

# Ruby Explorations XI

Mark Keane...CSI...UCD



# Rails Basics

More Ruby on Rails:

A: understanding Rake's migration & rollback

B: itunes in Rails

C: using **rails console** to look at db

D: **link\_to** has a third possible arg...

E: When time passes and my bundle is old...

# Controller

Model

\$ rake db:migrate

Views

\$ rails server

# Part A:

## Understanding Rake migration & rollback

# Recall...

Database tables are produced by a combination of migrations and editing of tables (and changing dbs directly)...in all of this **schema.rb** is the definitive source for the db structure

it is very easy to fail to understand how this all works and to encounter strange errors

so, lets look at it carefully....

[http://guides.rubyonrails.org/command\\_line.html](http://guides.rubyonrails.org/command_line.html)

# Steps

\$ rails new chicken

\$ cd chicken migration\_file only created

\$ rails generate model Bird name:string

\$ rake db:migrate schema.rb file created

....now imagine that we go into the migration file and alter it...and re-run **rake db:migrate...**

what will **schema.rb** look like now ?

# \$ rails generate model Bird name:string

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The left sidebar displays the project structure under 'Project' for the 'chicken' application. Key components shown include:

- app:** Contains assets, controllers, helpers, mailers, models (with concerns and .keep), views, and bin.
- db:** Contains db/migrate (with 20130916185447\_create\_birds.rb) and seeds.rb.
- lib:** Contains assets, tasks, log, public, test, vendor (with .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc).

The main editor window shows the migration file `20130916185447_create_birds.rb` with the following code:class CreateBirds < ActiveRecord::Migration
 def change
 create\_table :birds do |t|
 t.string :name
 t.timestamps
 end
 end
end

A yellow callout box points to the `create_table` method with the text "methods for defining and destroying tables". Another callout box points to the `20130916185447_create_birds.rb` file in the db/migrate folder with the text "folder and file created".

# rake db:migrate does...

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "chicken" and the file being edited is "schema.rb". The left sidebar displays the project structure under "Project /Users/mkeane/Desktop/...". The "schema.rb" file is selected in the "db" folder. The code in the editor is as follows:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 20130916185447) do

  create_table "birds", force: true do |t|
    t.string "name"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

Below the editor, the text "schema.rb created" is displayed.

# say, we add feet...

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "chicken" and the current file is "20130916185447\_create\_birds.rb". The project structure on the left shows a directory tree for the "chicken" application, including "app", "db", "lib", and "test" directories. The "db/migrate" folder contains several files: "development.sqlite3", "schema.rb", "seeds.rb", and the currently edited file "20130916185447\_create\_birds.rb". The code editor on the right displays the following migration code:

```
class CreateBirds < ActiveRecord::Migration
  def change
    create_table :birds do |t|
      t.string :name
      t.integer :feet
      t.timestamps
    end
  end
end
```

A yellow horizontal bar highlights the line "t.integer :feet". To the right of the code, the text "we add feet field" is displayed.

# and do `rake db:migrate` again...

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The left sidebar displays the project structure under the 'Project' tab, showing a directory named 'chicken' containing 'app', 'db', 'lib', 'log', 'public', 'test', 'vendor', and several configuration files like '.gitignore', 'config.ru', and 'Gemfile'. The 'db' folder contains 'migrate' which includes migration files such as '20130916185447\_create\_birds.r' and 'schema.rb', along with 'development.sqlite3' and 'seeds.rb'. The right panel shows the code editor with the 'schema.rb' file open. The code defines a database schema for a 'birds' table with columns 'name', 'created\_at', and 'updated\_at'. A note at the top of the file advises against editing it directly and instead recommends using ActiveRecord migrations. The code editor has syntax highlighting and code completion features.

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 20130916185447) do

  create_table "birds", force: true do |t|
    t.string   "name"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

!!!eeek...  
schema.rb  
is unchanged

# rake db:migrate

does not re-run the edited file...it just looks to see if there are any explicitly undone migrations

as we did the Bird migration already...it does not pick up the edits...for safety reasons

so, we need to explicitly rollback the db and rebuild the tables to get **schema.rb** updated

so, we never attempt to change **schema.rb** directly by editing it !

# rake db:rollback does this...

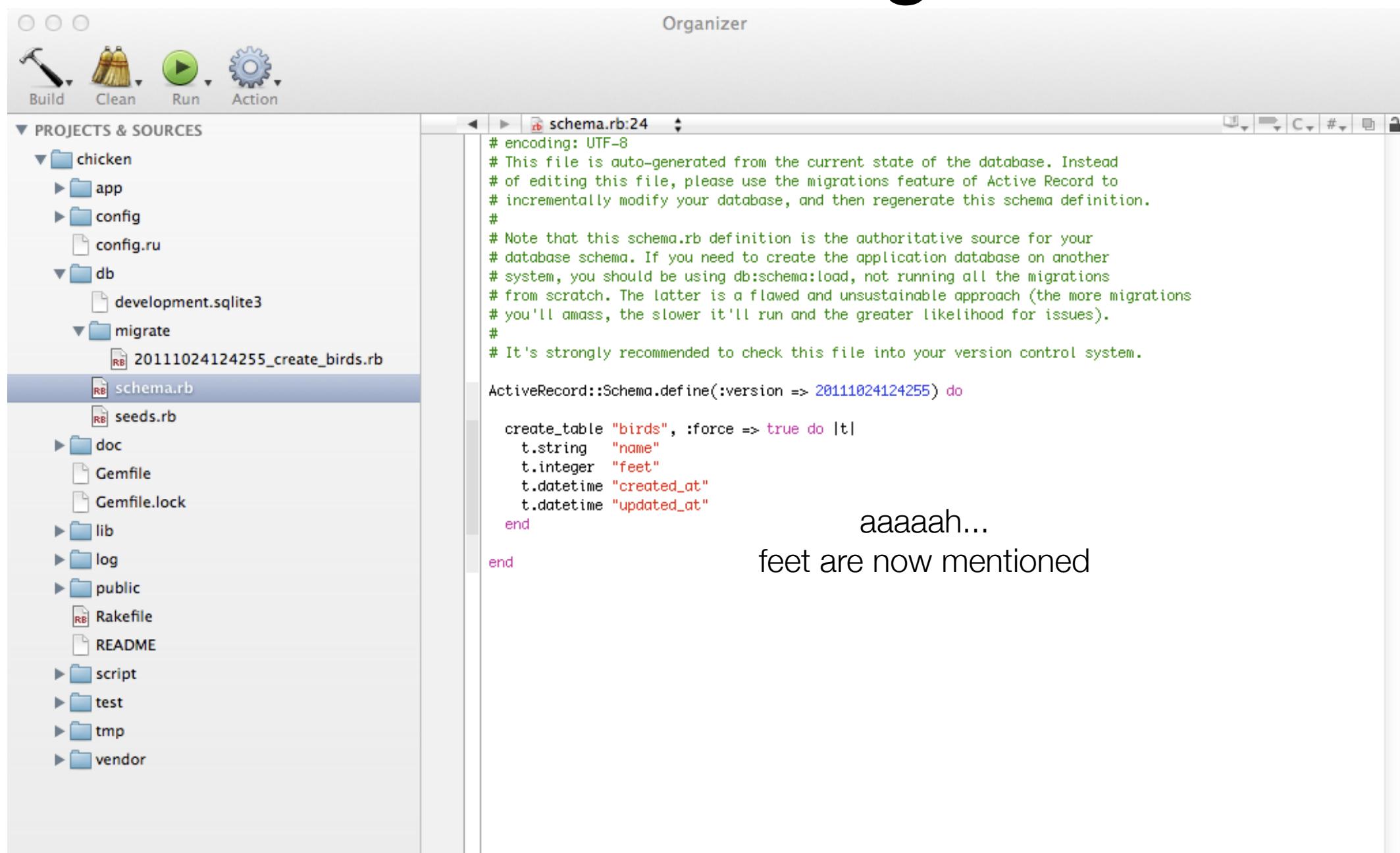
The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar reads "chicken - [~/Desktop/chicken] - .../db/schema.rb - JetBrains RubyMine 3.2.4". The left sidebar displays the project structure under "Project /Users/mkeane/Desktop/...". The "schema.rb" file is selected in the "db" directory. The main editor window shows the following code:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 0) do
end
```

A status message "schema.rb is emptied" is displayed in the bottom right corner of the editor area.

# then redo rake db:migrate ...



The screenshot shows a software interface with a toolbar at the top featuring icons for Build, Clean, Run, and Action. Below the toolbar is a sidebar titled "PROJECTS & SOURCES" containing a project structure for a "chicken" application. The "db" directory contains "development.sqlite3" and "migrate". Inside "migrate" is a file named "20111024124255\_create\_birds.rb". The main pane displays the contents of "schema.rb". The code is as follows:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20111024124255) do
  create_table "birds", :force => true do |t|
    t.string   "name"
    t.integer  "feet"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

In the bottom right corner of the code editor, there is a text overlay that says "aaaaah..." above the sentence "feet are now mentioned".

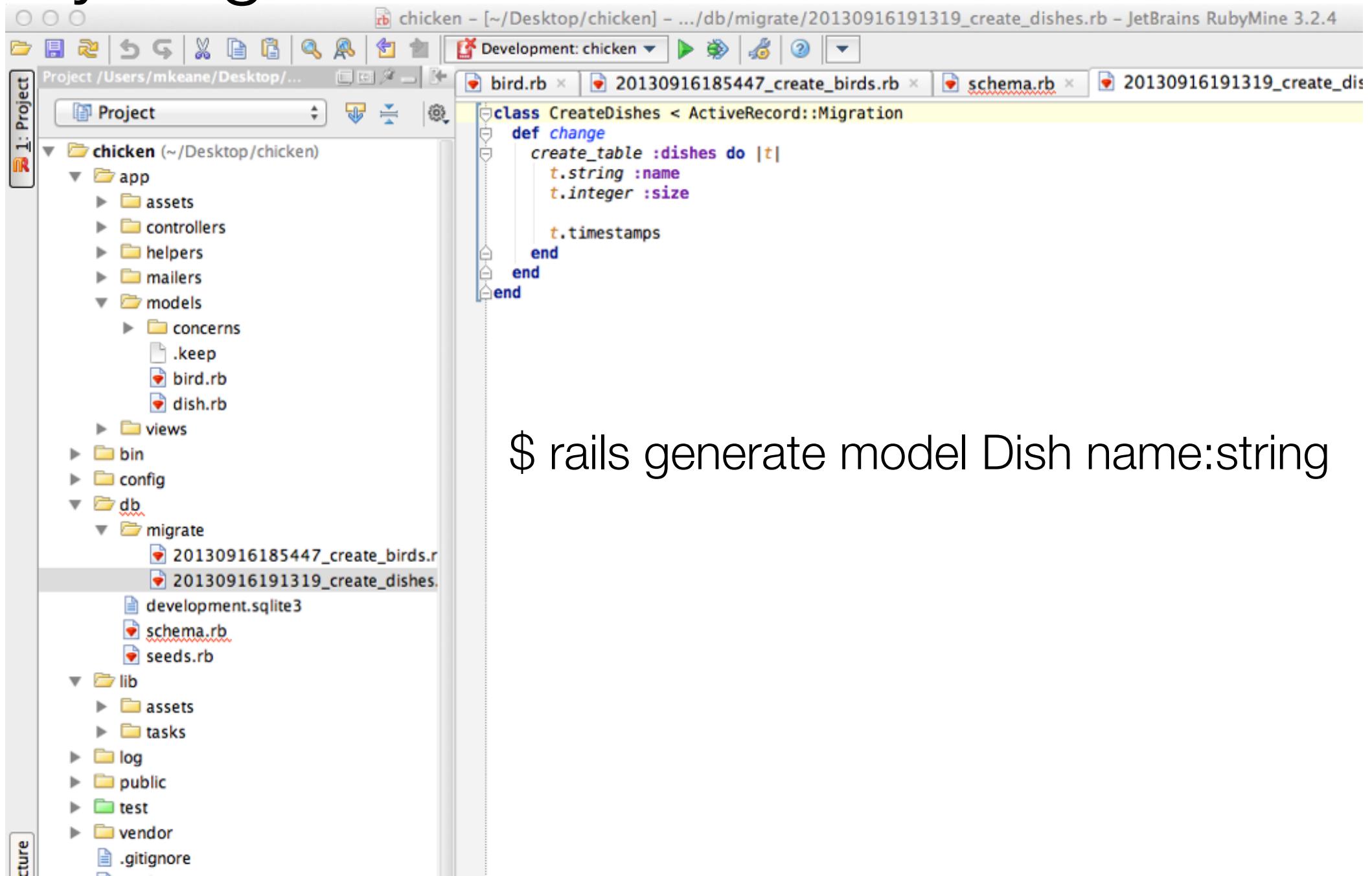
# rake db:migrate...again

and that's not the end of it...if you have done several migrations then rollback goes back through them one step at a time

so, in more complex apps with multiple models you need to make sure you have rolled back right and your **schema.rb** file is what you want

this is only the tip of the iceberg...online discussions suggest that **schema.rb** may be in part build from the current state of the db

# if you generate a new model...

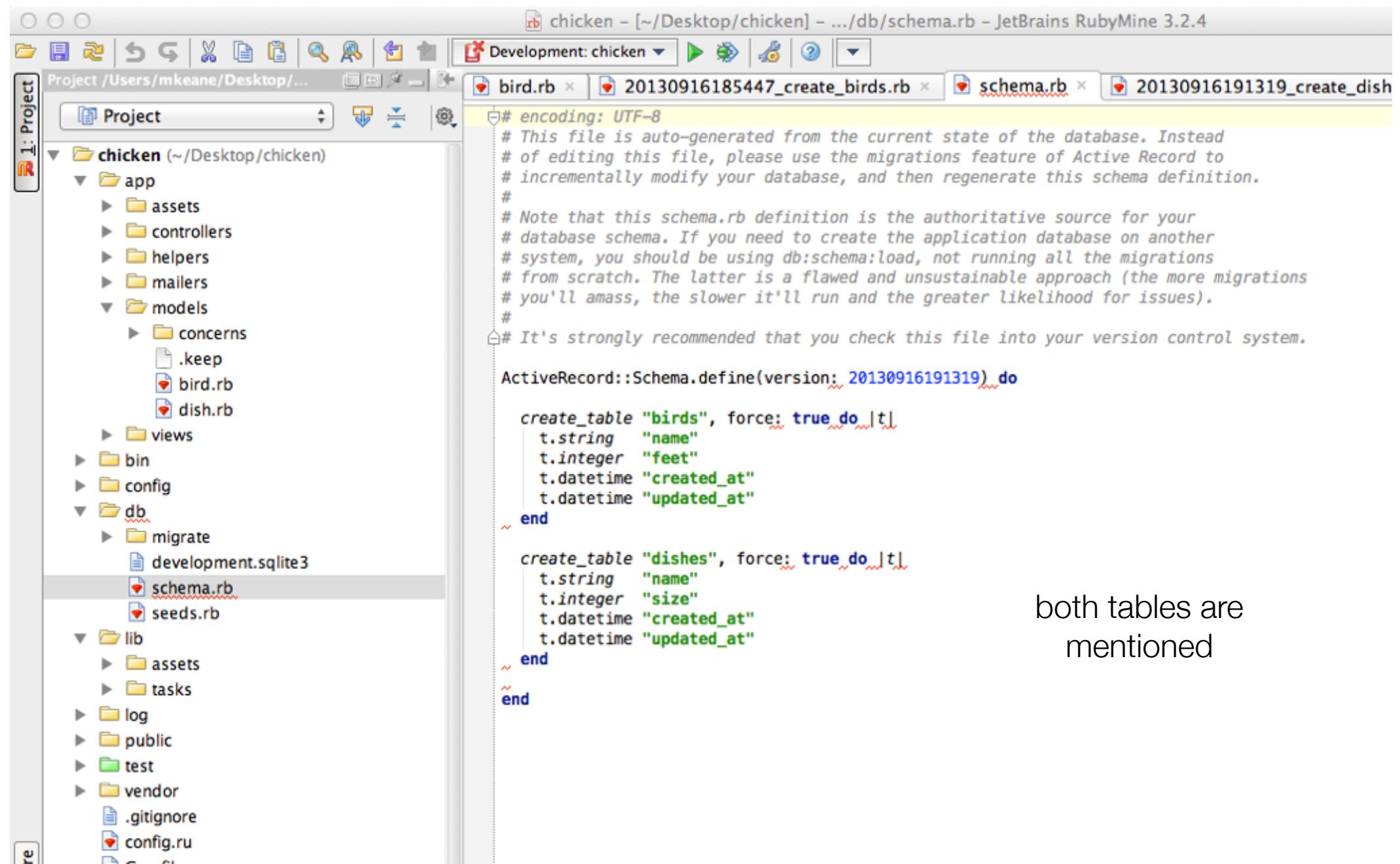


The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "chicken" and the current file is "bird.rb". The left sidebar displays the project structure under "chicken":

- app
  - assets
  - controllers
  - helpers
  - mailers
  - models
    - concerns
    - .keep
    - bird.rb
    - dish.rb
  - views
- bin
- config
- db
  - migrate
    - 20130916185447\_create\_birds.rb
    - 20130916191319\_create\_dishes.rb
  - development.sqlite3
  - schema.rb
  - seeds.rb
- lib
  - assets
  - tasks
- log
- public
- test
- vendor
- .gitignore

