By S.S. Ahmed

The Internet Transfer Control (ITC) is a handy control for Internet programming, but there is another control that is even more robust and helps programmers to create more flexible applications. The Winsock control comes with Visual Basic 6.0 (VB6) and is used to create applications that access the low-level functions of the Transmission Control Protocol/Internet Protocol (TCP/IP).

TCP/IP is a specification that defines a series of protocols used to standardize how computers exchange information with each other. TCP/IP provides communication across interconnected networks that use diverse hardware architectures and various operating systems. The protocols in TCP/IP are arranged in a series of layers known as a protocol stack. Each layer has its own functionality.

Winsock is a standard that is maintained by Microsoft. This standard is basically a set of routines that describe communications to and from the TCP/IP stack. These routines reside in a dynamic link library (DLL) that runs under Windows. The Winsock DLL is interfaced with TCP/IP and from there through the Internet.

This article will show how to use the Winsock in a client server environment. We will create two separate applications, one of which will be a server and the other will be a client. Both client and server will interact with each other to exchange data. Client will send a request to the server, and the server, which will be connected to a database, will retrieve the information requested by the client from the database and will return the requested information back to the client. You will find a database with this article. The database contains item numbers and their prices. In real-life situations, the database might be located on a machine different from the one that hosts the client application.

Ports

Let's talk about the ports before we proceed any further. A port is a special memory location that exists when two computers are in communication via TCP/IP. Applications use a port number as an identifier to other computers. Both the sending and receiving computers use this port to exchange data.

To make the job of communication easier, some port numbers have been standardized. These standard port numbers have no inherent value other than that users have agreed to use them with certain applications. The table below lists a number of popular and publicly accepted port numbers and their corresponding applications.

Service	Port
HTTP	80
FTP	20, 21
Gopher	70
SMTP	25
POP3	110
Telnet	23
Finger	79
Local	loops/callbacks 0

Using the Winsock Control

Winsock is above the TCP/IP protocol stack in the ISO/OSI model. TCP/IP is an industry standard communication protocol that defines methods for packaging data into packets for transmission between computing devices on a heterogeneous network. TCP/IP is the standard for data transmission over networks, including the Internet. TCP establishes a connection for data transmission and IP defines the method for sending data packets.

The Microsoft Winsock control makes using the TCP/IP a breeze. Microsoft has wrapped up the Winsock and NETAPI API calls into a neat package that you can easily incorporate into your Visual Basic applications.

Winsock Operating Modes

The Transport layer (also known as the Host-to-Host Transport layer) is responsible for providing the Application layer with session and datagram communication services. The core protocols of the Transport layer are TCP and User Datagram Protocol (UDP). The Winsock control supports the following two operating modes:

- sckTCPProtocol
- sckUDPProtocol

Winsock Properties

Winsock enables you to create clients and servers using the same control. This dual functionality enables you to specify through property setting the type of application you will be building. The Winsock control uses a number of the same properties, whether you are creating a client or a server, thereby all but eliminating the learning curve needed to create applications. Some of the important properties of the control are as follows:

BytesReceived Property

This property returns the number of bytes currently in the receive buffer. This is a read-only property and is unavailable at design time. The value returned is a long integer.

LocalHostName Property

The LocalHostName property returns the name of the local host system. This is a read-only property and is unavailable at design time. The value returned is a string.

LocalIP Property

The LocalIP property returns the local host system IP address in the form of a string, such as 11.0.0.127. This property is read-only and is unavailable at design time.

LocalPort Property

This property returns or sets the local port number. This can be both “read from” and “written to” and is available at both design time and runtime. The value returned is a long integer.

Protocol Property

Returns or sets the protocol, either TCP or UDP, used by the Winsock control.

RemoteHost Property

The RemoteHost property returns or sets the remote host. This can be both “read from” and “written to” and is available both in design time and runtime. The value returned is a string and can be specified either as an IP address or as a DNS name.

RemotePort Property

This property returns or sets the remote port number.

State Property

This returns the state of the control as expressed by an enumerated list. This is a read-only property and is unavailable at design time.

Winsock Methods

Some of the important methods of Winsock control are as follows:

Accept Method

It accepts the request for connection from the client system. For this method to be used, the control must be in the listening state.

Close Method

The Close method terminates a TCP connection from either the client or server applications.

GetData Method

GetData is the method that retrieves the current block of data from the buffer and then stores it in a variable of the variant type.

PeekData Method

The PeekData method operates in a fashion similar to the GetData method. However, it does not remove data from the input queue.

Listen Method

This is invoked on the server application to have the server application wait for a TCP request for connection from a client system.

SendData Method

This method dispatches data to the remote computer. It is used for both the client and server systems.

Connect Method

The Connect method requests a connection to a remote computer.

I am not going to discuss events here. You can find more Winsock details at the following address: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon98/html/vbconusingwinsockcontrol.asp

As stated earlier, this article describes the creation of two applications — one server and one client. The database used in the sample is also provided with the code. The database — Prices.mdb –. contains a single table with the fields item number and price. The client sends the item number to the server and the server retrieves the price against that item number from the database and sends it back to the client. One of the current trends in software development is the issue of thick clients versus thin clients. A thick client is basically an application that performs the bulk of the processing on the individual client PC, whereas a thin client performs the processing on the server.

Creating the Client

Follow the steps below:

1. Start a new EXE project.
2. Add a Winsock control to your application.
3. Add all the controls to the form (see the application for details).

