

# Vital Rails

Ruby on Rails Training

1



2

## **ActionController and ActionView**

3

review

4

M      V      C

5

M      C ↔ V

6

M odel

Represents the  
business model  
the "things" in  
your application

7

C ontroller

stage director in  
your application.  
What goes where

8

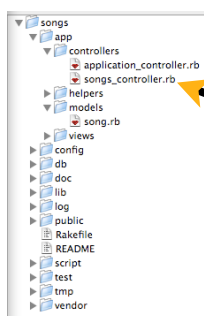
V iew

How things are  
represented in  
the application

9

let's make a list

10



open the  
controller file

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController  
end
```

11

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController  
  
  def index  
    render :text => "Hello World"  
  end  
  
end
```

12

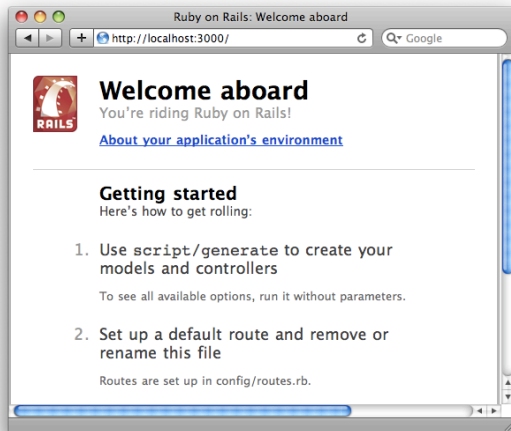
```
$ cd vital_ruby
$ cd src
$ cd songs
$ script/server
=> Booting Mongrel
=> Rails 2.3.2 application starting
on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
```

13

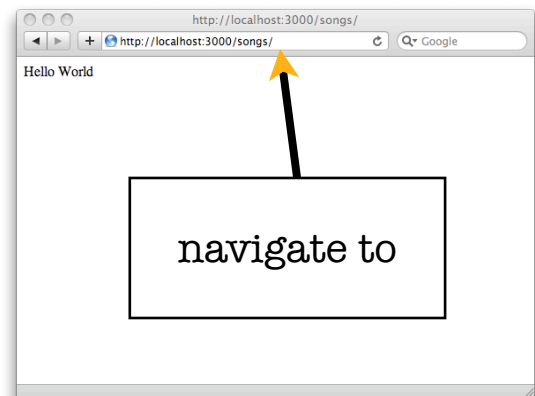
```
$ cd
$ cd
$ cd songs
$ script/server
=> Booting Mongrel
=> Rails 2.3.2 application starting
on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
```

server now listening on:  
**http://localhost:3000**

14



15



16

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController
  def index
    @songs = Song.find :all
    render :text =>
      @songs.collect { |s| s.name }
  end
end
```

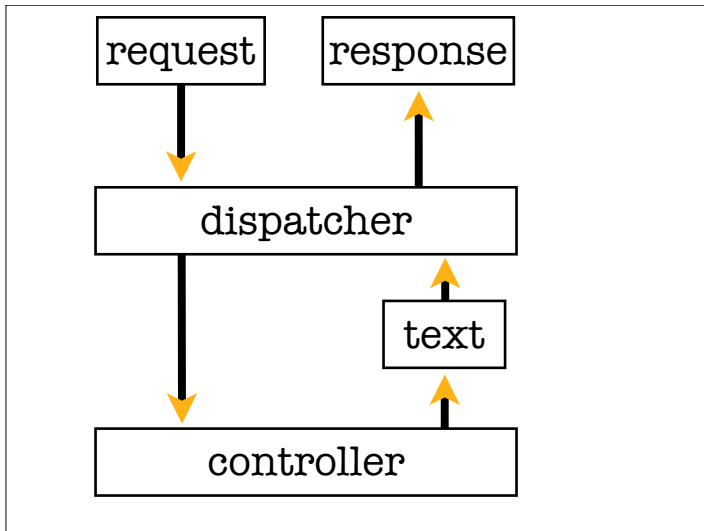
17

http://localhost:3000/songs/

AllentownUptown GirlAn Innocent ManAngry Young ManA Matter of TrustOnly the Good Die YoungBaby GrandBig ShotHonestyGoodnight Saigon

refresh

18



19

we want a real view

20

app/controllers/songs\_controller.rb

```

class SongsController < ApplicationController
  def index
    @songs = Song.find :all
  end
end
  
```

remove call to render

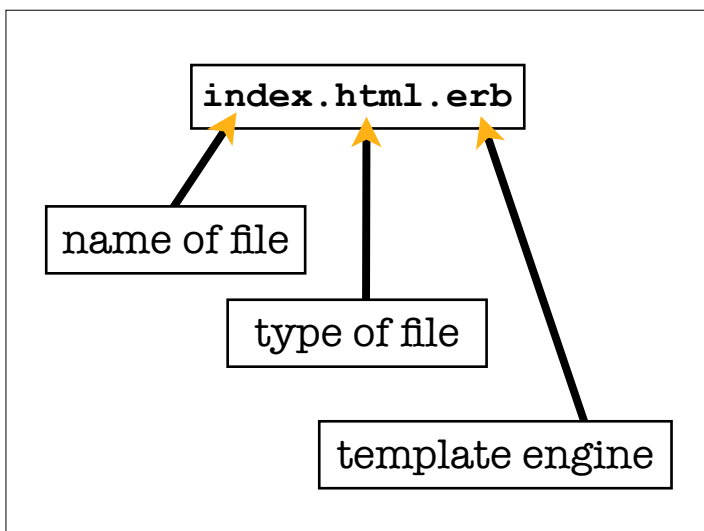
21

create this file

```

$ touch app/views/songs/index.html.erb
  
```

22



23

name of file

usually the same as the controller

24

type of file

this can be  
html, xml, etc ...

25

template engine

what is building the  
template?  
haml, erb,  
builder, etc...

26

app/views/songs/index.html.erb

```
<html>
<body>
  <h1>Songs</h1>
  <ul>
    <% @songs.each do |song| %>
      <li>Name: <%= song.name %></li>
    <% end %>
  </ul>
</body>
</html>
```

27



refresh

28

request

response

dispatcher

controller

view

29

app/controllers/songs\_controller.rb

```
def index
  @songs = Song.find :all
end
```

instance variables are  
available to the views

app/views/songs/index.html.erb

```
<% @songs.each do |song| %>
  <li>Name: <%= song.name %></li>
<% end %>
```

30

erb

app/views/songs/index.html.erb

```
<html>
<body>
  <h1>Songs</h1>
  <ul>
    <% @songs.each do |song| %>
      <li>Name: <%= song.name %></li>
    <% end %>
  </ul>
</body>
</html>
```

`<% %>`  
interpret but do not print

31

erb

`<%= %>`  
interpret and print the result  
(call `.to_s` on result)

