# Time to learn more automation

1. **Finding an element**

We have seen how to get Xpath of an element. Now we would use this knowledge to get reference of an element at [www.questionselenium.com](http://www.questionselenium.com)

**Problem:** Get reference to search element on [www.questionselenium.com](http://www.questionselenium.com)

**Program**

require 'watir-webdriver'

*#open a firefox browser*

oPage = Watir::Browser.new :firefox

*#navigate to questionselenium.com*

oPage.goto "www.questionselenium.com"

*#get the object reference of search box*

o\_searchBox = oPage.element(:xpath => "//\*[@id='searchbox']/input[1]")

*#check what kind of element does osearchBox hold*

puts o\_searchBox.class

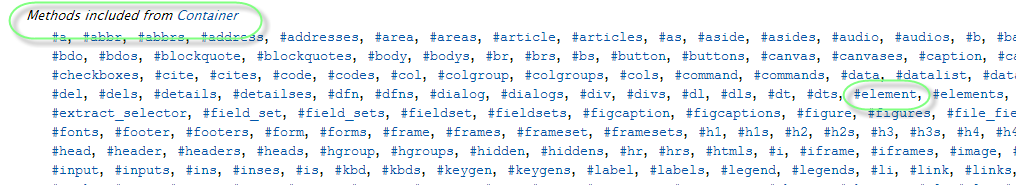
**Output**:

Watir::HTMLElement

**Explanation**:

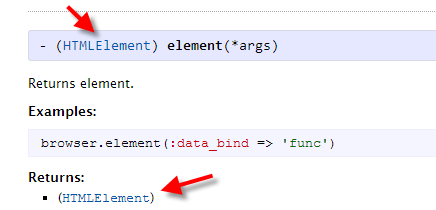
Let us have a look at [Browser API](http://rdoc.info/gems/watir-webdriver/Watir/Browser) because we have called **element** method on top of Browser object.

Note this is not a direct method from Browser API. Direct methods are under Instance Method Summary.

You can identify element method only under **Methods included from Container** section. 

Container is included into Browser class. Refer this [stackoverflow](http://stackoverflow.com/questions/10254689/multiple-inheritance-in-ruby) question to understand multiple inheritance concept. This is how the watir-webdriver developers might included the container future into Browser class.

Let us have a look at element method [API](http://rdoc.info/gems/watir-webdriver/Watir/Container#element-instance_method). Look at the return value of the method.



This means, when you say oPage.element(:xpath => “abc”) you will get in return, an object of type HTMLElement. You need to look at HTMLElement [API](http://rdoc.info/gems/watir-webdriver/Watir/HTMLElement) to understand what all operations are possible with the reference we just managed to get using method: element()

1. **Confirming if an element is really present on the loaded page**

Try this code below

require 'watir-webdriver'

*#open a firefox browser*

oPage = Watir::Browser.new :firefox

*#navigate to questionselenium.com*

oPage.goto "www.questionselenium.com"

*#get the object reference of search box*

o\_searchBox = oPage.element(:xpath => "//xpath\_that\_does\_not\_exist")

*#check what kind of element does osearchBox hold*

puts o\_searchBox.class

Note: We have passed in xpath as “//xpath\_that\_does\_not\_exist”. Isn’t it funny that there we no errors and output doesn’t differ from previous one.

The problem is element method would return an HTMLElement object despite whatever object you pass to it. So when does this trigger problem? Try below code

require 'watir-webdriver'

*#open a firefox browser*

oPage = Watir::Browser.new :firefox

*#navigate to questionselenium.com*

oPage.goto "www.questionselenium.com"

*#get the object reference of search box*

o\_searchBox = oPage.element(:xpath => "//xpath\_that\_does\_not\_exist")

*#check what kind of element does osearchBox hold*

puts o\_searchBox.class

o\_searchBox.click

**Output:**

Watir::HTMLElement

c:/Ruby193/lib/ruby/gems/1.9.1/gems/watir-webdriver-0.6.4/lib/watir-webdriver/elements/element.rb:490:in `assert\_exists': **unable to locate element, using {:xpath=>"//xpath\_that\_does\_not\_exist"} (Watir::Exception::UnknownObjectException)**

from c:/Ruby193/lib/ruby/gems/1.9.1/gems/watir-webdriver-0.6.4/lib/watir-webdriver/elements/element.rb:118:in `click'

from C:/Day2/TrainingBasicTasks/GetAnElement.rb:11:in `<main>'

**Only when you perform some operation with the returned object of element() method, you could see error like above.**

So this lies on top of an automation engineer to confirm if the reference we received is actually a reference of element we are looking for or a fake one.

