# SYMFONY 6

## A PRACTICAL GUIDE

HOW TO BUILD QUALITY FAST!

"I just wish this book had existed 5 years ago"

Tomáš Votruba
Creator of Rector

KERRIAL NEWHAM

# Symfony 6

## A Practical Guide

Kerrial Beckett Newham

This book is for sale at http://leanpub.com/symfony-6-a-practical-guide

This version was published on 2023-09-19

*for a little pickle*

# Contents

# Preface

For those of you that are more observant, you will have probably figured out the objective of this book is to have a practical guide for developing Symfony projects.

The reason I deemed this book necessary was the frustration due to the lack of practical resources available when I started learning programming with the Symfony framework.

I find the Symfony documentation too theoretical and more importantly, without real world examples of how to build common web application features quickly.

However, let's not be overly negative. If you don't like something, either change it or shut up! I decided to write this book and thus, I hope to change the situation.

My name is Kerrial Newham, nice to meet you curious readers. I've been developing for just under a decade, I taught myself programming via various online and offline resources.

I'm a fastidiously practical chap. I've worked with some of the largest and smallest companies in the world, developing Symfony applications. I'm not part of the official Symfony Team, I am not an expert, but merely sharing my experience of developing with this framework.

So enough with the jibber-jabber, bring on the code.

# Acknowledgement

If there is any vague sense of value one derives from this book, it'll be due to the many people I have been fortunate enough to have learnt from over an extended period of time. We all stand on the shoulders of giants.

Shan Newham: For editing support and providing insight and knowledge of the English language and for your valued input in discussions.

Sheyla Newham: For your unequivocal positivity and support from the outset of this endevour and for being the best, most patient, unwilling duck in the world.

Tomas Votruba: For his friendship and valuable input in both the beginning and end phases of the book, and for being an unbearably positive and intensely inspirational person to be around.

I'd like to also thank all my other friends for their patience for putting up with many of our conversations being dominated by the topic of this book.

# Introduction

This book has been written to reflect the real world of a development process that I have experienced over the years. but, keep in mind there is no objectively right way of doing this, the world is your code oyster and if it works for you, who cares. Alas, that being said, I have attempted to add practicality, quality, and simplicity within these pages.

I'm not going to focus on technical theory or how Symfony internals work, to find this information check the documentation and other resources. As much as possible, I'm going to show you how to do 'stuff' that I find practically useful, hopefully you will too.

I find in the world of programming, more often than not, that there is a certain level of intellectual snobbery and the gate keepers of knowledge get rather angery at questions not thought through.

In my option there is no such thing as a bad question. Even if one asks a stupid question (I have asked many stupid questions), if this allows one to progress their knowledge by the realisation of what is a good question, then it's worth it. The road to mastery is pathed with failure, bad questions and mistakes.

However, intellectual snobbery and gate keeping creates an atmosphere of unconducive for learning, cooperation and general curiosity.

Therefore, I'm a big advocate of the Feynman Technique of learning. The Feynman Technique often involves explaining complex topics as if you were teaching them to a child, using simple language and clear explanations.

Here are the steps in the technique:

1. Concept: Choose a concept you want to learn.
2. Teach It to a Child: Pretend you are teaching the chosen topic to a child. Use simple language and clear explanations. This step helps you break down complex ideas into easily digestible parts.
3. Identify Gaps: As you explain the topic in simple terms, you may encounter areas where your understanding is not as clear as you thought. Take note of these gaps in your knowledge.
4. Simplify: Simplify, repeat step two & three and create analogies.

Try not to hide your lack of knowledge no matter how 'advanced' you become. Celebrate and credit others who coming up with better, more effective solutions than your own. Be the fool and ask the dumb questions and you might see behid the facade of complexity.

# Prerequisites

Ideally, if you're new to Symfony, you'll probably get the most out of this book. That being said, seasoned developers could also find this book useful as a resource to gain inspiration, explore different approaches, and, hopefully, boost creativity, productivity, and quality.

The minimum amount of knowledge assumed in this book is that you know OOP (Object-oriented programming) basics, in this case, PHP syntax and simple terminal commands.

# How to use this book

Most programming books follow the development of a project, which is all good and grand. However, we will be doing it a little differently. This book is a collection of features, which is what any project consists of.

Use this book like a cheat sheet, e.g. you want to know how to send emails. Go to "Email Service Feature" in the Features section and find out quickly how it can be done, remembering that there are many, many approaches to building any feature, but even if you use this book as inspiration to come up with your own, better solutions, then great!

If you see something wrapped in double angle brackets, yours should replace this value; this is called a placeholder for example: "`<<replace with your value>>`" or "`<>`"

In every single relevant code snippet, you will see a comment at the top, this will tell you the relative path (from the project root directory) of the snippet file.