```
<html>
<body>
  <h1>Songs</h1>
  <ul>
    <% @songs.each do |song| %>
      <li>Name: <%= song.name %></li>
    <% end %>
  </ul>
</body>
</html>
```

32

controller magic

if not told otherwise,  
controller will render  
a view with the same  
name as itself

33



34

we will cover URL  
magic in a bit,  
hang tight

35

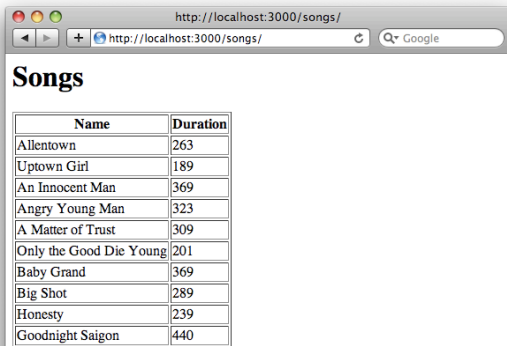
app/views/songs

refactoring:  
use a table

```
<html>
<body>
  <h1>Songs</h1>
  <table border=1>
    <tr>
      <th>Name</th>
      <th>Duration</th>
    </tr>
    <% @songs.each do |song| %>
      <tr>
        <td><%= song.name %></td>
        <td><%= song.duration %></td>
      </tr>
    <% end %>
  </table>
</body>
</html>
```

(make a designer cry)

36



Name	Duration
Allentown	263
Uptown Girl	189
An Innocent Man	369
Angry Young Man	323
A Matter of Trust	309
Only the Good Die Young	201
Baby Grand	369
Big Shot	289
Honesty	239
Goodnight Saigon	440

refresh

37

lets show an individual song

38

app/controllers/songs\_controller.rb

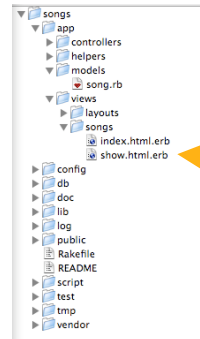
```
class SongsController
  def index
    @songs = Song.find :all
  end

  def show
    @song = Song.find(params[:id])
  end
end
```

we get the id from the parameters and pass it to find



39



create this file

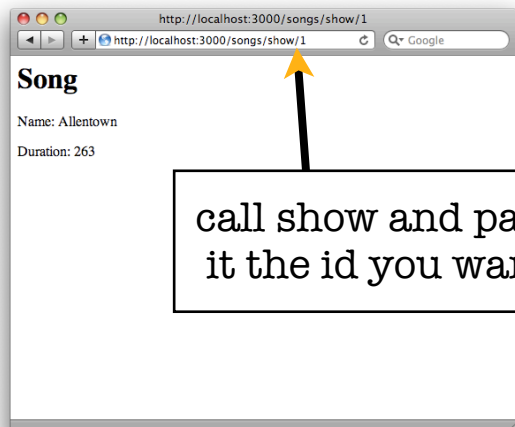
```
$ touch app/views/songs/show.html.erb
```

40

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <p>Name: <%= @song.name %></p>
  <p>Duration: <%= @song.duration %></p>
</body>
</html>
```

41



call show and pass it the id you want

42

"... but I don't know  
the id"

"And you **SHOULDN'T**  
know the id!"

43

app/views/songs/index.html.erb

name of link

the url

```
...  
<% @songs.each do |song| %>  
  <tr>  
    <td><%= song.name %></td>  
    <td><%= song.duration %></td>  
    <td><%= link_to "show", song_path(song.id) %>  
  </tr>  
<% end %>  
...
```

44

app/views/songs/index.html.erb

we get the id from  
the song in order to  
build the url

`song_path(song.id)`

45

app/views/songs/show.html.erb

the url

```
<html>  
  <body>  
    <h1>Song</h1>  
    <p>Name: <%= @song.name %></p>  
    <p>Duration: <%= @song.duration %></p>  
    <p><%= link_to "Back", songs_url %>  
  </body>  
</html>
```

name of link

46

yes, we will get  
to URL's

47

http://localhost:3000/songs

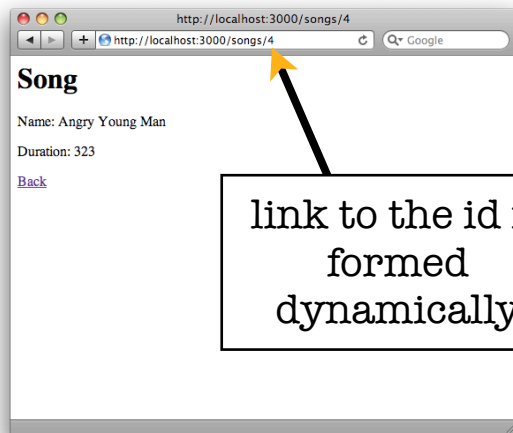
Songs

Name	Duration	
Allentown	263	<a href="#">show</a>
Uptown Girl	189	<a href="#">show</a>
An Innocent Man	369	<a href="#">show</a>
Angry Young Man	323	<a href="#">show</a>
A Matter of Trust	309	<a href="#">show</a>
Only the Good Die Young	201	<a href="#">show</a>
Baby Grand	369	<a href="#">show</a>
Big Shot	289	<a href="#">show</a>
Honesty	239	<a href="#">show</a>
Goodnight Saigon	440	<a href="#">show</a>

link to show

48





49

\* take a deep breath \*

50

## LAB 5

Initial Rails Application

51

## Conference Submission

- take your submit\_it application and create a list of talks
- create a show page for talks
- link them together

52

lets create some  
objects

53

two step create  
process

get the create form

fill out and submit form

54

app/controllers/songs\_controller.rb

```
class SongsController
  def index
    @songs = Song.all
  end

  def show
    @song = Song.find(params[:id])
  end

  def new
    @song = Song.new
  end
end
```

add a **new** method  
to the controller

55

app/controllers/songs\_controller.rb

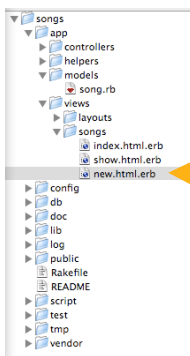
```
class SongsController
  ...

  def new
    @song = Song.new
  end

  def create
    render :text => params.inspect
  end
end
```

add a placeholder  
create method  
simply printing out  
the parameters

56



create this  
file

```
$ touch app/views/songs/new.html.erb
```

57

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song_name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'song_duration' %><br/>
    <%= f.text_field :duration %></p>

    <p>
      <%= submit_tag 'Add Song' %>
      or <%= link_to 'cancel', songs_path %>
    </p>
  <% end -%>
</body>
</html>
```

58

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% (form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song_name' %><br/>
    <%= f.text_field :name %></p>

  </%>
</body>
</html>
```

form\_for takes the name  
of the object (AR Class)  
being created

59

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% (form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song_name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'duration' %><br/>
    <%= f.text_field :duration %></p>

    <p>
      <%= submit_tag 'Add Song' %>
      or <%= link_to 'cancel', songs_path %>
    </p>
  <% end -%>
</body>
</html>
```

the url the form  
should submit to

60

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'song duration' %><br/>
    <%= f.text_field :duration %></p>

  </p>
</form>
</body>
</html>
```

then tell it what type  
of field goes with what  
attribute

61

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'song duration' %><br/>
    <%= f.text_field :duration %></p>

