Yalantis

Лекция 14: Инфраструктура Rails

Курс лекций по основами web-разработки на языке программирования Ruby

Yalantis

Тема

- ➔ Rake
- ➔ Генераторы
- ➔ ActiveJob
- ➔ ActionMailer
- ➔ Internationalization.

Yalantis

Rake

Язык сборки, написанный на Ruby

Был включен по умолчанию в версию 1.9, вместо make

Используется для автоматизации рутинных задач

Задача - поименованный блок кода с инструкциями

Задачи могут быть связанными

Yalantis

Часто используемые задачи

Узнать список существующих задач можно командой *rake -T*

- db - отвечает за операции с базой данных
- doc - создает документацию в каталоге doc/app
- tmp - операции с временными файлами
- stats - обзор компонентов приложения
- notes - анализирует заметки в коде с пометкой TODO, FIXME, etc
- about - информация о версии ключевых компонентов приложения
- secret - создание секретного ключа для приложения
- assets - менеджмент загружаемых ресурсов (ассетов)
- routes - показывает все маршруты приложения

Yalantis

Начало работы

Установка

```
gem install rake  | |  sudo gem install rake
```

Первый пример

```
task :default do
    puts "Hello World!"
end
```

Запуск первой задачи

```
$ rake

Hello world
```

Yalantis

Варианты расширений

- rakefile.rb
- rakefile
- Rakefile
- Rakefile.rb
- .rake - наиболее часто используемый

Yalantis

Пространство имен

rake db:drop

rake db:seed

rake db:rollback

rake db:migrate:status

```
namespace :db do
  desc 'Migrating some stuff'
  task :migrate do
    ...
  end
end
```

Yalantis

Зависимости

Зависимость - возможность связать выполнение задач.

```
task :turn_the_key do
  puts ‘Key is turning’
end

task :open_the_door => :turn the key do
  puts ‘You opened the door'
end

# вызов таски выведет следующее

$ rake open_the_door
‘Key is turning’
‘You opened the door’
```

```
task :turn_the_key do
  puts ‘Key is turning’
end

task :open_the_door
  puts ‘You opened the door'
end

task :open_the_door => :turn_the_key

# вызов таски выведет следующее

$ rake open_the_door
‘Key is turning’
‘You opened the door’
```

Yalantis

Создание большого числа зависимостей

```
task :ring_the_bell do
  puts ‘The bell is ringing’
end

task :turn_the_key do
  puts ‘Key is turning’
end

...

task :open_the_door => [:ring_the_bell, :turn the key, …] do
  puts ‘You opened the door'
end

$ open_the_door

The bell is ringing  =>  Key is turning  =>  …  =>  You opened the door
```

Yalantis

Передача параметров

Существует два способа передать параметр в таску:

- использование локальных переменных Ruby,
- передача параметра с помощью синтаксиса Rake

```
# создадим rake таску
desc “This task can output evironmental variables”
task :system_greetings do
    puts “Hello, #{ENV[‘user_name]}
end
# вызов
$ rake system_greetings NAME= ‘Guest’
# выведет
 Hello, Guest
```

```
task :greetings do |t, args|
    puts “Hello #{args.name}”
end
name = ‘Archer’
# далее вызовем таску, передав в нее переменную name
$ rake “greetings[name]
# выведет
 Hello, Archer
```

Yalantis

```
namespace :production_seed do
  desc 'Generate all required instances'
  task all: :environment do
      ActiveRecord::Base.transaction do
      Rake::Task['production_seed:admin'].invoke
      # some other tasks ....
      end
  end
  desc 'Generate admin'
  task admin: :environment do
      require_relative '../../db/production_seeds/admin'
      ProductionSeeds::Admin.new(email: ENV['ADMIN_EMAIL'],
                            password: ENV['ADMIN_PASSWORD']).perform
  end
end
```

Yalantis

Генераторы

Инструмент для создания одинаковых структур для множества элементов в одном и том же приложении.

rails generate (rails g) - показывает все существующие генераторы

Наиболее часто используемые:

- Controller
- Model
- Migration
- Helper
- Scaffold

Yalantis

Controller

Генерирует файл контроллера, связанные представления, тесты, хелперы и таблицу стилей

```
rails generate controller NAME [action action] [options]

# например

create  app/controllers/greetings_controller.rb
     route  get 'greetings/hello'
     invoke  erb
     create    app/views/greetings
     create    app/views/greetings/hello.html.erb
     invoke  test_unit
     create    test/controllers/greetings_controller_test.rb
     invoke  helper
     create    app/helpers/greetings_helper.rb
     invoke    test_unit
     invoke  assets
     invoke    scss
     create      app/assets/stylesheets/greetings.scss
```

