pageobject Documentation

Release 0.0.52

Author

Feb 06, 2017

Pageobject is an open source library built on top of selenium WebDriver intended for creating powerful and easy to maintain browser automation layer for your test automation or web scraping project in python. How ? Read on.

Note: Please note that this documentation is still a work in progress.

The documentation has two main parts:

- *User Guide*
- *API Documentation*

CHAPTER 1

Overview

1.1 What is pageobject

Pageobject library implements the so-called *Page Object design pattern* on steroids. This library/implementation will be henceforth referred to as `pageobject`.

1.2 What is Page Object design pattern

In essence, Page Objects model the web application and serve as an interface to actions that can be performed on and data that can be retrieved from it.

In its simplest, traditional form, a Page Object is just a collection of functions common to a web page to prevent repeating oneself.

While this is usually where the understanding of the Page Object design pattern ends for many implementations, it is where it just starts for `pageobject`.

1.3 Features

1.3.1 Nesting of page objects

`pageobject` sticks to the notion that a page object can be not only a web page but also any object on the page and each such page object can contain another page objects.

The beauty of this approach lies in abstraction and simply follows the way a human brain operates: When you think of an object, like a house, you will probably think of a couple of walls, windows, a door, a roof, maybe a chimney and that's it. A door knob or the color of the bathroom walls probably don't concern you at this level.

Consider a simple `login_page` with only a few components. Compare the nested model of the UI to the flat one:

```
login_page
- top_panel
|   - logo
|   |   - is_visible()
|   - search_form
|       - input
|       |   - get_value()
|       |   - set_value()
|       - submit_button
|           - is_enabled()
|           - click()
- login_form
    - username
    |   - get_value()
    |   - set_value()
    - password
    |   - get_value()
    |   - set_value()
    - submit_button
        - is_enabled()
        - click()

login_page
- is_logo_visible()
- get_search_input_value()
- set_search_input_value()
- is_submit_search_button_enabled()
- click_submit_search_button()
- get_username_value()
- set_username_value()
- get_password_value()
- set_password_value()
- is_submit_login_button_enabled()
- click_submit_login_button()
```

In the above example, both structures provide the same methods, just named differently, e.g. `search_form.submit_button.click()` vs. `click_submit_search_button()` or `login_form.submit_button.click()` vs. `click_submit_login_button()`.

Apart from being a feature in itself, the nesting has several implications which other features are based upon.

After closer inspection, it becomes immediately obvious that each page object is a separate namespace. This is crucial because not only you don't need to worry about conflicting names (`submit_button` within `search_form` is different from `submit_button` within `login_form`), but also each page object can inherit methods like `click()` from common base class and you don't need to reimplement and come up with some crazy name for it. In fact, even the `submit_button` can be a reusable component, because semantically it is and does the same thing, just in a different context (the context being location of the button webelement in the DOM of the webpage; more on that later).

1.3.2 Chained locators

The ability to chain locators is a direct consequence of nesting.

Let's narrow down the nested model to the two submit buttons for now (the string following the dash character is an xpath of the page object relative to its parent):

```
login_page - //body
- top_panel - //*[@class='topPanel']
```

```
|   - search_form - //form[@name='search']
|       - submit_button - //button
- login_form - //form[@name='login']
    - submit_button - //button
```

As you can see, both submit buttons have the same relative xpath, it's just the context - the location of the parent page object - that's different for each of them. This means that when you are locating a page object, you can safely disregard everything above its parent. That's huge.

By the way, chaining the locator to its parent is optional for each page object individually, which means your page object tree can resemble the actual DOM structure as closely as you want (as in the above example), or not resemble it at all. This comes handy when a component logically and visually fits as a child of another page object, but is located somewhere else in the DOM (like all kinds of dropdown menus, tooltips, etc.).

CHAPTER 2

Getting Started

2.1 Installation

Installation is straightforward with pip (virtualenv is recommended):

```
$ pip install pageobject
```

2.2 Basic Interactive Use

`pageobject` can be used interactively to play around and build your page object tree from scratch:

```
>>> from selenium import webdriver # we still need selenium
>>> from pageobject import Page, PageObject
>>>
>>> wd = webdriver.Chrome() # or any other browser
>>> python_org = Page(url='http://www.python.org', webdriver=wd, name='python_org')
>>> python_org.load()
<Page(SinglePageObjectBase) (full_name="python_org")>
>>> python_org.search_input = PageObject('#id-search-field')
```

CHAPTER 3

pageobject package

3.1 Subpackages

3.1.1 pageobject.commands package

Submodules

pageobject.commands.clear module

pageobject.commands.clear.**clear**(*self*, *log=True*, *press_enter=False*)
Clear the page object.

Parameters

- **log** (*bool*) – whether to log or not (defualt is True)
- **press_enter** (*bool*) – whether to press enter key after the element is cleared (defualt is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.click module

pageobject.commands.click.**click**(*self*)
Click the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.get_attribute module

pageobject.commands.get_attribute.**get_attribute**(*self*, *attribute*, *log=True*)
Return an attribute value of the page object.

