Ruby Explorations XII

Mark Keane...CSI...UCD

De End: Final Bits...

A: De Puppies
B: De Helpers
C: De Scaffolding
D: Putting De App on De Web...
E: De YAML
F: De Exam

A: De Puppies

Ruby & Rails

Puppies V1.0: Ruby & Rails

Rails on Ruby

Puppies 4.1.6

B: De Helpers

Putting more functionality into where...

Where's the Beef?

Ruby functionality in the controllers seems quite simple; the complexity arises in co-ordinating between Model-View and Controller

This may leave you slightly disappointed...its really not like writing Ruby programs

You can have a lot more functionality, if you want it; if so, it tends to go into files in the Helpers Folder

Many hidden app-level helpers exist (e.g., **link_to**)

link_to_*criteria* eg

let's look at defining our own helpers, extending the way the existing **link_to** works, so that it takes an **id**

we will also complex-up a controller, to show it doing a bit more than usual

the application is called **clovis**, why ?

it has a controller show (list method) and a model for Song (with name, artist, album, id, fields)

after doing....

rails new clovis

cd clovis

rails generate controller show list

rails generate model Song name:string artist:string album:strin

…..edit various files with stuff…

use rails console to look at db and what sort_by does

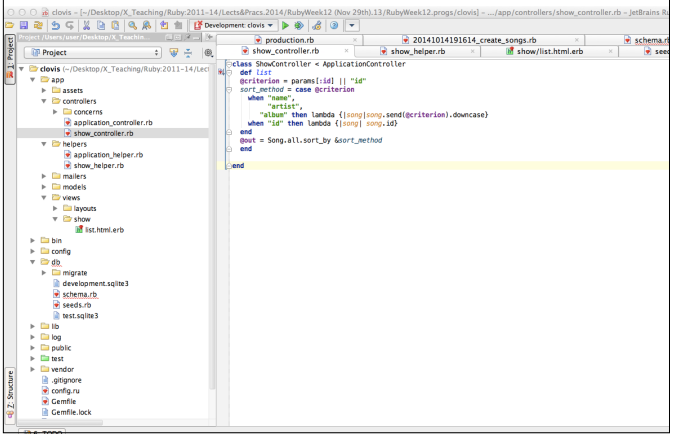
rake db:create

rake db:migrate

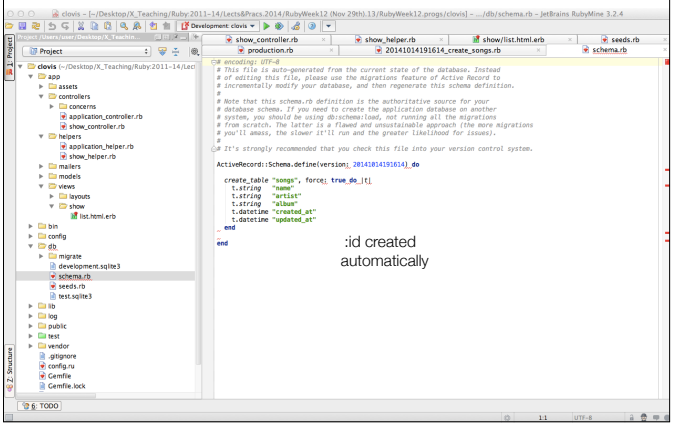
rake db:seed

rails server

clovis controller **show/list**

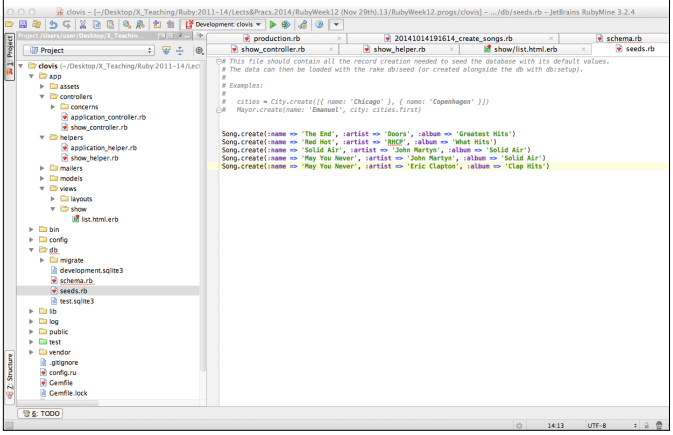


clovis model **Song**



:id created automatically

clovis **seeds**



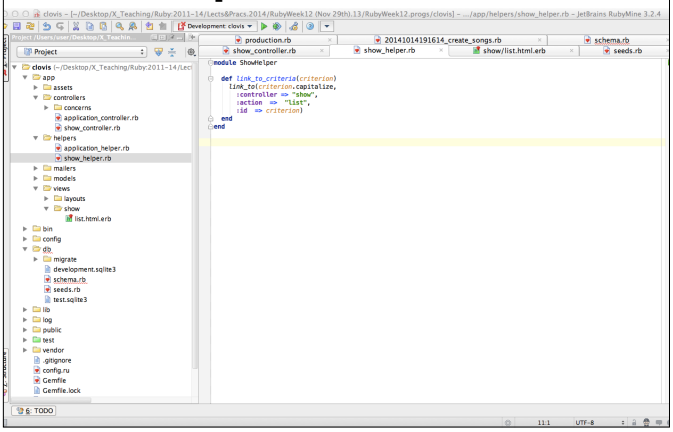
link_to_criterion

if there is a method you what to use in several places, in a given view, you can define it in the helper file; every controller has its own automatically created ***controller_name*_helper.rb**

