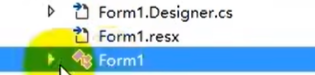
**1、Winform窗体**

**桌面应用程序，是一种智能客户端，我们可以使用winform应用程序可以获得信息或者传输信息。**



**.Designer.cs就是form自动生成的窗体文件。**

**.resx是资源文件**

**Form就是需要编写的后台。**

2、属性

Name：在后台要得到前台的控件对象，需要使用name属性。

Anchor：表示随着随着窗体的变动：改变控件大小。

Backcolor:背景颜色。

BackgroundImage：背景图片。

ContextMenustrip：右键菜单。

Cousr:职责移到此处显示的图标。

Enabled：指示是否启动该控件。

Visible：确定该控件是否可见还是隐藏。

Sizemode：设置背景布局。

Enabled：true或者false,表示控件可用不可用

Tag：标记，与对象关联的用户数据

3、事件

发生一件事情。

注册事件：双击控件注册的都是控件默认被选中的那个事件。

触发事件：

4、

在main函数当中创建的对象称为这个窗体应用程序的主窗体。也就意味着你将主窗体关闭之后，整个程序就关闭啦。

只要控件有小箭头：就可以点进去看一下。

Textbox控件：文本框输入控件

默认的事件：textchanged(文本框的值一发生改变就引起的事件)。

其中常用的属性：wordwrap：文本框自动换行；passwordchar：表示文本框显示单一个char符号；Scrollbars：是否显示滚动条。

其中的方法：textbox名.focus（）；把texbox作为焦点

Labeli：显示文本控件

/// <summary>

/// 当文本款输入的时候，执行此方法

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox1\_TextChanged(object sender, EventArgs e)

{

//将输入文本框的值，传送给标签

label1.Text = textBox1.Text;

}

Timer：组件，相当于定时器，在指定的时间间隔内做指定的一件事情。

多选和单选框控件：checkBox：多选框；rediobutton：单选框。Checked这个属性：指示这个控件是否处于选中状态。

默认情况下，在一个窗体中，所有的单选框，只能允许选中一个，可以使用groupbox进行分组。

对于单选框，分组的话需要容器groupbox。

MDI窗体设计：

1. 首先确定一个父窗体（在窗体属性中有isMdicontainer属性来确定本窗体是MDI窗体的容器）
2. 构建菜单栏，需要菜单MenuStrip：构建菜单栏，对每个项目，进行设计。
3. 创建子窗体，并且设置他们的父窗体。
4. 排列MDI主窗体内的子窗体：通过layoutMDI方法。

LayoutMDI（MDIlayout.TilerHorizontal）横向排列。

TileVertical纵向排列。

/// <summary>

/// 点击菜单栏中的显示子窗体，创建子窗体，并显示子窗体

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

//创建子窗体对象，并显示，而且限制其范围

Form2 fom2 = new Form2();

fom2.MdiParent = this;

fom2.Show();

Form3 fom3 = new Form3();

fom3.MdiParent = this;

fom3.Show();

Form4 fom4 = new Form4();

fom4.MdiParent = this;

fom4.Show();

}

/// <summary>

/// 点击横向排列，就是将MDI主窗体中的所有子窗体排列

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

显示图片：

控件picturebox：旋转框

之后对其属性image：显示图片

Prcturebox名.image=image.formfile(文件路径)

需要diretory这个来读取文件夹

String[] name=Diretory.get file(@“文件夹目录”);

string[] path = Directory.GetFiles(@"F:\wallhaven壁纸下载地");

int nums=0;

/// <summary>

/// 当打开窗体，就启动程序

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//启动窗体，就读取文件夹中的文件，之后随便展示一张图片

Random number = new Random();

number.Next(0, path.Length);

pictureBox1.Image = Image.FromFile(path[number.Next(0,path.Length)]);

}

/// <summary>

/// 点击按钮则会显示，下一张图片

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

nums++;

if (nums==path.Length)

{

nums = 0;

}

pictureBox1.Image = Image.FromFile(path[nums]);

}

/// <summary>

/// 点击显示上一张

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

nums--;

if (nums<0)

{

nums = path.Length - 1;

