Rails Guia de Bolso

Sobre o Rails Guia de Bolso

Uso o Ruby on Rails cerca de um ano e meio, o que não é muito tempo, se comparado com muitos de meus colegas. Conforme minha paixão pelo framework e pela linguagem cresciam, também aumentava meu apetite por mais conhecimento. Agora que tenho aproximadamente 12 livros sobre Ruby on Rails e sobre Ruby, penso haver verdadeira necessidade de um guia de bolso para o Ruby on Rails.

Este livro não contém informações novas, que não possam ser facilmente acessadas online ou em livros impressos. Não tem a intenção de "dar uma nova luz" à linguagem ou ao framework. É apenas uma tentativa de coletar o máximo possível de informações que um típico desenvolvedor de Rails como eu penso serão úteis no dia-a-dia. Grande parte do livro contém a documentação original do Rails, fornecida por sua comunidade, além de anotações e exemplos meus e de outras pessoas.

Começando

Para ver se o Rails está instalado em seu computador, digite o seguinte em um prompt de comando:

```
rails --version
```

Uma resposta afirmativa seria similar a esta (exemplo do Rails 2.1.0 instalado em Mac OS X):

```
Rails 2.1.0
```

RubyGems

RubyGems é um gerenciador de pacotes para Ruby (http://rubygems.rubyforge.org). Ele foi criado por Jim Weirich (http://onestepback.org). O gerenciador instala os pacotes do software Ruby como Rails e os mantém atualizados. É bem fácil aprendê-lo e usá-lo, mais fácil do que ferramentas como o utilitário Unix/Linux (http://www.gnu.org/software/tar) ou o utilitário jar do Java (http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/jar .html).

Para mais informações, leia a documentação do RubyGems no endereço http://docs.rubygems.org. O RubyGems User Guide (http://docs.rubygems.org/read/book/1) oferece quase tudo o que você precisa saber sobre a utilização do RubyGems. Um guia de comandos também está disponível (http://docs.rubygems.org/read/book/2).

Instalando o Rails com RubyGems

O RubyGems é instalado com o Ruby, mas caso queira maiores informações, consulte o Capítulo 3 do RubyGems User Guide para instruções completas sobre a instalação (http://rubygems.org/read/chapter/3).

Para ver qual versão do RubyGems está instalada, digite:

```
gem --version
```

Para obter ajuda do RubyGems, você pode tentar:

```
gem --help
```

Para visualizar uma lista de comandos para o RubyGems, digite:

```
gem help commands
```

Para ver exemplos de comandos com descrições, digite:

```
gem help examples
```

Para visualizar ajuda em um comando específico, digite:

```
gem help [command]
```

Para instalar ou atualizar o Rails (`sudo` exige uma senha root), digite:

```
sudo gem install rails
```

Utilizando o Instant Rails em Windows

O Instant Rails é uma solução baseada em Windows para rodar o Ruby on Rails com Apache e MySQL. Com o Instant Rails, você pode configurar facilmente sua pasta raiz do Rails e colocá-lo para funcionar. Não há necessidade de alterar o ambiente do sistema.

Para instalar o Instant Rails e criar uma aplicação chamada myapp, faça o seguinte:

1. Baixe e descompacte o Instant Rails no endereço http://rubyforge.org/projects/instantrails.
2. Certifique-se de que não haja caracteres de espaço no caminho de instalação, em seguida, inicie o *InstallRails.exe*.
3. O Instant Rails irá detectar que está sendo iniciado a partir de um novo diretório e perguntará se você quer recriar os arquivos de configuração. Clique em OK.
4. Clique sobre o botão I para abrir o menu principal. Em seguida, selecione Rails Applications → Manage Rails Applications...
5. No editor que será exibido, clique sobre o botão Create New Rails App...
6. Um console será aberto. Digite **`rails myapp`**.
7. Agora, feche a janela Rails Configuration e abra-a novamente, seguindo o passo 4.
8. Agora, você verá *myapp* na **lista Rails Application**. Clique na caixa de seleção próxima a ela e, em seguida, clique em "Start with Mongrel".
9. Abra um navegador e acesse http://127.0.0.1:3000.
10. Bem vindo a bordo!

