sto.cx 思兔閱讀-小說下載程式

關鍵字/套件: pyqt5, qtdesigner, selenium, BeautifulSoup

1 程式架構

主程式: stocx.py
瀏覽器執行緒: BrowserThread.py
搜尋執行緒: SearchThread.py
下載執行緒: DownloadThread.py
圖示: img_handling
介面設計: ui

2 主程式

程式開啟，初始化時:

```
self.checkOs = self.win_or_mac()
if os.path.exists("path.json"):
    with open("path.json", 'r') as f:
        self.path = json.load(f)
else:
    if self.checkOs == 'Windows':
        self.path = "c:\\novel_tmp"
    elif self.checkOs == 'Darwin':
        self.path = "/Users"
    else:
        self.lblStatus.setText("不支援的作業系統, 無法確保正常執行")
```

目的為檢查開啟程式的作業系統為 Windows 或 macOS，有以下兩點影響:

1. 下載目錄的儲存路徑
2. 程式運行時的圖示(icon)顯示

```
self.disable_gui()
# BrowserThread
self.browserThread = BrowserThread(self.path)
self.browserThread.callback.connect(self.browser_thread_callback)
self.browserThread.start()

self.btnSearch.clicked.connect(self.btn_search_clicked)
self.btnDownload.clicked.connect(self.btn_download_clicked)
self.btnPath.clicked.connect(self.btn_path_clicked)
# press Enter to search
self.txtInput.installEventFilter(self)
```

```
def browser_thread_callback(self, browser):
    self.browser = browser
    self.enable_gui()
```

```
def eventFilter(self, source, event):
    if event.type() == QEvent.KeyPress and source is self.txtInput:
        if event.text() == '\r':
            self.btn_search_clicked()
    return super(MainWindow, self).eventFilter(source, event)
```

將可互動的介面元件停用，呼叫瀏覽器執行緒，成功在背景開啟 chrome 後，介面元件始可互動.

回傳瀏覽器的使用權至 browser_thread_callback 函數

定義每個按鈕被按下時要執行的程式碼, installEventFilte 與 eventFilter 的作用是讓使用者輸入完畢後, 按下鍵盤的 Enter 即可查詢

```
def btn_search_clicked(self):
    self.listWidget.clear()
    self.input_text = self.txtInput.text()
    self.search_mode = ''
    if self.input_text == '':
        dialog = QMessageBox()
        dialog.setWindowTitle('思兔閱讀下載')
        dialog.setText("請輸入作者 / 作品")
        dialog.exec()
        return
    if self.rbtnAuthor.isChecked():
        self.search_mode = 'Author'
    elif self.rbtnNovel.isChecked():
        self.search_mode = 'Novel'
    self.lblStatus.setText('搜尋中...')
    self.disable_gui()
    # SearchThread
    self.searchThread = SearchThread(self.browser, self.input_text, self.search_mode)
    self.searchThread.callback.connect(self.search_thread_callback)
    self.searchThread.search_result.connect(self.search_thread_result)
    self.searchThread.start()
```

