WordPress 主題教學

建立 WordPress 佈景其實不難，只要你從現在開始認真學習這個教學，從零一步一步開始，你就會成為一個 WordPress 主題製作高手，至少你會修改現有主題。

下面是一個從零開始製作 WordPress 主題的教學，這個教學最初翻譯自 So you want to create WordPress themes huh? 經過多次修正以適應中文習慣和做了個人理解，這個教學它會一步一步教你如何製作 WordPress 主題。

內容目錄

推薦和贊助商

WordPress 主題教學是Denis發佈在我愛水煮魚上的第一個非常完整的 WordPress 相關教學，Pseric校對並重新編製成正體中文，發佈於免費資源網路社群，同時也修改了部分內容讓正體中文的讀者能夠更明白其中的意思。自從發佈 PDF 電子書之後，截至目前為止已經被下載超過 12354 次，並且這個統計只是 box.net 提供的資料，其他下載站的資料無法統計，估計至少還有1萬次的下載，從2009年12月份來，Denis已經對這個教學進行大的修正，今天（2010-1-17）發佈修正後的第一版，對這個為了能夠使得這一教學能夠持續的修正，目前已開始和贊助商接洽。另外Denis也計畫將發佈 WordPress 外掛製作教學，從零開始使用 WordPress 等系列 PDF 電子書。

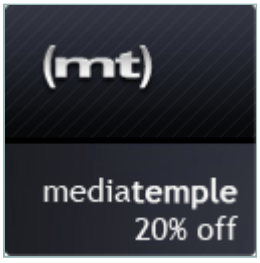
如果您對這一教學或者以後的教學感興趣，想贊助我們的話，請聯繫 Denis（http://wpjam.com/contact/），台灣方面可以和 Pseric 聯繫（http://www.freegroup.org/contact/）。

WordPress JAM 是國內第一個提供 WordPress 自訂化服務的團隊，目前已經有了相當多WordPress 案例。

WordPress JAM 長期承接 Wordpress 主題製作、網誌自訂化、WordPress 外掛自訂、WordPress 網誌 SEO 等等。

WordPress JAM 團隊陣容強大，WordPress 主題設計，外掛製作等各方面人才兼備。

Media Temple 主機是我愛水煮魚現在使用的主機，也是 Denis目前使用過最好的主機，到目前為止保持 99% 的 up time。



我愛水煮魚網誌使用過很多主機，但是最後選擇了 Media Temple，這也驗證了傳言中的國內技術網誌的主機升級之路，從國內主機到國外虛擬主機，最後都殊途終歸的選擇了 Media Temple，當然這是誇張的傳言，但是從另外一個角度這也說明了 Media Temple 主機在技術 blogger 心中獨一無二的地位。

如果你正需要為你的網誌找款主機請點選這裡直接購買 Media Temple 主機，點選這裡查看 Media Temple 主機介紹。

免費資源網路社群是以免費資源為主題的部落格，提供最新免費資訊，包含免費空間、免費軟體、Web 2.0, 網頁設計與站長工具。

WordPress 主題教學：從零開始製作 WordPress 主題

從零開始製作 WordPress 主題其實不難，只要你從現在開始認真閱讀這個教學，一步一步認真學習，你就會成為一個 WordPress 主題製作高手。至少你會修改現有主題。

網絡上已經有很多關於製作 WordPress 主題的教學，並且 WordPress 官方網站上也有指導文章。但是當你不懂這方面的術語的話，這些教學可能不一定會幫助你，甚至還會誤導你，所以這個教學會真正從零開始教你如何建立 WordPress 主題。

建立 WordPress 主題所需的工具和準備

開始真正製作主題之前，你需要使用到下面這些工具：

- 為了測試方便和快速，你首先需要在本機**安裝 WordPress**，至於如何在 Windows 系統上安裝 WordPress，你可以參考這篇文章：在 WordPress 本機安裝 WordPress。
- 如果由於某種原因不能在本機安裝 WordPress，那麼你也可以的伺服器上安裝一個測試版的 WordPress。這個時候你必須要有一個支援 WordPress 主機的伺服器，一般我使用 LAMP 主機（Linux+Apache+MySQL+PHP）主機，Win+IIS 主機可能會有很多問題，調整測試也比較麻煩，而 LAMP 主機，從我個人使用經驗來說，我推薦 (MT) Media Temple 主機。
- 原始碼編輯工具，如 NotePad++ 或者 Vim 都可以，主要是適合自己個人使用習慣。
- FTP 工具，用於上傳主題到伺服器上測試，這方面的工具很多，如 Filezilla，SmartFTP 等，如果你先安裝軟體麻煩（對啊，現在是雲端運算時代，誰還裝軟體），你也可以安裝 Firefox 的 FTP Add-ons，Fireftp，直接在 Firefox 中上傳檔案到伺服器上。
- XHTML 驗證器和 CSS 驗證器。你將需要這些工具去驗證你的主題是否符合 XHTML 和 CSS 標準，並且可以使用它查出奇正錯誤的地方。

這篇就介紹到這裡，主要介紹了製作 WordPress 主題所需的工具和應該做哪些準備，下面就開始要瞭解和開始製作 WordPress 主題。

WordPress 主題教學 #1：介紹

WordPress 主題教學 #1：介紹是從零開始建立 WordPress 主題系列教學 的第一篇。

從零開始製作 WordPress 主題的教學不會一次就教會你所有的東西，那樣也是不可能的，這個教學也不是 WordPress 主題製作的參考，我所做的是一步一步從零開始教你如何製作 WordPress 主題，所以一定要耐心。

所以這一篇介紹首先是 WordPress 主題製作的一個最基本的介紹。這裡會涉及到 HTML 和 WordPress 的基本規則，一些專業術語，以及 WordPress 主題的層式結。這些概念是很重要的，在接下來教學的很多地方都會涉及到，所以開始之前一定要搞清楚。

基本規則：

- **規則 #1**：以正確順序關閉所有 HTML 標籤。

  The right way to close:

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

  The wrong way to close:

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

  在上圖中在錯誤關閉標籤的演示中，關閉的 **ul** 標籤是不按次序的。

每個 HTML 標籤都是在 **<** 和 **>** 中，如果有斜線 **/**，則說明這個標籤是開始標籤，沒有則是結束標籤。如：**<>** 是開始標籤，而**</>** 是結束標籤。

在上面的例子中，使用 **ul**（無序列表）**li**（列表元素）標籤。注意 **li** 的開始和結束標籤在 **ul** 的開始和結束標籤的裡面，這就是標籤正確嵌套方式。

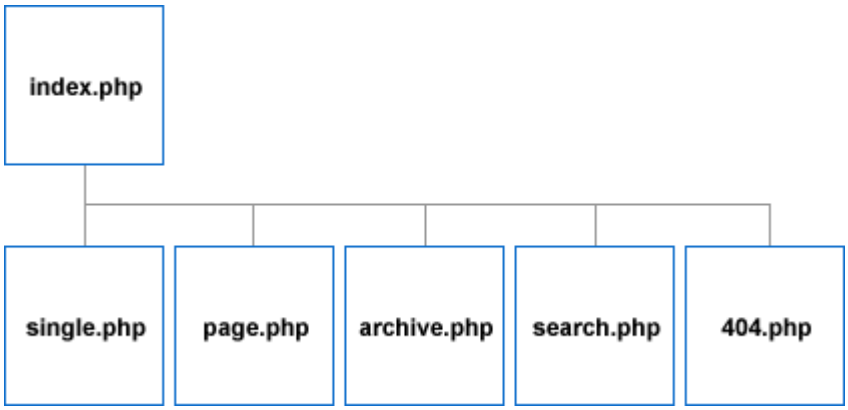
- **規則 #2**：每個主題至少要有這兩個檔案 - **style.css** 和 **index.php**。index.php 告訴主題中所有的元素如何佈局，style.css 則告訴主題中所有的元素該如何展示和樣式。下面是一個完整的主題含有的檔案列表（現在我們不用詳細瞭解這個列表每個檔案的意思，有個這樣的印象就可以了）：
    - style.css
    - index.php
    - home.php
    - single.php
    - page.php
    - archive.php
    - category.php
    - search.php
    - 404.php
    - comments.php
    - comments-popup.php
    - author.php
    - date.php

專業術語：

- **Template（模板）** -- 其實就是一個原始碼集，主題中很多地方會利用到這個原始碼集，所以把它們整合成一個模板，這樣就就不必一遍遍輸入這些重複原始碼。
- **Template file（模板檔案）** -- 一個包含一個或者多個原始碼集（模板）檔案。每個主題是由多個模板檔案組成的，如：index.php，style.css，sidebar.php 等等。
- **Theme（主題）或者 WordPress theme（WordPress 主題）** -- 所有你正在使用的檔案：文本，圖片，原始碼等等。注意： WordPress theme（主題）和 WordPress template(s)（模板）是兩個不同的東西，儘管有些人認為他們一樣。
- **Post（文章）** -- 現在你讀的就是一篇文章。此外，它是你 blog 的一個簡單的項目，如：一個頁面或者一篇日記。
- **Page（靜態頁面）** -- 一種特殊的 post，它不是以分類組織的。它有別於你其他的文章。注意：在 WordPress，page（頁面）和 Page（靜態頁面）是兩種不同的東西。

層式結構：

下圖就是 WordPress 的層式結果，它簡單的向你展示，一旦你主題中的某個檔案遺失了，WordPress 主題系統將會尋找什麼模板檔案來代替。這裡列出了 6 個檔案而不是完整的 13 個，因為這 6 個是相對更重要一些，不過在接下來的教學中，餘下的檔案也都涉及到。



我們可以透過上面這張圖的所處位置知道各個主題檔案的重要性，越靠左越重要。

這裡可能大家有個疑問，為什麼會存在 WordPress 模板檔案的層式結構，或者說是重要性級別呢？因為 WordPress 利用這個層式結構去尋找相應的模板檔案顯示頁面，並且在相應的檔案遺失之後如何處理。

如果 archive.php 模板檔案（用來顯示存檔頁面）遺失了，那麼 WordPress 將會使用 index.php 來控制存檔頁面如何顯示。

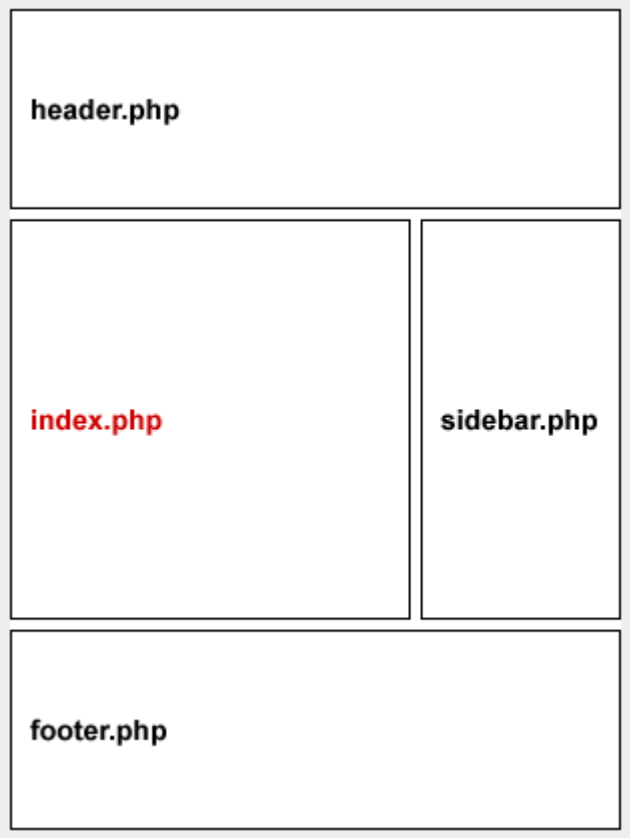
如果 single.php 模板檔案遺失了呢，哪個模板檔案它會去尋找用來顯示單一文章呢？它會尋找 index.php。

WordPress 主題教學 #2：模板檔案和模板

模板檔案（**template files**）和**模板**（**template**）是從零開始建立 WordPress 主題系列教學的第二篇。開始之前，你要確保你已經看過WordPress 主題教學 #1：介紹，否則你將無法理解在教學 #2 中使用的名詞。

在WordPress 主題教學 #1：介紹中，我們已經學過了 WordPress 的兩條基本規則和術語，而這篇將會深入講解模板檔案，模板，以及每個頁面的結構。

WordPress 網誌的**每個頁面是由多個模板檔案組成的**，下面是首頁的例子：



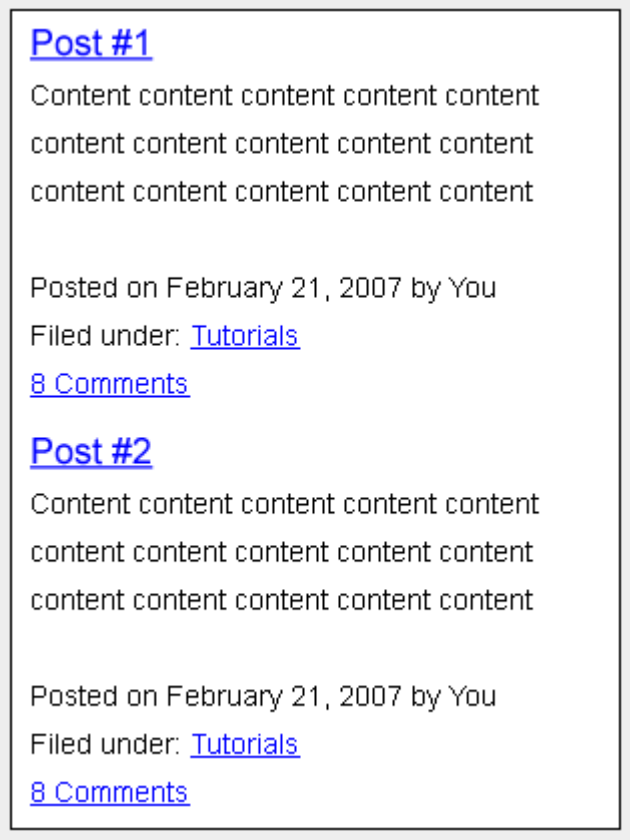
在上圖中，我們可以看出主題的 index.php 是由 4 個模板檔案組成： header.php，index.php，sidebar.php 和 footer.php。

Header 模板檔案:

通常在這個檔案中包含網誌的**標題**（title）和**描述**（description）。而且它們通常在整個網誌中都是一樣的。

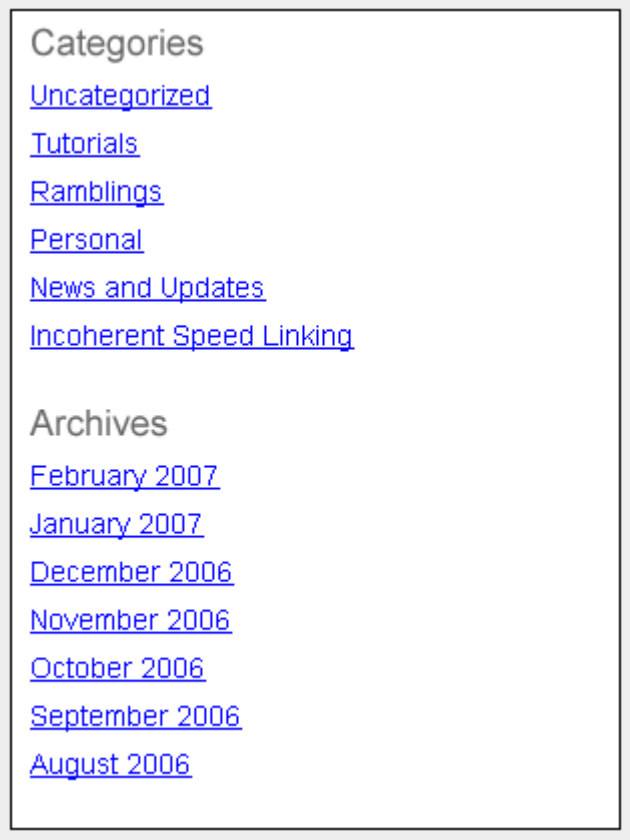
Index 模板檔案：

這個模板檔案包含你的**文章的標題**，**文章的內容**（就是每篇文章的文本和圖片）和**文章的元資料（Metadata）**（元資料是每篇文章的額外訊息，如作者是誰，文章發佈的時間，在哪個分類下，有多少留言等等）。



Sidebar 模板檔案

這個模板檔案主要用於控制網誌的頁面列表，類別列表，存檔列表，友站連結列表和其他一些列表。

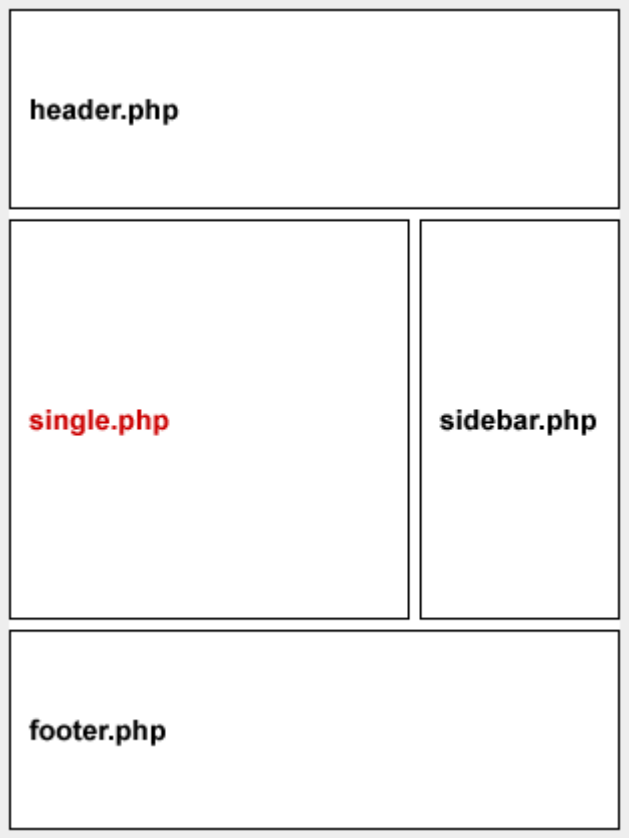


Footer 模板檔案：

像 header.php 模板檔案一樣，footer.php 通常不會因為頁面的改變而改變，你可以在這裡放置任何東西，但是通常是**版權訊息**。

現在讓我解釋為什麼把上面圖片中的 index.php 所在的區域標為紅色的。引文這塊區域是會根據不同類型的頁面而發生變化。

如果你在單一文章頁面，這時候頁面將會包含這四個模板檔案：
header.php，**single.php**，sidebar.php 和 footer。



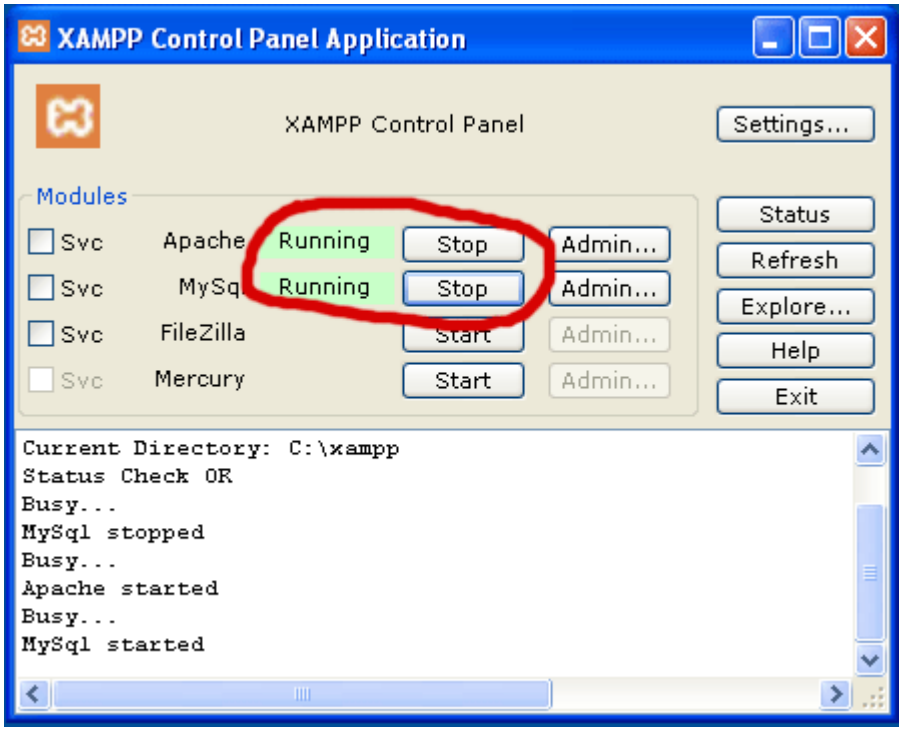
WordPress 主題教學 #3：開始 Index.php

開始 Index.php 是從零開始建立 WordPress 主題系列教學的第三篇。在介紹了 WordPress 主題的一些規則和術語，以及對 WordPress 模板和模板檔案瞭解之後，現在是開始動手建立 WordPress 主題了的時候。

