Writing by FADI Abdel-Qader/fadi822000@yahoo.com
My Space: http://spaces.msn.com/members/csharp2005

Chapter 2
Streaming & Network Layer Programming

2- Streaming and Network Layer Programming

a. Managed I/O: Streams, Readers, and Writers
   i. Stream Classes
   ii. Stream Members
   iii. Stream Manipulation
   iv. Simple Remote Control Application Using StreamReader & StreamWriter Classes

b. Socket & Network Layer Programming
   i. Socket Programming
   ii. Socket Class Members
   iii. TCP & UDP Classes Members
   iv. Asynchronous Sockets

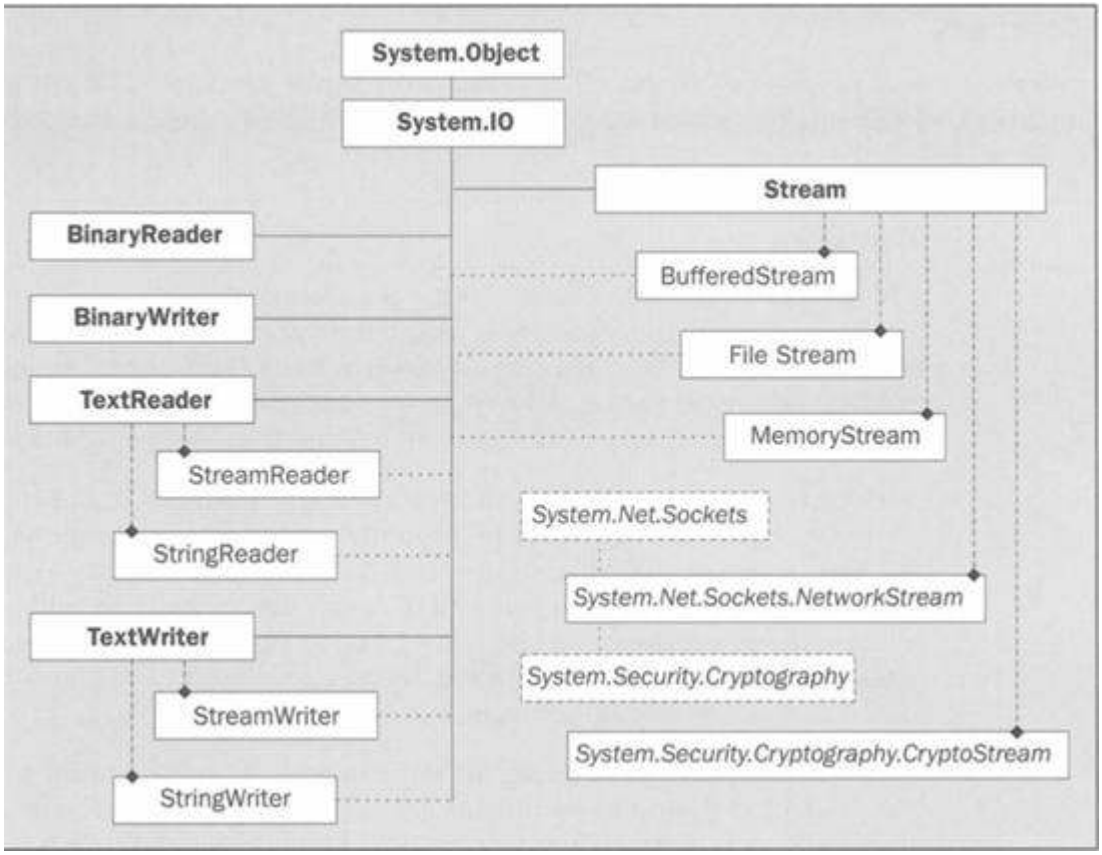
بسم الله الرحمن الرحيم

الدرس العاشر: Managed I/O: Streams, Readers, and Writers :

تحدثنا سابقا في الجزء الأول بشكل عام عن استخدامات ال Streams Library واستخدامها لإرسال Binary Data و Text Data من جهاز إلى آخر وكمثال قمنا بإرسال صورة من ال Client إلى ال Server باستخدام ال Binary Reader & Binary Writer ..
إن الهدف من إنشاء مكتبات ال Stream هو تسهيل عملية نقل البيانات من مكان إلى أخر سواء عبر الشبكة أو داخل نفس الجهاز كما هو الحال بتعامل مع الملفات أو التعامل مع الطابعة أو أي طرفية أو جهاز آخر موصول بالكمبيوتر حيث تسهل علينا عملية تحويلها إلى Byte Array وإرسالها وهو ما حل الكثير من المشاكل التي كانت تواجه المبرمجين في التعامل مع Binary Data ..
يمكن التعامل مع ال Stream بأسلوبين المتزامن **Synchronous** والغير متزامن **Asynchronous** وبشكل افتراضي تعمل جميع ال IO Streams بالأسلوب المتزامن لاكن العيب فيه هو تأثيره الشديد على أدائية النظام إذ يقوم بإغلاق ال Processing Unit في ال Thread المخصصة للبرنامج بحيث لا يسمح بتنفيذ أي أمر آخر إلا بعد الانتهاء من العملية الجارية ولا ينصح ابدآ استخدام الأسلوب المتزامن في حالة إذا كنت تتعامل مع أجهزة قراءة وكتابة بطيئة نسبيا مثل ال Floppy Disk أو ال Magnetic Tape لكنها مهمة جدا بالبرمجيات التي تعتمد على أنظمة الزمن الحقيقي أو ال Real Time Systems حيث أنها تعتمد الأسلوب المتزامن في عملية إرسال واستقبال البيانات وهو ما يمنع القيام بأي عملية أخرى إلى حين الانتهاء من تنفيذ الأمر ومن الأمثلة عليها أنظمة السحب أو الإيداع في الرصيد البنكي أو أنظمة حجز التذاكر أو شحن بطاقة الهاتف وغيرها .. طبعا في حالة إذا كان برنامجك لا يحتاج إلى وجود الخواص السابقة عندها ينصح باستخدام الأسلوب الغير متزامن Asynchronous حيث تستطيع من خلاله تنفيذ عمليات أخرى في وحدة المعالجة وبدون الحاجة لانتظار إنهاء العملية الجارية إذ يتم إنشاء Separate thread لكل عملية طلب إدخال أو إخراج مما لا يؤثر على أدائية النظام وينصح باستخدامه إذا كانت عملية القراءة أو الكتابة تجرى من خلال أجهزة بطيئة نسبيا ويمكن تميز الميثود المتزامن عن الغير متزامن في الدوت نيت بوجود كلمة Begin أو End في بداية اسم الميثود الغير متزامن وكمثال عليها BeginWrite و BeginRead و ال EndWrite و ال EndRead ..

أولا: Stream Classes

تدعم الدوت نيت عمليات ال Streams بمجموعة من ال Classes والمندرجة تحت النيم سبيس System.IO والتي تستخدم لعمليات الإدخال و الإخراج لنقل البيانات .
تستخدم بعض ال Stream Classes ، Backing storage ، ومن الأمثلة عليها FileStream و ال BufferedStream وال MemoryStream وكذلك فإن بعضها لا يستخدم أي Back Storage ومن الأمثلة عليها ال NetworkStream والتي تستخدم لنقل ال Stream عبر الشبكة وبدون استخدام Backing Storage ، و تقسم ال Stream Classes في الدوت نيت كما في الشكل التالي :



1- BufferedStream Class : ويستخدم بشكل أساسي لحجز مقدار معين من الذاكرة بشكل مؤقت لتنفيذ عملية معينة كما تستخدم بعض البرمجيات ال Buffering لتحسين الأدائية حيث تكون كذاكرة وسيطة بين المعالجة و الإرسال أو الاستقبال وكمثال عليها برمجيات الطباعة حيث تستخدم الطابعة ذاكرة وسيطة لتخزين البيانات المراد طباعتها بشكل مؤقت ، يكمن الهدف الأساسي من استخدام ال Buffering في العمليات التي يكون فيها المعالج أسرع من عمليات الإدخال و الإخراج حيث يتم معالجة البيانات ووضعها في ال Buffer في انتظار إرسالها وهو ما يساهم في تحسين الأدائية بشكل كبير ، ويستخدم ال BufferedStream عادتا في برمجيات الشبكات مع ال NetworkStream لتخزين البيانات المراد إرسالها عبر الشبكة في الذاكرة حيث لا يستخدم هذا الكلاس Backing storage كما ذكرنا سابقا ..

