

ASP.NET week 2 : webform control 1

Contributed by Administrator
Monday, 01 June 2009

materi week 2 asp.net. Membahas tentang masing2 webform control. Materinya diambil dari MSDN.

No	Materi	Sub Materi	t	note	Link	1	WebForm control	- Link Button
5	imageurl, navigateurl, target	button			Hyperlink	5	- Property yg penting:	
	items, selectedvalue,selectedindex	hyperlink			DropDownlist	5	- Property yg Penting :	
		dropdownlist			ListBox	10	- Bisa memilih lebih dari	
	1 item - Item yang penting :							
	items,selectionMode							
	listBox1.Items							
	dan checkboxlist							
10	- Bedanya apa?							
5	- Bedanya apa ?							
	Panel							
15	- Fungsinya apa?							
	Placeholder							
	AdRotator							
30	- Fungsinya apa?							
40	- Membuat suatu input misal biodata, yang memanfaatkan sebanyak mungkin web control diatas. Inputan dimasukkan ke suatu panel, dan saat diproses diganti yg visible panel yang ambil nilai2nya.							
	latihan							

Using ASP.NET button Web server controls on ASP.NET Web pages allows users to indicate that they have completed the form or that they want to perform a specific command. Web server controls include three kinds of buttons, each of which appears differently on Web pages. Types of Buttons

The following table lists the types of buttons that you can create by using Web server controls. Control Description

Button Web server control

Presents a standard command button, which is rendered as an HTML input element.

LinkButton Web server control

Renders as a hyperlink in the page. However, it contains client-side script that causes the form to be posted back to the server. (You can create a true hyperlink by using the HyperLink Web server control.)

ImageButton Web server control

Allows you to specify a graphic as a button. This is useful for presenting a rich button appearance. The ImageButton controls also pinpoint where in the graphic a user has clicked, which allows you to use the button as an image map.

Button Events

When a user clicks any of the three types of buttons, a form is submitted to the server. This causes the Web page to be processed and any pending events to be raised in server-based code. The buttons can also raise their own Click events, for which you can write event handlers. Button Controls and Validation

If a page contains ASP.NET validator controls, by default, clicking a button control causes the validator control to perform its check. If client-side validation is enabled for a validator control, the page is not submitted if a validation check has failed.

The following table describes the properties supported by button controls that enable you to control the validation process more precisely. Property Description

CausesValidation property

Specifies whether clicking the button also performs a validation check. Set this property to false to prevent a validation check.

ValidationGroup property

Allows you to specify which validators on the page are called when the button is clicked. If no validation groups are established, a button click calls all of the validators that are on the page. Button Postback Behavior

When users click a button control, the page is posted back to the server. By default, the page is posted back to itself, where the same page is regenerated and the event handlers for controls on the page are processed.

You can configure buttons to post the current page to another page. This can be useful for creating multi-page forms. For details, see Cross-Page Posting in ASP.NET Web Pages. By default, the Button control submits the page by using an HTML POST operation. The LinkButton and ImageButton controls cannot directly support an HTML POST operation.

Therefore, when you use those buttons, they add client script to the page that allows the controls to submit the page programmatically. (The LinkButton and ImageButton controls therefore require that client script is enabled on the browser.) [HyperLink Web Server Control Overview](#)

The HyperLink Web server control creates links on a Web page that allow users to move from page to page in your application.

Advantages of Using the HyperLink Control

The primary advantage of using a HyperLink control is that you can set link properties in server code. For example, you can dynamically change the link text or target page based on conditions in your page.

Another advantage of using the HyperLink control is that you can use data binding to specify the target URL for the link (and parameters to be passed with the link, if necessary). A typical example is to create HyperLink controls based on a list of products; the target URL points to a page where the user can read more detail about the product. The HyperLink control can display clickable text or an image. Unlike most Web server controls, the HyperLink control does not raise an event in server code when users click it. Instead, the control simply performs navigation. [DropDownList Web Server Control Overview](#)

The DropDownList Web server control allows users to select an item from a predefined list. It differs from the ListBox Web server control in that the list of items remains hidden until users click the drop-down button. In addition, the DropDownList control differs from the ListBox control in that it does not support multi-selection mode. [Modifying the Appearance of the DropDownList Control](#)

You can control the look of the DropDownList control by setting its height and width in pixels. Some browsers do not support setting the height and width in pixels and will use the row-count setting instead.

