主题开发

官方文档： https://developer.wordpress.org/themes/getting-started/
模仿： https://xingpingcn.top/
参考文章： https://www.zhuige.com/index.php/news/cat/17.html

工具：vscode; phpstudy; edge浏览器
插件：php debug ; live server ; chinese中文简体
技术：html+css+js+php

准备：

下载代码： https://cn.wordpress.org/download/
打开调试模式

```
define('WP_DEBUG', true);
```

1.创建主题文件夹

在这个目录里 wp-content/themes，新建一个文件夹，名称随意

2.新建文件 style. css

通过注释来配置主题的基本信息，
包括主题名称、描述、作者和版本号

```
/*
Theme Name: 猫叔的主题01
Description: 一个超级简单的入门主题 01
Author: 写代码的猫叔
Version: 1.0
*/
```

3.新建index.php

主题的入口文件，用于显示网站的内容

```php
<?php
get_header(); // 获取头部
?>

<main>
  <h1>马上就要2024年了！ </h1>
  <h2>这里是二级标题</h2>
  <p>欢迎跟猫叔一起学习wordpress的开发.</p>
</main>

<?php
get_footer(); // 获取尾部
?>
```

主题文件

style.css：主题的样式文件，包含主题的元数据和样式信息。
index.php：主题的入口文件，用于显示默认的内容。
header.php：网站的头部部分，包括导航菜单和页眉内容。
footer.php：网站的尾部部分，包括页脚信息。

single.php：用于显示单篇文章的模板文件。
page.php：用于显示页面的模板文件。

functions.php：主题的功能文件，可以在其中添加自定义功能和钩子。
assets文件夹：存放CSS、JavaScript和图像等资源的文件夹。

模板标签（内置函数）

主题中使用模板标签来显示不同类型的内容。

引入文件

```php
get_header();//获取头部部分。

get_footer();//获取尾部部分。

get_sidebar();//侧边栏
```

文章相关

```
have_posts();
the_post();

the_title();//显示文章或页面的标题。

the_content();//显示文章或页面的内容。

the_excerpt();//显示文章的摘要。

get_permalink(); //获取文章的链接

get_the_author(); //文章作者

get_the_date('Y-m-d');//日期
the_date('Y-m-d','<span class="date">日期: ','</span>');

the_post_thumbnail();//显示文章的特色图片。

get_template_part();//获取模板
//如果传了name 他会自动去找这个模板 "{$slug}-{$name}.php"
get_template_part( $slug, $name = null, $args = array() )
```

```
wp_nav_menu();//显示导航菜单。

get_search_form();//搜索框

bloginfo( 'name' );//显示网站名称
bloginfo( 'version' );//显示wp版本

home_url();//首页链接
home_url('/test');

wp_head();
wp_footer();
```

用户相关

```
//获取当前用户（已弃用）
get_currentuserinfo();
//获取当前用户（推荐使用）
wp_get_current_user();


//根据邮箱获取用户头像
get_avatar( get_the_author_email(), '60' );


//获取当前用户的头像
global $current_user;

echo get_avatar( $current_user->user_email, 32);
```

输出文章列表

```
while (have_posts()) {
        the_post();
    the_title();//输出文章标题
    the_excerpt(); //输出文章摘要
    the_author();//输出作者名称
    echo get_the_time('Y-m-d G:i:s');//输出文章发布时间
    echo get_permalink(); //输出文章的链接
    echo get_author_posts_url(get_the_author_meta('ID'));//根据作者ID输出作者页面的链接
}

//使用模板后：
 while(have_posts()){
          the_post();
          get_template_part('templates/cons');

}
```

```
/**
 * 将摘要的"[...]" 替换为“阅读更多”
 *
 */
```

```php
function replace_excerpt_more( $link ) {
    if ( is_admin() ) {
        return $link;
    }

    $link = sprintf(
        '<p class="link-more"><a href="%1$s" class="more-link ">%2$s</a>
</p>',
        esc_url( get_permalink( get_the_ID() ) ),
        '阅读更多'
    );
    return ' &hellip; ' . $link;
}
add_filter( 'excerpt_more', 'replace_excerpt_more' );
```

