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

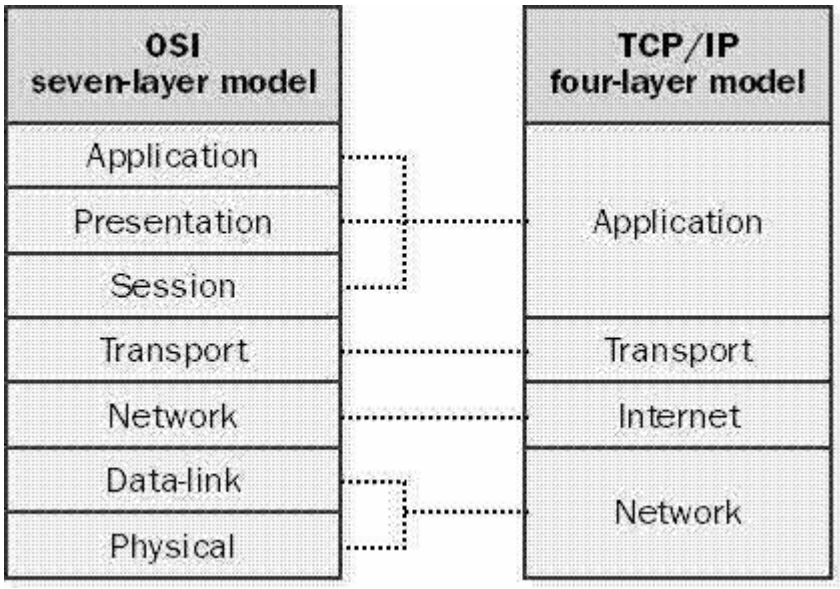
Chapter 1
An Overview on Network & TCP/IP Programming

1- Introduction to Network and TCP/IP Programming
   a. Connection Oriented Via TCP
   b. Connection Less Via UDP
   c. Using Streams to Send & Receive Binary Data
   d. IP Multicasting
   e. Application Layer Programming
      i. DNS Programming
      ii. SMTP Programming
      iii. POP3 Programming
      iv. HTTP Programming
      v. Web Services & XML Programming
      vi. FTP Programming

بسم الله الرحمن الرحيم

الدرس الأول : مقدمة في برمجة الشبكات و بروتوكول TCP/IP

من المعروف أن الشبكة هي مجموعة من الأجهزة متصلة مع بعضها عبر وسيلة اتصال معينة ومن هنا سيندرج لدينا التقسيم المعروف لمنظمة OSIلعملية الاتصال والتي تمر بسبعة طبقات لكل طبقة منها وظيفة معينة وتم اختصارها إلى أربعة طبقات(خمسة في بعض الكتب) في برتوكول TCP/IP وتبين الصورة المرفقة هذه الطبقات:



لإجراء عملية الاتصال بين Client و Server يلزم ما يلي :

في الجهاز المرسل Client :

1- تبدأ عملية توليف الرسالة المرسلة في ال Application Layer ووظيفتها هنا التعامل مع الرسالة نفسها وتحويلها من صيغة نصية إلى Data يمكن إرسالها عبر الشبكة ، ففي برمجيات الدردشة Chat يتم تحويل النص المكتوب إلى ASCII Code ثم يتم تحويل هذا الأسكي إلى مجموعة من Binary Code توضع في مصفوفة لتجهيزها وإرسالها عبر Socket (بقية الطبقات) وهنا توضيح هذه الخطوة في الكود

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

في نموذج OSI يتم تقسيم ال upper Layers إلى ثلاثة طبقات

Application لتعامل مع البرنامج نفسه أو ما يسمى interface user

Presentation تمثيل البيانات المرسلة وهي كما ظهرت سابقا بتحويل البيانات إلى

الASCII .

Session وفيها البدء بعملية التخاطب بين الجهازين و التعريف ببعضهم البعض(فتح الجلسة) والتي سآتي على شرحها بعد قليل

أما في بروتوكول ال TCP/IP فكتفا بوجود طبقة Application والتي تقوم بعمل الطبقات الثلاث الأولى في OSI ، في session Layer يتم التعرف وفتح الجلسة بعدة خطوات وهي كما يلي :

1- إجراء الإيصال المبدئي بجهاز server عبر ال IP و البورت المحدد وذلك بعد تحديد عملية الاتصال سواء عبر UDP أو عبر TCP

2- التعريف بنفسه وعمل ال Authentication إذا تطلب جهاز السيرفر ذلك

3- قبول أو رفض الجلسة ويتم ذلك بإرسال الموافقة على فتح الجلسة أو رفضها

4- بدأ الجلسة وقيام السيرفر بعمل Listening على البورت الخاص بالبرنامج

عندما يتم الموافقة على فتح الجلسة والبدء بعملية التخاطب يقوم جهاز المرسل Client بتحميل الرسالة إلى الطبقة الأخرى وهي هنا طبقة Transport وفي هذه الطبقة يتم تحديد طبيعة الاتصال سواء عبر TCP - Connection Protocol أو عبر ال UDP - Connectionless Protocol ففي البروتوكول الأول يتم تحديد طرفين وهما المرسل والمستقبل وبورت الاتصال أما ال UDP فيتم تحديد الطرف المرسل و المستقبل (اختياري) أي انه يمكن عمل ال Broadcast بدون تحديد جهة معينة لاستقبال الرسالة أي أن أي شخص يقوم بتصنت عبر هذا البورت Listening يستطيع استقبال الرسالة ، وهنا مثال يوضح عمل هذه الطبقة

```
TcpClient tcpclnt = new TcpClient();
tcpclnt.Connect("192.168.0.2",8001);
```

ولإرسال الرسالة عبر الشبكة نستخدم في الدوت نت كلاس جاهز يقوم بهذه العملية ويسمى Network Stream وهو المسئول عن التعامل مع وسيلة الاتصال وإرسال الرسالة إلى الطرف المعني بشكل Stream Data أو باستخدام ال Socket نفسه وكمثال على ذلك:

```
NetworkStream mynetsream = tcpclnt.GetStream ();
StreamWriter myswrite = new StreamWriter (mynetsream);
myswrite.WriteLine("Your Message");
```

وبعد ذلك تسلم إلى Network Layer وهي مكون من Datalink Layer و Network Layer في OSI طبعا يتولى نظام التشغيل و بروتوكول TCP/IP إرسال الرسالة عبر الشبكة

وهنا ملخص عمل كل من Network Layer و Data Link Layer

Network layer :

Layer 3 of the Open Systems Interconnection (OSI) reference model for networking. The network layer is responsible for functions such as the following:

- Logical addressing and routing of packets over the network
- Establishing and releasing connections and paths between two nodes on a network
- Transferring data, generating and confirming receipts, and resetting connections

Example about network layer : IP-internet protocol , ICMP -internet control message protocol , Routing

Data Link layer: The link layer provides physically means Example : ARP Address Resolution Protocol , RARP Reverse Address Resolution Protocol.

وبهذا قمت بشرح كيفية الاتصال عبر Layers.

أما بنسبة للجهاز المستقبل Server يقوم بالمرور على نفس الطبقات ولكن بالعكس حيث يستلم كرت الشبكة ال Data Packets لتحول إلى Data link ثم Network ثم Transport ثم Application ومنها تحول من Binary إلى ASCII ومن ASCII إلى Text .. وهذه الكود يوضح مبدأ عمل ال Server

```
TcpListener myList=new TcpListener("127.0.0.1",8001);
myList.Start();
Socket s=myList.AcceptSocket();
byte[] b=new byte[100];
int k=s.Receive(b);
for (int i=0;i<k;i++)
Console.Write(Convert.ToChar(b[i]));
s.Close();
```

الدرس الثاني : Connectionless Sockets Via UDP

تحدثنا سابقا عن ال TCP – Connection Oriented Protocol وقلنا أن بروتوكول ال TCP هو بروتوكول موجه وهذا يعني انه يلزم احتواء ال Header الخاص به على عنوان المرسل و عنوان المستقبل كما يلزم أيضا القيام بعمليات التحقق Authentication و يدعم عمليات التحقق من الوصول و التسليم بشكل الصحيح .... لكن ماذا لو كان كل ذلك غير مهم بنسبة لك إذ تريد من برنامجك أن يقوم بعملية بث إذاعي Broadcast لرسالتك ولا يهمك من سوف يستلم الرسالة و أن السرعة في الإرسال و الاستقبال هي الهدف الأساسي إذا وجب عليك ترك بروتوكول ال TCP والتوجه نحو ال UDP User Datagram Protocol ويسمى أيضا بال Connectionless Protocol في هذا البروتوكول تستطيع عمل ما يسمى بال Broadcast و ال Multicast (Broad-يعني الإرسال إلى الكل و Multi-يعني الإرسال إلى مجموعة اثنان أو أكثر واليوني-يعني الإرسال لواحد فقط) يوجد شرط وحيد يلزم أن تأخذه بعين الاعتبار عند استخدام ال UDP لعملية البث باستخدام Broadcast وهو أن الشبكة التي تريد عمل بث لها تتصل معها بشكل مباشر Direct Connection أي بدون وجود Router بينك وبين المستقبل إذ أن ال Router يمنع عمليات البث الإذاعي Broadcast حيث يلزم أن تكون الشبكة ضمن ال Range Class سواء A او B او C

لاستخدام ال UDP يلزم أولا تعريف النيم سبيس System.Net و ال System.Net.Socket لاحظ انه في ال TCP كان يلزم تعريف رقم البورت والعنوان للجهاز المستقبل أما في ال UDP فتستطيع تعريفه كما هو في TCP كما وتستطيع عمل Broadcast باستخدام IPAddress.Any بعد اشتقاق كائن من الكلاس IPEndPoint (وتعني نقطة الهدف) وتستطيع أيضا عدم تحديد رقم البورت وذلك باستخدام الميثود Bind() حيث يتم تعريفها بنقطة الهدف ب 0 ويتم كل ذلك كما يلي كمثال :

التالي هو الجزء الخاص بالسيرفر ووظيفته فتح البورت 9050والتنصت عليها ثم استلام الرسالة عبر هذا البورت وتوزيعها على الكل بدون تحديد رقم بورت معين حيث يتم تسليمها على البورت المخصص لعملية البرود كاست وهو البورت صفر:

لتعريف نقطة الهدف ورقم البورت الخاصة بسير فر ونستخدمها لكي يقوم السيرفر باستلام الرسالة

```
IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
```

لتحديد نوع البرتوكول المستخدم يتم ذلك كما يلي

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

ثم إعطاء نقطة الهدف ورقم البورت إلى الميثود Send ويستخدم هذا الميثود عند الاستقبال فقط

```
newsock.Bind(ipep);
```

الآن تم استقبال الرسالة ونريد بثها إلى كل من يتصل مع السيرفر على البورت السابقة ولعمل ذلك يلزم أولا تعريف نقطة الهدف كما يلي

```
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)(sender);
```

لاحظ أن عنوان نقطة الهدف هو Any ورقم البورت صفر وهذا يعني إرسال الرسالة المستلمة إلى الكل وبما فيهم الشخص مرسل الرسالة و السيرفر

هنا يتم استلام الرسالة من السيرفر إلى السيرفر مرة أخرى عبر الشبكة

```
recv = newsock.ReceiveFrom(data, ref Remote)
```

لطباعة عنوان مرسل الرسالة و الرسالة نفسها

```
Console.WriteLine("Message received from {0}:", Remote.ToString());
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

نقوم هنا بإرسال رسالة ترحيبية لكل جهاز جديد يشبك على السيرفر نخبره بها انه تم الموافقة على دخوله ضمن الأجهزة طبعا هذه رسالة اختيارية تستطيع حذفها اذا كنت لا تريدها

```
string welcome = "Welcome Customer ...";
data = Encoding.ASCII.GetBytes(welcome);
newsock.SendTo(data, data.Length, SocketFlags.None, Remote);
```

هنا Loop لا نهائي الهدف منه هو عند استقبال أي رسالة في أي وقت من قبل أي جهاز يقوم السيرفر باستلامها وتسليمها إلى كل من هو على الشبكة ... إذا أردت تحديد عدد معين من الرسائل المستلمة تستطيع تغيير True إلى أي رقم تريده

```
while(true)
 {
   data = new byte[1024];
   recv = newsock.ReceiveFrom(data, ref Remote);
```

```
    Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    newsock.SendTo(data, recv, SocketFlags.None, Remote);
  }
server.Close();
```

هنا يتم إغلاق السوكت في حالة إذا تم الخروج من Loop ألا نهائي طبعا هنا لن يتم الوصول إلى هذه النقطة إلا إذا وضعنا كلمة بريك داخل الوب وفق شرط معين وهنا نستطيع وضع جملة شرطية انه في حالة استقبال رسالة أو نص رسالة معينة اخرج من Loop وقم بإغلاق السوكت وهذا يعني انك تستطيع عمل تحكم عن بعد للإغلاق السيرفر كما يمكنك وضع جملة تشغيل أي ملف تنفيذي على السيرفر في حالة ورود نص معين وهكذا .

تم الإنهاء الآن من شرح الجزء الخاص سيرفر .. وسأعرض هنا الكود الخاص به بشكل كامل لكي تستطيع تطبيقه وهذا هو الكود:

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

سوف نستخدم System.Text نيم سبيسس لتعامل مع ال ASCII Class

```
using System.Text;
class SimpleUdpSrvr
{
 public static void Main()
 {
  int recv;
  byte[] data = new byte[1024];
  IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
  Socket newsock = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
  newsock.Bind(ipep);
  Console.WriteLine("Waiting for a client...");
  IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
  EndPoint Remote = (EndPoint)(sender);
  recv = newsock.ReceiveFrom(data, ref Remote);
  Console.WriteLine("Message received from {0}:", Remote.ToString());
  Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

```
  string welcome = " Welcome Customer ...";
  data = Encoding.ASCII.GetBytes(welcome);
  newsock.SendTo(data, data.Length, SocketFlags.None, Remote);
  while(true)
  {
    data = new byte[1024];
    recv = newsock.ReceiveFrom(data, ref Remote);

    Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    newsock.SendTo(data, recv, SocketFlags.None, Remote);
  }
 }
}
```