在這篇中，你將要著手開始寫 WordPress 原始碼。這裡建議在本機電腦上安裝 WordPress，而不是安裝到伺服器上，因為本機更方便測試。

第1步：打開 XAMPP 控制台。

在 XAMPP 檔案夾（通常是：**C:\xampp**），執行 **xampp-control.exe** 將會彈出一個新的視窗。點選 Apache 和 MySQL 的啟動按鈕。如下圖所示：



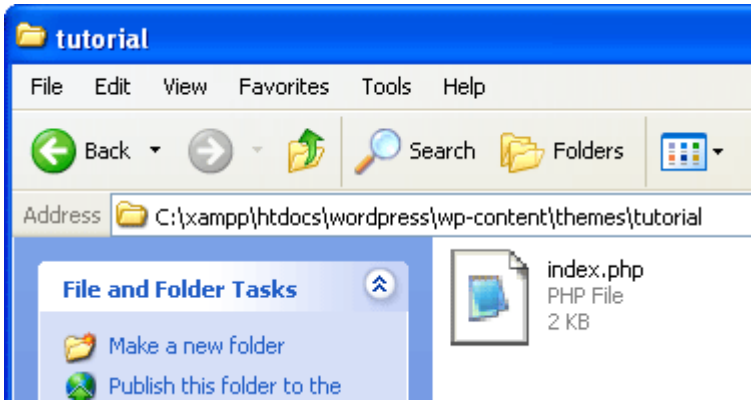
啟動之後你看最小化視窗了。

第2步：建立你的主題檔案夾。

轉到你本機安裝的 WordPress 主題檔案夾，應該在**xampp/htdocs/wordpress/wp-content/themes**。建立一個新的檔案夾，命名為 **tutorial**。

第3步：建立 index.php 和 style.css 檔案。

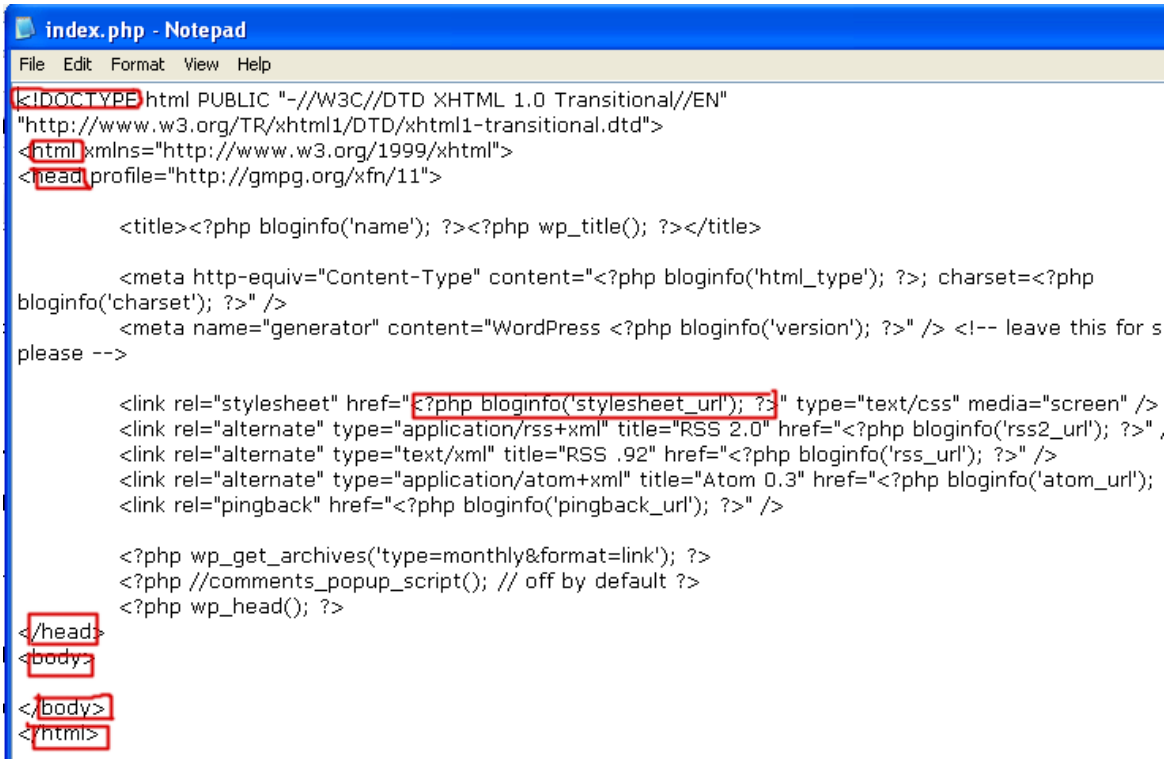
打開記事本或者你選擇的文字編輯器，把 index.txt 這個檔案中的所有內容都拷貝到你的記事本。儲存為 **index.php**。



打開另外一個記事本，直接儲存為 **style.css** 到相同的檔案夾下.

現在有兩個檔案了: index.php 和 style.css.

index.php 解釋：



點選上面的圖片查看大圖。我會向你解釋每個紅色圓圈區域是什麼意思。

Doctype - 指明你用哪種類型的原始碼來編碼你的主題。這裡你暫時還不用管它的詳細意思。

<html> 是網頁開始的地方。

<head> 是你的網頁頭部開始的地方。每個網頁都有一個頭部和主體。**</head>** 是頭部結束的地方。

<?php bloginfo("stylesheet_url"); ?> 是一個 PHP 函數，它能取得 style.css 檔案所在的路徑，這樣主題就能使用 style.css 樣式化頁面上所有元素。任何時候，PHP 原始碼都是在 **<?php** 和 **?>** 之間的。PHP 原始碼和 HTML 的原始碼是不一樣的，在 PHP 中，**<?php** 代表開始 PHP 原始碼而 **?>** 是結束 PHP 原始碼。

所以：

- **<?php** - 開始 PHP 原始碼
- **bloginfo("stylesheet_url")** - 導入 style.css 檔案所在的路徑
- **;** - 停止導入函數。分號是用來結束一個 PHP 語句。
- **?>** - 結束 PHP 原始碼

<body> - 這是網頁主體開始的地方。你能在網頁上看到和讀到的東西就是主體部分。你正在閱讀的這個教學說明你在正在看當前這個網頁的主體部分。**</body>** 是網頁主體結束的地方。

</html> 是網頁結束的地方，沒有東西在它的後面了。

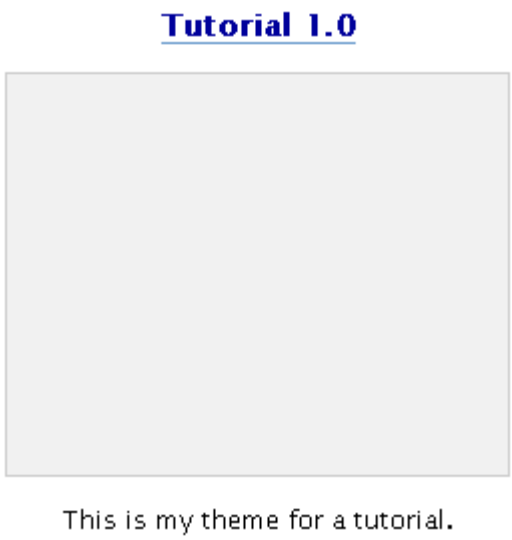
第4步：建立 style.css。

把 style.txt 中所有的原始碼拷貝到你的 style.css 檔案中。儲存和關閉它。

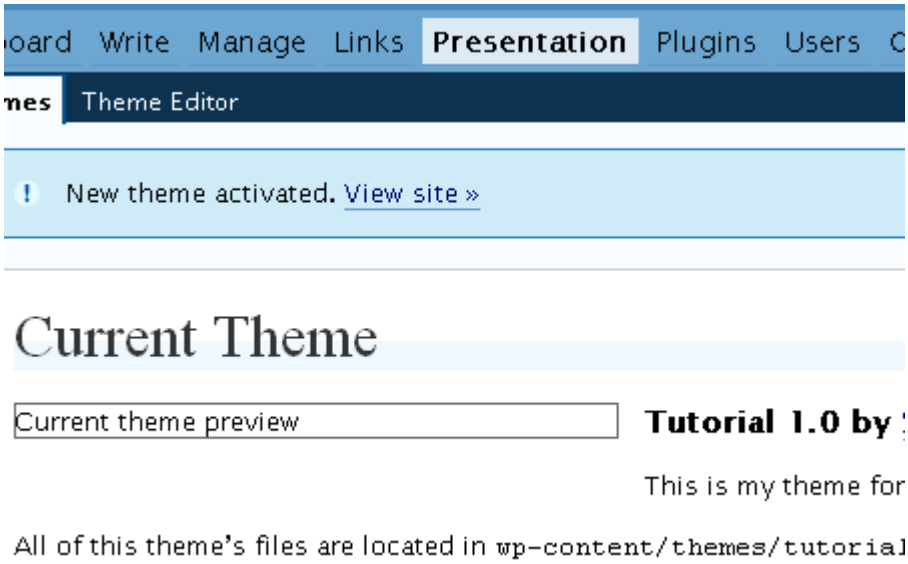
第5步：安裝你的主題。

打開瀏覽器。在網址列輸入輸入 http://localhost/wordpress/wp-login.php。登入到你的 WordPress 管理後台。（這裡能夠看到 WordPress 登入頁面是因為你在第1步的時候啟動了 Xampp。否則的話，在這裡你的瀏覽器會出現找不到的錯誤。）

在管理界面下到 **外觀 (Apperance)** 選單並啟動名為 **Tutorial** 的主題。



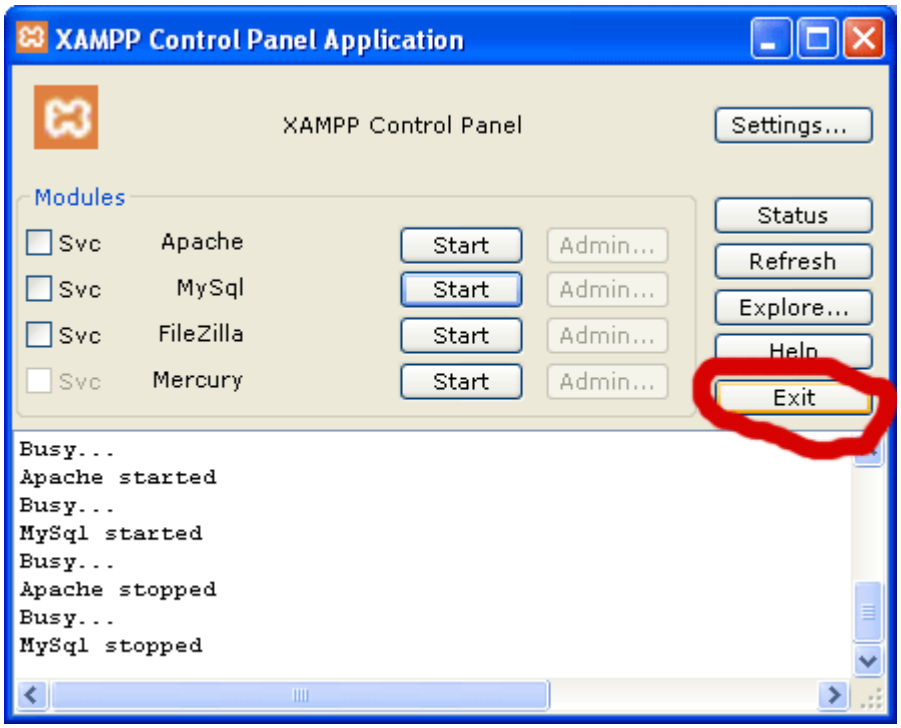
注意，你的主題檔案沒有畫面略縮圖，所以是空白的。一旦啟動了，WordPress 就會告訴你啟動訊息。



現在打開一個新的瀏覽器或者標籤頁（如果你的瀏覽器支援標籤頁瀏覽）並在網址列輸入 **http://localhost/wordpress**。你應該得到一個空白頁面，嗯，完全空白的頁面。如果不是，那你就是在錯誤的頁面上了。

你的主題已經建立好了，這就是這個課程，下一步我們將討論主題頭部模板。

不要忘記關閉 Xampp。點選它在任務欄中小圖標，點選 Apache 和 MySQL 的停止按鈕，然後點選推出。



WordPress 主題教學 #4a：Header 模板

Header 模板是從零開始建立 WordPress 主題系列教學的第四篇。前面我向你講解了如何安裝和啟動 XAMPP，安裝 WordPress 主題以及介紹了 PHP 語言的最基本語言，這篇我們將繼續 PHP 並學習如何導入網誌的標題和連結。

盡量輸入所有原始碼而不是直接拷貝我給你的原始碼，這樣可以讓你盡量記住你所學到的的。

第1步：打開 XAMPP 和主題檔案夾。

打開 Xampp，然後打開上次建立的主題檔案夾，**xampp/htdocs/wordpress/wp-content/themes/tutorial**。我們應該看到上次建立的兩個檔案：index.php 和 style.css。

index.php 和 style.css 檔案的內容應該和index.txt 和 style.txt 一致。

第2步：打開 index.php

打開瀏覽器，轉到 **http://localhost/wordpress**。因為上次安裝了一個空白的主題，這時我們應該看到一個空白的頁面。

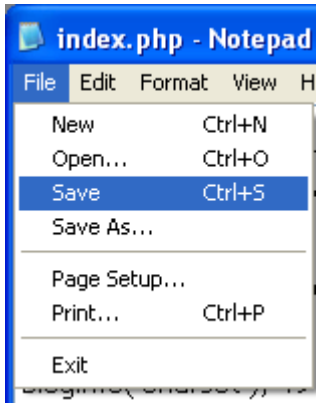
返回主題檔案夾並打開 index.php 檔案。

到目前為止，我們已經打開了**主題檔案夾**，**瀏覽器**和 **index.php 檔案**。

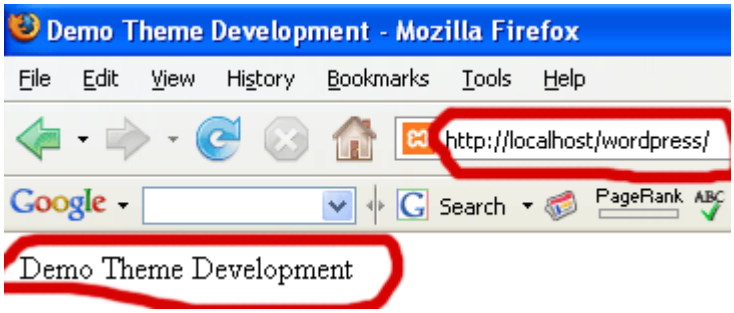
第3步：導入網誌標題

編輯 index.php 檔案。在 **<body>** 和 **</body>** 這兩個標籤之間輸入 **<?php bloginfo('name'); ?>**，然後儲存它。

```
<body>

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

</body>
```

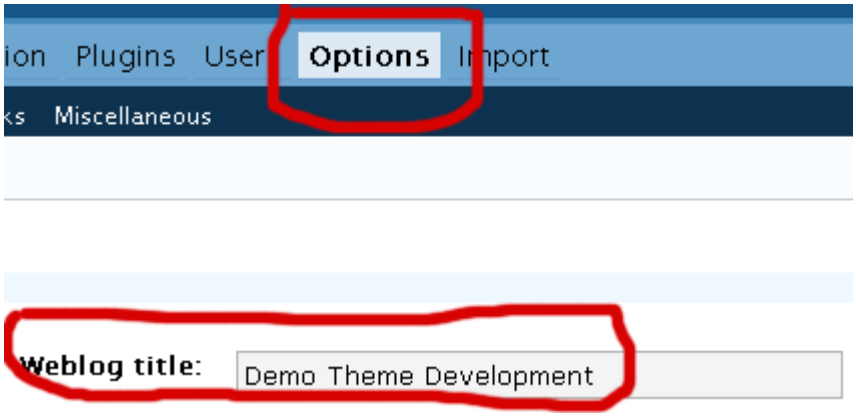


返回到瀏覽器並**刷新**。這時候我們應該能夠看到網誌的標題。網誌的標題是 Demo Theme Development。



剛才發生什麼了？

我們在網頁的主體 (body) 之間加入了一行 PHP 原始碼到 index.php。**bloginfo()** 是導入網誌的訊息的函數。其中參數 name 代表了它導入的是網誌的標題。這個名字是在 option 頁面中設定的 **Weblog Title**。



<?php - 開始 PHP 原始碼
bloginfo('name') - 導入網誌訊息，具體是網誌的標題。
; - 結束導入網誌訊息
?> - 結束 PHP 原始碼

每次我們在 index.php 檔案中增加或者更改任何東西之後，都可以儲存，然後重新整理網頁去查看結果。

第4步：導入網誌連結

導入了網誌的標題之後，接下來就要把網誌的標題放入超連結中，這時候需要一個 XHTML 標籤。

返回 index.php 檔案。

在同一行增加 **** 和 ****。此時新行的原始碼應該是：
<?php bloginfo('name'); ?>

返回到瀏覽器，重新整理，然後就可以看到網誌的標題變成了連結。

現在它是一個連結，但是它沒有連結到哪裡。因為這個是網誌的標題，我們應該讓它連結到首頁。為此，在 **href=**後的雙引號中輸入 **<?php bloginfo('url'); ?>**