Parameters

- **attribute** (*str*) – attribute name
- **log** (*bool*) – whether to log or not (default is True)

Returns attribute value

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.get_value module

pageobject.commands.get_value.**get_value**(*self*)
Return value of the page object.

Returns value of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.index module

pageobject.commands.index.**index**(*self*, *value*)
Return index of the first child containing the specified value.

Parameters **value** (*str*) – text value to look for

Returns index of the first child containing the specified value

Return type int

Raises **ValueError** – if the value is not found

pageobject.commands.is_enabled module

pageobject.commands.is_enabled.**is_enabled**(*self*, *log=True*)
Return True if page object is enabled, False otherwise.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns whether page object is enabled

Return type bool

pageobject.commands.is_existing module

pageobject.commands.is_existing.**is_existing**(*self*, *log=True*)
Return True if page object exists in the DOM, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object exists in the DOM

Return type bool

pageobject.commands.is_visible module

pageobject.commands.is_visible.**is_visible**(*self*, *log=True*)
DEPRECATED! Use is_displayed command instead.

pageobject.commands.load module

pageobject.commands.load.**load**(*self*, *log=True*)
Load the web page.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns *self*

Return type *PageObjectBase* instance

pageobject.commands.move_to module

pageobject.commands.move_to.**move_to**(*self*)
Move mouse over the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.send_keys module

pageobject.commands.send_keys.**send_keys**(*self*, *keys*, *log=True*)
Send keys to the page object.

Parameters

- **keys** (*iterable of string type*) – keys to send to the page object
- **log** (*bool*) – whether to log or not (default is True)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found

- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.set_value module

pageobject.commands.set_value.**set_value**(*self*, *value*, *press_enter=False*)
Set value of the page object.

Parameters

- **value** (*str*) – value to set to the page object
- **press_enter** (*bool*) – whether to press enter key after setting the value (default is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.text module

pageobject.commands.text.**text**
Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.wait_for_enabled module

pageobject.commands.wait_for_enabled.**wait_for_enabled**(*self*, *timeout=None*)
DEPRECATED! Use wait_until_enabled command instead.

pageobject.commands.wait_for_exist module

pageobject.commands.wait_for_exist.**wait_for_exist**(*self*, *timeout=None*)
DEPRECATED! Use wait_until_existing command instead.

pageobject.commands.wait_for_vanish module

pageobject.commands.wait_for_vanish.**wait_for_vanish**(*self*, *timeout=None*)
DEPRECATED! Use wait_until_vanished command instead.

pageobject.commands.wait_for_visible module

pageobject.commands.wait_for_visible.**wait_for_visible**(*self*, *timeout=None*)
DEPRECATED! Use wait_until_displayed command instead.

pageobject.commands.wait_until module

pageobject.commands.wait_until.**wait_until**(*self*, *func*, *func_args=[]*, *func_kwargs={}*, *timeout=None*, *error_msg=None*, *reverse=False*)
Wait until a condition is met.

Condition is an arbitrary function with optional args and kwargs that returns bool. If reverse=True, wait until the function returns False, otherwise wait until the function returns True (default).

Parameters

- **func** (*function*) – function returning bool that is repeatedly invoked until it returns correct value
- **func_args** (*list*) – list of args to be passed to func
- **func_kwargs** (*dict*) – dict of kwargs to be passed to func
- **timeout** (*int*) – number of seconds to try to call func, if not provided, PageObject.DEFAULT_WAIT_TIMEOUT is used
- **error_msg** (*str*) – error message to attach to the exception raised when the condition is not met in time
- **reverse** (*bool*) – flag indicating whether to wait until the condition is True or False

Raises **TimeoutException** – if the condition is not met in time

pageobject.commands.webelement module

pageobject.commands.webelement.**webelement**
Return a webelement instance.

Returns webelement instance

Return type selenium.webdriver.remote.webelement.WebElement

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

Module contents

pageobject.commands.**text**
Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

pageobject.commands.**text_values**
Return list of text values of PageObjectList children.

Returns index of the first child containing the specified value

Returns list of text values (innerHTML)

Return type list of str

pageobject.commands.**webelement**
Return a webelement instance.

Returns webelement instance

Return type selenium.webdriver.remote.webelement.WebElement

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

3.2 Submodules

3.3 pageobject.page module

class pageobject.page.**Page**(*url=None*, *locator=''*, *chain=True*, *webdriver=None*, *name=None*)
Bases: *pageobject.singlepageobjectbase.SinglePageObjectBase*

Web page class.

Create a page and its children page objects.

Parameters

- **url** (*str*) – Url of the page. Must start with a valid protocol (like http)
- **locator** (*str*) – Xpath describing location of the page object in the DOM.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **webdriver** (selenium.webdriver.Remote instance or None) – Only needs to be provided if the page is also a root page object.
- **name** (*str*) – Name used when the page is a root.

