Introducción a Ruby y Rails

Julio 2015

Fernando Martínez

1 - Preparativos

Instalación y configuración

¿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, ...

Instalación y configuración

- Doc oficial
- muchas opciones

paquete

instalador

manejador

compilar

nuestra elección:

RVM

instrucciones detalladas en rubyonrails.org.es

ejemplo (Linux):

```
$ sudo apt-get install -y git-core subversion
...
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
...
$ \curl -sSL https://get.rvm.io | bash -s stable --rails --ruby
...
$ type rvm | head -n -1
rvm is a function

$ ruby -v
ruby 2.2.xpyyy (2015-zz-ww revision RRRRR) [x86_64-linux]
```

alternativa - online

Cloud9 - c9.io

Nitrous - n2o.io

Digital Ocean

2 - Introduccion a Ruby

Hablando con Ruby (por telegrama)

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

Hablando con Ruby (por sms)

```
$ irb
>

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

>> puts "Hola mundo"
Hola mundo
=> nil
```

IRB, variables, I/O básica y 1er programa

¿De qué están hechos los programas?

Datos (variables, classes, objects...)

Decisiones (conditionals)

Repetición (loops)

Datos

Datos - Clases nativas

Numeric

String

Array (lista)

Hash (diccionario)

Range

Calculadora gratis!

```
>> 3 + 2
=> 5
>> 3 * 2
=> 6
>> 3 ** 2
=> 9
>> Math.sqrt(9)
=> 3.0
```

Números en Ruby

son objetos

los operadores son métodos

la sintaxis lo disimula

Números

```
>> 42
=> 42
>> 42.class
=> Fixnum
>> 1.2.class
=> Float
>> 1_000_000_000_000_000_000.class
=> Fixnum
>> 10_000_000_000_000_000_000.class
=> Bignum
```

Números como objetos

```
>> 12.odd?
=> false
>> 12.even?
=> true
>> 12.zero?
=> false
>> 12.nonzero?
=> 12
```

Números como objetos

```
>> 1 + 1
=> 2
>> 2.* 2
=> 4
>> 4.-(1)
=> 3
>> 3.send(:**, 2)
=> 9
```

String - métodos

```
>> 'hola'.length
=> 4
>> 'hola'.reverse
=> "aloh"
>> 'hola'.upcase
=> "HOLA"
>> 'Hola'.downcase
=> "hola"
```

String - métodos

```
>> 'hola'.next
=> "holb"

>> # "¿por qué?"

>> ('a'..'h').to_a
=> ["a", "b", "c", "d", "e", "f", "g", "h"]
```

String - comillas

```
>> name = "Fer"
=> "Fer"
>> "my name is #{name}"
=> "my name is Fer"
>> 'my name is #{name}'
=> "my name is #{name}"
```

String - interpolación

```
>> "my name is #{nome}"
=> "my name is "
>> "my name is " + nome
=> KABOOM!
```

Array

lista

almacenta 'filas' de valores

no limita tipo

anida

Array

```
>> a = [1,1,2,3,5,8,13]
=> [1, 1, 2, 3, 5, 8, 13]
>> a.length
=> 7
>> a[0]
=> 1
>> a[-1]
=> 13
>> a[1,2]
=> [1, 2]
>> a[3..5]
=> [3, 5, 8]
```

Hash

diccionario

almacena valores tras claves

la clave puede ser cualquier objeto

se suelen usar símbolos :como, :estos

Hash

```
>> juan = { name: 'Juan', age: 32 }
=> {:name=>"Juan", :age=>32}
>> juan[:name]
=> "Juan"
>> juan[:age]
=> 32
>> juan[:height]
=> nil
```

Hash, fetch y default

```
>> juan.fetch(:height)
KeyError: key not found: :height

>> luis = Hash.new('desconocido')
=> {}
>> luis.merge!(name: 'Luis')
=> {:name=>"Luis"}
>> luis[:age]
=> "desconocido"
```

experimentos

http://www.sitepoint.com/algorithmic-fun-ruby-hashes/

Range

como en matematicas

relacionado con los enumerables

se puede hacer sobre muchos tipos

si tienen .prev y .next

Range

member?, include?, ===

min, max

each, step(n)

to_a

Range

```
>> (1..10).to_a
=>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>> ('a'..'z').to_a
=> ["a", "b", "c", "d", "e", "f", "g", "h", "i"...]
```

Métodos, definición, llamada, argumentos y retorno

Métodos

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

# se llaman asi:
nombre('p1', 'p2')
nombre('p1')
nombre 'p1'
nombre 'p1', 'p2' # cuidado!
```

Ejercicio: Métodos (irb)

Definid un metodo y ejecutadlo

Solución: Métodos (irb)

```
>> def h
>> puts "Hello World!"
>> end
=> :h # devuelve el nombre del método
>> h()
Hello World!
=> nil
>> h
Hello World!
=> nil
```

Ejercicio: Parámetros (irb)

Definid un método con parámetro

Solución: Parámetros

```
>> def greet(name)
>> puts "Hello #{name.capitalize}!"
>> end
=> :greet
>> greet "chris"
Hello Chris!
=> nil
>> greet
ArgumentError: wrong number of arguments (0 for 1)
...
```

Ejercicio: Parámetros 2 (irb)

Definid un método con parémetro (opcional)

Solución: Parámetros 2

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

Control de flujo, bucles e iteraciones

Decisiones / Condicionales

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

action if condition

action unless condition
```

Decisiones / Condicionales (cont)

```
foo = condition ? value_if_true : other_value


condition && value_if_true || other_value
# vs
condition and action_if_true or other_action
```

precedencia de operadores

```
valor = a_condition and another
(valor = a_condition) and another

valor = a_condition && another
valor = (a_condition && another)

# . > < == && || .. ... (?:) = not and or if unless while until
```

Repetición / bucles

```
while conditional [do]
   code
end

code while condition

until conditional [do]
   code
end

code until conditional
```

Más Repeticiones

```ruby
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-2.2.2/Enumerable.html
```

Orientación a objetos, clases e instancias

Ejercicio: Nuestro primer archivo ruby

```ruby
# escribid esto en 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

```
# desde la carpeta en que está greeter.rb
$ 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
```

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 :)
```

Iteración de colecciones: Enumerable

Enumerable

implementa each

incluye el modulo Enumerable

Array, Hash, Range, ActiveRecord::Collection

Enumerable: preguntas

all?

any?

include? (ailas: member?)

none?

one?

Enumerable: filtros

detect (alias: find)

select (alias: find_all)

reject

Enumerable: filtros cont.

min

min_by

max

max_by

count

partition

Enumerable: colección

map (alias: collect)

sort

sort_by

group_by

inject (alias: reduce)

Ejercicio:

```
words = File.read('/usr/share/dict/spanish').split("\n")
```

Ejercicio:

¿la palabra mas larga?

