

# **Module 7: Building Web Applications**

# Overview

- Introduction to ASP.NET
- Creating Web Form Applications
- Building Web Services
- Using Web Services

# ◆ Introduction to ASP.NET

- Overview of ASP.NET
- Using Response and Request Objects
- Maintaining Client-Side State
- Maintaining Server-Side State
- Managing an ASP.NET Application
- Overview of ASP.NET Security
- Using Global Events with Global.asax

# Overview of ASP.NET

- **Code behind development**
  - Intuitive approach to development similar to Windows Forms
- **Code can be compiled using any .NET-compatible language**
  - Significant performance improvement
- **ASP.NET pages run side-by-side with ASP pages**
  - Files with the .aspx extension run side-by-side with current ASP applications on IIS
- **Automatic support for multiple browsers**
  - Rich DHTML, HTML 3.2, and small devices
- **ASP.NET namespaces provide rich functionality**
- **Built-in support for Web Services**

# Using Response and Request Objects

- The System.Web namespace provides common Web functionality
- Requesting information from the client
  - Request object

```
Dim strValue As String = Request.Form("txtInput")
```

- Sending information to the client
  - Response object

```
Response.Write("<H2>The date is: " & Now.Date & "</H2>")
```

# Maintaining Client-Side State

- Maintaining control state across multiple requests
  - Set **EnableViewState** property of control
  - Use the **StateBag** class to store extra data

```
ViewState("TempData") = 25
```

- Send data to client as hidden string

```
<input type="hidden" name="__VIEWSTATE" value="d0...NnU=" />
```

- Using cookies for enhanced client-side state

```
Response.Cookies("User_FullName").Value = strFullName  
...  
strFullName = Request.Cookies("User_FullName").Value
```

# Maintaining Server-Side State

- Application object maintains application-level state
  - Data for use by all users

```
Application("App_StartTime") = Now
```

- Session object maintains session-level state
  - Data for use by a single client browser
  - Data recoverable by means of Windows Service or SQL Server database

```
Session("Session_StartTime") = Now
```

# Managing an ASP.NET Application

## ■ Configuring ASP.NET applications

- XML configuration file – Web.config
- Human-readable and writeable
- Stored with the application

## ■ Deploying ASP.NET applications

- XCOPY the pages, components, and configuration
- No registration required

## ■ Updating ASP.NET applications

- Copy new files over old files
- No update utility, restart, or reboot required
- Live update keeps applications running

# Overview of ASP.NET Security

- Security settings stored in web.config file
- Out-of-box authentication support
  - Windows authentication – Basic, Digest, or Integrated
  - Microsoft Passport authentication
  - Forms (Cookie) authentication
- Role-based security architecture

# Using Global Events with Global.asax

- Useful for initializing data for application or session state

```
Sub Application_Start(ByVal Sender As Object, ByVal e As EventArgs)
    Application("SessionCounter") = 0
End Sub

Sub Session_Start(ByVal Sender As Object, ByVal e As EventArgs)
    Application("SessionCounter") = Application("SessionCounter") + 1
    Session("StartTime") = Now
End Sub

Sub Session_End(ByVal Sender As Object, ByVal e As EventArgs)
    Application("SessionCounter") = Application("SessionCounter") - 1
End Sub
```

# ◆ Creating Web Form Applications

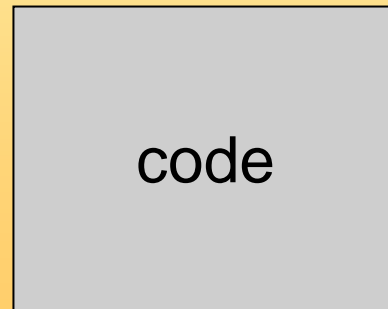
- Structure of Web Forms
- Using HTML Controls
- Advantages of Web Server Controls
- Using Web Server Controls
- Handling Events

# Structure of Web Forms

- Web Forms separate declarative tags from logic
  - The .aspx file contains HTML and other tags
  - The code-behind file contains logic and event handling



