



**Ministerul Educației, Culturii și Cercetării a Republicii Moldova**

**Universitatea de Stat a Moldovei**

**Facultatea Matematică și Informatică**

# Raport

**Lucrare de laborator Nr.3**

**la disciplina Framework-uri pentru Dezvoltarea de Aplicații Web  
Tema: ”Servicii”**

**A efectuat : Bafîr Ana**

**A verificat : Bodrug Svetlana**

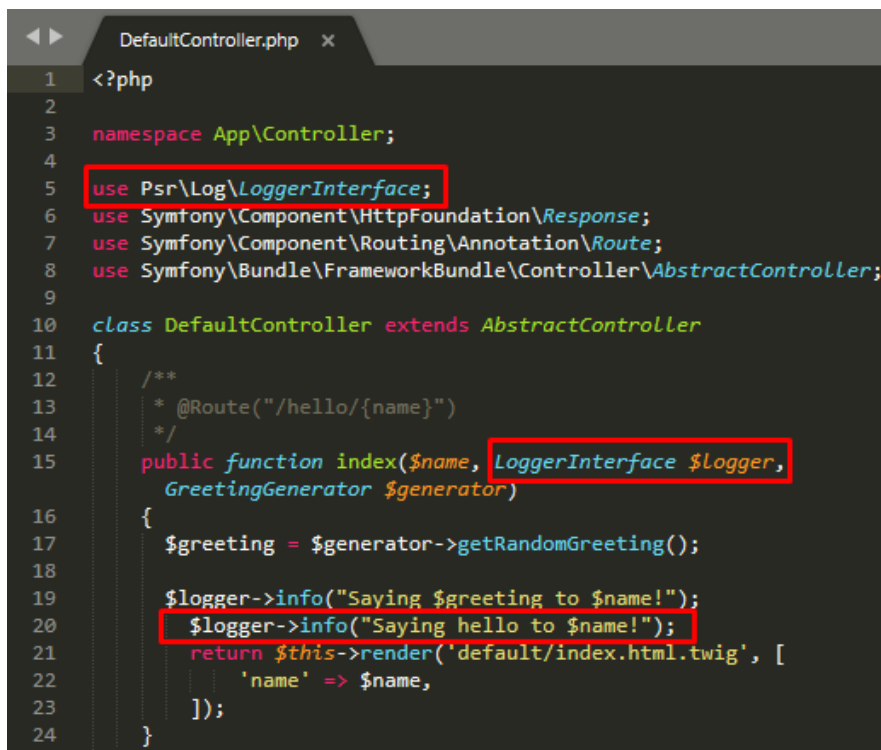
**Chișinău 2020**

## Adăugarea jurnalului

Noua aplicație Symfony este microscopică: constă în esență doar dintr-un sistem de rutare și un controler. Dar, datorită Flex, instalarea de noi funcții este ușoară.

```
F:\symfony\Lab2\quick_tour>composer require logger
```

Aceasta instalează și configurează (prin rețetă) puternica bibliotecă Monolog. Pentru a utiliza loggerul într-un controler, adaug un argument nou, tastat de LoggerInterface:



```
1 <?php
2
3 namespace App\Controller;
4
5 use Psr\Log\LoggerInterface;
6 use Symfony\Component\HttpFoundation\Response;
7 use Symfony\Component\Routing\Annotation\Route;
8 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
9
10 class DefaultController extends AbstractController
11 {
12     /**
13      * @Route("/hello/{name}")
14      */
15     public function index($name, LoggerInterface $logger,
16         GreetingGenerator $generator)
17     {
18         $greeting = $generator->getRandomGreeting();
19         $logger->info("Saying $greeting to $name!");
20         $logger->info("Saying hello to $name!");
21         return $this->render('default/index.html.twig', [
22             'name' => $name,
23         ]);
24     }
25 }
```

## Servicii și automatizare

Pot cere de la Symfony să-mi trimită un serviciu folosind tastarea. Cu ajutorul comenzii de mai jos voi afla cumpot fi utilizate alte clase sau interfețe:

```
F:\symfony\Lab2\quick_tour>php bin/console debug:autowiring
```