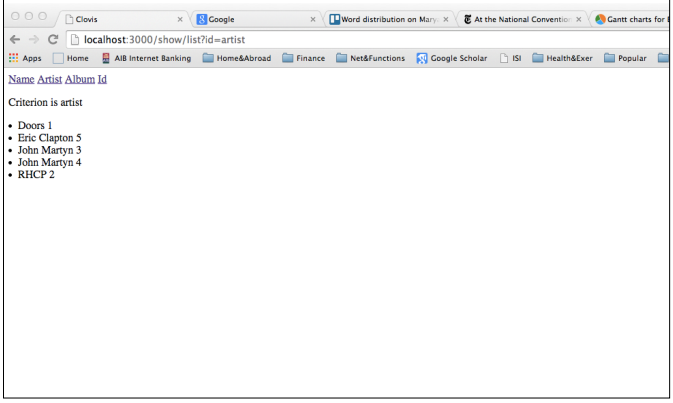
Note, it is just a Ruby module

in this eg we define a method that produces different flavours of a **link_to** method depending on the situation we are in

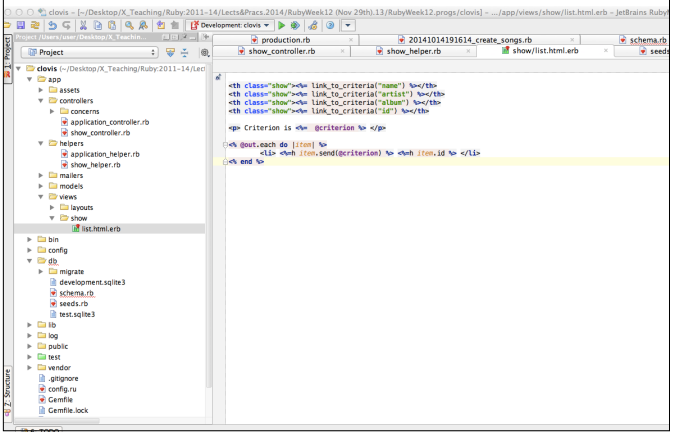
show **helper**



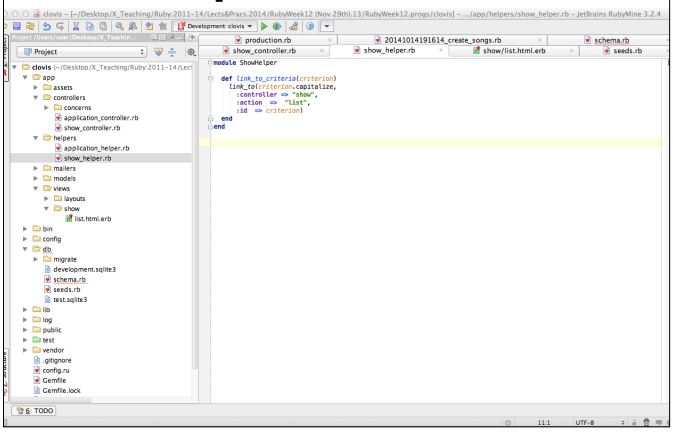
link_to screen for view



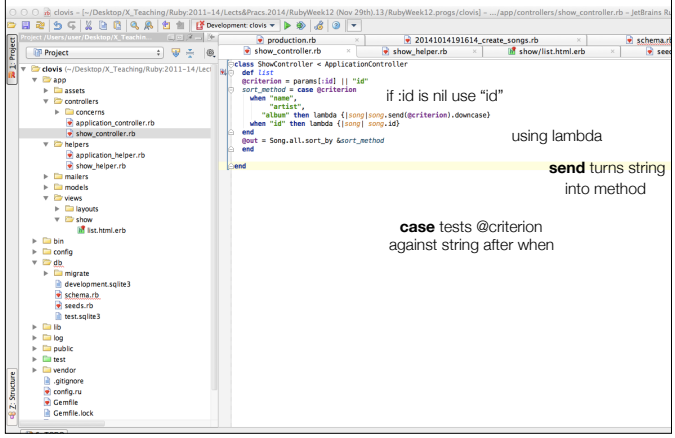
show/list view



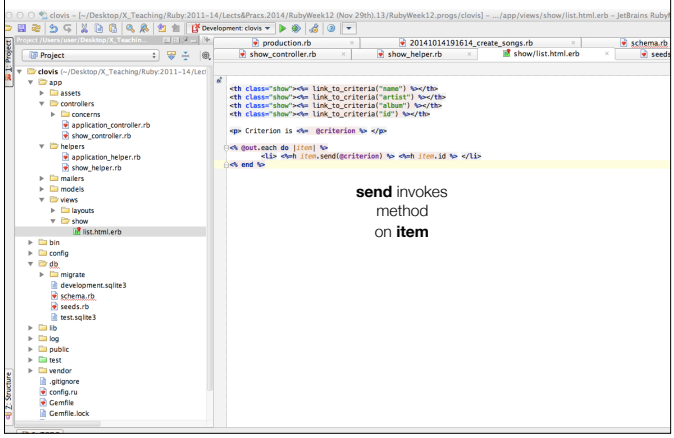
show **helper**



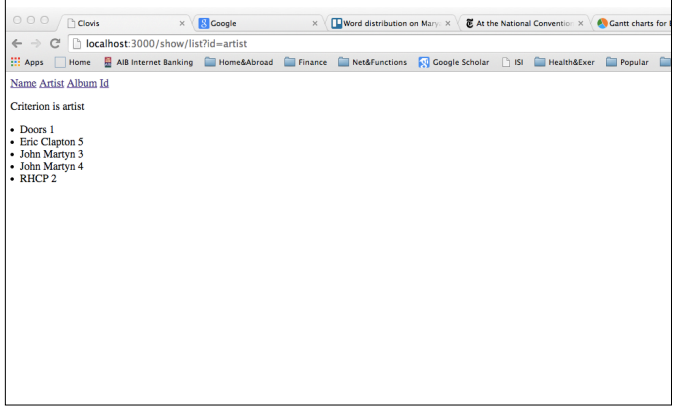
clovis controller **show/list**



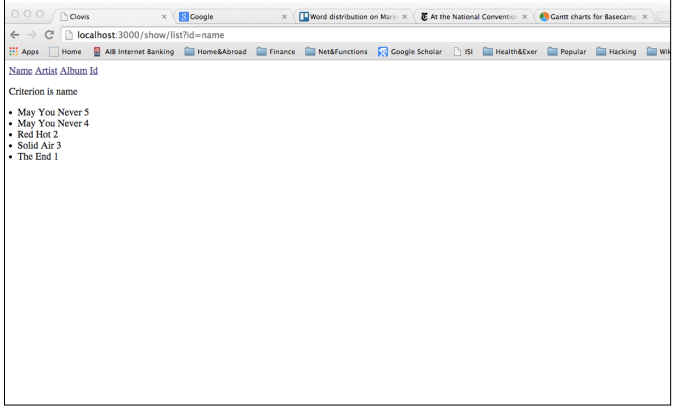
show/list view



link_to screen for view



link_to screen for view



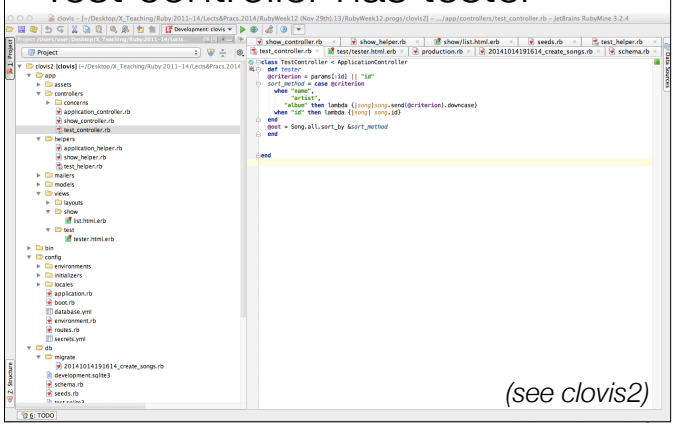
Where can you use it...

methods defined in **show_helper.rb** seem to be accessible to any view-file via the method name called; if you define a method in another helper with the same name, it goes to the helper-view consistent file first *(see clovis2 and 3)*

