Ruby on Rails

Silesian Ruby Users' Group

3 listopada 2008

Dlaczego web development?

- przenośność
- niezależność od systemów operacyjnych i przeglądarek
- łatwe zarządzanie i utrzymanie

Dlaczego Ruby on Rails?

- przyjemność z programowania
- open source
- czytelność kodu
- propagowanie dobrych praktyk programistycznych
- szybkość programowania
- łatwość reagowania na zmiany
- niezależność od systemu operacyjnego
- niezależność od środowiska pracy (NetBeans, RadRails, Emacs, Vim, JEdit)

Ruby

- 1995 rok, Yukihiro Matsumoto aka Matz
- inspirowany przez CLU, Eiffel, Lisp, Perl, Python, Smalltalk
- interpretowany
- wieloparadygmatowy
- bardzo wysokiego poziomu (VHLL)
- w pełni obiektowy

Ruby - implementacje

- interpretery: MRI, Ruby Enterprise, JRuby, IronRuby
- maszyny wirtualne: MagLev, Rubinius, YARV

Samokomentujący się kod

```ruby
"!dlroW ,olleH".reverse

3.times { puts "Ruby rulez!" }

exit if restaurants.include? "wiedeński"

[5, 20, 35, 10, 10, 30].sort.last

raise ArgumentError unless argument.kind_of? String

return words.size if words.respond_to? "size"
```

Nazewnictwo

```
$uczelnia = "Polibuda"

class Straz
  def self.wezwij(akademik)
    # ...
  end
end

class OndraszekPortierka
  AKADEMIK = :ondraszek
  @@klucze = OndraszekKlucze.new

  def initialize(kolor)
    @kamizelka = Kamizelka.new(kolor)
  end

  def wezwij_straz
    Straz.wezwij(AKADEMIK)
  end
end
```

Stałe są zmienne

```
SRUG_EMAIL = "spotkania@srug.pl"
SRUG_EMAIL = "admin@srug.pl"
# warning: already initialized constant SRUG_EMAIL
```

Symbole

- omijane przez garbage collector
- wykorzystywane m.in. jako klucze w tablicach asocjacyjnych

Przykład

```
"string".object_id
-606377798
"string".object_id
-606384058
:symbol.object_id
204898
:symbol.object_id
204898
```

Tablice i hasze

```
a = [1, 2, 3]
a[0]
#=> 1

h = { :name => "John", :surname => "Doe" }
h[:name]
#=> "John"


[1, "napis", [1, 2], :symbol, { :one => 1 }]

{ :conditions => { :title => "SRUG" } }

{ [1, 2] => :array }
```

Dziedziczenie

- dziedziczenie jednobazowe
- moduły - rodzaj dziedziczenia wielobazowego pozwalający włączyć gotową implementację zbioru metod do danej klasy (*mixin*)
- dziedziczenia używa się znacznie oszczędniej i rzadziej niż np. w Javie

Przykład

```
module NazwaModulu
  # ...
end

class KlasaPochodna < KlasaBazowa
  include NazwaModulu
end
```

"If it walks like a duck and quacks like a duck, I would call it a duck."

James Whitcomb Riley

Duck typing

- rozpoznawanie typów na podstawie ich zachowania, a nie deklaracji

Duck typing - przykład

```ruby
class Duck
  def quack
    "Quack!"
  end
end
class Dog
  def quack
    "Quack!"
  end
end
def make_it_quack(duck)
  puts duck.quack
end

duck = Duck.new
dog = Dog.new

make_it_quack(duck)
# Quack!
make_it_quack(dog)
# Quack!
```

Domknięcia

- bloki kodu mogą być przekazywane jako argumenty i zwracane jako wynik działania funkcji (metody)
- są podstawową cechą języków funkcyjnych
- odwołują się do zmiennych z kontekstu, w którym zostały stworzone, a nie z którego są wywoływane

