



# Introducción a Ruby on Rails

Mayo 2013  
Fernando Martínez

 **redradix**

# Parte 1

**¿Qué es Ruby?**

**¿Qué es Rails?**

**¿Que es un framework?**

**¿Qué es una web?**

**¿Qué es una "aplicación"?**

## Muchas piezas...

*Sistema Operativo:* OSX, GNU/Linux, Windows

*Base de Datos:* SQLite, Mysql, Postgres

*Editor/IDE:* Vim, Emacs, Textmate

*Navegador:* Safari, Chrome, Firefox

*Runtime:* Webrick, Thin, Puma, Passenger,  
Mongrel2

*Servidor web:* Apache, Nginx, IIS

*Repositorio de código:* Git, SVN, Mercurial

*Gestión de bugs:* Basecamp, GHIssues, ...

# ¿ Ruby ? ¿ Rails ?

**Prerequisites: curl, bash, build-essential...**

**Ruby: RVM <https://rvm.io/> (gemsets y .rvmrc)**

**Bundler: <http://gembundler.com/>**

**¿Tenemos todo?**

Ruby para llevar

# Hablando con Ruby

```
$ ruby -e 'print "Hola mundo\n"'
```

```
Hola mundo
```

```
$ irb
```

```
>> "Hola mundo"
```

```
=> "Hola mundo"
```

```
>> puts "Hola mundo"
```

```
Hola mundo
```

```
=> nil
```

# Calculadora gratis!

```
>> 3+2
```

```
=> 5
```

```
>> 3*2
```

```
=> 6
```

```
>> 3**2
```

```
=> 9
```

```
>> Math.sqrt(9)
```

```
=> 3.0
```

# Variables

```
>> a = 3 ** 2  
=> 9  
>> b = 4 ** 2  
=> 16  
>> Math.sqrt(a+b)  
=> 5.0
```



# Métodos

```
def nombre(parametro1, parametro2 = "por defecto")  
    # código aquí  
    # se devuelve con: return valor  
    # o el valor de la última expresión  
end
```

# se llaman así:

```
nombre('p1', 'p2')  
nombre('p1')  
nombre 'p1'  
nombre 'p1', 'p2' # cuidado!
```

# Métodos

```
>> def h
>> puts "Hello World!"
>> end
=> nil
>> h()
Hello World!
=> nil
>> h
Hello World!
=> nil
```

# Parámetros

```
>> def h(name = "World")  
>> puts "Hello #{name.capitalize}!"  
>> end  
=> nil  
>> h "chris"  
Hello Chris!  
=> nil  
>> h  
Hello World!  
=> nil
```

# Ejercicio: Nuestro primer archivo ruby

```
# greeter.rb
class Greeter
  def initialize(name = "World")
    @name = name
  end
  def say_hi
    puts "Hi #{@name}!"
  end
  def say_bye
    puts "Bye #{@name}, come back soon."
  end
end
```

## Usando código de un fichero

```
# from greeter.rb folder
$ irb
>> load 'greeter.rb'
=> true
>> g = Greeter.new
=> #<Greeter:0x23ef560 @name="World">
>> g.say_hi
Hi World!
=> nil
```

cont.

```
>> g_fer = Greeter.new('Fer')  
=> #<Greeter:0x23c7588 @name="Fer">  
>> g_fer.say_hi  
Hi Fer!  
=> nil  
>> g_fer.say_bye  
Bye Fer, come back soon.  
=> nil
```

## los secretos de los objetos...

```
>> g = Greeter.new("Pat")
>> g.@name
SyntaxError: compile error...
>> Greeter.instance_methods
=> ["method", "send", "object_id", "singleton_methods", "__send__",
    "equal?", "taint", "frozen?", ...]
>> Greeter.instance_methods(false)
=> ["say_bye", "say_hi"]
```

## objetos cont.

```
>> g.respond_to?("name")  
=> false  
>> g.respond_to?("say_hi")  
=> true  
>> g.respond_to?("to_s")  
=> true  
# to_s: where does it come from?
```



## parcheando... de nuevo

```
>> class Greeter
>> attr_accessor :name
>> end
=> nil
>> g = Greeter.new("Andy")
>> g.respond_to?("name")
=> true
>> g.respond_to?("name=")
=> true
>> g.say_hi
Hi Andy!
```

¿rayos X?

```
>> g.name = "Betty"
=> "Betty"
>> g
=> #<Greeter:0x3c9b0 @name="Betty">
>> g.name
=> "Betty"
>> g.say_hi
Hi Betty
=> nil
```

¿De qué están hechos los programas?

**Datos (variables, classes, objects... )**

**Decisiones (conditionals)**

**Repetición (loops)**

# Decisiones / Condicionales

```
if cond
  #do stuff
elsif other_cond
  #do other stuff
end
```

```
action unless condition
```

```
var = condition ? value_if_true : other_value
```

```
condition && value_if_true || other_value
```

```
# vs
```

```
condition and action_if_true or other_action
```

# Repetición / bucles

```
while conditional [do]  
  code  
end
```

```
code while condition
```

```
until conditional [do]  
  code  
end
```

```
code until conditional
```

# Más Repeticiones

```
for n in (1..10) do
  break if n > 7
  next if (n % 3 == 0)
  puts "#{n} > 2 and #{n/3} != 0"
end
```

```
(1..10).each do |n| # <- this is ruby 'fashion'
  # rewrite yourselves!!
end
```

```
# Beware: redo, retry
```

```
# read and try examples:
```

```
# http://ruby-doc.org/core-1.9.3/Enumerable.html
```

# Ejercicio

```
# Get code
$ git clone https://gist.github.com/5534016.git ej1
# Go to part I
$ cd ej1
# Open mega_greeter.rb
# Code until you get sample output :)
```

¿Por qué ir (o no) en tren?



**Rails no es magia**

**Rails no vale para todo**

**Rails no vale para todos**

**El Marketing siempre es un poco mentira**

***07/11/2005:* David Heinemeier Hansson,  
el creador del framework Ruby on Rails  
de desarrollo web muestra como crear...**

**un blog en 15 minutos.**



Ami también me j... fastidia



La fama ... CUESTA

**it is hard**

<http://goo.gl/LDJjE>

**(2012-05-08: Kakubei «Why I hate Rails»)**

**But gets through!**

Hay alternativas

**Blog -> Wordpress**

**CMS -> Drupal**

**Office + Intranet -> Sharepoint**

**Web Evento -> jekyll**

un *¿framework?*

garabateando servilletas

Why the lucky stiff (\_why)

Camping (the Microframework)

Sinatra (aka Rails lil' cousin)

Example REST MVC: <http://goo.gl/q4xfi>

Let's read/play together!