}

pictureBox1.Image = Image.FromFile(path[nums]);

}

浏览器控件：Webbrowser：网络浏览器控件

主要是其属性UrI，去调用网站，当然我们要创建uri的对象，其构造函数，可以将string类型的网址，转成uri，之后

Webrowser.uri接受。

//首先收集网站

string name = textBox1.Text;

//因为webbrowser.uri只接受隐式string

//所以我们创建uri对象

Uri url = new Uri("http://"+name);

webBrowser1.Url = url;

combobox：下拉框控件

dropdownstyle:控制下拉框的样式，其默认的事件，就是当选择一个你在下拉框中设定好的东西，就可以发生。

/// <summary>

/// 当年份确定之后，给定可以选择的月份

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

//先要判断combbox1.text是否为空

//当重复选择年份的时候需要将之前的月份清空

comboBox2.Items.Clear();

if (comboBox1.Text!=null)

{

for (int i =1; i <=12; i++)

{

comboBox2.Items.Add(i);

}

}

}

给控件添加右键菜单：contextmenustrip控件：

在其需要绑定的控件中的属性：contextmenustrip属性选择，我们所调用contextmenutrip控件。

Listbox：选择框。

//可以通过list集合，来将读取出来的文件名保存。

List<string> listname = new List<string>();

/// <summary>

/// 当启动窗体，就让listbox，扫描一个文件夹里面的所有图片，得到文件名。

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//将所有jpg的文件的全路径全部读取出来。

string[] pathname = Directory.GetFiles(@"F:\wallhaven壁纸下载地","\*.png");

//将全路径里面的文件名读取出来。

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

{

string Name = Path.GetFileName(pathname[i]);

listBox1.Items.Add(Name);

//将图片的全路径添加到泛型集合list中

listname.Add(pathname[i]);

}

}

/// <summary>

/// 当双击选中的文件名时，出现对应的图片

/// 将得到的文件名，做下标

/// </summary>

/// <param name="sender"></param>

private void listBox1\_DoubleClick(object sender, EventArgs e)

{

//这里list.selectedindex，就表示了我们在listbox双击

//选中文件，在items集合中元素的下标。

pictureBox1.Image = Image.FromFile(listname[listBox1.SelectedIndex]);

}

打开对话框：openfiledialog打开文件类

首先创建打开文件对话框的对象

OpenfileDialog：打开文件对话框。

SaveFileDialog类:提示用户选择文件保存的位置。

Fontadilog类：字体对话框。

//首先要创建打开文件的类的对象

OpenFileDialog ofd = new OpenFileDialog();

//设置打开文件的，窗体名

ofd.Title = "文憔悴的所有文件";

//设置对话框的可以多选文件。

ofd.Multiselect = true;

//设置对话框的初始目录

ofd.InitialDirectory = @"F:\edge下载地";

//设置对话框可以打开文件的类型

ofd.Filter = "文本文件|\*.txt|媒体文件|\*.jpg|所有文件|\*.\*";

//展示对话框

ofd.ShowDialog();

//将我们打开的文本文件在textbox上打开。

string path = ofd.FileName;

using (FileStream fls=new FileStream(path,FileMode.OpenOrCreate,FileAccess.Read))

{

//读取到的是文件的字节

byte[] buffer = new byte[5 \* 1024 \* 1024];

int number= fls.Read(buffer,0,buffer.Length);

textBox1.Text = Encoding.Default.GetString(buffer,0,number);

}

保存文件对话框： savefiledialog保存文件类

SaveFileDialog sfl = new SaveFileDialog();

sfl.Title = "保存到";

sfl.InitialDirectory = @"D:\桌面";

sfl.Filter = "文本文件|\*.txt|所有文件|\*.\*";

sfl.ShowDialog();

string path = sfl.FileName;

if (path == " ")

{

return;

}

using (FileStream file = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Write))

{

byte[] buffer = Encoding.GetEncoding("GBK").GetBytes(textBox1.Text);

file.Read(buffer, 0, buffer.Length);

}

MessageBox.Show("保存成功");

字体和颜色对话框：

Fontdialog字体类：

FontDialog fo = new FontDialog();

fo.ShowDialog();

textBox1.Font = fo.Font;