Domknięcia - przykład

```ruby
def say_something
  something = "Hello!"
  lambda { puts something }
end

result = say_something
something = "Bye!"
result.call
# Hello!
```

Domknięcia - wykorzystanie

```
3.times { print "SRUG" }
# SRUGSRUGSRUG

5.upto(10) { |i| print i, " " }
# 5 6 7 8 9 10

["Żubr", "Żywiec", "Harnaś"].select do |drink|
  drink.beer?
end
#=> ["Żubr", "Żywiec"]

open "data.txt" do |file|
  file.each_line do |line|
    MyParser.parse line
  end
end
```

Otwarte klasy - przykład

```ruby
class Array
  def shuffle
    sort_by { rand }
  end
end

a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
a.shuffle
#=> [2, 9, 4, 5, 1, 7, 8, 3, 6]
```

Monkey patching - problem

```ruby
array = [1, 7, 2, 8, 7, 10, 15, 2, 11]

array.index(8)
#=> 3

array.index { |element| element > 10 }
# ArgumentError: wrong number of arguments (0 for 1)
```

Monkey patching - rozwiązanie

```
class Array
  alias_method :old_index, :index

  def index(object = nil)
    if object.nil?
      self.each_with_index do |element, i|
        return i if yield element
      end
      nil
    else
      old_index(object)
    end
  end
end
```

Metaprogramowanie

- metaprogramowanie w Rubim jest **proste**
- dysponujemy programowalnym językiem programowania
- główne narzędzie służące do budowy tzw. DSL

Metaprogramowanie - problem

```
class Song
  def name
    @name
  end
  def name=(value)
    @name = value
  end
  def duration
    @duration
  end
  def duration=(value)
    @duration = value
  end
end
```

Metaprogramowanie - rozwiązanie

```
class Module
  def attr_accessor(*symbols)
    symbols.each do |symbol|
      module_eval "def #{symbol}
                     @#{symbol}
                   end"
      module_eval "def #{symbol}=(value)
                     @#{symbol} = value
                   end"
    end
  end
end
```

Metaprogramowanie - rozwiązanie, c.d.

```ruby
class Song
  attr_accessor :name, :duration
end

song = Song.new
song.name, song.duration = "Miles Davis - So What", 565
puts "#{song.name}, #{song.duration} s"
```

Użycie method_missing

```
class SupermarketTeller
  def method_missing(method_name, *args)
    if method_name.to_s.include? "lidl"
      puts "Lidl jest tani!"
    else
      raise NoMethodError,
        "undefined method `#{method_name}' for #{self}"
    end
  end
end
```

Użycie method_missing, c.d.

```
supermarket_teller = SupermarketTeller.new

supermarket_teller.jaki_jest_lidl?
# Lidl jest tani!

supermarket_teller.jaka_jest_biedronka?
# NoMethodError: undefined method ...
```

Podsumowanie

- „pseudo-code that runs” - skupianie się na rozwiązaniu problemu, nie na języku
- język zaprojektowany **dla ludzi**
- radość z programowania
- TIMTOWTDI - wolność wyboru (jak w Perlu, przeciwnie niż w Pythonie)
- zasada najmniejszego zaskoczenia - Ruby jest intuicyjny

Rake - Ruby Make

```
task :default => [:evince]

SRC = ["srug1.tex"]

rule ".pdf" => ".tex" do |t|
  sh "pdflatex -interaction=nonstopmode #{t.source}"
end

file SRC.ext("pdf") => SRC

desc "Compile PDF"
task :evince => SRC.ext("pdf")

desc "Show compiled PDF in Evince."
task :evince => :pdf do
  sh "evince #{SRC.ext("pdf")}"
end
```

RSpec - framework BDD

```ruby
require "portierka"

describe OndraszekPortierka do
  before do
    @portierka = OndraszekPortierka.new(:niebieski)
  end

  it "should call Straz.wezwij with dormitory name" do
    Straz.should_receive(:wezwij).with(:ondraszek).and_return(true)
    response = @portierka.wezwij_straz
    response.should == true
  end
end
```

RubyGems

- system paczek RubyGems
- zależności pomiędzy gemami
- łatwa aktualizacja gemów
- prawie 4000 gemów w repozytorium

Przykład

```
$ gem install rails
```

"I always thought Smalltalk would beat Java. I just didn't know it would be called 'Ruby' when it did."

