Taskmanager - technológie POKRAČOVANIE

Vytvorenie prihlasovania - jednoducho

- Starter Kits

- `composer require laravel/breeze --dev`
- `php artisan breeze:install`
  - `0: blade`

Smerovanie

- nastavíme koreňovú cestu, aby smerovala na zoznam úloh
- `use App\Http\Controllers\TaskController;`

- `Route::get('/', [TaskController::class, 'index']);`
- `Route::resource('tasks',TaskController::class);`

Smerovanie - resource

- `Route::get('/tasks', [TaskController::class, 'index']);`
- `Route::get('/tasks/create', [TaskController::class, 'create']);`
- `Route::post('/tasks/', [TaskController::class, 'store']);`
- `Route::get('/tasks/{task}/', [TaskController::class, 'show']);`
- `Route::get('/tasks/{task}/edit/', [TaskController::class, 'edit']);`
- `Route::put('/tasks/{task}', [TaskController::class, 'update']);`
- `Route::delete('/tasks/{task}/', [TaskController::class,'destroy']);`

Taskmanager sprístupnime iba autentifikovaným používateľom

Upravíme šablóny

- V layout partials `nav.blade.php` za `navbarsExampleDefault`
  - vložíme fragment na nasledujúcom slajde
  - a odoberieme v `navbarsExampleDefault` triedu `navbar`

Upravíme šablóny 2

```
 @auth
            <div class="text-white">
                {{ Auth::user()->name }} | {{ Auth::user()->email }}
                <div>
                    <!-- Authentication -->
                    <form method="POST" action="{{ route('logout') }}">
                        @csrf
                        <x-responsive-nav-link :href="route('logout')"
                            onclick="event.preventDefault();
this.closest('form').submit();">
                            {{ __('Log Out') }}
                        </x-responsive-nav-link>
                    </form>
                </div>
            </div>
@endauth
```

Úprava `auth` šablón

Vo `views/auth` upravíme všetky šablóny tak, aby používali `layout/app.blade.php`

- Odoberieme

```
<x-guest-layout>
```

- Pridáme

```
@extends('layout.app')
@section('content’)
…
@endsection
```

Úprava partial `head`

- V `layout/partials/head.blade.php` doplníme

```
<!-- Scripts -->
@vite(['resources/css/app.css',
'resources/js/app.js'])
```

Nastavíme `auth` middleware v `routes`

- nastavíme middleware `auth`

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


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

Neželané presmerovanie

- Po registrácii nás presmeruje na `/dashboard`, chceme na `index`
- Pomenujeme route (získa alias `dashboard`):

```
Route::get('/', [TaskController::class, 'index'])-
>middleware(['auth'])->name('dashboard’)
```

FYI: presmerovanie prebieha v triedach `RegisteredUserController`, resp. `AuthenticatedSessionController` (`app/Http/Controllers/Auth/`)

Úlohe priradíme jej autora

Zmena v DP – úlohe pridáme `user_id`

- vytvoríme migračný súbor

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

- v metóde `up:`

```
Schema::table('tasks', function (Blueprint $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;
…
$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(User::class, '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
views/tasks/index.blade.php

- doplníme stĺpec s autorom:

```
<th scope="col">Autor</th>
…
<td>{{$task->author->name}}</td>
```

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 (ak neexistuje)

`app/Policies/TaskPolicy.php`

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

Zaregistrujeme oprávnenie (v 11 discover režim)

- **nie je potrebné, ak je dodržaná konvencia, Laravel automaticky zistí policy**

- v `app/providers/AppServiceProvider` zaregistrujeme oprávnenie k modelu úlohy

```
use App\Policies\TaskPolicy;
use App\Task;
use Illuminate\Support\Facades\Gate;
...
```

v metode `boot`:

```
Gate::policy(Task::class, TaskPolicy::class);
```

TaskPolicy

- editovať a vymazať úlohu povolíme iba jej 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 úloh

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

```
@can('update', $task)
…
@endcan

@can('delete', $task)
…
@endcan
```

Routing + middleware

- pred vykonaním akcie sa musí overiť oprávnenie:

```
use App\Task;

…
Route::get('/tasks', [TaskController::class, 'index'])-
>middleware(['auth']);