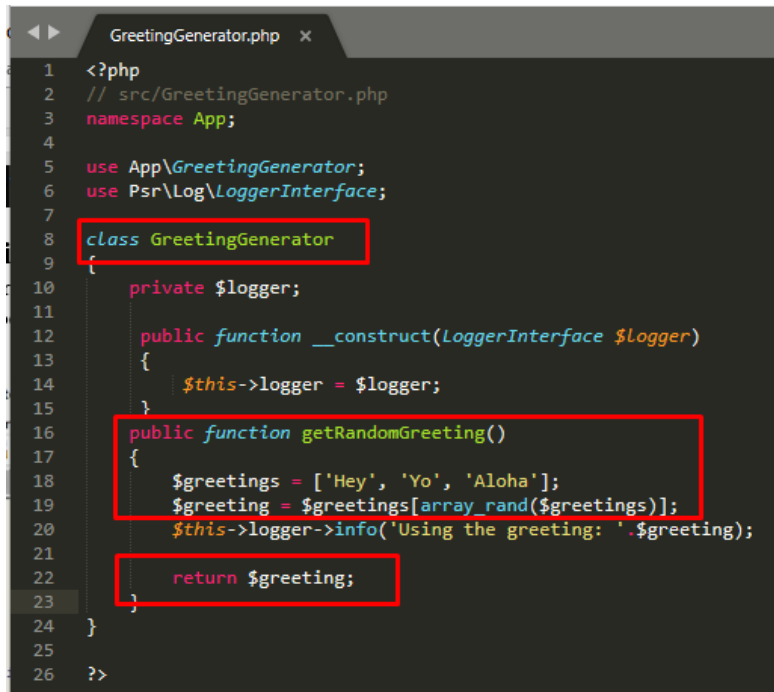
```
Validates PHP values against constraints.
Symfony\Component\Validator\Validator\ValidatorInterface (debug.validator)

Covers most simple to advanced caching needs.
Symfony\Contracts\Cache\CacheInterface (cache.app)

Allows invalidating cached items using tags.
Symfony\Contracts\Cache\TagAwareCacheInterface (cache.app.taggable)
```

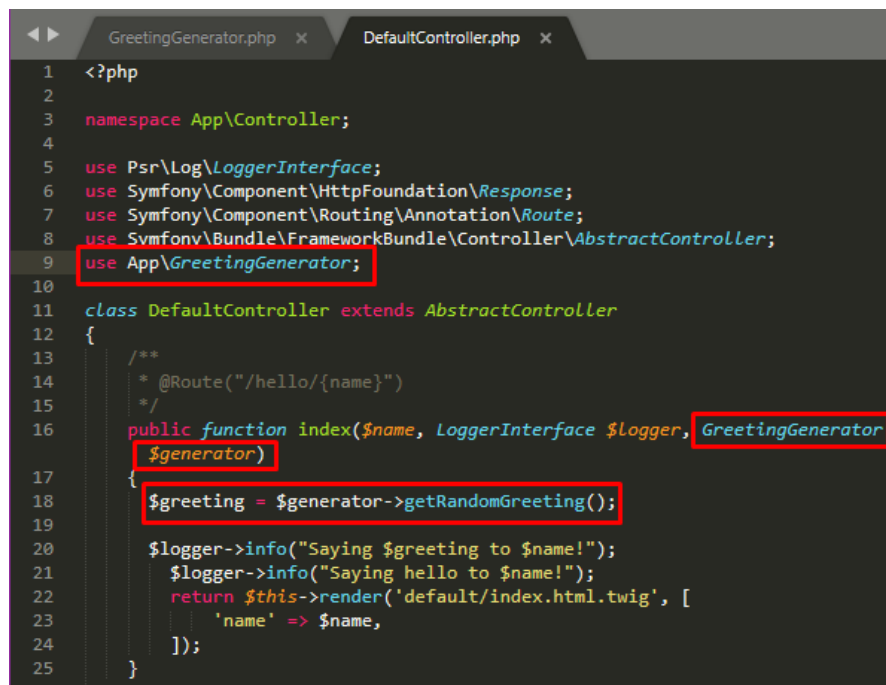
## Crearea serviciului

Pentru a menține codul organizat, pot să-mi creez propriile servicii! Dacă doresc să generez o felicitare aleatorie (de exemplu, „Bună ziua”, „Yo” etc.). În loc să introduc acest cod direct în controler, creez o nouă clasă:



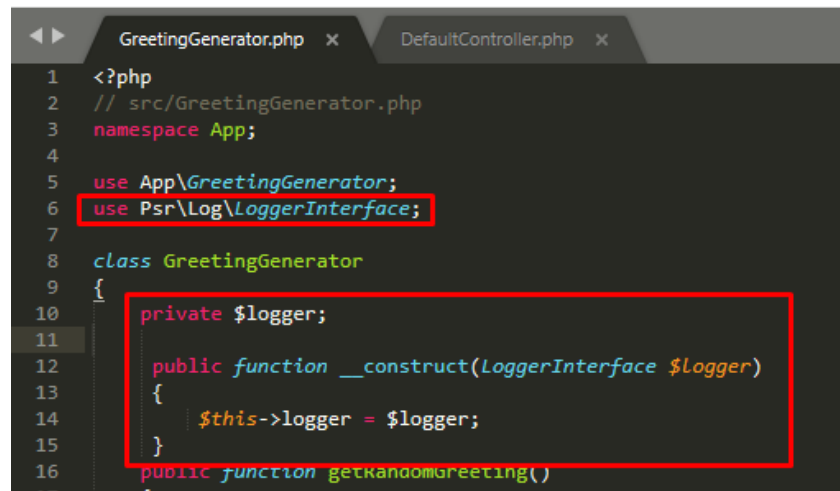
```
1 <?php
2 // src/GreetingGenerator.php
3 namespace App;
4
5 use App\GreetingGenerator;
6 use Psr\Log\LoggerInterface;
7
8 class GreetingGenerator
9 {
10     private $logger;
11
12     public function __construct(LoggerInterface $logger)
13     {
14         $this->logger = $logger;
15     }
16
17     public function getRandomGreeting()
18     {
19         $greetings = ['Hey', 'Yo', 'Aloha'];
20         $greeting = $greetings[array_rand($greetings)];
21         $this->logger->info('Using the greeting: '.$greeting);
22
23         return $greeting;
24     }
25 }
26 ?>
```

Pot utiliza acest lucru în controler imediat:



```
1 <?php
2
3 namespace App\Controller;
4
5 use Psr\Log\LoggerInterface;
6 use Symfony\Component\HttpFoundation\Response;
7 use Symfony\Component\Routing\Annotation\Route;
8 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
9 use App\GreetingGenerator;
10
11 class DefaultController extends AbstractController
12 {
13     /**
14      * @Route("/hello/{name}")
15      */
16     public function index($name, LoggerInterface $logger, GreetingGenerator $generator)
17     {
18         $greeting = $generator->getRandomGreeting();
19
20         $logger->info("Saying $greeting to $name!");
21         $logger->info("Saying hello to $name!");
22         return $this->render('default/index.html.twig', [
23             'name' => $name,
24         ]);
25     }
26 }
```

