Ruby on Rails

Alexandre Takinami
Bruno Yoshimura
Marcelo de Rezende Martins
Marcio Vinicius dos Santos

Ruby é ...

dinâmica, open source com foco na simplicidade e na produtividade.

Os ideais do criador

- Yukihiro Matsumoto "matz"

- "trying to make Ruby natural, not simple,"

- Satisfação do desenvolvedor

Ruby's architect, Yukihiro Matsumoto

Em Ruby
Tudo é um objeto

```
5.times {
print "Tudo é  objeto"
 }
```

Em Ruby

Tudo é extensível

```
class Numeric
  def plus(x)
    self.+(x)
  end
end

y = 5.plus 6
# y é agora igual a 11
```

Em Ruby

Blocos aumentam flexibilidade

```
search_engines =
  %w[Google Yahoo MSN].map do |engine|
    "http://www." + engine.downcase + ".com"
  end
```

Em Ruby

Blocos aumentam flexibilidade

```
def calc(a, b)
    yield a, b
end

calc(3,4){ |x,y|   x*y}
```

Em Ruby

Módulos resolvem herança multipla

```ruby
module Helloworld
  def say
    puts "Hello world"
  end
end

class Mixin
  include Helloworld
end
```

Aprenda Ruby em

- http://www.ruby-lang.org/en/
- http://tryruby.hobix.com/
- http://www.ruby-lang.org/en/documentation/quickstart/

Rails é ...

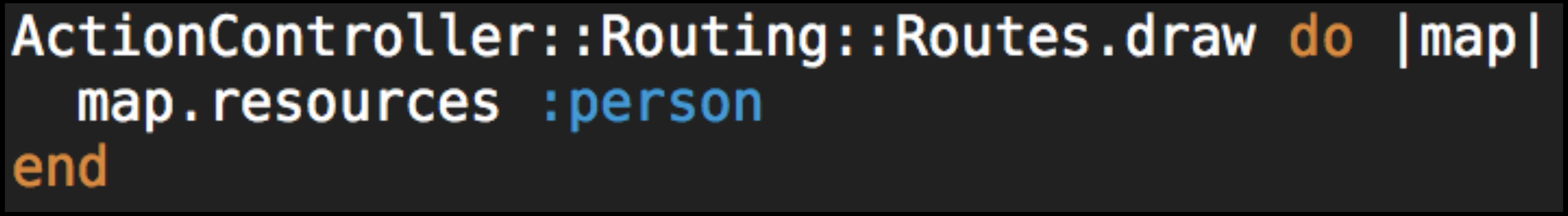
“framework web open-source voltado para a felicidade do programador e produtividade sustentável.”

Rails

Extende o Ruby através de Domain S pecific L anguage s

```
class Project < ActiveRecord::Base
  belongs_to                :portfolio
  has_one                   :project_manager
  has_many                  :milestones
  has_and_belongs_to_many   :categories

  validates_presence_of     :name, :description
  validates_acceptance_of   :non_disclosure_agreement
  validates_uniqueness_of   :key
end
```

Rails é
REST



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

Rails é
REST

```
POST /people
GET /people/1
PUT /people/1
DELETE /people/1
```

Rails é
REST + CRUD

GET	POST	PUT	DELETE
find	create	update	destroy
SELECT	INSERT	UPDATE	DELETE

Rails é
Convention over Configuration

```ruby
class Project < ActiveRecord::Base
  set_table_name  "projects"
  set_primary_key "id"

  belongs_to :account,    :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  set_table_name  "milestones"
  set_primary_key "id"

  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

Rails é
Convention over Configuration

```
class Project < ActiveRecord::Base
  set_table_name  "projects"
  set_primary_key "id"

  belongs_to :account,     :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  set_table_name  "milestones"
  set_primary_key "id"

  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

Rails é
Convention over Configuration

```
class Project < ActiveRecord::Base
  set_primary_key "id"

  belongs_to :account,     :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  set_primary_key "id"

  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

Rails é
Convention over Configuration

```ruby
class Project < ActiveRecord::Base
  set_primary_key "id"

  belongs_to :account,     :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  set_primary_key "id"

