**皓月非凡ORM技术文档**

|  |  |  |  |
| --- | --- | --- | --- |
| **框架版本(Version)** | 1.0 | | |
| **时间** | **操作** | **操作人** | **版本** |
| 2017.09.12 | 提交 | Xhp | 1.0 |

**一、ORM组成：**

**HYFrameWork.Core+ HYFrameWork.DAL.SqlServer**

**二、ORM 常用方法**

2.1、数据库连接对象创建

String connStr= "SqlserverConStr".ValueOfAppSetting(); //获取连接字符串

IDbConnection conn= = new SqlConnection("连接字符串"); //实例Sqlserver连接对象

public IRepository<User> Repository = new SqlServerRepository (conn); //创建一个User表的仓储

2.2 数据库字段特性

AutoIncrementAttribute //自增特性：添加后在执行单体插入、修改、删除时将过滤该字段

CheckAttribute //检查特性：暂时保留

DefaultAttribute //默认特性：暂时保留

NonWriteAttribute //忽略特性：添加后在执行单体插入、修改、删除时将过滤该字段

NotNullAttribute //非空特性：暂时保留

PrimaryKeyAttribute //主键特性：添加后在执行单体插入、修改、删除时将作为默认条件

UniqueAttribute //唯一特性：暂时保留

2.2、实体查询

2.2.1 单体查询

var obj = Repository.Get(where => where.UserName == "测试");

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName = @P0)

var obj = Repository.Get(where =>

where.UserName == "测试"&&

where.Id >0||

where.Datetime > DateTime.Now);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (((UserName=@P0) AND (Id > @P1)) Or (DateTime > ‘2017/09/23 11:27:00’))

2.2.2 单体选择查询

var obj = Repository.GetEx(where =>

where.UserName == "AddTran测试0",

u => new User());

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName = @P0)

var obj = Repository.GetEx(where =>

where.UserName == "AddTran测试0",

u => new{u.Id,u.userName });

SELECT Id, UserName

FROM [User]

WHERE (UserName = @P0)

var obj = Repository.GetEx(where => where.UserName == "AddTran测试1",

order => order.Desc(o => o.Id),

u => new { u.Id, u.Name });

SELECT Id,Name

FROM [User]

WHERE (UserName = @P0)

ORDER BY Id DESC

2.2.3 列表查询

var obj = Repository.GetList(where => where.UserName.StartsWith("AddTran测试"));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName LIKE @P0)

var obj = Repository.GetList(where => where.UserName.StartsWith("AddTran测试"), 10);

SELECT TOP 10

Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName LIKE @P0)

var obj = Repository.GetList(where => where.UserName.StartsWith("AddTran测试"),

order => order.Desc(o => o.Id),10);

SELECT TOP 10

Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName LIKE @P0)

ORDER BY Id DESC

2.2.4 列表选择查询

var obj = Repository.GetListEx( where => where.UserName.StartsWith("AddTran测试"),

u => new User());

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName LIKE @P0)

var obj = Repository.GetListEx<User>(where => where.UserName.StartsWith("AddTran测试"), order => new DbSort<User>().Desc(o => o.Id),

u=>{u.Id,u.Name}, 10);

SELECT TOP 10 Id,Name

FROM [User]

WHERE (UserName LIKE @P0)

ORDER BY Id DESC

2.2.5 分页查询

//分页语句返回的一个PageList对象，其中已经包含了总数据量

//每页10条数据，查询第二页

var obj = Repository.PageList(where => where.UserName.StartsWith("AddTran测试"),2,10);

var obj = Repository.PageList(where => where.UserName.StartsWith("AddTran测试"),

order => order.Desc(o => o.Id),10,2);

var obj = Repository.PageListEx(where => where.UserName.StartsWith("AddTran测试"),

u => new {u.Id,u.Name}, 10, 3);

SELECT COUNT(1) FROM [User] WHERE (UserName LIKE 'AddList%') ;

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM (

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime ,ROW\_NUMBER() OVER(ORDER BY Id ASC) AS RowNumber

FROM [User]

WHERE (UserName LIKE 'AddList%')) A

WHERE RowNumber > 10 AND RowNumber <= 20 "

2.2.6 数量统计查询

var obj = Repository.Count(where => where.UserName.StartsWith("AddTran测试"));

