MODUL 12: WEB CRAWLING DAN SCRAPING

12.1 Deskripsi Singkat

Web crawling adalah teknik pengumpulan data yang digunakan untuk mengindeks informasi pada halaman menggunakan URL (Uniform Resource Locator) dengan menyertakan API (Application Programming Interface) untuk melakukan penambangan dataset yang lebih besar. Sedangkan web scraping adalah teknik mengekstraksi data dari suatu halaman web. Web scraping merupakan cara yang powerful untuk mengumpulkan data ketika website tidak menyediakan data API.

12.2 Tujuan Praktikum

Setelah praktikum pada modul 12 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

1) Dapat mempraktekkan teknik scraping untuk mengekstraksi elemen tertentu dari halaman web.
2) Dapat mempraktekkan teknik scraping dengan headless browsers.

12.3 Material Praktikum

Tidak ada

12.4 Kegiatan Praktikum

A. Scraping menggunakan Request dan BeautifulSoup

Selanjutnya Anda akan melakukan scraping suatu halaman website menggunakan library Request dan BeautifulSoup. Import terlebih dahulu library yang diperlukan.

```
from bs4 import BeautifulSoup as Soup
import os
import re
import requests
```

Buat fungsi berikut untuk mengekstraksi konten dari suatu halaman website.

```
def downloader(link):
    req = requests.get(link)
    req.encoding = "utf8"
    return req.text
```

Panggil fungsi tersebut, misalnya untuk mengakses halaman berikut.

```
https://jurnal.stis.ac.id/index.php/jurnalasks/
```

```
contents =
downloader("https://jurnal.stis.ac.id/index.php/jurnalasks/")

print(contents)
```

Perhatikan isi dari variabel contents di atas. Kemudian bandingkan dengan membuka halaman website lalu klik `View Page Source`.

Anda dapat merapikan tampilan output menggunakan kode berikut.

```
soup = Soup(contents, "lxml")
print(soup.prettify())
```

Selanjutnya Anda akan mencoba mengakses elemen html dari halaman web tersebut. Misalnya Anda akan mengakses tag html title.

```
soup.title
```

Anda juga dapat mengakses elemen html dengan atribut tertentu. Misalnya Anda akan mengakses elemen link dengan id = article-532.

```
soup.find_all("a", attrs={"id": "article-532"})
```

Selain itu, Anda juga dapat menerapkan regular expression untuk memfilter isi atribut yang diakses. Misalnya Anda akan mengakses elemen link dengan id yang terdapat string ”article”.

```
urls = soup.find_all("a", attrs={"id": re.compile(r"(article)")})
```

Kemudian ekstraksi judul dari setiap link jurnal tersebut.

```
for u in urls:
  content_u = downloader(u['href'])
  soup_u = Soup(content_u, "lxml")
  print(soup_u.find("h1", attrs={"class": "page_title"}).text)
```

Ekstraksi pula nama author pada artikel yang ditampilkan di halaman beranda jurnal pada halaman beranda, kemudian simpan ke dalam csv.

B. Scraping menggunakan Headless Browser

Selanjutnya Anda akan melakukan scraping suatu halaman website yaitu Playwright. Playwright adalah suatu cross-platform dan cross-language web browser automation toolkit. Fungsi utamanya yaitu digunakan sebagai website test suite, namun juga sangat mampu digunakan sebagai browser automation dan web scraping. Playwright dapat digunakan untuk mengotomatisasi web headless browsers seperti Firefox atau Chrome untuk menelusuri web seperti yang dapat dilakukan manusia, seperti: mengunjungi URLs, klik buttons, menulis teks dan mengeksekusi javascript.

