HTML és Rails
Gyakorlat

Kovács Gábor

2018. március 13.

A gyakorlat célja, hogy kialakítsuk a félév során megoldandó feladat képernyőit HTML-ben, ahol lehet Rails metódusok felhasználásával.

Az első lépés a webalkalmazásunk keretének kialakítása ezt a nézetek között az alkalmazásszintű nézetben, vagyis a `layouts/application.html.erb`-ben tehetjük meg. Rendezzük el úgy az oldalunkat, hogy legyen benne egy fejrész, egy központi rész, amely bal oldalon egy keskeny menüsávból áll, jobb oldalon a tartalomból, és egy lábrész. Az elrendezést `div`-ekkel valósítjuk meg, mindegyikhez egyedi `id`-t rendelve.

```
<div id="body">
  <div id="header"></div>
  <div id="main">
    <div id="menu">
    </div>
    <div id="content">
      <%= yield %>
    </div>
  </div>
  <div id="footer">Copyright, RoR 2018</div>
</div>
```

Második lépésként készítsük el az oldal stíluslapját, amivel ezek a helyükre kerülnek, és helyezzünk el benne minimális mennyiségű formázási információt. Az oldal legyen 800 pixel széles. A fejrész legyen világosszürke és 100 képpont magas. Az oldal központi része legyen 600 pixel magas. A menüsávot a központi részben, a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, magasságát a központi rész magassága definiálja. Az oldal tartalmi része világosszürke háttérrel rendelkezzen, és a menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. A lábrészben a szöveget igazítsuk középre, és legyen az is 100 pixel magas, valamint a

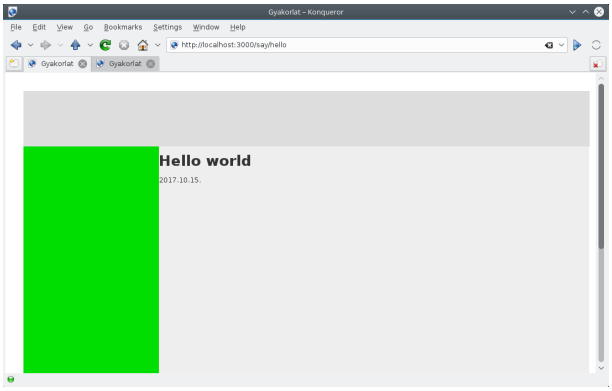
fejléccel megegyező színű.

```
div#body {
        width: 800px;
}
div#header {
        background-color: #ddd;
        height: 100px;
}
div#main {
        height: 600px;
}
div#menu {
        width: 24%;
        float: left;
        background-color: #0d0;
        height: 100%;
}
div#content {
        width: 76%;
        float: left;
        background-color: #eee;
        height: 100%;
}
div#footer {
        clear: both;
        background-color: #ddd;
        text-align: center;
        height: 100px;
}
```

Az így kialakított elrendezést például az 1. ábra mutatja.

Kétféle felhasználóra készülünk fel egyelőre, egy látogatóra és egy belépett felhasználóra, aki korábban keresztülment egy regisztrációs folyamaton. A látogató csak böngészhet, ugyanakkor regisztrálhat. A bejelentkezett felhasználó számára több funkciót is elérhetővé teszünk.

Kezdjük a látogató menüjével. Helyezzünk el a menüben egy a belépést lehetővé tevő formot! Ezt részleges rendereléssel tesszük meg. A formot tartalmazó fájl alkalmazás szinten kezeljük, ezért a `layouts` könyvtárban helyezzük el, így a be nem jelentkezett felhasználó bármelyik oldalon bejelentkezhet. A form tartalmazzon egy felhasználónévre utaló címkét és egy szövegbeviteli mezőt, valamint egy a jelszóra utaló címkét és jelszóbeviteli mezőt, továbbá egy Login feliratú nyomógombot. A formot a `form_tag` Ra-



1. ábra. Az oldal elrendezésének kialakítása

ils helperrel valósítjuk meg, aminek első paramétere a formot kezelő URL, vagyis a form `action` attribútuma, illetve adjuk meg, hogy HTTP POST-tal kívánjuk elküldeni. A form mezőit rendre a `label_tag`, `text_field_tag`, `password_field_tag` és `submit_tag` helperekkel hozzuk létre, és a beviteli mezőket 16 karakter hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé az elfelejtett jelszó visszaszerzését, ezt egy link hozzáadásával tesszük meg.

