臺灣銀行匯率爬蟲

一.前置作業

1. 預先安裝:

```
pip install selenium pandas concurrent-log-handler
```

2.下載 Microsoft Edge WebDriver:

前往官方下載頁面: https://developer.microsoft.com/zh-tw/microsoft-edge/tools/webdriver/
確認您當前使用的 Microsoft Edge 版本:

1.打開 Edge 瀏覽器
2.點擊右上角三點選單
3.選擇「設定」
4.選擇「關於 Microsoft Edge」
5.查看版本號

下載與瀏覽器版本相符的 WebDriver

3. 驅動程式配置:

```
# Windows
set EDGE_DRIVER_PATH=C:\python\msedgedriver.exe

# Linux/macOS
export EDGE_DRIVER_PATH=~/bin/msedgedriver
```

4. 執行指令:

```
python currency_scraper.py --max_workers 8 --retries 3
```

二.程式說明

1. 模組導入

```python
import time
import random
import logging
import threading
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.edge.service import Service
from selenium.common.exceptions import TimeoutException, StaleElementReferenceException
from concurrent.futures import ThreadPoolExecutor, as_completed
from datetime import datetime
```

2. 日誌設定

```python
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler(f'currency_scraper_{datetime.now().strftime("%Y%m%d_%H%M%S")}.log', en
        logging.StreamHandler()
    ]
)
logger = logging.getLogger(__name__)
```

3. 全域變數

```python
info_list = []
info_lock = threading.Lock()
DRIVER_PATH = "C:/python/msedgedriver.exe"
```

4. 初始化 WebDriver

```python
def init_driver():
    options = webdriver.EdgeOptions()
    options.add_argument('--start-maximized')
    options.add_argument('--disable-notifications')
    options.add_argument('--disable-blink-features=AutomationControlled')
    options.add_experimental_option('excludeSwitches', ['enable-logging'])
    service = Service(DRIVER_PATH)
    return webdriver.Edge(service=service, options=options)
```

5.抓取單個幣別的匯率

```
def get_exchange_rate(index, max_retries=3):
    for attempt in range(max_retries):
        driver = None
        try:
            logger.info(f"開始處理第 {index + 1} 個幣別 (第 {attempt + 1} 次嘗試)")
            driver = init_driver()

            # 隨機化操作間隔
            time.sleep(2 + random.random()*3)

            # 主頁面操作
            driver.get("https://rate.bot.com.tw/xrt")
            history_buttons = WebDriverWait(driver, 10).until(
                EC.presence_of_all_elements_located(
                    (By.XPATH, '//a[contains(text(),"歷史查詢")]'))
            )

            # 分頁處理
            history_buttons[index].click()
            WebDriverWait(driver, 10).until(
                lambda d: len(d.window_handles) > 1)
            driver.switch_to.window(driver.window_handles[-1])

            # 數據解析
            WebDriverWait(driver, 10).until(
                EC.element_to_be_clickable((By.ID, "single"))).click()

            currency_element = WebDriverWait(driver, 10).until(
                EC.presence_of_element_located(
                    (By.CSS_SELECTOR, "div.visible-phone.print_hide h2")))

            # 結構化存儲
            rows = driver.find_elements(
                By.XPATH, '//table[@class="table"]/tbody/tr')

            local_data = []
            for row in rows:
                try:
                    cells = row.find_elements(By.TAG_NAME, "td")
```

```python
                    if len(cells) >= 4:
                        local_data.append({
                            "幣別": currency_element.text.split()[0],
                            "日期": cells[0].text,
                            "現金買入": cells[2].text,
                            "現金賣出": cells[3].text,
                            "抓取時間": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                        })
                except Exception as e:
                    logger.warning(f"數據解析異常: {str(e)}")

            # 線程安全寫入
            with info_lock:
                info_list.extend(local_data)

            return True

        except Exception as e:
            logger.error(f"處理異常: {str(e)}")
        finally:
            if driver:
                driver.quit()
    return False
```

6. 將結果保存到文件

```
def save_to_file():
    try:
        filename = f"exchange_rates_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"
        with open(filename, 'w', encoding='utf-8') as f:
            for info in info_list:
                f.write(f"幣別: {info['幣別']}\n")
                f.write(f"更新時間: {info['更新時間']}\n")
                f.write(f"買入價: {info['買入價']}\n")
                f.write(f"賣出價: {info['賣出價']}\n")
                f.write(f"抓取時間: {info['抓取時間']}\n")
                f.write("-" * 50 + "\n")
        logger.info(f"數據已保存到文件: {filename}")
    except Exception as e:
        logger.error(f"保存文件時發生錯誤: {str(e)}")
```

7. 主程序

```python
def main():
    try:
        # 初始化環境
        driver = init_driver()
        driver.get("https://rate.bot.com.tw/xrt")

        # 動態獲取幣別數量
        history_buttons = WebDriverWait(driver, 10).until(
            EC.presence_of_all_elements_located(
                (By.XPATH, '//a[contains(text(),"歷史查詢")]'))
        total = len(history_buttons)
        driver.quit()

        logger.info(f"總需處理幣別數量: {total}")

        # 並行處理
        with ThreadPoolExecutor(max_workers=8) as executor:
            futures = {executor.submit(get_exchange_rate, i): i for i in range(total)}

            for future in as_completed(futures):
                idx = futures[future]
                try:
                    if future.result():
                        logger.success(f"幣別 {idx+1} 完成")
                except Exception as e:
                    logger.critical(f"嚴重錯誤: {str(e)}")

        # 數據輸出
        if info_list:
            df = pd.DataFrame(info_list)
            csv_name = f"匯率數據_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
            df.to_csv(csv_name, index=False, encoding='utf-8-sig')
            logger.info(f"CSV 文件已保存: {csv_name}")

    except Exception as e:
        logger.error(f"主程序錯誤: {str(e)}")
```

8. 程式入口與 CSV 輸出

```
if __name__ == "__main__":
    main()

# 將結果輸出為 CSV
df = pd.DataFrame(info_list)
filename = f"exchange_rates_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
df.to_csv(filename, index=False, encoding='utf-8-sig')
```

效能優化參數

```
# 最佳化參數配置建議
OPTIMIZATION_CONFIG = {
    "max_workers": 6,           # 根據CPU核心數調整
    "retry_delay_range": [1, 3],# 隨機重試間隔
    "timeout_threshold": 15,    # 單一頁面超時限制
    "max_retries": 3,           # 最大重試次數
    "random_delay": True,       # 啟用隨機延遲
    "headless_mode": False      # 無頭模式切換
}
```

故障排除指南

```
常見錯誤處理流程:

if "element not found":
    1. 檢查XPath/CSS選擇器是否過期
    2. 增加WebDriverWait時間
    3. 驗證網站結構是否變更

elif "stale element reference":
    1. 重新獲取元素
    2. 加入隨機延遲
    3. 使用retrying套件

elif "timeout":
    1. 調整timeout參數
    2. 檢查網路連線
    3. 驗證代理伺服器設置

else:
    查看日誌文件並提交issue
```