

## 那么你想创建 WordPress 主题吗？

从零开始创建 WordPress 主题其实不难，我会手把手教你。

关于创建 WordPress 主题的教程这个话题已经被人写过，并且 [WordPress 官方网站](#)上也有指导文章。但是当你不懂这方面的行话的话，这些教程或者向导真的对你有帮助吗？甚至当我阅读 WordPress 上向导的时候，我都迷失了。

工具，在我继续之前，你需要下面这些工具：

- 安装 **WordPress** 在你的电脑上，你可以参考在 [Installing WordPress Locally Under Windows XP](#) 的安装说明。如果你由于无论原因不能在你的电脑上安装 **WordPress**，不用担心，你可以在你的服务器上安装一个测试版的 WordPress。
- 记事本或者其他的文本编辑器，我用记事本。（译者注：推荐 **NotePad++**）
- **SmartFTP** - 如果你要在线测试主题，我建议你下载和安装使用 [SmartFTP](#) 或者其他的 **FTP** 程序去上传你的主题文件。
- 收藏 [XHTML 验证器](#)和 [CSS 验证器](#)。你将需要这些工具去验证你的主题。当你需要去认出和改正错误的时候，他们也有用武之地。

## 完整课程列表

[#1 Intro — 介绍](#)

[#2 Template Files and Templates — 模板文件和模板](#)

[#3 Starting Index.php — 开始 Index.php](#)

[#4 Header Template — Header 模板](#)

[#4b Header Template Continues — Header 模板 2](#)

[#5 The Loop — 主循环](#)

[#5b The Content — 日志内容](#)

[#5c Postmetadata — 日志元数据](#)

[#5d Else, Post ID, Link Title — Else, 日志 ID, 链接标题](#)

[#5e Posts Nav Link — 日志导航链接](#)

[#6 Sidebar \(and Categories\) — 侧边栏](#)

[#6b Page-Link Listing \(wp\\_list\\_pages\) — 页面链接列表](#)

[#6c Get Archives and Links — 存档和链接列表](#)

[#6d Search Form and Calendar — 搜索框和日历](#)

[#6e Widgetizing Sidebar — 窗体化侧边栏](#)

[#7 Footer — 尾部](#)

[#8 How to Validate — 如何验证](#)

[#9 Style.css and CSS Intro — Style.css 和 CSS 介绍](#)

[#10 hex Codes and Styling Links — 十六进制颜色代码和样式化链接](#)

[#11 Widths and Floats — 宽度和布局](#)

[#12 Post Formatting and Miscellaneous — 日志样式化和其他杂项](#)

[#13 Styling Sidebar — 样式化侧边栏](#)

[#14 Footer and Diving Index — 底部和拆分 Index](#)

[#15 Sub-template Files — 子模板文件](#)

[#16 Comments Template — 留言模板](#)

## WP Theme 教程 #1：介绍

这篇是我的关于创建 WordPress 主题 [系列教程](#) 的第一篇。我不会一次就教你所有的东西，那样只会让你更加迷惑。我向你展示的也不是 WordPress 主题制作的参考，而我所做的是一步一步的教你如何制作主题。如果你想所有关于 WordPress 主题的参考资料，请查阅 [WordPress.org 的文档，设计和布局](#)。否则就请看读我的教程。

这篇将涉及到：

- 基本规则
- 术语 / 行话
- 层式结构

基本规则：

- 规则 **#1**：以正确顺序关闭所有标签。

The right way to close:

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

The wrong way to close:

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

在错误关闭标签的方法中，关闭的 **ul** 标签是不按次序的。

每个标签都是在 `<` 和 `>` 中，是否有斜线 `/`，则指明这个标签是开始标签还是结束标签。如：`<>` 是开始标签，而`</>` 是结束标签。在上面的例子中，我使用 `ul`（无序列表）`li`（列表元素）标签。注意开始和结束 `li` 标签在开始和结束 `ul` 标签的里面，这就是标签正确嵌套。

- **规则 #2:** 每个主题至少有这两个文件 - **style.css** 和 **index.php**。`index.php` 告诉你的主题所有的元素该如何布局，`style.css` 则告诉你主题所有的元素该如何展示。

下面是完整的主题文件列表（现在不用担心这个列表）：

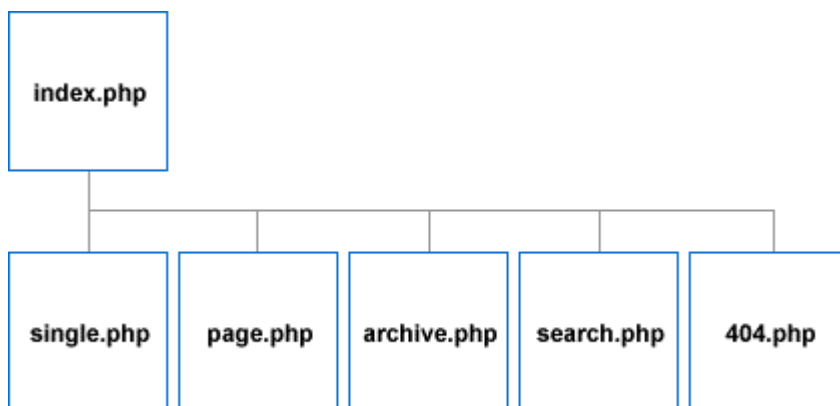
- style.css
- index.php
- home.php
- single.php
- page.php
- archive.php
- category.php
- search.php
- 404.php
- comments.php
- comments-popup.php
- author.php
- date.php

## 专业术语：

- **Template（模板）** — 其实就是一个代码集，很多地方会利用到这个代码集，把它整合成一个代码集，就不必一遍遍输入这些重复代码。
- **Template file（模板文件）** — 一个包含一个或者多个代码集（模板）文件。每个主题是由多个模板文件组成的，如：`index.php`，`style.css`，`sidebar.php` 等等。
- **Theme（主题）或者 WordPress theme（WordPress 主题）** — 所有你正在使用的文件：文本，图像，代码等等。注意：**WordPress theme（主题）**和 **WordPress template(s)（模板）**是两个不同的东西，尽管有些人认为他们一样。
- **Post（日志）** — 现在你读的就是一篇日志。此外，它是你 **blog** 的一个简单的条目，如：一个页面或者一篇日记。
- **Page（静态页面）** — 一种特殊的 **post**，它不是以分类组织的。它有别于你其他的日志。注意：在 **WordPress**，**page（页面）**和 **Page（静态页面）**是两种不同的东西。

## 层式结构：

下面这幅图简单的向你展示，一旦你主题中的某个文件丢失了，WordPress 主题系统将会寻找什么。这里列出了 6 个文件而不是完整的 13 个，因为这 6 个是你需要或更大时间去关注的。在接下来的教程中，你将学到余下的文件。



我们可以通过他们的在图中位置知道他的重要性级别，上面，左边是最重要的，下面和右边是最不重要的

为什么会存在模板文件的层式结构或者说是重要性级别呢？因为如果 `archive.php` 文件（用来显示存档页面）丢失了，那么 WordPress 将会指定 `index.php` 去控制存档页面如何显示。

如果 `single.php` 模板文件丢失了呢，哪个模板文件它会去寻找用来显示单一日志呢？它会寻找 `index.php`。

如果你有什么问题，请放手来问，请最好给我们留言而不是给我们 email。那样的话，如果其他人寻找相同的答案就不用重复他们的问题了。

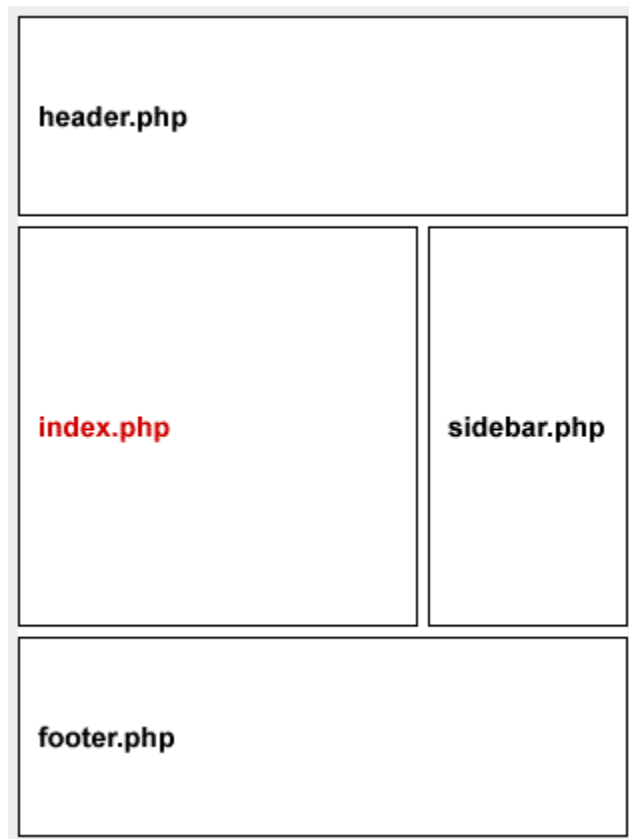
## WP Theme 教程 #2：模板文件和模板

模板文件（`template files`）和模板（`template`）是我的关于创建 WordPress 主题 [系列教程](#) 的第二篇。如果你还没有看过[教程 #1](#)，请先回去看教程一，否则你将无法理解在教程二中使用的名词。

现在我们已经学过了规则和术语，这篇将让你更熟悉模板文件，模板，已经每个页面的结构。

有件事情你必须牢记的是：你博客的每个页面是有多个模板文件组成的。

这里是首页的例子：



在上面的例子，我们可以看出首页是由 4 个模板文件组成：**header.php**，**index.php**，**sidebar.php** 和 **footer.php**。

### Header 模板文件：



通常在这个文件中包含你博客的**标题**（title）和**描述**（description）。而且它们通常在整个博客中保持一致。

### Index 模板文件：

这个模板文件包含你的**日志的标题**，**内容**（每篇日志的文本和图片）和**日志的元数据**（指的是每篇日志的信息，包括谁是作者，什么时候发表，在那个分类下，有多少留言等等）。

### [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

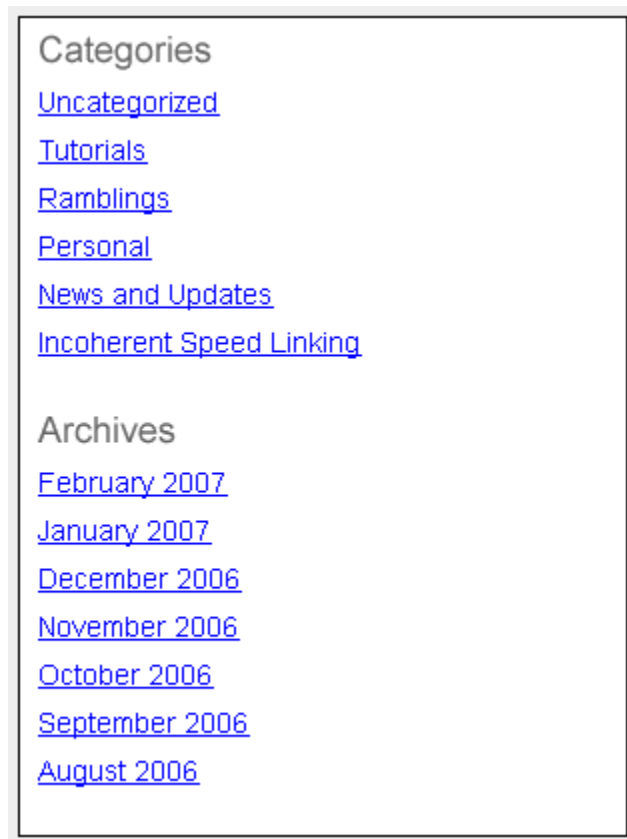
Posted on February 21, 2007 by You

Filed under: [Tutorials](#)

[8 Comments](#)

## Sidebar 模板文件

这个模板文件控制你的[页面链接列表](#)，[类别链接列表](#)，[存档链接列表](#)，**blogroll 列表**和[其他东东](#)。（侧边栏不一定要求布局到右边，我正在使用的主题是经典的两列的 博客布局。）



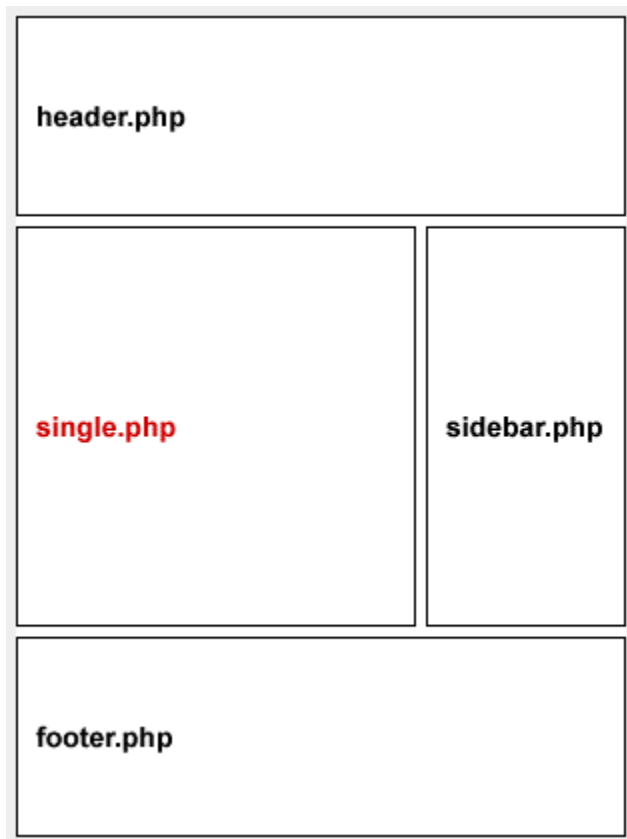
## Footer 模板文件:



像 `header.php` 模板文件一样，`footer` 通常不会因为页面的改变而改变，你可以在这里放置任何东西，但是通常是**版权信息**。

现在让我解释为什么上面图片中的 `index.php` 所在的区域是红色的。`Index.php` 所在区域是红色是说明那块区域将会改变，取决于你在博客的那个页面。

如果你在单一日志页面，这时候页面将会包含这四个模板文件：**header**，**single**，**sidebar** 和 **footer**。



## WP Theme 教程 #3 : 开始 Index.php

开始 Index.php 是我创建 WordPress 主题[系列教程](#)的第三篇。如果你没有看过该教程的[一](#)和[二](#)，我建议你先看它们。否则你会对我这篇讲什么迷惑。

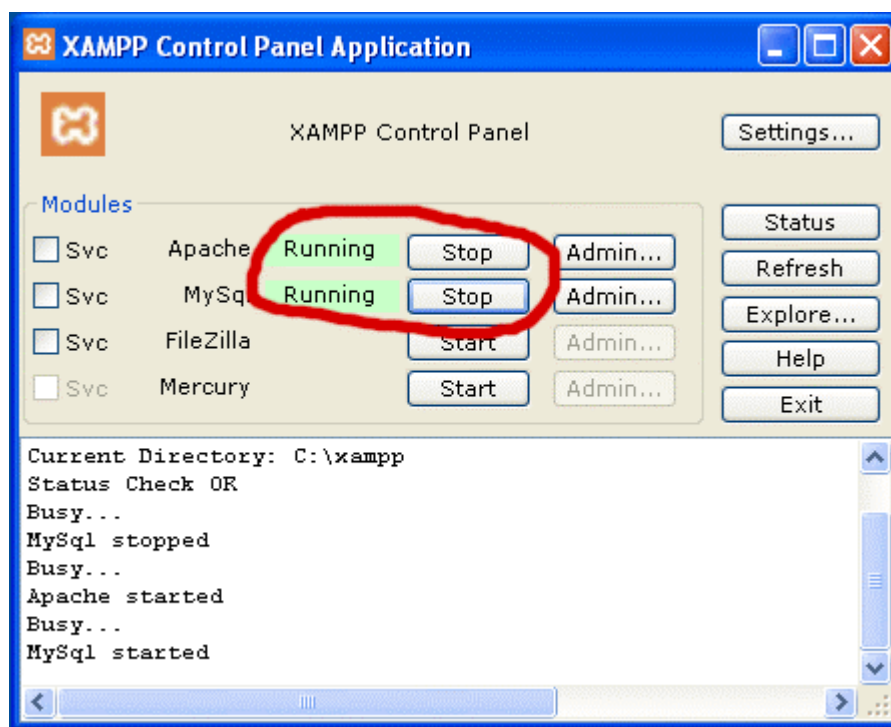
现在是开始动手创建你的 WordPress 主题的时候了。在这篇中，你将要着手写些 WordPress 代码。在这里你真正需要[把 WordPress blog 安装到你的电脑上](#)，不是安装到服务器上，因为本地更方便测试。

### 第 1 步：打开 Xampp 控制面板。

转到你的 xampp 文件夹。通常是：C:\xampp。

双击 **xampp-control.exe** 将会弹出一个新的窗口。单击 Apache 和 MySQL 的启动按钮。如下图所示：





启动之后你看最小化窗口了。

## 第 2 步：创建你的主题文件夹。

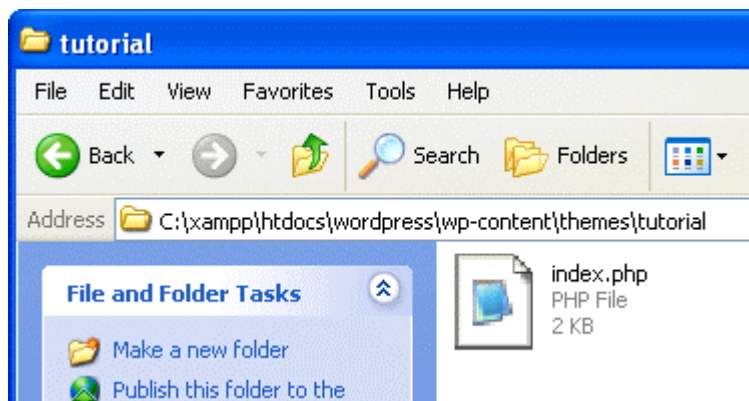
转到你的 WordPress 主题文件夹，应该在 **xampp/htdocs/wordpress/wp-content/themes**。创建一个新的文件夹，命名为 **tutorial**。

## 第 3 步：创建 **index.php** 和 **style.css** 文件。

打开记事本或者你选择的文本编辑器，记事本是在开始 > 程序 > 附件 > 记事本。

把 [index.txt](#) 这个文件的所有内容拷贝到你的记事本。

保存为 **index.php** 到 **tutorial** 文件夹。



打开另外一个记事本，直接保存为 **style.css** 到相同的文件夹下。

现在有两个文件了: index.php and style.css.

