# خود آموز طراحی و ساخت پوسته وردپرس

شامل یازده درس گام به گام جهت ساخت
یک پوسته برای وردپرس به زبان ساده.


پدید آورنده: حمید کاظمی

با همکاری حامد تکمیل

## مقدمه ی مولف

با توجه به کمبود مطلب و مرجع کامل در مورد طراحی پوسته های استاندارد وردپرس در وب فارسی، تصمیم گرفتم این مجموعه را برای شما دوستان تهیه نمایم. این کتاب الکترونیکی به صورت رایگان در اختیار کاربران عزیز قرار می گیرد . لطفا به من در پخش مناسب و هر چه بیشتر این کتاب کمک نمایید.

این کتاب مناسب افرادی است که آشنایی خوبی با مباحث طراحی وب داشته و قصد دارند وارد دنیای طراحی پوسته برای سیستم مدیریت محتوای محبوب وردپرس شوند.

در این کتاب آموزشی یازده درس مختلف خواهیم داشت و هدف یادگیری روش ساخت پوسته های غنی و به روز برای سیستم مدیریت محتوای وردپرس است. به تمامی نکاتی که برای تبدیل شدن شما به یک توسعه دهنده ی پوسته های وردپرس نیاز باشد در این کتاب قدم به قدم و از ابتدا خواهیم پرداخت.

بعد از مطالعه این کتاب شما قادر خواهید بود پوسته ای با امکانات بسازید زیر:

- رعایت تمامی مواردی که برای بهبود کار پوسته در موتورهای جستجو نیاز است
- افزودن میکرو فرمت هایی که توسط گوگل پشتیبانی می شود
- رعایت اصول طراحی و کدنویسی وب برای تولید خروجی استاندارد
- پوسته ی پیشرفته مبتنی بر سی اس اس
- کلاس های قوی برای مطالب نوشته ها و کامنت ها
- ابزارک های هوشمند و قوی
- و تمامی امکانات فوق العاده ای که پوسته های مدرن و پیشرفته وردپرس دارا هستند

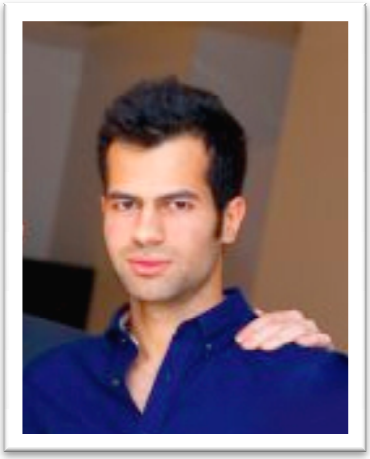
در ادامه لازم می دانم از همکاری دوست عزیزم حامد تکمیل که ویرایش و صفحه آرایی این نوشتار را پذیرفت قدردانی نمایم.

**پاییز ۱۳۹۱**

**حمید رضا کاظمی**

## حق نشر (Copyright)

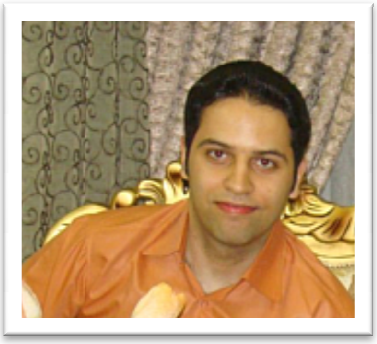
## حمید رضا کاظمی



همیشه علاقه زیاد من به اینترنت و توسعه محتوای وب فارسی باعث می شود انگیزه ای برای تولید محتواهای آموزشی داشته باشم و این هدیه کوچکی است از طرف من به تمامی دوستانی که علاقه ای همانند من دارند. در اینترنت محتوای آموزشی زیادی تولید کرده ام و امیدوارم برای شما علاقه مندان مفید واقع شده باشد. در آخر امیدوارم کارها و مطالبی که برای شما تهیه می کنم می کنم مورد پسند واقع شود و اگر پروژه های کاری داشتید خوشحال می شوم در خدمتتان باشم.

*aghahamidgol@gmail.com*
*http://www.facebook.com/HeamDesignStudio*

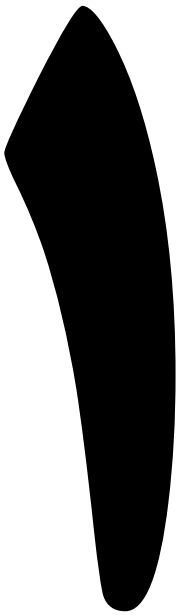
## حامد تکمیل



من یک خیال پرداز خوش بین هستم که همیشه دنبال عملی کردن ایده هام هستم. از طریق آزادکاری هزینه های زندگی رو در میارم و با وردپرس رفاقت دیرینه ای دارم. من اگر حرفی برای گفتن داشته باشم اون رو توی وبلاگم منعکس می کنم و در شبکه های اجتماعی آرام و بی هیاهو به فعالیت خودم ادامه می دم. چیزهای رایگان زیاد می سازم تا هم تفریح کرده باشم و هم اعتماد شما رو بیشتر جلب کنم تا از این رهگذر اگر رضایت خاطری برای شما حاصل شد افتخار همکاری در پروژه تجاری رو با هم داشته باشیم. با من در تماس باشید. ضمنا سایر کتاب های منتشر شده ی من به صورت رایگان در داخل وبلاگم در دسترس شما است.

*silvercover@gmail.com*
*http://silvercover.me*

# فصل اول : ابزار های لازم برای ساخت و توسعه پوسته



قبل از شروع کار باید ابزار مورد نیاز کار را تهیه نمایم. در این فصل از کتاب به توضیح و معرفی ابزار های کارآمد برای ادامه کار می پردازم. قبل از شروع هر کاری این فصل را به دقت مطالعه نمایید.

## سرور محلی[1] – XAMPP یا MAMP

بهترین کار برای توسعه پوسته های وردپرس انجام دادن این کار روی رایانه شخصی و بدون نیاز به سرور واقعی است! سرعت بالای سرورهای محلی و راحتی دسترسی به شما کمک می کند تغییرات را راحت تر اعمال نمایید. همان طور که اکثرا می دانید برای نصب وردپرس نیاز به وب سرورآپاچی، بانک اطلاعاتی مای اس کیو ال و مفسر زبان پی اچ پی داریم که همه ی این ها در بسته های نرم افزاری ارایه شده وجود دارند و قابل نصب می باشند.

اگر کاربر ویندوز است می توانید از برنامه XAMPP استفاده کنید که تنها کافی است آنرا در گوگل جستجو نمایید و به صورت رایگان نصب کنید.

کاربران مک نیز فراموش نشده اند و برای شما دوستان نیز برنامه MAMP توصیه می شود که در آدرس www.mamp.info قابل دسترس است.

## وردپرس

پر واضح است که برای توسعه وردپرس باید نسخه ای از این برنامه را دانلود کنیم. برای دانلود بسته وردپرس به سایت wordpress.org مراجعه نمایید و آخرین نسخه ی آنرا دانلود و روی سرور محلی خود نصب نمایید. روش نصب را در ضمیمه ی انتهای کتاب برایتان توضیح خواهم داد.

[1] - Local Host

## محتوای الکی !

از آنجا که نسخه ی وردپرسی که نصب می کنیم خالی از محتوا است می توان آنرا به راحتی پر نموده و برای اینکار نیازی نیست وقت زیادی مصرف نمایید. ما تنها یک سری محتوای فرضی را دانلود می کنیم و وارد سیستم[۲] وردپرس می کنیم. برای این کار مراحل زیر را طی نمایید :

از مرور گر خود وارد با وارد کردن آدرس localhost/wp-admin وارد بخش مدیریت ورپرس شوید، سپس وارد منو Tools > Import (ابزارها > درون ریزی) شوید و در لیست مربوط روی وردپرس کلیک نمایید. اکنون تنها به اطلاعات و محتوای الکی برای وارد سازی در سیستم نیاز داریم که می توانید برای دریافت این محتوا از لینک زیر استفاده نمایید :

*http://codex.wordpress.org/Theme_Unit_Test*

بعد از دانلود و درون ریزی کردن سایت شما از محتواهای الکی و بسیار مفید پر می شود !

## فایرفاکس

شما می توانید آزادانه هر مرورگری را که دوست دارید مورد استفاده قرار دهید ولی چیزی که من توصیه می کنم فایرفاکس است.

همچنین دو افزونه خیلی ضروری نیز نیاز داریم که روی فایرفاکس نصب نماییم. نام افزونه ها به شرح زیر است :

*Web Developer Add-on for Firefox*
*Firebug Add-on for Firefox*

افرونه های فوق از آدرس زیر قابل دسترسی و نصب هستند:

*https://addons.mozilla.org/en-US/firefox*

[2] - Import

## اچ تی ام ال[3] و سی اس اس[4]

نمی توانیم خودمان را گول بزنیم ! برای توسعه ی پوسته های وردپرس نیاز به دانش مقدماتی در مورد سی اس اس و اچ تی ام ال داریم که پیشنهاد می کنم قبل از خواندن این کتاب آنها را مطالعه نمایید.

## پی اچ پی[5]

آیا دانستن پی اچ پی نیز ضروری است؟ دانستن پی اچ پی مزیت بزرگی در این مرحله خواهد بود ولی ندانستن آن مشکلی ندارد ! در طول آموزش مواردی که نیاز به استفاده از پی اچ پی داریم کامل توضیح داده خواهد شد و جایی برای نگرانی نیست.

## ویرایشگر متن

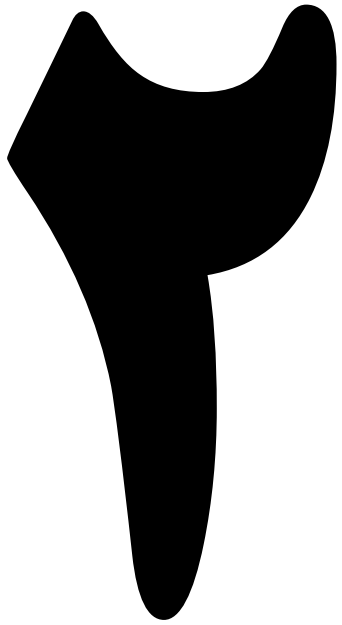
یکی از مهمترین ابزار مورد نیاز ما یک ویرایشگر متنی خوب است. هر کسی به صورت سلیقه ای برنامه ای برای خود انتخاب می کند . برای کاربران ویندوز برنامه Notepad++ را توصیه می کنم و دوستانی که از مک استفاده می کنند می توانند از TextWrangler استفاده نمایند.

ویرایشگر شما حتما نباید خیلی پیشرفته و با امکانات خیلی خاص باشد ، یک ویرایشگر ساده ی متنی نیز کار را راه خواهد انداخت.

در حال حاضر ویرایشگر متنی را نصب کرده ایم. سرور محلی نیز نصب و وردپرس روی آن نصب شده است و همه چیز برای شروع کار آماده است! امیدوارم فایرفاکس و افزونه هایی که نام بردم را فراموش نکرده باشید. به عنوان یک توسعه دهنده ی وب امیدوارم مشکلی در جستجو ، یافتن و نصب برنامه ها نداشته باشید.

