Guida Ruby on Rails: il linguaggio Ruby – Parte 2

Bentrovato a questo nuovo capitolo della guida su Ruby on Rails.
Quest'oggi proseguiremo, sulla scia della precedente lezione, nel tentativo di rendere la sintassi di Ruby più chiara a tutti. La scorsa volta abbiamo parlato di stringhe, array, hash e regular expression. Oggi quindi, iniziamo con lo sviscerare la parte logica del linguaggio.

Le strutture di controllo

Come la maggior parte dei linguaggi che si rispettino, anche Ruby possiede tutte le consuete strutture di controllo come *if*, *while* e i *loops*. Come abbiamo già visto in precedenza, l'unica differenza significativa sta nell'assenza di parentesi graffe a favore dell'utilizzo di *end*. Per esempio, un ciclo *if* appare così:

mentre un ciclo *while*, ha più o meno questo aspetto:

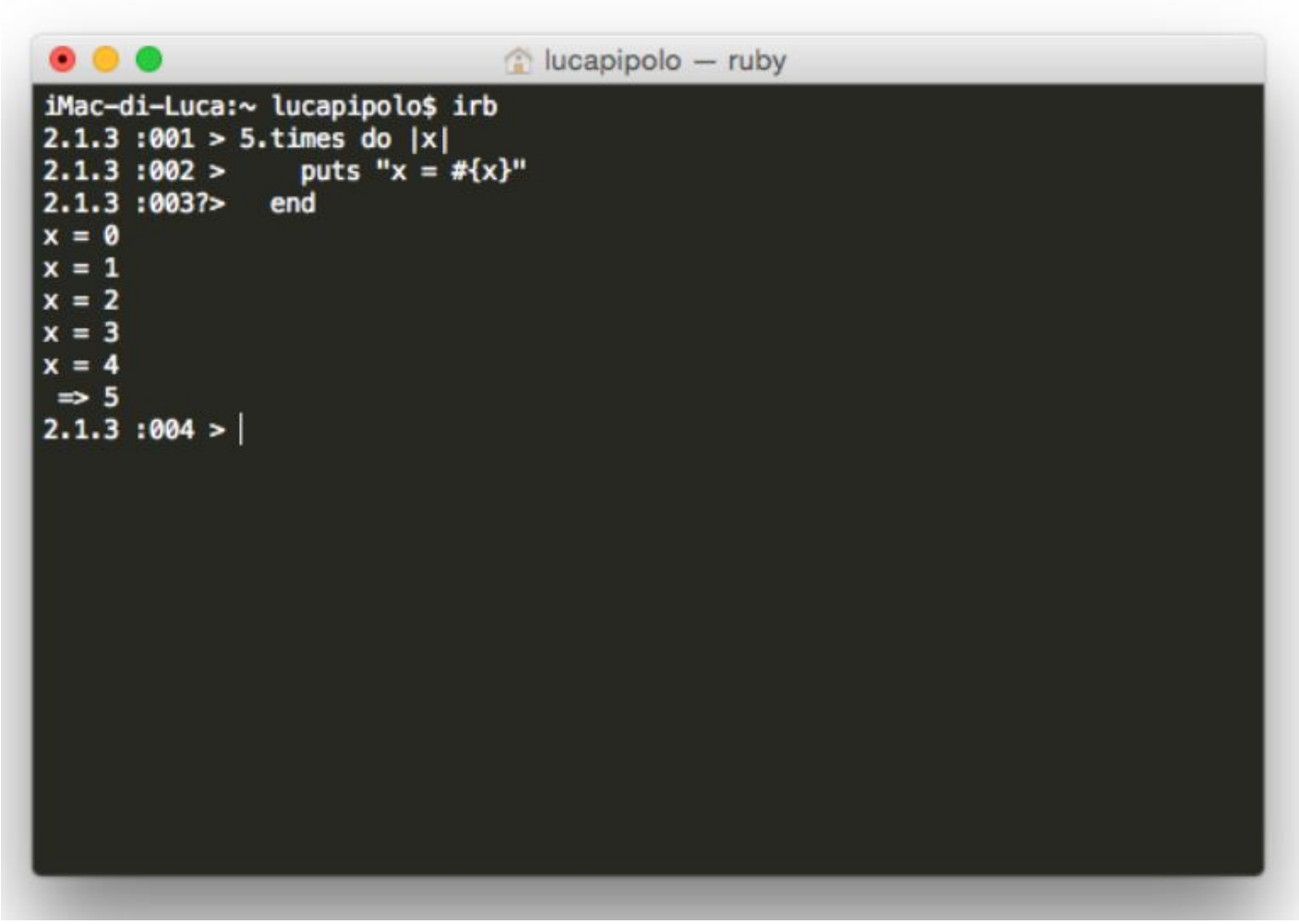
Ruby utilizza, però, anche alcune varianti di queste strutture: *unsless*, ad esempio, è molto simile al controllo *if*, a differenza del fatto che controlla che la condizione non sia vera. Parimenti *until* rispecchia *while*, ad eccezione del fatto che permette il ripetersi del loop finché la condizione risulti vera.

