[*Authors Note*: This article is an excerpt from Chapter 21, Sections 21.1 and 21.2 of *C# How to Program* and *Visual Basic .NET How to Program, Second Edition*.]

## 21.1 Introduction[1] To ASP .NET and Web services

Creating dynamic link libraries (DLLs—a DLL contains compiled classes and methods) facilitates software reusability and modularity—the cornerstones of good object-oriented programming. However, DLLs are limited by the fact that they must reside on the same machine as the programs that use them. This article introduces Web services (sometimes called *XML Web services*) and how they promote software reusability in distributed systems.

Distributed-systems technologies allow applications to execute across multiple computers on a network. A Web service is a software component that enables distributed computing by allowing an application on one computer to call methods that reside on other computers via common data formats and protocols, such as XML and HTTP. In .NET, these method calls are implemented using the *Simple Object Access Protocol (SOAP)*—an XML-based protocol that describes how to mark up requests and responses so they can be transferred via protocols such as HTTP. Using SOAP, applications represent data with XML. The underlying implementation of the Web service is irrelevant to clients of that Web service.

Microsoft is encouraging software vendors and e-businesses to deploy Web services. As more people worldwide connect to the Internet, applications that call methods across the Internet become more practical. Web services represent the next step in object-oriented programming: Instead of developing software from a small number of class libraries provided at one location, programmers can access countless libraries in multiple locations.

Web services also facilitate business collaboration. By purchasing Web services that are relevant to their businesses, companies that create applications can spend less time coding and more time developing new products from existing components. In addition, e-businesses can employ Web services to provide customers with enhanced shopping experiences. As a simple example, consider an online music store that enables users to purchase music CDs or to obtain information about artists. Now, suppose another company that sells concert tickets provides a Web service that determines the dates of upcoming concerts by various artists and allows users to buy concert tickets. By licensing the concert-ticket Web service for use on its site, the online music store can sell concert tickets to its customers, which likely will result in increased traffic to its site. The company that sells concert tickets also benefits from the business relationship. In addition to selling more tickets, the company receives revenue from the online music store in exchange for licensing the Web service.

Visual Studio and the .NET Framework provide a simple way to create Web services. This article explores creating and accessing Web services. For this example, we provide the code for the Web service, then show an example of an application that might use the Web service. Our example is designed to offer a brief introduction to Web services and how they work in Visual Studio.

1. Internet Information Services (IIS) must be running to create a Web service in Visual Studio.

A Web service is a software component stored on one computer that can be accessed by another computer over a network. The computer on which the Web service resides commonly is referred to as a *remote machine*. The application that accesses the Web service sends a method call to the remote machine, which processes the call and sends a response back to the application. Such distributed computing benefits various systems. For example, a Web service might provide limited access to data that is not normally accessible directly from a client. Or, a Web service might perform a computing-intensive task for a client application residing on a less-powerful computer.

A Web service, in its simplest form, is a method of a class. Normally, when we want to include a class in a project, we either define the class in our project or add a reference to the compiled DLL. This compiled DLL is placed in the **`bin`** directory of an application by default. As a result, all pieces of our application reside on one computer. When using Web services, the class (and its compiled DLL) we wish to include in our project are stored on a remote machine—a compiled version of this class is not placed in the current application.

Methods in a Web service are invoked remotely using a *Remote Procedure Call* (*RPC*). These methods, which are marked with the ***`WebMethod`*** attribute, often are referred to as *Web-service methods*. Declaring a method with this attribute makes the method accessible to other classes via an RPC. The declaration of a Web-service method with attribute **`WebMethod`** is known as *exposing* the method—i.e., enabling it to be called remotely.

