آموزش سوکت پروگرمینگ در سی شارپ

Socket Programming in C#

پیش نیاز : آشنایی با مفاهیم شبکه ای مانند IP و Port و Socket و Send - Receive و Client و Server

در مدل کلاینت - سرور ، مبادلات زیر بین کلاینت و سرور رخ میدهد:

۱. سرور سوکتی را تعریف میکند.
۲. سرور سوکت را به یک IP که همان IP خودش است و یک پورت Bind میکند یا اختصاص میدهد.
۳. سرور به پورت گوش میدهد.
۴. کلاینت سوکتی را تعریف میکند و IP و پورت سرور را به آن اختصاص میدهد.
۵. کلاینت درخواست اتصال یا کانکت شدن به سرور را میدهد.
۶. سرور درخواست کلاینت را دریافت و آن را می پذیرد.
۷. کلاینت اطلاعاتی را ارسال می کند.
۸. سرور اطلاعات را می گیرد.
۹. سرور اطلاعات را ارسال میکند و کلاینت آن را میگیرد.
۱۰. سرور بسته می شود.
۱۱. کلاینت بسته می شود.

ابتدا برنامه سمت سرور را مینویسیم . در این برنامه می بایست یک پورت را باز کرده و به آن گوش دهیم و دریافتی را نمایش دهیم.

ابتدا باید فضای نام های زیر را با استفاده از using به کامپایلر سی شارپ معرفی کنیم :

System

System.Net

System.Net.Socket

System.Text

این کار را به این صورت انجام می دهیم :

```
using System;

using System.Net;

using System.Net.Sockets;

using System.Text;
```

اکنون متغیری به نام را به صورت سراسری و static تعریف میکنیم (از آنجایی که متد Main یک متد static است کلیه متغیرها و توابع مورد استفاده در آن نیز باید static باشند.)

متغیرها و متدهای Static متغیرها و توابعی هستند که در یک کلاس به طور مشترک بین کلیه اشیاء آن کلاس استفاده می شوند. نه اینکه به ازای هر شیء یک نمونه از آن ایجاد شود. دسترسی به این متغیرها از طریق نام کلاس ممکن خواهد بود. در کلاس از نوع static ما می توانیم به متدها و متغییر های آن کلاس بدون ساخت شی دسترسی داشته باشیم.

یک شیء از کلاس سوکت به صورت سراسری و static ایجاد می کنیم :

```
static Socket sktListener;
```

در متد Main این سوکت را new می کنیم تا به آن حافظه اختصاص داده شود :

```
static void Main()

{

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

AddressFamily.InterNetwork به معنای این است که از شبکه ای استفاده می کنیم که دارای IP نسخه 4 است.

SocketType.Stream برای سوکت هایی است که می خواهند به صورت Stream داده ها را تبادل کنند

ProtocolType.Tcp که نوع پروتکل ما را مشخص میکند.

اکنون می باید آدرس IP و یک Port به سوکت مان اختصاص دهیم :

```
IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, 1800);

sktListener.Bind(ipLocal);
```

از آنجایی که این برنامه در سمت سرور اجرا می شود آدرس IP خاصی به آن نمی دهیم و پورت ۱۸۰۰ را باز می کنیم.
کلاس IPEndPoint برای مشخص نمودن یک نود یا یک کامپیوتر در شبکه به کار می رود.

متد Bind نود مشخص شده را به سوکت اختصاص میدهد.

اکنون زمان گوش دادن به پورت است :

```
sktListener.Listen(100);
```

عدد ۱۰۰ نشانه آن است که حداکثر ۴ connection می توانند در صف قرار گیرند .

اگر در این لحضه در command prompt دستور netstat –an را تایپ کنید می توانید ببینید که پورت ۱۸۰۰ باز شده و در حال گوش دادن است.

حال می باید تقاضای کانکت شدن کلاینت را بپذیریم:

```
sktListener = sktListener.Accept();
```

حال برای گرفتن داده ها ، می بایست یک بافر تعریف نماییم.

نکته : در سوکت پروگرمینگ ، داده ها به صورت آرایه ای از بایت ها منتقل می شوند. برای ارسال رشته های یونیکد و .... بایست آنها را کد گذاری کنیم. برای کد گذاری و کد گشایی از کلاس System.Text و متدهای آن استفاده کنیم. مثلا دستور زیر رشته salam را با فرمت Ascii به آرایه ای از بایت ها تبدیل می کند.