儲存，現在的原始碼應該是：
<a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?>

返回到瀏覽器，重新整理，當游標在連結上面的時候，瀏覽器的狀態欄應該顯示 **http://localhost/wordpress**

現在點選這個連結，它就會讓我們返回首頁。可能現在看到還是是相同的頁面，但是用 **#** 或者 **http://localhost/wordpress** 作為連結地址是完全不一樣的。在接下來的課程我們會學到到他們之間的不同。

剛才發生什麼了？

我們把網站名字變成了連結，並使它連結到網誌的主頁。

bloginfo('url') - 導入網誌基本訊息，實際是首頁的的網址或者 URL
<a> - 是一個用於加入連結的 XHTML 標籤
**** - 連結的結束標籤。否則網頁將不知道哪裡結束連結，這樣會使得頁面接下來的內容全部都變成連結。還記得**規則 #1**嗎？正確關閉打開的所有標籤。
href="" - 超連結的簡寫。在引號之間就是它的值。

最終原始碼為：
<a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?>

意思為：開始一個連結，連結的網址是網誌的URL，用 PHP 函數 **bloginfo('url')** 去導入這個網址或者URL。這個連結的文本是網誌的標題並使用 PHP 函數 **bloginfo('name')** 去導入網誌的標題。最後結束連結。

這篇主要介紹了 WordPress 主題的 XHTML 原始碼，下一篇我們將繼續 Header 模板。

WordPress 主題教學 #4b：Header 模板 2

Header 模板 2是從零開始建立 WordPress 主題教學系列教學的第四篇第二部分。最後說一次，開始之前務必先讀下前面的文章。這篇會完成 Herder 模板，並且開始介紹 DIV Box 模型。

第1步：開啟 XAMPP 和打開 index.php

- 啟動 Xampp
- 打開 Tutorial 的主題檔案夾
- 打開瀏覽器，在網址列輸入 http://localhost/wordpress
- 返回主題檔案夾，用記事本打開 index.php

第2步：給網誌的標題加入 H1 的標籤

現在，index.php 的原始碼是：

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

給它加入 **<h1>** 和 **</h1>** 標籤。H1 標籤意思是標題一。HTML 一共可以有7級標題：H1，H2，H3，H4，H5，H6。按照預設，H1是字型最大而H6是則最小。

加入之後的的 index.php 檔案是：
<h1><a href="<?php bloginfo("url"); ?>"><?php bloginfo("name"); ?></h1>

儲存，返回瀏覽器並重新整理。

第3步：加入網誌描述

導入網誌的描述，則在網誌標題連結的下面輸入以下原始碼： **<?php bloginfo("description"); ?>**。

現在變成了：

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

儲存並重新整理瀏覽器，可以看到在網誌標題連結的下面出現網誌的描述，我們可以到 WordPress 管理後下修改網誌的描述。

<?php - 開始 PHP 原始碼
bloginfo("description") - 導入網誌訊息，這裡的是描述
; - 停止導入
?> 結束 PHP 原始碼

第4步：DIV 標籤

這步將介紹一個新的標籤 -- DIV。

給以上原始碼加入 ** ** 和 ** **標籤：

** **

<h1>"><?php bloginfo("name"); ?></h1>
<?php bloginfo("description"); ?>

** **

儲存，重新整理瀏覽器，應該看到沒有任何變化

可以把 **DIV** 想像成一個無形的盒子 (box)。在這裡它把**網誌標題連結**和**網誌描述**從其他東西中區分開。如果沒有對它進行樣式化，它無非是單獨的內容，以後我們可以會用 style.css 這個檔案去樣式化這個無形的盒子。我們可以給 DIV 附上 **邊框（borders）**，**填充（paddings）**，**頁邊空白（margins）**，**背景顏色（background color）**，**背景圖片（background images）**，以及其他一些東東。

第5步：加入 Header DIV 標籤

加入 **id="header"** 到 DIV 標籤，如下：
** **

儲存並重新整理瀏覽器。

同樣也沒有改變，這裡給 **DIV** 標籤指定了 **ID** ，因為將會有更多的 DIV 標籤或者無形的盒子，我們使用 ID 來區分！

WordPress 主題教學 #5：主循環

導入網誌文章的**主循環（The Loop）**是 WordPress 中最重要的 PHP 原始碼集，幾乎所有的頁面都會用到它。這也是從零開始建立 WordPress 主題系列教學的第五篇。

在開始繼續學習之前，我們先複習下到目前為止學到了什麼？

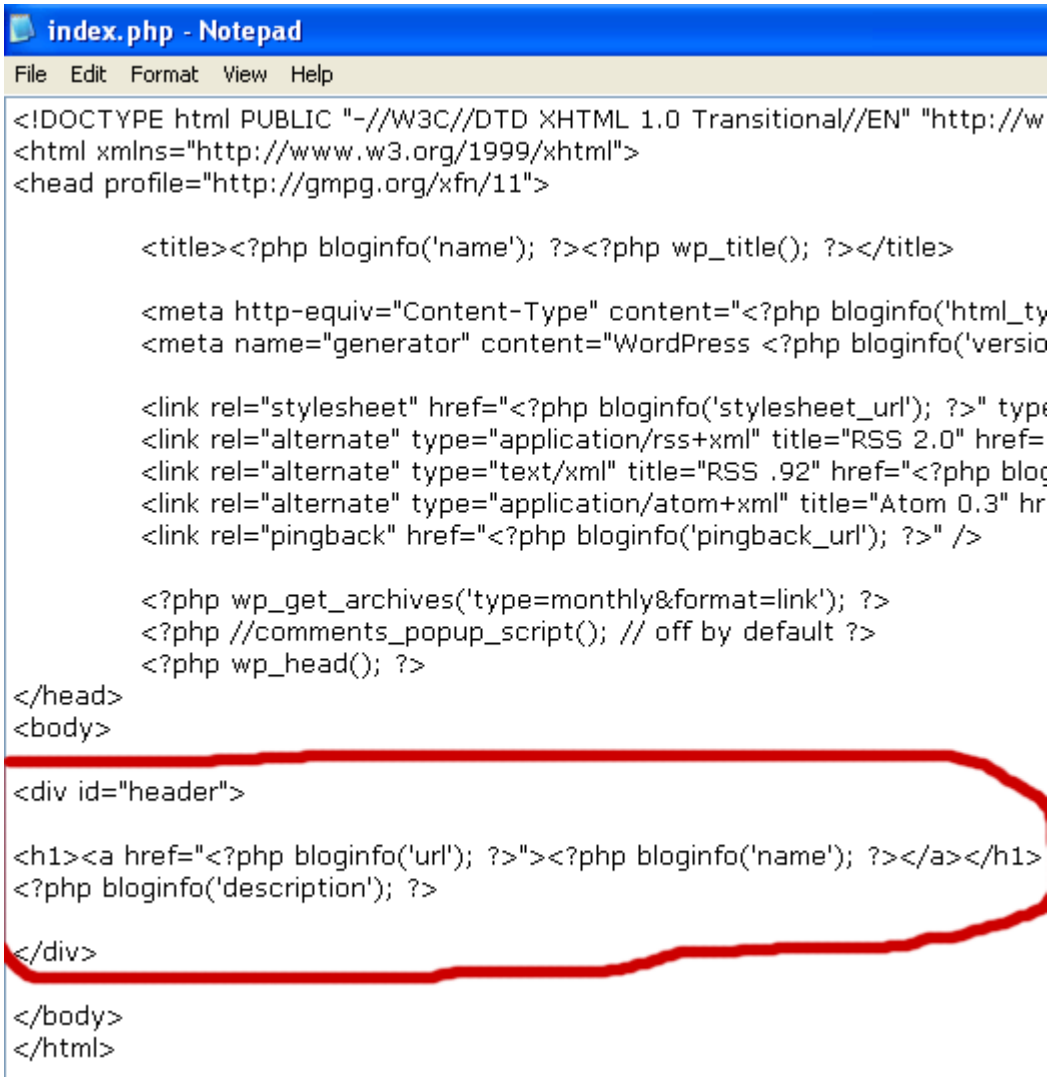
到目前為止，我們已經學到：:

- 規則，術語和 WordPress 主題的層式結構
- 每個頁面有哪些部分組成
- 如何安裝你的主題
- 如何導入網誌的標題和把它做成一個連結
- 如何導入網誌的描述和如何把 header 和其他部分分開

現在讓我們開始第五篇：**主循環（The Loop）**

打開 Xampp，「tutorial」主題檔案夾，瀏覽器，並且在瀏覽器中轉到 http://localhost/wordpress，最後打開 index.php 檔案。

下面應該是這時候 index.php 檔案中的內容：



記住，為了學習這些原始碼，請盡量手工輸入而不是複製和貼上。

第1步：建立 container Div

在 header DIV 標籤下加入一個 DIV 標籤，並給它的 ID 賦值為 「container」，如下：

** **

** **

「container」 這個 DIV 標籤是用把網誌的主要內容和其他東西都區分開，比如 sidebar 和 footer 等。

第2步：輸入主循環原始碼

在 Container 的 DIV 標籤中加入如下原始碼：

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

<?php endwhile; ?>

<?php endif; ?>

這段原始碼就是 WordPress 中的**主循環（The Loop）**。在詳細解釋這些原始碼作用之前，我們來看下現在 index.php 所包含的原始碼：

```
<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>
```

你可能已經注意到**Container DIV 中的每一行都被縮進了**，這是為了更好的**組織原始碼**，更加利於閱讀(**使用 tab 鍵而不是空白鍵進行原始碼縮進**，)。

剛才發生了什麼？

- **if(have_posts())** - 檢查網誌是否有文章。
- **while(have_posts())** - 如果有文章，那麼當網誌有文章的時候，執行下面 the_post() 這個函數。
- **the_post()** - 導入實際的文章來顯示。
- **endwhile;** - 遵照規則 #1，這裡用於關閉 **while()**
- **endif;** - 關閉 **if()**

- **註釋**：並不是所有的原始碼都需要兩部分用來打開和關閉。有些原始碼能夠自我關閉，這就解釋了 **have_posts()** 和 **the_post();** 這兩個函數。因為 **the_post();** 在 **if()** 和 **while()** 的外面，只需要分號去結束或者關閉。

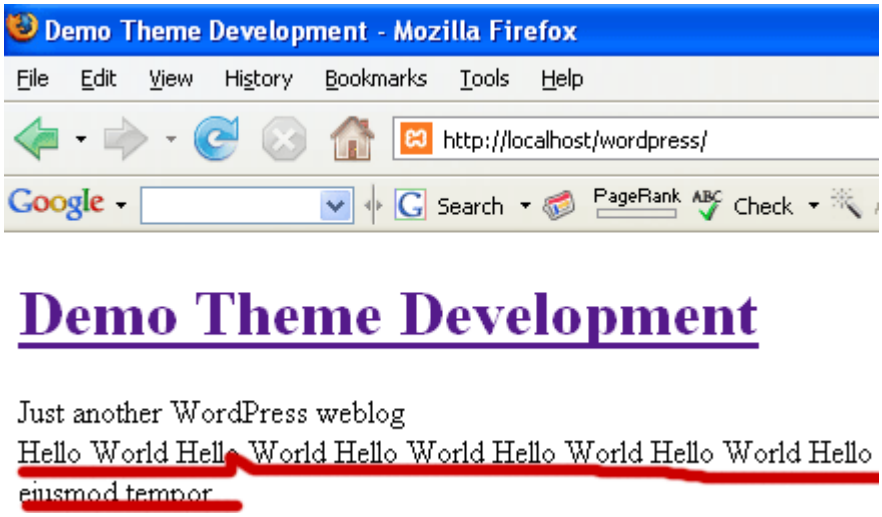
第3步：導入文章標題

在前面的課程中，我們學習了使用 bloginfo('name') 去導入網誌的標題。現在我們將學習在**主循環（The Loop）**中如何導入**文章標題**。

在 **the_post(); ?>** 的後面和 **<?php endwhile; ?>** 的前面輸入 **<?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>
```

儲存 index.php 檔案並重新整理瀏覽器，這時候應該看到在網誌描述的下方出現 **Hello World** ，預設安裝 WordPress 之後，網誌只有一篇文章。而我的測試的網誌有多篇文章，所以這裡有多個文章標題，而且因為我所用的文章標題是一樣的，我也沒有進行組織整理他們，所以它們看起來像很長的一行 Hello World。



第4步：給文章標題加上連結

把**文章標題**轉變成**文章標題連結**。還記得怎樣吧網誌的標題轉變成一個連結的？

在**<?php the_title(); ?>** 兩邊增加 **** 和 ****。

儲存並重新整理你的瀏覽器。現在文章的標題都變成了連結了，但是它們並沒有指向哪裡。為了使得每個標題都能指向正確的文章，我們需要把 **#** 替換為 **the_permalink()**。

<a href="**<?php the_permalink(); ?>**「><?php the_title(); ?>

the_permalink() 是用來導入每篇文章網址的 PHP 函數。儲存並重新整理瀏覽器。

如果只有一個 **Hello World** 標題，把游標移到連結上面，觀察你的瀏覽器底部的狀態欄，他不再是 **http://localhost/wordpress/#**。

如果有不止一個的標題連結，我們將看到每個連結會鏈到不同的文章或者網頁。但是我們的文章標題依然在同一行上面。為了分開它們，在文章標題連結原始碼的兩邊加入 **<h2>** 和 **</h2>** 標籤。

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

記住 **H1** 用作你的網誌的標題，那是網頁的標題。**H2** 被用作子標題。現在你的文章標題連結是子標題了，每一個都是一行。儲存 index.php 檔案並重新整理瀏覽器，結果如下：

WordPress 主循環就介紹到這裡，現在 index.php 檔案內容應該是：

```
<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>
```

WordPress 主題教學 #5b：文章內容

文章內容是從零開始建立 WordPress 主題系列教學第五篇的第二部分，在這篇中，我們將展示如果顯示網誌文章的內容，並且使用一個 DIV 標籤把網誌文章的**內容**和文章的**標題**區分開。再次強調一次，上一篇關於 WordPress 主循環介紹的課程非常重要，你需要徹底明白之後才能繼續學習。

下面開始這篇課程。首先還是打開 XAMPP，「tutorial」主題檔案夾，瀏覽器並在瀏覽器網址列輸入：http://localhost/wordpress，最後打開 index.php。

第1步：使用 the_content() 函數顯示文章內容

在文章標題原始碼下面輸入：**<?php the_content(); ?>**。

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

<?php endif; ?>
```

儲存並重新整理瀏覽器，現在在文章標題下面看到了一些文本：

Demo Theme Development

Just another WordPress weblog

Hello World

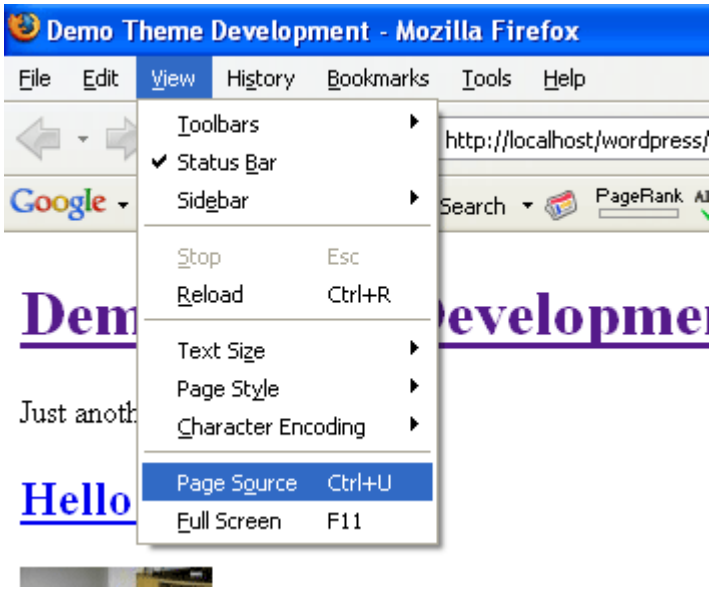
 Lorem ipsum dolor sit amet, consectetur adipisi
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor i
officia deserunt mollit anim id est laborum.

剛才發生什麼了?