```
$ rails generate model Dish name:string
```

# then do rake db:migrate ...



The screenshot shows the JetBrains RubyMine IDE interface. The title bar indicates the project is named "chicken" and the file being edited is "schema.rb". The left sidebar displays the project structure under "Project /Users/mkeane/Desktop/...". The "chicken" directory contains several subfolders: app, bin, config, db, lib, log, public, test, and vendor. Inside the db folder, there are files named "development.sqlite3", "schema.rb", and "seeds.rb". The main editor window shows the content of the "schema.rb" file:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 20130916191319) do

  create_table "birds", force: true do |t|
    t.string   "name"
    t.integer  "feet"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "dishes", force: true do |t|
    t.string   "name"
    t.integer  "size"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

end
```

A status message on the right side of the editor states: "both tables are mentioned".

# then \$ rake db:rollback

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar reads "chicken - [~/Desktop/chicken] - .../db/schema.rb - JetBrains RubyMine 3.2.4". The left sidebar displays the project structure under "Project /Users/mkeane/Desktop/...". The "schema.rb" file is selected in the "Project" view. The main editor window shows the following code:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 20130916185447) do

  create_table "birds", force: true do |t|
    t.string   "name"
    t.integer  "feet"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

A tooltip on the right side of the editor window says "only dishes disappears".

# then rake db:rollback again

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The left sidebar displays the project structure for a 'chicken' application located at '~/Desktop/chicken'. The 'db' directory contains 'migrate', 'development.sqlite3', 'schema.rb', and 'seeds.rb'. The right pane shows the 'schema.rb' file open in the editor. The code in 'schema.rb' is as follows:

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 0) do
end
```

In the bottom right corner of the slide, there is a text overlay: "finally, both are gone, so I can now rebuild them all".

# rake has a lot options...

```
rake about          # List versions of all Rails frameworks and the environment
rake assets:clean   # Remove compiled assets
rake assets:precompile # Compile all the assets named in config.assets.precompile
rake db:create       # Create the database from config/database.yml for the current Rails.env ...
rake db:drop         # Drops the database for the current Rails.env (use db:drop:all to drop al...
rake db:fixtures:load # Load fixtures into the current environment's database.
rake db:migrate      # Migrate the database (options: VERSION=x, VERBOSE=false).
rake db:migrate:status # Display status of migrations
rake db:rollback     # Rolls the schema back to the previous version (specify steps w/ STEP=n).
rake db:schema:dump   # Create a db/schema.rb file that can be portably used against any DB supp...
rake db:schema:load   # Load a schema.rb file into the database
rake db:seed          # Load the seed data from db/seeds.rb
rake db:setup         # Create the database, load the schema, and initialize with the seed data ...
rake db:structure:dump # Dump the database structure to an SQL file
rake db:version        # Retrieves the current schema version number
rake doc:app           # Generate docs for the app -- also available doc:rails, doc:guides, doc:p...
rake log:clear         # Truncates all *.log files in log/ to zero bytes
rake middleware        # Prints out your Rack middleware stack
rake notes             # Enumerate all annotations (use notes:optimize, :fixme, :todo for focus)
rake notes:custom       # Enumerate a custom annotation, specify with ANNOTATION=CUSTOM
rake rails:template    # Applies the template supplied by LOCATION=(/path/to/template) or URL
rake rails:update       # Update configs and some other initially generated files (or use just upd...
rake routes            # Print out all defined routes in match order, with names.
rake secret             # Generate a cryptographically secure secret key (this is typically used t...
rake stats              # Report code statistics (KLOCs, etc) from the application
rake test               # Runs test:units, test:functionals, test:integration together (also avail...
rake test:recent        # Run tests for {:recent=>"test:prepare"} / Test recent changes
```

chicken - [~/Desktop/chicken] - .../Gemfile - JetBrains RubyMine 3.2.4

Project /Users/mkeane/Desktop/... Project chicken (~/Desktop/chicken)

source '<https://rubygems.org>'

```
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'  
gem 'rails', '4.0.0'  
  
# Use sqlite3 as the database for Active Record  
gem 'sqlite3'  
  
# Use SCSS for stylesheets  
gem 'sass-rails', '~> 4.0.0'  
  
# Use Uglifier as compressor for JavaScript assets  
gem 'uglifier', '>= 1.3.0'  
  
# Use CoffeeScript for .js.coffee assets and views  
gem 'coffee-rails', '~> 4.0.0'  
  
# See https://github.com/sstephenson/execjs#readme for more supported runtimes  
# gem 'therubyracer', platforms: :ruby  
  
# Use jquery as the JavaScript library  
gem 'jquery-rails'  
  
# Turbolinks makes following links in your web application faster. Read more: https://github.com/turbolinks/turbolinks  
gem 'turbolinks'  
  
# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder  
gem 'jbuilder', '~> 1.2'  
  
group :doc do  
  # bundle exec rake doc:rails generates the API under doc/api.  
  gem 'sdoc', require: false  
end  
  
# Use ActiveModel has_secure_password  
# gem 'bcrypt-ruby', '~> 3.0.0'  
  
# Use unicorn as the app server  
# gem 'unicorn'  
  
# Use Capistrano for deployment  
# gem 'capistrano', group: :development
```

6: TODO

# Gemfile...

Project /Users/mkeane/Desktop/...      Development: chicken

1: Project      2: Structure      3: TODO

bird.rb      20130916185447\_create\_birds.rb      schema.rb

GEM