# Ejercicio

```
$ git clone https://gist.github.com/5534133.git ej2
$ cd ej2
$ rvm gemset use --create ruby-1.9.3@sinatra
$ bundle
Fetching gem metadata from https://rubygems.org/.....
$ ruby todo.rb
$ browser localhost:4567
# Jugad!
```

**Clase de san Isidro**

**por favor votad**

**esta en vuestro correo**

**<http://www.doodle.com/rcc97n2vz38exv64>**

**si alguien no puede en absoluto...**

**... la haremos el lunes**

**diapos del tema 1: [http://j.mp/rr\\_slides](http://j.mp/rr_slides)**

**Ruby on Rails is an open-source web framework that's optimized for programmer happiness and sustainable productivity.**

**It lets you write beautiful code by favoring convention over configuration.**

**Bla bla bla, veamos:**

**Ruby** on Rails is an open-source web framework  
that's optimized for **programmer happiness**  
and sustainable productivity.  
It lets you write beautiful code  
by favoring convention over configuration.

en el principio fue Ruby

**Dynamic**

**Orientado a Objetos (*realmente*)**

**Open-source (de nuevo)**

**Simple (esconde la complejidad)**

**Elegante (Japón, koans, Heroku)**

**"Feliz"**

Ruby on Rails is an **open-source** web framework  
that's optimized for programmer happiness  
and sustainable productivity.  
It lets you write beautiful code  
by favoring convention over configuration.

*una cultura **opensource***

**Rubygems (CPAN, PEAR)**

**OS & tools, not products (Rails vs  
Basecamp)**

**Git & Github!**

Ruby on Rails is an open-source **web** framework  
that's optimized for programmer happiness  
and sustainable productivity.  
It lets you write beautiful **code**  
by favoring convention over configuration.



¿Que es una «web»?

¿HTML, Browser?

¿Js?

¿API?

¿Qué es un «app»?

¿Datos?

¿Función?

¿UX?

Ruby on Rails is an open-source web **framework**  
that's optimized for programmer happiness  
and sustainable productivity.  
It lets you write beautiful code  
by favoring convention over configuration.

<https://es.wikipedia.org/wiki/Framework>

**un conjunto estandarizado de  
conceptos, prácticas y criterios  
enfocados a resolver  
un tipo particular de problema  
que se usa como referencia,  
para enfrentar y resolver  
problemas nuevos de índole similar.**

**estandarizado -> CoC**

**conceptos -> MVC, REST, UJS**

**prácticas -> TDD/BDD...**

**criterios -> REST, naming, DRY**

**tipo particular de problema:**

**aplicación web con BBDD**



# Introducción a Ruby on Rails

2 - Despegando

Mayo 2013  
Fernando Martínez

 **redradix**

# Parte 2

Despegando

**Instalar**

**Comenzar una aplicación**

**Colaborar**

**Desplegar**

# Instalación

¿Que es esto de las *gemas*?

Librerías empaquetadas (eggs, pear, cpan...)

Script de instalación **gem**

Servidores públicos <https://rubygems.org/>

y más <http://gems.github.com/list.html>



## ¿y cómo...?

**\$ gem**

RubyGems is a sophisticated package manager for Ruby.

This is a basic help message containing pointers to more information.

Usage:

gem -h/--help

gem -v/--version

gem command [arguments...] [options...]

Examples:

gem install rake

gem list --local

gem build package.gemspec

gem help install

Further help:

gem

## Ejercicio

List local gems

Search 'rainbow' gem

Search remotely

Install

```
$ ruby -rubygems -e 'require "rainbow";  
puts "red".foreground(:red)'  
red
```

pero...

**Many projects, many gems**

**Dependencies**

**Conflict**

**RVM Gemsets**

**Bundler & Gemfile**

# Ejercicio Gemsets

**Abrir la documentacion de RVM**

**Crear dos gemsets *foo* & *bar***

**Cambiar manualmente en tre uno y otro**

**Crear fichero *'.rvmrc***

**Cambiar automáticamente de gemset**

# Ejercicio de bundler

**bueno, esperad...**

Comienzo

```
$ gem install rails -v 3.2.13
```

```
...
```

```
$ rails -v
```

```
Rails 3.2.13
```

```
$ rails new prueba
```

```
...
```

```
.
```

```
.
```

```
Your bundle is complete! Use `bundle show [gemname]` to see where a  
bundled  
gem is installed.
```

**¿que tenemos? (carpetas)**

**app**

**config**

**db**

**lib**

**log**

**public**

**script**

**test**

**tmp**

**vendor**



¿que tenemos? (gemas)

**sqlite3**

**sass-rails**

**coffee-rails**

**uglifyer**

**jquery-rails**

Hay otras formas:

<http://railsapps.github.io/rails-composer/>

**Conocimiento de 1era mano vs  
experiencia colectiva**

**Aprender cómo vs desarrollar *rápido***

**Ajuste fino vs plantilla (Bootstrap)**

**Colaboración (tech)**

**escribir código (bugs)**

**guardar una versión**

**cambiar, añadir código**

**guardar una versión**

**...**

**Descubrir bug**

**Buscar cuándo se agregó...**

## SCM

**El control de versiones (SCM) es un sistema que registra los cambios en un fichero o conjunto de ficheros a lo largo del tiempo de modo que cada version específica pueda ser recuperada posteriormente.**



# Trabajando con Git

edito un fichero y lo salvo

reviso la lista de ficheros tocados *stat*

reviso los cambios *diff*

agrego el fichero a una 'reserva' *add*

confirmo un cambio con un 'mensaje' *commit*

'envio' el cambio a un servidor *push*

<http://ndpsoftware.com/git-cheatsheet.html>

## git 1 - lo básico

```
$ git config --global user.name "Your Name"  
$ git config --global user.email "your@mail.com"  
$ mkdir Gittest  
$ cd Gittest  
$ git init  
$ edit README.txt  
$ edit myscript.rb  
$ git status  
$ git add README.txt myscript.rb  
$ git status  
$ git commit -m "Now with files"
```

## git 2 - por las ramas

```
$ git branch
$ git branch experiment
$ git checkout experiment
$ git branch
$ edit new_file.txt
$ git add .
$ git commit -m "experiment exclusive file"
$ git log
$ ls
$ git checkout master
$ ls
$ git log
```



## git 2 - por las ramas

```
$ git status  
$ git diff experiment  
$ git merge experiment  
$ git log -p  
$ ls
```

## **Ejercicio: conflictos**

- 1. haz una rama, edita un fichero, haz commit**
- 2. regresa a master, edita el mismo fichero/linea**
- 3. haz commit a master y mezcla la rama**

**Hay conflictos? ¿Por qué?**

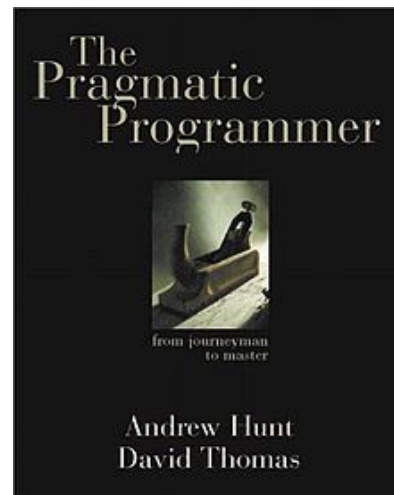
**¿Puedes resolverlos?**

**Configura Git en la aplicación Rails**

**Colaboración (org)**



<http://goo.gl/IZJIF>



<http://goo.gl/7lh1Z>

# Metodologías

[agilemanifesto.org](http://agilemanifesto.org)

