Luca Scherer
INF17H

Quicknote AB151-04

Flash in Rails

Flash-Meldungen in Rails sind eine einfache Methode, um eine einseitige Kommunikation vom Server zum Client zu realisieren. Meistens werden Flash-Meldungen verwendet, um den Benutzer darüber zu informieren, ob eine CRUD-Aktion, die er versucht hat durchzuführen, erfolgreich war oder nicht. In Unserem Fall wird ein Feedback der Registration oder des Anmeldevorgangs zurückgegeben. Die vier Typen:

- Success
- Notice
- Alert
- Error

Fancy Flash in Rails mit toastr

In diesem Schritt haben wir das gem 'toastr' installiert. Es ermöglicht es uns Flash-Meldungen in einem schönen Pop-Up darzustellen. Flash-Meldungen kann man standardmässig wie folgt ausgeben:

```
<p class="notice"><%= notice %></p>
<p class="alert"><%= alert %></p>
```

Ein Toast wird wie folgt erstellt:

```
<script type="text/javascript">
      toastr.success("Wow, a success")
 </script>
```

Um für jede Flash-Meldung ein Toast zu generieren, iteriert man durch alle Flashes:

```
<script type="text/javascript">
      <% flash.each do |key, value| %
            toastr.<%= type %>('<%= value %>')
      <% end %>
</script>
```

Aufgabe: Error Rails flash bei falscher Benutzerregistration

Um Error-Flashes bei einer falscher Benutzerregistration auszugeben habe ich eine Partial-View mit folgendem Code geschrieben:

```
<% if resource.errors.any? %>
  <script type="text/javascript">
      <% resource.errors.full_messages.each do |value| %>
      toastr.error('<%= value %>')
      <% end %>
  </script>
<% end %>
```

Die wird dann in der new-View gerendert.

User Profile Page

Für unsere User Profile Page habe ich das User Model mit einer Migration erweitert. Die Migration sah wie folgt aus:

```
class AddFieldsToUser < ActiveRecord::Migration[5.2]
  def change
    add_column table_name :users, column_name :website, type :string
    add_column table_name :users, column_name :bio, type :text
    add_column table_name :users, column_name :provider, type :string
    add_column table_name :users, column_name :uid, type :string
    add_column table_name :users, column_name :image, type :string
  end
end
```

Im Anschluss ergänzte ich folgende Zeile in routes.rb:

```
resources :users, only: [:index, :show]
```

Diese grenzt die Views auf index und show ein.

Danach habe ich einen manuell den users_controller.rb erstellt. In welchem ich folgende Methode implementiert habe:

```
def show
  @user = User.find(params[:id])
end
```

Diese gibt einen User abhängig der ID in der URL zurück.

Gravatar für User Photo in show-View

In diesem Schritt habe ich zuerst eine Methode im ApplicationHelper geschrieben, welche die URL des Bildes eines Users von Gravatar zurückgibt. Die Methode verschlüsselt die Email des Users nach dem MD5-Verfahren und hängt diese an «https://www.gravatar.com/avatar/»

Gravatar ist ein Service der Emailadressen mit Profilbildern assoziiert. Somit kann jede Emailadresse ein Profilbild hochladen, auf welches verschiedene Webseiten zugreifen.

Danach habe ich die Profil-Seite gebaut. Dies war sehr aufwändig und dauerte demensprechend lange. Die Seite zeigt alle Eigenschaften eines Users und dessen Profilbild. Falls man die Profilseite von sich selbst aufruft, wird ebenfalls ein Setting-icon angezeigt, mitwelchem man sein Profil bearbeiten kann.

link_to Profile Page

In diesem Schritt habe ich lediglich den Profil-Button oben rechts angepasst. Nun linkt er direkt auf die eigene Profil-Seite.

Edit User Page

Zu guter Letzt habe ich die Edit-Page erstellt. Dazu konnte ich eine fix-fertige html-erb-Seite verwenden. Wenn man nun einige Werte änderte, wurden die nach dem speichern noch nicht übernommen. Das liegt daran, weil devise dass Passwort für jede Änderung benötigt. Um dies zu umgehen, habe ich folgenden RegistrationsController ergänzt:

```ruby
class RegistrationsController < Devise::RegistrationsController

  protected

  def update_resource(resource, params)
    resource.update_without_password(params)
  end
end
```

Dieser erbt vom Devise-RegistrationController. Danach funktionierte auch das updaten von Usern.

