Programming Skills

C#/.NET

Web Services — Server side of ASP.NET
Remote method invocation over HTTP.
Web Reference, based on SOAP.
Convert temperatures.
Reuse model classes.
VS generates code to create and configure the service as a WebService, connected with a Session and an Application, and describe it using WSDL for a Web Reference (proxy).

Internet Information Server runs production code, receives a request, invokes a method on the service, and sends the result value back to the client.

VS 2005 uses a private web server for testing. WebService tends to play the role of a model.
Proxy Pattern

Client tries to invoke method at service.
Proxy needs to be generated with service interface.
Proxy and request (e.g., web server) agree on line protocol.
Request actually invokes method at service.
Remote transport is transparent for client and service.
Example: Page for Temperature Conversion

Select **File/New/Web Service** in Visual Studio.
Select **Template: ASP.NET Web Service** and **Language: Visual C#**.
using Axel.Conversions;

[WebService(Namespace = "http://www.cs.rit.edu/axel/conversions/")] [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

    IReversibleFunction c2f;
    IFunction f2c;

    public Service () {
        c2f = new ReversibleLinearFunction(9.0 / 5.0, 32.0);
        f2c = c2f.inverse();
    }

    [WebMethod]
    public double Celsius (double fahrenheit) {
        return f2c.Y(fahrenheit);
    }

    [WebMethod]
    public double Fahrenheit (double celsius) {
        return c2f.Y(celsius);
    }
}
Select **Website/Add Reference** and add *libraries* with the **ReversibleLinearFunction**.

Executables have to be renamed — the compiler appears not to reference `.exe` files.

**Build/Build Website** and **Debug/Start Debugging**.

Visual Studio provides a private webserver for testing which refuses non-local connections.
VS generates a description and a test page for each method. Each test page sketches how the service could be tested using, e.g., `telnet`. A reply is represented in XML.
The service can be used as a reference anywhere, even on a JDK platform.

E.g., use Visual Studio to create a Windows Application.

Design a form for temperature conversion, create KeyPress event handlers.
Web Reference

Run the service.

In the client's **Solution Explorer**, right-click on References to Add Service Reference.

Click on **Discover** to check the solution.
Web Reference: Proxy

Enter the URL:

http://localhost:port/site/Service.asmx

where \texttt{port} points to the private web server and \texttt{site} is the web service project.

Change the \textbf{Namespace} to \texttt{temperatures} and press \textbf{Ok}.

This uses the WSDL description obtained at the URL to create a proxy class in the namespace \texttt{temperatures}.

\texttt{app.config} contains the port number.
protected temperatures.ServiceSoapClient model =
    new temperatures.ServiceSoapClient(); // proxy

protected void toFahrenheit (object sender,
    KeyPressEventArgs e) {
    switch (e.KeyChar) {
    case '\n':
    case '\r':
        fahrenheit.Text = model
            .Fahrenheit(double.Parse(celsius.Text)).ToString();
        break;
    }
}
Asynchronous Client

Using a delegate the client can make asynchronous calls:

```csharp
var model = new temperatures.ServiceSoapClient();
delegate double Y (double x);
var call = new Y(model.Fahrenheit);

var r = call.BeginInvoke(32.0, new AsyncCallback(done),
    new object[]{ call, fahrenheit });

// ...
void done (IAsyncResult r) {
    var call = (Y)((object[])r.AsyncState)[0];
    var fahrenheit = (Control)((object[])r.AsyncState)[1];
    double celsius = call.EndInvoke(r);
    Invoke((Action)delegate {
        fahrenheit.Text = celsius.ToString();
    });
}
```

Asynchronous Service

The service can arrange to process a synchronous call asynchronously:

```csharp
IFunction f2c;
delegate double Y (double x);
[WebMethod]
public IAsyncResult BeginCelsius (double fahrenheit,
    AsyncCallback callback, object data) {
    Y call = new Y(f2c.Y);
    return call.BeginInvoke(fahrenheit, callback, call);
}
[WebMethod]
public double EndCelsius (IAsyncResult handle) {
    return ((Y)handle.AsyncState).EndInvoke(handle);
}
```

Threading

WebHost is constructed for each request. Requests are serviced concurrently. Application returns HttpApplicationState which is thread-safe. Application.Lock() Application.UnLock() provide a transaction facility, e.g., to construct unique global objects.
HTTP Session

Session returns HttpSessionState which is not thread-safe (but connected to one client).

Session support must be enabled:

```csharp
[WebMethod(EnableSession=true)]
method
```

Client must support cookies:

```csharp
Service s = new Service();
s.CookieContainer = new System.Net.CookieContainer();
s.method(...); s.method(...); ...
```

New Session for each new proxy on client. See service model for ServiceSoapClient.

Summary

An ASP.NET web service provides an event loop for serving WSDL requests and remote method invocations.

VS silently creates WSDL descriptions, viewed with http://Service.asmx?WSDL

Application provides communication between all methods and all users of an application.

For interoperation with the JDK, see http://www.cs.rit.edu/~ats/talks/ws-rjugg/