Rails routing, formok, helperek, stílusok Gyakorlat

Kovács Gábor

2010. november 13.

A gyakorlat során az előző gyakorlatokon megkezdett feladat megoldását folytatjuk. Először néhány apró kiegészítést végzünk el.

A megoldások modellben (`Solution.rb` fájl) a fájl egyértelműsítése érdekében a fájl elérési útjába belekódoljuk az aktuális felhasználó Neptun-kódját. Minden felhasználó számára létrehozunk egy a Neptun-kódjával megegyező könyvtárat, ha az még nem létezik, és ott tároljuk a saját fajljait. A fájlok elnevezését még annyiban módosítjuk, hogy a nem olvasható karaktereket aláhúzásjellel helyettesítjük.

```
def self.saveFile(user,upload)
  dir = "/usr/src/feladat3/public/data/"+user.neptun
  if !File.exists?(dir)
    Dir.mkdir(dir)
  end
  fname = File.basename(upload['file'].
    original_filename)
  fname.gsub!(/[^\w\.\_]/,'_')
  path = File.join(dir, fname)
  File.open(path,"w") { |f| f.write(upload['file'].
    read) }
end
```

Az útvonalakat tartalmazó `config/routes.rb` fájlban állítsuk át a webszerverünk gyölerét a bejelentkezési képernyő `index` oldalára.

```
root :to => "sessions#index"
```

Ezzel párhuzamosan a `sessions_controller.rb` kilépés akcióját követően a következő oldalnak a `index` nézetet tesszük meg.

```
def destroy
```

```
    reset_session
    redirect_to :controller => "sessions", :action => "
       index"
  end
```

A megoldások kontrollerének (`solutions_controller.rb`) beadás (`create`) akcióját egészítsük ki azzal, hogy beállítjuk a késés (`late`) attribútumot, ha az aktuális feladatbeadás a határidő után történik. Ugyanitt úgy módosítjuk a böngészés folyamatár, hogy a következő oldal az a feladatok index oldala legyen.

```
def create
 solution.late = (Time.now > @task.deadline)
 ...
 redirect_to :controller => "tasks", :action=>"index"
end
```

Az előző gyakorlaton nekiláttunk a `scaffold`-dal létrehozott feladatok kontroller és nézet átformálásához, most ezzel folytatjuk. Először az index oldalt (`tasks/index.html.erb`) formázzuk át úgy, hogy az a felhasználó szerepkörének megfelelő akciókat tegye lehetővé a feladatok listája mellett. A hallgató típusú felhasználó beadhatja a megoldást, és megnézheti a saját megoldását, az oktató típusú felhasználó megnézheti a megoldásokat, illetve szerkesztheti és esetleg törölheti a feladatot, továbbá számára látszik az új feladatot létrehozása link az oldal alján.

```
<h1>Listing tasks</h1>

<table>
  <tr>
    <th>Number</th>
    <th>Url</th>
    <th>Deadline</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @tasks.each do |task| %>
  <tr>
    <td><%= task.number %></td>
    <td><%= link_to task.url, task.url %></td>
    <td><%= task.deadline %></td>
```

```
    <% if @user then if @user.student? %>
    <td><%= link_to "Bead", :controller => "solutions",
        :action => "new", :id => task.number%></td>
    <td><%= link_to 'Mutat', task %></td>
    <% else %>
    <td><%= link_to 'Mutat', task %></td>
    <td><%= link_to 'Szerkeszt', edit_task_path(task) %
       ></td>
    <td><%= link_to 'Töröl', task, :confirm => 'Biztos
       ?', :method => :delete %></td>
    <% end end %>
  </tr>
<% end %>
</table>

<% if @user then if !@user.student? %>
<br />

<%= link_to 'új feladat', :action => "new" %>
<% end end %>
```

A `show` akciót, amely mindkét felhasználó típus számára hozzáférhető, és a hozzá tartozó nézetet számunkra megfelelőbb alakra hozzuk a `tasks_controller.rb`-ben. A `show` akció kikeresi az adatbázisból a megfelelő feladatot, majd session paraméterként beállítja, hogy ez az aktuális oldal. Ez a visszanavigáció során nyújthat segítséget.

```
def show
    @solution = Solution.find_by_user_id_and_task_id
       @user.id, @task.id
    session[:return_to] ||= request.referer
end
```

A visszalépés az alábbi kódrészlettel lehetséges:

```
redirect_to session[:return_to]
```

A feladatok megtekintése nézet (`tasks/show.html.erb`) megjeleníti a feladat tulajdonságait, és ha a felhasználó hallgató típusú, akkor a beadott verzióra vonatkozó információit is. A `late` attribútum boolean típusú, helyette azonban magyar nyelvű szavakat szeretnénk látni, ezért módosítjuk annak alapértelmezett kiírását. Ha van már beadott megoldás, akkor letölthetővé tesszük azt egy link segítségével. Ha a felhasználó oktató típusú, akkor en-

gedélyezzük számára a feladat szerkesztését, vagyis a `scaffold` által a lap alján elhelyezett linket.