بشكل افتراضي يتم حجز 4096 bytes عند استخدام ال BufferedStream ويمكن زيادتها أو تقليلها حسب الحاجة .. ويستخدم ال BufferedStream كما يلي كمثال :

```
using System;
using System.Text;
using System.IO;
namespace Network_Buffering
{
   class Program
   {
      static void Main(string[] args)
      {
         ASCIIEncoding asen = new ASCIIEncoding();
         byte[] xx = asen.GetBytes("Hello Buffering");

         MemoryStream ms = new MemoryStream(xx);
         readBufStream(ms);
```

```
        }
  public static void readBufStream(Stream st)
      {
       // Compose BufferedStream
BufferedStream bf = new BufferedStream(st);
byte[] inData = new Byte[st.Length];

// Read and display buffered data
bf.Read(inData, 0, Convert.ToInt32(st.Length));
Console.WriteLine(Encoding.ASCII.GetString(inData));
        }
    }
}
```

حيث قمنا بتحويل نص إلى Byte Array باستخدام ال ASCIIEncoding وتحميله في عبر ال MemoryStream ثم ارسلناه إلى الميثود readBufStream والتي انشأناها حيث استقبلنا من خلالها ال Stream وحملناه في ذاكرة مؤقتة بستخدام الكلاس ال BufferedStream ثم قمنا بطباعة محتوياته بعد تحويله إلى نص مرة اخرى بستخدام الEncoding.ASCII وطباعته ..

2- MemoryStream Class : وهو شبيه بعملية ال Buffring السابقة إذ يعتبر كحل جيد لتخزين البيانات بشكل مؤقت في الذاكرة قبل الإرسال أو الأستقبال حيث يغنيك عن تخزينها على شكل ملف مما يسرع العملية بشكل كبير ويستخدم كما يلي كمثال حيث استخدمنها لتخزين صورة في الذاكرة :

```
MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

byte[] arrImage = ms.GetBuffer();
ms.Close();
```

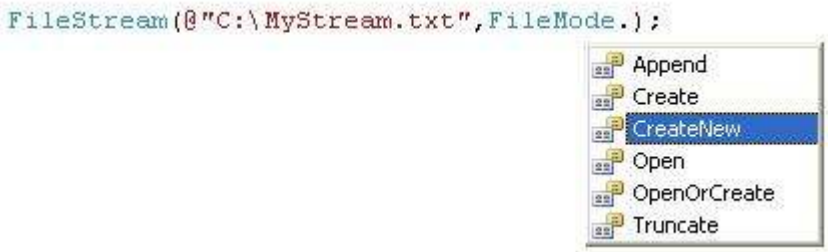
3- NetworkStream Class : وكما قمنا باستخدامها سابقا ، حيث تقوم بتعامل مع ال Stream لإرساله عبر الشبكة باستخدام ال Socket ويتم استدعائها من النيم سبيسس System.Net.Sockets ويعتبر الكلاس NetworkStream بأنه unbuffered إذ لا يحتوي على Backing Storage ويفضل استخدام ال BufferedStream Class معه لتحسين الأداء وتستخدم كما يلي كمثال حيث نريد إرسال الصورة التي قمنا بتخزينها في المثال السابق بذاكرة إلى جهاز أخر عبر ال Socket :

```
TcpClient myclient = new TcpClient ("localhost",5020);//Connecting with
server

NetworkStream myns = myclient.GetStream ();

BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to above address
mysw.Close ();
myns.Close ();
myclient.Close ();
```

4- FileStream : يتم استدعائها باستخدام النيم سبيسس System.IO وتستخدم بشكل اساسي في التعامل مع الملفات سواء للكتابة إلى ملف أو القراءة من ملف وتعتبر هذه الكلاس Backing Storage Class حيث تستخدم ذاكرة Buffer لتحزين البيانات بشكل مؤقت في الذاكرة لحين الإنتهاء من عملية الكتابة أو القراءة ومن الأمور الهامة فيها تحديد مسار الملف المراد القراءة منه أو الكتابة عليه وتستخدم كما يلي :

```
FileStream FS = new FileStream(@"C:\MyStream.txt",
FileMode.CreateNew);// Any Action For Example CreateNew to Create Folder
```

يمكننا استخدام ال Enumeration التالية مع ال FileMode :



1- Append لإضافة نص ما إلى الملف الموجود اصلا
2- Create لإنشاء ملف جديد ويقوم بعمل overwriting في حالى إذا كان الملف موجود بشكل مسبق
3- CreateNew وهو كما في ال Create إلا انه يعطي Exception في حالة وجود الملف بشكل مسبق
4- Open لقراءة ملف ما حيث يعطي Excption في حالة عدم وجود الملف المحدد
5- OpenOrCreate في حالة إذا وجد الملف يقوم بقراءته وفي حالة عدم وجوده يقوم بإنشائه.
6- Truncate ويستخدم لحذف محتويات الملف وجعله فارغا

ثانيا : Stream Members :

هنالك مجموعة من الخواصو و الميثودس التي تشترك بها مكتبات ال Stream وهي كما يلي :

1- CanRead و CanWrite وتستخدم لمعرفة إذا كان ال Stream المستخدم يقبل عملية القراءة أو الكتابة أم لا حيث ترجع قيمة True في حالة إذا كان يقبل و False في حالة أنه لا يقبل ويستخدم عادة قبل إجراء عملية القراءة أو الكتابة لفحص مدى الصلاحية قبل المحاولة ..
2- CanSeek حيث يستخدم ال Seeking عادة لتحديد موقع ال Current Stream والعادة تدعم الكلاسات التي تستخدم Backing Storage هذه العملية مثل ال

FileStream وعندها ترجع قيمة True وترجع قيمة false في حالة إذا كان ال Stream Class لا يحتوي على Backing Storage .
3- CanTimeout وترجع قيمة True في حالة إذا كان ال stream يحتوي على خاصية ال Timeout والتي تعطي وقت محدد للعملية .
4- Length وتستخدم لمعرفة حجم ال Stream بال Byte ويمكن الاستفادة منها لمعرفة نهاية ال Stream أو لتحديد حجم المصفوفة بناء على حجم ال Stream .
5- Position وتستخدم ال Get و Set لمعرفة أو تحديد الموقع ل Stream
وتشترك مكتبات ال Stream بمجموعة من الميثودس وهي كما يلي :

1- المثودس المتزامنة Synchronous Methods :

I. Read و ReadByte وتستخدم لقراءة Stream Data وتخزينه في ال Buffer ويمكن تحديد عدد البايتات التي سيتم قراءتها باستخدام ال ReadByte كما نستطيع من خلالها معرفة نهاية ال Stream حيث ترجع ال Read قيمة 0 وال ReadByte قيمة -1 في حالة انتهاء ال Stream.
II. Write وال WriteByte وتستخدم لعملية الإرسال عبر ال Stream ويمكن تحديد عدد البايتات التي سيتم كتابتها في كل مرة باستخدام ال WriteByte.

2- المثودس غير المتزامنة Asynchronous Methods :

I. BeginRead وال BeginWrite وتستخدم لعملية القراءة أو الكتابة باستخدام ال Stream الغير المتزامن وتأخذ خمسة بارومترات كما في الشكل التالي :

```
FS.BeginRead(
IAsyncResult FileStream.BeginRead (byte[] array, int offset, int numBytes, AsyncCallback userCallback, object stateObject)
array: The buffer to read data into.
```

1- ال Byte Buffer والتي سوف تستخدم لعملية القراءة منه أو الكتابة عليه
2- ال offset والذي سوف يحدد فيه موقع القراءة أو الكتابة
3- ال numByte والذي سوف يتم فيه تحديد الحد الأقصى من البايتات التي سيتم كتابتها أو قراءتها
4- ال AsyncCallback وهو Optional Delegate حيث يتم استدعائه عند الانتهاء من عملية القراءة أو الكتابة
5- ال Stateobject وهي User Provided Object وتستخدم لتميز ال Read & Write Request عن غيره ال Requests .
ترجع ال Begin Methods ال IAsyncResult والذي يمثل حالة ال Stream Operation .

II. EndRead وال EndWrite وتستخدم في حالة إذا أردنا تنفيذ ال Stream Operation بعد الانتهاء من ال Stream Operation الحالي، حيث يبقى بانتظار انتهاء العملية السابقة ثم ينفذ العملية المطلوبة

وهناك بعض المثود والتي تستخدم لإدارة ال Stream وهي :

1- Flush وتستخدم لتفريغ محتويات ال Buffer بعد إتمام العملية المحددة حيث يتم نقل محتويات ال Buffer إلى ال Destination الذي تم تحديده في Stream Object.

