

Aprenda a desenvolver

Temas WordPress

Guilherme Mazetto

www.guiawp.com.br

Aprenda a desenvolver
Temas WordPress

Guilherme Mazetto
www.guiawp.com.br

Prefácio

A internet é lugar de todos. Boa parte dos usuários de internet utilizam-se dos blogs como meio de comunicação e referência para buscar informações úteis aos seus cotidianos. A vontade de se expressar e expor pontos de vista no mundo digital é tão grande que o número de produtores de conteúdo cresce a cada dia.

Deparado com tal situação é possível ver a necessidade desses produtores em entender o funcionamento da publicação de material para web. Com intuito de agilizar esse trabalho surgem os sistemas gerenciadores de conteúdo: interfaces altamente simplificadas e auto-explicativas capazes de realizar funções anteriormente conseguidas apenas com um bom conhecimento técnico.

No cenário existente, o WordPress destaca-se como o mais popular entre os sistemas gerenciadores por possuir uma forte comunidade colaborativa, interface amigável, elevado número de funcionalidades e diversas razões mais.

Tamanha popularidade agrava o problema da personalização dos trabalhos, onde muitos possuem o sistema e compartilham dos mesmos recursos e elementos visuais a serem exibidos aos visitantes.

O trabalho proposto tratará fundamentalmente de torná-lo capaz de desenvolver suas próprias soluções para exposição de conteúdos mostrando um modo inteligente de integrar o desenvolvimento web padrão ao sistema do WordPress de modo a aproveitar bem os seus recursos.

Sumário

Prefácio.....	3
Sumário.....	4
Conteúdo.....	5
Introdução.....	12
Desenvolvimento Web.....	17
Início dos trabalhos.....	29
Construção do tema.....	47
Estrutura do tema.....	67
Aprimoramentos.....	77
Considerações Finais.....	96
Apêndice A: Referência de funções.....	97
Apêndice B: Funções utilizadas.....	161

Conteúdo

Prefácio.....	3
Sumário.....	4
Conteúdo.....	5
Introdução.....	12
Para quem é feito o livro?.....	13
Projeto.....	14
Didática.....	14
Notificações.....	15
Alerta.....	15
Notas adicionais.....	15
Anexos.....	15
Ajuda.....	16
Desenvolvimento Web.....	17
Conceitos básicos.....	19
W3C e Padrões web.....	19
HTML.....	19
XHTML.....	19
CSS.....	20
Linguagens de Navegadores (Browser Scripting).....	20
Linguagem de Servidores (Server Scripting).....	20
Banco de Dados.....	21
PHP.....	21
jQuery.....	22

Tableless.....	22
MySQL.....	22
CMS.....	22
Wordpress.....	23
Plugins.....	25
Temas.....	26
Porquê desenvolver um tema WordPress?.....	27
Início dos trabalhos.....	29
Wireframe.....	32
Padrão de codificação WordPress.....	34
HTML.....	34
Validação.....	34
Fechamento de Tags.....	34
Atributos e tags.....	35
Aspas.....	35
Indentação.....	36
PHP.....	37
Aspas simples e duplas.....	37
Indentação.....	37
Estilo das Chaves.....	38
Uso de espaços.....	39
Variáveis, funções, nomes de arquivos, e operadores	39
Sinalização de parâmetros.....	40
Internacionalização.....	41
Arquivos POT.....	41
Arquivos PO.....	41

Arquivos MO.....	42
Criando entradas.....	42
Criando um template web.....	44
index.php.....	45
style.css.....	45
Construção do tema.....	46
Estilo.....	47
Template Tags.....	49
Cabeçalho.....	49
Dados não confiáveis.....	53
Arquivo de Funções.....	54
Menus.....	54
Sidebar.....	56
Navegação.....	56
Listas aninhadas.....	56
Widgets.....	57
Personalizando a Sidebar.....	58
Rodapé.....	60
Trabalhando o conteúdo.....	61
Loop.....	61
The e Get_the.....	62
Plugin API Hooks.....	64
Actions.....	64
Filters.....	64
Estrutura do tema.....	66
Hierarquia.....	67

Arquivos Modelos.....	69
Includes Tags.....	71
Incluindo arquivos.....	72
Single.php.....	74
Aprimoramentos.....	76
Resumo.....	78
Search.....	78
Conditional Tags.....	79
Archive.....	80
Posts e páginas.....	82
Modelos de Páginas.....	83
Criando um novo modelo de página.....	83
Páginas adicionais.....	84
Sem comentários.....	84
Sem Sidebar.....	84
Página 404.....	85
Classes do Tema.....	86
Javascript.....	87
Registrando scripts.....	88
Validação do formulário.....	88
Folha de estilos.....	90
Classes do WordPress.....	90
Registrando estilos.....	91
Tradução.....	92
PoEdit.....	92
Traduzindo.....	93

Considerações Finais.....	95
Apêndice A: Referência de funções.....	96
__.....	97
_e.....	98
bloginfo.....	99
body_class.....	100
comments_popup_link.....	101
comments_template.....	102
dynamic_sidebar.....	103
esc_attr.....	104
get_author_posts_url.....	105
get_day_link.....	106
get_footer.....	107
get_header.....	108
get_month_link.....	109
get_option.....	110
get_search_form.....	111
get_search_query.....	112
get_sidebar.....	113
get_template_part.....	114
get_the_author.....	115
get_the_author_meta.....	116
get_the_category.....	117
get_the_category_list.....	118
get_the_date.....	119
get_the_tag_list.....	120

get_the_tags.....	121
get_userdata.....	122
get_year_link.....	123
have_posts.....	124
is_author.....	125
is_category.....	126
is_day.....	127
is_home.....	128
is_month.....	129
is_page.....	130
is_single.....	131
is_tag.....	132
is_year.....	133
language_attributes.....	134
load_theme_textdomain.....	135
next_posts_link.....	136
post_class.....	137
previous_posts_link.....	138
register_nav_menu.....	139
single_tag_title.....	140
single_cat_title.....	141
register_sidebar.....	142
the_author.....	143
the_date.....	144
the_excerpt.....	145
the_permalink.....	146

the_post.....	147
the_search_query.....	148
the_title.....	149
wp_enqueue_script.....	150
wp_enqueue_style.....	151
wp_footer.....	152
wp_get_archives.....	153
wp_head.....	155
wp_nav_menu.....	156
wp_title.....	159
Apêndice B: Funções utilizadas.....	160

Introdução

O livro aborda os conceitos essenciais para desenvolvimento de um layout e integração do mesmo com o WordPress. Uma seqüência lógica foi criada com o intuito de facilitar o entendimento do mesmo; atribuindo, apresentando e explicando novos conceitos à medida em que serão empregados no processo de desenvolvimento do tema. Tal processo será tratado no decorrer de 5 principais partes, são elas:

Desenvolvimento Web

Uma breve apresentação de conceitos que devem ser de conhecimento do leitor antes de qualquer atividade relacionada ao desenvolvimento propriamente dito. O aprendizado do funcionamento das questões tratadas implica numa melhor assimilação do restante do material.

Início dos trabalhos

Etapas iniciais de criação de um template web. São apresentados os padrões estabelecidos pelo WordPress em sua programação e a capacidade de um Tema tornar-se internacional com suporte a múltiplos idiomas.

Construção do Tema

A integração do template criado com WordPress tem seu início. O modo como o modelo é tratado no sistema e a criação de arquivos padrão podem ser vistos em amplo funcionamento.

Estrutura do Tema

O tema em construção passa por modificações estruturais. Novos arquivos são criados. Conheça as técnicas de reutilização de modelos em diferentes arquivos do Temas.

Aprimoramentos

São expostas muitas funcionalidades que permitem tornar um tema qualquer em outro mais sofisticado com mais recursos e interatividade com o visitante e as páginas que esse visita.

Para quem é feito o livro?

Para desenvolver as atividades propostas é recomendado que você tenha um breve conhecimento de HTML, CSS e PHP. Essas noções básicas facilitarão e muito o entendimento dos códigos que serão exibidos.

Alguns conceitos das linguagens citadas, como a correta marcação de tags, laços de repetição, declaração de variáveis, seletores e outros; serão citados, porém superficialmente. A razão disso é o nosso propósito: criação de Temas WordPress e não o de aprendizado de uma linguagem específica.

O aprendizado é constante, relacionado a ele estará a qualidade de seu trabalho. Para obter melhores resultados, não deixe de pesquisar os itens mencionados e sempre pratique os conceitos explorados fazendo seus próprios Temas WordPress.

Projeto

Durante os capítulos desse livro iremos desenvolver um Tema WordPress na íntegra. Além do Tema finalizado, ao término do livro você estará apto a desenvolver suas próprias criações uma vez que os conceitos estarão bem sedimentados.

Você entenderá como é o funcionamento do WordPress como um todo e com relação a interpretação e exibição do tema criado. Saberá o significado dos termos técnicos e estará capacitado a entender e utilizar-se de outros e novos recursos da ferramenta.

Didática

Desenvolveremos nosso tema passo a passo, entendendo e colocando em prática algumas das principais funções do WordPress. A apresentação das funções será feita gradativamente com as etapas a serem desenvolvidas. Elas surgirão avulsas em meio as explicações, quando aplicadas ao tema. Para uma descrição mais detalhada consulte o apêndice de funções.

A melhor maneira de se aprender a fazer algo é fazendo. Por isso é muito importante que ao ler esse livro, você vá reproduzindo as etapas ao mesmo passo em que evolui as páginas. A cada função apresentada, coloque-a em prática em seu tema próprio. Caso você tenha dificuldades para iniciar a criação de algo, utilize os arquivos da produção do livro. Comece do princípio e não pule etapas, isso será fundamental para um melhor proveito do material que está lendo.

Notificações

Em determinados pontos do livro será preciso incluir determinadas informações para que o estudo seja mais dinâmico e acrescente conteúdo relevante ao que está em pauta. A maneira encontrada para atingir esse objetivo foi criar notificações que enfatizam o que está sendo visto. Veja quais são as notificações e suas respectivas funções dentro do livro:

**Alerta**

Você deve dar atenção para determinados pontos do texto. Reitera informações e deixa claro circunstâncias onde não podem haver dúvidas.

**Notas adicionais**

São explicações adicionais ao assunto que não necessariamente se enquadram no texto, mas de importante conhecimento.

**Anexos**

Ao final de cada capítulo haverá a referência para o download dos arquivos criados durante o mesmo.

Abaixo está um exemplo de como os códigos serão exibidos dentro do livro. No topo estará o nome do arquivo onde o código está inserido ou aquilo que ele representa.

Exemplo de código – Nome do arquivo

```
// Código Fonte do arquivo
```

Ajuda

Caso você tenha dúvida com o uso do WordPress, instalação de temas, manuseio de arquivos, gerenciamento de informação dentro do CMS no formato de usuário da ferramenta e não do desenvolvedor, acesse www.guiawp.com.br.

No blog você encontrará tudo isso, informações atualizadas e muito mais sobre WordPress. Caso tenha interesse fique à vontade também para expressar sua opinião utilizando a ferramenta de comentários ou então o formulário de contato.

Parte I

Desenvolvimento Web

Desenvolver material de qualidade para web pode se tornar uma tarefa fácil desde que você domine o espaço que pretende adentrar. O web design existe como uma extensão do design, cuja finalidade é justamente a criação de elementos que possam trazer melhores resultados ao material publicado na internet.

Para o processo de criação ser bem realizado é preciso ter em mente que muitos fatores convergem para a produção de um material de qualidade. Entre eles destacam-se a usabilidade, acessibilidade e a arquitetura da informação diretamente relacionadas ao web design transmitindo informações úteis sobre diagramação, layout e disposição dos diferentes componentes em tela.

Tendo em mãos um briefing bem elaborado, abordando as principais questões referentes ao conteúdo do projeto, é necessário definir seus objetivos, público alvo, serviços e produtos explorados, diferenciais de mercado e todas as características que possam determinar o melhor modo de expor tudo em tela.

Um conceito importante a ser explorado é o de usabilidade. Ela define a experiência do visitante no site tornando sua experiência prazerosa ou um tormento como resultado de informações confusas e de becos sem saídas. Uma boa usabilidade mostra ao usuário onde ele está, quais opções tem de prosseguir, como encontrar as informações que procura, além de propiciar uma interface amigável a ele onde todos elementos dispostos em tela devem existir mediante um objetivo.

Sabendo seus objetivos, traçando metas, aplicando bem o layout ao seu projeto; respeitando as funções dos elementos; já traz ao site uma boa aspiração profissional.

Conceitos básicos

W3C e Padrões web

A W3C (World Wide Web Consortium) é o órgão responsável por recomendar padrões de desenvolvimento para a internet. Por meio destes padrões se pode classificar: web sites de acordo com suas características técnicas, indo além do visual e; navegadores, de acordo com sua capacidade em atender aos padrões definidos.

O grande objetivo de seguir os padrões do W3C é de possibilitar que a informação veiculada pelo site permaneça independente do dispositivo utilizado pelo visitante e que seja acessível.

HTML

A Linguagem de Marcação de Hipertexto (HTML - HyperText Markup Language) é uma linguagem de marcação utilizada para produzir páginas na Web que são interpretadas pelos navegadores. Os documentos em HTML são arquivos de texto simples que podem ser criados e editados em qualquer editor de textos comum, como o Bloco de Notas do Windows.

XHTML

A XHTML (Extensible Hypertext Markup Language) é atualmente a base do Desenvolvimento Web. Ela é a estrutura de toda a informação que é apresentada na Internet, como imagens, textos, formulários, links e muito mais.

CSS

A XHTML depende, em essência, da CSS (Cascading Style Sheets) para formatar a estrutura do seus códigos nos Navegadores de Internet. É uma linguagem de formatação simples e poderosa. Com ela você pode, praticamente, formar qualquer tipo de layout, de maneira muito mais clara e eficiente, se comparada com a antiga formatação incluída em códigos HTML. A CSS anda lado a lado com a XHTML.

Linguagens de Navegadores (Browser Scripting)

As Linguagens de Navegadores são códigos de programação inseridos no código XHTML com a finalidade de incrementar as suas funcionalidades, como inserir data e hora atual, validar formulários, retornar valores matemáticos e muito mais. Esses códigos são interpretados pelos Navegadores de Internet (Browsers) no momento em que as páginas são carregadas (por isso possuem esse nome).

As principais Linguagens de Navegadores utilizadas são o JavaScript e o VbScript, mas o JavaScript é de longe a mais utilizada e aceita pelos navegadores.