```
remote: https://rubygems.org/
specs:
  actionmailer (4.0.0)
  actionpack (= 4.0.0)
  mail (~> 2.5.3)
  actionpack (4.0.0)
  activesupport (= 4.0.0)
  builder (~> 3.1.0)
  erubis (~> 2.7.0)
  rack (~> 1.5.2)
  rack-test (~> 0.6.2)
  activemodel (4.0.0)
  activesupport (= 4.0.0)
  builder (~> 3.1.0)
  activerecord (4.0.0)
  activemodel (= 4.0.0)
  activerecord-deprecated_finders (~> 1.0.2)
  activesupport (= 4.0.0)
  arel (~> 4.0.0)
  activerecord-deprecated_finders (1.0.3)
  activesupport (4.0.0)
  i18n (~> 0.6, >= 0.6.4)
  minitest (~> 4.2)
  multi_json (~> 1.3)
  thread_safe (~> 0.1)
  tzinfo (~> 0.3.37)
  arel (4.0.0)
  atomic (1.1.14)
  builder (3.1.4)
  coffee-rails (4.0.0)
  coffee-script (>= 2.2.0)
  railties (>= 4.0.0.beta, < 5.0)
  coffee-script (2.2.0)
  coffee-script-source
  execjs
  coffee-script-source (1.6.3)
  erubis (2.7.0)
  execjs (2.0.1)
  hike (1.2.3)
  i18n (0.6.5)
  jbuilder (1.5.1)
```

bundle info...

# Some Useful Links...

**<http://rails-nutshell.labs.oreilly.com/ch03.html>**

**[http://guides.rubyonrails.org/active\\_record\\_querying.html](http://guides.rubyonrails.org/active_record_querying.html)**

**<http://stackoverflow.com/questions/1007187/exactly-what-does-rake-dbmigrate-do>**

**[http://railstutorial.org/book#sec:the\\_first\\_application](http://railstutorial.org/book#sec:the_first_application)**

**<http://api.rubyonrails.org/classes/ActiveRecord/Migration.html>**

# Part B

itunes in Rails...the long, dark tea-time of the soul

Part B(i):  
iTunes Rails Design...

# General Design

Models: DB tables for songs and actors (more plausible to create albums on the fly)

Views: there will be the usual views for each action/screen; could also have one for different users (like Buyer, Administrator)

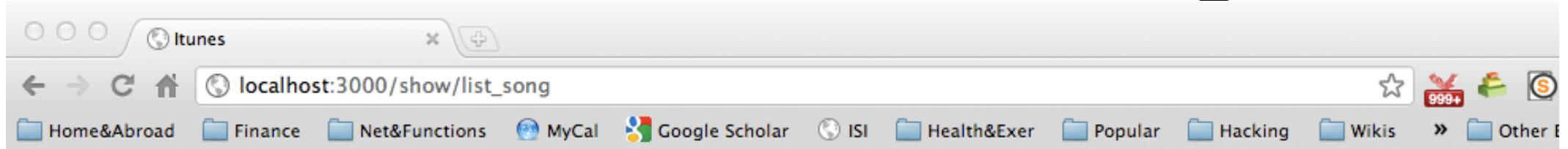
Controllers: methods to show songs/albums/actors, buy, add songs...

Typically, every screen has a method-view pair and a model too...

## Part B(ii):

Some Screens...its not pretty...but hey it works

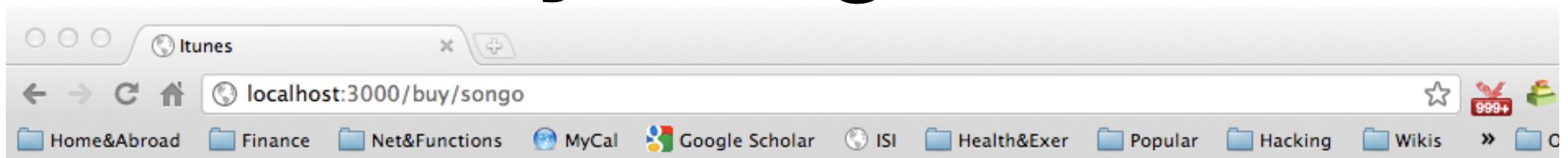
# iTunes: show/list\_song



## Song Listing

- Stairway to Heaven by Led Zep, in Led Zep IV : 1 [Owned by Actor ID 1]
- Holiday by Sex Pistols, in God Save... : 2 [Owned by Actor ID 2]

# iTunes: buy/songo

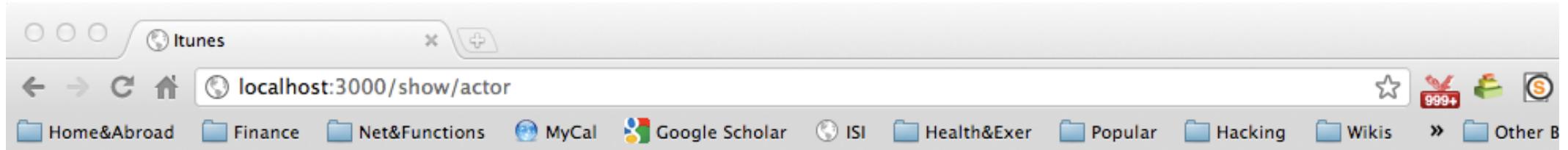


## Buy a Song

Give me your name:

Song you want to buy:

# iTunes: show/actor

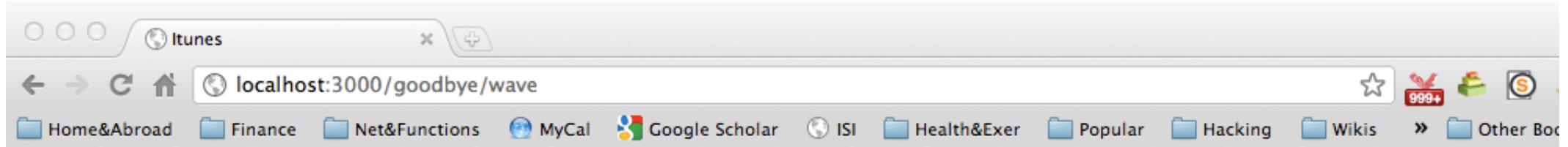


## Actor Listing

- Mark the Admin, likes reggae : 1
- Bloggie the Buyer, likes punk : 2

Say [Goodbye, Mark](#)

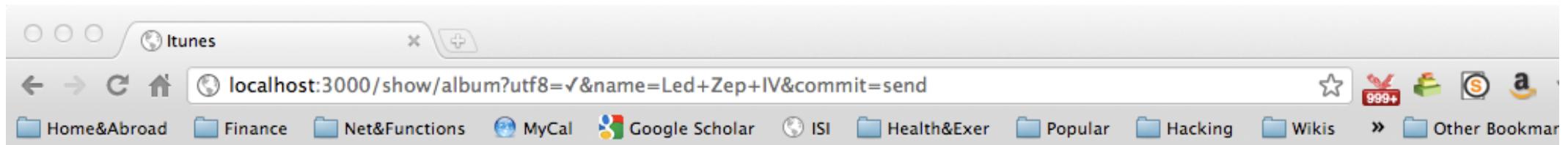
# iTunes: goodbye/wave



**Goodbye#actor\_wave**

bye bye mark

# iTunes: show/album



## Album Listing

Search for an Album named:

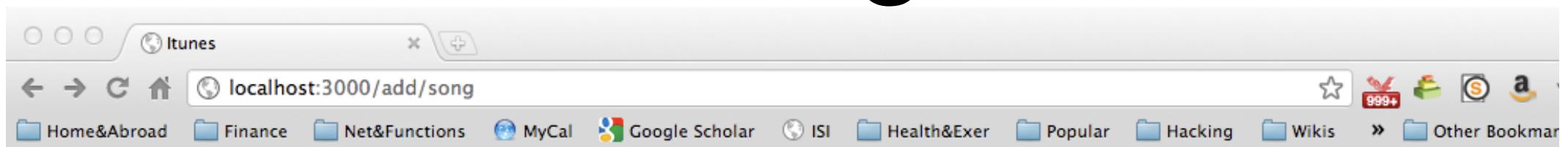
 

The Album "Led Zep IV" has the following tracks:

- Stairway to Heaven by Led Zep

The Album is 4 minutes long.

# iTunes: add/song



## Add a Song to the DB

Please enter Details Below:

Song Name:

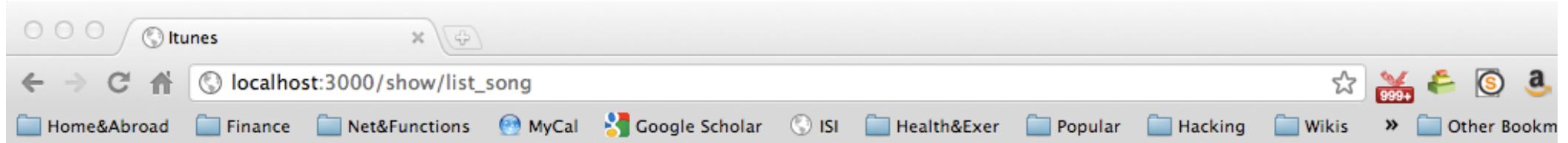
Artist:

Length of Track:

Album its in:

Please [review your inputs](#)

# iTunes: show/list\_song



- Stairway to Heaven by Led Zep, in Led Zep IV : 1 [Owned by Actor ID 1]
- Holiday by Sex Pistols, in God Save... : 2 [Owned by Actor ID 2]
- Gloria by Patti Smith, in Horses : 3 [Owned by Actor ID ]

# Part B(iii):

## Walk through of iTunes Rails

# Setup: Controllers & Models

```
$ rails new itunes
```

```
$ cd itunes
```

```
$ rails generate controller show list_song album actor
```

```
$ rails generate controller buy songo
```

```
$ rails generate controller add song
```

```
$ rails generate model Actor name:string likes:string
```

```
$ rails generate model Song
```

```
$ rails server
```

...if you make a mistake you can do...

```
$ rails destroy controller buy
```

# Setup: Model...

```
$ rails generate model Song
```

```
  invoke  active_record
```

```
  create   db/migrate/20101115115939_create_songs.rb
```

```
  create   app/models/song.rb
```

```
  invoke  test_unit
```

```
  create   test/unit/song_test.rb
```

```
  create   test/fixtures/songs_yml
```

itunes - ~/Desktop/X\_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek11 (Nov 22th).13/RubyLect11.progs/itunes] - .../app/controllers/add

Project /Users/user/Desktop/X... Development: itunes

Project itunes (~/Desktop/X\_Teaching/Ruby:2011)

- app
  - assets
  - controllers
    - add\_controller.rb
    - application\_controller.rb
    - buy\_controller.rb
    - goodbye\_controller.rb
    - show\_controller.rb
  - helpers
  - mailers
  - models
  - views
    - add
      - song.html.erb
      - buy
      - goodbye
      - layouts
      - show
- config
- db
- doc
- lib
- log
- public
  - 404.html

Development: itunes

test.sqlite3 Gemfile application\_controller.rb show\_controller.rb add\_controller.rb buy\_controller.rb goodbye\_controller.rb add\_controller.rb schema.rb

```

class AddController < ApplicationController
  def song
    if params[:artist]
      then @song = Song.create(:artist => params[:artist], :title => params[:title],
      :time => params[:length], :in_album => params[:album]) end
    end
  end
end

```

Started GET "/assets/goodby  
2014-10-11 21:36:46 +0100

Started GET "/assets/buy.cs  
4-10-11 21:36:46 +0100

Started GET "/assets/jquery  
014-10-11 21:36:46 +0100

Started GET "/assets/add.cs  
4-10-11 21:36:46 +0100

Started GET "/assets/show.j  
4-10-11 21:36:46 +0100

Started GET "/assets/jquery  
at 2014-10-11 21:36:46 +010

Started GET "/assets/add.js  
-10-11 21:36:46 +0100

Started GET "/assets/goodby  
2014-10-11 21:36:46 +0100

Started GET "/assets/buy.js  
-10-11 21:36:46 +0100

Started GET "/assets/applic  
at 2014-10-11 21:36:46 +01  
^C[2014-10-11 21:37:57] INF  
[2014-10-11 21:37:57] INFO  
e.  
Exiting

MobileMac:RubyLect11.progs user\$ cd itunes  
 MobileMac:itunes user\$ rails server  
 > Booting WEBrick  
 > Rails 4.1.6 application starting in development on http://0.0.0.0:3000  
 > Run `rails server -h` for more startup options  
 > Notice: server is listening on all interfaces (0.0.0.0). Consider using 127.0.0.1 (--binding option)  
 > Ctrl-C to shutdown server  
 onfig.eager\_load is set to nil. Please update your config/environments/\*.rb files accordingly:  
 \* development - set it to false  
 \* test - set it to false (unless you use a tool that preloads your test environment)  
 \* production - set it to true  
 2014-10-11 23:09:05] INFO WEBrick 1.3.1  
 2014-10-11 23:09:05] INFO ruby 2.0.0 (2014-05-08) [universal.x86\_64-darwin13]  
 2014-10-11 23:09:05] INFO WEBrick::HTTPServer#start: pid=4567 port=3000

itunes2 -

Started GET "/assets/goodby  
2014-10-11 21:36:46 +0100

Started GET "/assets/buy.cs  
4-10-11 21:36:46 +0100

Started GET "/assets/jquery  
014-10-11 21:36:46 +0100

Started GET "/assets/add.cs  
4-10-11 21:36:46 +0100

Started GET "/assets/show.j  
4-10-11 21:36:46 +0100

Started GET "/assets/jquery  
at 2014-10-11 21:36:46 +010

Started GET "/assets/add.js  
-10-11 21:36:46 +0100

Started GET "/assets/goodby  
2014-10-11 21:36:46 +0100

Started GET "/assets/buy.js  
-10-11 21:36:46 +0100

Started GET "/assets/applic  
at 2014-10-11 21:36:46 +01  
^C[2014-10-11 21:37:57] INF  
[2014-10-11 21:37:57] INFO  
e.  
Exiting

MobileMac:itunes2 user\$ ls  
 Gemfile README  
 doc log  
 p  
 Gemfile.lock Rakefile  
 lib put  
 vendor  
 MobileMac:itunes2 user\$  
 MobileMac:itunes2 user\$ ls  
 Gemfile config.ru  
 Gemfile.lock db  
 README doc  
 Rakefile lib  
 app log  
 config public  
 MobileMac:itunes2 user\$



## Organizer

Build Clean Run Action

The Project Navigator on the left shows the file structure of the 'itunes' project:

- itunes2
- itunes
  - app
    - controllers
      - application\_controller.rb
      - buy\_controller.rb
      - input\_controller.rb
      - show\_controller.rb (selected)
    - helpers
    - models
    - views
      - buy
      - end
      - input
      - layouts
      - show
        - album.html.erb
        - list\_song.html.erb
  - config
  - db
  - doc
  - lib
  - log
  - public
    - Rakefile
    - README
  - script
  - test
  - tmp
  - vendor
  - hello

show\_controller.rb:1 class ShowController < ApplicationController

```
class ShowController < ApplicationController
  def list_song
  end

  def album
  end

end
```

in time, we will fill  
in these methods  
with things we want to do

# Load Page

nb: will not load if  
browser is in offline mode;  
even tho' local files

load page in browser:

<http://localhost:3000/show>

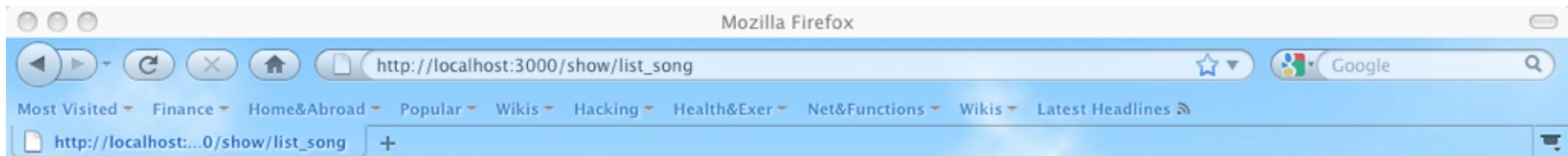


have only given controller  
but not method name

# Load Page

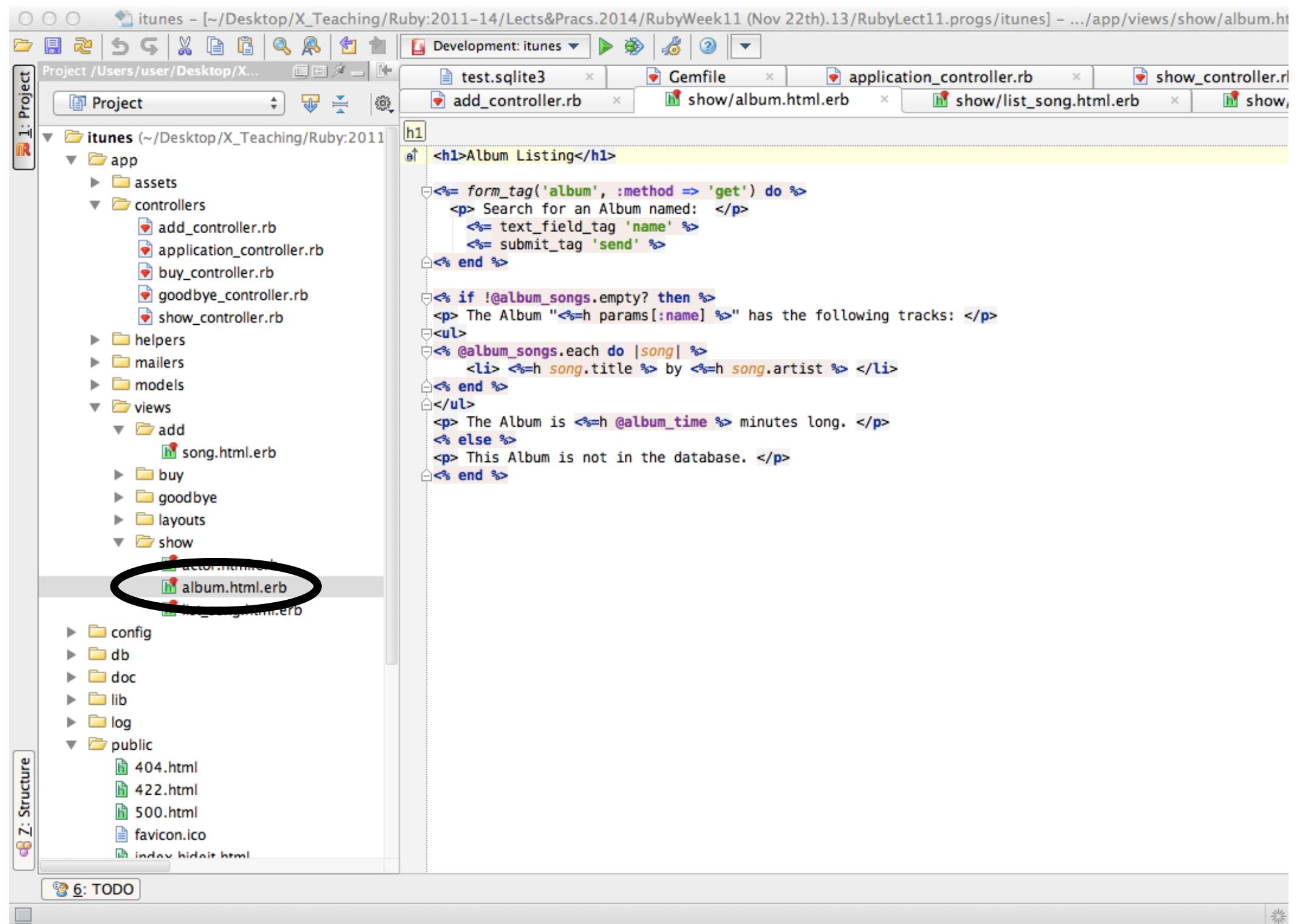
load page in browser:

[http://localhost:3000/show/list\\_song](http://localhost:3000/show/list_song)



**Show#list\_song**

Find me in app/views/show/list\_song.html.erb



## Part B(iv):

Setting up tables, models and starting the db

# What's happening, mammy?

Within Rails' bootage ActiveRecord connects to a database based on what is in **itunes/config/database.yml**

When you have defined your migrations/models (relations), you use **rake db:migrate** to create your **schema.rb** file (but wait til end)

**rake db:seed** can also load in a set of seed records from **seeds.rb**

itunes - [~/Desktop/Teaching/Ruby:2011-13/Lects&Pracs2013/RubyWeek11 (Nov 22th).12/RubyLect11.progs/itunes] - .../config/database.yml -

Project /Users/mkeanu... Project database.yml

1: Project

itunes (~/Desktop/Teaching/Ru app assets controllers add\_controller.rb application\_controller.rb buy\_controller.rb goodbye\_controller.rb show\_controller.rb helpers mailers models views add buy songo.html.erb goodbye wave.html.erb layouts application.html.erb show config environments initializers locales application.rb boot.rb database.yml environment.rb routes.rb db doc

6: TODO

