智慧化企業整合 PROJECT 3

工程知識平台 (Engineer Knowledge Platform, EKP)

106034803 趙敏樺

摘要

本專題旨在設計並開發一智慧化工程知識平台架構,以下用 EKP 簡稱。必須包含資料管理、資料分析、和諮詢服務等面向。由於產品生命週期過程龐大且複雜，詳細的設計仍需經由不斷的需求訪談、需求分析、和設計分析等軟體開發的歷程，本次報告僅提供概略的平台架構，分析各項技術的可行性並實作。

內容

圖目錄

1 5W1H 分析

隨著時代演進，人工智慧相關技術發票日新月異，如何應用人工智慧協助產業升級並提升競爭力是現今的關鍵議題。本章以 5W1H 分析描述此專案想解決的問題。

- Why

在產品生命週期過程中會產生大量的工程文件，但文件種類繁多，包含 ISO 文件、技術文件、合約管理、SOP 文件、和重要機密文件等，格式各異，且分散在企業內各不同部門手中，各部門又有自己的管理模式。這樣大量的文件裡其實暗藏了大量的資訊，但無法有效妥善運用。且文件的保存、效率、和安全機制都是很重要的議題。

- What

EKP 必須包含資料管理、資料分析、和諮詢服務等面向。

- Who

企業內的所有成員都會參與。

- When

當有任何文件產生的同時，都應仔細思考如何管理並保存這些文件，讓其中的知識能被保存並運用。

- Where

在產品生命週期過程中的任何一個階段，從前端的需求確認、設計分析，到製造檢驗，再到後端的後勤品保和客戶服務，舉凡會產生任何工程文件的活動，都可以納入此平台。

- How

發展一資訊平台，先納管文件，並分析作業流程，進而分析各項文件並產生有效的資訊。

1.1 預期成果

當所有文件都資訊平台上納管並有效分析萃取其中的知識，預期能透過人工智慧以自然語言解析並分析文件、更精準地提供報價、成本、或利潤分析等數據，進而開發智能化工程諮詢機人提供用戶更完整的體驗。

2 文獻回顧

本章回顧相關文獻。

2.1 Java 物件導向程式語言

Java 是一種廣泛使用的電腦程式設計語言，擁有跨平台、物件導向、泛型程式設計的特性，廣泛應用於企業級 Web 應用開發和行動應用開發[1]。預期 EKP 實作中會大量引用到外部資源，且考量後續若要提供行動版的服務，Java 跨平台的特性會是一大優勢。

2.2 TOMCAT

Tomcat 是由 Apache 軟體基金會屬下 Jakarta 專案開發的 Servlet 容器，按照 Sun Microsystems 提供的技術規範，實現了對 Servlet 和 JavaServer Page（JSP）的支援，並提供了作為 Web 伺服器的一些特有功能，如 Tomcat 管理和控制平台、安全域管理和 Tomcat 閥等。由於 Tomcat 本身也內含了 HTTP 伺服器，因此也可以視作單獨的 Web 伺服器[2]。當選定 Java 作為主要的程式語言時，採用 Tomcat 作為伺服器的選擇也是順理成章的。

2.3 ZK

ZK 是一套以 AJAX／XUL／Java 為基礎的網頁應用程式開發框架，用於豐富網頁應用程式的使用介面。最大的好處是，在設計 AJAX 網路應用程式時，輕鬆簡便的操作就像在設計桌面程式一樣。ZK 包含了一個以 AJAX 為基礎、事件驅動（event-driven）、高互動性的引擎，同時還提供了多樣豐富、可重複使用的 XUL 與 HTML 元件，以及以 XML 為基礎的使用介面設計語言 ZK User-interfaces Markup Language（ZUML）。ZK 提供超過 120 個 XUL 元件及 80 個 XHTML 元件。舉凡 listbox, slider, audio, slider, tree, combobox, tabbox, auto-completion 等均有支援。ZK 亦提供 CKeditor 和 Google Maps 的元件，讓使用者直接以 Java 控制，無須使用 JavaScript[3]。

同樣地，選定了 Java 作為主要的程式語言，ZK 讓使用者可以直接用 Java 控制頁面，協助我們大量減少編寫 html 和 JavaScript 所需要的時間。另外，不管是 MVC (Model-View-Controller)架構或是 MVVM(Model-View-Viewmodel)架構，ZK 都能完整支援。

2.4 LUIS

Language Understanding Intelligent Service (LUIS)是一種 API 雲端式服務，可將自訂機器學習智慧套用至使用者的對話、自然語言文字中，以預測整體意義，並找出相關的詳細資訊。LUIS 的用戶端應用程式是任何對話應用程式，可與使用者透過自然語言溝通以完成工作。用戶端應用程式的例子包括社群媒體應用程式、聊天機器人，以及具備語音功能的桌面應用程式。LUIS 應用程式發佈後，用戶端應用程式會將語句 (文字) 傳送至 LUIS 自然語言處理端點 API，並以 JSON 回應的形式接收結果。聊天機器人是 LUIS 的常見用戶端應用程式之一[4]。

3 現況說明和目標模型

本章簡要說明沒有使用文件管理平台的現況,和預期平台可以提供的目標模型。

3.1 現況說明

以某企業的傳統文件管理現況作為案例說明，如圖 3.1 所示。在沒有完整的整合性平台時，會有以下的問題。

圖 3. 1 傳統文件管理現況(來源：https://www.newtype.com.tw/[5])

- 分享方式

當缺乏整合性平台時，檔案往往透過電子郵件等網站傳輸方式，或是透過隨身碟傳遞，導致同一份文件散落在各處，無法識別真正有效的文件。

- 版本管理

承上，當檔案重覆出現在不同地方，且又有多人共同編輯時，檔案的版本管理變得非常困難，甚至容易出現重工或是變更記錄遺失的問題。

- 缺乏流程概念

當一份檔案是需要透過逐層簽核時，沒有引入流程管理將無法有效識別該文件的簽核記錄。

- 安全性

上述三點，都將導致文件的資料安全性和機密性不齊全。

3.2 目標模型

文件管理架構示意圖如圖 3. 2 所示。

圖 3. 2 文件管理架構(來源：http://new.lyserp.com.tw[6])

透過統一的平台管理企業內部的文件，可以解決 3.1 提到的各項問題。

- 分享方式

資訊平台上的版本為主，其他個人磁碟或隨身碟裡的都只是備份或是暫存。

- 版本管理

平台上可以查詢各文件在每個版本的快照，並能分析各版本之間的差異。

- 簽核流程

引入流程引擎，線上簽核各項文件，不只提升效率，更能完整記錄文件的簽核歷程。

- 安全性

把文件交給資訊系統管理後，可以針對各使用者設定權限之外，也可以把文件加密或是上鎖，限制可取用的管道，確保資料取用的安全性。當匯出文件時，也可以打上浮水印和仿偽識別，避免文件遭不當篡改。

4　平台架構實作

本章說明本專案欲開發的智慧化工程諮詢平台實作。本專案選用 Java 作為核心的開發語言，本章將依序說明 Eclipse 集成開發環境(Integrated Development Environment, IDE)、Apache Tomcat 伺服器、ZK 網頁專案、Java 商業層、和 JDBC 連接 MySQL 等主題。

4.1 Eclipse IDE

本專案以 Java 程式語言進行架構開發，採用 Eclipse Oxygen 版本作為主要的開發工具。

Eclipse 初次啟動時，要先創建一個資料夾作為工作平台(Workspace)，之後所有的程式碼都會在該資料夾中。接著，在 windows->preference 下的 java->Build Path->User Libraries，配置環境會用到的函式庫(libraries)。這些函式庫是可以給 Workspace 中所有的專案使用的。如圖 4. 1 所示。

