Laravel
hands on

kurz Webové technológie

Eduard Kuric

STU FIIT

3. 11. 2020

TaskManager – pokračovanie

- prednáške predchádza tutoriál z cvičenia

Routing

- nastavíme koreňovú cestu, aby smerovala na zoznam úloh

```
Route::get('/',
'TaskController@index');
```

Routing – Route:resource

- **Route::resource('tasks', 'TaskController');**
- Route::get('tasks/', 'TaskController@index');
- Route::get('tasks/create/', 'TaskController@create');
- Route::post('tasks/', 'TaskController@store');
- Route::get('tasks/{task}/', 'TaskController@show');
- Route::get('tasks/{task}/edit/', 'TaskController@edit');
- Route::put('tasks/{task}/', 'TaskController@update');
- Route::delete('tasks/{task}/', 'TaskController@destroy');

Úlohy sprístupnime iba autentifikovaným používateľom

Built-in autentifikácia

- použijeme autentifikáciu poskytovanú Laravelom

```
php artisan make:auth
```

v laravel6

```
composer require laravel/ui
```

```
php artisan ui vue --auth
```

Autentifikácia upravíme šablóny

- z `resources/views/layouts/app.blade.php` skopírujeme časť
  `<!-- Right Side Of Navbar -->`

  vložíme do …`/layout/partials/nav.blade.php`

- vo `views/auth`
  upravíme všetky šablóny, aby použili
  `layout/app.blade.php`

Routing- `Auth:routes()`

- **`Auth::routes();`**
- `$this->get('login', 'Auth\AuthController@showLoginForm');`
- `$this->post('login', 'Auth\AuthController@login');`
- `$this->get('logout', 'Auth\AuthController@logout');`
- `$this->get('register', 'Auth\AuthController@showRegistrationForm');`
- `$this->post('register', 'Auth\AuthController@register');`
- `$this->get('password/reset/{token?}', 'Auth\PasswordController@showResetForm');`
- `$this->post('password/email', 'Auth\PasswordController@sendResetLinkEmail');`
- `$this->post('password/reset', 'Auth\PasswordController@reset');`

Zaregistrujme používateľa ...

- po prihlásení nás presmeruje na

`http://127.0.0.1:8000/home`

- musíme zmeniť nastavenie prihlásenia

Autentifikovaný - `redirectTo`

- v `app/Http/Controllers/Auth/LoginController`
- v `app/Http/Controllers/Auth/RegisterController`

zmeníme presmerovanie:

- `protected $redirectTo = '/home';`

- `protected $redirectTo = '/';`

Iba autentifikovaný používateľ

- **nastavíme middleware** `auth`

```
Route::resource('tasks', 'TaskController',
['middleware' => 'auth']);
```

```
Route::get('/', 'TaskController@index')
->middleware('auth');
```

Úlohe priradíme jej autora, stĺpec (atribút) `user_id`

DB – pridanie `user_id`

- vytvoríme migračný súbor

```
php artisan make:migration
add_userid_to_tasks_table --table=tasks
```

- metóda `up()`

```
Schema::table('tasks', function($table) {
  $table->unsignedInteger('user_id');
  $table->foreign('user_id')
        ->references('id')->on('users');
});
```

- `php artisan migrate`
- **POZOR.: ak uvidíte exception, pravdepodobne už máte vytvorenú nejakú úlohu, zmazte ju z DB, alebo si osetrite migraciu ☺**

Prídáme `user_id` do `store()`

- do metódy `store()`, ktorá vytvára úlohu doplníme `user_id` (teda id aktuálne prihláseného používateľa)
- `TaskController@store`

```
use Illuminate\Support\Facades\Auth;

//$user = Auth::user();
$userId = Auth::id();

$task = Task::create([
   'title' => $request->title,
   'description' => $request->description,
   'user_id' => $userId]);
```

