# Leverage .NET Web Services and Domino


**XML Web services and Microsoft .NET are two hot technologies. Expand your Lotus programming horizons by tapping into their power.**

By John Duggan

**VERSIONS**

- Lotus Domino 5.0.2c+
- Lotus Notes 5.0.2c+
- Microsoft Internet Explorer 5.01+
- Microsoft Windows 2000/XP Professional

**TECHNOLOGIES**

- Microsoft .NET Framework Software Development Kit
- Internet Information Server 5.0
- C#

**RESOURCE CD**

Article ID: DUGGJ08

**DOWNLOAD**

Subscribers can get this article's download at http://Advisor.com/Article/DUGGJ08

You've probably heard a lot about Web services along with Microsoft .NET, and wondered what's it all about. You may have read articles describing Web services and why it's important, including "Share the Wealth with Web Services & XSLT" by Rose Kelleher and Bryan Palmer in the June 2002 issue. This article shows you how to create a Web service in the .NET Framework and integrate it into a Domino application using several techniques.

A few years ago, I developed a workflow application for Notes R4 that contained a Java chart displaying a yearly summary of how many workflow documents have been approved in a month (figure 1).

The users found the information so useful they wanted their clients to be able to view the chart. The first alternative was an extranet application that let external users view the chart via a Web browser. This meant the external users required a Web browser with a Java plug-in, which not all users had. The second alternative was to generate the chart on the server and send it as an e-mail attachment, but some users weren't able to receive attachments, others had mail quotas, and many other technical constraints prevented users from receiving the chart via e-mail.

Each proposed solution had requirements the users' environments may not have met. Another possibility was to provide users with the data and let their IT departments find a way to present it. At the time, this wasn't feasible. However, with Web services, now it's fairly straightforward to solve the challenge of this scenario.

This article focuses on Web services within a .NET environment. Many Microsoft-based organizations are migrating to .NET, and you can easily integrate Lotus Notes/Domino applications with Microsoft development tools including the .NET Framework.

## Overview of Web services

First, I'll provide a brief overview of Web services. If you'd like additional information, see the Resources section on page 18.

Web services lets applications from different environments work together regardless of where they're physically located or how they're implemented. Web services is built upon a number of standards, including XML, HTTP, SOAP, WSDL, and UDDI. Web services applications use the HTTP protocol and SOAP messages to communicate with each other.

### Microsoft .NET

In a nutshell, the objective of .NET is to provide software as a service, which is a radical change for Microsoft. Microsoft is re-inventing itself with a range of products, such as the .NET Framework, Visual Studio .NET, and Windows .NET.

### Writing a .NET Web service

Because this article demonstrates a .NET example, you'll need to have:

- .NET Framework Software Development Kit (http://msdn.microsoft. com/netframework)
- Microsoft Windows 2000 or XP professional
- Internet Information Server (IIS), which is included with Windows and is installed under the Add/Remove Windows Components option within the control panel
- Microsoft XML parser, included with Internet Explorer 5.01 or later

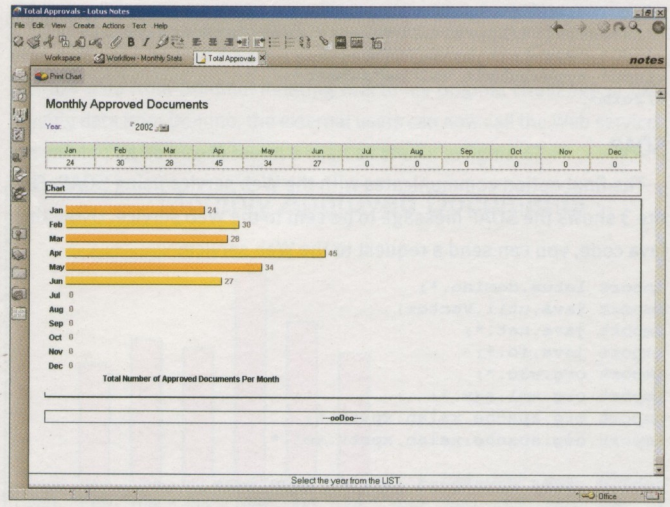


***Figure 1:* Java applet—Shows the number of approved workflow documents per month.**

- Lotus Domino or Notes 5.0.2c or above

You can find more information about the installation of .NET and Domino in my article "Domino and .NET" in the June 2002 issue of ADVISOR EXPERT ON LOTUS NOTES AND DOMINO R5.

