Ruby i Ruby on Rails

Mateusz Drożdżyński

Co to Ruby?

- Dynamiczny, obiektowy język programowania, stworzony przez Yukihiro Matsumoto
- Pojawił się w tym samym roku co Java (1995)
- Przejrzysta, prosta i intuicyjna składnia
- Wywodzi się z Lispa, Perla, Smalltalka
- Ma wiele cech wspólnych z Pythonem
- Metaprogramming

Prosta składnia

- Elementy składni zaczerpnięte z języków takich jak Perl, Lisp, itp.
- Kod zrozumiały dla każdego
- Kod w wielu przypadkach jest odzwierciedleniem zdań odpowiadających danej operacji
- Keep It Simple, Stupid!

Pełna obiektowość

- Obiektami w Ruby są:
  - liczby
  - napisy
  - klasy (!)
  - metody (!)
  - ... a nawet sam dokument!

Hello World!

Klasy i metody

Blok, Proc i Lambda

- Wszystkie implementują przekazywanie bloków kodu i ich wykonywanie
- Jakie są podobieństwa?
- Jakie są różnice?
- Znane z innych języków, np.: Lambda z Lispa czy metody anionimowe z Javy

Instrukcje warunkowe

Wyrażenia regularne

Otwarte klasy, czyli jak dynamicznie ułatwiać sobie życie

Moduły, dziedziczenie i Mixins

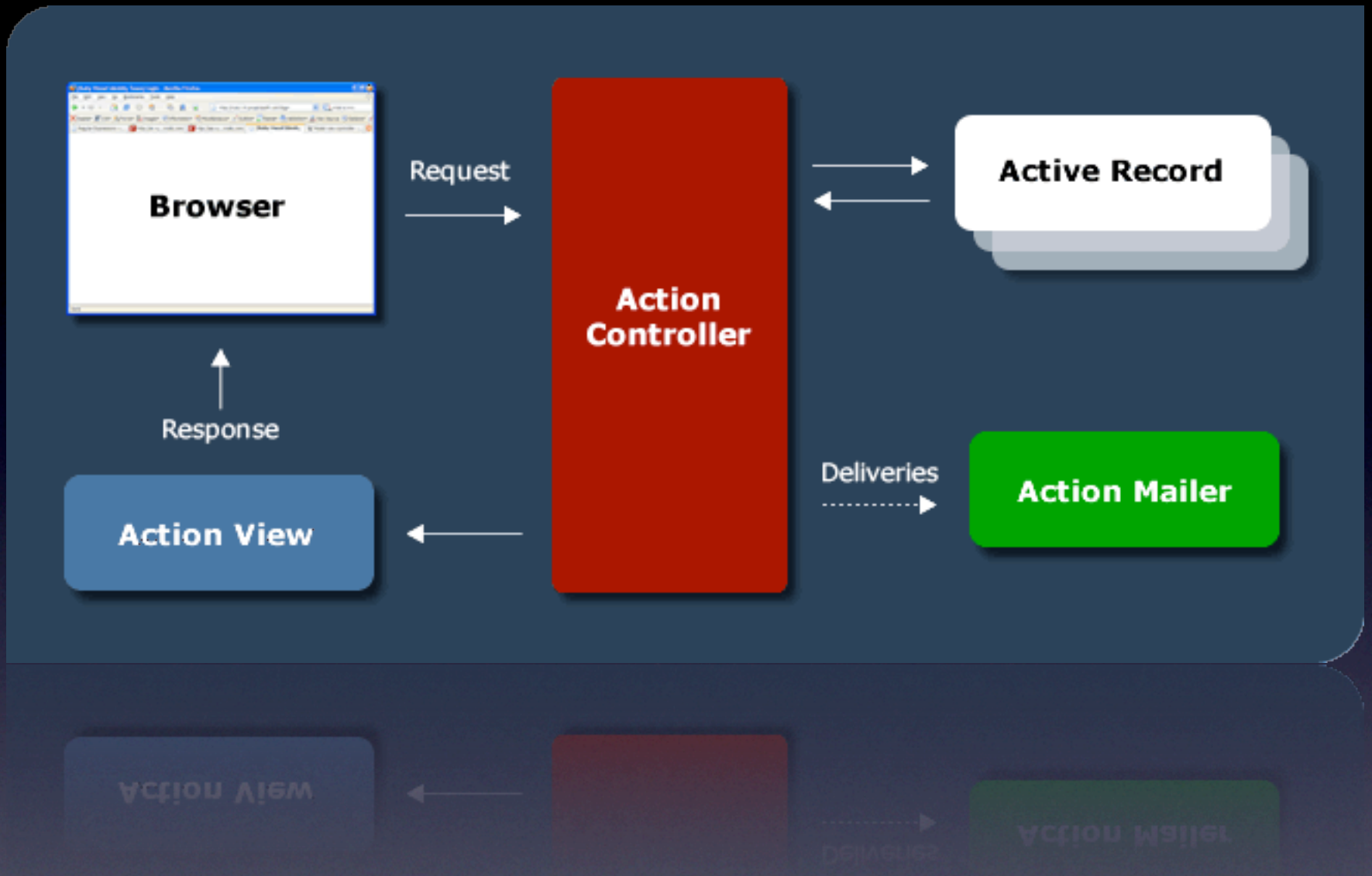
- Moduł jako Namespace
- Dziedziczenie klas
- Moduł jako Mixin, czyli sposób na dziedziczenie po wielu klasach w Ruby