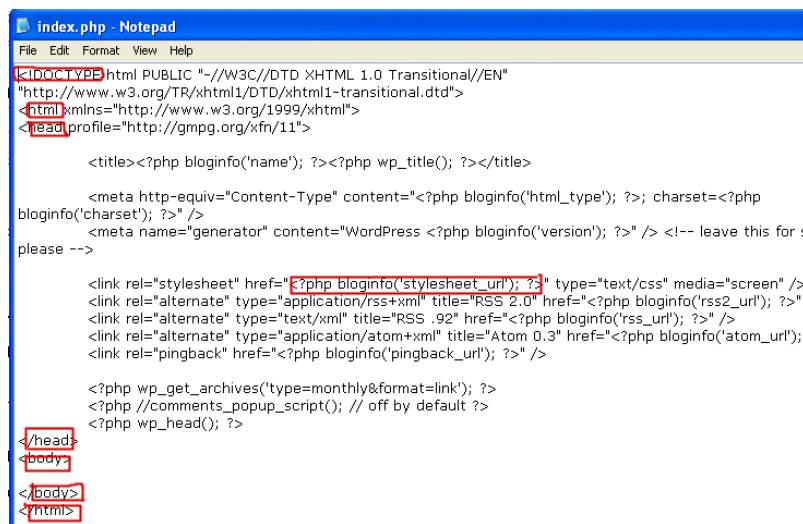


index.php  
PHP File  
2 KB



style.css  
Cascading Style Sheet Document  
0 KB

## index.php 解释:



点击上面的图片查看大图。我会向你解释每个红色圆圈区域是什么意思。

**Doctype** - 指明哪种类型的代码你用来编码你的主题。**Doctype** 此时还不必去思考它的确切的意思。

**<html>** 是网页开始的地方。

**<head>** 是你的网页头部开始的地方。每个网页都有一个头部和主体。**</head>** 是头部结束的地方。

`<?php bloginfo('stylesheet_url'); ?>` 是一个 PHP 函数，它能取得 **style.css** 文件所在的路径，这样主题就能链接到它并能样式化页面上所有元素。任何时候，PHP 代码都是在 `<?php` 和 `?>` 之间的。PHP 代码和其他的代码不一样的，在 PHP 中，`<?php` 时开始 PHP 代码而 `?>` 是结束 PHP 代码。

所以：

- `<?php` - 开始 PHP 代码
- `bloginfo('stylesheet_url')` - 调用 **style.css** 文件所在的路径
- `;` - 停止调用函数。分号是用来结束一个 PHP 语句。
- `?>` - 结束 PHP 代码

继续.....

`<body>` - 这是网页主体开始的地方。你能在网页上看到和读到的东西就是主体部分。你正在阅读的这个教程说明你在正在看当前这个网页的主题部分。

`</body>` 是网页主体结束的地方。

`</html>` 是网页结束的地方，没有东西在它的后面了。

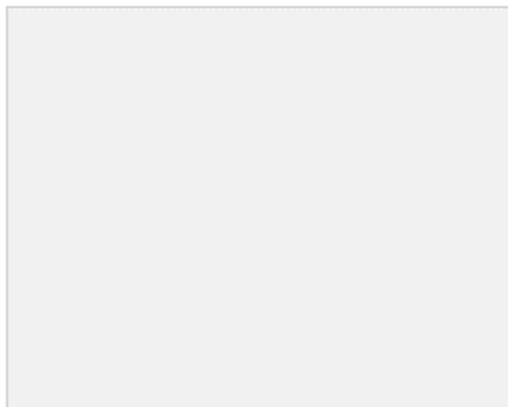
**第 4 步：**把 [style.txt](#) 中所有的东西拷贝到你的 **style.css** 文件中。保存和关闭它。

**第 5 步：**安装你的主题。

打开浏览器。在地址栏输入输入 <http://localhost/wordpress/wp-login.php>。登录到你的 WordPress 管理界面。（这里能够看到 WordPress 登录页面是因为你在第 1 步的时候启动了 Xampp。否者的话，在这里你的浏览器会报找不到的错误。）

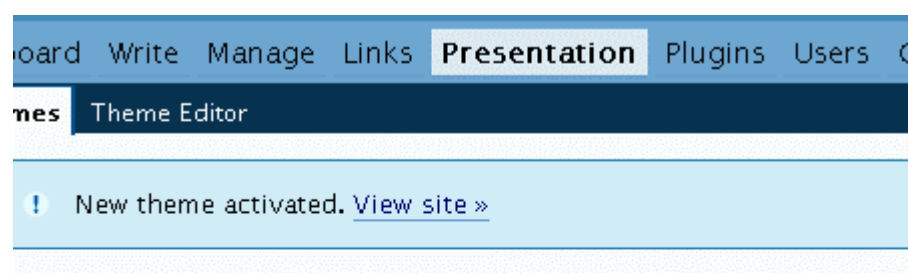
在管理界面下到 **Presentation** 菜单并点击名字为 **Tutorial** 的主题去激活它。

## Tutorial 1.0



This is my theme for a tutorial.

注意，你的主题文件没有屏幕缩略图，所以是空白的。一旦激活了，WordPress 就会告诉你激活信息。



## Current Theme

Current theme preview

**Tutorial 1.0 by :**

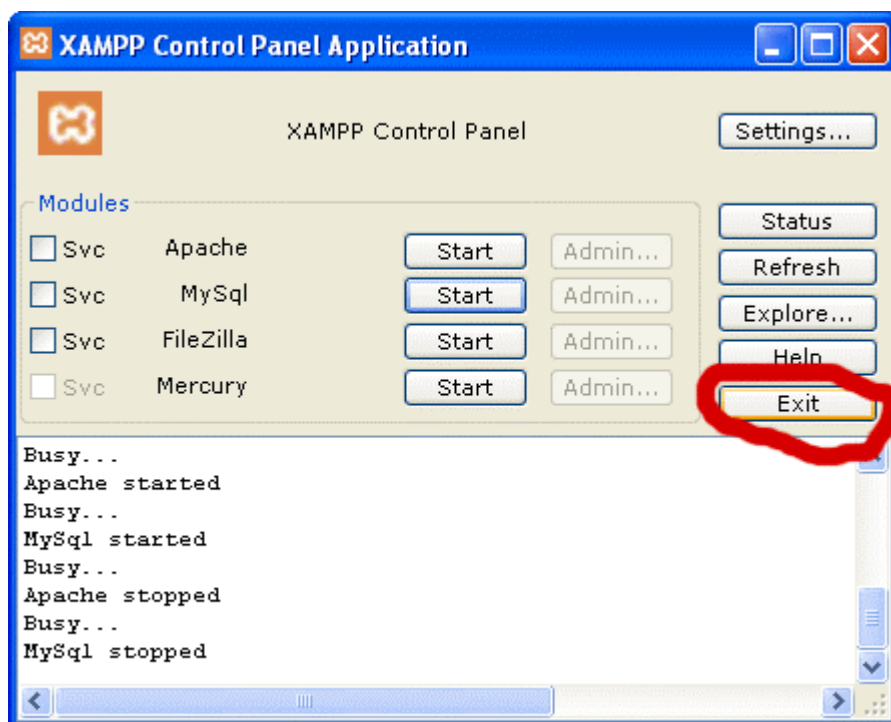
This is my theme for

All of this theme's files are located in `wp-content/themes/tutorial`

现在打开一个新的浏览器或者标签页（如果你的浏览器支持标签页浏览）并在地址栏输入 <http://localhost/wordpress>。你应该得到一个空白页面，恩，完全空白的页面。如果不是，那你就是在错误的页面上了。

你的主题已经创建好了，这就是这个课程，下一步我们将讨论头部模板。

不要忘记关闭 **Xampp**。双击它在任务栏中小图标，点击 **Apache** 和 **MySQL** 的停止按钮，然后点击推出。



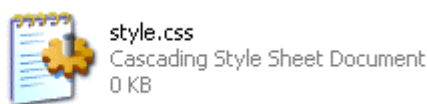
## WP Theme 教程 #4 : Header 模板

如果你尚未准备好，就请从头查看我的关于创建 WordPress 主题的[系列教程](#)。如果不是，我们继续。前面我向你展示了如何启动 Xampp，安装主题并向你介绍了 PHP，这篇我们将继续 PHP 并学习如何调用你博客的标题。

记住，输入所有代码。而不是拷贝我给你的代码。这都是让你尽量记得你所学到的。

### 第 1 步:

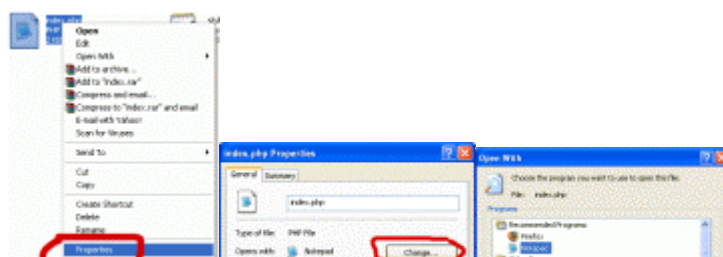
打开 Xampp。然后打开你的主题文件夹，**xampp/htdocs/wordpress/wp-content/themes/tutorial**。你应该看到你上次创建的两个文件：**index.php** 和 **style.css**。



你的 index.php 和 style.css 文件的内容应该符合：[index.txt](#) 和 [style.txt](#)。

### 第 2 步

- 打开新的浏览器窗口，转到 **http://localhost/wordpress**。你应该看到一个空白的页面因为上次你安装了一个空白的主题。
- 返回你的主题文件夹并打开 **index.php** 文件。为此，你应该右击 **index.php**，选择**属性**，点击**更改**，找到**记事本**或者你使用的文本编辑器，选择记事本并按**OK**。更改之后，双击 **index.php** 文件。

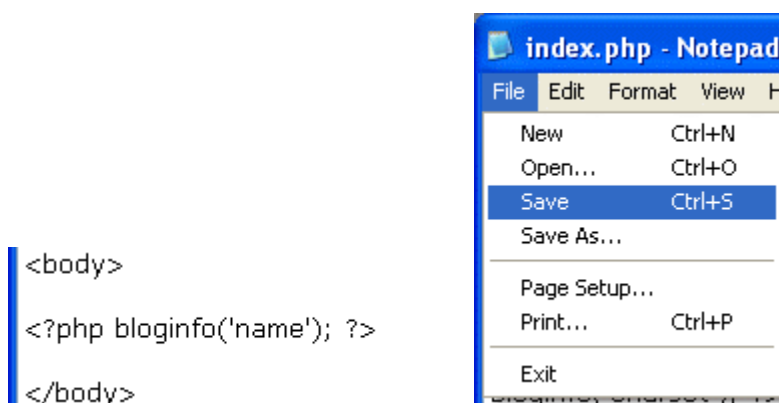


到目前为止，你打开了你的主题文件夹，浏览器和 **index.php** 文件。

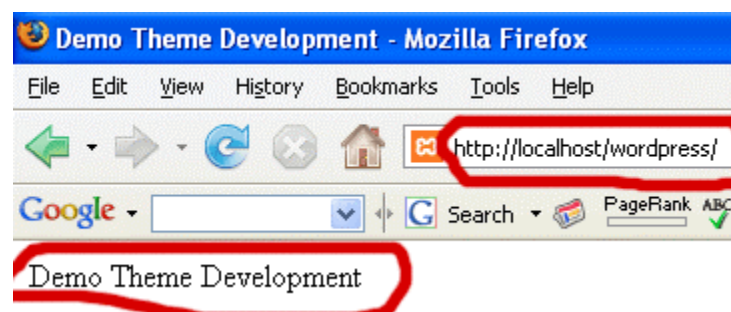


### 第 3 步:

编辑 **index.php** 文件。在 **<body>** 和 **</body>** 这两个标签之间输入 **<?php bloginfo('name'); ?>**，保存它。

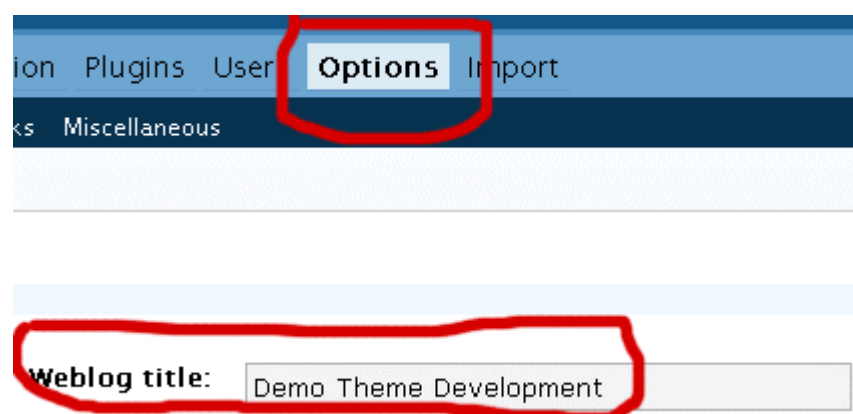


返回到你的浏览器，**刷新**。你应该能够看到你的博客的标题。博客的标题是 **Demo Theme Development**。



刚才发生什么了？

你在网页的主体之间加入了一行 PHP 代码到你的 `index.php`。哪一行调用了你博客的信息呢？`bloginfo()`。具体它是调用了你博客的标题。这个名字是你在 `option` 页面中的 **Weblog Title**。



`<?php` - 开始 PHP 代码

`bloginfo('name')` - 调用博客信息，具体是博客的标题。

`;` - 结束调用博客信息

`?>` - 结束 PHP 代码

每次你在 `index.php` 文件中增加或者更改任何东西之后，你都可以保存，然后刷新页面去查看结果。

#### 第 4 步：

把博客的标题放入超链接中，你需要一个 `XHTML` 标签。

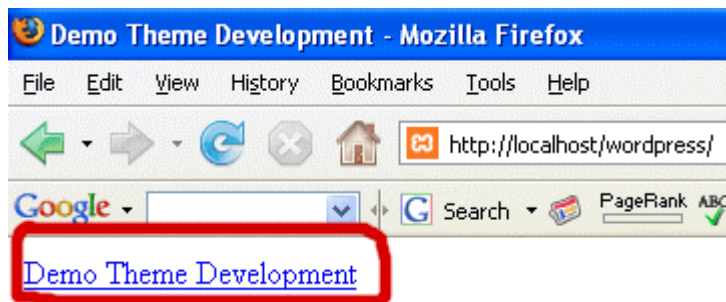
返回你的 `index.php` 文件。

在同一行增加 `<a href="#">` 和 `</a>`。此时新行的代码应该是：

`<a href="#"><?php bloginfo('name'); ?></a>`

返回到你的浏览器，刷新，然后你就可以看到你的博客的标题变成了链接。





现在它是一个链接，但是它没有链接到哪里。因为这个是你博客的标题，应该让它链接到首页。为此，在 **href=**后的双引号中输入 **<?php bloginfo('url'); ?>**

保存，你现在的代码应该是：

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

返回到你的浏览器，刷新，当你鼠标在你的链接上面的时候，浏览器的状态栏应该显示 <http://localhost/wordpress>



现在你点击这个链接，它应该让你返回首页。你可能看到是相同的页面，但是用 **#** 或者 **http://localhost/wordpress** 作为你的链接地址是完全不一样的。你会在接下来的课程学习到他们之间的不同。

刚才发生什么了？

你把你的网站名字变成了链接，并使它链接到博客的主页。

**bloginfo('url')** - 调用你博客基本信息，具体是首页的地址或者 URL

**<a>** - 是一个用于添加链接的 XHTML 标签

**</a>** - 链接的结束标签。否则你的网页将不知道哪里结束链接，这样会使得页面接下来的内容都变成链接。还记得**规则 #1**吗？关闭你打开的所有标签。

**href=""** - 超文本的简写。在引号之间就是它的值。

最终代码为：

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

意思为：开始一个链接，链接的地址是博客的 URL，用 PHP 函数 **bloginfo('url')** 去调用这个地址或者 URL。这个链接的文本是博客的标题并使用 PHP 函数 **bloginfo('name')** 去调用博客的标题。最后结束链接。



## WP Theme 教程 #4b : Header 模板 2

这是我的关于创建 WordPress 主题 [系列教程](#) 的第四篇。最后说一次，请你务必能先读下前面的日志。否则你很难理解这篇日志。这篇会很快完成 header 模板并介绍 DIV 盒子模型

### 第 1 步

- 启动 Xampp
- 打开 Tutorial 的主题文件夹
- 打开浏览器，在地址栏输入 `http://localhost/wordpress`
- 返回主题文件夹，用记事本打开 `index.php`

### 第 2 步

现在，`index.php` 包含：

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

给它添加 `<h1>` 和 `</h1>` 标签。H1 标签意思是标题一。一共可以有 7 级标题：H1, H2, H3, H4, H5, H6。按照默认，H1 是字体最大而 H6 是则最小。

现在你的 `index.php` 文件是：

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

保存，返回浏览器并刷新。

### 第 3 步

调用博客的描述，则在博客标题链接的下面输入 `<?php bloginfo('description'); ?>`。现在变成了：

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

```
<?php bloginfo('description'); ?>
```

保存，刷新你的浏览器，可以看到在你博客标题链接的下面出现博客的描述，你可以到管理界面下改变你博客的描述。

`<?php` - 开始 PHP 代码

`bloginfo('description')` - 调用博客信息，具体的是描述

;- 停止调用  
?> 结束 PHP 代码

## 第 4 步

这一步将向你介绍一个新的标签 — DIV。

给以上代码添加 `<div>` 和 `</div>` 标签：

```
<div>

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

</div>
```

保存，刷新浏览器，你看到没有任何变化

可以把 **DIV** 想像成一个无形的盒子。在这里它把**博客标题链接**和**博客描述**从其他东西中区分开。如果你没有对它进行样式化，它无非是单独的内容。以后你会用 **style.css** 这个文件去样式化这个无形的盒子。你可以给 DIV 附上 **边框 (borders)**，**填充 (padding)**，**页边空白 (margins)**，**背景颜色 (background color)**，**背景图片 (background images)**，以及其他一些东东。

## 第 5 步

添加 `id="header"` 到 DIV 标签，如下：

```
<div id="header">
```

保存，刷新浏览器。

同样也没有改变，到现在为止，我们指派了一个 **ID** 给 **DIV** 标签，因为将会有更多的 **DIV** 标签或者无形的盒子。我们使用 **ID** 来区分！

## WP Theme 教程 #5：主循环

调用你博客日志的主循环（**The Loop**）是 WordPress 中最重要的 PHP 代码集。这也是关于创建 WordPress 主题[系列教程](#)的第五篇。在继续学习之前，我们先复习下到目前为止学到了什么？

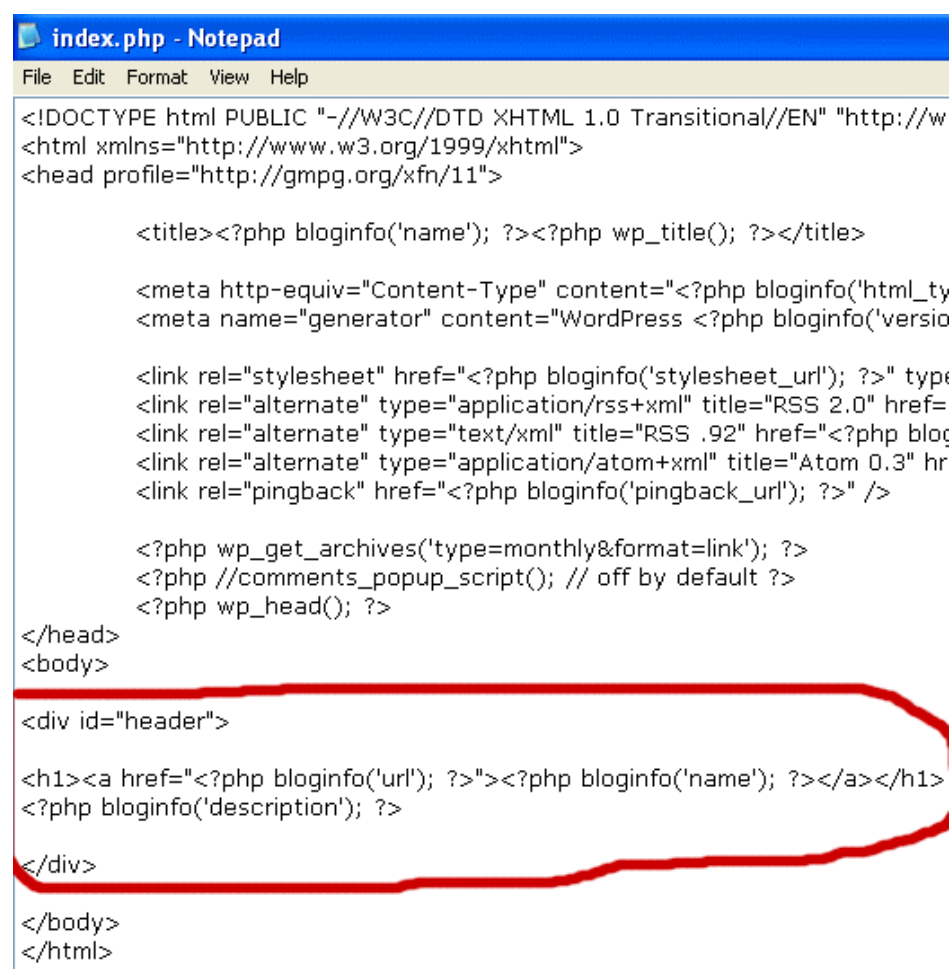
到目前为止，你已经学到：

- [规则，结构和 WordPress 主题的层式结构](#)
- [每个页面有哪些部分组成](#)
- [如何安装你的主题](#)
- [如何调用博客的标题和把它做成一个链接](#)
- [如何调用博客的描述和如何把 header 和其他内容分开](#)

