

以 Selenium 自動化網頁瀏覽器

Selenium 是一個功能強大的網頁瀏覽器自動化工具。它允許您以程式設計方式控制瀏覽器執行操作，例如導覽至網頁、填寫表單、點擊按鈕和提取數據。這對於各種任務都非常有用，包括網頁抓取、測試網頁應用程式和自動化重複性任務。

以下是如何使用 Selenium 與 Python 抓取 CSDN 部落格的基本範例：

```
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
```

```
```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):
```

```
 """
```

使用 Selenium 抓取 CSDN 部落格並提取頁面原始碼中的所有連結 (a 標籤)，過濾以 "[https://blog.csdn.net/lzw\\_java/article](https://blog.csdn.net/lzw_java/article)" 開頭的連結。

Args:

url (`str`): CSDN 部落格的網址。

```
"""
```

```
try:
```

# 設定 Chrome 選項為無頭瀏覽

`chrome_options = Options()`

`chrome_options.add_argument("--headless") # 以無頭模式運行 Chrome`

`chrome_options.add_argument("--disable-gpu") # 關閉 GPU 加速 (無頭模式建議)`

`chrome_options.add_argument("--no-sandbox") # 繞過作業系統安全模型`

`chrome_options.add_argument("--disable-dev-shm-usage") # 克服資源受限問題`

```

初始化 Chrome 驅動程式
driver = webdriver.Chrome(options=chrome_options)

載入網頁
driver.get(url)

尋找所有 'a' 標籤元素
links = driver.find_elements(By.TAG_NAME, 'a')

if not links:
 print(" 網頁上沒有找到任何連結。")
 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" 文字: {text}")
 print(f" 網址: {href}")
 print("-" * 20)

 except Exception as e:
 print(f" 提取連結時發生錯誤: {e}")
 continue

except Exception as e:
 print(f" 發生錯誤: {e}")

finally:
 # 關閉瀏覽器
 if 'driver' in locals():
 driver.quit()

if __name__ == "__main__":
 blog_url = "https://blog.csdn.net/lzw_java?type=blog" # 請替換為實際網址
 scrape_csdn_blog(blog_url)

```