  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

Rails é
Convention over Configuration

```ruby
class Project < ActiveRecord::Base
  belongs_to :account,    :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

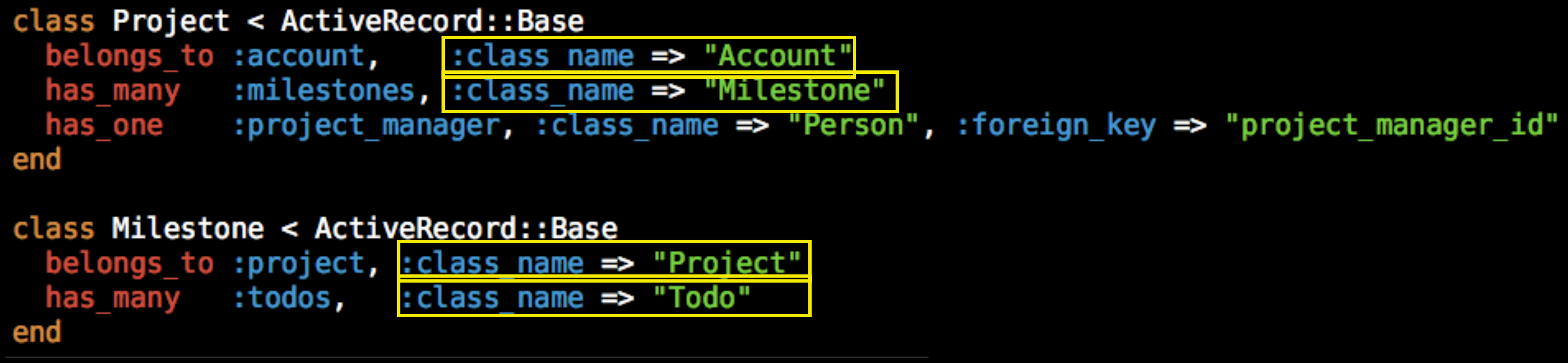
Rails é
Convention over Configuration

```
class Project < ActiveRecord::Base
  belongs_to :account,    :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

Rails é
Convention over Configuration

```ruby
class Project < ActiveRecord::Base
  belongs_to :account,     :class_name => "Account"
  has_many   :milestones, :class_name => "Milestone"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  belongs_to :project, :class_name => "Project"
  has_many   :todos,   :class_name => "Todo"
end
```

Rails é
Convention over Configuration



```
class Project < ActiveRecord::Base
  belongs_to :account,     :class_name => "Account"
  has_many   :milestones, :class_name => "Milestone"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  belongs_to :project, :class_name => "Project"
  has_many   :todos,   :class_name => "Todo"
end
```

Rails é
Convention over Configuration

```
class Project < ActiveRecord::Base
  belongs_to :account
  has_many   :milestones
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  belongs_to :project
  has_many   :todos
end
```

Rails é
Convention over Configuration

```
class Project < ActiveRecord::Base
  set_table_name  "projects"
  set_primary_key "id"

  belongs_to :account,     :class_name => "Account",   :foreign_key => "account_id"
  has_many   :milestones, :class_name => "Milestone", :foreign_key => "milestone_id"
  has_one    :project_manager, :class_name => "Person", :foreign_key => "project_manager_id"
end

class Milestone < ActiveRecord::Base
  set_table_name  "milestones"
  set_primary_key "id"

  belongs_to :project, :class_name => "Project", :foreign_key => "project_id"
  has_many   :todos,   :class_name => "Todo",    :foreign_key => "todo_id"
end
```

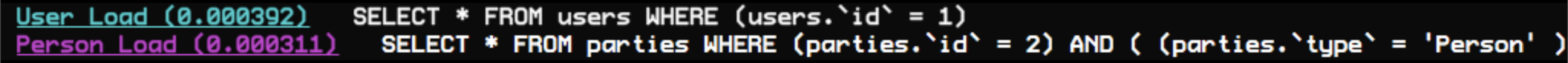
Convention + REST

```
class PostsController < ApplicationController
  # GET /posts
  # GET /posts.xml
  def index end

  # GET /posts/1
  # GET /posts/1.xml
  def show end

  # GET /posts/1/edit
  def edit end

  # POST /posts
  # POST /posts.xml
  def create end

  # PUT /posts/1
  # PUT /posts/1.xml
  def update end

  # DELETE /posts/1
  # DELETE /posts/1.xml
  def destroy end
end
```

