Webbrowser-Automatisierung mit Selenium

Selenium ist ein leistungsstarkes Werkzeug zur Automatisierung von Webbrowsern. Es ermöglicht die programmgesteuerte Steuerung eines Browsers, um Aktionen wie das Navigieren zu Webseiten, das Ausfüllen von Formularen, das Klicken auf Schaltflächen und das Extrahieren von Daten durchzuführen. Dies kann für eine Vielzahl von Aufgaben nützlich sein, darunter Web Scraping, das Testen von Webanwendungen und die Automatisierung sich wiederholender Aufgaben.

Hier ist ein grundlegendes Beispiel für die Verwendung von Selenium mit Python zum Scrapen eines CSDN-Blogs:

```
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):
    """
    Scrapt einen



```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):
    """
    Scrapt einen CSDN-Blog und extrahiert alle Links (a-Tags) aus dem Seitenquelltext mithilfe von Selenium.
    Filtert nach Links, die mit "https://blog.csdn.net/lzw_java/article" beginnen.

    Args:
        url (str): Die URL des CSDN-Blogs.
    """
    try:
        # Chrome-Optionen für Headless-Browsing konfigurieren
        chrome_options = Options()
        chrome_options.add_argument("--headless")  # Chrome im Headless-Modus ausführen
```
```

```
        chrome_options.add_argument("--disable-gpu")  # GPU-Beschleunigung deaktivieren (empfohlen für Headles
        chrome_options.add_argument("--no-sandbox")  # OS-Sicherheitsmodell umgehen
        chrome_options.add_argument("--disable-dev-shm-usage")  # Behebung von Problemen mit begrenzten Resso

        # Chrome-Treiber initialisieren
        driver = webdriver.Chrome(options=chrome_options)

        # Webseite laden
        driver.get(url)

        # Alle 'a'-Tag-Elemente finden
        links = driver.find_elements(By.TAG_NAME, 'a')

        if not links:
            print("Keine Links auf der Seite gefunden.")
            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"Fehler beim Extrahieren des Links: {e}")
                continue

    except Exception as e:
        print(f"Ein Fehler ist aufgetreten: {e}")
    finally:
        # Browser schließen
        if 'driver' in locals():
            driver.quit()

if __name__ == "__main__":
```

```
blog_url = "https://blog.csdn.net/lzw_java?type=blog"  # Ersetzen Sie dies durch die tatsächliche URL
scrape_csdn_blog(blog_url)
```