Inoltre, qualora l'espressione sia singola, esiste la possibilità di scrivere tutto su una riga senza troppi fronzoli:

Blocchi

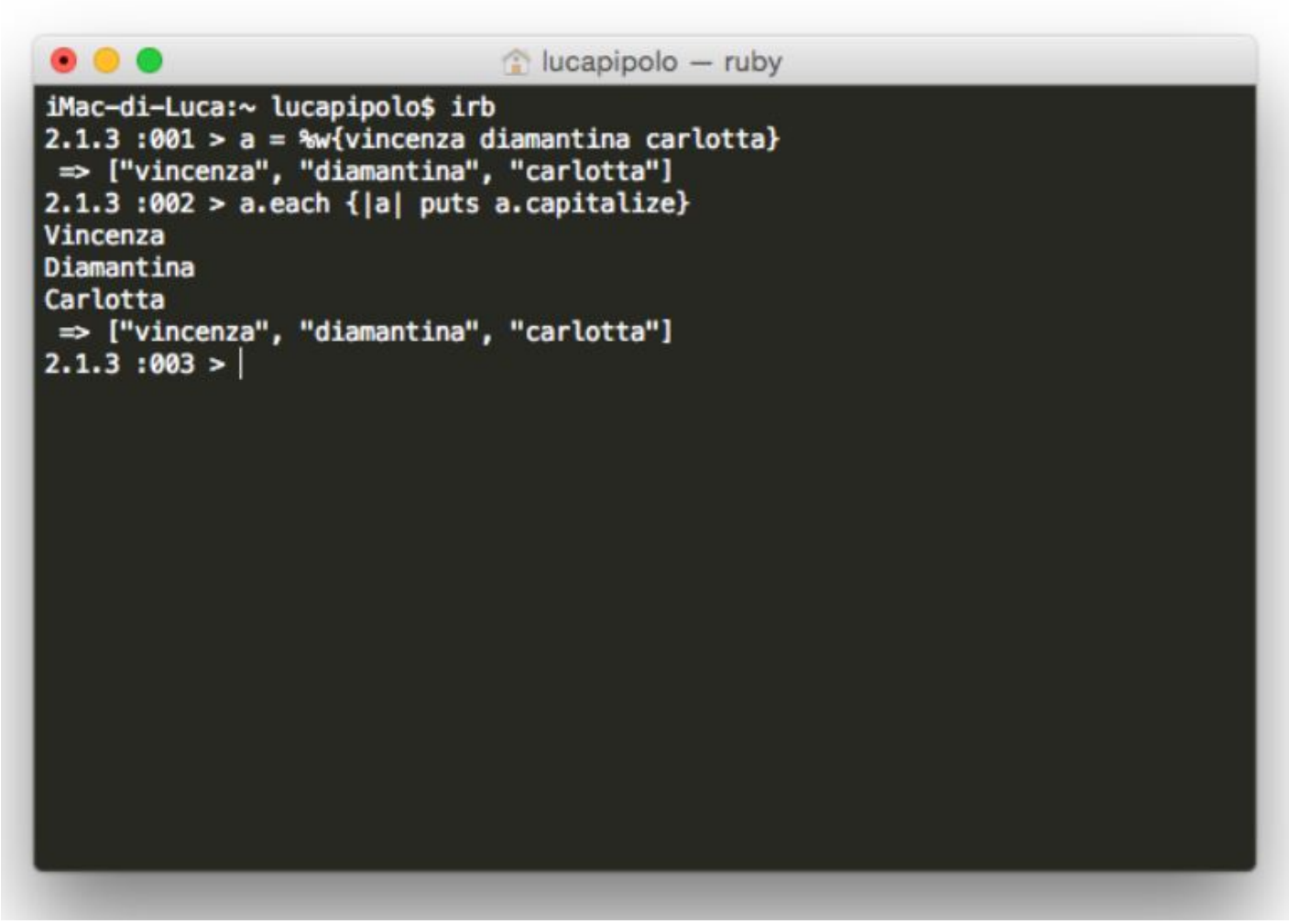
I blocchi non sono altro che porzioni di codice tra parentesi graffe o tra *do... end*. Nello specifico è comune convenzione utilizzare le parentesi per i blocchi su singola linea e *do... end* per i blocchi estesi su più linee.

