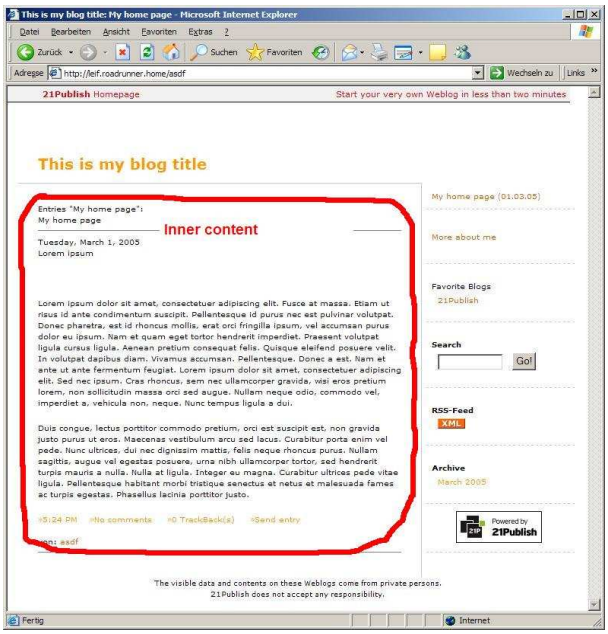
Creating Layouts for 21Publish-based blogportals

How to create weblog and portal templates for 21Publish systems

by Leif M. Koch

1. Structure of 21Publish Layouts

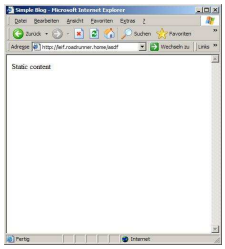
When displaying a weblog, there are various items to be displayed: recent entries, latest comments, categories, blogrolls, and many more. When navigating through the blog, its outer navigational frame usually doesn't change: categories, blogrolls, title etc. stay the same, only the content to be displayed is being exchanged. Thus, we separated the layout of a weblog into two distinct areas: the outer frame, including categories, blogrolls, and all other items that should not change when navigating through the weblog, and the inner content. This inner content differs depending on what is supposed to be displayed: entries within a particular category, a single entry along with its comments, or a profile page (see image below).



The outer frame (mostly referred to as the weblog layout, since this file really defines the appearance of the whole blog) is a standard HTML page. Thus, it contains a header and a body section. If you were using the blog merely to display a static HTML page, you could upload a file like this one (static_blog.htm):

```
<html>
<head>
<title>Simple Blog</title>
</head>

<body>
Static content
</body>
</html>
```



The inner part of the blog layout that displays varying content is an HTML fragment. Thus, it does not have a header definition and no body tags. It merely contains HTML tags formatting the content to be displayed the way you like it.

For example, your inner part could be the following lines (let's call it sample_content.htm):

```
<h1>Sample header</h1>
<p>This is some static HTML content</p>
```

21Publish allows you to create three different content templates for the following sections: entries within a weblog, a single entry with its comments, and a profile page. The system automatically detects which template file to use, depending on the URL you need to access a page.
Now if you're wondering how the system knows where to display our example sample_content.htm within your static_blog.htm file: it doesn't. Our example was so simple it merely displays the static_blog.htm file without generating any dynamic content. In order to display the sample_content.htm file, you need to add the following word somewhere in the layout page: $layoutContent$. Using our static_blog.htm file we might come up with the following (simple_blog.htm):

```
<html>
<head>
<title>Simple Blog</title>
</head>

<body>
Static content <br />

$layoutContent$

</body>
</html>
```

When accessing your blog, simple_blog.htm and simple_content.htm would result in the following output for your web browser:

```
<html>
<head>
<title>Simple Blog</title>
</head>

<body>
Static content <br />

<h1>Sample header</h1>
<p>This is some static HTML content</p>

</body>
</html>
```

As you can see, $layoutContent$ has been replaced with the content of your simple_content.htm file. The next section now explains this template language in more detail.