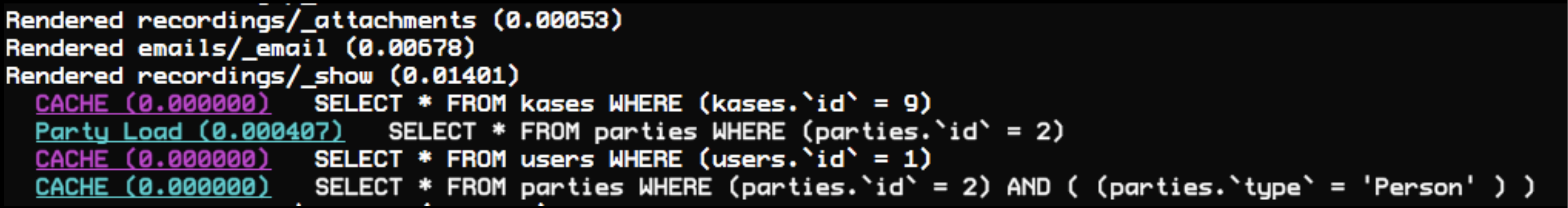
Rails é
Don't Repeat Yourself + Model View Control

```
$ ./script/generate scaffold Post title:string body:text
      exists  app/models/
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/posts
      exists  test/functional/
      exists  test/unit/
      create  app/views/posts/index.html.erb
      create  app/views/posts/show.html.erb
      create  app/views/posts/new.html.erb
      create  app/views/posts/edit.html.erb
      create  app/controllers/posts_controller.rb
      create  test/functional/posts_controller_test.rb
      create  app/helpers/posts_helper.rb
      create  test/unit/helpers/posts_helper_test.rb
       route  map.resources :posts
      exists    test/unit/
      create    app/models/post.rb
      create    test/unit/post_test.rb
      create    test/fixtures/posts.yml
      exists    db/migrate
      create    db/migrate/20090504220328_create_posts.rb
```

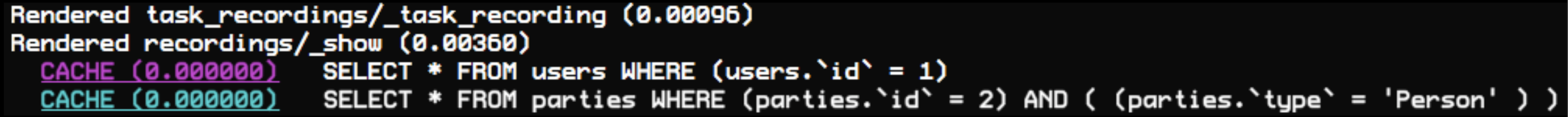
Rails tem
Query Cache



```
User Load (0.000392)   SELECT * FROM users WHERE (users.`id` = 1)
Person Load (0.000311)   SELECT * FROM parties WHERE (parties.`id` = 2) AND ( (parties.`type` = 'Person' )
```



