

Implement an ASP.NET Back Control

Give Web pages a server-side control that redirects the user's browser to the referring page.

by Juval Löwy

Technology Toolbox

- VB.NET
- C#
- SQL Server 2000
- ASP.NET
- XML
- VB6

Go Online!

Use these Locator+ codes at www.visualstudiomagazine.com to go directly to these related resources.

Download

VS0307QA Download the code for this article, which includes the BackLink ASP.NET user control.

Discuss

VS0307QA_D Discuss this article in the ASP.NET forum.

Read More

VS0307QA_T Read this article online.

VS0305QA_T Q&A, "Publish Events Defensively," by Juval Löwy

VS0304AN_T ASP.NET, "Maintain State With Dynamic Controls," by Garry McGlennon

VSEP020722AN_T "Communicate Between User Controls" by Jonathan Goodyear

Q: Implement an ASP.NET Back Control

I want to add a link to my ASP.NET pages that goes back to the page that referred them. How do I use a server-side control to do this? I need to have control over the referred page, and using a browser history isn't an option.

A:

You have two ways to implement a "back" link on a Web page. The first is to use a client-side

script to access the browser history of visited pages and insert a redirection to the previous page:

```
<a href="javascript:history.back()">Back</a>
```

However, this technique has several disadvantages. The application has no control over where the user is redirected. You often want to keep the users inside the application, and you don't want

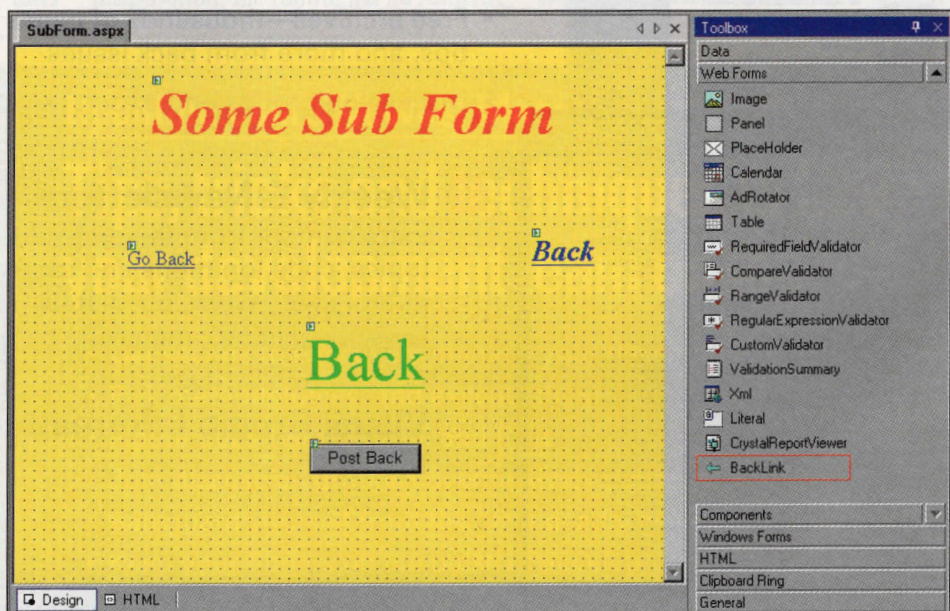


Figure 1 Use the BackLink User Control. The BackLink control presents the same design-time properties as the LinkButton control. The BackLink control redirects the user's browser to the previous page at run time.

C# • Implement the BackLink User Control

```
[ToolboxData("<0>:BackLink
runat=server></0>:BackLink>")]
[ToolboxBitmap(typeof(BackLink), "BackLink.bmp")]
public class BackLink : LinkButton
{
    public BackLink()
    {
        Text = "Back";
        ToolTip =
            "Click to go to the previous page";
    }

    protected override void OnClick(EventArgs e)
    {
        Uri backURL = (Uri)Page.Session[
            "Referring URL"];
        Page.Session["Referring URL"] = null;
        if(backURL != null)
        {
            Page.Response.Redirect(
                backURL.AbsoluteUri);
        }
    }

    protected override void OnLoad(EventArgs e)
    {
        Uri backURL = Page.Request.UrlReferrer;

        if(backURL == null) //No referrer
            information
        {
            Enabled = false;
            return;
        }
        if(backURL.AbsolutePath !=
            Page.Request.Url.AbsolutePath)
        {
            Page.Session["Referring URL"] =
                backURL;
            Enabled = true;
            return;
        }
        else
        {
            object obj = Page.Session[
                "Referring URL"];
            if(obj != null)
            {
                Enabled = true;
            }
        }
        base.OnLoad(e);
    }
}
```

Listing 1 The BackLink control derives from LinkButton, which does the actual rendering. BackLink caches the referrer page at load time in a session variable, and it redirects the user to it by overriding OnClick().

them to wander off to other pages. This solution works only if the browser supports client-side script. Its biggest disadvantage is that it's inconsistent with the ASP.NET programming model. One of ASP.NET's greatest benefits is that—unlike classic ASP—it doesn't require you to rely on client-side script. You simply use server-side controls that you write in managed code, and ASP.NET does the rest. When you use client-side script, you disconnect the back link from the rest of your application, which executes on the server and uses server-side controls. This leaves you no easy way to enable or disable the back link or control the redirection, based on server-side event processing.

