Chapter 18 – VB and Networking, Part I

# Objectives

* The WebBrowser Control
* The WebClient Class
* Sending e-mail Messages
* Miscellaneous Network Classes
* Introduction to ASP.NET Projects

Microsoft discovered the Internet in 1996: sort of. Office 95 had been released and it had very limited Internet smarts built in – if you typed in a URL in Word, nothing happened. Microsoft got caught quite by surprise as to what impact the Internet would have on everyday computing. E-mail and web-smart applications were clamored for by even casual computer users.

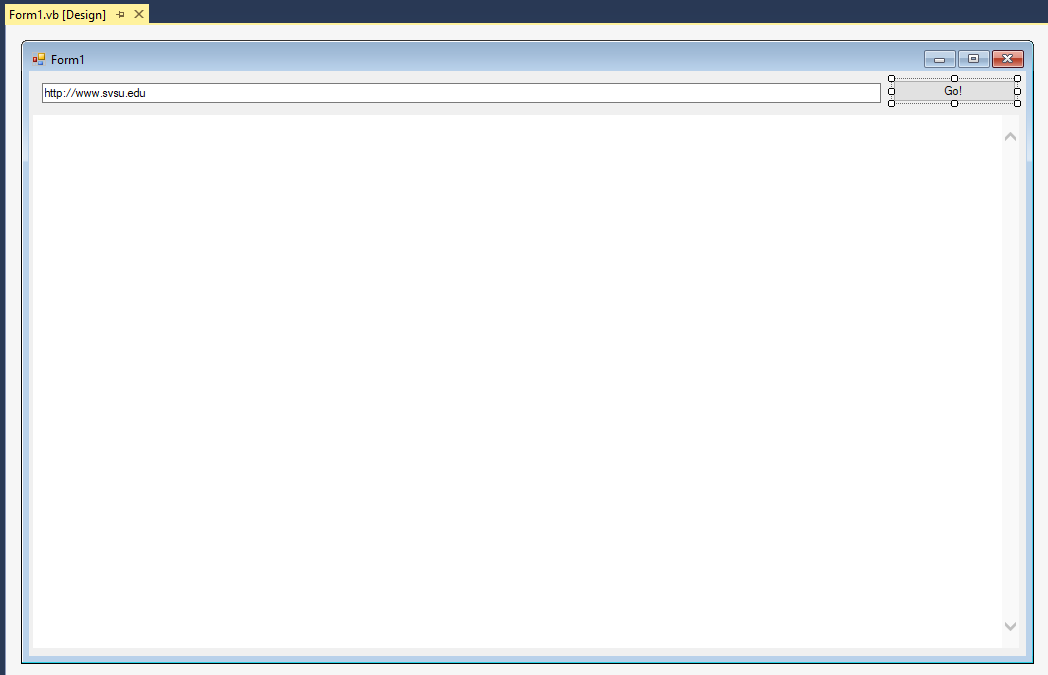
Bill Gates then declared that all Microsoft applications would be web-enabled – each program should understand when and where web access was needed. Since that time, all of Microsoft’s products have a web component that is tightly integrated into the main application. Indeed, it was one of Microsoft’s earliest visions with their .NET strategy that we wouldn't get applications delivered on CD or DVD anymore – it took almost 15 years, but it did happen. Today we all use applications hosted out on Internet based servers. Can you say cloud computing and Software as a Service (SaaS)?

For many big software companies, this approach makes a lot of sense: (1) software piracy would no longer be an issue since at least part of the functionality is on the cloud, (2) only one version of the application has to be maintained, (3) all users can be instantly updated to the newest version when it is released, and (4) software is licensed on a subscription basis, meaning there is a continual revenue stream for the company. With the large-scale availability of high bandwidth and reliable networks, this vision is realistic. We are on another digital divide of the haves and have nots though – there are still people in rural areas with either no, or very limited, broadband speeds to whom this technology cannot be used.

When it comes to developing applications with networking support built in, fortunately for us, VB is very well equipped. In fact, almost all the acronym alphabet soup of web technologies such as DHTML, CSS, SOAP, JQuery, ECMAScript and XML can be handled by VB. This chapter, the first part of two on networking technology, is going to concentrate on a few general items that can make our networking lives simpler.

# The WebBrowser Control

The WebBrowser control is a standard control found in the toolbox that will let you build a working web browser! On your form, you will need a textbox (txtURL) that allows a user to enter the URL he/she wants to navigate to and a command button (cmdNavigate) that will be used to go to the TextBox's URL. Don't forget to add the WebBrowser control; you can leave the default name of WebBrowser1 alone. You will drag the control out to cover most of your form, just like the RichTextBox control. Here's a screenshot of my completed form:



Now get ready to do lots of typing. Add the following code to the Command button’s Click event – again this is assuming that you named things as I indicated above.

'Chapter 18 - Program 1

Public Class Form1

Private Sub cmdNavigate\_Click(sender As Object, e As EventArgs)

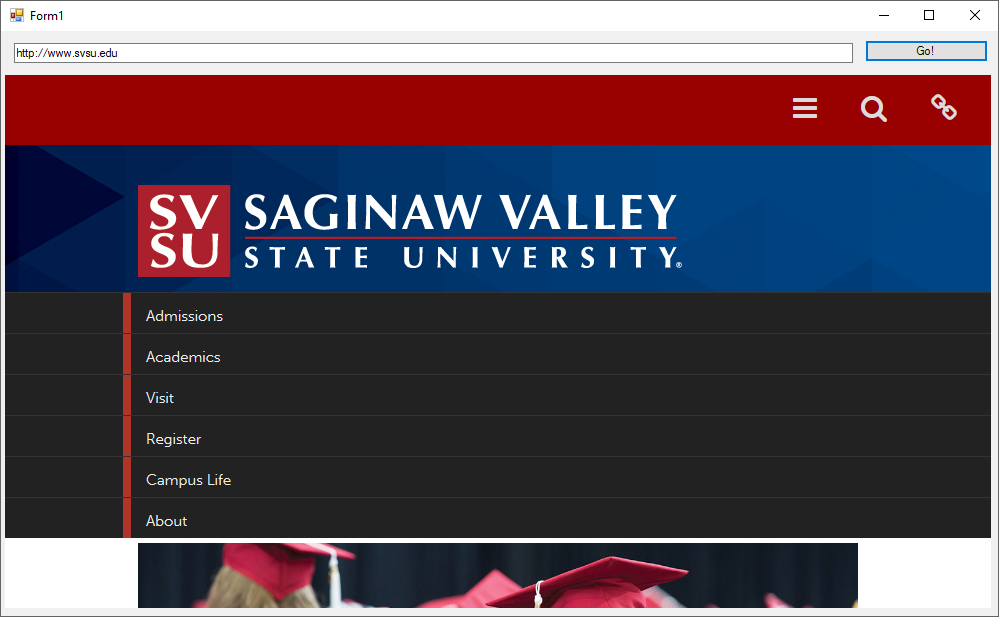
Handles cmdNavigate.Click

WebBrowser1.Navigate(txtURL.Text)

End Sub

End Class

Here’s the web browser application up and running. This assumes that you have a network connection available to the outside world:



That was it! Now think about how easy it would be to add a “help” option to an application that would automatically point a web browser at the application developer’s web site for help. Web based support of your application will not get any easier than this. You would only have one copy of help information to maintain and every one of your clients would point at that same single set up-to-date help set.

Here is a list of some of the other methods that you might want to use in the WebBrowser control:

|  |  |
| --- | --- |
| *Method* | *Purpose* |
| GoBack | Navigates back one page. |
| GoHome | Navigates to the default home page . |
| GoForward | Navigates forward one page. |
| GoSearch | Navigate to the default search provider. |
| Navigate | Retrieve the web page from a destination address. |
| Refresh | Reloads the current document. |

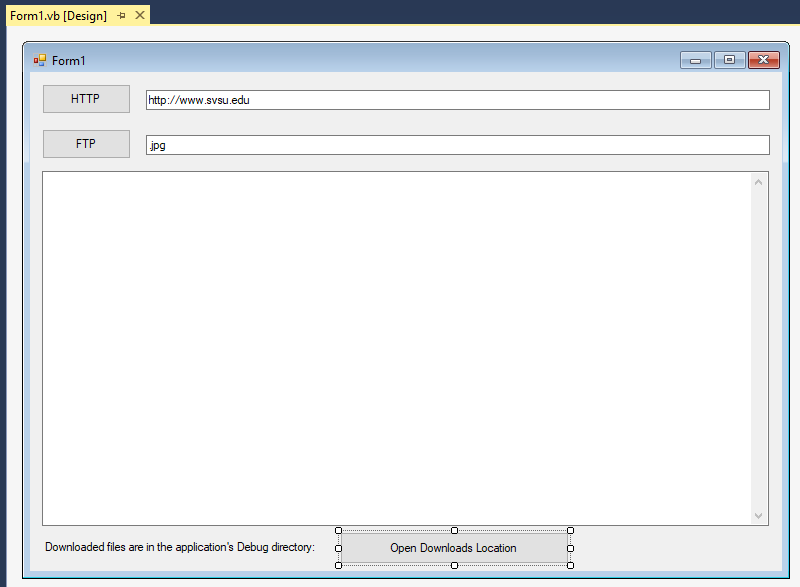
There are a plethora of properties, methods and events available on this control that allow you to handle just about any kind of customization that you desire.

# The WebClient Class

If you are interested in transferring information by FTP (File Transfer Protocol) or HTTP (HyperText Transport Protocol), then the WebClient class is just what you need.

