WordPress

O WordPress é uma plataforma semântica de vanguarda para publicação pessoal, com foco na estética, nos Padrões Web e na usabilidade. O WordPress é, ao mesmo tempo, um software livre e gratuito. Em outras palavras, o WordPress é o que você usa quando você quer trabalhar e não lutar com seu software de publicação de blogs.

Caracteríticas do WordPress

WordPress é uma poderosa plataforma de publicação pessoal que vem com um grande conjunto de características concebidas para tornar sua experiência, como editor na Internet, fácil, agradável e o mais atraente possível.

- Instalado localmente
- Core Portátil
- UTC amigável
- Gestão de usuários
- Perfis de usuário
- Fácil instalação e atualização
- Geração dinâmica de páginas
- Internacionalização e Localização
- Feeds
- Links Permanentes sem "Códigos estranhos"
- Comunicação entre-blogs
- Template Tags
- Temas
- Plugins
- Senha de proteção
- Mensagem para o futuro
- Postagens Multi-paginada
- Upload de imagens e arquivos
- Categorias
- Emoticons
- Salvar rascunhos
- Visualizar conteúdo não publicado
- Ferramentas Desktop Disponíveis
- Blog por e-mail
- Formatação
- Arquivamento
- Pesquisa Internat
- Importação/Exportação

Instalando o WordPress

O WordPress é um dos CMS's mais fáceis de se utilizar, e sua instalação não poderia ser mais fácil. Isto supondo que o usuário já possua alguma experiência com sistemas similares, e também com a criação de bancos de dados, definição de usuários, senhas, upload de arquivos, etc.

Não é complicado, mas aqui será aplicado um tutorial básico voltado principalmente aos usuários que estão iniciando o uso do WordPress.

Pacote de instalação

Você pode baixar do site internacional, ou da comunidade brasileira, que já vem em português.

Após o download, efetue a descompactação do arquivo .ZIP obtido, e você obterá uma pasta chamada “wordpress”, com uma série de arquivos e também 3 subpastas, conforme abaixo:

- wp-admin: arquivos de administração (Painel, instalador, ...)
- wp-content: arquivos do website (Uploads, temas, plugins, cache...)
- wp-includes: bibliotecas e classes (jQuery, SimplePie, FCKEditor, Prototype...)

Configuração

Este é um procedimento que poderá variar de acordo com o sistema operacional, painel de controle e recursos disponibilizados no plano de hospedagem que você assinou. Em alguns painéis de controle, como o Cpanel, a criação de bases de dados e usuários, e também a definição de acessos, pode ser feita através da opção “MySQL Databases” (bases de dados MySQL), localizada dentro do grupo de opções “Databases” (bases de dados).

Algumas empresas de hospedagem trabalham com outros painéis de controle, onde este procedimento pode variar bastante, e existem até mesmo empresas que trabalham com painéis de controle “próprios”.

Ou seja, é muito difícil encontrarmos um “guia definitivo” relativo a este ponto, mas vale ressaltar que, independentemente do painel de controle, do sistema operacional ou até mesmo da empresa de hospedagem que você utiliza, o que você precisa fazer é o seguinte:

- Criar uma base de dados MySQL;
- Criar um usuário e definir uma senha para a base;
- Disponibilizar para este usuário “acesso” completo à base de dados recém criada.

Alguns painéis de controle criam automaticamente o usuário e definem uma senha, outros não permitem a definição de um nome para a base de dados que é fornecida automaticamente. De qualquer forma, após os procedimentos acima você terá em mãos o “nome da base de dados”, o “usuário” que acessará esta base de dados e a “senha”.

Utilizando um editor de textos qualquer, abra o arquivo “wp-config-sample.php”. Ele se encontra da seguinte maneira:

```
define('DB_NAME', 'nomedoBD');
define('DB_USER', 'usuarioMySQL');
define('DB_PASSWORD', 'senha');
define('DB_HOST', 'localhost');
```

Você deverá alterar somente os seguintes dados:

- *nomedoBD:* nome da base de dados;
- *usuarioMySQL:* usuário com acesso à base de dados;
- *senha:* substitua pela senha do usuário com acesso à base de dados;
- *localhost:* geralmente este valor não precisa ser alterado, a não ser que o provedor de hospedagem possua servidores diferentes para arquivos e base de dados;

Após as alterações acima, salve o arquivo com o seguinte nome: wp-config.php e acesse o endereço do website.

```
http://www.seusite.com.br/wp-admin/
```

Temas

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 do seu site é 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. Digamos 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 está limitado pela sua imaginação.

