

Como criar um plugin para WordPress

Por @leandrovieira

O que é um plugin WordPress?

Um plugin WordPress é um programa, um conjunto de uma ou várias funções, escrito em PHP e adiciona recursos ou serviços específicos ao WordPress através de sua API.

Devo realmente criar um plugin WordPress?

- Consulte o diretório de plugins WordPress em <http://wordpress.org/extend/plugins/>
- Há mais de 5 mil publicados;
- 28 de Janeiro declarado como o “Thank a Plugin Developer” quando o diretório atingiu a marca de 4 mil plugins publicados;
- Alguma função nativa do WordPress? Consulte http://codex.wordpress.org/Function_Reference
- Posso adaptar um plugin para minha necessidade sabendo como criá-los.

O que você precisa para criar um plugin WordPress?

- Um problema pra resolver;
- Ambiente de desenvolvimento em PHP (XAMPP);
- Conhecimentos em PHP;
- Conhecimentos na API de Plugin do WordPress;
- Arquivos de exemplo: <http://leandrovieira.com/download/9>

Estrutura de um plugin WordPress

- No mínimo um arquivo PHP;
- Se tiver vários arquivos armazenar num diretório;
- Nome do plugin: único e descritivo. Exemplo: CMS Brasil 2009;
- Nome do arquivo PHP: cms-brasil-2009.php;
cmsbrasil2009.php
- Nome do diretório: cms-brasil-2009/cms-brasil-2009.php;
cmsbrasil2009/cmsbrasil2009.php
- Armazenado em /wp-content/plugins/
- Se for hospedar o plugin no diretório do WordPress.org é necessário criar o arquivo readme.txt;
- Licença de uso do plugin GPL ou alguma compatível;
- Cabeçalho do plugin.

Cabeçalho de um plugin WordPress

```
<?php
/*
Plugin Name: CMS Brasil 2009
Plugin URI: http://cmsbrasil2009.com/
Description: Conferência internacional de CMS
Author: iMasters
Version: 1.0
Author URI: http://imasters.uol.com.br
*/
?>
```

Como instalar um plugin WordPress

- Via FTP;
- Upload do plugin em formato ZIP (a partir da versão 2.7);
- Baixar do diretório de plugins do WordPress.org;
- Ativar o plugin desejado.

API de Plugin do WordPress

- Hooks (ganchos);
- Acrescentar funcionalidades sem editar o CORE do WordPress;
- Há dois tipos de hooks (ganchos):
 - Actions (ações);
 - São ganchos executados pelo WordPress em momentos específicos durante a execução ou quando algum evento ocorre.
 - Filters (filtros)
 - São ganchos executados pelo WordPress para modificar textos ou outros tipos antes de inserir no banco de dados ou exibir no navegador de internet.

Actions

Como registrar uma função para responder um evento do WordPress

- No arquivo do plugin crie uma função PHP;
- Algumas ações recebem mais de um parâmetro. É possível definir quantos receber;
- Registre-a com o WordPress chamando a função `add_action`;

```
add_action ( 'hook_name', 'your_function_name', [priority], [accepted_args] );  
add_action ( 'publish_post', 'nome_funcao' );
```

Actions

Exemplo:

```
function cmsbrasil2009_url_after_login()  
{  
    global $redirect_to;  
    if ( !isset( $_GET['redirect_to'] ) )  
        $redirect_to = get_bloginfo( 'home' );  
}  
add_action( 'login_form', 'cmsbrasil2009_url_after_login' );
```

Filters

Como registrar uma função de filtro para tratar dados no WordPress

- No arquivo do plugin crie uma função PHP para fazer o processo de filtragem;
- Alguns filtros recebem mais de um parâmetro. É possível definir quantos receber;
- Registre-a com o WordPress chamando a função `add_filter`.

```
add_filter ( 'hook_name', 'your_filter', [priority], [accepted_args] );  
add_filter ( 'the_title', 'nome_funcao' );
```

Filters

```
function cmsbrasil2009_cms_title( $title )
{
    $cms = preg_match( "/(wordpress|joomla|drupal)/i", $title, $matches );
    $cms_name = strtolower( $matches[0] );
    switch( $cms_name ) :
        case 'wordpress' :
            return '' . $title;
            break;
        case 'drupal' :
            return '' . $title;
            break;
    endswitch;
}
add_filter( 'the_title', 'cmsbrasil2009_cms_title' );
```

Filters

```
function primeira( $title )
{
    return 'T1: ' . $title;
}
add_filter( 'the_title', 'primeira' );
```

```
function segunda( $title )
{
    return 'T2: ' . $title;
}
add_filter( 'the_title', 'segunda' );
```

Resultado: T2: T1: Título do post

```
function primeira( $title )
{
    return 'T1: ' . $title;
}
add_filter( 'the_title', 'primeira', 11 );
```

```
function segunda( $title )
{
    return 'T2: ' . $title;
}
add_filter( 'the_title', 'segunda', 10 );
```

Resultado: T1: T2: Título do post

Removendo Actions e Filters

Da mesma forma que podemos incluir ações para responder eventos do WordPress e filtros para tratar dados, podemos também desabilitar um desses recursos nativos do WordPress ou de algum outro plugin.

```
remove_action( 'nome_do_gancho', 'nome_da_funcao' );  
remove_filter( 'nome_do_gancho', 'nome_da_funcao' );
```

Atente-se ao fato que para remover um gancho criado usando prioridade diferente de 10, se faz necessário informa-la ao chamar a função `remove_action`.

```
remove_filter( 'nome_do_gancho', 'nome_da_funcao', 11 );
```

Funções "Pluggable"

Além dos ganchos (actions e filters) existem outras maneiras de modificar o comportamento do WordPress: sobreescrevendo suas funções.

