BAB II
DASAR TEORI

2.1. Struktur Web

HTML, CSS dan *JavaScript* adalah bahasa pemrograman yang penting digunakan dalam sebuah pengembangan web (1Training, 2017). *Html* dan *css* dibuat untuk mengatur konten *style* dan struktur dari sebuh halaman web. Sedangkan *JavaScript* dibuat untuk menambahkan fitur interaktif pada halaman web (1Training, 2017).

2.1.1. HTML

HTML *(HyperText Markup Language)* adalah sebuah bahasa pemrograman yang digunakan untuk mengembangkan sebuah web . Web yang dibangun dengan format *html* terdiri dari beberapa elemen yang saling berkaitan satu sama lain. Tiap-tiap elemen diapit oleh tanda lebih kecil (<) dan tanda lebih besar (>), contohnya <HTML> yang disebut dengan *tag*. Selain elemen, dalam format *HTML* juga terdapat atribut. Atribut memiliki tugas khusus untuk memberikan informasi atau sifat tambahan yang akan diberikan kepada *tag* dan elemen yang mengandung atribut tersebut.

Hampir semua *tag* dalam dokumen *html* harus diakhiri dengan *tag* penutup, yang menandakan selesainya perintah tersebut. *Tag* penutup dituliskan dengan menambahkan karakter *slash* (/), contohnya <HTML/>. Secara umum format *tag HTML* tersebut dapat dituliskan sebagai berikut :

```
<TAG>Teks yang akan dipengaruhi oleh tag</TAG>
```

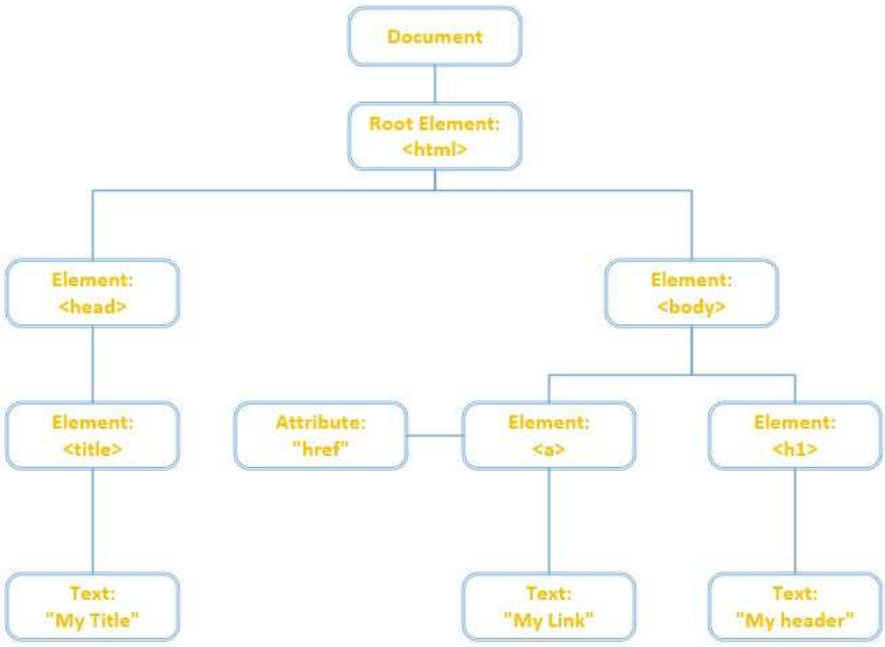
Tidak semua *tag html* memiliki *tag* penutup. Beberapa *tag* hanya memiliki tag pembuka. *Tag* tersebut dinamakan *tag* tunggal, yang digunakan untuk mendefinisikan elemen-elemen yang tidak memiliki awal dan akhir. Sebagai contoh *tag line-break* (ganti baris), yang dituliskan dengan

