Audit de qualité

ToDo & Co est une startup dont le cœur de métier est une application de gestion des tâches quotidiennes.

L’entreprise vient tout juste d’être crée, et l’application a dû être développée rapidement pour permettre de présenter à de potentiels investisseurs que le concept est viable (Minimum Viable Product ou MVP) avec le Framework Symfony dans sa version 3.1.

La mission qui m’a été confier et de développer de nouvelles fonctionnalités, et d’analyser la qualité du code ainsi que ses performances.

Ce document explique les améliorations apportées au code et compare les performances avant et après modifications.

Sommaire

1 – Le code initial

* *1.1 Les premières optimisations : Upgrader la version de Symfony et PHP*
* *1.2 Mesure de performance avant et après l’upgrade (BlackFire)*

2 – Améliorations

* *2.1 Coté Backend*
* *2.2 Coté Frontend*

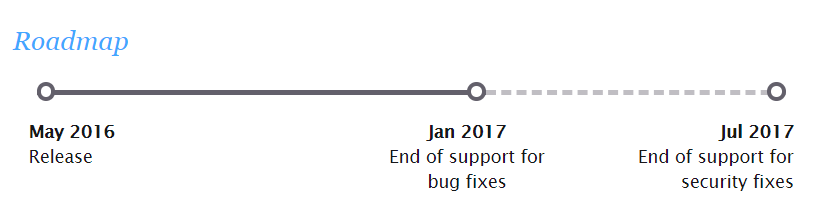
3 – Tests et performance du nouveau code

* Test unitaire et fonctionnel (PHP Unit)
* Test fonctionnel (Selenium)
* Mesure de performance finale (BlackFire)
* 1 – Le code initial

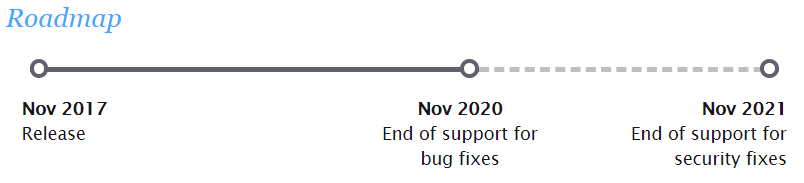
**1.1 Première optimisation : Upgrader la version de Symfony**

La version de Symfony initial de l’application est la 3.1.

Cette version à été mis en ligne en mai 2016 puis maintenu à jour jusqu’en janvier 2017 pour les corrections de bugs et jusqu’en juillet 2017 pour les correctifs de sécurité.



Passer sur la version 3.4 (*LTS = Long Term Support*) de Symfony permet de recevoir des mises à jour jusqu’en Novembre 2020 et des correctifs de sécurité jusqu’en Novembre 2021.



**Seconde optimisation : Upgrader la version de PHP**

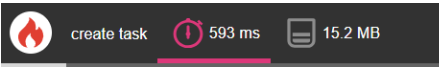
La version de PHP utilisé est la 5.5.9, passer à la version 7.2 de PHP augmentera les performances au niveau de l’utilisation des ressources et offrira la possibilité d’utilisé les dernières fonctionnalités de PHP comme le typage des variables qui rendra le code plus stable.

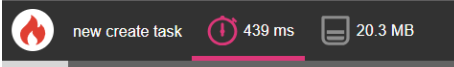
Comparaison des performances de Symfony 3.1 avec PHP 5 et la version 3.4 de Symfony avec PHP 7.2 d’après le site php benchmarks : <http://www.phpbenchmarks.com/fr/benchmark/apache-bench/php-7.2/select-version/symfony.html>



**1.2 Mesure de performance : Avant et après l’upgrade**

L’outil utilisé pour les mesures est [BlackFire.io](https://blackfire.io/) . Outil développé par Sensio Lab créateur du framework Symfony spécialisé dans les mesures d’applications PHP.

