Ruby on Rails para iniciantes

Ruby on Rails para iniciantes

mágico

O que significa esse nome estranho?

Ruby

Linguagem

Ruby

Linguagem

Rails

Framework

Começando por Ruby

Inventado em 1995 por Yukihiro Matsumoto, mais conhecido como Matz.

Começando por Ruby

Inventado em 1995 por Yukihiro Matsumoto, mais conhecido como Matz.

Matz queria que Ruby fosse mais poderoso que Perl e mais orientada a objetos que Phyton **e mágico**.

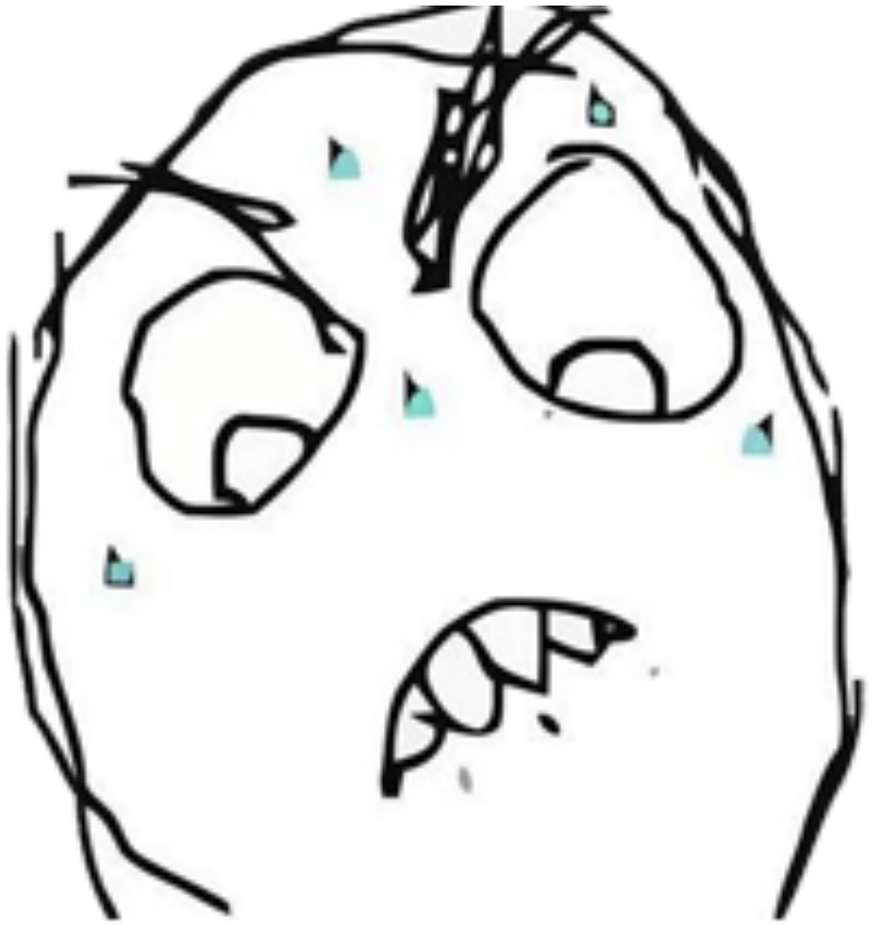
Instalação

http://ruby-lang.org/pt

Características

- Completamente orientado a objetos até mesmo os tipos primitivos

O que é tipo primitivo mesmo...?



Inteiro/Integer:
102

Inteiro/Integer:
102

Cadeia de caracteres/String:
“Ola mundo”

Inteiro/Integer:
102

Cadeia de caracteres/String:
“Ola mundo”

Arranjo/Array:
[1, 2, 3]

Inteiro/Integer:
102

Cadeia de caracteres/String:
“Ola mundo”

Arranjo/Array:
[1, 2, 3]

Entre outros

Lembrei, mas e a orientação a objetos?

```
1.9.3-p286 :010 > 101.class
 => Fixnum
```

Ou seja, 123 é um número da classe Fixnum

Variáveis são **mágicas**, elas possuem métodos.

