

Insider Extra 3

Picture Sizer

This Insider Extra is a Microsoft ASP.NET page that reads a picture file on a Web server and delivers a different-sized version to the browser. This is useful for creating thumbnail pictures on the fly, or any time you want pictures of different physical sizes to appear uniformly sized in the browser.

Unlike some other methods, the Picture Sizer physically resizes the picture in memory before sending it to the Web browser. This consumes much less bandwidth than sending the browser a full sized picture for display in a small space. The fact that it resizes pictures on the fly means that thumbnails and full-sized pictures never get out of sync, and that you can display thumbnails of pictures even if Web visitors just uploaded them. It's also useful for displaying a set of pictures at the same size, even if the originals are different sizes.

Because this is an ASP.NET page, it runs only on Microsoft Windows 2000, Windows XP Professional, or Windows Server 2003 computers running Microsoft's Web server, IIS. In addition, the Microsoft .NET Framework must be installed on the server. The .NET Framework comes preinstalled with Windows Server 2003; for the other systems, browse www.microsoft.com/net/, and follow the download links.

Tip If you're not sure whether your Web server can run ASP.NET pages, try running the ASP.NET Self-Diagnosis Web Page in Insider Extra 2.

To install the Picture Sizer, first install the Insider Extra Web site from the companion CD. The Sample Files setup program installs this site at [My Documents]\Microsoft Press\FrontPage 2003 Inside Out\fp1\extras. Then copy the picsizer/picsizer.aspx file into your Web site.

Using the Picture Sizer

The Picture Sizer is neither a Microsoft Office FrontPage 2003 component nor a standard picture file. As a result, the Design view editor doesn't provide a great deal of help adding it to your page. Here, however, is one approach:

- 1 Open the Web page that you want to display the resized picture.
- 2 Set the insertion point where you want the picture to appear.
- 3 Choose Quick Tag Editor from the Edit menu.

Microsoft Office FrontPage 2003 Inside Out

- 4 When the Quick Tag Editor dialog box appears, make sure that Insert HTML is selected in the drop-down list. Then, in the text box and between the angle brackets, enter the following text:

```
img src=path1/picsizer.aspx?url=path2/clown.jpg
```

- *path1* is the path, if any, from the current Web page to the picsizer.aspx page.
- *path2* is the path, if any, from the picsizer.aspx page to the picture file.

Here's an example:



- 5 The code you inserted so far displays the picture at full size. To resize the picture, append the following to the picture file name:
- An ampersand (&).
 - The keyword *size*, *width*, or *height*. Table IE3-1 describes these keywords, as well as the *url* parameter you already entered.
 - An equal sign (=).
 - A size in pixels.

Table IE3-1. Picture Sizer Query String Parameters

Parameter	Description
<i>url</i>	The URL of the picture to resize.
<i>size</i>	The height or width of the output picture, whichever is larger.
<i>width</i>	The width of the output picture. Ignored if <i>size=</i> is present.
<i>height</i>	The height of the output picture. Ignored if <i>size=</i> or <i>width=</i> is present.

Here's several examples, this time showing the complete tag:

```
<img src=picsizer.aspx?url=../clown.jpg&size=75>
```

```
<img src=picsizer.aspx?url=images/chart.gif&height=100>
```

```
<img src=picsizer.aspx?url=gallery/Hawaii.jpg&width=90>
```

The different path and file names reflect different picture locations relative to the picsizer.aspx page. The first tag sizes the picture's largest dimension to 75 pixels. The second sized the picture to a height of 100 pixels, and the third to a width of 90 pixels. After sizing one dimension, picsizer.aspx sizes the other dimension proportionally.

- 6 Click the check mark button to save your work, or click the X button to cancel.

If your development Web site resides on a server that can run the picsizer.aspx page, the resized picture will appear in Design view. Otherwise, a broken picture icon will appear. In the latter case, to view the finished results, you'll need to publish the Web page to a server that can run ASP.NET pages and browse the page using an *http://* URL.

Picture Sizer

Tip Although the FrontPage Insert Picture command won't accept `picsizer.aspx` as the URL of a picture, other features such as dragging, Cut, Copy, Paste, and the Picture Properties dialog box work fine.

If you specify a corrupt picture file or a picture that the `picsizer.aspx` page can't find, a gray rectangle will appear in place of the picture you expect.

Figure IE3-1 shows a Web page that displays the same picture three times, each at a different size. On disk, the picture is 216 pixels wide and 319 pixels tall.

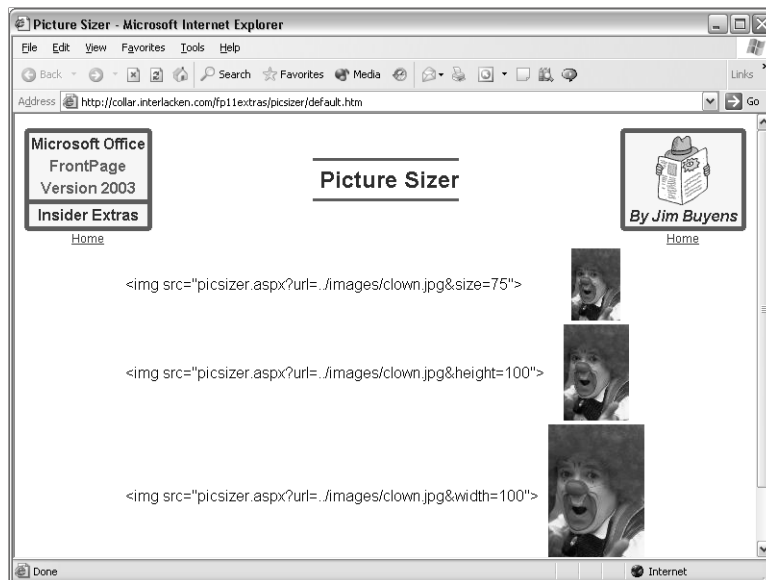


Figure IE3-1. The Picture Sizer Insider Extra reduces the size of pictures before sending them to the browser. All three clown pictures are the same size on disk.

