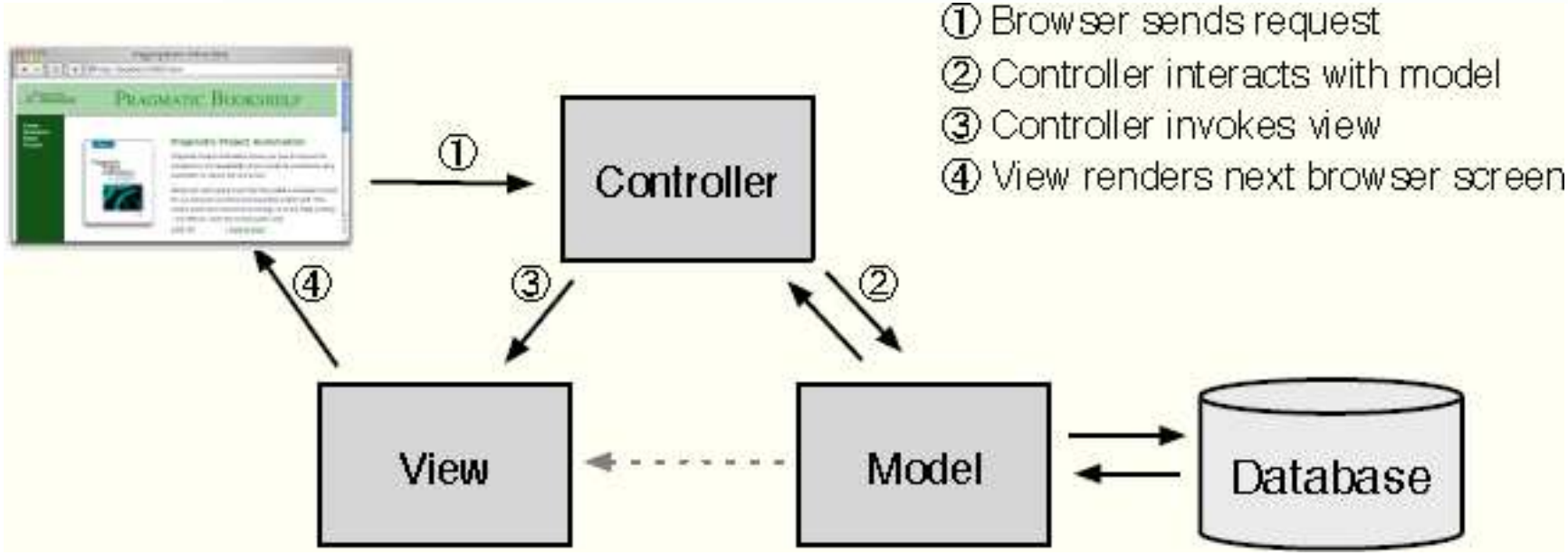
ActionController és ActionView

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

ActionPack

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

Routing – Alapkérdés

- 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

- A Rails 5 kétféle útvonalleírást támogat
  - Az ún. resourceful útvonalak, ez az alapértelmezett, REST alapú útvonalakat generál

    ```
    resources :tasks
    ```
  - Nem resourceful útvonalak

    ```
    match ':controller(/:action(/:id(.:format)))'
    ```
- Most
  - Csak nem resourceful útvonalaktól lesz szó, a resourceful a REST tárgyalásakor
  - A REST nem resourceful útvonalak konvenciókon alapuló halmaza
  - Útvonalak tesztelése később

Routing – Útvonalak 1

- Kötött útvonalak: a HTTP kérés URL-jének közvetlen leképezése egy kontroller 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` kontroller, `show` akció, `params[:id]` értéke 1

Routing – Útvonalak 2

- Dinamikus útvonal részek, paraméterek:
  - Tetszőleges számú paraméter megadható
  - Például: `:controller/:action/:user_id(/:task_id)`, `/users/show/1/2`
  - Paraméter az URL-be kódolva: `/users/show/1?task_id=2`
- Statikus útvonal részek:
  - Tetszőleges statikus útvonal szakasz megadható a leképezésben
  - Például: `:controller/:action/ut/ide/:id` és `/users/show/ut/ide/1`

Routing – Útvonalak 3

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

Routing – Útvonalak 4

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

Routing – Útvonalak 5

- A gyökér
  - `root to: 'users#index'`
- HTTP metódus
  - Egy új kontroller 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

