國立臺北科技大學

資訊工程系碩士班

碩士論文

編號：108598051

支援多國語言的Robot Framework網頁自動化驗收測試工具的功能改善與擴充

研究生：林稟宸



**資訊工程系碩士班**

**碩士學位論文**

**支援多國語言的Robot Framework網頁自動化驗收測試工具的功能改善與擴充**

**Further Improvement and Extension of an**

**Automated Web Testing Tool**

**with Robot Framework to Support**

**Internationalization**

**研究生：林稟宸**

**指導教授：鄭有進 教授、謝金雲 教授**

**中華民國一百一十年七月**

**摘要**

論文名稱：支援多國語言的Robot Framework網頁自動化驗收測試工具的功能改善與擴充

頁數：

校所別：國立台北科技大學 資訊工程系碩士班

畢業時間：一百零九學年度 第二學期

學位：碩士

研究生：林稟宸

指導教授：鄭有進 教授、謝金雲 教授

相關學術名詞：Robot Framework、xpath、關鍵字(Keyword)、代理關鍵字(Keyword Proxy)、i18n

以往基於Robot Framework的網頁驗收測試，同一份測試腳本的測試對象往往只限於一種語言的網頁，但現今的國際化社會，隨著商業模式的改變，一個網頁可能有多種語言的版本給不同國家的人使用。為了做出類似或相同的測試目的，測試人員需要撰寫更多重複性高的測試腳本，如此一來，測試的成本便會大大增加。根據朱峻平的論文: ”支援多國語言的Robot Framework 網頁自動化驗收測試” (i18n)，透過使用者提供的JSON格式多國語言網頁翻譯檔，建立出一份單字對應翻譯的翻譯路徑檔，並在程式中定義翻譯的邏輯以及代理關鍵字，即可在不改動Robot Framework 測試腳本的情況下，完成多國語言網頁自動化驗收測試。然而，現在的i18n工具仍然存在著許多待改善之處，會造成使用上的困難。分別是: 支援的Robot Framework原生關鍵字只有七種、翻譯邏輯只支援三種HTML屬性、一詞多譯無法讓使用者選擇期望的翻譯、該工具尚未發佈並支援安裝。因此，本論文將針對上述幾點待改善的地方，進行功能上的補強，希望藉此讓i18n工具在未來變得更易於使用。

**ABSTRACT**

Title：Further Improvement and Extension of an Automated Web Testing Tool with Robot Framework to Support Internationalization

Pages：

School：National Taipei University of Technology

Department：Computer Science and Information Engineering

Time：July, 2021

Degree：Master

Researcher：Bing-Chen Lin

Advisor：Yu Chin Cheng Ph.D. and Chin-Yun Hsieh Ph.D.

Related terms：Robot Framework, xpath, Keyword, Keyword Proxy, I18n

In the past, the test target of an acceptance test script with Robot Framework was often a web page with single language. However, nowadays a web page may have multiple language versions for people who live in different countries. Therefore, test engineers often need to write test scripts which are highly repetitive, and the cost of testing would also increase.According to the thesis “Internationalization Support for Automated Web Testing with Robot Framework”(i18n) written by Chun-Ping Chu, i18n tool can build the mapping between words and their translations by JSON format translation files given by user. Besides, through the translation logic and keyword proxies implemented in i18n tool, we can run web tests which have same movements but different languages by a same test script without changing a word.However, the i18n tool still have many flaws to be improved currently; namely, supporting only 7 Robot Framework native keywords, limiting to 3 HTML attributes as translate targets, not supporting user choices to translations of words, not supporting installation by pip.Therefore, I will improve the flaws of the i18n tool mentioned above. Hoping that the i18n tool would be better and more user-friendly in the future.

**誌 謝**

首先感謝父母對我從小到大的栽培，也感謝教授在北科的這兩年，不但教我們程式設計，也讓我從實驗室不輕鬆卻充實的日常中收穫良多。

在論文的研究過程中，感謝發哥提供了我許多寶貴的意見與討論，也特別感謝朱峻平學長，工作之餘抽空指引我論文的研究方向，共同討論如何讓此i18n工具成為一個更好的設計。

不期望可以做出多偉大的軟體，但要做出一個有用的軟體。本論文秉持著這一精神，盡力完善，不好或缺失的地方也在論文限制處如實敘述，希望透過現在的努力與之後的維護，能讓”在不改動腳本下，實現多國語言網頁自動化驗收測試”的i18n工具，在日後成為測試者們的一個選項。

最後，也和自己說聲辛苦了，寫論文不像平時隨手寫一些有趣的專案那般信手拈來，要說過程中完全沒有遭遇挫折，我相信那是騙人的。謝謝自己的堅持，才有了今天的成果，也祝自己早日交到女朋友。

程式雖然不像人一般有感情，但一旦你投入心血，它卻不會背叛你，共勉之。

**目 錄**

[**摘要** i](#_Toc73612491)

[**ABSTRACT** iii](#_Toc73612492)

[**誌 謝** v](#_Toc73612493)

[**目 錄** vi](#_Toc73612494)

[**表目錄** viii](#_Toc73612495)

[**圖目錄** ix](#_Toc73612496)

[**第一章** **緒論** 1](#_Toc73612497)

[**1.1 研究背景與動機** 1](#_Toc73612498)

[**1.2 研究目標** 2](#_Toc73612499)

[**1.3 論文組織架構** 2](#_Toc73612500)

[第二章 背景知識 3](#_Toc73612501)

[**2.1 國際化** 3](#_Toc73612502)

[**2.2 自動化驗收測試** 3](#_Toc73612503)

[**2.3 Robot Framework** 3](#_Toc73612504)

[**2.3.1 Robot Framework測試腳本** 3](#_Toc73612505)

[**2.3.2 Robot Framework 測試報表** 4](#_Toc73612506)

[**2.4 XPath** 4](#_Toc73612507)

[第三章 研究方法與實作 5](#_Toc73612508)

[**3.1 系統設計** 5](#_Toc73612509)

[**3.1.1 第一版i18n工具的現存類別與原生類別** 6](#_Toc73612510)

[**3.1.2 本論文新擴充的類別** 7](#_Toc73612511)

[**3.1.3 系統執行流程** 8](#_Toc73612512)

[**3.2 擴充與修改代理關鍵字使其完整支援原生關鍵字** 11](#_Toc73612513)

[**3.3 改善XPath翻譯邏輯使其能應對各種html屬性** 14](#_Toc73612514)

[**3.4 提供圖形化使用者介面解決一詞多譯的問題** 16](#_Toc73612515)

[**3.5 將i18n工具設計成為可以安裝的模組** 20](#_Toc73612516)

[第四章 測試案例分析 22](#_Toc73612517)

[**4.1 新增與修改之代理關鍵字的單元測試** 22](#_Toc73612518)

[**4.1.1 Alert Should Be Present** 22](#_Toc73612519)

[**4.1.2 Count Values In List** 23](#_Toc73612520)

[**4.1.3 Dictionaries Should Be Equal** 23](#_Toc73612521)

[**4.1.4 Select From List By Label** 24](#_Toc73612522)

[**4.1.5 \*Page Should Contain Element** 25](#_Toc73612523)

[**4.1.6 Table Should Contain** 26](#_Toc73612524)

[**4.2 改善翻譯邏輯後的驗收測試示例** 29](#_Toc73612525)

[**4.3 一詞多譯情況下所產生的圖形化使用者介面** 31](#_Toc73612526)

[**4.4 使用涵蓋多項代理關鍵字的測試腳本** 34](#_Toc73612527)

[第五章 結論與未來展望 36](#_Toc73612528)

[**5.1 結論** 36](#_Toc73612529)

[**5.2 本論文限制與未來展望** 36](#_Toc73612530)

**表目錄**

**圖目錄**

1. **緒論**
   1. **研究背景與動機**

根據朱峻平的論文:“支援多國語言的Robot Framework 網頁自動化驗收測試” (i18n)[3]，透過使用者提供的JSON格式多國語言網頁翻譯檔，建立出一份單字對應翻譯的翻譯路徑檔，並在程式中定義翻譯的邏輯以及代理關鍵字，即可在不改動Robot Framework 測試腳本的情況下，完成多國語言網頁自動化驗收測試。

然而，現在的i18n[5]工具仍然存在著許多待改善之處，例如:

1. 目前i18n工具只支援7種Robot Framework[7]原生關鍵字，如果未來測試腳本使用到其他未支援的Robot Framework原生關鍵字，便會發生錯誤
2. 程式目前的翻譯對象僅限於網頁上的text(), normalize-space(),以及@title三種HTML屬性。屆時，若使用者的測試腳本是用沒有列舉出來的屬性撰寫，例如:@placeholder, @arial-label等等，便會出錯。
3. 測試腳本執行期間，若遭遇一詞多譯的情況，目前i18n工具僅於報表上列出該待翻譯詞的可能翻譯有哪些，並從中選取會使測試通過的翻譯詞當作當前翻譯；然而，使用者無法自己選擇期望的翻譯去跑測試腳本。此外，假如存在多個翻譯詞都會通過此測試(可能是因為XPath[11]使用contains語法，使得畫面上要驗證的資訊只要包含於翻譯詞，測試就會通過；又或者畫面上剛好有多個元件符合翻譯後的測試腳本)，翻譯過後腳本的測試對象，就會偏離了使用者原先的預期。
4. 目前i18n工具尚未包裝成可以讓使用者直接安裝後使用的擴充工具，停留在使用者必須將github上的專案clone下來，並且在該專案上開發測試腳本的階段。如此會造成使用者的許多不便。

有鑑於此，本論文將對現有的i18n工具進行功能的改善與擴充。

* 1. **研究目標**

本論文旨在改善i18n工具上述提及的四項缺陷，希望透過以下努力可以讓此I18n工具在未來執行多國語言網頁自動化驗收測試[[1]](#參考1)[4]時，更易於使用。

實作的目標分別列舉如下:

1. 擴充完剩下的Robot Framework原生關鍵字代理，使得所有常用原生關鍵字的參數部分都能正確的被翻譯。
2. 撰寫出一套新的XPath翻譯邏輯，支援除了@title、normalize-space()、text()以外，其他HTML屬性的翻譯檢查。
3. 執行完含有一詞多譯的測試腳本後，如果測試通過，便開啟一個圖形化介面，記錄了執行翻譯時當下關鍵字的參數組合，並顯示所有可能的翻譯詞，讓使用者可以從中去選擇，並產生一個設定檔。之後再次執行測試腳本時，i18n工具便會根據設定檔的內容去選擇適當的翻譯詞，同時消除報表上的warning提示。
4. 將i18n工具包裝成為可以pip[12] install 的library。
   1. **論文組織架構**