按下搜尋鍵時, 呼叫搜尋執行緒執行搜尋的程式碼,

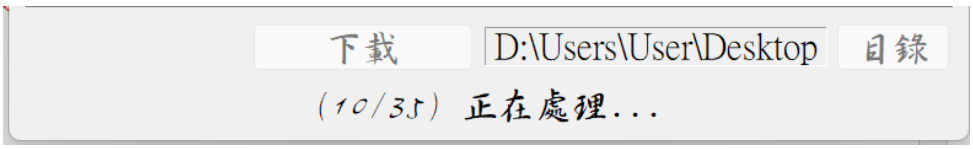
回傳的資料 callback, search_result

分別傳入 search_thread_callback , search_thread_result 函數

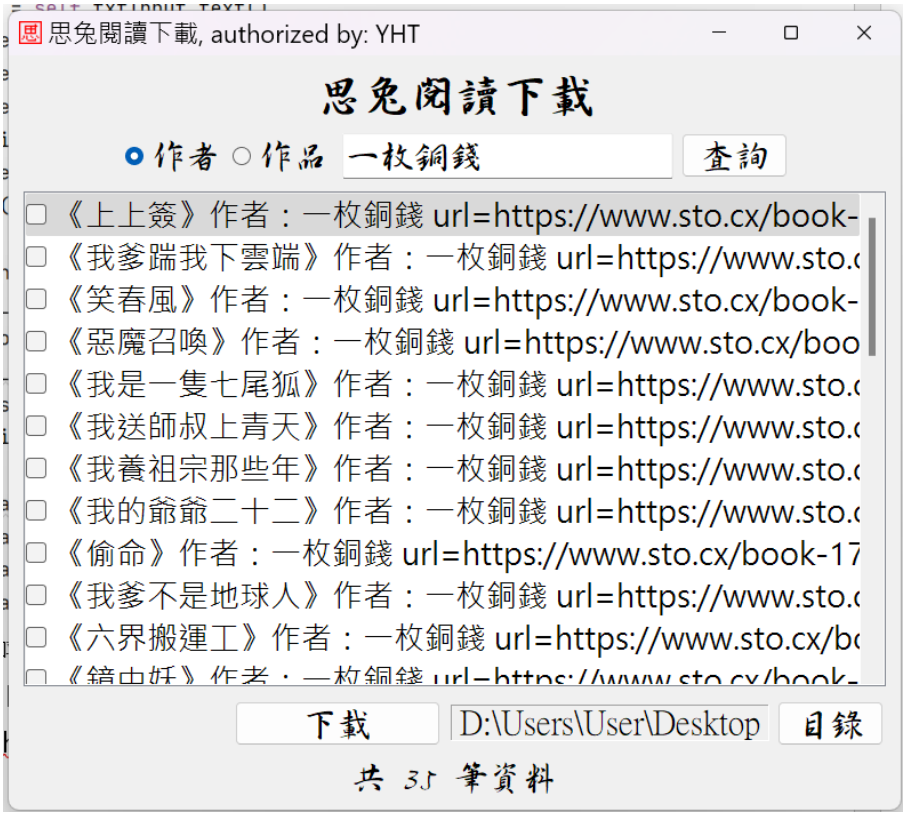
```
def search_thread_result(self, msg):
    self.lblStatus.setText(msg)

T3nG
def search_thread_callback(self, links):
    self.enable_gui()
    self.setStyleSheet('''
        QCheckBox::indicator{
            width: 18px;
            height: 18px;
        }
    ''')
    if links is not None:
        for key in links.keys():
            item = QListWidgetItem()
            self.listWidget.addItem(item)
            box = QCheckBox(links[key])
            self.listWidget.setItemWidget(item, box)
```

search_thread_resul 顯示訊息於視窗下方區域

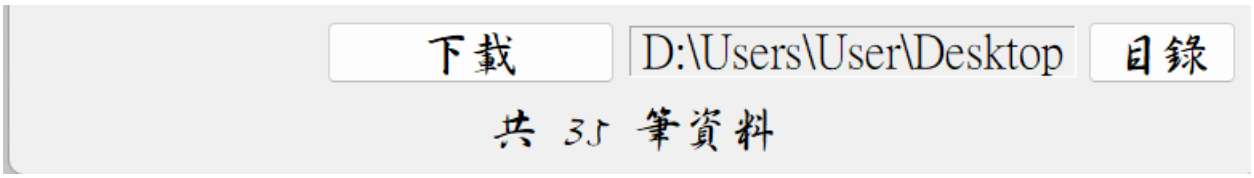


search_thread_callback 於中央區域顯示查詢的結果




```python
def btn_path_clicked(self):
    path = QFileDialog.getExistingDirectory()
    if path != '':
        # if file cannot be saved, try different path slashes
        if self.checkOs == 'Windows':
            self.path = path.replace("/", "\\")
        elif self.checkOs == 'Darwin':
            self.path = path
        with open("path.json", 'w') as f:
            json.dump(self.path, f)
        self.lblPath.setText(self.path)
```

目錄鍵按下後彈出視窗，讓使用者自訂下載目錄的位置，並記錄成 path.json 檔，下次開啟時，自動載入更改後的目錄位置



```python
def btn_download_clicked(self):
    count = self.listWidget.count()
    if count < 1:
        dialog = QMessageBox()
        dialog.setWindowTitle("思兔閱讀下載")
        dialog.setText("請輸入作者 / 作品")
        dialog.exec()
        return
    boxes = [self.listWidget.itemWidget(self.listWidget.item(i)) for i in range(count)]
    checked = []
    for box in boxes:
        if box.isChecked():
            checked.append(box.text())
    self.disable_gui()
    # DownloadThread
    self.downloadThread = DownloadThread(self.browser, checked, self.path)
    self.downloadThread.callback.connect(self.download_thread_callback)
    self.downloadThread.finished.connect(self.download_thread_finished)
    self.downloadThread.start()

def download_thread_callback(self, msg):
    self.lblStatus.setText(msg)

def download_thread_finished(self, msg):
    self.lblStatus.setText(msg)
    self.enable_gui()
```

