CAPÍTULO 17

Apêndice: Rotas e Rack

"Não é possível estar dentro da civilização e fora da arte"
— Rui Barbosa

O modo como urls são ligadas a controladores e actions pode ser customizado no Rails. O módulo responsável por esta parte é o que foi criado com o seu projeto `NomeDoProjeto::Application.routes` e as rotas podem ser customizadas no arquivo config/routes.rb.

17.1 - RACK

O `rack` é uma abstração das requisições e respostas HTTP da maneira mais simples possível. Criando uma API unificada para servidores, frameworks, e softwares (os conhecidos middleware) em apenas uma chamada de método.

A grande motivação da criação do Rack é que, diferente do mundo java onde existe uma especificação que abstrai todo o HTTP, no mundo ruby cada framework havia criado a sua forma de tratar as requisições e respostas. Por isso, escrever um servidor ou mesmo permitir que o framework X pudesse rodar em um servidor que já existisse era um trabalho realmente complicado. Graças ao surgimento do rack e da sua padronização hoje é possível que qualquer servidor que conheça rack consiga executar qualquer aplicação que se comunique com o HTTP através do rack.

Mais do que isso, hoje também é possível fazer uma "aplicação" web em apenas uma linha. Exemplo:

```
run Proc.new {|env| [200, {"Content-Type" => "text/html"},
        ["Hello World"]]}
```

Basta salvar esse arquivo, por exemplo como **hello.ru**, e subir nosso servidor pelo Terminal com o seguinte comando:

```
$ rackup hello.ru
```

Para criar uma "aplicação" em rack tudo o que precisamos é criar um método que retorne [httpStatusCode, headers, body], como no exemplo acima.

O comando rackup é criado quando instalamos a gem 'rack' e serve para iniciar aplicações feitas em rack. Elas nada mais são que um arquivo ruby, mas devem ser salvos com a extensão .ru (RackUp) e devem chamar o método `run`.

17.2 – Exercícios – Testando o Rack

1. Vamos fazer uma aplicação rack.

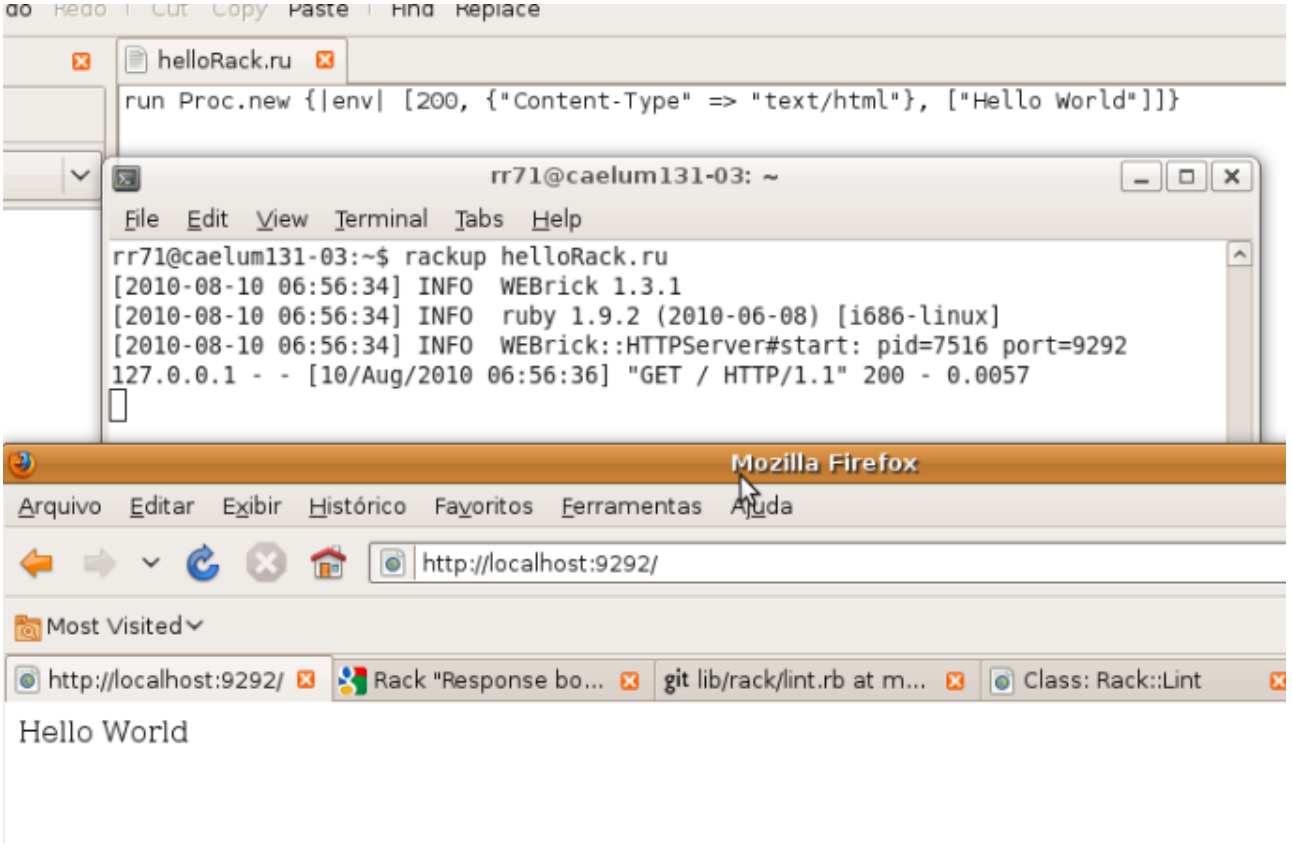
   a. Crie um arquivo chamado **"helloRack.ru"**

