Matière : Symfony

Date : 01/02/2023

Formateur : Ludovic UTRERA

Mail : ludovic.utrera@aflokkat.com

SYMFONY : Présentation

Auteur : Ludovic UTRERA

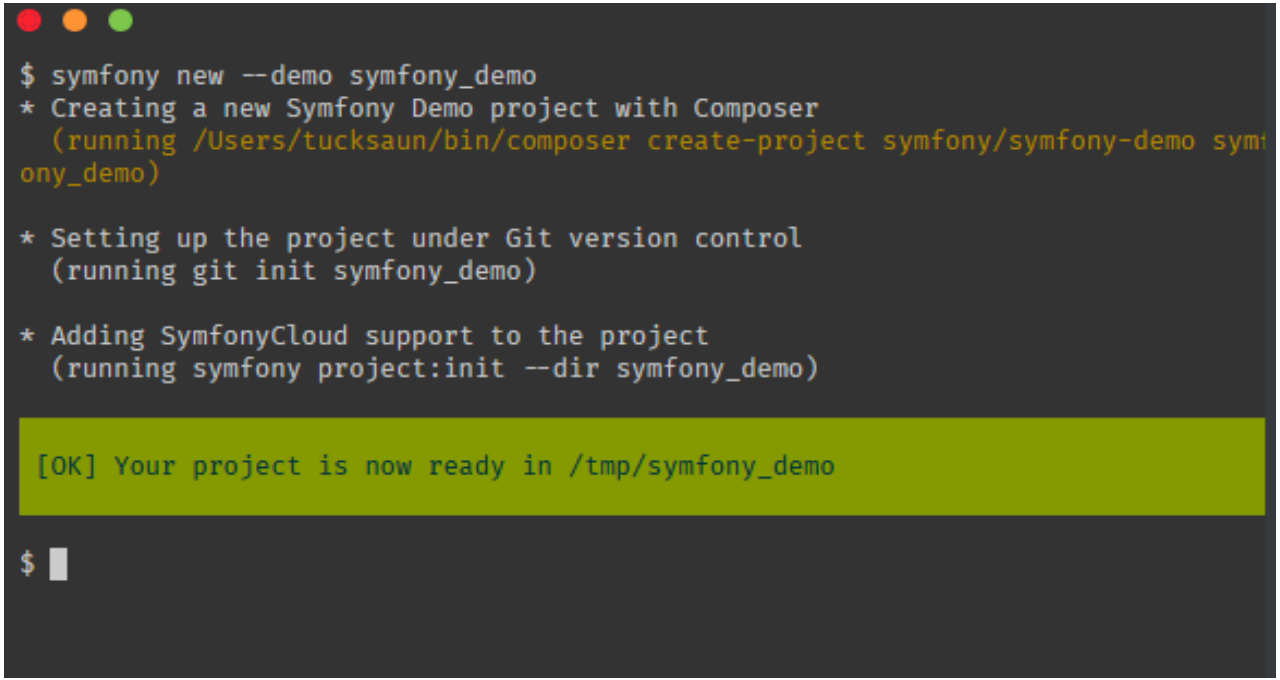
Contributeur : Sylvain MALERBA

SOMMAIRE

AFLOKKAT
Conseil et Formation

01 Introduction

AFLOKKAT
Conseil et Formation



AFLOKKAT
Conseil et Formation

Introduction

Nous allons étudier comment développer une application web complète avec Symfony.

La maitrise s'acquiert par la pratique

- En parallèle du cours, nous allons développer une application de A à Z
- Chaque notion théorique sera donc associé une mise en situation pratique

AFLOKKAT
Conseil et Formation

Symfony : présentation

Symfony est le framework PHP le plus utilisé et le plus mature.

