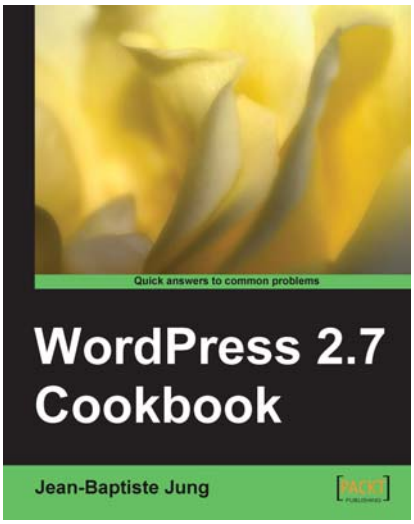
WordPress 2.7 Cookbook

Jean-Baptiste Jung



Chapter No. 10
"Enhancing User Experience"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.10 "Enhancing User Experience"

A synopsis of the book's content

Information on where to buy this book

About the Author

Jean-Baptiste Jung is a Web developer, Web designer, and blogger born in Paris, France and now living in Wallonia (French-speaking part of Belgium) with his wife and cat.

Jean unearthed the World Wide Web in 1998 and started creating web sites three years later. In 2006, while working as a freelance Web developer for a well known French TV channel, Jean started to work with blogs and WordPress. A few months later, he created his first blog.

He became immensely passionate about WordPress and launched a blog dedicated to WordPress hacks, `http://www.wprecipes.com/`, which quickly managed to become one of the most popular WP-related web sites over the Internet. Meanwhile, Jean is also an author on some prestigious blogs, such as WpHacks, ProBlogDesign, and Smashing Magazine.

When he's not blogging or tweaking web sites, Jean enjoys travelling and spending time with his wife and cat. He has a strong love for animals and always stands up to defend animal rights.

I'd like to thank my wife Emmanuelle as well as our cute cat for being here with me. They mean so much to me.

WordPress 2.7 Cookbook

About 120,000 blogs are created every day. Most of them quickly die, but a few stay, grow up, and then become well known and respected places on the Web. If you are seriously interested in being in the top league, you will need to learn all the tricks of the trade. WordPress 2.7 Cookbook focuses on providing solutions to common WordPress problems, to help make sure that your blog will be one of the ones that stay.

The author's experience with WordPress enables him to share insights on using WordPress effectively, in a clear and friendly way, giving practical hands-on solutions to WordPress problems, questions, and common tasks—from themes to widgets and from SEO to security.

Are you feeling limited with WordPress, or are you wondering how popular blogs do a certain kind of thing that you can't? With this cookbook, you will learn many WordPress secrets and techniques with step-by-step, useful recipes dedicated to achieving a particular goal or solving a particular problem. You'll learn the secret of expensive premium themes, how to optimize your blog for SEO and online profits, and how to supercharge WordPress with killer functions used by the most popular blogs over the Internet.

What This Book Covers

Chapter 1 introduces you to WordPress. It introduces you to some basic—but often forgotten—built-in tools to make your blogger life easier.

Chapter 2 discusses the various WordPress themes and provides you with the location where to find professional—but free—themes. It also teaches you how to install and customize these themes.

Chapter 3 teaches you how to customize any existing WordPress theme and make it fi t your taste and need.

Chapter 4 describes an easy procedure to install plugins. It shows what different plugins can do for you in particular situations. It also teaches you how to download and install widgets and how to make a WordPress theme widget-ready.

Chapter 5 describes the procedure to display posts and to retrieve post information from WordPress.

Chapter 6 teaches you to manage a multi-author blog and integrate powerful functions for creating an author page template.

Chapter 7 educates you about hacks, plugins, and tips and tricks to secure your database and your WordPress blog.

Chapter 8 teaches you—by providing you with tips and tricks—the art of getting traffic from search engines to your blog.

Chapter 9 discusses the monetization solution that can be used in a WordPress blog. It will also provide you with many tips and tricks to make money while blogging.

Chapter 10 helps you make your blog easy and functional for your visitors.

Chapter 11 provides you with tips and hacks to make your blog better than your competitor's blog.

Chapter 10
Enhancing User Experience

As a blogger, I read loads of blog posts every day, on many different blogs. Very often, I'm scared to see how many blogs have a non user-friendly interface.

How often does it happen that you can't click on the logo to go back to the blog homepage, or can't find what you're looking for by using the search engine? It is a well known fact that in blogging the content is king, but a nice, user friendly interface makes your blog look a lot more professional, and much easier to navigate.

In this chapter, I'll show you what can be done for enhancing user experience and makeing your blog a better place.

In this chapter, you will learn:

- Replacing the Next and Previous links by a paginator
- Highlighting searched text in search results
- Using the CSS sliding doors technique within WordPress
- Creating a dropdown menu for your categories
- Adding a breadcrumb to your theme
- Displaying related posts
- Display tabs on your sidebar

Replacing the Next and Previous links by a paginator

When a web site, or blog, publishes lots of articles on a single page, the list can quickly become very long and hard to read. To solve this problem, paginations were created. Pagination allows displaying 10 articles (for example) on a page. If the user wants, then he or she can go to the next page, or click on a page number to directly go to the related page.