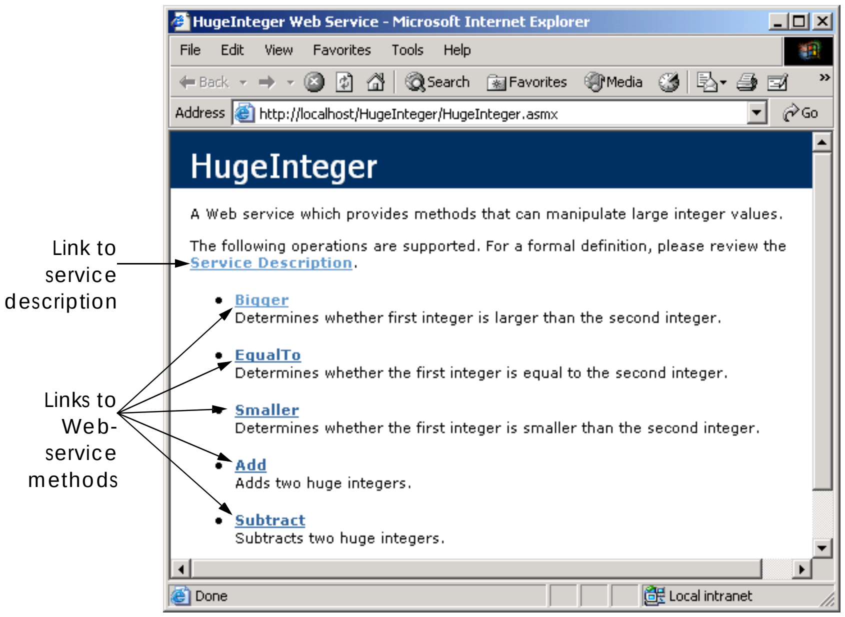
**Common Programming Error 21.1**

*Attempting to call a remote Web service method if the method is not declared with the* ***`WebMethod`*** *attribute is a compilation error.*

Most requests to and responses from Web services are transmitted via SOAP. This means that any client capable of generating and processing SOAP messages can use a Web service, regardless of the language in which the Web service is written.

Web services have important implications for *business-to-business* (*B2B*) *transactions*—i.e., transactions that occur between two or more businesses. Now, instead of using proprietary applications, businesses can conduct transactions using simpler and more efficient Web services. Web services and SOAP are platform- and programming-language independent, so companies can collaborate and use Web services without worrying about the compatibility of various technologies or programming languages. In this way, Web services are an inexpensive, readily-available solution that facilitates B2B transactions.

A Web service created in Visual Studio .NET has two parts—an *ASMX* file and a *code-behind* file. The ASMX file, by default, can be viewed in any Web browser and contains valuable information about the Web service, such as descriptions of Web-service methods and mechanisms to test these methods. The code-behind file provides the implementation for the methods that the Web service encompasses. Figure 21.1 depicts Internet Explorer rendering an ASMX file.


HugeInteger Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/HugeInteger/HugeInteger.asmx
Go
Done
Local intranet
Link to service description
Links to Web-service methods

HugeInteger

A Web service which provides methods that can manipulate large integer values.

The following operations are supported. For a formal definition, please review the Service Description.

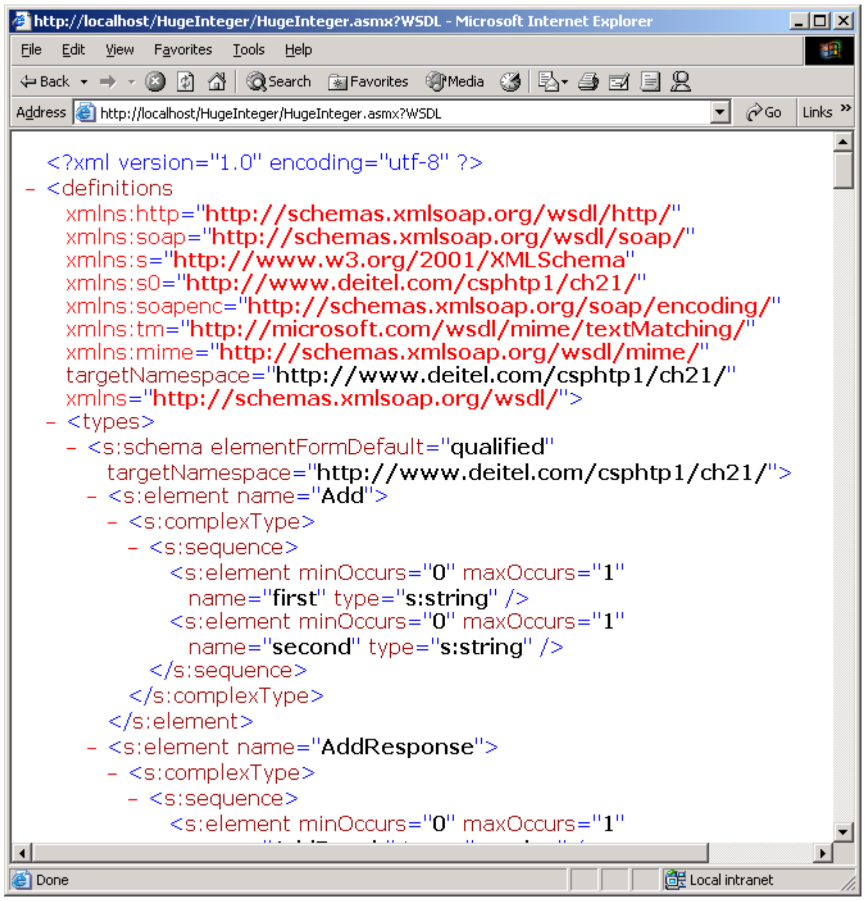
- Bigger
  Determines whether first integer is larger than the second integer.
- EqualTo
  Determines whether the first integer is equal to the second integer.
- Smaller
  Determines whether the first integer is smaller than the second integer.
- Add
  Adds two huge integers.
- Subtract
  Subtracts two huge integers.