Logon.aspx



Logon.aspx.vb

# Using HTML Controls

- Direct relationship to preexisting HTML tags
  - Client side by default
  - Server side with **runat=server** directive
  - Properties correspond one-to-one with HTML, weakly typed
  - Defined with HTML tag

```
<input type=text id="text1" value="some text" runat="server">
```

- ASP.NET includes HTML controls for commonly used HTML elements

# Advantages of Web Server Controls

- **Automatic browser detection**
  - Detect capabilities of client and render accordingly
- **Strongly typed, consistent object model**
  - Enables compile-time type checking
- **Declared with XML tags**
  - Server side only using `runat=server` directive

```
<asp:textbox id="text2" text="some text" runat="server">  
</asp:textbox>
```

- **Rich functionality**
  - Example: **Calendar** or **RadioButtonList** control



# Handling Events

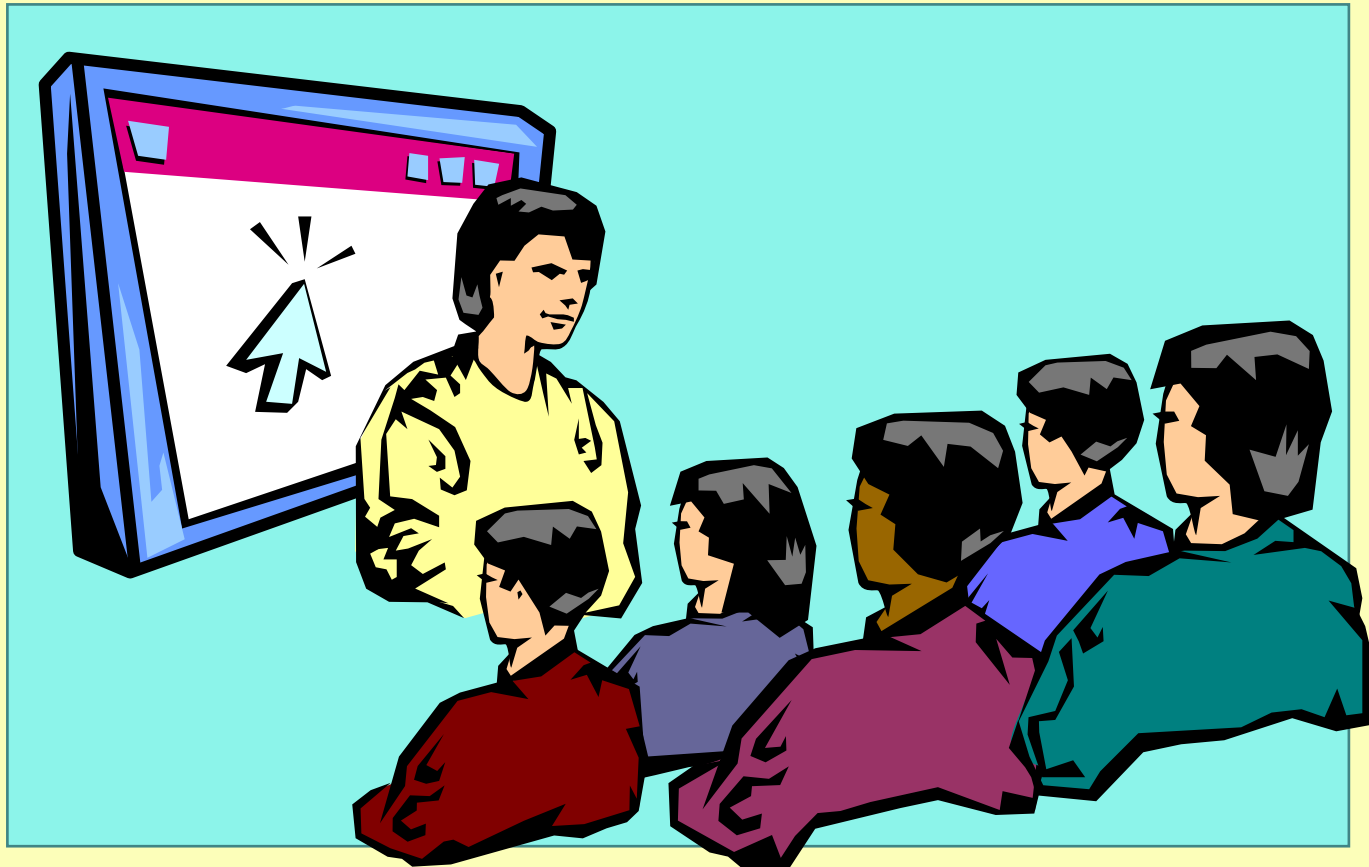
- Event handlers contain code for user interactions
- Page events: Init, Load, PreRender, UnLoad

```
Private Sub Page_Load(ByVal Sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load  
    If Not IsPostBack Then 'IsPostBack is also available via 'Me'  
        'Perform action first time page is displayed  
    End If  
End Sub
```

