Le basi dell'upload di files

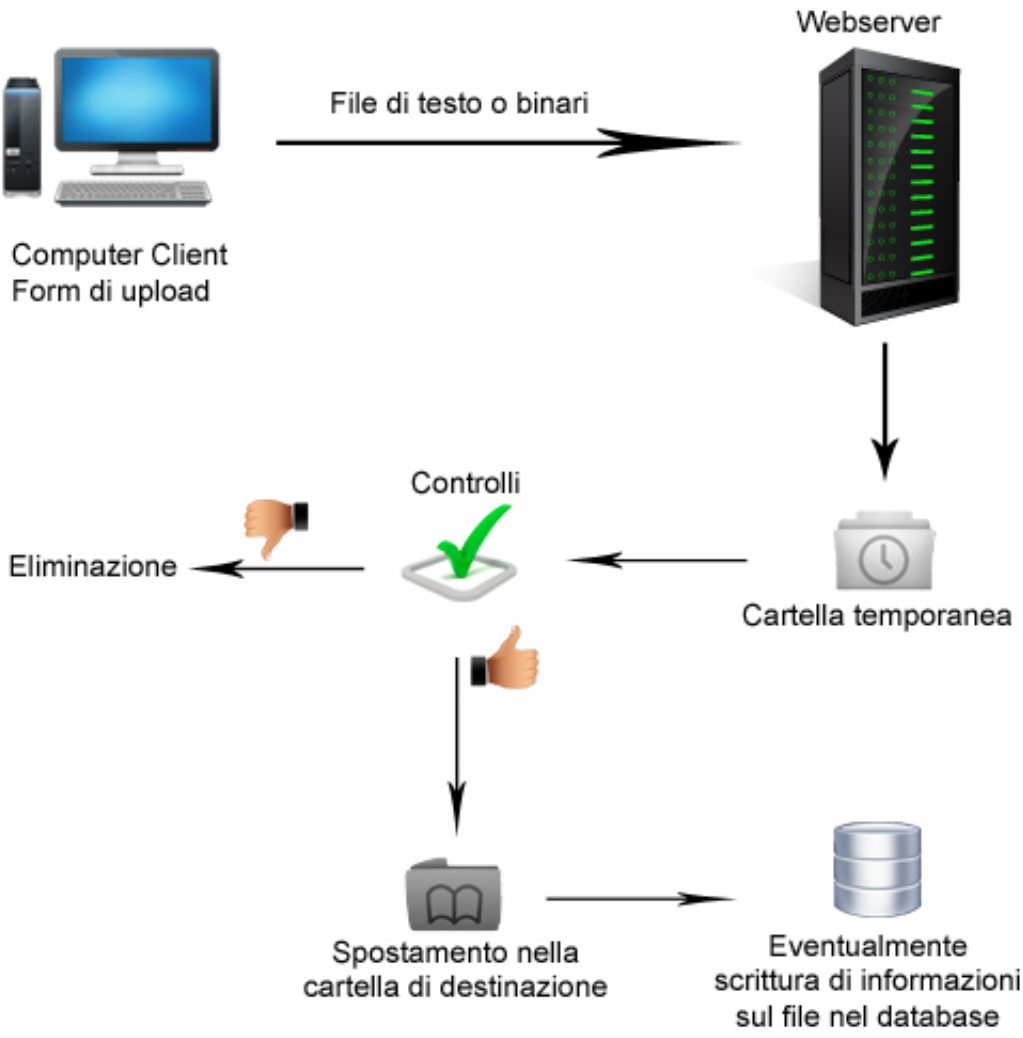
In questa serie di articoli tratteremo un aspetto particolare della programmazione per il web, ovvero **il processo di upload di files**. Si tratta di un'operazione non esente da insidie e che va quindi gestita con cura.
Questa serie è composta da cinque tutorial:

- Le basi dell'upload di file: dove vedremo come viene gestito il processo e quali funzioni utilizzare.
- Una classe avanzata per gestire gli upload: dove svilupperemo una classe robusta per la gestione dell'upload di file.
- Utilizzare la classe upload: dove vedremo come utilizzare la classe sviluppata nell'articolo precedente.
- Introduzione alla libreria GD: siccome il più delle volte utilizziamo il processo di upload per caricare immagini, introdurremo la libreria GD con la quale è possibile creare e manipolare immagini.
- Applicazioni della libreria GD: dove vedremo l'utilità di questa libreria per gestire le immagini caricate. Ad esempio la creazione "al volo" di thumbnail, o la posa di un watermark.

Il tutto ci porterà a sviluppare una semplice gallery. Pronto per iniziare?

Come avviene il processo di upload di un file?

Il processo di upload è rappresentato nel seguente schema:



- Il file viene scelto dall'utente tramite un form che deve avere delle caratteristiche precise, il file può essere indifferentemente un file di testo o un file binario.
- Il file viene caricato nella cartella temporanea del webserver.
- A questo punto possiamo svolgere tutti i controlli necessari.
- Se questo controlli non saranno superati il file sarà eliminato (i file scritti nella cartella temporanea al termine dell'operazione vengono cancellati automaticamente dello script).
- Se i controlli daranno esito positivo, il file sarà spostato nella cartella di destinazione e rinominato.
- E' possibile infine salvare nel database le informazioni del file che abbiamo caricato. L'idea di inserire direttamente il file binario nel database è da scartare. Si andrebbe infatti ad incidere notevolmente sulle prestazioni.