I definitely don't understand why WordPress still doesn't have a built-in pagination system. Instead, at the bottom of each page you'll find a **Next** link to go to the next page, and a **Previous** link to go back. This works fine when you're on page two and would like to go to page three, but what if you're on page one, and remember a very interesting article which was located on page eight? Are you going to browse page per page until you find your article? The answer is yes, because you don't have the choice. You can't jump from page one to page eight.

In this recipe, I'll show you how to integrate a pagination plugin in your WordPress blog theme. One very good point of this recipe is that the plugin file is embedded in your theme, so if you're a theme designer, you can distribute a theme which has a built-in pagination system.

Getting ready

To execute this recipe you need to grab a copy of the WP-PageNavi plugin, which can be found at `http://wordpress.org/extend/plugins/wp-pagenavi/`. I have used version 2.40 of the Wp-PageNavi plugin in this example.

Once you have downloaded it, unzip the zip file but don't install the plugin yet.

How to do it

1. Open the WP-PageNavi directory and copy the following files into your WordPress theme directory (For example, `http://www.yourblog.com/wp-content/theme/yourtheme`).
   - `wp-pagenavi.php`
   - `wp-pagenavi.css`
2. Once done, edit the `index.php` file

You can do the same with other files, such as `categories.php` or `search.php` as well.

3. Find the following code (or similar) in your `index.php` file:

```
<div class="navigation">
<div class="alignleft"><?php next_posts_link('Previous
    entries') ?></div>
<div class="alignright"><?php previous_posts_link('Next
    entries') ?></div>
</div>
```

4. Replace that with the following code:

```
<?php
include('wp-pagenavi.php');
if(function_exists('wp_pagenavi')) { wp_pagenavi(); }
?>
```

5. Save the `index.php` file. If you visit your blog now, you'll see that nothing has changed. This is because we have to call a function in the `wp-pagenavi.php` file.

6. Open this file and find the following code (line 61):

```
function wp_pagenavi($before = '', $after = '') {
global $wpdb, $wp_query;
```

7. We have to call the `pagenavi_init()` function, so let's do this in the following way:

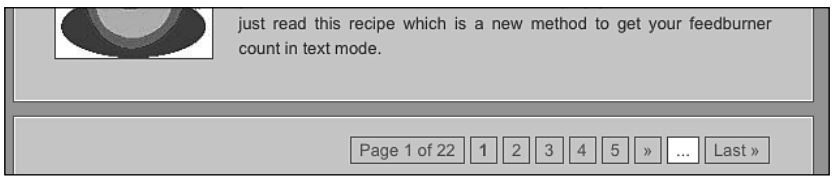
```
function wp_pagenavi($before = '', $after = '') {
global $wpdb, $wp_query;
pagenavi_init(); //Calling the pagenavi_init() function
```

8. Now, save the file and refresh your blog. The pagination is now displayed! This is great news, but the pagination doesn't look good.

9. To solve this problem, you simply have to integrate the `wp-pagenavi.css` file that you copied earlier in your theme directory. To do so, open the `header.php` file from your theme and paste the following line between the `<head>` and `</head>` tags:

```
<link rel="stylesheet" href="<?php echo TEMPLATEPATH.'/pagenavi.
    css';?>" type="text/css" media="screen" />
```

10. Visit your blog homepage one more time. The pagination looks a lot better. You may have to edit the `wp-pagenavi.css` file in order to make your pagination look and feel fit your blog style.



How it works

In this recipe, you have discovered a very useful technique that I often use on my blogs, or in the themes that I distribute—the integration of a WordPress plugin into a theme.

When the plugin is integrated into your theme as I have shown you in this example, there's no activation process needed. All of the work is done directly from the theme.

The WP-PageNavi plugin itself works by using two values—the number of posts to be displayed per page and the first post to be displayed. Then, it executes the relevant query to WordPress database to get the posts.

The pagination bar is calculated by using the total number of posts from your blog, and then dividing this value by the number of posts per page.

One good point of this technique is that the plugin is integrated in your blog and you can redistribute the theme if you want. The end user will not have to install or configure anything.

Highlighting searched text in search results

I must admit that I'm not a big fan of the WordPress built-in search engine. One of its weakest features is the fact that searched text aren't highlighted in the results, so the visitor is unable to see the searched text in the context of your article.

Getting ready

Luckily, there's a nice hack using regular expressions to automatically highlight searched text in search results. This code has been created by Joost de Valk who blogs at `www.yoast.com`.

