Action Hooks: Costruiamo un Plugin

Introduzione

All’interno di WordPress una caratteristica primaria delle Plugin API sono gli hook. Gli hook permettono di “agganciarsi” all’interno del flusso operativo dei codici all’interno del CMS e ne alterano il funzionamento senza apportare modifiche al *core* di WordPress, quindi anche aggiornando ad una nuova versione del CMS, non verrà persa nessun modifica.

I plugin non potrebbero funzionare senza hook in quanto non avrebbero altro modo per modificare le funzioni di WordPress. In WordPress abbiamo due tipologie di hook: action hook e filter hook, vediamoli.

Action Hook

Questa tipologia di hook permette di eseguire codice in uno specifico momento durante il processo di caricamento del flusso di codici del CMS. Agganciando una funzione ad un action hook, comunichiamoa WordPress che vogliamo “fare qualcosa” in quel preciso evento. La funzione:

do_action()

crea e lancia un hook di azione. Quando agganciamo un action all’hook, non chiamiamo la funzione direttamente. Vediamo un esempio:

do_action($tag, $args = '');

questa funzione accetta due parametri:

1. $tag: il nome o l’identificativo per l’action hook;

2. $args: il valore o i valori (separati da virgola) che passiamo per registrare le actions.

Esempio di funzione con parametri multipli:

```
do_action ( $tag, $arg_1, $arg_2, $arg_3 );
```

Un hook d'azione molto diffuso in WordPress è wp_head usato in particolare per aggiungere metadati SEO e l'Open Graph per i tag social.

Una Action è una funzione PHP o una classe che viene registrata all'interno di un hook di azione. Senza azioni aggiunte, l'hook non farà nulla, allora iniziamo a fargli fare qualcosa!

Creiamo un piccolo plugin

Quando costruiamo un plugin dobbiamo creare delle action personalizzate che eseguono task specifici, aggiungendole ad action hook. WordPress ci fornisce una *add_action()* helper function.

```
<?php
add_action(
      string $tag,
      callable $function_to_add,
      int $priority = 10,
      int $accepted_args = 1
);
```

la funzione sopra, accetta fino a 4 parametri:

1. $tag: il nome dell'action hook ed è il primo parametro per do_action();
2. $function_to_add: una funzione che viene eseguita quando l'action viene avviato;
3. $priority: stabilisce l'ordine secondo cui l'action viene eseguito, di default il valore è 10 e accetta anche valori negativi. Poniamo l'ipotesi che altri

plugin installati (o temi o stesso WordPress) nel nostro CMS potrebbero avere priorità più o meno del nostro, quindi prima di impostarla è bene avere una panoramica completa di cosa accade nel nostro WordPress;
4. $accepted_args: il numero di parametri da passare alla funzione di callback.

Dopo questa introduzione doverosa, iniziamo a costruire il nostro piccolo plugin che mostrerà un messaggio di base.

```
<?php
/** Plugin Name: Footer Message Developress
*                          Plugin                          URI:
https://www.developress.it/footer-message-developress
* Description: Mostriamo un messaggio nel footer dell sito web
* Author: Developress
* Author URI: https://www.developress.it
*/

add_action(   'wp_footer',   developress_footer_message',
PHP_INT_MAX );

function developress_footer_message() {
      esc_html_e('Benvenuto su Developress, segui il blog e
resta sempre ifnormato sul CMS WordPress', 'developress')
}
```

Cosa abbiamo fatto? Alla funzione add_action() abbiamo passato 3 parametri:

1. wp_footer che è l'action hook che comunica a WordPress a quale hook agganciarsi;
2. developress_footer_message è il nome della funzione da eseguire quando l'hook viene avviato;
3. PHP_INT_MAX è la priorità dell'action che abbiamo impostato al numero intero più grande possibile.

Si tratta di una semplice applicazione degli hook di azione ma serve comprendere il loro funzionamento, una volta che abbiamo chiara la logica è molto più semplice e naturale creare nuove

applicazioni molto più complesse di questa.