Wordpress Saibal Edition: filosofia di un'ottimizzazione

Autore: **Lorenzo Forti** (lorenzo.forti@gmail.com)

Lorenzone.it, versione 6.0, è finalmente finito. Il lavoro è stato lungo ed ha riguardato soprattutto la personalizzazione di **Wordpress**.

Il prototipo in esercizio è stata modificato in alcune sezioni del *core*, molto nei *plugin*. Se questo, da una parte, ha permesso di andare incontro alle mie esigenze, dall'altra ha reso di fatto "unica" questa *release*, perdendo la possibilità di eseguire **update automatici** dei plugin.

E' uno svantaggio? E' un vantaggio? Dipende da quello che volete ottenere da questa piattaforma.

Per quel che mi riguarda presto attenzione non solo a ciò che scrivo ma anche all'ottimizzazione del codice e molte altre cose che un utente medio solitamente non considera, soprattutto perchè richiede un lavoro extra non di poco conto.

Una delle maggiori critiche che muovo ai **responsabili** di **Wordpress** è il non aver mai creato una valida struttura di controllo sul **codice di terze parti**. So bene che questo possa essere un compito molto dispendioso sia in termini economici che di tempo ma l'aver lasciato piena libertà a tutti gli sviluppatori (anche improvvisati) ha portato alla creazione di un enorme repository di plugin, molti dei quali sviluppati male, senza logica, non ottimizzati o comunque disomogenei tra loro per comportamento e funzionalità.

E' il mondo dell'*opensource* dirà qualcuno. No, è semplice **approssimazione** dico io. **Opensource** non significa che non si possa pretendere il rispetto delle regole solo perchè le persone partecipano (in molti casi) gratuitamente allo sviluppo della piattaforma.

Attenzione! Non sto assolutamente chiedendo di formare un'*elite* di programmatori per i plugin di **Wordpress**. Diamo a tutti la possibilità di partecipare, è nel mio carattere, ma prestiamo maggior attenzione a quello che viene incluso nel repository centrale.

Ad onor del vero le linee guida ci sarebbero anche ma sono di carattere troppo generalista e comunque non c'è nessun controllo successivo: ognuno è libero di non seguirle e programmare come vuole.

Il caso emblematico è la **disinstallazione** dei plugin: alcuni, giustamente aggiungo io, rimuovono le variabili che eventualmente hanno inserito nella tabella option (oppure in altri punti); la maggior parte dei moduli, invece, lascia dati inutili nel database.

Provare diversi plugin, magari per capire se fanno al caso nostro, significa ritrovarsi decine di opzioni inutili sparse per il db e, nel peggiore dei casi, tabelle vere e proprie che nessuno leverà mai.
Basterebbe richiedere, come requisito fondamentale, la possibilità di scelta per l'utente che deciderebbe in prima persona se rimuovere o meno i dati di configurazione del modulo.

Per queste ragioni avrei suddiviso l'archivio ufficiale in due *branche*: **certified** e **beta**.

Tutti i nuovi plugin verrebbero messi nel secondo ramo in attesa di essere installati e testati dagli utenti che certificherebbeo o meno il rispetto delle linee guida fondamentali e quindi il passaggio nella sezione "certified".

A tal proposito prendiamo in considerazione la modalità attuale con cui i plugin si integrano nell'amministrazione.

La maggior parte inserisce il proprio link nella sezione **Impostazioni**: gli utenti si aspettano questo; altri invece, per scelta del programmatore, finiscono sotto **Strumenti**; alcuni, infine, decidono di creare un box indipendente da inserire alla fine del menù laterale: è il caso di quei plugin molto complessi con molte pagine.

L'utente si trova nella condizione di cercare le impostazioni dei singoli moduli in tre luoghi diversi. Il mio primo compito è stato normalizzare questa situazione modificando le funzioni **add_menu** di ogni singolo plugin. Adesso si trovano tutti in una stessa sezione sotto la voce Impostazioni.

Altra modifica sostanziale è stata quella di spostare e raggruppare tutti i fogli di stile in un'unica cartella nel tema attivo.

Normalmente ogni plugin che abbia uno strato di presentazione porta con se un **file css** che aggiunge all'header una volta attivato.

I css, scritti da mani diverse, spesso vanno in contrasto tra loro o non sono ottimizzati. Più plugin si installano più css diversi vengono aggiunti al caricamento della pagina. Situazione che non ho mai apprezzato.

Dato che non cambio spesso la struttura del sito e quindi non levo/metto moduli ogni giorno, ho ritenuto opportuno modificare i file necessari e fare in modo che i fogli di stile fossero centralizzati. Adesso sono tutti nella cartella:

/wp-content/themes/MIO_TEMA/css/

Per il corretto funzionamento di Wordpress ho comunque lasciato un file **style.css** nella root del tema (contiene solo le informazioni sul layout).

Stesso discorso per i javascript: ogni plugin ha il suo che, normalmente, viene caricato nelle pagine anche quando non serve. Prendiamo ad esempio il caso di **Wp-ratings**, che permette di votare un post. Questo è il js che si porta dietro:

```
 /*  */
```

Perchè inserirlo in tutte le pagine? Capisco che dopo la prima chiamata i file esterni vadano in cache ma trovo ridondante includere un codice quando non serve.

Per risolvere è bastato inserire un IF nel file wp-postratings.php

```
if (is_single()) {
    add_action('wp_head', 'the_ratings_header');
 }
```

La spiegazione mi sembra semplice: dato che la votazione può avvenire solo nella pagina del post (**single.php**) faccio un banale controllo e se viene richiesto un articolo lancio anche la funzione che aggiunge i javascript necessari al funzionamento del plugin.

Qualcuno dirà che il mio è un caso isolato, che in un altro template la votazione poteva essere inserita anche nella pagine archive.php o page.php oppure search.php. In questo caso rispondo che un programmatore più attento avrebbe semplicemente dato all'utente la possibilità di scegliere dove inserire il codice attraverso una banale select multipla nelle impostazioni. Qualcosa come:

Inserisci il codice in:

- nella home
- nei post
- nelle pagine
- nei risultati della ricerca
- negli archivi

Wordpress infatti ci mette nativamente a disposizione delle funzioni condizionali per individuare velocemente le sezioni del sito:

`is_home(), is_page(), is_single(), is_archive() etc etc`

Perchè non usarle? Basta poco per mantenere tutte le funzionalità e strizzare l'occhio alle prestazioni e al codice.

Lo stesso discorso può essere fatto per quei plugin che caricano css propri. Basta capire in quali sezioni devono essere caricati e aggiungere il solito IF:

```
if (is_home() || is_page())  {
       wp_register_style('blabla','blabla.css',false,'1','all');
}
```

to be continued...