Como instalar novos temas

Se o tema que você está instalando fornece instruções, certifique-se de ler e seguir as instruções para a instalação ser bem-sucedida. É recomendável que os desenvolvedores do tema ofereçam instruções de instalação para os seus próprios temas, pois os temas podem ter funcionalidades opcionais ou especiais que podem requerer medidas a mais do que as etapas de instalação discutidas aqui. Se o tema não funciona depois de seguir as instruções fornecidas, entre em contato com o autor do tema para a ajuda.

Para adicionar um novo tema para sua instalação do WordPress, siga estas etapas básicas:

1. Baixe o arquivo do tema e extraia os arquivos que ele contém. Pode ter necessidade de preservar a estrutura de diretórios no arquivo quando extrair esses arquivos. Siga as orientações fornecidas pelo autor do tema.
2. Usando um cliente de FTP para enviar ao seu servidor web, crie um diretório para conter o seu tema no diretório wp-content/themes fornecidos pelo WordPress. Por exemplo, um tema chamado de "teste" deve ser enviado para wp-content/themes/test. Seu tema pode ter este diretório, como parte do arquivo.

3. Upload os arquivos para o novo diretório no seu servidor.
4. Siga as instruções abaixo para selecionar o novo tema.

Adicionando novos temas utilizando o Painel de Administração

Você pode baixar temas diretamente para o seu blog, usando a opção Adicionar novo Tema no sub-menu Aparência.

1. Ir no Painel de Administração.
2. Selecione Aparência e então Temas.
3. Selecione Adicionar Novo Tema
4. Use o sub-menu ou a pesquisa (se quiser marque opções de filtro) para localizar um tema que você gostaria de usar.
5. Clique no link Visualizar para visualizar o tema antes de fazer download e instalar.
6. Utilize o link Download no topo do sub-menu para fazer download do tema.

