COIN-078B
Internet Programming with XML

Consuming Web Services

Module Objectives

- Consuming Web Services in Internet Explorer
- Consuming Web Services in Java
- Consuming Web Services in .NET

Reading Assignments:

- **Textbook:** chapters 19 and 21
- **Teach Yourself Web Services in 24 Hours:** Part III

Consuming Web Services in Internet Explorer using WebService Behavior

The WebService behavior enables IE client-side script to invoke remote methods exposed by Web Services, or other Web servers, that support the SOAP and Web Services Description Language (WSDL) 1.1. The **WebService behavior** is implemented with an HTML Component (HTC) file as an attached behavior, so it can be used in Microsoft Internet Explorer 5 and later versions. Once the **WebService behavior** is attached to a Web page, Internet Explorer can invoke methods from Web Services and use the results directly in client-side script.

The **WebService behavior** is a dynamic SOAP-based web services proxy that implements XMLHttp object in AJAX model. It contains:

- **WSDL parser** - loads and parses the WSDL and binds the operations, parameters and datatypes into a webservice interface definition object model in JavaScript. The `onserviceavailable` event is triggered upon completion of this step.
- **Serializer** - encode a javascript request object and marshal into an xml structure and wrap in SOAP envelop.
- **XMLHttp object** - Sends the SOAP request message to the provider's endpoint and receives the SOAP response message asynchronously by default (can be set to synchronous). Any callback handler function is invoked when an event (`onresult`) that loads and deserializes the response message is triggered.
- **Deserializer** - unmarshal the parsed SOAP response message `result.raw` and decode into a javascript response object `result.value`.

To attach a **WebService behavior** to a page, first download the WebService HTC File file and put a copy in the same directory as the pages that use the behavior. In your html page, add a `div` tag with an `id` and `style="behavior:url(webservice.htc)"`. You can add the `onresult` event handler in the tag or declare it in the script.

The following sample shows how the **WebService behavior** can be attached to a page. The behavior is used to call a simple arithmetic function named `add` from a Web Service. See Using the WebService Behavior (a good place to start learning how to use this WebService proxy in your web page) for a detailed explanation of the following code. Please note that the following example assumes that an ASP.net webservice exists and the wsdl "math.asmx" is found in "services" directory in an MS-IIS server. The "math" webservice has an operation named "add" and requires 2 numeric input values and return a single numeric value. MSDN does not provide you this service. You have to write your own or use an existing webservice.

```
<html>
<head>
<script language="JavaScript">
var iCallID;

function init()
{
  service.useService("/services/math.asmx?WSDL","MyMath");
  iCallID = service.MyMath.callService("add",5,6);
}
```

```
function onWSresult()
{
  if((event.result.error)&&(iCallID==event.result.id))
  {
    var xfaultcode = event.result.errorDetail.code;
    var xfaultstring = event.result.errorDetail.string;
    var xfaultsoap = event.result.errorDetail.raw;

    // Add code to output error information here
  }
  else
  {
    alert("The method returned the result : " + event.result.value);
  }
}
</script>
</head>
<body onload="init()">
<div id="service" style="behavior:url(webservice.htc)" onresult="onWSresult()">
</div>
</body>
</html>
```

Note that this is a simple webservice example and `event.result.value` returns an atomic primitive value. In most webservices the `event.result.value` is an object, you must expand the object to its atomic constituents to see anything.

```
var s = "";
var v = event.result.value;
if(typeof v=="object")
{
  for(var x in v)
  {
    s += x + ' = ' + v[x] + '\n';
  }
}
else s = v;
alert(s);
```

`result.raw` is raw SOAP response message as XML document object. Use its xml property `result.raw.xml` to display it in text.

Another way of invoking callService() using an onresult handler and a call object is illustrated in this example.

Note: the above codes work fine locally but you need to set the content-type to "text/x-component" for Apache webserver to be able to download "webservice.htc" file to the browser. Depending on the Apache configuration, you can add the statement `AddType text/x-component .htc` in the config file or .htaccess file.
krypton.fhda.edu is an Apache web server and does not have MS-IIS installed.

Also note that since the WebService behavior is implementing XMLHttpRequest for client/server communication, it's subject to Cross-Domain Restrictions in Browsers if the application is accessed from a website.

Consuming Web Services in Java with Apache Axis

- Consuming Web Services with Axis
- Using WSDL with Axis

Consuming Web Services in Java with Apache CFX

- Creating Dynamic Client using CXF
- WSDL to Java in CFX

Consuming Web Services in Java with NetBeans

- Web Services (JAX-WS) in Java EE 5 using NetBeans

Resources

- WebService Behavior in Internet Explorer
- WebService HTC File

- Web Services (JAX-WS) in Java EE 5 using NetBeans
- Code Project - Consuming Web Service in Java 5
- Consuming Web Services with Apache Axis
- Using WSDL with Apache Axis
- Creating Dynamic Client using Apache CXF
- Apache CXF User's Guide: WSDL to Java
- Make a SOAP client using Java
- Consuming Web Services with Macromedia JRun 4 and Flash Remoting
- Creating and Consuming Web Services With PHP

Hands-on Assignment:

- **Final Project** due **Week 12**