[extremeprogramming.org](http://extremeprogramming.org)

[scrum.org](http://scrum.org)

[lean: goo.gl/ePctp](http://lean: goo.gl/ePctp)

**¿Conoceis otros?**

# Metodologías - Estrategia

**Elige una  
Ponte objetivos  
Aguanta!  
Mide  
Evalúa  
Adapta  
Repite**

# Metodologías - Factores

## Técnicos

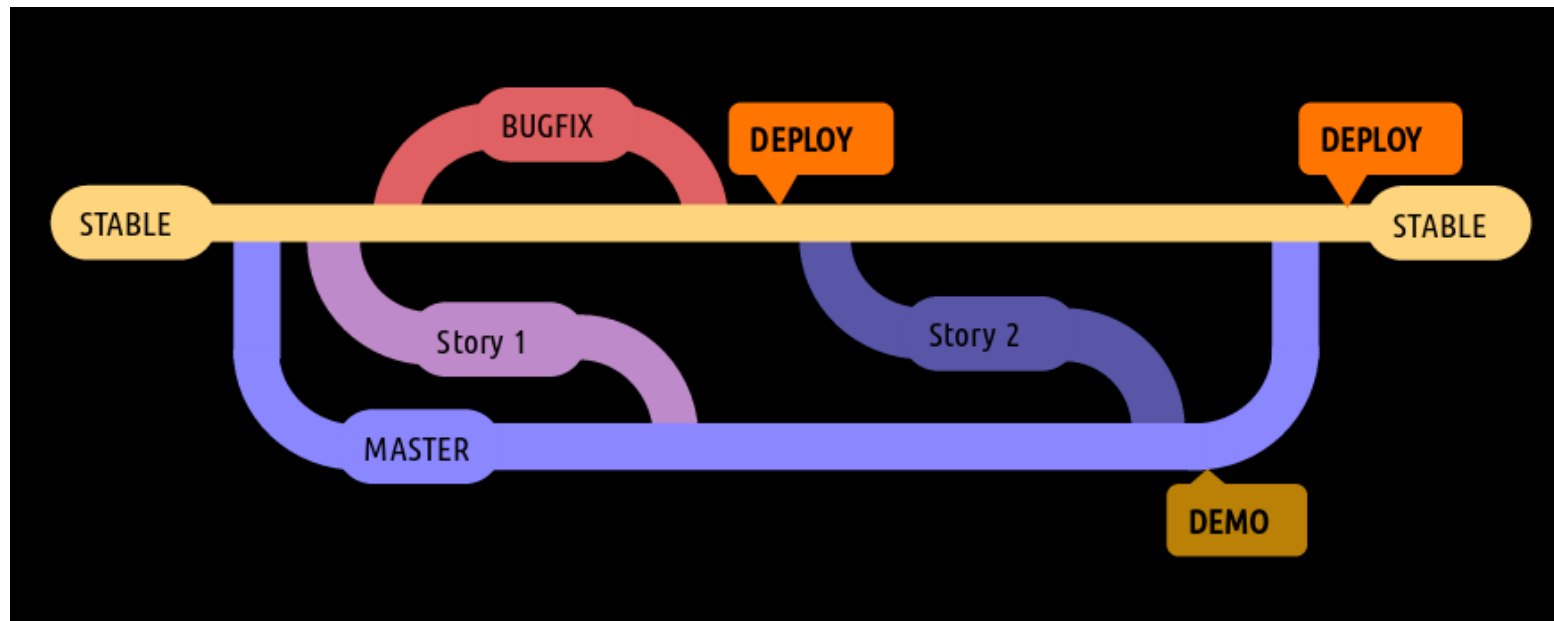
Tests  
Mocks  
Fases  
CI  
CD  
Remoto?  
Pares

## Organizativos

Tamaño equipo  
Ritmo  
Jerarquía  
Especificidad metas  
Specs vs objetivos  
Legal, PI



## Ejemplo: Git y Scrum



Despliege

del IDE a la Web

HTML+CSS → FTP upload

+html+css+js → FTP+Restart

Downtime, Releases Rollback

**herramientas** Capistrano, Sprockets (artículo)

**PaaS** EngineYard, Heroku, Nitrous.IO...

## Ejercicio

Crear una aplicación

Ponerla en Git

Registrarse en heroku <http://api.heroku.com/signup>

Bajarse la herramientas de heroku

<https://toolbelt.heroku.com/>

\$ heroku login # funciona?

Poner la 'app' online

## ayudas: Creando la aplicación

```
$ gem list # rails esta? si no, instala
$ rails new curso
...
$ cd curso
$ rvm --rvmrc --create 1.9.3@curso
$ rvm gemset list
$ gem list
$ bundle
...
```

## ayudas: configurando Git

```
$ git status
```

```
$ git add .
```

```
...
```

```
$ git status # see difference with previous status
```

```
$ git commit -m "Crear app rails"
```

```
$ # git remote add origin <url>
```

```
$ # git push origin master
```

## ayudas: "desarrollando" en Rails

```
$ cd curso # si no estas  
$ rails s  
$ open <http://localhost:3000>  
$ echo "Hola mundo" > public/index.html  
$ # reload browser
```

# ayudas: adaptar a heroku

```
# Gemfile
```

```
group :production do
```

```
  gem 'pg', '0.12.2'
```

```
end
```

```
$ bundle install --without production
```

```
# no instala nada pero cambia Gemfile.lock
```

```
$ git push heroku master # empuja el HEAD de master
```

```
$ heroku open
```

```
$ heroku rename fer_curso_rr # cada uno el suyo
```





# Introducción a Ruby on Rails

## 3 - Las Maletas

Mayo 2013  
Fernando Martínez



# Parte 3

## Las Maletas

### Bases de Datos

**Evolución de la estructura (migraciones)**

**Consultas *sin* SQL**

**Validación de datos**

**Reacciones (callbacks)**

# Modelos

¿Que son los modelos?

¿Señoras flacas + ropa cara?

**Datos**

**Relaciones**

**Comportamiento**

**¿Aplicaciones y datos?**

**¿Que son los datos?**

**¿Aplicaciones sin datos?**

**¿Datos sin aplicaciones?**

**Persistencia**

**Modelos en Rails**

**Migraciones**

**Validaciones y Callbacks**

**Asociaciones**

**Consultas**

# Migraciones

# Migraciones: ¿qué aspecto tienen?

```
class CreateCourses < ActiveRecord::Migration
  def up
    create_table :courses do |t|
      t.string :name, :null => false
      t.text :description
      t.timestamps # <- no es un tipo!
    end
  end

  def down
    drop_table :courses
  end
end
```



Migraciones: ¿qué se puede hacer?

