Web Browser Automation with Selenium

Selenium is a powerful tool for automating web browsers. It allows you to programmatically control a browser to perform actions such as navigating to web pages, filling out forms, clicking buttons, and extracting data. This can be useful for a variety of tasks, including web scraping, testing web applications, and automating repetitive tasks.

Here's a basic example of how to use Selenium with Python to scrape a CSDN blog:

~~~
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException
import time

def scrape_csdn_blog(url):
    """
    Scrapes a



```python
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException
import time

def scrape_csdn_blog(url):
    """
    Scrapes a CSDN blog and extracts all the links (a tags) from the page source using Selenium,
    filtering for links that start with "https://blog.csdn.net/lzw_java/article".

    Args:
        url (str): The URL of the CSDN blog.
    """
    try:
        # Set up Chrome options for headless browsing
        chrome_options = Options()
        chrome_options.add_argument("--headless")  # Run Chrome in headless mode
        chrome_options.add_argument("--disable-gpu")  # Disable GPU acceleration (recommended for headless)
        chrome_options.add_argument("--no-sandbox")  # Bypass OS security model
~~~

```
        chrome_options.add_argument("--disable-dev-shm-usage")  # overcome limited resource problems

        # Initialize the Chrome driver
        driver = webdriver.Chrome(options=chrome_options)

        # Load the webpage
        driver.get(url)

        # Find all 'a' tag elements
        links = driver.find_elements(By.TAG_NAME, 'a')

        if not links:
            print("No links found on the page.")
            driver.quit()
            return

        for link in links:
            try:
                href = link.get_attribute('href')
                if href and href.startswith("https://blog.csdn.net/lzw_java/article"):
                    text = link.text.strip()

                    print(f"Text: {text}")
                    print(f"URL: {href}")
                    print("-" * 20)

            except Exception as e:
                print(f"Error extracting link: {e}")
                continue

    except Exception as e:
        print(f"An error occurred: {e}")
    finally:
        # Close the browser
        if 'driver' in locals():
            driver.quit()

if __name__ == "__main__":
    blog_url = "https://blog.csdn.net/lzw_java?type=blog"  # Replace with the actual URL
    scrape_csdn_blog(blog_url)
```