HTML és Rails
Gyakorlat

Kovács Gábor

2011. október 12.

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="header">
Header
</div>
<div id="menu">
</div>
<div id="main">
  <%= yield %>
</div>
<div id="footer">
  Copyright, RoR 2011
</div>
```

Második lépésként készítsük el az oldal stíluslapját, amivel ezek a helyükre kerülnek, és helyezzünk el benne minimális mennyiségű formázási információt. Az oldal szélessége legyen 800 képpont. A fejrész legyen világosszürke és 100 képpont magas. A menü a szélesség 24%-át töltse ki, legyen zöld hátterű, 600 képpont magas, balra igazított. Az oldal központi része legyen szintén 600 képpont magas, az oldal szélességének 76%-t töltse ki, világosszürke háttérrel rendelkezzen, és legyen szintén balra igazított. A lábrészben a szöveget igazítsuk középre, és legyen a lábrész magassága 100 képpont.

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

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

#menu {
background-color: #00dd00;
height: 600px;
width: 24%;
float: left;
}

#main {
background-color: #EEEEEE;
height: 600px;
width: 76%;
float: left;
}

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

Kétféle felhasználóra készülünk fel egyelőre, egy látogatóra, és egy belépett felhasználóra, aki korábban kereszülment egy regisztrációs folyamaton. A látogató csak böngészhet, ugyanakkor regisztrálhat. A bejelentkezett felhasználó számára több funkciót is elérhetővé teszük. 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 `_layouts.html.erb`! A formot ágyazzuk egy `fieldset`-be `Login` fejléccel, és tartalmazzon egy felhasználónévre utaló címkét és szövegbeviteli mezőt és egy a jelszóra utaló címkét és jelszóbevi-

teli 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 20 hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé a regisztrációt és az elfelejtett jelszó visszaszerzését, ezt két 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=>'20' %<
      br />
    <%= label_tag 'password', "Password" %><br />
    <%= password_field_tag 'password', '', :size=>'20'
      %><br />
    <%= submit_tag "Login" %>
<% end %>
</fieldset>
<%= link_to "Register", '/users/new' %>
<%= 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 menü `div`-ben a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról.

```
<% if logged_in? %>
Hello user
<% else %>
<%= render 'layouts/loginform' %>
<% 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 show edit forgotten
```

A parancs futtatásával létrejött az `users` kontroller és a hozzá kapcsolódó nézetek köztük az új felhasználó létrehozását lehetővé tevő `new`, a felhaszáló adatait megmutató `show`, a felhasználó profil szerkesztését megvalósító `show`, és a hiányzó jelszót pótló `forgotten` felület, illetve akció.

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 az öt elem rendre a következő: egy húsz karakter széles felhasználónév szövegbeviteli mező a hozzá kapcsolódó címkével, egy email cím szövegbeviteli mező a hozzá kapcsolódó címkével, két darab jelszóbeviteli mező a hozzájuk kapcsolódó címkével, a két jelszómező eltérő azonosítóval rendelezzék, és végül egy nyomógomb. Az oldal alján helyezzünk el egy visszalépés gombot azon felhasználóknak akik véletletül tévedtek ide. A visszalépés Railsben a `:back` URL-lel lehetséges, ami vagy a HTTP fejrészből kivett hivatkozó oldalra mutat, vagy JavaScripttel valósul meg.