列表分页

导入测试数据

> https://github.com/WordPress/theme-test-data
> 下载文件：themeunittestdata.wordpress.xml

```php
the_posts_pagination( array(
      'mid_size' => 2, //当前页码数的 两边 显示几个页码。
      'prev_text' =>'上一页', //上一页
      'next_text' =>'下一页', //下一页
      'screen_reader_text'=>' '
      ) );

//paginate_links()

$total = 100; // 文章页码总数
$current = 5; // 当前页面的文章索引

// 使用paginate_links函数生成分页链接
$pagination_links = paginate_links( array(
    'total' => $total,
    'current' => $current,
    'base' => get_pagenum() ? get_pagenum_link() : '',
    'format' => '/page/%#%',
```

```
    'prev' => '<span class="nav-prev">«</span>',
    'next' => '<span class="nav-next">»</span>',
    'dot' => '...',
) );

// 将分页链接输出到主题中
echo $pagination_links;
```

导航菜单

functions（后台）

```
//注册菜单分类
register_nav_menus(array(
    'primary' => '主导航菜单',
    'footer' => '页脚菜单',
));

//下面这个可以单独使用，是开启菜单工具的，上面的函数内就带着这句一起执行了
add_theme_support( 'menus' );
```

前台页面

```
echo wp_nav_menu( array(
                    'container' => 'div',//容器标签
                    'container_class' => 'navbar-box',//ul父节点class值
                    'container_id' => 'nav-bar',//ul父节点id值
                    'theme_location' => 'headerMenu',//导航别名
                    'items_wrap' => '<ul class="navbar-nav">%3$s</ul>', //
包装列表
                    ) );
```

搜索框

```
get_search_form();//搜索框

//或者:
get_search_form([
```

```
        "echo"=>true,
        "aria_label"=>"aria_label测试文字"
]);

//获取主题路径
get_template_directory_uri()
```

搜索结果页

文件：search. php

```
//获取搜索的内容:
get_search_query()
```

小工具（侧边栏）

```
//启用小工具
add_theme_support( 'widgets' );

// functions 自定义侧边栏
function my_custom_sidebar() {
    register_sidebar(
        array (
            'name' => '测试侧边栏',//侧边栏名称
            'id' => 'test-side-bar',//侧边栏ID
            'description' => '这里是侧边栏的描述',//侧边栏描述
            'before_widget' => '<div class="widget-content">',//侧边栏前面
的代码
            'after_widget' => "</div>",//侧边栏后面的代码
            'before_title' => '<h3 class="widget-title">',//侧边栏标题的前
面的代码
            'after_title' => '</h3>',//侧边栏标题的后面的代码
        )
    );

        //可同时注册多个小工具
        register_sidebar(
        array (
            'name' => '测试侧边栏2',//侧边栏名称
```

```
            'id' => 'test-side-bar2',//侧边栏ID
            'description' => '这里是侧边栏的描述',//侧边栏描述
            'before_widget' => '<div class="widget-content2">',//侧边栏前面
的代码
            'after_widget' => "</div>",//侧边栏后面的代码
            'before_title' => '<h3 class="widget-title2">',//侧边栏标题的前
面的代码
            'after_title' => '</h3>',//侧边栏标题的后面的代码
        )
    );
}
add_action( 'widgets_init', 'my_custom_sidebar' );
```

```
//写在调用侧边栏的页面内
get_sidebar();
//或者 指定名称
get_sidebar('test2');

//写在对应的侧边栏里面（sidebar.php）
<?php dynamic_sidebar( 'test-side-bar' ); ?>

//例如：
<div class="sider-bar">
    <p>我是侧边栏内容1</p>
    <?php dynamic_sidebar( 'test-side-bar' ); ?>
</div>
```

详情页

single.php

显示文章内容和作者信息等

```
        the_post();
        the_title('<h1>','</h1>');
        the_author();//输出作者名称
        echo get_the_time('Y-m-d H:i:s');//输出文章发布时间
```

```
        the_content();
        // echo get_author_posts_url(get_the_author_meta('ID'));
```

