

# Preparation Guide for 70-528

## TS: Microsoft .NET Framework 2.0 - Web-Based Client Development

---

This certification exam measures your ability to develop and implement Web-based applications with Web forms, ASP.NET, and the .NET Framework.

### How to use this guide

Below you will find a listing of each of the topics that you should understand in order to successfully complete the exam. Next to each topic is an empty circle, whenever you feel that you understand this topic fill in the circle. This should make it much easier to track your progress and knock out any outstanding topics. You can fill in the circle by using a pencil to shade in a printed form, or simply double-click the circle and choose the filled in bullet and click ok.

If you hover over each of the topics you should see an option to ctrl-click on it. What this allows you to do is to follow each link and learn more about that particular subject. Some of the links may bring you to later sections inside this document, while others will take you to particular sites with more information on the topic. Therefore, you should make sure that you are connected to the internet before continuing.

## Creating and Programming a Web Application

### Create and configure a Web application.

- [Create a new Web application.](#)
- [Add Web Forms pages to a Web application.](#)

### Add and configure Web server controls.

- [Add Web server controls to a Web Form.](#)
- [Configure the properties of Web server controls programmatically.](#)
- [Configure Web server control properties by using the Microsoft Visual Studio Property Editor.](#)
- [Specify whether events of a control cause a Web Form to post to the server.](#)
- [Configure a control to receive postback events.](#)
- [Access controls in Web Forms pages when working with naming containers and child controls.](#)
- [Create HTML server controls in the designer.](#)
- [Set HTML server control properties programmatically.](#)
- [Use HTML server controls to programmatically access HTML tags.](#)
- [Create HTML controls as elements in an HTML document.](#)
- [Use the AdRotator Web server control to manage banners and pop-up windows.](#)
- [Use the Button Web server control to send a command to the server when a button is clicked.](#)
- [Display a calendar on a Web page by using the Calendar Web server control.](#)
- [Implement the CheckBox Web server control.](#)
- [Implement the FileUpload Web server control.](#)
- [Create and manipulate links on a Web Form by using the HyperLink Web server control.](#)

- [Display an image on a Web Form by using the Image Web server control.](#)
- [Implement a button on a Web Form by using the ImageButton Web server control.](#)
- [Define hotspot regions within an image by using the ImageMap Web server control.](#)
- [Use the Label Web server control to display customized text on a Web page.](#)
- [Display a hyperlink style button on a Web Form by using the LinkButton Web server control.](#)
- [Display lists of information by using controls that derive from the ListControl class.](#)
- [Create a Web Form with static text by using the Literal Web server control.](#)
- [Implement pagination for controls on a page by using the Pager Web server control.](#)
- [Use the Panel Web server control to arrange controls in groups on a page.](#)
- [Create a container for a group of View controls by using the MultiView Web server control.](#)
- [Use the View Web server control to create a Web application.](#)
- [Create a mutually exclusive set of choices by using the RadioButton Web server control.](#)
- [Construct a table by using the Table, TableRow, and TableCell Web server controls.](#)
- [Enable users to type information into a Web Form by using the TextBox Web server control.](#)
- [Create a wizard by using the Wizard Web server control to collect data through multiple steps of a process.](#)
- [Use the XML Web server control to create XML data at the location of the control.](#)
- [Customize the appearance of Web server controls by using Web control templates.](#)
- [Programmatically edit settings in a Web site's configuration file.](#)
- [Dynamically add Web server controls to a Web Forms page.](#)

#### **Create event handlers for pages and controls.**

- [Create event handlers for a page or control at design time.](#)
- [Respond to application and session events.](#)

#### **Manage state and application data.**

- [Manage state of an application by using client-based state management options.](#)
- [Manage state of an application by using server-based state management options.](#)
- [Maintain state of an application by using database technology.](#)

#### **Implement globalization and accessibility.**

#### **Implement site navigation and input validation.**

- [Use the SiteMap Web server control to display a representation of a Web site's navigation structure.](#)
- [Use validation controls to perform Web Forms validation.](#)
- [Validate against values in a database for server controls by using a CustomValidator control.](#)
- [Create a CustomValidator control and tie it to a custom function.](#)
- [Test programmatically whether a user's input passed validation before running code.](#)
- [Specify the location of a validation error message for server controls.](#)
- [Format validation error messages for server controls.](#)
- [Specify the layout for in-place messages on server controls.](#)
- [Disable validation for server controls.](#)
- [Display custom error messages for server controls.](#)
- [Validate server controls programmatically.](#)