الآن الجزء الخاص بال Client ، يقتصر العمل هنا على قيام ال Client بفتح جلسة مع السيرفر وذلك بعد تعريفه بنقطة الاستلام ورقم البورت وكما تم في السابق إلا أن الاختلاف هو في الوظيفة إذا يقتصر فقط على استقبال الرسالة من

السيرفر وإرسال أي رساله له عبر البورت المخصص للقيام بهذه العملية انظر الكود التالي :

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SimpleUdpClient
{
public static void Main()
{
 byte[] data = new byte[1024]; string input, stringData;
 IPEndPoint ipep = new IPEndPoint(
  IPAddress.Parse("127.0.0.1"), 9050);
 Socket server = new Socket(AddressFamily.InterNetwork,
  SocketType.Dgram, ProtocolType.Udp);
//في حالة فقدان الإتصال مع السيرفر يظهر الرسالة التاية
  string welcome = "Hello, are you there?";
 data = Encoding.ASCII.GetBytes(welcome);
 server.SendTo(data, data.Length, SocketFlags.None, ipep);
```

```
 IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
 EndPoint Remote = (EndPoint)sender;

data = new byte[1024];
 int recv = server.ReceiveFrom(data, ref Remote);
 Console.WriteLine("Message received from {0}:", Remote.ToString());
 Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
هذا Loop لكي تستطيع إرسال عدد غير محدد من الرسائل
while(true)
 {
  input = Console.ReadLine();
 Exit  في حالة إذا أردت إنهاء الجلسة اكتب
if (input == "exit")
   break;
  server.SendTo(Encoding.ASCII.GetBytes(input), Remote);
  data = new byte[1024];
  recv = server.ReceiveFrom(data, ref Remote);
  stringData = Encoding.ASCII.GetString(data, 0, recv);
  Console.WriteLine(stringData);
 }
 Console.WriteLine("Stopping client");
 server.Close();
}
}
```

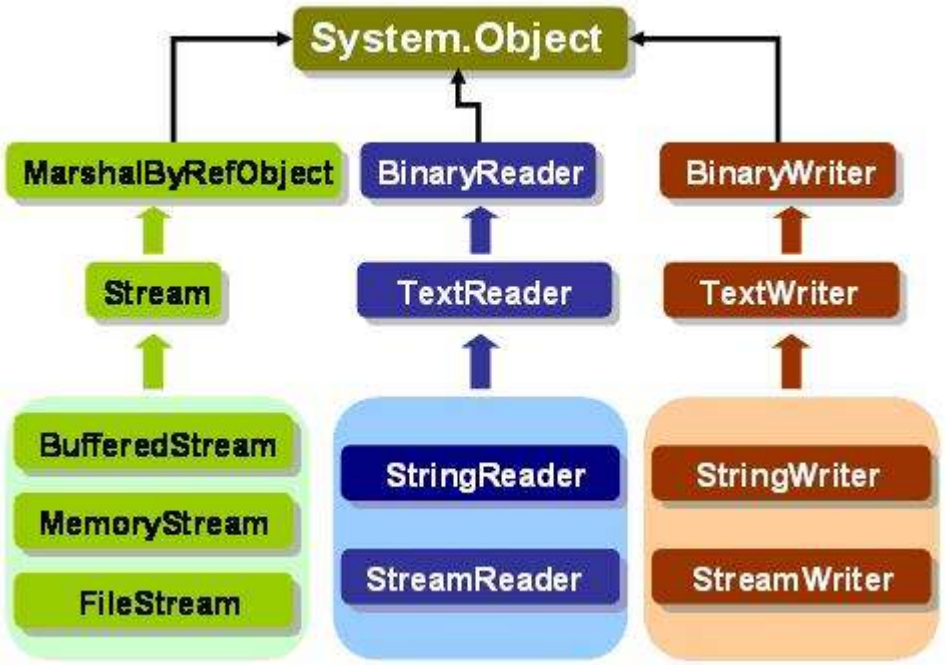
بينا في هذا الدرس كيفية التعامل مع ال UDP Connectionless Protocol وبينا الفرق بينه وبين TCP Connection Oriented Protocol ولاحقا إنشاء الله سوف نشرح بشكل أكثر تفصيلا عن ال Network Layer Programming ..

الدرس الثالث : Using Streams to Send & Receive Binary Data

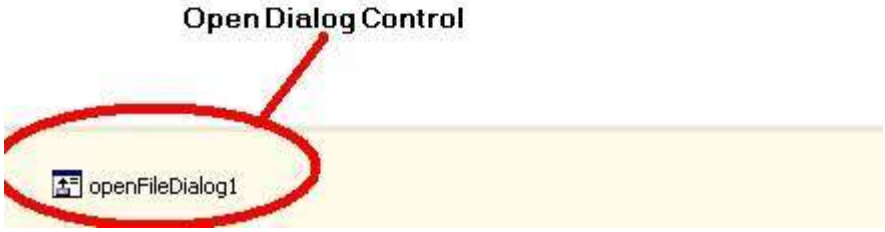
قمنا سابقا بتعرف على أجزاء OSI و TCP/IP وبينا كيفية التعامل مع هذه الطبقات في البرنامج ، وفي هذا الدرس سوف نبين كيفية التعامل مع ال Stream Library لإرسال Binary Data بالإضافة إلى استخدام ال Thread في برمجيات الشبكة..

أولا : ال Socket قلنا سابقا 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 نأتي على شرح الكثير منها لاحقا..

يمكنك الStream Classes من نقل Text أو Object ،حيث بينا سابقا كيفية التعامل من السوكت لنقل Text باستخدام Stream Reader وال Stream Writer وفي هذا الدرس سنبين كيفية التعامل معه لنقل Object (أي نوع أخر من البيانات ويمكن أن يكون صورة Image أو صوت Voice أو أي شيء آخر يمكن أن يحول إلى Binary Data ..)، وكما هو الحال في نقل ال Text كنا نحول Text إلىASCII Code ثم إلى Binary أما في Object فيتم التعامل معه باستخدام ال Stream Classes والتي يتم الوصول إليها من النيم سبيسس System.IO وتحتوي هذه Classes على الBinary Reader و ال Binary Writer والتي تمكنك من التعامل مع أي Object انظر الصورة المرفقة :



حيث تساعدك هذه المكتبة على تحويل أي اوبجكت إلى Binary باستخدام Binary Writer لتسهل إرساله عبر الشبكة باستخدام Network Stream ثم تحويله مرة أخرى إلى اوبجكت باستخدام Binary Reader ، وكمثال تطبيقي على هذا سوف نقوم ببناء برنامج يقوم بعملية نقل الصورة من جهاز إلى أخر Client/Server وللبدء قم بإنشاء New Form جديد كما هو في الشكل التالي :



في البداية قم بإضافة النيم سبيسس التالي :

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

للإجراء عملية الإرسال لا بد أولا من اشتقاق اوبجكت من الكلاس MemoryStream والتي سوف نستخدمها لتخزين الصورة داخل الذاكرة بشكل Stream مؤقت لكي نحولها لاحقا إلى مصفوفة Binary ثم إرسالها باستخدام NetworkStream عبر ال Socket إلى جهاز السيرفر والذي سآتي على شرحه بعد قليل ، انظر الكود التالية :

```
try
{
تحديد الباث الخاص بصورة
openFileDialog1.ShowDialog ();
string mypic_path =openFileDialog1.FileName ;
pictureBox1.Image = Image.FromFile(mypic_path);

MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms,pictureBox1.Image.RawFormat);
تخزين الصورة ووضعها في مصفوفة من النوع بايت
byte[] arrImage = ms.GetBuffer();
ms.Close();
الاتصال بجهاز السيرفر عبر العنوان والبورت المحدد
TcpClient myclient = new TcpClient (txt_host.Text,5020);//Connecting with
server
```

إرسال الصورة المخزنة إلى جهاز السيرفر

```
NetworkStream myns = myclient.GetStream ();
BinaryWriter  mysw = new BinaryWriter (myns);
```

```
mysw.Write(arrImage);//send the stream to above address
```

إغلاق السوكت والجلسة و ال Stream

```
mysw.Close ();
myns.Close ();
myclient.Close ();

}
catch (Exception ex){MessageBox.Show(ex.Message );}
```

ثانيا : ال Server

سوف ابدأ في هذا الجزء شرح الجزء الخاص بالسيرفر والذي يقوم بعملية التصنت على البورت واستقبال ال Stream عبر ال Socket و قراءتها باستخدام ال Binary Reader وتحويله إلى اوبجكت (صيغته التي كان عليها قبل الإرسال) مرة أخرى ، في هذا المثال نريد استقبال صورة وفي هذه الحالة وفرت لدينا الدوت نيت خصائص جديدة في ال Controls الموجودة فيها ومن ضمنها خاصية Image.FromStream الخاصة ب ال PictureBox والتي تسهل علينا إمكانية عرض الصورة المرسلة من خلال Stream لكي يتم تحويلها من Binary Stream إلى صورة تعرض على ال Picturebox انظر المثال التالي :

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

// Objects  Declaration
TcpListener mytcpl; // Declare TCP Listener
Socket mysocket; // Declare an object from Socket Class
NetworkStream myns; //
StreamReader mysr;
void Image_Receiver()
{

mytcpl = new TcpListener (5000);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();// Accept Any Request From Client and
Start The Session

myns = new NetworkStream (mysocket);// Receive The Binary Data From
Port
pictureBox1.Image = Image.FromStream(myns); // Show The Image that
Resaved as Binary Stream
mytcpl.Stop();// Close TCP Session

    if (mysocket.Connected ==true)//if Connected Start Again
{

while (true)
{
Image_Receiver();// Back to First Method
}
}
}
```

ولتطبيق قم بإنشاء New Form جديد كما في الشكل التالي :

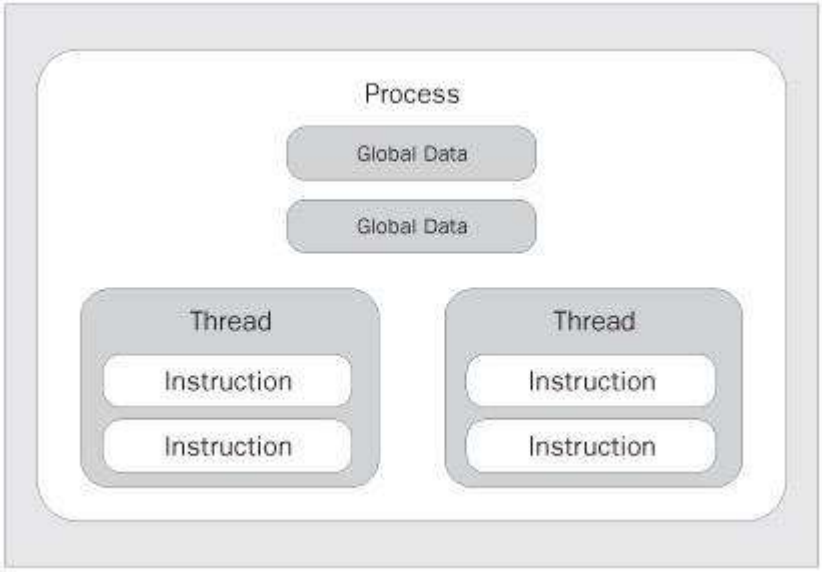
أضف الميثود السابقة في كلاس البرنامج ثم قم باستدعائها بواسطة وضع
Image_Receiver()اما في ال Constructer الخاص بالبرنامج أو بحدث بدأ التشغيل
الخاص بالفورم ، وقم بوضع الميثود التالية في حدث Closing الخاص بالفورم :

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

وذلك لتأكد من إغلاق السوكت عند إنهاء البرنامج ،..
قم بإضافة الكود التالي إلى ال Save لكي تتمكن من تخزين الصورة المستقبلة

```
private void menuItem1_Click(object sender, System.EventArgs e)
{
 try
 {
  saveFileDialog1.ShowDialog ();
  string mypic_path = saveFileDialog1.FileName;
  pictureBox1.Image.Save(mypic_path);
 }
 catch (Exception){}  }
```

قم بتشغيل البرنامج الآن ...
ماذا تلاحظ !! لقد لاحظت أن البرنامج بطيء جدا لدرجة لا يمكن فيها فتح أي برنامج أخر ، فكر بالسبب ..
لن أترككم تفكرون كثيرا كون أن ذلك قد يؤدي إلى تعليق الجهاز بالكامل وخاصة إذا لم يتوفر لديك الحجم الكافي من ذاكرة RAM أو إذا كان المعالج لديك بطيء نسبيا ..
السبب في ذلك Loop الذي لا ينتهي وعملية التصنت على البورت والتي لا تنتهي أيضا حيث أن البرنامج يعمل على الجزء العام والمخصص لإدارة نظام التشغيل في المعالج وهذا يعني انه لا يوجد مجال لفتح برنامج جديد إذ أن الموارد جميعها محجوزة ، إذا ما هو الحل ....
لقد وفرت الدوت نيت الحل لهذه المشكلة وهي باستخدام تكنولوجيا ال Threading والتي تسمح بالمعالجة المتوازية على نفس المعالج وذلك من خلال تقسيم المهام على المعالج وعمل Session منفصلة لكل برنامج وهو ما يسمى بال Multitasking..
وهنا لا يؤثر البرنامج على موارد النظام بشكل كبير انظر الشكل التالي :