现在让我们开始第五篇：

现在已经打开了 Xampp，“tutorial”主题文件夹，浏览器中打开 <http://localhost/wordpress> 以及在文本编辑器中打开 index.php 文件。

下面应该是在你这时候 index.php 文件中的内容：



```
index.php - Notepad
File Edit Format View Help

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w
<html xmlns="http://www.w3.org/1999/xhtml">
<head profile="http://gmpg.org/xfn/11">

    <title><?php bloginfo('name'); ?><?php wp_title(); ?></title>

    <meta http-equiv="Content-Type" content="<?php bloginfo('html_ty
    <meta name="generator" content="WordPress <?php bloginfo('versio

    <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" typ
    <link rel="alternate" type="application/rss+xml" title="RSS 2.0" href=
    <link rel="alternate" type="text/xml" title="RSS .92" href="<?php blo
    <link rel="alternate" type="application/atom+xml" title="Atom 0.3" hr
    <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />

    <?php wp_get_archives('type=monthly&format=link'); ?>
    <?php //comments_popup_script(); // off by default ?>
    <?php wp_head(); ?>
</head>
<body>

<div id="header">

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

</div>

</body>
</html>
```

记住，为了学习这些代码，请手工输入所有的东西，而不是拷贝和粘贴。

## 第 1 步

在 header DIV 标签下添加一个 DIV 标签，并给它的 ID 赋值为“container”，如下：

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

```
</div>
```

这个 DIV 标签是用来干什么的呢？它是用于把你的内容和它下面的所有东东都区分开，比如 sidebar 和 footer 这些东东。

## 第 2 步

在 Container 的 DIV 标签中添加如下代码：试着完全手工输入这些代码，如果遇到错误，重新再输入一遍。

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

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

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

手工输入代码可以让你更容易知道每个代码集是干什么的。你刚才输入的是 WordPress 中的主循环（**The Loop**）。在我解释这些代码作用之前，你的 index.php 应该是这样了：

```
<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>
```

注意我缩进了 **Container DIV** 中的每一行。这是为了更好的组织代码。仅仅输入代码是不够的，我们需要组织它们。对于缩进，使用 **tab** 键而不是空格键。

刚才发生了什么？

- **if(have\_posts())** - 检查你的博客是否有日志。

- **while(have\_posts())** - 如果你有，当你有日志的时候，执行 **the\_post()** 这个函数。
- **the\_post()** - 调用具体日志来显示。
- **endwhile;** - 遵照规则 #1，这里用于关闭 **while()**
- **endif;** - 关闭 **if()**
- **注释：**并不是所有的代码都需要两部分，为了打开和关闭自己。有些能够自我关闭，这就解释了 **have\_posts()** 和 **the\_post()**；这两个函数。因为 **the\_post()**；在 **if()** 和 **while()** 的外面，它需要分号去结束或者关闭自己。

### 第 3 步

在前面的课程中，你学习了使用 **bloginfo('name')** 去调用你的博客的标题。现在你将学习在主循环（**The Loop**）中如何调用日志标题。

在 **the\_post(); ?>** 的后面和 **<?php endwhile; ?>** 的前面输入 **<?php the\_title(); ?>**

```
<body>

<div id="header">

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

</div>

<div id="container">

    <?php if(have_posts()) : ?><p

        <?php the_title(); ?>

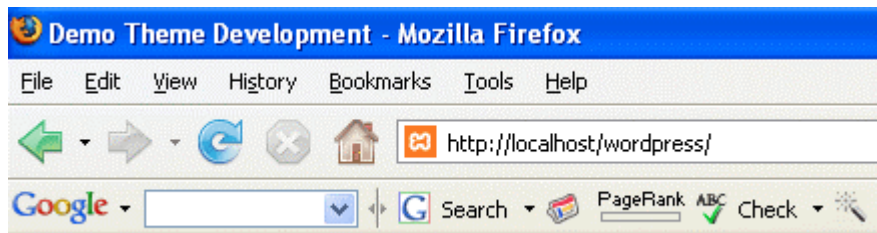
    <?php endwhile; ?>

    <?php endif; ?>

</div>

</body>
</html>
```

保存 **index.php** 文件，刷新你的浏览器，你应该看到在你博客描述的下方出现 **Hello World**，默认情况下，你的博客只有一篇日志。而我的测试的博客有多篇日志，所以我有多个日志标题。而且因为我所用的日志标题是一样的，并我没有进行组织整理他们，所以它们看起来像很长的一行 **Hello World**。



## Demo Theme Development

Just another WordPress weblog

Hello World Hello World Hello World Hello World Hello World Hello  
eiusmod tempor

### 第 4 步

把日志标题转变成日志标题链接。还记得你怎样把你博客的标题转变成一个链接呢？是在 `<?php the_title(); ?>` 两边增加 `<a href="#">` 和 `</a>`。

保存并刷新你的浏览器。现在你的标题变成了链接了，但是它们并没有指向哪里。为了使得每个标题都能指向正确的日志，你需要把 `#` 替换为 `the_permalink()`。

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

`the_permalink()` 是用来调用每篇日志地址的 PHP 函数。保存并刷新你的浏览器。如果你只有一个 **Hello World** 标题，把鼠标移到链接上面，观察你的浏览器底部的状态栏，他不再是 `http://localhost/wordpress/#`。

如果你有不只一个的标题链接，你将看到每个链接会链到不同的日志或者网页。但是我们的日志标题依然在同一行上面。为了分开它们，在你的链接代码两边添加 `<h2>` 和 `</h2>` 标签。

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

记住 **H1** 用作你的博客的标题，那是你网页的标题。**H2** 被用作子标题。现在你的日志标题链接是子标题了，每一个都是一行。保存 `index.php` 文件并刷新你的浏览器，观察更改的结果。

# Demo Theme Development

Just another WordPress weblog

[Hello World](#)

[Hello World](#)

这一篇到此为止。你的 **index.php** 文件现在应该是：

```
<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(); ?>
        <h2><a href="php the_permalink(); ?"><?php the_title(); ?></a></h2>
        <?php endwhile; ?>
    <?php endif; ?>
</div>
</body>
</html>
```

**注释：**如果你还没有创建一些测试的日志来测试你的主题，那么请到管理界面，点击 **Write** 按钮，并确认你是在 **Write Post** 页面下，写一篇日志并重复几次。你可以像我一样给它们唯一的标题 **Hello World**。

## WP Theme 教程 #5b：日志内容

在这篇中，我们将解决真正的东西。**如何展示你博客的内容？**然后你将看到更多的 **DIV** 标签，用于把博客的内容和在上一篇中创建的**日志标题**区分开。

（**注意：**上一篇课程是非常，非常重要的。如果你不是完全明白我上篇所讲的，你需要去重新看一篇和问我问题直到搞明白为止。）

下面开始这篇课程。打开 **Xampp**，打开 **“tutorial”**主题文件夹，打开**浏览器**，输入地址：**http://localhost/wordpress**，最后用文本编辑器打开 **index.php**。

### 第 1 步

在日志标题代码下面输入 `<?php the_content(); ?>`。

```
<?php if(have_posts()) : ?><?php while(have_posts()) : ?>
    <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?>
    <?php the_content(); ?>
<?php endwhile; ?>
<?php endif; ?>
```

保存并刷新浏览器，你应该在日志标题下面看到了一些文本。

## Demo Theme Development

Just another WordPress weblog

### Hello World



Lorem ipsum dolor sit amet, consectetur adipiscing  
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in  
officia deserunt mollit anim id est laborum. 😊

#### 刚才发生什么了？

你使用了 PHP 函数 **the\_content()** 调用了 日志的内容。现在，日志的内容只是一长行的文本，一直到窗口的右边，因为你还没有样式化它。还记得 **style.css** 这个文件吗？我们以后用它来控制所用元素显示的。

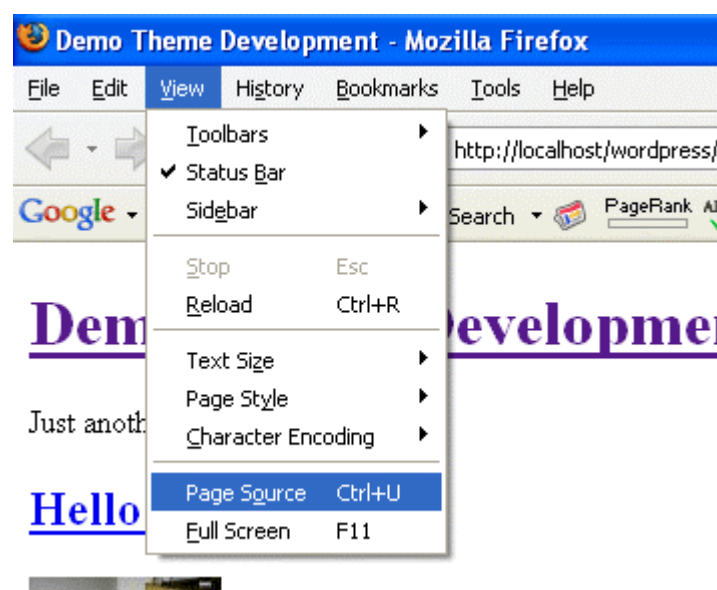


上一篇，我教你输入多篇测试的日志以便可以看到多篇日志的列出的样子。如果你还没有做，那现在就去输入吧，然后返回首页，应该看到多篇日志是这样子的：



继续.....

返回你的浏览器，点击**查看**选择**页面源代码**。就会弹出一个源代码窗口，如果你使用的是 **Internet Explorer**，那么弹出的是记事本。



我使用的是 **FireFox** 浏览器，下面是在 **FireFox** 中显示的样子：

```

<div id="container">

    <h2><a href="http://

    <p><img id="image13"
alt="dcfn0032.JPG" />Lorem ipsum do.
ad minim veniam, quis nostrud exerc:
velit esse cillum dolore eu fugiat :
laborum. <img src='http://localhost,
<p>

```

注意到 **index.php** 文件和**源代码**之间的区别了吗？你所有的文本，图像和其他附加的东西，所有在那个窗口或者记事本中的东西都是被这个函数 **the\_content()** 调用来的。是不是很有用？不依赖 **blogging** 软件和具体的 WordPress 模板系统，你应该自己对这些文本和图片进行编码。

还有，有没有注意到我圈出的开启和关闭的 **P** 标签。他们都没有在 **index.php** 文件中出现，但是他们在源代码中出现了。

### **P** 标签，为什么和如何使用？

**为什么** - 当你输入你的日志的时候，每次你跳过一行就是一个段落，你需要一个方法去展示？每个段落会在 **P**（段落，**paragraph**）标签之间。那就是你能够在你的段落之间有行距的原因。**如何** - 非常容易，WordPress 模板系统会帮你产生 **P** 标签。

## 第 2 步

给 **the\_content()** 两边添加 **DIV** 标签并给该 **DIV** 标签附上 **class="entry"** 属性，如下：

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

```
</div>
```

你现在的 **index.php** 文件应该是：

```
<div id="container">

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

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

            <div class="entry">

                <?php the_content(); ?>

            </div>

        <?php endwhile; ?>

    <?php endif; ?>

</div>
```

保存并刷新你的浏览器，你再次去查看源代码，你会发现你的每篇日志内容都被 `class="entry"` 的 DIV 标签给围住。

### 为什么？

第一个原因是，你现在知道日志题目哪里结束还有日志内容哪里开始。第一个原因是这是为了能让 `style.css` 文件能进行样式话准备的。如果你愿意，你可以样式你的日志的内容而不影响其他别的东西。

**id** 和 **class** 之间有什么区别呢？

到目前为止，对于每个 DIV 标签，你可以用 **id** 去命名它。记得 `id="header"` 吗？那么有什么不同呢？**id** 是唯一的而 **class** 不是。如果你从头到尾浏览你的源代码，你会发现只有一个 `id="header"` 和一个 `id="container"`，但是有多个 `class="entry"`。

那么 **header** 和 **container** 能用 **class** 去取代 **id** 吗？完全可以。

牢记这一点，你不能重复任何 **id**。比如，你不能在同一页面上有两个 `id="header"`。当你想一遍又一遍重新利用一些东西如日志的题目，那么请使用 **class**。

### 第 3 步：

用一个 DIV 标签把日志的标题和内容一起围住。并把这个 DIV 标签命名为：`class="post"`。

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

```
</div>
```

（你可以按照你自己的想法把 **class** 和 **ID** 的名字命名成你想要的名字，比如你最喜欢的食物，但是 **post** 和 **entry** 更简洁，简单并更容易记，对吧？）

现在你的 `index.php` 文件为：

```
v id="container">
    <?php if(have_posts()) : ?><?php while(have_
        <div class="post">
            <h2><a href="<?php the_permalink(
                <div class="entry">
                    <?php the_content(); ?>
                </div>
            </div>
        <?php endwhile; ?>
```

这个是经过缩进整理后的版本：

```
<div class="post">
    <h2><a href="<?php the_permalink
    <div class="entry">
        <?php the_content(); ?>
    </div>
</div>
```

我使用 **tab** 键而不是空格键在 `index.php` 文件中产生缩进的。为什么进行组织呢？不像我的屏幕截图上一样，你的代码不会有红色或者绿色的高亮显示，你需要有个能够跟踪你的代码的方法，通过缩进，使你更容易知道哪个 `</div>` 是结束哪个 `<div>`。

保存并刷新浏览器，然后查看源代码中的代码。

为什么你要添加另外一个 **DIV** 标签去围住日志标题和日志内容？

增加这个 **DIV class="entry"** 去把日志标题和日志内容区分开。而这个 **div class="post"** 是吧你的日志和其他内容区分开。

## Hello World



Lorem ipsum dolor  
nisi ut aliquip ex ea commodo conse  
officia deserunt mollit anim id est lab

Post  
number  
1

r adipisc  
e dolor ir



Lorem ipsum dolor  
nisi ut aliquip ex ea commodo conse  
officia deserunt mollit anim id est lab

r adipisc  
e dolor ir

Lorem ipsum dolor sit amet,  
nostrud exercitation ullamc  
eu fugiat nulla pariatur. E

lipisc  
ut ali  
occaeca

Lorem ipsum dolor sit amet, consec  
ea commodo consequat. [\(more...\)](#)

sed do t

## Hello World

Lorem ipsum dolor sit amet, consec  
ea commodo consequat. Duis aute i  
anim id est laborum.

Post  
number  
2

sed do t  
enderit ir

这就是这篇教程的所有内容，可能需要更多时间去消化，所以有问题请务必给我留言，下一篇我们将对付日期(**date**)，分类(**categories**)和评论(**comments**)。

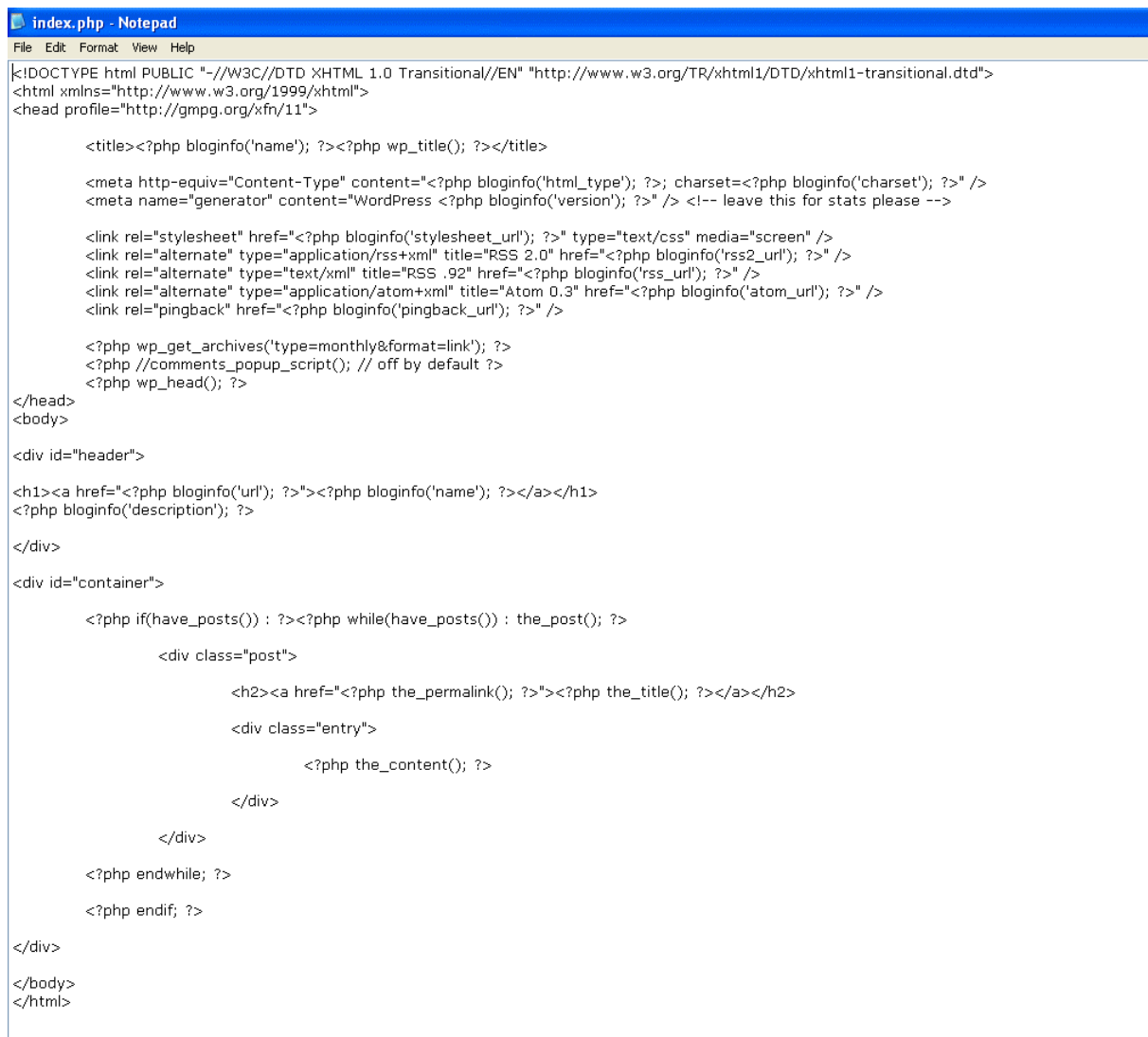
如果你看到我教程中错误或者不一致的地方，请尽快告诉我以便我能立刻改掉，另外，到目前为止，你觉得我的课程怎么样？是否需要我讲的更具体些，或者它们太长了？

