jQuery+PHP: come automatizzare gli inserimenti nel database?

Abbiamo già affrontato la questione dei form diverse volte; tra i vari articoli, abbiamo visto come creare un plugin di jQuery per la validazione di formulari e come realizzare un modulo contatti in php.

In questo articolo, invece, utilizzeremo **PHP e jQuery** per creare un script che, all'invio di un form, **crei la query sql** in base ai suoi campi; poi, una volta pronta, la invii alla nostra pagina in php che si **occuperà di inserirla nel database**.

Questo script nasce più che altro per colmare una esigenza riscontrata col tempo; infatti, mi è capitato spesso, che nei panelli di amministrazione fatti ex-novo, da un sito all'altro cambiassero solo alcune voci nei form per inserire le news piuttosto che in quelli per inserire i prodotti.
Ciò significava dover ogni volta aprire i file per modificare le pagine che si occupavano di inserire le query nel database, ed aggiungere qualche voce nuova, oppure toglierla.

Per questo motivo ho creato un **semplice e piccolissimo script** che si occupasse di generare una **query sql in maniera automatica** a seconda del modulo che si sta inviando.

Non mi soffermerò in modo particolare sulla validazione dei dati inviati tramite il modulo, poichè sono argomenti che sono stati spiegati molto bene in articoli precedenti, come in questo: Web design elements: come validare i form con jQuery?

Preparazione della struttura

Per avere un'idea più chiara di ciò che andremo a fare in questo articolo, puoi cliccare sul seguente link per vedere lo script in funzione.

Per realizzare la nostra auto-query abbiamo bisogno dei seguenti file:

- **savequery.php**
  il file che si occuperà di inserire la query nel database
- **site.js**
  il file che conterrà lo script per generare la query
- **index.html**
  la pagina statica nella quale risiederà il nostro form

Solitamente è necessario avere le idee ben chiare prima di procedere con la creazione di uno script/plugin, proviamo a scrivere due righe sul funzionamento del nostro:

1. Il nostro script JavaScript viene eseguito all'invio del form.
2. Vengono analizzati tutti gli input del form utilizzando un ciclo for.
3. Viene creata una Query SQL.
4. La Query SQL viene inviata ad un file php che si occuperà di eseguirla.
5. L'utente visualizza un messaggio con l'esito dell'operazione.

Come puoi vedere, il lavoro da eseguire non è poi così tanto; iniziamo quindi con il vedere il codice del nostro script JavaScript:

Inseriamo **tutto il codice all'interno del metodo ready()** di jQuery, il quale esegue il suo contenuto, una volta caricato tutto il DOM.

All'interno puoi notare il metodo submit() applicato al selettore $('form.autoquery'); questo significa che il suo contenuto verrà eseguito all'invio di ogni form avente la classe "autoquery".
All'interno del metodo submit() inseriremo tutto il contenuto per creare la nostra Query SQL , e prima della sua chiusura, inseriamo un *return false*, per evitare che il form venga inviato.

Andiamo avanti e vediamo il codice che andrà inserito all'interno del metodo submit().

Procediamo con dichiarare, all'interno della nostra function, una serie di variabili che utilizzeremo durante lo script; vediamole una ad una:

1. $elements
   contiene un'oggetto; il selettore di jQuery, filtra tutti gli **input, textarea e select** che si trovano all'interno del formulario (che all'interno della function viene individuato dalla

keyword this), viene inoltre utilizzato il selettore :not() che si occupa di fare un'ulteriore filtro: verranno esclusi dalla selezione gli input che hanno come type button e submit, ed inoltre verranno esclusi tutti gli input che hanno la classe exclude.