  </p>
</form>
</body>
</html>
```

simply creates a  
label for this field

62

app/views/songs/show.html.erb

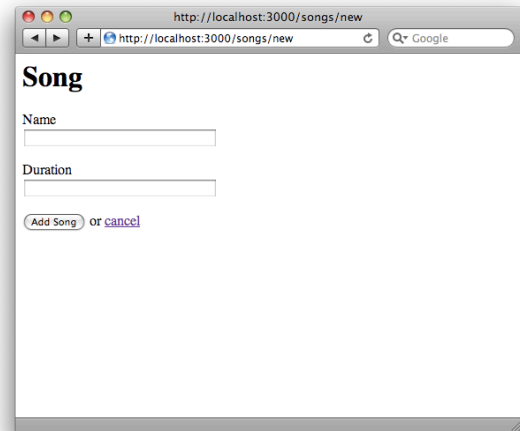
```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'song duration' %><br/>
    <%= f.text_field :duration %></p>

    <p>
      <%= submit_tag 'Add Song' %>
      or <%= link_to 'cancel', songs_path %>
    </p>
  </form>
</body>
</html>
```

creates the submit  
button with that text

63



64

Source of <http://localhost:3000/songs/new>

```
<html>
<body>
  <h1>Song</h1>

  <form action="/songs" method="post">
    <div style="margin:0;padding:0">
      <input name="authenticity_token" type="hidden"
      value="mx6qe9iFxcw5QxAqt50gUCBwWI2vNiQ7Js8BU99De58=" /></div>
      <p><label for="song_name">Name</label><br/>
      <input id="song_name" name="song[name]" size="30"
      type="text" /></p>

    <p>
      <input type="submit" value="Add Song" />
      or <a href="/songs" />cancel</a>
    </p>
  </form>
```

authenticity token  
that is generated  
behind the scenes

65

Source of <http://localhost:3000/songs/new>

```
<label for="song_name">Name</label>
<input id="song_name"
  name="song[name]"
  size="30"
  type="text" />
```

notice the syntax  
here.

66

http://localhost:3000/songs/new

## Song

Name  
The Warrior's Code

Duration  
150

Add Song or cancel

fill in values and  
submit

67

http://localhost:3000/songs

```
{
  "commit"=>"Add Song",
  "authenticity_token"=>"mx6qe9iFxcwSQxAqt50gUCBwWI2vNiq7JsBBU99De58=",
  "action"=>"create", "controller"=>"songs", "song"=>{
    "name"=>"The Warrior's Code", "duration"=>"150"}
}
```

parameter  
contents

68

Source of <http://localhost:3000/songs/new>

```
...
"song"=>{"name"=>"The Warrior's Code",
"duration"=>"150"}
```

song comes in as its  
own hash

69

take a moment  
to review

70

remember this?

```
>> song = Song.create :name =>
'Allentown', :duration => 263
=> #<Song id: 12, name: "Allentown",
duration: 263>
>> song.save
=> true
```

71

create takes a hash

```
>> song = Song.create :name =>
'Allentown', :duration => 263
=> #<Song id: 12, name: "Allentown",
duration: 263>
>> song.save
=> true
```

72

**save** returns true or false depending on validations

```
>> song = Song.create :name =>
'Allentown', :duration => 263
=> #<Song id: 12, name: "Allentown",
duration: 263>
>> song.save
=> true
```

73

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController
  ...

  def create
    @song = Song.create(params[:song])

    if @song.save
      redirect_to songs_url
    else
      render :new
    end
  end
end
```

74

**redirect\_to(url)**

Will redirect to an action in this controller

**render(form)**

renders the form requested

75

**NOTE: This means, all the variables need to be setup**

**render(form)**

renders the form requested

76

http://localhost:3000/songs/new

Song

Name  
The Warrior's Code

Duration  
150

Add Song or cancel

77

http://localhost:3000/songs

Songs

Name	Duration	
Allentown	263	<a href="#">show</a>
Uptown Girl	189	<a href="#">show</a>
An Innocent Man	369	<a href="#">show</a>
Angry Young Man	323	<a href="#">show</a>
A Matter of Trust	309	<a href="#">show</a>
Only the Good Die Young	201	<a href="#">show</a>
Baby Grand	369	<a href="#">show</a>
Big Shot	289	<a href="#">show</a>
Honesty	239	<a href="#">show</a>
Goodnight Saigon	440	<a href="#">show</a>
The Warrior's Code	150	<a href="#">show</a>

78

http://localhost:3000/songs/new

## Song

Name  
Citizen C.I.A.

Duration

or [cancel](#)

add without a duration

79

http://localhost:3000/songs/new

## Song

2 errors prohibited

There were problems with the following fields:

- Duration can't be blank
- Duration is not a number

Song name  
Citizen C.I.A.

Song duration

or [cancel](#)

error messages we created earlier

80

http://localhost:3000/songs/new

## Song

2 errors prohibited

There were problems with the following fields:

- Duration can't be blank
- Duration is not a number

Song name  
Citizen C.I.A.

Song duration

or [cancel](#)

automatically generated

81

the 'flash'

Rails uses 'the flash' for messages back and forth. Quick persistent memory

82

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController
  ...

  def create
    @song = Song.create(params[:song])

    if @song.save
      flash[:success] = "Added a new song successfully"
      redirect_to songs_url
    else
      render :new
    end
  end
end
```

83

http://localhost:3000/songs

## Song

2 errors prohibited

There were problems with the following fields:

- Duration can't be blank
- Duration is not a number

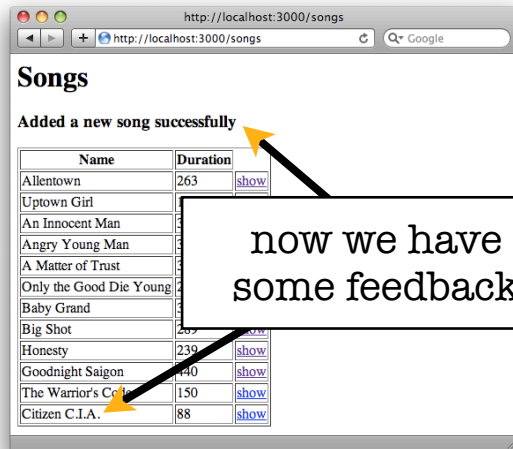
Song name  
Citizen C.I.A.

Song duration  
88

or [cancel](#)

add in a good number here

84



now we have  
some feedback

view templates

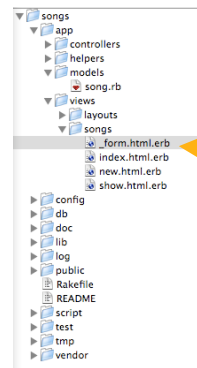
app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <p><%= label_tag 'song_name' %><br/>
    <%= f.text_field :name %></p>

    <p><%= label_tag 'song_duration' %><br/>
    <%= f.text_field :duration %></p>

    <p>
      <%= submit_tag 'Add Song' %>
      or <%= link_to 'cancel', songs_path %>
    </p>
  <% end -%>
</body>
</html>
```

