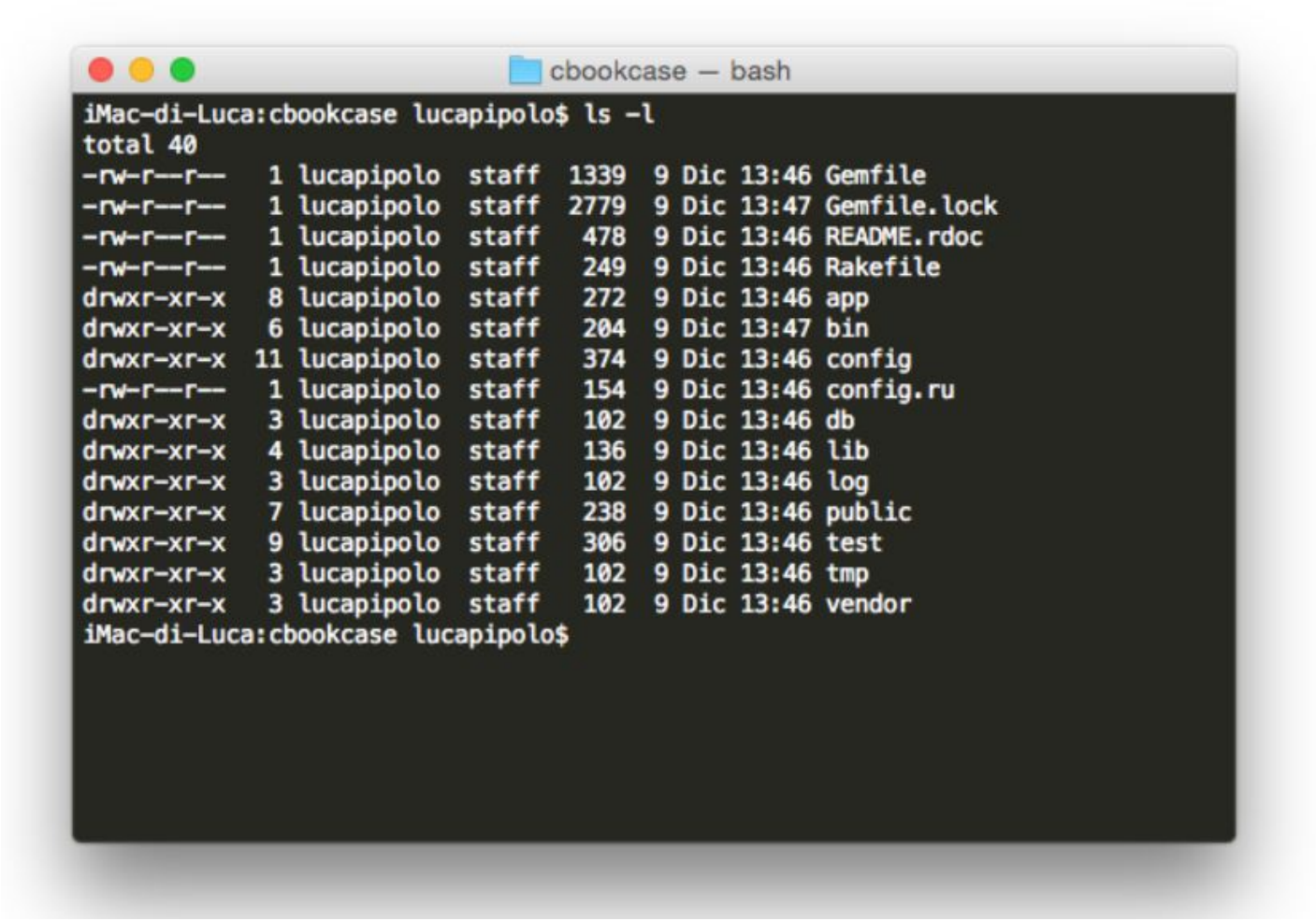
Guida Ruby on Rails: creiamo c-Bookcase

Eccoci finalmente pronti ad affrontare la parte più interessante della guida: la creazione di una vera e propria web app dedita a gestire un catalogo online di libri! Nel corso di questa lezione, per spezzare la monotonia delle lunghe lezioni teoriche precedenti, passeremo subito all'azione, salvo ritornare, nel corso dei prossimi articoli, a definire il processo logico e strutturale dell'applicazione. Adesso, basta con le chiacchiere e mettiamoci all'opera!

