Getting started with Symfony

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. HTTP flow

III. Tests

IV. Twig

V. Webpack Encore

VI. Doctrine

VII. Forms

What is Symfony?

SensioLabs
Créateur de Symfony

Training program

I. The Symfony project

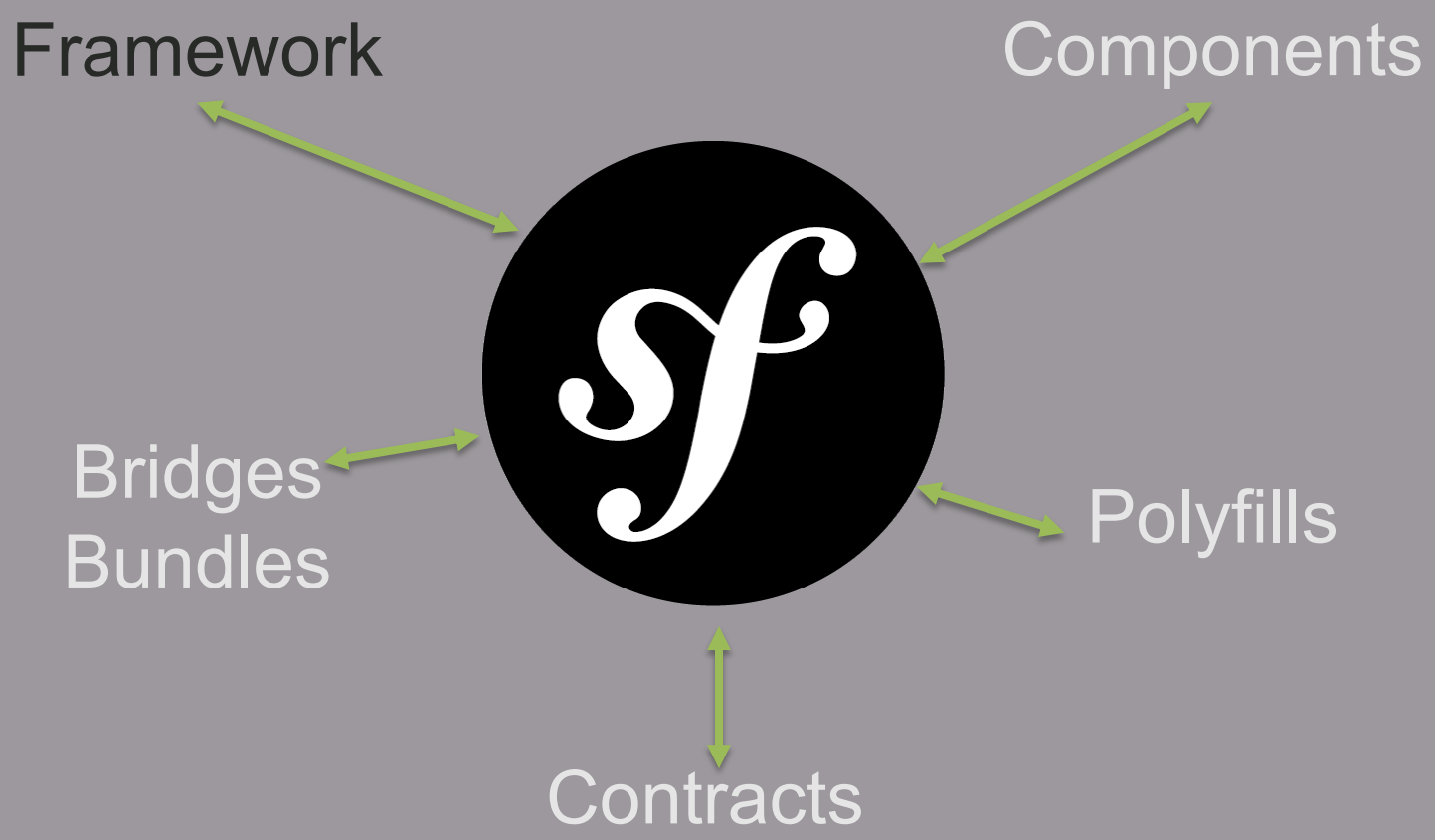
II. Environment

The Symfony project

Symfony project

- Created by Fabien Potencier in 2004
- Maintained by Symfony and its **community**
- Open-source: MIT
- https://symfony.com/contributors

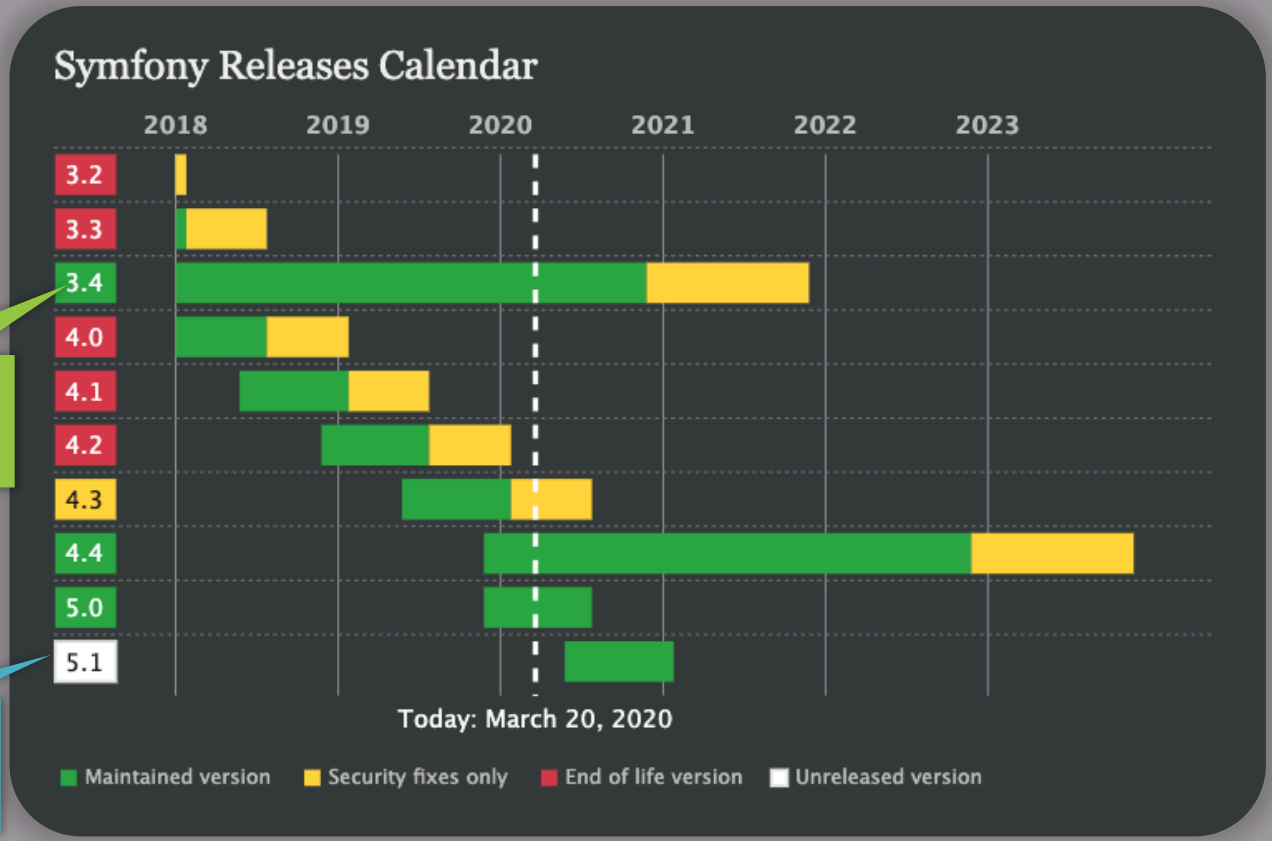
Symfony architecture



Symfony components

- Standalones
- Decoupled
- Cohesives
- https://symfony.com/components

Symfony lifecycle



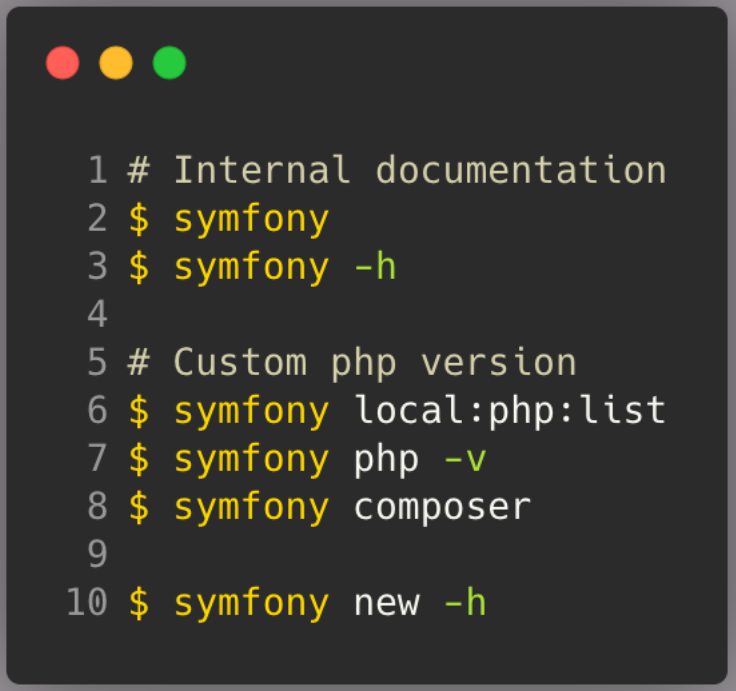
LTS Releases (3.4, 4.4, 5.4)