Dependências Gem

As dependências Gem foram adicionadas no Rails 2.1. Isso significa que as gems requisitadas podem ser especificadas no arquivo *config/environment.rb*, e serão carregadas automaticamente quando a aplicação for iniciada. Por exemplo:

```
Rails::Initializer.run do |config|

  # Especificar as gems das quais depende esta aplicação.
  # Elas podem ser instaladas com "rake gems:install" em novas
instalações.
  config.gem "bj"
  config.gem "hpricot", :version => '0.6',
              :source => "http://code.whytheluckystiff.net"
  config.gem "aws-s3", :lib => "aws/s3"
  ...
```

Além disso, agora há uma tarefa `rake`, que instala todos os `config.gems` em seu sistema de destino:

```
rake gems:install
```

Se você quiser inserir estas gems na origem de sua aplicação, poderá fazê-lo com a tarefa `rake gems:unpack`:

```
# Desempacotar todas as gems para vendor/gems.
rake gems:unpack
```

Também é possível desempacotar gems individuais:

```
# Desempacotar somente a gem hpricot para vendor/gems
rake gems:unpack GEM=aws-s3
```

Esta ação irá desempacotar a gem para o diretório *vendor/gems/aws-s3x.x.x*, que é buscado automaticamente como parte da inicialização de `config.gem`.

Para construir gems com extensões nativas, você pode utilizar a tarefa rake gems:build:

```
rake gems:build
# Ou construir uma gem específica
rake gems:build GEM=aws-s3
```

Comandos e Configuração do Rails

Uma vez que o Rails esteja instalado, o comando `rails` pode ser usado para gerar novas aplicações Rails com uma estrutura de diretórios padrão e configuração no caminho especificado.

Para criar uma aplicação Rails chamada myapp, digite:

```
rails myapp
```

Quando a aplicação funcionar, você verá uma lista de diretórios e arquivos gerada pelo comando. Esta é sua aplicação Rails. A pasta principal é myapp, ou **RAILS ROOT**.

Uso e Opções

Para obter ajuda para o comando rails, digite:

```
rails --help
```

Uso

rails *[/caminho/para/sua/nomedaaplicacao] [opções]*

Opções

-r, --ruby=path
: Caminho para o **binário Ruby** de sua escolha

-d, --database=name
: Pré-configurado para uma base de dados selecionada (por exemplo: `mysql`, `Oracle`, `postgresql`, `sqlite2`, `sqlite3`)

-f, --freeze
: Congela o Rails em vendor/rails a partir das gems que geram a frame

-v, --version
: Mostra o número da versão Rails e finaliza

-p, --pretend
: Roda mas não faz alterações

--force
: Sobrescreve arquivos já existentes

-s, --skip
: Pula arquivos já existentes

-q, --quiet

Suprime os dados de saída normais

-t, -- backtrace

Depuração; mostra um caminho de volta (backtrace) de erros

-c, --svn

Modifica arquivos com subversão (svn deve estar no caminho)

Estrutura de Arquivos Rails

Após gerar uma aplicação Rails, um diretório e uma estrutura de arquivos padrão são criados. Veja a Tabela 1-1.

Caminho	Descrição
app	Contém todo o código específico para esta determinada aplicação.
app/controllers	Contém as classes de controllers que devem ser nomeadas como *users_controller.rb* para mapeamento automatizado de URL. Todos os controladores devem ser descendentes de `ApplicationController`, que descende de `ActionController::Base`.
app/models	Contém os modelos que devem ser nomeados, como *product.rb*. A maioria dos modelos descende de `ActiveRecord::Base`.
app/views	Contém os arquivos de template para a visualização que deve ser nomeada, como *users/index.html.erb para a ação* `UsersController index`.Todas as visualizações usam sintaxe eRuby.
app/views/layouts	Contém os arquivos de template dos layouts que devem ser usados com as visualizações. Estes arquivos modelam o método comum de cabeçalho/rodapé para empacotar as visualizações. Em suas visualizações, defina um layout utilizando o layout *:default*, e crie um arquivo chamado *default.html.erb*. Dentro de *default.html.erb*, chame `<%= yield %>` para renderizar a visualização utilizando este layout.
app/helpers	Contém classes auxiliares que devem ser nomeadas, como *users_helper.rb*. Estes helpers são gerados automaticamente quando se utiliza script/generate para controladores. Os helpers podem ser usados para empacotar funcionalidades de visualizações dentro dos métodos.
Config	Arquivos de configuração para o ambiente Rails, mapa de roteamento, base de dados, e outras dependências.

Caminho	Descrição
db	Contém o esquema de base de dados em *schema.rb.db/migrate*; contém todas as sequências de migrações para seu esquema.
doc	Este diretório armazenará a documentação de sua aplicação quando gerada através de `rake doc:app`.
lib	Bibliotecas específicas da aplicação. Basicamente, qualquer tipo de código personalizado que não pertença a controladores, modelos, ou auxiliares. Este diretório fica no caminho de carregamento.
public	Diretório disponível para o servidor web. Contém subdiretórios para imagens, folhas de estilo e JavaScripts. Contém também os dispacthers (despachadores) e os arquivos HTML padrão. Deve ser configurado como DOCUMENT ROOT de seu servidor web.
script	Scripts auxiliares para automação e geração.
test	Testes de unidade e funcionais, além de fixtures. Ao utilizar os scripts script/generate, os arquivos de template de testes serão gerados para você e inseridos neste diretório.
Vendor	Bibliotecas externas das quais a aplicação depende. Também inclui o subdiretório de plu-gins. Este diretório fica no caminho do carregamento.