我們使用了 PHP 函數 **the_content()** 導入了 **文章的內容**。現在，文章的內容只是一長行的文本，一直到視窗的右邊，因為我們還沒有樣式化它。還記得最開始說到的 **style.css** 這個檔案嗎？我們以後用它來控制所有頁面元素的顯示和佈局。

我們在 WordPress 後台輸入多篇多篇測試文章，就可以看到多篇文章一起被顯示的樣子：

返回瀏覽器，點選「查看」選擇「頁面原始碼」，就會跳出一個原始碼視窗，如果你使用的是 **Internet Explorer**，那麼跳出的是記事本。



我使用的是 **Firefox**瀏覽器，下面是在 FireFox 中顯示的樣子：

```
<div id="container">


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

                <p><img id="image13'
alt="dcfn0032.JP" />Lorem ipsum do.
ad minim veniam, quis nostrud exerc:
velit esse cillum dolore eu fugiat :
laborum. <img src='http://localhost,
<p><img id="image14" src="http://loc
dolor sit amet, consectetur adipisi(
exercitation ullamco laboris nisi ut
fugiat nulla pariatur. Excepteur si:
<p><code>Lorem ipsum dolor sit amet,
veniam, quis nostrud exercitation u.
cillum dolore eu fugiat nulla pariat
laborum.</code></p>
<p>Lorem ipsum dolor sit amet, conse
quis nostrud exercitation ullamco l
href="http://localhost/wordpress/in(
</p>
```

你注意到 index.php 檔案和它的**原始碼**之間的區別了嗎？所有的文本，圖片和其他東西等所有上圖展示的東西都是透過 **the_content()** 這個函數導入來的。是不是很有用？注意這些原始碼是不依賴實際的 WordPress 主題，我們應該自己的這些文本和圖片進行編碼和樣式化。

還有，有沒有注意到我圈出的開啟和關閉的**P**標籤。他們都沒有在 index.php 檔案中出現，但是他們在原始碼中出現了。

P 標籤，為什麼和如何使用？

為什麼 - 當我們輸入文章的時候，每次跳過一行就是一個段落，這個時候需要一個方法去展示？我們可以透過 **P** （段落，paragraph）標籤，每個段落會在 P 標籤之間，這就是為什麼段落之間有行距的原因，

如何使用 - 非常容易，WordPress 模板系統會自動幫我們產生 **P** 標籤。

第2步：DIV 標籤把網誌文章的內容和標題區分開

給 **the_content()** 兩邊加入 DIV 標籤並給該 DIV 標籤附上**class="entry"**屬性，如下：

** **

** **

你現在的 index.php 檔案應該是：

```
<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>
```

儲存並重新整理瀏覽器，我們再次去查看**原始碼**的話，就會發現每篇**文章內容**在 class="entry" 的 DIV 標籤中。

這樣我們就很容易知道**文章標題**在哪裡結束和**文章內容**在哪裡開始，這樣做也是以後使用style.css 檔案對它進行樣式化做準備，透過 class 我們就可以準確定位到文章內容，樣式化文章的內容而不影響頁面上其他別的內容。

id 和 **class**之間有什麼區別呢？

到目前為止，對於每個 DIV 標籤，我們可以用 **id** 去命名它，如 **id="header"**，那麼**id** 和 **class**之間有什麼區別呢？**id** 是唯一的而 **class**不是。如果從頭到尾瀏覽原始碼，你會發現只有一個 id="header" 和一個 id="container"，但是有多個 class="entry"。

那麼 **header** 和 **container** 可以用 **class** 去取代 **id** 嗎？完全可以。

但是不能重複任何 **id**，比如，不能在同一頁面上有兩個 id="header" 。當你想一遍又一遍重新利用一些東西如**文章的標題**，那麼請使用 **class**。

第3步：給文章的標題和內容加入 class="post" 的 DIV 標籤

用一個 DIV 標籤把文章的標題和內容一起圍住。並把這個 DIV 標籤命名為：class="post"。

** **

** **

（class 和 ID 的名字不是一定要嚴格和上面一樣，可以把 class 和 ID 的名字設定任何你想要的名字，但是 post 和 entry 更加簡潔明瞭，也更容易記。）

現在你的 index.php 檔案為：

```
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; ?>
```

這個是經過縮進整理後的版本：

```
<div class="post">

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

	<div class="entry">

		<?php the_content(); ?>

	</div>

</div>
```

一般我們使用 **tab** 鍵而不是空格鍵產生縮排的。**為什麼進行要對原始碼進行縮排呢？**實際上的原始碼不像我上面的畫面截圖一樣有紅色或者綠色的高亮顯示，我們需要有個能夠跟蹤原始碼的方法，透過縮排就能更容易知道哪個 ** ** 是結束哪個 ** **。

儲存並重新整理瀏覽器，然後查看原始碼中的原始碼。

為什麼你要加入另外一個 DIV 標籤去圍住文章標題和文章內容？

增加這個 DIV **class="entry"** 去把**文章標題**和**文章內容**區分開。而這個**div class="post"**是把當前文章和其他內容區分開。

WordPress 主題教學 #5c：文章元資料（Metadata）

文章元資料（Metadata）是從零開始建立 WordPress 主題系列教學的五篇的第三部分，今天我們將開始講解文章的元資料（Metadata）（Postmetadata）：**日期（date）**，**分類（categories）**，**作者（author）**，**評論數（number of comments）**，以及其他和文章有關係的訊息。

同樣請打開 XAMPP，主題檔案夾，瀏覽器以及 index.php 檔案。

先讓我們複習下，現在的 index.php 檔案應該有下面這些原始碼了：

index.php - Notepad

File Edit Format View Help

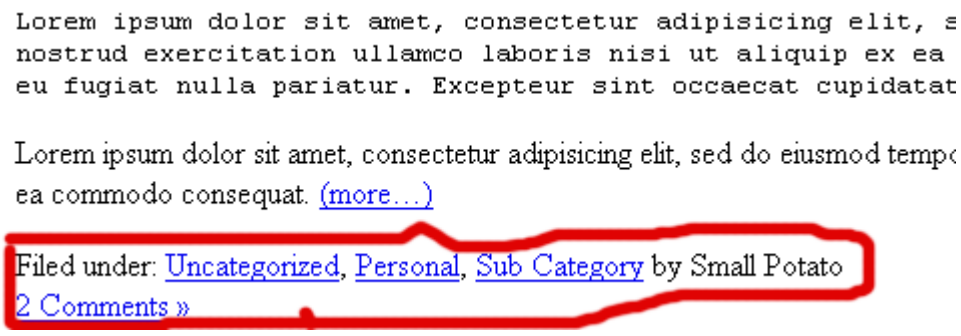
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head profile="http://gmpg.org/xfn/11">

	<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>
```

把 postmetadata.txt 中的原始碼複製到 **<?php the_content(); ?>** 下面。（注意：在這部分，我們只需要複製和貼上。當我製作 WordPress 主題的時候，我也只是複製和貼上這部分原始碼。對於這部分原始碼，你不需要完全理解它，只要知道每部分幹什麼已經足夠了。）

下面的畫面截圖是為了適應文章的大小而只裁剪了一部分，它主要你關注文章元資料（Metadata）原始碼的位置：

```
<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>
```

儲存並重新整理瀏覽器，現在應該是：



我們同樣可以透過查看原始碼來看文章元資料（Metadata）是怎樣的？

詳細解釋：

** ** 和 ** ** - 所有的文章元資料（Metadata）都在一個 **class="postmetadata"** 的段落標籤中，因為我想把文章元資料（Metadata）和文章內容區分開。如果沒有段落標籤，文章元資料（Metadata）訊息將在文章內容結束的地方繼續，這樣就沒有任何間距去區別內容和文章元資料（Metadata）。

<?php _e("Filed under:"); ?> - : 是導入冒號「:」的原始碼；
把**Filed under:**放入**<?php _e(" "); ?>**中不是必須的，這樣主要為了使得 **Filed under:** 可翻譯。如果你的主題不需要支援多語言，可以簡單輸入 **Filed under:**；

<?php the_category(", ") ?> - the_category() 是用來導入文章的在的所有類別的 PHP 函數。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() 中的逗號是用來區分類別名。返回文章元資料（Metadata）的畫面截圖，我們就可以注意到在類別連接中的逗號；

<?php _e("by"); ?> - 和 **Filed under:**一樣。如果你建立的是私人用的的主題， **by** 外面的 **_e()** 不是必須的。**_e()** 是用來建立可以翻譯的主題，如果主題被來自不同國家的上百人使用的話，這是非常重要的。如果你是建立公共使用的主題，最後加上 **_e()** 以便你的主題可翻譯化。

<?php the_author(); ?> - 它是輸出當前文章作者的名字。

**
** - 如果你想要一個空行，又不想用段落標籤來產生行間距，使用 BR。注意斜線 / 。這是能自我關閉的標籤。

<?php comments_popup_link("No Comments »", "1 Comment »", "% Comments »"); ?> - 當跳出留言的功能啟動的話，**comments_popup_link()** 導入一個跳出的留言視窗，如果沒有啟動，**comments_popup_link()** 則只是簡單的顯示留言列表。**No Comments ?** 是在沒有留言的時候顯示的。**1 Comment »** 是用於當剛好只有1條留言時候。**% Comments &187;** 是用於當有多於一條留言的時候。比如：8 Comments »。百分號 % 用來顯示數字。» 是用來顯示一個雙層箭頭 ?。

<?php edit_post_link("Edit", " | ", "); ?> - 這個只有當我們以管理員或者作者身份登入的的時候才可見。 **edit_post_link()** 只是簡單顯示一個可以用來編輯當前文章的編輯連結，這樣就可以讓我們不必去管理界面搜尋該文章就能直接編輯。**edit_post_link()** 有三個參數。第一個是用來確定哪個詞你將用在編輯連結的連結標題。如果你使用 **Edit post**，那麼將顯示 **Edit post** 而不是 **Edit**。第二個參數是用來顯示在連結前面的字符，在這裡是豎線 **|**，原始碼就是**&124;**。第三個參數是用於顯示在編輯連結後面的字符，在這裡沒有使用。

登入 WordPress 之後，再返回到首頁就可以看到「**Edit**」的連結和一條豎線。

WordPress 主題教學 #5d：Else，文章 ID，連結標題

Else，文章 ID，連結標題是從零開始建立 WordPress 主題系列教學的五篇的第四部分，這篇課程將講解其他3個可以增加到文章中的元素：**Else**，**post ID**, 和 **連結的 title 值**。儘管它們是可選的，但是我們幾乎可以在我每一個免費的主題中都能找到。

開始之前，不要忘記啟動 Xampp。

第1步：Else

在 **<?php endwhile; ?>** 的下面輸入以下原始碼：
<?php else : ?>

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

** **

大致如下：

```
        <?php endwhile; ?>

        <?php else : ?>

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

        <?php endif; ?>
```

儲存重新整理瀏覽器，但是應該注意到沒有任何變化。我們返回教學 #5 -- 主循環，去解釋你剛才上面輸入的是什麼？

這裡就是主循環的部分原始碼：
<?php if(have_posts()) : ?><?php while(have_posts()) : the_post(); ?>

<?php endwhile; ?>

<?php endif; ?>

第一，**if(have_posts())** 檢查網誌是否有文章，
第二，**while(have_posts())** 執行 **the_post()** 去導入文章。而 **Else** 是當網誌完全沒有文章的時候執行的。 **while()** 和 **endwhile;** 應該嵌套在 **if()** 和 **else :**之間。 所以 **<?php else : ?>** 應該在 **<?php endwhile; ?>** 之後。

現在你知道什麼是 **else** 了吧，當沒有任何文章或者當找不到任何文章的時候，告訴 WordPress 怎麼處理，讓 WordPress 顯示錯誤訊息 **Not Found**，或者其他任何你想要的東西。我們可以下載任一款免費主題，看一下它的 index.php 檔案怎麼寫的。

在上面的例子中，**Not Found** 錯誤訊息是在 **<?php _e(''); ?>** 之中。如我上一篇所說，這不是必需的，只是為了讓主題可翻譯。

整個訊息和原始碼 **Not Found** 外面有 **<h2>** 和 **</h2>**。這個同樣也不是必需的。你可以簡單使用：

** **
Not Found
** **

但是，給這個錯誤訊息使用上 **<h2>**（子標題）標籤能夠使它更明顯，讓訪問者注意到這個頁面上沒有任何東西。

那麼 ** ** 和 ** ** 用來做什麼的呢？嗯，我們肯定不希望你的錯誤訊息在「茫茫蠻荒之地」之間滯留，對吧？我們用 ** ** 和 ** ** 標籤圍住每篇文章。所以同樣，儘管是錯誤訊息不是真正的文章內容，但是我們其實可以把它當作文章來處理。

第2步：文章 ID

增加 **id="post-<?php the_ID(); ?>"** 到 ** **

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

儲存並重新整理瀏覽器。然後 **查看 > 頁面原始碼**。現在我們會發現現在每篇文章都附加上了一個數字或者說是文章 ID。**the_ID()** 只是導入每篇文章的 ID。

為什麼使用它呢？這是用來自訂個別的文章的面貌。後面，當你使用 style.css 檔案去告訴你的主題文章將看起像怎麼樣。如果透過給每篇文章附加唯一的 **ID**，你就可以針對單獨的一篇文章進行樣式化，使得它和其他文章看起來不一樣。如果沒有 ID，你將沒有辦法透過 **style.css** 檔案使它和其他文章不一樣。

同時把 **class** 和 **id** 賦予同一個 **DIV** 標籤，可以嗎？**DIV** 是標籤，**class** 是一個屬性，**id** 也是是一個屬性。每個標籤能擁有多個屬性，如 **DIV** 就可以同時有 **class** 和 **id**這兩個屬性。（**註釋**：id 是一個 XHTML 屬性。the_ID() 是 PHP 函數。他們是不同的，）

第3步：連結標題

增加 **title="<?php the_title(); ?>"** 到文章的標題連結。

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

儲存並重新整理瀏覽器。然後再去查看原始碼，找到任何**文章的標題連結**，如果文章的標題連結是 **Hello World**，那麼他的左邊應該有 **title="Hello World"**。

title="" 是 **<a>**（連結）標籤的另一個屬性。在雙引號中的是連結的描述。在這裡，每篇文章的標題也是連結的描述。這就是為什麼我們要再次使用 **the_title()** 這個 PHP 函數。

如果不使用 **the_title()** 作為 **title=""**的值，那麼每篇文章標題連結將會有同樣的描述。舉個例子，如果用 **title="Click me"** 取代 **the_title()**，每篇文章標題連結都將會用 **Click me** 作為描述。

返回頁面。把游標移到任何一篇文章標題的連結上，描述訊息將會跳出，這就是剛剛增加的。增加描述到連結是非常有用的，當你其他網站需要掃瞄你的網誌的時候，如 Technorati.com，每次你發表文章的時候，WordPress 通知 Technorati 和其他網站你的網誌已經更新了。Technorati 然後就會來到你的網誌，掃瞄它，並索引得到一個你文章的摘要，這其中會包括你連結標題的描述。

WordPress 主題教學 #5e：文章導航連結

文章導航連結是從零開始建立 WordPress 主題系列教學的第五篇的第五部，在絕大多數的 WordPress 網誌的底部，都會有 下一頁 **(Next Page)** 或者 上一頁 **(Previous Page)** 這樣的導航連結。我們可以透過 WordPress 的模板系統中的 **posts_nav_link()** 這個函數導入這些連結。

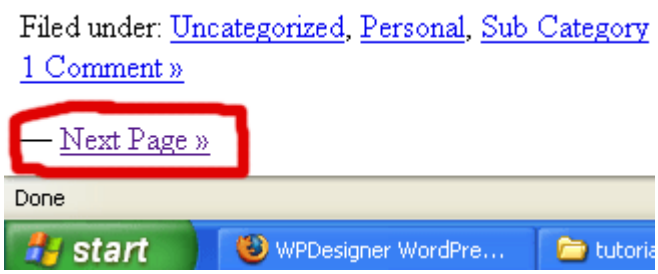
在 **<?php endwhile; ?>** 和 **<?php else : ?>**之間加入如下原始碼：

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

```
        <?php endwhile; ?>

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

        <?php else : ?>
```

** ** - 開始一個名為 **navigation** 的 DIV 標籤。
<?php - 開始 PHP 原始碼
posts_nav_link() - 導入後一頁和前一頁的連結。
; - 停止導入。
?> - 結束 PHP 原始碼
** ** - 結束名為 **navigation** 的 DIV 標籤。

效果如下：



儲存並重新整理，查看後一頁或者前一頁的連結。預設情況下，如果沒有超過10篇文章的話，是不會顯示導航連結。如果沒有超過10篇文章，依然想看到導航連結，登入到管理界面，選擇 **Settings > Reading**，然後把它設定為比所有文章少一篇。如，有6篇文章，就設定為5。

如何自訂化 posts_nav_link()：

和前面 **postmetadata** 課程中介紹的函數一樣，我們也可以給這個函數3個參數，分別給連結的中間，前面和後面的設定字符，如下：

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

第1個參數是顯示在後一頁和前一頁連結的中間。第2個參數顯示在前面。第3個參數顯示在後面。

這裡是一個自訂化 **posts_nav_link()** 的例子：

WordPress 主題教學 #6：側邊欄

側邊欄是從零開始建立 WordPress 主題系列教學的第六篇，這一篇我們主要講解 WordPress 主題的側邊欄，讓你很快掌握它的結構，並能編碼和樣式化它。

在開始側邊欄之前，這是現在 index.php 檔案的樣子。

第1步：建立 id 為 "sidebar" 的 DIV

首先讓我們建立一個名字為 **sidebar** 的 DIV，這樣可以把側邊欄中的所有東西都放入其中。在 **container** 的後面和 **</body>** 標籤的前面輸入以下原始碼：

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

```
</div> closing tag for the box with id named container

<div class="sidebar">

</div>

</body>
</html>
```

第2步：給側邊欄的 DIV 加入無序列表

在新的 **sidebar** 的 DIV 標籤中建立一個新的無序列表。

<ul> - 開始無序列表
**
** - 結束無序列表

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

第3步：給這個無序列表加入原屬

增加一個列表元素（**LI**）到無序列表（**UL**）的中間並把一個子標題加入到這個列表中。

**
<h2><?php _e("Categories"); ?></h2>**

**
**

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

        </li>
</ul>
```

注意加入製表符到**
** 和 **
** 標籤之前為了原始碼縮排。

<li> - 開始列表元素
<h2> - 開始子標題
<?php _e("Categories"); ?> - 輸出字符 **Categories**
</h2> - 結束子標題
**
** - 結束列表項目

儲存 index.php 檔案並重新整理瀏覽器。現在應該可以看到 **Categories** 子標題結構應該這樣：

子標題前面的小圓點指明這個子標題是在一個列表元素中（**LI**）。如果無序列表（**UL**）有兩個列表元素，那麼將有兩個小點。

第4步：加入分類連結列表

在列表項目中加入下面原始碼：

<ul>
<?php wp_list_cats("sort_column=name&optioncount=1&hierarchical=0′); ?>
**
**

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

這裡是上面原始碼的解釋：

<ul> - 開始另一個無序列表
<?php wp_list_cats(); ?> - 導入分類連結列表
**
** - 結束無序列表

儲存並重新整理瀏覽器。下面是分類連結列表的樣子：

category-links.gif

預設的分類是 **Uncategorized**。如果你沒有把文章發佈到多個分類下面，那麼你的列表連結列表應該是只有一個連結 Uncategorized。

更進一步的解釋：

- **sort_column=name** - 把分類按字符順序排列
- **optioncount=1** - 顯示每個分類含有的文章數
- **hierarchial=0** - 不按照層式結構顯示子分類，這就解釋了為什麼子分類連結是列在列表中第一級。
- **&** - 每次增加另一個參數的時候，需在它之前要輸入 **&** 用來把和現有的參數區分開。如 **&** 在 **sort_column** 和 **optioncount**之間。

為什麼不把 **<?php wp_list_cats(); ?>** 放入**
** 和 **
** 標籤中呢？

當我們使用 **wp_list_cats()** 這個函數導入連結列表函數的時候，它會自動附上一組 **
** 和 **
（列表項目）標籤在每個連結的左右。查看頁面原始碼**；可以看到每個連接的周圍都已經有一組列表元素的標籤。

當處理側邊欄，無序列表和列表元素的時候，我們一定記得**規則 #1**：按順序關閉所有標籤。

The right way to close:

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

The wrong way to close:

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

WordPress 主題教學 #6b：頁面連結列表

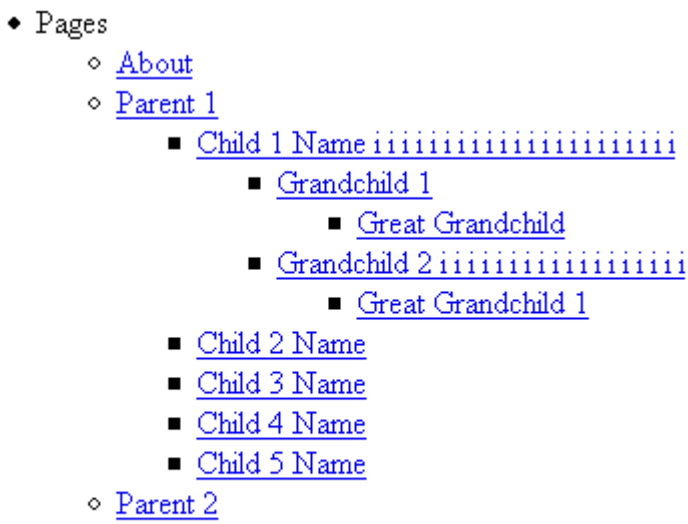
頁面連結列表是從零開始建立 WordPress 主題系列教學的第六篇的第二部分，透過上一篇的學習，現在已經熟悉了側邊欄的結構，接下來我們將繼續修改側邊欄，完成**頁面連結（Page-link）**列表。當完成常規的側邊欄之後，我們將學習如何模組化（widgetize）側邊欄。