take this block  
and cut it



create this  
file

```
$ touch app/views/songs/_form.html.erb
```

app/views/songs/\_form.html.erb

```
<p><%= label_tag 'song_name' %><br/>
<%= f.text_field :name %></p>

<p><%= label_tag 'song_duration' %><br/>
<%= f.text_field :duration %></p>

<p>
  <%= submit_tag 'Add Song' %>
  or <%= link_to 'cancel', songs_path %>
</p>
```

paste the  
contents in here

app/views/songs/show.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song, :url => songs_path do |f| -%>
    <%= render :partial => 'form',
      :locals => { :f => f } %>
  <% end -%>
</body>
</html>
```

add this in place  
of the form

http://localhost:3000/songs/new

**Song**

Song name

Song duration

or [cancel](#)

make sure it still works :-)

91

two step create  
process

remember this?

get the create form

fill out and submit form

92

app/controllers/songs\_controller.rb

when we bring up the form,  
it should be populated

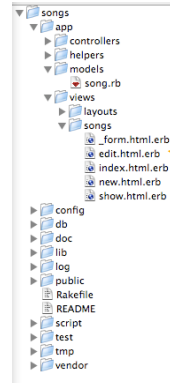
```
class SongController
  ...

  def new
    @song = Song.new
  end

  def edit
    @song = Song.find(params[:id])
  end

end
```

93



create this  
file

```
$ touch app/views/songs/_form.html.erb
```

94

app/views/songs/edit.html.erb

```
<html>
<body>
  <h1>Song</h1>
  <%= error_messages_for :song %>
  <% form_for :song,
    :url => edit_song_path(@song.id) do |f| -%>
    <%= render :partial => 'form',
      :locals => { :f => f } %>
  <% end -%>
</body>
</html>
```

similar, except we are  
posting to update

95

so what do we do with  
the new information?

96



app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController
  ...

  def update
    @song = Song.find(params[:id])
    result = @song.update_attributes(params[:song])

    if result
      flash[:success] =
        "Edited #{@song.name} successfully"
      redirect_to songs_url
    else
      render :edit
    end
  end
end
```

again, similar to  
create

97

app/controllers/songs\_controller.rb

```
class SongsController
  ...

  def update
    @song = Song.find(params[:id])

    if @song.update_attributes(params[:song])
      flash[:success] =
        "Edited #{@song.name} successfully"
      redirect_to songs_url
    else
      render :edit
    end
  end
end
```

slight inline  
refactoring

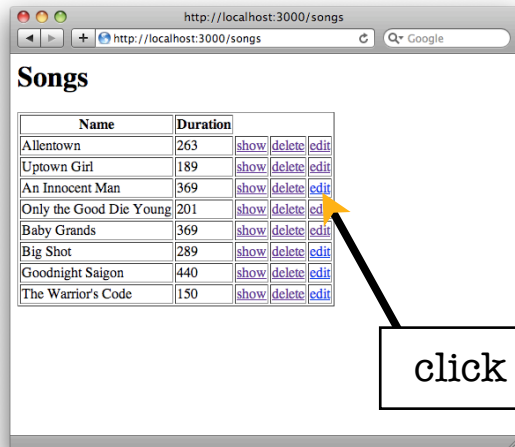
98

app/views/songs/edit.html.erb

```
...
<tr>
  <td><%= song.name %></td>
  <td><%= song.duration %></td>
  <td><%= link_to "show", song_path(song.id) %></td>
  <td><%= link_to "edit", edit_song_path(song.id) %></td>
</tr>
```

add an edit link  
to the list

99



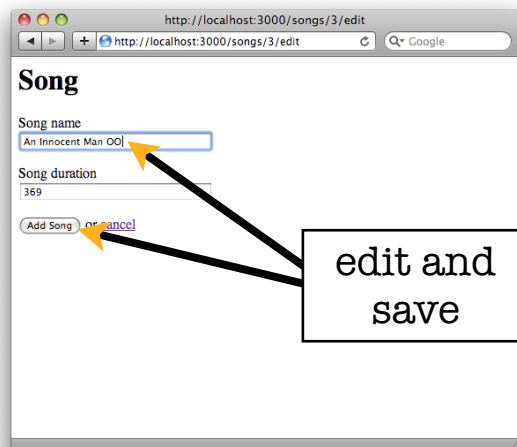
http://localhost:3000/songs

### Songs

Name	Duration			
Allentown	263	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Uptown Girl	189	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
An Innocent Man	369	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Only the Good Die Young	201	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Baby Grands	369	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Big Shot	289	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Goodnight Saigon	440	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
The Warrior's Code	150	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>