**up**  
**down**  
**change**

## Migraciones *change*:

**add\_column**

**add\_index**

**add\_timestamps**

**create\_table**

**remove\_timestamps**

**rename\_column**

**rename\_index**

**rename\_table**

## Migraciones *sólo up/down*:

**change\_table**

**change\_column**

**drop\_table**

**remove\_column**

**remove\_index**

Migraciones *qué dicen*:

**say**

**say\_with\_time**

**suppress\_warnings{}**

Algo más? <http://api.rubyonrails.org/classes/ActiveRecord/Migration.html>

## Migraciones: ¿como se usan?

```
$ rake --tasks db
```

```
...
```

```
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).
```

```
...
```

# Migraciones: peligros

```
# db/migrate/20100513121110_add_active_to_course.rb
class AddActiveToCourse < ActiveRecord::Migration
  def change
    add_column :courses, :active, :boolean
    Course.all.each{ |c| c.update_attributes!(:active => true) }
  end
end

# app/model/course.rb
class Course < ActiveRecord::Base
  validates :active, :presence => true
end
```

# Migraciones: peligros

```
# db/migrate/20100515121110_add_loc_to_product.rb
class AddLocationToCourse < ActiveRecord::Migration
  def change
    add_column :courses, :location, :string
    Course.reset_column_information
    Course.all.{ |c| course.update_attribute(:location, 'Madrid')}
    # Course.update_all no valida y podria dejar datos no validos!
  end
end

# app/model/course.rb
class Course < ActiveRecord::Base
  validates :active, :presence => true
  validates :location, :presence => true
end
```

## Migraciones: peligros

undefined method `location' for #<Course...>

**¿Como lo evitamos?**

**No siempre igual**

**Validación condicional (se queda!)**

**Modelo mínimo dentro de la migración**

**Valores por defecto en vez de inicializar**



## Migraciones: ejercicio

**crea** Course **con** name, desc, timestamps

**Ejecuta ida y vuelta** (rake --tasks)

**añade** active (bool), ajustado a *true* por defecto

**Agrega al menos una validación**

**Migra de nuevo**

# Validaciones

Validaciones: por qué

**Convención:** validar en ruby, no en db

No guardar datos incorrectos

No depender de la implementación BBDD

Mejores (mágicos) mensajes de error

**sí validan:**

**create**

**create!**

**save \***

**save!**

**update**

**update\_attributes**

**update\_attributes!**

no validan:

decrement!

decrement\_counter

increment!

increment\_counter

toggle!

touch

update\_all

**update\_attribute**

update\_column

update\_counters

## validando 'a mano'

```
>> course.valid?  
>> course.invalid?  
provocan validaciones  
rellenan model.errors  
devuelven true/false  
  
>> course.errors  
=> {:field => ["can't be something"], ...}  
  
>> Course.new.errors[:name].any?  
=> false  
>> Course.create.errors[:name].any?  
=> true
```

## Validaciones: acceptance

`validates :terms_of_service, :acceptance => true`

**Para usar ckeck box (lo veremos)**

**¿Es imprescindible salvar?**

**Opción :on (:create, :update, :save)**

**¿Cuándo? ¿Ejemplos?**

## associated

```
belongs_to :course # veremos 'belongs_to' T.4  
validates_associated :course, :if => :course_id  
# sólo en un lado de la asociacion!
```

```
validates :course_id, :presence => true
```

**comprueba validez, no presencia**



# confirmation

```
validates :email, :confirmation => true
```

```
# En la vista hay dos campos en vez de uno:
```

```
<%= text_field :user, :email %>
```

```
<%= text_field :user, :email_confirmation %>
```

```
# Sólo se mira si hay field_confirmation
```

```
validates :email_confirmation, :presence => true
```

# inclusion

```
validates :size, :inclusion => { :in => 1..12,  
  :message => "%{value} is not a valid month" }
```

**La lista dada a `:in` es cualquier cosa que responda a `.include?`  
Array, Range, String\*...**

<http://ruby-doc.org/core-1.9.3/Enumerable.html#method-i-include-3F>

**Tiene un gemelo malvado llamado...**

en un giro sorprendente de los acontecimientos:

## exclusion

```
validates :number, :exclusion => { :in => 1..3,  
  :message => "%{value} is not a basket player number" }
```

## format

```
validates :email,  
  :format => { :with => /\A.+@\w+\.\w+\z/,  
  :message => "not a valid email" }
```

### Usos:

Correo

Nombres de usuario

DNI/NIE

Contraseñas (¡Ojo!)

Códigos postales

Teléfonos

# Custom: validates\_with

Proveemos una *clase validadora*

```
class Person < ActiveRecord::Base
  validates_with GoodnessValidator
end

class GoodnessValidator < ActiveModel::Validator
  def validate(record)
    if record.first_name == "Evil"
      # errores al modelo entero, no a un atributo
      record.errors[:base] << "This person is evil"
    end
  end
end
```

# Custom: validates\_each

## Hacemos un bloque

```
class Person < ActiveRecord::Base
  validates_each :name, :surname do |record, attr, value|
    msg = 'must start with upper case'
    record.errors.add(attr, msg) if value =~ /\A[a-z]/
  end
end
```

## Ejercicio

**Generar un scaffold de Course**

**Validar que haya nombre**

**Arrancar rails mostrando los cursos**

**Dar un curso de alta (sin nombre)**

**Modificarlo, Borrarlo**

**¿Preguntas?**

## ayudas

```
$ rails g scaffold Course name:string ... -s  
invoke active_record  
skip db/migrate/20130422213321_create_courses.rb  
skip app/models/course.rb  
$ rails s ... Ctrl-C  
$ rm public/index.html  
$ edit config/routes.rb  
# root :to => 'courses#index'  
$ rails s  
$ browser localhost:3000
```



# Tests

**¿Test Driven Development?**

**¿Por qué probar? El precio de la seguridad**

**<http://rubykoans.com/>**

**Entornos de Rails:  
producción, desarrollo y pruebas**

**Tecnologías de test:  
rspec, cucumber...**

**Mocking y Stubbing**

# lo que rails trae

**Estructura preparada  
unit, functional, integration, benchmark**

**Base de datos separada**

**Helper específicos**

**Fixtures (datos de prueba)**

**Tareas rake (para ejecutarlos)**

# Testeos Unitarios

Comprueban el modelo/objeto aislado

No probamos que AR funciona

Pueden hacerse con la *BD en memoria*

Pueden hacerse *sin BD* pero

Buen sitio para probar especificidades BD

¡Principio de Demeter!

Demeter y los Mocks/Stubs

¿cómo se testea ?

**Testeamos mediante aserciones  
son como alarmas**

**"pasa esto, o me avisas"**

`assert_something` esperado, testeado, mensaje

**veamos algunas de los helpers**

## ¿existe? ¿es cierto?

```
assert bool_val, [msg]
```

```
course = Course.new
```

```
assert course.active?, "debe estar activo por defecto"
```

```
assert_nil obj, [msg]      # obj.nil?
```

```
assert_not_nil obj, [msg]  # !obj.nil?
```

# ¿es lo mismo?

```
assert_equal obj1, obj2, [msg] # testing for '=='  
assert_not_equal obj1, obj2, [msg]  
assert_same obj1, obj2, [msg] # testing for '.equal?'  
assert_not_same obj1, obj2, [msg]
```

```
> 1 == 1.0          # => true  
> 1.equal?(1.0)     # => false  
> "a" == "a"        # => true  
> "a".equal? "a"    # => false  
> :a.equal? :a      # => true  
> c1, c2 = Course.last, Course.last  
> c1 == c2          # => true  
> c1.equal? c2      # => false
```

