Rapport d’audit de l'application :

Todo List

Version 1.0 mai 2024

Sommaire

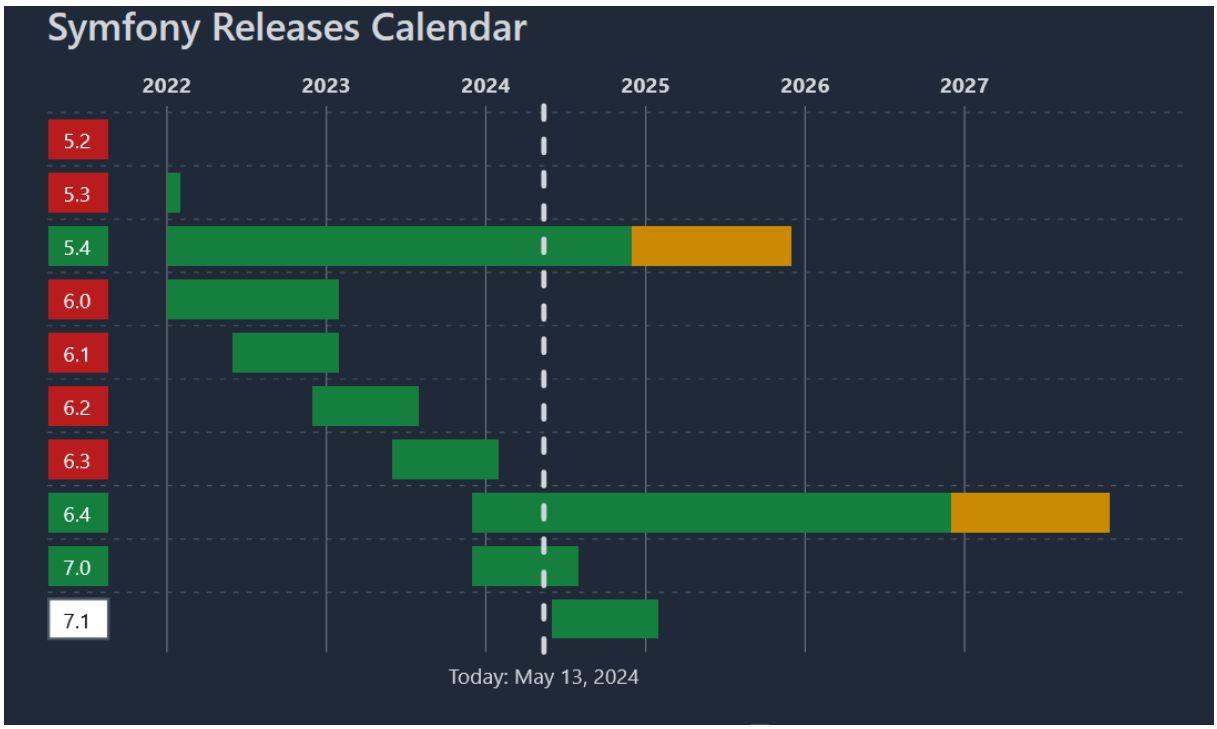
Table des matières

Introduction : Rappel du contexte

Dans le cadre de la reprise d'un projet existant lié à nouvel apport financier, nous avons été mandatés pour établir un état des lieux de l'application autant d'un point de vue technique que du point de vue utilisateur. Par la suite un diagnostique de performance sera établit et permettra l'amélioration de l'application

1. Migration de la version de symfony

La première migration a effectué s'est faite sur la version mineure de symfony de 3.1 à 3.4. Ensuite pour une meilleure intégrité de l'application le projet a été importé directement dans une version de symfony 6.4.7 qui correspond à la Long-Term Support Release de Symfony (https://symfony.com/releases).



Les modifications nécessaires ont été effectué suivant les fonctions dépréciées les bundles intégrés à Symfony et les demandes spécifiques du projet.

Le projet nécessite au minimum php 8.1 pour fonctionner. Nous avons utilisé php 8.1.14 et notre base de donnée est une base MySql 5.7.36 .

2. Les anomalies à corriger

Dans l'ordre de mission plusieurs fonctionnalités nécessitaient d'être retravaillé :

Les tâches doivent être automatiquement attachées à un utilisateur lors de leur création. L'auteur des tâches ne peut pas être modifié lors de leur modification.

Les tâches déjà créées doivent être rattachées à un utilisateur "anonyme".