Kent Beck

Ruby on Rails

- David Heinemeier Hansson, 2004 r.
- kompletny framework do tworzenia aplikacji internetowych opartych o bazy danych
- wzorzec MVC

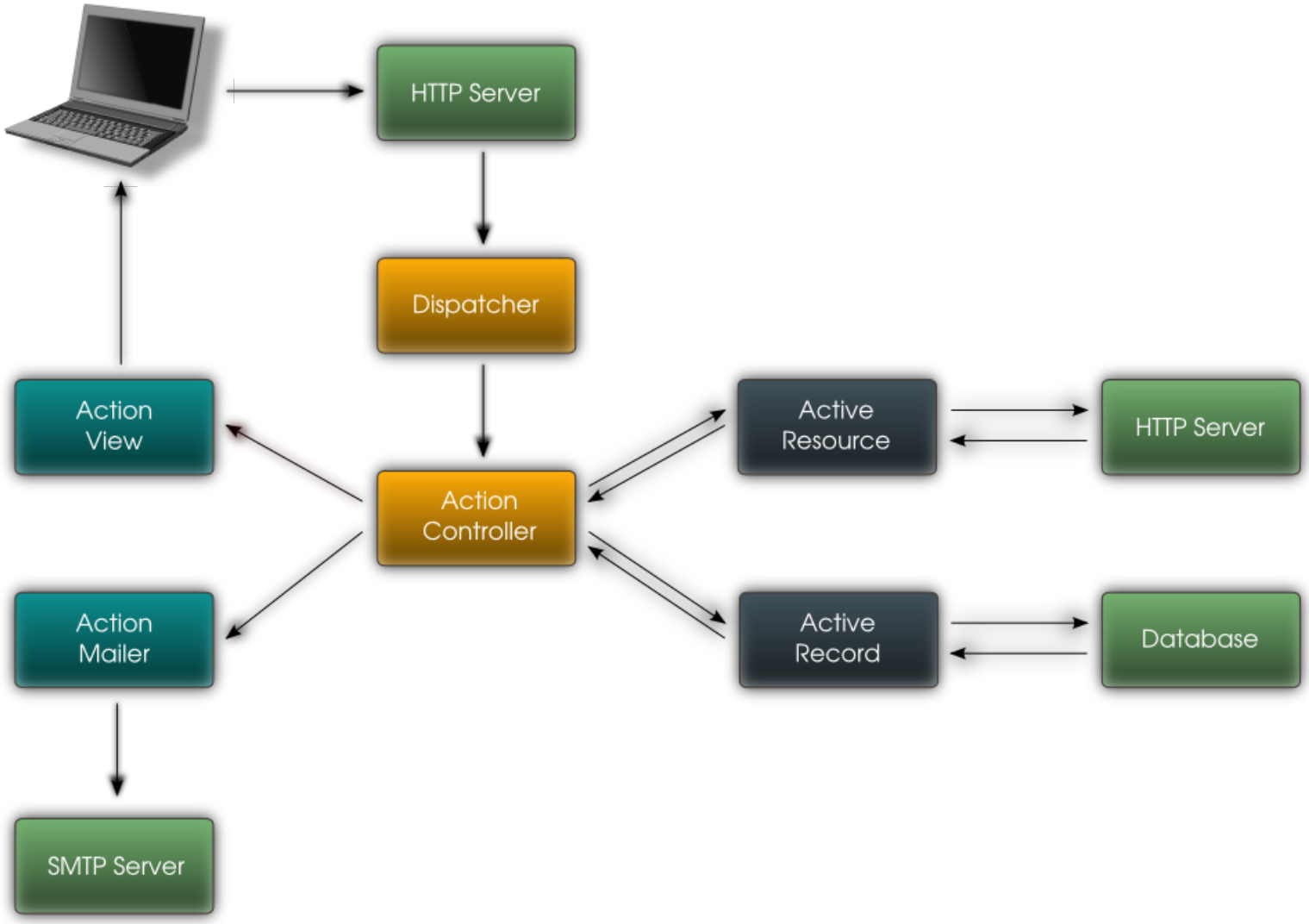
Rails way

- Convention over Configuration
- Don't Repeat Yourself

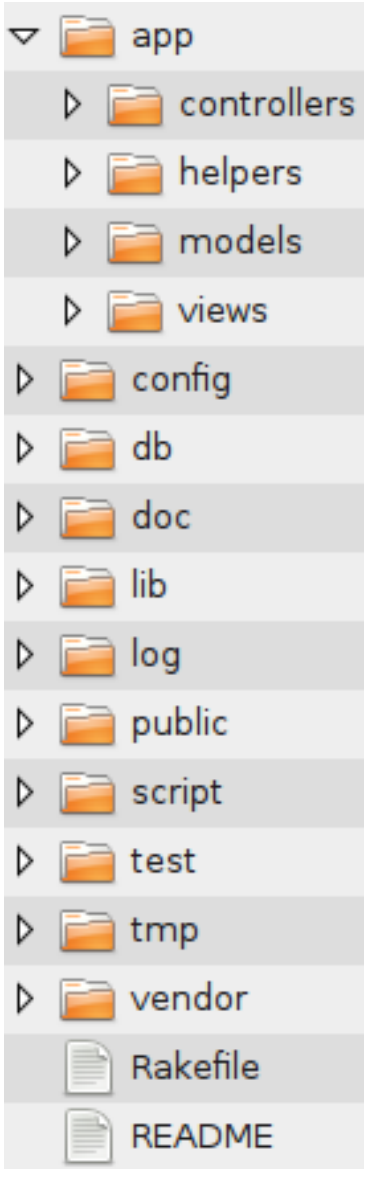
Z czego składa się Rails?

- ActiveRecord
- ActionPack
- ActiveResource
- ActionMailer
- ActiveSupport

Architektura Rails



Zaczynamy!




```
# gem install rails

$ rails myapp

$ cd myapp

$ ./script/server
```

ActiveRecord

- domyślny ORM dla Rails
- są też inne opcje (np. DataMapper)
- implementacja wzorca Active Record

ActiveRecord - co dostajemy?

- prosta konfiguracja
- migracje bazy danych
- proste tworzenie asocjacji
- zapytania
- walidatory
- wywołania zwrotne
- transakcje
- kilka innych rzeczy

ActiveRecord - szybki start

```
development:
  adapter: mysql
  database: demo
  username: admin
  password: password
  host: localhost

$ ./script/generate model post title:string content:text

$ rake db:migrate

class Post < ActiveRecord::Base
end

Post.create :title => "SRUG",
            :content => "Ruby on Rails"
```

ActiveRecord - migracje

- proste w użyciu wersjonowanie schematu bazy, historia zmian
- Ruby zamiast SQL-a
- praca w zespołach

ActiveRecord - migracje - przykład

```
class CreateStudents < ActiveRecord::Migration
  def self.up
    create_table :students do |t|
      t.string :name
      t.integer :beers_count, :null => false
      t.belongs_to :dormitory
      t.timestamps
    end
  end

  def self.down
    drop_table :students
  end
end
```

ActiveRecord - asocjacje

- mapowanie powiązań pomiędzy obiektami ActiveRecord
- wyrażają relację takie jak “user has many projects” czy “product belongs to category”
- oparte na metaprogramowaniu

ActiveRecord - asocjacje - przykład

```
class Post < ActiveRecord::Base
  belongs_to :category
  has_many :comments
end

class Category < ActiveRecord::Base
  has_many :posts
end

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

ActiveRecord - asocjacje - przykład, c.d.