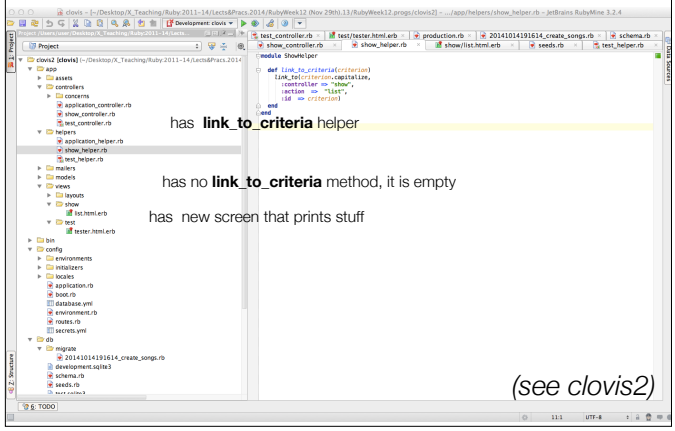
but, you can officially make this so by putting them in the **application_helper.rb** file

nb we could have put the functionality of **link_to_criterion** in the view, but then it would be more than a view (sort of), using a helper is better encapsulation and cud aid maintenance

Test controller has tester



show **helper in clovis2**



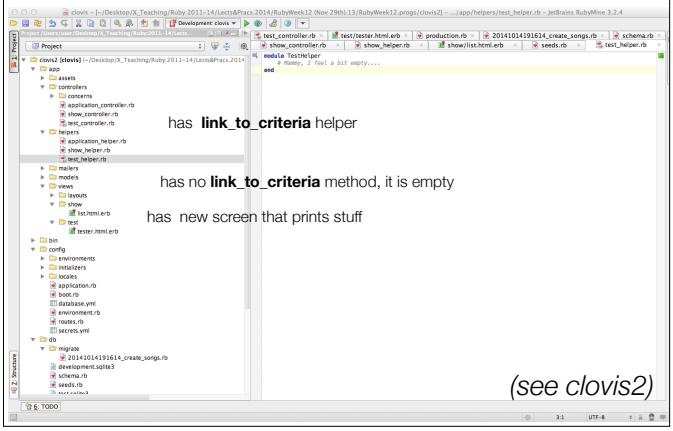
has **link_to_criteria** helper

has no **link_to_criteria** method, it is empty

has new screen that prints stuff

(see clovis2)

test **helper in clovis2**



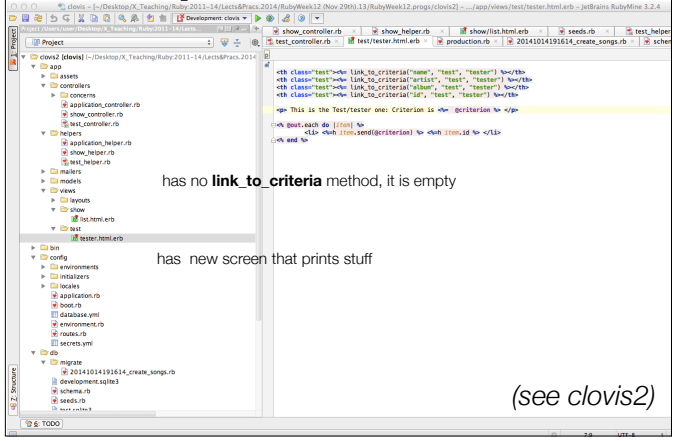
has **link_to_criteria** helper

has no **link_to_criteria** method, it is empty

has new screen that prints stuff

(see clovis2)