Let's create an application that will show us the contents of a raw web page and also allow us to download certain filetypes from a website. We’ll create three TextBoxes: one for the URL we want to work with (txtURL), one to indicate the types of files we want to download (txtFileType) and the third to show the contents of our activities (txtOutput). We’ll need three Command buttons: the first one (cmdHTTP) will retrieve the URL and show its RAW form in the big TextBox – we'll see the HTML as actual text, rather than as the nice neat rendered form that we get in a web browser. If we press the second Command button (cmdFTP), any files that exist at the specified URL of the type specified will attempted to be downloaded. Finally, the third Command button (cmdOpenDownloadsLocation) will do just what it says.

Everything else happens in the application… Here's a screenshot of my application's form:



Remember to set the bottom textbox (txtOutput) to be multiline and to support ScrollBars. Again, this application will retrieve web pages and display the text that makes the page up.

Here’s the application's source code:

'Chapter 18 - Program 2

'We need to import the .Net (network) namespace

'as well as RegularExpressions for some of the

'work we are going to do here...

Imports System.Net

Imports System.Text.RegularExpressions

Public Class Form1

Private Sub cmdHTTP\_Click(sender As Object, e As EventArgs)

Handles cmdHTTP.Click

'Create a new web client instance

Dim MyWebClient As New WebClient

'Clear out the main textbox

txtOutput.Text = ""

'Download the URI as a string and shove it into the Output Textbox

txtOutput.Text = MyWebClient.DownloadString(txtURL.Text)

'Delete the web client

MyWebClient = Nothing

End Sub

Private Sub cmdFTP\_Click(sender As Object, e As EventArgs)

Handles cmdFTP.Click

'Create a new web client instance

Dim MyWebClient As New WebClient

Dim strInitialPageToScan As String

Dim myRegExMatches As MatchCollection

Dim strRemoteFilename As String

Dim strLocalFilename As String

'Clear out the main textbox

txtOutput.Text = ""

'Get the page in a string

strInitialPageToScan = MyWebClient.DownloadString(txtURL.Text)

'Find all matches to our search string using Regex

myRegExMatches = Regex.Matches(strInitialPageToScan,

"http\S+[^-;:?]\" & txtFileType.Text)

'For each match that we have

For Each aMatch In myRegExMatches

'Keep track of the server filename of the file

strRemoteFilename = aMatch.ToString

'Create a local filename for the file

strLocalFilename = Environment.CurrentDirectory & "\" &

strRemoteFilename.Substring(strRemoteFilename.LastIndexOf("/") + 1)

strRemoteFilename.Substring(strRemoteFilename.LastIndexOf("/") + 1)

'Update the textbox and try to download the file...

txtOutput.Text &= "Attempting Downloading of File : " &

aMatch.ToString()

Try

MyWebClient.DownloadFile(New Uri(strRemoteFilename),

strLocalFilename)

Catch ex As Exception

txtOutput.Text &= " \*\*\* FAILED \*\*\* "

Finally

txtOutput.Text &= vbCrLf

End Try

Next

End Sub

Private Sub cmdOpenDownloadsLocation\_Click(sender As Object, e As EventArgs)

Handles cmdOpenDownloadsLocation.Click

'Start a Windows Explorer session pointing at the application's

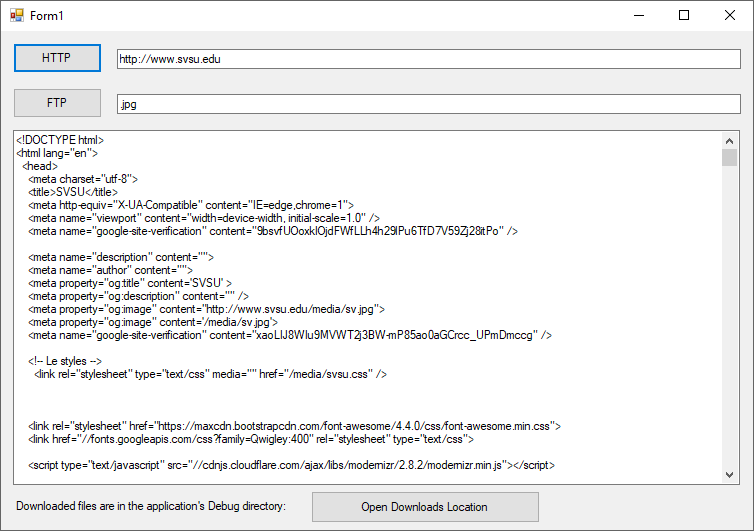
'Debug folder, which is where our downloaded files landed

Process.Start(Environment.CurrentDirectory & "\")

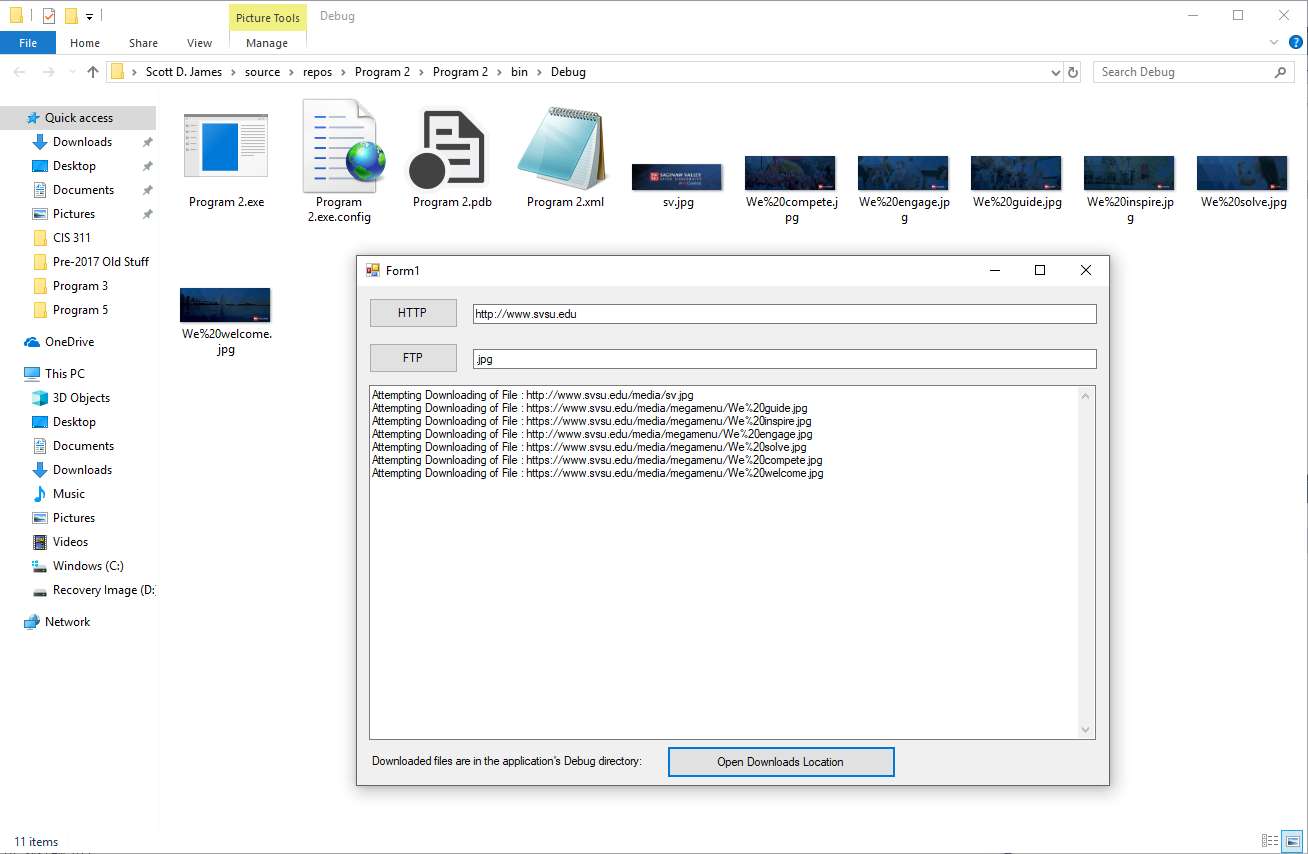
End Sub

End Class

Here’s a screenshot of the application while it's running, before downloading any files:



After we press FTP and open the location where the files got downloaded, we see this:



Basically any address that starts with http:, https:, ftp:, or file: can be accessed with this class. Let’s look at some of the properties of the WebClient class:

|  |  |
| --- | --- |
| *Name* | *Description* |
| [BaseAddress](http://msdn.microsoft.com/en-us/library/system.net.webclient.baseaddress.aspx) | Gets or sets the base URI for requests made by a WebClient. |
| [CachePolicy](http://msdn.microsoft.com/en-us/library/system.net.webclient.cachepolicy.aspx) | Gets or sets the application's cache policy for any resources obtained by this WebClient instance using [WebRequest](http://msdn.microsoft.com/en-us/library/system.net.webrequest.aspx) objects. |
| [Credentials](http://msdn.microsoft.com/en-us/library/system.net.webclient.credentials.aspx) | Gets or sets the network credentials that are sent to the host and used to authenticate the request. |
| [Encoding](http://msdn.microsoft.com/en-us/library/system.net.webclient.encoding.aspx) | Gets and sets the [Encoding](http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx) used to upload and download strings. |
| [Headers](http://msdn.microsoft.com/en-us/library/system.net.webclient.headers.aspx) | Gets or sets a collection of header name/value pairs associated with the request. |
| [IsBusy](http://msdn.microsoft.com/en-us/library/system.net.webclient.isbusy.aspx) | Gets whether a web request is in progress. |
| [Proxy](http://msdn.microsoft.com/en-us/library/system.net.webclient.proxy.aspx) | Gets or sets the proxy used by this WebClient object. |
| [QueryString](http://msdn.microsoft.com/en-us/library/system.net.webclient.querystring.aspx) | Gets or sets a collection of query name/value pairs associated with the request. |
| [ResponseHeaders](http://msdn.microsoft.com/en-us/library/system.net.webclient.responseheaders.aspx) | Gets a collection of header name/value pairs associated with the response. |
| [UseDefaultCredentials](http://msdn.microsoft.com/en-us/library/system.net.webclient.usedefaultcredentials.aspx) | Gets or sets a [Boolean](http://msdn.microsoft.com/en-us/library/system.boolean.aspx) value that controls whether the [DefaultCredentials](http://msdn.microsoft.com/en-us/library/system.net.credentialcache.defaultcredentials.aspx) are sent with requests. |

