Test Structurel Unitaire – PHP

Envol 2014

19 Nov. 2014

Université de Franche-Comté
Institut FEMTO-ST

DISC, Besançon, France

Présentation de Unitestor

Un robot qui se déplace sur une planète lointaine...

Composants

- une horloge interne
- une batterie
- un panneau solaire
- un GPS
- un capteur de sol

Fonctionnalités

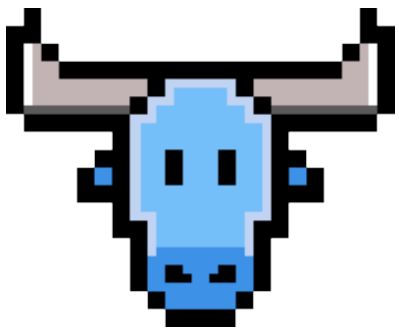
- déplacement : en fonction du sol et de la distance, plus ou moins d'énergie sera consommée
- rechargement de la batterie : le panneau solaire recharge la batterie de manière constante

Architecture

- `Source/Unitestor` : sources du projet
  - `Clock.php`
  - `Coordinates.php`
  - `LandSendor.php`
  - `Robot.php`
  - `Vector.php`
- `Source/Unitestor/tests/units` : tests unitaires manuels

Présentation atoum

- Téléchargeable à `http://atoum.org`
- Un framework xUnit PHP
- Simple : utilisation et déploiement facile
- Moderne : utilise PHP 5.3+ et compatibilité avec les outils industriels standards
- Intuitif : écriture et lecture des tests “naturelles”
- Documentation : `http://docs.atoum.org/fr`



Configuration

- Vérifier la version de atoum `php mageekguy.atoum.phar -v`
- Vérifier le chemin de atoum dans le fichier `.bootstrap.atoum.php` à la racine du projet : modifier si nécessaire la valeur de `require_once`
- Pour lancer les tests d'un fichier particulier utiliser l'option `-f` *<chemin/fichier_test>* :
  Par exemple : `php mageekguy.atoum.phar -f UniTestor/tests/units/Coordinates.php`
- Sinon pour tous les tests lancer `php mageekguy.atoum.phar`

Étape 1 - Premiers pas atoum

Écriture des tests pour : Coordinates.php

```
<?php
namespace UniTestor\tests\units;
use \atoum;

class Coordinates extends atoum{
        public function testXY( ) {
            $x = 7.0;
            $y = 42.0;
            $coordinates = new \UniTestor\Coordinates($x, $y);
            $this
                ->float($coordinates->getX())
                    ->isEqualTo($x)
                ->float($coordinates->getY())
                     ->isEqualTo($y)
             ;
    }
    }
>
```

A faire

- Réalisation du test pour vérifier le "Cast" des coordonnées X et Y.
- `Vector.php` : tester les *getters* et le calcul de la longueur d'un vecteur (avec et sans arrondi).

Étape Mock

Mes premiers pas avec les mocks !

- Les mocks permettent de modifier le comportement d'une méhode.
- `tests/units/Clock.php`, dans la fonction `testReset`, on mock l'objet `Clock` pour changer le comportement de la fonction `getCurrentTime` pour qu'elle renvoie une valeur fixe (contenue dans `nextTime`.
- Ainsi la fonction `reset` ne remet pas l'horloge à zéro mais à cette valeur.

A faire :

- `Clock.php` : tester la méthode `getDifference()`
- `Landsensor.php` : tester la méthode `getNeededEnergy()` (mock orphelin et mock de fonction)