Scraping Informasi Tempat pada Google Maps

1. Lakukan instalasi playwright terlebih dahulu.

   ```
   pip install playwright
   ```

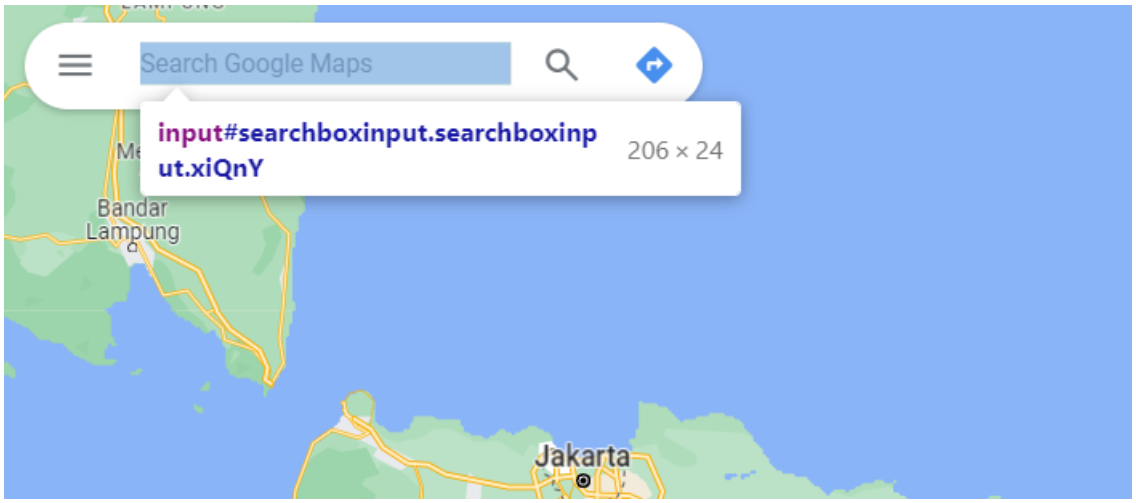
2. Kemudian instal browser yang akan digunakan. Pada modul ini, akan digunakan browser Chromium.
3. Tulis kode berikut untuk melakukan otomatisasi mengetikkan teks pada search box Google Maps.

   ```
   from playwright.sync_api import sync_playwright
   with sync_playwright() as p:
       browser = p.chromium.launch(headless=False)
       page = browser.new_page()
   ```

```
    page.goto("https://www.google.com/maps", timeout=60000)
    # wait is added for dev phase. can remove it in production
    page.wait_for_timeout(5000)

    page.locator('//input[@id="searchboxinput"]').fill("museum")
    page.wait_for_timeout(3000)

    page.keyboard.press("Enter")
    page.wait_for_timeout(5000)
    browser.close()
```

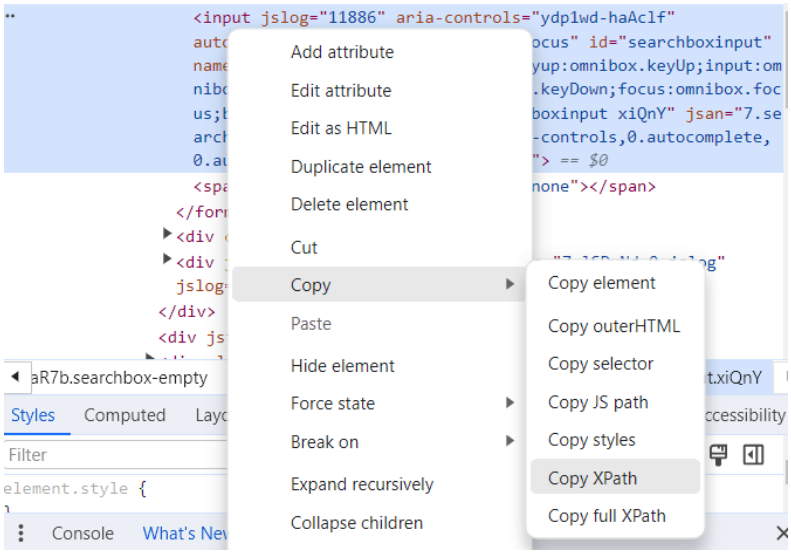
Perhatikan yang terjadi pada browser yang otomatis terbuka.

