Web Services with ASP .NET

Asst. Prof. Dr. Kanda Saikaew
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

Agenda

- Introduction to Programming Web Services in Managed Code
- Programming the Web with Web services
- Creating Web Services in Managed Code
- XML Web Services Created Using ASP.NET and XML Web Service Clients

What is ASP .NET?

- **ASP.NET** is a web application framework developed and marketed by Microsoft
- Allow programmers to build dynamic web sites, web applications and web services
- ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language

What is MS .NET Framework?

- It is a software framework that is available with several Microsoft Windows operating systems
- It includes a large library of pre-coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework
- The .NET Framework is a key Microsoft offering and is intended to be used by most new applications created for the Windows platform

ASP .NET Web Services

- Using the ASP.NET page framework, enabling these Web services to access the many features of the .NET Framework
  - Authentication, caching, and state management.
- ASP.NET and the .NET Framework are the basis for Web services in managed code
  - Developers can focus on creating or accessing Web services without needing to write infrastructure code.

ASP .NET Application Model

- Web pages intended for the browser use the .aspx extension
- Web services use the .asmx extension.
- There are two fundamental roles when working with Web services:
  - **Creating a Web service**
  - **Accessing a Web service**

Web Services Scenarios

- Web services can be either stand-alone applications or sub-components of a larger Web application
- For example, suppose you are creating a Web application that sells books online
  - How might your Web application interact with Web services?

Book sellers and Web Services (1/2)

- **Creating Web service**
  - Your application exposes its order processing logic as a Web service
  - Your affiliate Web sites can in turn use in their Web applications to sell books through your online store without requiring their customers to visit your site.

Book sellers and Web Services (2/2)

- **Accessing Web service**
  - Your application accesses Web service provided by another online company that specializes in writing and supplying book reviews for online booksellers
  - When a visitor to your online store views the details of a particular book, they also see the reviews of that book on the same page.

Agenda

- Introduction to Programming Web Services in Managed Code
- Programming the Web with Web services
- Creating Web Services in Managed Code
- XML Web Services Created Using ASP.NET and XML Web Service Clients

Building a Basic XML Web Service using ASP .NET (1/2)

1. Create a file with an .asmx file name extension and declare a Web service in it using an @WebService directive
2. Create a class that implements the Web service. The class can optionally derive from the WebService class

Building a Basic XML Web Service using ASP .NET (2/2)

3. Optionally, apply the WebService attribute to the class implementing the Web service
4. Define the Web service methods that compose the functionality of the Web service

Declaration of Web Services (1/2)

- When you create a Web service in ASP.NET, you place the required @ WebService directive at the top of a text file with an .asmx file name extension
- The presence of the .asmx file and the @ WebService directive correlate the URL address of the Web service with its implementation

Declaration of Web Services (2/2)

- By applying the optional WebService attribute to a class that implements a Web service
  - You can set the default XML namespace for the Web service along with a string to describe the Web service
- It is highly recommended that this default namespace, which originally is http://tempuri.org, be changed before the Web service is made publicly consumable

Declaring a Web Service Example1 (1/2)

- To declare a Web service whose implementation resides in the same file
  - Add an @WebService directive to the top of a file with an .asmx file name extension
  - Specify the class that implements the Web service
  - Specify the programming language that is used in the implementation

Declaring a Web Service Example1 (2/2)

- C#

```
<%@ WebService Language=“C#”
  Class=“Util” %>
```

- Visual Basic

```
<%@ WebService Language="VB"
  Class="Util" %>
```

Declaring a Web Service Example2 (1/2)

- To declare a Web service whose implementation resides in an assembly
  - Add an @ WebService directive to the top of a file with an .asmx extension
  - Specify the class that implements the Web service
  - Specify the assembly that contains the implementation
  - Specify the programming language that is used in the implementation

Declaring a Web Service Example2 (2/2)

- The following **@ WebService** directive is the only line in a file with an .asmx extension
  - Specifying that the MyName.MyWebService class resides in the MyAssembly assembly within the \Bin directory
  - <%@ WebService Language="C#" Class="MyName.MyWebService,MyAssembly" %>

Building a Basic XML Web Service using ASP .NET (1/2)

1. Create a file with an .asmx file name extension and declare a Web service in it using an @WebService directive
2. Create a class that implements the Web service. The class can optionally derive from the WebService class