Linguagem de Servidores (Server Scripting)

As Linguagens de Servidores talvez sejam os artifícios mais poderosos da Internet. O acesso e a manipulação de dados armazenados em Bancos de Dados são uns dos seus principais recursos, e são amplamente utilizados na Internet.

Ao contrário do que acontece nas Linguagens de Navegadores, as Linguagens de Servidor são processadas nos servidores onde estão armazenadas as páginas, mesmo possuindo, algumas vezes, a codificação na própria página.

Existem várias Linguagens de Servidores disponíveis atualmente. Entre as mais populares estão: ASP, ASP.NET, JSP e PHP. Todas possuem suas vantagens e desvantagens, mas no geral elas realizam as mesmas tarefas.

Banco de Dados

Para os Desenvolvedores que utilizam Linguagens de Servidor é de suma importância conhecer os conceitos de Bancos de Dados Relacionais e algum Sistema de Gerenciamento de Banco de Dados (SGBD). Os Bancos de Dados são utilizados de diversas maneiras na Internet, como armazenar informações sobre produtos de sites de Comércio Eletrônico, manter cadastro de clientes e um infinidade de outras aplicações.

PHP

PHP, que significa PHP: Hypertext Preprocessor, é uma linguagem de programação de ampla utilização, interpretada, que é especialmente interessante para desenvolvimento para a Web e pode ser mesclada dentro do código HTML. A sintaxe da linguagem lembra C, Java e Perl, e é fácil de aprender. O objetivo principal da linguagem é permitir a desenvolvedores escreverem páginas que serão geradas dinamicamente rapidamente, mas você pode fazer muito mais do que isso com PHP.

JQuery

JQuery é uma biblioteca JavaScript que pode ser utilizada e modificada sem qualquer custo. Ajuda os desenvolvedores a se concentrarem na lógica dos sistemas da web e não nos problemas de incompatibilidade dos navegadores.

Tableless

Uma forma de desenvolvimento de sites, sugerida pela W3C, que não utiliza tabelas para disposição de conteúdo na página, pois defende que os códigos HTML devem ser usados para o propósito que foram criados, sendo que tabelas foram criadas para exibir dados tabulares.

MySQL

É um banco de dados relacional que está entre os mais utilizados no mundo. Ele é gratuito (open source) e permite a você armazenar, organizar e ler dados de uma maneira muito rápida e eficiente.

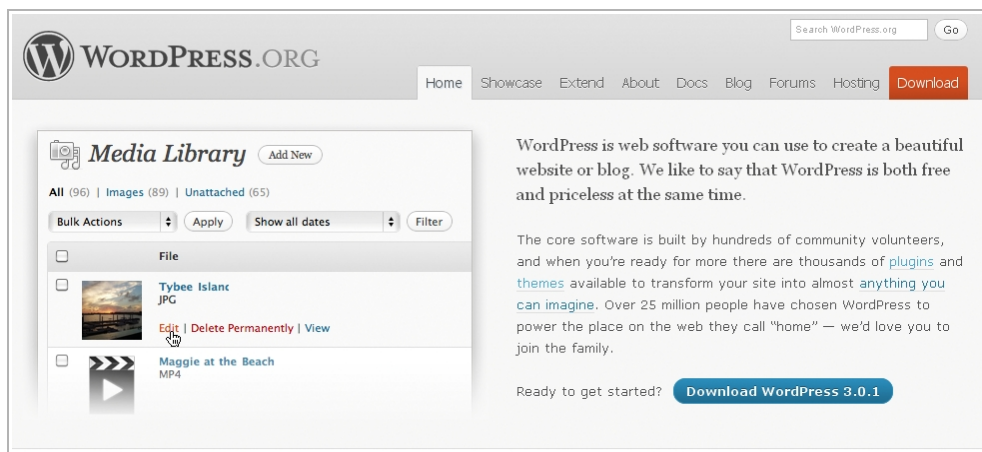
CMS

Sistema de Gerenciamento de Conteúdo (em inglês Content Management Systems). Sistema gestor de websites, portais e intranets que integra ferramentas necessárias para criar e gerenciar conteúdos em tempo real, sem a necessidade de programação de código.

Wordpress

Em sua essência ele é um CMS, gerenciador de conteúdos; voltado especialmente para blogs. Entretanto experiências diversas de seus usuários tem mostrado ele com o potencial para gerir portais e sistemas de diferentes tipos e tamanhos.

O WordPress surgiu em 2003 com um simples código para melhorar a tipografia de escrever todos os dias e com poucos usuários. Atualmente é utilizado e visto em milhões de sites pelo mundo todo. Programado em PHP e base de dados em MySQL; o sistema agrada também os programadores que conseguem dar maior flexibilidade à ele e criar extensões, plugins e temas.



Site do WordPress

O sistema tem como grande diferencial e talvez a resposta para seu rápido crescimento de atuação na internet, o fato do sistema ser código aberto e também possuir interface altamente amigável e personalizável ao usuário.

Para realizar o download do WordPress acesse www.wordpress.org. Esse é o site dos desenvolvedores do sistema, onde encontram-se disponíveis também plugins e temas que você poderá manuseá-los como bem entender.



WordPress.org é o site dos desenvolvedores do WordPress, onde podemos fazer o download do CMS bem como de seus plugins e manuseá-los como bem entender.

WordPress.com é o site que oferece gratuitamente o serviço de hospedagem para o WordPress.

Plugins

Os plugins são funcionalidades que os programadores desenvolvem com as rotinas existentes do WordPress (ou não) e o fazem para obter melhor resposta para diferentes objetivos, seja exibir notícias reacionadas, ou uma galeria de imagens personalizada.

Os plugins são os maiores colaboradores daqueles que entendem pouco de programação, pois fazem tudo o que propõe bastando ao usuário movê-los para o diretório de plugins de seu blog. Muitas vezes possuem páginas administrativas que facilitam ainda mais o trabalho do usuário.

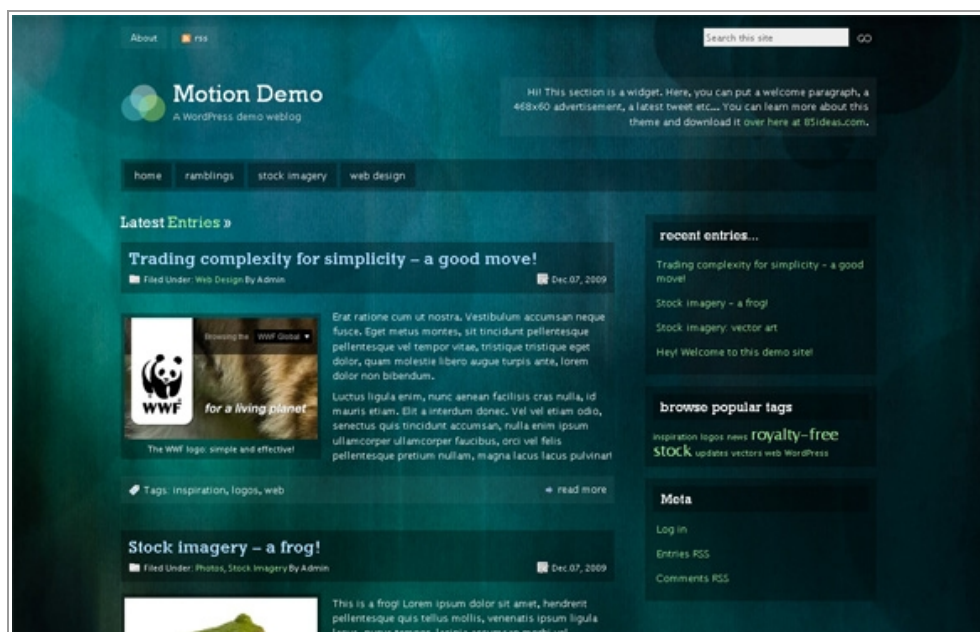
Como o WordPress é projetado para ser leve, maximizar a flexibilidade e minimizar o acúmulo de código; os Plugins existem para oferecer funções personalizadas para que cada usuário possa personalizar seu site segundo suas necessidades específicas.



Entre os plugins mais conhecidos está o Akismet que verifica a presença de spams em meio aos comentários do blog, é muito utilizado e vem como padrão do WordPress.

Temas

O WordPress possui uma distinta separação entre o conteúdo gerenciável de um blog nele criado com a sua respectiva formatação e exibição em tela. O banco de dados MySQL armazena todas as informações passadas ao sistema, este por sua vez possui uma série de funcionalidades que permitem a criação de diferentes modelos de exibição, os temas, para o usuário final.



Motion, um dos muitos temas populares do WordPress

Um tema é justamente a parte visual do seu blog, como ele será apresentado para seu visitante. Ele carrega consigo os elementos visuais que compõe as formatações de páginas e uma série de funcionalidades herdadas do WordPress além de abrir espaço ao desenvolvedor de implantar as suas próprias.


Possui uma coleção de arquivos que trabalha em conjunto para produzir uma interface gráfica única de um blog. Modifica a forma como o site é exibido, sem modificar no entanto, o WordPress onde está sendo executado. Os temas podem incluir arquivos de imagem, folhas de estilos, scripts, bem como quaisquer arquivos de código necessário.

Porquê desenvolver um tema WordPress?

- Para criar um visual único para o seu site;
- Aperfeiçoar temas, funções e recursos existentes do WordPress obtendo melhores resultados;
- Criar modelos alternativos de páginas com características específicas destinadas unicamente a cada tema desenvolvido;
- Oportunidade para aprender mais sobre desenvolvimento web e aumentar sua experiência no ramo;
- Estimula a criatividade;
- Receba críticas de seu trabalho após compartilhar o tema criado. As críticas poderão lhe ajudar a aperfeiçoar suas técnicas;
- Comercialização dos temas criados.

EscritórioDigital

EVENTOSNOVIDADESPRODUTOSSERVIÇOSOBRECONTATO




Decoração de ambiente

Aliquam erat volutpat. Aenean consectetur volutpat erat, et fringilla nisl semper at.

[Leia mais...](#)


Tópicos recentes



Revestimentos e estofados

Nunc tempor tincidunt ipsum egestas semper. Duis cursus risus a sem consequat bibendum. Morbi vestibulum quam ac magna venenatis e...


[Leia mais...](#)



Decoração de ambiente

Aliquam erat volutpat. Aenean consectetur volutpat erat, et fringilla nisl semper at...

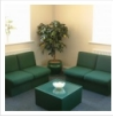
[Leia mais...](#)



Bom iluminação

h2 - Maecenas sapien Lorem, mollis in viverra non, lobortis ut dui. Aliquam mollis blandit nunc, nec tincidunt lacus pellentesque...


[Leia mais...](#)



Móveis de escritório

Aliquam erat volutpat. Phasellus dolor nibh, tempor sed pellentesque sed, ullamcorper eu risus. Morbi tellus nulla, tristique id ...


[Leia mais...](#)



Harmonia de objetos

Nulla vitae elit euismod nulla sodales laoreet id ut ligula. Donec porttitor ante eget risus lacinia at adipiscing...

[Leia mais...](#)



Sala de espera

Vivamus vitae arcu velit. Aliquam aliquet venenatis tellus, nec scelerisque justo dapibus eget. Fusce consectetur justo tincidunt ...

[Leia mais...](#)

Categorias	Arquivo	Lista de Links	Meta
Eventos	setembro 2010	Documentation	Administração
Novidades	agosto 2010	Plugins	Logout
Produtos	julho 2010	Suggest Ideas	Posts RSS
Serviços	junho 2010	Support Forum	RSS dos comentários
	maio 2010	Themes	WordPress.org
	abril 2010	WordPress Blog	
	março 2010	WordPress Planet	

Tema desenvolvido por [WordPressb](#)

Tema Escritório Digital

Aprenda a desenvolver Temas WordPress – www.guiawp.com.br

Parte II

Início dos trabalhos

Ao desenvolver um tema é de suma importância tomar alguns cuidados para otimizá-lo e deixá-lo mais flexível para futuras e eventuais alterações. É preciso ter em mente que o tema envolve somente a estrutura do site e não as informações nele contidas. Assim sendo todo o material que for incluído diretamente através do tema deve ser relevante a ele e também que servirá em todas as suas aplicações.

Como já foi dito anteriormente, no desenvolvimento de sites para web, muitos conhecimentos são colocados em prática. É preciso ter em mente que para se ter um bom resultado final, deve ser conhecido exatamente o quê desejamos ter como resultado final de nossa aplicação; até por razões comparativas. Isso quer dizer que não se pode simplesmente começar logo de cara digitando linhas de código que não se sabe onde te levarão.

Em primeiro lugar é necessário planejar o trabalho a ser executado. Trabalhe bem a idéia do Brainstorm. Nela você e todos aqueles envolvidos na tarefa de desenvolver o tema deverão expressar todas as suas idéias por mais surreias que sejam, sendo possíveis ou não de se fazer; com intuito de se ter um grande volume de informações com as quais trabalhar. Nessa etapa faça questionamentos como:

- Qual assunto será trabalhado?
- Qual tipo de informação será veiculada?
- Como a informação será tratada (páginas, notícias, links)?
- Quais seções o site deverá possuir?

Enfim, questões que possam te dar uma idéia de como poderá ser o seu resultado final. Faça algumas pesquisas na própria web com intuito de obter referências para seu trabalho. Muito material de qualidade pode ser encontrado, até mesmo de forma gratuita.



Tome cuidado e não confunda tomar um trabalho de outra pessoa como referência; com plágio, cópia não autorizada. Ter um tema como referência é acessá-lo e estudar o posicionamento dos componentes, estilos empregados, harmonia do design; para posteriormente em sua criação, trabalhar com algum(s) desses elementos sem que necessariamente seu resultado final seja semelhante ao modelo fonte.

Para ilustrar esse conceito, defini como resultado de um Brainstorm, as seguintes condições para nosso tema:

- Um blog será desenvolvido
- Uma paginação de notícias será necessária
- Páginas de buscas
- Páginas personalizadas
- Deverá exibir nome e descrição do blog
- Necessariamente existirá um menu de páginas
- Espaço para conteúdo interativo, especificado por quem mais for utilizar o tema
- Créditos

Wireframe

Definimos nosso conteúdo e todo o material a ser explorado dentro do nosso site, precisamos agora criar um esboço do nosso trabalho. Faremos então uma prévia, sem programar ainda, que nos possibilite visualizar como será o tema para os visitantes do blog. Para isso criaremos um Wireframe.

