

Z Rails do Merba i... z powrotem

Marcin Kulik, Piotr Solnica

13/01/2009, KRUG, Lunar Logic Polska


- 1 Wprowadzenie
 - Na początku był Rubytime
 - Rodzi się "konkurencja"
 - Rubytime 3
 - I ponownie Rails
- 2 Merb
 - Założenia Merba
 - Router
 - Kontrolery
 - Slices
 - Parts
- 3 Skutki połączenia Merb z Rails
 - Na plus
 - Na minus
- 4 The End

Plan prezentacji

- 1 Wprowadzenie
 - Na początku był Rubytime
 - Rodzi się "konkurencja"
 - Rubytime 3
 - I ponownie Rails
- 2 Merb
 - Założenia Merba
 - Router
 - Kontrolery
 - Slices
 - Parts
- 3 Skutki połączenia Merb z Rails
 - Na plus
 - Na minus
- 4 The End


Jest 2005 rok
Rails to nowość


LLP potrzebuje systemu do timetrackingu
Powstaje Rubytime 1



RUBYTIME


[Your data](#)
[Clients](#)
[Projects](#)
[Activities](#)
[Invoices](#)
[Users](#)
[Roles](#)


Welcome:
ALEXANDER


 Your profile


 Enter activity

 **Calendar**

 Help

 Export data

 Previous day

 Logout

Credits
v.1.3.0

Calendar view

Year: [2006](#) **2007** [2008](#)
 Month: [Jan](#) [Feb](#) [Mar](#) [Apr](#) **[May](#)** [Jun](#) [Jul](#) [Aug](#) [Sep](#) [Oct](#) [Nov](#) [Dec](#)

Mon	Tue	Wed	May Thu	Fri	Sat	Sun
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
		4:00 doing sth »		4:00 doing sth »		
		total: 4:00		total: 4:00		
21	22	23	24	25	26	27
28	29	30	31	1	2	3
			1:00 Yesterday act... »			
			total: 1:00			

[Back](#)

Rubytime 2 jako naturalne przedłużenie wersji 1

Nadal Rails

W między czasie powstają:

- Ramaze
- Mack
- Sinatra
- Camping
- Merb

Jednak Rails nadal "bezkonkurencyjny":

- największy userbase spośród wymienionych
- coraz więcej literatury
- coraz łatwiejsze hostowanie (Passenger)

Ale czy napewno jedyny liczący się framework dla Ruby'ego?
Niekoniecznie!

Potrzeba "odświeżenia" RT
+
zbliżający się wielkimi krokami Merb 1.0
=
Rubytime 3

Welcome, Admin Adminiusz | [Preferences](#) [Logout](#)

 **rubytime.**

Activities Invoices Manage + Add new activity

List Calendar

List of activities

Found 1 matching activities from 25-11-2008 .



Client: Apple



Project: Big project for Apple



Role: Developer



User

Date

Hours



Admin Adminiusz

01-12-2008

2:15



Total for role: 2:15

Total for project: 2:15

Total for client: 2:15

[Export this list to CSV](#)

Filter

Client

Project

Role

User

From

To

Include ☒ All
☐ Invoiced
☐ Not invoiced

Wrażenia?



Merb łączy się z Rails ?!

Rails 2.3 ewoluuje w kierunku Rails 3
Merb 1.0 ewoluuje w kierunku Merb 2
Merb 2 == Rails 3

Plan prezentacji

- 1 Wprowadzenie
 - Na początku był Rubytime
 - Rodzi się "konkurencja"
 - Rubytime 3
 - I ponownie Rails
- 2 Merb
 - Założenia Merba
 - Router
 - Kontrolery
 - Slices
 - Parts
- 3 Skutki połączenia Merb z Rails
 - Na plus
 - Na minus
- 4 The End

Szybkość i zużycie pamięci:

- No code is faster than no code (Call stack w Rails to 41-58 wywołań, w Merb 27-32 wywołań)
- Wszystko co spowalnia Merba jest bugiem
- Rails 24K-50K LOC, Merb 10K-17K LOC
- Bez magicznych sztuczek (alias_method_chain, method_missing itp wypadają...)

Elastyczność

- ORM (DataMapper, ActiveRecord, Sequel...)
- Javascript library (jQuery, Prototype...)
- Testowanie (unit, rspec...)
- Template engine (erb, haml...)