本論文分為五章，第一章介紹研究背景與動機，以及期望達成的研究目標。第二章介紹本論文相關的背景知識。第三章探討本論文的研究方法與實作，詳述擴充代理關鍵字的流程、改善翻譯邏輯的完整思路，以及如何藉由圖形化使用者介面去改善i18n工具目前遇到一詞多譯時的處理方法。第四章展示實際的測試案例，呈現做了上述改動後，使用者執行多國語言網頁自動化驗收測試時，會得到什麼結果，並可自行與第一版i18n工具[3]做比較。第五章則是結論以及未來展望，概述本論文的成果以及尚且存在的使用限制和待改善之處。

# 第二章 背景知識

* 1. **i18n工具相關知識**
     1. **國際化**

國際化(Internationalization)[5]，簡寫為i18n，其中18代表i和n之間的18個英文字母。國際化是開發軟體時，將軟體本身和特定語言、地區脫鉤的一個過程，除了可以滿足不同的地區、文化的大眾需求，移植軟體到不同的語言環境時，也不須改變內部程式的實作。

* + 1. **自動化驗收測試**

驗收測試(Acceptance Testing)[[1]](#參考1)，是一種站在使用者的立場，去檢驗一個真實存在的系統，是否滿足使用者需求與預期的測試方法。

而透過撰寫自動化測試腳本[13]，如今我們可以執行更符合成本，且更加精確的自動化驗收測試。改善了過往手動執行驗收測試時，存在的人為操作誤差、系統問題無法即時呈現、成本較高等問題。

* + 1. **XPath**

XPath[11]，全名為 XML Path Language，可以用來定位XML檔案中某節點所處在的位置。在使用Robot Framework撰寫的網頁自動化驗收測試中，我們便時常需要藉助XPath，去找到並確定畫面上某元件的位置，以對其狀態進行測試。

* + 1. **JSON格式翻譯檔在前端框架中的運作**

JSON格式的多國語言翻譯檔，其結構是由撰寫者自定義的‘Key’，對應網頁上所要呈現的翻譯詞 ‘Value’所構成；而一個Value，可以使用一個或以上的Key

階層結構定位。例如:在英文JSON翻譯檔下，i18n這個Value，其Key值是“EXAMPLE.INTERNATIONALIZATION”(如圖xx)

而在前端框架中，若我們要透過JSON格式的多國語言翻譯檔，進行多國語言網頁翻譯，則必須使用第三方套件，在此以AngularJS[]搭配第三方套件angular-translate[]為例，將英文網頁翻譯成中文網頁。首先，將待翻譯詞定義在JSON格式的英文翻譯檔(如圖xx)，以及中文翻譯定義在JSON格式的中文翻譯檔(如圖xx)，並在HTML內的需翻譯詞處加上對應的標記(如圖xx)。之後，AngularJS便會根據當前網頁的語言-中文，以及待翻譯元件標記處的Key值-“EXAMPLE.INTERNATIONALIZATION”，去中文的JSON翻譯檔查找該Key值對應的Value，最後將找到的翻譯詞-「國際化」顯示於網頁上。

* 1. **Robot Framework**

Robot Framework[7][8]是一個開源的框架語言，可以用來執行自動化驗收測試或機器人自動化，核心框架是由Python編寫而成，測試者可以使用Python或Java擴充其函式庫。其特色是擁有簡單的語法，以及容易理解的原生關鍵字(Keyword)，測試者可以視需求使用並包裝成更接近自然語言的關鍵字。

* + 1. **Robot Framework 關鍵字**

Robot Framework透過許多的關鍵字(Keyword)來構成一份測試腳本，關鍵字分為兩種: Robot Framework的原生關鍵字，以及使用者自己定義的關鍵字。Robot Framework 的原生關鍵字透過引入Library來使用，能夠達成絕大多數對網頁上元件的基本操作，例如:Click Element、Open Browser；或是一些基礎的邏輯判斷，例如:Should Be Equal；以及一些對變數的設定與印出，例如:Set Variable、Log。

而使用者自定義的關鍵字，則是根據當下測試案例(Test Case)的需要，將一或多個原生關鍵字包裝成為更具體、易讀、能夠描述當下測試動作的關鍵字；例如:想使用測試腳本去點擊Microsoft官方網頁上的「支援」按鈕，則可撰寫 “Click Support Button On Microsoft Page”這個自定義的關鍵字，裡面包裝了原生關鍵字Open Browser以及Click Element，分別執行打開瀏覽器和點擊按鈕的動作。

此外，實際使用上，隨著每個人或組織的要求程度不同，包裝原生關鍵字成為自定義的關鍵字也會有嚴謹程度上的不同。但是Robot Framework本身對這部分並沒有強制的規定，是很自由的。

* + 1. **Robot Framework測試腳本**

一個Robot Framework 的測試腳本基本上由三個區塊構成，分別是

1. Settings: 包含了使用到的library與resource file，也可以將Test Setup(測試腳本執行前要做的動作)、Test Teardown(測試腳本執行後要做的動作)定義於此。
2. Test Cases: 在此測試者可以為各項想要驗證的使用者需求，撰寫核心的測試腳本
3. Keywords: 如果測試腳本使用並非Robot Framework的原生關鍵字，或未被定義於其他resource file中，測試者可以在此處撰寫出新的關鍵字去達到測試目的。
4. Variables: 測試者可以在此處定義測試案例需要用到的Global變數。
   * 1. **Robot Framework 測試報表**

在測試腳本運行結束後，Robot Framework會產生出一份測試報表，記錄了整體的執行狀況，包含測試通過或失敗、腳本執行時間，並透過可展開的階層式圖表，方便測試者去追蹤到當前出現問題的關鍵字，進而修復錯誤。

* + 1. **Robot Framework Listener**

Robot Framework提供了Listener[]的介面，用以接收測試執行到不同時期的資訊，並可以透過實作此介面，在測試腳本執行的不同時期做出額外的反應；例如實作end\_suite()函式，在測試結束後開啟一詞多譯UI，供使用者選擇。

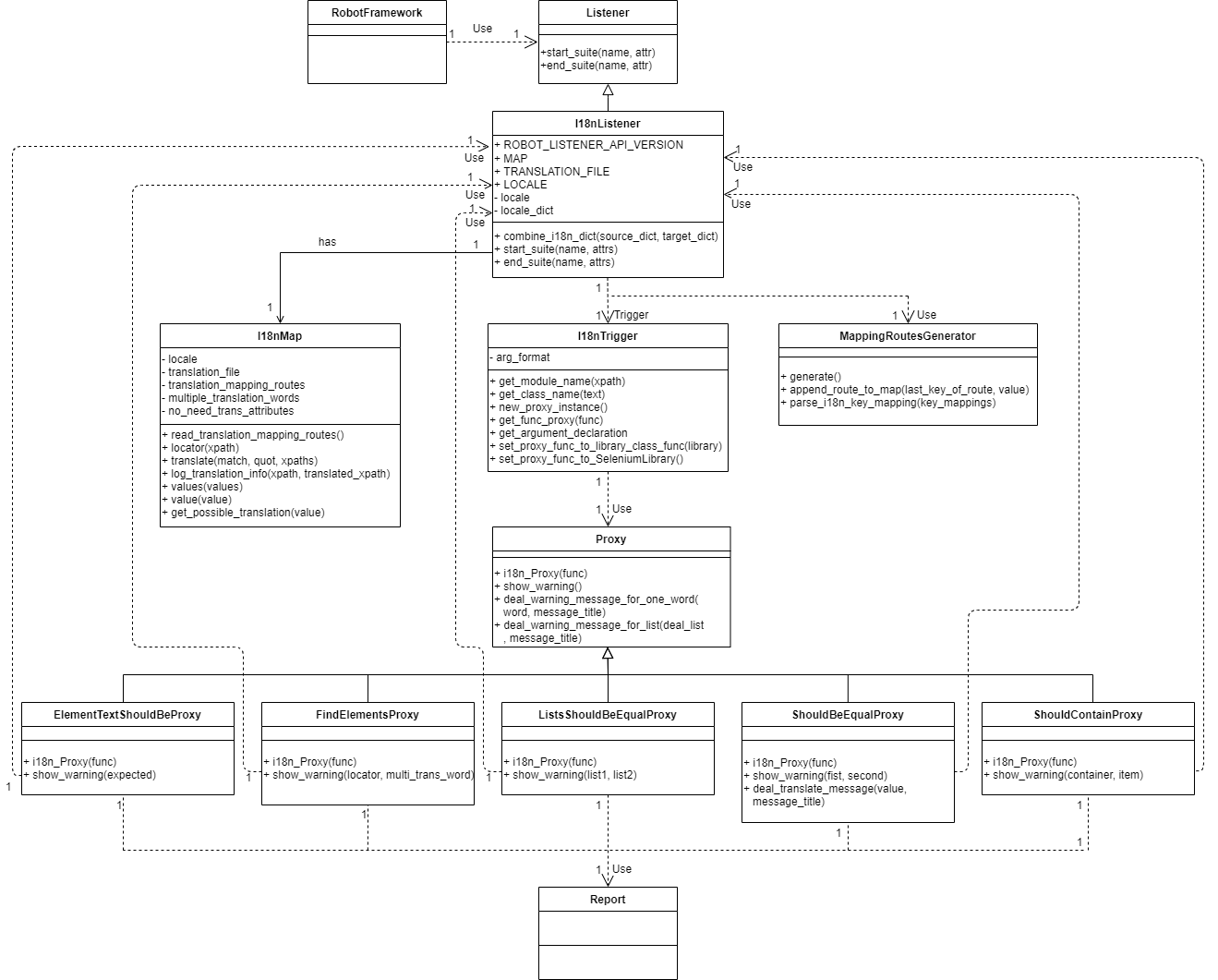
此外，Robot Framework的Listener分為2和3兩種版本；以本論文來說，使用到的是版本2，必須在實作Listener處加上:

以供系統辨認要實作的Listener版本。而為了讓系統在測試執行時呼叫Listener，在執行測試腳本前必須設定--listener組態，來指定需要使用到哪些自定義的Listener。