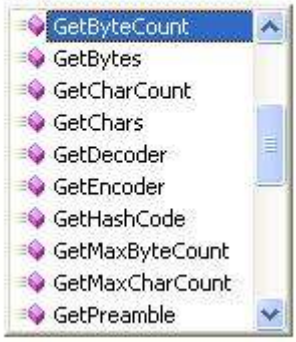
2- Close وتستخدم لإغلاق ال Stream وتحرير ال Resources المحجوزة من قبل ال Stream Object وينصح باستخدامها في الجزء الخاص ب Finally block ولتأكد من أن ال Stream سيتم إغلاقه وتحرير كافة الموارد في حالة حدوث أي Exception إثناء التنفيذ ولضمان عدم بقاء هذه الموارد في الذاكرة بعد إغلاق البرنامج.

3- SetLength وتستخدم لتحديد حجم ال Stream والذي نريد إرساله أو استقباله لاكن في حالة إذا كان ال Stream أقل من المحدد في ال SetLength سوف يؤدي ذالك إلى انقطاع ال Stream وعدم وصوله بشكل سليم ، لن تستطيع استخدام هذه الخاصية إلا إذا تأكدت انك تملك الصلاحية لذلك من خلال الخاصية CanWrite و CanSeek لذا ينصح بفحص الصلاحية أولا قبل تحديد حجم ال Stream .

ثالثا : Stream Manipulation:

يمكن استخدام مكتبات ال Stream لنقل Binary Data أو Text وفي العادة يتم استخدام ال BinaryReader و ال BinaryWriter لتعامل مع ال Binary Data ويتم استخدام ال StreamReader وال StreamWriter لتعامل مع ال Text ،
ويتم استخدام ال ASCIIEncoding أو UnicodeEncoding لتحويل من Stream إلى Text عند الاستقبال ومن Text إلى Stream عند الإرسال حيث تستخدم مجموعة من الميثودس وهي كما في الشكل التالي :

```
ASCIIEncoding asc = new ASCIIEncoding();
asc.
```



1
- GetByteCount وهي Overloaded Method حيث تأخذ Character Array أو String وترجع عدد البايتات التي سوف نحتاجها لنقل نص معين ..
2- GetBytes لتحويل ال String إلى Byte Array حتى نستطيع إرسالها باستخدام ال Stream .
3- GetCharCount حيث تأخذ Byte Array وترجع عدد الأحرف التي سوف تكون في ال String أو في ال Character Array .
4- GetChars وتستخدم لتحويل من Byte Array إلى String وتستخدم عند استقبال البيانات من ال Stream حيث نحولها إلى نص مرة أخرى .

ولتعامل مع ال StreamReader و ال StreamWriter لنقل Text يجب أولا استدعائها من ال System.IO نيم سبيسس وتستخدم كما يلي:

StreamReader للقراءة من ملف:

```
StreamReader str = File.OpenText(openFileDialog1.FileName);
textBox1.Text = str.ReadToEnd();
```

StreamWriter للكتابة إلى ملف:

```
string fname = saveFileDialog1.FileName;
StreamWriter fsave = new StreamWriter(fname);
fsave.WriteLine(textBox1.Text);
```

و لتعامل مع ال BinaryReader وال BinaryWriter لنقل Binary Data يتم استدعائها من ال System.IO نيم سبيسس وتستخدم كما يلي:

BinaryReader لقراءة Binary Data من ال Stream:

```
NetworkStream myns = new NetworkStream(mysocket);
BinaryReader br = new BinaryReader(myns);
```

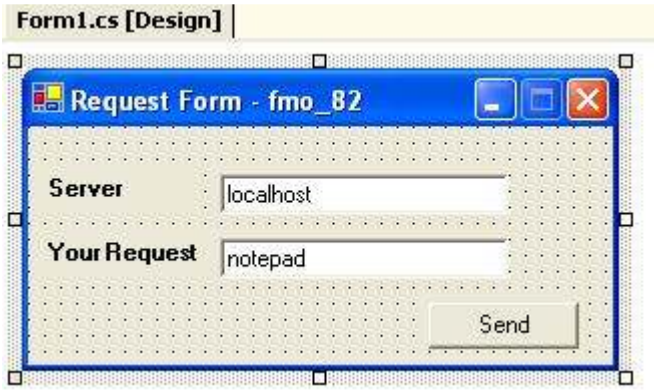
BinaryWriterr لإرسال BinaryData إلى ال Stream عبر ال Socket:

```
TcpClient myclient = new TcpClient("localhost", 5020);
NetworkStream myns = myclient.GetStream();
BinaryWriter mysw = new BinaryWriter(myns);
mysw.Write(arrImage);
```

Remote Control Example باستخدام ال Stream Reader & Writer:

مثال تطبيقي بسيط سوف نستخدم فيه برنامج شبيه ب Chatting لاكن سوف نستخدمه لإعطاء أوامر إلى ال Server حيث يفترض إذا قمنا بإرسال كلمة notepad إلى ال server بأن يقوم بفتح ال notepad فيه وإذا قمنا مثلا بكتابة Calc وإرسالها إلى السيرفر سوف يفتح الآلة الحاسبة فيه وهكذا :

أولا : إنشاء برنامج الإرسال Client : لا يختلف برنامج الإرسال عن برنامج ال Client Chat الذي قمنا بإنشائه في ال Chapter1 ويستخدم فيه كل من TCP Connection وال NetworkStream و ال StreamWriter لإجراء عملية الإرسال فباستخدام الميثود WriteLine الموجودة ضمن ال StreamWriter Object تتم عملية تحويل النص المكتوب في ال Textbox إلى مجموعة من ال Bytes ليتم إرسالها باستخدام ال NetworkStream عبر ال TCP Socket Connection إلى برنامج السيرفر وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي :



ثم قم بإضافة النيم سبيسس التالية :

```
using System.Net.Sockets ;
using System.IO;
```

في Send Button قم بكتابة الكود التالي:

```
try
        {
TcpClient myclient = new TcpClient (txt_host.Text,5020); // تعريف السوكت
NetworkStream myns = myclient.GetStream (); // إسناده إلى اللستريم اوبجكت
StreamWriter mysw = new StreamWriter (myns);
mysw.WriteLine(txt_msg.Text);
```

```
mysw.Close ();
myns.Close ();
myclient.Close ();
        }
catch (Exception ex) {MessageBox.Show (ex.Message );}
```

ولإنشاء برنامج ال Server والذي يعمل على استقبال ال Stream وتحويله إلى Text مرة أخرى .. قم بإنشاء مشروع جديد كما في الشكل التالي :

قم بإضافة النيم سبيسس التالية :

```
using System.Net.Sockets ;
using System.IO;
using System.Threading;
```

ثم إضافة التعاريف التالية :

```
TcpListener mytcpl;// Objects  Declaration
Socket mysocket;
NetworkStream myns;
StreamReader mysr;
```

ثم نقوم بإنشاء ميثود جديدة كما يلي :

```
void our_Server ()
                {
mytcpl = new TcpListener (5020);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();// Accept Any Request From Client and
Start a Session
myns = new NetworkStream (mysocket);// Receives The Binary Data From
Port
mysr = new StreamReader (myns);// Convert Received Data to String
string order = mysr.ReadLine();

// you can add any order and Response Here
if (order=="notepad") System.Diagnostics.Process.Start("notepad");
else if (order=="calc") System.Diagnostics.Process.Start("calc");
else MessageBox.Show("Sorry Sir Your Request is not in my hand",order);

mytcpl.Stop();// Close TCP Session

if (mysocket.Connected ==true)// Looping While Connected to Receive
Another Message
        {
        while (true)
```

```
{
our_Server ();// Back to First Method
}
}
        }
```

حيث تقوم هذه الميثود بتصنت على ال Socket في حالة ورود أي Request يقوم بالموافقة عليه وإنشاء Session جديدة معه وفي حالة ورود أي بيانات عبر السوكت يتسلمها باستخدام ال StreamReader ويحولها إلى Text ثم نقوم بفحص الرسالة باستخدام الجمل الشرطية فمثلا إذا كانت الرسالة هي notepad يتم استدعائها باستخدام الميثود Start الموجودة ضمن الكلاس Process والموجودة في النيم سبيسس System.Diagnostics...
ولتشغيلها ضمن Thread جديد لابد من وضع تعريف الثريد في حدث بدأ التشغيل للفورم كما يلي :

```
private void Form1_Load(object sender, System.EventArgs e)
        {
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(our_Server));
myth.Start ();
        }
```

ثم قم بإضافة التالي في حدث ال Form Closing وذلك لتأكد من إغلاق السوكت وال Stream في البرنامج ..