Fortunately, you can readily integrate Domino and IIS, and you can find further information about setting them up in the Domino Administration Help. I've written all the Web services in this article in C#, which is similar to Java. An advantage of .NET is that you can write a Web service in almost any programming language.

I used a standard text editor to write and implement all the samples for this article. A better alternative is Visual Studio .NET, but if you don't have access to it, I recommend ASP.NET Web Matrix (http://www.asp.net/webmatrix/). You can use this free WYSIWYG editor to create Web services, ASP pages, connections to SQL databases, mobile applications, and more.

This example shows how to create a simple Web service that returns the number of approved documents for a particular month. The next example shows how to incorporate Notes data into a chart that displays in a browser, solving the problem in the scenario I described earlier.

When the Web service is called, it:

1. Opens a local Notes database, webservice.nsf (which you can download from http://Advisor.com/Article/DUGGJ08)
2. Opens a hidden view showing all documents categorized by month
3. Returns the number of documents within that category.

Here's the complete code for this Web service:

```
// Tell the compiler to use C# and the class
// name.
<%@ WebService Language="C#" class="Approved-
NotesDocumentsWebService"%>

using System;
using System.Web.Services;
// Important: This tells the compiler to use
// the Domino classes.
using domobj;

[WebService(Namespace="http://127.0.0.1/")]
// The class you're using for your Web service,
// which is inherited from the WebService class.
public class
ApprovedNotesDocumentsWebService : WebService
{

// A special identifier that states the method
// will be available for Web service clients.
[WebMethod]
public int GetMonthlyStats(string Month)
{
NotesSession session = new NotesSession();
//Remember to put your password in here, otherwise
//it won't work!
 session.Initialize("lotusnotes");
 NotesDatabase db = session.GetDatabase
      ("", "webservice.nsf",false);
 NotesView view = db.GetView("(By Month)");
 NotesDocumentCollection dc = view.GetAllDocumentsByKey
       ("2002" + Month,true);

 return dc.Count;  //Return the total count.
 }
}
```

Save this code in a text file with the extension .asmx, such as Simple.asmx. You should save the file within the IIS Web directory path, such as c:\inetpub\wwwroot\webservices.

Before running the Web service, you have to create the domobj.dll, because the Domino classes aren't included as standard within the .NET framework. (The details are provided in my .NET and Domino article, therefore I'll just give an outline here.) From the command prompt, go to the Notes directory and type:

```
tlbimp domobj.tlb /out:domobj.dll
```

Copy the DLL into the bin directory of your Web application to let IIS find the DLL. (Note: If the bin directory doesn't exist, you should create it.)

```
copy domobj.dll c:\inetpub\wwwroot\bin\domobj.dll
```

If you load up the Web browser and type the URL http://127.0.0.1/webservices/simple.asmx, you'll see the screen in figure 2.

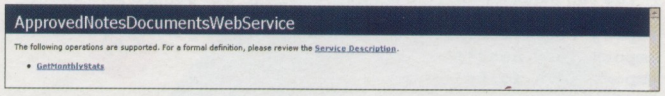


***Figure 2:* Web service—The simple .NET Web service within a Web browser.**

You can see your available function GetMonthyStats. If you click on it, you'll get the screen in figure 3.

