

Rails from the Ground Up!

with Jared Richardson
RoleModel Software

Rails

Part Three

Misc Topics

Request Lifecycle

Browser

GET / OR

GET index.html

Rails Server

Listens on port 3000

Reads request

Match / to board/index

Create a new params hash

Stock params hash with data from the URL & browser

Call board.index(params) # class.method(parameters)

- Use params

- Run code. Save data in @ vars if you need to use it in views

- Call views/board/index.html.erb

- Runs the <% Ruby code %> Uses the @ vars from index

Returns a string containing HTML to the browser

Web Resources

RubyOnRails.org

ApiDock.com

PragProg.com

PeepCode.com

RailsCasts.com

Ruby-toolbox.com

Cukes.info

<http://webchat.freenode.net> => rubyonrails

<http://rspec.info/>

Blog

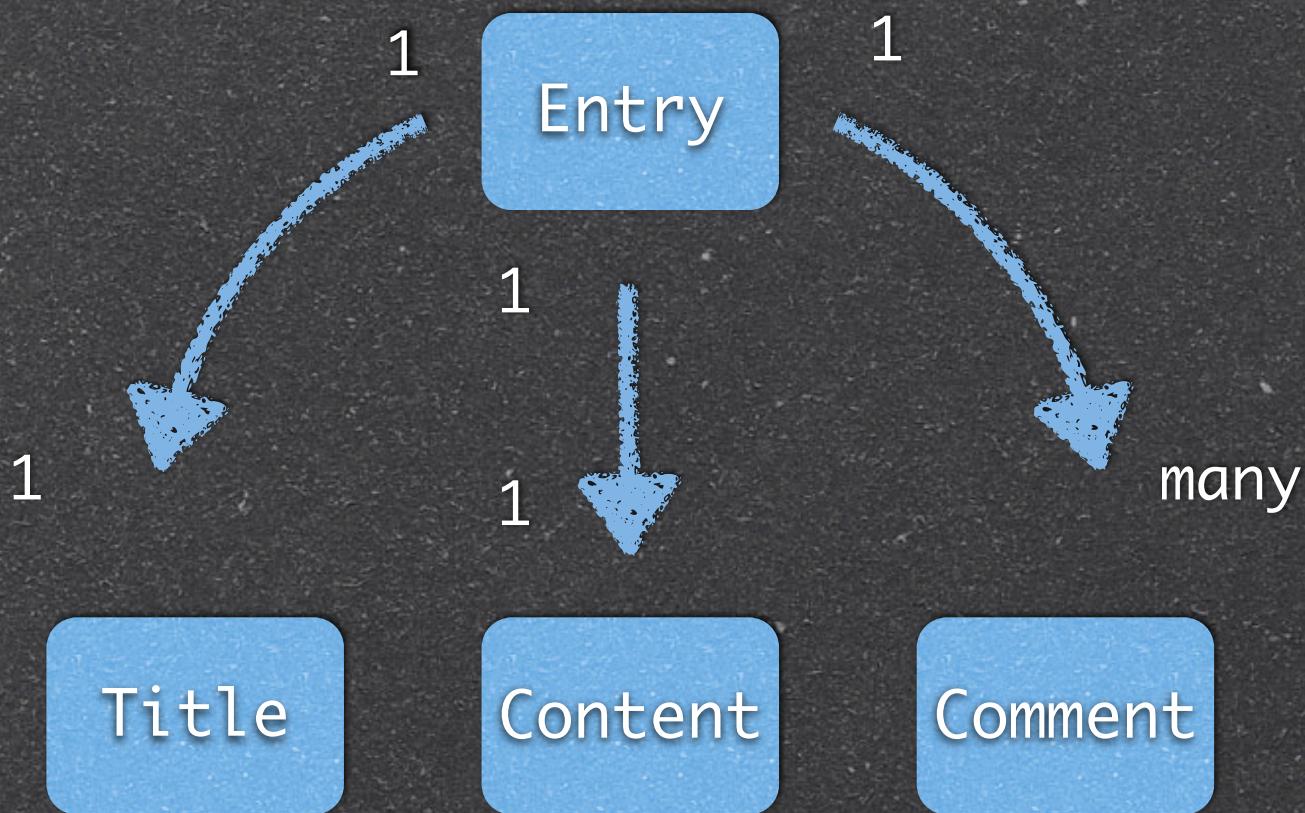
Entry

- has one title
- has one body (content)
- has many comments

Comments

- belong to an article

Blog



Ready?

```
rails new blog
```

```
cd blog
```

```
rails generate scaffold Blog_Entry  
name:string title:string body:text
```

