

Ruby Explorations XIII

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
```

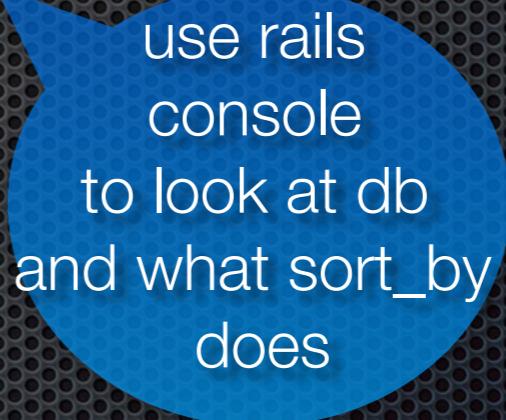
```
$ .....edit various files with stuff...
```

```
$ rake db:create
```

```
$ rake db:migrate
```

```
$ rake db:seed
```

```
$ rails server
```



use rails
console
to look at db
and what sort_by
does

clovvis controller show/list

The screenshot shows the JetBrains RubyMine IDE interface. The title bar indicates the project is named "clovis". The left sidebar displays the project structure under "clovis", including "app", "db", "lib", "log", "public", "test", "vendor", and various configuration files. The main editor area shows the file "show_controller.rb" with the following code:

```
class ShowController < ApplicationController
  def list
    @criterion = params[:id] || "id"
    sort_method = case @criterion
      when "name",
        "artist",
        "album" then lambda{|song| song.send(@criterion).downcase}
      when "id" then lambda{|song| song.id}
    end
    @out = Song.all.sort_by &sort_method
  end
end
```

Other tabs visible in the editor include "production.rb", "20141014191614_create_songs.rb", "show_helper.rb", "show/list.html.erb", and "schema.rb". The bottom status bar shows "6 TODO".

clovis model Song

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The project navigation bar at the top indicates the current workspace is 'clovis'. The left sidebar displays the project structure under 'clovis', including 'app', 'db', 'lib', 'log', 'public', 'test', and 'vendor' directories. The 'db' directory is expanded, showing 'development.sqlite3', 'schema.rb', 'seeds.rb', and 'test.sqlite3'. The main editor area shows the 'schema.rb' file, which contains 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: 20141014191614) do

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

A blue callout bubble with a white border and a black arrow points from the bottom right towards the word ':id' in the 'force: true do |t|' block of the code. The text inside the bubble reads ':id created automatically'.

clovise seeds

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "clovis" and the current file is "seeds.rb". The left sidebar displays the project structure under "clovis (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis)". The "app" directory contains "assets", "controllers", "helpers", "mailers", "models", and "views" sub-directories. The "views" directory has "layouts" and "show" sub-directories, with "list.html.erb" being the currently selected file. The "db" directory contains "migrate", "development.sqlite3", "schema.rb", "seeds.rb", and "test.sqlite3". The "lib", "log", "public", "test", "vendor", ".gitignore", "config.ru", "Gemfile", and "Gemfile.lock" files are also listed. The main editor window shows the "seeds.rb" file content:

```
# 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)

Song.create(:name => 'The End', :artist => 'Doors', :album => 'Greatest Hits')
Song.create(:name => 'Red Hot', :artist => 'RHCP', :album => 'What Hits')
Song.create(:name => 'Solid Air', :artist => 'John Martyn', :album => 'Solid Air')
Song.create(:name => 'May You Never', :artist => 'John Martyn', :album => 'Solid Air')
Song.create(:name => 'May You Never', :artist => 'Eric Clapton', :album => 'Clap Hits')
```

link_to_criterion

- if there is a method you want 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

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the current project is 'clovis' located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis'. The main window displays the 'show_helper.rb' file under the 'helpers' directory of the 'app' folder. The code defines a module named 'ShowHelper' with a single method, 'link_to_criteria', which generates a link to a specific criterion's show page.

