# **Photon Server游戏服务器教程(四) NHibernate 对数据库的操作**

NHibernate 是一个基于.Net 的针对关系型数据库的对象持久化类库。NHibernate 来源于非常优秀的基于Java的Hibernate 关系型持久化工具。

NHibernate 从数据库底层来持久化你的.Net 对象到关系型数据库。NHibernate 为你处理这些，远胜于你不得不写SQL去从数据库存取对象。你的代码仅仅和对象关联，NHibernat 自动产生SQL语句，并确保对象提交到正确的表和字段中去。

**1、添加引用**

在vs2013项目中，右键解决方案选择“管理NuGet程序包”，搜索NHibernate安装添加其引用，或在官网http://nhibernate.info下载安装。

**2、NHibernate数据库连接配置**

在项目下添加新建项XML文件，名称为hibernate.cfg.xml（固定名称），修改属性“复制到输出目录”为始终复制，添加如下配置格式：

&lt;?xml version="1.0" encoding="utf-8" ?&gt;

&lt;hibernate-configuration xmlns="urn:nhibernate-configuration-2.2"&gt;

&lt;session-factory&gt;

&lt;property name="connection.provider"&gt;NHibernate.Connection.DriverConnectionProvider&lt;/property&gt;

&lt;property name="dialect"&gt;NHibernate.Dialect.MySQL5Dialect&lt;/property&gt;&lt;!--数据库版本--&gt;

&lt;property name="connection.driver\_class"&gt;NHibernate.Driver.MySqlDataDriver&lt;/property&gt; &lt;!--使用什么数据库--&gt;

&lt;property name="connection.connection\_string"&gt;Server=localhost;Database=mygamedb;User ID=root;Password=root;&lt;/property&gt; &lt;!--数据库连接Server--&gt;

&lt;property name="show\_sql"&gt;true&lt;/property&gt;

&lt;/session-factory&gt;

&lt;/hibernate-configuration&gt;

**3、类和表的映射**

在项目下新建文件夹Model，添加User.cs类，代码如下：

public class User ｛

public virtual int Id ｛ get; set; ｝

public virtual string Username ｛ get; set; ｝

public virtual string Password ｛ get; set; ｝

public virtual DateTime Registerdate ｛ get; set; ｝

｝

再添加一个映射文件夹Mappings，新建一个XML配置文件User.hbm.xml(类名)，修改属性“生成操作”为嵌入的资源，打开文件添加如下配置格式：

&lt;?xml version="1.0" encoding="utf-8" ?&gt;

&lt;hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"

assembly="ServerSql"

namespace="ServerSql.Model"&gt;

&lt;!--assembly：程序集--&gt;

&lt;!--namespace：命名空间--&gt;

&lt;class name="User" table="users"&gt;

&lt;id name="Id" column="id" type="Int32"&gt; &lt;!--ID:配置主键, column: 表里的列, type: NHibernate 类型 http://nhibernate.info/doc/nhibernate-reference/mapping.html#mapping-types--&gt;

&lt;generator class="native"&gt;&lt;/generator&gt; &lt;!--自增长--&gt;

&lt;/id&gt;

&lt;property name="Username" column="username" type="String"&gt;&lt;/property&gt;

&lt;property name="Password" column="password" type="String"&gt;&lt;/property&gt;

&lt;property name="Registerdate" column="registerdate" type="Date"&gt;&lt;/property&gt;

&lt;/class&gt;

&lt;/hibernate-mapping&gt;

**4、解析数据库链接配置文件和表映射文件**

var configuration = new Configuration();

configuration.Configure();//解析nhibernate.cfg.xml

configuration.AddAssembly(Assembly.GetExecutingAssembly());//解析 映射文件 User.hbm.xml

**5、创建连接session进行添加操作**

ISessionFactory sessionFactory = null;

