

# Page-Object Quick Reference



## Getting Started

Load the Page-Object library

```
require 'page-object'
```

Create a page-object

```
page = MyWebPage.new(@browser)
```

Create a page-object and navigate to its' url

```
page_url 'http://www.google.com'
```

```
page = MyWebPage.new(@browser, true)
```

Go to a specific URL

```
page.navigate_to 'http://www.google.com'
```

Page creation hook

```
def initialize_page
```

## Page Level Functionality

Return the html of the page or any element

```
page.html
```

Return the text of any page

```
page.text
```

Return the title of a document

```
page.title
```

Wait for a page level event -> until block returns true

```
page.wait_until(timeout, message, block)
```

Attach to another window

```
page.attach_to_window(:title => 'other title')
```

Page Navigation

```
page.forward, page.back
```

## Handling Frames and iFrames

The page-object gem provides a way to declare items are within frames or iframes.

```
class RegistrationPage
  include PageObject

  in_frame(:id => 'left-frame') do |frame|
    text_field(:address, :id => 'address_id', :frame => frame)
  end
end
```

This example shows how you might declare a text field that resides inside of a frame. You can also nest frames within frames.

```
class RegistrationPage
  include PageObject

  in_frame(:id => 'left-frame') do |frame|
    in_frame({:id => 'left-top-frame'}, frame) do |frame|
      text_field(:address, :id => 'address_id', :frame => frame)
    end
  end
end
```

Here you see a text field that is inside a frame with an id of left-top-frame which is itself inside a frame with an id of left-frame.

Notice how in both examples we had to pass the frame that is passed into the block to the elements that are inside the block. Also notice how we had to place {} around the in\_frame identifier when it is nested.

After the declaration of the frames you can use the element like any other declared on the page.

## Javascript Popups

The page-object gem handles javascript popups by intercepting the call to the popup and cause it to not happen at all.

### Alert

```
message = @page.alert do
  @page.button_that_causes_the_alert
end
message.should == "My alert text"
```

### Confirm

```
message = @page.confirm(true) do # reply with a true confirmation
  @page.button_causing_the_confirm
end
```

### Prompt

```
results = @page.prompt("Cheese") do # enter "Cheese" in the prompt
  @page.what_do_you_like_button
end
results[:message].should == "What do you like?"
```

## Ajax

The page-object gem supports waiting at a page level and at an element level for different triggers to occur.

### Page level waiting

```
def wait_until(timeout = 30, message = nil, &block)
def wait_for_ajax(timeout = 30, message = nil)
```

### Element level waiting

```
def when_present(timeout = 5)
def when_visible(timeout = 5)
def when_not_visible(timeout = 5)
def wait_until(timeout = 5, message = nil, &block)
```

## Generated Accessor Methods

Element method	HTML Element	Generated methods	How to identify an element
<b>button</b> (:name, :value => 'Save')	<button> <input type='button'> <input type='image'> <input type='reset'> <input type='submit'>	name # <i>presses button</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :alt (input type=image only)</li> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :src (input type=image only)</li> <li>• :text</li> <li>• :value</li> <li>• :xpath</li> </ul>
<b>cell</b> (:name, :id => 'cellid')	<td>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text</li> <li>• :xpath</li> </ul>
<b>checkbox</b> (:name, :id => 'cbname')	<input type='checkbox'>	check_name # <i>checks</i> uncheck_name # <i>unchecks</i> name_checked? # <i>is checked</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :value</li> <li>• :xpath</li> </ul>

Element method	HTML Element	Generated methods	How to identify an element
<b>div</b> (:name, :id => 'divname')	<div>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text</li> <li>• :xpath</li> </ul>
<b>file_field</b> (:name, :id => 'ffieldid')	<input type='file'>	name= # <i>sets value in field</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :title</li> <li>• :xpath</li> </ul>
<b>form</b> (:name, :id => 'formid')	<form>	name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :action</li> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :xpath</li> </ul>
<b>h1</b> (:name, :id => 'headingid') <b>h2</b> (:name, :id => 'headingid') <b>h3</b> (:name, :id => 'headingid') <b>h4</b> (:name, :id => 'headingid') <b>h5</b> (:name, :id => 'headingid') <b>h6</b> (:name, :id => 'headingid')	<h1> <h2> <h3> <h4> <h5> <h6>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text</li> <li>• :xpath</li> </ul>