```
# SQLite version 3.x
#   gem install sqlite3
#
#   Ensure the SQLite 3 gem is defined in your Gemfile
#   gem 'sqlite3'
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The database defined as "test" will be re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

you change this if you want to use another db

# Setup tables etc...

We need to set up a bunch of migrations to define our tables for the db

Our initial commands have created a number of files for actors and song:

**\$ rails generate model Actor name:string likes:string**

Look in *itunes/db/migrate*

We need to define the relations between those tables

Look in *itunes/app/models/*

Organizer

Build Clean Run Action

The screenshot shows the Xcode Organizer interface. On the left, there's a file browser tree for a project named "itunes". The tree includes "app", "models" (containing "actor.rb" and "song.rb"), "views", "config", "db" (with "development.sqlite3" and "migrate" folder), "doc", "Gemfile", "Gemfile.lock", and "lib". A black oval highlights the "migrate" folder under "db", which contains two files: "20111024135910\_create\_ac..." and "20111024135938\_create\_so...". On the right, the main pane displays the code for the first migration file:

```
20111024135910_create_actors.rb:1 class CreateActors < ActiveRecord::Migration
  def change
    create_table :actors do |t|
      t.string :name
      t.string :likes

      t.timestamps
    end
  end
end
```

all this is produced  
automatically (along with  
hidden key/id)

Organizer

Build Clean Run Action

itunes

- app
  - assets
  - controllers
  - helpers
  - mailers
- models
  - actor.rb
  - song.rb
- views
  - add
  - buy
  - goodbye
  - layouts
  - show
- config
- config.ru
- db
  - development.sqlite3
  - migrate
    - 20111024135910\_create\_ac...
    - 20111024135938\_create\_so...
    - SCHEMA
    - seeds.rb
    - test.sqlite3
- doc
- Gemfile
- Gemfile.lock
- lib

20111024135910\_create\_actors.rb:1 class CreateActors < ActiveRecord

```
class CreateActors < ActiveRecord::Migration
  def change
    create_table :actors do |t|
      t.string :name
      t.string :likes
      t.string :status
      t.timestamps
    end
  end
end
```

now, lets write in a new field t.string :status

The screenshot shows the Xcode Organizer interface. On the left, the file browser displays the project structure under the 'itunes' target. The 'db/migrate' folder is highlighted with a black oval. Inside this folder, two migration files are visible: '20111024135910\_create\_actors.rb' and '20111024135938\_create\_songs.rb'. The '20111024135910\_create\_actors.rb' file is open in the main editor area, showing its code. The code defines a 'CreateActors' class that inherits from 'ActiveRecord::Migration'. The 'change' method creates a table named 'actors' with columns for 'name', 'likes', and 'status', and includes timestamps. A cursor is visible in the code editor at the end of the 'status' column definition. The status bar at the bottom indicates the file is at line 1.

Organizer

Build Clean Run Action

The screenshot shows the Xcode Organizer window. On the left, the file structure of the 'itunes' project is displayed, including 'app', 'models' (with 'actor.rb' and 'song.rb'), 'views', 'config', 'db' (with 'development.sqlite3'), and 'migrate' (containing '20111024135910\_create\_act...' and '20111024135938\_create\_s...'). A black oval highlights the 'migrate' folder. The main pane shows the code for '20111024135938\_create\_songs.rb'. The code defines a migration class 'CreateSongs < ActiveRecord::Migration' with a 'change' block that creates a 'songs' table with columns for title, in\_album, time, artist, actor\_id, and timestamps.

```
class CreateSongs < ActiveRecord::Migration
  def change
    create_table :songs do |t|
      t.string :title
      t.string :in_album
      t.integer :time
      t.string :artist
      t.integer :actor_id
      t.timestamps
    end
  end
end
```

so add these  
nb; here we refer to  
actor id/key

# Types in ActiveRecord

Abstract Type	SQLite Type	Ruby Class
integer	integer	Fixnum
decimal	decimal	BigDecimal
float	float	Float
string	varchar(255)	String
text	text	String
binary	blob	String
date	date	Date
datetime	datetime	Time
timestamp	datetime	Time
time	time	Time
boolean	boolean	TrueClass/FalseClass

# Setup tables etc...

We need to set up a bunch of migrations to define our tables for the db

Our initial commands have created a number of files for actors and song:

**\$ rails generate model Actor name:string likes:string**

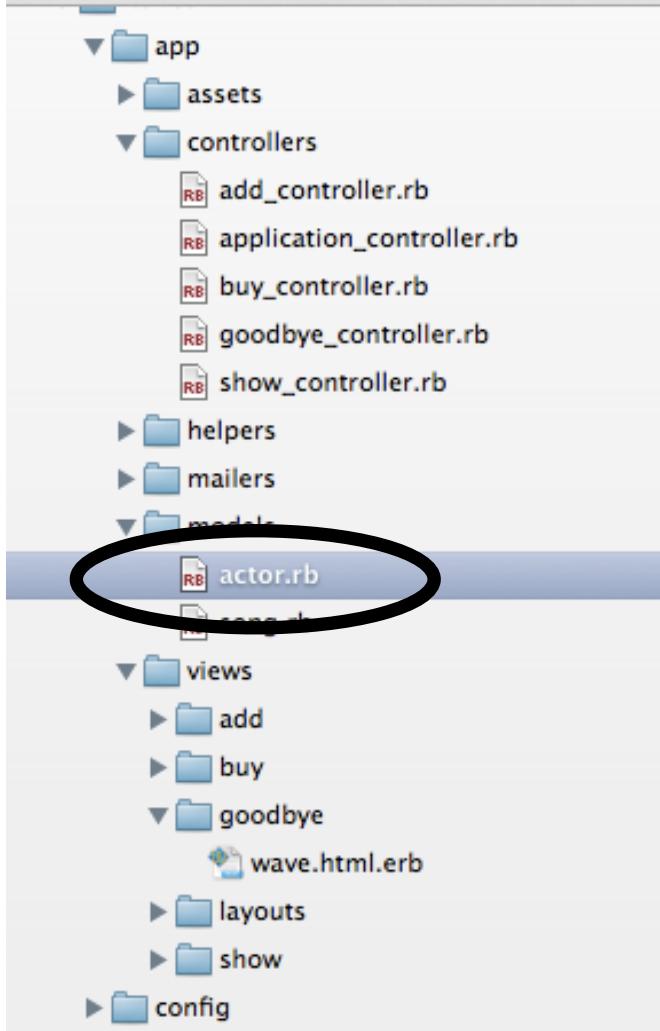
Look in *itunes/db/migrate*

We need to define the relations between those tables

Look in *itunes/app/models/*

Organizer

Build Clean Run Action



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

relation between actor  
and songs

app

assets

controllers

- add\_controller.rb
- application\_controller.rb
- buy\_controller.rb
- goodbye\_controller.rb
- show\_controller.rb

helpers

mailers

models

- actor.rb
- song.rb

views

- add
- buy
- goodbye
  - wave.html.erb
- layouts
- show

config

config.ru

db

- development.sqlite3
- migrate
  - 20111024135910\_create\_act...
  - 20111024135938\_create\_so...
- schema.rb

Organizer

Build Clean Run Action

relation between song and actor

```
song.rb:1 class Song < ActiveRecord::Base
  belongs_to :actor
end
```

app

- assets
- controllers
  - add\_controller.rb
  - application\_controller.rb
  - buy\_controller.rb
  - goodbye\_controller.rb
  - show\_controller.rb
- helpers
- mailers
- models
  - song.rb
- views
  - add
  - buy
  - goodbye
    - wave.html.erb
  - layouts
  - show

config

- config.ru

db

- development.sqlite3
- migrate
  - 20111024135910\_create\_act...
  - 20111024135938\_create\_so...
- schema.rb

# Types in ActiveRecord

Do not do **rake db:migrate**  
until you have all your models  
and migrations are defined



**rake db:migrate** creates a **schema.rb** file that records the current version of the migrations

...never change  
**schema.rb** by hand

...never create  
supporting files for  
controller by hand

# Models recipe...

- 1) set up models
- 2) check and/or edit migration files
- 3) include db relations (if required) in model files (eg actor.rb)
- 4) edit your seed files (will return to this later)
- 5) do **\$ rake db:migrate**
- 6) do **\$ rake db:seed**

...then you can move to your browser and start running...

Grab File Edit Capture Window Help

Organizer

Build Clean Run Action

itunes2

```

schema.rb:1
# This file is auto-generated from the current state of the database. Instead of editing this file,
# please use the migrations feature of Active Record to incrementally modify your database, and
# then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your database schema. If you need
# to create the application database on another system, you should be using db:schema:load, not running
# all the migrations from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll mass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20101105121308) do

  create_table "actors", :force => true do |t|
    t.string "name"
    t.string "likes"
    t.string "status"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "songs", :force => true do |t|
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end

```

schema.rb created

database firing up

```

CREATE TABLE "schema_migrations" ("version" varbinary)
PRAGMA index_list("schema_migrations")
CREATE UNIQUE INDEX "unique_schema_migrations"
SELECT name
FROM sqlite_master
WHERE type = 'table' AND NOT name = 'sqlite_sequence'

SELECT version FROM schema_migrations
Migrating to CreateActors (20101105121001)
CREATE TABLE "actors" ("id" INTEGER PRIMARY KEY
ar(255), "likes" varchar(255), "status" varchar(255), "created_
INSERT INTO schema_migrations (version) VALUES
Migrating to CreateSongs (20101105121300)
CREATE TABLE "songs" ("id" INTEGER PRIMARY KEY
datetime, "updated_at" datetime)
INSERT INTO schema_migrations (version) VALUES
SELECT name
FROM sqlite_master
WHERE type = 'table' AND NOT name = 'sqlite_sequence'

SELECT version FROM schema_migrations
SELECT name
FROM sqlite_master
WHERE type = 'table' AND NOT name = 'sqlite_sequence'

PRAGMA index_list("actors")
PRAGMA index_list("songs")

Processing AddController#index (for 127.0.0.1 at 2010-11-05 12:00:00)
ActionController::UnknownAction (No action responded to index)

Rendering rescues/layout (not_found)

Processing AddController#song (for 127.0.0.1 at 2010-11-05 13:00:00)
ActiveRecord::UnknownAttributeError (unknown attribute: artist)
app/controllers/add_controller.rb:16:in `song'

Rendered rescues/_trace (194.7ms)
Rendered rescues/_request_and_response (0.7ms)
Rendering rescues/layout (internal_server_error)

```

Terminal — tcsh — 225x20

```

[MouseKing6-3:~/Desktop] markkearn% ls
Apr29 Memory&Creativity          CogPsych017:11
Batch 45 review summary.xls       GIBSON
CSPPracticeAug18:2010 copy        India Talk July7 copy
Cog Psych Lecture 7 (21 Oct).copy.ppt Indie V2 copy
CogPsyExams&Paper               RubyJuly8.2 copy
[MouseKing6-3:~/Desktop] markkearn% cd itunes2
[MouseKing6-3:~/Desktop/itunes2] markkearn% rake db:migrate
(in /Users/markkearn/Desktop/itunes2)
== CreateActors: migrating =====
-- create_table(:actors)
-> 0.0043s
== CreateActors: migrated (0.0049s)

== CreateSongs: migrating =====
-- create_table(:songs)
-> 0.0042s