Examine the Model

rails consoleBack to the Console

```
e = Entry.new
```

What do you see?

```
ActiveRecord::StatementInvalid: Could not find table 'entries'
```

```
rake db:migrations
```

Save It

e.save!

What do you see?

```
1.9.3-p0 :008 > e.save
  (0.1ms)  begin transaction
  SQL (0.8ms)  INSERT INTO "entries" ("body", "created_at", "name", "title"
, "updated_at") VALUES (?, ?, ?, ?, ?)  [["body", nil], ["created_at", Sun,
19 Feb 2012 01:45:37 UTC +00:00], ["name", nil], ["title", nil], ["updated
_at", Sun, 19 Feb 2012 01:45:37 UTC +00:00]]
  (1.3ms)  commit transaction
=> true
```

Are nils Valid?

Let's fix it

Active Record validations

<http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html>

validates

```
class Entry < ActiveRecord::Base  
  validates :name,  :presence => true  
  validates :title, :presence => true,  
            :length => { :minimum => 5 }  
end
```

reload!

`rails console` command

Reloads code on disk

Does NOT re-load pre-existing
instances (in memory variables)

Make new ones

Recreate & save

```
e = Entry.new
```

```
e.save
```

What's Wrong?

```
1.9.3-p0 :032 > e.save
  (0.3ms)  begin transaction
  (0.1ms)  rollback transaction
=> false
1.9.3-p0 :033 > █
```

1) Returned False

```
1.9.3-p0 :032 > e.save
  (0.3ms)  begin transaction
  (0.1ms)  rollback transaction
=> false
1.9.3-p0 :033 > 
```

2) It's Not This!

```
1.9.3-p0 :032 > e.save
  (0.3ms) begin transaction
  (0.1ms) rollback transaction
=> false
1 0 2 n0 .022 ~ □
```

```
1.9.3-p0 :023 > e.save
  (0.1ms) begin transaction
  (0.4ms) UPDATE "entries" SET "name" = 'My First Blog Entry!
', "updated_at" = '2012-02-19 03:01:10.240021' WHERE "entries".
"id" = 4
  (1.1ms) commit transaction
=> true
```

Saving

```
# runs with no feedback on errors
```

```
e.save
```

```
# runs and throws exceptions on errors
```

```
e.save!
```

errors

```
e = Entry.new
```

```
e.valid? # or e.save or e.save!
```

```
errors = e.errors
```

```
e.errors[:name]
```

```
errors.full_messages
```

error listing as a hash

```
errors.full_messages.each{ | msg |  
  puts msg  
}
```

Fix It!

```
e.errors[:name] # shows name's errors  
e.name="My first blog entry!"  
e.valid?  
e.errors[:name]  
:)
```

What Else?

```
e.errors
```

```
e.errors[:title]
```

```
["can't be blank",
```

```
"is too short (minimum is 5 characters)"]
```

```
e.title = "A Freshly Minted Entry!"
```

```
e.valid?
```

```
TRUE!
```

