



The Handbook

Codex

<http://codex.wordpress.org/Pages>

Version Date

6 August 2005

Pages

Introduction

Pages, or **WordPress Pages** are like Posts, except they can do much more than Posts, and they live outside of the normal blog chronology. You can use Pages to organize and manage any amount of content. Pages were added as new feature in WordPress 1.5.

But how do **Pages** in WordPress actually operate? The purpose of this document is to attempt to explain what a Page is and what it is not, to describe what a Page can do, and to give a few examples.

Terms Used in this Article

For the purpose of clarity, throughout this document:

- "page", with a lowercase "p", will describe *any* [HTML](#) document on the web.
- "Page", with a capital "P", will refer to a "*WordPress Page*", the feature of WordPress that this wiki page details. Use "**Page**", in **bold**, when the use of the term is ambiguous.

For further **Page** related nomenclature issues, see [A Note on Nomenclature](#).

What is a Page?

Posts are time-oriented objects. You write them at a specific time, and that time defines their context.

Pages, on the other hand, are most often used to present information about yourself or your site that is somehow timeless - information that is always applicable. For example, you might write a Post describing what you did or thought on a particular morning ("Breakfast was good"), but on a Page you might write something whose context is less time dependent ("This site is about breakfast").

Of course, this is *your* WordPress; you can do whatever you want with it and its features. Pages can be used to present any information you want to live "outside" your blog. Experiment and be creative. And, as always, have fun doing it!

Some examples of Pages to create on your site may include, Copyright, Legal Information, Reprint Permissions, Contact Information, About Me, About Site, Accessibility Statement, among other things.

In general, Pages are very similar to Posts in that they both have Titles and Content and can use your site's Presentation Templates to maintain a consistent look throughout your site. Pages, though, have several key distinctions which makes them quite different from Posts.

Pages in a Nutshell

What Pages Are

- **Pages** are for content that is less time dependant than Posts.
- **Pages** can be organised into pages and [SubPages](#).
- **Pages** can use different [Page Templates](#) which can include [Template Files](#), [Template Tags](#) and other PHP code.

What Pages are Not

- **Pages** are not Posts. They do not cycle through your blog's main page, **nor can they be associated with Categories**.
- **Pages** are not files. They are stored in your database just like your Posts are.
- Although you can put Template Tags and PHP code into a Page Template, you cannot put these into the content of a Page and expect them to run. (**Note:** You can achieve this by use a PHP evaluating Plugin such as [RunPHP](http://dev.wp-plugins.org/wiki/RunPHP) (<http://dev.wp-plugins.org/wiki/RunPHP>). See also the list of [Posts Formatting Plugins](#).)

Creating Pages

To create a new Page, log in to your WordPress installation with sufficient admin privileges to create new articles, and click on the *Write* tab in the admin interface, which will have a *Write Page* tab. Clicking the *Write Page* tab will lead you to the page where you can create your new page.

Note: Your .htaccess file *must* be writeable for Page Permalinks to work, otherwise you must update your .htaccess file every time you create a Page.

Listing Your Pages on Your Site

WordPress is able to *automatically* generate a list of Pages on your site, for example within the sidebar, using a [Template Tag](#) called `wp_list_pages()`. Please see the [documentation describing this tag's use](#) for information on how to

- sort the list of Pages (to fully customize the order in which the Pages are listed, you might find the "Page Order" field on the Write->Write Page administration panel useful),
- [exclude](#) (or 'hide') a Page from the list,
- control which Pages are displayed (i.e. all Pages or just certain Sub-Pages), and
- control how deep into your Page hierarchy the list goes.

Naturally, you can also link to Pages manually with an HTML link. For example, if you want your Copyright Page listed in your footer, that link might read

```
<a title="Copyright information" href="wordpress/?page_id=14">Copyright 1996-2006</a>
```

if you do not have [Permalinks](#) set up, or

if you *do* have [Permalinks](#) set up.

Note: Your .htaccess file *must* be writeable for Page Permalinks to work, otherwise you must update your .htaccess file every time you create a Page.

Organizing Your Pages

Just as you can have Sub-categories within your Categories, you can also have **SubPages** within your Pages, creating a hierarchy of pages.