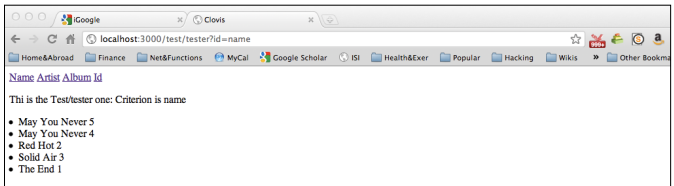
test **helper in clovis2**



has no **link_to_criteria** method, it is empty

has new screen that prints stuff

(see clovis2)



so, this is working off **link_to_criteria** in the show helper, so the tester view must have access to it; see also **clovis3** for definitive test

how can this happen?

C. De Scaffolding

Scaffolding & REST

Certain aspects of controller and model creation can be automated more

For example, most of the time we create controllers for getting inputs, updating tables and showing their contents...perhaps this could be done for us?

Scaffold & REST do these sort of tasks

But, like any framework once you buy into them, constraints can follow, that you have to live with

http://en.wikipedia.org/wiki/Representational_State_Transfer

REST is...

Representational **S**tate **T**ransfer is a lightweight architect style for designing networked applications; in its use of HTTP protocol to make calls between machines

Contrasts to heavy-weight CORMA, RPC, SOAP

RESTful applications use HTTP requests to post data (create/update), read data (e.g., make queries) and delete data; uses HTTP for all four CRUD operations (create/read/update/delete)

So, we will see GET and POST being used a lot

http://en.wikipedia.org/wiki/Representational_State_Transfer

History [edit]

The REST architectural style was developed in parallel with HTTP/1.1, based on the existing design of HTTP/1.0.[6] The largest implementation of a system conforming to the REST architectural style is the World Wide Web. REST exemplifies how the Web's architecture emerged by characterizing and constraining the macro-interactions of the four components of the Web, namely origin servers, gateways, proxies and clients, without imposing limitations on the individual participants. As such, REST essentially governs the proper behavior of participants.

Concept [edit]

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that may be used next time the client chooses to initiate a new state transition.[7]

REST was initially described in the context of HTTP, but is not limited to that protocol. RESTful architectures can be based on other Application Layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximize the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on top of it.

HTTP examples [edit]

HTTP, for example, has a very rich vocabulary in terms of verbs (or "methods"), URIs, Internet media types, request and response codes, etc. REST uses these existing features of the HTTP protocol, and thus allows existing layered proxy and gateway components to perform additional functions on the network such as HTTP caching and security enforcement.

SOAP RPC contrast [edit]

SOAP RPC over HTTP, on the other hand, encourages each application designer to define a new and arbitrary vocabulary of nouns and verbs (for example `getUsers()`, `savePurchaseOrder(...)`), usually overlaid onto the HTTP POST verb. This disregards many of HTTP's existing capabilities such as authentication, caching and content type negotiation, and may leave the application designer re-inventing many of these features within the new vocabulary.[8] Examples of doing so may include the addition of methods such as `getNewUsersSince(Date date)`, `savePurchaseOrder(string customerLogon, string password, ...)`.

http://en.wikipedia.org/wiki/Representational_State_Transfer

Blog example

$ rails new blog

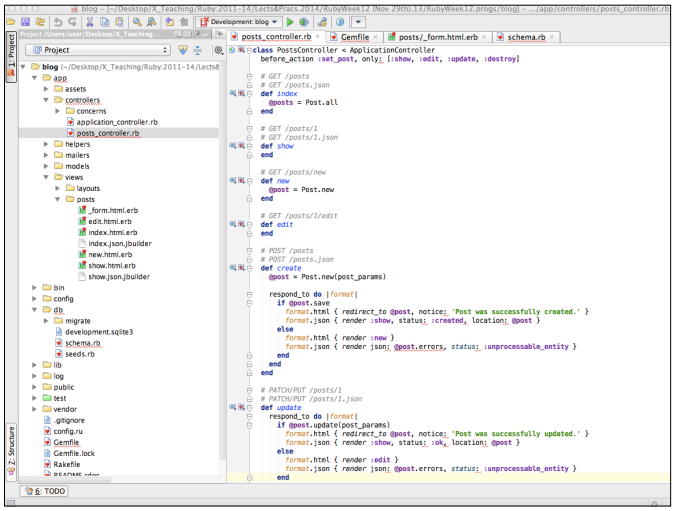
$ cd blog

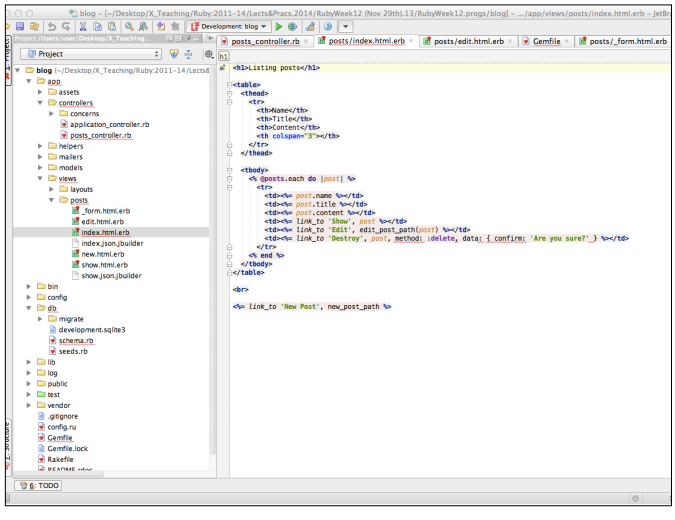
$ rails generate scaffold Post name:string title:string content:string

