scrAPI 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

scrAPI cheat sheet

Scraping

`Scraper.scrape(source, options?) => result`

Scrapes source and returns the result. Source is one of:

`String`	HTML content: parse and scrape.
`URI`	Page URL: read, parse and scrape.
`HTML::Node`	Element or document, for use with structures.

Options for reading: `:last_modified`, `:etag`, `:redirect_limit`, `:user_agent`, `:timeout`.

Options for parsing: `:root_element`, `:parser`, `:parser_options` (pass to Tidy).

Scraper Definition

`process(selector, *values, extractor?) { |element| .... }`
`process(symbol?, selector, *values, extractor?) { |element| ... }`

Use `selector` to select elements, and `extractor` to extract and store values. Pass each selected element to the block. Block can be used instead of or in combination with extractors.

Extract value from (see more options in API):

`:element`	Element itself
`:text`	Text value of element
`Class`	Another scraper
`"elem"`	Element if name matches
`"@attr"`	Attribute if specified
`"elem@attr"`	Attribute if specified on element
`[value, ...]`	First value that matches

Extract value to (more options in API):

`symbol`	Instance variable (e.g. :title to @title)
`symbol[]`	Array instance variable (e.g. "links[]")
`:skip`	If `true`, do not further process this element

Attribute accessors defined for each symbol used.

If first argument is `symbol`, uses that name to replace any other processing rule with same name.

`process_first(selector, *values, extractor?) { |element| .... }`
`process_first(symbol?, selector, *values, extractor?) { |element| ... }`

Similar to `process` but only processes the first selected element (if more than one).

scrAPI cheat sheet

`result(*symbols)`
Specifies which instance variables to return. One symbol: return value of that variable. Multiple symbols: return object with suitable accessors.

`array(*symbols)`
Specifies which instance variables are arrays. Otherwise, stores only last extracted value.

`selector(symbol, selector, *values)`
`selector(symbol?, selector, *values) { |elements| ... }`
Defines selector method with the given name (`symbol`) that takes an element as argument and returns array of selected elements, empty if no elements selected. Defines `first_` method that returns the first selected element or `nil`.

Instance Methods

`extracted`	True if any processing rule returned true (also `extracted = true/false`).
`skip(element?)`	Do not further process element. See also `:skip=>true`.
`stop()`	Stop processing.
`prepare(document)`	Called on document before any processing.
`collect()`	Called after processing is done.
`result()`	Called to return result. By default returns `self` if `extracted`. You can override method, or redefine with class method `result`.

Example

```
require "rubygems"
require "scrapi"

ebay_auction = Scraper.define do
  process "h3.ens>a",             :description=>:text, :url=>"@href"
  process "td.ebcPr>span",        :price=>:text
  process "div.ebPicture>a>img",  :image=>"@src"
  result :description, :url, :price, :image
end

ebay = Scraper.define do
  array :auctions
  process "table.ebItemlist tr.single", :auctions=>ebay_auction
  result :auctions
end

puts ebay.scrape(URI.parse("http://search.ebay.com/..."))
```