```
module ShowHelper
  def link_to_criteria(criterion)
    link_to(criterion.capitalize,
            :controller => "show",
            :action   => "list",
            :id      => criterion)
  end
end
```

The project structure on the left shows the following directory tree:

- clovis (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis)
 - app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - show_controller.rb
 - helpers
 - application_helper.rb
 - show_helper.rb
 - mailers
 - models
 - views
 - layouts
 - show
 - list.html.erb
 - bin
 - config
 - db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
 - test.sqlite3
 - lib
 - log
 - public
 - test
 - vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock

link to screen for view

A screenshot of a web browser window. The title bar shows several open tabs: "Clovis", "Google", "Word distribution on Mary...", "At the National Convention...", and "Gantt charts for B...". The address bar displays the URL "localhost:3000/show/list?id=artist". Below the address bar is a navigation bar with links: "Apps", "Home", "AIB Internet Banking", "Home&Abroad", "Finance", "Net&Functions", "Google Scholar", "ISI", "Health&Exer", "Popular", and a folder icon.

Name Artist Album Id

Criterion is artist

- Doors 1
- Eric Clapton 5
- John Martyn 3
- John Martyn 4
- RHCP 2

show/list view

The screenshot shows the JetBrains RubyMine IDE interface. The title bar indicates the project is 'clovis' located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis'. The main window displays several tabs: 'production.rb', '20141014191614_create_songs.rb', 'schema.rb', 'seeds.rb', 'show_controller.rb', 'show_helper.rb', and 'show/list.html.erb'. The 'show/list.html.erb' tab is active, showing the following code:

```
<th class="show"><%= link_to_criteria("name") %></th>
<th class="show"><%= link_to_criteria("artist") %></th>
<th class="show"><%= link_to_criteria("album") %></th>
<th class="show"><%= link_to_criteria("id") %></th>



Criterion is <%= @criterion %>


<% @out.each do |item| %>
    <li> <%=h item.send(@criterion) %> <%=h item.id %> </li>
<% end %>
```

The left sidebar shows the project structure under 'clovis': app (assets, controllers, concerns, application_controller.rb, show_controller.rb), helpers (application_helper.rb, show_helper.rb), mailers, models, views (layouts, show), and db (migrate, development.sqlite3, schema.rb, seeds.rb, test.sqlite3). The bottom left corner shows 'Z: Structure'.

show helper

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the current project is 'clovis' located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis'. The main window displays the 'show_helper.rb' file under the 'helpers' directory of the 'app' folder. The code defines a module named 'ShowHelper' with a single method, 'link_to_criteria', which generates a link to a specific criterion's show page.

```
module ShowHelper
  def link_to_criteria(criterion)
    link_to(criterion.capitalize,
            :controller => "show",
            :action   => "list",
            :id      => criterion)
  end
end
```

The project structure on the left shows the following directory tree:

- clovis (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis)
 - app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - show_controller.rb
 - helpers
 - application_helper.rb
 - show_helper.rb
 - mailers
 - models
 - views
 - layouts
 - show
 - list.html.erb
 - bin
 - config
 - db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
 - test.sqlite3
 - lib
 - log
 - public
 - test
 - vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock

clov is controller show/list

The screenshot shows the IntelliJ IDEA interface with the 'clovis' project open. The left sidebar displays the project structure, including the 'app/controllers/show_controller.rb' file which is currently selected. The main editor pane shows the code for the 'ShowController' class:

```
class ShowController < ApplicationController
  def list
    @criterion = params[:id] || "id"
    sort_method = case @criterion
      when "name",
        "artist",
        "album" then lambda { |song| song.send(@criterion).downcase}
      when "id" then lambda { |song| song.id}
    end
    @out = Song.all.sort_by &sort_method
  end
end
```

Annotations with callouts explain specific parts of the code:

- A callout points to the line `@criterion = params[:id] || "id"` with the text "if :id is nil use "id"".
- A callout points to the line `when "id" then lambda { |song| song.id}` with the text "using lambda".
- A callout points to the line `song.send(@criterion)` with the text "send turns string into method".
- A large callout covers the entire `case` block from `when "name"` to `when "id"` with the text "case tests @criterion against string after when".

show/list view

The screenshot shows the JetBrains RubyMine IDE interface. The top navigation bar displays the project name "clovis" and the file path ".../app/views/show/list.html.erb". The main window has several tabs open: "production.rb", "20141014191614_create_songs.rb", "show_controller.rb", "show_helper.rb", "show/list.html.erb", "schema.rb", and "seeds.rb". On the left, the "Project" tool window shows the directory structure of the "clovis" project, including "app", "db", "lib", "log", "public", "test", "vendor", and various files like "application_controller.rb", "show_controller.rb", "application_helper.rb", and "show_helper.rb". The "show/list.html.erb" file is currently selected and is displayed in the central editor area. The code in the editor is:

```
<th class="show"><%= link_to_criteria("name") %></th>
<th class="show"><%= link_to_criteria("artist") %></th>
<th class="show"><%= link_to_criteria("album") %></th>
<th class="show"><%= link_to_criteria("id") %></th>



Criterion is <%= @criterion %>


<% @out.each do |item| %>
    <li> <%=h item.send(@criterion) %> <%=h item.id %> </li>
<% end %>
```

A blue callout bubble points to the line of code `<%=h item.send(@criterion) %>` with the text: "send invokes method on item".

link to screen for view

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/show/list?id=artist`. The page content displays a table with three columns: Name, Artist, Album, and Id. Below the table, a message states "Criterion is artist" followed by a bulleted list of artists and their counts.

Name	Artist	Album	Id
Doors	1		
Eric Clapton	5		
John Martyn	3		
John Martyn	4		
RHCP	2		

Criterion is artist

- Doors 1
- Eric Clapton 5
- John Martyn 3
- John Martyn 4
- RHCP 2

[Name](#) [Artist](#) [Album](#) [Id](#)

Criterion is artist

- Doors 1
- Eric Clapton 5
- John Martyn 3
- John Martyn 4
- RHCP 2

link to screen for view

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:3000/show/list?id=name
- Open Tabs:** Clovis, Google, Word distribution on Mary..., At the National Convention, Gantt charts for Basecamp.
- Toolbar:** Apps, Home, AIB Internet Banking, Home&Abroad, Finance, Net&Functions, Google Scholar, ISI, Health&Exer, Popular, Hacking, Wiki
- Content Area:** A table with columns: Name, Artist, Album, Id. The table contains five rows of data.
- Text:** Criterion is name
- List:** A bulleted list of five items: May You Never 5, May You Never 4, Red Hot 2, Solid Air 3, The End 1.

Name	Artist	Album	Id
May You Never	5		
May You Never	4		
Red Hot	2		
Solid Air	3		
The End	1		

- May You Never 5
- May You Never 4
- Red Hot 2
- Solid Air 3
- The End 1

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 aid maintenance

Test controller has tester

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named "clovis" and the current file is "test_controller.rb". The left sidebar displays the project structure under "clovis2 [clovis]". The "app/controllers" directory contains "concerns", "application_controller.rb", "show_controller.rb", and "test_controller.rb", which is currently selected. The "views/show/test" directory contains "tester.html.erb". The main editor window shows the code for "test_controller.rb". The code defines a class `TestController < ApplicationController` with a method `tester`. This method takes a parameter `@criterion` (defaulting to `params[:id] || "id"`) and sorts a collection of `Song` objects. The sorting logic uses case statements for "name", "artist", and "album", and a lambda for "id". The result is assigned to `@out` via `Song.all.sort_by &sort_method`. The code ends with an `end` keyword. The status bar at the bottom shows "6: TODO".

```
class TestController < ApplicationController
  def tester
    @criterion = params[:id] || "id"
    sort_method = case @criterion
      when "name",
        "artist",
        "album" then lambda { |song| song.send(@criterion).downcase}
      when "id" then lambda { |song| song.id}
    end
    @out = Song.all.sort_by &sort_method
  end
end
```

(see clovis2)

show helper in clovis2

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is 'clovis' located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis2'. The main window displays the file 'show_helper.rb' which contains the following code:

```
module ShowHelper
  def link_to_criteria(criterion)
    link_to(criterion.capitalize,
           :controller => "show",
           :action   => "list",
           :id      => criterion)
  end
end
```

The project structure on the left shows the directory 'clovis2 [clovis]' containing 'app', 'config', 'db', and other standard Rails directories. The 'app/helpers/show_helper.rb' file is selected in the project tree.

Three callout bubbles highlight specific aspects of the code:

- A blue callout bubble points to the 'link_to_criteria' method definition with the text: "has **link_to_criteria** helper".
- A blue callout bubble points to the 'list.html.erb' file in the 'views/show' directory with the text: "has no **link_to_criteria** method, it is empty".
- A blue callout bubble points to the 'tester.html.erb' file in the 'views/test' directory with the text: "has new screen that prints stuff".

In the bottom right corner, the text "(see clovis2)" is displayed.

test helper in clovis2

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The left pane displays the project structure for 'clovis2' under the 'Development: clovis' tab. The right pane shows several open files in tabs, with 'test_helper.rb' currently selected.

```
module TestHelper
  # Mammy, I feel a bit empty....
end
```

Annotations with blue callouts point to specific parts of the code:

- A callout points to the 'test_helper.rb' file in the project tree, stating: "has link_to_criteria helper".
- A callout points to the 'list.html.erb' file in the 'views/show' directory, stating: "has no link_to_criteria method, it is empty".
- A callout points to the 'tester.html.erb' file in the 'test' directory, stating: "has new screen that prints stuff".

In the bottom right corner, the text "(see clovis2)" is displayed.

test helper in clovis2

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is 'clovis' located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/clovis2'. The main window displays the project structure on the left and the code editor on the right.

Project Structure:

- clovis2 [clovis] (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014)
 - app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - show_controller.rb
 - test_controller.rb
 - helpers
 - application_helper.rb
 - show_helper.rb
 - test_helper.rb
 - mailers
 - models
 - views
 - layouts
 - show
 - list.html.erb
 - test
 - tester.html.erb
- bin
- config
 - environments
 - initializers
 - locales
 - application.rb
 - boot.rb
 - database.yml
 - environment.rb
 - routes.rb
 - secrets.yml
- db
 - migrate
 - 20141014191614_create_songs.rb
 - development.sqlite3
 - schema.rb
 - seeds.rb
 - test.sqlite3

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

has new screen that prints stuff

(see clovis2)

A screenshot of a web browser window. The title bar shows three tabs: "iGoogle", "Clovis", and a third tab with a plus sign. The address bar displays "localhost:3000/test/tester?id=name". Below the address bar is a horizontal menu bar with various links: "Home&Abroad", "Finance", "Net&Functions", "MyCal", "Google Scholar", "ISI", "Health&Exer", "Popular", "Hacking", "Wikis", and "Other Bookmarks". A red badge with "999+" is visible next to the "Popular" link. The main content area contains a table with columns "Name", "Artist", "Album", and "Id". The "Name" column has a link labeled "Name Artist Album Id". Below the table, a message says "This is the Test/tester one: Criterion is name". A bulleted list follows: "• May You Never 5", "• May You Never 4", "• Red Hot 2", "• Solid Air 3", and "• The End 1".

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...

- **R**epresentational **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 CORBA, 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, ...)`.

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
```

whooooo....Nellie !!!

The screenshot shows an IDE interface with the following details:

- Title Bar:** "blog - [~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog] - .../app/controllers/posts_controller.rb".
- Toolbar:** Standard file operations (New, Open, Save, Find, etc.) and development tools.
- Project Structure:** On the left, the project tree shows the directory structure:
 - Project:** Project
 - blog (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog)**
 - app**
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - posts_controller.rb**
 - helpers
 - mailers
 - models
 - views
 - layouts
 - posts
 - _form.html.erb
 - edit.html.erb
 - index.html.erb
 - index.json.jbuilder
 - new.html.erb
 - show.html.erb
 - show.json.jbuilder
 - bin
 - config
 - db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
 - lib
 - log
 - public
 - test
 - vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock
 - Rakefile
 - README.md
- Code Editor:** The main window displays the `posts_controller.rb` file content. The code defines a `PostsController` class that inherits from `ApplicationController`. It includes a `before_action` filter for `:set_post` on the `:show, :edit, :update, :destroy` actions. The controller has methods for index, show, new, edit, create, and update. The `create` and `update` methods handle both HTML and JSON formats, saving the post and rendering the response accordingly. Error handling is provided for failed saves.
- Status Bar:** Shows "6: TODO" at the bottom left.

