شناسنامه مقاله

نويسنده: فرشاد فكري نجات

وب سايت: www.fekrinejat.com

ايميل: fekrinejat@yahoo.com

چاپ شده توسط: ماهنامه وب

شماره: ۳۷

صفحه: ۱۵

برنامه نويسي Socket بوسيله .NET

مقدمه :

در گذشته و قبل از متولد شدن محيط هاي توزيع شده و شبكه هاي كامپيوتري مانند اينترنت، برنامه‌ها بصورت محلي و تنها درون يك كامپيوتر اجرا شده و هيچ گونه ارتباط بصورت Remote بين سرويس‌ها و نرم‌افزارهاي مختلف وجود نداشت. خوشبختانه امروزه با رشد اينترنت و نياز نرم‌افزارها به ارتباط با دنياي خارج از محيط بسته يك كامپيوتر، مواجه هستيم. از اين رو برنامه‌نويسان و توسعه گران نرم‌افزار بطور قطع خود را براي انجام اين امر مهم و ارتباط با كامپيوترها تجهيز مي‌كنند. بسياري از نرم‌افزارها مانند مرورگرهاي وب ، سرويس‌هاي Chat قصد تبادل اطلاعات و ارتباط بين كاربران را داشته و يا نرم‌افزارهايي براي بروزرساني Online در محيط شبكه‌اي، نيازمند استفاده از بسترها و ابزارهاي مناسب مي‌باشند. هم اكنون همه اين فعاليت‌ها بوسيله برنامه‌نويسي Socket در زبان‌هاي مختلف برنامه‌نويسي انجام مي‌گيرد. بنابراين يك برنامه‌نويس مترقي در دنياي مدرن امروزي براي تجهيز نرم‌افزار خود مي‌بايست با مفاهيم و اصول اين نوع برنامه‌نويسي براي ارتباط در محيط هاي چند كاربره آشنايي كامل داشته باشد. البته متاسفانه اين روش برنامه‌نويسي گاهي اوقات جهت ايجاد برنامه‌هاي مخرب مانند Torjan ها و برخي ويروس‌هاي پيشرفته توسط برنامه‌نويسان بداندیش و برخي هكرها انجام مي‌گيرد كه كاربران با استفاده از ابزارهاي مفيدي همانند FireWall قادر به مقابله با اينگونه برنامه‌هاي مخرب خواهند بود. در اين مقاله ابتدا به بررسي مفاهيم Client/Server پرداخته و سپس با عبارات Socket ، Port و IP آشنا خواهيد شد. در نهايت شما را با اين سبك برنامه‌نويسي توسط FCL و كلاس‌هاي .NET. آشنا خواهم كرد كه قادر به ايجاد اينگونه نرم‌افزارها به زبان .NET VB و يا هر زبان تحت حمايت .NET Framework. خواهيد بود.

مباني مفهوم Socket :

در اين قسمت به مفاهيم IP و Port بصورت مختصر و مفيد خواهيم پرداخت. بطور قطع اكثر كاربران هوشمند و برنامه‌نويسان متخصص اينترنت با اين مفاهيم آشنايي مناسبي دارند. ولي به دليل پوشش كامل موضوع بر مفهوم Socket ها مي‌بايست مفاهيم پايه‌اي مورد بررسي نسبي قرار گيرند.

تعريف IP :
در دنياي حقيقي همه مكان‌ها از طريق آدرس‌هايي قابل شناسايي است. در زندگي واقعي اين آدرس‌ها شامل كشور، شهر ، خيابان ، كوچه و غيره مي‌باشد. در محيط اينترنت نيز همه كامپيوترهاي متصل به اين دهكده جهاني، داراي يك آدرس منحصر به فرد به نام IP مي‌باشند. اين آدرسدهي در پروتوكل استاندارد اينترنت (TCP/IP) قابل شناسايي و استفاده است. آدرس IP (در IPV4) از ۴ قسمت ۳۲ بيتي تشكيل شده كه هر قسمت داراي عددي بين ۰ تا ۲۵۵ است. به محض اتصال كامپيوتر كاربر به اينترنت، يك آدرس IP به سيستم شما تعلق خواهد گرفت و از اين پس هر كامپيوتر ديگري در اينترنت از طريق آدرس IP ، قادر به شناسايي و ارتباط با شما خواهد بود. پس از قطع شدن كامپيوتر از اينترنت، آدرس IP مورد نظر آزاد شده و توسط ISP و بصورت خودكار به كامپيوتر ديگري كه به اينترنت متصل مي‌گردد سپرده مي‌شود. براي مشاهده آدرس IP كامپيوتر خود در اينترنت، دو روش وجود دارد :