Standard Releases (5.0, 5.1, 5.2, 5.3)

Official website: https://symfony.com/releases

Environment

Symfony CLI



```
# Internal documentation
$ symfony
$ symfony -h

# Custom php version
$ symfony local:php:list
$ symfony php -v
$ symfony composer

$ symfony new -h
```

Symfony CLI in your project

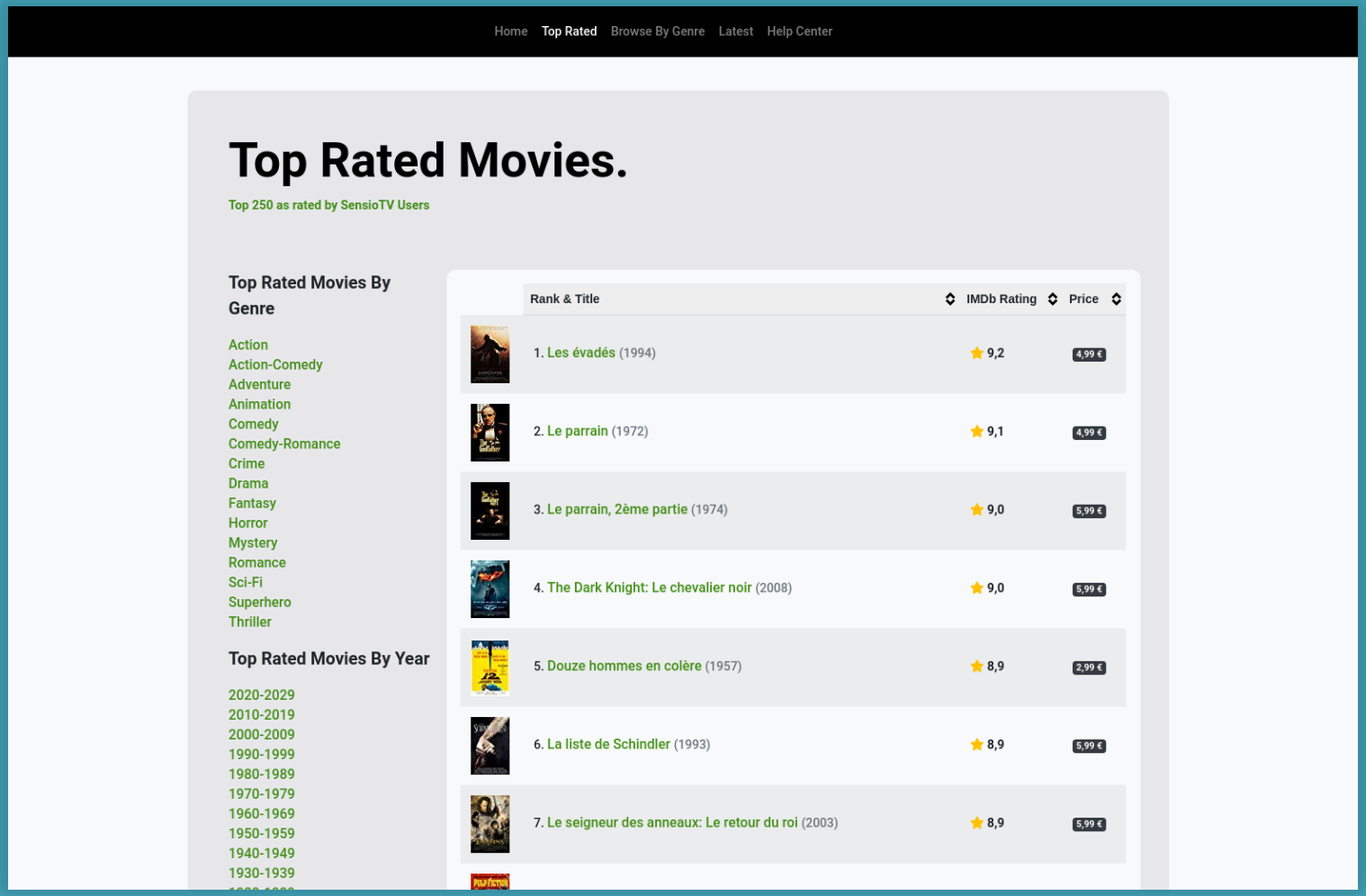
```
$ symfony console
$ symfony local:check:requirements
$ symfony local:check:security

# Start the PHP server
$ symfony serve -d

# If you need to execute long-running commands
$ symfony run -d your_command

$ symfony server:status
$ symfony server:log

$ symfony server:stop
```

Installation

Exercises

About the project

Exercises


Home
Top Rated
Browse By Genre
Latest
Help Center
Top Rated Movies.
Top 250 as rated by SensioTV Users
Top Rated Movies By Genre
Action
Action-Comedy
Adventure
Animation
Comedy
Comedy-Romance
Crime
Drama
Fantasy
Horror
Mystery
Romance
Sci-Fi
Superhero
Thriller
Top Rated Movies By Year
2020-2029
2010-2019
2000-2009
1990-1999
1980-1989
1970-1979
1960-1969
1950-1959
1940-1949
1930-1939
Rank & Title
IMDb Rating
Price
1. Les évadés (1994) 9,2 4,99 €
2. Le parrain (1972) 9,1 4,99 €
3. Le parrain, 2ème partie (1974) 9,0 5,99 €
4. The Dark Knight: Le chevalier noir (2008) 9,0 5,99 €
5. Douze hommes en colère (1957) 8,9 2,99 €
6. La liste de Schindler (1993) 8,9 5,99 €
7. Le seigneur des anneaux: Le retour du roi (2003) 8,9 5,99 €

Exercises

1. Use Symfony CLI to create a new SF5 project
    1. Create a project based on website-skeleton
    2. Ask your trainer whether you work with Git or not

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/

https://symfony.com/what-is-symfony

https://github.com/symfony/symfony

https://symfony.com/community

https://symfony.com/releases

First steps with Symfony

SensioLabs
Créateur de Symfony

Training program

I. Request – Response flow

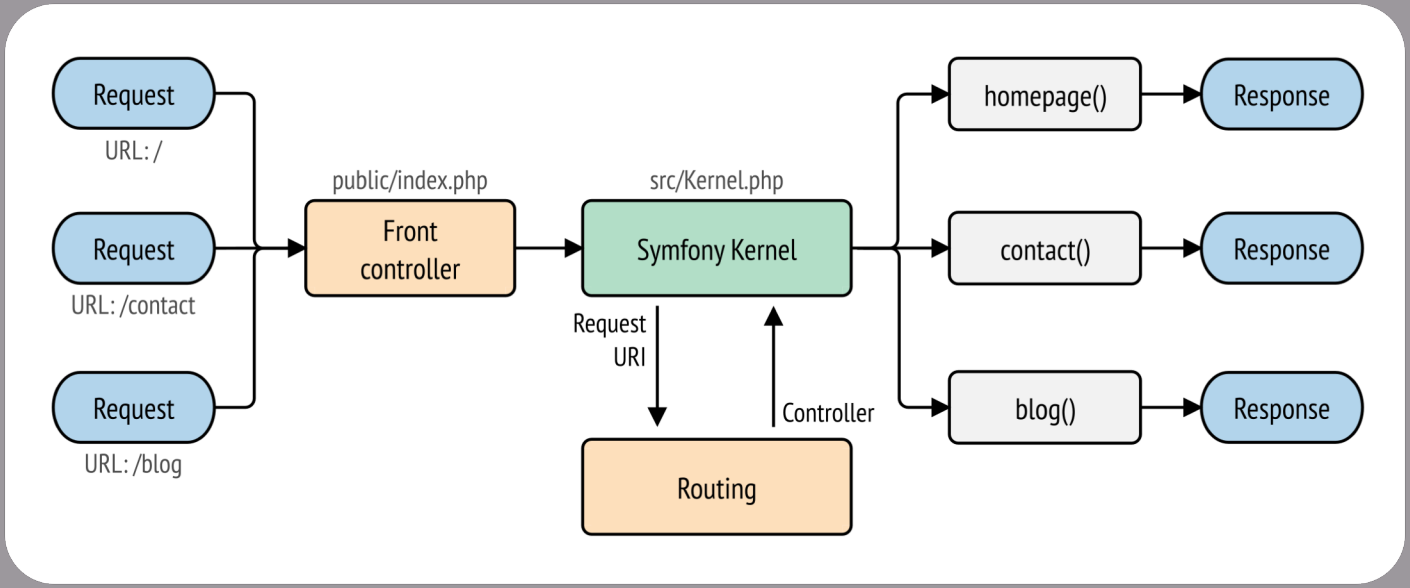
II. First web page

III. Routing

IV. Controllers

Request – Response flow

Overview


Request
URL: /
Request
URL: /contact
Request
URL: /blog
public/index.php
Front controller
src/Kernel.php
Symfony Kernel
Request URI
Controller
Routing
homepage()
contact()
blog()
Response
Response
Response