```

rake db:migrate

RubyWeekexamMon copy	oct1.tiff	projectGuide.pdf
Similarity Papers	actors.tiff	sc.tiff
Suprise Paper	hello	song2.tiff
TAX	itunes	songs.tiff
XOld	itunes2	trailsJune29 copy

itunes - [~/Desktop/Teaching/Ruby:2011-13/Lects&Pracs2013/RubyWeek11 (Nov 22th).12/RubyLect11.progs/itunes] - .../db/schema.rb - JetB

Project /Users/mkea... Development: itunes

Project iTunes (~/Desktop/Teaching/Ru

app assets controllers add\_controller.rb application\_controller.rb buy\_controller.rb goodbye\_controller.rb show\_controller.rb helpers mailers models views add buy songo.html.erb goodbye wave.html.erb layouts application.html.er show config db migrate development.sqlite3 schema.rb seeds.rb test.sqlite3 doc lib log public

show/list\_song.html.erb show/album.html.erb show/actor.html.erb buy/songo.html.erb application.html.erb development.sqlite3 test.sqlite3 data

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20111024135938) do

  create_table "actors", :force => true do |t|
    t.string "name"
    t.string "likes"
    t.string "status"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "songs", :force => true do |t|
    t.string "title"
    t.string "in_album"
    t.integer "time"
    t.string "artist"
    t.integer "actor_id"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

end
```

6: TODO 1:1

# In Rails 4.0 routes.rb

The screenshot shows a Java-based IDE interface with multiple tabs open. The tabs include 'show/list\_song.html.erb', 'show/album.html.erb', 'show/actor.html.erb', 'application.html.erb', 'development.sqlite3', 'test.sqlite3', and 'database.yml'. The main code editor area displays the contents of the 'routes.rb' file for the 'itunes' application. The code is as follows:

```
itunes::Application.routes.draw do
  get "goodbye/wave"
  get "buy/songo"
  get "add/song"
  get "show/list_song"
  get "show/album"
  get "show/actor"

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

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

  # Sample of named route:
  #   match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
  # This route can be invoked with purchase_url(:id => product.id)

  # You can have the root of your site routed with "root"
  # just remember to delete public/index.html.
  root :to => 'show#list_song'

  # See how all your routes lay out with "rake routes"

  # This is a legacy wild controller route that's not recommended for RESTful applications.
  # Note: This route will make all actions in every controller accessible via GET requests.
  # match ':controller(/:action(/:id(.:format)))'
end
```

On the left side of the interface, there is a project structure tree labeled '1: Project'. It shows the directory structure of the 'itunes' application, including 'app' (with 'assets', 'controllers' containing files like 'add\_controller.rb', 'application\_controller.rb', 'buy\_controller.rb', 'goodbye\_controller.rb', 'show\_controller.rb'), 'views' (with 'add', 'buy' (containing 'songo.html.erb'), 'goodbye' (containing 'wave.html.erb'), 'layouts' (containing 'application.html.erb'), 'show'), and 'config' (with 'environments', 'initializers', 'locales' (containing 'application.rb', 'boot.rb'), 'database.yml', 'environment.rb', 'routes.rb'). The 'routes.rb' file is currently selected in the tree.

Text on the right side of the code editor area reads: "all this is produced automatically".

itunes - [~/Desktop/Teaching/Ruby:2011-13/Lects&Pracs2013/RubyWeek11 (Nov 22th).12/RubyLect11.progs/itunes] - .../config/routes.rb - Je

Project /Users/mke... Project

1: Project

itunes (~/Desktop/Teaching/Ru app assets controllers add\_controller.rb application\_controller.i buy\_controller.rb goodbye\_controller.rb show\_controller.rb helpers mailers models views add buy songo.html.erb goodbye wave.html.erb layouts application.html.er show config environments initializers locales application.rb boot.rb database.yml environment.rb routes.rb lib doc

show/list\_song.html.erb show/album.html.erb show/actor.html.erb application.html.erb development.sqlite3 test.sqlite3 database.yml

```
Itunes::Application.routes.draw do
  get "goodbye/wave"
  get "buy/songo"
  get "add/song"
  get "show/list_song"
  get "show/album"
  get "show/actor"

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

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

  # Sample of named route:
  #   match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
  # This route can be invoked with purchase_url(:id => product.id)

  # You can have the root of your site routed with "root"
  # just remember to delete public/index.html.
  root :to => 'show#list_song'

  # See how all your routes lay out with "rake routes"

  # This is a legacy wild controller route that's not recommended for RESTful applications.
  # Note: This route will make all actions in every controller accessible via GET requests.
  # match ':controller(/:action(/:id(.:format)))'
end
```

6: TODO

delete comment and add action

Organizer

Build Clean Run Action

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ruby on Rails: Welcome aboard</title>
    <style type="text/css" media="screen">
      body {
        margin: 0;
        margin-bottom: 25px;
        padding: 0;
        background-color: #f0f0f0;
        font-family: "Lucida Grande", "Bitstream Vera Sans", "Verdana";
        font-size: 13px;
        color: #333;
      }

      h1 {
        font-size: 28px;
        color: #000;
      }

      a {color: #03c}
      a:hover {
        background-color: #03c;
        color: white;
        text-decoration: none;
      }

      #page {
        background-color: #f0f0f0;
        width: 750px;
        margin: 0;
        margin-left: auto;
        margin-right: auto;
      }

      #content {
        float: left;
        background-color: white;
        border: 3px solid #aaa;
        border-top: none;
        padding: 25px;
        width: 500px;
      }
    </style>
  </head>
  <body>
    <h1>Ruby on Rails: Welcome aboard</h1>
    <div id="content">
      <p>This is the default starting point for your application. You can change or remove it as you like, and add new files as needed. If you're using a database, you'll find it in db/migrate. If you want to change the look and feel of the site, you can do that in the public/assets/stylesheets directory. You can also add new routes in config/routes.rb, and new controllers in app/controllers.
    </div>
  </body>
</html>
```

hide this file  
if there?

Organizer

Build Clean Run Action

itunes

app

config

- application.rb
- boot.rb
- database.yml
- environment.rb
- environments
- initializers
- locales

routes.rb

config.ru

db

- development.sqlite3
- migrate
- schema.rb
- seeds.rb
- test.sqlite3

doc

- Gemfile
- Gemfile.lock

lib

log

public

- 404.html
- 422.html
- 500.html
- favicon.ico
- index.hideit.html
- robots.txt

routes.rb:25

```
Itunes::Application.routes.draw do
  get "goodbye/wave"
  get "buy/songo"
  get "add/song"
  get "show/list_song"
  get "show/album"
  get "show/actor"

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

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

  # Sample of named route:
  #   match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
  # This route can be invoked with purchase_url(:id => product.id)

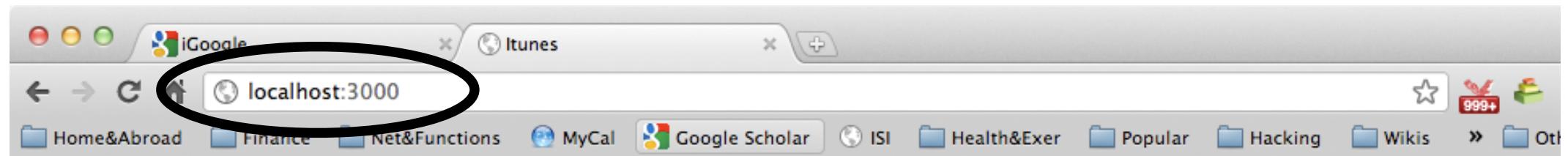
  # You can have the root of your site routed with "root"
  # just remember to delete public/index.html.
  root :to => 'show#list_song'

  # See how all your routes lay out with "rake routes"

  # This is a legacy wild controller route that's not recommended for RESTful applications.
  # Note: This route will make all actions in every controller accessible via GET requests.
  # match ':controller(/:action(/:id(.:format)))'
end
```

looks for index.html

by changing name.



## Song Listing

- Stairway to Heaven by Led Zep, in Led Zep IV : 1 [Owned by Actor ID 1]
- Holiday by Sex Pistols, in God Save... : 2 [Owned by Actor ID 2]
- Gloria by Patti Smith, in Horses : 3 [Owned by Actor ID ]

# Part B(v):

## Editing the Controllers & Views

# Controllers & Views

most of our controllers are set up with empty methods and without view templates

so, we need to write the methods and the ERb to display what we want to display

there should be no surprises here as we do not do anything beyond the functionality we saw in earlier rails apps

I am ignoring layout issues, for simplicity

# Add Controller

The screenshot shows the JetBrains RubyMine IDE interface. The top bar displays the project name 'itunes' and the file path '.../app/controllers/add\_controller.rb'. The left sidebar shows the project structure under 'Project /Users/user/Desktop/X...'. A black oval highlights the file 'add\_controller.rb' located in the 'app/controllers/add' directory. The main editor window displays the following Ruby code:

```
class AddController < ApplicationController
  def song
    if params[:artist]
      then @song = Song.create(:artist => params[:artist], :title => params[:title],
      :time => params[:length], :in_album => params[:album]) end
    end
  end
```

The code uses a ternary operator to create a 'Song' record only if the 'artist' parameter is present in the request parameters. The code editor has syntax highlighting and red squiggly lines under the word 'then', indicating a syntax error or warning.

Annotations on the right side of the screen provide explanatory text:

- A yellow box contains the text: "only create a record if you have been given an input"
- A blue box contains the text: "we don't use @song but you may want it later"

# Add View

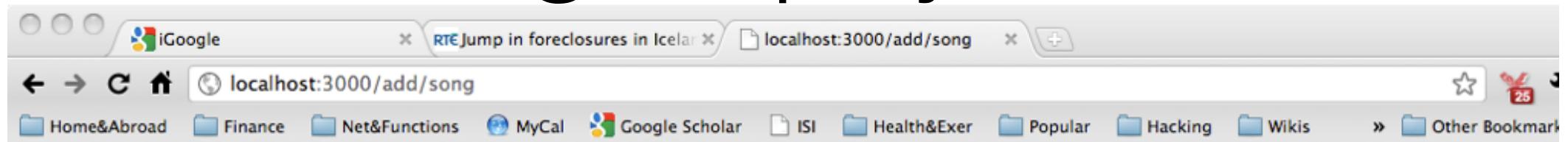
The screenshot shows the JetBrains IDE interface with the 'itunes' project open. The left sidebar displays the project structure under '1: Project'. The 'views' folder contains a subfolder 'add' which has a file named 'song.html.erb'. This file is circled with a black oval. The main editor window shows the contents of 'song.html.erb'.

```
<h1>Add a Song to the DB</h1>
<p> Please enter Details Below: </p>
<=? form_tag('song', :method => 'get') do ?>
  <p> Song Name:<br/>
    <=? text_field_tag 'title' ?>
  <p> Artist:<br/>
    <=? text_field_tag 'artist' ?>
  <p> Length of Track:<br/>
    <=? text_field_tag 'length' ?>
  <p> Album its in:<br/>
    <=? text_field_tag 'album' ?>
  <=? submit_tag 'send' ?>
<=? end ?>
<p> Please
  <=? link_to 'review your inputs', :controller => "show", :action => "list_song" ?></p>
```

sets up the form for entering data

links us to the show/list\_song method

# add/song Displays...



## Add a Song to the DB

Please enter Details Below:

Song Name:

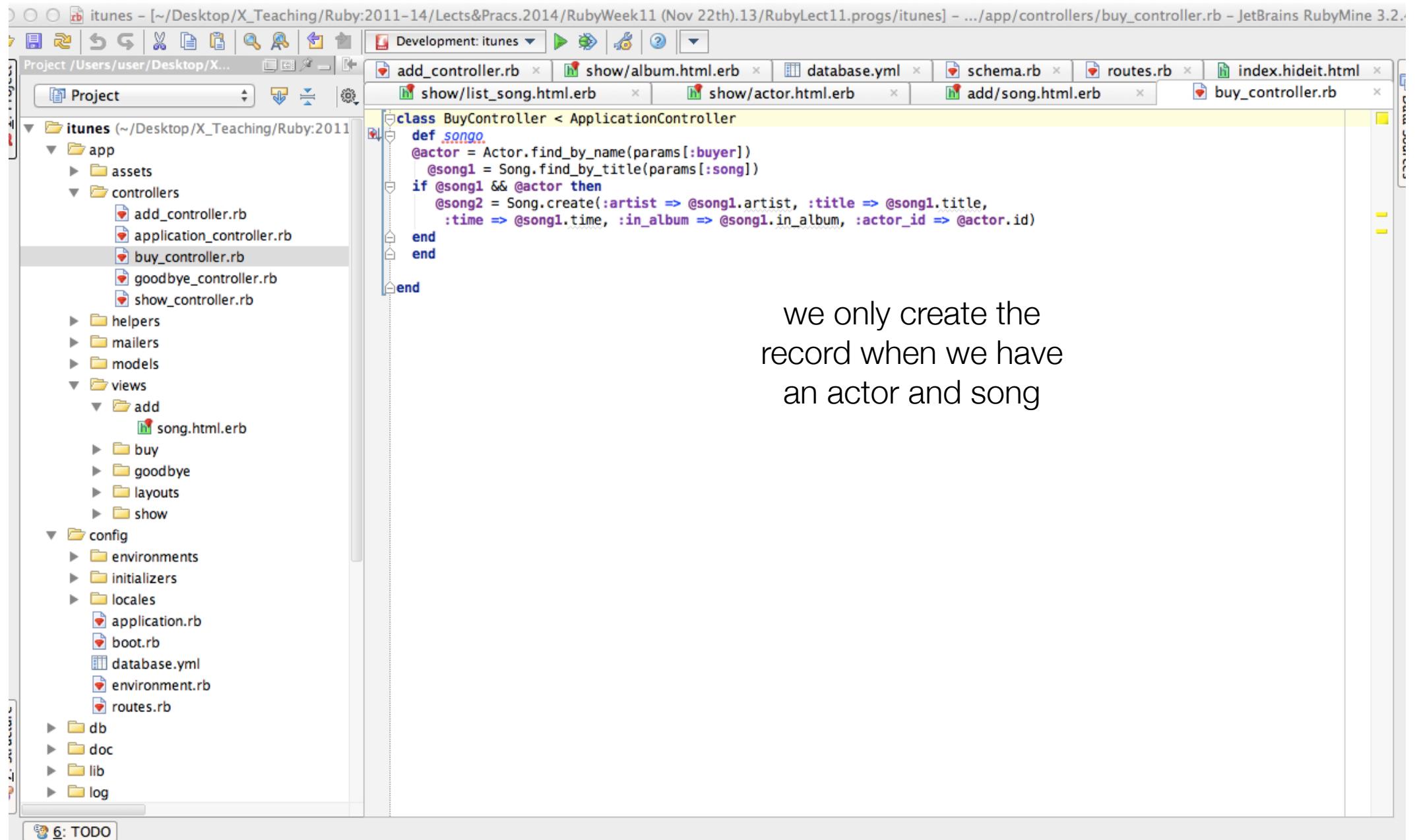
Artist:

Length of Track:

Album its in:

Please [review your inputs](#)

# Buy Controller



The screenshot shows the JetBrains RubyMine 3.2.1 IDE interface. The left sidebar displays the project structure for a Rails application named 'itunes'. The 'controllers' directory contains several files: add\_controller.rb, application\_controller.rb, buy\_controller.rb (which is currently selected), goodbye\_controller.rb, and show\_controller.rb. The 'views' directory contains subfolders for add, buy, goodbye, layouts, and show, each containing an HTML template file (e.g., song.html.erb). The right pane shows the code for the buy\_controller.rb file:

```
class BuyController < ApplicationController
  def songo
    @actor = Actor.find_by_name(params[:buyer])
    @song1 = Song.find_by_title(params[:song])
    if @song1 && @actor then
      @song2 = Song.create(:artist => @song1.artist, :title => @song1.title,
                           :time => @song1.time, :in_album => @song1.in_album, :actor_id => @actor.id)
    end
  end
end
```

A tooltip or explanatory text is overlaid on the right side of the code editor, stating: "we only create the record when we have an actor and song".

# Buy/songo View

The screenshot shows the JetBrains RubyMine IDE interface. The left sidebar displays the project structure for a 'itunes' application, including 'app', 'config', 'db', 'doc', and 'lib' directories. The 'views' directory under 'app' contains 'add', 'buy', and 'show' sub-directories. The 'buy' directory is currently selected, and its contents are shown in the main editor area. The file 'songo.html.erb' contains the following Ruby code:

```
<h1>Buy a Song</h1>

<%= form_tag('songo', :method => 'get') do %>
  <p> Give me your name: </p>
  <%= text_field_tag 'buyer' %>
  <p> Song you want to buy: </p>
  <%= text_field_tag 'song' %>
  <%= submit_tag 'buy' %>
<% end %>

<% if params[:buyer] && params[:song] then %>
  <% if @song1 && @actor then %>
    <p> "<=?h params[:buyer] ?>" you are now the proud owner of "<=?h params[:song] ?>" </p>
  <% elsif @actor.nil? then %>
    <p> I am afraid that you are not in my database. </p>
  <% elsif @song1.nil? then %>
    <p> I am afraid I do not have this song: <=?h params[:song] ?>. </p>
  <% end %>
<% end %>
```

Annotations on the right side of the code provide explanatory text:

- "we only display when we been given an input" points to the conditionals at the bottom of the code.
- "we test for what is coming back from the data base" points to the same conditionals.
- "be aware that :find returns arrays and find\_by... a record" points to the first conditional block.
- "so, you may need to deal with a nil or a []" points to the first conditional block.

# buy/songo Displays...

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/buy/songo`. The page title is "Buy a Song". The form contains two input fields: one for the name and one for the song title, followed by a "buy" button.

Engineering & Computer Sci X Cycling Training Log, Calor X Itunes X Active Record Query Interf X

localhost:3000/buy/songo

Apps Home AIB Internet Banking Home&Abroad Finance Net&Functions Google Scholar ISI Health&Exer Popular

## Buy a Song

Give me your name:

Song you want to buy:

# show Controller

The screenshot shows the JetBrains RubyMine IDE interface. The left sidebar displays the project structure for 'itunes'. The 'app/controllers' folder contains several files: add\_controller.rb, application\_controller.rb, buy\_controller.rb, goodbye\_controller.rb, and show\_controller.rb. The 'views' folder contains 'add' and 'show' subfolders, each with its own HTML files (e.g., song.html.erb, songo.html.erb). The right pane shows the code for 'show\_controller.rb'. The code defines a class ShowController < ApplicationController with three methods: list\_song, album, and actor.

```
class ShowController < ApplicationController
  def list_song
    @songs = Song.all
  end

  def album
    @name = params[:name]
    if !@name.blank?
      then @album_songs = Song.where(in_album: @name)
          @album_time = (@album_songs.collect {|song| song.time}).sum
      else @album_songs = [] end
    end

  def actor
    @actors = Actor.all
  end
end
```

method for each page

# show/actor View

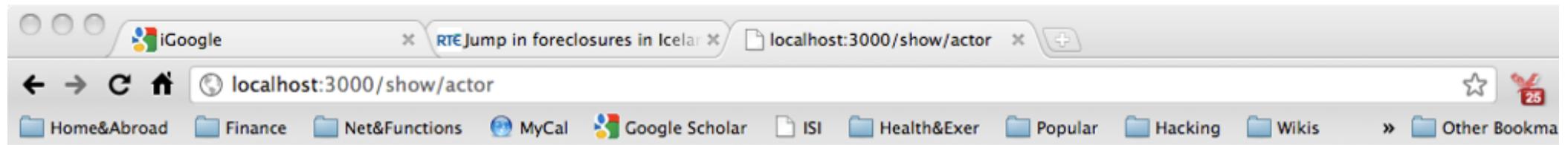
The screenshot shows the JetBrains RubyMine IDE interface. The left sidebar displays the project structure under 'itunes'. The 'views' folder contains 'add', 'buy', 'goodbye', 'layouts', and 'show' subfolders. The 'show' folder is currently selected, and its contents include 'actor.html.erb', 'album.html.erb', and 'list\_song.html.erb'. The main editor window shows the content of 'actor.html.erb'. The code is an ERB template:

```
<h1>Actor Listing</h1>
<ul>
  <% @actors.each do |actor| %>
    <li> <%= h actor.name %> the <%= h actor.status %>, likes <%= h actor.likes %> : <%= h actor.id %> </li>
  <% end %>
</ul>

<p> Say
  <%= link_to 'Goodbye, Mark', :controller => 'goodbye', :action => 'wave', :id => "foobar" %></p>
```

The code editor has syntax highlighting for HTML tags (red), ERB tags (blue), and variable names (orange). The 'Structure' tool window on the right provides navigation and search features.

# Displays...



## Actor Listing

- Mark the Admin, likes reggae : 1
- Bloggie the Buyer, likes punk : 2
- Mark the Admin, likes reggae : 3
- Bloggie the Buyer, likes punk : 4
- Mark the Admin, likes reggae : 5
- Bloggie the Buyer, likes punk : 6

# show/album View

The screenshot shows the JetBrain RubyMine IDE interface. The left sidebar displays the project structure for 'itunes' located at `~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek11 (Nov 22th).13/RubyLect11.progs/itunes`. The 'views' folder under 'app' contains subfolders 'add', 'buy', and 'show'. The 'show' folder is currently selected, and its contents are shown in the main editor area.

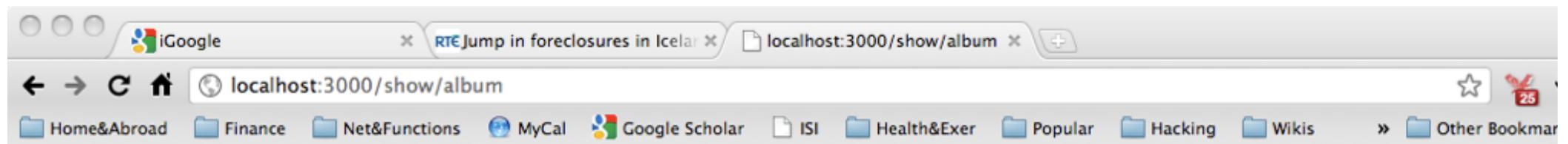
The main editor area shows the content of `show/album.html.erb`:

```
<h1>Album Listing</h1>

<=? form_tag('album', :method => 'get') do ?>
  <p> Search for an Album named: </p>
  <=? text_field_tag 'name' ?>
  <=? submit_tag 'send' ?>
<?= end ?>

<?= if !@album_songs.empty? then ?>
  <p> The Album "<=?=h params[:name] ?>" has the following tracks: </p>
  <ul>
    <?= @album_songs.each do |song| ?>
      <li> <=?=h song.title ?> by <=?=h song.artist ?> </li>
    <?= end ?>
  </ul>
  <p> The Album is <=?=h @album_time ?> minutes long. </p>
<?= else ?>
  <p> This Album is not in the database. </p>
<?= end ?>
```

# Displays...



Search for an Album named:

The Album "Led Zep IV" has the following tracks:

- Stairway to Heaven by Led Zep

The Album is 4 minutes long.

# show/song\_list View

The screenshot shows the JetBrains RubyMine IDE interface with the following details:

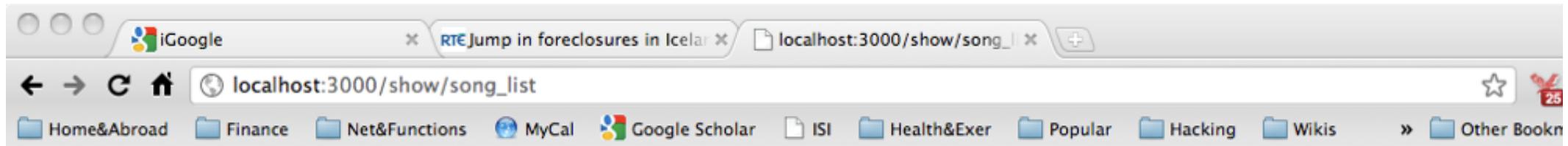
- Title Bar:** itunes - [~/Desktop/X\_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek11 (Nov 22th).13/RubyLect11.progs/itunes] - .../app/views/show/list\_song.html.erb - JetBrains Rub
- Toolbars:** Development, Run, Stop, Refresh, Open, Save, Find, Replace, Copy, Paste, Cut, Delete, Undo, Redo.
- Project Tree (1: Project):**
  - Project /Users/user/Desktop/X...
  - itunes (~/Desktop/X\_Teaching/Ruby:2011)
    - app
      - assets
      - controllers
        - add\_controller.rb
        - application\_controller.rb
        - buy\_controller.rb
        - goodbye\_controller.rb
        - show\_controller.rb
      - helpers
      - mailers
      - models
      - views
        - add
          - song.html.erb
        - buy
          - songo.html.erb
        - goodbye
        - layouts
        - show
          - actor.html.erb
          - album.html.erb
          - list\_song.html.erb
    - config
      - environments
      - initializers
      - locales
        - application.rb
        - boot.rb
        - database.yml
        - environment.rb
      - routes.rb

- Code Editor:** The current file is 'list\_song.html.erb'. The code contains an H1 header and an Ul loop.

```
<h1>Song Listing</h1>


<% @songs.each do |song| %>
  <li> <%= h song.title %> by <%= h song.artist %>, in <%= h song.in_album %> : <%= h song.id %>
    [Owned by Actor ID <%= h song.actor_id %>] </li>
<% end %>
</ul>
```
- Status Bar:** 6: TODO, 1:1, UTF-8

# Displays...



## Song Listing

- Stairway to Heaven by Led Zep, in Lex Zep IV : 1 [Owned by Actor ID 1]
- Holiday by Seex Pistols, in God Save... : 2 [Owned by Actor ID 2]
- Forest by The Cure, in Seventheen Seconds : 3 [Owned by Actor ID ]
- The end by The cure, in Three Imaginary Boys : 4 [Owned by Actor ID ]
- Holiday by Seex Pistols, in God Save... : 5 [Owned by Actor ID 2183125080]
- Holiday by Seex Pistols, in God Save... : 6 [Owned by Actor ID 2182658940]
- Holiday by Seex Pistols, in God Save... : 7 [Owned by Actor ID 2182305980]
- Holiday by Seex Pistols, in God Save... : 8 [Owned by Actor ID 2182928100]
- Holiday by Seex Pistols, in God Save... : 9 [Owned by Actor ID 2182511280]
- Holiday by Seex Pistols, in God Save... : 10 [Owned by Actor ID 2182153620]
- Holiday by Seex Pistols, in God Save... : 11 [Owned by Actor ID 2181724300]
- Holiday by Seex Pistols, in God Save... : 12 [Owned by Actor ID 2183213180]
- Stairway to Heaven by Led Zep, in Led Zep IV : 13 [Owned by Actor ID 1]
- Holiday by Sex Pistols, in God Save... : 14 [Owned by Actor ID 2]

# Oops...I forgot a controller method

if you forgot to input a method for a controller; say buy song

then it is best to re-generate it...(or use destroy then...)

