

ToDo & Co - ToDoList

Audit de qualité

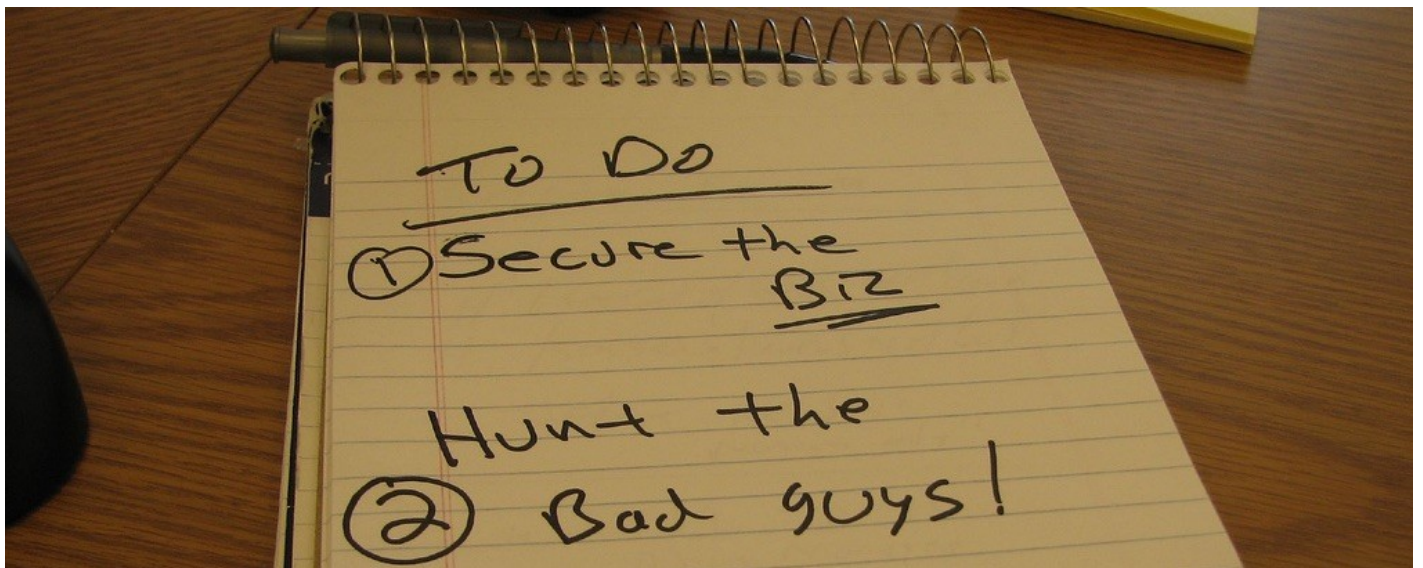


Table des matières

ToDo & Co – ToDoList.....	1
1. Contexte.....	3
Présentation.....	3
2.1 Version de Symfony.....	3
2.2 Qualité du code.....	4
2.2.1 Qualité du code.....	4
2.2.1 Codacy :.....	4
2.2.3 Symfony insight :.....	4
.....	4
2.2.3 Symfony insight suite :.....	5
2.3 PHP code sniffer (PHPCS).....	5
2.3 Conclusion.....	7
3. Performances.....	8
3.1 Analyse Blackfire.....	8
3.2 Optimisation.....	10
3.2.1 Composer.....	10
3.2.2 Mise en place de Opcache.....	11
4 Bilan.....	12
4.1 Optimisation.....	12
4.2 OPCache.....	13
.....	13
4.1 Optimisation du code.....	14
.....	14
.....	14

1. Contexte

Présentation

TodoList est une application appartenant à la jeune startup ToDo & Co.

L'application a dû être développée en urgence pour présenter le concept à de potentiels investisseurs.

La présentation a été un succès l'entreprise a réussi à lever des fonds pour le développement de l'application et son expansion.