How to do it

This useful code is definitely easy to use on your own blog:

1. Open your `search.php` file and find the following:

```
echo $title;
```

2. Replace it with the following code:

```
<?php
$title = get_the_title();
$keys= explode(" ",$s);
$title = preg_replace('/('.implode('|', $keys) .')/iu',
    '<strong class="search-excerpt">\0</strong>',$title);
?>
```

3. Save the `search.php` file and open the `style.css` file. Append the following line to it:

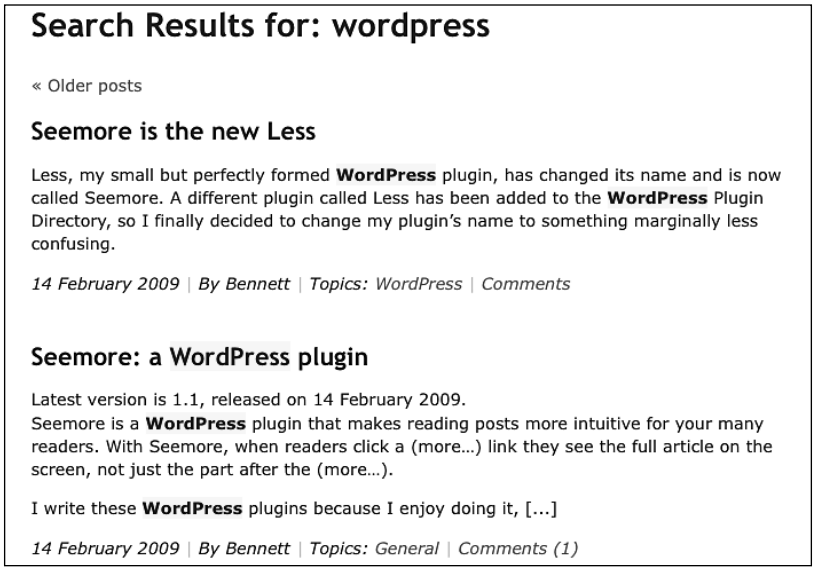
```
strong.search-excerpt { background: yellow; }
```

4. You're done. Now, the searched text will be highlighted in your search results.

How it works

This code is using PHP regular expressions to find the searched terms in the text returned by WordPress. When an occurrence has been found, it is wrapped in a `<strong>` HTML element.

Then, I simply used CSS to define a yellow background to this element.



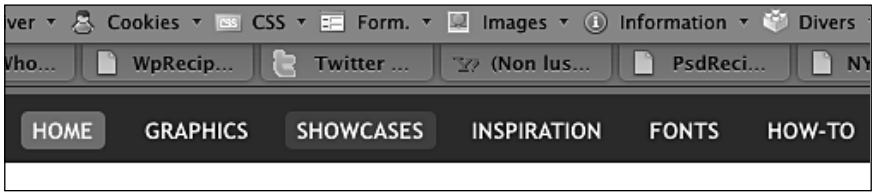
Using the CSS sliding doors technique within WordPress

The CSS "sliding doors" technique allows you to create sophisticated tabs for your menus. Sadly, WordPress doesn't allow you to use a `<span>` element in the `wp_list_pages()` and `wp_list_categories()` functions.

Getting ready

In order to achieve this very cool menu, you first need to have a bit of knowledge about the CSS sliding doors technique, and the necessary images.

If you need to know more about this technique, you should consider reading this article: `http://www.alistapart.com/articles/slidingdoors/`



If you don't already know this awesome technique here's a quick example to get you started:

Let's start by building a typical navigation list:

```
<ul id="nav">
<li><a href="#">link n°1</a></li>
<li><a href="#">link n°2</a></li>
<li><a href="#">link n°3</a></li>
</ul>
```

If we use CSS to apply background images to our links in order to make this menu look prettier, we'll quickly come across a big problem. We must add a fixed width to the links otherwise, the image will be truncated for a very short link, or the link will overflow the image if its width is too long.

That's why sliding doors are very useful. We just have to add a span element inside the link. Then, in our CSS, assign a different background image to both the span element and the link.

```
<ul id="nav">
<li><a href="#"><span>link n°1</span></a></li>
<li><a href="#"><span>link n°2</span></a></li>
<li><a href="#"><span>link n°3</span></a></li>
</ul>
```

Our CSS should look like this:

```
#nav a, #nav a:visited {
    display:block;
}
#nav a:hover, #nav a:active {
    background:url(images/tab-right.jpg) no-repeat 100% 1px;
    float:left;
}
#nav a span {
    float:left;
```

```
    display:block;
}
#nav a:hover span {
    float:left;
    display:block;
    background: url(images/tab-left.jpg) no-repeat 0 1px;
}
```

Please note, as this is only an example, the preceding CSS isn't complete and only shows how to apply the sliding doors hack.

How to do it

I have read many WordPress users who modified their WordPress core files to achieve this technique. I have already talked about how modifying core files is a bad idea. Instead, let's use some regular expressions (as we did in the previous recipe!) to obtain the desired effect.

Applying this hack to pages

1. Open your `header.php` file (or any file you'd like to apply this technique).
2. Look up for the following function:

```
<ul>
wp_list_pages();
</ul>
```

3. Replace it by the following code:

```
<ul id="nav">
<li><a href="<?php echo get_option('home');
     ?>/"><span>Home</span></a></li>
<?php echo preg_replace('@\<li([^>]*)>\<a([^>]*)>(.*?)\<\/a>@i',
     '<li$1><a$2><span>$3</span></a>', wp_list_pages('echo=
     0&orderby=name&exlude=181&title_li=&depth=1')); ?>
</ul>
```

4. Save the `header.php` file and open the `style.css` file. Finally append the following styles:

```
#nav a, #nav a:visited {
display:block;
}
#nav a:hover, #nav a:active {
background:url(images/tab-right.jpg) no-repeat 100% 1px;
float:left;
}
#nav a span {
```

```
float:left;
display:block;
}
#nav a:hover span {
float:left;
display:block;
background: url(images/tab-left.jpg) no-repeat 0 1px;
}
```

You may have to style it a bit more in order to make it compatible with your blog's look and feel, but basically, you're done.