**\$ rails generate controller buy song**

will prompt you for Y to overwrite your file and check to see whether you want to overwrite view too

Don't be tempted to do it yourself...by creating files..

# Oops...I forgot to put in..

```
$ rails generate controller buy song
exists app/controllers/
exists app/helpers/
exists app/views/buy
exists test/functional/
exists test/unit/helpers/
overwrite app/controllers/buy_controller.rb? (enter "h" for help) [Ynaqdh] Y
  force app/controllers/buy_controller.rb
identical test/functional/buy_controller_test.rb
identical app/helpers/buy_helper.rb
identical test/unit/helpers/buy_helper_test.rb
overwrite app/views/show/song.html.erb? (enter "h" for help) [Ynaqdh] n
  create app/views/show/song.html.erb
```

# Oops...I forgot to put in..db

you can rollback and re-migrate everything with edits

or just create a new migrate, to fix the problem

better practice, because it is more explicit for dbs

# Commands in migrations

`create_table(name, options)` : creates a table

`drop_table(name)`: Drops the table called `name`.

`rename_table(old_name, new_name)`: Renames the table called `old_name` to `new_name`.

`add_column(table_name, column_name, type, options)`: Adds a new column

`rename_column(table_name, column_name, new_column_name)`: Renames a column

`change_column(table_name, column_name, type, options)`: Changes the column

`remove_column(table_name, column_name)`: Removes the column named

`add_index(table_name, column_names, options)`: Adds a new index with the name of the column. Other options include `:name` and `:unique` (e.g. `{ :name => "users_name_index", :unique => true }`).

`remove_index(table_name, index_name)`: Removes the index specified by `index_name`.

# Migrations to fix things...

Not all migrations change the schema. Some just fix the data:

```
class RemoveEmptyTags < ActiveRecord::Migration
  def self.up
    Tag.find(:all).each { |tag| tag.destroy if
      tag.pages.empty? }
  end

  def self.down
    # not much we can do to restore deleted data
    raise ActiveRecord::IrreversibleMigration,
    "Can't recover the deleted tags"
  end
end
```

Others remove columns when they migrate up instead of down:

```
class RemoveUnnecessaryItemAttributes < ActiveRecord::Migration
  def self.up
    remove_column :items, :incomplete_items_count
    remove_column :items, :completed_items_count
  end

  def self.down
    add_column :items, :incomplete_items_count
    add_column :items, :completed_items_count
  end
end
```

# Part B(vi):

## Seeding the DB

# Seed.rb

this file which sits in the db directory is a clean and quick way of seeding a db with initial records

it is clean ‘cos you just write object descriptions, like you normally do in Ruby

there is an explicit **rake** command to load it looks like this...

# Write seeds.rb like this...

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "itunes" and the current file is "seeds.rb". The left sidebar shows the project structure under the "itunes" root, including "app", "config", "db", and "public" directories. The "db" directory contains "migrate", "development.sqlite3", "schema.rb", and "seeds.rb", with "seeds.rb" currently selected. The main editor window displays the contents of the seeds.rb file:

```
# This file should contain all the record creation needed to seed the database with its default values.
# The data can then be loaded with the rake db:seed (or created alongside the db with db:setup).
#
# Examples:
#
#   cities = City.create([{ name: 'Chicago' }, { name: 'Copenhagen' }])
#   Mayor.create(name: 'Emanuel', city: cities.first)

Actor.create(:name => 'Mark', :likes => 'reggae', :status => 'Admin', :id => 1)
Actor.create(:name => 'Bloggie', :likes => 'punk', :status => 'Buyer', :id => 2)
Song.create(:title => 'Stairway to Heaven', :artist => 'Led Zep', :time => 4, :in_album => 'Led Zep IV', :actor_id => 1)
Song.create(:title => 'Holiday', :artist => 'Sex Pistols', :time => 2, :in_album => 'God Save...', :actor_id => 2)
```

# **rake db:seed**

allows you to seed the database with a few initial instances

you just write them as object instance creations

then you run **rake db:seed** to load them in

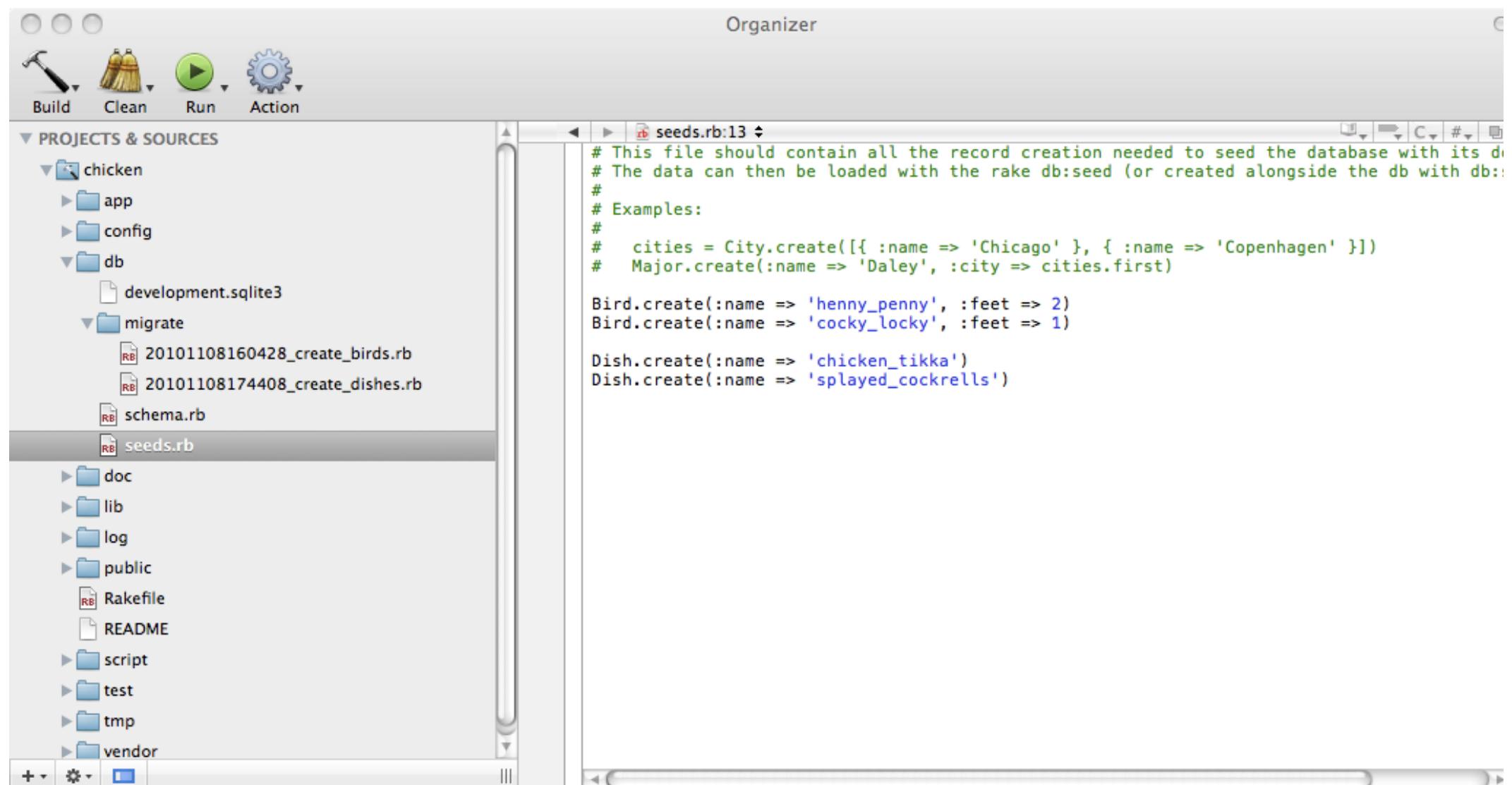
# Models recipe...

**REM:**

- 1) set up models
- 2) check and/or edit migration files
- 3) include db relations (if required) in model files (eg actor.rb)
- 4) edit your seed files (will return to this later)
- 5) do **\$ rake db:migrate**
- 6) do **\$ rake db:seed**

...then you can move to your browser and start running...

# another seeds.rb eg...



The screenshot shows an IDE interface with the title "Organizer" at the top. Below the title is a toolbar with icons for Build (hammer), Clean (broom), Run (play button), and Action (gear). The main area is divided into two panes. The left pane, titled "PROJECTS & SOURCES", displays a project structure for a "chicken" application. It includes a "db" folder containing "development.sqlite3", "migrate" subfolders with "20101108160428\_create\_birds.rb" and "20101108174408\_create\_dishes.rb", and "schema.rb". The "seeds.rb" file is currently selected and highlighted with a grey bar at the bottom of the list. The right pane contains the content of the "seeds.rb" file, which is a Ruby script for seeding a database:

```
# This file should contain all the record creation needed to seed the database with its default values.
# The data can then be loaded with the rake db:seed (or created alongside the db with db:create).
#
# Examples:
#
#   cities = City.create([{ :name => 'Chicago' }, { :name => 'Copenhagen' }])
#   Major.create(:name => 'Daley', :city => cities.first)
#
# Bird.create(:name => 'henny_penny', :feet => 2)
# Bird.create(:name => 'cocky_locky', :feet => 1)
#
# Dish.create(:name => 'chicken_tikka')
# Dish.create(:name => 'splayed_cockrells')
```

# then load them in...

```
$ rake db:seed
```

```
(in /Users/mkeane/Desktop/chicken)
```

```
rake aborted!
```

```
Could not find table 'birds'
```

```
$ rake db:migrate
```

```
(in /Users/mkeane/Desktop/chicken)
```

```
== CreateBirds: migrating ==-- create_table(:birds)
```

```
-> 0.0018s
```

```
== CreateBirds: migrated (0.0023s)
```

```
== CreateDishes: migrating
```

```
-- create_table(:dishes)
```

```
-> 0.0013s
```

```
== CreateDishes: migrated (0.0015s)
```

```
$ rake db:seed
```

```
(in /Users/mkeane/Desktop/chicken)
```

# When your schema.rb is wrong...

The screenshot shows a web browser window with two tabs: "Ryan's Scraps: What's New in" and "Action Controller: Exception". The main content area displays an error message: "ActiveRecord::UnknownAttributeError in AddController#bird". Below the title, the error message "unknown attribute: name" is shown. Further down, the RAILS\_ROOT path is listed as "/Users/mkeane/Desktop/chicken". There are three navigation links: "Application Trace", "Framework Trace", and "Full Trace". The bottom half of the screen displays a detailed stack trace:

```
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:2906:in `assign_attributes'  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:2902:in `each'  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:2902:in `assign_attributes'  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:2775:in `attributes='  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:2473:in `initialize'  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:724:in `new'  
/Library/Ruby/Gems/1.8/gems/activerecord-2.3.8/lib/active_record/base.rb:724:in `create'  
/Users/mkeane/Desktop/chicken/app/controllers/add_controller.rb:3:in `bird'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/base.rb:1331:in `send'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/base.rb:1331:in `perform_action_without_filters'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/filters.rb:617:in `call_filters'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/filters.rb:610:in `perform_action_without_benchmark'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/benchmarking.rb:68:in `perform_action_without_rescue'  
/Library/Ruby/Gems/1.8/gems/activesupport-2.3.8/lib/active_support/core_ext/benchmark.rb:17:in `ms'  
/Library/Ruby/Gems/1.8/gems/activesupport-2.3.8/lib/active_support/core_ext/benchmark.rb:17:in `ms'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/benchmarking.rb:68:in `perform_action_without_rescue'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/rescue.rb:160:in `perform_action_without_flash'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/flash.rb:151:in `perform_action'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/base.rb:532:in `send'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/base.rb:532:in `process_without_filters'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/filters.rb:606:in `process'  
/Library/Ruby/Gems/1.8/gems/actionpack-2.3.8/lib/action_controller/base.rb:391:in `process'
```

# or this in command line...

```
$ rake db:seed
```

```
(in /Users/mkeane/Desktop/chicken)
```

```
rake aborted!
```

```
unknown attribute: feet
```

```
(See full trace by running task with --trace)
```

# Bundle...

\$ bundle exec rake db:migrate

explicitly use the version of rake in the gemfile

(or update your gem file version)

# Running old apps...

```
$ cd itunes2
```

```
$ rails server
```

Could not find rake-10.0.2 in any of the sources

Run `bundle install` to install missing gems.

```
$ bundle install
```

Fetching gem metadata from <http://rubygems.org/>.....

Fetching gem metadata from <http://rubygems.org/>..

Enter password to install the bundled RubyGems to system:

Installing rake (10.0.2)

Installing i18n (0.6.1) ....