click

100



http://localhost:3000/songs/3/edit

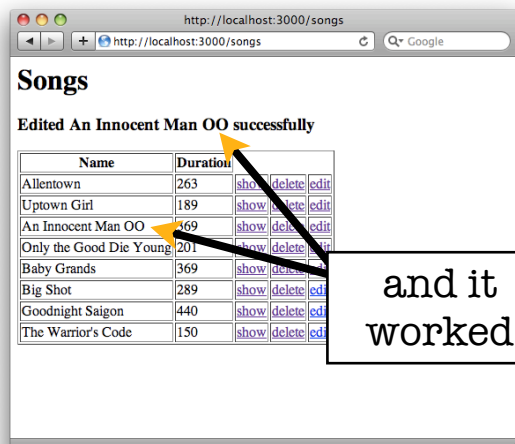
### Song

Song name

Song duration

edit and  
save

101



http://localhost:3000/songs

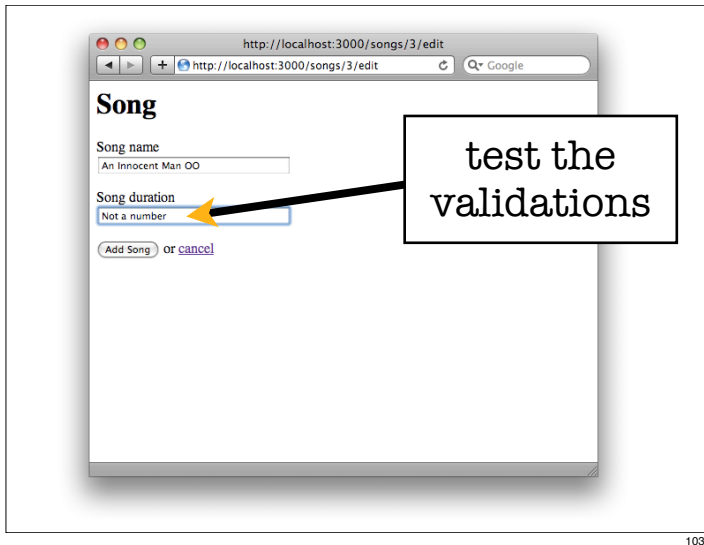
### Songs

Edited An Innocent Man OO successfully

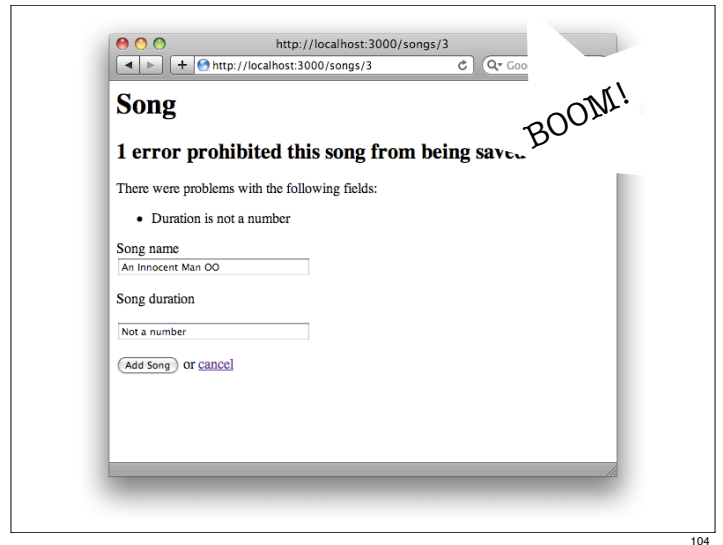
Name	Duration			
Allentown	263	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Uptown Girl	189	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
An Innocent Man OO	369	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Only the Good Die Young	201	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Baby Grands	369	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Big Shot	289	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
Goodnight Saigon	440	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>
The Warrior's Code	150	<a href="#">show</a>	<a href="#">delete</a>	<a href="#">edit</a>

and it  
worked

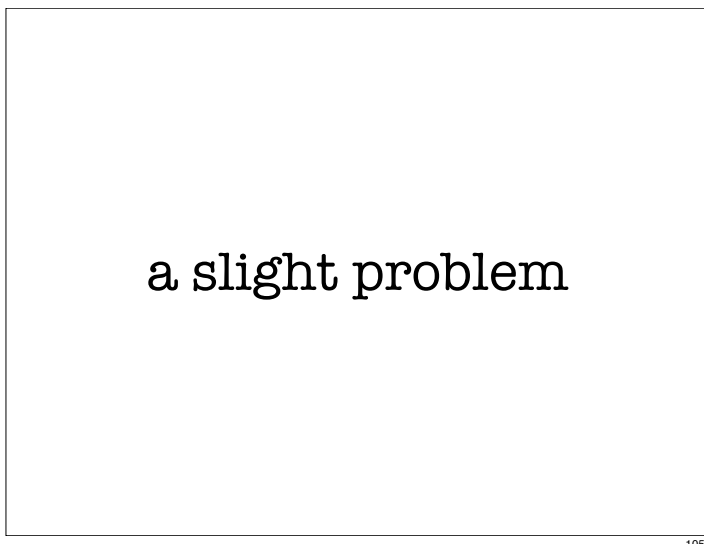
102



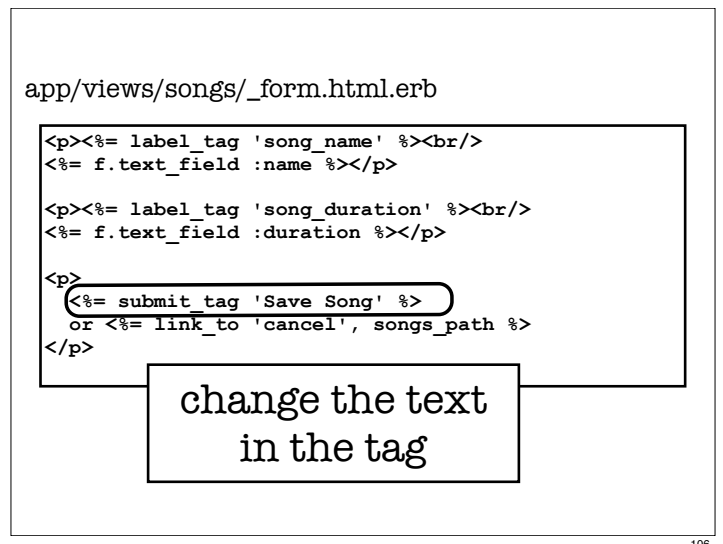
103



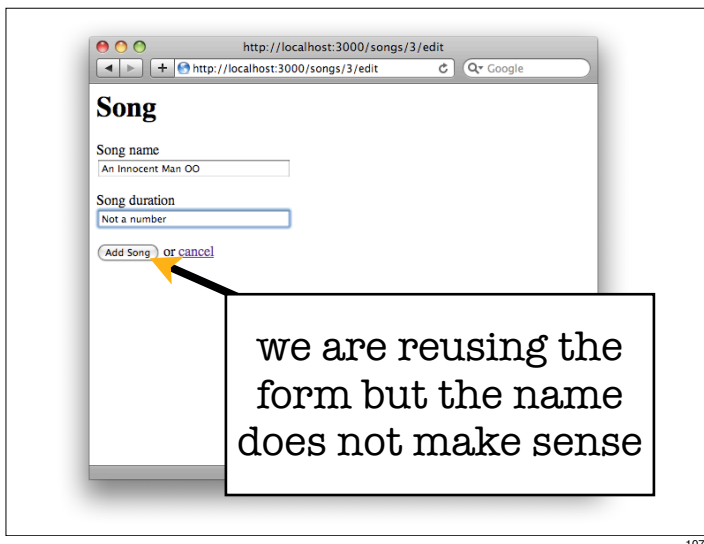
104



105



106



107



108

app/views/songs/index.html.erb

add this to  
the table

```
...  
<% @songs.each do |song| %>  
  <tr>  
    <td><%= song.name %></td>  
    <td><%= song.duration %></td>  
    <td><%= link_to "show", song_path(song.id) %>  
    <td><%= link_to "delete",  
      { :action => "destroy", :id => song.id },  
      :method => :delete %></td>  
  </tr>  
<% end %>  
...
```

109

app/controllers/songs\_controller.rb

```
class SongsController < ApplicationController  
  ...  
  
  def destroy  
    song = Song.find(params[:id])  
    song.destroy  
  
    flash[:success] = "Song: #{song.name} was deleted"  
    redirect_to songs_url  
  end  
  
end
```

110

http://localhost:3000/songs

### Songs

Name	Duration	
Allentown	263	<a href="#">show</a> <a href="#">delete</a>
Uptown Girl	189	<a href="#">show</a> <a href="#">delete</a>
An Innocent Man	369	<a href="#">show</a> <a href="#">delete</a>
Angry Young Man	323	<a href="#">show</a> <a href="#">delete</a>
Only the Good Die Young	201	<a href="#">show</a> <a href="#">delete</a>
Baby Grand	369	<a href="#">show</a> <a href="#">delete</a>
Big Shot	289	<a href="#">show</a> <a href="#">delete</a>
Goodnight Saigon	440	<a href="#">show</a> <a href="#">delete</a>
The Warrior's Code	150	<a href="#">show</a> <a href="#">delete</a>