[3] - HTML
[4] - CSS
[5] - PHP

# فصل دوم : ساخت ساختار اچ تی ام ال پوسته



تازه کار اصلی ما از این بخش آغاز می شود. باید کدهای HTML مربوط به پوسته را بنویسیم. برای کد نویسی همیشه باید دو نکته ی خیلی مهم را در نظر داشته باشید. نکته اول مختصر نویسی و تمیزی کد است و نکته ی دوم این است که کد شما باید معنا دار و خوانا باشد. از همین رو هنگام کد نویسی سعی می کنیم کدهایی قوی و مختصر با کمک تگ های HTML بنویسیم و از نام های استاندارد برای خوانا بودن کدمان استفاده نماییم.

ساختار کد HTML ما به صورت زیر است :

```
<html>
<head>
</head>
<body>
<div id="wrapper" class="hfeed">
     <div id="header">
          <div id="masthead">
               <div id="branding">
               </div><!-- #branding -->
               <div id="access">
               </div><!-- #access -->
          </div><!-- #masthead -->
     </div><!-- #header -->
     <div id="main">
          <div id="container">
               <div id="content">
               </div><!-- #content -->
          </div><!-- #container -->
          <div id="primary" class="widget-area">
          </div><!-- #primary .widget-area -->
          <div id="secondary" class="widget-area">
          </div><!-- #secondary -->
     </div><!-- #main -->
     <div id="footer">
          <div id="colophon">
               <div id="site-info">
               </div><!-- #site-info -->
          </div><!-- #colophon -->
     </div><!-- #footer -->
</div><!-- #wrapper -->
</body>
</html>
```

کد فوق را در ویرایشگر متنی خود کپی نمایید و در جایی ذخیره نمایید چون از این ساختار کد بسیار استفاده خواهیم کرد. قبل از ادامه ی آموزش یک سری نکات مهم را باید یادآور شد.

متوجه شده اید که در بخش اصلی HTML، ۲ بخش برای ابزارک ها در نظر گرفته شده است که بعد از بخشی که مربوط به مطالب سایت است کدنویسی شده اند. همین چند خط ساده به مرور و در فصل های بعدی و پس از اتمام دوره تبدیل به یک پوسته حرفه ای وردپرس خواهد شد.

# فصل سوم : ساختار پوسته و دایرکتوری آن

# ۳

ساده ترین پوسته ها در وردپرس ممکن است تنها شامل یک فایل index.php و style.css باشد ولی پوسته هایی که ما خواهیم کار کرد کمی پیشرفته تر هستند و برای کارهای مختلف و گسترده در نظر گرفته شده اند.

پوسته ای که در این کتاب خواهیم ساخت شامل ۶ فایل است. به پوشه وردپرس در مسیر زیر بروید :

*wp-content/themes*

و پوشه ای برای پوسته جدید خود ایجاد نمایید. در این کتاب من از اسم your-theme استفاده می کنم ولی این اسم هر چیزی می تواند باشد و به انتخاب شما خواهد بود. پس از ایجاد پوشه، فایل های زیر را در پوشه ایجاد شده بسازید. ( این فایل ها خالی خواهند بود)

- *index.php*
- *header.php*
- *sidebar.php*
- *footer.php*
- *functions.php*
- *style.css*

اکنون آخرین فایلی را که ساختیم در ویرایشگر متن خود باز می کنیم. بله منظور من فایل style.css است. در اول این فایل باید کدهایی را به صورت کامنت[۶] یا توضیح درج کنیم که در پنل مدیریت وردپرس هم ظاهر شود و اطلاعات کلی در مورد پوسته به سیستم وردپرس اعلام نماید.

ما با کمک این کدها که وجودشان ضروری است به وردپرس اطلاعاتی از قبیل نام پوسته، آدرس اینترنتی پوسته ، توضیحات ، سازنده ی پوسته و نسخه ی آن را اعلام می نماییم.

[6] - Comment

```
/*
Theme Name: Your Theme
Theme URI: http://eXAMPPle.com/eXAMPPle/
Description: A search engine optimized website framework for WordPress.
Author: You
Author URI: http://eXAMPPle.com/
Version: 1.0
Tags: Comma-separated tags that describe your theme
Your theme can be your copyrighted work.
Like WordPress, this work is released under GNU General Public License,
version 2 (GPL).
http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
*/
```

## ساخت فایل های footer.php و header.php

فایل HTML ای که در فصل قبل نوشتیم را باز کرده و همه ی کدهای بخش بالای کار را تا کد موجود در خط  درفایلی به نام header.php ذخیره می کنیم. این فایل را قبلا ساخته ایم و محتوای آن خالی بوده است ولی اکنون محتوای این فایل به صورت زیر است :

```
<html>
<head>
</head>
<body>
<div id="wrapper" class="hfeed">
    <div id="header">
        <div id="masthead">

            <div id="branding">
            </div><!-- #branding -->

            <div id="access">
            </div><!-- #access -->

        </div><!-- #masthead -->
    </div><!-- #header -->
    <div id="main">
```

در مرحله ی بعد فایل footer.php را که شامل کدهای زیر می شود می سازیم.

```
    </div><!-- #main -->
    <div id="footer">
        <div id="colophon">
            <div id="site-info">
            </div><!-- #site-info -->
        </div><!-- #colophon -->
    </div><!-- #footer -->
</div><!-- #wrapper -->
</body>
</html>
```

در حال حاضر فایل های فوتر و هدر ما کامل شده و فعلا آنها را ذخیره می کنیم و سراغ فایل index.php می رویم و کد های مابین و باقی مانده را در این فایل کپی و ذخیره می نماییم.

```
<div id="container">
    <div id="content">
    </div><!-- #content -->
</div><!-- #container -->
<div id="primary" class="widget-area">
</div><!-- #primary .widget-area -->
<div id="secondary" class="widget-area">
</div><!-- #secondary -->
```

در قدم بعدی در بالای فایل index.php و در خط اول، کد زیر را اضافه می کنیم:

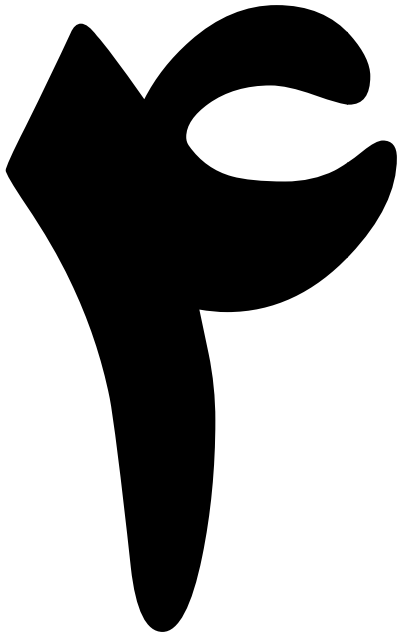
```
<?php get_header(); ?>
```

قطعه کد بالا باعث می شود فایل هدر را فراخوانی نماییم. نام تابع فوق get_header است. همین کار را برای فوتر انجام می دهیم و کد زیر را در پایین ترین جای index.php قرار می دهیم.

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

اکنون اگر وارد بخش مدیریت وردپرس شوید و پوسته را فعال نمایید و صفحه را بارگذاری مجدد نمایید و سورس صفحه را مشاهده نمایید، دقیقا کد اصلی html ما در آن موجود است. امیدوارم تا اینجای کار بدون مشکل به پیش آمده باشید. اگر همه چیز خوب پیش رفته است پس پیش به سوی فصل بعد.

# فصل چهارم : پوسته مربوط به  هدر



در این فصل در بخش هدر متمرکز خواهیم بود و با کدهای php زیادی سر و کار خواهیم داشت. البته کارهایی نیز برای سازگاری بهتر پوسته با موتورهای جستجو خواهیم کرد. در کل فصل مهم و مفیدی را در پیش رو خواهیم داشت.

در حال حاضر پوسته ما معتبر نیست و این موضوع به دلیل نداشتن ویژگی Doctype در کد HTML است. doctype باعث می شود که به مرورگر اعلام شود چگونه صفحه باز شده تفسیر شود. برای افزودن این امکان به پوسته خود، فایل header.php را باز نمایید و کد زیر را در اول پوسته و قبل از هر کد دیگری قرار دهید.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

هنوز کار ما در بالای کدها تمام نشده است. نوبت به افزودن چند صفت به تگ اولیه HTML است. کد زیر را جایگزین تگ html نمایید:

```
<html xmlns="http://www.w3.org/1999/xhtml" <?php language_attributes(); ?>>
```

اکنون وارد بخش <head> می شویم. در بخش head معمولا اطلاعاتی در مورد صفحه قرار می گیرد که شامل عنوان صفحه که در بالای مرورگر نمایش داده می شود ، فیدهای RSS و همین طور وصل نمودن فایل CSS به صفحه است.

تا یادمان نرفته باید گفت که لازم است کمی تغییرات در فایل functions.php ایجاد کنیم. فایل خالی functions.php را در ویرایشگر متنی خود باز کرده و بعد از تایپ کردن عبارت php?> کد زیر را به آن منتقل نمایید :

```
// Make theme available for translation
// Translations can be filed in the /languages/ directory
load_theme_textdomain( 'your-theme', TEMPLATEPATH . '/languages' );

$locale = get_locale();
$locale_file = TEMPLATEPATH . "/languages/$locale.php";
if ( is_readable($locale_file) )
    require_once($locale_file);

// Get the page number
function get_page_number() {
    if ( get_query_var('paged') ) {
        print ' | ' . __( 'Page ' , 'your-theme') . get_query_var('paged');
    }
} // end get_page_number
```

تابع load_theme_textdomain به وردپرس می گوید ما می خواهیم پوسته مان برای ترجمه به زبان های مختلف در دسترس قرار بگیرد. همین طور این تابع اعلام می کند فایل های مربوط به ترجمه در پوشه ای به نام languages قرار خواهند گرفت. در نظر داشته باشید که یک توسعه دهنده ی چیره دست باید به فکر تمامی زبان ها باشد و تا جایی که ممکن است تمامی بخش های کار را اعم از پوسته و افزونه های مختلف در زبان های مختلف و قابل توسعه منتشر نماید. این کار باعث می شود توسعه دهنده های مختلف از سراسر جهان و با زبان های مختلف، پوسته و افزونه های شما را ترجمه کنند و کار شما ابعاد جهانی پیدا کند.

در تابع بعدی get_page_number() شما متن های قابل ترجمه ای همانند زیر خواهید دید :

```
__( 'Page ' , 'your-theme')
```

متن قابل ترجمه ما در اینجا Page میباشد که همراه با آدرس دایرکتوری پوسته ما همراه شده است. توجه داشته باشید نباید نام پوسته را از your-theme به چیز دیگری تغییر دهید.

آیا می توانید حدس بزنید که این تابع چکار می کند؟ اگر نگاهی به داخل تابع کنید خواهید فهمید که با کمک یک دستور شرطی If این تابع شماره صفحه ای که در آن قرار دارید را چاپ خواهد کرد.