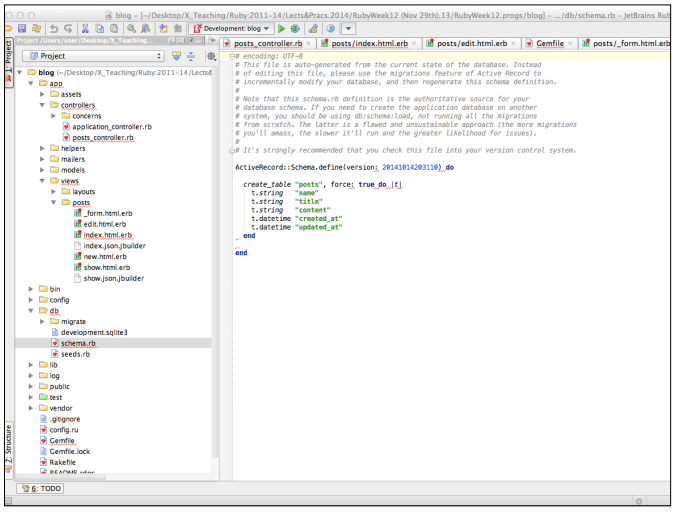
$ rake db:migrate

$ rails server...and look at what is there

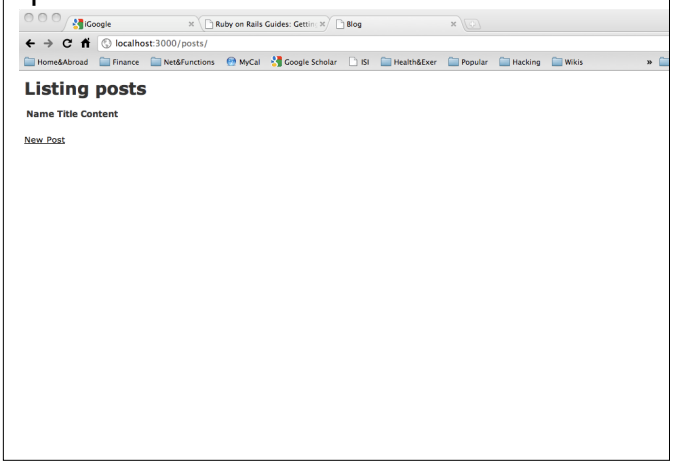
whooooa.....Nellie !!!