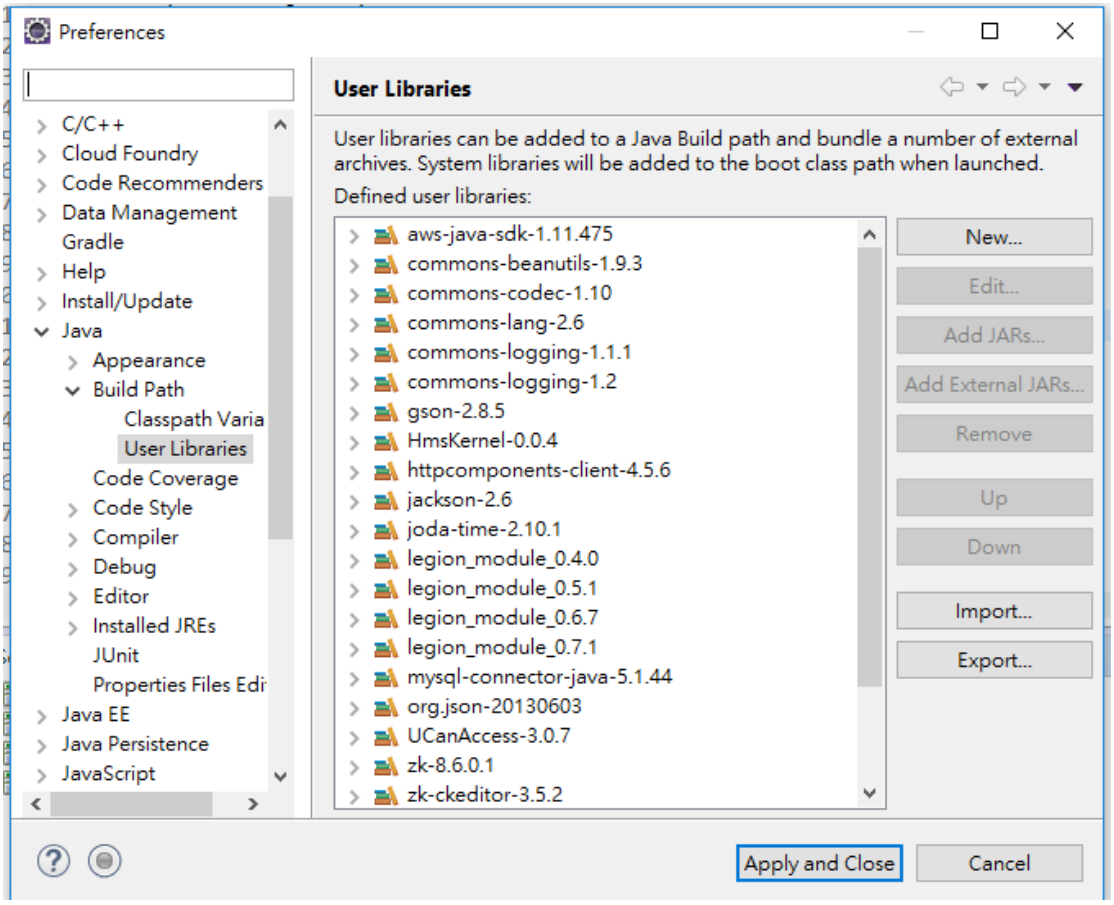

圖 4. 1 Eclipse User Libraries

各專案在開發過程中，隨著功能的需要會引入各個函式庫。目前 EKP 用到的函式庫如圖 4. 2 所示，包含基本的 Java 開發套件(Java Development Kit, JDK)、用以和 Apache Tomcat 伺服器溝通的函式庫、和透過 ZK 設計前端頁面的相關函式庫等；另外還有一些通用模組 commons.*、處理 JSON 格式的 org.json、負責網路連線的 httpcomponents-client、負責跟資料庫連接的 mysql-connector、和單元測試的 Junit4 等。

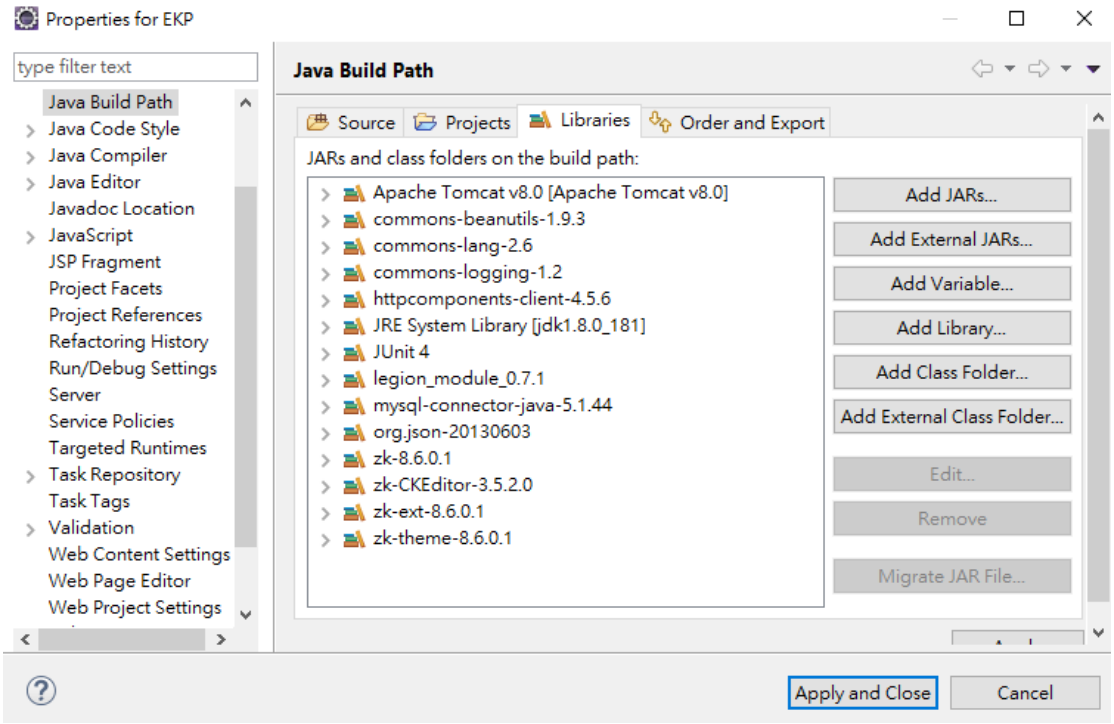

圖 4. 2 Eclipse Project Java Build Path

另外，有一個自定義的 legion_module 函式庫，作為一個代理函式庫，把和 AWS、LUIS、和 Google API 等的實作介面封裝起來，讓 EKP 能透過較簡化的介面直接呼叫各個應用程式。此外，還有一些通用的實作，像是 JDBC 的連線、SQL 語法處理、和檔案存取等架構，也透過 legion_module 函式庫完整封裝。

4.2 ZK 網頁專案

EKP 的前端將以網頁平台的型式提供服務。由於 EKP 以 Java 為主要的編程語言，為了簡化寫 html 和 css 語法的過程，EKP 採用普奇科技開發的以 Java 為基的 ZK 套件，實作 MVC 架構的前端網頁設計。首先，在 ZK 的官網上下載 ZK CE 最新的 8.6.0.1 版本，如圖 4. 3 所示。

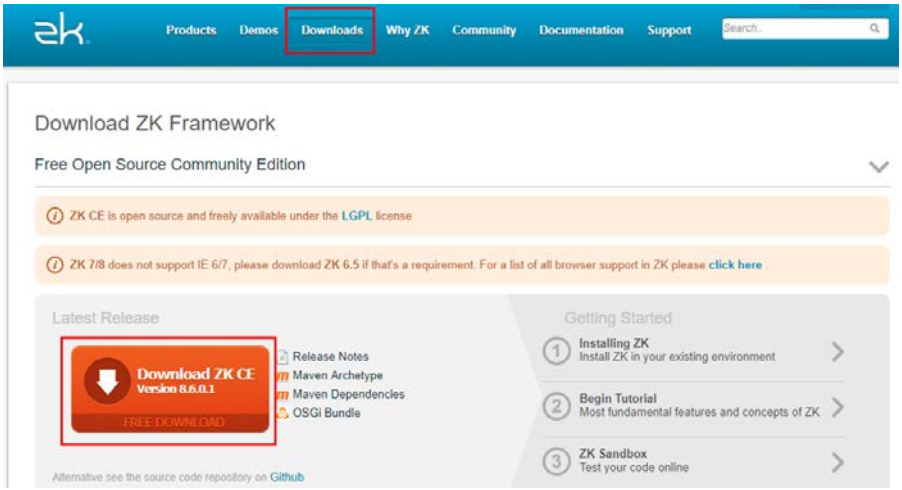

圖 4. 3 ZK 下載頁面(來源：https://www.zkoss.org/download/zk[7])

ZK 的 PE(專業版)和 EE(企業版)提供更強大的功能，但那些都是要付費的，只有 CE 版本是免費的。下載完會得到一個 zip 檔，解壓縮後，找到 zk-bin-8.6.0.1->dist->lib 資料夾，我們需要把裡面所有的 jar 檔匯入到專案環境的函式庫中。如圖 4.4 所示。這樣我們就可以在程式中引用 ZK 提供的各個 API。