در مرحله بعدی کد <head></head> را پیدا نمایید و با کد زیر جایگزین نمایید:

```
<head profile="http://gmpg.org/xfn/11">
    <title><?php
        if ( is_single() ) { single_post_title(); }
        elseif ( is_home() || is_front_page() ) { bloginfo('name'); print ' |
'; bloginfo('description'); get_page_number(); }
        elseif ( is_page() ) { single_post_title(''); }
        elseif ( is_search() ) { bloginfo('name'); print ' | Search results
for ' . wp_specialchars($s); get_page_number(); }
        elseif ( is_404() ) { bloginfo('name'); print ' | Not Found'; }
        else { bloginfo('name'); wp_title('|'); get_page_number(); }
    ?></title>>

    <meta http-equiv="content-type" content="<?php bloginfo('html_type'); ?>;
charset=<?php bloginfo('charset'); ?>" />

    <link rel="stylesheet" type="text/css" href="<?php
bloginfo('stylesheet_url'); ?>" />

    <?php if ( is_singular() ) wp_enqueue_script( 'comment-reply' ); ?>

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

```
    <link rel="alternate" type="application/rss+xml" href="<?php
bloginfo('rss2_url'); ?>" title="<?php printf( __( '%s latest posts', 'your-
theme' ), wp_specialchars( get_bloginfo('name'), 1 ) ); ?>" />
    <link rel="alternate" type="application/rss+xml" href="<?php
bloginfo('comments_rss2_url') ?>" title="<?php printf( __( '%s latest
comments', 'your-theme' ), wp_specialchars( get_bloginfo('name'), 1 ) ); ?>"
/>
    <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
</head>
```

اگر این کدها برایتان آشنا نیست نگران نباشید به بررسی تمامی بخش های کد خواهیم پرداخت جای هیچ نگرانی نیست.

در بخش اول که کدهای آنرا به صورت جدا در زیر می بینید عنوان صفحه را مناسب برای موتورهای جستجوگر استخراج می کنیم. با توجه به قرار گیری کاربر در بخش های مختلف سایت عنوان مناسب را به خروجی می فرستیم.

```
<title><?php
    if ( is_single() ) { single_post_title(); }
    elseif ( is_home() || is_front_page() ) { bloginfo('name'); print ' | ';
bloginfo('description'); get_page_number(); }
    elseif ( is_page() ) { single_post_title(''); }
    elseif ( is_search() ) { bloginfo('name'); print ' | Search results for '
. wp_specialchars($s); get_page_number(); }
    elseif ( is_404() ) { bloginfo('name'); print ' | Not Found'; }
    else { bloginfo('name'); wp_title('|'); get_page_number(); }
?></title>
```

در بخشی دیگر اطلاعاتی از محتوای صفحه خود با کمک متا تگ ها به مرورگر می دهیم.

```
<meta http-equiv="content-type" content="<?php bloginfo('html_type'); ?>;
charset=<?php bloginfo('charset'); ?>" />
```

در خط بعدی صفحه خود را به فایل style.css متصل می کنیم.

```
<link rel="stylesheet" type="text/css" href="<?php
bloginfo('stylesheet_url'); ?>" />
```

در بخشی دیگر از کد، ویژگی نظردهی  توسط کاربران سایت را فعال نموده و همچنین RSS ها را با کمک کدهای زیر تعریف نموده ایم:

```
<link rel="alternate" type="application/rss+xml" href="<?php
bloginfo('rss2_url'); ?>" title="<?php printf( __( '%s latest posts', 'your-
theme' ), wp_specialchars( get_bloginfo('name'), 1 ) ); ?>" />
<link rel="alternate" type="application/rss+xml" href="<?php
bloginfo('comments_rss2_url') ?>" title="<?php printf( __( '%s latest
```

```
comments', 'your-theme' ), wp_specialchars( get_bloginfo('name'), 1 ) ); ?>"
/>
<link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
```

در این مرحله می خواهیم به بخش بالایی سایتمان عنوان سایت (لینک به خانه)، توضیحات وبلاگ و سایت و منو ها را قرار دهیم. فایل header.php را باز نمایید و #branding را در کد پیدا نمایید. در اینجا ما عنوان و توضیحات مربوط به سایت و وبلاگ را اضافه خواهیم کرد. در اکثر پوسته های وردپرس به موتوهای جستجو اعلام می شود که عنوان صفحه مهمترین عنصر موجود در صفحه است. توسعه دهنده ها این کار را با کمک قرار دادن عنوان صفحه در تگ h1 انجام می دهند. به هر حال برای اجرای این بخش تنها از تگ های وردپرس و HTML استفاده می کنیم. کدهای این بخش را در زیر مشاهده می کنید.

```
<div id="branding">
                <div id="blog-title"><span><a href="<?php bloginfo( 'url' )
?>/" title="<?php bloginfo( 'name' ) ?>" rel="home"><?php bloginfo( 'name' )
?></a></span></div>
<?php if ( is_home() || is_front_page() ) { ?>
                    <h1 id="blog-description"><?php bloginfo( 'description' )
?></h1>
<?php } else { ?>
                    <div id="blog-description"><?php bloginfo( 'description'
) ?></div>
<?php } ?>
            </div><!-- #branding -->
```

اگر جزو توسعه دهندگان مبتدی هستید، توصیه می کنم سعی کنید کد بالا را به دقت مطالعه و درک نمایید. من در این کد از تابعی[۷] به نام bloginfo() استفاده کرده ام. از این تابع برای برای گرفتن URL بلاگ، تیتر و توضیحات بلاگ استفاده کرده ام.

البته این تنها تمامی کاربردهای این تگ نیست. اگر این تگ را به دقت مطالعه نمایید بخش بزرگی از اتفاقاتی که در پوسته های وردپرس می افتد را می توانید درک نمایید.

تاکنون با پوسته ساده ی HTML ای که نوشتیم و افزودن تگ های PHP و وردپرس پیش رفتیم و قدم به قدم در حال توسعه ی پوسته استاندارد و کاربردی خود هستیم. کمی به پایین تر می رویم و #access را پیدا می کنیم و کد زیر را اضافه می کنیم. این کد باعث می شود افرادی که از screen reader (همچون تبلت ها) ها استفاده می کنند منو ها را مشاهده نکنند و مستقیم مطالب را مشاهده کنند.

[۷] - به این توابع تگ های پوسته یا Template Tag نیز گفته می شود

```
<div class="skip-link">
      <a href="#content" title="<?php _e( 'Skip to content', 'your-theme' )
?>"><?php _e( 'Skip to content', 'your-theme' ) ?></a>
</div>
```

در مرحله بعدی سراغ افزودن منو ها می رویم. کد زیر ساده ترین کد ممکن برای افزودن منو ها است که تنها از یک آرگومان در آن استفاده شده است.

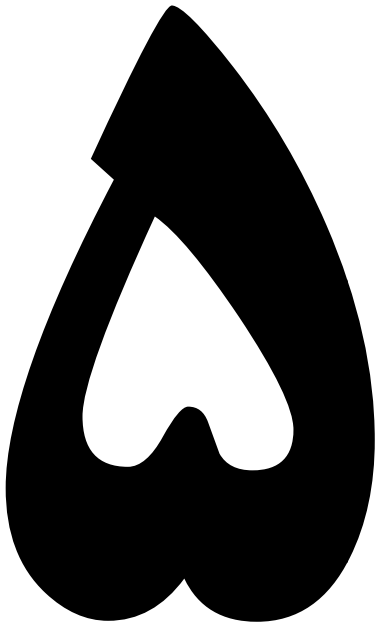
```
<?php wp_page_menu( 'sort_column=menu_order' ); ?>
```

اگر بخواهیم یک جمع بندی کنیم باید گفت که در حال حاضر بخش #access باید به صورت زیر شده باشد :

```
<div id="access">
    <div class="skip-link"><a href="#content" title="<?php _e( 'Skip to
content', 'your-theme' ) ?>"><?php _e( 'Skip to content', 'your-theme' )
?></a></div>
    <?php wp_page_menu( 'sort_column=menu_order' ); ?>
</div><!-- #access -->
```

بخش header پوسته ما به اتمام رسید و اکنون آماده ی استفاده است. کد ها طوری نوشته شده اند که مورد پسند موتورهای جستجو نیز باشند. اگر خسته شده اید یک استراحت کوچک کرده و برگردید و اگر مثل من وقتی پای کاری می نشینید نمی توانید از آن دل بکنید پس بشینید تا با هم به فصل بعد برویم.

# فصل پنجم : اتمام فایل ایندکس



بدون شک فایل index.php مهمترین فایل کاری است. اگر این فایل به دقت طراحی و کد شود تاثیر بی نظیری در کارکرد سایر بخش های پوسته خواهد داشت. هرگز از هیچ تلاشی برای بدون ایراد کردن فایل index دریغ نکنید.

## حلقه یا همان Loop معروف

در مورد حلقه شاید خیلی شنیده باشید. فایل مهم index.php با حلقه شروع و به اتمام می رسد. بدون حلقه شما در اصل هیچ چیز نخواهید داشت و تمام زحمات شما بدون نتیجه خواهد بود. حالا این کد حلقه که اینقدر مهم است چه شکلی است؟

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

شاید تعجب کرده باشید که چگونه کد به این سادگی نقش به این مهمی ایفا می کند ؟ تا هنگامی که در بانک اطلاعاتی خود نوشته و اطلاعاتی دارید این حلقه فعالیت می کند تا تمامی محتوای بانک اطلاعاتی را استخراج و به میل و سلیقه ما نمایش دهد.

حالا وارد بخش عملی کار شویم. کد زیر مربوط به حلقه است.آنرا در پوسته خود و در div به نام #content قرار می دهیم. این div همان طور که در جریان هستید در فایل index.php قرار دارد.

```
<ul>
<?php while ( have_posts() ) : the_post() ?>
<li>
    <?php the_excerpt(); ?>
</li>
<?php endwhile; ?>
</ul>
```

در این کد با کمک یک حلقه مطالب موجود در بانک اطلاعاتی را استخراج می کنیم و به صورت یک لیست آنها را نمایش می دهیم. برای این کار از تگ های خود وردپرس مانند the_excerpt() استفاده شده است. در واقع این حلقه آنقدر ادامه پیدا می کند تا مطالب مورد نظر ما به صورت کامل نمایش داده شود.

اکنون با کد ساده ی حلقه آشنا شدید. ولی این کافی نیست و باید حلقه ای قوی تر و کامل تر تهیه کنیم. در حلقه جدید کدهایی مربوط به تگهای بیشتر و تگ صفحه بعد را می خواهیم اضافه نماییم.

```
<div class="entry-content">
  <?php the_content( __( 'Continue reading <span class="meta-
nav">&raquo;</span>', 'your-theme' ) ); ?>
  <?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>
</div><!-- .entry-content -->
```

حال نوبت نمایش عنوان پست ها شده است ولی چگونه؟ جواب سوال خیلی ساده است. برای اینکار از تگ the_title() استفاده می کنیم تا عنوان هر پست را از بانک اطلاعاتی دریافت نماییم. عنوان گرفته شده را با کمک تگ <a> به تگ the_permalink() لینک خواهیم کرد و به همین راحتی کاربر با کلیک روی عنوان پست به لینک مربوط به آن خواهد رفت.