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
        {
mytcpl.Stop ();
Application.ExitThread ();
Application.Exit();
        }
catch (Exception ex) {MessageBox .Show (ex.Message );}
```

وهنا قد تم الانتهاء من شرح ال Stream وال Stream Class في الدرس القادم سوف نتحدث عن ال Network Layer Programming والتي سوف نتطرق فيها إلى أمور أكثر تقدما في برمجة TCP وال UDP وال Members الخاصة بها...

الدرس الحادى عشر: Socket & Network Layer Programming :

في هذا الجزء سوف نبين بشكل أكثر تفصيلا عن برمجة طبقة ال Network Layer وهي التي يتم التعامل معها لإرسال واستقبال البيانات بعد تحويلها من و إلى Stream عبر الشبكة، قمنا سابقا باستخدام ال TCP و UDP للإرسال وللاستقبال وبينا الفرق بينهما وفي هذا الجزء سوف نتحدث عن ال Socket Programming وال TCP & UDP Classes وفي النهاية سوف نتحدث عن Asynchronous Socket .

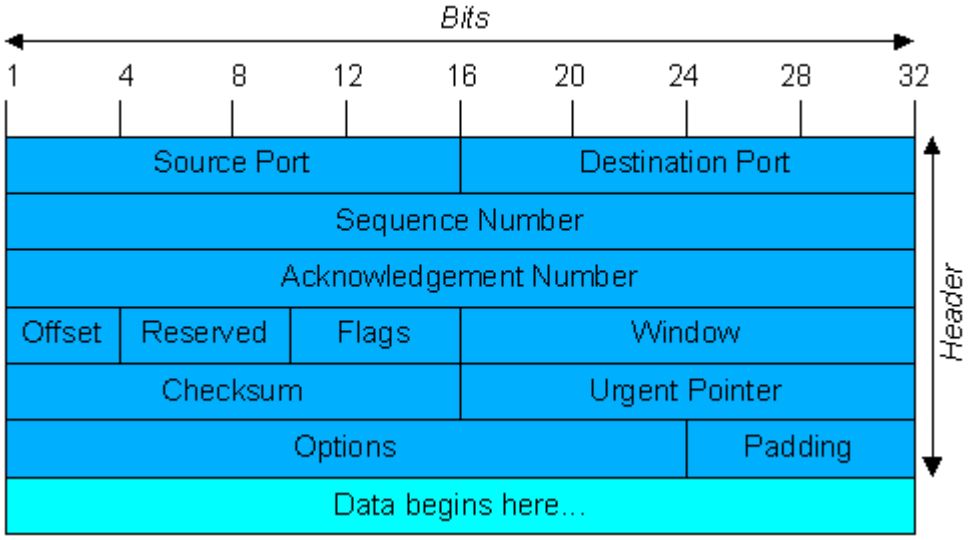
أولا: Socket Programming

من المعروف أن السوكت هي الأداة التي يتم نقل البيانات من خلالها من جهاز إلى أخر ولاستخدامها يلزم في البداية تعريف النيم سبيس System.Net.Sockets حيث يحتوي هذا النيم سبيس على عدد ضخم من ال Classes والتي يتم استخدامها في برمجيات الشبكة وسوف نتحدث عن أهما وهو Socket Class إذ يمكننا بمن التعامل مع ال TCP أو ال UDP أو مع أي نوع أخر من البرتوكولات بشكل مباشر ويتكون ال Socket Object Method من ثلاثة باروميترات كما يلي:

```
Socket MySocket = new Socket(AddressFamily. , SocketType., ProtocolType.);
```

حيث يتم في الباروميتر الأول تحديد نوعية ال IP Address والذي سوف تتعامل معه ويعطيك عدد كبير من الخيارات ومنها IPX والمستخدم في شبكات ال Novel أو ATM والمستخدم في شبكات ال ATM Networks أو NetBIOS Address وغيرها ... ومن أهم هذه الخيارات الInterNetwork وهو ما نستخدمه بشكل دائم مع البرمجيات الخاصة بالشبكات ويعرف على أن نوع IP هو من النوع IPv4 وهو المعتاد مع نظام مايكروسوفت وأغلب أنظمة التشغيل المعروفة حاليا وفي المستقبل القريب جدا سيتم الإستغناء عنه وليحل محله ال IPv6، في الباروميتر الثاني يتم تحديد نوع ال Socket أي هل سوف نستخدم Stream لإرسال البيانات أو شيء اخر وعادة ما يتم استخدام ال Stream لهذه المهمة حيث اننا سنعتمد نمطية التراسل من النوع Stream ، وأخيرا نحدد نوع البروتوكول المستخدم للإتصال فهل هو من النوع UDP أو TCP أو بروتوكولات أخرى مثلICMP Internet Control Message Protocol أو IGMP Internet Group Management Protocol أو اننا نريد مثلا إنشاء ال Socket لتعريف IP Security Header بإختيار IPSecAuthenticationHeader وغيرها وسوف نأتي على شرح مثل هذه الأمور لاحقا إنشاء الله، وهنا سوف نختار ال TCP أو UDP ومن المعروف أن برتوكول ال TCP هو برتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما محدد أما برتوكول ال UDP فهو برتوكول سريع نسبيا و لاكنه لا يدعم عملية التحقق من الوصول السليم للبيانات المرسلة وهو مفيد جدا لإجراء عملية البث الإذاعي Broadcast وإنشاء مجموعات البث Multicast Group وهو ما شرحناه في الدرس الأول والثاني أنظر إلى الشكل التالي ويبين فيه ال Header الخاص بال TCP وال Header الخاص بال UDP ولاحظ الفرق بينهما:

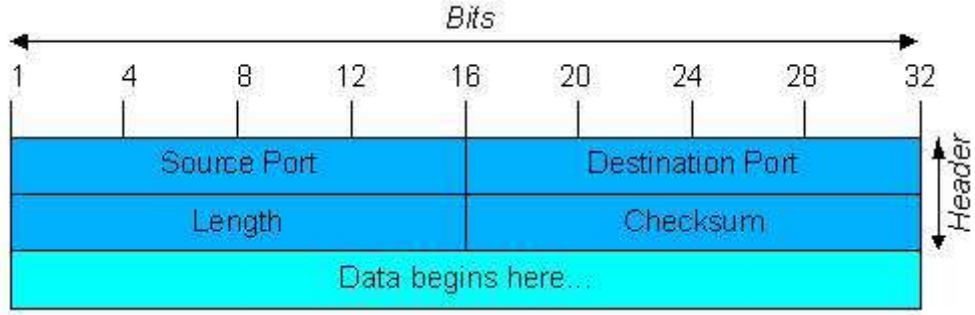
أولا ال TCP Header ويتكون من 32 Bits للبكت الواحد حيث يتم فيه تخزين عنوان المرسل في 16 Bits والمستقبل في 16 Bits والرقم التسلسلي للبكت في 32 Bits ورقم التحقق بالإضافة إلى ال Checksum وفي النهاية يتم وضع الجزء الخاص بالبيانات :



TCP Header

Data Offset: 4 bits the number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.
Window: The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.
...

ثانيا ال UDP Header ويتكون من 32 Bits من البيانات للبكت الواحد ويحتوي على عنوان المرسل 16 Bits أما المستلم و ال Checksum فهما اختياريان وبشكل افتراضي لا يتم استخدامهم في عملية الإرسال:

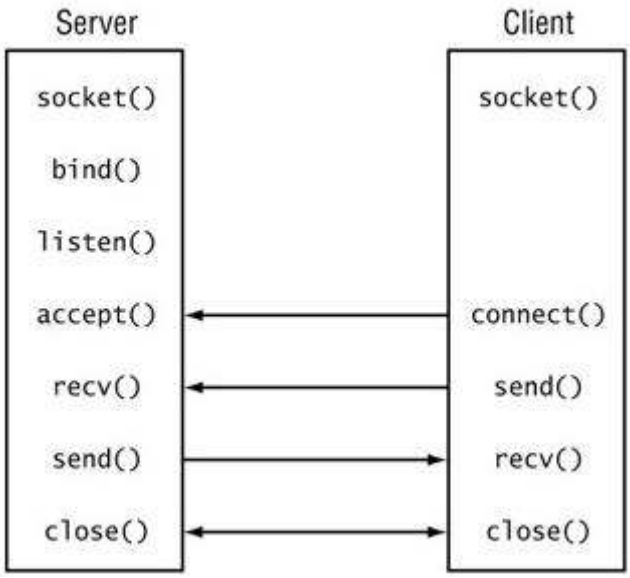


UDP Header

The Checksum in UDP Header. 16 bits.
Computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is cleared to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF.
...

استخدام ال Socket Programming لإنشاء TCP Connection :

تمر عملية الاتصال باستخدام ال TCP Socket Connection بمجموعة من المراحل وهي كما في الشكل التالي :



إذ تبدأ العملية في ال Client و ال server بإنشاء ال Socket كما يلي :

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

ثم ربط ال Socket مع الكمبيوتر الحالي باستخدام الميثود Bind وتستخدم فقط عند الاستقبال وكما يلي :