Fig. 21.1 ASMX file rendered in Internet Explorer.

The top of the page provides a link to the Web service's ***Service Description***. A service description is an XML document that conforms to the *Web Service Description Language* (*WSDL*)—an XML vocabulary that defines the methods that the Web service makes available and specifies how clients interact with those methods. The WSDL document also specifies lower-level information that clients might need, such as the required formats for requests and responses. Visual Studio .NET generates the WSDL service description. When the client programs are compiled, the compiler can use the service description to confirm that the client program invokes the Web service methods correctly.

The programmer should not alter the service description, as it defines how clients use the Web service. When a user clicks the **Service Description** link at the top of the ASMX page, WSDL is displayed that defines the service description for this Web service (Fig. 21.2).

Below the **Service Description** link, the Web page shown in Fig. 21.1 lists the methods that the Web service provides (i.e., all methods in the application that are declared with **`WebMethod`** attributes). Clicking any method name requests a test page that describes the corresponding method (Fig. 21.3). After explaining the method's arguments, the test page allows users to test the method by entering the proper parameters and clicking **Invoke**. Below the **Invoke** button, the page displays sample request and response messages using SOAP, HTTP **`get`** and HTTP **`post`**. These protocols are the three options for sending and receiving messages in Web services. The protocol used to transmit request and response messages is sometimes known as the Web service's *wire protocol* or *wire format*, because the protocol specifies how information is transmitted "along the wire." Notice that Fig. 21.3 uses the HTTP **`get`** protocol to test a method.


http://localhost/HugeInteger/HugeInteger.asmx?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/HugeInteger/HugeInteger.asmx?WSDL
Go Links
Done
Local intranet

