Framework et POO côté serveur

Natacha Braun
braun.natacha@gmail.com

Framework

Framework d'application web

Un framework web est un ensemble d'outils conçus pour faciliter le processus de développement d'application web. L'objectif d'un framework c'est d'apporter les composants de bases qui constituent toutes les applications web tel un système de gestion des routes, un moteur de templates...

Avantages de l'utilisation d'un framework

- respect des standards et des bonnes pratiques
- rapidité du développement
- communication facilitée entre développeurs
- stabilité, solidité et sécurité grâce à une communauté active
- évolutivité et modularité du code

Inconvénients de l'utilisation d'un framework

- learning curve
- possibilité que l'outil ne se plie pas exactement aux exigences

Principaux frameworks PHP

- Symfony **:**
  - Symfony is a set of PHP Components, a Web Application framework, a Philosophy, and a Community — all working together in harmony.
- Laravel :
  - Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.
- Phalcon :
  - High performance, full-stack PHP framework delivered as a C extension
- Lumen :
  - The stunningly fast micro-framework by Laravel.

Symfony

Qu'est-ce que Symfony

Symfony est un framework PHP Open Source créé en 2005 par Fabien Potencier. Il est maintenu et distribué par la société parisienne Sensio Labs.

Symfony est un framework basé sur le modèle MVC (Model View Controller).

Le modèle MVC sépare les requêtes de la base de données (Model) de la logique relative au traitement des demandes (Controller) du rendu de la présentation (View).

Framework HTTP centré sur un kernel (noyau) => Kernel.php, et la notion de requête/réponse.

Il se compose de plusieurs librairies indépendantes les unes des autres résolvant chacune une problématique liée au développement d'applications web : Routing (correspondance entre une URL et un contrôlleur), accès et manipulation de bases de données (via l'ORM Doctrine), gestion des formulaires, sécurité, mails, queues de messageries, i18n...

Les versions récentes de Symfony (àpd 4.0) nous permettent de ne plus installer le framework dans une version monolithique ` (incluant l'ensemble des composants qu'une application web contient traditionnellement) mais plutôt avec une philosophie "micro-framework" : n'installer que les composants nécessaires au fonctionnement de notre application (ex: si l'application n'utilise pas de formulaires ou de templates html, l'installation des librairies est facultative). Cette approche permet de conserver une application aussi légère que possible, d'améliorer les performances (vitesse de chargement, impact sur la mémoire) et d'éviter les failles de sécurité qui pourraient exister dans des composants qui ne sont pas utilisés.

https://symfony.com/about

https://symfony.com/at-a-glance

Les composants du framework Symfony

Symfony n'est pas un logiciel monolithique. C'est une agrégation de plusieurs bibliothèques apportant chacune des solutions à une problématique liée au développement d'applications web.

Symfony, en tant que framework, propose une implémentation possible permettant d'intégrer et de faire interagir les différents composants.

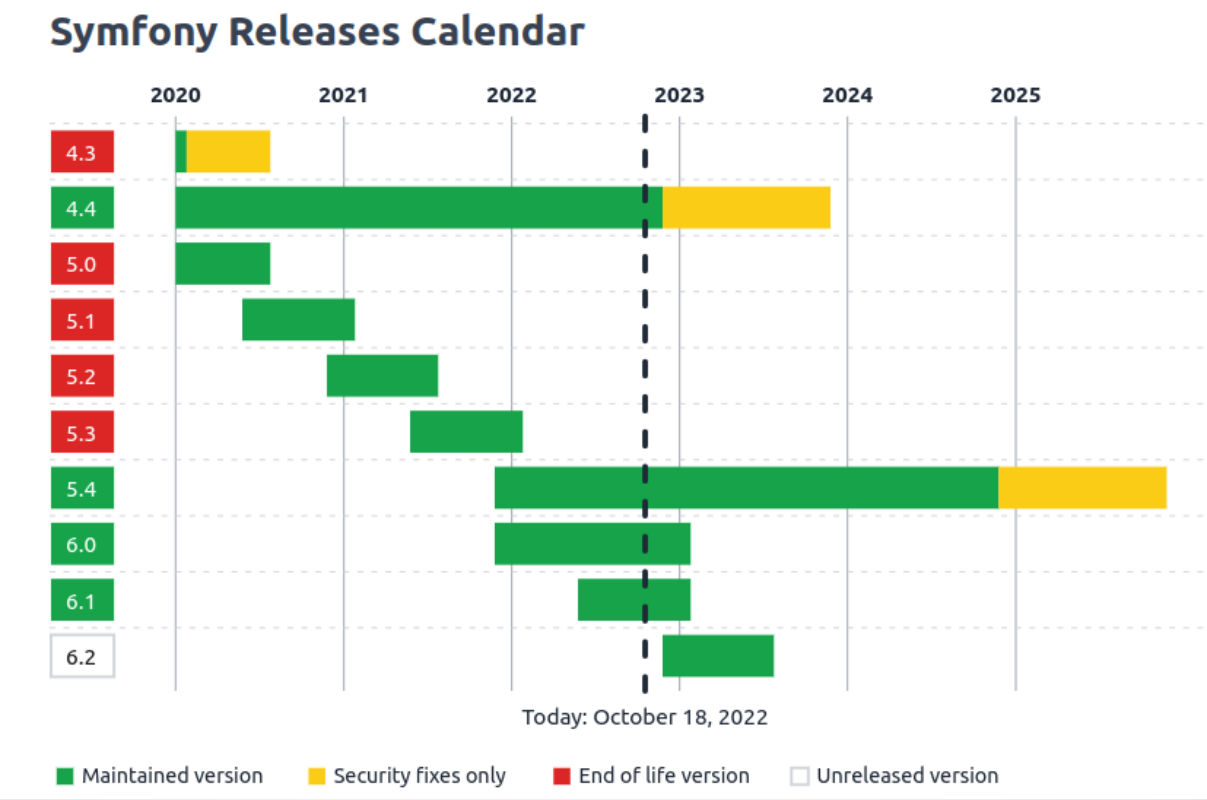
En tant que développeur de bibliothèque open source, et communauté, Symfony fournit des outils et se charge de la maintenance et de l'évolution de ceux-ci.

La plupart des composants utilisés par le framework Symfony sont développés de manière à les rendre autonomes. Il est donc tout à fait possible d'intégrer seulement certaines des briques de Symfony dans des projets de moindre envergure ou pour faire évoluer progressivement un projet existant.

https://symfony.com/doc/current/components/index.html

Version de Symfony

- Nous utiliserons la version « long term support » pour ne pas devoir changer de version en cours de route
- Version 5.4



Installer Symfony

Prérequis à l'installation de Symfony 5.4:

- PHP 7.4 ou >
- Composer

Il existe deux façons d'installer Symfony :

- Via **Composer** : outil de gestion de dépendances PHP
- Via **Symfony CLI** : l'outil en ligne de commande symfony

Symfony propose deux versions à l'installation :

- symfony/skeleton : La version minimale du framework : Inclut uniquement les composants essentiels à son fonctionnement (HttpKernel, Routing). Généralement utilisé dans le développement d'API ou de microservices.
- Symfony/ webapp: installe tous les composants nécessaires à la création d'une application web complète (templating, base de données…)

La documentation officielle contient toutes les informations nécessaires à l'installation du framework

https://symfony.com/doc/5.4/setup.html

Environnement de développement

- **Serveur Symfony** : https://symfony.com/doc/5.4/setup/symfony_server.html
  - Installer PHP, composer et symfony cli (https://symfony.com/download)
  - Utiliser le serveur web de développement embarqué de symfony
    - Démarrer le serveur : `# symfony server:start`
    - `Aide : symfony server:start --help`
  - Installer un SGBD (MySQL ou PostrgreSQL)
- Environnement Dockerisé
  - Environnement pré-configuré proposé dans les apps symfony

Conseil : utilisez le serveur web embarqué dans le cadre du cours
Prérequis : avoir php en variable d'environnement

Docker :

