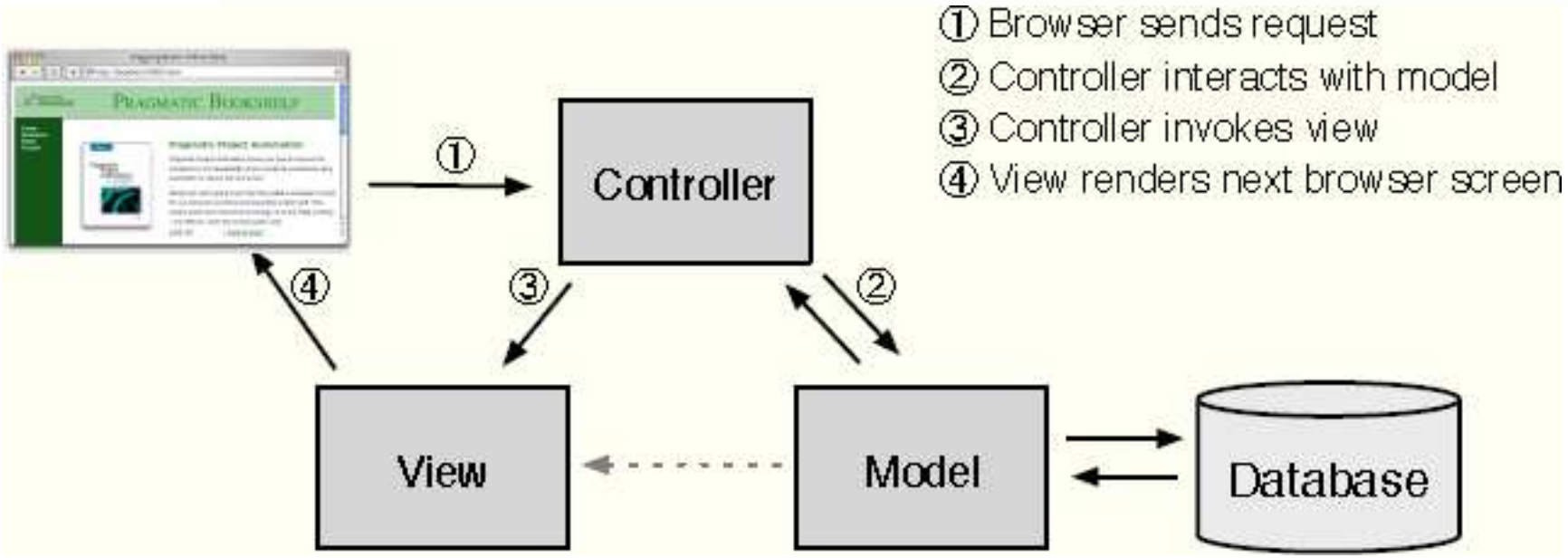
HTML és Rails

Kovács Gábor

kovacsg@tmit.bme.hu

BME-TMIT

MVC


① Browser sends request
② Controller interacts with model
③ Controller invokes view
④ View renders next browser screen
Controller
View
Model
Database
①
②
③
④

A Rails fő HTML dokumentuma

```
<!DOCTYPE html>
<html>
  <head>
    <title>Gyakorlat</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>

    <%= stylesheet_link_tag "application", media: "all", 'data-turbolinks-track': 'reload' %>
    <%= javascript_pack_tag "application", 'data-turbolinks-track': 'reload' %>
  </head>

  <body>
    <%= yield %>
  </body>
</html>
```

HTML bevezetés

- HyperText Markup Language
- HTML dokumentumok alkalmasak weboldalak tartalma leírására
- A HTML dokumentum elemekből (vagy címkékből) áll
- A tartalommal rendelkező HTML elemekbe más HTML elemek ágyazhatók
- A legtöbb HTML elemhez attribútumok, egyedi azonosító, illetve osztály(ok) rendelhető(k)

HTML szintaxis

- Egy HTML elem általánosan egy nyitó HTML tagből (például `<div>`), tartalomból és egy záró HTML tagből (például `</div>`) áll
- Egyes HTML elemek nem rendelkeznek tartalommal, a nyitó tag lezárja őket (például `<br />`)
- A legtöbb HTML elem attribútumokkal rendelkezhet, melyek a nyitó tagben jelennek meg egymástól szóközzel elválasztva.
- A HTML tag attribútum formálisan egy névből, egy egyenlőségjelből és egy idézőjelek közötti szövegből áll (például `<div class="box" id="box">`), az idézőjelek közötti szöveg nem tartalmazhat HTML tag-eket
- HTML tag betűmérete nem számít, és tag nem kezdődhet szóközzel
- A speciális karakterek HTML kódoltan írandók (például `&eacute;`)
- Megjegyzéseket a `<!--` és `-->` jelek között tehetünk

HTML dokumentumok struktúrája

```
<!DOCTYPE html>
```

- A HTML dokumentum verziója és szintakitkai szabályainak azonosítása a böngésző számára

```
<html>
```

- A HTML dokumentum nyitó tage,

```
<head>...</head>
```

- A HTML dokumentum fejrésze, amely a többek között oldal címét, metaadatokat (például nyelv, karakterkódolás), az oldalon használt stílusok és szkriptek definícióját vagy elérhetőségét tartalmazza

```
<title>Gyakorlat</title>
```

- A HTML dokumentum címe, ez lesz a böngészőablak neve

```
<body>...</body>
```

- A HTML dokumentum törzse, ezt a tartalmat jeleníti meg a böngésző

```
</html>
```

- A HTML dokumentum záró tage

Nézetek létrehozása

- Kézzel: létrehozzuk a fájlt
- Automatikusan generálva

```
kovacsg@debian:~/gyakorlat> rails g controller say hello
      create  app/controllers/say_controller.rb
       route  get 'say/hello'
      invoke  erb
      create    app/views/say
      create    app/views/say/hello.html.erb
      invoke  test_unit
      create    test/controllers/say_controller_test.rb
      invoke  helper
      create    app/helpers/say_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    scss
      create      app/assets/stylesheets/say.scss
```

Nézet fájlok

- `erb` kiterjesztés: Embedded Ruby
- HTML fájl Ruby kódrészletekkel: `hello.html.erb`
- További reprezentációk: `hello.js.erb`, `hello.xml.erb`, `hello.json.erb`
- Az `I18n.locale = :hu` hatására a megjelenítendő template-ek `html.erb` helyett `hu.html.erb` végű fájlok: `hello.hu.html.erb`

Ruby kód beágyazása, ERB

- Scriptletekkel és helper metódusokkal így a designer csak HTML tageket látja
- Egyetlen Ruby kifejezés beágyazása:
  `<title>Gyakorlat <%= Time.now %></title>`
- Ruby kódrészlet beágyazása: `<% %>`
- Komment: `<%# %>`
- A HTML forrás is elrejthető Ruby helper metódussal:

```
content_tag :p, "Hello, world"
```

Helper modulok

- Gyakran ismétlődő HTML kódrészleteket generáló API modulok
- A helper modulok összes függvénye felhasználható a nézetekben
- Saját helper modul generálódik minden kontrollerhez, amely csak annak a kontrollernek a nézeteire érvényes
- Kontrollerben felhasználható
  - `include MyHelper`
  - osztálymetódusként: `MyHelper.m`

Egybetűs helperek

- Többnyelvűsítés (I18n) helperei
  - `I18n.translate`, `t`: fordítás szótárral
  - `I18n.localize`, `l`: dátum, pénznem
- HTML escape-elése, beágyazása
  - escape-elés `<%=h @value %>`
  - beágyazás `<%= raw @value %>`

HTML elemek helperei – link

- A link `<a>..</a>` segítségével egy másik dokumentumra vagy az aktuális dokumentumok belül egy másik pozícióra ugorhatunk
- `<a href="/say/hello">Hello, world!</a>`, a `href` utáni URI-ra mutat
- `<a name="sayhello">Hello, world!</a>`, a `name` egy az oldalon belüli pozíciót definiál, amelyre a `href` attribútummal ugorhatunk, az oldalon belüli pozíciót az URI `#` utáni részében adjuk meg
- Link előállítása Rails-szel a `<a href="/say/hello">Hello, world!</a>`
  - Linkkel: `<%= link_to "Hello, world!", :controller=>"say", :action=>"hello"%>`
  - Nyomógombbal: `<%= button_to "Hello, world!", :controller=>"say", :action=>"hello"%>`
  - Az utóbbi egy HTML form-ot készít, lásd később
- A mailto protokoll azonosító az URI-ban navigáció helyett a levelés küldését kezdeményezi.

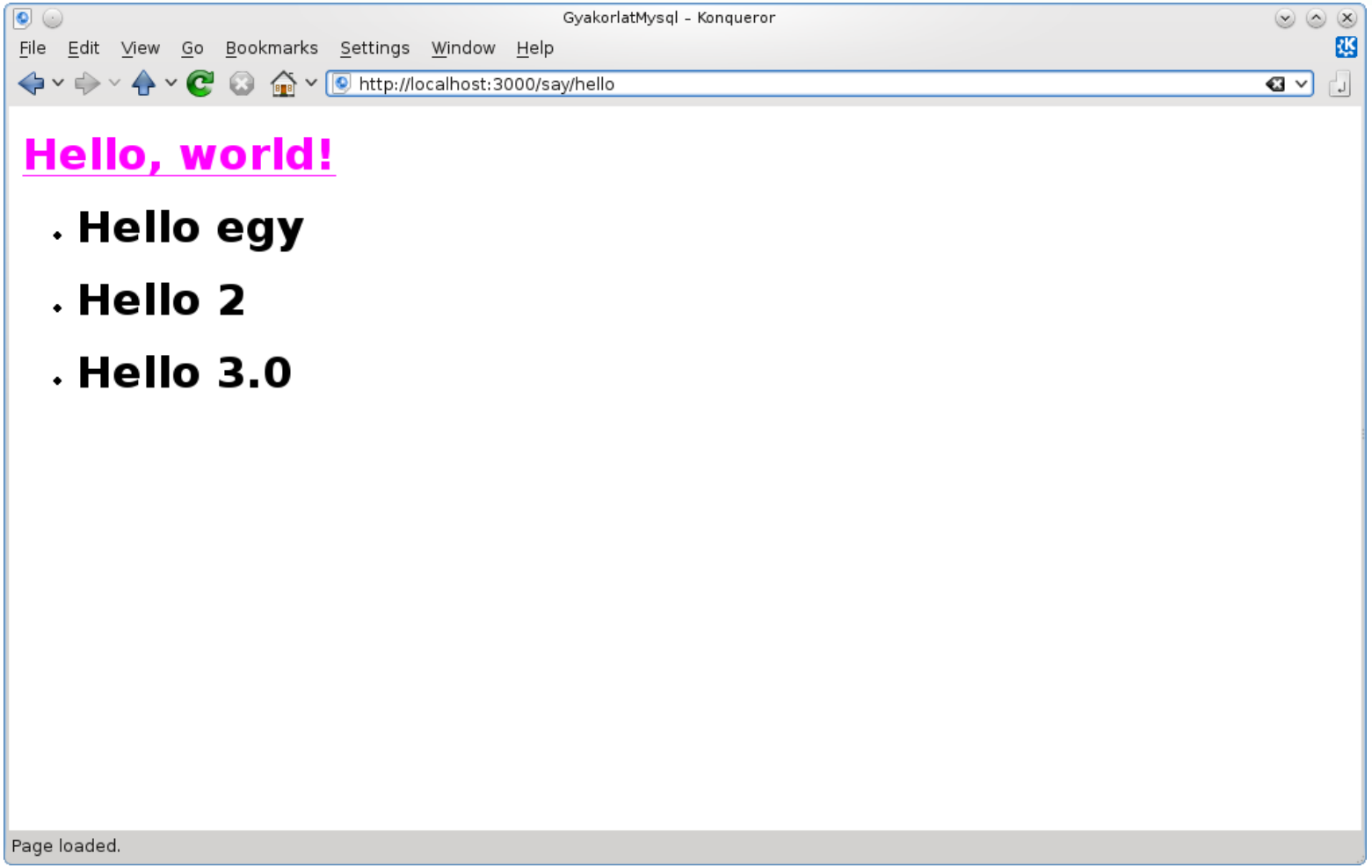
```
<%= mail_to "kovacsg@tmit.bme.hu", Kovács Gábor %>
<a href="mailto:kovacsg@tmit.bme.hu">Kovács Gábor</a>
```

HTML elemek helperei – listák

- HTML-ben a `<ul>..</ul>` elemmel felsorolást, az `<ol>..</ol>` elemmel számozott listát definiálhatunk. A lista elemeit az `<li>..</li>` elem tartalmában adjuk meg.
- Listák egymásba ágyazhatók, illetve tetszőleges más HTML elemet tartalmazhatnak
- Rails-ben a lista elemeit jellemzően egy `for` ciklussal íratjuk ki

```
<ul>
<% for i in [egy, 2, 3.0] %>
<li><h1>Hello <%= i%></h1></li>
<% end %>
</ul>
```

HTML elemek helperei – link és lista példa



HTML elemek helperei – táblázat

- HTML-ben táblázatot a `<table>..</table>` elemmel definiálhatunk.
- A táblázat `<tr>..</tr>` elemmel definiált sorokból áll.
- A táblázat egy sora `<td>..</td>` elemmel definiált cellákból áll, a cellák tartalma tetszőleges.
- Ezen kívül értelmezett még a `<thead>..</thead>`, a `<tbody>..</tbody>` és a `<th>..</th>` elem.
- Rails-ben modell példányok megjelenítése:

```
<%= content_tag_for(:tr, @person) do %>
  <td><%=h @person.first_name %></td>
  <td><%=h @person.last_name %></td>
<% end %>

<tr id="person_123" class="person">....</tr>
```

HTML elemek helperei – objektum, kép

- A HTML `<object>..</object>` elemmel kép, hang, videó, flash stb. objektumok ágyazhatók be az oldalba.
- A `data` attribútum az objektum URI-ját, a `type` attribútum a beágyazott objektum MIME típusát, a `width`, illerve a `height` attribútum az objektum méretét adja meg
- Speciálisan képekre az `<img>..</img>` elem is használható, ahol a forrást a `src` attribútum adja meg, és alternatív szöveg jeleníthető meg az `alt` attribútum értékével, ha a kép nem lenne elérhető
- Példa:

```
<img src="kep.jpg" alt="szoveg" width="640" height="480" />
```

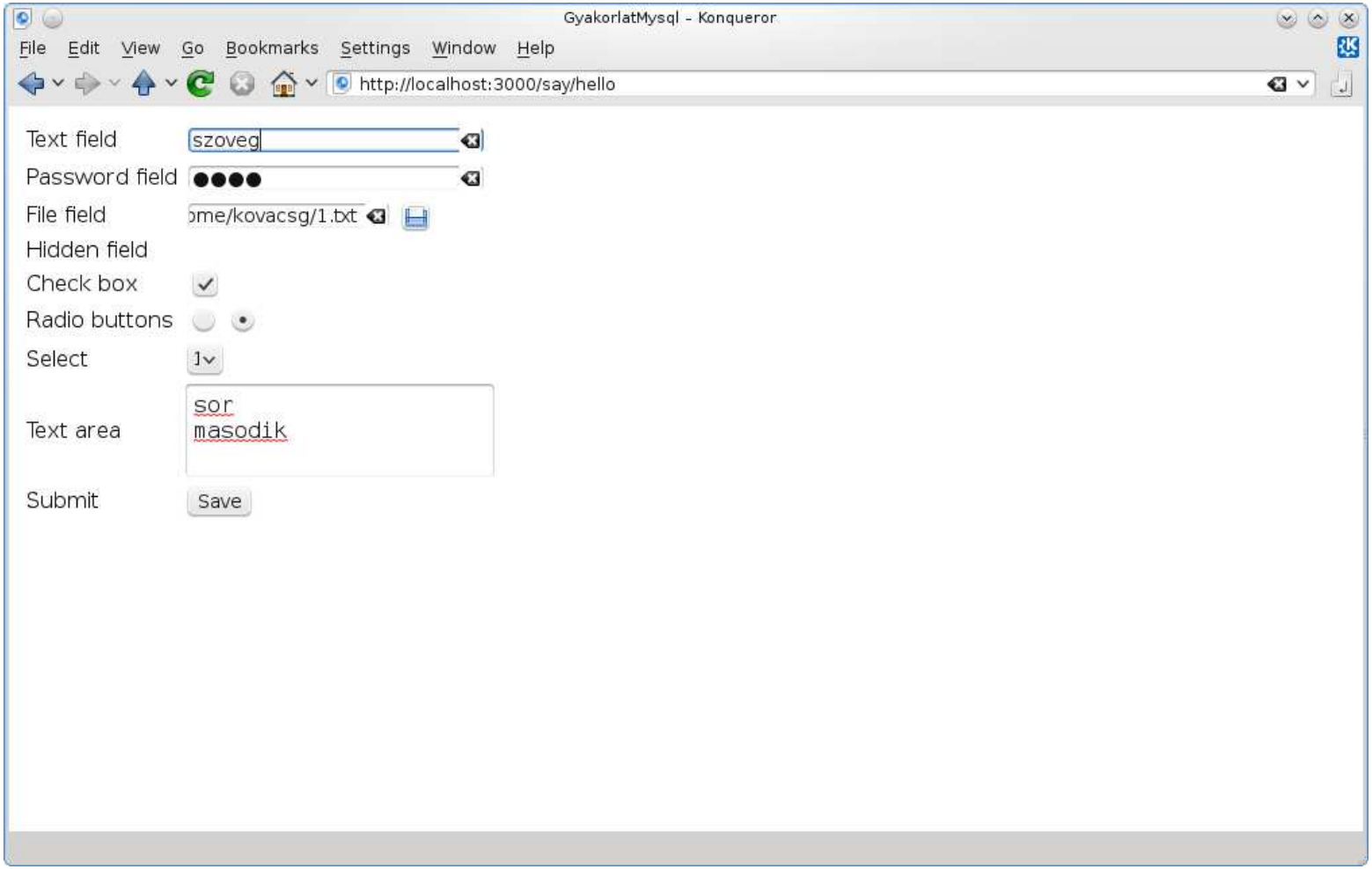
- Lásd még `<embed>..</embed>` elem

HTML elemek helperei – form

- HTML `<form>..</form>` elemmel adatok küldhetők a szerver számára.
- A form tetszőleges HTML elemeket tartalmazhat, közöttük `<input type="..." />` elemeket.
- A form kötelező attribútuma a `action`, ami az adatokat feldolgozó URI-ra mutat
- A feldolgozó kontrollerben a `name` attribútummal hivatkozhatunk a mező értékére, amelyet a `value` attribútummal inicializálhatunk
- Opcionális az `method` attribútum, amely az adatküldés HTTP metódusa adható meg. Ennek a RESTful HTTP szempontjából van jelentősége.
- Rails-szel:

```
<%= form_for @task, :url=>{:action=>"hello", :controller=>"say"} do |t| %>
<%= t.label :number %>: <%= t.text_field :number %><br />
<%= t.label :url %>: <%=t.text_field :url%><br />
<%= t.submit %>
<% end %>
```

HTML elemek helperei – fontosabb form input típusok



HTML elemek helperei – form input típusok 1

- Beviteli mező címkéje
  - Rails: `t.label :number`
  - HTML: `<label for="task_number">Number</label>`
- Szövegmező, `<input type="text" />`
  - Rails: `t.text_field :number`
  - HTML:

```
<input id="task_number" name="task[number]" size="30" type="text" />
```

- Jelszómező, `<input type="password" />`
  - Rails: `t.password_field :login, :pass, :size=>20`
  - HTML:

```
<input type="password" id="login_pass" name="login[pass]" size="20" value="#{@login.pass}" />
```

HTML elemek helperei – form input típusok 2

- Szövegdoboz, `<textarea />`
  - Rails: `t.text_area :comment, :cols=>20, :rows => 3`
  - HTML:

```
<textarea cols="20" id="task_comment" name="task[comment]" rows="3" />
```

- Fájlkiválasztó mező, `<input type="file" />`
  - Rails: `t.file :data`
  - HTML:

```
<input id="task_data" name="task[data]" type="file" />
```

- Rejtett mező, nem jelenik meg a felületen, `<input type="hidden" />`
  - Rails: `t.hidden_field :url, value=>"http://"`
  - HTML:

```
<input id="task_url" name="task[url]" type="hidden" value="http://" />
```

HTML elemek helperei form input típusok 3

- Jelölőnégyet, `<input type="checkbox" />`
  - Rails: `t.check_box :late`
  - HTML:

```
<input id="task_late" name="task[late]" type="checkbox" value="1" />
```

- Nyomógomb, `<input type="submit" />`
  - Rails: `t.submit "Save"`
  - HTML:

```
<input id="task_submit" name="commit" type="submit" value="Save" />
```

- Rádiógomb, véges elemszámú halmazból választás, `<input type="radio" />`
  - Rails: `t.radio_button :late, "yes"`
  - HTML:

```
<input id="task_late_yes" name="task[late]" type="radio" value="yes" />
```

HTML elemek helperei – form input típusok 4

- Választómező, véges elemszámú halmazból választás, `<select>..</select>`
- A választható alternatívák `<option>..</option>` elemként felsorolva
- Az opciók `<optgroup>..</optgroup>` elemekkel csoportosíthatók
  - Rails:

```
t.select :number, Task.all.collect {|task| [task.number]}, {:include_blank=>true}
```

  - HTML:

```
<select id="task_number" name="task[number]">
<option value=""></option>
<option value="1">1</option>
</select>
```

HTML elemek helperei – form input típusok 5

- Dátum- és időválasztómező
  - Év, hónap, nap: `date_select`
  - Év, hónap, nap, óra, perc: `datetime_select`
  - Idővel paraméterezhető mezők: `select_date`, `select_datetime`
- Országválasztómező: `country_select`
- Szabad opciólista: `options_for_select ["Egy", "Ketto"]`

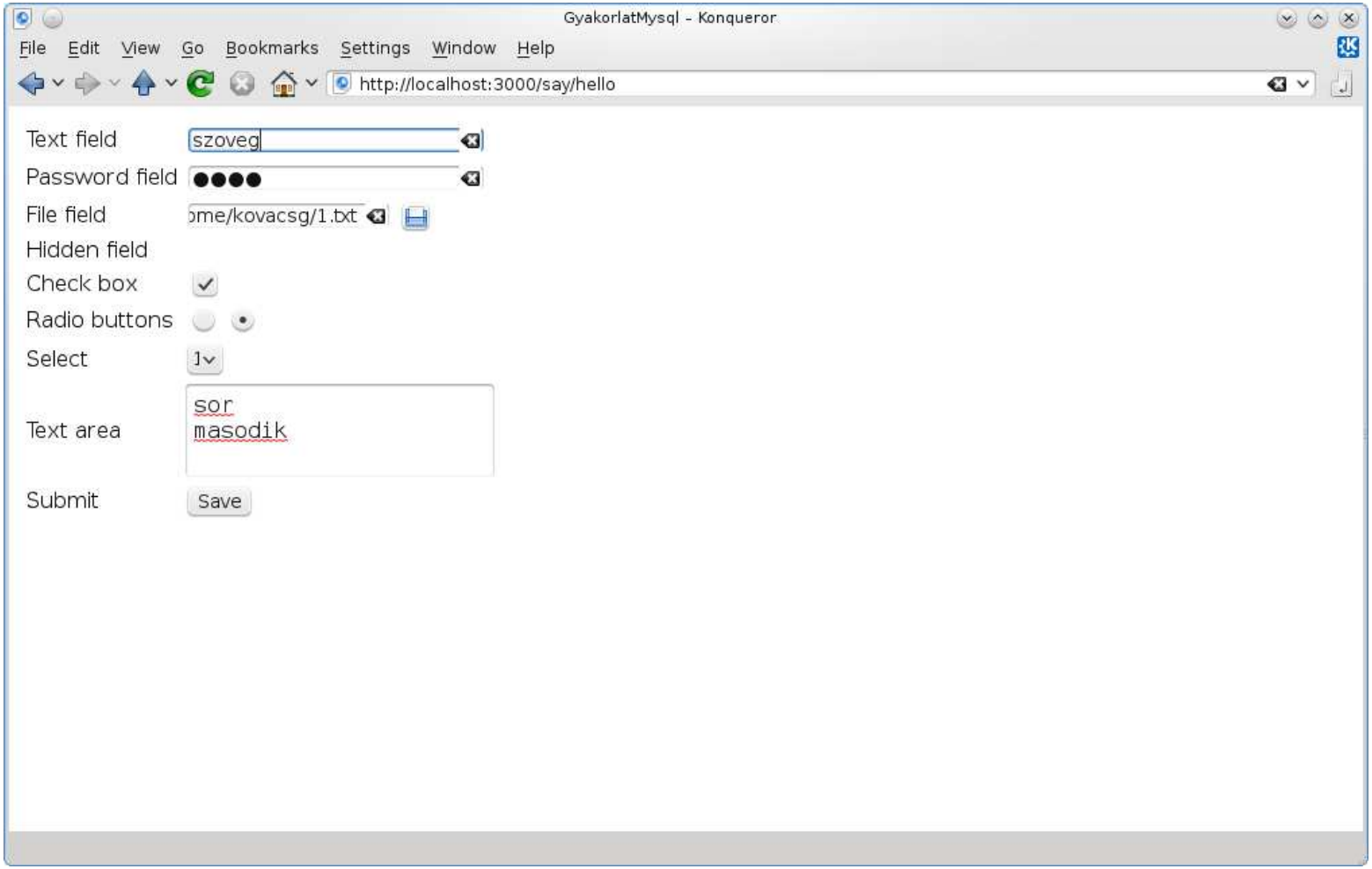
HTML elemek helperei – form példa 1

```
<%= form_for @task, :url => url_for(:controller =>"say", :action =>"hello") do |t| %>
<table>
<tr><td><%= t.label "text field" %></td><td><%= t.text_field :number %></td></tr>
<tr><td><%= t.label "password field" %></td><td><%= t.password_field :pass %></td></tr>
<tr><td><%= t.label "file field" %></td><td><%= t.file_field :file %></td></tr>
<tr><td><%= t.label "hidden field" %></td><td><%= t.hidden_field :val, :value=>"http://" %></td></tr>
<tr><td><%= t.label "check box" %></td><td><%= t.check_box :late %></td></tr>
<tr><td><%= t.label "radio buttons" %></td><td><%= t.radio_button  :setlate, "yes" %>
        <%= t.radio_button :setlate, "no" %></td></tr>
<tr><td><%= t.label "select" %></td><td>
        <%= t.select :num, Task.all.collect {|task| [task.number]}, {:include_blank=>true} %></td></tr>
<tr><td><%= t.label "text area" %></td><td><%= t.text_area :comment, :cols=>20, :rows=>3 %></td></tr>
<tr><td><%= t.label "submit" %></td><td><%= t.submit "Save" %></td></tr>
</table>
<% end %>
```

HTML elemek helperei – form példa 2

```
<form accept-charset="UTF-8" action="/say/hello" class="new_task" id="new_task" method="post">
<div style="margin:0;padding:0;display:inline"><input name="utf8" type="hidden" value="&#x2713;" />
<input name="authenticity_token" type="hidden" value="7wZHzwk2vc8L5WHQ5HtrKgxDoodgje77lg1qvC0dhfM=" />
</div>
<table>
<tr><td><label for="task_text field">Text field</label></td>
    <td><input id="task_number" name="task[number]" size="30" type="text" /></td></tr>
<tr><td><label for="task_password field">Password field</label></td>
    <td><input id="task_pass" name="task[pass]" size="30" type="password" /></td></tr>
<tr><td><label for="task_file field">File field</label></td>
    <td><input id="task_file" name="task[file]" type="file" /></td></tr>
<tr><td><label for="task_hidden field">Hidden field</label></td>
    <td><input id="task_val" name="task[val]" type="hidden" value="http://" /></td></tr>
<tr><td><label for="task_check box">Check box</label></td>
    <td><input name="task[late]" type="hidden" value="0" />
        <input id="task_late" name="task[late]" type="checkbox" value="1" /></td></tr>
<tr><td><label for="task_radio buttons">Radio buttons</label></td>
    <td><input id="task_setlate_yes" name="task[setlate]" type="radio" value="yes" />
        <input id="task_setlate_no" name="task[setlate]" type="radio" value="no" /></td></tr>
<tr><td><label for="task_select">Select</label></td>
    <td><select id="task_num" name="task[num]">
        <option value=""></option>
        <option value="1">1</option>
        </select></td></tr>
<tr><td><label for="task_text area">Text area</label></td>
    <td><textarea cols="20" id="task_comment" name="task[comment]" rows="3"></textarea></td></tr>
<tr><td><label for="task_submit">Submit</label></td>
    <td><input id="task_submit" name="commit" type="submit" value="Save" /></td></tr>
</form>
```

HTML elemek helperei – form példa 3



Oldalak elrendezése Rails-szel

- Az oldal alapértelmezett struktúráját a `layouts/application.html.erb` definiálja
- Ez nézetenként a kontrollerben felüldefiniálható: `layout 'main'`
- Töredék megjelenítése: A `render :menu` a `_menu.html.erb` tartalmát ágyazza be
- A `render @user` a `@user` objektum típusának megfelelő, `_user.html.erb` nevű nézetet ágyazza be

Oldalak elrendezése HTML-lel

- Két stratégia: rugalmas `div` és gyors `table`
- A `div` elem osztásokat jelöl ki a HTML dokumentumon belül, amelyhez stíluslapokon keresztül területeket rendelhetünk a képernyőn
- Az osztások tetszés szerinti mélységben egymásba ágyazhatók
- Korábban: `frame`

Oldalak elrendezése – ismétlődő részek

- Stratégiák gyakran ismétlődő HTML részletek megjelenítésére:
  - Töredékekkel, ha a rész inkább HTML kódót tartamlaz: A `render :menu` a `_menu.html.erb` tartalmát ágyazza be
  - Helperekkel, ha a rész inkább Ruby kódot tartalmaz
  - A `content_for` helperrel a megjeleníthető adat más nézetben töredékben érhető el:

```
<% if content_for?(:p) %>
  <% yield :p %>
<% end %>
...
<% content_for :p do %>
  <p>Hello</p>
<% end %>
```

- `iframe`: cross-site scripting

Alapértelmezett stílussal rendelkező HTML elemek

- Címsor elemek: `h1`, `h2`, `h3`, `h4`, `h5`, `h6`
- Szövegkiemelés: `b`, `i`, `em`, `code`, `strong`
- Szövegformázás: `p`, `pre`, `br`
- Szövegrészek csoportosítása: `span` stíluslapokkal kiegészítve

HTML dokumentum fejrész

- A Rails alkalmazás alapértelmezett fejrésze a következő:

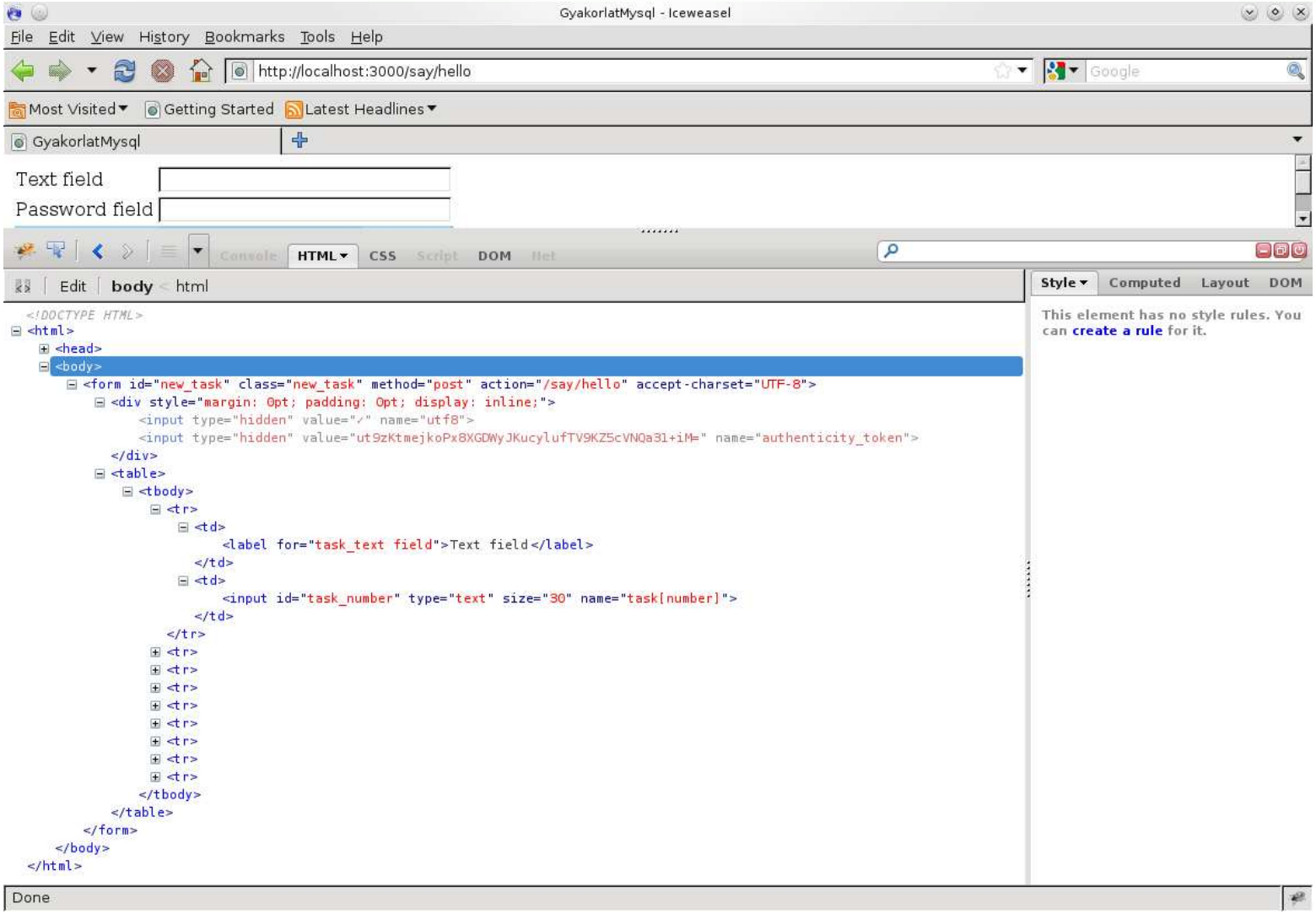
```
<title>Gyakorlat</title>
<%= csrf_meta_tag %>
<%= csp_meta_tag %>

<%= stylesheet_link_tag "application", media: "all", 'data-turbolinks-track': 'reload' %>
<%= javascript_pack_tag "application", 'data-turbolinks-track': 'reload' %>
```