لاحظ انه قبل إضافة ال Thread كان البرنامج يعمل على منطقة ال Global Area وهذا هو سبب البطء الشديد وبعد استخدام ال Thread تم عمل Session خاص للبرنامج بحيث يعمل بشكل متوازي مع بقية البرامج ..
ولاستخدام ال Thread في البرنامج يلزم أولا تعريف المكتبة أو انيم سبيس الخاص بها وهو

```
using System.Threading;
```

ثم اشتقاق اوبجكت منه وإدراج اسم الميثود التي تريد عمل Thread لها كما يلي : اكتب هذا الكود في حدث بدأ التشغيل للفورم Form1_Load

```
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Image_Receiver));
myth.Start ();
```

ألان قم بإضافة Application.ExitThread في حدث ال Closing Form كما يلي

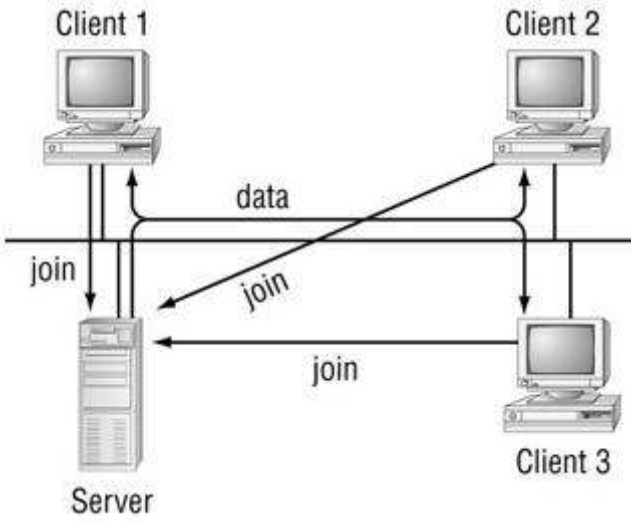
```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
```

```
try
{
mytcpl.Stop ();
Application.ExitThread ();
Application.Exit();
}
catch (Exception ex) {MessageBox .Show (ex.Message );}

}
```

ميزة ال Thread رائعة جدا إذ تمكنك من تشغيل أكثر من Thread وفي نفس الوقت وفي نفس البرنامج وهو ما يسمى بال Multithreading والذي سآتي على شرحه لاحقا ..

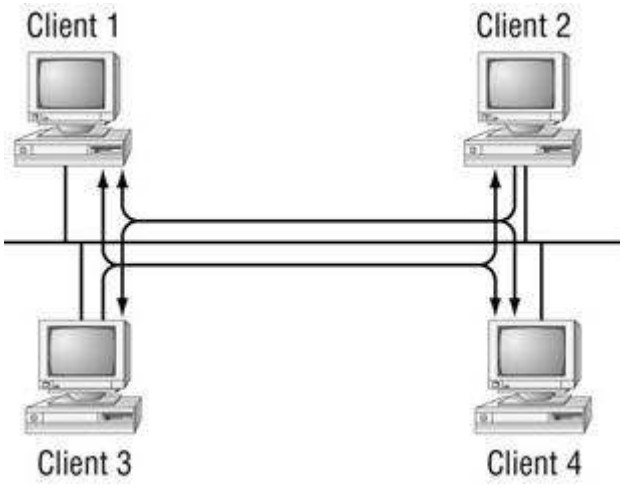
الدرس الرابع: IP Multicasting واستخدامها لعمل Multicasting Group

تحدثنا سابقا عن برتوكول ال UDP وشرحنا كيفية استخدامه لعمل برود كاست حيث تستطيع عمل البرود كاست بطريقتين إما باستخدام IPAddress.Any والذي يلزمه وجود سيرفر يقوم بعملية التصنت على البورت المحدد حيث يستقبل من خلاله أي رسالة ثم يقوم ببثها إلى كل الأجهزة أو باستخدام IPAddress.Broadcast والذي من خلاله يمكن عمل بث إلى كل الأجهزة حيث لا ضرورة لوجود جهاز سيرفر بحيث أن الكل يمكنه التصنت على بورت معين يستقبل ويرسل من خلالها أي رسالة إلى كل الأجهزة وتشبه عملية البرودكاست عملية البث الإذاعي حيث أن الجميع يستمع إلى شخص واحد ولاكن يختلف بأن أي شخص يستطيع الإرسال و الاستقبال وفي نفس الوقت ... ، وفي هذا الدرس سوف نتحدث عن ال IP Multicasting وذلك بهدف استخدامه لعمل ال Multicasting ، يعتبر هذا الموضوع من المواضيع المهمة جدا في برمجيات الشبكات ولهذا خصصت له درس منفصل عن البقية إذ أن اغلب برمجيات ال Conferences تعتمد عليه بشكل كبير ويعرف Multicast على انه الإرسال إلى مجموعة من المستخدمين (اثنان أو أكثر) سواء كان Managed باستخدام Client/Server حيث يكون هنالك جهاز Server في الشبكة وظيفته استقبال الرسائل من ال Clients Group ثم إرسالها إلى كامل المجموعة مرة أخرى انظر إلى الشكل التالي :



لاحظ انه يتم إرسال طلب الانضمام إلى المجموعة من قبل ال Clients وإذا وافق السيرفر على الطلب يقوم بضم عنوان الجهاز إلى ال IP Address List Members الخاصة به حيث يعيد توزيع الرسائل المستقبلة إلى كل الأعضاء الموافق عليهم و الموجودين في قائمة عناوين الأعضاء .

النوع الثاني ويسمى بال peer-to-peer Technique - unmanaged حيث أن كل جهاز يعمل ك server و client في نفس الوقت ولا وجود لجهاز سيرفر مركزي مخصص لعملية الاستقبال والتوزيع حيث تتم الموافقة على طلب الانضمام إلى المجموعة بشكل تلقائي وأي جهاز في المجموعة له الحق في الانضمام ثم الاستقبال و الإرسال إلى كامل المجموعة لاحظ الشكل التالي :



تم تخصيص عناوين خاصة للـ Multicasting وهو ما يسمى بال IP Multicast Address وهي كما يلي :
المدى من 224.0.0.0 إلى 224.0.0.255 لشبكات المحلية LAN
المدى من 224.0.1.0 إلى 224.0.1.255 لل Internetwork
المدى من 224.0.2.0 إلى 224.0.255.255 لل AD-HOC Network block
كما يوجد تخصيصات أخرى له سوف آتي على ذكرها عند الحاجة ...

قدمت الدوت نيت دعم كبير للـ IP Multicast باستخدام ال Socket Namespace حيث يتم تعريفها باستخدام ال الميثود SetSocketOption والتي تقوم بإدارة عمليات الانضمام والخروج من والى المجموعة multicast group (join & leave) كما تستخدم لإضافة وإلغاء العضوية AddMembership و DropMembership و تستخدم الميثود UdpClient Object لتحديد رقم البورت والتي سيتم استقبال البيانات من خلالها بالإضافة إلى تعريف ال IP Multicasting والذي من خلاله تحدد الجهات التي سوف تستقبل الرسالة من خلال تحديد ال Range IP الخاص بشبكات المحلية LAN حيث يستطيع أي شخص يتنصت على هذا البورت ويستخدم نفس ال Range استقبال هذه الرسالة ، يستخدم الكود التالي لإرسال رسالة إلى عدة جهات بحيث نستخدم رقم البورت 9050 و ضمن ال Range 224.100.0.1 كمثال:

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

class MultiSend
{
  public static void Main()
  {
   Socket server = new Socket(AddressFamily.InterNetwork,
          SocketType.Dgram, ProtocolType.Udp);
   IPEndPoint iep = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);

   byte[] data = Encoding.ASCII.GetBytes("This is a test message");
   server.SendTo(data, iep);
   server.Close();
}
```

}
في البداية قمنا بتعريف السوكت بتحديد الجهة التي سوف تستقبل الرسالة وهي (أي شخص يتنصت على الشبكة) ثم تحديد نوع السوكت والبرتوكول المستخدم ، وبعد ذلك تحديد نقطة الهدف وذلك بوضع ال IP Multicast الذي نريد ويتبعه رقم البورت التي سيتم استقبال البيانات من خلالها (بقية الكود تم شرحه سابقا عندما استخدمنا ال UDP لعمل برود كاست) ..

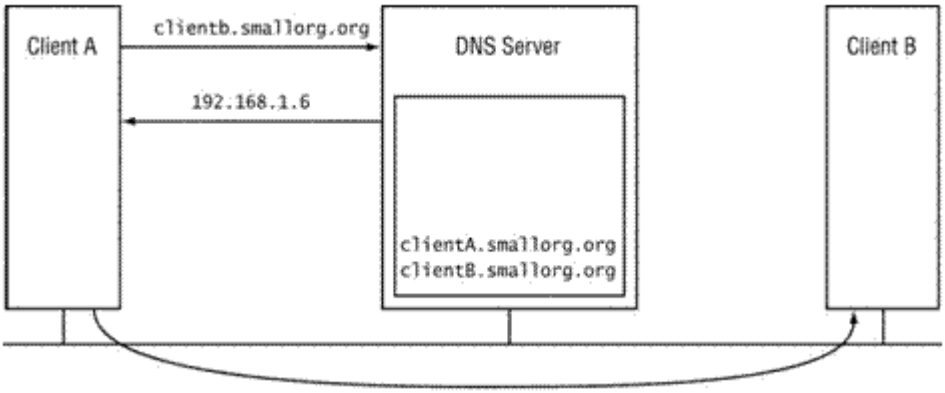
ولإنشاء برنامج الاستقبال سوف نستخدم تعريف السوكت نفسه ونضيف ال UdpClient Object ونسند له رقم البورت التي نريد التصنت عليها

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class UdpClientMultiRecv
{
  public static void Main()
  {
   UdpClient sock = new UdpClient(9050); // التصنت على رقم البورت هذا

   sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50);
    وهذا يعني انك سوف تتصنت على المدى المحدد

   IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

 //  استقبال البيانات وتعبئة الرسالة في مصفوفة من النوع بايت
   byte[] data = sock.Receive(ref iep);

// التحويل إلى اسكي كود ثم طباعة الرسالة على الشاشة
   string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
   Console.WriteLine("received: {0} from: {1}", stringData, iep.ToString());
   sock.Close();
  }
}
```

لاحظ انه توجد طرق متعددة لاستقبال البيانات و إرسالها كما يمكن استخدام الكوديين السابقين في نفس البرنامج للإرسال و الاستقبال كما يمكنك إرسال صورة إلى جانب النص (انظر الدرس الثاني) أو أي شيء أخر يمكن تحويله إلى Binary إذ ما عليك سوى إضافة ال memory Stream وال Binary Reader وال Binary Writer إلى كود الإرسال و الاستقبال كما يمكنك عمل برنامج لإرسال صورة عبر الكاميرا إلى جهات متعددة باستخدام نفس الخاصية والتي سآتي على شرحها في الدروس اللاحقة إنشاء الله ...
في هذا الدرس قمنا بتوضيح الأساسيات في ال IP Multicasting وسوف نأتي على شرح التفاصيل لاحقا إنشاء الله.

الدرس الخامس: DNS Programming

تعتبر خدمة DNS واحدة من أهم الخدمات التي تستخدم في الإنترنت والشبكات بشكل عام، وتختصر وظيفة DNS بالقيام بعملية ترجمة ال Domain Name إلى Domain IP من وإلى العكس ويتم ذلك من خلال مجموعة كبيرة جدا من مزودات DNS (والتي تقوم بتحديث قاعدة البيانات الخاصة بها كل فترة معينة) ، تبدأ هذه العملية بقيام الـClient A بطلب ال Domain الخاص بال Client B وذلك بإدخال Domain Name الخاص به - حيث تم مسبقا قيام ال Client B بتعريف نفسه في قاعدة البيانات الخاصة ب DNS Server - كما يحتوي كل Client على قاعدة بيانات تحتوي على عناوين ال Domains وتسمى بال local DNS حيث يقوم بالبحث بداخلها على عنوان Domain من خلال Domain Name فإذا لم يجده يقوم بطلب عنوان الدومين من ال DNS Server وبعد إيجاده يقوم ال DNS Server بإرسال العنوان إلى ال Client ويقوم بدوره بتخزين العنوان في Local DNS الخاص به ، انظر إلى الشكل التالي:



في الدوت نيت يمكننا التعامل مع DNS باستخدام النيم سبيس System.Net والتي تحتوي على جميع ال DNS Classes والتي تحتوي على كل ال Methods الخاصة ب DNS وتقسم هذه الميثودس إلى قسمين متزامن Synchronous Methods و غير متزامن Asynchronous Methods وهي كما يلي:

أولا الميثودس المتزامنة Synchronous Methods وهي :
GetHostName والتي تستخدم لجلب اسم الهوست وترجع هذه الميثود قيمة String تحتوي على ال Computer Name ولا تأخذ هذه الميثود أي باراميترات ويمكن استخدامها كما يلي :

```
string hostname = Dns.GetHostName();
```

الميثود GetHostByName و الميثود GetHostByAddress وتستخدم كل منها كما يلي :

```
IPHostEntry host_ip = Dns.GetHostByName(Computer_Name); // لجلب العنوان باستخدام الاسم
IPHostEntry host_name = Dns.GetHostByAddress(IP_Address); // لجلب الاسم باستخدام العنوان
```

