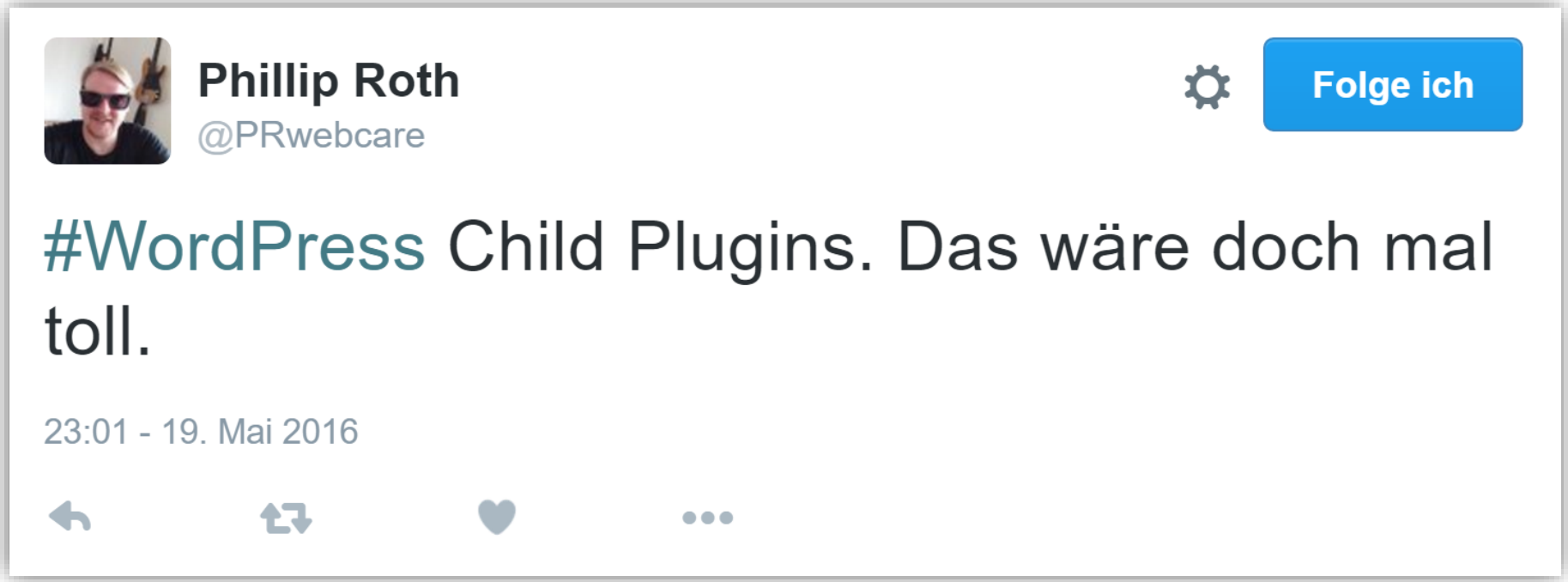
Child Plugins

WordCamp Frankfurt 2016

Wer ich bin

- Bernhard Kau
- Wahlberliner
- PHP-Entwickler
- WordPress-Plugin Hobby-Entwickler
- Organisator des WP Meetup Berlin
- Blogge auf kau-boys.de
- Twittere unter @2ndkauboy

Es begann alles mit einem Tweet



https://twitter.com/PRwebcare/status/733402190195253250

Was sind Child Plugins?

Ein Mythos ☺

Wozu überhaupt Child Plugins?

- Plugins erweitern
- Plugin an die eigenen Anforderungen anpassen
- Ohne den Code des Plugins anzufassen
- Um keine bösen Überraschungen bei Updates zu erleben
- Genau wie bei Child Themes

Anpassungsmöglichkeiten

1. Konstanten
2. Hooks (Actions und Filters)
3. Funktionen überschreiben
4. Templates überschreiben
5. Plugin forken

Konstanten in Plugins (Beispiel)

Code im Plugin:

```
if ( ! defined( 'WPCF7_LOAD_JS' ) ) {
   define( 'WPCF7_LOAD_JS', true );
}
```

Verwendung im Child Plugin:

```
define( 'WPCF7_LOAD_JS', false );
```

Konstanten in Plugins (Probleme)

- Sie sind konstant ☺
- Man kann sie also nicht nachträglich verändern
- Daher sollte den Einsatz von Konstanten vermeiden
- Besser durch einen Filter ersetzen

Hooks in Plugins

- Es gibt zwei Arten von Hooks:
  - Actions
  - Filters
- In Actions „tut man etwas“
  - Haben in der Regel „Seiteneffekte“
  - Haben normalerweise keinen Rückgabewert
- In Filtern „verändert man etwas“
  - Bekommen meistens eine oder mehrere Variablen übergeben
  - Geben die veränderte Eingabe zurück
  - Sollten keine Seiteneffekte haben

Actions in Plugins (Beispiel)

```php
public function save() {
   // ...

   if ( $post_id ) {
      // ...

      if ( $this->initial() ) {
         $this->id = $post_id;
         do_action( 'wpcf7_after_create', $this );
      } else {
         do_action( 'wpcf7_after_update', $this );
      }

      do_action( 'wpcf7_after_save', $this );
   }

   return $post_id;
}
```

