Guida alla sviluppo di plugin per wordpress: il pannello di amministrazione (3/3)

Nel precedente articolo ci siamo lasciati con il plugin quasi completamente funzionante. Rimane solo da dare vita alle opzioni riguardanti il CSS (colore di background del box) e il javascript (effetto di chiusura del box).

Come rendere dinamico il CSS?

La prima idea che potrebbe venire é la seguente:

- prendere l'attuale foglio di stile;
- salvarlo come file .php;
- forzarlo come css tramite l'header;
- inserire le opzioni mediante la funzione *get_option*.

Proviamo:
Cambiamo il nome del file *style.css* in *style.css.php* e modifichiamolo in questo modo:

Successivamente modifichiamo il nome del file indicato nella funzione *wp_enqueue_style* del plugin con il nuovo nome del file (style.css.php) e proviamo il funzionamento. Non funziona, perché?

Il foglio di stile non viene eseguito nel runtime di WordPress, viene semplicemente inserito il link. Sarà poi il client a leggere il file come avviene sempre. Dunque, nel file *style.css.php* la funzione *get_option* non esiste (come pure nessun'altra funzione del core di WordPress), e questo causerà un fatal error.

Per avere a disposizione le funzioni del core di WordPress, dovremmo includere il file *wp-load.php* che si trova nella directory radice. Dunque in questo modo funzionerà:

Bene, ora che funziona ti posso dire che **questa procedura é da evitare fin tanto che é possibile**. Includere *wp-load.php* significa inizializzare WordPress (che non é quello che si dice una libellula). Se pensi che dovrà essere inizializzato alla richiesta della pagina e poi, una volta che la pagina arriva al client un'altra volta, é piuttosto irrazionale. Poi magari utilizziamo questa procedura anche per rendere dinamico il javascript e quindi inizializziamo un'altra volta WordPress. Se poi tutti quelli che sviluppano plugin utilizzassero questa procedura, alla fine la richiesta di una pagina corrisponderebbe ad un numero indefinito di inizializzazioni del core di WordPress. Ovviamente questo é inacettabile.

Vediamo allora le possibili soluzioni.

La prima potrebbe essere quella di mantenere nel foglio di stile (style.css) le parti statiche (noi non abbiamo molto, ma immagina dei fogli di stile lunghi e complessi). Le parti dinamiche le svilupperemo nel file principale del plugin.

Proviamo: ritorniamo al file *style.css* così come era prima di queste ultime modifiche. Eliminiamo l'attributo background-color: …..; e nel file del plugin ripristiniamo il corretto indirizzo in *wp_enqueue_style*.

Ora potremmo aggiungere una nuova funzione da passare al gancio head:

Oppure: se avessimo ad esempio una scelta limitata di colori per il background, ecco che potremmo definire le varie classi nel foglio di stile e poi passare la classe corrispondente direttamente al box. Una cosa del genere:

Questo unicamente con un numero limitato di colori.

Come rendere dinamico il javascript?

Ora dovremo trovare una soluzione per lo script di chiusura del box che non comprenda l'utilizzo di *wp-load.php*. Nel caso di javascript possiamo semplicemente valorizzare le variabili che ci servono e spedirle nell'header, per poi recuperarle nello script.
Iniziamo proprio da questo.

Come puoi vedere valorizziamo la variabile *effectType* che poi utilizzeremo nello script.
Facciamo poi stampare questo codice al gancio *wp_print_scripts*, ovvero prima che gli script vengano caricati.

Ora possiamo riprendere il nostro script per la chiusura del box e modificarlo in questo modo:

Come vedi utilizziamo una struttura di controllo switch per analizzare la variabile *effectType* e applicare così il giusto effetto di chiusura. Bene ora il plugin é terminato e funzionante!

Conclusione

In questa serie di tutorial abbiamo visto nel dettaglio gli elementi di base per **sviluppare un plugin per WordPress**. Grazie a questi pochi elementi sei ora in grado di fare (quasi) tutto.

Le funzioni che abbiamo visto possono essere utilizzare anche nel file *functions.php* di un tema in modo da aggiungere funzionalità o modificare il comportamento di WordPress quando il tema é attivo. O ancora: creare la pagina di amministrazione di un tema non é tanto diverso dal creare un plugin.

É possibile pensare un ulteriore approfondimento andando a sviscerare funzionalità più avanzate. Ad esempio se volessi realizzare un plugin per gestire una newsletter dovrei saper creare e gestire una nuova tabella del database di WordPress (non posso certo salvare gli iscritti come opzioni). Inoltre dovrei anche saper sviluppare un widget (che di base é un plugin) per metterci il form di iscrizione. Magari in futuro vedremo di approfondire ulteriormente.

E ora, cosa dovrei fare se volessi condividere con la commmunity il plugin? Come renderlo disponibile su repository ufficiale di WordPress?
Questo sarà il tema del prossimo e ultimo tutorial.

Capitoli di questa guida