```
<h3>Registration</h3>
<%= flash[:notice] %>

<div>
  <fieldset>
    <legend>Register a new user</legend>
```

```
    <%= form_for :user, :url => { :action => "create" }
        do |form| %>
      <div>
        <%= form.label :username %>:<br />
        <%= form.text_field :username, :size=>20 %>
      </div>
      <div>
        <%= form.label :email %>:<br />
        <%= form.text_field :email %>
      </div>
      <div>
        <%= form.label :password %>:<br />
        <%= form.text_field :password, :size=>20 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>:<br />
        <%= form.text_field :password_confirmation, :
            size=>20 %>
      </div>
      <%= submit_tag "Register" %>
    <% end %>
  </fieldset>
</div>
<%= 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 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.

```
  <fieldset>
    <legend>Register a new user</legend>
    <form accept-charset="UTF-8" action="/users/create"
        method="post"><div style="margin:0;padding:0;
        display:inline"><input name="utf8" type="hidden"
        value="&#x2713;" /><input name="
        authenticity_token" type="hidden" value="
        iUr0v03mfCcW+h2t9CRHIPnjc0IYvfmQJiUp/wyG6f4=" />
        </div>
```

```
      <div>
        <label for="user_username">Username</label>:<br
            />
        <input id="user_username" name="user[username]"
            size="20" type="text" />
      </div>
      <div>
        <label for="user_email">Email</label>:<br />
        <input id="user_email" name="user[email]" size=
            "30" type="text" />
      </div>
      <div>
        <label for="user_password">Password</label>:<br
            />
        <input id="user_password" name="user[password]"
            size="20" type="password" />
      </div>
      <div>
        <label for="user_password_confirmation">
          Password confirmation</label>:<br />
        <input id="user_password_confirmation" name="
          user[password_confirmation]" size="20" type=
          "password" />
      </div>
      <input name="commit" type="submit" value="
        Register" />
</form>  </fieldset>
</div>
<a href="javascript:history.back()">Back</a>
```

Az 1. ábra a regisztációs oldal képernyőképét mutatja. Láthatóan sikerült az elrendezést megvalósítanunk, és a formokat létrehozunk.

Az új analógiájára hozzuk létre az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk, csak az email címet tartalmazza, és a címkék különböznek az előző példához képest.

```
<fieldset>
  <legend>Forgotten password</legend>
  Please, give your email address<br />
  <%= form_for :user, :url => {:action => "
      send_forgotten"} do |form| %>
    <div>
```

```
        <%= form.label :email %>:<br />
        <%= form.text_field :email %>
    </div>
  <%= form.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 kilépés vagy az új megjegyzés felvitele. Az első a `users` kontroller `edit` akciójára és nézetére mutat, a második a `sessions` kontroller `destroy` akciójára, a harmadik a `quotes` kontroller `new` akciójára és nézetére. Az utóbbi kettőhöz új modellt és kontrollert is létre kell hoznunk.

```
%= link_to "Profile", url_for(:controller=>"users", :
    action=>"edit") %><br />
<%= link_to "Logout", url_for(:controller=>"sessions",
    :action=>"destroy") %> <br />
<%= link_to "Submit quote", url_for(:controller=>"
    quotes", :action=>"new") %> <br />
```

A profil szerkesztése lényegében megegyezik az új felhasználó létrehozásával, vagyis a regisztrációval a különbség a feldolgozó akcióban és a címkékben áll.

```
<h3>Profile</h3>
<%= flash[:notice] %>

<div>
  <fieldset>
    <legend>Update user profile</legend>
    <%= form_for :user, :url => {:action =>"update"} do
        |form| %>
      <div>
        <%= form.label :username %>:<br />
        <%= form.text_field :username, :size=> 20 %>
      </div>
      <div>
        <%= form.label :email %>:<br />
        <%= form.text_field :email %>
      </div>
```

```
      <div>
        <%= form.label :password %>:<br />
        <%= form.password_field :password, :size=> 20
          %>
      </div>
      <div>
        <%= form.label :password_confirmation %>:<br />
        <%= form.password_field :password_confirmation,
            :size=> 20 %>
      </div>
      <%= submit_tag "Update" %>
    <% end %>
  </fieldset>
</div>
<%= link_to "Back", :back %>
```

Az idézetek link megvalósítása előtt hozzuk létre az idézetek modellt és kontrollert.