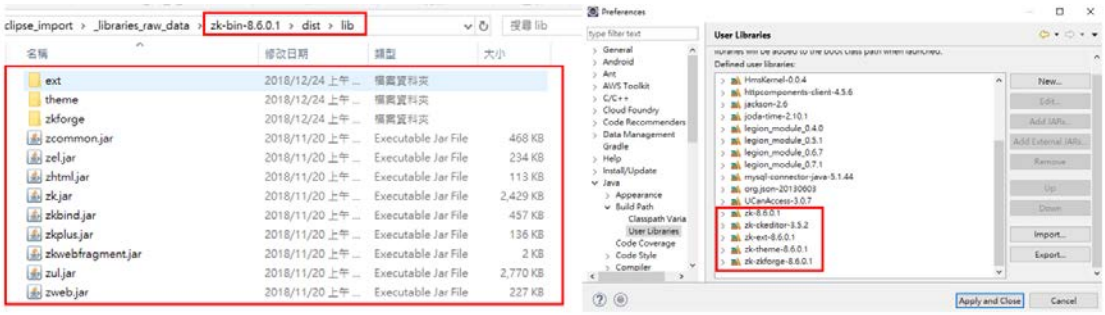

圖 4. 4 ZK 函式庫設定

完成引用函式庫後，我們要在 eclipse 環境裡編輯 ZK 的頁面，還需要引用 ZK 為 Eclipse 提供相對應的 IDE—ZK Studio。在 Eclipse 的 Help->Eclipse Marketplace 中，輸入 zk studio 並搜尋，安裝 ZK Studio 2.0.3 版本。如圖 4.5 所示。

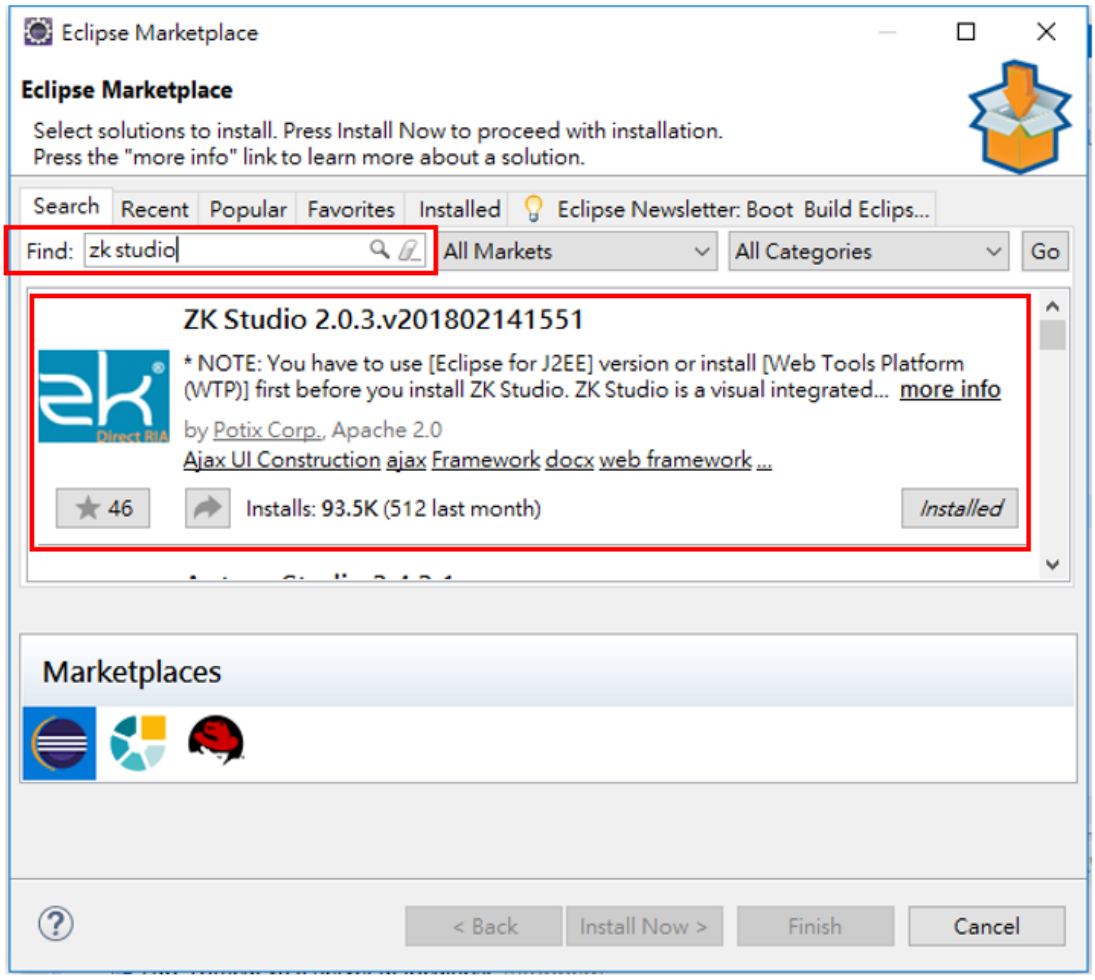

圖 4. 5 ZK Studio 下載

安裝完 ZK Studio 後，在 Eclipse 的 File->New 選單中，可以看到 ZK 相對應的 ZK Project 和頁面檔案 ZUL；在 Preference 選單也可以看到相對應的 ZK 設定。如圖 4. 6 所示。

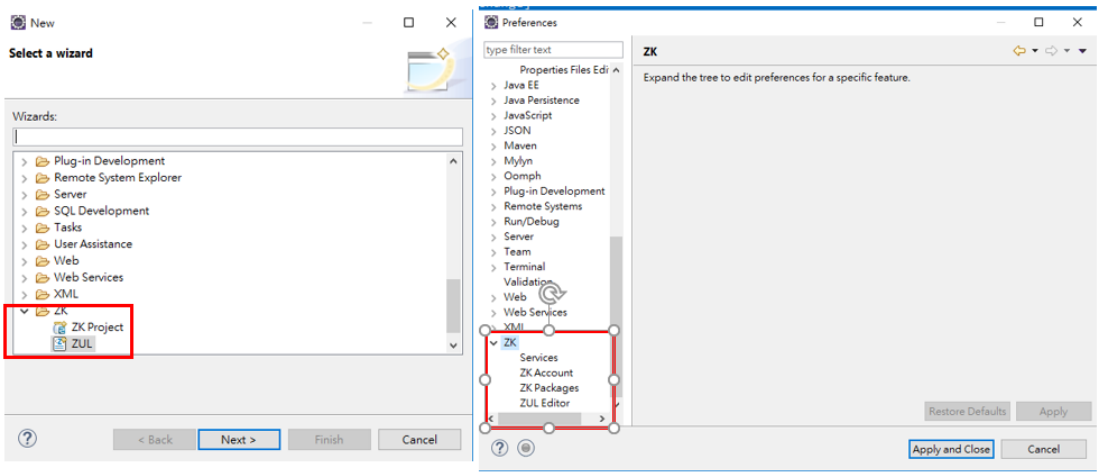

圖 4. 6 ZK 環境設定

4.3 Tomcat 伺服器

本專案採用 Tomcat 8.0.46 版本的伺服器。在 Tomcat 的官網可以選擇下載各個版本的 Tomcat 伺服器，如圖 4. 7 所示。

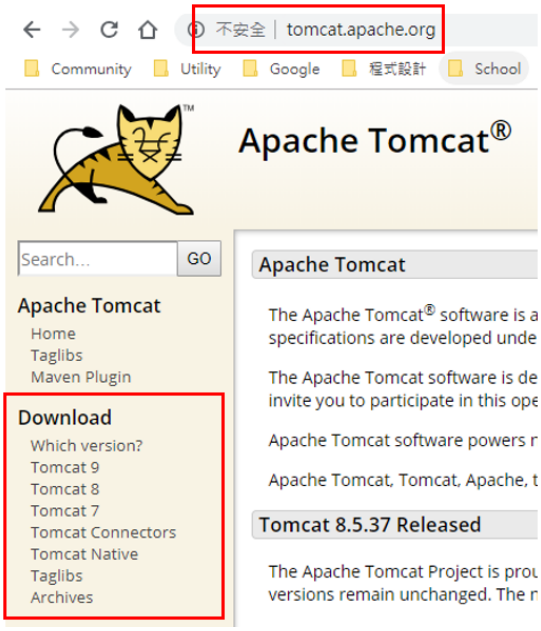

圖 4. 7 Apache Tomcat 下載頁面(來源：http://tomcat.apache.org[8])

在 Eclipse 中的 windows->preference->Server->Runtime Environment 中，指定下載下來的 Tomcat Server 路徑，並指定搭配的 Java 運行環境(JRE)。如圖 4. 8 所示。