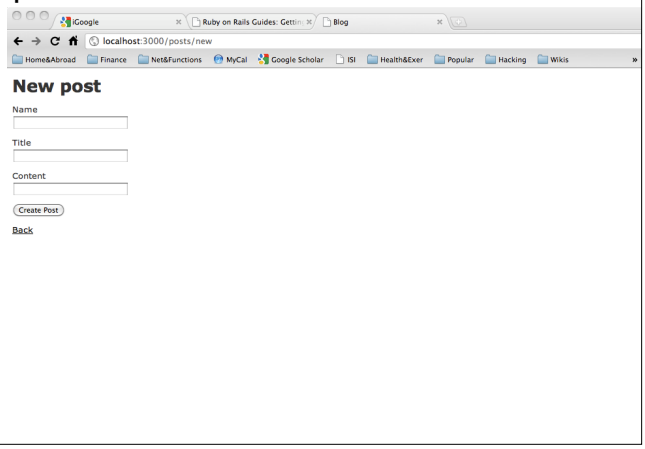




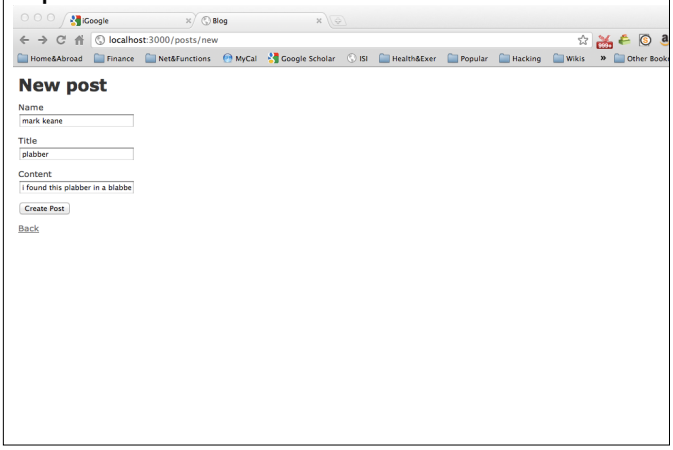
posts/



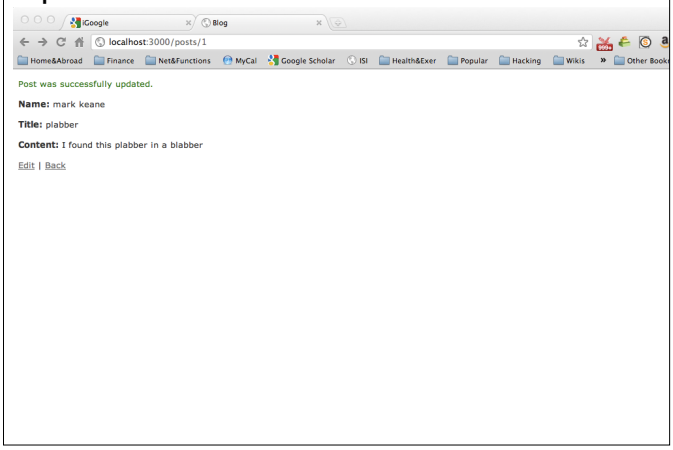
posts/new



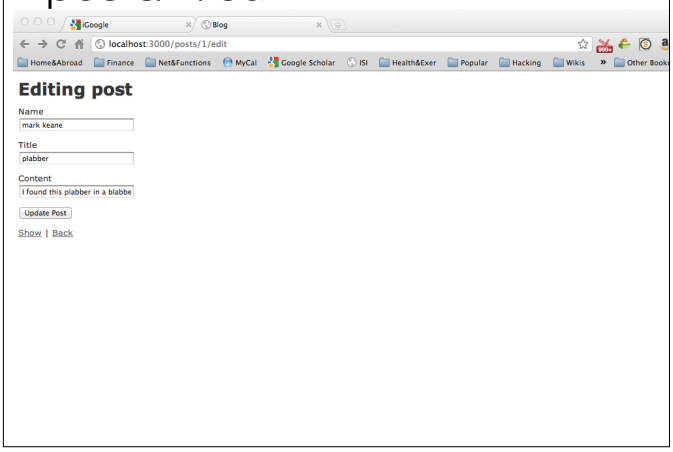
posts/new



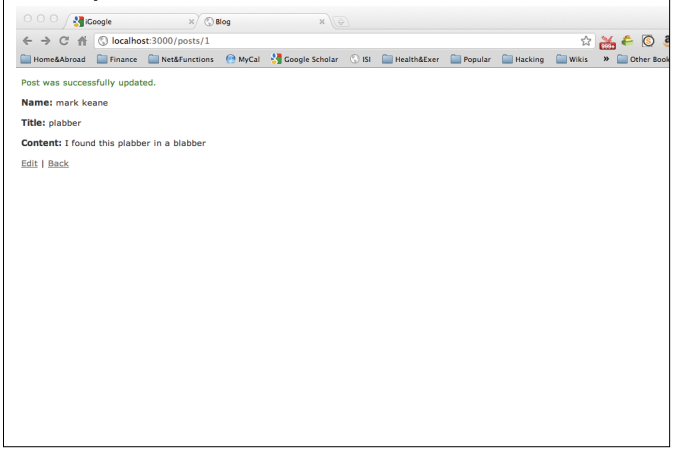
posts/1



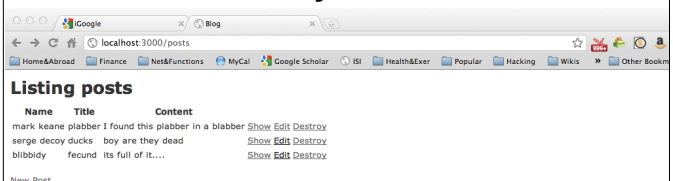
posts/1/edit



...updated...

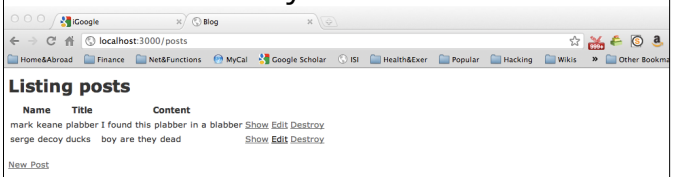


...back sends you to...

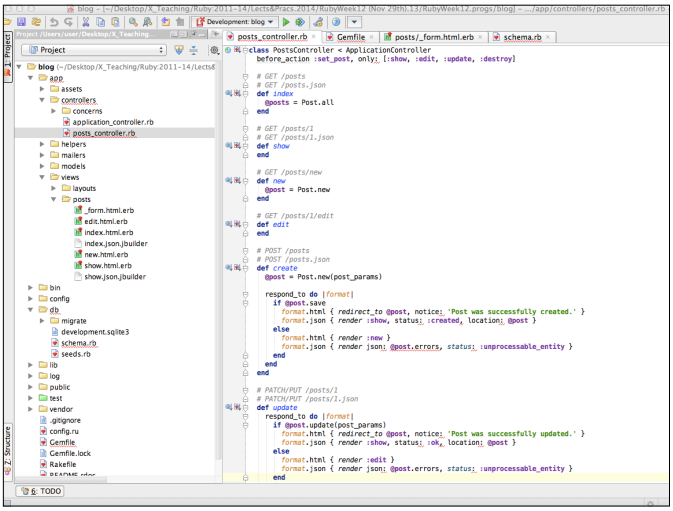


localhost:3000/posts
defaults to…
localhost:3000/posts/index

...and destroy...



localhost:3000/posts
defaults to…
localhost:3000/posts/index



Scaffold creates

File	Purpose
db/migrate/20100207214725_create_posts.rb.rb	Migration to create the posts table in your database (your name will include a different timestamp)
app/models/post.rb	The Post model
test/fixtures/posts.yml	Dummy posts for use in testing
app/controllers/posts_controller.rb	The Posts controller
app/views/posts/index.html.erb	A view to display an index of all posts
app/views/posts/edit.html.erb	A view to edit an existing post
app/views/posts/show.html.erb	A view to display a single post
app/views/posts/new.html.erb	A view to create a new post
app/views/posts/_form.html.erb	A partial to control the overall look and feel of the form used in edit and new views
app/helpers/posts_helper.rb	Helper functions to be used from the post views
test/unit/post_test.rb	Unit testing harness for the posts model
test/functional/posts_controller_test.rb	Functional testing harness for the posts controller
test/unit/helpers/posts_helper_test.rb	Unit testing harness for the posts helper
config/routes.rb	Edited to include routing information for posts
public/stylesheets/scaffold.css	Cascading style sheet to make the scaffolded views look better

D: Deploying a Rails App...

Getting to the world...wide web

Deployment Options...

There are several different options for launching your rails app on the Web, assuming you have a domain name for it

Depends on whether (i) you are hosting it on your own server or (ii) using some service

Often this may demand some db changes...to a more robust db than sqlite

nb. it will use the production version of the system

By Hand Yourself...