```
<h2 class="entry-title">
<a href="<?php the_permalink(); ?>" title="<?php printf( __('Permalink to
%s', 'your-theme'), the_title_attribute('echo=0') ); ?>" rel="bookmark"><?php
the_title(); ?></a>
</h2>
```

شاید شما هم مثل من بخواهید اطلاعات بیشتری از بانک اطلاعاتی استخراج کنید. اطلاعاتی مانند زمان منتشر شدن پست ، فردی که آنرا پست کرده است ، دسته بندی مطلب ، تگ ها و کامنت های مربوط به آن.

در ادامه برای تکمیل این بخش من نام نویسنده ی مطلب و تاریخ انتشار را قبل از مطلب اصلی و اطلاعاتی مانند دسته بندی ، تگ ها و کامنت ها را بعد از متن نوشته قرار می دهم. اگر بخواهیم تمامی نکاتی را که گفتم به صورت کد تبدیل کنیم کدمان به صورت زیر خواهد شد:

```
<?php /* The Loop — with comments! */ ?>
<?php while ( have_posts() ) : the_post() ?>

<?php /* Create a div with a unique ID thanks to the_ID() and semantic
classes with post_class() */ ?>
                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
<?php /* an h2 title */ ?>
                    <h2 class="entry-title"><a href="<?php the_permalink();
?>" title="<?php printf( __('Permalink to %s', 'your-theme'),
the_title_attribute('echo=0') ); ?>" rel="bookmark"><?php the_title();
?></a></h2>

<?php /* Microformatted, translatable post meta */ ?>
                    <div class="entry-meta">
                        <span class="meta-prep meta-prep-author"><?php _e('By
', 'your-theme'); ?></span>
                        <span class="author vcard"><a class="url fn n"
href="<?php echo get_author_link( false, $authordata->ID, $authordata-
>user_nicename ); ?>" title="<?php printf( __( 'View all posts by %s', 'your-
theme' ), $authordata->display_name ); ?>"><?php the_author(); ?></a></span>
                        <span class="meta-sep"> | </span>
```

```
                        <span class="meta-prep meta-prep-entry-date"><?php
_e('Published ', 'your-theme'); ?></span>
                        <span class="entry-date"><abbr class="published"
title="<?php the_time('Y-m-d\TH:i:sO') ?>"><?php the_time( get_option(
'date_format' ) ); ?></abbr></span>
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t" ) ?>
                    </div><!-- .entry-meta -->

<?php /* The entry content */ ?>
                    <div class="entry-content">
<?php the_content( __( 'Continue reading <span class="meta-
nav">&raquo;</span>', 'your-theme' )  ); ?>
<?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>
                    </div><!-- .entry-content -->

<?php /* Microformatted category and tag links along with a comments link */
?>
                    <div class="entry-utility">
                        <span class="cat-links"><span class="entry-utility-
prep entry-utility-prep-cat-links"><?php _e( 'Posted in ', 'your-theme' );
?></span><?php echo get_the_category_list(', '); ?></span>
                        <span class="meta-sep"> | </span>
                        <?php the_tags( '<span class="tag-links"><span
class="entry-utility-prep entry-utility-prep-tag-links">' . __('Tagged ',
'your-theme' ) . '</span>', ", ", "</span>\n\t\t\t\t\t\t\t<span class=\"meta-
sep\">|</span>\n" ) ?>
                        <span class="comments-link"><?php
comments_popup_link( __( 'Leave a comment', 'your-theme' ), __( '1 Comment',
'your-theme' ), __( '% Comments', 'your-theme' ) ) ?></span>
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t\n" ) ?>
                    </div><!-- #entry-utility -->
                </div><!-- #post-<?php the_ID(); ?> -->

<?php /* Close up the post div and then end the loop with endwhile */ ?>

<?php endwhile; ?>
```

## پیمایشگر[8]

برای پیمایش بین پست های مختلف بدون شک باید راهی در نظر بگیریم. برای انجام این کار از دو تگ وردپرس به نام های next_posts_link() و previous_posts_link() استفاده خواهیم کرد. درک این بخش کمی منطقی نیست. تابع اول کارش این است که وقتی کاربر روی آن کلیک می کند به مطلب قبلی می رود و اگر روی لینکی که تابع دوم یعنی previous_posts_link() ایجاد می کند کلیک کنید پست بعدی

[8] - Navigator

نمایش داده می شود. از نظر ترجمه زبانی و واقعیتی که این تابع ها چه کار می کنند کمی گیج کننده است. این نکته را حتما در نظر داشته باشید.

همانند تمامی اجزای صفحه index.php باید دقت خوبی در استفاده از این کد ها داشته باشید تا به بهترین نحو ممکن کار کنند. من دوست دارم پیمایشگر وبلاگم در بالا و پایین مطلب اصلی نمایش داده شود. البته در صورتی که خواستید می توانید پیمایشگر ها را در شرایطی خاص مخفی نمایید. برای مثال در برگه ها جستجو و یا برگه های که دیگر پست های قدیمی تر وجود ندارد نیازی به نمایش خروجی این توابع نداریم پس بهتر است کدی بنویسیم تا چیزهای غیر ضروری را مخفی نماید و باعث شلوغی بیهوده صفحه نشود.

```
<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
<?php } ?>
```

کاری که در کد بالا انجام داده ایم چیست؟ در اصل ما کنترل می کنیم تا بفهمیم حداکثر تعداد برگه های که در حلقه باید اجرا شود چقدر است و اگر تعداد برگه ها بیشتر از ۱ باشد ما پیمایشگر را به خروجی می فرستیم. کدی که برای نمایش در قبل و بعد از حلقه اصلی نیاز داریم به صورت زیر خواهد بود.

### قبل از حلقه

```
<?php /* Top post navigation */ ?>
<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-above" class="navigation">
                    <div class="nav-previous"><?php next_posts_link(__(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link(__(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-above -->
<?php } ?>
```

### بعد از حلقه

```
<?php /* Bottom post navigation */ ?>
<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-below" class="navigation">
                    <div class="nav-previous"><?php next_posts_link(__(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link(__(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-below -->
<?php } ?>
```

باید خدمت شما عرض کنم که کار ما با فایل index.php به اتمام رسیده است. فقط در آخر این فایل خط زیر را قبل از تگ get_footer() قرار دهید تا ستون کناری هم به پوسته اضافه شود.

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

# فصل ششم : نوشته ها ، ضمایم و خطای ۴۰۴

در این فصل به سه بخش مختلف و مهم پوسته های وردپرس خواهیم پرداخت. نوشته هایی که معمولا در وبلاگ ها و به تعداد زیاد و روزانه آپدیت می شوند. ضمیمه های برگه ها و نوشته ها و در آخر خطای ۴۰۴ که باید پوسته ای برایش در نظر بگیریم تا کمک کنیم کاربر در مسیر صحیح قرار گیرد.

همان طور که در فصل قبل با من همراه بودید فایل index.php را تمام کردیم. در این مرحله فایل single.php را در ویرایشگر خود باز نمایید و کد زیر را در آن کپی کنید. با کمی توجه به کد متوجه خواهید شد که ساختار کلی این کد شباهت خیلی زیادی به کدهای فایل index.php دارد.

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

        <div id="container">
            <div id="content">

                <div id="nav-above" class="navigation">
                </div><!-- #nav-above -->

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                </div><!-- #post-<?php the_ID(); ?> -->

                <div id="nav-below" class="navigation">
                </div><!-- #nav-below -->


            </div><!-- #content -->
        </div><!-- #container -->

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

ولی تفاوت های قابل توجه و مهمی وجود دارد. در این کد ما تگ the_post() را در نزدیکی بالای صفحه فراخوانی کرده ایم دقیقا قبل از پیمایشگر خود. در این صفحه ما نیازی به حلقه معروف وردپرس نخواهیم داشت. خوشبختانه با کمک تابع the_permalink() و کمک وردپرس مشکل ما حل می شود و دیگر در این بخش حلقه را به کار نخواهیم گرفت.

در نوشته ها ما نیاز به سیستم نظردهی و پیام گذاری داریم. برای این کار نیاز به تابع comments_template() داریم. در این مرحله تنها یک خط کد خواهیم داشت و سیستم نظردهی را در صفحه ای جدا به نام comments.php کدنویسی خواهیم کرد. این خط کد را بعد از کد پیمایشگر قرار می دهیم.

```
<?php comments_template('', true); ?>
```

## پیمایشگر مربوط به پست ها

برای خوانا بودن و منطقی بودن کدهایمان در این قسمت به جای استفاده از توابعی که در اسم گذاری آنها منطقی وجود ندارد از توابع previous_post_link() و next_post_link() استفاده خواهیم نمود. این توابع دقیقا کاری را انجام می دهند که شما انتظار خواهید داشت.

```
<div id="nav-above" class="navigation">
    <div class="nav-previous"><?php previous_post_link( '%link', '<span
class="meta-nav">&laquo;</span> %title' ) ?></div>
    <div class="nav-next"><?php next_post_link( '%link', '%title <span
class="meta-nav">&raquo;</span>' ) ?></div>
</div><!-- #nav-above -->

<div id="nav-below" class="navigation">
    <div class="nav-previous"><?php previous_post_link( '%link', '<span
class="meta-nav">&laquo;</span> %title' ) ?></div>
    <div class="nav-next"><?php next_post_link( '%link', '%title <span
class="meta-nav">&raquo;</span>' ) ?></div>
</div><!-- #nav-below -->
```

## عنوان پست ها

اگر یادتان باشد در فصلی که فایل header.php را کامل کردیم از یک دستور شرطی If بهره بردیم تا کاربران معمولی را با آنهایی که از سایر دستگاه ها استفاده می کنند جدا نماییم.

در ادامه عنوان و تیتر را در تگ h1 قرار می دهیم و کدی همانند زیر را می نویسیم.

```
<h1 class="entry-title"><?php the_title(); ?></h1>
```

سراغ تگی که کلاس آن entry-utility است می رویم و کدی را که در ادامه می بینید برای آن می نویسیم. این کد با کمک دستورات شرطی هوشمند شده و مطابق با عملکرد وردپرس تهیه شده است. به این معنی که حالت های مختلفی مانند باز بودن نظردهی و بازخوردها[9] در شرایط مختلف مورد مقایسه قرار می گیرد. برای اینکه هر مرحله را خوب درک کنید و کد خوانا تر باشد جلوی هر بخش در کد توضیحات لازم درج شده است.

```
<div class="entry-utility">
                    <?php printf( __( 'This entry was posted in %1$s%2$s.
Bookmark the <a href="%3$s" title="Permalink to %4$s"
rel="bookmark">permalink</a>. Follow any comments here with the <a
href="%5$s" title="Comments RSS to %4$s" rel="alternate"
type="application/rss+xml">RSS feed for this post</a>.', 'your-theme' ),
                        get_the_category_list(', '),
```

[9] - Track Back

```
                        get_the_tag_list( __( ' and tagged ', 'your-theme' ),
', ', '' ),
                        get_permalink(),
                        the_title_attribute('echo=0'),
                        comments_rss() ) ?>

<?php if ( ('open' == $post->comment_status) && ('open' == $post-
>ping_status) ) : // Comments and trackbacks open ?>
                        <?php printf( __( '<a class="comment-link"
href="#respond" title="Post a comment">Post a comment</a> or leave a
trackback: <a class="trackback-link" href="%s" title="Trackback URL for your
post" rel="trackback">Trackback URL</a>.', 'your-theme' ),
get_trackback_url() ) ?>
<?php elseif ( !('open' == $post->comment_status) && ('open' == $post-
>ping_status) ) : // Only trackbacks open ?>
                        <?php printf( __( 'Comments are closed, but you can
leave a trackback: <a class="trackback-link" href="%s" title="Trackback URL
for your post" rel="trackback">Trackback URL</a>.', 'your-theme' ),
get_trackback_url() ) ?>
<?php elseif ( ('open' == $post->comment_status) && !('open' == $post-
>ping_status) ) : // Only comments open ?>
                        <?php _e( 'Trackbacks are closed, but you can <a
class="comment-link" href="#respond" title="Post a comment">post a
comment</a>.', 'your-theme' ) ?>
<?php elseif ( !('open' == $post->comment_status) && !('open' == $post-
>ping_status) ) : // Comments and trackbacks closed ?>
                        <?php _e( 'Both comments and trackbacks are currently
closed.', 'your-theme' ) ?>
<?php endif; ?>
<?php edit_post_link( __( 'Edit', 'your-theme' ), "\n\t\t\t\t\t<span
class=\"edit-link\">", "</span>" ) ?>
                    </div><!-- .entry-utility -->
```

برخلاف فایل index.php در فایل single.php محتوا خیلی ساده است. فقط یک تابع را در آن فراخوانی می کنیم.

