Programação para Wordpress

Sumário

1 - Plugins

1.1 – Introdução

Plugin Handbook
https://developer.wordpress.org/plugins/

Tipos de plug-ins WordPress

Os plug-ins podem realizar muitas tarefas. O que todos eles têm em comum é que adicionam funcionalidades extras ao seu site. Os tipos de plug-in WordPress incluem:

• plug-ins de manutenção de sites para itens como segurança, desempenho ou backups
• plug-ins de marketing e vendas para coisas como SEO, mídia social ou comércio eletrônico
• plug-ins de conteúdo, como tipos de postagem personalizados, widgets, códigos de acesso, formulários, galerias e feeds de vídeo
• Plug-ins de API que funcionam com a API REST do WordPress ou extraem conteúdo externo de serviços como o Google Maps
• plug-ins de comunidade que adicionam recursos de redes sociais

E muito mais! Para ter uma ideia do que os plug-ins podem fazer, verifique o diretório de plug-ins do WordPress e o mercado CodeCanyon.

Felizmente, o WordPress torna o processo bastante fácil. Será necessário algum conhecimento de codificação, mas não é muito difícil aprender como criar um plugin básico para adicionar funcionalidades ao seu site.

Há uma diferença entre WordPress.com e WordPress.org. A versão .org é a opção de código aberto que pode ser baixada gratuitamente e usada para criar um site personalizado. É a versão que abordaremos neste post. A versão .com é um construtor de sites hospedado com o qual você pode criar um site limitado gratuitamente.

https://developer.wordpress.org/plugins/

Efetuar uma pesquisa para ver se já existe um plugin que te atenda e que esteja em plena atividade
Caso decida efetuar um planejamento para a criação do plugin

Finalmente, antes de mergulhar na construção, você vai querer ler os Padrões de Codificação do WordPress (https://codex.wordpress.org/WordPress_Coding_Standards). Isso é particularmente importante se você planeja compartilhar ou vender seu plugin. Esses padrões de codificação são um conjunto de diretrizes e práticas recomendadas que os desenvolvedores devem tentar seguir ao criar temas e plug-ins para WordPress.

Criar um ambiente de desenvolvimento e testes
- Com servidor web, php, mysql e cia.
- Gedit ou Xed
- VSCode

Que permita instalar o WP e criar os plugins, etc.

Requisitos

Existem basicamente duas exigências do Wordpress para a criação de um plugin:
- Primeiro que ele esteja na pasta wp-content/plugins. Ele pode estar em sua própria pasta ou apenas em um arquivo php na pasta wp-content/plugins, como é o plugin hello.php, que vem com o WP.
- Segundo e último ele precisa estar em um arquivo do tipo php e conter um cabeçalho com comentário do tipo phpdoc. Neste bloco de comentários apenas uma linha é obrigatória, a que contém o nome do plugin, mas podem existir muitas informações:

```
/*
 * Plugin Name: YOUR PLUGIN NAME
 */
```

Uma observação que quase não é necessária é que os nomes de plugins devem ser únicos.

Aqui encontrará uma boa relação de itens que podemos adicionar no cabeçalho:

https://developer.wordpress.org/plugins/plugin-basics/header-requirements/

Agora vou fazer alguns testes com a criação de plugins.

1) Criar um plugin apenas com um único arquivo na pasta wp-content/plugins e somente com uma única linha no cabeçalho.

Após instalar o Wordpress...

Criar o arquivo:

wp-content/plugins/ola-mundo.php

Contendo:

```
<?php
/*
 * Plugin Name: Olá Mundo
 */
```

Após salvar já podemos ver na administração do site, em Plugins, nosso Olá Mundo, que ainda não está ativado. Mas não ativarei, pois ele ainda não tem nenhuma funcionalidade.

2) Criar um plugin apenas com um único arquivo na pasta wp-content/plugins e somente com uma única linha no cabeçalho.
Criar o arquivo:

wp-content/plugins/ola-mundo2.php

Contendo:

```
<?php
/*
Plugin Name: Olá Mundo
*/
```

Para experimentar eu retirei o * da linha do Plugin Name.

Atualizarei a seção Plugins do site para ver.
Veja que ele aceita assim também. Particularmente me parece mais coerente, na primeira forma, contendo asteriscos em todas as linhas do cabeçalho, mas isso para mim.

