ActiveResource, RESTful webszolgáltatások, AJAX, ActionMail

Kovács Gábor

kovacsg@tmit.bme.hu

BME-TMIT

REST ismétlés

- A HTTP kérés URI-ja egy erőforrást azonosít
- Egy erőforrásnak több reprezentációja van: HTML, XML, JSON stb.
- A HTTP metódus határozza meg az erőforrással teendő akciót
- Az alkalmazás állapotát az erőforráson elérhető linkek definiálják
- Például:

```
DELETE http://localhost:3000/tasks/1
GET http://localhost:3000/tasks/1
PUT http://localhost:3000/tasks/1
```

REST parancsok

HTTP metódus	Akció	Eseménykezelő	SQL
GET	index,show	index, show	select
POST	new	create	insert
PUT	edit	update	update
DELETE	delete	destroy	delete

Resourceful útvonalválasztás 1

- A scaffolddal létrehozott `Task` egy `resources :tasks` sort szúrt be

HTTP metódus	Útvonal	Kontroller akció	Útvonal helper
GET	/tasks	index	`tasks_path`
GET	/tasks/new	new	`new_task_path`
POST	/tasks	create	`tasks_path`
GET	/tasks/:id	show	`task_path(id)`
GET	/tasks/:id/edit	edit	`edit_task_path(id)`
PUT	/tasks/:id	update	`task_path(id)`
DELETE	/tasks/:id	destroy	`task_path(id)`

Resourceful útvonalválasztás 2

- Azonosító nélküli útvonal `resource :task` sort szúrt be

HTTP metódus	Útvonal	Kontroller akció	Útvonal helper
GET	/task/new	new	`new_task_path`
POST	/task	create	`task_path`
GET	/task	show	`task_path`
GET	/task/edit	edit	`edit_task_path`
PUT	/task	update	`task_path`
DELETE	/task	destroy	`task_path`

Resourceful útvonalválasztás 3

- Az útvonalhoz tartozó alapértelmezett kontroller átállítható:
  `resources :users, :controller :felhasznalo`
- Útvonal prefix felüldefiniálása: `:as => 'feladat'`
- Útvonal postfix felüldefiniálása: `:path_names=>{:new=>uj}`
- Útvonalak kihagyása: `:only`, `:except`
- Kontrollerek modulokba rendezése esetén az útvonalakat névtérhez rendelhetjük. Például az `Admin` modul útvonalaira:

```
namespace :admin do
  resources :tasks
end
```

- Ekkor az útvonalak a névtér nevének megfelelő előtaggal bővülnek, a példában `admin`

Resourceful útvonalválasztás 4

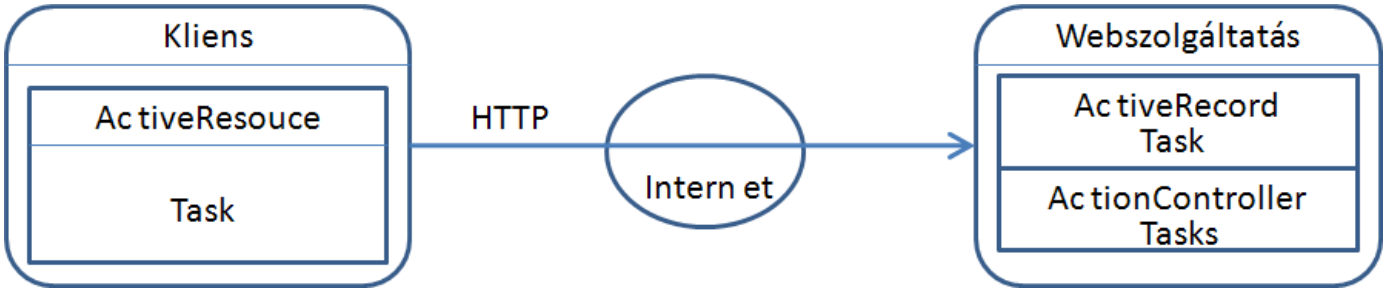
- Modell osztályok közötti relációk esetén (`has_many`, `belongs_to`) az útvonalak is egymásba ágyazhatók

```
resources :tasks do
  resources :users, :solutions
end
```

- A relációban lévő modell is elérhető ekkor a másik kontroller útvonalán kereszül `tasks/1/users/1`

RESTful webszolgáltatás

- Szolgáltatás: „A szolgáltatás egy absztrakt tevékenységek végrehajtásának képességét reprezentáló erőforrás, amely koherens funkcionalitást mutat mind a szolgáltató, mind a felhasználó entitások számára.”
- Webszolgáltatás: XML alapú hálózati kommunikáció alkalmazások között meghatározott interfész alapján



- A kontroller akciónak válaszolnia kell tudnia XML kérésekre

```
respond_to do |format|
  format.html { render :action => "new" }
  format.xml { render :xml => @task }
end
```

ActiveResource 1

- Az ActiveResource erőforrás megegyezik az ActiveRecord erőforrással
- Hivatkozás a távoli erőforrásra: `self.site`
- Hivatkozás másik erőforrásra, például az `User` ActiveResource-ból