2. $result
   contiene l'oggetto risultato dal seguente selettore di jQuery: **$('div#result')**, e cioè, il div che ha come id "result" e che verrà utilizzato in seguito per mostrare l'esito della query.
3. db_table
   questa variabile contiene il valore dell'input che ha come nome "db_table", e rappresenta il nome della tabella nel database nella quale inserire la query.
   In questo caso utilizziamo il metodo val() di jQuery per ricavare il valore dell'input.
4. db_query
   variabile che contiene la sintassi SQL della query da eseguire; questa variabile verrà modificata in seguito, all'interno del ciclo, in modo da ottenere come risultato finale, qualcosa del genere: "Insert Into *nome_tabella* Values('*input_uno*', '*input_due*', '*input_tre*') "
5. totInputs
   contiene il numero totale di elementi trovati all'interno dell'oggetto $elements; per ricavarli utilizziamo la proprietà JavaScript length, la quale ci restituisce il numero totale di elementi all'interno dell'oggetto. (in questo caso sarebbe stato possibile utilizzare anche il metodo size() di jQuery, ottenendo lo stesso identico risultato)

Una volta definite le variabile che utilizzeremo nel nostro script, possiamo procedere con il ciclo degli elementi, vediamo di seguito il codice utilizzato:

Nel precedente codice puoi notare che viene applicato il metodo each() di jQuery al selettore che abbiamo prima definito: $elements.
Il metodo each ci permettere di **effettuare un ciclo degli elementi all'interno dell'oggetto $elements**; come parametro della sua function, abbiamo specificato "i", **che indica il numero dell'attuale loop**, partendo da 0, **che viene aggiornato ad ogni esecuzione della funzione.**
All'interno del metodo each() la *keyword this* (che funge da "puntatore") viene attribuito all'elemento attuale.

Anche questa volta abbiamo definito due variabili, vediamole:

1. **$this**
   questa variabile è semplicemente un abbreviazione; contiene l'oggetto jQuery frutto del seguente selettore: $(this)
2. **val**
   contiene il valore dell'elemento attuale ricavato dal *metodo val()*; puoi inoltre notare che al valore restituito da *val()* viene applicato il metodo replace() di JavaScript.
   Il **primo valore passato** al *metodo replace()* è una *regexp* (in sostanza seleziona tutti gli apostrofi all'interno della stringa), mentre il secondo valore è quello che verrà utilizzato per

sostituire **tutte le occorrenze trovate dalla regexp**

Le espressioni regolari sono un argomento abbastanza complicato, io ho semplificato il tutto in poche righe, per chi volesse approffondire l'argomento, lascio il seguente link: http://www.regular-expressions.info/javascript.html

Magari ti starai chiedendo per quale motivo stiamo facendo un replace di tutti gli apostrofi presenti all'interno del valore dell'input, il motivo è il seguente:

Supponiamo di avere la seguente stringa pronta da passare al nostro script PHP:

Questa stringa, passata tramite il metodo POST al nostro script PHP viene così transformata:

A questo punto dovremmo utilizzare la funzione stripslashes() di PHP, la quale si occupa di togliere dalla stringa passata come argomento, tutti i backslash. Il risultato, quindi, sarebbe simile al seguente:

A questo punto potrebbe essere tutto a posto, ma rimane un piccolo problema ancora: se l'utente inserisce un'apostrofo all'interno di una stringa, anche da questo verrebbe tolto il backslash, e di conseguenza, eseguire la query con un apostrofo di troppo darebbe un errore di sintassi.

Per questo motivo, tramite il metodo replace() di JavaScript, sostituiamo tutti gli apostrofi (solo quelli inseriti dall'utente) con la stringa "|||".

***NB**: L'utilizzo delle tre Barre Verticali seguite da un apostrofo è una scelta personale, si potrebbe sostituire gli apostrofi con qualsiasi cosa.*

Una volta eseguito il replace sui valori inseriti dall'utente, la query finale dovrebbe essere simile a questa:

Una volta passata la stringa tramite POST allo script PHP, possiamo passarla come parametro alla funzione PHP stripslashes(), ed il suo risultato lo utilizziamo nella la funzione str_replace() di PHP nel seguente modo:

La funzione str_replace() funziona di base con 3 parametri : La stringa da cercare, la stringa da utilizzare per sostituirla e la stringa originale nella quale cercare.
Noi come stringa da cercare passiamo "|||", la sostituiamo con "\'", e quindi con l'apostrofo ed il suo

backslash davanti, e la stringa nella quale cercare è il risultato di stripslashes(*stringa_ottenuta_tramite_post*).

Il risultato finale è il seguente:

La variabile $query contiene la sinstassi corretta per inserire una nuova riga nel nostro database.

nb: la funzione str_replace() accetta anche un 4° paramentro; quest'ultimo è un Integer che indica il numero massimo di occorrenze prima che la funzione si fermi.

