Web Services

Creating

Web Service

- Consists only of methods in a class, some of which are callable by outside clients
- Visual Studio presents you with both an asmx page and asmx.cs page
- asmx page is target for browser if you want to check behavior

```
using System;
using System.Web.Services;

namespace MathWebService
{
            public class MathService : System.Web,Services.WebService
            {
                        public MathService() {}

                        [WebMethod]
                        public int Multiply (int a, int b)
                        {
                                    return a * b;
                        }
            }
}
```

```
<%@ WebService Language="C#"
CodeBehind="~/App_Code/MathService.cs" Class="MathService" %>
```

Class descends from WebService class – Advantages

Gain access to several ASP.NET objects including
* Application and Session objects
* User object
* Context object

Web Service

- May have private helper methods
- For each method you want to expose
  - declare as public
  - put [WebMethod] attribute before declaration
- [WebMethod] other properties

[WebMethod(property=value,property=value)]

WebMethod properties

- BufferedResponse=true|false
- CacheDuration=# of seconds
- Description="What this method does"
- EnableSession=true|false
- MessageName="an alias"
- TransActionOption=

WebService Attribute

[WebService(property=value,property=value)]

- Description="A description of the class"
- Name="Some Name"
- Namespace="make up a uri"

Data Types

- Can use any CLR supported primitive data type as either a parameter or return type
- Includes all primitives, Object, arrays, List<>, ArrayList, and user defined custom classes

Custom classes

- Class must have a default constructor
- If class uses properties, both get and set must be coded even if only one of them is used

Database Considerations

- Only ADO.NET object returnable by a WebService is a DataSet
- Serialized in a form that is quite fat
- Only use if returning multiple tables with relationships
- If returning results of one table, use an array, ArrayList, or List to be more efficient

Discovery

- Developers of consuming apps need to find out what is available – the discovery process
- Description of web service specified in a wsdl file
- Enter url to asmx page and add ?wsdl

http://www.myserver/math/mathservice.asmx?wsdl

Discovery

- View asmx in browser and click service description link
- Or…create a disco file (xml document in same folder as asmx page)

disco /out:<output directory name> http://path to asmx file

- Will create mathservice.disco and mathservice.wsdl

Consuming the WebService

- Consuming app must generate a proxy
- 2 ways to do this
  - Manually – more flexibility and features
  - Let VS do it - easier

Generating proxy manually

- Use an SDK tool named wsdl.exe

wsdl MathService.wsdl …or…
wsdl http://server/folder/MathService.asmx?wsdl

- Output is a source code file containing the proxy class
- Use csc to compile and put dll in bin folder of consuming app

Generating proxy manually

- Now dll is in consuming app's bin folder, just elect to "add a reference" in Visual Studio as you would do for any other dll
- Create it
  MathService proxy = new MathService();
- Use it
  int result = proxy.Multiply(5, 7);

Generating proxy

- Let Visual Studio do it