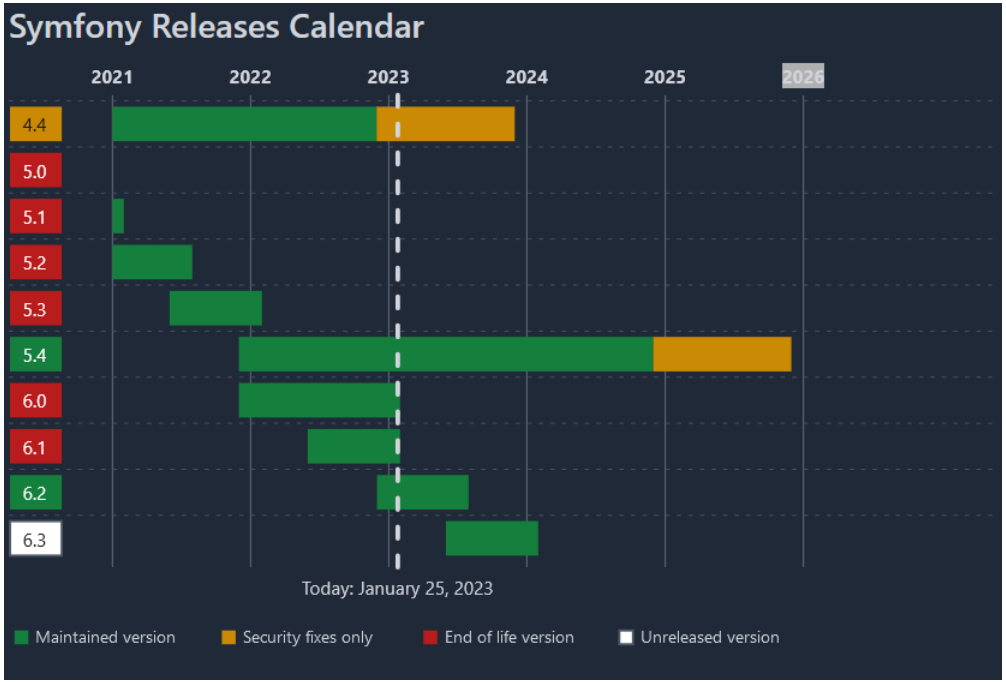
AFLOKKAT
Conseil et Formation

Symfony : 6 bonnes raisons

- Réputation
- Durabilité
- Références
- Innovation
- Ressources
- Interopérabilité