در روش ساده‌تر، پس از اتصال به اينترنت، بر روي دو مانيتور كوچك () گوشه سمت راست پائين (در قسمت System Tray) توسط ماوس كليك كرده و در پنجره ظاهر شده قسمت Detail را انتخاب نماييد. عدد موجود در مقابل عبارت Client IP Address نشان‌دهنده آدرس اينترنت كامپيوتر شما در اينترنت مي‌باشد.

در روش دوم از طريق Start > Run و استفاده از دستور CMD (در ويندوز نسخه ۲۰۰۰ و بالاتر) به محيط خط فرمان مراجعه كرده و دستور IPConfig را صادر نماييد. در قسمت IP Address در نتيجه حاصل از اجراي اين دستور، قادر به مشاهده IP سيستم خود خواهيد بود.

تعريف Port :
كلمه Port در سيستم عامل به مفهوم درگاه بوده و به منفذهايي جهت ارتباط نرم‌افزارها با دنياي اينترنت و تبادل اطلاعات، مورد استفاده قرار مي‌گيرد. در واقع Port هاي نرم‌افزاري براي تكميل آدرسدهي IP ها و بصورت مجازي در سيستم عامل‌هاي مختلف از طريق پروتوكل TCP و يا UDP تعبيه شده‌اند. پورت‌ها در همه كامپيوترها، چه متصل و يا غيرمتصل به اينترنت، وجود دارند. درگاه در سيستم عامل، عددي بين ۰ تا ۶۵۵۳۵ مي‌باشد كه بسياري از آنها طبق استانداردها براي مصارف خاصي رزرو شده‌اند و بسياري نيز به عنوان پورت‌هاي آزاد شناخته مي‌شوند كه برنامه‌نويسان به دلخواه از آنها جهت ارتباط با دنياي اينترنت در نرم‌افزارهاي خود استفاده مي‌كنند. به عنوان مثال از پورت‌هاي رزرو شده مي‌توان به سرويس‌هاي اينترنتي موجود در سيستم عامل مانند FTP (پورت ۲۱) ، SMTP (پورت ۲۵) ، POP3 (پورت ۱۱۰) ، HTTP (پورت ۸۰) و غيره .. اشاره نمود. اما نرم‌افزاري مانند Yahoo Messenger نيز از پورت رزرو نشده ۵۰۵۰ در سرور خود استفاده مي‌كند. معمولا پورت‌هاي بالاي ۱۰۲۴ رزور نشده بوده و شما به عنوان برنامه‌نويس قادر به استفاده از اين پورت‌ها در نرم‌افزارهاي شخصي خود خواهيد بود.

مشاهده وضعيت پورت‌هاي سيستم :
پورت‌ها در سيستم عامل به دو وضعيت بسته (Close) و باز (Open) قابل شناسايي مي‌باشند كه در ابتدا تمامي پورت‌ها بطور پيش‌فرض بصورت بسته بوده و قابل استفاده نيستند. برنامه‌نويسان جهت ارتباط نرم‌افزار با دنياي خارج از اينترنت مي‌بايست از طريق سورس كدهاي برنامه‌نويسي ، پورت دلخواه خود را بر روي كامپيوتر به حالت باز در آوردند. پورت‌هاي باز هميشه در انتظار هرگونه تماسي مي‌مانند و تا برقراري تماس هيچ عملي را انجام نداده و فقط پورت مورد نظر را اشغال كرده و آن را به حالت Listen (گوش داد) و انتظار نگه مي‌دارند. براي مشاهده وضعيت پورت‌ها و ارتباط كامپيوتر شما با دنياي اينترنت، ابزاري به نام Netstat در خود ويندوز تعبيه شده است. جهت استفاده از اين ابزار، از طريق منوي Start > Run ، دستور CMD را اجرا كرده و سپس فرمان Netstat -a را تايپ نماييد. در خروجي اين

دستور پورت‌هاي مختلف و نوع برنامه‌اي كه آن را به حالت باز در آورده را مشاهده خواهيد كرد.