Ajout de la possibilité de choisir un rôle pour un utilisateur lors de sa création (utilisateur ou administrateur).

Autorisation :

- Seuls les utilisateurs avec le rôle administrateur peuvent accéder aux pages de gestion des utilisateurs.
- Les utilisateurs peuvent supprimer uniquement les tâches qu'ils ont créées.
- Les tâches rattachées à l'utilisateur "anonyme" peuvent être supprimées uniquement par les administrateurs.

3. Les anomalies découvertes et corrigés

Durant la migration plusieurs points ont nécessité une attention particulière.

Les librairies et recettes du projets n'était pas à jour. Leur update via composer était nécessaire. Les dernières mise à jour nécessitait également une mise à jour complète du code pour s'adapter aux nouvelles syntaxes et règles jugées plus efficaces dans les dernières versions.

Ainsi voici une liste des changements apportés :

- La structure générale est modifiée pour correspondre à celle de Symfony 6.4.7.
- Les annotations des méthodes des contrôleurs sont en accords avec la dernière version de Symfony.
- La validation des formulaires nécessitent désormais l'utilisation de la fonctions isSubmitted() en plus de la fonctions isValid().

La précédente version ne prévoyait pas de restriction d'accès en fonction du rôle des utilisateurs.

De même le cas de tâches sans utilisateurs n'était pas permis.

4. Les fonctionnalités ajoutées

Dans un souci d'une meilleure expériences utilisateurs plusieurs modifications ont été apportées en plus de celles demandées. L'idée générale était de présenter toujours les liens de navigation quelques soit l'endroit où l'on se trouve sur le site et d'afficher plus clairement les informations disponibles sur les tâches.

Une barre de navigation a été ajouté en vérifiant que celle-ci propose tous les liens accessibles à chaque visiteur du site en fonction de son statut.

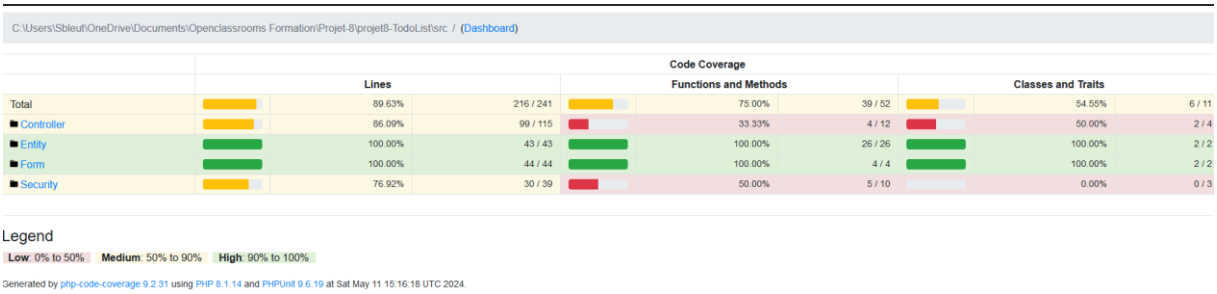
Les tâches sont séparées en deux pages tâches terminées et à faire, chacune disponible dans la barre de navigation. De même le passage d'une tâche d'un statut à l'autre nous emmène sur la page respective de ce statut.

Les tâches anonymes sont mentionnées avec un statut particulier supplémentaire affiché en gris pour plus de clarté.

5. Les tests unitaires et fonctionnelles

Les tests de bases pour chaque action du site ont été implantés. La validité des Entités et des contrôleurs est vérifiée séparément. Une base de données de test a été crée en vérifiant que les données factices sont suffisantes pour effectuer les tests.

Mise en place de tests unitaires et fonctionnels avec PHPUnit.



6. Qualité du code codacy & PSR1

L'analyse de la qualité du code est assurée en utilisant l'outil codacy qui permet une indication des différentes erreurs et non-respect des bonnes pratiques.