Prepojíme modely `user - task`

- vytvoríme vzťah 1:1 – **úloha má práve jedného autora**
- v modeli `Task` vytvoríme metódu `author()`

```
public function author()
{
    return $this->belongsTo('App\User', 'user_id');
}
```

- **POZOR.: ak by sme explicitne nepovedali, že cudzý kľúč je `user_id`, hľadal by `author_id`**
- **nezabudnime** na `fillable`

```
protected $fillable = ['title','description', 'user_id'];
```

V zozname úloh zobrazíme autora úlohy

TaskController@index

- v metóde `index()` vyberieme úlohy aj s ich autormi

- zmeníme

```
$tasks = Task::all();
```

- na

```
$tasks = Task::with('author')->get();
```

Šablóna `index.blade.php`

- v šablóne pre zoznam úloh doplníme stĺpec s autorom

`.../views/tasks/index.blade.php`

```
<th scope="col">Autor</th>
```

```
<td>{{$task->author->name}}</td>
```

Vytvorme úlohu...

Úlohu môže editovať a vymazať iba jej autor

Vytvorme nového používateľa a úlohu ...

Oprávnenie pre úlohu

- chceme, aby akcie (CRUD) nad modelom `Task` prešli najskôr kontrolou na oprávnenie
- vytvoríme oprávnenie – policy – pre úlohu

`app/Policies/TaskPolicy.php`

```
php artisan make:policy TaskPolicy
```

Zaregistrujeme oprávnenie

- v **app/providers/AuthServiceProvider** zaregistrujeme oprávnenie k modelu úlohy

```
use App\Policies\TaskPolicy;
use App\Task;
protected $policies = [
  'App\Task' => 'App\Policies\TaskPolicy',
];
```

TaskPolicy

- editovať a vymazať úlohu povolíme iba autorovi

```
use App\Task;

public function update(User $user, Task $task) {
  return $user->id === $task->user_id;
}

public function delete(User $user, Task $task) {
   return $user->id === $task->user_id;
}
```

Editovať/Vymazať v zozname

- v zozname úloh zobrazíme tlačidlo editovať/vymazať iba používateľovi, ktorý má oprávnenie (autorovi)

```
@can('update', $task)
@endcan

@can('delete', $task)
@endcan
```

Ešte routing + middleware

- pred vykonaním akcie sa musí overiť oprávnenie

```
use App\Task;
```

- Route::get('tasks/', 'TaskController@index')->middleware('auth');
- Route::get('/', 'TaskController@index')->middleware('auth');
- Route::get('tasks/create/', 'TaskController@create')->middleware('auth');
- Route::post('tasks/', 'TaskController@store')->middleware('auth');
- Route::get('tasks/{task}/', 'TaskController@show')->middleware('auth');
- Route::get('tasks/{task}/edit/', 'TaskController@edit')->middleware('auth', **'can:update,task');**
- Route::put('tasks/{task}/', 'TaskController@update')->middleware('auth', **'can:update,task');**
- Route::delete('tasks/{task}/', 'TaskController@destroy')->middleware('auth', **'can:delete,task');**

Úlohu môže editovať a vymazať iba jej autor **alebo** **ADMINISTRÁTOR**

Úlohu môže editovať a vymazať iba jej autor **alebo** **ADMINISTRÁTOR**

... používateľ, môže mať priradenú rolu ...

Vytvoríme model `Role`

- vytvoríme model s migračným súborom

```
php artisan make:model Role –m
```

- v migračnom súbore v metóde `up()` pridáme

```
$table->string('name');
$table->string('description');
```

Väzobná tabuľka

- používateľ môže mať niekoľko rolí, rovnakú rolu môže mať viacero používateľov, vzťah M:N
- vytvoríme migračný súbor pre väzobnú tabuľku

```
php artisan make:migration
create_role_user_table
```

- v migračnom súbore v metóde `up()` pridáme

```
$table->integer('role_id')->unsigned();
$table->integer('user_id')->unsigned();
```