#### **Write an ASP.NET handler to generate images dynamically for display on a Web page.**

#### **Configure settings for a Web application.**

- [Configure system-wide settings in the Machine.config file.](#)

- [Configure settings for a Web application in the Web.config file.](#)
- [Manage a Web application's configuration by using the Web Site Administration Tool.](#)

#### **Program a Web application.**

- [Redirect users to another Web page by using a server-side method.](#)
- [Detect browser types in Web Forms.](#)
- [Ascertain the cause of an unhandled exception at the page level.](#)
- [Programmatically access the header of a Web page.](#)
- [Implement cross-page postbacks.](#)
- [Assign focus to a control on a page when the page is displayed.](#)
- [Avoid performing unnecessary processing on a round trip by using a page's IsPostBack property.](#)
- [Access encapsulated page and application context.](#)
- [Avoid unnecessary client-side redirection by using the HttpServerUtility.Transfer method.](#)
- [Avoid round trips by using client-side scripts.](#)
- [Use a page's Async attribute to create a page that has built-in asynchronous capabilities.](#)
- [Convert HTML server controls to HTML elements.](#)

## **Integrating Data in a Web Application by Using ADO.NET, XML, and Data-Bound Controls**

#### **Implement data-bound controls.**

- [Use tabular data source controls to return tabular data.](#)
- [Use hierarchical data source controls to display hierarchical data.](#)
- Display data by using simple data-bound controls.
- Display data by using composite data-bound controls.
- Display data by using hierarchical data-bound controls.
- Use the FormView control to display the values of a single table record from a data source.

#### **Manage connections and transactions of databases.**

- [Configure a connection to a database graphically by using the Connection Wizard.](#)
- [Configure a connection by using Server Explorer.](#)
- Configure a connection to a database by using the connection class.
- Connect to a database by using specific database connection objects.
- Enumerate through instances of Microsoft SQL Server by using the DbProviderFactories.GetFactoryClasses method.
- Open a connection by using the Open method of a connection object.
- Close a connection by using the connection object.
- Secure a connection to protect access to your data source.
- [Create a connection designed for reuse in a connection pool.](#)
- Control connection pooling by configuring ConnectionString values based on database type.
- Use connection events to detect database information.
- Handle connection exceptions when connecting to a database.
- [Perform transactions by using the ADO.NET Transaction object.](#)

#### **Create, delete, and edit data in a connected environment.**

- [Retrieve data by using a DataReader object.](#)
- Build SQL commands visually in Server Explorer.
- [Build SQL commands in code.](#)

- [Create parameters for a command object.](#)
- [Perform database operations by using a command object.](#)
- [Retrieve data from a database by using a command object.](#)
- [Perform asynchronous operations by using a command object.](#)
- [Perform bulk copy operations to copy data to a SQL Server computer.](#)
- [Store and retrieve binary large object \(BLOB\) data types in a database.](#)

**Create, delete, and edit data in a disconnected environment.**

- [Create an instance of the DataSet class programmatically.](#)
- [Create a DataSet graphically.](#)
- Create a DataSet programmatically.
- [Add a DataTable to a DataSet.](#)
- [Add a relationship between tables.](#)
- [Navigate a relationship between tables.](#)
- [Merge DataSet contents.](#)
- [Copy DataSet contents.](#)
- [Create a strongly typed DataSet.](#)
- [Create DataTables.](#)
- [Manage data within a DataTable.](#)
- [Create and use DataViews.](#)
- [Represent data in a DataSet by using XML.](#)
- [Access an ADO Recordset or Record by using the OleDbDataAdapter object.](#)
- [Generate DataAdapter commands automatically by using the CommandBuilder object.](#)
- [Generate DataAdapter commands programmatically.](#)
- [Populate a DataSet by using a DataAdapter.](#)
- [Update a database by using a DataAdapter.](#)
- [Resolve conflicts between a DataSet and a database by using the DataAdapter.](#)
- [Respond to changes made to data at the data source by using DataAdapter events.](#)
- [Perform batch operations by using DataAdapters.](#)

**Manage XML data with the XML Document Object Model (DOM).**

- [Read XML data into the DOM by using the Load method.](#)
- [Modify an XML document by adding and removing nodes.](#)
- [Modify nodes in an XML document.](#)
- [Write data in XML format from the DOM.](#)
- [Work with nodes in the XML DOM by using XmlNamedNodeMap and the XmlNodeList.](#)
- [Handle DOM events.](#)
- [Modify XML declaration.](#)