```
category = Category.find_by_title "SRUG"
post = category.posts.first
post.comments.find_all_by_author "John Doe"
post.comments.create :content => "useful comment"
post.comments.last
post.comments.delete_all
```

ActiveRecord - zapytania

- Różne sposoby budowania zapytań
- Nazwane zapytania

ActiveRecord - przykłady zapytań

```ruby
Movie.first
# SELECT * FROM "movies" LIMIT 1

Book.last
# SELECT * FROM "books" ORDER BY books.id DESC LIMIT 1

Beer.find(1)
# SELECT * FROM "beers" WHERE ("beers"."id" = 1)

Subject.find([3, 5, 7])
# SELECT * FROM "subjects" WHERE ("subjects"."id" IN (3,5,7))

Student.all(:conditions => { :beer_count => 10..20 })
# SELECT * FROM "students" WHERE
#  ("students"."beer_count" BETWEEN '10' AND '20')
```

ActiveRecord - przykłady zapytań c.d.

```ruby
Post.all(:conditions => ["title LIKE ? ", "Rails"])
# SELECT * FROM "posts" WHERE (title LIKE 'Rails')

Post.first(:include => [:category, :comments])
# SELECT * FROM "posts" LIMIT 1
# SELECT * FROM "categories" WHERE ("categories"."id" IN (1))
# SELECT "comments".* FROM "comments" WHERE ("comments".post_id IN (1))

User.first(:conditions => { :login => "srug", :password => "secret" })
# SELECT * FROM "users" WHERE
#   ("users"."password" = 'secret' AND "users"."login" = 'srug')

User.first(:conditions => { :login => ["srug", "Srug", "SRUG"]})
# SELECT * FROM "users" WHERE
#   ("users"."login" IN ('srug','Srug','SRUG')) LIMIT 1
```

ActiveRecord - przykłady zapytań c.d.

```
Student.count
# SELECT count(*) AS count_all FROM "students"

Product.sum(:price)
# SELECT sum(price) AS sum_price FROM "products"

Visit.average("duration / 3600.0", :group_by => "day")
# SELECT sum(duration / 3600.0)
#   AS sum_duration_3600_0, day AS day
#   FROM "visits" GROUP BY day
```

Named scope - przykłady

```ruby
class Post < ActiveRecord::Base
  belongs_to :category

  named_scope :recent, lambda do
    { :conditions => ["created_at > ?", 2.hours.ago] }
  end
  named_scope :published,
              :conditions => { :published => true }
end

Post.recent
# SELECT * FROM "posts"
# WHERE (created_at > '2008-10-29 19:12:15')

Post.recent.published
# SELECT * FROM "posts"
# WHERE (("posts"."published" = 't') AND
#        (created_at > '2008-10-29 19:12:22'))
```

ActiveRecord - walidatory

- gwarantują poprawność wprowadzanych danych
- przeniesienie walidacji z poziomu bazy danych do modelu
- można walidować: format, długość, obecność, unikalność, powiązane obiekty, etc.
- łatwe tworzenie własnych walidatorów

ActiveRecord - walidatory - przykład

```
class Post < ActiveRecord::Base
  validates_presence_of      :title,
                             :content
  validates_uniqueness_of    :title,
                             :scope => :category_id
  validates_format_of        :title,
                             :with => /\A[\w\s]+\Z/
  validates_numericality_of  :rating,
                             :greater_than => 0
  validate                   :niceness_of_title

  def niceness_of_title
    errors.add :title, "is lousy" unless title.nice?
  end
end
```

ActiveRecord - wywołania zwrotne

- wyzwalanie logiki przed lub po zmianie stanu obiektu
- manipulacja atrybutami obiektu przed jego utworzeniem, zapisem, usunięciem lub walidacją
- przeniesienie logiki z kontrolera do modelu

ActiveRecord - callbacks - przykład