Configurando o Rails

Toda aplicação Rails baseia-se em seus arquivos de configuração para especificar como a aplicação irá trabalhar. Estes arquivos são encontrados na pasta *config*:

boot.rb
: Arquivo de inicialização do Rails. Tipicamente, este arquivo não precisa ser modificado.

routes.rb
: Arquivo de configuração no qual as rotas são especificadas.

environment.rb
: Configurações gerais da aplicação Rails.

environments/development.rb
: Configurações específicas do ambiente de desenvolvimento.

environments/test.rb
: Configurações específicas do ambiente de testes.

environments/production.rb
: Configurações específicas do ambiente de produção.

database.yml

Configura sua conexão com a base de dados.

initializers/inflections.rb

Adiciona novas regras de inflexão à aplicação (isto é: plural, singular, irregular, incontável).

initializers/mime_types.rb

Adiciona novos tipos mime para uso em blocos `respond_to` (isto é, rtf ou iPhone).

initializers/new_rails_defaults.rb

Estas configurações alteram o comportamento das aplicações em Rails 2 e serão configurações padrão no Rails 3. *Este arquivo será retirado no Rails 3*.

Para mais informações sobre rotas, ambientes e configurações de bases de dados, consulte as seções deste livro referentes a estes assuntos.

Scripts

O Rails vem com scripts auxiliares para automação e geração de código. Utilizando estes scripts, os desenvolvedores podem construir aplicações rapidamente, ao mesmo tempo em que mantêm o controle sobre o conteúdo gerado.

script/about

Exibe informações sobre o ambiente de sua aplicação:

```
Ruby version          1.8.6 (universal-darwin9.0)
RubyGems version      1.1.1
Rails version         2.1.0
Active Record version 2.1.0
Action Pack version   2.1.0
Active Resource version 2.1.0
Action Mailer version 2.1.0
Active Support version 2.1.0
Application root      /Users/berry/Sites/myapp
Environment           development
```

script/console

O console lhe dá acesso ao seu ambiente Rails, onde você pode interagir com o modelo de domínio. Aqui, você irá configurar todas as partes de sua aplicação, da mesma forma como ela será quando estiver funcionando. Você pode inspecionar os modelos de domínio, alterar valores e salvá-los na base de dados.

Inicie o console com o script de console:

```
./script/console
```

Iniciar o console sem argumentos fará com que o console rode utilizando o ambiente de desenvolvimento.

Para sair do console, digite:

```
quit ou exit
```

Com a versão mais recente do Rails, você pode recarregar seus modelos e controladores com o seguinte comando:

```
reload!
```

Para resetar sua aplicação, digite:

```
Dispatcher.reset_application!
```

Uso

./script/console *[ambiente] [opções]*

Opções

-s, --sandbox

Repete as modificações da base de dados na saída

--irb=[irb]

Chama um *irb* diferente

DICA

Utilizar o console em modo sandbox pode ser uma ferramenta muito poderosa para o desenvolvedor Rails. Por exemplo: se você criar um script que atualize todas as linhas de uma tabela, poderá fazer uma rodada de testes primeiro no modelo sandbox para verificar os dados existentes e certificar-se de que as alterações corretas foram

feitas. Se algo der errado, apenas saia do console e as alterações não serão efetivadas na base de dados.

script/destroy

O script irá destruir todos os arquivos criados pelo comando `script/generate` correspondente. Por exemplo: `script/destroy migration CreatePost` excluirá o arquivo *###_create_post.rb* correspondente em *db/migrate*, enquanto `script/destroy` **`scaffold Post`** excluirá o controlador e as visualizações da postagem, o modelo e a migração da postagem, todos os testes associados e a linha `map.resources :posts` de *confif/routes.rb*.

Uso

```
./script/destroy generator [opções] [args]
```

Exemplo

```
./script/destroy controller Products

      rm  app/helpers/products_helper.rb
      rm  test/functional/products_controller_test.rb
      rm  app/controllers/products_controller.rb
    rmdir  test/functional
  notempty  test
    rmdir  app/views/products
  notempty  app/views
  notempty  app
  notempty  app/helpers
  notempty  app
  notempty  app/controllers
  notempty  app
```

script/generate

Os geradores são usados para criar código instantâneo que será utilizado em aplicações Rails. Ao rodar o gerador (através do script/generate), novos arquivos (controllers, models, views etc.) são gerados e adicionados a sua aplicação. Você também pode criar geradores personalizados. Para mais informações sobre geradores personalizados, assista ao **Railscast** de **Ryan Bates** no endereço http://railscasts.com/episodes/58.

Para obter mais ajuda sobre o script gerador, digite:

```
./script/generate --help
```