First web page

MakerBundle

- Interactive command to create some code

```
$ symfony composer require maker --dev

# Let's have a look
$ symfony console list make
```

First page – easy way

```
$ symfony console make:controller

 Choose a name for your controller class (e.g. FiercePuppyController):
 > BookController

 created: src/Controller/BookController.php
 created: templates/book/index.html.twig


  Success!


 Next: Open your new controller class and add some pages!

 # Open your browser on https://localhost:8000/book
```

Routing

Routing annotations in action

```php
use Symfony\Component\Routing\Annotation\Route;

/**
 * @Route("/book", name="book_")
 */
class BookController extends AbstractController
{
    /**
     * @Route("/{page<\d+>?1}", name="list", methods={"GET"})
     */
    public function index(int $page)
    {
        // ...
    }
}
```

Debugging routes

```
1 $ symfony console debug:router
2 $ symfony console debug:router book_list
3 $ symfony console debug:router --show-controllers
4 $ symfony console router:match /book/1
```

More about routes

- Documentation : https://symfony.com/doc/current/routing.html

Controllers

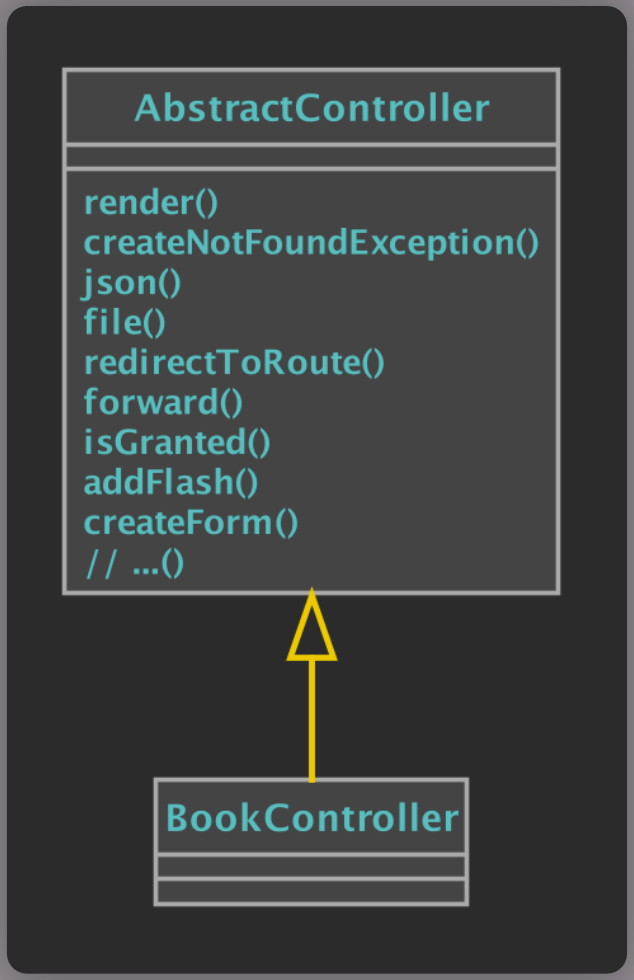
Generated controller

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class BookController extends AbstractController
{
    /**
     * @Route("/book", name="book")
     */
    public function index()
    {
        return $this->render('book/index.html.twig', [
            'controller_name' => 'BookController',
        ]);
    }
}
```

Role of a controller

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

/**
 * @Route("/book", name="book_")
 */
class BookController extends AbstractController
{
    /**
     * @Route("/{id}", name="update", methods={"GET", "POST"})
     */
    public function update(int $id, Request $request): Response
    {
        // Handle a request
        // Return a response

        // In addition, use some *services* to do your job

        return new Response();
    }
}
```

Controller helpers



More about controllers

- Documentation: https://symfony.com/doc/current/controller.html

Exercises

1. Create a new home page
2. Create a page to display movie description
    1. For now, don’t care about the template, focus on the controller and its route.

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/routing.html

https://symfony.com/doc/current/controller.html

Introduction to functional tests

SensioLabs
Créateur de Symfony

Training program

I. Concepts

II. PHPUnit

III. Write your first smoke test

IV. Data providers

Concepts

What's functional testing?

- Functional testing focuses on **testing the interface of the application** to ensure **that all user requirements** for a properly working application **are met**.

Follow the F.I.R.S.T rule

- Tests suite must run as **fast** as possible
- Tests must be:
  - **Isolated** from each other
  - **Repeatable** indefinitely
  - **Self-explanatory**
  - Must come **Timely**

PHPUnit

What is PHPUnit?

- A unit testing framework for PHP
- Written by Sebastian Bergmann
- Allow to write and run unit tests suite

Official website: phpunit.de

Symfony PHPUnitBridge

- A **modified PHPUnit script**
- **Provides helpers** to especially test a Symfony application:
  - List of deprecations
  - A functional testing kit (BrowserKit, Crawler, Profiler...)

Symfony doc: PHPUnit Bridge

Write your first smoke test

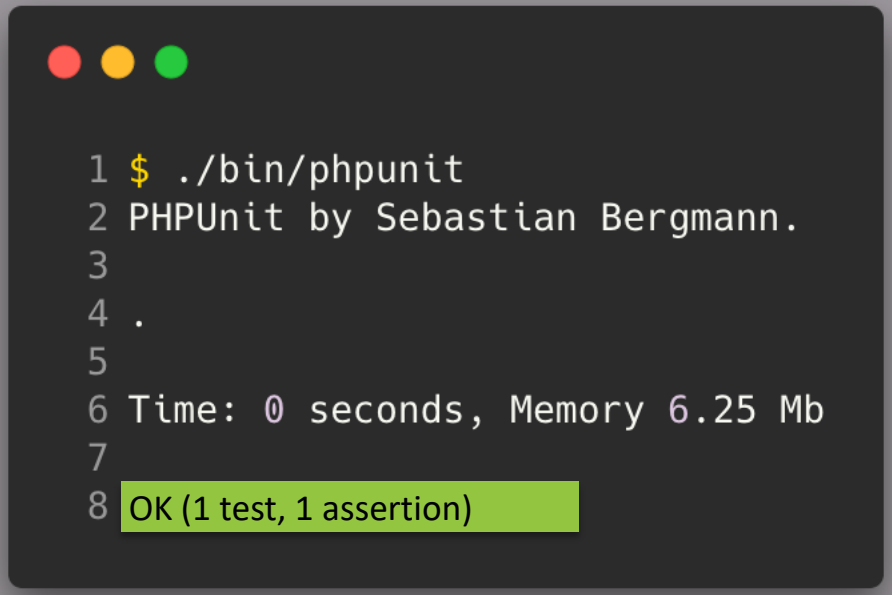
Controller class

```php
class DefaultController extends AbstractController
{
    /**
     * @Route("/", name="homepage", methods={"GET"})
     */
    public function homepage(): Response
    {
        return $this->render('default/homepage.html.twig');
    }
}
```

Test class

```php
namespace App\Tests\Controller;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

/**
 * @group functional
 */
class DefaultControllerTest extends WebTestCase
{
    public function testHomepageIsSuccessful(): void
    {
        $client = self::createClient();
        $client->request('GET', '/');
        self::assertResponseIsSuccessful();
    }
}
```

