



The Handbook

Codex

[http://codex.wordpress.org/The Loop in Action](http://codex.wordpress.org/The_Loop_in_Action)

Version Date

8 August 2005

The Loop In Action

Introduction

"[The Loop](#)" is a term that refers to the main process of WordPress. You use The Loop in your [template files](#) to show posts to visitors. You could make templates without The Loop, but you'd only be able to display data from one post.

The first thing WordPress does is check that all the files it needs are present. Next, it collects the default settings, as defined by the [blog administrator](#), from the database. This includes things like the number of posts to display per page, whether commenting is enabled, and the like. Once these defaults are established, WordPress checks to see what the user asked for. This information is used to determine which posts to fetch from the database.

If the user didn't ask for a specific post, category, page, or date, WordPress uses the previously collected default values to determine which posts to prepare for the user. For example, if the blog administrator has selected to [display 5 posts per page](#), then WordPress will fetch the five most recent posts from the database. If the user did ask for a specific post, category, page, or date, then WordPress will use that information to specify which post(s) to fetch from the database.

Once all this is done, WordPress connects to the database, retrieves the specified information, and stores the results in a variable. It is The Loop that accesses this variable, and uses the values for display in your templates.

By default, if the visitor did not select a specific post, page, category, or date, WordPress uses `index.php` to display everything. For the first part of this discussion of The Loop we'll focus only on `index.php`, and the default display of your blog. Later on, once you understand how things work, we'll investigate tricks with The Loop in other template files.

The World's Simplest Index Page

The following is a fully functional index which will display the contents (and just the contents) of each post, according to the conditions used to prepare The Loop. The only purpose for showing you this is to demonstrate how little is actually necessary for the functioning of The Loop. The bulk of the stuff in your `index.php` is CSS, HTML, and PHP declarations to make The Loop look pretty.

```
<?php
get_header();
if (have_posts()) :
    while (have_posts()) :
        the_post();
        the_content();
    endwhile;
endif;
get_sidebar();
get_footer();
?>
```

Now, let's look at the bulk of the stuff that makes The Loop look pretty.

The Default Loop

The following is a step-by-step look at the default usage of the Loop that comes with the *default* and *classic* theme in the standard installation of WordPress v1.5.

Begin The Loop

Found at the top of the default `index.php` template file is the starting code for the [The Loop](#).

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

1. First it checks whether any posts were collected with the `have_posts()` function.
2. If there are any posts, a PHP [while](http://www.php.net/while) (<http://www.php.net/while>) loop is started. A `while` loop will continue to execute as long as the condition in the parenthesis is logically true. So as long as the function `have_posts()` returns a true value, The Loop will keep going.
3. The function `have_posts()` simply checks the next item in the collection of posts: if there's another item, return true; if there is no next item, return false.

Generating the Post

The function `the_post()` takes the current item in the collection of posts and makes it available for use inside this iteration of The Loop. Without `the_post()`, many of the [Template Tags](#) used in your theme would not work.

Once the post data is made available, the template can start showing post data to the visitor.

Title, Date and Author

The following [template tags](#) get the current post's title, as well as the time it was posted and who posted it.

```
<h2 id="post-<?php the_ID(); ?>">
<a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link to <?php
the_title(); ?>">
<?php the_title(); ?></a></h2>
<small><?php the_time('F jS, Y') ?> <!-- by <?php the_author() ?> --></small>
```

Post Content

The [the_content\(\)](#) template tag displays the content of the post. This is the meat and potatoes of each pass through The Loop:

```
<div class="entry">
<?php the_content('Read the rest of this entry &raquo;'); ?>
</div>
```

If you include the [Quicktag](#) button called **more**, and shown as `<!--more-->`, in the body of your post, only the portion *above* that line will be displayed to viewers. So, if you only want your front page to show the first sentence or two of every post, simply insert `<!--more-->` after the first line into every post you make.

When viewing a single post, the `<!-- more -->` delimiter is skipped. So putting the `<!-- more -->` delimiter into all your posts forces readers to click through to each individual post if they want to read the whole thing.

