NET Framework 4 - ASP.NET

**Control State vs. View State Example**

This example shows how to create a custom control named IndexButton that uses control state to maintain critical state information across page requests. Control state, introduced in ASP.NET version 2.0, is similar to view state but functionally independent of view state. A page developer can disable view state for the page or for an individual control for performance. However, control state cannot be disabled. Control state is designed for storing a control's essential data (such as a pager control's page number) that must be available on postback to enable the control to function even when view state has been disabled. By default, the ASP.NET page framework stores control state in the page in the same hidden element in which it stores view state. Even if view state is disabled, or when state is managed using [Session](http://msdn.microsoft.com/en-us/library/system.web.ui.page.session.aspx), control state travels to the client and back to the server in the page. On postback, ASP.NET deserializes the contents of the hidden element and loads control state into each control that is registered for control state.

|  |
| --- |
| **Description: NoteNote** |
| Use control state only for small amounts of critical data that are essential for the control across postbacks. Do not use control state as an alternative to view state. |

The example illustrates a custom control that saves state in both control state and view state. In the example, the IndexButton control derives from the [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) class and defines an Index property that it saves in control state. For comparison, IndexButton also defines an IndexInViewState property that it stores in the [ViewState](http://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstate.aspx) dictionary. To see the difference between control state and view state, use the IndexButton control as demonstrated in the .aspx page listed in the "Test Page for the IndexButton Control" section later in this topic.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifCode Listing for the IndexButton Control

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl00_code');" \o "Copy Code)

' IndexButton.vb

Option Strict On

Imports System

Imports System.ComponentModel

Imports System.Security.Permissions

Imports System.Web

Imports System.Web.UI

Imports System.Web.UI.WebControls

Namespace Samples.AspNet.VB.Controls

< \_

AspNetHostingPermission(SecurityAction.Demand, \_

Level:=AspNetHostingPermissionLevel.Minimal), \_

AspNetHostingPermission(SecurityAction.InheritanceDemand, \_

Level:=AspNetHostingPermissionLevel.Minimal), \_

ToolboxData("<{0}:IndexButton runat=""server""> </{0}:IndexButton>") \_

> \_

Public Class IndexButton

Inherits Button

Private indexValue As Integer

< \_

Bindable(True), \_

Category("Behavior"), \_

DefaultValue(0), \_

Description("The index stored in control state.") \_

> \_

Public Property Index() As Integer

Get

Return indexValue

End Get

Set(ByVal value As Integer)

indexValue = value

End Set

End Property

< \_

Bindable(True), \_

Category("Behavior"), \_

DefaultValue(0), \_

Description("The index stored in view state.") \_

> \_

Public Property IndexInViewState() As Integer

Get

Dim obj As Object = ViewState("IndexInViewState")

If obj Is Nothing Then obj = 0

Return CInt(obj)

End Get

Set(ByVal value As Integer)

ViewState("IndexInViewState") = value

End Set

End Property

Protected Overrides Sub OnInit(ByVal e As EventArgs)

MyBase.OnInit(e)

Page.RegisterRequiresControlState(Me)

End Sub

Protected Overrides Function SaveControlState() As Object

' Invoke the base class's method and

' get the contribution to control state

' from the base class.

' If the indexValue field is not zero

' and the base class's control state is not null,

' use Pair as a convenient data structure

' to efficiently save

' (and restore in LoadControlState)

' the two-part control state

' and restore it in LoadControlState.

Dim obj As Object = MyBase.SaveControlState()

If indexValue <> 0 Then

If obj IsNot Nothing Then

Return New Pair(obj, indexValue)

Else

Return indexValue

End If

Else

Return obj

End If

End Function

Protected Overrides Sub LoadControlState(ByVal state As Object)

If (state IsNot Nothing) Then

Dim p As Pair = TryCast(state, Pair)

If p IsNot Nothing Then

MyBase.LoadControlState(p.First)

indexValue = CInt(p.Second)

Else

If (TypeOf (state) Is Integer) Then

indexValue = CInt(state)

Else

MyBase.LoadControlState(state)

End If

End If

End If

End Sub

End Class

End Namespace

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl01_code');" \o "Copy Code)

// IndexButton.cs

using System;

using System.ComponentModel;

using System.Security.Permissions;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Samples.AspNet.CS.Controls

{

[

AspNetHostingPermission(SecurityAction.Demand,

Level = AspNetHostingPermissionLevel.Minimal),

AspNetHostingPermission(SecurityAction.InheritanceDemand,

Level=AspNetHostingPermissionLevel.Minimal),

ToolboxData("<{0}:IndexButton runat=\"server\"> </{0}:IndexButton>")

]

public class IndexButton : Button

{

private int indexValue;

[

Bindable(true),

Category("Behavior"),

DefaultValue(0),

Description("The index stored in control state.")

]

public int Index

{

get

{

return indexValue;

}

set

{

indexValue = value;

}

}

[

Bindable(true),

Category("Behavior"),

DefaultValue(0),

Description("The index stored in view state.")