**Read and write XML data by using the XmlReader and XmlWriter.**

- [Read XML data by using the XmlReader.](#)
- [Read all XML elements and attribute content.](#)
- Read specific element and attribute content.
- [Read XML data by using the XmlTextReader class.](#)
- [Read node trees by using the XmlNodeReader.](#)
- [Validate XML data by using the XmlValidatingReader.](#)
- [Write XML data by using the XmlWriter.](#)

**Creating Custom Web Controls**

#### **Create a composite Web application control.**

- [Create a user control.](#)
- [Convert a Web Forms page to a user control.](#)
- [Include a user control in a Web Forms page.](#)
- [Manipulate user control properties.](#)
- Handle user control events within the user control code-declaration block or code-behind file.
- [Create instances of user controls programmatically.](#)
- Develop user controls in a code-behind file.
- [Create a templated user control.](#)

#### **Create a custom Web control that inherits from the WebControl class.**

- Create a custom Web control.
- [Add a custom Web control to the Toolbox.](#)
- Individualize a custom Web control.
- [Create a custom designer for a custom Web control.](#)

#### **Create a composite server control.**

- [Create a base class for composite controls.](#)
- [Create a composite control.](#)

#### **Develop a templated control.**

- [Create a templated control.](#)
- [Develop a templated data-bound control.](#)

## **Tracing, Configuring, and Deploying Applications**

#### **Use a Web setup project to deploy a Web application to a target server.**

- [Create a Web setup project.](#)
- [Configure deployment properties for a Web setup project.](#)
- [Install a Web application on a target server.](#)

#### **[Copy a Web application to a target server by using the Copy Web tool.](#)**

#### **[Precompile a Web application by using the Publish Web utility.](#)**

#### **Optimize and troubleshoot a Web application.**

- [Customize event-level analysis by using the ASP.NET health-monitoring API.](#)
- [Use performance counters to track the execution of an application.](#)
- [Troubleshoot a Web application by using ASP.NET tracing.](#)
- [Optimize performance by using the ASP.NET Cache object.](#)

## **Customizing and Personalizing a Web Application**

#### **Implement a consistent page design by using master pages.**

- [Create a master page.](#)

- [Add a ContentPlaceHolder control to a master page.](#)
- Specify default content for a ContentPlaceHolder.
- [Reference external resources in a master page.](#)
- Define the content of a particular page in a content page.
- [Create a content page.](#)
- [Add content to a content page.](#)
- [Reference a master page member from a content page.](#)
- [Handle events when using master pages.](#)
- [Create a nested master page.](#)
- [Change master pages dynamically.](#)

#### **Customize a Web page by using themes and user profiles.**

- [Apply a theme declaratively.](#)
- [Apply a theme programmatically.](#)
- [Apply a user-selected theme programmatically.](#)
- [Define custom themes.](#)
- [Define the appearance of a control by using skins.](#)
- [Enable users to personalize an application by using Web Parts.](#)
- [Track and store user-specific information by using user profiles.](#)
- [Personalize a Web page by dynamically adding or removing child controls in a Placeholder control at run time.](#)

#### **Implement Web Parts in a Web application.**

- [Track and coordinate all Web Parts controls on a page by adding a WebPartManager control.](#)
- [Connect Web Parts to each other by using connection objects.](#)
- [Divide a page that uses Web Parts into zones by using WebPartZones.](#)
- [Present a list of available Web Parts controls to users by using CatalogPart controls.](#)
- [Enable users to edit and personalize Web Parts controls on a page by using EditorPart controls.](#)

## **Implementing Authentication and Authorization**

#### **Establish a user's identity by using forms authentication.**

- [Configure forms authentication for a Web application by using a configuration file.](#)
- [Enable cookieless forms authentication by setting the cookieless attribute.](#)
- [Use membership APIs and the Membership class to manage users.](#)
- [Enable anonymous identification.](#)

#### **Use authorization to establish the rights of an authenticated user.**

- [Manage roles in the Web Site Administration Tool.](#)
- [Ascertain whether a specific user is in role.](#)
- [Get the roles for a specific user by using the Roles object or the User object.](#)
- [Store role information in a cookie.](#)
- [Restrict access to files by using file authorization.](#)
- [Restrict access to portions of an application by using URL authorization.](#)

#### **Implement Microsoft Windows authentication and impersonation.**

- [Establish a user's identity by using Windows authentication.](#)
- [Use impersonation to control access to resources.](#)

**Use login controls to control access to a Web application.**

- [Use the Login Web server control.](#)
- [Use the LoginView Web server control to view a user's login status.](#)
- [Use the PasswordRecovery Web server control to allow a user to recover a password.](#)
- [Use the LoginStatus Web server control to display either a login or logout link.](#)
- [Use the LoginName Web server control to display a user's login name on a Web page.](#)
- [Use the CreateUserWizard Web server control as a UI for creating new Web application user accounts.](#)
- [Use the ChangePassword Web server control to allow users to change their passwords.](#)
- [Specify the membership provider used for logging on.](#)
- Configure a mail server so that login controls can be used to send e-mail messages to users.