   b. Adicione as seguintes linhas:

```
$ run Proc.new {|env| [200, {"Content-Type" => "text/html"},
  ["Hello World"]]}
```

   c. Inicie a aplicação com o comando `rackup helloRack.ru`

   d. Teste no browser pela url: http://localhost:9292/



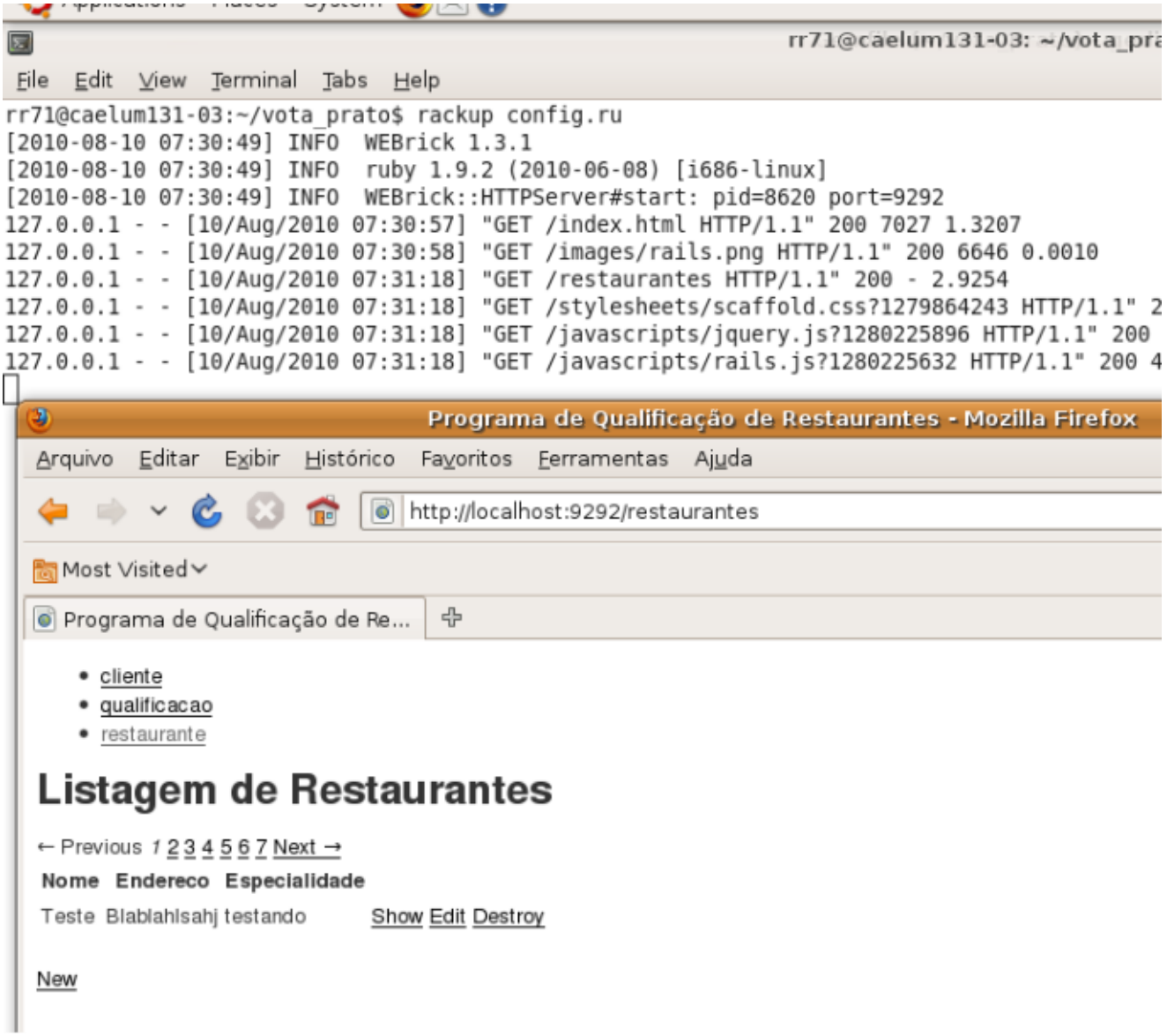
Tire suas dúvidas no novo GUJ Respostas

O GUJ é um dos principais fóruns brasileiros de computação e o maior em português sobre Java. A nova versão do GUJ é baseada em uma ferramenta de *perguntas e respostas* (QA) e tem uma comunidade muito forte. São mais de 150 mil usuários pra ajudar você a esclarecer suas dúvidas.

