HTML és Rails
Gyakorlat

Kovács Gábor

2014. október 14.

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="banner"></div>

<div id="menu">
</div>

<div id="main">
<%= yield %>
</div>

<div id="footer">
Copyright, RoR, 2014
</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. A menüsávot a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, és legyen 800 pixel magas. Az oldal tartalmi része legyen 800 képpont magas, világosszürke háttérrel rendelkezzen, és a

menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. A lábrészben a szöveget igazítsuk középre, és legyen a lábrész magassága 100 képpont.

```
div#page {
  width: 800px;
}

#banner {
  background-color: #dddddd;
  height: 100px;
}

#menu {
  float: left;
  background-color: #cccccc;
  height: 800px;
  width: 24%;
}

#main {
  float: left;
  background-color: #eeeeee;
  height: 800px;
  width: 76%;
}

#footer {
  background-color: #dddddd;
  height: 100px;
  text-align: center;
  clear: both;
}
```

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. 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 18 hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé az elfelejtett jelszó visszaszerzését, ezt egy link hozzáadásával tesszük meg.

```
<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', :method => :post  do %>
    <%= label_tag 'username', "Username" %> <br />
    <%= text_field_tag 'username', '', :size => 14 %> <br/>
    <%= label_tag 'password', "Password" %> <br />
    <%= password_field_tag 'password', '', :size => 14 %> <
        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.

```
<%= render 'layouts/loginform' %>
```

Modellezzük azt az esetet is, amikor egy felhasználó már bejelentkezett. Ezt egy saját helper metódussal tesszük meg, amit a `helpers/application_helper` állományba helyezünk el. Itt egyelőre manuálisan állítjuk, hogy be van-e jelentkezve a felhasználó. A metódus értelemszerűen boolean visszatérési értékű.

```
module ApplicationHelper
  def logged_in?
    false
  end
end
```

Ezt visszavezetve a keretbe a menüt megvalósító `loginform`-ban a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról. „Jelenkteztessük” be és ki a felhasználót, hogy el-

lenőrizhessük, hogy a megfelelő menü jelenik-e meg a vendég és a felhasználó számára.

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

Hozzuk létre a játékok modelljét és nézeteit. A strukturált megjelenítésért csaláshoz folyamodunk, és mindezt egyetlen paranccsal, a `scaffold`-dal tesszük meg. Ez létrehozza a modell osztályt, a kapcsolódó adatbázis-migrációt, a tesztadatok fájlját, a modell egységtesztjeit, a kontroller osztályát, a kapcsolódó nézetek kezdeti változatát, a kontrollerek funkcionális tesztjeit, illetve a nézetekhez kapcsolódó útvonalakat. A parancs úgy nevezett resourceful útvonalakat definiál a `routes.rb`-ban, amelyek eltérnek az eddig megszokottaktók (parancssoron a `rake routes` paranccsal nézhetjük meg, hogy mi változott). A játékok adatstruktúrája tartalmazza a játékban részt vevő két szereplőt, a játék végeredményét, a játék kezdetét és a játék utolsó lépését. A parancs után hajtsuk végre az adatbázis migrációját, és nézzük meg az eredményt!

```
rails generate scaffold game player1:integer player2:
  integer result:string
```

```
rake db:migrate
```

Négy nézet jött létre: `index`, `new`, `edit`, `show`, amelyek egyszerű táblázatos formában jelenítik meg az összes vagy az azonosító HTTP kérés paraméterben megadott modell példány adatait. A nézetekhez kapcsolódó kontroller akciókon kívül további három metódus jön létre a kontrollerben: `create`, `update`, `destroy`. Ezek együttesen megfelelnek a RESTful HTTP filozófiájának.

Nézzük meg a létrejött nézeteket! Az `index` nézet egy fejléccel rendelkező táblázatban megjeleníti a `Game` osztály példányait külön sorokban, az oszlopokban az attribútumok találhatók. A táblázatban három extra oszlop található fejléc nélkül, amelyek a `show`, a `edit` nézetre mutató linkeket, illetve egy az objektumok törlő, a `destroy` akcióra mutató linket tartalmaz. E három link paraméterezve van a játék azonosítójával, így az mindig a táblázat megfelelő sorában megjelenő játékra vonatkozik. Az oldal alján egy új játék létrehozását lehetővé tevő nézetre mutató link található. Az, hogy e linkek közül melyek lesznek elérhetők a vendég felhasználók, a bejelentkezett felhasználók és az adminisztrátor számára, egy későbbi döntés kérdése.

A `new` és a `edit` nézetek egy linktől és egy címkétől eltekintve azonosok, mindkét nézet megjeleníti a `form` nevű részleges nézetet. A `form` nézet egymás utáni `<div>` elemekben a `Game` adatstruktúrája mezői típusainak megfelelő adatbeviteli mezők jelennek meg egy a mező nevével megegyező címke után. Minden típus előre definiált adatbeviteli mezőre képeződik le, így az időpont öt darab legördülő menüre, a string és az egész szövegbeviteli mezőre. Az oldalak tetején megjelenhetnek az esetleges hibaüzenetek.

A `show` nézet a játék adatstruktúra mezőinek értékeit írja ki, de nem abban a formátumban, ahogy az nekünk majd kelleni fog. Módosítsuk a nézetet, írjuk ki a két játékost, rajzoljunk egy táblát, amin a játék zajlani fog, és írjuk ki az eredményt. Az alábbi kódrészletből a tábla üres rácsát kirajzoló `tr` és `td` elemekből álló részt kitakartam.

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

<!--p>
  <strong>Player1:</strong>
  <%= @game.player1 %>
</p>

<p>
  <strong>Player2:</strong>
  <%= @game.player2 %>
</p-->

<p><%= @game.player1%> - <%= @game.player2 %>


<table border="1">
...
</table>

<p>
  <strong>Result:</strong>
  <%= @game.result %>

</p>

<%= link_to 'Edit', edit_game_path(@game) %> |
<%= link_to 'Back', games_path %>
```

Töltsük fel az `index` nézet táblázatát adatokkal, a nézetben táblázatunk törzsét kiíró `for` ciklusban láthatjuk, hogy a `@games` kontroller példányválto-

zó elemein iterálunk végig. A kontrollerben megnézve e változót, azt látjuk, hogy ez az adatbázisból betölti az összes rekordját, azonban mivel jelenleg nincsenek ott adataink, inicializáljuk másképpen e változót.

```
def index
  #@games = Game.all
  @game1 = Game.new
  @game1.player1 = 1
  @game1.player2 = 2
  @game1.result = '1 nyert'
  @game1.created_at = Time.now
  @game1.updated_at = Time.now
  @game1.id = 1
  @game2 = Game.new
  @game2.player1 = 1
  @game2.player2 = 2
  @game2.result = '2 nyert'
  @game2.created_at = Time.now
  @game2.updated_at = Time.now
  @game2.id = 2
  @games = [@game1, @game2]
end
```

Hasonlóan az `edit` és a `show` nézetek `@game` kontroller példányvátozóit is inicializáljuk a `set_game` nevű privát metódusban, majd ellenőrzizhetjük, hogy megjelennek-e a nézeten ezek az adatok. Az `edit` nézet esetén azt látjuk, hogy az adatstruktúra mezői értéke alapján inicializálódtak a form adatbeviteli mezői..

```
def set_game
  #@game = Game.find(params[:id])
  @game = Game.new
  @game.player1 = 1
  @game.player2 = 2
  @game.result = '1 nyert'
  @game.created_at = Time.now
  @game.updated_at = Time.now
  @game.id = 1
end
```

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:

```
rails generate controller users new edit index show
    forgotten
```

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`, az ellenfél és barátnak jelölhető felhasználó keresését lehetővé tevő `index`, a felhasználói profilt megjelenítő `show`, és a hiányzó jelszót pótló `forgotten` felület, illetve akció. 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 kontrollenébe.

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 konvenciót követve. A metódus blokkjának van egy paramétere a `form`, amin kereszül definiáljuk fogjuk az űrlap elemeit. Legyen a hat elem rendre a következő: egy tizennégy karakter széles felhasználónévre vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy harminc karakter széles email cím szövegbeviteli mező a hozzá kapcsolódó címkével, két darab 14 karakter széles jelszóbeviteli mező a hozzájuk kapcsolódó címkével, a két jelszómező eltérő azonosítóval rendelkezzék, az egyik prefixe `_confirmation`-re végződjék, egy szövegbeviteli mező a bankszámlaszámnak, és végül egy nyomógomb. Az oldal alján helyezzünk el egy visszalépés gombot azon felhasználóknak, akik véletletül tévedtek ide. A `:back` szimbólummal hivatkozott értékben a Rails a `HTTP_REFERER` HTTP fejléc elem vagy a Javascript `history` attribútuma alapján tárolja a megelőző HTTP kérésben használt URI-t.

```
<h1>Registration</h1>

<%= flash[:error] %>

<fieldset>
<legend>Register a new user</legend>
<%= form_for @user, :url => { :action => 'create' } do |f|
    %>
  <div>
    <%= f.label :username %> :
    <%= f.text_field :username, :size => 14 %>
```

```
  </div>
  <div>
    <%= f.label :email %> :
    <%= f.text_field :email, :size => 30 %>
  </div>
  <div>
    <%= f.label :password %> :
    <%= f.password_field :password, :size => 14 %>
  </div>
  <div>
    <%= f.label :password_confirmation %> :
    <%= f.password_field :password_confirmation, :size =>
        14 %>
  </div>
  <%= f.submit "Register" %>
<% end %>
</fieldset>

<%= link_to "Back", :back %>
```

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 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 kell inaktívvá tennünk, illetve az eseménykezelő kontroller akciót kell módosítanunk. A form eseményét a Rails konvenció szerint a `update` kontroller metódus fogja kezelni. 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>Profile</h1>

<%= flash[:error] %>
```

```
<fieldset>
<legend>Edit user profile</legend>
<%= form_for @user, :url => { :action => 'update' } do |f|
    %>
  ..
  <%= f.submit "Update" %>
<% 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, akárcsak a játék kontroller esetén, 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.

```
def new
  @user = User.new
end

def edit
  @user = User.new
  @user.id = 1
  @user.username = 'Valaki'
  @user.email = 'valaki@mail.bme.hu'
  @user.password = 'titok'
end
```

A felhasználók listáját megjelenítő `index` nézetben a `@users` példányváltozóban tárolt halmaz elemein lépkedjünk végig egy ciklussal, és írjuk ki a felhasználók nevét egy listában. A felhasználónevek mellé helyezzünk el két linket: a kihívás linkjét és a barátnak jelölés linkjét. Az előbbi mutasson az új játék létrehozása linkre, aminek át kell adnunk majd később két felhasználóazonosító, az utóbbi mutasson a `friend` linkre, aminek át kell majd adnunk a barátnak jelölt felhasználó azonosítóját.

```
<h1>Users</h1>
<ul>
<%= for u in @users do %>
```

```
  <li><%= u.username %> <%= link_to "Challenge", "/games/
      new" %> <%= link_to "Friend", "/users/friend" %></li>
<% end %>
</ul>
```

Az `index` nézethez tartozó kontroller akcióban tehát inicializálnunk kell a `@users` példányváltozót, és egyelőre még üres törzzsel definiálnunk kell a `friend` metódust. Mivel az `index` nézetben csak a felhasználó nevét használjuk, ezért csak azt az attribútumot és az `id`-t inicializáljuk a kontrollerben.

```
def index
  @user1 = User.new
  @user1.id = 1
  @user1.username = 'Valaki'
  @user2 = User.new
  @user2.id = 2
  @user2.username = 'Valakimas'
  @users = [ @user1, @user2 ]
end

def friend
end
```

A felhasználói profilon legyen megtekinthető a felhasználó neve, statisztikai adatai és az üzenőfala. Az utóbbi kettőről még nem tudunk semmit, ezért azokat egyelőre hagyjuk üresen.

```
<h1>Profile</h1>
<%= @user.username %>

<h3>Statistics</h3>

<h3>Message board</h3>
```

A `show` nézet számára inicializálnunk kell a `@user` példányváltozót.

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

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

```
<h1>Request forgotten password</h1>

<fieldset>
<legend>Please give your user name</legend>
<%= form_for @user, :url => { :action => 'send_forgotten'
   }, :method => :post do |f| %>
 <div>
   <%= f.label :username %>
   <%= f.text_field :username, :size => 14 %>
 </div>
 <%= f.submit 'Send' %>
<% end %>
</fieldset>

<%= link_to 'Back', :back %>
```

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. Ha a felhasználó meggondolná magát, és megsem kívánná elküldetni magának a jelszavát, egy `Back` feliratú linkkel biztosítjuk számára a lehetőséget az előző oldalra való visszatérésre.

```
<h1>Request forgotten password</h1>

<fieldset>
<legend>Please give your user name</legend>
<%= form_for @user, :url => { :action => 'send_forgotten'
   }, :method => :post do |f| %>
 <div>
   <%= f.label :username %>
   <%= f.text_field :username, :size => 14 %>
 </div>
 <%= f.submit 'Send' %>
<% end %>
</fieldset>

<%= link_to 'Back', :back %>
```

Térjünk át a belépett felhasználó használati eseteire. Módosítsuk az alkalmazás szintű helperünk visszatérési értékét `true`-ra, és rendereljünk a belépett felhasználó menüjébe is használható linkeket. Ilyen például a profil szerkesztése, a személyes statisztika megtekintése, a játékpartner keresése, illetve a kilépés. Ezen funkciókhoz hozzunk létre egy-egy linket a menüben.

A kilépés művelet mutasson a még nem létező `sessions` kontroller `destroy` akciójára, a statisztika pedig a `users` kontroller még nem létező `statistics` akciójára. A felhasználói profil szerkesztéséhez használjuk a `edit` akciót, ennek azonban át kell majd később adnunk a bejelentkezett felhasználó azonosítóját, ami még nem áll a rendelkezésünkre, így e link még inaktív lesz. A felhasználók listája pedig mutasson a `index` nézetre.

```
Welcome, valaki!<br/>
<%= link_to "Profile", '/users/edit/' %> <br/>
<%= link_to "Statistics", '/users/statistics' %> <br/>
<%= link_to "Show users", '/users/' %><br/>
<%= link_to "Logout", '/sessions/destroy' %>
```