تعريف Socket :
جهت برنامه‌نويسي Socket و تبادل اطلاعات بين كامپيوترها در اينترنت، نياز به هر دو آدرس IP و Port مي‌باشد. هر كدام از مفاهيم IP و Port به تنهايي نشان‌دهنده بخشي از آدرسدهي كامپيوتر مورد نظر مي‌باشند. ولي با تلفيق اين دو، داراي آدرس كاملي بر روي اينترنت خواهيم بود كه به آن Socket گفته مي‌شود.
مانند :

127.0.0.1:80

آدرس 127.0.0.1 به مفهوم كامپيوتر خود شما و آدرس Local مي‌باشد و مثال فوق به مفهوم آدرسدهي پورت ۸۰ (مربوط به HTTP) بر روي اين سيستم است. توجه كنيد كه استفاده از IP بدون Port مانند دادن نشاني منزل بدون پلاك مي‌باشد. در واقع براي پيدا كردن نشاني منزل و يا محل كار شما، تنها آدرس كشور ، شهر و خيابان كافي نبوده و شخص مورد نظر مي‌بايست پلاك و زنگ شما را نيز بداند تا با داشتن آدرس كامل به راحتي مراجعه نمايد.

برنامه‌نويسي Client/Server :
هر نرم‌افزار Client/Server كه در اينترنت مشغول فعاليت مي‌باشد، از دو نرم‌افزار مجزا به عنوان Client و Server جهت ارتباطات مختلف تشكيل شده است. به عنوان مثال نرم‌افزار Yahoo Messenger كه در اختيار تعداد زيادي از كاربران اينترنت قرار دارد به عنوان نرم‌افزار Client مي‌باشد. نرم‌افزار مكملي نيز به نام Server درون سايت شركت ياهو قرار گرفته تا ارتباط كاربران از طريق برنامه Server با يكديگر برقرار گردد. در مجموع بدانيد كه هر نرم‌افزار Client براي اجرا، نياز به يك نرم‌افزار Server دارد و اين دو مفهوم مكمل يكديگر بوده ولي بصورت مجزا طراحي و ايجاد مي‌گردند.

۱- نرم‌افزار Server :
نرم‌افزاري كه درون يك كامپيوتر و يا سرور اجرا شده و يكي از پورت‌هاي سيستم عامل را باز كرده و به انتظار ارتباط از طرف نرم‌افزار Client مي‌ماند. نرم‌افزار Server برخلاف Client فقط يكبار بر روي يك كامپيوتر مشخص (كامپيوتر مقصد) اجرا و نصب مي‌گردد.

۲- نرم‌افزار Client :
نرم‌افزاري است كه به تعداد زياد در اختيار كاربران مختلف قرار گرفته و از طريق برنامه‌نويسي Socket به پورتي كه نرم‌افزار Server آن را باز كرده متصل مي‌گردد.

به عنوان مثال استفاده از مرورگرهاي وب، مصداق و مثال مناسبي جهت نمايش اين مطلب مي‌باشد. شما به عنوان كاربر از مرورگرهايي با نام‌هاي IE ، FireFox ، Opera و غيره جهت ارتباط با اينترنت استفاده مي‌كنيد. اين مرورگرها به عنوان نرم‌افزار Client از طريق پورت ۸۰ (مربوط به HTTP) به سرويس دهنده وب (به عنوان نرم‌افزار Server) مانند IIS و Apache و غيره متصل شده و صفحات وب را براي كاربران مرور مي‌كنند. در اين مثال نيز مشاهده مي‌كنيد كه نرم‌افزار Client (مرورگر وب) به تعداد زياد و نامحدودي در اختيار كاربران قرار دارد ولي نرم‌افزار Server (سرويس دهنده وب) براي هر ISP ، تنها يك نسخه واحد مي‌باشد.