Come hai potuto notare prima del blocco *do… end* è stato utilizzato il metodo *times()* che indica, in associazione con un numero intero, il numero di volte che il blocco deve essere ripetuto. Nel caso affrontato poc'anzi la console restituirà:



Spesso i blocchi vengono utilizzati in unione alle strutture di controllo di cui abbiamo discusso nel paragrafo precedente. Ad esempio, è prassi comune imbattersi in un codice del tipo:

che generà un output di questo tipo:



Come hai potuto notare dai due precedenti esempi, all'interno del simbolo || i valori vengono passati al blocco come parametri.

Classi

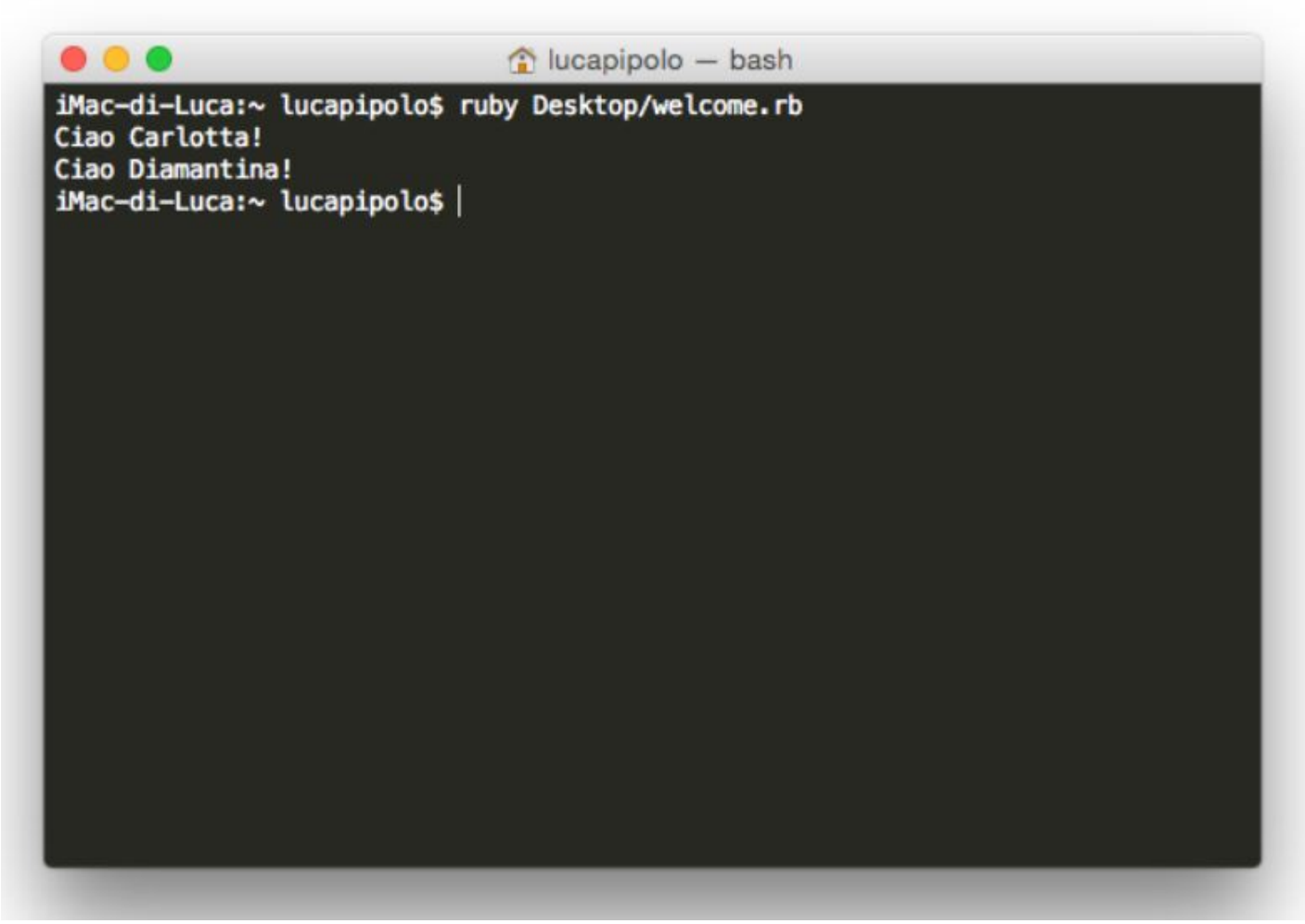
Le classi in Ruby rispettano i canoni logici comuni agli altri linguaggi di programmazione, ovvero incarnano uno schema per la rappresentazione di un oggetto in base alla differenziazione per insiemi e sotto-insiemi.