Ruby on Rails

- Framework oparty o wzorzec Model-View-Controller, umożliwiający szybkie tworzenie aplikacji internetowych
- Łatwy i zrozumiały kod dzięki przejrzystej składni języka Ruby
- Łatwe programowanie API dzięki simply_restful
- Liczne rozszerzenia ułatwiające i przyspieszające implementację szablonowych rozwiązań

Części składowe

- Active Record
- Action Pack:
  - Action Controller
  - Action View
- Action Mailer
- Action Web Service



Schemat działania

Active Record, czyli Model w stylu RoR

- Automatyczne mapowanie tabel baz danych w obiekty
- Ułatwia zarówno proste, jak i bardziej skomplikowane operacje na bazie danych
- Obsługa wielu typów baz danych, w tym MySQL, PostgreSQL, SQLite czy bazy Oracle.
- Każdy model to klasa, można więc dopisywać do niego dowolne metody

Action Controller

- Odpowiedzialny za obsługę zapytań użytkownika/innej aplikacji
- Metody to akcje kontrolera
- Parametry żądania są przekazywane poprzez metodę params[]
- Zmienne instancji są przekazywane do widoku
- ApplicationController, czyli jak definiować wpólne metody dla wszystkich kontrolerów
- Filtry

Action View

- Łatwe tworzenie widoków, dzięki wbudowaniu w szablony języka Ruby
- Możliwość tworzenia kilku widoków dla jednej akcji:
  - osobny widok renderuje XHTML
  - inny XML dla zewnętrznej aplikacji
  - a trzeci feed RSS dla użytkowników
  - ... wszystko w jednej akcji!

Scaffolding, czyli generatory dla leniwych

- Generowanie:
  - migracji
  - modeli
  - kontrolerów
  - widoków

Tworzymy pierwszą aplikację

- Prywatny system zakładek a'la Del.icio.us
  - CRUD
  - Tagi
  - Eksport do XML
  - Wyszukiwarka

Polecenie `rails`, czyli jak zacząć

Konfiguracja bazy danych

Konsola, czyli bez czego nie można się objeść

- Pozwala na wykonywanie dowolnego kodu Ruby
- Ładuje całe środowisko naszej aplikacji (modele, itd.)
- Pozwala na szybkie testowanie i debugging
- Umożliwia uruchamianie różnych środowisk (production, development)
- ./script/console

Generowanie części aplikacji

- ./script/generate migration ...
- ./script/generate model ...
- ./script/generate controller ...

Scaffolding

- Tworzymy model i migrację:
  - ./script/generate model ...
- Tworzymy szkielet akcji (scaffold) dla modelu:
  - ./script/generate scaffold ...