Applying this hack to categories

Of course, you can easily apply the previous hack to categories. Follow these simple steps:

1. Open your `header.php` or `sidebar.php` file, depending on where your categories are listed.
2. Look up for the following function:

```
<ul>
wp_list_categories();
</ul>
```

3. Replace it by this code:

```
<ul id="nav">
<li><a href="<?php echo get_option('home');
     ?>/"><span>Home</span></a></li>
<?php echo preg_replace('@\<li([^>]*)>\<a([^>]*)>(.*?)\<\/a>@i',
     '<li$1><a$2><span>$3</span></a>', wp_list_categories('echo=
     0&orderby=name&exlude=181&title_li=&depth=1 )); ?>
</ul>
```

4. Save the file and paste these styles to your `style.css` file:

```
#nav a, #nav a:visited {
    display:block;
}
#nav a:hover, #nav a:active {
    background:url(images/tab-right.jpg) no-repeat 100% 1px;
    float:left;
}
#nav a span {
    float:left;
    display:block;
}
```

```
#nav a:hover span {
    float:left;
    display:block;
    background: url(images/tab-left.jpg) no-repeat 0 1px;
}
```

How it works

As you can see, applying this hack to categories or pages is almost the same. Basically, you just have to change the `wp_list_categories()` function for categories and `wp_list_pages()` for pages.

To embed `<span>` elements in the list, this code uses the PHP `preg_replace()` function with the `wp_list_categories()` function as a second argument. In the `wp_list_categories()` function, the `echo=0` parameter is specified, which means that the function doesn't print the result on screen but instead returns the result to be used in PHP.

Creating a drop-down menu for your categories

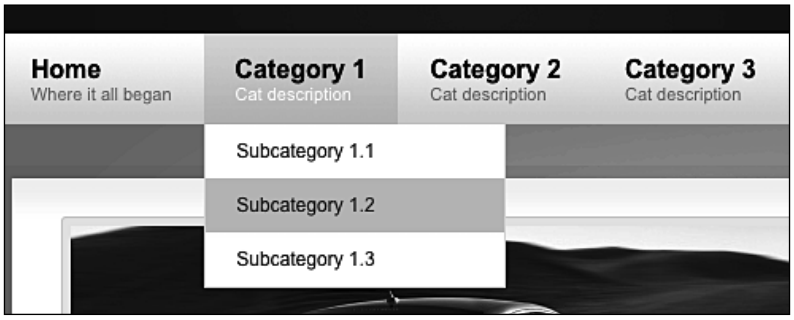
Do you use a lot of categories along with their sub-categories? If so, using a drop-down menu is a nice way to categorize content, especially on larger sites. However, giving a quick access to the categories or sub-categories to readers can become a pain.

Over the years, the drop-down menu has become very popular on the Internet. In this recipe, I'm going to show you how to create your own drop-down menu for your WordPress blog categories.

Getting ready

The menu you are going to create will first list your pages, and then at last a tab called **Categories** will obviously list your categories.

This menu is achieved only with XHTML and CSS. No JavaScript is needed (unless you want to maintain compatibility with it commonly referred to as IE6) to ensure the best SEO possible for your WordPress blog.



How to do it

In order to make this recipe more readable, I have divided it in 3 steps—the PHP and the HTML, the CSS, and the JavaScript for IE6 compatibility.

Step 1: PHP and HTML

Open the `header.php` file from your theme and paste the following code where you'd like your drop-down menu to be displayed:

```
<ul id="nav" class="clearfloat">
<li><a href="<?php echo get_option('home'); ?>/"
    class="on">Home</a></li>
<?php wp_list_pages('title_li='); ?>
<li class="cat-item"><a href="#">Categories</a>
<ul class="children">
<?php wp_list_categories('orderby=name&title_li=');
$this_category = get_category($cat);
if (get_category_children($this_category->cat_ID) != "") {
    echo "<ul>";
    wp_list_categories('orderby=id&show_count=0&title_li=
    &use_desc_for_title=1&child_of='.$this_category->cat_ID);
    echo "</ul>";
    }
?>
</ul>
</li>
</ul>
```

The purpose of this code is to make a list of all our pages and subpages, as well as a last list element named **Categories**. When a reader hovers on one of the top-level menu, the subpages (or categories) are displayed.