**Creating ASP.NET Mobile Web Applications**

[Create a mobile Web application project.](#)

[Use device-specific rendering to display controls on a variety of devices.](#)

[Use adaptive rendering to modify the appearance of Web server controls.](#)

[Use the mobile Web controls to display content on a device.](#)

# Chapter I

## Creating and Programming a Web Application

---

### Section I

#### Create and configure a Web application.

Topic A: Create a new Web application

##### To create a new Web project

1. On the **File** menu, point to **New** and then click **Web Site** to display the **New Web Site** dialog box.
2. In the **Templates** pane, select the type of Web project you want to create.
  - ASP.NET Web Site
  - ASP.NET Web Service
  - Empty Web Site
3. In the **Language** drop down, specify the primary language that you would like this to be created with.
  - Visual Basic
  - Visual C#
  - Visual J#
4. In the **Location** drop down, specify the place where this will be stored.
  - File System
  - HTTP
  - FTP

For example, to create a project titled myWebApp at the default localhost location, the URL string placed in the Location text box will look like this:

```
http://localhost/myWebApp
```

This allows for exact placement of your project files in a directory on a Web server.

**Note** If the local computer you are working on is also a Web server, you can create your Web project on that computer. The first time you start Visual Studio, the default location of a Web project is your local computer. In subsequent Visual Studio sessions, you can select the location of a previously used Web server from a drop-down list. If you want to choose a different server, you can click the **Browse** button to find other servers on your network, or you can type in the URL to a server (for example, http://domain.com) and press ENTER.

5. Click **OK** to create the Web project. Visual Studio connects to the server and adds the appropriate project items to Solution Explorer in the project node.

Topic B: Add Web Forms pages to a Web Application

1. On the **File** menu, point to **New** and then click **File** to display the **Add New Item** dialog box.
2. In the **Templates** pane, select **Web Form**
3. In the **Language** drop down, specify the primary language that you would like this to be created with.
  - Visual Basic
  - Visual C#
  - Visual J#
4. Provide a unique name for this Web Form
5. Click **OK** and a new web form will be created in the web project that you are currently in.

## Section II

### Add and configure Web server controls.

#### Topic A: Add Web server controls to a Web Form

##### To add a control using ASP.NET syntax

1. Switch to HTML view.
2. Type the element representing the control into the .aspx file. In Visual Studio, you do this in HTML view. The exact syntax you use depends on the control you are adding, but in general the following applies:

Controls must include the attribute **runat="server"**.

Set the control's **ID** attribute unless the control is part of a complex control and will be repeated (as in the **Repeater**, **DataList**, and **DataGrid** controls).

Web server controls are declared with an XML tag that references the **asp** namespace. Control declarations must be properly closed. You can specify an explicit closing tag, or, if the control has no child elements, you can specify a self-closing tag. The only exceptions are HTML input controls that cannot have child elements, such as the input controls (for example, [HtmlInputText](#), [HtmlImage](#), and [HtmlButton](#)).

Control properties are declared as attributes.

The following examples show typical declarations for Web server controls:

```
<!-- Textbox Web server control -->
<asp:textbox id=TextBox1 runat="Server" Text=""></asp:textbox>

<!-- Same, but with self-closing element -->
<asp:textbox id=TextBox1 runat="Server" Text="" />

<!-- Web DropDownList control, which contains subelements -->
<asp:DropDownList id=DropDown1 runat="server">
  <asp:ListItem Value="0">0</asp:ListItem>
  <asp:ListItem Value="1">1</asp:ListItem>
  <asp:ListItem Value="2">2</asp:ListItem>
  <asp:ListItem Value="3">3</asp:ListItem>
</asp:DropDownList>

<asp:Repeater id=Repeater2 runat="server">
  <HeaderTemplate>
    Company data:
  </HeaderTemplate>
  <ItemTemplate>
    <asp:Label runat="server"
      Font-Name="verdana" Font-Size="10pt"
      Text='<%= Container.DataItem.Name %>' />
    <asp:Label runat="server"
      Font-Name="verdana" Font-Size="10pt"
      Text='<%= Container.DataItem.Ticker %>' />
  </ItemTemplate>
  <SeparatorTemplate>
    /
  </SeparatorTemplate>
</asp:Repeater>
```

