

SYMFONY

framework PHP

Installation et explication

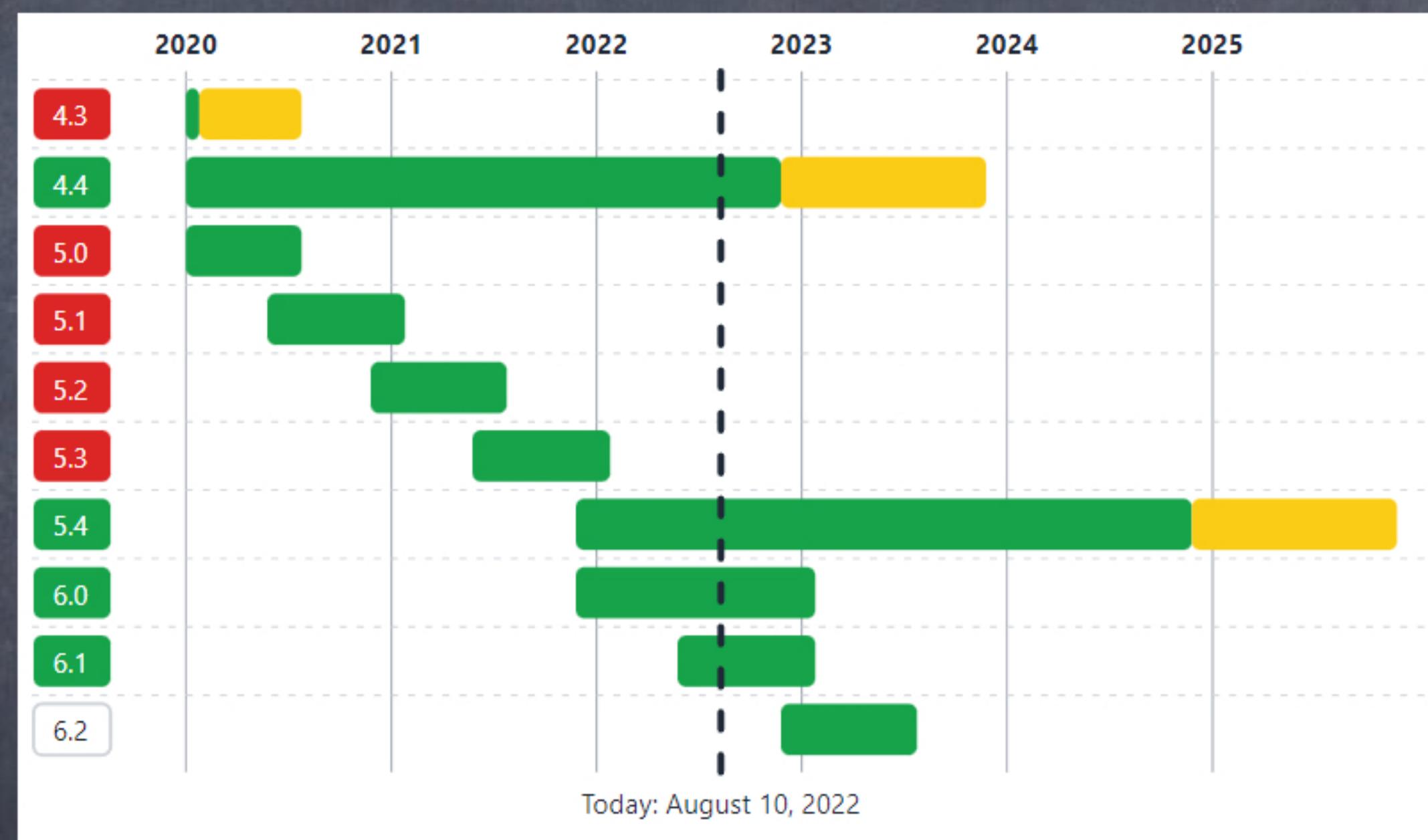
Anthony PARIS - Formateur

Presentation symfony
conception MVC
Présentation des outils
installation
Création d'un bundle

PRÉSENTATION

Symfony est un framework PHP.

Nous utiliserons la version 6 de Symfony même s'il ne s'agit pas d'une version LTS (Long time support).