Example usage

```
from pageobject import Page
from selenium import webdriver
wd = webdriver.Chrome()
python_org_page = Page(url="http://www.python.org", webdriver=wd)
```

DEFAULT_ROOT_NAME = 'page'

load(*log=True*)
Load the web page.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns *self*

Return type *PageObjectBase* instance

requested_url
Return requested url, None by default.

May be overridden to take precedence over the url provided to constructor.

Returns requested url of the page

Return type str

3.4 pageobject.pageobject module

class pageobject.pageobject.**PageObject**(*locator*, *chain=True*, *webdriver=None*, *name=None*)
Bases: *pageobject.singlepageobjectbase.SinglePageObjectBase*

Main general-purpose page object class.

Create a page object and its children.

Parameters

- **locator** (*str*) – Xpath describing location of the page object in the DOM.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **webdriver** (selenium.webdriver.Remote instance or None) – Only needs to be provided for root page object.
- **name** (*str*) – Name used when the page object is a root.

Example usage

```
from pageobject import PageObject
top_panel = PageObject("//*[@class='topPanel']")
```

DEFAULT_POLL_INTERVAL = 0.25

DEFAULT_ROOT_NAME = 'page_object'

DEFAULT_WAIT_TIMEOUT = 60

NAME_SEPARATOR = '.'

children
Return dict of page object children.

Returns children of the page object

Return type dict

clear(*log=True*, *press_enter=False*)
Clear the page object.

Parameters

- **log** (*bool*) – whether to log or not (defualt is True)
- **press_enter** (*bool*) – whether to press enter key after the element is cleared (defualt is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

click()
Click the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

default_locator
Return default locator, None by default.

May be overridden to take precedence over the locator provided to constructor.

Returns default locator

Return type *None* (default) or *str* (if overridden)

full_name
Return full name of the page object instance.

If parent exists, ask for its child full name, otherwiser use normal short name.

Returns Full name of the pge object.

Return type *str*

See also:

_get_child_full_name()

get_attribute(*attribute*, *log=True*)
Return an attribute value of the page object.

Parameters

- **attribute** (*str*) – attribute name
- **log** (*bool*) – whether to log or not (default is True)

Returns attribute value

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

get_value()
Return value of the page object.

Returns value of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

init_children()
Initialize children of the page object.

Intended to be overridden by page objects containing other page objects.

Return type None

is_displayed(*log=True*)
Return True if page object is displayed, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object is displayed

Return type bool

is_enabled(*log=True*)
Return True if page object is enabled, False otherwise.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns whether page object is enabled

Return type bool

is_existing(*log=True*)
Return True if page object exists in the DOM, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object exists in the DOM

Return type bool

is_interactive(*log=True*)
Return True if page object is interactive, False otherwise.

Interactive means both displayed and enabled. This is called "clickable" in selenium, which may be misleading, as an element can be both displayed and enabled, but not really clickable, because another element may cover it and prevent it from receiving the click.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns whether page object is interactive

Return type bool

is_visible(*log=True*)
DEPRECATED! Use is_displayed command instead.

locator
Publicly exposed locator value.

Returns locator value

Return type str

logger
Return the logger object.

Returns standard logging module

Return type logging

move_to()
Move mouse over the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

name
Return name of the page object instance.

If parent exists, ask for its child name, otherwise use the name provided to constructor. If that doesn't exist either, use *DEFAULT_ROOT_NAME*.

Returns Name of the page object.

Return type *str*

parent
Return the parent of the page object.

Returns Parent page object.

Return type *pageobject.pageobjectbase.PageObjectBase* or *None* (default)

send_keys(*keys*, *log=True*)
Send keys to the page object.

Parameters

- **keys** (*iterable of string type*) – keys to send to the page object
- **log** (*bool*) – whether to log or not (default is True)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

set_value(*value*, *press_enter=False*)
Set value of the page object.

Parameters

- **value** (*str*) – value to set to the page object
- **press_enter** (*bool*) – whether to press enter key after setting the value (default is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

text

Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

tree

Returns Hierarchical tree of page object and its descendants.

Return type *dict*

wait_for_enabled(*timeout=None*)
DEPRECATED! Use wait_until_enabled command instead.

wait_for_exist(*timeout=None*)
DEPRECATED! Use wait_until_existing command instead.

wait_for_interactive(*timeout=None*)
DEPRECATED! Use wait_until_interactive command instead.

wait_for_vanish(*timeout=None*)
DEPRECATED! Use wait_until_vanished command instead.

wait_for_visible(*timeout=None*)
DEPRECATED! Use wait_until_displayed command instead.

wait_until(*func*, *func_args=[]*, *func_kwargs={}*, *timeout=None*, *error_msg=None*, *reverse=False*)
Wait until a condition is met.

Condition is an arbitrary function with optional args and kwargs that returns bool. If reverse=True, wait until the function returns False, otherwise wait until the function returns True (default).

