Yalantis

Лекция 15: Настройка Rails

Курс лекций по основами web-разработки на языке программирования Ruby

Подробнее о rails new

```
rails new --help
Usage:
  rails new APP_PATH [options]
Options:
      [--skip-namespace], [--no-skip-namespace]            # Skip namespace (affects only isolated applications)
  -r, [--ruby=PATH]                                        # Path to the Ruby binary of your choice
                                                           # Default: /Users/yevhenii/.rbenv/versions/2.6.5/bin/ruby
  -m, [--template=TEMPLATE]                                # Path to some application template (can be a filesystem path or URL)
  -d, [--database=DATABASE]                                # Preconfigure for selected database
                                                           # Default: sqlite3
      [--skip-gemfile], [--no-skip-gemfile]                # Don't create a Gemfile
  -G, [--skip-git], [--no-skip-git]                        # Skip .gitignore file
      [--skip-keeps], [--no-skip-keeps]                    # Skip source control .keep files
  -M, [--skip-action-mailer], [--no-skip-action-mailer]    # Skip Action Mailer files
      [--skip-action-mailbox], [--no-skip-action-mailbox]  # Skip Action Mailbox gem
      [--skip-action-text], [--no-skip-action-text]        # Skip Action Text gem
  -O, [--skip-active-record], [--no-skip-active-record]    # Skip Active Record files
      [--skip-active-storage], [--no-skip-active-storage]  # Skip Active Storage files
  -P, [--skip-puma], [--no-skip-puma]                      # Skip Puma related files
  -C, [--skip-action-cable], [--no-skip-action-cable]      # Skip Action Cable files
  -S, [--skip-sprockets], [--no-skip-sprockets]            # Skip Sprockets files
      [--skip-spring], [--no-skip-spring]                  # Don't install Spring application preloader
      [--skip-listen], [--no-skip-listen]                  # Don't generate configuration that depends on the listen gem
  -J, [--skip-javascript], [--no-skip-javascript]          # Skip JavaScript files
      [--skip-turbolinks], [--no-skip-turbolinks]          # Skip turbolinks gem
  -T, [--skip-test], [--no-skip-test]                      # Skip test files
      [--skip-system-test], [--no-skip-system-test]        # Skip system test files
      [--skip-bootsnap], [--no-skip-bootsnap]              # Skip bootsnap gem
      [--dev], [--no-dev]                                  # Setup the application with Gemfile pointing to your Rails checkout
      [--edge], [--no-edge]                                # Setup the application with Gemfile pointing to Rails repository
      [--rc=RC]                                            # Path to file containing extra configuration options for rails command
      [--no-rc], [--no-no-rc]                              # Skip loading of extra configuration options from .railsrc file
      [--api], [--no-api]                                  # Preconfigure smaller stack for API only apps
  -B, [--skip-bundle], [--no-skip-bundle]                  # Don't run bundle install
  --webpacker, [--webpack=WEBPACK]                         # Preconfigure Webpack with a particular framework (options: react, vue, angular, elm, stimulus)
      [--skip-webpack-install], [--no-skip-webpack-install] # Don't run Webpack install
```

Yalantis

config/application.rb

```
require_relative 'boot'

require "rails"
# Pick the frameworks you want:
require "active_model/railtie"
require "active_job/railtie"
require "active_record/railtie"
require "active_storage/engine"
require "action_controller/railtie"
require "action_mailer/railtie"
require "action_mailbox/engine"
require "action_text/engine"
require "action_view/railtie"
require "action_cable/engine"
# require "sprockets/railtie"
# require "rails/test_unit/railtie"

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(*Rails.groups)

module TestApi
 class Application < Rails::Application
   # Initialize configuration defaults for originally generated Rails version.
   config.load_defaults 6.0
 # Settings in config/environments/* take precedence over those specified here.
    # Application configuration can go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded after loading
    # the framework and any gems in your application.

   config.api_only = true
 end
end
```

Yalantis

config/initializer/cors.rb

```
# Avoid CORS issues when API is called from the frontend app.
# Handle Cross-Origin Resource Sharing (CORS) in order to accept cross-origin AJAX requests.

# Read more: https://github.com/cyu/rack-cors


Rails.application.config.middleware.insert_before 0, Rack::Cors do
  allow do
    origins ENV['FRONTEND_ORIGIN']

    resource '*',
             headers: :any,
             methods: %i[get post put patch delete options head],
  end
end
```

