using System;

using System.Collections.Generic;

using System.Drawing;

using System.Reflection;

using System.Runtime.Remoting.Messaging;

using System.ServiceModel;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using WCFConnect.ServiceClient;

namespace WCFConnect

{

public partial class Form1 : Form

{

private static InstanceContext instanceContex { set; get; }

private static ServiceContractClient client { set; get; }

private static appleAcount userInfo { set; get; }

GroupBox gb = new GroupBox();

PropertyInfo[] propertyInfo { set { } get { return T.GetProperties(); } }

private static Type T { set { } get { return userInfo.GetType(); } }

private string Order { set; get; }

public Form1()

{

InitializeComponent();

client = new ServiceContractClient();

}

//获得所有的用户信息并绑定到dataGridView上

private void btnConWcf\_Click(object sender, EventArgs e)

{

var allUserInfos = client.GetAllUserInfos();

this.dataGridView1.DataSource = allUserInfos;

for (int i = 0; i < this.dataGridView1.Columns.Count; i++)

{

if (!this.dataGridView1.Columns[i].ToString().Contains("login"))

{

this.dataGridView1.Columns[i].Visible = false;

}

}

}

//通过ID获得用户信息

private void btnUserInfo\_Click(object sender, EventArgs e)

{

userInfo = client.GetAUserInfo(Convert.ToInt32(txtID.Text));

GetTextBox(userInfo);

}

//BUTTON修改用户信息

private void btnEditUserInfo\_Click(object sender, EventArgs e)

{

#region getDictionary

Dictionary<string, string> dic = new Dictionary<string, string>();

string a = null;

string b = null;

foreach (Control item in gb.Controls)

{

if (item is TextBox & item.Tag.ToString().Contains("name"))

{

a = item.Text;

}

else if (item is TextBox & item.Tag.ToString().Contains("value"))

{

b = item.Text;

}

if (a != null & b != null)

{

dic.Add(a, b);

a = null;

b = null;

}

}

#endregion

for (int i = 0; i < propertyInfo.Length; i++)

{

foreach (var item in dic)

{

if (item.Key == propertyInfo[i].Name)

{

PropertyInfo P = propertyInfo[i];

try

{

P.SetValue(userInfo, item.Value);

}

catch

{

}

}

}

}

client.EditUserInfo(userInfo);

}

//BUTTON随机获得一个用户信息

private void btnCirclyGetUserInfo\_Click(object sender, EventArgs e)

{

userInfo = client.GetAUserInfo(9999);

GetTextBox(userInfo);

}

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

{

ThreadPool.QueueUserWorkItem((a) =>

{

while (true)

{

this.Invoke(new Action(() =>

{

labHostState.Text = client.GetOrder();

Order = client.GetOrder();

if (Order == "同步登陆" | Order == "登陆")

{

labOper.Text = "同步登陆中";

string txt = LoginTest(client).remarks;

labOper.Text = txt;

GetTextBox(userInfo);

}

else if (Order == "异步登陆")

{

labOper.Text = "异步登陆中";

string txt =AsyncLoginTest();

labOper.Text = txt;

}

}));

Thread.Sleep(3000);

}

});

}

#region 异步登陆

#region 方法一

private void asyGetUserInfo\_Click(object sender, EventArgs e)

{

AsyncLoginTest();

}

delegate appleAcount delegateTryAsyncLoginTest(ServiceContractClient client);

private string AsyncLoginTest()

{

delegateTryAsyncLoginTest asyncDelegate = new delegateTryAsyncLoginTest(LoginTest);

string t = "";

asyncDelegate.BeginInvoke(client, new AsyncCallback(asyncResult =>

{

AsyncResult result = (AsyncResult)asyncResult;

delegateTryAsyncLoginTest getOriginalDelegate = (delegateTryAsyncLoginTest)result.AsyncDelegate;//获得原委托

appleAcount dataOfLoginTest = getOriginalDelegate.EndInvoke(result);//执行原委托的EndInvoke方法，获得LoginTest方法执行后的结果

GetTextBox(dataOfLoginTest);

t = dataOfLoginTest.remarks;

}), null);

return t;

}

#endregion

#endregion

// 把userInfo映射成表以textBox的方式显示出来

private void GetTextBox(appleAcount user)