在分類連結上面加入以下原始碼：

<?php wp_list_pages(); ?>

```
<ul>

        <?php wp_list_pages(); ?>

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

儲存並重新整理瀏覽器。效果如下所示：



在預設情況下只有一個頁面連結，就是 About 連結。我在我的本機的網誌增加了很多多頁面和子頁面，這樣我就有四級頁面連結。

查看頁面原始碼，我們可以看到 **wp_list_pages()** 產生的完整結構以及原始碼，如下：

```
<li>Pages

    <ul>

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

    </ul>

</li>
```

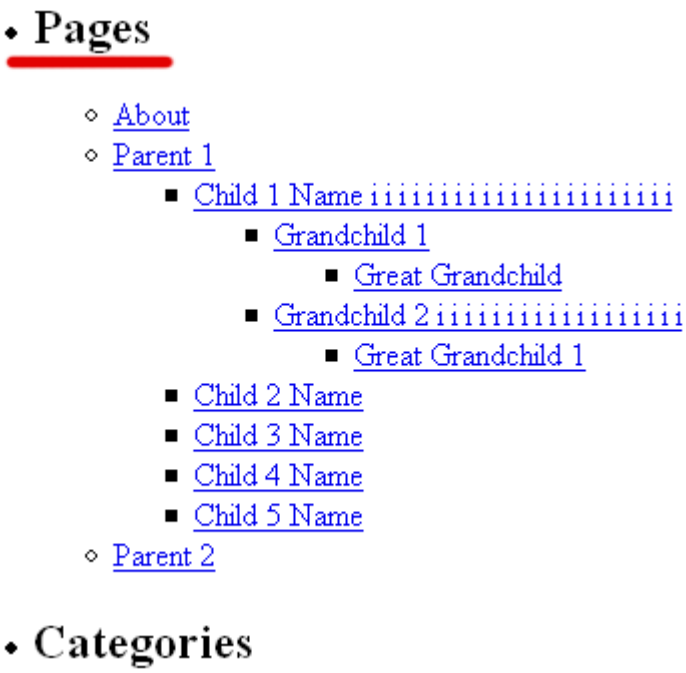
第一，它把所有東西放入列表元素標籤（**LI**），第二，它給列表一個名字，**Pages**。第三，它增加一個無序列表（**UL**）。第四，它把每個連結放入到 **
** 和 **
**標籤之間。

在上面的截圖中，注意到「**Pages**」這個列表標題和「**Categories**」這個分類連結標題的大小不一樣。

如何使它們一致呢？ 加入 **"title_li=<h2>Pages</h2>"** 到 **wp_list_pages()** 作為參數。

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

儲存並重新整理瀏覽器結果如下：



title_li 是一個用來自訂化頁面連結列表的標題的參數。**<h2>Pages</h2>** 是 **title_li** 這個參數的值

進一步自訂化：

在我的例子中，我有四級頁面連結。由於佈局或者設計的原因使得不能在側邊欄處理那麼多級別的連結。為了限制顯示列表的層數，增加了 **depth** 這個參數，並把它設定為 **3**：

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

注意，我加入了 **depth=3&** 而不是僅僅 **depth=3**。這個 **&** 在這兒用於把 **depth** 和 **title_li** 這兩個參數區分開。（如果你只有一個 about 頁面連結，你將不會注意有什麼不同。）

這裡是我的列表的不同之處：（對比這個截圖和上面的截圖。）

- Pages
  - About
  - Parent 1
    - Child 1 Name iiiiiiiiiiiiiiiiiiiii
      - Grandchild 1
      - Grandchild 2 iiiiiiiiiiiiiiiiii
    - Child 2 Name
    - Child 3 Name
    - Child 4 Name
    - Child 5 Name
  - Parent 2

WordPress 主題教學 #6c：存檔和連結列表

存檔和連結列表是從零開始建立 WordPress 主題系列教學的第六篇的第三分，這篇將比較簡單，講解如何導入存檔連結列表和友站連結（blogroll）列表。

第1步 - 增加存檔連結列表。

在側邊欄區域的 **Categories** 列表下面輸入以下原始碼:

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

複製之後檢查下原始碼是否和下面一樣：

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

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

儲存並重新整理瀏覽器。結果如下所示：

發生什麼了？

我們使用了 **wp_get_arhives()** 這個 PHP 函數，並用了 **type** 這個參數以及 **monthly** 作為它的值，這樣就按月導入存檔連結列表。

- **<li>** - 開始列表元素
- **<h2>** - 開始子標題
- **<?php _e("Archives"); ?>** - 子標題文本
- **</h2>** - 結束子標題
- **<ul>** - 開始在存檔連結這個無序列表
- **<?php wp_get_archives("type=monthly"); ?>** - 按月導入存檔列表連結，並把每個連結放入 **
** 和 **
** 標籤中。如果查看原始碼，我們會看到 **wp_get_archives()** 為每個連結產生了列表元素（**LI**）標籤，就像 **wp_list_cats()** 這個函數一樣。
- **
** - 結束在子標題下的無序列表
- **
** - 結束列表元素

第2步：增加友站連結列表

在存檔連結列表下輸入以下原始碼：

<?php get_links_list(); ?>

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

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

儲存並重新整理，結果如下：

預設情況下，我的 blogroll 和你的是沒有什麼不同，這裡是它在原始碼中的樣子：

```
<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>
```

上面的原始碼完全沒有正確的被縮排，因為它們是由函數 **get_links_list()** 產生的，就像上一篇所學的函數 **wp_list_pages()** 產生的原始碼一樣，但是它遵循規則 #1，按正確順序關閉所有的東西。我已經圈出了元素和無序列表的標籤讓你看得更明顯。

WordPress 主題教學 #6d：搜尋框和日曆

搜尋框和日曆是從零開始建立 WordPress 主題系列教學的第六篇的第四部分，儘管這篇的題目是 **搜尋框（Search Form）** 和 **日曆（Calendar）**，但是我同樣也會介紹 **元資料（Metadata）（Meta）**。這一篇我們會結束常規的側邊欄，然後將在下一篇將介紹如何模組化（widgetize）化側邊欄。

第1步：增加搜尋框

建立一個新檔案，然後把該空白檔案儲存為 **searchform.php**（當然是和 index.php 在同一個檔案夾下）。把 searchform.txt 中的內容複製到 searchform.php。

在 index.php 檔案，在側邊欄的最頂部輸入以下原始碼：

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

```
<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>
	</li>

	<?php get_links_list(); ?>

</ul>
```

儲存並重新整理瀏覽器，結果如下：

剛才發什麼了呢？

- **<li id="search">** - 開始一個名字為 **search** 的列表元素，給它一個 ID，這樣就能夠以後樣式化它。
- **include()** - 導入任何你想導入的檔案。這和使用 WordPress 模板函數去導入模板檔案是不同的，因為 **include()** 只是簡單導入已經存在的檔案。這裡是導入在 searhform.php 檔案中的原始碼。被導入的訊息應該在一個網誌上基本不會被改變的。
- **TEMPLATEPATH** - 主題檔案夾的位置，這裡是：**wp-content/themes/tutorial**
- **'/searchform.php'** - 檔案名：**/searchform.php**
- 在 TEMPLATEPATH 和 "/searchform.php" 中間的點把它們連接起來，所以最終得到：**wp-content/themes/tutorial/searchform.php**
- **
** - 結束列表元素

注意，搜尋框不像分類，歸檔，頁面或者 Blogroll 一樣有子標題。當然如果你願意，也可以給它一個子標題。

第2步：增加日曆

在搜尋框或者頁面連結列表下面輸入以下原始碼：

```
<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>
```

儲存並重新整理瀏覽器，結果如下：

- Search

- **Calendar**

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	

« Oct

發生了什麼？

- **<li id="calendar">** - 開始一個 ID 為 「Calendar」 的列表元素
- **<h2>** - 開始一個子標題
- **<?php _e("Calendar"); ?>** - 輸出 **Calendar** 這個詞
- **</h2>** - 關閉子標題
- **get_calendar()** - 使用 get_calendar() 這個 WP 函數導入日曆
- **
** - 結束列表元素

這樣日曆就完成了

第3步：增加元資料（Metadata）

在 **get_links_list()** 函數下輸入以下原始碼：

```
<li><h2><?php _e("Meta"); ?></h2>
<ul>
<?php wp_register(); ?>
```

**
<?php wp_loginout(); ?>
**
<?php wp_meta(); ?>
**
**
**
**

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

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

儲存並重新整理瀏覽器，結果如下：

（如果你沒有登入 WordPress）

- Meta
  - Register
  - Login

（如果你已經登入）

- Meta
  - Site Admin
  - Logout

那麼這是怎麼回事呢？

你開始一個列表元素（**LI**），跟著是一個子標題（**H2**）Meta。在子標題下，嵌入了一個無序列表（**UL**）。最後把每個連結都放入了列表元素中（**LI**）。

wp_register() 這個函數能產生一組 **
** 和 **
** 標籤，如果你沒有登陸，它顯示**註冊（Register）**連結，如果登入了，它顯示的是 **網站管理（Site Admin）**的連結。**wp_loginout()** 不會產生列表元素標籤，所以需要我們手工輸入列表元素標籤，當你沒有登入的時候，得到的是 登入（**Login**） 的連結，當已經登入的時候，得到的是登出（**Logout**）連結。

到目前為止，**wp_meta()** 沒有做任何事情，他在網頁上和原始碼中都不會產生東西，現在不要考慮 **wp_meta()**，實際上你已經在使用它了。

到此為止，我們已經完成 Meta 並最終完成了常規的側邊欄。

WordPress 主題教學 #6e：模組化側邊欄

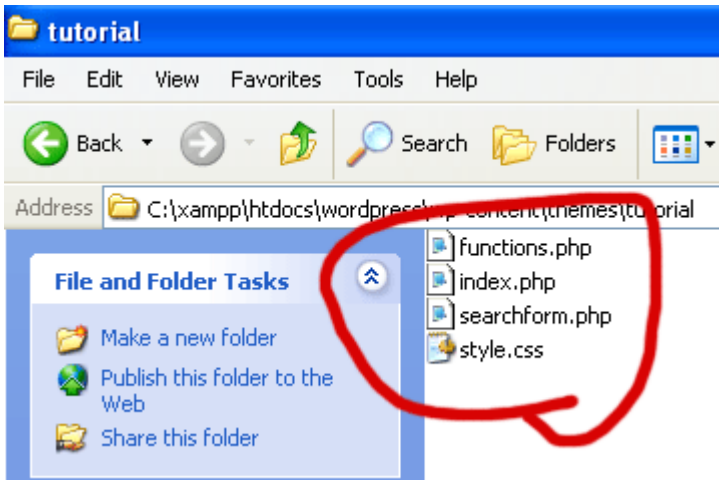
模組化側邊欄是從零開始建立 WordPress 主題系列教學的第六篇的第五部分，一個支援 Widget 的側邊欄或者說是模組化（widgetized）的側邊欄幾乎是 WordPress 主題的標準。

首先，什麼是模組化（widgetizing）呢？簡單的說，模組化就是能夠透過拖曳就能夠整理側邊欄的模塊。比如我們需要更改分類和存檔的位置，只需要簡單把分類和存檔列表拖到它們的位置即可，根本不用去修改側邊欄的原始碼。

第1步：建立 functions.php 檔案

打開記事本，然後把空白檔案儲存為 **functions.php**。把 functions.txt 檔案中所有的內容複製到 **functions.php** 中。

回顧一下，現在在「**tutorial**」主題檔案夾下應該有4個檔案。



第2步：模組化側邊欄

直接在側邊欄的第一個 **<ul>** 標籤輸入以下原始碼：

<?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(); ?>
```

直接在 **
** 標籤之前輸入以下原始碼：

<?php endif; ?>

```
<?php endif; ?>

</ul>

</div>

</body>
</html>
```

儲存 index.php 檔案，然後我們到 WordPress 後台 => 外觀 => Widget 就可以把 Widget 拖到側邊欄了。

WordPress 主題教學 #7：尾部

尾部（footer）是從零開始建立 WordPress 主題系列教學的第七篇，這篇教學將會很簡單，去只要在側邊欄下增加個 DIV 標籤，然後輸入一些版權訊息。其實你完全可以不用我說明就能自己去做，可以先自己嘗試下，然後返回這裡再仔細檢查下。

第1步：增加個 DIV 標籤

在側邊欄的 DIV 標籤下輸入以下原始碼：

** **
** **

```
</div>end of Sidebar

<div id="footer">

</div>

</body>
</html>
```

第2步：加入版權訊息

把尾部的文本放入段落標籤中，你可以輸入任何你想要的東西，這裡是我的：

** **
Copyright © 2007 <?php bloginfo("name"); ?>
** **

```
<div id="footer">
<p>
Copyright &#169; 2007 <?php bloginfo('name'); ?>
</p>
</div>
```

儲存並重新整理瀏覽器，結果如下：

- Archives
  - October 2006

Copyright © 2007 Demo Theme Development

© 用於顯示版權符號，還記得在 header 的時候使用的 **bloginfo()** 函數嗎？這裡再次使用，「**name**」是用於導入網誌標題，而「**url**」導入網誌的網址。

如果你想你的網誌標題成為一個連結，查下頭部就知道怎麼做了。

WordPress 主題教學 #8：驗證 XHTML

驗證 XHTML 是從零開始建立 WordPress 主題系列教學的第八篇。在開始學習 CSS 並修改 style.css 檔案之前，我們需要學習如何驗證原始碼，簡單說，驗證（Validate/

Validating/Validation）就是檢查下原始碼有沒有錯誤，而驗證又分為：XHTML Validator 和 CSS Validator。這篇我們學到 XHTML 驗證器。

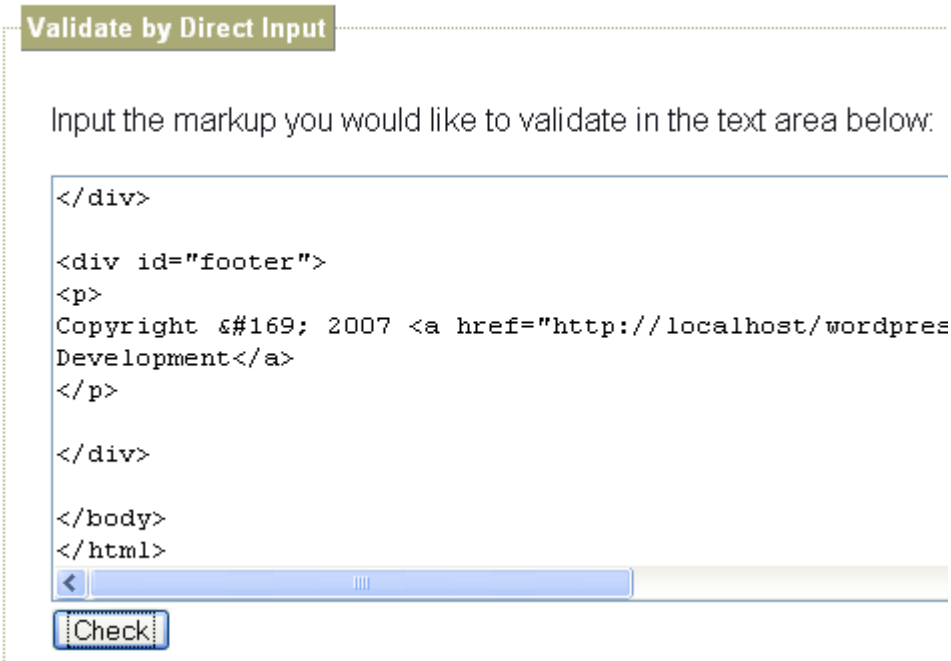
首先打開 **Xampp Control** 和瀏覽器，並進入 **http://localhost/wordpress**。

然後**查看 > 頁面原始碼**。

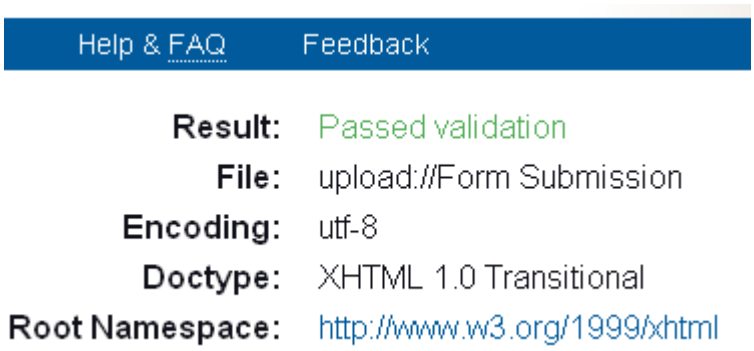
全選並所有的原始碼。

然後到 XHTML Validator。

把剛才複製的原始碼貼上到 **Validate by Direct Input** 框中。



點選 **Check** 之後，驗證器會就會檢查原始碼，然後把檢測結果回應給我們。如果回應回來的結果是綠色的，那麼原始碼沒有錯誤。

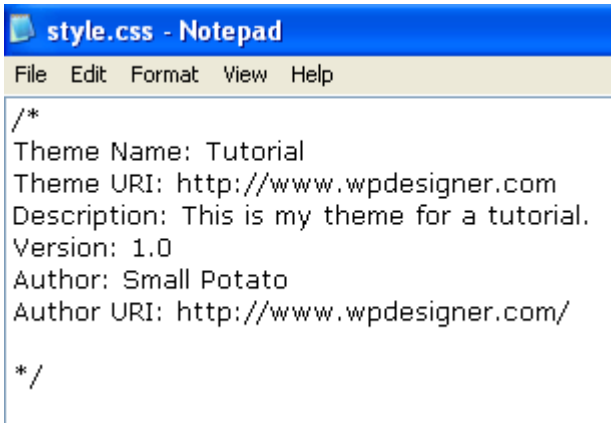


如果有錯誤，則根據其提示進行修改。

WordPress 主題教學 #9：Style.css 和 CSS 介紹

Style.css 和 CSS 介紹是從零開始建立 WordPress 主題系列教學的第九篇，學習 CSS 最好的方法就是去使用它，不像 XHTML 和 PHP 需要接觸模板的核心檔案，同樣不要需要理解任何基本概念，只要去用它，透過試用和修正錯誤是可以讓你快速學會。

我們現在已經在 style.css 檔案有些內容，讓我們先來看看這部分內容是幹什麼的。



- 第一行顯而易見就是主題的名字。
- 第二行是這個主題的網址，如果你的主題只是自用的而不準備發佈的話，那就不用管它了。
- 第三行是主題的描述。
- 第四行是版本號，這是非常重要的，特別是當你公開發佈你主題新版本的時候。
- 第五和第六行分別是主題作者的名字和網頁。