QU'EST CE QU'UN FRAMEWORK

Un framework (ou infrastructure logicielle en français) désigne en programmation informatique un ensemble d'outils et de composants logiciels à la base d'un logiciel ou d'une application. C'est le framework, encore appelé structure logicielle, canevas ou socle d'applications en français, qui établit les fondations d'un logiciel ou son squelette applicatif. Tous les développeurs qui l'utilisent peuvent l'enrichir pour en améliorer l'utilisation.

L'objectif du framework est de simplifier et d'uniformiser le travail des développeurs. Il fonctionne comme un cadre ou un patron, mais son maniement suppose d'avoir déjà un profil expérimenté. En général, une infrastructure logicielle est associée spécifiquement à un langage de script ou de programmation.

Par exemple, Symfony est un framework pour PHP et Django pour Python.
A noter : React au sens propre est un librairie, cependant son ecosystème évolue avec Redux, il n'est pas rare de voir que React est un framework.(Il ne fournit pas de cadre de travail).
Ceci dit Angular est un framework et React est un librairie.

CONNAISSANCES NECESSAIRES POUR DÉMARRER

- Grande connaissance en PHP
- Grande connaissance en POO
- Être à l'aise avec les namespaces
- Comprendre la notion de MVC
- Connaitre composer
- Connaitre Webpack (ici utilisation de webpack Encore) Symfony
6
- Comprendre l'intérêt d'un ORM

CONCEPTION MVC

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

MODÈLE : cette partie gère les données de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.

VUE : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

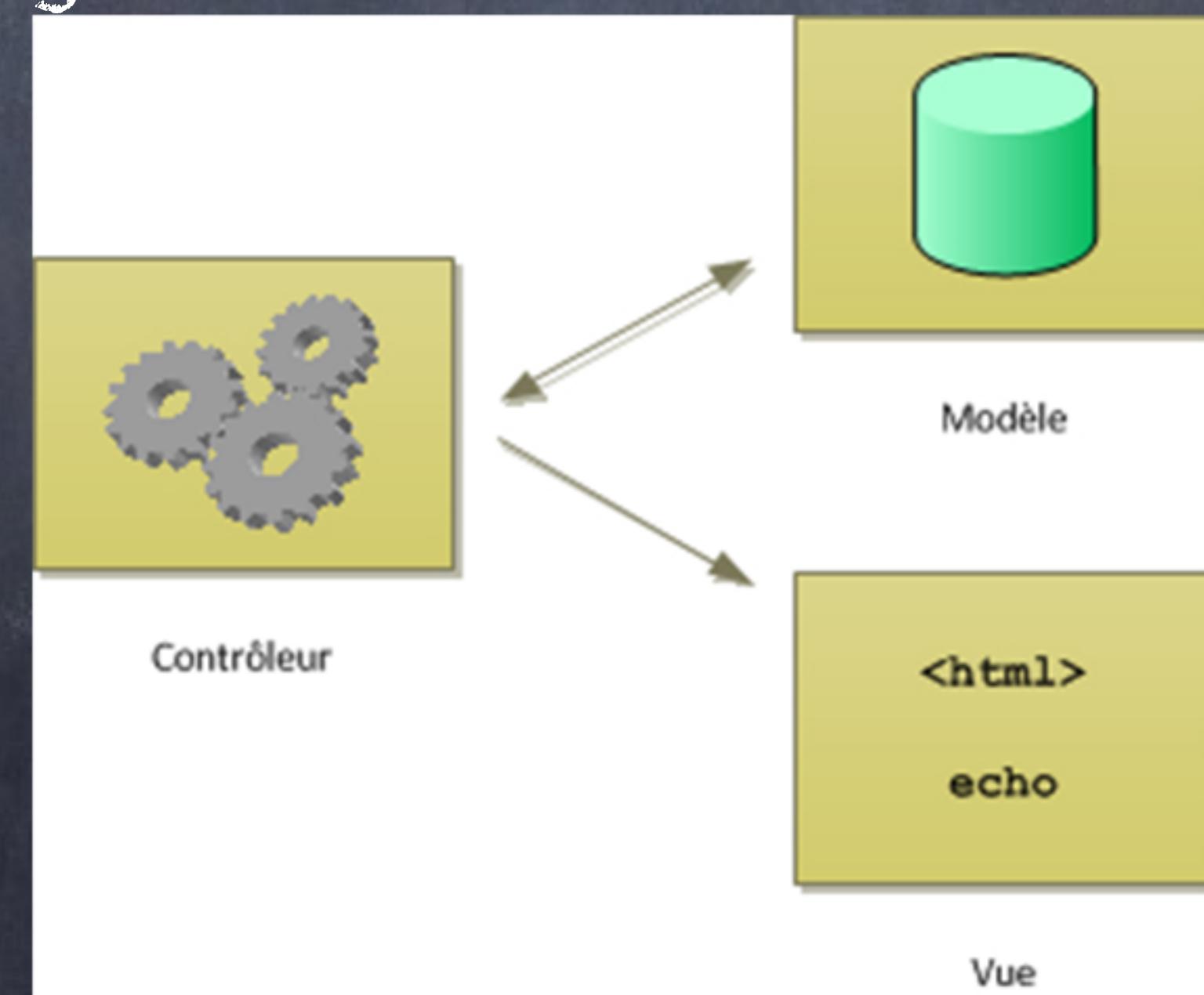
CONTRÔLEUR : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

