Chapter 15
Remotting & Distributed Systems Programming & Design

- The Distributed Systems Concept & Design
    - Design a Distributed eCommerce System by Using ASP.NET and Web Services.

- Serialization Programming
    - The Serialization (Classes & Members)
    - Using BinaryFormatter & SoapFormatter to Serialize Objects & Images Throw Network

- Remotting Programming
    - Remotting (Classes & Members)
    - Using Remotting Applications in Dot Net
    - Create an Advanced Distributed eLearning System (Remote Class Room)
    - Create an Advanced Remote Desktop Application With Remote (Mouse/Keyboard) Control Features

بسم الله الرحمن الرحيم

أولا - The Distributed Systems Concept & Design :

فقط في النسخة الورقية من الكتاب

www.fadidotnet.org

ثانيا - The Serialization Programming :

The Serialization Classes & Members : يمكننا من خلال الـSerialization عمل Serializing لـ Object من جهاز إلى آخر سواء عبر الشبكة المحلية باستخدام الـ Stream Socket أو عبر الإنترنت باستخدام الـ Web Services والـRemotting ، ويمكننا استخدام الـ Serialization في الدوت نيت من خلال الـ System.Runtime.Serialization Namespace حيث يحتوي هذا الـ Namespace على عدد ضخم من الـ Classes والتي تساعد في عمل Serialization لأي Object عبر الإنترنت أو الشبكة المحلية، ويبين الجدول التالي أهم هذه الـ Classes والتي يمكننا الاستفادة منها في بناء النظم الموزعة:

Formatters , Formatter & FormatterServices	ويحتوي هذا الـ Class على كل الوظائف الرئيسية لعمل Serialization لأي Object ومنها يستخدم الـ Serialize Method لإرسال Object Serializing عبر الـ Stream والـ Deserialize والتي تستخدم للاستقبال الـ Serializing Object من الـStream و تحويله إلى Object مرة أخرى. وتأخذ عملية الإرسال نوعين هما الإرسال عبر بروتوكول الـ Soap ويأتي ضمن الـ Namespace *System.Runtime.Serialization.Formatters.Soap* حيث يجب إضافته إلى الـ Reference للمشروع وقد بينا في الفصل السابق كيفية عمل بروتوكول الـ SOAP حيث يحول الـ Object إلى XML Serial Stream ، والطريقة الثانية هي بتحويل الـObject إلى Binary Serial Stream ويأتي ضمن الـNamespace : *System.Runtime.Serialization.Formatters.Binary* والذي سيتم شرحه بالتفصيل في الأمثلة القادمة في هذا الفصل.
FormatterConverter	حيث يحتوي على الـ IFormatter Interface والـ IConvertible Interface لأستخدمهما في عمليات تمثيل الـ Object حتى يمكن إرساله عبر الـ Serializing Stream ...
ObjectIDGenerator	ويستخدم لتوليد رقم ID عشوائي لأي Object حيث يحتوي هذا الـ Class على Tow Methods الأول يقوم بتوليد رقم تسلسلي للـ Object يسمى GetID والثاني يرجع قيمة الـ ID الخاص بالـ Object ويسمى HasId ويأخذ كل منهما اسم الـ Object الذي نريد توليد رقم تسلسلي له وقيمة True أو False لتحديد فيما إذا كانت هذه هي المرة الأولى التي تم فيها توليد رقم تسلسلي للـ Object أم لا... GetId (object_name,out bool) →to Set an ID HasId(object_name,out bool) → to Return the ID
SerializationInfo	ويحتوي على كل المعلومات التي يمكن أن نحتاج لها لعملية الـ Serialize و الـ Deserialize لأي Object مثلا AssemblyName و الـ FullTypeName والـ MemberCount وتحويلات الـ Data Types للـ Objects وغيرها...

لإنشاء أي Serializable Class يجب أولا أن يعرف بالـ Attribute [Serializable] حيث يعرف أن هذا الـ Class من الـ Classes التي يمكن أن يتم عمل Serializing عليها وكما يلي كمثال لعمل Serializable Class للـ Bank_Tier الخاص بالمثال السابق:

C#:

```
[Serializable]
public class Bank_Tier
{
public string Customer_Name;
public string Address;
public int zip_code;
public string payment_method;
public string ecard_number;
public float depositor;
public string acount_number;

public Bank_Tier()
        {
        // Get_Session_ID();
        }
}
```

VB.NET:

```
<Serializable()> _
Public Class Bank_Tier
   Public Customer_Name As String
   Public Address As String
   Public zip_code As Integer
   Public payment_method As String
   Public ecard_number As String
   Public depositor As Single
   Public acount_number As String

   Public Sub New()
      ' Get_Session_ID()
   End Sub
End Class
```

ولإنشاء البرنامج الخاص بعملية الإرسال لابد من تحديد طريقة الإرسال سواء كـ Binary Stream Data أو XML Data ففي الأول سنستخدم الـ BinaryFormatter والموجود ضمن الـ System.Runtime.Serialization.Formatters.Binary Namespace أما الثاني فسيستخدم مع الـ SOAP Protocol والمعرف ضمن الـ SOAPFormatter Class والموجود ضمن الـSystem.Runtime.Serialization.Formatters.Soap Namespace .

ولعمل Serialization للـ Class السابق باستخدام الـ SOAP Formatter لابد أولا من تعريف الـ Namespaces التالية:

System.Runtime.Serialization
System.Runtime.Serialization.Formatters.Soap

ثم نقوم بعمل Instance من الـ Bank Web Services حيث نستطيع فيما بعد إسناد المعلومات الآتية من الـ Presentation Tier إليه ثم إرسالها إلى الـ Bank Server باستخدام الـ Remotting أو الـNetworkStream أو حفظها على الجهاز كـ Binary Data أو XML File وكما يلي كمثال:

لتخزين المخرجات بـ XML File باستخدام بروتوكول الـ SOAP :

C#:

```
Bank_Tier bt = new Bank_Tier();
bt.acount_number = acount_number;
bt.Address = Address;
bt.depositor=depositor;
bt.ecard_number = ecard_number;
bt.payment_method = payment_method;
bt.zip_code = zip_code;

Stream str = new FileStream("bank_result.xml", FileMode.Create,
FileAccess.ReadWrite);

IFormatter formatter = new SoapFormatter();
formatter.Serialize(str, bt);
str.Close();
```

VB.NET:

```
Dim bt As Bank_Tier = New Bank_Tier
bt.acount_number = acount_number
bt.Address = Address
bt.depositor=depositor
bt.ecard_number = ecard_number
bt.payment_method = payment_method
bt.zip_code = zip_code

Dim str As Stream = New FileStream("bank_result.xml", FileMode.Create,
FileAccess.ReadWrite)

Dim formatter As IFormatter = New SoapFormatter
formatter.Serialize(str, bt)
str.Close()
```

