<TElement> GetResult<TElement>() 返回是可枚举的类型

string tname = string.Empty;

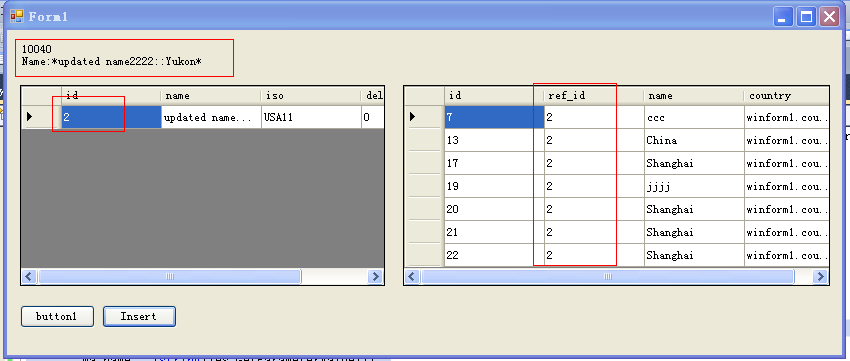
IMultipleResults iee = db.call\_max(2,ref tname);

this.dgv1.DataSource = iee.GetResult<country>().ToList();

this.dgv2.DataSource = iee.GetResult<city>().ToList();

this.label1.Text = iee.ReturnValue.ToString() + "\nName:\*" + tname + "\*";

输出结果如图：



注意我们如果使用其他的表类来接收返回的结果集合。则会使用忽略合并。

如下：

[Table(Name = "depart")]

public class dep

{

private int \_id;

private string \_dd;

[Column(Name = "id", IsDbGenerated = true, IsPrimaryKey = true)]

public int id

{

get { return \_id; }

set { \_id = value; }

}

[Column(Name = "depart", DbType = "varchar(50)")]

public string depp

{

get { return \_dd; }

set { \_dd = value; }

}

}

――――――――――――――――――――――――――――――――――――――――――

string tname = string.Empty;

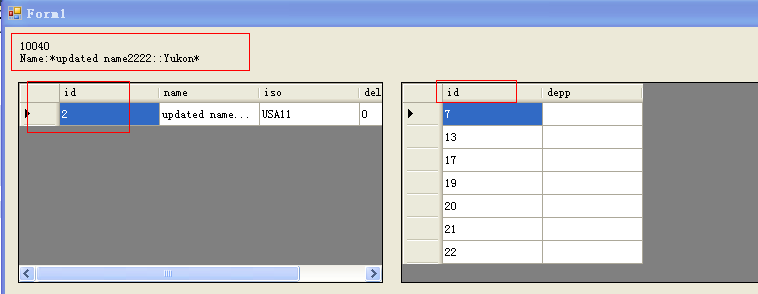
IMultipleResults iee = db.call\_max(2,ref tname);

this.dgv1.DataSource = iee.GetResult<country>().ToList();

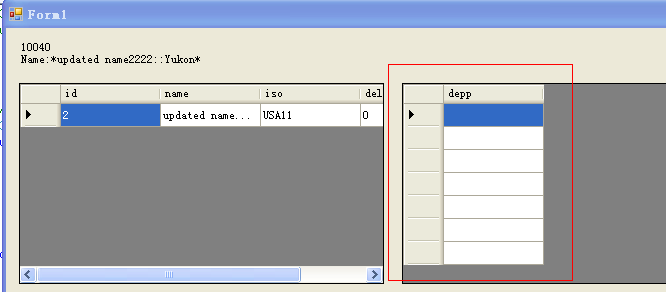
this.dgv2.DataSource = iee.GetResult<dep>().ToList();

this.label1.Text = iee.ReturnValue.ToString() + "\nName:\*" + tname + "\*";