The pages in figures 2 and 3 are automatically created by the .NET Framework, and give lots of useful information, as you'll see later. Type in a value for the month and click on the Invoke button. Another screen is returned with the following XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="http://127.0.0.1/">24</int>
```

You now have a simple Web service running on the .NET framework returning the number of Notes documents approved for a specific month.

## Call the Web service programmatically

There are three ways to call this Web service: HTTP GET, HTTP POST, and SOAP. The output returned from figure 3 gives you all the information you need to call this Web service programmatically.

*Continued*

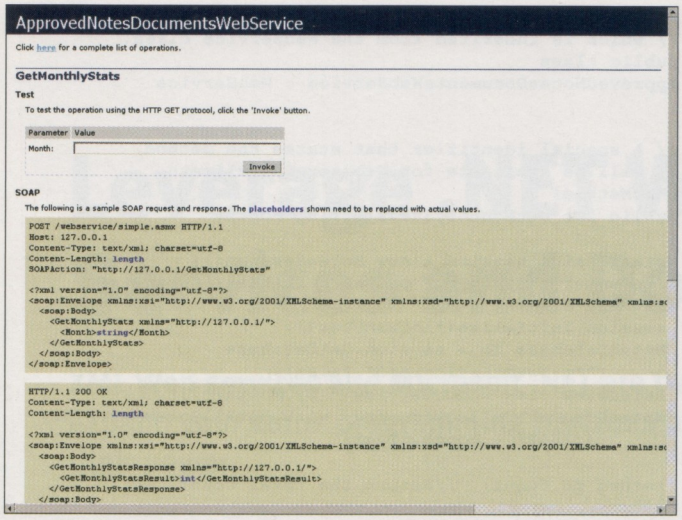


***Figure 3:*** **Test the Web service**—**The shaded boxes show what you can pass and return from the Web service.**

## HTTP GET

The first method is the simplest in that you can pass a URL and the data is returned via XML. To try this, type the following into the browser URL:

```
http://127.0.0.1/webservice/Simple.asmx/GetMonthlyStats?Month=1
```

From LotusScript, you can call the Web service using HTTP GET:

```
'Get a handle to the MS XML parser.
Set source = CreateObject("Microsoft.XMLDOM")
source.async = False
source.load(http://127.0.0.1/webservice/ &_
"ApprovedDocuments.asmx/GetMonthlyStats?Month=1")

Set objRoot = source.documentElement
Set objField = objRoot.selectSingleNode
    ("NumApproved")
Print objField.Text
```

You can also call the Web service with HTTP GET using a Java agent:

```
import lotus.domino.*;
import java.net.*;
import java.io.*;

public class JavaAgent extends AgentBase
{
 public void NotesMain()
 {
  try
  {
   Session session = getSession();
   AgentContext agentContext = session.getAgentContext();

   URL url = new  URL
 ("http://127.0.0.1/webservice/"+
   "ApprovedDocuments.asmx/GetMonthlyStats?Month=1");
   URLConnection con = url.openConnection();
   BufferedReader br = new BufferedReader(new
   InputStreamReader(con.getInputStream()));
  while(br.ready())
   {  System.out.println(br.readLine());  }
  }
  catch(Exception e)
  {
   e.printStackTrace();
  }
 }
}
```

## HTTP POST

The second option is HTTP POST, which posts data to the Web service. Using the following HTML within a Web page or Domino form, you can get your values back from the Web service:

```
<form method=postaction
="/webservice/ApprovedDocuments.asmx/GetMonthlyStats">
<input type=text name=Month>
<input type=submit value="Submit">
</form>
```

## SOAP

The final option communicates with the Web service using SOAP. Figure 3 shows the SOAP message to be sent to the Web service. Using this Java code, you can send a request to the Web service:

```
import lotus.domino.*;
import java.util.Vector;
import java.net.*;
import java.io.*;
import org.w3c.*;
import org.xml.sax.*;
import org.apache.xalan.xpath.*;
import org.apache.xalan.xpath.xml.*;

public class JavaAgent extends AgentBase
{
 public void NotesMain() {
 try {
  Session session = getSession();
  AgentContext agentContext = session.getAgentContext();

  // Build up the SOAP message into a string.
  // This is required so you can work out the size
  // to put into the content-length header.
  StringBuffer SOAPmsg = new StringBuffer
  ("<?xml version=\'1.0\' encoding=\'UTF-8\'?>");
  SOAPmsg.append("<soap:Envelope"+
"xmlns:soap=\http://schemas.xmlsoap.org/soap/envelope/\"+
"xmlns:xsi=\http://www.w3.org/2001/XMLSchema-instance\"+
"xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\">");
  SOAPmsg.append("<soap:Body>");
  SOAPmsg.append
    ("<GetMonthlyStats xmlns=\"http://127.0.0.1/\">");
  SOAPmsg.append("<Month>1</Month>");
  SOAPmsg.append("</GetMonthlyStats>");
  SOAPmsg.append("</soap:Body>");
  SOAPmsg.append("</soap:Envelope>\r\n");