وسكون الـ Output ملف XML يحتوي على البيانات التي تم إدخالها وكما يلي:

```
- <SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
- <SOAP-ENV:Body>
- <a1:Form1_x002B_Bank_Tier id="ref-1"
    xmlns:a1="http://schemas.microsoft.com/clr/nsassem/Serialization/Serialization
    %2C%20Version%3D1.0.2309.39316%2C%20Culture%3Dneutral%2C%20Publ
    icKeyToken%3Dnull">
  <Customer_Name xsi:null="1" />
  <Address id="ref-3">Amman Jordan</Address>
  <zip_code>962</zip_code>
  <payment_method id="ref-4">Master Card</payment_method>
  <ecard_number id="ref-5">66-6655-444433-55655</ecard_number>
  <depositor>600</depositor>
  <acount_number id="ref-6">3399948845</acount_number>
```

```
</a1:Form1_x002B_Bank_Tier>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

وبتأكيد نستطيع استخدام هذا المثال لإرسال البيانات عبر الـ Network Stream أو باستخدام الـ Remotting ، كما يمكننا الاستفادة من الـ Serialization لتمثيل أي Object على شكل XML أو Binary Data لاستخدامها في أمور أخرى...

سنبين في المثال التالي كيفية الاستفادة من الـ BinaryFormatter Class لإرسال Object أو Binary Data من جهاز إلى آخر عبر الشبكة كما سنبين كيفية الاستفادة من الـ SOAP Protocol لإرسال Binary Data كـ XML Stream:

وكمثال نريد إرسال صورة من جهاز إلى آخر بعد تحويلها إلى Binary Stream باستخدام الـ MemoryStream وكما يلي:

<u>أولا : باستخدام الـ BinaryFormatter في برنامج الإرسال:</u>

<u>C#:</u>

```
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization;
.
.
private void Send_Image_Using_BinaryFormatter
{
MemoryStream ms = new MemoryStream ();
pictureBox1.Image.Save (ms,System.Drawing.Imaging.ImageFormat.Jpeg);
object op = (object) ms;

BinaryFormatter br = new BinaryFormatter ();

TcpClient myclient = new TcpClient ("localhost",5000);
NetworkStream myns = myclient.GetStream ();
br.Serialize (myns,op);

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

<u>VB.NET</u>

```
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Runtime.Serialization
.
.
Private Property Send_Image_Using_BinaryFormatter() As void
Dim ms As MemoryStream = New MemoryStream ()
pictureBox1.Image.Save (ms,System.Drawing.Imaging.ImageFormat.Jpeg)
Dim op As Object = CObj(ms)

Dim br As BinaryFormatter = New BinaryFormatter

Dim myclient As TcpClient = New TcpClient("localhost", 5000)
```

```
Dim myns As NetworkStream = myclient.GetStream()
br.Serialize (myns,op)

myns.Close ()
myclient.Close ()
End Property
```

استخدام الـ BinaryFormatter في برنامج الاستقبال:

ولاستخدام بروتوكول الـ BinarFormatter في برنامج الاستقبال سنستخدم الميثود المعاكسة للـ Serialize وهي Deserialize ثم نقوم بعمل Casting للـ Object المستقبل ونحوله إلى MemoryStream Object مرة أخرى ، عندها نستطيع عرضه على الـ Picture box باستخدام الـ FromStream Method والموجودة ضمن الـ Image Class :

C#:

```
void Image_Receiver()
{

NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
Thread myth;

mytcpl = new TcpListener (5000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
BinaryFormatter br = new BinaryFormatter ();
object op;

op= br.Deserialize (myns); // Deserialize the Object from Stream
MemoryStream mm = (MemoryStream) op; // Casting The Object to
MemoryStream Object
pictureBox1.Image = Image.FromStream(mm);
mytcpl.Stop();
if (mysocket.Connected ==true)
        {while (true){Image_Receiver();}}
}
.
.
private void server_Load(object sender, System.EventArgs e)
{
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Image_Receiver));
myth.Start ();
}
```

VB.NET

```
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Runtime.Serialization
.
.
Private Sub Image_Receiver()

    Dim myns As NetworkStream
    Dim mytcpl As TcpListener
    Dim mysocket As Socket
    Dim myth As Thread

    mytcpl = New TcpListener(5000)
    mytcpl.Start()
    mysocket = mytcpl.AcceptSocket()
    myns = New NetworkStream(mysocket)
    Dim br As BinaryFormatter = New BinaryFormatter
    Dim op As Object

    op = br.Deserialize(myns) ' Deserialize the Object from Stream
    Dim mm As MemoryStream = CType(op, MemoryStream) ' Casting The
Object to MemoryStream Object

    pictureBox1.Image = Image.FromStream(mm)
    mytcpl.Stop()

    If mysocket.Connected = True Then
        Do While True
            Image_Receiver()
        Loop
    End If
End Sub
.
.
Private Sub server_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim myth As Thread
    myth = New Thread(New System.Threading.ThreadStart(AddressOf
Image_Receiver))
    myth.Start()
End Sub
```

ثانيا : استخدام الـ SOAP في برنامج الإرسال:

حيث سترسل الصورة كـ XML Data ثم تحول في الطرف المقابل إلى صورة مرة أخرى وكل ذلك يتم باستخدام بروتوكول الـ SOAP وكما يلي في برنامج الإرسال:

C#:

```
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization;
.
.
private void Send_Image_Using_ SoapFormatter
{
server fr = new server ();
fr.Show();

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

TcpClient myclient = new TcpClient ("localhost",5000);
NetworkStream myns = myclient.GetStream ();

IFormatter formatter = new SoapFormatter();
formatter.Serialize(myns, op);
myns.Close();
}
```

VB.NET:

```
Imports System.Runtime.Serialization.Formatters.Soap
Imports System.Runtime.Serialization
.
.
Private void Property SoapFormatter() As Send_Image_Using_
Dim fr As server = New server
fr.Show()

Dim ms As MemoryStream = New MemoryStream
pictureBox1.Image.Save (ms,System.Drawing.Imaging.ImageFormat.Jpeg)
Dim op As Object = CObj(ms)

Dim myclient As TcpClient = New TcpClient("localhost", 5000)
Dim myns As NetworkStream = myclient.GetStream()

Dim formatter As IFormatter = New SoapFormatter
formatter.Serialize(myns, op)
myns.Close()
End Property
```

استخدام الـ SoapFormatter في برنامج الاستقبال:

