WordPress　主题超详细简明教程

每一套 WP 主题都由结构层、表现层和数据层构成，可以说是典型的、符合 Web 体系标准的“三层结构”。WP 主题的这种模式块化的特点决定了其设计其实是非常简单，但又极其灵活的。可以说，学会了制作 WP 主题，就相当于基本理解了 Web 开发的客户端模型，对进一步学习掌握 Web 技术具有重要意义。这也正是 WP 的魅力所在！

在一套 WP 主题中，最基本的两个文件是 index.php 和 style.css。其中，前者定义结构和内容，后者定义样式。所谓结构，就是由指由 XHTML 标签构成的网页基本架构。在 WP 主题中，结构层主要使用的是添加了适当的 id 或（和）class 属性的 div 和 ul 元素，以便更好地通过表现层来控制页面的布局和外观。所谓表现层，其实就是网页的布局和样式——也就是外观。表现层由 CSS（层叠样式表）规则定义构成。而数据层，顾名思义，也就是网站中实际显示的内容，是通过调用 WP 内置的函数（有时需要传递适当的参数）从数据库中取得的。

说明：为简明起见，本教程不涉及如何定义样式表（CSS）文件的内容。

事实上，当我们打开某个主题的文件夹时，看到的并不止这两个文件，而是更多。但一般来说，在一个完整的 WP 主题文件夹中都应该包含下列文件（也称为模板文件）：

页面	模板文件	用途
首页	index.php	显示网站首页
单页	single.php	显示博文的页面（相当于细节页）
静态页	page.php	显示静态页的页面（包含各级静态页面）
分类页	category.php	显示分类页的页面（相当于栏目页）
存档页	archive.php	显示存档页的页面（相当于按时间归类的栏目页）
搜索页	search.php	显示搜索结果的页面
评论页	comments.php	显示评论的页面
弹出式评论页	comments-popup.php	显示弹出式评论的页面
404 错误页	404.php	显示 404 错误信息的页面
层叠样式表	style.css	控制页面布局外观

除此之外，一套主题模板中还可以包含 author.php、home.php、date.php、searchform.php 以及 functions.php 等页面（其中部分页面稍后介绍）。

虽然上面列出了与 WP 内置功能对应的 9 个 php 文件，但制作一套主题远没有想像得那么复杂。因为事实上，你只需要制作一个 index.php 文件，就可以派生出另外 8 个文件来！从 WP 应用主题的机制来说，这 9 个模板文件是存在优先级差别的，也可以认为是重要性不同。它们的优先级顺序是：

index.php -> single.php -> page.php -> archive.php -> search.php -> 404.php。

这样，当不存在后边的页面时，WP 会自动调用前面的页面，直至调用 index.php。比如，当程序调用页面页 page.php 时，如果 page.php 模板文件不存在，那么程序会尝试调用前面的文件——single.php。而如果 single.php 也不存在，那么就会调用最终的 index.php 来显示页面页。可见 index.php 属于“垫底儿”的缺省页面，它的重要性是最高的。但当存在具体页面时，还是要优先使用具体的页面，可见具体的页面优先级最高。

在明确了 index.php 是一套 WP 主题的核心之后，我们就可以将制作 WP 主题的过程简单地分成两步，即——定义主模板文件 index.php 和派生其他模板文件。

定义主模板文件 index.php

从页面布局的角度上，有必要将主模板文件 index.php 拆分成 header.php、sidebar.php 和 footer.php 三个子页面。WP 专门为在 index.php 中包含这三个子页面提供了对应的

get_header()、get_sidebar() 和 get_footer() 函数。
下面，就来详细介绍一下制作 index.php 页面的过程：
首先，在 myThemes 文件夹中建立一个文本文件并将其重命名为 index.php，然后再建立一个 style.css 文件（内容暂时留空）。
然后，用你喜欢的文本编辑器打开 index.php 并输入下列代码（最好复制，因为这一部分不重要）：

```
<!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">
<meta http-equiv="Content-Type" content="<?php bloginfo('html_type'); ?>;
charset=<?php bloginfo('charset'); ?>" />
<title><?php bloginfo('name'); ?> <?php if ( is_single() ) { ?> » Blog Archive <?php } ?>
<?php wp_title(); ?></title>
<meta name="generator" content="WordPress <?php bloginfo('version'); ?>" />
<!– leave this for stats –>
<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>"
type="text/css" media="all" />
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>
/print.css" type="text/css" media="print" />
<link rel="alternate" type="application/rss+xml" title="<?php bloginfo('name'); ?>
RSS Feed" href="<?php bloginfo('rss2_url'); ?>"/>
<link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
<?php wp_head(); ?>
</head>
<body>
</body>
</html>
```

