

Web Scraping Project: U.S. Infusion Centers Locator

Overview

This project focuses on web scraping healthcare facility data from the [Infusion Center Locator](#), a public-facing directory that lists infusion centers across the United States. Infusion centers provide essential outpatient services for patients with autoimmune diseases, cancer, and chronic illnesses.

The goal of this project is to automate the extraction of all center names and their physical addresses into a structured format (Excel/CSV), enabling future use in analytics, mapping, and healthcare accessibility research.

Objectives

- Scrape the full list of infusion centers from the locator website.
- Extract the **center name** and **address** for each location.
- Save the collected data into an Excel file for easy access and analysis.
- Build a clean, modular scraping solution suitable for a data science portfolio.

Tools and Technologies

- **Python** – For scripting and automation
- **Selenium** – To automate interaction with the website and extract data
- **Pandas** – To store, manipulate, and export the collected data
- **Jupyter Notebook** – For documenting the entire process interactively

Output

A downloadable Excel file named `infusion_centers_all.xlsx` containing:

- Center Name
- Address


Why This Project?

- Demonstrates real-world web scraping skills
- Showcases automation of dynamic web content using Selenium
- Builds a portfolio-worthy dataset in the healthcare domain
- Provides potential use cases in public health analysis and business intelligence

Section 2: Tools and Library Imports

In this project, we will use the following Python libraries:

- **Selenium**: Automates browser interactions (used to navigate the website, click buttons, and scrape dynamic content).
- **Pandas**: Used for organizing the scraped data into a structured DataFrame and exporting it to Excel.
- **Time**: Adds delays to allow dynamic content to load before scraping.

 Make sure these libraries are installed before running the code.

Installation (if not already installed)

You can install these libraries using pip:

```
pip install selenium pandas openpyxl
```

```
In [36]: # 📦 Import necessary Libraries
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

import pandas as pd
import time
```

Section 3: Setting Up Selenium WebDriver (with webdriver-manager)

Instead of manually downloading and managing the ChromeDriver, we'll use `webdriver-manager` to automatically fetch the correct version. This makes the setup smoother and platform-independent.

```
In [37]: # 🦄 Set up headless Chrome WebDriver
chrome_options = Options()
chrome_options.add_argument("--headless")           # Run browser in headless mode (no GUI)
chrome_options.add_argument("--disable-gpu")        # Disable GPU rendering
chrome_options.add_argument("--no-sandbox")         # Required for some Linux environments

# Launch the browser
driver = webdriver.Chrome(options=chrome_options)

# Confirm it's working
print("✅ Chrome WebDriver launched successfully.")
```

✅ Chrome WebDriver launched successfully.

```
In [38]: # 📦 Install webdriver-manager if not already installed
!pip install selenium webdriver-manager --quiet
```

```
In [39]: # 🦄 Set up WebDriver using webdriver-manager
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
import pandas as pd
import time

# Configure Chrome to run headless
chrome_options = Options()
#chrome_options.add_argument("--headless")
chrome_options.add_argument("--disable-gpu")
chrome_options.add_argument("--no-sandbox")

# Initialize driver using ChromeDriverManager
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=chrome_options)

print("✅ Chrome WebDriver launched successfully (via webdriver-manager).")
```

✅ Chrome WebDriver launched successfully (via webdriver-manager).

Section 4: Load the Website and Trigger the Search

Now that the browser is running, we'll:

1. Navigate to the Infusion Center Locator website.
2. Wait for the page to fully load.
3. Click the **Search** button to display all available centers across the U.S.

We use `time.sleep()` to ensure the page loads properly before we interact with elements.

```
In [41]: from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# 🌐 Reload the site
driver.get("https://locator.infusioncenter.org/")
time.sleep(3)

try:
    # 🎨 Wait until the backdrop disappears
    WebDriverWait(driver, 20).until(
        EC.invisibility_of_element_located((By.CLASS_NAME, "MuiBackdrop-root"))
    )

    # 🟢 Wait until the 'Search' button is clickable and enabled
    search_button = WebDriverWait(driver, 20).until(
        EC.element_to_be_clickable((By.XPATH, "//button[contains(text(), 'Search')]"))
    )

    # ✅ Click the button
    search_button.click()
    print("✅ 'Search' button clicked successfully.")
    time.sleep(10) # Wait for results to load

except Exception as e:
    print("❌ Error:", e)
```

✅ 'Search' button clicked successfully.

Section 5: Scroll to Load All Results

Since the website loads more centers as you scroll, we automate scrolling to the bottom repeatedly until no new centers load.