¿en que letra del diccionario hay mas voces?

¿en que letra hay voces mas largas?

¿cual es la palabra con mas vocales / consonantes?

3 - Introducción al Control de versiones con Git

Control de versiones

Erase una vez...

```
tar -czvf ../proj_$(date +%Y%m%d_%H%M%S).tgz
```

Control de versiones

¿Usais control de versiones?

¿Alternativas?

CVS 1990-2008

Subversion (Apache) 2000-...

Git (Linus) 2005-...

material de referencia

web http://git-scm.com/

documentación http://gitref.org/

libro: http://git-scm.com/book/en/v2

Chuleta http://www.ndpsoftware.com/git-cheatsheet.html

vocabulario

repository

fork

clone

version

branch

commit

antes de empezar

Le decimos a Git quien somos y git confía

```
$ git config --global user.name "Your Name"
$ git config --global user.email "your@mail.com"
```

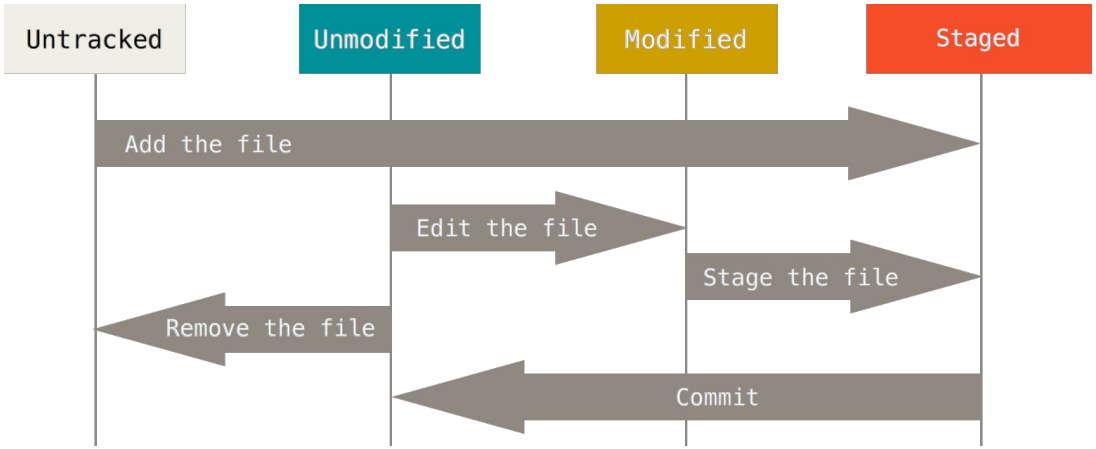
repositorio existente

```
$ git clone git@github.com:usuario/nombre_repo.git
$ cd nombre_repo
$ git status
...
```

repositorio nuevo

```
$ mkdir shiny
$ cd shiny
$ git init .
Initialized empty Git repository in /tmp/shiny/.git/
$ echo "Shiny" >> README.txt
$ git add README.txt
$ git commit -m 'Initial commit'
```

ciclo de vida



stash

workspace

index

local repository

upstream

workspace

como está la carpeta

respecto al estado conocido por git

.gitignore (excepciones)

index

cambios preparados

nuevo archivo

modificación

eliminación

borrado

local repository

ya en la BD de git

sólo en mi maquina

copia entera de la historia

`git fetch`

upstream repository

BD de git (de referencia)

en otra maquina (servidor)

por defecto: origin

```
git remote -v
```

comandos

status

add/rm

commit/reset

push/pull

merge

conflicto

dos ramas

ancestro común

cambios contradictorios

git no sabra como mezclar

conflicto cont.

status muestra ficheros implicados

marcas <<<<< ====== >>>>>

consola vs herramientas

commit 'vacío' para acabar

4 - Anatomía de Rails

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 ?

Prerequisitos: curl, bash, build-essential...

Ruby: RVM https://rvm.io/

.ruby-gemset y .ruby-verison

Bundler: http://gembundler.com/

¿Tenemos todo?

¿Por qué ir (o no) en tren?

¿... usar frameworks ?

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.


THAT MOMENT WHEN...

it is **hard**

http://goo.gl/LDJjE
(2012-05-08: Kakubei «Why I hate Rails»)

But gets throught!

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 use --create --ruby-version ruby-2.2@sinatra
$ bundle
Fetching gem metadata from https://rubygems.org/.........
$ ruby todo.rb
$ browser localhost:4567
# cambiar cosas
```

la Frase

«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 prinicipio 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

Instalación

¿Que es esto de las gemas?

Librerías (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
```

pero...

Muchos proyectos, y gemas

Dependencias

Conflicto!

RVM Gemsets

Bundler & Gemfile

Gemset

una opcion de *RVM*

es un sandbox (entorno aislado)

http://rvm.io/gemsets

Ejercicio Gemsets

Abrir la documentacion de RVM

Crear dos gemsets foo & bar

Cambiar manualmente entre uno y otro

Crear fichero '.ruby-version'

Cambiar automáticamente de gemset

Bundler

un gestor de dependencias

se configura con Gemfile

se puede usar dentro de un gemset

http://bundler.io/

Ejercicio de bundler

bueno, esperad...

Comienzo

porque rails usa bundler

```
$ gem install rails -v 4.2.1
...
$ rails -v
Rails 4.2.1

$ rails new --help
$ rails new enlazeitor
...
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled
gem is installed.
```

ayudas: "desarrollando" en Rails

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

ayudas: "desarrollando" en Rails

```
$ rails generate
$ rails g controller
$ rails g controller Pages home about contact
$ rake routes
$ edit config/routes.rb # cambiad root a pages#home
$ rails s
$ browser http://localhost:3000
```

ejercicio

vamos a hacernos una 'web'

muchas apps la tienen

¿que tenemos? (carpetas)

app (MVC)

bin (script loaders)

config (entornos, initializers, locales)

db (migrate)

lib (assets, tasks)

log (logs por entornos)

public (estáticos)

test (TDD)

tmp (sesiones, caché, uploads)

vendor (assets ajenos)

¿que tenemos? (gemas)

rails

sqlite3 https://www.sqlite.org/

sass-rails http://sass-lang.com

uglifier http://lisperator.net/uglifyjs/

coffee-rails http://coffeescript.org/

jquery-rails http://jquery.com/

turbolinks https://github.com/rails/turbolinks

jbuilder https://github.com/rails/jbuilder

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)

notas

DOC

Instalar : http://rubyonrails.org.es/instala.html

API : http://api.rubyonrails.org/

Ruby : http://ruby-doc.org/

extra : http://apidock.com/rails

ayudas:

IDE's/GUI's

RubyMine

Aptana

NetBeans

ayudas:

Editores

Vim-Rails

Emacs Rinari

ayudas:

Miscelanea

Rails command_line

Sqlite Browser

5 - Datos, Modelos y Almacenamiento

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 CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :title
      t.string :url
      t.string :description

      t.timestamps
    end
  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).
...
rake db:setup # Create the database, load the schema, and initialize with the
seed data...
...
```

Migraciones: peligros

```ruby
# db/migrate/20100513121110_add_active_to_post.rb
class AddActiveToPost < ActiveRecord::Migration
  def change
    add_column :posts, :active, :boolean
    Post.all.each{ |c| c.update_attributes!(:active => true) }
  end
end
# app/model/post.rb
class Post < ActiveRecord::Base
  validates :active, :presence => true
end
```

Migraciones: peligros

```ruby
# db/migrate/20100515121110_add_loc_to_post.rb
class AddLocationToPost < ActiveRecord::Migration
  def change
    add_column :posts, :location, :string
    Post.reset_column_information
    Post.all.{ |c| post.update_attribute(:location, 'Madrid')}
    # Post.update_all no valida y podria dejar datos no validos!
  end
end
# app/model/post.rb
class Post < ActiveRecord::Base
  validates :active, :presence => true
  validates :location, :presence => true
end
```

Migraciones: peligros

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

¿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 `Posts` con `title, url, description, timestamps`

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

añade `published` (bool), ajustado a true por defecto

Migra de nuevo

Migraciones: ejercicio

crea `Posts` con `title, url, description, timestamps`

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

añade `published` (bool), ajustado a true por defecto

Migra de nuevo

```
$ rails g migration CreatePost title:string url:string description:string
timestamps
...

$ rake db:migrate # ejecutar la migracion creada
$ sqlitebrowser db/development.sqlite3 # examiniar la tabla en la BD
$ rake db:rollback # deshacer la modificación de BD
$ gedit db/migrate/20150716133017_create_posts.rb # editar y corregir
$ rake db:migrate # re-ejecutar la migracion corregida
```

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'

```
>> post.valid?
>> post.invalid?
provocan validaciones
rellenan model.errors
devuelven true/false

>> post.errors
=> {:field => ["can't be something"], ...}

>> Post.new.errors[:name].any?
=> false
>> Post.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)

¿Cuando? ¿Ejemplos?

associated

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

validates :post_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-2.0.0/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/resource de `Post`

Validar que haya titulo

Arrancar rails mostrando los posts

Dar un entrada de alta (sin nombre)

Modificarlo, Borrarlo

¿Preguntas?

ayudas

```
$ rails g scaffold Post title:string ... -s
invoke active_record
skip db/migrate/20150717113321_create_posts.rb
skip app/models/post.rb
$ rails s ... Ctrl-C
$ rm public/index.html
$ edit config/routes.rb
# root :to => 'posts#index'
$ rails s
$ browser localhost:3000
```

6 - Testing de modelos controladores y aceptación

¿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
models, controllers, 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]


post = Post.new
assert post.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 = Post.last, Post.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 de modelo

Escribir tests/migraciones para Post:

Título no puede estar vacío

Publicado no puede ser nulo

Ejercicio: ayudas

```ruby
# test/model/post_test.rb
require 'test_helper'
class PostTest < ActiveSupport::TestCase
 test "a post without title" do
   @post = Post.new
   assert !@post.valid?, "can't be valid"
   assert_equal false, @post.save, "must not save"
   assert_raise(ActiveRecord::RecordInvalid){ @post.save! }
 end
end
# En la consola:
# ruby -Itest test/models/post_test.rb
# rake test:models
```

Fixtures

Valores que se cargan en la BD de pruebas

Representan un 'estado inicial'

Pueden usarse juntas o separadas

Escalan regular 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 cosas 'web'

se usan en tests de controladores e integración

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

assert_difference(expressions, difference = 1, message = nil) {...}
assert_no_difference(expressions, message = nil, &block)

assert_difference 'Post.count' do
  post :create, :post => {...}
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

¿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

```ruby
# Rails guarda en params los parametros de la peticion HTTP

Post.where("title = #{params[:title]}") # NO! SQLi

Post.where("title = ?", params[:title])

Post.where(" title = ? AND created_on > ?", [loc, date] )

Post.where("created_on <= :date", {:date => Date.today})
```

where + hash

Válido para algunas consultas

```
Post.where(published: true) # ... WHERE active = 1


# yday = (Time.now.midnight - 1.day)..Time.now.midnight
Post.where(created_at: yday)
# ... created_at BETWEEN '2013-05-06' 00:00:00
# AND '2013-05-07' 00:00:00


Post.where(title: ['Prueba','Prueba2'])
```

order

Especifica el orden, pueden ser varios, cuidado con ambigüedad

```
Post.order(:created_at)

Post.order("created_at")

Post.order("created_at DESC")

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

select

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

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

Ejercicio

"filtros" en index (o búsqueda)

posts#index muestra solo publicados

entradas salen en orden anticronológico

completar formularios

ver consultas en log

ver dev-tools/firebug

railspanel...

el resto?

Hay más...

7 - Relaciones e interacciones

MOAR MODELS!

distintas entidades y sus relaciones

y magia!

en realidad es OO

objetos como atributos de otros

```
@post.comments
```

O_o

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 Comment < ActiveRecord::Base
  belongs_to :post
end
```

el modelo que pertenece depende del otro

almacena su id (hay que migrar)

belongs_to

```
class Comment < ActiveRecord::Base
  belongs_to :post, :dependent => :destroy
end
```

... y puede destruirse automaticamente

belongs_to nos da

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

¿Como se llaman las de `Comment` ?

belongs_to nos da

```
@comment.post(force_reload = false)
@comment.post=(associate)
@comment.build_post(attributes = {})
@comment.create_post(attributes = {})
@comment.create_post!(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: http://goo.gl/TCeI9

¿estaban ya en favoritos?

http://ruby-doc.com/

http://api.rubyonrails.org/

http://apidock.com/

propina: http://devdocs.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 borrame?

`: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 Post < ActiveRecord::Base
  has_many :comments, :inverse_of => :post
end
```

has_many

```
collection(force_reload = false)
collection<<(object, …)
collection.delete(object, …)
collection=objects
collection_singular_ids
collection_singular_ids=ids
collection.clear
...
```

has_many

```
....
collection.empty?
collection.size
collection.find(…)
collection.where(…)
collection.exists?(…)
collection.build(attributes = {}, …)
collection.create(attributes = {})
```

has_many

```
@post.comments          # [<Comment>, <Comment>]
@post.comments << Comment.new( … )
@post.comments.delete( comment_for_delete_id )
@post.comments += my_new_comment
@post.comments.clear    # borra todos
@post.comments.empty?   # true | false
@post.comments.size     # 42
@post.comments.find(3)  # <Comment id:3, text: … >
...
```

has_many

```
...
@post.comments.where( "created_at > :date", date: 7.days.ago)
@post.comments.exists?(…)
@post.comments.build( text: 'Yeah, good post!' )
@post.comments.create( text: 'commenting is overrated')
```

¿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

una entrada puede tener varios comentarios

vamos a crear `Comment`

Ejercicio

una entrada tiene varios comentarios

Un comentario pertenece a una entrada

Actualizamos *updated_at* de la entrada

«Actividad reciente»

tiene `votos` ¿entero?