  `self.site=http://localhost:3000/tasks/:task_id`
- A szerver oldali validációs hibából HTTP hibaüzenet lesz
- Hibák: a HTTP válasz státusz kódjának megfelelő kivétel dobódik,
  400-499: `ActiveResource::ClientError`,
  500-599: `ActiveResource::ClientError`,
  30x: `ActiveResource::Redirection`
- Hibás rekord: `task.errors.on :number`

ActiveResouce 2

```
require 'active_resource'
class Task < ActiveResource::Base
  self.site='http://localhost:3000'
end
tasks=Task.find :all # GET
task = Task.find :first
t = Task.new(:url=>'http://server.com/task4',
  :number=>4, :deadline=>Time.now)
t.save # POST
t.number=5
t.save # PUT
t.destroy # DELETE
```

ActiveResource 3

- Az ActiveResource saját sémát is definiálhat string, integer és float attribútumokból

```
class Message < ActiveResource::Base
  schema do
    integer 'from'
    integer 'to'
    string 'message'
    string 'date'
  end
end
```

Javascript Rails-ben

- Az alapértelmezett keretrendszer a Prototype
- A Prototype kiiktatható a projekt létrehozásakor a `--skip-prototype` kapcsolóval
- Rails 3-ban már lecserélhető jquery-re az `config/application.rb`-ben

```
config.action_view.JavaScript_expansions[:defaults] = %w(jquery rails application)
```

- A `public/javascripts` könyvtár tartalma:
  - Javascript keretrendszer: `prototype.js` vagy `jquery.js`
  - A Rails Javascript keretrendszer: `rails.js`
  - Saját Javascript források

AJAX támogatás Rails-ben

- HTML 5 attribútumok
  - `data-remote`: AJAX kérés
  - `data-method`: a használandó REST HTTP metódus
  - `data-confirm`: ellenőrző kérdés, lásd scaffold `delete` akció

AJAX hívás létrehozása

1. A `data-remote` attribútum `true`-ra állítása a form vagy a link HTML elemben: `:remote => true`
2. A kontroller akciók `respond_to` blokkjaihoz hozzáadjuk a Javascript formázást

```
respond_to do |format|
  format.html { render :action => "new" }
  format.js
end
```

3. A Rails válasza az AJAX kérésre egy Javascript betöltése, amelyet a nézetek között helyezünk el a kontroller akciónak megfelelő néven.
4. A hívás után a visszaadott Javascript (például `$(body).html('...')`) végrehajtásra kerül.

Rails AJAX események

- A Rails hat darab saját Javascript esemény definiál:
  - `ajax:before`: az AJAX hívás előtt hajtódik végre
  - `ajax:loading`: az AJAX hívás előtt, de a kérés összeállítása után hajtódik végre
  - `ajax:success`: sikeres AJAX hívás esetén hajtódik végre
  - `ajax:failure`: sikertelen AJAX hívás esetén hajtódik végre
  - `ajax:complete`: `ajax:success` vagy `ajax:failure` után hajtódik végre
  - `ajax:after`: az AJAX hívás elküldése után hajtódik végre

ActionMailer 1

- Az Action Mailer email üzenetek küldését teszi lehetővé Rails alkalmazásból
- A kliensek a `app/mailers` alatt jönnek létre a `rails generate mailer` parancs hatására
- A levél összeállítása:
  - A `default` hash: az összes levélre közös mezők
  - A `mail` levélküldő metódus: az aktuális levél mezői
  - A `headers` metódus: speciális fejrészek beállítása, `headers['MIME-Version]='1.0'`
  - Az `attachments` csatolmányokat fűz a levélhez, `attachments['me.png']=File.read('me.png')`
- Ezek elemei a levél fejléceit állítják, például `:from`, `:to`, `:subject`

ActionMailer 2

- A mailer kontrollere az `app/mailers` mailer osztálya, amely mailer akciókat tartalmaz (pl. `def notification`)
- A mailer nézetei, vagyis az üzenetek törzse az `app/views` alá kerülnek a mailer nevével megegyező könyvtárba, és a mailer akcióknak megfelelő néven
- Például `nofitication.html.erb` HTML törzsű levél esetén vagy `notification.text.erb` egyszerű szöveg esetén
- Ha több nézet is jelen van, akkor multipart MIME üzenetet hoz létre a Rails
- Az alapértelmezett nézet felüldefiniálható:

```
mail() do |format|
  format.html {render 'me'}
  format.text {render 'body'}
end
```

ActionMailer 3

- Csatolmányok befűzése
  - Inline: `attachments.inline['me.png']`
  - Linként:
    `<%= image_tag attachments['me.png'].url %>`
  - Kódolva:

```
attachments['me.png']={:mime_type => 'application/pdf',
  :encoding => 'pdf', :content => File.read('me.pdf')}
```

ActionMailer 4

- Levél fogadása
  1. A mailer osztályban egy fogadó metódus (pl. `receive`) definíciója
  2. A levelező portálon a levél továbbításának beállítása a `script/rails runner Mailer.receive(STDIN.read)` alkalmazásnak

ActionMailer 5

- A levelező kiszolgáló paramétereinek beállítása konfigurációs változókkal a `config` könyvtár egyik alkalmas fájljában (application.rb, development.rb, boot.rb stb.)
- Például

```
config.action_mailer.delivery_method = :smtp #:sendmail
config.action_mailer.smtp_settings={
  :address => 'mail.tmit.bme.hu',
  :port => '25' }
```