Colordialog颜色类：

ColorDialog co = new ColorDialog();

co.ShowDialog();

textBox1.ForeColor = co.Color;

Process：是一个非静态类

给我们提供了很多静态：其中的静态方法，.getprocesses（）;

返回值是一个进程数组。

//获得当前程序中所有正在运行的程序

//Process[] number= Process.GetProcesses();

// foreach (var item in number)

// {

// Console.WriteLine(item);

// }

// Console.ReadKey();

//通过进程打开一些应用

//Process.Start("calc");

//Process.Start("iexplore","http://www.baidu.com");

//Process.Start("iexplore","http://www.douyin.com");

//Console.ReadKey();

//通过一个进程打开指定的文件

//第一：创建进程对象

//第二：创建一个人processstartinfo的对象

//第三：之后将processStartinfo的对象，传给process对象，调用startinfo的属性；

ProcessStartInfo psi = new ProcessStartInfo(@"F:\yy\YY.exe");

Process pro = new Process();

pro.StartInfo = psi;

pro.Start();

Console.ReadKey();

单线程给我们带来的问题？

让计算机“同时”做多件事情，会假死。

多线程：

让计算机同时“同时”做多件事情，节约时间。

多线程可以让一个程序“同时”处理多个事情，

后台运行程序，提高程序的运行效率，也不会是主界面出现无响应的情况。

获得当前线程和当前进程。

线程：

前台线程：只有所有前台线程都关闭才能完成程序关闭。

后台线程：只要所有的前台线程结束，后台线程自动结束。

产生线程的4步骤

1. 编写产生线程所要执行的方法。
2. 引用system.Threading命名空间。
3. 实例化thread类，并传输入一个指向线程所要运行方法的委托。（这个时候线程已经产生但是没有运行）
4. 调用thread实列的start方法，标记该线程可以被cpu执行了，但具体执行时间由cpu决定。

线程常用的方法：

静态方法：thread sleep（）；可是使当前的线程停止一段时间。

非静态方法：线程对象.start(),启动线程

在.net平台下，无法跨线程访问。

如果线程执行的方法需要参数，那么要求这个参数必须是object类型。

**ThreadStart** 委托：这个委托表示一个不接受参数的线程方法。你可以将这种委托用于创建一个不需要参数的线程方法，例如：

Thread thread = new Thread(new ThreadStart(MyThreadMethod));

**ParameterizedThreadStart** 委托：这个委托表示一个接受一个**object**类型参数的线程方法。你可以将这种委托用于创建需要传递参数的线程方法，例如

Thread thread = new Thread(new ParameterizedThreadStart(MyThreadMethodWithParameter));

其中，**MyThreadMethodWithParameter** 是一个接受一个 **object** 参数的线程方法。

如果你的线程方法需要多个参数，可以将它们打包到一个对象中，然后传递给线程方法，或者使用其他适当的方法来传递参数。注意，在使用 **ParameterizedThreadStart** 时，需要小心确保在线程方法内部对参数进行适当的类型转换和处理，以避免类型不匹配或其他问题。

并且将其放在，可启动这个线程的方法里面。线程名.start(参数);

//创建一个线程对象

Thread th;

/// <summary>

/// 调用进程

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//让线程得到委托的方法

th = new Thread(Test);

//因为自己所创建的前程是前台线程，当主线程结束后，所创建的前台线程还在执行

//所以需要将其转为后台线程

th.IsBackground = true;

//告诉cpu可以执行方法，具体什么时候执行，cpu决定。

th.Start();

}

public void Test()

{

for (int i = 0; i < 10000; i++)

{

textBox1.Text = i.ToString();

}

}

/// <summary>

/// 因为需要调用的控件在主线程中，自己创建的前程无法调用主线程中的控件

/// 所以需要取消对跨线程的访问

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//当窗体启动停止对跨线程的检查取消

Control.CheckForIllegalCrossThreadCalls = false;