  // Create a connection to the Web service.
URL url = new URL

("http://127.0.0.1/webservice/ApprovedDocuments.asmx");
  URLConnection con = url.openConnection();
  con.setDoOutput (true);

  // Put in the length of the SOAP message.
  con.setRequestProperty ("Content-Type", "text/xml");
  con.setRequestProperty
 ("Content-Length", Integer.toString (SOAPmsg.length()));
  con.setRequestProperty
("SOAPAction","\"http://127.0.0.1/GetMonthlyStats\"");

  // Send everything to the Web service.
  OutputStream out = con.getOutputStream();
  out.write(SOAPmsg.toString().getBytes());
  out.flush();
  out.close();

  // Read the response.
  BufferedReader br = new BufferedReader(new InputStream-
Reader(con.getInputStream()));
  while(br.ready())
  {  System.out.println(br.readLine());  }
  } catch(Exception e)
  {  e.printStackTrace();  }
 }
}
```

If you're a LotusScript developer, you should install the Microsoft SOAP Toolkit from http://msdn.microsoft.com/soap/. It's the simplest solution and only takes three lines of code:

```
Set mySOAPClient = CreateObject("MSSOAP.SoapClient")
mySOAPClient.mssoapinit
"http://127.0.0.1/webservice/Simple.asmx?WSDL"
Print mySOAPClient.GetMonthlyStats(1)
```

## Charts

You've seen how to create a simple .NET Web service and how it can retrieve data from Domino. Relating this to my original challenge of displaying data from Domino, the external users can now call the Web service, but wouldn't it be nice to return a chart as shown in figure 4?

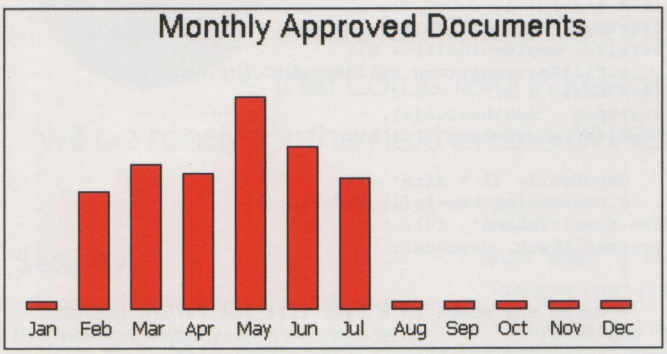


***Figure 4:* Display a chart with Notes data—The Web service generated this simple chart.**

How can you return a chart when you can only return XML? In two ways:

1. Create the chart locally on the Web server and return the URL within the XML that points to the newly created chart on the local Web server.
2. Create the chart and return it as bytes within the XML to the browser.

The .NET Framework is powerful and will let you generate charts within a Web service. You can add the following method to your original service:

```
[WebMethod(Description="This will
create a chart and return a URL
to the chart.")]
public string GetImageURL()
{
 // Create a new bitmap object.
 Bitmap bmp = new Bitmap(450,250);
 Graphics g =
Graphics.FromImage(bmp);
 g.Clear(Color.White);

 // Constants for the boxes and descriptions.
int size=35; int l=15; int h=5;
int startpos=200; int barsize=20; int scale =3;

 // Connect to the Notes database.
 NotesSession session = new NotesSession();
 session.Initialize("lotusnotes");
NotesDatabase db = session.GetDatabase
("", "webservice.nsf",false);
 NotesView view = db.GetView("(By Month)");

 // Put the values from each month into an array.
 int[] monthvalue = new int[12];
 for (int i=0;i<monthvalue.Length;i++)
 {
   NotesDocumentCollection dc =
      view.GetAllDocumentsByKey("2002" + i,true);
   monthvalue[i]=dc.Count*scale;
 }

  // Set up the labels.
  String[] Labels = new String[12] {"Jan",
    "Feb","Mar","Apr","May","Jun",
    "Jul","Aug","Sep","Oct","Nov","Dec"};

 // Write the chart title.
 g.DrawString("Monthly Approved Documents",
```

*Continued*

```
new Font(FontFamily.GenericSansSerif,16),
new SolidBrush(Color.Black), new PointF(100,5));