Execute tests with PHPUnitBridge



Output when a test fails

```
$ .bin/phpunit
PHPUnit by Sebastian Bergmann.

.F

There was 1 failure:
1) DefaultControllerTest::testHomepageIsSuccessful
Failed asserting that the Response is successful.

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

Data Providers

What are data providers?

- Functions providing data sets
- Run a single test several times with several input values

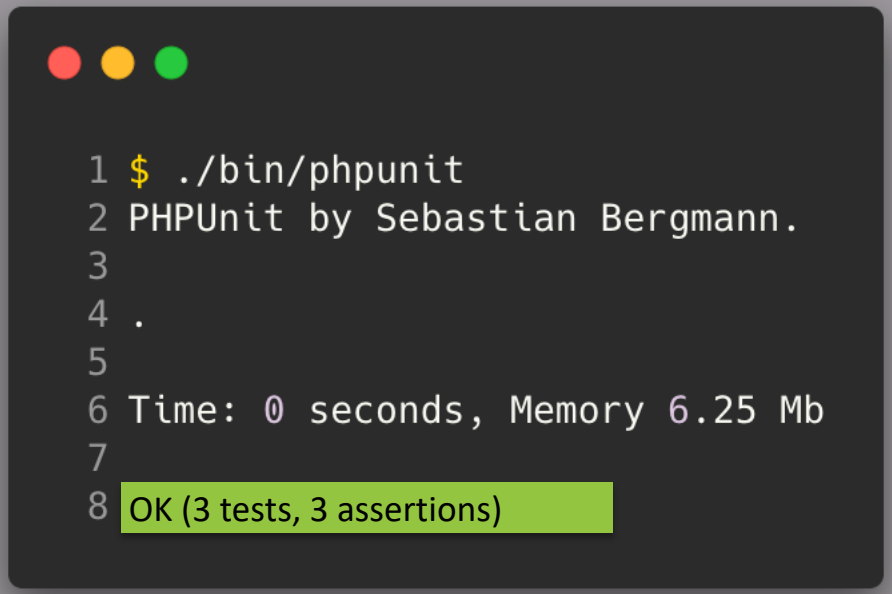
Test with data providers

```php
class RoutingTest extends WebTestCase
{
    public function provideUrisWithStatusCodes(): \Generator
    {
        yield ['/', Response::HTTP_OK];
        yield ['/login', Response::HTTP_OK];
        yield ['/logout', Response::HTTP_FOUND];
    }

    /**
     * @dataProvider provideUrisWithStatusCodes
     */
    public function testApplicationRoutes(string $uri, int $expectedStatusCode): void
    {
        $client = static::createClient();
        $client->request(Request::METHOD_GET, $uri);
        self::assertResponseStatusCodeSame($expectedStatusCode);
    }
}
```

Prepare your tests

```php
class DefaultControllerTest extends WebTestCase
{
    /** @var KernelBrowser */
    public static $webClient;

    public static function setUpBeforeClass(): void
    {
        // Do something before the tests of this class
        // For example:
        self::$webClient = static::createClient();
    }

    protected function setUp(): void
    {
        // Do something before each test
    }

    // ...
}
```

Test results



Exercises

1. Write your functional tests to cover your app.

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/testing.html

https://phpunit.readthedocs.io/fr/latest/

Twig

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. Basic syntax

III. Dot syntax

IV. Control structures

V. Template relationships

VI. Debug

VII. Symfony helpers

Introduction

What is Twig

- Template engine for PHP
- Symfony official library
- Why?
  - Concise syntax
  - Extensible
  - Many helpers available out of the box
  - Easy to learn

Twig vs. PHP

```
1 {{ variable }}
```

```
1 <?php
2
3 echo htmlspecialchars(
4   $variable,
5   ENT_QUOTES,
6   'UTF-8'
7 );
8 ?>
```

Basic syntax

Concise syntax

```twig
{# Comment here… #}
{% do something… %}
{{ display something… }}

{# Combine with filters and function: #}
{{ something|some_filter }}
{{ some_function(something) }}
```

```twig
<p>
    {# Display some sentence #}
    {% set sentence = 'Hello, ' ~ random(['John', 'Tom', 'Paul']) %}
    {{ sentence|upper }}
</p>
```

Syntax reference

- Official documentation:
  - https://twig.symfony.com/doc/3.x/
- Syntax:
  - Tags
  - Filters
  - Functions
  - Operators, tests, misc.

Dot syntax

Dot syntax

```twig
{{ article.title }}
```

```php
echo $article['title'];
echo $article->title;
echo $article->title();
echo $article->getTitle();
echo $article->isTitle();
echo $article->hasTitle();

# Stop with the first allowed access
```

Control structures

Conditions

```
1 {% if product.stock > 10 %}
2    Available
3 {% elseif product.stock > 0 %}
4    Only {{ product.stock }} left!
5 {% else %}
6    Sold-out!
7 {% endif %}
```

See more: https://twig.symfony.com/doc/3.x/tags/if.html

Loops

```twig
{% for post in posts %}
    <h2>N°{{ loop.index }}: {{ post.title|title }}</h2>
    <div>
        {{ post.body }}
    </div>
{% else %}
    <p>No published posts yet.</p>
{% endfor %}
```

See more: https://twig.symfony.com/doc/3.x/tags/for.html

Operators

Operators

- Literals
- Math
- Logic
- Comparisons
- Containment
- Tests
- Others

https://twig.symfony.com/doc/3.x/templates.html#expressions

Template relationships

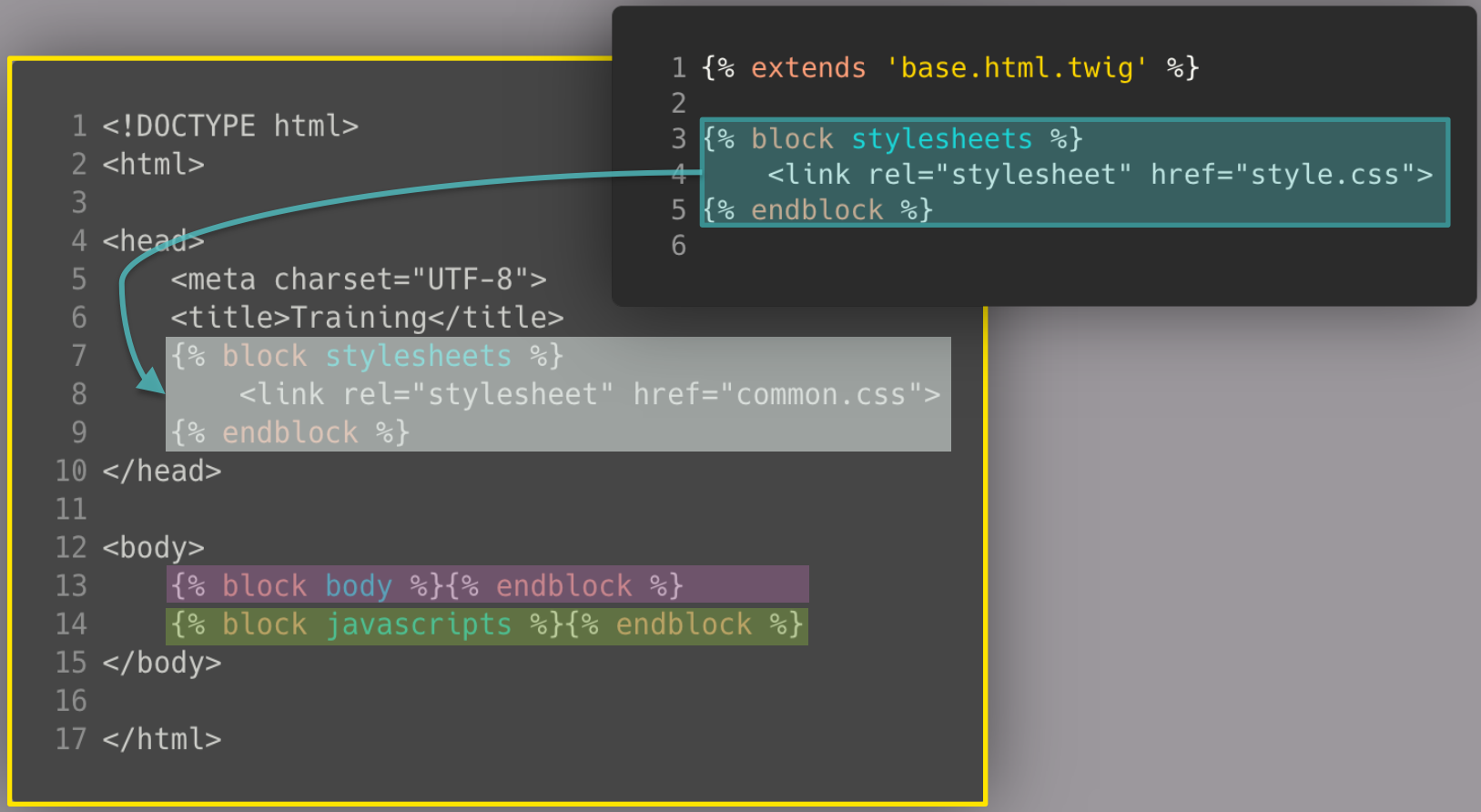
Template inclusion

```
1 <header>
2     {{ include('_menu.html.twig') }}
3 </header>
```

Template scopes

```
1 <header>
2   {{ include('_menu.html.twig', {param: '…'}, with_context = false) }}
3 </header>
```

Template inheritance



Use cases