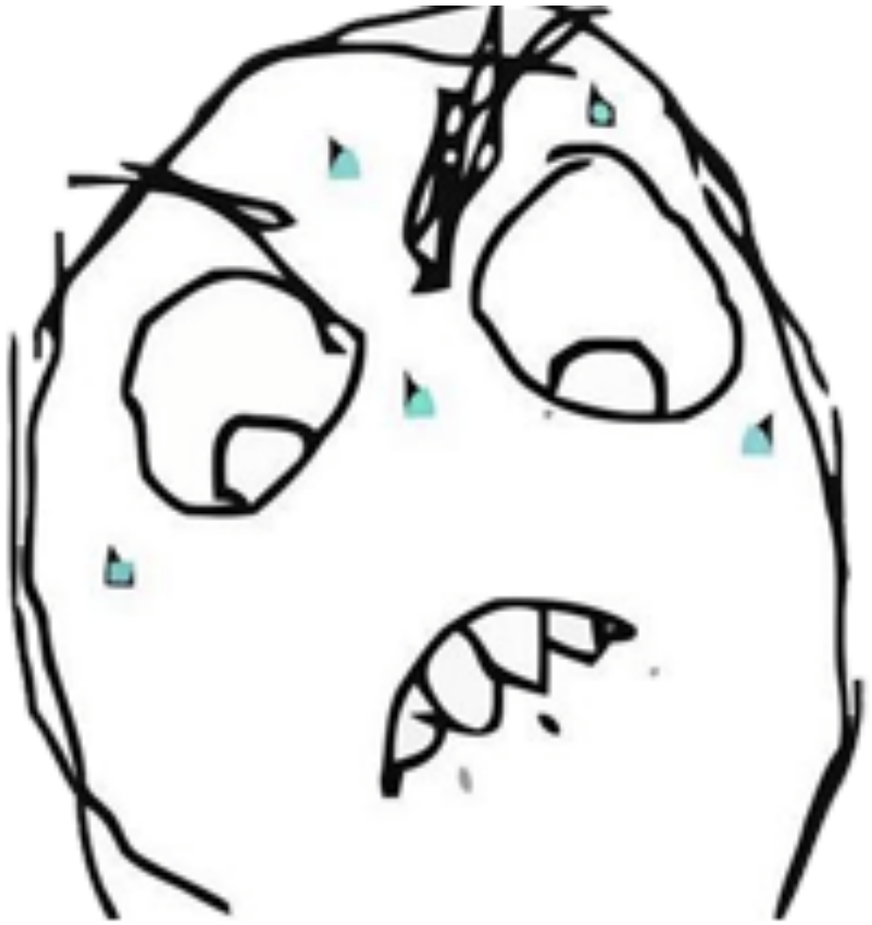
Exemplo:

```
1.9.3-p286 :011 > 101.methods
 => [:to_s, :-@, :+, :-, :*, :/, :div, :
%, :modulo, :divmod, :fdiv, :**, :abs, :magnitude, :==,
:===, :<=>, :>, :>=, :<, :<=, :~, :&, :|, :^, :
[], :<<, :>>, :to_f, :size, :zero?, :odd?, :even?, :succ,
:integer? ...
```

Características

- Completamente orientado a objetos até mesmo os tipos primitivos
- Tipagem dinâmica e forte

O que é Tipagem dinâmica e tipagem forte?



Tipagem dinâmica

idade = 13

nome = “Joaozinho”

peso = 50.0

Tipagem dinâmica

idade = 13

nome = “Joaozinho”

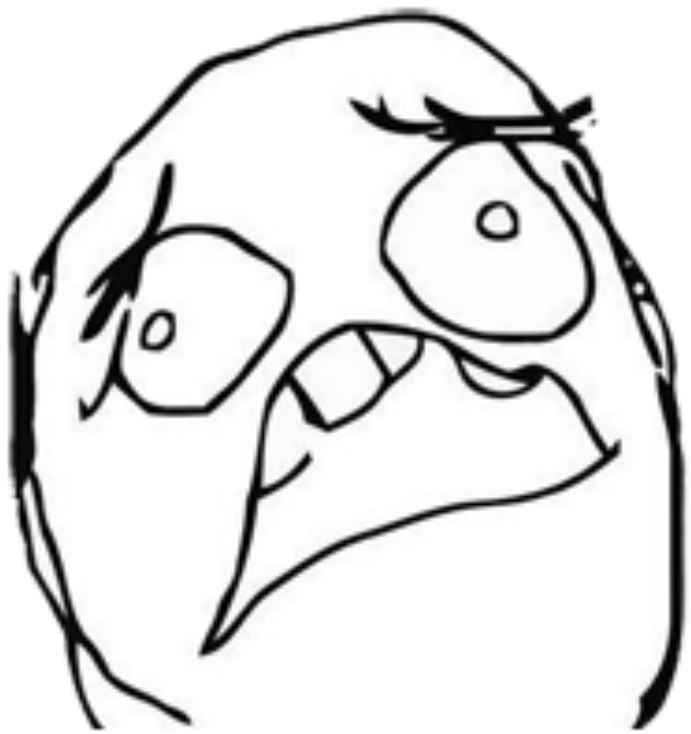
peso = 50.0

Ou seja, o ruby **magicamente** descobre o tipo da variável pra pra você.

