HTML és Rails
Segédlet

Kovács Gábor

2011. március 14.

A segédlet a 2011. március 14-én elmaradt gyakorlat helyett nyújt segítséget a következő feladat megoldásában. A segédlet a HTML keret és az egyes nézetek Rails helperekkel való létrehozását és stíluslapok használatát tárgyalja.

Az előző gyakorlatok során egyetlen modellt hoztunk létre a `Task` osztályt, most az új feladat létrehozására, a feladat szerkesztésére és a feladatok listázására hozunk létre nézeteket. A többi nézetet egyelőre csak keret szinten definiáljuk.

A `Task` osztály jelenleg három attribútummal rendelkezik: az egész típusú sorszám (`number`), a `string típusú \verburl!` és a `datetime` típusú határidő (`deadline`). Ezek lesznet tehát azok az adatok, amelyeket meg kell jelenítenünk, illetve szerkeszthetővé kell tennünk.

A feladatokat megjelenítő nézeteket és a hozzá kapcsolódó kontrollereket a `rails generate controller task new show edit` paranccsal hozzuk létre, amely a nézetek könyvtárában (`app/views/task`) létrehozza a `new`, a `show` és a `edit` oldalakat, továbbá a kapcsolódó kontroller osztályban üres törzsű metódusokat definiál, létrehoz egy helper modult és két teszt osztályt.

A kontrollerben (`task_controller.rb`) egyelőre egyedileg létrehozott `Task` példányokkal ellenőrizzük a nézetek megjelenését, nem használjuk ki az adatbázis hozzáférést. Hozzáadunk egy `create` és egy `update` metódust, amely az új feladatot létrehozó, illetve a feladatot módosító form eseményeket kezeli le, egyelőre üres törzssel definiáljul.

```
class TaskController < ApplicationController
  def new
  end

  def create
  end
```

```
  def show
    @task2 = Task.new :url=>'https://twiki.db.bme.hu/
        twiki'+
          '/bin/view/Student/Ruby/RubyElsoFeladat20111',
          :deadline=>DateTime.parse('2011-02-28 23:59'),
          :number=>1
    @task1 = Task.new :url=>'https://twiki.db.bme.hu/
        twiki'+
          '/bin/view/Student/Ruby/
             RubyMasodikFeladat20111',
          :deadline=>DateTime.parse('2011-03-14 23:59'),
          :number=>2
    @tasks = [@task1, @task2]
  end

  def edit
    @task = Task.new :url=>'https://twiki.db.bme.hu/
        twiki'+
          '/bin/view/Student/Ruby/RubyElsoFeladat20111',
          :deadline=>DateTime.parse('2011-02-28 23:59'),
          :number=>1
  end

  def update
  end
end
```

A `show` nézet a feladatok listáját jeleníti meg. Az elérhető feladatok, vagyis a kontroller azonos nevű metódusában `@tasks` néven elérhetővé tett lista feladatai sorszámainak listáját egy táblázatban jelenítjük meg a `content_tag_for` helper segítségével. A sorszámok linkek, amelyek a feladatot szerkesztő oldalra mutatnak. Az oldal alján elérhetővé tesszük az új feladatot létrehozó oldalra való navigációt.

```
<div>
<h1>Feladatok listája</h1>
<table class="listbox">
<% for t in @tasks %>
  <%= content_tag_for(:tr,t) do %>
  <td>Sorszám</td>
  <td><%= link_to t.number, :action=>"edit", :id=>t.
```

```
      number %></td>
  <% end %>
<% end %>
</table>

  <div class="newbox">
    <%= link_to "Új feladat", :action=>"new" %>
  </div>
</div>
```