O Wireframe é um desenho básico, como um esqueleto, que demonstra de forma direta a arquitetura de como o layout será de acordo com as especificações determinadas; seu objetivo é auxiliar o entendimento dos requisitos que foram escolhidos com relação as funções e objetos que o sistema deverá possuir.

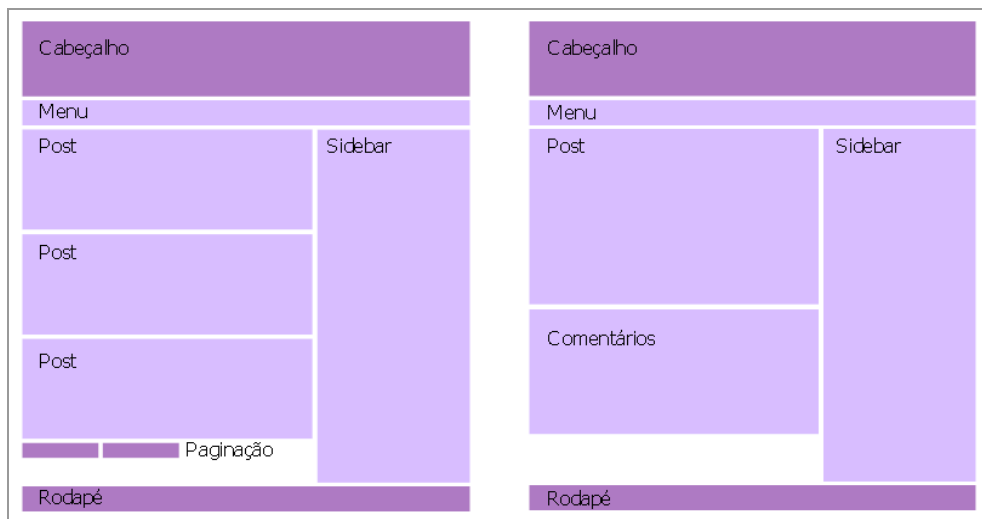
A criação de um wireframe pode ser feita manualmente com materiais de escritório como lápis, caneta, borracha e folhas sulfite. Desse modo você consegue além de planejar o futuro layout do blog, economizar tempo no processo. Ou então você pode utilizar algum editor de imagens (vetoriais preferencialmente) com o qual tenha intimidade com o manuseio e obter excelentes resultados, de melhor qualidade, com a mesma economia de tempo investido.



Importante ressaltar que o resultado final do trabalho a ser desenvolvido independe da ferramenta utilizada.

Para edição dos arquivos PHP, por exemplo, utilizar editores robustos ou o bloco de notas não define o código a ser criado como bom ou ruim, dependerá unicamente do que você irá digitar. O mesmo serve para softwares de criação e edição de imagens.

O tema a ser desenvolvido foi denominado 'Aprendiz'. Nele trabalharemos os principais conceitos e recursos que o WordPress oferece para o processo de desenvolvimento e customização de temas. Por isso, nossos esquemas deverão respeitar os seguintes esboços:



Wireframes do projeto Aprendiz



Os termos vistos como Página, Post, Cabeçalho, Sidebar, Rodapé e muitos outros serão amplamente explorados por tratar-se de um blog e WordPress.

Muitas explicações sobre cada um deles serão realizadas ao decorrer do livro, não se preocupe em memorizá-los agora.

Padrão de codificação WordPress

Antes de desenvolver o tema, existem algumas normas criadas para WordPress com intuito de padronizar seu código-fonte. Seguir tais padrões não é obrigatório, porém é muito interessante quando se quer compartilhar algo desenvolvido e que outras pessoas entendam facilmente o material.

HTML

Validação

Todas as páginas HTML devem ser verificadas pelo validador da W3C certificando que a marcação está sendo bem feita. Esse recurso não necessariamente indica que um código é bom ou ruim, mas ajuda a entendê-lo e encontrar problemas que poderão existir quando o código for aplicado ao servidor. Ainda assim é sempre necessário uma revisão manual do código fonte.

Fechamento de Tags

Todas as tags devem estar fechadas.

Para tags que não possuem fechamento pelo padrão da marcação `<tag></tag>` como o caso das tags `
` e `` a terminação com a barra invertida `"/"` é imprescindível.

Nesses casos ainda o correto é deixar um espaço entre a tag e seu fechamento, passando de `
` na forma incorreta, para `
` no modelo correto dos padrões.

Atributos e tags

Todas as tags e seus atributos devem estar em letras minúsculas.

Os atributos deverão respeitar a norma de caixa baixa quando escritos com o propósito de serem lidos apenas por máquinas. Se a informação do atributo deverá ser interpretado por humanos, deverá respeitar a maior legibilidade dos dados.

Para máquinas

```
<meta http-equiv="content-type" content="text/html" />
```

Para humanos

```
<a href="http://exemplo.com/" title="Descrição">Exemplo.com</a>
```

Aspas

De acordo com o W3C, todos os atributos devem possuir um valor, e deve ser usado para este aspas simples ou duplas necessariamente.

A seguir veja os exemplos do modo correto e incorreto do uso de aspas para delimitar os valores dos atributos das tags.

Correto

```
<input type="text" name="email" disabled="disabled" />
```

Incorreto

```
<input type=text name=email disabled>
```

Indentação

Como no PHP, a indentação no HTML deve sempre refletir a estrutura lógica e deverá ser feita com tabulações e não espaços. Ao misturar os códigos PHP e HTML, os blocos de indentação PHP deverão respeitar o código HTML de modo que os níveis de abertura e fechamento de um se encaixem ao do outro.

Correto

```
<?php if ( ! have_posts() ) : ?>
    <div id="post-1" class="post">
        <h1 class="entry-title">Not Found</h1>
        <div class="entry-content">
            <p>Apologies, but no results were found.</p>
            <?php get_search_form(); ?>
        </div>
    </div>
<?php endif; ?>
```

Incorreto

```
<?php if ( ! have_posts() ) : ?>
    <div id="post-0" class="post error404 not-found">
        <h1 class="entry-title">Not Found</h1>
        <div class="entry-content">
            <p>Apologies, but no results were found.</p>
<?php get_search_form(); ?>
        </div>
    </div>
<?php endif; ?>
```

PHP

Aspas simples e duplas

Use aspas simples e duplas quando apropriado. Se você não estiver tratando nada na string, use aspas simples. Você nunca deve escapar aspas HTML numa string, porque você apenas precisa alternar entre os tipos de aspas, assim:

Exemplo

```
echo '<a href="link" title="Título">Nome do link</a>';  
echo "<a href='$link' title='$titulodo$link'>$nomedolink</a>";
```

A única exceção é no JavaScript, que as vezes reque aspas simples ou duplas. Textos que venham dentro de atributos devem passar pelo `attribute_escape()` assim as aspas simples ou duplas não fecham o atributo e invalidam o XHTML causando um problema de segurança.

Indentação

Sua indentação deve sempre refletir uma estrutura lógica. Use tabs reais e não espaços, pois isso permite maior flexibilidade entre clientes. Regra de ouro: tabs devem ser usadas no início das linhas e espaços devem ser usados no meio das linhas. Exceção: se você tem um bloco de código que seja mais legível se estiver alinhado, use espaços:

Exemplo

```
$foo    = 'algunvalor';  
$foo2   = 'algunvalor2';  
$foo3   = 'algunvalor3';  
$foo4   = 'algunvalor4';
```

Estilo das Chaves

Chaves devem ser usadas em múltiplos blocos. Se você tiver um bloco muito grande, considere quebrá-lo em dois ou mais blocos ou funções. Caso seja realmente necessária a existência desse longo bloco, por favor ponha um pequeno comentário no final para que as pessoas percebam de relance o que aquela chave de fechamento está fechando.

Exemplo

```
if ( condicao ) {  
    acao1();  
    acao2();  
} elseif ( condicao2 && condicao3 ) {  
    acao3();  
    acao4();  
} else {  
    acaopadrao();  
}
```

Normalmente isso é apropriado para blocos lógicos, maiores que cerca de 35 linhas, mas qualquer código que não seja intuitivamente óbvio pode ser comentado. Blocos de uma linha apenas pode omitir as chaves para ficarem mais concisos:

Exemplo

```
if ( condicao )  
    acao1();  
else  
    acao2();
```

Uso de espaços

Sempre coloque espaços:

Após as vírgulas

```
array( 1, 2, 3 )
```

Em ambos os lados das atribuições de operadores lógicos

```
x == 23
```

Em ambos os lados dos parenteses

```
foreach ( $foo as $bar ) {
```

Quando definindo ou chamando funções, entre os parâmetros

```
function minhafuncao( $param1 = 'foo', $param2 = 'bar' ) {  
    minhafuncao( $param1, outrafuncao( $param2 ) );
```

Variáveis, funções, nomes de arquivos, e operadores

Use letras minúsculas em nomes de variáveis e funções. Separe as palavras por sublinhados (underscores).

Exemplo

```
function algum_nome( $alguma_variavel ) { [...] }
```

Arquivos devem ser nomeados descritivamente usando letras minúsculas. Hífens devem separar as palavras.

Exemplo

```
nome-do-meu-plugin.php
```

Sinalização de parâmetros

Para sinalizar parâmetros para funções prefira valores em string a apenas true e false quando chamar funções; e sempre com nomes auto-explicativos.

Incorreto

```
function comer( $oque, $devagar = true ) {  
    ...  
}  
comer( 'cogumelos' );  
comer( 'cogumelos', true );  
// O que significa true?  
comer( 'comida de cachorro', false );  
// O que significa false? O oposto de true?
```

Já que o PHP não suporta argumentos em forma de nomes, os valores dos sinalizadores não tem significados e toda vez que aparece uma função como essa nós temos que pesquisar pela definição da função. O código pode ficar mais legível se usarmos textos descritivos, ao invés de booleanos:

Correto

```
function comer( $oque, $velocidade = 'devagar' ) {  
    ...  
}  
comer( 'cogumelos' );  
comer( 'cogumelos', 'devagar' );  
comer( 'comida de cachorro', 'rapido' );
```


Internacionalização

A Internacionalização é o processo que deixa uma aplicação preparada para receber traduções. No WordPress significa marcar os textos padrão dos temas que deverão ser traduzidos de um modo especial. A internacionalização também é conhecida como i18n por possuir 18 letras entre o i e o n.

A tradução dos termos é feita com o uso do gettext, biblioteca que no PHP já é permitida por padrão em suas extensões; e o Wordpress faz uso desse artifício.

Arquivos POT

O gettext percorre os arquivos do sistema indicados pelo editor à procura de itens previamente preparados para tradução, aqueles onde existir uma chamada para a biblioteca.

Ao encontrar as entradas de tradução ele extrai as informações gerando seu padrão de tradução dos arquivos POT exemplificado abaixo:

Exemplo

```
#: wp-content/themes/vitrine/page.php:15
msgid "Edit"
msgstr "Editar"
```

Arquivos PO

Para traduzir temas, plugins e o próprio WordPress; cada tradutor especifica nas linhas msgstr a tradução para o idioma pretendido.

O resultado dessa tradução é um arquivo PO no mesmo formato de um arquivo POT, porém com cabeçalhos específicos e obviamente as traduções editadas.

Arquivos MO

Ao gerar um arquivo PO de tradução é criado automaticamente um arquivo MO, do tipo binário, que carrega consigo todas as entradas e traduções num formato adequado para rápida extração das informações traduzidas.

Criando entradas

Para tornar um texto traduzível, adicione ele dentro das funções de tradução. As mais comumente utilizadas são:

Atribuindo o valor da tradução a uma variável

```
$texto = __('My text...');
```

Exibindo o texto diretamente em tela

```
_e('My text...');  
// Esse último recurso é equivalente a:  
echo __('My text...');
```



Informe os textos sem uso de caracteres especiais ou acentuação para evitar erros posteriores. Em nosso projeto utilizarei termos em inglês para desenvolver o tema. Desse modo ao final teremos o tema completo em dois idiomas!

Criando um template web

Com base nos Wireframes criados podemos enfim começar a programar nosso tema. Primeiramente criaremos apenas um Template web. Template é o modelo a ser utilizado, seria o mesmo que Tema, porém este último mais específico tratando dos templates preparados para uso com o WordPress; enquanto que o primeiro possui apenas formatações web sem intervenção de funções de qualquer CMS.

Criaremos um arquivo index.php apenas com marcações em HTML que nos permita, logo em seguida com ajuda do CSS, obter a formatação pretendida. Ao programar não se esqueça dos padrões estudados.



Visualização da index.php em um navegador web

index.php

Vemos a nitida separação do conteúdo em camadas, ao melhor estilo Tableless, cada uma com um identificador único (id) e alguns elementos acompanhados de classes (class) para facilitar suas formatações.

style.css

O estilo explorado traz de modo compacto um reset de início. Essa simples instrução faz com que (de modo superficial) todos os navegadores tenham um ponto em comum na definição de sua folha de estilos.



É imprescindível o bom entendimento de CSS para uma boa formatação dos elementos.

Repare em nosso exemplo como poucas linhas, mesmo desprovidas de elementos gráficos mais robustos, fizeram uma enorme diferença no resultado final obtido.



Os arquivos utilizados nesse capítulo estão disponíveis no arquivo **início-dos-trabalhos.rar** anexo ao livro.



Tomaremos esses arquivos como base para a continuação do desenvolvimento.

Para fins didáticos e de fixação do conteúdo é altamente recomendável que você pegue os arquivos e os altere enquanto executa a leitura do livro.

Parte III

Construção do tema

Os temas WordPress ficam alojados na pasta wp-content/themes/, partindo da pasta de instalação do CMS. Cada tema possui ainda uma pasta dentro da pasta de temas e dentro dela todos os seus arquivos de imagem, estilo, funções, etc.



O WordPress inclui um tema padrão em sua instalação. Examine os arquivos do tema padrão cuidadosamente para ter uma melhor idéia de como criar seus próprios arquivos de tema.

Os temas basicamente consistem de três principais tipos de arquivos, além de imagens e arquivos JavaScript, são eles:

- **style.css**

Controla a apresentação (design e layout) das páginas do site.

