Configurare un ambiente di sviluppo completo per Ruby

Leonardo Miliani | 11 marzo 2009

Il Python è diffusissimo è documentatissimo, e chi si avvicina a questo linguaggio trova in rete tutte le guide ed i consigli che gli servono. Ciò non si può dire per il **Ruby**, linguaggio molto bello e potente che, però, gode nel mondo occidentale di una diffusione più limitata (pensate che in Giappone, patria del suo creatore, è più diffuso del Python).

E siccome a me personalmente le cose "di nicchia" piacciono sempre più delle cose di moda, ho perso un paio di giorni in rete cercando di documentarmi su come poter approntare un ambiente di sviluppo completo per Ruby in modo da poter offrire a chi si avvicina a questo linguaggio la lista di ciò che serve per essere operativi in brevissimo tempo.

- *Premessa: i passaggi che leggerete qui sotto sono principalmente per il sistema operativo GNU/Linux, nella fattispecie Ubuntu Intrepid Ibex 8.10…. ma, con piccole modifiche, possono essere adattati anche ad altre distribuzioni e/o sistemi operativi.*

Cominciamo con l'interprete ed alcuni programmi accessori. **Ruby** inteso come interprete del linguaggio in genere è già presente in qualunque distribuzione Linux, Ubuntu compreso, ma se così non fosse, utilizzate il vostro gestore di pacchetti per installarlo. Controllate inoltre di aver installati anche i seguenti pacchetti, che vi serviranno nel proseguo del vostro lavoro con Ruby:

```
sudo apt-get install rdoc1.8 ri1.8 irb1.8 ruby1.8-dev rubygems
```

Per controllare che l'installazione sia a posto basta dare da terminale il comando `ruby -v`. Se tutto è a posto, l'interprete risponderà con la versione installata, come nell'esempio qui sotto:

```
ruby 1.8.7 (2008-08-11 patchlevel 72) [i486-linux]
```

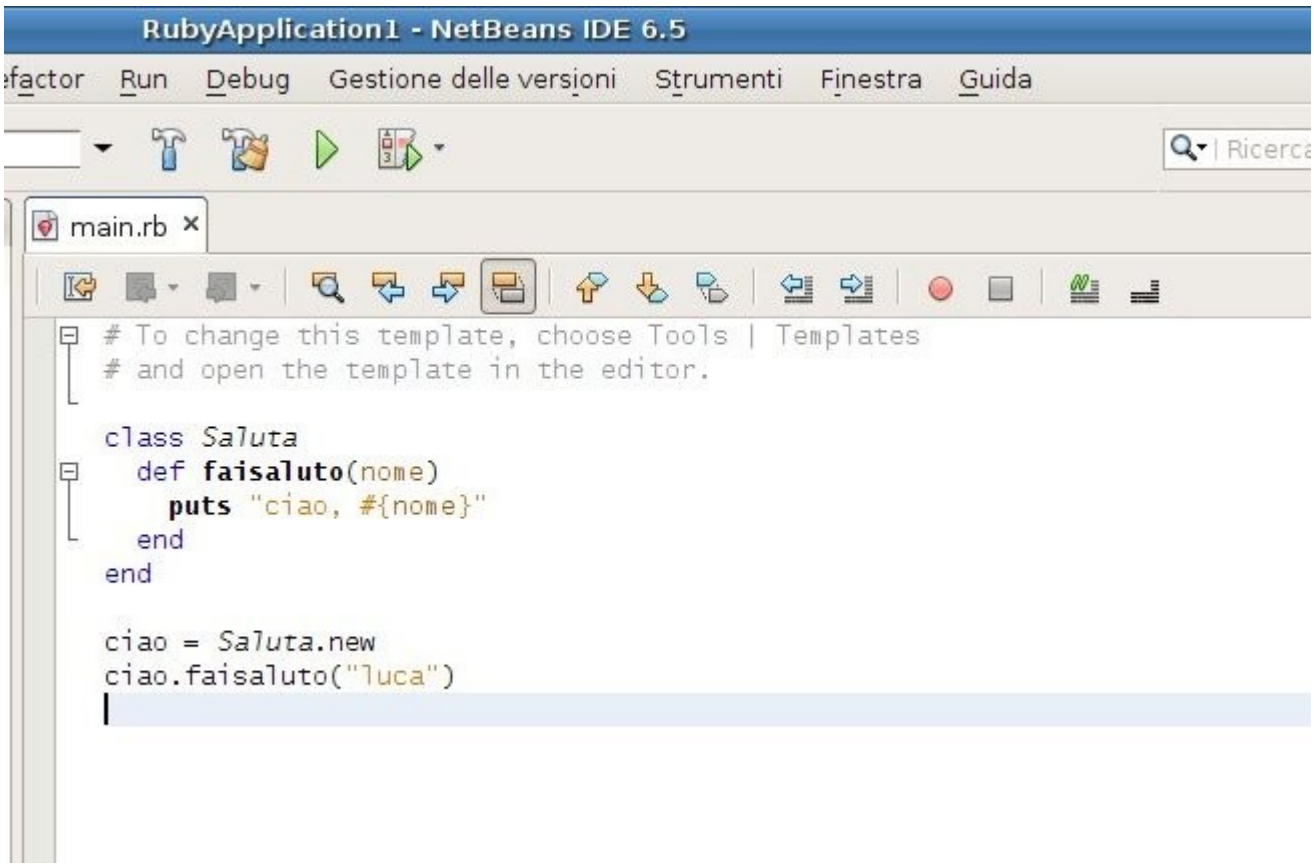
Adesso passiamo all'editor. Ne esistono a bizzeffe, a partire dai classimi emacs e vi. Ma io sono un tipo smemorato che non si ricorda mai la sintassi dei comandi per cui ho bisogno di un editor robusto con un buon auto-completamento del codice e voglio anche un editor "completo" e grafico. La mia scelta è perciò ricaduta su **NetBeans 6.5**, che offre un buon supporto a Ruby. Per scaricarlo andate sulla pagina multilingua di download e selezionate la versione con supporto per Ruby (è il 3° download, quello di 57 MB). Perché non affidarsi al pacchetto precompilato della propria distribuzione? Perché in Ubuntu è presente NetBeans 6.1 mentre io volevo non solo l'ultima versione ma anche quella con già il plugin per gestire Ruby.