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.
...
*/
```

Um tema é composto de dois arquivos básicos e obrigatórios.

- style.css
- index.php

Os modelos WordPress se encaixam como peças de um quebra-cabeça para gerar as páginas em seu site. Alguns modelos são usados em todas as páginas da web (como o cabeçalho e o rodapé, e estilos, por exemplo), enquanto outros são usados somente em condições específicas.

A pergunta a se fazer é: Qual arquivo de modelo(s) o WordPress usará quando ele exibe um certo tipo de página?

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 parte das 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 solicitada.

Os modelos são então escolhidos - e o conteúdo da página web é gerada - na ordem sugerida pela hierarquia Template WordPress, dependendo de quais modelos estão disponíveis num determinado Tema 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.
- *index.php:* O template principal. É usado para exibir qualquer conteúdo quando um template específico não for encontrado.
- *comments.php:* Listagem de comentários logo abaixo dos posts.
- *comments-popup.php:* Lista de popups em uma nova janela aberta via Javascript.
- *home.php:* A capa do site.
- *single.php:* Um post sozinho.
- *page.php:* Uma página.
- *category.php:* Listagem de posts de uma categoria.
- *tag.php:* Listagem de posts de um tag.
- *taxonomy.php:* Listagem de uma taxonomia qualquer.
- *author.php:* Listagem de posts de um autor.
- *date.php:* Listagem de um intervalo de data (ano, mês, dia).
- *archive.php:* Usado de maneira generalista para category.php, author.php, e date.php.
- *search.php:* Resultados de uma busca.
- *404.php:* Página de erro para conteúdo não encontrado.

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 variandos suas derivações de nome, ou mesmo alterando seu código com expressões que usam os mesmos tags condicionais.

Exemplos

Se o seu blog está em http://example.com/wp/ e um visitante clica em um link para uma página da categoria como http://example.com/wp/category/your-cat/, WordPress procura por um modelo arquivo no diretório do tema atual que corresponde ao ID da categoria.

Se a categoria de identificação é 4, o WordPress procura por um arquivo de modelo category-4.php . Se estiver em falta, o WordPress procura por um arquivo de modelo genérico,

category.php .

Se este arquivo não existe o WordPress procura por outro arquivo de modelo genérico, archive.php.

Se não exisitir, então ele procura pelo arquivo principal, index.php .

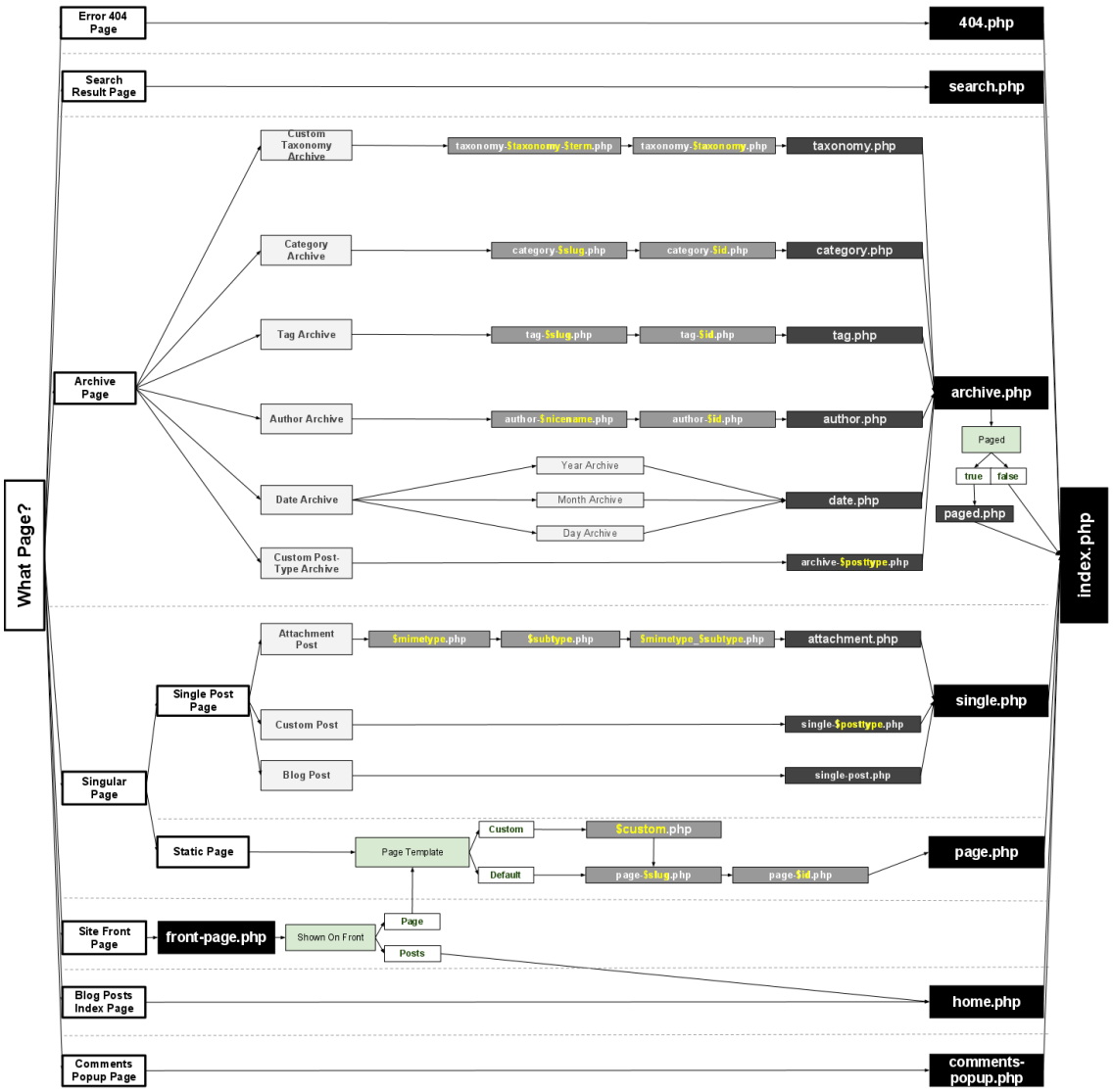
Se um visitante vai para sua homepage na http://example.com/wp/, o WordPress primeiro determina se há uma Página estática. Se uma página estática foi definida, o WordPress exibe primeiro a página principal e depois esta página definida, de acordo com o modelo de hierarquia.

Se uma página estática não foi definida, então o WordPress procura por um arquivo home.php e o usa para gerar a página solicitada.

Se não há arquivo home.php, o WordPress procura por um arquivo chamado index.php no diretório do tema ativo, e usa esse modelo para gerar a página.

A Hierarquia de Modelos em detalhe

O seguinte diagrama mostra quais arquivos de modelo são chamados para gerar uma página do WordPress com base na hierarquia de Template WordPress.



As seções seguintes descrevem a ordem na qual os arquivos de modelo são chamados pelo WordPress, para cada tipo de consulta.

Exibir Home page

1. home.php
2. index.php

Exibir Postagem Individualmente

1. single.php
2. index.php

Exibir Página

1. custom template - Se a página tem um modelo próprio definido.
2. page-{slug}.php - Se o slug da página é sobre-mim, o WordPress will look to use page-sobre-mim.php
3. page-{id}.php - Se o ID de tal página é 6, o WordPress procurará por page-6.php
4. page.php
5. index.php

Mostrar Categoria

1. category-{slug}.php - Se o slug de uma categoria for dicas, então o WordPress procurará por category-dicas.php
2. category-{id}.php - Se o IDde uma categoria for 12, então o WordPress procurará por category-12.php
3. category.php
4. archive.php
5. index.php

Mostrar Tag

1. tag-{slug}.php - Se o slug de uma tag for especies, então o WordPress procurará por tag-especies.php
2. tag-{id}.php - Se ID de uma tag for 6, então o WordPress procurará por tag-6.php
3. tag.php
4. archive.php
5. index.php

Custom Post Tipos

1. single-{post_type}.php - Se o tipo de post for algum-post, então o WordPress procurará por single-algum-post.php
2. single.php
3. index.php

Taxonomias Personalizadas

1. taxonomy-{taxonomy}-{term}.php - Se a taxonomia for vertebrados , e o slug dessa taxonomia for primatas, o WordPress procurará por taxonomyvertebrados-primatas.php
2. taxonomy-{taxonomy}.php - Se a taxonomia for invertebrados, o WordPress procurará por taxonomyinvertebrados.php
3. taxonomy.php
4. archive.php
5. index.php

Mostrar Dados de Autor

1. author-{nicename}.php - Se o "nicename" do autor for dianakc, o WordPress procurará por author-dianakc.php
2. author-{id}.php - Se o ID de um autor for 10, o WordPress procurará por author-10.php
3. author.php
4. archive.php
5. index.php

Arquivo por Data

1. date.php
2. archive.php
3. index.php

Exibir Resultados da Pesquisa

1. search.php
2. index.php

Exibir página 404 (Não Encontrado)

1. 404.php
2. index.php

Exibir Anexos

1. MIME_type.php - qualquer tipo MIME (image.php, video.php, audio.php, application.php ou qualquer outros).
2. attachment.php
3. single.php
4. index.php

Referenciando arquivos em templates

Ao chamar arquivos via HTML dentro dos templates, você deve sempre chamá-los com uma URL completa, tais como:
http://www.meusite.com.br/wp-content/uploads/2010/03/04/meuarquivo.ext.

Para facilitar reescritas de endereço e fazer com que o tema seja portável, use a função bloginfo.

