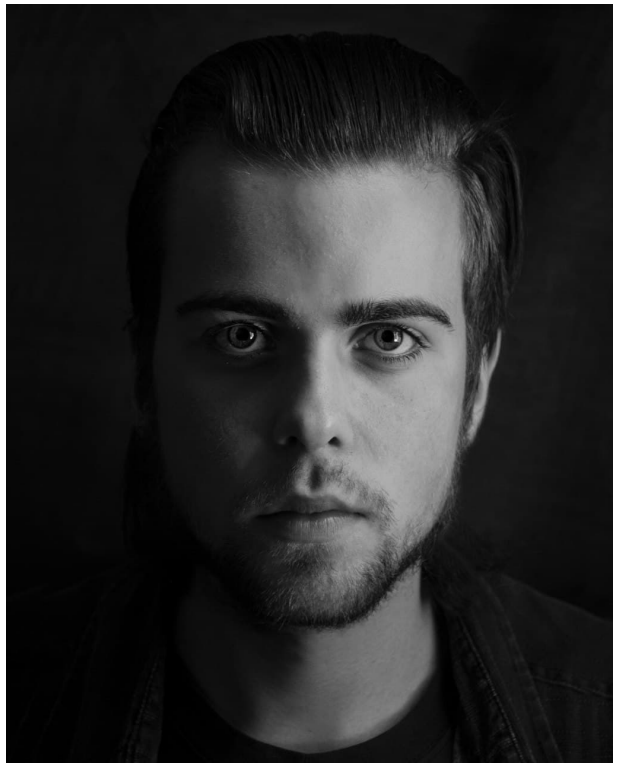
Веб разработка на PHP Symfony

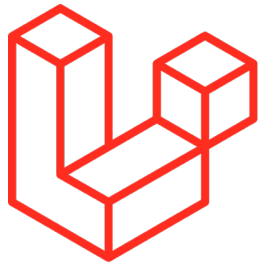
Роутер, контроллеры, ParamConverter, views

Introduction



Демьян Костельный
Middle PHP Developer

demian-kostelny-613b90151



Тема урока

Роутер, контроллеры, ParamConverter, views

План урока

1. Про контроллеры и роутинг
2. Annotations vs Yaml
3. Параметры Url и их валидация
4. Генерация Url
5. Установка ParamConverter, понятие Bundle
6. Использование ParamConverter

Про контроллеры и роутинг

Про контроллеры и роутинг

Контроллеры предоставляют в Symfony возможность полноценно прописывать всю логику приложения, которую вы хотите. Контроллер может принимать и отправлять запросы, а также взаимодействует со всеми другими элементами паттерна MVC (это Model и Views).

Перед тем как генерировать новый контроллер, нужно воспользоваться спец. инструментом от разработчиков Symfony, который называется symfony/maker-bundle. Установить его нужно с помощью Composer:

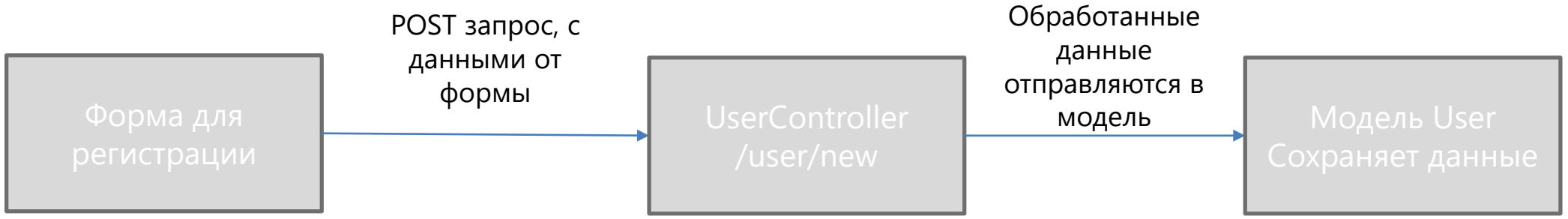
```
composer require symfony/maker-bundle
```

Чтобы создать новый контроллер, нужно воспользоваться командой:

```
php bin/console generate:controller NewController
```