{

Task.Factory.StartNew(() =>

{

#region Invoke

//新建的工作线程是非创建Form窗体的窗口线程，所以不能直接调用窗口线程。

//开启一个一部委托捆绑要执行的方法。

this.Invoke(new Action(() =>

{

gb.Dispose();

gb = new GroupBox();

int x = 40;

int y = 80;

gb.Width = 900;

gb.Height = 800;

this.Controls.Add(gb);

for (int i = 0; i < propertyInfo.Length; i++)

{

TextBox tb = new TextBox();

tb.Tag = "name";

tb.Text = propertyInfo[i].Name;

tb.Location = new Point(x, y);

tb.Width = 130;

tb.Height = 20;

gb.Controls.Add(tb);

TextBox tbValue = new TextBox();

tbValue.Tag = "value";

try { tbValue.Text = propertyInfo[i].GetValue(user).ToString(); }

catch { }

tbValue.Location = new Point(x + 130, y);

tbValue.Width = 200;

tbValue.Height = 20;

gb.Controls.Add(tbValue);

y += 20;

}

}));

#endregion

});

}

// 登陆测试，登陆成功后，修改数据库remarks=“登陆成功”并返回用户信息

private static appleAcount LoginTest(ServiceContractClient client)

{

appleAcount user = client.LoginTest();

userInfo = user;

OrderMethods orderMethod = new OrderMethods();

string loginResult = orderMethod.LoginTest(user.loginAppleId, user.loginPassword);

user.remarks = loginResult;

client.EditUserInfo(user);

return user;

}

}

}

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />

</startup>

<system.serviceModel>

<bindings>

<wsDualHttpBinding>

<binding name="WSDualHttpBinding\_IServiceContract" maxReceivedMessageSize="2147483647"

useDefaultWebProxy="false">

<security mode="None" />

</binding>

</wsDualHttpBinding>

</bindings>

<client>

<endpoint address="http://192.168.31.29:8001/getSql" binding="wsDualHttpBinding"

bindingConfiguration="WSDualHttpBinding\_IServiceContract" contract="ServiceClient.IServiceContract"

name="WSDualHttpBinding\_IServiceContract" />

</client>

</system.serviceModel>

</configuration>

using System;

using System.Diagnostics;

using System.Threading;

using WCFParallelTest.ServiceClient;

namespace WCFParallelTest

{

class Program

{

private static ServiceContractClient client = new ServiceContractClient();

static void Main(string[] args)

{

int t = 1;

Stopwatch sw = new Stopwatch();

sw.Start();

while (client.GetOrder()=="写入测试")

{

appleAcount user = client.ParallelTest();

Console.WriteLine("ID:" + user.ID + "\t");

user.district += "0";

user.cookies = "已写入";

int x=client.EditUserInfo(user);

Console.Write("写入次数：" + t + "\t" + "写入时间间隔：" + sw.ElapsedMilliseconds+"\tID:"+x+"\r\n");

Thread.Sleep(5000);

t++;

user = client.ParallelTest();

sw.Restart();