```
<?php bloginfo('url'); ?>
<!-- resulta em http://www.meusite.com.br -->
<?php bloginfo('template_directory'); ?>
<!-- resulta em http://www.meusite.com.br/wpcontent/
themes/meutema -->
```

Dentro do arquivo CSS não é necessário oferecer o endereço completo de imagens, uma vez que elas serão sempre relativas à folha de estilo.

```
h1 {background-image: url('images/my_background.jpg');}
```

Ganchos para Plugins

Sempre inclua chamadas no seu tema, para que o WordPress e eventuais plugins possam saber o que acontece durante o carregamento da página. Se um plugin insere JavaScript no final das páginas, ele precisará do gancho wp_footer, por exemplo.

```
<?php
// Antes de fechar a tag <head>:
wp_head();
// Antes de fechar a tag <html>:
wp_footer();
?>
```

Template tags

Template Tags são utilizadas nos modelos do seu blog para exibir informações de forma dinâmica ou personalizar seu blog, fornecendo as ferramentas para possibilitar a você desfrutar ao máximo das possibilidades de personalização que o WordPress oferece. Abaixo está uma lista das tags que estão disponíveis no WordPress, classificadas por categoria de função específica.

O Loop

O loop é a estrutura básica do WordPress. Assim chamado, ele recebe uma query string e executa tudo o que está em seu interior, instanciando as variáveis em um objeto que serve de referência para funções sintaxicamente comuns e fáceis de lembrar.

Começamos verificando se existem posts chamados e instanciando as variáveis, e terminamos fechando os comandos PHP.