Here are the main WebClient methods that we are interested in:

|  |  |
| --- | --- |
| *Name* | *Description* |
| [CancelAsync](http://msdn.microsoft.com/en-us/library/system.net.webclient.cancelasync.aspx) | Cancels a pending asynchronous operation. |
| [DownloadData(String)](http://msdn.microsoft.com/en-us/library/xz398a3f.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified. |
| [DownloadData(Uri)](http://msdn.microsoft.com/en-us/library/ms144188.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified. |
| [DownloadDataAsync(Uri)](http://msdn.microsoft.com/en-us/library/ms144190.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified as an asynchronous operation. |
| [DownloadDataAsync(Uri, Object)](http://msdn.microsoft.com/en-us/library/ms144191.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified as an asynchronous operation. |
| [DownloadDataTaskAsync(String)](http://msdn.microsoft.com/en-us/library/hh138334.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified as an asynchronous operation using a task object. |
| [DownloadDataTaskAsync(Uri)](http://msdn.microsoft.com/en-us/library/hh159275.aspx) | Downloads the resource as a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array from the URI specified as an asynchronous operation using a task object. |
| [DownloadFile(String, String)](http://msdn.microsoft.com/en-us/library/ez801hhe.aspx) | Downloads the resource with the specified URI to a local file. |
| [DownloadFile(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144194.aspx) | Downloads the resource with the specified URI to a local file. |
| [DownloadFileAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144196.aspx) | Downloads, to a local file, the resource with the specified URI. This method does not block the calling thread. |
| [DownloadFileAsync(Uri, String, Object)](http://msdn.microsoft.com/en-us/library/ms144197.aspx) | Downloads, to a local file, the resource with the specified URI. This method does not block the calling thread. |
| [DownloadFileTaskAsync(String, String)](http://msdn.microsoft.com/en-us/library/hh159424.aspx) | Downloads the specified resource to a local file as an asynchronous operation using a task object. |
| [DownloadFileTaskAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/hh193917.aspx) | Downloads the specified resource to a local file as an asynchronous operation using a task object. |
| [DownloadString(String)](http://msdn.microsoft.com/en-us/library/fhd1f0sw.aspx) | Downloads the requested resource as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx). The resource to download is specified as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx) containing the URI. |
| [DownloadString(Uri)](http://msdn.microsoft.com/en-us/library/ms144200.aspx) | Downloads the requested resource as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx). The resource to download is specified as a [Uri](http://msdn.microsoft.com/en-us/library/system.uri.aspx). |
| [DownloadStringAsync(Uri)](http://msdn.microsoft.com/en-us/library/ms144202.aspx) | Downloads the resource specified as a [Uri](http://msdn.microsoft.com/en-us/library/system.uri.aspx). This method does not block the calling thread. |
| [DownloadStringAsync(Uri, Object)](http://msdn.microsoft.com/en-us/library/ms144203.aspx) | Downloads the specified string to the specified resource. This method does not block the calling thread. |
| [DownloadStringTaskAsync(String)](http://msdn.microsoft.com/en-us/library/hh138332.aspx) | Downloads the resource as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx) from the URI specified as an asynchronous operation using a task object. |
| [DownloadStringTaskAsync(Uri)](http://msdn.microsoft.com/en-us/library/hh194294.aspx) | Downloads the resource as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx) from the URI specified as an asynchronous operation using a task object. |
| [GetWebRequest](http://msdn.microsoft.com/en-us/library/system.net.webclient.getwebrequest.aspx) | Returns a [WebRequest](http://msdn.microsoft.com/en-us/library/system.net.webrequest.aspx) object for the specified resource. |
| [GetWebResponse(WebRequest)](http://msdn.microsoft.com/en-us/library/a3xa6ys0.aspx) | Returns the [WebResponse](http://msdn.microsoft.com/en-us/library/system.net.webresponse.aspx) for the specified [WebRequest](http://msdn.microsoft.com/en-us/library/system.net.webrequest.aspx). |
| [GetWebResponse(WebRequest, IAsyncResult)](http://msdn.microsoft.com/en-us/library/c2xze5ez.aspx) | Returns the [WebResponse](http://msdn.microsoft.com/en-us/library/system.net.webresponse.aspx) for the specified [WebRequest](http://msdn.microsoft.com/en-us/library/system.net.webrequest.aspx) using the specified [IAsyncResult](http://msdn.microsoft.com/en-us/library/system.iasyncresult.aspx). |
| [OpenRead(String)](http://msdn.microsoft.com/en-us/library/781fwaz8.aspx) | Opens a readable stream for the data downloaded from a resource with the URI specified as a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx). |
| [OpenRead(Uri)](http://msdn.microsoft.com/en-us/library/ms144209.aspx) | Opens a readable stream for the data downloaded from a resource with the URI specified as a [Uri](http://msdn.microsoft.com/en-us/library/system.uri.aspx) |
| [OpenReadAsync(Uri)](http://msdn.microsoft.com/en-us/library/ms144211.aspx) | Opens a readable stream containing the specified resource. This method does not block the calling thread. |
| [OpenReadAsync(Uri, Object)](http://msdn.microsoft.com/en-us/library/ms144212.aspx) | Opens a readable stream containing the specified resource. This method does not block the calling thread. |
| [OpenReadTaskAsync(String)](http://msdn.microsoft.com/en-us/library/hh159420.aspx) | Opens a readable stream containing the specified resource as an asynchronous operation using a task object. |
| [OpenReadTaskAsync(Uri)](http://msdn.microsoft.com/en-us/library/hh138335.aspx) | Opens a readable stream containing the specified resource as an asynchronous operation using a task object. |
| [OpenWrite(String)](http://msdn.microsoft.com/en-us/library/3t27yce3.aspx) | Opens a stream for writing data to the specified resource. |
| [OpenWrite(Uri)](http://msdn.microsoft.com/en-us/library/ms144214.aspx) | Opens a stream for writing data to the specified resource. |
| [OpenWrite(String, String)](http://msdn.microsoft.com/en-us/library/0a7f52z4.aspx) | Opens a stream for writing data to the specified resource, using the specified method. |
| [OpenWrite(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144215.aspx) | Opens a stream for writing data to the specified resource, by using the specified method. |
| [OpenWriteAsync(Uri)](http://msdn.microsoft.com/en-us/library/ms144217.aspx) | Opens a stream for writing data to the specified resource. This method does not block the calling thread. |
| [OpenWriteAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144218.aspx) | Opens a stream for writing data to the specified resource. This method does not block the calling thread. |
| [OpenWriteAsync(Uri, String, Object)](http://msdn.microsoft.com/en-us/library/ms144219.aspx) | Opens a stream for writing data to the specified resource, using the specified method. This method does not block the calling thread. |
| [OpenWriteTaskAsync(String)](http://msdn.microsoft.com/en-us/library/hh159557.aspx) | Opens a stream for writing data to the specified resource as an asynchronous operation using a task object. |
| [OpenWriteTaskAsync(Uri)](http://msdn.microsoft.com/en-us/library/hh159290.aspx) | Opens a stream for writing data to the specified resource as an asynchronous operation using a task object. |
| [OpenWriteTaskAsync(String, String)](http://msdn.microsoft.com/en-us/library/hh194289.aspx) | Opens a stream for writing data to the specified resource as an asynchronous operation using a task object. |
| [OpenWriteTaskAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/hh159307.aspx) | Opens a stream for writing data to the specified resource as an asynchronous operation using a task object. |
| [UploadData(String, Byte())](http://msdn.microsoft.com/en-us/library/tdbbwh0a.aspx) | Uploads a data buffer to a resource identified by a URI. |
| [UploadData(Uri, Byte())](http://msdn.microsoft.com/en-us/library/ms144221.aspx) | Uploads a data buffer to a resource identified by a URI. |
| [UploadData(String, String, Byte())](http://msdn.microsoft.com/en-us/library/ktfa4fek.aspx) | Uploads a data buffer to the specified resource, using the specified method. |
| [UploadData(Uri, String, Byte())](http://msdn.microsoft.com/en-us/library/ms144223.aspx) | Uploads a data buffer to the specified resource, using the specified method. |
| [UploadDataAsync(Uri, Byte())](http://msdn.microsoft.com/en-us/library/ms144225.aspx) | Uploads a data buffer to a resource identified by a URI, using the POST method. This method does not block the calling thread. |
| [UploadDataAsync(Uri, String,Byte())](http://msdn.microsoft.com/en-us/library/ms144226.aspx) | Uploads a data buffer to a resource identified by a URI, using the specified method. This method does not block the calling thread. |
| [UploadDataAsync(Uri, String,Byte(), Object)](http://msdn.microsoft.com/en-us/library/ms144227.aspx) | Uploads a data buffer to a resource identified by a URI, using the specified method and identifying token. |
| [UploadDataTaskAsync(String,Byte())](http://msdn.microsoft.com/en-us/library/hh193934.aspx) | Uploads a data buffer that contains a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array to the URI specified as an asynchronous operation using a task object. |
| [UploadDataTaskAsync(Uri, Byte())](http://msdn.microsoft.com/en-us/library/hh138525.aspx) | Uploads a data buffer that contains a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array to the URI specified as an asynchronous operation using a task object. |
| [UploadDataTaskAsync(String, String, Byte())](http://msdn.microsoft.com/en-us/library/hh138382.aspx) | Uploads a data buffer that contains a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array to the URI specified as an asynchronous operation using a task object. |
| [UploadDataTaskAsync(Uri, String,Byte())](http://msdn.microsoft.com/en-us/library/hh159561.aspx) | Uploads a data buffer that contains a [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx) array to the URI specified as an asynchronous operation using a task object. |
| [UploadFile(String, String)](http://msdn.microsoft.com/en-us/library/36s52zhs.aspx) | Uploads the specified local file to a resource with the specified URI. |
| [UploadFile(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144229.aspx) | Uploads the specified local file to a resource with the specified URI. |
| [UploadFile(String, String, String)](http://msdn.microsoft.com/en-us/library/esst63h0.aspx) | Uploads the specified local file to the specified resource, using the specified method. |
| [UploadFile(Uri, String, String)](http://msdn.microsoft.com/en-us/library/ms144230.aspx) | Uploads the specified local file to the specified resource, using the specified method. |
| [UploadFileAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144232.aspx) | Uploads the specified local file to the specified resource, using the POST method. This method does not block the calling thread. |
| [UploadFileAsync(Uri, String, String)](http://msdn.microsoft.com/en-us/library/ms144233.aspx) | Uploads the specified local file to the specified resource, using the POST method. This method does not block the calling thread. |
| [UploadFileAsync(Uri, String, String, Object)](http://msdn.microsoft.com/en-us/library/ms144234.aspx) | Uploads the specified local file to the specified resource, using the POST method. This method does not block the calling thread. |
| [UploadFileTaskAsync(String, String)](http://msdn.microsoft.com/en-us/library/hh194005.aspx) | Uploads the specified local file to a resource as an asynchronous operation using a task object. |
| [UploadFileTaskAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/hh194308.aspx) | Uploads the specified local file to a resource as an asynchronous operation using a task object. |
| [UploadFileTaskAsync(String, String, String)](http://msdn.microsoft.com/en-us/library/hh193921.aspx) | Uploads the specified local file to a resource as an asynchronous operation using a task object. |
| [UploadFileTaskAsync(Uri, String, String)](http://msdn.microsoft.com/en-us/library/hh138336.aspx) | Uploads the specified local file to a resource as an asynchronous operation using a task object. |
| [UploadString(String, String)](http://msdn.microsoft.com/en-us/library/0645045y.aspx) | Uploads the specified string to the specified resource, using the POST method. |
| [UploadString(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144236.aspx) | Uploads the specified string to the specified resource, using the POST method. |
| [UploadString(String, String, String)](http://msdn.microsoft.com/en-us/library/d0d3595k.aspx) | Uploads the specified string to the specified resource, using the specified method. |
| [UploadString(Uri, String, String)](http://msdn.microsoft.com/en-us/library/ms144237.aspx) | Uploads the specified string to the specified resource, using the specified method. |
| [UploadStringAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/ms144239.aspx) | Uploads the specified string to the specified resource. This method does not block the calling thread. |
| [UploadStringAsync(Uri, String, String)](http://msdn.microsoft.com/en-us/library/ms144240.aspx) | Uploads the specified string to the specified resource. This method does not block the calling thread. |
| [UploadStringAsync(Uri, String, String, Object)](http://msdn.microsoft.com/en-us/library/ms144241.aspx) | Uploads the specified string to the specified resource. This method does not block the calling thread. |
| [UploadStringTaskAsync(String, String)](http://msdn.microsoft.com/en-us/library/hh193920.aspx) | Uploads the specified string to the specified resource as an asynchronous operation using a task object. |
| [UploadStringTaskAsync(Uri, String)](http://msdn.microsoft.com/en-us/library/hh194309.aspx) | Uploads the specified string to the specified resource as an asynchronous operation using a task object. |
| [UploadStringTaskAsync(String, String, String)](http://msdn.microsoft.com/en-us/library/hh159423.aspx) | Uploads the specified string to the specified resource as an asynchronous operation using a task object. |
| [UploadStringTaskAsync(Uri, String, String)](http://msdn.microsoft.com/en-us/library/hh138531.aspx) | Uploads the specified string to the specified resource as an asynchronous operation using a task object. |
| [UploadValues(String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/9w7b4fz7.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI. |
| [UploadValues(Uri, NameValueCollection)](http://msdn.microsoft.com/en-us/library/ms144243.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI. |
| [UploadValues(String, String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/900ted1f.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI, using the specified method. |
| [UploadValues(Uri, String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/ms144244.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI, using the specified method. |
| [UploadValuesAsync(Uri, NameValueCollection)](http://msdn.microsoft.com/en-us/library/ms144246.aspx) | Uploads the data in the specified name/value collection to the resource identified by the specified URI. This method does not block the calling thread. |
| [UploadValuesAsync(Uri, String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/ms144247.aspx) | Uploads the data in the specified name/value collection to the resource identified by the specified URI, using the specified method. This method does not block the calling thread. |
| [UploadValuesAsync(Uri, String, NameValueCollection, Object)](http://msdn.microsoft.com/en-us/library/ms144248.aspx) | Uploads the data in the specified name/value collection to the resource identified by the specified URI, using the specified method. This method does not block the calling thread, and allows the caller to pass an object to the method that is invoked when the operation completes. |
| [UploadValuesTaskAsync(String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/hh472354.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI as an asynchronous operation using a task object. |
| [UploadValuesTaskAsync(Uri, NameValueCollection)](http://msdn.microsoft.com/en-us/library/hh472287.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI as an asynchronous operation using a task object. |
| [UploadValuesTaskAsync(String, String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/hh485749.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI as an asynchronous operation using a task object. |
| [UploadValuesTaskAsync(Uri, String, NameValueCollection)](http://msdn.microsoft.com/en-us/library/hh472286.aspx) | Uploads the specified name/value collection to the resource identified by the specified URI as an asynchronous operation using a task object. |