You may notice in the code sinppets I have left out the `use` statements, this was difficult decision, as I know how useful that can be sometimes, but unfortunately, it would just clutter the book too much.

# What is Symfony?

Symfony is an open-source PHP web application framework that follows the MVC(Model-View-Controller) architectural pattern.

It provides a set of reusable components and libraries that help developers build robust and scalable web applications.

Basically, that's a fancy way of saying we can build websites and console commands using the framework.

Symfony was initially released in 2005 and has since gained significant popularity in the PHP community. It aims to simplify the development process by promoting reusable code, maintainability, and scalability.

# The Basics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Coding Standards

So before we start on this saga. We must establish some coding conventions that we will be following. This may seem obvious to some of you, but not to others.

The following tools will be used for ensuring coding standards and quality:

- Rector[1]
- Easy Coding Standard[2]
- Php stan[3]

### Rector

from the docs "Rector instantly upgrades and refactors the PHP code of your application. It can help you in 2 major areas: Instant Upgrades and Automated Refactoring" we will be using it for automated refactoring, meaning that it'll fix our code automatically.

- run `composer require rector/rector --dev`
- run `vendor/bin/rector init`

This should create a configuration file in the root directory called "rector.php" let's update is to look something like this:

[1] https://github.com/rectorphp/rector
[2] https://github.com/symplify/easy-coding-standard
[3] https://github.com/phpstan/phpstan

```php
<?php

// ./rector.php

declare(strict_types=1);

return static function (ContainerConfigurator $containerConfigurator): void {

    $parameters = $containerConfigurator->parameters();
    $parameters->set(Option::AUTO_IMPORT_NAMES, true);
    $parameters->set(Option::PHPSTAN_FOR_RECTOR_PATH, __DIR__ . './phpstan.neon');

    $parameters->set(Option::PATHS, [
        __DIR__ . '/src',
        __DIR__ . '/config',
        __DIR__ . '/tests'
    ]);

    $containerConfigurator->import(SetList::CODE_QUALITY);
    $containerConfigurator->import(SetList::TYPE_DECLARATION);
    $containerConfigurator->import(SymfonySetList::SYMFONY_60);
    $containerConfigurator->import(SetList::DEAD_CODE);

};
```

"This is all self-explanatory", probably the most common phrase amongst bad communicators. So let's break this down quickly.

- `Option::AUTO_IMPORT_NAMES` will auto import the class instead of using the full qualified class name.
- `Option::PHPSTAN_FOR_RECTOR_PATH` will enable phpstan inside Rector to determine types more accurately.
- `Option::PATHS` notice we are checking the `/src`, `/config` and `/tests`, this will run rector in those directories.

I strongly recommend you run each rule or rule set individually and commit[4] it. It makes reverting bad changes back much quicker and less of a headache.

Now, let's update the `./composer.json`

[4] https://www.atlassian.com/git/tutorials/saving-changes/git-commit

```
{
  "scripts": {
    "rector-dry": "vendor/bin/rector p --dry-run --ansi",
    "rector": "vendor/bin/rector p --ansi"
  }
}
```

Now we can run `composer rector-dry` or `composer rector` instead of writing out the whole command, the `--ansi` flag will enable ANSI color output in the terminal.

let's check it's working, run `composer rector-dry`

## Easy Coding Standard

Easy Coding Standard fixes how our code looks. let's install it:

run `composer require symplify/easy-coding-standard --dev`

create a file called `./esc.php` in the root directory, now let's update it:

```
<?php

// ./esc.php

declare(strict_types=1);

return static function (ECSConfig $containerConfigurator) {
    $parameters = $containerConfigurator->parameters();
    $parameters->set(Option::PATHS, [
        __DIR__ . '/src',
        __DIR__ . '/config',
        __DIR__ . '/tests',
    ]);

    $containerConfigurator->sets([SetList::COMMON, SetList::PSR_12, SetList::SYMPLIF\
Y]);

    $ruleConfigurations = [
        [
            IncrementStyleFixer::class,
            ['style' => 'post'],
        ],
        [
            CastSpacesFixer::class,
```

```
25              ['space' => 'none'],
26          ],
27          [
28              YodaStyleFixer::class,
29              [
30                  'equal' => false,
31                  'identical' => false,
32                  'less_and_greater' => false,
33              ],
34          ],
35          [
36              ConcatSpaceFixer::class,
37              ['spacing' => 'one'],
38          ],
39          [
40              CastSpacesFixer::class,
41              ['space' => 'none'],
42          ],
43          [
44              OrderedImportsFixer::class,
45              ['imports_order' => ['class', 'function', 'const']],
46          ],
47          [
48              NoSuperfluousPhpdocTagsFixer::class,
49              [
50                  'remove_inheritdoc' => false,
51                  'allow_mixed' => true,
52                  'allow_unused_params' => false,
53              ],
54          ],
55          [
56              DeclareEqualNormalizeFixer::class,
57              ['space' => 'single'],
58          ],
59          [
60              BlankLineBeforeStatementFixer::class,
61              ['statements' => ['continue', 'declare', 'return', 'throw', 'try']],
62          ],
63          [
64              BinaryOperatorSpacesFixer::class,
65              ['operators' => ['&' => 'align']],
66          ],
67      ];
```

```
68
69     array_map(static fn($parameters) => $containerConfigurator->ruleWithConfiguratio\
70 n(...$parameters), $ruleConfigurations);
71 };
```

