# HTML és Rails
# Gyakorlat

Kovács Gábor

2018. október 9.

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

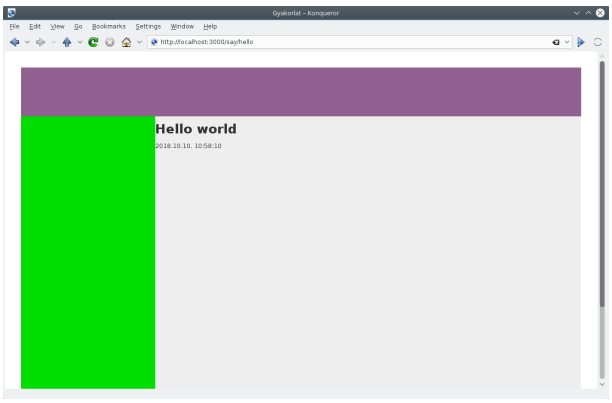
Az első lépés a webalkalmazásunk keretének kialakítása ezt a nézetek között az alkalmazásszintű nézetben, vagyis a `layouts/application.html.erb`-ben tehetjük meg. Rendezzük el úgy az oldalunkat, hogy legyen benne egy fejrész, egy központi rész, amely bal oldalon egy keskeny menüsávból áll, jobb oldalon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

```
<div id="page">
  <div id="header"></div>
  <div id="menu">
  </div>
  <div id="main">
    <%= yield %>
  </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.

```
#page {
 height: 800px;
}
#header {
  height: 100px;
  background-color: #906090;
}
#footer {
  height: 100px;
  clear: both;
  background-color: #dddddd;
  text-align: center;
}
#menu {
  width: 24%;
  float: left;
  height: 600px;
  background-color: #0d0;
}
#main {
  width: 76%;
  height: 600px;
  float: left;
  background-color: #eee;
}
```

Az így kialakított elrendezést például az 1. ábra mutatja.




1. ábra. Az oldal elrendezésének kialakítása

Háromféle felhasználóra készülünk fel egyelőre, egy látogatóra és egy belé-

pett felhasználóra, aki lehet hallgató vagy oktató, és korábban keresztülment egy regisztrációs folyamaton. A látogató csak böngészhet, bejelentkezhet és jelszóemlékeztetők kérhet. 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 Rails konvenció szerint a részlegesen renderelt állományok neve aláhúzásjellel kezdődik. Legyen a fájlunk neve ezért `_loginform.html.erb`! 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` Rails 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.

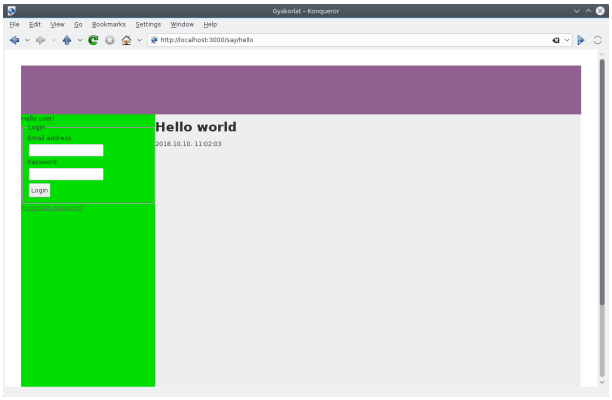
```
Hello user!
<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', method: :post do %>
    <%= label_tag 'email', "Email address" %>
    <%= text_field_tag 'email', '', size: '18' %><br/>
    <%= label_tag 'password', "Password" %>
    <%= password_field_tag 'password', '', size: '18' %><br
        />
    <%= submit_tag 'Login' %>
  <% end %>
</fieldset>
<%= link_to "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/loginform' %></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 helper metódussal tesszük meg. Itt egyelőre manuálisan állítjuk, hogy be van-




2. ábra. A vendégfelhasználó menüje

e jelentkezve a felhasználó. A metódus értelemszerűen boolean visszatérési értékű.

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

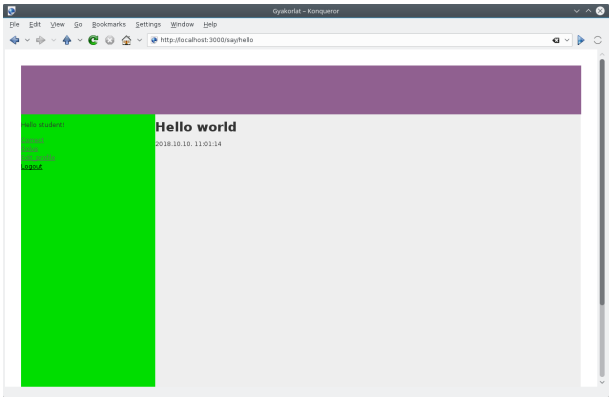
Ezt visszavezetve a keretbe a menüt megvalósító `menu`-ban a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról. „Jelentkeztessük" be és ki a felhasználót, hogy ellenőrizhessük, hogy a megfelelő menü jelenik-e meg a vendég és a felhasználó számára.

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

A bejelentkezett hallgató típusú felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre (`_student_menu.html.erb`). Egyelőre négy akciót definiálunk: az események listájának megtekintését, a profiloldal megtekintését, valamint szerkesztését és a kijelentkezést.

```
<p>Hello student!</p>
<%= link_to 'Correct', '/quizzes/3' %><br/>
<%= link_to 'Solve', '/quizzes/3' %><br/>
<%= link_to 'Edit_profile', '/users/edit' %><br/>
<%= link_to 'Logout', '/sessions/destroy' %><br/>
```

A bejelentkezett hallgató típusú felhasználó menüjének megvalósítását a 3. ábra mutatja.




3. ábra. A bejelentkezett felhasználó menüje

A bejelentkezett oktató típusú felhasználó menüjét ismételten egy helperrel különböztetjük meg a bejelentkezett hallgató típusú felhasználó menüjétől.

```
<% if is_lecturer? %>
 <%= render 'layouts/lecturer_menu' %>
<% else %>
 <%= render 'layouts/student_menu' %>
<% end %>
```

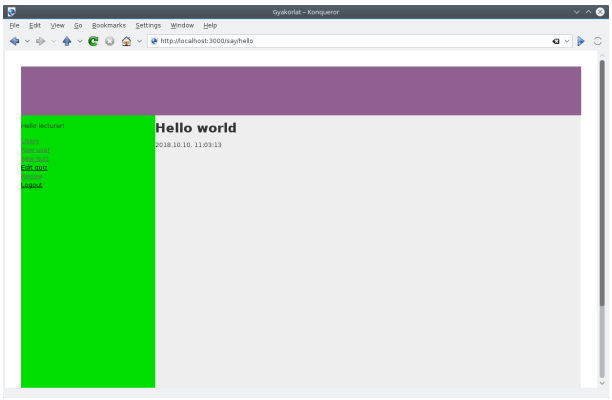
A helpert is a `logged_in?` helper mintájára készítjük el.

```
module ApplicationHelper
  def is_lecturer?
    true
  end
end
```

A menü pedig álljon a következő elemekből:

```
<p>Hello lecturer!</p>
<%= link_to 'Users', '/users/index' %><br/>
<%= link_to 'New user', '/users/new' %><br/>
<%= link_to 'New quiz', '/quizzes/new' %><br/>
<%= link_to 'Edit quiz', '/quizzes/3/edit' %><br/>
<%= link_to 'Review', '/tasks' %><br/>
<%= link_to 'Logout', '/sessions/destroy' %><br/>
```

A bejelentkezett oktató típusú felhasználó menüjének megvalósítását a 4. ábra mutatja.




