# Wordpress IV

# Create your own look

## Instructor:  Don Bremer

***Presented and co-sponsored by:***

11 East Superior Street, Suite 210
Duluth, MN 55802
218-726-7298 (main)
888-387-4594 (toll free)
[www.umdced.com](http://www.umdced.com)
[ced@umdced.com](mailto:ced@umdced.com)

## Contents

## Basic Rules

### Rule 1 – Close everything in the order that you open them.

The right way to close:

```
<ul>
        <li>
        </li>
</ul>
```

The wrong way to close:

```
<ul>
        <li>
        </ul>
</li>
```

### Rule 2 – Every theme has at least two files – style.css and index.php

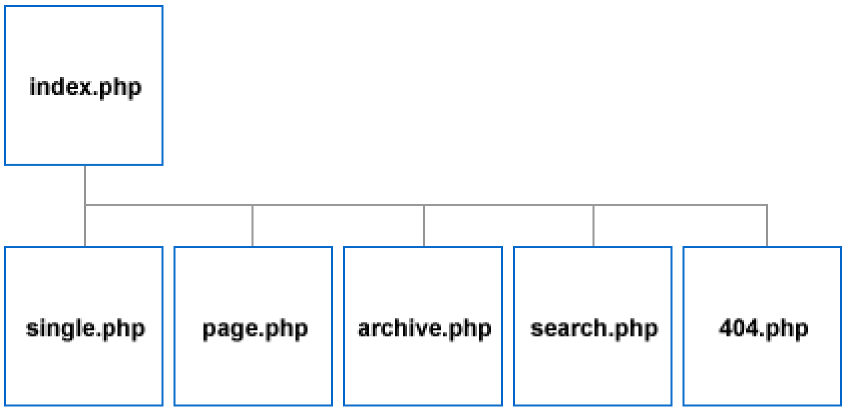
You tell your theme where everything goes within index.php and how everything should look like within style.css.

A complete list of files are as follows:

| | | | |
|---|---|---|---|
| style.css | index.php | home.php | single.php |
| page.php | archive.php | category.php | search.php |
| 404.php | comments.php | comments-popup.php | author.php |
| | | date.php | |

### Heirarchy

The diagram below simply shows you what the WordPress system will look for, in case one of your theme file is missing. I listed only six files, instead of thirteen in the diagram because those are the main ones that you should focus on.

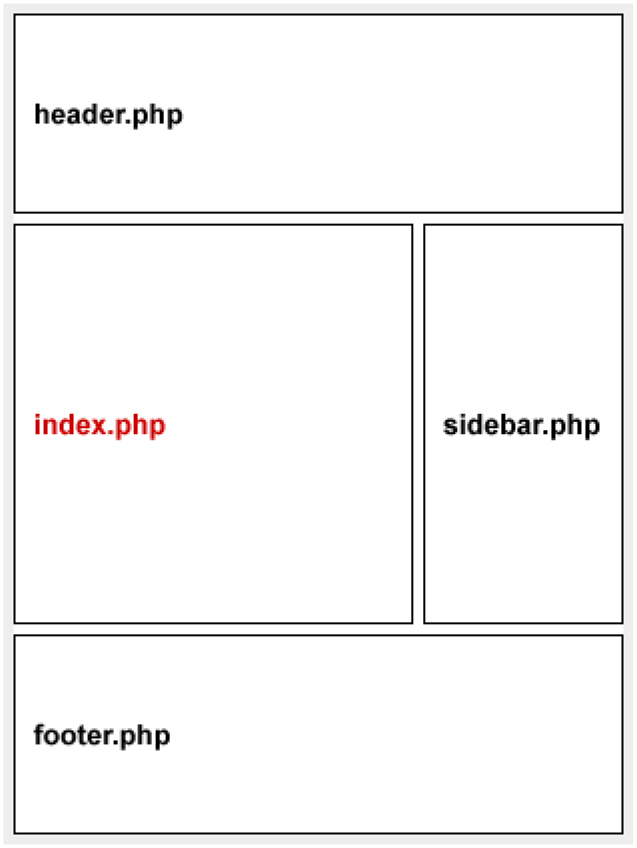


T he level of importance of each file by its position. Top, left is most important. Bottom, right is least important. Hierarchy or levels of importance exists for template files because if the archive.php file, which handles the look for archive pages, is missing, then WordPress will point to the index.php to control how the archive page will look like.

If the single.php template file is missing, which template file does it look for to display a single post view? It looks for index.php.

## Templates

The one thing to remember is that **each page** of your blog **is made up of multiple template files**. Here's an example of the front page.



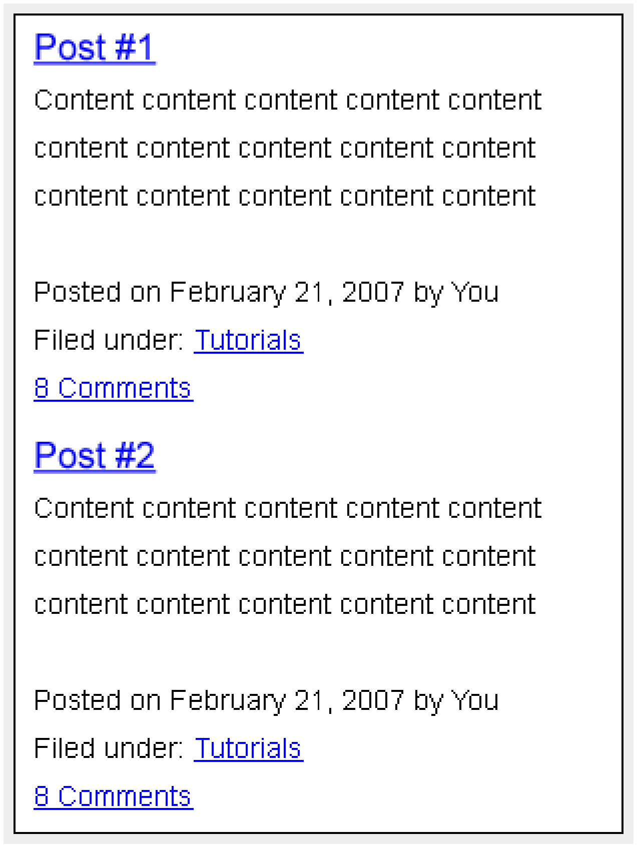
In this example, your front page is made up of four template files: **header**, **index**, **sidebar**, and **footer**.

**Header Template File:**

The *blog's title* and *description* sit in this file. They usually remain the same throughout the whole blog.

## Index Template File

This contains the *post title*, *content* (text and images of each entry), and *post meta data*(information about each post like: who's the author, when you published it, under which categories, and how many comments).


Post #1
Content content content content content content content content content content content content content content content

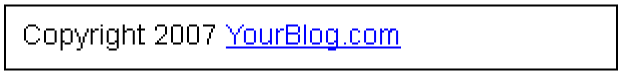
Posted on February 21, 2007 by You
Filed under: Tutorials
8 Comments

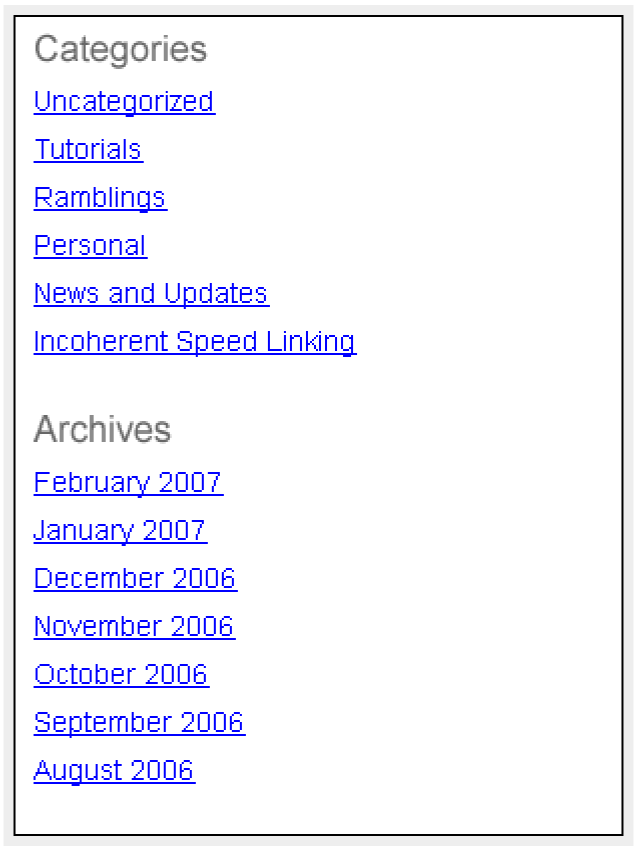
Post #2
Content content content content content content content content content content content content content content content

Posted on February 21, 2007 by You
Filed under: Tutorials
8 Comments

## Footer Template File

Like the header.php template file, the footer usually doesn't change from page to page. Anything can be put in this file, but usually your *copyright information*.


