

## Insider Extra 4

# Converting Text to GIF Format

Like the Picture Sizer in Insider Extra 3, this Insider Extra is an ASP.NET page that sends pictures directly to the browser. Rather than sending a resized picture, however, the Text-To-GIF Converter sends a pictorial representation of text.

Experienced Web designers often send small amounts of text to the browser not as ordinary text but as picture files. The usual reason is predictability. When the browser displays text, the visitor's computer might not have the font that the designer specified. If it doesn't, the text will appear in a different font (and take up a different amount of space) than the designer intended. Also, the visitor might have chosen to display text at larger than or smaller than normal size. Again, this produces different results than the designer intended.

Pictures, on the other hand, are always displayed the same. A pixel is a pixel. Of course, displaying text as pictures of text has disadvantages as well:

- Pictures use much more bandwidth than text. Expressed as text and formatting commands, a typical heading requires 50 to 100 bytes. As a picture, the same heading will probably require 1000 to 2000 bytes..
- Creating small picture files can be a hassle. Typically, you must open a separate editor, create the picture, save it, add it to your Web site, and then add it to your Web page.
- There's no good way to display variable text, such as the date or time, as a picture.

There's no sure cure for the bandwidth problem, other than minimizing the use of pictures to display text. Sometimes, however, picture files that display text are truly the best solution to a problem. In such cases, the Text-To-GIF Converter makes creating picture files remarkably easy.

Like all ASP.NET pages, this Insider Extra runs only on Microsoft Windows 2000, Windows XP Professional, or Windows Server 2003 computers running Microsoft IIS. In addition, the Microsoft .NET Framework must be installed on the server. If you have Windows Server 2003, the .NET Framework is an integral component. To obtain the .NET Framework for Windows 2000 and Windows XP Professional, browse [www.microsoft.com/net/](http://www.microsoft.com/net/), and follow the download links.

To install the Text-To-GIF Converter, 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\fp11extras. Then, copy the `text2gif/text2gif.aspx` file into your Web site.

## Using the Text-To-GIF Page

To use the `text2gif.aspx` page, you specify it as the source of an HTML `<img>` tag. Here's an example:

```

```

The query string argument `text=Hello` specifies the text to display: *Hello*, in this case. If your text contains any characters that aren't valid in a URL, code them as a percent sign (%) followed by a hex code. Table IE4-1 lists some common examples.

**Table IE4-1. URL Encoding for Common Special Characters**

Character	Name	URL equivalent
	Space	%20
"	Quotation mark	%22
#	Pound sign	%23
%	Percent sign	%25
&	Ampersand	%26
'	Apostrophe	%27
+	Plus sign	%2b
=	Equal sign	%3d
?	Question mark	%3f

Remember as well that a plus sign in a URL indicates a space. All the following URLs, for example, would display Insider Extras:

```
text2gif.aspx?text=Insider Extras
text2gif.aspx?text=Insider+Extras
text2gif.aspx?text=Insider%20Extras
```

By default, the `text2gif.aspx` page displays text as black, 24 point, Times New Roman on a white background. To override this format, append any of the query string arguments listed in Table IE4-2.

**Table IE4-2. URL Encoding for Common Special Characters**

Argument name	Description
text	Specifies the text to display. Be sure to URL-encode this string (for example, use %20 or + rather than a space).
font	The name of the font to use. This can be any font installed on the Web server. Fonts installed on the client are irrelevant.
size	The height of the text in pixels. Windows adds a margin to this value.
bold	Displays boldface characters. You must assign a value, but the value doesn't matter. <code>bold=y</code> , <code>bold=n</code> , and <code>bold=xxx</code> all request boldface.

## Converting Text to GIF Format

**Table IE4-2. URL Encoding for Common Special Characters**

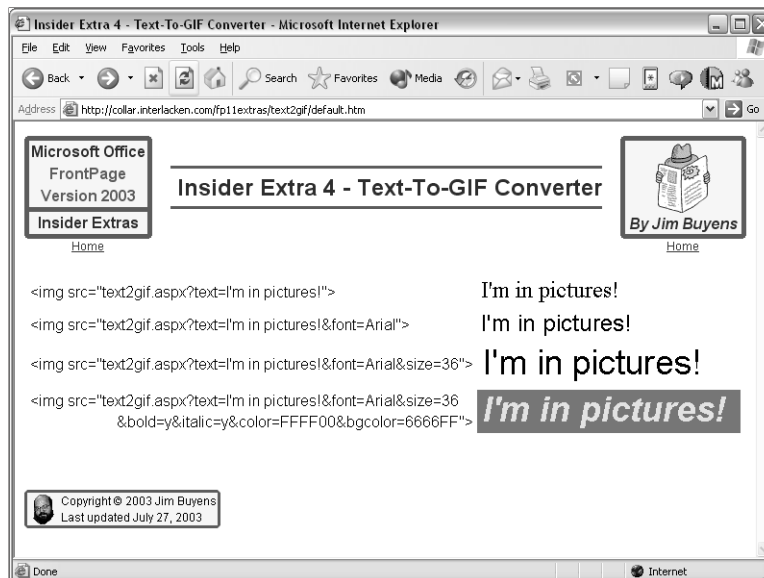
Argument name	Description
italic	Displays italic characters. You must assign a value.
strikeout	Displays a strikethrough line through the text. You must assign a value.
underline	Underlines the text. You must assign a value.
color	Specifies the text color.
bgcolor	Specifies the background color.

For color values, specify 3, 4, 6, or 8 hex digits. Table IE4-3 shows how the text2gif.aspx page will interpret these digits. For opacity, values of 0 or 00 mean transparent; F or FF mean opaque.

**Table IE4-3. Color Value Interpretations**

Digits	Opacity	Red	Green	Blue
3		0-F	0-F	0-F
4	0-F	0-F	0-F	0-F
6		00-FF	00-FF	00-FF
8	00-FF	00-FF	00-FF	00-FF

Figure IE4-1 shows some typical URLs and the corresponding results.



**Figure IE4-1.** The Text-To-GIF Converter transmits pictures containing text from the HTML.

## Microsoft Office FrontPage 2003 Inside Out

To display the resulting picture in a Web page, you must, of course, specify its URL in the `src=` attribute of an `<img>` tag. Unfortunately, none of the usual Insert Picture commands in Microsoft FrontPage will insert an `.aspx` file; instead, they insist on the usual picture file extensions such as `.gif` and `.jpg`. The easiest way to use the Text-To-GIF Page is therefore to create a tag with the Quick Tag Editor. Here's a typical procedure:

- 1 Open the Web page that you want to display the text, and set the insertion point where you want the text to appear.
- 2 Choose Quick Tag Editor from the Edit menu.
- 3 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=path/text2gif.aspx?url=text
```

In this statement:

- `path` is the path, if any, from the current Web page to the `text2gif.aspx` page.
  - `text` is the text you want to display.
- 4 For each text attribute you want to apply, append the following to the picture file name:
    - An ampersand (&)
    - An argument name from Table IE4-2
    - An equal sign (=)
    - An argument value as described in Table IE4-2
  - 5 Click the check mark button to save your work, or the X button to cancel.

**Tip** Although the FrontPage Insert Picture command won't accept `text2gif.aspx` as the URL of a picture, other features such as dragging, Cut, Copy, Paste, and the Picture Properties dialog box work fine.

If your development Web site resides on a server that can run the `text2gif.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 then browse the page using an `http://` URL.

To display variable text, you'll need to write script code that adds the text to the `text2gif.aspx` URL. The following tag, for example, displays the current date and time as a graphic:

```

```

Because you can't use normal FrontPage dialog boxes for adding or configuring pictures that the Text-To-GIF Page creates, crafting your own pictures in a graphics editor might sometimes be easier. Nevertheless, creating text pictures on the fly can be a real problem-solver in some situations, especially for problems related to variable text.

## Converting Text to GIF Format

# Understanding the Text-To-GIF Converter

This section explains how the Text-To-GIF Converter actually works. In all probability, this will make sense only if you have some background in ASP.NET programming. Otherwise, just use the Text-To-GIF Converter as the previous section described, and forget about the internals.

The full source code for this page resides in the `text2gif/text2gif.aspx` file. As you continue reading, you should have this file open in Code view or otherwise available for reference.

Most processing for the `text2gif.aspx` page occurs in the `Page_Load` subroutine, which runs automatically each time ASP.NET runs the page. After declaring and initializing all the variables it needs, this subroutine iterates through each member of the `Request.QueryString` collection. (This is the collection of name-value pairs that you specify after the question mark in a URL.) Code within this loop detects each valid name-value pair and overrides the default formatting properties accordingly.

When this process is complete, the code verifies that it received some text to display. If it didn't, it defaults to displaying one space.

Next the following statement creates a `Font` object that incorporates the Web designer's specifications:

```
fntOut = New Font(strFontFamily, intFontSize, fstOut, _
    GraphicsUnit.Pixel)
```

In this statement:

- The `strFontFamily` variable is a `String` variable that contains the font name.
- The `intFontSize` variable is an `Integer` variable that contains the font size.
- The `fstOut` variable is a `.NET FontStyle` object. The loop that inspected the `Request.QueryString` collection saved any *bold*, *italic*, *strikeout*, and *underline* settings as bits in this object.
- The `GraphicsUnit.Pixel` argument specifies that graphic measurements will be in pixels.

Next the code must determine how large the output picture must be. The `.NET Framework` has a `MeasureString` method that does this, but it operates only in the context of an existing picture. The code therefore creates a 1-by-1-pixel `Bitmap` object, creates a `Graphics` object based on the bitmap, and calls the `MeasureString` method of the `Graphics` object to get the size of the string. Here's the code:

```
imgOut = New Bitmap(1, 1, PixelFormat.Format24bpprgb)
gfxOut = Graphics.FromImage(imgOut)
szfOut = gfxOut.MeasureString(strOut, fntOut)
```

Notice that the `MeasureString` method accepts two arguments: `strOut`, which contains the text that the designer specified, and `fntOut`, the `Font` object that contains the designer's specifications for font name, font size, boldface, italics, strikethrough, and underlining. The result is a `SizeF` object named `szfOut`.

## Microsoft Office FrontPage 2003 Inside Out

Armed with the information in the *szfOut* object, the code can now create a *Bitmap* object of exactly the right size. Here's the code:

```
imgOut = New Bitmap(Cint(szfOut.Width),Cint(szfOut.Height), _
    PixelFormat.Format24bpprgb)
```

Next the code creates a *Graphics* object to manipulate the new bitmap, and clears the bitmap picture to the background color that the designer specified. This requires the following statements:

```
gfxOut = Graphics.FromImage(imgOut)
gfxOut.Clear(cclrBackG)
```

Drawing the text onto the cleared bitmap requires a brush that uses the color the designer specified. The following statement creates this brush:

```
brsOut = New SolidBrush(cclrBrush)
```

The *DrawString* method of the *Graphics* object *gfxOut* can now draw the text onto the empty bitmap. Here's the required statement:

```
gfxOut.DrawString(strOut, fntOut, brsOut, New PointF(0,0))
```

As before, the *strOut* variable contains the text that the designer specified in the URL. The *fntOut* variable points to the *Font* object that contains the designer's typographical specifications. The *brsOut* variable points to the brush that the code just created. The last parameter is a new *PointF* object that specifies coordinates (0,0) as the spot to start drawing the text.

The last task is sending the picture to the visitor. This requires the following statements. The first informs the browser that a GIF picture is forthcoming. The second writes the *Bitmap* object *imgOut*, in GIF format, into the *Response.OutputStream* that the Web server sends to the browser.

```
Response.ContentType="image/gif"
imgOut.Save(Response.OutputStream, imageFormat.gif)
```

The *text2gif.aspx* page contains two additional functions named *GetColor* and *HexToInt*. The *GetColor* function processes any *color* or *bgcolor* settings that the designer specifies. There are no tricks to this code. It simply removes the leading pound sign (#), if any, and then tests the length of the argument. If the argument is 3, 4, 6, or 8 bytes in length, the code breaks the argument into its constituent values, runs each value through the *HexToInt* function, and uses the results to create a .NET *Color* object. The *Color* object is the return value from the function.

The *HexToInt* function receives one argument, a *String* variable that contains a series of characters in the range 0–9 or A–F. To process this string, the function first initializes an *Integer* variable named *intResult* to zero. Then, for each character in the argument string, it multiplies the previous value of *intResult* by 16 and adds the hexadecimal value of the current character. This value is 0–15 for input characters 0–9 and A–F, respectively.