#### Topic B: Configure the properties of Web server controls programmatically

1. Switch to the source code view
2. Place your cursor on the line where you would like to change the controls properties
3. Type the name of id of the control, such as btnCancel, and then type a period
4. The intellisense should be revealed, now you can choose which property to edit

## Topic C: Configure Web server control properties by using the Microsoft Visual Studio Property Editor

1. Switch to the designer view for the page that you would like to change the property
2. Right-click on the control that you want to edit the properties of
3. Choose the property option on the context menu
4. Now you are able to edit the properties of this control using the Property Editor

## Topic D: Specify whether events of a control cause a Web Form to post to the server.

An event that is OnChanged orIndexChanged will cause a postback. To cause a control's events to postback please follow these instructions.

1. Switch to the source view for the page containing the control you want to add event to
2. In the properties of the control add AutoPostBack = true
3. Setting the autopostback to true will cause the page to post back whenever an even is fired for the control, by default autopostback is false

## Topic E: Configure a control to receive postback events.

Simply subscribing to an event in the code will allow you to receive a postback event. In order to accomplish this, simply do the following couple of steps.

1. Switch to the codebehind view and add the following in the on\_init event (you will have add this if it is not already visible, try to override it)
2. `btnCancel.Click += new EventHandler(btnCancel_Click);`

## Topic F: Access controls in Web Forms pages when working with naming containers and child controls.

There are a couple of approaches you could take. Below is one set of sample code that accesses a button called btnCancel from its parent web form.

```
Button btnCancel = this.page.FindControl("btnCancel");
```

## Topic G: Create HTML server controls in the designer.

1. Switch to design view on the page you want to add the HTML control
2. Choose a HTML control from the toolbox and drop it on the page
3. Right-click on the new control and select Run as Server Control
4. Now, if you switch to source view you should see the `runat=server` on this control

## Topic H: Set HTML server control properties programmatically.

You can switch to the code behind and access the HTML server control by its id

```
this.btnCancel.Text = "Cancel";
```

## Topic I: Use HTML server controls to programmatically access HTML tags.

You can programmatically access HTML tags by adding the `runat="server"` in the source view for the HTML tag that you would like to access. So, if you want to access a HTML `<table>` tag, the resulting tag will look like `<table id="tblMyTable" runat="server">`. Then to access this tag in your C# code, simply access it by its id.


## Topic J: Create HTML controls as elements in an HTML document.

This is very straightforward, you simply drag and drop a HTML control from the toolbox or add it yourself to the HTML page.

## Topic K: Use the AdRotator Web server control to manage banners and pop-up windows.

First you add the **AdRotator** control to a page, and then link it to the ads you wish to display. Link ads to the control either by using a separate XML file containing the ad information or by linking the ads to the control within an event handler at run time.

### To add an AdRotator control to a Web Forms page

1. In Design view, drag an **AdRotator** control from the **Web Forms** tab of the Toolbox onto the page.
2. Link the **AdRotator** to an XML advertisement file:
  - a. Select the **AdRotator**, open the Properties window, and then click the ellipsis button () next to the **AdvertisementFile** property. The **Select Advertisement XML File** dialog box appears.
  - b. Navigate to the location of the XML file holding the information about your ads and click **Open**.

You can dynamically select ads by creating custom logic within an **AdCreated** event handler. If you have more than one **AdRotator** control in your project it is better to create a global event handler that responds to all of your **AdRotator** controls. In this way the logic for rotating the ads in each control can be handled by the same event.

### To select ads using the AdCreated event

1. Create an **AdCreated** event handler for the control.  
The second parameter passed in the handler contains a reference to the ad being created.
2. Set properties of the ad object to specify the image and navigation URLs of the ad to deploy.

The following example shows how you can set the **ImageUrl**, **NavigateUrl**, and **AlternateText** properties of an **AdRotator** control (`AdRotator1`) in the **AdCreated** event handler:

```
// C#
protected void AdRotator1_AdCreated (object sender,
    System.Web.UI.WebControls.AdCreatedEventArgs e)
{
    e.ImageUrl = "images/msft ad.gif";
    e.NavigateUrl = "http://www.microsoft.com/";
    e.AlternateText = "Where do you want to go today?";
}
```

Topic L: Use the Button Web server control to send a command to the server when a button is clicked.