ولاستخدام بروتوكول الـ SOAP في برنامج الاستقبال سنستخدم الميثود المعاكسة للـ Serialize وهي Deserialize ثم نقوم بعمل Casting للـ Object المستقبل ونحوله إلى MemoryStream Object مرة أخرى ، عندها نستطيع عرضه على الـ Picture box باستخدام الـ FromStream Method والموجودة ضمن الـ Image Class :

C#:

```
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization;
.
.
void Image_Receiver()
{
NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;

mytcpl = new TcpListener (5000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);

object op;
IFormatter formatter = new SoapFormatter();
op= formatter.Deserialize(myns);
myns.Close();

MemoryStream mm = (MemoryStream) op;
pictureBox1.Image = Image.FromStream(mm);

mytcpl.Stop();

        if (mysocket.Connected ==true)
        {
        while (true)
                {Image_Receiver();}
        }
}
.
.
private void server_Load(object sender, System.EventArgs e)
{
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Image_Receiver));
// Start Thread Session
myth.Start ();
}
```

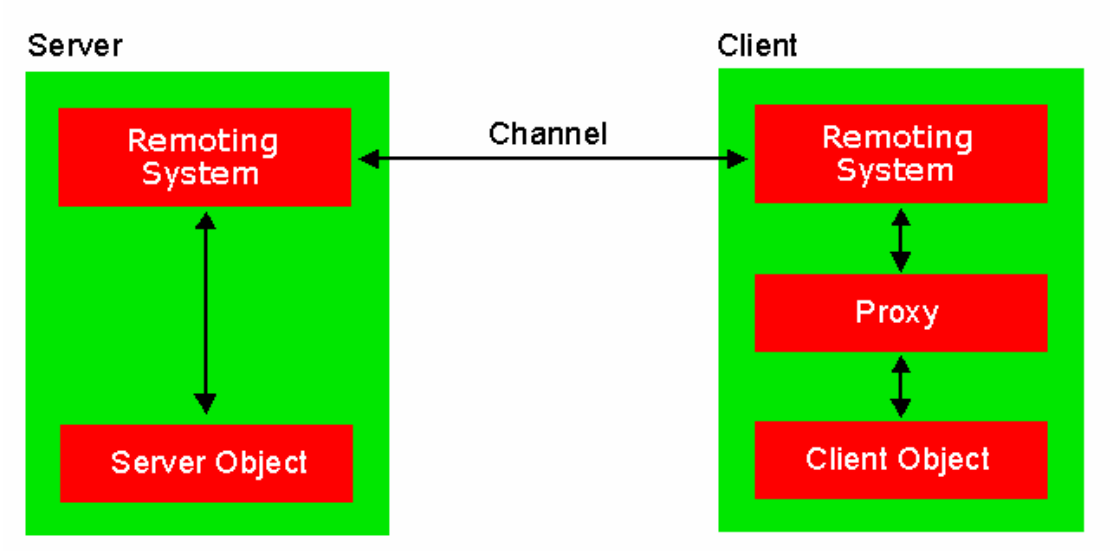
VB.NET:

```
Imports System.Runtime.Serialization.Formatters.Soap
Imports System.Runtime.Serialization
.
.

Private Sub Image_Receiver()

    Dim myns As NetworkStream
    Dim mytcpl As TcpListener
    Dim mysocket As Socket

    mytcpl = New TcpListener(5000)
    mytcpl.Start()
    mysocket = mytcpl.AcceptSocket()
    myns = New NetworkStream(mysocket)

    Dim op As Object
    Dim formatter As IFormatter = New SoapFormatter
    op = formatter.Deserialize(myns)
    myns.Close()

    Dim mm As MemoryStream = CType(op, MemoryStream)
    pictureBox1.Image = Image.FromStream(mm)

    mytcpl.Stop()

    If mysocket.Connected = True Then
        Do While True
            Image_Receiver()
        Loop
    End If
End Sub
.
.
Private Sub server_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim myth As Thread
    myth = New Thread(New System.Threading.ThreadStart(AddressOf
Image_Receiver)) ' Start Thread Session
    myth.Start()
End Sub
```

وهكذا بينا كيفية استخدام الـ Serialization في بيئة الدوت نيت بنوعيها الـ Binary والـXML Data ، سيتم الحديث في الجزء التالي من هذا الفصل عن الـ Remotting واستخدامها في برمجيات الشبكات والنظم الموزعة...

ثالثا: Remotting Programming :

يعتبر موضوع الـ Remotting من المواضيع الهامة جدا في برمجة الشبكات ، إذ يقدم لنا إمكانيات رائعة في مشاركة و استخدام الـ Remote Methods والموجودة على الـ Remote Server وتشبه عملية التعامل مع الـ Remotting عملية الـ Web Services إلى حد كبير حيث يتطلب استخدامها عمل Proxy Object في طرف الـClient، لاكن مع إمكانيات اكبر حيث تقوم بإنشاء الـServer بنفسك ولست مضطرا لتثبيت الـ IIS أو غيره من التطبيقات الوسيطة للاستفادة من خدمات الـ Remote Server ، وتستطيع من خلاله تنفيذ أي عملية معالجة على الـ Server مما يعزز من مبدأ الـ Distributed Systems والذي شرحناه في الجزء الأول من هذا الفصل حيث تبقى الـ methods و عملية المعالجة الخاصة بها على الـ Server في حين يتم إرسال الـ Result إلى الـ Clients كـ XML Data أو Binary Data حسب قناة الاتصال المستخدمة سواء HttpChannel أو TcpChannel ، وتمرر إلى الـ Client عبر الـ Proxy Object والذي ربط الـ Remote Class الموجود على الـ Server مع الـ Client لاحظ الشكل التالي:



1- Using Remotting in Dot Net :

يمكن استخدام الـ Remotting بطريقتين الأولى عبر بروتوكول الـ HTTP حيث تتم عملية التراسل كـ XML Data بين الـ Server والـ Clients وتتم باستخدام بروتوكول الـ SOAP أما الطريقة الثانية فتتم باستخدام الـ TCP Protocol حيث تكون عملية التراسل كـ Binary Data بين الـ Server وبقية الـ Clients ، ويفضل دائما استخدام الـ TCP Channel في الحالتين التاليتين:

الأولى: إذا كانت قناة الاتصال سريعة نسبيا

الثانية: توافقية البيئات حيث تستخدم نظام التشغيل Microsoft Windows في جميع الـ clients ، ويعتبر هذه الأمر من أهم عيوب استخدام الـ TCP Channel وخاصة في حالة الـWeb حيث لا تضمن البيئة التي سيعمل عليها الـ Client ويفضل في هذه الحالة استخدام الـ HTTP Channel ، لاكن تبقى أدائية الـ TCP Channel أفضل من الـ HTTP ...