Modularność ("nie, nie potrzebujesz wszystkiego")

- Rails 6 gemów
- Merb 18 (core + more)

Stabilne publiczne API i pluginy

- API miało się nie zmieniać dla wersji major, czas pokaże
- pluginy to Gemy
- zapewniają zależności
- korzystają z API zamiast opierać się na ciągle zmieniających się metodach prywatnych

Rack

- odpowiednik Pythonowego WSGI czy Javowego Servlet API
- nie przywiązuje aplikacji do konkretnego serwera
- obsługa w Passenger

Generowanie aplikacji

- fullstack: merb-gen app
- core: merb-gen core
- simple: merb-gen flat
- single-file: merb-gen very_flat

Router

```
Merb::Router.prepare do
  # resource'y
  resources :articles do
    resources :comments
  end

  # mapowanie domeny
  match(:domain => "somecooldomain.com").to(:controller => "main").name(:main)

  # mapowanie subdomen do kont uzytkownikow
  match('/', :subdomains => /(.)+/.to(:controller => 'accounts', :action => 'dashboard')

  # opcjonalne elementy
  match("/articles/(/:year(/:month(/:day)))/:title").to(:controller => "articles",
    :action => "show").name(:article)
end

url(:main)
resources(:articles)
resource(@article)
resource(@article, :comments)
resource(@article, @comment)
```

provides + display + render

```
class Posts < Application
  provides :html, :json, :xml

  def index
    @posts = Post.all
    display @posts # jesli nie ma szablonu, @posts.to_json
  end

  def edit
    render # renderuje posts/edit.{html, json, xml}.erb
  end

  def foo
    only_provides :html
  end
end
```


Przetwarzanie w tle

```
class Videos < Application
  def index
    @posts = Post.all
    display @posts
  end

  def create
    video = Video.create(params[:video])
    run_later do
      video.process
    end
    redirect_to resources(:videos)
  end
end
```

Asset bundling

```
<%= css_include_tag 'reset.css', 'master.css', 'jquery.treeview.css',  
  :bundle => true %>  
  
<%= js_include_tag 'jquery-1.2.3.pack', 'jquery.treeview.pack.js', 'application.js',  
  :bundle => true %>
```

Wyjątki:

raise NotFound odpali: kontroler Exceptions, akcja :not_found,
status 404

action-args

```
class Videos < Application
  def index(offset=nil)
    @posts = Post.all(:offset => ofsset, :limit => 15)
    display @posts
  end

  def create(video)
    video = Video.create(video)
    redirect_to resources(:videos)
  end
end
```



Slice = aplikacja w aplikacji, zawiera własne:

- kontrolery,
- model,
- widoki
- routes'y

merb-auth = "restful_authentication done right"

Plan prezentacji

- 1 Wprowadzenie
 - Na początku był Rubytime
 - Rodzi się "konkurencja"
 - Rubytime 3
 - I ponownie Rails
- 2 Merb
 - Założenia Merba
 - Router
 - Kontrolery
 - Slices
 - Parts
- 3 Skutki połączenia Merb z Rails
 - Na plus
 - Na minus
- 4 The End

Pozytywne aspekty połączenia:

- większy Rails core-team (Merb core-team to mądrzy goście)
- router oraz bootloader Merb w Rails (prace już trwają)

Pozytywne aspekty połączenia:

- `respond_as/respond_with` jako odpowiednik merbowych `provides` i `display` (prace już trwają)
- Rails mają zyskać `slice'y`, jeszcze lepsze niż były w Merbie
- czyszczenie kodu

Negatywne aspekty połączenia:

- brak poważnej konkurencji = mniejsze tempo rozwoju
- DHH.. Czy współpraca z Davidem przyniesie zakładane wyniki?

Plan prezentacji

- 1 Wprowadzenie
 - Na początku był Rubytime
 - Rodzi się "konkurencja"
 - Rubytime 3
 - I ponownie Rails
- 2 Merb
 - Założenia Merba
 - Router
 - Kontrolery
 - Slices
 - Parts
- 3 Skutki połączenia Merb z Rails
 - Na plus
 - Na minus
- 4 The End

Dziękuję. Pytania?