

Laboratorio de Computación IV

Clase 9

Andrés Fortier

¿Consultas?

- Comando: cp y mv
- Contenidos web: MVC web.

Comandos del día: history

- history == historial de comandos

```
$ history
$ history -c
$ ls
$ history
1  ls
2  history
```

- Reverse search

```
$ ls | grep "Doc"
...
[CTRL + R]
(reverse-i-search)`gre': ls | grep "Doc"
```

Comandos del día: history

- Algunas opciones (en `~.bashrc`)

```
export HISTSIZE=5000
export HISTCONTROL=erasedups
export HISTTIMEFORMAT='%F %T  '
shopt -s histappend
```

- Ahora

```
$ history
...
1997  2015-04-18 15:58:39  vim custom_ps1
1998  2015-04-18 15:58:39  vim custom_ps1
1999  2015-04-18 15:58:39  exit
2000  2015-04-18 15:58:43  history
2001  2015-04-18 16:04:42  history
...
```

Comandos del día: history

- history N

```
$ history 3
1995  2015-04-18 16:14:33  ll
1996  2015-04-18 16:14:42  ll | grep Doc
1997  2015-04-18 16:16:41  history 3
```

Comandos del día: history

- Combinado con grep

```
$ history | grep "ls -l"
 329  2015-04-18 15:58:39  ls -la
 384  2015-04-18 15:58:39  ls -la
 405  2015-04-18 15:58:39  ls -lh | grep "prueba"
 415  2015-04-18 15:58:39  ls -lh | grep cont
1042  2015-04-18 15:58:39  ls -la
1043  2015-04-18 15:58:39  ls -la > salida.txt
1166  2015-04-18 15:58:39  ls -l
1175  2015-04-18 15:58:39  ls -l subdirectorio/
1178  2015-04-18 15:58:39  ls -l subdirectorio/
1201  2015-04-18 15:58:39  ls -l
1204  2015-04-18 15:58:39  ls -l sub3/
2003  2015-04-18 16:06:20  history | grep "ls -l"
```

Tarea para el hogar

- Crear una aplicación Rails de ejemplo

```
$ mkdir ejemplorails  
  
$ cd ejemplorails/  
  
$ rvm use --create ruby-2.0.0-p598@ejemplorails  
...  
  
$ sqlite3 --version  
3.8.2 2013-12-06 14:53:30  
27392118af4c38c5203a04b8013e1afdb1cebd0d  
  
$ gem install rails -v 4.1.8  
  
$ rails new . -T
```

Tarea para el hogar

- Crear una aplicación Rails de ejemplo

```
$ bin/rails server
=> Booting WEBrick
=> Rails 4.1.8 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
```


RoR

- Setear ruby version y gemset

```
$ echo "ruby-2.0.0-p598" > .ruby-version
$ echo "ejemplorails" > .ruby-gemset
$ cd ..
$ rvm use ruby-2.0.0-p598@global
$ rvm current
ruby-2.0.0-p598@global
$ cd ejemplorails/
$ rvm current
ruby-2.0.0-p598@ejemplorails
```

- *Nota: rvm-prompt*
 - <https://rvm.io/workflow/prompt>

RoR

- Crear un repo git y hacer el primer commit

```
$ git init
$ git add .
$ git ci -m 'First commit'
[master (root-commit) 51ebb01] First commit
53 files changed, 902 insertions(+)
...
```

- Crear/editar los tres archivos de ejemplo de la clase pasada y verificar que funcione.

```
config/routes.rb
```

```
app/controllers/example_controller.rb
```

```
app/views/example/hello.erb
```

RoR

```
$ cat Gemfile
source 'https://rubygems.org'

# Bundle edge Rails...
gem 'rails', '4.1.8'
# Use sqlite3 as the database for Active Record
gem 'sqlite3'
# Use SCSS for stylesheets
gem 'sass-rails', '~> 4.0.3'
# 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...
# gem 'therubyracer', platforms: :ruby
```