2 .Dette technique

2.1 Version de Symfony

La version actuelle symfony qui constitue l'application ne satisfait plus, en effet la version 3.1.10 n'étaient plus maintenue par symfony cette dernière représente des risques en terme de sécurité . Elle sera donc mise a jours sur le dernière LSR (Latest Stable Release) a savoir là 5.3.10

Application avant modification :

Symfony	3.1.10
php	7.2.34
doctrine-bundle	1.6
doctrine-orm	2.5

Application après modification

Symfony	5.3,10
php	7.4
doctrine-bundle	2.0.7
doctrine-orm	2.7

2.2 Qualité du code

2.2.1 Qualité du code

Pour établir un rapport de qualité sur le code actuelle je me suis appuyé sur 3 outils,

1. Codacy
2. Symfony insight
3. PHP code sniffer

2.2.1 Codacy :

Codacy nous retourne la note global de C et de multiples issus (71 au totals) dont

1. 39 concernant des problèmes de sécurités
2. 29 de code styles
3. 3 Unused code (Code non utilisé)

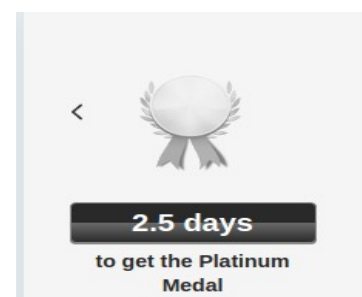
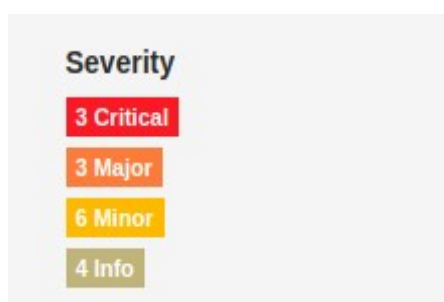
Catégories	Minor/Medium	Critique	Total
Security	23	16	39
Code Style	14	15	29
Unused code	3	0	3

Analyse disponible : <https://app.codacy.com/gh/dinasty-serv/ToDoList/dashboard?branch=master>

2.2.3 Symfony insight :

Symfony insight nous permet d'avoir plus d'informations sur les problèmes de sécurité de l'application, les résultats de l'analyse remonte des problèmes intrinsèquement lié a la version de symfony.

Symfony insight n'était pas en mesure d'appliquer une note à l'application, en causes 3 erreurs critique.



2.2.3 Symfony insight suite :

Liste des issus de sécurité lié a la version obsolète du framework Symfony.

▼ Your project must not rely on dependencies with known security issues

Read doc Ignore all Security Critical

The checker detected 21 security issues in package symfony/symfony installed in version 3.1.10.0

- 1) [CVE-2017-16652: Open redirect vulnerability on security handlers.](#)
- 2) [CVE-2017-16653: CSRF protection does not use different tokens for HTTP and HTTPS.](#)
- 3) [CVE-2017-16654: Intl bundle readers breaking out of paths.](#)
- 4) [CVE-2017-16790: Ensure that submitted data are uploaded files.](#)
- 5) [CVE-2018-11385: Session Fixation Issue for Guard Authentication.](#)
- 6) [CVE-2018-11386: Denial of service when using PDOSessionHandler.](#)
- 7) [CVE-2018-11406: CSRF Token Fixation.](#)
- 8) [CVE-2018-11407: Unauthorized access on a misconfigured LDAP server when using an empty password.](#)
- 9) [CVE-2018-11408: Open redirect vulnerability on security handlers.](#)
- 10) [CVE-2018-14773: Remove support for legacy and risky HTTP headers.](#)
- 11) [CVE-2018-19789: Temporary uploaded file path disclosure.](#)
- 12) [CVE-2018-19790: Open Redirect Vulnerability on login.](#)
- 13) [CVE-2019-10909: Escape validation messages in the PHP templating engine.](#)
- 14) [CVE-2019-10910: Check service IDs are valid.](#)
- 15) [CVE-2019-10911: Add a separator in the remember me cookie hash.](#)
- 16) [CVE-2019-10912: Prevent destructors with side-effects from being unserialized.](#)
- 17) [CVE-2019-10913: Reject invalid HTTP method overrides.](#)
- 18) [CVE-2019-18887: Use constant time comparison in UriSigner.](#)
- 19) [CVE-2019-18888: Prevent argument injection in a MimeTypeGuesser.](#)
- 20) [CVE-2019-18889: Forbid serializing AbstractAdapter and TagAwareAdapter instances.](#)
- 21) [CVE-2021-21424: Prevent user enumeration via response content in authentication mechanisms.](#)