- 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
- Publikus akciók elfedése: `hide_action` metódus

ActionController – A kontroller környezete

- A kontrollerek az `ApplicationController` leszármazottai, közvetve pedig a `ActionController::Base`-é
- Örökölt környezet:
  - `action_name`: az épp feldolgozott akció neve
  - `cookies`: egy hash a sütiket 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
- Örökölt viselkedés: `protect_from_forgery`

ActionController – A bejövő kérés

- 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

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

ActionController – Válasz a felhasználónak 2

- Nézet megjelenítése: `render`
- Ez az alapértelmezett hívás, ha az akció metódusa üres vagy nincs definiálva
- Paraméterezhető:
  - 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

- Fájl küldése `send_data`, `send_file`

```
send_data(pic, :type=>'image/png', :disposition=>'inline')
send_file('/public/data/aaaaaa_2.pdf', :type=>'application/pdf')
```

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

ActionController – Válasz a felhasználónak 4

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

ActionController – Válasz a felhasználónak 5

- Hibakezelés, kivételkezelés

```
class ApplicationController < ActionController::Base
  rescue_from ActiveRecord::RecordNotFound, with: :record_not_found

  private
    def record_not_found
      render plain: "404 Not Found", status: 404
      # redirect_to :file=>'public/404.html
    end
end
```

ActionController – Sütik

- A HTTP állapotmentes, a memóriát a szerver által a böngészőben elhelyezett sütik adják
- A sütiket a `cookies` hash tárolja, amely kizárólag string típusú adatokat tartalmaz
- 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
- A süti letiltható böngészőben

ActionController – Sessionök

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

ActionController – Sessionök tárolása

- Stratégiák a `session_store` beállítására
- 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
- A session-ök takarítására a session sütihez kötött `:expires` opcióval gondoskodhatunk

ActionController – Flash

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

ActionController – Szűrők

- Több akció során előforduló kódrészleteket privát metódusokkal és szűrőkkel valósíthatunk meg
- Típusok: `before_filter`, `after_filter`, `around_filter`
- Az előszűrő a példányváltozók inicializálására alkalmas
- Az utószűrő a generált `response` módosítására, például tömörítésére alkalmas
- 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

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

A nézet és a kontroller

- Milyen adatokhoz férnek hozzá a nézetek?
  - A kontroller példányváltozóhoz
  - A kontroller objektumahoz (a flash kivételével csak debug célra)
  - Az aktuális kontrollerhez
  - Az elérési úthoz `base_path`

Paraméterátadás formokban 1

users_controller.rb

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

Paraméterátadás formokban 2

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

- `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

- A params hash a memóriában

```
params={
  :controller=>"users",
  :action=>"update",
  :id=>1,
  :user=>{
    username=>"valaki",
    ...
  }
}
```

Paraméterátadás formokban 5

users_controller.rb

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

Paraméterek védelme

- Csak a fehér listában szereplő paraméternevek használhatók
- Minden más kivételt generál

