École Polytechnique de Montréal

Département de génie informatique et génie logiciel

LOG3430

Méthodes de test et de validation du logiciel

Travail pratique #1 – Tests unitaires

Automne 2018

Soumis par :

Son-Thang Pham (1856338)

Cedric Behr (1972888)

Présenté à :

Hiba Bagane

Section (01-B1)

2 Octobre 2018

Choix de tests utilisées

Nous avons utilisé des *spies et des stubs* pour bien tester les méthodes demandées.

Les spies sont utiles afin de pouvoir observer une méthode, par exemple en regardant ses arguments, valeurs de retour et exceptions retournées sans affecter cette même méthode.

Nous avons surtout utilisé les spies pour les exceptions des tests du Client. En effet, puisque nous voulons juste observer si une exception s'est produite, nous optons pour l'usage de spies puisqu'il ne modifie pas la méthode.

Les stubs sont des spies avec un comportement préprogrammé qui remplace la méthode. Nous pouvons les utiliser quand nous voulons forcer une partie de code ou bien pour le prévenir.

Nous avons utilisé des stubs pour les tests de Client et jsonClient. Nous utilisons les stubs pour empêcher les appels API.

Réponses aux questions

Quelle méthode avez -vous choisi pour empêcher la classe JsonClient d'exécuter les vrais appels API durant vos tests unitaires ?

Nous avons utilisé des stubs pour empêcher les vrais appels API. En effet, comme mentionné plus haut, nous avons le choix d'utiliser un stub pour prévenir une partie de code et c'est ce que nous faisons ici. En les utilisant, nous empêchons le programme de faire les nombreuses requêtes HTTP demandées tel qu'un POST ou un PATCH en les simulant.

Quelle méthode avez-vous choisi pour empêcher la classe Client d'exécuter les vrais appels API contenus dans les méthodes de la classe JsonClient durant vos tests unitaires?

Nous avons utilisé des stubs pour empêcher les vrais appels API. En effet, comme pour les tests pour le JsonClient, nous avons le choix d'utiliser un stub pour prévenir une partie de code et c'est ce que nous faisons ici encore une fois. En les utilisant, nous empêchons le programme de faire des requête HTTP en utilisant des résultats que nous déterminons pour ces requêtes afin de pouvoir tester la méthode principale sans prendre en compte les sous-méthodes.

Est-ce que vos tests unitaires ont découvert une ou plusieurs défaillances ? Si oui, expliquez ce que vous avez trouvé ?

Les méthodes toJson() des classes Recipient.js et Sharedbox.js des helpers semblent être problématiques. En effet, pour le test du Sharedbox.js nous obtenons les valeurs de recipients et attachments sont exacts. Pour les autres valeurs nous obtenons undefined. Nous pouvons conclure que le test a découvert un problème dans cette méthode. Pour celui de la classe Recipient, nous obtenons que des valeurs undefined.

Le projet implémente un type d'exceptions personnalisé SharedBoxException qui étend le type standard Error. Lorsque vous avez testé les exceptions qui peuvent être lancées par la classe Client, était-il possible de vérifier le type de l'exception ? Si non, expliquez pourquoi et proposez une solution.

Oui, il est possible de vérifier le type de l'exception en utilisant le *throw* de la librairie chai et de pouvoir bien les tester. D'ailleurs, il est aussi possible d'utiliser les spies pour observer les exceptions retournées.