Funzioni degli Action Hook

Introduzione

Quando costruiamo un plugin, spesso utilizziamo soltanto le funzioni add_action() e do_action() per lavorare con gli hook. Ci sono molte altre funzioni utili messe a disposizione da WordPress. Vediamo le principali.

remove_action()

Questa funzione ci permette di rimuovere una action che è stata precedentemente agganciato ad un hook; a volte può essere utile rimuovere una action aggiunta da WordPress oppure da altri plugin/temi. La funzione ci restituisce *true* se l'action è stata correttamente rimossa e *false* se l'action non è stata registrata o non può essere rimossa.

```
<?php
remove_action( string $tag, callable $function_to_remove, int $priority = 10);
?>
```

La funzione accetta 3 parametri:

1. $tag: il nome dell'hook che vogliamo rimuovere
2. $function_to_remove: il nome dell'azione richiamabile da rimuovere dall'action hook
3. $priority: la priorità dell'action da rimuovere

remove_all_action()

Questa funzione è meno utilizzata, solitamente si utilizza per fare dei test oppure per creare un plugin di debugging e permette di rimuovere tutte le action per un determinato hook oppure tutte le action che hanno una determina priorità in un

hook.

```
<?php
remove_all_action( string $tag, int | bool $priority = false
);
```

Il parametro $tag dovrà essere il nome dell'action hook dal quale rimuovere tutte le action. Impostando la priorità ad un determinato numero, verranno rimosse soltanto le action con questa priorità. Se non impostiamo la priorità, verranno rimosse tutte le relative action, a prescindere dalla priorità.

Se abbiamo necessità di eliminare tutte le action dall'hook wp_head perché, ad esempio, stiamo riscontrando conflitti con i CSS/JS o metatag, o in generale dobbiamo eliminare qualunque codice che viene aggiunto nella sezione <head>, possiamo farlo in questo modo:

```
<?php
remove_all_action('wp_head');
```

con la funzione sopra stiamo eliminando tutte le action dall'hook che hanno priorità 1.

do_action_ref_array()

La funzione do_action_ref_array() è molto simile a do_action(), entrambe creano un hook ma la differenza sta nel numero di parametri che possiamo passare, infatti con do_action_ref_array() possiamo passare un array di argomenti.

```
<?php
do_action_ref_array( string $tag, array $arg );
```

has_action()

Si tratta di un condizionale che ci restituisce *true* se l'action viene trovata oppure *false* se non sono state trovate

action. Il controllo funzionerà soltanto se l’action è stat precedentemente aggiunta.

```
<?php
has_action( string $tag, callable|bool $function_to_check = false );
```

anche per questa funzione, come abbiamo visto per le altre, il primo parametro è il nome dell’hook che vogliamo cercare. Il secondo parametro è opzionale e di default impostato su *false*. Se vogliamo controllare una funzione specifica o metodo agganciato all’hook, possiamo aggiungere il parametro “callable”. Facciamo un esempio:

```
<?php
if ( has_action('wp_footer' ) ) {
       echo '<p>Ci sono action registrate nel footer</p>';
} else {
echo '<p>Non ci sono action registrate nel footer</p>';
}
```

Possiamo anche fare un altro esempio per controllare se una specifica action è agganciata ad un hook. Vediamo come fare per controllare se l’action wp_print_footer_scripts è agganciata a wp_footer:

```
<?php
$priority = has:action( 'wp_footer', 'wp_print_footer_scripts' );

if ( false !== $priority ) {
       printf(
                  'wp_print_footer_scripts ha una priorita di %d',
                 absint ( $priority )
                );

}
```

did_action()

did_action() è una funzione condizionale come has_action(), che determina se un hook di azione è stato già eseguito.

```
<?php
did_action ( string $tag );
```

la funzione accetta un solo parametro $tag e restituisce un intero che corrisponde al numero di volte che l'hook di azione è stato avviato. Una delle possibili applicazioni è all'interno di un *if* PHP:

```
<?php
did_action( 'plugin_loaded' ) ) {
        // Faccio qualcosa, imposto una variabile ad esempio
}
```

plugin_loaded è un hook che viene eseguito una volta caricato un singolo plug-in attivato.

Conclusioni

Le funzioni analizzate in questo articolo non sono tutte quelle disponibili ma l'obiettivo è stato quello di fornire una breve guida con gli elementi principali e spiegare la logica di programmazione degli hook e come gestirli. Se sei interessato puoi leggere gli articoli correlati oppure sfogliare la sezione plugin o temi.