Isso é chamado de tipagem dinâmica

Tipagem forte

Significa que é preciso converter os tipos explicitamente caso queira realizar alguma operação com as variáveis.



Exemplo

```
idade = 13
nome = “Joaozinho”

print nome + “,“ + idade + “ anos.”
```

A saída esperada era:
“Joaozinho, 13 anos.”

Exemplo

```
idade = 13
nome = “Joaozinho”

print(nome + “,“ + idade + “ anos.”)
```

TypeError: can't convert Fixnum into String

Exemplo Corrigido

```
idade = 13
nome = “Joaozinho”

print(nome + “,“ + idade.to_s + “ anos.”)
```

to_s é um método que converte uma variável(idade, no caso) para o tipo String

Por que não é toString(idade)?

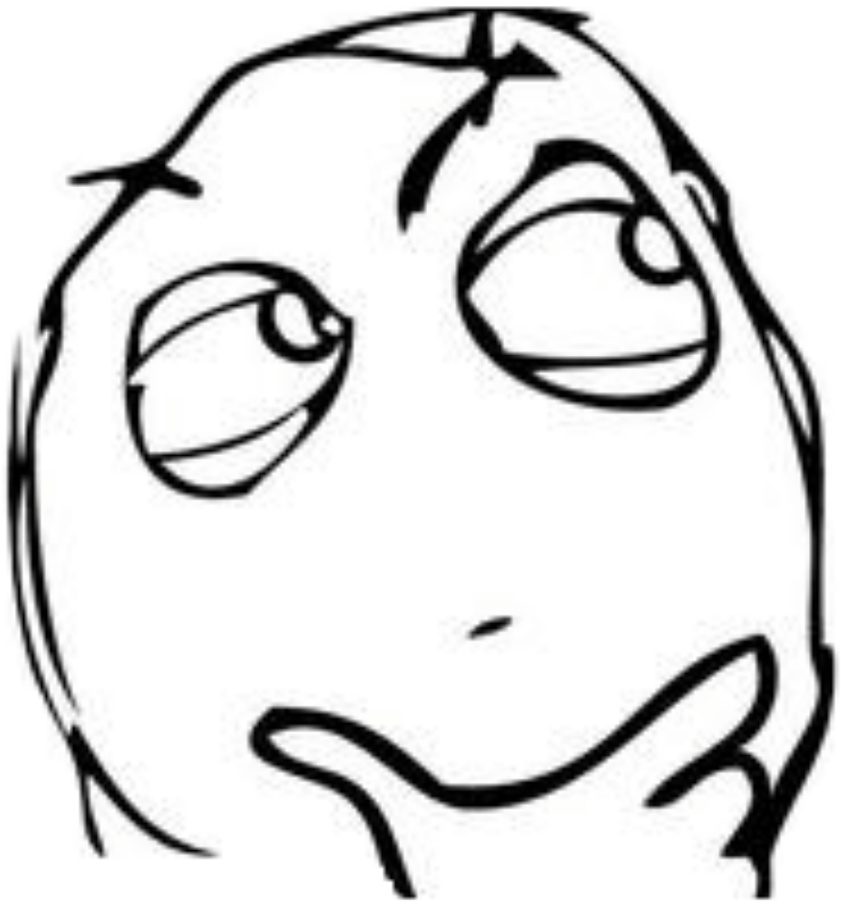
(ou derivados)

No Ruby, **os tipos são objetos e possuem métodos**, lembra?

Características

- Completamente orientado a objetos até mesmo os tipos primitivos(números inteiros, strings etc)
- Tipagem dinâmica e forte
- Sistema de pacotes (RubyGems)

E o que faço com esse sistema de pacotes ai?



Instale pacotes, como:

- O próprio Rails!
- Devise: faz toda a autenticação da sua web app para você.
- RMagick: te ajuda a converter imagens, adicionar marca d'agua etc.
- E infinitamente mais... Ruby tem um ecossistema imenso de soluções prontas e elegantes

Como?

no terminal

gem install rails
gem install devise
gem install rmagick

...

O RubyGems lida **magicamente** com as dependências dos pacotes.