- Control events: Click, Changed, PreRender

```
Private Sub btn_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btn.Click  
    btn.Text = "clicked"  
End Sub
```

# Demonstration: Creating Web Forms



# Lab 7.1: Creating the Customer Logon Web Forms



# ◆ Building Web Services

- What Are Web Services?
- Creating a Web Service
- Enabling Web Service Discovery
- Deploying and Publishing a Web Service

# What Are Web Services?

- Components accessible by means of the Internet rather than distributed COM
- Built using open Internet protocols
- Consumers send and receive XML messages
- Defined in terms of the message order and format
- Site offers a description of the Web Services it offers
- Search for a site that offers a Web Service with a given capability

XML, XSD, HTTP, SMTP

SOAP

Web Services  
Description Language

Discovery Document

UDDI



# Creating a Web Service

1. Add a Web Service module to the project
  - The .asmx file contains **WebService** directive

```
<%@ WebService Language="vb" Codebehind="User.asmx.vb"  
Class="WebApp.User"%>
```

2. Create a Web Service description
3. Add public subroutines or functions to .asmx.vb file
  - Add **WebMethod** attribute to procedure definitions

```
<WebMethod()> Public Function AddUser(...) As String  
    'Functionality to add a user and return new ID  
    Return strNewId  
End Function
```

# Enabling Web Service Discovery

## ■ Discovery document

- Enables location and interrogation of Web Service descriptions
- Contains links to resources that describe services
- Stores information in XML format
- Created manually or dynamically

```
<?xml version="1.0" ?>  
<discovery xmlns="http://schemas.xmlsoap.org/disco/" ...>  
  <contractRef ref="http://www.nwtraders.msft/Shopping/User.asmx?wsdl"  
    docRef="http://www.nwtraders.msft/Shopping/User.asmx"  
    xmlns="http://schemas.xmlsoap.org/disco/wsdl/" />  
  ...  
</discovery>
```