Parameters

- **func** (*function*) – function returning `bool` that is repeatedly invoked until it returns correct value
- **func_args** (*list*) – list of args to be passed to func
- **func_kwargs** (*dict*) – dict of kwargs to be passed to func
- **timeout** (*int*) – number of seconds to try to call func, if not provided, PageObject.DEFAULT_WAIT_TIMEOUT is used
- **error_msg** (*str*) – error message to attach to the exception raised when the condition is not met in time
- **reverse** (*bool*) – flag indicating whether to wait until the condition is True or False

Raises **TimeoutException** – if the condition is not met in time

wait_until_displayed(*timeout=None*)
Wait until page object to be displayed.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_enabled(*timeout=None*)
Wait until page object is enabled.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_existing(*timeout=None*)
Wait until page object is existing in the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_interactive(*timeout=None*)
Wait until page object is interactive.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_vanished(*timeout=None*)
Wait until page object vanishes from the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

webdriver
Return the instance of WebDriver.

If parent exists, use the webdriver property of the parent. Otherwise use the value provided to constructor.

Returns reference to the webdriver instance

Return type *selenium.webdriver.Remote*

Raises **AssertionError** – if the webdriver is not a valid WebDriver

webelement
Return a webelement instance.

Returns webelement instance

Return type `selenium.webdriver.remote.webelement.WebElement`

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

3.5 pageobject.pageobjectbase module

class `pageobject.pageobjectbase.`**PageObjectBase**
Bases: `object`

Abstract page object base class.

All the other classes inherit from this one.

DEFAULT_POLL_INTERVAL = 0.25

DEFAULT_ROOT_NAME = 'root'

DEFAULT_WAIT_TIMEOUT = 60

NAME_SEPARATOR = '.'

default_locator
Return default locator, None by default.

May be overridden to take precedence over the locator provided to constructor.

Returns default locator

Return type *None* (default) or *str* (if overridden)

full_name
Return full name of the page object instance.

If parent exists, ask for its child full name, otherwiser use normal short name.

Returns Full name of the pge object.

Return type *str*

See also:

`_get_child_full_name()`

locator
Publicly exposed locator value.

Returns locator value

Return type str

logger
Return the logger object.

Returns standard logging module

Return type `logging`

name
Return name of the page object instance.

If parent exists, ask for its child name, otherwise use the name provided to constructor. If that doesn't exist either, use *DEFAULT_ROOT_NAME*.

Returns Name of the page object.

Return type *str*

parent
Return the parent of the page object.

Returns Parent page object.

Return type *pageobject.pageobjectbase.PageObjectBase* or *None* (default)

tree

Returns Hierarchical tree of page object and its descendants.

Return type *dict*

webdriver
Return the instance of WebDriver.

If parent exists, use the webdriver property of the parent. Otherwise use the value provided to constructor.

Returns reference to the webdriver instance

Return type *selenium.webdriver.Remote*

Raises **AssertionError** – if the webdriver is not a valid WebDriver

3.6 pageobject.pageobjectlist module

class pageobject.pageobjectlist.**PageObjectList**(*locator*, *chain=True*, *children_class=None*, *children_locator=None*, *count_locator=None*)
Bases: pageobject.pageobjectlistbase.PageObjectListBase

Create page object list of children of the same type.

Parameters

- **locator** (*str*) – Xpath locator of children page objects for simple indexing of children in a one-level of nesting.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **children_class** (PageObjectBase) – Class to use for instantiation of children page objects.
- **children_locator** (*str*) – Locator of children page objects offering more control over what will be indexed, necessary for more deeply nested children.
- **count_locator** (*str*) – Xpath determining the number of children, necessary for more deeply nested children.

children
Return list of children page objects.

Returns list of children page objects

Return type list of PageObjectBase instances

children_class
Return class to use for children instantiation.

Returns Class for children instantiation.

Return type PageObjectBase subclass

default_children_locator
Return defualt children locator, None by default.

May be overridden to take precedence over the children_locator provided to constructor.

Returns default children locator

Return type None (default) or str (if overridden)

default_count_locator
Return default count locator, None by default.

May be overridden to take precedence over the count_locator provided to constructor

Returns defualt count locator

Return type `None` (default) or `str` (if overridden)

3.7 pageobject.select module

class `pageobject.select.`**`Select`**(*locator*, *chain=True*, *webdriver=None*, *name=None*)

Bases: *pageobject.pageobject.PageObject*

Select page object class.

Extends PageObject class and attempts to delegate unrecognized attributes to selenium Select class.

See also:

selenium Select class documentation (external link)

Create a page object and its children.

Parameters

- **`locator`** (*str*) – Xpath describing location of the page object in the DOM.
- **`chain`** (*bool*) – Determines whether to chain locator to its parent.
- **`webdriver`** (`selenium.webdriver.Remote` instance or None) – Only needs to be provided for root page object.
- **`name`** (*str*) – Name used when the page object is a root.

Example usage

```
from pageobject import PageObject
top_panel = PageObject("//*[@class='topPanel']")
```

`elem`

Return select element to which to delegate webdriver methods.