Características

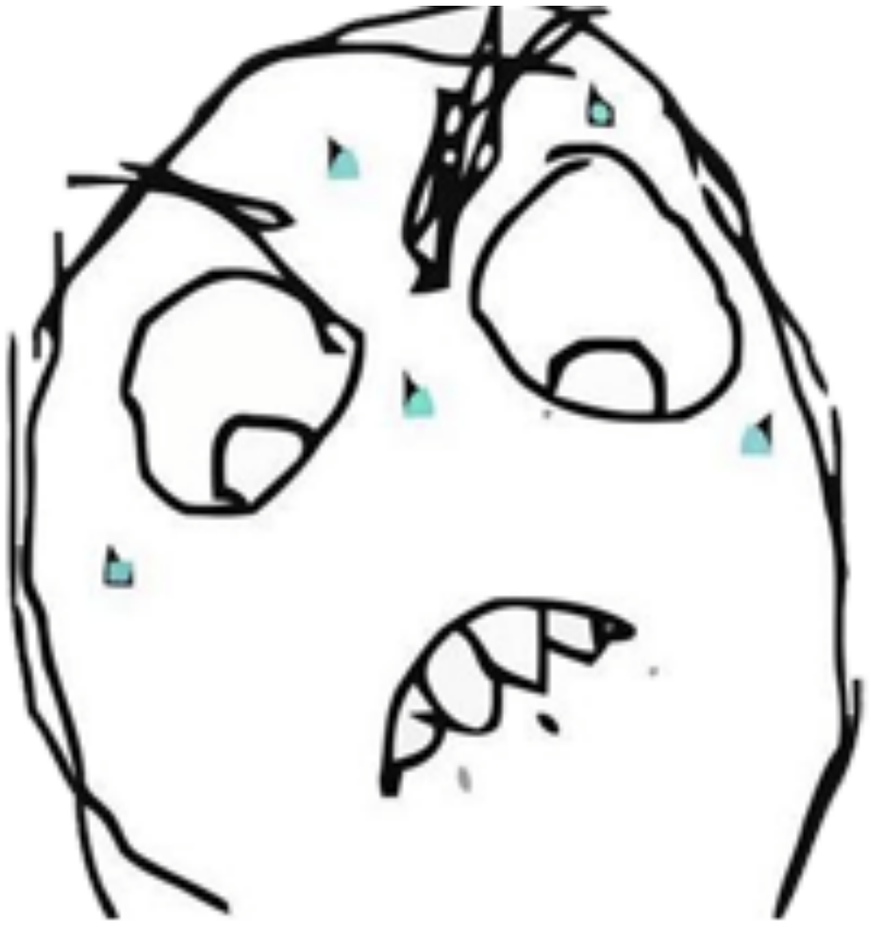
- Completamente orientado a objetos até mesmo os tipos primitivos(números inteiros, strings etc)
- Tipagem dinâmica e forte
- Sistema de pacotes (RubyGems)
- Disponível para diversos Sistemas Operacionais

Características

- Completamente orientado a objetos até mesmo os tipos primitivos(números inteiros, strings etc)
- Tipagem dinâmica e forte
- Sistema de pacotes (RubyGems)
- Disponível para diversos Sistemas Operacionais
- **Mágico** e interpretado

Que é mágico eu já entendi, mas e

interpretado?



Linguagem compilada

O código executado, após o processo de compilação, é o **código de máquina**.

Linguagem interpretada

O código executado, após o processo de compilação, é executado por uma **máquina virtual(VM)** ou **interpretador**.

Ok, e o que isso muda?

- Linguagens interpretadas são **BEM** mais lerdas na execução.
- Código pode ser executado independente de plataforma.
- Reflexão: o programa pode observar ou modificar sua própria estrutura e funcionamento.

Então minha aplicação web em **Rails** será lerda?

Não!

O gargalo de toda aplicação web geralmente está no **IO.** Por exemplo, na espera de uma consulta em um banco de dados.

Ok, gostei desse Ruby. Por onde começo?

Livro do TaQ:

http://eustaquiorangel.com/tutorial-ruby

código

Saindo um pouco do Ruby....


RAILS

Framework

Conjunto de classes implementadas que auxiliam no desenvolvimento de software para algum domínio específico.

Diferentemente de uma biblioteca, um framework dita o fluxo de controle da aplicação (Inversão de controle).

“The Rails Way”

- Convention over configuration(CoC)
- Active Record Pattern
- Don’t Repeat Yourself(DRY)
- MVC

Convention over configuration

Active Record Pattern

- Padrão de projeto "descoberto" por Martin Fowler
- Uma tabela do banco de dados é mapeada por uma classe/objeto(Object-relational mapping).
- Uma instância dessa classe representa uma linha da tabela.

Vantagens

- Não deixa o código da sua aplicação totalmente presa a um banco de dados
- Geralmente não te obriga a pensar na query bizarra que vai escrever, e sim no objeto que quer.
- Produtividade!