3) Agora vou criar uma pasta e um arquivo dentro dela

wp-content/plugins/ola-mundo3/index.php

Contendo:

```
<?php
/*
 * Plugin Name: Olá Mundo 3
 */
```

Quando se cria uma pasta, geralmente o arquivo .php tem o mesmo nome da pasta, que é o nome do plugin, mas também podemos usar o nome do arquivo como index.php

Atualizemos o site em Plugins e aparece nosso terceiro plugin.

1.2 - Alguns cuidados com a segurança do plugin

Garantir que o plugin somente seja executado através do WP e não diretamente

```
if ( ! defined( 'ABSPATH' ) ) {
        exit; // Exit if accessed directly.
}
```

ou

```
if(!defined('ABSPATH')) die();
```

Outro bom recurso é deixar o arquivo principal do plugin com o nome do plugin/pasta e dentro da pasta um index.php contendo somente:

```
<?php
# Silence is golden.
```

Isto chama quem consegue acessar o plugin para o index.php vazio e o desvia do arquivo principal

1.3 - O que são hooks do WordPress e codificação

Os plug-ins interagem com o código principal do WordPress usando hooks. Existem dois tipos diferentes de hooks no WordPress:

• hooks de actions – adicionar ou remover funções.
• hooks de filters – modificam dados produzidos por funções.

Hooks are not just for plugin developers; hooks are used extensively to provide default functionality by WordPress core itself. Other hooks are unused place holders that are simply available for you to tap into when you need to alter how WordPress works. This is what makes WordPress so flexible.

Por exemplo, todos os plugins usam hooks/ganchos para interagir com o WordPress.

Um hook é como um plugin se conecta ao código pré-existente da programação principal do WordPress. Em outras palavras, o hook é o ponto de ancoragem onde um plugin se insere para adicionar ou alterar a funcionalidade de um site.

Os hooks são uma parte importante do desenvolvimento do WordPress. Existem centenas de hooks que podem ser usados como gatilhos para um plugin, e você pode até criar novos, se necessário.
https://developer.wordpress.org/reference/hooks/

Mas, por enquanto, existem dois tipos de hooks que você precisa considerar ao criar seu plugin:

1. Actions: adicionam ou alteram a funcionalidade do WordPress e constituem a maioria dos ganchos.
2. Filters: São usados para modificar a funcionalidade das ações.

The 3 basic hooks you'll need when creating a plugin are the register_activation_hook() , the register_deactivation_hook() , and the register_uninstall_hook() .

The activation hook is run when you activate your plugin. You would use this to provide a function to set up your plugin — for example, creating some default settings in the options table.

The deactivation hook is run when you deactivate your plugin. You would use this to provide a function that clears any temporary data stored by your plugin.
These uninstall methods are used to clean up after your plugin is deleted using the WordPress Admin. You would use this to delete all data created by your plugin, such as any options that were added to the options table.

Adding Hooks

You can add your own, custom hooks with do_action() , which will enable developers to extend your plugin by passing functions through your hooks.

Removing Hooks

You can also use invoke remove_action() to remove a function that was defined earlier. For example, if your plugin is an add-on to another plugin, you can use remove_action() with the same function callback that was added by the previous plugin with add_action() . The priority of actions is important in these situations, as remove_action() would need to run after the initial add_action() .

How WordPress Loads Plugins

When WordPress loads the list of installed plugins on the Plugins page of the WordPress Admin, it searches through the plugins folder (and its sub-folders) to find PHP files with WordPress plugin header comments.

Para codificar seu plugin, você precisará se familiarizar com os hooks e como eles funcionam. Felizmente, o Plugin Handbook do WordPress pode ajudá-lo a começar.

Para este guia, usaremos o seguinte código (fonte) como exemplo:

```
function modify_read_more_link() {
   return '<a class="more-link" href="' . get_permalink() . '">Clique para ler!</a>';
}
add_filter( 'the_content_more_link', 'modify_read_more_link' );
```

Como você pode ver, este código usa um filtro para modificar o link padrão “Leia mais”, substituindo-o por um valor diferente: “Clique para ler!”

Observe também que o código do plugin de exemplo acima funciona apenas para sites que utilizam temas clássicos. Se você usa o editor de site integrado ao WordPress – que está no software principal há vários anos – para criar o layout do seu site usando blocos, o código acima não fará muito por você.