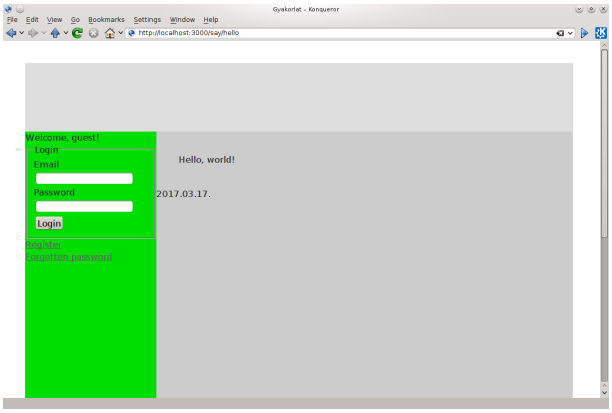
```
Welcome, guest!
<fieldset>
  <legend>Login</legend>
  <%= form_tag 'sessions/create', method: :post do %>
    <%= label_tag 'email', "Email address" %>
    <%= text_field_tag 'email', '', size: 18 %><br/>
    <%= label_tag 'password', "Password" %>
    <%= password_field_tag 'password', '', size: 18 %><br/>
    <%= submit_tag 'Login' %>
  <% end %>
</fieldset>
<%= link_to "Register", '/users/new' %><br/>
<%= link_to "Forgotten password", '/users/forgotten' %>
```

Ezután a `menu` azonosítójú `div`-ben meghivatkozhatjuk ezt az oldalt. A Rails konvenció szerint az aláhúzásjelet el kell hagynunk.

```
<div id="menu"><%= render 'layouts/guest_menu' %></div>
```

A vendégfelhasználó menüjének megvalósítását a 2. ábra mutatja.

Modellezzük azt az esetet is, amikor egy felhasználó már bejelentkezett. Ezt egy, a `helpers/application_helper` állományban elhelyezendő saját



2. ábra. A vendégfelhasználó menüje

helper metódussal tesszük meg. Itt egyelőre manuálisan állítjuk, hogy be van-e jelentkezve a felhasználó. A metódus értelemszerűen boolean visszatérési értékű.

```
module ApplicationHelper
  def logged_in?
    true
  end
end
```

Ezt visszavezetve a keretbe a menüt megvalósító `menu`-ban a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról. „Jelentkeztessük" be és ki a felhasználót, hogy ellenőrizhessük, hogy a megfelelő menü jelenik-e meg a vendég és a felhasználó számára.

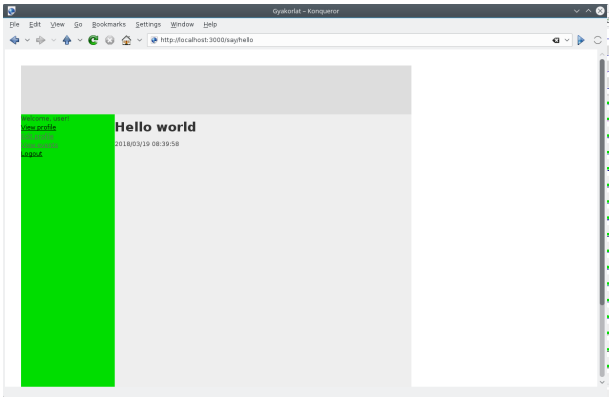
```
<% if logged_in? %>
  ...
<% else %>
  ...
<% end %>
```

A bejelentkezett felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre. Egyelőre négy akciót definiálunk: az események listájának megtekintését, a profiloldal megtekintését, valamint szerkesztését és a kijelentkezést.

```
Welcome, user!
<br/>
```

```
<%= link_to "View profile", '/users/show' %><br/>
<%= link_to "Edit profile", '/users/edit' %><br/>
<%= link_to "View events", events_path %><br/>
<%= link_to "Logout", 'sessions/destroy' %>
```

A bejelentkezett felhasználó menüjének megvalósítását a 3. ábra mutatja.



3. ábra. A bejelentkezett felhasználó menüje

Nézzük meg a be nem lépett felhasználó regisztációs folyamatát! Az előző gyakorlat alkalmával már létrehoztuk a felhasználó modellünk kezdetleges változtatás, így arra már tudhatunk hivatkozni egy Rails formban, amely az MVC tervezési minta szerint szorosan kapcsolódik a nézethez. Hozzuk létre a felhasználó nézetét és fontosabb akcióit a következő paranccsal:

```
kovacsg@debian:~/gyakorlat# rails g controller users new
    edit forgotten show
      create  app/controllers/users_controller.rb
       route  get 'users/show'
       route  get 'users/forgotten'
       route  get 'users/edit'
       route  get 'users/new'
      invoke  erb
      create    app/views/users
      create    app/views/users/new.html.erb
      create    app/views/users/edit.html.erb
      create    app/views/users/forgotten.html.erb
      create    app/views/users/show.html.erb
      invoke  test_unit
      create    test/controllers/users_controller_test.rb
      invoke  helper
```

```
      create    app/helpers/users_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/users.coffee
      invoke    scss
      create      app/assets/stylesheets/users.scss
```

