pytest-{{cookiecutter.plugin_name}} Documentation

Release 0.0.1

{{cookiecutter.full_name}}

Apr 01, 2018

Contents

Core engine for tierra_qa package

This Pytest plugin was generated with Cookiecutter along with @hackebrot's Cookiecutter-pytest-plugin template.

CHAPTER 1

Features

`pytest-pypom-navigation` is not intended to be used as a standalone package.

It provides the core engine (pytest fixtures) needed by the strong opinionated scaffolding solution called cookiecutter-qa that let you generate a fully working QA testing hello world project based on Selenium/Splinter with just one command.

It is also used by the pytest-play engine for collecting variables for tests parametrization.

CHAPTER 2

Tests

You can run “pytest-pypom-navigation” tests via tox:

```
$ pip install tox
$ tox -epy36
```

CHAPTER 3

Contributing

Contributions are very welcome. Tests can be run with tox, please ensure the coverage at least stays the same before you submit a pull request.

CHAPTER 4

License

Distributed under the terms of the Apache Software License 2.0 license, “pytest-pypom-navigation” is free and open source software

CHAPTER 5

Issues

If you encounter any problems, please file an issue along with a detailed description.

5.1 Welcome to pytest-pypom-navigation's documentation!

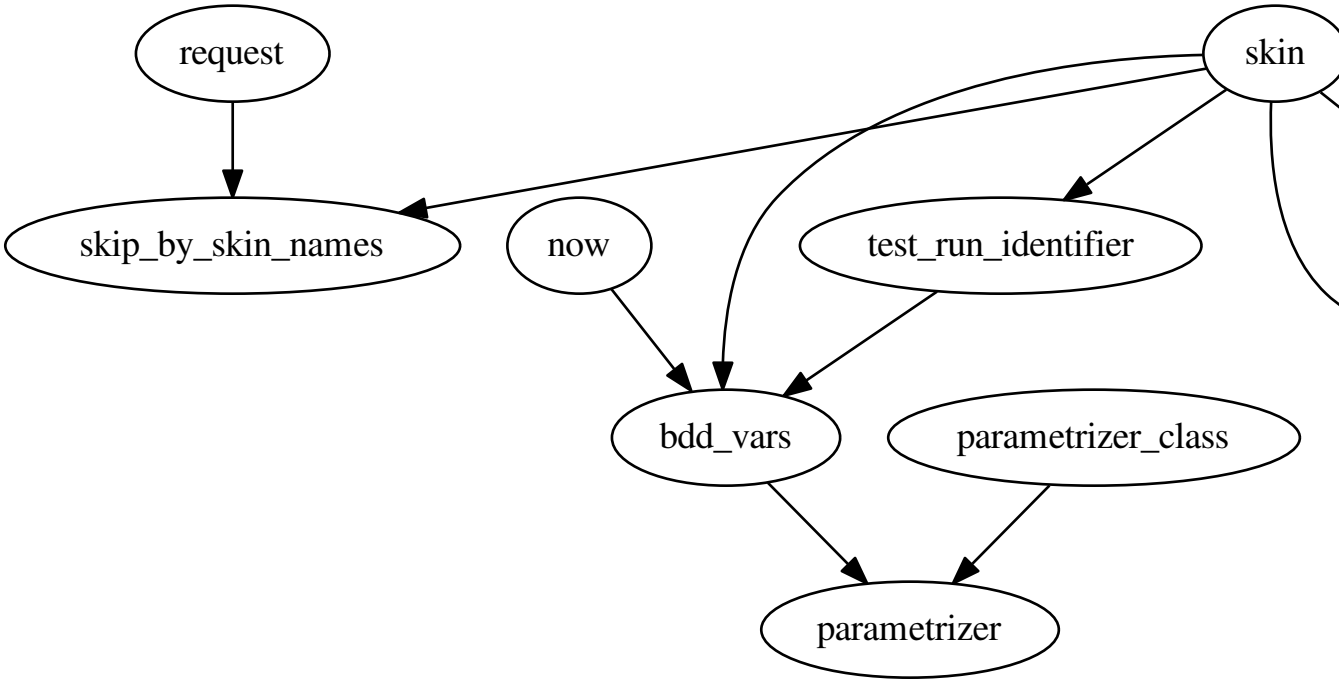
Contents:

5.1.1 API

Here you can see the technical documentation.