Save this example as a PHP file and upload it to the plugins folder.

```
<?php
/*
Plugin Name: Add Text To Footer
*/
// Hook the 'wp_footer' action hook, add the function named 'mfp_Add_Text' to it
add_action("wp_footer", "mfp_Add_Text");
// Define 'mfp_Add_Text'
function mfp_Add_Text()
{
        echo "<p style='color: black;'>After the footer is loaded, my text is added!</p>";
}
```

https://wordpress.org/plugins/hello-dolly/

```
function my_greeting() {
   echo 'Welcome to my website!';
}
add_action('the_content', 'my_greeting');

function add_welcome_message() {
   echo 'Welcome to my blog!';
}
add_action('wp_head', 'add_welcome_message');

function add_signature($content) {
```

```
    $signature = '<p>Thanks for reading! - Your Name</p>';
    return $content . $signature;
}
add_filter('the_content', 'add_signature');

add_action( 'the_content', 'test_example_text' );

function test_example_text ( $content ) {
return $content .= '&lt;p&gt;This is my first plugin!&lt;/p&gt;';
}

function recipe_custom_post_type() {
register_post_type( 'recipe', array( 'public' =&gt; true, 'label' =&gt; 'Recipes' ) );
}
add_action( 'init', 'recipe_custom_post_type' );

if( !function_exists("extra_post_info") ) {
    function extra_post_info($content)   {
        $extra_info = "EXTRA INFO";
            return $content . $extra_info;
    }
    add_filter('the_content', 'extra_post_info');
}
```

Boas Práticas e Dicas ao Criar Plugins Personalizados

Como seu site precisa evoluir continuamente, você precisará revisitar o código dos seus plugins para implementar novidades e corrigir falhas de segurança.

Com isso em mente, é uma boa ideia seguir algumas boas práticas no desenvolvimento de plugins desde o início. Com isso, você facilitará todo o processo para você mesmo e para quaisquer desenvolvedores que venham a trabalhar no seu plugin futuramente.

Além disso, você pode dar uma olhada nos melhores plugins WordPress para ter um pouco de inspiração. Analise seus códigos-fonte, como eles organizam suas pastas e outras práticas que você pode aplicar ao criar os seus próprios plugins.

Aqui estão algumas dicas de escrita de código e desenvolvimento de plugins que poderão te ajudar na criação do seu primeiro plugin WordPress:

- Desenvolva e teste seus plugins num ambiente de teste. Desta forma, você não corre o risco de quebrar o seu site no caso de algum código problemático.
- Crie uma estrutura de pastas lógica. Crie subpastas para cada funcionalidade e divida o código em diferentes arquivos, com base em seu propósito ou na linguagem de programação escolhida, para evitar bagunça.
- Nomeie cada arquivo, pasta e elemento com cuidado. Use prefixos únicos, para que eles não entrem em conflito com os nomes de arquivos de outros plugins ou do próprio WordPress.
- Adicione comentários para explicar cada função. Isso permite que outros desenvolvedores entendam seu código ao atualizá-lo ou corrigir seus bugs.
- Crie uma documentação. Essa prática é particularmente vantajosa caso você crie plugins com funcionalidades complexas para muitos usuários.

• Use um software de controle de versão para acompanhar as mudanças feitas no seu código. Saber quem adicionou ou modificou cada código ajuda a previnir conflitos entre atualizações e reduzir o número de bugs.
• Use o Codex do WordPress como referência para seguir padrões de código específicos de cada linguagem. Certifique-se de seguir os padrões ao colaborar num projeto.
• Ative o WP_DEBUG ou use uma ferramenta de debugging ao desenvolver plugins. Isso facilitará a detecção de bugs, acelerando o desenvolvimento como um todo.

https://wordpress.org/plugins/

In WordPress plugin development, you can save data in a Database easily, but be careful

WordPress provides a global object named $wpdb, you can always call this object and use it by writing global $wpdb; code in your function. Something like this:

```
<?php
function insertDataToDatabase(){
  global $wpdb;
  $table_name = $wpdb->prefix . "custom_table";
  $wpdb->insert($table_name, array('name' => "Test", 'email' => "test@test.com") );
}
```

$wpdb object contains all of the database functions that you need.
in the above code, we used insert() function of $wpdb object to insert our data to custom_table table in the database.