A `edit` nézetre a `show` nézet sorszámmal rendelkező linkjére kattintva juthatunk át. Az oldal egy táblázatba elhelyezett formot tartalmaz, amely a ki nem fejtett `update` akcióra mutat A sorszám az `1..6` tartományból választ értéket, és azt a `task[number]` névhez köti. Az URL egy szövegbeviteli mező, amely a `task[url]` névhez rendel értéket. A határidő egy dátumválasztó HTML select-ekből álló csoport, amely `task[..]` prefixszel és `year`, `month`, `day`, `hour`, illetve `minute` kulccsal rendelkezik. Az oldal a kezdeti értékeket a kontroller azonos nevű metódusában definiált `@task` példányváltotóból veszi.

```
<div>
<h1>Feladat szerkesztése</h1>
<form action="task/update" method="post">
  <table class="editbox">
    <tr>
      <td>Sorszám</td>
      <td><%= select("task","number", 1..6) %></td></tr
          >
    <tr>
      <td>URL</td>
      <td><input type="text" name="task[url]" size="60"
          value="<%= @task.url %>" /></td></tr>
    <tr>
      <td>Határidő</td>
      <td><%= select_datetime(@task.deadline,:prefix=>"
          task") %></td>
    </tr>
    <tr>
      <td><%= submit_tag "Mentés" %></td><td></td>
    </tr>
  </table>
</form>
</div>
```

Az új feladatot létrehozó `new` nézet hasonló a `show` nézetet azzal a különbséggel, hogy itt nincsen a kezdeti értéket megadó példányváltotónk, és a formot lekezelő kontroller akció a `create` lesz.

```
<div>
<h1>Új feladat</h1>
<form action="create" method="post">
  <table class="editbox">
    <tr>
      <td>Sorszám</td>
      <td><%= select("task","number", 1..6) %></td></tr
          >
    <tr>
      <td>URL</td>
      <td><input type="text" size="60" name="task[url]"
          /></td></tr>
    <tr>
      <td>Határidő</td>
      <td><%= select_datetime(Time.now, :prefix=>"task"
          ) %></td>
    </tr>
    <tr>
      <td><%= submit_tag "Mentés" %></td><td></td>
    </tr>
  </table>
</form>
</div>
```

Az alkalmazásunk alapértelmezett keretét kiegészítjük egy menüvel, amely lehetővé teszi a bejelentkezett felhasználó számára a legfőbb funkciók közötti navigálást, illetve a be nem jelentkezett felhasználóknak a regisztrációt és a bejelentkezést.

Azt, hogy a felhasználó be van-e jelentkezve a `logged_in?` helper metódussal állapítjuk meg, amelyet az alkalmazásszintű segédmetódusok modulját, az `application_helper.rb`-ben helyezünk el. Mivel a felhasználókezelést egyelőre nem valósítottuk meg, ez térjen vissza mindig ugyanazzal az értékkel.

```
def logged_in?
  true
end
```

A menüt az alkalmazásunk HTML fájljába helyezzük el, az `application.html.erb`-ben. Azon linkeket, amelyhez rendelkezésünkre áll már a kontoller, definiálju a `link_to` Rails helperrel, a többit egyelőre szövegként meghagyjuk. Mivel ékezetes karaktereket használunk a Ruby forrásban, nem feledkezhetünk meg a karakterkódolás deklarációjáról!

```
<div id="menu">
  <table id="menu">
  <tr>
  <% if logged_in? %>
      <td>
      Beállítások
      </td>
      <td>
      Kijelentkezés
      </td>
      <td>
      <%= link_to "Feladatok", :controller => "task", :
          action => "show" %>
      </td>
  <% else %>
      <td>
      Regisztráció
      </td>
      <td>
      Bejelentkezés
      </td>
  <% end %>
  </tr>
  </table>
</div>
```

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!

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
- a feladat módosító form stílusát
- a feladatok listája nézet táblázata stílusát
- az új feladat link térközét

```
#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;
}
.editbox {
   border: 2px solid blue;
}
.listbox {
   border: 2px solid blue;
```

```
}
.listbox tr td {
  border: 1px solid black;
}
.newbox {
  padding-top: 5px;
}
```