odoo testing on steroids

Leonardo Pistone

Camptocamp

About me

Leonardo Pistone

- Developer @ Camptocamp
- OCA committer & delegate member
- @lepistone

what I'd love

```
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
def compute_tax(base, rate=0):
    return 0
```

```python
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
def compute_tax(base, rate=0):
    return 0
```

```python
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
    def test_zero_rate(self):
        assert compute_tax(50, 0) == 50
```

```python
def compute_tax(base, rate=0):
    return base
```

```python
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
    def test_zero_rate(self):
        assert compute_tax(50, 0) == 50
```

```python
def compute_tax(base, rate=0):
    return base
```

```python
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
    def test_zero_rate(self):
        assert compute_tax(50, 0) == 50
```

```python
    def test_positive_rate(self):
        assert compute_tax(100, 0.05) == 105
```

```python
    def test_negative_rate(self):
        assert compute_tax(100, 0.05) == 95
```

```python
def compute_tax(base, rate=0):
    return base * (1 + rate)
```

```python
    def test_no_price_no_tax(self):
        assert compute_tax(0) == 0
```

```python
    def test_zero_rate(self):
        assert compute_tax(50, 0) == 50
```

```python
    def test_positive_rate(self):
        assert compute_tax(100, 0.05) == 105
```

```python
    def test_negative_rate(self):
        assert compute_tax(100, 0.05) == 95
```

```
# python -m unittest discover

....
----------------------
Ran 4 tests in 0.002s

OK
```

```
# python -m unittest discover

....
----------------------
Ran 4 tests in 0.002s

OK
```

```
$ python -m unittest discover

F.F.
======================================================================
FAIL: test_positive_rate_increases_amount (test_tax.TestTax)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "tdd/test_tax.py", line 13, in test_positive_rate_increases_amount
    self.assertEqual(compute_tax(100, 0.05), 105)
AssertionError: 100 != 105

----------------------------------------------------------------------
Ran 4 tests in 0.000s

FAILED (failures=2)
```

expressive

expressive

fast

expressive

fast

relevant output

expressive

fast

relevant output

maintainable

expressive

fast

relevant output

maintainable

¯_(ツ)_/¯

...really?

```
2015-03-19 13:24:51,806 23693 INFO openerp_test openerp.addons.sale_exception_nostock.tests.
2015-03-19 13:24:51,811 23693 INFO openerp_test openerp.addons.sale_exception_nostock.tests.
2015-03-19 13:24:51,812 23693 INFO openerp_test openerp.addons.sale_exception_nostock.tests.
2015-03-19 13:24:51,812 23693 INFO openerp_test openerp.addons.sale_exception_nostock.tests.
2015-03-19 13:24:52,933 23693 INFO openerp_test openerp.modules.module: module sale_owner_st
2015-03-19 13:24:53,048 23693 INFO openerp_test openerp.modules.loading: loading sale_owner_
2015-03-19 13:24:53,155 23693 INFO openerp_test openerp.modules.loading: loading sale_owner_
2015-03-19 13:24:53,227 23693 INFO openerp_test openerp.modules.module: openerp.addons.sale_
2015-03-19 13:24:53,228 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:54,618 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:55,716 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:56,809 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:58,537 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:58,537 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:58,537 23693 INFO openerp_test openerp.modules.module: openerp.addons.sale_
2015-03-19 13:24:58,538 23693 INFO openerp_test openerp.modules.module: openerp.addons.sale_
2015-03-19 13:24:58,538 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:59,572 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:59,998 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:24:59,999 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:24:59,999 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:25:00,000 23693 ERROR openerp_test openerp.addons.sale_owner_stock_sourcing.te
2015-03-19 13:25:00,000 23693 INFO openerp_test openerp.addons.sale_owner_stock_sourcing.tes
2015-03-19 13:25:00,000 23693 ERROR openerp_test openerp.modules.module: Module sale_owner_s
2015-03-19 13:25:01,305 23693 INFO openerp_test openerp.modules.module: module sale_partner_
2015-03-19 13:25:01,459 23693 INFO openerp_test openerp.modules.loading: loading sale_partne
2015-03-19 13:25:01,611 23693 INFO openerp_test openerp.modules.loading: loading sale_partne
2015-03-19 13:25:01,753 23693 INFO openerp_test openerp.modules.loading: 45 modules loaded i
2015-03-19 13:25:03,314 23693 ERROR openerp_test openerp.modules.loading: At least one test
2015-03-19 13:25:03,347 23693 INFO openerp_test openerp.modules.module: openerp.addons.base.
2015-03-19 13:25:03,348 23693 INFO openerp_test openerp.addons.base.tests.test_xmlrpc: test
```