Returns select webelement

Return type `selenium.webdriver.support.ui.Select` instance

class `pageobject.select.`**`WebDriverSelect`**(*webelement*)

Bases: `selenium.webdriver.support.select.Select`, `object`

Temporary abstract class.

This is a workaround for selenium.webdriver.support.ui.Select class not inheriting from object.

TODO: Get rid of this class when the below PR is merged: https://github.com/SeleniumHQ/selenium/pull/3067

Constructor. A check is made that the given element is, indeed, a SELECT tag. If it is not, then an UnexpectedTagNameException is thrown.

Args

- webelement - element SELECT element to wrap

Example: from selenium.webdriver.support.ui import Select

Select(driver.find_element_by_tag_name("select")).select_by_index(2)

3.8 pageobject.singlepageobjectbase module

class pageobject.singlepageobjectbase.**SinglePageObjectBase**
Bases: *pageobject.pageobjectbase.PageObjectBase*

children
Return dict of page object children.

Returns children of the page object

Return type dict

get_attribute(*attribute*, *log=True*)
Return an attribute value of the page object.

Parameters

- **attribute** (*str*) – attribute name
- **log** (*bool*) – whether to log or not (default is True)

Returns attribute value

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

init_children()
Initialize children of the page object.

Intended to be overridden by page objects containing other page objects.

Return type None

is_existing(*log=True*)
Return True if page object exists in the DOM, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object exists in the DOM

Return type bool

text
Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

wait_for_exist(*timeout=None*)
DEPRECATED! Use wait_until_existing command instead.

wait_for_vanish(*timeout=None*)
DEPRECATED! Use wait_until_vanished command instead.

wait_until(*func*, *func_args=[]*, *func_kwargs={}*, *timeout=None*, *error_msg=None*, *reverse=False*)
Wait until a condition is met.

Condition is an arbitrary function with optional args and kwargs that returns bool. If reverse=True, wait until the function returns False, otherwise wait until the function returns True (default).

Parameters

- **func** (*function*) – function returning `bool` that is repeatedly invoked until it returns correct value
- **func_args** (*list*) – list of args to be passed to func
- **func_kwargs** (*dict*) – dict of kwargs to be passed to func
- **timeout** (*int*) – number of seconds to try to call func, if not provided, PageObject.DEFAULT_WAIT_TIMEOUT is used
- **error_msg** (*str*) – error message to attach to the exception raised when the condition is not met in time
- **reverse** (*bool*) – flag indicating whether to wait until the condition is True or False

Raises **TimeoutException** – if the condition is not met in time

wait_until_existing(*timeout=None*)
Wait until page object is existing in the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_vanished(*timeout=None*)
Wait until page object vanishes from the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

webelement
Return a webelement instance.

Returns webelement instance

Return type `selenium.webdriver.remote.webelement.WebElement`

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

3.9 Module contents

pageobject's main module

class `pageobject`.**PageObject**(*locator*, *chain=True*, *webdriver=None*, *name=None*)
Bases: *pageobject.singlepageobjectbase.SinglePageObjectBase*

Main general-purpose page object class.

Create a page object and its children.

Parameters

- **locator** (*str*) – Xpath describing location of the page object in the DOM.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **webdriver** (`selenium.webdriver.Remote` instance or None) – Only needs to be provided for root page object.
- **name** (*str*) – Name used when the page object is a root.

Example usage

```
from pageobject import PageObject
top_panel = PageObject("//*[@class='topPanel']")
```

DEFAULT_POLL_INTERVAL = 0.25

DEFAULT_ROOT_NAME = 'page_object'

DEFAULT_WAIT_TIMEOUT = 60

NAME_SEPARATOR = '.'

_descendants
Return descendants of the page object.

Returns hierarchical tree of descendants of the page object

Return type `dict`

_get_child_full_name(*child_po*)
Return full name of a child page object.

Returns full name of a child page object

Return type `str`

_get_child_name(*child_po*)
Return name of a child page object.

Returns name of a child page object

Return type `str`

_locator

Returns Locator of the page object

Return type Locator instance

_locator_class

Returns locator class

Return type Locator

_locator_value

Returns processed locator value ready to be passed to a webdriver find method

Return type str

_log_id_long

Returns String identifying the page object by its full name and locator.

Return type str

_log_id_short

Returns String identifying the page object by its full name.

Return type str

_parent_locator
Return the locator of the parent page object.

Returns Locator of parent or None if parent does not exist

Return type Locator or None

_parent_locator_value

Returns value of the parent locator

Return type str

_provided_locator

Returns locator string provided either to the constructor or as an overridden default_locator property

Return type str

children
Return dict of page object children.

Returns children of the page object

Return type dict

clear(*log=True*, *press_enter=False*)
Clear the page object.

Parameters

- **log** (*bool*) – whether to log or not (defualt is True)
- **press_enter** (*bool*) – whether to press enter key after the element is cleared (defualt is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

click()
Click the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

default_locator
Return default locator, None by default.

May be overridden to take precedence over the locator provided to constructor.