Fixtures

The following diagram shows the interactions between the pytest fixtures created in the `pypom_navigation` package:



pypom_navigation.plugin.**skip_skins**(*skins*)

Decorator to mark tests to be skipped for the given skin ids.

ie. @skip_skins([‘skin1’, ‘skin2’]) :return pytest.mark:

pypom_navigation.plugin.**skin**()

This fixture provides the skin associated with the application on which starts the test session.

For example:

@pytest.fixture(scope=’session’, params=mypackage.DEFAULT_PAGES.keys())

def skin(request): return request.param

pypom_navigation.plugin.**default_pages**()

A mapping with the default page object class for each skin

It’s up to you override this fixture with your settings.

For example:

```
DEFAULT_PAGES = {
    'skin1': 'mypackage.pages.BasePage',
}
```

pypom_navigation.plugin.**page_mappings**()
Returns the page mappings that describes for each page id info like the page path, the page object class to be used or any other information:

```
PAGE_MAPPINGS = {
    'HomePage': {'path': '/'},
    'LoginPage': {'path': '/'},
}
```

It's up to you override this fixture with your settings.

Returns dictionary with all known pages

Return type dict‘

pypom_navigation.plugin.**skin_base_url**(*skin*, *variables*)
Returns the skin_base_url associated to the skin.

pypom_navigation.plugin.**credentials_mapping**(*skin*, *variables*)
This fixture provides users credentials via a file specified on the –variables option. The file format is one supported by pytest-variables.

Returns credentials mapping dictionary with all available credentials

Return type dict

pypom_navigation.plugin.**default_page_class**(*skin*, *page_mappings*, *default_pages*)
Returns the default page object base class.

Returns base page object class

Return type `tierra_qa.pages.BasePage`

pypom_navigation.plugin.**default_timeout**(*variables*)
Default page timeout

pypom_navigation.plugin.**navigation**(*navigation_class*, *default_page_class*, *page_mappings*, *credentials_mapping*, *skin*, *skin_base_url*, *request*, *variables*, *default_timeout*)
Wraps a page and a page mappings accessible by pages.

`navigation.page` is meant to be mutable since through the BDD steps the page instance could change.

pypom_navigation.plugin.**navigation_class**()
Returns the navigation class used for wrap pages

pypom_navigation.plugin.**skip_by_skin_names**(*request*, *skin*)
Skip by skin name.

We support validation for multi skin applications providing the best page object class match.

We expect many failures we want to avoid because many tests will fail because the related page object implementation still not exists.

If you want you can omit a test execution for a given skin adding a a `` `@pytest.mark.skip_skins(['skin2'])` `` decorator on your tests.

Tests marked with a skin2 skip will be executed for all skins except for skin2.

See http://bit.ly/2dYnOSv for further info.

pypom_navigation.plugin.**test_run_identifier**(*skin*)
Return a session based random prefixed UUID used for identifying data created in this test run.

pypom_navigation.plugin.**now**()
Now fixture, returns current datetime object

pypom_navigation.plugin.**bdd_vars**(*test_run_identifier*, *skin*, *now*)
BDD step vars for test parametrization for dynamic values such as test_run_identifier or datetime

pypom_navigation.plugin.**parametrizer_class**()
Provides a parametrizer class used for convert parametrized json values to regular python dicts.

pypom_navigation.plugin.**parametrizer**(*parametrizer_class*, *bdd_vars*)
Parametrizer object

Navigation and base page

class pypom_navigation.pages.base.**BasePage**(*driver*, *base_url=None*, *timeout=10*, ***url_kwargs*)
Base page

current_url
Returns the current url

Returns current_url of the driver instance

Return type str

wait_for_url_change(*url*)
Wait for url change occurred.