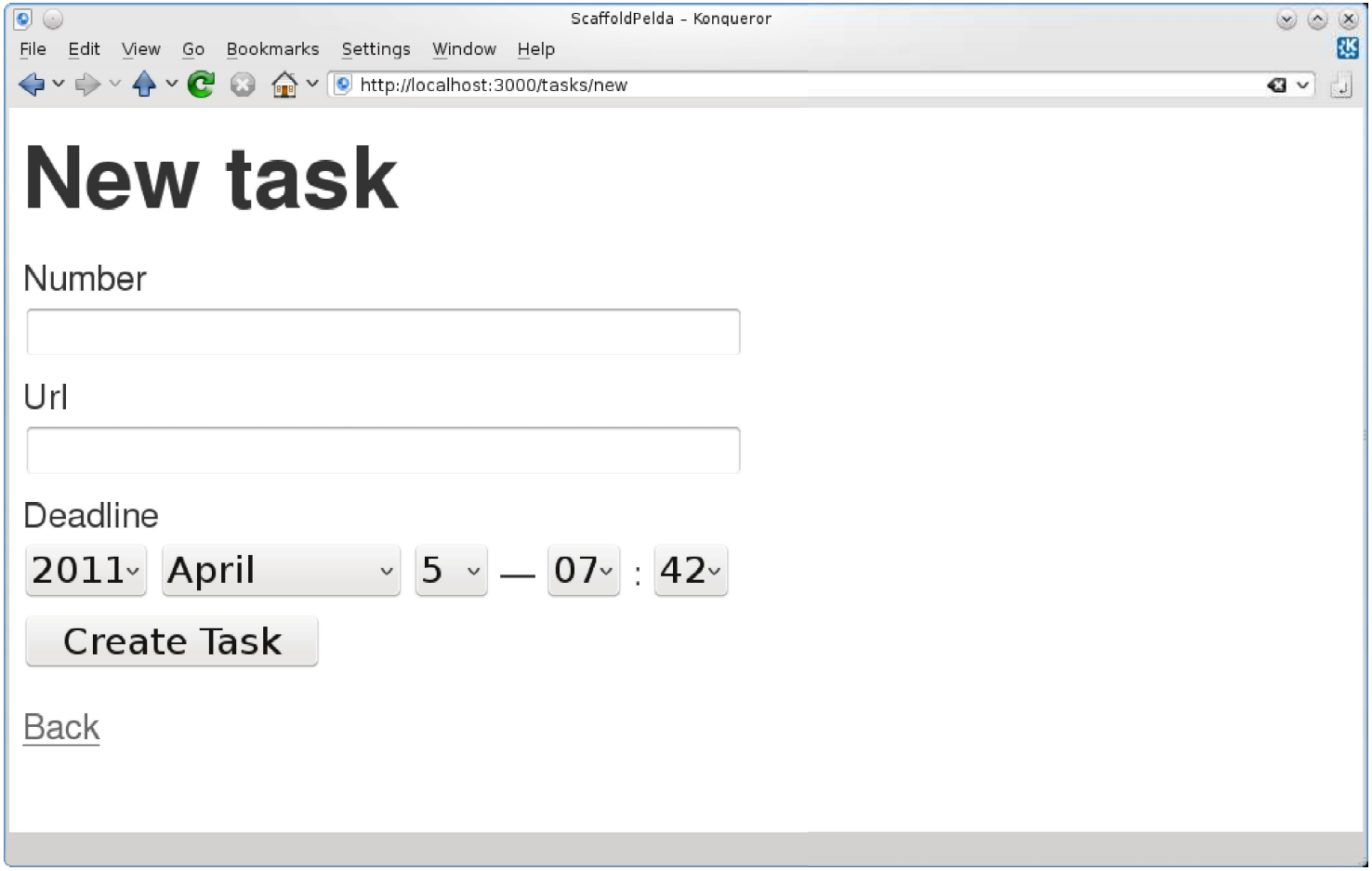
```
class UsersController < ActionController::Base
  def create
    @user = User.new(user_params)
  end
  private
    def user_params
      params.require(:user).permit(
        :username, :password, :name, :email,
        :password_confirmation)
    end
end
```

Scaffold 1

- 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` nétetekben 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 Task number:integer url:string deadline:datetime
```



ActiveSupport 1

- A Ruby nyelv kiterjeszései
- Már nem töltődik be automatikusan, külön kérnünk kell

```
require 'active_support'
require 'active_support/core_ext/object/json'

require 'active_support/all'
```

- A betöltés lehet: függvényenként, típusonként vagy minden

ActiveSupport 2

- Általános kigészítések (minden osztály):
  - Szerializáció: `to_json`, `to_yaml`
  - Nincs tartalom (üres tömb, string csak whitespace karakterekből): `blank?`
- 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 3

- `String` kiegészítések
  - `at`, `from`, `to`, `first`, `last`, `starts_with`, `ends_with`
  - Többes szám: `"user".pluralize`
  - Egyes szám: `"users".singluarize`
  - Olvashatóvá tétel: `"jelszó_még_egyszer".humanize`
  - Osztálynévvé alakítás: `"admin_user".camelize`

ActiveSupport 4

- `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`
  - String konverzió: `t.to_s(:phone)`, `t.to_s(:currency)`, `t.to_s(:human)`, `t.to_s(:human_size)`, `t.to_s(:rounded)`, `t.to_s(:delimited)`, `t.to_s(:percentage)`
- `Time` és `Date`
  - String konverzió: `t.to_s(:short)`, `t.to_s(:long)`, `t.to_s(:db)`