}

/// <summary>

/// 当我们关闭主窗体的时候，也就是结束主线程的时候，

/// 因为没有对跨线程的访问，所以无法将所创建线程里面的资源释放

/// 所以需要判断是否其线程是否执行完啦

/// 没有就终止

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

//当我点击窗体关闭的时候，判断线程是否为null

if (th !=null)

{

th.Abort();

//abort终止线程之后，无法重新启动

}

}

Socket：进程机制，套接字，用于描述IP地址和端口，是一个通信链的句柄。（两台计算机，建立连接，收发数据就需要socket来通信）其实就是两个程序通信用的。

通信：

客户端想和服务器中得各种各样得应用程序通信。

1、首先通过ip地址连接到服务器；

2、之后通过端口号再通信到对应的应用程序，一般端口号50000以后；

3、客户端，服务器都必须有socket。

4、服务器有一个负责监听的socket，监听应用程序的ip地址和端口号，看是否有人连接。

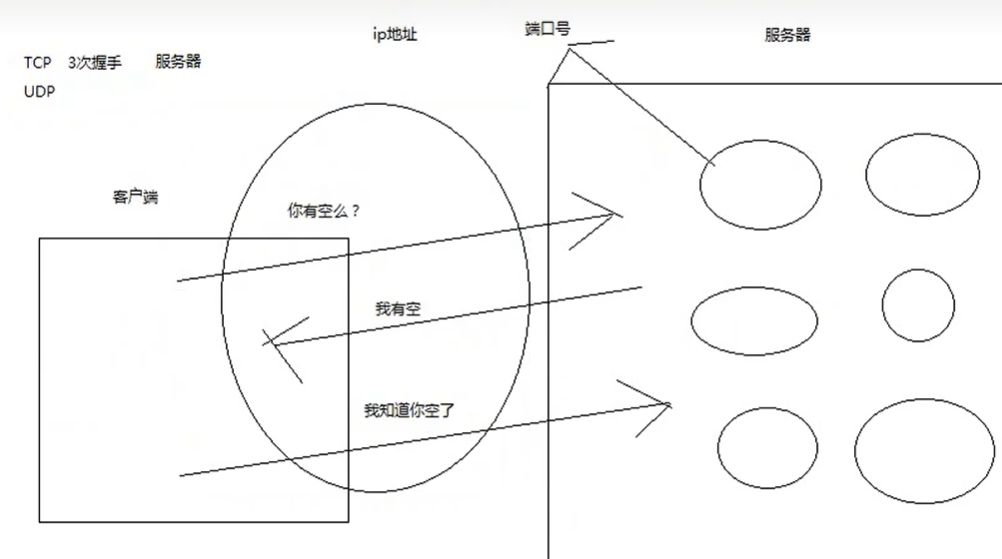
5、负责监听的socket发现有客户端连接之后，创建一个和客户端通信的socket。

6、客户端只有一个socket。

需要通信协议：

Tcp：安全稳定，效率低，需要经历3次握手，要求必须有服务器。流式Socket。

需要发送，回复，在回复接受到，这三次握手。三次握手有一次失败，就发送失败。



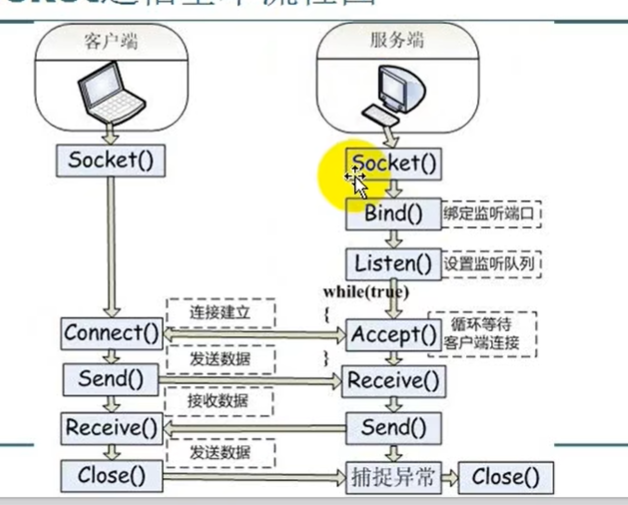
Udp：快速，效率高，不稳定，容易发生数据丢失。不管服务器，接不接受，就直接发送数据。数据报式Socket。传输视频的时候，用Udp。