4. Untuk memahami kode di atas, jalankan brwoser, buka halaman Google Maps. Lalu lakukan inspect element pada button search.



Id dari search box di atas adalah searchboxinput. Pada Playwright, Anda dapat melakukan mencari suatu elemen pada web dengan menggunakan fungsi `Locator`.

5. Suatu elemen tertentu dapat diakses berdasarkan Xpath. Misalnya, setelah melakukan inspect elemen search box, copy XPath dengan cara melakukan klik kanan pada elemen tersebut sebagai berikut.



XPath yang tersalin yaitu: `//*[@id="searchboxinput"]`

6. Kemudian buat file python baru, lalu tulis kode berikut untuk membuat struktur data yang akan dikumpulkan.

```
"""This script serves as an example on how to use Python
   & Playwright to scrape/extract data from Google Maps"""
from __future__ import annotations
from dataclasses import dataclass, asdict, field
import pandas as pd

@dataclass
class Business:
    """holds business data"""

    name: str = None
    address: str = None
    website: str = None
    phone_number: str = None
    reviews_count: int = None
    reviews_average: float = None


@dataclass
class BusinessList:
    """holds list of Business objects,
    and save to both excel and csv
    """

    business_list: list[Business] = field(default_factory=list)

    def dataframe(self):
        """transform business_list to pandas dataframe

        Returns: pandas dataframe
        """
        return pd.json_normalize(
            (asdict(business) for business in self.business_list),
sep="_"
        )

    def save_to_excel(self, filename):
        """saves pandas dataframe to excel (xlsx) file

        Args:
            filename (str): filename
        """
        self.dataframe().to_excel(f"{filename}.xlsx", index=False)

    def save_to_csv(self, filename):
        """saves pandas dataframe to csv file

        Args:
            filename (str): filename
        """
        self.dataframe().to_csv(f"{filename}.csv", index=False)
```

7. Kemudian buat fungsi main berikut untuk melakukan scraping data suatu tempat dari Google Maps berdasarkan kata kunci yang diketikkan.

```
from playwright.sync_api import sync_playwright
def main():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False)
        page = browser.new_page()

        page.goto("https://www.google.com/maps", timeout=60000)
        # wait is added for dev phase. can remove it in production
        page.wait_for_timeout(5000)
        page.locator('//input[@id="searchboxinput"]').fill(search_for)
        page.wait_for_timeout(3000)

        page.keyboard.press("Enter")
        page.wait_for_timeout(5000)

        # scrolling
        page.hover('//a[contains(@href,
"https://www.google.com/maps/place")]')

        # this variable is used to detect if the bot
        # scraped the same number of listings in the previous iteration
        previously_counted = 0
        while True:
            page.mouse.wheel(0, 10000)
            page.wait_for_timeout(3000)

            if (
                page.locator(
                    '//a[contains(@href,
"https://www.google.com/maps/place")]'
                ).count()
                >= total
            ):
                listings = page.locator(
                    '//a[contains(@href,
"https://www.google.com/maps/place")]'
                ).all()[:total]
                listings = [listing.locator("xpath=..") for listing in
listings]
                print(f"Total Scraped: {len(listings)}")
                break
```

```
            else:
                # logic to break from loop to not run infinitely
                # in case arrived at all available listings
                if (
                    page.locator(
                        '//a[contains(@href,
"https://www.google.com/maps/place")]'
                    ).count()
                    == previously_counted
                ):
                    listings = page.locator(
                        '//a[contains(@href,
"https://www.google.com/maps/place")]'
                    ).all()
                    print(f"Arrived at all available\nTotal Scraped:
{len(listings)}")
                    break
                else:
                    previously_counted = page.locator(
                        '//a[contains(@href,
"https://www.google.com/maps/place")]'
                    ).count()
                    print(
                        f"Currently Scraped: ",
                        page.locator(
                            '//a[contains(@href,
"https://www.google.com/maps/place")]'
                        ).count(),
                    )

        business_list = BusinessList()

        # scraping
        for listing in listings:
            listing.click()
            page.wait_for_timeout(5000)

            name_xpath = '//div[contains(@class,
"fontHeadlineSmall")]'
            address_xpath = '//button[@data-item-
id="address"]//div[contains(@class, "fontBodyMedium")]'
            website_xpath = '//a[@data-item-
id="authority"]//div[contains(@class, "fontBodyMedium")]'
            phone_number_xpath = '//button[contains(@data-item-id,
"phone:tel:")]//div[contains(@class, "fontBodyMedium")]'
            reviews_span_xpath = '//span[@role="img"]'

            business = Business()

            if listing.locator(name_xpath).count() > 0:
                business.name =
listing.locator(name_xpath).inner_text()
            else:
                business.name = ""
            if page.locator(address_xpath).count() > 0:
                business.address =
page.locator(address_xpath).inner_text()
```

```
            else:
                business.address = ""
            if page.locator(website_xpath).count() > 0:
                business.website =
page.locator(website_xpath).inner_text()
            else:
                business.website = ""
            if page.locator(phone_number_xpath).count() > 0:
                business.phone_number =
page.locator(phone_number_xpath).inner_text()
            else:
                business.phone_number = ""


            if listing.locator(reviews_span_xpath).count() > 0:
                business.reviews_average = float(
                    listing.locator(reviews_span_xpath)
                    .get_attribute("aria-label")
                    .split()[1]
                    .replace(",", ".")
                    .strip()
                )
                business.reviews_count = int(
                    listing.locator(reviews_span_xpath)
                    .get_attribute("aria-label")
                    .split()[2]
                    .replace(".", "")
                    .strip()
                )
            else:
                business.reviews_average = ""
                business.reviews_count = ""
            print(business)
            business_list.business_list.append(business)
        print(business_list)
        # saving to both excel and csv just to showcase the features.
        business_list.save_to_excel("google_maps_data")
        business_list.save_to_csv("google_maps_data")

        browser.close()
```

