

Ruby on Rails by a J2EE programmer

Ing. Francesco Cioffi (<http://www.fcioffi.net>)

10 giugno 2007

Sommario

Se dovessi spiegare ad un programmatore J2EE cosa è Rails gli direi di immaginare un progetto in Java basato su EJB e Struts e magari Tiles, il tutto con un terzo delle righe di codice e con solo un paio di file di configurazione. Poi ci penserei un po' su e gli direi che forse sono stato un po' esagerato, tenendo conto della complessità di Struts in Java, Rails è molto più immediato e non richiede quella miriade di righe di codice XML e librerie esterne necessarie a configurare Struts e gli EJB.

1 Introduzione

Se dovessi cercare di impressionare qualcuno con Ruby[1, 2], Rails[3, 4] è il framework che farei conoscere. Da programmatore J2EE[5] orientato alle applicazioni web, vedo in Ruby on Rails uno stile di programmazione pulito (grazie a Ruby) ed efficace (grazie a Rails). Ritrovo in Rails un'ottima produttività ottenuta semplicemente scrivendo il codice e non basandosi su Content Management System[11] (CMS), Portlet[12] o quanto altro prometta produttività nel mondo J2EE come anche Struts[7] + Tiles[8], gli Enterprise JavaBeans[6] (ejb). Poi se si aggiunge il semplice piacere di scrivere codice Ruby, sensazione che non provavo da tempo, penso che ogni buon programmatore debba almeno conoscere o valutare direttamente questo linguaggio.

2 J2EE vs. Rails

Uno sviluppatore J2EE con un minimo di esperienza e con po' di progetti alle spalle imposta il proprio progetto sul pattern Model View Controller[9] (MVC) magari basandosi su Struts e magari sul pattern Data Access Object[10] (DAO) a meno di usare gli EJB. Poi per gestire al meglio il layout potrebbe usare Tiles integrandolo in Struts avendo così la possibilità di gestire i layout grafici. Quindi abbiamo almeno un file di configurazione per impostare gli EJB, un file di configurazione per configurare i controller e le azioni, ed un file di configurazione per associare ad ogni vista il layout grafico adatto. Insomma per le semplici operazioni Create Read Update Delete (CRUD) dobbiamo mettere le mani in almeno tre file di configurazione (Struts, Tiles, EJB) e scriverci la logica del modello e di controllo ... però abbiamo del codice davvero pulito ed una bella applicazione robusta! Per ogni nuovo oggetto la procedura è sempre la stessa, ma perché perdere tanto tempo a fare qualcosa di così meccanico e banale? Il concetto di base del framework Rails è proprio questo!

3 Casi pratici

Per prima cosa è da notare che in Rails non abbiamo la necessità di importare nel nostro progetto alcun file di configurazione ma basterà scrivere a riga di comando:

```
rails nome_applicazione
```

oppure utilizzando un editor di sicuro familiare ai programmatori J2EE quale RadRails figlio di Eclipse,

Nuovo Progetto – Rails Project.

A questo punto il framework crea la struttura di directory necessaria per la nostra applicazione, ci mette a disposizione WebRick un server di sviluppo, imposta gli ambienti di sviluppo, testing e produzione.

Per aggiungere un oggetto con la logica di base è sufficiente scrivere:

```
rails script/generate scaffold X
```

ottenendo così un oggetto mappato da una tabella sul database. L'oggetto X quindi corrisponde alla tabella X quindi, con i metodi messi a disposizione dalla classe ActiveRecord di Rails, possiamo prelevare o memorizzare / aggiornare direttamente l'oggetto sul database chiamando un solo metodo della classe. Di default le proprietà dell'oggetto corrispondono ai nomi dei campi della corrispondente tabella sul database; tranne se espressamente indicato, il nome della tabella sarà il nome al plurale della classe e per le relazioni di 1 a molti e molti a molti basterà indicarne l'oggetto e la relazione con una riga come quella di seguito:

```
belongs_to :oggetto contenuto
```

all'interno del modello.

Ogni metodo del controller corrisponderà ad un'azione e di conseguenza ad una vista. Per esempio se vogliamo implementare l'azione Y basterà definire il metodo Y nel controller e scrivere una pagina di vista chiamata Y, ed il gioco è fatto. La vista si baserà sul layout standard dell'applicazione implementata in un solo file senza nessuna riga aggiuntiva in ogni pagina

Il giro appena realizzato in Rails in Struts comporta la modifica del file di configurazione "struts-config.xml" dove bisogna aggiungere alcune righe di codice XML:

```
<action
  attribute='bean'
  input='pagina_di_provenienza'
  name='bean_contenente_i_dati'
  path=\
    'path_di_accesso_al_controller'
  type='percorso_del_controller' >
  <forward name='A' \
    path='vista_nel_caso_A' />
  <forward name='B' \
    path='vista_nel_caso_B' />
</action>
```

e nel file "tiles-defs.xml", per ogni pagina:

```
<definition name='A' >
  <put name='menu' \
    value='pagina_di_sinistra' />
  <put name='body' \
    value='pagina_di_destra' />
</definition>
```

4 Conclusioni

Potrei continuare così ancora per molto ma poi finirei per scrivere un manuale di Rails e non ne ho le competenze, tanto meno questo l'obiettivo che mi sono prefissato. Spero solo di aver suscitato un po' di curiosità, tale da invogliarvi a "perdere" una mezza giornata del vostro tempo per dare uno sguardo a questo giovane e promettente approccio alla programmazione!

Per chi volesse approfondire mi sento di consigliare la lettura degli articoli di Gennaio e Febbraio 2007 pubblicati su Mokabyte[13] (una comunità Java :-)

Riferimenti bibliografici

- [1] <http://www.ruby-lang.org>, Marzo 2007
- [2] <http://www.ruby-it.org>, Marzo 2007
- [3] <http://www.rubyonrails.org>, Marzo 2007
- [4] <http://rails.it>, Marzo 2007
- [5] <http://www.java.sun.com>, Marzo 2007
- [6] <http://www.java.sun.com/products/ejb>, Marzo 2007
- [7] <http://struts.apache.org>, Marzo 2007
- [8] <http://struts.apache.org/1.x/struts-tiles>, Marzo 2007
- [9] <http://java.sun.com/blueprints/patterns/MVC-detailed.html>,
Marzo 2007
- [10] [http://java.sun.com/blueprints/corej2eepatterns/Patterns/
DataAccessObject.html](http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html), Marzo 2007
- [11] http://it.wikipedia.org/wiki/Content_management_system,
Marzo 2007
- [12] WebSphere Portal Development Team. "Portlet Development
Guide", IBM
- [13] <http://www.mokabyte.it>, Marzo 2007