blog - [~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog] - .../app/views/posts/index.html.erb - JetBrains

Project /Users/user/Desktop/X_Teaching...

posts_controller.rb × posts/index.html.erb × posts/edit.html.erb × Gemfile × posts/_form.html.erb

Project

blog (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog)

- app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - posts_controller.rb
 - helpers
 - mailers
 - models
 - views
 - layouts
 - posts
 - _form.html.erb
 - edit.html.erb
 - index.html.erb
 - index.json.jbuilder
 - new.html.erb
 - show.html.erb
 - show.json.jbuilder
- bin
- config
- db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
- lib
- log
- public
- test
- vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock
 - Rakefile
 - README.rdoc

Structure

posts_controller.rb

index.html.erb

index.html.erb

```
<h1>Listing posts</h1>



| Name             | Title             | Content             |                      |                      |                                                                           |
|------------------|-------------------|---------------------|----------------------|----------------------|---------------------------------------------------------------------------|
| <%= post.name %> | <%= post.title %> | <%= post.content %> | <a href="#">Show</a> | <a href="#">Edit</a> | <a href="#" data-method="delete" data-confirm="Are you sure?">Destroy</a> |

<= link_to 'New Post', new_post_path %>
```

6: TODO

1: Project

Project /Users/user/Desktop/X_Teaching... 1

posts_controller.rb posts/index.html.erb posts/edit.html.erb Gemfile posts/_form.html.erb

Project

blog (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog)

- app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - posts_controller.rb
 - helpers
 - mailers
 - models
- views
 - layouts
 - posts
 - _form.html.erb
 - edit.html.erb
 - index.html.erb
 - index.json.jbuilder
 - new.html.erb
 - show.html.erb
 - show.json.jbuilder
- bin
- config
- db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
- lib
- log
- public
- test
- vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock
 - Rakefile
 - README.md

Z: Structure

6: TODO

```

# 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: 20141014203110) do

  create_table "posts", force: true do |t|
    t.string   "name"
    t.string   "title"
    t.string   "content"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end

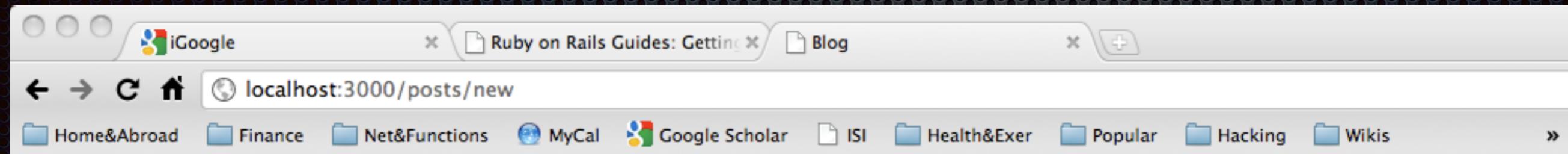
```

posts/

A screenshot of a web browser window titled "localhost:3000/posts/". The browser has three tabs open: "iGoogle", "Ruby on Rails Guides: Getting Started", and "Blog". The "Blog" tab is active. The address bar shows the URL "localhost:3000/posts/". Below the address bar is a toolbar with links to various categories: Home&Abroad, Finance, Net&Functions, MyCal, Google Scholar, ISI, Health&Exer, Popular, Hacking, and Wikis. The main content area is titled "Listing posts" and contains a table header with columns "Name", "Title", and "Content". There is one row of data in the table, which is a link labeled "New Post".

Name	Title	Content
		New Post

posts/new



New post

Name

Title

Content

[Create Post](#)

[Back](#)

posts/new



New post

Name

mark keane

Title

plabber

Content

i found this plabber in a blabbe

[Create Post](#)

[Back](#)

posts/1

A screenshot of a web browser window. The title bar shows "iGoogle" and "Blog". The address bar displays "localhost:3000/posts/1". Below the address bar is a horizontal menu bar with various links: "Home&Abroad", "Finance", "Net&Functions", "MyCal", "Google Scholar", "ISI", "Health&Exer", "Popular", "Hacking", "Wikis", and "Other Bookn". On the right side of the menu bar, there are several icons, including a star, a red box with "999+", a green folder, a blue square with a white letter "S", and a yellow "a".