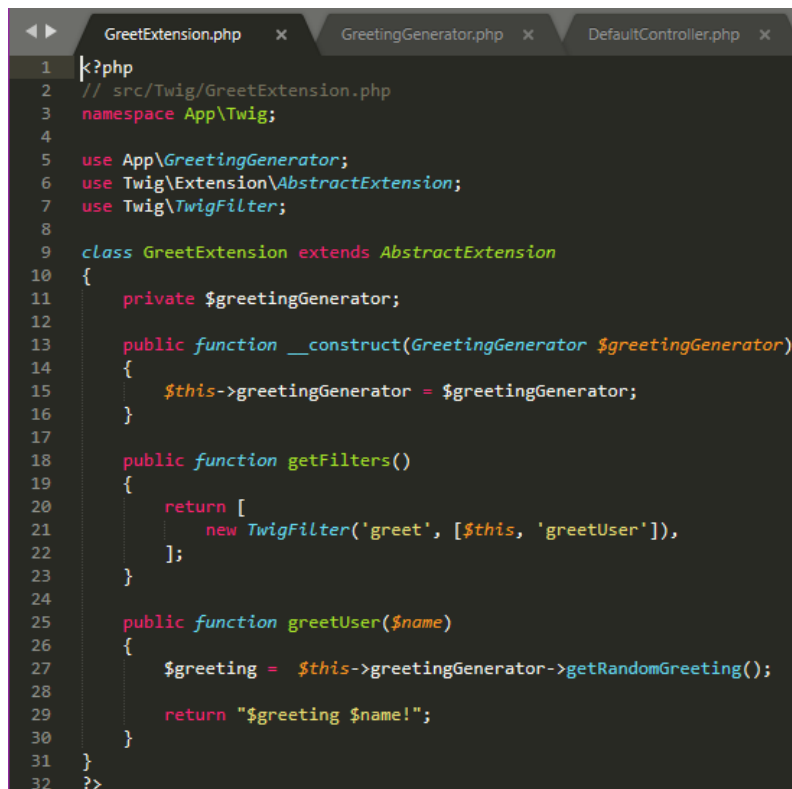
Symfony va instanția GreetingGenerator automat și îl va transmite ca argument. Pot utiliza cablarea automată în cadrul unui serviciu pentru a accesa alte servicii. Singura diferență este că acest lucru se face în constructor:



```
1 <?php
2 // src/GreetingGenerator.php
3 namespace App;
4
5 use App\GreetingGenerator;
6 use Psr\Log\LoggerInterface;
7
8 class GreetingGenerator
9 {
10     private $logger;
11
12     public function __construct(LoggerInterface $logger)
13     {
14         $this->logger = $logger;
15     }
16     public function getRandomGreeting()
17 }
```

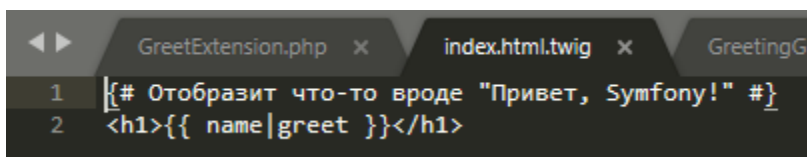
## Extinderea și autoconfigurarea Twig

Prin gestionarea serviciilor Symfony, pot extinde Symfony în multe moduri, cum ar fi crearea unui abonat la eveniment sau a unui alegător de securitate pentru reguli complexe de autorizare. Să adaug un nou filtru la Twig numit greet, creând o clasă care extinde AbstractExtension:



```
1 <?php
2 // src/Twig/GreetExtension.php
3 namespace App\Twig;
4
5 use App\GreetingGenerator;
6 use Twig\Extension\AbstractExtension;
7 use Twig\TwigFilter;
8
9 class GreetExtension extends AbstractExtension
10 {
11     private $greetingGenerator;
12
13     public function __construct(GreetingGenerator $greetingGenerator)
14     {
15         $this->greetingGenerator = $greetingGenerator;
16     }
17
18     public function getFilters()
19     {
20         return [
21             new TwigFilter('greet', [$this, 'greetUser']),
22         ];
23     }
24
25     public function greetUser($name)
26     {
27         $greeting = $this->greetingGenerator->getRandomGreeting();
28
29         return "$greeting $name!";
30     }
31 }
32 ?>
```

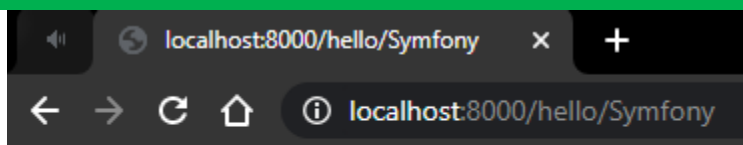
După crearea unui singur fișier, îl pot utiliza imediat:



A screenshot of a code editor with three tabs: 'GreetExtension.php', 'index.html.twig', and 'GreetingG'. The 'index.html.twig' tab is active, showing two lines of code:

```
1 |{{# Отобразит что-то вроде "Привет, Symfony!" #}}
2 |<h1>{{ name|greet }}</h1>
```

## REZULTAT:



**Yo Symfony!**