# Set up Ruby Project

# RORLab 77번째 모임

## 2014. 12. 23.
## 2015. 8. 31. update

# Daekwon Kim

## SMARTSTUDY Software & System Engineer

@nacyo_t

가 각 간 갓

나 낙 난 낫

다 닥 단 닷

ㄱ ㄴ ㅅ

(ㄱ,ㄴ,ㅅ)x(가,나,다)

# 낚읏

# Ruby On Rails

# rails new

```
$ rails new <PROJECT_NAME>
```

# 프로젝트 트리

```
$ tree -d -L 2
.
├── app
│   ├── assets
│   ├── controllers
│   ├── models
│   └── views
├── bin
├── config
│   ├── environments
│   ├── initializers
├── db
│   └── migrate
├── lib
├── log
├── public
├── spec
│   ├── controllers
│   ├── models
└── tmp
```

# 프로젝트 트리(해설 첨부)

```
$ tree -d -L 2
.
├── app                 # 어플리케이션 코드
│   ├── assets          # 에셋
│   ├── controllers     # 컨트롤러
│   ├── models          # 모델
│   └── views           # 뷰
├── bin                 # 실행 파일
├── config              # 설정
│   ├── environments    # 환경별 설정
│   ├── initializers    # 어플리케이션 초기화 스크립트
├── db                  # 데이터베이스 관련
│   └── migrate         # 마이그레이션
├── lib                 # 라이브러리
├── log                 # 로그
├── public              # 정적 파일
├── spec                # 테스트
│   ├── controllers     # 컨트롤러 테스트
│   ├── models          # 모델 테스트
└── tmp                 # 임시 파일
```

# 스스로 자신의 위치를 알고 있음

# 레일스는 프레임워크

# 루비

# 레일스는 프레임워크

# 루비도 프레임워크?

# ruby new

```
$ mkdir my_awesome_ruby_project
```

# プロジェクト トリ

```
$ tree -d -L 2
.
├── foo/
├── bar/
├── my_library/
└── awesome.rb
```

# 자유!

# 하지만, 관습적인 구조가 존재

```
.
├── lib
├── test
├── Gemfile
├── Rakefile
└── README.md
```

# 직접 만들어보자!

# 일단 README부터 만들기

```
$ touch README.md
```

읽어보기 : Readme Driven Development

# lib/ 디렉토리 만들기

```
$ mkdir lib
```

# 라이브러리 파일 만들기

```
$ touch lib/my_awesome_ruby_project.rb
```

```
# lib/my_awesome_ruby_project.rb
class MyAwesomeRubyProject
end
```

```
require './lib/my_awesome_ruby_project/awesome_cat'
```

# 네임스페이스

```
$ mkdir lib/my_awesome_ruby_project
$ touch lib/my_awesome_ruby_project/awesome_cat.rb
```

```
# lib/my_awesome_ruby_project_awesome_cat.rb
class MyAwesomeRubyProject::AwesomeCat
  def mew
    "Nyaa~"
  end
end
```

# ライブラリ 사용하기

```
[1] pry(main)> require('./lib/my_awesome_ruby_project')
=> true
[2] pry(main)> my_cat = MyAwesomeRubyProject::AwesomeCat.new
=> #<MyAwesomeRubyProject::AwesomeCat:0x007f96f29e6ef8>
[3] pry(main)> my_cat.mew
=> "Nyaa~"
```

# 다시 살펴보기

```
.
├── lib
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat.rb
│   └── my_awesome_ruby_project.rb
└── README.md
```

# Test 환경 구축하기

# RSpec

```
$ bundle exec rspec
/Users/... in `block in replace_gem':
rspec-core is not part of the bundle.
Add it to Gemfile. (Gem::LoadError)
from /Users/...:22:in `<main>'
```

# Bundler

# 의존성 관리 도구

# 의존성

# 프로젝트에서 사용하는 라이브러리

# Gemfile

# 의존성 선언 파일

# Gemfile 생성하기

```
$ bundle init
Writing new Gemfile to /.../my_awesome_ruby_project/Gemfile
```

```
# Gemfile
source "https://rubygems.org"
```

# Gemfile에 rspec 추가하기

```
source "https://rubygems.org"

gem "rspec"
```

# bundle install

```
$ bundle install
Fetching gem metadata from https://rubygems.org/.........
Resolving dependencies...
Using diff-lcs 1.2.5
Using rspec-support 3.1.2
Using rspec-core 3.1.7
Using rspec-expectations 3.1.2
Using rspec-mocks 3.1.3
Installing rspec 3.1.0
Using bundler 1.7.3
Your bundle is complete!
Use `bundle show [gemname]` to see where a bundled gem is installed.
```