SELECT COUNT(1)

FROM [User]

WHERE (UserName LIKE @P0)

2.3、数据库方法/WHERE条件语法

（注）WHERE 条件表达式支持参数对象调用方法，但尽量不要使用，或者不要过量嵌套调用方法（会增加解析耗时和激活反射机制消耗性能）

var obj = Repository.Get(where => where.Id > user.Id);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Id > @P0)

var obj = Repository.Get(where => where.Name != "");

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Name != '')

var obj = Repository.Get(where => where.Name == " ");

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Name = @P0)

var obj = Repository.Get(where => where.Sex || !where.Sex);

var obj = Repository.Get(where => where.Sex == true || where.Sex == false);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE ((Sex = @P0) OR (Sex = @P1))

var obj = Repository.Get(where =>

!where.Name.IsNullOrEmpty() ||

where.Name.IsNullOrEmpty());

var obj = Repository.Get(where =>

where.Name == null ||

where.Name != null);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE ((Name IS NOT NULL) OR (Name IS NULL))

var obj = Repository.Get(where => where.Name.Contains("2"));

//SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime FROM [User] WHERE (Name LIKE @P0 )

var obj = Repository.Get(where => where.Name.EndsWith("2"));

SLECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Name LIKE ‘%2’)

var obj = Repository.Get(where => where.UserName.StartsWith("2"));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UserName LIKE ‘2%’)

var obj = Repository.Get(where => where.UpdateTime > DateTime.Now.AddDays(-3));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (UpdateTime > @P0)

var obj = Repository.Get(where => new DbMethod().DateDiff(DateEnum.Day, where.UpdateTime, DateTime.Now) >= 1);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (（DATEDIFF(DAY,UpdateTime,’2017-09-25 00:00:00’)）>= @P0)

var obj = Repository.Get(where => new DbMethod().CharIndex("测试",where.UpdateTime) > 0);

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (（CHARINDEX(‘测试’,UpdateTime)）>= @P0)

List<int> ids = new List<int>() { 11, 22, 33 };

List<string> names = new List<string>() { "测试", "22", "33" };

var arr = new int[] { 11, 22, 33 };

var strArr = new string[] { "测试", "22", "33" };

var obj11 = Repository.Get(where => where.Id.In(ids));

var obj11\_2 = Repository.Get(where => ids.Contains(where.Id));

var obj11\_3 = Repository.Get(where => where.Id.In(arr));

var obj11\_4 = Repository.Get(where => where.Id.In(new[] { 11, 22, 33 }));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Id IN (11,22,33))

var obj11\_5 = Repository.Get(where => where.UserName.In(names));

var obj11\_6 = Repository.Get(where => names.Contains(where.UserName));

var obj11\_7 = Repository.Get(where => where.UserName.In(strArr));

var obj11\_8 = Repository.Get(where => where.UserName.In(new[] { "测试", "22", "33" }));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Name IN ('测试','22','33'))

var obj = Repository.GetList(where => where.Id.NotIn(ids));

SELECT Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime

FROM [User]

WHERE (Id NOT IN (11,22,33))

**四、增加**

4.1 单个实体增加

var obj = Repository.Add(new User()

{

Name = "测试",

UserName = "Add",

PassWord = "Add",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

});

//INSERT INTO [User] (Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime)

//SELECT @Name,@UserName,@PassWord,@Sex,@BirthDay,@CreateTime,@UpdateTime

4.2 单个实体增加（不存在时）

var obj = Repository.AddIfNotExists(new User()

{

Name = "测试",

UserName = "AddIfNotExists",

PassWord = "AddIfNotExists",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, u => u.UserName == "AddIfNotExists");

//INSERT INTO [User] (Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime)

//SELECT @Name,@UserName,@PassWord,@Sex,@BirthDay,@CreateTime,@UpdateTime

//WHERE NOT EXISTS(

// SELECT TOP 1 Id,Name,UserName,PassWord,Sex,BirthDay,CreateTime,UpdateTime,Status

// FROM [User]

// WHERE (UserName = @P0))

4.3 批量增加

List<User> us = new List<User>();

for (int i = 0; i < 10000; i++)

{

us.Add(new User()

{

Name = "测试",

UserName = "AddList" + i,

PassWord = "AddList" + i,

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

});