Returns default locator

Return type *None* (default) or *str* (if overridden)

full_name
Return full name of the page object instance.

If parent exists, ask for its child full name, otherwiser use normal short name.

Returns Full name of the pge object.

Return type *str*

See also:

_get_child_full_name()

get_attribute(*attribute*, *log=True*)
Return an attribute value of the page object.

Parameters

- **attribute** (*str*) – attribute name
- **log** (*bool*) – whether to log or not (default is True)

Returns attribute value

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

get_value()
Return value of the page object.

Returns value of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

init_children()
Initialize children of the page object.

Intended to be overridden by page objects containing other page objects.

Return type None

is_displayed(*log=True*)
Return True if page object is displayed, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object is displayed

Return type bool

is_enabled(*log=True*)
Return True if page object is enabled, False otherwise.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns whether page object is enabled

Return type bool

is_existing(*log=True*)
Return True if page object exists in the DOM, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object exists in the DOM

Return type bool

is_interactive(*log=True*)
Return True if page object is interactive, False otherwise.

Interactive means both displayed and enabled. This is called "clickable" in selenium, which may be misleading, as an element can be both displayed and enabled, but not really clickable, because another element may cover it and prevent it from receiving the click.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns whether page object is interactive

Return type bool

is_visible(*log=True*)
DEPRECATED! Use is_displayed command instead.

locator
Publicly exposed locator value.

Returns locator value

Return type str

logger
Return the logger object.

Returns standard logging module

Return type logging

move_to()
Move mouse over the page object.

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

name
Return name of the page object instance.

If parent exists, ask for its child name, otherwise use the name provided to constructor. If that doesn't exist either, use *DEFAULT_ROOT_NAME*.

Returns Name of the page object.

Return type *str*

parent
Return the parent of the page object.

Returns Parent page object.

Return type *pageobject.pageobjectbase.PageObjectBase* or *None* (default)

send_keys(*keys*, *log=True*)
Send keys to the page object.

Parameters

- **keys** (*iterable of string type*) – keys to send to the page object
- **log** (*bool*) – whether to log or not (default is True)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

set_value(*value*, *press_enter=False*)
Set value of the page object.

Parameters

- **value** (*str*) – value to set to the page object
- **press_enter** (*bool*) – whether to press enter key after setting the value (default is False)

Returns self

Return type PageObjectBase instance

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

text
Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

tree

Returns Hierarchical tree of page object and its descendants.

Return type *dict*

wait_for_enabled(*timeout=None*)
DEPRECATED! Use wait_until_enabled command instead.

wait_for_exist(*timeout=None*)
DEPRECATED! Use wait_until_existing command instead.

wait_for_interactive(*timeout=None*)
DEPRECATED! Use wait_until_interactive command instead.

wait_for_vanish(*timeout=None*)
DEPRECATED! Use wait_until_vanished command instead.

wait_for_visible(*timeout=None*)
DEPRECATED! Use wait_until_displayed command instead.

wait_until(*func*, *func_args=[]*, *func_kwargs={}*, *timeout=None*, *error_msg=None*, *reverse=False*)
Wait until a condition is met.

Condition is an arbitrary function with optional args and kwargs that returns bool. If reverse=True, wait until the function returns False, otherwise wait until the function returns True (default).

Parameters

- **func** (*function*) – function returning `bool` that is repeatedly invoked until it returns correct value
- **func_args** (*list*) – list of args to be passed to func
- **func_kwargs** (*dict*) – dict of kwargs to be passed to func
- **timeout** (*int*) – number of seconds to try to call func, if not provided, PageObject.DEFAULT_WAIT_TIMEOUT is used
- **error_msg** (*str*) – error message to attach to the exception raised when the condition is not met in time
- **reverse** (*bool*) – flag indicating whether to wait until the condition is True or False

Raises **TimeoutException** – if the condition is not met in time

wait_until_displayed(*timeout=None*)
Wait until page object to be displayed.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_enabled(*timeout=None*)
Wait until page object is enabled.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_existing(*timeout=None*)
Wait until page object is existing in the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_interactive(*timeout=None*)
Wait until page object is interactive.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_vanished(*timeout=None*)
Wait until page object vanishes from the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

webdriver

Return the instance of WebDriver.

If parent exists, use the webdriver property of the parent. Otherwise use the value provided to constructor.

Returns reference to the webdriver instance

Return type *selenium.webdriver.Remote*

Raises **AssertionError** – if the webdriver is not a valid WebDriver

webelement

Return a webelement instance.

Returns webelement instance

Return type `selenium.webdriver.remote.webelement.WebElement`

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

class `pageobject.`**PageObjectList**(*locator*, *chain=True*, *children_class=None*, *children_locator=None*, *count_locator=None*)

Bases: `pageobject.pageobjectlistbase.PageObjectListBase`

Create page object list of children of the same type.