在主題訊息兩邊的 /* 和 */ 符號是為了讓主題的訊息不影響該檔案的其他內容，這是 CSS 的註釋。當輸入 CSS 原始碼去樣式化你的網頁的時候，你可能想在這裡增加些註釋使得能夠在以後更清楚知道這部分是幹什麼的。顯然我們並不想你的註釋影響實際的原始碼，所以可以使用 /* 和 */ 這一對符號使得註釋不被解釋。

下面是處理後的主題訊息

第1步：打開 style.css 檔案

- 打開 Xampp，主題檔案夾，FireFox，IE 瀏覽器和 style.css 檔案。
- 在兩個瀏覽器的網址列都輸入：http://localhost/wordpress

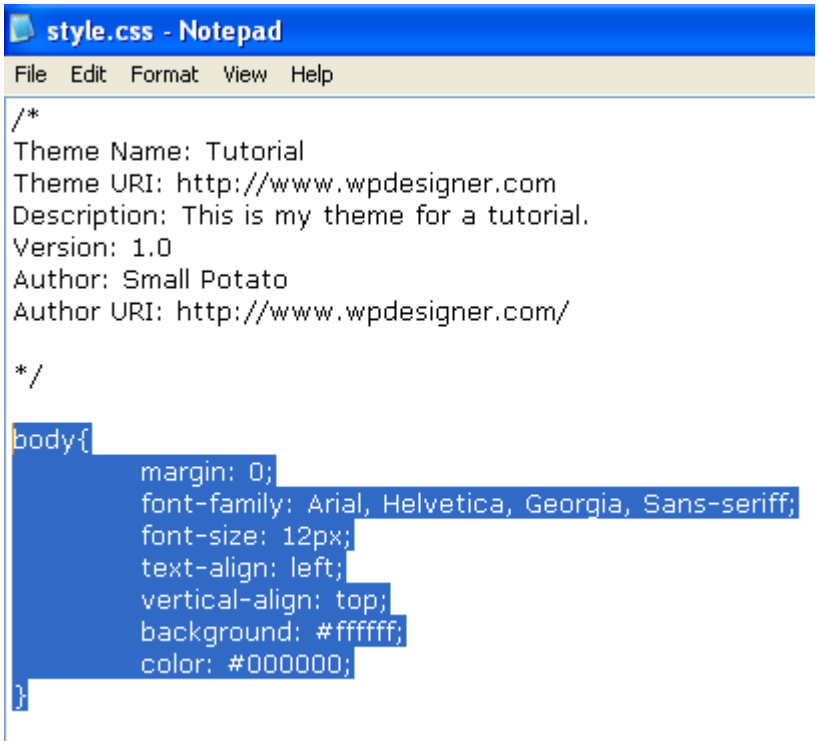
從這裡開始，我們需要同時在 FireFox 和 IE 上測試主題，不同的瀏覽器對 CSS 的原始碼解釋是不同的。如果能夠在盡可能多的瀏覽器器上和盡可能多的操作系統上測試你的主題是最好的（Safari，Opera，Linux，Netscape 等等）。如果你和我一樣懶，那就只在 FireFox 和 IE 上測試你的主題吧。

第2步：加入 CSS 原始碼

在 style.css 檔案中輸入以下原始碼：

```
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 一樣，透過製表符增加縮排來組織原始碼：



儲存 style.css 檔案並**重新整理** 兩個瀏覽器 **Firefox** 和 **Internet Explorer** 查看變化。

把 **body{ }** 看作一個標籤，然後它裡面所有的東西看作屬性和值，和處理 XHTML 時一樣。**{** 是開始符，**}** 是結束符。在 **{** 和 **}** 之間，**冒號意思是開始而分號意思是結束**。（我在涉及到 XHTML，PHP，CSS的時候都使用標籤，屬性和值這些術語是為了保持簡單，實際上 PHP 和 CSS 有不同術語。如參數（parameters），選擇器（selector）和屬性（property）。）

在我們繼續之前，我需要解釋下為什麼使用 **body{ }**（CSS 選擇器），是因為你是在樣式化網頁的絕大基本部分（或者說是總體部分），**<body>** 標籤。你不會樣式 <head> 因為這個標籤沒有東西需要樣式化。你網頁上展示的絕大部分的東西是在 **<body>** 和 **</body>** 標籤之間。

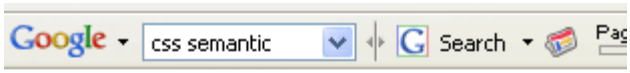
然後，在後面你會樣式化 ID 為 **header** 的 DIV 標籤。

進一步的解釋：

margin: 0; 處理 body 標籤的預設的頁邊空白，如果你要頁邊空白或者更大的頁面空白，把 0 改成 10px，20px 或者其他。PX 意思是像素。每個像素使你電腦畫面的一個點。當你的頁邊空白是 0 的話，就不需要後面跟上 px。

在下面的圖片中，紅色高亮的區域就是應用於 body 標籤的預設的頁邊空白。

當給其樣式化為 **margin: 0;**，下面是沒有頁邊空白的相同頁面：



Demo Theme Development

Just another WordPress weblog

Hello World

font-family: Arial, Helvetica, Georgia, Sans-serif; 為你的網頁或者網站選擇使用哪種字型。這些字型中的第一個，**Arial** 是可替換的，如果你的用戶沒有在他們的電腦上安裝 **Arial** 這種字型，**style.css** 檔案就會尋找 **Helvetica**，然後是 **Georgia**，再接著是 **Sans-serif**。你可以在字型檔案夾（**我的電腦 > 控制台 > Windows**下面）找到你的字型列表。

font-size: 12px; 顯而易見是字型大小。可以把它改大或改小以查看變化。

text-align: left; 讓你的文本向左對齊。把它改成 text-align: right; 去查看不同之處。

vertical-align: top; 使得所有的東西從上面開始。如果是中部或底部排行你的 body 標籤，所有東西將會向下推。

background: #ffffff; 意思是白色背景。#ffffff 是白色十六進制原始碼。#000000 是黑色十六進制原始碼。

color: #000000; 意思是文本顏色是黑色。

如果你想向前更進一步或者自己學習 CSS，最好的地方是 w3schools.com

WordPress 主題教學 #10：十六進制顏色原始碼和樣式化連結

十六進制顏色原始碼和樣式化連結是從零開始建立 WordPress 主題系列教學的第十篇。這篇繼續昨天介紹 CSS 的課程，我們今天將介紹如何著色和**十六進制顏色原始碼**。

顏色屬性，跟著的是一個十六進制原始碼，是用於給文本上色。如 **body { color: #000000;}** 意思是你頁面 body 內所有文本將是黑色的。

背景顏色屬性，跟著的是一個十六進制原始碼，是給除背景上色。如 **body{ background: #ffffff; }** 意思是為 body 上白色背景。

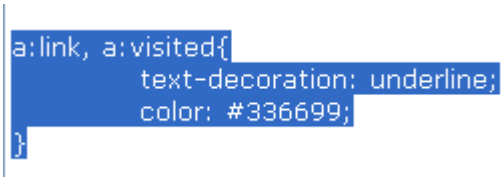
十六進制原始碼

- 每個十六進制原始碼前都有 # 號，然後跟著六位數字。這些數字的範圍是從 **#ffffff**（白色）到 **#000000**（黑色）。
- #ffffff， #eeeeee， #dddddd， #cccccc， #bbbbbb， #aaaaaa， #999999， #888888， #777777， #666666， #555555， #444444， #333333， #222222， #111111
- 前兩位表示紅色，第三和第四代表綠色，而最後兩位代表藍色。**#ff0000** 是紅色（red）。**#550000** 是暗紅色（dark red）。 **#220000** 是更黑色的紅色（darker red）。 **#00ff00** 是綠色（green）。 **#0000ff** 是藍色（blue）。那麼哪個十六進制原始碼是黃色呢？ **#ffff00** 就是黃色（yellow）。 **#ff00ff** 是紫色（violet）。

第1步：加入連結顏色

在 **body{ }** 選擇器下輸入以下原始碼：

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



- **這些原始碼是幹嗎用的？**，給所有的連結都加上下劃線的（**text-decoration: underline;**）和上了藍色（**color: #336699;**）。這是不是純正的藍色，但它確實是藍色是因為最後兩個數字（代表藍色）是最高值的數字。
- **a:link** 用於樣式化連結。當你想把一個詞轉變為連結的時候，用什麼實現呢？使用 **<a>** 和 **** 這對標籤，因此樣式化連結就是樣式化 **a:link**。
- **a:visited** 用於樣式化已經訪問過的連結。
- 另外一種輸入方式：
  a:link{
  text-decoration: underline;
  color: #336699;
  }
  和

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

- 當給**a:link** 和 **a:visited**這兩個選擇器應用相同的 **text-decoration: underline;** 和 **color: #336699;** 這兩個屬性的時候。可以把它麼你放在一起，使用**逗號**來區分。

第2步：加入滑鼠移至連結顏色

在 **a:link, a:visited{ }** 下輸入以下原始碼：

a:hover{
text-decoration: none;
}

這些原始碼是幹嗎用的呀？ 當把指針移到連結上面時候下劃線消失。

如果不想在預設情況下有下劃線而是在當把指針移到連結上面的時候才出現下劃線，那麼就在 **a:link** 和 **a:hover** 之間交換下 **text-decoration:** 的值。

如果你想更改你滑鼠移至連結時的顏色，那麼就增加 **color:** 和任何你想要的十六進制原始碼，如：

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

WordPress 主題教學 #11：寬度和佈局

寬度和佈局是從零開始建立 WordPress 主題系列教學的第十一篇，這篇將介紹如何設定每個 DIV 的寬度和佈局排版，並且也會展示如何讓主題顯示正確，並同時在 Firefox 和 IE 下兼容，顯示一致。

在我們開始之前，打開 **Xampp Control**，**主題檔案夾**，**Firefox** 瀏覽器，**IE** 瀏覽器，**index.php**和**style.css**這兩個檔案。

第1步：設定頁面總體寬度

現在我們首先要確定的是主題的總體寬度。我們使用 750px；主題的大小取決於網誌絕大多數訪問者的畫面分辨率。需要避免的是使用過大寬度的的主題，如果網誌的讀者都大多數使用 800px × 600px 的畫面，這樣的話，如果是使用 900px 寬的主題將會有 100多像素超出他們的畫面，顯然這是對用戶很不友善的。

不管怎樣，我們怎麼樣把主題的總體寬度設定為 750px 呢？

我們需要把現在主題中的所有的東西（header，container，sidebar 和 footer）放入一個 750px 的 DIV 標籤中。

在 **<body>** 之後增加：** **

在 **</body>** 之前增加：** **

在 **style.css** 檔案中輸入以下原始碼：

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

在 CSS，**#** 號是透過 **id** 來定位頁面中的元素，而點號是透過 **class** 來定位頁面中的元素，如果你的原始碼是 ** **，那麼就應該用 **.wrapper** 來替代 **#wrapper** 去定位 **wrapper** 這個 DIV 標籤。

同時儲存 index.php 和 style.css 檔案。重新整理 Firefox 和 IE 瀏覽器（按 F5）查看所做的改動。

詳細解釋：

- **margin: 0 auto 0 auto;** 意思是（注意順序）：**0上頁邊空白，自動右頁面空白，0下頁邊空白和自動左頁面空白。**從現在開始，記得設定左右頁邊空白為自動將使得置中對齊。
- **width: 750px;** 顯而易見是 750 像素。
- **text-align: left;** 是讓 **wrapper DIV** 中的文本向左對齊因為我們等下要要將 **body{ text-align: left;}** 改成 **text-align: center;**

第2步：自動頁面置中

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

為什麼？（我假設你使用的是 Firefox 和 Internet Explorer 6）。你的佈局可能你看起來是正確的，但對於使用早前版本的 IE 用戶可能不正確。還記得設定左邊和右邊的頁邊空白為自動是置中嗎？但是這並不是對所有的 IE 都適用，所以 **body{ text-align: center; }** 是讓 **wrapper** DIV 置中在舊版本 IE 的一種解決方案。

（隨便說一下，在 Firefox 和 IE 中文本大小是不同的，我們稍後解決。）

第3步：設定 header 寬度和佈局

讓 **Header** 浮到左邊並且設定它的寬度為 750px：

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

第4步：設定 Container 寬度和佈局

讓 **Container** 浮到左邊並且寬度為 500px：

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

第5步：設定 Sidebar 寬度和佈局

讓 **Sidebar** 浮到左邊，寬度為240px，並且給它灰色的背景：

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

#ffffff 是白色而**background: #eeeeee;** 是非常淺的灰色。我們給側邊欄增加一個背景顏色只是去查看當增加剩下的 10 像素之後的不同之處。

第6步：設定 Footer 的寬度和佈局

讓 **Footer** 浮到左邊，左右兩邊都沒有東西，並且寬度為：750px：

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

Header 和 **Footer** 的樣式有什麼區別呢？答案是 **footer{}** 中有 **clear: both;**。它在那兒使得 Footer 不能和它上面的東西（如 Sidebar 或者 Container）連接起來。

儲存並重新整理瀏覽器。

第7步：給側邊欄增加其餘的 10 像素

給側邊欄增加其餘的 10 像素的頁邊空白。現在側邊欄的 CSS 應該是：

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

儲存並重新整理瀏覽器去查看 10 像素的空白增加到側邊欄的左邊了。

margin: 0 0 0 10px; 實際的意思是：上邊空白為0，右邊空白為0，底部空白為0，左邊空白為10像素。當大小為0的時候，**px** 單位不是必需的。

第8步（額外的步驟）：修正 IE 的雙倍頁邊距 bug

Internet Explorer 有個雙倍頁邊距的 bug，這樣在 IE 下，我們的頁面距就是 20像素，20像素的頁邊距可能會破壞佈局並把側邊欄擠到頁面的底部，因為一個20像素的

頁邊距使得 Container 和 Sidebar 的寬度之和為 760px 而不是 750px。為了解決這個問題，增加 **display: inline;** 到側邊欄。現在你的側邊欄應該是：

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

這裡是現在的 index 和 style 檔案的內容。

WordPress 主題教學 #12：文章樣式化和其他雜項

文章樣式化和其他雜項是從零開始建立 WordPress 主題系列教學的第十二篇，這篇主要講解如何樣式文章，這篇不需要 index.php，

打開**Xampp Control**，**theme 檔案夾**，**Firefox**，**Internet Explorer** 和 **style.css** 檔案。

第1步：Reset CSS

在 **style.css** 檔案中的 **body{}** 上面輸入以下原始碼來處理大部分頁邊空白和填充：

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

- 這裡我們使用的是 **margin: 0;** 而不是 **margin: 0 0 0 0;**。因為所有的值都是一樣的話，只用一個數字就夠了，對於填充的設定也是一樣的。
- 儲存，重新整理 Firefox 和 IE。接下來我們可以增加空白和填充到需要的地方。

第2步：樣式化 H1 標題

在 **body{}** 之後輸入以下原始碼：