文章推荐

```
the_post_navigation(
        array(
            'prev_text' => '<span class="nav-subtitle">上一篇：</span>
<span class="nav-title">%title</span>',
            'next_text' => '<span class="nav-subtitle">下一篇：</span>
<span class="nav-title">%title</span>',
            )
    );
```

评论

评论内容

1.在详情页内调用

```
//显示文章评论（获取读取模板）
comments_template();
```

2. 对应的评论模板： comments.php (默认，可以改为其他文件名)

```
//评论数量
<h4>共 <?php echo get_comments_number();?> 条评论</h4>

//评论列表内容
<?php wp_list_comments(); ?>

//上一页 下一页
<?php the_comments_navigation();?>

//评论输入框
<?php comment_form(); ?>
```

评论列表模板：

在上面的模板内：

```
wp_list_comments(
    array(
        'callback'          => 'custom_comment',//回调函数
    )
);
```

```
-- walker，用于列出评论的Walker类的实例。默认为null。
-- max_depth，评论嵌套最大层级。
-- style，列表的样式。接受“ul”、“ol”或“div”。“div”将不会产生额外的列表标记。默认值“ul”。
-- callback，要使用的回调函数，通过回调函数来自定义你的评论展示方式。默认为null。
-- end-callback，要在末尾使用的回调函数。默认为null。
-- type，要列出的评论的类型。接受“all”、“comment”、“pingback”、“trackback”、“ping”。默认为“all”。
-- page，要列出其评论的页面ID。
-- per_page，每页要列出的评论数。
-- avatar_size，头像大小的高度和宽度尺寸。默认值32
-- rerverse_top_level，评论数据是否倒序显示，默认为null。
-- reverse_children，子评论数据是否倒序显示。默认为null。
-- format，如何设置评论列表的格式。接受“html5”、“xhtml”。如果主题支持，则默认为“html5”。
-- short_ping，是否输出短信号。默认值为false。
-- echo，是回显输出还是返回输出。默认值为true。
```

回调函数：

```
//获取评论者和评论者id，返回是一个a标签的字符串
get_comment_author_link();

//获取评论者名字
get_comment_author();
```

```php
//获取头像
get_avatar($comment,'48');

//获取评论时间
get_comment_time('Y-m-d H:i:s')

// 显示评论的编辑链接
edit_comment_link( '编辑', '<p class="edit-link">', '</p>' );

// 显示评论的回复链接
comment_reply_link( array_merge( $args, array(
'reply_text' =>  '回复',
'after'      =>  ' <span>&darr;</span>',
'depth'      =>  $depth,
'max_depth'  =>  $args['max_depth'] ) ) )
);

//评论分页（在后台可设置）
//获取当前评论列表页码
$cpage = get_query_var('cpage');
//获取每页评论显示数量
$cpp=get_option('comments_per_page');
```

```php
//评论模板
function custom_comment($comment, $args, $depth){
    $GLOBALS['comment'] = $comment;
    $ava = get_avatar($comment,'48');
    $author_link = get_comment_author_link();
    // $author = get_comment_author();
    // echo $author;

    echo '
    <li class="comment-list">
        <div class="avatar">
        '.$ava.'
        </div>
        <div class="comment_content">
            <div class="comment_author">
```

```
                '.$author_link.'
            </div>

            <div class="comment_time">
            '.get_comment_time('Y-m-d H:i:s').'
            </div>

            '.comment_text().'
        ';
        ?>
            <div class="edit-line">
                <?php
                    // 显示评论的编辑链接
                    edit_comment_link( '编辑', '<p class="edit-link">',
'</p>' );
                ?>
                <div class="reply">
                    <?php
                        // 显示评论的回复链接
                        comment_reply_link( array_merge( $args, array(
                        'reply_text' =>  '回复',
                        'after'      =>  ' <span>&darr;</span>',
                        'depth'      =>  $depth,
                        'max_depth'  =>  $args['max_depth'] ) ) );
                    ?>
                </div>
            </div>
        </div>
    </li>
    <?php
}
```

