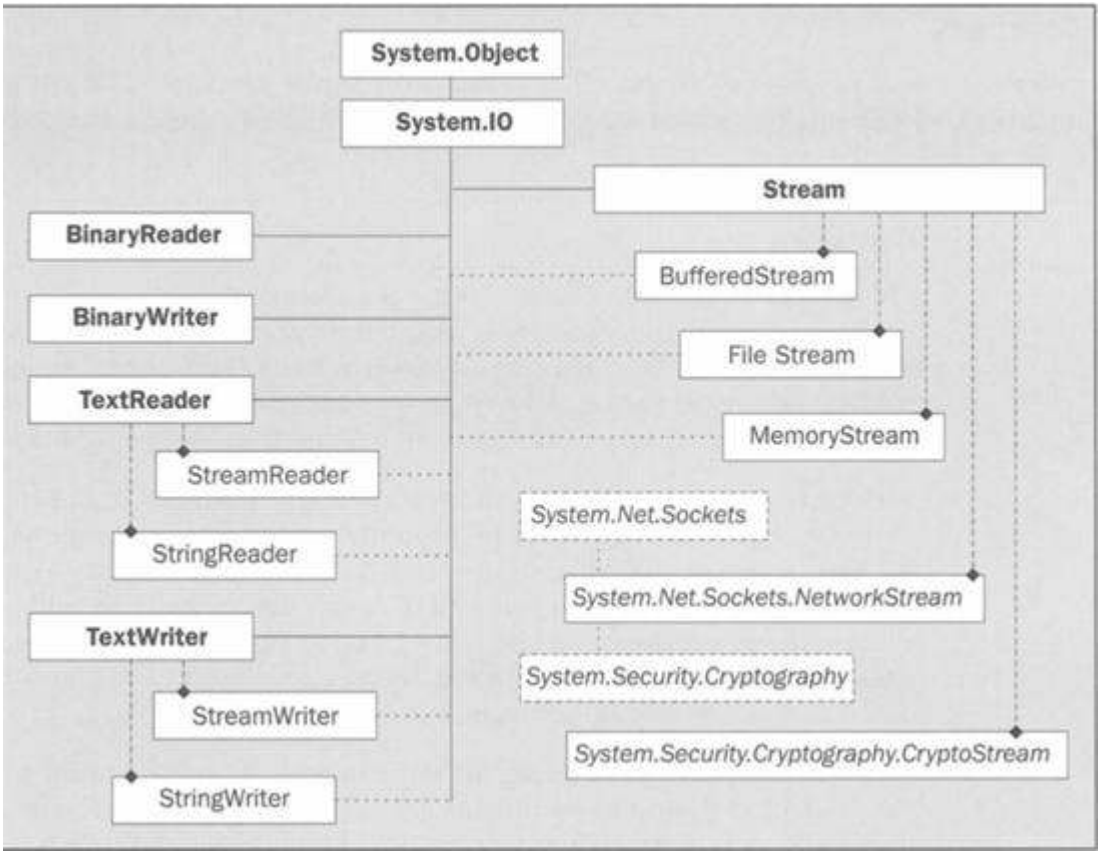
بسم الله الرحمن الرحيم

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 Stream Reader & Stream 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 سبيسس System.object و System.IO والتي تستخدم لعمليات نقل البيانات ال Binary Data . 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 MemoryStream عبر ال readBufStream ثم ارسلناه إلى الميثود 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- الميثودس 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- الميثودس غير المتزامنة Asynchronous Methods :

I. BeginRead وال BeginWrite وتستخدم لعملية القراءة أو الكتابة باستخدام ال Stream الغير المتزامن وتأخذ خمسة باروميترات 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 الميثود والتي تستخدم لإدارة ال Stream وهي :

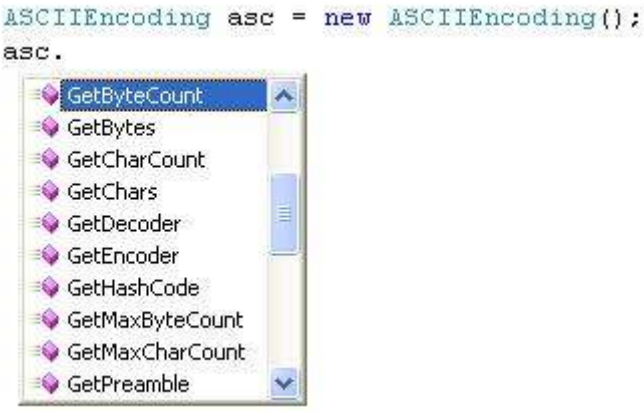
1- Flush وتستخدم لتفريغ محتويات ال Buffer بعد إتمام العملية المحددة حيث يتم نقل محتويات ال Buffer إلى ال Destination الذي تم تحديده في Stream Object.

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 :



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);
```

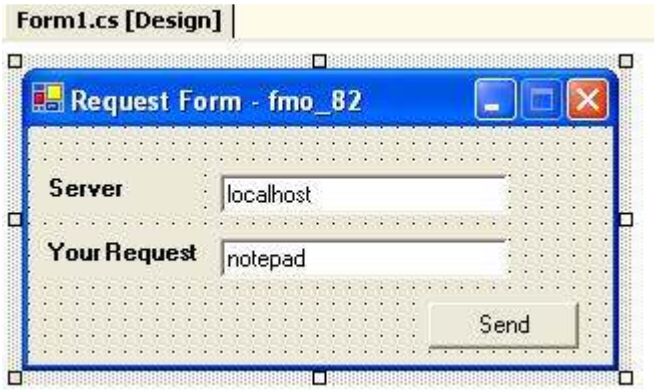
BinaryWriterr لإرسال BinaryData إلى ال Stream عبر ال Socket:

```
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 اكثر تقدما في برمجة TCP وال UDP ..