Register your site domain -- www.mysite.com -- and point it to your site

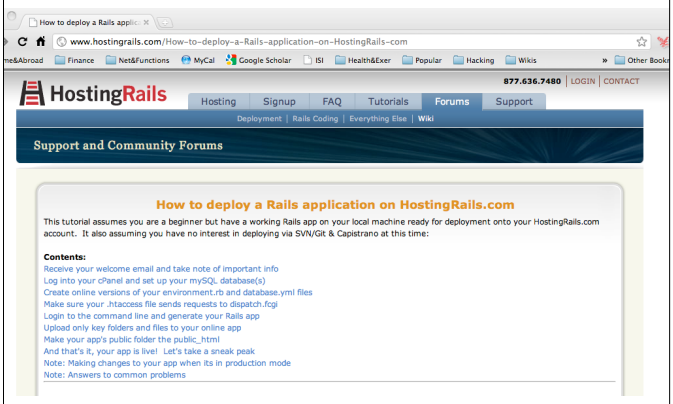
Modify e.g. Apache config file (diff in diff servers)

```
<VirtualHost www.mysite.com>
ServerName www.mysite.com
ServerAlias mysite.com
DocumentRoot "usr/local/Share/railsapp/mysite/public"
</VirtualHost>
```

Apache looks at dir for .**htaccess** file (in public directory of Rails app; see README)

it will trigger execution of dispatcher, calling controllers appropriately

Via a Service...



http://www.hostingrails.com/

Various Tools

Capistrano: used for running scripts on multiple servers; automates task of making new version of application available on one/more servers

Subversion (SVN) for version control

VRM for version control

Git and Github for version control

E: YAML

for dumping dbs and passing around data

YAML definition

is a machine parse-able data serialization format designed for human readability and for use with scripting languages (e.g., perl and python)

is optimized for data serialization, formatted dumping, configuration files, log files, internet messaging and filtering

YAML on arrays

```
require 'yaml'

test_1 = ["this", "is", "an", ["embedded", "array"], 2, 3]
test_2 = ["level1", ["level2", ["level 3"]], 2, 3]

p test_1
puts YAML::dump(test_1)

p test_2
puts YAML::dump(test_2)

yml_obj = YAML::dump(test_1)

p YAML::load(yml_obj)
```

yaml.rb

YAML on arrays

```
require 'yaml'

test_1 = ["this", "is", "an", ["embedded",
"array"], 2, 3]
test_2 = ["level1", ["level2", ["level 3"]],
2, 3]

p test_1
puts YAML::dump(test_1)

p test_2
puts YAML::dump(test_2)

yml_obj = YAML::dump(test_1)

p YAML::load(yml_obj)
```

yaml.rb

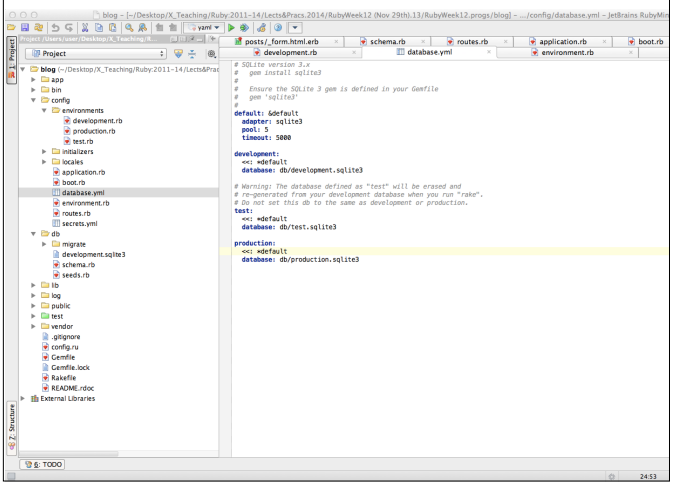
```
ruby yaml.rb
["this", "is", "an", ["embedded", "array"], 2, 3]
---
- this
- is
- an
- - embedded
  - array
- 2
- 3
["level1", ["level2", ["level 3"]], 2, 3]
---
- level1
- - level2
  - - level 3
- 2
- 3
["this", "is", "an", ["embedded", "array"], 2, 3]
```

YAML on config files

in any rails app the config folder has **database.yml**, that looks like this:

```
# SQLite version 3.x
#   gem install sqlite3

default: &default
  adapter: sqlite3
  pool: 5
  timeout: 5000

development:
  <<: *default
  database: db/development.sqlite3

# Warning: …
# Do not set this db to the same as development or production.
test:
  <<: *default
  database: db/test.sqlite3

production:
  <<: *default
  database: db/production.sqlite3
```

database.yml



See what YAML does…

```
require 'yaml'

file = File.open('database.txt')

p YAML::load(file)
```

loader.rb

$ ruby loader.rb

{"default"=>{"adapter"=>"sqlite3", "pool"=>5, "timeout"=>5000},
"development"=>{"adapter"=>"sqlite3", "pool"=>5, "timeout"=>5000,
"database"=>"db/development.sqlite3"}, "test"=>{"adapter"=>"sqlite3",
"pool"=>5, "timeout"=>5000, "database"=>"db/test.sqlite3"},
"production"=>{"adapter"=>"sqlite3", "pool"=>5, "timeout"=>5000,
"database"=>"db/production.sqlite3"}}

database.yml for mysql

```
development:
  adapter: mysql2
  encoding: utf8
  database: blog_development
  pool: 5
  username: root
  password:
  socket: /tmp/mysql.sock
  ....
```