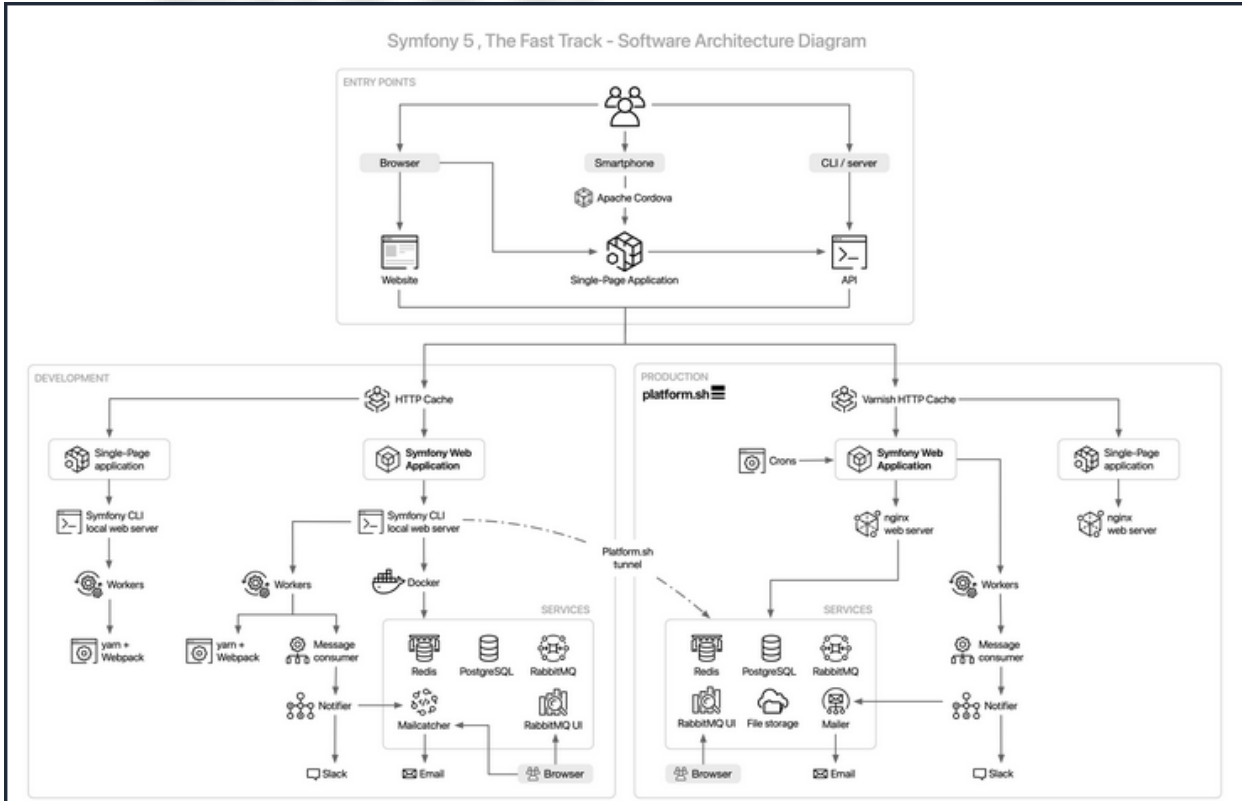
AFLOKKAT
Conseil et Formation

Symfony : Releases



02

Livre d’or


Symfony 5 , The Fast Track - Software Architecture Diagram
ENTRY POINTS
Browser
Smartphone
CLI / server
Apache Cordova
Website
Single-Page Application
API
DEVELOPMENT
HTTP Cache
Single-Page application
Symfony Web Application
Symfony CLI local web server
Symfony CLI local web server
Workers
Workers
Docker
yarn + Webpack
yarn + Webpack
Message consumer
SERVICES
Redis
PostgreSQL
RabbitMQ
Notifier
Mailcatcher
RabbitMQ UI
Slack
Email
Browser
Platform.sh tunnel
PRODUCTION
platform.sh
Varnish HTTP Cache
Crons
Symfony Web Application
Single-Page application
nginx web server
nginx web server
Workers
Message consumer
SERVICES
Redis
PostgreSQL
RabbitMQ
RabbitMQ UI
File storage
Mailer
Notifier
Browser
Email
Slack

AFLOKKAT
Conseil et Formation

Présentation du projet

Le projet a pour but d'obtenir un retour d'expérience sur des conférences : une liste des conférences sur la page d'accueil ainsi qu'une page pour chacune d'entre elles, pleine de commentaires sympathiques. Un commentaire est composé d'un petit texte et d'une photo, optionnelle, prise pendant la conférence.

AFLOKKAT
Conseil et Formation

Initialiser le projet

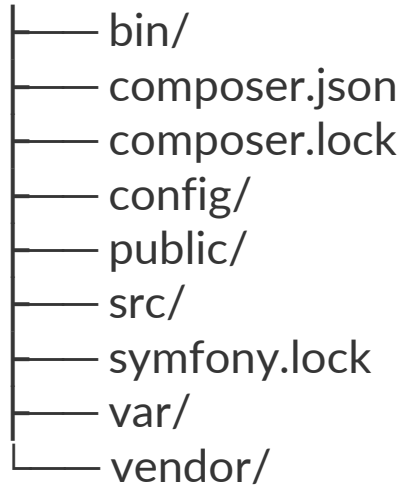
symfony new guestbook --version=6.2 --php=8.1 --webapp
cd guestbook

Cette commande est une mince surcouche de Composer qui facilite la création de projets Symfony. Elle utilise un squelette de projet qui inclut uniquement les composants Symfony requis par presque tous les projets : un outil console et l'abstraction HTTP nécessaire pour créer des applications web.