Parameters

- **locator** (*str*) – Xpath locator of children page objects for simple indexing of children in a one-level of nesting.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **children_class** (`PageObjectBase`) – Class to use for instantiation of children page objects.
- **children_locator** (*str*) – Locator of children page objects offering more control over what will be indexed, necessary for more deeply nested children.
- **count_locator** (*str*) – Xpath determining the number of children, necessary for more deeply nested children.

DEFAULT_POLL_INTERVAL = 0.25

DEFAULT_ROOT_NAME = 'root'

DEFAULT_WAIT_TIMEOUT = 60

NAME_SEPARATOR = '.'

_children_count

_children_locator_value

Returns processed children locator value ready to be passed to a webdriver find method

Return type str

_count_locator_value

Returns processed count locator value ready to be passed to a webdriver find method

Return type str

_descendants
Return descendants of the children_class dummy instance.

Returns hierarchical tree of descendants of the children_class dummy instance

Return type dict

_get_child_full_name(*child_po*)
Return indexed full name of a child page object.

Returns indexed full name of a child page object

Return type str

_get_child_name(*child_po*)
Return indexed name of a child page object.

Returns indexed name of a child page object

Return type str

_locator

Returns Locator of the page object

Return type Locator instance

_locator_class

Returns locator class

Return type Locator

_locator_value

Returns processed locator value ready to be passed to a webdriver find method

Return type str

_log_id_long

Returns String identifying the page object by its full name and locator.

Return type str

_log_id_short

Returns String identifying the page object by its full name.

Return type str

_parent_locator
Return the locator of the parent page object.

Returns Locator of parent or None if parent does not exist

Return type Locator or None

_parent_locator_value

Returns value of the parent locator

Return type str

_provided_children_locator

Returns children locator string provided either as an overridden default_children_locator or passed to the constructor

Return type str

_provided_count_locator

Returns count locator string provided either as an overridden default_count_locator or passed to the constructor

Return type str

_provided_locator

Returns locator string provided either to the constructor or as an overridden default_locator property

Return type str

children

Return list of children page objects.

Returns list of children page objects

Return type `list` of PageObjectBase instances

children_class

Return class to use for children instantiation.

Returns Class for children instantiation.

Return type PageObjectBase subclass

default_children_locator

Return defualt children locator, None by default.

May be overridden to take precedence over the children_locator provided to constructor.

Returns default children locator

Return type `None` (default) or `str` (if overridden)

default_count_locator

Return default count locator, None by default.

May be overridden to take precedence over the count_locator provided to constructor

Returns defualt count locator

Return type `None` (default) or `str` (if overridden)

default_locator

Return default locator, None by default.

May be overridden to take precedence over the locator provided to constructor.

Returns default locator

Return type *None* (default) or *str* (if overridden)

full_name

Return full name of the page object instance.

If parent exists, ask for its child full name, otherwiser use normal short name.

Returns Full name of the pge object.

Return type *str*

See also:

_get_child_full_name()

index(*value*)
Return index of the first child containing the specified value.

Parameters **value** (*str*) – text value to look for

Returns index of the first child containing the specified value

Return type int

Raises **ValueError** – if the value is not found

locator
Publicly exposed locator value.

Returns locator value

Return type str

logger
Return the logger object.

Returns standard logging module

Return type logging

name
Return name of the page object instance.

If parent exists, ask for its child name, otherwise use the name provided to constructor. If that doesn't exist either, use *DEFAULT_ROOT_NAME*.

Returns Name of the page object.

Return type *str*

parent
Return the parent of the page object.

Returns Parent page object.

Return type *pageobject.pageobjectbase.PageObjectBase* or *None* (default)

text_values
Return list of text values of PageObjectList children.

Returns index of the first child containing the specified value

Returns list of text values (innerHTML)

Return type list of str

tree

Returns Hierarchical tree of page object and its descendants.

Return type *dict*

webdriver
Return the instance of WebDriver.

If parent exists, use the webdriver property of the parent. Otherwise use the value provided to constructor.

Returns reference to the webdriver instance

Return type *selenium.webdriver.Remote*

Raises **AssertionError** – if the webdriver is not a valid WebDriver

class pageobject.**Page**(*url=None*, *locator=''*, *chain=True*, *webdriver=None*, *name=None*)

Bases: *pageobject.singlepageobjectbase.SinglePageObjectBase*

Web page class.

Create a page and its children page objects.

Parameters

- **url** (*str*) – Url of the page. Must start with a valid protocol (like http)
- **locator** (*str*) – Xpath describing location of the page object in the DOM.
- **chain** (*bool*) – Determines whether to chain locator to its parent.
- **webdriver** (selenium.webdriver.Remote instance or None) – Only needs to be provided if the page is also a root page object.
- **name** (*str*) – Name used when the page is a root.

Example usage

```
from pageobject import Page
from selenium import webdriver
wd = webdriver.Chrome()
python_org_page = Page(url="http://www.python.org", webdriver=wd)
```

DEFAULT_POLL_INTERVAL = 0.25

DEFAULT_ROOT_NAME = 'page'

DEFAULT_WAIT_TIMEOUT = 60

NAME_SEPARATOR = '.'

