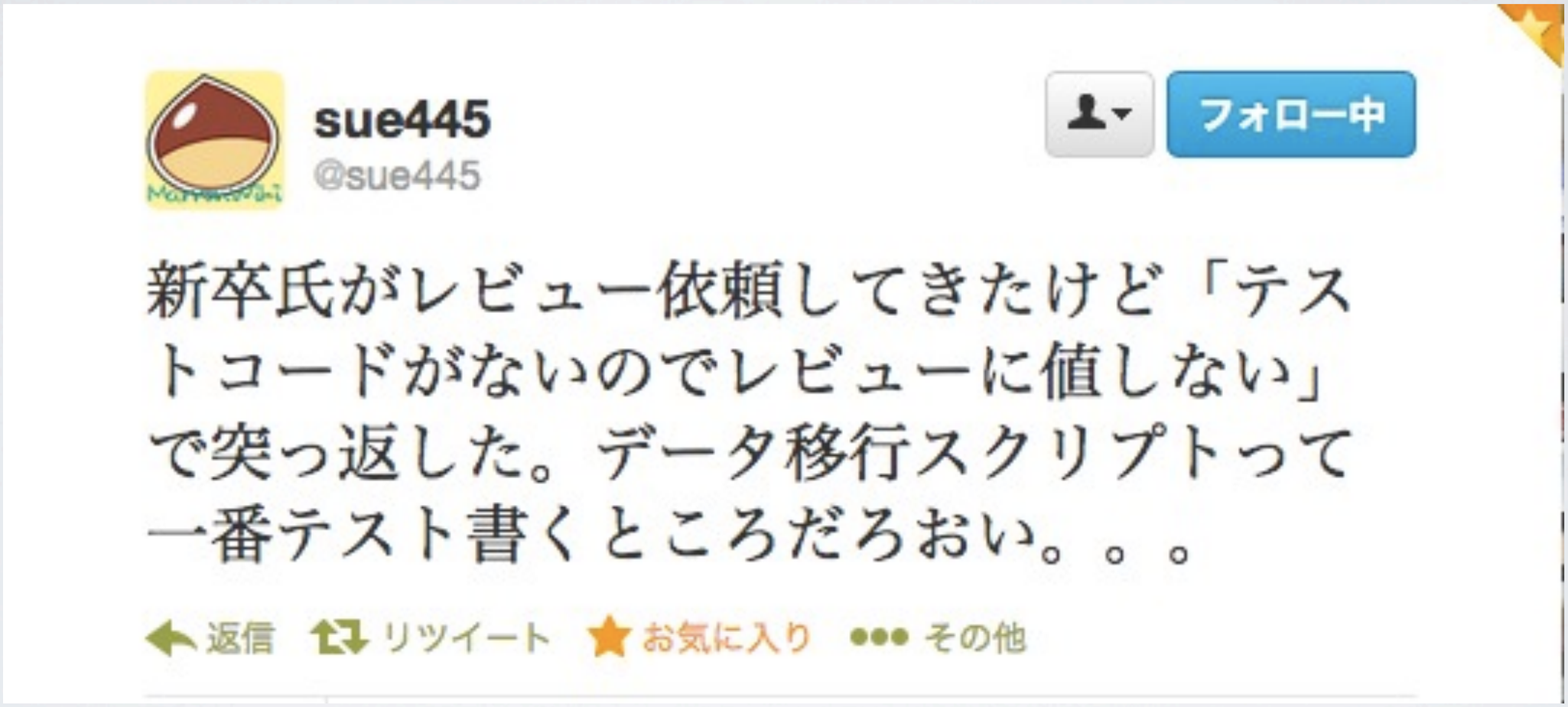
Shell Script Testing Framework
shunit2

Ruby On Railsが社内で
使用されるようになり、
UnitTestが書かれるように
なってきました

＿人人人人人人人人人人人人人人人人人人人人人人人人人人人＿
＞　　テストコードが無いのでレビューに値しない　　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣



僕の所属するチームはShell(Bash)で

色々作業する事が多いんですが

一方でスクリプトが多く

なってきたので

ちゃんとテストを書きたいと思い

shunit2を導入しました

使い方

※注意点

1.テスト関数は''test''から
はじまる名前にしてください

2.shunit2は最後にinclude
してください

shunit2/src/shunit2 をinclude

```
1 #!/usr/local/bin/bash
2
3 . ../lib/shunit2-2.1.6/src/shunit2
4
```

testではじまる関数を書く

```
1 #!/usr/local/bin/bash
2
3 testDemo() {
4   A=1
5   assertEquals $A 1
6 }
7
8 . ../lib/shunit2-2.1.6/src/shunit2
```

scriptの実行

```
onigra% ./test_demo.sh
testDemo


Ran 1 test.


OK
onigra%
-azfu-
```