解决评论头像不显示

网址:
https://cravatar.com/developer/for-wordpress

```
if ( ! function_exists( 'get_cravatar_url' ) ) {
    /**
```

```
     * 替换 Gravatar 头像为 Cravatar 头像
     *
     * Cravatar 是 Gravatar 在中国的完美替代方案，您可以在
https://cravatar.com 更新您的头像
     */
    function get_cravatar_url( $url ) {
        $sources = array(
            'www.gravatar.com',
            '0.gravatar.com',
            '1.gravatar.com',
            '2.gravatar.com',
            'secure.gravatar.com',
            'cn.gravatar.com',
            'gravatar.com',
        );
        return str_replace( $sources, 'cravatar.cn', $url );
    }
    add_filter( 'um_user_avatar_url_filter', 'get_cravatar_url', 1 );
    add_filter( 'bp_gravatar_url', 'get_cravatar_url', 1 );
    add_filter( 'get_avatar_url', 'get_cravatar_url', 1 );
}
if ( ! function_exists( 'set_defaults_for_cravatar' ) ) {
    /**
     * 替换 WordPress 讨论设置中的默认头像
     */
    function set_defaults_for_cravatar( $avatar_defaults ) {
        $avatar_defaults['gravatar_default'] = 'Cravatar 标志';
        return $avatar_defaults;
    }
    add_filter( 'avatar_defaults', 'set_defaults_for_cravatar', 1 );
}
if ( ! function_exists( 'set_user_profile_picture_for_cravatar' ) ) {
    /**
     * 替换个人资料卡中的头像上传地址
     */
    function set_user_profile_picture_for_cravatar() {
        return '<a href="https://cravatar.com" target="_blank">您可以在
Cravatar 修改您的资料图片</a>';
    }
    add_filter( 'user_profile_picture_description',
'set_user_profile_picture_for_cravatar', 1 );
```

```
}
```

发表评论的表单

```
comment_form();

//判断用户是否已登录
is_user_logged_in()

//是否开启了邮箱和名称必填项
get_option( 'require_name_email' )
//是否启用了分页
get_option( 'page_comments' )
//是否开启了评论嵌套
get_option( 'thread_comments' )
//获取站点链接
get_option( 'siteurl' );// 和这个效果一样： home_url(); site_url();
```

函数介绍

comment_form($args = array(), $post = null)

$args:
array数组，要覆盖的默认参数和表单字段。

`fields`(array) :修改未登录用户显示的字段样式，将以html的样式显示在页面里。
注意它是数组，数组里可选填重新定义4个字段，注意里面的name不能随便改；
另外还可以增加其他字段

`author`(string):评论者 HTML。
`email`(string):邮箱 HTML。
`url`(string):评论者URL HTML。
`cookies`(string):是否存储cookies的勾选框HTML

`comment_field`(string): textarea输入框的HTML。

`must_log_in`(string):“必须登录才能发表评论”提示文字的HTML。

`logged_in_as`(string): `以[用户]身份登录`消息的HTML元素。
`comment_notes_before`(string):如果用户未登录，则在重新定义字段之前显示的消息的HTML元素。默认值为`不会发布您的电子邮件地址`。
`comment_notes_after`(string) :HTML元素，用于在textarea字段之后显示的消息。
`action`(string):重新定义表单元素的action属性。默认为`site_url( `/wp-comments-post.php` )`。
`id_form`(string):重新定义表单元素的id属性。默认为` commentform`。
`id_submit`(string):重新定义提交元素的id属性。默认为`submit`。

`class_container`(string):重新定义表单的包裹容器的class。默认为`comment-respond`
`class_form`(string):重新定义表单元素的类属性。默认为`comment-form`。
`class_submit`(string):重新定义提交元素类的属性。默认为`submit`。
`name_submit`(string):重新定义提交元素名称属性。默认为`submit`。
`title_reply`(string):可翻译的`reply`按钮标签。默认为`发表回复`。
`title_reply_to`(string):可翻译的`回复`按钮标签。默认为`留下对%s的回复`，其中%s是要回复的评论的作者。
`title_reply_before`(string):在评论表单标题之前显示的HTML。默认值:<h3 id="reply-title" class="comment-reply-title">。
` title_reply_after`(string):在评论表单标题之后显示的HTML。默认值:< /h3>。
` cancel_reply_before`(string):在取消回复链接之前显示的HTML。默认为