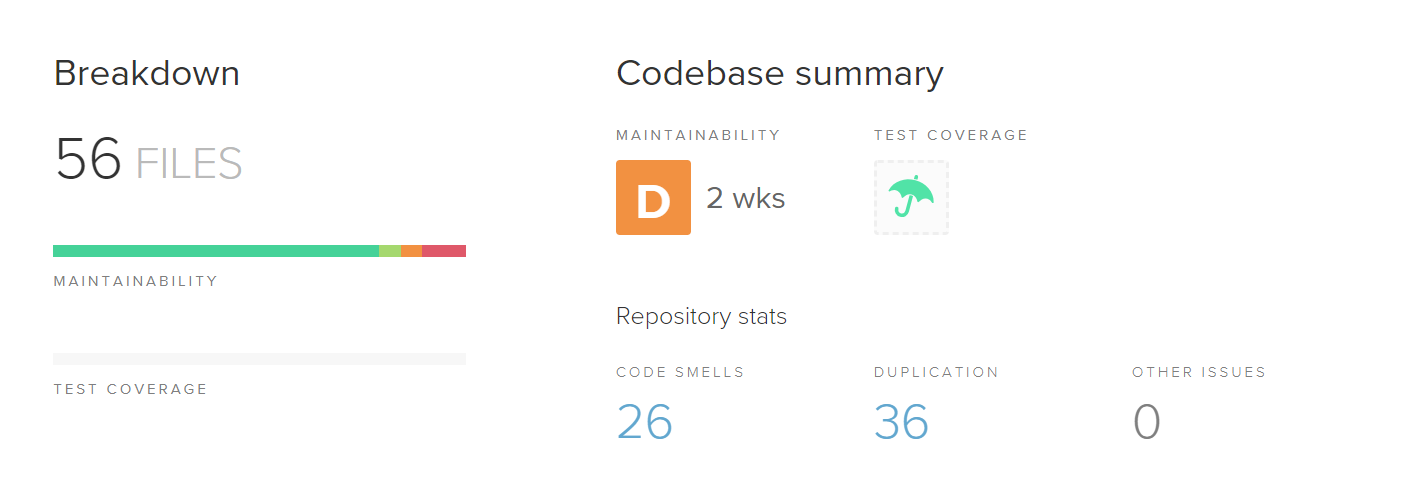




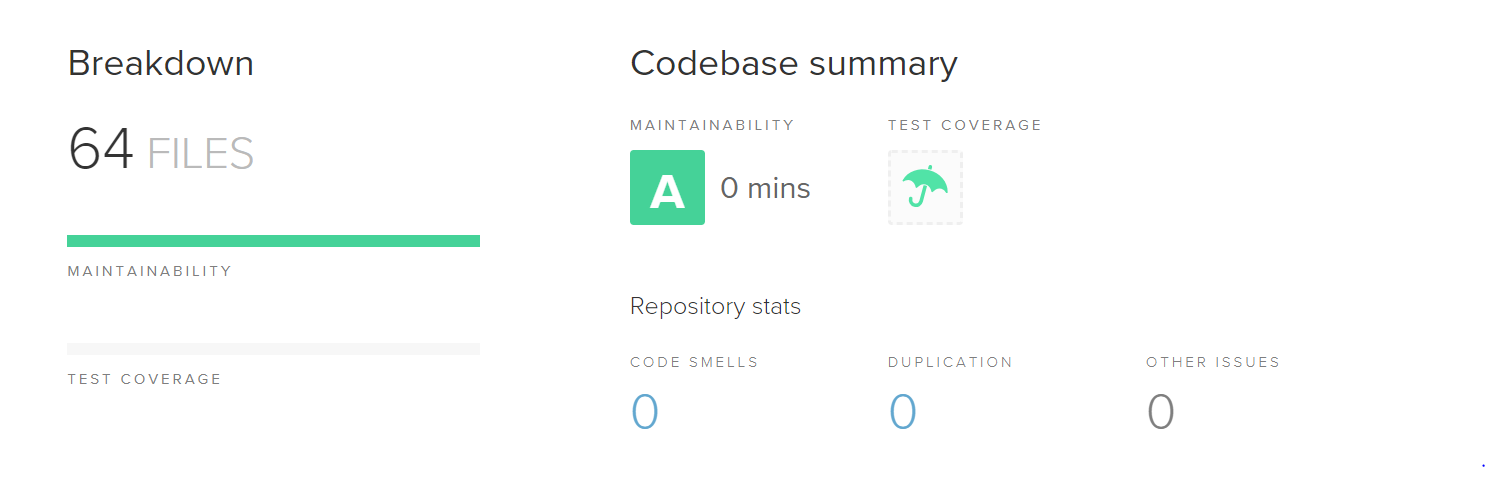
Mesure BlackFire de la page Create Task avant et après l’upgrade (en environnement de développement)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Route | Temps 1 | Mémoire 1 | Temps 2 | Mémoire 2 |
| /tasks |  |  |  |  |
| /users |  |  |  |  |
| /users/create |  |  |  |  |
| /tasks/create |  |  |  |  |
| /login |  |  |  |  |
| / |  |  |  |  |