* 1. **第一版i18n的系統介紹**

以下將分成第一版i18n的系統架構、執行流程，以及執行測試腳本時會產生「一詞多譯」問題的緣由，來說明第一版i18n工具的相關背景知識。

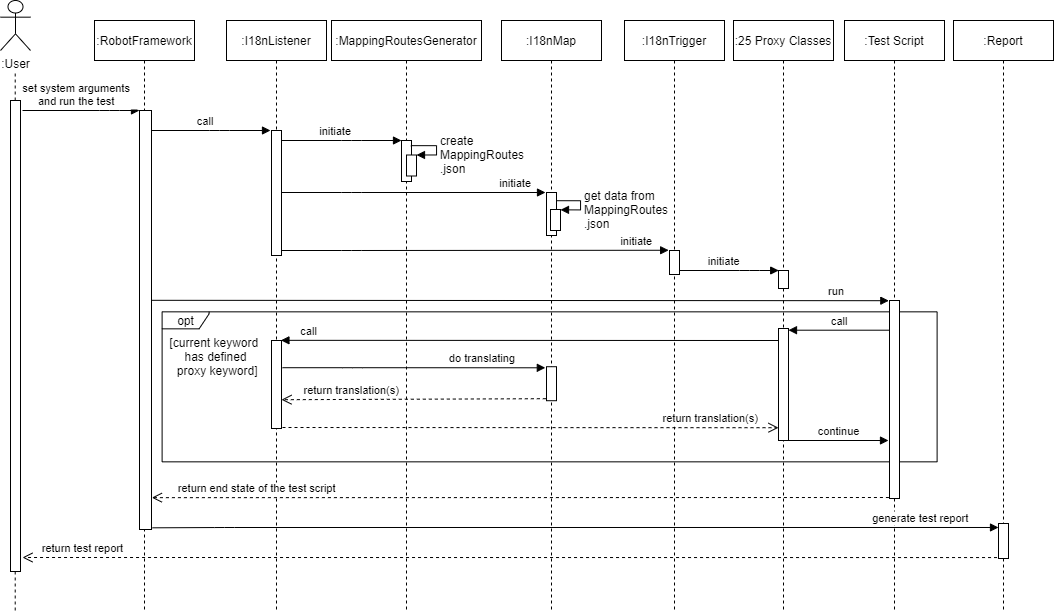
* + 1. **第一版i18n的系統架構**



第一版i18n的系統類別圖(如圖)，包含了3個 Robot Framework原生類別: RobotFramework，Listener，和Report；第一版i18n所設計的5個系統類別: I18nListener、MappingRoutesGenerator、I18nMap、I18nTrigger、Proxy，以及5個實作自Proxy[]類別的代理關鍵字類別: ShouldContainProxy、ElementTextShouldBeProxy、FindElementProxy、ListsShouldBeEqualProxy、ShouldBeEqualProxy。[]

I18nListener類別實作Listener類別，負責程式開始與結束時執行start\_suite()和end\_suite()的內容，並對MappingRoutesGenerator、I18nMap、I18nTrigger三個類別進行初始化的動作。I18nMap類別負責處理各個代理關鍵字類別透過I18nListener類別的呼叫，並將待翻譯詞做翻譯後回傳。I18nTrigger類別透過new\_proxy\_instance()函式建立各個代理關鍵字類別的實例；並使用set\_proxy\_func\_to\_library\_class\_func()、set\_proxy\_func\_to\_SeleniumLibrary()兩函式，將代理關鍵字類別的實作包裝於Robot Framework原生關鍵字之外，使每次呼叫關鍵字時，必先執行代理關鍵字的實作。MappingRoutesGenerator類別負責以英文JSON翻譯檔為基準，產生出翻譯路徑檔，讓測試腳本運行在其他語言環境下，可以根據此路徑，在所屬語言的JSON翻譯檔下找到正確的翻譯。Proxy類別提供了一個介面，讓實作Proxy類別的代理關鍵字類別可以根據各自的需要，去擴充內部的實作；其中，i18n\_Proxy()函式提供各代理關鍵字類別撰寫核心的功能。

* + 1. **第一版i18n的系統執行流程**



第一版i18n系統流程的Sequence Diagram (如圖XX)，在使用i18n工具執行測試前，使用者必須於Red編輯器[15]中的Additional Robot Framework arguments設定系統參數，如圖3-1-3-1，設定為-d out –L debug -–listener i18n/listeners/I18nListener.py:zh-TW，zh-TW代表當前語言為繁體中文-台灣(如圖xx)。並且，使用此i18n工具的前提，必須建立在「使用者對於自己所提供的JSON翻譯檔，有充分的了解」，如此，當遭遇一詞多譯時，使用者才會明確知道在當下情況，什麼翻譯是正確的。(此前提套用到新版的i18n工具上亦然)

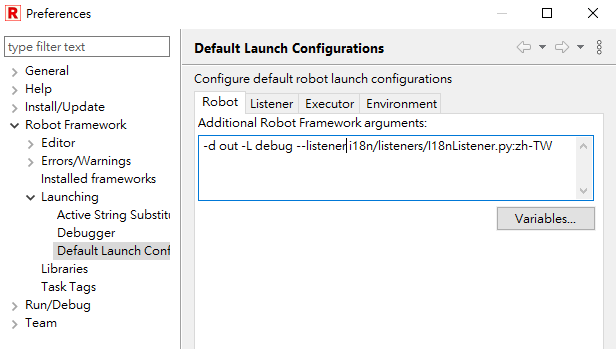
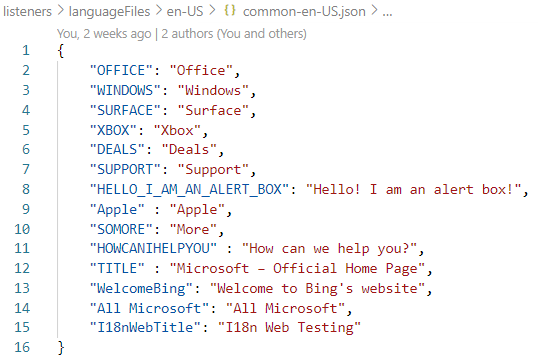
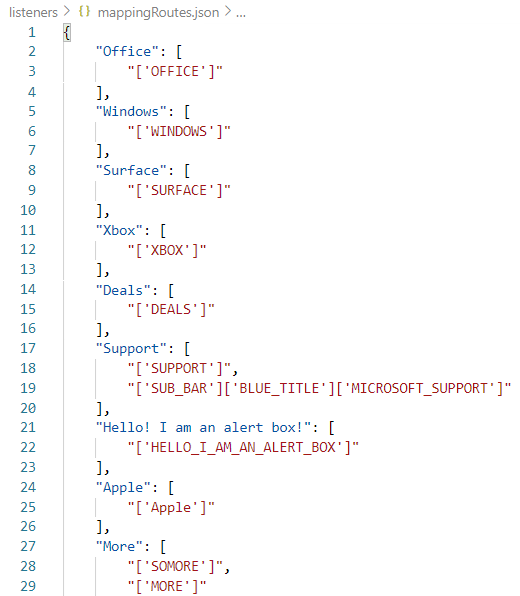


圖3-1-3-1 Robot Framework系統參數設定

測試執行時，系統會先去呼叫I18nListener類別並執行其實作。之後初始化MappingRoutesGenerator類別，藉由讀取由使用者自行提供的英文JSON格式翻譯檔(如圖3-1-3-2)，建立出一份由”待翻譯詞”和“Key階層”構成的翻譯路徑檔(如圖3-1-3-3) ；例如待翻譯詞｀Support＇，有兩種 Key 階層的路徑:“[’SUPPORT’]” 和“[’SUB\_BAR’][’BLUE\_TITLE’][’MICROSOFT\_SUPPORT’]”，代表｀Support＇在多國語言網頁中，會在不同情境下被翻譯成兩種不同的詞彙。接著系統會依序初始化I18nMap、I18nTrigger，以及所有代理關鍵字類別。



圖xx 英文的JSON格式翻譯檔示例



圖xx 翻譯路徑檔部分示例

當執行完上述工作後，測試腳本才會正式開始執行；每當腳本運行到一個有定義代理之關鍵字，系統就會去呼叫對應的代理關鍵字物件，並執行翻譯邏輯。最後，當測試腳本執行結束，系統則會將一詞多譯的warning資訊顯示在報表上。

* + 1. **產生一詞多譯問題的緣由**

執行多國語言網頁自動化驗收測試時，會產生「一詞多譯」問題的根本原因，源自於前端多國語言的網頁設計中，可能必須根據當下網頁情況的不同，將同一個詞語，在不同語言的網頁上翻譯成為不同的詞。例如: 在Microsoft英文官方網頁上的‘Support’，在中文版網頁的不同對應頁面上，被分別翻譯為「支援」與「支援服務」。

此外，前端開發人員會隨著多國語言網頁的設計，會產出一包JSON翻譯檔，裡面定義著不同語言之間，同一個詞語應該被如何翻譯；並將此包JSON翻譯檔交給測試人員，以進行多國語言的網頁自動化驗收測試。

然而，對於測試人員而言，假如要驗證畫面上的一個元件，其text()屬性值是‘Support’的翻譯，在遭遇一詞多譯時，JSON檔卻只能告知‘Support’可能被翻譯成「支援」或「支援服務」；JSON檔內的Key階層對於測試人員而言並沒有太大的意義，因為測試腳本的撰寫是站在模擬使用者操作的角度，並不會把前端開發者的設計思維考慮進去。如此，就會造成多國語言網頁的一詞多譯特性，在測試人員進行自動化驗收測試的階段時，發生系統不知該如何選擇當下正確翻譯詞的問題。

第一版的i18n工具，在遭遇一詞多譯時，僅於測試報表上顯示warning資訊，提供待翻譯詞可能的翻譯有哪些；雖然成功的呈現了一詞多譯的問題，但尚未給出一個有效的解法。

# 第三章 新版i18n的系統設計

本章節將詳述如何針對i18n工具現存的四項問題，進行思考與改善，並點出與朱峻平的論文“支援多國語言的Robot Framework網頁自動化驗收測試”(第一版i18n工具)的不同之處。

* 1. **系統擴充**

新版i18n系統類別圖(如圖3-1-1)，沿用第一版i18工具的架構(如圖)，經過部分實作的改善，並且新增了一個用於顯示一詞多譯選項的圖形化使用者介面類別(UI)，以及20種代理關鍵字的類別。(新增的類別在圖中以紅色方框標示)