Secara umum sebuah dokumen *html* terbagi atas dua bagian utama, yang pertama adalah ”*head*” atau kepala sebuah dokuman yang kedua ”*body*” atau isi sebuah dokumen *html* yang akan ditampilkan. Dokumen *html* paling sederhana terdiri atas :

```
<html>
<head>
<title>judul halaman web</title>
</head>
<body>
isi halaman web
<!--ini adalah komentar-->
</body>
```

Struktur *html* menggunakan *DOM*. *Document Object Model* (DOM) merupakan sebuah ketentuan yang dikembangkan oleh W3C untuk berinteraksi dengan objek-objek yang ada di dalam *HTML, XML*, maupun *XHTML* (Bertzzie, n.d.). *DOM* bersifat *cross-platform* dan *language-independent*, yang artinya *DOM* dapat digunakan dengan bahasa pemrograman apapun, dalam sistem operasi manapun. Standar *DOM* dikembangkan untuk berinteraksi dengan elemen-elemen dokumen *HTML* dan *XML*, mulai dari pembuatan elemen baru sampai dengan manipulasi dan penghapusan elemen.

Sebagai contoh, *root* elemen *html* memiliki dua elemen di dalamnya, yaitu elemen *head* dan elemen *body*. Di dalam elemen *head* terdapat elemen *tittle* yang berisi teks untuk *tittle* itu sendiri. Sedangkan dalam elemen *body* berisi dua elemen yaitu elemen *h1* dan elemen *a*. Elemen *h1* merupakan ukuruan dari teks *My Header* dan elemen *a* yang mempunyai atribut *href*. Atribut *href* digunakan untuk memasukan teks berupa *link url* pada elemen *a*. Sederhananya, contoh dari struktur *html* diatas dapat direpresentasikan sebagai sebuah pohon pada Gambar 2-1.



Gambar 2-1. Struktur *HTML*

Pada Gambar 2-1 dapat dilihat bahwa *html* memiliki banyak elemen-elemen di dalamnya. Dan pada elemen-elemen di dalam *html* juga mengandung elemen di dalamnya. *Html* adalah sebagai *root* elemen, yaitu elemen yang menampung seluruh elemen yang ada didalamnya. Sedangkan elemen yang ada di dalam *root* elemen disebut sebagai *leaf.*

Pengetahuan atau teori tentang struktur dokumen *html* ini dibutuhkan untuk memahami isi dari dokumen *html*. Pemahaman tentang elemen, *tag* dan atribut digunakan untuk memilih elemen-elemen dan atribut yang digunakan untuk proses *scraping*. Lokasi data yang akan diekstraksi dalam proses *scraping* adalah data teks yang berada di dalam elemen-elemen *html*. Sehingga digunakan *Xptah selector* untuk memilih data-data yang ada pada masing-masing elemen yang akan di ekstraksi.

2.1.2. CSS

CSS merupakan kumpulan kode-kode yang bertujuan untuk menghias dan mengatur gaya tampilan*/layout* halaman web supaya lebih elegan dan menarik (Rahman, 2013). *CSS* adalah singkatan dari "*Cascading Style Sheets*" dan setiap *style sheet* menunjukkan sebuah dokumen. *CSS* juga membantu untuk menyesuaikan halaman web sesuai dengan jenis perangkat dan ukuran layar yang menampilkan halaman web tersebut. *Cascading Style Sheet* terdiri dari *selector*, deklarasi, properti dan *value.* Seperti pada *html, php* dan bahasa pemrograman lainnya, css juga memiliki aturan penulisan untuk dirinya sendiri. Contoh penulisan kode *css* :

```
Body {background-color: white;}
```

Body adalah Selektor, { } adalah deklarasi, *background-color* adalah properti dan *white* adalah value. Maksud dari kode diatas adalah mengatur warna latar belakang (*background color*) dari tag *Body* sebuah halaman web.