* Analyse de la qualité de code avec CodeClimate :



1 Avant la modification



Après la modification

2 – Améliorations

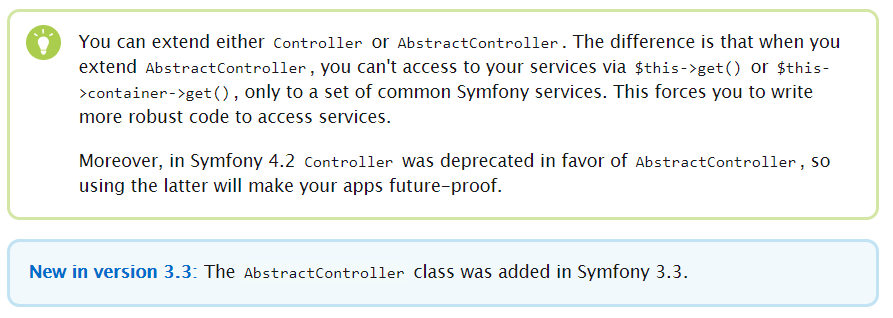
**2.1 Améliorations coté Backend**

Plusieurs modifications ont été apportées sur les contrôleurs.

* Modification des contrôleurs :

Les contrôleurs étendaient de la classe **Controller** (classe déprécié sur les nouvelles versions de Symfony elle et remplacer par la classe **AbstractControler)** .

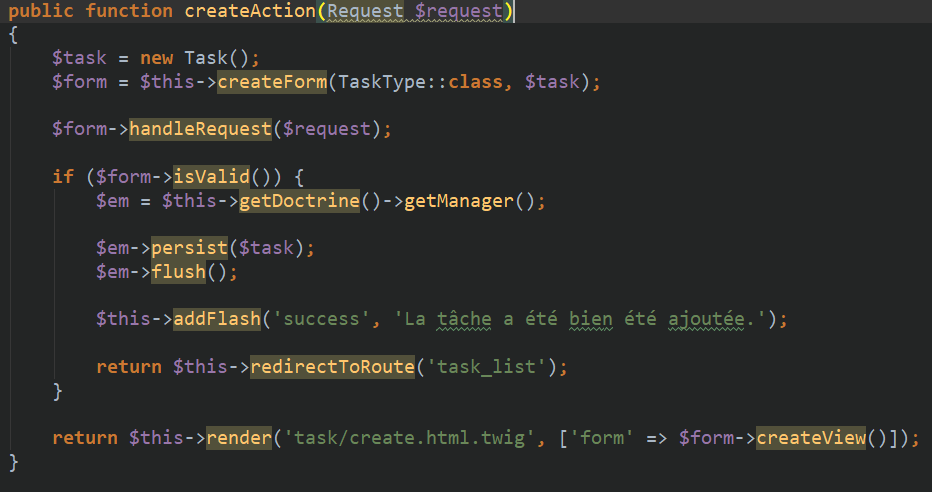
La classe Controller à donc été supprimer, mais pour de meilleur performance je n’ai pas étendu la classe vers sa remplaçante mais plutôt choisi d’injecter les services vraiment utiles aux contrôleurs.