You should be careful when working with a database, especially when saving user inputs.
Remember this: never trust the user's input.

Add submenu section

```
function crunchify_add_menu() {
        add_submenu_page("options-general.php", "Crunchify Plugin", "Crunchify Plugin",
"manage_options", "crunchify-hello-world", "crunchify_hello_world_page");
}
add_action("admin_menu", "crunchify_add_menu");
```

We are using add_action WordPress hook to add submenu for our plugin.

mkdir plugins/cadastro_de_cliente

Todo plugin fica na pasta

wp-content/plugins

Cada plugin deve ter sua pasta

Cada plugin deve ter no mínimo os arquivos
index.php
style.css

E no index.php obrigatoriamente deve ter o cabeçalho com dados sobre o plugin, no mínimo o Plugin Name

```
<?php

/**
 * Plugin Name: Nome do plugin
 * Test Domain: cadastro_de_cliente
 */

// Evitar que plugin seja acessado pela URL
if(!function_exists('add_action')){
	echo __('O plugin não pode ser passado diretamente', 'cadastro_de_cliente');
	exit;
}

// setup

// hooks
register_activation_hook(__FILE__, 'dc_activate');

// includes
include('includes/activate.php');
// shorcodes
```

```
mkdir plugins/cadastro_de_cliente/includes

nano plugins/cadastro_de_cliente/includes/activate.php
```

```
<?php
function dc_activate(){
	if(version_compare(get_bloginfo('version'), '4.5', '<')){
		wp_dir(__('Precisa atualizar o Wordpress para usar este plugin',
'cadastro_de_cliente'));
	}
}
```

Exemplos de funções para plugins:

```
function modify_read_more_link() {
   return '<a class="more-link" href="' . get_permalink() . '">Clique para ler!</a>';
}
add_filter( 'the_content_more_link', 'modify_read_more_link' );
```

Observe também que o código do plugin de exemplo acima funciona apenas para sites que utilizam temas clássicos. Se você usa o editor de site integrado ao WordPress – que está no software principal há vários anos – para criar o layout do seu site usando blocos, o código acima não fará muito por você.

```
add_action("wp_footer", "mfp_Add_Text");
// Define 'mfp_Add_Text'
```

```
function mfp_Add_Text()
{
        echo "<p style='color: black;'>After the footer is loaded, my text is added!</p>";
}

function my_greeting() {
   echo 'Welcome to my website!';
}
add_action('the_content', 'my_greeting');

function add_welcome_message() {
   echo 'Welcome to my blog!';
}
add_action('wp_head', 'add_welcome_message');

function add_signature($content) {
   $signature = '<p>Thanks for reading! - Your Name</p>';
   return $content . $signature;
}
add_filter('the_content', 'add_signature');

add_action( 'the_content', 'test_example_text' );

function test_example_text ( $content ) {
return $content .= '&lt;p&gt;This is my first plugin!&lt;/p&gt;';
}

function recipe_custom_post_type() {
register_post_type( 'recipe', array( 'public' =&gt; true, 'label' =&gt; 'Recipes' ) );
}
add_action( 'init', 'recipe_custom_post_type' );

if( !function_exists("extra_post_info") ) {
   function extra_post_info($content)   {
       $extra_info = "EXTRA INFO";
           return $content . $extra_info;
   }
   add_filter('the_content', 'extra_post_info');
}
```

Acessando banco de dados

In WordPress plugin development, you can save data in a Database easily, but be careful WordPress provides a global object named $wpdb, you can always call this object and use it by writing global $wpdb; code in your function. Something like this:

```
<?php
function insertDataToDatabase(){
  global $wpdb;
  $table_name = $wpdb->prefix . "custom_table";
  $wpdb->insert($table_name, array('name' => "Test", 'email' => "test@test.com") );
}
```

$wpdb object contains all of the database functions that you need.
in the above code, we used insert() function of $wpdb object to insert our data to custom_table table in the database.

You should be careful when working with a database, especially when saving user inputs.
Remember this: never trust the user's input.