برنامه‌نويسي Socket در چارچوب .NET :

پياده سازي تمامي موارد فوق و ايجاد نرم‌افزار Client و Server از طريق زبان‌هاي مختلف برنامه‌نويسي در همه سيستم عامل‌ها به خوبي قابل انجام است. قبل از متولد شدن فناوري .NET ، برنامه‌نويسان ويندوز از WinSock API و فايل WS2_32.dll جهت ايجاد نرم‌افزارهاي Client/Server استفاده مي‌كردند. اما با رشد صنعت نرم‌افزار و حضور چارچوب دات‌نت به عنوان يك Solution مناسب جهت رفع معايب برنامه‌هاي مختلف، باعث ترفيع سطح كيفي و ارتقاي دانش برنامه‌نويسان شده و امروزه اكثر برنامه‌نويسان به سمت استفاده از اين چارچوب بسيار مفيد در حركت هستند. اين Platform از يك كتابخانه بسيار غني به نام FCL برخوردار بوده و در اين كتابخانه، كلاس‌هايي به نام‌هاي TCPListener و TCPClient در NameSpace ، System.Net.Sockets جهت استفاده از Socket ها و برنامه‌نويسي آنها در نظر گرفته شده كه جايگزين روش WinSock API در مدل قديمي مي‌باشد. در اين بخش مقاله با ارائه سورس كدي ساده و كوچك مربوط به يك برنامه Client/Server ، قابليت مقدماتي و بنيادين اين روش برنامه‌نويسي را به معرض نمايش خواهيم گذاشت. سورس كدهاي ارائه شده زير را در يك پروژه Console Application در نرم‌افزار Visual Studio .NET پياده‌سازي نماييد.

نوشتن برنامه Server :

به قطعه برنامه زير توجه نماييد :

```
Imports System.Net.Sockets
Class Server_Application
    Shared Sub Main()

1:      Dim IP As Net.IPAddress = Net.IPAddress.Parse("127.0.0.1")
2:      Dim TcpListener As New TcpListener(IP, 8000)
3:      TcpListener.Start()
4:      Console.WriteLine("Waiting for connection...")


5:          Dim tcpClient As TcpClient = TcpListener.AcceptTcpClient()
6:          Console.WriteLine("Connection Accepted.")


7:          Dim networkStream As NetworkStream = tcpClient.GetStream()
8:          Dim bytes(tcpClient.ReceiveBufferSize) As Byte
9:          networkStream.Read(bytes, 0, CInt(tcpClient.ReceiveBufferSize))
10:         Dim clientdata As String = _
            System.Text.Encoding.ASCII.GetString(bytes)
11:         Console.WriteLine(("Client sent: " + clientdata))


12:         Dim responseString As String = "Connected to server."
13:         Dim sendBytes As [Byte]() = _
            System.Text.Encoding.ASCII.GetBytes(responseString)
14:         networkStream.Write(sendBytes, 0, sendBytes.Length)


15:         tcpClient.Close()
16:         TcpListener.Stop()
17:         Console.WriteLine("Press Any Key to Exit ...")
18:         Console.ReadLine()
    End Sub
```

```
End Class
```

اين قطعه برنامه به زبان VB .NET نوشته شده ولي به دليل استفاده از FCL ، با تغييراتي كوچك در Syntax اين زبان به راحتي مي‌توان آن را به C# و يا هر زبان تحت پوشش .NET تبديل نمود. خطوط مهم برنامه به دليل سهولت در توضيح عملكرد دستورات، با استفاده از Label شماره گذاري شده كه اين شماره ها هيچ تاثيري در كيفيت و نحوه اجراي برنامه نخواهند داشت.

توضيح :

ابتدا قبل از هر كدي با استفاده از دستور Imports ، NameSpace حاوي كلاس‌هاي Socket را در دسترس برنامه قرار خواهيم داد تا از تكرار ذكر مسير System.Net.Sockets جلوگيري به عمل آيد. در خط اول برنامه، متغيري به نام IP در نظر گرفته شده و توسط كلاس IPAddress موجود در فضاي نام System.Net عمليات ذخيره‌سازي و تنظيم آدرس IP كامپيوتر مورد نظر به درون متغير IP قرار مي‌گيرد كه منظور از آدرس 127.0.0.1 كامپويتر خودتان است. سپس در خط دوم برنامه با استفاده از كلاس TCPListener ، شماره پورت مورد نظر (در اين مثال عدد ۸۰۰۰) را به دلخواه و در صورتي كه توسط برنامه ديگري اشغال نشده باشد در نظر گرفته و سپس توسط متد Start اين كلاس، عمليات باز شدن پورت مورد نظر انجام مي‌گيرد. سپس در خط پنجم برنامه با استفاده از متد AcceptTcpClient به انتظار برقراري ارتباط از طرف يك برنامه Client خواهيم ماند. به محض برقراري ارتباط از طرف برنامه Client ، عبارت موجود در خط ششم برنامه به نمايش در خواهد آمد.