显然，这是一些构成网页文件的 HTML 代码，当然其中包含 PHP 代码。如前所述，HTML 代码定义结构，而 PHP 代码用于生成内容。
在 WP 后台的“主题”模块中，选择 myTheme 主题。然后，在浏览器中观察你的 WP 外观。此时，应该显示页面一片空白。但通过“查看源文件”，你会发现 index.php 中的 PHP 代码已经生成了一些头部内容。
接着，我们开始动手定义 index.php 中 body 部分的结构和内容。
（一）构建 header
在 body 元素内，新建如下结构化标记元素，注意各元素都带有不同的 id 属性：

```
<div id="page">
<div id="header"></div>
<div id="content"></div>
<div id="sidebar"></div>
<div id="footer"></div>
</div>
```

然后，在该 元素的两个标签之间输入下列代码：

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

bloginfo('name'); ?></h1>
<?php bloginfo('description'); ?>
这里用到了 WP 内置的 bloginfo 函数来生成内容，其中：
bloginfo('url')返回网站主页链接；
bloginfo('name')返回网站标题；
bloginfo('description')返回网站描述。
保存 index.php 文件，然后在浏览器中按 F5 刷新一下页面，看能看到什么？再通过"查看源文件"，核对一下由 WP 的 bloginfo() 函数生成的相关信息。
（二）构建 content
在 中，我们要通过循环显示博文，包括每个博文的标题、作者、发表日期以及其他相关信息。并且，可以分页显示博文（取决于 WP 后台的设置）。
首先，在 与 之间输入下列代码：

```
<?php while (have_posts()) : the_post(); ?> <div class="post" id="post-<?php the_ID() ?>">
<!– 博文标题及链接 –>
<h2><a href="<?php the_permalink() ?>" rel="bookmark" title="<?php the_title(); ?>">
<?php the_title(); ?></a></h2>
<!– 发表日期 –>
<div class="post-date">
<span class="post-month"><?php the_time('M') ?></span>
<span class="post-day"><?php the_time('d') ?></span>
</div>
<!– 作者 –>
<span class="post-author"><?php _e('Author'); ?>：<?php the_author(', ') ?></span>
<!– 类别 –>
<span class="post-cat"><?php _e('Categories'); ?>：<?php the_category(', ') ?></span>
<!– 注释 –>
<span class="post-comments">
<?php comments_popup_link('No Comments »', '1 Comment »', '% Comments »'); ?></span>
<!– 内容 –>
<div class="entry">
<?php the_content('更多内容 »'); ?>
</div>
<!– 其他元（Meta）数据 –>
<div class="post-meta">
<?php edit_post_link('编辑',' | ',''); ?>
</div> </div>
<?php endwhile; ?><div class="navigation">
<span class="previous-entries"><?php next_posts_link('前一篇') ?></span> <span class="next-entries"><?php previous_posts_link('后一篇') ?></span>
</div>
<?php else : ?>
<div class="post">
<h2><?php _e('Not Found'); ?></h2>
</div><?php endif; ?>
```

看似复杂，其实不然。首先：

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

<?php else : ?>

<?php endif; ?>

这三行，在 WP 中表示 if 控制结构。注意，if 语句通过测试 have_posts() 函数来判断是否存在博文。而

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

<?php endwhile; ?>

这两行，是 WP 中的 while 循环。其中，while 语句通过测试 have_posts() 决定是否调用 the_post() 函数。如果测试 have_posts() 返回 true，则调用 the_post() 函数，初始化与博文相关的内置变量。

在 while 循环内部，首先要注意通过 div、h2、span 这三个元素定义的嵌套语义结构，以及相应元素的 class 和 id 属性（其中只为 class 为 post 的 div 元素定义了一个 id 属性——post-<?php the_ID() ?>）。这是将来使用 CSS 控制外观的关键所在。在这个 div 元素中，为显示博文的相关信息，分别调用了以下 WP 函数：

the_ID()：返回博文 ID；

the_permalink()：返回博文固定链接 URL；

the_title()：返回博文标题；

the_time('M')：返回发表日期中的月份；

the_time('d')：返回发表日期中的天；

the_author()：返回博文作者；

the_category()：返回博文的类别；

the_content()：返回博文的内容，其中的参数表示用于"更多内容"的链接文本；

以上函数都是以 the_ 开头的，加上后面的函数名不仅颇有自解释的味道，而且令人联想到 this 关键字。此外_e() 函数是一个包装函数，这个函数主要用于语言的转换，如果调用该函数并传递标准的 WP 术语，如：Author 或 Categories，则返回你相应语言包中的译文，在中文包中分别是"作者"和"类别"。当然，不用也可。但会失去一些适应性。

