OIM3640 - Problem Solving and Software Design

Web Scraping*

* *based on ZenRows-What is Web Scraping? In-Depth Guide*

What is Web Scraping?

- **Web Scraping** refers to the process of automatically extracting and collecting data from websites using specialized software.
- It is also commonly referred to as "web crawling" or "web spidering".

Why not API

- APIs provide a convenient way to access data from other systems
- Unfortunately, many services don't provide an API.
- Even when an API is available, it may only provide limited functionality, or it may be subject to usage restrictions such as rate limits or access fees.

What is Web Scraping Used For?

- **Price Monitoring**
  - **E-commerce**: tracking competition prices and availability.
  - **Financial services**: detect stock price changes, volume activity, anomalies, etc.
- **Lead Generation**
  - **Extract contact information**: names, email addresses, phones, or job titles.
  - **Identify new opportunities**, i.e., in Yelp, YellowPages, Crunchbase, etc.

What is Web Scraping Used For?

- **Market Research**
  - **Real Estate**: supply/demand analysis, market opportunities, trending areas, etc.
  - **Automotive/Cars**: dealers distribution, most popular models, best deals, etc.
  - **Travel and Accommodation**: available rooms, hottest areas, best discounts, prices by season, etc.
  - **Job Postings**: most demanded jobs. Industries on the rise. Biggest employers. Supply by sector, etc.
  - **Social Media**: brand presence and growing influencers tracking. New acquisition channels, audience targeting, etc.
  - **City Discovery**: track new restaurants, shops, trending areas, etc.

What is Web Scraping Used For?

- **Aggregation**: i.e. news from many sources.
- **Inventory and Product Tracking**
  - Collect product details and specs.
  - New products.
- **SEO (Search Engine Optimization)**: Keywords' relevance and performance. Competition tracking, brand relevance, new players' rank.
- **ML/AI/Data Science**: Collect massive amounts of data to train machine learning models; image recognition, predictive modeling, NLP.
- **Bulk downloads**: PDFs or massive Image extraction at scale.

Web Scraping Process

- Web scraping follows the same basic client-server communication process as a standard **HTTP** request:
    - The client (e.g. web scraper) sends a request to the server (e.g. website) for specific information.
    - The server responds with the requested information, typically in the form of **HTML**, CSS, or JavaScript code.
- Web scrapers typically **extract** the data they need by parsing the HTML code returned by the server.
    - Once the relevant data has been extracted, it can be further processed or saved in a variety of formats, such as CSV, JSON, or a database.

Request - made by the browser

- **URL**: the specific address on the website that the client is requesting data from.
- **Method**:
  - **GET** is used to retrieve data from the server.
  - **POST** is used to submit data, such as form input, to the server.
- **Headers**:
  - User-Agent, Cookies, Browser Language, etc
  - **Tricky** parts of communication. Websites strongly focus on this data to determine whether a request comes from a **human** or a **bot**. Web scrapers may need to modify headers to mimic human-like behavior and avoid detection.
- **Body**: contains user-generated input, such as form data, to be submitted to server.