Ejercicio:notas

vamos a mezclar cosas aun no vistas...

...para que el ejemplo se más real

¿Formularios?

Veamos algo de vistas/rutas/controlador

rutas normales

HTTP Verb	Path	action	used for
GET	/comments	index	display a list of all comments
GET	/comments/new	new	return an HTML form for creating a new comment
POST	/comments	create	create a new comment
GET	/comments/:id	show	display a specific comment
GET	/comments/:id/edit	edit	return an HTML form for editing an comment
PUT	/comments/:id	update	update a specific comment
DELETE	/comments/:id	destroy	delete a specific comment

rutas anidadas

HTTP Verb	Path	action	used for
GET	/posts/:post_id/comments	index	display a list of all comments for a specific post
GET	/posts/:post_id/comments/new	new	return an HTML form for creating a new comment belonging to a specific post
POST	/posts/:post_id/comments	create	create a new comment belonging to a specific post
GET	/posts/:post_id/comments/:id	show	display a specific comment belonging to a specific post
GET	/posts/:post_id/comments/:id/edit	edit	return an HTML form for editing an comment belonging to a specific post
PUT	/posts/:post_id/comments/:id	update	update a specific comment belonging to a specific post
DELETE	/posts/:post_id/comments/:id	destroy	delete a specific comment belonging to a specific post

¿Cómo lo hacemos?

¿todos juntos o por separado?

no partimos del scaffold

sin formulario anidado (para simplificar)

sin gemas de formularios (por ahora)

Ejercicio ayudas

```
$ rails g model Comment body:text votes:integer post:references
invoke active_record
create db/migrate/20130503213742_create_comments.rb
create app/models/comment.rb
invoke test_unit
...
```

y después

relaciones en los modelos

rutas anidadas

relaciones en los modelos

como hemos visto en el tema

las hacemos inversas

modelo Comment

```ruby
# app/models/comment.rb
class Comment < ActiveRecord::Base
  # attr_accesible vs strong_params!
  belongs_to :post, inverse_of: :comments
end
```

rutas anidadas

las probamos con `rake routes`

rutas

```
# config/routes.rb
resources :posts do
  resources :comments # y los votes?
end
```

visita rapida al mundo de los controladores

`get_post` en `CommentsController`

variables de instancia en controlador

helpers de ruta y redirecciones

testeos y validaciones (ver scaffold de post)

controlador CommentsController

```ruby
# app/controllers/comments_controller.rb
class CommentsController < ApplicationController
  before_action :get_post

private
  def get_post
    if params[:post_id]
      @post = Post.find(params[:post_id])
    end
  end
end
```

ya que estamos

Evolución de datos

planificación

integridad referencial

`get_post` en `CommentsController`

un método privado en el controlador

si es privado no es una acción

¿es DRY? hablemos de CanCanCan

variables de instancia en controlador

cargar a partir de `@post`

✓ `@post.comments.find(params[:id])`

✗ ~~`Comment.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

plugins para molar extra: SimpleForm

formularios (REST) aprovechar la convención:

```
<%= form_for @post do |f| %>
 <%= f.label :title %>: <%= f.text_field :title %>
 <%= f.label :url %>: <%= 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`

En casa: testeos y validaciones

¿Os atreveis a testear el controlador?

usando los asserts vistos

encontrad algo que podría hacer y aun no hace

kaminari ?

8 - Buscar y encontrar

veremos:

cosas de la BD

otras cosas que se guardan

más alla de `first`, `last` y `all`

joins

scopes, condition override

dinamic_finders

calculations

```ruby
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
```

joins automágicos

```
>> 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

```
# Queries complejas o que se repiten:

scope :published, where(published: true)

scope :last_week, lambda { # callback ejecuta después
  where("created_at > ?", 1.week.ago )
}

scope :last_week, -> { # lo mismo que lambda
  where("created_at > ?", 1.week.ago )
}
```

scopes con argumento

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

Ejercicio 1

un scope que acepte argumentos o no

no nil sino ser llamado sin argumentos

Post.recent (por defecto una semana)

Post.recent(1.day.ago)

¿podeis encontrarlo?

salida deseada

```
>> Post.recent(1.day.ago)
post Load (0.5ms) SELECT "posts".* FROM "posts" WHERE (updated_at < '2015-06-
12 14:53:44.752681')
=> [#<post id: 1...]
>> Post.recent
post Load (0.5ms) SELECT "posts".* FROM "posts" WHERE (updated_at < '2015-06-
07 14:53:47.827892')
=> []
```

solución

```
>> class Post < ActiveRecord::Base
>> scope :recent, lambda {|*args|
>> where( "updated_at < ?",
>> (args && args.first ? args.first : 1.week.ago))}
>&;t; end
```

Ejercicio 2

revisar el código de la aplicación

pensar puntos de aprovechamiento

agregar un `includes` y un `scope`

puede ser un scope que hace includes

¿se testea?, ¿a que nivel?

ayudas

en la app de ejemplo:

`link_to_unless`

Posts filtrados con scope

params no REST (como el n para limit)

Dinamic finders

dinamic finders (Rails 3)

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()

https://github.com/rails/activerecord-deprecated_finders

new dinamic finders (Rails 4)

find_all_by_... ⟶ where(...)

find_last_by_... ⟶ where(...).last

scoped_by_... ⟶ where(...)

find_or_initialize_by_... ⟶ find_or_initialize_by(...)

find_or_create_by_... ⟶ find_or_create_by(...)

find_or_create_by_...! ⟶ find_or_create_by!(...)