I think you can figure out that this will give your code a specific style, I won't explain everything in to configuration, but you can adapt it to what you like. Now, let's update the `./composer.json`

```
1 {
2   "scripts": {
3     "check-cs": "vendor/bin/ecs check --ansi",
4     "fix-cs": "vendor/bin/ecs check --fix --ansi"
5   }
6 }
```

Let's check it's working, run `composer check-cs`

## Phpstan

PHPStan focuses on finding errors in your code without actually running it. you run it, it gives you a list of problem you have to fix manually.

run `composer require phpstan/phpstan --dev`

create a new configuration file called phpstan.neon (yes, neon. personally I find this a strange choice, but I'm sure there is good reason for it... I hope)

```
1 # ./phpstan.php
2 parameters:
3     level: 8
4     paths:
5         - ./src
6         - ./config
7         - ./tests
```

Now, let's update the `./composer.json`

```
1   "scripts": {
2     "phpstan": "php -d memory_limit=256M vendor/bin/phpstan analyze"
3   }
4 }
```

Let's check it's working, run `composer phpstan`

# Entities

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Enums

What is an enum?

short for “enumeration” is a data type that represents a set of predefined, named values.

Introduced in version 8.1, enums are very useful. Here is how we can use them in an entity:

```php
<?php

// ./src/Entity/Product.php

namespace App\Entity;

#[ORM\Entity(repositoryClass: ProductRepository::class)]
class Product
{
    #[ORM\Id]
    #[ORM\GeneratedValue(strategy: 'CUSTOM')]
    #[ORM\Column(type: 'uuid', unique: true)]
    #[ORM\CustomIdGenerator(UuidGenerator::class)]
    private Uuid $id;

    #[ORM\Column(type: 'string', length: 255, enumType: ProductTypeEnum::class)]
    private ProductTypeEnum $productType;

    public function __construct()
    {
        $this->createdAt = new DateTimeImmutable();
    }

    public function getId(): null|Uuid
    {
        return $this->id;
    }

    public function getProductType(): ProductTypeEnum
    {
```

```
31          return $this->productType;
32      }
33
34      public function setProductType(ProductTypeEnum $productType): static
35      {
36          $this->productType = $productType;
37
38          return $this;
39      }
40
41  }
```

and then the `ProductTypeEnum` file:

```
1   <?php
2
3   // ./src/Enum/ProductTypeEnum.php
4
5   namespace App\Enum;
6
7   enum ProductTypeEnum : string
8   {
9       case AUTOMOTIVE = 'AUTOMOTIVE';
10      case ELECTRONICS = 'ELECTRONICS';
11      case CLOTHING = 'CLOTHING';
12      case FURNITURE = 'FURNITURE';
13      case BEAUTY = 'BEAUTY';
14  }
```

Now when we access the `$productType` in a twig template like so {{ product.productType }} it'll return an object of `ProductTypeEnum` type. you can access the `name` or `value` like so respectively. `{{ product.productType.name }}` and '{{ product.productType.value }}.

# Routing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Controllers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Forms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Factories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Repositories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Configuration Format

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Data Transfer Objects

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Messenger

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Scheduler (cron)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Event Listeners & Subscribers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Creating an API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Console Commands

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Doctrine

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## ArrayCollections

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## First & Last

These are both pretty obvious, and return the first or last element respectively. Annoyingly, both these methods returns either an object or false, in my option would be better if it was an object or null, but oh well.

```
<?php

// ./src/Entity/Product.php

  public function getFirstReview() : Review|false
    {
        $firstReview = $this->getReviews()->first();
    }
```

This will also set the internal iterator to the first element.

```php
<?php

// ./src/Entity/Product.php

    public function getLastReview() : Review|false
    {
        $lastReview = $this->getReviews()->last();
    }
```

This will also set the internal iterator to the last element.

Just a note on the collection order, `first()` or `last()` returns the element based on the order of the collection, which by default is insertion order.

## FindFirst

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## ForAll

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Filter

Want to filter out elements based on some criteria?