CONCEPTION MVC

Le contrôleur est le chef d'orchestre : c'est lui qui reçoit la requête du visiteur et qui contacte d'autres fichiers (le modèle et la vue) pour échanger des informations avec eux.

Le fichier du contrôleur demande les données au modèle sans se soucier de la façon dont celui-ci va les récupérer. Par exemple : « Donne-moi la liste des 30 derniers messages du forum numéro 5 ». Le modèle traduit cette demande en une requête SQL, récupère les informations et les renvoie au contrôleur.

Une fois les données récupérées, le contrôleur les transmet à la vue qui se chargera d'afficher la liste des messages.

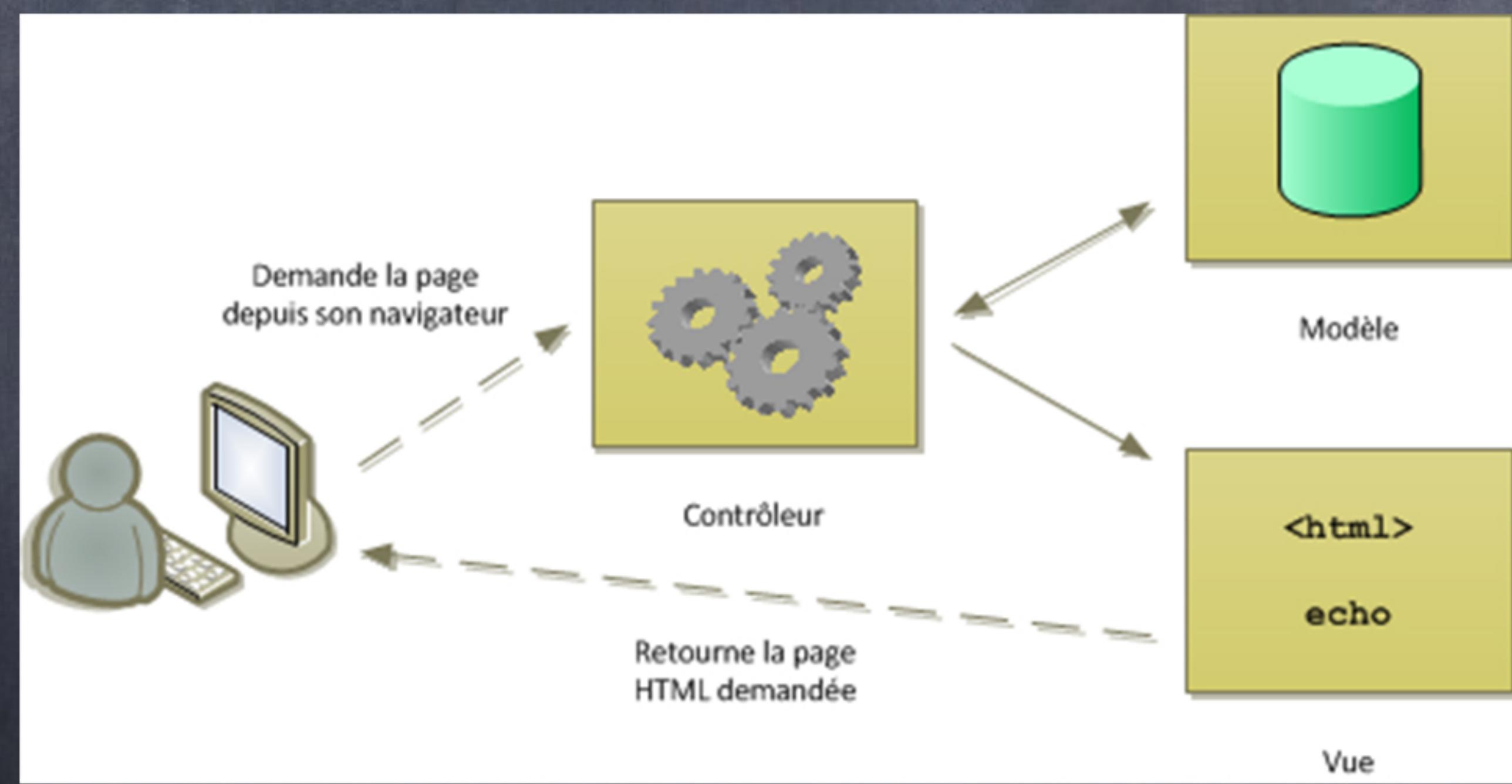


CONCEPTION MVC

Majoritairement, le contrôleur servira à faire la jonction modèle/vue s'il y a des calculs ou des vérifications d'autorisations à faire, c'est lui qui s'en chargera.

Concrètement, le visiteur demandera la page au contrôleur et c'est la vue qui lui sera renvoyée.

Cela est transparent pour lui, il ne voit pas tout ce qui se passe sur le serveur. C'est un schéma plus complexe que ce à quoi vous avez été habitués. Cependant tout gros projet repose sur ce type d'architecture pour faciliter le travail des développeurs.



CONCEPTION MVC

MVC est un modèle général.

L'idée est de séparer les trois aspects (modèle, vue et contrôle) d'une application.

Maintenant, cette idée peut être réalisée de différentes manières selon les spécificités d'une situation / application.

Une façon consiste à avoir de nombreux contrôleurs, chacun répondant à une action ou à un ensemble d'actions.

Une autre façon est d'avoir un contrôleur principal qui reçoit toutes les actions et les envoie ensuite à différents contrôleurs, celui-ci est appelé Front Controller Pattern (votre framework)

Ainsi, le modèle de contrôleur frontal est un modèle MVC .

TELECHARGER SYMFONY

Plusieurs possibilités : (.exe, composer, etc...)

Nous utiliserons la méthode cli symfony.

Depuis 2018 il existe plusieurs sous versions de symfony dit «bundle less».

A la base, symfony était fourni avec un ensemble de fonctionnalité assez lourd n'étant pas toujours nécessaire dans un projet.

C'est pourquoi aujourd'hui nous pouvons télécharger des versions plus «light» en fonction de notre projet.

RDV sur packagist pour vérifier comment l'installer.

Cherchez : symfony/website-skeleton (sous ensemble de SF spécifique au site internet)

Nous utiliserons symfony-cli pour cela rendez-vous sur :
<https://symfony.com/download>