```
<?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>
```

### ضمیمه های نوشته ها

استفاده از ضمیمه ها در وردپرس زیاد فراگیر نیست ولی این دلیل نمی شود که ما از این بخش چشم پوشی نماییم. وقتی شما عکسی در پست خود قرار می دهید در اصل شما آنرا ضمیمه آن پست نموده اید. البته شما قادر به ضمیمه کردن تصاویری بیشتر از یکی نیز می باشید.

در این قدم شروع به ویرایش و ساخت فایل attachments.php می کنیم. ساده ترین کار این است که تمام محتوای فایل single.php را به فایل attachment.php منتقل نماییم و تغییرات جزیی زیر را در آن اعمال

کنیم. اول از همه بخش پیمایشگر بالایی را حذف کنید ما به این بخش نیاز نخواهیم داشت. جای آن کد زیر را که مربوط به عنوان و لینک های بازگشت است را قرار دهید.

```
<h1 class="page-title"><a href="<?php echo get_permalink($post->post_parent)
?>" title="<?php printf( __( 'Return to %s', 'your-theme' ), wp_specialchars(
get_the_title($post->post_parent), 1 ) ) ?>" rev="attachment"><span
class="meta-nav">&laquo; </span><?php echo get_the_title($post->post_parent)
?></a></h1>
```

هنگامی که عنوان صفحه در تگ h1 است باید تگ نوشته را در یک تگ h2 قرار دهیم.

```
<h2 class="entry-title"><?php the_title(); ?></h2>
```

در واقع باید این صفحه پیوست ها و ضمیمه های نوشته را نشان  می دهد و از آنجایی که اکثر پیوست ها تصویری می باشند ما باید با کمک دستور های شرطی If آنها را کنترل نماییم.

```
<div class="entry-content">
                        <div class="entry-attachment">
<?php if ( wp_attachment_is_image( $post->id ) ) : $att_image =
wp_get_attachment_image_src( $post->id, "medium"); ?>
                        <p class="attachment"><a href="<?php echo
wp_get_attachment_url($post->id); ?>" title="<?php the_title(); ?>"
rel="attachment"><img src="<?php echo $att_image[0];?>" width="<?php echo
$att_image[1];?>" height="<?php echo $att_image[2];?>"  class="attachment-
medium" alt="<?php $post->post_excerpt; ?>" /></a>
                        </p>
<?php else : ?>
                        <a href="<?php echo wp_get_attachment_url($post->ID)
?>" title="<?php echo wp_specialchars( get_the_title($post->ID), 1 ) ?>"
rel="attachment"><?php echo basename($post->guid) ?></a>
<?php endif; ?>
                        </div>
                        <div class="entry-caption"><?php if ( !empty($post-
>post_excerpt) ) the_excerpt() ?></div>


<?php the_content( __( 'Continue reading <span class="meta-
nav">&raquo;</span>', 'your-theme' )  ); ?>
<?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>

                    </div><!-- .entry-content -->
```

بخش پایینی پیمایشگر را که از فایل single.php کپی کردیم حذف نمایید تا کار ما با فایل مربوط به پیوست ها خاتمه یابد.

## پوسته مربوط به خطای ۴۰۴

وقتی کاربر صفحه ای را نمی تواند پیدا نماید باید به صفحه و پوسته مربوط به خطا منتقل شود تا با کمکی که به او می شود دوباره به مسیر وبلاگ و سایت برگردد و راهنمایی لازم برای وی صورت بگیرد. داشتن چنین پوسته خطایی خیلی ضروری است و اکثر توسعه دهنده های حرفه ای این ویژگی را در پوسته های خود قرار می دهند.

همان کارها و مراحل قبلی را برای فایل 404.php دوباره انجام می دهیم و تنها کد زیر را جایگزین می نماییم. اما همانند قبل کد مربوط به پیمایشگرها را نیز حذف می کنیم.

```
<div id="post-0" class="post error404 not-found">
   <h1 class="entry-title"><?php _e( 'Not Found', 'your-theme' ); ?></h1>
                <div class="entry-content">
                    <p><?php _e( 'Apologies, but we were unable to find what
you were looking for. Perhaps searching will help.', 'your-theme' ); ?></p>
<?php get_search_form(); ?>
                </div><!-- .entry-content -->
</div><!-- #post-0 -->
```

روش های خلاقانه زیادی را می شود برای نمایش صفحه خطا به کاربر به کار برد ولی در این مثال من فقط از کاربر عذرخواهی می کنم و با قرار دادن امکان جستجو در صفحه به او کمک خواهم کرد مطلب مورد نظر خود را دوباره جستجو نماید.

# فصل هفتم : سیستم درج نظر و پیام گذاری

# ۷

تهیه این بخش برای من خیلی عذاب آور هست و هیچ وقت علاقه ای به تهیه پوسته برای بخش کامنت ها نداشته ام. ولی خب واقعا نمی شود از یکی از مهمترین بخش های کار چشم پوشی کرد. سیستم پیام ها و نظردهی جزو مهمترین بخش های وردپرس است. در این فصل برای راحتی کار شما و درک بهتر کارهایی صورت داده ام تکه کدی در اختیارتان قرار می دهم که کافی است به فایل functions.php اضافه نمایید. کدهای آماده ای در اختیارتان می گذارم تا کمتر وارد این بخش شویم و تنها ساختار کلی و توسعه این پوسته را در اولویت قرار دهیم.

اما نگاهی کنیم به کارهایی که قرار است در این پوسته صورت بگیرد:

- افزودن امکان پست هایی که توسط پسورد رمزگذاری می شوند
- چک نمودن اینکه دیدگاهی وجود دارد یا خیر
- شمارش تعداد کامنت ها و بازخورد ها
- صفحه بندی کردن نظر ها در صورت زیاد بودن آنها
- اگر بازخورد وجود داشت آنها را نمایش دهیم
- نمایش سیستم نظرات ها و پاسخ گویی[10]

تمامی مواردی که ذکر شد در پوسته و کدی که در ادامه به شما می دهم نوشته شده است. این ۲ تابع نوشته شده را در فایل functions.php اضافه نمایید.

## تابع اول

```
// Custom callback to list comments in the your-theme style
function custom_comments($comment, $args, $depth) {
  $GLOBALS['comment'] = $comment;
    $GLOBALS['comment_depth'] = $depth;
  ?>
    <li id="comment-<?php comment_ID() ?>" <?php comment_class() ?>>
        <div class="comment-author vcard"><?php commenter_link() ?></div>
        <div class="comment-meta"><?php printf(__('Posted %1$s at %2$s <span
class="meta-sep">|</span> <a href="%3$s" title="Permalink to this
comment">Permalink</a>', 'your-theme'),
                    get_comment_date(),
                    get_comment_time(),
                    '#comment-' . get_comment_ID() );
                    edit_comment_link(__('Edit', 'your-theme'), ' <span
class="meta-sep">|</span> <span class="edit-link">', '</span>'); ?></div>
```

[10] - Reply

```
  <?php if ($comment->comment_approved == '0') _e("\t\t\t\t\t<span
class='unapproved'>Your comment is awaiting moderation.</span>\n", 'your-
theme') ?>
          <div class="comment-content">
            <?php comment_text() ?>
        </div>
        <?php // echo the comment reply link
            if($args['type'] == 'all' || get_comment_type() == 'comment') :
                comment_reply_link(array_merge($args, array(
                    'reply_text' => __('Reply','your-theme'),
                    'login_text' => __('Log in to reply.','your-theme'),
                    'depth' => $depth,
                    'before' => '<div class="comment-reply-link">',
                    'after' => '</div>'
                )));
            endif;
        ?>
      <?php } // end custom_comments
```

## تابع دوم

```
// Custom callback to list pings
function custom_pings($comment, $args, $depth) {
       $GLOBALS['comment'] = $comment;
        ?>
            <li id="comment-<?php comment_ID() ?>" <?php comment_class() ?>>
                <div class="comment-author"><?php printf(__('By %1$s on %2$s
at %3$s', 'your-theme'),
                        get_comment_author_link(),
                        get_comment_date(),
                        get_comment_time() );
                        edit_comment_link(__('Edit', 'your-theme'), ' <span
class="meta-sep">|</span> <span class="edit-link">', '</span>'); ?></div>
    <?php if ($comment->comment_approved == '0') _e('\t\t\t\t\t<span
class="unapproved">Your trackback is awaiting moderation.</span>\n', 'your-
theme') ?>
            <div class="comment-content">
                <?php comment_text() ?>
            </div>
      <?php } // end custom_pings
```

ما نیاز به یک تابع دیگر نیز داریم ، نام این تابع را custome_comments() می گذاریم. به کد مربوط نگاهی می اندازیم :

```
// Produces an avatar image with the hCard-compliant photo class
function commenter_link() {
    $commenter = get_comment_author_link();
    if ( ereg( '<a[^>]* class=[^>]+>', $commenter ) ) {
        $commenter = ereg_replace( '(<a[^>]* class=[\'"]?)', '\\1url ' ,
$commenter );
    } else {
        $commenter = ereg_replace( '(<a )/', '\\1class="url "' , $commenter
);
    }
    $avatar_email = get_comment_author_email();
    $avatar = str_replace( "class='avatar", "class='photo avatar",
get_avatar( $avatar_email, 80 ) );
    echo $avatar . ' <span class="fn n">' . $commenter . '</span>';
} // end commenter_link
```

این کد تنظیمات مربوط به gravatar را دارا است که البته می توانید سایز آنرا نیز تغییر دهید. سایز پیشفرض ۸۰ پیکسل است که می توانید آنرا ویرایش نمایید.

### پوسته مربوط به نظر ها

کد کامل بخش مربوط به پوسته نظر ها در زیر قابل دسترس است.