A questo punto possiamo iniziare dal form.

Come creare un form di upload?

Crea un nuovo file e salvalo come *upload.php*

Le principali ed indispensabili caratteristiche di un form di upload sono:

- l'attributo *enctype* deve essere *multipart/form-data*
- il metodo deve essere POST (ovviamente è impossibile passare un file con il metodo GET)

Dunque iniziamo a dichiarare il nostro form

A questo punto potremmo inserire un campo hidden specifico dei form di upload che serve (o servirebbe) a limitare la dimensione dei file caricati.

Con questo campo indichiamo che la dimensione massima del file caricato non deve superare i 1048576 bytes, ovvero 1MB.

Personalmente **ritengo questo passaggio superfluo ed addirittura fuorviante**. Tutto quello che viene dichiarato lato client è facilmente aggirabile. Disponiamo di diversi metodi sicuri lato server, quindi utilizzeremo quelli.

Passiamo dunque al campo che ci permetterà di scegliere il file da caricare. E' un campo di *input* con *type = "file"*.

È chiaramente possibile inserire anche tutti gli altri tipi di campo disponibili per i form. Il loro contenuto lo troveremo poi nell'array $_POST, mentre le informazioni sul file caricato le troveremo nell'array $_FILES. Ma questo lo vedremo più avanti.

Non ci resta che dichiarare il pulsante di submit ed il form sarà pronto per l'uso.

Ottenendo questo risultato

Gli amanti dell'estetica si mettano il cuore in pace. Per un motivo a me ignoto, non è possibile gestire tramite CSS il pulsante "sfoglia".

Come gestire il processo

Ora ci sposteremo nel vero cuore del processo di upload. Il form punta al file *upload_process.php*; crealo.

Come accennato precedentemente, nella pagina di destinazione disporremo di alcune informazioni sul file caricato all'interno dell'array $_FILES, vediamo quali sono queste informazioni:

- $_FILES['upload_name']['name'] contiene il nome originale del file caricato;
- $_FILES['upload_name']['tmp_name'] contiene il nome assegnato al file nella cartella temporanea;
- $_FILES[upload_name']['size'] contiene la dimensione in bytes del file caricato;
- $_FILES['upload_name']['type'] contiene il mime del file (ad esempio image/png);
- $_FILES['upload_name']['error'] contiene informazioni su eventuali errori del processo (vedremo nel prossimo tutorial);

In questo primo tutorial non faremo nessun tipo di controllo, ci limiteremo a spostare il file dalla cartella temporanea alla cartella di destinazione. Lo faremo con l'apposita funzione move_uploaded_file che passa due parametri:

- il nome del file temporaneo
- il percorso/nome del file di destinazione

Quindi il nostro *upload_process.php* potrebbe essere così

Scritto in questo modo, il file verrà prelevato dalla cartella temporanea e salvato nella cartella *img* con lo stesso nome che aveva sul computer che lo ha inviato.

Il nostro sistema di upload ora funziona, ma ad un livello estremamente basilare per non dire rudimentale. Per rendere questo sistema robusto, dobbiamo tenere conto di alcuni aspetti:

- Limitazione delle estensioni: E' meglio limitare le possibili estensioni dei file allo stretto necessario. Non procedendo a questa limitazione è possibile ad esempio caricare dei file PHP maligni e, conoscendo la cartella di destinazione, eseguirli con tutte le conseguenze del caso. E mi fermo qui.
- Limitazione della dimensione: Come abbiamo visto è inutile farlo dal form. Nel nostro script lato server abbiamo a disposizione il valore $_FILES['upload_name']['size'] grazie al quale potremmo fare la nostra verifica. E' possibile, come misura aggiuntiva, modificare i valori upload_max_filesize e/o post_max_size nel php.ini.
- Verificare la legittimità del file: Prima di spostare il file dalla cartella temporanea a quella di destinazione, é fortemente consigliato l'utilizzo della funzione is_uploaded_file alla quale

passeremo il nome del file. Questa funzione verifica che il file sia stato caricato tramite il protocollo HTTP POST e non magari grazie a qualche astuta peripezia. L'utilizzo di questa funzione ci mette al riparo da tutta una serie di scherzetti che possono costare caro.
- Rinominare il file: Non esiste nessun motivo per il quale si debba salvare il file con lo stesso nome che aveva sul computer client. L'operazione di rinomina del file con una stringa casuale è una procedura diffusissima (guarda ad esempio il nome delle tue immagini caricate su facebook) ed ha due ragioni principali:
    - Otterremo un **nome normalizzato**, quindi senza spazi o caratteri speciali che possono creare problemi nel salvataggio.
    - Se non usiamo questo metodo, dovremo controllare se il nome del file che abbiamo caricato non esista già nella cartella di destinazione. E' un'operazione che, se abbiamo molti file, può diventare molto onerosa.

Conclusione

In questo primo tutorial abbiamo visto le basi dell'upload di file. Abbiamo sviluppato un sistema molto rudimentale e piuttosto lacunoso, anche se funziona perfettamente.
Alla fine abbiamo visto quali aspetti bisognerebbe tenere conto per migliorare la sicurezza e la stabilità dell'applicazione.

Nel prossimo tutorial svilupperemo una classe robusta ed altamente configurabile che terrà conto di questi, ma anche di altri aspetti e che potrai utilizzare nei tuoi lavori.