PySpec Documentation

Release 0.0.1

Zac Stewart

May 10, 2014

Contents

PySpec is a behavior-driven development library in the spirit of RSpec. It enables you to test your code by specifying expected outcomes in prose-like language:

```
mug = Mug()
with description(Mug):
    with description('.fill')
        with context('coffee'):
            with specification('fills the mug with coffee'):
                mug.fill('coffee')
                expect(mug.contents).to(eq('coffee'))
```

CHAPTER 1

Contents

1.1 Expectations

Expectations are a way to defined expected outcomes using examples. They may be used outside of PySpec by importing `pyspec.expections`.

The basic usage pattern of PySpec Expectations is:

```
expect(actual).to(matcher(expected))
expect(actual).not_to(matcher(expected))
```

1.1.1 What are matchers?

Matchers are objects that have the following methods:

- **`match`**(*expected*, *actual*)
- **`failure_message`**()
- **`failure_message_when_negated`**()

1.1.2 Built-in matchers

PySpec Expectations comes with several matchers for testing common outcomes.

Object equivalence

Passes if `actual == expected`:

```
expect(actual).to(eq(expected))
```

Object identity

Passes if `actual is expected`:

```
expect(actual).to(be(expected))
```

Comparison

Passes if `actual > expected`:

```
expected(actual).to(be_gt(expected))
```

Passes if `actual < expected`:

```
expect(actual).to(be_lt(expected))
```

Passes if `actual >= expected`:

```
expect(actual).to(be_gte(expected))
```

Passes if `actual <= expected`:

```
expect(actual).to(be_lte(expected))
```

Passes if `(actual <= expected + delta) and (actual >= expected - delta)`:

```
expect(actual).to(be_within(delta).of(expected))
```

Passes if `re.match(pattern, actual) is not None`:

```
expect(actual).to(match(pattern))
```

Types and Classes

Passes if `isinstance(actual, expected)`:

```
expect(actual).to(be_an_instance_of(expected))
```

Passes if `type(actual) is expected`:

```
expect(actual).to(be_of_type(expected))
```

Truthiness, Falsiness, and Existentialism

Passes if `bool(actual)`:

```
expect(actual).to(be_truthy())
```

Passes if `not bool(actual)`:

```
expect(actual).to(be_falsy())
```

Lists and Collections

Passes if `expected in actual`:

```
expect(actual).to(include(expected))
```

Errors

Passes if `calling actual raises expected`:

```
expect(callable_actual[, *arguments[, **kwargs]]).to(raise_error(expected))
```

1.1.3 API

exception `pyspec.expectations.`**`ExpectationNotMetError`**(*message*)
Represents a failed expectation.

class `pyspec.expectations.`**`NegativeHandler`**(*matcher*, *actual*, **args*, ***kwargs*)
Used to resolve match of actual against a matcher and propogate a failure if it does.

`resolve`()
Raises an `ExpectationNotMetError` error with *matcher*'s `failure_message_when_negated` if *matcher* matches *actual*.

class `pyspec.expectations.`**`PositiveHandler`**(*matcher*, *actual*, **args*, ***kwargs*)
Used to resolve match of *actual* against *matcher* and propogate a failure if it does not.

`resolve`()
Raises an `ExpectationNotMetError` error with *matcher*'s `failure_message` if *matcher* does not match *actual*.

class `pyspec.expectations.`**`Target`**(*target*, **args*, ***kwargs*)
Represents a value against which expectations may be tested.

`not_to`(*matcher*)
Checks the negative case of an expectation being met.

`to`(*matcher*)
Checks the positive case of an expectation being met.

`pyspec.expectations.`**`be`**(*expected*)
Tests identify of expected and actual.

`pyspec.expectations.`**`be_an_instance_of`**(*class_*)
Tests that *actual* is an instance of *expected* or a descendant thereof

`pyspec.expectations.`**`be_falsy`**()
Tests that *actual* is falsy

`pyspec.expectations.`**`be_gt`**(*expected*)
Tests that *actual* is greater than *expected*

`pyspec.expectations.`**`be_gte`**(*expected*)
Tests that *actual* is greater than or equal to *expected*

`pyspec.expectations.`**`be_lt`**(*expected*)
Tests that *actual* is less than *expected*

`pyspec.expectations.`**`be_of_type`**(*type_*)
Tests that *actual* is specifically an instance of *expected*

`pyspec.expectations.`**`be_truthy`**()
Tests that *actual* is truthy

`pyspec.expectations.`**`be_within`**(*delta*)
Tests that *actual* is within *delta* of *expected

pyspec.expectations.**eq**(*expected*)
Tests equality of expected and actual.

pyspec.expectations.**expect**(*target*, **args*, ***kwargs*)
Returns a Target to test expectations against.

pyspec.expectations.**include**(**expecteds*)
Tests that *actual* is in *expected*

pyspec.expectations.**match**(*pattern*)
Tests that *actual* matches *pattern*

pyspec.expectations.**raise_error**(**expecteds*)
Tests that calling *actual* raises error

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

p