Copyright 2007 YourBlog.com


Categories
Uncategorized
Tutorials
Ramblings
Personal
News and Updates
Incoherent Speed Linking

Archives
February 2007
January 2007
December 2006
November 2006
October 2006
September 2006
August 2006

## Sidebar Template File

This controls the *Page links listing*, *category links listing*, *archive links listing*, *blogroll listing*, etc. (The Sidebar doesn't need to be on the right side of the layout.)

## Why is index red?

Index.php is red to indicate that, that area will change, depending on which page of the blog you're on.

If you were on a single post page, your page would consist of these four template files: header, **single**, sidebar, and footer.

## Writing the files

### Create a “theme” folder

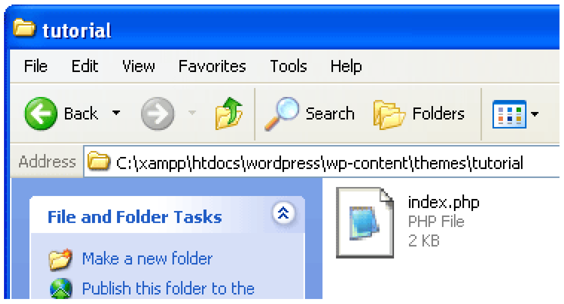
Go to your wordpress themes folder. Create a new folder. Name it **tutorial**.

### Create index.php and style.css files

Open up Notepad or the text editor of your choice. Notepad is at **Start > Programs > Accessories > Notepad**.

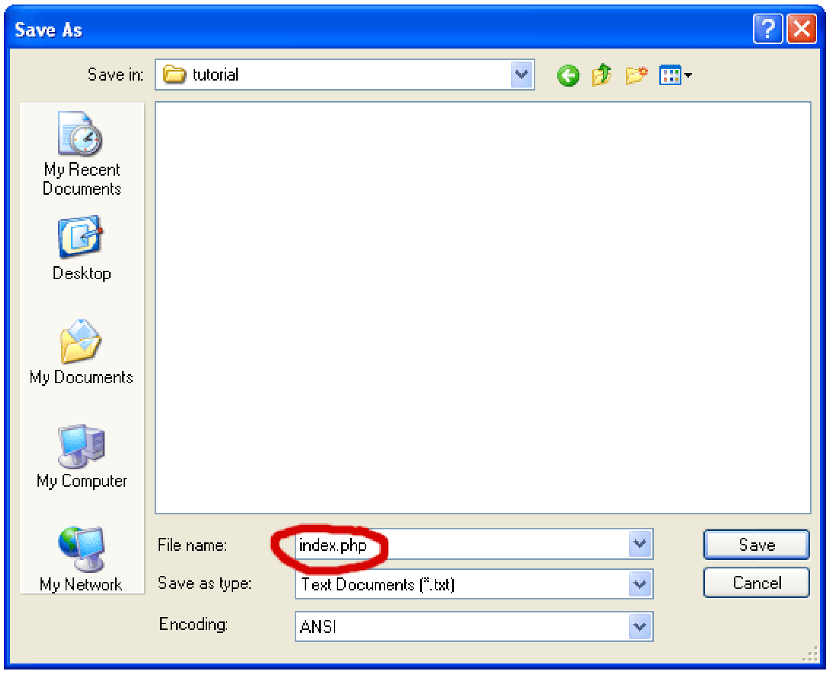
Copy and paste everything from index.txt in the Wordpress IV file in the My Documents folder to your Notepad window.

Save your notepad in the tutorial folder as **index.php**



Address C:\xampp\htdocs\wordpress\wp-content\themes\tutorial

index.php
PHP File
2 KB



Save in: tutorial

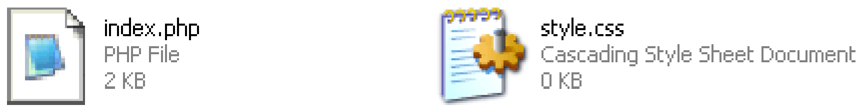
File name: index.php

Save as type: Text Documents (*.txt)

Encoding: ANSI

Open another notepad. Leave it empty. Save the empty notepad as **style.css**, in the same folder. Close the style.css notepad.

So now there are two files: index.php and style.css.



index.php
PHP File
2 KB

style.css
Cascading Style Sheet Document
0 KB

Open the index.php. Below tells what each section does.

**<html>** is where the web page starts.

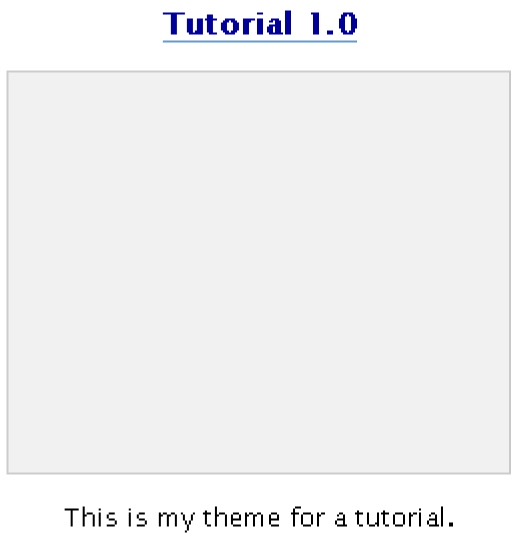
**<head>** is where the head of the web page starts. Every web page has a head and a body.
**</head>** is where the head ends.

**<?php bloginfo('stylesheet_url'); ?>** is a PHP function that calls for the location of the **style.css** file so the theme can link to it and style everything on my pages. Anytime codes are wrapped in**<?php** and **?>**, it's PHP and it's different from the rest of my codes. In PHP, **<?php** is start and **?>** is end.
So:

- **<?php** - start PHP
- **bloginfo('stylesheet_url')** - call for the location of style.css
- **;** - stop calling for style.css. The semicolon is one way of closing a set of codes within PHP.
- **?>** - end PHP

**<body>** - This is where the body starts. The body is everything that is seen and read on a web page.
**</body>** is where the body ends.

**</html>** is where the web page ends. Nothing else after that.


Tutorial 1.0

This is my theme for a tutorial.

Copy and paste everything from style.txt in the Wordpress IV file in the My Documents folder to the style.css file.

## Setting up the theme

Open up a browser.

Type: **http://localhost/wordpress/wp-login.php**. Go to it and login into your WordPress administration area.

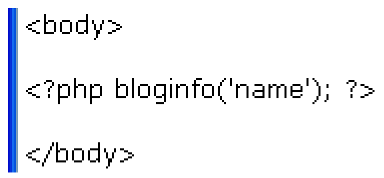
Notice, your theme does not have a thumbnail screenshot yet. The box is empty. Once activated, WordPress will tell you.

Now open up a new browser or tab (if your browser has tab browsing) and go to**http://localhost/wordpress**. There should then get a blank page. I mean *completely* blank. If it isn't blank, you're at the wrong page.

## Adding some meat to the Tutorial Theme

Let's add some information to the theme. Go to the file index.php and open it in Notepad if not already open. Type:

**<?php bloginfo('name'); ?>** in between the **<body>** and **</body>** tags. Now save it.



```
<body>

<?php bloginfo('name'); ?>

</body>
```

Go back to the browser and refresh it. You should see the blog's title.

**What just happened?**
One line of PHP codes to your index.php within the body area of your web page. That one line called for your blog's information, hence, **bloginfo()**. Specifically, it called for the name of your blog. This is the name that you have as **Weblog Title** on your Options page.

