

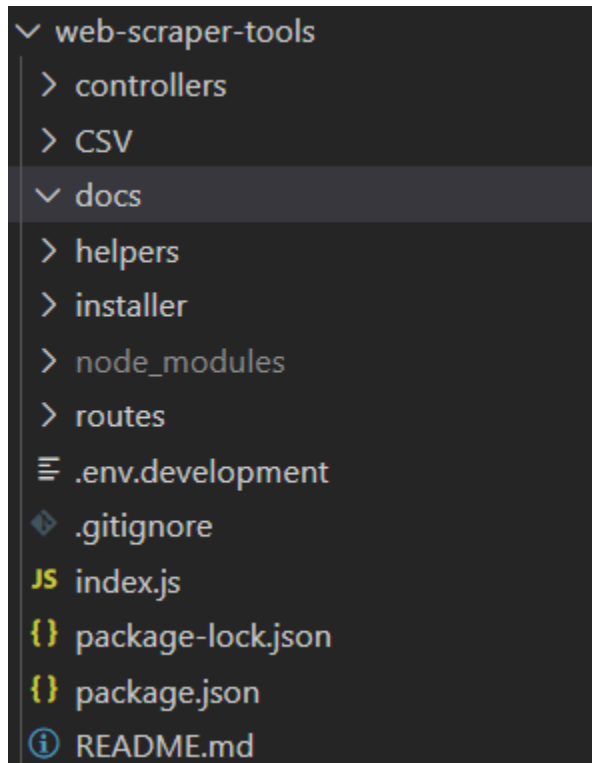
# HOW I SOLVE WEB SCRAPING PROBLEM

By Melvin Toni Gustave

---

## Step 1 - Create project main structure

At first my attempt is to create my own Restful API to scraping data from Tokopedia, so I initialize the project using NPM and Node.js. I also create main structure for this project by separate between middleware, routes, controllers and file helpers.



## Step 2 - Implement website-scrapers library

After the project structure completed, I try to find the library and found **website-scrapers** library from NPM. Basically how it works is by download the entire files from the URL we are given, like HTML files, image files, JS, etc. For small website it working fine and fast but for website like Tokopedia it just not a good option, because the process to download is slow and even though download is finished, I need to find necessary data from all those files, parse it and convert it

into CSV files. So basically it turned out the process was really slow. **Then I decided not to use this library and try to find another solution.**

### Step 3 - Implement Axios and Cheerio Library

After that I found an awesome article from <https://pusher.com/tutorials/web-scraper-node>, from this web I learn about the concept and how to implement the correct library for web scraper.

Then I decided to use **Axios as lib for make an HTTP request and get DOM HTML** also use **Cheerio as a lib to access the DOM easily with jQuery syntax-like.**

With Axios I can get the DOM HTML structure and then with Cheerio I pick the HTML parent of item list, then I looping through that parent div and get necessary data from each DOM.

```
axios(url).then((response) => {
  const html = response.data;
  const $ = cheerio.load(html);
  const parentDiv = $('div.css-1313178.e1nlzf110 > div');
  const result = [];

  parentDiv.each(function () {
    result.push({
      product_name: $(this).find('a > div.css-16vw0vn > div.css-11s9vse > span').text(),
      description: $(this).find('a > div.css-16vw0vn > div.css-11s9vse span.css-1kr22w3:first-child').text(),
      image_link: $(this).find('a > div.css-16vw0vn').find('img').attr('src'),
      price: $(this).find('a > div.css-16vw0vn > div.css-11s9vse span.css-o5uqvq').text(),
      rating: $(this).find('a > div.css-16vw0vn > div.css-11s9vse > div.css-153qjw7 > div > img').length,
      store_name: $(this).find('a > div.css-16vw0vn > div.css-11s9vse span.css-1kr22w3:last-child').text()
    });
  });

  const fields = ['product_name', 'description', 'image_link', 'price', 'rating', 'store_name'];
  const opts = { fields };
  const csv = parse(result, opts);
  const filePath = './CSV/' + 'result.csv';
  fs.writeFileSync(filePath, csv);
});
```

### Step 4 - Implement Puppeteer Library

For static web or pre-populated web page, Axios work great but unfortunately Tokopedia implement **lazy-loading** for its website, which means the list of the product will be populated after user scroll to certain point. Then for that purpose I use **Puppeteer** library.

Puppeteer basically can simulate user behavior when access page, with these libs I will be able to implement auto-scrolling which automatically activates Tokopedia lazy-loading feature.

```

async function autoScroll(page) {
  await page.evaluate(async () => {
    await new Promise((resolve, reject) => {
      var totalHeight = 0;
      var distance = 100;
      var timer = setInterval(() => {
        var scrollHeight = document.body.scrollHeight;
        window.scrollBy(0, distance);
        totalHeight += distance;

        if(totalHeight >= scrollHeight) {
          clearInterval(timer);
          resolve();
        }
      }, 100);
    });
  });
}

```

After all data populated, the rest basically same like step no 3 which I will iterate the data and insert into JSON format.

## Step 5 - Return data in CSV format

After get all necessary data and return it into JSON format, I be able to convert it into CSV using the **json2csv** library and write the file into **CSV** folder.

```

const fields = ['product_name', 'description', 'image_link', 'price', 'rating', 'store_name'];
const opts = { fields };
const csv = parse(result, opts);
const filePath = './CSV/' + 'result.csv';
fs.writeFileSync(filePath, csv);

```

And thus the result appears successfully.