Additional Details

Beneath each post's content in the `index.php` template file is a place to display more information about the post, such as the categories, date, and comment information. Known as the [post meta data section](#), if you're a logged in user of sufficient privilege (or the post's author), you will also see an "Edit This" link, thanks to the [edit_post_link\(\)](#) template tag function.

```
<p class="postmetadata">
Posted in <?php the_category(', ') ?>
<strong>|</strong>
<?php edit_post_link('Edit','', '<strong>|</strong>'); ?>
<?php comments_popup_link('No Comments »', '1 Comment »', '% Comments
»'); ?></p>
```

If commenting is enabled, or if the post has comments, the [comments_popup_link\(\)](#) template tag will display a link to the comments. If you're using the [comments popup window](#), this link will open the comments window; otherwise it will jump right to this post's comments.

If the visitor is viewing an index of posts (*i.e.*: more than one post in The Loop), the `comments_popup_link()` link will take the reader to this post's individual page.

Trackback Autodiscovery

The [trackback_rdf](#) template tag's function is to output machine-readable code used for [trackback](#) auto-discovery.

```
<!--
<?php trackback_rdf(); ?>
-->
```

Note: The `trackback_rdf()` tag is supposed to be used with [comments](#) around it. It is not "turned off".

Ending The Loop

The following ends The Loop. After this, the various post-related template tags will not work as expected (or if they do, they will use the last post from The Loop). This means, that if you need to use a template tag that works **within The Loop**, you need to put it in before this point.

```
<?php endwhile; ?>
```

This section, immediately after the end of The Loop, displays navigation controls to move forward and backward by each web page.

```
<div class="navigation">
<div class="alignleft"><?php posts_nav_link('','&laquo; Previous
Entries') ?></div>
<div class="alignright"><?php posts_nav_link('','Next Entries
&raquo;','') ?></div>
</div>
```

If the blog is set to display 10 posts per page, and the conditions used by The Loop collect 25 posts, there will be three pages to navigate: two pages of 10 posts each, and one page of 5 posts. The navigation links will allow the visitor to move forward and backward through the collection of posts.

The navigation controls are included *outside* The Loop, but *inside* the `if` condition, so that they only show up if there are any posts. The navigation functions themselves also check whether or not there is anything to which they will link, based on the current Loop, and only display links if there's something to link.

```
<?php else : ?>
<h2 class="center">Not Found</h2>
<p class="center">
<?php _e("Sorry, but you are looking for something that isn't here."); ?></p>
```

The `else :` clause determines what to do if `have_posts()` (from way up at the top) is false. That is to say, the stuff after the **else** will only be executed/displayed if The Loop had zero posts. No posts show up if, for example, the visitor requested a specific day for which no posts were made or a search was performed that produced no results.

```
<?php endif; ?>
```

This ends the conditional test of "if there are posts do this, else if there are no posts, do that". Once the conditional test is finished, the default `index.php` template next includes the sidebar, and finally the footer.

The Loop In Other Templates

WordPress can use different template files for displaying your blog in different ways. In the default WordPress theme, there are [template files](#) for the index view, category view, and archive view, as well as a template for viewing individual posts. Each of these uses [The Loop](#), but does so with slightly different formatting, as well as different uses of the [template tags](#).

For any view which does not have a separate template file, WordPress will use `index.php` by default. If a visitor requests a single post, WordPress will first look for a file named `single.php`. If that file exists, it will be used to present the post to the visitor. If that file does not exist, WordPress will use `index.php` to present the post to the visitor. This is called the [Template Hierarchy](#).

If you are making your own [Theme](#), it's often helpful to look at the [template files](#) from the default Theme as a point of reference. It's also helpful to use your theme's `index.php` as a template for your other template files. Doing so may give you a known and working page from which to begin making changes as you create more template files.

Different Archive Format

An *archive* is a collection of historical posts. In the default usage, the posts displayed on your main index are recent chronological postings. When a visitor clicks on one of your archive links, or if they manually request a specific date (<http://www.example.com/blog/index.php?m=200504> or <http://www.example.com/blog/2005/04> to select all posts from April, 2005), WordPress will display an *archive* view. By default, the archive will use `index.php`, and thus look the same as your front page, just displaying the posts from April 2005.

When WordPress prepares an [archive view](#) for a visitor, it specifically looks for a file named `archive.php` in your current theme's directory. If you'd like to visually disambiguate archives from your front page, simply copy `index.php` to `archive.php`, and edit `archive.php` as necessary!

