

Alternativas a Rails para sitios y servicios web ultraligeros



Raúl Murciano
Conferencia Rails 2009

The background of the slide is white, decorated with several large, irregular red splatters of varying sizes and densities, resembling blood or paint. These splatters are scattered across the upper and middle portions of the slide, with some smaller droplets trailing off from the larger ones.

```
while conferencia_ror_09  
  self.hands.dirty!  
end
```


github.com/raul/conferencia_rails_2009

Quién soy

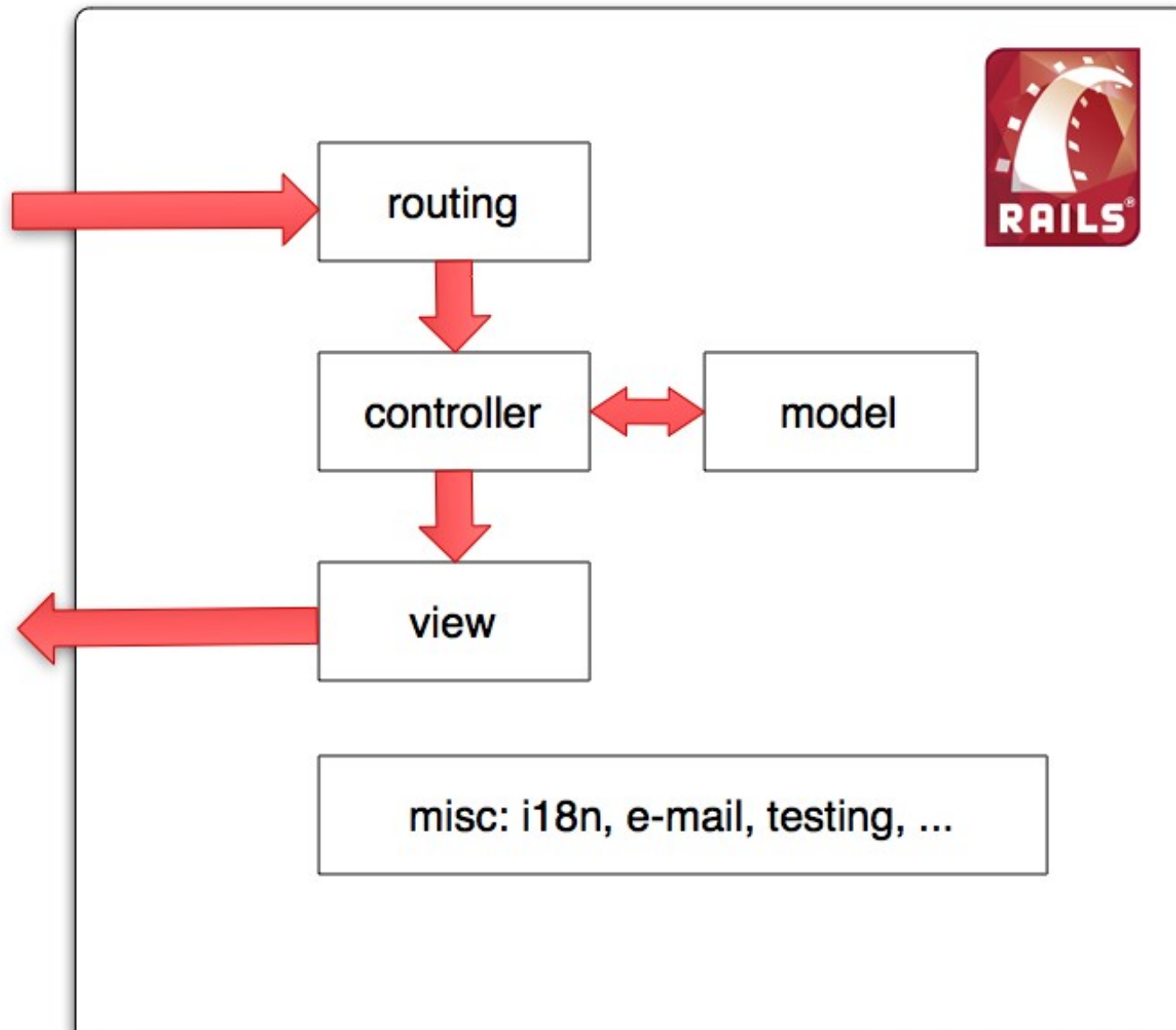
- Freelance: **raul.murciano.net**
- Consultoría/Formación: **PRORUBYTEAM.COM**
- linkedin.com/in/raulmurciano
- Comunidad: spainrb.org - srug.org
- github.com/raul
- twitter.com/happywebcoder

Background tecnología

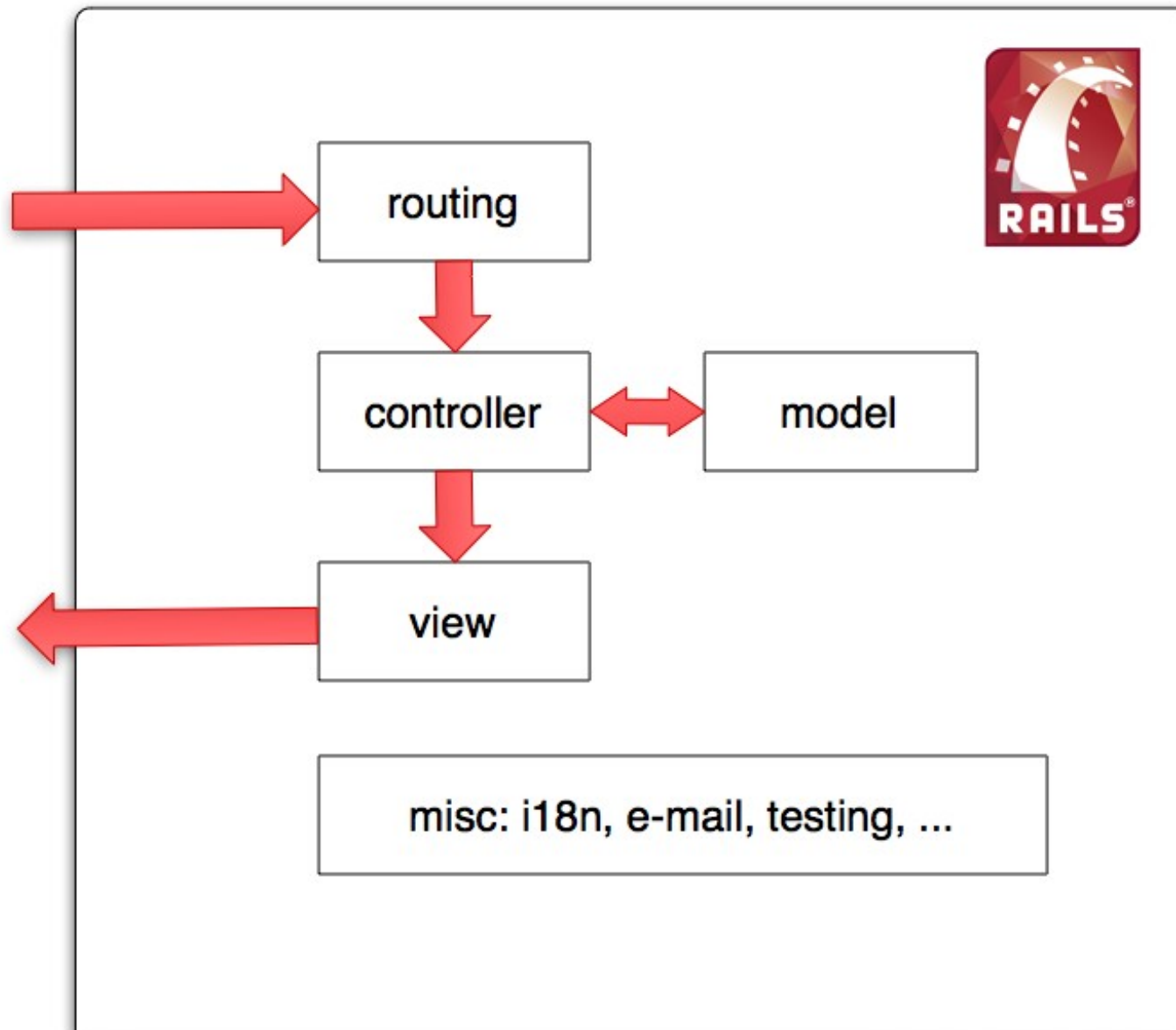
- Java → PHP → Ruby on Rails
- Huyo de la complejidad
- Me encanta el código expresivo y conciso