Post was successfully updated.

Name: mark keane

Title: plabber

Content: I found this plabber in a blabber

[Edit](#) | [Back](#)

posts/1/edit

iGoogle Blog

localhost:3000/posts/1/edit

Home&Abroad Finance Net&Functions MyCal Google Scholar ISI Health&Exer Popular Hacking Wikis Other Bookmarks

999+

Editing post

Name

mark keane

Title

plabber

Content

I found this plabber in a blabbe

Update Post

Show | Back

Editing post

Name

mark keane

Title

plabber

Content

I found this plabber in a blabbe

[Update Post](#)

[Show](#) | [Back](#)

...updated...

A screenshot of a web browser window. The title bar shows 'iGoogle' and 'Blog'. The address bar displays 'localhost:3000/posts/1'. The page content area shows a green success message: 'Post was successfully updated.' Below it, three fields are listed: 'Name: mark keane', 'Title: plabber', and 'Content: I found this plabber in a blabber'. At the bottom left, there are 'Edit' and 'Back' links.

Post was successfully updated.

Name: mark keane

Title: plabber

Content: I found this plabber in a blabber

[Edit](#) | [Back](#)

...back sends you to...

A screenshot of a web browser window titled "Blog". The address bar shows "localhost:3000/posts". The page content is titled "Listing posts" and displays a table of posts:

Name	Title	Content	
mark keane	plabber	I found this plabber in a blabber	Show Edit Destroy
serge	decoy ducks	boy are they dead	Show Edit Destroy
blibbidy	fecund	its full of it....	Show Edit Destroy

[New Post](#)

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

...and destroy...

A screenshot of a web browser window titled "Blog". The address bar shows "localhost:3000/posts". The page content is titled "Listing posts" and displays a table with two rows of data. The columns are labeled "Name", "Title", and "Content". The first row contains "mark keane plabber" and "I found this plabber in a blabber", with links "Show", "Edit", and "Destroy". The second row contains "serge decoy ducks" and "boy are they dead", also with "Show", "Edit", and "Destroy" links. At the bottom left is a link "New Post".

Name	Title	Content
mark keane	plabber	I found this plabber in a blabber
serge	decoy ducks	boy are they dead

[Show](#) [Edit](#) [Destroy](#)

[Show](#) [Edit](#) [Destroy](#)

[New Post](#)

localhost:3000/posts

defaults to...

localhost:3000/posts/index

rb blog - [/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog] - .../app/controllers/posts_controller.rb

1: Project

Project /Users/user/Desktop/X_Teaching...

posts_controller.rb x Gemfile x posts/_form.html.erb x schema.rb x

Development: blog

posts_controller.rb

```
class PostsController < ApplicationController
  before_action :set_post, only: [:show, :edit, :update, :destroy]

  # GET /posts
  # GET /posts.json
  def index
    @posts = Post.all
  end

  # GET /posts/1
  # GET /posts/1.json
  def show
  end

  # GET /posts/new
  def new
    @post = Post.new
  end

  # GET /posts/1/edit
  def edit
  end

  # POST /posts
  # POST /posts.json
  def create
    @post = Post.new(post_params)

    respond_to do |format|
      if @post.save
        format.html { redirect_to @post, notice: 'Post was successfully created.' }
        format.json { render :show, status: :created, location: @post }
      else
        format.html { render :new }
        format.json { render json: @post.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /posts/1
  # PATCH/PUT /posts/1.json
  def update
    respond_to do |format|
      if @post.update(post_params)
        format.html { redirect_to @post, notice: 'Post was successfully updated.' }
        format.json { render :show, status: :ok, location: @post }
      else
        format.html { render :edit }
        format.json { render json: @post.errors, status: :unprocessable_entity }
      end
    end
  end
end
```

2: Structure

Project /Users/user/Desktop/X_Teaching...

blog (~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog)

- app
 - assets
 - controllers
 - concerns
 - application_controller.rb
 - posts_controller.rb
 - helpers
 - mailers
 - models
- views
 - layouts
 - posts
 - _form.html.erb
 - edit.html.erb
 - index.html.erb
 - index.json.jbuilder
 - new.html.erb
 - show.html.erb
 - show.json.jbuilder
- bin
- config
- db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
- lib
- log
- public
- test
- vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock
 - Rakefile
 - README.rdoc

6: TODO

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...