Yalantis

Model

Создает заглушку модели, тестов и миграцию

```
rails generate model NAME [field[:type][:index] field[:type][:index]] [options]

#например

rails generate model admin/account
        invoke  active_record
        create db/migrate/20200519072918_create_admin_accounts.rb
        create app/models/admin/account.rb
        create app/models/admin.rb
        invoke rspec
        create          spec/models/admin/account_spec.rb
        invoke          factory_bot
        create          spec/factories/admin/accounts.rb
```

Yalantis

Migration

Создает заглушку модели, тестов и миграцию.

```
rails generate migration NAME [field[:type][:index] field[:type][:index]] [options]

# например
rails generate migration AddTitleBodyToPost title:string body:text published:boolean
# создаст файл миграции со следующими записями
        add_column :posts, :title, :string
        add_column :posts, :body, :text
        add_column :posts, :published, :boolean

rails generate migration AddPartNumberToProducts part_number:string:index
# создаст файл миграции, добавит поля:
        add_column :products, :part_number, :string
        add_index :products, :part_number
```

Yalantis

Helper

Создает заглушку хелпера, тесты к нему.

```
rails generate helper NAME [options]

# например
rails generate helper CreditCard
#создаст
        app/helpers/credit_card_helper.rb
```

Yalantis

Scaffold

Создает шаблон целого ресурса:

- контроллер
- модель
- миграции
- вьюхи
- тесты

```
rails generate scaffold NAME [field[:type][:index] field[:type][:index]] [options]
```

Yalantis

New

Создает Rails приложение со стандартной структурой директорий и конфигурацией

```
rails new ~/Code/Ruby/weblog

# создаст скелет Rails приложения по адресу ~/Code/Ruby/weblog.
```

Yalantis

Дополнительно

- Базу ваших генераторов можно расширить, устанавливая гемы генераторы
- *rails delete <generator>* удалит созданную генератором структуру
- Генерация шаблонов может перекрыть существующий функционал

Yalantis

ActiveJob

Фреймворк для объявления и запуска задач

Каждое приложение Rails имеет инфраструктуру для заданий

Не зависит от бэкенда

Асинхронная реализация

Yalantis

Создание

```
rails generate job NAME [options]

# например

rails generate job guests_cleanup

invoke  test_unit
create          test/jobs/guests_cleanup_job_test.rb
create  app/jobs/guests_cleanup_job.rb

class GuestsCleanupJob < ApplicationJob
  queue_as :default

  def perform(*args)
      # Do something later
  end
end
```

Yalantis

Помещение в очередь

Существует два способа постановки задачи в очередь - **perform_later** и **perform_now**

Примеры помещения в очередь:

- **MyJob.perform_later guest** - задание выполнится, как только освободится система
- **MyJob.set(wait_until: Date.tomorrow.noon).perform_later(guest)** - выполнится завтра, в полдень
- **MyJob.set(wait: 1.week).perform_later(guest)** - выполнится через неделю

Yalantis

Бэкенды

- Sidekiq - https://github.com/mperham/sidekiq/wiki/Active-Job
- Resque - https://github.com/resque/resque/wiki/ActiveJob
- Sneakers - https://github.com/jondot/sneakers/wiki/How-To:-Rails-Background-Jobs-with-ActiveJob
- Sucker Punch - https://github.com/brandonhilkert/sucker_punch#active-job
- Queue Classic - https://github.com/QueueClassic/queue_classic#active-job
- Delayed Job - https://github.com/collectiveidea/delayed_job#active-job
- Que - https://github.com/que-rb/que#additional-rails-specific-setup

Yalantis

Очереди

```
# определение очереди для задания
class GuestsCleanupJob < ApplicationJob
  queue_as :low_priority
  #....
end

# добавления префикса для очереди
module YourApp
  class Application < Rails::Application
          config.active_job.queue_name_prefix = Rails.env
  end
end

# production => production_low_priority
# staging      => staging_low_priority

# переопределения префикса в теле джобы
   self.queue_name_prefix = nil
```