## WP Theme 教程 #5c : 日志元数据

这一篇我们开始解决日志的元数据 (Postmetadata)：日期 (**date**)，分类 (**categories**)，作者 (**author**)，评论数 (**number of comments**)，以及其他隶属于日志的杂项信息。

我们现在在教程 5C，你可能已经注意到我将会把教程归纳成更大块（更少的步骤）。在开始之前，请打开 Xampp，主题文件夹，浏览器以及 index.php 文件。

先让我们复习下，现在的 `index.php` 文件应该有下面这些代码了：



```
index.php - Notepad
File Edit Format View Help
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head profile="http://gmpg.org/xfn/11">

    <title><?php bloginfo('name'); ?><?php wp_title(); ?></title>

    <meta http-equiv="Content-Type" content="<?php bloginfo('html_type'); ?>; charset=<?php bloginfo('charset'); ?>" />
    <meta name="generator" content="WordPress <?php bloginfo('version'); ?>" /> <!-- leave this for stats please -->

    <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" type="text/css" media="screen" />
    <link rel="alternate" type="application/rss+xml" title="RSS 2.0" href="<?php bloginfo('rss2_url'); ?>" />
    <link rel="alternate" type="text/xml" title="RSS .92" href="<?php bloginfo('rss_url'); ?>" />
    <link rel="alternate" type="application/atom+xml" title="Atom 0.3" href="<?php bloginfo('atom_url'); ?>" />
    <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />

    <?php wp_get_archives('type=monthly&format=link'); ?>
    <?php //comments_popup_script(); // off by default ?>
    <?php wp_head(); ?>
</head>
<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(); ?>

        <div class="post">

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

            <div class="entry">

                <?php the_content(); ?>

            </div>

        </div>

    <?php endwhile; ?>

<?php endif; ?>

</div>

</body>
</html>
```

## 第 1 步

把 [postmetadata.txt](#) 中的代码拷贝到 `<?php the_content(); ?>` 下面。（注意：在这部分，你只需要拷贝和粘贴。当我把 WordPress 主题放在一起的时候，我也只是拷贝和粘贴这部分代码。再重复一次，你不需要完全理解它，知道每部分代码集干什么已经足够了。

下面的这个屏幕截图为了适应日志的大小裁剪了一部分。注意日志元数据代码的组织 and 位置。不要把你的代码像下面的截图一样裁剪到 **1 Comment &#**。

```

        <div class="entry">

            <?php the_content(); ?>

            <p class="postmetadata">
<?php _e('Filed under&#58;'); ?> <?php the_category(', ') ?> <?php
<?php comments_popup_link('No Comments &#187;', '1 Comment &#
            </p>

        </div>

```

保存，刷新你的浏览器，你应该看到：

Lorem ipsum dolor sit amet, consectetur adipisicing elit, s  
nostrud exercitation ullamco laboris nisi ut aliquip ex ea  
eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod temp<  
ea commodo consequat. [\(more...\)](#)

Filed under: [Uncategorized](#), [Personal](#), [Sub Category](#) by Small Potato  
[2 Comments »](#)

你同样可以通过查看源代码来看日志元数据是怎样的？还记得怎么样查看源代码吗？在浏览器的菜单：[查看 > 页面源代码](#)。

解释：

**<p class="postmetadata">** 和 **</p>** - 所有的日志元数据都在一个 **class="postmetadata"** 的段落标签中，因为我想把日志元数据和日志内容区分开。如果没有段落标签，日志元数据信息将在日志内容结束的地方继续，这将没有任何间距去区别内容和日志元数据。

**<?php \_e('Filed under&#58;'); ?>** - **&#58;** 是调用冒号“:”的代码；把 **Filed under&#58;**放入**<?php \_e(' '); ?>**中不是必须的。你可以简单输入 **Filed under:**

**<?php the\_category(', ') ?>** - **the\_category()** 是用来调用你当前日志的在的所有类别的 PHP 函数。如果你把 **Filed under:** 和 **the\_category()** 放在一起，你可以得到：**Filed under: Name of category 1, Name of category 2**。**the\_category()** 中的逗号是用来区分类别名。返回日志元数据的屏幕截图，你可以注意到在类别连接中的逗号。

**<?php \_e('by'); ?>** - 和 **Filed under:**一样。如果你创建私人用的的主题，**by**外面的 **\_e()** 不是必须的。**\_e()** 是用来创建可以翻译的主题，如果主题被来自不

同国家的上百人使用的话，这是非常重要的。如果你是创建公共使用的主题，最后加上 `_e()` 以便你的主题可翻译化。

`<?php the_author(); ?>` - 它是输出当前日志作者的名字。

`<br />` - 如果你想要一个空行，又不想用段落标签来产生行间距，使用 `BR`。注意斜线 `/`。这是能自我关闭的标签。

`<?php comments_popup_link('No Comments &#187;', '1 Comment &#187;', '% Comments &#187;'); ?>` - 当弹出留言的功能激活的话，`comments_popup_link()` 调用一个弹出的留言窗口。如果没有激活，`comments_popup_link()` 则只是简单的显示留言列表。**No Comments &#187;** 是在没有留言的时候显示的。**1 Comment &#187;** 是用于当你刚好有 1 条留言时候。**% Comments &#187;** 是用于当你有多于一条留言的时候。比如：8 Comments »。百分号 `%` 用来显示数字。`&#187;` 是用来显示一个双层箭头 »。

`<?php edit_post_link('Edit', '|', ''); ?>` - 这个只有当你以管理员身份登录的时候（译者注释：以编辑者，Editor 登录也可以）才可见。`edit_post_link()` 只是简单为你显示一个可以用来编辑当前日志的编辑链接，这样可以让你不必去去管理界面搜寻该日志而能直接编辑。`edit_post_link()` 有三个参数。第一个是用来确定哪个词你将用在编辑链接的链接标题。如果你使用 **Edit post**，那么将显示 **Edit post** 而不是 **Edit**。第二个参数是用来显示在链接后面的字符，在这里是竖线 `|`，代码就是**&124;**。第三个参数是用于显示在编辑链接后面的字符，在这里没有使用。

登录 WordPress 之后，再返回到首页就可以看到“**Edit**”的链接和一条竖线。

## WP Theme 教程 #5d : Else , 日志 ID , 链接标题

这篇课程将讨论其他 3 个可以增加日志中的元素。**Else**, **post ID**, 和 **链接的 title 值**。尽管它们是可选的，但是你几乎可以在我每一个免费的主题中都能找到。

开始之前，不要忘记启动 Xampp。

### 第 1 步

在 `<?php endwhile; ?>` 的下面输入以下代码：

```
<?php else : ?>
```

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

```
<h2><?php _e('Not Found'); ?></h2>
```

```
</div>
```



大致如下：

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

保存，但是你应该注意不到任何变化。我们应该返回教程 #5，主循环，去解释你刚才上面输入的是什么？

这里就是主循环的部分代码：

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

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

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

第一，**if(have\_posts())** 检查你的博客是否有日志，第二，**while(have\_posts())** 执行 **the\_post()** 去调用你的日志。而 **Else** 是当你的博客完全没有日志的时候执行的。想一下 **while()** 和 **endwhile;** 应该嵌套在 **if()** 和 **else :**之间。这就是为什么 **<?php else : ?>** 应该在 **<?php endwhile; ?>** 之后。

现在你知道什么是 **else**，当你没有任何日志或者当它找不到任何日志的时候，你想告诉 WordPress 怎么处理呢？让 WordPress 显示错误信息 **Not Found**，或者其他任何你想要的东西。你可以下载我的任一款主题，看一下它的 **index.php** 文件怎么写的。

在上面的例子中，**Not Found** 错误信息是在 **<?php \_e(' '); ?>** 之中。如我上一篇所说，这不是必需的。

整个信息和代码 **Not Found** 外面有 **<h2>** 和 **</h2>**。这个同样也不是必需的。你可以简单使用：

```
<div class="post">
Not Found
</div>
```

但是，给这个错误信息使用上 **<h2>**（子标题）标签能够使它更明显，让你访问者注意到这个页面上没有任何东西。

那么 `<div class="post">` 和 `</div>` 用来做什么的呢？恩，你肯定不想你的错误信息在“茫茫蛮荒之地”之间滞留，对不？你用 `<div class="post">` 和 `</div>` 标签围住每篇日志。所以同样，尽管是错误信息不是真正的内容，但是它其实是像日志的文本。

## 第 2 步

增加 `id="post-<?php the_ID(); ?>"` 到 `<div class="post">`

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

保存，然后刷新你的浏览器。然后 **查看 > 页面源代码**。你现在会发现现在每篇日志都附加上了一个数字或者说是日志 ID。`the_ID()` 只是调用每篇日志的 ID；仅此而已。

为什么使用它呢？这是用来定制个别的日志的面貌。后面，当你使用 **style.css** 文件去告诉你的主题日志将看起来怎么样。如果通过给每篇日志附加唯一的 ID，你就可以针对单独的一篇日志，使得它和其他日志看起来不一样。如果没有 ID，你将没有办法通过 **style.css** 文件使它和其他日志不一样。

你如何同时把 **class** 和 **id** 赋给同一个 **DIV** 标签呢？**DIV** 是标签，**class** 是一个属性，**id** 也是是一个属性。每个标签能拥有多个属性，如 **DIV** 就可以同时有 **class** 和 **id** 这两个属性。（注释：**id** 是一个 XHTML 属性。`the_ID()` 是 PHP 函数。他们是不同的，）

## 第 3 步

增加 `title="<?php the_title(); ?>"` 到日志的标题链接。

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

保存并刷新浏览器。然后再去查看你的源代码。当查看它的源代码，查找任何日志的**标题链接**。如果日志的标题链接是 **Hello World**，那么他的左边应该有 `title="Hello World"`。

`title=""` 是 `<a>`（链接）标签的另一个属性。在双引号中的是链接的描述。在这里，每篇日志的标题也是链接的描述。这就是为什么要再次使用 `the_title()` 这个 PHP 函数。

如果你不使用 `the_title()` 作为 `title=""` 的值，那么每篇日志标题链接将会有同样的描述。举个例子，如果用 `title="Click me"` 取代 `the_title()`，每篇日志标题链接都将会用 **Click me** 作为描述。

现在，返回你的网页。把你光标移到任一篇日志标题连接，描述信息将会弹出，那就是刚刚增加的。增加描述到你的链接是非常有用的，当你其他站点需要扫描你的博客的时候。如 [Technorati.com](http://Technorati.com)，每次你发表日志的时候，WordPress 通知 [Technorati](http://Technorati.com) 和其他网站你的博客已经更新了。[Technorati](http://Technorati.com) 然后就会来到你的博客，扫描它，并索引得到一个你日志的摘要，这其中会包括你链接标题的描述。

## WP Theme 教程 #5e : 日志导航链接

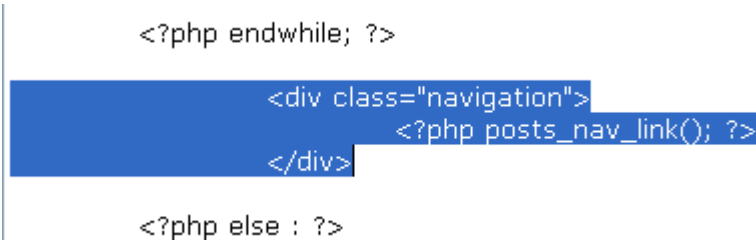
在绝大多数的 WordPress 博客的底部中，都会有 **Next Page** 或者 **Previous Page** 这些导航链接。你可以通过 WordPress 的模板系统中的 `posts_nav_link()` 这个函数调用这些链接。在这篇中，只有一个步骤！

真的需要我再次告诉你一次去打开 Xampp，主题文件夹，浏览器和 `index.php` 文件吗？😄

### 第 1 步（只有一步）

在 `<?php endwhile; ?>` 和 `<?php else : ?>` 之间添加如下代码：

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



```

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

注意到上面截图中的缩进，我一般使用 `tab` 键产生缩进来组织代码。

**<div class="navigation">** - 开始一个名字为 **navigation** 的 DIV 标签来围住后一页和前一页的链接区域。

**<?php** - 开始 PHP 代码

**posts\_nav\_link()** - 调用后一页和前一页的链接。

**;** - 停止调用。

?> - 结束 PHP 代码

</div> - 结束名字为 **navigation** 的 DIV 标签。

效果如下：

Filed under: [Uncategorized](#), [Personal](#), [Sub Category](#)  
[1 Comment](#) »



保存并刷新，查看后一页或者前一页的链接。默认情况下，如果你没有超过 10 篇日志的话，是不会显示导航链接。如果你没有超过 10 篇日志，依然想看到导航链接，登录到管理界面，选择 **Options > Reading**，然后把它设置为比你所有日志少一篇。如，你有 6 篇日志，就设置为 5。

如何定制化 **posts\_nav\_link()**：

就像一些在 **postmetadata** 课程的函数中，你可以给这个函数 3 个参数，分别为你想显示在后一页和前一页链接的中间，前面和后面的任何字符，如下：

**<?php posts\_nav\_link('in between','before','after'); ?>**

第 1 个参数是显示在后一页和前一页链接的中间。第 2 个参数显示在前面。第 3 个参数显示在后面。

这里是一个定制化 **posts\_nav\_link()** 的例子：



## WP Theme 教程 #6 : 侧边栏

是否你已经很期待学习侧边栏（Sidebar）了呢？可能第一眼，感觉侧边栏很难，其实它根本不难，一旦你掌握了它的结构，就能很快编码和样式化它。

在开始侧边栏之前，这是现在 [index.php](#) 文件的样子。

不要忘记打开 Xampp.....

## 第 1 步

让我们创建一个名字为 **sidebar** 的 DIV 标签用来围住侧边栏中的所有东西。  
在 **container** 的后面和 **</body>** 标签的前面输入以下代码：

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

```
</div> closing tag for the box with id named container
<div class="sidebar">
</div>
</body>
</html>
```

## 第 2 步

在新的 **sidebar** 的 DIV 标签中创建一个新的无序列表。

**<ul>** - 开始无序列表  
**</ul>** - 结束无序列表

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

## 第 3 步

增加一个列表元素（**LI**）到无序列表（**UL**）的中间并把一个子标题添加到这个列表中。

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

```
</li>
```

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

注意添加制表符到<li> 和 </li> 标签之前为了组织代码。

<li> - 开始列表元素

<h2> - 开始子标题

<?php \_e('Categories'); ?> - 输出字符 **Categories**

</h2> - 结束子标题

</li> - 结束列表条目

再次重复一次，如果你只是为自己创建主题，你不需要将 **Categories** 放入 <?php e(' '); ?>。使用<h2>Categories</h2> 作为你的子标题也完全没有问题。

保存 index.php 文件并刷新你的浏览器。你应该看到 **Categories** 子标题结构应该这样：

— [Next Page »](#)



子标题前面的小圆点指明这个子标题是在一个列表元素中（LI）。如果你的无序列表（UL）有两个列表元素，那么将有两个小点。

## 第 4 步

在列表条目中添加下面代码

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

```
<ul>
  <li><h2><?php _e('Categories'); ?></h2>
    <ul>
      <?php wp_list_cats('sort_column=name&optioncount=1&hierarchical=0'); ?>
    </ul>
  </li>
</ul>
```

这里是上面代码的解释：

**<ul>** - 开始另一个无序列表

**<?php wp\_list\_cats(); ?>** - 调用分类链接列表

**</ul>** - 结束无序列表

保存并刷新浏览器。下面是我的分类链接列表的样子：

## • Categories

- ◊ [Personal](#) (11)
- ◊ [Sub Category](#) (10)
- ◊ [Uncategorized](#) (11)



你默认的分类是 **Uncategorized**。如果你没有把日志发布到多个分类下面，那么你的列表链接列表应该是只有一个链接 **Uncategorized**。

更进一步的解释：

- **sort\_column=name** - 把分类按字符顺序排列
- **optioncount=1** - 显示在每个分类下面的日志数
- **hierarchical=0** - 不把子分类放到子列表元素中，这就解释了为什么我的子分类 链接是列在列表中第一级。
- **&** - 每次你增加另一个参数的时候，你需在它之前要输入 **&** 用来把和现有的参数区分开。如 **&** 在 **sort\_column** 和 **optioncount** 之间。

为什么你不把 **<?php wp\_list\_cats(); ?>** 放入 **<li>** 和 **</li>** 标签中呢？

当你使用 **wp\_list\_cats()** 这个函数调用链接列表函数的时候，它会自动附上一组 **<li>** 和 **</li>**（列表条目）标签在每个链接的左右。查看**页面源代码**；你可以看到每个连接的周围都已经有一组列表元素的标签。

当你处理侧边栏，无序列表和列表匀速的时候，这是非常重要去记得：

**规则 #1：**按顺序关闭所有标签。

### The right way to close:

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

### The wrong way to close:

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

## WP Theme 教程 #6b : 页面链接列表

现在你已经熟悉了侧边栏的结构，接下来我们将继续修改侧边栏，完成**页面链接**（**Page-link**）列表。当完成常规的侧边栏之后，我将向你展示如何窗体化（**widgetize**）你的侧边栏。

在分类链接上面添加以下代码：

**<?php wp\_list\_pages(); ?>**

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

保存并刷新浏览器。效果如下所示：



- [illegible]

## • Categories

在默认情况下，你只有一个页面链接，就是 **About** 链接。我在我的离线的 **blog** 增加了更多页面和子页面，这就是为什么我有四级页面链接。

到浏览器的菜单 查看 > 页面源代码 可以看到 `wp_list_pages()` 能够产生完整的结构以及代码，如下：

