Programowanie w Ruby

Wykład 11

Marcin Młotkowski

7 stycznia 2019

Plan wykładu

Model

Model

Reprezentacja danych biznesowych i sposobów ich przetwarzania.

Model

Model

Reprezentacja danych biznesowych i sposobów ich przetwarzania.

ActiveRecord

Obiektowa reprezentacja modeli, zapewnienie ich trwałości i zachowań.

Modele danych — praktycznie

- 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 'obiektowe' tworzenie i dostęp do danych (bez udziału SQL'a)
- Dba o zgodność modelu ze strukturą w bazie danych w przypadku migracji

Konwencje

Model Product

Klasa Product w pliku app/models/wyklad.rb

W bazie danych jest tabela Wyklads

Plik(i) migracyjny(e) *_create_wyklads.rb

```
class Wyklad < ApplicationRecord

end
```

```
class Wyklad < ApplicationRecord
    self.table_name = "Wyklady"
end
```

Plan wykładu

Migracja

Migracja

Modyfikacja struktury danych.

Po co migracje

- Na początku struktura jest pusta, więc trzeba wykonać migrację (zerową);
- dostosowywanie struktury danych do potrzeb;
- można cofać zmiany w strukturze bazy danych.

Utworzenie modelu wraz z migracją

W wierszu poleceń

```
bin/rails generate model Przyklad
```

Tworzy model wraz z pierwszą migracją.

Utworzenie modelu wraz z migracją

W wierszu poleceń

```
bin/rails generate model Przyklad nazwa: string opis:text
```

Tworzy model wraz z pierwszą migracją.

Uruchomienie

```
bin/rails db:migrate
```

Klucz główny

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

Inna postać pliku migracyjnego

```
class CreateProducts < ApplicationRecord
    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
```

Metoda self.down

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

Kolejne migracje

Kolejność migracji jest ustalana na podstawie prefiksu pliku migracji z czasem.

Kolejne migracje

Kolejność migracji jest ustalana na podstawie prefiksu pliku migracji z czasem.

Zamiast liczyć czas można użyć generatora migracji.

```
bin/rails generate migration AddProwadzacyToZajecia
prowadzacy:string
```

Uruchomienie migracji

```
$ rake db:migrate
```

Uruchomienie migracji

```
$ rake db:migrate
```

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

Uruchomienie migracji

```
$ rake db:migrate

$ rake db:migrate VERSION=5

$ rake db:migrate RAILS_ENV=test
```

Jeszcze inne migracje

bin/rails generate migration RemoveProwadzacyFromZajecia prowadzacy:string

I jak to sprawdzić

Wykład 6.

I jak to sprawdzić

Wykład 6.

```
bin/rails console
```

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 new demo --database=sqlite3
```

Plan wykładu

Rusztowanie (scaffold)

Co to jest rusztowanie

wygenerowany automatycznie szkielet służący do operowania na modelu.

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

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

Przykład

```
$ bin/rails generate scaffold Wykladowca imie:string
nazwisko:string
```

Wynik działania skryptu

Przykład

```
$ bin/rails generate scaffold Wykladowca imie:string nazwisko
```

Wynik działania skryptu

Przykład

$ bin/rails generate scaffold Wykladowca imie:string nazwisko

Migracja

Powstaje plik migracji, konieczne jest

```
$ rake db:migrate
```

Wynik działania skryptu

Przykład

$ bin/rails generate scaffold Wykladowca imie:string nazwisko

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

$ bin/rails generate scaffold Wykladowca imie:string nazwisko

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

Model – przypomnienie

- Modele to podklasy klasy ActiveRecord::Base; w nowszych Railsach dodano pośrednią klasę ApplicationRecord;
- 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, :punkty
- validates_numericality_of :punkty,
  :message => 'Pole punkty nie jest liczbą', :if => test()
- validates_uniqueness_of :nazwa

Zastosowanie

app/models/wyklady.rb:

```
class Wyklad < ApplicationRecord

  validates_uniqueness_of :nazwa

end
```

Walidacja własna

Metoda w klasie modelu:

```
class Wyklad < ApplicationRecord
    protected
    def validate
        errors.add(:punkty, 'punkty powinne być dodatniw')
                if punkty.nil? || punkty < 0.01
    end
end
```