Про контроллеры и роутинг

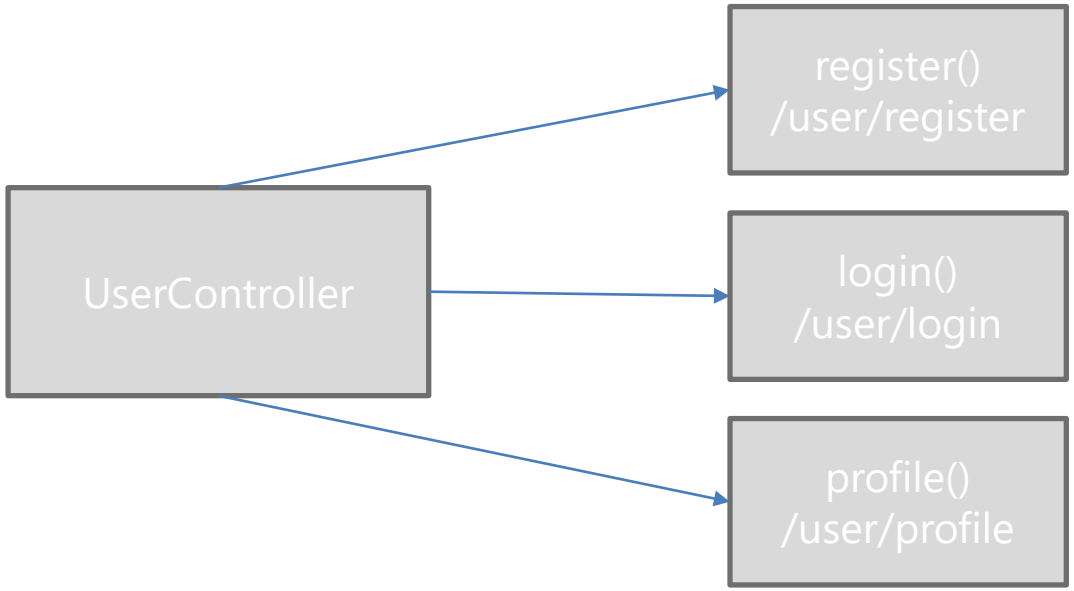
Используя маршрутизацию, контроллеры способны работать с самыми разными типами HTTP запросов. Принимаются любые типы запросов GET, POST, PUT, DELETE и другие. Задавать тип запросов можно как раз таки через маршрутизацию (роутинг).



Роутинг

Роутинг

Роутинг – это то, с помощью чего можно полностью прописывать всю логику url адресов в приложении. Проще говоря – это создание собственных url адресов, привязанных к контроллерам и к их функциям.



Роутинг

Ниже приведён простой пример создания маршрута с url /test для TestController и функцией в нём testfunc в файле config/routes.yaml:

```
test_routes:
  path: /test
  controller: App\Controller\TestController::testfunc
```

Роутинг

Есть также и другой способ создания маршрутов, прямо через прописывание комментариев перед функциями контроллера, но делается это благодаря аннотациям о которых будет говориться дальше.

```php
class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home")
     */
    public function index()
    {
        // …
    }
}
```

Annotations vs Yaml

Annotations vs Yaml

Для использования маршрутизации с помощью annotations нужно для начала установить пакет с данным набором инструментов, но чаще всего данный пакет уже по умолчанию установлен в новом symfony приложении.

```
composer require doctrine/annotations
```

Настроить где именно annotations должны брать маршруты можно в файле config/routes/annotations.yaml:

```yaml
controllers:
  resource: ../../src/Controller
  type: annotation

kernel:
  resource: ../../src/Kernel.php
  type: annotation
```

Annotations vs Yaml

Что же касается роутинга с помощью Yaml, то уже был один пример на создание маршрута для TestController:

```yaml
test_routes:
  path: /test
  controller: App\Controller\TestController::testfunc
```

Annotations vs Yaml

Annotations представляет собой более простой способ создания url адресов для приложения, ведь все маршруты прописываются прямо в контроллерах. Что же касается Yaml, то здесь уже нужно отдельно в файле маршрутизации config/routes.yaml прописывать маршруты.