Comme nous créons une application web complète, nous avons ajouté quelques options qui nous faciliterons la vie :

--webapp: Par défaut, une application avec le moins de dépendances possibles est créée. Pour la plupart des projets web, il est recommandé d'utiliser le paquet webapp. Il contient la majeure partie des paquets nécessaires pour une application web "moderne". Le paquet "webapp" ajoute un nombre important de paquets Symfony, comme Symfony Messenger et MariaDB via Doctrine.

Structure d'un projet



Le répertoire bin/ contient le principal point d'entrée de la ligne de commande : console. Vous l'utiliserez tout le temps.

Le répertoire config/ est constitué d'un ensemble de fichiers de configuration sensibles, initialisés avec des valeurs par défaut. Un fichier par paquet. Vous les modifierez rarement : faire confiance aux valeurs par défaut est presque toujours une bonne idée.

Le répertoire public/ est le répertoire racine du site web, et le script index.php est le point d'entrée principal de toutes les ressources HTTP dynamiques.

Le répertoire src/ héberge tout le code que vous allez écrire ; c'est ici que vous passerez la plupart de votre temps. Par défaut, toutes les classes de ce répertoire utilisent le namespace PHP App. C'est votre répertoire de travail, votre code, votre logique de domaine. Symfony n'a pas grand-chose à y faire.

Le répertoire var/ contient les caches, les logs et les fichiers générés par l'application lors de son exécution. Vous pouvez le laisser tranquille. C'est le seul répertoire qui doit être en écriture en production.

Le répertoire vendor/ contient tous les paquets installés par Composer, y compris Symfony lui-même. C'est notre arme secrète pour un maximum de productivité. Ne réinventons pas la roue. Vous profiterez des bibliothèques existantes pour vous faciliter le travail. Le répertoire est géré par Composer. N'y touchez jamais.

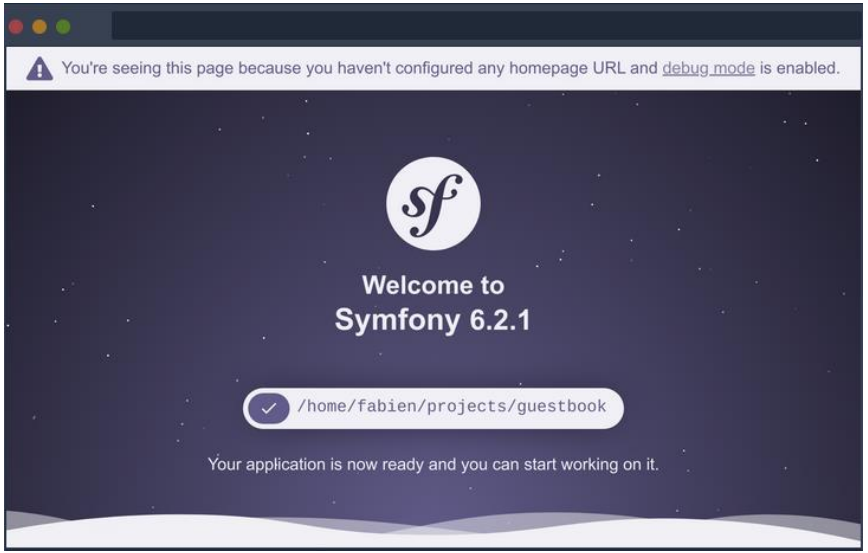
Lancer un serveur web local

symfony server:start –d

Le serveur a démarré sur le premier port disponible (à partir de 8000). Pour gagner du temps, vous pouvez ouvrir le site web dans un navigateur à partir de la ligne de commande :

symfony open:local

Votre navigateur favori devrait recevoir le focus et ouvrir un nouvel onglet affichant une page similaire à celle-ci :



Pour résoudre les problèmes, exécutez symfony server:log ; cette commande affiche les derniers logs de votre serveur web, de PHP et de votre application.