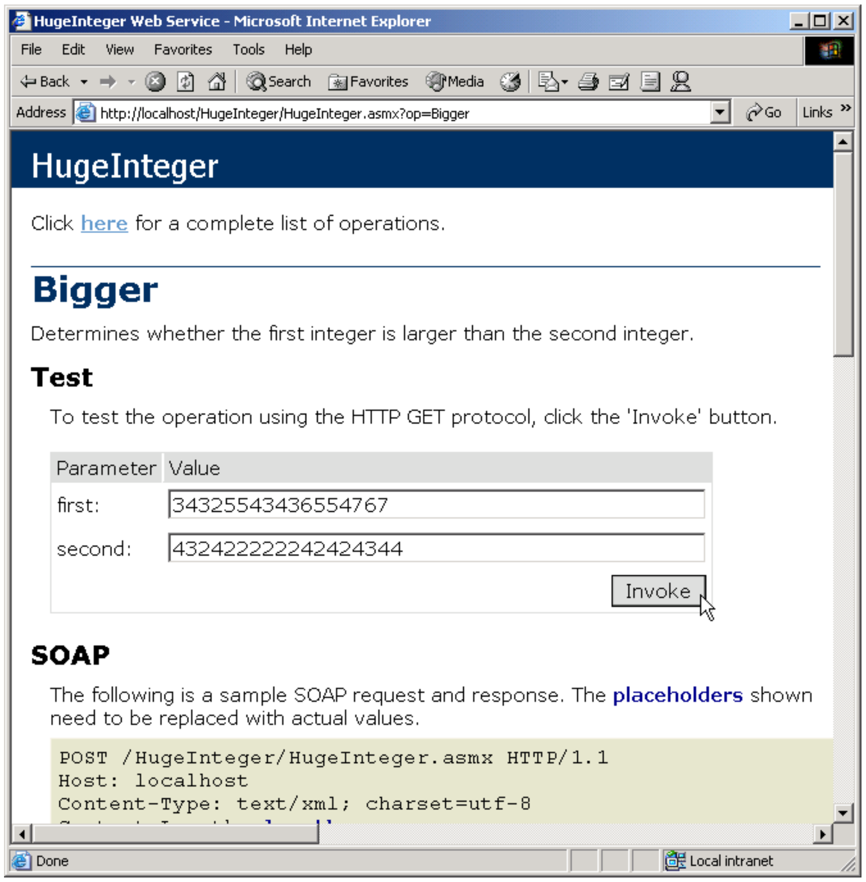
```
  <?xml version="1.0" encoding="utf-8" ?>
- <definitions
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://www.deitel.com/csphtp1/ch21/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://www.deitel.com/csphtp1/ch21/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://www.deitel.com/csphtp1/ch21/">
      - <s:element name="Add">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1"
                name="first" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
                name="second" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="AddResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1"
```

Fig. 21.2 Service description for a Web service.

In Fig. 21.3, users can test a method by entering **Value**s in the **first:** and **second:** fields and then clicking **Invoke** (in this example, we tested method **Bigger**). The method executes, and a new Web browser window opens to display an XML document that contains the result (Fig. 21.4).

## Testing and Debugging Tip 21.1

*Using the ASMX page of a Web service to test and debug methods makes that Web service more reliable and robust; it also reduces the likelihood that clients using the Web service will encounter errors.*


HugeInteger Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/HugeInteger/HugeInteger.asmx?op=Bigger
Go Links
Done
Local intranet

HugeInteger

Click here for a complete list of operations.

Bigger

Determines whether the first integer is larger than the second integer.

Test

To test the operation using the HTTP GET protocol, click the 'Invoke' button.

| Parameter | Value |
|---|---|
| first: | 34325543436554767 |
| second: | 432422222242424344 |

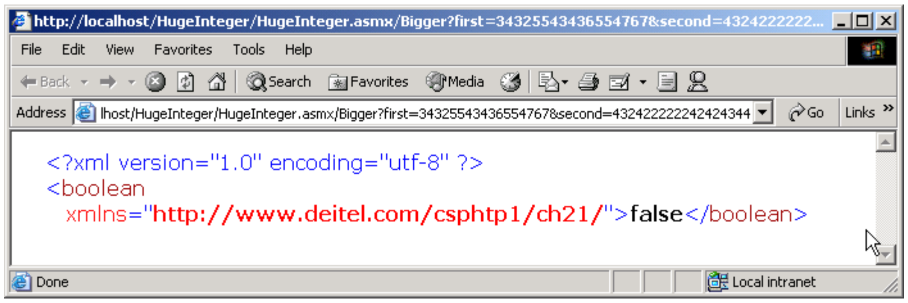
Invoke

SOAP

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /HugeInteger/HugeInteger.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
```

Fig. 21.3 Invoking a method of a Web service from a Web browser.


http://localhost/HugeInteger/HugeInteger.asmx/Bigger?first=343255434365547678second=4324222222...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address lhost/HugeInteger/HugeInteger.asmx/Bigger?first=343255434365547678second=432422222242424344
Go Links
Done
Local intranet

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean
  xmlns="http://www.deitel.com/csphtp1/ch21/">false</boolean>
```

Fig. 21.4 Results of invoking a Web-service method from a Web browser.