Testing - Asercje

assert

assert_nil

assert_not_nil

flunk

assert_nothing_raised

assert_raise

assert_instance_of

assert_kind_of

assert_respond_to

assert_not_equal

assert_equal

assert_in_delta

assert_no_math

assert_match

assert_same

assert_not_same

assert_operator

assert_throws

assert_send

Gems & Bundler

RubyGems – zarządzają bibliotekami ruby dostępnymi w danej instalacji interpretera (np. ruby 1.9.3)

Bundler –

- izoluje gemsety w poszczególnych projektach
- zarządza ich wzajemnymi relacjami (requirements)
- zapewnia jednakową wersję bibliotek dla wszystkich osób pracujących nad danym projektem (git add Gemfile.lock !!!)

Gems & Bundler

- gem list
- gem install <nazwa> [-v wersja]
- gem install nokogiri -v 1.4.2
- gem uninstall
- bundle install <nazwa>
- bundle update <nazwa>
- bundle update
- bundle exec <komenda>

Struktura aplikacji Rails

- **app** <= tu znajduje się kod główna część kodu aplikacji – MVCś
  - **controllers**
  - **views**
  - **models**
  - **helpers** <= moduły pomocnicze – są automatycznie dołączane do widoków
  - **assets** <= od rails 3.1 tutaj znajdują się pliki, z których generowane są js oraz css, obrazki oraz inne statyczne zasoby serwowane przez serwer (poprzez asset pipeline)
- **config** <= pliki konfiguracyjne (baza danych, inicjalizacja bibliotek, routing oraz deploy)
- **db** <= migracje (tam, gdzie jest ORM)
- **lib** <= taski rake oraz wszelkie biblioteki i patche, które nie znalazły miejsca gdzie indziej
- **log**
- **public** <= root serwera www, do niedawna tu można było znaleźć wszystkie pliki js i css, zmieniło się to wraz z wprowadzeniem asset pipeline
- **script** <= skrypty startowe aplikacji (obecnie wystarczy po prostu 'rails')
- **test**
  - unit
  - functional
  - integration
  - performance
  - fixtures
  - ...
- **tmp**
- **vendor** <= tutaj trzymane są zewnętrzne biblioteki, które należy dołączyć bezpośrednio do projektu (np. nie są opublikowane w katalogu rubygems)

Kontrolery

- przetwarzanie żądań
- filtry
- renderowanie widoków
- przekierowania

REST

• index	/posts.xml	GET	Zwróc wszystkie elementy
• show	/posts/1.xml	GET	Zwróc jeden element o id = 1
• create	/posts.xml	POST	Utwórz element
• update	/posts/1.xml	PUT	Modyfikuj element o id = 1
• delete	/posts/1.xml	DELETE	Usuń element o id = 1

Widoki

- szablony o ściście określonej hierarchii
- layout -> widok -> partiale
- domyślny silnik => erb
- dużo lepiej używać #haml

Modele

- ORM (poprzez bibliotekę Arel)
- scopes
- callbacks
- associations
  - belongs_to
  - has_one
  - has_many
  - habtm, has_many :through

Ajax w Rails

- javascript (prawie) bez użycia javascript – wszystko jest opakowane w helpery
- :remote => true w funkcjach link_to oraz form_for
- respond_to + respond_with w kontrolerach
- szablony js (np. views/posts/create.js.erb)