

WordPress is not PHP

Thursday, April 21st 2005

There have many concerns and question about the template system used in [WordPress](#). Mostly the concerns are with n00bs and the complications of PHP. I want to change this and that, but don't know PHP. I hear this a lot. But the thing is, you don't need to know PHP to change the [Templates](#) in [WordPress](#).

The Template System

Lets' start with [the Loop](#) as we call it. Probably the most confusing part of the [Template](#) for a non-programmer. The Loop looks like this:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

    Template Section 1: post content Template Tags go here

<?php endwhile; else: ?>

    Template Section2: no posts found stuff here.

<?php endif; ?>
```

It may look complicated, but it's actually quite simple. In [Template Section 1](#), all the posts information will be displayed. Within that section we will put all the posts [Template Tags](#), and our HTML to format them. In [Template Section 2](#), we will display a message telling the user no posts were found, if we find none to meet their criteria.

Template Section 1: The Posts

We'll start with outputting the post tile. Simply add in the [Template Tag](#) `<?php the_title() ?>`. That will output the title of the post.

Now let's output the date of the post. Simply add the date's [Template Tag](#) `<?php the_date() ?>`. That outputs the date of the post.

We will also, of coarse, need the content of the actual post. Simply, again, add the [Template Tag](#) `<?php the_content() ?>`.

Now let's put it all together with some HTML:

```
<div class="post">
  <h2> <?php the_title() ?> </h2>
  <p> <?php the_date() ?> </p>

  <?php the_content() ?>
</div>
```

And there's our simple template. Now let's add some more.

First let's add a link to the post, permalink as it's called, on the title. We get the actual URL, something like <http://mysite.com/archives/2004/09/post-title/>, with the [Template Tag](#) `<?php the_permalink() ?>`. We will need to put that in our `` HTML tag.

We also want to add in links to each page of our post, next page, previous page, etc.. We simply call another [Template Tag](#), `<?php link_pages('<p>','</p>') ?>`. Now here's another confusion people have. We've added in, what programmers would call, arguments to our [Template Tag](#). That is, we've added in `'<p>'`, `'</p>'` to the brackets of the [Template Tag](#). In this case we are just saying, if there are pages to link, output the page links with a `<p>` at the start, and a `</p>` at the end. Hence, enclosing it all in HTML paragraph tags. Most [Template Tags](#) have different arguments you can pass to them, causing them to format their output differently.

Now let's add in our new tags.

```
<div class="post">
  <h2> <a href=""><?php the_permalink() ?></a> <?php the_title() ?> </h2>
  <p> <?php the_date() ?> </p>

  <?php the_content() ?>

  <?php link_pages('<p>','</p>') ?>
</div>
```

Now we have a fully functional Posts section, [Template Section 1](#). Let's move on to [Template Section 2](#).

Template Section 2: No Posts Found

For our [Template Section 2](#), the no post found section, all we need to do is put in a message to tell the user nothing was found, to meet their criteria. Whether it was a search or a mis-typed URL. So let's use the following simple HTML:

```
<p>Sorry, no posts could be found to match your criteria.</p>
```

Now let's put it all together.

Complete Template

Putting all we learned above together, we get a nice simple [Loop Template](#).

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<div class="post">
  <h2> <a href="<?php the_permalink() ?>"> <?php the_title() ?> </a> </h2>
  <p> <?php the_date() ?> </p>

  <?php the_content() ?>

  <?php link_pages('<p>', '</p>') ?>
</div>

<?php endwhile; else: ?>

  <p>Sorry, no posts could be found to match your criteria.</p>

<?php endif; ?>
```

As you can see, we have done no PHP coding, just some simple HTML with [WordPress](#) Template Tags added in.

Obviously this is not a complete template, as we have no header and footer containing all the `<html><head><body>` tags. But with some simple HTML and more [Template Tags](#), and no PHP, we can easily add all that is needed for a complete HTML document. That's right, an HTML document, not a PHP script.

As we can see [WordPress](#) is not PHP. [WordPress](#) is powered by PHP, but uses a simple template system, easily modified by even the most novice of users. None of that complicated Perl code like MT has ... Yes, believe it or not, MT is also powered by a complicated programming language. The only difference is marketing. MT has been marketed as having simple templates with no programming skills needed. However, somehow, and I don't know why, [WordPress](#) has gotten a rap of being complicated and only for the hardcore programmers out there. But as we can see, using the [Template Tags](#) is extremely easy and requires no knowledge of PHP at all.

Changing the format and style of your [Template](#) is as easy as using a little HTML, CSS and Template Tags. Just like other Weblog systems out there.

However, as with any software, there is room for improvement.

Improving Templates

One of the problems with the [WordPress Template System](#), in my opinion, is the default [Template](#), or theme, itself. Specifically the sidebar, with all the `is_*` functions. We shouldn't expect a novice [WordPress](#) user to understand what these functions mean. I would suggest removing all the `if(is_*)` statements in place of a simple sidebar with just the essentials. Say, Search, Pages, Archives, Categories, Links, Meta information and take out all the complicated if statements and includes.

Another problem I see with the default [Template](#) is the CSS in the header. I think all CSS should be contained in the style sheets themselves. Again remove the if statements. When a user wants to modify the CSS it seems logical to have just the simple CSS in one place, to easily modify.

Another problem I see is with some of the [Template Tags](#). Such as `<?php bloginfo() ?>`. Instead of having one function to display some necessary information, like blog name, description, url, etc., why not have specific tags for the highly used information. Like say, `<?php blog_name() ?>`, `<?php blog_description() ?>`, `<?php blog_url() ?>`, etc., to maintain consistency. For some of the less used information, charset, version, etc., the `<?php bloginfo() ?>` tag would be fine.

And some of the other advanced [Template Tags](#) in use in the default [Templates](#), like `<?php list_cats(0, "", 'name', 'asc', "", 1, 0, 1, 1, 1, 1, 0, "", "", "", "") ?>`, should be simplified to not contain all those arguments. Maybe use `<?php wp_list_cats() ?>` where all those settings would be set in the admin area.

It would also be nice to try to simplify the loop, although I can't see any way of doing that now, to use as little actual PHP, like if else statements, as possible. Also take out the else, for when no posts are found, and always use the 404 template file.

The point I'm trying making is to take the PHP out of the [Templates](#), as much as possible, so users don't get confused and think they need PHP skills to modify the [Templates](#).

Those are just my thoughts on the matter.

Update:

Stuff I write doesn't usually get this much attention so I thought I should clarify some things, as I usually just write what's on my mind and do little to no editing afterwards.

The intentions of this article was not insult anyones intelligence by making something that does take time to learn, seem so simple a [monkey](#) could do it (I like Monkeys). I was just trying to show that the templates can be easy if you think about it the right way and not get stuck on all the PHP. (and please be nice to the [monkey](#) , he's very nice and doesn't usually bite)

Update 2:

If you need any help with your templates, or would like to learn more about the monkey, I'd be glad to help out. Just [drop me a line](#).