

Ruby On Rails

Engenharia de Software



PUC Minas
Poços de Caldas

Luiz Alberto Ferreira Gomes

Curso de Ciência da Computação

8 de setembro de 2019

Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho
- 3 Esqueleto da Aplicação
- 4 Primeira Aplicação
- 5 Versionando a Primeira Aplicação
- 6 Para Saber Mais

Rails

Rails é um **framework** para construção de **aplicações web** baseado na **linguagem Ruby**.

Rails

- Uma **gem** (pacote) escrita em Ruby
- 100% **open-source**, MIT License
- Fornece **geradores** de código e scripts de **automação** de testes
- Ferramentas adicionais são fornecidas no ecossistema Rails:

Ferramentas Adicionais

- **Rake** - utilitário similar ao **make do Unix** para criar e migrar bancos de dados, limpar sessões de uma Web app
- **Puma** - servidor web de desenvolvimento para execução de aplicações Rails
- **SQLite** - um servidor de banco de dados simples pré-instalado como o Rails
- **Rack Middleware** - interface padronizado para interação entre um servidor web e uma Web App

Histórico do Rails (1)

- David Hanson **derivou** a partir do BaseCamp da 37Signals
- 07/2014 - a primeira versão de código aberto liberada
- 02/2015 - direitos **colaboração** com o projeto foram liberados
- 08/2006 - Apple distribui no Mac OS X "Leopard"
- Rails é utilizado pela companhias Airbnb, Disney, GitHub, Shopify e Twitter.

Histórico do Rails (2)

Version history			
Version	Date	Notes	
1.0 ^[22]	December 13, 2005		
1.2 ^[23]	January 19, 2007		
2.0 ^[24]	December 7, 2007		
2.1 ^[25]	June 1, 2008		
2.2 ^[26]	November 21, 2008		
2.3 ^[27]	March 16, 2009		
3.0 ^[28]	August 29, 2010		
3.1 ^[29]	August 31, 2011		
3.2 ^[30]	January 20, 2012		
4.0 ^[31]	June 25, 2013		
4.1 ^[16]	April 8, 2014		
4.2 ^[17]	December 19, 2014		
5.0 ^[18]	June 30, 2016		
5.1 ^[19]	May 10, 2017		
5.2 ^[32]	April 9, 2018		
6.0 ^[33]	August 16, 2019		
<div><div>Old version</div><div>Older version, still supported</div><div>Latest version</div><div>Future release</div></div>			

Filosofia do Rails

- Convention Over Configuration (CoC)
- Don't Repeat Yourself (DRY)
- Representational State Transfer (REST)

Convention Over Configuration

se a nomeação segue certas convenções, não há necessidade de arquivos de configuração.

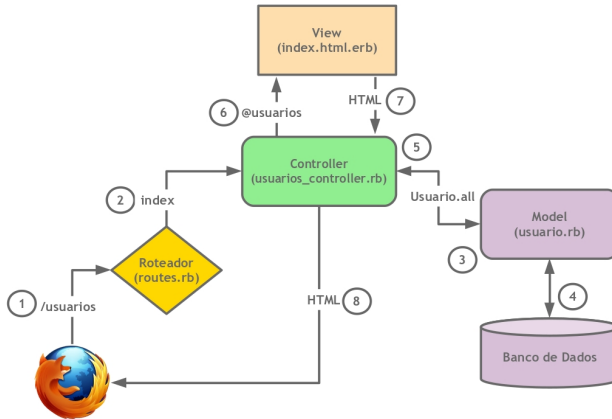
Don't Repeat Yourself

sugere que escrever que o mesmo código várias vezes
é uma coisa ruim

Representational State Transfer (1)

organiza a sua aplicação em torno de **recursos** e **padrões** HTTP
(verbs)

