Capitolo 1

Installare Rails

Nella Parte I di questo libro, vi presenteremo il linguaggio Ruby e il framework Rails, ma non potrete andare troppo lontano finché non li installerete entrambi e verificherete che funzionino correttamente. Per avere Rails sul vostro sistema vi serve quanto segue.

- Un interprete Ruby: Rails è scritto in Ruby, e anche voi scriverete le vostre applicazioni con questo linguaggio. Rails 4.0 consiglia la versione Ruby 2.0.0, ma sarà operativo anche sulla 1.9.3. Non funzionerà invece sulle versioni di Ruby 1.8.7 o 1.9.2.
- Ruby on Rails: questo libro è stato scritto usando Rails 4.0 (nello specifico Rails 4.0.0).
- Un interprete JavaScript: sia Microsoft Windows sia Mac OS X prevedono interpreti JavaScript, e Rails utilizzerà la versione già presente nel vostro sistema. Con gli altri sistemi operativi, potreste dover installare un interprete JavaScript separato.
- Alcune librerie, a seconda del vostro sistema operativo.
- Un database: in questo libro utilizzeremo SQLite 3 e MySQL 5.5.

Questo è tutto quanto vi occorre per una macchina di sviluppo, a parte un editor, di cui parleremo successivamente. Tuttavia, se intendete sviluppare un'applicazione, dovrete installare anche un server web (come minimo) e del codice di supporto che consenta a Rails di funzionare in modo efficiente. Il Capitolo 16 è dedicato a questo argomento, quindi non ne parleremo ora.

Ora, dove trovare tutto quello che c'è da installare? Dipende dal vostro sistema operativo…

In questo capitolo

- **1.1 Installare su Windows**
- **1.2 Installare su Mac OS X**
- **1.3 Installare su Linux**
- **1.4 Scegliere una versione di Rails**
- **1.5 Impostare l'ambiente di sviluppo**
- **1.6 Rails e i database**

1.1 Installare su Windows

Il modo più semplice per installare Rails su Windows consiste nell'utilizzare il pacchetto RailsInstaller (`http://railsinstaller.org/`). Al momento della stesura di questo libro, la versione più recente di RailsInstaller era la 2.2.1, che include Ruby 1.9.3 e Rails 3.2. Finché non verrà rilasciata una nuova versione che supporta Rails 4.0.0 o Ruby 2.0, sentitevi liberi di utilizzare la versione 2.1 di RailsInstaller.

L'installazione iniziale è un gioco da ragazzi. Dopo aver scaricato il pacchetto, fate clic su *Run* e poi su *Next*. Selezionate *I accept all of the Licenses* (dopo averle lette attentamente!) e poi fate clic su *Next*, su *Install* e su *Finish*.

Si aprirà una finestra che vi inviterà a immettere il vostro nome e il vostro indirizzo e-mail; questi dati servono solo per impostare il sistema *git* per il controllo della versione. Per gli esercizi di questo libro, non preoccupatevi della chiave *ssh* che viene generata.

Chiudete la finestra e aprite un nuovo prompt dei comandi. In Windows 8, digitate `cmd` sulla schermata *Start* e premete Invio. Sulle versioni di Windows precedenti alla 8 selezionate *Start* > *Esegui*, digitate `cmd` e fate clic su *OK*.

Gli utenti di Windows 8 devono compiere un passaggio in più, ovvero installare node.js (`http://nodejs.org/download/`).

Al termine della procedura, chiudete la finestra del prompt e apritene una nuova perché le modifiche a `%PATH%` abbiano effetto. Verificate che l'installazione sia corretta con il comando `node -v`.

Se si verifica qualche problema, cercate aiuto nella pagina *Troubleshooting* sul sito di RubyInstaller (`https://github.com/oneclick/rubyinstaller/wiki/Troubleshooting`).

Se la versione di RailsInstaller che avete utilizzato ha installato la versione di Ruby 1.9.3 o superiore, non sarà necessario aggiornare a una versione di Ruby più recente. Passate al Paragrafo 1.4 per assicurarvi che la vostra versione di Rails coincida con quella descritta in questa edizione. Ci vediamo là.

1.2 Installare su Mac OS X

Poiché Mac OS X incorpora già Ruby 1.8.7, dovrete scaricare una nuova versione di Ruby che funzioni con Rails 4.0. Il modo più veloce per farlo è utilizzare RailsInstaller, che al momento della stesura di questo libro installa Ruby 1.9.3. Un altro sistema è utilizzare la versione di sviluppo più recente di RVM per installare Ruby 2.0.0. Ruby 2.0 è la versione raccomandata dal team di Ruby, ed è nettamente più veloce di Ruby 1.9.3, anche se per seguire questo libro possono essere usate l'una o l'altra. Qui descriveremo entrambi i metodi, scegliete voi quello che più vi si adatta.

Prima di iniziare, aprite la cartella *Utility* e trascinate l'applicazione Terminale sul dock: la utilizzerete sia durante l'installazione e sia più avanti come sviluppatori Rails.

Installare tramite RailsInstaller

Visitate RailsInstaller (`http://railsinstaller.org/`) e fate clic sul grosso pulsante verde *Mac OS X 10.7 & 10.8* (oppure *10.6* se questa è la vostra versione). Al termine del download, fate doppio clic sul file e decomprimetelo. Se utilizzate Mac OS X Lion o Mountain Lion, prima di fare clic sul file dell'applicazione che viene generato, tenete

premuto il tasto Ctrl (o fate clic con il tasto destro del mouse) e selezionate l'opzione *Apri*. Aprendo l'app in questo modo, avrete l'opportunità di installare un programma da uno sviluppatore che non è noto al Mac App Store. Da qui, selezionate la lingua, rispondete a un paio di domande (come il vostro nome, che verrà usato per configurare `git`), e l'installazione procederà.
Aprite l'applicazione Terminale e al prompt immettete il seguente comando:

```
$ ruby -v
```

Il risultato dovrebbe essere questo:

```
ruby 1.9.3p392 (2013-02-22 revision 39386) [x86_64-darwin11.4.0]
```

Ora aggiornate Rails alla versione utilizzata per questo libro:

```
$ gem install rails --version 4.0.0 --no-ri --no-rdoc
```

> **NOTA**
>
> Al momento della stesura del libro non esisteva una versione compatibile con Mac OS X 10.9 Mavericks.

Siete pronti! Saltate al Paragrafo 1.4 e unitevi agli utenti Windows.

Installare usando RVM

Per prima cosa, scaricate e installate i più recenti Command Line Tools for Xcode (gennaio 2013) per il vostro sistema operativo (OS X Lion o OS X Mountain Lion) usando il pannello delle preferenze *Downloads* di Xcode.
Aprite il Terminale e immettete il seguente comando per installare la versione di sviluppo di RVM:

```
$ curl -L https://get.rvm.io | bash -s stable
```

Seguite le note di aggiornamento nell'output di questo comando. Al termine delle istruzioni, potete passare a installare l'interprete Ruby:

```
$ rvm install 2.0.0 --autolibs=enable
```

Questo passaggio può durare un po', perché scarica, configura e compila gli eseguibili necessari. Una volta finito, usate (`use`) quell'ambiente e installate Rails (`install rails`):

```
$ rvm use 2.0.0
$ gem install rails --version 4.0.0 --no-ri --no-rdoc
```

A eccezione dell'istruzione `rvm use`, tutte quelle precedenti devono essere eseguite una volta soltanto. L'istruzione `rvm use` deve essere ripetuta ogni volta che aprite una finestra della shell. La parola chiave `use` è facoltativa, quindi potete abbreviare in `rvm 2.0.0`. Potete anche decidere di renderlo l'interprete Ruby predefinito per le nuove sessioni del Terminale con questo comando:

```
$ rvm --default 2.0.0
```

Verificate se l'installazione ha avuto successo con il seguente comando:

```
$ rails -v
```

In caso di problemi, consultate i suggerimenti nella sezione *Troubleshooting Your Install* sul sito di RVM (`https://rvm.io/rvm/install`).
E anche gli utenti OS X sono a posto. Vi aspettiamo al Paragrafo 1.4 insieme agli utenti Windows.

1.3 Installare su Linux

Partite dal sistema nativo di gestione dei pacchetti della vostra piattaforma, sia esso `apt-get`, `dpkg`, `portage`, `rpm`, `rug`, `synaptic`, `up2date` o `yum`.
Il primo passo consiste nell'installazione delle dipendenze necessarie. Le istruzioni riportate di seguito sono quelle per Ubuntu 13.04 (Raring Ringtail); se lavorate con un altro sistema operativo, potreste dover modificare sia il comando sia i nomi dei pacchetti:

```
$ sudo apt-get install apache2 curl git libmysqlclient-dev mysql-server nodejs
```

Vi verrà chiesta una password per il vostro server mysql; se non la inserite, vi verrà richiesta più volte. Se specificate una password, dovrete usarla quando creerete il database in Iteration K1 (nel Capitolo 16).
Sebbene il team di Rails raccomandi Ruby 2.0 per Rails 4.0, se volete utilizzare una versione di Ruby già installata nel sistema potete ricorrere alla 1.9.3; vi renderà operativi in fretta.
A partire da Ubuntu 12.04, potete installare Ruby 1.9.3 e Rails 4.0 con i seguenti comandi:

```
$ sudo apt-get install ruby1.9.3
$ sudo gem install rails --version 4.0.0 --no-ri --no-rdoc
```

Se funziona tutto, avrete terminato, e potrete passare al Paragrafo 1.4.
Molte persone preferiscono avere sulla propria macchina un'installazione di Ruby distinta che supporti l'applicazione, ragione per cui decidono di scaricare e costruire Ruby. Il modo più semplice per farlo è utilizzare RVM. L'installazione di RVM è descritta sul sito `https://rvm.io/rvm/install`; qui vi proponiamo una panoramica sui vari passaggi. Per iniziare, installate RVM:

```
$ curl -L https://get.rvm.io | bash -s stable
```

A seguire, selezionate la casella di controllo *Eseguire il comando come una shell di login nelle preferenze di Terminale*. Fate riferimento alla pagina *Integrating RVM with gnome-terminal* per le istruzioni (`https://rvm.io/integration/gnome-terminal/`).
Uscite dalla finestra di comando o dal Terminale e apritene una nuova. Il `.bash_login` verrà ricaricato.
Eseguite il comando riportato di seguito, che installa i prerequisiti specifici necessari per il vostro sistema operativo:

```
$ rvm requirements --autolibs=enable
```

Al termine, potete passare a installare l'interprete Ruby:

```
$ rvm install 2.0.0
```

Questo passaggio può durare un po', perché scarica, configura e compila gli eseguibili necessari. Una volta finito, usate (`use`) quell'ambiente e installate Rails (`install rails`):

```
$ rvm use 2.0.0
$ gem install rails --version 4.0.0 --no-ri --no-rdoc
```

A eccezione dell'istruzione `rvm use`, tutte quelle precedenti devono essere eseguite una volta soltanto. L'istruzione `rvm use` deve essere ripetuta ogni volta che aprite una finestra di shell. La parola chiave `use` è facoltativa, quindi potete abbreviare in `rvm 2.0.0`. Potete anche decidere di renderlo l'interprete Ruby predefinito per le nuove sessioni del Terminale con questo comando:

```
$ rvm --default 2.0.0
```

Verificate se l'installazione ha avuto successo con il seguente comando:

```
$ rails -v
```

In caso di problemi, consultate i suggerimenti nella sezione *Troubleshooting Your Install* sul sito di RVM (`https://rvm.io/rvm/install`).
A questo punto abbiamo trattato Windows, Mac OS X e Linux. Le istruzioni che seguono sono valide per tutti e tre i sistemi operativi.

1.4 Scegliere una versione di Rails

Le istruzioni riportate in precedenza vi hanno aiutato a installare la versione di Rails utilizzata negli esempi di questo libro. Tuttavia, può accadere che non vogliate eseguire quella versione, magari perché ne esiste una più recente con alcune correzioni o funzionalità nuove, o magari perché state sviluppando su una macchina ma volete distribuire su un'altra che contiene una versione di Rails su cui non avete controllo.
Se questo è il caso, dovrete essere consapevoli di un paio di cose. Per i principianti, potete trovare tutte le versioni di Rails installate usando il comando `gem`:

```
$ gem list --local rails
```

Potete anche verificare qual è la versione predefinita attraverso il comando `rails --version`. Il comando dovrebbe restituire 4.0.0; in caso contrario, immettete la versione di Rails preceduta e seguita da underscore prima di ogni parametro di qualsiasi comando `rails`. Per esempio:

```
$ rails _4.0.0_ --version
```

È una soluzione particolarmente comoda quando create una nuova applicazione, perché se lo fate con una versione specifica di Rails, poi dovrete continuare a utilizzare quella, anche se sul sistema vengono installate versioni più recenti, almeno finché non decidete che è venuto il momento di aggiornare. Per farlo, è sufficiente aggiornare il numero in `Gemfile` nella directory root della vostra applicazione ed eseguire l'installazione del bundle. Tratteremo questo comando nel dettaglio nel Paragrafo 24.3.

1.5 Impostare l'ambiente di sviluppo

La routine quotidiana di scrittura dei programmi Rails è piuttosto lineare. Ognuno lavora in modo diverso; questo è il nostro.

La riga di comando

Utilizziamo molto la riga di comando. Per quanto ci siano sempre più strumenti GUI che aiutano a generare e gestire un'applicazione Rails, riteniamo la riga di comando la soluzione più potente. Vale quindi la pena acquisire un po' di familiarità con essa nel vostro sistema operativo. Scoprite come usarla per modificare i comandi che digitate, come cercare e modificare quelli già immessi e come completare i nomi dei file e dei comandi che inserite.
Il completamento tramite il tasto Tab è uno standard nelle shell Unix, come Bash e zsh; vi basta digitare i primi caratteri di un nome di file e poi premere Tab perché la shell cerchi e completi il nome in base ai file che hanno una corrispondenza con quanto immesso.

Controllo della versione

Tutto il nostro lavoro rientra in un sistema di controllo della versione (attualmente Git). Per noi è fondamentale controllare ogni nuovo progetto Rails in Git quando lo creiamo e confermare le modifiche una volta superati i test (in genere più volte ogni ora).
Se lavorate a un progetto Rails con altre persone, potete pensare a un sistema di integrazione continua (CI). Ogni volta che qualcuno apporta delle modifiche, il sistema CI produce una copia nuova di zecca dell'applicazione e avvia tutti i test. È un modo semplice per assicurarsi il massimo dell'attenzione in caso di danni accidentali. Potete anche impostare il sistema CI perché i vostri clienti lo utilizzino per giocare con la versione più aggiornata della vostra applicazione in corso d'opera; questa trasparenza è un ottimo sistema per essere certi che il progetto non deragli.

Editor

Scriviamo i nostri programmi Rails attraverso un editor specifico. Negli anni ci siamo resi conto che editor diversi funzionano meglio con linguaggi e ambienti diversi; per esempio, Dave inizialmente ha scritto questo capitolo in Emacs perché ritiene che la sua modalità Filladapt sia ineguagliabile quando si tratta di formattare al meglio l'XML durante la digitazione. Sam ha aggiornato il capitolo usando Vim. Tuttavia, per molte persone né Emacs né Vim sono l'ideale per l'ambiente Rails. Per quanto la scelta dell'editor rimanga personale, vediamo alcuni suggerimenti per quando si tratta di cercarne uno specifico per Rails.

- Supporto dell'evidenziazione della sintassi di Ruby e HTML. Supporto ideale per i file `.erb` (un formato di file Rails che incorpora gli snippet Ruby in HTML).
- Supporto dell'indentazione e della reindentazione automatiche dei sorgenti Ruby. È qualcosa di più di una funzione estetica: un editor che indenta il programma mentre lo digitate è il modo migliore per individuare gli annidamenti errati nel codice. La capacità di reindentare è importante nel refactoring del codice e quando ne spostate delle parti (la capacità di TextMate di reindentare il codice quando lo incolla dagli *Appunti* è molto comoda).
- Supporto per l'inserimento di costrutti comuni di Ruby e Rails. Scriverete numerosi metodi brevi: se l'IDE è in grado di creare le strutture dei metodi con la pressione di un paio di tasti, voi potrete concentrarvi sul materiale più interessante.