```
<?php /* The Comments Template – with, er, comments! */ ?>
            <div id="comments">
<?php /* Run some checks for bots and password protected posts */ ?>
<?php
    $req = get_option('require_name_email'); // Checks if fields are
required.
    if ( 'comments.php' == basename($_SERVER['SCRIPT_FILENAME']) )
        die ( 'Please do not load this page directly. Thanks!' );
    if ( ! empty($post->post_password) ) :
        if ( $_COOKIE['wp-postpass_' . COOKIEHASH] != $post->post_password )
:
?>
                <div class="nopassword"><?php _e('This post is password
protected. Enter the password to view any comments.', 'your-theme') ?></div>
            </div><!-- .comments -->
<?php
        return;
    endif;
endif;
?>

<?php /* See IF there are comments and do the comments stuff! */ ?>
<?php if ( have_comments() ) : ?>

<?php /* Count the number of comments and trackbacks (or pings) */
```

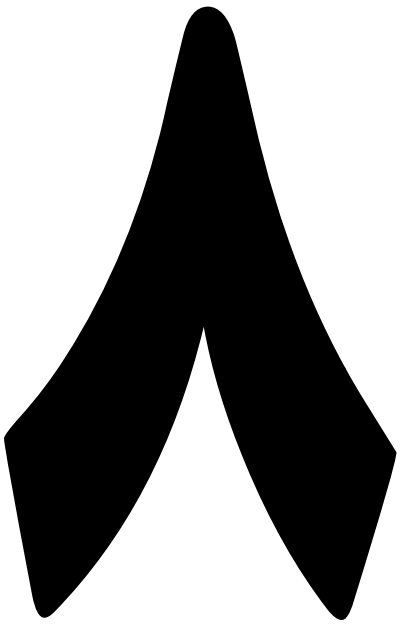
```
$ping_count = $comment_count = 0;
foreach ( $comments as $comment )
    get_comment_type() == "comment" ? ++$comment_count : ++$ping_count;
?>

<?php /* IF there are comments, show the comments */ ?>
<?php if ( ! empty($comments_by_type['comment']) ) : ?>

                <div id="comments-list" class="comments">
                    <h3><?php printf($comment_count > 1 ? __('<span>%d</span>
Comments', 'your-theme') : __('<span>One</span> Comment', 'your-theme'),
$comment_count) ?></h3>

<?php /* If there are enough comments, build the comment navigation  */ ?>
<?php $total_pages = get_comment_pages_count(); if ( $total_pages > 1 ) : ?>
                    <div id="comments-nav-above" class="comments-navigation">
                                    <div class="paginated-comments-links"><?php
paginate_comments_links(); ?></div>
                    </div><!-- #comments-nav-above -->
<?php endif; ?>

<?php /* An ordered list of our custom comments callback, custom_comments(),
in functions.php   */ ?>
                    <ol>
<?php wp_list_comments('type=comment&callback=custom_comments'); ?>
                    </ol>

<?php /* If there are enough comments, build the comment navigation */ ?>
<?php $total_pages = get_comment_pages_count(); if ( $total_pages > 1 ) : ?>
                <div id="comments-nav-below" class="comments-navigation">
                        <div class="paginated-comments-links"><?php
paginate_comments_links(); ?></div>
            </div><!-- #comments-nav-below -->
<?php endif; ?>

                </div><!-- #comments-list .comments -->

<?php endif; /* if ( $comment_count ) */ ?>

<?php /* If there are trackbacks(pings), show the trackbacks  */ ?>
<?php if ( ! empty($comments_by_type['pings']) ) : ?>

                <div id="trackbacks-list" class="comments">
                    <h3><?php printf($ping_count > 1 ? __('<span>%d</span>
Trackbacks', 'your-theme') : __('<span>One</span> Trackback', 'your-theme'),
$ping_count) ?></h3>

<?php /* An ordered list of our custom trackbacks callback, custom_pings(),
in functions.php   */ ?>
                    <ol>
<?php wp_list_comments('type=pings&callback=custom_pings'); ?>
                    </ol>

                </div><!-- #trackbacks-list .comments -->
```

```
<?php endif /* if ( $ping_count ) */ ?>
<?php endif /* if ( $comments ) */ ?>

<?php /* If comments are open, build the respond form */ ?>
<?php if ( 'open' == $post->comment_status ) : ?>
                <div id="respond">
                    <h3><?php comment_form_title( __('Post a Comment', 'your-
theme'), __('Post a Reply to %s', 'your-theme') ); ?></h3>

                    <div id="cancel-comment-reply"><?php
cancel_comment_reply_link() ?></div>

<?php if ( get_option('comment_registration') && !$user_ID ) : ?>
                    <p id="login-req"><?php printf(__('You must be <a
href="%s" title="Log in">logged in</a> to post a comment.', 'your-theme'),
                    get_option('siteurl') . '/wp-login.php?redirect_to=' .
get_permalink() ) ?></p>

<?php else : ?>
                    <div class="formcontainer">


                        <form id="commentform" action="<?php echo
get_option('siteurl'); ?>/wp-comments-post.php" method="post">

<?php if ( $user_ID ) : ?>
                            <p id="login"><?php printf(__('<span
class="loggedin">Logged in as <a href="%1$s" title="Logged in as
%2$s">%2$s</a>.</span> <span class="logout"><a href="%3$s" title="Log out of
this account">Log out?</a></span>', 'your-theme'),
                                get_option('siteurl') . '/wp-
admin/profile.php',
                                wp_specialchars($user_identity, true),
                                wp_logout_url(get_permalink()) ) ?></p>

<?php else : ?>

                            <p id="comment-notes"><?php _e('Your email is
<em>never</em> published nor shared.', 'your-theme') ?> <?php if ($req)
_e('Required fields are marked <span class="required">*</span>', 'your-
theme') ?></p>

            <div id="form-section-author" class="form-section">
                                <div class="form-label"><label
for="author"><?php _e('Name', 'your-theme') ?></label> <?php if ($req)
_e('<span class="required">*</span>', 'your-theme') ?></div>
                                <div class="form-input"><input id="author"
name="author" type="text" value="<?php echo $comment_author ?>" size="30"
maxlength="20" tabindex="3" /></div>
            </div><!-- #form-section-author .form-section -->

            <div id="form-section-email" class="form-section">
                                <div class="form-label"><label
for="email"><?php _e('Email', 'your-theme') ?></label> <?php if ($req)
_e('<span class="required">*</span>', 'your-theme') ?></div>
```

```
                                <div class="form-input"><input id="email"
name="email" type="text" value="<?php echo $comment_author_email ?>"
size="30" maxlength="50" tabindex="4" /></div>
                </div><!-- #form-section-email .form-section -->

                <div id="form-section-url" class="form-section">
                                <div class="form-label"><label
for="url"><?php _e('Website', 'your-theme') ?></label></div>
                                <div class="form-input"><input id="url"
name="url" type="text" value="<?php echo $comment_author_url ?>" size="30"
maxlength="50" tabindex="5" /></div>
                </div><!-- #form-section-url .form-section -->

<?php endif /* if ( $user_ID ) */ ?>

                <div id="form-section-comment" class="form-section">
                                <div class="form-label"><label
for="comment"><?php _e('Comment', 'your-theme') ?></label></div>
                                <div class="form-textarea"><textarea
id="comment" name="comment" cols="45" rows="8" tabindex="6"></textarea></div>
                </div><!-- #form-section-comment .form-section -->

                <div id="form-allowed-tags" class="form-section">
                    <p><span><?php _e('You may use these <abbr title="HyperText
Markup Language">HTML</abbr> tags and attributes:', 'your-theme') ?></span>
<code><?php echo allowed_tags(); ?></code></p>
                </div>

<?php do_action('comment_form', $post->ID); ?>

                            <div class="form-submit"><input id="submit"
name="submit" type="submit" value="<?php _e('Post Comment', 'your-theme') ?>"
tabindex="7" /><input type="hidden" name="comment_post_ID" value="<?php echo
$id; ?>" /></div>

<?php comment_id_fields(); ?>

<?php /* Just … end everything. We're done here. Close it up. */ ?>

                        </form><!-- #commentform -->
                    </div><!-- .formcontainer -->
<?php endif /* if ( get_option('comment_registration') && !$user_ID ) */ ?>
                </div><!-- #respond -->
<?php endif /* if ( 'open' == $post->comment_status ) */ ?>
            </div><!-- #comments -->
```

این کد برای شروع کد مناسبی هست و برای راحتی شما و خوانا بودن دارای توضیح است که می توانید به سلیقه خود آن را ویرایش کرده و توسعه دهید.

# فصل هشتم : پوسته ی مربوط به برگه ها و بخش جستجو



کد نویسی این برگه ها بسیار ساده است و هیچ پیچیدگی نخواهد داشت. همانند فصل های گذشته از کد زیر استفاده خواهیم کرد.

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

        <div id="container">
            <div id="content">

                <div id="nav-above" class="navigation">
                </div><!-- #nav-above -->

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                </div><!-- #post-<?php the_ID(); ?> -->

                <div id="nav-below" class="navigation">
                </div><!-- #nav-below -->


            </div><!-- #content -->
        </div><!-- #container -->

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

درست است که برای هر دو پوسته از این کد استفاده خواهیم کرد ولی آنها را ویرایش کرده و مطابق میل خود خواهیم کرد.

### پوسته مربوط به صفحه جستجو

در فایل search.php دوباره از حلقه ی وردپرس استفاده خواهیم کرد. از یک دستور شرطی استفاده می کنیم و اگر نوشته ای با نتایج جستجو مطابقت داشته باشد حلقه آنرا به نمایش خواهد گذاشت. کد مربوط به صفحه جستجو به صورت زیر خواهد شد:

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

        <div id="container">
            <div id="content">

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

<?php while ( have_posts() ) : the_post() ?>
<!-- this is our loop -->
<?php endwhile; ?>

<?php else : ?>
```

```
<!-- here's where we'll put a search form if there're no posts -->

<?php endif; ?>

            </div><!-- #content -->
        </div><!-- #container -->

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

توسط کد زیر بررسی می کنیم که با صفحه یا پست کار خواهیم کرد.

```
<?php if ( $post->post_type == 'post' ) { ?>
<?php } ?>
```

در ادامه به تکمیل بخش content می پردازیم و آنرا به صورت زیر کامل خواهیم کرد:

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

                <h1 class="page-title"><?php _e( 'Search Results for: ',
'your-theme' ); ?><span><?php the_search_query(); ?></span></h1>

<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-above" class="navigation">
                    <div class="nav-previous"><?php next_posts_link( __(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link( __(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-above -->
<?php } ?>

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

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                    <h2 class="entry-title"><a href="<?php the_permalink();
?>" title="<?php printf( __('Permalink to %s', 'your-theme'),
the_title_attribute('echo=0') ); ?>" rel="bookmark"><?php the_title();
?></a></h2>

<?php if ( $post->post_type == 'post' ) { ?>
                    <div class="entry-meta">
                        <span class="meta-prep meta-prep-author"><?php _e('By
', 'your-theme'); ?></span>
                        <span class="author vcard"><a class="url fn n"
href="<?php echo get_author_link( false, $authordata->ID, $authordata-
>user_nicename ); ?>" title="<?php printf( __( 'View all posts by %s', 'your-
theme' ), $authordata->display_name ); ?>"><?php the_author(); ?></a></span>
                        <span class="meta-sep"> | </span>
                        <span class="meta-prep meta-prep-entry-date"><?php
_e('Published ', 'your-theme'); ?></span>
                        <span class="entry-date"><abbr class="published"
title="<?php the_time('Y-m-d\TH:i:sO') ?>"><?php the_time( get_option(
'date_format' ) ); ?></abbr></span>
```

```
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t" ) ?>
                    </div><!-- .entry-meta -->
<?php } ?>

                    <div class="entry-summary">
<?php the_excerpt( __( 'Continue reading <span class="meta-
nav">&raquo;</span>', 'your-theme' )  ); ?>
<?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>
                    </div><!-- .entry-summary -->

<?php if ( $post->post_type == 'post' ) { ?>
                    <div class="entry-utility">
                        <span class="cat-links"><span class="entry-utility-
prep entry-utility-prep-cat-links"><?php _e( 'Posted in ', 'your-theme' );
?></span><?php echo get_the_category_list(', '); ?></span>
                        <span class="meta-sep"> | </span>
                        <?php the_tags( '<span class="tag-links"><span
class="entry-utility-prep entry-utility-prep-tag-links">' . __('Tagged ',
'your-theme' ) . '</span>', ", ", "</span>\n\t\t\t\t\t\t\t<span class=\"meta-
sep\">|</span>\n" ) ?>
                        <span class="comments-link"><?php
comments_popup_link( __( 'Leave a comment', 'your-theme' ), __( '1 Comment',
'your-theme' ), __( '% Comments', 'your-theme' ) ) ?></span>
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t\n" ) ?>
                    </div><!-- #entry-utility -->
<?php } ?>
                </div><!-- #post-<?php the_ID(); ?> -->

<?php endwhile; ?>

<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-below" class="navigation">
                    <div class="nav-previous"><?php next_posts_link( __(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link( __(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-below -->
<?php } ?>

<?php else : ?>

                <div id="post-0" class="post no-results not-found">
                    <h2 class="entry-title"><?php _e( 'Nothing Found', 'your-
theme' ) ?></h2>
                    <div class="entry-content">
                        <p><?php _e( 'Sorry, but nothing matched your search
criteria. Please try again with some different keywords.', 'your-theme' );
?></p>
    <?php get_search_form(); ?>
                    </div><!-- .entry-content -->
```

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

## پوسته برگه ها

جدا از نوشته های مربوط به  بلاگ ها بخشی در وردپرس وجود دارد که به برگه ها یا صفحه ها اختصاص پیدا می کند.  معمولا در برگه ها بخش نظردهی وجود ندارد و اصلا دلیلی هم برای وجود نظردهی در برگه های وبلاگ یا سایت وجود ندارد. ما به صورت پیشفرض نظر ها را در برگه هامان نمایش نخواهیم داد ولی برای افرادی که می خواهند زمینه های دلخواه[11] قرار دهند فضای مناسب را خواهیم ساخت و مقدار آنرا commens قرار می دهیم تا هر کسی مایل بود آن را بکار گیرد. نظرتان در مورد این روش چیست؟ هم شرایط استاندارد را در نظر گرفته ایم و هم اینکه برای کاربرانی که نیاز به نظردهی در برگه ها دارند روشی در نظر گرفته ایم. یک پوسته حرفه ای فکر تمامی کاربران را می کند.

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

        <div id="container">
            <div id="content">

<?php the_post(); ?>

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                    <h1 class="entry-title"><?php the_title(); ?></h1>
                    <div class="entry-content">
<?php the_content(); ?>
<?php wp_link_pages('before=<div class="page-link">' . __( 'Pages:', 'your-
theme' ) . '&after=</div>') ?>
<?php edit_post_link( __( 'Edit', 'your-theme' ), '<span class="edit-link">',
'</span>' ) ?>
                    </div><!-- .entry-content -->
                </div><!-- #post-<?php the_ID(); ?> -->

<?php if ( get_post_custom_values('comments') ) comments_template() // Add a
custom field with Name and Value of "comments" to enable comments on this
page ?>
            </div><!-- #content -->
        </div><!-- #container -->
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

[11] - Custom Field

# فصل نهم – آرشیو، نویسنده، دسته بندی و برچسب های پوسته

۹

تا اینجا چطور پیشرفته اید؟ توجه داشته باشید هر فصل را به دقت و قدم به قدم جلو ببرید و اگر به مشکلی خوردید آنرا در گوگل جستجو کنید یا برای من ایمیل بزنید. حالا سراغ کامل تر کردن امکانات پوسته می رویم.

کار فایل archive.php چیست؟ این فایل بر اساس خواست ما، تمامی پست ها را بایگانی شده نمایش می دهد. آرشیو می تواند بر اساس تاریخ ، نویسنده ، دسته بندی و یا برچسب ها ی مرتبط مرتب و تهیه شود. در واقع این فایل خیلی شبیه فایل index.php است.

همانند قبل با کد مثال خود شروع می کنیم و شروع به ویرایش آن خواهیم کرد.

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

        <div id="container">
            <div id="content">

                <div id="nav-above" class="navigation">
                </div><!-- #nav-above -->

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                </div><!-- #post-<?php the_ID(); ?> -->

                <div id="nav-below" class="navigation">
                </div><!-- #nav-below -->
            </div><!-- #content -->
        </div><!-- #container -->
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

در ادامه چه روندی را برای فایل آرشیو طی خواهیم کرد؟

- تابع the_post() را فراخوانی خواهیم کرد
- کنترل که پوسته چگونه عمل کند
- پوسته مناسب با کار تهیه می کنیم
- مرور کردن نوشته ها با کمک تابع rewind_posts()
- استفاده از حلقه معروف وردپرس

طبق معمول با کمک دستورات شرطی همه ی موارد را کنترل می کنیم و کدمان به شکل زیر خواهد شد:

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

<?php if ( is_day() ) : ?>
                <h1 class="page-title"><?php printf( __( 'Daily Archives:
<span>%s</span>', 'your-theme' ), get_the_time(get_option('date_format')) )
?></h1>
```

```
<?php elseif ( is_month() ) : ?>
                <h1 class="page-title"><?php printf( __( 'Monthly Archives:
<span>%s</span>', 'your-theme' ), get_the_time('F Y') ) ?></h1>
<?php elseif ( is_year() ) : ?>
                <h1 class="page-title"><?php printf( __( 'Yearly Archives:
<span>%s</span>', 'your-theme' ), get_the_time('Y') ) ?></h1>
<?php elseif ( isset($_GET['paged']) && !empty($_GET['paged']) ) : ?>
                <h1 class="page-title"><?php _e( 'Blog Archives', 'your-
theme' ) ?></h1>
<?php endif; ?>

<?php rewind_posts(); ?>

<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-above" class="navigation">
                    <div class="nav-previous"><?php next_posts_link(__(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link(__(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-above -->
<?php } ?>

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

                <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                    <h2 class="entry-title"><a href="<?php the_permalink();
?>" title="<?php printf( __('Permalink to %s', 'your-theme'),
the_title_attribute('echo=0') ); ?>" rel="bookmark"><?php the_title();
?></a></h2>

                    <div class="entry-meta">
                        <span class="meta-prep meta-prep-author"><?php _e('By
', 'your-theme'); ?></span>
                        <span class="author vcard"><a class="url fn n"
href="<?php echo get_author_link( false, $authordata->ID, $authordata-
>user_nicename ); ?>" title="<?php printf( __( 'View all posts by %s', 'your-
theme' ), $authordata->display_name ); ?>"><?php the_author(); ?></a></span>
                        <span class="meta-sep"> | </span>
                        <span class="meta-prep meta-prep-entry-date"><?php
_e('Published ', 'your-theme'); ?></span>
                        <span class="entry-date"><abbr class="published"
title="<?php the_time('Y-m-d\TH:i:sO') ?>"><?php the_time( get_option(
'date_format' ) ); ?></abbr></span>
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t" ) ?>
                    </div><!-- .entry-meta -->

                    <div class="entry-summary">
<?php the_excerpt( __( 'Continue reading <span class="meta-
nav">&raquo;</span>', 'your-theme' )  ); ?>
                    </div><!-- .entry-summary -->

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

```
                        <span class="cat-links"><span class="entry-utility-
prep entry-utility-prep-cat-links"><?php _e( 'Posted in ', 'your-theme' );
?></span><?php echo get_the_category_list(', '); ?></span>
                        <span class="meta-sep"> | </span>
                        <?php the_tags( '<span class="tag-links"><span
class="entry-utility-prep entry-utility-prep-tag-links">' . __('Tagged ',
'your-theme' ) . '</span>', ", ", "</span>\n\t\t\t\t\t\t\t<span class=\"meta-
sep\">|</span>\n" ) ?>
                        <span class="comments-link"><?php
comments_popup_link( __( 'Leave a comment', 'your-theme' ), __( '1 Comment',
'your-theme' ), __( '% Comments', 'your-theme' ) ) ?></span>
                        <?php edit_post_link( __( 'Edit', 'your-theme' ),
"<span class=\"meta-sep\">|</span>\n\t\t\t\t\t\t\t<span class=\"edit-link\">",
"</span>\n\t\t\t\t\t\t\n" ) ?>
                    </div><!-- #entry-utility -->
                </div><!-- #post-<?php the_ID(); ?> -->

<?php endwhile; ?>

<?php global $wp_query; $total_pages = $wp_query->max_num_pages; if (
$total_pages > 1 ) { ?>
                <div id="nav-below" class="navigation">
                    <div class="nav-previous"><?php next_posts_link( __(
'<span class="meta-nav">&laquo;</span> Older posts', 'your-theme' )) ?></div>
                    <div class="nav-next"><?php previous_posts_link( __(
'Newer posts <span class="meta-nav">&raquo;</span>', 'your-theme' )) ?></div>
                </div><!-- #nav-below -->
<?php } ?>
```

در این مرحله سراغ پوسته مربوط به نویسنده مطالب می رویم. این بار کارهایی را که قبلا کردیم تکرار نخواهیم کرد. فایل archive.php را کپی کنید و نام آنرا به author.php تغییر دهید. حال تنها چیزی که نیاز داریم تغییر عناوین بالای صفحه هست. آسان بود مگه نه ؟

```
<h1 class="page-title author"><?php printf( __( 'Author Archives: <span
class="vcard">%s</span>', 'your-theme' ), "<a class='url fn n'
href='$authordata->user_url' title='$authordata->display_name'
rel='me'>$authordata->display_name</a>" ) ?></h1>
<?php $authordesc = $authordata->user_description; if ( !empty($authordesc) )
echo apply_filters( 'archive_meta', '<div class="archive-meta">' .
$authordesc . '</div>' ); ?>
```

نوبتی هم باشد نوبت پوسته مربوط به دسته ها[12] است. نیازی به دوباره کاری نیست. فایل archive.php را کپی کنید و آنرا به category.php تغییر نام دهید. فایل functions.php را باز نمایید و کد زیر را در آن درج کنید تا کمی پوسته خود را مفید تر نماییم.

```
// For category lists on category archives: Returns other categories except
the current one (redundant)
function cats_meow($glue) {
    $current_cat = single_cat_title( '', false );
    $separator = "\n";
    $cats = explode( $separator, get_the_category_list($separator) );
    foreach ( $cats as $i => $str ) {
        if ( strstr( $str, ">$current_cat<" ) ) {
            unset($cats[$i]);
            break;
        }
    }
    if ( empty($cats) )
        return false;

    return trim(join( $glue, $cats ));
} // end cats_meow
```

تابع اختصاصی ما cats_meow() است که کارش حذف و عدم نمایش دسته ی جاری که کاربر در آن قرار دارد از برگه های مربوط است. در اصل از شر دسته بندی های تکراری و مطالب اضافی راحت خواهیم شد.