_descendants

Return descendants of the page object.

Returns hierarchical tree of descendants of the page object

Return type dict

_get_child_full_name(*child_po*)

Return full name of a child page object.

Returns full name of a child page object

Return type str

_get_child_name(*child_po*)

Return name of a child page object.

Returns name of a child page object

Return type str

_locator

Returns Locator of the page object

Return type Locator instance

_locator_class

Returns locator class

Return type Locator

_locator_value

Returns processed locator value ready to be passed to a webdriver find method

Return type str

_log_id_long

Returns String identifying the page object by its full name and locator.

Return type str

_log_id_short

Returns String identifying the page object by its full name.

Return type str

_parent_locator
Return the locator of the parent page object.

Returns Locator of parent or None if parent does not exist

Return type Locator or None

_parent_locator_value

Returns value of the parent locator

Return type str

_provided_locator

Returns locator string provided either to the constructor or as an overridden default_locator property

Return type str

_provided_url

Returns url string provided either as an overridden requested_url attribute or passed to the constructor

Return type str

children
Return dict of page object children.

Returns children of the page object

Return type `dict`

default_locator
Return default locator, None by default.

May be overridden to take precedence over the locator provided to constructor.

Returns default locator

Return type *None* (default) or *str* (if overridden)

full_name
Return full name of the page object instance.

If parent exists, ask for its child full name, otherwiser use normal short name.

Returns Full name of the pge object.

Return type *str*

See also:

_get_child_full_name()

get_attribute(*attribute*, *log=True*)
Return an attribute value of the page object.

Parameters

- **attribute** (*str*) – attribute name
- **log** (*bool*) – whether to log or not (default is True)

Returns attribute value

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

init_children()
Initialize children of the page object.

Intended to be overridden by page objects containing other page objects.

Return type None

is_existing(*log=True*)
Return True if page object exists in the DOM, False otherwise.

Parameters **log** (*bool*) – whether to log or not (default is True)

Returns whether page object exists in the DOM

Return type bool

load(*log=True*)
Load the web page.

Parameters **log** (*bool*) – whether to log or not (defualt is True)

Returns *self*

Return type *PageObjectBase* instance

locator
Publicly exposed locator value.

Returns locator value

Return type str

logger
Return the logger object.

Returns standard logging module

Return type logging

name
Return name of the page object instance.

If parent exists, ask for its child name, otherwise use the name provided to constructor. If that doesn't exist either, use *DEFAULT_ROOT_NAME*.

Returns Name of the page object.

Return type *str*

parent
Return the parent of the page object.

Returns Parent page object.

Return type *pageobject.pageobjectbase.PageObjectBase* or *None* (default)

requested_url
Return requested url, None by default.

May be overridden to take precedence over the url provided to constructor.

Returns requested url of the page

Return type str

text
Return text of the page object.

Returns text of the page object

Return type str

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

tree

Returns Hierarchical tree of page object and its descendants.

Return type *dict*

wait_for_exist (*timeout=None*)
DEPRECATED! Use wait_until_existing command instead.

wait_for_vanish (*timeout=None*)
DEPRECATED! Use wait_until_vanished command instead.

wait_until (*func*, *func_args=[]*, *func_kwargs={}*, *timeout=None*, *error_msg=None*, *reverse=False*)
Wait until a condition is met.

Condition is an arbitrary function with optional args and kwargs that returns bool. If reverse=True, wait until the function returns False, otherwise wait until the function returns True (default).

Parameters

- **func** (*function*) – function returning `bool` that is repeatedly invoked until it returns correct value
- **func_args** (*list*) – list of args to be passed to func
- **func_kwargs** (*dict*) – dict of kwargs to be passed to func
- **timeout** (*int*) – number of seconds to try to call func, if not provided, PageObject.DEFAULT_WAIT_TIMEOUT is used

- **error_msg** (*str*) – error message to attach to the exception raised when the condition is not met in time
- **reverse** (*bool*) – flag indicating whether to wait until the condition is True or False

Raises **TimeoutException** – if the condition is not met in time

wait_until_existing(*timeout=None*)
Wait until page object is existing in the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

wait_until_vanished(*timeout=None*)
Wait until page object vanishes from the DOM.

Parameters **timeout** (*int*) – number of seconds to wait, if not provided PageObject.DEFAULT_WAIT_TIMEOUT is used

webdriver
Return the instance of WebDriver.

If parent exists, use the webdriver property of the parent. Otherwise use the value provided to constructor.

Returns reference to the webdriver instance

Return type *selenium.webdriver.Remote*

Raises **AssertionError** – if the webdriver is not a valid WebDriver

webelement
Return a webelement instance.

Returns webelement instance

Return type `selenium.webdriver.remote.webelement.WebElement`

Raises

- **NoSuchElementException** – if the element cannot be found
- **InvalidSelectorException** – if the selector is invalid or doesn't select an element

See also:

selenium WebElement documentation (external link)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

p

Symbols

C

D

E

F

G

I

W