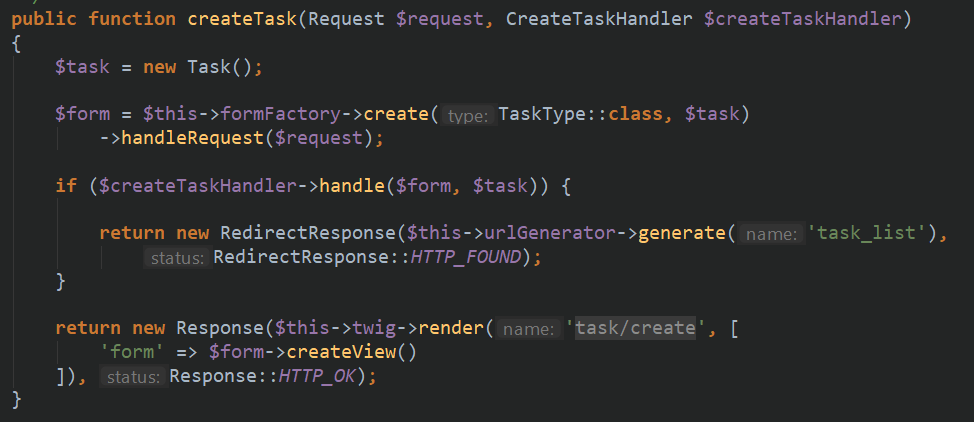
*Réf :* [*https://symfony.com/doc/3.4/controller.html#the-base-controller-classes-services*](https://symfony.com/doc/3.4/controller.html#the-base-controller-classes-services)

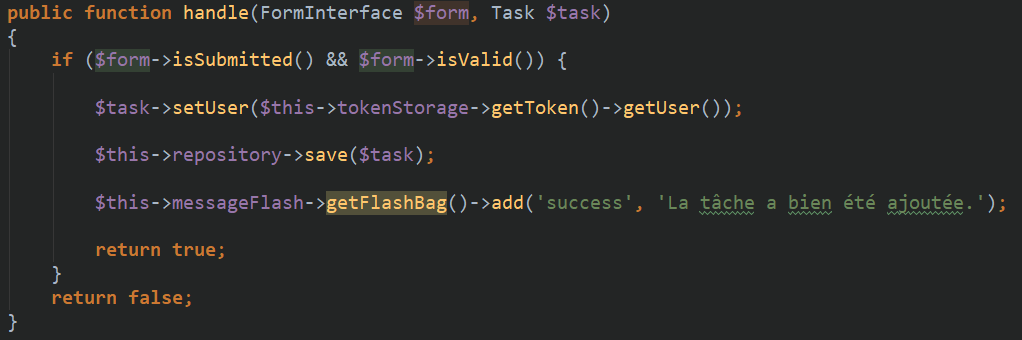
* Ajout de Handler :

Il est une bonne pratique de ne pas laissez toute la logique métier dans les contrôleurs. J’ai donc déplacé le traitement des formulaires dans des classes **Handler** (AppBundle/FormHandler).



3 - Contrôleur avant modification





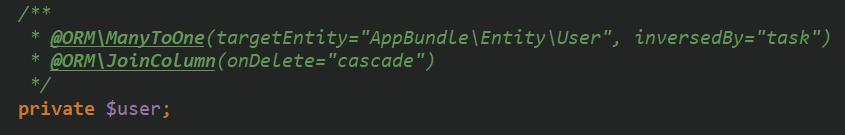
- Contrôleur et son handler après modification

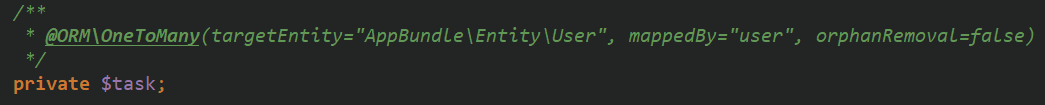
* Modification du **SecurityController** :

Suppression des méthodes sans retour **Login\_Check** et **Logout** qui indiqué seulement la route pour la vérification du login et celle de la déconnexion. Les routes on été ajouté dans le fichier **routing.yml** (app/config/routing.yml).