```
1 {% extends 'base.html.twig' %}
2
3 {# Remove the parent's content #}
4 {% block stylesheets %}
5 {% endblock %}
6
7 {# Replace the parent's content #}
8 {% block stylesheets %}
9    <link rel="stylesheet" href="file.css">
10 {% endblock %}
11
12 {# Append the parent's content #}
13 {% block stylesheets %}
14    {{ parent() }}
15
16    <link rel=”stylesheet”>
17  {% endblock %}
```

Block syntax

```
1 {% extends 'base.html.twig' %}
2
3 {% block title       'Latest Posts' %}
4 {% block description '...' %}
5
6 {% block body %}
7     <h2>Latest posts</h2>
8 {% endblock body %}
```

Debug

Dumping variables

```twig
{% set names   = ['John', 'Tom', 'Paul'] %}
{% set numbers = 1..5 %}

{{ dump(names, numbers) }}
{% dump names %}

{{ dump() }}
{% dump %}
```

Commands

```
# Check Twig file syntax
$ symfony console lint:twig templates/

# List Twig active features (filters, functions, …)
$ symfony console debug:twig

# Dump the exhaustive configuration of Twig in your project
$ symfony console debug:config twig
```

Exercises

1. Use given mockups to fulfill your templates.
2. Create a navbar in a specific Twig file, still without links.
3. In your “movie_details” page:
    1. Create an array in your controller to set fake movie data.
    2. Use Twig to display the movie title, its released date and its genres.

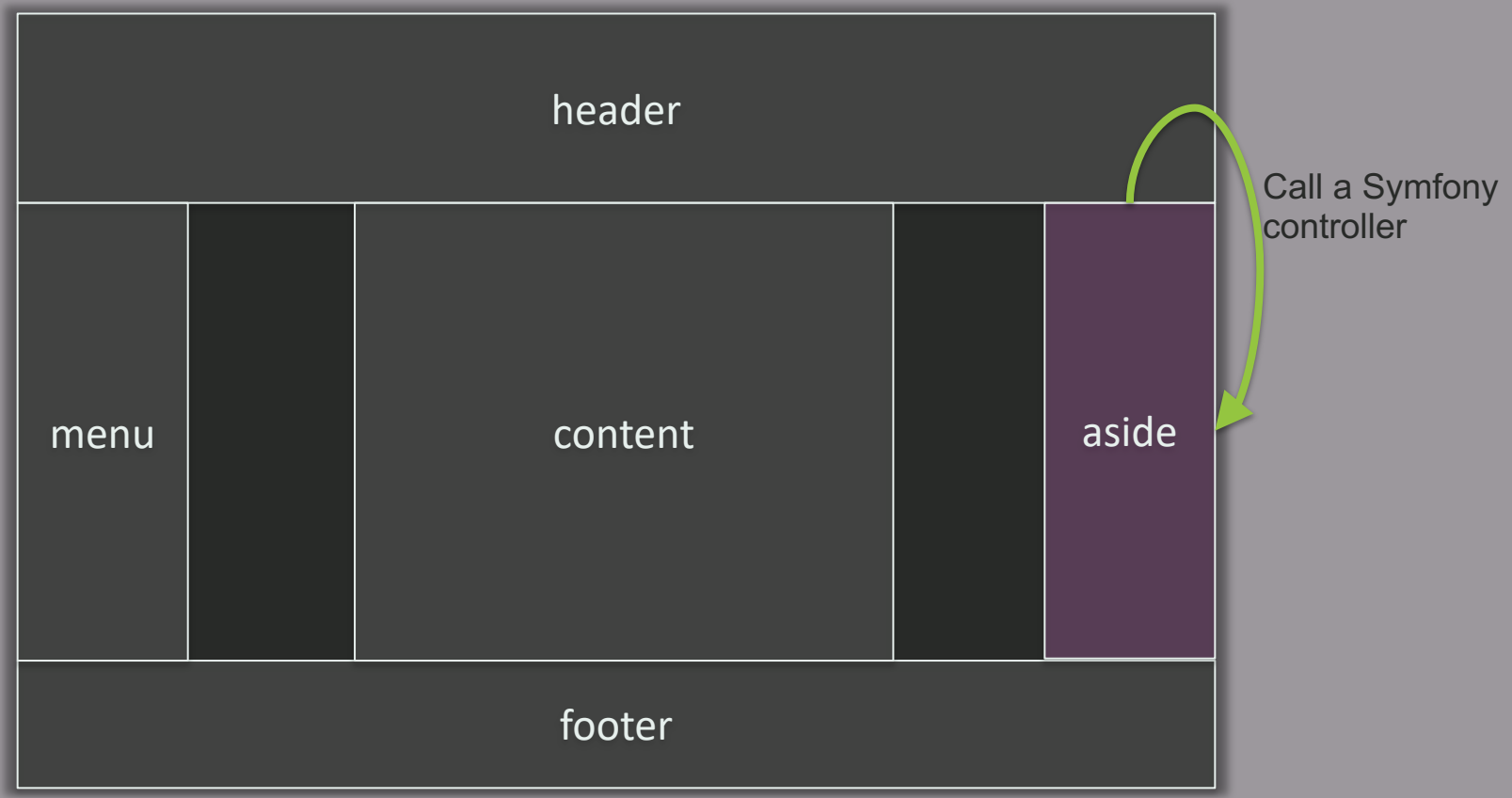
Symfony helpers

Generating URLs

```twig
<a href="{{ path('book_show', {'id': book.id}) }}">
  Show this book (internal link)
</a>
```

```twig
<a href="{{ url('homepage') }}">
  Visit our website (external link)
</a>
```

Split your view in parts



Fragments

```
1 {# Use a route name… #}
2 {{ render(url('book_latest_fragment')) }}
3
4 {# or a controller name. #}
5 {{ render(controller('App\\Controller\\BookController::latest', {'max': 3})) }}
6
7 {# ESI caching strategy, using HTTP cache for the fragment. #}
8 {{ render_esi(controller('App\\Controller\\BookController::latest', {'max': 3})) }}
```

Exercises

1. Update the navbar to create links.

SensioLabs

Créateur de Symfony

Resources

https://twig.symfony.com/doc/3.x/templates.html

https://twig.symfony.com/doc/3.x/#reference

Webpack Encore

SensioLabs
Créateur de Symfony

Training program

I. Webpack Encore introduction

II. Yarn

III. Installation and configuration

IV. Webpack Encore common features

V. Webpack Encore optimizations

Webpack Encore introduction

What is Webpack Encore?

- Symfony library that wraps **Webpack**
- **Process, compile, and package** any resources or assets
- A **powerful API** to solve most common Webpack use cases for PHP

Yarn

What is Yarn?

- A **JS package manager**
- Developed by **Facebook** in 2016
- Similar to NPM, **same NPM registry**
- Focused on **improving performance** and issues around versioning

Best practice

- Yarn should be **installed globally** in your system, with the **classic version 1.X**
- For the second one, make a **per-project install** with the **Yarn 2.x binary**

Official website: https://yarnpkg.com/

Prerequisites: NPM

```
$ curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash - sudo apt-get install -y nodejs
```

Official website: https://nodejs.org/en/download/

Install and update dependencies

```
# Install Yarn
$ curl -sS https://dl.yarnpkg.com/debian/pub key.gpg | sudo apt-key add -
$ sudo apt install yarn

# Install and update Yarn packages
$ yarn install
$ yarn upgrade

# Test it
$ yarn -v
$ yarn -h
```

Official website: https://yarnpkg.com/getting-started/install

Installation and configuration

Webpack Encore installation

- Webpack and Encore are both Node libraries
- In a Symfony project:
  1. Install the Symfony bundle
  2. Install the JavaScript library

Basic installation in Symfony

```
# Install and configure the bundle
$ symfony composer require encore

# Install Webpack Encore JS modules
$ yarn install
```

Webpack.config.js entrypoint

```
1  // Non exhaustive Webpack configuration
2
3  var Encore = require('@symfony/webpack-encore');
4
5  Encore
6      // Directory where compiled assets will be stored
7      .setOutputPath('public/build/')
8
9      // Public path used by the web server to access the output path
10     .setPublicPath('/build')
11
12     // One entry for each "page" of your app, in order to map all of its .css and .js related files
13     .addEntry('app', './assets/js/app.js')
14
15     // Hash filenames to cache your assets through HTTP
16     .enableVersioning(Encore.isProduction())
17
18 module.exports = Encore.getWebpackConfig();
```

Compile assets

```
# Compile all assets in a dev environment
$ yarn encore dev

# Recompile assets automatically as soon as they change
# (Note that the following command remains running to watch resources)
$ yarn encore dev --watch

# Compile all assets and optimize production build (minify assets ...)
$ yarn encore production --progress

# Shortcuts:
$ yarn dev
$ yarn watch
$ yarn build
```

Restart Encore each time you update your config file.

Using Webpack assets in Twig

```twig
{# CSS #}
{% block stylesheets %}
    {# 'app' must be declared as an entry in webpack.config.js #}
    {{ encore_entry_link_tags('app') }}
{% endblock %}