Add submenu section

```
function crunchify_add_menu() {
	add_submenu_page("options-general.php", "Crunchify Plugin", "Crunchify Plugin",
"manage_options", "crunchify-hello-world", "crunchify_hello_world_page");
}
add_action("admin_menu", "crunchify_add_menu");
```

We are using add_action WordPress hook to add submenu for our plugin.

admin-header.php

O WP tem um arquivo

wp-admin/admin-header.php

Se adicionarmos uma mensagem ao final dele ela aparecerá no painel

Adicionar ao final

```
...
print '<center><h3>Como vai Ecoviver</h3></center>';
```

Mas quando o WP for atualizado ele apagará nossa mensagem.

Podemos criar um pequeno plugin com uma pequena função que force o WP a chamar nossa mensagem

nano wp-content/plugins/plg_msg.php contendo

```
<?php
/**
 * Plugin name: Mensagem
 */

define( 'ABSPATH' ) !! exit;

function msg(){
	print '<center><h3>Como vai Ecoviver</h3></center>';
}
// actions
add_action('all_admin_notices', 'msg');
```

```
/* Add a paragraph only to Pages. */
function my_added_page_content ( $content ) {
    if ( is_page() ) {
        return $content . '<p>Your content added to all pages (not posts).</p>';
    }

    return $content;
}
add_filter( 'the_content', 'my_added_page_content');
```

Filtros retornam algo
Actions somente executam

Pega os últimos 15 posts de uma categoria qualquer.

```
<?php
query_posts('category_name=categoria_qualquer&posts_per_page=15'); ?>

<?php while (have_posts()) : the_post(); ?>
<!-- Faz coisas com a “categoria_qualquer”. -->
<?php endwhile;?>
```

https://www.youtube.com/watch?v=pDvXHS0ongY

https://www.youtube.com/playlist?list=PLYW_GTMrtrwxVDL2Ygxo-WoQgOkgKFGA3

https://developer.wordpress.org/plugins/

https://codex.wordpress.org/WordPress_Coding_Standards

https://developer.wordpress.org/reference/hooks/

2 – Templates

Os temas definem o layout e design dos sites WordPress. HTML5, CSS3, e o princípio do design responsivo são muito importante para websites mais modernos. Os temas fazem uso de variadas funções principais do WordPress, então o conhecimento básico de PHP é altamente recomendado.

Fundamentalmente, o sistema de temas WordPress é uma forma de "skin" do seu weblog. No entanto, é mais do que apenas uma "skin". Sem skin, implica que só o design de seu siteé alterado. Temas WordPress podem oferecer muito mais controle sobre a aparência e a apresentação do material em seu site.

Os temas WordPress são uma coleção de arquivos que trabalham juntos para produzir uma interface gráfica com um design subjacente e unificador para um weblog. Estes arquivos são chamados arquivos de modelo (template files). Um tema modifica a maneira como o site é exibido, sem modificar o software subjacente e os dados que ele gerencia. Os temas podem incluir arquivos de modelos personalizados, arquivos de imagem (*.jpg, *.gif),

folhas de estilos (*.css), páginas personalizadas, bem como de quaisquer arquivos de código necessário (*.php).

Os temas são um jogo totalmente novo. Vamos dizer que você escreve muito sobre o futebol e música. Através do uso inovador do Loop WordPress e arquivos de modelo, você pode personalizar suas postagens de forma diferente, de acordo com a categoria de tais postagens.

Assim, suas postagens sobre futebol podem aparecer num fundo verde, e as postagens sobre música num fundo branco, por exemplo.

Com este poderoso controle sobre como diferentes páginas e categorias aparecem em seu site, você só estpa limitado pela sua imaginação.

2.1 - Arquivos de um tema

Seção obrigatória no arquivo style.css:

```
/*
Theme Name: Rose
Theme URI: the-theme's-homepage
Description: a-brief-description
Author: your-name
Author URI: your-URI
Template: use-this-to-define-a-parent-theme--optional
Version: a-number--optional
...
General comments/License Statement if any.
...
*/
```

De maneira bem minimalista, um tema é composto de 2 arquivos básicos e obrigatórios.

• style.css
• index.php

A idéia geral

O WordPress utiliza o Query String - Informações contidas no interior de cada link em seu site - para decidir qual o modelo ou conjunto de modelos serão utilizados para exibir a página.

