the Rails roundabout

Daniel Jackson
web app = data conversion

update request

tipster: share tips with friends

home search add entity logout

Reviewing Clover

Review
Enter your review here

Rating
One star is bad

Submit review

request

<table>
<thead>
<tr>
<th>id</th>
<th>rating</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>yummy!</td>
</tr>
</tbody>
</table>

database table

reviews

<table>
<thead>
<tr>
<th>id</th>
<th>by</th>
<th>content</th>
<th>rating</th>
<th>about</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>yummy!</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>neat</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

show request

tipster: share tips with friends

[Edit subject]

Name
Category Food
Rating ♦♦♦♦♦(3.3)

Add a review
Chloe Closure [Edit] (4 days ago)

My favorite food truck. Delicious vegetarian dishes, and relatively low ecoli count.

Ben Bitdiddle (4 days ago)

I like this place too. And they have really cool strategy for taking orders, with people outside the ...more

Alice Alert (4 days ago)

Yeah, sure, the food is good. But what about the atmosphere? Especially in winter when it's snowing.

response

<table>
<thead>
<tr>
<th>id</th>
<th>rating</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>yummy!</td>
</tr>
</tbody>
</table>
**tipster: share tips with friends**

**home search add entity logout**

**Reviewing Clover**

*Review*

Enter your review here

*Rating*

One star is bad

Submit review

---

**update request**

**show request**

### step 1: routing & params

**request**

<table>
<thead>
<tr>
<th></th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>2</td>
</tr>
<tr>
<td>rating</td>
<td>5</td>
</tr>
<tr>
<td>content</td>
<td>yummy!</td>
</tr>
</tbody>
</table>

**database table**

<table>
<thead>
<tr>
<th></th>
<th>by</th>
<th>content</th>
<th>rating</th>
<th>about</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>yummy!</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>neat</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**response**

<table>
<thead>
<tr>
<th></th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>2</td>
</tr>
<tr>
<td>rating</td>
<td>5</td>
</tr>
<tr>
<td>content</td>
<td>yummy!</td>
</tr>
</tbody>
</table>
routing

example
› request PUT /subjects/2/reviews/1
› in routes.rb
  put 'subjects/:subject_id/reviews/:id' => 'reviews#update'
› call
  class ReviewsController ...
    def update
      subject_id = params[:subject_id]
      review_id = params[:id]
      new_content = params[:content]
      ...
    end
what is routing giving us?

separation of concerns
› client’s URL versus developer’s method name
› bidirectional: path helpers to create links

resourceful routing
› standard patterns

*instead of explicit route:*
put 'subjects/:subject_id/reviews/:id' => 'reviews#update'

can use resource abbreviations:

```ruby
resources :subjects do
  resources :reviews end
```
what is params hash giving us?

abstraction
› over HTTP method and incoming representation

params[:id]
› get value of \textit{id} bound in URL pattern
› get value associated with key \textit{id} in query string
› get value associated with key \textit{id} in submitted form
update request

show request

step 2: accessing model objects
defining the model

three step process
› define an empty model class
› define a “migration” to create a table
› run the migration with `rake db:migrate`

why not just write the schema in SQL?
› Rails can reverse a migration if you change your mind
› a migration can update the data in the database too
a model class

```ruby
class Review < ActiveRecord::Base
 end
```

where do you put this?
in app/models/review.rb

where are the instance variables?
› remember that Ruby doesn’t need decls
› methods are generated from schema by Rails
› this is what extending ActiveRecord does
a migration

class CreateReviews < ActiveRecord::Migration
  def change
    create_table :reviews do |t|
      t.integer :rating
      t.text :content
      t.timestamps
    end
  end
end

where's the id?
  - primary column called id added implicitly

where's about, by?
  - we'll see these later when we talk about associations

what's t.timestamps?
  - adds two columns created_at, updated_at

where do you put this?
in db/migrate, starting with template (generates file name with timestamp):
  rails generate migration CreateReviews

note naming conventions
  model class: Review
  table: reviews
model methods

create a new object
› r = Review.new

set and get fields of object
› r.content = "yummy"
› c = r.content

save object to database
› r.save

get objects from database
› reviews = Review.all
› r = Review.find_by_id(id)

updating a review

id = params[:id]
new_content = params[:content]
@review = Review.find_by_id(id)
@review.content = new_content
@review.save

why @review?
can be local variable instead
but if want to view using template,
use instance variable in controller
update request

request

<table>
<thead>
<tr>
<th>id</th>
<th>rating</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>yummy!</td>
</tr>
</tbody>
</table>

database table

<table>
<thead>
<tr>
<th>id</th>
<th>by</th>
<th>content</th>
<th>rating</th>
<th>about</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>yummy!</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>neat</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

response

<table>
<thead>
<tr>
<th>id</th>
<th>rating</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>yummy!</td>
</tr>
</tbody>
</table>

step 3: displaying model objects
class ReviewsController < ApplicationController
  def index
    @reviews = Review.all
  end
end

what does this do?
› looks up all rows in table called reviews
› creates an object of model class Review for each
› creates an array of these objects
› binds this array to the instance variable @reviews
› implicitly renders a view template

where do you put this?
in app/controllers/ReviewsController.rb

which view is rendered?
app/views/reviews/index.html.erb
view template

syntax
› standard HTML
› Ruby code `<% %>`
› Ruby exprs `<%= %>`

note
› instance var from controller action
› path helper defined by route

```html
<h1>Listing Reviews</h1>
<table>
  <tr>
    <th>About</th>
    <th>Rating</th>
    <th>Content</th>
    <th></th>
  </tr>
  <% @reviews.each do |review| %>
    <tr>
      <td><%= review.about.name %></td>
      <td><%= review.rating %></td>
      <td><%= review.content %></td>
      <td><%= link_to "Edit", edit_review_path(review) %></td>
    </tr>
  <% end %>
</table>
```
from update to index

suppose we design this flow
  › when review is updated, 
    go back to index page of all reviews

how is this implemented?
  › controller method edit responds with form
  › controller method update updates the review row, 
    and instead of rendering, redirects to index

```ruby
def update
  @review = Review.find_by_id(params[:id])
  @review.content = params[:content]
  if @review.save
    redirect_to action: 'index'
  else
    render "edit"
  end
end
```
how to grasp all this?

what makes Rails challenging to learn
› many concepts (although mostly easy ones)
› lots of small details
› naming conventions & implicit calls

what I recommend
› generate an app with scaffolding and play around with it
› look at the demo apps we created for you
› read the Rails guides at [http://guides.rubyonrails.org](http://guides.rubyonrails.org)
› use StackOverflow