Guida Ruby on Rails: creiamo il carrello

Bentornato a questo nuovo appuntamento con la guida Ruby on Rails.

Nel corso della scorsa lezione abbiamo abbellito la nostra applicazione aggiungendo un layout. Quest'oggi, invece, ci occuperemo della funzionalità che ci permette di aggiungere una parte fondamentale al nostro store: il carrello!

Basta tergiversare: rimbocchiamoci le maniche ed iniziamo!

Troviamo il carrello

Nella logica della nostra applicazione vogliamo che l'utente, dopo aver visionato gli articoli che offriamo, aggiunga quelli che desidera ad un cartello. Questo carrello deve essere modificabile e ovviamente accessibile fino al momento del checkout durante il quale sarà possibile procedere al pagamento e quindi alla chiusura del carrello stesso.

Per questo motivo abbiamo bisogno di conservare il carrello nel nostro database e il suo identificativo unico, *cart.id*, all'interno della sessione. In questo modo, ogni volta che l'utente lo richiede, possiamo risalire al carrello relativo alla giusta identità.

Apriamo, come al solito, il terminale e digitiamo:

Seguito dal comando per la migrazione del database:

Poiché Rails fa apparire, al controller, la sessione attuale come un hash, dobbiamo memorizzare l'ID del carrello all'interno della sessione riferendoci ad esso con il simbolo *:cart_id*.

Apriamo quindi il file *app/controllers/concerns/current_cart.rb* ed aggiungiamo il seguente codice:

Come puoi vedere, il metodo *set_cart()*, prima cerca il simbolo *:cart_id* all'interno della sessione, e subito dopo il carrello corrispondente proprio a quell'ID. Nel caso in cui la ricerca risulti vana, procederà alla creazione di un nuovo carrello, memorizzando l'ID del carrello appena creato proprio all'interno della sessione e restituendo il carrello stesso.

Colleghiamo i libri al carrello

Ovviamente, affinché il nostro carrello serva veramente a qualcosa, dobbiamo prevedere la possibilità di aggiungere ad esso dei prodotti, selezionandoli dal nostro catalogo online.

Ancora una volta, l'operazione è semplice ed è sintetizzabile in questo comando da terminale:

seguito dal consueto:

Adesso il nostro database avrà uno spazio dedicato in cui memorizzare i collegamenti tra le voci del carrello, il carrello e i prodotti. Per capire meglio il tipo di relazione che lega questi tre oggetti, apriamo il file *line_item.rb* dalla cartella *app/models*. Questo è quello che vedremo:

Possiamo facilmente intuire che la chiamata del metodo *belongs_to()* impone un importante legame di "parentela": le righe nella tabella *line_items* sono figlie delle righe nella tabella *carts* così come di quella *books*. In poche parole, non può esistere una voce del carrello (una *line item*) senza una corrispondenza con il relativo carrello e libro.

Tuttavia, la corrispondenza deve anche essere inversa. Apriamo quindi il file *app/models/cart.rb* e specifichiamo che:

Dove:

- *has_many :line_items* indica che per ogni carrello possono esistere una ma anche più voci;
- *dependent: :destroy* indica che l'esistenza delle voci all'interno del carrello dipende, come impone la logica, proprio dall'esistenza del carrello stesso;

Infine, come ultimo step, è opportuno modificare il model di *book* (*app/models/book.rb*) affinché più voci di carrello possano riferirsi ad un singolo libro (immagina più utenti con più carrelli aperti che comprano lo stesso libro) e affinché, prima di procedere alla distruzione di un libro, si abbia la certezza che non esistano carrelli pendenti che hanno proprio quel libro come voce. Quindi, il nostro file *models/book.rb* diventerà:

Aggiungiamo un pulsante

Ricapitolando: abbiamo creato il carrello con un ID univoco e le sue voci, collegandole in modo reciproco all'elenco di libri e al carrello stesso. Cosa resta da fare? Aggiungere il pulsante "Add to Cart" al nostro elenco di libri!

Per fare ciò scegliamo di utilizzare il metodo *button_to()* specificando il testo che vogliamo mostrare sul nostro pulsante e, soprattutto, specificando l'URL che dovrà essere aperto. Poiché nel momento in cui clicchiamo sul pulsante stiamo creando una voce all'interno del carrello, il nostro interlocutore sarà *LineItem* ed il metodo *create()* che invocheremo corrisponderà proprio a quello specificato nel suo controller. Recupereremo, quindi, il percorso da *LineItem* utilizzando ancora una volta la magia di Rails e semplicemente utilizzando il metodo *line_items_path* e passando tra parentesi l'ID del libro al quale ci stiamo riferendo.

La spiegazione può sembrare contorta, ma l'esecuzione è sfacciatamente semplice. All'interno di *app/views/store/index.html.erb*, aggiungeremo esclusivamente questa singola linea di codice subito dopo il tag *span* relativo al prezzo:

Adesso è il turno di modificare proprio il controller *LineItems* per accertarsi di trovare il giusto carrello associato alla sessione o crearne uno nuovo qualora ancora non ce ne fosse uno. Apriamo quindi *app/controllers/line_items_controller.rb* e all'inizio del file digitiamo:

Andando oltre, all'interno dell'azione *create()* dobbiamo aggiungere la variabile *book* alla quale assoceremo il parametro *book_id* utilizzato dalla request creata poco fa. Inoltre, proprio attraverso quest'ultima variabile, costruiremo una nuova voce nel carrello. Ecco, quindi, come appare adesso l'azione *create()* modificata:

Non ci resta che cambiare la *view* relativa all'azione *show* di *cart*. Apriamo *app/views/carts/show.html.erb* e sostituiamo il contenuto predefinito con:

Il risultato finale sarà:

Conclusioni

Come puoi vedere dalle ultime due immagini il carrello sviluppato non è il massimo né in termini di layout né, soprattutto, in termini di funzionalità visto che per due voci dello stesso elemento visualizziamo il titolo ripetuto invece di utilizzare ed incrementare un indicatore di quantità. Nel corso del prossimo articolo ci dedicheremo proprio a rendere più funzionale il nostro carrello e utilizzando, più avanti ancora, tecniche AJAX per dare quel pizzico di dinamicità!

In attesa del prossimo capitolo, puoi scaricare l'intero codice sorgente aggiornato dell'applicazione tramite questo link.

GUIDA RUBY ON RAILS: INDICE LEZIONI