```
h1{
font-family: Georgia, Sans-serif;
```

font-size: 24px;
padding: 0 0 10px 0;
}

- **font-family: Georgia, Sans-seriff;** 把 H1 標題的字型從 Arial 改成 Georgia。如果沒有 Georgia，網頁就會尋找 Sans-serif；
- **font-size: 24px;** 我們在 **body{}** 中把字型設定為 **12px**，**H1** 和 **H2** 標籤是不會遵守的。這就是因為標題標籤遵循他們自己的規則。為了控制他們，我們需要特別的去樣式化它們。
- **padding: 0 0 10px 0;** 意思是 10 像素的底部填充。這是為了在網誌的標題和描述之間增加空間。

儲存，重新整理，結果如下：

第3步：樣式化文章

在 **#container{}**下面輸入以下原始碼：（可以在輸入每塊原始碼之後，儲存並重新整理去查看其中的變化。）

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

（給每個 class 名字為 **post** 的 DIV 增加 10 像素的頂部和底部空白。）

.post h2{
font-family: Georgia, Sans-serif;
font-size: 18px;
}

（.post h2 不是一般的 CSS 規則。他是特別樣式化在 class 名為 post 的 DIV 中的 H2 子標題。在側邊欄中的 H2 子標題就不受影響。）

.entry{
line-height: 18px;
}

（設定 entry DIV 中行距。）

第4步：設定文章段落填充

在 **a:hover{}** 下輸入以下原始碼：

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

（給每個段落標籤增加 10 像素的頂部填充。）

第5步：樣式化文章雜項

在 **.entry{}** 下面輸入：

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

對於 **postmetadata** 這個段落便簽，給它增加一個灰色的邊框和10像素頂部空白。

border-top 意思是僅僅頂部邊框 **border-left** 意思是僅僅左邊邊框，等等。 如果只是單獨的 **border**，沒有 **-top**，**-right**，**-bottom** 或者 **-left** 則意味著所有的邊框。如 **border: 1px solid #ccc;** 意思為所有的四邊都有1像素的灰色的邊框。

第6步：樣式化導航列

在 **p.postmetadata{}** 下輸入：

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

對於 **Next page** 和 **Previous page** 連結外面的的 **navigation** DIV 標籤，我們

- 增加了一個10像素的頂部填充。
- 把字型大小改成14像素。
- 把字型改成粗體。
- 把行高增加到18像素。

WordPress 主題教學 #13：樣式化側邊欄

樣式化側邊欄是從零開始建立 WordPress 主題系列教學的第十三篇，這篇主要講解如何樣式化側邊欄裡面的所有元素，在對側邊欄樣式化之後，這系列教學就將差不多結束了。

打開 XAMPP，主題檔案夾，Firefox，IE 和 style.css 檔案。

第1步：樣式化側邊欄的無序列表

在 **.sidebar{}** 下輸入：

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

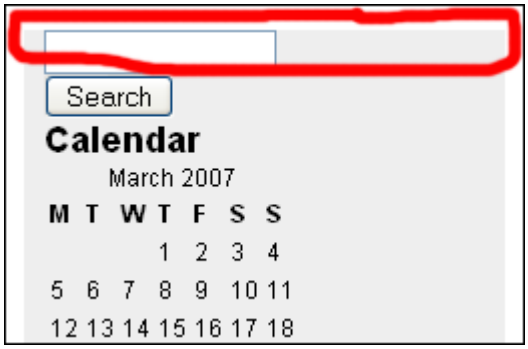
現在已經為側邊欄樣式化父級無序列表（UL）標籤。所有的子 UL 或者內嵌的 UL 將會擊繼承同樣的樣式。在這裡，它是無列表樣式，零空白和10像素的填充。

如下所示：

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

第二級的（或內嵌的）UL 繼承了第一級 UL 的樣式。如果你給了第一級 UL 應用了邊框，第二級的 UL 同樣也會有個邊框。

儲存並重新整理就可以看到列表項目現在已經沒有前面的圓點了。



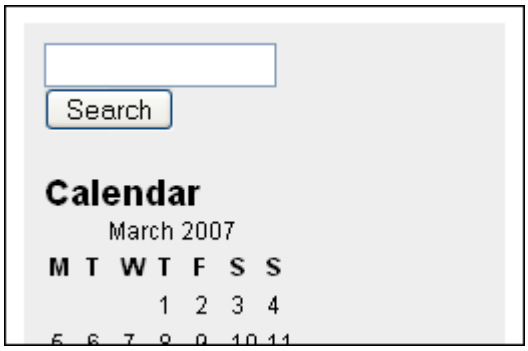
注意下你是如何增加頂部和底部填充的。

第2步：給 LI 加入填充

在 **.sidebar ul{}** 下輸入：

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

這是現在的填充：



在進行這步之前，**搜尋框**和**日曆**之間以及**日曆**和**頁面**之間是沒有空間，如何給這些模塊之間加入空間呢，我們需要給 **.sidebar ul li{}** 加入的10像素頂部和底部填充。**為什麼不在第一個地方的 UL 標籤增加10像素的填充呢？**這樣的話將會有20像素的頂部填充和20像素的底部填充。如果你還是不明白，那麼就去給 **.sidebar ul{}** 增加頂部和底部填充，就會看到問題的所在了。

第3步：樣式化側邊欄下的子標題

在 **.sidebar ul li{}** 下輸入：

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

font-size: 14px;
}

還記得我們已經樣式化了在 .post{} 下的子標題，但是這個是不會對側邊欄的子標題起作用的，因為前面我們僅僅樣式化在 .post{} 下的子標題？現在我們是在樣式化側邊欄下的子標題，結果如下：

這就是我的頁面連結的樣子。可能預設安裝下的 WordPrss 只有一個連結：**About**。我的離線 WordPress 增加了多重頁面連結是為了測試最低級別的連結看起的樣子，注意到我已圈出在底部有不必要額外的填充，這是一個非常好的關於樣式繼承的例子。這裡不是10像素而是20。

因為你給 **.sidebar ul li{}** 增加了填充，為了解決這個問題，直行第4步。

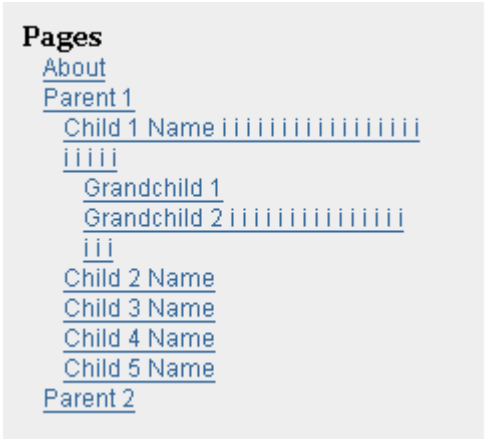
第4步：清除子 UL 下的 LI 填充

在 **.sidebar ul li h2{}** 下輸入：

.sidebar ul ul li{
padding: 0;
}

在 **.sidebar ul ul li{}** 中連續的 **UL** 指明了我們是在定義第二級別的 **LI**s。再說一次，當所有的值都為 0 的時候，你不需要 px 這個後綴。

結果如下：



行距太近了，所以我們把行高改成 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，搜尋框下有多出了額外的空白，在下面增加 form：

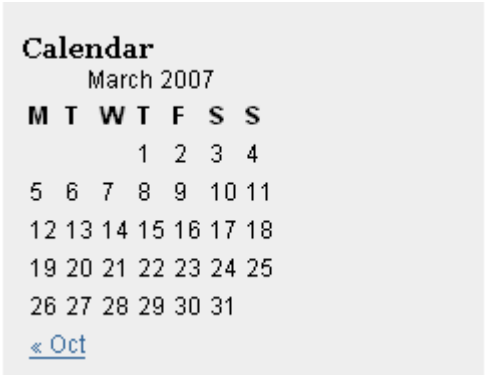
```
body, h1, h2, h3, h4, h5, h6, address, blockquote, dd, dl, hr, p{
margin: 0;
padding: 0;
}
```

改成：

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

第5步（可選的）：擴充日曆寬度到整個側邊欄

執行這一步，如果你想讓你的日曆的資料能夠擴充並覆蓋整個側邊欄的寬度。當前你的日曆應該是這樣的：



為了樣式化日曆，找出在裡面的標籤和這個便簽的名字或者 id。 **查看 > 頁面原始碼或者原始碼**，側邊欄是在底部，所以到原始碼的底部找到 Calendar。

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

現在我們知道日曆是在一個 **TABLE** 標籤中並以 **wp-calendar** 作為 **id**。那麼如何在 **style.css** 檔案中鎖定 **wp-calendar table** 呢？

答案是 **table#wp-calendar{}**。為什麼？早前，你學了使用 # 號當樣式化使用 **id** 而不是 **class** 命名的 DIV。在這裡，是 **table** 而不是 **DIV**，跟著是 **id** 的值，**wp-calendar**。

如果僅僅 #wp-calendar{} 也是可以的因為它是唯一的而且 WordPress 不會使用 #wp-calendar 給別的標籤。但是你應該試著特定化當能夠的時候。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{}**. 因為列表項目（LI）包含日曆子標題和一個 **id** 被命名為 calendar 的日曆表格。

現在你知道可以使用什麼，如何怎麼擴充 table，給表格加上 **width: 100%;** 。

在 **.sidebar ul ul li{}** 下輸入：

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

儲存和重新整理，結果如下：

width: 100%; 因為你想日曆表格適應到側邊欄的寬度，無論你把側邊欄改成多少像素。

可能這樣看起來並不好，但是我相信你已經知道如何改進。日曆需要更多的樣式去看起來更好。**技巧**：再次查看原始碼，找出在 **TABLE** 下的哪個標籤你可以樣式化。

WordPress 主題教學 #14：底部和拆分 Index

底部和拆分 Index是從零開始建立 WordPress 主題系列教學的第十四篇，這篇我們完成對主題的樣式化和開始把 **index.php** 檔案分成多個小檔案。在這篇中，首先要對 **style.css** 檔案進行修改，然後把 **index.php** 分成一些新的檔案。

打開 XAMPP，主題檔案夾，Firefox，IE，index.php 和 style.css。

第1步：樣式化 footer

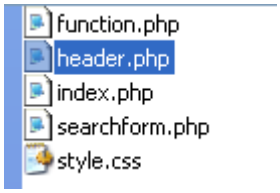
給 **footer** DIV 增加 **10px** 頂部填充。你還記得如何增加填充？這次我不提供原始碼。

第2步：設定 footer P 的行距

給 footer 裡的所有的 **P** 標籤 **18px** 行距。那是 **#footer p{}**.。（今天關於 CSS 的就這麼多。）

第3步：header.php

- 建立一個新檔案，把它命名為 **header.php**。
- 在 **index.php** 檔案中，把 **header** DIV 及以上所有東西都複製到 **header.php** 檔案中。



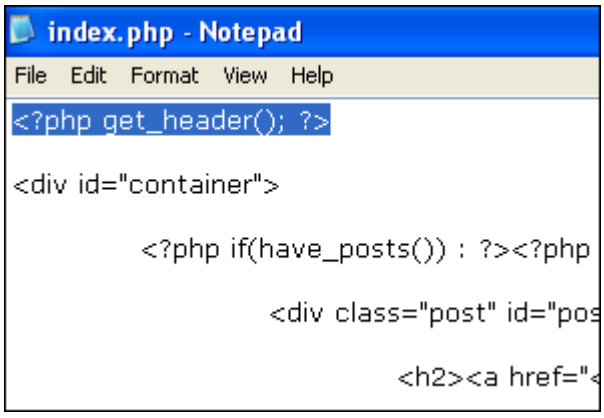

```
        <?php wp_get_archives('type
        <?php //comments_popup_sc
        <?php wp_head(); ?>
</head>
<body><div id="wrapper">

<div id="header">

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

</div>

<div id="container">
```

這是我的 header.php 檔案。不要從我的這裡複製，從你自己的 **index.php** 檔案複製。

第4步：在 index.php 中導入 header.php

為了使所有從 **index.php** 中複製的內容依然在 **index.php** 檔案中，輸入以下原始碼：

<?php get_header(); ?>



這是個 WordPress 主題系統特別用來導入 **header.php** 檔案的函數，而不用使用 PHP 的函數：**<?php include (TEMPLATEPATH . "/header.php"); ?>**.

儲存並重新整理瀏覽器，你應該看到沒有變化。如果你的改變破壞了主題，那麼肯定有錯誤。

第4步：sidebar.php

- 和第4步一樣，更多相同的事情。這次，建立 **sidebar.php** 檔案。
- 把 **index.php** 檔案中的 **Sidebar** DIV 從開始到結尾都複製到 **sidebar.php** 檔案中。
- 那麼，在 **index.php** 檔案，將其取代為：**<?php get_sidebar(); ?>**.
- 儲存並重新整理瀏覽器，再一次，你應該看到沒有變化。
- 這是我的 sidebar.php 檔案。

```
        <?php endif; ?>

</div>

<?php get_sidebar(); ?>

<div id="footer">
<p>
Copyright &#169; 2007 <a href="
</p>
</div>

</div></body>
</html>
```

第5步：footer.php

- 為 footer.php 重複上面的步驟。
- 這是我的 footer.php 檔案。

```
        <?php endif; ?>

</div>

<?php get_sidebar(); ?>

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

教學回顧

- 建立了三個新檔案：**header.php**，**sidebar.php** 和 **footer.php**。
- 使用了三個新的函數：**get_header()**，**get_sidebar()** 和 **get_footer()**。
- 下面是這節課結束之後的檔案：index，style，header，sidebar，footer。

WordPress 主題教學 #15：子模板檔案

子模板檔案是從零開始建立 WordPress 主題系列教學的第十五篇，這篇將和像上一篇建立 header.php，sidebar.php 和 footer.php 這些模板檔案一樣建立更多的子模板檔案。

現在 index.php 檔案已被拆分，這一切都變得更簡單。

第1步：archive.php

在做這步之前，查看你的側邊欄，點選其中的一個存檔連結，結果的頁面是不是和首頁沒有什麼不同？

- 建立一個新檔案：archive.php
- 把 index.php 中所有東西複製到 archive.php
- 儲存 archive.php
- 在 archive.php 檔案，把 the_content 改成 the_excerpt。
- 再次儲存 archive.php 檔案

透過建立一個 archive.php 檔案並把它改成和 index.php 不一樣，這就是**自訂化存檔頁面的**外觀。

現在如果你重新整理你的存檔頁面，它將只顯示摘要而不是全文的文章。

為什麼你想這麼做呢？ -- 防止 Google 以為重複內容懲罰你的網誌，如果一個存檔頁面和首頁顯示相同的內容，那就是重複的內容。

如果是私人的網誌呢？那麼就沒有必要去區分首頁和存檔頁面。但這並不是說摘要對私人網誌沒有用。

同樣 -- 預設你的**類別頁面**將使用 archive.php 顯示內容，如果你沒有 archive.php 檔案，類別頁面將使用 index.php 顯示內容。

如果你想**類別頁面**和**首頁**和**存檔頁面**看起來不一樣，那麼新增一個 category.php 檔案並自訂化它。

第2步：search.php

- 建立一個新檔案：search.php
- 把 archive.php 中所有東西複製到 search.php
- 儲存就完成了。

現在所有的，所有的搜尋結果將會返回摘要。如果沒有 search.php 這個模板檔案，搜尋選項將會使用index.php 去顯示搜尋結果。

（可選）你可以返回到課程1去回顧層次結構。

第3步：page.php 和 single.php

- 建立兩個新檔案：page.php 和 single.php
- 把 index.php 中所有內容複製到 page.php 和 single.php。（從現在開始，頁面和單篇文章應該是一樣的。）
- 儲存頁面和單篇文章檔案，關閉它們。

第4步：自訂 page.php

還記得**靜態頁面**和**頁面**之間的不同嗎？page.php 模板檔案是用來自訂化這些特殊靜態靜態頁面。

第一，在 page.php 中的 <?php the_content(); ?> 下輸入以下原始碼：

<?php link_pages(" <code>Pages: ", " ", "number"); ?>

和

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

第二，從 page.php 中移除 postmetadata 原始碼。結果如下：

```
<div class="entry">

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

</div>
```

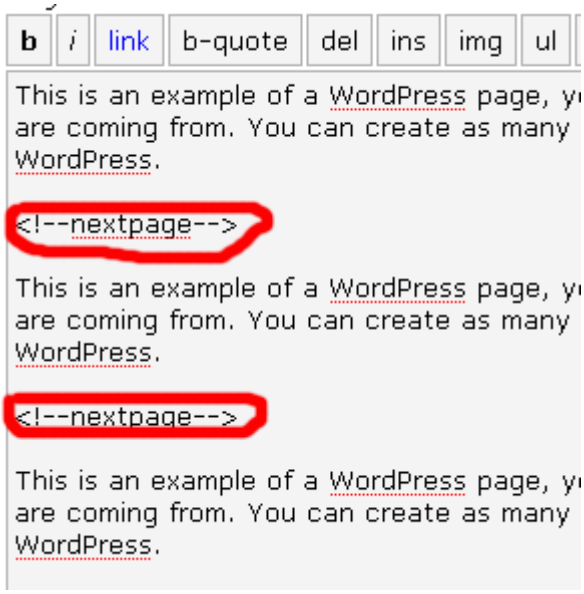
第三，在 page.php 中移除 posts_nav_link() 或者**導航**模塊。

```
        <?php endwhile; ?>

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

        <?php else : ?>
```

剛才發生了什麼？

第一行原始碼是用於顯示頁面的分頁連結。

舉個例子，編輯 **About** 頁面。根據我的畫面截圖增加原始碼：



當你想把一個非常長的頁面分成幾個頁面的時候，這是非常有用的。

第二行原始碼是用於顯示可以用來編輯靜態頁面的**編輯**連結。

通常頁面是沒有分類，並且通常不想給他們顯示建立時間，所以需要去移除 postmetadata。同樣要移除 posts_nav_link() 原始碼因為**靜態頁面**不會顯示後一頁和前一頁的連結。

儲存 page.php 檔案並關閉它。

第5步：自訂 single.php

點選一個文章的標題去閱讀文章其餘部分就會帶你到單篇文章查看模式。single.php 模板就是用於處理查看單篇文章時的外觀。

在 single.php 中的 <?php the_content() ?> 下輸入：

<?php link_pages("<p>Pages: ", "</p>", "number"); ?>

是的，這是相同的用於編碼頁面的分頁連結的原始碼。同樣我們也可以把文章分成多篇子文章。

第二，在 **postmetadata** 區域，移除 <?php comments_popup_link(); ?> 函數和前面的 **
** 標籤。不要移除整個 postmetadata。

移除了留言連結函數是因為在單篇文章查看模式下留言連結函數是不起作用，所以要在 singel.php 檔案中移除它。只有管理員可見的編輯連結，在 **BR** 標籤的左邊。你不想跳過一行才能看到這個本來你可以在右邊看到連結？這就是移除 **BR** 標籤的原因。

第三，用以下原始碼取代 <?php posts_nav_link(); ?>：

<?php previous_post_link("? %link") ?> <?php next_post_link(" %link ?") ?>

在前面，存檔，分類和搜尋頁面，我們使用 posts_nav_link() 函數去導入後一頁和前一頁的連結。對於查看單一文章的頁面，它是沒有後一頁和前一頁連結的，我們可以使用 **previous_post_link()** 和 next_post_link() 函數去導入前一篇文章和後一篇文章的連結。

儲存 single.php 檔案，到某篇文章下查看在導航區域的不同。

課程回顧

- 建立了四個新的檔案或者或者子模板：archive.php，search.php，page.php 和 single.php。
- archive.php 和 search.php 模板檔案是相同的。
- **Pages**（和文章不同）是沒有分類的，他們同樣沒有後一頁和前一頁的連結。
- Single.php 不會顯示留言連結（被 comments_popup_link() 函數導入）並且他不用 posts_nav_link() 去導入導航連結。

WordPress 主題教學 #16：留言模板

這篇教學是在 WordPress 2.7 之前撰寫的，而 WordPress 2.7 之後支援了 Thread Comments，這裡有讓你的主題實現 WordPress 2.7 的 Thread Comments 的方法。但是還是建議你查看下這篇教學。

留言模板是從零開始建立 WordPress 主題系列教學的最後一篇。這篇將涉及到網誌一個比較重要的東西：**評論模板**。

你應該知道：

- 沒有快速的方式在 comments.php 建立評論模板
- 大部分的 WordPress 設計者使用來自 WordPress 預設主題（**Kubrick**）的**預設評論模板根據**。
- 一些設計者會修改預設的評論模板去適合他們自己的需求。
- **你將使用我的對預設評論模板的修改版本。**

第1步：建立 comments.php

- 建立一個新檔案：comments.php。
- 把我的 comments.txt 檔案中的內容複製到 **comments.php**。
- 儲存 comments.php 檔案。

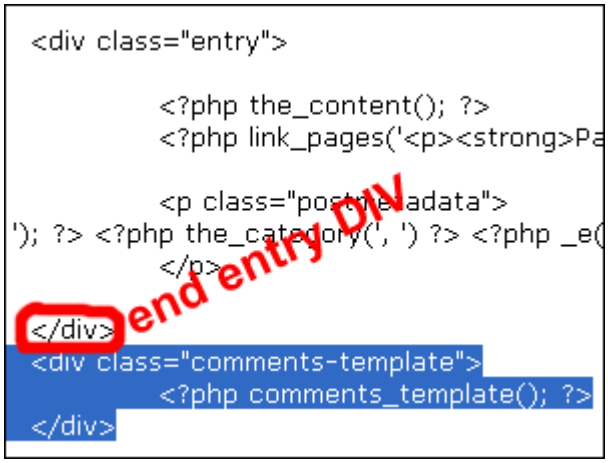
第2步：樣式化留言

- 把我的 comments-template-css 檔案中的內容複製到你的 **style.css** 檔案中。
- 複製到 style.css 的底部或者剛好 **#footer** 的上面。

第3步：在 single.php 加入留言模板

在 single.php 檔案中，**entry** DIV 的下面，輸入以下原始碼：

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



comments_template() 這個函數是用來從 **comments.php** 檔案導入評論模板。comments.php 檔案然後就會根據它的模板（或者原始碼）去顯示評論列表。列表中的每個項目是一條評論。

如果想讓人們可以在靜態頁面也可以留言，同樣可以把 comments_template() 函數用到 page.php 檔案。

第4步：驗證原始碼

第四步是驗證你的原始碼，然而可以不進行第四步的，因為你在使用的是我已經整理過的預設主題評論模板的修改版。**我已經替你驗證過原始碼了。**

驗證：

- **查看 > 頁面原始碼**
- 複製所有原始碼
- 然後到 validator。
- 把你的原始碼貼上到 **Direct Input** 框中。
- 點選 **Check**。

以後的參考（當你建立你自己的主題和評論模板），下面是需要驗證的頁面：

- 主頁 -- Home page
- 存檔頁面 -- Archive pages
- 分類頁面 -- Category pages（如果你自定義了類別頁面）
- 搜尋結果頁面 -- Search result pages
- 靜態頁面 -- Pages（如：About）
- 單一文章頁面 -- Single post view page
- 單一文章沒有留言 -- Single post with no comments
- 單一文章有留言 -- Single post with comments
- 單一文章含有必須登入訊息 -- Single post with must login message

- 單一文章沒有必須登入訊息 -- Single post with no login required message
- 密碼保護的單一文章並有留言 -- Password protected single post with comments