Time to fix: about 1 day

Read doc Comment Ignore Open Issue Permalink

2.3 PHP code sniffer (PHPCS)

PHPCS permet d'analysé les différents fichier/class (PHP,HTML,CSS ,JAVASCRIPTS) de l'application afin détecté les manquements au norme de codage, en effet dans le développement web (Comme dans la plupart des cors de métiers) il y a des règles,norme, lois qu'il faut respecter afin de garantir une application facilement maintenable et qualitatifs.

Cette analyse remonte différents problème en terme de code PHP, les plus remonté sont :

```
FILE: /home/ndefontaine/Project/docker-p8/todo/src/AppBundle/Entity/Task.php
-----
FOUND 27 ERRORS AFFECTING 23 LINES
-----
 2 | ERROR | Missing file doc comment
 8 | ERROR | Missing short description in doc comment
11 | ERROR | Missing @category tag in class comment
11 | ERROR | Missing @package tag in class comment
11 | ERROR | Missing @author tag in class comment
11 | ERROR | Missing @license tag in class comment
11 | ERROR | Missing @link tag in class comment
14 | ERROR | Missing short description in doc comment
19 | ERROR | Private member variable "id" must be prefixed with an underscore
21 | ERROR | Missing short description in doc comment
24 | ERROR | Private member variable "createdAt" must be prefixed with an underscore
26 | ERROR | Missing short description in doc comment
30 | ERROR | Private member variable "title" must be prefixed with an underscore
32 | ERROR | Missing short description in doc comment
36 | ERROR | Private member variable "content" must be prefixed with an underscore
38 | ERROR | Missing short description in doc comment
41 | ERROR | Private member variable "isDone" must be prefixed with an underscore
43 | ERROR | Missing doc comment for function __construct()
49 | ERROR | Missing doc comment for function getId()
54 | ERROR | Missing doc comment for function getCreatedAt()
59 | ERROR | Missing doc comment for function setCreatedAt()
64 | ERROR | Missing doc comment for function getTitle()
69 | ERROR | Missing doc comment for function setTitle()
74 | ERROR | Missing doc comment for function getContent()
79 | ERROR | Missing doc comment for function setContent()
84 | ERROR | Missing doc comment for function isDone()
89 | ERROR | Missing doc comment for function toggle()
-----
```

2.3 Conclusion

Nous ne pouvons tout simplement pas proposer un produit de qualité avec un framework obsolète !

La première étape avant de commencer les différents développements lié à l'amélioration de l'application est la mise à jours de Symfony et de ces dépendances.

La qualité du code doit être amélioré, et l'ajout de commentaires pour décrire les différentes fonction (surtout dans les controller) est vivement recommandé afin de garantir un code facilement compréhensible pour nos collaborateurs. Enfin pour assurer le suivis du développement le projet devras disposer d'un système de versionning (git).

3. Performances

3.1 Analyse Blackfire



Blackfire est un outil qui nous permet de savoir :

- le temps de chargement de l'application,
- la mémoire utilisée pour son chargement
- les fonctions utilisées et son nombre d'appels.

Chaque fonction est détaillée avec son temps d'exécution et la mémoire utilisée. Un diagramme est consultable pour avoir le plan d'exécution de l'application, avec une couleur spécifique pour les fonctions les plus gourmandes. Les tests ici présents, sont faits avec la licence gratuite de blackfire, ce qui limite l'analyse du site.