AFLOKKAT
Conseil et Formation

Ajouter des ressources publiques

Tout ce qui se trouve dans le répertoire public/ est accessible par un navigateur. Par exemple, si vous déplacez votre fichier GIF animé (nommez-le under-construction.gif) dans un nouveau répertoire public/images/, il sera alors disponible à une URL comme https://localhost/images/under-construction.gif.

```
mkdir public/images/
```

```
php -r "copy('http://clipartmag.com/images/website-under-construction-image-6.gif', 'public/images/under-construction.gif'); »
```

Naviguez vers /images/under-construction.gif. Vous devriez retrouvez l'image à droite

AFLOKKAT
Conseil et Formation

Ajouter une favicon

Pour éviter que nos logs ne se remplissent de messages d'erreur 404 à cause d'une favicon manquante, ajoutons-en une maintenant :

```
php -r "copy('https://symfony.com/favicon.ico', 'public/favicon.ico');"
```

AFLOKKAT
Conseil et Formation

Diagnostiquer les problèmes

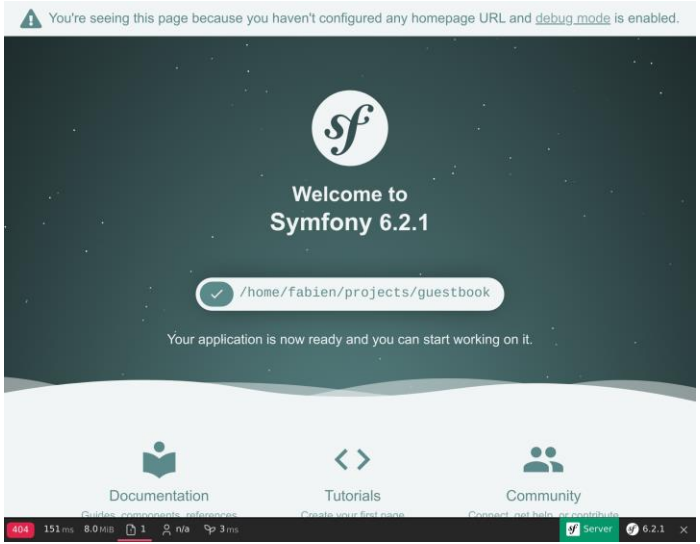
Mettre en place un projet, c'est aussi avoir les bons outils pour déboguer les problèmes. Fort heureusement, des assistants très utiles sont inclus avec le paquet webapp.
Nous allons découvrir les outils de débogage de Symfony.

Symfony profiler

Pour commencer, le Symfony Profiler vous fait gagner un temps fou lorsque vous avez besoin de trouver l'origine d'un problème.
Si vous regardez la page d'accueil, vous devriez voir une barre d'outils en bas de l'écran :
La première chose à voir, c'est le 404 en rouge.

Nous n'avons toujours pas défini de route pour la page d'accueil. Si vous cliquez sur le petit point d'exclamation en rouge vous obtenez le vrai message d'exception.

Vous pouvez aussi consulter les logs avec la commande symfony server:log



AFLOKKAT
Conseil et Formation

Les environnements Symfony

Comme le Symfony Profiler n'est utile que pendant le développement, nous voulons éviter qu'il soit installé en production. Par défaut, Symfony ne l'installe que pour les environnements de dev et de test.

Symfony intègre une notion d'environnement. Par défaut, il y en a trois, mais vous pouvez en ajouter autant que vous le souhaitez : dev, prod et test. Tous les environnements partagent le même code, mais ils représentent des configurations différentes.

Par exemple, tous les outils de débogage sont activés en environnement de dev. Dans celui de prod, l'application est optimisée pour la performance.

Basculer d'un environnement à l'autre peut se faire en changeant la variable d'environnement APP_ENV.

03

Controller

AFLOKKAT
Conseil et Formation

AFLOKKAT
Conseil et Formation

Créer un Controller

La page d'accueil de notre projet est une ennuyeuse page d'erreur 404. Corrigeons cela.

Lorsqu'une requête HTTP arrive au serveur, comme pour notre page d'accueil (http://localhost:8000/), Symfony essaie de trouver une route qui corresponde au chemin de la requête (/ ici). Une route est le lien entre le chemin de la requête et un "callable" PHP, une fonction devant créer la réponse HTTP associée à cette requête.