- Buona navigazione tra i file. Come vedrete, le applicazioni di Rails si distribuiscono su più file; per esempio, un'applicazione Rails appena creata nasce con 46 file distribuiti in 34 directory. E questo ancora prima che scriviate qualcosa. Vi occorre quindi un ambiente che vi aiuti a spostarvi rapidamente tra i file. Aggiungerete una riga a un controller per caricare un valore, passerete alla vista per aggiungere una riga per visualizzarlo e poi al test per verificare di aver fatto tutto nel modo corretto. Un sistema come quello di Blocco note, dove vi servite della finestra di dialogo *Apri* per selezionare i file da modificare, non si presta bene allo scopo. Preferiamo una combinazione dell'albero dei file in un riquadro, una serie di scelte rapida tastiera per trovare uno o più file nell'albero in base al nome e alcuni espedienti predefiniti che ci aiutino a navigare, per esempio, tra un'azione di un controller e la vista corrispondente.
- Completamento dei nomi. In Rails i nomi tendono a essere lunghi. Un buon editor vi consente di digitare i primi caratteri e poi vi suggerisce il possibile completamento alla pressione di un tasto.

Non vi consiglieremo degli editor specifici perché, sinceramente, ne abbiamo provati pochi e sicuramente lasceremmo fuori quelli preferiti da qualcuno. Nondimeno, per aiutarvi a partire con qualcosa di più di Blocco note, ecco alcuni suggerimenti.

- TextMate era lo standard Mac OS X *de facto* per Ruby on Rails (`http://macromates.com/`).
- Sublime Text (`http://www.sublimetext.com/`) è un'alternativa multipiattaforma che qualcuno vede come successore di TextMate.
- Aptana Studio 3 (`http://www.aptana.com/products/studio3`) è un ambiente di sviluppo Rails integrato che gira in Eclipse, su Windows, Mac OS X e Linux. Inizialmente si chiamava RadRails, e nel 2006 ha vinto un premio come miglior strumento di sviluppo open source basato on Eclipse; Aptana è diventata la casa del progetto nel 2007.
- jEdit (`http://www.jedit.org/`) è un editor completo con supporto per Ruby. Ha un supporto molto esteso per i plug-in.
- Komodo (`http://www.activestate.com/komodo-ide`) è l'IDE di ActiveState per i linguaggi dinamici, Ruby compreso.
- RubyMine (`http://www.jetbrains.com/ruby/features/index.html`) è un IDE commerciale per Ruby ed è disponibile gratuitamente per i progetti educativi qualificati e open source. Gira su Windows, Mac OS X e Linux.
- Ruby for NetBeans (`http://plugins.netbeans.org/plugin/38549`) è un plug-in open source per il celebre IDE NetBeans.

Chiedete a qualche sviluppatore esperto che utilizza il vostro stesso sistema operativo con quale editor lavora. Dedicate una o due settimane a provare le varie alternative, prima di decidere.

Il desktop

Non vi diremo come organizzare il vostro desktop mentre lavorate con Rails, ma vi descriveremo com'è il nostro.

La maggior parte delle volte, scriviamo del codice, eseguiamo i test e giochiamo con l'applicazione in un browser. Sul nostro desktop principale, quindi, sono sempre aperte

una finestra dell'editor e una del browser. Poiché vogliamo tenere d'occhio il registro generato dall'applicazione, teniamo aperta anche una finestra del Terminale, dove utilizziamo `tail -f` per scorrere il contenuto del file di registro man mano che viene aggiornato. Per questa finestra scegliamo in genere un font molto piccolo, così che occupi meno spazio; se vediamo qualcosa d'interessante passarci davanti agli occhi, lo ingrandiamo per studiarlo. Ci occorre anche accedere alla documentazione delle API di Rails, che visualizziamo in un browser. In precedenza abbiamo parlato dell'uso del comando `gem server` per eseguire un server web locale che contiene la documentazione di Rails; il comando è comodo, ma purtroppo suddivide la documentazione in più alberi separati. Se siete online, visitate `http://api.rubyonrails.org/` per vedere tutta la documentazione di Rails assemblata in un unico punto.

Dov'è il mio IDE?

Se approdate a Ruby e a Rails provenendo da linguaggi come C# e Java, potreste farvi qualche domanda sugli IDE. Dopotutto, sappiamo tutti che è impossibile creare il codice per le applicazioni moderne senza almeno 100 MB di IDE che supportano ogni singola pressione di un tasto. Per gli illuminati, questo è il punto del libro in cui vi consigliamo di sedervi, magari sorretti ai lati da due pile di manuali di migliaia di pagine sui framework.