4. ábra. A bejelentkezett felhasználó menüje

Nézzük meg a be nem lépett felhasználó létrehozásának 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/app/views/layouts# rails g
    controller users index new edit forgotten
      create  app/controllers/users_controller.rb
       route  get 'users/index'
get 'users/new'
get 'users/edit'
get 'users/forgotten'
      invoke  erb
      create    app/views/users
      create    app/views/users/index.html.erb
      create    app/views/users/new.html.erb
      create    app/views/users/edit.html.erb
      create    app/views/users/forgotten.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 a felhasználók listáját tartalmazó `index`, 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 a hat elem rendre a következő: egy 30 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 6 karakter széles, a felhasználó Neptun-kódjára vonatkozó szövegbeviteli mező, 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, és végül egy a felhasználó típusát megadni hivatott szövegbeviteli mező, amelyet később legördülő menüre cserélünk le.

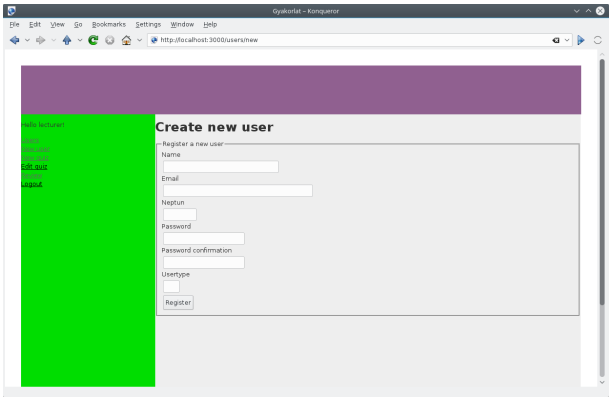
```
<h1>Create new user</h1>
<div>
  <fieldset>
    <legend>Register a new user</legend>
    <%= form_for :user, url: '/users/create', method: :post
        do |form| %>
      <div>
        <%= form.label :name %>
        <%= form.text_field :name, size: 30 %>
      </div>
```

```
      <div>
        <%= form.label :email %>
        <%= form.text_field :email, size: 40 %>
      </div>
      <div>
        <%= form.label :neptun %>
        <%= form.text_field :neptun, size: 6 %>
      </div>
      <div>
        <%= form.label :password %>
        <%= form.password_field :password, size: 20 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>
        <%= form.password_field :password_confirmation,
            size: 20 %>
      </div>
      <div>
        <%= form.label :usertype %>
        <%= form.text_field :usertype, size: 1 %>
      </div>
      <div>
        <%= form.submit 'Register' %>
      </div>
    <% end %>
  </fieldset>
</div>
<%= link_to "Back", :back %>
```

A felhasználói regisztráció nézetét a 5. ábra mutatja.

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 hashére 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




5. ábra. A regisztráció nézete

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/index'
  get 'users/new'
  post 'users/create'
  get 'users/edit'
  put 'users/update'
  get 'users/forgotten'
  post 'users/recover'
  delete 'users/destroy'
```

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 user profile</h1>
<div>
  <fieldset>
    <legend>Edit user profile</legend>
    <%= form_for :user, url: '/users/update', method: :put
```

```
        do |form| %>
      <div>
        <%= form.label :name %>
        <%= form.text_field :name, size: 30 %>
      </div>
      <div>
        <%= form.label :email %>
        <%= form.text_field :email, size: 40 %>
      </div>
      <div>
        <%= form.label :neptun %>
        <%= form.text_field :neptun, size: 6 %>
      </div>
      <div>
        <%= form.label :password %>
        <%= form.password_field :password, size: 20 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>
        <%= form.password_field :password_confirmation,
            size: 20 %>
      </div>
      <div>
        <%= form.label :usertype %>
        <%= form.text_field :usertype, size: 1 %>
      </div>
      <div>
        <%= form.submit 'Update' %>
      </div>
    <% end %>
  </fieldset>
</div>
```