Ces " callables " sont nommés "contrôleurs". Dans Symfony, la plupart des contrôleurs sont implémentés sous la forme de classes PHP. Vous pouvez créer ces classes manuellement, mais comme nous aimons aller vite, voyons comment Symfony peut nous aider.

AFLOKKAT
Conseil et Formation

Se faciliter la vie avec le Maker Bundle

Pour générer des contrôleurs facilement, nous pouvons utiliser le paquet symfony/maker-bundle, qui a été installé en tant que composant du paquet webapp.

Le Maker Bundle vous permet de générer un grand nombre de classes différentes. Nous l'utiliserons constamment dans ce livre. Chaque "générateur" correspond à une commande et chacune d'entre elles appartient au même namespace make.

La commande list, intégrée nativement à la console symfony, permet d'afficher toutes les commandes disponibles sous un namespace donné ; utilisez-la pour découvrir tous les générateurs fournis par le Maker Bundle :

```
symfony console list make
```

Générer un contrôleur

AFLOKKAT
Conseil et Formation

Tout d'abord, vous allez créer votre premier controller avec la commande suivante :

symfony console make:controller ConferenceController

La commande crée une classe ConferenceController dans le répertoire src/Controller/. La classe générée contient du code standard prêt à être ajusté :

```
namespace App\Controller;

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

class ConferenceController extends AbstractController
{
    #[Route('/conference', name: 'conference')]
    public function index(): Response
    {
        return $this->render('conference/index.html.twig', [
            'controller_name' => 'ConferenceController',
        ]);
    }
}
```

Générer un contrôleur

AFLOKKAT
Conseil et Formation

L'attribut #[Route('/conference', name: 'conference')] est ce qui fait de la méthode index() un contrôleur (la configuration est à côté du code qu'elle configure).

Lorsque vous visitez la page /conference dans un navigateur, le contrôleur est exécuté et une réponse est renvoyée.

Modifiez la route afin qu'elle corresponde à la page d'accueil :

```
-    #[Route('/conference', name: 'conference')]
+    #[Route('/', name: 'homepage')]
```

Le nom de la route (name) sera utile lorsque nous voudrons faire référence à la page d'accueil dans notre code. Au lieu de coder en dur le chemin /, nous utiliserons le nom de la route.

À la place de la page par défaut, retournons une simple page HTML :

```
return new Response('
<html>
  <body>
    <img src="/images/under-construction.gif" />
  </body>
</html>');
```

Générer un contrôleur

AFLOKKAT
Conseil et Formation

Pour montrer comment une réponse peut tirer parti de l'information contenue dans la requête, ajoutons un petit easter egg. Lorsqu'une requête vers la page d'accueil sera réalisée avec un paramètre d'URL comme ?hello=Fabien, nous ajouterons du texte pour saluer la personne :

```
public function index(Request $request): Response
   {
     $greet = '';
     if ($name = $request->query->get('hello')) {
        $greet = sprintf('<h1>Hello %s!</h1>', htmlspecialchars($name));
     }
      return new Response('
<html>
   <body>
     '.$greet.'
     <img src="/images/under-construction.gif" />
   </body>
</html>');
}
```

Symfony expose les données de la requête à travers un objet Request. Lorsque Symfony voit un argument de contrôleur avec ce typage précis, il sait automatiquement qu'il doit vous le passer. Nous pouvons l'utiliser pour récupérer le nom depuis le paramètre d'URL et ajouter un titre <h1>.

Dans un navigateur, rendez-vous sur /, puis sur /?hello=Fabien pour constater la différence.

04

Conclusion

Dans cette première partie nous avons :

Découvert l'environnement Symfony
Vu comment configurer un projet Symfony
Appris à associer le code de notre application à une route à l'aide des Controller

AFLOKKAT

Sources

https://symfony.com
https://stackoverflow.com/
https://openclassrooms.com

AFLOKKAT
Conseil et Formation

MERCI DE VOTRE ATTENTION !