# ¿encaja?

```
# en la Regexp: %r{r}.match(s) , /r/ =~ s  
assert_match regexp,string,[msg]  
assert_no_match regexp,string,[m]
```



¿casca?

```
assert_throws( symbol, [msg] ) { block }  
assert_raise( exception1, ... ) { block }
```

## Ejercicio: testeos unitarios

**Escribir tests/migraciones para Course:**

**Título no puede estar vacío**

**Activo no puede ser nulo**

**Fecha debe ser "futura" si está activo**

**Extra:**

**Completar el formulario y probar validaciones**  
*checkbox, dateselect*

# Ejercicio: ayudas

```
# test/unit/course_test.rb
require 'test_helper'

class CourseTest < ActiveSupport::TestCase
  test "a course without title" do
    @course = Course.new
    assert !@course.valid?, "can't be valid"
    assert_equal false, @course.save, "must not save"
    assert_raise(ActiveRecord::RecordInvalid){ @course.save! }
  end
end

# En la consola:
# ruby -Itest test/unit/course_test.rb<br>
# rake test:units
```

# Fixtures

**Valores que se cargan en la BD de pruebas**

**Representan un 'estado inicial'**

**Pueden usarse juntas o separadas**

**Escalan mal con la complejidad**

**Alternativas: Machinist, Factory Girl...**

# Más alla de los unitarios

**functional -> controlador**

**integration -> caso de uso**

**performance carga/respuesta**

# Aserciones de Rails

**Para testear cosa 'web'  
se usan en funcionales/integración**

```
assert_valid(record) # deprecate a -> assert record.valid?
```

```
assert_difference(expressions, difference = 1, message = nil) {...}
```

```
assert_no_difference(expressions, message = nil, &block)
```

```
assert_difference 'Course.count' do
```

```
  post :create, :course => {...}
```

```
end
```

# genera y reconoce tal ruta

assert\_recognizes(expected\_options, path, extras={}, message=nil)

assert\_generates(expected\_path, options, defaults={}, extras = {}, message=nil)

# es un 200 OK ?, un 302 redirect

assert\_response(type, message = nil)

assert\_redirected\_to(options = {}, message=nil)

# usa tal plantilla/vista

assert\_template(expected = nil, message=nil)



en el tintero/para ampliar

## **mocking/stubs: tras asociaciones**

Mejor despues del curso / 1ª app:

<http://blowmage.com/minitest-rails/>

<http://rspec.info/>

<http://www.fastrailstests.com/>

# Consultas

**Ya sé que estaba en el tema 5 del índice, pero es difícil seguir sin verlo**

**¿por qué no SQL?**

**DRY**

**DB agnóstico**

**tipos Ruby**

**funcionalidad extra en modelos**

**relaciones**

**consultas dinámicas**

**evaluacion perezosa**

# find

```
Model.find( 1 )
```

```
# SELECT * FROM clients WHERE (clients.id = 10) LIMIT 1
```

```
Model.find( [1,3,7] )
```

```
# SELECT * FROM clients WHERE (clients.id IN (1,3,7))
```

```
# ActiveRecord::RecordNotFound si no vienen todos
```

# first, last

`Model.first`

```
# SELECT * FROM clients LIMIT 1
```

`Model.last`

```
# SELECT * FROM clients ORDER BY clients.id DESC LIMIT 1
```

`Model.last(10)`

```
# SELECT * FROM clients ORDER BY clients.id DESC LIMIT 10
```

```
# first! y last! -> raise RecordNotFound
```

# where + string

# Rails guarda en params los parametros de la peticion HTTP

```
Course.where("location = #{params[:location]}") # NO! SQLi
```

```
Course.where("location = ?", params[:location])
```

```
Course.where(" location = ? AND start_date > ?", [loc, date] )
```

```
Course.where("start_date <= :start", {:start => Date.today})
```

# where + hash

Válido para algunas consultas

```
Course.where(:active => true) # ... WHERE active = 1
```

```
# yday = (Time.now.midnight - 1.day)..Time.now.midnight
```

```
Course.where(:created_at => yday)
```

```
# ... created_at BETWEEN '2013-05-06' 00:00:00
```

```
# AND '2013-05-07' 00:00:00
```

```
Course.where(:location => ['Madrid', 'Barcelona'])
```



# order

**Especifica el orden, pueden ser varios, cuidado con ambigüedad**

```
Course.order(:created_at)
```

```
Course.order("created_at")
```

```
Course.order("created_at DESC")
```

```
Course.order("created_at ASC")
```

# select

```
Course.select("location")  
# retorna objetos de sólo lectura  
# para qué
```

```
Course.select("location").uniq  
# por ejemplo para sugerir en un combo
```

# Ejercicio

**"filtros" en index**

**courses#index muestra solo activos  
cursos salen en orden anticronológico**

**completar formularios**

**ver consultas en log**

**ver dev-tools/firebug**

**railspanel...**

el resto?

**Hay más...**

**...en el tema 5**



# Introducción a Ruby on Rails

## 4 - Conectando vuelos

Mayo 2013  
Fernando Martínez



# Parte 4

**MOAR MODELS!**

**distintas entidades**

**y sus relaciones**

**y magia!**

**en realidad es OO**

**objetos como atributos de otros**

**@post.comments**

0\_0





# RELACIONES

**belongs\_to**

**has\_many**

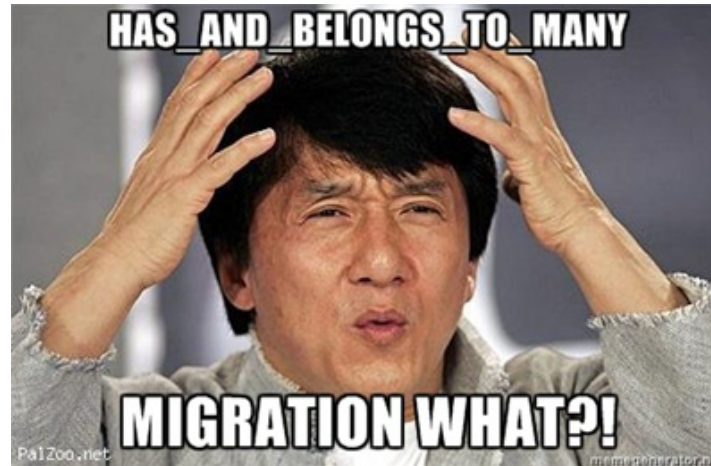
**has\_one**

**has\_many :through**

**has\_one :through**

**has\_and\_belongs\_to\_many**

# WTF



dependiendo de la experiencia de cada uno en BBDD  
relacionales

por partes: como Jack el destripador

**veamos** belongs\_to y has\_many

## belongs\_to

```
class Edition < ActiveRecord::Base
  belongs_to :course, :dependent => :destroy
end
```

**el modelo que pertenece depende del otro  
y almacena su id (hay que migrar)  
puede destruirse automaticamente (Samurai)**

**belongs\_to nos da**

```
association(force_reload = false)
association=(associate)
build_association(attributes = {})
create_association(attributes = {})
```

**¿Como se llaman las de Edition ?**

belongs\_to nos da

```
@edition.course(force_reload = false)
@edition.course=(associate)
@edition.build_course(attributes = {})
@edition.create_course(attributes = {})
```

**pero no me voy a acordar!**

**ni falta que hace**

**se irá quedando**

**hasta entonces...**

**para recordar uno: <http://goo.gl/it6eZ>**

**para elegir entre las que hay:  
<http://goo.gl/TCel9>**

¿estaban ya en favoritos?