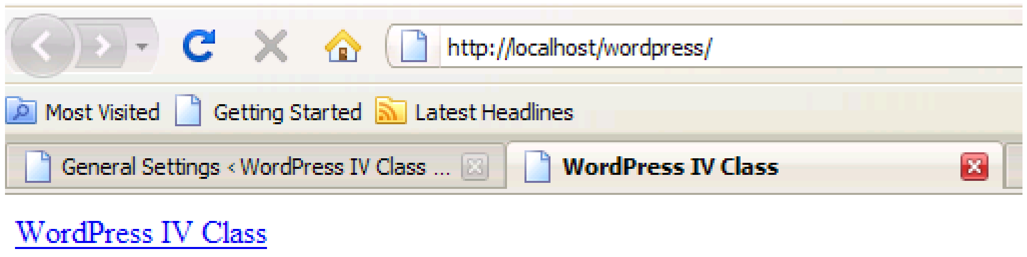
**<?php** - start PHP
**bloginfo('name')** - call for blog's information, specifically blog's title.
**;** - stop calling for blog's information
**?>** - end PHP

Each time you add or change anything in your index.php file, save, then refresh your page to see the change.

**Step 4**:
To turn your blog's title into a text link. It requires an XHTML tag. Go back to your index.php file.

Add **<a href="#">** and **</a>** to the same line. Your new line of codes should be:
**<a href="#"><?php bloginfo('name'); ?></a>**

Go back to the browser, refresh and you should see your title turned into a link.


http://localhost/wordpress/
Most Visited
Getting Started
Latest Headlines
General Settings ‹ WordPress IV Class ...
WordPress IV Class

WordPress IV Class

Now, it's a link, but it's a link that points to nowhere. Since it is your weblog's title, you want to make it link to your front page. To do that, type:

**<?php bloginfo('url'); ?>** in between the quotes of **href=**

Save it, now you have:

**<a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?></a>**

Go back to the browser and refresh. When you hover on the link now, the status bar should display **http://localhost/wordpress**

**What just happened?**
You turned your website's title into a link and pointed the link to your front page or blog's home page.

**bloginfo(’url’)** - calls for the your blog’s information, speficially the address or URL of the front page
**<a>** - is an XHTML tag for opening a link
**</a>** - is the closing tag for a link. Otherwise, your web page will not know where to end the link and the rest of your page will be one big link. Remember part of **rule #1**? Close everything you open.
**href=”"** - is short for hypertext value. Whatever’s in between the quotation marks of href=”" is the value
In words, the codes:

**<a href=”<?php bloginfo(’url’); ?>”><?php bloginfo(’name’); ?></a>**

means: I’m starting a link. The value of my link is my blog’s URL; I’m using the PHP function **bloginfo(’url’)** to call for the address/URL. The name of the link is my blog’s title; I’m using the PHP function **bloginfo(’name’)** to call for my blog’s title. Close my link.

Call for your blog’s description, type **<?php bloginfo(’description’); ?>** right under the codes for the title link. Now you have:

**<h1><a href=”<?php bloginfo(’url’); ?>”><?php bloginfo(’name’); ?></a></h1>**
**<?php bloginfo(’description’); ?>**

Save it, refresh your browser to see your blog’s description appear under the title link. You can change your blog’s description in the administration area later.

To find out all the tags you can call in “bloginfo”, check out the Wordpress codex at:
http://codex.wordpress.org/Template_Tags/bloginfo

## The DIV Tag

Type **<div>** and **</div>** around everything:

```
<div>
 <h1>
  <a href=”?php bloginfo(’url’); ?>”><?php bloginfo(’name’); ?> </a>
 </h1>
<?php bloginfo(’description’); ?>
</div>
```

Save it, refresh your browser. You should see no changes.

Think of **DIV** as an invisible box. It’s there to separate your **blog’s title link** and **blog’s description** from everything else. If you don’t style it, it does nothing other than separate content. Later on, you’ll use the **style.css** file to style your invisible box. You can give DIV **borders**, **paddings**, **margins**, **background color**, **background images**, etc.

Add **id=”header”** to the DIV tag, like this:

```
<div id=”header”>
```

Save your notepad, refresh your browser.

This change also does nothing, for now. We assign an **ID** to the **DIV** tag because there will be more DIVs or invisible boxes. You need a way to separate one invisible box from another right?

## The Loop

**The Loop** calls for your blog's entries. It's the most important set of PHP codes.

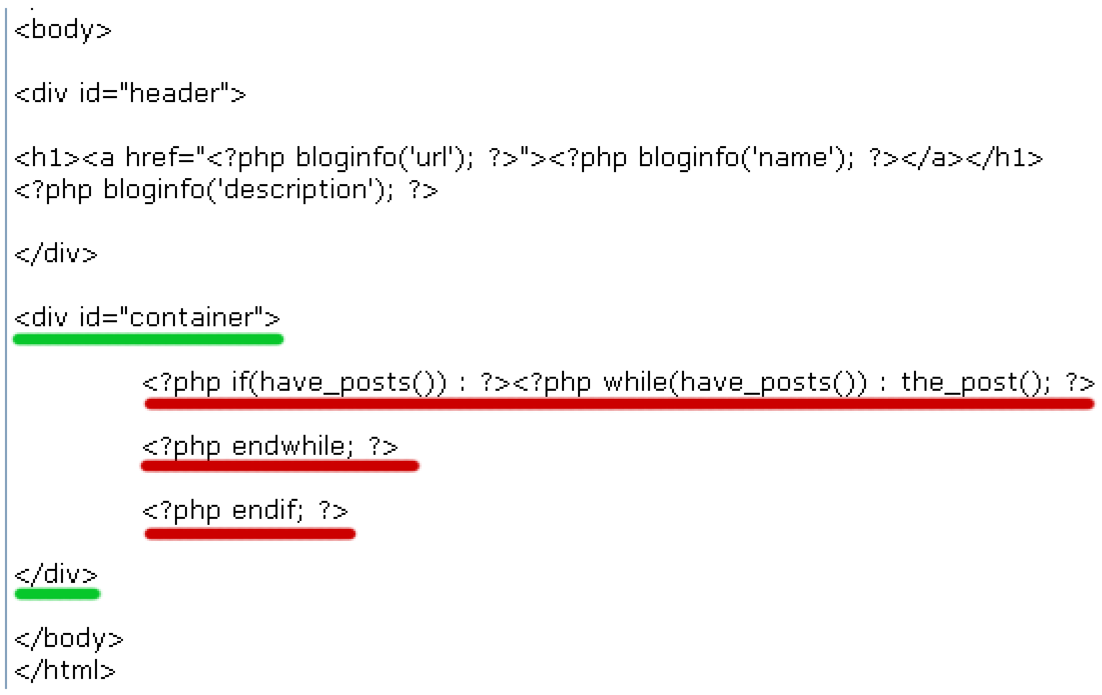
Add a DIV (invisible box) under the header area. Give it "container" for ID, like this:

```
<div id="container">
</div>
```

What is this DIV for? It's for separating your content from everything else that comes after it, stuff like the sidebar and footer.

Add the following codes between the Container DIV tags. Try to type this out exactly like you see it. If you get any errors, retype it.

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



```
<body>

<div id="header">

<h1><a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?></a></h1>
<?php bloginfo('description'); ?>

</div>

<div id="container">

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

        <?php endwhile; ?>

        <?php endif; ?>

</div>

</body>
</html>
```

**What just happened?**

- **if(have_posts())** - checks to see **if** you **have** any **post**.
- **while(have_posts())** - if you do have it, **while** you **have** any **post**, execute the_post().
- **the_post()** - call for the posts to be displayed.
- **endwhile;** - sticking to rule #1, this is to close **while()**
- **endif;** - close **if()**
- **Note**: not every set of codes need two parts in order to open and close itself. Some can self close, which explains **have_posts()** and **the_post();**. Because **the_post();** sits outside of **if()** and **while()**, it needs its own semicolon to end/close itself.

You learned how to call for your blog's title by usingbloginfo('name'). Now, you learn how to call for the post titles in between The Loop.