(ノ°□°)ノ ⌒┻━┻)

(ノ°□°)ノ ⁀ ┻━┻)

Problems

Problems

output

Problems

output

slowness

Problems

output

slowness

dependencies

Problems

output

slowness

dependencies

brittleness

unittest

```
class TestItBlocks(TransactionCase):
    def test_it_can_block(self):
        self.order.order_line.budget_tot_price = 80.0
        self.order.order_line.price_unit = 100.0

        self.order.action_button_confirm()

        self.assertEqual('draft', self.order.state)
```

unittest

```
class TestItBlocks(TransactionCase):
    def test_it_can_block(self):
        self.order.order_line.budget_tot_price = 80.0
        self.order.order_line.price_unit = 100.0

        self.order.action_button_confirm()

        self.assertEqual('draft', self.order.state)
```

```
    def setUp(self):
        super(TestItBlocks, self).setUp()
        # boring stuff
```

YAML

```
-
  I create a quotation with a dropshipping line.
-
  !record {model: sale.order, id: so_4}:
    partner_id: base.res_partner_3
    order_line:
      - product_id: product.product_product_7
        product_uom_qty: 8
        route_id: route_drop_shipping
-
  I confirm the sale order, run the scheduler, and check that the address of
  the sale order has been propagated to the automatically generated purchase
  order.
-
  !python {model: sale.order, id: so_4}: |
    from nose.tools import *

    self.action_button_confirm()
    self.env['procurement.order'].run_scheduler()
    proc = self.order_line[0].procurement_ids
    assert_equal(
        proc.purchase_id.dest_address_id.id,
        ref('base.res_partner_3'),
    )
```

OERPScenario

```
Feature: Invoice workflow

  Scenario: Validation of an invoice
    Given I entered a supplier invoice for 1000 EUR
    When I validate the invoice
    Then the state of the invoice is "open"
```

OERPScenario

```
Feature: Invoice workflow

  Scenario: Validation of an invoice
    Given I entered a supplier invoice for 1000 EUR
    When I validate the invoice
    Then the state of the invoice is "open"
```

- need to write steps to implement phrases

OERPScenario

```
Feature: Invoice workflow

  Scenario: Validation of an invoice
    Given I entered a supplier invoice for 1000 EUR
    When I validate the invoice
    Then the state of the invoice is "open"
```

- need to write steps to implement phrases
- can abstract from implementation

OERPScenario

```
Feature: Invoice workflow

  Scenario: Validation of an invoice
    Given I entered a supplier invoice for 1000 EUR
    When I validate the invoice
    Then the state of the invoice is "open"
```

- need to write steps to implement phrases
- can abstract from implementation
- same test could be used for backend + browser

OERPScenario

```
Feature: Invoice workflow

  Scenario: Validation of an invoice
    Given I entered a supplier invoice for 1000 EUR
    When I validate the invoice
    Then the state of the invoice is "open"
```