You cannot specify the number of items that are displayed in the list when users click the drop-down button. The length of the displayed list is determined by the browser. [List Items](#)

The DropDownList control is actually a container for the list items, which are of type ListItem. Each ListItem object is a separate object with its own properties. These properties are described in the following table. [Property Description](#)

Text

Specifies the text that is displayed in the list.

Value

Contains the value that is associated with an item. Setting this property allows you to associate a value with a specific item without displaying it. For example, you can set the Text property to the name of a U.S. state and the Value property to its postal abbreviation.

Selected

Indicates whether the item is selected by means of a Boolean.

To work with list items programmatically, use the Items collection of the DropDownList control. The Items collection is a standard collection, and you can add item objects to it, delete items, clear the collection, and so on.

The currently selected item is available in the DropDownList control's SelectedItem property. [Binding Data to the Control](#)

You can use a DropDownList Web server control to list options that are made available to the page using a data source control. Each item in the DropDownList control corresponds to an item — typically a row — in the data source.

The control displays one field from the source. Optionally, you can bind the control to a second field to set the value of an item, which does not display. [DropDownList Events](#)

The DropDownList control raises an event — the SelectedIndexChanged event — when users select an item. By default, this event does not cause the page to be posted to the server, but you can cause the control to force an immediate post by setting the AutoPostBack property to true. [ListBox Web Server Control Overview](#)

The ListBox Web server control enables users to select one or more items from a predefined list. It differs from a DropDownList control in that it can display multiple items at once and it optionally enables the user to select multiple items. [ListBox Control Appearance](#)

The ListBox control is typically used to display more than one item at once. You can control the look of the list in these ways:

- [Number of rows displayed](#). You can set the control to display a specific number of items. If the control contains more items than that, it displays a vertical scroll bar.
- [Height and width](#). You can set the size of the control using pixels. In that case, the control ignores the number of rows you have set and displays as many as will fit into the height of the control. Some browsers do not support setting the height and width in pixels and will use the row count setting.

As with other Web server controls, you can use style objects to specify the appearance of the control. For details, see [ASP.NET Web Server Controls and CSS Styles](#). [Items](#) |

The `ListBox` control is a container for one or more list items. Each list item is an object of type `Listltem` with its own properties, described in the following table.

Property	Description
----------	-------------

Text

The text displayed in the list.

Value

The value associated with an item. Setting this property enables you to associate a value with a specific item without displaying it. For example, you can set the `Text` property to the name of a U.S. state and the `Value` property to its postal abbreviation.

Selected

A Boolean value indicating whether the item is selected.

To work with items programmatically, you work with the `Items` collection of the `ListBox` control. The `Items` collection is a standard collection, and you can add item objects to it, delete items, clear the collection, and so on.

The currently selected item is available in the `ListBox` control's `SelectedItem` property. For ease of use, the `ListBox` control also supports a property called `SelectedItem`. If the control is set to single-selection mode, this property returns the one selected item, making it unnecessary to loop through the entire `Items` collection to get at the current selection.

Single vs. Multiple Selection

Users can normally select a single item in the list by clicking it. If you set the `ListBox` control to enable multiple selections, users can hold down the `CTRL` or `SHIFT` key while clicking to select multiple items. [Binding Data to the Control](#)

You can use a `ListBox` Web server control to list options that are made available to the page using a data source control. Each item in the `ListBox` control corresponds to an item — typically a row — in the data source.

The control displays one field from the source. Optionally, you can bind the control to a second field to set the value (which is not displayed) of an item.