Type `<?php the_title(); ?>` after the_post(); ?> and before `<?php endwhile; ?>`

Now turn the titles into post title links. Remember how you turned the blog title into a link? Same thing here, add `<a href="#">` and `</a>` around `<?php the_title(); ?>`

Save index.php and refresh browser. Now the titles are title links, but they point to nowhere. To make each title point to the right post, the pound sign # needs to be replaced with the_permalink().

```
<a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
```

`the_permalink()` is the PHP function that calls for the address or location of each post. Save and refresh the browser. If you have only one Hello World title, hover over that link, look at the status bar, at the bottom of your browser; it's no longer http://localhost/wordpress/#.

If you have more than one title link, you'll see that each title links to a different post or web page. But, our post title links are still on the same line. To separate them, add the <h2> and </h2>tags around your link codes.

```
<h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
```

Remember H1 that we used for the blog title? That's your web page's heading. H2 is used for sub-headings. Now that your title links are sub-headings, each gets its own line. Save index.php and refresh the browser to see the change.

```
<body>

<div id="header">
	<a href='<?php bloginfo('url'); ?>'><?php bloginfo('name');
?></a><br>
	<?php bloginfo('description'); ?>
</div>
```

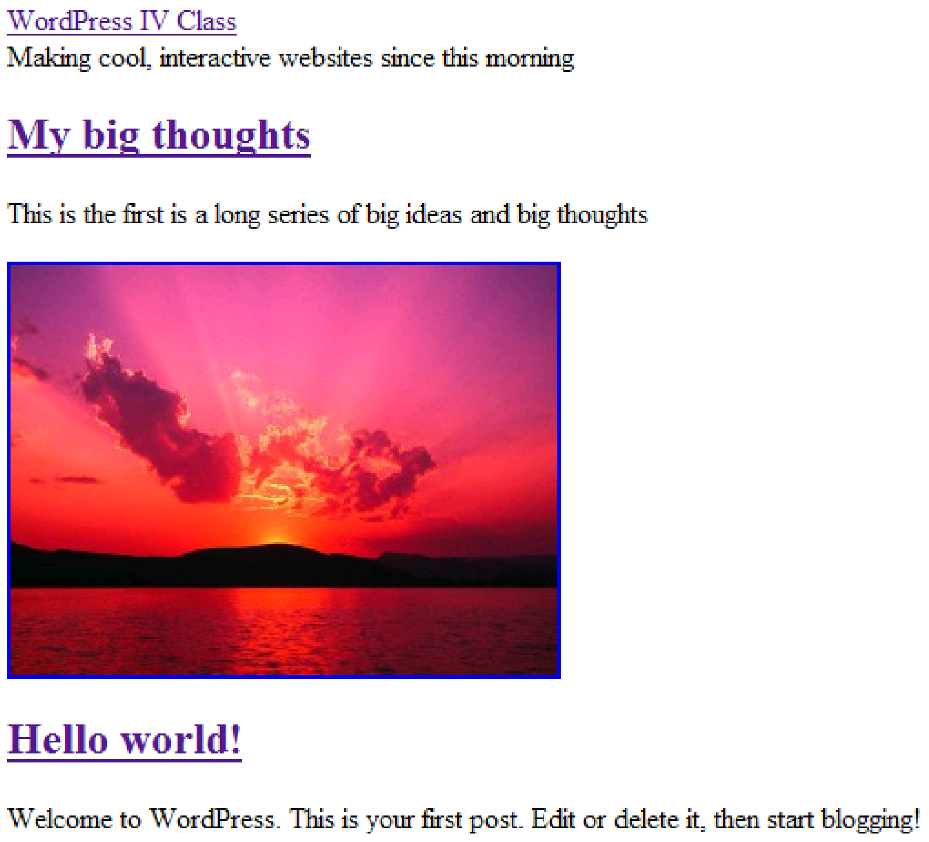
```
<div id="container">

     <?php if(have_posts()) : ?>
          <?php while(have_posts()) : the_post(); ?>
               <H2><a href="<?php the_permalink(); ?>"><?php
the_title(); ?></A></H2>
          <?php endwhile; ?>
     <?php endif; ?>

</div>

</body>
</html>
```

Write a few new posts – how do they look?

## The Content

Type `<?php the_content(); ?>` under the post title codes.

Save the notepad and refresh your browser, and there should be some texts under your post titles.



WordPress IV Class
Making cool, interactive websites since this morning

My big thoughts

This is the first is a long series of big ideas and big thoughts

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

**What just happened and why does it look like that?**
You used the PHP function **the_content()** to call for **post entries** (content). Right now, the content is just one long line of text, all the way to the window's right side, because you haven't styled it yet. Remember the **style.css** file? Later on, we'll use that file to control how everything looks.

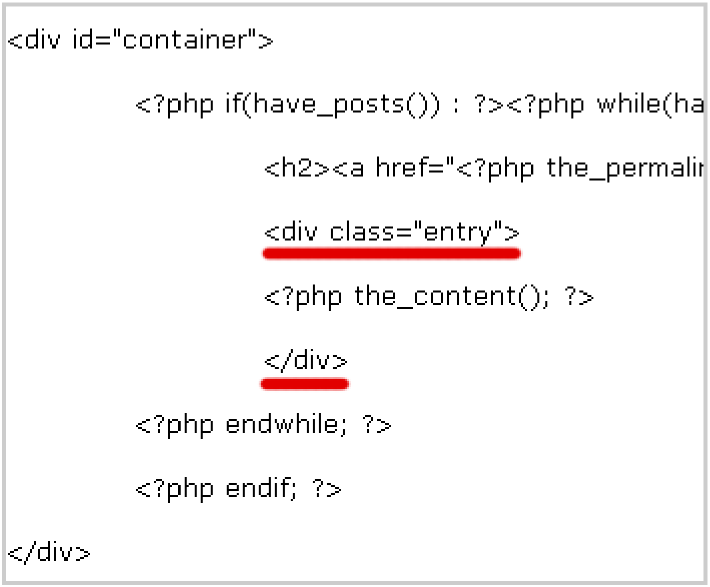
Go back to your browser, click on **View** and select **Page Source** or **Source**. A window of codes will pop up. If you're using **Internet Explorer** then a notepad will pop up.

**P tags, Why and How?**
**Why** - While typing your entries, each time you skip a line is a paragraph. You need a way to show that right? Each paragraph sits inside a set of **P** (paragraph) tags. That's how you're getting the spacing in between paragraphs.

**How** - Easy, the WordPress template system generate the **P** tags for you.

Wrap an invisible box (DIV) around **the_content()** and give it **class="entry"** like this:

```
<div class="entry">
</div>
```



```
<div id="container">

        <?php if(have_posts()) : ?><?php while(ha

                <h2><a href="<?php the_permalir

                <div class="entry">

                <?php the_content(); ?>

                </div>

        <?php endwhile; ?>

        <?php endif; ?>

</div>
```

Save and refresh your browser, if you go to **Viev > Page Source** again, you'll see the class="entry" DIV tags wrapped around each of your **post entries**.

**Why?**
First reason, now you can tell where the **post titles** end and where the **post entries** begin. Second reason, it's for styling with the **style.css** file. If you want to, you'll be able style your post entries without affecting everything else.

What is the difference between **id** and **class**?
So far, for every invisible box or DIV, you've used **id** to name it. Remember **id="header"**? So what's the difference? **id** is unique and **class** isn't. If you look through your source codes, there's only one **id="header"** and there's only one **id="container"**, but there are multiple**class="entry"**.

Can **header** and **container** be **class**es instead of **id**s? Sure.

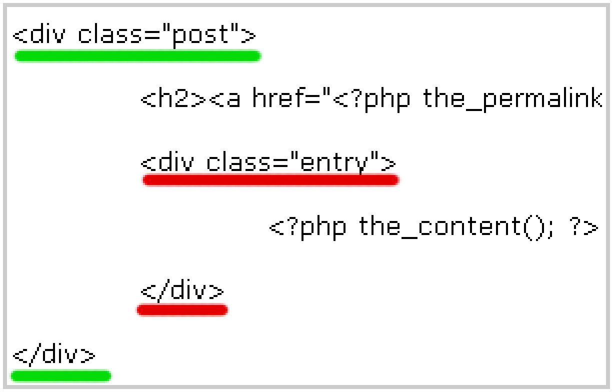
Keep this in mind; you cannot repeat any **id**. For example, you can't have two **id="header"** on the same page. When you want to reuse something over and over again like the **post entries**, use **class**.