# 테스트 실행하기

```
$ bundle exec rspec
No examples found.

Finished in 0.00017 seconds (files took 0.04335 seconds to load)
0 examples, 0 failures
```

# No examples found

# 테스트 준비하기

```
$ mkdir -p spec/my_awesome_ruby_project/
$ touch spec/spec_helper.rb
$ touch spec/my_awesome_ruby_project/awesome_cat_spec.rb
```

```
# spec/spec_helper.rb
require './lib/my_awesome_ruby_project'
```

# 첫 (가짜) 테스트 작성하기

```
# spec/my_awesome_ruby_project/awesome_cat_spec.rb
require './spec/spec_helper'

describe MyAwesomeRubyProject::AwesomeCat do
  it "My first awesome test"
end
```

# 테스트 실행하기

```
$ bunedl exec rspec
Pending:
  MyAwesomeRubyProject::AwesomeCat My first awesome test
    # Not yet implemented
    # ./spec/my_awesome_ruby_project/awesome_cat_spec.rb:4

Finished in 0.00029 seconds (files took 0.10246 seconds to load)
1 example, 0 failures, 1 pending
```

# 첫 (진짜) 테스트 작성하기

```
require './spec/spec_helper'

describe MyAwesomeRubyProject::AwesomeCat do
  it "My first awesome test" do
    expect(MyAwesomeRubyProject::AwesomeCat.new.mew).to eq "Nyaa~"
  end
end
```

# 다시, 테스트 실행하기

```
$ rspec --color --format doc
MyAwesomeRubyProject::AwesomeCat
  My first awesome test

Finished in 0.00086 seconds (files took 0.09874 seconds to load)
1 example, 0 failures
```

# .rspec

```
--color
--format doc
```

# 여기까지 구조

```
.
├── lib
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat.rb
│   └── my_awesome_ruby_project.rb
├── spec
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat_spec.rb
│   └── spec_helper.rb
├── .rspec
├── Gemfile
├── Gemfile.lock
└── README.md
```

# 적절한 도구를 활용한 Test Driven Development

# Guard

*Guard is a command line tool to easily handle events on file system modifications.*

# bundler에 Guard 추가하기

```
# Gemfile
source "https://rubygems.org"

gem "rspec"
gem "guard"
gem "guard-rspec"
```

# 의존성 설치하기

```
$ bundle install
```

# Guardfile 생성하기

```
$ bundl exec guard init
20:14:38 - INFO - Writing new Guardfile to /Users/.../Guardfile
20:14:38 - INFO - rspec guard added to Guardfile, feel free to edit it
```

# Guardfile 직접 만들기

```
# Guardfile
guard :rspec, cmd: 'bundle exec rspec' do
  watch(%r{^spec/.+_spec\.rb$})
  watch(%r{^lib/(.+)\.rb$}) { |m| "spec/#{m[1]}_spec.rb" }
  watch('spec/spec_helper.rb') { "spec" }
end
```

# Test Driven Demo

# 최종 구조

```
.
├── lib
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat.rb
│   └── my_awesome_ruby_project.rb
├── spec
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat_spec.rb
│   └── spec_helper.rb
├── .rspec
├── Gemfile
├── Gemfile.lock
├── Guardfile
└── README.md
```

# 최종 구조(해설)

```
.
├── lib                                          # 라이브러리 코드
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat.rb
│   └── my_awesome_ruby_project.rb
├── spec                                         # 테스트
│   ├── my_awesome_ruby_project
│   │   └── awesome_cat_spec.rb
│   └── spec_helper.rb                           # 테스트 관련 설정 파일
├── .rspec                                       # rspec 설정 파일
├── Gemfile                                      # 의존성 선언 파일
├── Gemfile.lock                                 # 의존성 고정 파일
├── Guardfile                                    # Guard 설정 파일
└── README.md                                    # README(프로젝트 설정)
```

# 일반적인 루비 프로젝트 구조

```
.
├── bin/              # 실행 파일
├── lib/              # 라이브러리 코드
├── spec/             # 테스트 코드
├── tmp/              # 임시 파일
├── Gemfile           # Bundler 패키지 선언 파일
├── Guardfile
├── Rakefile          # Rake Task 정의 파일
├── LICENSE.txt       # 라이센스 파일
└── README.md         # README 파일
```

# Thank you!

## @nacyo_t