pluck

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

Model.where(:active => true).pluck(:id)
# SELECT id FROM models WHERE active = 1

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

Calculations

contar

```ruby
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

Extra Credit:

¿os atreveis a agregar un buscador?

en "/posts?q=cosa_buscada"

por ahora sin AJAX

busca todos los campos de texto

resaltar resultados? http://goo.gl/bBtCfG

9 - Vistas, interfaz e interacción

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.html.erb`

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

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

Helper: repetición interna (código) `link_to`

LAYOUT

Se busca implicitamente en: **app/views/layouts/application.html.erb**
Uso: **layout :foobar** en el controlador o en un render
busca en **app/views/layouts/foobar.html.erb**

VISTA

Se busca (implícitamente) en: **app/views/***controlador***/***accion***.html.erb**
Uso: **render :foobaz** en una acción busca en:
app/views/*controlador***/foobaz.html.erb** y en **app/views/application/foobaz.html.erb**

HELPER

Uso: **<%= nombre_metodo(opciones) %>**
Se busca en: **app/helpers/application_helper.rb**
app/helpers/*controlador***_helper.rb**
en Rails (ver API)

_PARTIAL

Uso: **render @cosa** (clase Cosa y partial '_cosa')
render partial: 'foo', object: @foo
render partial: 'foo', collection: @foos
render partial: 'foo', locals: { foo: @foo, bar: @bar }
Se busca como las vistas (con prefijo '_')
Variable local con el nombre de partial (sin '_') y valor del object/collection

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}.{ext}.{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://guides.rubyonrails.org/layouts_and_rendering.html

Partial

en `app/views/{controller}/_{nombre}.{ext}.{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 http://haml.info/

Sass/Scss 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

`/posts/1` **no** es @post

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, decubridlos 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_form">
  <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

Ejercicio:

añadir contenido en layout (barra navegación)

acciones del post en posts_helper.rb

errores de campo a applicaction_helper.rb

comentarios de app/views/comments/_comment en post/show

Cache

¿caché o qué?

antes: http://guides.rubyonrails.org/caching_with_rails.html

pero en caché, Rails 4 lo peta

Resúmen: http://goo.gl/CRXub

Largo: http://goo.gl/T0mBYV

Buscad a David: http://youtu.be/yhseQP52yIY?t=39m47s

cache ANTES: tenía esta pinta

```ruby
# en el controlador
def accion_que_modifica_datos_cacheados
  List.update(params[:list][:id], params[:list])
  # la clave de la cache es siempre la misma
  expire_page action: 'show', id: params[:list][:id]
  redirect_to action: 'show', id: params[:list][:id]
end

# en la vista
<% cache( ...cosas malas a mano...) do %>
  (datos que se cachean)
```

CACHÉ AHORA:

el contenido de una clave nunca cambia

una actualización genera una clave nueva

AR lo hace sólo en el método cache_key

Memcache descarta lo mas viejo primero

CACHÉ AHORA: tiene esta pinta

```
# en la vista
<% cache @user do %>
(user view data)

<% cache [ 'details', @user ] do %>
(user details view data)
```

bonus

Señor que hizo esto con cronómetro http://www.appneta.com/blog/russian-doll-caching/

Ejercicio

separad el texto del post a un partial

excluid los comentarios

¿os atreveis a cacheadlo?

aprovechad: http://timeago.yarp.com/

10 - Seguridad y control de acceso

la Web - entorno abierto

hay que desconfiar :'(

sanitizar entradas...

... y salidas (`h()` y `raw()`)

y restringir

Autentificación:

hubo una vez que se hacía a mano

30-8-2006: Dave Astels vs Rails Recipes

mucho ha llovido, ahora tenemos Devise peeero

Ctrl-F -> "Starting with Rails?"

¿entonces? por ejemplo Authlogic

Ejercicio

Pistas

proteged acciones delicadas con el before_action `:required_login`

filtrad los recursos de usuario con *current_user.posts*

¿y si hay distintos tipos de usuarios?

¿y que más?

recuperar contraseña por mail

activar, bloquear usuarios

limitar intentos de acceso

recordar al usuario

Devise lo automatiza más pero...

Autorización/permisos/roles

hay conocidos y conocidos...

el cartero puede repartir tu correo...

...pero no leerlo

¿Quien puede hacer qué? (pensando en REST)

CanCanCan

Compatible con Devise, Sorcery y Authlogic
convencion: *current_user*

...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
```

11 - Extras

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 dificil 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

```yaml
# config/locales/es.yml
es:
  courses:
    index:
      title: "Título"
```

```erb
# app/views/courses/index.html.erb
<%= t '.title' %>
```

Ejemplo - I18n

```yaml
# 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, SimpleForm...

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 Best Practices

para petarlo

Metaprogramming Ruby de Paolo Perrotta

Blog de Aaron Patterson

Practical Object-Oriented Design in Ruby

Introducción a Ruby on Rails

Mayo 2015

Fernando Martínez

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 ?

Prerequisitos: curl, bash, build-essential...

Ruby: RVM https://rvm.io/

.ruby-gemset y .ruby-verison

Bundler: http://gembundler.com/

¿Tenemos todo?

Ruby para llevar

Hablando con Ruby (por telegrama)

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

Hablando con Ruby (por sms)

```
$ 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
```

String - comillas

```
>> name = "Fer"
=> "Fer"
>> "my name is #{name}"
=> "my name is Fer"
>> 'my name is #{name}'
=> "my name is #{name}"
```

String - interpolación

```
>> "my name is #{nome}"
=> "my name is "
>> "my name is " + nome
=> KABOOM!
```

String - métodos

- length
- reverse
- capitalize
- upcase
- downcase
- next ?

Ejercicio: progarma

Métodos

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

# se llaman asi:
nombre('p1', 'p2')
nombre('p1')
nombre 'p1'
nombre 'p1', 'p2' # cuidado!
```

Ejercicio: Métodos (irb)

Definid un metodo y ejecutadlo

Solución: Métodos (irb)

```
>> def h
>> puts "Hello World!"
>> end
=> :h # devuelve el nombre del método
>> h()
Hello World!
=> nil
>> h
Hello World!
=> nil
```

Ejercicio: Parámetros (irb)

Definid un método con parámetro

Solución: Parámetros

```
>> def greet(name)
>> puts "Hello #{name.capitalize}!"
>> end
=> :greet
>> greet "chris"
Hello Chris!
=> nil
>> greet
ArgumentError: wrong number of arguments (0 for 1)
...
```

Ejercicio: Parámetros 2 (irb)

Definid un método con parémetro (opcional)

Solución: Parámetros 2

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

Ejercicio: Nuestro primer archivo ruby

```ruby
# escribid esto en 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

```
# desde la carpeta en que está greeter.rb
$ 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

```ruby
if cond
  # do stuff
elsif other_cond
  # do other stuff
else
  # do default stuff
end

action if condition

action unless condition
```

