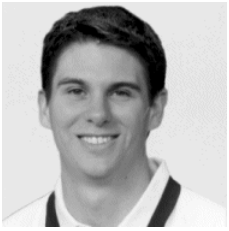
ASP+ Web Services

Rob Howard, Microsoft

Introduction

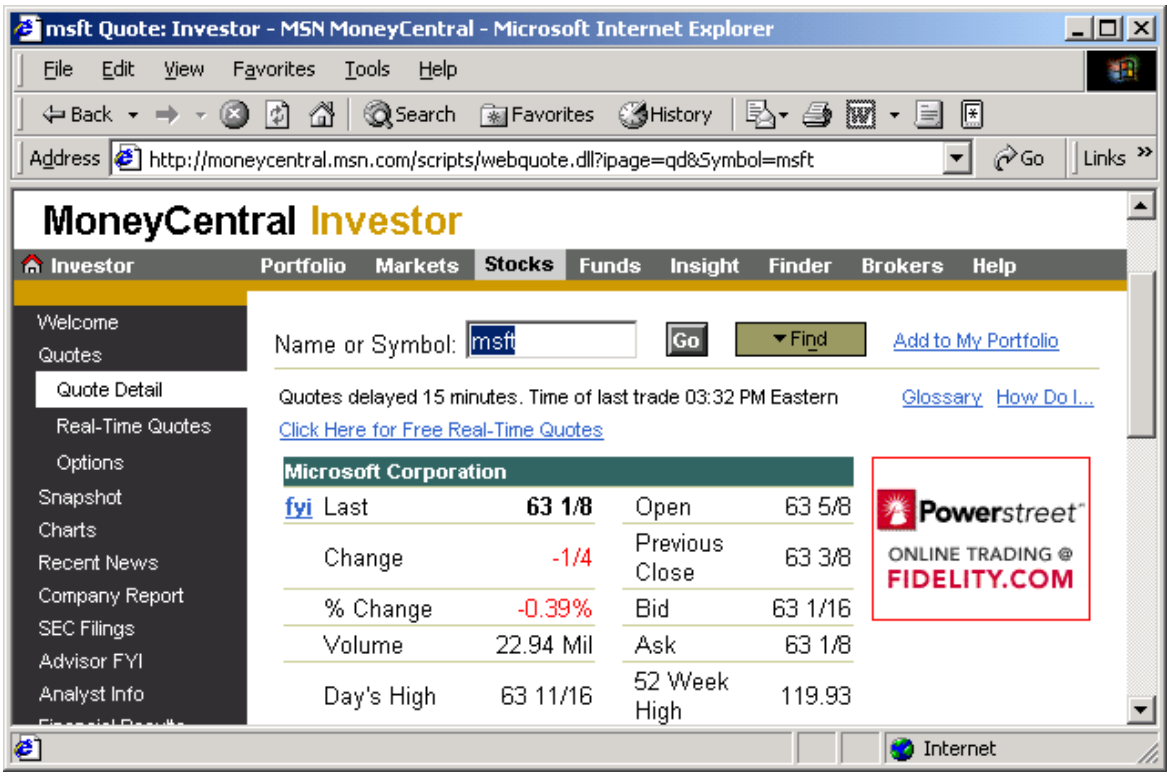
A web service is application logic that is programmatically available, and can be exposed using the Internet. Just as applications have targeted the rich services of the platform in the past, web applications of the future will take advantage of web services for programmability and feature enhancement.



Robert Howard
Robert is a technical evangelist for Microsoft's Developer relations Group. He advises organizations on how best to implement Microsoft products, and he writes and reviews code and provides technical support to those organizations.