The second solution is to use an ASP.NET custom user control. The source files accompanying this article contain the BackLink ASP.NET user control in the WebControlsEx class-library assembly (download the source code from the *VSM* Web site; see the Go Online box for details). To use it, right-click on the Web Forms toolbox, and select Add/Remove Items... from the popup context menu. Click on the Browse... button on the .NET Frameworks Components tab. Select the WebControlsEx assembly and click on Open. This adds the BackLink user control to the toolbox. You can simply drag and drop it onto your forms to add a link-button-like control to the form, with the text set to "Back." The control presents the same design-time properties as the standard link button: You can change the text, font size, color, boldness, annotation, and other properties, such as link style (see Figure 1). The back-link control redirects the browser to the previous page, if one is available, when the user clicks on the button at run time.

Implementing the BackLink user control is more challenging than meets the eye. You must start with a DLL class library in order to build a user Web control. You can provide a user control either by deriving from a class called WebControl and doing the rendering

yourself, or by deriving from an existing control and specializing its behavior. Deriving from a LinkButton is the better option by far for a back-link control, because the LinkButton control provides most of the difficult-to-implement functionality.

Derive a class called BackLink from LinkButton and set the Text property in its constructor to Back, to provide some meaningful default value:

```
public class BackLink : LinkButton
{
    public BackLink()
    {
        Text = "Back";
    }
}
```

The link button derivation does the actual work of rendering the text to appropriate HTML. It also exposes all the LinkButton's properties—such as default property, font setting properties, and events—from the BackLink to the visual designer.

The LinkButton has a protected virtual method called OnClick(), which .NET calls after a post back to the server, letting the link button publish the click event to the event subscribers. The LinkButton must override OnClick() and redirect the user to the referring page.

The important question is how you can get the page that referred the current page at the server side, in the scope of the BackLink control. Fortunately, ASP.NET enables this: The HttpRequest object exposes a public property of type Url called UrlReferrer, which is the URL of the page that referred the current page. The LinkButton user control can obtain a reference to the page that hosts it through the Control

class's Page property. Control is the base class of WebControl, which is the base class of LinkButton. The control can also use the Page property to access its HttpResponse object (the Response property) to redirect the user to the referring page.

However, the back link doesn't work if you use this code to implement the BackLink control:

```
public class BackLink : LinkButton
{
    public BackLink()
    {
        Text = "Back";
    }
    protected override void
        OnClick(EventArgs e)
    {
        Uri backURL =
            Page.Request.UrlReferrer;
        Page.Response.Redirect(
            backURL.AbsoluteUri);
    }
}
```

The reason lies in the way ASP.NET keeps track of referring pages. For example, consider the page Home.aspx, which redirects the user to the page SubForm.aspx. If the SubForm.aspx page has a BackLink

control on it and the user clicks on the link, this triggers a post back to the server. The value of the UrlReferrer property on the server is "SubForm.aspx"—not "Home.aspx"—because SubForm.aspx refers to itself in the case of a post back. The solution is to cache the referring page in a session variable during the control's load time and redirect to the referring page in OnClick().

Now you have more issues to deal with: ASP.NET isn't always capable of providing pages with referring information, in which case ASP.NET sets the value of UrlReferrer to null. You also want to provide a "smart" back link that always redirects to the logical previous page. In other words, if a page that contains a BackLink is reloading itself (after handling some control's click event), the back link should be smart enough to detect this and redirect to the "real" previous page, not to itself. Note that this functionality is impossible if you rely on client-side script, because the back history variable will yield the same page in the case of a post back.

BackLink overrides its base class's OnLoad() method (see Listing 1). OnLoad() is called when the page containing the BackLink control is loaded. OnLoad() checks first whether the referring page information is available. If UrlReferrer is null, then OnLoad() disables the back link. If referring information is available, you need to verify that the referring page isn't the current page. You can do this easily by comparing the referring URL with the requested URL:

```
if(backURL.AbsolutePath !=
    Page.Request.Url.AbsolutePath)
```

BackLink stores the referring URL in a session variable if the two URLs are different:

```
Page.Session["Referring URL"] =
    backURL;
```

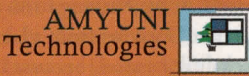
Then it enables the control. If the addresses are the same, BackLink verifies that a referring URL is cached already, before enabling the control. In OnClick(), you must access the session variable, and if the session variable exists, redirect the user to the referring page.

You can also provide the visual designer with information about the user control. The ToolboxData attribute instructs VS.NET what to insert in the ASPX file when you drop the control on a Web form. The ToolboxBitmap contains a reference to the control icon, in the form of an embedded resource. As Figure 1 shows, this icon appears in the toolbox once you add the control to it. **VSM**

Juval Löwy is a software architect and the principal of IDesign, a consulting and training company focused on .NET design and .NET migration. Juval is Microsoft's regional director for the Silicon Valley, working with Microsoft on helping the industry adopt .NET. His latest book is *Programming .NET Components* (O'Reilly & Associates). Juval speaks frequently at software-development conferences. Contact him at www.idesign.net.

Additional Resources

Programming .NET Components by Juval Löwy [O'Reilly & Associates, 2003, ISBN: 0596003471]




**We personalize
your needs!**

NEW

Document Converter Suite PDF-RTF-HTML-Excel-JPEG

Amyuni's new Document Converter gives you more document conversion power in a more streamlined product. Our flagship PDFConverter technology has been applied to RTF, HTML, Excel® and JPEG formats and integrated them in one global Amyuni printer driver.

www.amyuni.com 

info: sales@amyuni.com

Americas

Toll Free: 1-866-926-9864

Support: (514) 868-9227

Europe

Sales: (+33) 1 30 61 07 97

Support: (+33) 1 30 61 07 98

All trademarks are property of their respective owners. © 2002 AMYUNI Technologies All rights reserved.