`<small>`。
` cancel_reply_after`(string):取消回复链接后显示的HTML。默认为`</small>`。
`cancel_reply_link`(string):可翻译的`取消回复`按钮标签。默认为`取消回复`。
`label_submit`(string):可翻译的`submit`按钮标签。默认为`发表评论`。
` submit_button`(string): `提交`按钮的HTML格式。默认值:<input name="%1$s" type="submit" id="%2$s" class="%3$s" value="%4$s" />。
` submit_field`(string): HTML格式，用于围绕`提交`按钮和注释隐藏字段的标记。默认值: %1$s %2$s 。其中%1 $s是提交按钮标记，%2 $s是注释隐藏字段。
`format`(string):注释表单格式。默认为`xhtml`。接受` xhtml`, ` html5`。

$post

$ post_id(int | WP_Post) (可选) :要为其生成表单的Post ID或WP_Post对象。默认值:null

文章详情分页

如果文章插入了分页符，那么就可以实现分页功能，用下面的函数即可

```
wp_link_pages()
```

文章所属分类

注意：文章是可以同时所属多个分类的

```
//获取当前文章的id
get_the_ID()

//返回一个数组，里面是所属分类信息
get_the_category();

//返回的样例
Array
(
    [0] => WP_Term Object
```

```
        (
            [term_id] => 6
            [name] => 分类1
            [slug] => fl1
            [term_group] => 0
            [term_taxonomy_id] => 6
            [taxonomy] => category
            [description] =>
            [parent] => 0
            [count] => 1
            [filter] => raw
            [cat_ID] => 6
            [category_count] => 1
            [category_description] =>
            [cat_name] => 分类1
            [category_nicename] => fl1
            [category_parent] => 0
        )

    [1] => WP_Term Object
        (
            [term_id] => 7
            [name] => 分类2
            [slug] => fl2
            [term_group] => 0
            [term_taxonomy_id] => 7
            [taxonomy] => category
            [description] =>
            [parent] => 0
            [count] => 1
            [filter] => raw
            [cat_ID] => 7
            [category_count] => 1
            [category_description] =>
            [cat_name] => 分类2
            [category_nicename] => fl2
            [category_parent] => 0
        )

)
```

获取指定分类下的文章列表

```
get_posts()
```

post_type：指定要获取的文章类型，默认为 'post'。
post_status：指定要获取的文章状态，默认为 'publish' 已发布，'draft, publish, future, pending, private'等。

posts_per_page：指定每页获取的文章数量，默认为 -1（获取所有文章）。
orderby：指定文章的排序规则，默认为 'date'，可以设置为 'title'、'meta_value' 等。
order：指定文章的排序顺序，默认为 'DESC'（降序），可以设置为 'ASC'（升序）。

```
$posts  = get_posts(array(
      'numberposts'      => 5,
      'category'=>get_the_category()[0]->cat_ID,
      'orderby'          => 'date',
      'order'            => 'ASC',
      'exclude'=>array(get_the_ID())

    ));
    ?>
    <div class="list">


    <?php
    foreach ($posts as $key => $value) {
      echo '
      <div class="list-line">
        <a href="'.get_permalink($value->ID).'" target="_blank">
        '.$value->post_title.'
        </a>
      </div>
      '
      ;
    }
    ?>
    </div>
```

orderby排序

根据数据库字段来排序，支持多个 `'orderby'=>'comment_count,date'`，

- 'none' 无排序
- 'name' 文章名称（别名）
- 'author'作者
- 'date'日期
- 'title'文章标题
- ‘type’文章类型
- 'modified'修改日期
- 'menu_order'
- 'parent'父id
- 'ID'文章id
- 'rand'随机排序
- 'relevance' 相关性
  参考文件：wp-includes/class-wp-query. php的parse_search_order函数
  必须有搜索词时才能生效，会去匹配文章标题 摘要 内容，相关性越高越靠前
- 'RAND(x)' (where 'x' is an integer seed value)
- 'comment_count'评论数量