```
byte[] byt = Encoding.ASCII.GetBytes("salam");
```

و متد زیر آن را رمزگشایی می کند :

```
string str = Encoding.ASCII.GetString(byt);
```

ما عمل رمزنگاری را موقع ارسال داده ها و عمل رمز گشایی را موقع دریافت آنها انجام می دهیم.

اکنون می خواهیم داده ها را دریافت کرده و رمز گشایی کنیم :

```
byte[] buffer = new byte[500];

sktListener.Receive(buffer);

string Data = Encoding.ASCII.GetString(buffer);
```

حال میتوانیم داده ها را پردازش کنیم.

سورس برنامه سمت سرور

```
using System;

using System.Net;

using System.Net.Sockets;

using System.Text;


static Socket sktListener;
```

```
static void Main(string[] args)
{
        sktListener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, 1800);
        sktListener.Bind(ipLocal);
        sktListener.Listen(100);
        sktListener = sktListener.Accept();
        byte[] buffer = new byte[500];
        sktListener.Receive(buffer);
        sktListener.close();
}
```

برنامه سمت کلاینت

ایتدا یک سوکت تعریف می کنیم :

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

حال سرور را به آن معرفی می نماییم و به آن کانکت می شویم :

```
sktClient.Connect("127.0.0.1", 1800);
```

اگر دوباره دستور netstat –an را در Command prompt تایپ کنیم می بینیم که ارتباط برقرار شده است. به شماره پورتها در آن دقت کنید.

اکنون داده های ارسالی را آماده می کنیم :

```
string str = "Hello Server....";
byte[] buffer = Encoding.ASCII.GetBytes(str);
```

داده ها را ارسال می کنیم :

```
sktClient.Send(buffer);
```

و سوکت را می بندیم :

```
sktClient.Close();
```

```
using System;

using System.Net;

using System.Net.Sockets;

using System.Text;


static void Main(string[] args)

{

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

        sktClient.Connect("127.0.0.1", 1800);

        string str = "Hello Server....";

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

        sktClient.Send(buffer);

        sktClient.Close();

}
```

برنامه سمت سرور

کدهایی که تا به اینجا دیدیم برای ایجاد سوکت های همگام یا سنکرون بوده است. این سوکت ها در برنامه های ویندوز و کلا سیستم های مالتی تسک کاربردی ندارند. چرا که زمانی که از متد accept استفاده نموده ایم، در این حالت برنامه تا رسیدن یک سوکت به آن قفل شده و قادر به انجام کاری نیست.

در سوکت های آسنکرون از متدهای آسنکرون برای گوش دادن ، ارسال ، دریافت و ... استفاده می کنیم. در این آموزش ، یک برنامه سمت سرور به صورت آسنکرون طراحی می کنیم که قادر به گوش دادن به یک کلاینت است.

نکته : قبل از ادامه ، آشنایی با delegate ها الزامی است. اگر بخواهیم در یک جمله Delegate ها را تعریف کنیم می توانیم بگوییم :

delegate در حقیقت چیزی نیست جز اشاره گر به تابع.

در سوکت های آسنکرون ، از delegate ای به نام AsyncCallback استفاده می کنیم. این Delegate به تابعی اشاره میکند که تنها یک آرگومان ورودی از نوع IAsyncResult دارد. متدهایی که به صورت آسنکرون کار می کنند ، اطلاعات مورد نظر خود را به صورت یک شی از این نوع به تابع مورد نظر خود ارسال می کنند.

نکته: متدهای آسنکرون با پیشوندهای Begin و End شروع میگردند.

برای نوشتن برنامه ، ابتدا یک سوکت تعریف می کنیم که عمل گوش دادن را انجام دهد :

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

سپس عملیات معمول را بر روی سوکت انجام میدهیم :

```
IPEndPoint server = new IPEndPoint(IPAddress.Any, 1800);
Mainlistener.Bind(server);
```

همانگونه که می بینید ، در این برنامه سوکت مورد نظر ما به پورت ۱۸۰۰ گوش می دهد.

اکنون زمان آن است که یک delegate ایجاد کرده و آن را به تابع پردازشگر که در این مثال AcceptCallBack نام دارد ، منتسب کنیم.