Decisiones / Condicionales (cont)

```
foo = condition ? value_if_true : other_value


condition && value_if_true || other_value
# vs
condition and action_if_true or other_action
```

precedencia de operadores

```
valor = a_condition and another
(valor = a_condition) and another

valor = a_condition && another
valor = (a_condition && another)

# . > < == && || .. ... (?:) = not and or if unless while until
```

Repetición / bucles

```
while conditional [do]
   code
end

code while condition

until conditional [do]
   code
end

code until conditional
```

Más Repeticiones

```ruby
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-2.2.2/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 :)
```

Ejercicio: mini servidor

```
carpeta = ARGV[0] || Dir.pwd
puerto = ARGV[1] || 3000

# configurar el servidor
servidor = WEBrick::HTTPServer.new(:Port => puerto, :DocumentRoot => carpeta )

trap('INT'){ servidor.shutdown } # parar en 'Control-C'
servidor.start

ruby -rwebrick mini_servidor.rb . 3000
```

Frameworks:

Lo anterior es un juguete

Un framwework te evita las tareas repetitivas

Ejemplo *mínimo*

Añadir http://www.sinatrarb.com/(Sinatra)

Queda como esto: (demo server)

¿Por qué ir (o no) en tren?

¿... usar frameworks ?

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.


THAT MOMENT WHEN...

it is **hard**

http://goo.gl/LDJjE
(2012-05-08: Kakubei «Why I hate Rails»)

But gets throught!

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 use --create --ruby-version ruby-2.2@sinatra
$ bundle
Fetching gem metadata from https://rubygems.org/.........
$ ruby todo.rb
$ browser localhost:4567
# Jugad!
```

la Frase

«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 prinicipio 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-Junio 2015

Fernando Martínez

Parte 2

Despegando

Instalar

Comenzar una aplicación

Desplegar

Instalación

¿Que es esto de las gemas?

Librerías (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
```

pero...

Muchos proyectos, y gemas

Dependencias

Conflicto!

RVM Gemsets

Bundler & Gemfile

Gemset

una opcion de *RVM*

es un sandbox (entorno aislado)

http://rvm.io/gemsets

Ejercicio Gemsets

Abrir la documentacion de RVM

Crear dos gemsets foo & bar

Cambiar manualmente entre uno y otro

Crear fichero '.ruby-version'

Cambiar automáticamente de gemset

Bundler

un gestor de dependencias

se configura con Gemfile

se puede usar dentro de un gemset

http://bundler.io/

Ejercicio de bundler

bueno, esperad...

Comienzo

porque rails usa bundler

```
$ gem install rails -v 4.2.1
...
$ rails -v
Rails 4.2.1

$ rails new --help
$ rails new enlazeitor
...
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled
gem is installed.
```

ayudas: "desarrollando" en Rails

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

ayudas: "desarrollando" en Rails

```
$ rails generate
$ rails g controller
$ rails g controller Pages home about contact
$ rake routes
$ edit config/routes.rb # cambiad root a pages#home
$ rails s
$ browser http://localhost:3000
```

ejercicio

vamos a hacernos una 'web'

muchas apps la tienen

¿que tenemos? (carpetas)

app (MVC)

bin (script loaders)

config (entornos, initializers, locales)

db (migrate)

lib (assets, tasks)

log (logs por entornos)

public (estáticos)

test (TDD)

tmp (sesiones, caché, uploads)

vendor (assets ajenos)

¿que tenemos? (gemas)

rails

sqlite3 https://www.sqlite.org/

sass-rails http://sass-lang.com

uglifier http://lisperator.net/uglifyjs/

coffee-rails http://coffeescript.org/

jquery-rails http://jquery.com/

turbolinks https://github.com/rails/turbolinks

jbuilder https://github.com/rails/jbuilder

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)

notas

DOC

Instalar : http://rubyonrails.org.es/instala.html

API : http://api.rubyonrails.org/

Ruby : http://ruby-doc.org/

extra : http://apidock.com/rails

ayudas:

IDE's/GUI's

RubyMine

Aptana

NetBeans

ayudas:

Editores

Vim-Rails

Emacs Rinari

ayudas:

Miscelanea

Rails command_line

Sqlite Browser

Introducción a Ruby on Rails

3 - Las Maletas

Mayo-Junio 2015
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 CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :title
      t.string :url
      t.string :description

      t.timestamps
    end
  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).
...
rake db:setup # Create the database, load the schema, and initialize with the
seed data...
...
```

Migraciones: peligros

```ruby
# db/migrate/20100513121110_add_active_to_post.rb
class AddActiveToPost < ActiveRecord::Migration
  def change
    add_column :posts, :active, :boolean
    Post.all.each{ |c| c.update_attributes!(:active => true) }
  end
end
# app/model/post.rb
class Post < ActiveRecord::Base
  validates :active, :presence => true
end
```

Migraciones: peligros

```ruby
# db/migrate/20100515121110_add_loc_to_post.rb
class AddLocationToPost < ActiveRecord::Migration
  def change
    add_column :posts, :location, :string
    Post.reset_column_information
    Post.all.{ |c| post.update_attribute(:location, 'Madrid')}
    # Post.update_all no valida y podria dejar datos no validos!
  end
end
# app/model/post.rb
class Post < ActiveRecord::Base
  validates :active, :presence => true
  validates :location, :presence => true
end
```

Migraciones: peligros

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

¿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 `Posts` con `title, url, description, timestamps`

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

añade `published` (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'

```
>> post.valid?
>> post.invalid?
provocan validaciones
rellenan model.errors
devuelven true/false

>> post.errors
=> {:field => ["can't be something"], ...}

>> Post.new.errors[:name].any?
=> false
>> Post.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)

¿Cuando? ¿Ejemplos?

associated

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

validates :post_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-2.0.0/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/resource de `Post`

Validar que haya titulo

Arrancar rails mostrando los posts

Dar un entrada de alta (sin nombre)

Modificarlo, Borrarlo

¿Preguntas?

ayudas

```
$ rails g scaffold Post title:string ... -s
invoke active_record
skip db/migrate/20131102213321_create_posts.rb
skip app/models/post.rb
$ rails s ... Ctrl-C
$ rm public/index.html
$ edit config/routes.rb
# root :to => 'posts#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
models, controllers, 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]


post = Post.new
assert post.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 = Post.last, Post.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 de modelo

Escribir tests/migraciones para Post:

Título no puede estar vacío

Publicado no puede ser nulo

Ejercicio: ayudas