```
rails generate model quote
```

Legyen az idézetek tábla a következő. Legyen az idézetnek egy forrása (`source`), annak a személynek a neve, akitő az idézet elhangzott, ez egy string típusú attribútum. Legyen egy string típusú `subject` attribútum, amely az idézethez köthető kurzust írja le. Az idézet (`quote`) lehet hosszú is, ezért ennek a típusa legyen text. Az idézet elhangzásának időpontja `datetime` típusú, legyen az attribútum neve `released_at`. Az idézet átlagos értékelését az `evaluation` attribútum hordozza. Az összes értékelés számát a `votes` attribútum tartalmazza. Ezen kívül adjuk még hozzá a szokásos időpecsétet.

```
class CreateQuotes < ActiveRecord::Migration
  def change
    create_table :quotes do |t|
      t.string :source
      t.string :subject
      t.text :quote
      t.datetime :released_at
      t.integer :evaluation, :limit=>1
      t.integer :votes
      t.timestamps
    end
  end
```

```
end
```

Az ismert módon hajtsuk végre az adatbázis migrációt, és hozzuk létre a kapcsolódó `quotes` kontrollert. A `new` akció az új idézet felvitelére szolgált, az `edit` egy meglévő idézet módosítására, a `comment` egy idézet véleményezésére, a `show` pedig az idézetek megjelenítésére. Törölni változatlanul nem engedünk.

```
rake db:migrate
rails generate controller quotes new edit comment show
```

Új idézet elmentésekor négy attribútumot állíthatunk be. Itt is a `form_for` helpert használjuk. Az első argumentum a `quote` modell, az akció a `create` kontroller metódusra mutat. A forrás és a kurzus egy-egy címkével ellátott szövegbeviteli mező. Az idézet hosszabb is lehet, ezért azt egy címkével ellátott szövegdobozban tesszük megadhatóvá, ami 80 karakter széles és 10 karakter magas. Az elhangzás dátumát a `select_datetime` helperrel tesszük beállíthatóvá. Ennek kezdeti értékét a kontrollerben létrehozott új `Quote` modell példányából vesszük.

```
<fieldset>
  <legend>Submit new quote</legend>
  <%= form_for :quote, :url => {:controller => "quotes"
      , :action => "create"} do |form| %>
  <div>
    <%= form.label :source %>: <br />
    <%= form.text_field :source %>
  </div>
  <div>
    <%= form.label :subject %>: <br />
    <%= form.text_field :subject %>
  </div>
  <div>
    <%= form.label :quote %>: <br />
    <%= form.text_area :quote, :cols=>80, :rows=> 10 %>
  </div>
  <div>
    <%= form.label :released_at %>: <br />
    <%= select_datetime(@quote.released_at, :prefix =>
       "quote") %>
  </div>
  <%= form.submit "Save" %>
     <% end %>
```

```
</fieldset>
```

Módosítanunk kell tehát a `quotes_controller.rb`, ahol a `new` metódusban kezdeti értéket kell rendelnünk egy `Quote` típusú példányváltozó `released_at` attribútumához. Ezután az új idézet bevitele már lehetséges.

