assert_select Cheat Sheet

Selectors

`*`	any element
`E`	an element of type E
`E.warning`	an E element whose class is "warning" (the document language specifies how class is determined).
`E#myid`	an E element with ID equal to "myid".
`E[foo]`	an E element with a "foo" attribute
`E[foo="bar"]`	an E element whose "foo" attribute value is exactly equal to "bar"
`E[foo~="bar"]`	an E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "bar"
`E[foo^="bar"]`	an E element whose "foo" attribute value begins exactly with the string "bar"
`E[foo$="bar"]`	an E element whose "foo" attribute value ends exactly with the string "bar"
`E[foo*="bar"]`	an E element whose "foo" attribute value contains the substring "bar"
`E[hreflang|="en"]`	an E element whose "hreflang" attribute has a hyphen-separated list of values beginning (from the left) with "en"
`E:root`	an E element, root of the document
`E:nth-child(n)`	an E element, the n-th child of its parent
`E:nth-last-child(n)`	an E element, the n-th child of its parent, counting from the last one
`E:nth-of-type(n)`	an E element, the n-th sibling of its type
`E:nth-last-of-type(n)`	an E element, the n-th sibling of its type, counting from the last one
`E:first-child`	an E element, first child of its parent
`E:last-child`	an E element, last child of its parent
`E:first-of-type`	an E element, first sibling of its type
`E:last-of-type`	an E element, last sibling of its type
`E:only-child`	an E element, only child of its parent
`E:only-of-type`	an E element, only sibling of its type
`E:empty`	an E element that has no children (including text nodes)
`E:not(s)`	an E element that does not match simple selector s
`E F`	an F element descendant of an E element
`E > F`	an F element child of an E element
`E + F`	an F element immediately preceded by an E element
`E ~ F`	an F element preceded by an E element

Substitution Values

`.?`	Class name
`#?`	ID attribute
`[foo=?]`	Attribute value

May be string or regular expression, e.g. `"[foo=?]"`, `/bar/i`.

assert_select Cheat Sheet

Methods

assert_select(selector, *values, equality?, message?) { |elems| ... }
assert_select(element, selector, *values, equality?, message?) { |elems| ... }

Use `selector` to select elements from response page or first argument (`element`), and evalute `equality` test. Raises exception with `message` if equality tests fail.

Equality tests include:

`true`	At least one element found (`:minimum=>1`)
`false`	No element found (`:count=>0`)
`text, :text=>text`	All elements found have the text contents (string or regexp)
`n, :count=>n`	Exactly n elements found
`:minimum=>n`	At least n elements found
`:maximum=>n`	At most n elements found
`n..m`	Between n and m elements found

If no count specified, default is `:minimum=>1`.

With block, calls block with all selected elements. Calling `assert_select` (or any of the other functions) within that block operates on element selected by outer block.

assert_select_rjs(id?) { |elems| ... }
assert_select_rjs(statement, id?) { |elems| ... }
assert_select_rjs(:insert, position, id?) { |elems| ... }

Asserts that RJS statement updates/inserts HTML content and allows nested assertions on the content.

With `id`, selects only RJS statement affecting elements with that id. With `statement`, RJS statements that `:replace`, `:replace_html` or `:insert`. With `:insert` can limit position (`:before`, `:after`, etc).

assert_select_email() { |elems| ... }

Assertions on the (HTML) body of the delivered e-mail.

assert_select_encoded(element?) { |elems| ... }

For operating on encoded HTML (e.g. RSS item description).

css_select(selector, *values) => array
css_select(element, selector, *values) => array

Returns an array with selected elements (empty if no elements selected).

Example

```
assert_select "html:root>head>title", "Login"
assert_select "form[action=?]", url_for(:action=>"login") do
  assert_select "input[type=text][name=username]"
  assert_select "input[type=password][name=password]"
end
```