الميثود Resolve وهي Overloaded Method حيث ترجع Host Name إذا أرسلت لها IP Address وترجع Host Address إذا أرسلت لها Host Name في ال IPHostEntry ولا يختلف استخدامها عن استخدام الميثودس السابقة .
وهذا المثال يبين طريقة استخدامها :

```
using System;
using System.Net;

class FMO_DNS
{
	public static void Main()
	{
		IPHostEntry IPHost = Dns.Resolve("www.yahoo.com"); // الدومين الذي نريد معرفة الأي بي الخاص به
		Console.WriteLine(IPHost.HostName); //  جلب اسم الدومين بالكامل

		IPAddress[] addr = IPHost.AddressList; //  وضع قائمة العناوين في مصفوفة

		for(int i= 0; i < addr.Length ; i++) // طباعة عناصر المصفوفة
		{
			Console.WriteLine(addr[i]);
		}
	}
}
```

ثانيا الميثودس غير المتزامنة Asynchronous Methods :
وتبدأ عادة بكلمة Begin أو End ومن الأمثلة عليها :
BeginGetHostByName و EndGetHostByName و BeginResolve و EndResolve
طبيعة عملها كما هو الحال في الميثودس المتزامنة لكنها تختلف بكون انه لا يشترط تنفيذها لإكمال عمل البرنامج في حين المتزامن لا تسمح بلإنتقال إلى الخطوة الثانية في البرنامج إلا في حالة انتهاء عملها وقد تسبب هذه السيئة بخفض البريفورمانس بشكل عام في البرنامج لذلك ينصح باستخدام الطريقة الغير متزامنة وتستخدم كما يلي : Begin___

```
public static IAsyncResult BeginResolve(string hostname,
  AsyncCallback requestCallback, object stateObject)
```

حيث يتم وضع الهوست نيم في البراميتر الأول و البراميتر الثاني يعرف فيه ال delegate وتسمح لك بتمرير مدخلات إلا delegate ، ويستخدم End____ كما يلي :

```
public static IPHostEntry EndResolve(IasyncResult ar)
```

وهنا مثال شامل و بسيط يقوم بجلب جميع الأيبز الموجودة على الشبكة حيث يعمل على جلب ال host names من ProcessStartInfo من خلال الخاصية StandardOutput حيث يتم تحويله إلى host name من خلال الميثود GetMachineNamesFromProcessOutput ثم تخزينها في Collicaion ثم يتم تحويل الأسماء إلى عناوين من خلال الميثود Dns.Resolve .. طبعا يتم استخدام ال StreamReader لقرائة ال collection الخاص بال ProcessStartInfo وهذا هو المثال :

```
using System;
using System.IO;
using System.Diagnostics;
using System.Net;
using System.Collections.Specialized;

namespace NetworkIPs
{
	public class Names

	{
		public StringCollection GetNames()

		{
			ProcessStartInfo _startInfo = new
ProcessStartInfo("net","view");
			_startInfo.CreateNoWindow = true;
			_startInfo.UseShellExecute = false;
			_startInfo.RedirectStandardOutput = true;
			Process _process = Process.Start(_startInfo);
			StreamReader _reader = _process.StandardOutput;
			StringCollection _machineNames =
GetMachineNamesFromProcessOutput(_reader.ReadToEnd());
			StringCollection _machineIPs = new StringCollection();
			foreach(string machine in _machineNames)
			{
				_machineIPs.Add(IPAddresses(machine));
			}
			return _machineIPs;
		}
		private static string IPAddresses(string server)
		{
			try
			{
		System.Text.ASCIIEncoding ASCII = new
System.Text.ASCIIEncoding();
				// Get server related information.
				IPHostEntry heserver = Dns.Resolve(server);
				//assumin the machine has only one IP address
				return heserver.AddressList[0].ToString();
			}
			catch
			{
				return "Address Retrieval error for " + server;
			}
		}
		//string manipulations
		private StringCollection
GetMachineNamesFromProcessOutput(string processOutput)
```

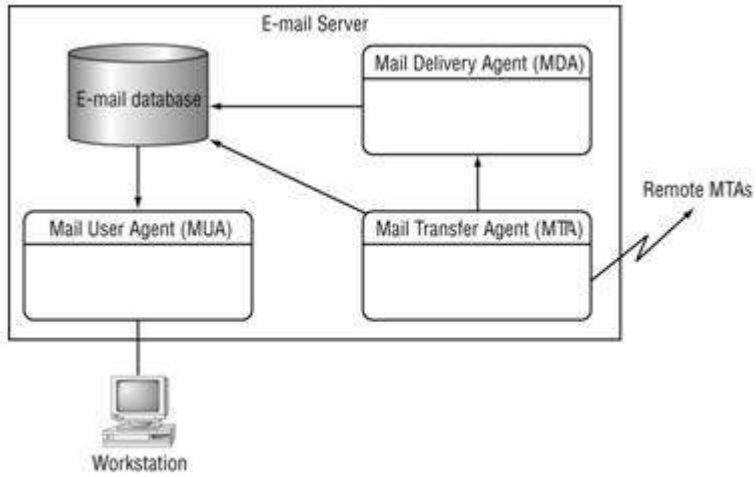
```
            {
                string _allMachines = processOutput.Substring(
processOutput.IndexOf("\\"));
                StringCollection _machines= new StringCollection();
                while(_allMachines.IndexOf("\\") != -1 )
                {
      _machines.Add(_allMachines.Substring(_allMachines.IndexOf("\\"),
                        _allMachines.IndexOf("
",_allMachines.IndexOf("\\")) -
_allMachines.IndexOf("\\")).Replace("\\",String.Empty));
                        _allMachines =
_allMachines.Substring(_allMachines.IndexOf(" ",_allMachines.IndexOf("\\") +
1));
                }
                return _machines;
            }
        }
        public class Runner
        {
            static void Main()
            {
                Names _names = new Names();
                StringCollection names = _names.GetNames();
                foreach(string name in names)
                Console.WriteLine(name);
                Console.ReadLine();
            }
        }
```

الدرس السادس: SMTP & POP3 Programming

تحدثنا في الدرس السابق عن برمجة برتوكول DNS والمسئول عن عملية ترجمة Domain من اسم نطاق إلى IP وبالعكس وبينا كيفية القيام بهذه العملية في سي شارب ، في هذا الدرس سوف نتحدث عن برمجة بعض البروتوكولات الأخرى لطبقة ال Application Layer وهما هنا ال SMTP والمسئول عن إرسال الرسائل عبر البريد الإلكتروني و ال POP3 والمسئول عن عملية توصيل الرسالة إلى الزبون من خلال عمل Download لها من ال Mail Server وفي الدرس اللاحق سوف نتحدث عن ال HTTP Programming والذي يستخدم بشكل أساسي في تصفح ال Web ، مع العلم انه يوجد بروتوكولات كثيرة سوف آتي على شرحها عند الحاجة ..

الجزء الأول: SMTP – Simple Mail Transfer Protocol Programming

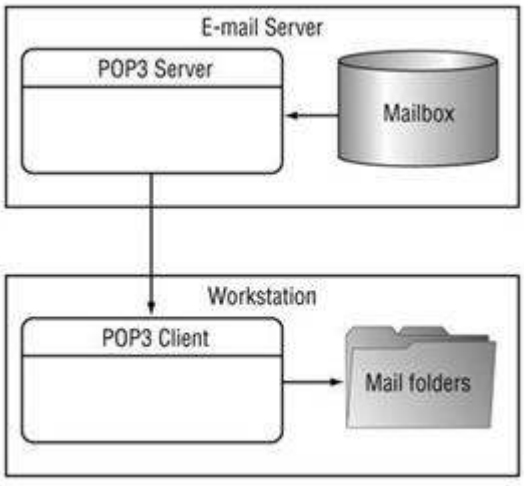
من المعروف أن ال Mail Server يقوم بتجزئة عمليات إرسال و استقبال البريد الإلكتروني عبر الإنترنت إلى ثلاثة أجزاء وهي كما في الشكل التالي :



Message Transfer Agent – MTA والمسئول عن الإرسال Outgoing والتوصيل Incoming للرسائل

Message Delivery Agent -MDA و المسئول عن عمليات ال filtering والتأكد من وصول الرسالة

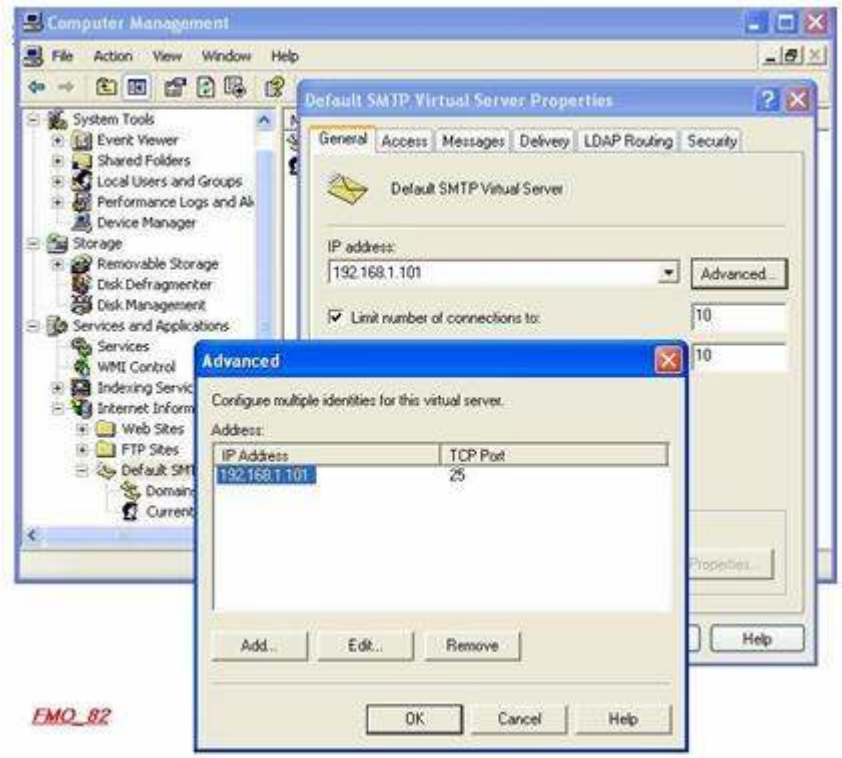
Message User Agent -MUA والمسئول عن عملية قراءة و تخزين الرسالة في Database لدى المستقبل Client وتتم هذه العملية باستخدام بروتوكول POP - Post Office Protocol انظر إلى الشكل التالي :



و يستخدم برتوكول ال SMTP Simple Mail Transfer Protocol بشكل أساسي في ال MTA أي عمليات إرسال Outgoing وتوصيل Incoming الرسائل .

لتطبيق يجب أولا التأكد من أنك تملك حساب SMTP من ال Internet Provider الخاص بك تستطيع تجربة ال Account الخاص بك من خلال برنامج ال Outlook Express الموجود مع ال Windows إذا كنت لا تملك حساب SMTP تستطيع تجربة البرنامج من خلال إنشاء Virtual SMTP Server عن طريق ال IIS وذلك بتثبيتها من : Control Panel >> Add/Remove Programs تأكد من تفعيل كل من ال IIS وال SMTP كما في الشكل التالي :

ثم إعداد السيرفر من ال IIS كما في الشكل التالي :



تدعم الدوت نيت استخدام بروتوكول ال SMTP من خلال النيم سبيس System.Web.Mail و تحتوي على الكلاس SmtpMail والتي من خلالها نستخدم الميثود Send والتي تستخدم لإرسال الرسالة عبر البورت 25 وهو البورت المخصص لبروتوكول SMTP و تعتبر الميثود " Send overloaded Method " حيث تأخذ عدة أشكال إذ بإمكانك استخدامها مع براميتر واحد إلى أربعة باراميترات ، وبشكل افتراضي نستخدم البرامترات التالية :

SmtpMail.Send(string from, string to, string subject, string body)
البراميتر الأول يوضع فيه عنوان المرسل والثاني يوضع فيه عنوان المرسل إليه و البراميتر الثالث لعنوان الرسالة والرابع لنص الرسالة .

ولعمل برنامج يقوم بإرسال البريد الإلكتروني قم بإنشاء فورم كما في الشكل التالي:

ثم قم بإضافة النيم سبيس System.Web.Mail ، (إذا لم تظهر لديك Mail. قم بإدراج النيم سبيس System.Web إلى ال References) ثم قم بكتابة الكود التالي :

لا تنسى إضافة النيم سبيس هذا في بداية البرنامج
```
using System.Web.Mail;
```

ثم كتابة الكود هذا في زر الإرسال
```
try
{
string from = textBox1.Text;
string to = textBox2.Text;
string subject = textBox3.Text;
string body = textBox4.Text;
SmtpMail.SmtpServer = textBox5.Text;
SmtpMail.Send(from, to, subject, body);
}
catch (Exception ex) {MessageBox.Show(ex.Message);}
```

<u>ملاحظة هامة جدا :</u>
هذا الكود يعمل بشكل جيد، لاكن يجب التأكد من تفعيل ال SMTP من ال IIS كما ذكر في السابق وقم بوضع IP الخاص ب ال SMTP (والذي تم تعريفه مسبقا في SMTP Virtual Server)بال SMTP Server Textbox ، يجب التأكد أيضا من ال SMTP Server لديك يدعم استخدام المكتبة CDO2 - Microsoft Collaboration Data Objects Version 2 وإلا سوف تحصل على Exception يخبرك بأنه لا يستطيع الوصول إلى CDO2 Object ، في العادة يتم استخدمها مع Windows XP و Windows 2000 وتعمل بشكل افتراضي عند تثبيت ال SMTP Virtual Server أو مع Microsoft Exchange Server2003 أما إذا كنت تستخدم Exchange Version 5 أو 5.5 فسوف تحصل على ال Exception السابق الذكر .