]

public int IndexInViewState

{

get

{

object obj = ViewState["IndexInViewState"];

return (obj == null) ? 0 : (int)obj;

}

set

{

ViewState["IndexInViewState"] = value;

}

}

protected override void OnInit(EventArgs e)

{

base.OnInit(e);

Page.RegisterRequiresControlState(this);

}

protected override object SaveControlState()

{

// Invoke the base class's method and

// get the contribution to control state

// from the base class.

// If the indexValue field is not zero

// and the base class's control state is not null,

// use Pair as a convenient data structure

// to efficiently save

// (and restore in LoadControlState)

// the two-part control state

// and restore it in LoadControlState.

object obj = base.SaveControlState();

if (indexValue != 0)

{

if (obj != null)

{

return new Pair(obj, indexValue);

}

else

{

return (indexValue);

}

}

else

{

return obj;

}

}

protected override void LoadControlState(object state)

{

if (state != null)

{

Pair p = state as Pair;

if (p != null)

{

base.LoadControlState(p.First);

indexValue = (int)p.Second;

}

else

{

if (state is int)

{

indexValue = (int)state;

}

else

{

base.LoadControlState(state);

}

}

}

}

}

}

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifCode Discussion

The implementation of the IndexButton control illustrates the three tasks that you must perform to enable a control to participate in control state:

* Override the [OnInit](http://msdn.microsoft.com/en-us/library/system.web.ui.control.oninit.aspx) method and invoke the [RegisterRequiresControlState](http://msdn.microsoft.com/en-us/library/system.web.ui.page.registerrequirescontrolstate.aspx) method to register with the page for participation in control state. This must be done with each request.
* Override the [SaveControlState](http://msdn.microsoft.com/en-us/library/system.web.ui.control.savecontrolstate.aspx) method to save data in control state.
* Override the [LoadControlState](http://msdn.microsoft.com/en-us/library/system.web.ui.control.loadcontrolstate.aspx) method to load data from control state. This method calls the base class method and gets the base class's contribution to control state. If the indexValue field is not zero and the base class's control state is not null, the [Pair](http://msdn.microsoft.com/en-us/library/system.web.ui.pair.aspx) class is used as a convenient data structure to save and restore the two-part control state.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifTest Page for the IndexButton Control

The following example illustrates a page that disables view state by setting the [EnableViewState](http://msdn.microsoft.com/en-us/library/system.web.ui.page.enableviewstate.aspx) attribute to **false** in the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive. The page uses the IndexButton control and adds 1 to the values of the Index and IndexInViewState properties of the control in the **Page\_Load** event handler. The labels in the page display the values of the Index and IndexInViewState properties.

Because the Index property is stored in control state, which cannot be disabled, the Index property maintains its value on postback and increases by one each time the page is posted back to the server. In contrast, because the IndexInViewState property is stored in view state, which is disabled for the page, the IndexInViewState property always has its default value of zero.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_ctl00_ctl02_code');" \o "Copy Code)

<%@ Register TagPrefix="aspSample"

Namespace="Samples.AspNet.vB.Controls" Assembly="Samples.AspNet.VB" %>

<%@ Page Language="VB" Trace="true" EnableViewState="false" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

Sub Page\_Load(ByVal sender As Object, ByVal e As EventArgs)

Label1.Text = IndexButton1.Index.ToString()

Label2.Text = IndexButton1.IndexInViewState.ToString()

IndexButton1.Index += 1

IndexButton1.IndexInViewState += 1

End Sub

</script>

<html >

<head id="Head1" runat="server">

<title>IndexButton test page</title>

</head>

<body>

<form id="form1" runat="server">

Click the button:

<aspSample:IndexButton Text="IndexButton"

ID="IndexButton1" runat="server"/>

<br />

<br />

The value of the Index property of IndexButton is:<br />

<asp:Label ID="Label1" Runat="server" Text="Label">

</asp:Label>

<br />

<br />

The value of the IndexInViewState property of IndexButton is:

<br />

<asp:Label ID="Label2" Runat="server" Text="Label">

</asp:Label>

<br />

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_ctl00_ctl03_code');" \o "Copy Code)

<%@ Register TagPrefix="aspSample"

Namespace="Samples.AspNet.CS.Controls" Assembly="Samples.AspNet.CS" %>

<%@ Page Language="C#" Trace="true" EnableViewState="false" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

void Page\_Load(object sender, EventArgs e)

{

Label1.Text = (IndexButton1.Index++).ToString();

Label2.Text = (IndexButton1.IndexInViewState++).ToString();

}

</script>

<html >

<head id="Head1" runat="server">

<title>IndexButton test page</title>

</head>

<body>

<form id="form1" runat="server">

Click the button:

<aspSample:IndexButton Text="IndexButton"

ID="IndexButton1" runat="server"/>

<br />

<br />

The value of the Index property of IndexButton is:<br />

<asp:Label ID="Label1" Runat="server" Text="Label">

</asp:Label>

<br />

<br />

The value of the IndexInViewState property of IndexButton is:

<br />

<asp:Label ID="Label2" Runat="server" Text="Label">

</asp:Label>

<br />

</form>

</body>

</html>