Continuiamo con il nostro JavaScript

Ora che hai capito il perchè del metodo replace, possiamo procedere con il resto del nostro script in JavaScript.

Dopo la dichiarazione delle due variabili puoi notare un controllo tramite if/else:

In questo passaggio, quello che viene fatto è accodare alla query (definita all'interno della variabile dbQuery) il valore dell'input corrente all'interno del ciclo.

Esaminiamo il seguente codice:

Supponiamo di avere 5 input, e di ciclarli; nel metodo each() di jQuery l'indice (rappresentato dalla variabile i) è un numerico che aumenta di uno ogni volta che si ripete il ciclo e che parte però da 0.

Quando il nostro indice è uguale ad il totale degli inputs, meno 1, vuol dire che siamo arrivati all'ultimo elemento da ciclare.

Questo controllo viene effettuato per capire se dobbiamo inserire il valore dell'input con una virgola, oppure, essendo arrivati alla fine della query, dobbiamo ometterla.

Una volta eseguito il ciclo for, utilizzando il metodo each() di jQuery, dobbiamo accodare un nuovo valore alla Query SQL: la parentesi di chiusura.

La query è pronta, ora lasciamo fare a PHP

La nostra **query è pronta per essere passata allo script PHP**, il quale si occuperà di inserirla nel database (dopo aver effettuato le due operazioni che abbiamo visto prima).

Per farlo utilizzeremo il metodo $.post() di jQuery, il quale s**i occupa di inviare dei valori** ad un'altra pagina **tramite HTTP POST**, il tutto in maniera asincrona.

Vediamo di seguito il codice per effettuare questa operazione:

Il primo parametro rappresenta la pagina di destinazione del HTTP POST, nel nostro caso è './savequery.php'.
Il secondo parametro è invece un oggetto JavaScript il quale contiene una serie di proprietà/valori; la proprietà rappresenta il nome che poi ricaveremo tramite l'array $_POST di PHP.

Come puoi vedere, i valori che passiamo sono due: ajxInsert e query; il primo contiene una stringa di testo mentre il secondo contiene la nostra Query SQL.

Il terzo parametro è invece la funzione di callback, che viene chiamata quando il metodo riceve una risposta dalla pagina alla quale inviamo i nostri valori.
Puoi notare, inoltre, che come parametro della funzione di callback, abbiamo inserito la variabile "r", quest'ultima contiene la risposta ottenuta dalla pagina di destinazione.

All'interno della funzione di callback facciamo un'ulteriore controllo: se la risposta della pagina è uguale ad "ok", allora inserisco un messaggio di successo all'interno del div#result ($result), tramite il metodo html() di jQuery, e subito dopo utilizziamo il metodo slideDown() per mostrare il div.

Il metodo html(), quando riceve una stringa di testo come parametro, si occupa di inserirla in tutti gli elementi trovati dal selettore di jQuery; invece il metodo slideDown() mostra un div non visibile con una piccola animazione di slide verso il basso.

Per quanto riguarda lo script JavaScript, questo è tutto. Rivediamolo ora al completo.

Andiamo avanti, e lasciamo finire il lavoro a PHP

In questo articolo, come già anticipato, non tratteremo in modo particolare la sicurezza dello script oppure la validazione, quindi il codice PHP è veramente molto semplice, vediamolo:

Come puoi notare, il codice viene eseguito solamente se $_POST['ajxInsert'] e $_POST['query'] non sono NULL; per controllarlo utilizziamo la funzione isset() di PHP.

Superato questo passaggio, la nostra Query SQL viene ripulita da tutti i backslash non neccessari. In seguito, se presenti, viene sostituita la stringa precedentemente inserita da noi (|||') con un

apostrofo ed un backslash, in modo che una volta eseguita la query, non ci siano errori di sintassi.

Fatto questo possiamo inserire i valori nel nostro database e stampare l'esito dell'operazione, in questo caso: "ok" se l'esito è positivo e "ko" se l'esito è negativo.

Conclusione ed ultime considerazioni

Se "messo in sicurezza" questo semplice script potrebbe farci risparmiare del tempo in alcune situazioni.
In che modo gestisci i forms nei tuoi pannelli di controllo ? E pensi che questo metodo potrebbe tornarti utile ?