ListBox Events

The `ListBox` control raises the `SelectedIndexChanged` event when users select an item. By default, this event does not cause the page to be posted to the server, but you can cause the control to force an immediate postback by setting the `AutoPostBack` property to `true`. [CheckBox and CheckBoxList Web Server Controls Overview](#) You can use two types of Web server controls to add check boxes to a Web Forms page: individual `CheckBox` controls or a `CheckBoxList` control. Both controls provide a way for users to input Boolean data: `true` or `false`, `yes` or `no`. [CheckBox Control versus CheckBoxList Control](#)

You add individual `CheckBox` controls to a page and work with them singly. Alternatively, the `CheckBoxList` control is a single control that acts as a parent control for a collection of check-box list items. It derives from a base `ListControl` class, and therefore works much like the `ListBox`, `DropDownList`, `RadioButtonList`, and `BulletedList` Web server controls. For that reason, many of the procedures for working with the `CheckBoxList` control are the same as the procedures for the other list Web server controls.

Each type of control has advantages. By using individual `CheckBox` controls, you get more control over the layout of the check boxes on the page than by using the `CheckBoxList` control. For example, you can include text (that is, non-check-box text) between each check box. You can also control the font and color of individual check boxes.

The `CheckBoxList` control is a better choice if you want to create a series of check boxes from data in a database. (You can still bind an individual `CheckBox` control to data.) [CheckBox Events](#)

Events work differently between individual `CheckBox` controls and the `CheckBoxList` control. Individual `CheckBox` Controls Individual `CheckBox` controls raise the `CheckedChanged` event when users click the control. By default, this event does not cause the page to be posted to the server, but you can have the control force an immediate post by setting the `AutoPostBack` property to `true`. [CheckBoxList Control](#)

In contrast, the `CheckBoxList` control raises a `SelectedIndexChanged` event when users select any check box in the list. By default, the event does not cause the form to be posted to the server, although you can specify this option by setting the `AutoPostBack` property to `true`.

As with individual `CheckBox` controls, it is more common to test the state of the `CheckBoxList` control after the form has been posted some other way. [Binding Data to the Control](#)

As with any Web server control, you can bind an individual `CheckBox` control to a data source, and you can bind any property of the `CheckBox` control to any field of the data source. For example, it is typical to set the control's `Checked`

property from information in a database. You can also bind a `CheckBoxList` control to a data source. In that case, the check boxes each represent a different record in the data source. **RadioButton and RadioButtonList Web Server Controls Overview** You can use two types of Web server controls to add radio buttons to an ASP.NET Web page: individual `RadioButton` controls or a `RadioButtonList` control. Both controls allow users to select from a small set of mutually exclusive, predefined choices. The controls allow you to define any number of radio buttons with labels and to arrange them horizontally or vertically. **RadioButton Control vs. RadioButtonList Control**

You add individual `RadioButton` controls to a page and work with them singly. Typically, you group two or more individual buttons together. In contrast, the `RadioButtonList` control is a single control that acts as a parent control for a collection of radio button list items. It derives from a base `ListControl`, and therefore works much like the `ListBox`, `DropDownList`, `BulletedList`, and `CheckBoxList` Web server controls. Therefore, many of the procedures for working with the `RadioButtonList` control are the same as they are for the other list Web server controls.

Each type of control has advantages. Individual `RadioButton` controls give you more control over the layout of the radio button group. For example, you can include text (that is, non-radio-button text) between radio buttons.

The `RadioButtonList` control does not permit you to insert text between the buttons, but is far easier to use if you want to bind the buttons to a data source. It is also slightly easier to write code that determines which button has been selected.

Grouping Radio Buttons

Radio buttons are rarely used singly. Instead, they are grouped to provide a set of mutually exclusive options. Within a group, only one radio button can be selected at a time. You can create grouped radio buttons in these ways:

- Add individual `RadioButton` Web server controls to a page and then manually assign them all to a group. In this case, the group is an arbitrary name; all radio buttons with the same group name are considered part of a single group.
- Add a `RadioButtonList` Web server control to the page. The list items in the control are automatically grouped.