https://app.codacy.com/gh/Sbleut/todolist/dashboard?branch=dev

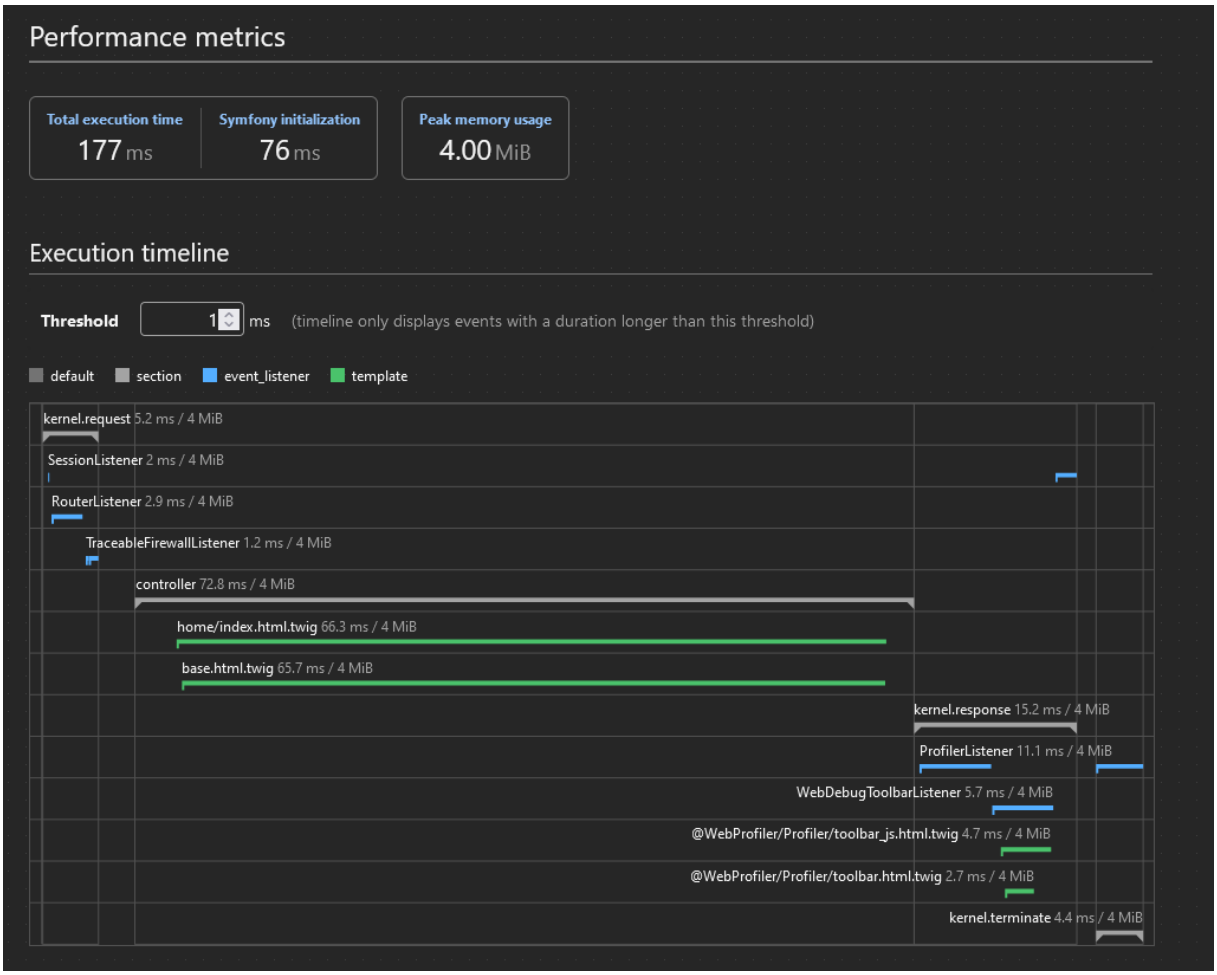
Codacy a permis de corriger les parties où du code inutilisé était présent, mais également de respecter la disposition, et l'indentation du code.