Probably one of the most straightforward questions you can be asked. You simply need to subscribe to the Button click event. Or you could use the OnClick action in the source view of the button inside your page. If you would like more information on events then please read [this article](#) on events.

Topic M: Display a calendar on a Web page by using the Calendar Web server control.

You can either add the calendar control to your page by dragging and dropping it or switching to the source view and use <asp:Calendar>. Make sure that you specify an id and that the control has a runat="server". For more information on the calendar control read [this article](#).

Topic N: Implement the CheckBox Web server control.

#### [To add a CheckBoxList control to a Web Forms page](#)

1. In **Design** view, drag the **CheckBoxList** control from the Toolbox onto the page.
2. Optionally, change the orientation of the caption by setting the **TextAlign** property.

#### [To get or set CheckBox Web server control selection](#)

Get or set the **CheckBox** control's **Checked** property. A value of **true** means the check box is selected.

**Note** Testing the value of a check box does not tell you if the user changed the value of the control, only if it is selected. For details about how to check for a change in the control, see [Responding to User Selection in a CheckBox Web Server Control](#).

In the following example, if the check box `chkCanWeSendEmail` is not selected, the check box `chkSendInHTMLFormat` is automatically cleared as well.

```
// C#
if(chkCanWeSendEmail.Checked == false){
    chkSendInHTMLFormat.Checked = false;
}
```

Topic O: Implement the FileUpload Web server control.

This is similar to the above examples; you simply add the control to the page then respond to a button event to fire the upload process. For a more detailed implementation from Microsoft, please follow the following [link](#).

Topic P: Create and manipulate links on a Web Form by using the HyperLink Web server control.

Simply add a Hyperlink control to a page <asp:HyperLink> and then you can access this and change the NavigateURL property from the code behind.

```
<asp:HyperLink id="lnkMicrosoft" runat="server" NavigateURL=www.microsoft.com>
```

## Topic Q: Display an image on a Web Form by using the Image Web server control.

Similar to Topic P, except you add a <asp:Image> to the page and set its ImageURL to the URL of the image that you would like to add to the web form. The important thing to note, is that the URL can be a relative path to the image file.

## Topic R: Implement a button on a Web Form by using the ImageButton Web server control.

You can get by with dragging and dropping an ImageButton control on to a web form and then set its property for ImageURL to the path of the image that you would like to use to represent this button.

## Topic S: Define hotspot regions within an image by using the ImageMap Web server control.

Below is an example of this control being used to set an area of an image that can direct a user to a certain URL whenever the area is clicked. If you would like more information on this control please check out the information [here](#).

```
<asp:ImageMap ID="ImageMap1" ImageUrl="~/Images/MapTest.PNG" runat="server"
    HotSpotMode="PostBack">
  <asp:RectangleHotSpot Bottom="100" Left="100" Right="200"
    PostBackValue="Northeast" /> <asp:RectangleHotSpot Bottom="200"
    Right="100" Top="100" PostBackValue="Southwest" />
  <asp:RectangleHotSpot Bottom="200" Left="100" Right="200" Top="100" />
  <asp:CircleHotSpot Radius="50" X="100" Y="100" HotSpotMode="Navigate"
    NavigateUrl="http://www.contoso.com/" /> <asp:PolygonHotSpot
    Coordinates="0,0;100,100;200,0" PostBackValue="Triangle1" />
</asp:ImageMap>
```

## Topic T: Use the Label Web server control to display customized text on a Web page.

This is very easy, simply create an `<asp:Label>` control and set its text property to the text that you would like to see displayed on the page.

## Topic U: Display a hyperlink style button on a Web Form by using the LinkButton Web server control.

Simply use the `<asp:LinkButton>` control and set its `NavigateURL` to the place where you would like to navigate to whenever the button is clicked.

## Topic V: Display lists of information by using controls that derive from the ListControl class.

Several controls derive from the `ListControl` class, these include the following:

- CheckBoxList
- DropDownList
- ListBox
- RadioButtonList
- Listitem

Remember, you can simply bind a `datasource` to add data to the list that is displayed. You should select one of the columns in the `datasource` to be the text displayed and another to be the value that represents that text.

## Topic W: Create a Web Form with static text by using the Literal Web server control.

You can use a literal control similar to how you would use a label to display text. Simply add the literal control to the page and set its text property to the text that you would like displayed. Keep in mind that it will take if you would like to change the way the text looks, you will need to set the text property to include the html.

```
myLiteral.Text = "<span class='blueText'>Hello</span>";
```

## Topic X: Implement pagination for controls on a page by using the Pager Web server control.

This control does not appear to exist in 2.0. However, what does exist is a `PageCatalogPart` webpart. Therefore, there should not be any questions on the test about this page control.