RadioButton Events

Events work slightly differently between individual `RadioButton` controls and the `RadioButtonList` control. **Individual RadioButton Controls**

Individual `RadioButton` controls raise a `CheckedChanged` event when users click the control. By default, this event does not cause the page to be posted to the server, but you can have the control force an immediate post by setting the `AutoPostBack` property to true. For details about responding to this event directly, see [Responding to a User Selection in a RadioButton Group](#). **Note**

The ability of a `RadioButton` control to post to the server when it is checked requires that the browser support ECMAScript (JScript, JavaScript) and that scripting be enabled on the user's browser. Whether a `RadioButton` control posts to the server or not, it is not usually necessary to create an event handler for the `CheckedChanged` event. Instead, it is more common to test which button is selected when the form has been posted to the server by a control such as a `Button` control. For details, see [Setting and Getting the Selection in a RadioButton Web Server Control](#). **RadioButtonList Control**

In contrast, the `RadioButtonList` control raises a `SelectedIndexChanged` event when users change which radio button in the list is selected. By default, the event does not cause the form to be posted to the server, although you can specify this option by setting the `AutoPostBack` property to true. For details, see [Responding to Changes in a List Web Server Control](#). **Note**

The ability of a `RadioButtonList` control to post to the server when it is selected requires that the browser support ECMAScript (JScript, JavaScript) and that scripting be enabled on the user's browser.

As with individual `RadioButton` controls, it is more common to test the state of the `RadioButtonList` control after the form has been posted some other way. For details, see [Determining the Selection in a List Web Server Control](#). **Binding Data to the Control**

As with any Web server control, you can bind an individual `RadioButton` control to a data source, and you can bind any property of the `RadioButton` control to any field of the data source. For example, you might set the control's `Text` property from information in a database. However, because radio buttons are used in groups, binding a single radio button to a data source isn't a common scenario. Instead, it is more typical to bind a `RadioButtonList` control to a data source. In that case, the data source dynamically generates radio buttons (list items) for each record in the data source. **Image Web Server Control Overview**

The `Image` Web server control allows you to display images on an ASP.NET Web page and manage these images in your own code. **Specifying Graphics Files** You can specify the graphics file for an `Image` control at design time or at run time programmatically. You can also bind the control's `ImageUrl` property to a data source to display graphics based on database information.

Unlike most other Web server controls, the `Image` control does not support any events. For example, the `Image` control does not respond to mouse clicks. Instead, you can create an interactive image by using the `ImageMap` or the `ImageButton` Web server controls. **Specifying Text Elements**

In addition to displaying a graphic, the Image control allows you to specify various types of text for the image, such as the following:

- **ToolTip** This is the text that is displayed in a ToolTip in some browsers.
- **AlternateText** This is the text displayed if the graphics file cannot be found. If no ToolTip property is specified, some browsers will use the AlternateText value as a ToolTip.
- **GenerateEmptyAlternateText** If this property is set to true, the alt attribute of the rendered image element will be set to an empty string.

Panel Web Server Control Overview

The Panel Web server control provides a container control within a Web Forms page that you can use as a parent for static text and for other controls. In addition, you can use the Panel control for a variety of other purposes. Grouping Controls and Markup

You can manage a group of controls and associated markup as a unit by putting them in a Panel control and then manipulating the Panel control. For example, you can hide or show a group of controls inside a panel by setting the panel's Visible property. Forms with Default Buttons

You can put TextBox controls and Button controls inside the Panel control and then define a default button by setting the Panel control's DefaultButton property to the ID of a button in the panel. If users press ENTER while typing in a text box inside the panel, it has the same effect as if the user had clicked the specified default button. This can help users work more efficiently with entry forms. Container for Dynamically Generated Controls

The Panel control provides a convenient container for controls that you create at run time. For details, see Adding ASP.NET Controls Programmatically. You can use the Panel control to create areas on the page that have custom appearance and behavior, such as the following:

- **Creating a grouping box with title** You can set the GroupingText property to display a title. When the page renders, the Panel control is displayed with a box around it that contains a title with the text you specify.
- **Creating areas on the page with a custom color or other appearance** The Panel control supports appearance properties such as BackColor and BorderWidth that you can set to create a unique look for a region on a page.