<http://ruby-doc.com/>

<http://api.rubyonrails.org/>

<http://apidock.com/>

**propina:** <http://dochub.io/>



## **belongs\_to: opciones**

`:autosave, :class_name, :conditions, :counter_cache,  
:dependent, :foreign_key, :include, :inverse_of,  
:polymorphic, :readonly, :select, :touch, :validate`

**para saltarse convención,**

**lo mejor: no, hasta que os haga falta**

**los defaults son astutos**

**o funcionar de otra forma (veremos)**

**belongs\_to: opciones destacadas**

**:dependent - ¿qué hacer en asociado al borrar?**

**:inverse\_of - informa a has\_many (!duplicados)**

**has\_one**

**relacion 1:1**

**muy parecido a belongs\_to**

**mismos métodos 'gratis'**

**diferencia sobre todo conceptual**

**el *id* esta en el belongs\_to**

# has\_many

```
class Course < ActiveRecord::Base
  has_many :editions, :inverse_of => :course
end
```

# has\_many

```
collection(force_reload = false)
collection<<(object, ...)
collection.delete(object, ...)
collection=objects
collection_singular_ids
collection_singular_ids=ids
collection.clear
collection.empty?
collection.size
collection.find(...)
collection.where(...)
collection.exists?(...)
collection.build(attributes = {}, ...)
collection.create(attributes = {})
```

¿qué vemos respecto a belongs\_to?

**hay más métodos**

**build y create son distintos**

**<<, empty, size, delete funciona como Array**

**antes de seguir...**

**veámoslo en la aplicación  
un *Curso* se puede hacer varias veces  
separemos la identidad del resto  
creemos Edition**

## **Ejercicio**

**Un curso tiene ediciones**

**Una edición pertenece a un curso**

**se queda las fechas, ubicación**

**tiene seats(aforo), precio**



## **Ejercicio: notas**

**vamos a mezclar cosas aun no vistas...**

**...para que el ejemplo se más real**

**¿Formularios?**

**Veamos algo de vistas/rutas/controlador**

# rutas anidadas

HTTP Verb	Path	action	used for
GET	/courses/:course_id/editions	index	display a list of all editions for a specific course
GET	/courses/:course_id/editions/new	new	return an HTML form for creating a new edition belonging to a specific course
POST	/courses/:course_id/editions	create	create a new edition belonging to a specific course
GET	/courses/:course_id/editions/:id	show	display a specific edition belonging to a specific course
GET	/courses/:course_id/editions/:id/edit	edit	return an HTML form for editing an edition belonging to a specific course
PUT	/courses/:course_id/editions/:id	update	update a specific edition belonging to a specific course
DELETE	/courses/:course_id/editions/:id	destroy	delete a specific edition belonging to a specific course

# ¿Cómo lo hacemos?

todos juntos o por parejas

partimos del *scaffold*

sin formulario anidado (para simplificar)

sin gemas de formularios (por ahora)

mirando: <http://goo.gl/7MhPA> (DRY :-)

Parent → Course, Child → Edition

¡Ojo con las migraciones! que ya tenemos Course

## Ejercicio ayudas

```
$ rails g scaffold Edition start:date finish:date seats:integer  
price_in_cents:integer location:string course:references  
invoke active_record  
create db/migrate/20130503213742_create_editions.rb  
create app/models/edition.rb  
invoke test_unit  
...
```

y después

relaciones en los modelos

rutas anidadas

**quitar** start\_date y location **de** Course

## visita rapida al mundo de los controladores

`load_course` en `EditionsController`  
**variables de instancia en controlador**  
**helpers de ruta y redirecciones**  
**formularios:** `number_to_currency`  
**atributos proxy** `price`, `price=`  
**testeos y validaciones (\*)**

**relaciones en los modelos**  
**como hemos visto en el tema**  
**las hacemos inversas**

**rutas anidadas**

**como en el enlace**

**las probamos con rake routes**



quitar start\_date y location de course

**editando la migración**

**copiando los datos de Course**

**borrando las columnas duplicadas**

**con up y down**

**conservando datos en ambos!**

ya que estamos

**Evolución de datos**

**planificación**

**integridad referencial**

`load_course` en `EditionsController`

un metodo privado en el controlador

si es privado no es una *acción*

¿es DRY? hablemos de CanCan

variables de instancia en controlador

**cargar a partir de @course**

`@course.editions.find(params[:id])`

**seguridad estructural (Diáspora)**

helpers de ruta en redirecciones

**más magia de nombres...**

**que no tenemos que recordar**

`rake routes`

formularios: básico

`form_helper`

**hay más en la Parte 6**

**si os suena a uzbeko lo miramos**

# formularios (REST)

```
<%= form_for(@course) do |f| %>
  <%= f.label :title %>: <%= f.text_field :title %>
  <%= f.label :description %>: <%= f.text_field :description %>
  <%= f.submit %>
<% end %>
```

**que más tienen los formularios**

**campos de errores**

**html estructural para ayudar a los estilos**

**opción :remote**

**el tema 6 :-)**



`number_to_currency, price, price=`

**Superman 3 y los medios centavos**

**MVP, necesidades/recursos**

**guardamos una cosa distinta a la mostrada**

**para producción:** <https://github.com/RubyMoney/money-rails>

## **En casa: testeos y validaciones**

**¿Os atreveis a testear el controlador?  
usando los asserts vistos  
testead al menos una de las vistas  
encontrad algo que podría hacer y aun no hace**



# Introducción a Ruby on Rails

## 5 - La cinta de equipaje

Mayo 2013  
Fernando Martínez



# Parte 5

**veremos:**

**cosas de la BD**

**otras cosas que se guardan**

más alla de *first*, *last* y *all*

**joins**

**scopes, condition override**

**dinamic\_finders**

**calculations**

```
class Category < ActiveRecord::Base
  has_many :products
end

class Product < ActiveRecord::Base
  belongs_to :category
  has_many :comments
  has_many :tag
end

class Comment < ActiveRecord::Base
  belongs_to :product
  has_one :guest
end

class Guest < ActiveRecord::Base
  belongs_to :comment
end

class Tag < ActiveRecord::Base
  belongs_to :product
end
```

## joins automáticos

```
>> Category.joins(:products)
```

```
SELECT categories.* FROM categories  
INNER JOIN products ON products.category_id = categories.id
```

```
>> Product.joins(:category, :comments)
```

```
SELECT products.* FROM products  
INNER JOIN categories ON products.category_id = categories.id  
INNER JOIN comments ON comments.product_id = products.id
```

```
>> Product.joins(:comments => :guest)
```

```
SELECT products.* FROM products  
INNER JOIN comments ON comments.product_id = products.id  
INNER JOIN guests ON guests.comment_id = comments.id
```



## triple salto mortal

```
>> Category.joins(:products => [{:comments => :guest}, :tags])
```

```
SELECT categories.* FROM categories
```

```
INNER JOIN products ON products.category_id = categories.id
```

```
INNER JOIN comments ON comments.product_id = products.id
```

```
INNER JOIN guests ON guests.comment_id = comments.id
```

```
INNER JOIN tags ON tags.product_id = products.id
```

acerca de los *joins*

se puede limitar, con `where`

`.joins` sólo hace *inner joins*

(aunque hay una forma de hacer *outer*)

N+1!