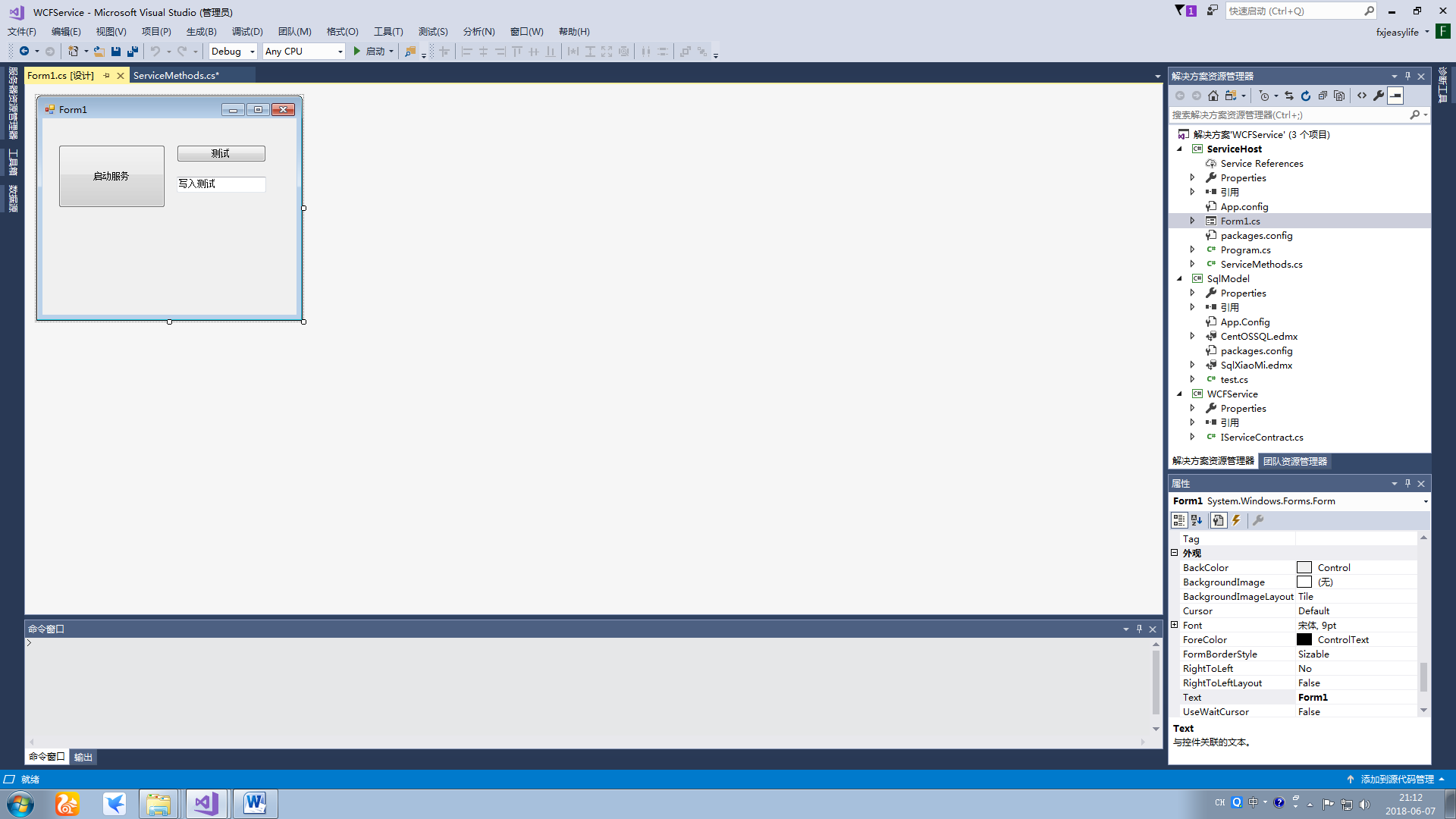
}

Console.Read();

}

}

}



using SqlModel;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.ServiceModel;

using System.Windows.Forms;

using WCFService;

namespace ServiceHost

{

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]

public class ServiceMethods : IServiceContract

{

private static DbContext db { set; get; }

private static DbSet<appleAcount> dbSet { set; get; }

private static appleAcount userInfo { set; get; }

private static Button btn { set; get; }

private static System.ServiceModel.ServiceHost host { set; get; }

private static int Count { set; get; }

private string \_order;

public string Order

{

set

{

\_order = Order;

}

get

{

if (btn.Text!="")

{

return btn.Text;

}

else

{

return "无命令";

}

}

}

public ServiceMethods()

{

db = new CENTOS7SQLEntities();

dbSet = db.Set<appleAcount>();

Count = dbSet.Count<appleAcount>();

}

//开启服务，将按键Text改为"已启动服务"。

public void start(Button f)

{

if (host==null)

{

host = new System.ServiceModel.ServiceHost(typeof(ServiceMethods));

}

btn = f;

//&f 0x0963e4cc System.Windows.Forms.Button&\*

Order = btn.Text;

if (host.State!=CommunicationState.Opened)

{

host.Open();

f.Text = "已启动服务";

}

}

//更新命令

public void refresh(string txtOrder)

{

if (host == null)

{

host = new System.ServiceModel.ServiceHost(typeof(ServiceMethods));

}

Order = txtOrder;

btn.Text = txtOrder;

if (host.State != CommunicationState.Opened)

{

btn.Text = "未启动服务";

}

}

//获得指令

public string GetOrder()

{

return Order;

}

public DateTime GetDateTime()

{

return DateTime.Now;

}

//获得所有的用户信息

public List<appleAcount> GetAllUserInfos()