―――――――――――――――――――――――――――――――――――――――――

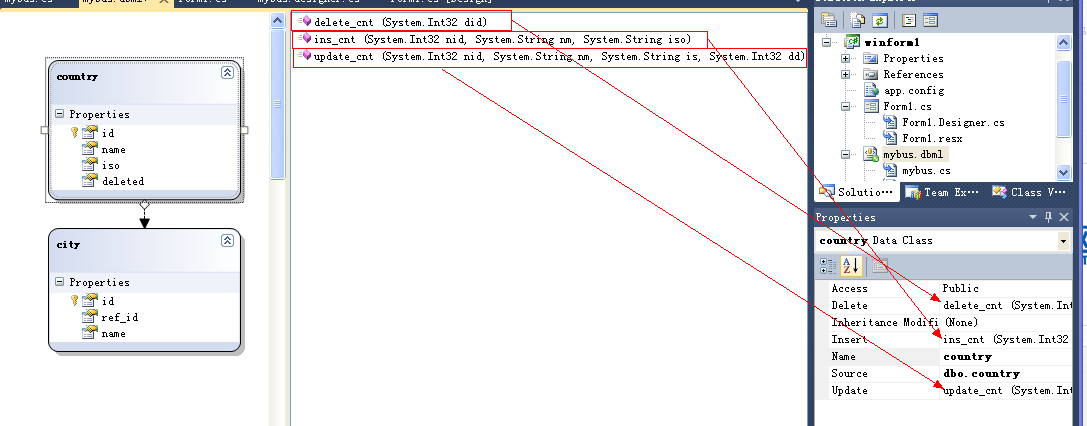


如果我们把 id 也屏蔽掉： 则产生空行：

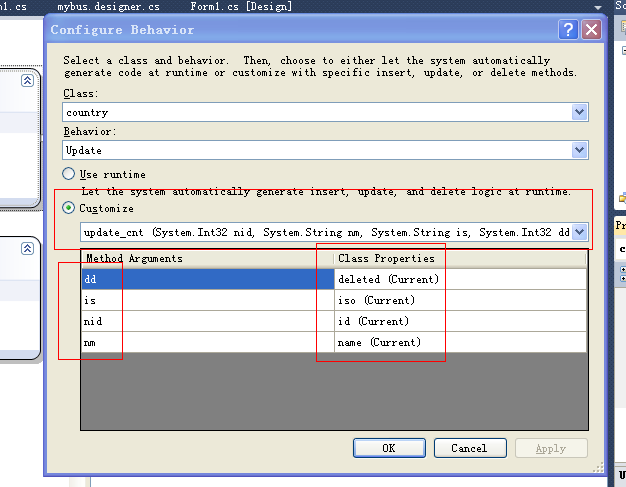


数据库存储过程被应用到数据表的insert, update, delete操作：

第一步：把存储过程从数据库拉到 linq to sql class: 然后指定到相应的操作。



第二步：对各个字段做相应的映射：



这样就完成了操作的映射关系：

删除，更新操作：

――― Delete ―――――――――――――――――――――――――――

country nc = db.countries.LastOrDefault();

db.countries.DeleteOnSubmit(nc);

db.SubmitChanges();

――― Delete and update simultaneously ―――――――――――――――――――――――――――

country nc = db.countries.Where(e=>e.id==38).FirstOrDefault();

db.countries.DeleteOnSubmit(nc);

nc = db.countries.FirstOrDefault();

nc.name = "Micheal Jason";

nc.iso = "iso jason";

db.SubmitChanges();

新增操作：

如果使用自定义的存储过程来替代 自动的 Runtime : 则会有以下的问题出现：

country nc = new country {

name="wilson",

iso = "iso will"

};

db.countries.InsertOnSubmit(nc);

db.SubmitChanges();

this.label1.Text = "new inser id:" + nc.id.ToString();

注意： 1) id 是自动标识，使用自定义的存储过程， 无法将自动生成的 id 号码传回给 客户端的对象。

[Column(Storage="\_id", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int id {

get { return this.\_id; }

set {

if ((this.\_id != value)) {

this.OnidChanging(value);

this.SendPropertyChanging();

this.\_id = value;

this.SendPropertyChanged("id");

this.OnidChanged();

}

}

}

如果插入操作使用的是自动的 “ Use Runtime”, 则数据库生成的ID 号码会自动同步到客户端， 只要直接访问对象的id, 即可获得id 值。

对于新插入的记录的 自动 id 如何同步回客户端， 需要寻找解决方案。

insert country(name, iso, deleted) values('Ins: ' + @nm, 'Ins: ' + @iso, 1)

select convert(int, scope\_identity()) as value;

以上代码不能解决问题！