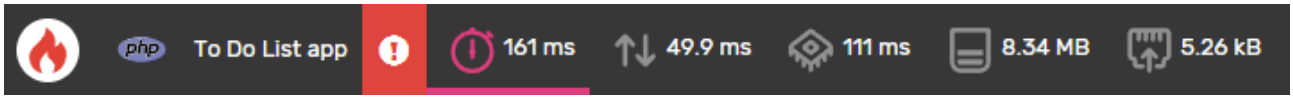
Sommaire

I - 1er test de performance



II - Recommandation de Symfony

Les éléments ci dessus proviennent de la documentation de Symfony, directement traduit.

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

II.1) OpCache pour Maximisé la performance

```
; php.ini
opcache.preload=/path/to/project/config/preload.php

; required for opcache.preload:
opcache.preload_user=www-data
```

II.2) Timestamps

En production, les fichiers PHP ne doivent jamais changer, à moins qu'une nouvelle version de l'application ne soit déployée. Cependant, par défaut, OPcache vérifie si les fichiers mis en cache ont changé leur contenu depuis leur mise en cache. Cette vérification crée une surcharge qui peut être évitée comme suit :

```
; php.ini
opcache.validate_timestamps=0
```

II.3) Configurer le cache PHP realpath

Lorsqu'un chemin relatif est transformé en son chemin réel et absolu, PHP met en cache le résultat pour améliorer les performances. Les applications qui ouvrent de nombreux fichiers PHP, comme les projets Symfony, doivent utiliser au moins ces valeurs

```
; php.ini
; maximum memory allocated to store the results
realpath_cache_size=4096K

; save the results for 10 minutes (600 seconds)
realpath_cache_ttl=600
```

III - Optimiser l'autoloader de composer

Les éléments suivants proviennent de la documenation de composer

https://getcomposer.org/doc/articles/autoloader-optimization.md

III.1) Optimisation de niveau 1 : Class map generation

« Class map generation » convertit essentiellement les règles PSR-4/PSR-0 en règles de classmap. Cela rend tout un peu plus rapide car pour les classes connues, la class map renvoie instantanément le chemin, et Composer peut garantir que la classe est là, donc aucune vérification du système de fichiers n'est nécessaire

Commande :

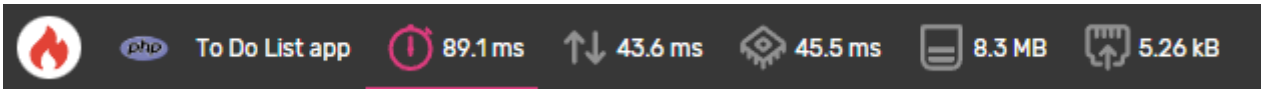
```
$ composer dump-autoload -o –optimize
```

III.2) Optimisation Niveau 2/A:Authoritative maps

L'activation de cette option active automatiquement les optimisations de classmap de niveau 1.Cette option indique que si quelque chose n'est pas trouvé dans le classmap, alors il n'existe pas et le chargeur automatique ne doit pas essayer de regarder le système de fichiers selon les règles PSR-4.

```
$ composer dump-autoload -a --classmap-authoritative
```

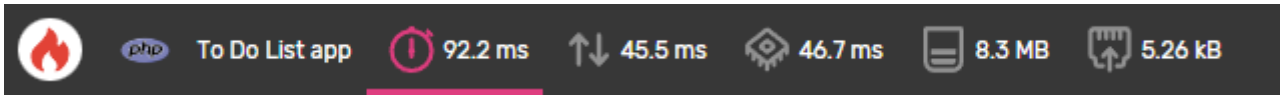
Après une combinaison d'optimisation de niveau 1 et 2/A, les résultats suivants sont obtenus



III.3) Optimisation Niveau 2/B:APCu cache

Cette option ajoute un cache APCu comme solution de secours pour la classmap. Cependant, il ne générera pas automatiquement la classmap, vous devez donc toujours activer les optimisations de niveau 1 manuellement si vous le souhaitez.Qu'une classe soit trouvée ou non, ce fait est toujours mis en cache dans APCu, de sorte qu'il peut être renvoyé rapidement lors de la prochaine requête.

Après une combinaison d'optimisation de niveau 1 et 2/B, les résultats suivants sont obtenus



Ici la combinaison niveau et niveau 1 et niveau 2A semble légèrement supérieur à celle de la combiniason niveau 1 et niveau 2B.

Nous choisrons donc l'optimisation Authoritative maps+ classmap generation (Niveau 2A+ niveau1)

Il n'est pas possible de combiner le niveau 2A et le niveau 2B.