注意：如果用下面这个，就需要使用meta_key来指定字段名

- 'meta_value' 非数字类型的自定义字段名
- 'meta_value_num' 数字类型的自定义字段
- 'post**in' 根据 post**in参数内提供的文章id排序，此时其他order将无效

```
'orderby'=>'post__in',
'post__in'=>array(16,51,49),
```

- 'post_name**in' 根据post_name**in参数内提供的文章别名排序
- 'post_parent**in' 根据post_parent**in参数内提供的父文章id排序

自定义字段

例如：文章阅读次数（或点赞次数）

```
//获取自定义字段值
get_post_meta( $post_id, $key = '', $single = false )
//删除自定义字段值
delete_post_meta( $post_id, $meta_key, $meta_value = '' )
//添加自定义字段值
add_post_meta( $post_id, $meta_key, $meta_value, $unique = false )
//修改自定义字段值
update_post_meta( $post_id, $meta_key, $meta_value, $prev_value = '' )
```

文章归档页

archive.php

```
//获取归档标题
the_archive_title()
```

头部工具栏

在页尾设置：

```
<div id="footer" style="text-align: center; padding: 20px 0;">
Copyright © 猫叔 2023-2024 <?php echo home_url();?> <?php echo bloginfo(
'name' )?>
</div>
<?php
  wp_footer();
?>
```

插入脚本和样式：

1.直接在header. php内

```
get_stylesheet_uri();//获取style.css的路径 （http://wp.tt/wp-
content/themes/ms1/style.css）
esc_url(get_stylesheet_uri());//清理掉多余的字符


<link rel="stylesheet" href="<?echo esc_url(get_stylesheet_uri())?>">
```

```
get_template_directory_uri(); //获取主题目录路径
```

2. 在functions. php内

> 注意，一定要加上

1. 使用 `wp_enqueue_script()` 将 JS 文件插入队列
2. 使用 `wp_enqueue_style()` 将 CSS 文件插入队列

`wp_enqueue_style ( $handle, $src, $deps, $ver, $media );`

- **$handle** 样式表名称（唯一）。
- **$src** 样式文件所在的位置，其余参数是可选的。
- **$deps** 指的是此样式表是否依赖于另一个样式表。如果设置了此项，则除非首先加载其依赖的样式表，否则不会加载此样式表。
- **$ver**：版本号。
- **$media**：可以指定要加载此样式表的媒体类型，例如 ‘all’, ‘screen’, ‘print’ 或 ‘handheld’。

```
define('_MS_VERSION','v1.0.0');
function ms1_scripts() {

        wp_enqueue_style( 'style',
get_stylesheet_uri(),array(),_MS_VERSION);
        wp_enqueue_style( 'index_style', get_template_directory_uri() .
'/assets/css/index.css',array(),_MS_VERSION);
        wp_enqueue_script('test',get_template_directory_uri() .
'/assets/test.js',array(),_MS_VERSION,array(

        'in_footer' => true

    ));
}
add_action( 'wp_enqueue_scripts', 'ms1_scripts' );
```

wp_script_is () 判断 js文件是否已加入队列

```
if ( !wp_script_is( 'jquery-ui' ) ) {
        wp_enqueue_script( 'jquery-ui' ,'xxxx.js');
}
```

案例：增加代码高亮功能
functions.php里引入css和js

```
if(is_single()){
        wp_enqueue_style( 'high_style',
'https://cdn.bootcdn.net/ajax/libs/highlight.js/11.8.0/styles/atom-one-
dark.min.css',array());

wp_enqueue_script('high_js','https://cdnjs.cloudflare.com/ajax/libs/highli
ght.js/11.9.0/highlight.min.js');
}
```

header.php里引入js代码

```
if(is_single()){
echo '<script>hljs.highlightAll();</script>';
}
```

functions. php

文章摘要长度

```
function custom_excerpt_length($length) {
    return 20; // 修改文章摘要长度为20个单词
}
add_filter('excerpt_length', 'custom_excerpt_length');
```

启用功能

```
add_theme_support('post-thumbnails'); // 启用特色图片
add_theme_support('post-formats', array('aside', 'gallery', 'quote',
'image', 'video')); // 启用文章格式
```

```
add_theme_support('automatic-feed-links');//添加自动Feed链接到`<head>`中，使用户能够订阅文章和评论的RSS Feed。
```