¿ N+1 ?

```
>> clients = Client.includes(:address).limit(10)
>> clients.each do |client|
>> puts client.address.postcode
>> end
```

11 peticiones, a la BD, con sus viajes de red

```
>> Category.includes(:posts => [{:comments => :guest}, :tags]).find(1)
2 peticiones
```

# scopes

**Cuando alguna de estas peticiones se reutiliza  
o la abstraeríamos en un metodo  
hacemos un *scope***

```
scope :published, where(:published => true)  
scope :last_week, lambda { where("created_at <  
                                ?", Time.zone.now ) }
```

## scopes con argomento

```
class Edition < ActiveRecord::Base
  scope :starting_soon, lambda { |date|
    where("start < ?", date)
  }
end
```

## Ejercicio 1

un scope acepte argumentos o no  
no *nil* sino ser llamado *sin* argumentos

Edition.starting\_soon

Edition.starting\_soon(1.week.since(Time.now) )

¿podeis encontrarlo?

## salida deseada

```
>> Edition.soon(1.day.since(Time.now))
```

```
Edition Load (0.5ms) SELECT "editions".* FROM "editions" WHERE (start < '2013-05-07 14:53:44.752681')
```

```
=> [#<Edition id: 1...]
```

```
>> Edition.soon
```

```
Edition Load (0.5ms) SELECT "editions".* FROM "editions" WHERE (start < '2013-05-13 14:53:47.827892')
```

```
=> []
```

## solución

```
>> class Edition < ActiveRecord::Base
>> scope :soon, lambda {|*args|
>>   where( "start < ?",
>>   (args && args.first ? args.first : 1.week.since(Time.now)))}
>> end
```



## **Ejercicio 2**

**revisar el código de la aplicación  
localizar puntos de aprovechamiento  
agregar un joins, un includes y un scope  
¿se testea?, ¿a que nivel?**

ayudas

en la app de ejemplo:

`link_to_unless`

**Courses filtrados con scope  
params no REST**

# Dinamic finders

## dynamic finders

`Model.find_by_foo_and_bar`

`Model.find_all_by_foo`

`Model.find_last_by_foo`

`Model.where(...).first_or_create(atts)`

`Model.where(...).first_or_initialize()`

# pluck

```
Edition.pluck(:id) # en vez de: Edition.select(:id).map{|e| e.id}
```

```
Edition.where(:active => true).pluck(:id)  
# SELECT id FROM editions WHERE active = 1
```

```
Edition.uniq.pluck(:location)  
# SELECT DISTINCT location FROM editions  
# por ejemplo para sugerir locations en el form
```

# Calculations

# count

`Client.count`

```
# SELECT count(*) AS count_all FROM clients
```

`Client.where(:first_name => 'Ryan').count`

```
# SELECT count(*) AS count_all FROM clients WHERE (first_name = 'Ryan')
```

# contar 'pro'

```
Client.includes("orders").where( :first_name => 'Ryan',  
                                :orders => { :status => 'received' }).count
```

```
# SELECT count(DISTINCT clients.id) AS count_all FROM clients  
# LEFT OUTER JOIN orders ON orders.client_id = client.id WHERE  
# (clients.first_name = 'Ryan' AND orders.status = 'received')
```



# ¿que tenemos?

**Comodos:**

minimun

maximum

sum

**Versátil**

calculate

# ¿y si ActiveRecord 'arrastra' ?

**si las consultas van lentas...**

**usar explain**

**que puede ser automático:**

`config.active_record.auto_explain_threshold_in_seconds`

**y varía segun el motor de BD**

Cache

¿caché o qué?

**aquí iba un rollo sobre las caches en Rails**

**las viejas:**

[http://guides.rubyonrails.org/caching\\_with\\_rails.html](http://guides.rubyonrails.org/caching_with_rails.html)

**pero en caché, Rails 4 *lo peta***

**Resúmen:** <http://goo.gl/CRXub>

**Buscad a david: (en unos días)**

<http://confreaks.com/events/railsconf2013>



# Introducción a Ruby on Rails

## 6 - Fotos de Vacaciones

Mayo 2013  
Fernando Martínez



# Parte 6

# V de Vendetta, o de Vista

**Layouts, vistas y parcial**

**Motores de rendering**

**Recursos y representaciones**

**Helpers y form helpers**

**UI Frameworks & kits (T.8)**

## Layouts, vistas y parcial

**Layout:** repetición externa `application.erb`

**Vista:** representacion del recurso `new.erb`

**Partial:** repetición interna (plantilla) `_form.erb`

**Helper:** repetición interna (código) `link_to`



## Layout

**Ficheros en:** app/views/layouts

**Por defecto:** application.html.erb

**A nivel controlador:** *método*

`layout "admin"`

**A nivel acción:** *opción*

`render :layout = false`

**Para más 'rellenos':**

`content_for :wadus`

## Vista

**en** app/views/{controller}/{accion}.{motor}  
**render "edit" en update sólo cambia de acción**  
**entra en el yield del *layout***  
**recibe las variables de instancia @cosa de la acción**  
**llama a *partials* y *helpers***

## Vista o qué

se pueden renderizar:

:nothing cabeceras, status, sin body

:inline AKA plantilla entre comillas (\*)

:template "controlador/accion", no ejecuta la otra acción

:file ruta absoluta, fichero literal

:text llamadas AJAX - html

:builder llama a .to\_xml

:json JSON llama a .to\_json

:js javascript 'en seco'

**uso avanzado**

**el camino *default* es *casi* automatico  
para todo lo demás**

**por ejemplo :status (esencial para ajax)**

**<http://apidock.com/rails/ActionController/Base/render>**

## Partial

**en** `app/views/{controller}/_{nombre}.{motor}`

**un trozo que aparece en varias vistas**

**o en una repetidamente**

**variable 'mágica' con nombre fichero**

**se asigna con** `object: @post` **o** `collection:`  
`@post.comments`

`collection` **itera tácitamente con**  
`@col.each do |nombre_partial|`

# motores de rendering

**ERB (html,css,js)**

**Haml/Sass/Scss**

<http://haml.info/> | <http://sass-lang.com/>

**Slim** <http://slim-lang.com/>

**Builder (xml) Atom/Feed...**

# ERB

`<% Ruby code -- inline with output %>`  
`<%= Ruby expression -- replace with result %>`  
`<%# comment -- ignored -- useful in testing (*) %>`  
`<%% or %%> -- replace with <% or %> respectively`

**Viene con Ruby**  
**Default en Rails**  
**Fácil desde PHP, ASP...**

# Recursos y Representaciones

¿que dice REST?