I  Rails

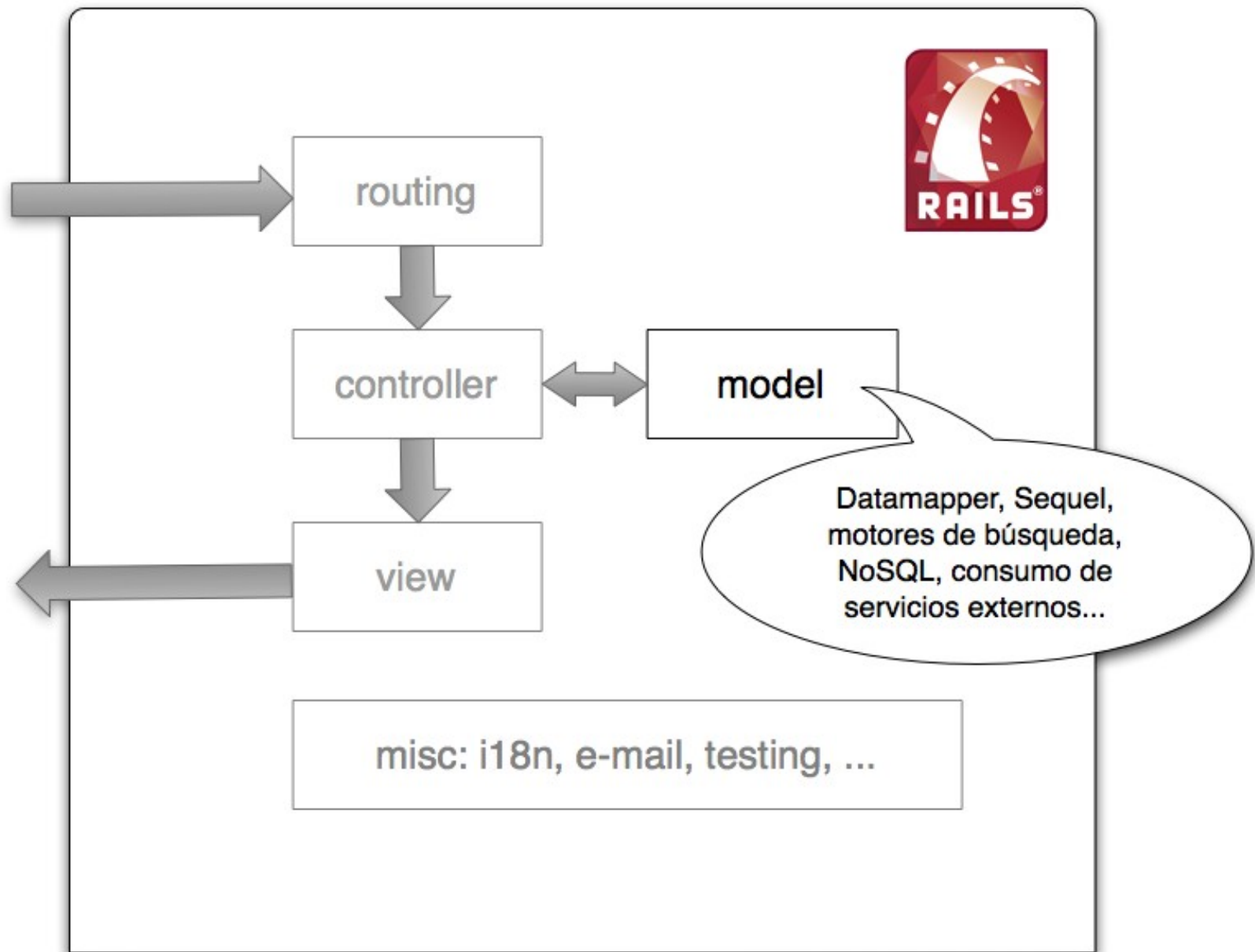
Rails aporta muchas cosas



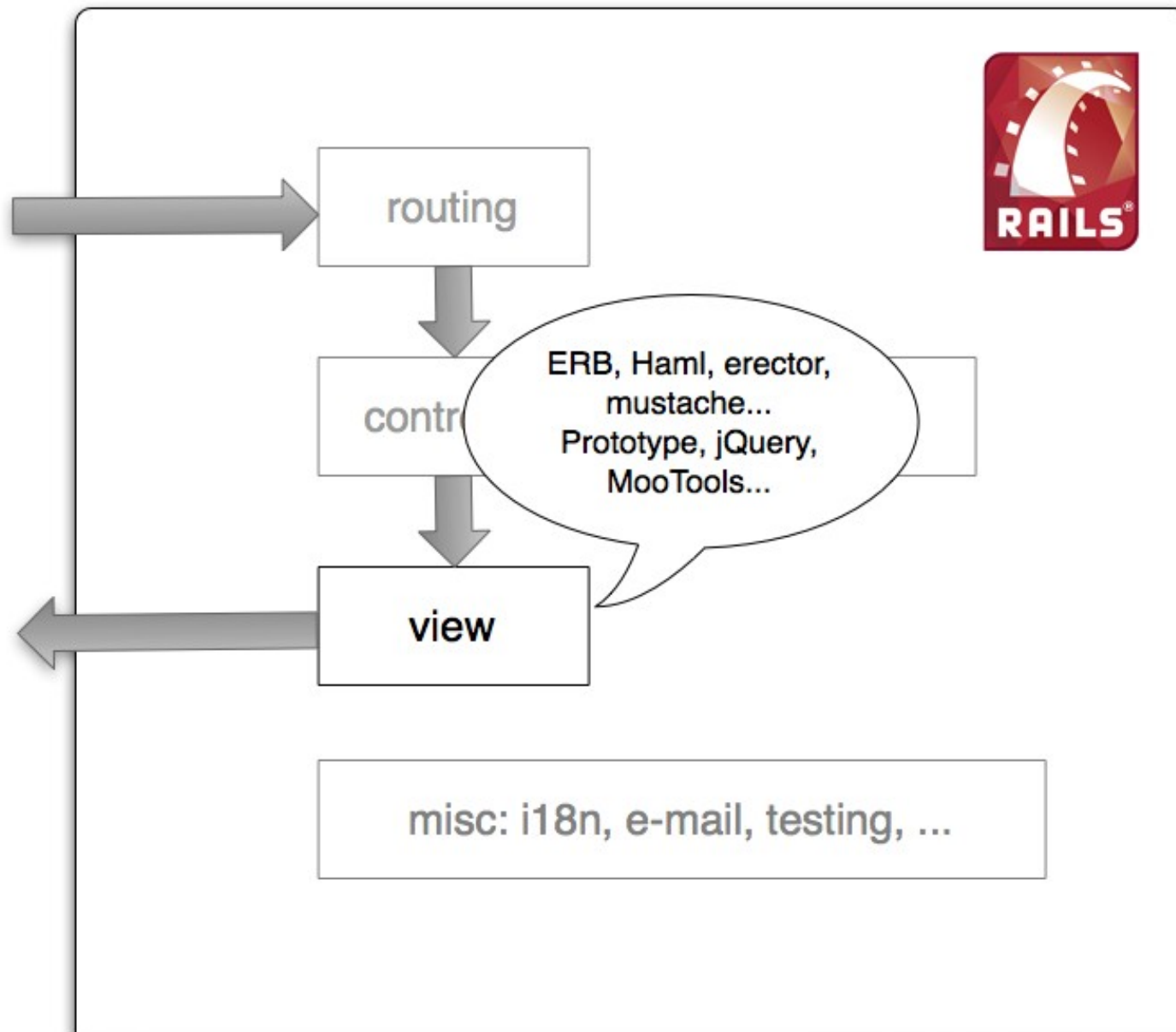
...pero no hay balas de plata



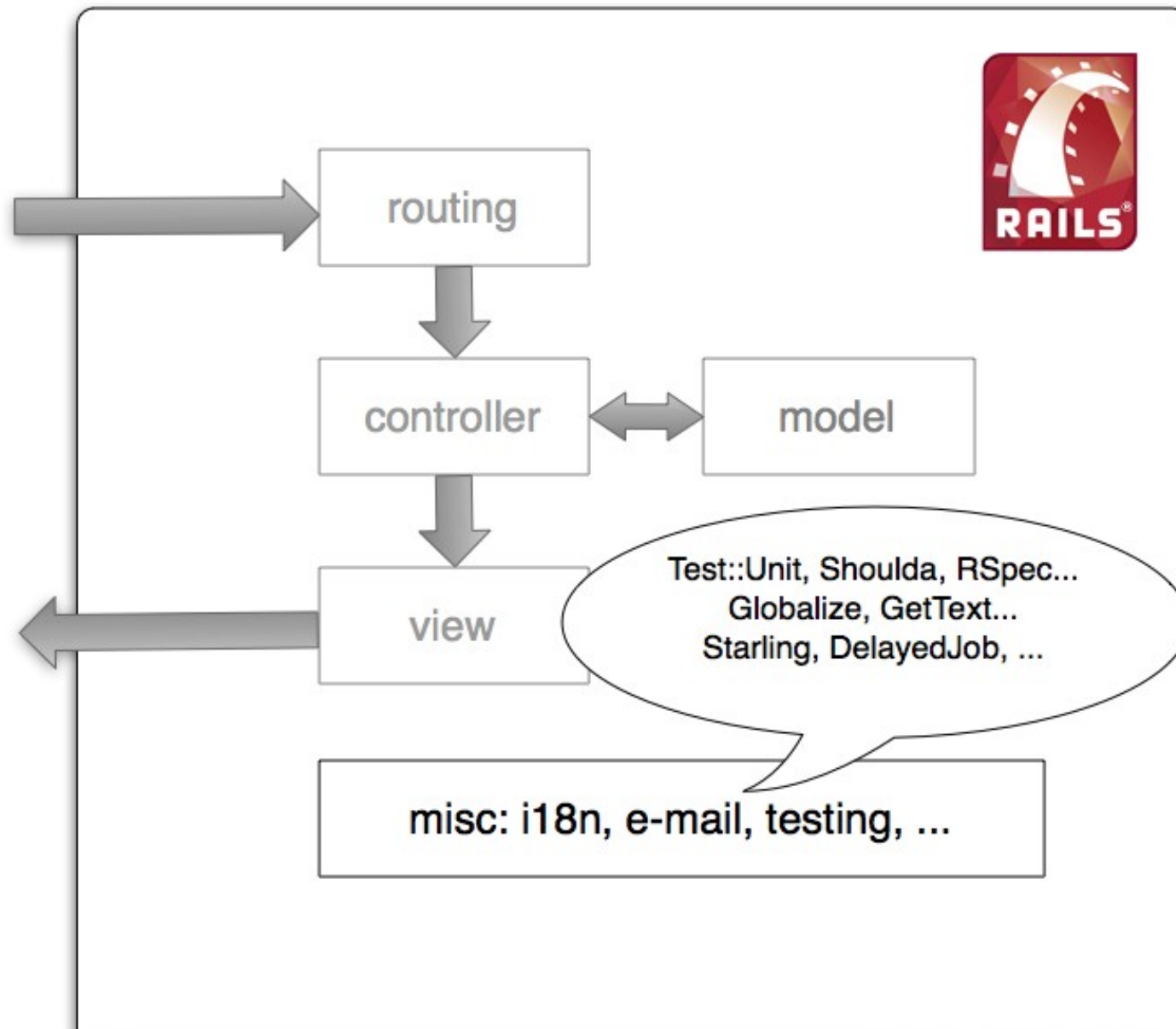
...pero no hay balas de plata



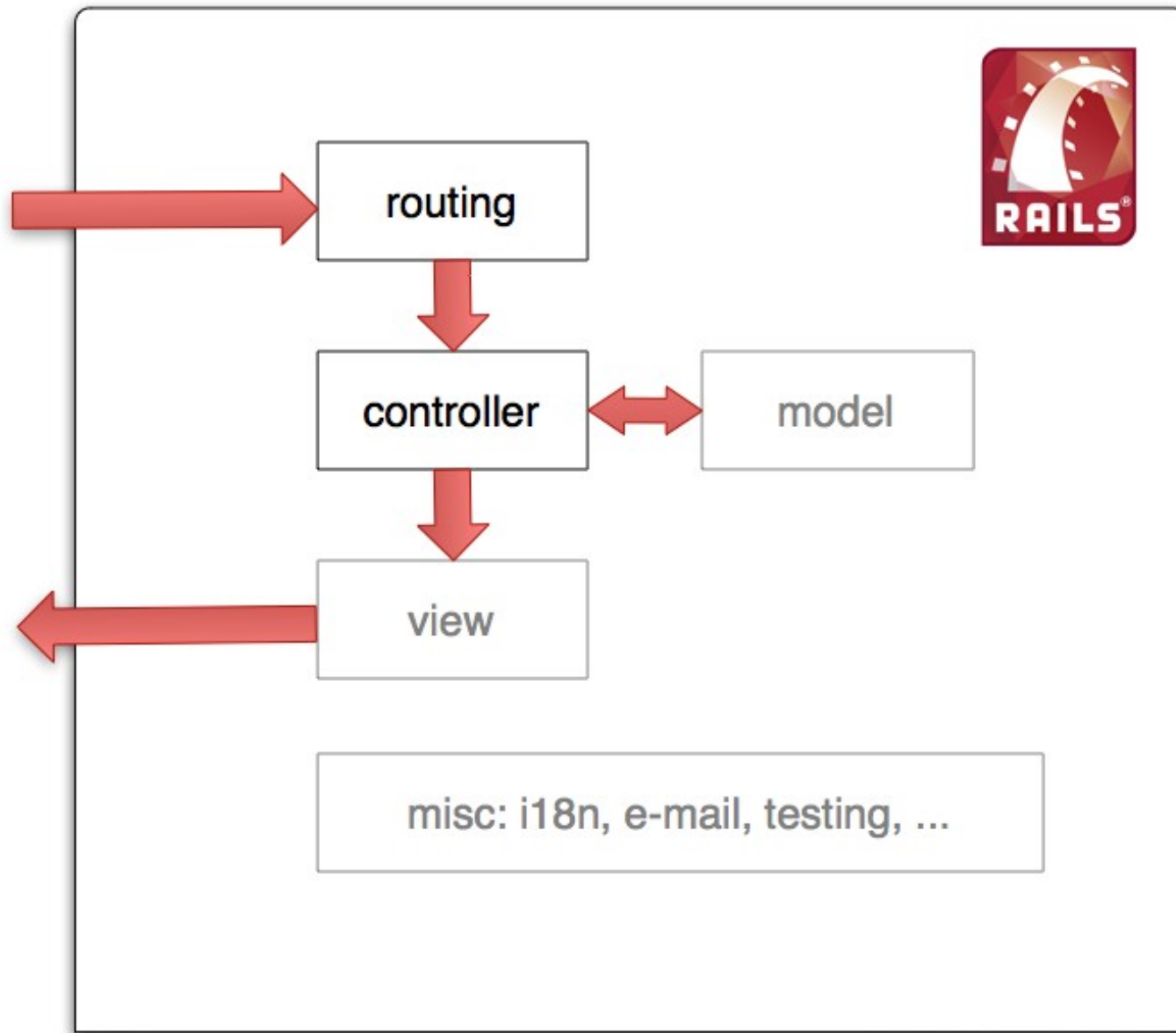
...pero no hay balas de plata



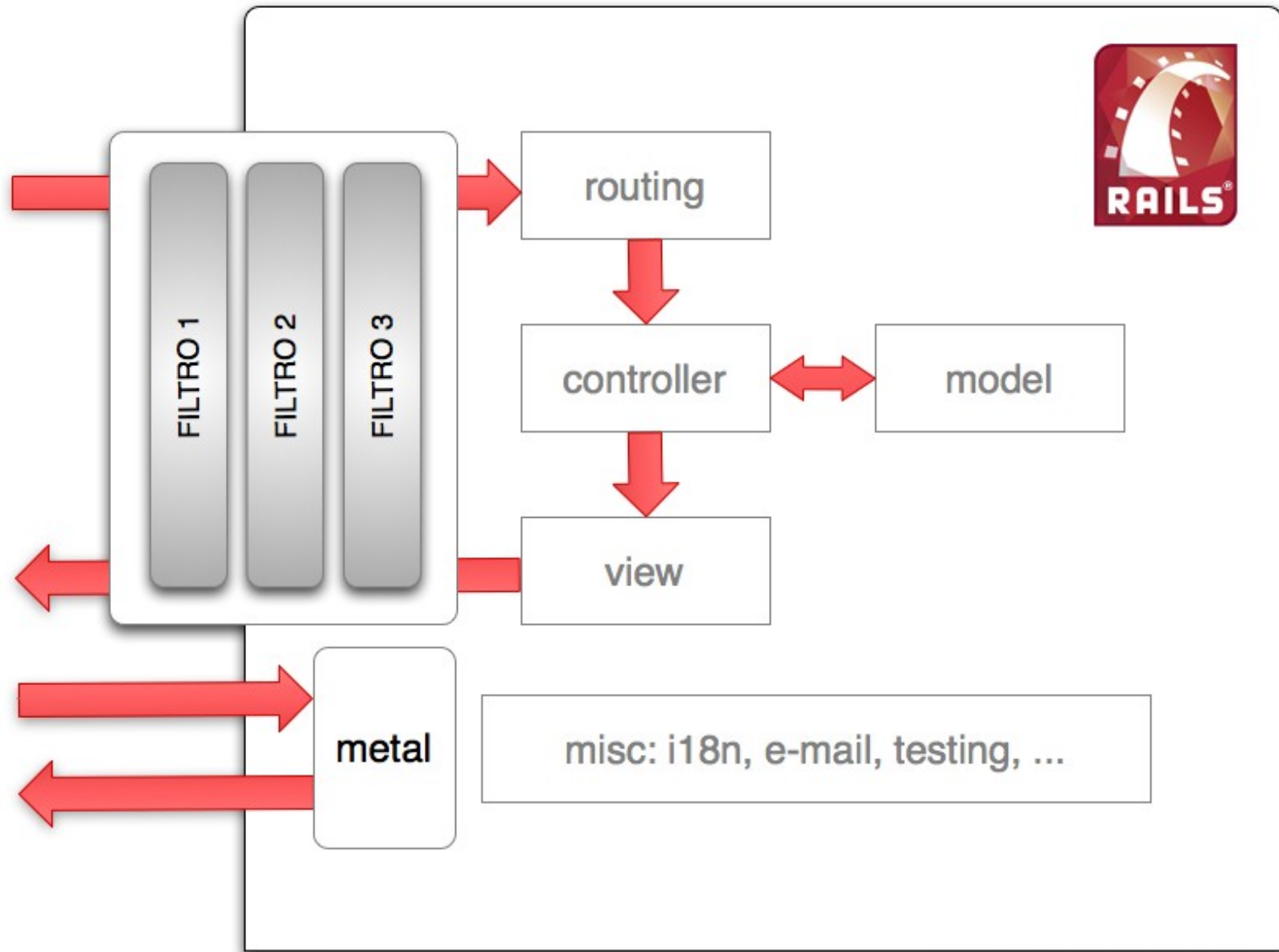
...pero no hay balas de plata



...pero no hay balas de plata



Rails Middleware / Metal



En el capítulo de ayer... “Rails 3: performance & rack integration”



[albertoperdomo](#): impressive improvements in rails 3 regarding modularity and performance... 3-10 faster than rails 2... wow [#conferenciaror](#)

about 2 hours ago from *TweetDeck* · [Reply](#) · [View Tweet](#)



[happywebcoder](#): maybe my talk has no sense now... :P RT: [@albertoperdomo](#): impressive improvements in rails3's modularity and performance [#conferenciaror](#)

about 2 hours ago from *Echofon* · [Reply](#) · [View Tweet](#)



[railshoster](#): [#rails3](#) The action itself is a rack object. Use existing rack middleware in a controller. How cool is that? [#conferenciaror](#)

about 2 hours ago from *Tweetie* · [Reply](#) · [View Tweet](#)



[biilmann](#): [#conferenciaror](#) Rack integration in Rails 3 is looking great

about 1 hour ago from *web* · [Reply](#) · [View Tweet](#)

En el capítulo de ayer... “Rails 3: performance & rack integration”



[albertoperdomo](#): impressive improvements in rails 3 regarding modularity and performance... 3-10 faster than rails 2... wow [#conferenciaror](#)

about 2 hours ago from *TweetDeck* · [Reply](#) · [View Tweet](#)



