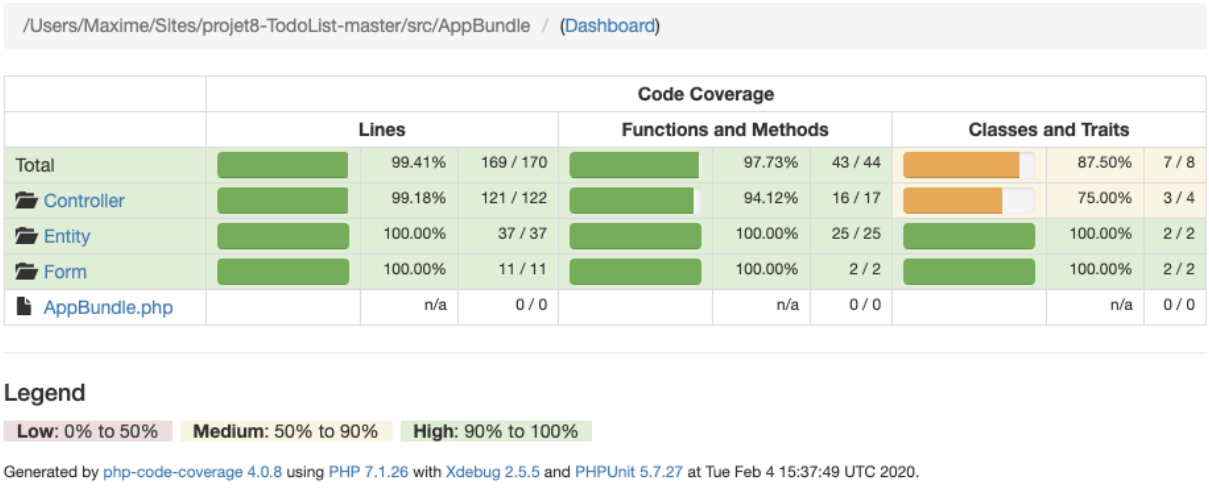
Rapport de couverture des tests unitaires et fonctionnels.

Ce document a pour but d'analyser le résultat des tests unitaires et fonctionnels effectués à travers le bundle PHP Unit.

Table des matières

I. Résultat final



/Users/Maxime/Sites/projet8-TodoList-master/src/AppBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		99.41%	169 / 170		97.73%	43 / 44		87.50%	7 / 8
Controller		99.18%	121 / 122		94.12%	16 / 17		75.00%	3 / 4
Entity		100.00%	37 / 37		100.00%	25 / 25		100.00%	2 / 2
Form		100.00%	11 / 11		100.00%	2 / 2		100.00%	2 / 2
AppBundle.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0

Legend

Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

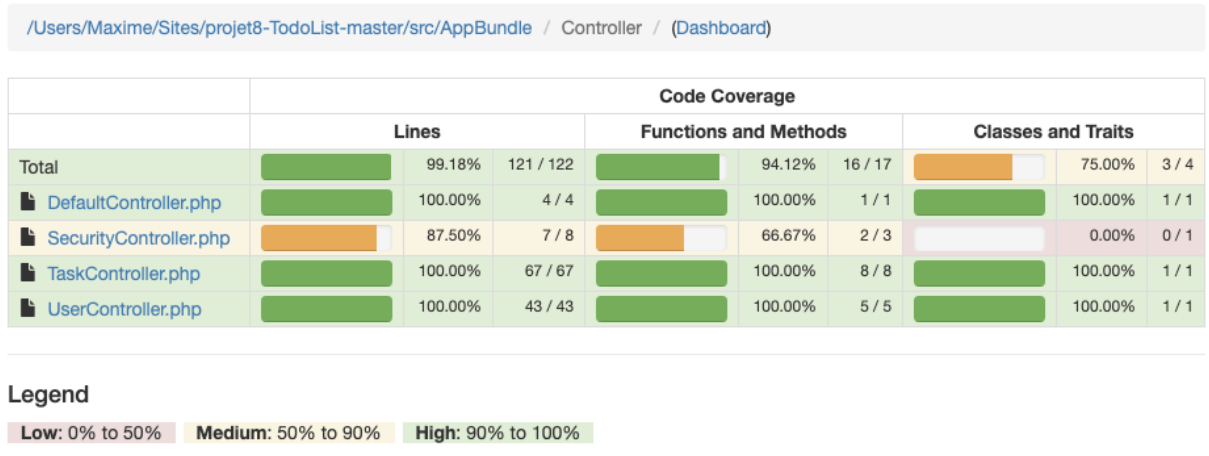
Generated by php-code-coverage 4.0.8 using PHP 7.1.26 with Xdebug 2.5.5 and PHPUnit 5.7.27 at Tue Feb 4 15:37:49 UTC 2020.