Yalantis

config/credentials.yml.enc

Пример файла

```
b3/TaEqIAFdgrCC05HXKioIHtB+G3aC/IELP3a61aY5VTEVh/A6cNs/imxdekoeQoQk4nnkUvYyNQJmxIzK16hGw4eD1QX+0djbgzlrDUAeYMTT6AuTiUV16pIyEqnjL7bx4OLO0b7sfQz01+8+F
g87LkDbtRkLn7XELacZJPa9XiHNSXwiKZU+5N3m0pll+p6z37Ntfmiai0yY/xMwmsq5rmpu/OyUSWEFQsp/hPS5MqQ0PbR3ZayVxIXJZmjZzd0UmCIBEXodp0/2PnJOSjLxFP8qQfe3Rcy8firjo4
d5doIj0bSQR20zg+UModgDSxWQjABZP+Qy1K3iG45VsDO8R/QfGOoFkKG/HRNwunUGN92f4SZY+J/4R15XYU7bV7UAfwvBJeqdEoPNeLpwZO4Rres7WNe/XzJhL--qrJQ4G0OThjDezvL--
S5S87SJD5KrydabWgAXVNg==
```

Для чтения нужен **config/master.key**

```
5eaff7b631f487b8990fc895a9f33de6
```

EDITOR=nano bin/rails credentials:edit

```
production:
  aws:
    access_key_id: 123
    secret_access_key: 345

Использование в коде:
Rails.application.credentials.production[:aws][:secret_access_key].
```

Yalantis

Dotenv

https://rubygems.org/gems/dotenv

.env

```
AWS_ACCESS_KEY_ID=123
AWS_SECRET_ACCESS_KEY=12345
AWS_REGION=eu-central-1

Использование в коде
ENV[‘AWS_ACCESS_KEY_ID’]
```

.env.sample

```
AWS_ACCESS_KEY_ID=*****
AWS_SECRET_ACCESS_KEY=*****
AWS_REGION=*****
```

Dry-Container

Brief Example

```
container = Dry::Container.new
container.register(:parrot) { |a| puts a }

parrot = container.resolve(:parrot)
parrot.call("Hello World")
# Hello World
# => nil
```

Yalantis

Dry-Container применение

config/initializers/container.rb

```
require 'aws-sdk-sns'

Container = Dry::Container.new

aws_sns_client =
  if ENV['RAILS_ENV'] == 'test'
    require 'account_management/stubs/aws_sns'
   Stubs::AwsSns.new
  else
    creds = Aws::Credentials.new(
      ENV.fetch('AWS_SNS_ACCESS_KEY_ID'),
      ENV.fetch('AWS_SNS_SECRET_ACCESS_KEY')
    )

    Aws::SNS::Client.new(region: ENV.fetch('AWS_SNS_REGION'), credentials: creds)
  end

Container.register(:sms_service, aws_sns_client)
```

Using the same

```
Container.resolve(:sms_service)
```

Yalantis

Sentry

https://github.com/getsentry/raven-ruby/

```
config/initializers/sentry.rb

Raven.configure do |config|
  config.environments = %w[production]
  config.excluded_exceptions += %w(ActionController::RoutingError ActiveRecord::RecordNotFound)
  config.sanitize_fields = Rails.application.config.filter_parameters.map(&:to_s)
end
```

Gem vs Plugin vs Engine

Whats a gem?

Gem is just a fancy term for a packaged Ruby library. A library is just a bit of code providing a set of functionalities to anyone who integrates it in its code.

The only thing you need to know to understand the difference with an engine is that a gem is pure Ruby code.

What's an engine?

Engines are actually gems. All engines can be gems (if packaged) but not all gems are engines.

Engines are a Ruby on Rails feature. That's where the difference lies. They are meant to work within a Ruby on Rails application which means that they can contains Rails-specific entities: models, controllers, views, migrations and so on.

Yalantis

Engines

Engine - это миниатюрное приложение, предоставляющее функциональность содержащим его приложениям. Приложение Rails - это "прокачанный" engine с классом Rails::Application, унаследовавшим большую часть своего поведения от Rails::Engine.