```
<li>Pages
  <ul>
    <li><a href="#">Your Link</a></li>
    <li><a href="#">Your Link 2</a></li>
  </ul>
</li>
```

第一，它把所有东西放入列表元素标签（**LI**），第二，它给你的列表一个名字，**Pages**。第三，它在页面标题下，列表条目内另外增加一个无序列表（**UL**）。第四，它把每个链接放入到 `<li>` 和 `</li>` 标签之间。

在上面的截图中，注意到“**Pages**”这个列表标题和“**Categories**”这个分类链接标题的大小不匹配。

如何使它们匹配呢？添加 `'title_li=<h2>Pages</h2>'` 到 `wp_list_pages()` 作为参数。

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

保存并刷新浏览器结果如下:

• Pages

- [illegible]

- Categories

**title\_li** 是一个用来客户化页面链接列表的标题的参数。**<h2>Pages</h2>** 是 **title\_li** 这个参数的值

进一步客户化:

在我的例子中，我有四级页面链接。由于布局或者设计的原因使得不能在侧边栏处理那么多级别的链接。为了限制显示列表的层数，增加 **depth** 这个参数到函数 **wp\_list\_pages()** 并把它设置为 **3**：

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

注意，我添加了 **depth=3&** 而不是仅仅 **depth=3**。这个 **&** 在这儿用于把 **depth** 和 **title\_li** 这两个参数区分开。（如果你只有一个 **about** 页面链接，你将不会注意有什么不同。）

这里是我的列表的不同之处：（对比这个截图和上面的截图。）

## ◆ Pages

- ◇ [About](#)
- ◇ [Parent 1](#)
  - [Child 1 Name iiii](#)
    - [Grandchild 1](#)
    - [Grandchild 2 iiii](#)
  - [Child 2 Name](#)
  - [Child 3 Name](#)
  - [Child 4 Name](#)
  - [Child 5 Name](#)
- ◇ [Parent 2](#)

## WP Theme 教程 #6c : 存档和链接列表

这篇课程将比较轻松。你将学到如何调用归档链接列表和 **blogroll** 链接列表，下一篇将是日历和搜索框。

在开始之前，我想解释下为什么我把我的课程分成比较小的节。你所学的一切都需要几个月时间来消化！所以当我第一次向你展示的时候，你能够理解所有的东西是非常重要的，这样你就不用会去重新学习。

### 第 1 步 - 增加归档链接。

在侧边栏区域的 **Categories** 列表下面输入以下代码：

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

给上面代码增加制表符空间是用于产生缩进组织代码。看下你的代码是否和下面一样：

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

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

保存，刷新浏览器。结果如下所示：

## • Categories

- ◊ [Personal](#) (11)
- ◊ [Sub Category](#) (10)
- ◊ [Uncategorized](#) (11)

## • Archives

- ◊ [October 2006](#)

发生什么了？

你使用了 `wp_get_archives()` 这个 PHP 函数，并用了 **type** 这个参数以及 **monthly** 作为它的值按月调用存档列表。

- `<li>` - 开始列表元素
- `<h2>` - 开始子标题
- `<?php _e('Archives'); ?>` - 子标题文本
- `</h2>` - 结束子标题
- `<ul>` - 开始在子标题下，在列表标题中的无序列表
- `<?php wp_get_archives('type=monthly'); ?>` - 按月调用存档列表链接，并把每个链接放入 `<li>` 和 `</li>` 标签中。如果你查看你的源代码，你会看到 `wp_get_archives()` 为每个链接产生了列表元素（LI）标签，就像 `wp_list_cats()` 这个函数一样。
- `</ul>` - 结束在子标题下的无序列表
- `</li>` - 结束列表元素

## 第 2 步：增加 blogroll 链接

在存档链接列表下输入以下代码：

<?php get\_links\_list(); ?>

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

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

保存，刷新，结果如下：

## • Archives

◊ [October 2006](#)

## • Blogroll

◊ [Mike](#)  
◊ [Dougal](#)  
◊ [Matt](#)  
◊ [Ryan](#)  
◊ [Alex](#)  
◊ [Michel](#)  
◊ [Donncha](#)

默认情况下，我的 blogroll 和你的是没有什么不同，这里是它在源代码中的样子：

```
<li id="linkcat-1"><h2>Blogroll</h2>
<ul>
<li><a href="http://zed1.com/journalized/">Mike</a></li>
<li><a href="http://dougal.gunters.org/">Dougal</a></li>
<li><a href="http://photomatt.net/">Matt</a></li>
<li><a href="http://boren.nu/">Ryan</a></li>
<li><a href="http://www.alexking.org/">Alex</a></li>
<li><a href="http://zengun.org/weblog/">Michel</a></li>
<li><a href="http://blogs.linux.ie/xeer/">Donncha</a></li>
</ul>
</li>
```

上面的代码完全没有被组织，因为它们是由函数 `get_links_list()` 产生的，就像上一篇所学的函数 `wp_list_pages()` 一样，然而它遵循规则 #1，按正确顺序关闭所有的东西。我已经圈出了元素和无序列表的标签让你看得更明显。

## WP Theme 教程 #6d : 搜索框和日历

尽管这篇的题目是 **搜索框 (Search Form)** 和 **日历 (Calendar)**，我同样也会介绍 **元数据 (Meta)**。让我们在这一篇结束常规的侧边栏，然后将在下一篇指出如何窗体化 (widgetize) 化侧边栏，怎么样？好吧，让我们开始...

### 第 1 步：增加搜索框

在我给你需要输入的代码之前，打开记事本，然后保存该空白文件为 **searchform.php**（当然是和 **index.php** 在同一个文件夹下）。把 [searchform.txt](#) 中的内容拷贝到 **searchform.php**。就是这么多，从现在开始，让 **searchform.php** 文件保持单独。

在 **index.php** 文件，在侧边栏的最顶部输入以下代码：

```
<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&option=sort_order'); ?>
    </ul>
  </li>

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

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

保存，然后刷新浏览器，结果如下：



刚才发什么了呢？

- **<li id="search">** - 开始一个名字为 **search** 的列表元素，你给它了一个 ID，你就能够以后样式化它。
- **include()** - 导入任何你想导入的文件。这和使用 WordPress 模板函数去调用东东是不同的，因为 **include()** 只是简单导入已经存在的文件。这里是调用在 **searchform.php** 文件中的代码。被导入的信息应该在一个博客上基本不会被改变的。也就是说，我的搜索框应该看起来很像你的。
- **TEMPLATEPATH** - 主题文件夹的位置，**wp-content/themes/tutorial**
- **'/searchform.php'** - 位置和文件的名字，**/searchform.php**
- 在 **TEMPLATEPATH** 和 **"/searchform.php"** 中间的点把他们连接起来，所以最终得到：  
**wp-content/themes/tutorial/searchform.php**
- **</li>** - 结束列表元素

注意，搜索框不像分类，归档，页面或者 Blogroll 一样有子标题。如果你愿意你也可以给它一个子标题，但是基于保持你的常规侧边栏和 **widgetized** 版本（我们将在下一篇讲述）的侧边栏尽可能接近的考虑，不要给它子标题。

## 第 2 步：增加日历

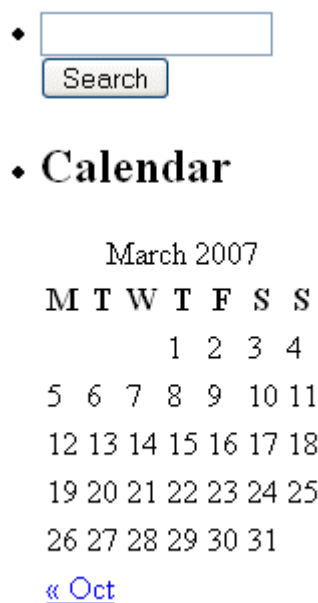
在搜索框或者页面链接列表下面输入以下代码：

```
<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>
```

保存并刷新浏览器，结果如下：



发生了什么？

- **<li id="calendar">** - 开始一个 ID 为 “Calendar” 的列表元素
- **<h2>** - 开始一个子标题
- **<?php \_e('Calendar'); ?>** - 输出 **Calendar** 这个词
- **</h2>** - 关闭子标题
- **get\_calendar()** - 使用 `get_calendar()` 这个 WP 函数调用日历
- **</li>** - 结束列表元素

这样日历就完成了

### 第 3 步：增加元数据

在 `get_links_list()` 函数下输入以下代码：

```
<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>
```

保存并刷新浏览器，结果如下：

（如果你没有登录 WordPress）

## • Meta

- ◊ [Register](#)
- ◊ [Login](#)

（如果你已经登录）

## • Meta

- ◊ [Site Admin](#)
- ◊ [Logout](#)

那么这是怎么回事呢？

你开始一个列表元素（LI），跟着是一个子标题（H2） **Meta**。在子标题下，你嵌入了一个无序列表（UL）。最后把每个链接都放入了列表元素中（LI）。

**wp\_register()** 这个函数能产生一组 **<li>** 和 **</li>** 标签，如果你没有登陆，它显示注册（**Register**）链接，当你登录的时候，它显示的是 **管理站点（Site Admin）** 的链接。

**wp\_loginout()** 不会产生自己的列表元素标签，所以你需要手工输入列表元素标签，当你没有登录的时候，你得到的是 **登录（Login）** 的链接，当已经登录的时候，你得到的是**登出（Logout）** 链接。

到目前为止，**wp\_meta()** 没有做任何事情，他在网页上和源代码中都不会产生东西，现在不要考虑 **wp\_meta()**，实际上你已经在使用它了。

到此为止，你应完成 **Meta** 并最终完成了常规的侧边栏。

## WP Theme 教程 #6e : 窗体化侧边栏

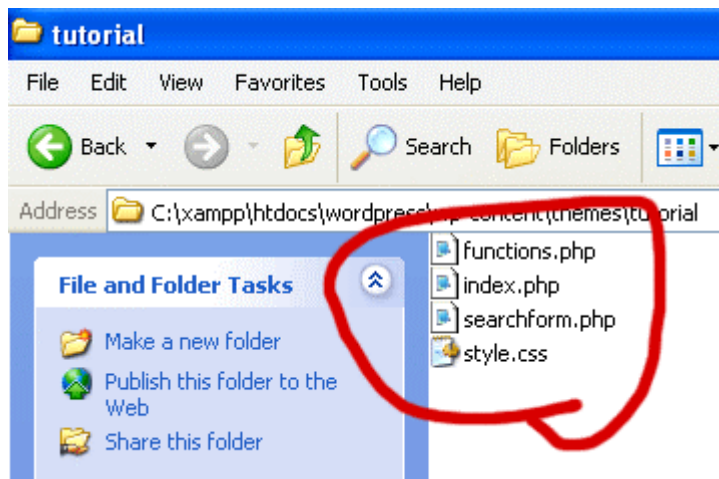
一个适合 **Widget** 的侧边栏或者说是窗体化（**widgetized**）的侧边栏几乎是 **WordPress 2.0** 以后的主题标准。但是首先，什么是窗体化（**widgetizing**）呢？简单的说，窗体化就是让侧边栏适合 **Widget** 这个插件（译者注：**WordPress 2.2** 已经内置了 **Wodgets**），这个插件让你非常容易就能够重新整理侧边栏的模块。

举个例子说，不用去修改侧边栏的代码去更改分类和存档的位置，你只需要简单把分类和存档列表拖到它们的位置即可。

### 第 1 步：创建 **functions.php** 文件

打开记事本，然后把空白文件保存为 **functions.php**。把 [functions.txt](#) 文件中所有的内容拷贝到 **functions.php** 中。保存并关闭。

回顾一下，现在在“**tutorial**”主题文件夹下应该有 4 个文件。



### 第 2 步：窗体化侧边栏

直接在侧边栏的第一个 **<ul>** 标签输入以下代码：

```
<?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(); ?>
    </li>
</ul>
```

直接在 `</ul>` 标签之前输入以下代码：

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

```
<?php endif; ?>

</ul>

</div>

</body>
</html>
```

保存 `index.php` 文件，你将在网页上看不到任何变化直到你装了 **Widget** 插件。  
(译者注：WordPress 2.2 之后就不用安装这个插件了)

## WP Theme 教程 #7 : 尾部

关于 **尾部 (footer)** 的这篇教程将会很简单。只要在侧边栏下增加个 **DIV** 标签，然后输入一些版权信息。其实你完全可以不用我说明就能自己做，去试下吧。然后返回这里再仔细检查下。

### 第 1 步：增加个 **DIV** 标签

在侧边栏的 **DIV** 标签下输入以下代码：

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

**`</div>`**

```
</div>end of Sidebar
```

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

```
</div>
```

```
</body>
```

```
</html>
```



第 2 步：把尾部的文本放入段落标签中，你可以输入任何你想要的东西，这里是我的：

```
<p>
```

```
Copyright &#169; 2007 <?php bloginfo('name'); ?>
```

```
</p>
```

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

```
<p>
```

```
Copyright &#169; 2007 <?php bloginfo('name'); ?>
```

```
</p>
```

```
</div>
```

保存并刷新浏览器，结果如下：

## • Archives

◊ [October 2006](#)

Copyright © 2007 Demo Theme Development

如果你没有使用上面的尾部文本，&#169; 用于显示版权符号。还记得在头部代码时候使用的 **bloginfo()** 函数吗？我在尾部再次使用，“name”是用于调用博客标题，而“url”调用博客的地址。

如果你想你的博客标题成为一个链接，查下头部就知道怎么做了。

## WP Theme 教程 #8：如何验证

在进入学习 CSS 和 修改 **style.css** 文件之前，你需要学习如何验证你的代码。简单说，验证（Validate/Validating/Validation）就是确认代码没有错误。在

开始这个系列教程之前，我给了你两个链接，[XHTML Validator](#) 和 [CSS Validator](#)。在这篇，你将学到 XHTML 验证器。

我会保持这篇非常简短，因为你需要更多的时间去理解你所学到 XHTML 和 PHP 知识。不要一直往前学习哦，花些时间在脑海中回想和回顾下代码，你如何开始 DIV 标签的？ID 和 Class 之间有什么区别呢？分别哪个 PHP 函数可以用来调用博客的标题，博客的描述呢？你回想的越多，你会记得更多。

这篇之后，你最好准备好，我们将开始 CSS，我将深入研究它，因为我们将会有 80% 的时间花在和 style.css 文件打交道。

对于这篇的课程，你只需打开 **Xampp Control** 和浏览器（进入 <http://localhost/wordpress>）。

在浏览器，到菜单查看 > 页面源代码。

全选。然后拷贝所有的源代码。

把浏览器导向 [XHTML Validator](#)。

到页面的底部，把源代码粘贴到 **Validate by Direct Input** 框中。

**Validate by Direct Input**

Input the markup you would like to validate in the text area below:

```
</div>

<div id="footer">
<p>
Copyright © 2007 <a href="http://localhost/wordpress">
Development</a>
</p>
</div>

</body>
</html>
```

点击 **Check** 之后，验证器会检查你的代码，然后反馈给你结果页面。如果反馈回来的信息是绿色的，那么你的代码没有错误。

Help & FAQ Feedback

**Result:** Passed validation

**File:** upload://Form Submission

**Encoding:** utf-8

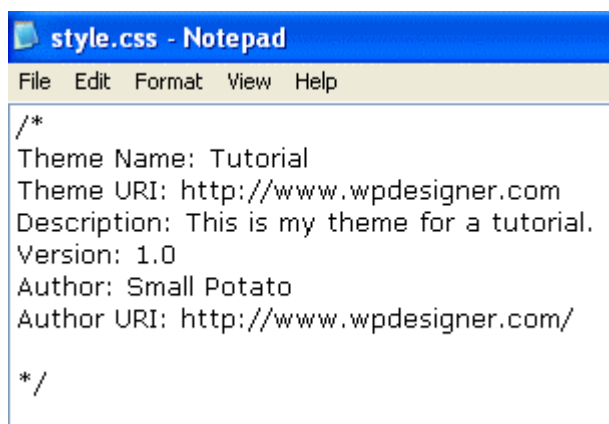
**Doctype:** XHTML 1.0 Transitional

**Root Namespace:** <http://www.w3.org/1999/xhtml>

## WP Theme 教程 #9 : Style.css 和 CSS 介绍

学习 CSS 最好的方法就是去用它，不像 XHTML 和 PHP，你不需要接触模板的核心文件，你同样不要需要理解任何基本概念，只要去用它。试用和错误是你前进的方法。

在我们开始之前，你应该已经有些内容在你的 **style.css** 文件了。让我们现在找出这部分信息是干什么的？



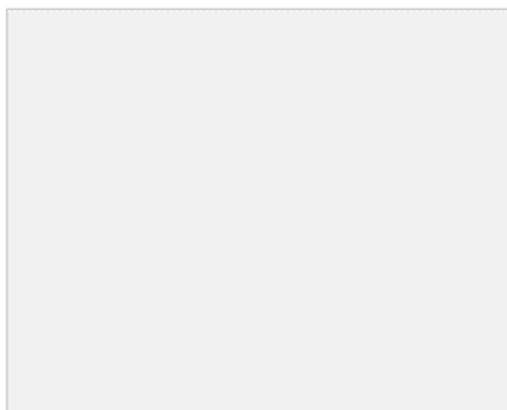
- 第一行显而易见就是主题的名字。
- 第二行是这个主题的地址，如果你的主题只是私用的而不准备发布的话，那就不用考虑它。
- 第三行是主题的描述。
- 第四行是版本号，这是非常重要的，特别是当你公开发布你主题新版本的时候。
- 第五和第六行分别是你的名字和主页。

在主题信息两边的 `/*` 和 `*/` 符号是为了阻止主题的信息影响该文件的其他内容。这是 CSS 的注释。当输入 CSS 代码去样式化你的网页的时候，你可能

想在这里增加些注释使得能够往后追踪代码。但是你并不想你的注释影响你的代码，所以你使用 `/*` 和 `*/` 这一对符号使得你的注释无形。

下面是处理后的主题信息

### Tutorial 1.0



This is my theme for a tutorial.

## 第 1 步

- 打开 Xampp，主题文件夹，FireFox，IE 浏览器和 `style.css` 文件。
- 在两个浏览器的地址栏都输入：`http://localhost/wordpress`

从这里开始，你需要同时在 FireFox 和 IE 上测试你的主题，不同的浏览器对 CSS 的代码解释是不同的。如果能够在尽可能多的浏览器上和尽可能多的操作系统上测试你的主题是最好的（Safari, Opera, Linux, Netscape 等等）。如果你和我一样懒，那就只在 FireFox 和 IE 上测试你的主题吧。

（注释：在这篇中你不必在记事本中打开 `index.php` 文件。如果你不能用记事本打开 `style.css` 文件，右键点击该文件，选择属性，点击更给打开方式，选择记事本。）

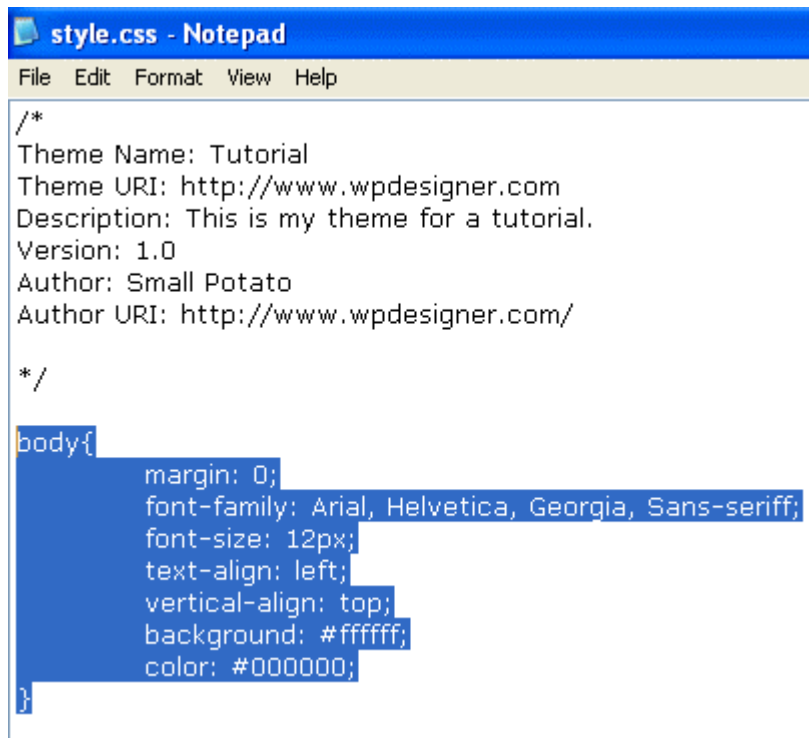
## 第 2 步

在 `style.css` 文件中输入以下代码：

```
body{  
margin: 0;  
font-family: Arial, Helvetica, Georgia, Sans-serif;  
font-size: 12px;  
text-align: left;
```

```
vertical-align: top;  
background: #ffffff;  
color: #000000;  
}
```