- need to write steps to implement phrases
- can abstract from implementation
- same test could be used for backend + browser
- readable by non-developers

new()

```
class TestUnitCheck(TransactionCase):

    def test_over_budget(self):
        order = self.env['sale.order'].new({
            'total_budget': 80.0,
            'amount_total': 100.0,
        })
        self.assertTrue(order.over_budget())
```

new()

```
class TestUnitCheck(TransactionCase):

    def test_over_budget(self):
        order = self.env['sale.order'].new({
            'total_budget': 80.0,
            'amount_total': 100.0,
        })
        self.assertTrue(order.over_budget())
```

- required fields are not enforced

new()

```
class TestUnitCheck(TransactionCase):

    def test_over_budget(self):
        order = self.env['sale.order'].new({
            'total_budget': 80.0,
            'amount_total': 100.0,
        })
        self.assertTrue(order.over_budget())
```

- required fields are not enforced
- not stored to the database

new()

```
class TestUnitCheck(TransactionCase):

    def test_over_budget(self):
        order = self.env['sale.order'].new({
            'total_budget': 80.0,
            'amount_total': 100.0,
        })
        self.assertTrue(order.over_budget())
```

- required fields are not enforced
- not stored to the database
- otherwise pretty real

mock

```
product = Mock(
    spec_set=self.env['product.product'],
    qty_available=20,
)
```

mock

```
product = Mock(
    spec_set=self.env['product.product'],
    qty_available=20,
)
```

- fake objects

mock

```
product = Mock(
    spec_set=self.env['product.product'],
    qty_available=20,
)
```

- fake objects
- canned responses

anybox.buildout.odoo / nosetests

anybox.buildout.odoo / nosetests

```
$ bin/nosetests_odoo -d d -- -w module_dir
INFO ? anybox.recipe.openerp.runtime.session: Opening database 'd'
.........

----------------------------------------------------------------------
Ran 9 tests in 21.794s

OK
```

anybox.buildout.odoo / nosetests

```
$ bin/nosetests_odoo -d d -- -w module_dir
INFO ? anybox.recipe.openerp.runtime.session: Opening database 'd'
.........

----------------------------------------------------------------------
Ran 9 tests in 21.794s

OK
```

- no update

anybox.buildout.odoo / nosetests

```
$ bin/nosetests_odoo -d d -- -w module_dir
INFO ? anybox.recipe.openerp.runtime.session: Opening database 'd'
.........

----------------------------------------------------------------------
Ran 9 tests in 21.794s

OK
```

- no update
- no irrelevant logging

anybox.buildout.odoo / nosetests

```
$ bin/nosetests_odoo -d d -- -w module_dir
INFO ? anybox.recipe.openerp.runtime.session: Opening database 'd'
.........

----------------------------------------------------------------------
Ran 9 tests in 21.794s

OK
```

- no update
- no irrelevant logging
- rerun only failing tests

anybox.buildout.odoo / nosetests

```
$ bin/nosetests_odoo -d d -- -w module_dir
INFO ? anybox.recipe.openerp.runtime.session: Opening database 'd'
.........

----------------------------------------------------------------------
Ran 9 tests in 21.794s

OK
```

- no update
- no irrelevant logging
- rerun only failing tests
- keep your old tests

split decisions and dependencies

split decisions and dependencies

```
class Invoice:
    def compute(amount, rate):   # put decisions here
        return amount * (rate + 1)
```

split decisions and dependencies

```
class Invoice:
    def compute(amount, rate):   # put decisions here
        return amount * (rate + 1)
```

```
    def update_tax(self):        # put dependencies here
        tax = Tax.search(self.partner.tax_conditions)
        self.amount_with_tax = self.compute(
            self.amount_untaxed,
            tax.rate,
        )
```

split decisions and dependencies

```
class Invoice:
    def compute(amount, rate):   # put decisions here
        return amount * (rate + 1)
```

```
    def update_tax(self):        # put dependencies here
        tax = Tax.search(self.partner.tax_conditions)
        self.amount_with_tax = self.compute(
            self.amount_untaxed,
            tax.rate,
        )
```

functional core, imperative shell (Gary Bernhardt)

thanks!

thanks!

OCA sponsors