Potrebbe sorprendervi sapere che la maggior parte degli sviluppatori Rails non usa gli IDE completi per Ruby o Rails (sebbene alcuni ambienti vi si avvicinino), bensì ricorre ai semplici, vecchi e buoni editor. Questo non comporta grandi problemi. Con gli altri linguaggi meno espressivi, i programmatori si basano sugli IDE perché questi facciano la maggior parte del lavoro, perché gli IDE generano codice, assistono nella navigazione ed eseguono una compilazione incrementale per avvertire quanto prima degli errori.

Con Ruby, gran parte di questo supporto non è necessario. Editor come TextMate e BBEdit vi danno il 90 percento di quello che ottenete da un IDE, ma sono molto più leggeri. L'unica funzionalità utile di un IDE che manca a Ruby è il supporto al refactoring.

1.6 Rails e i database

Gli esempi in questo libro sono stati scritti usando SQLite 3 (versione 3.7.4 o 3.*x*). Se volete seguire il nostro codice, è quindi meglio che lo usiate anche voi; se però fate un'altra scelta, non sarà un problema. Potreste dover apportare qualche lieve modifica al codice, ma Rails elimina quasi del tutto dalle applicazioni l'SQL specifico per il database. Se volete connettervi a un database diverso da SQLite 3, sappiate che Rails funziona anche con DB2, MySQL, Oracle, Postgres, Firebird e SQL Server. Per tutti questi, dovrete installare un driver di database, una libreria che Rails potrà utilizzare per connettersi al vostro motore di database e utilizzarlo. Questo paragrafo contiene i collegamenti alle istruzioni per farlo.

Tutti i driver del database sono scritti in C e sono distribuiti perlopiù come sorgenti. Se non avete voglia di costruire un driver dai sorgenti, guardate con attenzione sul sito web del driver: a volta gli autori distribuiscono anche le versioni binarie.

Creare la vostra documentazione delle API di Rails

Se volete, potete creare la vostra versione locale della documentazione delle API di Rails. Digitate quanto segue al prompt dei comandi:

```
rails_apps> rails new dummy_app
rails_apps> cd dummy_app
dummy_app> rake doc:rails
```

L'ultimo passaggio richiede un po' più di tempo. Al termine, troverete la documentazione in un albero di directory che inizia con *doc/api*. Vi suggeriamo di spostare questa cartella sul desktop e poi di eliminare l'albero *dummy_app*.
Per visualizzare la documentazione, aprite `doc/api/index.html` con il browser.

Se non riuscite a trovare la versione binaria o comunque preferite partire dai sorgenti, vi occorrerà un ambiente di sviluppo sulla vostra macchina per costruire la libreria. In Windows questo vuol dire avere una copia di Visual C++; in Linux dovrete avere gcc e simili (che però saranno probabilmente già installati).
In OS X, dovrete installare gli strumenti di sviluppo, che sono integrati nel sistema operativo, ma non sono installati di default. Dovrete anche installare il driver del database nella versione corretta di Ruby. Se avete installato la vostra copia di Ruby, ignorando quella integrata, ricordate che dovrete avere questa versione di Ruby al primo posto nel vostro percorso quando costruite e installate il driver del database. Potete utilizzare il comando `which ruby` per essere certi di non eseguire Ruby da `/usr/bin`.
Quelli che seguono sono gli adapter di database disponibili e i collegamenti alle rispettive home page.

- DB2: `http://raa.ruby-lang.org/project/ruby-db2` o `http://rubyforge.org/projects/rubyibm`.
- Firebird: `http://rubyforge.org/projects/fireruby/`.
- MySQL: `http://www.tmtm.org/en/mysql/ruby/`.
- Oracle: `http://rubyforge.org/projects/ruby-oci8`.
- Postgres: `https://bitbucket.org/ged/ruby-pg/wiki/Home`.
- SQL Server: `https://github.com/rails-sqlserver`.
- SQLite: `https://github.com/luislavena/sqlite3-ruby`.

Gli adapter di MySQL e SQLite sono disponibili per il download come pacchetto RubyGems (rispettivamente `mysql2` e `sqlite3`).

Riepilogo

- Abbiamo installato (o aggiornato) il linguaggio Ruby.
- Abbiamo installato (o aggiornato) il framework Rails.
- Abbiamo installato (o aggiornato) i database SQLite3 e MySQL.
- Abbiamo selezionato un editor.

Ora che Rails è installato, utilizziamolo. È il momento di passare al prossimo capitolo, dove creeremo la nostra prima applicazione.