Wrap a DIV around your post titles and post entries. Name it **class="post"**

```
<div class="post">
</div>
```

(The names for classes and IDs can be anything you want them to be. You can name them after… your favorite foods, but **post** and **entry** are short, simple, and easier to remember right?)

So now you have:

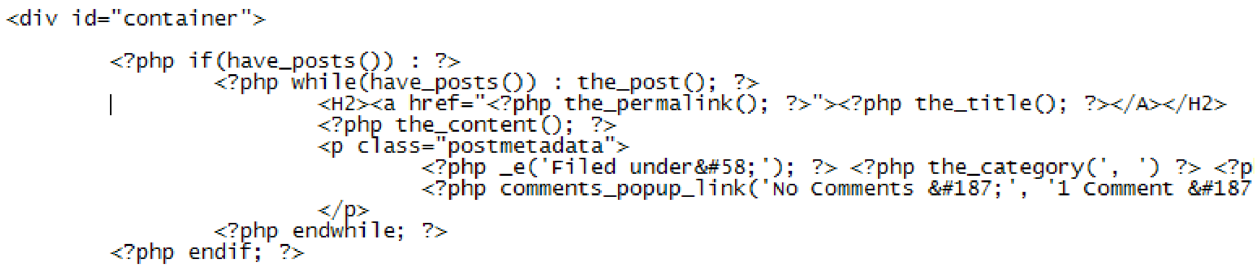


```
<div class="post">

        <h2><a href="<?php the_permalink

        <div class="entry">

                <?php the_content(); ?>

        </div>

</div>
```

## Postmetadata

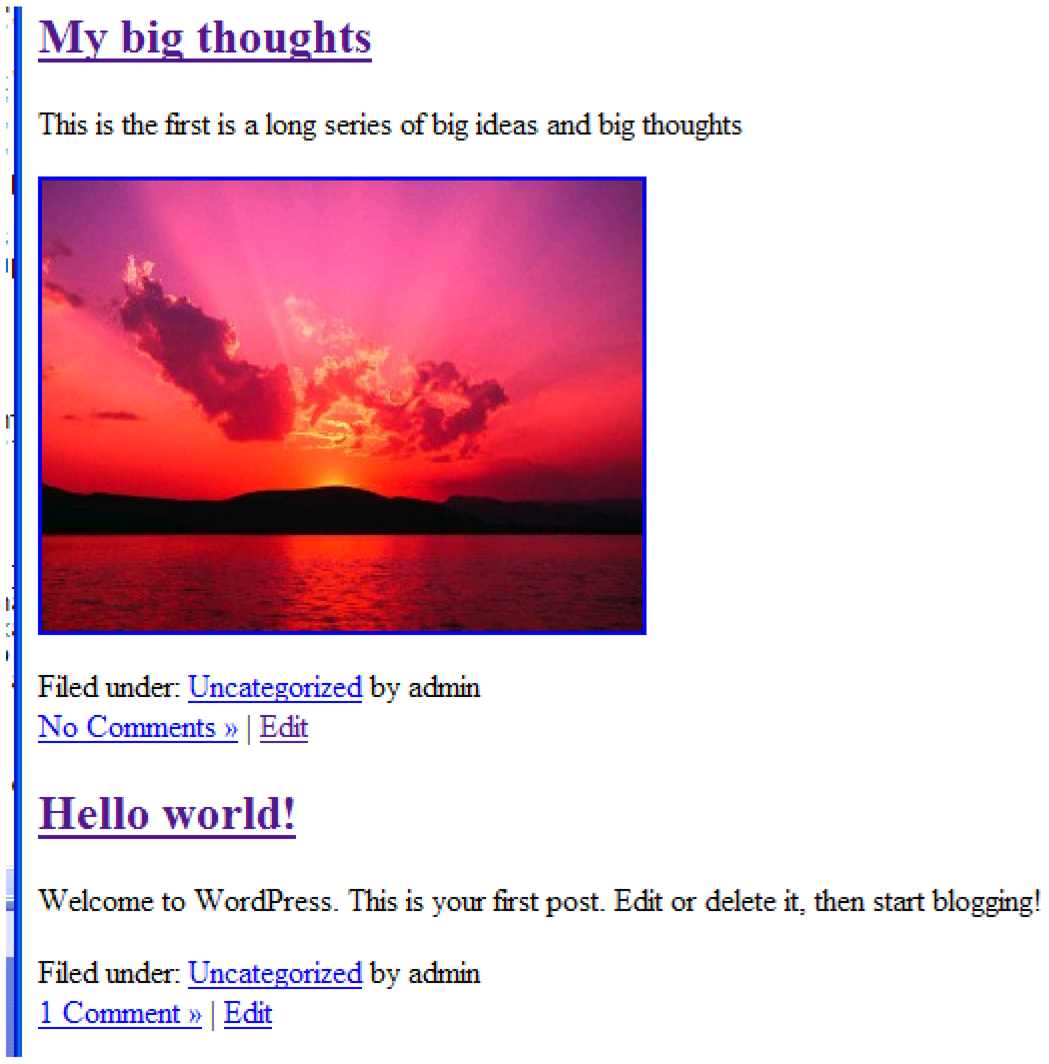
Now, we tackle the postmetadata: **date**, **categories**,**author**, **number of comments**, and any miscellaneous information attached to each post.

Copy and paste the codes in postmetadata.txt (Under Wordpress IV in My Documents) under `<?php the_content(); ?>`.



```
<div id="container">

        <?php if(have_posts()) : ?>
                <?php while(have_posts()) : the_post(); ?>
|                       <H2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></A></H2>
                        <?php the_content(); ?>
                        <p class="postmetadata">
                                <?php _e('Filed under&#58;'); ?> <?php the_category(', ') ?> <?p
                                <?php comments_popup_link('No Comments &#187;', '1 Comment &#187
                        </p>
                <?php endwhile; ?>
        <?php endif; ?>
```

Save it and refresh your browser, you should see:



My big thoughts

This is the first is a long series of big ideas and big thoughts

Filed under: Uncategorized by admin
No Comments » | Edit

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

Filed under: Uncategorized by admin
1 Comment » | Edit

**Explanations**:
**<p class="postmetadata">** and **</p>** - All the postmetadata information sit in between a set of paragraph tags named **class="postmetadata"** because we should separate postmetadata from the content or post entry. Without the paragraph tags, the postmetadata information will continue where the content left off, without any spacing to differentiate content and postmetadta.

**<?php _e('Filed under&#58;'); ?>** - &amp#58; is the code to call for a colon ":"
Wrapping **<?php _e(' '); ?>** around **Filed under&#58;** is not necessary. You can simply type out **Filed under:**

**<?php the_category(', '); ?>** - the_category() is the PHP function that calls for all the categories that you tagged your post with. If you put **Filed under:** and **the_category()** together. You'd get **Filed under: Name**

**of category 1, Name of category 2**. The comma within the_category() is a way of separating the category names

**<?php _e('by'); ?>** - same as **Filed under:**. If you're creating the theme for personal use, wrapping **_e()** around the word **by** isn't necessary. The **_e()** has to do with creating translatable theme, which is important when you're theme gets used by hundreds of people from various countries. If you're planning on creating themes for public use, then it's best to use _e() just in case you do look into translatable themes later on.

**<?php the_author(); ?>** - self explanatory. It simply prints your name or whoever published the post.

**<br />** - If you want a line break, but don't want the spacing that comes with paragraph tags, use BR. Notice the forward slash /. This is another tag that can self-close.

**<?php comments_popup_link('No Comments &#187;', '1 Comment &#187;', '% Comments &#187;'); ?>** - comments_popup_link() calls for a pop up window of your comments, when popup comment is activated. If popup comment isn't activated, comments_popup_link() simply send you to the list of comments. **No Comments &#187;** is what will be displayed when you have no comments. **1 Comment &#187;** is when you have exactly one comment. **% Comments &187;** is for when you have more than one comment. For example: 8 Comments ». The percent sign % means number. &#187; is a code to display a double arrow.