```
def new
  @quote=Quote.new
  @quote.released_at=Time.now
end
```

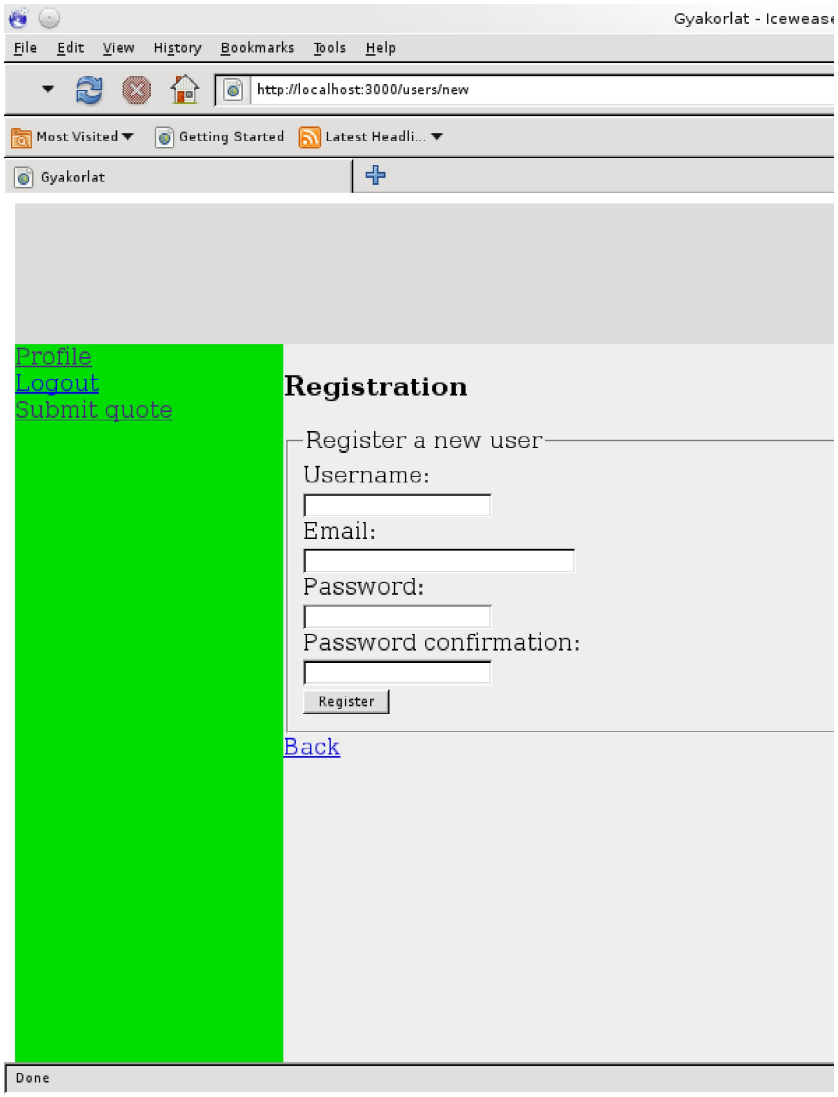
A gyakorlaton megvalósított weboldal kezdőoldala az idézetek gyűjteménye lesz. Készítsük el ennek az elsődleges vázát! Sok idézetet kell megjelenítenünk, amelyeket szintén a kontrollerben inicalizálunk jelenleg. A `show` metódus legyen a következő:

```
  def show
    @quote1=Quote.new
    @quote1.source="Me"
    @quote1.subject="RoR"
    @quote1.released_at=Time.now
    @quote1.quote="Some funny joke"
    @quote1.evaluation=5
    @quote1.votes=1
    @quote2=Quote.new
    @quote2.source="Student"
    @quote2.subject="RoR"
    @quote2.released_at=Time.now
    @quote2.quote="Some less funny joke"
    @quote2.evaluation=4
    @quote2.votes=1
    @quotes = [@quote1, @quote2]
  end
```

Tehát van két idézetünk, amelyet listában vagy táblázatosan jelenítsünk meg. A gyakorlaton a táblázatos megközelítést választottuk az illusztráció végett, azonban ez kevéssé praktikus, módosítani fogjuk. A példában végigiterálunk a `@quotes` tömb minden elemén, és minegyikhez egy új sort hozunk létre a táblázatban. Ezt a `content_tag_for` Rails helperrel valósítottuk meg.

```
<div>
<h3>Funny quote</h3>
<table class="listbox">
```

```
<% for t in @quotes %>
  <%= content_tag_for(:tr,t) do %>
  <td><%= t.subject %></td>
  <td><%= t.source %></td>
  <td><%= t.quote %></td>
  <% end %>
<% end %>
</table>
</div>
```



1. ábra. A létrehozott regisztrációs oldal