TELECHARGER SYMFONY

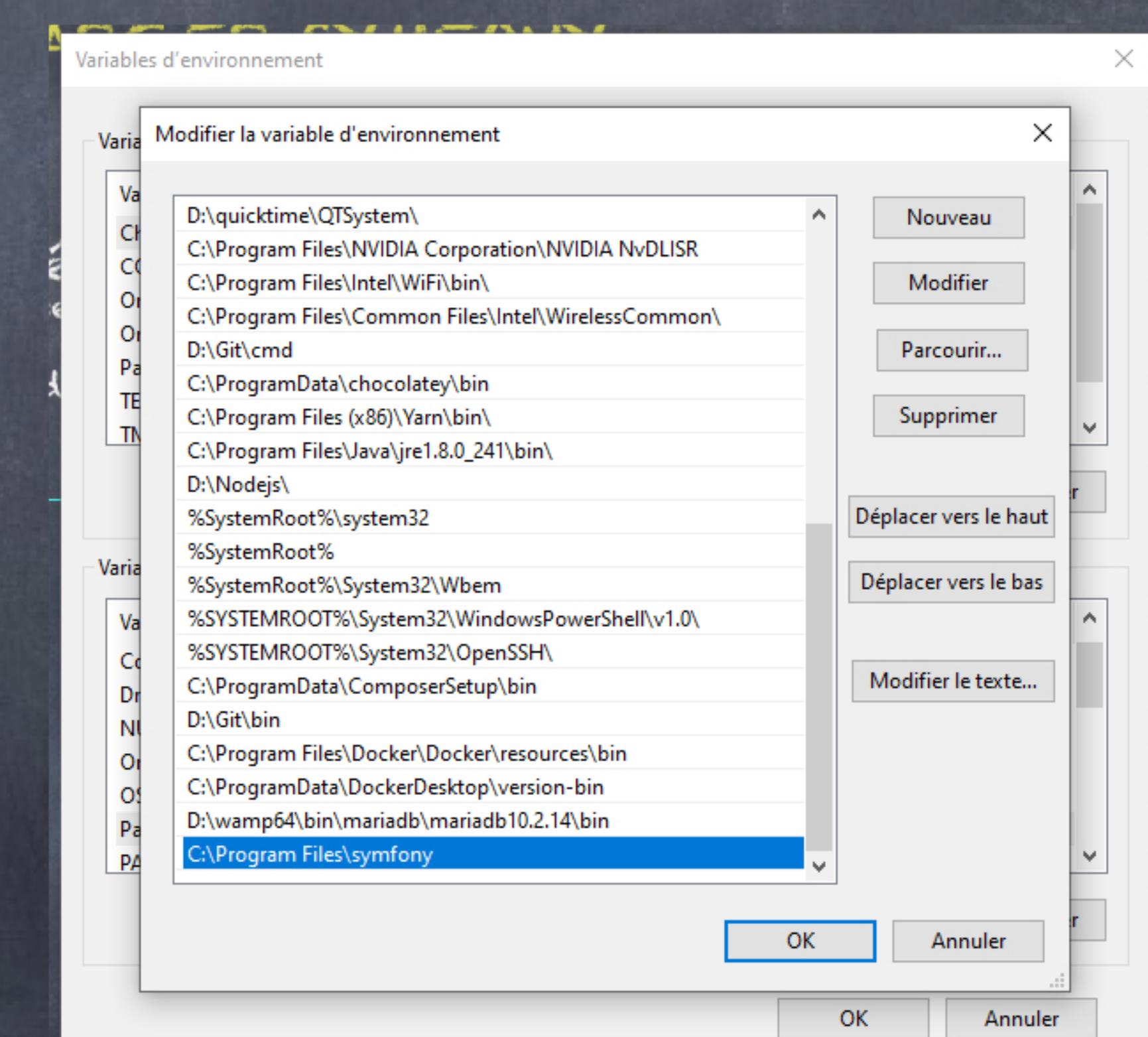