Primeiro, o WordPress partidas cadeia de consulta para todos os tipos de consulta - ou seja, ele decide que tipo de página (uma página de busca, uma página da categoria, a home page, etc) está sendo solicitado.

Os templates são então escolhidos - e o conteúdo da página web é gerada - na ordem sugerida pela hierarquia Template WordPress, dependendo do que templates estão disponíveis num determinado templates WordPress.

Os arquivos são estes:

style.css: Folha de estilo do tema. É obrigatória e deve conter todos os estilos ou chamadas de outras folhas de estilo para o tema.
• category.php: Listagem de posts de uma categoria.
• tag.php: Listagem de posts de um tag.
• taxonomy.php: Listagem de uma taxonomia qualquer.
• index.php: O template principal. É usado para exibir qualquer conteúdo quando um template específico não for encontrado.
• author.php: Listagem de posts de um autor.
• comments.php: Listagem de comentários logo abaixo dos posts.
• date.php: Listagem de um intervalo de data (ano, mês, dia).
• comments-popup.php: Lista de popups em uma nova janela aberta via Javascript.
• archive.php: Usado de maneira generalista para category.php,
• home.php: A capa do site.
• search.php: Resultados de uma busca.
• single.php: Um post sozinho.
• 404.php: Página de erro para conteúdo não encontrado.
• page.php: Uma página.
author.php, e date.php.

Todos estes arquivos são usados para substituir o index.php quando encontrados de acordo com os tags condicionais. Você pode tornar estes arquivos ainda mais específicos variando suas derivações de nome, ou mesmo alterando seu código com expressões que usam os mesmos tags condicionais.

3 - Widgets

4 – Ferramentas

wp-cli
https://wp-cli.org
https://make.wordpress.org/cli/handbook/

WP-CLI é a interface de linha de comando para WordPress. Você pode atualizar plugins, configurar instalações multisite e muito mais, sem usar um navegador da web.

Instalação

curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar

sudo mv wp-cli.phar /usr/local/bin/wp

sudo chmod +x /usr/local/bin/wp

Autocompletar

Baixar
https://raw.githubusercontent.com/wp-cli/wp-cli/v2.10.0/utils/wp-completion.bash

Salvar em alguma pasta e chamar

nano ~/.bashrc

Adicionar
source /pasta/wp-completion.bash

source ~/.bashrc

Exemplos de uso:

wp scaffold plugin nomeplugin

Não criar os testes unitários

wp scaffold plugin olamundo4 --skip-tests

5 – Funções do WorfPress

Referência de funções

Os arquivos do WordPress definem várias funções PHP úteis. Algumas das funções, conhecidas como Template Tags, foram definidas especialmente para uso nos Temas WordPress. Existem também algumas funções relacionadas com ações e filtros (a Plugin API), que são portanto usadas a princípio para desenvolvimento de Plugins. O resto é usado para criar as funcionalidades núcleo do WordPress.

Muitas das funções núcleo do WordPress se úteis aos desenvolvedores de Temas e Plugins. Então, este artigo lista a maioria das funções núcleo, excluindo as Template Tags. Ao final da página, tem uma seção listando outros recursos para se encontrar informações sobre as funções do WordPress. Além dessas informações, o WordPress phpdoc site detalha todas as funções do WordPress por versões desde a 2.6.1.

Post, Página, Anexo e Bookmarks

Posts

• get_children
• get_extended
• get_post
• get_post_ancestors
• get_post_mime_type
• get_post_status
• get_post_type
• get_posts
• is_post
• is_single
• is_sticky
• wp_get_recent_posts
• wp_get_single_post

Inserção/Remoção de Post

• wp_delete_post
• wp_insert_post
• wp_publish_post
• wp_update_post

Páginas

• get_all_page_ids
• get_page
• get_page_by_path
• get_page_by_title
• get_page_children
• get_page_hierarchy
• get_page_uri
• get_pages

• is_page
• page_uri_index
• wp_list_pages
get_the_author
get_the_content
wp_get_post_categories
wp_set_post_categories
wp_trim_excerpt

Categorias, tags e taxonomia

Categorias

• cat_is_ancestor_of
• get_all_category_ids
• get_cat_ID
• get_cat_name
• get_categories
• get_category
• get_category_by_path
• get_category_by_slug
• get_category_link
• get_category_parents
• get_the_category
• in_category
• is_category