设置内容宽度

```
// 设置内容宽度
if (!isset($content_width)) {
    $content_width = 800; // 像素
}
```

条件判断

特殊页面判断

- is_home()
  判断是否为首页

```
if (is_home()) {
    // 首页的代码
}
```

- is_single()
  文章页

```
if (is_single()) {
    // 文章页的代码
}
```

- is_search()
  搜索页

```
if (is_search()) {
    // 搜索结果页的代码
}
```

测试代码：

```
var_dump(is_home(),'是否为首页<br>');
var_dump(is_single(),'是否为文章页<br>');
var_dump(is_search(),'是否为搜索页<br>');
```

存档日期

- is_date()
  是否是日期页（例如年份归档或月份归档页）
  例如：/archives/date/2023/12

```
if (is_date()) {
    // 日期页的代码
}
```

- is_year()

显示年度存档时，返回true。

- is_month()

何时月度存档，返回true。

- is_day()

显示每日存档时，返回true。

- is_time()

当显示每小时，“分钟”或“每秒”存档时，返回true。

测试代码:

```
var_dump(is_year(),'is_year()<br>');
var_dump(is_month(),'is_month()<br>');
var_dump(is_day(),'is_day()<br>');
```

- is_new_day()

确定循环中当前文章的发布日期是否与上一篇的不同。
注意：这里需要调用 `the_date()` 函数才能更新时间。

is_category

分类页，显示“分类目录存档”页面时，返回true

```
if (is_category()) {
    // 分类页的代码
}

var_dump(is_category())
```

is_category('16')

显示ID为16的分类目录存档页面时，返回true

is_category('Stinky Cheeses')

当显示名称为“Stinky Cheeses”的分类目录存档页面时，返回true

is_category（'blue-cheese'）

显示别名为“blue-cheese”的分类目录存档页面时，返回true

is_category(array(9，'blue-cheese'，'Stinky Cheeses'))

显示ID为9，或别名为「blue-cheese」,或名称为「Stiky Cheeses」的分类目录存档页面时，返回true。

in_category('5')

如果当前文章在ID为5的分类目录中时，返回true。

in_category(array(1,2,3))

如果当前文章在ID为1、2或3的分类

目录中时，返回true。

!in_category(array(4,5,6))

如果当前分类不在ID为4、5或6的分类目录中时，返回true。

is_tag

是否为标签存档页面，例如这样的页面 /archives/tag/tag1

is_tag('文章类型')

是否为名称为「文章类型」的标签存档页面。

is_tag(array('sharp'，'mild'，'extreme'))

标签别名为“sharp”，“mild” 或 “extreme” 时，返回true。

has_tag()

当前文章有标签时，返回true。必须在文章循环中使用。

has_tag('文章类型')

当前文章的标签为“文章类型”时，返回true。

has_tag(array('sharp'，'mild'，'extreme'))

当前文章有别名为数组中的任意一个时，返回true.

is_page

是否为某个特定的单页面（Page）或者满足一组特定页面的条件。
这个函数对于根据页面类型动态加载内容或应用不同样式非常有用。

`is_page()`
当前正在浏览的页面是一个独立页面，一般是在后台页面中添加的。

`is_page('16')`

当前页面ID为16时，返回true。

`is_page('About Me')`

当前页面的标题为「About me」时，返回true。

is_page('about-me')
检查是否在具有特定slug（别名）的页面上

检查多个页面ID或slug

```php
$page_slugs = array('about', 'contact');
if ( is_page( $page_slugs ) ) {
   // 当前页面的slug是数组中列出的一个
}

$page_ids = array(12, 34, 56);
if ( is_page( $page_ids ) ) {
   // 当前页面ID在数组中列出的一个之内
}
```

是否在注册页

```php
if (is_page('注册') || is_page('wp-login.php?action=register')) {
    // 注册页的代码
}
```