```
IPEndPoint ip = new IPEndPoint(IPAddress.Any, 5020);
MySocket.Bind(ip);
```

ثم القيام بعملية التصنت على البورت المحدد باستخدام الميثود listen ويمكنك تحدد عدد الأجهزة التي سيتم قبولها ولوضع عدد غير محدد نمرر له الرقم -1 ثم نقوم بالموافقة على الاتصال باستخدام الميثود accept وكما يلي :

```
MySocket.Listen(-1);
MySocket.Accept();
```

ويتم استقبال البيانات من خلال الميثود Receive حيث تعبئ البيانات في مصفوفة من النوع Byte وكما يلي :

```
byte[]Received=new byte[1024];
MySocket.Receive(Received);
```

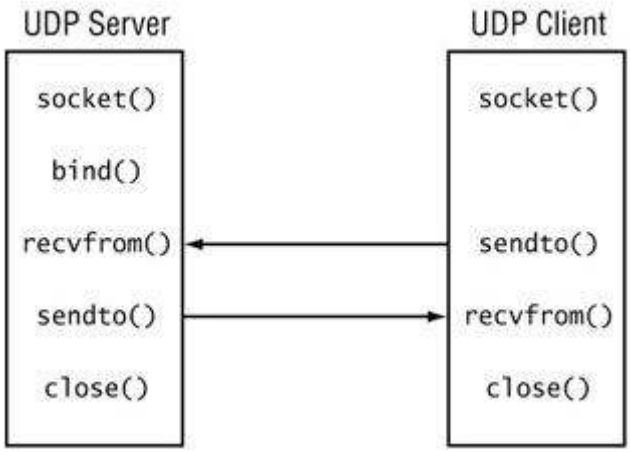
وهنا قمنا بإنشاء Connection من النوع TCP وبتعريفها على البورت(5020 كمثال) حيث يتم ربطها بال Socket باستخدام الميثود Bind وقمنا بتعريف Listen لا نهائي العدد -1 ..

ولتعريف برنامج الإرسال TCP Client باستخدام ال Socket لابد من تعريف السوكت مرة أخرى وإسناد عنوان السيرفر ورقم البورت بنقطة الهدف IPEndPoint ثم إرسال البيانات باستخدام الميثود Send وتتم عملية الإرسال بما تم تعريفه في ال socket حيث سنستخدم Stream Socket وكما يلي :

```
String str = Console.ReadLine();
ASCIIEncoding asen = new ASCIIEncoding();
byte[] msg = asen.GetBytes(str);

Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint remote = new IPEndPoint(IPAddress.Parse("192.168.1.101"),
5020);
MySocket.Connect(remote);
MySocket.Send(msg);
MySocket.Close();
```

استخدام ال Socket Programming لإنشاء UDP Connectionless :

تمر عملية الاتصال باستخدام ال UDP Socket Connection بمجموعة من المراحل وهي كما في الشكل التالي :



وتتشابه عملية الاتصال كما في ال TCP إذ تبدأ العملية في ال Client و ال server بإنشاء ال Socket كما يلي :

```
Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Udp);
```

ثم ربط ال Socket مع الكمبيوتر الحالي باستخدام الميثود Bind وتستخدم فقط عند الاستقبال وكما يلي :

```
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 5020);
MySocket.Bind(sender);
```

ولاستقبال البيانات نستخدم الميثود ReceiveFrom حيث نعرف في البداية End Point Reference بناء على ما تم تعريفه في السابقة ونمرره ك reference مع مصفوفة ال Byte إلى الميثود ReceiveFrom ومن ثم نستطيع تحويل المصفوفة إلى String من خلال الميثود GetString الموجودة ضمن الكلاس ASCII وكما يلي :

```
int recv;
byte[] data = new byte[1024];
EndPoint Remote = (EndPoint) (sender);
recv = newsock.ReceiveFrom(data, ref Remote);
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

ويتم في الإرسال استخدام الميثود SendTo حيث نمرر لها البيانات بعد تحويلها من String إلى Byte Array وحجم البيانات المرسلة إذ يمكننا معرفته من خلال الميثود Length ونوع ال Flags حيث نريد عمل Broadcast لرسالة المرسلة واخيرا نمرر له ال EndPoint Object وكما يلي:

```
string welcome = "Hello All";
data = Encoding.ASCII.GetBytes(welcome);
newsock.SendTo(data, data.Length, SocketFlags.Broadcast, Remote);
```

يمكن وضع هذا الأكواد في Infinity While Loop بحيث لا تنتهي أو يمكن تحديدها بعدد معين من عمليات الإرسال والاستقبال ..

ثانيا: Socket Classes Members :

1- IPAddress Class : ويستخدم لتعريف IP Address حيث يمكن إسناده إلى ال IPEndPoint كمثال والصيغة العامة له كما يلي:

```
IPAddress newaddress = IPAddress.Parse("192.168.1.1");
```

ويمكن الإختيار بين اربعة خيارات في تحديد العنوان وهي كما يلي :
Any ويستخدم لتمثيل أي عنوان متاح على الشبكة
Broadcast ويستخدم لتمثيل البث الإذاعي لجميع الأجهزة على الشبكة
Loopback ويستخدم لتمثيل العنوان المعروف لل loopback وهو 127.0.0.1
None ويستخدم في حالة عدم وجود Network Interfase في النظام

كما يدعم مجموعة من الميثود وأهمها :

Equals يستخدم هذا الميثود بشكل عام للمقارنة بين tow Objects وهنا سيستخدم للمقارنة بين عنوانين ويرجع True إذا كانا متشابهين و False إذا كانا مختلفين.

GetHashCode وتستخدم لإرجاع العنوان إلى صيغة Hash Code

HostToNetworkOrder ويرجع الجزء الخاص بال Network من العنوان

NetworkToHostOrder ويرجع الجزء الخاص بال Host من العنوان

2- IPEndPoint Class : حيث استخدمناه لتحديد العنوان وال Port لل Host والذي نريد الاتصال به والصيغة العامة له كما يلي :

```
IPEndPoint end = new IPEndPoint(IPAddress.Parse("192.168.1.1"), 5020);
```

مجموعة الخواص التي تدعم في ال Socket Class وهي كما يلي:

AddressFamily ويرجع مجموعة العناوين المعرفة على ال Socket

Available ويرجع حجم البيانات الجاهزة للقراءة من ال Socket

Blocking ويعطي Get أو Set لمعرفة إذا كان ال socket يستخدم ال Blocking Mode أم لا

Connected وتستخدم هذه الخاصية بكثرة لمعرفة إذا كان ال Socket متصل مع ال Remote Host أم لا

Handle ويستخدم لمعرفة نظام التشغيل الذي يتعامل مع ال Socket

ProtocolType ويستخدم لمعرفة البروتوكول الذي يستخدم في ال Socket

RemoteEndPoint ويرجع معلومات عن ال Socket الذي يستخدم مع ال Remote Host

وكمثال لاستخداماتها:

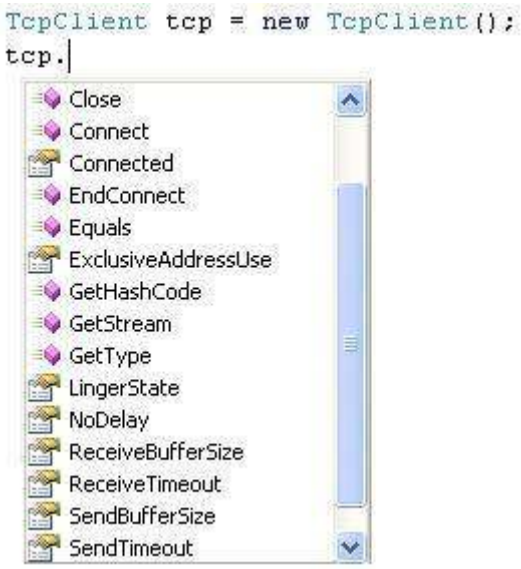
```
using System;
using System.Net;
using System.Net.Sockets;
class Socket_ Properties
{
   public static void Main()
   {
      IPAddress ia = IPAddress.Parse("127.0.0.1");
      IPEndPoint ie = new IPEndPoint(ia, 8000);
      Socket fmo = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
      Console.WriteLine("AddressFamily: {0}",
            fmo.AddressFamily);
      Console.WriteLine("SocketType: {0}",
            fmo.SocketType);
      Console.WriteLine("ProtocolType: {0}",
            fmo.ProtocolType);
      Console.WriteLine("Blocking: {0}", fmo.Blocking);
      fmo.Blocking = false;
      Console.WriteLine("new Blocking: {0}", fmo.Blocking);
      Console.WriteLine("Connected: {0}", fmo.Connected);
      fmo.Bind(ie);
      IPEndPoint iep = (IPEndPoint)fmo.LocalEndPoint;
      Console.WriteLine("Local EndPoint: {0}",
             iep.ToString());
      fmo.Close();
   }

}
```

حيث سترجع المعلومات التالية:

```
AddressFamily: InterNetwork
SocketType: Stream
ProtocolType: Tcp
Blocking: True
new Blocking: False
Connected: False
Local EndPoint: 127.0.0.1:8000
Press any key to continue . . .
```

ثالثا: TCP & UDP Classes Members :

أولا ال Classes الخاصة بال TCP Connection Oriented Protocol :



TcpClient Class-1: حيث تحتوي على مجموعة من ال Methods وال Properties وهي كما يلي:

أولا: أهم الميثود الخاصة بها TCPClient Methods :

Connect: وتستخدم لأجراء عملية الاتصال مع ال server حيث نمرر فيها عنوان ال IP الخاص بال Server و رقم البورت وكما يلي:

```
TcpClient tcp = new TcpClient();
tcp.Connect(IPAddress.Parse("192.168.1.1"),5020);
```

Close: لإنهاء الاتصال مع ال TCP Socket.
EndConnect: لإنهاء Asynchronies Connection حيث ترجع Asynchronies Result.
GetStream: ويستخدم لقراءة ال Stream من ال Socket في عملية الإرسال و الاستقبال.