  // Draw the bars and descriptions for each month.
  PointF descPos = new PointF(360, 210);
  for (int i=0;i<monthvalue.Length;i++)
  {
   g.DrawRectangle(Pens.Black,
(i * size) + 1,
startpos - monthvalue[i],
barsize, monthvalue[i] + h);
   g.FillRectangle(new SolidBrush(Color.Red),
(i * size) + 1,
startpos - monthvalue[i],
barsize, monthvalue[i] + h);

   descPos.X= (i * size) + 1;
   g.DrawString(Labels[i].ToString(),
new Font("Tahoma", 10),
Brushes.Black, descPos);
   }

  // Write the chart to a text file and return the URL.
  bmp.Save(
@"C:\Inetpub\wwwroot\approveddocschart.gif",
ImageFormat.Gif);
  return "http://127.0.0.1/approveddocschart.gif";

  // Use the following code to return the bytes.
  //MemoryStream mem = new MemoryStream();
  //bmp.Save(mem,ImageFormat.Gif);
  //byte[] ImageBytes = ms.GetBuffer();
  //return ImageBytes;
}
```

## RESOURCES

### Production Web Services

http://www.xmethods.com/

### .NET resources, Web Services & Samples

http://www.gotdotnet.com

### .NET Framework

http://www.microsoft.com/net

### Microsoft SOAP Toolkit

http://msdn.microsoft.com/soap/

### Microsoft XML Parser

http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/772/msdncompositedoc.xml

### Web Services Articles from Developerworks

http://www.ibm.com/developerworks/webservices

### Advisor Expert on Lotus Notes and Domino R5

- "Develop and Implement a Web Service for Apache SOAP and Domino," November 2001 (Web services using Apache SOAP)
- "Domino Standards Support Continues: UDDI and WSDL," December 2001 (Overview)
- "Domino and .NET: Leverage these Complimentary Technologies," June 2002 (Using .NET with Domino)

### Lotus Advisor

"Share the Wealth with Web Services & XSLT," June 2002 (Web services using WebSphere).

### WebSphere Advisor

Stay tuned for a new article from Rose Kelleher on practical Web services you can use now.


Although some of the Web services supporting technologies are still in their infancy, the good news for Notes/Domino developers is that Lotus/IBM is keeping pace and will be ready for the next phase of Web services.

## Security

Because the Web service is available over the HTTP protocol, anybody with a Web browser can access the data. The data in the charts contains confidential information, so I had to restrict who can view the information.

There were several ways to solve the security problem:


- Implement secure sockets layer (SSL), which encrypts the HTTP communication between the Web service and the Web service client.
- Restrict the network access by only letting certain IP addresses access the Web service.
- Turn on standard HTTP authentication, which will ask for a username and password and authenticate it with the local users and groups.
- Implement your own authentication logic within the Web service.

The last option means changing the Web service to have the following functions: getMonthlyStats(token,month) and logon(username, password). The logon function will return a token, which has to be passed to the ge MonthlyStats function, that will check whether the token is valid and return the data, otherwise reject the request.


There are XML security tools, such as the IBM XML Security Suite, which allows XML digital signatures, individual parts of the XML to be encrypted, and XML access controls. You can find out more about it at http://www.alphaworks.ibm.com/tech/xmlsecuritysuite.

The main security standard is WS-Security (http://www-106. ibm.com/developerworks/library/ws-secure/), developed by IBM, Microsoft, and VeriSign. (Sun has acknowledged support for it as well.) This standard defines a number of security extensions to the SOAP protocol. It has been submitted as an open standard to Organization for the Advancement of Structured Information Standards (OASIS, http://www.oasis open.org) and is awaiting approval. A number of new standards are expected to arrive as a result of this approval, so keep an eye on future ADVISOR articles for more information.

## Wrap up

This article demonstrated how to create and implement a simple Web service using Domino and the .NET Framework. Web services is a broad subject, and the supporting technologies are evolving rapidly.

This article didn't cover WSDL, transactions, messaging, and operational management, however you can obtain additional information on these subjects in LOTUS ADVISOR, WEBSPHERE ADVISOR, and other sources for both concepts and code.

Although some of the Web services supporting technologies are still in their infancy, the good news for Notes/Domino developers is that Lotus/IBM is keeping pace and will be ready for the next phase of Web services. ADVISOR