ایجاد ارتباط TCP **میان سرویس دهنده و سرویس گیرنده توسط** #C

پروتکل TCP (Transmission Control Protocol) یک پروتکل جریان گرا، ارتباط گرا، قابل اعتماد و نظیر به نظیر مي باشد. ارتباطTC P همانند ارتباط تلفني مي باشد. به عنوان مثال شما (سرویس گیرنده) به منزل دوست خود (سرویس دهنده) زنگ مي زنید. در صورتیکه دوست شما در منزل باشد تلفن را برداشته و با یکدیگر شروع به صحبت مي کنید (نظیر به نظیر، جریان گرا، قابل اعتماد) و در انتها شما و دوستتان گوشي تلفن را قطع کرده (ارتباط گرا) و به ارتباط خود خاتمه مي دهید. به عنوان مثال پروتکل HTTP از TCP استفاده مي کند.TCP از IP به عنوان پروتکل شبکه استفاده مي کند. IP یک پروتکل datagram (بسته هاي داده) گرا و Best-Effort (بسته هاي داده بصورتي فرستاده مي شود که تحویل و صحت آن تضمین نمي گردد) مي باشد.

در این مقاله یک مثال کوچک و ساده در مورد چگونگي برقراري ارتباط بین سرویس دهنده و سرویس گیرنده مورد بررسي قرار مي گیرد. این مثال از دو بخش تشکیل شده است. بخش سرویس دهنده و بخش

سرویس گیرنده.

کلاسهاي مورد نیاز

در این مثال تنها دو کلاس اصلي بکار برده شده است. در سمت سرویس گیرنده، از کلاس System.Net.Sockets.TcpClient و در سمت سرویس دهنده از کلاس System.Net.Sockets.TcpListener استفاده مي گردد. بطور کلي در سمت سرویس گیرنده یک TcpClient به سرویس دهنده متصل مي شود. سپس با یک کانالي (جریان) که به سرویس دهنده اختصاص داده شده است اعمال مربوطه انجام مي پذیرد.

```
// اتصال به سرویس دهنده 127.0.0.1:8080
TcpCleint client = new TcpClient("127.0.0.1", 8080);

// براي ردوبدل كردن داده ها گرفتن كانال ارتباطي
NetworkStream ns = client.GetStream();

//قطع ارتباط با سرویس  دهنده
client.Close();
```

سمت سرویس دهنده شامل مراحلي بيشتري مي باشد، اما صورت كلي كد به همان صورت كد سرویس گیرنده شبیه مي باشد. ابتدا باید یک درگاه محلي را به TcpListener اختصاص داد. در صورتیکه سرویس گیرنده به سرویس دهنده متصل شود به شما یک socket برگردانده مي شود. با استفاده از این socket شما مي توانید یک کانال ارتباطي (جریان) ایجاد نمایید.

```
// ایجاد یک listener برروي پورت ۸۰۸۰
TcpListener listener = new TcpListener(8080);
listener.Start();

//انتظار بمنظور دریافت درخواست از سمت سرویس  گیرنده
Socket server = listener.AcceptSocket();

// ایجاد یک کانال براي ردوبدل كردن داده ها
NetworkStream ns = new NetworkStream(server);
```

```
قطع ارتباط از سرویس  گیرنده//
server.Close();
```

قالب دستورات

در این مثال دستورات با استفاده از <CRLF> (دستورات با Enter از هم جدا مي شوند) از یکدیگر تمیز داده مي شوند. در صورتیکه سرویس گیرنده دستور "GET" را اجرا نماید پیغام "Hello World" براي او ارسال شده و در صورت ارسال دستور "EXIT" عبارت "BYE" براي سرویس گیرنده فرستاده شده و ارتباط قطع مي گردد.

سرویس دهنده

در دنیاي واقعي سرویس دهنده از دو قسمت اصلي تشکیل مي شود. قسمت Functional Layer و دیگري Commad Processor. قسمت پردازشگر دستورات واسطي بین لایه ارتباطي و Functional Layer مي باشد. لایه ارتباطي درخواست خود را به لایه پردازشگر دستورات ارسال مي کند این لایه درخواست را به لایه Functional ارسال کرده و پس از دریافت جواب از لایه Functional آنرا به لایه ارتباطي مي فرستد. سرویس دهنده ها معمولا به صورت سرویس هاي سیستم عامل اجرا مي شوند. در این مثال سرویس دهنده ما تنها یک لایه داشته و به عنوان یک برنامه کنسول اجرا مي گردد.

```
Console.WriteLine("Intitializing Server ...");
TcpListener listener = new TcpListener(8080);
listener.Start();
Console.WriteLine("Server initialized, waiting for incomming connections ...");
```

```
Socket s = listener.AcceptSocket();
// ايجاد يک جريان براي ارتباط
NetworkStream ns = new NetworkStream(s);
StreamReader r = new StreamReader(ns);
```