111

http://localhost:3000/songs

### Songs

Song: Angry Young Man was deleted

Name	Duration	
Allentown	263	<a href="#">show</a> <a href="#">delete</a>
Uptown Girl	189	<a href="#">show</a> <a href="#">delete</a>
An Innocent Man	369	<a href="#">show</a> <a href="#">delete</a>
Only the Good Die Young	201	<a href="#">show</a> <a href="#">delete</a>
Baby Grand	369	<a href="#">show</a> <a href="#">delete</a>
Big Shot	289	<a href="#">show</a> <a href="#">delete</a>
Goodnight Saigon	440	<a href="#">show</a> <a href="#">delete</a>
The Warrior's Code	150	<a href="#">show</a> <a href="#">delete</a>

112

Ladies and  
Gentlemen,  
meet REST

113

REST

index  
show  
new  
create  
edit  
update  
destroy

114

# LAB 6

## RESTful Controller

115

## Conference Submission

- finish off the Presentation controller
- create all the RESTful actions

116

## Conference Submission

- create a Speaker resource
- all it needs is a name
- create all the RESTful actions

117

## Conference Submission

- Make sure you can link all actions in Presentation
- Make sure you can link all actions in Speaker

118

## Associations

119

in DB's when we  
create relationships  
we tend to think of  
them backwards

120

```
mysql> describe artists;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO			
name	varchar(255)	YES			

2 rows in set (0.00 sec)

we concentrate  
on the key

```
mysql> describe songs;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
duration	int(11)	YES		NULL	
artist_id	int(11)	YES		NULL	

4 rows in set (0.00 sec)

121

```
class Artist < ActiveRecord::Base
```

```
  has_many :songs
```

```
end
```

```
class Song < ActiveRecord::Base
```

```
  belongs_to :artist
```

```
end
```

122

## ActiveRecord Conventions

# 3

has\_many associations  
must have integer column  
on the other table with  
[table name]\_id format

id	int(11)	NO	PRI	NULL	auto_increment
----	---------	----	-----	------	----------------

123

what???

124

```
class Artist < ActiveRecord::Base
```

```
  has_many :songs
```

```
end
```

we know we have  
an **artists** table

125

```
class Artist < ActiveRecord::Base
```

```
  has_many :songs
```

```
end
```

we know we have  
a **songs** table  
and a **Song** model

126

```
class Artist < ActiveRecord::Base
  has_many :songs
end
```

we also know that rails expects to find an **artists\_id** on **songs**

```
mysql> describe songs;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
duration	int(11)	YES		NULL	
artist_id	int(11)	YES		NULL	

4 rows in set (0.00 sec)

127

**has\_many :model**

Has many implies that the OTHER model has a column identifying itself as belonging to this model

128

so, first we'll create an Artist resource

129

```
class Artist < ActiveRecord::Base
  has_many :songs
end
```

we then make sure to add the relationships in

```
class Song < ActiveRecord::Base
  belongs_to :artist
end
```

130

**restart the server**

```
^CExiting
$ script/server
vital_rails(master)*
=> Booting Mongrel
=> Rails 2.3.2 application starting on
http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
```

hit CTL-C to exit

131

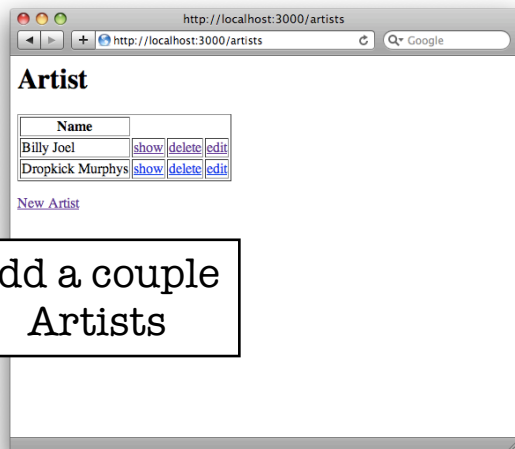
generally, adding classes means a restart of the server

132

generally, adding  
classes means a  
restart of the server

(or maybe not?)

133



134

```
>> s = Song.find :last
=> #<Song id: 16, name: "Sunshine Highway",
duration: 202, created_at: "2009-11-10 13:42:44",
updated_at: "2009-11-10 12:30:55">
>> Artist.find :all
=> [#<Artist id: 1, name: "Billy Joel", created_at: "2009-11-10 12:28:50", updated_at: "2009-11-10 12:30:55">, #<Artist id: 2, name: "Dropkick Murphys", created_at: "2009-11-10 12:30:24", updated_at: "2009-11-10 12:30:24">]
>> s.artist = Artist.find(2)
=> #<Artist id: 2, name: "Dropkick Murphys", created_at: "2009-11-10 12:30:24", updated_at: "2009-11-10 12:30:24">
>> s.save
=> true
```

associate an artist

135

```
>> s = Song.find :last
=> #<Song id: 16, name: "Sunshine Highway",
duration: 202, created_at: "2009-11-10 13:42:44",
updated_at: "2009-11-10 13:51:37">
>> s.artist
=> nil
```

wait a minute!

136

forgot the column  
on songs

137

```
$ script/generate migration add_artist_id_to_song
exists 0b/migrate
create 0b/migrate/
20091110140114_add_artist_id_to_song.rb
```

generate a  
migration

138

```
$ script/generate migration add_artist_id_to_song
exists db/migrate
create db/migrate/
20091110140114_add_artist_id_to_song.rb
```

make sure to name  
it something other  
than "adding  
column to songs"

139

db/migrate/[timestamp]\_add\_artist\_id\_to\_song.rb

```
class AddArtistIdToSong < ActiveRecord::Migration
  def self.up
    add_column :songs, :artist_id, :integer
  end

  def self.down
    drop_column :songs, :artist_id
  end
end
```

140

```
$ rake db:migrate
(in /Users/objo/git/vital_rails/src/songs)
== AddArtistIdToSong: migrating =====
-- add_column(:songs, :artist_id, :integer)
   -> 0.0008s
== AddArtistIdToSong: migrated (0.0009s) ==
```

run the migration

141

```
>> s = Song.find :last
=> #<Song id: 16, name: "Sunshine Highway",
duration: 202, created_at: "2009-11-10
13:42:44", updated_at:
artist_id: nil>
>> a = Artist.find(2)
=> #<Artist id: 2, name
created_at: "2009-11-10 12:30:24", updated_at:
"2009-11-10 12:30:24">
>> s.artist = a
=> #<Artist id: 2, name: "Dropkick Murphys",
created_at: "2009-11-10 12:30:24", updated_at:
"2009-11-10 12:30:24">
>> s.save
=> true
```

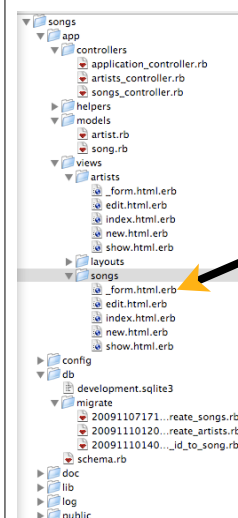
associate an  
artist again

142

```
>> s = Song.find :last
=> #<Song id: 16, name: "Sunshine Highway",
duration: 202, created_at: "2009-11-10
13:42:44", updated_at: "2009-11-10 14:18:31",
artist_id: 2>
>> s.artist
=> #<Artist id: 2, name: "Dropkick Murphys",
created_at: "2009-11-10 12:30:24", updated_at:
"2009-11-10 12:30:24">
>>
```

and now it  
works

143



open the \_form  
view for songs

144



app/views/songs/\_form.html.erb

add this to  
the form

```
...  
<p><%= label_tag 'song_artist' %><br/>  
<%= select("song", "artist_id",  
  Artist.all.collect {|a| [ a.name, a.id ] },  
  { :include_blank => true }) %></p>  
...
```

145

app/views/songs/\_form.html.erb

name of the  
model

```
...  
<p><%= label_tag 'song_artist' %><br/>  
<%= select("song", "artist_id",  
  Artist.all.collect {|a| [ a.name, a.id ] },  
  { :include_blank => true }) %></p>  
...
```

attribute  
name

146

app/views/songs/\_form.html.erb

what is wrong with this?

