Guida Ruby on Rails: un assaggio di dinamicità

Nel corso dello scorso capitolo di questa guida abbiamo creato una semplice applicazione all'interno della quale abbiamo generato il controller, per ora vuoto, denominato *say* e contenente le due azioni *hello* e *goodbye*. Cerchiamo quindi, nel corso di questa lezione, di capire come l'utente possa interagire con l'applicazione stessa.

Navigazione tramite URL

La prima cosa che devi sapere è che gli utenti possono interagire con l'applicazione utilizzando gli URL. Ad esempio, digitando nel browser il percorso *localhost:3000/say/goodbye* ci apparirà una schermata simile a questa:

che ci ricorda, per l'appunto, che le due azioni create precedentemente nel controller *say* sono accessibili mediante il loro indirizzo.

Aggiungiamo l'orario

Proviamo ad aggiungere un po' di movimento a questa pagina così statica: inseriamo l'orario corrente! Per fare ciò dobbiamo agire su due differenti parti dell'applicazione: il controller e la view.

Nel controller andremo a definire la variabile d'istanza *time* nel seguente modo:

Parallelamente, all'interno della view andremo a richiamare la variabile appena definita nel controller per mostrarne il contenuto. Il file che andremo a modificare sarà, come ti aspetterai, nella directory denominata *views* e, nello specifico, all'interno della cartella che porta lo stesso nome del controller: *say*. Il file *hello.html.erb* è quello che cercavamo. Cancelliamo il contenuto e sostituiamolo con il seguente.

Ti sembrerà una sintassi strana da utilizzare, così come l'estensione .erb associata all'HTML ti apparirà nuova. L'arcano è presto spiegato! L'estensione .erb segnala a Rails di espandere il file con un sistema denominato, per l'appunto, ERB e di processare il contenuto inserito tra come fosse codice Ruby. Tutto il resto verrà normalmente interpretato come codice HTML.

Colleghiamo le pagine

Con lo stesso metodo proviamo a rendere più interessante la nostra applicazione collegando le due pagine, e quindi le due azioni, *hello* e *goodbye*. All'interno della view di hello.html.erb, inseriamo questa linea di codice:

la quale produrrà il seguente risultato.

Adesso prendiamoci un momento per analizzare il codice appena inserito. *link_to* è un metodo che richiama il valore precompilato *say_goodbye_path* il quale contiene, per l'appunto, il percorso completo dell'azione goodbye all'interno del controller *say* e ne richiama la view; in breve, rimanda all'URL *say/goodbye*. A riprova di ciò, ci basterà visualizzare il codice HTML della view di say/hello per scoprire che l'output è:

Bene, ora che abbiamo capito il meccanismo, proviamo a riprodurre la stessa procedura per collegare la pagina *goodbye* a quella *hello*. Apriamo la *view say/goodbye.html.erb* e, al suo interno, modifichiamo il codice in questo modo.

Anche in questo caso ci basterà utilizzare il metodo *link_to* per associare questa pagina con *say_hello_path*, il percorso dell'azione *hello* nel controller *say*.

Formattiamo la data

Proviamo adesso a fare qualche altro esperimento. Non ti sembra che la data sia un po' bruttina da visualizzare in questo formato? Proviamo allora a darle un aspetto a noi più familiare e, in questo modo, impariamo come concatenare stringhe e come utilizzare alcuni metodi legati a *time*. Torniamo alla *view say/hello.html.erb* e modifichiamo la riga:

in

Aggiungendo il metodo *strftime* alla variabile @*time* possiamo settare la formattazione della data al formato *%H:%M*, ovvero **ora:minuti**. Il risultato sarà, ad esempio: **10:40**. Aggiungiamo, con lo stesso criterio, anche la data.

Come puoi osservare, l'operatore "+" collega le due stringhe, mentre il metodo *strftime* riporta la data nel formato *%d-%m-%Y* ovvero giorno e mese a due cifre seguito dall'anno a quattro cifre. A questo indirizzo sono elencati tutti i possibili valori di formattazione della data. Infine aggiungiamo due stringhe di testo per dare maggior senso e scorrevolezza alla lettura:

Conclusioni

Oggi abbiamo imparato, nella pratica, come dichiarare variabili, come interagiscono controller e view, e come utilizzare i metodi *link_to e strftime*. Nel corso delle prossime lezioni approfondiremo quello che oggi abbiamo realizzato spiegandone il funzionamento teorico, ovvero specificheremo l'interazione tra le varie componenti di un'applicazione Rails all'interno del design pattern MVC e, passo passo, impareremo la sintassi Ruby.

Come ultima cosa, ti ricordo che puoi effettuare il download del codice completo relativo a questa lezione da questo link.

GUIDA RUBY ON RAILS: INDICE LEZIONI