

Ruby on Rails

DACA 2008.1

Felipe Ribeiro - felipernb@gmail.com



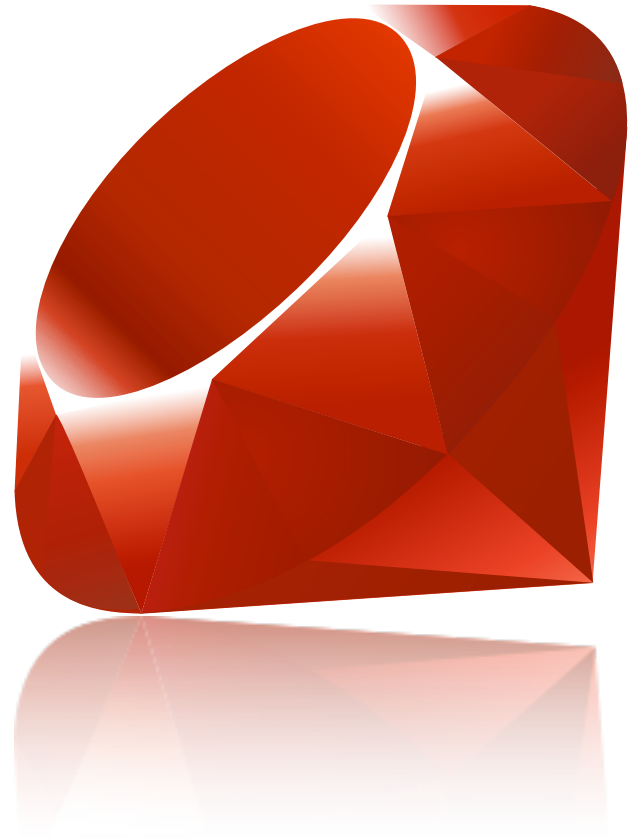
Ruby on Rails

linguagem de
programação

framework de
desenvolvimento web

Ruby é...

Uma linguagem dinâmica, open source com foco na simplicidade e na produtividade. Tem uma sintaxe elegante de leitura natural e fácil escrita.



Características do Ruby

- Orientado a objeto
- Tudo é um objeto, não há tipos primitivos.
- Herança única, com extensão por módulos
- Possibilidade de adicionar programação em runtime
- Traços de orientação a aspectos
- Otimizada para pessoas e não para máquinas

Influências

- Linguagem prática (Perl)
- Orientação a Objetos (Smalltalk)
- Metaprogramação (Smalltalk)
- Sintaxe (Smalltalk, Eiffel e Ada)
- Tratamento de Exceções (Java e Python)

*I always thought Smalltalk would beat Java, I just didn't
know it would be called 'Ruby' when it did.*
-- Kent Beck

De Java para Ruby

- Semelhanças:
 - A memória é gerida automaticamente através de um garbage collector.
 - Os Objetos são fortemente tipados.
 - Existem métodos públicos, privados e protegidos.
- Existem ferramentas de documentação embutidas (a de Ruby chama-se RDoc). A documentação gerada pelo RDoc tem um aspecto muito parecido aquela gerada pelo Javadoc.

De Java para Ruby

- Diferenças - Ao contrário de Java, em Ruby...
- Não precisará de compilar código-fonte, pois este é executado diretamente.
- Existem várias bibliotecas para interface gráfica (GUI).
- Utiliza-se a palavra-chave `end` para terminar a definição de uma classe ao invés de se colocar chaves em volta de blocos de código.
- Utiliza-se o `require` ao invés de `import`.

De Java para Ruby

- Mais diferenças...
 - Todas as variáveis de instância são privadas.
 - Os parênteses nas chamadas aos métodos são normalmente opcionais e muitas vezes omitidas.
 - Tudo é um objeto, incluído numeros como 2 ou o 3.14159.
- Não existe validação estática de tipos de dados.
- Não existe declaração de tipo de dados, estes são associados a novos nomes de variáveis conforme se necessite (por exemplo `a = [1,2,3]` em vez de `int[] a = {1,2,3};`).

De Java para Ruby

- Mais diferenças...
 - Escreve-se `foo = Foo.new("ola")` em vez de `foo = new Foo("ola")`.
 - O construtor é sempre chamado “initialize” em vez do nome da classe.
 - Temos “mixin’s” em vez de interfaces.
 - Utiliza-se `nil` em vez de `null`
- Pode-se sobrecarregar operadores como `+`, `=`, etc...
- Pode-se alterar classes nativas da linguagem até em tempo de execução.

```
class Person
  attr_reader :name #read-only access
  attr_accessor :age #read and write permission

  #Constructor - invoked by Person.new
  def initialize(name,age)
    @name = name
    @age = age
  end

  def say_hi
    puts "#{@name} says hi"
  end

  def is_joe?
    @name=="joe"
  end

end

john = Person.new("John",25)
john.say_hi #Will output "John says hi"
puts "His name is not Joe!" unless john.is_joe?
```

15,6%

dos programadores brasileiros já sabem Ruby

Fonte: Evans Data, INFO Exame 08/2007

33%

dos programadores brasileiros esperam aprender até o
final de 2008

Fonte: Evans Data, Info Exame 08/2007

Rails

Rails é um framework completo para o desenvolvimento de aplicações web com banco de dados de acordo com o padrão M.V.C. (Model-View-Control).