```
<?php if (have_posts()) : while (have_posts()): the_post(); ?>
<!-- Aqui colocamos tudo que queremos que seja feito com o conteúdo
chamado. -->
<?php endwhile; else: ?>
<!-- Uma mensagem dizendo que nenhum conteudo foi encontrado. -->
<?php endif; ?>
```

Exemplos de loop

Loop básico com conteúdo:

```
<!-- Comeca o loop. -->
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
<!-- Mostra o titulo como um link para o post. -->
<h2>
<a href="<?php the_permalink() ?>" rel="bookmark"
title="Link para <?php the_title_attribute(); ?>">
<?php the_title(); ?>
</a>
</h2>
<!-- Mostra a data no formato: November 16th, 2009, e faz um link para
outros posts deste autor. -->
<small>
<?php the_time('F jS, Y') ?> por <?php the_author_posts_link() ?>
</small>
<!-- Mostra o conteudo dentro de uma caixa div. -->
<div class="entry">
<?php the_content(); ?>
</div>
<!-- Mostra uma chamada para os comentarios -->
<?php comments_popup_link(); ?>
<!-- Mostra uma lista separada por virgulas das categorias. -->
<p class="postmetadata">
Posted in <?php the_category(', '); ?>
</p>
<!-- Para o loop. -->
<?php endwhile; else: ?>
<!-- Mostra uma mensagem caso nenhum post tenha sido encontrado. -->
<p>Sorry, no posts matched your criteria.</p>
<!-- REALMENTE para o loop. -->
<?php endif; ?>
```

Excluir posts da categoria 3:

```
<?php query_posts($query_string.'&cat=-3,-8'); ?>
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
<!-- Repeticao. -->
```

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

Mostra uma lista de posts duas vezes:

```
<?php while (have_posts()): the_post();?>
<!-- Primeira vez. -->
<?php endwhile; ?>
<?php rewind_posts(); ?>
<?php while (have_posts()): the_post();?>
<!-- Segunda vez. -->
<?php endwhile; ?>
```

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;?>
```

Se for necessário manter a query original, você pode criar um novo objeto:

```
<!-- Salva a query original. $wp_query e uma variavel global definida
pelo ambiente. -->
<?php $query_original = $wp_query; ?>
<!-- Faz uma query personalizada qualquer. -->
<?php $nova_query = new WP_Query('category_name=categoria');?>
<?php while ($nova_query->have_posts()) : $nova_query->the_post(); ?>
<!-- Faz qualquer coisa. -->
<?php endwhile; ?>
<!-- A funcao query_posts sobreescreve a variavel global $wp_query,
que e a query original. -->
<?php query_posts('category_name=outra_categoria'); ?>
<?php while (have_posts()) : the_post(); ?>
<!-- Faz qualquer coisa com a query construida. -->
<?php endwhile; ?>
<!-- Restora a variavel global $wp_query para a query original. -->
<?php $wp_query = $query_original; ?>
```

Plugins

Um plugin para o WordPress é um adendo de código capaz de interferir e modificar suas funcionalidades. Isto deve ocorrer principalmente através da API de chamadas que o core do WordPress executa a medida que desenvolve o processamento da página.

A facilidade desta categoria de desenvolvimento é uma peça chave do sucesso do WordPress. Para quase toda funcionalidade que se imagine, há um plugin.

Estrutura

Um plugin é basicamente um arquivo PHP com um cabeçalho que será usado para exibição na página administrativa de plugins. Este arquivo pode ser bem simples ou pode

chamar vários outros scripts, criar tabelas no banco de dados, e elaborar toda uma estrutura de dados própria, ficando bem complexo.