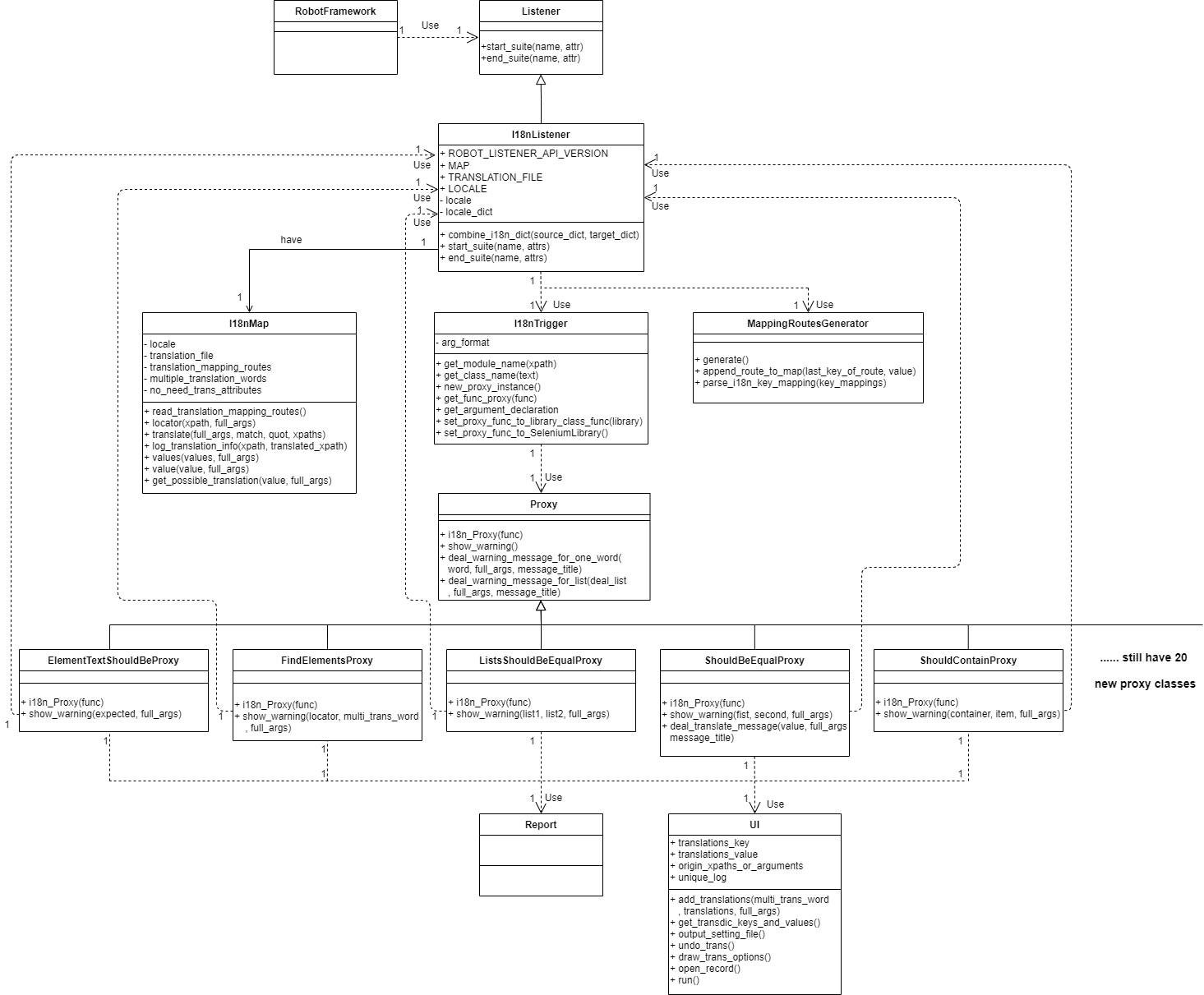


圖3‑1-1 新版i18n系統類別圖

本論文新增的UI類別會在程式執行期間，記錄下遭遇到一詞多譯的詞語以及其翻譯。在程式執行後，產生一詞多譯的圖形化使用者介面，使用者選擇並提交希望的翻譯後，便會產生一個系統設定檔記錄翻譯選擇。

而其他20種新增的代理關鍵字類別(如表3-1-2-1)，皆實作父類別Proxy，負責代理各自對應之原生關鍵字的參數部分翻譯，並將翻譯好的參數部分回傳給原生關鍵字。

|  |  |
| --- | --- |
| 代理關鍵字類別 | |
| 1. AlertShouldBePresentProxy  2. CountValuesInListProxy  3. DictionariesShouldBeEqualProxy  4. DictionaryShouldContainItemProxy  5. DictionaryShouldContainKeyProxy  6. DictionaryShouldContainValueProxy  7. GetMatchCountProxy  8. ListSelectionShouldBeProxy  9. ListShouldContainSubListProxy  10. ListShouldContainValueProxy | 11. ListShouldNotContainDuplicatesProxy  12. RemoveFromDictionaryProxy  13. RemoveValuesFromListProxy  14. SelectFromListByLabelProxy  15. SelectFromListByValueProxy  16. TableCellShouldContainProxy  17. TableColumnShouldContainProxy  18. TableRowShouldContainProxy  19. TableShouldContainProxy  20. TitleShouldBeProxy |

表3-1-2-1 本論文新增的20種代理關鍵字類別

系統執行流程相較於第一版i18n改動較大的部份，是在測試腳本結束後；除了將一詞多譯的warning資訊顯示在報表上外，若是遭遇過一詞多譯，便會跳出一詞多譯的UI(詳見3-4節)，記錄了執行翻譯當下關鍵字的參數組合，並顯示所有可能的翻譯詞，讓使用者可以從中去選擇並產生一個設定檔。之後再次執行測試腳本時，系統便會根據設定檔的內容去選擇適當的翻譯詞，同時消除報表上的warning提示資訊。

* 1. **擴充與修改代理關鍵字使其完整支援原生關鍵字**

我們需要為一個Robot Framework的原生關鍵字撰寫代理關鍵字，有以下兩點情況:

* 1. 關鍵字的參數部分包含純文字部分，所以有翻譯的需求，可能是基本的string、list、dictionary，例如: 關鍵字should\_be\_equal，其參數部分代入‘Support’，‘Support’，去驗證兩者是否相同。
  2. 關鍵字的參數部分包含XPath，且XPath內的HTML屬性會包含純文字的部分，所以有翻譯的需求。例如: 關鍵字find\_element，其參數部分代入XPath “//\*[@id ='supHomeAndLandingPageHeaderContainer']//\*[contains(text(), 'Support')]”，去驗證畫面上的橫向標題container是否包含Support字樣。

根據朱峻平的論文，第一版i18n工具只支援7種Robot Framework原生關鍵字的代理 (如表3-2-1)[]；並整理出尚有31種參數部分需要翻譯的原生關鍵字，未提供支援。如果測試腳本使用到這些未支援的原生關鍵字去跑多國語言網頁測試，則會導致出錯。因此，本論文的解法是擴充完目前Robot Framework版本剩下的代理關鍵字，使得有翻譯需求的原生關鍵字，其參數部分都能正確的被翻譯。

|  |
| --- |
| 第一版i18n已提供代理的關鍵字 |
| 1. find\_element 2. element\_text\_should\_be 3. lists\_should\_be\_equal 4. should\_be\_equal 5. should\_not\_be\_equal 6. should\_contain 7. should\_not\_contain |

表3-2-1 第一版i18n已提供代理的Robot Framework原生關鍵字

且又因為先前的i18n工具分別在Robot Framework版本3.0.4、SeleniumLibrary版本3.1.1下開發，而現在的Robot Framework版本已更新到3.2.2，SeleniumLibrary版本則更新到4.5.0，導致原先需要被支援的31種原生關鍵字，有兩種已被淘汰:

input\_text\_into\_prompt、xpath\_should\_match\_x\_times。因此，還有29種原生關鍵字需要提供代理(如表3-2-2)。(‘\*’標示為因版本更新導致被淘汰的原生關鍵字)

|  |  |
| --- | --- |
| 待擴充代理的關鍵字 | |
| Collections Library (3.2.2) | SeleniumLibrary (4.5.0) |
| 1. count\_values\_in\_list 2. dictionaries\_should\_be\_equal 3. dictionary\_should\_contain\_item 4. dictionary\_should\_contain\_sub\_dictionary 5. dictionary\_should\_contain\_key 6. dictionary\_should\_not\_contain\_key 7. dictionary\_should\_contain\_value 8. dictionary\_should\_not\_contain\_value 9. list\_should\_contain\_sub\_list 10. list\_should\_contain\_value 11. list\_should\_not\_contain\_value 12. list\_should\_not\_contain\_duplicates 13. remove\_from\_dictionary 14. remove\_values\_from\_list 15. get\_match\_count | 1. alert\_should\_be\_present 2. input\_text\_into\_alert 3. \*input\_text\_into\_prompt 4. \*xpath\_should\_match\_x\_times 5. list\_selection\_should\_be 6. select\_from\_list\_by\_label 7. unselect\_from\_list\_by\_label 8. select\_from\_list\_by\_value 9. unselect\_from\_list\_by\_value 10. title\_should\_be 11. table\_should\_contain 12. table\_header\_should\_contain 13. table\_footer\_should\_contain 14. table\_cell\_should\_contain 15. table\_row\_should\_contain 16. table\_column\_should\_contain |

表3-2-2 新版i18n將擴充代理的Robot Framework原生關鍵字

未來若又經歷了Robot Framework和SeleniumLibrary版本的更新，可以透過查閱Robot Framework和SeleniumLibrary在官方github[][]上的release notes來了解版本變更資訊，並對新版本新增/修改/刪除的原生關鍵字進行代理關鍵字實作上的更新。在此將以實際的操作步驟，說明本論文是如何透過查閱SeleniumLibrary官方github上的release notes，發現input\_text\_into\_prompt 和xpath\_should\_match\_x\_times關鍵字已被淘汰。

首先，到SeleniumLibrary的release頁面，因為本次i18n工具是從SeleniumLibrary的版本3.1.1升至4.5.0，所以一個個往上找，最後於版本4.0.0的release notes處發現了issue編號 #1274 “Remove keywords which where officially deprecated in previos releases”有異樣(如圖xx)，點擊連結進入此issue，進一步發現issue編號#1334，再次點擊進入，並在Files changed頁面中，發現input\_text\_into\_prompt和xpath\_should\_match\_x\_times關鍵字已在此次更新後被刪除(如圖xx)。

講完了新版i18n將要擴充的代理關鍵字有哪些，以及未來再次遇到Robot Framework相關Library版本更新的處理方法，以下將以 FindElementsProxy的Flow Chart為例，詳述現今配合新增的一詞多譯UI下，本論文是如何完成代理關鍵字的擴充和修改。(沿用第一版i18n的部分會以‘\*’號標註)

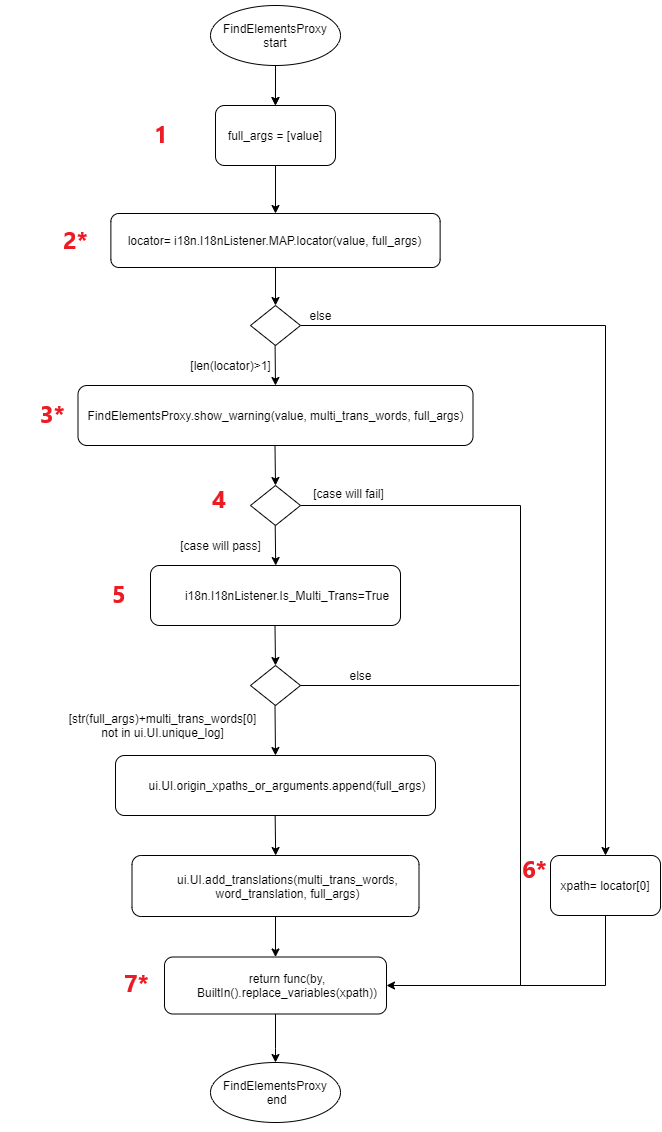


圖3-2-1 FindElementsProxy的Flow Chart

由圖3-2-1 FindElementProxy的Flow Chart可以得知，在新的i18n版本下，擴充一個新的代理關鍵字必須遵從著以下步驟: (以下編號可以對應到Flow Chart的數字)