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 index
```

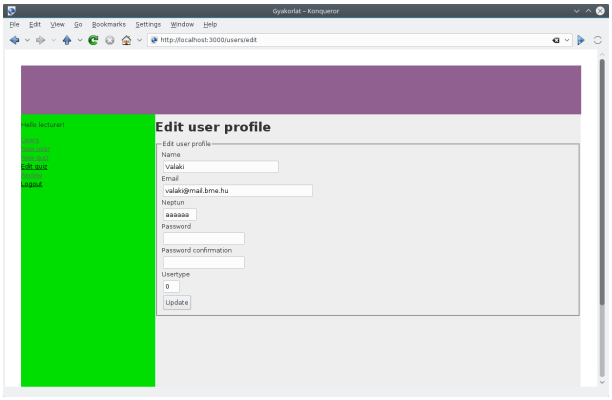
```
    @users = User.all
  end

  def new
    @user = User.new
  end

  def create
  end

  def edit
    @user = User.first
  end

  def update
  end

  def destroy
  end

  def forgotten
  end

  def recover
  end
end
```

A felhasználói profiloldal szerkesztésének nézetét az 6. ábra mutatja. Láthatjuk, hogy a Rails automatikusan inicializálta a form mezőit, ahol a hozzájuk tartozó érték elérhető volt – a jelszó mezők kivételével.

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

```
<h1>Recover forgotten password</h1>
<fieldset>
  <legend>Please, give your email address</legend>
  <%= form_tag '/users/recover', method: :post do %>
    <%= text_field_tag 'name', '', size: '50' %><br/>
    <%= submit_tag 'Recover' %>
  <% end %>
</fieldset>
```

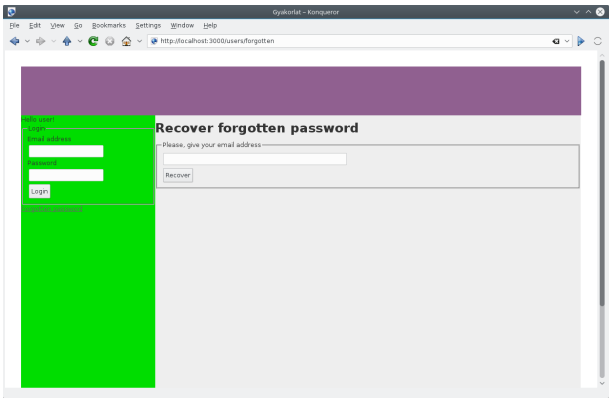
Az elfelejtett jelszó kiküldését a form eseményét kezelő kontroller akció, a `recoer` teszi majd meg, amit fel kell vennünk a kontroller osztályába egyelőre




6. ábra. A profiloldal nézete

üres törzzsel.

Az elfelejtett jelszó nézetét a 7. ábra mutatja.



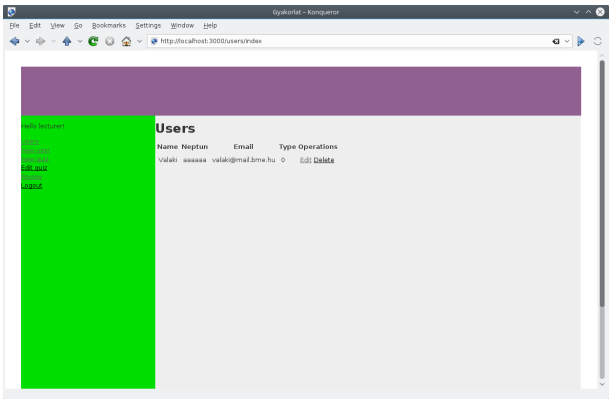

7. ábra. Az elfelejtett jelszó nézete

A csak az oktató típusú felhasznlók számára látható felhasználók listája nézeten (`index`) egy táblátot helyezünk el, amelyben látható a felhasználó neve, Neptun-kódja, email címe, típusa és szerkesztés és törlés műveleteket hajthatunk végre a sorhoz tartozó elemen. A felhasználókon egy `for` ciklussal iterálunk, és mindegyikhez létrehozunk egy sort a táblázatban