还有，omments_popup_link() 和 edit_post_link() 两个函数，分别显示注释和编辑链接，这里不多说了。

另外，在 <?php endwhile; ?> 后面显示了分页导航链接，调用的函数分别是：next_posts_link() 和 previous_posts_link()。此时，如果你的博文总数小于 WP 后台设置的最多显示数目，比如：你在后台设置最多显示 5 篇，而你有 10 篇博文，就会分页显示；否则，如果你的博文少于或等于 5 篇则看不到分页导航链接。

最后，不要丢下 <?php else : ?> 语句后面的内容：

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

显然，这是一个错误提示信息。

（三）构建 sidebar

sidebar 的内容当然要在 元素中构建了。sidebar，中文叫侧边栏，其中可以包含很多内容。比如：分类、页面、链接、日历等等导航及相关信息。

在 WP 中，sidebar 中的内容都以无序（ul）或有序（ol）列表的形式输出。因此，需要在 中输入以下标记：


```
<?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar() ) : ?>
<li id="search">
<?php include(TEMPLATEPATH .'/searchform.php'); ?>
</li> <li id="calendar">
<h2><?php _e('Calendar'); ?></h2>
<?php get_calendar(); ?>
</li> <?php wp_list_pages('title_li=<h2>页面</h2>'); ?> <li class="catnav">
<h2><?php _e('Categories'); ?></h2>
<ul>
<?php wp_list_cats('sort_column=name&optioncount=1&hierarchical=0′); ?>
</ul>
</li>
<li class="archivesnav">
<h2><?php _e('Archives'); ?></h2>
<ul>
<?php wp_get_archives('type=monthly'); ?>
</ul>
</li>
<li class="blogrollnav">
<h2><?php _e('Links'); ?></h2>
<ul>
<?php get_links('-1′, '<li>', '</li>', '<br />', FALSE, 'id', FALSE, FALSE, -1, FALSE); ?>
</ul>
</li>
<li class="meta">
<h2><?php _e('Meta'); ?></h2>
<ul><?php wp_register(); ?><li><?php wp_loginout(); ?></li>
<?php wp_meta(); ?></ul>
</li>
<?php endif ?></ul>
```

以上代码从第三行开始，分别通过包含 searchform.php 显示搜索表单；

调用 get_calendar() 函数显示日历；

调用 wp_list_pages() 函数显示页面导航；

调用 wp_list_cats() 函数显示分类导航；

调用 wp_get_archives() 函数显示存档导航；

调用 get_links() 函数显示链接导航。

在构建侧边栏时，要为生成搜索框新建一个 searchform.php 文件，其内容如下：

```
<form method="get" id="searchform" action="<?php bloginfo('home'); ?>/">
<div>
<input type="text" value="<?php echo wp_specialchars($s, 1); ?>" name="s" id="s" size="15″
/><br />
<input type="submit" id="searchsubmit" value="Search" />
</div>
</form>
```

将其保存在 myTheme 文件夹中，通过 include 语句包含进来就可以了。注意，常量 TEMPLATEPATH 中保存的是模板路径。

最后，说明一下以上代码第二行和倒数第二行。显然这是一个 if 语句块。那这个 if 语句块包含 sidebar 是何用意呢？这是部件化侧边栏的需要，就是让 sidebar 适合 Widget 插件（WP 2.0 后内置了 Widget，所以不用再安装了）。如果要使用 Widget 插件，必须对 sidebar 进行部件化。这样，在 WP 后台通过 Widget 插件你就可以使用拖动来方便地定义侧边栏的组件了。部件化侧边栏，除了在 ul 元素内侧放入这个 if 语句之外，还必须在 myTheme 文件夹中建立一个文件 functions.php，其内容如下：

```
<?php
if ( function_exists('register_sidebar') )
register_sidebar(array(
'before_widget' => '<li id="%1$s" class="widget %2$s">',
'after_widget' => '</li>',
'before_title' => '<h2 class="sidebartitle">',
'after_title' => '</h2>',
));
?>
```

（四）构建 footer

footer 中一般都一些版权信息和不太重要的链接。所以可以在 元素中简单地放入下列代码：

```
<p>Copyright © 2007 <?php bloginfo('name'); ?></p>
```

至此，核心 index.php 文件就算是大功告成了！

接下来，是拆分 index.php 和基于 index.php 派生子模板文件。

在 myTheme 文件夹中新建 header.php、sidebar.php 和 footer.php 三个文件。把 index.php 中的 、 和 三个结构化元素及其内容分别转移（剪切）到这三个新文件中。然后，在 原来的位置处输入代码：

