Kurs programowania Ruby on Rails

Routing i widoki

Z czego się uczyć??

Szkolenie Rails

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

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

http://guides.rubyonrails.org/
active_record_querying.html#passing-in-arguments

https://www.codeschool.com/courses/rails-for-zombies-2

GET show

Szkolenie Rails

get '/patients/:id', to: 'patients#show', as: 'patient'

Ścieżka może być typu: /patients/1 i przechodzi do akcji show w kontrolerze patients , natomiast dzięki as możemy odwoływać się w widoku poprzez:

patient_path(@patient.id) bądź krócej:

patient_path(@patient)

Przykład link_to

Szkolenie Rails

```
<%= link_to 'Patient Record', patient_path(@patient) %>
```

Radzenie sobie z obiektami

Szkolenie Rails

- w formularzach html możem korzystać z obiektów

- korzystanie odbywa się przez odwołanie nawiasy kwadratowe przykładowo:

article[title]

Przykład formularza z obiektem

Szkolenie Rails

```
<form accept-charset="UTF-8" action="/articles/create"
method="post" class="nifty_form">
  <input id="article_title" name="article[title]" type="text" /
>
  <textarea id="article_body" name="article[body]"
cols="60" rows="12"></textarea>
  <input name="commit" type="submit" value="Create" />
</form>
```

Przykład formularza z obiektem

Szkolenie Rails

```
<form accept-charset="UTF-8" action="/articles/create"
method="post" class="nifty_form">
  <input id="article_title" name="article[title]" type="text" /
>
  <textarea id="article_body" name="article[body]"
cols="60" rows="12"></textarea>
  <input name="commit" type="submit" value="Create" />
</form>
```

Przykład formularza z obiektem Rails

Szkolenie Rails

```
<%= form_for @article, url: {action: "create"}, html: {class: "nifty_form"} do |f| %>
  <%= f.text_field :title %>
  <%= f.text_area :body, size: "60x12" %>
  <%= f.submit "Create" %>
<% end %>
```

Różne składnie

Szkolenie Rails

```
## Tworzenie nowego artykułu:
# Długa składnia:
form_for(@article, url: articles_path)
# Krótsza składnia:
form_for(@article)

## Edytowanie artykułu
# Długa składnia:
form_for(@article, url: article_path(@article), html:
{method: "patch"})
# Krótka składnia:
form_for(@article)
```

Representational State Transfer

Szkolenie Rails

Representational State Transfer – wzorzec architektury oprogramowania wywiedziony z doświadczeń przy pisaniu specyfikacji protokołu HTTP. REST jest wzorcem architektury oprogramowania wprowadzającym dobre praktyki tworzenia architektury aplikacji rozproszonych.

REST wprowadza terminy takie jak jednorodny interfejs, bezstanowa komunikacja, zasób, reprezentacja.

Zaproponowany przez Roya T. Fieldinga w 2000 roku[1].

.

Przykład REST

Szkolenie Rails

Wywołanie klasyczne

http://example.com/article?id=1234&format=print
Wywołanie RESTful

http://example.com/article/1234/print

Problem

Szkolenie Rails

-Mamy już za pomocą mechanizmu ORM odwzorowane tabelkę na nasze obiekty

-Chcielibyśmy mieć takie same odzworowanie po stronie adresów URL

W Railsach :
Resources pomaga nam odwzorować adresy URL na nasze modele

Resources routing

Szkolenie Rails

resources :photos

- tworzy nam bardzo dużo ścieżek

- Korzysta z pojęcia REST jakim jest zasób

- dla modelu photo i kontrolera photos mamy odwzorowanie w routes

- zasobem jest w tym przypadku photo

Resources photos tabelka

Szkolenie Rails

HTTP Verb	Path	Action	Used for
GET	/photos	index	display a list of all photos
GET	/photos/new	new	return an HTML form for creating a new photo
POST	/photos	create	create a new photo
GET	/photos/:id	show	display a specific photo
GET	/photos/:id/edit	edit	return an HTML form for editing a photo
PATCH/PUT	/photos/:id	update	update a specific photo
DELETE	/photos/:id	destroy	delete a specific photo

Tworzone helpery-resources

photos_path zwraca /photos
new_photo_path zwraca /photos/new
edit_photo_path(:id) zwraca /photos/:id/edit
edit_photo_path(10) zwraca /photos/10/edit
photo_path(:id) zwraca /photos/:id
photo_path(10) zwraca /photos/10)

RESOURCES Tylko wybrana akcja


```
resources :photos, :only =>[:index]
```

Tworzy routes tylko z akcją index, pomijając inne akcje.