يعتبر المثال السابق مثال بسيط لإرسال رسائل عبر SMTP باستخدام CDO2 ، وفي العادة عند إنشاء برامج مثل برنامج ال Outlook يتم استخدام ال HTML Format بالإضافة إلا إمكانية إرسال ملحقات وطبعا يعطيك عدة خيارات لإرسال و استقبال البريد الإلكتروني هل باستخدام ال HTTP أو ال POP3 ... وهنا سوف نقوم بإنشاء برنامج بسيط يقوم بإرسال واستقبال البريد الإلكتروني باستخدام ال SMTP و POP3 بنسبة لاستخدام ال POP3 فيجب أن يتوفر لديك حساب POP3 من ال ISP الخاص بك أو أن تقوم بتثبيت Microsoft Exchange Server2003 على جهازك وإعداده بحيث يستخدم ال POP3 إذ عندها سوف تحتاج لوجود Domain Controller مثبت على الجهاز و Windows 2003 Server بالإضافة إلى تثبيت الActive Directory عليه.

قدمت الدوت نيت دعم ممتاز لاستخدام هذه الخواص وذلك من خلال النيم سبيس System.Web.Mail وباستخدام الكلاس MailMessage لدعم ال HTML Format و الكلاس MailAttachment لدعم إمكانية إرسال ملحقات مع الرسالة ولاكن لبرمجة ال POP3 يلزم استخدام النيم سبيس System.Net.Sockets و System.Net و System.IO حيث يتم عمل Session خاص مع السيرفر للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئتها في List Box أو Treelist حسب الحاجة وعند الضغط على إحداها يقوم ال Client بعمل Download لرسالة من ال Mail Server ولعمل Advanced SMTP eMail Sender قم بأخذ Object من الكلاس MailMessage كما يلي :

```
using System.Web.Mail;

try
{
 MailMessage mm = new MailMessage();

ثم إضافة الكود التالي وكما في السابق
   mm.From = textBox1.Text;
      mm.To = textBox2.Text;
  // mm.Cc = لاحظ انه يمكنك من الإرسال لأكثر من شخص هذه حسب الحاجة
  // mm.Bcc = لاحظ انه يمكنك من الإرسال لأكثر من شخص هذه حسب الحاجة
   mm.Subject = textBox3.Text;
   mm.Headers.Add("Reply-To", "fadi822000@yahoo.com"); // لوضع أي إضافات
تريدها مع الرسالة
   mm.Headers.Add("Comments", "This is a test HTML message");
   mm.Priority = MailPriority.High; // يمكنك وضع خيارات أهمية الرسالة
   mm.BodyFormat = MailFormat.Html; // نوع الفورمات المستخدم
   mm.Body = "<html><body><h1>" + textBox4.Text + "</h1></html>";
   SmtpMail.Send(mm);
}
catch (Exception ex) {MessageBox.Show(ex.Message);}
```

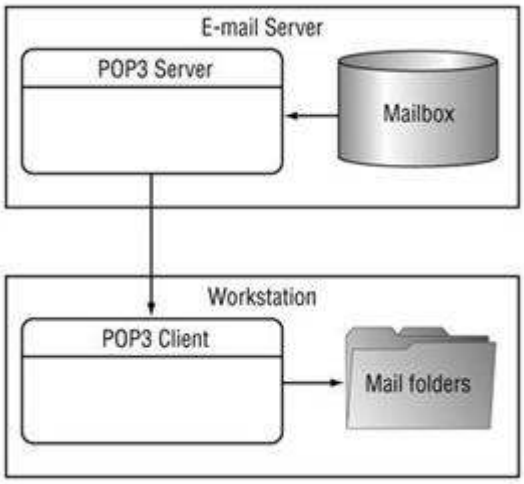
لاحظ أن جسم الرسالة يستخدم كود ال HTML وهذا يمكنك من وضع أي لون أو حجم أو أي شيء يمكن عمله باستخدام ال HTML (راجع قسم ال HTML بالمنتدى لتعرف على هذه اللغة السكربتية الرائعة) ، ولجعل البرنامج قادر على إرسال ملحقات يجب استخدام الكلاس MailAttachment وإدراج اسم الملف فيه وكما يلي بالكود :

```
MailAttachment myattach =
new MailAttachment("Your_Attached_File_path.extension", MailEncoding.Base64);
```

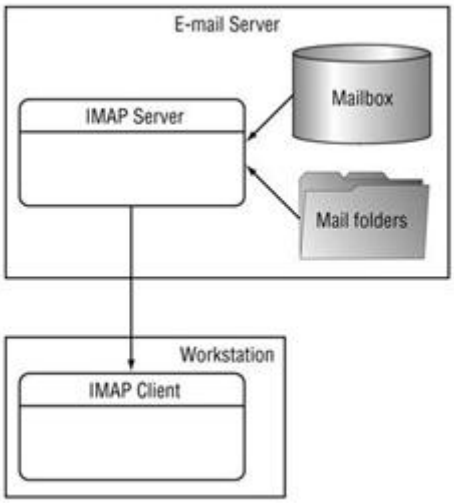
وهنا قد انتهينا من عمل برنامج ال SMTP بشكل كامل ، طبعا عملية ال Design وغيرها تعتمد على حسب ذوق وذكاء وخبرة المبرمج.

ثانيا : POP3- Post Office Protocol Version 3 Programming

كما تحدثنا سابقا فإن وظيفة برتوكول ال POP3 والذي يعمل في جزء ال MUA - Mail User Agent على بورت 110 ضمن برتوكول ال TCP تكمن في كونه المسئول عن عملية توصيل الرسالة إلى الزبون Client من خلال عمل Download لها من ال Mail Server حيث تحفظ الرسائل في ال Mail Folder والموجود أساسا في جهاز ال Client أنظر إلى الشكل التالي:



ومن البدائل لل POP3 برتوكول IMAP – Interactive Mail Access Protocol فمن خلاله يستطيع المستخدم إنشاء Mail Folder خاصة به ولاكن في ال Mail server وليس في جهاز الزبون وتعتبر هذه من ميزات ال IMAP وسيئاته بنفس الوقت إذ أن قراءة الرسالة تتم مباشرة من خلال ال Server حيث تستطيع قراءتها من أكثر من Client ولاكن المشكلة فيه هي تحكم مدير خادم الرسائل Mail Server Administrator بحجم ال Mail Folder إذ تكون في العادة سعتها محدودة أنظر إلى الشكل التالي :



لاحظ أن ال Mail Folder يقع ضمن Mail Server ويتم قراءته بعد التحقق Authentication من اسم المستخدم وكلمة المرور لاكن كما قلنا فإن مشكلته تكمن في محدودية سعة ال Mail Folder لذا ينصح لشركات الكبيرة استخدام ال POP3 كونه غير محدود السعة فالذي يتحكم في السعة هو ال Client ولا دخل ل Mail Server Administrator بها.

وبما أننا قررنا اعتماد ال POP3 لعملية قراءة الرسائل سوف نبدأ ببرمجته إذ يلزم الأمر استخدام النيم سبيس System.Net.Sockets و System.Net و System.IO حيث يتم

عمل Session خاص مع السيرفر باستخدام ال Socket للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئة عناوينها في List Box أو Treelist خاص حسب الحاجة وعند الضغط على إحداها يقوم ال Client بعمل Download لرسالة من ال Mail Server إلى ال Mail Folder ثم عرضها في Textbox.

ولتطبيق قم بإنشاء فورم جديد كما يظهر في الشكل التالي :

ثم قم بإضافة النيم سبيسس التالية :

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

لاحظ انه يتم التعامل مع ال Socket وال Stream لإنشاء Session مع السيرفر باستخدام برتوكول ال TCP وقراءة الرسالة من ال POP3 Server .

ثم قم بإضافة التعاريف التالية في بداية البرنامج (أي بعد تعريف الكلاس الرئيسي – في منطقة ال Global Declaration) :

```
public TcpClient Server;// اشتقاق اوبجكت من برتوكول التي سي بي وذلك بهدف إنشاء الجلسة
public NetworkStream NetStrm;// سوف نستخدمه لإرسال معلومات المستخدم
public StreamReader  RdStrm; // لقراءة المعلومات الواردة من البوب 3 سيرفر
public string Data; // لاستخدامها في معرفة عدد الرسائل
public byte[] szData; // لتخزين البيانات الواردة من البوب 3 سيرفر
public string CRLF = "\r\n";// لاستخدامها في البرنامج لعمل سطر جديد..
```

في ال Connect Button قم بإضافة الكود التالي :

```
// create server POP3 with port 110
// لإنشاء سيشن مع البوب سيرفر عبر البورت المخصص وهو 110
Server = new TcpClient(POPServ.Text,110);

try
{
// initialization
NetStrm = Server.GetStream();
RdStrm= new StreamReader(Server.GetStream());
Status.Items.Add(RdStrm.ReadLine());

// Login Process
//إدخال اسم المستخدم وكلمة المرور وتمريرها إلى البوب سيرفر
Data = "USER "+ User.Text+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
Data = "PASS "+ Passw.Text+CRLF;
// بعد التأكد من اسم المستخدم وكلمة المرور يتم قراءة صندوق الوارد الخاص
بالمستخدم
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());

Send STAT command to get information ie: number of mail and size
                لمعرفة عدد الرسائل الموجودة في POP3 Server باستخدام الأمر STAT
Data = "STAT"+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
```

قم بإضافة الكود التالي إلى ال Disconnect Button :

```
// Send QUIT command to close session from POP server
Data = "QUIT"+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
//close connection
NetStrm.Close();
RdStrm.Close();
```

ولقراءة الرسائل من صندوق الوارد(بشكل افتراضي سيتم قراءة الرسالة الأخيرة) قم بإضافة الكود التالي إلى ال Read Last Come Email Button :

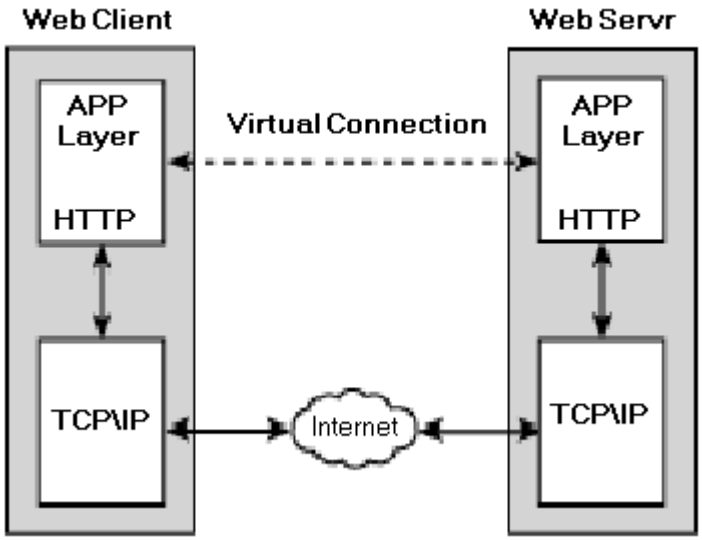
```
string szTemp;
Message.Clear();
try
        {

// retrieve mail with number mail parameter
Data = "RETR 1"+CRLF; // لتحديد رقم الرسالة المراد قراءتها

szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
szTemp = RdStrm.ReadLine(); // تخزين الرسالة بشكل مؤقت حتى يتم طباعتها
        if(szTemp[0]!='-')
        {
        while(szTemp!=".")
                {
        Message.Text += szTemp+CRLF;
        szTemp = RdStrm.ReadLine();
        }
                }
        else
        {Status.Items.Add(szTemp);}
        }

        catch(InvalidOperationException err)
        {Status.Items.Add("Error: "+err.ToString());}
```

وهنا قد قمت بشرح كيفية عمل البوب 3 وبرمجته في الدوت نت وهذا مثال بسيط تستطيع البدء منه لعمل مشروع كامل شبيه بال Outlook الخاص بميكروسوفت حيث تستطيع استخدام ملف ال DLL الخاص بالإنترنت إكسبلورر لعرض الرسائل الواردة بدلا من عرضها على شكل HTML Code كما تستطيع عمل Tree List لوضع الرسائل الواردة حيث يكون لكل رسالة رقم تسلسلي يتم وضعه في الكود السابق لقراءتها حيث استخدمت الرقم 1 بشكل افتراضي والذي يقوم بقراءة الرسالة الأخيرة الواردة ،

الدرس السابع: HTTP – Hyper Text Transfer Protocol Programming

تتلخص وظيفة ال HTTP بشكل عام على انه البرتوكول المستخدم لتوصيل طلب المستخدم User Request إلى الويب سيرفر ثم قيام ال web server بالرد على ال Request والذي يسمى ب Server Response وبتأكيد تستطيع نقل جميع أشكال ال (Multimedia) من النص وصورة و صوت و فيديو وغيره .. من ال Web Server إلى ال Client Application باستخدام Byte Stream object.

يعمل برتوكول ال HTTP على ال Application Layer وهذا يعني استخدامه بشكل مباشر من واجهة المستخدم كما هو الحال في DNS,SMTP,POP3,FTP انظر إلى الشكل التالي:



أولا : Downloading From Web Server

نستطيع التعامل مع ال Web Server في الدوت نيت باستخدام الكلاس WebClient الموجود في النيم سبيس System.Net إذ تقدم لنا جميع الإمكانيات لتوصيل طلب الزبون و الرد عليه User Request & Server Response وتدعم ال WebClient Class ثلاثة Methods لتحميل البيانات من ال Web Server وهي:

1- **DownloadData** ووظيفتها جلب البيانات من ال Web Server وتخزينها في Byte Array وتعرض على شكل HTML Code وتستخدم كما يلي كمثال :

```
using System;
using System.Net;
using System.Text;
class DownloadData_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
byte[] response =
wc.DownloadData("http://www.google.com");
Console.WriteLine(Encoding.ASCII.GetString(response));
}
}
```

2- **DownloadFile** ووظيفتها نقل ملف ما من ال Web Server وتخزينها مباشرة في Local Computer وهو سهل الاستخدام جدا إذ ما عليك سوا تمرير موقع الملف والمكان الذي تريد تخزين الملف فيه ويستخدم كما يلي كمثال :

```
using System;
using System.Net;

class DownloadFile_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
string filename = "C:\\ra.zip";

Console.WriteLine("Download in Progress Please Waite...");

wc.DownloadFile("http://www.personalmicrocosms.com/zip/ra.zip", filename);

Console.WriteLine("file downloaded");
}
}
```

3- **OpenRead** ووظيفتها إنشاء Read Only Stream بين الزبون والسيرفر لجلب بيانات من URL محدد وتخزينه في Stream Object بعد تمرير ال URL للموقع الذي تريد عرضه وباستخدام الميثود ReadLine نستطيع عرض البيانات المخزنة في ال Stream Object على شكل HTML Code .
ملاحظة : تستخدم الميثود Peek لمعرفة نهاية ال Stream Object .