テストに失敗するパターン

```
1 #!/usr/local/bin/bash
2
3 testDemo() {
4   A=1
5   assertEquals $A 2
6 }
7
8 . ../lib/shunit2-2.1.6/src/shunit2
9
```

結果

```
onigra% ./test_demo.sh
testDemo
ASSERT:expected:<1> but was:<2>

Ran 1 test.

FAILED (failures=1)
onigra%
```

解説

TestingFrameworkでは一般的に
Assertions(アサーション：表明)
という、プログラムの
動作結果が期待しているものと
確認するための機能があります

先程のデモでは

shunit2に備えられてる

assertEquals

という関数を使い、

2つの値を比較していました

assertEquals関数は

引数を2つ渡すとその2つを比較し

一致していたら成功、

そうでなかったら失敗という

レポートを出力します

```
1 #!/usr/local/bin/bash
2
3 testDemo() {
4   A=1
5   assertEquals $A 1
6 }
7
8 . ../lib/shunit2-2.1.6/src/shunit2
```

テスト通る

```
1 #!/usr/local/bin/bash
2
3 testDemo() {
4   A=1
5   assertEquals $A 2
6 }
7
8 . ../lib/shunit2-2.1.6/src/shunit2
9
```

テスト通らない

これがUnitTestの基本です

簡単ですね！

※注意点（大事な事なのでry）

1.テスト関数は”test”から
はじまる名前にしてください

2.shunit2は最後にinclude
してください

Demo2

https://github.com/YudaiSuzuki/shunit2_demo

引数を*2して返す関数のテスト

プロダクトのコード

```
1 #!/usr/local/bin/bash
2
3 # 引数を*2して返す
4 multBy2() {
5   expr $1 \* 2
6 }
```

テストのコード

```
3 # セットアップ テストするファイルをincludeする
4 oneTimeSetUp() {
5 . ../script/sample_functions.sh
6 }
7
8 # 引数が*2されることを確認する
9 testMultBy2() {
10  res=`multBy2 3`
11  assertEquals 6 ${res}
12 }
13
14 testMultBy10() {
15  res=`multBy2 10`
16  assertEquals 20 ${res}
17 }
```

oneTimeSetUp

テストの為の設定の読み込みを行う関数。
ファイルをincludeしたり、テストデータを
テスト用DBにロードしたりすることが
できます。

上記で読み込んだ物を消したい時は

oneTimeTearDown
という関数を使用します。

引数に数値以外を渡すと

エラーが出るようにしたい

```
3 # 引数の型チェック 数値以外はエラーとする
4 isInt?() {
5   A=`echo "$1" | sed -e 's/[a-zA-Z]//g'`
6
7   if [ "$1" = "$A" ]; then
8     return 0
9   else
10    return 1
11  fi
12 }
```

```
31 # 引数は数値のみを使用するため、文字列を引数とした場合エラーとする
32
33 # 正常系 引数が数値の場合
34 testCheckInt1() {
35   isInt? 9
36   assertTrue $?
37 }
38
39 testCheckInt2() {
40   isInt? 99999999
41   assertTrue $?
42 }
43
44 # 異常系 引数が数値以外の場合
45 testCheckString1() {
46  isInt? hoge
47  assertFalse $?
48 }
49
50 testCheckString2() {
51   isInt? 6gbkreo8
52   assertFalse $?
53 }
```

assertTrue / assertFalse

assertTrueはshellのconditionが0であれば
成功と見なします。
assertFalseは0以外であれば
成功とみなします。
ここでは、関数の返り値のチェックとして
使用しています。

元の関数に

型のチェック機能を入れる

```
14 # 引数を*2して返す
15 multBy2() {
16   isInt? $1
17
18   if [ $? -eq 0 ]; then
19     expr $1 \* 2
20   else
21     echo "引数エラー: 引数には数値を使用してください"
22     return 1
23   fi
24 }
```

```
19 # エラーメッセージが正しく表示されるかのテスト
20 testMultErrorMessage() {
21   res=`multBy2 hoge`
22   assertEquals "$res" "引数エラー: 引数には数値を使用してください"
23 }
24
25 # エラー発生時の終了ステータスのテスト
26 testMultErrorStatus() {
27   res=`multBy2 hoge`
28   assertFalse $?
29 }
```

shunit2は導入が簡単なので、

UnitTest書いたことない人も

割と簡単に試せると思います

おまけ

標準出力に色を付ける

shunit2のテスト結果表示は

デフォルトでは色がついてません

TDDするなら

やはり色がついてる方が

気分が乗るので…

https://github.com/YudaiSuzuki/shunit2_demo/commit/62fdb8bc13073267a1cb8c72a7df7eb8ce7bcddb

参考

http://a4p.me/archives/1157

Demo2のコードは下記にあるので
試したい人はcloneして使ってください

※masterブランチはshebangが
僕の個人環境仕様になってます。
linuxブランチは/bin/bashになってます

https://github.com/YudaiSuzuki/shunit2_demo

Enjoy Testing!!