For example, suppose you are creating a WordPress site for a travel agent and would like to create an individual Page for each continent and country to which the agency can make travel arrangements. You would begin by creating a Page called "Africa" on which you could describe general information on travel to Africa. Then create a series of Pages which would be SubPages to "Africa" and might include "Lesotho", "Cameroon", "Togo", and "Swaziland". Another individual Page is made for "South America" and would feature SubPages of "Brazil", "Argentina", and "Chile". Your site would then list:

- Africa
 - Cameroon
 - Lesotho
 - Swaziland
 - Togo
- South America
 - Argentina
 - Brazil
 - Chile

To begin the process, on the [Administration](#) > [Write](#) > [Write Page](#) panel, in the upper right corner of the panel, is a drop-down box called "Page Parent". This contains a list of all the Pages already created for your site. To turn your current Page into a SubPage, or "Child" of the "Parent" Page, select the appropriate Page from the drop down list. If you specify a Parent other than "Main Page (no parent)" from the list, the Page you are now editing will be made a Child of that selected Page. When your Pages are [listed](#), the Child Page will be nested under the Parent Page. The [Permalinks](#) of your Pages will also reflect this Page hierarchy.

In the above example, the [Permalink](#) for the Cameroon Page would be:

<http://example.com/africa/cameroon/>

Page Templates

Individual Pages can be set to use a specific, custom **Page Template** that you create within your Theme. This new Page Template will then override the default `page.php` Page Template included with your Theme. See [What Template is used to Display a Particular Page?](#), below, to find out exactly which Template will be used. But read what follows first, so you understand the answer :))

WordPress can be configured to use different Page Templates for different Pages. Toward

the bottom of the Write->Write Page administration panel is a drop-down labeled "Page Template". From there you can select which Template will be used when displaying this particular Page. **NOTE:** In order to access the Page Template selector, there must be at least one Page Template available in the active theme.

Default Theme Page Templates

The Default theme contains three Page Templates for your use:

- `page.php` - Default Page Template: displays Page content
- `archives.php` - ignores Page content and instead displays a list of Archives by Month and Archives by Subject (by Category)
- `links.php` - ignores Page content and instead displays your links using [get_links_list](#)

What Template is used to Display a Particular Page?

WordPress will look for several template files in your active Theme. The first one it finds will be used to display any given Page. Below is the order of files WordPress will look for:

1. The Page's selected "Page Template"
2. `page.php`
3. `index.php`

Creating your own Page Templates

The files defining each Page Template are found in your [Theme's](#) directory. To create a new Template for a Page you must create a file. Let's call our first Page Template for our Page `snarfer.php`. At the top of the `snarfer.php` file, put the following:

```
<?php
/*
Template Name: Snarfer
*/
?>
```

The above code defines this `snarfer.php` file as the "Snarfer" Template. Naturally, "Snarfer" may be replaced with most any text to change the name of the Page Template. This Template Name will appear in the Theme Editor as the link to edit this file.

The file may be named *almost* anything with a `.php` extension (see [reserved Theme filenames](#) for filenames you should *not* use; these are special file names WordPress reserves for specific purposes).

What follows the above five lines of code is up to you. The rest of the code you write will control how Pages that use the Snarfer Page Template will display. See [Template Tags](#) for a description of the various WordPress Template functions you can use for this purpose. You may find it more convenient to copy some other Template (perhaps `page.php` or `index.php`) to `snarfer.php` and then add the above five lines of code to the beginning of the file. That way, you will only have to *alter* the HTML and PHP code, instead of creating it all from scratch. Examples are shown [below](#).

Examples of Pages and Templates

The following is a list of instructional examples. Feel free to make additions.

Archives with Content

A Page Template that allows shows the Page's content at the top, and then displays a list of archive months and categories below it. This is designed to work with WordPress's Default theme (aka Kubrick), but will probably work with many other themes with a little modification.

Save this to `arc-cont.php`:

```
<?php
/*
Template Name: Archives with Content
*/
?>

<?php get_header(); ?>

<div id="content" class="widecolumn">

    <?php if (have_posts()) : while (have_posts()) : the_post();?>
    <div class="post">
        <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
        <div class="entrytext">
            <?php the_content('<p class="serif">Read the rest of this page »</p>'); ?>
        </div>
    </div>
    <?php endwhile; endif; ?>
    <?php edit_post_link('Edit this entry.', '<p>', '</p>'); ?>

</div>
<div id="main">

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

<h2>Archives by Month:</h2>
    <ul>
        <?php wp_get_archives('type=monthly'); ?>
    </ul>

<h2>Archives by Subject:</h2>
    <ul>
        <?php wp_list_cats(); ?>
    </ul>

</div>
<?php get_footer(); ?>
```

WordPress as a CMS

With the new Pages feature in 1.5, it's easy to use WordPress for basic content management.