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

class OpenRead_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
string response;

Stream strm = wc.OpenRead("http://www.google.com");
StreamReader sr = new StreamReader(strm);

while(sr.Peek() > -1)
{
response = sr.ReadLine();
Console.WriteLine(response);
}
sr.Close();
}
}
```

ويحتوي ال **WebClient Class** على مجموعة من ال **Properties** والتي تستخدم لجلب معلومات عن ال Web Host مثل property ResponseHeaders والذي يستخدم لجلب معلومات هامة عن ال web host مثل عدد ال Headers ونوع ال cash control واسم ال Server و نوع ال Encoding المستخدم وغيرها من المعلومات الهامة، ويستخدم كما يلي كمثال:

```
using System;
using System.Net;

class ResponseHeaders_property
{
public static void Main ()
{
WebClient wc = new WebClient();
byte[] response =
wc.DownloadData("http://www.google.com");
WebHeaderCollection whc = wc.ResponseHeaders;
Console.WriteLine("header count = {0}", whc.Count);
for (int i = 0; i < whc.Count; i++)
{
Console.WriteLine(whc.GetKey(i) + " = " + whc.Get(i));
}
}
}

//Output:
//header count = 6
//Cache-Control = private
//Content-Type = text/html
//Set-Cookie = PREF=ID=6ae22f44980c5d78…
//7JRA; expires=Sun, 17-Jan-2038 19:14:
//Server = GWS/2.1
//Transfer-Encoding = chunked
//Date = Wed, 23 Nov 2005 10:10:58 GMT
```

ثانيا : Uploading to Web Server

يدعم ال WebClient أربعة Methods لتحميل البيانات إلى ال Web Server وهي :

1- **OpenWrite** ويستخدم لإرسال Stream Data إلى ال Web Server وذلك بعد تمرير عنوان ال URL للملف والنص الذي نريد كتابته على ال Web Page طبعا يجب أن تملك الصلاحيات لذلك ويستخدم كما يلي كمثال :

```
using System;
using System.IO;
using System.Net;
method_class OpenWrite
{
public static void Main ()
{
WebClient wc = new WebClient();
string data = "<h1>Welcome to My Page</h1>";
Stream strm = wc.OpenWrite("C:\\mypage.html");
```

```
StreamWriter sw = new StreamWriter(strm);
sw.WriteLine(data);
sw.Close();
strm.Close();
}
}
```

2 – **UploadData** ويستخدم لنقل محتويات مصفوفة من النوع Byte إلى ال Web Server وهذا يعني انك تستطيع من خلالها رفع أي نوع من البيانات مثل النص الصور الفيديو وغيره إلى ال web server بعد تحويلها إلى Byte Array ويستخدم كما يلي كمثال :

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

Method_class UploadData
{
public static void Main ()
{
WebClient wc = new WebClient();
string data = "This is The Text Before Converted it to Byte";
byte[] dataarray = Encoding.ASCII.GetBytes(data);
wc.UploadData("C:\\mydata.txt", dataarray);
}
}
```

3- **UploadFile** وتستخدم هذه الميثود لرفع ملف من ال Local Computer إلى ال Web Host وهي بسطة الاستخدام جدا وتستخدم كما يلي كمثال :

```
using System;
using System.Net;

class UploadFile_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
wc.UploadFile("http://www.yoursite.com", "C:\\myfile.html");
}
}
```

4- **UploadValues** وتستخدم لرفع **Collection** من البيانات وال values الخاصة بها إلى الويب سيرفر وذلك بعد تحويل ال **Collection** إلى Byte Array ولتعريف **Collection** نستخدم الكلاس NameValueCollection الموجود في النيم سبيس System.Collections.Specialized وبعد تعريفه نستخدم الميثود add لإضافة ال Collection جديد.. وتستخدم كما يلي كمثال :

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Text;

class UploadValues_Method
{

public static void Main ()
{
WebClient wc = new WebClient();
NameValueCollection nvc = new NameValueCollection();
nvc.Add("firstname", "Fadi");
nvc.Add("lastname", "Abdel-qader");
byte[] response =
wc.UploadValues("http://localhost/mypage.aspx", nvc);
Console.WriteLine(Encoding.ASCII.GetString(response));
}
}
```

ثالثا :المواضيع الأكثر تقدما في ال HTTP Programming:

يعتبر هذا الجزء من أهم الأجزاء في برمجة تطبيقات Web Client Applications والذي سوف نتحدث فيه عن استخدام كل من ال HttpWebRequest Class و ال HttpWebResponse Class :

1- استخدام HttpWebRequest Class :

يحتوي هذا الكلاس على مجموعة من ال Properties والتي تستخدم بشكل أساسي في تطبيقات ال Web Client Applications لإنشاء مثل :

1- استخدام خاصية ال Web Proxy : والتي نمرر فيها عنوان ال Proxy Server ورقم البورت حتى نستطيع التعامل مع ال HTTP Web Requests من خلف Proxy Server أو Firewall ويتم تعريف ال Proxy Server Prosperity كما يلي كمثال :

```
using System;
using System.Net;

class ProxyServer_Property
{
public static void Main ()
{
HttpWebRequest hwr = (HttpWebRequest)WebRequest.Create(
"http://www.google.com");

WebProxy proxysrv = new
WebProxy("http://proxy1.server.net:8080");
hwr.Proxy = proxysrv;

}
}
```

نعرف في البداية ال HttpWebRequest Object ثم نعرف WebProxy Object من الكلاس webProxy ونسند له عنوان ال Proxy Server ورقم البورت وبعد ذلك نستطيع إسناده إلى أي اوبجكت باستخدام الخاصية Proxy التي تكون موجودة عادة في جميع HttpWebRequest Objects ..

2- استخدام ال HttpWebrequest لإرسال بيانات إلى الويب سيرفر باستخدام ال Streams وتستخدم كما يلي كمثال :

```
HttpWebrequest hwr =
(HttpWebRequest)WebRequest.Create("http://localhost");
Stream strm = hwr.GetRequestStream();
StreamWriter sw = new StreamWriter(strm);
sw.WriteLine(data);
```

بعد تعريف ال HttpWebRequest Object نقوم بتعريف Stream Object ونسند له ال Request Stream من خلال الميثود GetRequestStream .

2 - استخدام HttpWebResponse Class:

تستخدم ال **HttpWebResponse Object** لإرجاع بيانات من الويب سيرفر إلى ال Client حيث نستخدم الميثود **GetResponse** و الميثود **BeginGetResponse** لهذه العملية ولا يوجد فرق في وظيفة هذه الميثودس سوى أن **BeginGetResponse** تعتبر **asynchronous Method** .

يحتوي ال **HttpWebResponse Object** على عدد من ال **Properties** وهي :

1- **CharacterSet** : وتستخدم لتحديد نوع ال Character Set
2- **ContentEncoding** : وتستخدم لعملية ال encoding
3- **ContentLength** : وتستخدم لمعرفة حجم الرد
4- **ContentType** : لتحديد نوع ال Respone

5- **Cookies** : لتعامل مع ال Cookies ولستخدامها يجب أولا إنشاء ملف Cookie فارغ وتعريفه كما يلي كمثال :

```
HttpWebRequest hwr =
(HttpWebRequest)WebRequest.Create(http://www.amazon.com);
hwr.CookieContainer = new CookieContainer();
```

وذلك قبل ال HTTP Request ثم نسنده إليه كما يلي :

```
HttpWebResponse hwrsp = (HttpWebResponse)hwr.GetResponse();
hwrsp.Cookies = hwr.CookieContainer.GetCookies(hwr.RequestUri);
```

6- **Headers** : لمعرفة ال HTTP Headers
7- **LastModified** : يرجع فيه وقت وتاريخ أخر تعديل
8- **Method** : لمعرفة الميثود والتي تستخدم في ال HTTP Response
9 – **ProtocolVersion** : لمعرفة ال HTTP Version
10 – **ResponseUri** : ال URL الخاص بسيرفر
11 – **Server** : لمعرفة اسم السيرفر
12 – **StatusCode** : لمعرفة نوع ال Coding امستخدم
13 – **StatusDescription** : لإرجاع Text يحتوي على حالة ال HTTP

الدرس الثامن: Web Services Programming

تحدثنا في الدرس السابق عن برمجة ال HTTP وبينا فيه كيفية التفاعل بين ال web server وال client ويعتبر هذا الدرس مكمل لما تحدثنا عنه سابقا، تتلخص وظيفة استخدام ال web services بإمكانية الاستفادة من ال Methods الموجودة بال web server داخل برنامج الزبون وباستخدام برتوكول ال SOAP وهو اختصار ل Simple Object Access Protocol يتم نقل ال Result من ال web Services server إلى ال Client بعد تحويلها إلى ال XML - extensible Markup Language حيث تنقل عبر برتوكول ال HTTP إلى جهاز الزبون والهدف من استخدامه هو تسهيل وصول ال Data من ال web server إلى ال Client من خلال ال firewalls والبيئات المختلفة إذ أن جميع بيئات الشبكات تدعم برتوكول ال HTTP والذي يعمل على البورت 80 . ولا تختلف لغة ال XML عن ال HTML إذ تستخدم نفس القواعد في ال HTML وهي مجموعة من ال Elements وال Attributes مثل ال </> <> لاكن تتميز بمرونة اكبر وكمثال عليها :

```
<myStuff>
<myName>FADI Abdel-qader</myName>
<myTelephone>+962796...</myTelephone>
<myEmail>fadi822000@yahoo.com</myEmail>
<myAge>23</myAge>
<mySex>M</mySex>
</myStuff>
```

ويتم استدعائها في الدوت نيت باستخدام النيم سبيسس System.xml حيث يتم قراءتها باستخدام الميثود Load الموجود في ال XmlDocument Class كما يلي :

```
using System.Xml;
// Then you can Read any XML File as Below:
XmlDocument xDoc = new XmlDocument();
xDoc.Load(@"C:\myinfo.xml");
XmlNodeList name = xDoc.GetElementsByTagName("myName");
XmlNodeList telephone = xDoc.GetElementsByTagName("myTelephone");
XmlNodeList email = xDoc.GetElementsByTagName("myEmail");
XmlNodeList age = xDoc.GetElementsByTagName("myAge");
XmlNodeList sex = xDoc.GetElementsByTagName("mySex");