As funções que o WordPress permite serem sobreescritas são chamadas de Pluggable Functions e são definidas no seguinte arquivo: `wp-includes/pluggable.php`. Essas funções são carregadas somente após o carregamento de todos plugins e somente se elas não tiverem sido reescritas.

```
if ( !function_exists('is_user_logged_in') ) :  
function is_user_logged_in() {  
    ...  
}  
endif;
```

Templates tags (tags de templates)

Definir uma tag de template é simples. Escreva uma função PHP em seu plugin e explique aos usuários como e onde chama-las.

É boa prática sempre verificar se a função ou classe existe antes de chama-la.

```
<?php if ( function_exists( 'nome_da_funcao' ) ) nome_da_funcao(); ?>
```

```
<?php if ( class_exists( 'NomeDaClasse' ) ) NomeDaClasse::metodo(); ?>
```

Criando menus e páginas administrativas

Para criar páginas administrativas precisamos de três coisas:

- Criar uma função contendo os códigos utilizados para criar os menus;
 - `add_menu_page`;
 - `add_submenu_page`;
 - `add_options_page`;
 - ...
- Registrar tal função usando o gancho "admin_menu";
- Criar o HTML da página a ser exibida quando o menu for clicado.

* Os menus são criados "on the fly". Eles serão recriados a cada acesso a uma das páginas administrativa.

Criando menus e páginas administrativas

```
function cmsbrasil2009_menu()  
{  
    add_menu_page( 'Inscrição: iMasters CMS Brasil 2009', 'Inscrição', 10,  
'cmsbrasil2009/inscricao.php' );  
  
    add_submenu_page( 'cmsbrasil2009/inscricao.php', 'Inscrições pendentes',  
'Pendentes', 10, 'cmsbrasil2009/pendentes.php' );  
  
}  
add_action( 'admin_menu', 'cmsbrasil2009_menu' );
```

Armazenamento de dados

- Para pequena quantidade de dados ou dados estáticos, use o mecanismo de opções do WordPress;
- Para informações referente a posts, use Post meta;
- Para informações referente a usuários, use User meta;
- Para grande quantidade de dados, crie suas tabelas de banco de dados.

Options

O WordPress tem um mecanismo para salvar, atualizar, excluir e resgatar dados nomeado como opções. Os valores dessas opções podem ser strings, arrays ou objetos PHP. Esses dados serão serializados ou convertidos para string antes de serem armazenados e deserializados quando resgatados.

As opções devem ter nomes únicos para evitar conflitos com o WordPress e outros plugins.

Options

Funções para trabalhar com as opções do WordPress

- **add_option;**
 - `add_option($name, $value, $deprecated, $autoload);`
- **get_option;**
 - `get_option($option);`
- **update_option;**
 - `update_option($option_name, $newvalue);`
- **delete_option;**
 - `delete_option($name);`

1. A função `update_option` verifica se a opção existe antes de atualizá-la, se não existir ela será criada;
2. `deprecated` não é usado;
3. `autoload` é para a opção ser carregada pela função `get_alloptions`.

Options

Página de opções no padrão WordPress

```
<div class="wrap">
  <h2>Exemplo de opções</h2>
  <form method="post" action="options.php">
    <?php wp_nonce_field('update-options'); ?>
    <table class="form-table">
      <tr valign="top">
        <th scope="row">Nome da opção</th>
        <td><input type="text" name="cmsbrasil2009_opcao" value="<?php
echo get_option('cmsbrasil2009_opcao'); ?>" /></td>
      </tr>
    </table>
    <input type="hidden" name="action" value="update" />
    <input type="hidden" name="page_options" value="cmsbrasil2009_opcao" />
    <p class="submit">
      <input type="submit" class="button-primary" value="Salvar" />
    </p>
  </form>
</div>
```

Trabalhando com banco de dados

WordPress fornece uma classe de funções para toda manipulação em banco de dados. A classe é baseada na ezSQL escrita e mantida por Justin Vincent e se chama wpdb.

A classe está disponível no objeto global \$wpdb. Ele pode ser usada para trabalhar com todas tabelas do WordPress, as padrões e as criadas por plugins.

Quando for usar dentro de funções lembre-se de usar o global. O objeto acessa quantas tabelas existirem, mas somente um banco de dados, o do WordPress.

http://codex.wordpress.org/Function_Reference/wpdb_Class

Trabalhando com banco de dados

Funções

- query;
- get_var;
- get_row;
- get_col;
- get_results;
- insert;
- update;
- prepare;
- show_errors();
- hide_errors();
- print_error();
- get_col_info();
- flush;

Trabalhando com banco de dados

Variáveis

- `show_errors;`
- `num_queries;`
- `last_query;`
- `queries;`
- `last_results;`
- `col_info;`
- `insert_id.`

Melhores práticas

- Prefixo em nome de funções e variáveis; evite conflito;
- Considerações ao instalar e desinstalar plugins;
- `$wpdb->prefix` sempre;
- `$wpdb->prepare` sempre;
- Dê atenção a performance de suas consultas ao banco de dados; evite `SELECT * FROM tabela` se pode usar `SELECT ID FROM tabela`;
- `define('WP_DEBUG', true);` em `wp-config.php`
- Versão do WordPress;
- Versão do PHP;
- Use funções nativas do WordPress sempre que possível;
- Comente seus códigos;

Obrigado!

leandrovieira.com

apiki.com

[@leandrovieira](https://twitter.com/leandrovieira)