Simply Restful

- REST, czyli jak wykorzystać dodatkowe możliwości protokołu HTTP
  - Metody GET, POST, PUT, DELETE i ich znaczenie w zwiększeniu semantyczności aplikacji
- Simply Restful, czyli REST w Ruby on Rails

REST a mapowanie URL

Metoda HTTP	URL w REST	Akcja	URL bez REST
GET	/projects/1	show	GET /projects/show/1
DELETE	/projects/1	destroy	GET /projects/destroy/1
PUT	/projects/1	update	POST /projects/1/update
POST	/projects	create	POST /projects/create

Metody _path

Metoda _path	Metoda HTTP	URL	Akcja
projects_path	GET	/projects	index
projects_path(1)	GET	/projects/1	show
new_project_path	GET	/projects/new	new
edit_project_path(1)	GET	/projects/1;edit	edit
projects_path	POST	/projects	create
project_path(1)	PUT	/projects/1	update
project_path(1)	DELETE	/projects/1	destroy

REST a scaffolding

- Metoda ./script/generate scaffold_resource
- Generowane są:
  - Migracje
  - Model
  - Kontroler
  - Testy
  - Fixtures

Walidacje

- Łatwy sposób na sprawdzanie poprawności wprowadzanych danych
- Definiowane w modelu, przez co są dostępne niezależnie od tego, gdzie się do danego modelu odwołujemy
- Przykładowe metody:
  - validates_presence_of
  - validates_uniqueness_of
  - validates_numericallity_of

Relacje między modelami

- Dostępne relacje:
  - has_one
  - belongs_to
  - has_many
  - has_and_belongs_to_many
  - has_many :through

Test-driven development, czyli jak pisać testy

- Zalety test-driven development
  - Unikamy błędów w trakcie pisania aplikacji
  - Koniec sprawdzania każdej funkcji aplikacji w przeglądarce
  - Możemy uruchamiać testy w różnych środowiskach, więc przed każdym release można przetestować aplikację na serwerze produkcyjnym

Rodzaje testów

- Unit Testing
  - sprawdzanie poprawności relacji pomiędzy modelami oraz walidacji
  - sprawdzanie działania metod należących do logiki modelu
- Functional Testing
  - sprawdzanie działania kontrolerów, odwołujemy się do aplikacji jak przeglądarka użytkownika

- Integration Testing
- Performance Testing
  - Klasa Benchmark
- Używanie Mock Objects
  - Prosty sposób na testowanie aplikacji korzystających z zewnętrznych serwisów bez konieczności bezpośredniego łączenia z nimi

Łączenie się z zewnętrzną aplikacją przez REST

- Integralne działanie ActiveResource
- Łatwe tworzenie zarówno aplikacji klienckich, jak i serwerowych
- Pełna integracja z ActiveRecord
- Niestety, nie jest to częścią Rails 1.2.x; dostępne jedynie w Rails Edge

Pisanie własnych zadań Rake

- Rake umożliwia pisanie własnych zadań (zestawów poleceń) dotyczących naszych aplikacji
- Przykładowe wbudowane metody:
  - rake db:migrate
  - rake test
- Tworzymy własne zadanie

Cache w Ruby on Rails

- Mechanizmy dostępne w Ruby on Rails:
  - caches_page
  - caches_action
  - Fragment Cache
- A w drugą stronę...
  - expire_page
  - expire_action
  - ActionController::Caching::Sweeper

Ruby on Rails od strony serwera

- Możliwe sposoby serwowania aplikacji Ruby on Rails:
  - WEBrick
  - CGI
  - FastCGI
  - mod_ruby
  - Mongrel

Przykładowy przebieg zapytania – Mongrel

1. Przeglądarka WWW
2. Serwer HTTP (load balancing)
   - Apache, Lighttpd, nginX, Pound, Pen
3. Jedna z n instancji serwera Mongrel
4. Ruby on Rails

Capistrano, czyli Rails deployment made easy

- Capistrano umożliwia łatwą i kompleksową obsługę serwerów aplikacji i baz danych
- Przydatne polecenia:
  - cap --apply-to /path Application
  - cap setup
  - cap deploy
  - cap deploy_with_migrations
- http://manuals.rubyonrails.com/read/book/17

acts_as_versioned, czyli jak wersjonować modele

- Pełna integracja z ActiveRecord
- Działa bez żadnej dodatkowej integracji z naszej strony
- Pozwala na wyświetlanie poprzednich wersji
  - i cofanie się do nich
- ./script/plugin install acts_as_versioned