config.yml

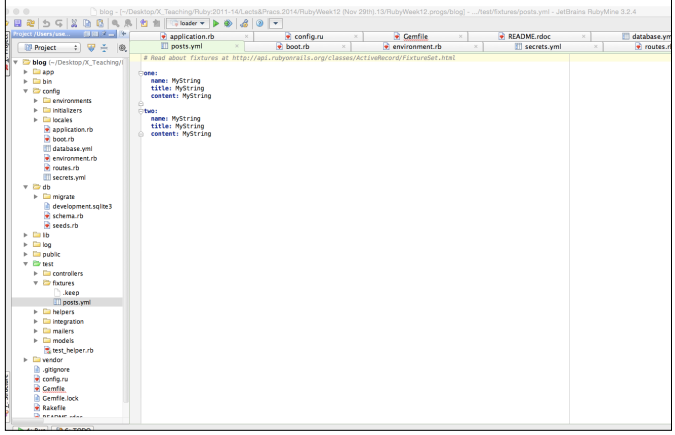
YAML on dumping a db

When moving dbs the contents of a db can be dumped into a YAML file and then re-loaded later into a new db

for test purposes there is a yml file in fixures that can be loaded into the database with:

$ rake db:fixtures:load

fixtures



```
hcp-892bc88c:clovis3 mkeane$ rails console
Loading development environment (Rails 4.0.1)
irb(main):001:0> Song.all
=> [#<Song id: 1, name: "The End", artist: "Doors", album: "Greatest Hits", created_at: "2010-11-16 18:37:00",
updated_at: "2010-11-16 18:37:00">, #<Song id: 2, name: "Red Hot", artist: "RHCP", album: "What Hits", created_at:
"2010-11-16 18:37:00", updated_at: "2010-11-16 18:37:00">, #<Song id: 3, name: "Solid Air", artist: "John Martyn",
album: "Solid Air", created_at: "2010-11-16 18:37:00", updated_at: "2010-11-16 18:37:00">, #<Song id: 4, name:
"May You Never", artist: "John Martyn", album: "Solid Air", created_at: "2010-11-16 18:37:01", updated_at:
"2010-11-16 18:37:01">, #<Song id: 5, name: "May You Never", artist: "Eric Clapton", album: "Clap Hits", created_at:
"2010-11-16 18:37:01", updated_at: "2010-11-16 18:37:01">]
irb(main):002:0> quit

dhcp-892bc88c:clovis3 mkeane$ rake db:fixtures:load
...
dhcp-892bc88c:clovis3 mkeane$ rails console
Loading development environment (Rails 4.0.1)
irb(main):001:0> Song.all
=> [#<Song id: 298486374, name: "MyString", artist: "MyString", album: "MyString", created_at: "2011-11-04
12:01:12", updated_at: "2011-11-04 12:01:12">, #<Song id: 980190962, name: "MyString", artist: "MyString", album:
"MyString", created_at: "2011-11-04 12:01:12", updated_at: "2011-11-04 12:01:12">, #<Song id: 980190963, name:
"The End", artist: "Doors", album: "Greatest Hits", created_at: "2011-11-04 12:02:29", updated_at: "2011-11-04
12:02:29">, #<Song id: 980190964, name: "Red Hot", artist: "RHCP", album: "What Hits", created_at: "2011-11-04
12:02:29", updated_at: "2011-11-04 12:02:29">, #<Song id: 980190965, name: "Solid Air", artist: "John Martyn",
album: "Solid Air", created_at: "2011-11-04 12:02:29", updated_at: "2011-11-04 12:02:29">, #<Song id: 980190966,
name: "May You Never", artist: "John Martyn", album: "Solid Air", created_at: "2011-11-04 12:02:29", updated_at:
"2011-11-04 12:02:29">, #<Song id: 980190967, name: "May You Never", artist: "Eric Clapton", album: "Clap Hits",
created_at: "2011-11-04 12:02:29", updated_at: "2011-11-04 12:02:29">]
```

F: De Exam

Question Types...

University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER I EXAMINATION – 2015/2016

COMP 41100

Exploring Programming in Ruby

Prof. S. Dobson

Prof. P. Cunningham

Prof. M. Keane*

Module Coordinator*

Time allowed: 2 hours

Instructions for candidates out of **7** listed

Answer Five Questions:

Question One from Part I is Compulsory,

Answer Any Two Questions from Part II

Answer Any Two Questions from Part III

All Questions carry equal marks.

Use of calculators is prohibited.

Instructions for invigilators

Use of calculators is prohibited.

Part I, Q1:Compulsory

3. What do the following evaluate to in Ruby:

a. “hello there big boy”
b. 56
c. 34.00
d. 0.222222354454365
e. [“a”, “b”, “c”]
f. +
g. PI.....

Part II:Programming Qs

Define a class for FamilyMember that has at least 5 properties (e.g., name, sex, status, age) and then create 5 instances of that class, each of which should be assigned to the following variables: fm1, fm2, fm3, fm4, fm5. Define two methods – parent? and child? that will test whether an instance of this class is a parent or a child and what type of parent or child (father, mother, son daughter, dog). The method should make use of two or more properties in doing this test.

Define an array of these family members and search through it using each, to print out their names, and status.

Define an array of these family members and search through it using do, to print out their names, and status.

Part III: Concept Qs

1. Write a short paragraph on any ***four*** of the key properties that are asserted to be characteristic of the Object-Oriented Programming paradigm, using appropriate examples.

2. What are Ruby Gems? Write an essay on the functionality of two Gems with which you are familiar, illustrating your answer with examples.

What I have tried to do...

in Ruby and Rails

Ruby Covered

Give you a overall sense of how the language works

Give you enough knowledge to understand any bits that were not covered

You should understand what any code *might* be from looking at its syntax etc...even if the methods are not familiar

Rails Covered

Give you a overall sense of how the Rails framework works

We have covered less of Rails, but much of the remainder is quite specific stuff about forms/cgi/validation/cookies

None of this should be surprising, though there will be domain-specific complexities in its use

Goodbye...Goodbye... Goodbye....