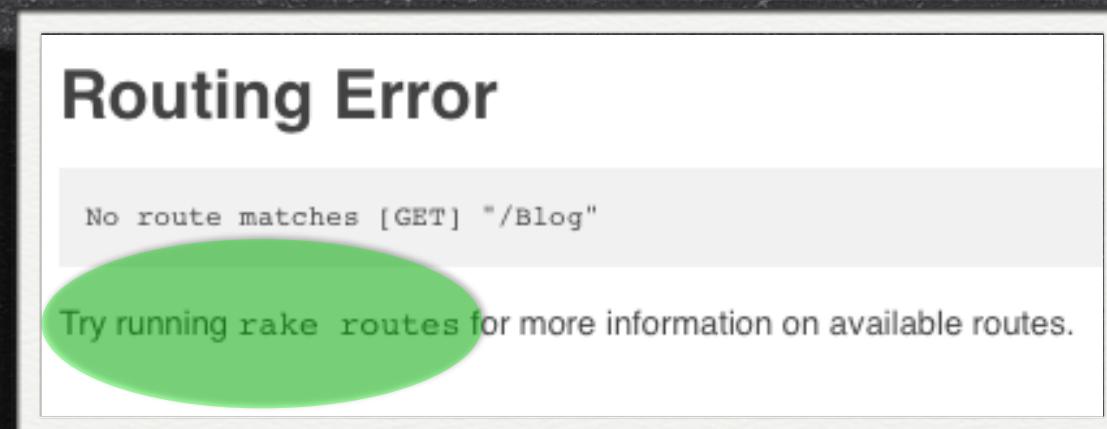
Clean Up

```
# wipe out and reset your database  
rake db:reset  
  
rails console  
  
# now recreate a blog entry  
e = Entry.new
```

Does It Run?

rails server

<http://localhost:3000/BlogRemember>
Comments?has many



Comments

```
rails generate model Comment
```

```
  author:string title:string body:text
```

```
rake routes
```

```
Load the comments page
```

Create a Link

open app/models/*.rb

```
1 class Comment < ActiveRecord::Base  
2   belongs_to :entry  
3 end  
4
```

```
1 class Entry < ActiveRecord::Base  
2   validates :name, :presence => true  
3   validates :title, :presence => true,  
4   :length => { :minimum => 5 }  
5   has_many :comments  
6 end
```

Associations

has_one

has_many

belongs_to

has_and_belongs_to_many

belongs_to in the model w/ the foreign
keyUnit Tests

Now, you have to tell Rails where your actual home page is located. Open the file `config/routes.rb` in your editor. This is your application's *routing file* which holds entries in a special DSL (domain-specific language) that tells Rails how to connect incoming requests to controllers and actions. This file contains many sample routes on commented lines, and one of them actually shows you how to connect the root of your site to a specific controller and action. Find the line beginning with `root :to` and uncomment it. It should look something like the following:



```
Blog::Application.routes.draw do  
  
  #...  
  # You can have the root of your site routed with "root"  
  # just remember to delete public/index.html.  
  root :to => "home#index"
```

The `root :to => "home#index"` tells Rails to map the root action to the home controller's index action.

Now if you navigate to <http://localhost:3000> in your browser, you'll see Hello, Rails!.

6.2 Adding a Link

To hook the posts up to the home page you've already created, you can add a link to the home page. Open `app/views/home/index.html.erb` and modify it as follows:



```
<h1>Hello, Rails!</h1>
<%= link_to "My Blog", posts_path %>
```

The `link_to` method is one of Rails' built-in view helpers. It creates a hyperlink based on text to display and where to go – in this case, to the path for posts.

Random Tips & Tricks

Some Ruby

Some Rails

time trick

rails c

```
time_string = "2010-07-21 11:52:31"
```

```
helper.time_ago_in_words(time_string) # =>  
about 5 hours
```

```
year = 1972
puts case year
    when 1970..1979: "Seventies"
    when 1980..1989: "Eighties"
    when 1990..1999: "Nineties"
end
```

<http://www.rubyinside.com/21-ruby-tricks-902.html>

ternary

```
puts x == 10 ? "x is ten" : "x is not ten"
```

rspec

```
01 describe "My Cool library" do
02   before do
03     @cool = Cool.new
04   end
05
06   it "should be full of penguins." do
07     @cool.get_penguins!
08     @cool.penguin_count.should == 10
09   end
10
11   it "should melt if it gets too warm"
12 end
```

erb vs haml

```
1 <%= print_date %>
2 <%= current_user.address %>
3
4 <%= current_user.email %>
5 <%= h current_user.bio %>
```

```
1 #profile
2   .left.column
3     #date= print_date
4     #address= current_user.address
5   .right_column
6     #email= current_user.email
7     #bio= h(current_user.bio)
```

Where to Dig In

- Ruby knowledge (koans, many)
- Rails blog example (on RubyOnRails.com)

Good Luck!