وما يميز الـ HTTP Channel هو استخدامه بروتوكول الـ SOAP والذي يتعامل مع الـXML في عملية التراسل مما يضمن توافقيته مع جميع البيئات.

يلزم استخدام الـ Remotting في الدوت نيت ثلاثة أمور :

أولا، تحديد طبيعة القناة المستخدمة سواء عبر الـ HTTP كـXML أو عبر الـ TCP كـBinary Formatter ودعمت الدوت نيت كلا منهما في الـHttpChannel Class والموجود ضمن System.Runtime.Remoting.Channels.Http Namespace والـTcpChannel Class والموجود ضمن الـ System.Runtime.Remoting.Channels.Tcp Namespace (يجب إضافته إلى الـ References في المشروع)، حيث يجب تعريفها في كلا الطرفين الـ Client والـ Server وكما يلي:

C#:

```
HttpChannel http_Channel = new HttpChannel(Port_Number);
```

VB.NET:

```
Dim http_Channel As HttpChannel = New HttpChannel(Port_Number)
```

وفي حالة استخدام الـ TcpChannel كما يلي:

C#:

```
TcpChannel Tcp_Channel = new TcpChannel(Port_Number);
```

VB.NET:

```
Dim Tcp_Channel As TcpChannel = New TcpChannel(Port_Number)
```

بعد ذلك يجب تسجيل القناة المستخدمة باستخدام الميثود RegisterChannel والموجودة ضمن الـ ChannelServices Class ضمن الـNamespace System.Runtime.Remoting.Channels ، ويتم ذلك في كلا الطرفين أيضا الـ Client والـ Server وكما يلي:

C#:

```
ChannelServices.RegisterChannel(chan);
```

VB.NET:

```
ChannelServices.RegisterChannel(chan)
```

بعد ذلك يجب تعريف الـ Class الذي سيتم مشاركته على الـ Server ويتم ذلك باستخدام الميثود RegisterWellKnownServiceType والموجودة ضمن الـ RemotingConfiguration Class والموجود ضمن الـ System.Runtime.Remoting Namespace ويتم ذلك كما يلي في طرف الـ Server:

C#:

```
RemotingConfiguration.RegisterWellKnownServiceType (Type.GetType
("Class_File_Name, Distributed_Class_Name"), "unique_Service_Name",
WellKnownObjectMode.SingleCall);
```

VB.NET:

```
RemotingConfiguration.RegisterWellKnownServiceType (Type.GetType
("Class_File_Name, Distributed_Class_Name"), "unique_Service_Name",
WellKnownObjectMode.SingleCall)
```

وتأخذ الـ RegisterWellKnownServiceType Method ثلاثة بارومیترات الأول الـType الخاص باسم الـ Remote Class مع اسم الـ Dll File ومساره والذي نريد توزيعه ويأخذ البارومیتر الثاني اسم الـ URI الخاص بالـ Object الذي نريد توزيعه ويجب أن يكون الـ URI Object اسم فريد unique على مستوى الـ Application Port ، ويأخذ البارومیتر الثالث نوع الـ Remote Instance Object فإذا تم اختيار SingleCall فهذا يعني إنشاء New Instance جديد لكل Client يقوم بإنشاء Instance من الـ Remote Class ،

وإذا تم اختيار Singleton فهذا يعني استخدام نفس الـ Remote Instance Object لكل الـ Clients ، وينصح استخدام SingleCall في حالة كان النظام الموزع سيستخدم Session لكل User وينصح باستخدام Singleton في حالة كان الـ Server سيبث نفس المعلومات إلى كل الـ Clients على الشبكة...

أما في الـ Client فيجب تعريف نوع الـ Channel سواء TCP أو HTTP Channel حسب نوع القناة التي تم استخدمها في الـ Server ولا يتم وضع رقم الـ Port في الـ Client Channel ونكتفي بوضعه في الـ URI Object ويتم تعريف القناة في الـ Client كما يلي:

C#:
```
HttpChannel http_Channel = new HttpChannel
```

VB.NET:
```
Dim http_Channel As HttpChannel = New HttpChannel()
```

وفي حالة استخدام الـ TcpChannel كما يلي:

C#:
```
TcpChannel Tcp_Channel = new TcpChannel();
```

VB.NET:
```
Dim Tcp_Channel As TcpChannel = New TcpChannel()
```

ويتم تسجيل القناةRegisterChannel في الـ Client كما هو الحال في الـ Server وكما يلي:

C#:
```
ChannelServices.RegisterChannel(chan);
```

VB.NET:
```
ChannelServices.RegisterChannel(chan)
```

بعد ذلك نقوم بتعريف New Instance Object من الـ Remote Client حيث يجب أن يحتوي الـ Client Project على الـ Dll File الخاص بالـ Distributed Class حيث نقوم في البداية بتعريف الـ URI والذي سيحتوي على الـ Protocol المستخدم ويتبع بعنوان الجهاز أو الـ DNS الخاص بالـ Server ومن ثم رقم الـ Port المستخدم و اسم الخدمة التي تم تعريفها في الـ Server ويتم ذلك كما يلي:

في حالة استخدام TCP Channel :

C#:
```
string URI = "Tcp://"+Remote_IP.Text+":Port/ unique_Service_Name ";
```

VB.NET:
```
Dim URI As String = "Tcp://" & Remote_IP.Text & ": Port/
unique_Service_Name "
```

في حالة استخدام HTTP Channel :

C#:
```
string URI = "Http://"+Remote_IP.Text+": Port / unique_Service_Name";
```

VB.NET:
```
Dim URI As String = " Http://" & Remote_IP.Text & ": Port /
unique_Service_Name "
```

وبعد ذلك نقوم بإنشاء New Instance Object من الـ Remote Class ويتم ذلك باستخدام الـ GetObject Method والموجود ضمن الـ Activator Class ونمرر لها الـ Distributed Class Type و الـ URI الذي عرفناه ، وكما يلي:

C#:

```
Distributed_Class_Name obj = (Distributed_Class_Name) Activator.GetObject
(typeof (Distributed_Class_Name), URI);
```