Anwendungszweck

Flash ist sicher sehr nützlich und finden sehr viele Anwendungszwecke. Gerade für User-Feedbacks, welche vom Server kommen ist es praktisch. Toastr ist ebenfalls ein praktisches Gem, da man dem User ein schönes Feedback in Form eines Toasts anzeigen möchte. Dies findet ebenfalls sehr viele Anwendungszwecke. Gravatar ist bereits ziemlich gut etabliert. Somit macht es sicher Sinn Bilder von Gravatar zu beziehen, falls der User ein solches hochgeladen hat.

Vor- und Nachteile

PRO	KONTRA
Mit Hilfe der hier vorgestellten Technologien, kann man dem User ein schönes und übersichtliches Feedback bieten.	Falls der User kein Bild auf Gravatar hinterlegt hat, wird auch auf der Webseite kein Bild angezeigt.
Gravatar ersetzt ein Hochladen eines Profilbildes auf jeder einzelnen Webseite. Man muss es nur auf Gravatar hinterlegen.	Falls die Webseite nicht auf Gravatar-Bilder zugreift, hilft auch Gravatar nicht weiter und man muss auf einzelnen Seiten das selbe Bild hochladen.

Selbstreflexion

1. Was habe ich gelernt?

- Ich habe gelernt, wie man in Rails mit Flash umgeht und durch dessen Meldungen iteriert. Ausserdem lernte ich wie man Meldungen an den Benutzer schön in einem Toast verpackt. Dazu weiss ich nun wie man auf die ID eines Gravatars kommt und dessen Bild in eine Webseite einbaut.

2. Was hat mich behindert?

- Behindert wurde ich durch nichts. Ich konnte Schritt für Schritt befolgen und kam gut vorwärts. Allerdings machte ich einige Flüchtigkeitsfehler beim Abschreiben des Codes für die Profile-View. Diese musste ich erst suchen und dann beheben.

3. Was habe ich nicht verstanden?

- Alle vorgestellten Technologien sind mir bekannt und ich verstehe deren Anwendungszweck, und Benutzung.

4. Was kann ich beim Studium besser machen?

- -

Fazit

Das Arbeitsblatt hat mir sehr gut gefallen. Ich lernte sehr viel Hilfreiches kennen und kam gut vorwärts. Einzig bei den Error-Toasts bei falscher Benutzerregistration wusste ich erst nicht, was ich genau tun

muss. Nach einer kurzen Absprache mit meinen Klassenkameraden musste ich aber Bescheid und konnte dies umsetzen.

Show.html.erb

```
<div class="row justify-content-md-center profile-wrapper">
  <div class="col-md-4 text-center">
    <%= image_tag avatar_url(@user), width: '152', height: '152', class: "round-img"%>
  </div>
  <div class="col-md-8">
    <div class="row">
      <p class="username"><%=@user.name %></p>
      <% if @user == current_user %>
        <%= link_to "Edit Profile", edit_user_registration_path, class: "btn btn-outline-dark common-btn edit-profile-btn" %>
        <button type="button" class="core-sprite setting" data-toggle="modal" data-target="#exampleModal"></button>
      <% end %>

      <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="exampleModalLabel">Settings</h5>
              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">×</span>
              </button>
            </div>
            <div class="list-group text-center">
              <a href="#" class="list-group-item list-gourp-item-action">Change Password</a>
              <%= link_to "Log out", destroy_user_session_path, method: :delete,
                class: "list-group-item list-group-item-action" %>
              <a href="#" class="list-group-item list-group-item-action" data-dismiss="modal">Cancel</a>
            </div>
          </div>
        </div>
      </div>
      </div>
    <div class="row">
      <p class="email"><%= @user.email %></p>
    </div>
  </div>
</div>
```

Application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>Instagram</title>
  <%= csrf_meta_tags %>
  <%= csp_meta_tag %>

  <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track': 'reload' %>
  <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
</head>

<body>
<%= if current_user
      render 'shared/navbar'
    end %>
<div class="container">
  <%= yield %>
</div>

<%= render 'shared/footer' %>
<% if flash.any? %>
  <script type="text/javascript">
    <%flash.each do |key, value| %>
      <% type = key.to_s.gsub('alert', 'error').gsub('notice', 'success') %>
      toastr.<%= type %>('<%=value%>');
      <% end %>
  </script>
  <% end %>
</body>

</html>
```

Application.scss

```scss
.profile-wrapper{
  padding: 60px 0;
  border-bottom: 1px solid #efefef;

  .username {
    font-size: 32px;
    line-height: 40px;
    font-weight: 200;
  }
  .email{
    font-size: 16px;
    font-weight: bold;
  }
}

.round-img{
  border-radius: 50%;
}
.common-btn{
  height: 28px;
  line-height: 24px;
  padding: 0 24px;
  border-color: #dbdbdb;
  font-size: 14px;
}

