



frontRails

(Rails para maquetadores)

María Martínez

<http://limalimon.com.es>

maria.martinez@the-cocktail.com

Fernando Blat

<http://www.inwebwetrust.net>

fernando.blat@the-cocktail.com

objetivos

- conceptos básicos necesarios para trabajar con RoR
- saber qué son los partials
- saber qué son los helpers
- entender cómo se integra javascript con el framework

conceptos básicos

MVC: Modelo – Vista – Controlador

M	→	datos
V	→	interfaz (web)
C	→	lógica de la aplicación

MVC: Modelo – Vista – Controlador

M	→	datos
V	→	interfaz (web)
C	→	lógica de la aplicación

conceptos básicos

vista

variables de instancia

ajax

script.aculo.us

helpers

partials

rutras / URLs

RJS

UJS

RHTML

convención sobre configuración

- un framework “con opinión”
- código ordenado y homogéneo
- DRY: don't repeat yourself

estructura de directorios

app / views

public /

stylesheets

javascripts

images

stylesheets

404.html

...

trabajando (lo justo) con el modelo

- tablas → clases
- filas → objetos
- columnas → atributos

`tabla: contacts`

`clase: Contact`

`objeto: @contact`

`@contact.name`

`@contact.friend.name`

`...`

trabajando con el controlador

variables de instancia

- controlador → carpeta con vistas
- método del controlador → vista asociada

controlador → `app/controllers/posts_controller.rb`

```
def view
  @post = Post.find(:first)
end
```

vistas → `app/views/posts/view.rhtml`

```
<h1><%= @post.title %></h1>
<%= simple_format(@post.body) %>
```

rutas

- `config/routes.rb`
- rutas personalizadas
- por defecto → controlador / acción / id

conceptos básicos

rutas

```
ApiController::Routing::Routes.draw do |map|

  map.connect '', :controller => "portadas", :action => 'index'

  map.connect 'login', :controller => 'account',
    :action => 'login'

  map.connect 'ficha/:nicename/:nicecategory',
    :controller => 'posts', :action => 'show'

  map.connect ':controller/:action/:id'

end
```

partials

partials

- fragmentos de código reutilizables
- DRY
- planificación para reutilizar al máximo el código

```
<%= render :partial => 'noticia' %>
```

helpers

helpers

Nos ayudan a escribir código html

```
<%= truncate mensaje.contenido, 9 %> → hola q..
```

```
<%= image_tag 'home.png' %>
```

```
<%= link_to 'Ver', :action => 'mostrar', :id =>
persona.id %>
```

```
<%= form_tag :action => 'create' %> →
<form action="create" method="post">
```

helpers sí, pero con precaución.

- las vistas deben ser html mayoritariamente
- sí, si realmente ayudan
- no, si suponen más problemas que ayuda

`<%= image_tag 'home.png' %>` (¿dónde pongo la clase?)

`<%= end_form_tag %>` (19 caracteres)

`</form>` (7 caracteres)

helpers

- redefine helpers de Rails
- crea tus propio helpers → DRY HTML

helpers

`application_helper.rb:`

```
def distance_of_time_in_words(from_time, to_time = 0,
  include_seconds = false)

  distance_in_minutes =
    ((to_time - from_time).abs)/60).round

  case distance_in_minutes
  when 0..1      then 'hace nada'
  when 2..45    then "#{distance_in_minutes} minutos"
  when 46..90   then 'hace una hora'
  when 90..1440 then "hace bastante rato"
  else          "hace mucho rato"
  end
end
```

ror y javascript

ror y javascript

- Ajax
- RJS y UJS

```
<%= javascript_include_tag :defaults %>
```

```
<%= link_to_remote "enlace", :update => "id", :url =>  
  {:action => "action", :controller => "controlador" } %>
```

ajax

script.aculo.us

```
<%= javascript_include_tag :defaults %>

<script src="/javascripts/prototype.js" type="text/javascript"></script>
<script src="/javascripts/effects.js" type="text/javascript"></script>
<script src="/javascripts/dragdrop.js" type="text/javascript"></script>
<script src="/javascripts/controls.js" type="text/javascript"></script>

<%= link_to_remote "enlace", :update => "id", :url =>
  {:action => "action", :controller => "controlador" } %>

<a href="#" onclick="new Ajax.Updater('id',
  'controlador/accion',
  {asynchronous:true, evalScripts:true}); return false;">
  enlace</a>
```

RJS: ¿qué es?

“JavaScript written in Ruby”

```
page[:mensaje].replace_html 'Hello World!`  
page.select('form').each { |f| f.reset }
```

RJS: ventajas

- no es necesario conocer la sintaxis de javascript
- actualización simultánea y asíncrona de varios elementos de una página

UJS: javascript no intrusivo

definir comportamientos usando selectores css

```
<% apply_behaviours {  
    on '#myform:submit', 'return validate(this)'  
    on 'a.widget:click', { |page| page.call 'Widget.activate'  
}  
%>
```

UJS: javascript no intrusivo

usar helpers y aplicar efectos de forma no intrusiva

```
<%= content_tag 'div', 'click me!',  
      :onclick => '$(this).visualEffect("highlight") ' %>
```

UJS: javascript no intrusivo

separar comportamiento / presentación / contenido



UJS: javascript no intrusivo

cachear comportamientos

```
cache_behaviour :index, :otro_mas
```

```
expire_behaviour :action => 'index'
```

resumen

resumen

Conceptos básicos necesarios para trabajar con RoR

- MCV, estructura directorios, rutas, controlador, etc.

Saber qué son los partials

- Ahorran tiempo
- DRY

Saber qué son los helpers

- ¿helpers? Unos si, otros no

Conocer la integración de javascript con el framework

- Integrado con el framework
- Ajax, RJS, UJS

más información

- <http://api.rubyonrails.com>
- <http://ujs4rails.com>
- <http://script.aculo.us>
- <http://oreilly.com/catalog/rjsrails/>

gracias



Salamanca 17

Madrid – 28020
España

tel. +34 91 567 0605

www.the-cocktail.com