{

List<appleAcount> tab = db.Set<appleAcount>().Where(a => true).ToList();

return tab;

}

//根须id获得用户信息

public appleAcount GetAUserInfo(int id)

{

if (id == 9999)

{

Random rad = new Random();

int i = rad.Next(Count);

userInfo = dbSet.Where(a => a.ID == i).FirstOrDefault();

}

else

{

userInfo = dbSet.Where(a => a.ID == id).FirstOrDefault();

}

return userInfo;

}

//返回用于登陆的用户信息

public appleAcount Login()

{

userInfo = dbSet.Where((u) => u.remarks != "登录成功" & u.remarks != "ID被禁用" & u.remarks != "正在尝试登录" & u.remarks != "未知错误").FirstOrDefault();

userInfo.remarks = "正在尝试登录";

db.Entry<appleAcount>(userInfo).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return userInfo;

}

//返回用于测试的用户信息

public appleAcount ParallelTest()

{

userInfo = dbSet.Where((u) => u.cookies != "正在写入" & u.cookies != "已写入").FirstOrDefault();

userInfo.cookies = "正在写入";

db.Entry<appleAcount>(userInfo).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return userInfo;

}

//删除用户信息

public bool DeleteUserInfo(appleAcount userInfo)

{

db.Entry<appleAcount>(userInfo).State = System.Data.Entity.EntityState.Deleted;

int de = db.SaveChanges();

return true;

}

//编辑用户信息

public int EditUserInfo(appleAcount editUserInfo)

{

appleAcount user = dbSet.Where(a => a.ID == editUserInfo.ID).FirstOrDefault();

CopyObj(ref user, editUserInfo);

db.Entry<appleAcount>(user).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return user.ID;

}

//深复制

public void CopyObj(ref appleAcount original, appleAcount editUserInfo)

{

Type T = editUserInfo.GetType();

PropertyInfo[] PI = T.GetProperties();

for (int i = 0; i < PI.Length; i++)

{

PropertyInfo P = PI[i];

P.SetValue(original, P.GetValue(editUserInfo));

}

}

//添加用户信息

public bool AddUserInfo(appleAcount userInfo)

{

dbSet.Add(userInfo);

int ad = db.SaveChanges();

return true;

}

#region 没用的代码

/// <summary>

/// 对象拷贝

/// </summary>

/// <param name="obj">被复制对象</param>

/// <returns>新对象</returns>

private object CopyOjbect(object obj)

{

if (obj == null)

{

return null;

}

Object targetDeepCopyObj;

Type targetType = obj.GetType();

//值类型

if (targetType.IsValueType == true)

{

targetDeepCopyObj = obj;

}

//引用类型

else

{

targetDeepCopyObj = Activator.CreateInstance(targetType); //创建引用对象

MemberInfo[] memberCollection = obj.GetType().GetMembers();

foreach (System.Reflection.MemberInfo member in memberCollection)

{

//拷贝字段

if (member.MemberType == MemberTypes.Field)

{

FieldInfo field = (FieldInfo)member;

Object fieldValue = field.GetValue(obj);

if (fieldValue is ICloneable)

{

field.SetValue(targetDeepCopyObj, (fieldValue as ICloneable).Clone());

}

else

{

field.SetValue(targetDeepCopyObj, CopyOjbect(fieldValue));

}

}

//拷贝属性

else if (member.MemberType == System.Reflection.MemberTypes.Property)

{

PropertyInfo myProperty = (PropertyInfo)member;

MethodInfo info = myProperty.GetSetMethod(false);

if (info != null)

{

try

{

object propertyValue = myProperty.GetValue(obj, null);

if (propertyValue is ICloneable)

{

myProperty.SetValue(targetDeepCopyObj, (propertyValue as ICloneable).Clone(), null);

}

else

{

myProperty.SetValue(targetDeepCopyObj, CopyOjbect(propertyValue), null);

}

}

catch (System.Exception ex)

{

}

}

}

}

}

return targetDeepCopyObj;

}

public object CopyObj(object obj)

{

Type T = obj.GetType();

object o = Activator.CreateInstance(T);

PropertyInfo[] PI = T.GetProperties();

for (int i = 0; i < PI.Length; i++)

{

PropertyInfo P = PI[i];

P.SetValue(o, P.GetValue(obj));

}

return o;