For example, if you want to show only post titles, and no post content, for your list of archives, you could use something like this:

```
<?php get_header(); ?>
<div id="content" class="narrowcolumn">

    <?php if (have_posts()) : ?>
        <?php while (have_posts()) : the_post(); ?>
            <div class="post">
                <h2 id="post-<?php the_ID(); ?>">
<a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link to <?php
the_title(); ?>"><?php the_title(); ?></a></h2>
                <small><?php the_time('F jS, Y') ?> <!-- by <?php the_author() ?> --
</small>
            </div>
        <?php endwhile; ?>
<div class="navigation">
<div class="alignleft">
<?php posts_nav_link('','&laquo; Previous Entries') ?>
</div>
<div class="alignright">
<?php posts_nav_link('','Next Entries &raquo;','') ?>
</div>
</div>
<?php else : ?>
    <h2 class="center">Not Found</h2>
    <p class="center"><?php _e("Sorry, but you are looking for something that isn't
here."); ?></p>
    <?php endif; ?>
</div>
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Different Category Format

Like the archive views, WordPress looks for a separate template file for [category views](#). If a visitor clicks on a link for a category in your blog, they will be taken to the category view. WordPress will prepare The Loop with posts from that category only, limiting the number of posts per the blog's default settings.

To make your category view different from your index view, copy `index.php` and rename it `category.php`. For a category view, it's probably not necessary to list the categories to which a post is assigned, so let's remove that portion. Instead, let's announce the category

at the top of the page:

```
<?php get_header(); ?>
<div id="content" class="narrowcolumn">
<p>
<strong>
  <?php single_cat_title('Currently browsing '); ?>
</strong><br />
<?php echo category_description(); ?>
</p>
<?php if (have_posts()) : ?>
  <?php while (have_posts()) : the_post(); ?>
    <div class="post">
      <h2 id="post-<?php the_ID(); ?>">
<a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link to <?php
the_title(); ?>">
<?php the_title(); ?></a></h2>
      <small>
        <?php the_time('F jS, Y') ?>
        <!-- by <?php the_author() ?> -->
      </small>
    </div>
  <?php endwhile; ?>
  <div class="navigation">
    <div class="alignleft">
      <?php posts_nav_link('','&laquo; Previous Entries') ?>
    </div>
    <div class="alignright">
      <?php posts_nav_link('','Next Entries &raquo;','') ?>
    </div>
  </div>
<?php else : ?>
  <h2 class="center">Not Found</h2>
<p class="center"><?php _e("Sorry, but you are looking for something that isn't
here."); ?></p>
<?php endif; ?>
</div>
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Different Formats for Different Categories

As explained in the [Template Hierarchy](#), it is possible to [create separate template files for each category](#). Simply name the file `category-X.php`, where X is the numerical ID of the category. Consider carefully whether you need a whole new template for a specific category.

Let's look at two categories, "Plants" and "Flowers", with category IDs 3 and 4, respectively. Next to each post title in the output you want to have picture of either a plant, or a flower, depending on which category is being displayed. You could:

- Use two separate files, `category-3.php` and `category-4.php`, each with a different `img` tag for each post title.
- Use a conditional test inside your default `category.php` file to check whether the current category is "Plants" or "Flowers" (or neither), and display the appropriate image:

```
<?php if (in_category('3') ):
// we're in the Plants category, so show a plant ?>
```

```

<img src='/images/plant.png' alt='a plant' />
<?php } elseif (in_category('4') ):
    // we're in the Flowers category, so show a flower ?>
    <img src='/images/flower.png' alt='a pretty flower' />
<?php endif; // end the if, no images for other other categories ?>

```

If you added another category, "Cars", which you wanted to display in a *significantly* different way, then a separate `category-x.php` would be more appropriate.

Different CSS For Different Categories

Many users want to create separate CSS files for a specific category. This, too, can be easily accomplished. It is important to remember that stylesheets are defined and loaded in the `<head>` section of the HTML document. WordPress uses the `header.php` file for this. In the default `header.php`, find this line:

```

<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>"
type="text/css" media="screen" />

```

And change it to something like this:

```

<?php if ( in_category('5') ) { // Load special CSS for "Cars" category ?>
    <link rel="stylesheet" href="<?php get_template_directory(); ?>/category-
5.css" type="text/css" media="screen" />
<?php } else { ?>
    <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>"
type="text/css" media="screen" />
<?php } ?>

```

Note: The Cars template uses the `category-5.php` file to override the default layout. In this example the CSS file is named after the category template file to which it will be applied, rather than the actual name of the category. Thus, you know that `category-5.css` goes with `category-5.php`.

Different Single Post Format

When viewing any single post (or [permalink](#)), WordPress will use `single.php`, if present.