For displaying single pictures, the thumbnail facility built into FrontPage will almost certainly be quicker and easier than using the Picture Sizer Insider Extra. The value of the Insider Extra is more evident for pictures subject to frequent updates. The FrontPage thumbnail feature would be impractical for pictures come from a Web cam, for example, or for pictures that visitors upload and wish to select immediately.

The ASP.NET Picture Library in Insider Extra 5 uses the Picture Sizer to generate thumbnails for all the picture files in a given folder. Expanding or reducing the picture library therefore requires adding or removing files—nothing more. There's no need to add or remove thumbnail pictures and full-sized pictures in unison.

Understanding the Picture Sizer Web Page

This section explains how the Picture Sizer Web page actually works. To understand this material, you'll need some existing background in ASP.NET programming. If you lack such background, just use the Picture Sizer as a black box and ignore this section.

The full source code for this Insider Extra resides inside the `picsizer/picsizer.aspx` file. As you continue reading, you should have this file open in Code view or otherwise available for reference.

Although the Picture Sizer is an ASP.NET Web page, its output doesn't consist of HTML. Instead, its output is a stream of binary data in GIF or JPEG format. The `picsizer.aspx` page tells the browser to expect a picture by executing one of these statements just before transmitting the picture information:

```
Response.ContentType = "image/gif"
```

```
Response.ContentType = "image/jpeg"
```

This, of course, begs the question of how the Web page resizes the picture and transmits the results. For starters, it uses the following statements to load the original picture file:

```
Dim bmpInput As Bitmap
bmpInput = Bitmap.FromFile( _
    Server.MapPath(Request.QueryString("url")))
```

The expression `Request.QueryString("url")` retrieves the URL you code after `url=` in the HTML that invokes the `picsizer.aspx` page. The `Server.MapPath` method converts this URL to a physical path, and the `Bitmap.FromFile` method reads the picture from disk and into the server's memory.

If the `Bitmap.FromFile` method can't read the file from disk, it throws an exception. To stop this from blowing up the Web page, the code executes the `Bitmap.FromFile` method inside a `Try...Catch...End Try` block. If an exception occurs, code in the `Catch` block loads the `bmpInput` variable with a gray rectangle that the code creates in memory. Here's how this looks altogether:

```
Try
    bmpInput = Bitmap.FromFile( _
        Server.MapPath(Request.QueryString("url")))
Catch
    bmpInput = New Bitmap(100,50)
    Dim gfxInput As Graphics = Graphics.FromImage(bmpInput)
    gfxInput.FillRectangle( _
        New SolidBrush(Color.FromARGB(255,204,204,204)), _
        0, 0, 100, 50)
    gfxInput.Dispose
End Try
```

Picture Sizer

The first statement after *Catch* creates a new 100-by-50-pixel bitmap. The second creates a *Graphics* object for manipulating this bitmap. The third statement, which spans three lines, draws a light gray color (opacity 255; RGB 204, 204, 204) from coordinates 0, 0 to coordinates 100, 50. The last statement before *End Try* disposes of the *Graphics* object.

To resize the picture, the page first retrieves the size, height, and width values, if any, from *Request.QueryString*. The height and width of the original picture are available in *bmpInput.Height* and *bmpInput.Width*. Knowing the ratio of height to width and one target dimension, the page can easily calculate the other target dimension. It stores the target height and width in two variables, named *lngHgt* and *lngWid*.

If the target size turns out larger than the original size, the page doesn't resize the picture. Instead, it calls a subroutine named *WritePicture*, passing the *bmpInput* picture as an argument. This transmits the picture at original size.

If the target size is smaller than the original size, the code executes this statement:

```
bmpThumb = New Bitmap(bmpInput, lngWid, lngHgt)
```

This creates a new *Bitmap* object named *bmpThumb* based on the existing *Bitmap* object *bmpInput*, but resized to *lngWid* pixels wide and *lngHgt* pixels high. The code then calls the *WritePicture* subroutine, passing the *bmpThumb* object as an argument.

The following code appears within the *WritePicture* subroutine. The argument name of the incoming *Bitmap* object is *abmpPic*.

```
If Lcase(right(Request.QueryString("url"),4)) = ".gif" Then
    Response.ContentType = "image/gif"
    abmpPic.Save(Response.OutputStream, ImageFormat.GIF)
Else
    Response.ContentType = "image/jpeg"
    abmpPic.Save(Response.OutputStream, ImageFormat.JPEG)
End If
```

This code is relatively straightforward. If the original file name ended in *.gif*, the code tells the browser that a GIF file will follow, and then it saves the *Bitmap* object received as *abmpPic* into the *Response.OutputStream* in GIF format. Otherwise, it tells the browser that a JPEG file will follow and then saves *abmpPic* into the *Response.OutputStream* in JPEG format. (The *Response.OutputStream* stream is the flow of bits that the Web server sends to the browser.)

After the *WritePicture* subroutine completes its work, the code takes care to execute the following statement:

```
Response.End
```

This immediately terminates all output from the Web page to the browser. For example, it ensures that the Web server won't append any tags such as `<html>` or `<script>` to the end of the picture data.