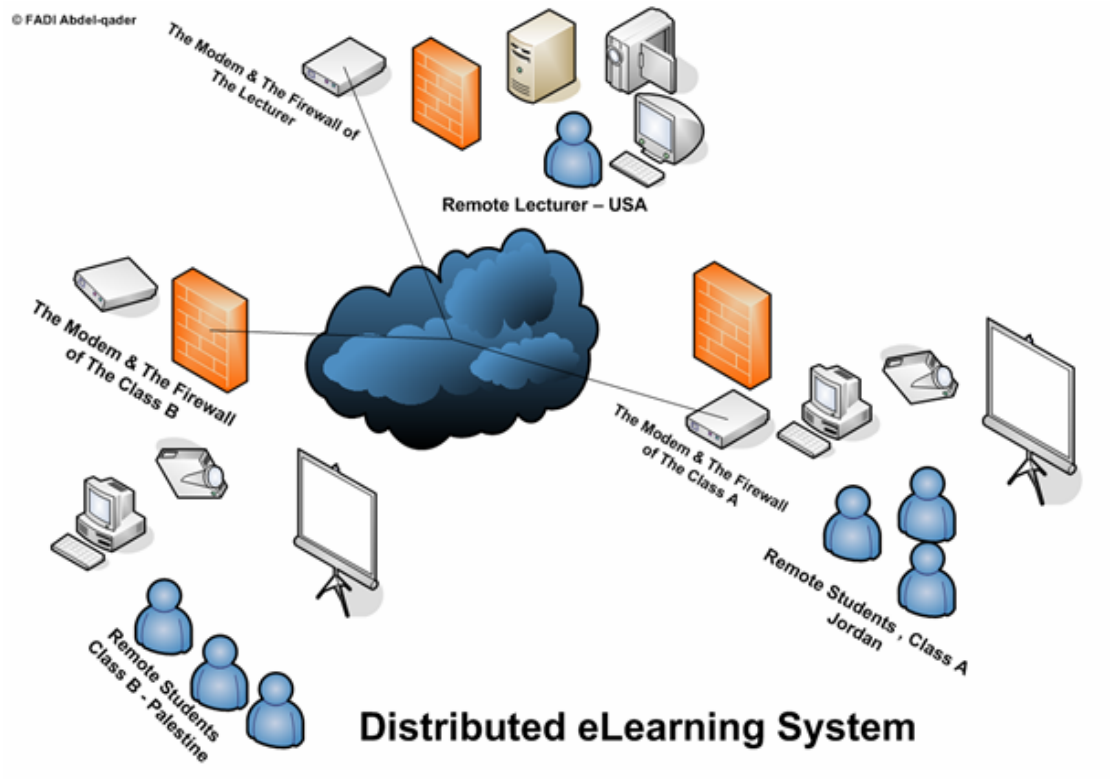
VB.NET:

```
Dim obj As Distributed_Class_Name =
CType(Activator.GetObject(GetType(Distributed_Class_Name), URI),
Distributed_Class_Name)
```

ملاحظة هامة جدا:

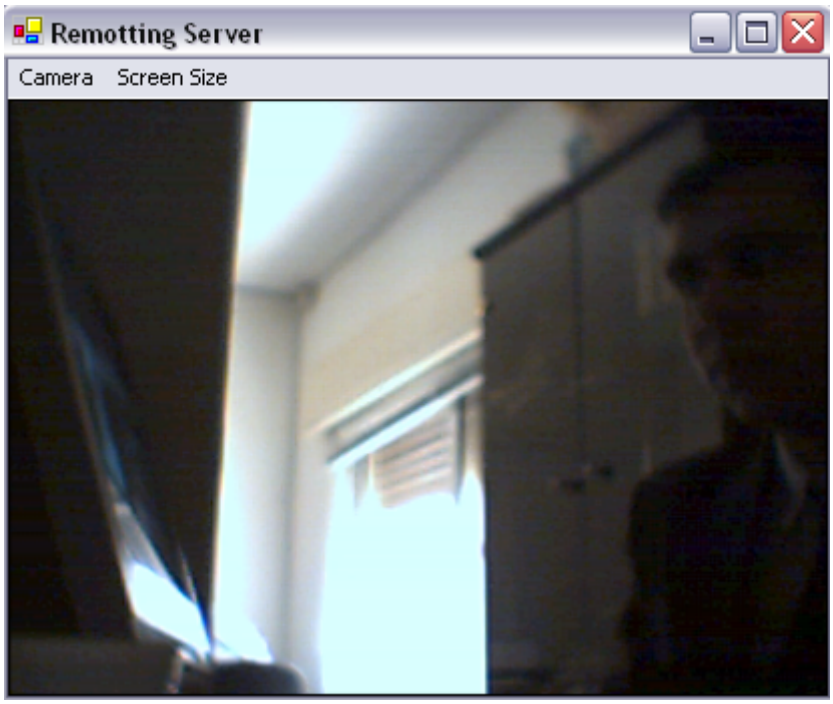
يجب أن يتم توريث الـ MarshalByRefObject Class إلى الـ Distributed Class الذي نريد توزيعه والموجود ضمن System Namespace وعند توريث الـ MarshalByRefObject إلى الـ Distributed Class سيتم تنفيذ كافة العمليات أو الـ Process على الـ Server وسيحصل الـ Client على الـ Result النهائي في حين إذا تم استخدام الـMarshalByValueObject عندها سيتم استخدام الـ Remote Class و لاكن ستتم عملية المعالجة على الـ Client Side ...

وهكذا بينا كيفية استخدام الـ Remotting في الدوت نيت ، سنقوم الآن بتطبيق هذه المفاهيم في مشروع حيث سنقوم بإنشاء Distributed Class يقوم بعملية التقاط لصورة سطح المكتب الخاص بالـ Server وبثها إلى الـ Client مستخدما الـ TcpChannel بين الـ Server والـ Client وسنفترض في هذا المثال إنشاء Remote Classroom حيث يقوم المعلم بإلقاء محاضرة إلى طلابه ويتم بث صورة الكاميرا وسطح المكتب الخاص بالمعلم عبر الإنترنت إلى الطلاب في الـ Classroom والموجودين في دول متعددة وكما في الشكل التالي:

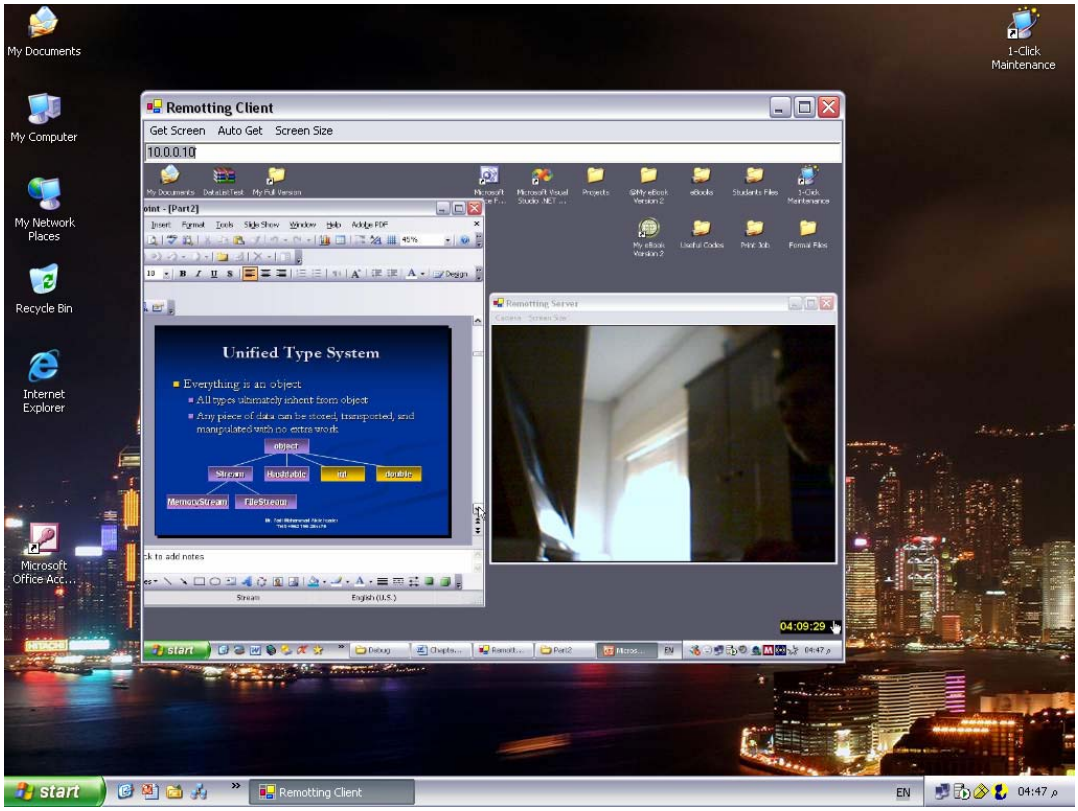


سنستخدم في هذا المثال الـ TCP Channel والـ HttpChannel وسنقارن الأداء في كل منهما ، سيكون الشكل العام للبرنامج كما في الشكل التالي:

برنامج المحاضر (Lecturer Project) وسيكون الشكل العام له كما هو مبين في الشكل التالي حيث:



برنامج الـ Client ويمكن أن يكون من خلال الإنترنت في حالة كان لديك Real IP أو في الشبكة المحلية وكما هو مبين في الشكل التالي:



ولإنشاء الـ Remote Class والذي سيقوم بجلب صورة سطح المكتب ، سنستخدم دوال الـ API ومنها الـ GetDesktopWindow Method والموجودة ضمن الـuser32.dll والـ BitBlt Method لرسم الصورة ، ثم سنقوم بتحويل الصورة إلى Byte Array وحتى نستطيع إرسالها إلى الـ Clients ، وبتأكيد سنستخدم طريقة MarshalByRefObject وحتى تتم عملية التقاط ومعالجة الصورة على الـ Server Side ، ويتم ذلك كما يلي :

C#:

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.IO;

public class ScreenCapture : System.MarshalByRefObject
{
[DllImport("user32.dll")]
private static extern IntPtr GetDesktopWindow();

[DllImport("gdi32.dll")]
private static extern bool BitBlt(
        IntPtr hdcDest, // handle to destination DC
        int nXDest, // x-coord of destination upper-left corner
        int nYDest, // y-coord of destination upper-left corner
        int nWidth, // width of destination rectangle
        int nHeight, // height of destination rectangle
        IntPtr hdcSrc, // handle to source DC
        int nXSrc, // x-coordinate of source upper-left corner
        int nYSrc, // y-coordinate of source upper-left corner
        System.Int32 dwRop // raster operation code
);

private const Int32 SRCCOPY = 0xCC0020;
[DllImport("user32.dll")]
private static extern int GetSystemMetrics(int nIndex);

private const int SM_CXSCREEN = 0;
private const int SM_CYSCREEN = 1;

public Size GetDesktopBitmapSize()
{
return new Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN));
}

public byte[] GetDesktopBitmapBytes()
        {
Size DesktopBitmapSize = GetDesktopBitmapSize();
Graphics Graphic = Graphics.FromHwnd(GetDesktopWindow());
Bitmap MemImage = new Bitmap(DesktopBitmapSize.Width,
DesktopBitmapSize.Height, Graphic);
Graphics MemGraphic = Graphics.FromImage(MemImage);
```

```
IntPtr dc1 = Graphic.GetHdc();
IntPtr dc2 = MemGraphic.GetHdc();
BitBlt(dc2, 0, 0, DesktopBitmapSize.Width, DesktopBitmapSize.Height, dc1, 0,
0, SRCCOPY);
Graphic.ReleaseHdc(dc1);
MemGraphic.ReleaseHdc(dc2);
Graphic.Dispose();
MemGraphic.Dispose();

Graphics g = System.Drawing.Graphics.FromImage(MemImage);
System.Windows.Forms.Cursor cur = System.Windows.Forms.Cursors.Arrow;
cur.Draw(g,new Rectangle(System.Windows.Forms.Cursor.Position.X-
10,System.Windows.Forms.Cursor.Position.Y-
10,cur.Size.Width,cur.Size.Height));

MemoryStream ms = new MemoryStream();
MemImage.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
return ms.GetBuffer();
}

}
```

VB.NET

```
Imports System
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices
Imports System.IO

Public Class ScreenCapture : Inherits System.MarshalByRefObject
   <DllImport("user32.dll")> _
   Private Shared Function GetDesktopWindow() As IntPtr
   End Function

   <DllImport("gdi32.dll")> _
   Private Shared Function BitBlt(ByVal hdcDest As IntPtr, ByVal nXDest As
Integer, ByVal nYDest As Integer, ByVal nWidth As Integer, ByVal nHeight As
Integer, ByVal hdcSrc As IntPtr, ByVal nXSrc As Integer, ByVal nYSrc As
Integer, ByVal dwRop As System.Int32) As Boolean
   End Function

   Private Const SRCCOPY As Int32 = &HCC0020
   <DllImport("user32.dll")> _
Private Shared Function GetSystemMetrics(ByVal nIndex As Integer) As
Integer
End Function

 Private Const SM_CXSCREEN As Integer = 0
 Private Const SM_CYSCREEN As Integer = 1
 Public Function GetDesktopBitmapSize() As Size
```

```
 Return New Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN))
   End Function