ISession session = null;

try｛

sessionFactory= configuration.BuildSessionFactory();

session = sessionFactory.OpenSession();//打开一个跟数据库的会话，即连接数据库

//进行一些操作

User user = new User() ｛ Username = "yeet", Password = "123564" ｝;

session.Save(user);

session.Flush();

｝

catch (Exception e)｛

Console.WriteLine(e);

｝

finally

｛

if (session != null) ｛

session.Close();

｝

if (sessionFactory != null)｛

sessionFactory.Close();

｝

｝

**6、用NHibernate进行事务操作**

什么时候使用事务？

在任何时候都要使用事务，即使是在读取、查询数据的时候，为什么呢？因为你不清楚数据库什么时候操作失败，如何恢复原来数据。而NHibernate中的事务（可以通过 transaction.Rollback()方法），帮助我们完成这些事情。

ITransaction transaction= session.BeginTransaction();

//进行操作

User user1 = new User() ｛ Username = "cydr34", Password = "g3463" ｝;

session.Save(user1);

transaction.Commit();

//使用完事务需要关闭

transaction.Dispose();

**7、创建NHibernateHelper管理会话工厂**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using NHibernate.Cfg;

public class NHibernateHelper ｛

private static ISessionFactory \_sessionFactory;

private static readonly object lockObj = new object();

private NHibernateHelper()｛｝

private static ISessionFactory SessionFactory｛

get ｛

if ( \_sessionFactory == null ) ｛

lock ( lockObj ) ｛

if (\_sessionFactory == null) ｛

var configuration = new Configuration();

configuration.Configure();

configuration.AddAssembly(Assembly.GetExecutingAssembly());

\_sessionFactory = configuration.BuildSessionFactory();

｝

｝

｝

return \_sessionFactory;

｝

｝

public static ISession OpenSession() ｛

return SessionFactory.OpenSession();

｝

｝

**8、创建管理接口**

interface IUserManager ｛

void Add(User user);

void Update(User user);

void Remove(User user);

User GetById(int id);

User GetByUsername(string username);

ICollection&lt;User&gt; GetAllUsers();

bool VerifyUser(string username, string password);

｝

class UserManager:IUserManager ｛

public void Add(Model.User user)｛

using (ISession session = NHibernateHelper.OpenSession())｛

using (ITransaction transaction = session.BeginTransaction())｛

session.Save(user);

transaction.Commit();

｝

｝

｝

public void Update(Model.User user) ｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

using (ITransaction transaction = session.BeginTransaction())｛

session.Update(user);

transaction.Commit();

｝

｝

｝

public void Remove(Model.User user) ｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

using (ITransaction transaction = session.BeginTransaction()) ｛

session.Delete(user);

transaction.Commit();

｝

｝

｝

public Model.User GetById(int id) ｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

using (ITransaction transaction = session.BeginTransaction()) ｛

User user = session.Get&lt;User&gt;(id);

transaction.Commit();

return user;

｝

｝

｝

public Model.User GetByUsername(string username) ｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

User user = session.CreateCriteria(typeof(User)).Add(Restrictions.Eq("Username", username)).UniqueResult&lt;User&gt;();

return user;

｝

｝

public ICollection&lt;Model.User&gt; GetAllUsers() ｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

IList&lt;User&gt; users = session.CreateCriteria(typeof(User)).List&lt;User&gt;();

return users;

｝

｝

public bool VerifyUser(string username, string password)｛

using (ISession session = NHibernateHelper.OpenSession()) ｛

User user = session

.CreateCriteria(typeof(User))

.Add(Restrictions.Eq("Username", username))

.Add(Restrictions.Eq("Password", password))

.UniqueResult&lt;User&gt;();

if (user == null) return false;

return true;

｝

｝

｝

**9、测试**

User user = new User() ｛ Username = "qwqjer", Password = "2544" ｝;

IUserManager userManager = new UserManager();

userManager.Add(user);

来自：https://blog.csdn.net/u010812661/article/details/78760394