## Topic Y: Use the Panel Web server control to arrange controls in groups on a page.

The page control can be accessed using the `<asp:Page>` control in your source view. You can either write this manually or drag the control onto the page from the toolbox. This control is

useful for organizing different sections of a page. Here is an example that puts a textbox in one area and a label in another. You can then take this example further and group more controls in the panel. Keep in mind, that if you set a panel's visible property to false, then everything within that panel will not be visible. This example does not implement all of the required properties, but is more of an example to show how you can group similar items in a panel.

```
<asp:Panel id="pnlTextBox" runat="server">
    <asp:Label Text="enter your text" runat="server" />
    <asp:TextBox runat="server" />
</asp:Panel>
<asp:Panel id="pnlLabel" runat="server">
    <asp:Label Text="This is an example:" runat="server" />
</asp:Panel>
```

## Topic Z: Create a container for a group of View controls by using the MultiView Web server control.

The following was taken from [http://msdn2.microsoft.com/en-us/library/ms227665\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms227665(VS.80).aspx)

The **MultiView** and **View** Web server controls act as containers for other controls and markup, and provide a way for you to easily present alternate views of information. You can use the **MultiView** and **View** controls to perform tasks such as the following:

Provide alternate sets of controls based on user choice or other conditions. For example, you might allow users to select from a list of feeds, each of which is configured in a separate **View** control. You can then display the **View** control that contains the user's choice of feeds. You can use the **MultiView** and **View** controls as an alternative to creating multiple **Panel** controls.

Create a multi-page form. The **MultiView** and **View** controls can provide behavior that is similar to the **Wizard** control. The **Wizard** control is particularly suited to creating forms that users fill in step by step. The **Wizard** control also includes support for more built-in UI elements, such as a header and footer, for **Previous** and **Next** buttons, and for templates. You might use a **MultiView** control in place of a **Wizard** if you wanted to create a display that changed based on condition (rather than sequentially) or if you did not need the extra features supported by the **Wizard** control.

You can move between views by setting the **MultiView** control's **ActiveViewIndex** property to the index value of the **View** control to display. The **MultiView** control also includes support for navigation buttons that you can add to each **View** control.

To create navigation buttons, you can add a button control (**Button**, **LinkButton**, or **ImageButton**) to each **View** control. You can then set the **CommandName** and **CommandArgument** properties of each button to reserved values to cause the **MultiView** control to move to another view. The following table lists the reserved **CommandName** values and the corresponding **CommandArgument** values.

CommandName value	CommandArgument value
<b>NextView</b>	(no value)
<b>PrevView</b>	(no value)
<b>SwitchViewByID</b>	ID of the <b>View</b> control to switch to.
<b>SwitchViewByIndex</b>	Index number of the <b>View</b> control to switch to.

The following example shows a **MultiView** control with three **View** controls. Each **View** control contains a **Button** control that moves to a specific **View** control.

```
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
    <asp:View ID="View1" runat="server"> View 1<br /> <br />
        <asp:Button ID="Button1" runat="server" CommandArgument="View2"
            CommandName="SwitchViewByID" Text="Go to View2" />
    </asp:View>
    <asp:View ID="View2" runat="server"> View 2<br /> <br />
```

```
        <asp:Button ID="Button2" runat="server" CommandArgument="View3"
                    CommandName="SwitchViewByID" Text="Go to View 3" />
    </asp:View>
    <asp:View ID="View3" runat="server"> View 3<br /> <br />
        <asp:Button ID="Button3" runat="server" CommandArgument="View1"
                    CommandName="SwitchViewByID" Text="Go to View 1" />
    </asp:View>
</asp:MultiView>
```

Topic  $\alpha$ : Use the View Web server control to create a Web application.

This is similar to the above steps. Simply encapsulate the parts of your page that you would like to be included into a particular view. Then, if you would like to navigate between them, follow the above steps.

Topic  $\beta$ : Create a mutually exclusive set of choices by using the RadioButton Web server control.

In order to do this, assign the GroupName property with the same text for all of the radio buttons that you would like to be mutually exclusive. Remember, mutually exclusive means that when one radio button is selected then the others are not selected within the same group.

Topic  $\delta$ : Construct a table by using the Table, TableRow, and TableCell Web server controls.