Analyse des pages de l'application avant les optimisations

Page	Temps (en ms)	Mémoire
/login	32ms	3,94MB
/users/create	41,3ms	5,37MB
/	36,2ms	4,73MB
/tasks/create	45,6ms	5,84MB
/tasks	38,8ms	4,79MB

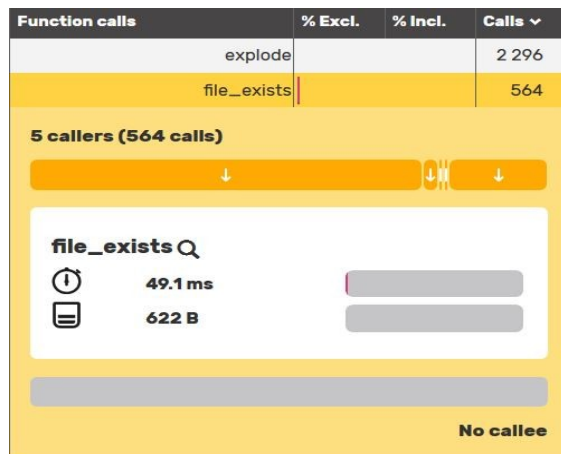
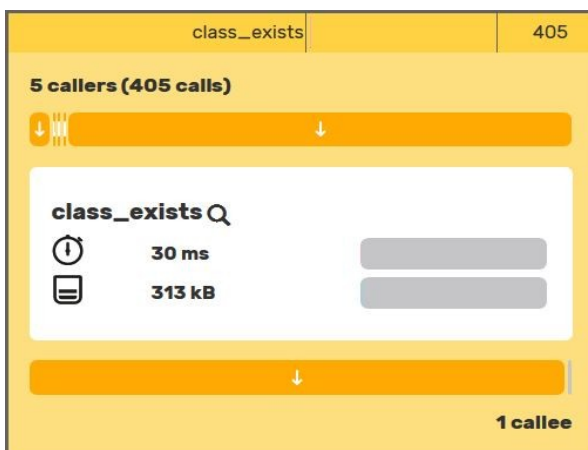
On constate une moyenne de 38,78MS de temps d'exécution et 4,93 MB de consommation de mémoire.

Lors de la première analyse nous avons pu constater un nombre très important d'appels sur les fonctions :

- file_exist
- class_exist.

Temps d'exécution	Mémoire utilisé
41,8 MS	552B
29,5MS	313KB

Ces données nous montre une mauvaise optimisation de composer



3.2 Optimisation

3.2.1 Composer

L'optimisation de composer est la première étape pour gagner en performance simplement, l'exécution de la commande ci-dessous permet de mettre en cache les classes chargées par composer

```
composer dump-autoload --no-dev --classmap-authoritative
```

- **--no-dev** exclut les classes qui ne sont nécessaires que dans l'environnement de développement (c'est-à-dire les dépendances require-dev et les règles autoload-dev)
- **--classmap-authoritative** crée un mappage de classe pour les classes compatibles PSR-0 et PSR-4 utilisées dans votre application, et empêche Composer d'analyser les classes qui ne se trouvent pas dans le mappage de classe

L'autoloader charge automatiquement les classes utiles à l'application. Dans le cas de composer, l'autoloader est dans un dossier « vendor » à la racine du projet.

Avec la commande écrite un peu plus haut, des fichiers sont créés pour faire une carte des classes à charger pour l'application, ce qui évite de charger tous les chemins à chaque chargement de pages.

Documentation :

<https://getcomposer.org/doc/articles/autoload-optimization.md>

3.2.2 Mise en place de Opcache

À partir de PHP 7.4, OPcache peut compiler et charger des classes au démarrage et les rendre disponibles pour toutes les requêtes jusqu'au redémarrage du serveur, améliorant considérablement les performances.

Lors de la compilation du conteneur (par exemple lors de l'exécution de (`cache:clear`),

Symfony génère un fichier avec la liste des classes à précharger dans le répertoire `var/cache/`.