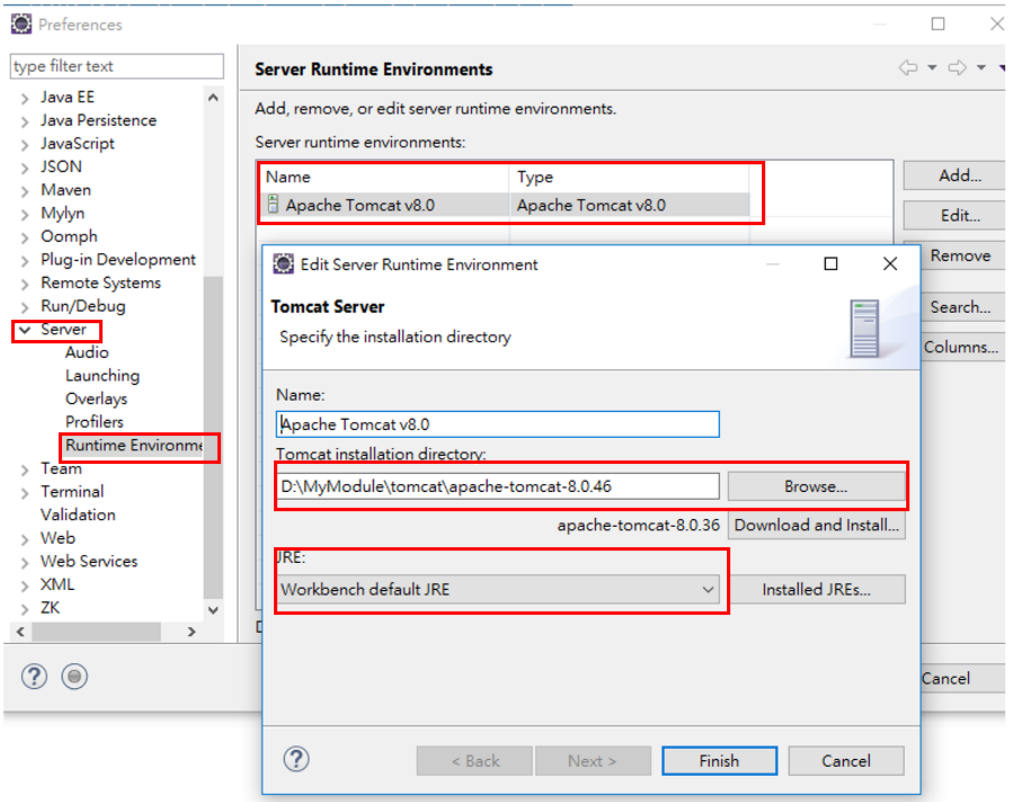

圖 4. 8 Tomcat 環境設定

至此伺服器的環境設定完成了，接下來要把應用程式掛載到伺服器上。如圖 4. 9 所示，在 Server 的視觀(Perspective)下，選擇 new server，選定剛剛建立好的伺服器運行環境，再把 EKP 專案引入，這樣一個搭配 EKP 專案的 Tomcat 伺服器就架構完成了。

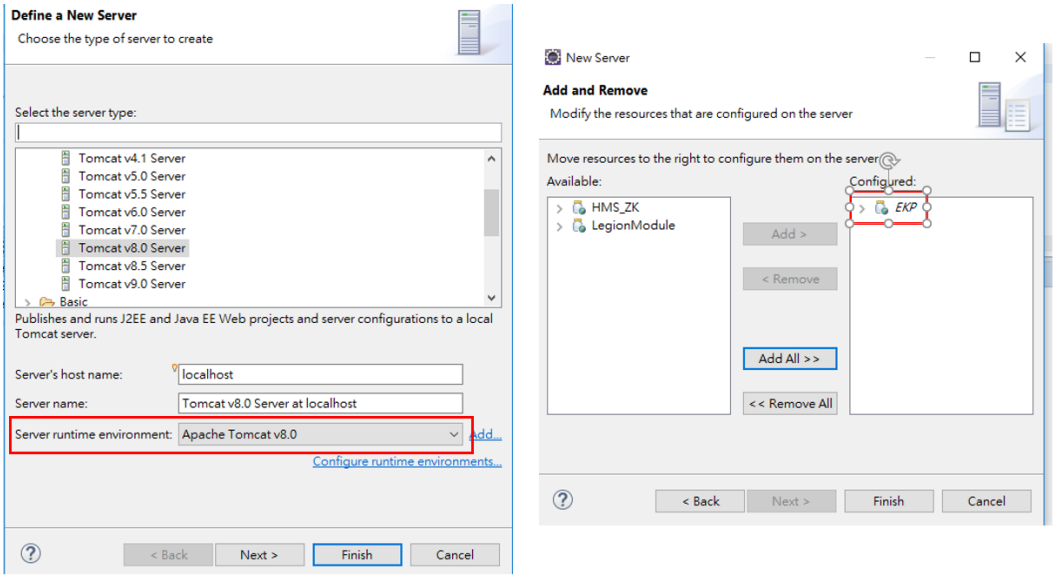

圖 4. 9 建立 Application Server

完成架構後，搭配 4.2 建好的 ZK 專案，啟動伺服器，如果可以看到 ZK 預設的 Hello World，如圖 4. 10，就代表環境建置成功了。

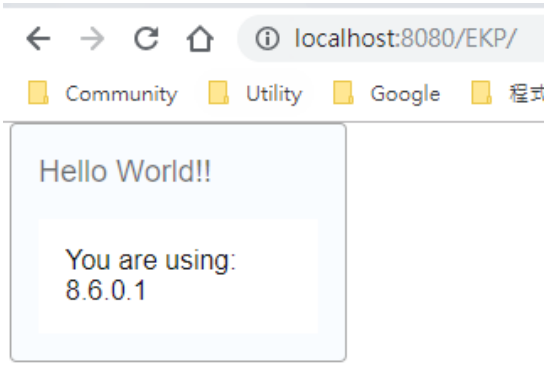

圖 4. 10 ZK Hello World

4.4 Java 商業層

本小節說明 EKP 的 Java 商業層架構，但不會詳細描述實體關連模型。主要原因是報告在說明架構並驗證可行性，另外本專案牽涉的商業範疇龐大，詳細的架構設計仍需要進一步的需求訪談、分析、和設計，並非目前可以完整描述。

從前端的頁面設計、商業層的 Java 物件導向程式設計、到後端的資料庫連線模組，如圖 4. 11 所示。明確區分各層的範疇，再搭配單元測試的應用，創造一個完整且容易維護的專案架構。

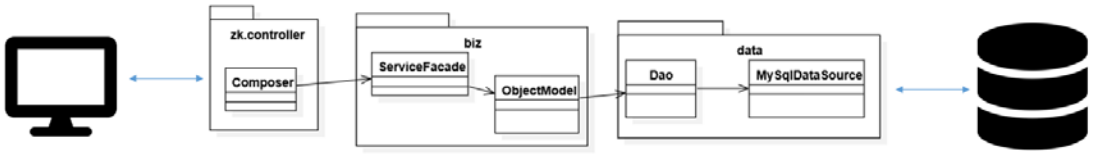

圖 4. 11 Java 架構

4.5 MySql 資料庫

本專案採用 Java 提供的 JDBC 函式庫連接 MySQL 資料庫，示意圖如圖 4. 12 所示。Java 應用程式透過 JDBC API 和對應的驅動程式(driver)便能直接和資料庫溝通。甚至如果哪天想要更換使用的資料庫，只要抽換相對應的 driver 即可，程式需要改動的幅度通常不大。

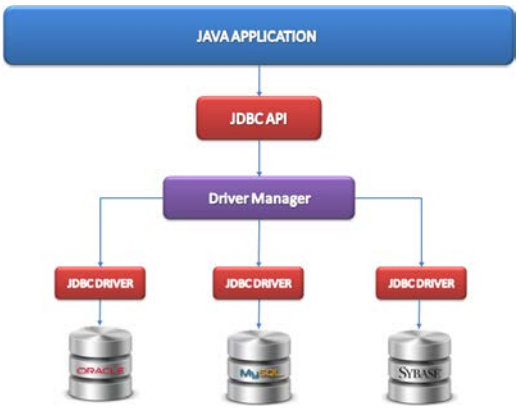

圖 4. 12 JDBC 架構

5 諮詢機器人

本章說明 EKP 採用的諮詢機器人實作。5.1 首先說明整體架構，5.2 說明 LUIS 模型，5.3 進一步說明在 Java 程式中呼叫 LUIS 提供的網路服務(Web Service)。

5.1 整體架構

本專案的諮詢機器人由一 BotMaster 作為核心，整體架構如圖 5. 1 所示。首先，在 ZK 頁面上打造出一個對話頁面，使用者在頁面輸入語句(utterance)後，會先交由預先建好的 LUIS 模型判讀語句最有可能的意圖(intent)和實體(entity)。BotMaster 會有一設定好的門檻值(threshold)，當該意圖的機率大於門檻值時，會依照該各個意圖執行相對應的函式呼叫，可能是從資料庫查詢資料，或是其他呼了 AWS 或 Google API 的服務，最後再把得到的結果回傳給使用者。