Criação de Categorias

• wp_create_category
• wp_insert_category

Tags

• get_tag
• get_tag_link
• get_tags
• get_the_tag_list
• get_the_tags
• is_tag

Taxonomia

• get_term
• get_the_term_list
• get_term_by
• get_term_children
• get_terms
• is_taxonomy
• is_taxonomy_hierarchical
• is_term
• register_taxonomy

• wp_get_object_terms
• wp_insert_term
• wp_update_term
get_rss
get_search_comments_feed_link
get_search_feed_link
get_the_category_rss
get_the_title_rss
permalink_single_rss
post_comments_feed_link
rss_enclosure
the_title_rss
the_category_rss
the_content_rss
the_excerpt_rss
wp_rss

Comentários, Ping, e Trackback

add_ping
check_comment
discover_pingback_server_uri
do_all_pings
do_enclose
do_trackbacks
generic_ping
get_approved_comments
get_comment
get_comments
get_enclosed
get_lastcommentmodified
get_pung
get_to_ping
next_comments_link
paginate_comments_links
pingback
previous_comments_link
privacy_ping_filter
sanitize_comment_cookies
trackback
trackback_url_list
weblog_ping
wp_allow_comment
wp_delete_comment
wp_filter_comment
wp_get_comment_status
wp_get_current_commenter
wp_insert_comment
wp_new_comment
wp_set_comment_status
wp_throttle_comment_flood
get_category_template

get_comments_popup_template
get_current_theme
get_date_template
get_header_image
get_header_textcolor
get_home_template
get_locale_stylesheet_uri
get_page_template
get_paged_template
get_query_template
get_search_template
get_single_template
get_stylesheet
get_stylesheet_directory
get_stylesheet_directory_uri
get_stylesheet_uri
get_template
get_template_directory
get_template_directory_uri
get_theme
get_theme_data
get_theme_mod
get_theme_root
get_theme_root_uri
get_themes
header_image
load_template
locale_stylesheet
preview_theme
preview_theme_ob_filter
preview_theme_ob_filter_callback
set_theme_mod
switch_theme
validate_current_theme

Formatação

add_magic_quotes
addslashes_gpc
antispambot
attribute_escape
backslashit
balanceTags
clean_pre
clean_url
convert_chars
convert_smilies
wp_make_link_relative
wp_rel_nofollow
wp_richedit_pre
wp_specialchars
zeroise

Diversas

Funções de Data/Hora

• current_time
• date_i18n
• get_calendar
• get_date_from_gmt
• get_lastpostdate
• get_lastpostmodified
• get_day_link
• get_gmt_from_date
• get_month_link
• get_the_time
• get_weekstartend
• get_year_link
• human_time_diff
• is_new_day
• iso8601_timezone_to_offset
• iso8601_to_datetime
• mysql2date

Serialização

• is_serialized
• is_serialized_string
• maybe_serialize
• maybe_unserialize

Opções

• add_option
• delete_option
• form_option
• get_alloptions
• get_user_option
• get_option
• update_option
• update_user_option

XMLRPC

• xmlrpc_getpostcategory
• xmlrpc_getposttitle
• xmlrpc_removepostdata
• user_pass_ok

Localização
• __
• _e
wp_nonce_field
wp_nonce_url
wp_notify_moderator
wp_notify_postauthor
wp_original_referer_field
wp_redirect
wp_referer_field

Campos Personalizados (postmeta)

• add_post_meta
• delete_post_meta
• get_post_custom
• get_post_custom_keys
• get_post_custom_values
• get_post_meta
• update_post_meta

Anexos

• get_attached_file
• is_attachment
• is_local_attachment
• update_attached_file
• wp_attachment_is_image
• wp_insert_attachment
• wp_delete_attachment
• wp_get_attachment_image
• wp_get_attachment_image_src
• wp_get_attachment_metadata
• wp_get_attachment_thumb_file
• wp_get_attachment_thumb_url
• wp_get_attachment_url
• wp_check_for_changed_slugs
• wp_count_posts
• wp_mime_type_icon
• wp_update_attachment_metadata

