OpenERP Testing Tools
Unittest, YAML, OERPScenario

Alexandre Fayolle <alexandre.fayolle@camptocamp.com>

camptocamp

INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS

Manual testing

- Is boring
- Is error prone
- Is repetitive
- Is boring
- Is tedious
- Is repetitive

What is automated testing?

- Our job as software developers is to teach the computer how to do the boring, tedious, repetitive stuff
- And then watch the computer do it for us
  - Again
  - And again
  - And again

What is this talk not about?

- Exploratory testing
- QA
- Arguing about what is unit testing vs. integration testing vs. behavior testing vs. Younameit testing
- Arguing about all the bad reasons you come up with imagine for not writing tests

So what is it about?

- Testing as part of the process of writing and maintaining code
- Convince you of the importance of writing automated tests
- Show you tools that will help you write and run tests
- Help you enhance the quality of your development with tests

Your duty as a software developer

- Write maintainable code
- Write tests for that code
  - Tests help understand the intent of a piece of code
  - Tests help understand the impact of a modification
- Even better: write the test for a feature / bug **first**, and **then** the code that makes the test pass, TDD style
- Write automated tests for bugs you see
- Ensure your code fixes the tests

TDD rhythm

- Write new test : Red
- Make test pass : Green
- Refactor : Green
- Start again !

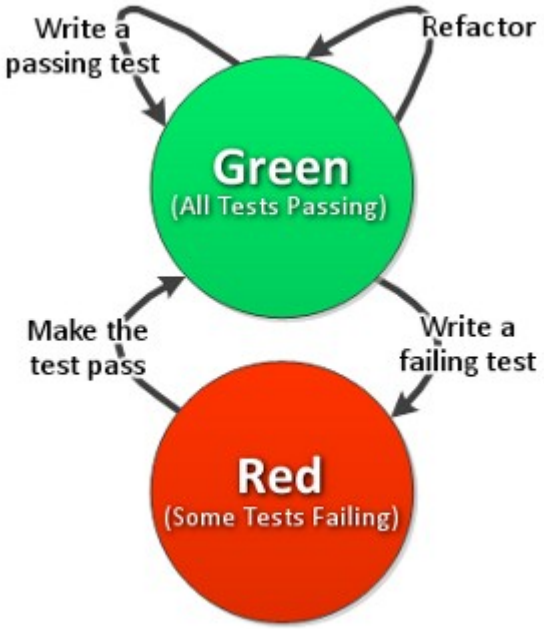

Image source:
http://www.peterprovost.org/blog/2012/05/02/kata-the-only-way-to-learn-tdd

Testing tools : unittest

- Written in Python, using the well known `unittest2` stdlib module
- Extensions to unittest2 available in **`openerp.test.common`**
  - new base test case **`class TransactionCase`**:
    - **`self.cr`**, **`self.uid`**
    - rollbacks between each tests method
  - new base test case **`class SingleTransactionTestCase`**:
    - **`self.cr`**, **`self.uid`**
    - one big transaction for all the test methods (caution there)
- Write the **`setUp`** method (don't forget to call **`super(YourClass, self).setUp()`**
- Write the different tests methods you want on the objects created in **`setUp`**
- Don't forget to add your test module in **`tests/__init__.py`**, generally in the `checks` suite
- Lots of information available on https://doc.openerp.com/trunk/server/05_test_framework/

Unittest example

lunch/tests/test_lunch.py

Testing tools : YAML tests

- Written in YAML
- YAML node shortcuts provided by OpenERP to:
  - create records (`!record`), with support for refering to other models by XML ID
  - send workflow messages (`!workflow`)
  - make testing assertions (`!assert`)
  - write raw Python in YAML blocks (`!python`)

YAML test example

sale_stock/test/picking_order_policy.yml

Testing tools : BDD (behave / OERPScenario)

- Behavior Driven Development style
- Possible to decouple the behavior of the test from the implementation
- Reusable DSL for common operations
- Interface between the end user and the developer

Behave

- Behave is a Python project bringing the Gherkin language (used by Ruby project Cucumber to express test features) to the Python world
- Features:
  - Implementation of Gherkin (`Feature`, `Scenario`, tags...)
  - Tabular data
  - Text blocks
  - Test feature discovery
  - Step definition helpers (Python decorators)
  - Test runner, with scenario selection using tags in the Feature definitions, optional syntax coloring, reports...