Faça sua pergunta.

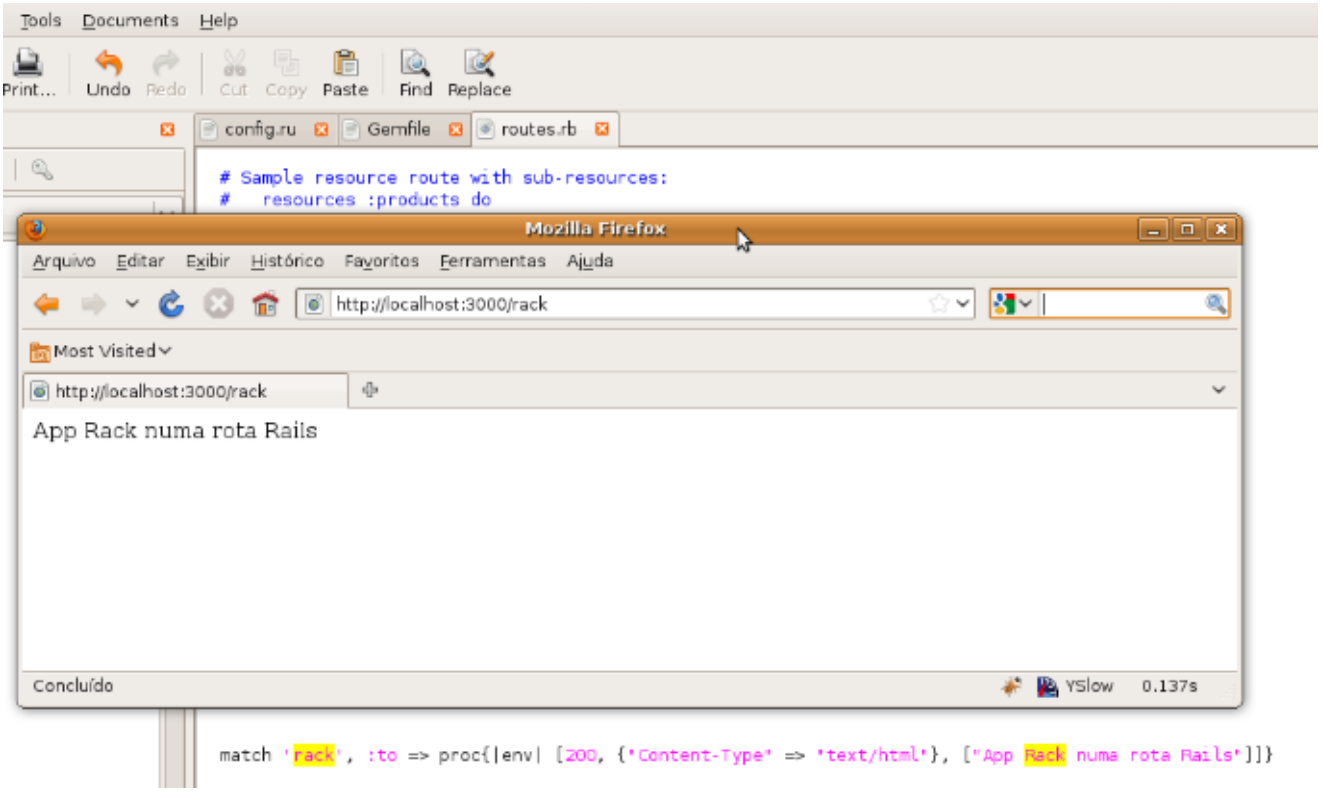
17.3 - Rails e o Rack

A partir do Rails 3, quando criamos uma nova aplicação, um dos arquivos que ele cria é config.ru na raiz do projeto e, mais que isso, podemos afirmar que toda aplicação Rails 3 é uma aplicação rack. Prova disso é que conseguimos iniciar a aplicação através do comando `rackup config.ru`



Outro ponto interessante sobre o rack e o Rails, é que agora é possível mapear uma aplicação rack diretamente em uma rota de uma aplicação rails.

```
# routes.rb
  match 'rack',
  :to => proc{|env| [200, {"Content-Type" => "text/html"},
      ["App Rack numa rota Rails"]]}
```



17.4 – Exercícios – Criando um rota para uma aplicação Rack

1. Vamos fazer uma aplicação rack.

   a. Abra o arquivo **"routes.rb"**

   b. Adicione a seguinte linha:

```
match 'rack',
 :to => proc{|env| [200, {"Content-Type" => "text/html"},
["App Rack numa rota Rails"]]}
```

   c. Inicie a aplicação com o comando `rails server`

   d. Teste no browser pela url: http://localhost:3000/rack

CAPÍTULO ANTERIOR:

Apêndice: Testes

PRÓXIMO CAPÍTULO:

Apêndice: Design Patterns em Ruby

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter