Python Web Scraping with Selenium

Peter Njoroge Mwangi

September 19, 2024

1 Introduction

This document explains how to use Python for web scraping using Selenium. We extract football match data (date, home team, score, and away team) from a website and store the data in a CSV file for analysis. The target website is https://www.adamchoi.co.uk/overs/detailedAdam Choi's football stats page.

2 Required Libraries

We use the following libraries for this project:

- **Selenium**: For web browser automation.
- **pandas**: For data manipulation and saving the scraped data.
- **webdriver_manager**: For automatically handling ChromeDriver installation.

Install these libraries using:

```
pip install selenium pandas webdriver_manager
```

3 Code Breakdown

3.1 Importing Required Modules

We start by importing the necessary modules for Selenium to control the browser and pandas for data processing.

```python
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from webdriver_manager.chrome import ChromeDriverManager
```

3.2 Setting up the WebDriver

Here, we use `ChromeDriverManager` to handle ChromeDriver installation. This simplifies the process of setting up a compatible driver for Selenium.

```python
driver = webdriver.Chrome(service=Service(ChromeDriverManager
    ().install()))
```

3.3 Navigating to the Website

We use Selenium's `get()` function to open the desired website.

```python
website = 'https://www.adamchoi.co.uk/overs/detailed'
driver.get(website)
```

3.4 Interacting with the Website

We locate and click the "All matches" button and select the 2024/2025 football season from a dropdown menu using XPath.

```python
all_matches_button = driver.find_element(By.XPATH, "//label[
    contains(text(), 'All matches')]")
all_matches_button.click()

try:
    season_dropdown = driver.find_element(By.XPATH, "//select
    [@id='season']")
    select = Select(season_dropdown)
    select.select_by_visible_text('2024/2025')
except Exception as e:
    print(f"Error selecting season: {e}")
```

3.5 Scraping Match Data

We use the following code to locate table rows (<tr>) and extract relevant details for each match.

```
matches = driver.find_elements(By.TAG_NAME, 'tr')

date = []
home_team = []
score = []
away_team = []

for match in matches:
    try:
        cells = match.find_elements(By.TAG_NAME, 'td')
        if len(cells) == 4:
            date.append(cells[0].text)
            home_team.append(cells[1].text)
            score.append(cells[2].text)
            away_team.append(cells[3].text)
    except Exception as e:
        print(f"Error processing row: {e}")
```

3.6 Cleaning and Saving the Data

Here, we clean the score column by replacing dashes with colons to avoid Excel misinterpreting them as dates, and save the data to a CSV file.

```
df = pd.DataFrame({
    'Date': date,
    'home_team': home_team,
    'score': score,
    'away_team': away_team
})

df['score'] = df['score'].apply(lambda x: x.replace('-', ':')
    )
df.to_csv('Football_data.csv', index=False)
```

3.7 Closing the Browser

Finally, we close the browser after the scraping process is complete.

```
driver.quit()
```

4 Additional Information

- **Error Handling**: We use `try-except` blocks to manage potential issues while interacting with the webpage or processing rows.
- **XPath and CSS Selectors**: XPath is used to target elements. In case the website's structure changes, update the XPaths accordingly.
- **Score Formatting**: By replacing dashes with colons in the score column, we prevent Excel from interpreting certain scores as dates.

This script can be extended to scrape additional data or adapted for use on other websites.

5 Full Code

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from webdriver_manager.chrome import ChromeDriverManager

# Define the website to scrape
website = 'https://www.adamchoi.co.uk/overs/detailed'

# Use ChromeDriverManager to automatically download and
    manage the correct ChromeDriver version
driver = webdriver.Chrome(service=Service(ChromeDriverManager
    ().install()))

# Open the website
driver.get(website)

# Click the "All matches" button to display all match data
all_matches_button = driver.find_element(By.XPATH, "//label[
    contains(text(), 'All matches')]")
all_matches_button.click()

# Select the 2024/2025 season from the dropdown
try:
    # Locate the season dropdown
```

```
    season_dropdown = driver.find_element(By.XPATH, "//select[@id='season']")

    # Create a Select object for interacting with the dropdown
    select = Select(season_dropdown)

    # Select the '2024/2025' season by its visible text
    select.select_by_visible_text('2024/2025')

except Exception as e:
    print(f"Error selecting season: {e}")

# Find all match rows by the 'tr' tag name
matches = driver.find_elements(By.TAG_NAME, 'tr')

# Initialize lists to hold the extracted data
date = []
home_team = []
score = []
away_team = []

# Loop through the table rows to extract match data
for match in matches:
    try:
        # Find all 'td' elements (columns) in the current row
        cells = match.find_elements(By.TAG_NAME, 'td')

        # Ensure that the row has exactly 4 columns (date, home team, score, away team)
        if len(cells) == 4:
            # Append each piece of data to its corresponding list
            date.append(cells[0].text)
            home_team.append(cells[1].text)
            score.append(cells[2].text)
            away_team.append(cells[3].text)

            # Print each row for debugging
            print(f"Date: {cells[0].text}, Home: {cells[1].text}, Score: {cells[2].text}, Away: {cells[3].text}")

    except Exception as e:
        print(f"Error processing row: {e}")

```

```
# Close the browser after extracting the data
driver.quit()

# Create a pandas DataFrame from the extracted data
df = pd.DataFrame({
    'Date': date,
    'home_team': home_team,
    'score': score,
    'away_team': away_team
})

# Replace hyphens in the 'score' column to avoid Excel interpreting scores as dates
df['score'] = df['score'].apply(lambda x: x.replace('-', ':'))

# Save the DataFrame to a CSV file
df.to_csv('Football_data.csv', index=False)

# Print the DataFrame
print(df)
```