Il sistema di gestione dei libri

Il primo passaggio è senz'altro quello di creare una nuova applicazione rails e denominarla c-Bookcase, il nome che ho scelto per questo esperimento. Ricordi come farlo, no? L'abbiamo visto nella seconda lezione! Apriamo il terminale e digitiamo:

Una volta completato il processo ed esserci spostati nella directory appena creata, dovrebbe apparirci una struttura di questo tipo:



Abbiamo, adesso, bisogno di creare ed associare un database alla nostra applicazione. Ho scelto di utilizzare, per questo specifico caso, SQLite3 che, fra l'altro, è il database che Rails sceglie come default, quindi proveremo immediatamente l'ebrezza di seguire il principio del *convention over configuration*, descritto nell'introduzione di questa guida. E, allora, il nostro database? È già stato creato e configurato senza alcun passaggio aggiuntivo. Comodo, no?!

Tutto quello che ci resta da fare, quindi, è riempire il nostro database con delle tabelle ed associarle ai componenti Rails, ovvero al model che gestirà le tabelle, alle views che costituiranno l'interfaccia e al controller che orchestrerà il tutto. In Rails è possibile fare tutto questo tramite un unico e semplice comando: *scaffold*. Accertiamoci di essere all'interno della cartella della nostra applicazione e digitiamo:

Tra i vari file generati da questo comando, nella directory *db/migrate* troveremo un file dal nome lungo e numerico, che termina con l'estensione .rb, come ad

esempio: *20141209131908_create_books.rb*. Questo file all'apparenza così insolito non è altro che una migrazione, ovvero un cambiamento dei dati indipendente dal database, attraverso il quale possiamo modificare sia lo schema del database sia i dati all'interno delle tabelle. Apriamo quindi il nostro file di migrazione che, come detto, troveremo nella directory *db/migrate*, e che si chiamerà con un nome numerico corrispondente alla timestamp UTC del momento in cui è stato creato. Specifichiamo che il valore *price*, può accettare 6 cifre di cui due dopo la virgola. Per fare ciò digitiamo:

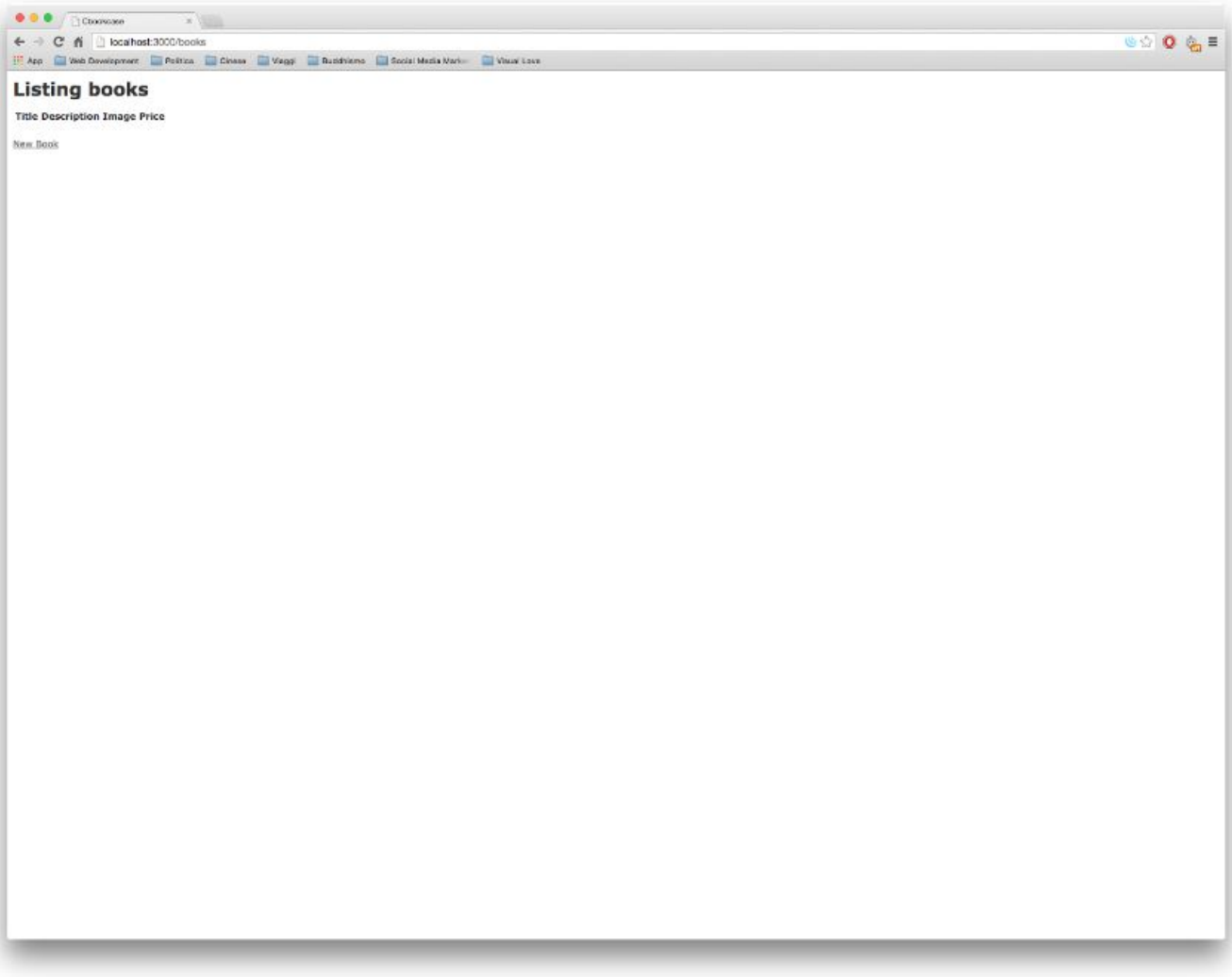
Una volta scritta questa modifica, dobbiamo applicarla. Come fare? Con un altro semplice comando denominato *rake*. *Rake* rappresenta il nostro assistente personale al quale poter delegare molti compiti. In questo caso utilizziamolo per applicare la migration appena modificata digitando da terminale:

Così facendo, *rake* applica tutte le modifiche non ancora attive.