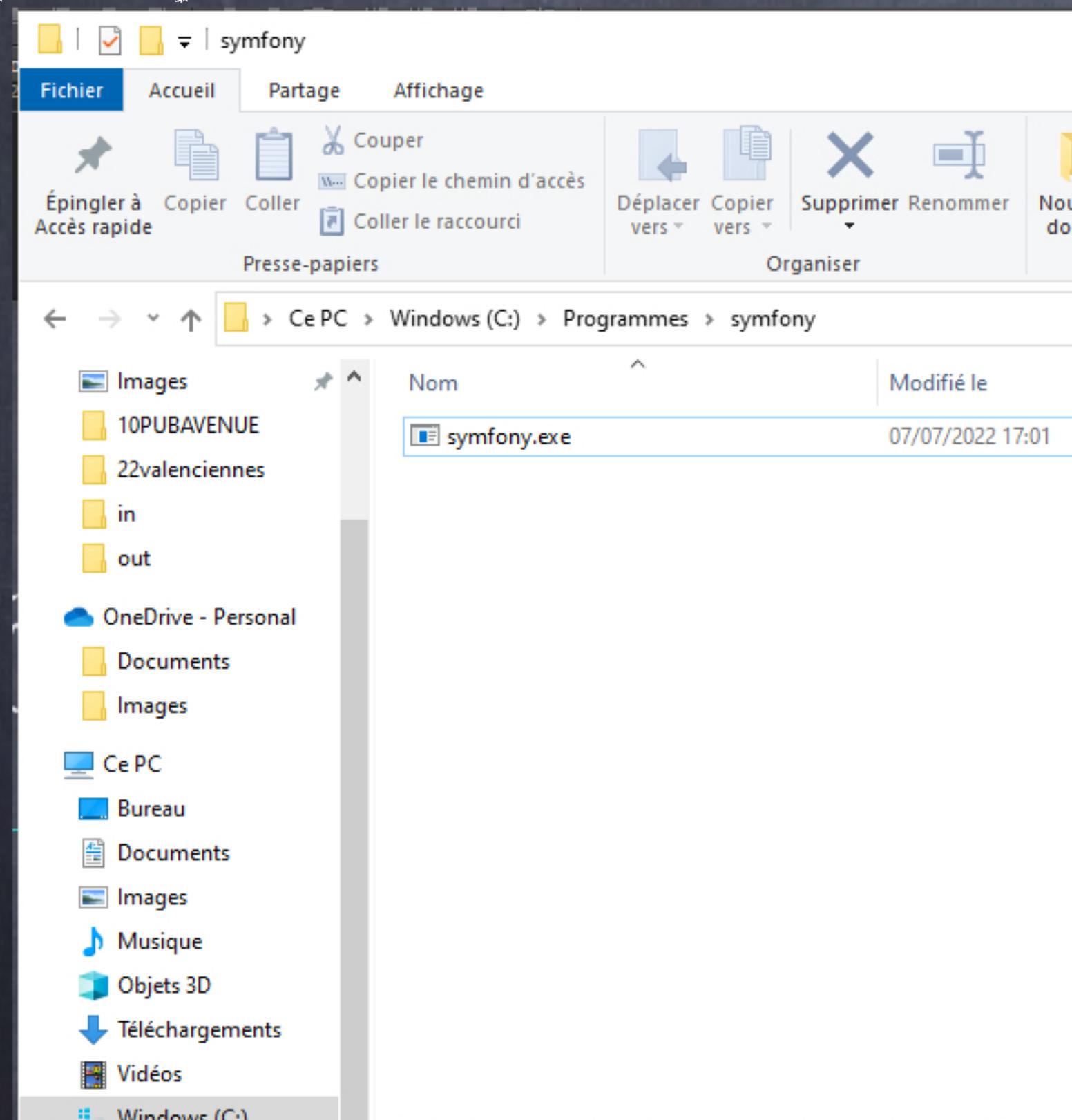
Pour mac : utilisez Brew