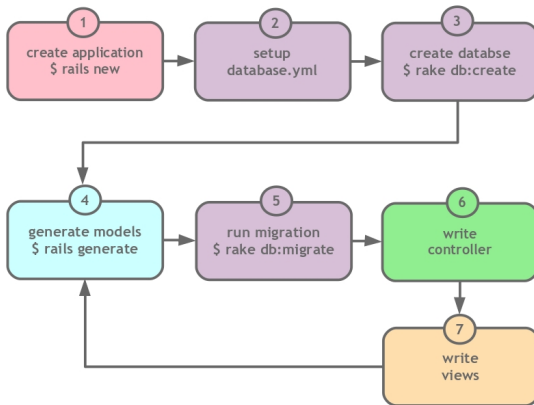
Model-View-Controller



Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho**
- 3 Esqueleto da Aplicação
- 4 Primeira Aplicação
- 5 Versionando a Primeira Aplicação
- 6 Para Saber Mais

Metodologia de Trabalho



Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho
- 3 Esqueleto da Aplicação**
- 4 Primeira Aplicação
- 5 Versionando a Primeira Aplicação
- 6 Para Saber Mais

Estrutura de uma Aplicação Rails (1)

File/Folder	Purpose
app/	Contains the controllers, models, views, helpers, mailers, channels, jobs, and assets for your application. You'll focus on this folder for the remainder of this guide.
bin/	Contains the rails script that starts your app and can contain other scripts you use to setup, update, deploy, or run your application.
config/	Configure your application's routes, database, and more. This is covered in more detail in Configuring Rails Applications .
config.ru	Rack configuration for Rack based servers used to start the application. For more information about Rack, see the Rack website .
db/	Contains your current database schema, as well as the database migrations.
Gemfile Gemfile.lock	These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the Bundler gem. For more information about Bundler, see the Bundler website .

Estrutura de uma Aplicação Rails (2)

lib/	Extended modules for your application.
log/	Application log files.
package.json	This file allows you to specify what npm dependencies are needed for your Rails application. This file is used by Yarn. For more information about Yarn, see the Yarn website .
public/	The only folder seen by the world as-is. Contains static files and compiled assets.
Rakefile	This file locates and loads tasks that can be run from the command line. The task definitions are defined throughout the components of Rails. Rather than changing Rakefile, you should add your own tasks by adding files to the lib/tasks directory of your application.
README.md	This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on.

Estrutura de uma Aplicação Rails (3)

storage/	Active Storage files for Disk Service. This is covered in Active Storage Overview .
test/	Unit tests, fixtures, and other test apparatus. These are covered in Testing Rails Applications .
tmp/	Temporary files (like cache and pid files).
vendor/	A place for all third-party code. In a typical Rails application this includes vendored gems.
.gitignore	This file tells git which files (or patterns) it should ignore. See GitHub - Ignoring files for more info about ignoring files.
.ruby-version	This file contains the default Ruby version.

Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho
- 3 Esqueleto da Aplicação
- 4 Primeira Aplicação**
- 5 Versionando a Primeira Aplicação
- 6 Para Saber Mais

Hora de Colocar a Mão na Massa (1)

1. Inicie uma janela de terminal e digite no prompt:

```
$ rails new blog
```

2. Mude para o diretório da aplicação (RAILS.root)

```
$ cd blog
```

3. Execute o servidor web embutido:

```
$ rails server
```

4. Abra uma janela do navegador e digite:

```
http://localhost:3000
```

Hora de Colocar a Mão na Massa (2)

5. Verifique o conteúdo do arquivo de configuração `database.yml`:

```
$ cat config/database.yml
```

6. Crie o banco de dados de desenvolvimento e testes:

```
$ rake db:create
```

7. Crie o modelo Post:

```
$ rails g model Post title:string body:text
```

8. Implemente modelo Post no banco de dados com *migrations*:

```
$ rake db:migrate
```

Hora de Colocar a Mão na Massa (3)

9. Crie o controlador PostsController:

```
$ rails g controller Posts
```

10. Modifique o arquivo config/routes.rb para acrescentar as rotas para o recurso posts:

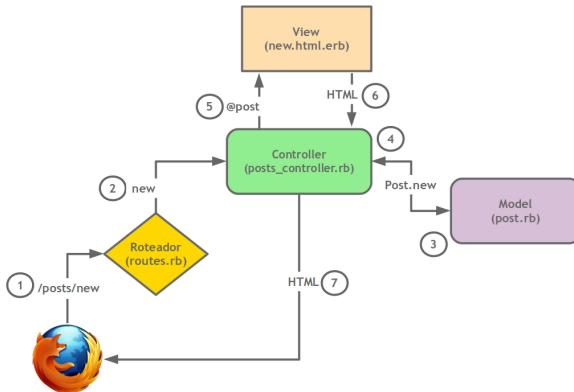
```
1 Rails.application.routes.draw do
2   resources :posts
3 end
```