Scaricato il file, eseguitelo da terminale con qualcosa di simile a questo (il comando può variare a seconda della versione):

```
sh ./netbeans-6.5-ml-ruby-linux.sh
```

Partirà l'installazione grafica che farà il resto. Durante il processo vi verrà chiesto se volete installare anche il web server GlassFish 3 Prelude: se avete intenzione di sviluppare applicazioni Ruby on Rails, fatelo, altrimenti potete risparmiare qualche decina di MB di spazio sul vostro disco.

Terminata l'installazione, troverete un'icona sulla scrivania del PC. Cliccateci per avviare NetBeans ed iniziate pure a scrivere le vostre prime righe di codice Ruby:



Editing di codice Ruby con NetBeans

Adesso un paio di “hack”. Di per sé NetBeans offre una versione preinstallata dell’interprete Ruby, JRuby (una versione per Java del linguaggio), comprese le sue proprie gemme. Ricordatevi di impostare tra le varie preferenze del programma l’uso dell’interprete installato sul computer. Inoltre, a causa delle impostazioni di Ubuntu, in cui l’utente *root* è disattivato ed al suo posto le operazioni di amministratore vengono svolte dall’utente normale mediante autenticazione con *sudo*, NetBeans non è in grado di installare nuove gemme sul sistema in uso. Questo perché lui chiede la password vera di root ma su Ubuntu, come detto, root è disabilitato di default. Le soluzioni sono 2: o si abilita tale utente (scelta sconsigliata) o bisogna dare il permesso di accesso alla cartella dove sono contenute le gemme anche al proprio utente (scelta consigliata). Per far ciò, basta dare i seguenti comandi:

```
cd /var/lib/gems
sudo chown -fR nome_utente .
```

(Non scordatevi il punto!)

Adesso dobbiamo pensare alla parte grafica. Un’applicazione moderna non può non avere una GUI con cui l’utente può interagire con il mouse. Tra le tante offerte disponibili (Fox, Tk, wxWidgets, GTK, Qt, ecc…) ho deciso di scegliere le wxWidgets, diffuse, moderne, documentate, libere, multipiattaforma. Esiste un binding apposito per Ruby che si chiama **wxRuby**: è molto documentato e si installa come una semplice “gemma” (chi non sa cosa sia una gemma, pensi ad essa come ad un pacchetto autoinstallante di un programma Ruby). Prima però dobbiamo accertarci che le librerie siano presenti sul nostro sistema:

```
sudo apt-get install libwxbase2.8-0 wx2.8-headers
```