If you know how to create a table using the html controls, then you can create a table server control pretty similarly. Instead of <table> use <asp:Table>, instead of <tr> use <asp:TableRow>, and instead of <td> or <th> use <asp:TableCell>.

## Topic κ: Enable users to type information into a Web Form by using the TextBox Web server control.

The textbox control looks exactly as you would expect it to, it is `<asp:TextBox>`. In order to accept input, make sure to place the textbox server control inside of a form tag. An important point to consider is if you want your textbox to allow multiple lines of input or not, if you do then simply set the multiline property to true.

## Topic η: Create a wizard by using the Wizard Web server control to collect data through multiple steps of a process.

The following was taken from [http://msdn2.microsoft.com/en-us/library/fs0za4w6\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/fs0za4w6(VS.80).aspx)

With the **Wizard** control, you use discrete steps to collect data, which allows users to navigate between steps at their discretion and creates a simpler user experience. As a developer, you do not have to worry about making your data persist across pages because the control maintains state while the user completes the various steps.

### Wizard Steps

The **Wizard** control uses steps to delineate different sections of user data input. Each step within the control is given a **StepType** to indicate whether it is the beginning step, intermediate step, or completion step. The wizard can have as many intermediate steps as needed. You can add different controls, such as a [TextBox](#) or [ListBox](#) control, to collect user input. When you reach the [Complete](#) step, all of your data is accessible. The following code example illustrates the **Wizard** control with two steps.

```
<asp:Wizard ID="Wizard1" Runat="server">
  <WizardSteps>
    <asp:WizardStep Runat="server" Title="Step 1"> </asp:WizardStep>
    <asp:WizardStep Runat="server" Title="Step 2"> </asp:WizardStep>
  </WizardSteps>
</asp:Wizard>
```

Within each step, you can add controls and labels, and accept user data. The **Wizard** control will help manage which step to display, and help maintain the collected data.

### Wizard Navigation

The **Wizard** control features both linear and non-linear navigation. The control's state management allows the user to move forward and backward between steps, and it allows the user to select individual steps at any point, as long as the sidebar is displayed. You can customize the text for navigation in the control's root **asp:Wizard** element by using the [StepNextButtonText](#), [StepPreviousButtonText](#), and [FinishCompleteButtonText](#) properties.

```
<asp:Wizard ID="Wizard1" Runat="server"
  StepNextButtonText=" Next >> " StepPreviousButtonText=" << Previous "
  FinishCompleteButtonText=" Done! ">
```

Topic ε: Use the XML Web server control to create XML data at the location of the control.

You can accomplish this by adding the XML server control to the page. Do this by either typing `<asp:Xml>` or by dragging the control from the toolbox directly. Next, set the `DocumentSource` to the path to the xml file you would like displayed. You can also set an xsl file to transform the document by setting the `TransformSource` to an xsl file.

Topic φ: Customize the appearance of Web server controls by using Web control templates.

[To create a Web server control template using ASP.NET syntax](#)

In the ASP.NET page, insert an element inside the control to identify what template you are creating, as shown in the following example:

```
<asp:DataList id="DataList1" runat="server">
  <ItemTemplate>
  </ItemTemplate>
</asp:DataList>
```

Inside the template element, add HTML text and other controls as the template's content. Include property and data-binding values for the embedded controls using normal syntax, as shown in the following example:

```
<asp:DataList id="DataList3" runat="server">
  <ItemTemplate> Name: <asp:Label ID="Label2" runat="server"
    Text='<%# DataBinder.Eval(Container, "DataItem.EmployeeName")%'>/>
  </ItemTemplate>
</asp:DataList>
```

The following example shows a complete declaration for a **DataList** Web server control with simple templates declared for the **Header**, **Item**, and **Separator** templates.

```
<asp:datalist id="DataList2" runat="server" > <HeaderTemplate> Items matching your query:
</HeaderTemplate> <ItemTemplate> Name: <asp:Label id="Label1" runat="server"
Text='<%# DataBinder.Eval(Container, "DataItem.EmployeeName")%'></asp:Label>
</ItemTemplate> <SeparatorTemplate> <br><hr> </SeparatorTemplate> </asp:datalist>
```

Topic γ: Programmatically edit settings in a Web site's configuration file.

This is very useful new feature in .NET 2.0.

Topic ι: Dynamically add Web server controls to a Web Forms page.

Simply create a new control that you want to add, such as a textbox. Next, simply add it to the page's control collection. You can do this by the following code `Page.Controls.Add(new TextBox)`.

**Section II**  
**Create Event Handlers for Pages and Controls.**