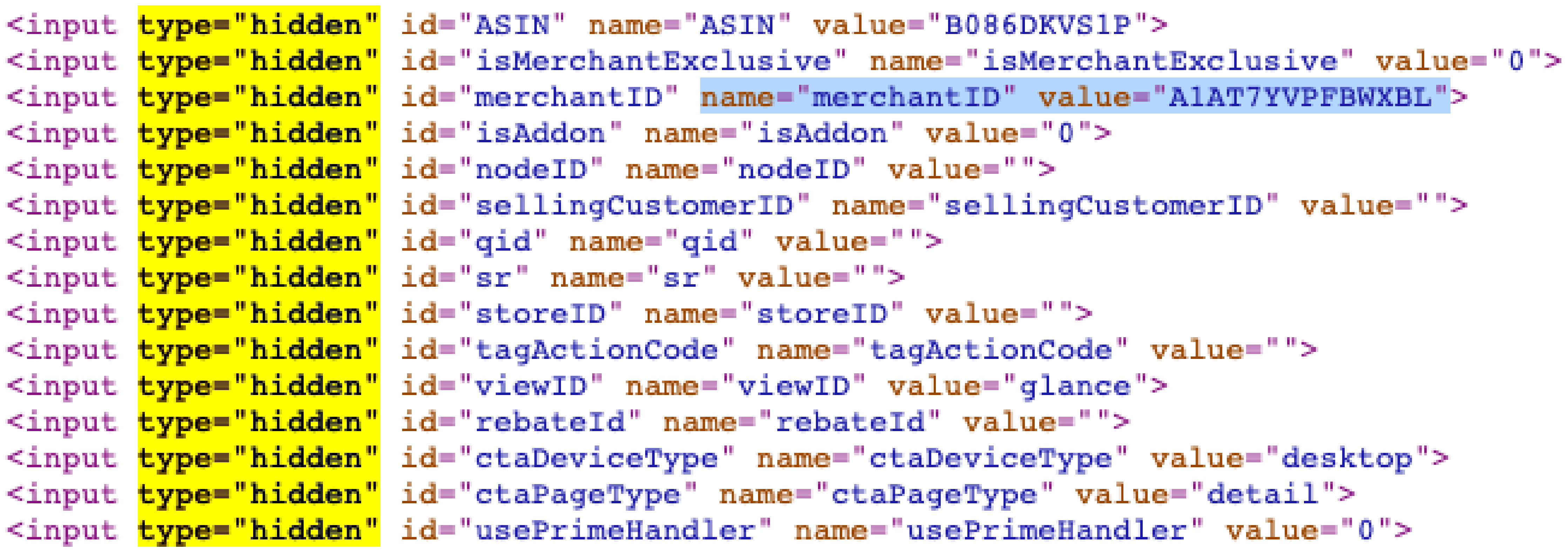
Response - returned by the server

- **HTTP Code**: a **number** indicating the status of the request.
  - **2xx** codes indicate success, such as **200** OK for a successful request.
  - **4xx** codes indicate client-side errors, such as the infamous **404** Not Found.
  - **5xx** codes indicate server-side errors, such as **500** Internal Server Error.
- **The content**:
  - **HTML**: responsible for rendering the website in a web browser.
  - **Auxiliary content types**: such as CSS, images, JSON, JavaScript, etc.
- **Headers**:
  - Just like Request Headers, these play a crucial role in communication.
  - One important part is instructing the browser to "**Set-Cookies**", which can be used to track a user's activity on a website.

Data Extraction - Parsing

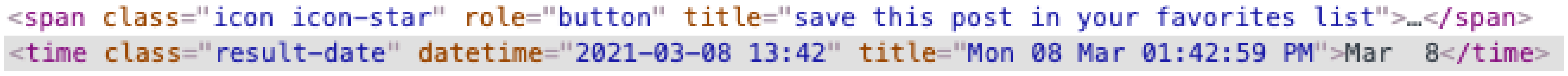
- The goal of web scraping is to extract **specific data** from a web page.
- **Parsing** is the process of extracting selected data and organizing it into a well-defined structure.
    - HTML is a **tree structure**, called the Document Object Model (DOM).
    - The extraction process begins by **analyzing** a website.
    - Popular Python parsing libraries such as `BeautifulSoup` and `Scrapy` can simplify the parsing process and make it easier to extract data from complex web pages.