.edit-profile-btn {
  margin: 5px 0 0 15px;
  font-weight: bold;
}

.setting{
  margin: 7px 0 0 10px;
  background-color: transparent;
  background-position: -125px -355px;
  height: 24px;
  width: 24px;
  border: none;
}
.setting:hover{
  cursor: pointer;
}
```

```scss
.edit-profile-wrapper {
  margin: 60px 0 0;
  .left-col {
    border: 1px solid #dbdbdb;
    padding: 0;
    background-color: white;

    .list-group-item {
      border: none;
      color: #262626;
      border-radius: 0;
      font-size: 16px;
      border-left: 2px solid transparent;
    }

    .list-group-item:first-child {
      font-weight: bold;
      border-left-color: #262626;
    }
    .list-group-item:not(:first-child):hover {
      border-left-color: #dbdbdb;
    }
  }
  .right-col {
    border: 1px solid #dbdbdb;
    border-left: none;
    background-color: white;
    .username-img {
      margin: 32px 0;
      .col-sm-9 {
        font-size: 20px;
        font-weight: 400;
        line-height: 20px;
      }
    }
    .form-group {
      font-size: 16px;
    }
  }
}
```

Applicationhelper.rb

```ruby
module ApplicationHelper
  def avatar_url user
    gravatar_id = Digest::MD5::hexdigest(user.email).downcase
    "https://www.gravatar.com/avatar/#{gravatar_id}.jpg"
  end
end
```

Registration_controller.rb

```ruby
class RegistrationsController < Devise::RegistrationsController

  protected

  def update_resource(resource, params)
    resource.update_without_password(params)
  end
end
```

Users_controller.rb

```ruby
class UsersController < ApplicationController
  def show
    @user = User.find(params[:id])
  end
end
```

Edit.html.erb

```erb
<div class="row edit-profile-wrapper">
  <div class="col-sm-3 left-col">
    <div class="list-group">
      <a href="#" class="list-group-item list-group-item-action">Edit Profile</a>
      <%= link_to "Log out", destroy_user_session_path, method: :delete,
                  class: "list-group-item list-group-item-action" %>
    </div>
  </div>
  <div class="col-sm-9 right-col">
    <%= form_for(resource, as: resource_name, url: registration_path(resource_name), html: { method: :put }) do |f| %>
      <%= render 'shared/devisemes' %>

      <div class="row username-img">
        <div class="col-sm-3 text-right">
          <%= image_tag avatar_url(@user), width: '38', height: '38', class: "round-img" %>
        </div>
        <div class="col-sm-9">
          <p><%= @user.name %></p>
        </div>
      </div>

      <div class="form-group row">
        <%= f.label :name, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
        <div class="col-sm-9">
          <%= f.text_field :name, autofocus: true, class: "form-control col-sm-9" %>
        </div>
      </div>

      <div class="form-group row">
        <%= f.label :website, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
        <div class="col-sm-9">
          <%= f.text_field :website, class: "form-control col-sm-9" %>
        </div>
      </div>

      <div class="form-group row">
        <%= f.label :bio, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
        <div class="col-sm-9">
          <%= f.text_field :bio, class: "form-control col-sm-9" %>
        </div>
      </div>

      <div class="form-group row">
        <%= f.label :email, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
        <div class="col-sm-9">
          <%= f.email_field :email, class: "form-control col-sm-9" %>
        </div>
      </div>
```

```
    <% if devise_mapping.confirmable? && resource.pending_reconfirmation? %>
      <div>Currently waiting confirmation for: <%= resource.unconfirmed_email %></div>
    <% end %>

    <div class="form-group row">
      <%= f.label :password, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
      <div class="col-sm-9">
        <%= f.password_field :password, autocomplete: "off", class: "form-control col-sm-9",
                             placeholder: "Leave blank if you don't want to change it" %>
      </div>
    </div>

    <div class="form-group row">
      <%= f.label :password_confirmation, class: "col-sm-3 col-form-label text-right font-weight-bold" %>
      <div class="col-sm-9">
        <%= f.password_field :password_confirmation, autocomplete: "off", class: "form-control col-sm-9" %>
      </div>
    </div>

    <div class="form-group row">
      <div class="col-sm-3"></div>
      <div class="col-sm-9">
        <%= f.submit "Update", class: "btn btn-primary" %>
      </div>
    </div>
  <% end %>
 </div>
</div>
```

Routes.rb

```
Rails.application.routes.draw do
  devise_for *resources :users, controllers: { registrations: 'registrations' }
  root 'pages#home'
  # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
  resources :users, only: [:index, :show]
end
```