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

在 原来的位置处输入代码：

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

在 原来的位置处输入代码：

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

前面说过，这三个 get 函数是 WP 专门为包含结构化的文件定义的。现在你的 index.php 文件应该如下所示：

```
<!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">
<meta http-equiv="Content-Type" content="<?php bloginfo('html_type'); ?>; charset=<?php
bloginfo('charset'); ?>" /><title><?php bloginfo('name'); ?> <?php if ( is_single() ) { ?> » Blog
Archive <?php } ?> <?php wp_title(); ?></title><meta name="generator" content="WordPress
<?php bloginfo('version'); ?>" /> <!– leave this for stats –><link rel="stylesheet" href="<?php
bloginfo('stylesheet_url'); ?>" type="text/css" media="all" />
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/print.css" type="text/css"
media="print" />
```

```
<link rel=”alternate” type=”application/rss+xml” title=”<?php bloginfo(’name’); ?> RSS Feed” href=”<?php bloginfo(’rss2_url’); ?>” />
<link rel=”pingback” href=”<?php bloginfo(’pingback_url’); ?>” /><?php wp_head(); ?>
</head>
<body>
<div id=”page”><?php get_header(); ?> <!– content –>
<div id=”content”>
<?php if (have_posts()) : ?>
<?php while (have_posts()) : the_post(); ?> <div class=”post” id=”post-<?php the_ID() ?>”>
<!– 博文标题及链接 –>
<h2><a href=”<?php the_permalink() ?>” rel=”bookmark” title=”<?php the_title(); ?>”>
<?php the_title(); ?></a></h2>
<!– 发表日期 –>
<div class=”post-date”>
<span class=”post-month”><?php the_time(’M’) ?></span>
<span class=”post-day”><?php the_time(’d’) ?></span>
</div>
<!– 作者 –>
<span class=”post-author”><?php _e(’Author’); ?>：<?php the_author(’, ‘) ?></span>
<!– 类别 –>
<span class=”post-cat”><?php _e(’Categories’); ?>：<?php the_category(’, ‘) ?></span>
<!– 注释 –>
<span class=”post-comments”>
<?php comments_popup_link(’No Comments »’, ‘1 Comment »’, ‘% Comments »’); ?></span>
<!– 内容 –>
<div class=”entry”>
<?php the_content(’更多内容 »’); ?>
</div>
<!– 其他元（Meta）数据 –>
<div class=”post-meta”>
<?php edit_post_link(’编辑’,’ | ‘,”); ?>
</div> </div>
<?php endwhile; ?> <div class=”navigation”>
<span class=”previous-entries”><?php next_posts_link(’ 前 一 篇 ’) ?></span> <span class=”next-entries”><?php previous_posts_link(’后一篇’) ?></span>
</div>
<?php else : ?>
<div class=”post”>
<h2><?php _e(’Not Found’); ?></h2>
</div><?php endif; ?>
</div><!– end content –><?php get_sidebar(); ?> <?php get_footer(); ?></div>
</body>
</html>
```

然后，是派生子模板文件。把这个“模块化”的 index.php 文件另存为 single.php、page.php、

archive.php、 search.php 和 category.php。当然，都保存在 myTheme 文件夹中。这样，WP 在显示页面时就会调用相应的页面文件了。比如，显示博文详细内容时，会调用 single.php；而显示页面内容时，则调用 page.php。
最后，要做的工作就是自定义这些子模板文件。

WP 全自动外链插件

在这里，向大家隆重推荐WP全自动外链插件：**Automatic Backlink Creator插件**。

这款软件本人使用过，效果非常不错，所以今天在这里推荐一下，希望可以节省大家做外链的时间和精力！

Automatic Backlink Creator 主要针对 wordpress 程序建立的网站。热衷于 WP 的站长朋友，尤其是针对做外贸的站长朋友们，主做 Google、Yahoo 搜索引擎 SEO，应该是一个非常好的消息！
这款软件类似于 WP 插件，是一个 WP 站外链的完美解决方案！只需要在网站后台轻松安装，就可以采用对搜索引擎有好的方式，让 WP 网站自动增加高权重外链。

最近本软件的官方网站，Automatic Backlink Creator 的价格只需要 37 美元，可以使用信用卡或者 paypal 付款，在国外卖的非常火爆！ 购买的同时，还赠送 MetaSnatcher 插件， 这款插件可以自动跟踪排在谷歌前几名的竞争对手网站核心关键，并且自动返回到软件，省去大量关键字分析的时间。Spin Master Pro 插件。这款插件相当于是 WP 离线伪原创和发布插件，安装这款插件之后，可以在自己的电脑进行内容伪原创并离线发布，节省大量时间。同时，本软件提供 60 天不满意退款保证。

点击查看

这款软件的开发者是一群 SEO 高手，结合谷歌和雅虎的外链算法，开发出了这款强大优秀的外链软件，在外链的 PR、OBL、FLAG 等方面考虑到了极致。并且通过这个系统可以产生稳定的、持续增加的高质量反链，例如.edu、.gov 等等网站的外链。

下载：最经典的SEO链轮解决方案