Ok, chega do blalabla e me mostra.

Exemplo

Quero os locais(places) cadastrados por um usuário(user) de id 1.

SQL:

```
SELECT "users".* FROM "users" WHERE "users"."id" = $1 LIMIT 1  [["id", 1]]
SELECT "places".* FROM "places" WHERE "places"."user_id" = 1
```

Usando o ActiveRecord:

```
User.find(1).places
```

Exemplo

Quero os locais(places) cadastrados por um usuário(user) de id 1.

SQL:

SELECT "user[illegible]* FRO[illegible]'us[illegible]VH[illegible]us[illegible]" = $[illegible]LIM[illegible] 1 [["id", 1]]

SELECT "places".* FRO[illegible]"places" VH[illegible]"places".user_id[illegible] 1

Mágico

Usando o ActiveRecord:

User.find(1).places

Exemplo

Quero criar um novo lugar(place)

SQL:
INSERT INTO “places” (“name”) VALUES (“Casa dos Artistas”)

ActiveRecord:
Place.create(name: 'Casa dos Artistas')

Exemplo

Quero criar um novo lugar(place)

SQL:
INSERT INTO “[illegible]

ActiveRecord:
Place.create(name: 'Casa dos Artistas')

Mágico

Don’t Repeat Yourself (DRY)

Não repita a si mesmo.

Evite repetições, reuse o código.

No Rails, é incomum a necessidade de duplicar arquivos de configuração. O **ActiveRecord** funciona sem necessidade alguma de “inicialização”.

DRY e Ruby

O próprio Ruby está enraizado em princípios DRY.

Tanto na sintaxe, como em conceitos como **metaprogramação**.

No Java:
Point point = new Point(23, 94);

