Techniques de test

Compte rendu TP2

Tests unitaires avec SimpleTest

Ricardo Rodríguez

Techniques de test
M1 ILSEN
Université d'Avignon et des Pays de Vaucluse

Introduction

Ce document présente un résumé du travail réalisé pour le TP2 du cours Techniques de tests.
On a travaillé avec l'outil SimpleTest qui propose un framework simple pour la création et exécution de tests unitaires. Nous avons travaillé les concepts de test case, test suite, mocks.

Partie A

A.1 - Un premier test avec SimpleTest

Dans une première partie j'ai créé le fichier <dossier_rendu>/testsA/testA.php qui va vérifier la fonction queFaireAujourdhui dans deux cas :

1. météo belle : true
   mer chaude : true
   présence requins : true

2. météo belle : false
   mer chaude : false
   présence requins : false

La fonction renvoie le mauvais résultat pour le test 1, elle aurait dû renvoyer "Plage:Bronze".
Le deuxième test passe sans problèmes.

Pour changer l'affichage des résultats par défaut de SimpleTest j'ai créé la classe MonRapporteurHTML qui étend les fonctionnalités d'un rapporteur basique.

A.2 - Générateur de tests

Dans un deuxième temps, j'ai créé le fichier <dossier_rendu>/testsA/generateur.php qui va recevoir comme entrée un fichier CSV représentant la table de décision associée à la fonction et qui va générer un fichier PHP avec le test de toutes les combinaisons.

Générateur pour tout type de fonction

En ce qui concerne la question sur si ce type d'automatisation (un générateur de tests) peut être utilisé dans n'importe quelle fonction, la réponse est **NON**. En fait, nous sommes en train de tester une fonction très spécifique qui reçoit des booléens et qui renvoie un string avec un format précis.
Mon générateur est déjà un peu complexe parce que j'ai essayé de gérer un fichier d'entrée qui pourrait avoir une quantité illimitée de conditions et de résultats possibles. Mais par exemple mon code est restreint à avoir un résultat du type "résultat1:résultat2".

Ce serait impossible de gérer tout type de retour pour n'importe quelle fonction. Un exemple serait le retour d'objets. Il faudrait changer beaucoup le générateur afin qu'il puisse tester le retour d'objets. Ensuite, si on veut tester le déclenchement d'exceptions il faudrait un autre type de test. Si on veut gérer les performances d'une fonction il faudrait encore d'autres types de tests. Enfin, si la fonction n'est pas

dépendante que d'une table de décision (par exemple dans le cas de fonctions aléatoires ou qui utilisent la date actuelle dans leurs calculs) ce ne serait pas possible d'écrire un générateur. Ou le générateur à écrire serait trop complexe et donc trop coûteux.

Exécution du rapporteur

Pour exécuter le générateur par console il faut faire :

cd <dossier_rendu>/testsA
php generateur.php data/table_decision.csv out/tests.php

Pour exécuter le générateur depuis un navigateur web :

URL : localhost/.../generateur.php?data=data/table_decision.csv&out=out/tests.php

Le fichier généré utilise mon rapporteur HTML personnalisé.
Le générateur va écrire dans le fichier de sortie les entêtes qui font l'inclusion des autres fichiers nécessaires, dont SimpleTest, et puis il ajoutera plusieurs fonctions de test, une pour chaque possible combinaison des valeurs d'entrée.

Erreur dans la fonction queFaireAujourdhui

La fonction a une erreur dans la ligne 6, lors de l'évaluation du paramètre $presence_requins. Le code corrigé de la fonction est :

```
function queFaireAujourdhui($meteo_belle,$mer_chaude,$presence_requins) {

      if ($meteo_belle) {
          if ($mer_chaude) {
              if ($presence_requins) {
                  return('Plage:Bronze');
              } else {
                  return('Plage:Nage');
              }
          } else {
              return('Plage:Bronze');
          }
      } else {
          return 'Lit:?';
      }

}
```

Partie B

Test d'une classe et utilisation de mocks

Dans cette deuxième partie nous allons définir plusieurs cas de test individuels et puis une suite de tests pour simplifier leur exécution. En plus, nous allons utiliser les fonctionnalités proposées par SimpleTest pour la définition de Mocks.

Le code source pour cette partie se trouve dans <dossier_rendu>/testsB/.

Implémentation de RevolverA5Coups

J'ai créé la classe RevolverA5Coups. Dans le sujet du TP je n'ai pas bien compris s'il fallait l'implémenter ou s'il fallait utiliser un mock. Il me semblais plus correct de l'implémenter et créer seulement le mock pour la classe Hasard.

J'utilise un tableau d'entiers pour représenter les balles de la cartouche. Une nouvelle cartouche met toutes les cellules à 1, la décharge met les cellules à zéro.

Pour le cas particulier de la méthode rangerSousClef, il me semblais judicieux que si on range le révolver on ne peut plus tirer. J'ai donc ajouté une variable statique booléenne qui, si elle est à True, empêche à la fonction appuyerSurDetente de retourner BANG.

Les différentes tests

J'ai créé un fichier PHP par fonction à tester. :

- testAppuyerSurDetente.php
- testChargerUneCartouche.php
- testRangerSousClef.php
- testTournerAleatoirementBarrillet.php
- testViderBarrillet.php

et puis je les ai tous inclus dans un suite de tests définie dans :

- all_tests.php

Les tests héritent d'une classe abstraite mère définie dans TP2BUnitTestCase.class.php. Cette hiérarchie nous permet de définir un comportement général pour les tests, notamment les fixitures et la création du mock HasardMock.

Les tests utilisent à nouveau le rapporteur définie pour la partie A.

Exécution des tests

Chaque test peut être exécuté indépendamment avec son URL. Par exemple :

http://localhost/.../testsB/testViderBarrillet.php

Ou sinon on peut exécuter toute la suite avec :

http://localhost/.../testsB/all_tests.php