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

<p>
  <b>Sorszám:</b>
  <%= @task.number %>
</p>

<p>
  <b>Url:</b>
  <%= link_to @task.url, @task.url %>
</p>

<p>
  <b>Határidő:</b>
  <%= @task.deadline %>
</p>

<% if !@solution.nil? && is_student? %>
<p>
  <b>Verzió</b>
  <%= @solution.version %>
</p>

<p>
  <b>Fájlnév</b>
  <%= link_to @solution.file_name, { :controller => "
      solutions", :action => "download", :id =>
      @solution.id } %>
</p>

<p>
  <b>Beadási idő</b>
  <%= @solution.updated_at %>
</p>

<p>
  <b>Késés</b>
  <% if @solution.late? %>
```

```
  Igen
  <% else %>
  Nem
  <% end %>
</p>

<% end %>

<% if !is_student? %>
<%= link_to 'Szerkeszt', edit_task_path(@task) %> |<%
   end %>
<%= link_to 'Vissza', tasks_path %>
```

A fenti kódrészletben azt, hogy a felhasználó okatató vagy hallgató típusúe egy helper metódussal döntöttük el, amelyet a `tasks_helper.rb` fájlban találunk.

```
  def is_student?
    if @user
      @user.student?
    end
  end
```

A megoldás letöltéséhez egy új, `download` nevű akciót definiálunk a `solutions` kontrollerben. A letöltéshez nincs szükség a `before_filter` `find_user_and_task` nevű szűrőjére, ezért hozzáadjuk a kivételek listájához.

A metódus a `params` hash-en keresztül paraméterezhető, az `id` paraméternek megfelelő megoldást keresi elő a fájlrendszerről, majd a `send_file` metódussal elküldi azt a kliens böngészőnek.

```
before_filter :find_user_and_task, :except => [:
   download]

def download
    solution = Solution.find_by_id params[:id]
    if !solution.nil?
      file = "/usr/src/feladat3/public/data/"+@user.
         neptun+"/"+solution.file_name
      send_file file
    end
  end
```

Az oktató típusú felhasználó részéről igény, hogy a beadott feladatokat letölteni és kommentezni tudja. Az értékelés felhasználási eset a feladatok megtekintése nézetből indul ki. Az oldal alján elhelyezünk egy csak oktatóknak szóló listát, amelyen kereszül a feladatra megoldást beadó hallgatók megoldásai közvetlenül elérhetők.

```
<% if !is_student? %>
  <p><b>Értékel</b></p>
  <ul>
  <% @task.solutions.each do |solution| %>
    <% user = solution.user %>
    <li><%= link_to user.neptun, :controller => "
        solutions", :action => "edit", :id => solution.
        id %></li>
  <% end %>
  </ul>
<% end %>
```

Mivel a `Solution` modell nem teszi lehetővé a megoldások kommentelését, ki kell az egészítenünk egy új, komment attribútummal, vagyik migrálnunk kell a modellt (`db/migrate/AddCommentToSolution`).

```
def self.up
  add_column :solutions, :comment, :text
end

def self.down
  remove_column :solutions, :comment
end
```

A komment hozzáfűzés megvalósításához a megoldás kontrollert ki kell egészítenünk egy szerkesztés és egy módosítás akcióval. A szerkesztés (`edit`) akcióhoz egy form fog tartozni a nézeten, amely eseményét a módosítás (`update`) akció fogja lekezelni. A felhasználók és feladatok kikeresését most nem a `find_user_and_task` metódussal végezzük el, ezért a `before_filter` szűrőhöz hozzáadunk két kivételt.

A `edit` akció közvetlenül elérhetővé tesz három példányváltozót a nézet számára, az aktuális megoldás, a megoldást beadó felhasználó és annak a feladatnak az azonosítóját, amelyhez a feladat tartozik. Az elsőt az `id` paraméter alapján keressük ki az adatbázisból, az utóbbi kettőt a `Solution` osztály `belongs_to` metódusán keresztül határozzuk meg.

A szerkesztés nézete (`solutions/edit.html.erb`) kiírja, hogy mely hallgató típusú felhasználó mely feladatra beadott megoldását kommenteli az

oktató típusú felhasználó. Az oldal egy linken keresztül letölthetővé teszi a fájlt, és egy szövegdoboz típusú beviteli mezőt, valamint egy nyomógombot tartalmazó formmal kommentelhetővé teszi azt.

