Introduzione

Ruby on Rails è un framework che facilita lo sviluppo, la distribuzione e la manutenzione delle applicazioni web. Nei mesi successivi al suo primo rilascio, Rails è passato dall'essere un giocattolo sconosciuto a un fenomeno mondiale; ancora più importante, è diventato il framework di riferimento per un'ampia gamma di applicazioni del Web 2.0. Come mai?

Rails è un tipo a posto

Molti sviluppatori erano frustrati dalle tecnologie che utilizzavano per creare applicazioni web. A prescindere che avessero scelto Java, PHP o .NET, provavano un senso crescente di difficoltà nel loro lavoro. Poi arrivò Rails, e le cose diventarono più semplici.

Ma la semplicità non è sufficiente. Stiamo parlando di professionisti che scrivono siti web del mondo reale. Volevano ottenere la percezione che le applicazioni che stavano realizzando reggessero la prova del tempo, che fossero progettate e implementate con tecniche moderne e professionali; per questo si buttarono a capofitto in Rails, e scoprirono che non era solo un strumento per pasticciare nei siti.

Per esempio, tutte le applicazioni Rails sono implementate usando l'architettura *Model-View-Controller* (MVC). Gli sviluppatori Java sono abituati a framework come Tapestry e Struts, che si basano anch'essi su MVC, ma Rails va oltre: quando sviluppate in Rails, iniziate con un'applicazione funzionante, c'è un posto per ogni frammento di codice e tutte le parti dell'applicazione interagiscono in modo standard.

I programmatori professionisti scrivono test, e tutte le applicazioni Rails prevedono un supporto ai test. Man mano che aggiungete funzionalità al codice, Rails crea automaticamente degli stub di test specifici; il framework facilita la verifica delle applicazioni, e il risultato è che le applicazioni Rails sono tendenzialmente ben controllate e verificate.

Le applicazioni Rails sono scritte in Ruby, un linguaggio di scripting moderno e orientato agli oggetti. Ruby è conciso senza essere incomprensibile, e il codice che ne deriva permette di esprimere le idee in modo naturale e ordinato. La conseguenza è che i programmi sono facili da scrivere e, soprattutto, da leggere quando li si riprende in mano mesi dopo.

Rails porta Ruby ai limiti, estendendolo in modi insoliti che facilitano la vita a chi scrive i programmi, rendendoli più corti e più leggibili. Inoltre consente di eseguire all'interno della base di codice alcune operazioni che in genere avvengono in file di configurazione esterni, contribuendo ad avere più chiaro quanto sta accadendo. Il codice riportato di seguito definisce la classe modello di un progetto. Per ora non preoccupatevi dei dettagli; considerate solo quante informazioni vengono fornite in poche righe di codice:

```
class Project < ActiveRecord::Base
  belongs_to :portfolio
  has_one    :project_manager
  has_many   :milestones
  has_many   :deliverables, through: milestones
  validates  :name, :description, presence: true
  validates  :non_disclosure_agreement, acceptance: true
  validates  :short_name, uniqueness: true
end
```

Vi sono poi altri due presupposti filosofici che rendono il codice Rails breve e leggibile: il concetto di DRY e il prevalere della convenzione sulla configurazione. DRY è l'acronimo di *Don't Repeat Yourself*, ovvero "non ripetetevi". Ogni componente di un sistema dovrebbe essere espresso in un solo punto, e Rails raggiunge questo obiettivo utilizzando Ruby. In un'applicazione Rails vi imbatterete raramente una duplicazione: quello che dovete dire lo dite in un unico punto – spesso suggerito dalle convenzioni dell'architettura MVC – e poi proseguite. Per chi è abituato ad altri framework web, in cui un semplice intervento sullo schema può implicare decine di modifiche e anche più, questa è stata una rivelazione.