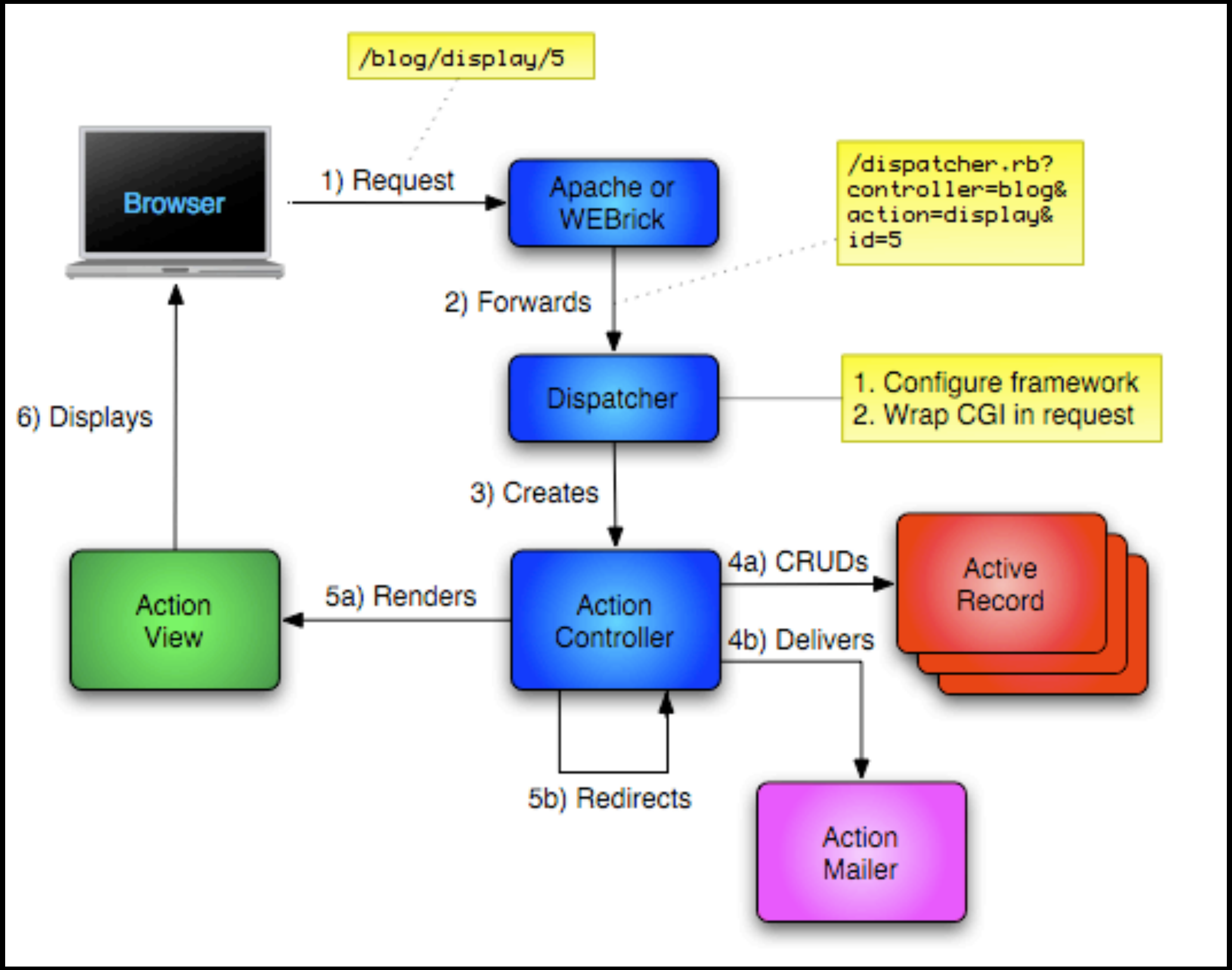
```
Rendered recordings/_attachments (0.00053)
Rendered emails/_email (0.00678)
Rendered recordings/_show (0.01401)
  CACHE (0.000000)   SELECT * FROM kases WHERE (kases.`id` = 9)
  Party Load (0.000407)   SELECT * FROM parties WHERE (parties.`id` = 2)
  CACHE (0.000000)   SELECT * FROM users WHERE (users.`id` = 1)
  CACHE (0.000000)   SELECT * FROM parties WHERE (parties.`id` = 2) AND ( (parties.`type` = 'Person' ) )
```



```
Rendered task_recordings/_task_recording (0.00096)
Rendered recordings/_show (0.00360)
  CACHE (0.000000)   SELECT * FROM users WHERE (users.`id` = 1)
  CACHE (0.000000)   SELECT * FROM parties WHERE (parties.`id` = 2) AND ( (parties.`type` = 'Person' ) )
```

```ruby
ActiveRecord::Base.cache do
  # SELECTs are cached until an INSERT, UPDATE, or DELETE happen
end
```

Rails tem
Simplicidade



ActionController

Core das requisições web no Rails

```ruby
class GuestBookController < ActionController::Base
  def index
    @entries = Entry.find(:all)
  end

  def sign
    Entry.create(params[:entry])
    redirect_to :action => "index"
  end
end
```

ActionView

EmbeddedRuBy - Ruby Templating

```
<% for post in @posts %>
  Title: <%= post.title %>
<% end %>

All post titles: <%= @posts.collect{ |p| p.title }.join ", " %>

<% unless @person.is_client? %>
  Not for clients to see...
<% end %>
```

ActiveRecord

ObjectRelationalMapping Pattern

"An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data." Martin Fowler

```
class Post < ActiveRecord::Base
end
```

Refaça seu projeto em Rails!

JAVA IN A NUTSHELL
JBoss 4.0 THE OFFICIAL GUIDE
Patterns of Enterprise Application Architecture
XDoclet IN ACTION
Java Web Services
Java Enterprise Best Practices
core J2EE Patterns
JUnit IN ACTION
Programming Jakarta Struts
Jakarta Struts Cookbook
Java Servlet & JSP Cookbook
Java and XSLT
Java Foundation Classes IN A NUTSHELL
JavaServer Pages
HIBERNATE IN ACTION
STRUTS IN ACTION
Programming Ruby
Agile Web Development with Rails