Princípios do Rails

- “DRY - Don’t Repeat Yourself” (Não se repita)
- “Convention over Configuration” (Convenção sobre configuração)
- “You Ain’t Gonna Need It” (Você não vai precisar disso)
- “Less is more” (Menos é mais)

DRY - Don't Repeat Yourself

- Talvez o princípio mais famoso do Rails, diz que você não deve repetir trabalho em sua aplicação, seja código, seja configuração ou qualquer outro aspecto. Quando você começa a copiar e colar coisas em vários lugares, é um forte sinal de que algo está errado. Hora de refatorar.

Convention over Configuration

- Você usa o que o Rails sugere...
- E em troca não precisa configurar **nada**.

Convention over Configuration

- Um exemplo rápido:
 - No desenho do projeto:
 - Um “Cliente” tem muitos “Carros”
 - *a client has many cars*
 - Na programação do Rails:

```
class Client  
  has_many :cars  
end
```

You Ain't Gonna Need It

- Estabelece que uma funcionalidade não deve ser adicionada ao produto até que esta seja totalmente necessária.

Todo desenvolvedor de software tem os momentos “Nossa, seria muito legal se a aplicação fizesse ...” e então horas e horas são gastas em algo que não é necessário e que pode causar problemas depois.

Less is More

- Um resultado de todos esses princípios, práticas e filosofia é o conceito de fazer menos software. Isso não significa fazer algo pela metade nem fazer de qualquer maneira, significa fazer o essencial e muito bem feito.

Por dentro do Rails

- ActiveRecord (ORM)
- ActionPack (Controllers)
- ActionMailer (E-mail service)
- ActiveSupport (Classes utilitárias, como de validação)
- ActiveResource (Web services)

ActiveRecord

- O “molho secreto” do Rails
 - Para mapear uma tabela **users** para sua aplicação, você só precisa ter uma classe assim:

```
class User < ActiveRecord::Base  
end
```
 - Segundo a convenção do Rails, classes são em singular e tabelas em plural. Ele já tem a funcionalidade para reconhecer até os plurais irregulares em inglês, como Person - people.
 - Só com essa informação, Rails já carrega todos os campos da tabela como atributos do objeto.

Migrations

- Migration é um recurso do ActiveRecord que permite que você gerencie mudanças nas suas tabelas do BD usando apenas Ruby para descrevê-las, além de proporcionar uma maneira de ter um controle de versão dessas mudanças
- Para efetuar as mudanças, é usado o Rake, que é o equivalente ao Ant no mundo Rails

```
rake db:migrate
```

Migrations

```
class CreateUsers < ActiveRecord::Migration
  def self.up
    create_table :users do |t|
      t.string :name
      t.integer :age

      t.timestamps
    end
  end

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

ActionPack

- Faz o mapeamento entre as URLs e os controllers/actions a serem chamados

```
class UsersController < ApplicationController
  #Accessed by mydomain.com/users or mydomain.com/users/index
  def index
    @users = User.find :all
  end

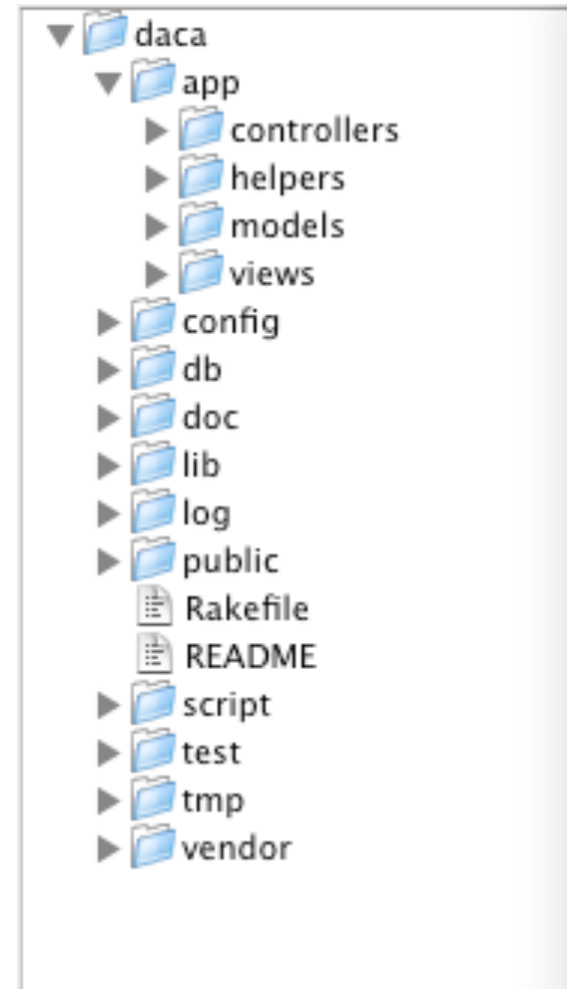
  #Accessed by mydomain.com/users/show/1
  def show
    @user = User.find params[:id]
  end
end
```

ActiveSupport

- Vários recursos úteis, entre eles a validação

```
class Text < ActiveRecord::Base
  validates_presence_of :title, :text
  validates_format_of :author_email,
    :with => /^[^@\s]+@((?:[-a-z0-9]+\.)+[a-z]{2,})$/i
end
```

Estrutura de diretórios do Rails



Scaffolding

- Ao executar o comando:
`./script/generate scaffold User name:string
age:integer`
- É gerado:
 - A classe de modelo do User
 - O controller com as actions para Criar, Remover, Editar, e Visualizar (CRUD) os Users
 - Os templates para visualização relativos ao controller
 - A migration para gerar a tabela no BD

**Talk is cheap.
Show me the code!**

Vamos agora desenvolver uma aplicação de
gerenciamento de um blog