This portion, from the WordPress default `single.php`, provides the [post meta data information](#) about the current post:

```

<p class="postmetadata alt">
<small>
This entry was posted on <?php the_time('l, F jS, Y') ?> at <?php the_time() ?>
and is filed under <?php the_category(' ', ' ') ?>.
You can follow any responses to this entry through
the <?php comments_rss_link('RSS 2.0'); ?> feed.
<?php
if (('open' == $post->comment_status) && ('open' == $post->ping_status)) {
    // Both Comments and Pings are open
    ?>
    You can <a href="#respond">leave a response</a>, or
    <a href="<?php trackback_url(display); ?>">trackback</a>
from your own site.
<?php
} elseif (!('open' == $post->comment_status) && ('open' == $post->ping_status))
{

```



```
// Only Pings are Open
?>
    Responses are currently closed, but you can
    <a href="<?php trackback_url(display); ?> ">trackback</a>
from your own site.
<?php
} elseif (('open' == $post->comment_status) && !('open' == $post->ping_status))
{
// Comments are open, Pings are not
?>
    You can skip to the end and leave a response. Pinging is currently not
allowed.
<?php
} elseif (!('open' == $post->comment_status) && !('open' == $post->ping_status))
{
// Neither Comments, nor Pings are open
?>
    Both comments and pings are currently closed.
<?php
}
edit_post_link('Edit this entry.', '', ''); ?>
</small>
</p>
```

This sort of information -- whether comments are open or closed -- is largely inappropriate on an index, archive, or category view; which is why it's only included in the `single.php` template file.

Other Loop Tricks

Now that you have a good introduction to the basic uses for the WordPress Loop, let's introduce you to some more Loop effects and tricks.

Static Front Page

How can you display something special *only* on the front page of your blog? That's right, only on the front page or home page, and have it not be seen anywhere else on your site. Easy! We call this the *static front page*. The front or first page of your site isn't really static. It's just using the Loop to make it look that way.

To make this Loop trick work, use the [is_home\(\)](#) conditional template tag function.

In your `index.php`, use an `if ()` test to conditionally output additional content:

```
<?php get_header(); ?>
<?php if (is_home()) {
    // we're on the home page, so let's show a picture of our new kitten!
    echo "<img src='/images/new_kitty.jpg' alt='Our new cat, Rufus!' />";
    // and now back to our regularly scheduled home page
} ?>
```

The function `is_home()` will only produce a true value if the visitor is not requesting a specific post, page, category, or date, so it only shows up on the "home" page.

For more information, see [Creating a Static Front Page](#).

Excerpts Only

The easiest way to display excerpts, instead of the full content, of posts, replace all instances of `the_content()` with `the_excerpt()`. If you have not created explicit excerpts for your posts, this function will automatically display the first 120 words of the post.

```
<div class="entry">
<?php the_excerpt(); ?>
</div>
```

Different Headers/Sidebars/Footers

WordPress offers the `get_header()`, `get_sidebar()`, and `get_footer()` [Include Tags](#) for use in your [template files](#). These functions make it easy to define a standard header/sidebar/footer which is easily editable. Any changes made to these files will immediately be made visible to viewers, without any work on your part.

But sometimes you might not *want* a sidebar. If you don't want a sidebar, simply exclude the call to the `get_sidebar()` function from your template. For example, the `single.php` template in the WordPress default theme does not include a sidebar.

To create your own **different** sidebar, you have two choices.

1. Include the sidebar contents directly into the template file on which you're working. If you want category-3 to have a different sidebar, edit `category-3.php` and include the necessary HTML and PHP to generate your distinctive sidebar.
2. Use the PHP [include](http://www.php.net/include) (<http://www.php.net/include>) function, to include another file. The WordPress `get_sidebar()` function *only* loads `sidebar.php`. If you make a file named `sideleft.php`, you would include it like this:

```
<?php include(TEMPLATEPATH . '/sideleft.php'); ?>
```

Using the WordPress default [Template Hierarchy](#), if you want to use the same elements on multiple or different templates, it's probably best to put them in separate template files and use the PHP `include()` function. If the element you're adding is specifically for one template file, it's probably best to include it directly in that template file.

Summary

We've just scratched the surface of what can be done with the Loop. As a reminder, the following are resources that will help you customize your own [WordPress Loop](#).

- [Template Files](#)
- [Template Tags](#)
- [Template Hierarchy](#)
- [Conditional Tags](#)