ثانيا: أهم الخصائص TCPClient Properties :

LingerState : وتأخذ get أو Set لتحديد أو معرفة ال Linger Time
NoDelay: وتأخذ get أو Set لتحديد أو معرفة إذا كان هناك وقت معين لتأخير أم لا
ExclusiveAddressUse: وتأخذ get أو Set لتحديد أو معرفة السوكت يسمح باستخدام ال Client Port أم لا.
SendBufferSize و ReceiveBufferSize: وتأخذ get أو Set لتحديد أو معرفة حجم ال Buffer المستخدم في ال stream والمعرف في TCP Client Object.
SendTimeout و ReceiveTimeout: وتأخذ get أو Set لتحديد أو معرفة الوقت المتاح لعملية الإرسال أو الإستقبال حيث يعطي Time Out في حالة أنه لم يجد الطرف الأخر خلال فترة زمنية معينة.

TcpListener Class-2: حيث تحتوي على مجموعة من ال Methods وال Properties وهي كما يلي:

```
TcpListener tcp_Listener = new TcpListener(IPAddress.Any,5020);
tcp_Listener.
```

أولا: أهم الميثود الخاصة بها TcpListener Methods :

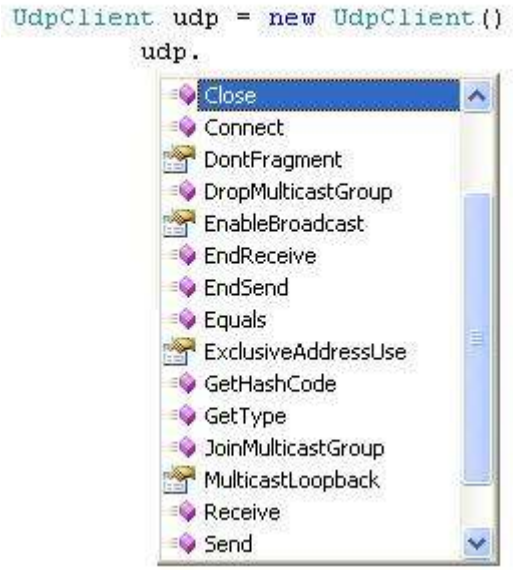
AcceptSocket: وتستخدم لقبول عملية الاتصال مع ال Client.
Start : وهي Overloaded Method حيث انه في حالة تمرير رقم إليها يتم تحديد عدد الأجهزة التي تسمح بوجودها في الطابور أو ال Qouee وبدون تحديد رقم معين يصبح ال Qouee غير محدد.
Stop : وتستخدم لإغلاق عملية التصنت ويفضل وضعها في ال Finally عند استخدام ال Try و ال Catch حتى يتم إنهاء عملية التصنت في حالة حدوث أي Exception.

ثانيا: أهم الخصائص في TcpListener:

LocalEndpoint : حيث يرجع ال IP ورقم البورت المستخدم في ال LocalEndpoint المحدد.
Server: ومن خلالها نستطيع الوصول إلى كل الخصائص و الميثود في ال TCP Server والتي شرحناها سابقا مثل ال Accept وال Sendto وال Receive و Listen وغيرها

ثانيا ال Classes الخاصة بال UDP Connectionless Protocol :

UdpClient Class-1: وتستخدم لتعريف UDP Datagram Protocol Connection قمنا سابقا بتعريفها والتعامل معها وفي هذا الدرس سنبين أهم محتوياتها وهي كما يلي :



ومن أهم الميثود والخصائص الخاصة بها :

JoinMulticastGroup و DropMulticastGroup: لضم أو إلغاء عنوان أو مجموعة من العناوين من ال Multicast Group.
EnableBroadcast: وتأخذ Get أو Set لتفعيل ال Broadcasting في ال socket.
MulticastLoopback: وتأخذ Get أو Set لمعرفة أو تحديد ال Mlticast Loopback.

MulticastOption Class-2: ويستخدم في ال Multicasting حيث يتم تخزين IP Address List لتعامل معها في Multicast Group لعمل Join و Drop لأي Multicast Group وتستخدم كما يلي كمثال لإضافة عضوية لاستقبال رسائل Multicast :

أولا نعرف ال UDP Socket وكما يلي :

```
mcastSocket = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
```

ثانيا نقوم بتعريف Address List ثم نسند إليها ال IP الذي نريد إدخاله في ال Group أو نجعل ال User يدخل العنوان بنفسه نربطها بالسكوت باستخدام الميثود Bind وكما يلي :

```
IPAddress localIPAddr = IPAddress.Parse(Console.ReadLine());
mcastSocket.Bind(IPlocal);
```

ثالثا نقوم بتعريف ال Multicast Option ونسند لها العنوان المحدد كما يلي:

```
MulticastOption mcastOption;
mcastOption = new MulticastOption(localIPAddr);
```

ومن ثم نضيف التغير على ال حيث تأخذ هذه الميثود ثلاثة بارومیترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP وفي حالتنا هذه سوف نستخدم التغير على IP إذ ما نريده هو ضم IP إلى Multicast Group وفي البارومیتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه ال MulticastOption Object والذي قمنا بإنشائه و كما يلي:

```
mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership,mcastOption);
```

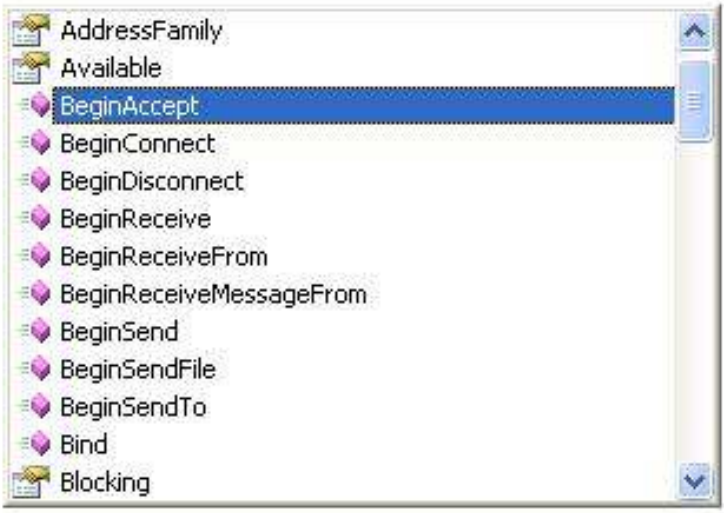
الدرس الثاني عشر: Asynchronous Sockets Programming :

سوف نتحدث في هذا الدرس عن استخدام ال Asynchronous Socket بشكل أكثر تفصيلا عما تحدثنا به سابقا وسوف نطبق مجموعة من الأمثلة العملية على استخدام الاتصال الغير متزامن في برمجيات الشبكات ...
من المعروف أن الاتصال المتزامن مهم جدا في البرمجيات التي تحتاج إلى العمل في الزمن الحقيقي حيث لا يسمح باستخدام الاتصال لأمر آخر إلى بعد انتهاء العملية الجارية واستخدامه مهم جدا في العمليات التي تتطلب مثل هذه الأمور لا كن لا ينصح أبدا استخدامه في حالة إذا كانت الجهة المستقبلة للبيانات تستخدم Slow Connection كاعتماد الشبكة على ال Dialup لربط الجهازين المرسل مع المستقبل أو في حالة إذا كان هنالك مجموعة كبيرة من المستخدمين تستخدم السيرفر حيث يمنع الأسلوب المتزامن بقية المستخدمين على الشبكة من إجراء عملية الإرسال في حالة كون ال Server يستقبل بيانات من جهاز آخر ، وفي هذه الحالة ينصح باستخدام الاتصال الغير المتزامن إذ يعتبر مهم جدا في حالة إذا أردنا من البرنامج القيام بعدة مهام وعلى نفس ال Thread وباستخدام نفس ال Connection ، أو كما ذكرنا سابقا في حالة إذا كان الاتصال بطيء نسبيا أو انه يوجد عدد مستخدمين يستخدمون نفس ال Server ..