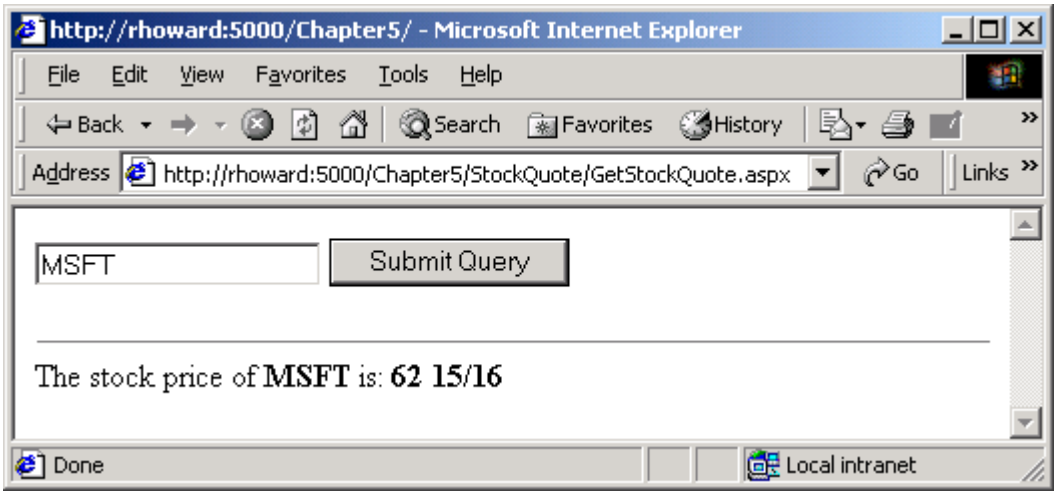
Web service overview

A web service is application logic that is programmatically available, exposed using the Internet. For example, we use application logic on the Internet on a daily basis. We interact with services through a browser for requests such as stock quote data:



In the above screen shot from Microsoft's MoneyCentral website, application logic is used to perform a look-up for a stock symbol (MSFT) and return the related information formatted in HTML to the browser. The application logic is tied to the browser's UI and the end user is limited to the usage of this information in the context in which it is presented. Technologies such as personalization and membership have been made available on some sites, but at most this is simply a filter to determine what data the server believes is relevant for consumption by the end user.

We can of course screen scrape this data from MoneyCentral and use it for our own application:



As seen in the above screenshot, I've written code to screen scrape MoneyCentral.com. In some ways this allows us to use the business logic that MoneyCentral.com provides, but it's somewhat of a hack since we're parsing the entire HTML document to simply find the price of our quote.

One of the driving forces behind Microsoft's web services efforts is further separation of application logic from presentation, and to enable the web such that developers can build new and interesting solutions without being tied strictly to the browser.

For example, rather than screen scraping, we would much rather have MoneyCentral's application logic programmatically available. What we really want is a **StockQuote** component with a `string GetQuote(string strSymbol)` method. Rather than parsing an HTML document for one piece of data, we simply use MoneyCentral's `GetQuote()` method remotely. This is what web services enables.

Building Web Services with .NET technologies

The goal for Web Services in ASP+ is to provide a programming abstraction that allows developers to easily expose programmatic functionality to the web, without the developers requiring knowledge of HTTP, COM, or data marshalling. Rather, developers author web services by simply creating classes and exposing methods and properties as web callable. Web callable in the sense of making the application logic available as a web service.

Web services are exposed in ASP+ through an `.asmx` file. Defined, an `.asmx` file is simply a declarative text file that contains:

- **Processing Directives** – Processing directives instruct the complier how the compile the given resource.
- **Class Attributes** – Used to 'decorate' classes and specify additional behaviors. These behaviors are applied or ignored based on the processing directives used.
- **Application Code** – A .NET class defined either declaratively in the .asmx file, or in pre-complied assembly.

A Web Service `.asmx` file is simply a .NET class with attributes that mark methods of the class as web callable.