Here is the complete code:

```
Option Explicit
Private Sub cmdClose_Click()

Winsock1.Close

shpGo.Visible = False

shpWait.Visible = False

shpError.Visible = True

End Sub

Private Sub cmdConnect_Click()

Winsock1.RemoteHost = “11.0.0.1” ‘Change this to your host IP

Winsock1.RemotePort = 1007

Winsock1.Connect

shpGo.Visible = True

txtItem.SetFocus

End Sub
```

```
Private Sub cmdSend_Click()

If Winsock1.State = sckConnected Then

   Winsock1.SendData txtItem.Text

   shpGo.Visible = True

   Label3.Caption = “Sending Data”

Else

   shpGo.Visible = False

   shpWait.Visible = False

   shpError.Visible = True

   Label3.Caption = “Not currently connected to host”

End If

End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)

Dim sData As String

Winsock1.GetData sData, vbString
```

```
'Label1.Caption = sData

txtPrice.Text = sData

Label3.Caption = "Received Data"

shpGo.Visible = True

shpWait.Visible = False

shpError.Visible = False

End Sub

Private Sub Winsock1_SendComplete()

Label3.Caption = "Completed Data Transmission"

End Sub
```

Creating the Server

The server portion of the price lookup example is designed to accept the item number sent from the client and look up the associated price in a database. The server than sends the information back to the client. There is a file named “path.txt” in the folder called “Server.” Locate that file and change the database path in that file to the database’s location on your machine. The connection to the database is made in the DataArrival event of the Winsock control. The following code segment opens the database and finds the first occurrence of the value in sItemData. When the record is found, the value contained in the price field is sent back to the client.

```
‘ Get clients request from database

   strData = “Item = ‘” & sItemData & “‘”
   rs.Open “select * from prices”, strConnect, adOpenKeyset, adLockOptimistic

   rs.Find strData

   strOutData = rs.Fields(“Price”)
```

Follow the steps below to create the server:

1. Start a new standard EXE in VB.
2. Add the Winsock control to your application.
3. Add the controls to the form as shown in the accompanying code (see the folder named “Server”).

Here is the complete code:

```
Option Explicit

Dim iSockets As Integer

Dim sServerMsg As String

Dim sRequestID As String
Private Sub Form_Load()

   Form1.Show

   lblHostID.Caption = Socket(0).LocalHostName

   lblAddress.Caption = Socket(0).LocalIP

   Socket(0).LocalPort = 1007

   sServerMsg = “Listening to port: ” & Socket(0).LocalPort

   List1.AddItem (sServerMsg)

   Socket(0).Listen

End Sub
```

```
Private Sub socket_Close(Index As Integer)

   sServerMsg = “Connection closed: ” & Socket(Index).RemoteHostIP

   List1.AddItem (sServerMsg)

   Socket(Index).Close

   Unload Socket(Index)

   iSockets = iSockets – 1

   lblConnections.Caption = iSockets

End Sub

Private Sub socket_ConnectionRequest(Index As Integer, ByVal requestID As Long)

   sServerMsg = “Connection request id ” & requestID & ” from ” & Socket(Index).RemoteHostIP

 If Index = 0 Then

   List1.AddItem (sServerMsg)

   sRequestID = requestID

   iSockets = iSockets + 1

   lblConnections.Caption = iSockets
```

```
    Load Socket(iSockets)

    Socket(iSockets).LocalPort = 1007

    Socket(iSockets).Accept requestID

  End If

End Sub

Private Sub socket_DataArrival(Index As Integer, ByVal bytesTotal As Long)

    Dim sItemData As String

    Dim strData As String

    Dim strOutData As String

    Dim strConnect As String

     ‘ get data from client

    Socket(Index).GetData sItemData, vbString

 sServerMsg = “Received: ” & sItemData & ” from ” & Socket(Index).RemoteHostIP & “(” & sRequestID & “)”

    List1.AddItem (sServerMsg)

‘strConnect = “Provider=Microsoft.Jet.OLEDB.4.0;Data Source=G:Prices.mdb;Persist Security Info=False”
```

```
Dim strPath As String

'Change the database path in the text file

Dim fso As New FileSystemObject, txtfile, _

fil1 As File, ts As TextStream

Set fil1 = fso.GetFile("path.txt")

' Read the contents of the file.

Set ts = fil1.OpenAsTextStream(ForReading)

strPath = ts.ReadLine

ts.Close

Set fso = Nothing

strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" & _

  "Persist Security Info=False;Data Source=" & strPath & _

  "; Mode=Read|Write"

Dim rs As New ADODB.Recordset

' Get clients request from database
```

```
    strData = “Item = ‘” & sItemData & “‘”

    rs.Open “select * from prices”, strConnect, adOpenKeyset, adLockOptimistic

    rs.Find strData

    strOutData = rs.Fields(“Price”)

    ‘send data to client

    sServerMsg = “Sending: ” & strOutData & ” to ” & Socket(Index).RemoteHostIP

    List1.AddItem (sServerMsg)

    Socket(Index).SendData strOutData

End Sub
```

Running the Example

1. Create the executable for both applications.
2. Launch both applications.
3. Click the Connect button.
4. Enter a value from 0 to 6 (currently the database contains only six records, error handling is not done in this code; you can add the error handling yourself) and click the Lookup button. The associated price will be displayed in the price field.

About the Author

S.S. Ahmed is a senior software engineer in a software development company that specializes in Web application development. To contact Ahmed with questions or comments, e-mail him at ss_ahmed1@hotmail.com.