Anche il prevalere della convenzione sulla configurazione è fondamentale. Significa che Rails ha delle impostazioni predefinite per quasi ogni aspetto dell'assemblaggio di un'applicazione. Seguite le convenzioni e potrete scrivere un'applicazione Rails usando meno codice di quello usato da un'applicazione Java nella configurazione XML. Se poi avete la necessità di aggirare le convenzioni, Rails ve lo permette senza problemi.

Gli sviluppatori che arrivano a Rails scoprono anche dell'altro. Non solo questo framework sta al passo con gli standard web *de facto*, ma contribuisce a definirli. Inoltre facilita agli sviluppatori l'integrazione di funzionalità come le interfacce Ajax e REST nel codice perché ne prevede il supporto (se non sapete di cosa stiamo parlando, ve lo spiegheremo più avanti nel libro).

Gli sviluppatori in genere sono preoccupati anche dalla distribuzione. Rails consente di distribuire più release di un'applicazione su diversi server con un unico comando (e anche di richiamarle altrettanto facilmente qualora qualcosa si dimostrasse men che perfetto).

Rails è stato estratto da un'applicazione commerciale del mondo reale; il modo migliore per creare un framework è trovare i temi centrali di un'applicazione specifica e assemblarli in una base di codice generica. Quando sviluppate in Rails, partite da un'applicazione parzialmente o totalmente già disponibile.

Ma Rails ha qualcos'altro, qualcosa difficile da descrivere. In qualche modo, è un tipo a posto. Naturalmente dovrete fidarvi di noi finché non inizierete a scrivere qualche applicazione (cioè nei prossimi 45 minuti, più o meno). Ecco di cosa si occupa questo libro.

Rails è agile

Il titolo originale del libro è *Agile Web Development with Rails 4*. Eppure potreste rimanere stupiti dal non trovare paragrafi che trattano esplicitamente l'applicazione di pratiche agili al codice di Rails.
La ragione è semplice e al contempo raffinata. L'agilità è parte del tessuto di Rails.
I valori espressi nell'*Agile Manifesto* sono principalmente l'insieme di quattro "prevalenze" (`http://agilemanifesto.org/`. Dave Thomas è uno dei diciassette autori di questo documento).

- I singoli e le interazioni prevalgono sui processi e gli strumenti.
- Il software funzionante prevale sulla documentazione estesa.
- La collaborazione dei clienti prevale sulla negoziazione contrattuale.
- La reazione alle modifiche prevale sull'adesione a un piano.

Rails si basa tutto sui singoli e sulle interazioni. Non esistono set di strumenti pesanti, configurazioni complesse o processi elaborati. Ci sono solo piccoli gruppi di sviluppatori, i loro editor preferiti e segmenti di codice Ruby. Questo garantisce la trasparenza: tutto quanto gli sviluppatori fanno si riflette immediatamente su ciò che viene visto dal cliente. È un processo intrinsecamente interattivo.
Rails non stigmatizza la documentazione, tanto che rende facilissima la creazione di quella HTML per l'intera base di codice. Nondimeno, il processo di sviluppo di Rails non è guidato dai documenti; non troverete specifiche di 500 pagine al cuore del suo progetto, bensì un gruppo di utenti e sviluppatori che valutano insieme le loro esigenze e i modi possibili per soddisfarle. Vedrete soluzioni che cambiano man mano che queste figure acquisiscono familiarità con i problemi che cercano di risolvere, e un framework che rilascia un software operativo fin dalle prime fasi del ciclo di sviluppo. Potrà essere un software che ha ancora bisogno di qualche ritocco, ma consentirà comunque agli utenti di farsi un'idea di quello che si sta creando.
In questo modo, Rails incoraggia la collaborazione dei clienti. Quando questi constatano la rapidità con cui un progetto Rails reagisce a una modifica, iniziano ad aver fiducia nel fatto che il team sarà in grado di produrre quanto necessario, e non solo quanto richiesto. I confronti vengono sostituiti da sessioni "What if?" ("e se…?").
Tutto è legato all'idea di saper rispondere ai cambiamenti. Il modo deciso, quasi ossessivo, con cui Rails rispetta il principio DRY fa sì che ogni modifica alle applicazioni influisca sul codice molto meno di quanto accade in altri framework. E poiché le applicazioni Rails sono scritte in Ruby, dove i concetti possono essere espressi in modo preciso e conciso, le modifiche tendono a essere localizzate e facili da scrivere. L'enfasi sui test funzionali e sulle unità, oltre al supporto per le correzioni e gli stub durante i test, fornisce agli sviluppatori la rete di sicurezza necessaria in fase di modifica. Disponendo di un buon set di test, ogni intervento diventa meno logorante.
Invece di cercare continuamente di collegare i processi di Rails ai principi dell'agilità, abbiamo deciso di lasciar parlare il framework. Mentre leggete i capitoli introduttivi, provate a immaginare di sviluppare un'applicazione web in quel modo, di lavorare fianco a fianco con i vostri clienti e di decidere insieme a loro le priorità e le soluzioni ai problemi. Arrivati ai concetti più avanzati, nella Parte III, considerate come la struttura