```
AsyncCallback callBackMethod = new AsyncCallback(AcceptCallback);
```

اکنون باید سوکت تعریف شده به صورت غیر همگام (آسنکرون) شروع به گوش دادن به پورت کند :

```
Mainlistener.Listen(4);
Mainlistener.BeginAccept(AcceptCallback,Mainlistener);
```

در این مثال ، مشخص کرده ایم که سوکت شروع به عمل گوش دادن و انتظار کند و سپس به محض کانکت شدن یک کلاینت به کامپیوتر ما ، تابع AcceptCallBack اجرا گردد و به اموری که تعیین می کنیم رسیدگی کند.

نکته: پارامتر دوم تابع BeginAccept ، شی ای است برای ارسال داده های وضعیت سوکت، به تابعی که به سوکت رسیدگی می کند. در این جا این شیء خود سوکت است. اگر سوکت را به صورت سراسری تعریف می کردیم، نیاز به ارسال این شیء نبود و به جای آن null قرار می دادیم. شی مربوطه در قالب یک شی از کلاس IAsyncResult ارسال خواهد شد.

تابع AcceptCallback باید این گونه تعریف شود.

```
private void AcceptCallback(IAsyncResult ar)
{
…
}
```

در این تابع ، آرگومانی از نوع IAsyncResult وجود دارد. این آرگومان اطلاعات وضعیت فراخوان تابع آسنکرون که در اینجا یک سوکت است را نگهداری می کند. ابتدا این اطلاعات را استخراج میکنیم :

```
Socket temp = ((Socket)ar.AsyncState);
```

سپس به گوش دادن برای پذیرفتن کلاینت خاتمه می دهیم چرا که اکنون دیگر کلاینت مورد نظر به سرور کانکت شده و آماده برای ارسال اطلاعات است :

```
Socket worker = temp.EndAccept(ar);
```

نکته: دو دستور قبل را می توانستیم در قالب یک دستور و به این شکل بنویسیم :

```
Socket temp = ((Socket)ar.AsyncState).EndAccept(ar);
```

بسیار خوب، اکنون که ارتباط کلاینت با برنامه ما برقرار گردیده است ، کافی است تا به صورت آسنکرون به دریافت اطلاعات مشغول شویم. باز هم مانند قسمت قبل، از متدهای آسنکرون استفاده کرده و تابعی تعرف می کنیم که به محض دریافت اطلاعات فراخوانی گردیده و عملیات مورد نظر ما را انجام دهد.

نکته ای که بسیار حائز اهمیت است این است که باید از یک بافر برای ذخیره اطلاعات دریافتی استفاده کنیم. این بافر که در حقیقت آرایه ای از بایت هاست را به صورت یک آرایه سراسری تعریف می کنیم :

```
byte[] buffer = new byte[1024];
```

نکته: با توجه به این که در این مثال صرفا یا یک کلاینت کار می کنیم، سراسری بودن بافر مشکلی ایجاد نمی کند، اما چنانچه قصد داشتیم با چند کلاینت کار کنیم برای هریک باید بافر مخصوص به خودش را تعریف می کردیم که اصولا پیاده سازی آن برنامه به گونه ای دیگر خواهد بود.

برای دریافت اطلاعات به صورت آسنکرون ، از متد BeginReceive استفاده می کنیم. البته باید بافر، اندیس اولیه ای که می خواهیم بافر از آنجا پر شود و همچنین اندیس حد نهایی بافر را مشخص کنیم.

```
AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);

worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,new
ReceiveMethod, worker);
```

پس تا اینجا ، متد AcceptCallback به این صورت است :

```
private void AcceptCallback(IAsyncResult ar)

{

 Socket temp = ((Socket)ar.AsyncState);

 Socket worker = temp.EndAccept(ar);

  AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);

  worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,new

                                        ReceiveMethod, worker);

}
```

اکنون متد ReceiveCallBack را تعریف می کنیم:

```
private void ReceiveCallBack(IAsyncResult ar)

{

…

}
```

درون این متد ، ابتدا اطلاعات وضعیت را به دست می آوریم:

```
Socket worker = ((Socket)ar.AsyncState);
```

سپس به گوش دادن به صورت موقت خاتمه می دهیم تا بتوانیم داده های فعلی را پردازش کنیم. این کار را با متد EndReceive انجام می دهیم . مقدار بازگشتی این متد، تعداد بایت های دریافت شده می باشد :

```
int bytesReceived = worker.EndReceive(ar);
```

حال باید اطلاعات دریافت شده که به صورت آرایه ای از بایت ها درون بافر هستند را پردازش کرده و جهت نمایش به رشته (string) تبدیل کنیم :

