Tecnologie web: progetto finale

Mattia Vinci

February 11, 2014

1 HowTo

- *URL:* http://labappweb.labinfo.educ.di.unito.it/studenti/st116761/mvc/
- *Credenziali admin:* admin pass
- *Credenziali utente:* test pass

Tutti gli utenti registrati hanno come password *'pass'*.

2 Nota sulla realizzazione

Cercando di non tralasciare nessuna specifica di progetto, ho tentato di concentrarmi maggiormente sugli argomenti non ancora trattati nei compiti precedenti, in particolare il pattern MVC e PHP OOP. Speciale attenzione e' stata dedicata alla sicurezza, sia lato client (tramite controlli dell'input ajax/javascript) che soprattutto lato server (tramite il sistema di autorizzazione, la validazione dell'input con pattern *whitelist* e l'uso dei prepared statements).
Altro obiettivo dell'applicazione e' quello di essere il piu' generale possibile, facilitando gestione, modifiche, e aggiunte di moduli. Purtroppo uno sbagliato calcolo della tempistica non mi ha permesso di rendere la grafica piu' piacevole e "professionale". Per lo stesso motivo i commenti all'interno del codice sono molto limitati.

3 Geografia dell'applicazione

I file sono raccolti all'interno della cartella `mvc/` al cui interno vi sono:

- `common/` librerie utili (validazione, form variabili globali)
- `controller/` insieme dei controller
- `img/` immagini usate nella grafica
- `j/` file javascript
- `model/` descrizione del modello
- `s/` fogli di stile
- `secret/` dati sensibili
- `usr_img/` cartella pubblica per l'upload di immagini
- `view/` insieme delle view
- `index.php` file principale che gestisce le richieste

Il file `index.php`, insieme a `model/init.php`, funge da multiplexer: ottiene una richiesta del tipo `index.php?req=/controller/me`
e la gestisce aprendo i file necessari. Si e' creato questo tipo di url avendo in mente la possibilita' di usare il modulo rewrite di apache per ottenere link del tipo `nome-sito.it/a/b/c/` che sembrino statici agli occhi dei motori di ricerca.
Controller e modelli sono gestiti ognuno da una classe principale (nella relativa cartella) che e' poi estesa da controller e modelli specifici.
Anche le view sono gestite dalla classe `view/view.php` che in base alle richieste del controller imposta alcune opzioni della view (file di javascript, messaggi, dati arbitrari ...) e richiama il file specifico.
Ogni controller ha un'omonima cartella all'interno di `view/`, nella quale risiedono le view relative ai metodi del controller.

4 Alcuni dettagli d'implementazione

4.1 Back-end

4.1.1 I modelli

Approssimativamente, vi e' un modello per ogni entita' presente nel database (stato, utente, luogo...)
Ognuno di questi modelli eredita la classe `entita` che fornisce metodi e variabili generali (gestione del database, dei commenti...)
Ho preferito creare anche dei modelli "manager", non strettamente necessari ma che rispecchiavano meglio la mia logica: questi modelli si occupano di gestire "dall'alto" le entita': ne contengono l'insieme completo, le creano e le rimuovono. Ad esempio, *utente_manager* crea, attiva, cancella e contiene il sommario di tutti gli utenti iscritti.
Tutti i modelli (a meno di qualche dimenticanza) formulano query al database mysql tramite l'uso dei prepared statement della classe php `mysqli`.

4.1.2 La sessione

La sessione e' rappresentata da un apposito modello, gestito dal controller principale `controller/controller.php` tramite il metodo `manage_ session(boolean $auth_ required)`
La sessione viene serializzata e salvata nella variabile globale `$_ SESS` all'indice `sess_data` ma anche rispecchiata nel database per essere cancellata (insieme ad altre eventualmente scadute) una volta fatto il logout.
La sessione ha una validita' di 30 minuti: se il login viene effettuato entro 30 minuti dall'ultimo accesso autorizzato la sessione nel database viene ripristinata, altrimenti eliminata e sostituita con una nuova.
Ad ogni inizio sessione il modello genera un numero pseudo-casuale che viene memorizzato nel database; questo seme viene poi salvato nella variabile di sessione in modo che non vengano mai a trovarsi lato client (se usati i cookie) le reali credenziali dell'utente, anche se sempre crittografate nel database l'algoritmo *md5*.