che soggiace a Rails vi consente di soddisfare le esigenze dei vostri clienti in modo più veloce e diretto.
Un ultimo aspetto riguardante l'agilità e Rails è che, per quanto non sia professionale dirlo, scrivere il codice diventa davvero divertente!

Destinatari del libro

Questo libro si rivolge ai programmatori che desiderano costruire e distribuire applicazioni basate sul Web. Include quindi coloro per i quali Rails (e magari anche Ruby) è una novità e coloro che invece hanno familiarità con i concetti di base ma vogliono raggiungere una comprensione più approfondita del framework.
Diamo per scontata una certa conoscenza dell'HTML, dei CSS (*Cascading Style Sheets*) e di JavaScript, in altre parole, la capacità di visualizzare i sorgenti sulle pagine web. Non dovrete essere degli esperti: al massimo vi chiederemo di copiare e incollare il materiale del libro, che è disponibile per il download.

Come leggere il libro

La prima parte del libro ha lo scopo di prepararvi. Alla fine di questa parte, vi avremo offerto un'introduzione a Ruby (il linguaggio) e una panoramica di Rails, avrete installato Ruby e Rails e avrete verificato l'installazione con un esempio semplice.
La parte successiva vi guida tra i concetti dietro le quinte di Rails tramite un esempio più esteso: costruiremo infatti uno store online. Non vi vengono spiegati i singoli componenti di Rails uno alla volta ("questo è il capitolo sui modelli", "questo è il capitolo sulle viste" e via di seguito); i componenti sono concepiti per lavorare insieme, e ogni capitolo di questa parte affronta un gruppo specifico di operazioni correlate che implica la collaborazione tra alcuni di essi.
La maggior parte delle persone sembra apprezzare la costruzione progressiva dell'applicazione mentre segue il libro. Se non volete digitare tutto il codice, potete scaricarlo all'indirizzo `http://pragprog.com/titles/rails4/source_code` (come archivio TAR compresso o file ZIP). Il download contiene set distinti di codice sorgente per Rails 3.0, Rails 3.1, Rails 3.2 e Rails 4.0. Poiché utilizzerete Rails 4.0, i file che vi occorrono sono quelli nella directory *rails40*. Leggete il file `README FIRST` per i dettagli.
Siate cauti se decidete di copiare i file direttamente dal download alla vostra applicazione, perché il server non saprà di dover prelevare queste modifiche se i timestamp sul file sono vecchi. Potete aggiornare i timestamp usando il comando `touch` su Mac OS X o Linux, oppure potete modificare il file e salvarlo. In alternativa, potete riavviare il server Rails.
La Parte III, "Rails in profondità", è una panoramica sull'intero ecosistema di Rails. Si inizia con le funzioni e i servizi di Rails con cui avete ormai acquisito familiarità, per poi passare a una serie di dipendenze chiave di cui il framework fa uso che contribuiscono direttamente alla funzionalità generale. Infine troverete una rassegna dei plug-in più diffusi che potenziano il framework e lo estendono, rendendolo un sistema aperto unico.
Nel libro vengono utilizzate una serie di convenzioni.