[happywebcoder](#): maybe my talk has no sense now... :P RT: [@albertoperdomo](#): impressive improvements in rails3's modularity and performance [#conferenciaror](#)

about 2 hours ago from *Echofon* · [Reply](#) · [View Tweet](#)



[railshoster](#): [#rails3](#) The action itself is a rack object. Use existing rack middleware in a controller. How cool is that? [#conferenciaror](#)

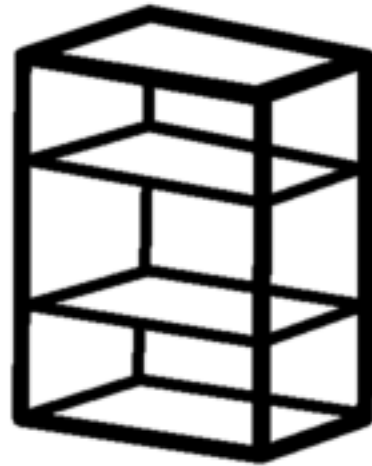
about 2 hours ago from *Tweetie* · [Reply](#) · [View Tweet](#)



[biilmann](#): [#conferenciaror](#) Rack integration in Rails 3 is looking great

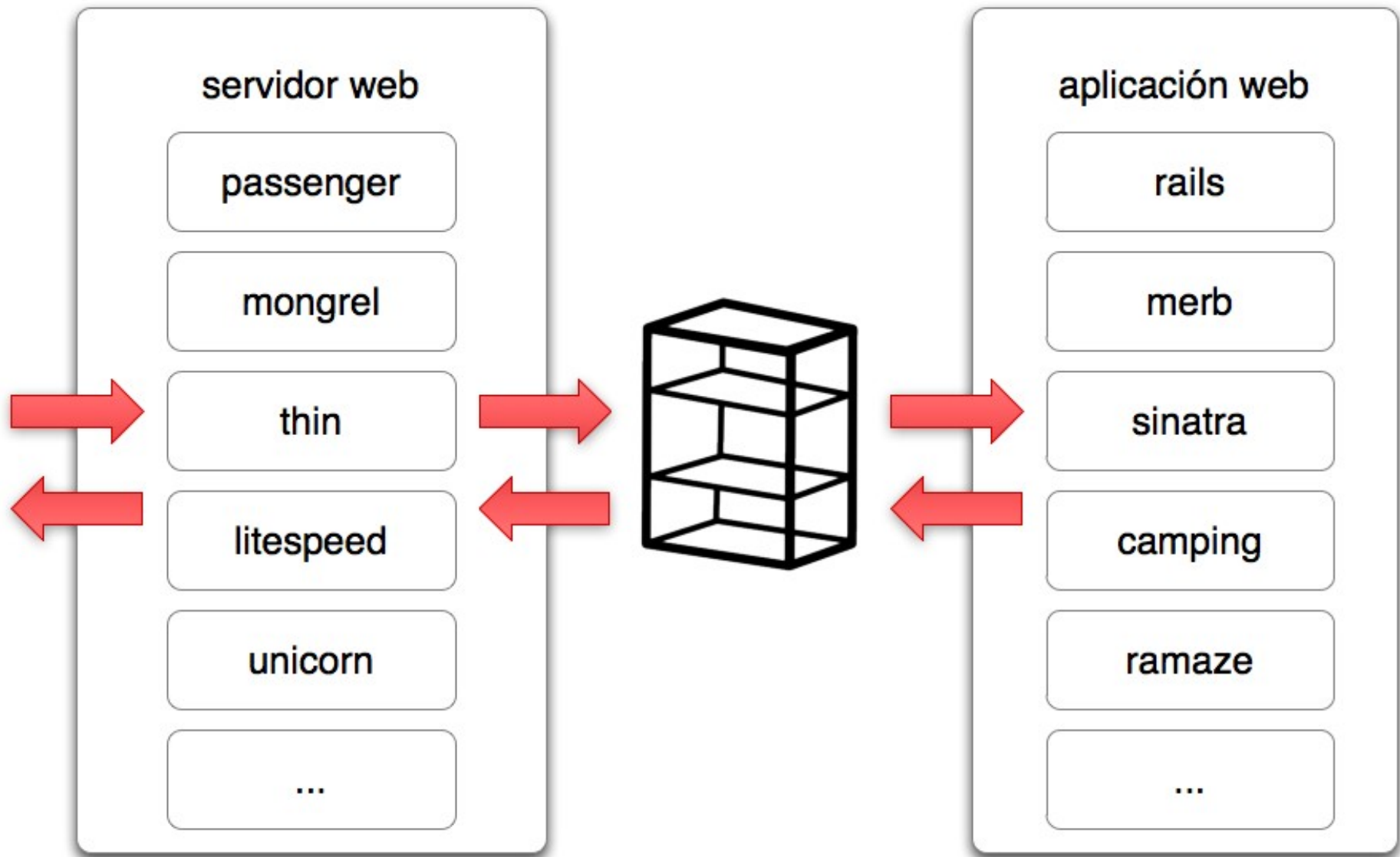
about 1 hour ago from *web* · [Reply](#) · [View Tweet](#)

What the f*ck is Rack?

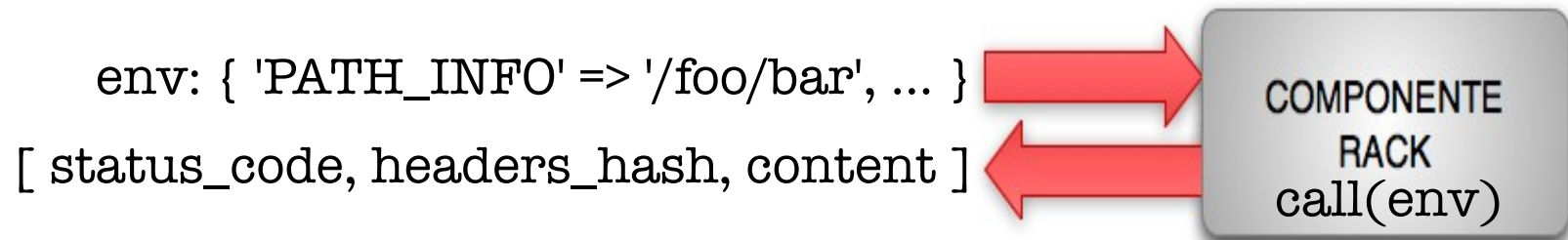


rack
powers web applications

Rack



Rack



```
1 # Minimal hello world rack application
2 #
3 # Run it via rackup: rackup config.ru
4
5 class HelloWorld
6   # Each incoming request invokes the application's call method
7   def call(env) # receives a hash with request data & rack config
8     [
9       200, # status code
10      {"Content-Type" => "text/html"}, # HTTP headers
11      ["Hello world!"] # content, should respond to 'each'
12    ]
13  end
14 end
15
16 # don't forget to run it
17 run HelloWorld.new
```

A screenshot of a web browser window with the address bar showing `http://localhost:9292/`. The page content displays "Hello world!".