- **functions.php**

Funções opcionais

- **Arquivos do tema (.php)**

Controlam a forma como as páginas do site gerenciam as informações do banco de dados para ser exibido no site

Estilo

Além de informações de estilo CSS para o tema, style.css fornece detalhes sobre o tema na forma de comentários. Dois temas não podem ter os mesmos dados em seus cabeçalhos. Caso isso ocorra haverá problemas ao selecionar o tema ou ativá-lo.

Como já temos o arquivo, agora incluiremos como nosso cabeçalho algumas informações sobre o tema a ser criado:

```
style.css
/*
Theme Name: Aprendiz
Version: 1.0
Description: Desenvolvido por <a
href="http://www.guiawp.com.br">Guia WordPress</a>.
Author: Guilherme Mazetto
Author URI: http://www.guiawp.com.br/
*/
```

O cabeçalho de linhas de comentário no style.css são necessários para o WordPress para ser capaz de identificar um tema e apresentá-lo no Painel Administrativo acessando Aparência > Temas. Nessa página estarão disponíveis todos os temas instalados.

Para ser melhor visualizado na escolha de temas, abra seu template e capture uma tela dele. Crie um arquivo de screenshot.png com as dimensões 300px de largura por 225px de altura e salve dentro da pasta do seu tema. Feito isso, essa imagem servirá como miniatura do seu tema e facilitará a escolha dele quando necessário.

Template Tags

As Template tags são códigos que instruem o WordPress para fazer determinadas ações ou pegar alguma informação. Para que a integração de um simples template web seja feita com o WordPress é necessário a utilização dessas tags. Elas vão ao banco de dados do blog e recuperam as informações personalizadas de cada página, categoria, autor e o que mais for requisitado referente ao blog em uso.

Muitas vezes nos é permitido enviar parâmetros para as tags com intuito de personalizar ainda mais a resposta que elas trarão. De forma geral, são consideradas Template Tags, todas as funções que usaremos em nosso tema para exibir nele as informações provenientes do WordPress; por exemplo: `bloginfo`, `wp_nav_menu`, `the_search_query`, `wp_get_archives`.

Cabeçalho

Trabalhando com os arquivo do tema, em nosso caso o `index.php`, começaremos a estudar suas partes. No topo da página, separamos o espaço para nosso cabeçalho. Compreendemos por cabeçalho a parte superior do site que vai do início até a exibição do menu.

O cabeçalho traz informações importantes para o servidor web e buscadores e também, em nosso caso, em sua parte visível exibirá o nome e a descrição do blog de acordo com os valores informados pelo WordPress. Para isso acontecer devemos começar a integrar nosso arquivo com as funções do WordPress.

Antes mesmo da tag `<head>` incluiremos a função que determina o

idioma a ser utilizado:

index.php

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

A função `language_attributes` exibe os atributos de linguagem para a tag `<html>`; identificando o idioma em uso e também a direção do texto para a página. Ainda dentro da tag `<head>`, adicionaremos a seguinte linha:

index.php

```
<link rel="profile" href="http://gmpg.org/xfn/11" />
```

Ela define o perfil de relacionamento que será respeitado, de acordo com as normas do endereço citado. O atributo `'rel'` descreve a relação do atual documento com uma âncora especificada pelo atributo `href`. O valor desse atributo é uma lista de tipos de links separados por um espaço simples.

O WordPress adota o padrão XFN (XHTML Friends Network) Rede de amigos XHTML, que visa representar nos links as relações humanas. Geralmente o atributo é preenchido automaticamente pelo sistema.



Para saber mais sobre o padrão XFN acesse <http://gmpg.org/xfn/>

Continuando a incluir informações relevantes dentro da tag `<head>`, nosso cabeçalho para servidores, adicionaremos as meta tags:

index.php

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

Vemos o uso da função `bloginfo`. Ela mostra informações sobre o blog, em sua maioria as que são definidas em Opções Gerais do Painel Administrativo do WordPress (Configurações > Geral).



A função `bloginfo` sempre imprime um resultado para o navegador. Caso precise do valor para uso no PHP, use `get_bloginfo`.

Os parâmetros usados foram:

- **html_type** – tipo do html em uso
- **charset** – codificação do texto
- **language** – idioma em uso

Utilizamos também a função `wp_title` que, no caso, exibirá o título da página com um separador de texto localizado a direita do título.

Utilizando ainda o `bloginfo`, substituiremos o endereço do arquivo de estilo, nome, descrição e links para página inicial do blog fechando temporariamente nosso cabeçalho:

index.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" <?php
language_attributes(); ?>>

<head>
  <link rel="profile" href="http://gmpg.org/xfn/11" />
  <meta http-equiv="content-type" content="<?php
bloginfo('html_type') ?>; charset=<?php bloginfo('charset') ?>" />
  <meta http-equiv="content-language" content="<?php
bloginfo('language'); ?>" />
  <title><?php wp_title( ' | ', true, 'right' );
bloginfo( 'name' ); ?></title>
  <link rel="stylesheet" type="text/css" media="all" href="<?php
bloginfo('stylesheet_url'); ?>" />
</head>
<body>

<div id="page-wrap">
  <div id="header">
    <p><a href="<?php bloginfo('url'); ?>" title="<?php
bloginfo('name'); ?>"><?php bloginfo('name'); ?></a></p>
    <span><a href="<?php bloginfo('url'); ?>" title="<?php
bloginfo('description'); ?>"><?php bloginfo('description'); ?
></a></span>
  </div>
```

Dados não confiáveis

Note que imprimimos no atributo `title` dos dois links o nome e descrição do blog respectivamente. O conteúdo dessas variáveis será discriminado por cada autor de blog. Supondo que um autor qualquer defina como nome ou descrição um valor contendo aspas poderia deixar o documento inválido para as normas do W3C ou até mesmo significar um desastre para o tema.



Devemos ficar atento quando imprimimos variáveis do WordPress como atributos de elementos HTML, geralmente os erros ocorrem nos atributos `alt`, `title` e `value` das tags

Nesses casos utilizaremos o `esc_attr`. A função codifica (apenas uma vez) os símbolos `<` `>` `&` `"` `'` (sinal de menor, sinal de maior, 'e' comercial, aspas duplas, aspas simples).

index.php

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

Arquivo de Funções

Um tema pode opcionalmente usar um arquivo de funções, que reside no subdiretório do tema e é chamado `functions.php`. Este arquivo basicamente funciona como um plugin, e se ele está presente no tema que você está usando, ele é carregado automaticamente durante a inicialização do WordPress (tanto para páginas do blog e páginas de administração).

Veja algumas das principais utilizações deste arquivo:

- Definir funções que serão utilizadas em diversos arquivos diferentes do seu tema;
- Habilitar recursos como os menus de navegação;
- Criar um menu de opções para o tema, dando opções para alterar cores, estilos e outros aspectos;

Menus

O WordPress incluiu um mecanismo fácil para uso de menus de navegação personalizados em um tema. O recurso trata-se, como o próprio nome já diz, de se responsabilizar pela criação e edição de menus para as versões mais recentes do WordPress.

Para incorporar o suporte a menus em um tema basta incluir algumas poucas linhas de código no arquivo `functions.php`:

```
functions.php  
register_nav_menu( 'main-menu', __( 'Main Menu' ) );
```

E no arquivo index.php substituiremos a lista de links por:

index.php

```
<div id="menu">
  <?php wp_nav_menu( array( 'location'=>'main-menu',
  'container'=>null ) ); ?>
</div>
```

O valor passado para location deverá ser o mesmo que o especificado anteriormente ao registrar o menu. Desse modo você está dizendo que aquele menu, deverá aparecer aqui. Container null, pois por padrão, adicionaria outra div, como já temos uma não será preciso outra.

Não tiramos a div e a incorporamos através da função, pois, se os menus não estiverem ativos; a formatação dos links ficará prejudicada. Após implantar esse recurso, acesse seu painel de administração do WordPress. Clique em Aparência > Menus e gerencie seu menu como bem entender. Construído o menu, ele aparecerá no local indicado.

Sidebar

A barra lateral (Sidebar) é uma coluna vertical pouco mais estreita que o espaço para o conteúdo, e geralmente recheado de muita informação sobre o blog. Encontrado na maioria dos blogs WordPress, a barra lateral é geralmente colocada ao lado direito ou esquerdo do conteúdo, embora em alguns casos, podemos ver duas ou mais barras laterais em diferentes posições no layout.

Navegação

Atualmente o objetivo principal da barra lateral é de prestar assistência de navegação para o visitante. Para isso são projetadas listas de itens para ajudar os visitantes do seu blog a encontrar determinadas informações, conteúdo.

Tais listas de navegação inclui Categorias, Páginas, Arquivos, etc. Outra ferramenta de navegação que você verá na barra lateral é um formulário de busca que também servirá para ajudar as pessoas a encontrarem o que estão procurando no seu site.

Listas aninhadas

O clássico de temas WordPress é usar listas aninhadas para apresentar informações da barra lateral. Listas aninhadas são uma série de listas não-ordenadas, situadas uma dentro de outro.

A utilização das listas aninhadas em sua barra lateral não é obrigatório, no entanto o seu entendimento é de grande importância dado

que, a maioria dos plugins e informações desenvolvidas para serem publicadas na Sidebar, fazem uso desse artifício.

Widgets

Widgets são como plugins, mas desenvolvidos para fornecer um simples modo de agrupar vários elementos no conteúdo da Sidebar sem ter que alterar nenhuma linha de código para aplicá-los.

Através da aba Aparência > Widgets podemos arrastá-los e trocá-los facilmente de posição no menu lateral, e como resultado customizar ainda mais o nosso tema. Para efetivar qualquer edição através da aba de um widget é preciso clicar sobre o respectivo botão salvar, caso contrário ela não será aplicada.

Personalizando a Sidebar

Como base em todos esses conhecimentos, vamos agora atualizar o conteúdo de nossa Sidebar. Dentro do formulário de busca informe como valor do atributo action o endereço inicial do blog e o valor do campo de pesquisa:

index.php

```
<form method="get" id="searchform" action="<?php
bloginfo('url'); ?>">
...
<input type="text" value="<?php the_search_query(); ?>" name="s"
id="s" />
```

Com as alterações, sempre que submetido o formulário redirecionará para a página inicial do blog e enquanto houver uma pesquisa ativa o valor será exibido na caixa de texto para pesquisa. O próximo passo é preparar a barra lateral para receber os Widgets.

functions.php

```
register_sidebar( array(
    'name'           => __( 'My Sidebar' ),
    'id'             => 'my-sidebar',
    'before_widget'  => '<ul><li>',
    'after_widget'   => '</li></ul>',
    'before_title'   => '<h3>',
    'after_title'    => '</h3>' )
);
```

Com ela registraremos uma área para Widgets com os atributos descritos. Voltando ao index.php, logo abaixo do formulário de busca substitua a lista criada por:

index.php

```
<?php if ( !dynamic_sidebar( 'my-sidebar' ) ) : ?>
<ul>
  <li><h3><?php _e( 'Archive' ); ?></h3></li>
  <li>
    <ul>
      <?php wp_get_archives(); ?>
    </ul>
  </li>
</ul>
<?php endif; ?>
```

A função `dynamic_sidebar` irá imprimir no referido espaço o conteúdo que será gerenciado pelos Widgets e retornará verdadeiro ou falso de acordo com a presença ou não deles. Fazemos a verificação e exibimos uma lista com o Arquivo do blog separado por mês de modo a não deixar o espaço vazio caso o usuário não tenha incluído nenhum widget pelo painel.

Rodapé

As opções para trabalhar com o rodapé são várias. É possível verificarmos a presença de listas de links, notícias, textos e muita informação nesse espaço. Uma prática bastante usada em WordPress é realizar a preparação do rodapé como Sidebars auxiliares e registrá-las tal como fora mostrado no item anterior permitindo a inserção de Widgets.

Mantendo o clássico padrão de exibir apenas os créditos no rodapé, anteriormente estipulamos o conteúdo que nosso rodapé teria. Agora vamos adaptá-lo ao WordPress possibilitando a tradução do texto informado.

index.php

```
<?php
printf( __( '%s Theme by %s.' ),
    '<a href="http://www.wordpress.org">wordpress</a>',
    '<a href="http://www.guiawp.com.br">Guia wordPress</a>'
);
?>
```

A função printf é do PHP, seu uso é altamente recomendado em circunstâncias como a demonstrada por facilitar a formatação dos textos a serem exibidos. Em nosso caso, o primeiro identificador %s exibirá o link para o site do WordPress, e o segundo, o link para o site do desenvolvedor.

Trabalhando o conteúdo

A partir desse ponto começaremos a integrar o conteúdo do nosso (antigo) template com as funções do WordPress. O conteúdo do blog basicamente será composto pelos posts, caracterizados pelas suas informações como título, data, autor, conteúdo e classificação em tags e categorias.

Para conseguir o resultado esperado será necessário entender bem o funcionamento do Loop do WordPress. O Loop nada mais é que o laço de repetição responsável por exibir os posts do blog.

Loop

O Loop é usado pelo WordPress para exibir cada uma de suas postagens. Usando o Loop, o WordPress executa os processos de cada uma das mensagens a serem exibidas na página atual e as formata de acordo com os critérios especificados. Qualquer código HTML ou PHP colocado no laço será repetido em cada post.

Exemplo

```
<?php if (have_posts ()): while (have_posts ()):? the_post ();?>  
  Conteúdo a ser executado a cada iteração...<br />  
  Seja ele em HTML ou <?php echo 'PHP'; ?>  
<?php endwhile; ?>  
<?php else: ?>  
  <p><?php _e ('Sorry, not found!');?></p>  
<?php endif;?>
```

Acabamos de dizer ao nosso tema para verificar a presença de conteúdo. Se existem conteúdos a serem exibidos, enquanto houver; mostre-me os posts um a um até que não haja mais. Se não existir conteúdo mostraremos uma mensagem de erro na tela informando a ausência do conteúdo.

The e Get_the

Uma das grandes vantagens das funções do WordPress é justamente a capacidade de entendimento que ela nos proporciona. Qualquer pessoa, com um nível básico de inglês, consegue entender o funcionamento que está implícito a cada função. Por exemplo, a exibição dos títulos dos posts com o respectivo link para ele:

index.php

```
<h1><a href="<?php the_permalink(); ?>" title="<?php esc_attr(
the_title() ); ?>"><?php the_title(); ?></a></h1>
```

A função `the_permalink` exibe o link permanente (Permalink) para o post, enquanto a `the_title` exibe o título do post; respectivamente nos locais onde estão especificadas.

No entanto algumas vezes não queremos exibir diretamente as informações, mas sim transmiti-las ao PHP para tratamento e somente depois enviar para impressão em tela. Nesses casos, a maioria das função `the_` possuem um equivalente `get_the_`, como a utilizada para exibir o nome do autor do post, destacada abaixo:

index.php

