Guida Ruby on Rails: convalida e test

Nel corso del precedente articolo abbiamo finalmente creato l'applicazione che ci accompagnerà per tutto il resto della guida. Nello specifico, abbiamo associato un database all'applicazione, generato il primo *scaffold*, utilizzato il comando *rake* per controllare una *migration* e, infine, abbiamo modificato le prime *views*.

Oggi ci occuperemo, invece, di accertarci che non vengano passati dati errati al nostro database. Ad esempio, sarebbe illogico accettare come valore del campo *price* una stringa di testo, così come sarebbe inutile consentire di lasciar vuoto il campo *title* o quello *description*.

Convalidiamo i campi

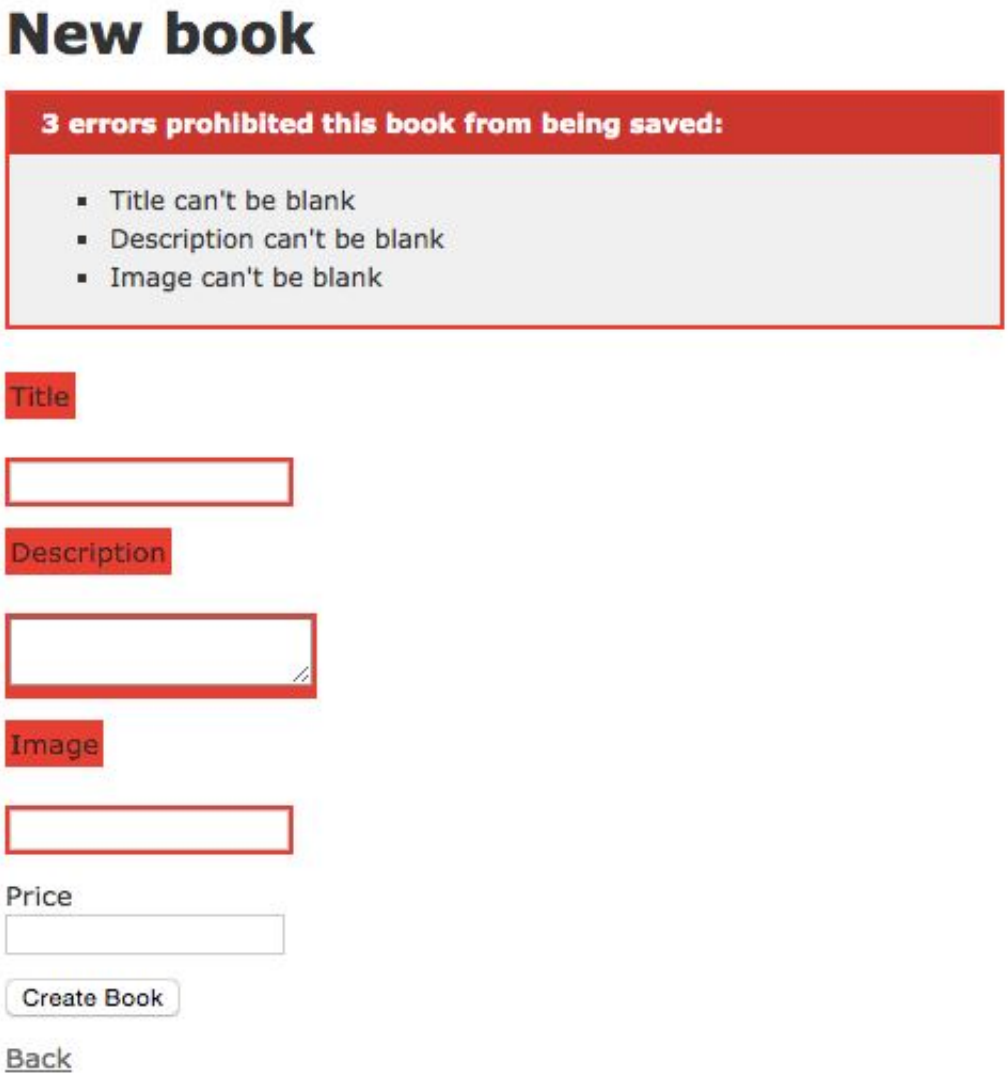
Il controllo di cui abbiamo discusso poc'anzi deve avvenire all'interno del *model*, come la logica del triumvirato MVC impone. Apriamo, quindi, il file *app/models/books.rb*; ci apparirà piuttosto vuoto:

Diamoci da fare e rendiamolo utile!