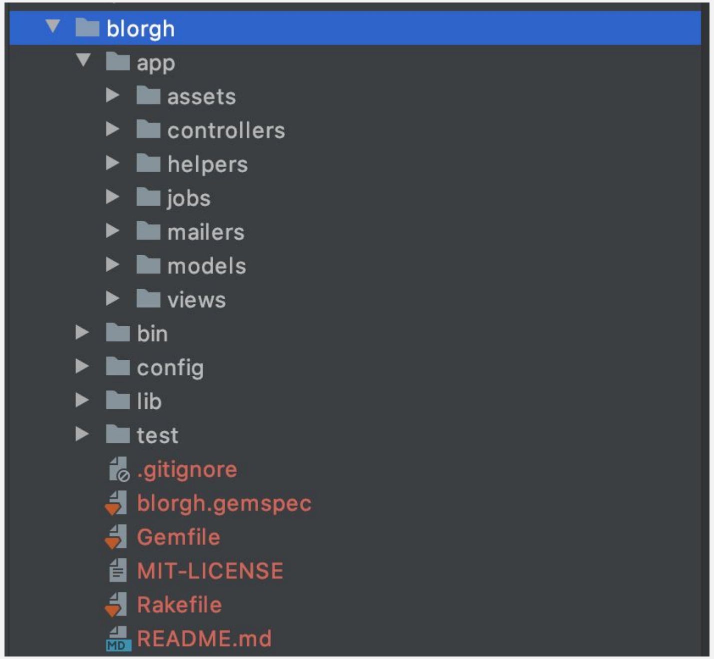
Примеры engines:

1. https://github.com/heartcombo/devise
2. https://github.com/rswag/rswag
3. https://github.com/mperham/sidekiq/wiki/Monitoring
4. https://activeadmin.info/
5. https://github.com/refinery/refinerycms

Yalantis

Generating an engine

```
rails plugin new blorgh --mountable

The full list of options for the plugin generator may be seen by typing:

$ rails plugin --help
```



```
blorgh
    app
        assets
        controllers
        helpers
        jobs
        mailers
        models
        views
    bin
    config
    lib
    test
    .gitignore
    blorgh.gemspec
    Gemfile
    MIT-LICENSE
    Rakefile
    README.md
```

Yalantis

--mountable vs --full

Опция --mountable добавит к опции --full:

Файлы манифестов ресурсов (application.js и application.css)
Пустой ApplicationController в пространстве имен
Пустой ApplicationHelper в пространстве имен
Шаблон макета вьюхи для engine
Изоляцию в пространстве имен для config/routes.rb:

```
Blorgh::Engine.routes.draw do
end
```

Изоляцию в пространстве имен для lib/blorgh/engine.rb:

```
module Blorgh
  class Engine < ::Rails::Engine
    isolate_namespace Blorgh
  end
end
```

Кроме того, опция --mountable сообщает генератору смонтировать engine в пустом тестовом приложении, расположенном в test/dummy, поместив следующую строку в маршрутный файл пустого приложения test/dummy/config/routes.rb:

```
mount Blorgh::Engine => "/blorgh"
```

Подключение engine в приложении

Если engine разрабатывается на локальной машине, необходимо указать опцию :path в Gemfile:

```
gem 'blorgh', path: 'engines/blorgh'
```

Далее нужно установить гем при помощи консольной команды bundle install

Чтобы функциональность engine была доступна в приложении, необходимо его смонтировать в файле config/routes.rb приложения:

```
mount Blorgh::Engine, at: "/blog"
```

Миграции базы данных в engine

Engine содержит миграции, которые необходимо создать в базе данных приложения, чтобы модели engine могли делать правильные запросы к ним. Скопировать миграции в приложение можно посредством этой команды, запущенной из корня приложения:

```
$ rails blorgh:install:migrations
```

Если имеется несколько engine, из которых необходимо скопировать миграции, используйте railties:install:migrations:

```
$ rails railties:install:migrations
```

Эта команда при первом запуске скопирует все миграции из engine. При следующем запуске она скопирует лишь те миграции, которые еще не были скопированы.