將所有查詢結果放入 boxes，將選中的選項放入 checked

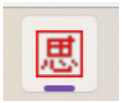
由 callback, finished 來接收回傳的訊息，並顯示訊息於視窗下方區域

正在下載《末日喪屍男友》作者：一枚銅錢，處理中(1/37) https://www.sto.cx//book-77350-1.html ...

```python
if __name__ == '__main__':
    if MainWindow.win_or_mac() == 'Windows':
        myAppId = u'myCompany.myProduct.subProduct.version'  # arbitrary string, unicode
        ctypes.windll.shell32.SetCurrentProcessExplicitAppUserModelID(myAppId)

    app = QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(MainWindow.icon_from_base64(favicon_ico)))
    mainWindow = MainWindow()
    mainWindow.show()
    app.exec()
```

程式進入點 app.exec() 視窗打開後等待指令

使程式能於 Windows 工作列正確顯示圖示 

setWindowIcon 設定圖示於所有視窗

3 瀏覽器執行緒

```
class BrowserThread(QThread):
    callback = pyqtSignal(object)

    T3nG
    def __init__(self, path):
        super().__init__(None)
        self.browser = None
        self.path = path

    T3nG
    def run(self):
        opt = Options()
        opt.add_argument('--headless')
        opt.add_argument('--disable-gpu')
        opt.add_argument('--window-size=1280,720')
        # random header info
        user_agent = UserAgent()
        opt.add_argument('--user-agent=%s' % user_agent.random)
        opt.add_experimental_option("excludeSwitches", ["enable-logging"])
        opt.add_argument('--disable-dev-shm-usage')

        ser = Service(ChromeDriverManager().install())
        browser = webdriver.Chrome(service=ser, options=opt)
        self.callback.emit(browser)
```

headless 使瀏覽器在背景執行

window-size 變更瀏覽器的視窗大小，讓搜尋執行緒能順利爬取網頁資料

user_agent 第三方套件，產生隨機 header，躲避網頁架設的非人為操作偵測機制

4 搜尋執行緒

```
def run(self):
    while True:
        self.browser.get("https://www.sto.cx/pcindex.aspx")
        if self.browser.current_url != "https://www.sto.cx/pcindex.aspx":
            print(self.browser.current_url)
            time.sleep(random.randint(5, 8) + random.random())
            self.search_result.emit(f'無法順利進入網站 {self.browser.current_url}, 重試中...')
            return
        else:
            break
```

網站會偵測是以手機或電腦進行瀏覽，使用 ChromeDriver 時，可能會被誤判為手機版，因而抓取不到網頁元素，過一段時間後，重新進入網站

```
try:
    WebDriverWait(self.browser, 20, 5).until(ec.presence_of_element_located((By.TAG_NAME, 'div')))
    self.browser.find_element(by="id", value="key").send_keys(self.input_text)
    if self.search_mode == 'Author':
        self.browser.find_element(by="id", value="sa").click()
    elif self.search_mode == 'Novel':
        self.browser.find_element(by="id", value="sn").click()
except Exception as e:
    print(e)
```

找尋網頁元素，模擬點擊

```
links = {}
while True:
    try:
        WebDriverWait(self.browser, 20, 5).until(ec.presence_of_element_located((By.TAG_NAME, 'div')))
        soup = BeautifulSoup(self.browser.page_source, "lxml")
        results = int(soup.find('span', string=re.compile("^相關結果共|^相关结果共")).text.split()[1])
        if results == 0:
            self.search_result.emit('找不到結果, 請確認是否輸入錯誤')
            break
        # {(href: title), }
        a_tags = soup.find_all('a', href=re.compile("^/book"))
        for a in a_tags:
            url = f'https://www.sto.cx{a["href"]}'
            if self.search_mode == 'Author':
                links.update({url: f'{a["title"]} url={url}'})
            elif self.search_mode == 'Novel':
                links.update({url: f'{a.text} url={url}'})
            self.search_result.emit(f'({len(links)}/{results}) 正在處理...')
        next_page = soup.find('a', string=re.compile("^下一頁|^下壹頁"))
        if next_page.get('disabled'):
            self.search_result.emit(f'共 {len(links)} 筆資料')
            break
        next_page_link = f'https://www.sto.cx/{next_page.get("href")}'
        self.browser.get(next_page_link)
        time.sleep(random.randint(5, 8) + random.random())
    except Exception as e:
        self.search_result.emit('錯誤發生', e)
        break
self.callback.emit(links)
```