1. 創出該次關鍵字呼叫的參數紀錄變數，full\_args。此變數之後會成為判斷當下待翻譯詞是否“已被使用者選擇翻譯” 的依據。並於測試腳本結束後，顯示於一詞多譯UI上，最後隨著使用者的選擇一併存入設定檔內。
2. 執行翻譯。如此例的:

locator = i18n.I18nListener.MAP.locator(BuiltIn().replace\_variables(value), full\_args)， 透過i18nListener類別的MAP變數(其內裝著i18nMap類別的初始化資訊)，去呼叫i18nMap類別的locator函式，對待翻譯詞value執行翻譯，且必須代入full\_args以實現1.的內容。

1. 若判定此代理關鍵字遭遇一詞多譯，則產生warning資訊於報表上，如:

FindElementsProxy.show\_warning(self, value, multiple\_translation\_words, full\_args)

1. 之後判斷此一詞多譯情況，最後會使測試通過或是失敗，如:

，以作為之後是否開啟一詞多譯UI的依據。因為在測試原本就會失敗的情況下，給使用者選擇錯誤待翻譯詞的翻譯，是毫無意義的。只有在原測試會通過的前提下，才需要讓使用者去選擇翻譯詞。

1. 若測試會通過，則對預計開啟的UI做一些資料的準備，如:

i18n.I18nListener.Is\_Multi\_Trans=True

if str(full\_args)+multiple\_translation\_words[0] not in ui.UI.unique\_log

ui.UI.origin\_xpaths\_or\_arguments.append(full\_args)

ui.UI.add\_translations(self, multiple\_translation\_words, word\_translation, full\_args)

先設定測試結束後，需要開啟一詞多譯UI。若該”參數+翻譯詞”組合尚未被翻譯過，則記錄下該次翻譯的參數部分和翻譯資訊。

1. 若此次代理關鍵字沒有遭遇一詞多譯，則如:

xpath = locator[0]，將第一筆翻譯同時也是唯一一筆翻譯指派給之後要回傳的xpath。

1. 將翻譯好的參數部分回傳給原生的Robot Framework原生關鍵字，如:

return func(self, by, BuiltIn().replace\_variables(xpath))

其他未能一一介紹的代理關鍵字，儘管彼此間實作仍然存在著差異，但都是遵循著以上的架構去設計。

* 1. **改善XPath翻譯邏輯使其能應對各種HTML屬性**

在第一版i18n的翻譯邏輯中，若一個XPath內部存在多種HTML屬性，i18n工具使用了列舉法，僅提供了@title、text()、normalize-space()三種屬性的翻譯規則(如下方的程式部分)。屆時，若測試腳本的XPath是用沒有列舉出來的屬性撰寫，但卻有翻譯的需求，例如:@placeholder、@arial-label等等，則會導致測試出錯。

Code Here

為了解決以上問題，本論文最後決定採用“負面表列法”[]，將確定不會執行翻譯的HTML屬性在程式中用list的方式儲存。若XPath中的HTML屬性不在該list中，則都去執行翻譯檢查。

此種作法相較於把所有HTML屬性都執行翻譯檢查，顧及到了系統執行效能，來的更加省時。相較於設計一個複雜邏輯，來訓練系統自行判斷當前HTML屬性是否要翻譯，在實作上則更為簡單，且把翻譯的決定權交給了測試者，而非全部靠系統。相較於第一版i18n，把翻譯規則用列舉法固定，此作法則更能顧及到未來測試腳本的不確定性，因為我們無法預期使用者會用怎麼樣的方式去編寫XPath。

以下將以新版i18n XPath翻譯邏輯的Sequence Diagram為例(如圖3-3-2)，輔以部分程式碼，詳述系統是如何根據當下情況，產生一套新的翻譯邏輯，以解決先前某些HTML屬性無法被翻譯的問題。

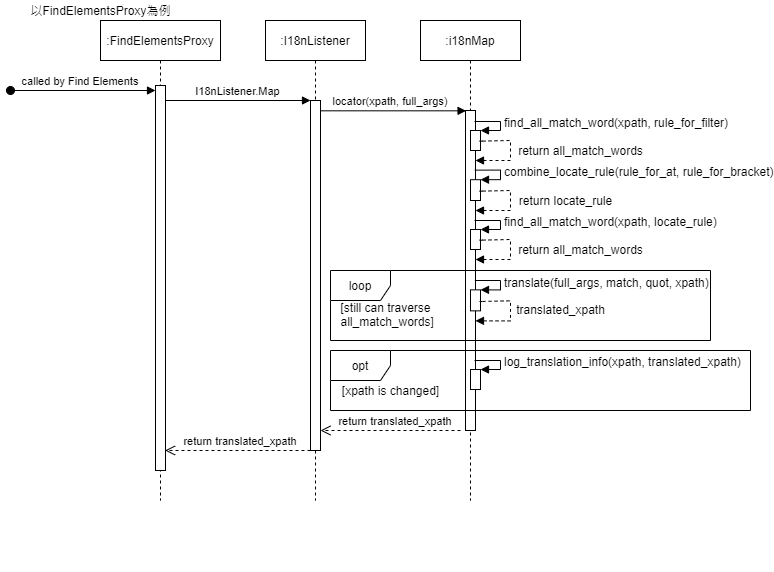


圖3-3-2 新版i18n XPath翻譯邏輯的Sequence Diagram

首先，被呼叫後，FindElementsProxy透過I18nListener類別，呼叫I18nMap類別的locator()函式執行XPath的翻譯。之後，I18nMap先定義一個內含HTML屬性特徵的rule\_for\_filter變數，如:

隨後呼叫find\_all\_match\_word()，來過濾出XPath中包含的所有屬性。

接著，判斷這些屬性是否包含在“不需被翻譯屬性”的list中。如:

若不包含其中，則根據屬性含有 ‘@’ 或 ‘()’，分別分配給rule\_for\_at和rule\_for\_bracket變數，並呼叫combine\_locate\_rule(rule\_for\_at, rule\_for\_bracket)，得到一套新的翻譯邏輯，locate\_rule。

之後，再次呼叫find\_all\_match\_word(xpath, locate\_rule)，得到所有符合新翻譯邏輯的HTML屬性，all\_match\_words。並利用迴圈遍歷每個all\_match\_words中的屬性，執行translate(full\_args, match, quot, xpath)，進行翻譯檢查。

接著，判斷經過翻譯後，原先的xpath是否發生變化；例如xpath從原本的//\*[text()= ‘Software’]，變成了翻譯後的//\*[text()= ‘軟體’]。若xpath改變，則執行log\_translation\_info(xpath, translated\_xpath)，將翻譯資訊顯示於測試報表上。

最後，將翻譯後的translated\_xpath，回傳給FindElementsProxy，完成一次XPath的翻譯。

* 1. **提供圖形化使用者介面解決一詞多譯的問題**

先前遇到一詞多譯問題時，第一版i18n工具提供了warning資訊於測試結束後的報表上，此作法確實提醒了使用者該腳本存在著一詞多譯問題，且也顯示出目前系統採用的翻譯詞，這是好的部分，本論文將繼續沿用。此翻譯邏輯是站在一個「最大化讓測試腳本通過」的立場，當遭遇一詞多譯時，系統會將所有的翻譯都執行看看，直到一包可能的翻譯中，出現了一個會使測試通過的翻譯，則算測試通過，並將該翻譯詞呈現於報表上。

然而，以上作法卻存在著一個缺陷，即是「使用者無法自由的選擇真正想要的翻譯詞」去執行腳本測試，而把決定全權授予系統。並有機會產生以下兩點問題:

1. 假如存在”多個翻譯都會使測試通過”，翻譯過後的腳本測試對象，就有機會偏離使用者原先的預期。其原因可能是XPath使用contains撰寫，使得翻譯只要包含特定詞即會讓測試通過；如: 翻譯過後的XPath為//\*[@id= ‘supHomeAndLandingPageHeaderContainer]//\*[contains(@text, ‘支援’)]，

而畫面上待驗證的元件顯示的詞是‘支援服務’，測試會通過，但遺憾的是使用者預期Support在此處應該要被翻譯為支援服務。

又或是畫面上同時存在含有不同翻譯的元件，且滿足測試腳本的XPath，而導致測試通過。如: 畫面上兩個元件，分別顯示支援與支援服務；原本使用者預期要驗證的是畫面上存在支援服務字樣，但因為‘Support’先被系統檢查的翻譯為支援，導致翻譯後的XPath //\*[@text= ‘支援’]率先通過了測試，而不會再去檢驗後續的翻譯。

2. 假如此測試腳本原先就會發生錯誤，經過翻譯後的XPath卻因為碰巧滿足畫面上的某個元件，而導致測試通過。在此特殊情形下，翻譯過後的測試對象，則更明顯的偏離了使用者原先的預期，且也改變了測試結果。如: 原腳本要驗證//\*[@text= ‘Service’]，測試會失敗，經過翻譯後XPath變成//\*[@text= ‘支援服務’]，測試卻會通過，因為畫面上剛好存在支援服務字樣。

所以，本論文將提供一個圖形化使用者介面，記錄下執行翻譯時當下的關鍵字參數組合，並顯示所有可能的翻譯詞，讓使用者從中去選擇，並產生一個設定檔，以便之後再次執行同一測試腳本時，根據設定檔的內容去選擇適當的翻譯詞。期望透過如此的方法來改善上述兩點一詞多譯所遭遇的問題。

以下，將透過圖表和Sequence Diagram的方式，來詳述一詞多譯UI的介面設計與整體實作:

一詞多譯UI的介面設計(如圖3-4-1)，本論文將其設計成兩個區塊，上半部以條列的方式呈現完整的翻譯資訊，包含關鍵字名稱、參數部分、待翻譯詞、可能的翻譯。其中，參數部分記錄了當前關鍵字接受的使用者傳入參數，以作為和其他相同待翻譯詞的區別；因為相同的待翻譯詞，在不同情況下，可能擁有不同的翻譯。