·流式Socket (stream):

是一种面向连接的Socket，针对子面向连接的TCP服务应用，安全，但是效率低;

．数据报式Socket (datagram):

是一种无连接的Socket,对应于无连接的UDP服务应用.不安全(丢失,顺序混乱,在接收端要分析重排及要求重发),但效率高。



这是服务器端：

//创建一个监听的socket

Socket socketWatch = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

//创建一个线程thread

Thread thr;

//创建一个dictionary集合，来存储ip地址和socket

Dictionary<string,Socket> dicsocket=new Dictionary<string,Socket>();

/// <summary>

/// 点击开始监听

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//在接受和通信的地方就容易出现异常，就需要try catch

try

{

//创建ip地址和端口号的对象

IPAddress ipname = IPAddress.Any;//会拿到我的ip地址，IP地址改变，这里也改变// IPAddress.Parse(textserver.Text);

//这里创建的开端口号，也需要将ip地址给入端口号

//我这里是给定的端口号。

IPEndPoint ippoint = new IPEndPoint(ipname, Convert.ToInt32(textport.Text));

//让负责监听的socket绑定ip和端口号

socketWatch.Bind(ippoint);

TestShow("监听成功");

//设置监听队列,在某一个时间点，可以连入服务端的最大客户端的数量

socketWatch.Listen(10);

//用一个线程来调用接受客户端的方法

thr = new Thread(Test);

//首先需要将这个前台线程变成后台线程

thr.IsBackground = true;

//之后告诉cpu，可以启动该线程啦

thr.Start();

}

catch

{

}

}

/// <summary>

/// 表现开始监听

/// </summary>

/// <param name="str"></param>

public void TestShow(string str)

{

//append追加

textlog.AppendText(str + "\r\n");

}

Socket socketSend;

/// <summary>

/// 接受客户端的socket，建在一个方法里面，用线程来调用

/// </summary>

public void Test()

{

try

{

//服务器端，可能有多个客户端连接过来。

while (true)

{

//负责监听的socket，来接受客户端的连接

//当不知道本机的局域网地址，可以在控制台输入ipconfig读取出本机局域网的ip地址

socketSend = socketWatch.Accept();

//将每次连接服务器的客户端的ip和端口号，存入dictionary集合

dicsocket.Add(socketSend.RemoteEndPoint.ToString(),socketSend);

//之后将每个客户端的ip和端口号放入下拉框

cboUsers.Items.Add(socketSend.RemoteEndPoint.ToString());

//获取客户端的ip地址和端口号,负责连接的socket对象调用.remoteEndPoint();

TestShow(socketSend.RemoteEndPoint.ToString());

//在创建一个线程，来接受客户端的信息

Thread thr = new Thread(Receive);

thr.IsBackground = true;

thr.Start(socketSend);

}

}

catch

{ }

}

/// <summary>

/// 当启动窗体时，要明白不可以因为不可以跨线程调用控件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//所以不可以检查该线程跨线程调用的窗体

Control.CheckForIllegalCrossThreadCalls = false;

}

/// 来接受客户端传输过来的信息

/// </summary>

/// <param name="o"></param>

public void Receive(object o)

{

//在网络中出现异常

Socket socketSend = o as Socket;

while (true)

{

try

{

//客户端连接成功之后，服务器接受客户端发送的信息

byte[] buffer = new byte[1024 \* 1024 \* 5];

//实际接受的有效字节

int number = socketSend.Receive(buffer);

//判断是否还有有效字节，也就是判断是否还有发送的信息

if (number != 0)

{

string str = Encoding.UTF8.GetString(buffer, 0, number);

TestShow(socketSend.RemoteEndPoint.ToString() + ":" + str);

}

else

{

//要是为0直接跳出循环

break;

}

}

catch

{

}

}

}

/// <summary>

/// 给客户端发送信息

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//首先将要发的字符串转为字节数组

string str = textMsg.Text;

byte[] buffer = Encoding.UTF8.GetBytes(str);

