

Programowanie w Ruby

Wykład 11

Marcin Młotkowski

19 grudnia 2012

Plan wykładu

Modele danych i bazy danych

Modele danych

Migracje

Bazy danych w Railsach

Rusztowania

Walidacja modeli

Modele danych

- ▶ Definicje danych
- ▶ Odczyt i zapis danych
- ▶ Kontrola poprawności danych
- ▶ Migracje danych

ORM – object-relational mapping

- ▶ Dane są reprezentowane i przetwarzane jako obiekty
- ▶ Dane są przechowywane w relacyjnych bazach danych

Moduł ActiveRecord

- ▶ Definiuje model danych (tj. jego strukturę, związki między danymi, warunki poprawności)
- ▶ Zapewnia 'obiektywne' tworzenie i dostęp do danych (bez udziału SQL'a)
- ▶ Dbą o zgodność modelu ze strukturą w bazie danych w przypadku migracji

Konwencje

Model `Product`

Klasa `Product` w pliku `app/models/product.rb`

W bazie danych jest tabela `Products`

Plik(i) migracyjny(e) `*_create_products.rb`

Plan wykładu

Modele danych i bazy danych

Modele danych

Migracje

Bazy danych w Railsach

Rusztowania

Walidacja modeli

Migracje danych

- ▶ Zmiana struktury danych
- ▶ Na początku struktura jest pusta, więc trzeba wykonać migrację (zerową)
- ▶ Można cofać zmiany w strukturze bazy danych

Szybka migracja

W wierszu poleceń

```
ruby script/generate model product
```

Plik migracyjny *_create_product.rb

```
class CreateProducts < ActiveRecord::Migration
  def self.up
    end

  def self.down
    end
end
```

Metoda self.up

```
create_table :products do |t|
  t.column :przedmiot, :string, :null => false
  t.column :opis      :text
  t.column :cena , :decimal,
    :precision => 10, :scale => 2, :default => 0.01
  t.timestamps
end
```

Klucz główny

- ▶ Rails często potrzebuje klucza głównego
- ▶ Domyślnie zawsze dodawane jest pole `id` będące kluczem

Metoda `self.down`

```
def self.down  
  drop_table :products  
end
```

Kolejne migracje

```
$ ruby script/generate migration add_column
```

Szkielet kolejnej migracji

```
class AddColumn < ActiveRecord::Migration
  def self.up
  end
  def self.down
  end
end
```

Implementacja migracji

```
def self.up
  add_column :products, :qty, :integer
end

def self.down
  remove_column :products, :qty
end
```

Uruchomienie migracji

```
$ rake db:migrate
```

Uruchomienie migracji

```
$ rake db:migrate
```

```
$ rake db:migrate VERSION=5
```

Obsługiwane silniki BD

- ▶ MySQL
- ▶ Sqlite
- ▶ DB2
- ▶ Oracle

Konfiguracja

config/database.yml

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000
test:
  ....
production: ....
```

Początkowa konfiguracja

```
$ rails demo --database=sqlite3
```

Obsługa danych poprzez przeglądarkę

- ▶ zbudowanie kontrolera

```
ruby script/generate controller admin
```

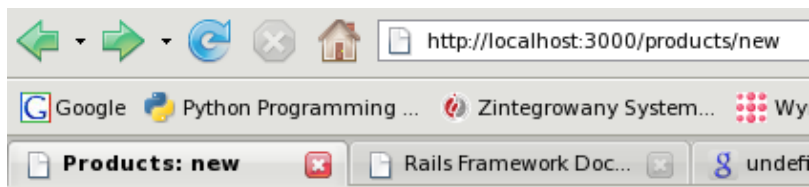
Obsługa danych poprzez przeglądarkę

- ▶ zbudowanie kontrolera

```
ruby script/generate controller admin
```

- ▶ W pliku `app/controllers/admin_controller.rb` trzeba dopisać linię

```
class AdminController < ApplicationController  
  scaffold :product  
end
```



New product

Title

Create

[Back](#)

Uwaga

W wersjach Railsów 2.* nie zadziała!

Plan wykładu

Modele danych i bazy danych

Modele danych

Migracje

Bazy danych w Railsach

Rusztowania

Walidacja modeli

Rusztowanie (scaffold)

Co to jest rusztowanie

wygenerowany automatycznie szkielet służący do operowania na modelu.

Rails wersja 2.*

Można automatycznie wygenerować tabelę, model, a także kontroler z interfejsem do obsługi modelu.

Rails wersja 2.*

Można automatycznie wygenerować tabelę, model, a także kontroler z interfejsem do obsługi modelu.

Przykład

```
$ ruby script/generate scaffold Wyklady title:string opis:text  
punkty:integer
```

Wynik działania skryptu

Przykład

```
$ ruby script/generate scaffold Wyklady title:string opis:text  
punkty:integer
```

Wynik działania skryptu

Przykład

```
$ ruby script/generate scaffold Wyklady title:string opis:text  
punkty:integer
```

Migracja

Powstaje plik migracji, konieczne jest

```
$ rake db:migrate
```

Wynik działania skryptu

Przykład

```
$ ruby script/generate scaffold Wyklady title:string opis:text  
punkty:integer
```

Migracja

Powstaje plik migracji, konieczne jest

```
$ rake db:migrate
```

Kontroler z obsługą dodawania, usuwania i edycji danych modelu (CRUD) i odpowiednimi widokami

Wynik działania skryptu

Przykład

```
$ ruby script/generate scaffold Wyklady title:string opis:text  
punkty:integer
```

Migracja

Powstaje plik migracji, konieczne jest

```
$ rake db:migrate
```

Kontroler z obsługą dodawania, usuwania i edycji danych modelu (CRUD) i odpowiednimi widokami

I jeszcze parę innych rzeczy

Plan wykładu

Modele danych i bazy danych

Modele danych

Migracje

Bazy danych w Railsach

Rusztowania

Walidacja modeli

Model – przypomnienie

- ▶ Modele to podklasy klasy ActiveRecord::Base
- ▶ Każda taka klasa odpowiada tabeli w relacyjnej bazie danych
- ▶ Obiekty tych klas odpowiadają wierszom tych tabel
- ▶ Metody w tych klasach ukrywają szczegóły związane z SQL i bazą danych

Podstawowe operacje na danych

- ▶ Wyszukiwanie, modyfikacja i tworzenie nowych obiektów
- ▶ Kontrola poprawności danych - walidacja

Walidacja — warunki wbudowane

Funkcje standardowe

- ▶ `validates_presence_of :nazwa, :cena`
- ▶ `validates_numericality_of :cena,`
`:message => 'Cena nie jest liczbą', :if => test()`
- ▶ `validates_uniqueness_of :nazwa`

Zastosowanie

```
app/models/wyklady.rb:  
class Wyklady < ActiveRecord::Base  
  
  validates_uniqueness_of :title  
  
end
```

Walidacja własna'

Metoda w klasie modelu:

```
class Wyklady < ActiveRecord::Base
  protected
  def validate
    errors.add(:cena, 'cena powinna być dodatnia')
      if cena.nil? || cena < 0.01
    end
  end
end
```

Plan wykładu

Modele danych i bazy danych

Modele danych

Migracje

Bazy danych w Railsach

Rusztowania

Walidacja modeli