OERPScenario

- OERPScenario is a project by Camptocamp to help write BDD features for OpenERP,
  - Original version written in Ruby + OOoR (OpenObject on Rails) for Cucumber
  - Ported to Python using ERPPeek as the low level layer for interacting with OpenERP
  - Port got inspiration from **openobject-mirliton** project https://github.com/florentx/openobject-mirliton
- Can use run OpenERP inside the test process
  - Or connect to remote instance
- Extensible library of steps / phrases for use in tests and instance configuration
- Helpers to refer to other records by name or by xml id

BDD example : feature definition

```
@communitydays
Feature: Test the sale and stock modules
 @test
 Scenario: Test sale order partial delivery
  Given I create a sale order for "Camptocamp France"
  And containing the following sale order lines:
    | product                    | qty |
    | by name: Optical Mouse     |   5 |
    | by name: AZERTY Keyboard   |   2 |
  And I deliver the following products to the customer:
    | product                    | qty |
    | by name: Optical Mouse     |   3 |
    | by name: AZERTY Keyboard   |   2 |
  Then the sale order should not be delivered
  And there should be a backorder picking for:
    | product                    | qty |
    | by name: Optical Mouse     |   2 |
```


BDD example : step definition

```
from support.tools import puts, set_trace, model

@step('I create a sale order for "{customer}"')
def impl_so_creation(ctx, customer):
    sale_order_obj = model('sale.order')
    shop_obj = model('sale.shop')
    partner_obj = model('res.partner')
    camptocamp_id = partner_obj.search([('name', '=', customer)])[0]
    defaults = sale_order_obj.default_get(['shop_id',
                                           'date_order',
                                           'state',
                                           'user_id',
                                           'name',
                                           'invoice_quantity',
                                           ])
    shop = shop_obj.browse(defaults['shop_id'])
    values = {'partner_id': camptocamp_id,
              'partner_invoice_id': camptocamp_id,
              'partner_shipping_id': camptocamp_id,
              'pricelist_id': shop.pricelist_id.id,
              }
    values.update(defaults)
    ctx.found_item = create_new_obj(ctx, 'sale.order', values)
```


Choosing the right tool

Use **unittest** when:

- you want to test a specific method of a model
- your testing needs the full flexibility of Python
- you need to see that the test pass on http://runbot.openerp.com

Choosing the right tool

Use **YAML** tests when:

- you need to create a complex setup, with different objects
- you need to test workflows
- the size of `!python` blocks you need to write is manageable
- you need to see that the test pass on http://runbot.openerp.com

Choosing the right tool

Use **behave/OERPScenario** tests when:

- you need to validate a feature with your customer
- you want to capture customer requirements
  - possibly long before implementation is starts
- you want to write cross module tests
- you need to abstract the implementation away from the test specification

Current state of testing in OpenERP 7

- There is an existing test base in the framework and the addons
  - The runbot ensures that they pass
  - **Caution**: the presence of YAML tests files in the sources of an addon does not mean that the tests are run
  - There are some unittests (less than 20 standard addons)
  - Hard to get an idea of the coverage
- **The state of community addons is alarming!**
  - Almost no YAML tests or unittests
  - Some behave tests available
  - No support of runbot for community addons

Bug reports, patches and tests

- In an ideal world, each bug fix comes with one or more tests that the developer added to reproduce the bug, and then check that the bug was fixed
  - When you report a bug against `openobject-server` or `openobject-addons`, try to include in addition to the "steps to reproduce" an automated test, to help the support team reproduce your issue
  - Same thing for community addons
- When you submit a patch, please make sure your patch includes a test showing what you are fixing
  - This greatly eases the work of the reviewers, and the backporting to the OCB branches
- When you submit new community modules, try to provide some tests
- When you submit patches to community modules, tests are appreciated too

How MP without tests should be handled

Message:
Please add at least one automated test showing the code works as intended
Review: needs fixing

URLs

- Projects
  - http://launchpad.net/oerpscenario
  - https://pypi.python.org/pypi/unittest2
  - https://doc.openerp.com/trunk/server/05_test_framework/
  - https://pypi.python.org/pypi/behave
  - https://pypi.python.org/pypi/anybox.recipe.openerp/
- Development philosophies
  - http://en.wikipedia.org/wiki/Test_Driven_Development
  - http://en.wikipedia.org/wiki/Behavior_driven_development
- Testing
  - http://tap.szabgab.com/
  - http://en.wikipedia.org/wiki/Exploratory_testing

Questions

- If time allows...