The screenshot shows a web browser window with the following details:

- Address Bar:** www.hostingrails.com/How-to-deploy-a-Rails-application-on-HostingRails-com
- Page Header:** The page is titled "How to deploy a Rails application on HostingRails.com".
- Navigation:** A horizontal menu bar includes links for "Hosting", "Signup", "FAQ", "Tutorials", "Forums" (which is currently selected), and "Support". Below this, a sub-menu for "Forums" lists categories: "Deployment", "Rails Coding", "Everything Else", and "Wiki".
- Content Area:** The main content area is titled "Support and Community Forums". It features a large orange header "How to deploy a Rails application on HostingRails.com". Below it, a text block states: "This tutorial assumes you are a beginner but have a working Rails app on your local machine ready for deployment onto your HostingRails.com account. It also assuming you have no interest in deploying via SVN/Git & Capistrano at this time:". A "Contents:" section follows, listing steps: "Receive your welcome email and take note of important info", "Log into your cPanel and set up your mySQL database(s)", "Create online versions of your environment.rb and database.yml files", "Make sure your .htaccess file sends requests to dispatch.fcgi", "Login to the command line and generate your Rails app", "Upload only key folders and files to your online app", "Make your app's public folder the public_html", "And that's it, your app is live! Let's take a sneak peak", "Note: Making changes to your app when its in production mode", and "Note: Answers to common problems".

<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", ["embed-  
"array"], 2, 3]
test_2 = ["level1", ["level2", ["leve-  
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)
```

```
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

The screenshot shows the JetBrains RubyMine IDE interface. The top bar displays the project name "blog" and its path: "~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog". Below the top bar is the toolbar with various icons for file operations like Open, Save, Find, and Run.

The left sidebar is the Project View, showing the directory structure of the "blog" project. It includes folders for app, bin, config, db, lib, log, public, test, vendor, and external libraries. Inside the config folder, there are environments (development.rb, production.rb, test.rb), initializers, locales, application.rb, boot.rb, database.yml, environment.rb, routes.rb, and secrets.yml. The db folder contains migrate, development.sqlite3, schema.rb, and seeds.rb. The vendor folder contains .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc.

The main editor area shows the "database.yml" file. The code defines database configurations for different environments:

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

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

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
<<: *default
  database: db/test.sqlite3

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

The "database.yml" file is currently selected in the Project View. The bottom status bar shows "6: TODO" and the time "24:53".

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 fixtures that can be loaded into the database with:

```
$ rake db:fixtures:load
```

fixtures

The screenshot shows the JetBrains RubyMine 3.2.4 IDE interface. The title bar indicates the project is named 'blog' and is located at '~/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek12 (Nov 29th).13/RubyWeek12.progs/blog'. The main window displays the 'posts.yml' fixture file under the 'test/fixtures' directory. The code in the file defines two fixtures, 'one' and 'two', each with attributes: name, title, and content, all of type MyString.

```
# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/FixtureSet.html
one:
  name: MyString
  title: MyString
  content: MyString
two:
  name: MyString
  title: MyString
  content: MyString
```

The left sidebar shows the project structure:

- blog (~/Desktop/X_Teaching/)
 - app
 - bin
 - config
 - environments
 - initializers
 - locales
 - application.rb
 - boot.rb
 - database.yml
 - environment.rb
 - routes.rb
 - secrets.yml
- db
 - migrate
 - development.sqlite3
 - schema.rb
 - seeds.rb
 - lib
 - log
 - public
- test
 - controllers
 - fixtures
 - .keep
 - posts.yml
 - helpers
 - integration
 - mailers
 - models
 - test_helper.rb
- vendor
 - .gitignore
 - config.ru
 - Gemfile
 - Gemfile.lock
 - Rakefile
 - README.rdoc

```
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....