2.1.3. JavaScript

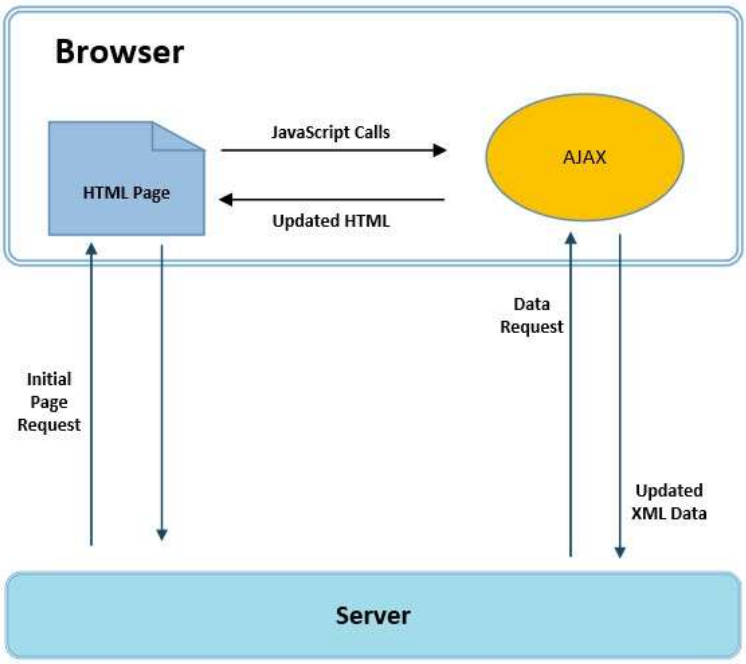
JavaScript adalah bahasa pemrograman yang dinamis. *JavaScript* populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti *Internet Explorer* (*IE*), *Mozilla Firefox, Netscape, Chrome* dan *Opera* (Wongso, 2017). Kode *JavaScript* dapat disisipkan dalam halaman web menggunakan *tag SCRIPT. JavaScript* dapat mengakses ke *HTML* melalui *DOM*. Dengan menggunakan model objek, *JavaScript* mendapatkan melakukan banyak hal yang dibutuhkan untuk membuat *HTML* dinamis:

1. *JavaScript* dapat mengubah semua elemen *HTML*
2. *JavaScript* dapat mengubah semua attibut *HTML*
3. *JavaScript* dapat mengubah semua *style CSS*
4. *JavaScript* dapat menghapus elemen dan atribut yang ada pada *HTML*
5. *JavaScript* dapat menambahkan elemen dan atribut *HTML* baru
6. *JavaScript* dapat bereaksi terhadap semua peristiwa yang ada pada *HTML*
7. *JavaScript* dapat membuat peristiwa *HTML* baru di halaman

Salah satu metode *JavaScript* yang sering digunakan di dalam sebuah web adalah *Ajax*. *Ajax* digunakan untuk mengirim dan menerima data dari server tanpa harus menyegarkan (*refresh*) laman itu. Pemahaman tentang *JavaScript* dan *Ajax* sangat dibutuhkan dalam proses *scraping*, kerena web yang akan diekstraksi adalah web yang menggunakan *Ajax* didalamnya.

2.2. AJAX

AJAX adalah singkatan dari *Asynchronous JavaScript and XML*. Pada dasarnya *ajax* menggunakan *XMLHttpRequest* object *JavaScript* untuk membuat *request* ke server secara *asynchronous* atau tanpa melakukan *refresh* pada halaman *website*. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability*. *Ajax* berkomunikasi dengan server untuk mengambil data dari *request JavaScript* tanpa mengubah tampilan web. Yang dibutuhkan agar *ajax* dapat berjalan adalah *javascript* harus di *enable* pada *browser* yang digunakan. *Ajax* dapat digunakan untuk melakukan banyak hal, seperti *loading* halaman *html* tanpa *refresh* halaman web, validasi *form* dan lain-lain.