像 XHTML 和 PHP 一样，通过制表符增加缩进来组织代码：



保存 **style.css** 文件，刷新 两个浏览器 **Firefox** 和 **Internet Explorer** 查看变化。（现在开始你将同时在这两个浏览器上工作。）

把 **body{ }** 看作一个标签，然后它里面所有的东西看作属性和值，和处理 XHTML 时一样。**{** 是开始符，**}** 是结束符。在 **{** 和 **}** 之间，冒号意思是开始而分号意思是结束。（我在涉及到 XHTML，PHP，CSS 的时候都使用标签，属性和值这些术语是为了保持简单，实际上 PHP 和 CSS 有不同术语。如参数（parameters），选择器（selector）和属性（property）。）

在我们继续之前，我需要解释下为什么使用 **body{ }**（CSS 选择器），是因为你是在样式化网页的绝大基本部分（或者说是总体部分），**<body>** 标签。你不会样式化 **<head>** 因为这个标签没有东西需要样式化。你网页上展示的绝大部分的东西是在 **<body>** 和 **</body>** 标签之间。

然后，在后面你会样式化 ID 为 **header** 的 DIV 标签。

进一步的解释：

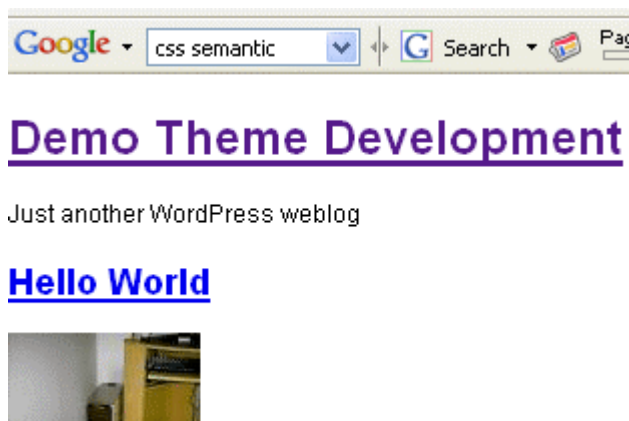


**margin: 0;** 处理 `body` 标签的默认的页边空白，如果你要页边空白或者更大的页面空白，把 0 改成 10px, 20px 或者其他。PX 意思是像素。每个像素使你电脑屏幕的一个点。当你的页边空白是 0 的话，就不需要后面跟上 px。

在下面的图片中，红色高亮的区域就是应用于 `body` 标签的默认的页边空白。



当给其样式化为 **margin: 0;**，下面是没有页边空白的相同页面：



**font-family: Arial, Helvetica, Georgia, Sans-serif;** 为你的网页或者网站选择使用哪种字体。这些字体中的第一个，**Arial** 是可替换的，如果你的用户没有在他们的电脑上安装 **Arial** 这种字体，**style.css** 文件就会寻找 **Helvetica**，然后是 **Georgia**，再接着是 **Sans-serif**。你可以在字体文件夹（我的电脑 > 系统盘 > Windows 下面）找到你的字体列表。

**font-size: 12px;** 显而易见是字体大小。可以把它改大或改小以查看变化。

**text-align: left;** 让你的文本向左对齐。把它改成 **text-align: right;** 去查看不同之处。

**vertical-align: top;** 使得所有的东西从上面开始。如果是中部或底部排行你的 `body` 标签，所有东西将会向下推。

**background: #ffffff;** 意思是白色背景。**#ffffff** 是白色十六进制代码。**#000000** 是黑色十六进制代码。

**color: #000000;** 意思是你的文本颜色是黑色。

如果你想向前更进一步或者自己学习 CSS，最好的地方是 [w3schools.com](http://w3schools.com)

继续昨天介绍 CSS 的课程，我们今天将介绍如何着色和十六进制颜色代码。

颜色属性，跟着的是一个十六进制代码，是用于给文本上色。如 **body { color: #000000; }** 意思是你页面的 body 的所有文本将是黑色的。

背景颜色属性，跟着的是一个十六进制代码，是给除背景上色。如 **body { background: #ffffff; }** 意思是为 body 上白色背景。

## 十六进制代码

- 每个十六进制代码前都有 # 号，然后跟着六位数字。这些数字的范围是从 #ffffff（白色）到 #000000（黑色）。
- #ffffff, #eeeeee, #dddddd, #cccccc, #bbbbbb, #aaaaaa, #999999, #888888, #777777, #666666, #555555, #444444, #333333, #222222, #111111
- 前两位表示红色，第三和第四代表绿色，而最后两位代表蓝色。**#ff0000** 是红色（red）。**#550000** 是暗红色（dark red）。**#220000** 是更黑色的红色（darker red）。**#00ff00** 是绿色（green）。**#0000ff** 是蓝色（blue）。等下，难道黄色不是主要的颜色吗？哪个十六进制代码是黄色呢？**#ffff00** 就是黄色（yellow）。**#ff00ff** 是紫色（violet）。

### 第 1 步：

在 **body{ }** 选择器下输入以下代码：

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

A screenshot of a code editor with a blue background. The text is white and shows the CSS code: `a:link, a:visited{` on the first line, `text-decoration: underline;` on the second line, `color: #336699;` on the third line, and `}` on the fourth line.

- 这些代码是干吗用的？给你所有的链接都加上下划线的（**text-decoration: underline;**）和上了蓝色（**color: #336699;**）。这是

不同色调的蓝色，但是它确实是蓝色因为最后两个数字（代表蓝色）是最高值的数字。

- **a:link** 用于样式化链接。当你想把一个词转变为链接的时候，用什么实现呢？使用 **<a>** 和 **</a>** 这对标签，因此样式化链接就是样式化 **a:link**。

- **a:visited** 用于样式化已经访问过的链接。
- 另外一种输入方式：

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

和

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

- 当给 **a:link** 和 **a:visited** 这两个选择器应用 **text-decoration: underline;** 和 **color: #336699;** 这两个属性的时候。可以使用逗号来区分。

## 第 2 步：

在 **a:link, a:visited{ }** 下输入以下代码：

```
a:hover{
text-decoration: none;
}
```

这些代码是干吗用的呀？ 你当你把指针移到链接上面时候下划线消失。

如果你不想在默认情况下有下划线而是在当你把指针移到链接上面的时候出现下划线，那么就在 **a:link** 和 **a:hover** 之间交换下 **text-decoration:** 的值。

如果你想更改你链接悬停时的颜色，那么就增加 **color:** 和任何你想要的十六进制代码，如：

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

## WP Theme 教程 #11 : 宽度和布局

这篇将涉及如何设置每个 DIV 的宽度和布局排版。同样也会向你展示如何让你的主题显示正确，并同时在 Firefox 和 IE 显示相同。

在我们开始之前，打开下面这些东东：**Xampp Control**，主题文件夹，**Firefox** 浏览器，**IE** 浏览器，**index.php** 和 **style.css** 这两个文件。

### 第 1 步：

第一件你要去做的事情就是你要决定你主题的总体宽度。我们使用 **750px**（750 像素）；每个 100 像素是 1 英寸。你的主题取决于你绝大多数博客访问者的查看规格。你需要避免的是使用 **900px** 的主题，如果你的读者都大多数使用 **800px × 600px** 的屏幕，这样的话，你的 **900px** 主题将会有 **100px** 超出他们的屏幕。那是无法接受的。

不管怎样，你怎么样把你总体宽度设置为 **750px** 呢？

你需要把所有的东东放入一个 **750px** 盒子中或者 **DIV** 标签。所有东西是指包括：**header**，**container**，**sidebar** 和 **footer**。

在 **<body>** 之后增加：**<div id="wrapper">**

在 **</body>** 之前增加：**</div>**

在 **style.css** 文件中输入以下代码：

```
#wrapper{  
margin: 0 auto 0 auto;  
width: 750px;  
text-align: left;  
}
```

在 CSS，具体是在 **style.css** 中，**#** 号是你用来指向跟着 **id** 的 **DIV**。而点号是用来指向跟着 **class** 的 **DIV**。举个 **class** 的例子，如果你的代码是 **<div class="wrapper">**，那么就用 **.wrapper** 来替代 **#wrapper** 去指向 **wrapper** **DIV** 标签。

同时保存 **index.php** 和 **style.css** 文件。刷新 **Firefox** 和 **IE** 浏览器（按 **F5**）查看所做的改动。

## 进一步解释：

- **margin: 0 auto 0 auto;** 意思是（准确次序）：**0** 上页边空白，右自动页面空白，**0** 下页边空白和左自动页面空白。从现在开始，记得设置左右页边空白为自动将使得居中对齐。
- **width: 750px;** 显而易见是 750 像素。
- **text-align: left;** 是让 **wrapper DIV** 中的文本向左对齐因为你会改变 **body{ text-align: left;}** 到 **text-align: center;**

## 第 2 步：

把 **body{}** 中的 **text-align: left;** 改成 **text-align: center;**。

为什么？（我假设你使用的是 Firefox 和 Internet Explorer 6）。你的布局可能你看起来是正确的，但对于使用早前版本的 IE 用户可能不正确。还记得设置左边和右边的页边空白为自动是居中吗？但是这并不是对所有的 IE 都适用，所以 **body{ text-align: center; }** 是让 **wrapper DIV** 居中对在旧版本 IE 的一种解决方案。

（随便说一下，在 Firefox 和 IE 中文本大小是不同的，我们稍后解决。）

## 第 3 步：

让 **Header** 浮到左边并且设置它的宽度为 750px:

```
#header{
float: left;
width: 750px;
}
```

## 第 4 步：

让 **Container** 浮到左边并且宽度为 500px:

```
#container{
float: left;
width: 500px;
}
```

## 第 5 步：

让 **Sidebar** 浮到左边，宽度为 240px，并且给它灰色的背景：

```
.sidebar{
float: left;
width: 240px;
background: #eeeeee;
}
```

注意，你没有使用 **#** 号去指出 **Sidebar** DIV，而是使用点号。**#ffffff** 是白色而 **background: #eeeeee;** 是非常浅的灰色。你给侧边栏增加一个背景颜色只是去查看当增加剩下的 10 像素之后的不同之处。

## 第 6 步：

让 **Footer** 浮到左边，左右两边都没有东西，并且宽度为：750px：

```
#footer{
clear: both;
float: left;
width: 750px;
}
```

**Header** 和 **Footer** 的样式有什么区别呢？答案是 **footer{} 中有 clear: both;**。它在那儿使得 **Footer** 不能和它上面的东西（如 **Sidebar** 或者 **Container**）连接起来。

保存并刷新浏览器。

## 第 7 步：

给侧边栏增加其余的 10 像素的页边空白。现在你的侧边栏应该是下面这个样子：

```
.sidebar{
float: left;
width: 240px;
background: #eeeeee;
margin: 0 0 0 10px;
}
```

保存并刷新浏览器去查看 10 像素的空白增加到侧边栏的左边了。

**margin: 0 0 0 10px;** 具体的意思是：上边空白为 0，右边空白为 0，底部空白为 0，左边空白为 10 像素。当大小为 0 的时候，**px** 单位不是必需的。

## 第 8 步（额外的步骤）：

这是为了一旦你使用了 20 像素的页边距而不是 10 像素。20 像素的页边距可能会破坏你的布局并把侧边栏挤到页面的底部，因为一个 20 像素的页边距使得 Container 和 Sidebar 的宽度之和为 760px 而不是 750px。这额外的步骤是 Internet Explorer 的错，这个使得双倍页边距的 bug 在 Firefox 中并不存在。

为了解决这个问题，增加 **display: inline;** 到侧边栏。现在你的侧边栏应该是：

```
.sidebar{
float: left;
width: 240px;
background: #eeeeee;
margin: 0 0 0 10px;
display: inline;
}
```

这篇到此为止，如果你有什么问题就请大胆问吧，我会尽自己最大努力帮你的，不要让你仅仅成为只是来这里逛逛的常客。

这里是我的现在的 [index](#) 和 [style](#) 文件的内容。

## WP Theme 教程 #12：日志样式化和其他杂项

这篇不需要 index.php。

打开 Xampp Control，theme 文件夹，Firefox，Internet Explorer 和 style.css 文件。

### 第 1 步：

在 style.css 文件中的 **body{}** 上面输入以下代码来处理大部分页边空白和填充：

```
body, h1, h2, h3, h4, h5, h6, blockquote, p{
margin: 0;
padding: 0;
}
```

- 我们使用的是 **margin: 0;** 而不是 **margin: 0 0 0 0;**。当所有的值都是一样的话，只用一个数字就够了，对于填充的设置也是一样的。

- 保存，刷新 Firefox 和 IE。不用当心，你可以增加空白和填充到你想要它们出现的地方。

## 第 2 步：

样式化 H1 标题，在 **body{}** 输入以下代码：

```
h1{
font-family: Georgia, Sans-serif;
font-size: 24px;
padding: 0 0 10px 0;
}
```

- **font-family: Georgia, Sans-serif;** 把 H1 标题的字体从 Arial 改成 Georgia。如果没有 Georgia，网页就会寻找 Sans-serif;
- **font-size: 24px;** 意思是显而易见的。你在 **body{}** 中把字体设置为 12px，H1 和 H2 标签是不会遵守的。这就是因为题目标签遵循他们自己的规则。为了控制他们，你需要特别的去样式化它们。
- **padding: 0 0 10px 0;** 意思是 10 像素的底部填充。这是为了在你博客的标题和描述之间增加空间。

保存，刷新，结果如下：



## 第 3 步：

在 **#container{}** 下面输入以下代码：（可以在输入每块代码之后，保存并刷新去查看其中的变化。）

```
.post{
padding: 10px 0 10px 0;
}
```

（给每个 class 名字为 **post** 的 DIV 增加 10 像素的顶部和底部空白。）

```
.post h2{
font-family: Georgia, Sans-serif;
```



```
font-size: 18px;
}
```

(.post h2 不是一般的 CSS 规则。他是特别样式化在 post DIV 中的 H2 子标题。在侧边栏中的 H2 子标题就不受影响。)

```
.entry{
line-height: 18px;
}
```

(设置 entry DIV 中行距。)

#### 第 4 步:

在 **a:hover{}** 下输入以下代码:

```
p{
padding: 10px 0 0 0;
}
```

(给每个段落标签增加 10 像素的顶部填充。)

#### 第 5 步:

在 **.entry{}** 下面输入:

```
p.postmetadata{
border-top: 1px solid #ccc;
margin: 10px 0 0 0;
}
```

还记得你赋给它 class 名字为 **postmetadata** 的段落标签呢? 样式化一个特定的段落标签和样式化 DIV 是非常相同的。你可以同时给两者应用边框, 框外空白, 填充和背景。

对于 **postmetadata** 这个段落便签, 你给它增加一个灰色的边框和 10 像素顶部空白。

**border-top** 意思是仅仅顶部边框 **border-left** 意思是仅仅左边边框, 等等。如果只是单独的 **border**, 没有 **-top**, **-right**, **-bottom** 或者 **-left** 则意味着所有的边框。如 **border: 1px solid #ccc;** 意思为所有的四边都有 1 像素的灰色的边框。

## 第 6 步:

在 `p.postmetadata{}` 下输入:

```
.navigation{
padding: 10px 0 0 0;
font-size: 14px;
font-weight: bold;
line-height: 18px;
}
```

对于 **Next page** 和 **Previous page** 链接外面的 `navigation` DIV 标签, 你

- 增加了一个 10 像素的顶部填充。
- 把字体大小改成 14 像素。
- 把字体改成粗体。
- 把行高增加到 18 像素。

这篇到此为止。

## WP Theme 教程 #13 : 样式化侧边栏

在对侧边栏样式化之后, 这系列教程将几乎完成了。以后我将通过案例学习来进一步探讨 (详细解释我如何创建我的主题) 。

再一次, 你今天不需要 `index.php` 文件。打开 Xampp, 主题文件夹, Firefox, IE 和 `style.css` 文件。

## 第 1 步

在 `.sidebar{}` 下输入:

```
.sidebar ul{
list-style-type: none;
margin: 0;
padding: 0 10px 0 10px;
}
```

现在你已经为侧边栏样式化父级无序列表 (UL) 标签。所有的子 ULs 或者内嵌的 ULs 将会继承同样的样式。在这里, 它是无列表样式, 零空白和 10 像素的填充。

如下所示:

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

第二级的(或内嵌的) UL 继承了第一级 UL 的样式。如果你给了第一级 UL 应用了边框, 第二级的 UL 同样也会有个边框。

保存并刷新就可以看到列表条目现在已经没有前面的圆点了。

注意下你是如何增加顶部和底部填充的。

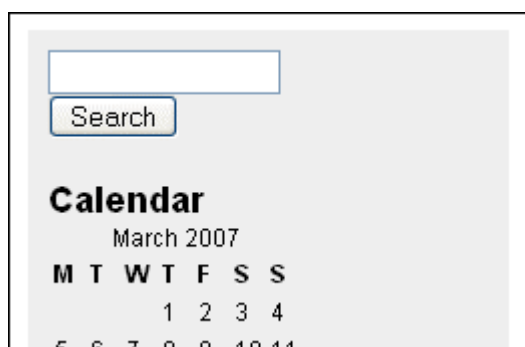


## 第 2 步:

在 `.sidebar ul{` 下输入:

```
.sidebar ul li{
padding: 10px 0 10px 0;
}
```

这是现在的填充:



为什么不在第一个地方的 **UL** 标签增加 **10 像素的填充**呢？你需要一种区分 **LI** 同其他 **LI** 的方法。在给 **.sidebar ul li** 增加填充之前，在搜索框和日历之间以及日历和页面之间是没有空间。在每个模块之间的空间就是为 **.sidebar ul li** 添加的 **10 像素顶部和底部填充**起的作用。如果给 **UL** 标签添加了四个方向的填充而不仅仅是左边和右边（**0 10px 0 10px**），那么你将会有 **20 像素的顶部填充和 20 像素的底部填充**在区分 **LI** 的时候，明白我在讲什么吗？

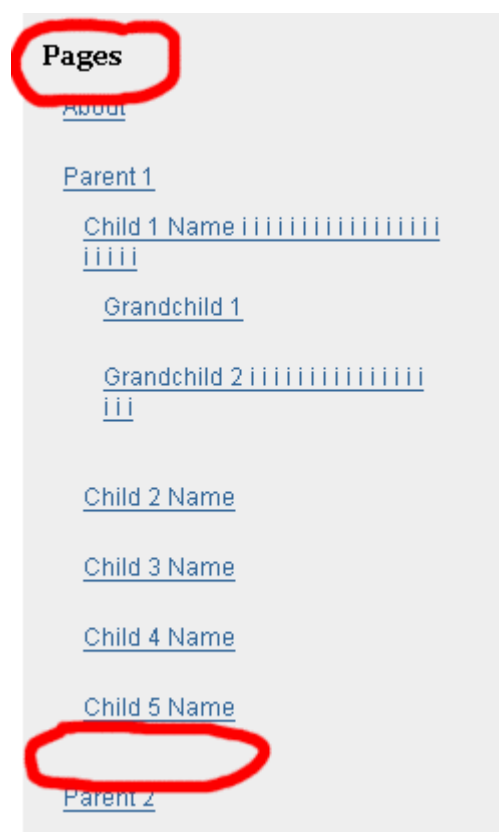
如果不明白，那么就去给 **.sidebar ul** 增加顶部和底部填充。

### 第 3 步

在 **.sidebar ul li** 下输入：

```
.sidebar ul li h2{  
font-family: Georgia, Sans-serif;  
font-size: 14px;  
}
```