Rack: handlers

```
1 #!/usr/bin/ruby
2
3 # Ruby script to run a Hello world rack app
4 # with a Mongrel handler included in rack.
5
6 require 'rubygems'; require 'rack'
7
8 class HelloWorld
9   def call(env)
10     [ 200, {"Content-Type" => "text/html"}, ["Hello world!"] ]
11   end
12 end
13
14 app = HelloWorld.new
15 Rack::Handler::Mongrel.run app, :Port => 9292 # defaults to 8080
16
17 # More handlers available:
18 # Rack::Handler::WEBrick.run app, :Port => 9292 # defaults to 80
19 # Rack::Handler...
```

Rack: rutas

```
1 # Rack::URLMap maps URLs or paths to applications
2
3 app1 = Proc.new{ |env| [200, {}, "Hi from app 1"] }
4 app2 = Proc.new{ |env| [200, {}, "Hi from app 2"] }
5 app3 = Proc.new{ |env| [200, {}, "Hi from app 3"] }
6
7 map "/app1" do
8   run app1
9 end
10
11 map "/app2" do
12   run app2
13 end
14
15 map "/app2/but-runs-app1" do # longer paths are preferred over short ones
16   run app1
17 end
18
19 # catches unknown requests, unmapped paths show a "not found" message
20 map "/" do
21   run app3
22 end
```

Rack: filtros

```
1 class Demo
2   def call(env)
3     post_params = Rack::Request.new(env).POST
4     body = <<-HTML
5       <form action="/" method="post">
6         <input type="text" name="_method" value="#{post_params['_method']}" />
7         <input type="submit" />
8       </form>
9       <p>REST method: #{env['REQUEST_METHOD']}</p>
10      HTML
11      [200, {"Content-Type" => "text/html"}, body]
12   end
13 end
14
15 use Rack::MethodOverride # applies the _method REST trick
16 run Demo.new
17
```

REST method: POST

REST method: PUT

Rack: filtros

```
1 class Demo
2   def call(env)
3     [ 200, {'Content-Type' => 'text/html'}, ['Hello world!'] ]
4   end
5 end
6
7 class UpcaseFilter
8   def initialize(app)
9     @app = app
10  end
11  def call(env)
12    status, headers, body = @app.call(env)
13    body.each{ |str| str.upcase! }
14    [ status, headers, body ]
15  end
16 end
17
18 use UpcaseFilter
19 run Demo.new
```

Rack: apilando filtros

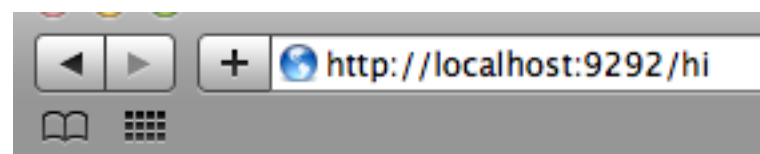
```
11 class Filter1 < BaseFilter
12   def call(env)
13     env['QUERY_STRING'] << ' 1'
14     response = @app.call(env)
15     response.last << ' 1'
16     response
17   end
18 end
19
20 class Filter2 < BaseFilter
21   def call(env)
22     env['QUERY_STRING'] << ' 2'
23     response = @app.call(env)
24     response.last << ' 2'
25     response
26   end
27 end
28
29 # filters are stacked
30 use Filter1
31 use Filter2
32 run Demo.new
```

req: 1 2

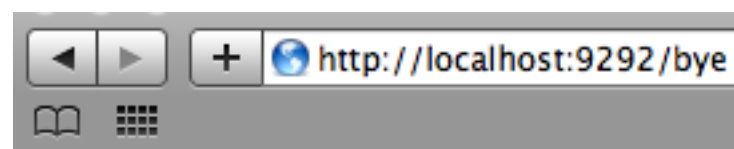
resp: 2 1

Rack: seleccionando filtros

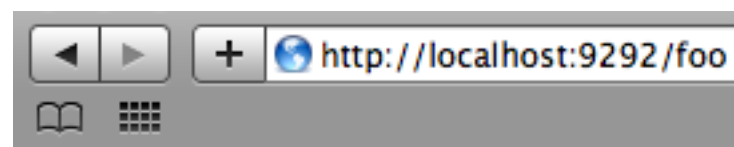
```
31 use Uppcase
32
33 map '/hi' do
34   run Hi.new
35 end
36
37 map '/bye' do
38   run Bye.new
39 end
40
41 map '/foo' do
42   use Reverse
43   run Hi.new
44 end
45
```



HELLO WORLD!



GOODBYE WORLD!



!DLROW OLLEH

Rails  Rack

Rails Middleware

rake middleware muestra la pila de filtros

```
use Rack::Lock
use ActionController::FailSafe
use ActionController::Session::CookieStore, #<Proc:0x020d98
use ActionController::ParamsParser
use Rack::MethodOverride
use Rack::Head
use ActiveRecord::ConnectionAdapters::ConnectionManagement
use ActiveRecord::QueryCache
run ActionController::Dispatcher.new
```

Rails Middleware

```
1 # save as RAILS_ROOT/lib/upcase.rb
2 class Upcase
3   def initialize(app)
4     @app = app
5   end
6   def call(env)
7     status, headers, response = @app.call(env)
8     response.each{ |str| str.upcase! }
9     [ status, headers, response ]
10  end
11 end
```

Rails Middleware

```
# RAILS_ROOT/config/environment.rb

Rails::Initializer.run do |config|
  # ...
  config.middleware.use 'Upcase'
  # ...
end
```

```
use Rack::Lock
use ActionController::FailSafe
use ActionController::Session::CookieStore, #<Proc:0x017f7c
use ActionController::ParamsParser
use Rack::MethodOverride
use Rack::Head
use Upcase
use ActiveRecord::ConnectionAdapters::ConnectionManagement
use ActiveRecord::QueryCache
run ActionController::Dispatcher.new
```

Rails Metal

```
script/generate metal HelloWorld
```

```
create app/metal  
create app/metal/hello_world.rb
```

```
use Rack::Lock  
use ActionController::FailSafe  
use ActionController::Session::CookieStore, #<Proc:0x017f4f  
use Rails::Rack::Metal  
use ActionController::ParamsParser  
use Rack::MethodOverride  
use Rack::Head  
use Upcase  
use ActiveRecord::ConnectionAdapters::ConnectionManagement  
use ActiveRecord::QueryCache  
run ActionController::Dispatcher.new
```

Rails Metal

```
1 # Allow the metal piece to run in isolation
2 unless defined?(Rails)
3   require(File.dirname(__FILE__) + "../../../config/environment")
4 end
5
6 class HelloWorld
7   def self.call(env)
8     if env["PATH_INFO"] =~ /^\/hello_world/
9       [200, {"Content-Type" => "text/html"}, ["Hello, World!"]]
10    else
11      [404, {"Content-Type" => "text/html"}, ["Not Found"]]
12    end
13  end
14 end
```

Ecosistema Rack

- Rack::Utils: utilidades para frameworks
escape_html, parse_query, ...
- Rack::Contrib: componentes Rack
ETags, cookies, profiling, ...
- Rack::Test
- CodeRack.org



Sinatra

- Micro framework, 1 ó 2 dependencias: rack, shotgun
- Micro aplicaciones

```
1 require 'rubygems'
2 require 'sinatra'
3
4 # runs by default at port 4567
5
6 get '/' do
7   'Hello world!'
8 end
```