Finally, here are the WebClient events:

|  |  |
| --- | --- |
| *Name* | *Description* |
| [DownloadDataCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.downloaddatacompleted.aspx) | Occurs when an asynchronous data download operation completes. |
| [DownloadFileCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.downloadfilecompleted.aspx) | Occurs when an asynchronous file download operation completes. |
| [DownloadProgressChanged](http://msdn.microsoft.com/en-us/library/system.net.webclient.downloadprogresschanged.aspx) | Occurs when an asynchronous download operation successfully transfers some or all of the data. |
| [DownloadStringCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.downloadstringcompleted.aspx) | Occurs when an asynchronous resource-download operation completes. |
| [OpenReadCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.openreadcompleted.aspx) | Occurs when an asynchronous operation to open a stream containing a resource completes. |
| [OpenWriteCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.openwritecompleted.aspx) | Occurs when an asynchronous operation to open a stream to write data to a resource completes. |
| [UploadDataCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.uploaddatacompleted.aspx) | Occurs when an asynchronous data-upload operation completes. |
| [UploadFileCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.uploadfilecompleted.aspx) | Occurs when an asynchronous file-upload operation completes. |
| [UploadProgressChanged](http://msdn.microsoft.com/en-us/library/system.net.webclient.uploadprogresschanged.aspx) | Occurs when an asynchronous upload operation successfully transfers some or all of the data. |
| [UploadStringCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.uploadstringcompleted.aspx) | Occurs when an asynchronous string-upload operation completes. |
| [UploadValuesCompleted](http://msdn.microsoft.com/en-us/library/system.net.webclient.uploadvaluescompleted.aspx) | Occurs when an asynchronous upload of a name/value collection completes. |
| [WriteStreamClosed](http://msdn.microsoft.com/en-us/library/system.net.webclient.writestreamclosed.aspx) | Obsolete. Occurs when an asynchronous operation to write data to a resource using a write stream is closed. |

As you can see, it’s pretty easy to access web based resources using this class. Even if you need to be authenticated, you can add that with very little hassle, as in:

'Chapter 18 - Program 3

Imports System.Net

Public Class Form1

Private Sub cmdNetworkAuthenticate\_Click(sender As Object, e As EventArgs)

Handles cmdNetworkAuthenticate.Click

Dim myWebClient As New WebClient

myWebClient.Credentials = New NetworkCredential(txtUsername.Text,

txtPassword.Text)

txtOutput.Text = myWebClient.DownloadString("http://www.svsu.edu/~james")