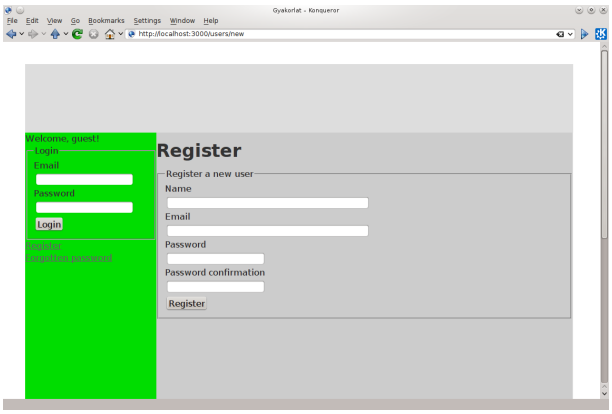
A parancs futtatásával létrejött az `users` kontroller és a hozzá kapcsolódó nézetek köztük az új felhasználó létrehozását lehetővé tevő `new`, a felhasználói profil szerkesztését megvalósító `edit`, a felhasználói adatait megjelenítő `show`, és az elfelejtett jelszó esetén az email címet elkérő `forgotten` nézet. Az tervezői kérdés, hogy az elfelejtett jelszó kezelését a felhasználók kontrollere részének tekintjük, vagy önálló kontrollert hozunk létre számára. A gyakorlaton amellett döntöttünk, hogy az elfelejtett jelszó kerüljön a felhasználók kontrollerébe.

Hozzunk mindjárt létre a regisztrációs nézetet! Legyen egy címsorunk, amely elmondja a felhasználónak, hogy melyik oldalon van. Az esetleges hibaüzeneteknek tartsunk fenn helyet. Ezután egy `fieldset`-ben definiáljuk egy formot, amely ez esetben egy konkrét, létező modellhez van kötve. Ezt a `form_for` Rails helperrel tehetjük meg. Ennek első paramétere a modell neve szimbólum formájában, második paramétere a formhoz kötött akció, amely legyen a `users` kontroller `create` akciója, a harmadik paramétere a HTTP metódus, ami POST. A metódus blokkjának van egy paramétere a `form`, amin kereszül definiáljuk fogjuk az űrlap elemeit. Legyen az öt elem rendre a következő: egy 40 karakter széles, a felhasználó nevére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 40 karakter széles, a felhasználó email címére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 20 karakter széles, a felhasználó telefonszámára vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, két darab 20 karakter széles jelszóbeviteli mező a hozzájuk kapcsolódó címkével, a két jelszómező eltérő azonosítóval rendelkezzék, az egyik prefixe `_confirmation`-re végződjék.

```
<h1>Register</h1>
<fieldset>
  <legend>Register a new user</legend>
  <%= form_for :user, url: { controller: 'users', action: '
      create' }, method: :post do |form| %>
    <div>
      <%= form.label :name %>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
```

```
      <%= form.label :email %>
      <%= form.text_field :email, size: 40 %>
    </div>
    <div>
      <%= form.label :password %>
      <%= form.password_field :password, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %>
      <%= form.password_field :name, size: 20 %>
    </div>
    <%= form.submit "Register" %>
  <% end %>
</fieldset>
<%= link_to "Back", :back %>
```

A felhasználói regisztráció nézetét a 4. ábra mutatja.



4. ábra. A regisztráció nézete

A létrejött oldal HTML forrását tekintve a következőt látjuk. A formok mezőinek `name` és `id` attribútuma tartalmazza a modell nevét és a mező nevét. A név attribútum Ruby hash mintájára készült el, a modell nevének hash-ére hivatkozik a mező Rails forrásban megadott neve. Az általunk megadott mezőkön kívül létrejött két hidden mező is, amelyek a form használójának hitelesítését hivatottak ellenőrizni. A visszalépés itt JavaScripttel valósul meg. A forrást megtekintve láthatjuk, hogy a `:password_confirmation` szimbólumból a Rails automatikusan a *Password confirmation* szöveget állította elő. A stringek és a szimbólumok így ezen elv mentén felcserélhetők a form helperek argumentumlistájában.