評論模板的進一步解釋

- 評論模板從根本上說是一個**有序列表**（OL），不是無序的，儘管它們基本上同樣方式工作。 無序列表是以圓點列表組織的。有序列表則是以數字列表組織的（每個項目都有一個數字，從1開始）。
- 在 **single.php** 檔案中，你用 **comments-template** DIV 圍住**comments_template()**。現在你的評論模板在一個 DIV 標籤中的一個有序列表中。

當你你的文章是密碼保護的，你的評論同樣是密碼保護的：

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

這個修改版的留言模板有一個 H2 子標題顯示 **Password Protected**。預設的留言模板是沒有的。

下面展示了哪些東西組成了你的留言列表：

```
<ol class="commentlist">
<?php foreach ($comments as $comment) : ?>

        <li class="<?php echo $oddcomment;

<div class="commentmetadata">
<strong><?php comment_author_link() ?></stro
comment_time() ?></a> <?php _e('Said&#58;')
                    <?php if ($comment->comm
                    <em><?php _e('Your comme
                    <?php endif; ?>
</div>

<?php comment_text() ?>
        </li>

<?php /* Changes every other comment to a di
        if ('alt' == $oddcomment) $oddcomme
        else $oddcomment = 'alt';
?>

<?php endforeach; /* end for each comment *
        </ol>
```

簡單整理下就是：

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

comment_text() 函數就是用來導入每條留言的。

我不會解釋留言模板的 CSS 原始碼的意思。不像 comments.php 檔案中的原始碼，你可以隨便測試你的 CSS 原始碼而不會弄壞留言模板。自己去測試回比我的解釋對你更有好處。

今天沒有課程回顧，你已經完成了 WordPress 主題製作教學。

---

erdaoo 的 WP Theme 教學學習筆記

本文由 erdaoo 學習我愛水煮魚的 WP Theme 教學之後的學習筆記整理，經我愛水煮魚整理，erdaoo 本人同意之後在我愛水煮魚發表，以便給更多學習 WP Theme 教學的人幫助。

學習本教學，需要：

1. 對PHP，WP，CSS，PS有一定的基礎，懂得初步的應用。
2. 你是一個想要表現自己的傢伙，並且不想再使用別人製作的主題。
3. 你要有耐心，細心，細心，耐心。

WP 主題簡介

在詳細分解原始碼之前，我們還是要先瞭解一下WP主題的大致情況。一個 WP 的主題是由幾個 templates 檔案組成的，每一個主題必有的二個檔案是：index.php 和 style.css（樣式表），除此之外還有一些其它的檔案（不是必須），它們和 index.php 檔案間存在優先級關係，如果它們存在，WP 模板系統就會導入它們顯示相應的頁面，否則模板系統會導入 index.php 來顯示。

它們有可能是以下檔案：

- single.php -- 單一文章檔案，用於顯示單一文章
- page.php -- 頁面模板檔案，用於顯示靜態頁面
- archive.php -- 存檔檔案，用於顯示存檔頁面
- category.php -- 類別檔案，用於顯示類別頁面
- search.php -- 搜尋檔案，用於顯示搜尋結果
- 404.php -- 錯誤檔案，用於顯示404頁面
- comments.php -- 評論檔案，用於顯示評論和評論框

index.php

首先製作index.php，我們知道在一個網頁中，原始碼主要分為二部分，一個是頁頭訊息，一個是頁面內容。

```
<html>
<head>
. . . . . .頁頭訊息
</head>
<body>
. . . . . .頁面內容
</body>
</html>
```

每個主題的頁頭訊息都是幾乎一樣，實際可以查看預設模板的 header.php 檔案（為保證所有頁面的頁頭訊息的一致性，所有頁頭訊息都放在 header.php 檔案。）

接下來我們談下一話題，關於母豬的產後護理……（我學的太雜了，都弄混了）

我們來談一下body中的內容。

它包含四個部分，每一部分都可以叫做一個集成模塊，其實一個主題就是由不同的模塊構成，模塊又是由不同的模塊構成。

- header WP 的頂部，顯示網誌的名字與描述，放置導航列，搜尋欄等等。
- content WP 的正文部分，顯示貼子的內容，作者，時間，分類，評論，編輯等等。
- sidebar WP 的側邊欄部分。
- footer WP 的尾部，這部分只有很少的內容，通常是版權訊息。

對於每一個集成模塊中的內容，理論上是可以隨意放置的，比如我們可以把header模塊中的搜尋欄放在sidebar模塊中去。

那如何區分這四個集成模塊呢？看以下原始碼。

```
<div  id="header">
      這是我的網誌
</div>
<div  id="content">
      這是我的文章</div>
<div  id="sidebar">
      搜尋欄，分類，存檔，友站連結
</div>
<div  id="footer">
      版權訊息，我是二道
</div>
```

透過 div 標籤，我們可以把這些模塊分隔開來。

header

現在開始我們第一部分的原始碼塊，不過在寫原始碼之前我還得要囉嗦一句，寫原始碼要有層次感，要記得縮排，不要用空格縮排而用TAB鍵。

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

id 是 div 的一個屬性，給 id 賦於不同的值，這樣就可以區分每一個div原始碼段。

bloginfo() 是 WP 中定義好的函數，參數 url 返回網址，參數 name 返回網站的名字，參數 description 返回網站描述。

在上面的原始碼中，就是為網誌的標題並加上一個超連結，並且顯示描述。

如果我們把上面的三行原始碼加上頁頭部分另存為一個新的檔案 -- header.php。這樣我們就可以透過以下 WP 函數導入它們。

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

這樣的好處是，你只要修改一下header.php檔案，所有導入這個檔案的頁面都會跟隨改變，而不用一個一個地去修改了。

content

現在開始我們第二部分的原始碼區塊：

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

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

這裡使用 if(have_posts() 來檢測是否有文章存在，如果有的話，就用 while 循環顯示。the_post() 就是導入文章的函數。

而每一篇文章又是有標題，有發佈時間，有分屬類別，有讀者的評論，這些又全部需要用 div 標籤來分隔開。看下面的原始碼：

```
<div id="content">
<?php if(have_posts()) : ?><!--開始檢測-->
<?php while(have_posts()) : the_post(); ?><!--以下面的格式顯示每篇文章-->
<div class="post">
<h2><a href="<?php the_permalink();?>"><?php the_title();?></a></h2><!--含有連結網址的文章標題-->
<div class="entry">
<?php the_content();?><!--文章內容-->
<p class="postmetadata"><!--文章元資料（Metadata）-->
<?php _e('Filed under:');?>
<?php the_category(',');?><!--導入文章的分類-->
<?php _e('by');?><!--使用_e()建立可翻譯的主題-->
<?php the_autnor('');?><!--導入文章作者-->
<br />
<?php comments_poopup_link('No Comments?','1 Comments?','% Comment?');?><!--導入一個跳出的留言視窗，如果這個功能沒有啟動，則是顯示留言列表-->
<?php edit_post_link('Edit','|','');?><!--只有在登陸後才可見到，對文章進行編輯的連結-->
</p>
</div><!--文章內容結束-->
</div><!--一篇文章徹底結束-->
<?php endwhile; ?>
<?php endif; ?>
</div>
```

class

現在我們得要說說 class 了，它是與 id 都是標籤的屬性，但是不同之處在於，id 的參數值是唯一的，它在一個頁面只能使用一次，而 class 的參數值是可以多次使用，比如 id="header" 只能出現一次，因為我們只有一個地方可以出現網誌的名字。而 class="entry" 會經常出現，那是因為我們的網誌裡不只是有一篇文章。

為什麼我們要用到 id 與 class，難道只用一個不行嗎，反正功能都是相同的。不要忘了我們前面說過的一個重要檔案，style.css 樣式表檔案。我們為某一段原始碼加入了屬性，如同起個名字而已，這樣在樣式表中我們就可以為這些名字來自訂它們的樣式了。

這樣說你還不明白？那就打個最簡單的比方吧，你可以有很多的兄弟，但是你們只能有一個爹，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。樣式表檔案就和你奶奶一樣，你爹再牛逼也得聽你奶奶的話，叫他怎麼樣他就得要怎麼樣。（老大你這個比喻寒啊，瀑布寒！！）

Not Found

前面的原始碼中有說到，如果檢測到有文章的話，就用循環調出來，可是如果沒有文章的話那要怎麼樣呢？

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

把這一段原始碼加在 <?php endwhile; ?> 之後就可以了。

頁面導航

當你的網誌內容越來越多的時候，在 WP 的後台又設定了首頁只顯示10個文章，那麼從第11個開始都無法在首頁顯示出來。

這樣在網誌的最後一篇文章下面就會出現後一頁或前一頁的連結。如果你還不到10個文章，這個連結就不會出現。

把下面的原始碼加入到 <?php endif; ?> 前面

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

分析一下 posts_nav_link() 這個 WP 函數，它可以有三個參數：<? posts_nav_link('in between','before','after')，第1個參數是顯示在後一頁和前一頁連結的中間。第2個參數顯示在後一頁和前一頁連結的前面。第3個參數顯示在後一頁和前一頁連結的後面。用什麼來顯示，你自己決定，常用的就是一些符號或是箭頭而已嘛。

現在再看一下我們已經有了哪些個原始碼：

```
<?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_poopup_link('No Comments?','1 Comments?','%
Comment?');?>
<?php edit_post_link('Edit','|','');?>
</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>
```

****

寫教學不是一個簡單的事，它不光讓我心煩，還讓我難以找到適當的詞來表達，所以要體會一下當老師的難處。

****

側邊欄

第三部分，關於側邊欄。側邊欄有一個特點，就是又臭又長，當然了這不是什麼纏腳布。先不要亂扯。因為地形有限，所以側邊欄裡的內容，多以列表的形式 排開。下面歡迎一對父子出場，他們的感情是相當的好，從來都是父子不分家，有父必有子，有子必有父，父中有子，子中有父。他們就是<UL>和<LI>!!!!!!!

```
<div  class="sidebar"><!--注意這裡使用的不是id-->
<ul>
<li>
<h2><?php _e('文章分類'); ?></h2>
</li>
</ul>
</div>
```

UL 表示無序列表，OL 表示列表元素。在側邊欄裡，你要有幾個不同的欄目，欄目的存在，就是為側邊欄進行了分類整理。每一個欄目又要有不同的分類列表，繼續為上面的原始碼加入內容。

```
<div  class="sidebar">
<ul>
<li><h2><?php _e('文章分類'); ?></h2>
<ul>
<?php wp_list_cats
('sort_column=name&optioncount=1&hierarchical=0'); ?>
</ul>
</li>
</ul>
</div>
```

wp_list_cats() 函數為導入文章分類列表，它的參數也有三個。每個參數之間用&來分隔。
sort_column=name -- 把分類按符號順序排列
optioncount=1 -- 顯示在每個分類下面的文章數
hierarchial=0 -- 不把子分類放到子列表項目中

說到分類，特別說一下靜態頁面這個欄目。我們在WP後台撰寫的時候，有二個選擇，一個是撰寫文章，一個是撰寫頁面。對於文章，還可以選擇儲存在哪一 個實際的分類下面。對於頁面就沒得選擇，只收錄於頁面欄目之下。 再回到前台，你可以看到每個分類都有顯示文章的數目，而不顯示標題。在頁面欄目裡，只排列了每一個頁面的標題，而不顯示數目。

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

參數depht=3為可選參數，表示可以設定顯示三級列表。

注意一點，本教學的原始碼是製作模版的原始碼（PHP 原始碼），在WP中使用一個主題也就是等於在套用一個模版。在網站中查看原始碼是看不到模版的原始碼的（已經被解釋成 HTML 原始碼）。

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

上面這一段模版原始碼，在網頁中查看原始碼，實際上顯示的是這樣的：

```
<li><h2>文章存檔</h2>
<ul>
<li><a  href="#">與愛情有關的分類貼子</a></li>
<li><a  href="#">與生活有關的分類貼子</a></li>
          ..............................      </ul>
</li>
```

增加一個存檔欄目：

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

wp_get_archives() 函數是用來獲取文章存檔的，參數'type=monthly'定義為以每個月的時間來進行分類存檔

增加一個友站連結欄目：

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

不用擔心沒有實際內容，它會自動導入在 WP 後台中加入的友站連結。

增加一個搜尋欄目：

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

這裡使用 include() 函數導入一個檔案，參數 TEMPLATEPATH 為主題檔案夾路徑，為了導入成功，你還需要有一個檔案：searchform.php。

增加一個日曆欄目：

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

這裡就不用多廢話了。

增加一個管理欄目：

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

wp_loginout() 來確定你是否登陸，如果登陸就顯示登出連結，如果沒有登陸，就顯示登陸的連結。
wp_register() 來確定你的身份，如果沒有登陸，就顯示註冊的連結，如果有的話，就顯示管理的連結。
而wp_meta() 卻是什麼也沒有做。也不用去理它，還沒有人來說明它是起什麼作用的。實際上它是 WordPress 的hook。

模組化側邊欄

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

在側邊欄開始的地方第一個
的後面，加上以上原始碼。也要在側邊欄結束的地方
前面加上一句

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

從 WP2.0 開始，已經在後台集成了一個側邊欄的外掛 - - Widget，它的功能就是可以很方便的在WP後台調整側邊欄中的內容，直接使用鼠標就可以移動每一個欄目的位置，而不需要去修改相應的原始碼。讓每一個欄目都以模組化存在。

function_exists('dynamic_sidebar') || !dynamic_sidebar()) 這兩個參數來自於一個新的檔案 -- functions.php 我們需要建立這個檔案才可以完成側邊欄的模組化.。

透過觀察不同的WP主題，會發現在側邊欄中的內容遠不止以上所列舉的，
要在學習中舉一反三，才會製作出更加出眾的主題。

至此，側邊欄中的內容結束，我們也可以把第三部分的原始碼另存為一個新的檔案 -- sidebar.php，在index.php中填加一句原始碼就可以使用側邊欄

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

順便再增加一行原始碼：

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

這是導入尾部檔案 footer.php 的原始碼。我想你應該知道如何處理一個簡單的 PHP 檔案了，要麼你就再重頭學一次本教學。

再一次查看一下index.php有了哪些原始碼

```
<?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_poopup_link('No Comments?','1 Comments?','%
Comment?');?>
<?php edit_post_link('Edit','|','');?>
</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 檔案的原始碼已經全在這裡了，但是只有第二部分內容需要詳細的原始碼，而其它的部分我們都可以導入外部檔案，至此一個 WP 的主題構造已經搭建好，再一次提醒各位，檢查原始碼，確認書寫正確。只有不厭其煩地寫原始碼才會對原始碼有更深刻的印像。

其他檔案

下面開始建立其它檔案

將index.php的全部原始碼另存為archive.php，並且把 the_content 改成 the_excerpt，
建立存檔檔案，它會顯示在分類欄目下的每篇文章的摘要。

將archive.php另存為 search.php，建立搜尋檔案，這樣就可以在搜尋中得到每篇文章的摘要。

將 index.php 的全部原始碼另存為 page.php，建立頁面模板檔案，在 <?php the_content(); ?> 下面輸入以下原始碼：

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

說明：如果一個頁面，篇幅超長的話，我們可以把它截斷分成幾頁來顯示，

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

說明：顯示一個可以編輯的連結

刪除掉 至 這一塊的原始碼
刪除掉以下原始碼：

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

說明：對於靜態頁面，它沒有屬於哪個分類，我們也不希望被某人評論，當然它也不能顯示與另一個頁面間的連接，所在要去掉一部分原始碼。

將index.php的全部原始碼另存為single.php，建立單篇文章檔案，點選文章的標題，可以查看全文內容。在 <?php the_content(); ?> 下輸入：

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

這段原始碼和上一例相同，都是可以用來截斷文章。

刪除以下原始碼：

```
<br />
<?php comments_poopup_link('No Comments?','1 Comments?','%
Comment?');?>
```

把 <?php posts_nav_link(); ?> 替換成 <?php previous_post_link('? %link') ?> <?php next_post_link(' %link ?') ?>

說明：在單篇文章的下面顯示的應是上一篇與下一篇的連結，而不是上一頁與下一頁的連結。

如何處理留言評論？

想一想，每一個留言評論都是對於一個文章而產生的，所以只要在單篇文章頁面裡加入一個導入評論的函數就可以。

在 single.php 檔案中 原始碼塊結束的 下面，輸入以下原始碼：

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

comments_template() 這個 WP 函數是用來從 comments.php 檔案導入評論模板。所以我們還要建立一個 comments.php 檔案。

主機推薦：Hostev Studios

Hostev Studios 是台灣虛擬主機提供商，能為 WordPress 架站者提供相當優質與穩定的網路空間，採用企業頂級頻寬，世界各地的訪客都能以相當順暢的連線速度進入你的網站。目前 Hostev Studios 提供台灣虛擬主機以及美國虛擬主機業務，數種方案可以選擇：

(TW) Standard

Hostev Studios 台灣虛擬網站方案內提供三種規格，一般使用者多會選擇 Standard 方案，包含 1GB 空間、10GB 流量、2 個附加網域以及 5 個資料庫，對於架設 WordPress 與中小型網站來說已相當充裕。Hostev Studios 使用相當安全的 256位元 SSL 客戶服務中心，保護你的個人隱私與安全。，主機端採用繁體中文 cPanel 控制台，建立資料庫只要三個步驟就能完成，如果需要，Hostev Studios 也能夠免費為客戶安裝 WordPress 系統以及其他必要的外掛程式，架站再也不會困擾你。

(TW) Professional

如果你覺得基本方案無法滿足你的需求，或是你希望在同個帳號內架設更多網站的話，Professional 專業方案會是最好的選擇。除了有更大的空間及流量外，包含以下特性：

- 儲存空間：提升為 2GB ，在多的檔案都不怕放不下。
- 可用流量：由 10GB 直接升為 20GB。
- 資料庫數：10 個（資料庫可以透過自訂前綴來安裝 N 個網誌，這個不是問題了）。
- 附加網域：10 個（包含無限的子網域數以及網域寄放）。

(US) 美國虛擬主機

如果你的訪客多半不在台灣，或是你想以較低的價格租用虛擬主機的話，不妨參考美國虛擬主機方案；但如果你是以中文撰寫網誌，或是流量多從亞洲地區而來，那麼我們還是強烈建議您選擇台灣虛擬主機。

台灣虛擬主機規格及價格比較

主機方案	半年價格	年付價格	儲存空間	流量	附加網域	資料庫

Beginner	920	1,720	200MB	6,000MB	1	1
Standard	2,170	4,160	1GB	10,000MB	2	5
Professional	3,640	6,910	2GB	20,000MB	10	10

付款方式

Hostev Studios 支援多種付款方式，包括 PayPal、ATM轉帳以及超商代收。如果你居住於台灣地區的話，可以使用轉帳或超商付款方式，以新台幣作為單位進行付款；若你居住在台灣以外的地區，建議您使用可以信賴的第三方 PayPal 進行付款。

如果你對 Hostev Studios 主機感興趣，在購買當中有任何問題，你可以找我（Email: Pserics@gmail.com 、MSN: j@go2.tw）諮詢。