Resources member

Szkolenie Rails

```
resources :photos do
  member do
    get 'preview'
  end
End
```

Tworzy nam ścieżkę /photos/1/preview i także odwołujemy się do akcji preview kontrolera photos

Do 1 odwołujemy się poprzez params[:id]

Resources collection

Szkolenie Rails

```
resources :photos do
  collection do
    get 'search'
  end
end
```

Tworzy nam ścieżkę /photos/search/ i także odwołujemy się do akcji search kontrolera photos. Możemy sobie wyobrazić ,że search jest „metodą statyczną”

Select formularz

Szkolenie Rails

```
<select name="city_id" id="city_id">
  <option value="1">Lisbon</option>
  <option value="2">Madrid</option>
  ...
  <option value="12">Berlin</option>
</select>
```

Select Railsy

Szkolenie Rails

```
<%= select_tag(:city_id, '<option value="1">Lisbon</
option>...') %>

<%= options_for_select([['Lisbon', 1], ['Madrid', 2], ...]) %>
```

wynik:

```
<option value="1">Lisbon</option>
<option value="2">Madrid</option>
```

Przykład Mapowania

Szkolenie Rails

```
<% cities_array = City.all.map { |city| [city.name, city.id] }
%>
<%= options_for_select(cities_array) %>
```

Bądź krócej:

```
<%=
options_from_collection_for_select(City.all, :id, :name) %>
```

Nested attributes

Szkolenie Rails

- używamy kiedy w jednym formularzu potrzebujemy mieć 2 powiązane modele
- accepts_nested_attributes_for :nazwa_powiazanego modelu

http://api.rubyonrails.org/classes/ActiveRecord/NestedAttributes/ClassMethods.html

Nested attributes przykład

Szkolenie Rails

```
class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts
End
```

- dzięki tej linijce mamy dodatkową metodę:

posts_attributes= (attribiutes)

Nested attributes params

Szkolenie Rails

```
params = { member: {
  name: 'joe', posts_attributes: [
    { title: 'Kari, the awesome Ruby documentation
browser!' },
    { title: 'The egalitarian assumption of the modern
citizen' },
    { title: '', _destroy: '1' } # this will be ignored
  ]
}}
```

Aby skorzystać w formularzu z nested attributes

- musimy dopisać linijkę:
accepts_nested_attributes_for

Dependent destroy

Szkolenie Rails

```
class Question < ActiveRecord::Base
  has_many :answers, :dependent => :destroy
End
```

Jeśli zostanie usunięte pytanie , to następnie zostaną usunięte odpowiedzi do tego pytania.

Zapytania

Szkolenie Rails

Kiedy w kontrolerze w kilku akcjach powtarzamy jakieś zapytania nurtuje nas problem , w jaki sposób zrobić to ,aby zastosować zasadę **DRY, która mówi nam aby nie powtarzać kodu kilku krotnie.**

Rozwiązanie:

Zapisujemy metodę statyczną, która działa na modelu.

Przykładowe rozwiązanie

Szkolenie Rails

```
class Post < ActiveRecord::Base
  def self.created_before(time)
    where("created_at < ?", time)
  end
end
```

Scope

Szkolenie Rails

- robi dokładnie to samo tylko używa krótszej składni.

Analogiczne rozwiązanie za pomocą scope było następujące:

```
class Post < ActiveRecord::Base
  scope :created_before, ->(time) { where("created_at < ?", time) }
end
```

Scope

Szkolenie Rails

- robi dokładnie to samo tylko używa krótszej składni.

Analogiczne rozwiązanie za pomocą scope było następujące:

```
class Post < ActiveRecord::Base
  scope :created_before, ->(time) { where("created_at < ?", time) }
end
```

Domyślny scope

Szkolenie Rails

```
class User < ActiveRecord::Base
  default_scope  { where state: 'pending' }
  scope :active, -> { where state: 'active' }
  scope :inactive, -> { where state: 'inactive' }
end
 User.all
# => SELECT "users".* FROM "users" WHERE
"users"."state" = 'pending'
 User.active
# => SELECT "users".* FROM "users" WHERE
"users"."state" = 'active'
```

Łączenie scope

Szkolenie Rails

```
class User < ActiveRecord::Base
  scope :active, -> { where state: 'active' }
  scope :inactive, -> { where state: 'inactive' }
end
```

Kurs programowania Ruby on Rails

Routing i widoki

Koniec