Yalantis

Зависимости от других гемов

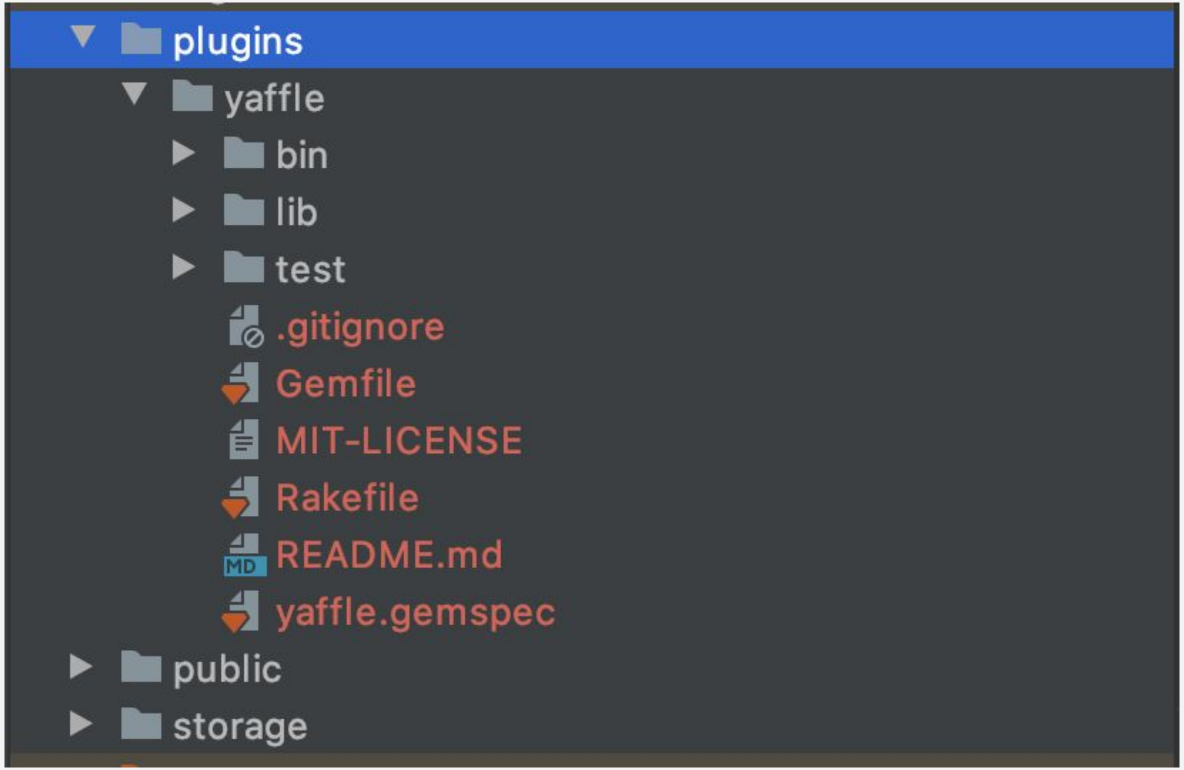
engines/blorgh/blorgh.gemspec

```
$:.push File.expand_path("lib", __dir__)

# Maintain your gem's version:
require "blorgh/version"

# Describe your gem and declare its dependencies:
Gem::Specification.new do |spec|
  spec.name        = "blorgh"
  spec.version     = Blorgh::VERSION
  spec.authors     = ["some author"]
  spec.email       = ["some_email"]
  spec.homepage    = "TODO"
  spec.summary     = "TODO: Summary of Blorgh."
  spec.description = "TODO: Description of Blorgh."
  spec.license     = "MIT"

  # Prevent pushing this gem to RubyGems.org. To allow pushes either set the 'allowed_push_host'
  # to allow pushing to a single host or delete this section to allow pushing to any host.
  if spec.respond_to?(:metadata)
    spec.metadata["allowed_push_host"] = "TODO: Set to 'http://mygemserver.com'"
  else
    raise "RubyGems 2.0 or newer is required to protect against " \
      "public gem pushes."
  end

  spec.files = Dir["{app,config,db,lib}/**/*", "MIT-LICENSE", "Rakefile", "README.md"]

  spec.add_dependency "rails", "~> 6.0.0"

  spec.add_development_dependency "sqlite3"
end
```

Yalantis

Generate a gemified plugin

```
rails plugin new yaffle
```



```
▼ plugins
  ▼ yaffle
    ▶ bin
    ▶ lib
    ▶ test
      .gitignore
      Gemfile
      MIT-LICENSE
      Rakefile
      README.md
      yaffle.gemspec
▶ public
▶ storage
```