MessageBox.Show(
"Name: " + name[0].InnerText +"\n"+
"Telephone: " + telephone[0].InnerText +"\n"+
"Email: "+ email[0].InnerText +"\n"+
"Age: "+ age[0].InnerText +"\n"+
"sex: "+ sex[0].InnerText +"\n"
```

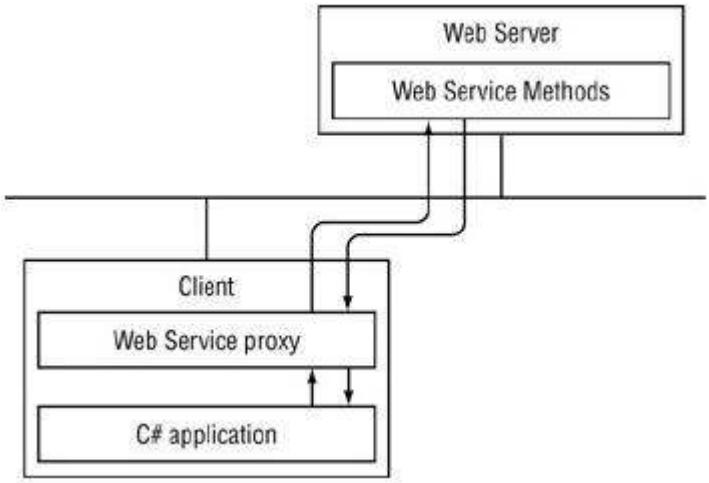
تمر عملية استخدام ال web services بثلاثة مراحل وهي :

1- The web service server : والذي يتم من خلاله إرسال واستقبال البيانات عبر برتوكول ال SOAP باستخدام ال IIS وال ASP.NET .

2- The proxy object : والذي يسمح لل Client بإرسال و استقبال البيانات من وإلى ال web Services Server حيث يتم تعريفه في ال HttpWebRequest من خلال الكلاس WebProxy وهو ما بينته في الدرس السابق.

3- The client application : وهو الواجهة الخاصة بزبون والتي يتم ربطها بال Web Services Server

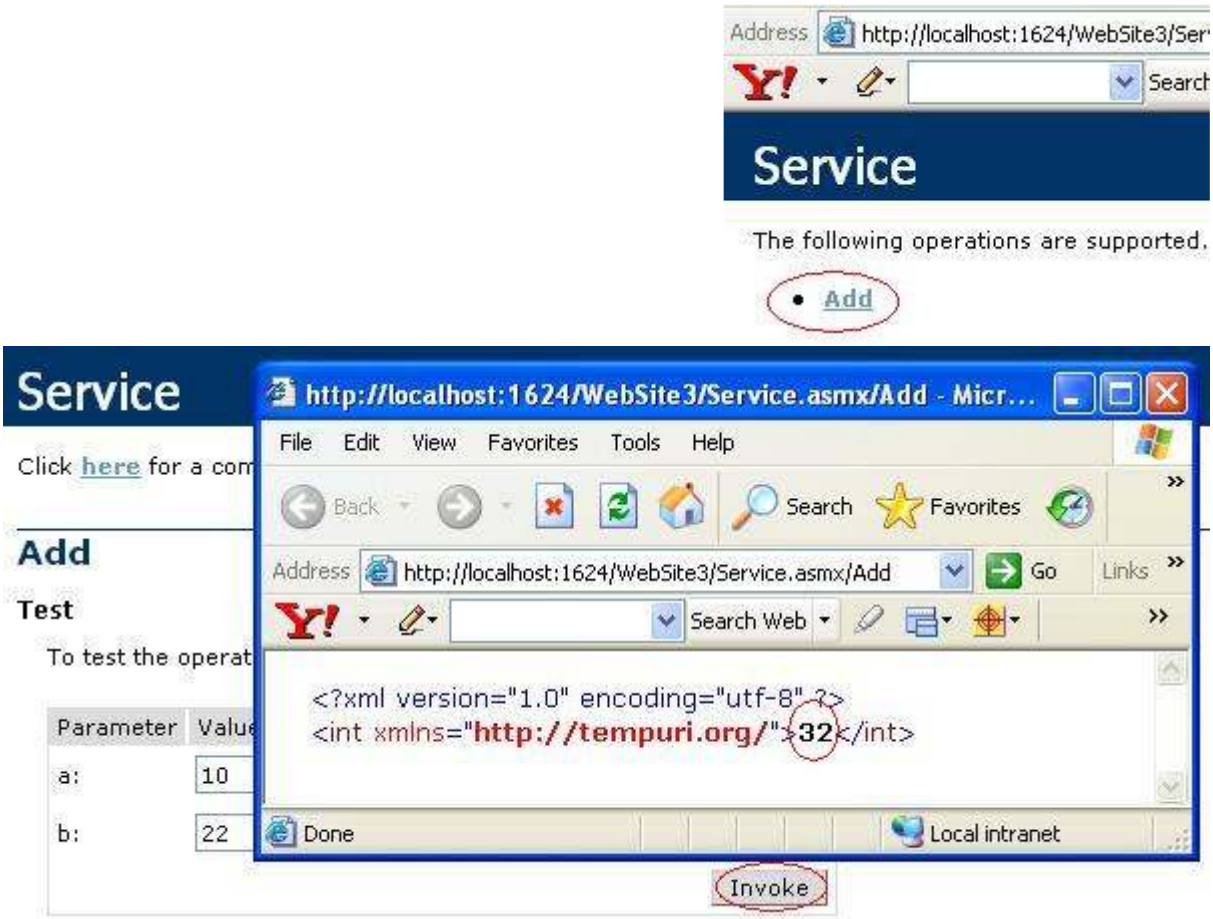
كما في الشكل التالي :



ولإنشاء web services server نقوم بعمل مشروع ASP.NET Web Services جديد ونستدعي النيم سبيس System.Web.Services ثم نقوم بتوريث الكلاس WebService للكلاس الرئيسي للمشروع وكما يلي كمثال:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Namespace = "http://my_url.com/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {}
    [WebMethod]
  public int Add(int a, int b)
  {
   return a + b;
  }

    }
```

حيث يتم استقبال قيمتين A و B وبعد ذلك يقوم بإرجاع ناتج جمع القيمة الأولى مع القيمة الثانية إلى ال Client على شكل XML باستخدام برتوكول ال SOAP وكما يظهر في الشكل التالي :



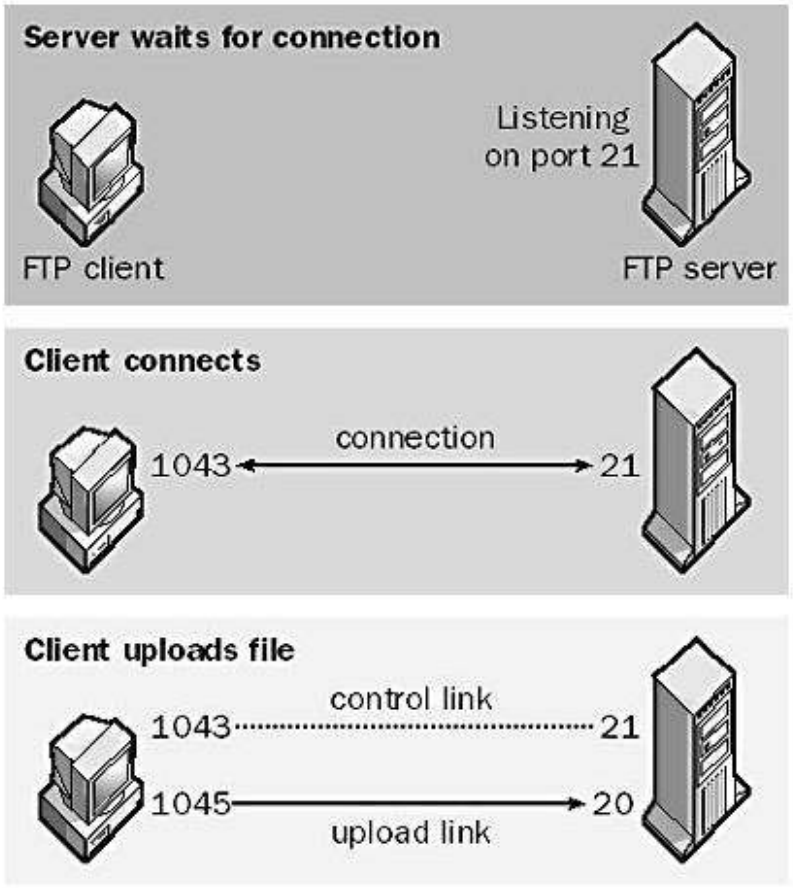
ولإنشاء برنامج ال Client يجب أولا تحويل الكلاس السابق إلى Dll File و إرفاقه بال Client Resources ويتم استخدامه كما يلي :

```
using System;
class Client_side
{
   public static void Main(string[] argv)
   {
      My_main_class mm = new My_main_class();
      int x = Convert.ToInt16(argv[0]);
      int y = Convert.ToInt16(argv[1]);
      int sum = mm.Add(x, y);
      Console.WriteLine(sum);
   }
}
}
```

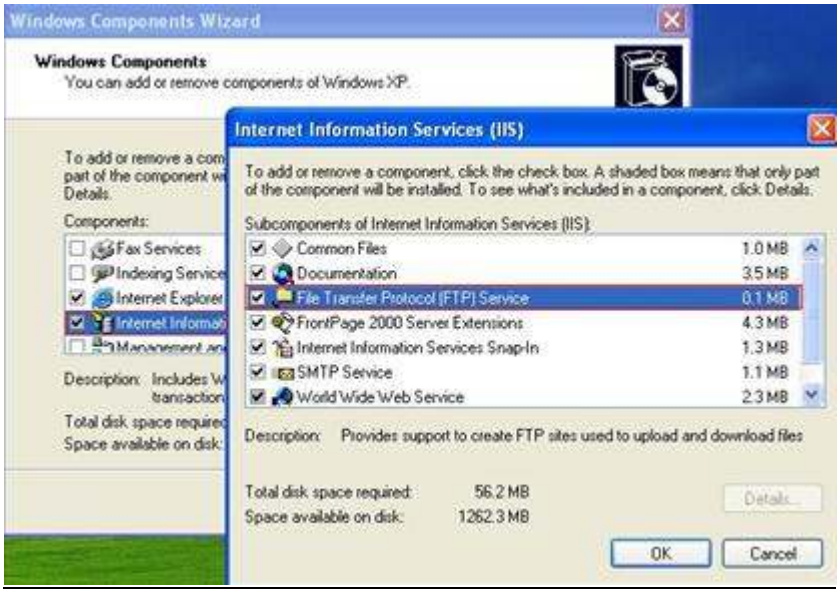
وهكذا بينا الأساسيات في ال Web services وسوف نأتي على التفاصيل لاحقا إنشاء الله.

الدرس التاسع FTP – File Transfer Protocol Programming :

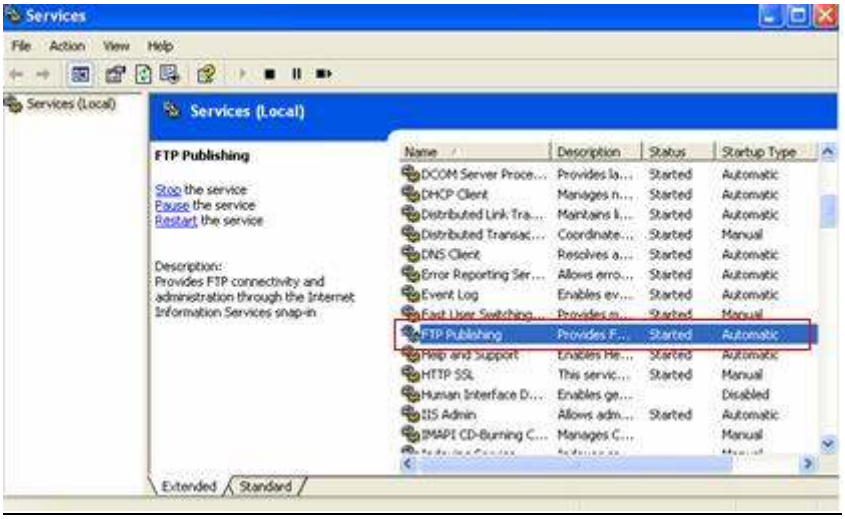
سوف نبدأ هنا بشرح برتوكول أخر من برتوكولات ال Application Layer وهو برتوكول ال FTP والذي يستخدم بشكل أساسي في عملية تنزيل downloading و رفع uploadingالملفات من و إلى ال FTP Server وكالعادة في اغلب برمجيات الشبكات و التي تعتمد على وجود Client/Server حيث يقوم السيرفر بتصنت على البورت المخصص لل FTP وهو البورت 21 باستخدام ال TCP Connection Oriented Protocolحيث يبقى السيرفر بوضع الانتظار لورود طلب من ال Client بإنشاء Session معه وبعد إجراء عمليات التحقق Authentication والتأكد من الصلاحيات يتم الموافقة على البدء بالجلسة حيث يتم تحديد رقم البورت والذي سوف يتم استقبال البيانات من خلاله ويتم الإرسال إلى جهاز الزبون عبر البورت 20 في السيرفر وتتضح هذه العملية كما في الشكل التالي :



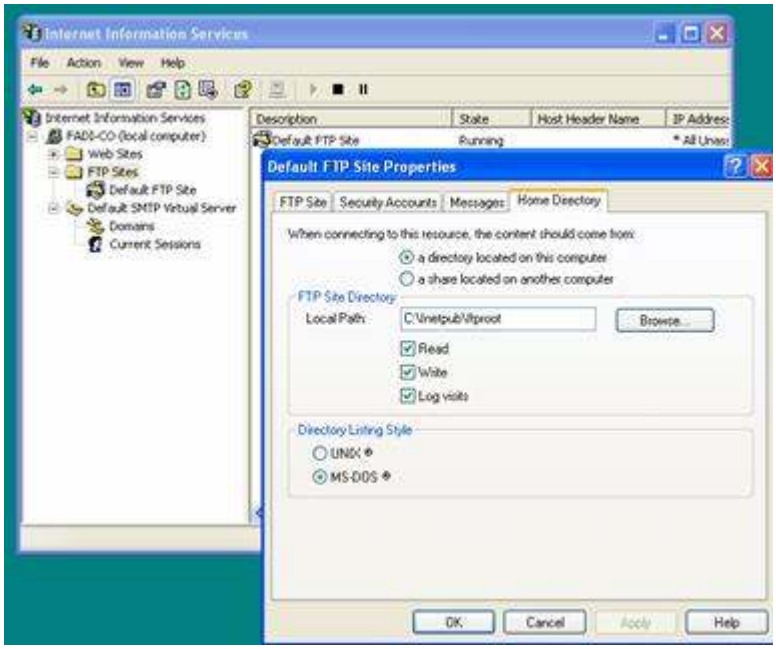
<u>ملاحظة:</u> لتفعيل خدمة ال FTP لديك بحيث يعمل جهازك ك FTP Server يجب أولا التأكد من أن ال FTP Services مثبتة لديك مع ال IIS و كما يظهر في الشكل التالي :



ومن ثم التأكد من تفعيلها ب Services من Control Panel ثم Administrative Tools ثم Services وكما يظهر في الشكل التالي :

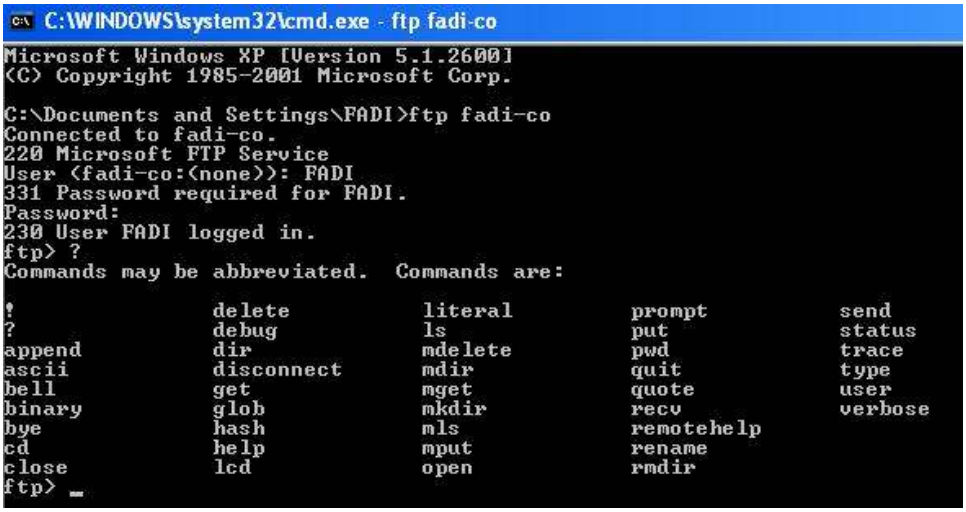


ثم التأكد منه في ال IIS بحيث يظهر كما في الشكل التالي :