End Sub

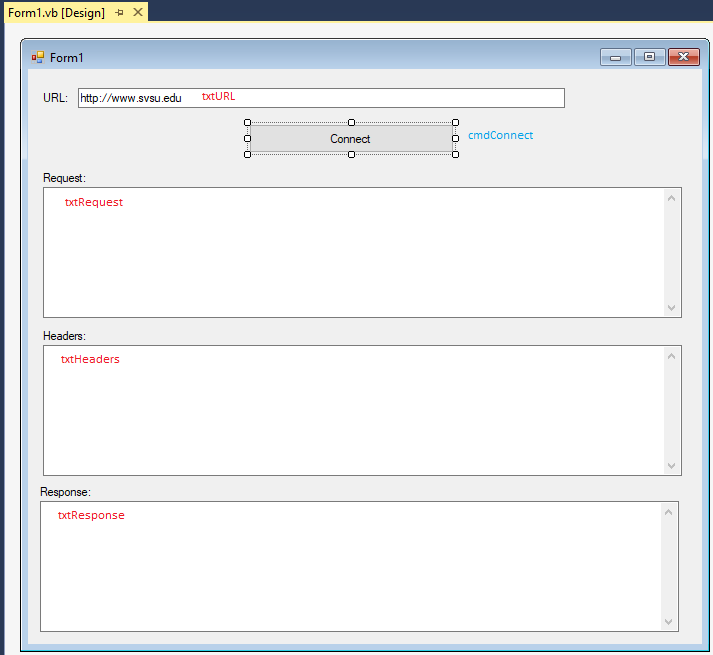
End Class

It’s also worthwhile to point out that the WebClient class actually sits overtop of several other classes that we can directly access if we are so interested. Here’s a list and a short description of each class:

|  |  |
| --- | --- |
| *Class Name* | *Purpose* |
| WebRequest | Makes a request to a URI |
| WebResponse | Provides a response from a URI |
| FileWebRequest/FileWebResponse | Provides a filesystem implementation of the WebRequest/WebResponse classes |
| FtpWebRequest/FtpWebResponse | Implements an FTP client/encapsulates a FTP server’s response to a request |
| HttpWebRequest/HttpWebResponse | Provides an HTTP-specific implementation of the WebRequest/WebResponse classes |

If you want to build protocol specific applications, then .NET provides everything that you need, but why worry about all the gory details when WebClient will handle just about everything you would run across?

Here’s an example program of working with the HTTPWebRequest and HTTPWebResponse objects, just so you can get a taste of what can be done. Here’s the main form with the control names. Note that the Textboxes are all multiline with scrollbars enabled:



Here’s the application’s source code:

'Chapter 18 - Program 4

Imports System.Net

Public Class Form1

Private Sub cmdConnect\_Click(sender As Object, e As EventArgs)

Handles cmdConnect.Click

Dim myWebRequest As HttpWebRequest

Dim myWebResponse As HttpWebResponse

Dim myWebHeaders As WebHeaderCollection

'We need to be able to process a stream that comes back to us

Dim sr As System.IO.StreamReader

'Create a web Request object and set a couple of headers

myWebRequest = WebRequest.Create(txtURL.Text)

myWebRequest.KeepAlive = False

myWebRequest.ContentType = "text/html"

txtRequest.Text = myWebRequest.RequestUri.ToString

'At this point we could grab the Request object's RequestStream

'and send POST information if we wanted...

'Iterate through and print out the headers on the Request

myWebHeaders = myWebRequest.Headers

txtHeaders.Text = "Request Headers: " & vbCrLf

For Each Entry In myWebHeaders.AllKeys

txtHeaders.Text &= Entry & " - " & myWebHeaders(Entry) & vbCrLf

Next

'Get the Response stream, access it with the StreamReader and

'pull all the information off the stream and into our textbox

myWebResponse = CType(myWebRequest.GetResponse(), HttpWebResponse)

sr = New System.IO.StreamReader(myWebResponse.GetResponseStream())

txtResponse.Text = sr.ReadToEnd()

sr.Close()

'Iterate through and print out the headers on the Response

myWebHeaders = myWebResponse.Headers

txtHeaders.Text &= vbCrLf & vbCrLf & "Response Headers: " & vbCrLf

For Each Entry In myWebHeaders.AllKeys

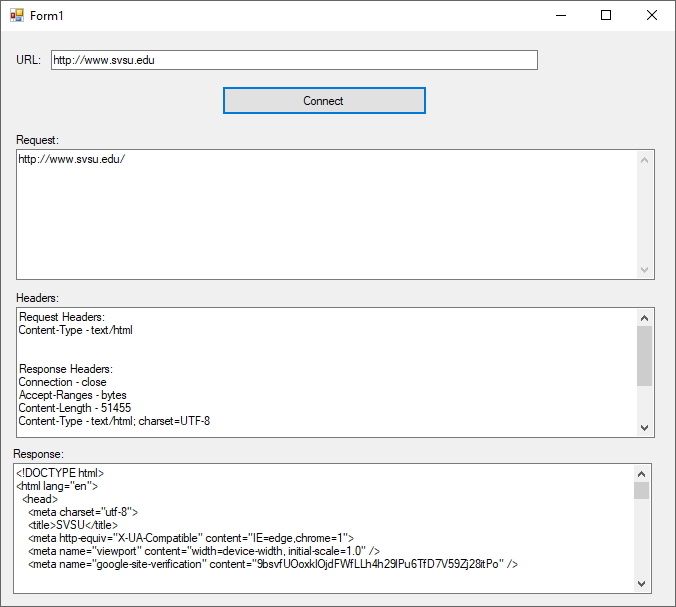
txtHeaders.Text &= Entry & " - " & myWebHeaders(Entry) & vbCrLf

Next

End Sub

End Class

Here’s the program’s output:



# Sending E-Mail Messages

We'll look at two different ways to handle sending e-mail messages. The first method uses the SMTP service that is built into the Windows operating system. VB has provided us with a MailMessage class so it makes life pretty simple.

The following is a list of the MailMessage properties that are you are most likely to use:

|  |  |
| --- | --- |
| *Property* | *Purpose* |
| Attachments | Specifies a list of attachments to be sent with the message |
| Bcc | Gets or sets the System.Net.Mail.MailAddressCollection holding the bcc addresses of the recipients of the message |
| Body | Gets or sets the body message |
| BodyEncoding | Gets or sets the encoding for the body of the message |
| CC | Gets or sets the System.Net.Mail.MailAddressCollection holding the cc addresses of the recipients of the message |
| From | Gets or sets the senders e-mail address (of type System.Net.Mail.MailAddress) |
| IsBodyHtml | Gets or sets whether the body of the message contains HTML |
| ReplyTo | Gets or set the MailAddress containing the reply address for the message |
| Subject | Gets or sets the message subject |
| SubjectEncoding | Gets or sets the encoding for the subject of the message |
| To | Gets or sets an entry into a MailAddressCollection containing the destination addresses for the message |

The only other class that we will need is the SmtpClient class, which allows us to specify what mail server we would like to use to send our message on.

These are the SmtpClient class properties and methods that we will use:

|  |  |
| --- | --- |
| *Property or Method* | *Purpose* |
| ClientCertificates | Specifies the X509Certificate you want added for communication to the SMTP server (if it requires one) |
| Credentials | Gets or sets the credentials used to send the message |
| EnableSsl | Gets or sets whether the SmtpClient should use SSL to communicate with the server |
| Host | Gets or sets the server name |
| Port | Gets or sets the server port |
| Send | Used to send a MailMessage object via an SMTP server |
| Timeout | How long a time-out (in milliseconds) is when trying to send mail |
| UseDefaultCredentials | Gets or sets whether the current default user credentials should be used when communicating with the SMTP server (if credentials are required) |

Let's take a look at a sample console program to send an e-mail message:

'Chapter 18 - Program 5

Imports System.Net.Mail

Module Module1

Sub Main()

'Create a SMTPClient object and a MailMessage object

Dim myServer As New SmtpClient

Dim myMailMsg As New MailMessage

'Provide the name of the server that we want to use

'This may or may not work because SVSU's ITS changes

'the email server and protocol requirements over time...

myServer.Host = "smtp.svsu.edu"

'Fill out the mail message fields that we are interested in

myMailMsg.To.Add("james@csis.svsu.edu")

myMailMsg.From = New MailAddress("james@svsu.edu")

myMailMsg.Subject = "Test Email"

myMailMsg.Body = "Demo email from inside of .NET"

Try

'Try to send the message

myServer.Send(myMailMsg)

Catch e As Exception

'Print out any problems sending

MsgBox("The following exception occurred: " & e.Message)

End Try

MsgBox("Mail message sender ending")

End Sub

End Module

If the program runs successfully, you should not see an exception messages – just the "mail message sender ending" message.

The second technique that you can use to send e-mail messages involves using automation to drive the Microsoft Outlook application. Microsoft Outlook is part of Microsoft Office, so if you don't have Office loaded, this example won't work. Incidentally, you have to use Outlook and not Outlook Express. For some strange reason, Microsoft didn't create an object model that's accessible inside of the Express version.

Here's the sample code for programming Microsoft Outlook to create a message. You will have to add a COM reference to the Microsoft Outlook Object Model. Again, select the newest version available on your computer. This is a Windows forms application with one Command button on the form with the default name:

'Chapter 18 - Program 6

Imports Microsoft.Office.Interop

Public Class Form1

Private Sub Button1\_Click(sender As Object, e As EventArgs)

Handles Button1.Click

'Here's access to the Outlook application

Dim myMailer As New Outlook.Application()

'Create an Outlook MailItem (remember Outlook does more than mail)

Dim item As Outlook.MailItem

'Now we can get to work making our message

item = myMailer.CreateItem(Outlook.OlItemType.olMailItem)

With item

'Which account are we going to try to send the mail from?

.SendUsingAccount = myMailer.Session.Accounts(1)

'Thes should be pretty obvious

.Recipients.Add("james@svsu.edu")

.Subject = "Sample email message"

.Body = "This is a sample message from me to me..."