Yalantis

Add an "acts_as" Method to Active Record

A common pattern in plugins is to add a method called acts_as_something to models.

```ruby
# yaffle/lib/yaffle/acts_as_yaffle.rb

module Yaffle
  module ActsAsYaffle
    extend ActiveSupport::Concern

    class_methods do
      def acts_as_yaffle(options = {})
        cattr_accessor :yaffle_text_field, default: (options[:yaffle_text_field] || :last_squawk).to_s
      end
    end
  end
end

# test/dummy/app/models/application_record.rb

class ApplicationRecord < ActiveRecord::Base
  include Yaffle::ActsAsYaffle

  self.abstract_class = true
end
```

RDoc Documentation

Once installed, you can create documentation using the rdoc command

```
$ rdoc [options] [names...]
```

For an up-to-date option summary, type

```
$ rdoc --help
```

A typical use might be to generate documentation for a package of Ruby source (such as RDoc itself).

```
$ rdoc
```

This command generates documentation for all the Ruby and C source files in and below the current directory. These will be stored in a documentation tree starting in the subdirectory doc.

You can make this slightly more useful for your readers by having the index page contain the documentation for the primary file. In our case, we could type

```
% rdoc --main README.rdoc
```

Yalantis

Rails Autoloading

In a normal Ruby program, dependencies need to be loaded by hand. For example, the following controller uses classes ApplicationController and Post, and normally you'd need to put require calls for them:

```
# DO NOT DO THIS.
require "application_controller"
require "post"
# DO NOT DO THIS.


class PostsController < ApplicationController
  def index
    @posts = Post.all
  end
end
```

The autoloading zeitwerk mode is enabled by default in Rails 6 applications running on CRuby:

```
# config/application.rb
config.load_defaults "6.0" # enables zeitwerk mode in CRuby
```

Yalantis

Project Structure

In a Rails application file names have to match the constants they define, with directories acting as namespaces.

For example, the file app/helpers/users_helper.rb should define UsersHelper and the file app/controllers/admin/payments_controller.rb should define Admin::PaymentsController.

By default, Rails configures Zeitwerk to inflect file names with String#camelize. For example, it expects that app/controllers/users_controller.rb defines the constant UsersController because

```
"users_controller".camelize # => UsersController
```

Yalantis

Nesting

```
module XML
  class SAXParser
    # (1)
  end
end
```

The nesting at any given place is the collection of enclosing nested class and module objects outwards. The nesting at any given place can be inspected with Module.nesting.

```
[XML::SAXParser, XML]
```

```
class XML::SAXParser
  # (2)
end
```

```
[XML::SAXParser]
```

Yalantis

Eager Loading

In production-like environments it is generally better to load all the application code when the application boots. Eager loading puts everything in memory ready to serve requests right away, and it is also CoW-friendly.

Eager loading is controlled by the flag config.eager_load, which is enabled by default in production mode.

The order in which files are eager loaded is undefined.

Yalantis

Resolution Algorithm for Relative Constants

В любом месте кода определяем cref как первый элемент вложения, если он не пустой, или Object в противном случае.

Алгоритм поиска относительных константных ссылок выглядит следующим образом:

Если вложение не пустое, постоянная ищется в ее элементах и в порядке. Предки этих элементов игнорируются.

Если не найден, алгоритм идет вверх по цепочке предков cref.

Если не найден, а cref является модулем, константа ищется в Object.

Если не найдено, const_missing вызывается в cref. Реализация по умолчанию const_missing вызывает NameError, но его можно переопределить.

Yalantis

Что почитать?

1. https://guides.rubyonrails.org/initialization.html
2. https://github.com/bkeepers/dotenv/
3. https://dry-rb.org/gems/dry-container/0.8/
4. https://github.com/getsentry/raven-ruby/
5. https://guides.rubyonrails.org/engines.html
6. https://www.hocnest.com/blog/testing-an-engine-with-rspec/
7. https://guides.rubyonrails.org/plugins.html
8. https://edgeguides.rubyonrails.org/autoloading_and_reloading_constants.html
9. https://github.com/fxn/zeitwerk
10. https://edgeguides.rubyonrails.org/autoloading_and_reloading_constants_classic_mode.html
11. https://guides.rubyonrails.org/configuring.html
12. https://www.oreilly.com/library/view/component-based-rails-applications/9780134774596/

Yalantis

Thanks!

Any questions? Feel free to contact us hello@yalantis.com