- Se ejecuta como un script ruby más

Sinatra: parámetros en las rutas

```
3
4 ▾ get "/hi/:name" do
5     "Hello #{params[:name]}"
6 ▴ end
7
8 ▾ get "/bye/:name?" do # optional parameter
9 ▾     if params[:name].nil?
10         "Who are you?"
11     else
12         "Goodbye #{params[:name]}"
13 ▴     end
14 ▴ end
15
16 # parameters can be accessed like block parameters:
17 ▾ get '/foo/:name/:surname' do |name, surname|
18     "#{surname}, #{name} #{surname}"
19 ▴ end
```

Sinatra: parámetro splat

```
4 # /raul/murciano/foobar/123/456:
5 # params[:splat] => ['raul/murciano','123/456']
6 # shows:
7 # Hello raul/murciano, 123/456!
8 get '/*/foobar/*' do
9   "Hello #{params[:splat].join(' ', ')}!"
10 end
11
12 # /raul/murciano/123
13 # params[:splat] => ['raul/murciano/123']
14 # shows:
15 # Bye raul/murciano/123!
16 get '/*' do
17   "Bye #{params[:splat].join(' ', ')}!"
18 end
```

Sinatra: vistas inline

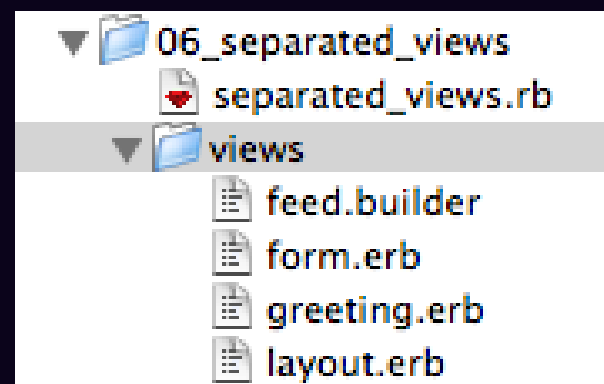
```
4 ▾ get '/' do
5   @now = Time.now.strftime("%H:%M:%S")
6   erb 'Hello world at <%= @now %>!'
7 ▴ end
8
9 ▾ get '/haml' do
10   haml '%h1 Hi Conf.Rails!'
11 ▴ end
12
13 ▾ get '/foo.xml' do
14   content_type 'text/xml'
15 ▾ builder do |xml|
16 ▾   xml.parent do
17     xml.child "Hi there!"
18 ▴   end
19 ▴ end
20 ▴ end
21
```

Sinatra: vistas infile

```
4 get '/' do
5   @now = Time.now.strftime("%H:%M:%S")
6   erb :home
7 end
8
9 # views can be declared with a block...
10 template :home do
11   "This is my view at <strong><%= @now %></strong>"
12 end
13
14 get "/about" do
15   erb :about
16 end
17
18 # ...or after the source code!
19
20 __END__
21
22 @@about
23 In-file views sample application for Conf.Rails 2009
```

Sinatra: vistas separadas

```
4 # The views directory can be configured with this setting:
5 # set :views, Proc.new { File.join(root, 'views') }
6 #
7 # And the root is configurable too:
8 # set :root, File.dirname(__FILE__)
9
10 get '/feed.xml' do
11   content_type :xml
12   builder :feed
13 end
14
15 get '/' do
16   erb :form
17 end
18
19 post '/' do
20   @username = params[:user][:first_name] + ' ' + params[:user][:last_name]
21   erb :greeting
22 end
23
24 # Sinatra looks for views on the source file first and at /views later
25 # ...so if I write a @@form view down here the /views/form.erb will be ignored
26
27 __END__
```

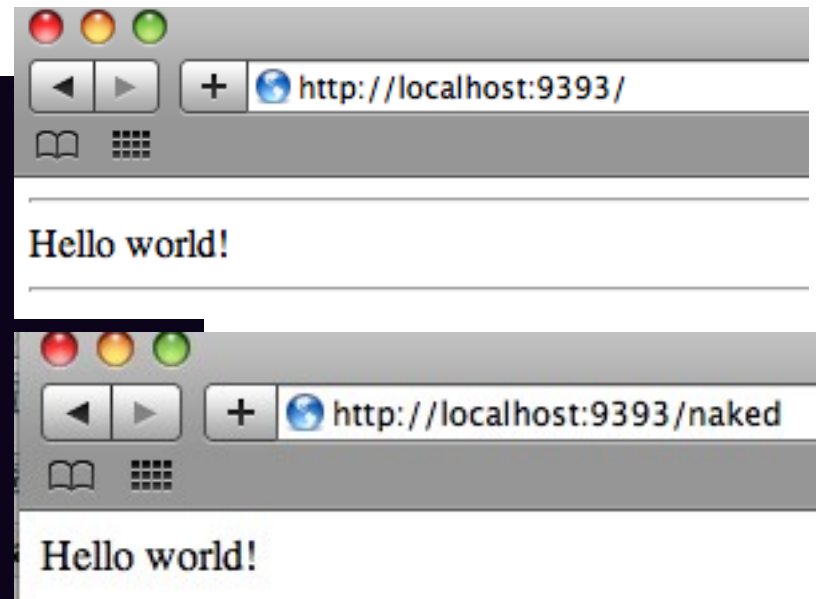


Sinatra: helpers

```
4 # helpers are available in controllers and views
5 helpers do
6   def time_format(time)
7     time.strftime("%H:%M:%S")
8   end
9 end
10
11 get '/' do
12   @time = time_format(Time.now)
13   erb :index
14 end
15
16 get '/inline' do
17   "Using helper in an inline view at #{time_format(Time.now)}"
18 end
19
20 __END__
21
22 @@index
23 Using helper in a separate view at <%= time_format(Time.now) %>
24
```

Sinatra: layout

```
4 ▾ get '/' do
5   erb :index
6 ▴ end
7
8 ▾ get '/naked' do
9   erb :index, :layout => false
10 ▴ end
11
12 ▾ template :layout do
13   "<hr /><%= yield %><hr />"
14 ▴ end
15
16 ▾ template :index do
17   "Hello world!"
18 ▴ end
```



Sinatra: partials

```
3
4 # partials are not built-in
5 helpers do
6   def erb_partial(template, options={})
7     erb template, options.merge!(:layout => false)
8   end
9 end
10
11 get '/' do
12   @where = 'Conferencia Rails 2009'
13   erb :index
14 end
15
16 __END__
17
18 @@index
19 Partial: <%= erb_partial :my_partial, :locals => {:place => @where} %>
20
21 @@my_partial
22 <strong>I'm talking at <%= place %></strong>
23
```


Sinatra: REST

```
3
4 # - - - - - Model - - - - - #
5 require 'dm-core'
6 DataMapper.setup(:default, 'sqlite3::memory:')
7
8 class User
9   include DataMapper::Resource
10   property :id,      Serial
11   property :name,    String
12   property :email,   String
13 end
14
15 User.auto_migrate!
16
```

PUT, DELETE _method
Params anidados

```
18 # - - - - - Controller - - - - - #
19
20 ▶ get '/' do...
23
24 ▶ get '/new' do...
27
28 ▶ post '/' do...
33
34 ▶ get '/edit/:id' do...
38
39 ▶ put '/:id' do...
45
46 ▶ delete '/:id' do...
51
```