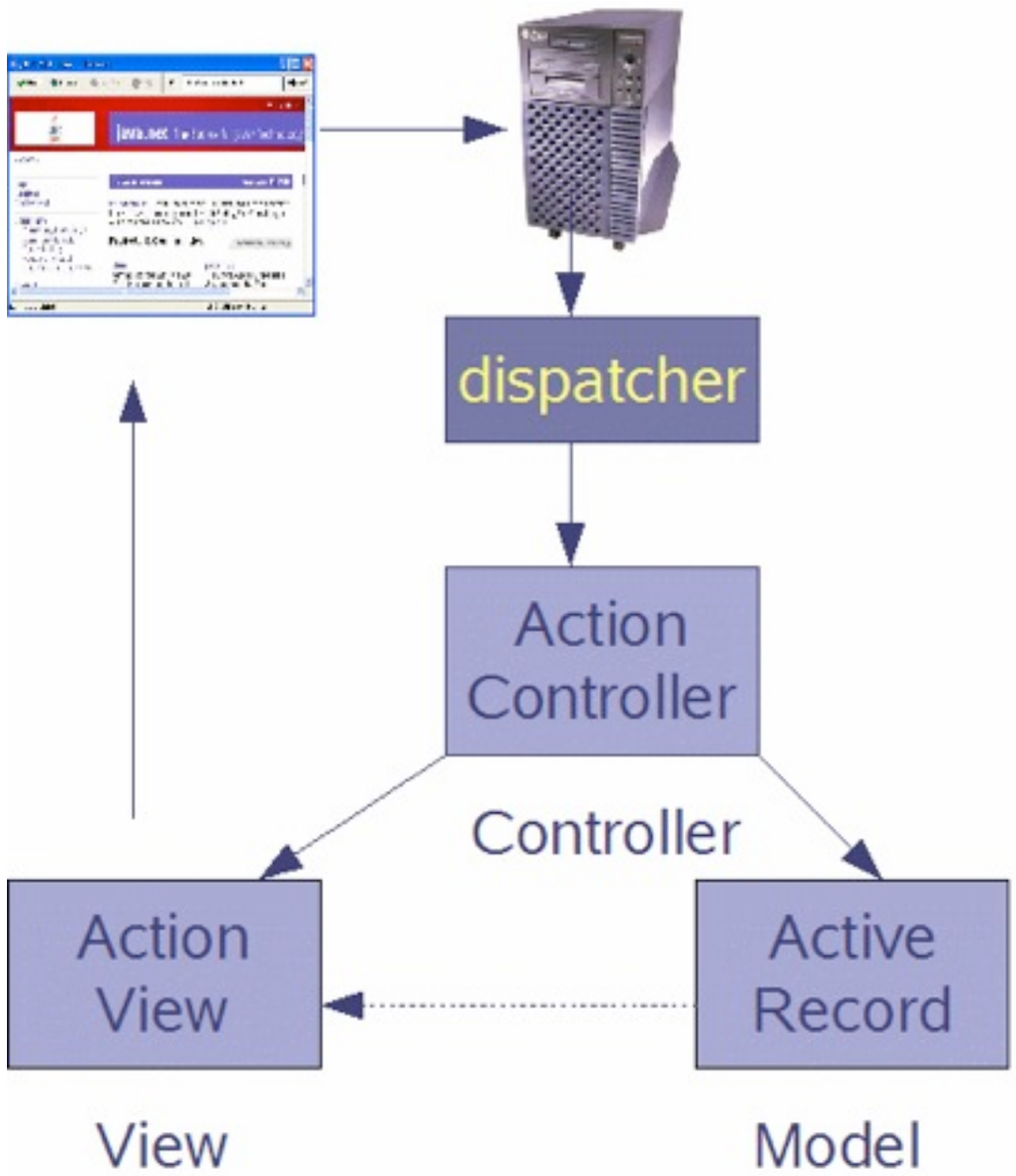
No Ruby:
point = Point.new

MVC

É uma arquitetura que isola a lógica da aplicação da interface com o usuário.

Com uma definição de onde cada tipo de código pertence, a manutenção é facilitada.