此外，`is_page()` 函数还可以结合其他条件来实现更复杂的逻辑判断，比如在自定义页面模板中使用时，可以用来加载不同的侧边栏、头部或脚部内容等。

注意：`is_page()` 函数只对WordPress中的"页面"类型生效，不适用于文章（posts）、分类存档、标签存档或其他非页面类型的定制内容。

is_page_template

确定当前页面是否使用了 **页面模版** 或是否使用了 **特定的页面模版。**

is_page_template()

是否使用了页面模板

is_page_template('about.php')

是否使用了「about.php」页面模板？请注意，与其他判断函数不同，如果要指定页面模板，则需要使用文件名，例如about.php或my_page_template.php。

注意：如果模版文件位于子目录中，参数中还需要包含目录，例如'page-templates/about.php'。

判断是否登录用户

```php
if (is_user_logged_in()) {
    // 登录用户的代码
}
```

current_user_can()函数

```php
current_user_can( 'edit_posts' );
current_user_can( 'edit_post', $post->ID );
current_user_can( 'edit_post_meta', $post->ID, $meta_key );
```

判断是否是管理员：

```php
if (current_user_can('manage_options')) {
    // 管理员的代码
}
```

add_action钩子

init
admin_init
widgets_init
wp_head
wp_enqueue_scripts
login_head
admin_footer
edit_form_after_title
admin_enqueue_scripts
admin_menu

wp*ajax*
wp*ajax_nopriv*

add_filter

wp_robots

后台设置页

添加一段新的设置

page : 'general', 'reading', 'writing', 'discussion', 'media'

```
add_settings_section( $id, $title, $callback, $page, $args = array());

//实际案例:
add_action('admin_init',function(){
    add_settings_section('maoshutest1','猫叔测1',function(){
        echo '这是猫叔测1';
    },'general');
});
```

添加自定义字段

```
add_settings_field($id, $title, $callback, $page, $section = 'default',
$args = array())
```

```
do_settings_sections( $page )
```

用户

获取用户列表

```
get_users($args)
```

role：指定要获取的用户角色，默认为空，表示获取所有角色的用户。
orderby：指定用户的排序规则，默认为 'login'，可以设置为 'ID'、'display_name' 等。
order：指定用户的排序顺序，默认为 'ASC'（升序），可以设置为 'DESC'（降序）。
number：指定要获取的用户数量，默认为 -1，表示获取所有用户。
meta_key 和 meta_value：用于根据用户的元数据进行过滤。

WordPress数据库表

wp_commentmeta： 存储评论的元数据，包括评论的键值对数据，用于扩展评论的信息。

wp_comments： 存储网站上的所有评论，包括评论的作者、内容、时间等信息。

wp_links： 存储友情链接（Blogroll）的信息，包括链接的URL、标题、描述等。

wp_options： 存储WordPress系统选项和插件、主题的配置信息。这是一个非常重要的表，用于存储全局配置数据。

wp_postmeta： 存储文章、页面、上传文件和修订版本的元数据，允许您为这些内容添加自定义信息。

wp_posts： 存储文章、页面、上传文件和修订版本的内容，包括标题、正文、作者等。

wp_terms： 存储每个分类和标签的信息，用于组织和分类内容。

wp_term_relationships： 存储文章、链接和对应分类的关系信息，将内容与分类关联起来。

wp_term_taxonomy： 存储每个分类和标签所对应的分类方法，例如文章分类或链接分类。

wp_usermeta： 存储用户的元数据，允许您为用户添加自定义信息，如社交媒体链接、个性化设置等。

wp_users： 存储用户的基本信息，包括用户名、密码、电子邮件地址等。

```
global $wpdb, $post;
//执行sql语句
$sql = "SELECT * FROM $wpdb->comments WHERE comment_post_ID = $post->ID
AND comment_type = ''";
$wpdb->get_results($sql)
```

```
//数据库查询次数
get_num_queries();
//用时
timer_stop(1, 7);

//内存消耗
memory_get_peak_usage() / 1024 / 1024;
```

分类

```php
get_categories();

wp_list_categories();

$categories = get_categories( array(
            'orderby' => 'name',
            'order'   => 'ASC',
            'parent'=>'',
            'hide_empty'      => false,
        ) );
```