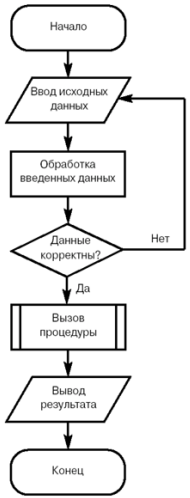
WordPress и ООП. Все что вы хотели, но боялись спросить

Катя Леурдо

WordPress разработчик, фрилансер

Процедурный подход



Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. (Википедия)

Объектный подход

Идеологически ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости[2], что существенно улучшает управляемость самим процессом моделирования, что в свою очередь особенно важно при реализации крупных проектов. (Википедия)

WordPress – событийная архитектура

События в лицевой части сайта

muplugins_loaded
registered_taxonomy
registered_post_type
plugins_loaded
sanitize_comment_cookies
setup_theme
load_textdomain
after_setup_theme
auth_cookie_malformed
auth_cookie_valid
set_current_user
init
widgets_init
register_sidebar
wp_register_sidebar_widget
wp_default_scripts
wp_default_styles
admin_bar_init
add_admin_bar_menus
wp_loaded
parse_request
send_headers
parse_query
pre_get_posts
posts_clauses
posts_selection
wp
template_redirect
get_header
twentyeleven_enqueue_color_scheme
wp_head
wp_enqueue_scripts
wp_print_styles
wp_print_scripts
get_search_form
loop_start

Допустим нам нужно перенаправить ссылку на страницу регистрации, если в запросе указана переменная register:

```php
add_action('init', 'redirect_to_register');
function redirect_to_register(){
    if( isset( $_GET['register'] ) ) {
        wp_redirect( site_url() . 'wp-register.php');
        exit;
    }
}
```

Фрагмент из: wp-settings.php VER 4.9.6

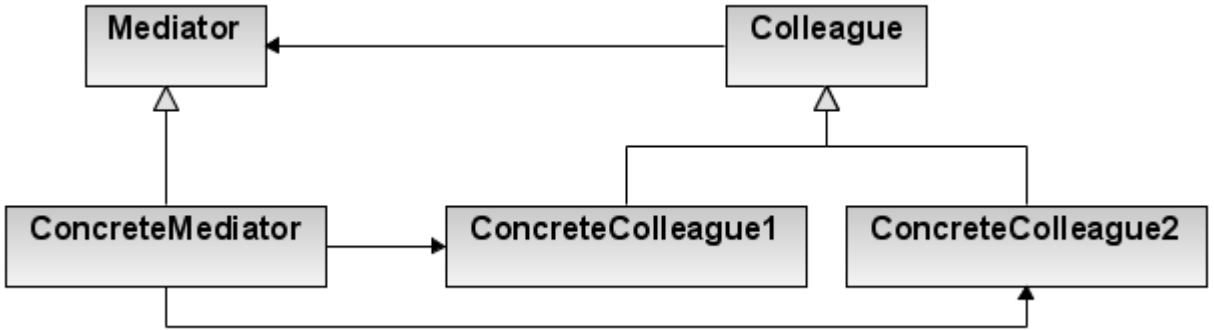
```php
...

// Set up current user.
$GLOBALS['wp']->init();

/**
 * Fires after WordPress has finished loading but before any headers are sent.
 *
 * Most of WP is loaded at this stage, and the user is authenticated. WP continues
 * to load on the {@see 'init'} hook that follows (e.g. widgets), and many plugins instanti
 * themselves on it for all sorts of reasons (e.g. they need a user, a taxonomy, etc.).
 *
 * If you wish to plug an action once WP is loaded, use the {@see 'wp_loaded'} hook below.
 *
 * @since 1.5.0
 */
do_action( 'init' );
```

https://wp-kama.ru/hooks/actions-order

Паттерн «посредник» (Mediator Pattern)



"Посредник" определяет интерфейс для обмена информацией с объектами "Коллеги", "Конкретный посредник" координирует действия объектов "Коллеги". Каждый класс "Коллеги" знает о своем объекте "Посредник", все "Коллеги" обмениваются информацией только с посредником, при его отсутствии им пришлось бы обмениваться информацией напрямую. "Коллеги" посылают запросы посреднику и получают запросы от него. "Посредник" реализует кооперативное поведение, пересылая каждый запрос одному или нескольким "Коллегам".

WordPress, классы и объекты

Самый известный класс в ядре WordPress

```php
/**
 * The WordPress Query class.
 *
 * @link https://codex.wordpress.org/Function_Reference/WP_Query Codex page.
 *
 * @since 1.5.0
 * @since 4.5.0 Removed the `$comments_popup` property.
 */
class WP_Query {

    /**
     * Query vars set by the user
     *
     * @since 1.5.0
     * @var array
     */
    public $query;

    /**
     * Query vars, after parsing
     *
     * @since 1.5.0
     * @var array
     */
    public $query_vars = array();

    /**
     * Taxonomy query, as passed to get_tax_sql()
     *
     * @since 3.1.0
     * @var object WP_Tax_Query
     */
    public $tax_query;
```

Я хочу свой класс

```php
class WP_Kickass_Plugin
{
    /**
     * Constructor.
     */
    public function __construct()
    {
        // Actions
        add_action('wp_enqueue_scripts', array($this, 'enqueue_script'));
    }

    /**
     * Enqueues our kickass scripts and stylesheets.
     */
    public function enqueue_script()
    {
        // ...
    }


}

new WP_Kickass_Plugin();
```