- Definiálja a weboldal címét
- Betölti a Rails alkalmazás globális stíluslapját (`application.css`) az `app/assets/stylesheets` könyvtárból
- Betölti a Rails alkalmazás globális JavaScript forrását (`application.js`) a `app/javascript/packs` könyvtárból
- Hitelesítési tokent deklarál meta adatként (`csrf_meta_tags`, `csp_meta_tag`)
- Emellett megadható még: az oldal nyelve, az oldal karakterkódolása, az oldal élettartama, az oldal `Content-Type`-ja

HTML dokumentumok objektum modellje 1

- A HTML DOM minden HTML elemhez egy objektumot, tulajdonságokat és hozzáférési (lekérdez, hozzáad, módosít, töröl) metódusokat rendel
- A HTML DOM modellben a teljes dokumentumot egy fa reprezentálja, ahol
  - minden HTML elem egy csomópont,
  - a HTML elem tartalma pedig egy él.
  - A szöveges tartalom és a HTML elemek attribútumai speciális (levél)csomópontok.

HTML dokumentumok objektum modellje 2



HTML dokumentumok objektum modellje 3

- DOM tulajdonságok:
  - `innerHTML`, `nodeName`, `nodeValue`, `nodeType`
  - `parentNode`, `childNodes`
  - `attributes`
- DOM műveletek:
  - `getElementById("id")`, `getElementsByTagName("tagname")`
  - `appendChild(node)`, `removeChild(node)`

Kliens oldali események kezelése – JavaScript 1

- Kliens oldali események JavaScript függvényeket hívhatnak meg, amelyek módosíthatják az oldal DOM struktúráját vagy a megjelenés stílusát
- Események, amelyek HTML egyes elemek attribútumaként érhetők el
  - Oldal be- és kitöltése: `onload`, `onunload`
  - Egéresemények: `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmousemove`, `onmouseover`, `onmouseout`
  - Billentyű események: `onkeypress`, `onkeydown`, `onkeyup`
  - Oldal események: `onselect`, `onchange`, `onfocus`, `onblur`

```
<table onmouseover="this.style.color='#ff0000'" onmouseout="this.style.color='#000000'">
```

Kliens oldali események kezelése – JavaScript 2

- A JavaScript egy a böngésző által futtatott értelmezett nyelv
- `<script type="text/javascript">..</script>`

```
<head>
<script type="text/javascript">
function hello() { alert("Hello World!"); }
</script>
</head>
...
<input type="submit" value="Hello!" onclick="hello()" />
```

HTML elemek megjelenési stílusa – CSS

- A Cascading Style Sheets dokumentum azt definiálja, hogy az egyes HTML elemek hogyan jelenjenek meg
- A formázási direktívákat a HTML dokumentumtól elkülönítve tároljuk, így ugyanannak a dokumentumnak a megjelenése könnyen lecserélhető
- Egy CSS szabály két részből áll: egy kijelölésből és formázási deklarációk listájából
- A HTML forrás: `<span id="redspan" class="redspan">..</span>`
- Szintaxis példával: `span {color:red;font-size:12px;}`
- A kijelölés lehet: HTML elem alapú: `span`, `id` attribútum alapú: `#redspan`, `class` attribútum alapú: `.redspan`
- A kijelölés egymásba ágyazható, például a `div.box span` csak a `class="box"` attribútummal rendelkező `div`-eken belüli `span`-eket jelöli ki a DOM fából