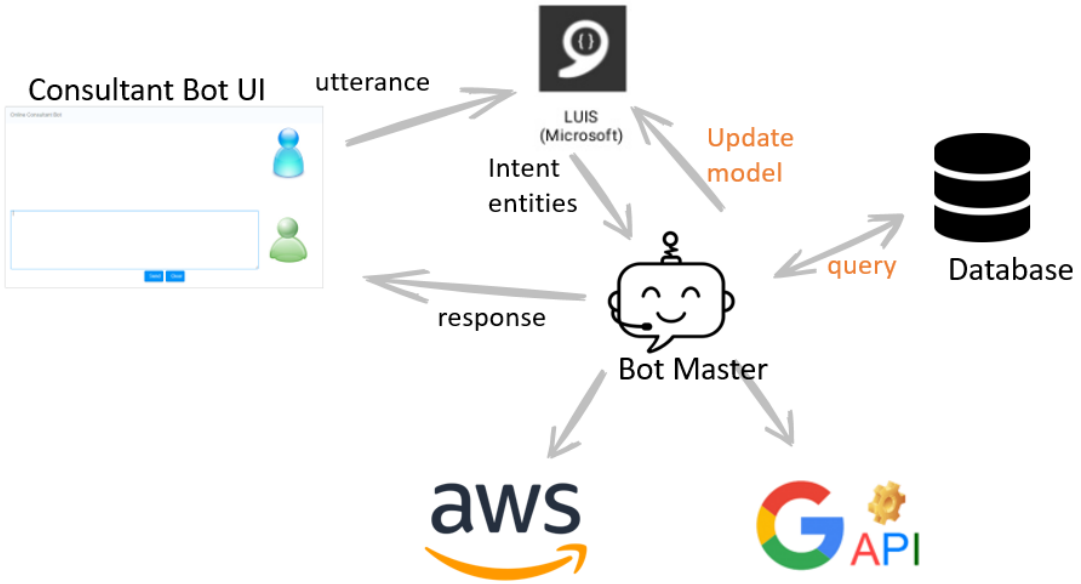

圖 5. 1 EKP 諮詢機器人架構

但若是該意圖的機率小於門檻值，BotMaster 會向使用者確認其意圖，並即時更新並重新發佈(publish)LUIS 模型。

目前 EKP 只有實作兩種意圖，包含「詢問名詞定義」和「查詢文件資料」，後續將再持續分析使用者情境並擴充。

5.2 LUIS 自然語言理解

在創建完 LUIS 的帳號後，可以在「My apps」目錄下按下「Create new app」來新增模型。如圖 5. 2 所示，「Engineer Knowledge」便是 EKP 使用到的 LUIS 模型。

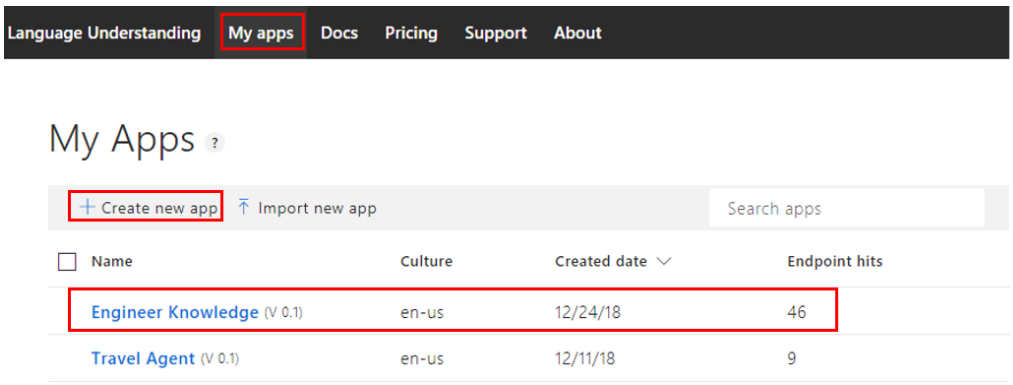

圖 5. 2 建立 LUIS App

點擊名稱進到模型主控台，如圖 5. 3 所示。在 Build 畫面，可以設定意圖(Intents)和實體(Entities)。目前此模型設定有 3 種意圖和 1 種實體。由於這只是在驗證可行性，模型仍不是很完整。一個完整的模型應該會有更多種類的意圖和實體，以實現更強大的自然語言處理能力。

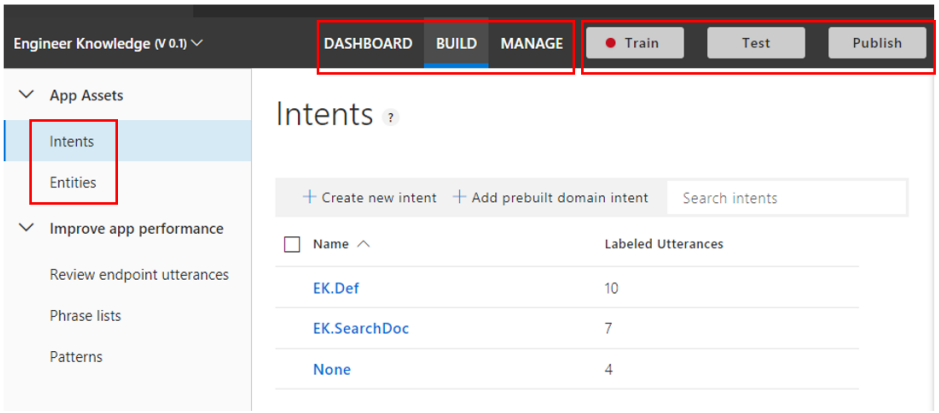

圖 5. 3 LUIS 主控台

設定完後透過「Train」進行訓練，訓練後每個語句會顯示它被理解為該意圖的機率。如圖 5. 4 所示。

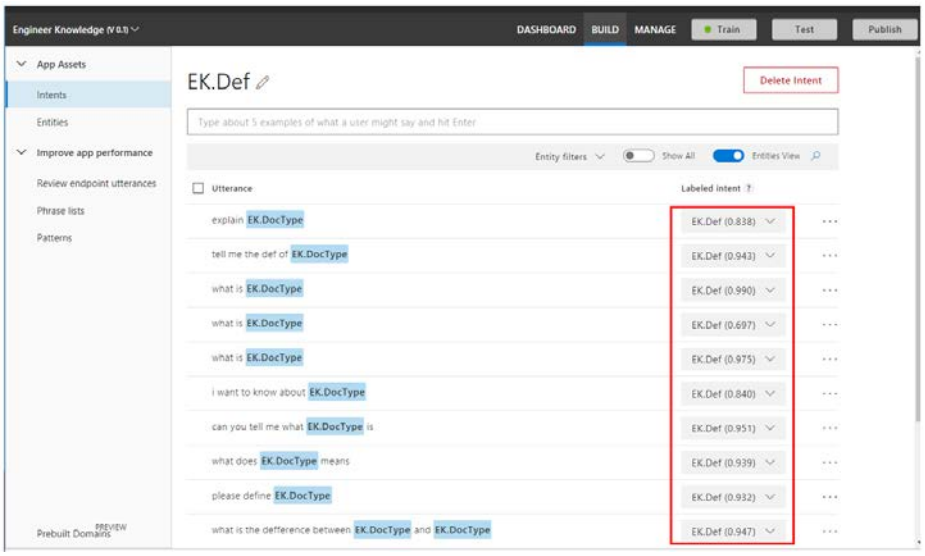

圖 5. 4 LUIS 訓練結果

5.3 Java 呼叫 LUIS 網路服務(Web Service)

此小節首先說明 LUIS 對於不同程式呼叫的 API 說明文件來源，接著說明 EKP 目前有實作的服務，最後再展示 EKP 的諮詢機器人所具備即時更新模型的功能。

5.3.1 說明文件

本專案的諮詢機器人建構在 Java 平台中，於是需要用 Java 程式語言呼叫 LUIS 的服務。微軟的網站上提供了[9]豐富的 API 說明，如圖 5.5 所示。左側依服務的種類區分選單，另外在下面也針對不同的程式語言提供範例程式碼，相當詳盡。

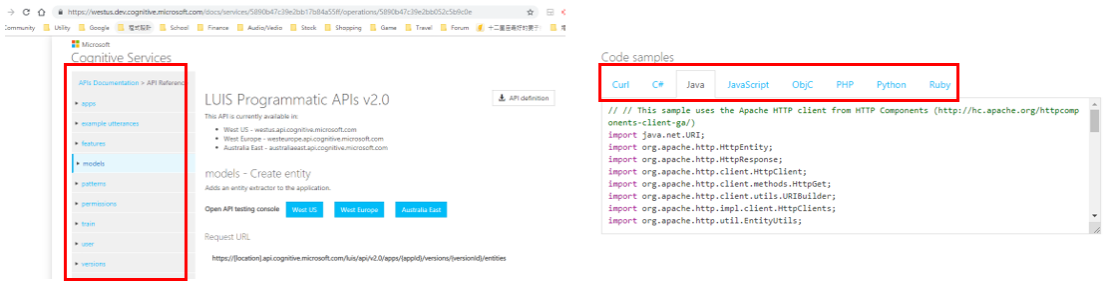

圖 5. 5 LUIS API 說明文件

5.3.2 實作的服務

EKP 目前實作包含取得所有意圖(getAllIntents)、取得指定意圖(getIntent)、建立意圖(createIntent)、理解語句(understanding)、新增語句(addUtterance)、訓練(train)、和發佈(publish)等服務，如圖 5. 6 所示。