Recurso != representacion

/courses/1 **no** es @course

ejemplo blog/feed

<http://weblog.rubyonrails.org/>

<http://weblog.rubyonrails.org/feed/atom.xml>



# Helpers

**helper:** función auxiliar  
abstrae comportamiento reutilizable  
relativo a la vista/representacion  
vienen muchos en Rails `ActionView::Helpers`

# Helpers generales

`stylesheet_link_tag`

`javascript_include_tag`

`link_to`

`url_for`

`number_to_currency`

`image_tag`

**hay muchos, descubridlos poco a poco**

**en serio: <http://apidock.com/rails/ActionView/Helpers>**

# FormHelpers

dos *familias*: los \*\_tag y los *object*

los *tag*: genera html normal

los de *objeto*: optimizados para recursos Rails

# los forms Rails

**html normal**

**convenciones de nombres**

**namespace con '[' y '']'**

**por ejemplo:**

## form\_for en acción

```
<%= form_for @article, :url => { :action => "create" },  
  :html => {:class => "nifty_form"} do |f| %>  
  <%= f.text_field :title %>  
  <%= f.text_area :body, :size => "60x12" %>  
  <%= f.submit "Create" %>  
<% end %>  
  
<form accept-charset="UTF-8" action="/articles/create" method="post" class="nifty  
  <input id="article_title" name="article[title]" size="30" type="text" />  
  <textarea id="article_body" name="article[body]" cols="60" rows="12"></textarea  
  <input name="commit" type="submit" value="Create" />  
</form>
```

hay gente que nunca tiene bastante

**Formtastic**

**SimpleForm**

algunos integran con Bootstrap y amigos

tutoriales en Railscasts

[https://www.ruby-toolbox.com/categories/rails\\_form\\_builders](https://www.ruby-toolbox.com/categories/rails_form_builders)



# Introducción a Ruby on Rails

## 7 y 8 - Consejos para Viajes complicados

Mayo 2013  
Fernando Martínez

**\*redradix**

# Parte 7



## la Web - entorno abierto

**hay que desconfiar :'(**  
**sanitizar entradas...**  
**... y salidas (h() y raw())**  
**y restringir**

# Autenticación:

hubo una vez que se hacía a mano

30-8-2006: [Dave Astels vs Rails Recipes](#)

mucho ha llovido, ahora tenemos [Devise](#) peero

Ctrl-F -> "Starting with Rails?"

¿entonces? por ejemplo [Authlogic](#)

## Ejercicio

En el repo teneis: Authlogic de corta-pega

El que dude, que lo siga (ojo rails 2) hasta quedar igual

El que se atreva que vaya a por la Versión moderna  
proteged acciones delicadas con `required_user`  
¿y si hay distintos tipos de usuarios?

# ¿y que más?

**recuperar contraseña por mail  
activar, bloquear usuarios  
limitar intentos de acceso  
recordar al usuario  
por todo eso existe Devise**

# Autorización/permisos/roles

hay conocidos y conocidos...  
el cartero puede repartir tu correo...  
...pero no leerlo

¿Quien puede hacer qué? (pensando en REST)

CanCan

Compatible con Devise y Authlogic

## ...un botón

```
<% if can? :update, @article %>  
  <%= link_to "Edit", edit_article_path(@article) %>  
<% end %>
```

```
class ArticlesController < ApplicationController  
  load_and_authorize_resource  
  
  def show  
    # @article is already loaded and authorized  
  end  
end
```

# Parte 8

# Consejos de viaje

**Aprovecha el Duty Free: ActiveSupport**

**Lleva un diccionario: Traducción y localización**

**Viaja ligero: responsive web design y mobile first**

**Frameworks de diseño: Bootstrap, Foundation,  
960.gs**



# Aprovecha el Duty Free: ActiveSupport

```
$ @var.blank? # @var.nil? || @bar.empty?
```

```
$ @var.present? # !@var.blank?
```

```
$ @var.presence # @var.present ? @var : nil
```

```
$ host = config[:host].presence || 'localhost'
```

## ActiveSupport - 2

```
$ @bar.try(:wadus,1) # @bar.nil? ? nil : @bar.wadus(1)
$ @person.try { |p| "#{p.first_name} #{p.last_name}" }
```

```
$ 1.in?(1,2) # => true
$ alias_attribute :login, :email
$ attr_accessor_with_default :port, 80
$ delegate :attr_name, :to => :asocc_name
```

## ActiveSupport - 3

```
$ "".html_safe? # => false
$ s = "<script>...</script>".html_safe
$ s.html_safe? # => true
$ s # => "<script>...</script>"
$ # no comprueba, solo recuerda
```

y la magia de nombres

truncate

pluralize, singularize, humanize, foreing\_key

camelize, titleize, underscore, dasherize

parametrize, tableize, classify, constantize

etc... [ActiveSupport CoreExtensions](#)

**Lleva un diccionario: Traducción y localización**

**es difícil explicarlo mejor que:**

**<http://guides.rubyonrails.org/i18n.html>**

**sí, es un peñazo**

**sí, es mucho trabajo extra**

**hacedlo si os reporta beneficio**

**desde el principio o muy difícil**

# Ejemplo - I18n, vistas

```
# config/locales/es.yml
```

```
es:
```

```
  courses:
```

```
    index:
```

```
      title: "Título"
```

```
# app/views/courses/index.html.erb
```

```
<%= t '.title' %>
```

# Ejemplo - I18n

```
# config/locales/es.yml
```

```
es:
```

```
  activerecord:
```

```
    models:
```

```
      course: Curso
```

```
  attributes:
```

```
    course:
```

```
      title: "Título" # las comillas conservan el acento
```

```
# app/views/courses/index.html.erb
```

```
...
```

```
<th><%= Course.human_attribute_name("title") %></th>
```

```
...
```

**No todo es horror**

**Rails trae mucho hecho**

**Mecanismo MUY completo: plurales, modelos,  
interpolación**

**gema con traducciones: [rails-i18n](#)**

**Errores, fechas, formatos en español:  
<http://goo.gl/npdIC>**

**integración con muchas gemas como Devise**

**Contenido: otra historia [Globalize](#)**



**Viaja ligero: responsive web design y mobile first**

**no voy a comisión ni nada pero:**

**ABA: Mobile First**

**ABA: Responsive Web Design**

**como la traducción: desde el principio mas fácil**

## Frameworks de diseño:

El rey: Twitter Bootstrap  
gema, generators, SimpleForm

Foundation dice ser *mobile first*

960.gs famoso

Bourbon Sass mixin, ligero

HTML KickStart

## Frameworks de diseño - cont

**elegir al principio**

**preguntad a vuestro frontend-person**

**personalizar por override = actualizable**

**ojo con Sprockets**

# Epílogo

Para empezar:

Aprended Ruby

Agile Web Development with Rails 4 ebook y papel: \$\$\$

Rails Tutorial Rails gratis, papel y videos: \$\$\$

Rails for Zombies gratis, continuaciones de pago

Para mejorar:

Ruby Koans Aprender ruby arreglando tests

The Rails 4 Way

Rails Recices

Rails Best Practices

para petarlo

Metaprogramming Ruby **de Paolo Perrotta**

Blog de Aaron Patterson