{% block content %}
    {# Asset function use to require a specific asset built by Webpack #}
    <img src="{{ asset('build/images/sf-logo.png') }}" alt="Symfony logo">
{% endblock %}

{# JS #}
{% block javascripts %}
    {# 'app' must be declared as an entry in webpack.config.js #}
    {{ encore_entry_script_tags('app') }}

    {# Result to:
            <script src="/build/app.js"></script>
            <script src="/build/runtime.js"></script>
            (Runtime useful to give a same object for a same module use by few entities)
    #}
{% endblock %}

{# The encore_entry_link_tags() and encore_entry_script_tags() functions read the exact filenames to render from an entrypoints.json file, automatically created in build/ directory #}

```

Webpack Encore common features

jQuery and other JS modules

```
1 $ yarn add jquery --dev
```

```
1 // Set Jquery for all Webpack scripts:
2 Encore
3     .autoProvidejQuery();
4
5 // And global access to jQuery outside Webpack:
6 import $ from 'jquery';
7
```

https://symfony.com/doc/current/frontend/encore/legacy-applications.html

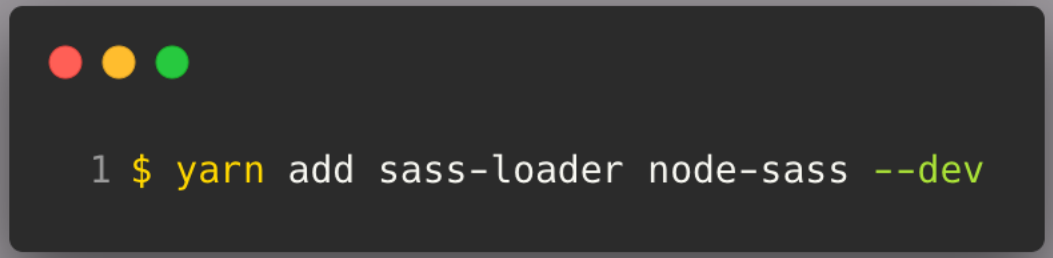
Bootstrap CSS and JS

```
$ yarn add bootstrap popper.js --dev
```

```
// Usage in a Scss file

// Import all of the styles
@import "~bootstrap/scss/bootstrap";

// Import a part of Bootstrap styles
@import "~bootstrap/scss/alert";
```

```
// Usage in a JS file

// Import the Bootstrap JS files
import 'bootstrap';
```

Preprocessors and source maps



```
$ yarn add sass-loader node-sass --dev
```

```
// If you want to use preprocessors, you will need to install the loader according to your needs:

Encore
    .enableSassLoader()
    .enableLessLoader()
    .enableStylusLoader()
    // Browsers can access to the original code with:
    .enableSourceMaps(!Encore.isProduction())

```

Copying images

```
.copyFiles({
    from: './assets/images',
    // optional target path, relative to the output dir
    to: 'images/[path][name].[ext]',
    // if versioning is enabled, add the file hash too
    to: 'images/[path][name].[hash:8].[ext]',
})

```

Do not forget the build/ directory in your asset() functions. You can also import images in your JS files and use relative paths.

Exercises

1. Use given assets to enhance your existing templates:
    1. Install and link Bootstrap 4
    2. Use Bootstrap scripts
        1. (Make sure jQuery is available)
    3. Use a given CSS file
        1. (Pay attention to your entrypoint)
    4. Configure all your images to be managed by Webpack Encore

Webpack Encore optimizations

Webpack dev server

- Your browser will be updated, each time you make an asset modification
- Take a look on new paths in entrypoints.json

https://symfony.com/doc/current/frontend/encore/dev-server.html

SensioLabs

Créateur de Symfony

Resources

https://packagist.org/packages/symfony/webpack-encore-bundle

https://yarnpkg.com/getting-started/install

https://symfony.com/doc/current/frontend.html

https://symfony.com/doc/current/frontend/encore/simple-example.html

Doctrine

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. Entities + mapping

III. Migrations

IV. Requests in database

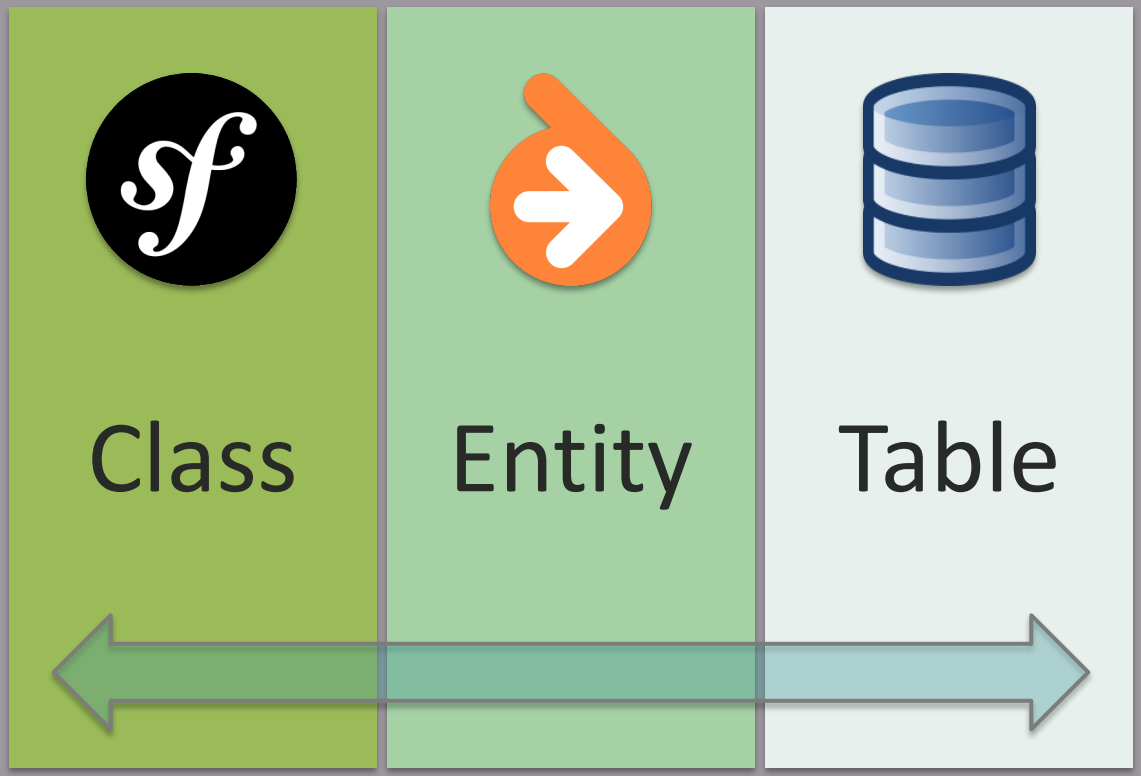
V. Relationships

Introduction

Introduction

- Storage into a database
- ORM:
  - DBAL
  - DataMapper

DataMapper



Configuration

- Open:
  - .env
  - config/packages/doctrine.yaml

Compatibility

- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL
- SAP SQL Anywhere
- SQLite
- Drizzle

Database layer

- doctrine/dbal:
  - Doctrine database abstraction and access layer
  - Uses PDO API
  - Already installed with doctrine/orm ☺

  - Double-check your PDO extensions though.

Create your database

```
# check first
$ symfony console doctrine:schema:validate --skip-mapping

$ symfony console doctrine:database:create
```

Delete a whole database

```
$ symfony console doctrine:database:drop --force
```

Exercises

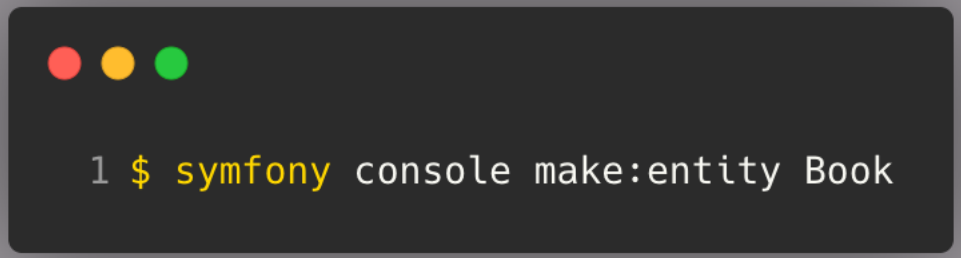
1. Configure Doctrine to access the database. Let's assume we will use a SQLite storage, located in var/data/main.db
2. Check the connection through Symfony console

Entities + mapping

Entities

- Configuration related to a PHP class
- Available formats:
  - Annotations (recommended)
  - Yaml
  - XML
  - PHP

New entity



Migrations

About

- Safe upgrade/downgrade of table definitions
- Operations stored in a single (versioned) file:
  - src/Migrations/ by default
- More here: https://www.doctrine-project.org/projects/doctrine-migrations/en/current/index.html

Check your status

