

EPI: Interfejs Graficzny 2010/2011

Podstawy Rails – powtórzenie

Aleksander Pohl

19 stycznia 2011

Zadanie

Zaimplementować funkcjonalność książki gości, bez użycia rusztowania.

1. Model Entry

1. rails g model entry name:string comment:text
2. rake db:migrate
3. Edytujemy app/models/entry.rb:

```
class Entry < ActiveRecord::Base
  validates_presence_of :name, :comment
end
```

2. Kontroler

1. rails g controller entries index new create
2. otwieramy app/controllers/entries_controller.rb
3. definiujemy 3 metody:
 - ▶ index – która wyświetla wszystkich gości
 - ▶ new – która wyświetla formularz do tworzenia wpisów
 - ▶ create – która zachowuje wpis w bazie danych

2. EntriesController

```
# encoding: utf-8
class EntriesController < ApplicationController
  def index
    @entries = Entry.find(:all, :order => "created_at DESC")
  end

  def new
    @entry = Entry.new
  end

  def create
    @entry = Entry.new(params[:entry])
    if @entry.save
      flash[:notice] = "Dziękujemy za wpis w książce gości!"
      redirect_to :action => "index"
    else
      render :action => "new"
    end
  end
end
```



3. Widok – index

Otwieramy plik: `app/views/entries/index.html.erb`

```
<h1>Książka gości biblioteki</h1>
```

```
<p>Ostatnie wpisy:</p>
```

```
<% for entry in @entries -%>
```

```
  <div>
```

```
    <div><%= entry.name %> napisał(a) <%= entry.created_at %>:</div>
```

```
    <div><%= entry.comment %></div>
```

```
  </div>
```

```
<% end -%>
```

```
<br />
```

```
<%= link_to "Nowy wpis", :action => "new" %>
```

4. Widok – new

Otwieramy plik: app/views/entries/new.html.erb

```
<h1>Nowy wpis w księdze gości</h1>
<% form_for :entry, :url => {:action => "create"} do |f| %>
  <% if @entry.errors.any? %>
    <ul>
      <% @entry.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
    </ul>
  <% end %>
  <p>
    <%= f.label :name, "Twoje imię:" %><br/>
    <%= f.text_field :name %>
  </p>
  <p>
    <%= f.label :comment, "Twój komentarz:" %> <br/>
    <%= f.text_area :comment %>
  </p>
  <%= f.submit "Dodaj" %>
<% end %>
```



5. Styl dla błędów

Otwieramy plik: `public/stylesheets.css`

```
.fieldWithErrors {  
  display: inline;  
}  
  
.fieldWithErrors input {  
  border: 1px solid red;  
}
```

Zadania:

- ▶ utwórz layout z linkami do nowego wpisu oraz listy wpisów
- ▶ poproś użytkownika o dostarczenie linku do jego avatara
- ▶ używając migracji dodaj kolumnę w bazie danych do przechowywania URL-a do avatara
- ▶ wyświetl avatar używając helpera `image_tag`
- ▶ dodaj walidacje do modelu:
 - ▶ długości imienia i komentarza
 - ▶ format URL-a avatara
- ▶ w layoucie dodaj link do głównego arkusza stylów, aby błędne pola były wyświetlane na czerwono
- ▶ zmodyfikuj akursz stylów aby ulepszyć sposób wyświetlania wpisów
- ▶ użyj helpera `simple_format` (`ActionView::Helpers::TextHelper`), aby poprawnie wyświetlić wielo-linijkowe komentarze

Zadania:

- ▶ użyj helpera `time_ago_in_words` (`ActionView::Helpers::DateHelper`) aby ulepszyć wyświetlanie czasu, który upłynął od dodania wpisu
- ▶ użyj helpera `truncate` (`ActionView::Helpers::TextHelper`), aby przyciąć komentarze oraz dodaj link „więcej”, który pozwala zobaczyć cały wpis
- ▶ użyj helpera `image_tag` (`ActionView::Helpers::AssetTagHelper`) i połącz go z metodą `link_to`, tak aby dostarczyć graficznej reprezentacji dla linku „Dodaj”
- ▶ zmodyfikuj url aplikacji (w konfiguracji trasowania), tak aby domyślnie wyświetlana była książka gości