Introduzione a Rails

Rails è un framework per lo sviluppo di applicazioni web basato sul pattern ***MVC***. Rails favorisce la metodologia di sviluppo detta **convention over configuration**, in altri termini il framework fa delle assunzioni su cosa vogliamo sviluppare e su come lo vogliamo sviluppare, piuttosto che costringerci a definire innumerevoli file di configurazione per descrivere il comportamento di ogni singolo meccanismo del framework.

Rails ha inoltre una visione del pattern MVC leggermente differente rispetto altri frameworks come ad esempio Struts. In particolare per quanto riguarda il MODEL lo strato del CONTROLLER. Infatti nel MODEL troveremo nella stragrande maggioranza dei casi delle classi ActiveRecord, invece per quanto riguarda lo strato del CONTROLLER, pur rimanendo un layer di connessione tra MODEL e VIEW, ospiterà gran parte della business-logic che vorremmo implementare nell'applicazione.

Componenti fondamentali

Rails viene distribuito in una librebria gem , che è costituita da questi componenti fondamentali:

1) ActionPack:
   (a) ActionController
   (b) ActionDispatch
   (d) ActionView

2) ActionMailer

3) ActiveModel

4) ActiveRecord

5) ActiveResource

6) ActiveSupport

In questa piccola guida cercherò di far capire il ruolo di questi componenti, sviluppando una serie di esempi, dedotti durante lo sviluppo di un'applicazione di social-networking (twitter-clone) seguendo una guida all'URL : http://www.railstutorial.org

MODEL

Nello strato del MODEL, vanno a finire tutte le classi del nostro dominio di business, le cosidette classi entity, mentre il più delle volte la business-logic viene eseguita nelle azioni del controllore. Nessuno comunque ci impedisce di creare un layer business-logic per i casi d'uso più complessi che coinvolgono diverse classi entity.

In ogni caso lo strato del MODEL è fortemente influenzato dal ***pattern Active Record*** e dalle ***Migrations***. Rails favorisce il progetto del dominio di business e del modello dei dati come un processo unico, supportato dalle Migrations, che ci permettono di definire lo schema dei dati senza conoscere DDL-SQL, e di gestire l'evoluzione dello schema senza ricorrere a istruzioni SQL, ma solo gestendo degli script Ruby.