How to scrape dynamic web apps using Ruby

Piotr Jaworski

Kraków- KRUG, 21.01.2020

Senior Ruby Developer @ Bionic Business, London, UK

www: https://piotrjaworski.pl
github: https://github.com/piotrjaworski
linkedin: https://www.linkedin.com/in/jaworskipiotr/

Is anyone familiar with web scraping or web crawlers?

What will I cover in this talk?

This talk will cover some basic examples - how can we scrape web apps using different libraries written in Ruby.
Also, how can we categorize web apps and what should we remember when we design web scrapers.

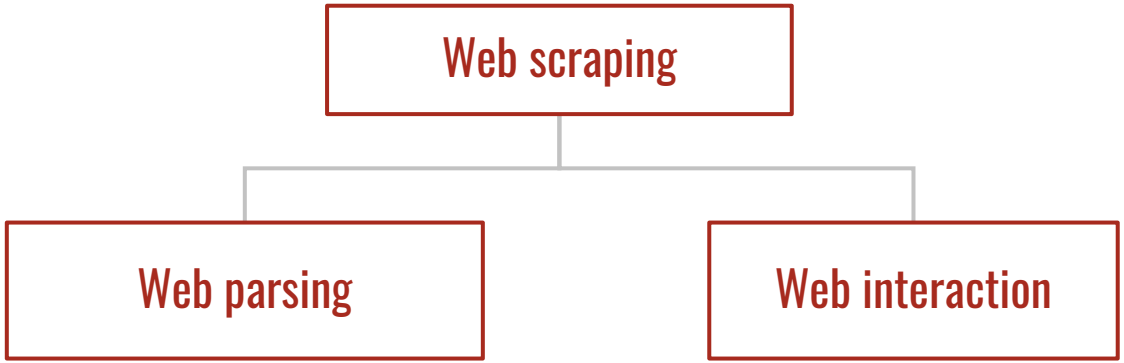
What is web scraping?

What is web scraping?

Web scraping (also known as web harvesting) is a data scraping used for extracting data and context from web pages.
It can be done manually by a user or automatically by a software (called a bot or a web crawler).
The whole process is based on parsing HTML code and interacting with HTML elements.

How can we divide web scraping?

How can we divide web scraping?



What is web parsing?

Web parsing is a part of web scraping. It allows us to process and read the HTML code of a web page.
Thanks to that, we can search for elements on a web page and extract from them a content, text.
Web parsing doesn’t mean interaction with webpages.

What is web interaction?

Web interaction is an interaction with elements on a web page, like clicking on links, filling forms, etc.

To interact with web pages, we need an instance of a browser (it could also be headless).

When is web-scraping useful?

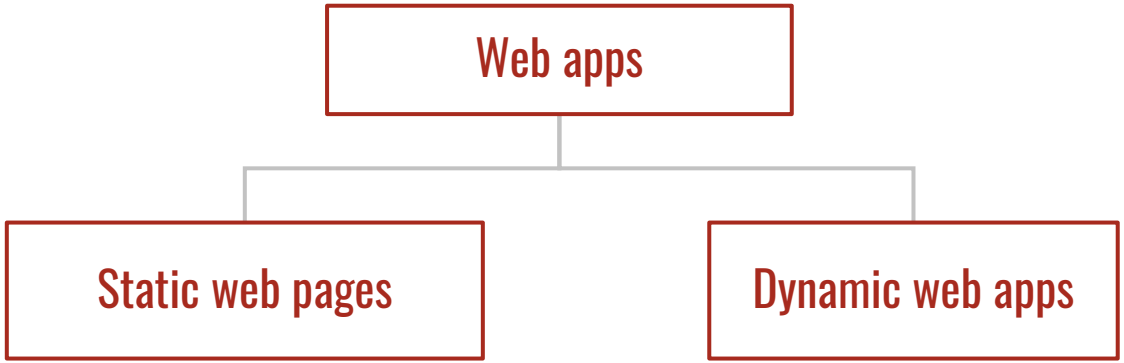
When is web-scraping is useful?

In most cases we use web scrapers when:

- any kind of data is not accessible via API,
- we want to automate any kind of process,
- we want to write a bot (remember to change IP address, use a proxy),
- we want to check if anything has been changed on a page,
- we want to test a front-end part of our web app.

How can we categorize web apps?

Web apps categories



Static web pages

What is a static web page (in web scraping)?

It’s a page where the whole content is loaded once, there are no elements which are loaded in a background, asynchronous.

For example, https://krug.org.pl/ is a static web page.

How to scrape them?

It’s quite easy, we just need to get an HTML code from a web page and parse it. We can do it even using net/http library:

```
require 'net/http'
source_code = Net::HTTP.get(URI('https://krug.org.pl'))
/our_regex/.match(source_code)
```

But it’s not a thing to parse HTML code manually, using regular expressions...

Which libraries (written in Ruby) can be used to scrape static web pages?

- Nokogiri gem - https://github.com/sparklemotion/nokogiri
- Mechanize gem - https://github.com/sparklemotion/mechanize

Nokogiri

Nokogiri

Nokogiri is an HTML and XML parser.
It parses and searches XML/HTML using native libraries (either C or Java, depending on your Ruby version), which means it's fast and standards-compliant.