}

Repository.AddList(us);

**五、删除**

Repository.Delete(new User() { Id = 1 });

//DELETE FROM [User] WHERE Id=@Id

Repository.Delete(w => w.Id > 0);

//DELETE FROM [User] WHERE (Id > @P0)

**六、修改**

6.1 单体修改

（注）单体修改会默认按照主键作为条件修改数据，当实体没有设置主键时，会抛出异常

var obj = Repository.Update(new User()

{

Id = 2,

Name = "测试",

UserName = "AddList",

PassWord = "AddList",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

});

6.2 条件修改

var obj = Repository.Update(c => c.Id > 0, o => new User() { Sex = true, Name = "asdas" });

//UPDATE [User] SET Name=@Name,UserName=@UserName,PassWord=@PassWord,Sex=@Sex,BirthDay=@BirthDay,CreateTime=@CreateTime,UpdateTime=@UpdateTime WHERE Id=@Id

6.3 随机条件修改

（注）此方法会随机选择指定数量的数据进行更新为指定实体值，并通过isInserted来确定是返回修改后的数据还是修改前的数据

var obj3 = Repository.UpdateSelect<User>(w => w.Id > 0,

o => new User()

{

Name = "测试",

UserName = "UpdateSelect",

PassWord = "UpdateSelect",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, null, 10);

UPDATE TOP(10) [User] WITH(UPDLOCK, READPAST) SET Sex = @Sex, Name = @Name OUTPUT INSERTED.Id AS Id, INSERTED.Name AS Name, INSERTED.UserName AS UserName, INSERTED.PassWord AS PassWord, INSERTED.Sex AS Sex, INSERTED.BirthDay AS BirthDay, INSERTED.CreateTime AS CreateTime, INSERTED.UpdateTime AS UpdateTime WHERE (Id > @P0)

**七、事务**

使用using (var tran = new UnitOfWork().BeginTransaction())创建事务，代码块中调用Sql语句的事务重载方法，结束后必须要调用tran.Commit();才会生效

7.1 本地事务

using (var tran = new UnitOfWork().BeginTransaction())

{

Repository.Add(new User()

{

Name = "测试",

UserName = "Add\_LoacalTransaction",

PassWord = "Add\_LoacalTransaction",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, tran);

Repository.AddIfNotExists(new User()

{

Name = "测试",

UserName = "AddIfNotExists\_LoacalTransaction",

PassWord = "AddIfNotExists\_LoacalTransaction",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, u => u.UserName == "AddIfNotExists\_LoacalTransaction", tran);

tran.Commit();

}

7.2 本地分布式事务（本地有多个数据库时）

using (var unitOfWork = new UnitOfWork())

{

Repository.Add(new User()

{

Name = "测试",

UserName = "Add\_LoacalDisTransaction",

PassWord = "Add\_LoacalDisTransaction",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, unitOfWork.BeginTransaction());

Repository2.AddIfNotExists(new User2()

{

Name = "测试",

Guid=Guid.NewGuid().ToString(),

UserName = "AddIfNotExists\_LoacalTransaction",

PassWord = "AddIfNotExists\_LoacalTransaction",

Sex = true,

BirthDay = DateTime.Now,

CreateTime = DateTime.Now,

UpdateTime = DateTime.Now

}, u => u.UserName == "AddIfNotExists\_LoacalTransaction", unitOfWork.BeginTransaction());

unitOfWork.Commit(); //此种方式，仅回滚出错的操作

unitOfWork.Commit(TransactionType.LocalDistribute);//此种方式，回滚所有操作

}

**五、存储过程**

var obj=Repository.QueryStoreProcedure<User>("UP\_GetRecordByPage", new DbParameter[] {

new SqlParameter("@tblName","User"),

new SqlParameter("@fldName","Id"),

new SqlParameter("@PageIndex","3"),

new SqlParameter("@strWhere","Id>0"),

});

//返回一个Int结果，只支持整型

var obj2 = Repository.StoreProcedure("UP\_GetRecordByPage",, new DbParameter[] {

new SqlParameter("@tblName","User"),

new SqlParameter("@fldName","Id"),

new SqlParameter("@PageIndex","3"),

new SqlParameter("@strWhere","Id>0"),

});