Example: Hidden Inputs on Amazon Products

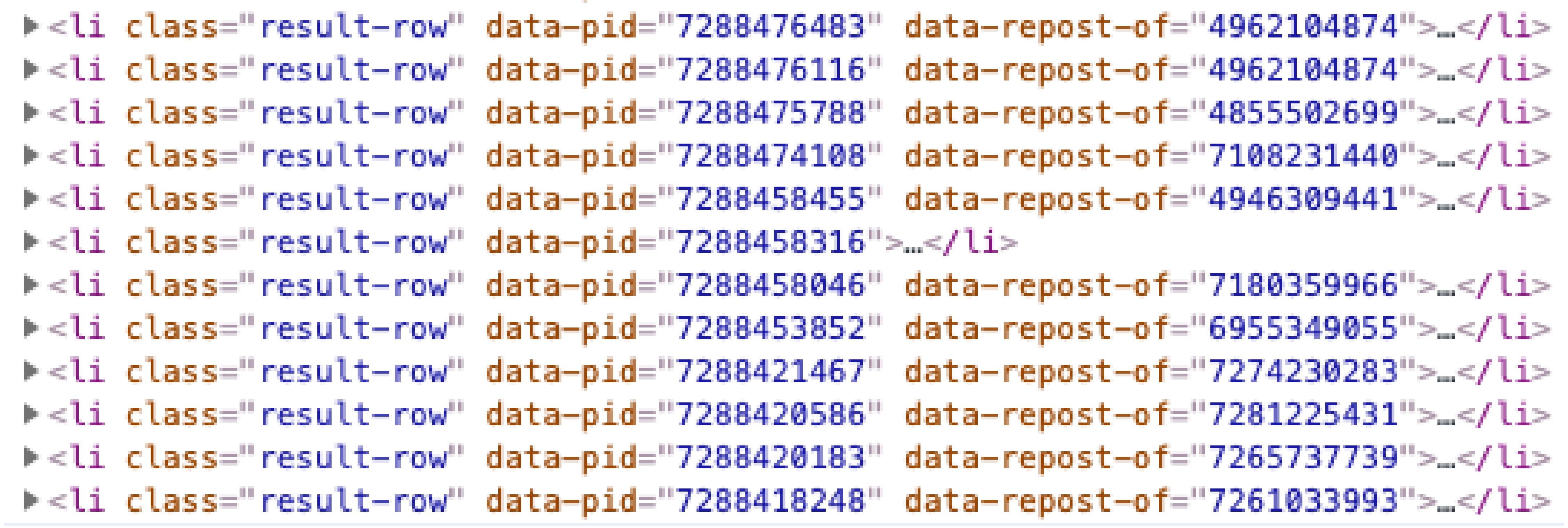


```
<input type="hidden" id="ASIN" name="ASIN" value="B086DKVS1P">
<input type="hidden" id="isMerchantExclusive" name="isMerchantExclusive" value="0">
<input type="hidden" id="merchantID" name="merchantID" value="A1AT7YVPFBWXBL">
<input type="hidden" id="isAddon" name="isAddon" value="0">
<input type="hidden" id="nodeID" name="nodeID" value="">
<input type="hidden" id="sellingCustomerID" name="sellingCustomerID" value="">
<input type="hidden" id="qid" name="qid" value="">
<input type="hidden" id="sr" name="sr" value="">
<input type="hidden" id="storeID" name="storeID" value="">
<input type="hidden" id="tagActionCode" name="tagActionCode" value="">
<input type="hidden" id="viewID" name="viewID" value="glance">
<input type="hidden" id="rebateId" name="rebateId" value="">
<input type="hidden" id="ctaDeviceType" name="ctaDeviceType" value="desktop">
<input type="hidden" id="ctaPageType" name="ctaPageType" value="detail">
<input type="hidden" id="usePrimeHandler" name="usePrimeHandler" value="0">
```

Example: HTML Attributes on Craiglist



```
<span class="icon icon-star" role="button" title="save this post in your favorites list">…</span>
<time class="result-date" datetime="2021-03-08 13:42" title="Mon 08 Mar 01:42:59 PM">Mar  8</time>
```



```
▶<li class="result-row" data-pid="7288476483" data-repost-of="4962104874">…</li>
▶<li class="result-row" data-pid="7288476116" data-repost-of="4962104874">…</li>
▶<li class="result-row" data-pid="7288475788" data-repost-of="4855502699">…</li>
▶<li class="result-row" data-pid="7288474108" data-repost-of="7108231440">…</li>
▶<li class="result-row" data-pid="7288458455" data-repost-of="4946309441">…</li>
▶<li class="result-row" data-pid="7288458316">…</li>
▶<li class="result-row" data-pid="7288458046" data-repost-of="7180359966">…</li>
▶<li class="result-row" data-pid="7288453852" data-repost-of="6955349055">…</li>
▶<li class="result-row" data-pid="7288421467" data-repost-of="7274230283">…</li>
▶<li class="result-row" data-pid="7288420586" data-repost-of="7281225431">…</li>
▶<li class="result-row" data-pid="7288420183" data-repost-of="7265737739">…</li>
▶<li class="result-row" data-pid="7288418248" data-repost-of="7261033993">…</li>
```

Web Scraping Challenges

- **Legal** Issues:
  - Still a gray area, YMMV
- **Ethical** Issues: could be used for malicious purposes
- **Technical** Challenges:
  - IP Rate Limit, Rotating Proxies,Headers/Cookies validation, Reverse Engineering Headers / Cookies generation, Javascript Execution, Headless Browsers, Captcha / reCAPTCHA (Developed by Google), Pattern Recognition, ...
- **Data Quality** Challenges:
  - Inconsistent or poorly structured data
  - Data cleaning and preprocessing is important.

Web Scraping - Example

- Install **Beautiful Soup**
  - `python -m pip install beautifulsoup4`
- Download imdb_top250.py
- Try with Yahoo Finance Trending Stocks