

SERWISY I KOMENDY

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	1
Serwis WeatherUtil	1
Komendy	5
Commit projektu do GIT.....	8
Podsumowanie.....	8

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie następujących umiejętności:

- zamykanie reużywalnej logiki biznesowej w serwisach;
- wykorzystanie serwisów w kontrolerach;
- wykorzystanie serwisów w komendach;
- tworzenie komend konsolowych.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie zasad tworzenia serwisów, komend.

Wejściówka?

UWAGA

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

SERWIS WEATHERUTIL

Utwórz nową klasę src/Service/WeatherUtil.php, a w niej deklaracje dwóch metod:

- `getWeatherForLocation($location)` – odpowiedzialna za pobranie pomiarów (prognoza pogody) na podstawie encji lokalizacji;
- `getWeatherForCountryAndCity($countryCode, $cityName)` – odpowiedzialna za pobranie pomiarów na podstawie kodu kraju i nazwy miasta. Wewnętrzna implementacja sprowadza się do pobrania lokalizacji na podstawie kodu kraju i nazwy miasta, a następnie wywołania metody `getWeatherForLocation` dla otrzymanej lokalizacji.

```
<?php
declare(strict_types=1);

namespace App\Service;

use App\Entity\Location;
use App\Entity\Measurement;

class WeatherUtil
{
    /**
     * @return Measurement[]
     */
    public function getWeatherForLocation(Location $location): array
    {
        return [];
    }

    /**
     * @return Measurement[]
     */
    public function getWeatherForCountryAndCity(string $countryCode, string $city): array
    {
        return [];
    }
}
```

Zmodyfikuj `WeatherController`, żeby wykorzystywał nowy serwis:

<pre>class WeatherController extends AbstractController { #[Route('/weather/{country}/{city}', name: 'app_weather')] public function city(#[MapEntity(mapping: ['country' => Location \$location, MeasurementRepository \$repository,): Response { \$measurements = \$repository->findBy return \$this->render('weather/city.html.twig', ['location' => \$location, 'measurements' => \$measurements]); } }</pre>	<pre>class WeatherController extends AbstractController { #[Route('/weather/{country}/{city}', name: 'app_weather', request: 'app_weather')] public function city(#[MapEntity(mapping: ['country' => 'country', 'city' => 'city'])] Location \$location, WeatherUtil \$util,): Response { \$measurements = \$util->getWeatherForLocation(\$location); return \$this->render('weather/city.html.twig', ['location' => \$location, 'measurements' => \$measurements,]); } }</pre>
--	---

Sprawdź, czy strona prognozy pogody wciąż działa. Nie powinno być żadnych błędów, jednakże zwracana lista pomiarów będzie pusta.

Zaimplementuj ciało metody `getWeatherForLocation()`. Wstaw zrzut ekranu kodu:

pogodynka > src > Service > WeatherUtil.php > PHP Intelephense > WeatherUtil > \$locationRepository

```

10 use App\Repository\MeasurementRepository;
11
12 class WeatherUtil
13 {
14     public function __construct(
15         private readonly LocationRepository $locationRepository,
16         private readonly MeasurementRepository $measurementRepository,
17     ) {
18     }
19
20     public function getWeatherForLocation(Location $location): array
21     {
22         $measurements = $this->measurementRepository->findByLocation($location);
23         return $measurements;
24     }

```

Punkty:

0

1

Zaimplementuj ciało metody getWeatherForCountryAndCity(). Wstaw zrzut ekranu kodu:

```

24     }
25
26     public function getWeatherForCountryAndCity(string $countryCode, string $city): array
27     {
28         $location = $this->locationRepository->findOneBy([
29             'country' => $countryCode,
30             'city' => $city,
31         ]);
32         $measurements = $this->getWeatherForLocation($location);
33
34         return $measurements;
35     }
36 }
37

```

Punkty:

0

1

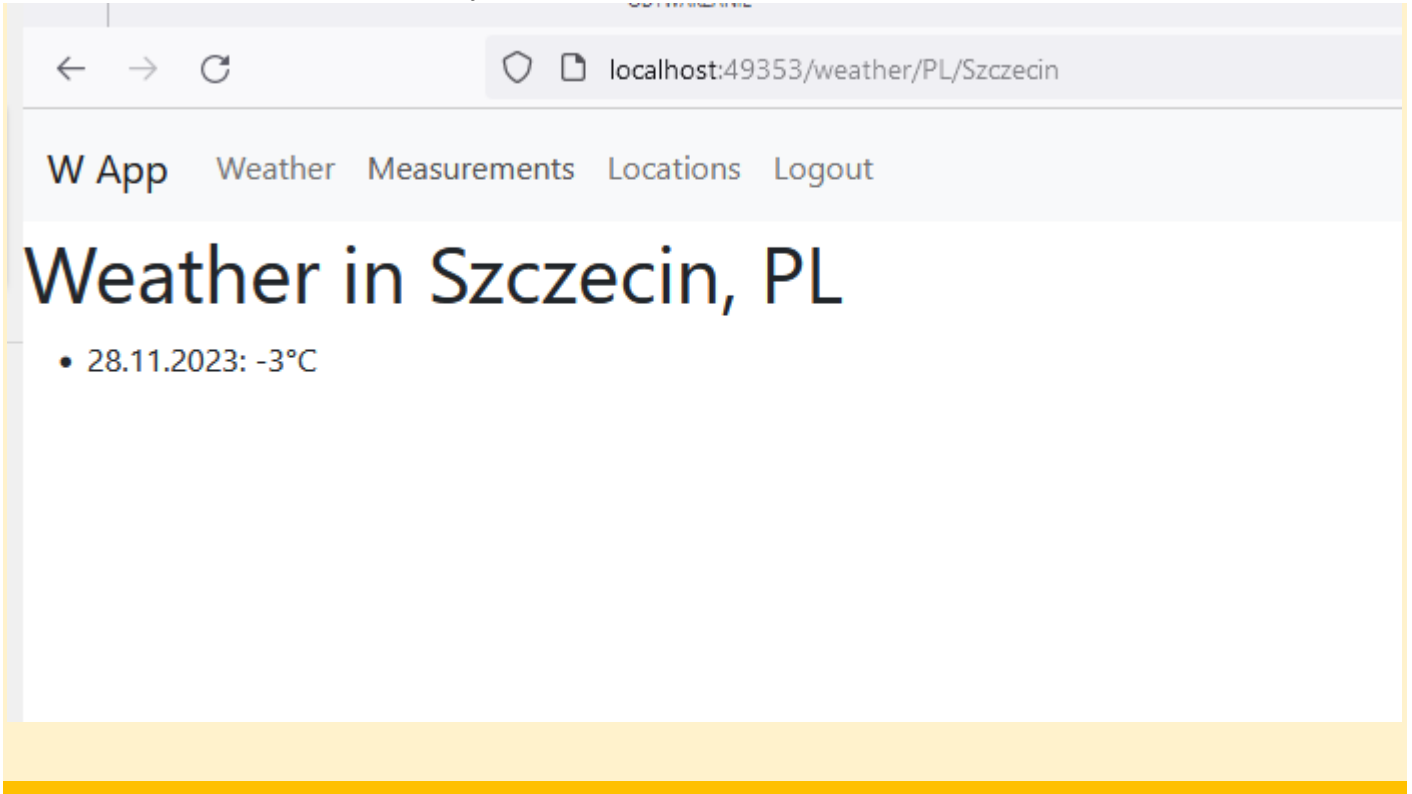
Wstaw zrzut ekranu kodu kontrolera wykorzystującego metodę z serwisu:

```

7  use Symfony\Component\HttpFoundation\Response;
8  use Symfony\Component\Routing\Annotation\Route;
9  use Symfony\Bridge\Doctrine\Attribute\MapEntity;
10 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
11
12 class WeatherController extends AbstractController
13 {
14     #[Route('/weather/{country}/{city}', name: 'app_weather')]
15     public function city(
16         #[MapEntity(mapping: ['country' => 'country', 'city' => 'city'])]
17         Location $location,
18         WeatherUtil $util,
19     ): Response {
20         $measurements = $util->getWeatherForLocation($location);
21
22         return $this->render('weather/city.html.twig', [
23             'location' => $location,
24             'measurements' => $measurements,
25         ]);
26     }
27     #[Route('/', name: 'app_weather_welcome')]
28     public function welcome(): Response
29     {
30         return $this->render('welcome.html.twig');
31     }
32     #[Route('/weather', name: 'app_weather_index')]
33     public function index(WeatherUtil $util): Response
34     {
35         return $this->render('weather/index.html.twig', [
36             'locations' => $util->getLocations(),
37         ]);
38     }
39 }
40

```

Wstaw zrzut ekranu prognozy pogody z pomiarami pobranymi z serwisu:



Punkty:	0	1
---------	---	---

KOMENDY

Wykorzystaj komendę `make:command` do utworzenia komendy `weather:location`, służącej do pobierania prognozy pogody dla lokalizacji:

```
php .\bin\console make:command

Choose a command name (e.g. app:agreeable-pizza):
> weather:location

created: src/Command/WeatherLocationCommand.php

Success!

Next: open your new command class and customize it!
Find the documentation at https://symfony.com/doc/current/console.html
```

Edytuj utworzony plik `src/Command/WeatherLocationCommand.php`:

- podłącz serwis do konstruktora;
- ustaw wymagany argument id lokalizacji;
- w `execute()` wykorzystaj serwis do pobrania prognozy pogody;
- wydrukuj prognozę pogody na widok.