Placeholder Web Server Control Overview

The Placeholder Web server control enables you to place an empty container control within the page and then dynamically add, remove, or loop through child elements at run time. The control only renders its child elements; it has no HTML-based output of its own.

As an example, you might want to have a variable number of buttons appear on a Web page, depending on options selected by users. That way, users are not confronted with potentially confusing choices that are either unavailable or not relevant to their individual needs. To add child controls to a Placeholder control at run time

1. Create an instance of the control you want to add to the Placeholder control.
2. Call the Add of the Placeholder control's Controls collection, passing it the instance you created.

The following example shows how to add two Button controls as children of a Placeholder control. The code also adds a Literal control in order to add a br tag between the buttons. Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

```
Dim TextBox11 As New System.Web.UI.WebControls.TextBox()
Dim TextBox12 As New System.Web.UI.WebControls.TextBox()
Dim Button11 As New System.Web.UI.WebControls.Button()
TextBox11.Text = "ini TextBox11"
TextBox12.Text = "ini TextBox12"
Button11.Text = "ini Button"
TextBox11.Columns = 15
TextBox11.ReadOnly = True
TextBox12.Columns = 20
Placeholder1.Controls.Add(TextBox11)
Placeholder1.Controls.Add(TextBox12)
Placeholder1.Controls.Add(Button11) End Sub
```

Calendar Web Server Control Overview

You can use the Calendar Web server control to do the following:

- **Display and select dates** The control displays a calendar through which users can navigate to any day in any year. Setting the SelectedDate property causes a specific date to be highlighted in the control. Optionally, users can move to arbitrary dates by clicking a day or moving from month to month. The calendar can be configured to allow users to select multiple dates, either an entire week or an entire month. A representation of the Calendar control displaying the month of October
- **Display appointment or other information in a calendar grid** The Calendar control can display specific details for individual days, such as a to-do list, a schedule of events, or similar information. This feature allows you to display day information from a database.

The Calendar control is based on the .NET Framework DateTime object, and therefore supports the full range of dates that is allowed by that object. Effectively, you can display any date between the years 0 and 9999 A.D.

When the Web Forms page runs, the Calendar control is rendered as an HTML table. Therefore, a number of the control's properties pertain to various aspects of table formatting. A few of these properties are not fully supported in some older browsers, so not all of the formatting features will be available in those browsers. Displaying and Selecting

Dates

The Calendar control displays the dates for one month at a time, with a total of six weeks appearing at once. The control supports several types of dates, which are described in the following table.

Type of date	Description
--------------	-------------

Today's Date

By default, this is set to match the current date on the server. However, you can adjust it so that the date appears correctly for a different locale.

Visible Date

This date determines which month appears in the calendar. The user can move from month to month on the calendar, thereby changing the visible date without affecting today's date. You can navigate between months by setting the visible date programmatically.

Selected Date, Selected Dates

This is the date or date range that the user chooses. In the control, the user can choose a single day, week, or month, but can only select contiguous dates. You can also programmatically set the selected dates; in that case, you can set non-contiguous selected dates.

[Enabling Date Selection](#)

By default, the calendar allows users to click an individual date to select it. If you are using the control as a read-only calendar, you can disable the date-selection functionality.

If date selection is enabled, each day of the calendar contains a `LinkButton` control that raises an event when clicked. If you enable week or month selection, a column of links is added to the left side of the calendar to enable the user to specify which week to select.

[Customizing the Calendar's Appearance](#)

You can set calendar properties to change the colors, size, text, and other visual features of the calendar. There are a number of ways to do this, as shown in the following table.

Customization method	Description
----------------------	-------------

Setting properties

You can set properties to display grid lines, change which day is displayed as the first day of the week, and change the appearance of the month and day names.

Setting extended style properties

You can use properties derived from the `Style` object to set the appearance of particular elements in the calendar, such as the current date or the title bar containing the month and navigation links. These style properties are supported in browsers that can use cascading style sheets; a reduced set of appearance styles is supported for earlier browsers.

Customizing the rendering of individual days

As the control renders individual days, it raises an event that you can handle to modify the stream being rendered to the browser. This is useful not just for changing the appearance of days, but for including custom content on each day as well. For details, see [How to: Customize Individual Days in a Calendar Web Server Control](#).