Pour windows utilisez le lien pour télécharger le binaire :

https://github.com/symfony-cli/symfony-cli/releases/download/v5.4.12/symfony-cli_windows_amd64.zip

Une fois téléchargé, vous allez créer un dossier dans programFiles et le glisser dedans.

Ajoutez ensuite le chemin dans les variables systèmes. Redémarrez votre ordinateur et utilisez la commande symfony -v en console



TELECHARGER SYMFONY

Placez vous à la racine de votre dossier WWW et lancez la commande de création

```
PS D:\wamp64\www> symfony new --webapp sf6m2icours --version="6.1.*"
```

Si symfony cli ne fonctionne pas, utilisez composer :

```
# run this if you are building a traditional web application
$ composer create-project symfony/skeleton:"6.1.*" my_project_directory
$ cd my_project_directory
$ composer require webapp
```

C'est assez long mais en vérifiant la commande, vous téléchargez énormément de choses et vous aide pour votre avancement!

```
* Creating a new Symfony 6.1.* project with Composer
  (running C:\ProgramData\ComposerSetup\bin\composer.phar create-project symfony/skeleton
D:\wamp64\www\sf6m2icours 6.1.* --no-interaction)

* Setting up the project under Git version control
  (running git init D:\wamp64\www\sf6m2icours)

  (running C:\ProgramData\ComposerSetup\bin\composer.phar require webapp --no-interaction
)
```

[OK] Your project is now ready in D:\wamp64\www\sf6m2icours

```
PS D:\wamp64\www> 
```

VERIFICATION INSTALLATION

Via votre navigateur RDV sur l'url d'installation.