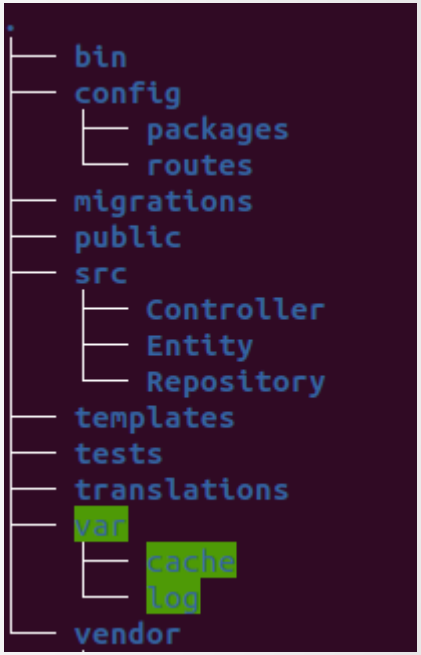
- Docker est une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation. Les services ou fonctions de l'application et ses différentes bibliothèques, fichiers de configuration, dépendances et autres composants sont regroupés au sein du container. Chaque container exécuté partage les services du système d'exploitation.
- Installlation sur window : https://docs.docker.com/docker-for-windows/install/

Suivi des projets via GIT :

- Git est un système de contrôle de version open source
- Sources sur GitLab ou GitHub ; Plateformes permettant d'héberger et de gérer des projets.
- Cheat sheet GIT :
  - Version GitHub : https://training.github.com/downloads/fr/github-git-cheat-sheet.pdf
  - Version GitLab : https://about.gitlab.com/images/press/git-cheat-sheet.pdf

Structure d’un projet Symfony

- bin/
  - Le répertoire continent les différents exécutables utiles à l’application. Ex : console; phpunit
- config/
  - Contient les fichiers de configuration.Pour le framework, la sécurité, les routes... config/packages contient les fichiers de configuration des différents packages (ensemble de librairies).
- migrations/
  - Contient les fichiers de migration, ils permettent la mise à jour de la DB
- public/
  - Le répertoire qui sert de racine au serveur web. Tous les fichiers publiquement accessibles seront mis ici. Lors de l’installation de nouveaux paquets, de nouveaux répertoires pourraient être créés ici, au besoin.
- src/
  - Le code PHP spécifique à l’application
- templates/
  - Les vues, templates Twig de l’application
- var/
  - Contient tous les fichiers générés par l’application; le cache (var/cache/) et les logs (var/log/). Le contenu de ce répertoire étant créé au runtime, il ne doit pas être versionné
- vendor/
  - Les librairies php installées par Composer, le package manager, et configurées dans le fichier ./composer.json à la main ou via la commande composer require



php bin/console

L’outil console est un outil en ligne de commande qui fournit un grand nombre de commandes facilitant la gestion et le développement d’applications Symfony.

Lister les commandes :

```
# php bin/console list
```

Ces commandes permettent de faire du debugging, générer du code, générer des migrations en DB et bien plus. La liste des commandes disponibles augmentent avec l’ajout de packages dans l’application.

elle fait.

Exemple : Génération d’un controlleur

```
# php bin/console make:controller
```

Symfony Flex et packages

Symfony Flex: plugin pour Composer

Flex est présent par défaut depuis Symfony 4. Il s'agit d'un plugin pour composer qui permet de faciliter l'installation et la désinstallations de packages pour Symfony. Ce plugin modifie le comportement des commandes require, update et remove de Composer.

À la différence des packages Composer, les packages Flex peuvent se composer de plusieurs librairies ainsi que d'un fichier de configuration manifest.json qui contient les différentes instructions pour l'installation, la configuration et la désinstallation.

On parle alors de recette (recipe), c'est à dire l'ensemble des différentes instructions nécessaires à l'installation du package.

Les packages officiels: https://flex.symfony.com/

doc: https://symfony.com/doc/current/setup/flex.html

Packages :

Tout les composant de symfony sont sous forme de packages ; twig, doctrine, les annotations …

Les packages sont installés via composer avec la commande

```
# Composer require leNomDuPackage
```

Controller & Routes

Controller

- L'un des composants de l'acronyme MVC, le contrôleur est une fonction PHP qui permet de lire une requête et renvoyer une réponse (qui peut être du HTML, JSON ou binary file tel qu'une image ou un PDF).