توجه كنيد كه براي انتقال اطلاعات بين برنامه Client و Server مي‌بايست از روش Streaming و با ايجاد تونل، اقدام به دريافت و ارسال پيغام نماييد. به دليل جلوگيري از پيچيده شدن مقاله و عدم درك واضح توسط كاربران تازه كار و مبتدي، به توضيحات جزئي در مورد عمل ايجاد تونل و استفاده از روشهاي Streaming بسنده مي‌كنم. دستورات خطوط ۷ الي ۱۱ در سورس كد فوق، جهت دريافت پيغام از برنامه Client و نمايش آن درون نرم‌افزار Server مورد استفاده قرار گرفته و دستورات خطوط ۱۲ تا ۱۴ نيز عملياتي را جهت انتقال پيغام از نرم‌افزار Server به Client انجام مي‌دهند. دستورات ۱۴ تا ۱۸ نيز جهت بستن ارتباط و قطع جريان انتقال اطلاعات بين نرم‌افزارهاي Client و Server مورد استفاده قرار مي‌گيرند. بطور قطع با تسلط و شناخت كامل‌تري از كلاس‌هاي Sockets ، قادر به ايجاد برنامه‌هاي كاملي مانند Chat ، سيستم بروزرساني و بسياري موارد كاربردي ديگر خواهيد بود كه براي شروع، اين قطعه كد كوچك بسيار مناسب مي‌باشد.

نوشتن برنامه Client :

جهت تكميل تكنيك Client/Server ، بعد از نوشتن برنامه Server هم اكنون نوبت به نوشتن برنامه Client رسيده است.

```
Imports System.Net.Sockets
Class Client_Application
    Shared Sub Main()

1:      Dim tcpClient As New TcpClient
2:      tcpClient.Connect("127.0.0.1", 8000)

3:      Dim networkStream As NetworkStream = tcpClient.GetStream()
4:      Dim Str As String = "Send Message From Client to Server"
5:      Dim sendBytes As [Byte]() = Text.Encoding.ASCII.GetBytes(Str)
6:      networkStream.Write(sendBytes, 0, sendBytes.Length)
```

```
7:        Dim bytes(tcpClient.ReceiveBufferSize) As Byte
8:        networkStream.Read(bytes, 0, CInt(tcpClient.ReceiveBufferSize))
9:        Dim returndata As String = Text.Encoding.ASCII.GetString(bytes)
10:       Console.WriteLine(("Server App Returned: " + returndata))

11:       Console.ReadLine()
      End Sub
End Class
```

توجه داشته باشيد كه آدرس IP و Port تنظيم شده در برنامه Client مغايرتي با اين آدرس‌ها در نرم‌افزار Server نداشته باشد. در ضمن برنامه Client را حتما پس از اجراي برنامه Server اجرا نماييد تا ارتباط به درستي برقرار گردد.

توضيح :

در خط دوم اين برنامه به راحتي با استفاده از متد Connect كلاس TcpClient قادر به تماس با پورت ۸۰۰۰ آدرس 127.0.0.1 خواهيم بود. دستورات خطوط ۳ الي ۶ مربوط به انتقال يك پيغام از طرف برنامه Client به Server بوده و دستورات خطوط ۷ الي ۱۰ نيز جهت دريافت پيغام ارسال شده از Server به Client و نماش آن در برنامه Client مورد استفاده قرار مي‌گيرد. در خط ۱۱ برنامه نيز با استفاده از دستور Console.ReadLine با جلوگيري از خروج برنامه، قادر به مشاهده پيغام‌هاي دريافت شده خواهيد بود.

نكته مهم :

سورس كدهاي فوق و مخصوصا نرم‌افزار Client را ابتدا از طريق منوي Build > Build Solution در VS .NET به فايل اجرايي تبديل كرده و سپس استفاده نماييد.

پس از اجراي برنامه در صورت مواجه با پنجره تاييد صلاحيت از طرف نرم‌افزار Firewall ، با پاسخ مثبت خود، اجازه اجراي نرم افزار Client را صادر نماييد.