Vzťah používateľ -> role

- v modeli `User` zadefinujeme vzťah k `Role`

```
public function roles()
{
   return $this->belongsToMany('App\Role')
         ->withTimestamps();
}
```

Vzťah rola -> používatelia

```
public function users()
{
    return $this->belongsToMany('App\User')
                        ->withTimestamps();
}
```

Insert + migrate

- v súbore `*_create_roles_table.php` doplníme v metóde `up()` vytvorenie role

```
DB::table('roles')->insert([
    'name' => 'ADMIN',
    'description' => 'ADMIN role'
]);
```

- `php artisan migrate`
- existujúcemu používateľovi priradíme rolu ADMINa (vzťah `user_role`)

Má používateľ rolu?

- v modeli `User` vytvoríme metódu, ktorá vráti `true`, ak má používateľ danú rolu, inak `false`

```
public function hasRole($role)
{
    if ($this->roles()->where('name', $role)->first()) {
        return true;
    }
    return false;
}
```

Upravíme `TaskPolicy`

- úlohu môže editovať, alebo vymazať jej autor, alebo používateľ, ktorý má rolu ADMIN

```
return $user->hasRole("ADMIN") ||
$user->id === $task->user_id;
```

Cache

- Pri zobrazení detailu úlohy, obsah je vybratý z DB a nacachovaný.
- Pri ďalšom zobrazení detailu, obsah je vybratý z cache (ak je platná).
- Cache je zneplatnená pri vymazaní a editovaní úlohy.

Cache - `TaskController`

- upravíme metódu `show`

```
use Illuminate\Support\Facades\Cache;

public function show(Task $task)
    {
        $id = $task->id;
        $task = Cache::remember('task-' . $id, 60,
            function () use ($id) {
                return Task::find($id);
            });

        return view('tasks.show', compact('task', $task));
    }
```

Skontrolujme vytvorenie cache súboru …

`storage/framework/cache/data`

Cache – `forget`

- vymažeme cache pre danú úlohu, ak zmeníme údaje v úlohe – metóda `update()`, alebo vymažeme úlohu – metóda `destroy()`,

```
if (Cache::has('task-'.$task->id)) {
    Cache::forget('task-'.$task->id);
}
```

Logging

Zalogujeme prihlásenie, resp. odhlásenie používateľa

Autentifikačné udalosti

- built-in aparát na autentifikáciu používateľa generuje udalosti
  - `Login`
  - `Logout`

- vytvoríme poslucháča, ktorý bude odberateľom udalosti a zároveň ju zaloguje
  - `LogSuccessfulLogin`
  - `LogSuccessfulLogout`

Zaregistrovanie udalostí

- **v app/providers/EventServiceProvide**r

```
protected $listen = [
        'Illuminate\Auth\Events\Login' => [
            'App\Listeners\LogSuccessfulLogin',
        ],

        'Illuminate\Auth\Events\Logout' => [
            'App\Listeners\LogSuccessfulLogout',
        ]
    ];
```

Vytvorenie poslucháčov

- zaregistrované udalosti vygenerujeme príkazom

```
php artisan event:generate
```

- príslušná handle metóda bude obsahovať

```
use Illuminate\Support\Facades\Log;

Log::info('Pouzivatel sa prihlasil',  ['id' =>
$event->user->id]);

Log::info('Pouzivatel sa odhlasil',  ['id' =>
$event->user->id]);
```

Vyskúšajme sa prihlásiť/odhlásiť...