أولا : FTP Commands

تشبه عملية الاتصال و الاستخدام لل FTP عملية ال Telnet إلى حد كبير حيث يدعم برتوكول ال FTP مجموعة من الأوامر والتي يتم من خلالها عملية التخاطب مع السيرفر أو مع ال Remote Host وتضح هذه العملية كما في الشكل التالي :

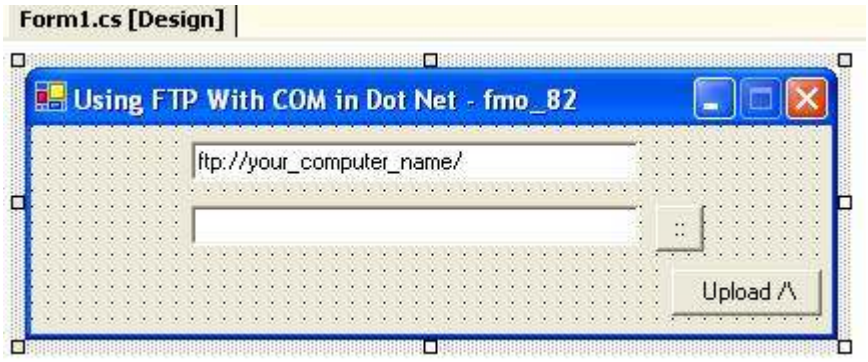


وهنا شرح لأهم ال FTP Commands:

USER <username> & PASS <password>	مطلوبة لعملية التحقق لإنشاء الجلسة
RETR <*filename*> أو RECV	ويستخدم لتنزيل ملف من السيرفر بعد تحديد اسم الملف
STOR <*filename*> أو SEND	ويستخدم لرفع الملف إلى السيرفر بعد تحديد اسم الملف
TYPE <*type indicator*>	لتحديد طبيعة أو هيئة البيانات التي يتم نقلها وكما يلي : -a ASCII –e EBCDIC – I for Binary - L<Byte Size> Data والذي سيتم نقله
PASV	لتحديد نوع الجلسة سواء Passive أو Active إذ انه في حالة ال Passive يتم تفعيل الاتصال فقط في حالة ورود أو رفع أي ملف من و إلى السيرفر .
STAT أو Status	لفحص حالة الاتصال و uploading & Downloading
Delete , cd , mkdir , rename …	وهي كما هو متعارف عليه في التعامل مع الملفات و المجلدات في نظام ال DOS
QUIT أو Close	لإنهاء الجلسة مع ال Remote Host

<u>ثانيا : التعامل مع ال FTP في الدوت نيت باستخدام COM Components :</u>

تدعم الدوت نيت استخدام ال FTP عبر ITC – Internet Transfer Control وهو جزء من ال COM Components Controls وللبدء قم بإنشاء New Windows Application كما في الشكل التالي :



ثم قم بإضافة النيم سبيسس التالية :

```
using System.IO;
using System.Reflection;
```

ثم إضافة الكود التالي إلى ال Upload Button :

```
private void button1_Click(object sender, System.EventArgs e)
{
FileInfo thisFile = new FileInfo(tbFile.Text);
Type ITC;
object[] parameter= new object[2];
object ITCObject;
ITC = Type.GetTypeFromProgID("InetCtls.Inet");
ITCObject = Activator.CreateInstance(ITC);
parameter[0] = (string)tbServer.Text;
parameter[1] = (string)"PUT " + thisFile.FullName + " /" +
thisFile.Name;
ITC.InvokeMember("execute", BindingFlags.InvokeMethod,null, ITCObject,
parameter);}
```

تم في البداية تعريف ال ITC من خلال ال Type Class والموجود ضمن النيم سبيس System.Reflection ثم عرفنا Array من النوع Object وذلك لاستخدامها في تمرير اسم الملف و ال FTP Server إلى الميثود InvokeMember والموجودة ضمن ال ITC Control Object ...

سوف تجد الملف الذي سيتم رفعه في المجلد :
C:\Inetpub\ftproot

ثالثا : التعامل مع ال FTP في الدوت نت باستخدام ال Web Class :

يمكن برمجة ال FTP باستخدام web Class والموجودة ضمن النيم سبيسس System.Net وتشبه عملية التعامل معه كما في التعامل مع ال WebRequest و ال webResponse Classes و التي تعاملنا معها في برمجة ال HTTP حيث يمكننا الاستفادة منها لتعامل مع ال FTP Protocol وهي كما يلي :

- WebClient إذ تم دعم dot net Framework 2 استخدام الكلاس WebClient والذي يدعم التعامل مع ال FTP والذي يتم استدعائه من النيم سبيسس System.Net ويتم تعريفه كما يلي :

```
using System;
using System.Net;

namespace Web_Client
{
   class Program
   {
      public static void Main(string[] args)
      {
         string filename = "ftp://ms.com/files/dotnetfx.exe";
         WebClient client = new WebClient();
         client.DownloadFile(filename, "dotnetfx.exe");

      }

   }
}
```

- FtpRequestCreator ويستخدم لتسجيل وبدأ العمل مع ال FTP ويعرف كما يلي :

```
using System;
using System.Net;

namespace FTP
{
   public class FtpRequestCreator : IWebRequestCreate
   {
      public FtpRequestCreator()
      {
      }

      public System.Net.WebRequest Create(System.Uri uri)
      {
         return new FtpWebRequest(uri);
      }
   }
}
```

- FtpWebRequest ويستخدم لعمل download or upload a file on an FTP server ويتم تعريفها كما يلي :

```
using System;
using System.Net;

namespace FTP
{
   public class FtpWebRequest : WebRequest
   {
      private string username = "Fadi";
      internal string password = "fff";
      private Uri uri;
      private bool binaryMode = true;
      private string method = "GET";

      internal FtpWebRequest(Uri uri)
      {
         this.uri = uri;
      }

      public string Username
      {
         get { return username; }
         set { username = value; }
      }

      public string Password
      {
         set { password = value; }
      }
```

```
        public bool BinaryMode
        {
            get { return binaryMode; }
            set { binaryMode = value; }
        }

        public override System.Uri RequestUri
        {
            get { return uri; }
        }

        public override string Method
        {
            get { return method; }
            set { method = value; }
        }

        public override System.Net.WebResponse GetResponse()
        {
            FtpWebResponse response = new FtpWebResponse(this);
            return response;
        }
    }
}
```

- FtpWebResponse ويستخدم لعملية الرد من قبل السيرفر ويتم تعريفها كما يلي:

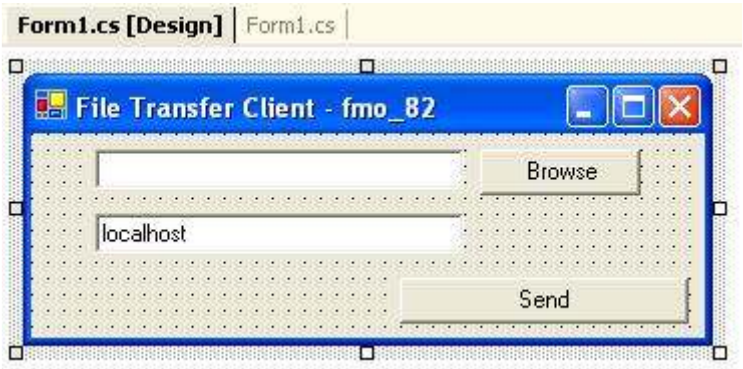
```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;

namespace FTP
{
    public class FtpWebResponse : WebResponse
    {
        private FtpWebRequest request;
        private FtpClient client;

        internal FtpWebResponse(FtpWebRequest request)
        {
            this.request = request;
        }
    }
}
```

- FtpWebStream ويستخدم لتعريف ال Stream والذي سوف يستخدم لعملية النقل ويعرف بشكل مبدئي كما يلي :

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

namespace FTP
{
   internal class FtpWebStream : Stream
   {
      private FtpWebResponse response;
      private NetworkStream dataStream;

 public FtpWebStream(NetworkStream dataStream, FtpWebResponse
response)
      {
         this.dataStream = dataStream;
         this.response = response;
      }
   }
}
```

رابعا : مثال تطبيقي لرفع ملف من جهاز Client إلى جهاز Server باستخدام ال Stream وال Socket:

في هذا الجزء سوف نقوم بإنشاء برنامجين Client / Server وبتعامل مع ال Stream Library سوف نقوم بتحويل الملف إلى Byte Array و إرساله عبر ال Stream باستخدام ال Socket و TCP Connection ، ولبرمجة الجزء الخاص بالإرسال أو ال Client قم بإنشاء مشروع جديد كما في الشكل التالي :



سوف نستخدم النيم سبيس التالية :

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

في ال Send Button قم بكتابة الكود التالي :

```
try
{
Stream fileStream = File.OpenRead(textBox1.Text);
```

```
// Alocate memory space for the file
byte[] fileBuffer = new byte[fileStream.Length];
fileStream.Read(fileBuffer, 0, (int)fileStream.Length);
// Open a TCP Connection and send the data
TcpClient clientSocket = new TcpClient(textBox2.Text,8880);
NetworkStream networkStream = clientSocket.GetStream();
networkStream.Write(fileBuffer,0,fileBuffer.GetLength(0));
networkStream.Close();
}
catch (Exception ex){MessageBox.Show(ex.Message);}
```

قمنا في البداية بقراءة الملف الذي نود إرساله وتخزينه ب Stream Object وحتى نستطيع إرساله عبر ال Socket لابد من تحويله إلى مصفوفة من النوع Byte وقمنا بتسميته ب fileBuffer ثم تعبئته باستخدام الميثود Read والموجودة ضمن fileStream وبعد ذلك قمنا بإنشاء اتصال مع ال Server باستخدام ال TCP Connection حيث تم إرسال محتويات ال fileBuffer إلى ال Server باستخدام NetworkStream Class ...

ولبرمجة جزء Server وهو المسئول عن استقبال الملف وتخزينه قم بإنشاء مشروع جديد كما يظهر في الشكل التالي :

سوف نستخدم النيم سبيسس التالية :

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

ثم قم بكتابة ميثود جديدة وليكن اسمها handlerThread وكما يلي :

```
public void handlerThread()
{
Socket handlerSocket = (Socket)alSockets[alSockets.Count-1];
NetworkStream networkStream = new
NetworkStream(handlerSocket);
```

```
int thisRead=0;
int blockSize=1024;
Byte[] dataByte = new Byte[blockSize];
lock(this)
{
// Only one process can access
// the same file at any given time
Stream fileStream = File.OpenWrite(@"c:\upload");

while(true)
{
thisRead=networkStream.Read(dataByte,0,blockSize);
fileStream.Write(dataByte,0,thisRead);
if (thisRead==0) break;
fileStream.Close();
}
lbConnections.Items.Add("File Written");
handlerSocket = null;
}
```

ثم قم بكتابة ميثود أخرى جديدة وذلك لفتح TCP Connection على البورت 8880 وهو افتراضي والتصنت عليها وليكن اسمها listenerThread وكما يلي :

```
public void listenerThread()
{
80);8TcpListener tcpListener = new TcpListener(8
tcpListener.Start();
while(true)
{
Socket handlerSocket = tcpListener.AcceptSocket();
if (handlerSocket.Connected)
{
lbConnections.Items.Add(handlerSocket.RemoteEndPoint.ToString() +
" connected.");
lock (this)
{
alSockets.Add(handlerSocket);
}
ThreadStart thdstHandler = new
ThreadStart(handlerThread);
Thread thdHandler = new Thread(thdstHandler);
thdHandler.Start();
}
}
}
```

ثم قم بإضافة الكود التالي إلى حدث بدأ تشغيل البرنامج Form Load :

```
private void Form1_Load(object sender, System.EventArgs e)
{
IPHostEntry IPHost = Dns.GetHostByName(Dns.GetHostName());

lbConnections.Text = "My IP address is " +
IPHost.AddressList[0].ToString();

alSockets = new ArrayList();

Thread thdListener = new Thread(new ThreadStart(listenerThread));
thdListener.Start();

}
```

باستخدام ال Thread تم تنفيذ ال listenerThread Method والتي قمنا فيها بتعريف ال tcpListener وتفعيله على البورت 8880 حيث سيتم قبول أي طلب يأتي من ال Client على هذا البورت وبعد ذلك استدعاء الميثود handlerThread والتي سيتم فيها استقبال ال Stream Data وتخزينها في Byte Array ثم قراءتها وتخزينها في المكان المحدد وباستخدام ال fileStream.Write حيث مررنا له ال Stream والذي يحتوي على اسم الملف thisRead وال dataByte Array ...

End of Chapter 1

وهنا قد تم الانتهاء من ال Chapter1 وفي الجزء التالي من هذه السلسلة سوف نتحدث عن أمور أكثر تقدما في برمجة الشبكات وسوف نأتي على بعض الجزيئيان ونفصلها بشكل اكبر إنشاء الله مثل :

Stream Libraries
Network Layer Programming
Remotting
Threading and the Asynchronous Pattern
Security & Authentications
...

Fadi822000@yahoo.com
http://spaces.msn.com/members/csharp2005