```
<?php printf( esc_attr__( 'More posts from %s' ), get_the_author()  
); ?>
```

As funções `the` e `get_the` possuem basicamente o mesmo funcionamento. As funções `the_` deverão estar dentro do loop. As funções `get_the_` variam, algumas devem estar dentro do loop e outras recebem identificadores de post como parâmetro.

Plugin API Hooks

Basicamente os hooks são ações onde poderemos chamar outras ações, os conhecidos callbacks. Dessa maneira sempre que determinada ação for executada no Wordpress, uma função definida por você poderá ser chamada. Esses 'ganchos' (Hooks) nos permitem personalizar ainda mais os temas com instalação de plugins e material de terceiros.

O WordPress ao deparar-se com algum desses ganchos, interrompe seu processamento e verifica se existe alguma função para ele para ele fazer alguma coisa. Caso afirmativo, a função é executada (podendo ser mais de uma) e o sistema continua seu processamento normal.

Existem dois tipos de Hooks: Action e Filter (Ação e Filtro)

Actions

Chama a função em determinado ponto. Usando o gancho 'admin_footer', sempre que o rodapé do painel administrativo for executado, a função para ele passada também será.

Filters

Passa argumentos, conteúdo para a referida função de gancho. Desse modo a função pode usar a informação transmitida para realizar sua tarefa. Com isso podemos (por exemplo) passar uma função ao filtro the_content que coloque tags de negrito para o termo WordPress, sem alterar de fato a funcionalidade de exibir o conteúdo.

Introduziremos alguns Hooks específicos dentro do nosso tema:

wp_head

Acompanha o elemento <head> de um tema no arquivo header.php.

wp_footer

Aparece no footer.php, logo antes do fechamento da tag </body>.

Visualmente não terão nenhum efeito sobre o tema, porém a ausência desses poderá implicar em não funcionamento e sérios problemas com plugins e outras funções que poderão ser implementadas futuramente.



Os arquivos utilizados nesse capítulo estão disponíveis no arquivo **construcao-do-tema.rar** anexo ao livro.

Parte IV

Estrutura do tema

Vimos até aqui como o `index.php` é flexível e pode ser usado incluindo todas as referências de cabeçalho, sidebar, rodapé, conteúdo, categorias, arquivo, busca, erro e qualquer outra página criada no WordPress.

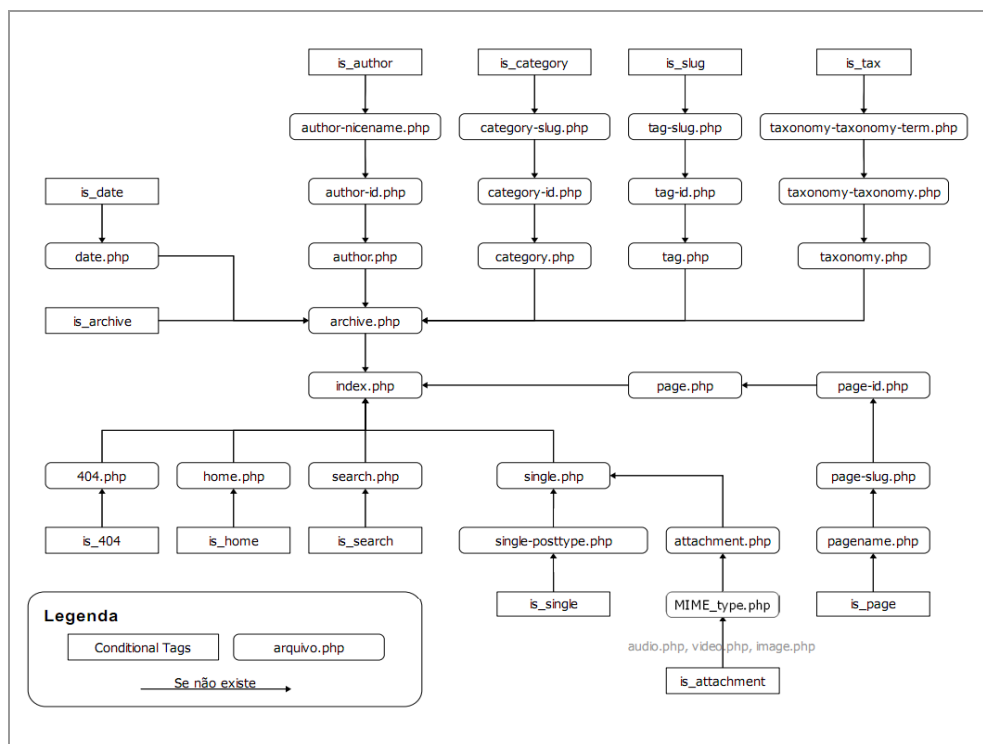
Por outro lado ele pode ser dividido em módulos, arquivos menores e mais específicos em sua função. Cada um desses arquivos fazendo apenas parte do tema total. Caso algum desses arquivos não esteja presente entre os demais arquivos do tema, o WordPress utiliza um arquivo padrão respeitando as normas de sua Hierarquia.

Hierarquia

Os arquivos `.php` do tema (separadamente conhecidos como templates ou modelos) se encaixam como peças de um quebra-cabeça para gerar as páginas em seu blog. Alguns são usados em todas as páginas da web (cabeçalho, rodapé e estilos), enquanto outros são usados somente em condições específicas (páginas personalizadas).

O WordPress utiliza informações contidas no interior de cada link em seu blog para decidir qual o modelo ou conjunto deles serão utilizados para exibir a página. Tendo como ponto de partida um verificador para todos os tipos de consulta que decide qual tipo de página está sendo solicitado, os modelos são então escolhidos (dependendo da disponibilidade) na ordem sugerida pela Hierarquia do WordPress.

Para ilustrar o funcionamento da hierarquia, supomos que o seu blog está instalado em `http://meublog.com/` e um visitante clica em um link que o direcionará para uma página de categoria, sendo essa chamada 'Minha categoria'; algo como `http://meublog.com/categoria/minha-categoria/`.



Hierarquia de modelos do WordPress

Após a requisição o WordPress irá procurar por um modelo na pasta do tema atual que corresponde ao slug da categoria, `category-minha-categoria.php`. Caso não encontre, o próximo passo é procurar pelo identificador (ID) da categoria. Se a categoria é de identificação 1, o WordPress procura por um modelo `category-1.php`. Se esse não existir, o WordPress procura pelo modelo genérico, `category.php`, seguido do `archive.php` e finalmente no `index.php`.

O `index.php` portanto é exibido sempre que os demais modelos não forem encontrados por isso é de suma importância e obrigatório em todos os temas WordPress.

Arquivos Modelos

Com base no que acabamos de ver, trabalharemos amplamente com modelos. Para facilitar nosso trabalho, o arquivo `index.php` será dividido em 4 partes (As reticências servem apenas para resumir o conteúdo dos arquivos. Dessa forma ao executar a separação dos arquivos, todos os códigos originais devem ser mantidos):

header.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" <?php
language_attributes(); ?>>
...
<div id="menu">
    <?php wp_nav_menu( array( 'location'=>'main-menu',
'container'=>null ) ); ?>
</div>
```

sidebar.php

```
<div id="sidebar">
...
</div><!-- #sidebar -->
```

Ainda dentro do Sidebar extrairemos nosso formulário de busca, deixando-o em um arquivo separado de nome `searchform.php`. Assim caso você queira utilizar o formulário em outro local também poderá.

searchform.php

```
<form method="get" id="searchform" action="<?php  
bloginfo('url'); ?>">  
<div>  
  <input type="text" value="<?php the_search_query(); ?>" name="s"  
id="s" />  
  <input type="submit" id="searchsubmit" value="Buscar" />  
</div>  
</form>
```

footer.php

```
</div><!-- #container -->  
...  
</html>
```

index.php

```
<div id="content">  
...  
</div><!-- #content -->
```



Repare como a marcação do conteúdo HTML com comentários do tipo `<!-- #nome-do-bloco-de-fechamento -->` auxilia e muito na distribuição dos códigos.

Includes Tags

As include tags são usadas nos modelos para executar HTML e PHP que se encontram em outros arquivos modelo. O PHP possui as instruções include e require para esse fim, no entanto para arquivos específicos do tema é aconselhável o uso das Includes Tags em razão da facilidade e padronização do código.

Em nosso arquivo index.php incluiremos as chamadas get_header, get_sidebar e get_footer:

```
index.php
<?php get_header(); ?>
<div id="content">
...
</div><!-- #content -->
<?php get_sidebar(); get_footer(); ?>
```

E no arquivo sidebar.php incluiremos get_search_form:

```
sidebar.php
<div id="sidebar">
<ul><li><?php get_search_form(); ?></li></ul>
...
```

Ao abrirmos novamente o nosso blog, constataremos que os demais arquivos foram chamados nos respectivos espaços. Caso algum modelo não seja encontrado, em seu lugar será incluído o arquivo homônimo da pasta wp-includes/theme-compat e quando esse não existir, no caso do formulário de busca, será automaticamente gerado.

Incluindo arquivos

Agora que sabemos o quão fácil pode ser nosso trabalho reaproveitando 'pedaços' do nosso tema, vamos estender essa prática para qualquer arquivo PHP. Para isso devemos ter um arquivo com as instruções desejadas denominado slug.php junto com os demais arquivos do tema. Para incluir as referidas instruções em um outro arquivo basta informar:

Exemplo

```
<?php get_template_part( $slug ); ?>
```



Slug é uma expressão do meio jornalístico para criar identificações mais claras e intuitivas para conteúdo publicado na web.

Aplicam-se no uso dos Permalinks com suas URL amigáveis aos visitantes do blog. Uma identificação slug não possui acentos, caracteres especiais ou espaços em branco.

Criaremos para esse passo um arquivo dizendo que a informação procurada não existe, ou seja, o usuário fez uma consulta que não teve resultados e precisamos informá-lo disso. Assim sendo edite:

index.php

```
// substitua  
<p><?php _e('Sorry, the content not found!');?></p>  
// por  
<?php get_template_part( 'no-results' ); ?>
```

Criaremos um arquivo chamado no-results.php e nele colocaremos a informação que desejamos exibir, copie a linha substituída dentro desse arquivo.

Da mesma forma faremos com o conteúdo dos posts. Criaremos um arquivo chamado `loop.php` e nele colocaremos todas as instruções do Loop que estavam no `index.php`. Por outro lado, nesse último arquivo, substituiremos as instruções por sua chamada. Veja como ficou o `index.php` final:

index.php

```
<?php get_header(); ?>
<div id="content">
    <?php get_template_part('loop'); ?>
</div><!-- #content -->
<?php get_sidebar(); get_footer(); ?>
```

Single.php

Iremos agora preparar o arquivo que receberá os posts quando visualizados separadamente e também poderá trabalhar com os comentários, item de muitíssima importância para qualquer blog.

Copie o arquivo loop.php e renomeie para single.php e inclua os arquivos de cabeçalho, sidebar e rodapé nos respectivos lugares. Não se esqueça da div de conteúdo que foi deixada no index.php.

Dentro da div entry-meta, logo após exibir o nome do autor, incluiremos um link que nos levará até os comentários utilizando a função `comments_popup_link`.

single.php

```
<?php comments_popup_link( __( 'Leave a comment' ), __( '1  
Comment' ), __( '% Comments' ), 'comments', __( 'Comments are  
close.' ) ); ?>
```

Por fim ao final da div com classe post, incluiremos a função `comments_template`.

single.php

```
</div><!-- .post -->  
<?php comments_template(); ?>  
<?php endwhile; ?>
```

Pronto, seu sistema de comentários está ativo e pronto para ser usado, no entanto algumas alterações no CSS serão necessárias para deixar a listagem e o formulário mais bem apresentáveis.

style.css

```
.commentlist { margin: 10px 0 30px 0; }
.commentlist li { list-style: Nenhum; }
.commentlist li ul { margin-left: 20px; }
.commentlist li .comment-body { background: #eee; padding: 10px;
margin-bottom: 10px; }
.commentlist img { float: left; margin: 5px; }
.commentlist p { clear: both; margin: 15px 0; color: #555; }
.reply { text-align: right; }
#commentform { margin-top: 15px; }
#commentform p { margin-bottom: 7px; }
```

2 respostas para “Olá, mundo!”



Sr. WordPress disse:

30 de outubro de 2010 às 14:41

Olá, isto é um comentário.

Para excluir um comentário, faça o login e veja os comentários dos posts.

Responder



mazetto disse:

1 de novembro de 2010 às 17:29

Resposta ao comentário

Responder

Listagem dos comentários do tema Aprendiz



Os arquivos utilizados nesse capítulo estão disponíveis no arquivo **estrutura-do-tema.rar** anexo ao livro.

Parte V

Aprimoramentos

Sabemos que nosso tema suporta exibir mais de um post na index. O número de posts a serem exibidos por página é configurável pelo painel em Configurações > Leitura. No entanto quando a quantidade de posts existentes no blog ultrapassar esse número em nosso tema atual, não teríamos condições de ver os posts mais antigos.

Para resolver a situação criaremos uma paginação para que nosso tema não deixe de exibir nenhum post, por mais antigo que ele seja. No arquivo loop.php após o final do Loop incluiremos:

loop.php

```
<div id="navigation">
    <?php previous_posts_link( __( 'Newer posts' ) ); ?>
    <?php next_posts_link( __( 'Older posts' ) ); ?>
</div>
```

Para melhor exibir em tela, no style.css também incluiremos:

style.css

```
/* Navegação */
#navigation a { background: #c22; color: #eee; padding: 4px;}
#navigation a:hover { background: #fc7; color: #333;}
```

Resumo

No entanto muitas vezes ao exibir a listagem dos posts o visitante não clica para ver o post separadamente pois já o leu por completo. Uma excelente prática muito utilizada é a de exibir apenas os resumos nas paginações e apenas no single.php exibir o restante do post.

Para isso no loop.php ao inves de informar the_content, responsável por exibir o conteúdo, informe the_excerpt.

loop.php

```
<div class="entry">
  <?php the_excerpt(); ?>
</div>
```

Nessas circunstâncias, caso o campo resumo foi preenchido durante a edição do post ele será exibido, senão um número limitado de palavras do texto original serão expostos seguido de um continuador de texto [...]