```
<p><b><%= @user.neptun %> megoldása a <%= @task.number
    %>. feladatra</b></p>

<%= flash[:notice] %>
<p>
  <b>Fájl</b>
  <%= link_to @solution.file_name, :action=>"download",
      :id=>@solution.id %>
</p>

<% form_for :solution, :url => {:action => "update", :
    id => @solution.id} do |form| %>
  <p>
  <b><%= form.label :comment %></b><br />
  <%= form.text_area :comment, :cols => 40, :rows =>
     10, :body => @solution.comment %>
  </p>
  <%= submit_tag "Elküld" %>
<% end %>
```

A `update` akció a nézet formját kezeli le. Az `id` paraméter alapján kikeresi az adatbázisból azt a megoldást, amelyet frissíteni szándékozik a felhasználó, a `solution` paraméter hash-éből elővesszük a `comment` kulcshoz tartozó értéket, és azt az azonos nevű attribútumhoz rendeljük. A sikeres mentés után a feladatok listájára irányítjuk át a felhasználót

```
  before_filter :find_user_and_task, :except => [:edit,
      :update, :download]

  def edit
    @solution = Solution.find_by_id params[:id] if
       params[:id]
    @user = @solution.user
    @task = @solution.task
  end

  def update
    @solution = Solution.find_by_id params[:id]
```

```
    s = params[:solution]
    @solution.comment= s['comment']
    @solution.save
      #task = @solution.task
      flash[:notice] = "Sikeres frissítés"
      redirect_to :controller=>"tasks", :id=>@solution.
          task.id
  end
```

Most, hogy már vannak elérhető a kommentelés funkció, megjeleníthetővé kell tennünk a megjegyzéseket a hallgató típusú felhasználók számára a feladatok kontroller mutat nézetében, ha már tartozik a feladathoz komment.

```
<%if has_comments? %>
<p>
  <b>Megjegyzés</b>
  <%= @solution.comment %>
</p>
<%end%>
```

Az, hogy a feladathoz tartozik-e komment a `has_comments?` helper dönti el, amely szintén a `tasks_helper.rb` fájlban található.

```
  def has_comments?
    @solution.comment?
  end
```

Az alkalmazásunk megjelenését stíluslapok segítségével formálhatjuk, amelyet alapértelmezés szerint alkalmazás szinten linkelünk be a `layouts/application.html.erb` fájlban található `stylesheet_link_tag` metódussal. Módosítsuk az alkalmazásunk menüjét, és rendeljünk hozzá egyedi stílust!

```
<div id="menu">
  <table id="menu">
  <tr>
  <% if logged_in? %>
      <td>
      <%= link_to "Beállítások", :controller => "users",
          :action => "show", :id => @current_user.id %>
      </td>
      <td>
      <%= link_to "Kijelentkezés", :controller => "
         sessions", :action => "destroy" %>
```

```
      </td>
      <td>
      <%= link_to "Feladatok", :controller => "tasks", :
          action => "index" %>
      </td>
  <% else %>
      <td>
      <%= link_to "Regisztráció", :controller => "users"
          , :action => "new"%>
      </td>
      <td>
      <%= link_to "Bejelentkezés", :controller => "
          sessions", :action => "new" %>
      </td>
  <% end %>
  </tr>
  </table>
</div>
```

Egy új, saját stílusfájlt hozunk létre a webszerver könyvtárában (`public/stylesheet/my.css`), amely a menü számára tartalmaz néhány formázást. Definiálja

- a menü szélességét,
- a menüt tartalmazó tábla formátumát,
- a tábla celláinak betűtípusát, méretét, térközét
- a tábla cellája feletti kurzor eseményét
- és a tábla cellája által tartalmazott linkek stílusát.

```
#menu {
  width:300px;
  color:blue;
}
#menu table {
  margin: 0 0 0 0;
  padding: 0 0 0 0;
  border-collapse: collapse;
}
#menu table td {
```

```
  font: 10pt sans-serif;
  border-right: 1px solid yellow
  font-weight: bold
  height: 20px
  width: 120px
  padding: 3px 20px 2px 20px
}
#menu td:hover {
  background: yellow
}
#menu table td a {
  text-decoration:none;
  color:blue;
}
```