Selenium 浏览器自动化

Selenium 是一个强大的网页浏览器自动化工具。它允许你以编程方式控制浏览器执行各种操作，例如导航到网页、填写表单、点击按钮和提取数据。这对于各种任务都非常有用，包括网页抓取、测试网页应用程序和自动化重复性任务。

下面是一个使用 Selenium 和 Python 抓取 CSDN 博客的基本示例：

```
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):
    """
    抓取 CSDN 博客并使用 Selenium 提取页面源代码中的所有链接（a 标签），
    过滤掉不以"https://blog.csdn.net/lzw_java/article" 开头的链接。

    Args:
        url (str): CSDN 博客的 URL。
    """
    try:
        # 设置 Chrome 选项为无头模式浏览
        chrome_options = Options()
        chrome_options.add_argument("--headless")  # 以无头模式运行 Chrome
        chrome_options.add_argument("--disable-gpu")  # 禁用 GPU 加速（推荐用于无头模式）
        chrome_options.add_argument("--no-sandbox")  # 绕过操作系统安全模型
        chrome_options.add_argument("--disable-dev-shm-usage")  # 克服资源受限问题

        # 初始化 Chrome 驱动
        driver = webdriver.Chrome(options=chrome_options)

        # 加载网页
        driver.get(url)

        # 查找所有'a'标签元素
        links = driver.find_elements(By.TAG_NAME, 'a')

        if not links:
            print(" 页面上没有找到链接。")
```

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

            except Exception as e:
                print(f" 提取链接时出错: {e}")
                continue

    except Exception as e:
        print(f" 发生错误: {e}")
    finally:
        # 关闭浏览器
        if 'driver' in locals():
            driver.quit()

if __name__ == "__main__":
    blog_url = "https://blog.csdn.net/lzw_java?type=blog"  # 请替换为实际的 URL
    scrape_csdn_blog(blog_url)
```