   Public Function GetDesktopBitmapBytes() As Byte()
      Dim DesktopBitmapSize As Size = GetDesktopBitmapSize()
      Dim Graphic As Graphics = Graphics.FromHwnd(GetDesktopWindow())
      Dim MemImage As Bitmap = New Bitmap(DesktopBitmapSize.Width,
DesktopBitmapSize.Height, Graphic)
      Dim MemGraphic As Graphics = Graphics.FromImage(MemImage)
      Dim dc1 As IntPtr = Graphic.GetHdc()
      Dim dc2 As IntPtr = MemGraphic.GetHdc()
      BitBlt(dc2, 0, 0, DesktopBitmapSize.Width, DesktopBitmapSize.Height,
dc1, 0, 0, SRCCOPY)
      Graphic.ReleaseHdc(dc1)
      MemGraphic.ReleaseHdc(dc2)
      Graphic.Dispose()
      MemGraphic.Dispose()

Dim g As Graphics = System.Drawing.Graphics.FromImage(MemImage)
Dim cur As System.Windows.Forms.Cursor =
System.Windows.Forms.Cursors.Arrow
cur.Draw(g, New Rectangle(System.Windows.Forms.Cursor.Position.X - 10,
System.Windows.Forms.Cursor.Position.Y - 10, cur.Size.Width,
cur.Size.Height))

Dim ms As MemoryStream = New MemoryStream
MemImage.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Return ms.GetBuffer()
   End Function
End Class
```

وبتأكيد يجب تحويل الـ Class السابق إلى Dll File ثم وضعه بجانب الملف التنفيذي للمشروع ، وأيضا إضافته ضمن الـ References في مشروع الـ Client وحتى نستطيع عمل الـ Casting على الـ Object القادم من الـ Sever ...

أولا : إنشاء الـ Remotting Server :

سيحتوي برنامج الـ Server على الأمور التالية:

Class	Object Name
pictureBox	pictureBox1
mainMenu	mainMenu1
WebCamCapture – External Dll File	WebCamCapture1

سنقوم الآن بإنشاء وتسجيل TCP Remote Channel ثم إنشاء الـ Server Object ويتم ذلك كما يلي:

C#:

```
TcpChannel chan = new TcpChannel(6600);
ChannelServices.RegisterChannel(chan);
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetType("Screen
Capture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton);
```

VB.NET

```
Dim chan As TcpChannel = New TcpChannel(6600)
ChannelServices.RegisterChannel(chan)
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetType("Screen
Capture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton)
```

قمنا بتسمية الـ Server Object بـ MyCaptureScreenServer و طبيعة الـ Instance سيكون Singleton وهذا يعني استخدام نفس الـ Instance Object لكل الـ Clients و في حالة إذا أردنا إنشاء New Instance Object لكل Client Request فيجب استخدام الـ SingleCall وفي حالتنا هذه ينصح باستخدام الأول وحتى لا يزيد الـ Load على الـServer في حالة ربطه على مجموعة كبيرة من الـ Clients إذ أن المعلومات التي يتم بثها إلى الـ Clients هي نفسها ...

ولربط الـ WebCamCapture Class في الـ Picturebox نضع الكود التالي في حدث ImageCaptured الخاص بالـ WebCamCapture Class وكما يلي:

C#:

```
private void webCamCapture1_ImageCaptured(object source,
WebCam_Capture.WebcamEventArgs e)
        {
pictureBox1.Image = e.WebCamImage;
        }
```

VB.NET:

```
Private Sub webCamCapture1_ImageCaptured(ByVal source As Object, ByVal
e As WebCam_Capture.WebcamEventArgs)
   pictureBox1.Image = e.WebCamImage
End Sub
```

والآن نستطيع تشغيل الكاميرا عند الضغط على الـ Start Button الموجود في الـ main Menu ونضع في الحدث الخاص بها كود التشغيل ونمرر له الـ Period Time لكل عملية التقاط وهو مثلا 1 Milliseconds ويتم ذلك كما يلي:

C#:

```
this.webCamCapture1.TimeToCapture_milliseconds = 1;
this.webCamCapture1.Start(0);
```

VB.NET:

```
Me.webCamCapture1.TimeToCapture_milliseconds = 1
Me.webCamCapture1.Start(0)
```

ثانيا : إنشاء الـ Remotting Client :
ويحتوي أيضا على الأمور التالية:

Class	Object Name
pictureBox	pictureBox1
mainMenu	mainMenu1
Timer	timer1
TextBox	textBox1
ScreenCapture Class – External Class in References	ScreenCapture

وسنقوم بتعريف المتغيرات والـ Objects التالية في الـ Global Declaration للبرنامج:

C#:
```
ScreenCapture obj;
TcpChannel chan;
string URI;
```

VB.NET:
```
Dim obj As ScreenCapture
Dim chan As TcpChannel
Dim URI As String
```

سنستخدم الـScreenCapture Object لعمل الـ Casting على الـ Incoming Remote Instance Object ، وسنستخدم الـ TcpChannel Object لإنشاء الـ Remotting Channel وسنضع في الـ URI String Variable الـ URI الخاص بالـ Server Object والذي تم تعريفه في الـ Server .

ثم سنقوم بوضع التعريفات الأساسية للـ TCP Channel والـ Remote Object في الـConstructer الخاص بالمشروع وكما يلي:

C#:
```
public Form1()
{

URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
chan = new TcpChannel();
ChannelServices.RegisterChannel(chan);
obj = (ScreenCapture)Activator.GetObject(typeof(ScreenCapture), URI);
 }
```

VB.NET:
```
URI = "Tcp://" & textBox1.Text & ":6600/MyCaptureScreenServer"
chan = New TcpChannel()
ChannelServices.RegisterChannel(chan)
obj = CType(Activator.GetObject(GetType(ScreenCapture), URI),
ScreenCapture)
```

حيث سيتم تمرير الـ Remote IP أو الـ DNS الخاص بالـ Server من الـ TextBox ويتبعه رقم الـ Port والـ Remote Object الذي تم تعريفه في الـ Client...

ولجلب الصورة سنضع الكود التالي في Timer حيث يجلب الصورة من الـ Server كل فترة محددة:

C#:

```
private void timer1_Tick(object sender, System.EventArgs e)
{
try
{
URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
byte[] buffer = obj.GetDesktopBitmapBytes();
MemoryStream ms = new MemoryStream(buffer);
pictureBox1.Image = Image.FromStream(ms);
}
catch(Exception ex){timer1.Enabled=false; MessageBox.Show(ex.Message);}
}
```

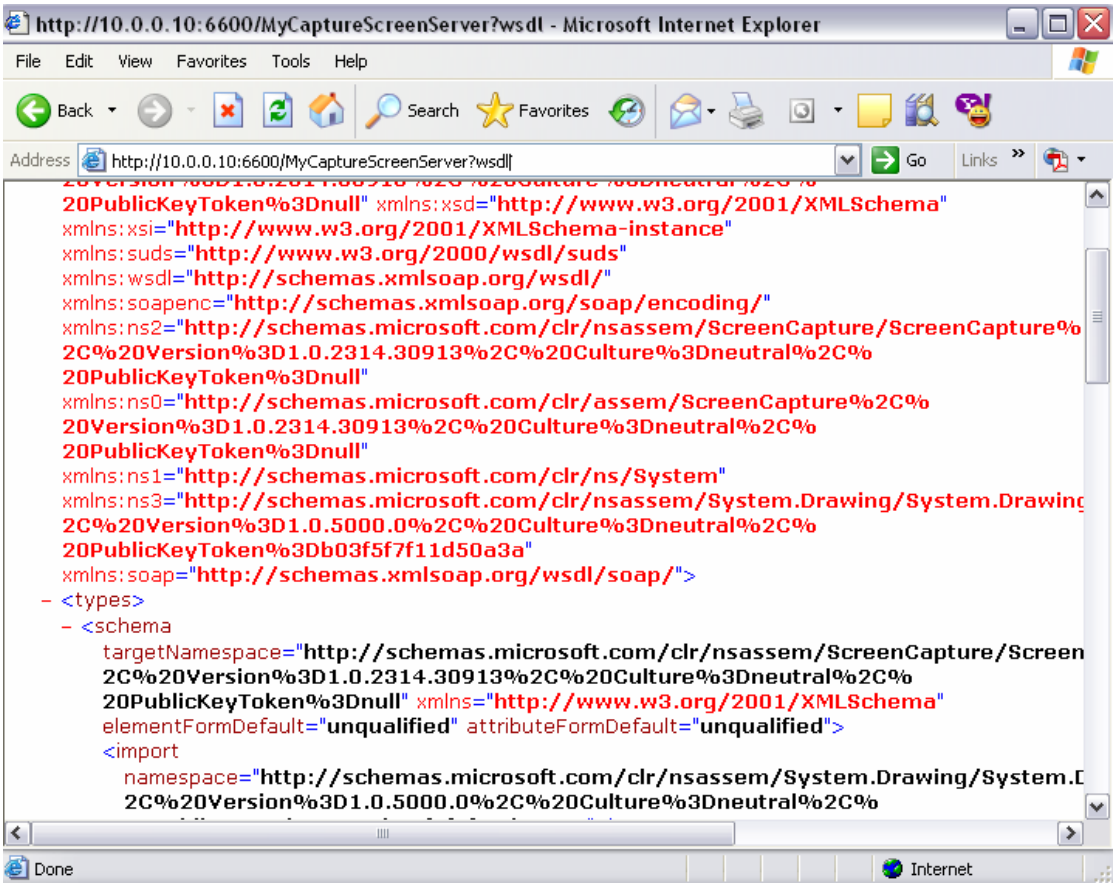
VB.NET:

```
 Private Sub timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs)
    Try
        URI = "Tcp://" & textBox1.Text & ":6600/MyCaptureScreenServer"
        Dim buffer As Byte() = obj.GetDesktopBitmapBytes()
        Dim ms As MemoryStream = New MemoryStream(buffer)
        pictureBox1.Image = Image.FromStream(ms)
    Catch ex As Exception
        timer1.Enabled = False
        MessageBox.Show(ex.Message)
    End Try
End Sub
```

ولتحويل قناة الاتصال إلى Http Channel فقط قم بتغيير الـ TcpChannel إلى HttpChannel وعندها ستتمكن من مشاهدة الـ Remote Methods عبر الـ Internet Browser ولمشاهدتها نضيف الـ WSDL – Services Definition Language Query التالي إلى الـ URI الخاص بالـ Remote Object:

http://10.0.0.10:6600/MyCaptureScreenServer?wsdl

وستظهر لنا الـ Remote Interface Class بهيئة XML لاحظ الشكل التالي:



وتستطيع الاستفادة من التركيب السابق للـ Class في حالة لم تتمكن من الحصول على الـ Dll File الخاص بالـ Remote Class ، حيث تستطيع تحويل الـ XML السابق إلى ملف Dll لوضعه مع الـ References في برنامج الـ Client ويتم ذلك باستخدام الـMetadata Namespace ومنها الـ w3cxsd2001 Standard حيث يوضع الجزء الأول الخاص بالـ XML Namespace في الـ Serializable Attribute وكما يلي كمثال:

C#:

```
using System;
using System.Runtime.Remoting.Messaging;
using System.Runtime.Remoting.Metadata;
using System.Runtime.Remoting.Metadata.W3cXsd2001;

[Serializable,SoapType(XmlNamespace=@"http://http://schemas.microsoft.co
m/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.3
0913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull")]
public class ScreenCapture : System.MarshalByRefObject
{// Your Methods with Return Values Only}
```

VB.NET:

```
Imports System
Imports System.Runtime.Remoting.Messaging
Imports System.Runtime.Remoting.Metadata
Imports System.Runtime.Remoting.Metadata.W3cXsd2001

<Serializable(),
SoapTypeAttribute(XmlNamespace:="http://http://schemas.microsoft.com/clr
```

```
/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.30913
%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull"),
Serializable(),
SoapTypeAttribute(XmlNamespace:="http://http://schemas.microsoft.com/clr
/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.30913
%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull")> _
Public Class ScreenCapture : Inherits System.MarshalByRefObject
'Your Methods with Return Values Only
End Class
```

وللاختيار بين الـ TCP أو الـ HTTP Channel فيفضل دائما في حالة كان الـ Remote Object موجود على جهاز تتصل معه عبر الإنترنت استخدام الـ HTTP Channel بسبب محدودية سرعة الاتصال بالإضافة إلى كونه يتوافق مع أي نظام سواء Windows أو غيره لاكن يبقى آداء وسرعة الـ TCP Channel أفضل فقط في الـ Local Network ...

وهكذا بينا كيفية استخدام الـ Remotting في الدوت نيت وطبقنا عملية إنشاء Distributed eLearning System ، وبتأكيد تستطيع الآن إضافة ميزة نقل الصوت إلى جانب نقل صورة الشاشة والكاميرا الخاصة بالمحاضر (لمزيد من المعلومات ارجع إلى الفصل الخاص بالـVOIP) ، وسنبين في الجزء التالي من هذا الفصل كيفية إنشاء برنامج Remote Desktop Application مع خاصية التحكم

بقية الموضوع فقط في النسخة الورقية من الكتاب

www.fadidotnet.org