- C#: `[WebMethod]`
- VB7: `<WebMethod()>`
- JScript: `WebMethodAttribute`

When web service files are compiled (either on the first request, or with a command line utility - available post beta 1) and `WebMethod` attributes on methods or properties are encountered, Web Services are enabled only for those methods and properties. Those methods and properties without the `WebMethod` attribute are not enabled for web services. We can think of `WebMethod` as being a class accessor that marks the method or property as 'web callable' or ‘web public’.

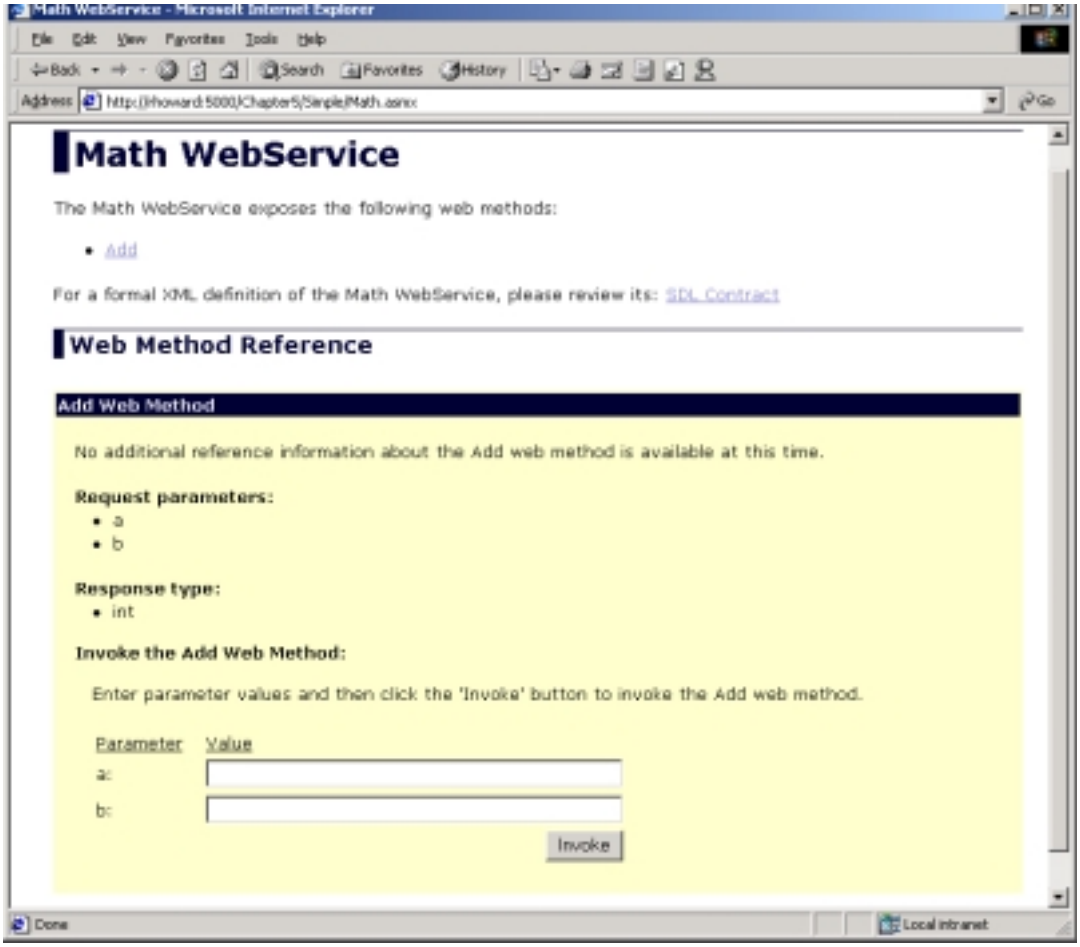
The purpose of defining methods as web callable rather than simply allowing any public method to be accessible is to prevent over exposure. For example, every .NET object derives from the class `Object`. Thus every class supports a base set of public methods (such as `ToString()`), however, these methods don't need to be made available as web callable.