Il primo passo è quello di accertarci che tutti i campi necessari contengano effettivamente un valore e non siano vuoti. Per fare ciò, ci basterà utilizzare il metodo validates che si occuperà di controllare uno o più campi a confronto di una o più condizioni. Nel nostro caso scriveremo:

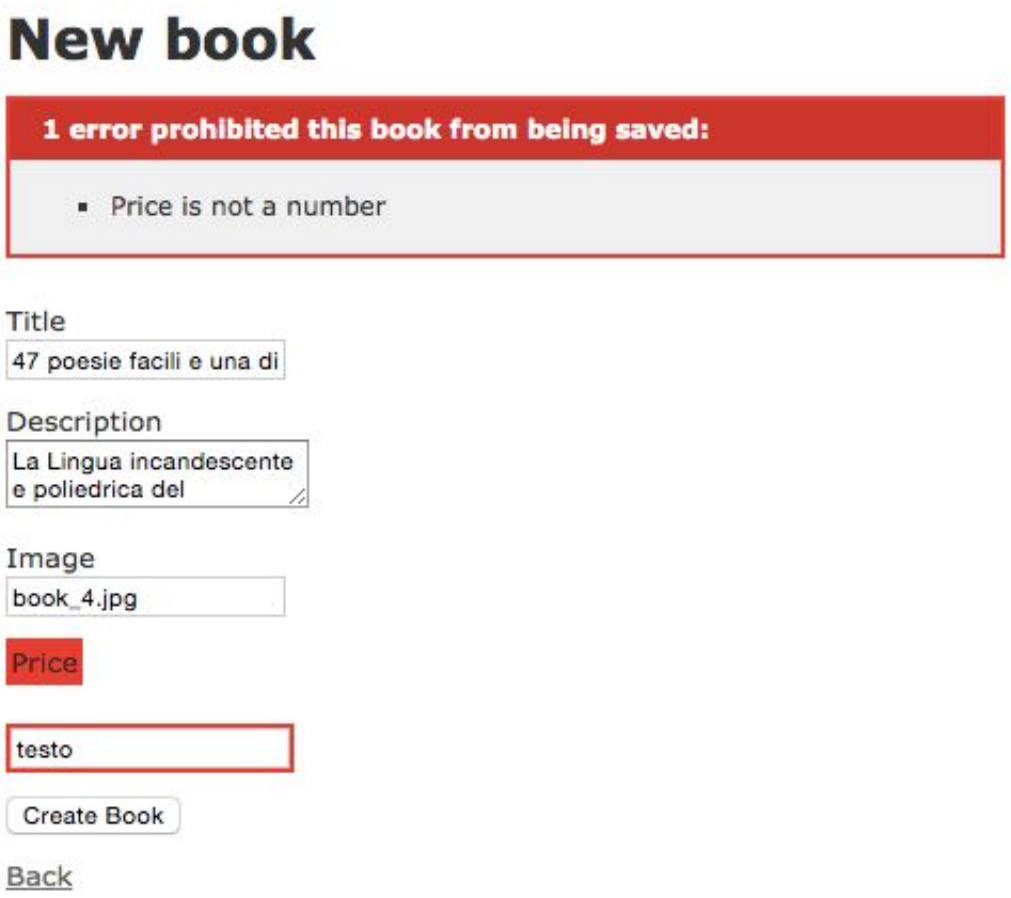
dove i campi da controllare sono *title*, *description* e *image*, mentre la condizione è, come facilmente si può intuire, rappresentata dal valore *true* di *presence*.

Proviamo adesso a creare un nuovo libro senza rispettare i parametri appena imposti. Ciò che apparirà ai nostri occhi sarà:



Come puoi notare, manca il controllo sul campo *price*. Questo perché non solo ci interessa controllare che il campo sia effettivamente compilato, ma anche che sia un valore numerico e soprattutto che sia maggiore di 0. Per far ciò la condizione da rispettare dev'essere:

che, in caso di errore, restituirà:



Non ci resta che accertarci di soddisfare ulteriori due condizioni: il campo *title* non deve risultare un duplicato e il campo *image* deve contenere l'estensione jpg, png o gif.

Per controllare la prima di queste due condizioni, basta ricorrere alla condizione *uniqueness* in questo modo:

La seconda, invece, richiede un meccanismo leggermente più complesso per funzionare a dovere: l'utilizzo di una *regular expression*. Nello specifico dobbiamo essere sicuri che, nel campo *image*, sia presente almeno un carattere seguito da un estensione .jpg, .png o .gif. Il codice da scrivere risulterà, quindi:

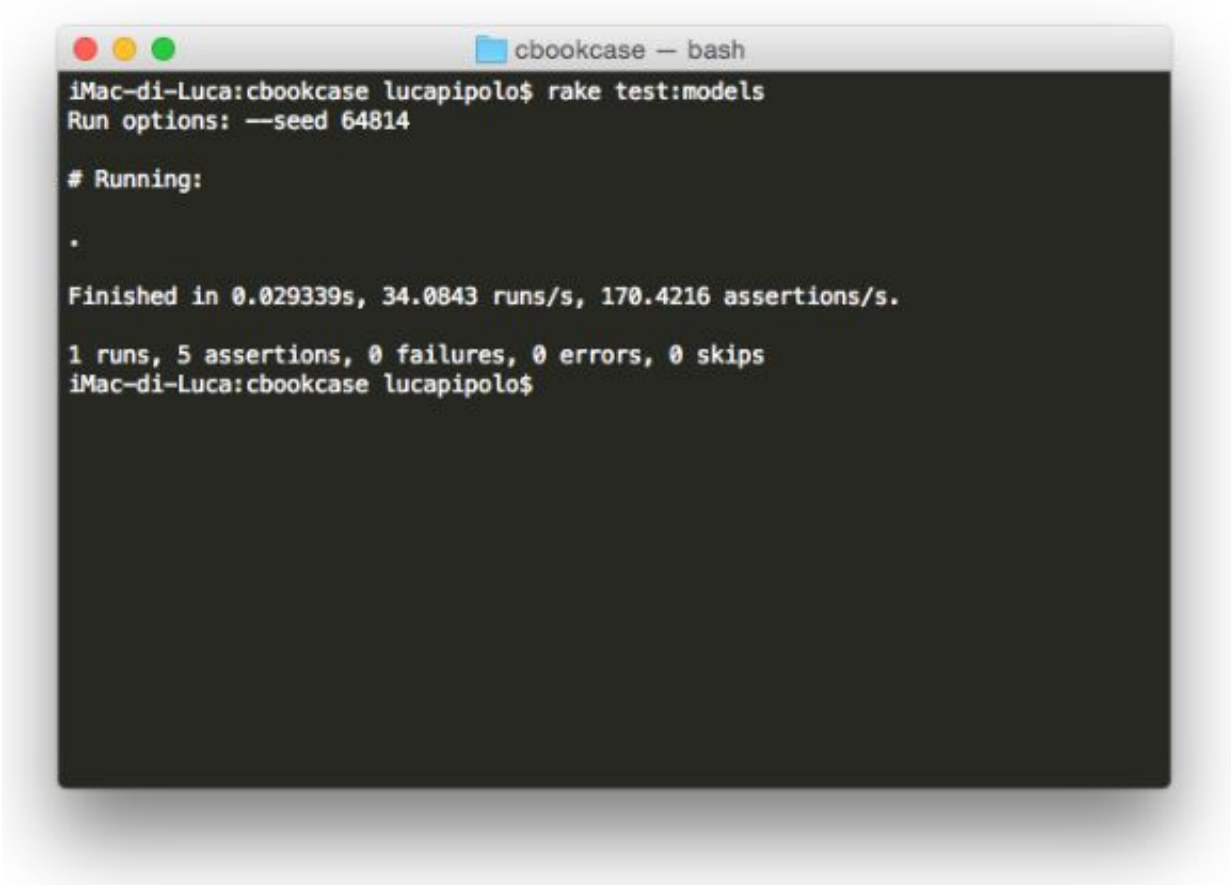
Prima di proseguire nella spiegazione è necessario aprire una breve parentesi. Le *regular expression* sono un argomento complesso e, in questo caso, mi sono limitato ad utilizzarne una che, seppur non completamente efficace, può essere facilmente spiegata a un pubblico nuovo

all'argomento. Se vuoi approfondire questo aspetto, qui puoi trovare una guida più esaustiva.

Model test

Una volta conclusa la fase della validazione, possiamo passare a quella più complessa del test. Occupiamoci, quindi, di testare il model che ci interessa aprendo *test/models/book_test.rb* e aggiungendo:

È facile intuire come il metodo *assert* restituisca *true* se la condizione è soddisfatta e *false* se la condizione non è soddisfatta. Con il codice appena creato abbiamo controllato che, aggiungendo un nuovo libro privo dei campi che abbiamo reso obbligatori, vengano restituiti i corrispondenti errori. Adesso, digitando da terminale *rake test:models*, avremo un output del genere:



Allo stesso modo testiamo il meccanismo riguardante il prezzo.

e quello relativo all’estensione delle immagini.

Conclusioni

Nel corso di questa guida torneremo più volte sul tema dei test poiché, come hai potuto capire, è un meccanismo fondamentale per assicurarci, passo dopo passo, di non tralasciare o sbagliare nulla. Nonostante ciò, se vuoi approfondire il testing prima del tempo, ti consiglio di far riferimento alla guida che trovi sul sito ufficiale. Nel prossimo articolo torneremo a sviluppare nuove funzioni per la nostra app; in particolare, aggiungeremo un carrello!

Prima di concludere, puoi scaricare il codice completo dell’app da questo indirizzo.

GUIDA RUBY ON RAILS: INDICE LEZIONI