Yalantis

Колбэки

```
class GuestsCleanupJob < ApplicationJob
  queue_as :default

  around_perform :around_cleanup

  def perform
    # Отложенное задание
  end

  private
    def around_cleanup
      # Делаем что-то перед perform
      yield
      # Делаем что-то после perform
    end
end
```

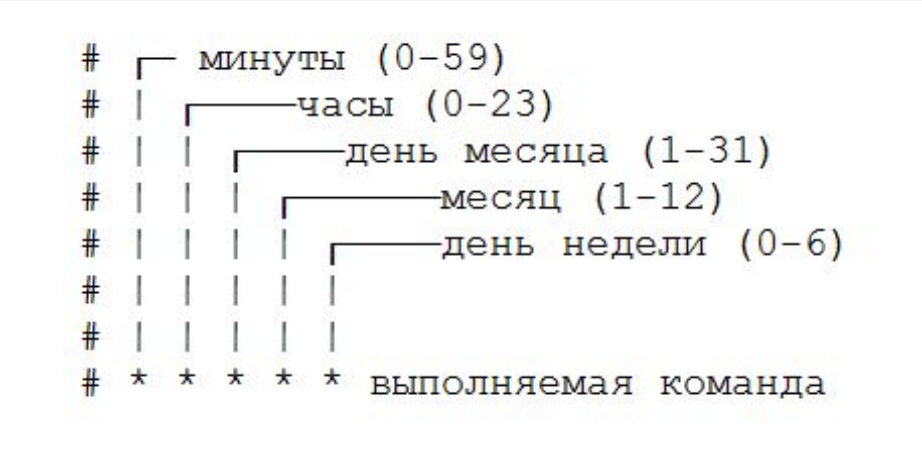
Yalantis

Доступные колбэки

- before_enqueue
- around_enqueue
- after_enqueue
- before_perform
- around_perform
- after_perform

Yalantis

Cron Jobs



Выполнение задач “по расписанию”

Сфера применения:

- Регулярные бэкапы
- мониторинг дискового пространства
- удаление файлов

Шаблон для создания:

Минуты Часы День Месяц День недели Команда

Yalantis

Отображение содержимого crontab-файла

```
$ crontab -l
```

Редактирование crontab-файла

```
$ crontab -e
```

Примеры создания

```
* * * * * <исполняемая-команда> - будет выполняться каждую минуту
30 * * * * <исполняемая-команда> - каждые 30 минут
0 */2 * * * <исполняемая-команда> - выполнение каждые 2 часа
0 0 * * SUN <исполняемая-команда>  - будет выполняться каждое воскресенье
0 0 1 1 * <исполняемая-команда> - исполнение каждый год 1-го января
```

Yalantis

Готовые задания

- @reboot — одиночное выполнение команды при загрузке;
- @yearly — раз в год;
- @annually — тоже раз в год;
- @monthly — раз в месяц;
- @weekly — один раз в неделю;
- @daily — раз в день;
- @midnight — тоже раз в день;
- @hourly — раз в час.

Пример выполнения команды каждый год:

```
$ @yearly <исполняемая-команда>
```

Для очистки всех заданий:

```
$ crontab -r
```

Yalantis

crontab.guru

Yalantis

ActionMailer

Отправление и получение электронной почты

Основную работу выполняют рассыльщики (Mailer)

У рассыльщиков есть:

- Экшены и связанные вьюхи
- Переменные экземпляра
- Возможность использовать партиалы
- Доступ к params

Yalantis

Создание рассыльщика

У рассыльщика есть свой генератор

```
$ rails generate mailer UserMailer
create  app/mailers/user_mailer.rb
create  app/mailers/application_mailer.rb
invoke  erb
create    app/views/user_mailer
create    app/views/layouts/mailer.text.erb
create    app/views/layouts/mailer.html.erb
invoke  test_unit
create    test/mailers/user_mailer_test.rb
create    test/mailers/previews/user_mailer_preview.rb
```

```
# app/mailers/application_mailer.rb
class ApplicationMailer < ActionMailer::Base
  default from: "from@example.com"
  layout 'mailer'
end

# app/mailers/user_mailer.rb
class UserMailer < ApplicationMailer
end
```

Также можно создать файл в *app/mailers*

```
class MyMailer < ActionMailer::Base
end
```