أولا Asynchronous Socket Class and its members :

تدعم الدوت نيت الاتصال غير المتزامن بمجموعة من ال methods الموجودة ضمن ال Socket Class والتي يتم استدعائها من ال System.Net.Socket Namespaces وقد ميزت الدوت نيت هذه الميثودس بوجود ال Begin في بداية أسم الميثود، ولكل Begin Method يوجد End Method مقابلة لها والتي تستخدم لإرجاع callback result عند انتهاء ال Begin Method من التنفيذ وهي كما يلي:

```
Socket MySocket = new Socket(AddressFamily
MySocket.b
```



1- BeginAccept و تستخدم لقبول ال Client Request وإسناده إلى ال Object AsyncCallback وباستخدام هذه الطريقة سوف يتمكن ال Server من استقبال عدد من ال Clients Requests في نفس الوقت وبدون الحاجة لانتظار الانتهاء من العملية الجارية حيث يتم في كل مرة استدعاء الميثود باستخدام ال AsyncCallback Delegate وتستخدم كما يلي كما يلي:

```
m_mainSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream,ProtocolType.Tcp);
IPEndPoint ipLocal = new IPEndPoint (IPAddress.Any, 5020);

m_mainSocket.Bind (ipLocal);
m_mainSocket.Listen (10);
m_mainSocket.BeginAccept (new AsyncCallback (Client_request_method),
null);
```

حيث سيتم إضافة ال Client Request في Callback Reference منفصل عن السابق وهنا لابد من إنشاء method لاستقبال ال Client Request وإنهاء ال Client Accepted Object باستخدام الميثود EndAccept :

```
public void Client_request_method(IAsyncResult ar)
   {
Socket listener = (Socket)ar.AsyncState;
Myclient = listener.EndAccept(ar);
Myclient.Send(/* data to be send*/ );
listener.BeginAccept(new AsyncCallback(Client_request_method), listener);
Console.WriteLine("Socket connected to {0}",
client.RemoteEndPoint.ToString());    }
```

<u>في Dot Net 2005 أصبحت الBeginAccept Method تأخذ عدة أشكال كما يلي:</u>

الشكل الأول في الدوت نيت 2003 و 2005 وتأخذ AsyncCallBack Delegate و State Object لإرجاع معلومات عن حالة ال Request في ال Socket وكما يلي:

```
MySocket.BeginAccept(AsyncCallback , object state)
```

الشكل الثاني في الدوت نيت 2005 حيث يمكنك فيه تحديد حجم البيانات المستلمة

```
MySocket.BeginAccept(int Data_ Receive_Size , AsyncCallback , object state)
```

الشكل الثالث في الدوت نيت 2005 حيث بمكن فيه تحديد ال Accepted Socket

```
MySocket.BeginAccept(Socket accept_Socket ,int Data_ Receive_Size ,
AsyncCallback , object state)
```

2- BeginConnect وتستخدم لبدأ Asynchronous Connection على ال Socket ورقم البورت المحدد حيث يسند لها ال IPEndPoint وال Asynchronous Callback وال State Object وكما يلي:

```
MySocket.BeginConnect(EndPoint IP,Syncallback Result,object state)
```

وتستخدم كما يلي كمثال:

```
Socket MySocket = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint ipend = new IPEndPoint(IPAddress.Parse("192.168.1.101"),
5020);

 MySocket.BeginConnect(ipend, new AsyncCallback(Connected), MySocket);
```

في ال Connected Method يتم تحديد ال CallBack Socket كما يلي:

```
public static void Connected(IAsyncResult iar)
{
    Socket sock = (Socket)iar.AsyncState;
    try
    {
        sock.EndConnect(iar);
    }
    catch (SocketException)
    {
        Console.WriteLine("Unable to connect to host");
    }}
```

3- BeginReceive وتستخدم لإستقبال بيانات من ال Client وتخزينها في Byte Array والصيغة العامة لها كما يلي:

```
MySocket.BeginReceive(Byte[] buffer,int offset, SocketFlags,AsyncCallback,
object sate)
```

ويستخدم كما يلي كمثال:

```
byte[] data = new byte[1024];
 MySocket.BeginReceive(data, 0, data.Length, SocketFlags.None, new
AsyncCallback(ReceivedData), MySocket);

void ReceivedData(IAsyncResult iar)
    {
  Socket remote = (Socket)iar.AsyncState;
  int recv = remote.EndReceive(iar);
  string receivedData = Encoding.ASCII.GetString(data, 0, recv);
  Console.WriteLine(receivedData);
    }
```

كما تستخدم الميثود BeginReceiveFrom لإستقبال البيانات من موقع محدد باستخدام ال UDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

4- BeginSend وتستخدم لإرسال بيانات إلى الطرف المستقبل عبر ال Asynchronous Socket والصيغة العامة لها كما يلي:

```
MySocket.BeginSend (Byte[] buffer,int offset, SocketFlags,AsyncCallback,
object sate)
```

وتستخدم كما يلي كمثال:

```
private static void SendData(IAsyncResult iar)
 {
        Socket server = (Socket)iar.AsyncState;
        int sent = server.EndSend(iar);
  }


byte[] data = Encoding.ASCII.GetBytes("Hello Word");
MySocket.BeginSend(data, 0, data.Length, SocketFlags.None,
new AsyncCallback(SendData), MySocket);
```

كما تستخدم الميثود BeginSendto لإرسال البيانات إلى Remote Host محدد باستخدام ال UDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

5- كما تم إضافة مجموعة من الميثود الجديدة في الدوت نيت 2005 وهي:
BegonDiconnect لإنهاء الاتصال و **BeginSendFile** لإرسال ملف و ال **BeginReceiveMessageFrom** والتي تستخدم لإستقبال عدد محدد من البيانات وتخزينها في مكان محدد في ال Bufer ..

تأخذ ال**BeginSendFile** التركيب التالي:

```
MySocket.BeginSendFile(string filename,AsyncCallback Asyn,object state)
```

وال **BeginReceiveMessageFrom** التركيب التالي:

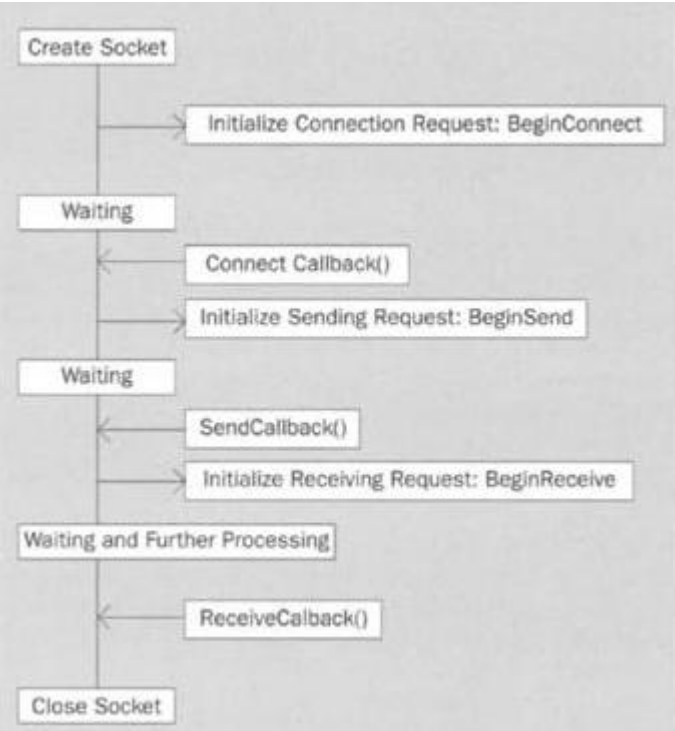
```
MySocket.BeginReceiveMessageFrom(byte Buffer ,int offset,int
size,SocketFlags sf,ref EndPoint,AsyncCallback ascb,object state)
```

وال **BegonDiconnect** التركيب التالي:

```
MySocket.BeginDisconnect(bool reuseSocket,AsyncCallback ascb,object state)
```

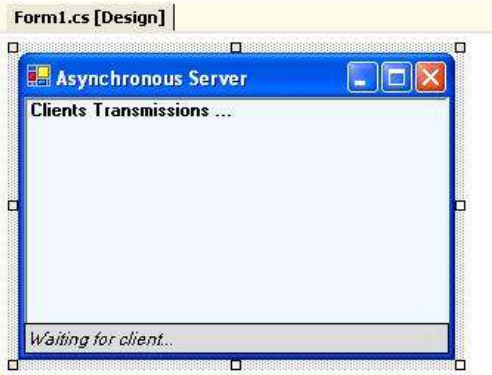
ثانيا: تطبيقات ال Asynchronous Socket في الدوت نت :

