*Tester et suivez l’état de votre application avec PHP*

**Mise en place d’un outil pour implémenter ses tests unitaires**

* Un test se décompose en trois actions :

1. Instanciation d’une classe
2. Appel d’une méthode avec passage ou non de paramètres
3. Vérification de la sortie

Pour lancer les tests on utilise dans la console la commande suivante **:  « vendor/bin/phpunit »**

* Le code coverage est un rapport indiquant si toutes les parties de notre code sont couverts par les tests ou non.

Pour récupérer le code coverage : **« «vendor/bin/phpunit –coverage-html web/test-coverage »**

**/!\ Attention :** Il faut que xdebug soit installé pour que le code coverage soit généré.

Il n’est pas forcement pertinent de tester la totalité du code, car cela peut s’avérer très chronophage ! Il faut en premier lieu se concentrer sur la logique centrale et/ou sensible de l’application.

**Premiers pars avec PHP Unit et les tests unitaires**

* Un test unitaire consiste à exécuter du code en provenance de l’application et à vérifier si tout s’est bien passé (résultat de sortie cohérent et/ou fonctions sensées être appelées correctement appelées). Les tests unitaires sont dits « tests boite blanche » parce qu’il faut en amont connaitre le code pour pouvoir le tester.
* Le nombre de tests à écrire pour tester une fonction donnée équivaut au nombre de « return » possibles de la dite fonction, additionnés à son nombre d’exceptions possibles. De la sorte on est sûr que tous les cas sont couverts.
* Un test doit être écrit dans une classe portant le même nom que la classe cible suffixé par « test » et contenue dans le dossier « tests » du projet tout en respectant l’arborescence de la classe cible : **« /tests/Bundle/Entity/CibleTest.php »**
* Les méthodes de test, elles, peuvent prendre n’importe quel nom mais sont obligatoirement préfixé par « test ».
* Il est possible, grâce aux data providers de passer un jeu de paramètres à nos tests. De la sorte PHPUnit va exécuter plusieurs fois le même test mais avec différentes valeurs en entrée, passé par le data provider.
* Si on ne veut pas exécuter tous les tests mais juste une seule fonction on peut utiliser l’option **« --filter=testFunction »**

**Les doublures (mocks)**

// …