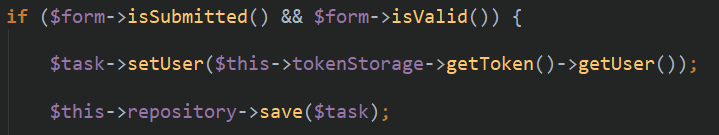
* Correction de l’anomalie à l’ajout d’une tâche :

Ajout d’une relation entre l’entité **Task** et **User**



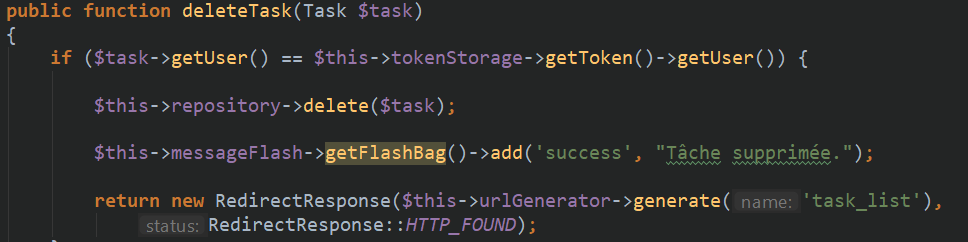


Dorénavant lors de l’ajout d’une tâche celle-ci est rattaché à l’utilisateur connecté (donc son auteur).



On récupère l'utilisateur actuellement connecté pour l'associé à la tâche avant sa sauvegarde en BDD

* Avec cette relation entre l’utilisateur et ses tâches celui-ci peut *visualiser, modifier, supprimer et marquer*  **uniquement** ses tâches. Une vérification est faite avant chaque action, le système vérifie que la tâche appartient bien à l’utilisateur connecté.



Exemple avec la fonction de suppréssion d’une tâche . On vérifie que l'utilisateur connecté est bien l'auteur de la tâche

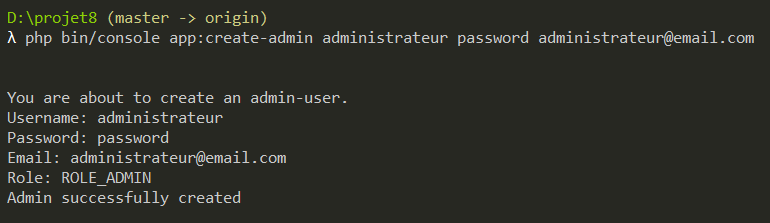
* Type Hinting :

La version initial de PHP (5) ne permettait pas de typé les retours de méthodes ou des paramètres. Le typage à été ajouté suite au passage de la version 7 de PHP.

* Validation password:

Ajout de la validation sur le mot de passe utilisateur car aucune été paramétré initialement.

* La commande **CreateAdminCommand** (dossier : AppBundle/Command/) pour facilité l’ajout d’un administrateur à l’application en ligne de commande.



**2.2 Améliorations coté Frontend**

* La version de Bootstrap initial été la 3, je l’ai remplacé par la version 4 via CDN qui améliore le chargement et qui apporte les dernières nouveautés du langage CSS offrant un design plus moderne avec des fonctionnalités plus poussé.

Suppression du code inutile dans le fichier Shop-homepage.css.

* Ajout d’une bannière obligatoire pour mentionner aux visiteurs que l’application utilise des cookies.
* Page tâche terminé :

Ajout d’une page pour visualiser les tâches terminées (/tasks-Is-Done).