11. Execute o comando rake para visualizar as rotas para os posts:

```
$ rake routes
```

12. Reinicie o servidor web e acesse a url ⟨http://localhost:3000/posts/new⟩. Veja o erro que ocorreu.

Ação: New (1)



Ação: New (2)

- Um novo objeto `@post` da classe `Post` é instanciado
- Procura pela visão `new.html.erb` para renderizar a resposta

Listing 1: `app/controllers/posts_controller.rb`

```
1  class PostsController < ApplicationController
2    def new
3      @post = Post.new
4    end
5  end
```


Visão: New (1)

- Reinicie o servidor web e acesse a url `<http:\localhost:3000/posts/new>`. Veja o erro que ocorreu.
- Implemente a visão `new.html.erb`:

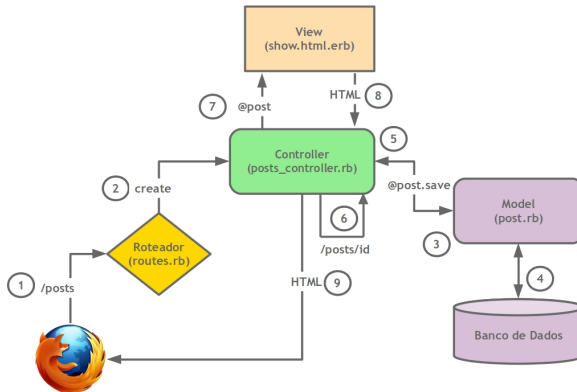
Listing 2: views/posts/new.html.erb

```
1 <h1>Novo Post</h1>
2 <%= form_with scope: :post, url: posts_path,
3   local: true do |form| %>
4   <p>
5     <%= form.label :title %><br>
6     <%= form.text_field :title %>
7   </p>
8
9   <p>
10    <%= form.label :body %><br>
```

Visão: New (2)

```
11      <%= form.text_area :body %>
12    </p>
13
14    <p>
15      <%= form.submit %>
16    </p>
17  <% end %>
```

Ação: Create (1)



Ação: Create (2)

- Um novo objeto `@post` da classe `Post` é criado com os parâmetros que foram passados pelo formulário `new`
- Tenta `salvar` o objeto `@post` no `banco de dados`

Listing 3: controllers/posts_controller.rb

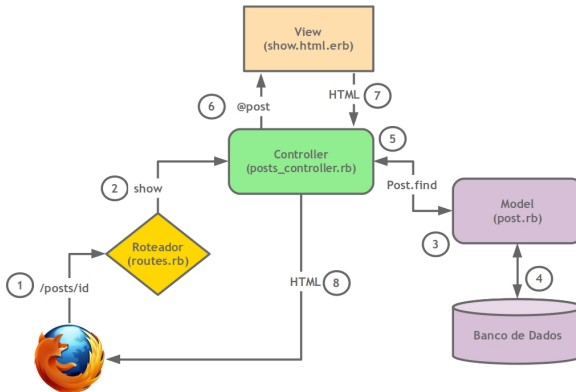
```
1 class PostsController < ApplicationController
2   def new
3     @post = Post.new
4   end
5
6   def create
7     @post = Post.new(post_params)
8
9     @post.save
10    redirect_to @post
11  end
```

Ação: Create (3)

```
12
13 private
14   def post_params
15     params.require(:post).permit(:title, :body)
16   end
17 end
```

- a linha 15 implementa **strong parameters** para aumentar a segurança da aplicação
- Como a ação **Show** ainda não foi implementada, ocorrerá uma erro quando o botão **Submit** for pressionado.