Plutôt que d'utiliser ce fichier directement, utilisez le fichier `config/preload.php` créé lors de l' [utilisation de Symfony Flex](#),

4 Bilan

4.1 Optimisation

Voici les résultats de l'optimisation après :

1. Migration de symfony 3 vers 5
2. Mise en place de OPCACHE
3. Optimisation de l'autoloader
4. Refactoring du code métier

	Avant optimisation		Après optimisation	
Page	Temps (en ms)	Mémoire	Temps (en ms)	Mémoire
/login	32ms	3,94MB	17,8MS	2,97MB
/users/create	41,3ms	5,37MB	37ms	5,2MB
/	36,2ms	4,73MB	23,6ms	3,9MB
/tasks/create	45,6ms	5,84MB	35ms	4,93MB
/tasks*	38,8ms	4,79MB	25,3ms	4MB

Suite aux optimisations ont observe une nette amélioration du temps de chargement et de la mémoire consommé.

le temps d'exécution a chuté de environ 65 % et la consommation de données de 18 %.

4.2 OPCache

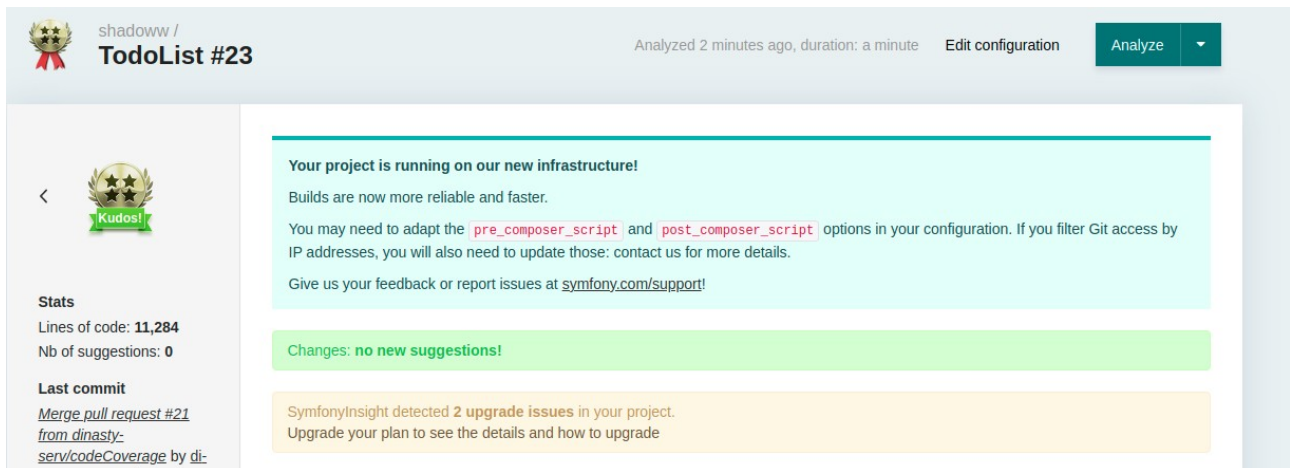
« OPCache améliore les performances de PHP en stockant le bytecode des scripts pré-compilés en mémoire partagée, faisant ainsi qu'il n'est plus nécessaire à PHP de charger et d'analyser les scripts à chaque demande. »
(src : <https://fr.docs.wp-rocket.me/article/675-quest-ce-que-opcache>)

Liens pour optimisation par OPCache : <https://www.ekino.fr/articles/php-comment-configurer-utiliser-et-surveiller-opcache>

4.1 Optimisation du code

Après avoir effectué la migration de symfony, et ré-factorisé le code voici les notes attribué par symfony insight et codacy

symfony insight



The image shows the Symfony Insight analysis interface for a project named 'shadowww / TodoList #23'. The interface includes a sidebar with project statistics and a main content area with analysis results.

Stats

- Lines of code: **11,284**
- Nb of suggestions: **0**

Last commit

[Merge pull request #21 from dynasty-serv/codeCoverage](#) by di-

Analysis Results:

- Your project is running on our new infrastructure!**
Builds are now more reliable and faster.
You may need to adapt the `pre_composer_script` and `post_composer_script` options in your configuration. If you filter Git access by IP addresses, you will also need to update those: contact us for more details.
Give us your feedback or report issues at symfony.com/support!
- Changes: no new suggestions!**
- SymfonyInsight detected 2 upgrade issues in your project.**
Upgrade your plan to see the details and how to upgrade

Codacy