# Deploying and Publishing a Web Service

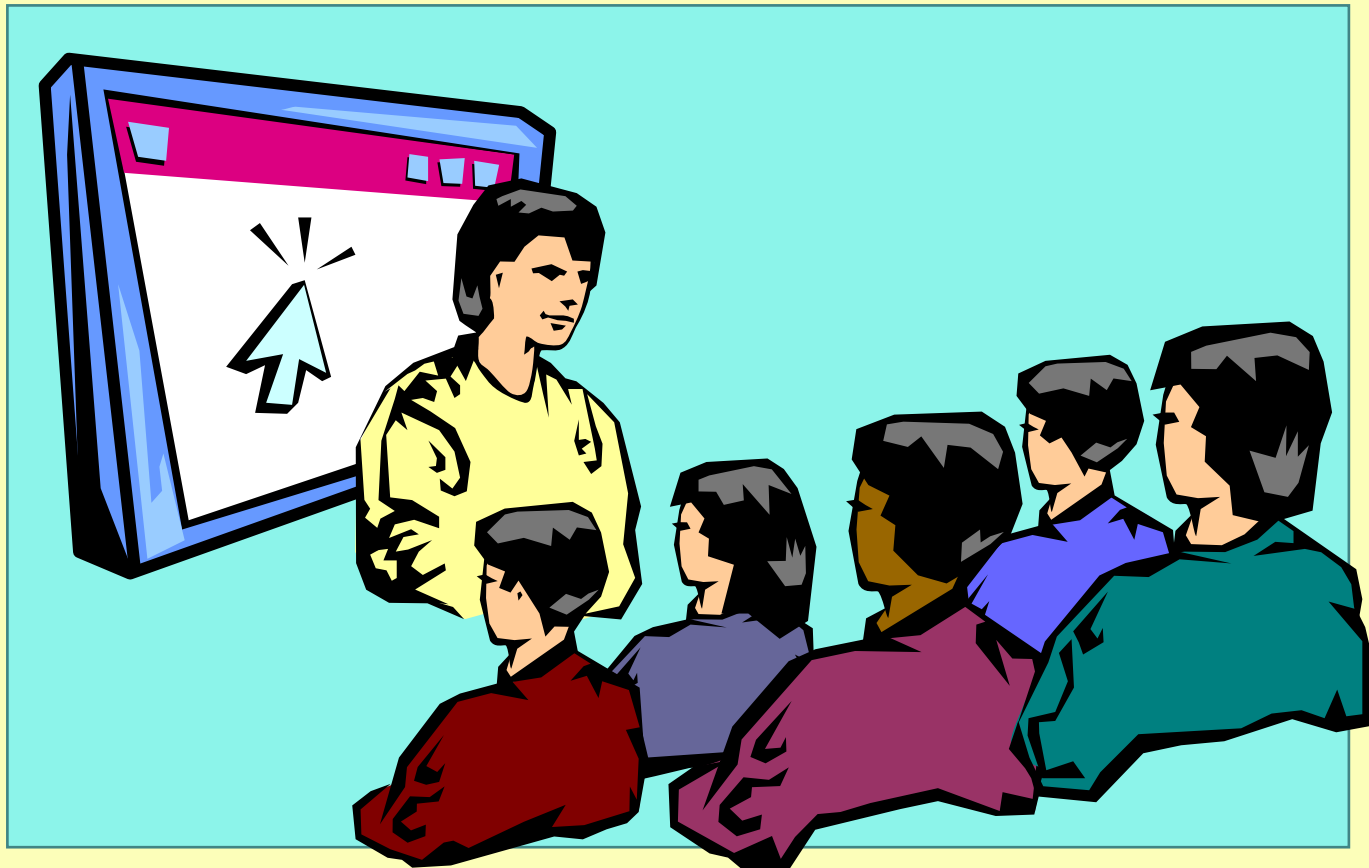
## ■ Deploying a Web Service

- Copy .asmx, Web.config, and any components to an IIS virtual directory

## ■ Publishing a Web Service

- The discovery document is copied to IIS virtual directory
- Dynamic discovery document produced by ASP.NET generates information for all services in all subfolders
- Manually created discovery document returns only explicitly defined information

# Demonstration: Creating a Web Service



# ◆ Using Web Services

- Exploring Web Services
- Invoking a Web Service from a Browser
- Invoking a Web Service from a Client

# Exploring Web Services

- **HTML description page**
  - Describes Web Service methods and arguments
  - Provides simple test utility for methods
  - Displays additional descriptions from attributes
  - Appears when you enter Web Service URL

`http://webservername/virtualdirectory/webservice.asmx`

- **WSDL describes methods, arguments, and responses**
  - Generated by using Web Service URL with ?WSDL switch

# Invoking a Web Service from a Browser

- Enter the URL for the Web Service with parameters

- Syntax:

*http://webservername/vdir/webservicename.asmx/  
MethodName?parameter=value*

- Example:

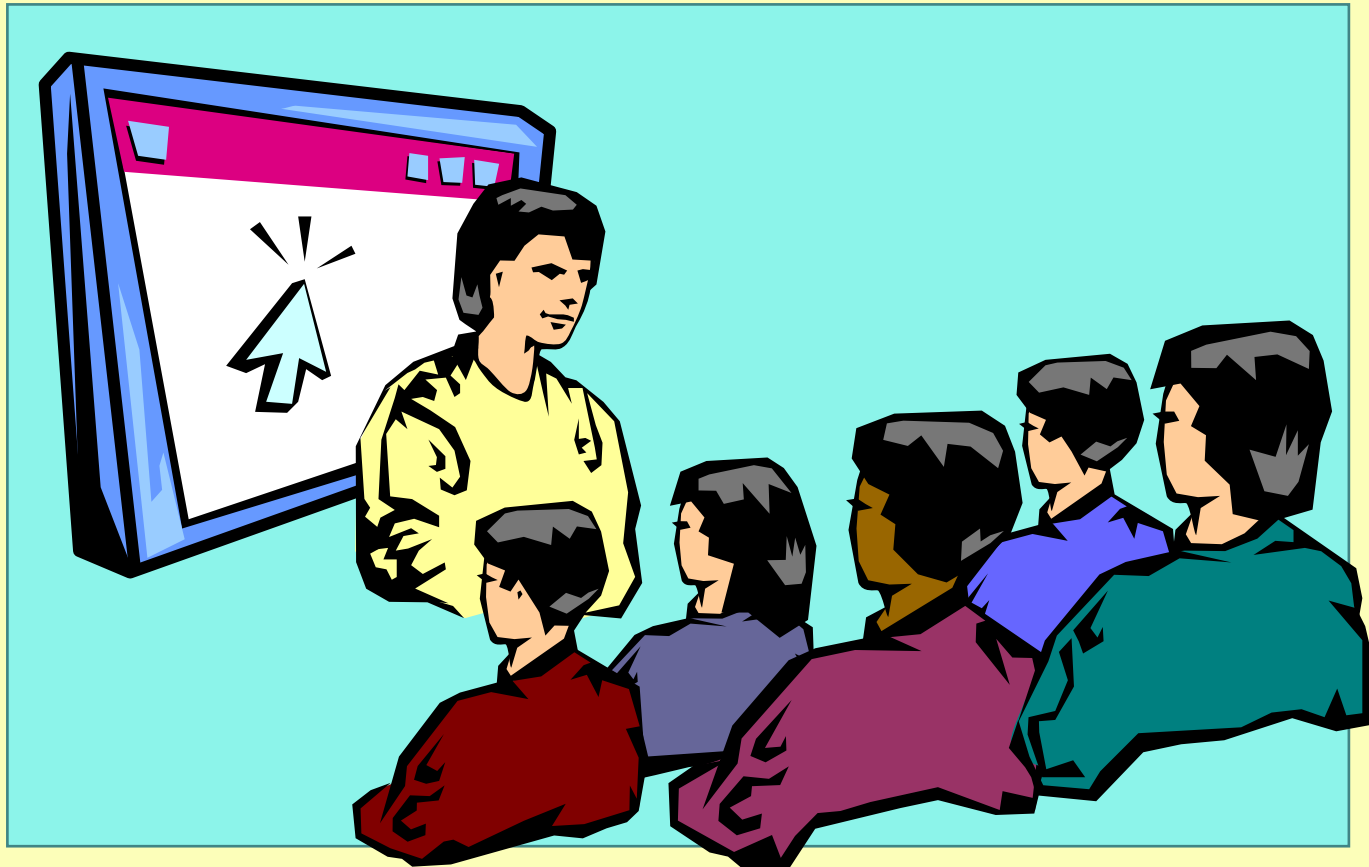
`http://www.nwtraders.msft/Shopping/User.asmx/AddUser?strName=Joe`