Step 2: The CSS

Open the `style.css` file from your theme and paste the following styles:

```
#nav{
    background:#222;
    font-size:1.1em;
}
#nav, #nav ul {
    list-style: none;
    line-height: 1;
}
#nav a, #nav a:hover {
    display: block;
    text-decoration: none;
    border:none;
}
#nav li {
    float: left;
    list-style:none;
    border-right:1px solid #a9a9a9;
}
#nav a, #nav a:visited {
    display:block;
    font-weight:bold;
    color: #f5f5f4;
    padding:6px 12px;
}
#nav a:hover, #nav a:active, .current_page_item a, #home .on {
    background:#000;
    text-decoration:none
}
#nav li ul {
    position: absolute;
    left: -999em;
    height: auto;
    width: 174px;
    border-bottom: 1px solid #a9a9a9;
}
#nav li li {
    width: 172px;
```

```
    border-top: 1px solid #a9a9a9;
    border-right: 1px solid #a9a9a9;
    border-left: 1px solid #a9a9a9;
    background: #777;
}
#nav li li a, #nav li li a:visited {
    font-weight:normal;
    font-size:0.9em;
    color:#FFF;
}
#nav li li a:hover, #nav li li a:active {
    background:#000;
}
#nav li:hover ul, #nav li li:hover ul, #nav li li li:hover ul, #nav
li.sfhover ul, #nav li li.sfhover ul, #nav li li li.sfhover ul {
    left: auto;
}
a.main:hover {
    background:none;
}
```

You may have to tweak this code a bit to match up to your blog's look and feel, for example, by adjusting colors. Once you are finished, simply save the file.

Step 3: Optional JavaScript

I'm not going to teach you something new here since, Internet Explorer 6 is a totally obsolete, crappy, and buggy browser. Sadly, many peoples are still using it and you may want to make sure that your blog is IE6 compliant.

Modern browsers as such as Safari, Firefox, Opera, and even Internet Explorer 7 will not have any problem with the `:hover` pseudo-class on `li` elements. But you guessed it, it is asking too much from the IE6.

1. To ensure backward compatibility on your WordPress blog, create a new file and call it `dropdown.js`.
2. Put this code in the `dropdown.js` file:

```
<![CDATA[//><!--
sfHover = function() {
var sfEls = document.getElementById("nav").
    getElementsByTagName("LI");
for (var i=0; i<sfEls.length; i++) {
    sfEls[i].onmouseover=function() {
        this.className+=" sfhover";
    }
```

```
sfEls[i].onmouseout=function() {
    this.className=this.className.replace(new
        RegExp(" sfhover\\b"), "");
    }
}
}
if (window.attachEvent) window.attachEvent("onload", sfHover);
//-><!]]>
```

3. Save the `dropdown.js` file and upload it to your `wp-content/themes/yourtheme` directory.
4. Open `header.php` and add the following line within the `<head>` and `</head>` HTML tags:

```
<!--[if lte IE 6]>
<script type="text/javascript" src="<?php bloginfo(
    'template_url');?>/dropdown.js"></script>
<![endif]-->
```

That's all! Your blog now has a very professional looking drop-down menu.

How it works

As IE6 cannot deal with `:hover` pseudo-classes on `<li>` elements, this small piece of code automatically adds a new CSS class, named `sfhover` to `<li>` elements when they are hovered over. When the mouse goes out of the top level element, a new function is executed, using a regular expression to remove the `sfhover` class.

There's more...

Now that I have shown you're the principle of creating a drop-down menu, you can use what you have just learned to create various kinds of menus. As an example, let's see how to re-use the previous code and create a very nice horizontal drop-down menu.

Creating a horizontal drop-down menu

As you'll notice by observing the code, there's a lot of similar things between this code and the one that you saw earlier.

Part 1: PHP and HTML

Simply copy this code where you want the menu to be displayed, for example, in your `header.php` file:

```
<ul id="nav2" class="clearfloat">
<li><a href="<?php echo get_option('home'); ?>/"
     class="on">Home</a></li>
```

```
<?php wp_list_categories('orderby=name&exlude=181&title_li=');
$this_category = get_category($cat);
if (get_category_children($this_category->cat_ID) != "") {
    echo "<ul>";
    wp_list_categories('orderby=id&show_count=0&title_li=
    &use_desc_for_title=1&child_of='.$this_category->cat_ID);
    echo "</ul>";
    }
?>
</ul>
```

Part 2: The CSS

In modern drop-down menus, CSS are a very important part. Indeed, in this example it is CSS that display our menus horizontally.

Paste the following code in your `style.css` file:

```
#nav2{
    background-color: #202020;
    display: block;
    font-size:1.1em;
    height:50px;
    width:100%;
}

#nav2, #nav2 ul {
    line-height: 1;
    list-style: none;
}

#nav2 a ,#nav2 a:hover{
    border:none;
    display: block;
    text-decoration: none;
}

#nav2 li {
    float: left;
    list-style:none;
}

#nav2 a,#nav2 a:visited {
    color:#109dd0;
    display:block;
    font-weight:bold;
```

```
    padding:6px 12px;
}

#nav2 a:hover, #nav2 a:active {
    color:#fff;
    text-decoration:none
}

#nav2 li ul {
    border-bottom: 1px solid #a9a9a9;
    height: auto;
    left: -999em;
    position: absolute;
    width: 900px;
    z-index:999;
}

#nav2 li li {
    width: auto;
}

#nav2 li li a,#nav2 li li a:visited {
    color:#109dd0;
    font-weight:normal;
    font-size:0.9em;
}

#nav2 li li a:hover,#nav2 li li a:active {
    color:#fff;
}

#nav2 li:hover ul, #nav2 li li:hover ul, #nav2 li li li:hover ul,
    #nav2 li.sfhover ul, #nav2 li li.sfhover ul, #nav2 li li li.
    sfhover ul {
    left: 30px;
}
```

Once you have added theses lines to your `style.css` file and saved it, your WordPress blog will feature a very cool horizontal menu for displaying your categories.