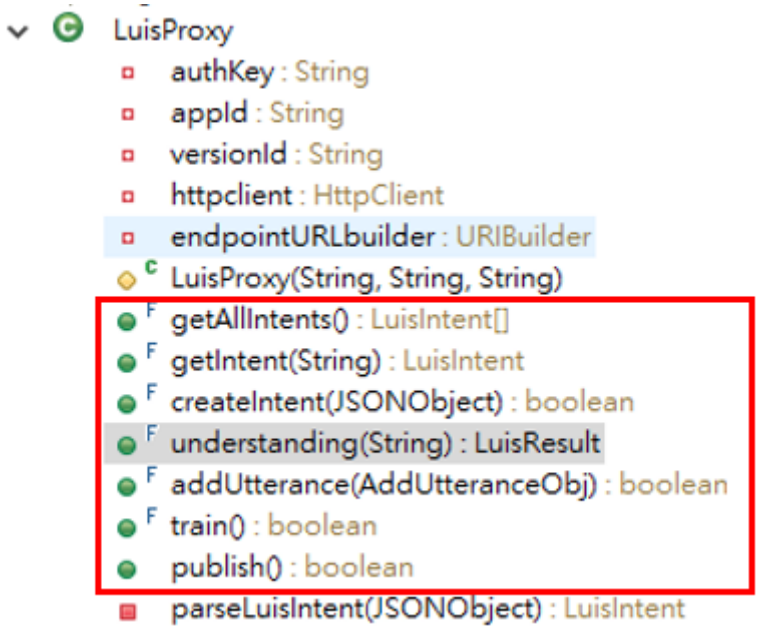

圖 5. 6 EKP 實作的 LUIS API

5.3.3 即時更新並發佈模型

我們知道機器學習或是深度學習的運作模式，都是在找出一個最適當的函式(function)來符合已知的資料集(dataset)。換句話說，機器學習或深度學習的結果就只是一個函式。在各個應用中實作機器學習或深度學習時，就是去呼叫這個已經訓練好的函式。如果這個函式的判斷結果不夠精確，應用程式端也無能為力。

EKP 透過 BotMaster 的語意和情境判斷，設計出讓使用者可以參與更新並發佈 LUIS 模型的能力。圖 5. 7 是一段由使用者更新模型的案例。

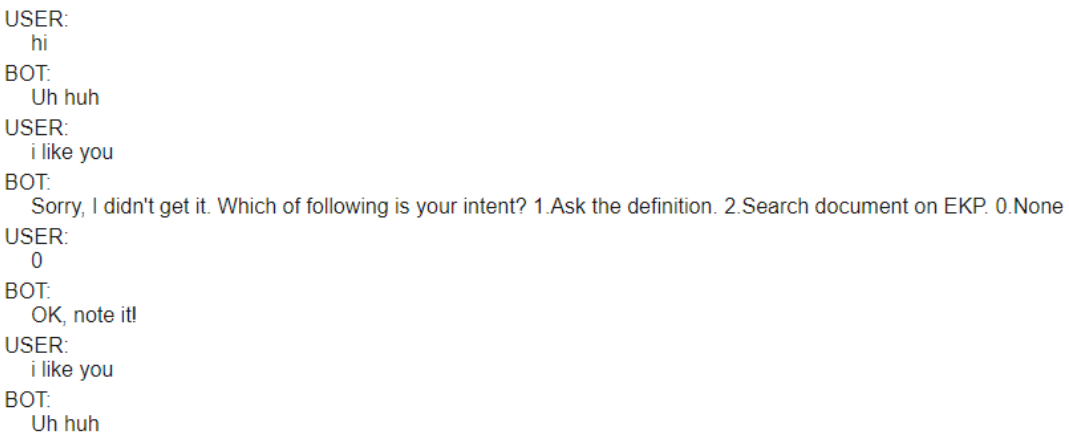

圖 5. 7 EKP 更新模型實例

EKP 的諮詢機器人在面對沒有意圖的語句時，會用「Uh huh」帶過。在對話一開始，使用者說「I like you」，這是諮詢機器人沒有事先被訓練過的語句，此時機器人回覆請使用者協助確認意圖。當使用者說明意圖為「None」時，機器人便開始學習、訓練、並重新發佈，完成後回答使用者「OK, note it!」。於是當使用者再一次說「I like you」時，機器人已經知道使用者的意思是「None」，便回覆「Uh huh」。這段對話展現了 EKP 的諮詢機器人可以在有限的條件下，隨著對話的進行而不斷強化其 LUIS 模型。

6 其他實作技術

本章說明其他本專案有實作的技術，包含文件儲存管理、呼叫 Google API、和呼叫 Amazon Web Service (AWS) API 等。

6.1 文件儲存管理

每個商業類別(class)都有可能會搭配相對應的一或多份文件。當商業物件(ObjectModel)存到資料庫的同時，也要在資料庫新增一筆文件檔案(DocFile)，同時要在指定的檔案存放目錄下放置該檔案的實體，這三者缺一不可。如圖 6. 1 所示。

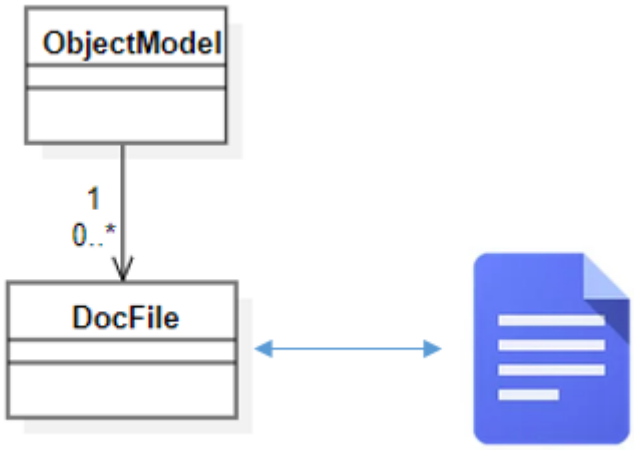

圖 6. 1 文件儲存架構

6.2 呼叫 Google API

Google 提供許多免費的 API 讓開發者串接，為了控制 API 使用的額度，Google 讓開發者可申請 API Key 金鑰，依照金鑰被呼叫的次數即可計算額度[10]。

本專案透過 Google 提到的 Web Service API 實作 Google 查詢的功能。以圖 6. 2 為例，查詢詞為「Request for information」，經過呼叫 Google API 後，列出前 10 項搜尋結果的網址如下。過程中需要注意的是，網址是不允許出現空格、「+」、「/」、「?」、「%」、「#」、「&」、和「=」等符號，必須要轉換成相對應的 URL 代碼，程式實作如圖 6. 3 所示。

```
String qry = DataFO.toUrlFormat("Request for information");
URL url = new URL("https://www.googleapis.com/customsearch/v1?key=" + key
        + "&cx=013036536707430787589:_pqjad5hr1a&q=" + qry + "&alt=json");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/json");
BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputSt

String output;
System.out.println("Output from Server .... \n");
while ((output = br.readLine()) != null) {

    if (output.contains("\"link\": \"")) {
        String link = output.substring(output.indexOf("\"link\": \"") + ("\"l
                output.indexOf("\","));
        System.out.println(link); // Will print the google search links
    }
```

Servers | Console | Problems | Search | Progress | Call Hierarchy | JUnit | AWS Explorer

<terminated> GoogleSearch.test [JUnit] C:\Java\jdk1.8.0_181\bin\javaw.exe (2018年12月30日 下午11:06:19)

```
Output from Server ....

https://en.wikipedia.org/wiki/Request_for_information
https://www.performance.gov/GEARcenter/GEAR_Center_RFI_amendment_8.28.2018.pdf
https://www.cftc.gov/sites/default/files/2018-12/federalregister121118.pdf
https://www.federalregister.gov/documents/2018/09/26/2018-20914/request-for-information-on-update
http://k12education.gatesfoundation.org/researchanddevelopment/
https://www.federalregister.gov/documents/2018/12/14/2018-27162/request-for-information-on-modify
https://www.iup.edu/procurement/howto/request-for-information-(rfi)/
https://www.consumerfinance.gov/policy-compliance/notice-opportunities-comment/open-notices/reque
https://www.thebalancesmb.com/improve-request-information-process-845348
https://grants.nih.gov/grants/guide/notice-files/NOT-OD-18-134.html
```

圖 6. 2 Google 查詢範例

```
//  特殊字符    代表含义    替换内容
//  +    URL 中+号表示空格    %2B
//  空格  URL中的空格可以用+号或者编码    %20
//  /    分隔目录和子目录  %2F
//  ?    分隔实际的URL和参数    %3F
//  %    指定特殊字符    %25
//  #    表示书签    %23
//  &    URL 中指定的参数间的分隔符  %26
//  =    URL 中指定参数的值    %3D
    public static String toUrlFormat(String _str) {
        if (_str == null)
            return "";
        return _str.replace("%", "%25").replace("+", "%2B").replace(" ", "%20").replace("/", "%2F")
                .replace("?", "%3F").replace("#", "%23").replace("&", "%26").replace("=", "%3D");

    }
```

圖 6. 3 URL 代碼處理

6.3 呼叫 AWS API

AWS 同樣提供許多免費的 API，有別於 LUIS 和 Google 透過 Web Service 的方式實作，AWS 提供了相對應的函式庫，主要有 v1.1 和 v2.0 的版本，本專案實作前者。函式庫的下載頁面如圖 6. 4 所示，下載完後，依照 4.1 提到的方式把函式庫加入專案環境即可使用。

圖 6. 4 AWS sdk for Java(來源：https://aws.amazon.com/tw/sdk-for-java[11])

本專題實作了在 Java 程式中使用 Amazon Comprehend 和 Amazon Translate 功能。以下分別簡要說明。

6.3.1 Amazon Comprehend

Amazon Comprehend 是 AWS 提供的自然語言處理服務，和微軟提供的 LUIS 相對應。同樣都有提供判斷實體(entity)和文字的情感(sentiment)分析，但 LUIS 額外提供了「意圖(intent)」的預測，而 Amazon Comprehend 則沒有。以 EKP 所需的應用中，少了意圖的判斷，儘管知道實體和關鍵字，也很難進行後續向資料庫查詢的動作。這是 EKP 最後選用 LUIS 作為諮詢機器人主要的自然語言處理機制的原因。圖 6. 5 是一用 Amazon Comprehend 偵測關鍵字和情感的範例。

```
DetectEntitiesRequest req = new DetectEntitiesRequest();
req.setLanguageCode("en");
System.out.println("req: " + req);
String text = "The ohm egg is really tender, 4-5 points; "
        + "the rest of the chicken chops and tomato sauce fried corn ham rice pop.";
req = req.withText(text);
System.out.println("----------------------detect entities----------------------");
DetectEntitiesResult result = cph.detectEntities(req);

List<Entity> entityList = result.getEntities();
for (Entity e : entityList) {
    System.out.println(e.getText() + "\t" + e.getType() + "\t" + e.getScore() + "\t" + e.getBeginOffset(
            + e.getEndOffset());
}
DetectSentimentRequest dsRequest = new DetectSentimentRequest();
dsRequest.setLanguageCode("en");
dsRequest = dsRequest.withText(text);
System.out.println("----------------------detect sentiment----------------------");
DetectSentimentResult dsResult = cph.detectSentiment(dsRequest);
System.out.println(dsResult.getSentiment() + "\t" + dsResult.getSentimentScore());
```

```
Servers Console Problems Search Progress Call Hierarchy JUnit AWS Explorer
<terminated> AwsTest.testDetect [JUnit] C:\Java\jdk1.8.0_181\bin\javaw.exe (2018年12月30日 下午11:37:53)
cph: com.amazonaws.services.comprehend.AmazonComprehendClient@2362f559
req: {LanguageCode: en}
----------------------detect entities----------------------
4       QUANTITY        0.91207063      30      31
5 points        QUANTITY        0.98267466      32      40
----------------------detect sentiment----------------------
POSITIVE        {Positive: 0.8792118,Negative: 0.0013215729,Neutral: 0.114472575,Mixed: 0.0049940757}
```

圖 6. 5 Amazon Comprehend 範例

6.3.2 Amazon Translate

Amazon Translate 是 AWS 提供的翻譯服務，實作如圖 6. 6 所示。

```
@Test
public void testTranslate() {
    AmazonTranslate translate = AmazonTranslateClientBuilder.defaultClient();
    System.out.println("translate: " + translate);
    TranslateTextRequest req = new TranslateTextRequest();
    req.setSourceLanguageCode("zh-TW");
    req.setTargetLanguageCode("en");
    req.setText("這是一間好吃的餐廳。");

    TranslateTextResult result = translate.translateText(req);
    System.out.println("result.getTranslatedText(): " + result.getTranslatedText());
}
```

```
Servers  Console  Problems  Search  Progress  Call Hierarchy  JUnit  AWS Explorer
<terminated> AwsTest.testTranslate [JUnit] C:\Java\jdk1.8.0_181\bin\javaw.exe (2018年12月30日 下午11:32:52)
translate: com.amazonaws.services.translate.AmazonTranslateClient@10683d9d
result.getTranslatedText(): It is a tasty restaurant.
```

圖 6. 6 Amazon Translate 範例

6.4 加密處理

加密技術是一個把易於讀取和了解的數據格式（原文）加以更改及轉變的過程，使其轉為不可讀取的格式（加密文本），令人看起來是一組無用及難以了解的文本。EKP 在存取使用者密碼時便實作了加密處理。採用的是 Sha-256 演算法。實作的 Java 程式碼[12]和效果如圖 6. 7 所示。

```
public static String getSHA256(String input) {

    String toReturn = null;
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        digest.reset();
        digest.update(input.getBytes("utf8"));
        toReturn = String.format("%040x", new BigInteger(1, digest.digest()));
    } catch (Exception e) {
        e.printStackTrace();
    }

    return toReturn;
}
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

uid	id	password	name	object_create_time	object_update_time
User2	user01	aad415a73c4cef1ef94a5c00b2642b571a3e5494536328ad960db61889bd9368	user01	2018-12-30T14:22:08.010	2018-12-30T14:22:08.010
User3	user02	76431fac8a187241af8f3f37156deb94732f52fb45eb07ec4f462051bd82f183	user02	2018-12-30T14:24:14.017	2018-12-30T14:24:14.017
NULL	NULL	NULL	NULL	NULL	NULL

圖 6. 7 Java 實作 Sha-256 加密演算法

6.5 部署在 AWS

此小節說明把專案部署在 AWS 虛擬機的過程。

6.5.1 EC2

本專案是 Java 應用程式，需要在 EC2 安裝 JDK，並設定環境變數，如圖 6. 8 所示。

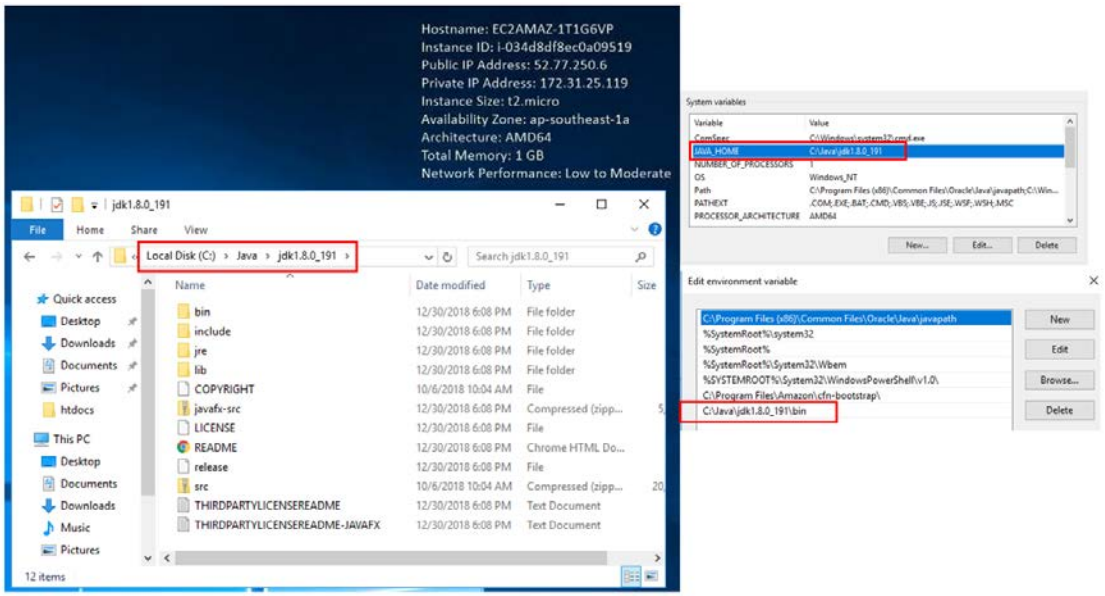

圖 6. 8 在 EC2 上安裝 Java 環境

設定完成後，接著在 EC2 上安裝 Tomcat 伺服器，並把 EKP 專案的部署檔放到 EC2 上。調整 Tomcat 伺服器的 server.xml 文檔，把 appBase 指向 EKP 專案所在的資料夾，這樣便完成設定。如圖 6. 9。

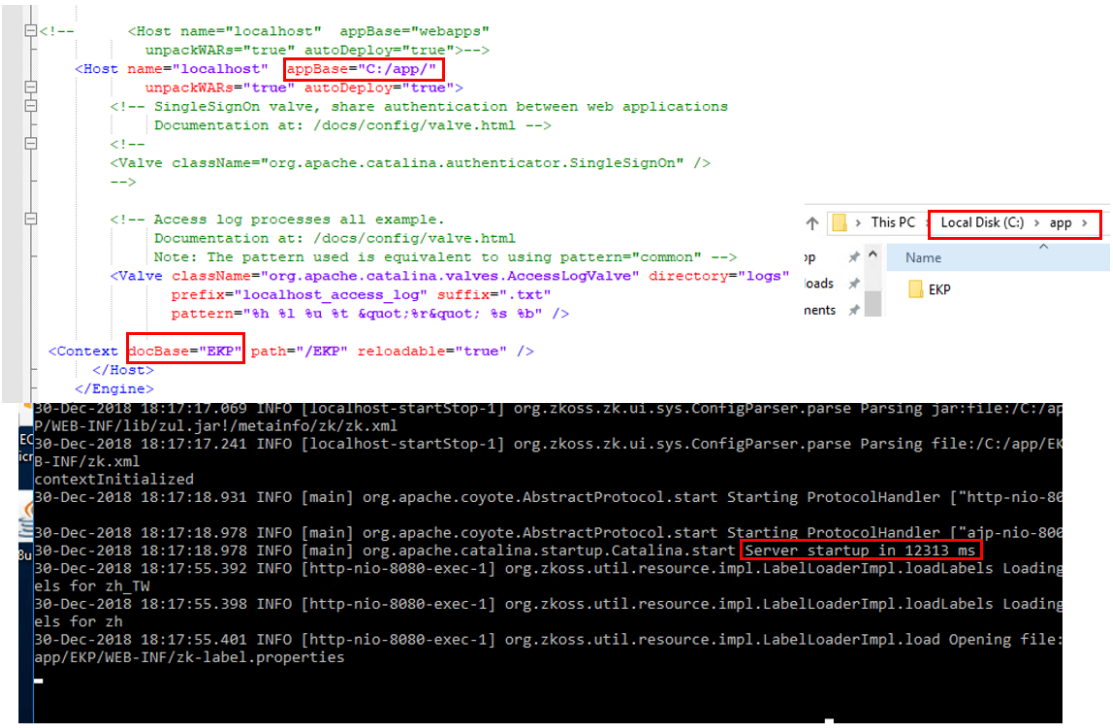

圖 6. 9 在 EC2 上設定 Tomcat 環境和 EKP 專案

6.5.2 RDS

在 AWS 的 RDS 上建出同樣的資料表，並更改程式的連線設定改連到 AWS 的 RDS。相關畫面截圖圖 6. 10 所示。

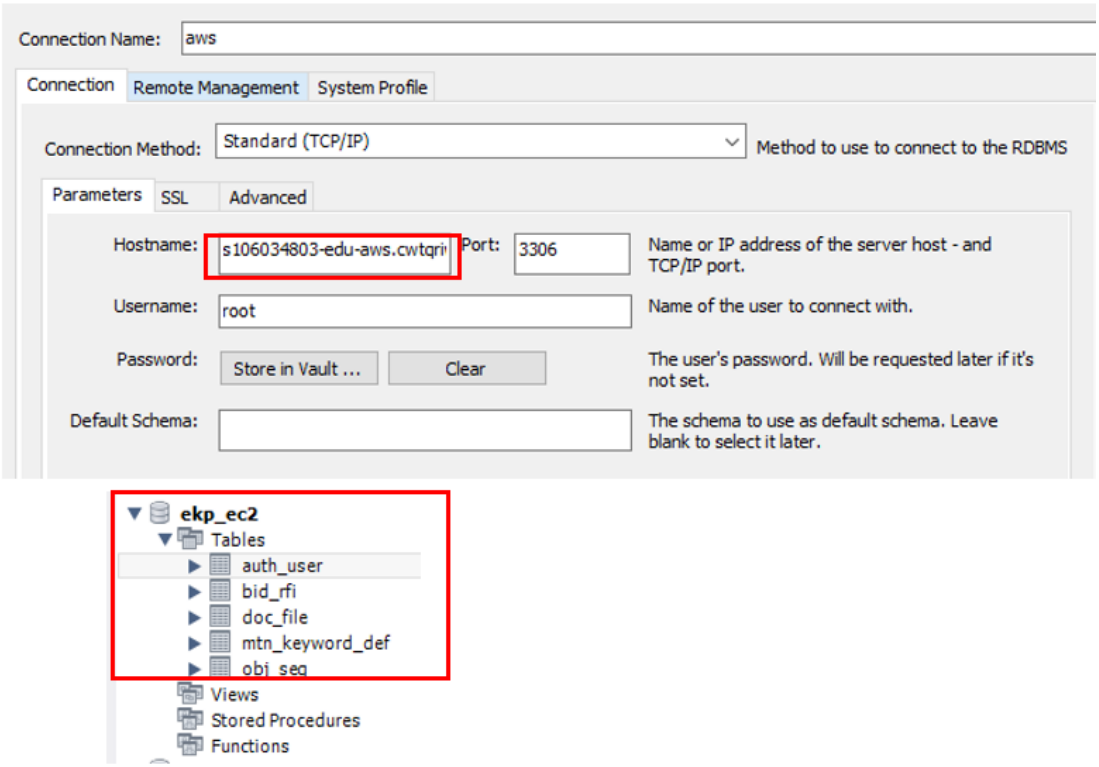

圖 6. 10 RDS 畫面截圖

7 後續研究

本章說明後續的研究方向。

7.1 平台整體架構規劃

此專案只針對了各項技術進行可行性的驗證，若要作為一個完整的資料平台，仍需要仔細設計並規劃整體架構。

7.2 Java 和 Python 平台溝通

目前主流的深度學習是透過Python實現，如何在Java程式中呼叫Python程式會是後續要克服的一大問題。另外，目前是用 Tomcat、ZK、搭配 Java 實作網頁框架，後續也可以考慮改用 Djiango 搭配 Python 實作網頁框架，這樣就不需要考慮 Java 和 Python 溝通的問題。

7.3 諮詢機器人服務框架

目前 EKP 實作的諮詢機器人僅能提供「詢問名詞定義」和「查詢文件資料」兩種服務，後續如何擴充並建構完整的諮詢機器人服務框架會是一項非常重要的課題。

7.4 加入機器學習或深度學習應用

本小節說明機器學習或深度學習可以應用的範疇。

7.4.1 自然語言處理

以目前 EKP 的應用為例，在「查詢文件資料」時，需要依「專案代碼」查詢。而專案代碼是使用者自行定義的，無法透過 LUIS 預先建立實體(entity)。如果在 LUIS 確認意圖是「查詢文件資料」時，搭配深度學習模型判斷出使用者的語句中所包含的「專案代碼」，便能更精準地提供服務。

7.4.2 文件分析

將來在 EKP 上會有大量的文件資料，透過非監督式學習萃取出文件中的知識將會是一大課題。

7.4.3 自然語言生成

目前 EKP 實作的諮詢機器人回覆給使用者的語句，仍未經過自然語言處理。從資料庫取得的資料，會直接丟回給使用者；若是在特定情境下的回答，還是依賴預先建立好的範本。後續如果能透過深度學習實作自然語言生成，將會讓諮詢機器人的回答更擬人化。

7.4.4 中文情境

LUIS 提供的自然語言處理只能使用英文，如何要建立中文的自然語言處理機制，將會是後續需要努力的方向。

參考資料

[1] https://zh.wikipedia.org/wiki/Java

[2] https://zh.wikipedia.org/wiki/Apache_Tomcat

[3] https://zh.wikipedia.org/wiki/ZK

[4] https://docs.microsoft.com/zh-tw/azure/cognitive-services/luis/what-is-luis

[5] https://www.newtype.com.tw

[6] http://new.lyserp.com.tw

[7] https://www.zkoss.org/download/zk

[8] http://tomcat.apache.org

[9] LUIS API https://westus.dev.cognitive.microsoft.com/docs/services

[10] Google API Key https://www.wfublog.com/2018/12/google-api-key-activate-quota.html

[11] https://aws.amazon.com/tw/sdk-for-java

[12] https://stackoverflow.com/questions/5531455/how-to-hash-some-string-with-sha256-in-java