4.1.3 La ricerca

La classe `ricerca` definisce un modello abbastanza versatile per l'esecuzione sicura di query all'interno del database. Permette di impostare la tabella di ricerca, i campi interessati e i valori da ricercare, cosi' come l'ordinamento dei risultati.
Ove possibile l'input e' validato tramite *white list* (tabelle, campi, interi, formati fissi), altrimenti tramite metodi della classe astratta `regexp`.
Nel sito vero e proprio si e' pero' deciso di alleggerire l'interfaccia non permettendo la selezione del tipo di ordinamento nel momento della ricerca, ma solo nella pagina di visualizzazione dei risultati (tramite una libreria javascript).

4.1.4 La libreria form

Una libreria implementata tramite la classe `common/form.php` permette di creare form dinamicamente tramite php, facilitando la gestione di attributi quali eventi javascript a cui rispondere (ad esempio con validazione del campo), classi css, campi richiesti.

4.2 Front-end

4.2.1 Validazione input

La validazione dell'input avviene tramite javascript, spesso con l'aiuto di metodi del controller `ajax`.
Se l'input e' valido il campo del modulo ha bordi verdi, altrimenti ha bordi rossi e viene stampato uno specifico messaggio di errore.
I campi richiesti sono controllati tramite il semplice attributo `required` di html5.

4.2.2 I commenti

I commenti, che possono essere allegati a qualsiasi entita' (utenti, stati, luoghi, commenti) vengono visualizzati in modo ricorsivo: nella view vengono pre-caricati e stampati i commenti "radice". Una coppia di link, *more* e *less* (purtroppo esteticamente brutti per le ragioni gia' accennate) permettono all'utente di leggere i commenti di risposta.
Quando viene cliccato *more* la prima volta, una richiesta ajax scarica i commenti richiesti dal database. Con *less* i commenti vengono nascosti ma rimangono nella pagina, in modo da non eseguire inutili query al db in caso di un'eventuale ulteriore richiesta degli stessi commenti.
Il sistema e' ricorsivo, per cui e' stato facile gestire i colori tramite css, assegnando un colore diverso ad ogni "profondita'" di commento nella discussione.

4.2.3 Librerie esterne

In alcuni casi il codice si appoggia alla libreria *jQuery*; soprattutto per effetti estetici (es. fading di oggetti del DOM) e sperimentazioni di alternative a javascript canonico (es. richieste ajax, presenti in entrambi i modi nel progetto).
Il menu e' una modifica di uno trovato in rete. Nella pagina ricerca ho usato la librearia *sorttable* (http://www.kryogenix.org/code/browser

5 Funzionalita'

Gli utenti non autenticati vengono invitati a fare il login o a registrarsi per accedere alle sezioni del sito (ma non alla home page).

5.1 Registrazione

Tramite la voce *Registrazione* del menu *Community* ci si puo' registrare inserendo un nick (lettere, numeri, - e _), una password e un valido indirizzo e-mail.
Una volta registrati, per attivare l'account e' necessario cliccare su un link ricevuto per email (nella dimostrazione apparira' anche nella pagina successiva).

5.2 Utente

5.2.1 Area privata

L'utente puo' modificare i propri dati (tranne il nome utente) inserendo ogni volta la password.
Puo' inserire una biografia/messaggio e una foto.

5.2.2 Contenuti

Un utente puo' aggiungere luoghi, messaggi di stato e commenti.
I contenuti possono essere eliminati ogni volta che appaiono nel sito.
Tramite la voce *Archivio* l'utente puo' fare una ricerca fra i commenti, i luoghi e gli utenti.

5.2.3 Community

Gli utenti possono possono visualizzare i contenuti (ad esempio i commenti e i luoghi) ma non le informazioni riservate (sommario attivita', messaggi di stato, profilo) di altri utenti.
Per farlo, devono fare una richiesta di amicizia che viene notificata (tramite una voce aggiuntiva del menu) e deve essere confermata per garantire l'accesso all'amico.
L'utente che riceve una richiesta di amicizia puo' vedere una versione limitata del profilo dell'altro.

5.2.4 Admin

Essendo presenti gia' nel sito le liste di entita' (raggiungibili tramite le voci principali del menu) non vi e' una vera e propria sezione admin; l'admin pero', in ogni pagina del sito, possiede l'autorizzazione di tutti gli utenti, avendo accesso quindi alle funzionalita' di modifica e rimozione dei contenuti.
Puo' inoltre, tramite la sezione utente, rimuoverli.