**Разработка системы клиент сервер с использованием технологии Remoting**

**объект WellKnow with CallBack ( с обратным вызовом основанном на обработке события)**

**Код определения R- объекта**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ComponentModel;

using System.Windows.Forms;

using System.Drawing;

using System.Threading;

using System.Runtime.Remoting.Messaging;

using System.Threading.Tasks;

namespace RemOb\_CallbackOb

{

//Обявление типа делегат dlgUserInfo

public delegate void dlgUserInfo(string UserId);

// класс для реализации прямого вызова

public class RemOb1 : System.MarshalByRefObject

{

//Обявление екземпляра NewUser делегата типа dlgUserInfo

public static dlgUserInfo NewUser;

// delegate in object

public static string UserId = "";

// объявление и реализация прямого метода с двумя прямыми параметрами без делегата

public int addf(int x, int y, bool pr)

{

if (pr == true) {return x + y;} else{return 2\*(x + y);}

}

public void RegClient(string ClientName, bool res, dlgUserInfo htc)

{

// Параметры метода string ClientName - id клиента

// bool res признак желания получить новый id клиента от сервера

// dlgUserInfo htc - имя метода генерации события в классе события RegClient( ожидается OnRegClient)

if (res)

{

UserId = "Server ID\_" + ClientName;

}

else

{

UserId = ClientName;

}

// генерируем событие

htc(UserId);

}

}

// класс для реализации обратного візова

public class RemOb1CallbackSink : System.MarshalByRefObject

{

//обьявление класса события RegClient с методом генерации события OnRegClient(string userid) связанного с типом делегата dlgUserInfo

public event dlgUserInfo RegClient;

// [OneWay]

// метод генерации события - вызова инициированного делегата

public void OnRegClient(string userid)

{

// MessageBox.Show("Я генератор события, мой параметр :"+ userid);

if (RegClient != null)

RegClient(userid);

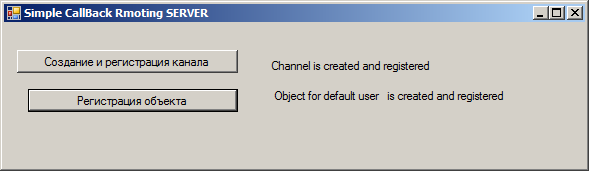
}

}

}

**Разработка R-сервера**

**Форма сервера (после выполнения)**



**Код R-сервера**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using RemOb\_CallbackOb;

using System.Text;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Runtime.Remoting.Channels.Http;

//using System.Runtime.Remoting.Channels.Ipc;

using System.Runtime.Serialization.Formatters.Binary;

//using System.Threading;

//using System.Threading.Tasks;

using System.Windows.Forms;

namespace SmipleCallBackServer

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

SoapServerFormatterSinkProvider serverFormatter = new SoapServerFormatterSinkProvider();

serverFormatter.TypeFilterLevel = System.Runtime.Serialization.Formatters.TypeFilterLevel.Full;

// we have to change the name since we can't have two channels with the same name.

Hashtable ht = new Hashtable();

ht["name"] = "ServerChannel";

ht["port"] = 8080;

// now create and register our custom HttpChannel

HttpChannel channel = new HttpChannel(ht, null, serverFormatter);

ChannelServices.RegisterChannel(channel, false);

label4.Text = "Channel is created and registered";

}

private void button3\_Click(object sender, EventArgs e)

{

RemotingConfiguration.RegisterWellKnownServiceType(typeof(RemOb1),"tolking",WellKnownObjectMode.Singleton);

label5.Text = "Object for default user "+ RemOb1.UserId+ " is created and registered";

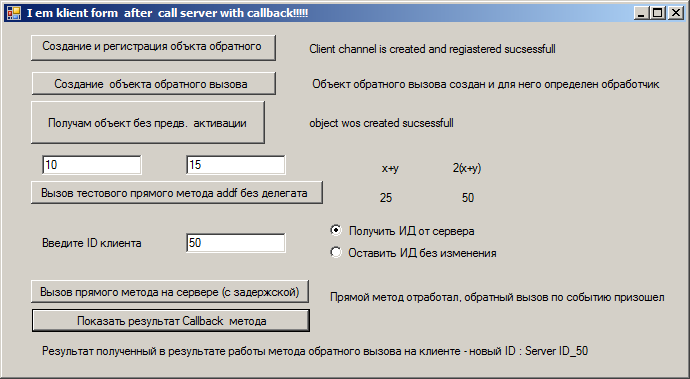
}

}

}

**Разработка R-Клиента**

**Форма R-Клиента (после выполнения)**



**Код R-клиента**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using RemOb\_CallbackOb;

using System.Text;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Runtime.Remoting.Channels.Http;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimpleCallBackClient

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

// Объект прямого вызова, живет на сервере

private RemOb1 \_RemObj1 = null;

// Объект обратного вызова , живет на клиенте

private RemOb1CallbackSink \_RemOb1CallbackSink = null;

private string userid;

private string Newuserid;

private void button1\_Click(object sender, EventArgs e)

{

// Создаем и регистирируем канал для обратного вызова

HttpChannel ClientChannel = new HttpChannel(9090);

ChannelServices.RegisterChannel(ClientChannel, false);

this.label2.Text = "Client channel is created and regiastered sucsessfull";

}

private void button2\_Click(object sender, EventArgs e)

{

Object obj = Activator.GetObject(typeof(RemOb1), "http://127.0.0.1:8080/tolking");

\_RemObj1 =(RemOb1)obj;

if (\_RemObj1 != null)

{ this.label3.Text = "object wos created sucsessfull"; }

else

{ this.label3.Text = "object wos NOT created sucsessfull"; }

}

private void button3\_Click(object sender, EventArgs e)

{

// MessageBox.Show("call server with delegate");

bool par = false;

userid = this.textBox1.Text;

if (radioButton1.Checked) { par = true; }

// вызов прямого метода с указанием метода генерации события

\_RemObj1.RegClient(userid, par, \_RemOb1CallbackSink.OnRegClient);

Form1.ActiveForm.Text = " I em klient form after call server with callback!!!!!";

this.label10.Text = "Прямой метод отработал, обратный вызов по событию призошел";

}

private void CallbackSink\_OnRegClient(string SerUserid)

{ // реализация на клиенте метода обратного вызова

MessageBox.Show("call back method is activated by generate event");

Newuserid = SerUserid;

// this.textBox1.Invoke(new dlgUserInfo(CallbackSink\_OnRegClient),new object[] {userid} );

}

private void button4\_Click(object sender, EventArgs e)

{

// Тест=пример - вызов метода( прямой вызов ) с параметрами без передачи делегата

int x, y, r;

x = int.Parse(textBox2.Text);

y = int.Parse(textBox3.Text);

r = \_RemObj1.addf(x, y, true);

this.label4.Text = r.ToString();

r = \_RemObj1.addf(x, y, false);

this.label5.Text = r.ToString();

// Form1.ActiveForm.Text = " I em klient form after simple call server!!!!!";

}

private void button5\_Click(object sender, EventArgs e)

{

// Создать объект класса обратного вызова

\_RemOb1CallbackSink = new RemOb1CallbackSink();

// задаем(добавляем)функцию - обработчик обратного вызова

\_RemOb1CallbackSink.RegClient += new dlgUserInfo(CallbackSink\_OnRegClient);

this.label7.Text = " Объект обратного вызова создан и для него определен обработчик";

}

private void button6\_Click(object sender, EventArgs e)

{

this.label9.Text = "Результат полученный в результате работы метода обратного вызова на клиенте - новый ID : "+ Newuserid;

}

}

}