SUGGERIMENTI

Se è vero che è necessario conoscere Ruby per scrivere applicazioni Rails, siamo consapevoli che alcuni di coloro che leggeranno questo libro impareranno contemporaneamente sia Ruby sia Rails. Trovate una breve introduzione al linguaggio Ruby nel Capitolo 4.

Codice

La maggior parte degli snippet di codice nel libro proviene da esempi completi e funzionanti che potete scaricare.
Per orientarvi, se un listato si trova nei file di download, lo snippet è preceduto da un titolo, come nell'esempio che segue.

rails40/demo1/app/controllers/say_controller.rb

```
 class SayController < ApplicationController
➤  def hello
➤  end

   def goodbye
  end end
```

Il titolo contiene il percorso al codice nei file di download. Alcuni browser potrebbero interpretare per errore alcuni template come file HTML; in questo caso, visualizzate il sorgente della pagina per vedere il codice effettivo.
In quei casi in cui può non essere subito chiaro quali righe cambiare per modificare un file esistente, trovate delle frecce (➤) a sinistra che ve le indicano. Nel codice di esempio appena mostrato ne vedete due.

Parola di David

Ogni tanto incontrerete dei riquadri intitolati "Parola di David". Qui David Heinemeier Hansson vi offre alcuni scoop su determinati aspetti di Rails: concetti, trucchi, raccomandazioni e altro. Considerato che David è uno degli inventori di Rails, sono informazioni che non potete tralasciare se volete diventare professionisti di questo framework.

Domanda di Joe

Joe, il mitico sviluppatore, a volte interrompe con alcune domande relative al testo. Gli risponderemo man mano.

Questo libro non va considerato come un manuale su Rails. Secondo la nostra esperienza, i manuali di questo tipo non sono il modo migliore per insegnare le cose alle persone. Tramite l'esposizione e gli esempi, vi illustreremo la maggior parte dei moduli e molti dei loro metodi mostrandovi come questi componenti sono utilizzati e come si integrano.

Non troverete nemmeno centinaia di pagine con listati di API, e per un'ottima ragione: la documentazione vi viene fornita nel momento in cui installate Rails, e sicuramente sarà molto più aggiornata del materiale in questo libro. Se installate Rails usando RubyGems (cosa che vi consigliamo), avviate il server della documentazione gem (attraverso il comando `gem server`), e potrete accedere a tutte le API Rails puntando il browser all'indirizzo `http://localhost:8808`. (Trovate ulteriori informazioni sulla documentazione e le guide nel Paragrafo 18.1.)

Rails vi aiuta anche a produrre risposte che identificano chiaramente ogni errore rilevato, oltre a indizi che vi dicono non solo il punto in cui l'errore si è verificato, ma anche come arrivarci. Potete vedere un esempio nella Figura 10.3. Per ulteriori informazioni, consultate il Paragrafo 10.2, dove imparerete a inserire dichiarazioni di registro.

Se doveste rimanere bloccati, in Rete esistono numerose risorse che possono venirvi in aiuto. Oltre ai listati di codice citati, c'è un forum (`http://forums.pragprog.com/forums/148`) dove potete porre domande e condividere esperienze; una pagina di errata corrige (`http://www.pragprog.com/titles/rails4/errata`) dove potete segnalare i bug; un wiki (`http://www.pragprog.com/wikis/wiki/RailsPlayTime`) dove potete discutere gli esercizi presenti nel libro.

Queste risorse sono tutte condivise. Sentitevi liberi non solo di inviare domande e sottoporre problemi al forum e al wiki, ma anche di contribuire con suggerimenti ed eventuali risposte a problemi posti da altri.

Partiamo! Il primo passo sarà quello di installare Ruby e Rails e di verificare l'installazione con una semplice dimostrazione.