- `storage/logs`

Slack

- nastavme webhook

https://wtechgroup.slack.com/apps/A0F7XDUAZ-incoming-webhooks?page=1

- webhook URL v config/logging.php
- logovanie na konkrétny kanál

```
Log::channel('slack')->critical('Pouzivatel sa prihlasil',  ['id' => $event->user->id]);
```

- PHP SSL certifikát

https://curl.haxx.se/docs/caextract.html

v php.ini:

curl.cainfo = "C:\xampp\php\extras\ssl\cacert.pem"

Localization

Vytvoríme jazykovú mutáciu pre aplikáciu

Smerovanie

- vytvoríme smerovanie, aktuálny jazyk budeme ukladať do session `app_locale`

```
Route::get('locale/{locale}', function ($locale) {
  if (in_array($locale,  config('app.supported_languages'))) {
     session(['app_locale' => $locale]);
  }
  return redirect()->back();
});
```

Config - podporované jazyky

- v `config/app.php` pridáme pole s podporovanými jazykmi:

```
'supported_languages' => array('sk', 'en'),
```

- nastavíme predvolený jazyk a fallback na `sk`

```
'locale' => 'sk',
'fallback_locale' => 'sk',
```

Middleware Language

- vytvoríme middleware `Language`

```
php artisan make:middleware Language
```

- do handle() metódy pridáme

```
use App;
App::setLocale(session('app_locale', config('app.locale')));
```

- zaregistrujeme middleware v `app\Http\Kernel.php`

```
protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\Language::class,
```

Úprava `nav` šablóny

- pridáme

```
<div class="btn-group">
  <button type="button" class="btn dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    {{App::getLocale()}}
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="/locale/sk">sk</a>
    <a class="dropdown-item" href="/locale/en">en</a>
  </div>
</div>
```

Prekladový súbor

- v `resources\lang` vytvoríme súbor `en.json`

```
{
  "Jednoduchý manažér úloh" : "Task manager",
  "Nová úloha" : "New task",
  "Prihlásenie" : "Login",
  "Registrácia" : "Register"
}
```

Obrázky k úlohám

Vytvoríme model `Image`

- k úlohe môže byť pripojených niekoľko obrázkov, vzťah 1:M
- vytvoríme model `Image`

```
php artisan make:model Image –m
```

- v migračnom súbore v metóde `up()` pridáme

```
$table->string('file');
$table->unsignedInteger('task_id');
  $table->foreign('task_id')
        ->references('id')->on('tasks');
```

Zadefinovanie vzťahu

- v modeli `Image` vytvoríme metódu

```
public function task() {
   return $this->belongsTo('App\Task');
}
```

- nezabudnime na **`fillable`**

```
 protected $fillable = ['file','task_id'];
```

Doplníme šablónu

- v šablóne `tasks/create.blade.php` v elemente `form` pridáme atribút

`enctype="multipart/form-data"`

- ďalej

```
<div class="form-group">
  <label for="image">Obrázok</label>
  <input type="file" name="image" id="image">
</div>
```

Upravíme metódu `store()`

```
use App\Image;
$validation = $request->validate([
          'title' => 'required|min:3',
          'description' => 'required',
          'image' => 'nullable|image|mimes:jpeg,png|max:1024'
]);

$userId = Auth::id();
$task = Task::create(['title' => $request->title,'description' => $request->description, 'user_id' => $userId]);

$file = $validation['image'];
$fileName = $task->id . '-' . md5($file->getClientOriginalName()) . time() . '.' . $file->getClientOriginalExtension();
$uploadedFile = $file->storeAs(config('app.tasks_images_path'), $fileName);
if ($uploadedFile) {
  Image::create(['file' => $fileName,'task_id' => $task->id]);
} else {}
```

Zadefinujeme cestu k obrázkom

- do `config/app.php` pridáme

```
'tasks.images.path' => 'public/tasks-images/',
```

- koreňová cesta je

`storage/app`

Nefunguje vymazanie úlohy...

- kvôli referenčnej integrite
- buď nastavíme OnDelete – kaskádu,
  - alebo predtým ako vymažeme úlohu, vymažeme obrázky

Transformácia obrázkov

- vyskúšajte

http://image.intervention.io/

```
composer require intervention/image:~2.4
```