O seguinte modelo de cabeçalho é usado para que o WordPress detecte um plugin colocado dentro da pasta “wp-content/plugins/”. Sem este cabeçalho um plugin sequer pode ser ativado no painel de administração.

```
<?php
/*
Plugin Name: Nome do Plugin
Plugin URI: http://URI_do_plugin
Description: Descricao de ate 140 caracteres.
Version: Numero de versao
Author: Nome do autor
Author URI: http://URI_do_autor
License: Um "slug" da licenca, algo como GPL2
*/
?>
```

API de Plugins

O WordPress quando é carregado pelos navegadores, para cada etapa deste carregamento ele faz uma pequena verificação para checar se há algum plugin ou tema que pede para que algo seja executado ali naquele momento. Neste caso, o que será executado é uma função “enganchada”, (e daí o nome hook). Estas funções são chamadas por estruturas divididas em duas categorias, de acordo com a sua funcionalidade:

- Ações: eventos disparados em certos momentos durante a execução;
- Filtros: lançados no momento de gravação ou apresentação de texto;

Os arquivos dos plugins funcionam principalmente à base de ganchos (hooks), e aqui é muito importante a fixação deste conceito.

Estas estruturas são “verificadores” durante o processamento do WordPress. Através de um gancho você atrela determinada função à um evento, acabando por executar esta função quando este evento ocorrer no carregamento.

Para definir os ganchos você pode se perguntar “Em que momento devo fazer isso?” para cada ação e então verificar a lista de ganchos de ações e filtros para achar qual lhe atende. Existem ações imagináveis para qualquer situação do WordPress. Dê uma olhada na referência de ações, pois esta parte do desenvolvimento é bem flexível.

Por exemplo, se quisermos exibir um alerta em Javascript de que a página terminou de ser carregada para toda e qualquer página de uma instalação, podemos fazer assim:

```
<?php
add_action ('wp_footer', 'exibe_alerta');
function exibe_alerta(){
?>
<script type=“text/javascript”>alert(“Pagina carregada!”);</script>
<?php
}
?>
```

Em bom português, “add_action ('wp_footer', 'exibe_alerta')” pede ao WordPress: “Execute a função 'exibe_alerta' quando você passar pela tag HTML </body>”.

A ação wp_footer, simplesmente executa a função especificada quando o “<?php wp_footer(); ?>” for alcançado no tema. Obviamente, esta ação depende que o tema utilizado pela instalação do WordPress atenda os padrões de desenvolvimento requeridos, tais como colocar as funções “wp_footer” no final do rodapé e “wp_head” no final do cabeçalho.

Outro exemplo é um gancho que notifique o administrador do site via e-mail a respeito de todas as publicações que os usuários de um site fazem.

```
<?php
add_action('publish_post','avisar_admin');
function avisar_admin($id){
$p=get_post($i=$id);
$email=get_option('admin_email');
$assunto="Novo post: $p->post_title";
$conteudo=apply_filters('the_content',$p->post_content);
if(wp_mail ($email, $assunto, $conteudo))
return true;
return false;
}
?>
```

Este pequeno trecho de código assume o conhecimento de algumas propriedades do WordPress. Vamos por partes.

Antes de qualquer coisa, registramos o que queremos fazer. A função “add_action ('publish_post', 'avisar_admin')” registra que a função “avisar_admin” deve ser lançada sempre um post for publicado. Esta ação passa o ID do post publicado para a função especificada, o que é obrigatório caso queiramos utilizar informações deste post na função.
A função “$p = get_post ($i = $id)” retorna um objeto post, que são todas as informações do post especificado por “$id”, tais como o conteúdo, autor, data de publicação, etc. Com isso temos o objeto em mãos para utilizar onde queremos.

A função “get_option()” consulta diversas opções centrais do blog, tais como o endereço, pasta de uploads, título e subtítulo, etc. Aqui queremos o e-mail do administrador.

Colocamos no assunto uma descrição que envolva o título do post publicado. O item “post_title” do objeto “$p” é este título.

A função “apply_filters” aplica o filtro do parâmetro um ao parâmetro dois. Há um filtro padrão do WordPress chamado “the_content”, este filtro formata um texto para adequar-se ao conteúdo de um post. Coisas em HTML como adicionar quebras de linha, abertura e fechamento de parágrafos, ou correções de tags quebradas. Também adiciona molduras legendadas às fotos.

Aqui a usamos para formatar a mensagem que será enviada por e-mail. A função wp_mail utiliza o recurso do servidor mais apropriado para o envio de correio eletrônico. Há plugins que usam esta função para automatizar autenticações SMTP, limpa de tags HTML etc., motivo pelo qual simplesmente não é usado a “mail()”, do PHP.