Below code explains how you can do this

require 'watir-webdriver'

*#open a firefox browser*

oPage = Watir::Browser.new :firefox

*#navigate to questionselenium.com*

oPage.goto "www.questionselenium.com"

*#get the object reference of search box*

o\_searchBox = oPage.element(:xpath => "//xpath\_that\_does\_not\_exist")

*#check what kind of element does osearchBox hold*

puts o\_searchBox.class

if **o\_searchBox.exists**?

o\_searchBox.click

else

puts "element not found"

end

You can call **exists?** method to check if element exists in real and later perform automation accordingly. In case of absence of element, you should be printing the error to log/console.

Note that **exists?** alone wouldn’t be sufficient to make sure element is present inside a web application.

Think of it, an element can exist on the webpage but it is not visible. There are many elements on page that are not visible. When you call a click operation on such hidden elements, you would run into error like below.

Selenium::WebDriver::Error::UnknownError: Element is not clickable at point (-99999800, 242.5)

To handle the above case, we have another method - **visible?**

Like **exists?** It would also return a Boolean value. You can write code to perform automation only if element is visible.

Instead of going for an **if** **statement** for checking existence of an element and then going for another **if statement** to check for visibility of element, we can go for one single method **present?** that would cover for both existence and visibility.

**present? = exists? + visible?**

Discuss example invisible.html

1. **Checking for text inside a web element**

Most of the elements with text would be similar to below html code.

<div> Howdy! </div>

You might find the tag and its attributes to be different for element you are looking for but the text lies between opening and close tag.

In such cases all we need to do is follow 3 steps.

* 1. Get reference to a valid element
  2. Call .text() on top of valid element collected in previous step
  3. Validate it against expected text

**Problem:** validate if Selenium IDE tab is having text as “Selenium IDE” in it.

require 'watir-webdriver'

*#open a firefox browser*

oPage = Watir::Browser.new :firefox

*#navigate to questionselenium.com*

oPage.goto "www.questionselenium.com"

*#get the object reference of IDE tab*

o\_searchBox = oPage.element(:xpath => "//\*[@id='PageList1']/div/ul/li[2]/a")

*#get the text inside Selenium IDE tab*

ideTabText = o\_searchBox.text

#compare if the element have valid text

if ideTabText != "Selenium IDE"

puts "Expected text not found. Found: #{ideTabText}"

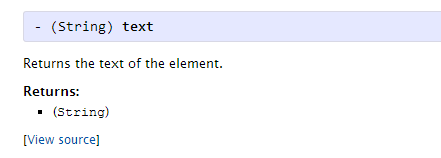
else

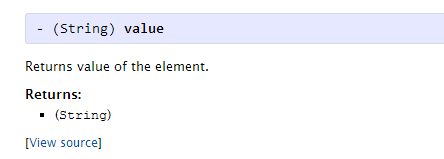
puts "Expected text found"

end

**Output:** Expected text found

Sometimes, text you see on an element is not always like the sample html with text between open and close tags as shown above. In such cases you need to try out .value





1. **Clicking on an element**

Use .click() method to perform a click operation on an element.

1. **Sending keyboard keys to a web element**

Use .send\_keys() method of HTMLElement class to send keyboard keys to a input field. This could be a string or a special key or a combination of both. Refer below link to see how you can pass such values using send\_keys method.

<http://watirwebdriver.com/sending-special-keys/>

1. **Selecting an option from a drop down**

We can do it two ways.

* First we select a list from the page and then we select an option

oPage.select\_list(:xpath => "//select").select("11112")

or

oPage.select(:xpath => "//select").select("11112")

* First we find an element (dropdown) and send keys to it.

oPage.element(:xpath => "//select").send\_keys("11112")

1. **Validating if a check box is checked**

[Checkbox API](http://rdoc.info/gems/watir-webdriver/Watir/CheckBox)

To check if a check box is checked:

oPage.checkbox(:xpath => "//input[@name='vehicle'][@value='Bike']").set?

To set a checkbox

oPage.checkbox(:xpath => "//input[@name='vehicle'][@value='Bike']").set

To unset or uncheck a checkbox

oPage.checkbox(:xpath => "//input[@name='vehicle'][@value='Bike']").set(false)

or

oPage.checkbox(:xpath => "//input[@name='vehicle'][@value='Bike']").clear

Another way, independent to Watir API

Whenever a checkbox is set, by default internally an attribute is added to the element called “checked=true”. One can check for the value of this attribute like below. Note that if a checkbox is not checked, it will return null.

oPage.element(:xpath => "//input[@name='vehicle'][@value='Bike']").attribute\_value("checked")