Tests unitaires

TP1

Billy Bouchard

Jacob Dorais

Un Travail présenté à :

Hiba Bagane

Département Génie informatique et Logiciel

Polytechnique Montreal

le 2 Octobre 2018

Stub, Spy et Mock

Malgré que ces 3 éléments permettent de créer de fausses fonctions et de faux objets lors des différents test, il devient très rapidement difficile d'identifier lesquels servent dans quelles cas. En effet, ces trois objets ont différent but dans les test et il est important de connaitre leur role. Le spy permet d'observer le comportement d'une fonction, Le stub permet d'en empecher l'execution et le mock permet de modifier un objet complet

Spy

Le spy permet d'observer le comportement d'une méthode sans en modifier l'exécution. Cela peut rapidement devenir utile lorsqu'on veux connaitre les arguments de la fonction utiliser ou encore tout simplement savoir combien de fois cette dernière a été appelé. Malgré qu'il ne faut pas utiliser de spy lorsqu'on a l'intention de vouloir modifier le fonctionnement d'une méthode ou d'une fonction.

0.1 Stub

Le stub permet de faire la même chose que le spy, mais en plus il empêche l'execution de la méthode

Question

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

Nous avons choisis d'utiliser un stub de la method fetch de l'objet nodeFetch. Il s'agit d'une modification de la méthode qui remplace la définition utilisée dans le code de base. Un stub permet donc de remplacer complètement l'implementation de la méthode fetch de l'objet nodeFetch. On peux alors choisir de renvoyer les réponse que l'on veux, que se soit une promesse ou une simple valeur de retour. Dans le cas du programme actuel, on utilise le stub pour envoyer des prommesse et éviter de faire de vrai requette http. On peux alors controler les paramètre des promesses afin de s'assurer que les résultats voulues soit appeler. On peux aussi choisir de faire différent retour en fonction de la nombre de fois que la fonction est appeler. Ainsi, on peux choisir de retourner certaines données lors du premier appel et d'autre données lors du second. Le tout nous permet d'obtenir des test encore plus précis sur certaines méthodes.

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 ?