Actions in Plugins (Beispiel)

```php
function child_plugin_wpcf7_after_save_do_things( WPCF7_ContactForm $contact_form ) {
   // Do something after the contact form was saved
}
add_action( 'wpcf7_after_save', 'child_plugin_wpcf7_after_save_do_things' );

function child_plugin_wpcf7_after_save_do_stuff( WPCF7_ContactForm $contact_form ) {
   // Do something else after the contact form was saved
}
add_action( 'wpcf7_after_save', 'child_plugin_wpcf7_after_save_do_stuff' );
```

Filters in Plugins (Beispiel)

```php
function allow_svg_mime_types( $mimes, $user ) {
    $mimes['svg'] = 'image/svg+xml';
    return $mimes;
}
add_filter( 'upload_mimes', 'allow_svg_mime_types', 10, 2 );
```

Filters in Plugins (Beispiel)

```php
function wpcf7_load_js() {
    return apply_filters( 'wpcf7_load_js', WPCF7_LOAD_JS );
}
```

```php
function child_plugin_deactivate_wpcf7_load_js( $load_js ) {
    return false;
}
add_action( 'wpcf7_load_js', 'child_plugin_deactivate_wpcf7_load_js' );
```

```php
add_action( 'wpcf7_load_js', '__return_false' );
```

Hooks in Plugins (Vorteile)

- Es können mehrere Funktionen für einen Hook registriert werden
- Werte können nachträglich geändert werden
- Hooks können auch wieder entfernt werden
- Innerhalb der Funktionen können Bedingungen geprüft werden
- Anpassungen können so einfacher ermöglicht werden

Funktionen überschreiben

```php
function hello_dolly_get_lyric() {
   /** These are the lyrics to Hello Dolly */
   $lyrics = "Hello, Dolly
Well, hello, Dolly
...";

   // Here we split it into lines
   $lyrics = explode( "\n", $lyrics );

   // And then randomly choose a line
   return wptexturize( $lyrics[ mt_rand( 0, count( $lyrics ) - 1 ) ] );
}

// This just echoes the chosen line, we'll position it later
function hello_dolly() {
   $chosen = hello_dolly_get_lyric();
   echo "<p id='dolly'>$chosen</p>";
}

// Now we set that function up to execute when the admin_notices action is called
add_action( 'admin_notices', 'hello_dolly' );
```

Funktionen überschreiben

```php
// remove original "Hello Dolly" output function and register the new one
function remove_hello_dolly_remove_action() {
   remove_action( 'admin_notices', 'hello_dolly' );
   add_action( 'admin_notices', 'we_are_the_champions' );
}
add_action( 'plugins_loaded', 'remove_hello_dolly_remove_action', 11 );

function we_are_the_champions_get_lyric() {
   /** These are the lyrics to We are the Champions */
   $lyrics = "I've paid my dues
Time after time
...";

   // …
}

// This just echoes the chosen line, we'll position it later
function we_are_the_champions() {
   $chosen = we_are_the_champions_get_lyric();
   echo "<p id='dolly'>$chosen</p>";
}
```

Funktionen überschreiben

Vorteile

- Man kann Plugins anpassen, die keine Hooks anbieten
- Ein Verändern des Originalcodes ist nicht notwendig
- Änderungen bleiben bei Updates erhalten

Nachteile

- Stichwort: Child Theme Dilemma
- Änderungen am Plugin müssen manuell übertragen werden
- Bei Updates kann es zu schweren Fehlern kommen
- Ein Anpassen ist nicht immer möglich oder kommt fast einer Neuimplementierung gleich

Templates überschreiben

Meine Empfehlungen für Templates in Plugins:

- Plugins sollten unabhängig vom Theme sein
- Sie sollten möglichst mit jedem Theme einsetzbar sein
- Das Mitliefern von Standardtemplates ist sinnvoll
- Solche Templates sollten nur die Struktur vorgeben
- Templates und sonstige Textausgaben sollten filterbar sein
- Suche nach Template: Child Theme -> Parent Theme -> Plugin

Templates überschreiben

```php
function my_custom_wc_locate_template( $template, $template_name, $template_path ) {

   $template_overwrite = dirname( __FILE__ ) .  '/templates/' . $template_name;

   if ( file_exists( $template_overwrite ) ) {
      $template = $template_overwrite;
   }

   return $template;
}

add_filter( 'woocommerce_locate_template', 'my_custom_wc_locate_template', 10, 3 );
```

Plugin forken

- Ist das Plugin nicht erweiterbar, kann sich ein Fork (Kopie) lohnen
- Hierzu das Plugin in der aktuellen Version in einem VCS speichern
- Einen Branch für die eigenen Anpassungen erstellen
- Den Pluginnamen ändern (sonst wird es per Update überschrieben)
- Änderungen aus dem Originalplugin bei Updates mergen
- Hierzu den „Development Log“ RSS-Feed abonnieren
- Eventuell einen Pull-Request auf das Originalplugin erstellen

Empfohlene Reihenfolge bei Anpassungen

1. Hooks, API, Template-Struktur und Konstanten nutzen
2. Im Supportforum um Anpassung bitten
3. Funktionen überschreiben
4. Das Plugin forken
5. Den Fork im Supportforum veröffentlichen und Merge vorschlagen
6. Das eigene Plugin veröffentlichen mit Verweis auf Originalplugin

Quellen

- https://kau-boys.de/2745/wordpress/child-plugins-konstanten-definieren-um-plugins-anzupassen
- http://torstenlandsiedel.de/2016/02/07/das-child-theme-dilemma/
- https://codex.wordpress.org/Plugin_API
- https://docs.woocommerce.com/document/template-structure/

Fragen?

Vielen Dank!