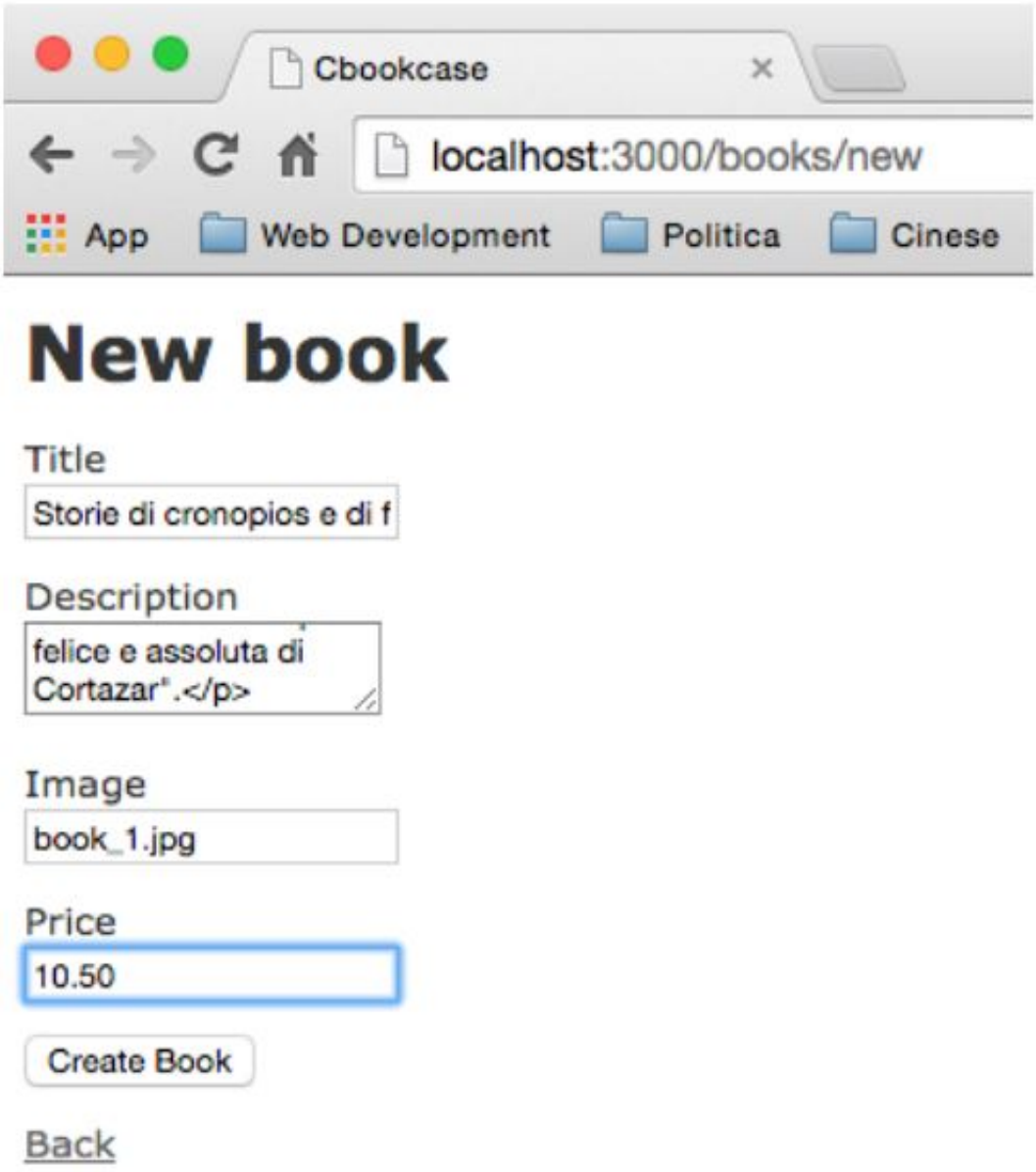
Visualizziamo la lista dei libri

Con pochi e semplicissimi comandi abbiamo creato un'applicazione, associato un database ad essa e modificato la nostra prima migration. Tutto quello che ci resta da fare, ora, è visualizzare il risultato dei nostri, seppur minimi, sforzi! Per far ciò, in una nuova finestra del terminale, digitiamo:

Queste due parole faranno partire il web server, nel caso di default WEBrick, e ci permetteranno di visualizzare la nostra applicazione semplicemente visitando l'indirizzo *localhost:3000/books*. La schermata che ci troveremo davanti sarà simile a questa:



Nient'altro che una semplice lista di libri… vuota! Riempiamola utilizzando le azioni che *scaffold* ha creato per noi. Facciamo click su new book e, nella pagina *http://localhost:3000/books/new* che si aprirà, riempiamo i campi come se volessimo aggiungere un elemento al nostro catalogo:



Tornando alla pagina principale della nostra app scopriremo che, all'interno del nostro database, è stato inserito correttamente il libro, che ora ci appare in lista con tutti i dati ad esso associati. Tramite le altre azioni possiamo modificare ed eliminare il libro appena creato.

Modifichiamo la lista

La nostra lista appare, però, un tantino spiacevole alla vista. Modifichiamo il file CSS per renderla più accattivante!