**<?php edit_post_link('Edit', ' | ', ''); ?>** - This is only visible when you are logged in as an administrator. **edit_post_link()** simply displays an edit link for you to select which post to edit, instead of having to browse through all the posts from the administration panel to search for the one you want to edit. **edit_post_link()** has three sets of single quotes. The first set is for what word you will use for the link title of the edit link. If you use **Edit post**, then it'll say **Edit post** instead of **Edit**. The second set of single quotes is for whatever that comes before the link. In this case, a vertical line **|**; that's what the **&124;** code is for. The third set of single quotes is for whatever that you want to come after the edit link. In this case, nothing comes after it.

## Error Checking

Sometimes you may use a theme in a way that wasn't originally thought of. In this case, you may wish to "bullet-proof" the theme by using **Else**, **post ID**, and **link title value**.

Type the following codes under <?php endwhile; ?>.

```
<?php else : ?>
      <div class="post">
            <h2><?php _e('Not Found'); ?></h2>
      </div>
```

Save it, but you won't notice any difference.

Now that you know what **else** is, what did you tell WordPress to do when you don't have any post or when it can't find any post? You told WordPress to display the error message **Not Found**. That message can be anything you want.

What are the **<div class="post">** and **</div>** for? Well, you don't want your error message to get stranded in the middle of nowhere right? You wrapped each entry within the **<div class="post">** and **</div>** tags. So same thing, although the error message isn't actual content, it is text just like the entries.

## Adding more information to the page

Now add:

```
id="post-<?php the_ID(); ?>"
```

to
<div class="post">

Why use it? It's for customizing the look of posts, individually. Later on, when we use the **style.css** file to tell your theme how the posts will look like, every post will look the same. With a unique **ID** attached to each post, you can target a single post and make it look different from the rest of the posts. Without the IDs, you have no way of differentiating the posts within the **style.css** file.

Now add:

```
title="<?php the_title(); ?>"
```

to the post title link.

**title=""** is another xhtml attribute for the **<a>** (link) tag. Whatever's within the quotes is the description of your link. In this case, the title of each post is also the link description. That's why you used the PHP function **the_title()** again.

If you don't use **the_title()** as a value for **title=""**, then every post title link will have the same description. For example, instead of **the_title()**, you use **title="Click me"**, every post title link will have **Click me** as its description.

## Post Nav Links

At the bottom of most WordPress blogs, there's a **Next Page** or **Previous Page** link. You call for those links by using the **posts_nav_link()** function of the WordPress template system.

Add the following codes between **<?php endwhile; ?>** and **<?php else : ?>**

```
<div class="navigation">
     <?php posts_nav_link(); ?>
</div>
```

**<div class="navigation">** - start an invisible box named navigation to wrap around your Next and Previous links area.
**<?php** - start PHP
**posts_nav_link()** - call for the Next and Previous links.
**;** - stop calling for them.
**?>** - end PHP
**</div>** - close the invisible box named **navigation**.

Save index.php, then refresh your browser to see your own Next or Previous link. By default, if you don't have more than ten posts, nothing will appear. If you don't have more than ten posts, but still would like to see it, login into the administration panel, select **Options > Reading**, then set it to one less than the amount of posts that you have. For example, if you have six posts, set it to five.

**How to customize posts_nav_link()**:
Just like some of the functions in the **postmetadata** lesson, you can give this function three sets of… anything that you want to be displayed in between, before, and after the Next and Previous links. It looks like this:

**<?php posts_nav_link('in between','before','after'); ?>**

The first set of single quotes is for holding anything that you want to appear between the Next and Previous links. The second set of single quotes is for holding anything you want to come before it. The third set is for holding anything that comes after it.

## The Sidebar

Have you been looking forward to the Sidebar? At first glance, the Sidebar looks difficult, but it isn't tricky at all. Once you get used to its structure, you'll be able to code and style it very quickly.

wrap a box with a class named, **sidebar** around everything in the Sidebar. Type this code under the **container** box and above the **</body>** tag:

```
<div class="sidebar">
</div>
```

Start an unordered list within your new sidebar box.

**<ul>** - open unordered list
**</ul>** - close unordered list

Add a list item (**LI**) inside the unordered list (**UL**) and put a sub-heading inside the list item (**LI**).

```
<li><h2><?php _e('Categories'); ?></h2>

</li>
```

**<li>** - open list item
**<h2>** - open sub-heading
**<?php _e('Categories'); ?>** - print the word **Categories**
**</h2>** - close sub-heading
**</li>** - close list item

Add the following codes within the list item:

```
<ul>
<?php wp_list_cats('sort_column=name&optioncount=1&hierarchical=0'); ?>
</ul>
```

Here's what that means:

**<ul>** - open another unordered list
**<?php wp_list_cats(); ?>** - call for the list of category links
**</ul>** - close unordered list

Your default category is **Uncategorized**. If you did not publish under multiple categories, then your list of category links should have only one link, the Uncategorized.

Further explanations:

- **sort_column=name** - list category links alphabetically
- **optioncount=1** - display the number of posts made under each category
- **hierarchial=0** - don't turn sub-categories into sub-list-items, which explains why my **Sub Category** link is listed in the first level of the list.
- **&** - each time you add on another attribute, you have to type **&** before it to separate it from the existing attributes. For example **&** sits in between **sort_column** and**optioncount**.

Why you didn't wrap the **<li>** and **</li>** tags around **<?php wp_list_cats(); ?>**:

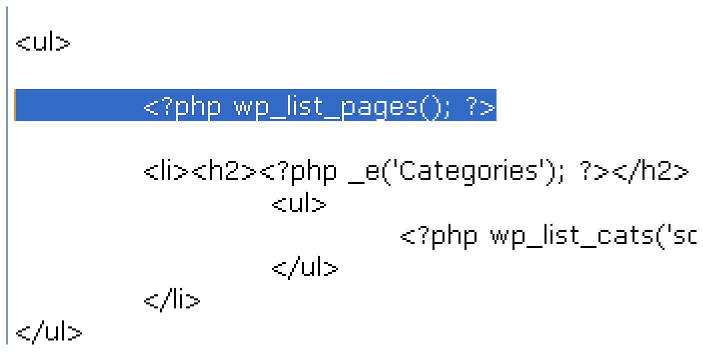
When you call for the category links list using **wp_list_cats()**, it automatically attaches a set of**<li>** and **</li>** (list item) tags around each link. Look at your browser, go to **View > Page Source**or **Source**; after the window pops up, scroll to the bottom to see the codes for the category links list; notice that each link has a set of list item tags around it.

## Page Links

After the regular Sidebar is complete, we will go on to widgetize the sidebar.

Add the following codes on top of the Categories block:

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



```
<ul>
	<?php wp_list_pages(); ?>

	<li><h2><?php _e('Categories'); ?></h2>
		<ul>
			<?php wp_list_cats('sc
		</ul>
	</li>
</ul>
```

Save your file and refresh the browser. By default, you have only one Page link, the About link.

Add more pages and child-pages to the theme blog.

Notice how the Pages heading doesn't match the Categories heading? This can be fixed! Just Add **'title_li=<h2>Pages</h2>'** to **wp_list_pages()**.

## Further Customization to wp_list_pages()

Some layouts/designs cannot handle that many levels of links within the Sidebar. To limit the amount of link levels to list, add the **depth** attribute to **wp_list_pages()** and set it to **3**.

```
<?php wp_list_pages('title_li=depth=3&<h2>Pages</h2>'); ?>
```

## Add an Archive List

Type the following codes in the Sidebar area, under the **Categories** listing:

```
<li><h2><?php _e('Archives'); ?></h2>
<ul>
<?php wp_get_archives('type=monthly'); ?>
</ul>
</li>
```

**What happened?**
You used the **wp_get_arhives()** PHP function with the **type** attribute and **monthly** value to call for the archive links by month.

- **<li>** - open list item
- **<h2>** - open sub-heading
- **<?php _e('Archives'); ?>** - text of the sub-heading
- **</h2>** - close sub-heading
- **<ul>** - open unordered list under the sub-heading, within the list item

- **<?php wp_get_archives('type=monthly'); ?>** - call for archive links by month, nest each link within **<li>** and **</li>** tags. If you check your source codes (View > Page Source). You'll see that **wp_get_archives()** generated list item (**LI**) tags for each link, just like the**wp_list_cats()** function.
- **</ul>** - close the unordered list sitting under the sub-heading
- **</li>** - close list item

## Add blogroll links

Type the following codes under the Archives link listing:

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

Save and refresh to review the result.