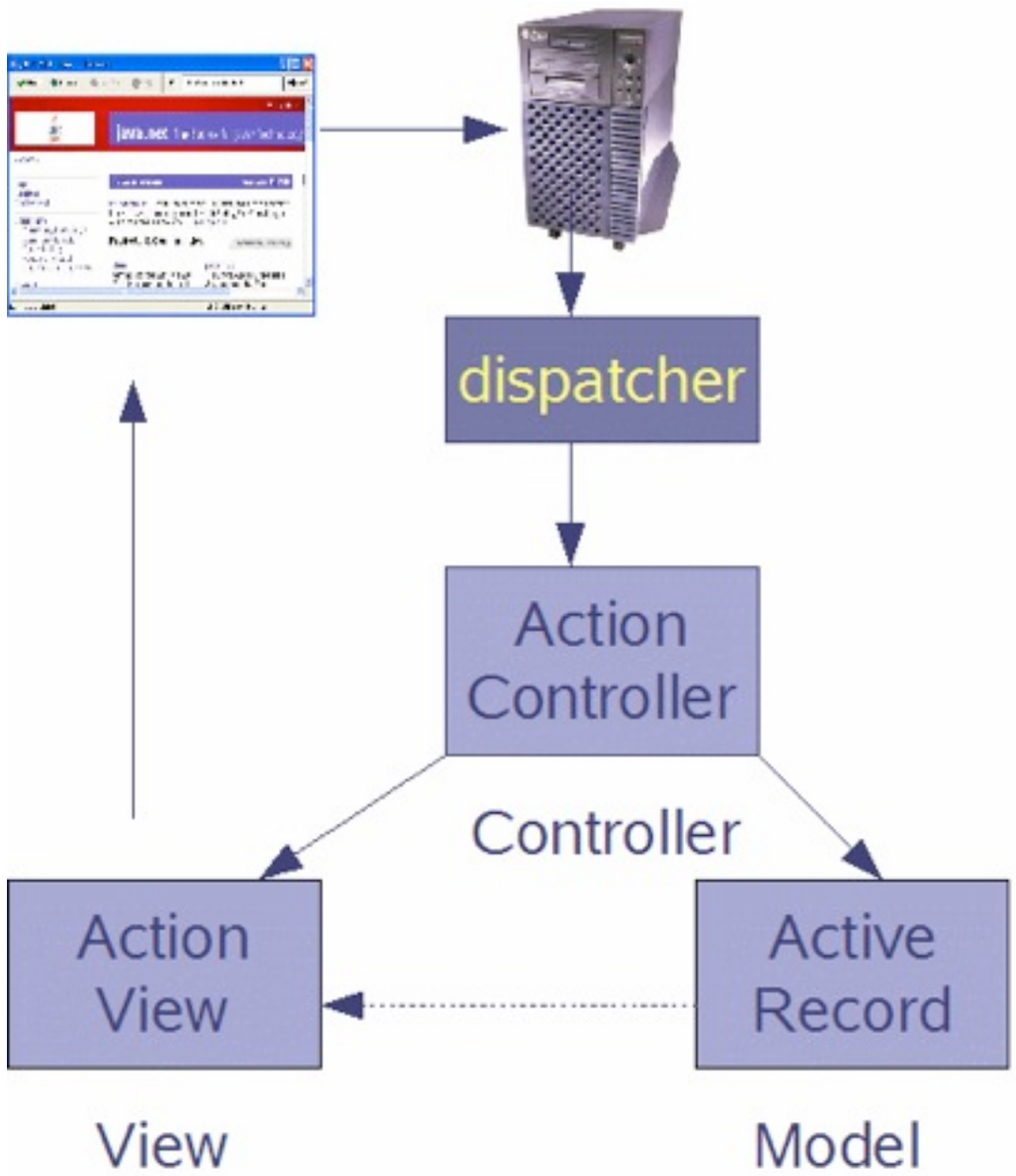
MVC


dispatcher
Action Controller
Controller
Action View
Active Record
View
Model

Model

É uma representação dos dados da aplicação, no Rails é utilizado para conversar com o banco de dados.

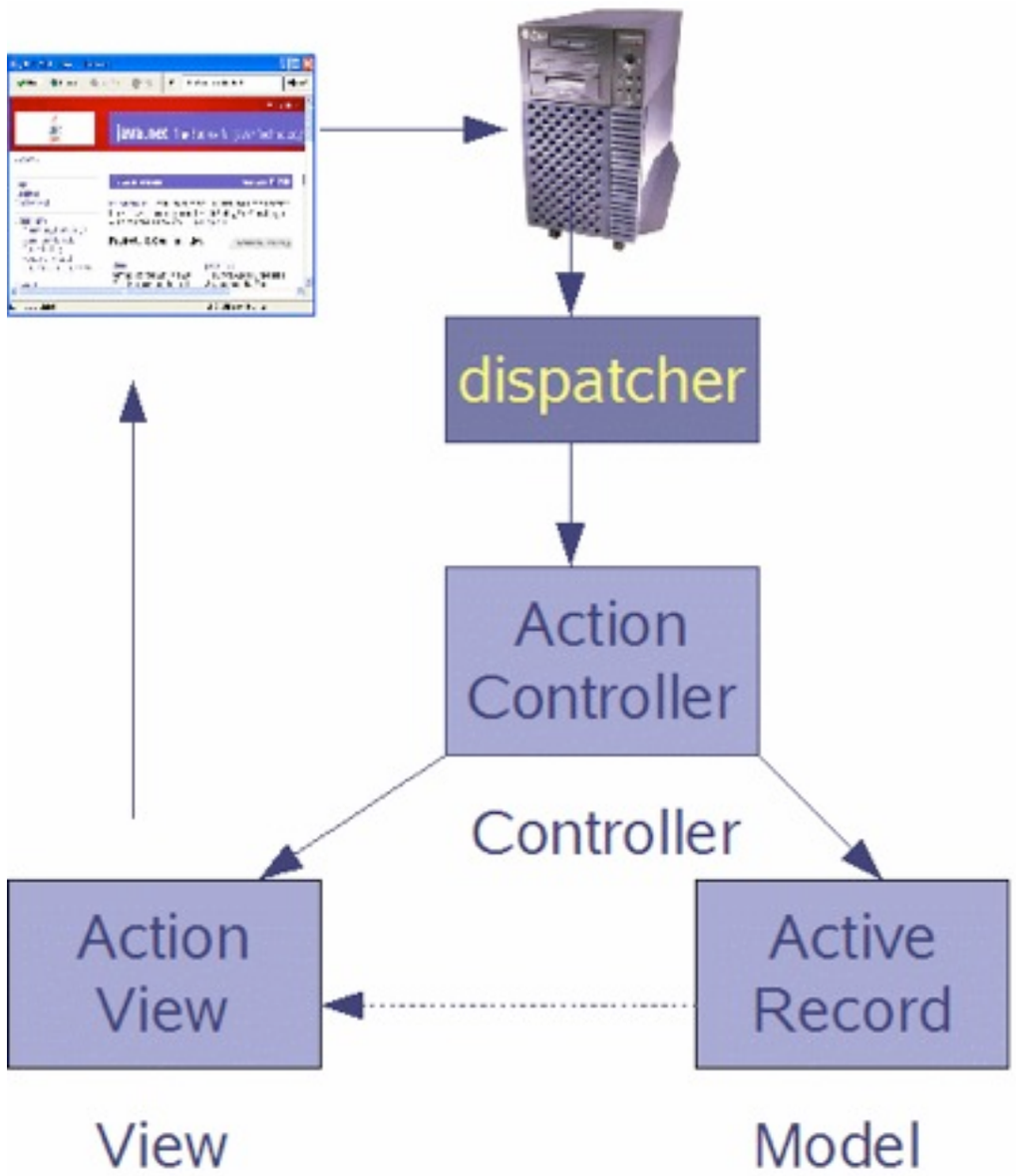
MVC


dispatcher
Action Controller
Controller
Action View
Active Record
View
Model

Views

É responsável pela interface da sua aplicação. É o HTML com código embedado Ruby (erb).