Ogni classe inizia con la parola chiave *class*, seguita dal nome attribuito che deve iniziare con la lettera maiuscola, e infine termina con il consueto *end*. All'interno del corpo possono essere definiti i metodi della classe e i metodi d'istanza. I metodi della classe sono preceduti da *self*. e rendono il metodo accessibile all'interno dell'intera applicazione. Contrariamente, i metodi d'istanza sono accessibili solamente all'interno della classe stessa. Prima di complicare ulteriormente il concetto arricchendolo di altre nozioni; facciamo un esempio:

Nel caso volessi procedere a rendere accessibili i valori di un metodo d'istanza all'esterno della classe, dovresti creare metodi che ritornino i loro valori. Ad esempio la classe:

richiamata dal seguente codice:

avrebbe come risultato:



Se vuoi eseguire questo codice all'interno della console ti conviene, a causa della sua corposità, di scriverlo all'interno di un file .rb e richiamarlo da terminale con il comando

Moduli

I moduli sono simili alle classi ma, a differenza di esse, non è possibile creare oggetti da un modulo. A cosa servono quindi i moduli? Semplice. I moduli si prefiggono due propositi: in primo luogo agiscono da namespace raggruppando quindi metodi simili non diversamente definiti altrove e, in secondo luogo, permettono di scambiare funzionalità tra classi differenti, pratica meglio definita con il nome di mixin. Anche in questo caso la sintassi è elementare ed estremamente simile a quella delle classi.

Conclusioni

Questa non è, e non vuole essere, una guida alla programmazione o al linguaggio Ruby; l'obiettivo di queste lezioni incentrate prettamente sul linguaggio Ruby è quello di darti una infarinatura generale di nozioni senza scendere negli infiniti dettagli che tale linguaggio meriterebbe, dandoti la possibilità di riconoscere i costrutti basilari. Per chi di voi volesse approfondire maggiormente Ruby potete far riferimento a tre guide considerate universalmente le migliori:

- Ruby User's Guide: della serie The Pragmatic Programmer's Guide, disponibile gratuitamente la prima versione inglese;
- Programming Ruby: traduzione inglese della guida originale giapponese redatta dallo stesso Yukihiro Matsumoto, creatore di Ruby;
- The Ruby Programming Wikibook: comodo cookbook in formato wiki adatto a persone con conoscenze di livello base o intermedio;

Ad ogni modo, procedendo con la guida, ci imbatteremo in altri concetti che spiegherò e chiarirò seduta stante in modo da non lasciare alcun buco nella realizzazione della nostra web app in Ruby on Rails. Quindi... non temere, tutto sarà chiaro!

GUIDA RUBY ON RAILS: INDICE LEZIONI
1) Introduzione
2) L'ambiente di lavoro e la nostra prima app
3) Un assaggio di dinamicità
4) Architettura di un'applicazione
5) Il linguaggio Ruby - Parte 1
6) Il linguaggio Ruby - Parte 2
7) Creiamo c-Bookcase
8) Convalida e test
9) Ruby on Rails: page layout
10) Guida Ruby on Rails: creiamo il carrello