# Part C:

Using **rails console** to talk to the db

# Console interact with db

you can get a interactive console to query the database with ruby commands, uses irb

in a command line of the app directory do :

```
$ rails console
```

# Console interact with db

```
MobileMac:itunes2 user$ rails console
config.eager_load is set to nil. Please update your config/environments/*.rb files accordingly:

* development - set it to false
* test - set it to false (unless you use a tool that preloads your test environment)
* production - set it to true

Loading development environment (Rails 4.1.6)
irb(main):001:0> Song.all
Song Load (1.2ms)  SELECT "songs".* FROM "songs"
=> #<ActiveRecord::Relation [#<Song id: 1, title: "Stairway to Heaven", in_album: "Led Zep IV", time: 4, artist: "Led Zep", actor_id: 1, created_at: "2011-10-25 15:27:37", updated_at: "2011-10-25 15:27:37">, #<Song id: 2, title: "Holiday", in_album: "God Save...", time: 2, artist: "Sex Pistols", actor_id: 2, created_at: "2011-10-25 15:27:37", updated_at: "2011-10-25 15:27:37">, #<Song id: 3, title: "Gloria", in_album: "Horses", time: 3, artist: "Patti Smith", actor_id: nil, created_at: "2011-10-25 15:42:57", updated_at: "2011-10-25 15:42:57">, #<Song id: 4, title: "Gloria", in_album: "Horses", time: 3, artist: "Patti Smith", actor_id: 1, created_at: "2011-10-26 10:47:40", updated_at: "2011-10-26 10:47:40">, #<Song id: 5, title: "Gloria", in_album: "Horses", time: 3, artist: "Patti Smith", actor_id: 1, created_at: "2011-10-26 10:47:48", updated_at: "2011-10-26 10:47:48">, #<Song id: 6, title: "Bibble", in_album: "Babble", time: 4, artist: "Bobble", actor_id: nil, created_at: "2012-11-19 15:10:45", updated_at: "2012-11-19 15:10:45">]>
irb(main):002:0> □
```

Part D:

**link\_to** has a third possible arg...

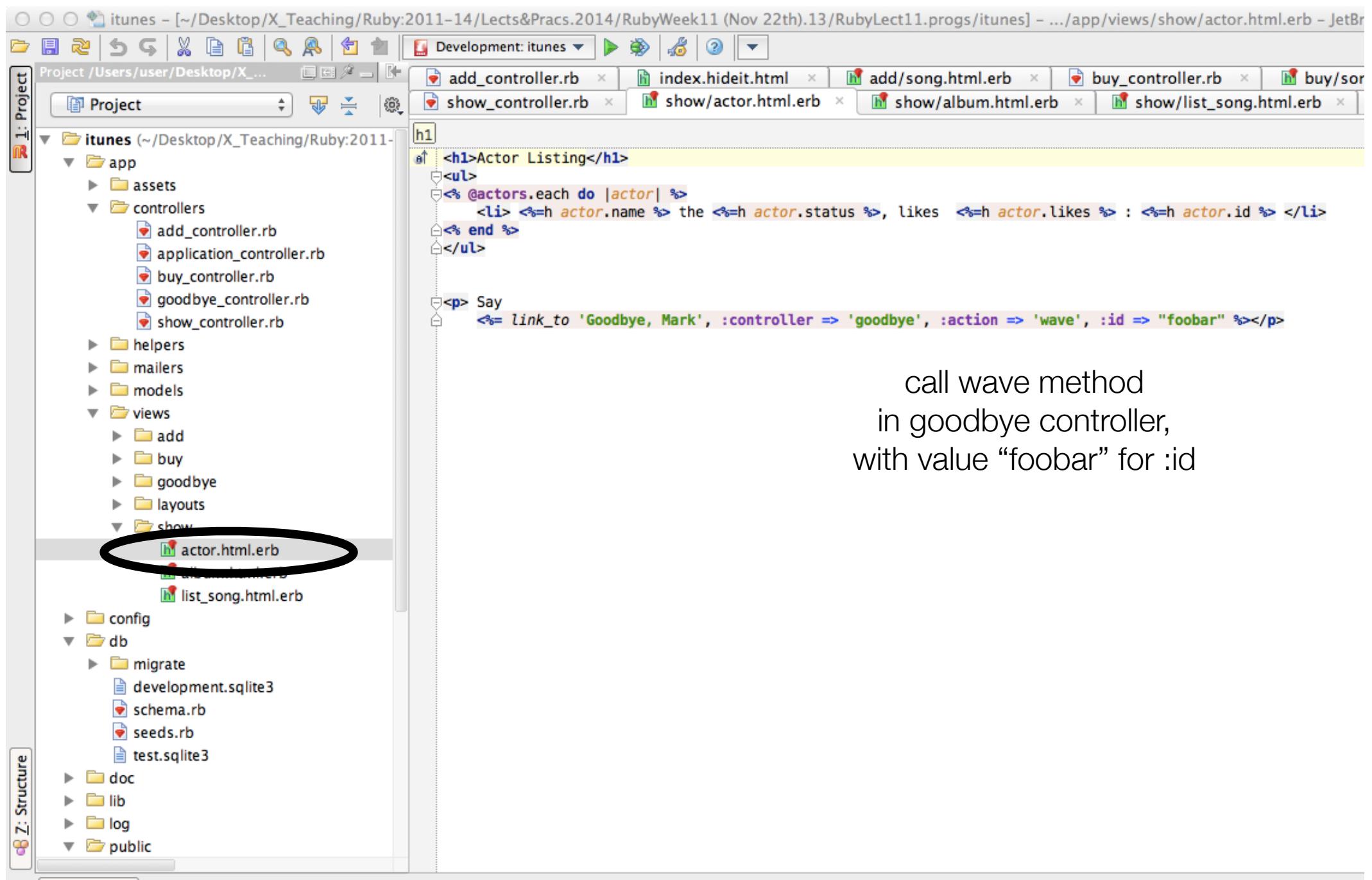
# What we will do...

we will have a **link\_to** in show/actor page, that will call goodbye/wave AND pass a value to that controller

we will then use that value in the page-view that controller renders; to just see it

could do the same thing to pass parameters to a method

# add link\_to to actor...



The screenshot shows the JetBrain RubyMine IDE interface. The title bar indicates the project is named 'itunes' and the current file is 'actor.html.erb'. The left sidebar displays the project structure under 'itunes/app/views/show'. A black oval highlights the 'actor.html.erb' file in the list. The main editor pane shows the following code:

```
h1
  <h1>Actor Listing</h1>
  <ul>
    <% @actors.each do |actor| %>
      <li> <%= h actor.name %> the <%= h actor.status %>, likes <%= h actor.likes %> : <%= h actor.id %> </li>
    <% end %>
  </ul>

  <p> Say
    <%= link_to 'Goodbye, Mark', :controller => 'goodbye', :action => 'wave', :id => "foobar" %></p>
```

On the right side of the editor, there is explanatory text:

call wave method  
in goodbye controller,  
with value “foobar” for :id

# add goodbye controller

The screenshot shows the JetBrains RubyMine IDE interface. The title bar indicates the project is named 'itunes' and the current file is 'goodbye\_controller.rb'. The left sidebar displays the project structure under 'Project /Users/user/Desktop/X\_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek11 (Nov 22th).13/RubyLect11.progs/itunes'. The 'app/controllers' folder contains several files, with 'goodbye\_controller.rb' highlighted and circled in black. The main editor window shows the code for the 'GoodbyeController' class:

```
class GoodbyeController < ApplicationController
  def wave
    @actors = Actor.all
    @actor = @actors.last
    @value = params[:id]
  end
end
```

# add goodbye view

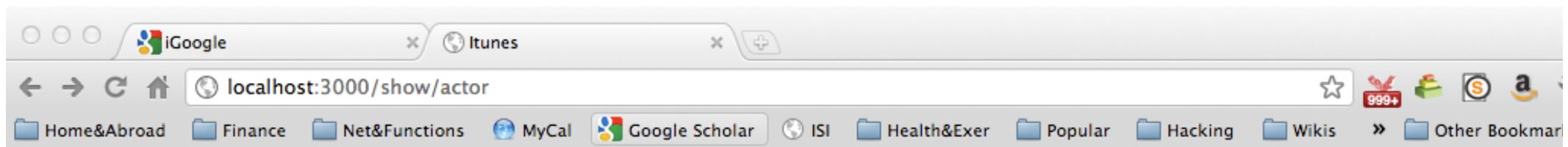
The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is 'itunes' located at '~/Desktop/X\_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek11 (Nov 22th).13/RubyLect11.progs/itunes'. The main window displays the project structure on the left and a code editor on the right.

**Project View:**

- Project: itunes (~/Desktop/X\_Teaching/Ruby:2011-14/Lect:)
- app
  - assets
  - controllers
    - add\_controller.rb
    - application\_controller.rb
    - buy\_controller.rb
    - goodbye\_controller.rb
    - show\_controller.rb
  - helpers
  - mailers
  - models
  - views
    - add
    - buy
    - goodbye
      - wave.html.erb
    - show
- config
- db
  - migrate
    - development.sqlite3
    - schema.rb
    - seeds.rb
    - test.sqlite3
- doc
- lib
- log
- public
- script
- test
- vendor
- .gitignore

```
h1
  ↳ <h1>Goodbye#actor_wave</h1>
  ↳ <p>bye bye <%= @actor.to_s %> <%h= @value %> </p>
  ↳ <p><%= @value %> </p>
```

# show/actor now looks...

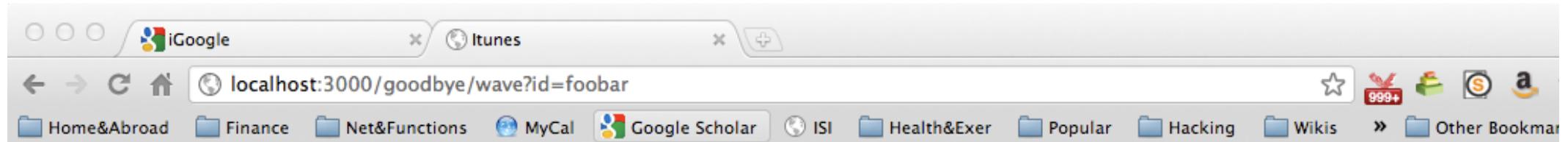


## Actor Listing

- Mark the Admin, likes reggae : 1
- Bloggie the Buyer, likes punk : 2

Say [Goodbye, Mark](#)

# when you click you get...



## Goodbye#actor\_wave

bye bye #<Actor:0x007ffa7cf49998>

foobar

## Part E:

Time has passed...how do I run my site with a new rails and gems?

# Look at old bundle...

The screenshot shows the JetBrain IDE interface with the project 'itunes' open. The left sidebar displays the project structure, and the right sidebar shows 'Data Sources'. The main window shows several files in the 'Development: itunes' tab, with the 'Gemfile' being edited. Two specific sections of the Gemfile are highlighted with black ovals:

```
source 'http://rubygems.org'

gem 'rails', '3.1.0'

# Bundle edge Rails instead:
# gem 'rails',      :git => 'git://github.com/rails/rails.git'

gem 'sqlite3'

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails', " ~> 3.1.0"
  gem 'coffee-rails', "~> 3.1.0"
  gem 'uglifier'
end

gem 'jquery-rails'

# Use unicorn as the web server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'ruby-debug19', :require => 'ruby-debug'

group :test do
  # Pretty printed test output
  gem 'turn', :require => false
end
```

# Do bundle update

```
Marks-MacBook-Pro-2:itunes2 mkeane$ bundle update
Fetching gem metadata from http://rubygems.org/.....
Fetching gem metadata from http://rubygems.org/..
Bundler could not find compatible versions for gem "railties":
  In Gemfile:
    coffee-rails (~> 3.1.0) ruby depends on
      railties (~> 3.1.0.rc1) ruby

    rails (= 3.2.8) ruby depends on
      railties (3.2.8)
```

Ukkk ! Need to update gems

# Change old versions to new...

```
source 'http://rubygems.org'

gem 'rails', '4.1.6'

# Bundler will use git instead:
# gem 'rails', :git => 'git://github.com/rails/rails.git'

gem 'sqlite3'

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails', "~> 4.0.3"
  gem 'coffee-rails', "~> 4.0.1"
  gem 'uglifier'
end

gem 'jquery-rails'

# Use unicorn as the web server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'ruby-debug19', :require => 'ruby-debug'

group :test do
  # Pretty printed test output
  gem 'turn', :require => false
end
```

# Change old *versions* to new...

```
source 'http://rubygems.org'

gem 'rails', '3.2.8'

# Bundle edge Rails instead:
# gem 'rails', :git => 'git://github.com/rails/rails.git'

gem 'sqlite3'

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails',    '~> 3.2.3'
  gem 'coffee-rails',  '~> 3.2.1'
  gem 'uglifier'
end

gem 'jquery-rails'

# Use unicorn as the web server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'ruby-debug19', :require => 'ruby-debug'
```

```
Marks-MacBook-Pro-2:itunes2 mkeane$ bundle update
Fetching gem metadata from http://rubygems.org/.....
Fetching gem metadata from http://rubygems.org/..
Enter your password to install the bundled RubyGems to your system:
Using rake (10.0.2)
Using i18n (0.6.1)
Using multi_json (1.3.7)
Using activesupport (3.2.8)
Using builder (3.0.4)
Using activemodel (3.2.8)
Using erubis (2.7.0)
Using journey (1.0.4)
Using rack (1.4.1)
Using rack-cache (1.2)
Using rack-test (0.6.2)
Using hike (1.2.1)
Using tilt (1.3.3)
Using sprockets (2.1.3)
Using actionpack (3.2.8)
Using mime-types (1.19)
Using polyglot (0.3.3)
Using treetop (1.4.12)
Using mail (2.4.4)
Using actionmailer (3.2.8)
Using arel (3.0.2)
Using tzinfo (0.3.35)
Using activerecord (3.2.8)
Using activeresource (3.2.8)
Using ansi (1.4.3)
Using bundler (1.2.1)
Using coffee-script-source (1.4.0)
Using execjs (1.4.0)
Using coffee-script (2.2.)
Using rack-ssl (1.3.2)
Using json (1.7.5)
Using rdoc (3.12)
Using thor (0.16.0)
Using railties (3.2.8)
Using coffee-rails (3.2.2)
Using jquery-rails (2.1.3)
Using rails (3.2.8)
Using sass (3.2.3)
Using sass-rails (3.2.5)
Using sqlite3 (1.3.6)
Using turn (0.9.6)
Using uglifier (1.3.0)
Your bundle is updated! Use `bundle show [gemname]` to see where a bundled gem is installed.
```

Marks-MacBook-Pro-2:itunes2 mkeane\$

Now, our rails site is running  
with new gems...but note  
you need to still test it, to  
see is all ok (see itunes2)

# Drinks ?....

