Nhibernate 4.0 教程

目录

[1. 下载Nhibernate 4.04 1](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259173)

[2. 入门教程 2](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259174)

[3. 测试项目详解 3](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259175)

[4. 总结 7](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259176)

[附：关联知识点 7](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259177)

[知识点1：C#静态构造函数 7](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259178)

[知识点2：关于Visual Studio文件生成操作 8](file:///C:\Users\yang\Desktop\NHibernate\WCF.docx#_Toc467259179)

前言：

为什么会有这个框架？这就牵扯进了Java和C#的恩恩怨怨，Java是开源的面向对象语言的代表，围绕Java的有不计其数的开源框架，其中ORM框架（不要问我什么是ORM框架）中的最耀眼的代表就是Hibernate；C#也是Microsoft紧跟在Java后面推出的面向对象的语言，这两个相似度太大了（我读书自己学习的Java，后面C#就没特殊的学习过，直接就进行拿来用了），.NET开发者也参照Hibernate开发了一个针对.NET平台下的ORM 框架，也就是Nhibernate。

开发环境：

Windows 7

Visual Studio 2013

Nhibernate 4.04

Microsoft Sql Server 2012

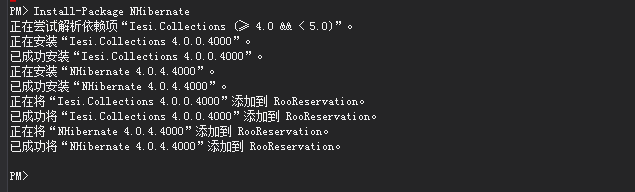
1. 下载Nhibernate 4.04

直接下载 官网地址<http://nhibernate.info/>（自己下载网速真的好慢）

NHibernate is a mature, open source object-relational mapper for the .NET framework. It's actively developed, fully featured and used in thousands of successful projects.

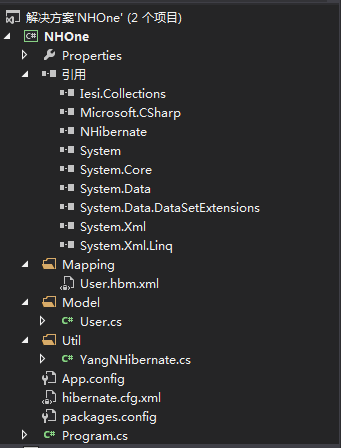
官网介绍了Nhibernate是针对.NET框架的成熟的、开源的面向关系型数据库映射（ORM）.

或者使用VS2013附带的NuGet管理程序直接安装（NuGet，.NET下面一个开源的程序包管理工具）：Install-Package Nhibernate(非常的快)



1. 入门教程

新建一个项目NHOne和测试程序，并且添加对于该项目的测试项目，项目架构如下：



划重点：其中对于Nhibernate的配置文件（hibernate.cfg.xml与ClassMapping文件，对于生成的操作，必须选择始终复制和嵌入式资源。（不然编译调试的时候会出现bug，比如没有Model Class的Mapping等等）。

1. 测试项目详解
   1. 初始化C#解决方案NHOne（控制台应用程序和对应的测试项目）

给每一个项目添加Nhibernate引用，测试项目也需要（直接NuGet命令安装即可）

* 1. 编写NHibernate配置文件

在NHOne的根目录下添加hibernate.cfg.xml文件（生成操作嵌入式资源和始终复制），在文档属性中架构选择hibernate-configuration-2.2 & hibernate-mapping-2.2两个文件，这样编写hibernate.cfg.xml就会有自动提示的功能（这两个文件在我们解决方案NHOne的packages/Nhibernate目录中）



hibernate.cfg.xml 具体内容如下：

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

<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">

<session-factory>

<property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>

<property name="connection.driver\_class">NHibernate.Driver.SqlClientDriver</property>

<property name="connection.connection\_string">Data Source=localhost;user=sa;password=12345;Initial Catalog=test</property>

<property name="dialect">NHibernate.Dialect.MsSql2012Dialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<!--Mapping files 嵌入式资源 表格table 名字不可以是user-->

<mapping assembly="NHOne"/>

</session-factory>

</hibernate-configuration>

<mapping assembly=”NHOne”/> 这里写的是项目的名字，然后就会搜寻项目下面所有的后缀是.hbm.xml的文件，也可以手动自己添加，比较麻烦，有机会再详细讲解。

* 1. 编写Model Class “User”

所有的属性都是使用virtual来修饰（延迟加载有用），添加对应的get set方法

在MSServer 中，user 是关键字，因此不能够有user的表格，可以使用其他来代替

在实际应用中，每一个model类，很可能需要提供Equals() HashCode() ToString()方法的重写

namespace NHOne.Model

{

public class User

{

private string id;

private string username;

private string password;

private char gender; //"F" & "M"

private int age;

private string phone;

public User()

{

}

public User(string username, string password, char gender, int age, string phone)

{

this.username = username;

this.password = password;

this.gender = gender;

this.age = age;

this.phone = phone;

}

public User(string id, string username, string password, char gender, int age, string phone)

