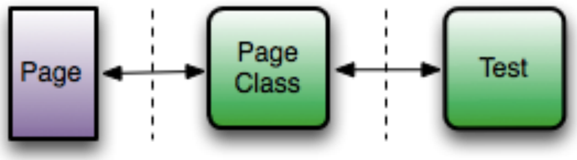
Page-Object Quick Reference



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
button(:name, :value => 'Save')	<button> <input type='button'> <input type='image'> <input type='reset'> <input type='submit'>	name # *presses button* name_element # *element*	• :alt (input type=image only) • :class • :id • :index • :name • :src (input type=image only) • :text • :value • :xpath
cell(:name, :id => 'cellid')	<td>	name # *returns text* name_element # *element*	• :class • :id • :index • :name • :text • :xpath
checkbox(:name, :id => 'cbname')	<input type='checkbox'>	check_name # *checks* uncheck_name # *unchecks* name_checked? # *is checked* name_element # *element*	• class • :id • :index • :name • :value • :xpath

Element method	HTML Element	Generated methods	How to identify an element
div(:name, :id => ‘divname’)	<div>	name # *returns text* name_element # *element*	• :class • :id • :index • :name • :text • :xpath
file_field(:name, :id => ‘ffieldid’)	<input type=‘file’>	name= # *sets value in field* name_element # *element*	• :class • :id • :index • :name • :title • :xpath
form(:name, :id => ‘formid’)	<form>	name_element # *element*	• :action • :class • :id • :index • :xpath
h1(:name, :id => ‘headingid’) **h2**(:name, :id => ‘headingid’) **h3**(:name, :id => ‘headingid’) **h4**(:name, :id => ‘headingid’) **h5**(:name, :id => ‘headingid’) **h6**(:name, :id => ‘headingid’)	<h1> <h2> <h3> <h4> <h5> <h6>	name # returns text name_element # element	• class • :id • :index • :name • :text • :xpath

Element method	HTML Element	Generated methods	How to identify an element
hidden_field(:name, :id => 'hfid')	<input type='hidden'>	name # *returns value* name_element # *element*	• :class • :css • :id • :index • :name • :text • :value • :xpath
image(:name, :id => 'imgname')	<img>	name_element # element	• :alt • :class • :id • :index • :name • :src • :xpath
link(:name, :text => 'linktext')	<a>	name # *clicks link* name_element # *element*	• :class • :href • :id • :index • :link • :link_text • :name • :text • :xpath

Element method	HTML Element	Generated methods	How to identify an element
list_item(:name, :id => ‘liid’)	<li>	name # *returns text* name_element # element	• class • :id • :index • :name • :xpath
ordered_list(:name, :id => ‘olid’)	<ol>	name_element # element	• :class • :id • :index • :name • :xpath
paragraph(:name, :id => ‘paraid’)	<p>	name # returns text name_element # element	• :class • :id • :index • :name • :xpath
radio_button(:name, :id => ‘rbid’)	<input type=‘radio’>	select_name # *checks* clear_name # *clears* name_selected? # *status* name_element # *element*	• :class • :id • :index • :name • :xpath

Element method	HTML Element	Generated methods	How to identify an element
select_list(:name, :id => ‘slid’)	<select>	name # *returns value* name= # *set the value* name_element # *element*	• class • :id • :index • :name • :text (Watir only) • :value (Watir only) • :xpath
span(:name, :id => ‘spanid’)	<span>	name # *returns text* name_element # *element*	• :class • :id • :index • :name • :xpath
table(:name, :id => ‘table_id’)	<table>	name_element # element	• :class • :id • :index • :name • :xpath
text_area(:name, :id => ‘area_id’)	<textarea>	name # *returns value* name= # *set the value* name_element # *element*	• :class • :css • :id • :index • :name • :xpath

Element method	HTML Element	Generated methods	How to identify an element
text_field(:name, :id => ‘field_id’)	<input type=‘text’> <input type=‘password’>	name # *returns value* name= # *set the value* name_element # *element*	• :class • :css • :id • :index • :name • :text (Watir only) • :title (Watir only) • :value (Watir only) • :xpath
unordered_list(:name, :id => ‘ulid’)	<ul>	name_element # element	• :class • :id • :index • :name • :xpath

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 you 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
```