On systems with .NET Framework installed, you can create and deploy WebService applications without using Visual Studio. Use any text editor (such as Notepad). Place your WebServices source code in text files with the file extension".asmx" in an IIS file folder. If you are running IIS locally on your workstation (and you have installed the .NET Framework), this is just saving the file to a suitable location on your local drive (e.g. "C:\inetpub\wwwroot\").

By default Internet Information Services (IIS) passes requests for only certain file types to ASP.NET for processing and serving. Files with file name extensions such as ".aspx", ".asmx", and ".ashx", are default mapped to the ASP.NET ISAPI extension (Aspnet_isapi.dll).

Instead of the ".aspx" file extension of an ASP.NET Web page, the WebServices are kept in files with the extension of ".asmx". Whenever the ASP.NET runtime receives a request for a file with an ".asmx"-extension", the runtime dispatches the call to the WebService handler.

The very first time an ASP.NET page (".aspx"-files), or WebService (".asmx"-files, usually), is requested by a client, the file is compiled to Microsoft Intermediate Language (MSIL) code. The code is then run by the server, and the output is sent to the client.

The example below lists the code for a WebService version of the "Hello World" application. It delivers the familiar string through an exposed method named HelloWorld(). To identify the class and method as a WebService to the compiler, the code uses some special directives. It also has an ASP.NET directive at the beginning of the file.

File "Hello.asmx" (access e.g. "http://www.mcduck.com/Hello.asmx"):

```
<%@ WebService Language="C#" Class="HelloWorldService" %>

using System.Web.Services;

public class HelloWorldService: WebService
{
        [WebMethod]
        public string HelloWorld()
        {
                return "Hello World";
        }
}
```

See article for more descriptions:
http://www.asp101.com/articles/colin/webservices/default.asp

Install IIS:

1. Goto the “Control Panel”
2. Start “Add/Remove Programs”
3. Click “Add/Remove Windows Components”
4. Select “Internet Information Services” -> follow procedure

Create a folder beneath the root folder of IIS web server, e.g .“C:\Inetpub\wwwroot\”, e.g. “MyASPdotNETServices”.

Map the extension “.asmx” to ASP.NET:

1. Start “Internet Services Manager” trough “Control Panel -> Administrative Tools”.
2. Select the folder beneath “Default Web Site” intended to hold the Web Service. Click right button and select “Properties” for the folder. Click the button “Configuration” at the tab “Directory”, or “Virtual Directory” (for the top folder the tab is named “Home Directory”).
3. Map the extension “.asmx” to the “aspnet_isapi.dll” mapper, eg “C:\WINNT\Microsoft.NET\Framework\v1.1.4322\ aspnet_isapi.dll”.

Make the folder holding the application an application folder:

1. Start “Internet Services Manager” trough “Control Panel -> Administrative Tools”.
2. Select the folder beneath “Default Web Site” holding the Web Service. Click right button and select “Properties” for the folder. Click the button “Configuration” at the tab “Directory”, or “Virtual Directory” (for the top folder the tab is named “Home Directory”).
3. Push the button “Create” and set an application name.

Configure ASP.NET:

1. Start a “cmd” window from “Start->Run…”
2. Change directory to the .NET distribution, eg “C:\WINNT\Microsoft.NET\Framework\v1.1.4322\”
3. Run the command “aspnet_regiis –ir”

MS short introduction step-by-step setup:

http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/q308/1/64.asp&NoWebContent=1

IE as WebService client (http://msdn.microsoft.com/en-us/library/ms531035(v=VS.85).aspx):
Use the WebService behaviour "webservice.htc" file that encapsulates SOAP etc from IE.

Enables calling WebServices by means of scripting:

```
iCallID = myService.MyMath.callService("add",int1,int2);
```

Steps:

1. Place "webservice.htc" file in the same folder as the web pages in question
2. Attach the WebService behaviour

```
<body>

<div id="myservice" style="behavior:url(webservice.htc)"></div>

</body>
```

3. Identify the WebService by mapping the Web Service URL (to its WSDL description) to a FriendlyName:

```
myservice.useService("/services/math.asmx?WSDL","MyMath");
```

Stand alone WebService client:

1. Generate a Proxy for the Web Service using the command line tool "wsdl.exe", or Developer Studio, by reading the WSDL description from a file, or query the service directly using the network.
2. Compile the Proxy as a DLL library, or assembly
3. Develop the Client

Directory for SOAP developers:
http://www.soapware.org

Understanding SOAP:
http://msdn.microsoft.com/en-us/library/ms995800.aspx

The Birth of Web Services:
http://msdn.microsoft.com/en-us/magazine/cc188933.aspx