Параметры url и их валидация

Параметры url и их валидация

Часто есть ситуации когда в самом url нам нужно передать какие-то данные, и здесь на помощь как раз и приходят параметры url. Простым примером url, в котором передаются параметры, может послужить /pages/{параметр номера страницы}

```
class PostsController extends AbstractController
{
    /**
     * @Route("/post/{name}", name="post_show")
     */
    public function show()
    {
        // …
    }
}
```

Параметры url и их валидация

Предыдущий пример только уже для YAML маршрутизации:

```yaml
posts_show:
  path: /post/{name}
  controller: App\Controller\PostsController:show
```

Параметры url и их валидация

Валидация URL параметров представляет использование регулярных выражений, то есть с помощью валидации параметров можно прописать правила, которым должны соответствовать нужные url в контроллере:

```
class PostsController extends AbstractController
{
    /**
     * @Route("/post/{name}", name="post_show", requirements={"name"="\d+"})
     */
    public function show()
    {
        // …
    }
}
```

Параметры url и их валидация

Пример с валидацией – только уже для YAML:

```yaml
posts_show:
  path: /post/{name}
  controller: App\Controller\PostsController:show
  requirements:
    page: '\d+'
```

Параметры url и их валидация

Если нужно будет достать параметры из url, то это можно сделать следующим способом:

```php
class PostsController extends AbstractController
{
    /**
     * @Route("/post/{name}", name="post_show", requirements={"name"="\d+"})
     */
    public function show(Request $request): Response
    {
        $route_name = $request->attributes->get('_route'); // Обращаемся к переменной
        request чтобы достать название маршрута
        $route_parameters = $request->attributes->get('_route_params'); // Здесь уже
        достаём доступные параметры нашего маршрута

        $all_parameters = $request->attributes->all(); // Если нужно достать все
        параметры которые только можно - то можно воспользоваться функцией all()
    }
}
```

Генерация Url

Генерация Url

В контроллерах можно генерировать нужные url адреса и задавать им имена без создания маршрутизации.

```php
class ListController extends AbstractController
{
    /**
     * @Route("/list", name="list_show")
     */
    public function show(): Response
    {
        $lists_show_page = $this->generateUrl('show-lists'); // Это сгенерирует url с названием show-list без параметров в url

        $list_show_page = $this->generateUrl('show-list', [ // В этом же примере генерируется url в котором принимается аргумент list_id
            'list_id' => $list_id
        ]);
    }
}
```

Установка ParamConverter и понятие Bundle

Установка ParamConverter и понятие Bundle

Bundle – представляет собой отдельные плагины, которые можно подключать к приложению, кроме этого их можно и самому создавать. К примеру в новом созданном приложении уже есть подключенные Bundl'ы от разработчиков Symfony (FrameworkBundle, SecurityBundle, TwigBundle и т.д.). Посмотреть конфигурацию Bundl'ов можно в файле config/bundles.php:

```php
return [
    Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
    Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle::class => ['all' => true],
    Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
    Symfony\Bundle\WebProfilerBundle\WebProfilerBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MonologBundle\MonologBundle::class => ['all' => true],
    Symfony\Bundle\DebugBundle\DebugBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
    Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
    Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle::class => ['all' => true],
    Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
    Twig\Extra\TwigExtraBundle\TwigExtraBundle::class => ['all' => true],
];
```

Установка ParamConverter и понятие Bundle

Для создания нового Bundle в первую очередь нужно создать папку с названием по адресу src/Acme/MyBundle. Acme – это может быть название разработчика либо компании. Следующим шагом уже будет создать файл с название Bundl'a, в нашем примере это будет MyBundle.php:

```php
namespace App\Acme\MyBundle;

use Symfony\Component\HttpKernel\Bundle\Bundle;

class AcmeMyBundle extends Bundle
{
    // Здесь уже будет код самого bundl'a
}
```

