Amusez-vous à tester avec les objets farceurs (mock)

André Espaze

31 mai 2009

Traduction du concept

Definition

mock : imite le comportement d'un objet de manière controlée

Traduction du concept

Definition

mock : imite le comportement d'un objet de manière controlée

En Anglais : mock = faux (adjectif) ou se moquer (verbe)

Traduction du concept

Definition

mock : imite le comportement d'un objet de manière controlée

En Anglais : mock = faux (adjectif) ou se moquer (verbe)
Traductions possibles :

- un objet mimétique

Traduction du concept

Definition

mock : imite le comportement d'un objet de manière controlée

En Anglais : mock = faux (adjectif) ou se moquer (verbe)
Traductions possibles :

- un objet mimétique
- un objet farceur

mocker : un module rempli de grâce

Ecrit par Gustavo Niemeyer
Même license que Python

mocker : un module rempli de grâce

Ecrit par Gustavo Niemeyer
Même license que Python

- concis : 2000 lignes de code en un module

mocker : un module rempli de grâce

Ecrit par Gustavo Niemeyer
Même license que Python

- concis : 2000 lignes de code en un module
- testé : 405 tests tournant en moins d'1 seconde

mocker : un module rempli de grâce

Ecrit par Gustavo Niemeyer
Même license que Python

- concis : 2000 lignes de code en un module
- testé : 405 tests tournant en moins d'1 seconde
- pratique : aucune dépendance

mocker : un module rempli de grâce

Ecrit par Gustavo Niemeyer
Même license que Python

- concis : 2000 lignes de code en un module
- testé : 405 tests tournant en moins d'1 seconde
- pratique : aucune dépendance

Livrable avec vos tests

Création d'un objet farceur

```
>>> import mocker as MCK

# création de l'entrepôt
>>> mocker = MCK.Mocker()

# création de l'objet farceur
>>> farceur = mocker.mock()

# enregistrement du comportement
>>> MCK.expect(farceur.salut("PT")).result("Ola")
>>> MCK.expect(farceur.salut("EN")).result("Hi")

# passage en mode action
>>> mocker.replay()
>>> farceur.salut("EN")
'Hi'
>>> farceur.salut("FR")
MCK.MatchError("Unexpected expression: farceur.salut('FR')")

# l'entrepôt vérifie le comportement de tous
>>> mocker.verify()
AssertionError("""\
=> farceur.salut('PT')
- Performed fewer times than expected.
""")
```

Vérification d'une interface

```
# Une interface
>>> class Simulation:
...     def run(self, steps_nb, tol):
...         raise NotImplementedError

# Un client périmé
>>> def launcher(simu, steps_nbs):
...     for step in steps_nbs:
...         simu.run(step)

# Un test
>>> import mocker as MCK
>>> mocker = MCK.Mocker()

>>> simu = mocker.mock(Simulation)
>>> simu.run(50)
>>> simu.run(100)
>>> mocker.replay()
>>> launcher(simu, [50, 100])
AssertionError("""
=> simu.run(50)
 - Specification is run(steps_nb, tol): 'tol' not provided
""")
```

Intégration avec unittest

```
# Un module de filtrage

def filter(results, tol):
    corrects = []
    for result in results:
        if result.less_than(tol):
            corrects.append(result)
    return corrects

# Un module de test
import mocker as MCK

class TestSelectResults(MCK.MockerTestCase):

   def test_filter_results_under_tolerance(self):
        res = [self.mocker.mock() for idx in range(3)]
        MCK.expect(res[0].less_than(MCK.ANY)).result(True)
        MCK.expect(res[1].less_than(MCK.ANY)).result(False)
        MCK.expect(res[2].less_than(MCK.ANY)).result(True)
        self.mocker.replay()

        self.assertEqual(filter(res, None), [res[0], res[2]])

import unittest; unittest.main()
```

Modification d'un objet vivant

```
# -*- coding: iso8859-15 -*-
from PyQt4 import QtGui as qt

# Un dialogue personalisé
class InputSelector(qt.QFileDialog):
    def run(self):
        self.setDirectory("/data/inputs")
        fname = None
        if (self.exec_()):
            fname = self.selectedFiles()[0]
        return fname

# Un test
import mocker as MCK
app = qt.QApplication([])
mocker = MCK.Mocker()
sel = InputSelector()
msel = mocker.patch(sel)
# Commentez la ligne suivante pour voir le dialogue
MCK.expect(msel.exec_()).result(qt.QDialog.Rejected)
mocker.replay()

assert (sel.run() is None)
mocker.verify()
mocker.restore()
```

Remplacement d'un identifiant

```
import sys

# Une fonction de la librairie
def run_simu(input, output):
    boost_run_simu(input, output)

# Traitement des arguments
def main():
    if len(sys.argv) == 3:
        run_simu(sys.argv[-2], sys.argv[-1])

# Un test
import mocker as MCK
mocker = MCK.Mocker()
run = mocker.replace(run_simu)
run("/data/input.h5", "/data/output.h5")
mocker.replay()

sys.argv = ["py", "/data/input.h5", "/data/output.h5"]
main()
mocker.verify()
mocker.restore()
```

Conclusion

Avantages :

- Test amusant du comportement

Conclusion

Avantages :

- Test amusant du comportement
- Vérification des interfaces

Conclusion

Avantages :

- Test amusant du comportement
- Vérification des interfaces
- Isolation controlée des implémentations

Conclusion

Avantages :

- Test amusant du comportement
- Vérification des interfaces
- Isolation controlée des implémentations

Limites :

- Mépriser par les non testeurs

Conclusion

Avantages :

- Test amusant du comportement
- Vérification des interfaces
- Isolation controlée des implémentations

Limites :

- Mépriser par les non testeurs
- Peut masquer un problème d'intégration

Conclusion

Avantages :

- Test amusant du comportement
- Vérification des interfaces
- Isolation controlée des implémentations

Limites :

- Mépriser par les non testeurs
- Peut masquer un problème d'intégration

Très intéressant si equilibré avec tests d'intégration et de validation