```
...  
<p><%= label_tag 'song_artist' %><br/>  
<%= select("song", "artist_id",  
  Artist.all.collect {|a| [ a.name, a.id ] },  
  { :include_blank => true }) %></p>  
...
```

147

since we abstracted  
out the form, it will  
now work for edit and  
new

148

http://localhost:3000/songs/new

**Song**

Song name

Song duration

Song artist

or [cancel](#)

149

http://localhost:3000/songs/17/edit

**Song**

Song name

Song duration

Song artist

or [cancel](#)

150

other associations

has\_one  
has\_many  
belongs\_to

has\_many :through  
has\_and\_belongs\_to\_many

151

## Extra Parts of ActiveRecord

152

## Extra Parts of ActiveRecord

153

named scopes

```
class Song < ActiveRecord::Base  
  
  LONG_SONG = 300  
  
  named_scope :short,  
    :conditions => ["duration < ?", LONG_SONG]  
  named_scope :highly_rated  
    :conditions => ["rating < ?", 4]  
  ...  
end
```

```
>> Song.highly_rated.short
```

154

callbacks

- before\_validation
- before\_validation\_on\_create
- after\_validation
- after\_validation\_on\_create
- before\_save
- before\_create
- after\_create
- after\_save

```
...  
  after_create :log_creation  
  ...  
  def log_creation  
    log.info { "Created song #{name} " }  
  end  
end
```

155

single table inheritance

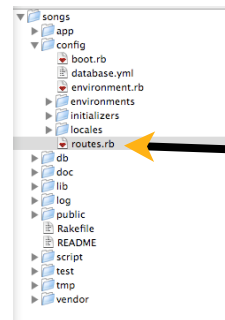
Allows us to use  
polymorphism  
with  
ActiveRecord.  
AR tracks the  
object type with  
a column called  
'type'

```
class Song < ActiveRecord::Base  
  ...  
end  
  
class MetalSong < Song  
end  
  
class CelticSong < Song  
end
```

156

# Rails Routes

157



open this file

158

the magic ones

```

ActionController::Routing::Routes.draw do |map|
  map.resources :artists

  map.resources :songs

  # The priority is based upon order of creation: first created -> ...

  # Sample of regular route:
  # map.connect 'products/:id', :controller => 'catalog', :action => ...
  # Keep in mind you can assign values other than :controller and :action
  ...

  # Sample of named route:
  # map.purchase 'products/:id/purchase', :controller => ...
  # This route can be invoked with purchase_url(id => product.id) ...
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'
end

```

159

run rake routes in terminal

```

$ rake routes
songs GET /songs/:id(.:format) {:controller=>"songs", :action=>"index"}
      POST /songs/:id(.:format) {:controller=>"songs", :action=>"create"}
new_song GET /songs/new(.:format) {:controller=>"songs", :action=>"new"}
edit_song GET /songs/:id/edit(.:format) {:controller=>"songs", :action=>"edit"}
song GET /songs/:id(.:format) {:controller=>"songs", :action=>"show"}
      PUT /songs/:id(.:format) {:controller=>"songs", :action=>"update"}
      DELETE /songs/:id(.:format) {:controller=>"songs", :action=>"destroy"}
{:controller=>"songs", :action=>"destroy"}
/:controller/:action/:id
/:controller/:action/:id(.:format)
/ {:controller=>"songs", :action=>"index"}

```

160

for a default route to / add this

```

ActionController::Routing::Routes.draw do |map|
  map.resources :artists

  map.resources :songs

  # The priority is based upon order of creation: first created -> ...

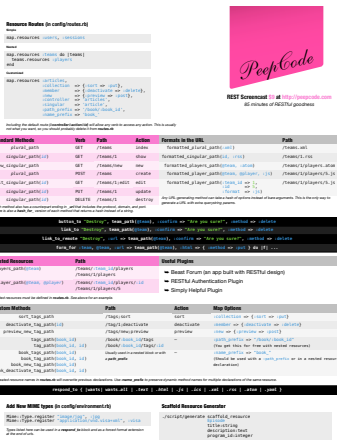
  # Sample of regular route:
  # map.connect 'products/:id', :controller => 'catalog', :action => ...
  # Keep in mind you can assign values other than :controller and :action
  ...

  # Sample of named route:
  # map.purchase 'products/:id/purchase', :controller => ...
  # This route can be invoked with purchase_url(id => product.id) ...
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'

  map.connect '/', :controller => 'songs', :action => 'index'
end

```

162



topfunky.com

or Google  
REST Cheatsheet

161

so what's happening?

163

Dispatcher

/songs/new/1

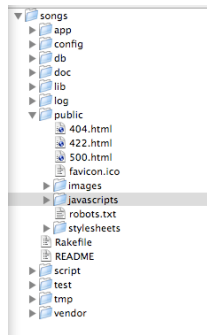
looks at URL

164

Dispatcher

/songs/new/1

looks in  
public directory  
for a match



165

Dispatcher

then pulls it apart

/songs/new/1

:controller

:action

:id

166

priority

```
ActionController::Routing::Routes.draw do |map|
  map.resources :artists

  map.resources :songs

  # The priority is based upon order of creation: first created -> ...

  # Sample of regular route:
  # map.connect 'products/:id', :controller => 'catalog', :action => ...
  # Keep in mind you can assign values other than :controller and :action
  ...