Nom	Modifie le	Type	Taille
.git	10/08/2022 17:50	Dossier de fichiers	
bin	10/08/2022 17:49	Dossier de fichiers	
config	10/08/2022 17:49	Dossier de fichiers	
migrations	10/08/2022 17:49	Dossier de fichiers	
public	10/08/2022 17:49	Dossier de fichiers	
src	10/08/2022 17:49	Dossier de fichiers	
templates	10/08/2022 17:49	Dossier de fichiers	
tests	10/08/2022 17:49	Dossier de fichiers	
translations	10/08/2022 17:49	Dossier de fichiers	
var	10/08/2022 17:49	Dossier de fichiers	
vendor	10/08/2022 17:49	Dossier de fichiers	
.env	10/08/2022 17:49	Fichier ENV	2 Ko
.env.test	10/08/2022 17:49	Fichier TEST	1 Ko
.gitignore	10/08/2022 17:49	Document texte	1 Ko
composer.json	10/08/2022 17:49	Fichier JSON	4 Ko
composer.lock	10/08/2022 17:49	Fichier LOCK	355 Ko
docker-compose.override.yml	10/08/2022 17:49	Fichier YML	1 Ko
docker-compose.yml	10/08/2022 17:49	Fichier YML	1 Ko
phpunit.xml.dist	10/08/2022 17:49	Fichier DIST	2 Ko
symfony.lock	10/08/2022 17:49	Fichier LOCK	8 Ko

Cela ressemble à notre framework, cliquez sur le dossier public!

VERIFICATION INSTALLATION

⚠ You're seeing this page because you haven't configured any homepage URL and [debug mode](#) is enabled.



Welcome to
Symfony 6.1.3

✓ D:\wamp64\www\sf6m2icours\

Your application is now ready and you can start working on it.



Documentation

[Guides, components, references](#)



Tutorials

[Create your first page](#)



Community

[Connect, get help, or contribute](#)

Pour éviter de mettre toujours public 3 solutions. A votre avis lesquels?

VERIFICATION INSTALLATION

- Serveur interne de PHP
`php -S localhost:8000 -t public/`
- Virtual host dans Wamp
- Serveur de développement symfony
Il suffit de lancer la commande `symfony server:start`

Vous pouvez lancer la commande :
`symfony server:ca:install` pour installer un certificat SSL local pour utiliser le https

VERIFICATION INSTALLATION

Rendez vous dans le dossier de votre projet avec le CMD :

```
PS D:\wamp64\www\sf6m2icours> symfony server:start
Following Web Server log file (C:\Users\hayu-\.symfony5\log\bef9c902cda0e31802e
67d9bf18430e4403f1ef6.log)
Following PHP-CGI log file (C:\Users\hayu-\.symfony5\log\bef9c902cda0e31802e67d
9bf18430e4403f1ef6\79ca75f9e90b4126a5955a33ea6a41ec5e854698.log)

[WARNING] The local web server is optimized for local development and MUST n
ever be used in a production setup.

[OK] Web server listening
The Web server is using PHP CGI 8.1.8
https://127.0.0.1:8000
```

Lancez l'url : 127.0.0.1:8000

Si conflit vous pouvez modifier l'url ou le port (voir lien dans diapo précédente.
La commande possède une configuration interne qui vous lance directement votre site internet (comme notre framework perso)

STRUCTURE DE SYMFONY

Le DOSSIER «BIN» :

Ce dossier contient les exécutables disponibles dans le projet, que ce soit ceux fournis avec le framework (la console Symfony) ou ceux des dépendances (phpunit, simple-phpunit, php-cs-fixer, phpstan).

Le DOSSIER «CONFIG» :

Il contient toute la configuration de votre application, que ce soit le framework, les dépendances (Doctrine, Twig, Monolog) ou encore les routes.

Le DOSSIER «PUBLIC»:

Par défaut, il ne contient que le contrôleur frontal de votre application, le fichier dont la responsabilité est de recevoir toutes les requêtes des utilisateurs.

SEUL CE DOSSIER DOIT ETRE ACCESSIBLE DE L'EXTERIEUR.

Le DOSSIER «SRC» :

C'est ici que se trouve votre application ! Contrôleurs, formulaires, écouteurs d'événements, modèles et tous vos services doivent se trouver dans ce dossier. C'est également dans ce dossier que se trouve le "moteur" de votre application, le kernel.

STRUCTURE DE SYMFONY

Le DOSSIER «TESTS» :

Dans ce dossier se trouvent les tests unitaires, d'intégration et d'interfaces.

Par défaut, l'espace de nom du dossier tests est App\Tests et celui du dossier src est App .

Le DOSSIER «TEMPLATES» :

Ce dossier contient les gabarits qui sont utilisés dans votre projet. Par exemple, si dans un contrôleur on fait :

```
$this->render('page.html.twig');
```

Alors le gabarit sera localisé dans le dossier templates templates/page.html.twig .