1) Нет доступа к объекту – давайте используем глобальную переменную

```php
$GLOBALS['kickass_plugin'] = new WP_Kickass_Plugin();
```

2) Хуки в конструкторе – давайте используем синглетон (singleton)

```php
class WP_Kickass_Plugin
{
    /**
     * The unique instance of the plugin.
     *
     * @var WP_Kickass_Plugin
     */
    private static $instance;

    /**
     * Gets an instance of our plugin.
     *
     * @return WP_Kickass_Plugin
     */
    public static function get_instance()
    {
        if (null === self::$instance) {
            self::$instance = new self();
        }

        return self::$instance;
    }

    /**
     * Constructor.
     */
    private function __construct()
    {
        // Actions
        add_action('wp_enqueue_scripts', array($this, 'enqueue_script'));
    }

    /**
     * Enqueues our kickass scripts and stylesheets.
     */
    public function enqueue_script()
    {
        // ...
    }
}

$kickass_plugin = WP_Kickass_Plugin::get_instance();
```

Идея №1 из: https://carlalexander.ca/singletons-in-wordpress/

```php
class WP_Kickass_Plugin
{
    /**
     * Registers our plugin with WordPress.
     */
    public static function register()
    {
        $plugin = new self();

        // Actions
        add_action('wp_enqueue_scripts', array($plugin, 'enqueue_script'));
    }

    /**
     * Constructor.
     */
    public function __construct()
    {
        // Setup your plugin object here
    }

    /**
     * Enqueues our kickass scripts and stylesheets.
     */
    public function enqueue_script()
    {
        // ...
    }

}
WP_Kickass_Plugin::register();
```

Идея №2 из: https://github.com/DevinVinson/WordPress-Plugin-Boilerplate

plugin-name.php

```php
/**
 * Begins execution of the plugin.
 *
 * @since    1.0.0
 */
function run_plugin_name() {
    $plugin = new Plugin_Name();
    $plugin->run();
}
run_plugin_name();
```

class-plugin-name.php

```php
/**
 * The core plugin class.
 *
 * This is used to define internationalization, admin-specific hooks, and
 * public-facing site hooks.
 */
class Plugin_Name {

    protected $loader;

  //...

    public function __construct() {
        //...
        $this->load_dependencies();
        $this->define_admin_hooks();
        $this->define_public_hooks();
    }

    private function load_dependencies() {
        //...
        /**
         * The class responsible for defining all actions that occur in the admin area.
         */
        require_once plugin_dir_path( dirname( __FILE__ ) ) . 'admin/class-plugin-name-admin.php';
        /**
         * The class responsible for defining all actions that occur in the public-facing
         * side of the site.
         */
        require_once plugin_dir_path( dirname( __FILE__ ) ) . 'public/class-plugin-name-public.php';
        $this->loader = new Plugin_Name_Loader();
    }

    private function define_public_hooks() {
        $plugin_public = new Plugin_Name_Public( $this->get_plugin_name(), $this->get_version() );
        $this->loader->add_action( 'wp_enqueue_scripts', $plugin_public, 'enqueue_styles' );
        $this->loader->add_action( 'wp_enqueue_scripts', $plugin_public, 'enqueue_scripts' );
    }

    public function run() {
        $this->loader->run();
    }
}
```

class-plugin-name-public.php

```php
class Plugin_Name_Public {

  private $plugin_name;

  public function __construct() {
      //...
    }
  /**
     * Register the stylesheets for the public-facing side of the site.
     *
     * @since    1.0.0
     */
  public function enqueue_styles() {

      wp_enqueue_style( $this->plugin_name, plugin_dir_url( __FILE__ ) . 'css/plugin-name-public.css', array(),
  }

  public function enqueue_scripts() {
    //...
  }
}
```

class-plugin-name-loader.php

```php
class Plugin_Name_Loader {

    protected $actions;

    protected $filters;

    public function __construct() {
        $this->actions = array();
        $this->filters = array();
    }
    /**
     * Add a new action to the collection to be registered with WordPress.
     *
     **/
    public function add_action( $hook, $component, $callback, $priority = 10, $accepted_args = 1 ) {
        $this->actions = $this->add( $this->actions, $hook, $component, $callback, $priority, $accepted_args );
    }

    /**
     * A utility function that is used to register the actions and hooks into a single
     * collection.
     */
    private function add( $hooks, $hook, $component, $callback, $priority, $accepted_args ) {
        $hooks[] = array(
            'hook'          => $hook,
            'component'     => $component,
            'callback'      => $callback,
            'priority'      => $priority,
            'accepted_args' => $accepted_args
        );
        return $hooks;
    }
    /**
     * Register the filters and actions with WordPress.
     *
     * @since    1.0.0
     */
    public function run() {
        foreach ( $this->actions as $hook ) {
            add_action( $hook['hook'], array( $hook['component'], $hook['callback'] ), $hook['priority'], $hook['accepted_args'] );
        }
    }
}
```

Мои контакты:
Телеграм: @katya_leurdo
Skype: katya.leurdo
Email: katya.leurdo@gmail.com

Спасибо за внимание!
Жду ваши вопросы