```
string str = System.Text.UTF8Encoding.UTF8.GetString(buffer);
```

حال می توانیم اطلاعات را نمایش دهیم. نکته مهمی که اینجا وجود دارد این است که در برنامه نویسی به شیوه آسنکرون دیگر نمی توانیم به صورت عادی این رشته را به یک TextBox و یا یک کنترل بفرستیم. دلیل این امر آن است که تنها thread ای که کنترل مورد نظر را در ابتدای برنامه ایجاد کرده است می تواند با کنترل کار کرده و اطلاعات لازم را به آن بفرستد. پس کاری که انجام می دهیم استفاده از یک متد کمکی است که ابتدا thread ایجاد کننده کنترل را پیدا کرده و

سپس کار مورد نظر را با استفاده از آن انجام می دهد. در این مثال نام این تابع را ShowInfo قرار داده ایم و تنها یک پارامتر از نوع رشته ای به آن ارسال می کنیم . البته فعلا دانستن جزئیات این متد چندان لازم نیست :

```
delegate void ShowInfoCallback(string text);

private void ShowInfo(string text)

{

if (this.txtMain.InvokeRequired)

 {

 ShowInfoCallback d = new ShowInfoCallback(ShowInfo);

  this.Invoke(d, new object[] { text });

 }

  else

  {

   txtMain.Text+=text+"\n";


  }


 }
```

نکته: در برنامه اصلی فرض کرده ایم کنترلی که قرار است اطلاعات را نمایش دهد یک RichTextBox به نام txtMain است.

بسیار خب ، تاکنون توانستیم اطلاعات را دریافت کرده و به صورت یک رشته در آوریم. سپس آن را پردازش کنیم. اگر به یاد داشته باشد برای پردازش کردن اطلاعات، عمل دریافت اطلاعات را متوقف کرده ایم. اکنون این عمل را برای دریافت ادامه اطلاعات از سر می گیریم :

```
AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);

worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,

            ReceiveMethod, worker);
```

پس متد ReceiveCallBack به این صورت است:

```
private void ReceiveCallBack(IAsyncResult ar)

{

 Socket worker = ((Socket)ar.AsyncState);
```

```
 int bytesReceived = worker.EndReceive(ar);
 string str = System.Text.UTF8Encoding.UTF8.GetString(buffer);
 ShowInfo(str);
 AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);
 worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,
                                  ReceiveMethod, worker);
}
```

کدهای کامل سمت سرور

```
byte[] buffer = new byte[1024];
 public Form1()
{
            InitializeComponent();

 Socket Mainlistener = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
  IPEndPoint server = new IPEndPoint(IPAddress.Any, 1800);
 Mainlistener.Bind(server);
  AsyncCallback callBackMethod = new AsyncCallback(AcceptCallback);
  Mainlistener.Listen(4);
 Mainlistener.BeginAccept(AcceptCallback,Mainlistener);
}
 private void AcceptCallback(IAsyncResult ar)
{
            Socket temp = ((Socket)ar.AsyncState);
            Socket worker = temp.EndAccept(ar);
            AsyncCallback ReceiveMethod = new
AsyncCallback(ReceiveCallBack);
```

```
worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,
ReceiveMethod, worker);
}
 private void ReceiveCallBack(IAsyncResult ar)
{
            Socket worker = ((Socket)ar.AsyncState);
            int bytesReceived = worker.EndReceive(ar);

            string str = System.Text.UTF8Encoding.UTF8.GetString(buffer);
            ShowInfo(str);
            AsyncCallback ReceiveMethod = new
AsyncCallback(ReceiveCallBack);
            worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,
ReceiveMethod, worker);
}
 delegate void ShowInfoCallback(string text);
private void ShowInfo(string text)
{
            if (this.txtMain.InvokeRequired)
            {
                ShowInfoCallback d = new ShowInfoCallback(ShowInfo);
                this.Invoke(d, new object[] { text });
            }
            else
            {
                txtMain.Text+=text+"\n";

            }
```

}

برای تست کردن برنامه می توانیم از Telnet استفاده کنیم. Telnet یکی از امکانات ویندوز است که به کمک آن می توان به یک آدرس IP و یک شماره پورت خاص وصل شده و به ارسال اطلاعات مشغول شویم. کافی است ابتدا برنامه را اجرا کرده و سپس در منوی RUN تایپ کنید :

```
telnet 127.0.0.1 1800
```

سپس در پنجره مشکی رنگ ظاهر شده ، شروع به تایپ کردن کنید. آنچه شما تایپ می کنید برای برنامه ارسال می گردد و می توانید آن را مشاهد فرمایید.