



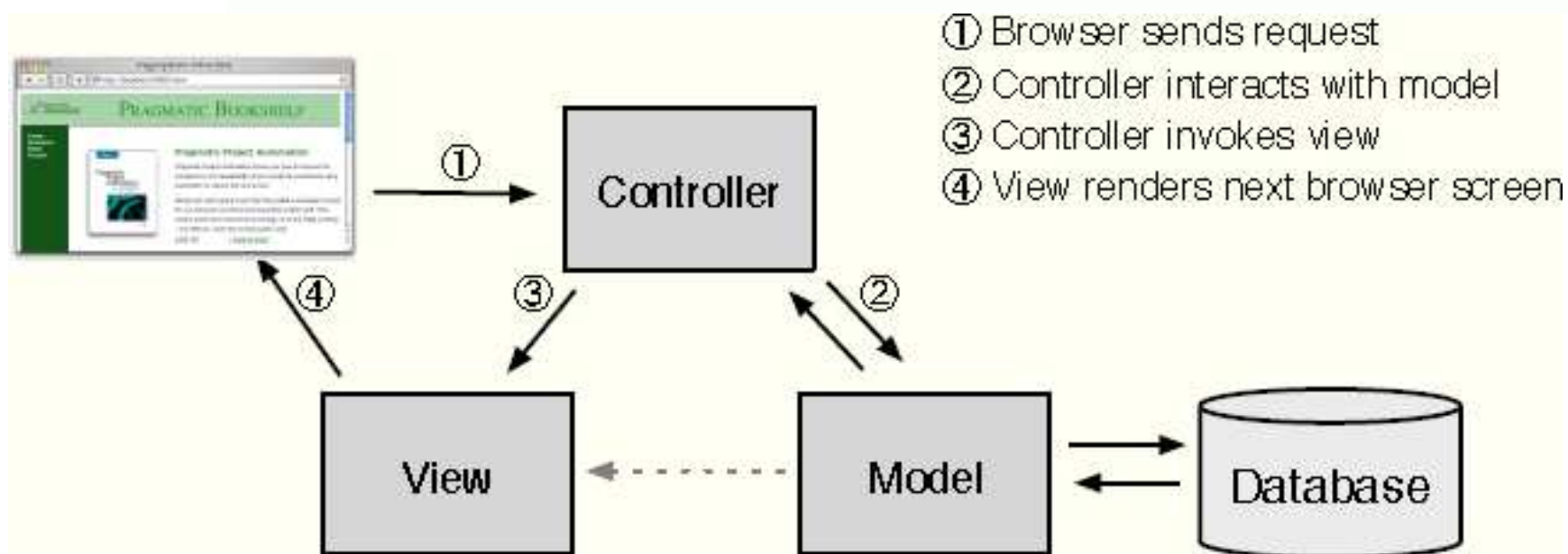
ActionController és ActionView

Kovács Gábor

`kovacsg@tmit.bme.hu`

BME-TMIT

MVC



ActionPack

- ⑥ Feladata a HTTP kérések feldolgozása és a HTTP válaszok generálása
- ⑥ Két modul:
 - △ **ActionController**
 - △ ActionView

Routing – Alapkérdés

- ⑥ Kérdés: *Honnan tudja a Rails, hogy egy kérést melyik kontroller szolgáljon ki?*
 1. A Rails a kérés URL-be kódolja ezt az információt több más paraméterrel együtt. A konfiguráció a `config/routes.rb`-ben található hash-en keresztül történik, ahol a bejegyzések sorrendjében történik a mintaillesztés.
 2. A kontroller azonosítása után a Rails példányosítja a kontroller osztályát, majd meghívja annak `process` metódusát átadva annak a kérés paramétereit (`params hash`)
 3. A kontroller kiválasztja és meghívja az `action`-nek megfelelő akciót a kontroller osztályban, amely feldolgozza a kérést.

Routing – típusok

6 A Rails 3 kétféle routingot támogat

- △ Az ún. resourceful útvonalak, ez az alapértelmezett, REST alapú útvonalakat generál

```
resources :issues
```

- △ Kötött, vagyis nem resourceful útvonalak

```
match ':controller(/:action(/:id(.:format)))'
```

6 Most

- △ Csak kötöttről lesz szó, a resourceful a REST tárgyalásakor
- △ Útvonalak tesztelése később

Routing – kötött útvonalak 1

- ⑥ Kötött útvonalak: a HTTP kérés URL-jének közvetlen leképezése egy controller egy akciójára
 - △ A `:controller` és a `:action` megadása
 - △ Minden egyéb paraméter a `params` hash-be kerül
 - △ Például: a `:controller(/:action(/:id(.:format)))`
 - △ Az `/users/show/1` jelentése: `users` controller, `show` akció, `params[:id]` értéke `1`

Routing – kötött útvonalak 2

⑥ Dinamikus útvonal részek, paraméterek:

- △ Tetszőleges számú paraméter megadható
- △ Például: `:controller/:action/:user_id(/:bid_id),`
`/users/show/1/2`
- △ Paraméter az URL-be kódolva: `/users/show/1?bid_id=2`

⑥ Statikus útvonal részek:

- △ Tetszőleges statikus útvonal szakasz megadható a leképezésben
- △ Például: `:controller/:action/ut/ide/:id` és
`/users/show/ut/ide/1`

Routing – kötött útvonalak 3

6 Alapértelmezett útvonalak megadása

- △ A `:controller` és az `:action` elhagyása
- △ Például: `match 'users/:id', to: 'users#show'`
- △ Paraméterezés: `:defaults` hash elemei a `params` hash-be kerülnek

6 Útvonalak elnevezése, helperek

- △ Útvonal átnevezése
`match 'index', to: 'users#index', as: :index`
- △ Két helper metódus jön létre: `index_path` és `index_url`
- △ Ezek a kontrollerben átirányításra használhatók fel, és `/index` értékkel térnek vissza

Routing – kötött útvonalak 4

6 Megkötések, joker karakterek

- △ A `:constraints` opcióval korlátozható

△ Például

```
match 'users/:id', to: 'users#show',  
constraints: { :id=>/[a-zA-Z0-9]{6}/ }
```

- △ A `*` karakter használható az útvonalakban, tetszőlegesen hosszú szakaszra illeszkedik

6 Átirányítás

- △ A `redirect` helperrel már az útvonalválasztáskor átirányítható egy kérés
- △ Például `match '/index', to: redirect('/users')`

Routing – Köötött útvonalak 5

6 A gyökér

- △ `root to: 'users#index'`

6 HTTP metódus

- △ Egy új controller létrehozásakor a következő kerül a `routes.rb`-ba

- △ `rails generate controller users index` esetén
`get 'users/index'`

- △ Ez megfelel a következő útvonal mintának:

```
match 'users/index', to: 'users#index', via: :get
```

ActionController – A kontroller működése

- ⑥ Kérés feldolgozásának menete
 1. Az `:action`-nel megegyező nevű publikus példánymetódus keresése
 2. A `method_missing` metódus meghívása, ha van
 3. Az akcióval megegyező template keresése a nézetek között
 4. Egyébként hiba
- ⑥ Publikus akciók elfedése: `hide_action` metódus

ActionController – A kontroller környezete

- ⌚ `action_name`: az épp feldolgozott akció neve
- ⌚ `cookies`: egy hash a sütitet tárolására
- ⌚ `headers`: a válasz HTTP fejrész hash-e
- ⌚ `params`: a kérés paraméterek hash-e
- ⌚ `request`: a bejövő kérés objektuma
- ⌚ `response`: az előállítandó válasz objektuma
- ⌚ `session`: egy hash session adatok tárolására

ActionController – A bejövő kérés

- ⑥ A `request` objektum a következő attribútumokkal rendelkezik
 - △ `domain`: a kérés domén neve
 - △ `remote_ip`: a távoli gép IP címe
 - △ `env`: a böngésző által beállított HTTP fejrészek hash-e
 - △ `method`: a kérés HTTP parancsa, illetve `get?` stb. a HTTP parancs azonosítására

ActionController – Válasz a felhasználónak 1

- ⑥ A kontrollernek és az akciónak megfelelő nézet (az `app/views`-ből) megjelenítése a kontrollerben beállított információkkal. Ez a leggyakoribb.
- ⑥ A kontroller közvetlenül készíti el a választ a nézettel való együttműködés nélkül. Például hibaoldalak megjelenítése.
- ⑥ A kontroller nem csinál semmit. Például AJAX kérés kiszolgálásakor.
- ⑥ A kontroller egyéb adatot, például fájlt küld.

ActionController – Válasz a felhasználónak 2

- ⑥ Nézet megjelenítése: `render`
- ⑥ Ez az alapértelmezett hívás, ha az akció metódusa üres vagy nincs definiálva
- ⑥ Paraméterezzhető:
 - △ Szöveg küldése a válaszban: `render(:text=>'Hello World')`
 - △ Beágyazott HTML küldése a válaszban:
`render(:inline=> %{<h1>Hello World</h1>})`
 - △ Más akció által előállított válasz küldése: `render(:action=>'new')`
 - △ Más elérési úttal megadott nézet küldése:
`render(:file=>'app/views/solutions/new.html.erb')`
 - △ Más nézet küldése: `render(:template=>'solutions/new')`
 - △ Üres törzs küldése: `render(:nothing)`

ActionController – Válasz a felhasználónak 3

6 Fájl küldése `send_data`, `send_file`

```
send_data(pic, :type=>'image/png', :disposition=>'inline')
send_file('/public/data/oweyoa_2.pdf', :type=>'application/pdf')
```

6 Állítható opciók:

- △ A Content-type a `:type` kulccsal
 - △ A böngésző viselkedése: megjelenítés (`inline`) vagy az alapértelmezett mentés (`attachment`)
 - △ Az elmentendő fájl neve: `:filename`
- 6 A Content-type és a többi HTTP válasz fejrész a `controller headers hash`-én keresztül is állítható

ActionController – Válasz a felhasználónak 4

- ⑥ Átirányítás: `redirect_to`
- ⑥ Hasonlít a `render :action-re`, azonban itt a kérés és a válasz URL különbözik
- ⑥ Paraméterek:
 - △ Átirányítás másik akcióra: `redirect_to :action=>:show`
 - △ Átirányítás másik nézetre:
`redirect_to :file=>'public/404.html'`
 - △ Átirányítás az előző oldalra: `redirect_to :back`

ActionController – Sütik

- ⑥ A HTTP állapotmentes, a memóriát a szerver által a böngészőben elhelyezett sütik adják
- ⑥ A sütiket a `cookies` hash tárolja, amely kizárólag string típusú adatokat tartalmaz
- ⑥ Süti opciók
 - △ A `:value` hordozza a süti értékét
 - △ A `:domain` és a `:path` az elérési úttal korlátozza a süti érvényességi körét
 - △ Az `:expires` élettartamot rendel a sütihez
`:expires => 7.days.from_now`
 - △ A `:secure` HTTPS kérésekre korlátozza a sütit
- ⑥ A süti letiltható böngészőben

ActionController – Sessionök

- ⑥ `session` egy összetett adatstruktúrák tárolására alkalmas hash, amely kérések között állandó marad
- ⑥ A sessiont egy `_session_id` azonosítja, amit sütiben küld át a Rails a kliensnek
- ⑥ A session túléli a verziófrissítést!
- ⑥ Session törlése: a `reset_session` metódussal például kilépéskor
- ⑥ Session opciók: a session is korlátozható
 - :`session_domain-re`, `session_path-ra` és
 - :`session_secure-ra`

ActionController – Sessionök tárolása

- ⑥ Stratégiák a `session_store` beállítására
- ⑥ A `session` hash tárolható
 - △ szerializált formában a fájlrendszeren a `tmp` könyvtárban. Ez (`CHI::Session::PStore`) az alapértelmezés
 - △ ActiveRecord-ként (`:active_records_store`) az adatbázisban egy `sessions` táblában
 - △ a hálózaton több szerver között elosztva (`:drb_store`)
 - △ memóriában (`:memory_store`), ez nem éli túl az alkalmazás újraindítását
- ⑥ A session-ök takarítására a `session` sütihez kötött `:expires` opcióval gondoskodhatunk

ActionController – Flash

- ⑥ Átirányítás (`redirect_to`) használata esetén új HTTP kérés jön létre, a korábban beállított példányváltozók értékei elvesznek
- ⑥ A `flash` hash egy temporális tárhely a következő kérés számára, `flash.now` az aktuális kérésben is elérhető
- ⑥ Leggyakoribb alkalmazása üzenetek küldése a következő oldalra
- ⑥ Például:

```
flash[:notice]='Sikeres bejelentkezés'
```

ActionController – Szűrők

- ⑥ Több akció során előforduló kódrészleteket privát metódusokkal és szűrőkkel valósíthatunk meg
- ⑥ Típusok: `before_filter`, `after_filter`, `around_filter`
- ⑥ Az előszűrő a példányváltozók inicializálására alkalmas
- ⑥ Az utószűrő a generált `response` módosítására, például tömörítésére alkalmas
- ⑥ Az `around` szűrő, ha tartalmaz `yield`-et, ami átadja a vezérlést a meghívott akciónak, akkor úgy viselkedik, mint egy elő- és utószűrő a `yield` előtti és utáni kódrészletet tekintve

ActionController – Gyorsítótár

- ⑥ Éles (production környezet) rendszerekben érdemes használni gyakori kérések gyorsabb kiszolgálására
- ⑥ Egész oldal (`cache_page`) vagy egy-egy akció (`cache_action`) cache-elhető
- ⑥ Mit érdemes? Olyan oldalakat, amiket a Rails alkalmazás kontrollál. Idő, session, és adatbázison kívüli adatokat tartalmazó oldalakat nem.
- ⑥ A gyorsítótár törölhető a `expire_page` és `expire_action` metódusokkal

A nézet és a controller

- ⑥ Milyen adatokhoz férnek hozzá a nézetek?
 - △ A controller példányváltozóhoz
 - △ A controller objektumához (a flash kivételével csak debug célra)
 - △ Az aktuális controllerhez
 - △ Az elérési úthoz `base_path`

Paraméterátadás formokban 1

```
6 users_controller.rb
```

```
def edit
  @user = User.find params[:id]
end
```

Paraméterátadás formokban 2

6 edit.html.erb

```
<%= form_for :user, :url=>{:action=>"update",:id=>@user.id} do |f| %>
  <div>
    <%= f.label :username %><br />
    <%= f.text_field :username, :size=>18 %><br />
  </div>
  ...
<% end %>
```

Paraméterátadás formokban 3

6 edit weboldal forrása

```
<form accept-charset="UTF-8" action="/users/update/1" method="post">
  ...
  <div>
    <label for="user_username">Username</label><br />
    <input id="user_username" name="user[username]"
      size="18" type="text" value="valaki" />
  </div>
  ...
</form>
```

Paraméterátadás formokban 4

6 A params hash a memóriában

```
@params={
  :id=>1,
  :user=>{
    username=>"valaki"
    ...
  }
}
```

Paraméterátadás formokban 5

6 users_controller.rb

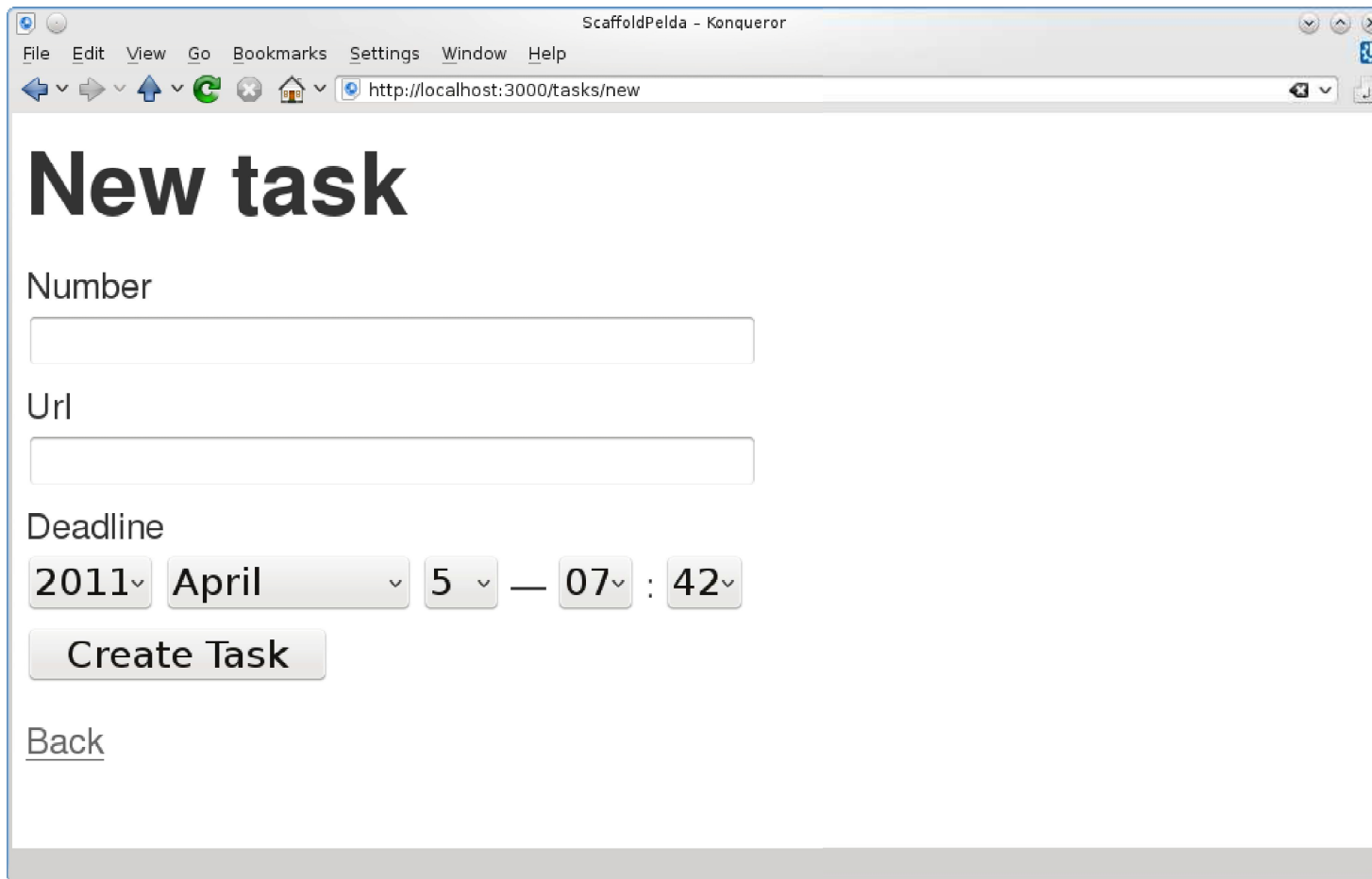
```
def update
  @user=User.find params[:id]
  user.update_attributes params[:user]
end
```

Scaffold 1

- ⑥ A `scaffold` egy automatikusan generálható keretrendszer modellek karbantartására. Létrehozza
 - △ a migrációt és a modell osztályt
 - △ `index`, `edit`, `show`, `new` nézeteket
 - △ egy formot, amelyet a `new` és `edit` nézetekben beágyazva jelenik meg
 - △ a kontroller osztályt a nézetek akcióival és a `create`, `update` és `destroy` metódusokkal
 - △ helper modulokat és teszt osztályokat
 - △ resourceful útvonalakat a `routes.rb`-ben
 - △ egy alapértelmezett stíluslapot

Scaffold 2

```
rails generate scaffold Issue number:integer url:string deadline:datetime
```



ScaffoldPelda - Konqueror

File Edit View Go Bookmarks Settings Window Help

http://localhost:3000/tasks/new

New task

Number

Url

Deadline

2011 April 5 07:42

Create Task

[Back](#)

ActiveSupport 1

⑥ Általános kiegészítések (minden osztály):

- △ Szerializáció: `to_json`, `to_yaml`
- △ Nincs tartalom (üres tömb, string csak whitespace karakterekből):
`blank`

⑥ Tömbök és Enumeration kiegészítések

- △ Tömb particionálása: `users.group_by{|u| u.student}`
- △ Tömbből hash: `users.index_by{|u| u.neptun}`
- △ Tömb összeg: `tasks.sum{|t| t.number}`
- △ Tömbből mondat: `failed_users.to_sentence`

ActiveSupport 2

6 String kiegészítések

- △ `at`, `from`, `to`, `first`, `last`, `starts_with`, `ends_with`
- △ Többes szám: `"user".pluralize`
- △ Egyes szám: `"users".singularize`
- △ Olvashatóvá tétel: `"jelszó_még_egyszer".humanize`

ActiveSupport 3

⑥ Number kiegészítések

- △ Páros/páratlan: `even?`, `odd?`
- △ Sorszámozás: `3.ordinalize # "3rd"`
- △ Skálázás: `3.bytes`, `3.kilobytes`, `3.megabytes` **stb.**
- △ Idő: `3.seconds`, `3.minutes`, `3.hours` **stb.**
- △ Időkülönbség: `ago`, `from_now`, `until`, `since`

⑥ Time és Date

- △ String konverzió: `t.to_s(:short)`, `t.to_s(:long)`, `t.to_s(:db)`