Route::get('/tasks/create', [TaskController::class, 'create'])-
>middleware(['auth']);

Route::post('/tasks/', [TaskController::class, 'store'])-
>middleware(['auth']);

Route::get('/tasks/{task}/', [TaskController::class, 'show'])-
>middleware(['auth']);

Route::get('/tasks/{task}/edit/', [TaskController::class, 'edit'])-
>middleware(['auth', 'can:update,task']);

Route::put('/tasks/{task}', [TaskController::class, 'update'])-
>middleware(['auth', 'can:update,task']);

Route::delete('/tasks/{task}/', [TaskController::class,'destroy'])-
>middleware(['auth', 'can:delete,task']);
```

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 (pivot table)

- 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 `user -> role`

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

```
public function roles()
{
    return $this->belongsToMany(Role::class)-
>withTimestamps();
}
```

Vzťah `role -> user`

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

```
public function users()
{
    return $this->belongsToMany(User::class)->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? (helper)

- 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;
```

- Pridajme do tabuľky `role_user` vzťah

Cache

Použitie cache – use case

- 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.

TaskController@show

- upravíme metódu `show`:

```
use Illuminate\Support\Facades\Cache;
…
$id = $task->id;
$task = Cache::remember('task-' . $id, 60,
     function () use ($id) {
          return Task::find($id);
     });
return view('tasks.show', ['task' => $task]);
```

- Skontrolujme vytvorenie cache …
- V súbore
  - `storage/framework/cache/data`
- Alebo v DB (predvolene od 11)
  - môžeme zmeniť v `config/cache.php` resp. v .env `CACHE_STORE`

Zneplatnenie cache

- vymažeme cache pre danú úlohu:
  - ak zmeníme údaje v úlohe – metóda `update`(),
  - alebo vymažeme úlohu – metóda `destroy`()
- za `save`, resp. `delete` prídáme:

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

Localization

Vytvoríme jazykovú mutáciu pre aplikáciu

Endpoint `locale/{locale}`

- vytvoríme smerovanie, aktuálny jazyk uložíme do session `app_locale`, ak je podporovaný

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

Config – zadefinujeme 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')));
return $next($request);
```

Zaregistrujeme middleware

- zaregistrujeme middleware v *bootstrap/app.php*

```
use App\Http\Middleware\Language;
...

->withMiddleware(function (Middleware $middleware) {
     $middleware->web(append: [
          Language::class,
     ]);
})
```

Úprava `nav` šablóny

- Pridáme za `navbarsExampleDefault`:

```
<div class="btn-group text-white">
<button type="button" class="btn dropdown-toggle"
data-toggle="dropdown" aria-haspopup="true" ariaexpanded="
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"
}
```

Prekladové reťazce (double underscore function)

- Upravme v `nav.blade.php`:

```
<a class="navbar-brand" href="/tasks">{{
__('Jednoduchý manažér úloh') }}</a>
```

Logging

Zalogujeme prihlásenie, resp. odhlásenie používateľa

Autentifikačné udalosti

- built-in aparát na autentifikáciu používateľa generuje udalosti:
  - `Login`
  - `Logout`
- vytvoríme poslucháčov, ktorý budú odberateľmi udalostí a zároveň ich zalogujú
  - `LogSuccessfulLogin`
  - `LogSuccessfulLogout`

Vytvorenie poslucháčov

- `php artisan make:listener` **`LogSuccessfulLogin --event=Login`**
- `php artisan make:listener` **`LogSuccessfulLogout --event=Logout`**

- Keďže chceme použiť špecifické autentifikačné udalosti generované Auth aparátom, musíme konkretizovať, o ktoré ide (do poslucháčov doplňme):
- `use Illuminate\Auth\Events\Login;`

resp.

- `use Illuminate\Auth\Events\Logout;`

Poslucháči – `handle` metóda

- príslušná handle metóda bude obsahovať

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

- resp.:

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

Nezabudnúť:

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

Vyskúšajme sa prihlásiť/odhlásiť...

- `storage/logs`

Slack integrácia

- Nastavme webhook
- https://wtech-corp.slack.com/apps/A0F7XDUAZ-incomingwebhooks?
- page=1&tab=more_info
- 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"`