A formok eseménykezelőihez a generate controller parancsunk nem generált útvonalakat, ezért azokat felvesszük a `config/routes.rb`-be. Jelenleg a nézetek kialakításához szingleton példányokkal dolgozunk, vagyis egyetlen felhasználó adatait vagyunk képesek megjeleníteni. Később szükségünk lesz egy `:id` szimbólumra, ami képessé tesz minket a felhasználók megkülönböztetésére, és azonos nevű paraméterként jelentkezik majd a kontrollerben.

```
  get 'users/new'
  post 'users/create'

  get 'users/edit'
  put 'users/update'

  get 'users/forgotten'
  post 'users/send_forgotten'

  get 'users/show'
```

A form eseményét a Rails konvenció szerint a `create` kontroller metódus fogja kezelni. Ez még nem létezik, ezért definiáljuk azt egyelőre üres törzzsel.

A felhasználói profil szerkesztésének nézetében (`edit.html.erb`) található form szinte teljesen megegyezik az új felhasználót létrehozó formmal. A felhasználónév módosítását inaktívvá tehetjük, illetve az eseménykezelő kontroller akciót kell módosítanunk. A form eseményét a Rails konvenció szerint a `update` kontroller metódus fogja kezelni, erre létrehozzuk az útvonalat. Ez még nem létezik, ezért definiáljuk ezt is egyelőre szintén üres törzzsel. Ezen kívül a nézetben a feliratokat kell még átírnunk regisztrációról profil szerkesztésére.

```
<h1>Edit profile</h1>
<fieldset>
  <legend>Edit user profile</legend>
  <%= form_for :user, url: { controller: 'users', action: '
      update' }, method: :put do |form| %>
    <div>
      <%= form.label :name %>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :email %>
      <%= form.text_field :email, size: 40 %>
    </div>
    <div>
      <%= form.label :password %>
```

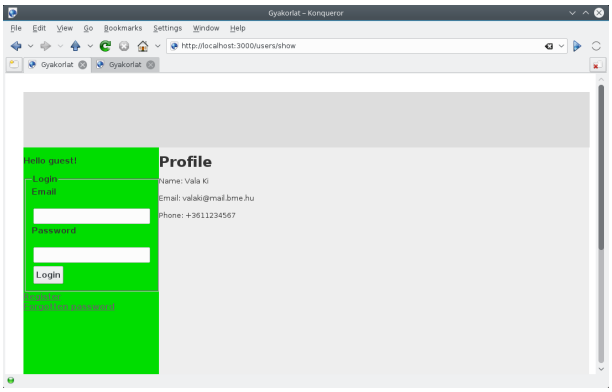
```
      <%= form.password_field :password, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %>
      <%= form.password_field :name, size: 20 %>
    </div>
    <%= form.submit "Save" %>
  <% end %>
</fieldset>
<%= link_to "Back", :back %>
```

Mivel a `form_for` Rails helper metódust használtuk a form létrehozására a `new` és az `edit` nézetekben, ezért szükséges a megfelelő kontroller akciókban a `@user` példányváltozó inicializálása. Ezeket egyelőre ne adatbázisból tegyük meg, hanem statikus tartalommal töltsük fel. Míg a `new` esetén a felhasználó még nem lézetik az adatbázisban, attribútumai inicializálatlanok, ezért elégséges egy frissen létrehozott példány használata, addig az `edit` esetén már ki kell töltenünk a struktúra mezőit beleértve az adatbázisbeli azonosító `id` attribútumot is.

```
class UsersController < ApplicationController
  def new
  end

  def edit
    @user = User.new
    @user.id = 1
    @user.name = 'Valaki'
    @user.email = 'valaki@mail.bme.hu'
  end

  def forgotten
  end

  def show
    @user = User.new
    @user.id = 1
    @user.name = 'Valaki'
    @user.email = 'valaki@mail.bme.hu'
  end
end
```

A felhasználói profiloldal szerkesztésének nézetét az 5. ábra mutatja. Láthatjuk, hogy a Rails automatikusan inicializálta a form mezőit, ahol a

hozzájuk tartozó érték elérhető volt – a jelszó mezők kivételével.