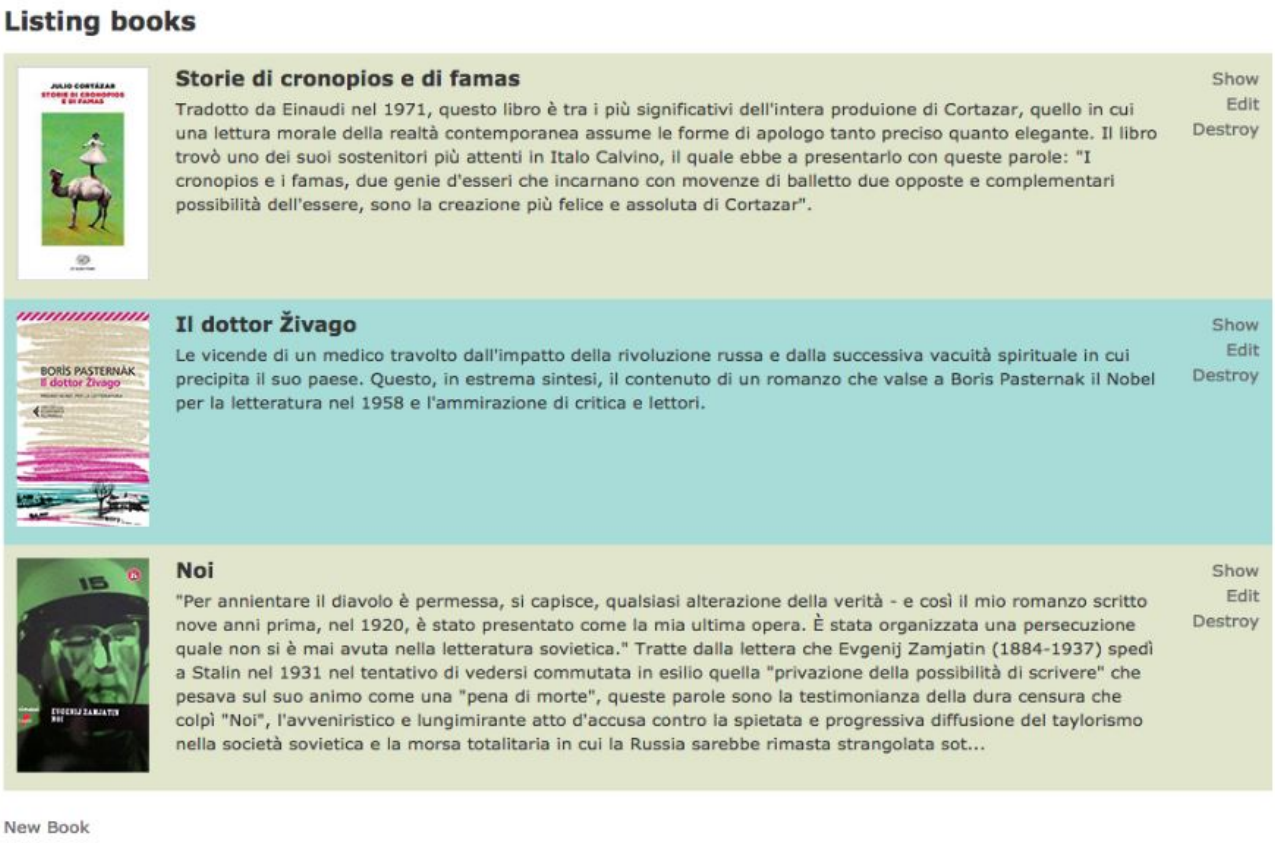
Apriamo la view della nostra lista (*app/views/books/index.html.erb*) e modifichiamola nel seguente modo:

Analizzando il codice, puoi facilmente capire che:

- abbiamo creato una tabella dove racchiudere i libri, contenente tre colonne, rispettivamente una per le immagini, una per il titolo e la descrizione, e una per le azioni;
- abbiamo usato il ciclo each per assegnare ad ogni libro una singola riga;
- abbiamo usato il metodo cycle per alternare il colore di sfondo tra righe pari e dispari;
- abbiamo usato il metodo truncate per limitare il numero di caratteri mostrato a 980 e il metodo strip_tags per rimuovere i tag html dall'output.

Modifichiamo, di conseguenza, il relativo file CSS (*app/assets/stylesheets/books.css.scss*) come meglio crediamo; nel file zip presente nelle conclusioni di questo articolo, puoi trovare quello creato secondo il mio gusto. Infine, non ci resta che aggiungere le copertine dei libri nel percorso *app/assets/images*, utilizzando come nome del file gli stessi che abbiamo specificato nell'applicazione.

Al termine di queste modifiche il risultato dovrebbe apparire pressappoco così:



Conclusioni

Nel corso della prossima lezione vedremo come controllare che non vengano passati parametri sbagliati durante l'immissione di nuovi elementi e come assicurarci che ogni aspetto della nostra applicazione funzioni come immaginiamo. Prima di salutarti ti ricordo che, come consuetudine, puoi scaricare il codice completo della lezione da questo link.

GUIDA RUBY ON RAILS: INDICE LEZIONI

10) Guida Ruby on Rails: creiamo il carrello