Przykładowe wywołanie:

```
php .\bin\console weather:location 1
Location: Szczecin
```

2023-09-26: 18

2023-09-27: 17

Przykładowy kod:

```
protected function execute(InputInterface $input, OutputInterface $output): int
{
    $io = new SymfonyStyle($input, $output);
    $locationId = $input->getArgument('id');
    $location = $this->locationRepository->find($locationId);

    $measurements = $this->weatherUtil->getWeatherForLocation($location);
    $io->writeln(sprintf('Location: %s', $location->getCity()));
    foreach ($measurements as $measurement) {
        $io->writeln(sprintf("\t%s: %s",
            $measurement->getDate()->format('Y-m-d'),
            $measurement->getCelsius()
        ));
    }

    return Command::SUCCESS;
}
```

Wstaw zrzuty ekranu kodu całości komendy:

```

17  }}
18  class WeatherLocationCommand extends Command
19  {
20      public function __construct(
21          private readonly WeatherUtil $weatherUtil,
22          private readonly LocationRepository $locationRepository,
23          string $name = null,
24      ) {
25          parent::__construct($name);
26      }
27
28      protected function configure(): void
29      {
30          $this
31              ->addArgument('id', InputArgument::REQUIRED, 'Location ID');
32      }
33
34      protected function execute(InputInterface $input, OutputInterface $output): int
35      {
36          $io = new SymfonyStyle($input, $output);
37          $locationId = $input->getArgument('id');
38          $location = $this->locationRepository->find($locationId);
39          $measurements = $this->weatherUtil->getWeatherForLocation($location);
40          $io->writeln(sprintf('Location: %s', $location->getCity()));
41          foreach ($measurements as $measurement) {
42              $io->writeln(sprintf(
43                  "\t%s: %s",
44                  $measurement->getDate()->format('Y-m-d'),
45                  $measurement->getCelsius()
46              ));
47          }
48
49          return Command::SUCCESS;
50      }
51  }
52

```

Wstaw zrzuty ekranu wyniku działania komendy dla dwóch lokalizacji:

```

PS C:\Users\micha\Documents\GitHub\ai2\pogodynka> php .\bin\console weather:location 1
Location: Szczecin
2023-11-28: -3
PS C:\Users\micha\Documents\GitHub\ai2\pogodynka>

```

```

PS C:\Users\micha\Documents\GitHub\ai2\pogodynka> php .\bin\console weather:location 2
Location: Police
2023-11-28: -15
PS C:\Users\micha\Documents\GitHub\ai2\pogodynka>

```

Punkty:

0

1

W analogiczny sposób utwórz komendę do pobierania lokalizacji na podstawie kodu kraju i nazwy miejscowości.

Wstaw zrzuty ekranu kodu całości komendy:

```

13
14 #[AsCommand(
15     name: 'weather:city',
16     description: 'Displays measurements for location by country and city',
17 )]
18 class WeatherCityCommand extends Command
19 {
20     public function __construct(
21         private readonly WeatherUtil $weatherUtil,
22         private readonly LocationRepository $locationRepository,
23         string $name = null,
24     ) {
25         parent::__construct($name);
26     }
27
28     protected function configure(): void
29     {
30         $this
31             ->addArgument('country', InputArgument::REQUIRED, 'Country code')
32             ->addArgument('city', InputArgument::REQUIRED, 'City name');
33     }
34
35     protected function execute(InputInterface $input, OutputInterface $output): int
36     {
37         $io = new SymfonyStyle($input, $output);
38         $country = $input->getArgument('country');
39         $city = $input->getArgument('city');
40         $measurements = $this->weatherUtil->getWeatherForCountryAndCity($country, $city);
41         $io->writeln(sprintf('Location: %s', $city));
42         foreach ($measurements as $measurement) {
43             $io->writeln(sprintf(
44                 "\t%s: %s",
45                 $measurement->getDate()->format('Y-m-d'),
46                 $measurement->getCelsius()
47             ));
48         }

```

Wstaw zrzuty ekranu wyniku działania komendy dla dwóch miejscowości:

```

● PS C:\Users\micha\Documents\GitHub\ai2\pogodynka> php .\bin\console weather:city PL Szczecin
Location: Szczecin
2023-11-28: -3

● PS C:\Users\micha\Documents\GitHub\ai2\pogodynka> php .\bin\console weather:city PL Police
Location: Police
2023-11-28: -15

```

Punkty:	0	1
---------	---	---

COMMIT PROJEKTU DO GIT

Zacommittuj zmiany. Wyślij zmiany do repozytorium (push). Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie `lab-f` na podstawie głównej gałęzi kodu.

Podaj link do brancha `lab-f` w swoim repozytorium:

...link, np. <https://github.com/ideaspot-pl/ai2-pogodynka-202310/tree/lab-f...>

<https://github.com/Michaldariusznowakowski/ai2/tree/lab-f>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium nauczyłem się do czego służą serwisy i jak ich używać.

Dodatkowo nauczyłem się tworzyć własne komendy które mogę używać z linii poleceń.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.