Gambar 2-2. Arsitektur *Ajax*

Arsitektur *ajax* dapat dilihat pada Gambar 2-2. *Browser* melakukan *initial page request* kepada server untuk menampilkan *html Page*. Untuk memuat *ajax*, *javascript* melakukan *request* ke ajax dan *ajax* meneruskan *request* ke server untuk mendapatkan data *XML*. Setelah mendapatkan data *XML*, *ajax* akan melakukan *update html* tanpa melakukan *refresh* pada halaman *website.*

Pemahaman tentang *ajax* dibutuhkan untuk pengembangan sistem *scraping*. Karena lokasi data yang akan diekstraksi terletak pada bagian *ajax*. *Selenium driver* adalah salah satu *webdriver* yang dapat memproses *ajax*.

2.3. Scraping

Scraping adalah salah satu teknik yang digunakan untuk mengekstraksi data dari halaman *website* (Foxer, 2017). Untuk melakukan *scraping* terdapat banyak *tools* yang dapat digunakan seperti *Rapid Miner*, *Import.io*, *Scrapy* dan lain-lain. Tidak semua data akan diambil dalam proses *scraping*. Data yang di ekstraksi dari halaman web adalah data teks dari dokumen *html*. Hasil data dari ekstraksi dapat disimpan dalam suatu database. Proses *scraping* dimulai dari *request url* sampai menghasilkan *output* data yang diinginkan.

Secara umum dalam pengimplementasian *scraping* dibutuhkan 5 tahap yaitu :

1. *Request* url web yang akan dijadikan target.
2. *Request* diproses oleh *server* atau *engine*.
3. Hasil dari *request* url.
4. Proses ekstraki data.
5. Hasil data yang telah berhasil di ekstraksi.

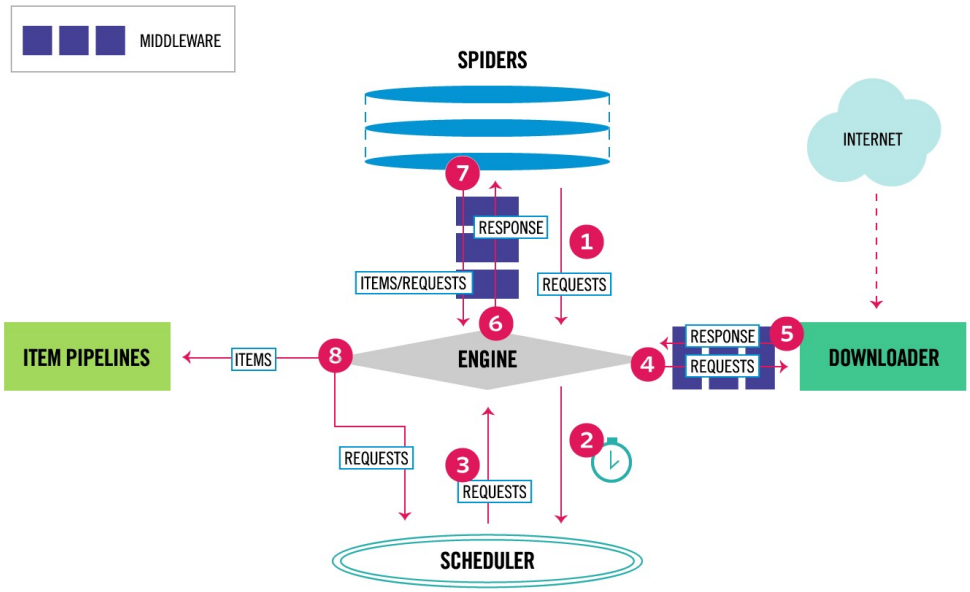
Tahap-tahap tersebut juga dapat dilihat pada Gambar 2-3.

Gambar 2-3. Tahap-tahap *scraping*

2.3.1. Scrapy

Scrapy adalah salah satu *framework opensource* yang digunakan untuk mengekstrak data dari *website* (Scrapy Documentation, 2016). *Scrapy* dibangun dengan bahasa pemrograman *Python*. *Scrapy* dapat mengekstraksi data yang tidak testruktur dari sebuah web dan menghasilkan data yang terstruktur.