```ruby
class Post < ActiveRecord::Base
  before_validation :sanitize_title, :escape_content
  after_destroy :clear_category_cache

  def sanitize_title
    title.sanitize!
  end
  # ...
end
```

ActiveRecord - transakcje

- bloki kodu, w których gwarantowana jest atomowość wszystkich operacji na bazie danych
- różne modele w jednej transakcji

Przykład

```
transaction do
  david.withdrawal 100
  mary.deposit 100
end
```

ActiveRecord

- Single Table Inheritance
- asocjacje polimorficzne
- optimistic locking
- acts_as: state_machine, taggable, nested_set, commentable, dictionary, geocodable

"I have never seen an Active Record implementation as complete or as useful as Rails"

Martin Fowler, software architect

ActionPack

- podział odpowiedzi aplikacji na dwie części:
  - ActionController
  - ActionView

ActionPack::ActionController

- request zostaje skierowany do odpowiedniej akcji (routing)
- akcja zwraca odpowiedź do przeglądarki
  - wyrenderowany widok
  - przekierowanie
  - błąd
- cookies
- sesje
- flash
- filtry

ActionController::Routing

- wiązanie URI z akcjami odpowiednich kontrolerów

Przykład

```
ActionController::Routing::Routes.draw do |map|
  map.resources :beers
end
```

ActionPack::ActionController

```
class BeersController < ApplicationController
  def index
    @beers= Beer.all
  end
end
```

ActionPack::ActionView

- szablony w widokach: ERb, Haml, Liquid i inne
- partiale
- helpery

ActionPack::ActionView - index.html.erb

```
<table class='beers' id='important'>
  <tr class='header'>
    <th>Name</th>
    <th>Type</th>
  <%- @beers.each do |beer| %>
  <tr>
    <td><%= beer.name %></td>
    <td><%= beer.type %></td>
  </tr>
  <%- end %>
</table>
```

ActionPack::ActionView - index.html.haml

```
%table#beers.important
  %tr.header
    %th Name
    %th Type
  - @beers.each do |beer|
    %tr
      %td= beer.name
      %td= beer.type
```

```
<table class='important' id='beers'>
  <tr class='header'>
    <th>Name</th>
    <th>Type</th>
  </tr>
  <tr>
    <td>Pilsner Urquell</td>
    <td>pilsner</td>
  </tr>
  <tr>
    <td>Żywiec Porter</td>
    <td>porter</td>
  </tr>
</table>
```

ActiveSupport

```
40.minutes.ago
#=> 2008-10-27 20:29:37 +0100
20.weeks.from_now
#=> 2009-03-16 21:10:45 +0100
Time.now.at_beginning_of_year
#=> Tue Jan 01 00:00:00 +0100 2008
5.gigabytes
#=> 5368709120
"good beer".titleize.pluralize
#=> "Good Beers"
["Pilsner Urquell", "Оболонь", "Paulaner"].to_sentence
#=> "Pilsner Urquell, Оболонь and Paulaner"
```

Serwery HTTP

- Webrick
- Mongrel, Thin, Ebb
- Passenger (Apache 2)

Przykłady wdrożeń

- Twitter
- Yellowpages
- Basecamp
- GitHub
- Shopify
- Slideshare

Jak zacząć?

- instant-rails
- github.com
- opensourcerails.com
- railscasts.com
- heroku.com

"Rails is the most well thought-out web development framework I've ever used. (...) Nobody has done it like this before."

James Duncan Davidson, Creator of Tomcat and Ant

Dziękujemy za wytrwałość!