Yalantis

Работа с рассыльщиками

```
#app/mailers/user_mailer.rb
class UserMailer < ApplicationMailer
  default from: 'notifications@example.com'

  def welcome_email
    @user = params[:user]
    @url  = 'http://example.com/login'
    mail(to: @user.email, subject: 'Welcome to My Awesome Site')
  end

end
```

```
#app/views/user_mailer/welcome_email.html.erb
<!DOCTYPE html>
<html>
  <head>
    <meta content='text/html; charset=UTF-8'
http-equiv='Content-Type' />
  </head>
  <body>
    <h1>Welcome to example.com, <%= @user.name %></h1>
    <p>
      To login to the site, just follow this link: <%= @url %>.
    </p>
    <p>Thanks for joining and have a great day!</p>
  </body>
</html>
```

Отправка

```
UserMailer.welcome_email.deliver_later # deliver_now
```

Yalantis

Методы ActionMailer

- headers[:field_name] = 'value' - определяет любой заголовок email
- attachments['file-name.jpg'] = File.read('file-name.jpg') - позволяет прикрепить файлы в email
- mail - отправка email
- default - определение параметров по умолчанию (например *to: -> { emails_list })*

Yalantis

Представления

Изменение шаблона по умолчанию

```
class UserMailer < ApplicationMailer
  default from: 'notifications@example.com'

  def welcome_email
    @user = params[:user]
    @url  = 'http://example.com/login'
    mail(to: @user.email,
         subject: 'Welcome to My Awesome Site',
         template_path: 'notifications',
         template_name: 'another')
  end
end

# мейлер будет искать шаблон app/views/notifications
```

Yalantis

Гибкая настройка

```
class UserMailer < ApplicationMailer
  default from: 'notifications@example.com'

  def welcome_email
    @user = params[:user]
    @url  = 'http://example.com/login'
    mail(to: @user.email,
         subject: 'Welcome to My Awesome Site') do |format|
      format.html { render 'another_template' }
      format.text { render plain: 'Render text' }
    end
  end

end
```

Посмотреть email, не отправляя его, можно по адресу: *test/mailers/previews*

Yalantis

Колбэки

```
class InvitationsMailer < ApplicationMailer
  before_action { @inviter, @invitee = params[:inviter], params[:invitee] }
  before_action { @account = params[:inviter].account }

  default to:       -> { @invitee.email_address },
          from:     -> { common_address(@inviter) },
          reply_to: -> { @inviter.email_address_with_name }

  def account_invitation
    mail subject: "#{@inviter.name} invited you to their Basecamp (#{@account.name})"
  end

  def project_invitation
    @project    = params[:project]
    @summarizer = ProjectInvitationSummarizer.new(@project.bucket)

    mail subject: "#{@inviter.name.familiar} added you to a project in Basecamp (#{@account.name})"
  end
end
```

Yalantis

Хэлперы

Action Mailer наследуется от *AbstractController.*

У вас есть доступ к тем же общим хелперам, как и в *Action Controller*.

Yalantis

Internationalization

Локализация - процесс адаптации программного продукта к языку и культуре клиента.

Интернационализация - проектирование и реализацию программного продукта или документации таким образом, который максимально упростит локализацию приложения.

Yalantis

Интернационализация в Rails

Гем I18n(сокращение для *internationalization*) - фреймворк для перевода вашего приложения на другой язык и поддержки мультиязычности.

Гем разделен на две части:

- Публичный API с методами, определяющими как работает библиотека
- Бэкенд по умолчанию, реализующий эти методы

Yalantis

Настройка

Использующиеся в приложении локали хранятся в *config/locales*

```
en:
  hello: "Hello world"
```

Настройка параметров локали производится в *config/application.rb*

```
# где библиотека I18n должна искать наши переводы
I18n.load_path += Dir[Rails.root.join('lib', 'locale', '*.{rb,yml}')]

# Разрешенные локали, доступные приложению
I18n.available_locales = [:en, :pt]

# устанавливаем локаль по умолчанию на что-либо другое, чем :en
I18n.default_locale = :pt
```

Yalantis

Изменение локали

Локаль может быть установлена в *around_action* в ApplicationController:

```
#controllers/application_contoller

around_action :switch_locale

def switch_locale(&action)
  locale = params[:locale] || I18n.default_locale
  I18n.with_locale(locale, &action)
end
```

