國立臺北科技大學

資訊工程系碩士班

碩士論文

編號：107598051

改善”支援多國語言的Robot Framework

網頁自動化驗收測試工具”的翻譯邏輯、一詞多譯、代理關鍵字，並發佈使其完整

研究生：林稟宸



**資訊工程系碩士班**

**碩士學位論文**

**改善**

**”支援多國語言的Robot Framework網頁自動化驗收測試工具” 的翻譯邏輯、一詞多譯功能、代理關鍵字，並發佈使其完整**

**Improve Translation Logic, Multi-Translations Function, and Proxies of “Automated Web Testing Tool with Robot Framework which Support Internationalization“, and Release It**

**研究生：林稟宸**

**指導教授：鄭有進 教授、謝金雲 教授**

**中華民國一百一零年六月**

**摘要**

論文名稱：改善”支援多國語言的Robot Framework網頁自動化驗收測試工具”的翻譯邏輯、一詞多譯功能、代理關鍵字，並發佈使其完整

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，測試者因此被迫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，例如:

1. 目前i18n工具只支援8種Robot Framework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. 測試腳本運行期間，只要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。使用者無法自己選擇預期的翻譯詞去跑腳本測試。

此外，假如存在多個翻譯詞都會通過此測試(可能是因為xpath使用contains，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。

1. 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。

因此，此篇論文我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：Improve Translation Logic, Multi-Translations Function, and Proxies of “Automated Web Testing Tool with Robot Framework which Support Internationalization“, and Release It

Pages：

School：National Taipei University of Technology

Department：Computer Science and Information Engineering

Time：June, 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 object of one acceptance test script based on Robot Framework was often limited to web page in one language. However, nowadays a web page may have multiple language versions for people live in different countries. Therefore, tester often need to write more repetitive test scripts, and the cost of testing will 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 translation mapping between words and their translations by JSON format translation files given by user. Besides, translation logic and proxies of keywords also let us can run the same web page acceptance test in different languages by the same test script without changing a word.

However, the current i18n tool still have many flaws to be improved, such as:

1. Currently, the i18n tool only support 8 Robot Framework native keywords. If other unsupported Robot Framework native keywords are used in the future, it may cause an error.
2. The current translation target of I18n tool are limited to text(), normalize-space(), and @title on the web page. So a test script will go wrong if it is written with attributes that are not listed above.
3. If a word have multiple translations currently, i18n tool only lists the possible translations on the test report, and selects the translation that would pass the test as the current solution. The user cannot choose the expected translation by himself to run the test script.

In addition, if there exist many translations would pass the test (maybe due to xpath is written by ‘contains()’, as long as the test target is included in the xpath, the test will pass. Or there are multiple components on the web page that match the translation), the way we take currently may deviate from the original expectation of the user.

1. At present, installation is not supported by the i18n tool, so users can only clone the project from github and develope their own test scripts of Robot Framework inside the project. This is not user friendly.

Therefore, I am going to improve the defects of the I18n tool that I mentioned above, hoping that the i18n tool will be easier to use 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，盡力在細節處做到最好，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心中的一個選項。

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](#_Toc73113693)

[**ABSTRACT** iii](#_Toc73113694)

[**誌 謝** v](#_Toc73113695)

[**目 錄** vi](#_Toc73113696)

[**表目錄** ix](#_Toc73113697)

[**圖目錄** x](#_Toc73113698)

[**第一章 緒論** 1](#_Toc73113699)

[**1.1 研究背景與動機** 1](#_Toc73113700)

[**1.2 研究目標** 1](#_Toc73113701)

[**1.3 論文組織架構** 2](#_Toc73113702)

[**第二章 背景知識** 3](#_Toc73113703)

[**2.1 國際化** 3](#_Toc73113704)

[**2.2 自動化驗收測試** 3](#_Toc73113705)

[**2.3 Robot Framework** 3](#_Toc73113706)

[**2.3.1 Robot Framework測試腳本** 3](#_Toc73113707)

[**2.3.2 Robot Framework測試報表** 4](#_Toc73113708)

[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(通過或失敗)以及時間，透過可展開的階層式圖表，可以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。 4](#_Toc73113709)

[**2.4 XPath** 4](#_Toc73113710)

[**第三章 研究方法與實作** 5](#_Toc73113711)

[**3.1 系統設計** 5](#_Toc73113712)

[**3.1.1 第一版i18n工具的現存類別與原生類別** 6](#_Toc73113713)

[Robot Framework的原生類別分別有RobotFramework，Listener，和Report。 6](#_Toc73113714)

[第一版i18n設計的類別則有I18nListener，MappingRoutesGenerator，I18nMap，I18nTrigger，Proxy，以及五個繼承Proxy的代理關鍵字類別: ElementTextShouldBeProxy，FindElementProxy，ListsShouldBeEqualProxy，ShouldBeEqualProxy，ShouldContainProxy。 6](#_Toc73113715)

[**3.1.2 本論文新擴充的類別** 6](#_Toc73113716)

[表格 3-1 本論文新擴充的類別 6](#_Toc73113717)

[**3.1.3 系統執行流程與比較** 10](#_Toc73113718)

[**3.2 擴充代理關鍵字使其完整支援原生關鍵字** 11](#_Toc73113719)

[**3.3 改善翻譯邏輯使其能應對各種html屬性** 12](#_Toc73113720)

[**3.4 提供圖形化使用者介面解決一詞多譯的問題** 13](#_Toc73113721)

[**3.5 將i18n工具設計成為可以安裝的模組** 16](#_Toc73113722)

[本論文透過Python的build模組，將i18n工具包裝成為一個名為”RF-AWT-i18n”的Library，並透過twine模組上傳至Pypi網站上，成為了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。 16](#_Toc73113723)

[以下將詳述實際的步驟: 16](#_Toc73113724)

[在將i18n工具打包成為一個Library前，必須先準備以下四個檔案: setup.cfg、pyproject.toml、README.md、LICENSE。此外必須確認將打包的資料夾下，包含\_\_init\_\_.py這個檔案，以被辨識為一個Python模組。 16](#_Toc73113725)

[圖3-5-1 setup.cfg實作 16](#_Toc73113726)

[setup.cfg內部設定name(模組名)、version(版本)、author(作者)、author\_mail(信箱)、description(模組描述)、long\_description(README.md的詳細說明)、long\_description\_content\_type(README.md的文件格式)、url(專案的網址)、classifiers(程式語言、憑證種類、作業系統)、packages=find:(透過find找出當前路徑的packages)、python\_require(Python需求版本)、include\_package\_data(是否包含其他類型檔案於package中)。 16](#_Toc73113727)

[圖3-5-2 pyproject.toml實作 16](#_Toc73113728)

[pyproject.toml 設定建構系統所需要的setuptools版本與wheel。 16](#_Toc73113729)

[圖3-5-3 README.md實作 16](#_Toc73113730)

[README.md 內部撰寫了關於此i18n工具的教學文件，讓使用者能夠更快的上手，使用此工具 16](#_Toc73113731)

[圖3-5-4 LICENSE實作 16](#_Toc73113732)

[此LICENSE是屬用MIT LICENSE，詳請參考choosealicense.com 16](#_Toc73113733)

[**第四章 測試案例分析** 17](#_Toc73113734)

[**4.1 新增與修改之代理關鍵字的單元測試** 17](#_Toc73113735)

[**4.2 改善翻譯邏輯後的驗收測試示例** 19](#_Toc73113736)

[**4.3 一詞多譯情況下所產生的圖形化使用者介面** 19](#_Toc73113737)

[**4.4 使用多項代理關鍵字的測試腳本** 20](#_Toc73113738)

[**第五章 結論與未來展望** 21](#_Toc73113739)

[**5.1 本論文限制** 21](#_Toc73113740)

[**5.2 結論** 21](#_Toc73113741)

[**5.3 未來展望** 23](#_Toc73113742)

**表目錄**

[表格 2‑1網頁前端框架分析 3](#_Toc42761886)

[表格 3‑1原生類別 11](#_Toc42761887)

[表格 3‑2本論文方法所設計的類別 11](#_Toc42761888)

[表格 3‑3目前已提供代理的關鍵字 24](#_Toc42761889)

[表格 3‑4尚未提供代理的關鍵字 25](#_Toc42761890)

**圖目錄**

[圖2‑1 angular-translate-英文翻譯檔 4](#_Toc42761891)

[圖2‑2 angular-translate-繁體中文翻譯檔 4](#_Toc42761892)

[圖2‑3 HTML內加入標記 4](#_Toc42761893)

[圖2‑4 Robot Framework腳本 7](#_Toc42761894)

[圖2‑5擴充測試函式庫(self\_util.py) 7](#_Toc42761895)

[圖2‑6 Robot Framework產生之報表 8](#_Toc42761896)

[圖2‑7 Robot Framework Listener version 9](#_Toc42761897)

[圖3‑1系統類別圖 10](#_Toc42761898)

[圖3‑2 Robot Framework Listener參數設置圖 13](#_Toc42761899)

[圖3‑3英文翻譯檔範例 15](#_Toc42761900)

[圖3‑4繁體中文翻譯檔範例 15](#_Toc42761901)

[圖3‑5翻譯查找路徑對照表 16](#_Toc42761902)

[圖3‑6建立翻譯查找路徑對照表循序圖 17](#_Toc42761903)

[圖3‑7 value method 19](#_Toc42761904)

[圖3‑8 get\_possible\_translation method 19](#_Toc42761905)

[圖3‑9 values method 20](#_Toc42761906)

[圖3‑10 translate method 20](#_Toc42761907)

[圖3‑11代理關鍵字Should Be Equal的循序圖 22](#_Toc42761908)

[圖3‑12代理關鍵字的擴充介面 23](#_Toc42761909)

[圖3‑13 xpath翻譯過後的資訊 26](#_Toc42761910)

[圖3‑14單字翻譯過後的資訊 27](#_Toc42761911)

[圖3‑15報表上方欄提示xpath內單字具有一詞多譯問題 28](#_Toc42761912)

[圖3‑16詳細提示xpath內單字具有一詞多譯問題 28](#_Toc42761913)

[圖3‑17報表上方欄提示參數具有一詞多譯問題 30](#_Toc42761914)

[圖3‑18詳細提示參數具有一詞多譯問題 30](#_Toc42761915)

[圖3‑19消除一個單字的一詞多譯warning提示之參數設定 31](#_Toc42761916)

[圖3‑20消除多個單字的一詞多譯warning提示之參數設定 31](#_Toc42761917)

[圖3‑21使用檔案消除多個單字的一詞多譯warning提示之參數設定 32](#_Toc42761918)

[圖3‑22消除xpath內單字一詞多譯warning提示 32](#_Toc42761919)

[圖3‑23消除參數一詞多譯warning提示 33](#_Toc42761920)

[圖4‑1 xpath含有待翻譯單字More的自動化測試腳本 34](#_Toc42761921)

[圖4‑2單字More顯示於英文頁面 35](#_Toc42761922)

[圖4‑3單字More顯示於繁體中文頁面 35](#_Toc42761923)

[圖4‑4關鍵字(Keyword) Get Text After It Is Visible 36](#_Toc42761924)

[圖4‑5 xpath含有待翻譯單字More的報表資訊呈現 36](#_Toc42761925)

[圖4‑6關鍵字使用到待翻譯單字More的自動化測試驗證腳本 37](#_Toc42761926)

[圖4‑7關鍵字使用到待翻譯單字More報表資訊呈現 37](#_Toc42761927)

[圖4‑8具有一詞多譯問題-英文頁面 38](#_Toc42761928)

[圖4‑9具有一詞多譯問題-繁體中文頁面 39](#_Toc42761929)

[圖4‑10 xpath內單字Support具有一詞多譯問題的自動化測試腳本 39](#_Toc42761930)

[圖4‑11報表上方欄提示xpath內單字Support具有一詞多譯問題 40](#_Toc42761931)

[圖4‑12詳細提示xpath內單字Support具有一詞多譯問題 40](#_Toc42761932)

[圖4‑13關鍵字(Keyword)參數Support具有一詞多譯問題的自動化測試腳本 41](#_Toc42761933)

[圖4‑14報表上方欄提示參數Support具有一詞多譯問題 42](#_Toc42761934)

[圖4‑15詳細提示參數Support具有一詞多譯問題 42](#_Toc42761935)

[圖4‑16消除單字Support的一詞多譯warning提示之參數設定 43](#_Toc42761936)

[圖4‑17消除xpath內單字Support一詞多譯warning提示 43](#_Toc42761937)

[圖4‑18消除參數Support一詞多譯warning提示 43](#_Toc42761938)

**第一章 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)，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，以下分別列出四點: 支援的Robot Framework原生關鍵字只有八種、翻譯邏輯只支援三種html屬性、一詞多譯無法讓使用者自己選擇、該工具尚未發佈並支援安裝。

* 1. **研究目標**

本論文旨在改善i18n工具上述提及的四項缺陷，希望透過我的努力，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時，更易於使用。

而實作的目標分別列舉如下:

1. Robot framework版本剩下的代理關鍵字，使得所有常用原生關鍵字都能正確的翻譯
2. 透過”負面表列法”，將xpath中確定不會執行翻譯的屬性(例如:@id , @class等等)儲存於list中，若腳本執行到當下的xpath需要被翻譯，且其中有屬性不在list中，則利用這些屬性們生成一個新的翻譯規則，再執行翻譯。
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 install 的library
   1. **論文組織架構**

本論文分為五個章節，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，記錄做了上述改動後，使用者執行網頁自動化驗收測試時，和先前的i18n工具相比，有甚麼差異之處。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。

1. **第二章 背景知識**
   1. **國際化**

國際化，英文讀作Internationalization，簡稱i18n，18意謂著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，英文讀作Acceptance Testing，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，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。改善了過往手動驗收測試存在的人為操作誤差、成本較高、系統問題無法即時呈現等問題。

* 1. **Robot Framework**

Robot Framework 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(test case執行前要做的動作)、Test Teardown(test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: 如果測試腳本即將使用並非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。
   * 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(通過或失敗)以及時間，透過可展開的階層式圖表，可以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. **XPath**

XPath，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. **第三章 研究方法與實作**

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)進行比較，呈現出先後的差異。

* 1. **系統設計**

圖3‑1-1為第一版i18n工具的系統類別圖，經過部分功能改善，並為其擴充代理關鍵字後，使其成為圖3-1-2當前版本的i18n系統類別圖。

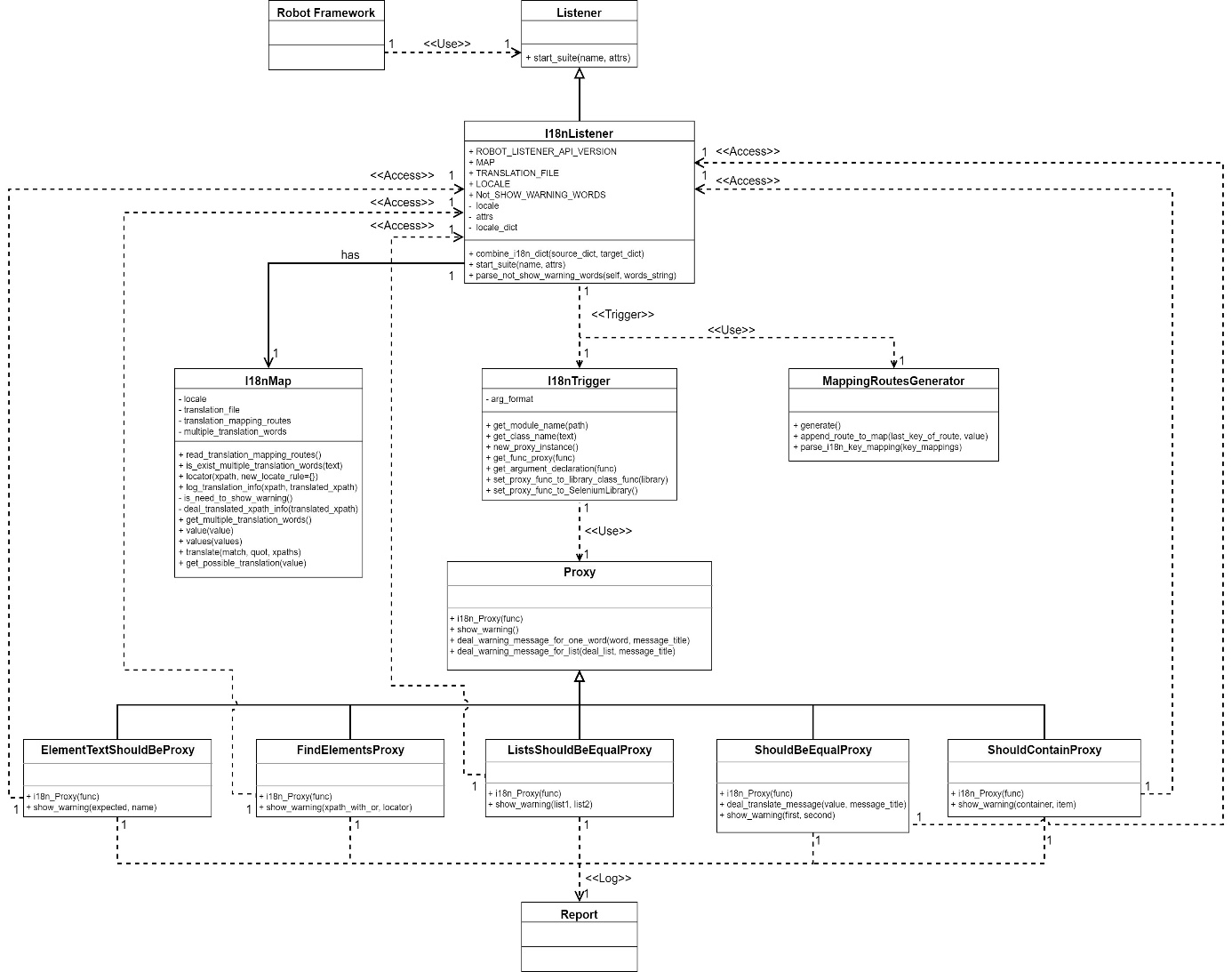


圖3‑1-1第一版i18n系統類別圖

圖3-1-2 當前版本i18n系統類別圖

以下將以概述的方式，介紹Robot Framework的原生類別，與i18n第一版已存在的類別(詳細可參考朱峻平的論文3-1)。並以表格3-1的方式呈現本論文所新增加的類別。

* + 1. **第一版i18n工具的現存類別與原生類別**

Robot Framework的原生類別分別有RobotFramework，Listener，和Report。

第一版i18n設計的類別則有I18nListener，MappingRoutesGenerator，I18nMap，I18nTrigger，Proxy，以及五個繼承Proxy的代理關鍵字類別: ElementTextShouldBeProxy，FindElementProxy，ListsShouldBeEqualProxy，ShouldBeEqualProxy，ShouldContainProxy。

* + 1. **本論文新擴充的類別**

表 3-1-2-1 本論文新擴充的類別

|  |  |
| --- | --- |
| **類別名稱** | **類別功能** |
| UI | 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UI介面供使用者選擇。 |
| AlertShouldBePresentProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Alert Should Be Present”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| CountValuesInListProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Count Values In List”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| DictionariesShouldBeEqualProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Dictionaries Should Be Equal”和”Dictionary Should Contain Sub Dictionary”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| DictionaryShouldContainItemProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Dictionary Should Contain Item”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| DictionaryShouldContainKeyProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Dictionary Should Contain Key”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。”Dictionary Should Contain Item”其內部實作也會使用到此Proxy。 |
| DictionaryShouldContainValueProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Dictionary Should Contain Value”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| GetMatchCountProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Get Match Count”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| ListSelectionShouldBeProxy | 繼承自父類別Proxy並改寫其實作，負責代理”List Selection Should Be”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| ListShouldContainSubListProxy | 繼承自父類別Proxy並改寫其實作，負責代理”List Should Contain Sub List”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| ListShouldContainValueProxy | 繼承自父類別Proxy並改寫其實作，負責代理”List Should Contain Value”和”List Should Not Contain Value”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| ListShouldNotContainDuplicatesProxy | 繼承自父類別Proxy並改寫其實作，負責代理”List Should Not Contain Duplicates”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| RemoveFromDictionaryProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Remove From Dictionary”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| RemoveValuesFromListProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Remove Values From List”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| SelectFromListByLabelProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Select From List By Label”, “Unselect From List By Label”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| SelectFromListByValueProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Select From List By Value”, “Unselect From List By Value”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| TableCellShouldContainProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Table Cell Should Contain”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| TableColumnShouldContainProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Table Column Should Contain”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| TableRowShouldContainProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Table Row Should Contain”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| TableShouldContainProxy | 繼承自父類別Proxy並改寫其實作，負責代理”Table Should Contain”,”Table Header Should Contain”,“Table Footer Should Contain”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |
| TitleShouldBeProxy | 繼承自父類別Proxy並改寫其實作，負責代理“Title Should Be”關鍵字的參數翻譯邏輯，並將翻譯好的參數部分回傳給原關鍵字。 |

* + 1. **系統執行流程與比較**

第一版的i18n其執行流程，首先使用者必須於Red編輯器中的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。

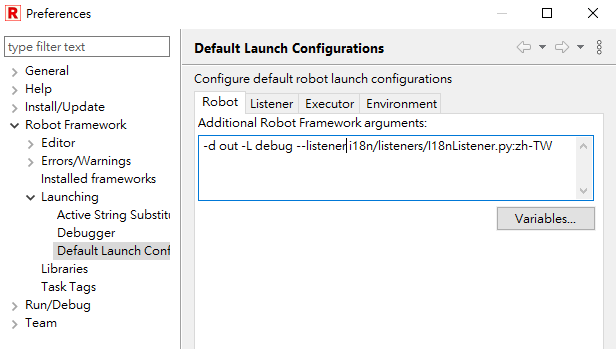


圖3-1-3-1 Robot Framework系統參數設定

測試腳本執行前，系統會去呼叫I18nListener.py，當前語言zh-TW代表繁體中文-台灣。之後讀取由使用者自行提供的JSON格式翻譯檔(圖3-1-3-2)，建立出一份”Key階層->翻譯”的翻譯路徑檔(圖3-1-3-3)，以及建立出所有自定義的代理關鍵字物件。測試腳本執行時，每當運行到一個有定義代理之關鍵字，就會去呼叫對應的代理關鍵字物件，並執行翻譯邏輯。測試腳本結束後，將一詞多譯的warning資訊顯示在報表上。若想隱藏特定的翻譯詞warning，可以事後在手動在組態設定。



圖3-1-3-2 JSON格式翻譯檔範例

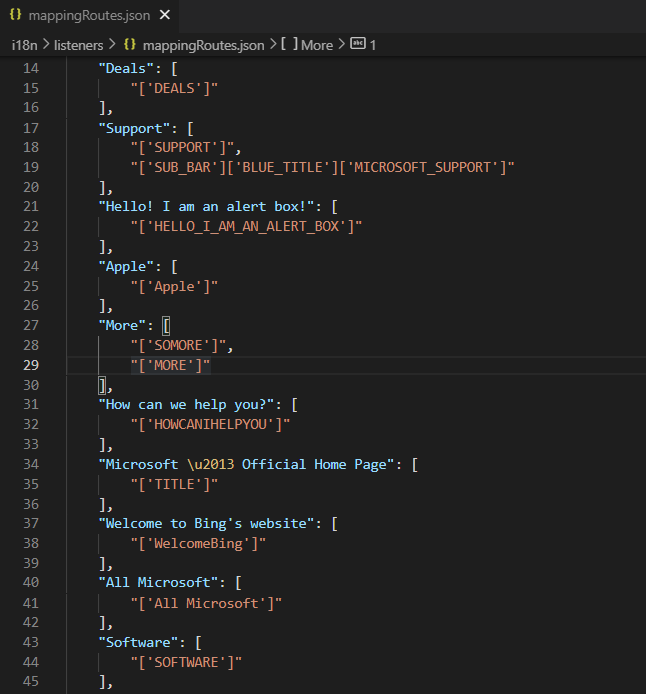


圖3-1-3-3 翻譯路徑檔部分示例

新版的i18n執行流程，基本沿用了第一版的架構，但擴充了代理關鍵字(詳見3-2)，也優化了翻譯邏輯(詳見3-3)。

而改變較大的部份，是在測試腳本結束後，如果有遭遇過一詞多譯，便會跳出一詞多譯的圖形化使用者介面(詳見3-4)，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**

在此前，第一版i18n工具只支援7種Robot Framework原生關鍵字(如表3-2-1)。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，使得所有原生關鍵字的參數部分都能正確得被翻譯。

|  |
| --- |
| 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。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)，因此只剩下29種原生關鍵字需要提供代理(如表3-2-2)(代理關鍵字功能詳請參考表3-1-2-1)。(\*標示為版本更新導致被淘汰的原生關鍵字)

|  |  |
| --- | --- |
| 當前i18n版本將擴充代理的關鍵字 | |
| Collections Library (3.0.4) | 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 當前i18n版本將擴充代理的Robot Framework原生關鍵字

以下將以 FindElementsProxy為例，詳述在現今配合一詞多譯UI的架構下，我是如何完成修改或擴充代理關鍵字的，以及和第一版i18n代理關鍵字的實作差異。

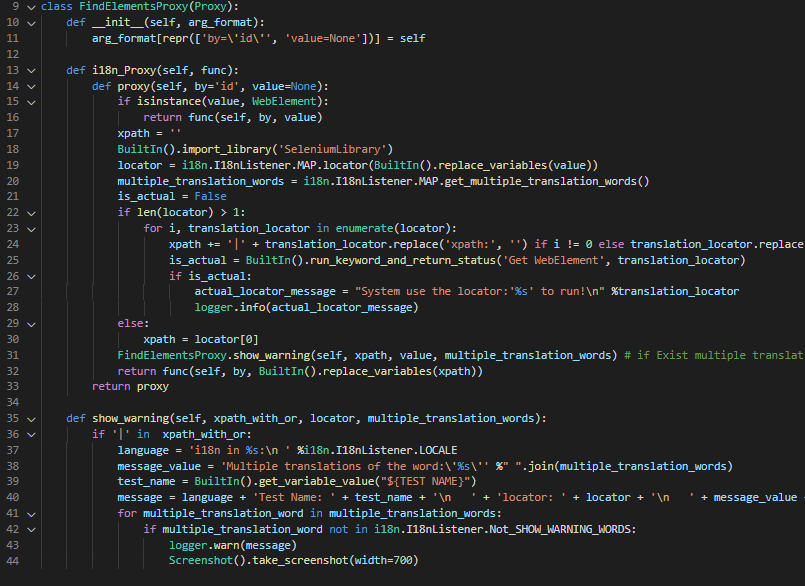


圖3-2-1 第一版i18n 的FindElementsProxy程式結構

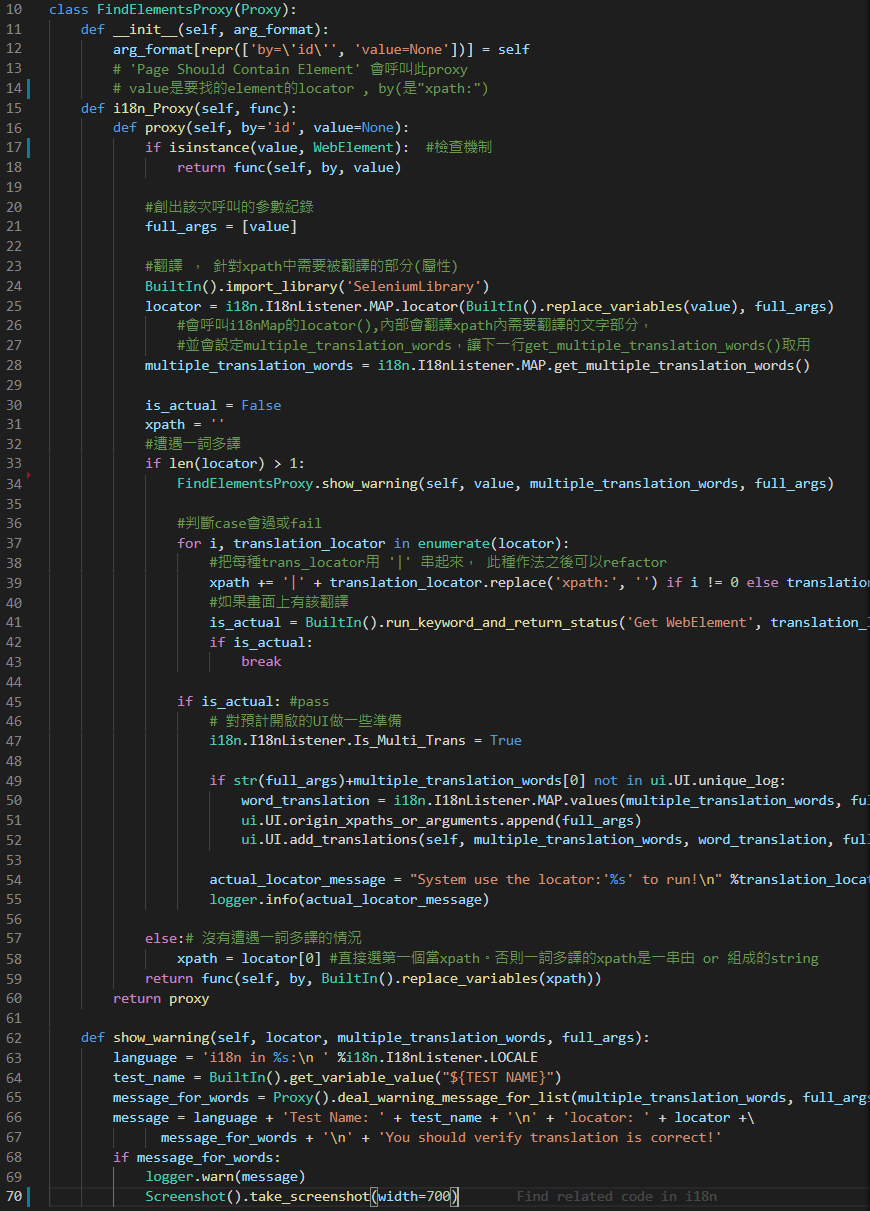


圖3-2-2 當前i18n版本的FindElementsProxy程式結構

由圖3-2-1與圖3-2-2的程式可以發現，建構於原本的i18n\_Proxy()之上，兩者的proxy()內部實作與結構存在著部分差異之處。

現今的i18n版本，擴充一個新的代理關鍵字必須遵從著以下步驟:

1. 創出該次關鍵字呼叫的參數紀錄，如21行的full\_args = [value]，此參數紀錄之後會顯示於一詞多譯UI上，並隨著使用者的選擇一併存入設定檔內。之後便會成為判斷該翻譯詞是否”執行一詞多譯” 的依據之一，又或是”已存在設定過的翻譯”。
2. 執行翻譯，此為沿用第一版的作法，如25行的

locator = i18n.I18nListener.MAP.locator(BuiltIn().replace\_variables(value), full\_args)， 差別是現在必須代入full\_args以讓系統判斷該待翻譯詞是否已經存在於設定檔的 翻譯選項中。

1. 若判定為遭遇一詞多譯，則產生warning資訊於報表上，如34行的

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37~43行，以作為之後是否開啟一詞多譯UI的依據。因為假使測試原本就會失敗，那麼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。
2. 若測試會通過，則對預計開啟的UI做一些資料的準備，如47行的

i18n.I18nListener.Is\_Multi\_Trans=True會讓測試結束後，去開啟一詞多譯UI。

49~52行，49行if str(full\_args)+multiple\_translation\_words[0] not in ui.UI.unique\_log:

首先去判斷該”參數+翻譯詞”組合是否已經被翻譯過，

之後51行 ui.UI.origin\_xpaths\_or\_arguments.append(full\_args) 記錄下該次翻譯的參數部分，52行 ui.UI.add\_translations(self, multiple\_translation\_words, word\_translation, full\_args)則記錄下此次的待翻譯詞和翻譯選項。

1. 考慮沒有遭遇一詞多譯的情況，如58行的 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。
2. 將整理好的翻譯後參數部分回傳給原生的Robot Framework原生關鍵字，如59行的

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

1. 定義每個代理關鍵字個別的show\_warning()函式，如62~70行。和圖3-2-1的36行做比較，可以發現由於上方proxy()內部實作的改變，在呼叫show\_warning()前已先判斷遭遇一詞多譯，所以如今不再需要傳入xpath\_with\_or參數去做判斷了。

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(如圖3-3-1的紅色框起部分)，僅提供了@title、text()、normalize-space()三種屬性的翻譯規則。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。

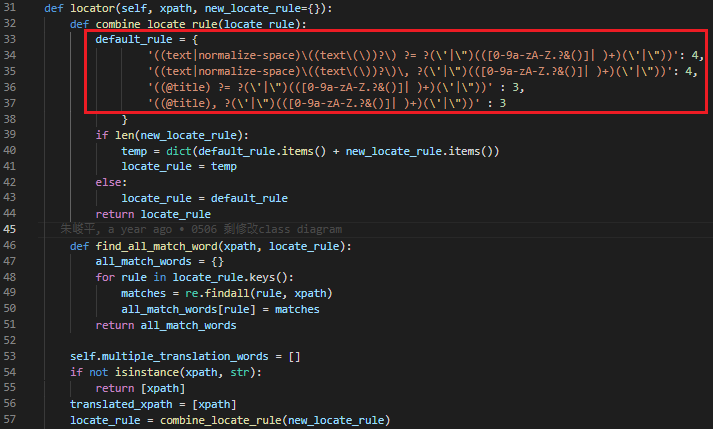


圖3-3-1 第一版i18n的XPath翻譯邏輯

為了解決以上問題，本論文早期設想了三種解決方向:

1. 設計一個新函式，每當遭遇html屬性時，都去執行翻譯，檢查看看。缺點:耗時。

2. 找出待翻譯屬性們之間的共通點，讓i18n程式自行判斷是否翻譯。缺點:此屬用AI範 疇，實作不易

3. 負面表列法，將確定不會執行翻譯的屬性在程式中用list的方式儲存，若XPath的屬性不在該list中，則執行1.的方法

在多方優缺點的考量下，本論文最後採用了第三種”負面表列法”去解決問題，因為此種作法相較於第一種解法顧及到了系統執行效能，來的更加省時。相較於第二種類似於AI的作法，在實作的複雜度上，則更為簡單，且把是否翻譯的決定權交給了測試者，而非全部靠系統。相較於原本”把翻譯規則用列舉法固定”相比，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。

以下將詳述新版i18n工具改善後的翻譯邏輯(如圖3-3-2)，並附上翻譯邏輯的循序圖(如圖3-3-3)以供參考。

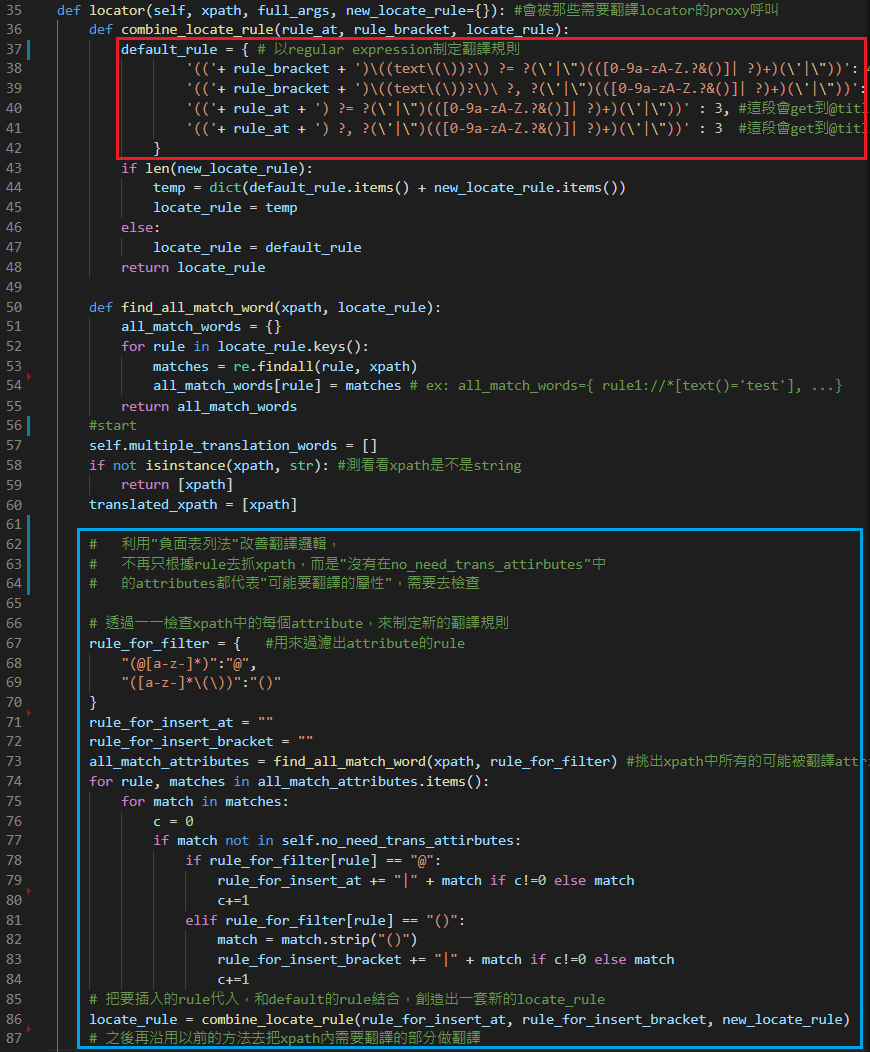


圖3-3-2 當前i18n版本改善後的翻譯邏輯

圖中37~42行紅色框起部分的default\_rule，現在以rule\_bracket 和 rule\_at取代原本的text()，normalize-space()，@title三種屬性。

62~87行藍色框起部分為此次新增的翻譯邏輯，67~70行定義了待翻譯屬性的格式，並由73行取出該XPath中所有可能需要翻譯的屬性。並執行74~84行，判斷這些屬性是否存在於no\_need\_trans\_attributes中，若不存在，則根據屬性含有 ‘@’ 或 ‘()’，分別分配給rule\_for\_insert\_at 和 rule\_for\_insert\_bracket。最後執行86行，把過濾出要翻譯的屬性代入，和37~42行的defult\_rule結合，成為一個新的locate\_rule。如此後面的實作便能沿用以前的方法把XPath內需要的部分做翻譯。

圖3-3-3當前i18n版本翻譯邏輯的Sequence diagram

* 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。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。

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。

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，將詳述一詞多譯UI的介面設計與整體實作，並附上一詞多譯UI的循序圖，以供比對:

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，上方紅色框起區塊為以條列的方式呈現完的整翻譯資訊，包含完整參數部分、待翻譯詞、可能的翻譯。下方藍色框起區塊包含一個提示功能資訊的Label、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。

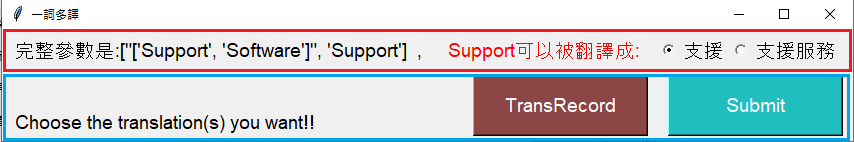


圖3-4-1一詞多譯UI的介面設計

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，上方紅色框起區塊以條列式呈現設定檔中現存的使用者翻譯選擇，下方藍色框起區塊包含一個Undo按鈕，用於將使用者的選擇從設定檔中刪除。

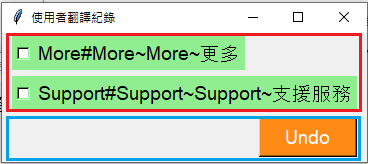


圖3-4-2 翻譯紀錄的介面設計

一詞多譯UI的實作部分則大致可以分成7個部分: run()、draw\_trans\_options()、get\_transdic\_keys\_and\_values()、open\_record()、undo\_trans()、output\_setting\_file()、add\_translations()，如圖3-4-3 ~ 3-4-5。

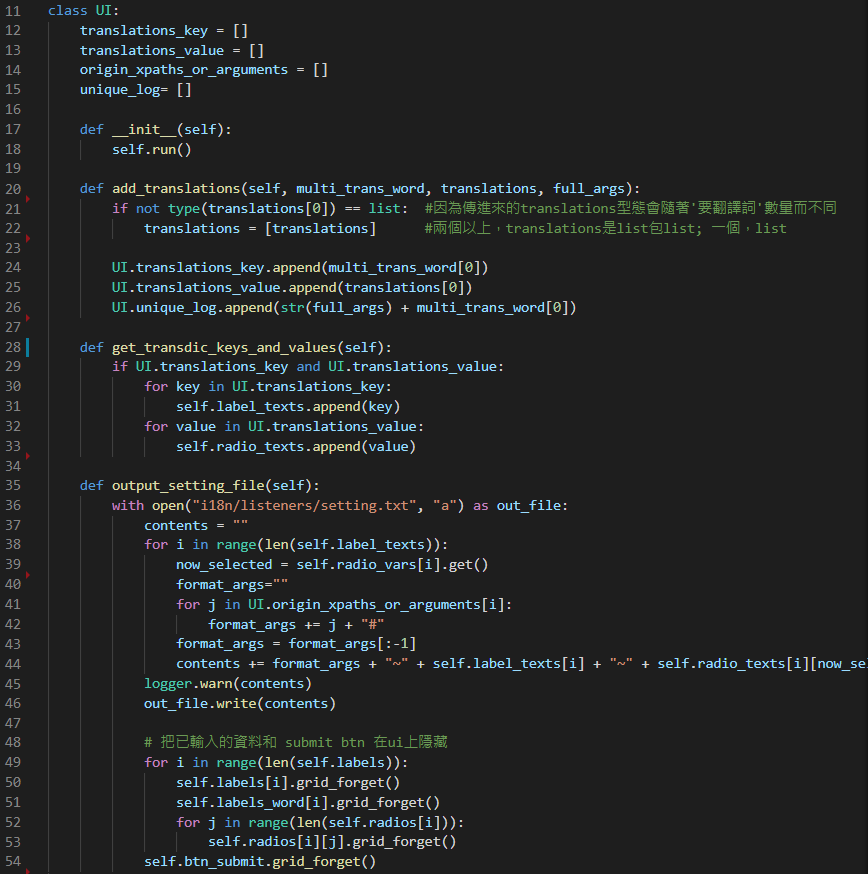


圖3-4-3 一詞多譯UI的程式實作(1)

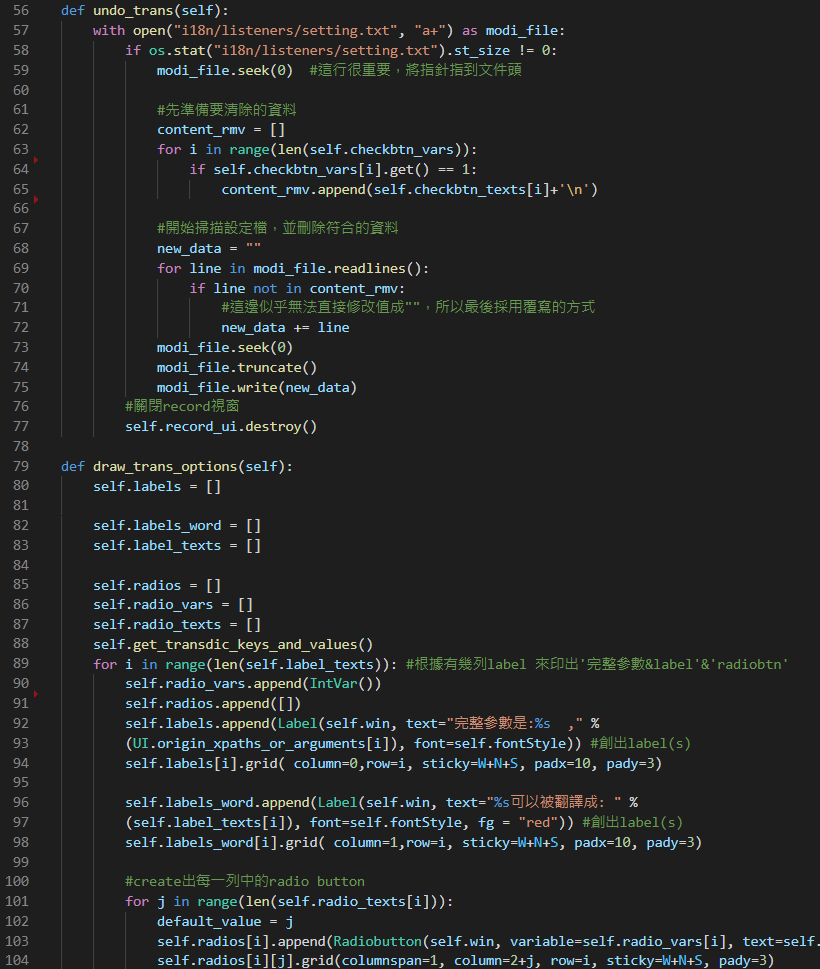


圖3-4-3 一詞多譯UI的程式實作(2)

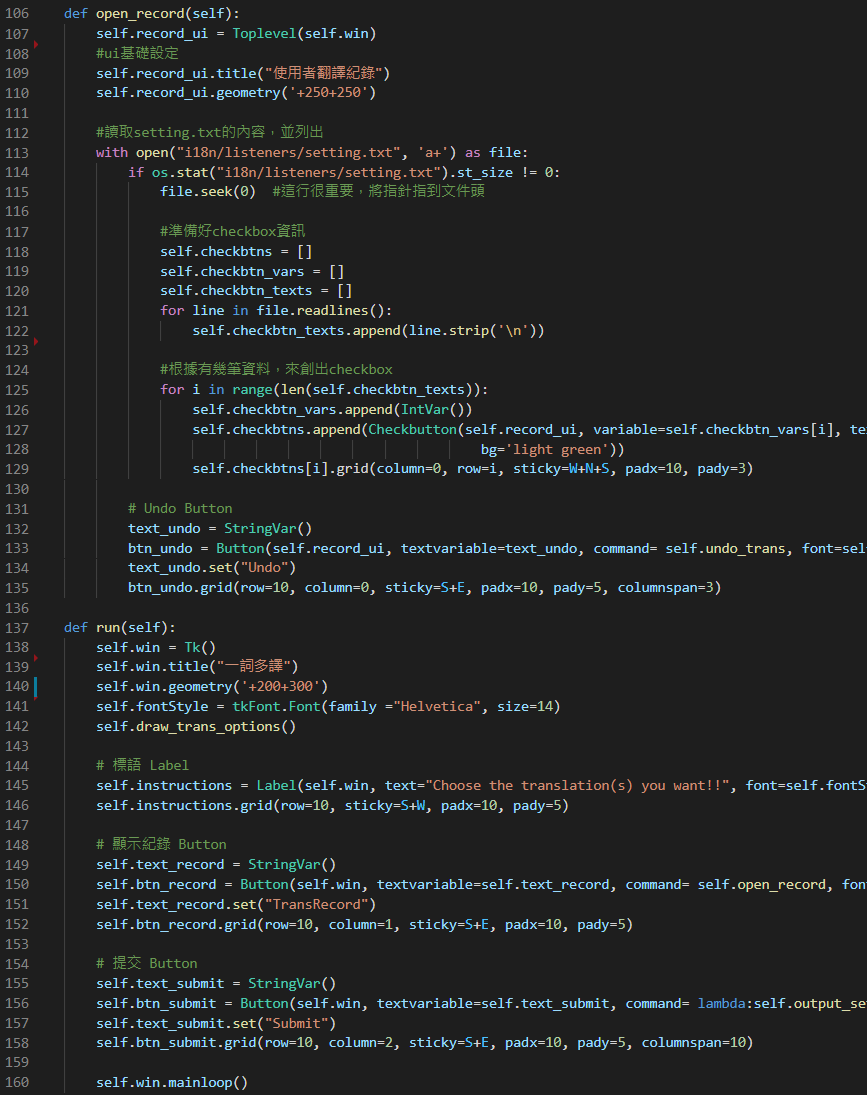


圖3-4-3 一詞多譯UI的程式實作(3)

1. run(): 負責產生出初始的一詞多譯UI介面，其中包含”Choose the translation(s) you want!!” 字樣，顯示翻譯紀錄的 TransRecord按鈕，以及寫入設定的Submit按鈕
2. draw\_trans\_options() : 負責把待翻譯詞和其翻譯，以及執行翻譯當下的完整參數，顯示於一詞多譯UI介面上
3. get\_transdic\_keys\_and\_values(): 從系統紀錄中，抓取待翻譯詞與其對應的翻譯，以利draw\_trans\_options()將其顯示於畫面上
4. open\_record(): 打開翻譯紀錄的頁面，會將先前使用者選擇過的翻譯詞，以條列式的方式呈現，並且頁面上包含一顆可以消除翻譯紀錄的Undo按鈕
5. undo\_trans(): 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。
6. output\_setting\_file(): 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。
7. add\_translations(): 負責將來自各個代理關鍵字傳入的待翻譯詞與其對應翻譯，分別儲存起來，方便之後get\_transdic\_keys\_and\_values()可以讀取。此外，將關鍵字的參數部分和待翻譯詞結合，成為可以辨別”此次執行測試腳本是否已執行過相同翻譯”的字串，並存入unique\_log中

圖3-4-4 為一詞多譯UI的sequence diagram

* 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-AWT-i18n”的Library，並透過twine模組上傳至Pypi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，必須先準備以下四個檔案: setup.cfg、pyproject.toml、README.md、LICENSE。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。

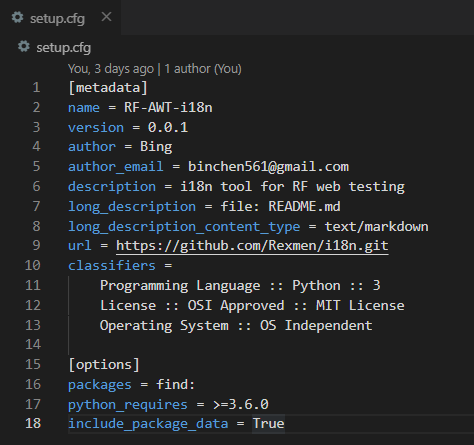


圖3-5-1 setup.cfg實作

setup.cfg內部設定name(模組名)、version(版本)、author(作者)、author\_mail(信箱)、description(模組描述)、long\_description(README.md的詳細說明)、long\_description\_content\_type(README.md的文件格式)、url(專案的網址)、classifiers(程式語言、憑證種類、作業系統)、packages=find:(透過find找出當前路徑的packages)、python\_require(Python需求版本)、include\_package\_data(是否包含其他類型檔案於package中)。

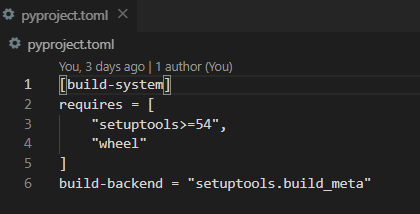


圖3-5-2 pyproject.toml實作

pyproject.toml 內部設定了建構系統所需要的setuptools版本與wheel。

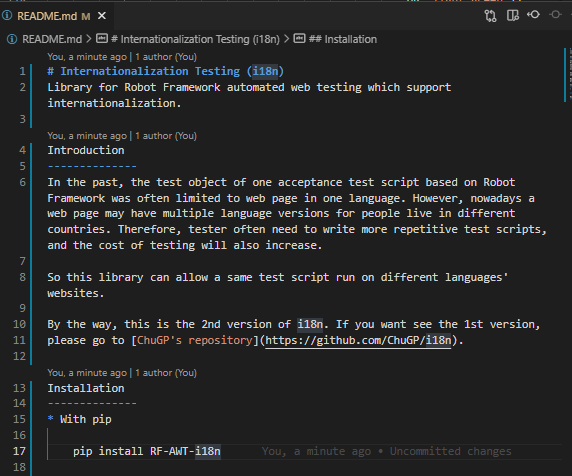


圖3-5-3 README.md實作

README.md 內部撰寫了關於此i18n工具的教學文件，讓使用者能夠更快的上手，使用此工具。

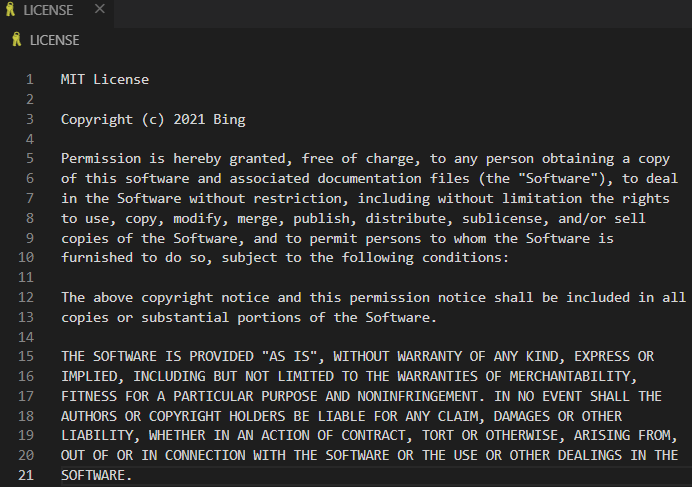


圖3-5-4 LICENSE內容

此LICENSE是屬用MIT LICENSE，詳請參考choosealicense.com

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(如圖3-5-5)，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-AWT-i18n模組檔案(如圖3-5-6)。



圖3-5-5 build指令

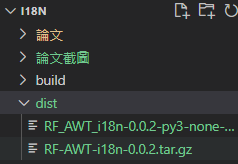


圖3-5-6 i18n模組檔案

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-AWT-i18n模組上傳至Pypi.org網站上了(如圖3-5-7)。之後在Pypi網頁上搜尋RF-AWT-i18n，便能看到剛才上傳的模組。

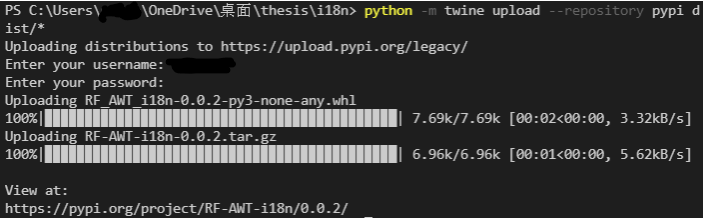


圖3-5-7 使用twine上傳模組至Pypi

使用者若要安裝此模組，只需執行pip install RF-AWT-i18n 指令即可(如圖3-5-8)。最後，使用者便可透過import此模組來使用i18n工具的功能。

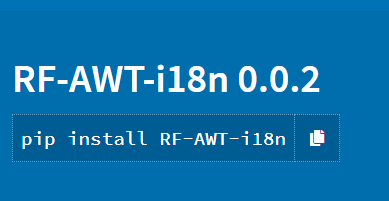


圖3-5-8 安裝RF-AWT-i18n指令

1. **第四章 測試案例分析**

本章節將為第三章擴充的代理關鍵字撰寫單元測試，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，最後套用至一個使用多項代理關鍵字的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**

以下將為各個新增或以修改實作之代理關鍵字，分別提供Robot Framework的單元測試腳本，以驗證關鍵字在新的擴充代理下，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。並在屬於修改實作的代理關鍵字前加上\*標示。

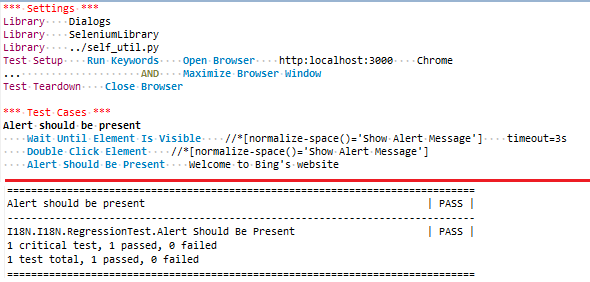


圖4‑1-1 Alert Should Be Present在代理關鍵字下執行的測試腳本與測試結果

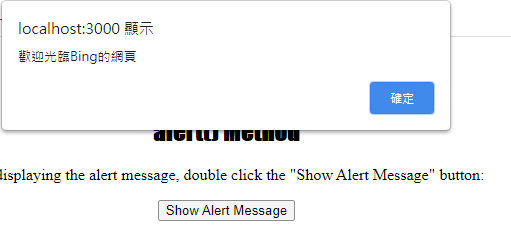


圖4‑1-2 網頁上的alert視窗顯示的文字



圖4‑1-3 Count Values In List在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-4 Dictionaries Should Be Equal在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-5 Dictionary Should Contain Key在代理關鍵字下執行的測試腳本與測試結果

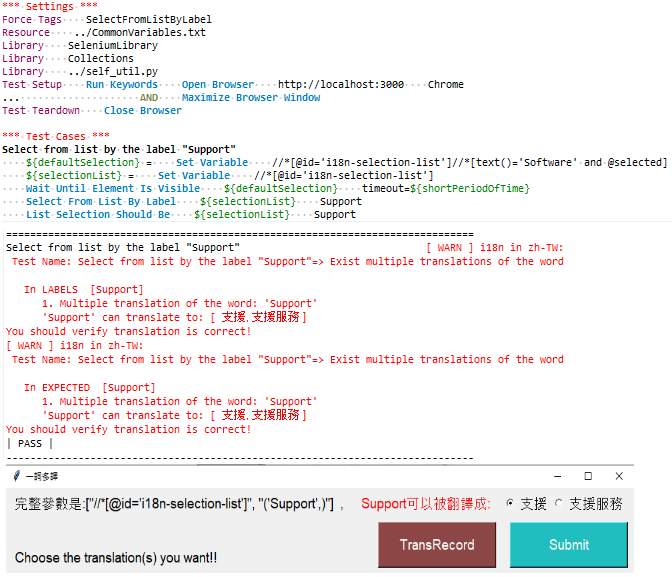


圖4‑1-6 Select From List By Label在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-7 Select From List By Label測試腳本要驗證的網頁元件



圖4‑1-8 \*Page Should Contain Element在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-9 \*Page Should Contain Element 測試腳本要驗證的網頁元件



圖4‑1-10 Remove Values From List在代理關鍵字下執行的測試腳本與測試結果

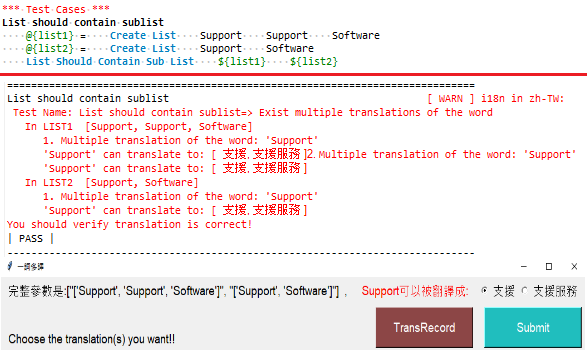


圖4‑1-11 List Should Contain Sub List在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-12 Table Should Contain在代理關鍵字下執行的測試腳本與測試結果



圖4‑1-13 Table Should Contain測試腳本要驗證的網頁元件

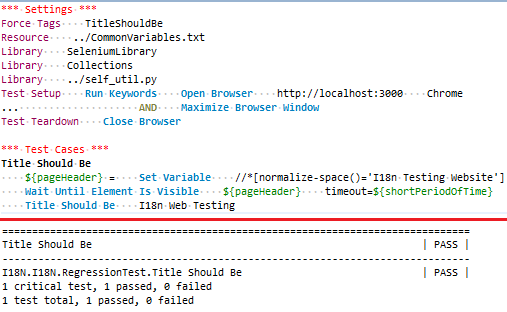


圖4‑1-14 Title Should Be在代理關鍵字下執行的測試腳本與測試結果

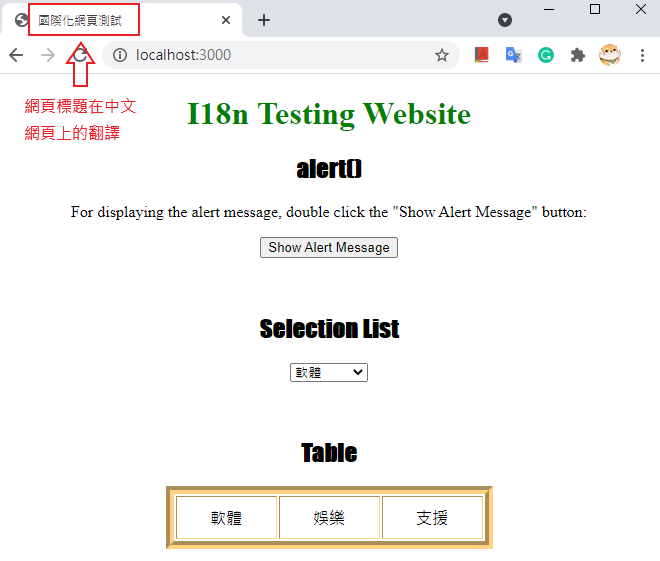


圖4‑1-15 Title Should Be測試腳本要驗證的網頁元件

* 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中文網頁上的文字。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:

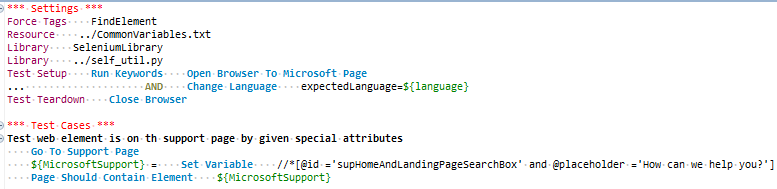


圖4-2-1 含有@placeholder屬性的XPath之測試腳本

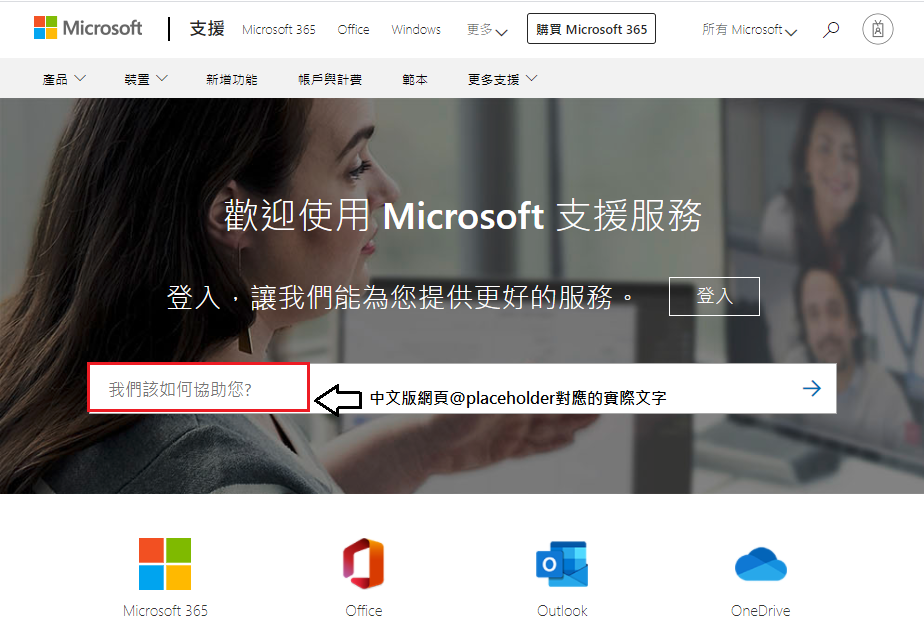


圖4-2-2 中文版Microsoft網頁@placeholder屬性對應的實際文字

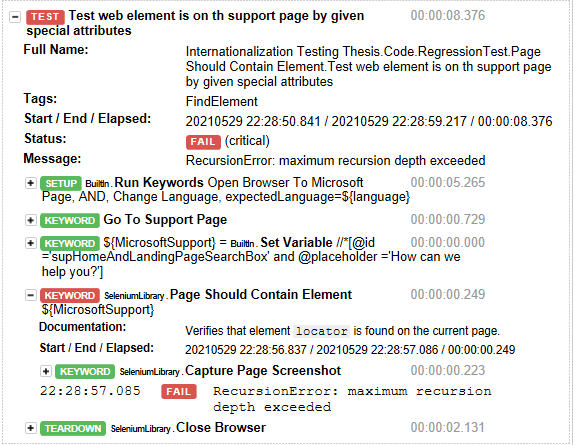


圖4-2-3 待翻譯XPath運行在未改善翻譯邏輯時的i18n工具之結果

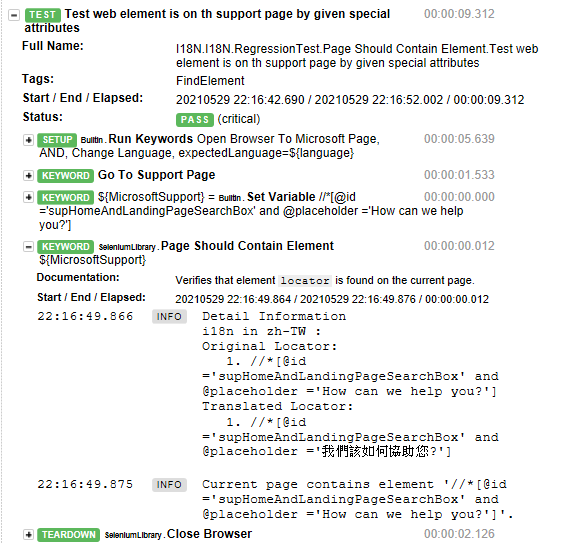


圖4-2-4 待翻譯XPath運行在改善翻譯邏輯後的i18n工具之結果

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。當前版本的i18n工具，因為改善翻譯邏輯而支援各種html屬性的翻譯，並將”how can we help you?”，成功翻譯為”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為例(如圖4-3-1)，系統於腳本運行結束時會產生一個圖形化使用者介面(如圖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)。

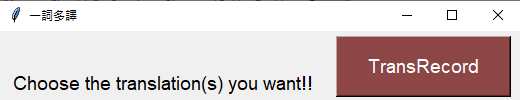


圖4-3-3 按下Submit按鈕後，將翻譯寫進設定檔，同時清除翻譯選項

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)。

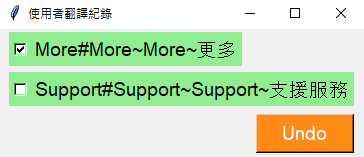


圖4-3-4打開翻譯紀錄介面，上面記錄著使用者翻譯選擇

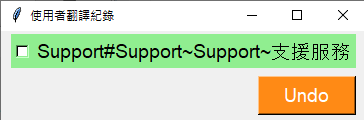


圖4-3-5 清除選擇的翻譯紀錄後，再次打開翻譯紀錄介面

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)。

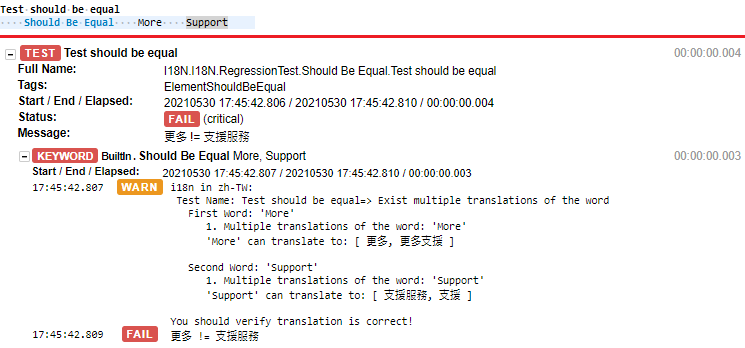


圖4-3-6 測試腳本不通過的情況下，不跳出UI，但會顯示warning資訊於報表

1. 測試腳本通過，且沒有遭遇一詞多譯，或是設定檔中已存在使用者的翻譯選擇:

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以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)

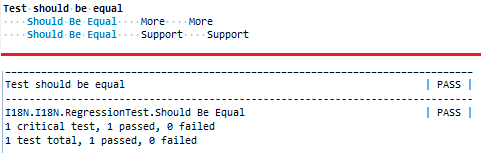


圖4-3-7 測試腳本通過且沒有遭遇一詞多譯的情況下，不跳出UI

* 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待翻譯屬性來撰寫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。

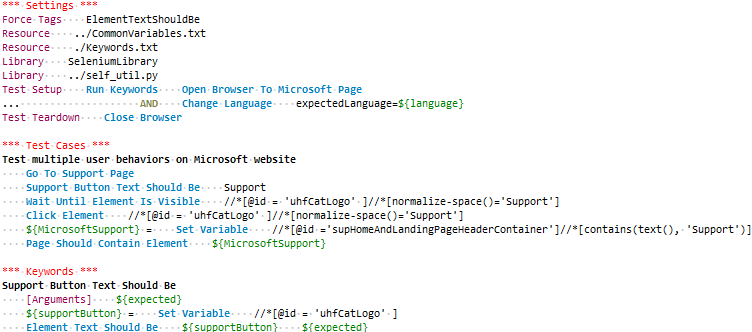


圖4-4-1 Test multiple user behaviors on Microsoft website測試腳本實作

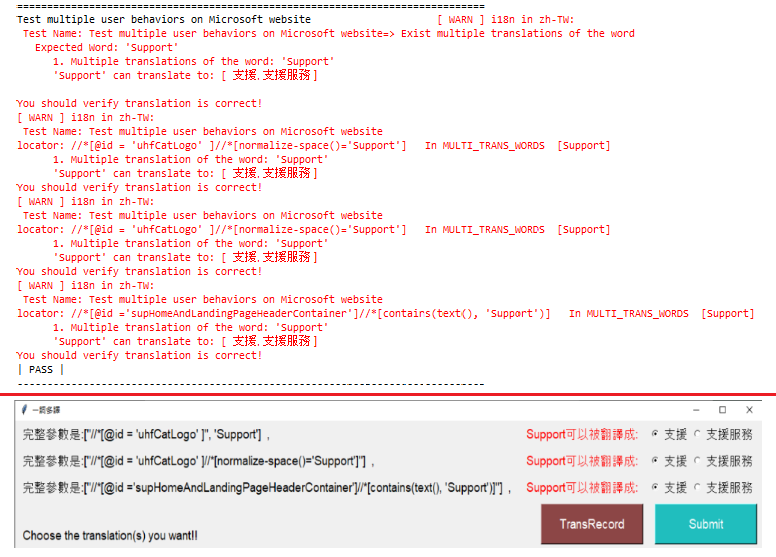


圖4-4-2 Test multiple user behaviors on Microsoft website測試腳本第一次執行結果

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。

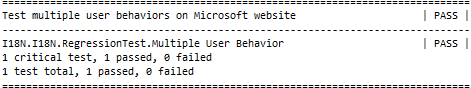


圖4-4-3 Test multiple user behaviors on Microsoft website測試腳本第二次執行結果

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。

1. **第五章 結論與未來展望**
   1. **本論文限制**

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5-1-1，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。

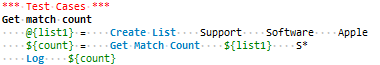


圖5-1-1 Get Match Count測試腳本

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 5-1-2 ，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5-1-3)。

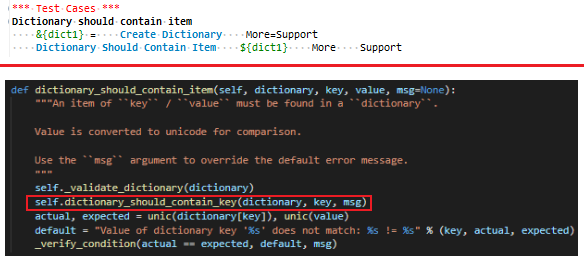


圖5-1-2 Dictioanry Should Contain Item測試腳本與其原生實作



圖5-1-3 第一次與第二次執行測試腳本，待翻譯詞的參數紀錄變化

* 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. **未來展望**

目前i18n工具的代理關鍵字尚存在著使用的限制，未來可以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。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，軟體工程，臺北：社團法人臺灣軟體工程學會，2013，第194頁
2. 鍾承翰，基於 Scrum 與 Mob programming 開發自動化驗收測試的準樣式，

碩士論文，國立臺北科技大學資訊工程系碩士班，臺北，2020。

1. 呂昭陞，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，國立臺北科技大學資訊工程系碩士班，臺北，2018。

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, San Francisco, CA, USA

1. “INTERNATIONALIZATION” [Online]. Available:

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

1. “Localization vs. Internationalization” [Online]. Available:

<https://www.w3.org/International/questions/qa-i18n>.

1. “JSON - Introduction” [Online]. Available:

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

1. “Robot Framework” [Online]. Available: [https://robotframework.org/.](https://robotframework.org/)
2. “Robot Framework User Guide” [Online]. Available:

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

1. “The Python Tutorial” [Online]. Available:

<https://docs.python.org/release/3.6.0/tutorial/index.html.>

1. “Microsoft official website” [Online]. Available: <https://www.microsoft.com/>.
2. “angular-translate” [Online]. Available:

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

1. “ng-i18next” [Online]. Available:

https://github.com/i18next/ng-i18next.

1. “angular-i18n” [Online]. Available:

https://github.com/nolazybits/angular-i18n.

1. “ngx-translate” [Online]. Available:

https://github.com/ngx-translate/core.

1. “angular-i18next” [Online]. Available:

https://github.com/Romanchuk/angular-i18next.

1. “Transloco” [Online]. Available:

https://github.com/ngneat/transloco.

1. “react-i18next” [Online]. Available:

https://github.com/i18next/react-i18next.

1. “redux-i18n” [Online]. Available:

<https://github.com/APSL/redux-i18n>.