Element method	HTML Element	Generated methods	How to identify an element
<b>hidden_field</b> (:name, :id => 'hfid')	<input type='hidden'>	name # <i>returns value</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :css</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text</li> <li>• :value</li> <li>• :xpath</li> </ul>
<b>image</b> (:name, :id => 'imgname')	<img>	name_element # element	<ul style="list-style-type: none"> <li>• :alt</li> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :src</li> <li>• :xpath</li> </ul>
<b>link</b> (:name, :text => 'linktext')	<a>	name # <i>clicks link</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :href</li> <li>• :id</li> <li>• :index</li> <li>• :link</li> <li>• :link_text</li> <li>• :name</li> <li>• :text</li> <li>• :xpath</li> </ul>

Element method	HTML Element	Generated methods	How to identify an element
<b>list_item</b> (:name, :id => 'liid')	<li>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>
<b>ordered_list</b> (:name, :id => 'olid')	<ol>	name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>
<b>paragraph</b> (:name, :id => 'paraid')	<p>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>
<b>radio_button</b> (:name, :id => 'rbid')	<input type='radio'>	select_name # <i>checks</i> clear_name # <i>clears</i> name_selected? # <i>status</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>

Element method	HTML Element	Generated methods	How to identify an element
<b>select_list</b> (:name, :id => 'slid')	<select>	name # <i>returns value</i> name= # <i>set the value</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text (Watir only)</li> <li>• :value (Watir only)</li> <li>• :xpath</li> </ul>
<b>span</b> (:name, :id => 'spanid')	<span>	name # <i>returns text</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>
<b>table</b> (:name, :id => 'table_id')	<table>	name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>
<b>text_area</b> (:name, :id => 'area_id')	<textarea>	name # <i>returns value</i> name= # <i>set the value</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :css</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>



Element method	HTML Element	Generated methods	How to identify an element
<b>text_field</b> (:name, :id => 'field_id')	<input type='text'> <input type='password'>	name # <i>returns value</i> name= # <i>set the value</i> name_element # <i>element</i>	<ul style="list-style-type: none"> <li>• :class</li> <li>• :css</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :text (Watir only)</li> <li>• :title (Watir only)</li> <li>• :value (Watir only)</li> <li>• :xpath</li> </ul>
<b>unordered_list</b> (:name, :id => 'ulid')	<ul>	name_element # element	<ul style="list-style-type: none"> <li>• :class</li> <li>• :id</li> <li>• :index</li> <li>• :name</li> <li>• :xpath</li> </ul>

## Nested Elements

It is possible to find elements nested within other elements. All elements have a complete set of \*\_element methods which look for the named element within its' own context. For example, if I want to find an unordered\_list nested within a div I can do the following:

```
div(:errors, :id => 'error_explanation')

def error_messages
  errors_element.unordered_list_element.text
end
```

This method is getting the div element and then calling the method on it to retrieve the unordered\_list.

## Page Factory

It is a good idea to keep your step definitions clean and not define instance variables for each page object. `page-object` has a module named `PageFactory` with two methods you can use to declare your intent to use a page-object:

```
def visit_page(page_class, &block)
  def on_page(page_class, &block)
```

Here is how you can use these methods in your step definitions:

```
visit_page RegistrationPage
```

This will cause the browser to open the page specified by the call to `page_url` in your page object.

```
page_url "http://mysite.com/registration"
```

If you wish to perform some activity on the page after navigating to it you can pass a block to the method.

```
visit_page RegistrationPage do |page|
  page.register_user
  page.do_something_else
end
```

If you are already on a page and wish to interact with it you can use the `on_page` method.

```
on_page CheckoutPage do |page|
  page.cancel_order
  page.do_something_else
end
```

Both methods also support a shortened version to do only one item on the page.

```
on_page(CheckoutPage).cancel_order
```