End With

'Send the message...

item.Recipients.ResolveAll()

item.Send()

myMailer.Quit()

MsgBox("Mail Sent")

End Sub

End Class

As you can see the general pattern for sending email messages using either style is pretty much the same. Where things get a bit handier with using Outlook is that it performs error handling on your behalf, whereas you would have to code against errors when you use the SmtpClient class. Also, Outlook has access to multiple email accounts that you may have set up and can recall and send your email messages through whatever account you prefer.

When trying to build an email application, you may need to get in touch with your systems administrator to try to determine the server, port, protocols and authentication requirements. SMTP used to be unencrypted and blast messages from port 25 – there’s no guarantee that’s the situation today.

# Miscellaneous Network Classes

We’ve looked at the major players that make up the Net namespace. Let’s conclude this section by taking a peek at just a couple more. It’s probably actually easier to wander through some source code and see the various methods and classes at play than to try to explain them first. So, here goes…

Our miscellaneous network console application’s source code:

'Chapter 18 - Program 7

'Our typical network namespace imports statement.

Imports System.Net

'This namespace will provide us information on our network.

Imports System.Net.NetworkInformation

Module Module1

Sub TestNetworkInformationNamespace()

'Create an array of NetworkInterfaces -- your computer

'usually has more than one. For example, you have at

'least one if you have a wired connection and a second

'one if you have a wireless connection

Dim AllNetworkIF() As NetworkInterface

'Can we "see" a network at all?

Debug.WriteLine("Is a network available? : " &

NetworkInterface.GetIsNetworkAvailable)

'Bring back information on all interfaces and print out

'their status:

AllNetworkIF = NetworkInterface.GetAllNetworkInterfaces

For Each NetIF As NetworkInterface In AllNetworkIF

Debug.WriteLine("Interface Name : " & NetIF.Name)

Debug.WriteLine("Description : " & NetIF.Description)

Debug.WriteLine("Physical address : " &

NetIF.GetPhysicalAddress.ToString)

Debug.WriteLine("Status : " & NetIF.OperationalStatus)

Debug.WriteLine("Type : " & NetIF.NetworkInterfaceType)

Debug.WriteLine("Speed : " & NetIF.Speed)

Debug.WriteLine("Bytes sent : " & NetIF.GetIPv4Statistics.BytesSent)

Debug.WriteLine("Bytes received : " &

NetIF.GetIPv4Statistics.BytesReceived)

For Each IPAddr In NetIF.GetIPProperties.UnicastAddresses

Debug.WriteLine("Address : " & IPAddr.Address.ToString &

" lease remaining " &

Now.AddSeconds(IPAddr.DhcpLeaseLifetime))

Next

Debug.WriteLine(StrDup(50, "\*"))

Next

End Sub

Sub TestDNSResolution()

'This routine will let us know if we are able to have

'DNS resolution take place or not.

Dim DNSEntry As IPHostEntry

Dim myList As New List(Of String)

myList.Add("www.svsu.edu")

myList.Add("www.google.com")

myList.Add("localhost")

For Each strQualifiedName In myList

DNSEntry = Dns.GetHostEntry(strQualifiedName)

For Each IPAddr In DNSEntry.AddressList

'The loopback will bring back an IPv4 and IPv6 version

Debug.WriteLine("DNS name : " & strQualifiedName & " Address : " &

IPAddr.ToString)

Next

Next

End Sub

Sub TestPing()

'This routine checks if we can ping our local loopback.

Dim myPing As New Ping

Dim myReply As PingReply

'Ping loopback with a timeout of 100ms

myReply = myPing.Send("127.0.0.1", 100)

If myReply.Status = IPStatus.Success Then

Debug.WriteLine("Successfully pinged loopback")

Else

Debug.WriteLine("Coult not ping loopback")

End If

End Sub

Sub Main()

TestNetworkInformationNamespace()

TestDNSResolution()

TestPing()

End Sub

End Module

Here’s the output that our code generates:

Is a network available? : True

Interface Name : Ethernet

Description : Realtek PCIe GBE Family Controller

Physical address : 9CB7AA653B

Status : 2

Type : 6

Speed : -1

Bytes sent : 0

Bytes received : 0

Address : fe80::54bd:d875:bd4f:b644%5 lease remaining 1/6/2022 1:55:42 PM

Address : 169.254.182.68 lease remaining 1/6/2022 1:55:36 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface Name : Local Area Connection\* 2

Description : Microsoft Wi-Fi Direct Virtual Adapter

Physical address : 5627AC76B7FD

Status : 2

Type : 71

Speed : -1

Bytes sent : 0

Bytes received : 0

Address : fe80::a5d6:2a25:3c9:66ea%15 lease remaining 1/6/2022 1:55:22 PM

Address : 169.254.102.234 lease remaining 1/6/2022 1:55:15 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface Name : Wi-Fi

Description : Broadcom BCM4352HMB 802.11ac 2x2 Wi-Fi Adapter

Physical address : 5427E46576A6

Status : 1

Type : 71

Speed : 144000000

Bytes sent : 80209695

Bytes received : 720615009

Address : fe80::a42a:a0db:45d0:818b%13 lease remaining 1/6/2022 1:55:26 PM

Address : 192.168.1.13 lease remaining 1/6/2022 2:27:01 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface Name : Bluetooth Network Connection 2

Description : Bluetooth PAN HelpText

Physical address : 54271EBE621B

Status : 2

Type : 6

Speed : 3000000

Bytes sent : 0

Bytes received : 0

Address : fe80::9026:249c:8e41:679c%10 lease remaining 1/6/2022 1:55:30 PM

Address : 169.254.103.156 lease remaining 1/6/2022 1:55:24 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface Name : Loopback Pseudo-Interface 1

Description : Software Loopback Interface 1

Physical address :

Status : 1

Type : 24

Speed : 1073741824

Bytes sent : 0

Bytes received : 0

Address : ::1 lease remaining 1/6/2022 1:55:47 PM

Address : 127.0.0.1 lease remaining 1/6/2022 1:55:47 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface Name : Teredo Tunneling Pseudo-Interface

Description : Teredo Tunneling Pseudo-Interface

Physical address : 00000000000000E0

Status : 1

Type : 131

Speed : 100000

Bytes sent : 286144

Bytes received : 2280

Address : 2001:0:9d38:6ab8:18bb:3b53:3f57:fef2 lease remaining 1/2/2022 7:58:27 PM

Address : fe80::18bb:3b53:3f57:fef2%6 lease remaining 1/2/2022 7:58:27 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DNS name : www.svsu.edu Address : 34.195.143.255

DNS name : www.google.com Address : 172.217.1.36

DNS name : localhost Address : ::1

DNS name : localhost Address : 127.0.0.1

Successfully pinged loopback

Bottom line: if there is something network related that you want to try to do, chances are that .NET already has it built in. There are a lot more network related classes that I didn’t cover – your friend the MSDN library will detail them if you are interested. I just showed you what I want you to know for this class…

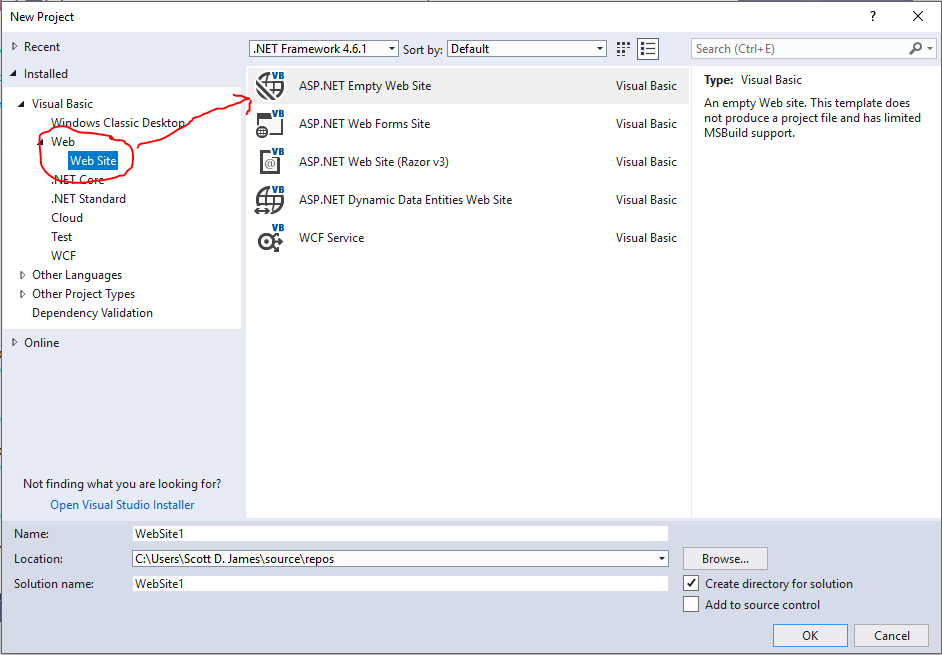
# Introduction to ASP.NET Projects

How much work would it be to port your VB application so that it runs on the web? Sure, we could probably convert a lot of it over to HTML and DHTML and then use some VBScript to start moving it, but that’s still a lot of effort.

VB comes bundled with a way for you to create your applications such that they can be downloaded from a website and run under your favorite browser. This is called an ASP.NET Web Project. I am only going to cover the rudiments of ASP.NET in this section – we could easily devote an entire academic year to really learning what ASP.NET can do for us.