```ruby
# test/model/post_test.rb
require 'test_helper'
class PostTest < ActiveSupport::TestCase
 test "a post without title" do
   @post = Post.new
   assert !@post.valid?, "can't be valid"
   assert_equal false, @post.save, "must not save"
   assert_raise(ActiveRecord::RecordInvalid){ @post.save! }
 end
end
# En la consola:
# ruby -Itest test/models/post_test.rb
# rake test:models
```

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 cosas 'web'

se usan en tests de controladores e integración

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

assert_difference(expressions, difference = 1, message = nil) {...}
assert_no_difference(expressions, message = nil, &block)

assert_difference 'Post.count' do
  post :create, :post => {...}
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 dificil 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

```ruby
# Rails guarda en params los parametros de la peticion HTTP

Post.where("title = #{params[:title]}") # NO! SQLi

Post.where("title = ?", params[:title])

Post.where(" title = ? AND created_on > ?", [loc, date] )

Post.where("created_on <= :date", {:date => Date.today})
```

where + hash

Válido para algunas consultas

```
Post.where(published: true) # ... WHERE active = 1


# yday = (Time.now.midnight - 1.day)..Time.now.midnight
Post.where(created_at: yday)
# ... created_at BETWEEN '2013-05-06' 00:00:00
# AND '2013-05-07' 00:00:00


Post.where(title: ['Prueba','Prueba2'])
```

order

Especifica el orden, pueden ser varios, cuidado con ambigüedad

```
Post.order(:created_at)

Post.order("created_at")

Post.order("created_at DESC")

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

select

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

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

Ejercicio

"filtros" en index (o búsqueda)

posts#index muestra solo publicados

entradas salen en orden anticronológico

completar formularios

ver consultas en log

ver dev-tools/firebug

railspanel...

el resto?

Hay más...

Introducción a Ruby on Rails

4 - Conectando vuelos

Mayo-Junio 2015
Fernando Martínez

Parte 4

MOAR MODELS!

distintas entidades y sus relaciones

y magia!

en realidad es OO

objetos como atributos de otros

```
@post.comments
```

O_o

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 Comment < ActiveRecord::Base
  belongs_to :post
end
```

el modelo que pertenece depende del otro

almacena su id (hay que migrar)

belongs_to

```
class Comment < ActiveRecord::Base
  belongs_to :post, :dependent => :destroy
end
```

... y puede destruirse automaticamente

belongs_to nos da

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

¿Como se llaman las de `Comment` ?

belongs_to nos da

```
@comment.post(force_reload = false)
@comment.post=(associate)
@comment.build_post(attributes = {})
@comment.create_post(attributes = {})
@comment.create_post!(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: http://goo.gl/TCeI9

¿estaban ya en favoritos?

http://ruby-doc.com/

http://api.rubyonrails.org/

http://apidock.com/

propina: http://devdocs.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 borrame?

`: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 Post < ActiveRecord::Base
  has_many :comments, :inverse_of => :post
end
```

has_many

```
collection(force_reload = false)
collection<<(object, …)
collection.delete(object, …)
collection=objects
collection_singular_ids
collection_singular_ids=ids
collection.clear
...
```

has_many

```
....
collection.empty?
collection.size
collection.find(…)
collection.where(…)
collection.exists?(…)
collection.build(attributes = {}, …)
collection.create(attributes = {})
```

has_many

```
@post.comments          # [<Comment>, <Comment>]
@post.comments << Comment.new( … )
@post.comments.delete( comment_for_delete_id )
@post.comments += my_new_comment
@post.comments.clear    # borra todos
@post.comments.empty?   # true | false
@post.comments.size     # 42
@post.comments.find(3)  # <Comment id:3, text: … >
...
```

has_many

```
...
@post.comments.where( "created_at > :date", date: 7.days.ago)
@post.comments.exists?(…)
@post.comments.build( text: 'Yeah, good post!' )
@post.comments.create( text: 'commenting is overrated')
```

¿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

una entrada puede tener varios comentarios

vamos a crear `Comment`

Ejercicio

una entrada tiene varios comentarios

Un comentario pertenece a una entrada

Actualizamos *updated_at* de la entrada

«Actividad reciente»

tiene `votos` ¿entero?

Ejercicio:notas

vamos a mezclar cosas aun no vistas...

...para que el ejemplo se más real

¿Formularios?

Veamos algo de vistas/rutas/controlador

rutas normales

HTTP Verb	Path	action	used for
GET	/comments	index	display a list of all comments
GET	/comments/new	new	return an HTML form for creating a new comment
POST	/comments	create	create a new comment
GET	/comments/:id	show	display a specific comment
GET	/comments/:id/edit	edit	return an HTML form for editing an comment
PUT	/comments/:id	update	update a specific comment
DELETE	/comments/:id	destroy	delete a specific comment

rutas anidadas

HTTP Verb	Path	action	used for
GET	/posts/:post_id/comments	index	display a list of all comments for a specific post
GET	/posts/:post_id/comments/new	new	return an HTML form for creating a new comment belonging to a specific post
POST	/posts/:post_id/comments	create	create a new comment belonging to a specific post
GET	/posts/:post_id/comments/:id	show	display a specific comment belonging to a specific post
GET	/posts/:post_id/comments/:id/edit	edit	return an HTML form for editing an comment belonging to a specific post
PUT	/posts/:post_id/comments/:id	update	update a specific comment belonging to a specific post
DELETE	/posts/:post_id/comments/:id	destroy	delete a specific comment belonging to a specific post

¿Cómo lo hacemos?

¿todos juntos o por separado?

no partimos del scaffold

sin formulario anidado (para simplificar)

sin gemas de formularios (por ahora)

Ejercicio ayudas

```
$ rails g model Comment body:text votes:integer post:references
invoke active_record
create db/migrate/20130503213742_create_comments.rb
create app/models/comment.rb
invoke test_unit
...
```

y después

relaciones en los modelos

rutas anidadas

relaciones en los modelos

como hemos visto en el tema

las hacemos inversas

modelo Comment

```ruby
# app/models/comment.rb
class Comment < ActiveRecord::Base
  # attr_accesible vs strong_params!
  belongs_to :post, inverse_of: :comments
end
```

rutas anidadas

las probamos con `rake routes`

rutas

```
# config/routes.rb
resources :posts do
  resources :comments # y los votes?
end
```

visita rapida al mundo de los controladores

`get_post` en `CommentsController`

variables de instancia en controlador

helpers de ruta y redirecciones

testeos y validaciones (ver scaffold de post)

controlador CommentsController

```ruby
# app/controllers/comments_controller.rb
class CommentsController < ApplicationController
  before_action :get_post

private
  def get_post
    if params[:post_id]
      @post = Post.find(params[:post_id])
    end
  end
end
```

ya que estamos

Evolución de datos

planificación

integridad referencial

`get_post` en `CommentsController`

un método privado en el controlador

si es privado no es una acción

¿es DRY? hablemos de CanCanCan

variables de instancia en controlador

cargar a partir de `@post`

✓ `@post.comments.find(params[:id])`

✗ ~~`Comment.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

plugins para molar extra: SimpleForm

formularios (REST) aprovechar la convención:

```
<%= form_for @post do |f| %>
 <%= f.label :title %>: <%= f.text_field :title %>
 <%= f.label :url %>: <%= 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`

En casa: testeos y validaciones

¿Os atreveis a testear el controlador?

usando los asserts vistos

encontrad algo que podría hacer y aun no hace

kaminari ?