}

#endregion

}

}

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

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<connectionStrings>

<add name="XIAOMISQLEntities" connectionString="metadata=res://\*/SqlXiaoMi.csdl|res://\*/SqlXiaoMi.ssdl|res://\*/SqlXiaoMi.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=DESKTOP-5165QCP\SQL\_XIAOMI;initial catalog=XIAOMISQL;user id=sa;password=APOLLO\_FXJ19881105;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

<add name="CENTOS7SQLEntities" connectionString="metadata=res://\*/CentOSSQL.csdl|res://\*/CentOSSQL.ssdl|res://\*/CentOSSQL.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=192.168.31.185;initial catalog=CENTOS7SQL;user id=sa;password=APOLLO\_FXJ19881105;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />

</startup>

<system.serviceModel>

<bindings>

<wsDualHttpBinding>

<binding name="wsDualHttpBinding\_IServiceContract" maxReceivedMessageSize="2147483647" useDefaultWebProxy="false">

<security mode="None"/>

</binding>

</wsDualHttpBinding>

</bindings>

<services>

<service name="ServiceHost.ServiceMethods" behaviorConfiguration="MyServiceTypeBehaviors">

<host>

<baseAddresses>

<add baseAddress="http://192.168.31.29:8001/getSql" />

</baseAddresses>

</host>

<endpoint address="" binding="wsDualHttpBinding" bindingConfiguration="wsDualHttpBinding\_IServiceContract" contract="WCFService.IServiceContract"></endpoint>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MyServiceTypeBehaviors">

<!-- 将下列元素添加到服务行为配置中。 -->

<serviceMetadata httpGetEnabled="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="mssqllocaldb" />

</parameters>

</defaultConnectionFactory>

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

</configuration>

using System;

using System.Windows.Forms;

using SqlModel;

namespace ServiceHost

{

public partial class Form1 : Form

{

private static ServiceMethods serviceMethods { set; get; }

public Form1()

{

InitializeComponent();

}

private void btnOpenHost\_Click(object sender, EventArgs e)

{

serviceMethods = new ServiceMethods();

serviceMethods.start(btnOpenHost);//开启服务

//&btnOpenHost 0x04c0a7d0 System.Windows.Forms.Button&\*

//&f 0x0963e4cc System.Windows.Forms.Button&\*

//serviceMethods对象中start方法用于接收一个Button对象，但是此处btnOpenHost的地址和start方法接收的对象地址不一样。

}

private void btnTestWriteIn\_Click(object sender, EventArgs e)

{

testee tt = new testee();

tt.ButtonText(btnTestWriteIn);

}

private void txtOrder\_TextChanged(object sender, EventArgs e)

{

if (txtOrder.Text != null & serviceMethods != null)

{

serviceMethods.refresh(txtOrder.Text);//文字更改后，自动更新命令。

}

}

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

{} }

}

using SqlModel;

using System;

using System.Collections.Generic;

using System.ServiceModel;

namespace WCFService

{

//[ServiceContract(SessionMode = SessionMode.Required)]

[ServiceContract]

public interface IServiceContract

{

[OperationContract]

DateTime GetDateTime();

[OperationContract]

List<appleAcount> GetAllUserInfos();

[OperationContract]

appleAcount GetAUserInfo(int id);

[OperationContract]

appleAcount Login();

[OperationContract]

appleAcount ParallelTest();

[OperationContract]

bool DeleteUserInfo(appleAcount userInfo);

[OperationContract]

int EditUserInfo(appleAcount userInfo);

[OperationContract]

bool AddUserInfo(appleAcount userInfo);

[OperationContract]

string GetOrder();

}

}

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Windows.Forms;

namespace SqlModel

{

public class testee

{

public void ButtonText(Button button)

{

button.Invoke(new Action(() => { button.Text = "tt"; }));

}

public List<SqlModel.appleAcount> getUserInfo1()

{

DbContext db = new XIAOMISQLEntities();

var tab = db.Set<appleAcount>().Where(a => a.ID == 1000).ToList();

appleAcount ap = db.Set<appleAcount>().Where(a => a.ID == 20).FirstOrDefault();

ap.district = "00";

db.Entry<appleAcount>(ap).State = System.Data.Entity.EntityState.Modified;

int t= db.SaveChanges();

db.Set<appleAcount>().Attach(ap);

db.Entry(ap);

db.Entry<appleAcount>(ap).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return tab;

}

}

}