# Invoking a Web Service from a Client

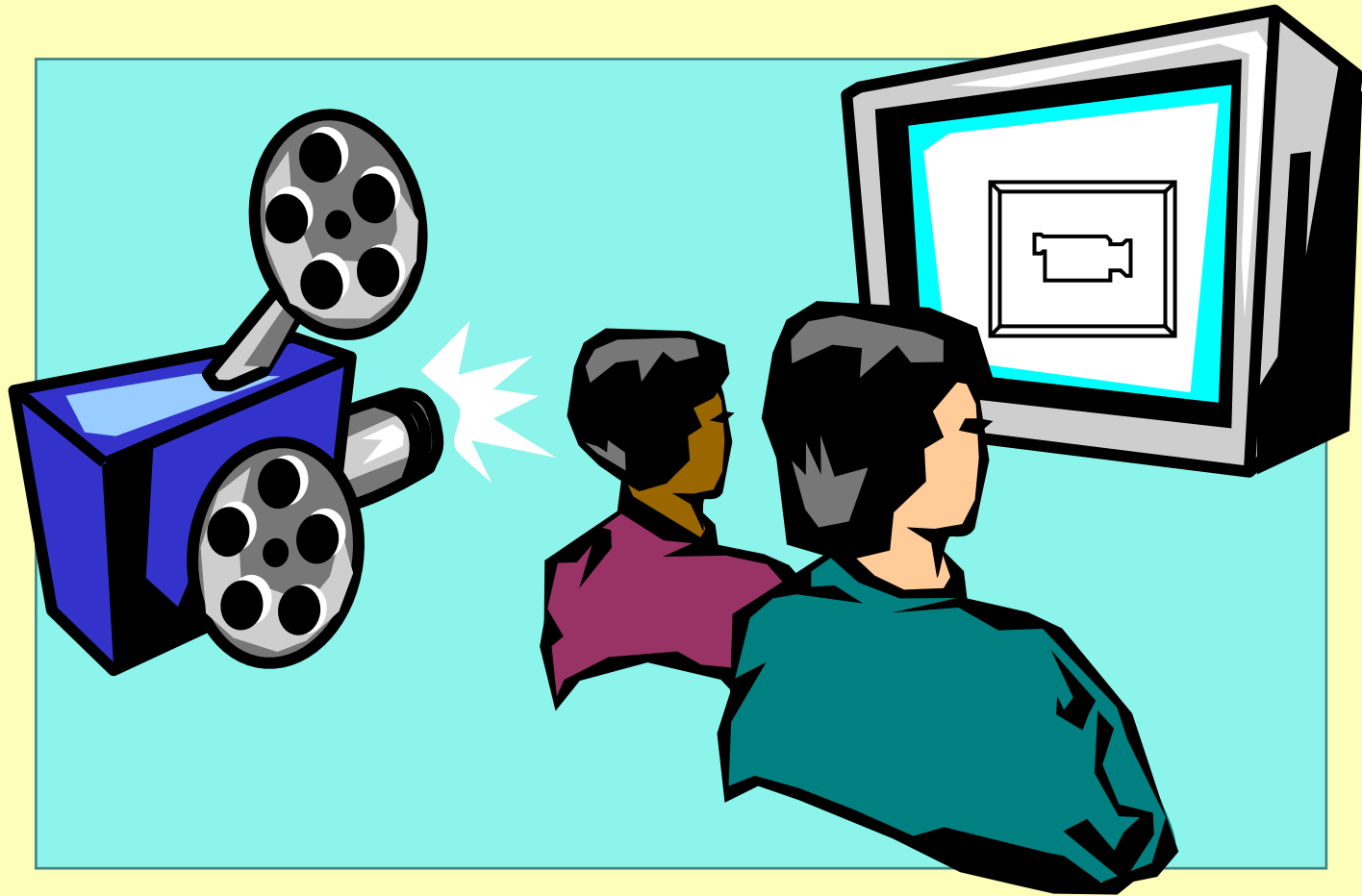
- Visual Basic .NET creates a proxy class for early binding
- Steps required:
  1. Add a Web reference
  2. Enter the URL for the .asmx file
  3. Create client code that uses appropriate namespaces

```
Sub btnSubmit_Click(...) Handles btnSubmit.Click
    Dim usr As New Services.User() 'Services is the given namespace
    MessageBox.Show(usr.AddUser(txtName.Text))
End Sub
```

