**Elfin Tutorial**

Author: Wilson Fan

Date: 8/10/2018

Introduction

Elfin is a high-performance Micro-ORM that supports SQL Server, MySQL, and Access. This tutorial will introduce you to the basic operations of using Elfin.

Link

Link to the latest version and sample files for this document：http://github.com/wilsonfanfan/Elfin/

Need help? Have a question?

Send email to : [wilson.fan@yahoo.com](mailto:wilson.fan@yahoo.com)

Looking for examples

* Many samples are available at http://github.com/wilsonfanfan/
* You can also send an email message to: [wilson.fan@yahoo.com](mailto:wilson.fan@yahoo.com)

About Elfin

**Elfin** is a NuGet library that you can add in to your project. The emergence of Elfin is not just an choose, maybe he is not perfect, and more hope that everyone can join in to improve and expand.

**Elfin** has no third-party library dependencies, It provides simpler physical operations.

Preparation

**Step 1: Install the Elfin Package**

**To install Elfin : run the following command in the Package Manager Console: PM> Install-Package Elfin -Version 1.2.6 And Install-Package Elfin.Mapping -Version 1.0.3 OR Nuget finds Elfin and Elfin.Mapping and initializes.**

**Step 2: Modify configuration file**

Before we started, we need still to do some configuration.

Find **PersistenceEntity.xml** , Modify the database connection configuration, **Like below:**

<?xml version='1.0' encoding='utf-8'?>

<configuration>

<connectinfo>

<parameter name="dbtype" value="sqlserver" />

<parameter name="server" value="." />

<parameter name="database" value="Test" />

<parameter name="userid" value="sa" />

<parameter name="password" value="test123123" />

</connectinfo>

<commoninfo>

<parameter name="outputdir" value="D:\TestProject\ElfinConsoleApp\" />

</commoninfo>

<customerinfo>

<parameter name="systemname" value="ElfinConsoleApp" />

<parameter name="systemid" value="ElfinConsoleApp" />

<parameter name="developcampany" value="2018" />

<parameter name="namespace" value="Entity" />

<parameter name="language" value="C#" />

</customerinfo>

</configuration>

**The PersistenceEntity.xml** Contains data information for the connection，He will be used when generating the mapping file.

**Step 3: Generation Mapping File**

In order to facilitate testing, we created an app. Generate a mapping file by referring to the following code:

using Elfin.Mapping;

namespace ElfinConsoleApp

{

internal class Program

{

internal static void Main(string[] args)

{

//Test database connection

MappingHelper.DatabaseInitialization();

//Generation Mapping File

MappingHelper.MappingAndIgnorePrimaryKey();

}

}

**The path of the generated file is the value of the outputdir node in PersistenceEntity.xml.**

After the method returns a successful result, we can find the generated mapping file in the output folder. They are, respectively:

**Entity (folder)**

**EntityManager (folder)**

**ApplicationConfig.xml**

**ORMapping.xml**

The **Entity** folder and the **EntityManager** folder contain database table mapping information and entity base methods respectively.

**ApplicationConfig.xml** Contains database connections and some global properties.

**ORMapping.xml** is the mapping information of the database table.

**Back to program,** **Open the App.config file and add nodes in appSettings ConfigPath and MapPath, The value of ConfigPath is the absolute path where ApplicationConfig.xml and Sys.ORMapping.xml are located. The value of MapPath is the absolute path where the program root directory is located. like this:**

<appSettings>

<add key="ConfigPath" value="D:\TestProject\TestConsoleApp\App\_Config\" />

<add key="MapPath" value="D:\TestProject\TestConsoleApp\" />

</appSettings>

**I usually put them in the program root directory App\_Config folder.**

The **Entity** and **EntityManager** folders can be placed in the corresponding project as needed, and we will use them in subsequent operations.

Classes under Entity and EntityManager need to reference the Elfin class library

Simple query

**Query by Primary Key**

//Get user information based on user ID

b\_User user = new b\_UserManager().GetEntityObject("wilson");

if (user.IsPersistent)

{

Console.WriteLine(user.UserID);

Console.WriteLine(user.UserName);

//....

}

**Or we can do this**

//Get user information based on user ID

b\_User user = new b\_User();

user.UserID = "wilson"; //UserID must be the primary key

user.Retrieve();

if (user.IsPersistent)

{

Console.WriteLine(user.UserID);

Console.WriteLine(user.UserName);

//....