Search

Ainda tratando de paginação de posts, criaremos agora o arquivo search.php que será exibido quando uma busca for executada. Copie o arquivo index.php e renomeie para search.php

search.php

```
<div id="content">
  <h1 id="archive"><?php printf( __( 'Search results for: <span>
%s</span>' ), get_search_query() ); ?></h1>
  <?php get_template_part('loop'); ?>
</div><!-- #content -->
```

Inclua as seguintes formatações no style.css

Style.css

```
/* Arquivo */
#archive { margin-bottom: 25px; border: 0;}
#archive span { color: #c22; text-decoration: underline;}
```

Conditional Tags

As tags condicionais podem ser usadas em seus modelos para mudar o conteúdo e como ele é exibido em uma certa página dependendo em quais condições a página se encontra. Isto é, fazemos a verificação se a página acessada é determinada página, e de acordo com a resposta obtida realizaremos ou não certas instruções.

As tags verificam se uma determinada condição é satisfeita, e em seguida, retorna verdadeiro ou falso. Algumas das tags condicionais geralmente usadas são: `is_single`, `is_home` e `is_page` que verificam se a página atual é a de posts, inicial ou uma página respectivamente.

Archive

Apurando os dados da mesma maneira, trataremos das buscas quando efetuadas por tag, categoria, autor ou data. Para isso, copie o arquivo recém modificado search.php e renomeie para archive.php. No local de exibição do título, faça:

archive.php

```
<div id="content">
  <h1 id="archive">
    <?php
      if ( is_day() )
        printf( __( 'Daily archives: <span>%s</span>' ) ,
get_the_date() );
      else if ( is_month() )
        printf( __( 'Monthly archives: <span>%s</span>' ) ,
get_the_date( 'F Y' ) );
      else if ( is_year() )
        printf( __( 'Yearly archives: <span>%s</span>' ) ,
get_the_date( 'Y' ) );
      else if ( is_tag() )
        printf( __( 'Tag: <span>%s</span>' ) , single_tag_title( '',
false ) );
      else if ( is_category() )
        printf( __( 'Category: <span>%s</span>' ) ,
single_cat_title( '', false ) );
      else if ( is_author() ){
        $author = get_userdata( $_GET['author'] );
```



```

    printf( __( 'Author: <span>%s</span>' ) , $author->display_name
);
}
else
    _e( 'Blog archives' );
?>
</h1>
<?php get_template_part('loop'); ?>
</div><!-- #content -->

```



Sempre criaremos arquivos para o tema respeitando a hierarquia das páginas. De nada adiantaria ter o arquivo acima se existissem também os arquivos `date.php`, `tag.php`, `category.php` e `author.php` por exemplo.

Posts e páginas

Posts são como notícias que você escreve com uma certa periodicidade e são mostrados no blog em ordem cronológica reversa, ou seja, os mais novos antes dos mais antigos. As páginas por sua vez contêm uma informação estática sem relevância com a data na qual foi criada, mas sim pelo seu conteúdo em si que geralmente trata questões sobre o site ou seus mantenedores.

Devido a essa clara discrepância de dinâmica e relevância baseada na data de sua publicação atribuída a páginas e posts, apenas esses últimos são incorporados no feed. Em uma explicação simbólica seria como se o blog se mantivesse dentro de uma linha do tempo determinada pelos posts e as páginas coexistiriam do lado de fora dessa linha.

Para fins de organização do conteúdo as páginas podem possuir uma hierarquia entre si quando marcado suas respectivas páginas pai (ou mãe) e ordem na caixa 'Atributos de página' da tela de edição. Por outro lado os posts possuem um sistema mais complexo utilizando-se de tags e categorias.

Modelos de Páginas

O WordPress pode ser configurado para usar diferentes modelos de página para páginas diferentes. Para alternar o modelo de página, enquanto edita uma página qualquer pelo painel de administração, selecione o modelo na lista chamada Modelo. O arquivo padrão que exibirá o conteúdo das páginas é a `page.php`.



Para acessar o seletor de modelo, deverá existir pelo menos um modelo de página personalizado disponível no tema ativo.

Criando um novo modelo de página

Os arquivos de modelo da página deverão estar na pasta do seu tema. Você poderá criar um arquivo com qualquer nome, exceto os nomes reservados do WordPress como no caso do header, footer, sidebar e afins. Para criar um modelo basta inserir no cabeçalho da página a instrução:

Exemplo

```
/* Template Name: Nome do modelo */
```

As etapas a seguir serão feitas para criarmos um modelo de exibição das nossas páginas:

- Copie a `single.php` e renomeie para `page.php`
- Apague a div de classe `post-utility`, pois páginas não tem categoria nem tags.

Páginas adicionais

Criaremos agora dois modelos de página; um sem os comentários e outro sem a Sidebar.



Note que assim como no item anterior, os passos de execução dos tópicos a seguir serão dados como itens de uma lista. Essa prática foi tomada para evidenciar ainda mais a facilidade que encontramos nessa fase do desenvolvimento, tendo bem estruturado e separado as partes do nosso tema.

Sem comentários

- Copie o arquivo page.php e renomeie para page-noreply.php;
- Para deixar a página sem comentários apague a linha com a instrução `<?php comments_template(); ?>`.

Sem Sidebar

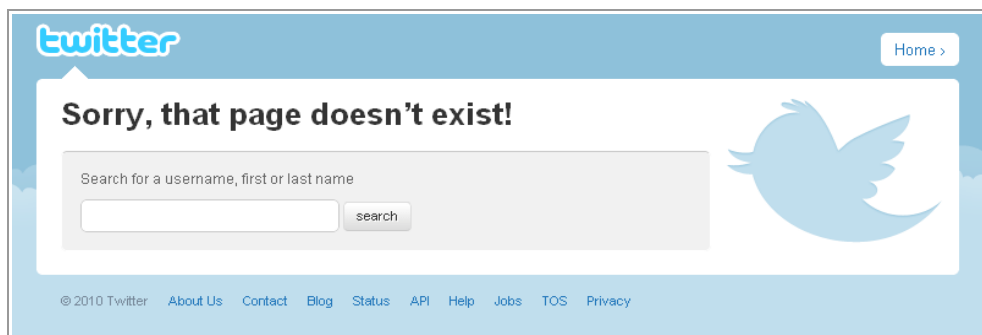
- Copie o recém criado arquivo page-noreply.php e o renomeie para page-nosidebar.php;
- Para deixar a página sem Sidebar apague o `get_sidebar();`
- Na div do conteúdo (`id="content"`) troque o identificador para `content-full`;
- Edite o CSS.

style.css

```
#content-full { width: 900px; padding: 20px; background: #fff;}
```

Página 404

Por vezes é comum acontecer de você apagar alguns posts ou o visitante digita um endereço inválido para teu blog e uma página de erro aparece. Na atual situação do nosso tema, a mensagem que é exibida é a contida no arquivo no-results.php.



Página de erro do Twitter

No entanto, tal arquivo e mensagem referem-se a ausência de conteúdo e não a inexistência dele, isto é, o endereço existe, porém não tem informações a serem exibidas; enquanto que para erros do tipo 404 o endereço é inválido, não existe.

- Copie o arquivo index.php
- Renomeie para 404.php
- Apague a instrução `<?php get_template_part('loop'); ?>` e no local informe sua mensagem de erro:

404.php

```
<h1 id="archive"><?php _e( 'Error 404! Invalid URL...' ); ?></h1>
```

Classes do Tema

Implementando as seguintes funções de classes para HTML, o WordPress gera automaticamente atributos de classe para body, post e elementos dos comentários. Para classes de posts, funciona apenas dentro do Loop.



Utilizaremos apenas o body e post já que os comentários estamos pegando pronto do WordPress e não necessitará de alterações.

header.php

```
<body <?php body_class(); ?>>
```

loop.php, single.php, page.php, ...

```
<div <?php post_class(); ?>>
```

Dessa forma você poderá customizar o CSS como bem entender e melhor aproveitar, por exemplo, para determinar títulos de uma categoria na cor verde basta incluir na folha de estilo:

Exemplo

```
.category-slug-da-categoria h1 { color: #0c0;}
```

JavaScript

Uma forma segura de acrescentar códigos javascripts a um tema WordPress é registrando-os inicialmente via página de funções do tema, e posteriormente incluindo as chamadas para os registros onde necessário, em geral no header.php.

Esse processo apenas inclui o script se ele já não tenha sido incluído, e também é capaz de carregar scripts embutidos do WordPress, como faremos com o jQuery no header.php:

header.php

```
wp_enqueue_script("jquery");
```

Repare que o script foi adicionado com sucesso por fazer parte da biblioteca de scripts do WordPress como mencionado. Além desse, incluiremos também um script responsável por permitir enviar uma resposta aos comentários sem a necessidade de recarregar a página. Faremos isso da seguinte forma logo abaixo da inserção da jQuery:

header.php

```
if ( is_singular() && get_option( 'thread_comments' ) )
    wp_enqueue_script( 'comment-reply' );
```



Veja a lista completa de scripts que o WordPress possui acessando a pasta wp-includes/js

Registrando scripts

Muitas vezes precisamos incorporar códigos javascript ao tema criado para termos uma maior interação com o visitante e os elementos HTML do layout. Vamos incluir um script utilizando o modo seguro do WordPress. Para isso faça o registro dentro do arquivo de funções do tema:

```
functions.php
wp_register_script( 'validator',
get_bloginfo( 'template_directory' ) . '/js/validator.js',
array( 'jquery' ), '1.0' );
```

Note que especificamos como terceiro parâmetro, a obrigatoriedade do script jQuery ser carregado antes do nosso. Isso porquê iremos fazer uso da biblioteca e se ea não estiver presente, nosso código não funcionará.

Validação do formulário

O script proposto irá fazer a validação do formulário de comentários, antes dele ser enviado. Perceba que foi criado uma pasta dentro da pasta do tema com o nome js, e dentro dela incluímos o arquivo validator.js



O objetivo do tópico é ilustrar o modo de inserir scripts em seu tema, por isso o código do arquivo validator.js não será publicado.

Para obtê-lo abra a pasta aprendiz/js do arquivo anexo a este capítulo.

O script verificador é muito interessante de ser utilizado pois reduz ainda mais o número de spams nos seus comentários. Falta agora incluir o

script ao tema. O melhor local para tal tarefa é dentro do arquivo do cabeçalho. Como a verificação somente será usada se o formulário de comentários estiver sendo exibido e houver a possibilidade de resposta; fizemos uma pequena verificação para saber se a página requisitada atende aos requisitos.

header.php

```
if ( is_singular() && get_option( 'thread_comments' ) ){
    wp_enqueue_script( 'comment-reply' );
    wp_enqueue_script( 'validator' );
}
```

Para tornar a exibição do erro mais chamativa iremos incluir algumas informações em nossa folha de estilos.

style.css

```
.error { font-weight: bold; color: #c22;}
```

Folha de estilos

O resultado final da formatação de um tema WordPress depende muito das folhas de estilo CSS. Com o uso delas você tem em suas mãos muitas opções de configurar um layout da maneira mais conveniente. Mesmo criando tudo do zero é preciso fazer uso, ou então padronizar, algumas formatações que são inerentes ao WordPress e seu funcionamento.

Classes do WordPress

O WordPress inclui automaticamente várias classes para as imagens e os elementos de bloco visando padronizar funções básicas, como por exemplo o alinhamento de imagens dentro dos posts através das classes `alignleft`, `aligncenter`, `alignright` e `alignnone`.

No caso das imagens, além da questão do alinhamento, elas ainda necessitam de formatação para suas respectivas legendas. Verifique as classes para tal formatação:

```

style.css
.aligncenter, div.aligncenter { display: block; margin: 0 auto;}
.alignleft { float: left;}
.alignright { float: right;}
/* As instruções abaixo são para formatar as legendas */
.wp-caption {}
.wp-caption img {}
.wp-caption p.wp-caption-text {}
    
```

Registrando estilos

Da mesma forma que registramos e utilizamos nossos scripts, podemos também incluir novos arquivos CSS ao nosso tema, além do style.css. Geralmente iremos adotar essa prática quando o arquivo de estilo principal estiver muito extenso e precisemos dividi-los em partes menores para melhorar o entendimento do mesmo. Ou então quando desejamos criar estilos adicionais ao já existente.

functions.php

```
wp_register_style( 'custom', get_bloginfo( 'template_directory' ) .
'/css/custom.css' );
```

custom.css

```
#menu { background: #777; color: #333;}
#menu ul li a { color: #fff;}
#menu ul li a:hover { color: #ff0;}
```

Dentro do nosso cabeçalho chamarei o arquivo apenas quando for exibida uma página de busca. Para surtir efeito é preciso ainda que a declaração esteja sendo exibida após a inserção do estilo principal.

header.php

```
if ( is_search() ) wp_enqueue_style( 'custom' );
```



Apesar do nome dado ao identificador do estilo ser o mesmo que o do arquivo, essa não é uma regra a ser seguida. O termo que irá ser recuperado será o do identificador.

A prática foi adotada por evitar confusões desnecessárias de nomenclatura. O mesmo alerta vale também para os scripts.

Tradução

Durante todo o desenvolvimento estivemos preparando nosso tema para tradução incluindo os termos a serem utilizados nas funções de internacionalização do PHP e WordPress. Agora iremos traduzir os termos com a ajuda de um software livre chamado PoEdit.



