Web-palvelinohjelmointi Ruby on Rails, kurssikoe 7.3.2017

Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

Kirjoita jokaiseen palauttamaasi paperiin nimesi ja opiskelijanumerosi sekä kurssin nimi.

Selitä jokaisessa tehtävässä asiat kooditasolla *sekä* abstraktimmalla tasolla, eli selitä myös mikä on kunkin tehtävän kannalta relevantin sovelluksen komponentin rooli toiminnallisuuden kannalta.

Kokeessa on jaossa 15 pistettä. Koeaika on 150 minuuttia. Käytettävissäsi on siis keskimäärin 10 minuttia per piste, eli älä juutu liian kauaksi aikaa mihinkään yksittäiseen tehtävään!

tehtävä 1 (2p)

HUOM: Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

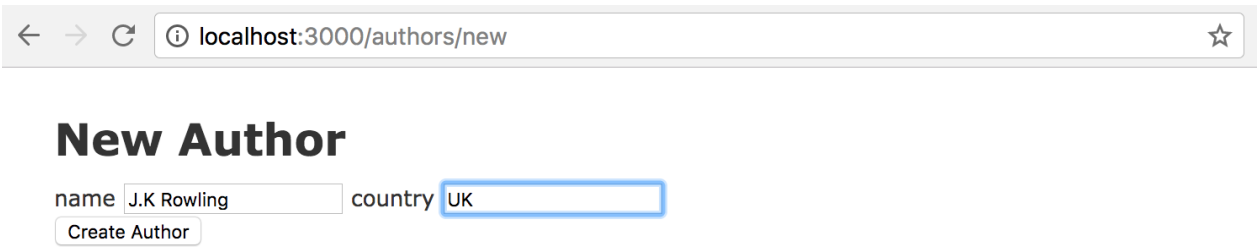
Tapahtuu HTTP GET -pyyntö osoitteeseen *http://localhost:3000/authors/1*

Selitä tarkasti mitä tapahtuu, kun pyyntö saavuttaa sovelluksen (voit olettaa että pyyntö onnistuu).

tehtävä 2 (4p)

HUOM: Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

Käyttäjä on navigoinut uuden kirjailijan luomissivulle:



Kerro mitä kaikkea sovelluksessa tapahtuu lomakkeen (views/authors/new.html.erb) lähettämisen seurauksena.

Kerro myös tarkasti mitä tapahtuu, jos kirjailijan luominen epäonnistuu, ja mikä voi aiheuttaa epäonnistumisen (voit jättää huomioimatta tietoliikenneprotokollan tai palvelimen tasolla tapahtuvat ongelmat).

tehtävä 3 (3p)

HUOM: Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

Kaikkien kirjailijoiden sivun *http://localhost:3000/authors* näyttää seuraavalta:

Sivun toteutus ei ole tällä hetkellä suorituskyvyltään erityisen hyvä. Jos kirjailijoita ja kirjoja olisi kirjoja tuhansia, olisi sivu todennäköisesti erittäin hidas.

Kerro mitä ongelmia sivun toteutuksessa on suorituskyvyn kannalta ja millä Railsin tarjoamilla tekniikoilla näitä voidaan optimoida?

Mitkä ovat kunkin optimointitekniikan hyvät ja huonot puolet? Mitä muutoksia näiden käyttöönotto edellyttäisi koodin tasolla?

tehtävä 4 (3p)

HUOM: Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

Selitä miten sovellusta laajennettaisiin siten, että mukaan tuotaisiin uusi käsite *genre* (Genre). Yksittäinen kirja voi liittyä useisiin genreihin (esim. *scifi* ja *kauhu*) ja vastaavasti genreen voi liittyä useita kirjoja.

Vastauksessa ei ole tarvetta kertoa näyttöjen ja kontrollerien osalta tarvittavia muutoksia. Kiinnostuksen kohteena ovat nyt erityisesti laajennuksen aiheuttamat muutokset modeleihin ja tietokantatasolle sekä se, miten nämä toteutetaan.

tehtävä 5 (3p)

HUOM: Kaikki tehtävät liittyvät koepaperin lopusta löytyvään koodiin.

- Toteuta luokan `Author` metodi `first_publication_at` siten, että metodi palauttaa kirjailijan (Author) kirjoista ensimmäisenä julkaistun julkaisuvuoden. Voit olettaa että kirjailia on julkaissut ainakin yhden kirjan.
- Tee testi, joka testaa edellisessä tehtävässä tekemääsi metodia.

Kaikkin kokeen tehtäviin liittyvä ohjelmakoodi (voit olettaa että kirjoihin liittyvä koodi on toteutettu järkevästi):

```
# config/routes.rb

Rails.application.routes.draw do
  get '/authors', to:'authors#index'
  get '/authors/new', to:'authors#form_for_new'
  post '/authors', to:'authors#create'
  get '/authors/:id', to:'authors#show'

  resources :books
end


# app/controllers/authors_controller.rb

class AuthorsController < ApplicationController
  def index
    @authors = Author.all
  end

  def show
    @author = Author.find(params[:id])
  end

  def form_for_new
    @author = Author.new
    render :new
  end

  def create
    @author = Author.new(params.require(:author).permit(:name, :country))

    if @author.save
      redirect_to @author, notice: 'Author was successfully created.'
    else
      render :new
    end
  end
```

```
end

# app/models/author.rb

class Author < ActiveRecord::Base
  has_many :books

  validates :name, uniqueness: true,
                   length: { minimum: 3, maximum: 30 }
  validates :country, presence: true

  def total_pages
    books.inject(0) { |sum, p| sum + p.pages }
  end

  def first_publication_at
    # to be implemented in question 5
  end
end


# db/migrate/20160215142931_create_books_and_authors.rb

class CreateAuthors < ActiveRecord::Migration
  def change
    create_table :authors do |t|
      t.string :name
      t.string :country
    end

    create_table :books do |t|
      t.string :title
      t.integer :pages
      t.integer :published
      t.integer :author_id
    end
  end
end



# app/views/auhtors/index.html.erb

<h1>Authors</h1>

<table>
  <thead>
    <tr><th>Name</th><th>Country</th><th>Total Pages</th><th>First publication</th></tr>
  </thead>

  <tbody>
    <% @authors.each do |author| %>
      <tr>
        <td><%= link_to author.name, author %></td>
        <td><%= author.country %></td>
        <td><%= author.total_pages %></td>
        <td><%= author.first_publication_at %></td>
      </tr>
    <% end %>
  </tbody>
</table>


# app/views/authors/new.html.erb

<h1>New Author</h1>

<%= form_for(@author) do |f| %>
  <% if @author.errors.any? %>
      <% @author.errors.full_messages.each do |msg| %>
        <p><%= msg %></p>
      <% end %>
  <% end %>

  name <%= f.text_field :name %>
  country <%= f.text_field :country %>

  <div class="actions">
```

```
    <%= f.submit %>
  </div>
<% end %>


# app/views/authors/show.html.erb

<h2><%= @author.name %></h2>

<p><%= @author.country %></p>

<h2>books</h2>
<ul>
  <% @author.books.each do |book| %>
    <li> <%= link_to book.title, book %> </li>
  <% end %>
</ul>

# spec/models/authors_spec.rb

require 'rails_helper'

RSpec.describe Author, type: :model do
  it "is saved with valid data" do
    author = Author.create name: "Robert Martin", country: "USA"

    expect(author).to be_valid
    expect(author.name).to eq("Robert Martin")
  end
end
```