處理搜索資料時的問題點，比如網頁中關鍵字偶爾的簡繁轉換

將搜尋完畢的結果以 links 回傳

5 下載執行緒

```
WebDriverWait(self.browser, 20, 5).until(ec.presence_of_element_located((By.TAG_NAME, 'div')))
s = BeautifulSoup(self.browser.page_source, 'lxml')
keywords = s.find('h1').text
get_title = re.search("(?<=《).*(?=》)", keywords).group(0)
get_author = re.search("(?<=作者：).*", keywords).group(0)
# handle illegal characters (when used as parts of file names)
title = get_title.replace("\\", "").replace("/", "_").replace(":", "_").replace("*", "") \
    .replace("?", "").replace("\"", "").replace("<", "").replace(">", "").replace("|", "")
author = get_author.replace("\\", "").replace("/", "_").replace(":", "_").replace("*", "") \
    .replace("?", "").replace("\"", "").replace("<", "").replace(">", "").replace("|", "")
```

將爬取到的作者名稱及小說名稱存為檔案名稱，去除不可為檔案名稱的特殊字元

```
    # put all pages into dict, key: value => page number: url
    pages = {}
    for page in s.find_all('option'):
        pages[page.text] = f'https://www.sto.cx/{page["value"]}'
    # if same file name exist, make new one then add _number before extension name
    filename = self.increment_filename(os.path.join(self.path, f'{title}_by_{author}.txt'))
    count = 0
    for key in pages.keys():
        count += 1
        self.browser.get(pages[key])
        try:
            WebDriverWait(self.browser, 20, 5).until(ec.presence_of_element_located((By.TAG_NAME, 'div')))
            soup = BeautifulSoup(self.browser.page_source, "lxml")
            content = soup.find(id='BookContent')
            word_list = content.get_text("\n", strip=True).splitlines(keepends=True)
            big_word_list = '\n'.join(word_list)
            self.callback.emit(f'正在下載{self.title}, 處理中({count}/{len(pages)}) {pages[key]} ...')
            with open(filename, 'a', encoding="utf-8") as f:
                f.write(big_word_list)
            time.sleep(random.randint(5, 8) + random.random())
        except Exception as e:
            with open('log.txt', 'w', encoding="utf-8") as f:
                f.write(str(e))
            return False
    return True
except Exception as e:
    with open('log.txt', 'w', encoding="utf-8") as f:
        f.write(str(e))
    return False
```

將抓取到的文字稍做處理後，寫入 txt 檔，若錯誤發生，生成 log.txt 以供除錯

```
@staticmethod
def increment_filename(f):
    fnew = f
    root, ext = os.path.splitext(f)
    i = 1
    while os.path.exists(fnew):
        i += 1
        fnew = '%s_%i%s' % (root, i, ext)
    return fnew
```

若檔案名稱已存在，不進行覆蓋，而是自動生成另一個 txt 檔

```python
def run(self):
    for i, chk in enumerate(self.checked):
        self.title = chk.split(' url=')[0]
        url = chk.split(' url=')[1]
        if not self.download(url):
            self.finished.emit('下載時, 錯誤發生')
            break
        else:
            self.callback.emit(f'{self.title} 下載完成({i+1}/{len(self.checked)})...')
            time.sleep(5)
            if i+1 != len(self.checked):
                self.callback.emit('準備開始下載下一個項目...')
            else:
                self.finished.emit('下載完畢')
```

為每一個選中的選項抓取資料，為避免偵測與造成過大流量，下載一個作品需要不少時間，若同時選取了許多選項，可能會花上不少時間

6 UI 與圖示

圖示 icon 透過轉成.py 檔直接嵌在程式中，UI 透過 Qt Designer 所設計，參考吳明學老師的 mp3 下載器版面

7 優化的可能性

尚有不少地方可以做修正與完善，比如在各階段遭遇錯誤時輸出 log 檔案以供除錯，估計並顯示下載完成的時間，測試在 macOS 執行的問題點，以及思考若使用者被封鎖 IP 或被其他偵測手段反制時的替代方案等