Returns BasePage instance

Return type object

has_text(*text*)
Check for text in page.

Returns True if the given text is present

Return type bool

Utils

pypom_navigation.util.**get_page_url**(*skin_name*, *page_mappings*, *page_id*)
Returns the page_url for the given page_id and skin_name

pypom_navigation.util.**get_page_class**(*skin_name*, *page_mappings*, *page_id=None*, *fallback=None*, *default_pages=None*)
Returns the page class for a given skin name and page mapping.

First of all, if there is no page id it will return the given fallback if defined of the default page for the skin in use.

If there is a page id, it will return: * the match for the given skin if defined * a fallback if defined * the given fallback if defined or the global default page class

Parametrizer

class pypom_navigation.parametrizer.**Parametrizer**(*mapping*)
This class let you parametrize your strings and convert them to regular Python dictionaries.

It supports also json.

Let's try with a **matching name**

```
>>> value = '{"baudrate": $baudrate_value}'
>>> mapping = {"baudrate_value": 250, "name": "a name"}
>>> parametrizer = Parametrizer(mapping)
```

With the `parametrize` method you'll get a parametrized string:

```
>>> parametrizer.parametrize(value)
'{"baudrate": 250}'
```

With the `json_loads` method you'll get a parametrized regular Python mapping:

```
>>> parametrizer.json_loads(value) == {'baudrate': 250}
True
```

And now with **non matching names**

```
>>> value = '{"name": "$a_name"}'
>>> mapping = {"name": "a name"}
>>> parametrizer = Parametrizer(mapping)
```

With the `parametrize` method you'll get a parametrized string:

```
>>> parametrizer.parametrize(value)
'{"name": "$a_name"}'
```

With the `json_loads` method you'll get a parametrized regular Python mapping:

```
>>> parametrizer.json_loads(value) == {'name': '$a_name'}
True
```

And **json not valid**

```
>>> value = '{"name": $name}'
>>> mapping = {"name": "a name"}
>>> parametrizer = Parametrizer(mapping)
```

With the `parametrize` method you'll get a parametrized string:

```
>>> parametrizer.parametrize(value)
'{"name": a name}'
```

Depending on Python version 2 vs 3 you will get a different exception:

- json.decoder.JSONDecodeError: Expecting value: ...
- ValueError: No JSON object could be decoded

```
>>> import pytest
>>> with pytest.raises(Exception):
...     parametrizer.json_loads(value)
```

parametrize(*value*)
Return the value with template substitution

json_loads(*value*)
Return the json load of template substitution

5.1.2 Changelog

2.0.3 (unreleased)

- Nothing changed yet.

2.0.2 (2018-04-01)

- make credentials and base url variables optional

2.0.1 (2018-01-03)

- fix `get_page_instance` (missing page kwargs before page construction)

2.0.0 (2018-01-02)

- navigation will no more be initialized automatically with an open browser by default since pypom_navigation is used by third party plugins even for non UI plugins. This way we avoid to open a browser if it is not needed and explicitly requested with a set page or visit page
- you can override the default page timeout using a `pytest-variables` configuration named `default_timeout`
- add new method `get_page_instance` on navigation

1.0.0 (2017-12-19)

- navigation initialized with kwargs (including variables coming from pytest variables too)
- add global timeout for all pages (default 10)
- base_page fixture no longer opens page by default. It's up to you visiting a page now

0.1.1 (2017-10-30)

- support fallback page classes in action_performed

0.1.0 (2017-10-12)

- Add update_page and action_performed methods on navigation.
- Wait for pages to load when visiting them.

0.0.1 (2017-06-13)

- First release

5.2 Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index