Ruby on Rails Webinar

- Michał Makaruk

Czas trwania

- Webinar będzie trwał od 1.5 h do 2 h.

Co dzisiaj???

- Mniej magii. Więcej zrozumienia.
- Modele, na końcu modifikacja wygenerowanych scaffoldów.

Bez Railsów i Modelu

- Tworzymy klasę , która jest odpowiedzialna za łączenie się z bazą danych.
- Musimy w naszym kodzie ręcznie pisać zapytania SQL.

Model

Modele w Rails reprezentowane są przez klasy dziedziczące po klasie bazowej ActiveRecord::Base.

Obiekty kodu modelu reprezentują rzeczy istniejące w domenie systemu – tak jak bilety w systemie biletów.

Model odpowiedzialny jest za logikę. Przykładowy model to User.

Przykład tworzenia modelu

```
MacBook-Pro-Micha:rails4app michalos$ rails g model User name:string email:strin
g description:text
      invoke  active_record
      create    db/migrate/20130919174310_create_users.rb
      create    app/models/user.rb
      invoke    test_unit
      create      test/models/user_test.rb
      create      test/fixtures/users.yml
```

Rails g model

- Rails g model Person name:string surname:string
- Rails g model Event name:string created_at:datetime
- Rails g model Attendee user_id:integer name:string
- Rails g model Form name:string description:text
- Rails g model Book name:string author:string

Rails g model nazwaModelu

-tworzy migracje bazy danych;

-tworzy klasę dziedziczącą po Activerecord::Base;

-tworzy testy;

-tworzy fixtures.

Przykładowe typy

-string (łańcuch znaków);

-text (dłuższy łańcuch znaków);

-decimal (liczba dziesiętna);

-date (data);

-boolean (wartość logiczna prawda bądź nieprawda).

Migracja

- migracja to „po prostu” skrypt Ruby;

- migracja odpowiedzialna jest za aktualizacje schematu bazy danych;

- dzięki migracji w łatwy sposób możemy przeglądać strukturę zmian bazy danych

- Więcej informacji na temat migracji:

http://guides.rubyonrails.org/migrations.html

Migracja

- Przykładowa migracja

```
class CreateUsers < ActiveRecord::Migration
  def change
    create_table :users do |t|
      t.string :name
      t.string :email
      t.text :description

      t.timestamps
    end
  end
end
```

Wykonanie migracji

```
MacBook-Pro-Micha:rails4app michalos$ rake db:migrate
==  CreateUsers: migrating ====================================================
-- create_table(:users)
   -> 0.0290s
==  CreateUsers: migrated (0.0291s) ===========================================
```

Podstawowe polecenia migracje

- polecenie rake db:migrate wywołuje wszystkie migracje, które nie zostały wywołane wcześniej

- polecenie rake db:rollback odwołuje ostatnią migracje

- polecenie rake db:roolback STEP=5 odwołuje 5 ostatnich migracji

Migracja dodająca atrybut do modelu

```
MacBook-Pro-Micha:cwiczenia_rails michalos$ rails g migration addOcupationToClie
nt occupation:string
      invoke  active_record
      create    db/migrate/20131007173028_add_ocupation_to_client.rb
```

Create – tworzenie obiektu

```
>> p = Person.new(:name => "John Doe")
=> #<Person id: nil, name: "John Doe", created_at: nil,
:updated_at: nil>
>> p.new_record?
=> true
>> p.save
=> true
>> p.new_record?
=> false
```

Create – tworzenie obiektu

```
>> p = Person.new(:name => "John Doe")
=> #<Person id: nil, name: "John Doe", created_at: nil,
:updated_at: nil>
>> p.new_record?
=> true
>> p.save
=> true
>> p.new_record?
=> false
```

Create – tworzenie obiektu 2

```
>> p = Person.new
>> p.name = "Adam"
>> p.save
=> true
```

Save

-metoda save zachowuje dane w bazie danych, kiedy dane są poprawne.

Read

User.last – zwraca ostatni element

User.first – zwraca pierwszy element

User.count – zwraca ilość elementów

User.order(:name) – zwraca uporządkowanego User po nazwie

User.limit(5) – zwraca tylko 5 Userów.

User.where(name: ''Michal'') – zwraca Wszystkich Userów o imieniu Michal

Read

Inne zapytania:

http://guides.rubyonrails.org/active_record_querying.html

Polecenie create

```
Zamiast pisać:
user = User.new
user.name = "Michał"
user.occupation = "programmer"
user.save

Możemy po prostu napisać:

user = User.create(name: "Michał",
occupation: "programmer")
```

Delete

```
irb(main):001:0> o = Organization.last
  Organization Load (0.3ms)  SELECT "organizations".* FROM "organizations" ORDER
 BY "organizations"."id" DESC LIMIT 1
=> #<Organization id: 3, name: "Webventure", number_of_employees: nil, address:
nil, created_at: "2013-09-19 16:33:55", updated_at: "2013-09-20 04:36:27">
irb(main):002:0> o.destroy
```

Scaffold

- Przy tworzeniu scaffoldu automatycznie generowany jest za nas model.
- Dodatkowo także widoki i kontrolery.

Seeds -ziarna

-Jest to plik , w którym tworzymy przykładowe dane bądź dane , które potrzebne są nam w środowisku produkcyjnym (na samym początku).
-plik znajduje się w db/seeds.rb
-plik wywołujemy za pomocą polecenia rake db:seed
-kilkukrotne wywołanie powoduje od nowa uruchomienie pliku.

Seeds przykładowe

Przykład pliku:

```
10.times do
  User.create(name: "Adam",occupation:
"programmer")
End

puts ''Users created''
```

Polecenie destroy all

Polecenie Model.destroy_all niszczy wszystkie obiekty w bazie danych.

Konsola

- Za pomocą konsoli możemy korzystać z języka Ruby plus mamy połączenie do bazy danych.
- Aby wywołać konsole wpisujemy rails c w katalogu projektu.

Tryb sandbox Konsola

-tryb sandbox używany jest wtedy kiedy nie chcemy robić zmian w bazie danych

```
MacBook-Pro-Micha:cwiczenia_rails michalos$ rails c --sandbox
Loading development environment in sandbox (Rails 4.0.0)
Any modifications you make will be rolled back on exit
irb(main):001:0>
```