Para fazer o download do software acesse o site oficial do software em <http://www.poedit.net/download.php>

PoEdit

Poedit é um editor de arquivos .po de modo visual, com recursos que facilitam o manuseio e aplicação das traduções dos textos. Para iniciar um processo de tradução execute o software e siga as etapas:

- Clique em 'Arquivo' e logo após 'Novo catálogo...';
- Na tela a seguir informe os dados do projeto com o qual está trabalhando, para tabela de caracteres escolha utf-8;
- Na aba 'Caminhos', insira o caminho base para seu tema; ou seja; o fluxo de pastas que o sistema deverá percorrer até encontrar os arquivos a serem traduzidos;
- Insira o caminho na listagem exibida abaixo do campo caminho base (Ex.: C:\wamp\www\wordpress\wp-content\themes\aprendiz);
- Na aba 'Palavras-chave' deverão ser inseridas as funções de tradução que estamos incluindo aos arquivos desde o início do

desenvolvimento do tema (Ex.: __, _e, _n);

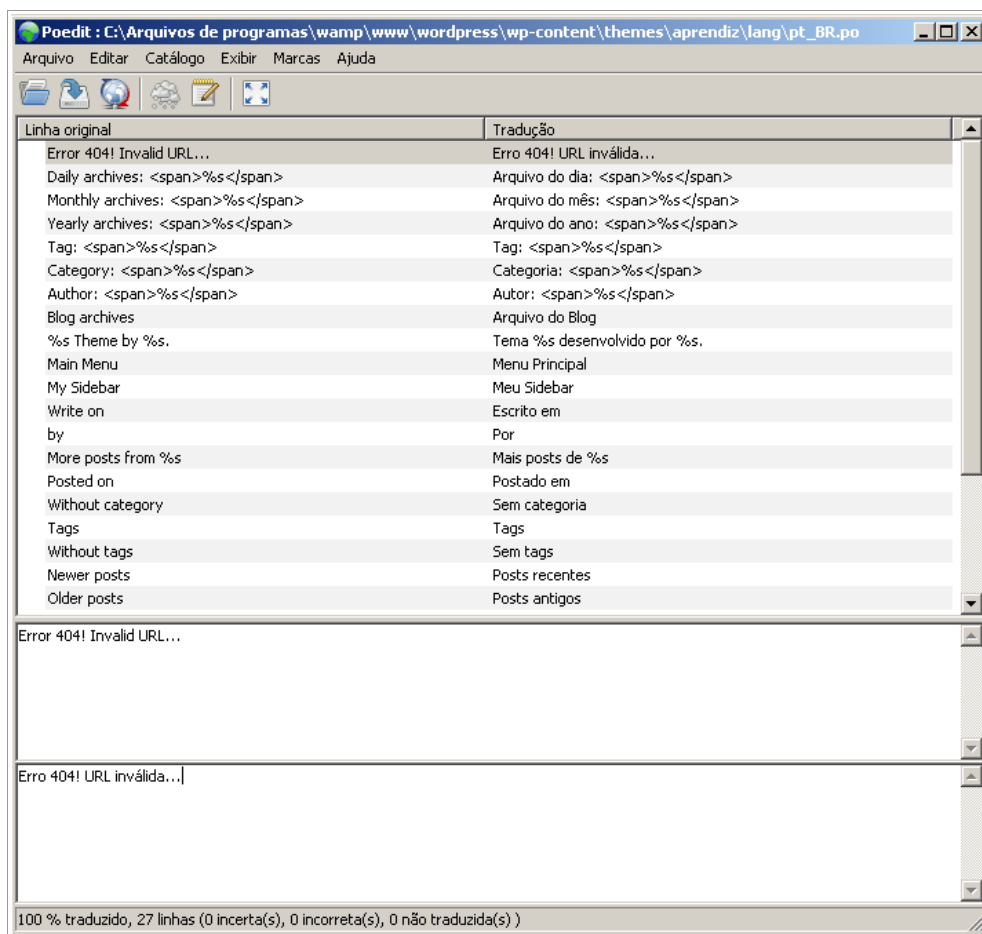
- Ao final clique em OK para confirmar suas ações;
- Feito isso clique agora em Catálogo > Atualizar a partir do código fonte' (ou no terceiro ícone, com o globo desenhado) para o PoEdit capturar todas as entradas dos arquivos existentes no endereço informado e mostrá-las em tela.

Traduzindo

Clicando sobre determinado texto, embaixo aparecerão dois campos de texto: o primeiro deles com o texto original e o segundo, editável, aguarda o texto traduzido. Apertando Alt + C quando escolher determinado termo, este é repetido para a tradução. Caso exista dúvida quanto a tradução de uma certa palavra, aperte Alt + U para marcá-la ou desmarcá-la como incerta dependendo de sua atual situação.

Realizada a tradução de todos os termos, no arquivo de funções functions.php inclua:

```
functions.php
load_theme_textdomain( 'Default', TEMPLATEPATH.'/lang' );
```



PoEdit com todos os termos traduzidos



Os arquivos utilizados nesse capítulo estão disponíveis no arquivo **aprimoramentos.rar** anexo ao livro.

Considerações Finais

Parabéns! Agora você já está apto a entrar de vez no universo de desenvolvimento do WordPress. Com os conceitos aqui vistos você conheceu a ferramenta, todo seu funcionamento de forma geral e já tornou-se capaz de produzir seu próprio material com base em um template web.

Você perceberá que também acabou de aprender a lidar mais facilmente com eventuais atualizações, temas, plugins e reparos ao seu sistema quando for necessário.

Ao continuar trabalhando com o WordPress você verá ainda o quão fácil é implementar recursos que possuem enormes atrativos visuais e funcionais ao teu trabalho sem complicações, em poucas linhas de código, baseado nas informações contidas nesse livro.

Para facilitar sua busca por informações quando utilizar o livro como referência, a seguir temos uma listagem com as funções utilizadas no tema. Todas elas bem explicadas e detalhadas com suas respectivas descrições, modo de uso, parâmetros e valores retornados.

Apêndice A: Referência de funções

Descrição

Um alias para `translate`. Retorna a string traduzida para o termo passado como parâmetro.

Uso

`__($text, $domain)`

Parâmetros

`$text (string)` (obrigatório)

Texto a ser traduzido

Padrão: Nenhum

`$domain (string)` (opcional)

Domínio de onde o texto traduzido será recuperado

Padrão: 'Default'

Retorna

(string) Texto traduzido

_e

Descrição

Exibe a string traduzida para o termo passado como parâmetro.

Uso

```
_e( $text, $domain )
```

Parâmetros

`$text` (string) (obrigatório)

Texto a ser traduzido

Padrão: Nenhum

`$domain` (string) (opcional)

Domínio de onde o texto traduzido será recuperado

Padrão: 'Default'

Retorna

Nada

bloginfo

Descrição

Mostra informações sobre o blog, em sua maioria as que são definidas nas Opções Gerais do Painel Administrativo do WordPress.

Uso

```
bloginfo( $option )
```

Parâmetros

`$option (string) (opcional)`

Palavra-chave que referencia a informação que se deseja obter.

Padrão: name

Lista de opções: name, description, admin_email, url, wpurl, stylesheet_directory, stylesheet_url, template_directory, template_url, atom_url, rss2_url, rss_url, pingback_url, rdf_url, comments_atom_url, comments_rss2_url, charset, html_type, language, text_direction, version

body_class

Descrição

Reúne e exibe classes para o elemento body

Uso

```
body_class( $class )
```

Parâmetros

`$class` (string) (opcional)

Uma ou mais classes para adicionar a lista de classes

Padrão: Nada

Retorna

(string) Lista de classes para o elemento body

comments_popup_link

Descrição

Exibe um link para uma janela popup ou um link normal para comentários.

Uso

```
comments_popup_link( $zero, $one, $more, $css_class, $none );
```

Parâmetros

`$zero` (string) (opcional)

Texto mostrado quando não há comentários

Padrão: 'No Comments'.

`$one` (string) (opcional)

Texto mostrado quando há um único comentário

Padrão: '1 Comment'.

`$more` (string) (opcional)

Texto mostrado quando há mais de um comentário. O símbolo '%' é substituído pelo número de comentários

Padrão: '% Comments'.

`$css_class` (string) (opcional)

Classe CSS para o link

Padrão: Nenhum

`$none` (string) (opcional)

Texto mostrado quando comentários estão desabilitados.

Padrão: 'Comments Off'.

Retorna

Nada

comments_template

Descrição

Carrega o modelo de comentários padrão do WordPress

Uso

```
comments_template( $file, $separate_comments )
```

Parâmetros

`$file` (string) (opcional)

O arquivo a ser carregado

Padrão: /comments.php

`$separate_comments` (boolean) (opcional)

Informa se os comentários deverão ser separados por tipo

Padrão: false

Retorna

Nada

dynamic_sidebar

Descrição

Imprime o conteúdo de cada um dos Widget ativos

Uso

```
dynamic_sidebar( $index )
```

Parâmetros

`$index` (integer|string) (opcional)

Nome ou ID da Sidebar

Padrão: 1

Retorna

(boolean) Verdadeiro se encontrou algum Widget e falso para o contrário.

esc_attr**Descrição**

Codifica < > & " ' (sinal de menor, sinal de maior, 'e' comercial, aspas duplas, aspas simples)

Uso

```
esc_attr( $texto )
```

Parâmetros

\$texto (string) (obrigatório)

O texto a ser codificado

Padrão: Nenhum

Retorna

(string) O texto codificado com entidades HTML

get_author_posts_url

Descrição

Retorna o permalink do autor com base no identificador passado

Uso

```
get_author_posts_url( $id )
```

Parâmetros

\$id (integer) (obrigatório)

ID do usuário que deseja se obter o link

Padrão: Nenhum

Retorna

(string) URL para a página de posts do referido usuário

get_day_link

Descrição

Retorna a URL de arquivo de um dia específico

Uso

```
get_day_link( $year, $month, $day )
```

Parâmetros

`$year` (boolean|integer) (opcional)

O ano. Informe " para o ano atual.

Padrão: Nenhum

`$month` (boolean|integer) (opcional)

O mês. Informe " para o mês atual.

Padrão: Nenhum

`$day` (boolean|integer) (opcional)

O dia. Informe " para o dia atual.

Padrão: Nenhum

Retorna

(string) Endereço do link do dia

get_footer

Descrição

Inclui o arquivo footer.php do tema atual

Uso

```
get_footer( $name )
```

Parâmetros

\$name (string) (opcional)

Executa a chamada para footer-name.php.

Padrão: Nenhum

Retorna

Nada

get_header

Descrição

Inclui o arquivo header.php do tema atual

Uso

```
get_header( $name )
```

Parâmetros

\$name (string) (opcional)

Executa a chamada para header-name.php.

Padrão: Nenhum

Retorna

Nada

get_month_link

Descrição

Retorna a URL de arquivo de um mês específico

Uso

```
get_month_link( $year, $month )
```

Parâmetros

`$year` (boolean|integer) (opcional)

O ano. Informe " para o ano atual.

Padrão: Nenhum

`$month` (boolean|integer) (opcional)

O mês. Informe " para o mês atual.

Padrão: Nenhum

Retorna

(string) Endereço do link do mês

get_option

Descrição

Uma maneira segura de se obter valores para uma opção recuperada do banco de dados.

Uso

```
get_option( $show, $default )
```

Parâmetros

`$show` (string) (obrigatório)

Nome da opção a ser recuperada

Padrão: Nenhum

`$default` (mixed) (opcional)

O valor padrão é retornado se a função não retorna nenhum valor

Padrão: false

Retorna

(mixed) O valor da opção requisitada

get_search_form**Descrição**

Mostra o formulário de busca com base no arquivo searchform.php se esse existir.

Uso

```
get_search_form()
```

Parâmetros

Nenhum

Retorna

Nada

get_search_query

Descrição

Retorna o termo de pesquisa para a requisição atual, se uma busca foi executada

Uso

```
get_search_form()
```

Parâmetros

Nenhum

Retorna

(string) Termo pesquisado

get_sidebar

Descrição

Inclui o arquivo sidebar.php do tema atual

Uso

```
get_sidebar( $name )
```

Parâmetros

\$name (string) (opcional)

Executa a chamada para sidebar-name.php.

Padrão: Nenhum

Retorna

Nada

get_template_part

Descrição

Carrega arquivos externos

Uso

```
get_template_part( $slug, $name )
```

Parâmetros

`$slug` (string) (obrigatório)

O termo slug do arquivo a ser incluído

Padrão: Nenhum

`$name` (string) (opcional)

O nome de um arquivo específico

Padrão: Nenhum

Retorna

Nada

get_the_author

Descrição

Recupera o autor do post

Uso

```
get_the_author()
```

Parâmetros

Nenhum

Retorna

(string) O nome de exibição do autor do post

get_the_author_meta

Descrição

Recupera uma meta-informação do autor do post

Uso

```
get_the_author_meta( $meta )
```

Parâmetros

\$meta (string) (obrigatório)

Identifica qua informação do autor se deseja obter

Padrão: Nenhum

Retorna

(string) Meta-informação do autor do post

get_the_category

Descrição

Retorna um array de objetos, sendo cada um deles, uma categoria a qual o post está incluso.

Uso

```
get_the_category( $id )
```

Parâmetros

`$id` (integer) (obrigatório)

O identificador do post.

Padrão: `$post->ID` (o ID do post atual)

Retorna

(array) As categorias do post

get_the_category_list**Descrição**

Retorna uma string contendo todas as categorias do post em forma de links

Uso

```
get_the_category_list( $sep )
```

Parâmetros

\$sep (string) (obrigatório)

Separador que será incluído entre os links

Padrão: Lista não ordenada,

Retorna

(string) Links HTML prontos para exibição

get_the_date

Descrição

Retorna a data em que o post foi escrito respeitando o formato passado como parâmetro

Uso

```
get_the_date( $d )
```

Parâmetros

\$d (string) (opcional)

Formato de data

Padrão: O formato de data escolhido via painel administrativo ('Formato das datas' em Configurações > Geral)

Retorna

(string) Data formatada

get_the_tag_list

Descrição

Gera um HTML contendo as tags associadas ao post atual, onde cada tag está linkada a respectiva página.

Uso

```
get_the_tag_list( $before, $sep, $after )
```

Parâmetros

`$before` (string) (opcional)

Texto a ser inserido antes da string de retorno

Padrão: 'Tags: '

`$sep` (string) (opcional)

Separador que será incluído entre os links

Padrão: ', '

`$after` (string) (opcional)

Texto a ser inserido ao final da string de retorno

Padrão: Nenhum

Retorna

(string) Links HTML prontos para exibição

get_the_tags**Descrição**

Retorna um array de objetos, sendo cada um deles, uma tag a qual pertence ao post

Uso

`get_the_tags()`

Parâmetros

Nenhum

Retorna

(array) As tags do post

get_userdata**Descrição**

Retorna um objeto com as informações referentes ao usuário cujo ID é passado para ele

Uso

```
get_userdata( $id )
```

Parâmetros

`$id` (integer) (obrigatório)

Identificador do usuário que se deseja obter informações

Padrão: Nenhum

Retorna

(object) Informações vinculadas ao usuário

get_year_link

Descrição

Retorna a URL de arquivo de um ano específico

Uso

```
get_year_link( $year )
```

Parâmetros

`$year` (boolean|integer) (opcional)

O ano. Informe " para o ano atual.