Fatto questo, installiamo wxRuby. Lo possiamo fare sia da terminale con

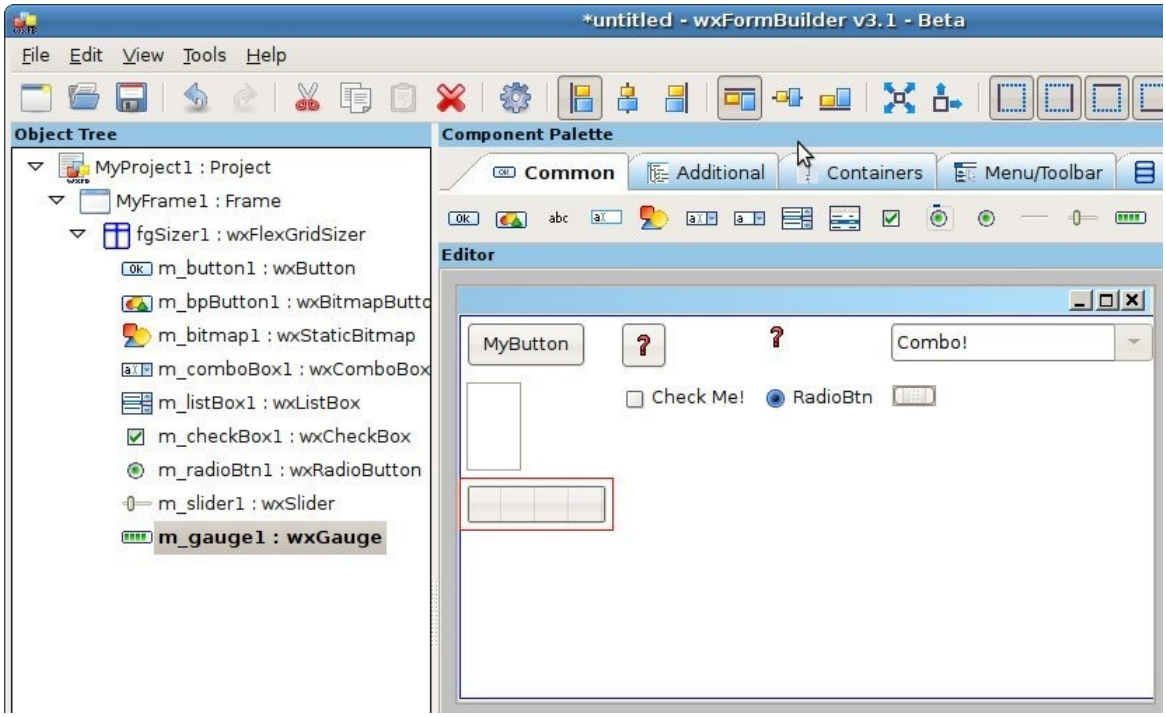
```
sudo gem install wxruby
```

sia tramite NetBeans (adesso funziona) andando in Strumenti/Ruby Gems. Scegliete il modo che più vi piace. Insieme a wxRuby installate anche la gemma **wxSugar**: servirà in seguito.

Fatto questo dobbiamo pensare al disegnatore di interfacce. Ho scelto **wxFormBuilder** perché è semplice, lavora con le wxWidgets, è multipiattaforma ed offre un repository per Ubuntu (ricordatevi di scaricare ed installare prima la chiave di autenticazione e poi di scegliere correttamente la versione della vostra distribuzione). Aggiunto il repository e riletta la lista dei pacchetti, installate wxFormBuilder con

```
sudo apt-get install wxformbuilder
```

Una volta installato, questo è disponibile in Applicazioni/Programmazione:



wxFormBuilder

wxFormBuilder salva le interfacce create in formato C++ oppure XRC. A noi il primo ovviamente non interessa perché stiamo usando Ruby come linguaggio, per cui salvate le interfacce in formato XRC. Quello di cui abbiamo bisogno adesso è un convertitore da XRC a Ruby. Il programma che servirà a questo scopo è **XRCise**, che è contenuto nella gemma wxSugar che abbiamo installato precedentemente. Però se diamo da terminale il comando xrcise, otterremo solo un messaggio di errore. Perché? Perché il file è installato in un percorso non standard ed il sistema semplicemente non lo trova. Le soluzioni sono 2: o lo invochiamo sempre con il suo percorso assoluto, che è

```
/var/lib/gems/1.8/bin/xrcise
```

oppure inseriamo la cartella dove risiede nella variabile di sistema che contiene il percorso di

ricerca dei comandi. Ovviamente questa seconda soluzione è la più pratica perché evita di digitare continuamente il percorso completo. Apriamo il file ~/.bashrc o da terminale con un editor di testo oppure anche con GEdit ed aggiungiamo, in fondo al file, le seguenti righe:

```
PATH=$PATH:/var/lib/gems/1.8/bin
export PATH
```

Fatto questo, chiudete tutti i terminali aperti e riavviatene uno per verificare che, scrivendo xrcise, il sistema trovi il comando nel percorso indicato.

Adesso siete pronti a programmare in Ruby con un ambiente di sviluppo veramente completo: avrete un ottimo disegnatore di interfacce, un programma per convertire le interfacce in codice Ruby ed un editor potente e completo per scrivere le vostre applicazioni. Buon divertimento!