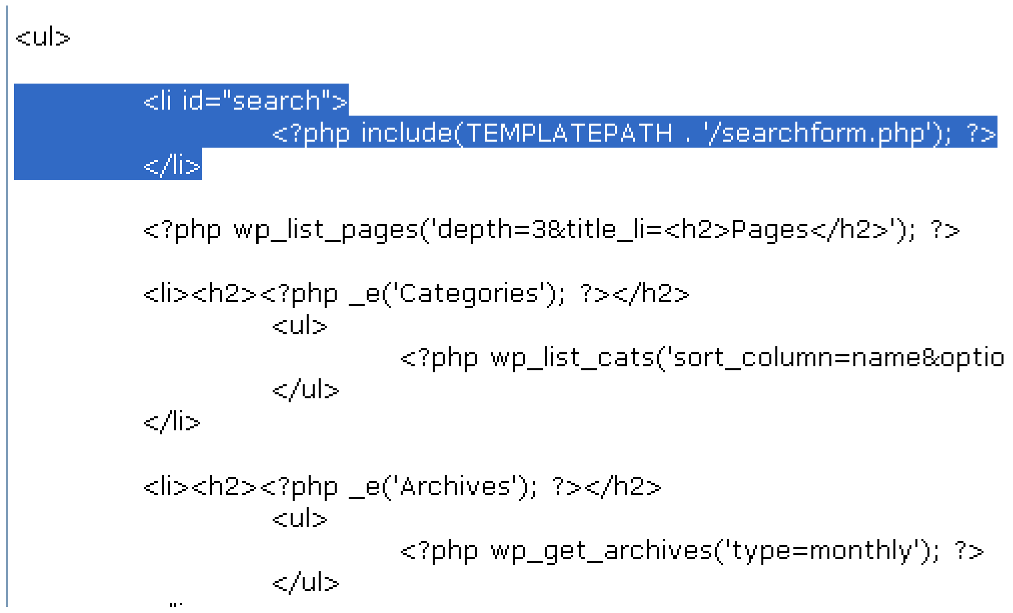
## Add search form

We are going to make the search form a new page of its own. Open a new notepad document, leave it blank, and save it as searchform.php in the same folder containing index.php.

Copy the codes in "searchform.txt" in the Wordpess IV folder to the searchform.php file.

In the index.php file, type the following codes on top of every listing within the Sidebar:

```
<li id="search">
<?php include(TEMPLATEPATH . '/searchform.php'); ?>
</li>
```



```
<ul>

        <li id="search">
                <?php include(TEMPLATEPATH . '/searchform.php'); ?>
        </li>

        <?php wp_list_pages('depth=3&title_li=<h2>Pages</h2>'); ?>

        <li><h2><?php _e('Categories'); ?></h2>
                <ul>
                        <?php wp_list_cats('sort_column=name&optio
                </ul>
        </li>

        <li><h2><?php _e('Archives'); ?></h2>
                <ul>
                        <?php wp_get_archives('type=monthly'); ?>
                </ul>
```

**So what happened?**

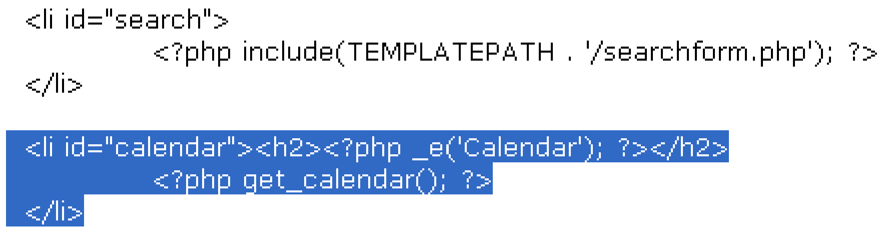
- **<li id="search">** - start a list item with an ID named, **search**. You're giving it an ID so you can style it later.
- **include()** - include whatever file you want to include. This is different from using a WordPress template function to call for something because **include()** is simply including what's already there. In this case, it's the codes within the **searhform.php** file. The information to be included does not change on a per blog basis. In other words, my search form looks exactly like yours.
- **TEMPLATEPATH** - the location of your theme's folder, **wp-content/themes/tutorial**
- **'/searchform.php'** - location and name of the file, **/searchform.php**
- The **period** between TEMPLATEPATH and '/searchform.php' connects them so you end up with: **wp-content/themes/tutorial/searchform.php**
- **</li>** - close the list item

Notice, the search form block doesn't have a sub-heading like Categories, Archives, Pages, or Blogroll. You can give it a sub-heading if you like…

## Add Calendar

Type the following codes under the Search-Form or Page-listing block:

```
<li id="calendar"><h2><?php _e('Calendar'); ?></h2>
<?php get_calendar(); ?>
</li>
```



```
<li id="search">
        <?php include(TEMPLATEPATH . '/searchform.php'); ?>
</li>

<li id="calendar"><h2><?php _e('Calendar'); ?></h2>
        <?php get_calendar(); ?>
</li>
```

**What happened?**

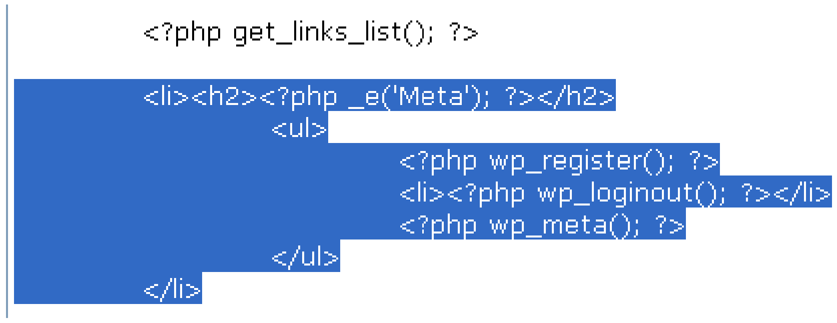
- **<li id="calendar">** - open a list item with an ID named, "calendar"
- **<h2>** - start a sub-heading
- **<?php _e('Calendar'); ?>** - print the word **Calendar**
- **</h2>** - close sub-heading
- **get_calendar()** - call for the calendar using get_calendar() function
- **</li>** - close list item

You're done with the calendar.

## Add Meta

Type the following codes under the **get_links_list()**:

```
<li><h2><?php _e('Meta'); ?></h2>
      <ul>
            <?php wp_register(); ?>
            <li><?php wp_loginout(); ?></li>
            <?php wp_meta(); ?>
      </ul>
</li>
```