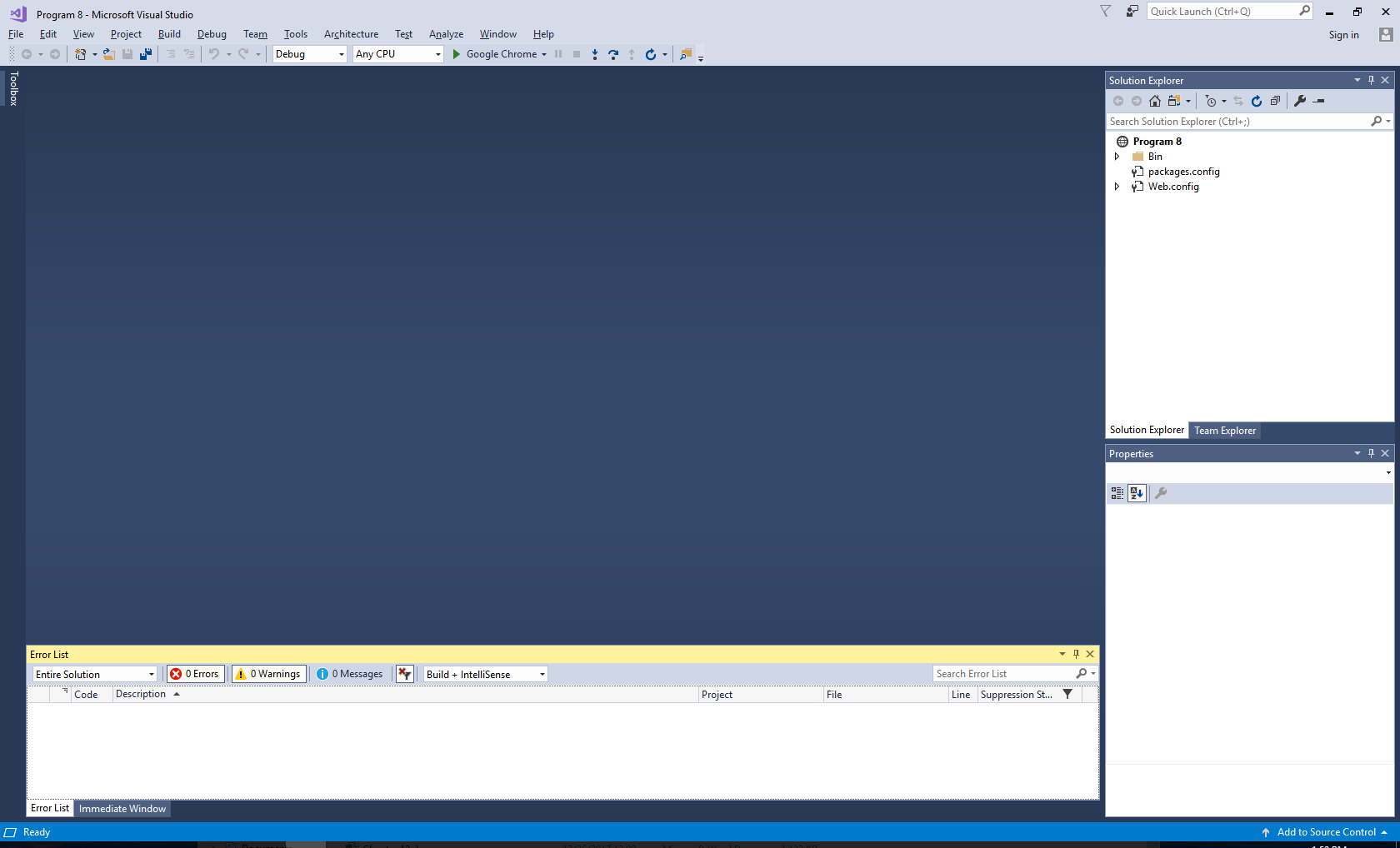
One aspect of ASP.NET makes it a server side language; actually it is Microsoft’s answer to PHP or JSP. Other aspects of ASP.NET like authentication make it much more akin to a framework environment. No matter how you choose to think about ASP.NET one thing is clear: this is how Microsoft wants you to do web development!

Let’s look at writing a simple calculator program that can be run under a web browser. You will start by creating a new project of type ASP.NET Empty Web Site. Note that this is not in the default template types that we have been using up to this point – we are looking at the Web section’s Web Site subsection, not the Windows section!



This project takes longer for VS to construct than a Windows form application, because it has to set things up in such a way that what we create will be run under a web server and sent to a client browser when requested. You may see “localhost” appearing in the client’s URL. If you didn't know, "localhost" is a quick way to refer to whatever machine you are currently working on. You can just let VB do what it needs by default and everything will work fine.

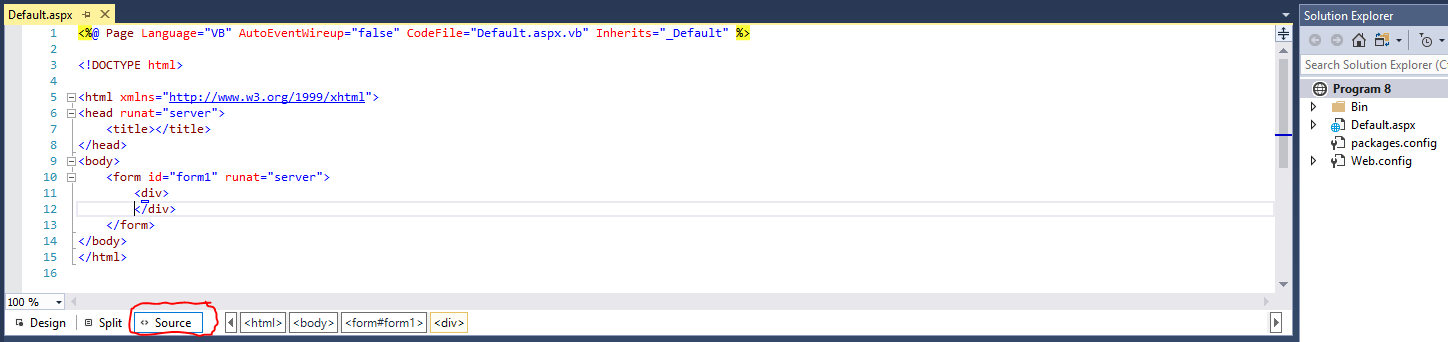
Once your blank project has finished being constructed, you will see that you don’t get your normal form. Instead, you are presented with an interface that doesn’t look very programmer friendly:



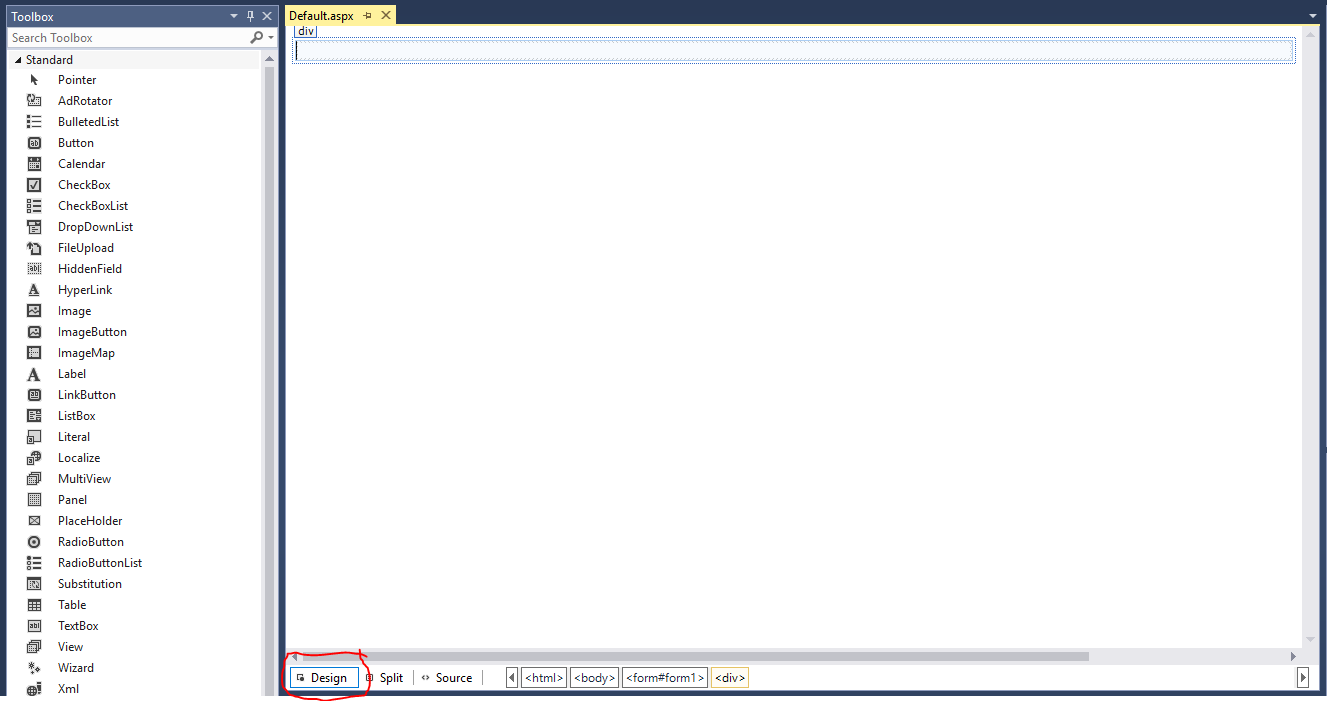
Now, if I had asked you to create an ASP.NET Web Forms Site project, you would have gotten a ton of files, a prebuilt template and a whole lot of stuff that would make you scratch your head and ask where to begin. By creating this empty web project, I can guide you through what ASP.NET is doing behind the scenes – then you can go back and play with a Web Forms Site project if you want.