Ação: Show (1)



Ação: Show (2)

- Recupera **uma** postagem específica no parâmetro **id** passado como parte da URL
- (Implicitamente) procura pelo **show.html.erb** para renderizar a resposta

Listing 4: controllers/posts_controller.rb

```
1 class PostsController < ApplicationController
2   def show
3     @post = Post.find(params[:id])
4   end
5
6   def new
7     @post = Post.new
8   end
9
```

Ação: Show (3)

```
10     def create
11         @post = Post.new(post_params)
12
13         @post.save
14         redirect_to @post
15     end
16 private
17     def post_params
18         params.require(:post).permit(:title, :body)
19     end
20 end
```


Visão: Show (1)

- Implemente a visão `show.html.erb`:

Listing 5: views/posts/show.html.erb

```
1 <p>
2   <strong>Title:</strong>
3   <%= @post.title %>
4 </p>
5 <p>
6   <strong>Body:</strong>
7   <%= @post.body %>
8 </p>
```

Database Console

- O comando **rails db** fornece uma console para acesso aos bancos dados MySQL, PostgreSQL e SQLite.

```
$ rails db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .headers on
sqlite> .mode columns
sqlite> select * from posts;
```

id	title	body	created_at	updated_at
5	A Linguagem Ruby	Ruby e legal.	2016-04-30 22:45:20.636363	2016-04-30 22:45:20.636363

```
sqlite>
```

- Dica: utilize **headers on** e **mode coluns**

Hora de Colocar a Mão na Massa (1)

- Inicialize **na pasta da aplicação** a console do banco de dados e configure a sua exibição:

```
$ rails db  
sqlite> .headers on  
sqlite> .mode columns
```

- Exiba os colunas da tabela posts:

```
sqlite> .schema posts
```

Hora de Colocar a Mão na Massa (2)

- Exiba todos os posts:

```
sqlite> SELECT * FROM posts;
```

- Exiba todos os posts ordenados pelo título (title):

```
sqlite> SELECT * FROM posts ORDER BY title;
```

- Exiba um post:

```
sqlite> SELECT * FROM posts LIMIT 1
```

- Exiba o post cujo id é 2:

```
sqlite> SELECT * FROM posts WHERE id=2;
```

Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho
- 3 Esqueleto da Aplicação
- 4 Primeira Aplicação
- 5 Versionando a Primeira Aplicação**
- 6 Para Saber Mais

Controle Automatizado de Versão (1)

- GitHub
- git
- git config
- git init
- git add
- git commit
- git remote
- git push

GitHub



Figura: www.github.com

git



Figura: <https://git-scm.com/>

git + GitHub



Figura: <https://git-scm.com/>

Hora de Colocar a Mão na Massa (1)

1. Crie uma conta com usuário e senha no GitHub
2. Crie o repositório **blog** no GitHub
3. Inicialize os parâmetros de usuário:

```
$ git config --global user.name "nome do programador"  
$ git config --global user.email "email do programador"
```

4. Na pasta da aplicação, inicialize o repositório local:

```
$ git init
```

5. Registre as mudanças realizadas no repositório local:

```
$ git add .
```

Hora de Colocar a Mão na Massa (2)

6. Efetive as mudanças realizadas no repositório local:

```
$ git commit -m "primeiro commit"
```

7. Associe o repositório local com o repositório remoto utilizando a sua URL:

```
$ git remote add origin <GITHUB REPOSITORY URL>
```

8. Registre as alterações no repositório remoto:

```
$ git push -u origin master
```

Agenda

- 1 Ruby on Rails
- 2 Metodologia de Trabalho
- 3 Esqueleto da Aplicação
- 4 Primeira Aplicação
- 5 Versionando a Primeira Aplicação
- 6 Para Saber Mais**

Para Saber Mais

- <https://www.ruby-lang.org/en/>
 - referência oficial da linguagem Ruby onde a toda a sua documentação está disponível para ser consultada.
- <http://rubyonrails.org/>
 - referência oficial do framework Rails onde a toda a sua documentação está disponível para ser consultada.
- <http://www.codecademy.com/pt/tracks/ruby>
 - curso iterativo em português sobre a linguagem Ruby.
- <https://gorails.com/setup/ubuntu/16.04>
 - guia para instalação do Ruby on Rails no Ubuntu e no Mac OSX.