2. Template language

The 21Publish template language dynamically extends HTML files using loops, conditions and values. This section will explain those structures in detail.
The first section has already used a command: $layoutContent$. As you can see, a $ indicates a command within our template language. First thing that comes to your mind is the question what if you actually want to display a $ and not mark a template command: you need to print $$ in your template file. Consider the following code fragment:

```
<h1>Some sample pricing</h1>
<p>Get this for just $4.99! </p>
```

If you were including this somewhere in your template files, the system would stop parsing your file and directly print the result so far:

```
<h1>Some sample pricing</h1>
<p>Get this for just
```

Thus, you need to tell the system you actually want to display a $:

```
<h1>Some sample pricing</h1>
<p>Get this for just $$4.99! </p>
```

2.1 Variable substitutions

Now let's get back to our commands. Any dynamic values in the system always have the following syntax:

```
$variableName$
```

This tells our system you would like to have the value of some variable with name "variableName" displayed. We call this "substitution", since $variableName$ is substituted with some value. For example, we have a substitution $weblogTitle$ that displays the title of your weblog. Using our original example, you can extend your blog layout to display the title of your blog in your browser's title bar:

```
<html>
<head>
<title>$weblogTitle$</title>
</head>

<body>
Static content <br />

$layoutContent$

</body>
</html>
```

Assuming you've named your blog "My test blog", this is what you get:

```
<html>
```

```
<head>
<title>My test blog</title>
</head>

<body>
Static content <br />

<h1>Sample header</h1>
<p>This is some static HTML content</p>

</body>
</html>
```

You can get substitutions available to you from the sample files or the appendix.
By the way: $layoutContent$ is the only substitution where no a simple value but the content of a whole file is printed.

2.2 Conditions

Next, you can use conditions. A condition is an if-then clause (or if-then-else) allowing you to display some parts only in certain situations. A condition has the following syntax:

```
$[condition$
$condition]$
```

or including the else-section:

```
$[condition$
$]condition[$
$condition]$
```

What happens is the following: if "condition" is met, any code between $[condition$ and $condition]$ is being displayed (or rather parsed by the system first, but never mind). For the if-then statement there is no output if "condition" is not met. For the if-then-else statement, either the part between $[condition$ and $]condition[$ is displayed (if "condition" is met), or the part between $]condition[$ and $condition]$ (if "condition" is not met).
For example, consider the following code fragment:

```
… some code…
$[displayUserImage$

<img src="/$userImage$"/>

$displayUserImage]$
…some code…
```

This fragment is using the condition "displayUserImage" which indicates whether you would like to have a permanent image displayed on your weblog. $userImage$ is a substitution containing the path to such image. If you don't want such an image displayed on your blog, the result you get when accessing your blog would be:

```
…some code…
```

```
…some code…
```

Otherwise, you might get

```
…some code…

<img src="/pub/sampleblog/thatsme.jpg" />

…some code…
```

2.3 Loops

Finally, you can use loops to display various items of the same kind, e.g. links of a blogroll or entries within your blog. Such loops have the following syntax:

```
$[loop lp$
$loop]$
```

As you can see, it looks very similar to a condition, since it is also using square brackets. However, there is a slight difference: in the first line you have an identifier for this loop, in our example it's "lp". Such an identifier references the current item in each iteration of the loop, so that you can access its values:

```
$[loop lp$
        $lp.someVariable$
$loop]$
```

Here, we assume there is a value named "someVariable " to be printed throughout the loop. In order to access "someVariable", we use the loop identifier: lp.someVariable. Otherwise, if you were merely writing $someVariable$ within the loop, the system would not necessarily which "someVariable" definition you have in mind.
Please note, while the name of the loop itself is provided by the system (as for conditions and substitutions), the identifier is user-defined and can thus be anything you like. For example, instead of using "lp" in the previous example, you could also use "myLoop":

```
$[loop myLoop$
        $myLoop.someVariable$
$loop]$
```

