Guida Ruby on Rails: page layout

Nello scorso articolo abbiamo accantonato momentaneamente lo sviluppo della nostra applicazione *cbookcase* per dare spazio alla fase della convalida e del testing. Riprendiamo quindi a scrivere codice per il nostro catalogo online di libri!

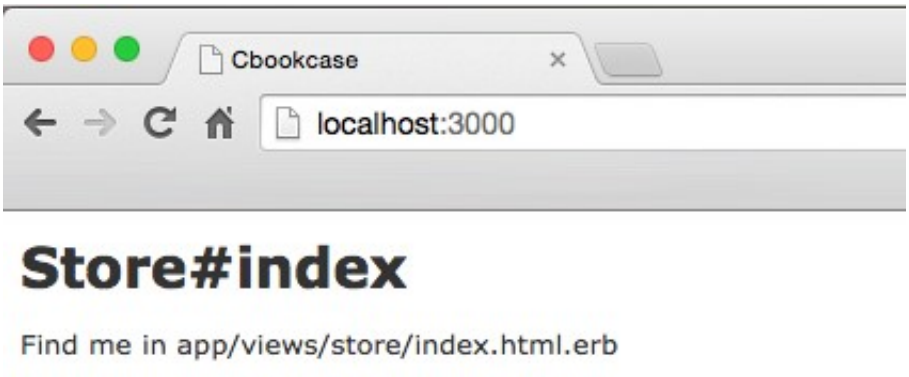
Fino ad ora ci siamo concentrati sulla parte amministrativa dell'applicazione, curando le view dedicate alla gestione dei libri. Ma come vogliamo mostrarla all'utente che visiterà il nostro sito? Quale sarà il layout dello store e come lo costruiremo? Scopriamolo subito!

Creiamo lo store

La prima cosa da fare è creare un controller denominato *store* con lo stesso procedimento già utilizzato in precedenza. Apriamo, quindi, il terminale e digitiamo:

A procedura ultimata possiamo, come è consuetudine, visualizzare il frutto dell'operazione all'indirizzo *localhost:3000/store/index*. Ma non ci basta: vogliamo che questa pagina compaia come index, dal momento che contiene tutto ciò che l'utente si aspetta di vedere a primo impatto visitando il nostro sito. Apriamo il file *config/routes.rb* e aggiungiamo al codice:

Il risultato dei nostri sforzi sarà che, aprendo l'index all'indirizzo *localhost:3000*, vedremo comparire:



Ovviamente in questa pagina attualmente vuota dobbiamo visualizzare tutti i prodotti attualmente disponibili, gli stessi che gestiamo dalla pagina di amministrazione. Come prima cosa occupiamoci di far comunicare il controller con il model per richiedere la lista dei prodotti disponibili ordinati per titolo. Apriamo *controllers/store_controller.rb* e aggiungiamo:

Scriviamo, quindi, la relativa view aprendo il file *views/store/index.html.erb* e digitiamo:

Fermiamoci un attimo ad analizzare il codice per capire esattamente cosa avviene.

Con il codice creiamo il ciclo che andrà a formare un div per ogni libro e che conterrà, per ognuno: il tag img generato dal metodo image_tag, la descrizione passata tramite sanitize e il prezzo formattato grazie a number_to_currency. A questa struttura dobbiamo ovviamente affiancare un foglio di stile; apriamo *assets/stylesheet/store.css.scss* e scriviamo:

Il risultato dovrebbe ora somigliare a:

Aggiungiamo un layout

Non ci basta certo che la index restituisca i libri a catalogo; un’intestazione e una barra di

navigazione sono elementi essenziali per completare una, seppur minimale, interfaccia grafica. Per far ciò, dobbiamo operare su quello che in gergo viene denominato layout e che, nelle applicazioni Ruby on Rails, è contenuto nella cartella *views/layouts*; nel nostro caso il file si chiamerà *application.html.erb*. Il nuovo contenuto sarà:

Anche questa volta analizziamo più da vicino il codice che abbiamo appena utilizzato, tralasciando l'ovvio codice html. Nella sezione *head* sono presenti importanti elementi:

- *stylesheet_link_tag* e *javascript_include_tag* — generano i collegamenti ai relativi fogli di stile e script e abilitano l'opzione data-turbolink-track;
- *csrf_meta_tags* — previene gli attacchi di tipo cross-site scripting. Avremo modo di approfondire questo argomento nel corso delle prossime lezioni.

Nel body — in particolare all'interno del tag — è invece presente , il quale ha il magico compito di riempire la pagina con la view invocata, nel nostro caso: store/index.html.erb.

Non ci resta che creare, nel file *assets/stylesheets/application.css.scss*, il relativo foglio di stile:

per ottenere questo risultato finale:

Conclusione

Adesso che abbiamo creato la pagina del nostro store ti sarai senz’altro reso conto dell’assenza

di una parte fondamentale: la possibilità di aggiungere un prodotto al carrello! Nel corso del prossimo articolo ci dedicheremo proprio alla creazione di un carrello basilare che, nelle successive lezioni, renderemo sempre più dinamico ed efficiente grazie all'utilizzo di tecnologie AJAX.

Nel frattempo, come al termine di ogni lezione, puoi scaricare il codice integrale da questo link.

GUIDA RUBY ON RAILS: INDICE LEZIONI