还记得样式化在 **.post** 下的子标题是不会起作用的因为你是仅仅在样式化在 **.post** 下的子标题？现在你在样式化侧边栏下的不受影响的子标题，结果如下：



这就是我的页面链接的样子。你的可能只有一个链接， **About**。我给我的离线 WordPress 增加了多重页面链接为了测试最低级别的链接看起的样子，注意到不必要额外的填充在底部，我已圈出。这是一个非常好的关于样式继承的例子。这里不是 10 像素而是 20。

因为你给 `.sidebar ul li` 增加了填充，为了解决这个问题，直行第 4 步。

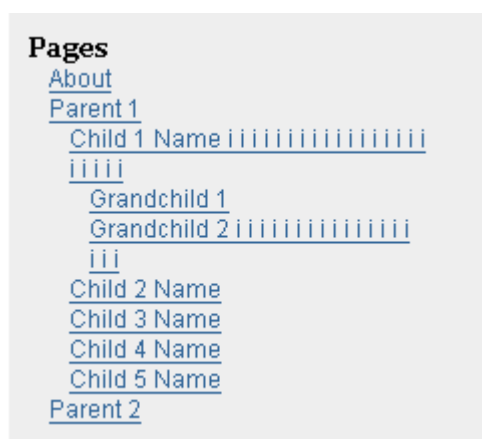
## 第 4 步：

在 `.sidebar ul li h2` 下输入：

```
.sidebar ul ul li{  
padding: 0;  
}
```

在 `.sidebar ul ul li` 中连续的 **UL** 指明了你是在定义第级别的 **LIs**。再说一次，当所有的值都为 0 的时候，你不需要 `px` 这个后缀。

结果如下：



行距太近了，所以我们把行高改成 24px。

增加 `line-height: 24px;` 到 `.sidebar ul ul li`。

```
.sidebar ul ul li{  
padding: 0;  
line-height: 24px;  
}
```

另外，如果你在 IE 下，搜索框下有多出了额外的空白，在下面增加 `form`：

```
body, h1, h2, h3, h4, h5, h6, address, blockquote, dd, dl, hr, p{  
margin: 0;
```

```
padding: 0;
}
```

改成:

```
body, h1, h2, h3, h4, h5, h6, address, blockquote, dd, dl, hr, p, form{
margin: 0;
padding: 0;
}
```

## 第 5 步（可选的）

执行这一步，如果你想让你的日历的数据能够扩展并覆盖整个侧边栏的宽度。当前你的日历应该是这样的：



March 2007						
M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
<a href="#">« Oct</a>						

为了样式化日历，找出在里面的标签和这个便签的名字或者 id。查看 > 页面源代码或者源代码。你的侧边栏是在底部，所以到源代码的底部查找 Calendar。

```
<li id="calendar"><h2>Calendar</h2>
  <table id="wp-calendar">
<caption>March 2007</caption>
<thead>
<tr>
```

现在你知道日历是在一个 **TABLE** 标签中并以 **wp-calendar** 作为 **id**。那么如何在 **style.css** 文件中锁定 **wp-calendar table** 呢？

答案是 **table#wp-calendar{}**。为什么？早前，你学了使用 **#** 号当样式化使用 **id** 而不是 **class** 命名的 **DIV**。在这里，是 **table** 而不是 **DIV**，跟着是 **id** 的值，**wp-calendar**。

如果仅仅 **#wp-calendar{}** 也是可以的因为它是唯一的而且 **WordPress** 不会使用 **#wp-calendar** 给别的标签。但是你应该试着特定化当能够的时候。如果要更加特定化 使用 **.sidebar ul li table#wp-calendar{}**，想更加特定化？好的，

使用 `.sidebar ul li#calendar table#wp-calendar{}`. 因为列表条目 (LI) 包含日历标题和一个 `id` 被命名为 `calendar` 的日历表格。

现在你知道可以使用什么, 如何怎么扩展 `table`, 给表格加上 `width: 100%;`。

在 `.sidebar ul ul li{}` 下输入:

```
table#wp-calendar{
width: 100%;
}
```

保存和刷新, 结果如下:



March 2007						
M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
<a href="#">« Oct</a>						

**width: 100%;** 因为你想日历表格适应到侧边栏的宽度, 无论你把侧边栏改成多少像素。

这看起来并不好, 但是你正在学习如何改进。日历需要更多的样式去看起来更好。

**技巧:** 再次查看源代码, 找出在 **TABLE** 下的哪个标签你可以样式化。

## WP Theme 教程 #14 : 底部和拆分 Index

这篇, 我们完成样式化和开始把 `index.php` 文件分成多个小文件。在这篇中, 你可能先需要 `style.css` 文件, 然后你将把 `index.php` 分成一些新的文件。

打开 Xampp, 主题文件夹, Firefox, IE, `index.php` 和 `style.css`。

### 第 1 步

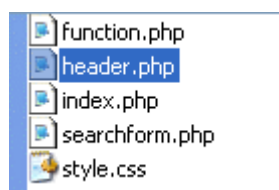
给 `footer` DIV 增加 **10px** 顶部填充。你还记得如何增加填充? 我这次不给你代码了。

### 第 2 步

给 footer 里的所有的 **P** 标签 **18px** 行距。那是 **#footer p{}**。（你已经在 style.css 中完成它。）

### 第 3 步

- 创建一个新文件，把它命名为 **header.php**。确认是 PHP 文件而不是文本文档。
- 在 **index.php** 文件中，把 **header** DIV 上面所有东西都拷贝到 **header.php** 文件中。



```
<?php wp_get_archives('type
<?php //comments_popup_sc
<?php wp_head(); ?>
</head>
<body><div id="wrapper">

<div id="header">

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

</div>

<div id="container">
```

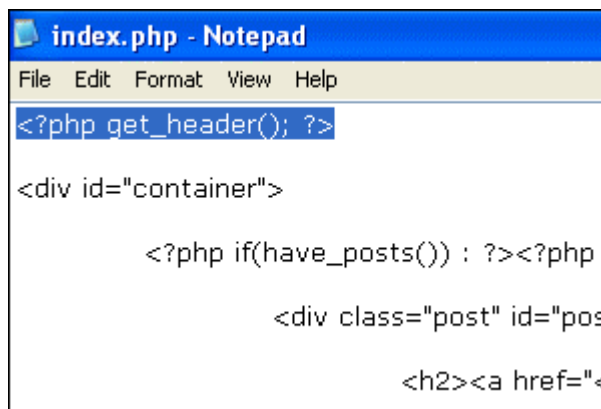
这是我的 [header.php](#) 文件。不要从我的这里拷贝，从你自己的 **index.php** 文件拷贝。

### 第 4 步

为了使所有的东西从 **index.php** 中拷出的东西依然在 **index.php** 文件中，输入以下代码：

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





```
index.php - Notepad
File Edit Format View Help
<?php get_header(); ?>

<div id="container">

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

        <div class="post" id="pos

            <h2><a href="<
```

这是个 WordPress 主题系统特别用来包含 **header.php** 文件的函数，而不用使用：**<?php include (TEMPLATEPATH . '/header.php'); ?>**。

保存并刷新浏览器，你应该看到没有变化。如果你的改变破坏了主题，那么肯定有错误。

## 第 4 步

- 和第 4 步一样，更多相同的事情。这次，创建 **sidebar.php** 文件。
- 把 **index.php** 文件中的 **Sidebar DIV** 开始到结尾都拷贝到 **sidebar.php** 文件中。
- 那么，在 **index.php** 文件，取代为：**<?php get\_sidebar(); ?>**。
- 保存并刷新浏览器，再一次，你应该看到没有变化。
- 这是我的 [sidebar.php](#) 文件。

```
        <?php endif; ?>
    </div>
    <?php get_sidebar(); ?>
    <div id="footer">
    <p>
    Copyright &#169; 2007 <a href="
    </p>
    </div>
</div></body>
</html>
```

## 第 5 步

- 为 **footer.php** 重复上面的步骤。
- 这是我的 [footer.php](#) 文件。

```
<?php endif; ?>

</div>

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

## 教程回顾

- 你创建了三个新文件：**header.php**，**sidebar.php** 和 **footer.php**。
- 你学到了三个新的函数：**get\_header()**，**get\_sidebar()** 和 **get\_footer()**。
- 下面是这节课结束之后的文件：[index](#)，[style](#)，[header](#)，[sidebar](#)，[footer](#)。不要拷贝并使用它们取代你自己打。如果有错误或者你的主题没有显示正确，在和我的代码对照检查下。

## WP Theme 教程 #15 : 子模板文件

继续我们教程，你将和像昨天的 **header.php**，**sidebar.php** 和 **footer.php** 一样创建更多的子模板文件。

现在 [index.php 文件已被拆分](#)，这一切都变得更简单。

### 第 1 步

在你做这步之前，查看你的侧边栏，点击其中的一个存档链接，结果的页面是不是和你的首页没有什么不同？

- 创建一个新文件：**archive.php**
- 把 **index.php** 中所有东西复制到 **archive.php**
- 保存 **archive.php**
- 在 **archive.php** 文件，把 **the\_content** 改成 **the\_excerpt**。
- 再次保存 **archive.php** 文件

通过创建一个 **archive.php** 文件并把改成和 **index.php** 不一样，就是在定制化存档页面的外观。

现在如果你刷新你的存档页面，它将只显示摘要而不是全文的日志。

为什么你想这么做呢？ – 防止 Google 以为重复内容惩罚你的博客，如果一个存档页面和首页显示相同的内容，那就是重复的内容。

如果是私人的博客呢？那么就没有必要去区分首页和存档页面。但这并不是说摘要对私人博客没有用。

同样 — 默认你的类别页面将会从 **archive.php** 上寻找说明如何显示内容。如果你没有 **archive.php** 文件，类别页面将会寻找 **index.php**。

如果你想类别页面和首页和存档页面看起来不一样，创佳一个 **category.php** 文件并定制化它。

## 第 2 步

- 创建一个新文件：**search.php**
- 把 **archive.php** 中所有东西复制到 **search.php**
- 保存就完成了。

现在所有的，所有的搜索结果将会返回摘要。如果没有 **search.php** 这个模板文件，搜索选项将会转向 **index.php** 去如何显示搜索结果。

（可选）你可以返回到[课程 1](#)去回顾者层次结构

## 第 3 步

- 创建两个新文件：**page.php** 和 **single.php**
- 把 **index.php** 中所有内容拷贝到 **page.php** 和 **single.php**。（从现在开始，页面和单篇日志应该是一样的。）
- 保存页面和单篇日志文件，关闭它们。

## 第 4 步

还记得静态页面和页面之间的不同吗？**page.php** 模板文件是用来定制化这些特殊特殊静态页面。

第一，在 **page.php** 中，在 `<?php the_content(); ?>` 输入以下代码：

```
<?php link_pages('<p><strong>Pages:</strong> ', '</p>', 'number'); ?>
```

和

```
<?php edit_post_link('Edit', '<p>', '</p>'); ?>
```

第二，从 **page.php** 中移除 **postmetadata** 代码。结果如下：

```
<div class="entry">

    <?php the_content(); ?>
    <?php link_pages('<p><strong>Pages: </strong> ',
    <?php edit_post_link('Edit', '<p>', '</p>'); ?>

</div>
```

第三，在 **page.php** 中移除 **posts\_nav\_link()** 或者导航模块。

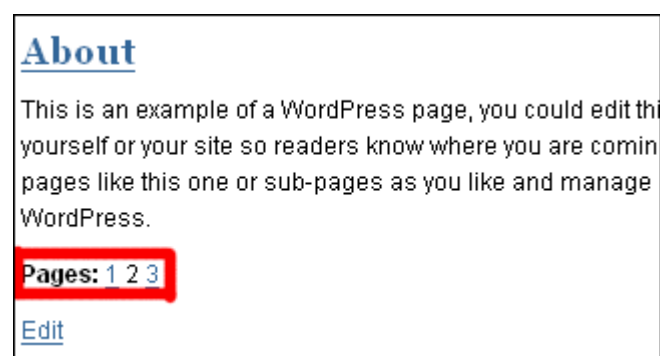
```
        <?php endwhile; ?>

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

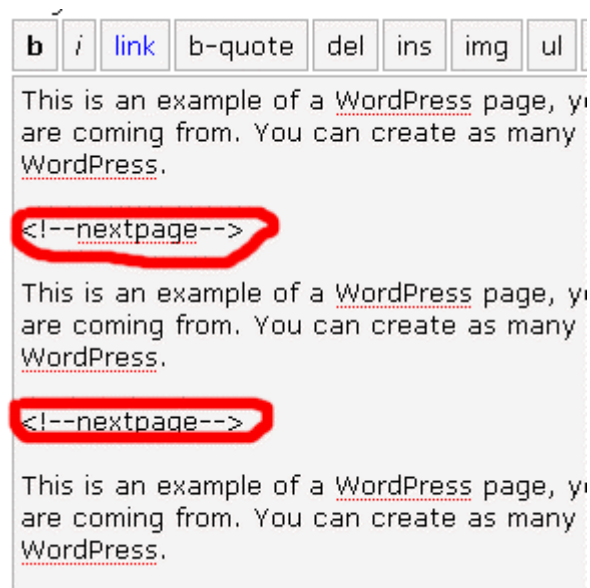
        <?php else : ?>
```

刚才发生了什么？

第一行代码是用于显示子页面的链接。



举个例子，编辑你的 **About** 页面。根据我的屏幕截图增加代码：



当你想把一个非常长的页面分成几个页面的时候，这是非常有用的。

第二行代码是用于显示你的管理员的编辑链接。

你的页面通常没有分类，并且你通常不想给他们显示时间戳，所以你需要去移除 `postmetadata`。你同样要移除 `posts_nav_link()` 代码因为静态页面不会显示后一页和前一页的链接。

保存 `page.php` 文件并关闭它。

## 第 5 步

点击一个日志的标题去阅读日志其余部分带你到了单篇日志查看模式。

`single.php` 模板就是用于处理单篇日志查看使得外观。

在 `single.php` 中，在 `<?php the_content() ?>` 下输入：

```
<?php link_pages('<p><strong>Pages:</strong> ', '</p>', 'number'); ?>
```

是的，这是相同的用于编码子页面链接的代码。你同样可以把日志分成多篇子日志。

第二，在 `postmetadata` 区域，移除 `<?php comments_popup_link(); ?>` 函数和前面的 `<br />` 标签。不要移除整个 `postmetadata`。

移除了留言链接函数是因为在单篇日志查看模式下留言链接函数是不起作用的所以它在 `single.php` 文件中是没有用的。只有管理员可见的编辑链接，在 **BR**

标签的左边。你不想跳过一行才能看到这个本来你可以在右边看到链接？这就是移除 **BR** 标签的原因。

第三，用以下代码取代 `<?php posts_nav_link(); ?>`:

```
<?php previous_post_link('&laquo; %link') ?> <?php next_post_link('%link &raquo;') ?>
```

在前面，存档，分类和搜索页面，你可以使用 `posts_nav_link()` 函数去调用后一页和前一页的链接。对于查看单一日志的页面，它是没有后一页和前一页链接的，你可以使用 `previous_post_link()` 和 `next_post_link()` 函数去调用前一篇日志和后一篇日志的链接。

保存 **single.php** 文件，到某篇日志下查看在导航区域的不同。

## 课程回顾

- 你创建了四个新的文件或者或者子模板：[archive.php](#)，[search.php](#)，[page.php](#) 和 [single.php](#)。
- **archive.php** 和 **search.php** 模板文件是相同的。
- **Pages** （和日志不同）是没有分类的，他们同样没有后一页和前一页的链接。
- **Single.php** 不会显示留言链接（被 `comments_popup_link()` 函数调用）并且他不用 `posts_nav_link()` 去调用导航链接。

## WP Theme 教程 #16 : 留言模板

你几乎完成了。坚持一下最后这一篇课程。这篇你将涉及到所有博客的最重要的东西：**评论模板**。在我看来，一个博客将不成为我博如果它不允许读者留言。可能对一些人来说，评论或者留言并不是那么重要，但是我觉得这是非常奇怪的去想象不可以留言博客。

你应该知道：

- 没有快速的方式在 `comments.php` 建立评论模板
- 大部分的 WordPress 设计者使用来自 WordPress 默认主题（**Kubrick**）的默认评论模板根据。
- 一些设计者会修改默认的评论模板去适合他们自己的需求。
- 你将使用我的对默认评论模板的修改版本。

## 第 1 步

- 创建一个新文件：**comments.php**。
- 把我的 [comments.txt](#) 文件中的内容拷贝到你的 **comments.php**。
- 保存 **comments.php** 文件。

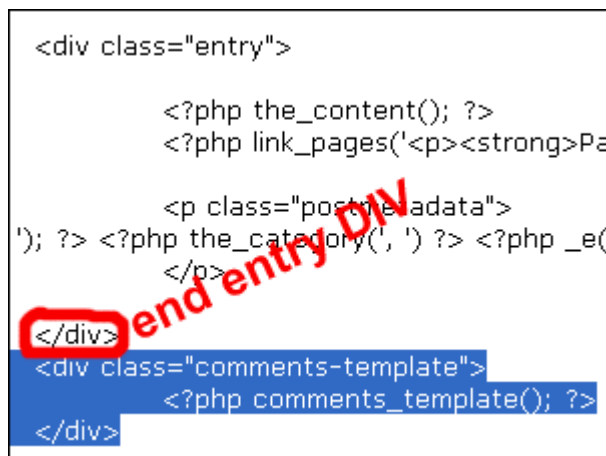
## 第 2 步

- 把我的 [comments-template-css](#) 文件中的内容拷贝到你的 **style.css** 文件中。
- 复制到 **style.css** 的底部或者刚好 **#footer** 的上面。

## 第 3 步

在 **single.php** 文件中，**entry DIV** 的下面，输入以下代码：

```
<div class="comments-template">
<?php comments_template(); ?>
</div>
```



**comments\_template()** 这个函数是用来从 **comments.php** 文件调用评论模板。 **comments.php** 文件然后就会根据它的模板（或者代码）去显示评论列表。列表中的每个条目是一条评论。

你同样可以把 **comments\_template()** 函数用到 **page.php** 文件以让人们可以在静态页面也可以留言。是否这么做取决于你，如果你喜欢，重复上面的三步修改 **page.php** 文件。

## 第 4 步

第四步是验证你的代码，然而可以不进行第四步的，因为你在使用的是我已经整理过的默认主题评论模板的修改版。我已经替你验证过代码了。

验证：

- 查看 > 页面源代码
- 拷贝所有源代码
- 然后到 [validator](#)。
- 把你的代码粘贴到 **Direct Input** 框中。
- 点击 **Check**。

以后的参考（当你创建你自己的主题和评论模板），下面是需要验证的页面：

- 主页 — Home page
- 存档页面 — Archive pages
- 类别页面 — Category pages （如果你自定义了类别页面）
- 搜索结果页面 — Search result pages
- 静态页面 — Pages （如：About）
- 单一日志页面 — Single post view page
- 单一日志没有留言 — Single post with no comments
- 单一日志有留言 — Single post with comments
- 单一日志含有必须登录信息 — Single post with must login message
- 单一日志没有必须登录信息 — Single post with no login required message
- 密码保护的单一日志并有留言 — Password protected single post with comments

## 评论模板的进一步解释

- 评论模板从根本上说是一个**有序列表（OL）**，不是无序的，尽管它们基本上同样方式工作。无序列表是以圆点列表组织的。有序列表则是以数字列表组织的（每个条目都有一个数字，从 1 开始）。
- 在 **single.php** 文件中，你用 **comments-template** DIV 围住 **comments\_template()**。现在你的评论模板在一个 DIV 标签中的一个有序列表中。

当你你的日志是密码保护的，你的评论同样是密码保护的：