Le DOSSIER «TRANSLATIONS»:

Symfony fournit un composant appelé Translation capable de gérer de nombreux formats de traductions, dont les formats yaml, xlf, po, mo... Ces fichiers seront situés dans ce dossier.

STRUCTURE DE SYMFONY

Le DOSSIER «VENDOR» :

Ce dossier contient votre chargeur de dépendances (ou "autoloader") et l'ensemble des dépendances de votre projet PHP installées à l'aide de Composer. Une autre façon de découvrir vos dépendances est d'utiliser la commande "composer show".

Le DOSSIER «VAR» :

Relatif au cache de votre application

Le DOSSIER «MIGRATIONS» :

Garde une trace de vos actions en rapport avec la base de données par exemple.
Afin de revenir en arrière en cas de problème.

SYMFONY FLEX

Flex est capable d'écouter les événements Composer, que ce soit l'installation, la mise à jour ou encore la suppression d'une dépendance.

Parmi les tâches qu'il est capable de réaliser :

- appliquer une configuration par défaut pour un plugin Symfony (autrement appelé "Bundle") ;
- création de fichiers/dossiers ;
- mise à jour de fichiers (par exemple le fichier config/bundles.php).

Imaginons par exemple que vous souhaitiez créer une API REST. Eh bien, avec Flex installé, il vous suffit d'exécuter la commande suivante :

composer require api

Et à l'aide de cette commande seulement, vous aurez une API REST installée et configurée

SYMFONY FLEX

Installation d'un bundle : sans Flex

- `composer require mon-package`
- Instancier le(s) bundle(s) dans le Kernel
- Créer la configuration dans `app/config/config.yml`
- Importer le routing dans `app/config/routing.yml`

Installation d'un bundle : avec Flex

- `composer require mon-package`
- ~~Instancier le(s) bundle(s) dans le Kernel~~
- ~~Créer la configuration dans `app/config/config.yml`~~
- ~~Importer le routing dans `app/config/routing.yml`~~

A noter depuis SF4 le dossier app n'existe plus, il est remplacé par config/

<https://symfonycasts.com/screencast/symfony4-upgrade/flex>

<https://afsy.fr/avent/2017/08-symfony-flex-la-nouvelle-facon-de-developper-avec-symfony>

LES VERSIONS DES DÉPENDANCES

```
"php": "^7.1.3",
"ext-ctype": "*",
"ext-iconv": "*",
"sensio/framework-extra-bundle": "^5.1",
"symfony/asset": "4.4.*",
"symfony/console": "4.4.*",
"symfony/dotenv": "4.4.*",
"symfony/expression-language": "4.4.*",
"symfony/flex": "^1.3.1",
"symfony/form": "4.4.*",
"symfony/framework-bundle": "4.4.*",
```

Valeur	Exemple	Description
Un numéro de version exact	4.0	Composer téléchargera cette version exacte.
Une plage de versions	>=4.0. 1,<5	Composer téléchargera la version la plus à jour, à partir de la version 4.0.1 et en s'arrêtant avant la version 5. Par exemple, si les dernières versions sont 4.0, 4.1 et 5.0, Composer téléchargera la version 4.1.
Une plage de versions sémantique	^4.1	Composer téléchargera la version la plus à jour, à partir de la version 4.1 et en s'arrêtant avant la prochaine version majeure, la 5.0. C'est une façon plus simple d'écrire " <code>>=4.1,<5.0</code> " avec la syntaxe précédente. C'est la façon la plus utilisée pour définir la version des dépendances.
Un numéro de version avec joker « * »	4.0.*	Composer téléchargera la version la plus à jour qui commence par 4.0. Par exemple, il téléchargerait la version 4.0.17, mais pas la version 4.1.1.
Un nom de branche « dev-XXX »	dev-master	C'est un cas un peu particulier, où Composer ira chercher la dernière modification d'une branche Git en particulier. N'utilisez cette syntaxe que pour les bibliothèques dont il n'existe pas de vraie version. Vous verrez assez souvent <code>dev-master</code> , où "master" correspond à la branche principale d'un dépôt Git.

Vous remarquerez beaucoup plus d'éléments dans votre package.json. Ces éléments auront seront utilisés par SF6 pour lancer des scripts, gérer l'autoloading, supprimer le cache, etc.. Ne rien supprimer «à la main» dans ce fichier.