Sviluppo Web con Rails - parte II

Impariamo ad usare lo scaffolding per una prima bozza dell'applicazione

Ora che abbiamo installato il framework Rails e creato un'applicazione (**$ rails nome_applicazione**) possiamo procedere alla stessa.

Accesso al database

Come visto nella precedente puntata e come e' facile immaginare, tutti I file di configurazione sono nella cartella *config*. Apriamo con un editor di testo (ad esempio emacs) il file *config/database.yml*. Quella strana estensione, .yml, e' quella usata dai file **YAML**. Si tratta di un linguaggio di markup (non a caso la sigla *Yet Another Markup Language*). Al suo interno troviamo le impostazioni che Rails usera' per accedere ai database. Vi chiederete cosa significano *development*, *test* e *production*... Sono i tre database che verranno utilizzati e rappresentano le tre fasi dello sviluppo dell'applicazione. Noi, in questa fase andremo ad utilizzare soltanto il database di sviluppo e quindi lasceremo inalterato il resto del file. Supponendo di usare **MySQL** (e di averlo gia' configurato), cio' che inseriremo nel nostro *config/database.yml* sara':

```
development:
adapter: mysql
database: myapp_test_db
username: db_user
password: password_sicura
```

A questo punto e' necessario creare il database...
mysql -u db_user -p password_sicura -qualcosa_boh_non_ricord...ehm...
Gia'... Non e' che tutti ricordino a memoria come creare al volo un database con MySQL, quindi lasciamo fare a Rails!

```
$ rake db:create
```

Semplice no? Ed e' anche comodo perche' e' un comando indipendente dal DBMS usato, quindi anche cambiando quest'ultimo non avremmo alcun comando nuovo da imparare.

E ora scaffold!

Parliamo ora dello scaffolding, qualcosa che fara' la gioia dei piu' pigri... Lo scaffolding consiste nella creazione automatizzata di una struttura di base su cui modellare l'applicazione. Essa fornisce modelli, controller e viste per 4 azioni fondamentali riunite sotto l'acronimo **CRUD**, *Create – Read – Update – Delete*, ovvero operazioni di creazione, lettura, aggiornamento e cancellazione dei dati.

Volendo, ad esempio, implementare un piccolo blog engine (o una parte di esso) potremmo sicuramente digitare:

```
$ script/generate scaffold Post message:text author:string
```

Non bisogna dimenticare pero' che lo scaffold, per quanto comodo, non genera un'applicazione completa, ma soltanto una base da cui partire. Lo scaffold generatocon il precedente comando fornisce delle semplici viste in htmlcon dei form e dei pulsanti, nulla piu'. A questo punto I piu' si staranno chiedendo come far partire l'applicazione e vedere queste viste autogenerate. Innanzitutto dobbiamo generare la tabella dei post all'interno del database:

```
$ rake db:migrate
```

Questo comando esegue una cosiddetta *migration*. Le migration sono script in ruby che eseguono operazioni di modifica del database. Le migration le troviamo in *db/migrate*. Per ogni modifica che vorremo eseguire sul database utilizzeremo (direttamente o indirettamente) delle migration. Il comando dato precedentemente esegue tutte le migration non ancora lanciate. A questo punto il database ha la tabella posts necessaria all'applicazione (notate che il nome della tabella e' il plurale del nome dell'entita'...), e possiamo avviare (se e' gia' attivo un dbms) il web server.

```
$ script/server
```

In questo modo viene avviato **webrick**, il web server di default di rails. Ne esiste un altro, scritto anch'esso in ruby e piu' performante: **mongrel**. Lo possiamo installare comodamente da gem, il gestore di pacchetti ruby

```
# gem install ruby
```

o ancora dal gestore di pacchetti della nostra distribuzione. Ora che abbiamo avviato il web server ci posizioniamo con un browser su **http://localhost:3000/posts** et voila'! Orribile, vero? Beh, lo scaffold, lo ribadiamo, e' solo una base da cui partire per sviluppare la propria applicazione. Le alternative al meccanismo di scaffolding proposto da rails non mancano e alcune delle piu' famose sono ActiveScaffold, Streamlined, Hobo, AutoAdmin ed ExtJS.
Provate ora ad eseguire il comando **tree** in *app/*. Vedrete che sia in *models*, sia in *controllers*, ma anche in *views* sono stati creati dei files. Ebbene si, lo scaffold ha provveduto alla creazione di un modello, di un controller e delle viste necessarie. E' ora di scrivere un po' di codice, in fondo programmare e' questo, non lasciare fare tutto ad una serie di script...

Nella prossima puntata vedremo come modificare modelli, controller e viste per dare un tocco di personalita' alla nostra applicazione...

Stay tuned!