Using a Page as the Front Page

Using the [Static Front Page Plugin](http://www.semiologic.com/projects/static-front/) (<http://www.semiologic.com/projects/static-front/>), it is possible to set any Page as the "front page" of your site. The plugin modifies the home page query and sticks the Page with a Page slug of "home" to the front page.

When the Page is being displayed as the Home Page, if a Page Template with the filename `home.php` exists for your active Theme, the plugin will override the Page's set Page Template and use `home.php` instead. The Page's set Page Template will still apply if the Page is visited like a standard Page (eg <http://example.com/home/>)

Including a Page

You might also want to include Pages in various places on your site. That way, you can have an easy way to edit elements of your website. There is a plugin called [Include Page](http://beetle.cbtlsl.com/archives/2005/03/02/wordpress-include-page-plugin/) (<http://beetle.cbtlsl.com/archives/2005/03/02/wordpress-include-page-plugin/>) that makes doing this easy.

```
<?php
/*
Plugin Name: Include Page
Version: 1.0
Plugin URI: http://beetle.cbtlsl.com/categories/include_page
Author: Brent Loertscher
Author URI: http://beetle.cbtlsl.com
Description: Adds an include_page() function to include the contents of a page
in a template.
Installation:
1. Download the file http://beetle.cbtlsl.com/wp-plugins/include_page.phps
2. Rename the file to include_page.php and put it in your wp-
content/plugins/directory.
3. Activate the plugin from your WordPress admin 'Plugins' page.
4. Make use of the function in your template.
function include_page ($post_id) - include page with corresponding post_id
*/
/*
Copyright (c) 2005, Brent Loertscher
Released under the GPL license
All rights reserved.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
```

```

IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
function include_page ($post_id) {
    global $wpdb;
    $pages = $wpdb->get_results ("SELECT " .
        "post_content " .
        "FROM $wpdb->posts " .
        "WHERE ID = " . $post_id);
    foreach ($pages as $page) {
        echo $page->post_content;
    }
}
?>

```

A Note on Nomenclature

A page can be *static* or *dynamic*. Static pages are those which have been created once and do not have to be regenerated every time a person visits it. In contrast, dynamic pages do need to be regenerated every time they are viewed; code for what to generate has been specified by the author, but not the actual page itself. These use extensive PHP code which is evaluated each time the page is visited, and the content is thus generated on the fly, upon each new visit.

Almost everything in WordPress is generated dynamically, including **Pages**. Everything you and others write in WordPress (Posts, **Pages**, Comments, Blogrolls, Categories, etc.) is stored in your [MySQL](#) database. When your site is accessed, that database information is then used by your WordPress [Templates](#) from your current [Theme](#) to generate the web page being requested. Thus, WordPress information is dynamic, including the information contained in your **Pages**.

An example of a static page might be an [HTML](#) document (without any [PHP](#) code) you've written as an addition to your dynamically generated WordPress pages, perhaps an "About Me" page. The problem with purely static pages is that they are difficult to maintain. Changes you make to your WordPress settings, [Themes](#) and [Templates](#) will not be propagated to pages coded only in HTML. The **Page** feature of WordPress was developed, in part, to alleviate this problem. By using **Pages**, users no longer have to update their static pages every time they change the style of their site. Instead, if written properly, their dynamic **Pages** will update themselves along with the rest of your blog.

Despite the dynamic nature of **Pages**, many people refer to them as being static. In the context of web publishing, static and dynamic mean what has been described above. More generally, however, static can mean "characterized by a lack of change". It is easy to see how this definition influenced the word's use in describing types of web pages. It is also to easy to see why people think of **Pages** as being static; Posts come and go, but **Pages** are here to stay since **Pages** are typically used to display information about your site which is constant (e.g. information about yourself, description of your site, etc.).

In other words, a **Page** contains *static information* but is *generated dynamically*. Thus, either "static" or "dynamic" may be validly used to describe the nature of the WordPress **Page** feature. However, in order to avoid confusion, and because **Pages themselves** are dynamic while it is only their *contents* which are in some way static, this document does not refer to **Pages** as being static.