```
<h1>Users</h1>
<table>
  <thead>
    <th>Name</th>
    <th>Neptun</th>
```

```
    <th>Email</th>
    <th>Type</th>
    <th>Operations</th>
  </thead>
  <tbody>
    <% for user in @users do %>
      <tr>
        <td><%= user.name %></td>
        <td><%= user.neptun %></td>
        <td><%= user.email %></td>
        <td><%= user.usertype %></td>
        <td>
          <%= link_to "Edit", '/users/edit' %>
          <%= link_to "Delete", '/users/destroy', method: :
              delete %>
        </td>
      </tr>
    <% end %>
  </tbody>
</table>
```

Az index nézetet a 8. ábra mutatja.




8. ábra. A felhasználók karbantartása

A felhasználó törlését megvalósító művelet, a `destroy` nem rendelkezik önálló nézettel.

A házifeladatjavító portálunk két elemét, a feladatsort és a feladatot az előző gyakorlaton `scaffold`-dal már létrehoztunk. Ezeket most kiegészítjük a javítás modellel, majd végrehajtuk a migrációt.

```
kovacsg@debian:~/gyakorlat$ rails g scaffold review mark:
    integer comment:string
       invoke  active_record
       create    db/migrate/20181009113053_create_reviews.rb
       create    app/models/review.rb
       invoke    test_unit
       create      test/models/review_test.rb
       create      test/fixtures/reviews.yml
       invoke  resource_route
        route    resources :reviews
       invoke  scaffold_controller
       create    app/controllers/reviews_controller.rb
       invoke    erb
       create      app/views/reviews
       create      app/views/reviews/index.html.erb
       create      app/views/reviews/edit.html.erb
       create      app/views/reviews/show.html.erb
       create      app/views/reviews/new.html.erb
       create      app/views/reviews/_form.html.erb
       invoke    test_unit
       create      test/controllers/reviews_controller_test.
          rb
       create      test/system/reviews_test.rb
       invoke    helper
       create      app/helpers/reviews_helper.rb
       invoke      test_unit
       invoke    jbuilder
       create      app/views/reviews/index.json.jbuilder
       create      app/views/reviews/show.json.jbuilder
       create      app/views/reviews/_review.json.jbuilder
       invoke  assets
       invoke    coffee
       create      app/assets/javascripts/reviews.coffee
       invoke    scss
       create      app/assets/stylesheets/reviews.scss
       invoke  scss
    identical    app/assets/stylesheets/scaffolds.scss
kovacsg@debian:~/gyakorlat/app/views/reviews# rails db:
    migrate
== 20181009113053 CreateReviews: migrating
    ===============================================
-- create_table(:reviews)
    -> 0.0204s
```

```
== 20181009113053 CreateReviews: migrated (0.0207s)
   ================================
```

Az mindhárom kontrollerben négy nézetet létezik: `index`, `new`, `edit`, `show`. A `show` nézetet használjuk a hallgató típusú felhasználók esetén a feladat szerkesztésére, illetve a javítás szerkesztésére, az összes többi funkció csak az oktató típusú felhasználó számára lesz elérhető.

A feladatsor feladatokból áll, ezért a feladatsor kitöltéséhez a feladatsor `show` nézetébe beágyazzuk a feladatsor `index` nézetenek a feladatokat felsoroló változatát, amelyet az `index` nézetből klónozunk `tasks/_index.html.erb` néven.

```
<%= render 'tasks/index' %>
```

A javítást szerkesztő nézet a `reviews/_form.html.erb`-ben van, így azt egy feladat megoldását javítását lehetővé tevő a `tasks/_form.html.erb`, de csak akkor, ha javításról, és nem feladatmegoldásról van szó. Egyelőre általánosan elhelyezzük ott. A javítás űrlapnak átadandó egy `Review` objektum, amelyet most helyben hozunk létre.

```
<%= render 'reviews/form', review: Review.new %>
```