  # Sample of named route:
  # map.purchase 'products/:id/purchase', :controller => ...
  # This route can be invoked with purchase_url(:id => product.id) ...
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'

  map.connect '/', :controller => 'songs', :action => 'index'
end
```

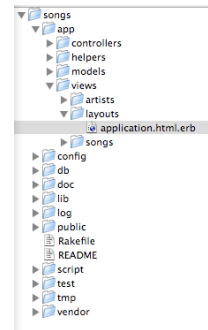
167

Views and Helpers

168

# layouts

169



create this  
file

```
$ touch app/views/layouts/application.html.erb
```

170

app/views/layouts/application.html.erb

```
<html>
<body>
  <%= yield %>
</body>
</html>
```

if you create a layout named  
'application' it will be picked  
up automatically

171

app/views/layouts/application.html.erb

```
<html>
<body>
  <%= yield %>
</body>
</html>
```

the yield is  
where your page  
content will go

172

app/views/songs/new.html.erb

```
<h2>Song</h2>
<%= error_messages_for :song %>
<% form_for :song, :url => songs_path do |f| -%>
  <%= render :partial => 'form', :locals => { :f => f }
%>
<% end -%>
```

now you can take  
away the <html>  
and <body> tags

173

taking it a step  
further

174

app/views/layouts/application.html.erb

```
<html>
<%= stylesheet_link_tag :all %>
<body>
<div id="wrap">
  <div id="header">
    <center><h1>rTunes: Your music on the web</h1></center>
  </div>
  <div id="main">
    <%= yield %>
  </div>
  <div id="sidebar">
    <h2>Navigation</h2>
    <ul>
      <li><%= link_to "Songs", songs_path %></li>
      <li><%= link_to "Create a Song", new_song_url %></li>
      <li><%= link_to "Artists", artists_url %></li>
      <li><%= link_to "Create an Artist", new_artist_url %></li>
    </ul>
  </div>
  <div id="footer">
    <p>Footer</p>
  </div>
</div>
...

```

yield is here

175

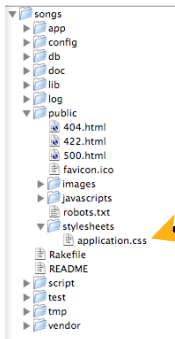
app/views/layouts/application.html.erb

```
<html>
<%= stylesheet_link_tag :all %>
<body>
<div id="wrap">
  <div id="header">
    <center><h1>rTunes: Your music on the web</h1></center>
  </div>
  <div id="main">
    <%= yield %>
  </div>
  <div id="sidebar">
    <h2>Navigation</h2>
    <ul>
      <li><%= link_to "Songs", songs_path %></li>
      <li><%= link_to "Create a Song", new_song_url %></li>
      <li><%= link_to "Artists", artists_url %></li>
      <li><%= link_to "Create an Artist", new_artist_url %></li>
    </ul>
  </div>
  <div id="footer">
    <p>Footer</p>
  </div>
</div>
...

```

add this line here. It dynamically grab all stylesheets and include them

176



create this file

\$ touch public/stylesheets/application.css

177

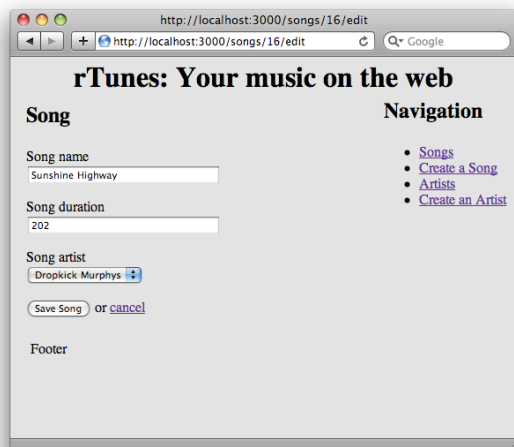
public/stylesheets/application.css

```
body,
html {
  margin:0;
  padding:0;
  color:#000;
  background:#ddd;
}
#wrap {
  width:550px;
  margin:0 auto;
}
#header {
  padding:5px 10px;
}
h1 {
  margin:0;
}
#main {
  float:left;
  width:380px;
  padding:5px;
}
h2 {
  ...
}

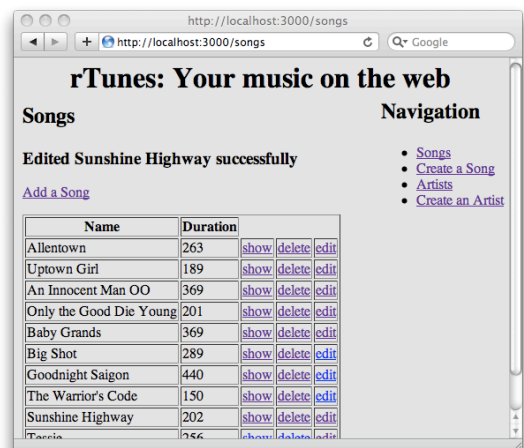
```

add some designer foo

178



179



180

## helpers

181

## refactoring views

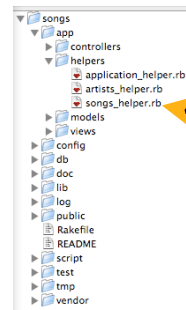
182

app/views/songs/new.html.erb

```
<h2>Songs</h2>
<% if flash[:success] -%>
  <h3><%= flash[:success] %></h3>
<% end -%>
<p><%= link_to "Add a Song",
  new_song_url %></p>
<table border=1>
  <tr>
    <td><%= @song.name %></td>
    <td><%= @song.duration %></td>
  </tr>
</table>
```

anytime you have  
logic in a view, use a  
helper (almost)

183



open this  
file

songs helper is for the  
songs controller

184

app/views/songs/new.html.erb

```
<h2>Songs</h2>
<%= flash_success %>
<p><%= link_to "Add a Song",
  new_song_url %></p>
<table border=1>
  <tr>
    <th>Name</th>
    <th>Duration</th>
  </tr>
  <tr>
    <td><%= @song.name %></td>
    <td><%= @song.duration %></td>
  </tr>
</table>
```

refactor to a  
method  
(extract  
method)

185

app/helpers/songs\_helper.rb

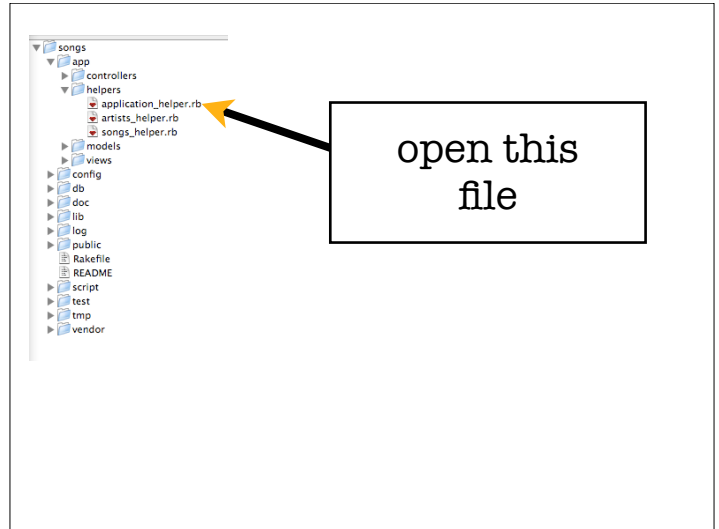
```
module SongsHelper
  def flash_success
    if flash[:success]
      "<h3>#{flash[:success]}</h3>"
    end
  end
end
```

return a string  
of html

186

but this will only be available for views in the songs controller

187



188

app/helpers/application\_helper.rb

```
module ApplicationHelper
  def flash_success
    if flash[:success]
      "<h3>#{flash[:success]}</h3>"
    end
  end
end
```

now it is available to all controllers

189