```
<h2><?php _e('Password Protected'); ?></h2>
<p><?php _e('Enter the password to view comments. '); ?></p>
```

这个修改版的留言模板有一个 H2 子标题显示 **Password Protected**。默认的留言模板是没有的。

下面展示了哪些东西组成了你的留言列表：



```
<ol class="commentlist">
<?php foreach ($comments as $comment) : ?>
|
    <li class="<?php echo $oddc<?php echo $comment->comment_text() ?>">
    <div class="commentmetadata">
    <strong><?php comment_author_link() ?></strong>
    <a href="#comment"><?php comment_time() ?></a> <?php _e('Said&#58;')
    <?php if ($comment->comment_text() ?>
    <em><?php _e('Your comment') ?>
    <?php endif; ?>
    </div>
    <?php comment_text() ?>
    </li>
    <?php /* Changes every other comment to a div class="comment">
    if ('alt' == $oddc) $oddc = 'comment';
    else $oddc = 'alt';
    ?>
    <?php endforeach; /* end for each comment */
    </ol>
```

还有这里是那些实际的意思：

```
<ol>
  <li>
    each comment sits here
  </li>
</ol>
```

**comment\_text()** 函数就是用来调用每条留言的。

我不会解释留言模板的 CSS 代码的意思。不像 **comments.php** 文件中的代码，你可以随便测试你的 CSS 代码而不会弄坏留言模板。自己去测试回比我的解释对你更有好处。

今天没有课程回顾，你已经完成了。

首批关于主题的系列教程已经完成了，将来的课程将会涉及到更加高级的设计和样式化技术。

## erdaoo 的 WP Theme 教程学习笔记

本文由 [erdaoo](#) 学习本站的 WP Theme 教程之后的[学习笔记整理](#)，经本站整理，erdaoo 本人同意之后在本站发表，以便给更多学习 WP Theme 教程的人帮助。

学习本教程，需要：

1. 对 PHP, WP, CSS, PS 有一定的基础，懂得初步的应用。
2. 你是一个想要表现自己的家伙，并且不想再使用别人制作的主题。
3. 你要有耐心，细心，细心，耐心。

### WP 主题简介

在详细分解代码之前，我们还是要先了解一下 WP 主题的大致情况。一个 WP 的主题是由几个 templates 文件组成的，每一个主题必有的二个文件是：index.php 和 style.css(样式表)，除此之外还有一些其它的文件(不是必须)，它们和 index.php 文件间存在优先级关系，如果它们存在，WP 模板系统就会调用它们显示相应的页面，否则模板系统会调用 index.php 来显示。

它们有可能是以下文件：

- single.php - 单日志文件，用于显示单日志
- page.php — 页面模板文件，用于显示静态页面
- archive.php — 存档文件，用于显示存档页面
- category.php — 类别文件，用于显示类别页面
- search.php — 搜索文件，用于显示搜索结果
- 404.php — 错误文件，用于显示 404 页面
- comments.php — 评论文件，用于显示评论和评论框

### index.php

首先制作 index.php，我们知道在一个网页中，代码主要分为二部分，一个是页头信息，一个是页面内容。

```
<html>
<head>
.....页头信息
</head>
<body>
.....页面内容
</body>
</html>
```

每个主题的页头信息都是几乎一样，具体可以查看默认模板的 header.php 文件（为保证所有页面的页头信息的一致性，所有页头信息都放在 header.php 文件。）

接下来我们谈下一话题，关于母猪的产后护理……（我学的太杂了，都弄混了）

我们来谈一下 body 中的内容。

它包含四个部分，每一部分都可以叫做一个集成模块，其实一个主题就是由不同的模块构成，模块又是由不同的模块构成。

- header WP 的顶部，显示博客的名字与描述，放置导航栏，搜索栏等等。
- content WP 的正文部分，显示贴子的内容，作者，时间，分类，评论，编辑等等。
- sidebar WP 的侧边栏部分。
- footer WP 的尾部，这部分只有很少的内容，通常是版权信息。

对于每一个集成模块中的内容，理论上是可以随意放置的，比如我们可以把 header 模块中的搜索栏放在 sidebar 模块中去。

那如何区分这四个集成模块呢？看以下代码。

```
<div id= "header" >
    这是我的博客
</div>
<div id= "content" >
    这是我的日志</div>
<div id= "sidebar" >
    搜索栏，分类，存档，友情链接
</div>
<div id= "footer" >
    版权信息，我是二道
</div>
```

通过 div 标签，我们可以把这些模块分隔开来。

## header

现在开始我们第一部分的代码块，不过在写代码之前我还得要啰嗦一句，写代码要有层次感，要记得缩进，不要用空格缩进而用 TAB 键。

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

```
</div>
```

id 是 div 的一个属性，给 id 赋予不同的值，这样就可以区分每一个 div 代码段。

bloginfo() 是 WP 中定义好的函数，参数 url 返回网址，参数 name 返回网站的名字，参数 description 返回网站描述。

在上面的代码中，就是为博客的标题并加上一个超链接，并且显示描述。

如果我们把上面的三行代码加上页头部分另存为一个新的文件 — header.php。这样我们就可以通过以下 WP 函数导入它们。

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

这样的好处是，你只要修改一下 header.php 文件，所有调用这个文件的页面都会跟随改变，而不用一个一个地去修改了。

## content

现在开始我们第二部分的代码块：

```
<div id= “content” >
    <?php if(have_posts()) : ?>
        <?php while(have_posts()) : the_post(); ?>

        <?php endwhile; ?>
    <?php endif; ?>
</div>
```

这里使用 if(have\_posts()) 来检测是否有日志存在，如果有的话，就用 while 循环显示。the\_post() 就是调用日志的函数。

而每一篇日志又是有标题，有发布时间，有分属类别，有读者的评论，这些又全部需要用 div 标签来分隔开。看下面的代码：

```
<div id= “content” >
    <?php if(have_posts()) : ?><!-- 开始检测 -->
    <?php while(have_posts()) : the_post(); ?><!-- 以下面的格式显示每篇日志 -->
        <div class= “post” >
            <h2><a href= “<?php the_permalink();?>” ><?php
the_title();?></a></h2><!-- 含有链接地址的日志标题 -->
            <div class= “entry” >
                <?php the_content();?><!-- 日志内容 -->
```

```

<p class= “postmetadata” ><!-- 日志元数据 -->
<?php _e(‘ Filed under’ . $category);?>
<?php the_category(‘ , ‘);?><!-- 调用日志的分类 -->
<?php _e(‘ by ‘);?><!-- 使用_e() 创建可翻译的主题 -->
<?php the_author(‘ ‘);?><!-- 调用日志作者 -->
<br />
<?php comments_popup_link(‘ No Comments’ . $comments_count, ‘ % Comment’ . $comments_count);?><!-- 调用一个弹出的留言窗口，如果这个功能没有激活，则是显示留言列表 -->
<?php edit_post_link(‘ Edit ‘ . $title);?><!-- 只有在登陆后才可见到，对日志进行编辑的链接 -->
</p>
</div><!-- 日志内容结束 -->
</div><!-- 一篇日志彻底结束 -->
<?php endwhile; ?>
<?php endif; ?>
</div>

```

## class

现在我们要说说 class 了，它是与 id 都是标签的属性，但是不同之处在于，id 的参数值是唯一的，它在一个页面只能使用一次，而 class 的参数值是可以多次使用，

比如 id=“header” 只能出现一次，因为我们只有一个地方可以出现博客的名字。而 class=“entry” 会经常出现，那是因为我们的博客里不只是有一篇日志。

为什么我们要用到 id 与 class，难道只用一个不行吗，反正功能都是相同的。不要忘了我们前面说过的一个重要文件，style.css 样式表文件。我们为某一段代码添加了属性，如同起个名字而已，这样在样式表中我们就可以为这些名字来定制它们的样式了。

这样说你还不明白？那就打个最简单的比方吧，你可以有很多的兄弟，但是你们只能有一个爹，你不能用你爹的名字叫你的兄弟，但是你爹可以用你兄弟的名字叫你。样式表文件就和你奶奶一样，你爹再牛逼也得听你奶奶的话，叫他怎么样他就得要怎么样。（老大你这个比喻寒啊，瀑布寒！！）

## Not Found

前面的代码中有说到，如果检测到有日志的话，就用循环调出来，可是如果没有日志的话那要怎么办呢？

```

<?php else:?>

```

```
<div class= “post” id= “post-<?php the_ID(); ?>” >
<?php _e( ‘ Not Found ’ );?>
</div>
```

把这一段代码加在 `<?php endwhile; ?>` 之后就可以了。

## 页面导航

当你的博客内容越来越多的时候，在 WP 的后台又设定了首页只显示 10 个日志，那么从第 11 个开始都无法在首页显示出来。

这样在博客的最后一篇日志下面就会出现后一页或前一页的链接。如果你还不到 10 个日志，这个链接就不会出现。

把下面的代码加入到 `<?php endif; ?>` 前面

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

分析一下 `posts_nav_link()` 这个 WP 函数，它可以有三个参数：`<?posts_nav_link( ‘ in between ‘ , ‘ before ‘ , ‘ after ‘ )`，第 1 个参数是显示在后一页和前一页链接的中间。第 2 个参数显示在后一页和前一页链接的前面。第 3 个参数显示在后一页和前一页链接的后面。用什么来显示，你自己决定，常用的就是一些符号或是箭头而已嘛。

现在再看一下我们已经有了哪些个代码：

```
<?php get_header(); ?>
<div id= “content” >
    <?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>
        <div class= “post” >
            <h2><a href= “<?php the_permalink(); ?>” >?php
the_title(); ?></a></h2>
            <div class= “entry” >
                <?php the_content(); ?>
                <p class= “postmetadata” >
                    <?php _e( ‘ Filed under’ . ‘ ’ ); ?>
                    <?php the_category( ‘ , ‘ ); ?>
                    <?php _e( ‘ by ‘ ); ?>
                    <?php the_autnor( ‘ ‘ ); ?>
                <br />
```

```

        <?php comments_popup_link(' No Comments&#187; ', ' 1
Comments&#187; ', ' % Comment&#187; ');?>
        <?php edit_post_link(' Edit ', '&#124; ', ' ');?>
    </p>
</div>
</div>
<?php endwhile; ?>
<div class= "navigation" >
    <?php posts_nav_link(); ?>
</div>
<?php else:??>
    <div class= "post" id= "post-<?php the_ID(); ?>" >
        <?php _e(' Not Found ');?>
    </div>
<?php endif; ?>
</div>
</body>
</html>

```

\*\*\*\*\*  
 写教程不是一个简单的事，它不光让我心烦，还让我难以找到适当的词来表达，所以要体会一下当老师的难处。  
 \*\*\*\*\*

## 侧边栏

第三部分，关于侧边栏。侧边栏有一个特点，就是又臭又长，当然了这不是什么缠脚布。先不要乱扯。因为地形有限，所以侧边栏里的内容，多以列表的形式排开。下面欢迎一对父子出场，他们的感情是相当的好，从来都是父子不分家，有父必有子，有子必有父，父中有子，子中有父。他们就是<UL>和<LI>!!!!!!

```

<div class= "sidebar" ><! - 注意这里使用的不是 id - >
    <ul>
        <li>
            <h2><?php _e(' 日志分类' '); ?></h2>
        </li>
    </ul>
</div>

```

UL 表示无序列表，OL 表示列表元素。在侧边栏里，你要有几个不同的栏目，栏目的存在，就是为侧边栏进行了分类整理。每一个栏目又要有不同的分类列表，继续为上面的代码添加内容。

```

<div class= "sidebar" >

```

```

<ul>
  <li><h2><?php _e(' 日志分类 '); ?></h2>
    <ul>
      <?php wp_list_cats
(' sort_column=name&optioncount=1&hierarchical=0 '); ?>
    </ul>
  </li>
</ul>
</div>

```

wp\_list\_cats() 函数为调用日志分类列表，它的参数也有三个。每个参数之间用&来分隔。

sort\_column=name — 把分类按字符顺序排列

optioncount=1 — 显示在每个分类下面的日志数

hierarchical=0 — 不把子分类放到子列表条目中

说到分类，特别说一下静态页面这个栏目。我们在 WP 后台撰写的时候，有二个选择，一个是撰写日志，一个是撰写页面。对于日志，还可以选择保存在哪一个具体的分类下面。对于页面就没得选择，只收录于页面栏目之下。再回到前台，你可以看到每个分类都有显示日志的数目，而不显示标题。在页面栏目里，只排列了每一个页面的标题，而不显示数目。

```

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

```

参数 depth=3 为可选参数，表示可以设定显示三级列表。

注意一点，本教程的代码是制作模版的代码（PHP 代码），在 WP 中使用一个主题也就是等于在套用一个模版。在网站中查看源代码是看不到模版的代码的（已经被解释成 HTML 代码）。

```

<li><h2><?php _e(' 日志分类 '); ?></h2>
  <ul>
    <?php wp_list_cats
(' sort_column=name&optioncount=1&hierarchical=0 '); ?>
  </ul>
</li>

```

上面这一段模版代码，在网页中查看源代码，实际上显示的是这样的：

```

<li><h2>文章存档</h2>
  <ul>
    <li><a href= “#” >与爱情有关的分类贴子</a></li>
    <li><a href= “#” >与生活有关的分类贴子</a></li>
    .....
  </ul>
</li>

```



增加一个存档栏目：

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

wp\_get\_archives() 函数是用来获取文章存档的，参数 'type=monthly' 定义为以每个月的时间来进行分类存档

增加一个友情链接栏目：

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

不用担心没有实际内容，它会自动调用在 WP 后台中添加的友情链接。

增加一个搜索栏目：

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

这里使用 include() 函数调用一个文件，参数 TEMPLATEPATH 为主题文件夹路径，为了调用成功，你还需要有一个文件：[searchform.php](#)。

增加一个日历栏目：

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

这里就不用多废话了。

增加一个管理栏目：

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

```
</li>
```

wp\_logout() 来确定你是否登陆, 如果登陆就显示登出链接, 如果没有登陆, 就显示登陆的链接。

wp\_register() 来确定你的身份, 如果没有登陆, 就显示注册的链接, 如果有的话, 就显示管理的链接。

而 wp\_meta() 却是什么也没有做。也不用去理它, 还没有人来说明它是起什么作用的。实际上它是 WordPress 的 hook。

窗体化侧边栏

```
<?php      /* Widgetized sidebar, if you have the plugin installed. */
            if ( !function_exists(' dynamic_sidebar ')
|| !dynamic_sidebar() ) : ?>
```

在侧边栏开始的地方第一个<ul>的后面, 加上以上代码。也要在侧边栏结束的地方</ul>前面加上一句

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

从 WP2.0 开始, 已经在后台集成了一个侧边栏的插件——Widget, 它的功能就是可以很方便的在 WP 后台调整侧边栏中的内容, 直接使用鼠标就可以移动每一个栏目的位置, 而不需要去修改相应的代码。让每一个栏目都以窗体化存在。

[function\\_exists\(' dynamic\\_sidebar '\)](#) || !dynamic\_sidebar()) 这两个参数来自于一个新的文件 — [functions.php](#) 我们需要创建这个文件才可以完成侧边栏的窗体化.。

通过观察不同的 WP 主题, 会发现在侧边栏中的内容远不止以上所列举的, 要在学习中学举一反三, 才会制作出更加出众的主题。

至此, 侧边栏中的内容结束, 我们也可以把第三部分的代码另存为一个新的文件 — sidebar.php, 在 index.php 中填加一句代码就可以使用侧边栏

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

顺便再增加一行代码:

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

这是调用尾部文件 footer.php 的代码。我想你应该知道如何处理一个简单的 PHP 文件了, 要么你就再重头学一次本教程。

再一次查看一下 index.php 有了哪些代码

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

```

<div id= "content" >
    <?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>
        <div class= "post" >
            <h2><a href= "<?php the_permalink();?>" ><?php
the_title();?></a></h2>
            <div class= "entry" >
                <?php the_content();?>
                <p class= "postmetadata" >
                    <?php _e(' Filed under' . ' ');?>
                    <?php the_category(' , ');?>
                    <?php _e(' by ');?>
                    <?php the_author(' ');?>
                    <br />
                    <?php comments_popup_link(' No Comments' . ' , ' 1
Comments' . ' , ' % Comment' . ' ');?>
                    <?php edit_post_link(' Edit ' , ' &#124; ' , ' ');?>
                </p>
            </div>
        </div>
    <?php endwhile; ?>
    <div class= "navigation" >
        <?php posts_nav_link(); ?>
    </div>
    <?php else: ?>
        <div class= "post" id= "post-<?php the_ID(); ?>" >
            <?php _e(' Not Found ');?>
        </div>
    <?php endif; ?>
</div>
<?php get_sidebar(); ?>
<?php get_footer();?>

```

index.php 文件的代码已经全在这里了，但是只有第二部分内容需要详细的代码，而其它的部分我们都可以调用外部文件，至此一个 WP 的主题构造已经搭建好，再一次提醒各位，检查代码，确认书写正确。只有不厌其烦地写代码才会对代码有更深刻的印像。

## 其他文件

下面开始创建其它文件

将 index.php 的全部代码另存为 archive.php，并且把 the\_content 改成 the\_excerpt，创建存档文件，它会显示在分类栏目下的每篇日志的摘要。

将 archive.php 另存为 search.php，创建搜索文件，这样就可以在搜索中得到每篇文章的摘要。

将 index.php 的全部代码另存为 page.php，创建页面模板文件，在 `<?php the_content(); ?>` 下面输入以下代码：

```
<?php link_pages(' <p><strong>Pages:</strong> ', ' </p> ',  
    'number '); ?>
```

说明：如果一个页面，篇幅超长的话，我们可以把它截断分成几页来显示，

```
<?php edit_post_link(' Edit ', ' <p> ', ' </p> '); ?>
```

说明：显示一个可以编辑的链接

删除掉 `<p class= "postmetadata" >` 至 `</p>` 这一块儿的代码  
删除掉以下代码：

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

说明：对于静态页面，它没有属于哪个分类，我们也不希望被某人评论，当然它也不能显示与另一个页面间的连接，所以要去掉一部分代码。

将 index.php 的全部代码另存为 single.php，创建单篇文章文件，点击文章的标题，可以查看全文内容。在 `<?php the_content(); ?>` 下输入：

```
<?php link_pages(' <p><strong>Pages:</strong> ', ' </p> ',  
    'number '); ?>
```

这段代码和上一例相同，都是可以用来截断文章。

删除以下代码：

```
<br />  
<?php comments_popup_link(' No Comments&#187; ', ' 1  
Comments&#187; ', ' % Comment&#187; '); ?>
```

把 `<?php posts_nav_link(); ?>` 替换成 `<?php previous_post_link(' &laquo;  
%link ') ?> <?php next_post_link(' %link &raquo; '); ?>`

说明：在单篇文章的下面显示的应是上一篇与下一篇的链接，而不是上一页与下一页的链接。

如何处理留言评论？

想一想，每一个留言评论都是对于一个文章而产生的，所以只要在单篇文章页面里添加一个调用评论的函数就可以。

在 single.php 文件中 `<div class= “entry” >` 代码块结束的 `</div>` 下面，输入以下代码：

```
<div class= “comments-template” >
<?php comments_template(); ?>
</div>
```

comments\_template() 这个 WP 函数是用来从 comments.php 文件调用评论模板。所以我们还要创建一个 [comments.php](#) 文件。