Padrão: Nenhum

Retorna

(string) Endereço do link do ano

have_posts

Descrição

Verifica a existência de posts para a atual consulta

Uso

`have_posts()`

Parâmetros

Nenhum

Retorna

(boolean) Caso a consulta tenha resultado retorna verdadeiro, do contrário, falso

is_author

Descrição

Verifica se uma página de arquivo de Autor está sendo exibida

Uso

```
is_author( $author )
```

Parâmetros

`$author` (integer|string) (opcional)

ID ou apelido do autor

Padrão: Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_category

Descrição

Verifica se uma página de arquivo de Categoria está sendo exibida

Uso

```
is_category( $category )
```

Parâmetros

`$category` (integer|string) (opcional)

ID, título ou slug da categoria

Padrão: Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_day

Descrição

Verifica se uma página de arquivo de Dia está sendo exibida

Uso

`is_day()`

Parâmetros

Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_home

Descrição

Verifica se a página inicial está sendo exibida

Uso

```
is_home()
```

Parâmetros

Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_month**Descrição**

Verifica se uma página de arquivo de Mês está sendo exibida

Uso

`is_month()`

Parâmetros

Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_page

Descrição

Verifica se uma página está sendo exibida

Uso

```
is_page( $page )
```

Parâmetros

`$page` (integer|string) (opcional)

ID, título ou slug da página

Padrão: Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_single

Descrição

Verifica se uma página de post está sendo exibida

Uso

```
is_single( $post )
```

Parâmetros

`$post` (integer|string) (opcional)

ID, título ou slug do post

Padrão: Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_tag

Descrição

Verifica se uma página de arquivo de Tag está sendo exibida

Uso

```
is_tag( $tag )
```

Parâmetros

\$tag (integer|string) (opcional)

ID, título ou slug da tag

Padrão: Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

is_year

Descrição

Verifica se uma página de arquivo de Ano está sendo exibida

Uso

`is_year()`

Parâmetros

Nenhum

Retorna

(Boolean) Verdadeiro ou falso para a condição

language_attributes

Descrição

Exibe os atributos de linguagem para a tag <html>. Identifica o idioma em uso e também a direção do texto para a página.

Uso

```
language_attributes( $doctype )
```

Parâmetros

\$doctype (string) (opcional)

O tipo html do documento 'xhtml' ou 'html'.

Padrão: 'html'

Retorna

Nada

load_theme_textdomain

Descrição

Carrega as strings traduzidas do tema

Uso

```
load_theme_textdomain( $domain, $path )
```

Parâmetros

`$domain` (string) (obrigatório)

Identificador único para recuperar as strings traduzidas

Padrão: Nenhum

`$path` (string) (opcional)

Caminho dos arquivos de tradução

Padrão: Pasta do tema ativo

Retorna

Nada

next_posts_link

Descrição

Imprime um link para o próximo conjunto de posts dentro da consulta atual

Uso

```
next_posts_link( $label, $max_pages )
```

Parâmetros

`$label` (string) (opcional)

Texto do link

Padrão: 'Next Page »'

`$max_pages` (integer) (opcional)

Limita o número de página nas quais o link é exibido

Padrão: 0 – Sem limite

Retorna

Nada

post_class

Descrição

Reúne e exibe classes para o elemento div do post

Uso

```
post_class( $class )
```

Parâmetros

`$class` (string) (opcional)

Uma ou mais classes para adicionar a lista de classes

Padrão: Nenhum

Retorna

(string) Lista de classes

previous_posts_link

Descrição

Imprime um link para o conjunto de posts anterior dentro da consulta atual

Uso

```
previous_posts_link( $label, $max_pages )
```

Parâmetros

`$label` (string) (opcional)

Texto do link

Padrão: '<< Previous Page'

`$max_pages` (integer) (opcional)

Limita o número de página nas quais o link é exibido

Padrão: 0 – Sem limite

Retorna

Nada

register_nav_menu

Descrição

Registra um único Menu de navegação personalizado

Uso

```
register_nav_menu( $location, $description )
```

Parâmetros

`$location` (string) (obrigatório)

Identificador de localização do menu, como um slug.

Padrão: Nenhum

`$description` (string) (obrigatório)

Não possui valor como padrão. Deverá ser especificado mesmo se vazio. Servirá para rotular o menu criado.

Padrão: Nenhum

Retorna

Nada

single_tag_title

Descrição

Mostra ou retorna o título da tag da página atual

Uso

```
single_tag_title( $prefix, $display )
```

Parâmetros

`$prefix` (string) (opcional)

Texto a ser retornado antes do título

Padrão: Nenhum

`$display` (boolean) (opcional)

Mostra ou retorna o título da tag se verdadeiro ou falso

Padrão: true

Retorna

(null|string) Nada ou título da tag

single_cat_title

Descrição

Mostra ou retorna o título da categoria da página atual

Uso

```
single_cat_title( $prefix, $display )
```

Parâmetros

`$prefix` (string) (opcional)

Texto a ser retornado antes do título

Padrão: Nenhum

`$display` (boolean) (opcional)

Mostra ou retorna o título da tag se verdadeiro ou falso

Padrão: true

Retorna

(null|string) Nada ou título da tag

register_sidebar

Descrição

Constrói a definição para uma única Sidebar

Uso

```
register_sidebar( $args )
```

Parâmetros

`$args` (string|array) (opcional)

Constrói a Sidebar baseado nos valores de 'name' e 'id'.

Padrão: Nenhum

Lista de parâmetros: name, id, description, before_widget, after_widget, before_title, after_title

Retorna

(integer|string) Identificador da Sidebar

the_author

Descrição

Mostra o nome de exibição do autor do post

Uso

`the_author()`

Parâmetros

Nenhum

Retorna

Nada

the_date

Descrição

Exibe a data de publicação do post

Uso

```
the_date( $format, $before, $after, $echo )
```

Parâmetros

`$format` (string) (opcional)

Formato da data

Padrão: F j, Y

`$before` (string) (opcional)

Texto a ser exibido antes da data

Padrão: Nenhum

`$after` (string) (opcional)

Texto a ser exibido depois da data

Padrão: Nenhum

`$echo` (boolean) (opcional)

Exibe a data quando verdadeiro ou retorna quando falso.

Padrão: true

Retorna

(null|string) Nada ou a data formatada

the_excerpt

Descrição

Mostra o resumo do post

Uso

`the_excerpt()`

Parâmetros

Nenhum

Retorna

Nada

the_permalink

Descrição

Exibe o link do post de acordo com as configurações de links permanente do painel administrativo

Uso

`the_permalink()`

Parâmetros

Nenhum

Retorna

Nada

the_post**Descrição**

Faz um post obtido através da iteração do Loop tornar-se global

Uso

`the_post()`

Parâmetros

Nenhum

Retorna

Nada

the_search_query

Descrição

Exibe o termo de pesquisa para a requisição atual, se uma busca foi executada

Uso

`the_search_form()`

Parâmetros

Nenhum

Retorna

Nada

the_title

Descrição

Exibe o título do post

Uso

```
the_title( $before, $after, $echo )
```

Parâmetros

`$before` (string) (opcional)

Texto a ser exibido antes do título

Padrão: "

`$after` (string) (opcional)

Texto a ser exibido depois do título

Padrão: "

`$echo` (boolean) (opcional)

Exibe o título quando verdadeiro ou retorna quando falso.

Padrão: true

Retorna

(null|string) Nada ou o título de acordo com o valor passado para `$echo`

wp_enqueue_script

Descrição

Um modo seguro de se adicionar scripts nas páginas do tema

Uso

```
wp_enqueue_script( $handle, $src, $deps, $ver, $in_footer )
```

Parâmetros

`$handle` (string) (obrigatório)

Nome do script, como um slug.

Padrão: Nenhum

`$src` (string) (opcional)

URL do script

Padrão: Nenhum

`$deps` (array) (opcional)

Lista de scripts que devem ser carregados antes do script principal, pelo fato desse último depender do outro.

Padrão: `array()`

`$ver` (string) (opcional)

Especifica qual a versão do script

Padrão: `false`

`$in_footer` (boolean) (opcional)

Normalmente os scripts são colocados na seção `<head>`. Se este parâmetro for verdadeiro, o script é colocado na parte inferior do `<body>`

Padrão: `false`

Retorna

Nada

wp_enqueue_style

Descrição

Um modo seguro de se adicionar estilos nas páginas do tema

Uso

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

Parâmetros

`$handle` (string) (obrigatório)

Nome do estilo, como um slug.

Padrão: Nenhum

`$src` (string) (opcional)

Caminho para o arquivo do estilo

Padrão: Nenhum

`$deps` (array) (opcional)

Lista de estilos que devem ser carregados antes do estilo principal

Padrão: array()

`$ver` (string) (opcional)

Especifica qual a versão do estilo

Padrão: false

`$media` (string) (opcional)

Especifica os meios para o qual o estilo foi definido, como 'all', 'screen', 'handheld' e 'print'

Padrão: false

Retorna

Nada

wp_footer

Descrição

Hook de identificação do rodapé

Uso

```
wp_footer()
```

Parâmetros

Nenhum

Retorna

Nada

wp_get_archives

Descrição

Exibe uma lista de arquivos baseada pela data.

Uso

```
wp_get_archives( $args )
```

Parâmetros

`type (string) (opcional)`

O tipo de lista de arquivo a ser mostrada.

Padrão: monthly

Valores válidos: yearly, monthly, daily, weekly, postbypost (posts ordenados por data de postagem), alpha (posts ordenados pelo título)

`limit (integer) (opcional)`

Número máximo de arquivos a se obter

Padrão: 0

`format (string) (opcional)`

Formato para a lista de arquivo. Valores válidos:

- `html` – Em listas HTML () - Padrão
- `option` – Em um menu de seleção ou de opções
- `link` – Dentro das tags <link>
- `custom` – Lista personalizada usando strings para antes e depois de cada link

`before (string) (opcional)`

Texto a ser exibido antes do link quando a opção `format` estiver definida como `custom`

Padrão: Nenhum

`after (string) (opcional)`

Texto a ser exibido depois do link quando a opção `format` estiver definida como `custom`

Padrão: Nenhum

`show_post_count (boolean) (opcional)`

Exibe ou não a quantidade de posts que o arquivo possui

Padrão: `False`

`echo (boolean) (opcional)`

Exibe ou retorna a lista quando `true` e `false` respectivamente

Padrão: `true`

Retorna

Nada

wp_head

Descrição

Hook de identificação do cabeçalho

Uso

`wp_head()`

Parâmetros

Nenhum

Retorna

Nada

wp_nav_menu

Descrição

Exibe um menu de navegação criado através do painel administrativo

Uso

`wp_nav_menu($args)`

Parâmetros

`$menu` (string) (opcional)

O menu desejado; aceita id, slug, name

Padrão: Nenhum

`$container` (string) (opcional)

Tag de blog onde o menu será inserido

Padrão: div

`$container_class` (string) (opcional)

A classe a ser aplicada ao container

Padrão: menu-{menu slug}-container

`$container_id` (string) (opcional)

O identificador a ser aplicado ao container

Padrão: Nenhum

`$menu_class` (string) (opcional)

A classe CSS da tag ul que compõe o menu

Padrão: menu

`$menu_id` (string) (opcional)

O identificador da tag `ul` que compõe o menu

Padrão: menu slug, incrementado

`$echo` (boolean) (opcional)

Determina se o menu será exibido ou retornado quando verdadeiro ou falso, respectivamente

Padrão: true

`$fallback_cb` (string) (opcional)

Se o menu não existir, executa a função aqui explícita

Padrão: `wp_page_menu`

`$before` (string) (opcional)

Texto exibido antes da tag `<a>` do link

Padrão: Nenhum

`$after` (string) (opcional)

Texto exibido depois da tag `<a>` do link

Padrão: Nenhum

`$link_before` (string) (opcional)

Texto exibido antes do texto do link

Padrão: Nenhum

`$link_after` (string) (opcional)

Texto exibido depois do texto do link

Padrão: Nenhum

`$depth` (integer) (opcional)

Quantos níveis de hierarquia serão incluídos

Padrão: 0 - Todos

`$walker` (object) (opcional)

Objeto personalizado

Padrão: `new Walker_Nav_Menu`

`$theme_location` (string) (opcional)

A localização do tema a ser utilizada. Deve ser registrado com `register_nav_menu()`, a fim de ser selecionável pelo usuário

Padrão: Nenhum

Retorna

(boolean) Verdadeiro ou falso se encontrou ou não um menu personalizado

wp_title

Descrição

Exibe o texto do título de acordo com a consulta que está sendo executada:

- Post ou página - Título do post ou página
- Arquivo de datas - A própria data
- Categorias - O nome da categoria
- Página do autor - O nome do autor do post

Uso

`wp_title($sep, $echo, $seplocation)`

Parâmetros

`$sep (string) (opcional)`

Texto a ser exibido antes ou depois do título

Padrão: `»`; (`»`)

`$echo (boolean) (opcional)`

Exibe o título quando verdadeiro ou retorna quando falso.

Padrão: `true`

`$seplocation (string) (opcional)`

Define a localização do texto separador do título. Qualquer valor exibirá o separador a esquerda do título, exceto pelo termo 'right' que exibe o separadr a direita do título.

Padrão: Nenhum

Retorna

Nada

Apêndice B: Funções utilizadas

bloginfo.....	51	language_attributes.....	50
body_class.....	86	load_theme_textdomain.....	93
comments_popup_link.....	74	next_posts_link.....	77
dynamic_sidebar.....	59	post_class.....	86
esc_attr.....	53	previous_posts_link.....	77
get_bloginfo.....	88	register_nav_menu.....	54
get_footer.....	71	register_sidebar.....	58
get_header.....	71	single_cat_title.....	80
get_option.....	87	single_tag_title.....	80
get_search_form.....	71	the_excerpt.....	78
get_search_query.....	79	the_permalink.....	62
get_sidebar.....	71	the_post.....	61
get_template_part.....	72	the_search_query.....	58
get_the_author.....	63	the_title.....	62
get_the_date.....	80	wp_enqueue_script.....	87
get_userdata.....	80	wp_enqueue_style.....	91
have_posts.....	61	wp_footer.....	65
is_category.....	80	wp_get_archives.....	59
is_day.....	80	wp_head.....	64
is_month.....	80	wp_nav_menu.....	55
is_singular.....	87	wp_register_script.....	88
is_tag.....	80	wp_register_style.....	91
is_year.....	80	wp_title.....	52