The good news is that you can still use most of the controls and the other goodies that you are used to in creating your form. Let’s start this out by going under Website🡪Add New Item🡪Web Form. Make sure that you picked Web Form! We’ll leave the default name of the form alone, which is Default.aspx.

This is what we will see when the web form has been created. I clicked on the circled Source button to get this view:

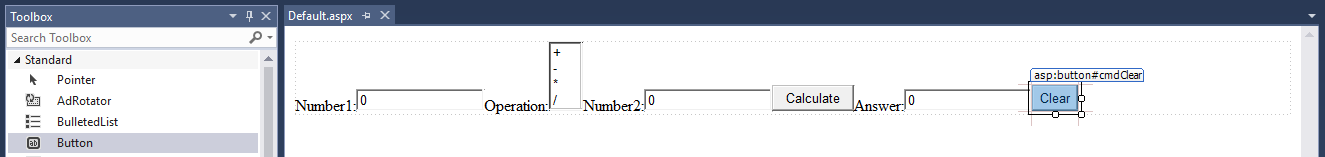


Congratulations! You wanted to do web, you got web – notice that we are basically seeing HTML with some ASP markup. Our controls are available in the left hand pane, so that’s good news, but where is our form? It’s there, we just need to view it. Notice that toward the bottom of the code editor window, there is a Design button (to the left of the Source button) – press it and you’ll see an HTML form:



So now we can start writing our simple calculator program. This will be a calculator with two TextBoxes for numbers (txtNumber1 and txtNumber2), a Listbox (lstOperation) for the four math operations of +, -, \*, / and a third TextBox for the answer (txtAnswer). We will also need two Command buttons: one to compute the answer (cmdCalculate) and the other to clear the textboxes out (cmdClear).

I built the form by simply dragging and dropping controls onto the Web form. About the only issue is that the Web form doesn’t allow you to just position a control anywhere you want because of the way webpage rendering happens (that can be altered through CSS though). I also prepopulated the ListBox with the 4 math functions I am going to allow. Here’s a screenshot of my final web form:



The form is rather ugly, but since you’re doing web, you’re expected to pretty up the form with some CSS later. Right now, our goal is functionality!

I clicked on the Source button at the bottom of the web form again and added some <br /> tags to move things around a bit:

<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="\_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label ID="Label1" runat="server" Text="Number1:"></asp:Label>

<asp:TextBox ID="txtNumber1" runat="server">0</asp:TextBox>

<br />

Operation:<asp:ListBox ID="lstOperation" runat="server">

<asp:ListItem>+</asp:ListItem>

<asp:ListItem>-</asp:ListItem>

<asp:ListItem>\*</asp:ListItem>

<asp:ListItem>/</asp:ListItem>

</asp:ListBox>

<br />

Number2:<asp:TextBox ID="txtNumber2" runat="server">0</asp:TextBox>

<br />

<asp:Button ID="cmdCalculate" runat="server" Text="Calculate" />

<br />

<asp:Label ID="Label2" runat="server" Text="Answer:"></asp:Label>

<asp:TextBox ID="txtAnswer" runat="server">0</asp:TextBox>

<br />

<asp:Button ID="cmdClear" runat="server" Text="Clear" />

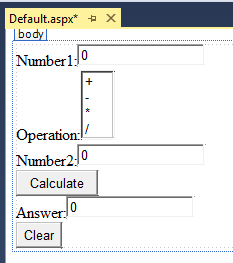
</div>

</form>

</body>

</html>

It’s still ugly, but as you can see we’re getting there:



Now we’re off to write code – you can double click on the Command buttons to get to your event handlers just like you did with a Windows form. As you can see, there is no bizarre code behind our code: it looks just like straight VB! Here’s the full code behind the Web form:

'Chapter 18 - Program 8

Partial Class \_Default

Inherits System.Web.UI.Page

Protected Sub cmdCalculate\_Click(sender As Object, e As EventArgs)

Handles cmdCalculate.Click

Select Case lstOperation.SelectedValue

Case "+"

txtAnswer.Text = CSng(txtNumber1.Text) + CSng(txtNumber2.Text)

Case "-"

txtAnswer.Text = CSng(txtNumber1.Text) - CSng(txtNumber2.Text)

Case "\*"

txtAnswer.Text = CSng(txtNumber1.Text) \* CSng(txtNumber2.Text)

Case "/"

txtAnswer.Text = CSng(txtNumber1.Text) / CSng(txtNumber2.Text)

End Select

End Sub

Protected Sub cmdClear\_Click(sender As Object, e As EventArgs)

Handles cmdClear.Click

txtNumber1.Text = "0"

txtNumber2.Text = "0"

txtAnswer.Text = "0"

End Sub

End Class

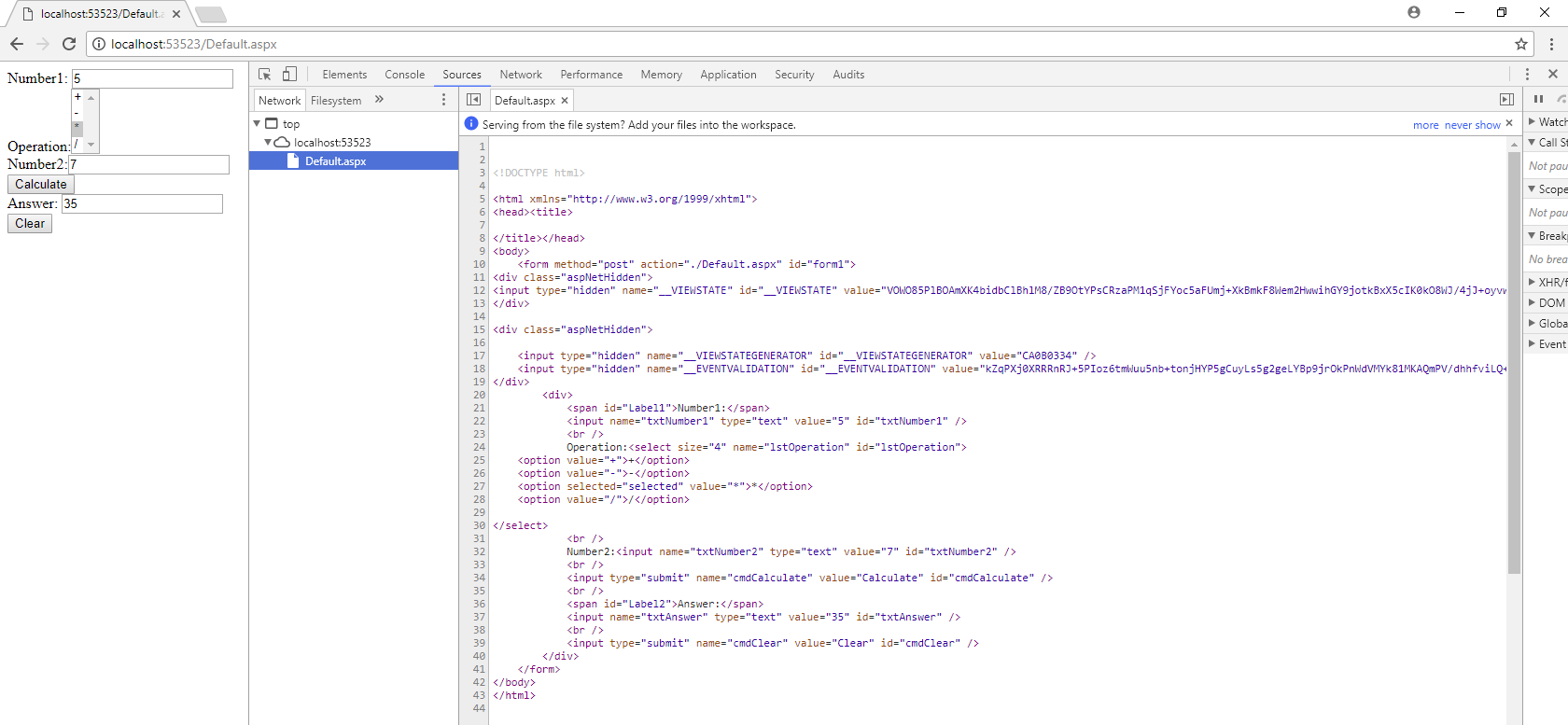
At this point, you're probably wondering just what the big difference is here? We certainly didn't seem to do anything new with our coding. Your payoff in is the fact that this application can be run under a web browser by users on the Internet. How's that for cool?

Here's a screenshot of the application once I’ve clicked on run:



First of all, this is running under Google Chrome – it’s not a normal Windows form application. Notice the aspx extension in the URL line? This lets you know that it’s an ASP application, and as such, it must be served by an ASP server such as Microsoft's Internet Information Services (IIS) Server. If you work at a company that uses Microsoft applications and servers, this might be a very quick and easy way to create web based application access… You can also pay for ASP hosting services by third party companies on the Internet.

Do realize that the actual execution of the code happens over on the server: ASP.NET is a *server*-side language. If we look at the source code behind the page in our browser, there’s no execution logic there at all:



Of course, this is just the tip of the iceberg of ASP.NET. You really need to do a lot of reading to become proficient with it and to understand what it will do for you. As you can tell, there are a whole bunch of things that must happen when VB generates a web project. There are many books available on just this topic; check one out for details. ASP.NET jobs are very plentiful, so it may be the de facto web technology that you want to invest in learning. Plus, you get to use your most favorite language and IDE in the world to develop in it!