listener برروي پورت ۸۰۸۰ آماده دريافت درخواست سرويس گيرنده ها مي باشد. متد AcceptSocket در زمان اتصال سرويس گيرنده يک socket را برمي گرداند. اين متد برنامه را تا زمان دريافت درخواست معلق مي کند. در صورت نياز مي توان از يک نخ براي اين منظور استفاده نمود. در ادامه کانال ارتباطي لازمه برقرار مي گردد و از کلاس StreamReader بمنظور دريافت داده ها از اين کانال استفاده مي شود. در اين قسمت فرمانهاي صادر شده از سوي سرويس دهنده بايد بررسي و پاسخ مورد نظر به آن داده شود.

```
bool loop = false;
while (!loop)
{
    // خواندن يک خط کامل
    string command = r.ReadLine();
    string result;
    Console.WriteLine("Executing remote commad: " + command);
    switch (command)
    {
        case "GET":
            result = "Hello World !";
            break;
        // خاتمه ارتباط
        case "EXIT":
            result = "BYE";
```

```
            loop = true;
            break;
        // دستور نامعتبر
        default:
            result = "ERROR";
            break;
    }
    if (result != null)
    {
        Console.WriteLine("Sending result: " + result);
        // افزودن CRLF به انتهاي پيغام پيغام
        result += "\r\n";
        // بايتها تبديل رشته به آرايه اي از //
        Byte[] res = System.Text.Encoding.ASCII.GetBytes( < BR > result.ToCharArray());
        // ارسال نتيجه به سرويس گيرنده //
        s.Send(res, res.Length, 0);
    }
}
```

در صورتیکه دستور GET در یافت شود سرویس دهنده پیغام Hello World را صادر کرده و ادامه دستورات پردازش مي گردد. در صورت صدور دستور ناشناخته نیز پردازش دستورات ادامه مي یابد. در صورت دریافت دستور EXIT پردازش دستورات خاتمه یافته و در ادامه ارتباط با سرویس گیرنده قطع مي شود.

```
Console.WriteLine("Clearing up server ...");
```

```
s.Close();
Console.Write("Press return to exit");
Console.ReadLine();
```

سرويس **گيرنده**

سرويس گيرنده تا حدودي از سرويس دهنده پيچيده تر مي باشد. سرويس گيرنده شامل دو قسمت مي باشد: واسط كاربر (كه يک برنامه كنسول ساده است) و پردازشگر دستورات كه شامل عناصر ارتباطي مي باشد. در ابتدا پردازشگر دستورات مورد بررسي قرار مي گيرد. براي اينكار ابتدا يک interface ايجاد مي كنيم. استفاده از interface قابليت انعطاف پذيري بيشتري در مورد استفاده از پروتكل هاي ديگر به شما مي دهد. در اين حالت سرويس گيرنده تنها اشيائي را مي پذيرد كه واسط مورد نظر را پياده سازي كرده باشند. اين عمل باعث استقلال سرويس گيرنده از پروتكل ارتباطي مي گردد.

```
public interface CommandProcessor
{
   // يک دستور را اجرا كرده و حاصل را برمي گرداند
   // در صورتيكه خروجي آن false باشد پردازش دستورات بايد متوقف گردد
   bool Execute(string command, ref string result);
}
```

در ادامه يک كلاس با نام TCPClientCommandProcessor ايجاد مي گردد. اين كلاس واسط CommadProcessor را پياده سازي مي نمايد. اين كلاس شامل ۳ متد و يک سازنده مي باشد. متد Connect وظيفه اتصال بر سرويس دهنده را برعهده دارد. بوسيله متد DisConnect ارتباط با سرويس دهنده قطع گشته و با متد Execute فرمانهاي مورد نظر به سرويس دهنده ارسال مي گردد.

```
public class TCPClientCommandProcessor : CommandProcessor
{
    // آدرس سرویس دهنده
    string host = null;
    // پورت سرویس دهنده
    private int port = -1;
    // واسط ارتباطي
    private TcpClient client = null;
    // جريان خروجي داده
    private NetworkStream outStream = null;
    // داده جريان ورودي
    private StreamReader inStream = null;

    public TCPClientCommandProcessor(string host, int port)
    {
        this.host = host;
        this.port = port;
        this.Connect();
    }

    // اتصال به سرویس دهنده
    public void Connect()
    {
        Console.WriteLine(String.Format("Connecting to {0}:{1} ...",
```

```
    this.host, this.port) );
    this.client = new TcpClient(this.host, this.port);
    this.outStream = this.client.GetStream();
    this.inStream = new StreamReader(this.outStream);
    Console.WriteLine(String.Format("Connected to {0}:{1}",
    this.host, this.port) );
}

// قطع ارتباط با سرویس دهنده
public void DisConnect()
{
    if (this.client != null)
    {
        this.client.Close();
        Console.WriteLine(String.Format("Connection closed: {0}:{1}",
        this.host, this.port) );
    }
}

// دستورات اجراي
public bool Execute(string command, ref string result)
{
    bool ret = true;
    command += "\r\n";
    Byte[] cmd = System.Text.Encoding.ASCII.GetBytes(
    command.ToCharArray() );
```

```
        // درخواست ارسال
        this.outStream.Write(cmd, 0, cmd.Length);
        // دریافت پیام
        result = this.inStream.ReadLine();
        ret = !result.Equals("BYE");
        return ret;
    }
}
```

واسط کاربر سرویس گیرنده

واسط کاربر سرویس گیرنده یک برنامه کنسول ساده مي باشد. در این برنامه یک شئ از کلاس TCPClientCommandProcessor ایجاد شده و اعمال مربوطه انجام مي گردد.

```
Console.WriteLine("Initializing client ...");
TCPClientCommandProcessor proc = new
TCPClientCommandProcessor("127.0.0.1", 8080);

string result = "";
string cmd = null;
while (!result.Equals("BYE"))
{
    Console.Write("Command : ");
    cmd = Console.ReadLine();
```

```
    proc.Execute(cmd, ref result);
    Console.WriteLine(result);
}

Console.WriteLine("Closing connection...");
proc.DisConnect();
Console.Write("Press return to exit");
Console.ReadLine();
```

برگرفته ار کتاب C#.NET Web Developer’s Guide انتشارات SYNGRESS