[Capturing User Interaction with the Calendar Web Server Control](#)

The Calendar control raises the `SelectionChanged` event when the user selects an individual date or range of dates and the `VisibleMonthChanged` event when the user navigates to a new month. By creating methods for these events, you can determine what day or dates the user has selected and respond appropriately. One response may be to customize the display of that date.

[Displaying Information from a Database in the Calendar Control](#)

A common scenario is to display information from a database in the calendar. For example, an events calendar is often based out of information that is stored in a database.

The Calendar control does not directly support data binding — that is, you do not bind the calendar as a whole to a data source. Instead, you create a method for the control's `DayRender` event, which is raised as each day in the current calendar month is being rendered. In the method for this event, you can extract information from a data source and add it to the stream being sent to the browser. For details, see [How to: Display Selected Dates from a Database in the Calendar Control](#).

Accessibility To make the Calendar control more accessible to users of assistive devices, the control supports a property named `UseAccessibleHeader`. When this property is set to true (which is the default setting),

the column headings containing the names of days are rendered using HTML th elements. AdRotator Web Server Control Overview

The AdRotator Web server control provides a convenient way to display advertisements (ads) on your ASP.NET Web pages. The control displays a graphic image that you provide — a .gif file or similar image. When users click the ad, they are redirected to a target URL that you have specified. The control automatically reads advertisement information, such as the graphic file name and the target URL, from a list of ads that you provide using a data source, which is usually an XML file or database table.

The AdRotator control selects ads randomly, changing the displayed ad each time the page is refreshed. Ads can be weighted to control the priority level of banners, making it possible to have certain ads display more often than others. You can also write custom logic that cycles through the ads.

Ad information can come from a variety of sources, such as the following:

- An XML file You can store ad information in an XML file that contains references to ad banners and their associated properties.
- Any data source control, such as the SqlDataSource or ObjectDataSource controls For example, you can store ad information in a database, and can use a SqlDataSource control to retrieve ad information, and then bind the AdRotator control to the data source control.
- Custom logic You can create a handler for the AdCreated event and select an ad in the event.

XML File Format for Ad Files

One method of storing ad-banner image locations, URLs for redirection, and associated properties is to put the information in an XML file. By using the XML file format, you can create and maintain a list of advertisements without having to change the code in your application whenever a change is made to an advertisement. For details, see How to: Display Ads From an XML File Using the AdRotator Web Server Control. Database Schema for Ad Files

Instead of creating an XML file for ad information, you can store ad information in a database table. The table requires a specific schema that the AdRotator control can read. For details, see How to: Display Ads From a Database Using the AdRotator Web Server Control.

```
<Advertisements>
  <Ad>
    <ImageUrl>/Path/Banner.gif</ImageUrl>
    <NavigateUrl>http://aspnet101.com</NavigateUrl>
    <AlternateText> The best ASP.Net web site in the world! </AlternateText>
    <Impressions>3</Impressions>
  </Ad></Advertisements>
  LATIHAN Membuat 1 halaman web yang digunakan untuk menginput biodata. Karena belum masuk ke materi database, maka input tadi cukup ditampilkan dulu pada halaman yang sama. Desain input nya sebagai berikut : dan desain untuk menampilkan datanya sebagai berikut :
  Jawab : Source pada codebehind-nya : Saat klik pada button simpan
  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Panel1.Visible = False
    Panel2.Visible = True
    Dim st1 As String
    As String
    st1 = "Selamat pagi " & TextBox1.Text & _
    " , anda lahir tanggal " & Calendar1.SelectedDate
    st
    suka "
    Dim li As ListItem
    For Each li In CheckBoxList1.Items
      If li.Selected Then
        If st2 = " , anda suk
        st2 & li.Text
      Else
        st2 = st2 & " , " & li.Text
      End If
    End If
    Next
    If st2 = " , anda
    apa-apa"
    st1 = st1 & st2
    Label2.Text = st1
    Image1.ImageUrl = RadioButtonList1.SelectedItem.Value
  End
```