Comme on peut le voir sur le rapport final dans l'ensemble c'est plutôt positif avec une couverture de test supérieure à 80% :

- 99,41% pour Lines
- 97,73% pour Functions & Methods
- 87,50% pour Classes & Traits

On constate que les dossiers Entity et Form atteignent même les 100% de couverture. Pour ce qui est du dossier Controller, on constate que dans son ensemble toutes les méthodes semblent être couvertes sauf une (16/17). Allons un peu plus loin dans le détail pour comprendre cette méthode non couverte en analysant le dossier Controller.

II. Résultat Dossier Controller



/Users/Maxime/Sites/projet8-TodoList-master/src/AppBundle / Controller / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		99.18%	121 / 122		94.12%	16 / 17		75.00%	3 / 4
DefaultController.php		100.00%	4 / 4		100.00%	1 / 1		100.00%	1 / 1
SecurityController.php		87.50%	7 / 8		66.67%	2 / 3		0.00%	0 / 1
TaskController.php		100.00%	67 / 67		100.00%	8 / 8		100.00%	1 / 1
UserController.php		100.00%	43 / 43		100.00%	5 / 5		100.00%	1 / 1

Legend

Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

Dans l'ensemble le constat est également positif avec 3 Controllers sur 4 couverts à 100%. On voit ici que c'est une méthode du SecurityController qui n'a pas été couverte et on comprend également que si une méthode n'est pas couverte alors c'est toute la Classe qui est égale 0.
Allons maintenant dans la classe SecurityController pour voir quelle méthode n'est pas couverte.

III. Résultat classe SecurityController

/Users/Maxime/Sites/projet8-TodoList-master/src/AppBundle / Controller / SecurityController.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		66.67%	2 / 3	CRAP	87.50%	7 / 8
SecurityController		0.00%	0 / 1		66.67%	2 / 3	3.02	87.50%	7 / 8
loginAction					100.00%	1 / 1	1	100.00%	6 / 6
loginCheck					100.00%	1 / 1	1	100.00%	1 / 1
logoutCheck					0.00%	0 / 1	2	0.00%	0 / 1

Ici on constate que c'est la méthode logoutCheck qui n'a pas été couverte. On pourrait penser que cette méthode a été omise lors des tests. Allons vérifier dans notre fichier de SecurityControllerTest :

SecurityControllerTest.php

```
31      }
32
33      public function testLogout()
34      {
35          $client = static::createClient();
36
37          // Request the route
38          $crawler = $client->request( method: 'GET', uri: '/login');
39
40          // Test if it is connected well
41          $this->assertSame( expected: 200, $client->getResponse()->getStatusCode());
42
43          $form = $crawler->selectButton( value: 'Se connecter')->form();
44
45          $form["_username"] = "User";
46          $form["_password"] = "hihi";
47
48          $client->submit($form);
49
50          $this->assertTrue($client->getResponse()->isRedirect());
51          $crawler = $client->followRedirect();
52
53          $link = $crawler->selectLink( value: 'Se déconnecter')->link();
54
55          $client->click($link);
56
57          //Test redirection
58          $this->assertTrue($client->getResponse()->isRedirect());
59          $crawler = $client->followRedirect();
60
61          //Test if it arrived back on login and well logout
62          $this->assertSame( expected: 1, $crawler->filter( selector: 'h1:contains("Login")')->count());
63      }
```

On constate qu'il y a un bien un test qui est effectué et qui ressort positif lors de l'exécution. Alors pourquoi le test ne passe par la fonction logout ? La réponse est en fait assez simple, le système de logout mis en place est géré plus en amont dans Symfony et ne tient compte que de l'annotation de la fonction logout. Donc il est normal que cette fonction ne soit pas couverte.

Conclusion

Bien que le taux de couverture ne soit pas de 100%, on peut tout de même estimer que le code est couvert à 100%. En effet, la seule méthode non couverte n'a pas de code à exécuter et est en fait une redirection vers un module spécifique de Symfony 3.4.