Part 3: (Optional) JavaScript

As usual, if you want to maintain backward compatibility with Internet Explorer 6, you'll have to use the Javascript code that you have already seen in the previous example.

Adding a breadcrumb to your theme

When you're looking for a way to improve your blog's usability, a breadcrumb is definitely an option to consider.

According to Wikipedia,

> *Breadcrumbs typically appear horizontally across the top of a webpage, usually below any title bars or headers. They provide links back to each previous page that the user navigated through in order to get to the current page, for hierarchical structures usually the parent pages of the current one. Breadcrumbs provide a trail for the user to follow back to the starting/entry point of a website. Generally, a greater than (>) is used as hierarchy separator, although other glyphs can be used to represent this.*

Getting ready

There are many different solutions available in order to implement a breadcrumb on your WordPress blog, such as hacks and plugins. In this recipe, I'm going to show you how to use the Yoast Breadcrumb plugin, that in my opinion, is the best solution available. But don't worry if you're a hack fanatic, or if you're just curious to know about how breadcrumbs work; I'll also be explaining how to create a breadcrumb without using a plugin.

How to do it

Using Yoast Breadcrumbs is definitely easy. Follow these simple steps to install and configure it on your own blog:

1. Go to `http://yoast.com/wordpress/breadcrumbs/` and download the plugin. In this example, I have used the 0.7.4 version of the plugin.
2. Follow the standard plugin installation procedure, as described in Chapter 4, to install and activate the Yoast Breadcrumb plugin.
3. Once the plugin is activated, you have to insert a code snippet on your blog. Open the files where you want your Breadcrumb to appear (in my opinion, it is a good thing to enable the breadcrumb at least in `single.php`, `page.php`, `search.php`; but you can also enable it in some custom pages that you may have on your blog, such as your archive page or author pages).

4. Paste the following code where you want the breadcrumb to be displayed:

```
<?php if ( function_exists('yoast_breadcrumb') ) {
    yoast_breadcrumb('<p id="breadcrumbs">','</p>');
} ?>
```

5. Save the files and visit your blog. Breadcrumbs are now displayed. Looks very nice, doesn't it? Note that if you want to style your breadcrumb with CSS, you can do so by using the `p#breadcrumb` ID.

The Yoast Breadcrumb plugin allows you to print your breadcrumb directly on your blog page or post. You may also want to get the result only as a PHP variable for further tweaking.

Getting the breadcrumb as a PHP variable

The following code will create a `$breadcrumb` variable that will contain your breadcrumb:

```
<?php if ( function_exists('yoast_breadcrumb') ) {
    $breadcrumbs = yoast_breadcrumb("","",false);
} ?>
```

How it works

The Yoast Breadcrumb plugin takes three arguments. Let's see in detail what these are:

- `$prefix`: The code that your breadcrumb should be prefixed with. Defaults to an empty string.
- `$suffix`: The code that should be added on the back of your breadcrumb. Defaults to an empty string.
- `$display`: If set to `false`, will return the breadcrumb path instead of echoing it. Defaults to `true`.

The Yoast Breadcrumbs plugin works by getting information from your blog, such as the post (or page) name, the category where the post belongs to, and the blog homepage URL. This can be seen in the following example:

```
Home > Category name > Sub-Category name (if any) > Post name
```

The Yoast Breadcrumbs plugin will then dynamically create a clickable breadcrumb, which allows visitors to click on any of the links to go on to the selected section of your blog.

For example, if a visitor clicks on the **Category name** link in the above Breadcrumb, he or she will be redirected to that category page. Clicking on **Home** will lead him or her to your blog homepage.

There's more...

As we have already discussed, it can sometimes be better to use a hack instead of a plugin. Now that we saw how to easily add a breadcrumb to your WordPress blog by using the excellent "Yoast breadcrumbs" plugin by Joost de Valk, let's see how we can create a breadcrumb on our own.

Using a hack to display breadcrumbs

One more time, WordPress conditional tags will be very useful. With them, we'll be able to know easily if the page displayed by a visitor is an article, a page, or a category archive.

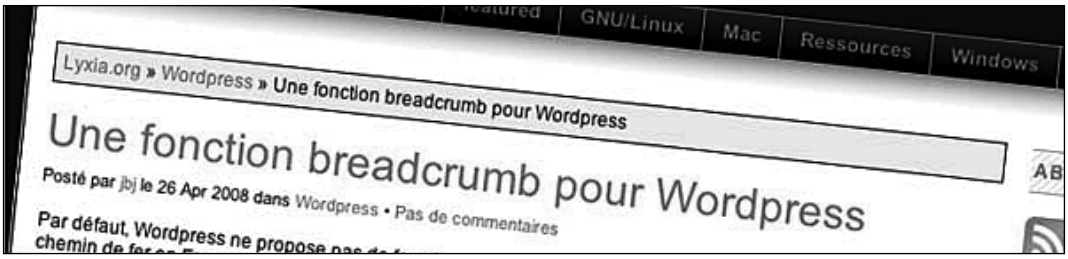
Then, we'll have to use the right functions to show the site's hierarchy. Nothing difficult here, WordPress has all of the functions that we need to get links to the homepage, articles, and single pages.