Selain mengekstraksi data, *scrapy* juga dapat digunakan untuk melakukan *crawling. Crawling* digunakan untuk membuka semua halaman web secara otomatis yang ada pada halaman yang akan di *scraping.* Jadi, dengan menggunakan *framework* scrapy proses *scraping* dan *crawling* dapat dilakukan secara bersamaan. Hasil dari ekstraksi *scrapy* dapat disimpan dalam format *file* json maupun langsung dapat disimpan dalam database. Gambar 2-4 merupakan gambar arsitektur *scrapy*.



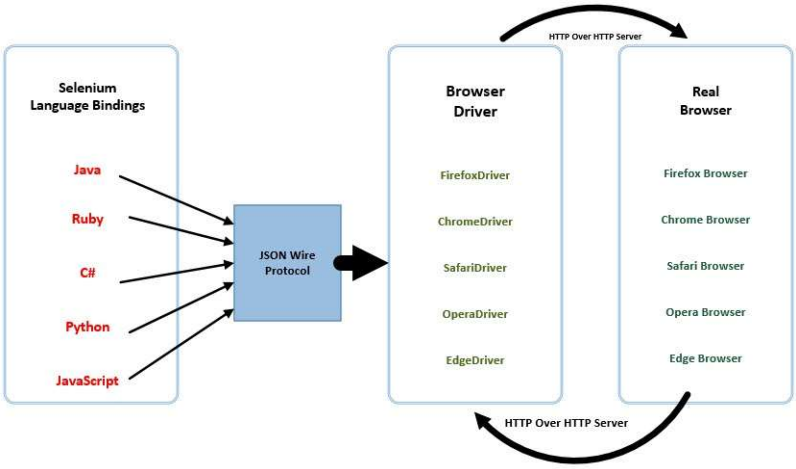
Gambar 2-4. Arsitektur *Scrapy (Scrapy Documentation, 2016)*

Proses *scraping* pada arsitektur *scrapy* dapat dijelaskan sebagai berikut :

1. *Spider* melakukan *start request* kepada *Engine*.
2. *Scrapy engine* menyalurkan *request* kepada *Scheduler.* Tugas *Scheduler* adalah untuk menentukan atau mengatur *Scheduler* dari semua *request url* yang diterima.
3. Ketika *Scheduler* sudah menentukan *request url* yang akan dijalankan, maka akan dikembalikan ke *Engine.*
4. *Engine* akan melakukan *request* dari *Scheduler* kepada *Downloader* melalui *Downloader Middlewares* untuk men-*download* data.
5. *Engine* selanjutnya akan menerima respon dari *Downloader* berupa data yang telah di *download*.
6. Selanjutnya, respon tersebut akan dikirim ke *Spider* melalui *Spider Middleware* untuk diproses.
7. *Spider* melakukan proses terhadap respon dan mengirimkannya ke *Engine*.
8. Setelah itu *Engine* akan mengirimkan respon yang telah diproses ke *Item Pipeline*.
9. Proses dari 1 sampai 8 terus berulang sampai tidak ada lagi *request* dari *Scheduler*.

2.3.2. Selenium

Selenium digunakan untuk mengontrol *browser*. *Selenium* sebagai *middleware* antara *scrapy* dan *chrome browser*. Dalam proses *scraping*, selenium berfungsi untuk mengendalikan dan membaca hasil data yang telah di *download* oleh browser.



Gambar 2-5. Arsitektur *Selenium WebDriver*

Gambar 2-5 merupakan gambar arsitektur *selenium*. Pada bagian *Selenium Language Bindings* merupakan bahasa-bahasa pemrograman yang digunakan oleh *selenium*. *JSON Wire Protokol* digunakan untuk mengendalikan *browser driver*. *Browser Driver* akan melakukan *request* kepada *Real Browser* melalui protokol *HTTP* lalu *Browser Driver* akan menerima respon dari *Real Browser.*