//给文本文件，的buffer最开头加上0，用以区别

List<byte> list = new List<byte>();

list.Add(0);

list.AddRange(buffer);

byte[] newbuffer = list.ToArray();

//负责通信的socket，来发送

//需要选择发送客户端的socket

string ip = cboUsers.SelectedItem.ToString();

dicsocket[ip].Send(newbuffer);

}

/// <summary>

/// 点击x选择需要发送文件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

OpenFileDialog olg = new OpenFileDialog();

olg.Title = "选择需要发送的文件";

olg.InitialDirectory = @"D:\桌面";

olg.Multiselect = true;

olg.Filter = "所有文件|\*.\*";

olg.ShowDialog();

string pathname=olg.FileName;

textpath.Text = pathname;

}

/// <summary>

/// 将我们选择的文件进行发送

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

string pathname = textpath.Text;

using (FileStream readfile=new FileStream(pathname,FileMode.OpenOrCreate,FileAccess.Read))

{

byte[] buffer = new byte[1024 \* 1024 \* 5];

int number = readfile.Read(buffer, 0, buffer.Length) ;

List<byte> list = new List<byte>();

list.Add(1);

list.AddRange(buffer);

byte[] newbuffer = list.ToArray();

//对于所传的文件需要从位开始传，使客户端接收，判断是否是文本文件

dicsocket[cboUsers.SelectedItem.ToString()].Send(newbuffer,0,number+1,SocketFlags.None);

}

}

/// <summary>

/// 给客户端发送一个震动

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

byte[] buffer = new byte [1];

buffer[0] = 2;

dicsocket[cboUsers.SelectedItem.ToString()].Send(buffer);

}

客户端：  
Socket socketsend;

/// <summary>

/// 点击连接就通信到服务端

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

//创建负责通信的socket

socketsend = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

//来接上ip和端口号

IPAddress ip = IPAddress.Parse(textip.Text);

IPEndPoint point = new IPEndPoint(ip, Convert.ToInt32(textpoint.Text));

//让socket连接上ip和端口号

socketsend.Connect(point);

Show(socketsend.RemoteEndPoint + ":");

//创建thread来实现接受信息

Thread thr = new Thread(Receive);

thr.IsBackground = true;

thr.Start();

}

catch

{ }

}

public void Show(string str)

{

textlog.AppendText(str + "\r\n");

}

/// <summary>

/// 发送信息.客户端给服务器

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

string str = textmsg.Text.Trim();

//将其转换成字节

byte[] buffer = System.Text.Encoding.UTF8.GetBytes(str);

socketsend.Send(buffer);

}

/// <summary>

/// 当启动窗体，不检查线程跨线程调用控件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Control.CheckForIllegalCrossThreadCalls = false;

}

/// <summary>

/// 用来接受服务器发送过来的信息

/// </summary>

public void Receive()

{

//因为可能需要一直接受

while (true)

{

try

{

//需要字节数组来接收

byte[] buffer = new byte[1024 \* 1024 \* 5];

int number = socketsend.Receive(buffer);

if (number == 0)

{

break;

}

//需要判断是否为文本

if (buffer[0] == 0)

{

//将字节转成字符串

string str = Encoding.UTF8.GetString(buffer, 1, number);

//将其展示在textmgs.text

Show(socketsend.RemoteEndPoint + ":" + str);

}

else if (buffer[0] == 1)

{

//如果需要对不同的文件，进行选定的话，需要可以在数组后面第二位，进行限定

SaveFileDialog sa = new SaveFileDialog();

sa.Title = "保存到";

sa.InitialDirectory = @"D:\桌面";

sa.Filter = "所有文件|\*。\*";

sa.ShowDialog(this);

string path = sa.FileName;

if (path == null)

{

return;

}

using (FileStream writefile = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Write))

{

writefile.Write(buffer, 1, buffer.Length);

}

MessageBox.Show("保存成功");

}

else if (buffer[0] == 2)

{

ZD();

}

}

catch { }

}

}

/// <summary>

/// 让主窗体在两个坐标之间，来回切换

/// </summary>

public void ZD()

{

for (int i = 0; i < 500; i++)

{

this.Location = new Point(200, 200);

this.Location = new Point(280, 280);

}

}