ASP+ supports three types of protocols we can use for Web Services all of which return XML in the body of an HTTP message:

- **HTTP-GET** – Support for the HTTP Get verb for encoding and passing of parameters, along with request semantics.
- **HTTP-POST** – Support for the HTTP Post verb for encoding and passing of parameters, along with request semantics.
- **SOAP** – Simple Object Access Protocol (SOAP), W3C submitted document that describes a platform neutral contract for serialization, and transport, of data using XML and HTTP.

Description

When we access a web service file on our web server through a browser it introspects (the programmatic term is reflection) on the compiled class, and returns an HTML view of the web service:



However, programmable clients that use services need a contract that defines what the service is providing such that clients can know the available methods, properties, and prototypes and call them correctly. A good example here is just as we needed Interface Definition Language (IDL) to describe the interfaces, methods, and properties in the COM world, we need a similar description language for web services.

Having a description language also provides us with the other benefits:

- **Early binding** - Early bind to services since we know their data types before we use them.
- **Development Features** - Take advantage of intelli-sense code completion in our Visual Studio development environment.

With the public announcement of .NET, Microsoft has also announced SOAP Contract Language (SCL). SCL is the public "contract" for our web service; it is an XML document that describes:

- **Service Transports** – The protocols, e.g. HTTP GET, used to access a web service and their respective end-points.

- **Invocation Semantics** – Defines how requests are made to the service, e.g. agreed XML serialization format, and how the service responds, e.g. agreed XML serialization format such as SOAP.
- **Call Order** – Defines the semantics of how a client and server communicate for a request.

The SCL document is created automatically by ASP+, and is accessible as:

http://[server:port]/[application path]/[file].asmx?sdl

> **Note: The HTTP GET parameter passed is SDL not SCL. SDL (Service Description Language) has been replaced by SOAP Contract Language and future versions of the .NET SDK will reflect this change.**

Here is an example of some SCL (automatically generated by ASP+):

```
...

<httpget xmlns="urn:schemas-xmlsoap-org:get-sdl-2000-01-25">
  <service>
    <requestResponse name="Add"
                  href="http://rhoward:5000/Chapter5/Simple/Math.asmx/Add">
      <request>
        <param name="a" />
        <param name="b" />
      </request>
      <response>
        <xmlMime ref="s0:int" />
      </response>
    </requestResponse>
  </service>
</httpget>

...
```

Discovery

In addition to the need to describe a web service, the need also exists for discovery of web services. Microsoft has release a public specification (DISCO) that addresses discovery for a variety of URI based resources, one of these being web services.

The DISCO specification describes the process whereby a client makes a request for a URI (e.g., www.ibuyspy.com) and is returned a default web document (such as default.htm) or a DISCO file. If a default web document is returned, this document should have a link to the DISCO file for the site.

The DISCO file itself contains links to the SCL documents. These links can point to either the server on which the DISCO document resides or to another server where the SCL file is accessible.

Building the proxy

The SCL provides us with a description of the web service. Clients that wish to use the web service can look to the SCL for the semantics of how a service may be used. These clients may choose to build a proxy class that clients can interact with to call the methods and properties of the web service.

A proxy class (or stub) is code that looks exactly like the class it is meant to represent, but does not contain any of the application logic. Instead a proxy class contains marshalling (how we order the data into a manner that allows it to be sent) and transport logic. Clients use the proxy, and treat it as if it is the class it is meant to represent.

Tools are provided with the .NET SDK that parse the SCL file and generate a strongly typed .NET class.

Further Resources

- Wrox Press: "A Preview of ASP+"
  - Chapter 5 covers ASP+ Web Services
- Web sites
  - http://msdn.microsoft.com
  - http://www.asptoday.com
  - http://www.develop.com/dm/default.asp
  - http://www.aspng.com/aspng/index.aspx
  - http://www.wrox.com/beta (for more on "Professional ASP+" by Wrox)