Let's say we want to get only the 5-star reviews of a product, inside the `Product` entity add a method like this:

```php
<?php

// ./src/Entity/Product.php
    public function getFiveStarReviews() : ArrayCollection
    {
        return $this->getReviews()->filter(function (Review $review) : bool
        {
            return $review->getRating() === 5;
        });
    }
```

and to use this method the twig template:

```
{% block body %}
<div>
   {% for review in product.fiveStarReviews %}
      <div>
         some beautiful prodcut review..
      </div
   {% endfor %}
</div>
{% endblock %}
```

Holy sh*t, That's cool!

## matching (criteria)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## exists

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Reduce

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Association Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Many-To-One (Unidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## One-To-One (Unidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## One-To-One (Bidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## One-To-One (Self-referencing):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## One-To-Many (Bidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## One-To-Many (Self-referencing):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Many-To-Many (Unidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Many-To-Many (Bidirectional):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Many-To-Many (unidirectional Self-referencing):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

### Many-To-Many (bidirectional Self-referencing):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Association Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

### Collection Default Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

### Many With More Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# QueryBuilder

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Reusable queries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Uuid

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## In

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Between

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## MemberOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## andHaving

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Features

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Init Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Docker-ising Feature

What is Docker?

Docker is a platform for developing, shipping, and running applications in isolated containers, making it easier to deploy and manage software across different environments.

Basically, you have a container to run PHP, a container to run your database, a container to compile your Javascript, etc. You can add as many containers as you like.

If we don't use docker, I'd have to explain how to install every single service we need like database, web server on each operating system, whereas if we use docker, it'll work on all operating systems, making it a much better development environment.

Before we get into dockerising our application, I'd stronly recommend installing Docker Desktop[1]. My suggestion is so strong I'm begging you, it'll make your life much easier, as containers can feel a bit invisible if you can only access them via the command line.

Symfony comes with a docker setup but in my option it's pretty much useless, I think they must be aware of this because it's even suggested in the doctumentation to use the docker set up we will be using.

Go to: https://github.com/dunglas/symfony-docker/

Clone the repo and copy the following files and directories to your projects root directory (the folder that contains your project):

- docker-compose.override.yml

[1] https://www.docker.com/products/docker-desktop/

- docker-compose.prod.yml
- docker-compose.yml
- .dockerignore
- Dockerfile
- docker/

run `composer config --json extra.symfony.docker 'true'` to enable the Docker support of Symfony Flex. Now when ever you install supported package the docker configuration files will be updated automaticlly. Check out the Support for Extra Services[2]

If you look in the `./docker-compose.yaml` file, you'll notice, that the database configuration is missing, this is because it's not included by default.

If you need a database, install doctrine if you don't already have it installed, run `composer require symfony/orm-pack`

Next run `rm symfony.lock` then `composer symfony:sync-recipes --force --verbose`

There is a slight problem here, because Symfony felx doesn't support PHP config files, you might get the last command above generating a bunch of YAML files in your config directory. Not much we can do about this but copy the respective and relevent data from the yaml to PHP files and hope that Symfony pull their finger out an introduce PHP config support. Crazy, I know! a PHP framework that uses PHP configs! But, I'm sure there is good reason it's not yet supported.

Now go to your `./.env.local` file and update the database URL:

```
DATABASE_URL="postgresql://app:!ChangeMe!@database:5432/app?serverVersion=13&charset\
=utf8"
```

Okie dokie, now build the docker containers `docker compose build --no-cache` and then finally start the application with `docker compose up -d`

Open the Docker Desktop application and you should see under the "Containers" section your application, go to the PHP container, probably called "php-1" or something like this.

Go to the PHP containers "logs" and you should see an output "ready to handle connections" if you do then we are ready to tango.

Go to the PHP containers "terminal" and run `bin/console d:s:u -f` this will update your database schemea. If you make any changes to your Entity directory, you will have to run this command to update your database.

Now finally go to `https://localhost/` in your browser and you should see a Symfony welcome page.

If you need to stop the application run `docker compose down`

[2] https://github.com/dunglas/symfony-docker/blob/main/docs/extra-services.md

# UserInterface Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## UserInterface - Current User Feautre

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Admin Panel Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Registration Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Login Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Roles Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Form Filter Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Entity Event Subscriber Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Entity Voter Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Localisation Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Mutiplue Currencies Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Timezones Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Email Service Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Email Styling Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Email Message Queue Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Email Schedule Feature (cron)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## File Upload Feature

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## External API Feature (Dictionary API)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Payment Gateway Feature (Stripe API)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Frontend

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Webpack Encore

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Switching to Sassy Cascading Style Sheets (SCSS)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Bootstrap

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Stimulus

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Vue

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

### Vue MPA (Multi-page Application)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## Vue SPA (Single Page Application)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# React

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

## React MPA (Multi-page Application)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide

# Feedback

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/symfony-6-a-practical-guide