```
# Check connection and db sync
$ symfony console doctrine:schema:validate

# Check your mapped entities
$ symfony console doctrine:mapping:info

# Overview about migrations
$ symfony console doctrine:migrations:status
```

New migration to be tested

```
$ symfony console make:migration

# Check your migration file
# If not satisfied, gently remove that file 😌
```

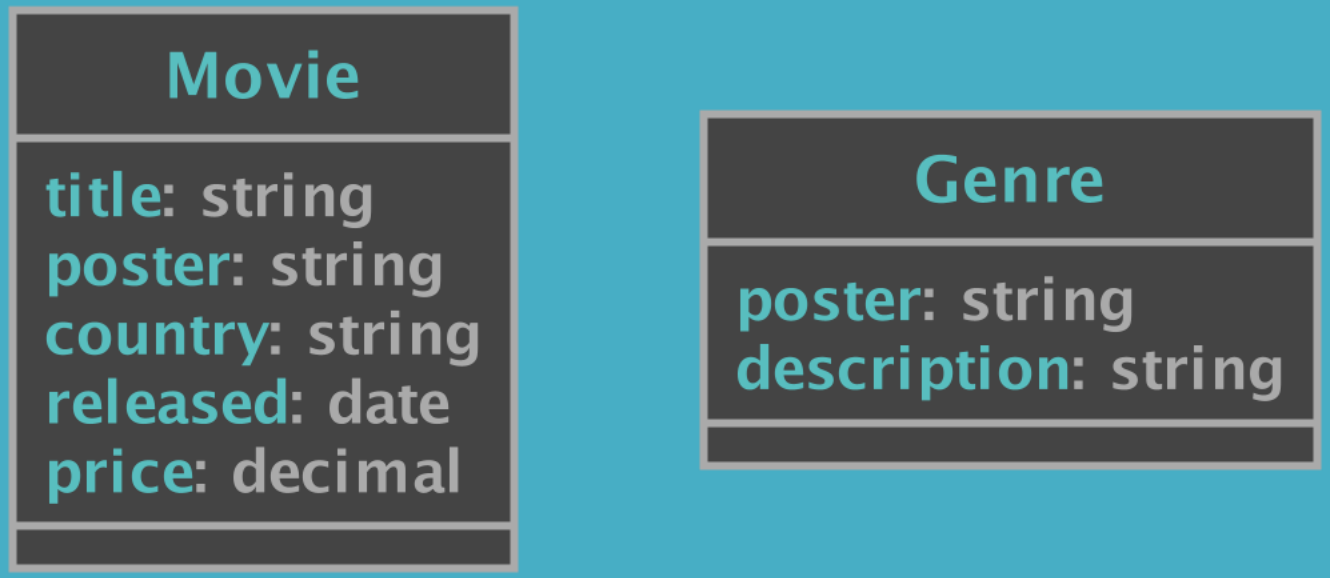
Apply migration

```
$ symfony console doctrine:migrations:migrate

# If something's wrong, then
$ symfony console doctrine:migrations:migrate down
```

Exercises

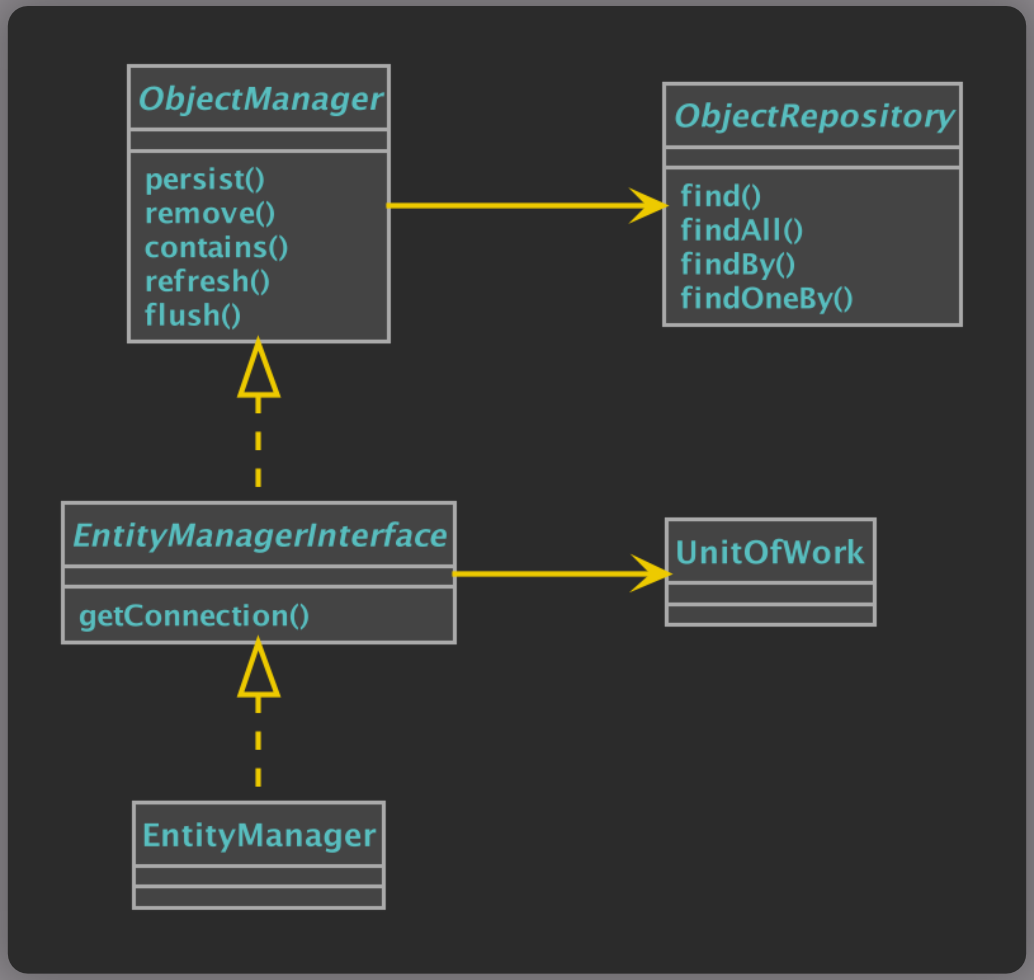
1. Create new entities:
    1. Movie
    2. Genre
2. Generate a new migration.
3. Store both tables in your database.

Exercises


Movie
title: string
poster: string
country: string
released: date
price: decimal
Genre
poster: string
description: string

Requests in database

Overview


ObjectManager
persist()
remove()
contains()
refresh()
flush()
ObjectRepository
find()
findAll()
findBy()
findOneBy()
EntityManagerInterface
getConnection()
UnitOfWork
EntityManager

New data

```
1
2 /**
3  * @Route("/create", name="book_creation", methods={"GET"})
4  */
5 public function create(ObjectManager $manager): Response
6 {
7     $book = (new Book())
8         ->setIsbn('0-86140-324-X')
9         ->setTitle('The Color of Magic')
10    ;
11
12    $manager->persist($book);
13    $manager->flush();
14
15
16    return $this->redirectToRoute('homepage');
17 }
```

Retrieve data

```php

/**
 * @Route("/list", name="book_list", methods={"GET"})
 */
public function all(BookRepository $repository): Response
{
    $books = $repository->findAll();

    return $this->render('book/list.html.twig', [
        'books' => $books,
    ]);
}
```

Custom requests

- Build queries in your repository:
  - DQL
  - QueryBuilder
  - SQL

Custom DQL request

```php
class BookRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Book::class);
    }

    public function findBookWithAuthors(string $isbn): ?Book
    {
        $dql = <<< DQL
            SELECT b, a
              FROM App\Entity\Book b
              JOIN b.authors a
              WHERE b.isbn = :isbn
        DQL;