5. ábra. A profiloldal nézete

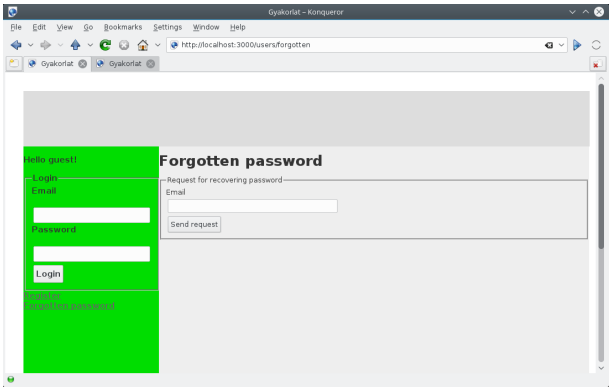
Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<h1>Forgotten password</h1>
<fieldset>
  <legend>Please give your your email address</legend>
  <%= form_for :user, url: { controller: 'users', action: '
      send_forgotten' }, method: :post do |form| %>
    <div>
      <%= form.label :email %>
      <%= form.text_field :email, size: 40 %>
    </div>
    <%= form.submit "Recover_password" %>
  <% end %>
</fieldset>
```

Az elfelejtett jelszó kiküldését a form eseményét kezelő kontroller akció, a `send_forgotten` teszi majd meg, amit fel kell vennünk a kontroller osztályába egyelőre üres törzzsel.

Az elfelejtett jelszó nézetét a 6. ábra mutatja.

Az eseménynyilválntartó portálunk központi elemei maguk az események, amelyeket az előző gyakorlaton `scaffold`-dal már létrehoztunk. Az események kontrollerében négy nézetet létezik: `index`, `new`, `edit`, `show`. Az `index` nézetet átalakítjuk úgy, hogy a táblázatból eltávolítjuk az adminisztratív funkciókat. A szerkesztés és a törlés csakis a `show` nézeten érhető el, ezeket a funkciókat átmozgatjuk oda, a `show` nézet maga pedig az esemény nevére



6. ábra. Az elfelejtett jelszó nézete

való kattintással érhető el. A tulajdonos felhasználó objektum memóriacíme helyett pedig a felhasználó nevét jelenítjük meg.

```
<p id="notice"><%= notice %></p>

<h1>Events</h1>

<table>
  <thead>
    <tr>
      <th>User</th>
      <th>Title</th>
      <th>Description</th>
      <th>Deadline</th>
      <th>Priority</th>
    </tr>
  </thead>

  <tbody>
    <% @events.each do |event| %>
      <tr>
        <td><%= event.user.name %></td>
        <td><%= link_to event.title, event %></td>
        <td><%= event.description %></td>
        <td><%= event.deadline %></td>
        <td><%= event.priority %></td>
      </tr>
    <% end %>
  </tbody>
```

```
</table>

<br>

<%= link_to 'New Event', new_event_path %>
```

A beágyazott nézetet a `show` nézet az esemény adatlapját mutatja. Ez kibővítjük az `index` nézetről átvett funkciókkal, megjelenítjük a résztvevők listáját egy felsorolással, és egy formmal lehetővé tesszük felhasználók az eseményre való meghívását. A felsorolás elemeit egy `for` ciklusba ágyazott kódrészlet állítja elő, a ciklus az esemény `participants` mezőjének elemein iterál végig. Ez utóbbit a `select_tag` helper segítségével tesszük meg, amelynek első paramétere az esemény neve, a második paramétere pedig a megjelenítendő opciók, melyeket az `options_for_select` helperrel állítunk elő egy tömbök tömbje típusú objektumból. A belső tömbök első, nulladik indexű eleme lesz a felirat, a második, egy indexű eleme pedig a `value` mező értéke.

```
<p id="notice"><%= notice %></p>

<p>
  <strong>User:</strong>
  <%= @event.user.name %>
</p>

<p>
  <strong>Title:</strong>
  <%= @event.title %>
</p>

<p>
  <strong>Description:</strong>
  <%= @event.description %>
</p>

<p>
  <strong>Deadline:</strong>
  <%= @event.deadline %>
</p>

<p>
  <strong>Priority:</strong>
  <%= @event.priority %>
</p>
```

```
<p>
  <strong>Participants:</strong>
  <ul>
  <% for i in @event.participants %>
    <li><%= i.name %></li>
  <% end %>
  </ul>
</p>

<p>
  <strong>Invite user:</strong>
  <%= form_tag 'invite/create', method: :post do %>
    <%= select_tag 'user', options_for_select([["Mi", 4], [
        "Ti", 5]]) %>
    <%= submit_tag 'Invite' %>
  <% end %>
</p>

<%= link_to 'Edit', edit_event_path(@event) %> |
<%= link_to 'Back', events_path %>
```

Mivel a `participants` nem eleme az esemény adatstruktúrájának, fel kell vennünk ideiglenesen egy setter-getter párost hozzá, és ezután az események kontrollerében inicializálhatjuk annak kezdeti értékét.

```
class Event < ApplicationRecord
  attr_accessor :participants
  belongs_to :user
end
```