Pastikan Anda memahami kode di atas.

8. Kemudian buat fungsi main untuk memanggil fungsi di atas.

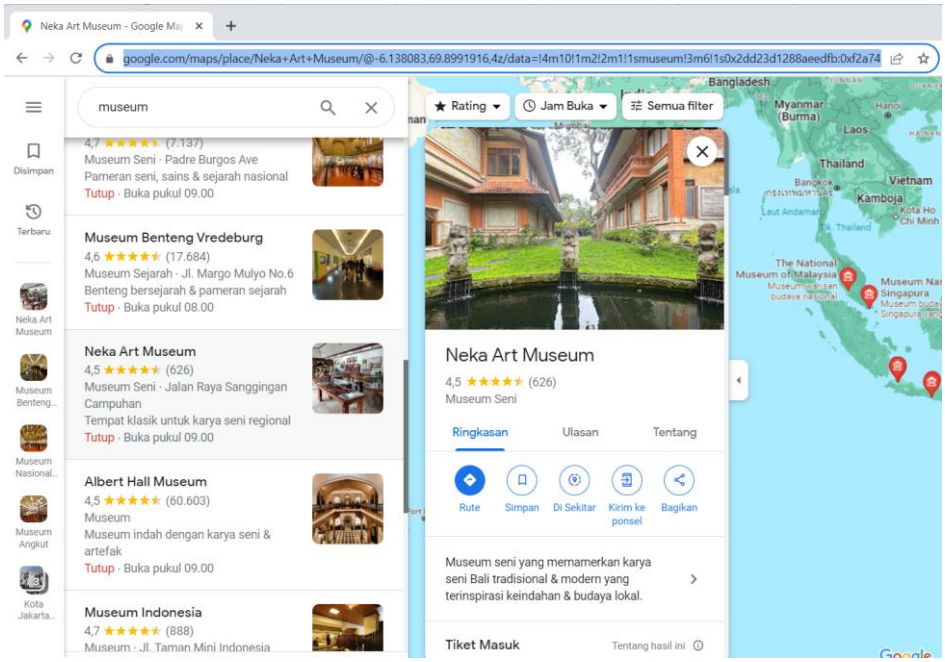
```
import argparse
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-s", "--search", type=str)
    parser.add_argument("-t", "--total", type=int)
    args = parser.parse_args()

    if args.search:
        search_for = args.search
    else:
        # in case no arguments passed
        # the scraper will search by defaukt for:
        search_for = "museum"
```

```
    else:
        # in case no arguments passed
        # the scraper will search by defaukt for:
        search_for = "museum"

    # total number of products to scrape. Default is 10
    if args.total:
        total = args.total
    else:
        total = 10

    main()
```