به فایل category.php برمی گردیم و کد عناوین و بالایی را به صورت زیر ویرایش می نماییم:

```
<h1 class="page-title"><?php _e( 'Category Archives:', 'your-theme' ) ?>
<span><?php single_cat_title() ?></span></span></h1>
<?php $categorydesc = category_description(); if ( !empty($categorydesc) )
echo apply_filters( 'archive_meta', '<div class="archive-meta">' .
$categorydesc . '</div>' ); ?>
```

و در بخش entry-utility. کد زیر را جایگزین می نماییم:

```
<span class="cat-links"><span class="entry-utility-prep entry-utility-prep-
cat-links"><?php _e( 'Posted in ', 'your-theme' ); ?></span><?php echo
get_the_category_list(', '); ?></span>
```

و همین طور :

```
<?php if ( $cats_meow = cats_meow(', ') ) : // Returns categories other than
the one queried ?>
                        <span class="cat-links"><?php printf( __( 'Also
posted in %s', 'your-theme' ), $cats_meow ) ?></span>
```

[12] - Category

```
                        <span class="meta-sep"> | </span>
<?php endif ?>
```

و آخرین بخشی که در این فصل به آن می پردازیم پوسته مربوط به برچسب های مطالب است. فایل archive.php را کپی نمایید و نام آنرا به tag.php تغییر دهید. در نظر داریم تابعی جدید با نام tag_ur_it() در فایل functions.php بنویسیم. کار این تابع مشابه کار تابعی است که با نام cats_meow() برای دسته بندی ها نوشتیم.

```
// For tag lists on tag archives: Returns other tags except the current one
(redundant)
function tag_ur_it($glue) {
    $current_tag = single_tag_title( '', '',  false );
    $separator = "\n";
    $tags = explode( $separator, get_the_tag_list( "", "$separator", "" ) );
    foreach ( $tags as $i => $str ) {
        if ( strstr( $str, ">$current_tag<" ) ) {
            unset($tags[$i]);
            break;
        }
    }
    if ( empty($tags) )
        return false;

    return trim(join( $glue, $tags ));
} // end tag_ur_it
```

اکنون فایل tag.php را در ویرایش گر متنی خود باز می کنیم و عنوان صفحه را با کد زیر جایگزین می کنیم :

```
<h1 class="page-title"><?php _e( 'Tag Archives:', 'your-theme' ) ?>
<span><?php single_tag_title() ?></span></h1>
```

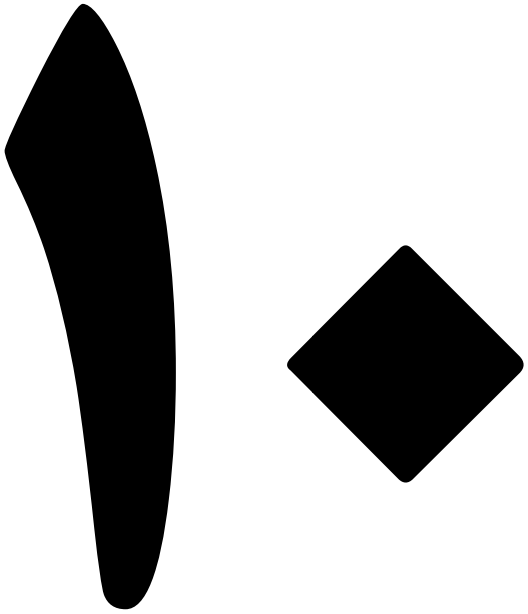
و البته در بخش entry-utility. نیز کد زیر را جایگزین کدهای قبل می کنیم :

```
<?php the_tags( '<span class="tag-links"><span class="entry-utility-prep
entry-utility-prep-tag-links">' . __('Tagged ', 'your-theme' ) . '</span>',
", ", "</span>\n\t\t\t\t\t\t\t<span class=\"meta-sep\">|</span>\n" ) ?>
```

و در قدم آخر

```
<?php if ( $tag_ur_it = tag_ur_it(', ') ) : // Returns tags other than the
one queried ?>
                        <span class="tag-links"><?php printf( __( 'Also
tagged %s', 'your-theme' ), $tag_ur_it ) ?></span>
<?php endif; ?>
```

# فصل دهم : پوسته مربوط به ستون کناری



نمی توان از مزیت ها و کاربردهای ستون کناری[13] به سادگی گذشت. داشتن یک ستون کناری قوی در پوسته های وردپرس نعمتی بزرگ است. البته تعداد ستون ها بستگی به خودمان و نوع پوسته ای که روی آن کار می کنیم دارد.

در ادامه تابعی اختصاصی برای ستون کناری خود خواهیم ساخت. البته یکی از نکاتی که همیشه باید توجه کنید و خیلی مهم است کارکرد پوسته و ستون کناری شما و سازگار بودن آن با ابزارک ها است.

فایل functions.php را باز می کنیم و با کمک کد زیر منطقه قابل دسترسی توسط ابزارک ها را کدنویسی می کنیم.

```
// Register widgetized areas
function theme_widgets_init() {
    // Area 1
    register_sidebar( array (
    'name' => 'Primary Widget Area',
    'id' => 'primary_widget_area',
    'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
    'after_widget' => "</li>",
    'before_title' => '<h3 class="widget-title">',
    'after_title' => '</h3>',
  ) );

    // Area 2
    register_sidebar( array (
    'name' => 'Secondary Widget Area',
    'id' => 'secondary_widget_area',
    'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
    'after_widget' => "</li>",
    'before_title' => '<h3 class="widget-title">',
    'after_title' => '</h3>',
  ) );
} // end theme_widgets_init
add_action( 'init', 'theme_widgets_init' );
```

با کمک تابع فوق دو منطقه برای ابزارک ها تهیه کردیم. فایل functions.php را نبندید، هنوز کارمان تمام نشده است. در ادامه باید ابزارک های پیشفرض را تعریف نماییم. ابزارک هایی مانند : جستجو ، برگه ها ، دسته بندی ها ، آرشیو و لینک ها که به صورت ابزارک در وردپرس ارایه می شوند. برای این کار نیازی به کدنویسی هر کدام نیست و تنها کافی است به وردپرس بگوییم کدام ابزارک ها را فراخوانی کند و در بخش مورد نظر ما بارگذاری نماید.

```
$preset_widgets = array (
```

[13] - Sidebar

```
    'primary_widget_area'  => array( 'search', 'pages', 'categories',
'archives' ),
    'secondary_widget_area'  => array( 'links', 'meta' )
);
if ( isset( $_GET['activated'] ) ) {
    update_option( 'sidebars_widgets', $preset_widgets );
}
// update_option( 'sidebars_widgets', NULL );
```

بعد از نوشتن این کد ابزارک های جستجو ، برگه ها ، دسته بندی ها و آرشیو در اولین منطقه ابزارک های اصلی به صورت پیشفرض بارگذاری خواهد شد.

در منطقه ی دوم مربوط به ابزارک ها لینک ها و متا تگها را فراخوانی کرده ایم. در آخرین خط کدی را می بینید که به صورت توضیح نوشته ام ، اگر به هر دلیلی خواستید تمام ابزارک ها را غیب و غیر فعال کنید تنها کافی است // را پاک کنید.

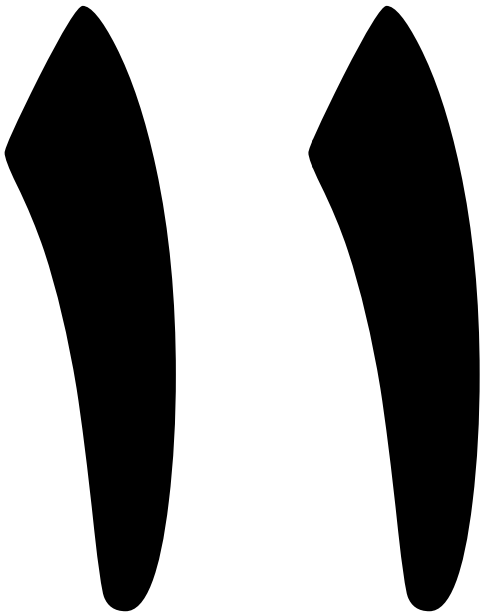
الان نوبت نوشتن کدی هست که کنترل نماید آیا ابزارک ها در بخش مدیریت تغییر کرده اند یا خیر ، اگر تغییری ایجاد شده است باید تغییرات اعمال شود.

```
// Check for static widgets in widget-ready areas
function is_sidebar_active( $index ){
  global $wp_registered_sidebars;
  $widgetcolums = wp_get_sidebars_widgets();
  if ($widgetcolums[$index]) return true;
    return false;
} // end is_sidebar_active
```

دیگر کار زیادی برای ستون کناری ها نمانده است. ۲ منطقه پویا برای ابزارک ها ثبت کرده ایم و ابزارک های پیشفرض را نیز تعریف نموده ایم.

```
<?php if ( is_sidebar_active('primary_widget_area') ) : ?>
        <div id="primary" class="widget-area">
            <ul class="xoxo">
                <?php dynamic_sidebar('primary_widget_area'); ?>
            </ul>
        </div><!-- #primary .widget-area -->
<?php endif; ?>

<?php if ( is_sidebar_active('secondary_widget_area') ) : ?>
        <div id="secondary" class="widget-area">
            <ul class="xoxo">
                <?php dynamic_sidebar('secondary_widget_area'); ?>
            </ul>
        </div><!-- #secondary .widget-area -->
<?php endif; ?>
```

# فصل یازدهم : کدهای CSS



آخرین فصل کتاب به CSS ها مرتبط می شود. با اینکه CSS ها مبحث ساده ای هستند ولی ساده گرفتن آنها و توجه نکردن به آنها می توانند درد سر ساز شوند.

کار ها و مراحلی که باید طی کنیم به شرح زیر است :

- نیاز به فایلی CSS ای داریم که برای بهتر کار کردن پوسته تمامی تنظیمات پیشفرض را ریست نماید
- فایلی دیگر برای کلاس های وردپرس
- ۶ فایل CSS که تمامی ظاهر و بخش های مختلف سایت و پروژه ما را تشکیل می دهد

تمامی کدهایی که در بالا ذکر شد به صورت رایگان و آزاد در اختیار شما قرار خواهند گرفت. تنها کافی است به لینک زیر مراجعه نمایید :

http://code.google.com/p/your-wordpress-theme/source/browse/#svn/trunk/styles

فایل ها را باز کنید و کد ها را در پروژه ی خود کپی کنید. اولین قدم ساخت پوشه ای به نام style در پوشه ی پوسته شما است. در این پوشه شما باید تمامی فایل های CSS خود را منتقل کنید.

## فایل ریست

کاری که این فایل می کند تمامی تنظیمات و مشخصات را یکسان می سازد تا نتیجه خروجی ما در همه ی مرورگرها یکسان شود. استفاده از این فایل خیلی ساده است تنها کد زیر را در بالای فایل style.css اضافه می کنیم تا فایل مربوط را فراخوانی نماید.

```
/* Reset default browser styles */

@import url('styles/reset.css');
```

بخش هایی در وردپرس هستند که همیشه ثابت هستند ونیازی به ویرایش یا تغییرات زیاد و از صفر نوشتن ندارند. این کدها و استایل ها در فایل wp.css نوشته و قرار گرفته اند. بکار گیری این فایل نیز اصلا سخت نیست. کدهای زیر این کار را انجام می دهند.

```
/* Basic WordPress Styles */

@import url('styles/wp.css');
```

از آنجایی که css ها جزو مباحثی هستند که با آن آشنا هستید می توانید با مرور کدها اطلاعات لازم را دریافت نمایید.

در پایان امیدوارم این کتاب برایتان مفید بوده باشد.

موفق باشید

CODE IS POETRY