RoR

```
# Use jquery as the JavaScript library
gem 'jquery-rails'

# Turbolinks makes following links in your web application
faster. Read more: https://github.com/rails/turbolinks
# gem 'turbolinks'

# Build JSON APIs with ease. Read more:
https://github.com/rails/jbuilder
# gem 'jbuilder', '~> 2.0'

# bundle exec rake doc:rails generates the API under
doc/api.
gem 'sdoc', '~> 0.4.0',                group: :doc
```

RoR

```
# Spring speeds up development by keeping your application
running in the background. Read more:
https://github.com/rails/spring
gem 'spring',          group: :development

# Use ActiveSupport has_secure_password
# gem 'bcrypt', '~> 3.1.7'

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

# Use Capistrano for deployment
# gem 'capistrano-rails', group: :development

# Use debugger
# gem 'debugger', group: [:development, :test]
```

RoR

- Default root (borrar comentarios)

```
config/routes.rb
Rails.application.routes.draw do

  root 'example#hello'

  get 'hello' => 'example#hello'

end
```

- Ir a <http://localhost:3000/>

RoR

- En otra consola

```
$ bin/rake routes
```

Prefix	Verb	URI Pattern	Controller#Action
root	GET	/	example#hello
hello	GET	/hello(.:format)	example#hello

RoR - Blog

- Sistema para ejemplificar
- Estándar
- Un blog es una colección de artículos
- Un artículo consiste de un título y un texto.

Blog - Indice de artículos

- Creemos la ruta

```
config/routes.rb
```

```
Rails.application.routes.draw do
  root 'articles#index'
  get 'articles' => 'articles#index'

  get 'hello' => 'example#hello'
end
```

- Vayan a <http://localhost:3000/>
 - Routing Error
 - uninitialized constant ArticlesController

Blog - Indice de artículos

- Creemos el controller

```
/app/controllers/articles_controller.rb  
class ArticlesController < ApplicationController  
  
  def index  
  end  
  
end
```

- Vayan a <http://localhost:3000/>
 - Template is missing
 - Missing template articles/index, application/index ...

Blog - Indice de artículos


- Creemos la vista

```
/app/views/articles/index.erb
```

```
<h1> Indice </h1>
```

- Vayan a <http://localhost:3000/>
- Commit!

RoR - Modelos y Persistencia



- Rails utiliza
 - Una BD relacional para persistir datos.
 - Un mecanismo para “mapear” objetos a la BD.
 - Un poco sobre ORMs.
- *ActiveRecord*: Implementación del patrón con el mismo nombre.

ORM



- *Object Relational Mapping.*
- Un poco sobre *Impedance mismatch*
 - Conceptual (datos vs. comportamiento).
 - Identidad de objetos.
 - Herencia.
 - ¿Lenguaje de consulta?
 - (En lenguajes tipados) Tipos vs. Constraints.
 - Normalización.

ActiveRecord



- En “*Patterns of Enterprise Application Architecture*” por Martin Fowler.

An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.

ActiveRecord

Cuenta
titular saldo
<u>all()</u> <u>where(filter)</u> ... save() delete() ... depositar(unMonto) ...

titular	saldo
"Juan Perez"	\$1.500,00
"Jose Julio"	\$300.000,00
"Anastacio Ponce"	\$12,50

```
c = Cuenta.new("Pepe")
c.save
c.depositar(100)
c.save
Cuenta.all
Cuenta.where({titular: "Pepe"})
```

ActiveRecord



- *Pros*
 - Muy rápido para lograr prototipos.
 - Conceptualmente simple.
 - Agrega validaciones en forma sencilla.
 - Muy conveniente para modelos basados en CRUDs y con poca lógica de negocios.

ActiveRecord



- *Cons*
 - De los ORMs en general: *Impedance mismatch*.
 - Viola SRP (*Single responsibility principle*).
 - Modelo de dominio atado a la BD
 - Puede ser poco eficiente por accesos a la BD.
 - Ej: loops y relaciones.
 - Complicado/lento para testear.