{

this.id = id;

this.username = username;

this.password = password;

this.gender = gender;

this.age = age;

this.phone = phone;

}

public virtual string Id { get; set; }

public virtual string Username { get; set; }

public virtual string Password { get; set; }

public virtual char Gender { get; set; }

public virtual int Age { get; set; }

public virtual string Phone { get; set; }

public override bool Equals(object obj)

{

if (this == obj)

{

return true;

}

User user = obj as User;

if (user == null)

{

return false;

}

else

{

return this.Username.Equals(user.Username);

}

}

public override int GetHashCode()

{

return Username.GetHashCode();

}

public override string ToString()

{

return "id:" + Id + "; Username:" + Username + "; Password:" + Password + "; Gender:" + Gender + "; Age:" + Age + "; Phone:" + Phone;

}

}

}

* 1. 编写Model的Mapping文件

位于Mapping目录下面新建文件User.hbm.xml文件（嵌入式资源，添加文件的架构为hibernate-mapping-2.2.xml），红色部分 assembly是项目的名称，namespace是在项目中Model类的命名空间，如果省略的话，需要在后面的class name属性中填写完整的名字空间和类名。

注意table的名字不能写成user，否则报错，因为user是数据库的关键字

内容如下：

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

<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="NHOne" namespace="NHOne.Model" default-lazy="true">

<class name="User" table="client">

<id name="Id">

<column name="user\_id" sql-type="char(32)" not-null="true"/>

<generator class="uuid.hex"/>

</id>

<property name="Username" column="username" not-null="true"/>

<property name="Password" column="password" not-null="true" />

<property name="Gender" column="gender" />

<property name="Age" column="age" />

<property name="Phone" column="phone" />

</class>

</hibernate-mapping>

* 1. 编写NHibernate的帮助类，初始化NHibernate的环境，获取Session

在Util命名空间下，添加YangNHibernate的类，具体内容如下:

namespace NHOne.Util

{

public class YangNHibernate

{

private static readonly ISessionFactory sessionFactory;

private static string HibernateHbmXmlFileName = "hibernate.cfg.xml";

//private static ISession session

static YangNHibernate()

{

sessionFactory = new Configuration().Configure().BuildSessionFactory();

}

public static ISessionFactory getSessionFactory()

{

return sessionFactory;

}

public static ISession getSession()

{

return sessionFactory.OpenSession();

}

public static void closeSessionFactory()

{

}

}

}

* 1. 编写测试代码

在测试类中测试保存User到数据库，使用事务的机制

[TestMethod]

public void TestSaveUser()

{

User user = createUser();

ISession session = YangNHibernate.getSession();

ITransaction tx = session.BeginTransaction();

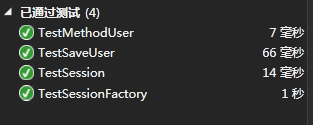
session.Save(user);

tx.Commit();

session.Close();

}

调试程序：



1. 总结

NHibernate是C#程序员，参照Java的ORM框架Hibernate实现的一个开源项目，在C#中项目开发中非常的便捷高效，让程序猿从复杂的SQL语句中解放出来，对于项目的模块化非常好用。

测试项目的源代码地址：

<https://github.com/hbhzsysutengfei/NHibernateOne.git>

参考文档：

1. NHibernate官方网站 <http://nhibernate.info/>
2. MSDN for C#：<https://msdn.microsoft.com/zh-cn/library/k9x6w0hc(v=vs.120).aspx>

附：关联知识点

知识点1：C#静态构造函数

C#关于静态构造函数的知识，自动调用初始化静态的成员属性，尤其是readonly修饰的静态属性，与Java中的静态代码块类似。

参考MSDN给出的解释：

A static constructor is used to initialize any [static](https://msdn.microsoft.com/zh-cn/library/98f28cdx(v=vs.120).aspx) data, or to perform a particular action that needs to be performed once only. It is called automatically before the first instance is created or any static members are referenced.

（<https://msdn.microsoft.com/zh-cn/library/k9x6w0hc(v=vs.120).aspx>）

因为静态构造函数是.NET自动调用的，因此改代码块就无需使用修饰符修饰。

关于C#静态构造函数的特征，具体如下：

A static constructor does not take access modifiers or have parameters.

A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.

A static constructor cannot be called directly.

The user has no control on when the static constructor is executed in the program.

A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.

Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method.

If a static constructor throws an exception, the runtime will not invoke it a second time, and the type will remain uninitialized for the lifetime of the application domain in which your program is running.

知识点2：关于Visual Studio文件生成操作

Visual Studio生成操作有以下几个选项：

Non（无）：也就是在项目生成的时候不进行任何的操作，在生成的目录中没有此文件，一般用于项目描述文件.

Content(内容)：也就是将文件直接复制到输出目录中，一般用于html等静态文件.

Compile(编译):编译代码文件，通常的代码文件需要编译

Embedded Resource(嵌入式资源)：将该文件作为DLL或可执行文件嵌入到主目录中，通常用于资源文件，比如Nhibernate的配置文件等等。