Route

- Une route est l'URL (ex. /about) d'une page et pointe vers un controller. Quand le serveur reçoit une requête HTTP, symfony regarde dans le fichier config/routes.yaml ou dans les annotations (via le package annotations routes) pour trouver la route, et cette route va définir le contrôleur a appelé dans ce cas.

Routing

Lorsque votre application reçoit une requête, elle exécute une action (méthode) d'un contrôleur pour générer la réponse. La configuration du routage définit quelle action correspond à chaque URL entrante. D'autres fonctionnalités utiles sont également fournies, comme la création d'URL SEO-friendly (par exemple: /read/intro-to-symfony à la place de index.php?article_id=57)

Création des routes :

config/routes.yaml : La configuration par défaut des routes de l'application se trouve dans config/routes.yaml.

```
## Definition d'une route
nom_de_la_route:
  # URL qui accede a la fonction
    path: /lucky/number
  # Controller qui répond a l'URL :: le nom de la fonction du
controller
    controller: App\Controller\RouteController::number
```

La méthode recommandée pour la configuration des routes, est cependant celle des annotations

Une annotation est une configuration sous forme de commentaire qui peut être lue par un script et ajouter un comportement à une méthode. Elles seront placées au dessus de la méthode contrôleur, ce qui simplifie la lecture : la route est directement en regard de la méthode qui la gère.

```
/**
* @Route("/", name="app_index")
**/
public function index() {
    return new Response('hello world');
}
```

pour utiliser les annotations dans une application Symfony, installer le pack avec Composer :

```
# composer require annotations
```

Requêtes et Réponses

Symfony fournit une approche à travers deux classes pour interagir avec la requête et la réponse HTTP. La classe Request est une représentation de la requête HTTP, tandis que la classe Response est évidemment une représentation de la réponse HTTP.

Un moyen de gérer ce qui se passe entre la requête et la réponse consiste à utiliser un contrôleur frontal (public/index.php). Ce fichier traitera chaque demande entrant dans notre application. Cela signifie qu'il sera toujours exécuté et qu'il gérera le routage de différentes URL vers différentes parties de notre application.

Dans Symfony, les demandes entrantes sont interprétées par le composant Routing et transmises aux fonctions PHP (contrôleurs) qui renvoient des réponse. Cela signifie que le contrôleur frontal transmettra la demande à Symfony. Ce dernier créera un objet de réponse et le transformera en en-têtes de texte et le contenu sera finalement renvoyé.

Controller & Routes

Exercices :

Tout les éléments nécessaires à la réalisation de ces exercices ce trouvent ici : **https://symfony.com/doc/5.4/page_creation.html**

Créer un projet symfony (full --webapp)

- Créer un controller et créer les routes suivantes : en utilisant les Annotation Routes pour définir la route
  - Une route retournant votre nom et prénom
  - Une route retournant un chiffre aléatoire avec la date du jour

Créer un second controller avec les routes suivantes :

  - Votre nom, prénom et adresse en JSON avec l'objet Response
  - Votre nom, prénom et adresse en JSON avec l'objet JsonResponse

- Doc : https://symfony.com/doc/5.4/components/http_foundation.html#creating-a-json-response

Afficher les routes dans le debuger intégré

```
# php bin/console debug:router
```

La Web Debug Toolbar

« One of Symfony's amazing features is the Web Debug Toolbar: a bar that displays a huge amount of debugging information along the bottom of your page while developing. This is all included out of the box using a Symfony pack called symfony/profiler-pack. »

Le profiler-pack permet d'installer la debug toolbar de Symfony: un outil puissant et pratique aidant au développement en fournissant de nombreuses informations sur l'application

En bas de votre page

Exercices :

Consulter la debug toolbar et voir les requêtes effectuées sur une des routes créés

- **https://symfony.com/doc/current/profiler.html**