It can be also used to build XML/HTML documents.

Nokogiri can’t be used to interact with webpages, it means that we can’t click on an element or fill a form with it.

Nokogiri - HTML web page parse example

```
require 'nokogiri'
require 'open-uri'
webpage = Nokogiri::HTML(open('http://www.nokogiri.org/tutorials/installing_nokogiri.html'))
# Search for nodes by css
webpage.css('nav ul.menu li a').each do |link|
  puts link.content
end
# Search for nodes by xpath
webpage.xpath('//nav//ul//li/a').each do |link|
  puts link.content
end
```

Mechanize

Mechanize

The Mechanize library is used to automate interaction with websites.
Mechanize automatically stores and sends cookies, follows redirects, and can follow links and submit forms.
It also keeps track of the sites that you have visited as a history.

Moreover, mechanize has already included nokogiri gem, so we can even parse web pages with it.

Mechanize - HTML web page interaction example

```
require 'mechanize'
browser = Mechanize.new { |agent| agent.user_agent_alias = 'Mac Safari' }
browser.get('http://google.com/') do |page|
  search_result = page.form_with(id: 'tsf') do |search|
    search.q = 'krug'
  end.submit

  search_result.links.each do |link|
    puts link.text
  end
end
```

Dynamic web apps

What is a dynamic web app (in web scraping)?

It’s an app where the whole content is not loaded once, there are elements which are loaded in a background, asynchronous.

For example, https://facebook.com is a dynamic web app.

What are the main problems during web scraping of dynamic web apps?

What are the main problem during web scraping dynamic web apps?

On dynamic web apps, a lot of content is loaded async.
It means that we need to monitor for all the changes or elements which we want to parse.
Also, some content may not be clickable - for example, a modal window covers part of a web page or an element does not persist on a page anymore.

Which tools can be used to scrape dynamic web apps?

Watir - https://github.com/watir/watir

An open-source Ruby library designed for automating tests - but it can be also used to write web scrapers.
Watir interacts with a browser in the same way people do: clicking links, filling out forms and validating text.
Moreover, it is powered by Selenium.

Which browsers does Watir support?

It’s pretty cool because Watir supports most popular browsers like:

- Chrome
- Safari
- Firefox
- Microsoft Edge (not fully supported yet)
- .. and even Microsoft Explorer!
- also can be run in a headless mode (for example with chrome-driver)

Main features:

- Waiting (for example, you can wait for N seconds for an element)
- Headless (you can run a browser without a monitor, or even in a background job processor)
- Basic browser authentication
- Browser alerts
- Browser downloads
- Cookies
- Screenshots
- Sending special keys (cmd + c etc.)
- Proxies
- Execute JavaScript code
- and much more...

Watir - example

```
require 'watir'
browser = Watir::Browser.new(:chrome)
browser.goto('https://google.com')
search_input = browser.text_field(class: 'gLFyf gsfi')
search_input.set('krug')
browser.form(id: 'tsf').submit

result = browser.div(class: 'rc', index: 2)
puts result.text

browser.close
```

krug.org.pl

Krakow Ruby Users Group (KRUG) (Kraków)

Would you like to share your concepts or dilemmas on one of our **KRUG** events? Or maybe you are up for sponsoring one of them? If your answer is yes to any ...

Of course, it’s just a really simple example.

We can really easy execute JavaScript code in a browser:

```
require 'watir'
browser = Watir::Browser.new(:chrome)
browser.goto('https://google.com')

browser.execute_script('alert("ok");')

browser.close
```

The headless mode does not require any screen, so you can run your code even in a background processor, like Sidekiq.

```
browser_flags = %w(ignore-certificate-errors disable-popup-blocking disable-translate disable-notifications start-maximized disable-gpu headless)
browser_options ||= begin
        selenium_options = Selenium::WebDriver::Chrome::Options.new
        browser_flags.each do |flag|
          selenium_options.add_argument(flag)
        end
        selenium_options
end

browser = Watir::Browser.new(:chrome, options: browser_options)
```

If you want to run Watir on Heroku, you need to specify a Chrome driver path and add a buildpack:

```
browser_options ||= begin
      selenium_options = Selenium::WebDriver::Chrome::Options.new
      # it requires the buildpack - https://github.com/jormon/minimal-chrome-on-heroku.git
      if chrome_bin = ENV['GOOGLE_CHROME_BIN']
        selenium_options.add_argument('no-sandbox')
        selenium_options.binary = chrome_bin
        Selenium::WebDriver::Chrome.driver_path = '/app/vendor/bundle/bin/chromedriver'
      end
      ...
      selenium_options
end
```

Of course, you can also use Watir to scrape static web pages!

It depends what do you want to achieve and how big is your/your company’s budget :)

Remember to find the right balance!

Some time ago (almost 3 years ago), I wrote an article which describes how to write a web scraper.
It scrapes Facebook and automatically likes the specified page and invites a friend.

You can read it here:
https://bit.ly/3avB10P

Thank you! Q&A time.