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

        <li><h2><?php _e('Meta'); ?></h2>
                <ul>
                        <?php wp_register(); ?>
                        <li><?php wp_loginout(); ?></li>
                        <?php wp_meta(); ?>
                </ul>
        </li>
```

Save notepad and refresh browser.

**So what happened?**
You started a list item (**LI**) with a sub-heading (**H2**) Meta. Under the sub-heading, you nested an unordered list (**UL**). And for each link, you wrapped list item (**LI**) tags around it.

The **wp_register()** generates its own set of **<li>** and **</li>** tags; when you're not logged in, it displays the **Register** link; when you are logged in, it gives you the **Site Admin** link.**wp_loginout()** doesn't generate its own list item tags so you wrapped list item tags around it; when not logged in, you get the **Login** link; when logged in, you get the **Logout** link. For now,**wp_meta()** does nothing; it's invisible on the web page and invisible in the source codes. Don't think about **wp_meta()** for now; you'll rarely use it.

## If you are lost at this point

I can see how we can get "lost" at this point. So, I created the entire sidebar code in "sidebar.txt" in the Wordpress IV folder. Remove what you have and you can place it in the location for the sidebar div .

## Widgetizing the sidebar

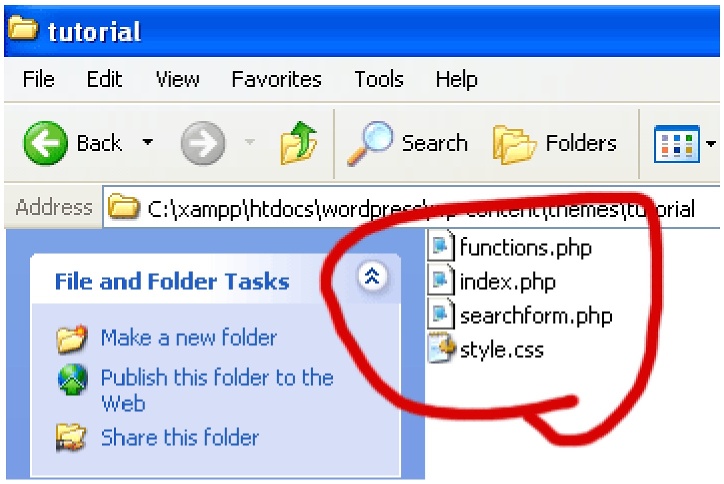
Widgetizing simply means getting the Sidebar ready for the Widget plugin; this plugin allows you to easily re-arrange features within the Sidebar.

For example, instead of having to modify the Sidebar codes to switch the positions of Categories and Archives, you simply drag the Archives and Catgories listings to their positions. This can be done in two simple steps.

## Step 1: Create a functions.php file

Start a new Notepad, leave it blank, save it as **functions.php**. Copy everything in functions.txt in the Wordpress IV folder to your **functions.php** file. Save and close the **functions.php** Notepad.

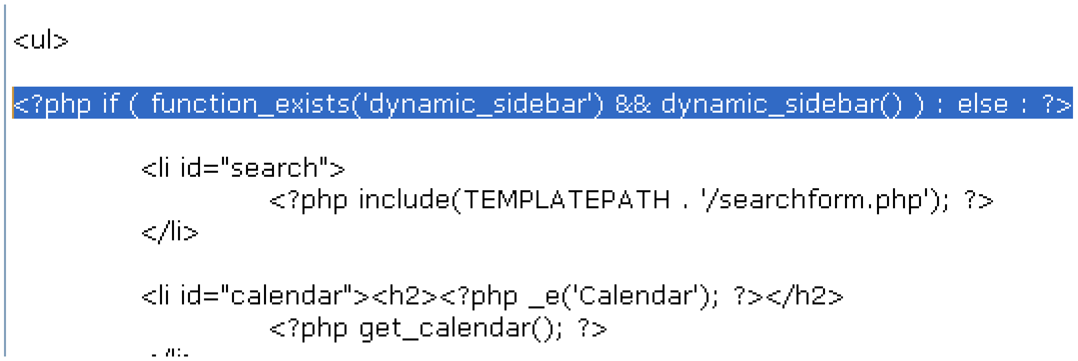
There should now be four files in the theme folder.



## Step 2: Widgetize the Sidebar

Type the following codes directly after the sidebar's first **<ul>** tag.

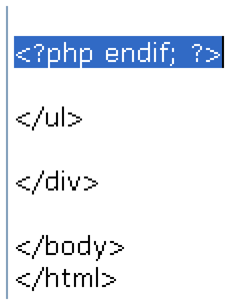
```
<?php if ( function_exists('dynamic_sidebar') && dynamic_sidebar() ) :
else : ?>
```



```
<ul>

<?php if ( function_exists('dynamic_sidebar') && dynamic_sidebar() ) : else : ?>

        <li id="search">
                <?php include(TEMPLATEPATH . '/searchform.php'); ?>
        </li>

        <li id="calendar"><h2><?php _e('Calendar'); ?></h2>
                <?php get_calendar(); ?>
```

Type this one directly before the **</ul>** tag:

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



```
<?php endif; ?>

</ul>

</div>

</body>
</html>
```

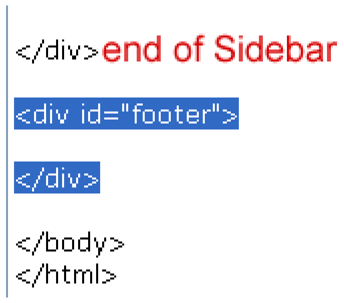
Save the index.php file. You will not see any changes on the web page until you've installed the Widget plugin.

## The Footer

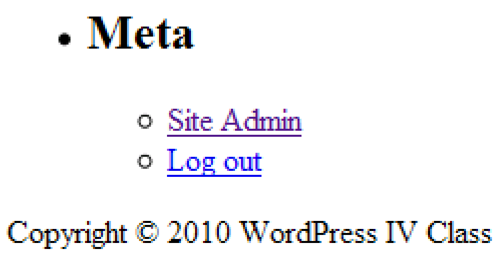
Add a DIV or invisible box under the Sidebar DIV and fill in some copyright text for the footer.

Type the following codes under the Sidebar DIV:

```
<div id="footer">
</div>
```



```
</div>end of Sidebar

<div id="footer">

</div>

</body>
</html>
```



- **Meta**
  - Site Admin
  - Log out

Copyright © 2010 WordPress IV Class

Add footer text within paragraph tags.

```
<p>
Copyright &#169; 2010 <?php bloginfo('name'); ?>
</p>
```

&#169; = © in HTML

## Style.css

CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

1. Open style.css in Notepad
2. Type the following codes in the style.css flie:

```
body{
	margin: 0;
	font-family: Arial, Helvetica, Georgia, Sans-serif;
	font-size: 12px;
	text-align: left;
	vertical-align: top;
	background: #ffffff;
	color: #000000;
}
```

Look familiar? This is the CSS file information that was created during Dreamweaver IV. It's the same language! If you can do it in Dreamweaver – we can use the same code in Wordpress (if we name our DIVS the same thing). How cool is that?

Let's add additional pieces:

```
a:link, a:visited{
	text-decoration: underline;
	color: #336699;
}

a:hover{
	text-decoration: none;
	color: #ff0000;
}
```

## Making the theme look good

This will be up to you. Everyone has ideas of what looks good and what doesn't.

Now, if we didn't originally do the design in Dreamweaver, we can go through add type out the stuff in our style.css. Since I'm too lazy for that, let's just copy design.txt to style.css.

## Breaking up the index file

Since leaving this as one big file doesn't give us much flexibility, let's create 3 blank files in our tutorial folder:

1. header.php
2. sidebar.php
3. footer.php

Take the information from our index template (in the correct locations) and place them in the corresponding files.

For the header file, copy everything from the </div> to the very beginning of the file. This includes that odd <!DOCTYPE … code.

For the sidebar and footer, just grab the information in the <div> tags (including the div tags!)

Now, where that information was, place in the codes:

1. <?php get_header(); ?>
2. <?php get_sidebar(); ?>
3. <?php get_footer(); ?>

## Sub-Template Files

The sub-template files makes other locations of your website look different from whence you came.

Look at the Sidebar, click on an Archives link. The resulting page doesn't look different from your front page right?

- Create a new file: **archive.php**
- Copy and paste everything from **index.php** to **archive.php**
- Save archive.php
- In the archive.php file, change **the_content** to **the_excerpt**.
- Save the archive.php file again.

By creating an **archive.php** file and changing it to make it different from **index.php**, you are **customizing** the appearance of **archive pages**.

Now, if you refresh your archive page, it will give you only excerpts, not the full posts.

**Why would you want to do this?** - to prevent Google from penalizing your website for having duplicate content. If one of the archive pages and the front page display the same content, that's duplicate content.

**What if you have a private website?** Then, it's not necessary to distinguish the archive pages from the front page. That's not to say excerpts aren't useful for private blogs.

**Also** - By default, your **category pages** will look for instructions on how to display content from the **archive.php** file. If you don't have an **archive.php** file, category pages will look for **index.php**. If you want the **category pages** to look different the **front page** and **archive pages**, create a **category.php** file and customize it.

### Create a search return page

- Create a new file: **search.php**
- Copy and paste everything from **archive.php** to **search.php**
- Save and you're done.

Now, all search results will be returned as excerpts. Without the **search.php** template file, the search option looks to **index.php** on how to display search results.

### Create a singular page item

- Create two new files: **page.php** and **single.php**
- Copy and paste everything from **index.php** to **page.php** and **single.php**. (For now, page and single should be the same.)
- Save page and single. Close index. Close single.

## Where do I go to find all of these *codes?*

Your life will be made lots easier if you stand on the shoulders of giants. Many things have been created for you. To find out what gets done, go to:

http://codex.wordpress.org/

The tags are located at:

http://codex.wordpress.org/Template_Tags/