Simply paste this code in the `functions.php` file of your theme:

```
function the_breadcrumb() {
    if (!is_home()) {
        echo '<a href="';
        echo get_option('home');
        echo '">';
        bloginfo('name');
        echo "</a> » ";
        if (is_category() || is_single()) {
            the_category('title_li=');
            if (is_single()) {
                echo " » ";
                the_title();
            }
        } elseif (is_page()) {
            echo the_title();
        }
    }
}
```

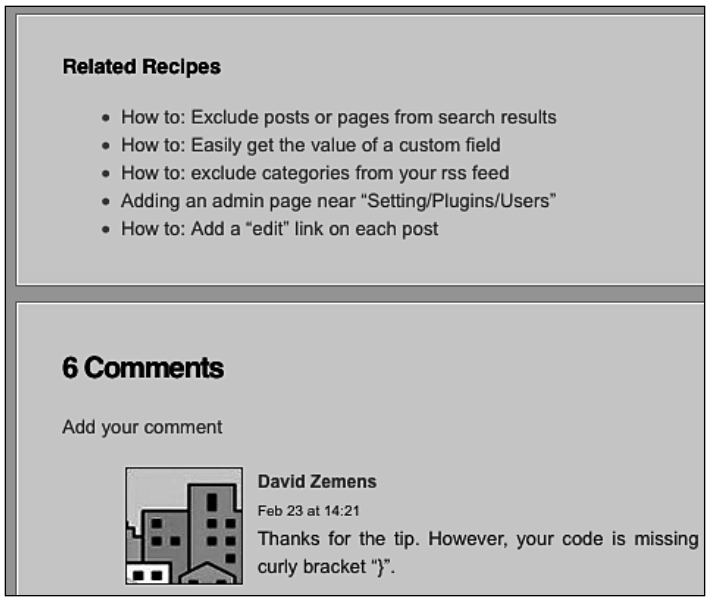
Now, when you'll want to display breadcrumbs, simply use `the_breadcrumb()` function:

```
<?php the_breadcrumb(); ?>
```



Displaying related posts

Guess what your readers will do when they finish reading one of your posts? 90% of them simply leave without even trying to see if you have any interesting related articles. It is a well known fact that the typical Internet user doesn't spend a lot of time on a web site: however, a few tips can increase your chance to have longer visits and maybe new RSS subscriptions. One of them is to display related posts at the end of your articles.



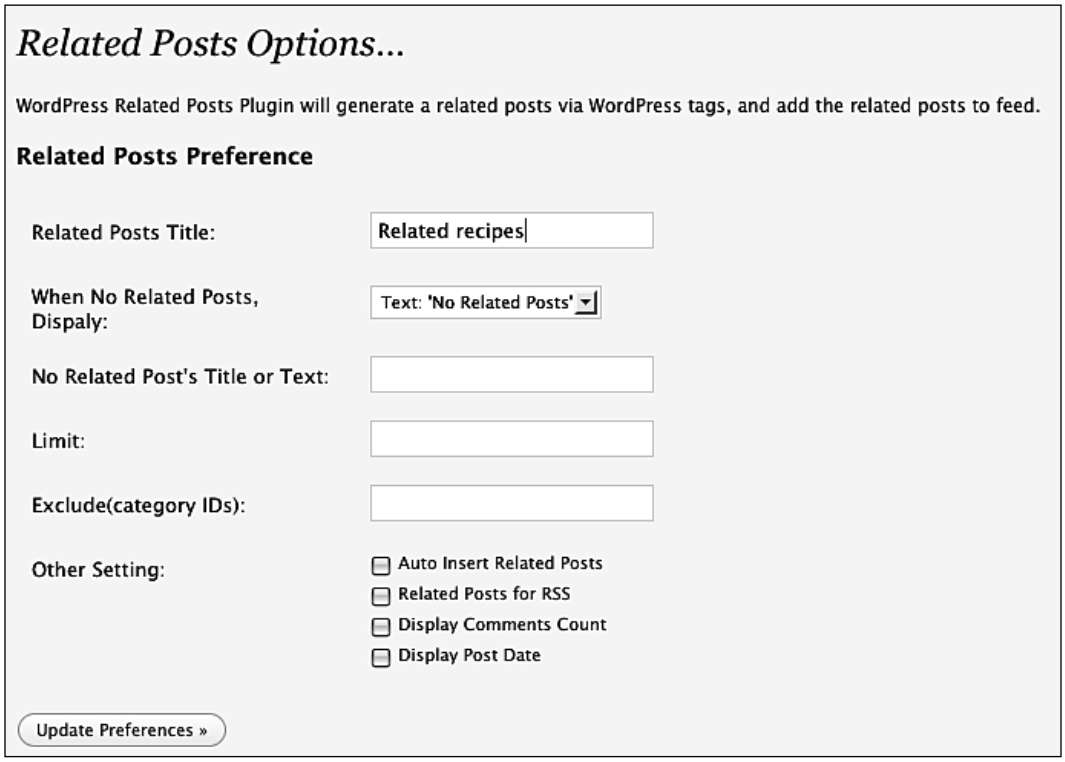
Getting ready

The easiest way to display related posts on your WordPress blog is to use the "WordPress Related Posts" plugin (Version 1.0). This plugin is compatible with WordPress 2.3 and later versions. You can use it with any theme.

