Come utilizzare la classe Upload

Nel precedente tutorial abbiamo sviluppato la classe *Upload*, una classe astratta che per essere utilizzata deve essere estesa. Ricordo che il nostro obiettivo é quello di creare una semplice gallery. Al caricamento dell'immagine verranno create le thumbnails mentre sulle immagini a dimensione piena aggiungeremo un watermark, il risultato finale sarà questo, lo puoi provare.

Iniziamo a fare un po' di ordine. A questo punto dovremmo avere i seguenti files:

- *upload.php*: che contiene il form di upload e che abbiamo creato nel primo tutorial;
- *upload.class.php*: che contiene la nostra classe upload;
- *upload_process.php*: che abbiamo sviluppato nel primo tutorial e che ora riscriveremo.

Aggiungiamo anche due cartelle:

- *img*: nella quale salveremo le immagini;
- *thumb*: nella quale salveremo le thumbnails (questo procedimento lo vedremo nel prossimo tutorial).

Creiamo un database

Per gestire al meglio la nostra galleria di immagini, salveremo i dati relativi all'immagine caricata in un database. In particolare salveremo il nome del file e l'estensione. Quando lavoreremo sulla libreria GD capirai perché tengo separati questi due dati. Chiaramente potremmo anche mettere più campi. Ad esempio nel form di upload potremmo aggiungere un campo di testo per il titolo o per una descrizione e salvare anche questi dati. In questo esercizio ci limiteremo allo stretto necessario.
Creiamo dunque il database *immagini* ed al suo interno la tabella che abbiamo descritto.

Creiamo il file di connessione

Come sempre, per comodità, creiamo un file di connessione che chiameremo *db_config.php*

Estendere la classe upload

Apriamo ora il file *upload_process.php* e cancelliamone il contenuto. Possiamo iniziare ora ad includere *upload.class.php* e a dichiarare la nuova classe *Upload_file* come estensione della classe *Upload*.

Il metodo onAbort

Iniziamo a sviluppare il metodo *onAbort()* che viene eseguito nella classe genitore in caso di errori nel processo.
Quello che faremo sarà avvertire che vi sono stati uno o più errori, ed in seguito scorrere l'array *error* (che ereditiamo dalla classe genitore) che contiene appunto la descrizione di questi errori.

Il metodo onSuccess

Il metodo *onSuccess()*, che viene invocato qualora il processo di upload sia andato a buon fine, dovrà procedere a salvare il nome e l'estensione del file nel database.
Per farlo includeremo il file *db_config.php* e lanceremo la query, in questo modo:

Come vedi abbiamo anche aggiunto un breve messaggio di conferma. Ed avrai anche notato che abbiamo invocato (commentandolo) il metodo *createThumb()*. Questo metodo salverà la miniatura dell'immagine caricata nella cartella *thumb* e sarà oggetto del prossimo tutorial.

Passare i parametri e istanziare la classe

Come primo passo, creiamo un array con le opzioni che vogliamo passare.

Abbiamo definito la cartella di destinazione, le estensioni ammesse (jpg e png) e la dimensione massima del file (1Mb). L'attributo name del campo file (upload_name) è invece già quello di default.

Ed ora passiamo le opzioni al costruttore:

A questo punto, avendo implementato i due metodi astratti, abbiamo fatto il minimo per far funzionare il nostro processo. Ora possiamo provare a caricare delle immagini, che dovrebbero finire nella cartella *img* ed i loro dati dovrebbero essere scritti nel database.

Conclusione

In questo tutorial abbiamo visto come utilizzare la classe *Upload* implementando i metodi astratti secondo le necessità del caso. Inoltre per chi è alle prime armi nell'ambito della programmazione

ad oggetti, questo è un ottimo esempio dei vantaggi che questo paradigma ci offre. Possiamo infatti vedere quanto la classe *Upload* sia “finita”, nel senso che contiene già tutto quello che serve per gestire il processo di upload. Nell’implementazione andremo ad aggiungere solo quello che ci serve nel caso specifico.
Nel prossimo tutorial introdurremo la potente libreria GD, grazie alla quale implementeremo il metodo che, caricando l’immagine, creerà la sua miniatura.