Retorna portanto, verdadeiro se um e-mail foi enviado, ou falso caso haja falha. Assim o $email receberá uma $mensagem com o $assunto personalizado para qualquer publicação.

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 usadas, a princípio, para desenvolvimento de Plugins. O resto é usado para criar as funcionalidades núcleo do WordPress.

Muitas das funções núcleo do WordPress são úteis aos desenvolvedores de Temas e Plugins. Esta é uma lista com 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

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
- 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

Usuários e Autores

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

- 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
- 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
- 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
- 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
- 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
- _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_nonce_field
- wp_nonce_url
- wp_notify_moderator
- wp_notify_postauthor
- wp_original_referer_field
- wp_redirect
- wp_referer_field
- wp_remote_fopen
- wp_salt
- wp_setcookie
- wp_upload_bits
- wp_upload_dir
- wp_verify_nonce

Referência de Ações

Requisições típicas

- muplugins_loaded
- load_textdomain
- update_option
- plugins_loaded
- load_textdomain
- sanitize_comment_cookies
- setup_theme
- load_textdomain
- auth_cookie_malformed
- set_current_user
- init
- widgets_init
- load_textdomain
- parse_request
- send_headers
- pre_get_posts
- posts_selection
- wp
- template_redirect
- get_header [first printed output to the browser]
- wp_head
- wp_enqueue_scripts
- wp_print_styles
- wp_print_scripts
- loop_start
- the_post
- loop_end
- get_footer
- wp_footer
- wp_print_footer_scripts

Páginas administrativas

- plugins_loaded
- sanitize_comment_cookies
- auth_cookie_malformed
- auth_cookie_valid
- set_current_user
- init
- admin_init
- parse_request
- send_headers
- admin_head
- admin_footer

Posts, páginas, anexos e categorias

- add_attachment
- add_category
- clean_post_cache
- create_category
- delete_attachment
- delete_category
- delete_post
- deleted_post
- edit_attachment
- edit_category
- edit_post
- pre_post_update
- private_to_publish
- publish_page
- publish_phone
- publish_post
- save_post
- wp_insert_post
- xmlrpc_publish_post

Comentários, pings e trackbacks

- comment_closed
- comment_id_not_found
- comment_flood_trigger
- comment_on_draft
- comment_post
- edit_comment
- delete_comment
- pingback_post
- pre_ping
- trackback_post
- wp_blacklist_check
- wp_set_comment_status

Links

- add_link
- delete_link
- edit_link

Feeds

- atom_entry
- atom_head
- atom_ns
- commentrss2_item
- do_feed_(feed)
- rdf_header
- rdf_item
- rdf_ns
- rss_head
- rss_item
- rss2_head
- rss2_item
- rss2_ns

Templates

- comment_form
- do_robots
- do_robotstxt
- get_footer
- get_header
- switch_theme
- template_redirect
- wp_footer
- wp_head
- wp_meta
- wp_print_scripts

Administração

- activate_(plugin file name)
- activity_box_end
- add_category_form_pre
- admin_head
- admin_init
- admin_footer
- admin_print_scripts
- admin_print_styles
- check_passwords
- dbx_page_advanced
- dbx_page_sidebar
- dbx_post_advanced
- dbx_post_sidebar
- deactivate_(plugin file name)
- delete_user
- edit_category_form
- edit_category_form_pre
- edit_tag_form
- edit_tag_form_pre
- edit_form_advanced
- edit_page_form
- edit_user_profile
- load-(page)
- login_form
- login_head
- lost_password
- lostpassword_form
- lostpassword_post
- manage_link_custom_column
- manage_posts_custom_column
- manage_pages_custom_column
- password_reset
- personal_options_update
- plugins_loaded
- profile_personal_options
- profile_update
- register_form
- register_post
- restrict_manage_posts
- retrieve_password
- set_current_user
- show_user_profile
- simple_edit_form
- update_option_(option_name)
- upload_files_(tab)
- user_register
- wp_ajax_(action)
- wp_authenticate
- wp_login
- wp_logout

Avançado

- admin_menu
- admin_notices
- blog_privacy_selector
- check_admin_referer
- check_ajax_referer
- generate_rewrite_rules
- init
- loop_end
- loop_start
- parse_query
- parse_request
- pre_get_posts
- sanitize_comment_cookies
- send_headers
- shutdown
- wp