This ensures we collect the full list before extraction.

```
In [44]: # Scroll to load all centers dynamically
prev_count = -1
while True:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(3)

    containers = driver.find_elements(By.CSS_SELECTOR, "div.locator-result-container")
    current_count = len(containers)
    print(f"Centers loaded: {current_count}")

    if current_count == prev_count:
        print("✅ All centers loaded.")
        break
    prev_count = current_count

# Extract name and address
data = []
for idx, container in enumerate(containers):
    try:
        # Use WebDriverWait to ensure elements are present inside this container
        WebDriverWait(container, 2).until(
            EC.presence_of_element_located((By.CLASS_NAME, "locator-result-name"))
        )

        # Try to get the name from either desktop or mobile view
        name = ""
        try:
            name = container.find_element(
                By.XPATH, ".*//div[contains(@class, 'hidden') and contains(@class, 'sm:block')]/div[contains(@class, 'locator-"]
            ).text.strip()
        except:
            try:
                name = container.find_element(
                    By.XPATH, ".*//div[contains(@class, 'sm:hidden')]/div[contains(@class, 'locator-result-name')]"
                ).text.strip()
            except:
                name = ""

        # Extract address
        address = ""
        try:
            address_div = container.find_element(By.CLASS_NAME, "locator-result-address")
            address_lines = address_div.find_elements(By.TAG_NAME, "div")
            street = address_lines[0].text.strip() if len(address_lines) > 0 else ""
            city_state = address_lines[1].text.strip() if len(address_lines) > 1 else ""
            address = f"{street}, {city_state}" if street and city_state else ""
        except:
            address = ""

        if name and address:
            data.append({"Center Name": name, "Address": address})

        else:
            print(f"⚠️ Skipped at index {idx} | Name: {name} | Address: {address}")

    except Exception as e:
        print(f"❌ Error at index {idx}: {e}")

# Convert and preview
import pandas as pd
df = pd.DataFrame(data)
print("\n📄 Preview of Top 5:")
print(df.head())
print(f"\n📦 Total Valid Centers Extracted: {len(df)}")
```

Centers loaded: 960

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

⚠ Skipped at index 933 | Name: | Address:
⚠ Skipped at index 934 | Name: | Address:
⚠ Skipped at index 935 | Name: | Address:
⚠ Skipped at index 936 | Name: | Address:
⚠ Skipped at index 937 | Name: | Address:
⚠ Skipped at index 938 | Name: | Address:
⚠ Skipped at index 939 | Name: | Address:
⚠ Skipped at index 940 | Name: | Address:
⚠ Skipped at index 941 | Name: | Address:
⚠ Skipped at index 942 | Name: | Address:
⚠ Skipped at index 943 | Name: | Address:
⚠ Skipped at index 944 | Name: | Address:
⚠ Skipped at index 945 | Name: | Address:
⚠ Skipped at index 946 | Name: | Address:
⚠ Skipped at index 947 | Name: | Address:
⚠ Skipped at index 948 | Name: | Address:
⚠ Skipped at index 949 | Name: | Address:
⚠ Skipped at index 950 | Name: | Address:
⚠ Skipped at index 951 | Name: | Address:
⚠ Skipped at index 952 | Name: | Address:
⚠ Skipped at index 953 | Name: | Address:
⚠ Skipped at index 954 | Name: | Address:
⚠ Skipped at index 955 | Name: | Address:
⚠ Skipped at index 956 | Name: | Address:
⚠ Skipped at index 957 | Name: | Address:
⚠ Skipped at index 958 | Name: | Address:
⚠ Skipped at index 959 | Name: | Address:

📊 Preview of Top 5:

	Center Name \
0	Oasis Family Health NP RN
1	Integrative Rheumatology of Westchester Dr. ...
2	Agile Infusion Services LLC - Hackensack, NJ
3	Thrivewell Infusion - Tarrytown/Elmsford
4	VIVO Infusion - Valhalla, NY

	Address
0	11 Medical Park Dr, Pomona, NY
1	838 Pelhamdale Ave, New Rochelle, NY
2	5 Summit Ave, Hackensack, NJ
3	555 Taxter Rd, Elmsford, NY
4	400 Columbus Ave, Valhalla, NY

📦 Total Valid Centers Extracted: 240

```
In [45]: df.head()
```

Out[45]:

	Center Name	Address
0	Oasis Family Health NP RN	11 Medical Park Dr, Pomona, NY
1	Integrative Rheumatology of Westchester Dr. ...	838 Pelhamdale Ave, New Rochelle, NY
2	Agile Infusion Services LLC - Hackensack, NJ	5 Summit Ave, Hackensack, NJ
3	Thrivewell Infusion - Tarrytown/Elmsford	555 Taxter Rd, Elmsford, NY
4	VIVO Infusion - Valhalla, NY	400 Columbus Ave, Valhalla, NY

```
In [46]: df.shape
```

Out[46]: (240, 2)

```
In [47]: df.to_csv("infusion_centers.csv")
```

```
In [ ]:
```