

Soluzione del tema d'esame di Applicazioni Web

11 febbraio 2008

Docente Matteo Vaccari

Vogliamo realizzare un'applicazione per la gestione di un albergo. Un cliente può prenotare una o più camere per un certo numero di notti a partire da una certa data. Vogliamo censire nel nostro database le camere, i clienti e le prenotazioni. Ad esempio, il signor Rossi manda un fax per indicare che vuole 2 camere dal 10 luglio per 4 notti. L'impiegato che riceve il fax usa la nostra applicazione per prenotare le stanze (poniamo) 340 e 342 per il signor Rossi per quella data.

Il sistema deve registrare la prenotazione delle due stanze con una sola registrazione, e non come due volte la prenotazione di una camera.

0. (8pt) Disegnare il diagramma entità-relazioni per gestire tutte le entità, e il codice sql (o migrations) necessario per implementarlo.

```
          *      *              *      1
camera ----- prenotazione ----- cliente
```

Un cliente può fare n prenotazioni (anche in tempi diversi); una prenotazione fa riferimento a una o più camere, ma a un solo cliente. Una camera, nel tempo, è oggetto di più prenotazioni.

In alternativa si può spezzare la relazione multi-a-molti in due relazioni uno-a-molti, ma non è strettamente necessario per questo tema. In entrambi i casi è necessario creare una tabella di supporto per collegare camere e prenotazioni.

```
create_table :camere do |t|
  # usiamo :string perché non lo usiamo per fare aritmetica.
  # andava bene anche :integer
  t.column :number, :string
end

create_table :camere_prenotazioni
  # le chiavi esterne sono fondamentali per realizzare
  # la relazione multi-a-molti
  t.column :camera_id, :integer
  t.column :prenotazione_id, :integer
end

create_table :prenotazioni do |t|
  t.column :data_inizio, :date
  t.column :numero_notti, :integer
  # la chiave esterna, fondamentale per realizzare
  # la relazione uno-a-molti: a quale cliente
  # appartiene questa prenotazione?
  t.column :cliente_id, :integer
end
```

```

create_table :clienti do |t|
  t.column :nome, :string
  t.column :cognome, :string
end

```

1. (8pt) (a) Scrivere le classi di ActiveRecord, comprese le associazioni fra entità. (b) E' necessario validare che ogni camera abbia un numero assegnato dall'operatore, e che non ci siano due camere con lo stesso numero. (c) La classe Cliente deve avere un metodo `trova_prenotazione_in_data` che, dato l'ID di un cliente e una data, restituisce un array di tutte le camere prenotate da quel cliente per quella data.

```

class Camera < ActiveRecord::Base
  has_and_belongs_to_many :prenotazioni
  validates_presence_of :numero
  validates_uniqueness_of :numero
end

```

```

class Prenotazione < ActiveRecord::Base
  has_and_belongs_to_many :camere
  belongs_to :cliente
end

```

```

class Cliente < ActiveRecord::Base
  has_many :prenotazioni, :order => "data_inizio desc"

  # metodo statico; non appartiene a nessuna istanza specifica di Cliente
  def Cliente.trova_prenotazione_in_data(cliente_id, data_prenotazione)
    # trova il nostro cliente dato l'id
    cliente = Cliente.find(cliente_id)
    # sfrutta l'associazione has_many
    cliente.prenotazioni.find(:all, :conditions => ["data_inizio = ?",
                                                    data_prenotazione])

    # implementazione alternativa in una sola istruzione:
    # Prenotazione.find(:all, :conditions =>
    # ["data_inizio = ? and cliente_id = ?", data_prenotazione, cliente_id])
  end
end

```

2. (7pt) La url `http://example.org/prenotazioni/cliente/123` deve mostrare l'elenco di tutte le prenotazioni fatte dal cliente 123 ordinate per data, con una tabella di questo tipo

data	n. camere	
2008-02-11	3	edit
2008-01-03	1	edit
:	:	

Scrivere il controller e la view.

- Per realizzare il link “edit”, usare uno helper, non codice html.
- La tabella deve avere uno sfondo rosa, e una cornice spessa 0.5mm di colore arancione. Le celle con il numero di camere devono essere tutte testo bianco su sfondo nero. Scrivere il codice CSS in uno stylesheet separato. Collegare lo stile alla tabella usando opportuni attributi HTML.

```

class PrenotazioniController < ApplicationController
  ...
  def cliente
    Prenotazioni.find(:all,
      :conditions => ["cliente_id = ?", params[cliente_id]],
      :order => "data_inizio desc")
  end
end

<table class="prenotazioni">
<% for prenotazione in @prenotazioni %>
  <tr>
    <td><%= prenotazione.data_inizio.to_s %></td>
    <td class="numero_camere"><%= prenotazione.camere.size %></td>
    <td><%= link_to "edit", :action => "edit", :id => prenotazione %></td>
  </tr>
<% end %>
</table>

table.prenotazioni {
  background-color: pink;
  border: 0.5mm solid orange;
}
td.numero_camere {
  background-color: black;
  color: white;
}

```

3. (4pt) (a) Che cos'è il protocollo CGI? Si scriva un semplice esempio di script CGI in Ruby.

Vedi il diario <http://matteo.vaccari.name/aw/diario>, lezione 4, gli appunti indicati

4. (3pt) (a) Si descriva la struttura del messaggio di richiesta HTTP.

Vedi il diario <http://matteo.vaccari.name/aw/diario>, lezione 0, i lucidi