Now, let's extend our simple_content.htm file to include a loop for displaying entries. The entry loop is named "weblogEntries", so you can start a simple enhancement as follows:

```
<h1>Sample header</h1>
<p>This is some static HTML content</p>

<h1>My entries</h1>
$[weblogEntries e$
        $e.entryText$ <br />
$weblogEntries]$
```

Using this file we get the following output (assuming there are only two entries texted "I am entry no. 1" and "I am entry no. 2"):

```
<html>
<head>
<title>My test blog</title>
</head>

<body>
Static content <br />

<h1>Sample header</h1>
<p>This is some static HTML content</p>

<h1>My entries</h1>
        I am entry no. 1 <br />
        I am entry no. 2 <br />

</body>
</html>
```

In addition, each loop has an optional else-section that can be used to print some data when no elements are contained in the loop. The syntax is the same as for a condition:

```
$[loop myLoop$
        $myLoop.someVariable$
$]loop[$
        No elements in this loop.
$loop]$
```

This else-section has the following effect: if at least one element of the loop exists, only the part between $[loop…$ and $]loop[$ is displayed. Otherwise, if there is no element at all, only the part between $]loop[$ and $loop]$ is displayed. Using our example with weblog entries, you can extend the loop as follows:

```
<h1>Sample header</h1>
<p>This is some static HTML content</p>

<h1>My entries</h1>
$[weblogEntries e$
        $e.entryText$ <br />
$]weblogEntries[$
        No entries found.
$weblogEntries]$
```

Assuming we have no entries in our blog, we would receive this output:

```
<html>
<head>
<title>My test blog</title>
</head>

<body>
Static content <br />
```

```
<h1>Sample header</h1>
<p>This is some static HTML content</p>

<h1>My entries</h1>
        No entries found.

</body>
</html>
```

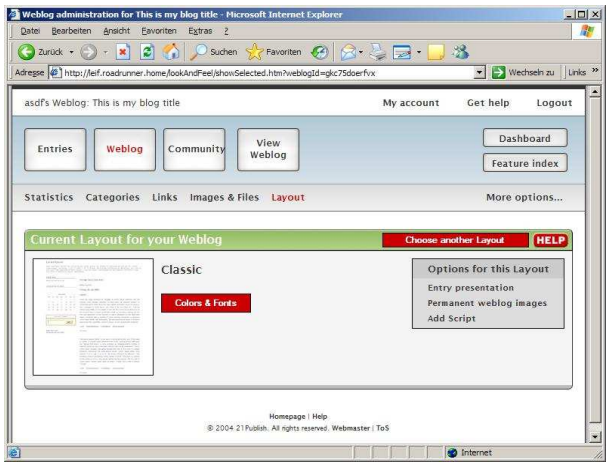
If at least one element existed, the output would not differ from the previous example without the else-section.

3. Uploading HTML template files

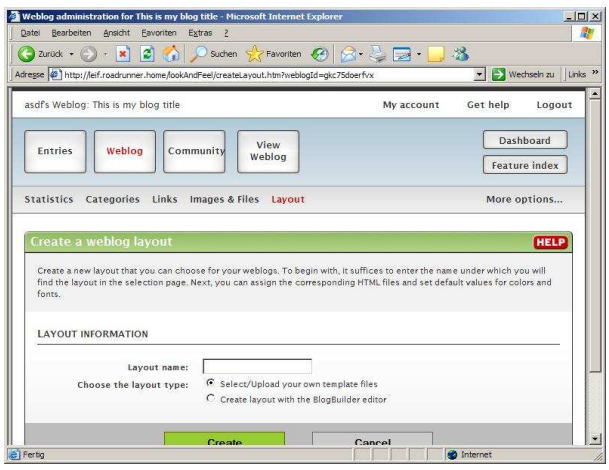
Once you have created your HTML files, you need to upload them onto your account. Depending on whether you are a portal administrator or a member of a portal, there are different options for you:

3.1 Uploading your weblog layout as a portal member

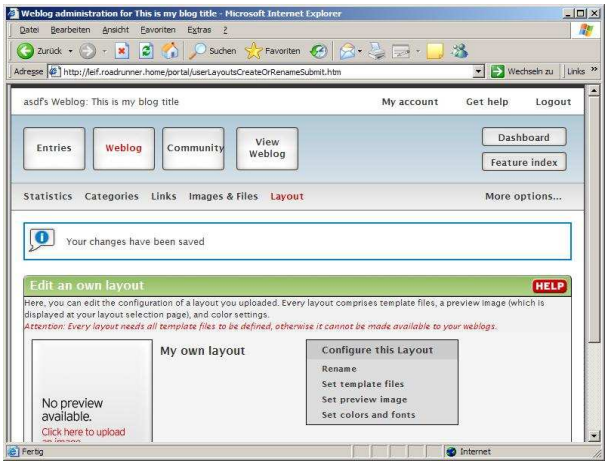
Provided your portal administrator has allowed uploading your own weblog templates, you first have to create a new weblog layout in your account. Therefore, go to "Layout" in your administration (fig. 3):



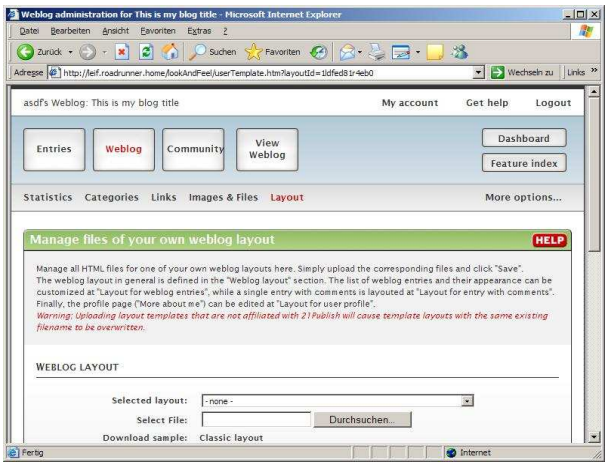
Next, click on "Choose another Layout". There you should find a link "Add own Layout". If not, you are not allowed to create your own weblog layouts.
You should find yourself on a page like this one:



Now you need to enter a name for the layout. This name will later be displayed on your layout selection screen, since you can create various template sets and then switch between them.
Note: if your portal administrator has allowed you to use a BlogBuilder template, you also find the link "Add own layout" and may enter a name. However, you may not upload your own files afterwards, since such a layout merely allows you to design your weblog using a drag&drop editor.
Once you've entered a name such as "My own layout", you will be forwarded to a configuration screen for this layout:



This screen allows you to modify your layout. You can add a preview image to this layout which will appear in your layout selection screen afterwards, rename this layout or upload your template files. The latter is what we are doing now. Thus, click on "Set template files" to proceed:



On this screen, you may upload all your template files. If you decide not to upload a file for a particular section (such as the profile page), the default file that is available to other blogs will be used.
Please note that any files with the same filename as one of those you are about to upload will be overwritten.

Once you've uploaded your files, you can view your weblog in your new design. If you want to modify your layout, simply navigate to the "Set template files" page and upload a new file.

3.2 Uploading your portal layout as a portal administrator

When creating your own layout for your portal, you can follow the same steps as a weblog user does when creating an own layout. The only difference is you have a slightly different dashboard, so menu items might differ somewhat. Besides this there is no difference in the layouts themselves, except you have a few more substitutions available.

3.3 Creating new member blog layouts as a portal administrator

As a portal administrator you not only have the option of uploading an own layout for your portal, but you can create additional layouts that are available to your members for their blogs. In order to do that, click on "Choose Layouts" at "Weblog Layouts for your Members". There, you find a link "Create own Layout". The process is then equivalent to the one for create an own weblog layout.

Note: If you are not using an ad-free account, you need to keep the 21Publish ad codes in place. Otherwise, your portal might be deactivated. Thus, you need to keep the following lines in this case:

```
$[isAdFree$
$]isAdFree[$
<div id="bannerCont">
$[showGoogleAds$ $standardBanner$ $]showGoogleAds[$ $alternativeBanner$ $showGoogleAds]$
</div>
$isAdFree]$
```