下半部包含一行功能資訊的提示、一個用於開啟翻譯紀錄頁面的TransRecord按鈕，以及提交使用者選擇並寫入設定檔的Submit按鈕。

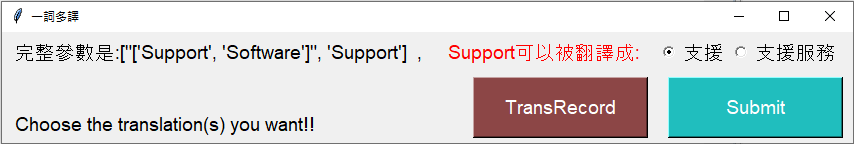


圖3-4-1一詞多譯UI的介面設計

翻譯紀錄的介面設計(如圖3-4-2)，本論文同樣將其設計成兩個區塊，上半部以條列式呈現設定檔中現存的使用者翻譯選擇。下半部包含一個Undo按鈕，考量到使用者可能不小心選錯了翻譯，用於將使用者的選擇從設定檔中刪除。

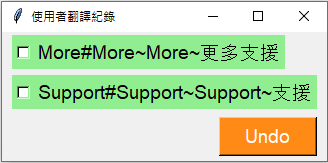


圖3-4-2 翻譯紀錄的介面設計

一詞多譯UI的實作，大致可以分成8個部分: run()、draw\_trans\_options()、get\_transdic\_keys\_and\_values()、open\_record()、undo\_trans()、output\_setting\_file()、add\_trans\_info ()、add\_keyword\_name()。各實作彼此之間的互動關係，則如圖3-4-3

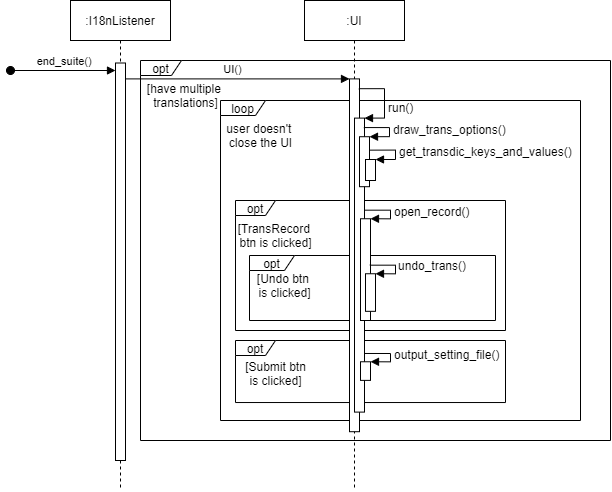


圖3-4-3產生一詞多譯UI的sequence diagram

當測試腳本結束時，會呼叫I18nListener類別的end\_suite()函式。

若本次測試腳本遭遇一詞多譯，則呼叫UI類別，使其初始化並執行run()函式，產生一詞多譯UI的介面，包含”Choose the translation(s) you want!!” 字樣、顯示翻譯紀錄的 TransRecord按鈕，以及寫入使用者選擇的Submit按鈕。之後，run()函式內部會呼叫draw\_trans\_options()函式，其內會再呼叫get\_transdic\_keys\_and\_values()函式，取出待翻譯詞與其對應的翻譯，最後隨著翻譯當下的關鍵字名稱和參數部分一併顯示於一詞多譯UI的介面上。

在一詞多譯介面上，點擊TransRecord按鈕會呼叫open\_record()函式，打開翻譯紀錄介面，其會將先前使用者選擇過的翻譯詞，以條列式的方式呈現，且介面上還包含一顆Undo按鈕。點擊Undo按鈕會呼叫undo\_trans()函式，將使用者選擇的翻譯紀錄從設定檔中清除掉，同時關閉翻譯紀錄頁面。

在一詞多譯介面上，點擊Submit按鈕則會呼叫output\_setting\_file()函式，將使用者選擇的待翻譯詞對應其翻譯以及完整參數寫入設定檔 “i18n/listeners/setting.txt”中，同時將剛剛已寫入的翻譯資訊從一詞多譯UI上隱藏。

此外，在UI類別中定義的add\_trans\_info ()和add\_keyword\_name()函式，雖然不會在一詞多譯UI執行期間被呼叫，但在測試腳本執行期間，卻能負責將代理關鍵字傳入的關鍵字名稱、待翻譯詞，與其對應翻譯，分別儲存於變數中，方便之後一詞多譯UI的讀取。並且，還會將關鍵字的參數部分和待翻譯詞結合，成為可以辨別“該次測試腳本是否執行過相同翻譯”的字串，存入unique\_log變數中。

* 1. **將i18n工具設計成為可以安裝的模組**

本論文透過Python的build模組，將i18n工具包裝成名為”RF-i18n-tool”的Library，並透過twine模組上傳至PYPI[16]網站上，使之成為可以直接讓使用者安裝的Python模組。解決了先前使用者只能從github上將i18n工具clone下來，並在該專案上開發測試腳本的不方便。

以下將詳述將i18n包裝成Python模組的步驟:

在將i18n工具打包成為一個Library前，必須先準備以下五個檔案: setup.cfg、pyproject.toml、README.md、LICENSE[17]、MANIFEST.in。此外必須確認將打包的資料夾下，包含\_\_init\_\_.py檔案，以被系統辨識為一個Python模組。

上述的檔案準備就緒後，便可以在i18n專案目錄下執行python –m build 指令，待建置完畢，便會產生一個名為dist的資料夾，裡面裝著打包好的RF-i18n-tool模組檔案。接下來執行python –m twine upload --repository pypi dist/\* 指令，輸入帳號密碼後，便能成功將dist資料夾中的RF-i18n-tool模組上傳至Pypi網站上了。之後在PYPI網站上搜尋RF- i18n-tool，便能看到上傳的模組。

使用者若要安裝此模組，只需執行pip install RF-i18n-tool 指令即可(如圖3-5-1)。

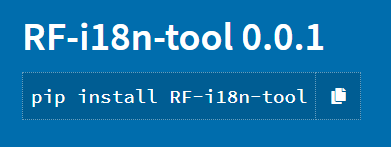


圖3-5-1 安裝RF- i18n-tool指令

最後，使用者便可透過在Additional Robot Framework arguments設定系統參數(如圖xx)，來使用i18n工具。自定義的JSON翻譯檔路徑必須遵守以下格式(如圖xx)

# 第四章 測試案例分析

本章將為擴充的代理關鍵字撰寫單元測試[18]，並且將改善翻譯邏輯與一詞多譯功能後的i18n工具，做功能的展示。最後將i18n工具套用至一個使用到多項代理關鍵字的Robot Framework測試腳本，以呈現“多國語言網頁自動化驗收測試”的效果。

* 1. **新增與修改之代理關鍵字的單元測試**

以下將以Microsoft中文官方網頁[10]，以及筆者使用Node.js[19]自行架設的i18n測試網頁(i18n Testing Website)為受測網站，為各個新增或已修改實作之代理關鍵字，分別提供Robot Framework的單元測試腳本，以驗證原生關鍵字在套用新的代理關鍵字下，能夠獨立正常運作。但為求精簡，本論文將不重複放上同類型關鍵字的測試腳本。(經修改的第一版i18n代理關鍵字前將加上‘\*’標示)

**4.1.1 Alert Should Be Present**

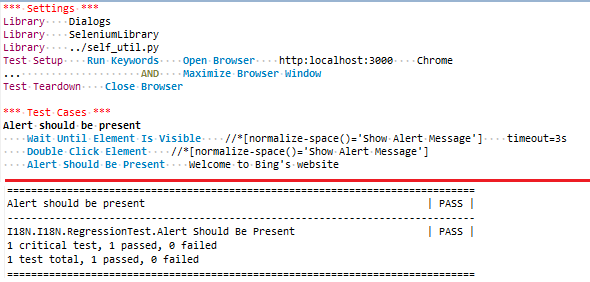


圖4.1.1-1 Alert Should Be Present在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.1-1)會打開i18n Testing Website，並雙擊Show Alert Message按鈕，使畫面跳出alert視窗(如圖4.1.1-2)。最後使用Alert Should Be Present來驗證alert資訊是否為“歡迎光臨Bing的網頁”。測試通過，代表參數“Welcome to Bing’s website”有成功被翻譯為“歡迎光臨Bing的網頁”(如圖)。

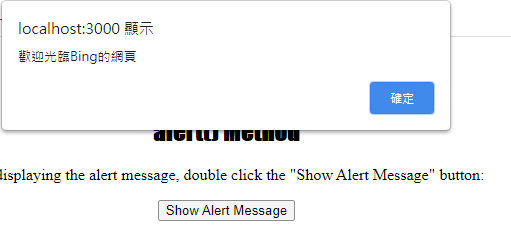


圖4.1.1-2 網頁上的alert視窗顯示的文字

**4.1.2 Count Values In List**

圖4.1.2-1 Count Values In List在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.2-1)，一開始會利用Create List創出一個包含“支援”和 ‘Software’的list，並使用Count Values In List來計算參數“Support”出現在此list中幾次。第一次執行測試通過(如圖4.1.2-2)，透過Log印出的數字為1，代表參數Support被翻譯為’支援’後，出現在list中一次，所以測試通過；且系統將“支援”當成“Support”翻譯的其中一種，並跳出一詞多譯UI。之後，使用者選擇“支援”作為唯一翻譯。第二次測試通過(如圖4.1.2-3)，透過Log印出的數字為1，且不跳出一詞多譯UI與warning資訊。

**4.1.3 Dictionaries Should Be Equal**

圖4.1.3-1 Dictionaries Should Be Equal在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.3-1)，會利用Create Dictionary創出兩個dictionaries，分別是{‘軟體’: ‘支援’}和{‘Software’: ‘Support’}(測試腳本的 ‘=’左側為key，右側為value)，並使用Dictionaries Should Be Equal檢查兩者是否相同。第一次執行測試通過(如圖4.1.3-2)，代表dictionary {‘Software’, ‘Support’}有成功被翻譯，且系統將“支援”當成“Support”翻譯的其中一種，並跳出一詞多譯UI。之後，使用者選擇“支援”作為唯一翻譯。第二次測試通過(如圖4.1.3-3)，且不跳出一詞多譯UI與warning資訊，代表‘Support’ 根據使用者選擇成功被翻譯為‘支援’。

**4.1.4 Select From List By Label**

4.1.4-1 Select From List By Label在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.4-1)，會打開i18n Testing Website，並使用Select From List By Label選取畫面上selection list的Support選項(如圖4.1.4-2)。之後用List Selection Should Be驗證結果是否正確。第一次測試通過(如圖4.1.4-3)，而畫面上選擇了‘支援’，代表參數Support有成功被翻譯，且系統將‘支援’當作‘Support’的其中一種翻譯，隨後跳出了一詞多譯UI。之後，使用者選擇了‘支援服務’當作Support的唯一翻譯。第二次測試腳本通過(如圖4.1.4-4)，畫面上被選擇的selection list顯示‘支援服務’，代表參數‘Support’成功被翻譯為使用者前一次的選擇。