7. Performance

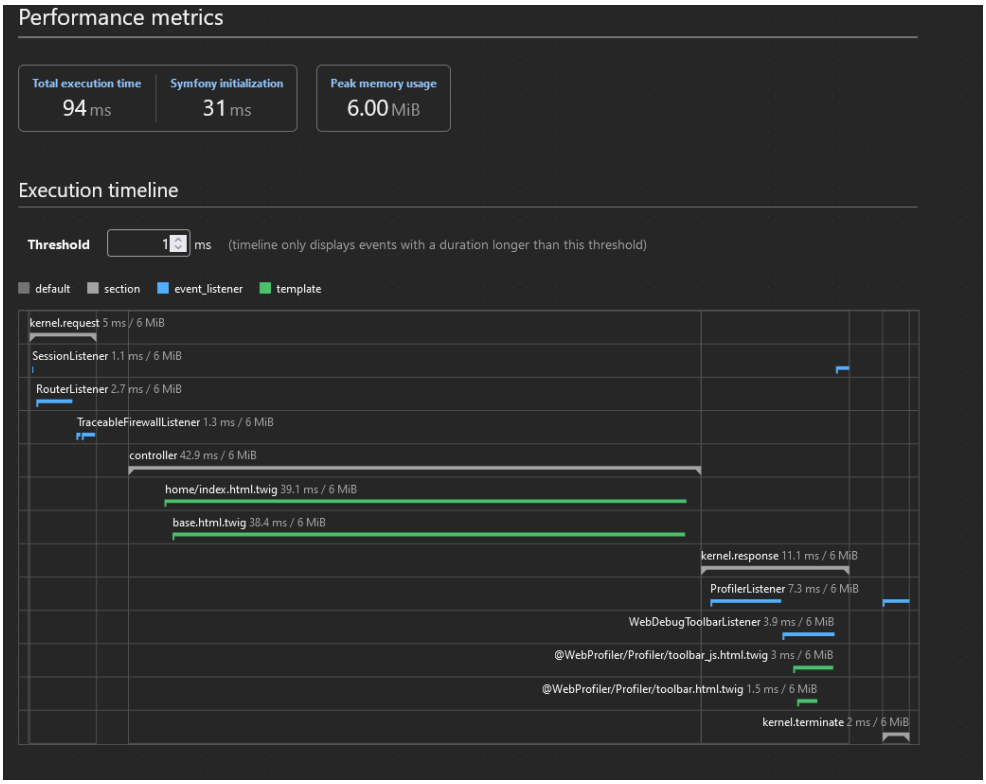
L'optimisation des performances actuelle a été effectué via le calcul de l'objet classmap qui indique l'emplacement de chaque classe php utilisée. Tant que l'application est en développement il serait un peu laborieux de recalculer les changement d'emplacement des classe utilisée, c'est pour ça qu'il est recommandé de le faire à la sortie de nouvelle version avec la commande suivante :

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

Voici les perfomrance avant optimisation :


Performance metrics
Total execution time
177 ms
Symfony initialization
76 ms
Peak memory usage
4.00 MiB
Execution timeline
Threshold
1
ms
(timeline only displays events with a duration longer than this threshold)
default
section
event_listener
template
kernel.request 5.2 ms / 4 MiB
SessionListener 2 ms / 4 MiB
RouterListener 2.9 ms / 4 MiB
TraceableFirewallListener 1.2 ms / 4 MiB
controller 72.8 ms / 4 MiB
home/index.html.twig 66.3 ms / 4 MiB
base.html.twig 65.7 ms / 4 MiB
kernel.response 15.2 ms / 4 MiB
ProfilerListener 11.1 ms / 4 MiB
WebDebugToolbarListener 5.7 ms / 4 MiB
@WebProfiler/Profiler/toolbar_js.html.twig 4.7 ms / 4 MiB
@WebProfiler/Profiler/toolbar.html.twig 2.7 ms / 4 MiB
kernel.terminate 4.4 ms / 4 MiB

Après optimisation on constate que le temps a réduit :



Pour les prochaines optimisations, il est possible
1. Compiler le conteneur de services en un seul fichier

Par défaut, Symfony compile le conteneur de services en plusieurs petits fichiers. Compiler l'ensemble du conteneur en un seul fichier peut améliorer les performances, en particulier lorsque vous utilisez le préchargement des classes avec PHP 7.4 ou des versions plus récentes.

Pour ce faire, définissez le paramètre `inline_factories` à `true` dans votre fichier `config/services.yaml` :

```
yaml
# config/services.yaml
parameters:
    # ...
    .container.dumper.inline_factories: true
```

Le paramètre `.container.dumper.inline_factories` est uniquement utilisé lors de la compilation du conteneur.

2. Utiliser et paramétrer OPcache

OPcache stocke les fichiers PHP compilés pour éviter de les recompiler à chaque requête. Depuis PHP 5.5, OPcache est intégré à PHP. Assurez-vous que OPcache est activé dans votre fichier `php.ini` :

```
ini
; php.ini
```

```
zend_extension=opcache.so
opcache.enable=1
```

En configurant correctement OPcache, vous pouvez grandement améliorer les performances de votre application Symfony.