Установка ParamConverter и понятие Bundle

После создания самого Bundl'a, нужно будет его подключить в файле конфигурации config/bundles.php:

```php
return [
    Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
    Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle::class => ['all' => true],
    Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
    Symfony\Bundle\WebProfilerBundle\WebProfilerBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MonologBundle\MonologBundle::class => ['all' => true],
    Symfony\Bundle\DebugBundle\DebugBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
    Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
    Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle::class => ['all' => true],
    Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
    Twig\Extra\TwigExtraBundle\TwigExtraBundle::class => ['all' => true],
    // Ниже добавляем namespace (путь) нашего Bundl'a
    App\Acme\MyBundle\MyBundle::class => ['all' => true]
];
```

Установка ParamConverter и понятие Bundle

Кроме того, что Bundl'ы представляют из себя полноценные плагины, которые можно подключать к приложению, они также являются небольшими мини-приложениями, которые также могут задавать в приложении свои маршруты и контроллеры. Вот структура директорий в Budl'е:

Controller/ - контроллеры в Bundl'е
DependencyInjection/ - DependencyInjection классы хранятся здесь
Resources/config/ - файлы конфигурации (к примеру для маршрутизации)
Resources/views/ - шаблоны и внешний вид в общем
Resources/public/ - публичные файлы (CSS, JS, изображения и т.д.)
Tests/ - модульные тесты

Установка ParamConverter и понятие Bundle

ParamConverter – инструмент, который можно использовать в аннотациях для того, чтобы конвертировать параметры запроса в нужные объекты. Они могут быть в аргументах функции, вот пример использования:

```php
// ...
use Symfony\Component\Routing\Annotation\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;

class PostsController extends AbstractController
{
    /**
     * @Route("/posts/{id}")
     * @ParamConverter("post", class="SensioBlogBundle:post")
     */
    public function show(Post $post)
    {

    }
}
```

Использование ParamConverter

Использование ParamConverter

В общих чертах уже понятно, что благодаря ParmaConverter мы конвертируем аргументы запроса в объект нужного нам класса. Сразу же хочу сказать, что конвертацию можно сделать автоматической. Для этого нужно сначала её включить в файле config/packages/sensio_framework_extra.yaml:

```yaml
sensio_framework_extra:
    router:
        annotations: true
    request:
        converters: true
        auto_convert: true
```

Использование ParamConverter

Пример с использованием автоконвертации:

```php
class PostsController extends AbstractController
{
    /**
     * @Route("/posts/{id}")
     */
    public function show(Post $post)
    {
        // Можно уже заметить что здесь не добавилось @ParamConverter
    }
}
```

Информационный видеосервис для разработчиков программного обеспечения

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA

Проверка знаний

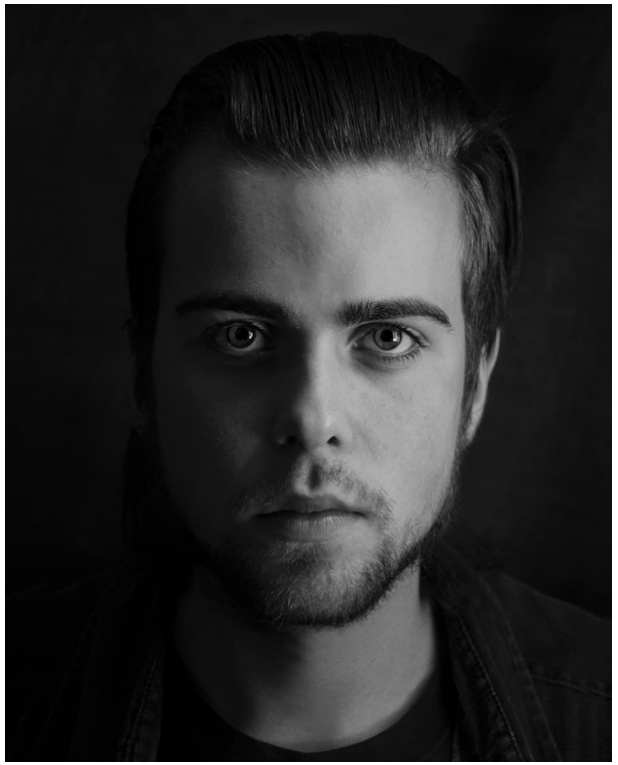
TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Проверьте как Вы усвоили данный материал на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!



Демьян Костельный
Middle PHP Developer