Daraz Automation & Scrapping Bot Documentation

Table of Contents

1. Introduction

The Daraz Automation Bot is a Python script built using Selenium, a web automation library. It performs web scraping on the Daraz website, specifically targeting laptop data. The bot uses a Chrome WebDriver to automate browsing actions such as navigating to specific pages, extracting laptop names, prices, URLs, ratings, and reviews. It also stores the scraped data in a MySQL database. The script is scheduled to run on a specific day and time using the schedule library. Overall, the bot enables automated data extraction from Daraz, facilitating efficient analysis and processing of laptop information.

2. Installation

2.1 Prerequisites

Installing Selenium

Step-by-step guide on installing the Selenium library using pip (Python package manager). For example:

1. Open a command prompt/terminal.
2. Run the following command to install Selenium:
   pip install selenium
3. Verify the installation by checking the Selenium version:
   python -c "import selenium; print(selenium.__version__)"

Setting up the WebDriver

Detailed instructions on setting up the WebDriver for your preferred browser (e.g., Chrome, Firefox):

- Downloading the WebDriver executable for your browser and operating system. For example:
- For Chrome: Visit the ChromeDriver download page (https://sites.google.com/a/chromium.org/chromedriver/downloads) and download the appropriate version for your Chrome browser.
- For Firefox: Visit the GeckoDriver download page (https://github.com/mozilla/geckodriver/releases) and download the appropriate version for your Firefox browser.
- Place the WebDriver executable in a directory accessible by your system's PATH environment variable.
- Verify the WebDriver setup by running a simple test script. For example

from selenium import webdriver

driver = webdriver.Chrome() # Replace with appropriate WebDriver class (e.g., Firefox)

driver.quit()

3. Configuration

3.1 WebDriver Configuration

WebDriver allows configuration options such as:

- Setting the browser window size
- Enabling or disabling JavaScript execution
- Managing timeouts and intervals

3.2 Chrome Options

Chrome Options allow customization of ChromeDriver's behavior, such as:

- Running in headless mode (without opening a visible browser window)
- Setting proxy configurations
- Modifying user agent string

4. Usage

4.1 Basic Usage

The usage of this bot is to automate the process of scraping laptop data from the Daraz website. By running the bot, you can extract information such as laptop names, prices, URLs, ratings, and reviews from multiple pages of the Daraz website. This automation saves time and effort compared to manually visiting each page and collecting the data. The scraped data can be stored in a database for further analysis, price comparison, or any other desired use. The bot provides a convenient and efficient way to gather laptop information from Daraz, enabling you to make informed decisions or conduct research based on the extracted data.

- Import the necessary Selenium modules:

  from selenium import webdriver
  from selenium.webdriver.common.by import By
  from selenium.webdriver.common.keys import Keys

- Open a web page:
  driver.get("https://www.daraz.pk")

- Close the browser:
  driver.quit()

4.2 Interacting with Web Elements

- Locating elements by different methods (e.g., ID, class name, XPath).

  element = driver.find_element(By.ID, "ElementId").

5. Conclusion

The provided bot is a Python-based web scraping tool that utilizes Selenium to automate data extraction from the Daraz website. Its main objective is to collect laptop details such as names, prices, URLs, ratings, and reviews. By leveraging Selenium's WebDriver, the bot navigates web pages, locates specific elements, and stores the extracted data in a MySQL database. With scheduled execution using the schedule library, it enables regular and efficient scraping without manual intervention. Overall, the bot offers a streamlined solution for extracting laptop information from Daraz, facilitating data-driven decision-making, market research, and analysis.

6. References

Official Selenium Documentation

Selenium Python Documentation

Selenium WebDriver API Documentation

WebDriver ChromeDriver Documentation

WebDriver GeckoDriver Documentation