Yalantis

Определение локали

1. Назначение локали из имени домена (*www.example.com => www.example.es*)

```
around_action :switch_locale

def switch_locale(&action)
  locale = extract_locale_from_tld || I18n.default_locale
  I18n.with_locale(locale, &action)
end

# Получаем локаль из домена верхнего уровня или возвращаем +nil+, если такая локаль недоступна
# Вам следует поместить что-то наподобие этого:
#   127.0.0.1 application.com
#   127.0.0.1 application.it
#   127.0.0.1 application.pl
# в ваш файл /etc/hosts, чтобы попробовать это локально
def extract_locale_from_tld
  parsed_locale = request.host.split('.').last
  I18n.available_locales.map(&:to_s).include?(parsed_locale) ? parsed_locale : nil
end
```

Yalantis

Определение локали

2. Назначение локали из параметров URL

Добавить default_url:

```
# app/controllers/application_controller.rb
def default_url_options
  { locale: I18n.locale }
end
```

Результат: каждый метод хелпера, зависимый от *url_for* теперь будут автоматически включать локаль в строку запроса.

Yalantis

Определение локали

3. Указание локали из пользовательских настроек

```
around_action :switch_locale

def switch_locale(&action)
  locale = current_user.try(:locale) || I18n.default_locale
  I18n.with_locale(locale, &action)
end
```

Yalantis

Определение локали

4. Определение локали из языка заголовка

```
def switch_locale(&action)
  logger.debug "* Accept-Language: #{request.env['HTTP_ACCEPT_LANGUAGE']}"
  locale = extract_locale_from_accept_language_header
  logger.debug "* Locale set to '#{locale}'"
  I18n.with_locale(locale, &action)
end

private
  def extract_locale_from_accept_language_header
    request.env['HTTP_ACCEPT_LANGUAGE'].scan(/^[a-z]{2}/).first
  end
```

Yalantis

Определение локали

4. Определение локали по IP геолокаци
5. Хранение локали в сессии или куки

Yalantis

Интернационализация и локализация

```
# app/controllers/home_controller.rb
class HomeController < ApplicationController
  def index
    flash[:notice] = "Hello Flash"
  end
end
```

```
# app/controllers/application_controller.rb
class ApplicationController < ActionController::Base
  around_action :switch_locale

  def switch_locale(&action)
          locale = params[:locale] || I18n.default_locale
          I18n.with_locale(locale, &action)
  end
end
```


```
# app/controllers/home_controller.rb
class HomeController < ApplicationController
  def index
    flash[:notice] = "Hello Flash"
  end
end
```


```
# app/views/home/index.html.erb
<h1>Hello World</h1>
<p><%= flash[:notice] %></p>
```

Yalantis

Абстракция локализованного кода

```
# app/controllers/home_controller.rb
class HomeController < ApplicationController
  def index
    flash[:notice] = t(:hello_flash)
  end
end
```

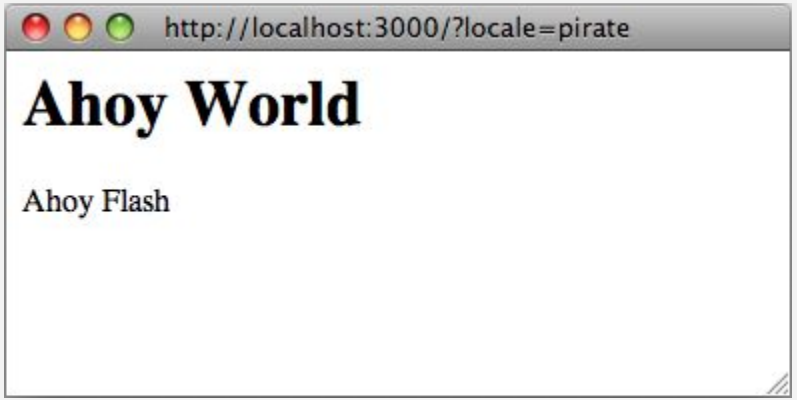
```
# app/views/home/index.html.erb
<h1><%= t :hello_world %></h1>
<p><%= flash[:notice] %></p>
```

Yalantis

Предоставление переводов для строк

```
# config/locales/en.yml
en:
  hello_world: Hello world!
  hello_flash: Hello flash!

# config/locales/pirate.yml
pirate:
  hello_world: Ahoy World
  hello_flash: Ahoy Flash
```



Yalantis

Передача переменных

```
# app/views/products/show.html.erb
<%= t('product_price', price: @product.price) %>
```

```
# config/locales/en.yml
en:
  product_price: "$%{price}"

# config/locales/es.yml
es:
  product_price: "%{price} €"
```

Yalantis

Спасибо за внимание!