Introducción a Ruby on Rails

5 - La cinta de equipaje

Mayo-Junio 2015
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

```ruby
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
```

joins automágicos

```
>> 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

```
# Queries complejas o que se repiten:
scope :published, { published: true } # (Rails 3)

scope :published, where(published: true)
scope :last\_week, lambda{ # callback ejecuta después
  where("created\_at < ?", Time.zone.now ) }
scope :last\_week, ->{ # lo mismo que lambda
  where("created\_at < ?", Time.zone.now ) }
```

scopes con argumento

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

Ejercicio 1

un scope acepte que argumentos o no

no nil sino ser llamado sin argumentos

Post.recent (por defecto una semana)

Post.recent(1.day.ago)

¿podeis encontrarlo?

salida deseada

```
>> Post.recent(1.day.ago)
post Load (0.5ms) SELECT "posts".* FROM "posts" WHERE (updated_at < '2015-06-
12 14:53:44.752681')
=> [#<post id: 1...]
>> post.recent
post Load (0.5ms) SELECT "posts".* FROM "posts" WHERE (updated_at < '2015-06-
07 14:53:47.827892')
=> []
```

solución

```
>> class Post < ActiveRecord::Base
>> scope :recent, lambda {|*args|
>> where( "updated_at < ?",
>> (args && args.first ? args.first : 1.week.ago))}
>> end
```

Ejercicio 2

revisar el código de la aplicación

pensar puntos de aprovechamiento

agregar un `includes` y un `scope`

puede ser un scope que hace includes

¿se testea?, ¿a que nivel?

ayudas

en la app de ejemplo:

`link_to_unless`

Posts filtrados con scope

params no REST (como el n para limit)

Dinamic finders

dinamic finders (Rails 3)

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()

https://github.com/rails/activerecord-deprecated_finders

new dinamic finders (Rails 4)

find_all_by_... ⟶ where(...)

find_last_by_... ⟶ where(...).last

scoped_by_... ⟶ where(...)

find_or_initialize_by_... ⟶ find_or_initialize_by(...)

find_or_create_by_... ⟶ find_or_create_by(...)

find_or_create_by_...! ⟶ find_or_create_by!(...)

pluck

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

Model.where(:active => true).pluck(:id)
# SELECT id FROM models WHERE active = 1

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

Calculations

contar

```ruby
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

Extra Credit:

¿os atreveis a agregar un buscador?

en "/posts?q=cosa_buscada"

por ahora sin AJAX

busca todos los campos de texto

resaltar resultados? http://goo.gl/bBtCfG

Introducción a Ruby on Rails

6 - Fotos de Vacaciones

Mayo-Junio 2015
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.html.erb`

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

Partial: repetición interna (plantilla) `_form.html.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}.{ext}.{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}.{ext}.{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 http://haml.info/

Sass/Scss 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

`/posts/1` **no** es @post

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, decubridlos 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_form">
  <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

Ejercicio:

añadir contenido en layout (barra navegación)

acciones del post en posts_helper.rb

errores de campo a applicaction_helper.rb

comentarios de app/views/comments/_comment en post/show

Cache

¿caché o qué?

antes: http://guides.rubyonrails.org/caching_with_rails.html

pero en caché, Rails 4 lo peta

Resúmen: http://goo.gl/CRXub

Largo: http://goo.gl/T0mBYV

Buscad a David: http://youtu.be/yhseQP52yIY?t=39m47s

cache ANTES: tenía esta pinta

```
# en el controlador
def accion_que_modifica_datos_cacheados
  List.update(params[:list][:id], params[:list])
  # la clave de la cache es siempre la misma
  expire_page action: 'show', id: params[:list][:id]
  redirect_to action: 'show', id: params[:list][:id]
end

# en la vista
<% cache( ...cosas malas a mano...) do %>
  (datos que se cachean)
```

CACHÉ AHORA:

el contenido de una clave nunca cambia

una actualización genera una clave nueva

AR lo hace sólo en el método cache_key

Memcache descarta lo mas viejo primero

CACHÉ AHORA: tiene esta pinta

```
# en la vista
<% cache @user do %>
(user view data)

<% cache [ 'details', @user ] do %>
(user details view data)
```

bonus

Señor que hizo esto con cronómetro http://www.appneta.com/blog/russian-doll-caching/

Ejercicio

separad el texto del post a un partial

excluid los comentarios

¿os atreveis a cacheadlo?

aprovechad: http://timeago.yarp.com/

Introducción a Ruby on Rails

7 - Viajes más complicados

Mayo-Junio 2015
Fernando Martínez

la Web - entorno abierto

hay que desconfiar :'(

sanitizar entradas...

... y salidas (`h()` y `raw()`)

y restringir

Autentificación:

hubo una vez que se hacía a mano

30-8-2006: Dave Astels vs Rails Recipes

mucho ha llovido, ahora tenemos Devise peeero

Ctrl-F -> "Starting with Rails?"

¿entonces? por ejemplo Authlogic

Ejercicio

Pistas

proteged acciones delicadas con el before_action `:required_login`

filtrad los recursos de usuario con *current_user.posts*

¿y si hay distintos tipos de usuarios?

¿y que más?

recuperar contraseña por mail

activar, bloquear usuarios

limitar intentos de acceso

recordar al usuario

Devise lo automatiza más pero...

Autorización/permisos/roles

hay conocidos y conocidos...

el cartero puede repartir tu correo...

...pero no leerlo

¿Quien puede hacer qué? (pensando en REST)

CanCanCan

Compatible con Devise, Sorcery y Authlogic
convencion: *current_user*

...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
```

Introducción a Ruby on Rails

8 - Consejos de Viaje

Mayo-Junio 2015
Fernando Martínez

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 dificil 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

```yaml
# config/locales/es.yml
es:
  courses:
    index:
      title: "Título"
```

```erb
# app/views/courses/index.html.erb
<%= t '.title' %>
```

Ejemplo - I18n

```yaml
# 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, SimpleForm...

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 Best Practices

para petarlo

Metaprogramming Ruby de Paolo Perrotta

Blog de Aaron Patterson

Practical Object-Oriented Design in Ruby