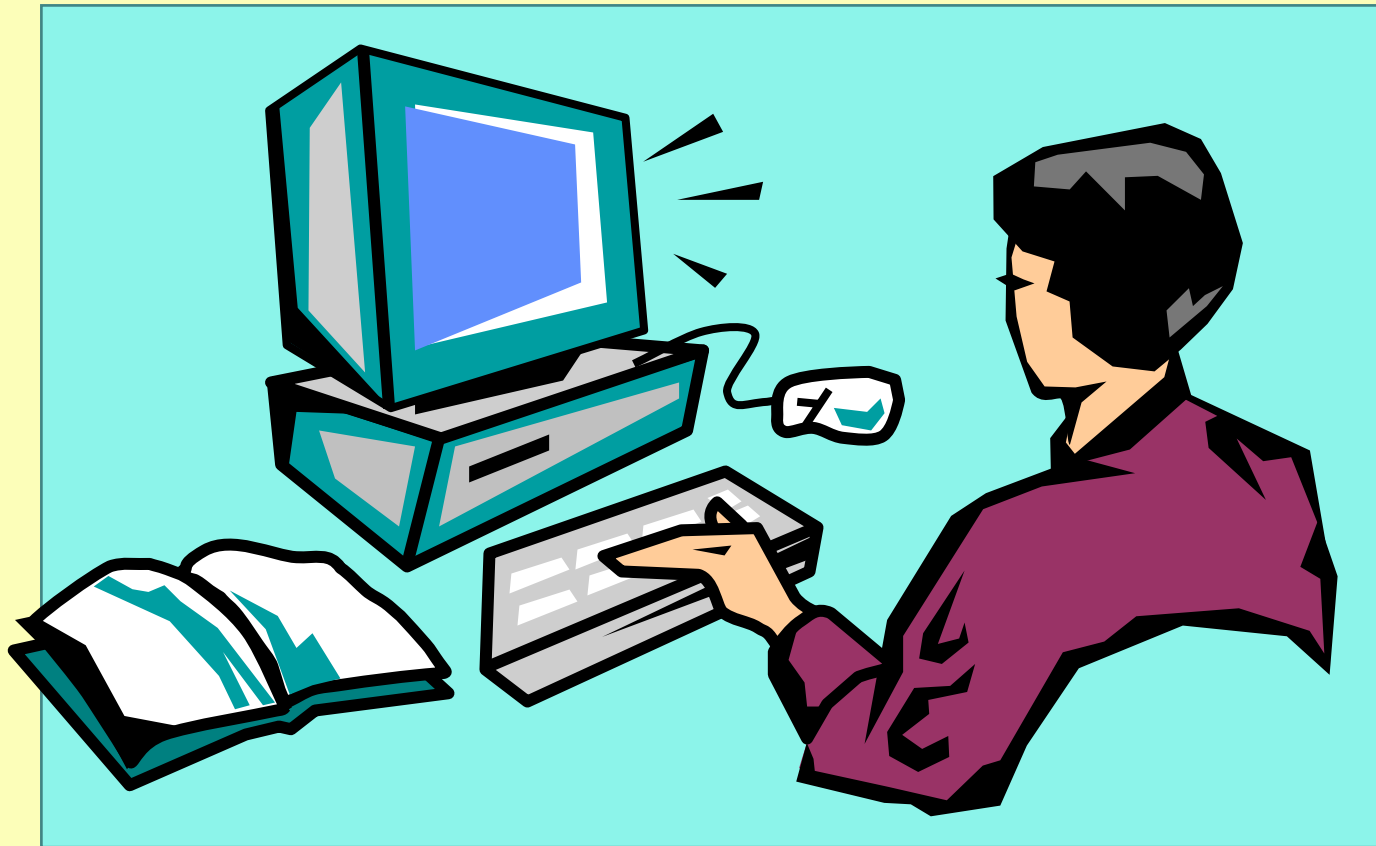
# Demonstration: Using a Web Service



# Multimedia: How Web Services Work



# Lab 7.2: Creating and Using the CustomerService Web Service



# Review

- Introduction to ASP.NET
- Creating Web Form Applications
- Building Web Services
- Using Web Services