تمر عملية الاتصال الغير متزامن بمجموعة من المراحل تبدأ بإنشاء الSocket Object في ال Server Side بعد ذلك يتم تعريف ال BeginConnect لبدأ Asynchronous Connection على ال Socket حيث يتم إسناد IPEndPoint Object والMethod Asynchronous Callback وال State Object لها وتبدأ في هذه الحالة عملية الاتصال بال Socket ، وبعد ذلك تمرر إلى ال BeginAccept لقبول ال Client Request حيث يتم قبول الطلب ويرسل Acknowledgement إلى ال Client ليعلمه فيها بقبول الجلسة وإمكانية البدء للإرسال و يستطيع ال Client بعد الموافقة على الجلسة البدء بالإرسال باستخدام الميثود BeginSend ويستقبل ال Server الرسالة من ال Client باستخدام الميثود BeginReceive وكما ذكرنا سابقا فإن لكل عملية Begin تقابلها الميثود End للاستعداد لإجراء عملية أخرى على نفس ال Thread في البرنامج وهو ما ميز الاتصال الغير متزامن عن الاتصال المتزامن.



وبناء على المفاهيم السابقة سوف نقوم الآن بإنشاء برنامج Client/Server Chatting يعتمد على ال Asynchronous Socket لإرسال واستقبال البيانات .

<u>وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:</u>



سوف نستخدم ال Namespaces التالية:

```
using System.Net;
using System.Net.Sockets;
using System.Text;
```

في ال Global Declaration (أي بعد تعريف ال Main Class)قم بإضافة التعاريف التالية:

```
public class Form1 : System.Windows.Forms.Form
{
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 5020);
private byte[] data = new byte[1024];
private int size = 1024;
```

في ال Form Load قم بإضافة الكود التالي حيث سنعرف Connection يعتمد على ال TCP ويعمل على البورت 5020 ثم تعريف عملية قبول الاتصال باستخدام ال BeginAccept :

```
private void Form1_Load(object sender, System.EventArgs e)
{
server = new Socket(AddressFamily.InterNetwork,SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 5020);
server.Bind(iep);
server.Listen(5);
server.BeginAccept(new AsyncCallback(AcceptConn), server);
}
```

ثم إنشاء Accept Callback Method والذي سيتم فيه إنهاء ال Accepted Request باستخدام ال EndAccept Method وبعد ذلك إرسال Acknowledgement إلى ال Client تخبره فيها بقبول الطلب وترسل باستخدام ال BeginSend Method كما يلي:

```
void AcceptConn(IAsyncResult iar)
{
Socket oldserver = (Socket)iar.AsyncState;
Socket client = oldserver.EndAccept(iar);
conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
string stringData = "Welcome to my server";
byte[] message1 = Encoding.ASCII.GetBytes(stringData);
client.BeginSend(message1, 0, message1.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
}
```

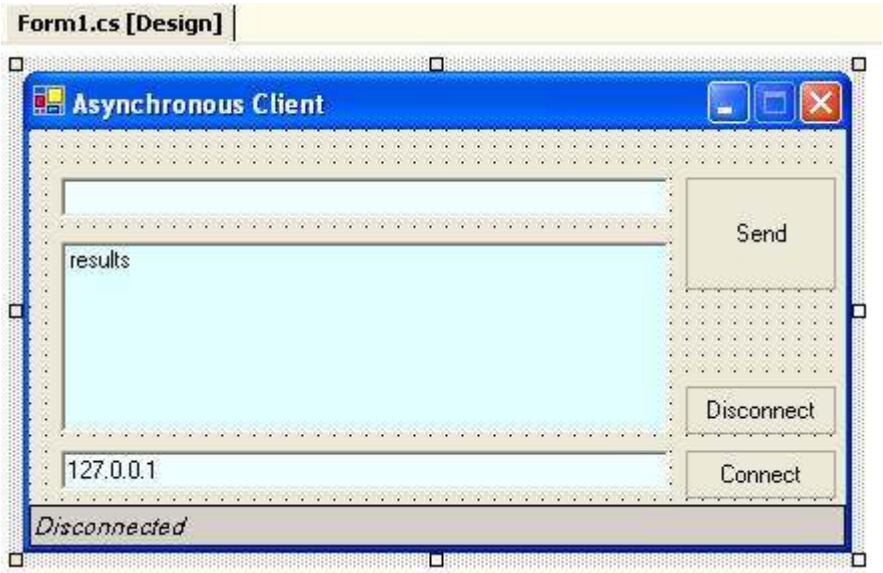
ثم إنشاء Send Callback method لإنهاء ال BeginSend وكما يلي:

```
void SendData(IAsyncResult iar)
{
Socket client = (Socket)iar.AsyncState;
int sent = client.EndSend(iar);
client.BeginReceive(data, 0, size, SocketFlags.None,new
AsyncCallback(ReceiveData), client);
}
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

```
void ReceiveData(IAsyncResult iar)
{
Socket client = (Socket)iar.AsyncState;
int recv = client.EndReceive(iar);
        if (recv == 0)
        {
                client.Close();
                conStatus.Text = "Waiting for client...";
                server.BeginAccept(new AsyncCallback(AcceptConn), server);
                return;
        }
string receivedData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(receivedData);
byte[] message2 = Encoding.ASCII.GetBytes(receivedData);
client.BeginSend(message2, 0, message2.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
}
```

وهنا قد تم الانتهاء من برنامج ال Server والآن سوف نقوم بإنشاء برنامج ال Client وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم ال Namespaces التالية:

```
using System.Net;
using System.Net.Sockets;
using System.Text;
```

في ال Global Declaration (أي بعد تعريف ال Main Class)قم بإضافة التعاريف التالية:

```
public class Form1 : System.Windows.Forms.Form
{
        private Socket client;
        private byte[] data = new byte[1024];
        private int size = 1024;
```

في ال Connect Button قم بكتابة الكود التالي:

```
conStatus.Text = "Connecting...";
Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(textBox1.Text), 5020);
newsock.BeginConnect(iep, new AsyncCallback(Connected), newsock);
```

ثم قم بإنشاء Callback Connect method كما يلي:

```
void Connected(IAsyncResult iar)
{
client = (Socket)iar.AsyncState;
        try
        {
client.EndConnect(iar);
conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
client.BeginReceive(data, 0, size, SocketFlags.None, new
AsyncCallback(ReceiveData), client);
        }
catch (SocketException)
                {
        conStatus.Text = "Error connecting";
                }
}
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

```
void ReceiveData(IAsyncResult iar)
{
Socket remote = (Socket)iar.AsyncState;
int recv = remote.EndReceive(iar);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(stringData);
}
```

ثم إضافة الكود التالي في ال Send Button :

```
try
{
byte[] message = Encoding.ASCII.GetBytes(newText.Text);
newText.Clear();
client.BeginSend(message, 0, message.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
newText.Focus();
}
catch(Exception ex){MessageBox.Show(ex.Message);}
```

ثم إنشاء Send Callback method لإنهاء ال BeginSend وكما يلي:

```
void SendData(IAsyncResult iar)
{
        try
        {
        Socket remote = (Socket)iar.AsyncState;
        int sent = remote.EndSend(iar);
remote.BeginReceive(data, 0, size, SocketFlags.None, new
AsyncCallback(ReceiveData), remote);
                        }
catch(Exception ex){MessageBox.Show(ex.Message);}

}
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

```
void ReceiveData(IAsyncResult iar)
{
        try
        {
Socket remote = (Socket)iar.AsyncState;
int recv = remote.EndReceive(iar);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(stringData);
        }
        catch(Exception ex){MessageBox.Show(ex.Message);}
}
```

وكما لاحظنا فإن برنامج ال Client لا يختلف كثيرا عن برنامج ال Server حيث نعرف في ال Server ال Socket Connection وال BeginAccept Method أما في ال Client فنعرف ال Socket Connection و ال BeginConnect Method وتبقى عملية الإرسال والاستقبال هي نفسها في Server وال Client ...

End of Chapter 2

وهنا قد تم الانتهاء من الجزء الثاني وفي الجزء الثالث من هذه السلسلة سوف نتحدث ال Multicasting بشكل أكثر تفصيلا كما سوف نتحدث Network Security Programming والتي تضم الCryptography وال Authentication Protocols ...

Copyrighted to: Mr. FADI – Abdel-qader..
Fadi822000@yahoo.com
http://spaces.msn.com/members/csharp2005