        return $this->getEntityManager()
            ->createQuery($dql)
            ->setParameter('isbn', $isbn)
            ->getOneOrNullResult()
        ;
    }
}
```

Exercises

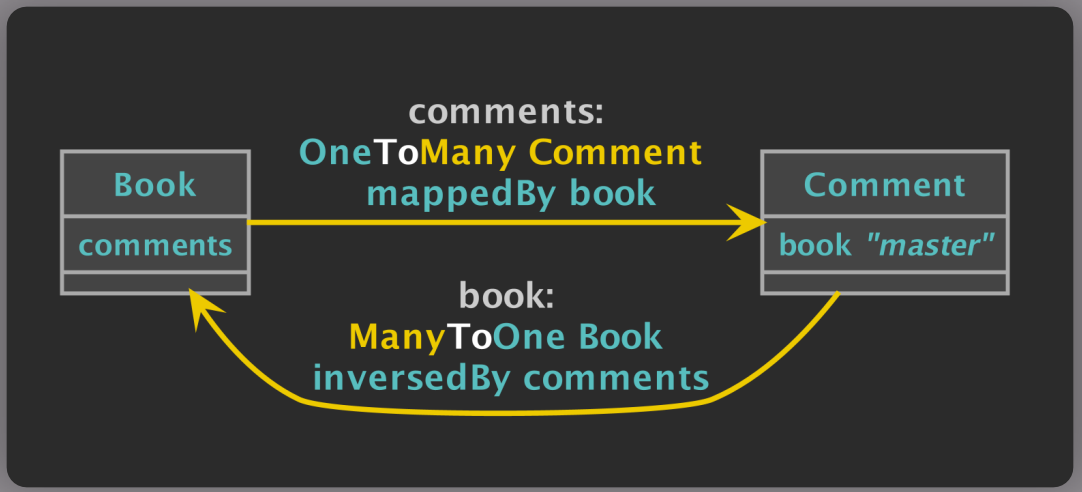
1. Update your “movie_details” page to retrieve movies from your database.
2. Update your “admin_movie_index” page to retrieve the list of movies as well.
3. (optional) Update the last section of your homepage to pick only 6 newest movies from your database.

Relationships

About

- Don’t care about foreign keys
- Class property as a relationship
- Relationships:
  - OneToOne
  - ManyToMany
  - OneToMany
  - ManyToOne

Bidirectional relationship



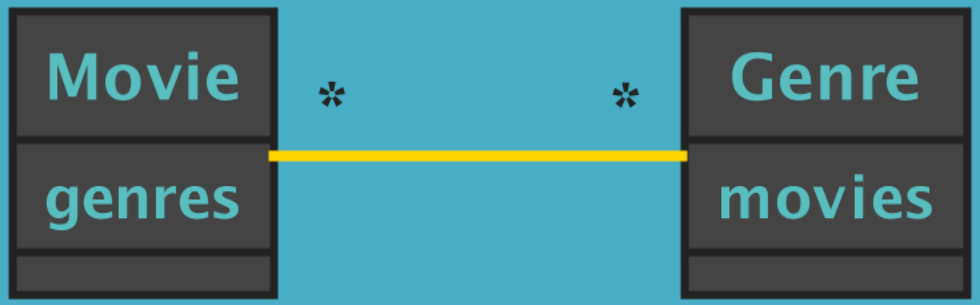
Create a new relationship

```
$ symfony console make:entity Book

# Create a field by using a wizard as a type 🧙
# Don't forget to apply your migration
```

Exercises

1. Create a bidirectional relationship between movie and genre entities.
2. Generate a new migration.
3. Store both tables in your database.
4. Check the current status of your migrations.

Exercises



SensioLabs

Créateur de Symfony

Resources

https://www.doctrine-project.org/projects/doctrine-orm/en/current/index.html

Forms

SensioLabs
Créateur de Symfony

Training program

I. Concepts

II. Building a form

III. Customize a form type

IV. Handle submitted data

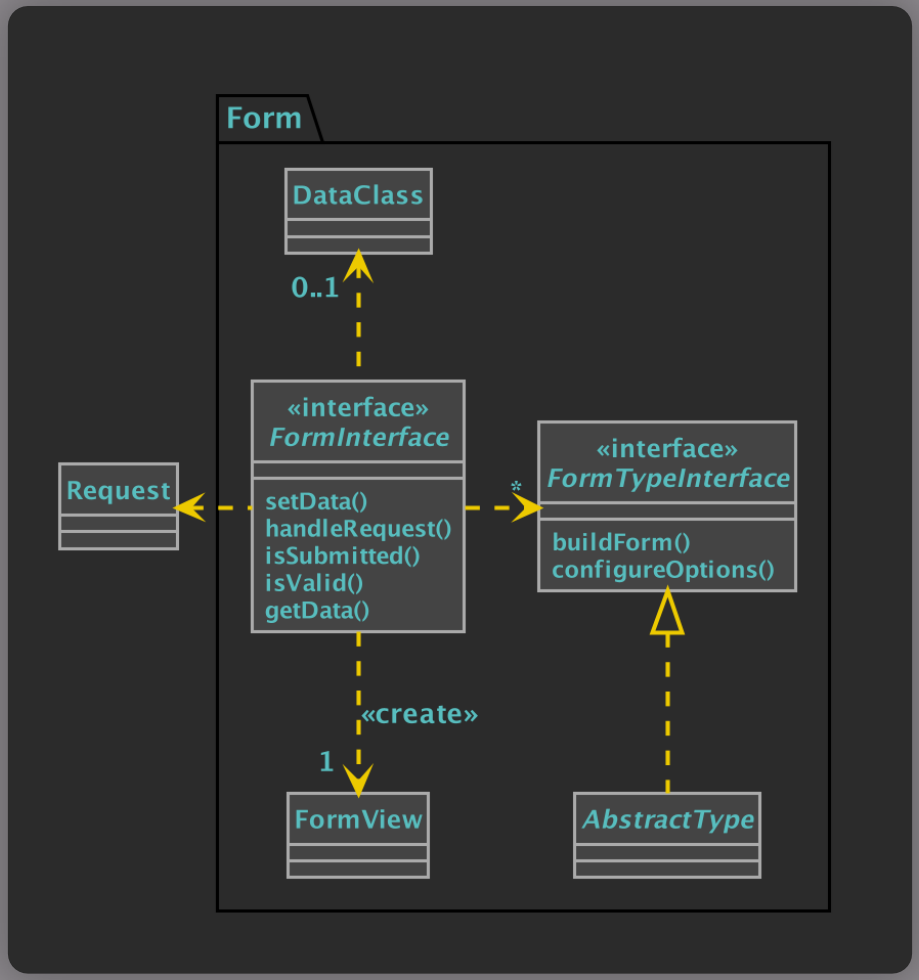
V. Data validation

Concepts

About forms

- Every form come with form *types*
- Native types:
  - Text
  - Date
  - Submit
  - Money
  - See more: https://symfony.com/doc/current/reference/forms/types.html#supported-field-types
- Custom types

APIs


Form
DataClass
0..1
«interface»
FormInterface
setData()
handleRequest()
isSubmitted()
isValid()
getData()
Request
«interface»
FormTypeInterface
buildForm()
configureOptions()
*
«create»
1
FormView
AbstractType

Building a form

Create a form type

```
1 symfony console make:form BookType Book
```

Default construction

```php
class BookType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('isbn')
            ->add('title')
            ->add('datePublished')
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => Book::class,
        ]);
    }
}
```

Create our form

```php
/**
 * @Route("/book")
 */
class BookController extends AbstractController
{
    /**
     * @Route("/create", name="book_create", methods={"GET"})
     */
    public function create()
    {
        $form = $this->createForm(BookType::class);

        return $this->render('book/create.html.twig', [
            'book_form' => $form->createView(),
        ]);
    }
}
```

```twig
{{ form(book_form) }}
```

Customize our form

Update our rendering

```yaml
# config/packages/twig.yaml

twig:
    default_path: '%kernel.project_dir%/templates'

    # Make Symfony write convenient html output for Bootstrap
    form_themes:
        - bootstrap_4_horizontal_layout.html.twig

# Now, don't forget to require Bootstrap in your stylesheets ♡
```

Update our type

```php
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('isbn', TextType::class, [
            'label' => 'ISBN',
        ])
        ->add('title', TextType::class, [
            'label' => 'Title',
        ])
        ->add('datePublished', DateType::class, [
            'label'  => 'Publication date',
            'widget' => 'single_text',
            'input'  => 'datetime_immutable',
        ])
        ->add('submit', SubmitType::class)
    ;
}
```

Dynamic scan of your types

```
# Dump known types
$ symfony console debug:form

# Dump options of a given type
$ symfony console debug:form 'App\Form\BookType'

# Have a look!
```

Exercises

1. Create a new form in order to update a movie.
    1. Create a new controller
    2. Create a new type
    3. Display the form
    4. Use Bootstrap theme ☺

Handle submitted data

Controller updates

```php
/**
 * @Route("/create", name="book_create", methods={"GET"})
 */
public function create(Request $request)
{
    $form = $this->createForm(BookType::class);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        dump($form->getData());
    }

    return $this->render('book/create.html.twig', [
        'book_form' => $form->createView(),
    ]);
}
```

Exercises

1. Register the new information about your movie
    1. Attach your form to an existing movie.
    2. If the form is submitted, then:
        1. save the data into your database
        2. Redirect the user to “admin_movie_index”

Data validation

About validation

- Configuration out-of-the-box
- Write validation rules on your data class

Validation

```php
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * @ORM\Entity(repositoryClass="App\Repository\BookRepository")
 * @UniqueEntity("isbn")
 */
class Book
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @Assert\NotBlank()
     * @Assert\Isbn(type="isbn13")
     * @ORM\Column(type="string", length=13)
     */
    private $isbn;

    // …
```

Exercises

1. Add constraints to your form data.
2. Check the validation with the Symfony Profiler.

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/forms.html

https://symfony.com/doc/current/reference/forms/types.html

https://symfony.com/doc/current/reference/constraints.html