}

**Multiple conditional queries**

//Query users whose UserID is wilson and whose gender is male

ElfinDictionary elfinDic = new ElfinDictionary();

elfinDic.Add(b\_User.F\_USERID,"wilson");

elfinDic.Add(b\_User.F\_GENDER,"male");

var users = new b\_UserManager().GetEntityObjects(elfinDic);

//Query name contains w data

RetrieveCriteria retrieveCriteria = new RetrieveCriteria(typeof(b\_User));

IFilter filter = retrieveCriteria.GetFilter();

string namePrefix = "w";

filter.AddCustomerCompare($"UserID like '%{namePrefix}%'");

var list = retrieveCriteria.GetCollection();

var dt = retrieveCriteria.GetDataTable();

var cursor = retrieveCriteria.GetCursor();

//Get all user

var users = new b\_UserManager().GetEntityObjects();

**Query part of the field**

//Query name contains data for w and gender non-male

new EntityManager();

Query query = new Query(typeof(b\_User));

query.AddFields(b\_User.F\_USERID);

query.AddFields(b\_User.F\_USERNAME);

var filter = query.GetWhere();

string namePrefix = "w";

filter.AddCustomerCompare(($"UserID like '%{namePrefix}%'"));

filter.AddNotEqualTo(b\_User.F\_GENDER,"male");

var dt=query.Execute();

**Excute Sql Query**

new EntityManager();

string strSql = @"select \* from b\_User

inner join r\_Score

on b\_User.UserID=r\_Score.UserID";

var dt = Query.ExecuteSQLQuery(strSql,SystemEnvironment.Instance.DefaultDataSource);

The **Query** class provides a variety of query operations waiting for everyone to discover and use.

Entity-based single table operation

**Insert**

//Add a new user data to the database

b\_User b\_User = new b\_User();

b\_User.UserID = "584280962@qq.com";

b\_User.UserName = "wilson ";

new b\_UserManager().InsEntityObject(b\_User);

**Or we can do this**

//Add a new user data to the database

new EntityManager();

b\_User b\_User = new b\_User();

b\_User.UserID = "584280962@qq.com";

b\_User.UserName = "wilson ";

b\_User.Insert();

**Update**

//Update UserName to wilson.fan

b\_User user = new b\_UserManager().GetEntityObject("584280962@qq.com");

user.UserName = "wilson.fan";

user.Update();

Need to include the operation in a transaction? Look at the code below:

Transaction ts = new Transaction();

try

{

b\_User user = new b\_UserManager().GetEntityObject("584280962@qq.com");

user.UserName = "wilson.fan";

user.Update(ts);

r\_Score score = new r\_ScoreManager().GetEntityObject("584280962@qq.com");

score.Score = 90;

score.Update(ts);

ts.CommitTransaction();

}

catch

{

ts.RollbackTransaction();

}

**Delete**

//Delete user

b\_User user = new b\_UserManager().GetEntityObject("584280962@qq.com");

user.Delete();

The database connection must be initialized before the entity operation is performed.Like below:

new EntityManager();

b\_User user = new b\_User();

user.UserID = "wilson";

user.Insert();

user.Update();

user.Delete();

user.Retrieve();

**Use new b\_UserManager() is a also method.**

**Let’s look at the delete operation below:**

new EntityManager();

DeleteCriteria deleteCriteria = new DeleteCriteria(typeof(b\_User));

IFilter filter = deleteCriteria.GetFilter();

filter.AddMatch(b\_User.F\_USERNAME,"w");

filter.AddNotEqualTo(b\_User.F\_USERID,"male");

int result=deleteCriteria.Execute();

Console.WriteLine(deleteCriteria.SqlString);

What is the result of **deleteCriteria.SqlString**?

Each entity has an **IsPersistent** attribute that is used to mark entities. IsPersistent is **TRUE** if it exists in the database, otherwise **FALSE.**

About **Transaction**, Most of the methods of operation in ELFIN provide a transaction interface, and we can refer to it in the things we need.

For reasons of time, these are some of the basic usage methods, including more design and summary information, which will be updated in future work. If you are using ELFIN now, or if you have problems during use, you can email me. I will reply to your question within 24 hours, thank you!