Sinatra: filtros

```
3
4 # - - - Before filters - - - #
5 before do
6   set_time
7 end
8
9 def set_time
10   @time = Time.now.strftime("%H:%M:%S")
11 end
12
13 # - - - Controllers - - -
14 get '/' do
15   "Hello at #{@time}"
16 end
17
```

(Aplican a todos, no hay :only ni :except)

sinatra / sinatra  fork  watch  download

Fork of [bmizerany/sinatra](#)

Description: Classy web-development dressed in a DSL (official / canonical repo)

Homepage: <http://sinatra.github.com>

Clone URL: <git://github.com/sinatra/sinatra.git> 

Adds after filters

Originally by jschimenti (<http://bit.ly/1RTt2H>)
Updated for Sinatra 1.0 by rtomayko



jschimenti (author)
December 22, 2008



rtomayko (committer)
November 11, 2009

Sinatra: configuración

```
3
4 set :foo, "bar"
5
6 # environment gets loaded by RACK_ENV value, development by default
7 # RACK_ENV=production ruby configuration.rb
8
9 ▾ configure :development do
10   set :my_server, 'example.com'
11 ▴ end
12
13 ▾ configure :production do
14   set :my_server, 'mysite.com'
15 ▴ end
16
17 ▾ get "/" do
18   "Foo: #{options.foo}, Server: #{options.my_server}"
19 ▴ end
20
```

Sinatra: gestión de errores

```
20 # enabling show_exceptions we would see the error backtrace
21 set :show_exceptions, false
22
23 # generic error
24 error do
25   "Flying whale goes here. | #{request.env['sinatra.error'].to_yaml}"
26 end
27
28 # customized error
29 class EpicFailError < Exception; end
30 error EpicFailError do
31   "Oh, oh: #{request.env['sinatra.error'].message}"
32 end
33
34 # 404:
35 not_found do
36   "Awesome 404 screen"
37 end
```

Sinatra Rack

```
3
4 class UpcaseFilter
5   def initialize(app)
6     @app = app
7   end
8   def call(env)
9     status, headers, body = @app.call(env)
10    body.each{ |str| str.upcase! }
11    [ status, headers, body ]
12  end
13 end
14
15 use UpcaseFilter
16
17 get "/" do
18   "Hello world!"
19 end
20
```

Sinatra: misc

- Sesiones, cookies vía Rack::Session
Gotchas ;)
 - no usa secret por defecto!
 - Si guardas un array has de deserializarlo a mano
- Cabeceras, redirecciones
- Archivos estáticos, /public
- Extensiones

github.com/nesquena/sinatra_more/

Sinatra: base

```
1 require 'rubygems'
2 require 'sinatra/base' # not just 'sinatra'!!!
3
4 class MyApp < Sinatra::Base
5
6   # Sinatra::Base has some configuration values
7   # by default that are different from a raw Sinatra application
8
9   get "/" do
10     "Hi there!"
11   end
12
13 end
14
15 MyApp.run!
```

Sinatra: ejemplos

- Taps: a simple database agnostic import/export app to transfer data to/from a remote database
- Integrity: easy and fun Continuous Integration server
- Panda: Video encoding made easy with AWS
- Hancock: Single Sign On Server
- Shrtr: a link shortener service
- TooPaste: pastie clone
- ...

Sinatra como Rails Metal

```
1 # RAILS_ROOT/app/metal/frank.rb
2
3 # Allow the metal piece to run in isolation
4 unless defined?(Rails)
5   require(File.dirname(__FILE__) + "../../config/environment")
6 end
7
8 # - - - application - - - #
9 require 'sinatra/base'
10 class AwesomeSinatraApp < Sinatra::Base
11   get "/frank" do
12     "New York, New York"
13   end
14 end
15
16 # - - - metal invokes the app - - - #
17 class Frank
18   def self.call(env)
19     @@app ||= AwesomeSinatraApp.new
20     if env["PATH_INFO"] =~ /^\/frank/
21       @@app.call(env)
22     else
23       [404, {"Content-Type" => "text/html"}, ["Not Found"]]
24     end
25   end
26 end
```

Rendimiento Rails Metal

Pratik Naik: 1-2ms de diferencia.



DHH:

- “the difference is tiny for regular HTML-based web application stuff”,
- “you shouldn’t jump to metal before Action Controller has proved itself to be too slow”

tinyurl.com/perf-rails-metal

Rendimiento Rails/Sinatra/Rack

tinyurl.com/ruby-web-performance

	-n 1000 -c 1	-n 1000 -c 10	-n 10000 -c 50		
Trials	Average Requests/sec	Average Requests/sec	Average Requests/sec		
Plain HTML					
Rails - Thin	668	722	714		
Rails - Apache Passenger	495	760	815		
Sinatra	1,814	2,132	2,175		
Rack	3,145	3,837	4,086		
Apache	3,948	5,894	5,802		
Nginx	6,975	7,925	8,004		
	2,841	3,545	3,599		
HAML					
Rails - Thin	647	700	661		
Rails - Apache Passenger	449	664	765		
Sinatra	831	921	844		
Rack	1,189	1,338	1,325		
	779	906	899		
ERB					
Rails - Thin	671	746	710		
Rails - Apache Passenger	423	674	755		
Sinatra	1,248	1,376	1,333		
Rack	2,328	2,721	2,782		
	1,167	1,379	1,395		
Builder					
Rails - Thin	610	635	624		
Rails - Apache Passenger	417	643	716		
Sinatra	1,153	1,313	1,242		
Rack	2,351	2,544	2,856		
	1,133	1,284	1,359		

There are no silver bullets

There are no silver bullets

...so Use the Best Tool for the Job

Use the Best Tool for the Job

Generadores de contenido estático:

- Jekyll: wiki.github.com/mojombo/jekyll
- Nanoc: nanoc.stonship.org
- StaticMatic: staticmatic.rubyforge.org
- Webby: webby.rubyforge.org

From Mike Gunderloy: gist.github.com/242751

Use the Best Tool for the Job

En mi opinión...

- Usar Rails middleware/metal por rendimiento sólo tiene sentido en casos muy puntuales
- Rails middleware/metal es una buena alternativa a mod_rewrite y similar

Use the Best Tool for the Job

En mi opinión...

- Rack es un protocolo perfecto, pero es demasiado áspero como framework
- Merece la pena echar un vistazo a Rack::Contrib, Rack::Utils y Rack::Test

Use the Best Tool for the Job

En mi opinión...

- Sinatra != Spaghetti code
github.com/raul/sinatra_template
- Sinatra es muy útil para:
 - dar interfaz web a un servicio ya existente
 - webservices a medida
 - proyectos que se alejan mucho del stack Rails
 - micro aplicaciones

Use the Best Tool for the Job

En mi opinión...

- Sinatra != Spaghetti code
github.com/raul/sinatra_template
- Sinatra es muy útil para:
 - dar interfaz web a un servicio ya existente
 - webservices a medida
 - proyectos que se alejan mucho del stack Rails
 - micro aplicaciones

¡ ¡ ¡divertirse!!!

Sinatra
Rails
everybody



Rack

Enlaces

- Rack:
rack.rubyforge.org
- Sinatra:
sinatrarb.com
- Rails & Rack (Middleware/Metal):
guides.rubyonrails.org/rails_on_rack.html

Gracias! :)

¿Preguntas?

github.com/raul/conferencia_rails_2009

twitter.com/happywebcoder

workingwithrails.com/person/5988-raul-murciano