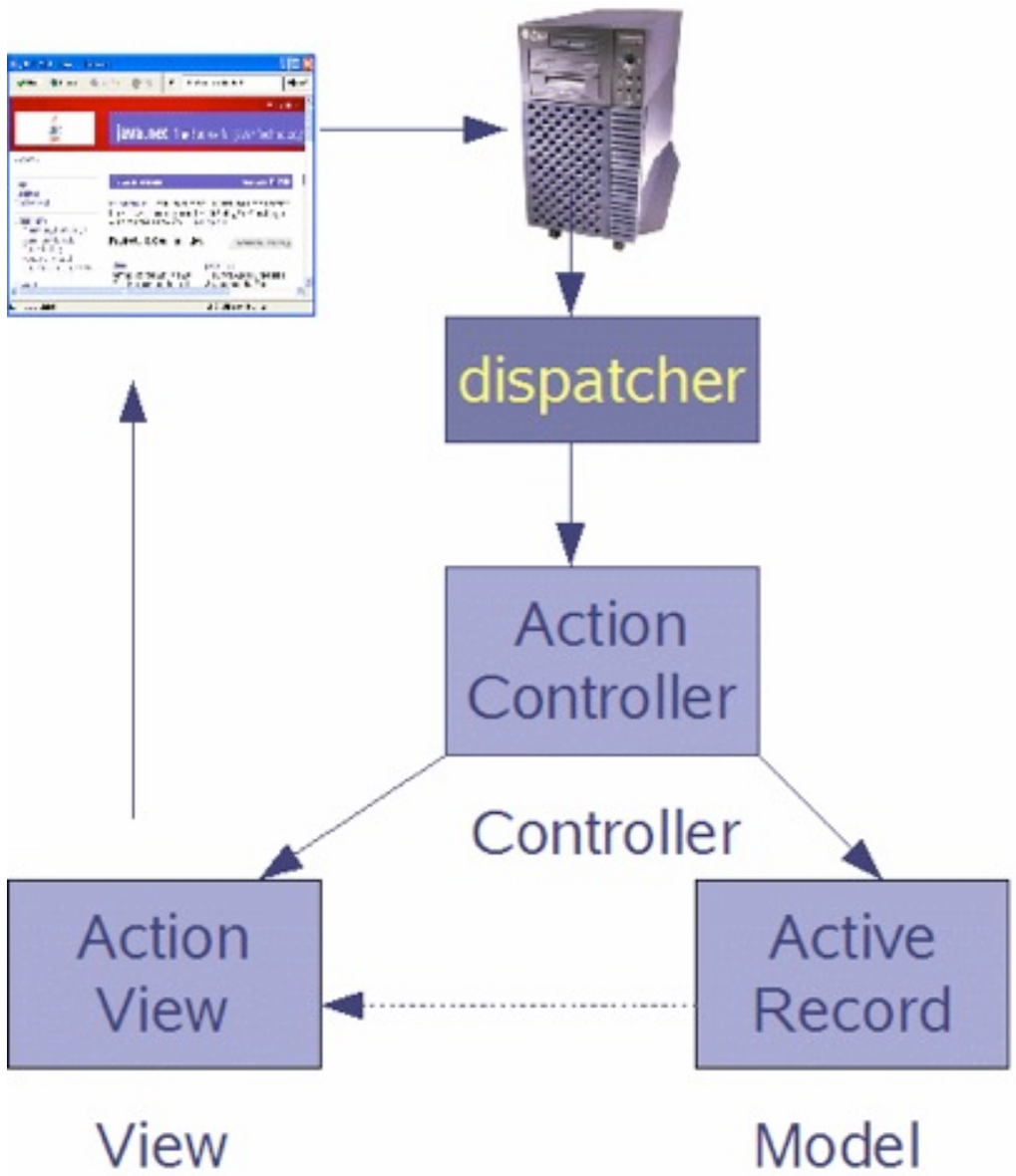
MVC


dispatcher
Action Controller
Controller
Action View
Active Record
View
Model

Controller

É o responsável por receber uma requisição, pedir ao Model algum dado e mostrar/renderizar a view.

MVC


dispatcher
Action Controller
Controller
Action View
Active Record
View
Model

Mercado

código

Obrigado!

Contato:
twitter: @th1agofm
e-mail: thiagown@gmail.com