Anda dapat menjalankan fungsi di atas dengan menggunakan arguments, misalnya:

```
python scraper2.py -s museum -t 10
```

Perhatikan yang terjadi pada browser.



12.5 Penugasan

1. Modifikasi kode scraping dengan Playwright untuk melakukan scraping review suatu tempat pada Google Maps. Misalnya tempat yang akan dikunjungi yaitu Jakarta Aquarium dengan link sebagai berikut:

   ```
   https://maps.app.goo.gl/SyXMj8rCRsco4JM38
   ```

   Petunjuk:

   a. Dengan menggunakan browser, buka link Maps untuk Jakarta Aquarium, kemudian akses tab Ulasan. Pilih salah satu ulasan teratas, dan identifikasi XPath dari elemen yang akan dituju untuk scraping. Kemudian bandingkan Xpath untuk elemen review kedua dan ketiga.

Perhatikan pola yang terbentuk. Kemudian, perhatikan pula XPath untuk elemen nama pemberi ulasan dan teks ulasannya.

b. Buat dataclass Review dan ReviewList dengan memodifikasi kode pada fungsi Business dan BusinessList. Misalnya atribut yang ingin disimpan yaitu id review, nama, dan teks review.

```
@dataclass
class Review:
    """holds reviews data"""

    id_review: str = None
    name: str = None
    review_text: str = None
```

c. Buat fungsi main untuk melakukan scraping, misalnya untuk sebanyak 10 review.

```
def main():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False)
        page = browser.new_page()
        page.goto("https://maps.app.goo.gl/SyXMj8rCRsco4JM38",
timeout=60000)
        page.wait_for_timeout(5000)
        page.locator('button:has-text("Ulasan lainnya")').click();
        review_list = ReviewList()

        page.wait_for_timeout(5000)

        for i in range(1,10):
            review = Review()
            review_element =
page.query_selector('//*[@id="QA0Szd"]/div/div/div[1]/div[2]/div/div[
1]/div/div/div[2]/div[9]/div['+str(3*i+-
2)+']/div/div/div[2]/div[2]/div[1]/button')
            review_id = review_element.get_attribute('data-review-
id')
            reviewer_name_xpath =
'//*[@id="QA0Szd"]/div/div/div[1]/div[2]/div/div[1]/div/div/div[2]/di
v[9]/div['+str(3*i+-2)+']/div/div/div[2]/div[2]/div[1]/button/div[1]'
            reviewer_name =
page.locator(reviewer_name_xpath).inner_text()
            review_xpath='//*[@id="'+review_id+'"]'
            if "… Lainnya" in
page.locator(review_xpath).inner_text():
                button_lainnya_xpath =
'//*[@id="'+review_id+'"]/span[2]/button'
                page.locator(button_lainnya_xpath).click();
            review_text = page.locator(review_xpath).inner_text()
            review.id_review = review_id
            review.name = reviewer_name
            review.review_text = review_text
            print(review)
            review_list.review_list.append(review)
```

```
        review_list.save_to_csv("google_maps_review_data")
        browser.close()
```

Pastikan Anda memahami kode di atas. Lalu jalankan fungsi tersebut.

d. Kemudian tambahkan kode untuk melakukan scroll daftar ulasan sehingga akan lebih banyak ulasan yang terkumpul.