4.1.4-2 Select From List By Label測試腳本要驗證的網頁元件

**4.1.5 \*Page Should Contain Element**

圖4.1.5-1 \*Page Should Contain Element在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.5-1)，會打開Microsoft中文官方網頁，前往支援頁面，之後使用Page Should Contain Element驗證畫面上的Support文字(如圖4.1.5-2)。第一次測試通過(如圖4.1.5-3)，代表參數Support有成功被翻譯，且系統將‘支援服務’當作‘Support’的其中一種翻譯，隨後跳出了一詞多譯UI。之後，使用者選擇了‘支援服務’當作Support的唯一翻譯。第二次測試腳本通過(如圖4.1.5-4)，且沒有跳出一詞多譯UI，代表參數‘Support’成功被翻譯為使用者前一次的選擇。



圖4.1.5-2 \*Page Should Contain Element 測試腳本要驗證的網頁元件

**4.1.6 Table Should Contain**

4.1.6-1 Table Should Contain在代理關鍵字下執行的測試腳本與測試結果

此測試腳本(如圖4.1.6-1)，會打開I18n Testing Website，之後使用Table Should Contain 驗證畫面上的Support文字(如圖4.1.6-2)。第一次測試通過(如圖4.1.6-3)，代表參數‘Support’有成功被翻譯，且系統將‘支援’當作‘Support’的其中一種翻譯，隨後跳出了一詞多譯UI。之後，使用者選擇了‘支援’當作‘Support’的唯一翻譯。第二次執行測試腳本通過(如圖4.1.6-4)，且沒有跳出一詞多譯UI，代表參數‘Support’成功被翻譯為使用者前一次的選擇。



4.1.6-1 Table Should Contain測試腳本要驗證的網頁元件

* 1. **改善翻譯邏輯後的驗收測試示例**

以下將舉Page Should Contain Element關鍵字的測試腳本為例，去驗證Microsoft中文官方網頁上的文字。並說明在改善XPath翻譯邏輯前後，執行含有@placeholder屬性的XPath之腳本，將分別得到何種結果。

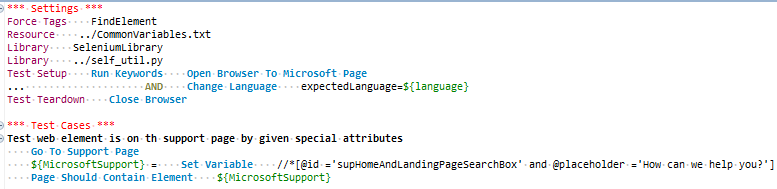


圖4-2-1 含有@placeholder屬性的XPath之測試腳本

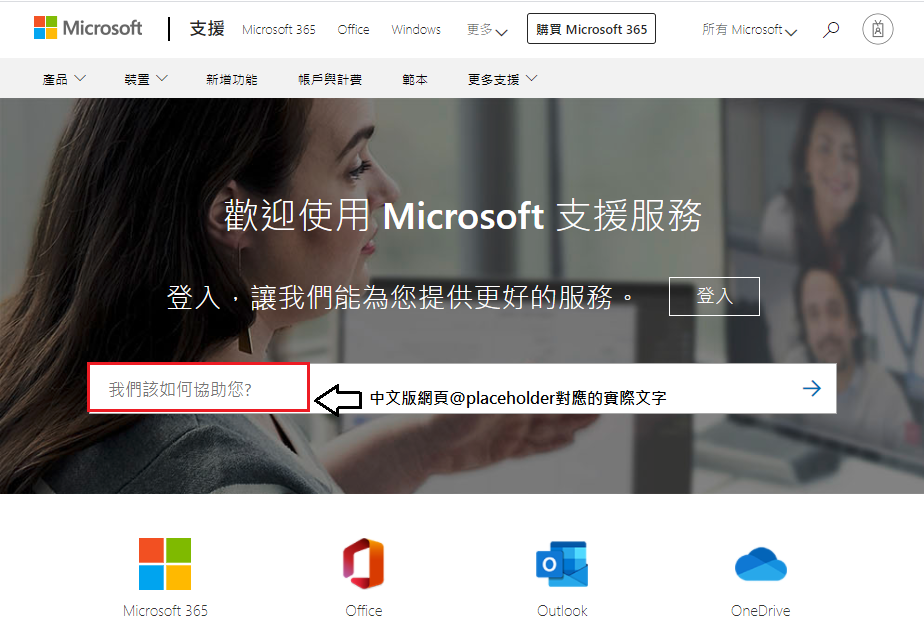


圖4-2-2 中文版Microsoft網頁@placeholder屬性對應的實際文字

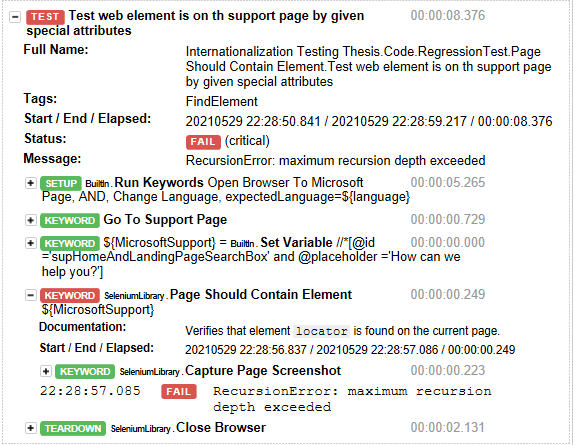


圖4-2-3 待翻譯XPath運行在未改善翻譯邏輯時的i18n工具之結果

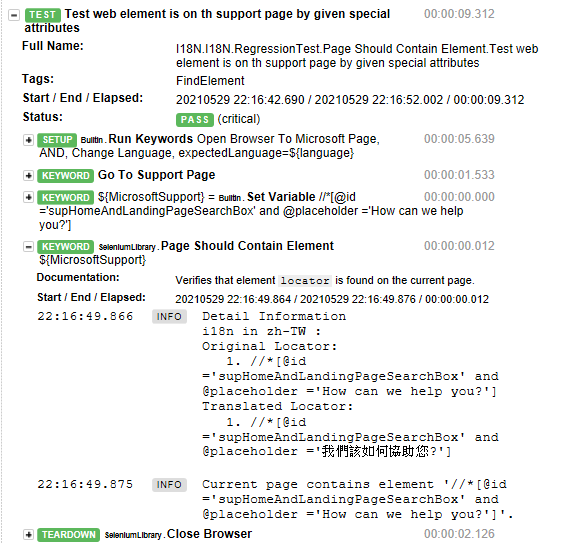


圖4-2-4 待翻譯XPath運行在改善翻譯邏輯後的i18n工具之結果

由以上的結果我們可以發現，第一版的i18n工具，會因為無法對XPath內的@placeholder屬性做翻譯，而使得測試發生錯誤。本論文新版的i18n工具，因為改善了翻譯邏輯，所以支援各種HTML屬性的翻譯。並成功將“how can we help you?”翻譯為”我們該如何協助您”，進而使測試通過。

* 1. **一詞多譯情況下所產生的圖形化使用者介面**

在遭遇一詞多譯且測試腳本會通過的情況下，以Test should be equal的測試腳本為例(如圖4-3-1)；分別創出兩個變數word1、word2，並用Should Be Equal檢驗兩個變數的值是否正確。測試執行結束後，系統會產生一個圖形化使用者介面(如4-3-2)，上面分別記錄了遭遇一詞多譯的關鍵字完整參數，以及所有可能的翻譯詞，供使用者做選擇。



圖4-3-1 Test should be equal 測試腳本



圖4-3-2 遭遇一詞多譯且測試通過，開啟一詞多譯UI

當使用者選擇了希望的翻譯並按下Submit按鈕後，系統便會產生一份設定檔，以利下次執行到同一份腳本的同一關鍵字時，直接套用選擇的翻譯，同時清空一詞多譯介面上的翻譯選項(如圖4-3-3)。

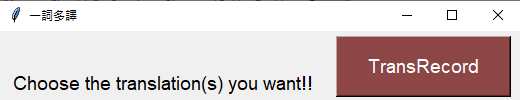


圖4-3-3 按下Submit按鈕後，將翻譯寫進設定檔，同時清除翻譯選項

假如使用者選擇錯誤時，則可以點擊TransRecord按鈕來打開翻譯紀錄的介面，上面記錄著使用者曾經的翻譯選擇，並有一顆Undo按鈕(如圖4-3-4)。使用者選擇了要清除的翻譯紀錄後，按下Undo按鈕，系統便會將設定檔中的該資料刪除，同時關閉翻譯紀錄介面，當再次打開翻譯紀錄介面時，可以發現被選擇清除的資料已消失(如圖4-3-5)。

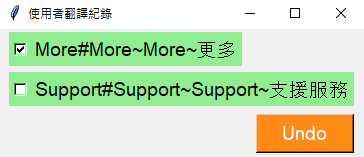


圖4-3-4打開翻譯紀錄介面，上面記錄著使用者翻譯選擇

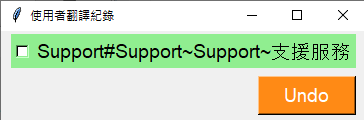


圖4-3-5 清除選擇的翻譯紀錄後，再次打開翻譯紀錄介面

然而，並非所有的測試腳本都會遭遇一詞多譯，且總是通過。因此，底下將分別說明在其他兩種情況下，系統會有甚麼不同的反應:

1. 測試腳本不通過:

若測試腳本不通過，則不論遭遇一詞多譯與否，皆不會跳出一詞多譯UI。其設計想法為，因為系統有檢驗的機制，所以此刻測試腳本不會通過，則代表原先未翻譯前的測試腳本也不會通過。而為原先就會發生錯誤的待翻譯詞選擇其翻譯，對使用者而言，顯然是毫無意義的，但是為了測試結果的完整性，系統還是會於測試報表上顯示一詞多譯的warning資訊(如圖4-3-6)。

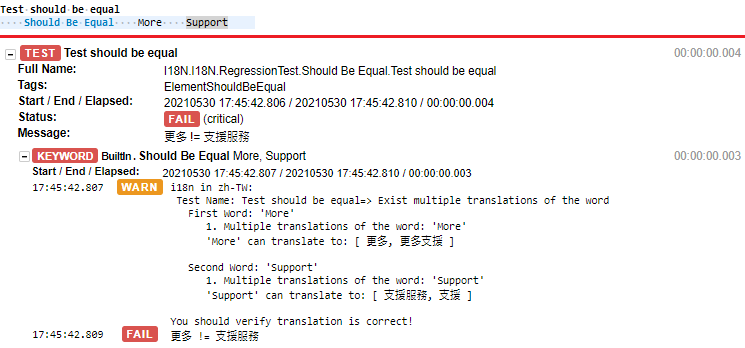


圖4-3-6 測試腳本不通過的情況下，不跳出UI，但會顯示warning資訊於報表

1. 測試腳本通過，且沒有遭遇一詞多譯:

在測試腳本通過且沒有遭遇一詞多譯的情況下，測試結束後便不會跳出一詞多譯UI介面。但有一種情況是，關鍵字的參數部分原本會遭遇一詞多譯，但因為前一次執行時使用者已針對此待翻譯詞選擇了翻譯，所以被系統視為只有一種翻譯，因此不跳出一詞多譯UI(如圖4-3-7)。(有關系統如何判定翻譯已被使用者選擇，詳請參考本論文3-2擴充代理關鍵字的實作說明)

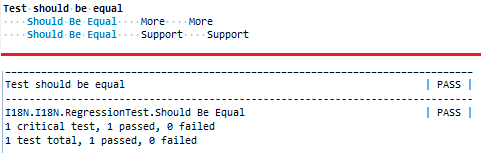


圖4-3-7 測試腳本通過且沒有遭遇一詞多譯的情況下，不跳出UI

* 1. **使用涵蓋多項代理關鍵字的測試腳本**

以上介紹的測試案例，分別呈現了本論文各項功能改善後的實際成果，但因為都偏向單一關鍵字或單一功能的測試，在此將設計一個較複雜的Robot Framework網頁自動化驗收測試腳本，內部使用到多種代理關鍵字，來對Microsoft的中文版網頁進行一系列操作。同時，會遭遇到一詞多譯，並且會使用相別於第一版i18n所列舉的HTML屬性來撰寫XPath。藉此來對新版i18n工具作一個驗收測試。

以下將執行“Test multiple user behaviors on Microsoft website”測試腳本兩次，分別呈現選擇一詞多譯翻譯前後，測試結果有何差異之處。

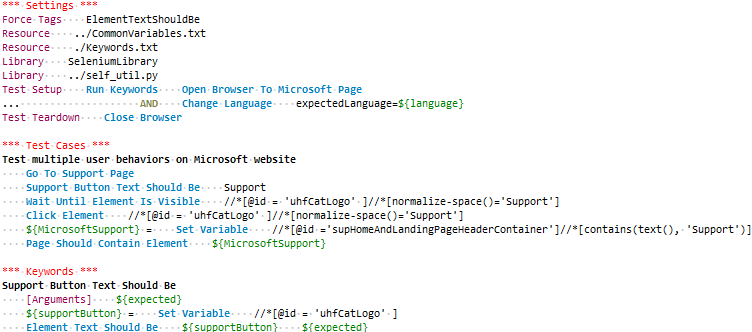


圖4-4-1 Test multiple user behaviors on Microsoft website測試腳本實作

此測試腳本(如圖4.4.1)，會打開Microsoft中文官方網頁，先到支援頁面，之後點擊‘支援’的Logo，到新的支援頁面。最後，驗證畫面上的橫向標語有包含‘支援服務’字樣。因為同時遭遇到3種‘Support’需要被翻譯，系統便會逐一將它們紀錄下來，待測試腳本結束後，分別顯示於一詞多譯UI上。

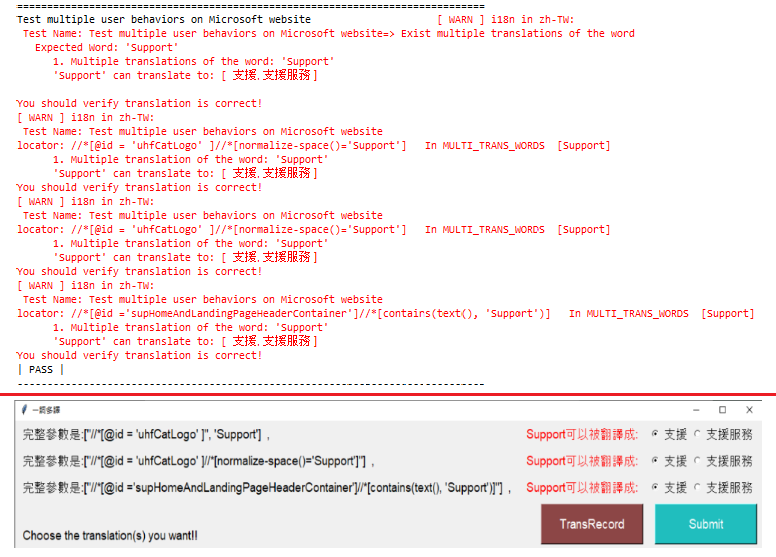


圖4-4-2 Test multiple user behaviors on Microsoft website測試腳本第一次執行結果

第一次執行測試腳本結束後，使用者分別為三種不同情況下遭遇一詞多譯的Support，選擇了支援、支援、支援服務為其正確翻譯，並點擊Submit按鈕，將選擇寫入設定檔中。

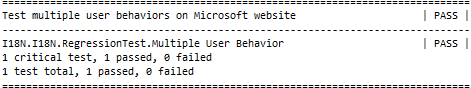


圖4-4-3 Test multiple user behaviors on Microsoft website測試腳本第二次執行結果

第二次執行測試腳本時，會根據設定檔內的翻譯紀錄，分別對三種Support進行翻譯。因為被系統判定為單一翻譯，所以測試結束後，並不會顯示一詞多譯UI。

# 第五章 結論與未來展望

* 1. **結論**

本論文提出了四項措施去改進第一版i18n工具所面臨的問題，分別是擴充代理關鍵字、支援所有html屬性的XPath翻譯邏輯、支援自由選擇翻譯的一詞多譯圖形化使用者介面、可以提供安裝的Python模組。透過以上努力，即可讓使用此i18n工具的多國語言網頁自動化驗收測試腳本，不再侷限於只能使用特定的關鍵字去撰寫。並且XPath的撰寫也不再需要考慮只有@title，text()，normalize-space()會被翻譯的問題，使用者的測試腳本可以用更多元的XPath寫法來搭配此i18n工具使用。遭遇一詞多譯時，使用者也能夠在第一次測試腳本結束後，選擇內心所預期的翻譯詞來執行接下來的測試。最後，藉由將i18n工具包裝成可以pip install 的Python模組，使用者可以隨安裝即用本工具所提供的功能，而無須將自己的測試腳本移動到i18n工具的資料夾下。

* 1. **i18n工具的使用限制與未來展望**

本論文的i18n工具尚存在著使用限制，其原因之一源自於某些Robot Framework的原生關鍵字，其參數部分接受的是特殊的字元，目前i18n工具是不支援的。如圖5.2-1，Get Match Count接受的第二個參數是S\*，用來抓出串列中符合的字串，若是強行將S\*翻譯，便會直接導致測試結果出錯。

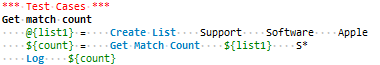


圖5.2-1 Get Match Count測試腳本

此外，某些Robot Framework的關鍵字，其內部實作會呼叫另一個原生的關鍵字，且被呼叫的關鍵字也有一詞多譯的需求。當使用者為第一次執行測試的一詞多譯選擇了翻譯後，並執行第二次測試腳本時，因為此時傳入代理關鍵字的參數部分已改變，系統會將其判定為不同的待翻譯詞，而需要使用者事後再選擇一次。如圖 5.2-2 ，Dictionary Should Contain Item原生關鍵字的實作會呼叫Dictionary Should Contain Key(紅色框起部分)，導致第一次執行測試時傳入的”More”，和第二次執行時傳入的”More”，由於參數紀錄的局部從[‘支援’, ‘支援服務’]變成[‘支援’]，因此被系統判定成不同的待翻譯詞，需要重新選擇(如圖5.2-3)。

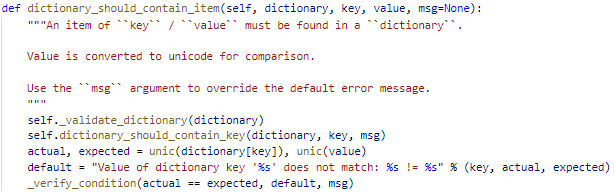


圖5.2-2 Dictioanry Should Contain Item測試腳本與其原生實作



圖5.2-3 第一次與第二次執行測試腳本，待翻譯詞的參數紀錄變化

未來展望的部分，可以根據i18n工具尚存在的使用限制，未來朝向優化特殊字元的翻譯方式，以及改善重複翻譯詞的判定邏輯去努力。而且目前JSON翻譯檔還需要由使用者提供，未來可以研究如何透過設計爬蟲，抓取多國語言網頁上的對應翻譯詞，自動將JSON翻譯檔產生出來，將人工的成本降低。記錄翻譯詞的部分，目前也只有記錄關鍵字的參數，少數情況下可能會和其他的關鍵字發生重疊，未來可以將關鍵字名稱一併記錄下來，以提高處理一詞多譯時的精確性。進而讓「多國語言網頁自動化驗收測試」工具變得更貼近測試者的需要，錯誤率低且容易使用。**參 考 文 獻**

1. 李允中(2013)。軟體工程。臺北：社團法人臺灣軟體工程學會。194頁。
2. 呂昭陞(2018)。一個改善XPath表達式在網頁應用程式自動化驗收測試之穩定

性的方法。碩士論文。臺北，國立臺北科技大學資訊工程系碩士班。

1. 朱峻平(2019)。[支援多國語言的Robot Framework網頁自動化驗收測試](https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=78srsD/record?r1=1&h1=0)。碩士

論文。臺北，國立臺北科技大學資訊工程系碩士班。

1. [Chin-Yun Hsieh](https://ieeexplore.ieee.org/author/38258708400), [Chen-Hsin Tsai](https://ieeexplore.ieee.org/author/37073881500), [Yu Chin Cheng](https://ieeexplore.ieee.org/author/38226238700). (2013, May).

Test-Duo: A framework for generating and executing automated acceptance tests from use cases. Paper presented at the 8th International Workshop on Automation of Software Test (AST). San Francisco, CA, USA.

1. INTERNATIONALIZATION.

Retrieved from <https://www.w3.org/standards/webdesign/i18n>.

1. JSON – Introduction.

Retrieved from <https://www.w3schools.com/js/js_json_intro.asp>.

1. Robot Framework.

Retrieved from [https://robotframework.org/.](https://robotframework.org/)

1. Robot Framework User Guide. Retrieved from

<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide>.

1. Python official website.

Retrieved from <https://www.python.org/>.

1. Microsoft official website.

Retrieved from <https://www.microsoft.com/>.

1. XPath Tutorial.

Retrieved from <https://www.w3schools.com/xml/xpath_intro.asp>.

1. Python PIP.

Retrieved from <https://www.w3schools.com/python/python_pip.asp>.

1. Automation Testing Tutorial: What is Automated Testing?

Retrieved from [https://www.guru99.com/automation-testing.html#1](https://www.guru99.com/automation-testing.html%231).

1. Proxy.

Retrieved from <https://refactoring.guru/design-patterns/proxy>.

1. RED – Robot Editor.

Retrieved from <https://github.com/nokia/RED>.

1. PYPI official website.

Retrieved from <https://pypi.org/>.

1. Choose an open source license.

Retrieved from <https://choosealicense.com/>.

1. Unit Testing. Retrieved from

<https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm>

1. Node.js official website.

Retrieved from <https://nodejs.org/en/>.