Bookmarks

• get_bookmark
• get_bookmarks
• wp_list_bookmarks

Outros

• add_meta_box
• get_the_ID

Usuários e Autores

• auth_redirect
• email_exists
• get_currentuserinfo
• get_profile
• get_userdata
• get_userdatabylogin
• get_usernumposts
• set_current_user
• user_pass_ok
• username_exists
• validate_username
• wp_get_current_user
• wp_set_current_user

User meta

• delete_usermeta
• get_usermeta
• update_usermeta

Inserção/Remoção de Usuários

• wp_create_user
• wp_delete_user
• wp_insert_user
• wp_update_user

Login / Logout

• is_user_logged_in
• wp_signon
• wp_logout

Feeds

bloginfo_rss
comment_author_rss
comment_link
comment_text_rss
do_feed
do_feed_atom
do_feed_rdf
do_feed_rss
do_feed_rss2
fetch_rss
get_author_feed_link
get_bloginfo_rss
get_category_feed_link
get_comment_link
get_comment_author_rss

get_post_comments_feed_link
• wp_update_comment
• wp_update_comment_count

Ações, Filtros e Plugins

Filters

• add_filter
• apply_filters
• merge_filters
• remove_filter

Actions

• add_action
• did_action
• do_action
• do_action_ref_array
• remove_action

Plugins

• plugin_basename
• register_activation_hook
• register_deactivation_hook
• register_setting
• settings_fields
• unregister_setting

Shortcodes

• add_shortcode
• do_shortcode
• do_shortcode_tag
• get_shortcode_regex
• remove_shortcode
• remove_all_shortcodes
• shortcode_atts
• shortcode_parse_atts
• strip_shortcodes

Relacionadas a Temas

Funções de Inclusão

• comments_template
• get_footer
• get_header
• get_sidebar
• get_search_form

Outras Funções

• add_custom_image_header
• get_404_template
• get_archive_template
• get_attachment_template
• get_author_template
ent2ncr
esc_attr
force_balance_tags
format_to_edit
format_to_post
funky_javascript_fix
htmlentities2
is_email
js_escape
make_clickable
popuplinks
remove_accents
sanitize_email
sanitize_file_name
sanitize_user
sanitize_title
sanitize_title_with_dashes
seems_utf8
stripslashes_deep
trailingslashit
untrailingslashit
utf8_uri_encode
wpautop
wptexturize
wp_filter_kses
wp_filter_post_kses
wp_filter_nohtml_kses
wp_iso_descrambler
wp_kses
wp_kses_array_lc
wp_kses_attr
wp_kses_bad_protocol
wp_kses_bad_protocol_once
wp_kses_bad_protocol_once2
wp_kses_check_attr_val
wp_kses_decode_entities
wp_kses_hair
wp_kses_hook
wp_kses_html_error
wp_kses_js_entities
wp_kses_no_null
wp_kses_normalize_entities
wp_kses_normalize_entities2
wp_kses_split
wp_kses_split2

wp_kses_stripslashes
wp_kses_version
• _ngettext
• esc_attr_e
• get_locale
• load_default_textdomain
• load_plugin_textdomain
• load_textdomain
• load_theme_textdomain

Cron (Agendamento)

• spawn_cron
• wp_clear_scheduled_hook
• wp_cron
• wp_get_schedule
• wp_get_schedules
• wp_next_scheduled
• wp_reschedule_event
• wp_schedule_event
• wp_schedule_single_event
• wp_unschedule_event

Diversas

• add_query_arg
• bool_from_yn
• cache_javascript_headers
• check_admin_referer
• check_ajax_referer
• do_robots
• get_bloginfo
• get_num_queries
• is_blog_installed
• make_url_footnote
• nocache_headers
• remove_query_arg
• status_header
• wp
• wp_check_filetype
• wp_clearcookie
• wp_create_nonce
• wp_die
• wp_explain_nonce
• wp_get_cookie_login
• wp_get_http_headers
• wp_get_original_referer
• wp_get_referer
• wp_hash
• wp_mail
• wp_mkdir_p
• wp_new_user_notification

• wp_nonce_ays
wp_remote_fopen
wp_salt
wp_setcookie
wp_upload_bits
wp_upload_dir
wp_verify_nonce

Do e-book WordPress Para Desenvolvedores
https://getflywheel.com/design-and-wordpress-resources/ebooks/the-perfect-wordpress-development-workflow/