How to do it

To display related posts, follow these simple steps:

1. Get your copy of the "WordPress Related Posts" at the following URL `http://wordpress.org/extend/plugins/wordpress-23-related-posts-plugin/`.
2. Install the plugin by following the standard plugin installation procedure as described in Chapter 4.
3. Once the plugin is installed successfully, login to your WordPress dashboard and go to **Settings | WordPress Related Posts** to configure the plugin.



Available options are:

- **Related Posts Title**: Enter **Related posts** or any other title in this tab
- **No Related Posts, Display**: What to display if no related posts are found
- **No Related Post's Title or Text**: Text to be displayed if no related posts are found

- **Limit**: Limit of posts to be displayed simultaneously
- **Exclude (category IDs)**: Categories to be excluded
- **Other Setting**:
  - **Auto Insert Related Posts**: Automatic insertion of related posts
  - **Related Posts for RSS feeds**: Display related posts in your RSS feed
  - **Display Comments Count**: Display the number of comments
  - **Display Posts Date**: Display the date of the posts

4. After you have configured the plugin to work the way you want, simply open your `single.php` file and insert the following line of code where you want your related posts to be displayed:

```
<?php wp_related_posts(); ?>
```

How it works

The "WordPress Related Posts" plugin executes some SQL queries based on tags. For example, if you tag a post with the `cats` tag and have two other posts tagged with `cats`, you can be sure that these two posts will be shown as **related** in the first post.

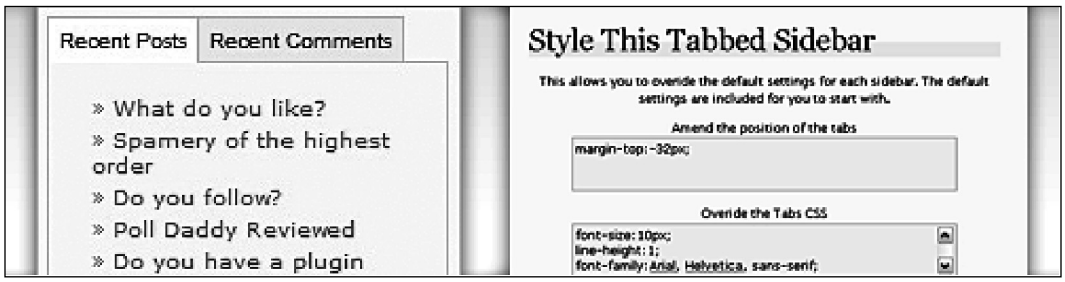
If the plugin doesn't seem to work properly, the first thing to check would be to make sure that your posts are tagged.

Display tabs on your sidebar

There are a lot of things that you can add in your blog sidebar such as Blogroll, Feedburner suscribers, categories, 125*125 pixels ads, and more. The problem is, your sidebar becomes quite lengthy which isn't visually appealing. And to make it worse, not too many people will scroll down your blog to see what is available at the very bottom of your sidebar.

Getting ready

You can create a tabbed sidebar by using some custom HTML, CSS, and JavaScript codes. But the easiest solution is to use the "Fun With Sidebar Tabs" plugin, by Andrew Rickmann. The latest version of the plugin is 0.5.4. The plugin author has stopped maintaining it, but the plugin has been tested for its compatibility with WordPress versions up to 2.8. So, there's no particular reason that the plugin will stop working properly for higher versions, but there's no guarantee either.



How to do it

The "Fun With Sidebar Tabs" plugin can be installed easily. Just follow these steps to get started.

1. Go to `http://wordpress.org/extend/plugins/fun-with-sidebar-tabs/` and download your copy of the plugin.
2. Install it on your blog by following the standard plugin installation procedure as described in Chapter 4.
3. Once the plugin is installed and activated, go to **Appearance | Widgets** on your WordPress dashboard. You can then use either the **Fun with Sidebars** widget (by adding each **Tabbed Sidebar** widget into your main sidebar) or you can simply insert the following tag anywhere in your theme files:

```
<?php the_tabbed_sidebar(1); ?>
```

How it works

The "Fun With Sidebar Tabs" plugin adds a new widget-ready sidebar which displays the widgets in it as tabs, instead of a list. It adds a widget and a template tag, so that it can be inserted wherever it needs to be.

Where to buy this book

You can buy WordPress 2.7 Cookbook from the Packt Publishing website:
`http://www.packtpub.com/wordpress-2-7-cookbook/book`

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com