Deriving from the WebService Class

- Classes that implement a Web service can optionally derive from the WebService class
  - To gain access to the common ASP.NET objects, such as
    - WebService.Application
    - WebService.Session
    - WebService.User
    - WebService.Context

Sample Derivation from WebService Class

```
<%@ WebService Language="C#"
   Class="Util" %>
using System;
using System.Web.Services;
public class Util: WebService
```

Building a Basic XML Web Service using ASP .NET (2/2)

3. Optionally, apply the WebService attribute to the class implementing the Web service
4. Define the Web service methods that compose the functionality of the Web service

Applying the WebService Attribute

- Apply the optional WebService attribute to a class that implements a Web service to set
  - The XML namespace for the Web service
  - The description of Web service
- The default namespace originally is http://tempuri.org

Applying WebService Attribute

```
<%@ WebService Language="C#" Class=
   "ServerVariables"%>
using System;
using System.Web.Services;
[ WebService(
Description="Calculator Service",
Namespace="http://campus.en.kku.ac.th/")]
public class Calculator: WebService {
```

Building a Basic XML Web Service using ASP .NET (2/2)

3. Optionally, apply the WebService attribute to the class implementing the Web service
4. Define the Web service methods that compose the functionality of the Web service

Definition of Web Service Methods

- Apply a WebMethod attribute to public methods in the class that implements a Web service
  - Have the ability to receive Web service requests and send back responses
- The method and class must be public and running inside an ASP.NET Web application

Declaring a Web Service Method

- Add public methods to the class that is implementing the Web service.
- Apply the **WebMethod** attribute to the public methods that you want to be mapped to Web service operations

Declaring a Web Service Method Example

```
<%@ WebService Language="C#" Class="Util" %>
using System.Web.Services;
using System;
[WebService
(Namespace="http://campus.en.kku.ac.th/")]
public class Util: WebService {
[ WebMethod]
 public long Multiply(int a, int b) {
    return a * b;
} }
```

Asynchronous XML Web Service Methods

- To improve performance of Web service methods that invoke long-running methods that block their thread
  - You should consider exposing them as asynchronous Web service methods
- Implementing an asynchronous Web service method allows that thread to execute other code when it is returned to the thread pool
  - Allows one more of the limited number of threads to execute, enhancing the overall performance and scalability of the system

Implementing Asynchronous Web Service Method (1/3)

- 1) Split a synchronous Web service method into two methods
  - Each with the same base name, one with that name starting with Begin and the other End
  - // Define the Begin method. [WebMethod] public IAsyncResult BeginGetAuthorRoyalties(String Author, AsyncCallback callback, object asyncState) {…}
  - // Define the End method. [WebMethod] public AuthorRoyalties EndGetAuthorRoyalties(IAsyncResult asyncResult) {…} }

Implementing Asynchronous Web Service Method (2/3)

- 2) The parameter list for the Begin method contains all the in and by reference parameters for the method's functionality plus two additional parameters
  - *By reference* parameters are listed as *in* parameters
  - The second from the last parameter must be an **AsyncCallback**. The **AsyncCallback** parameter allows a client to supply a delegate, which is invoked when the method completes
  - The last parameter is an **Object**. The **Object** parameter allows a caller to supply state information to the method
  - The return value must be of type **IAsyncResult**.

```
// Define the Begin method.
[WebMethod]
public IAsyncResult BeginGetAuthorRoyalties(String
    Author, AsyncCallback callback, object asyncState) {
    ...}
```

Implementing Asynchronous Web Service Method (3/3)

- 3) The parameter list for the *End* method consists of an **IAsyncResult** followed by any *out* and *by reference* parameters specific to the method's functionality.
  - The return value is the same type as the return value of a synchronous Web service method
  - *By reference* parameters are listed as *out* parameters

```
// Define the End method.
[WebMethod] public AuthorRoyalties
    EndGetAuthorRoyalties(IAsyncResult asyncResult) {
    ...}
```

Accessing Web Services

1. Locate the Web service you want to access
2. Create a proxy class for the Web service by adding a Web reference to your project
3. Reference the proxy class in the client code by including its namespace
4. Create an instance of the Web service proxy class in the client code
5. Access the Web service using the methods of the proxy

Accessing Web Services Example

```
Using System;
namespace Applicaion1 {
    class Class1 {
        static void Main() {
            Converter.Service1 cService = new Converter.Service1();
            Console.WriteLine("Temperature I degrees Fahrenheit:");
            double dFahrenheit = Convert.
```