**靜宜大學資訊學院 畢業專題口試 系統文件規格書**

後端框架設計與微服務實作

實驗室名稱：高性能計算與網路實驗室（主顧509）

指導教師：翁添雄 副教授

專題學生：資工三B 410516405 鍾騰

目錄

[1. 前言 3](#_Toc42417973)

[2.系統特色 4](#_Toc42417974)

[3.系統功能 5](#_Toc42417975)

[4. 使用對象 9](#_Toc42417976)

[5. 使用環境 10](#_Toc42417977)

[6. 開發工具 10](#_Toc42417978)

[7. 系統畫面 10](#_Toc42417979)

[8. 成本分析 11](#_Toc42417980)

[9. 現況與未來發展 11](#_Toc42417981)

# 1. 前言

　　在現今資訊爆炸、各路創業者磨刀霍霍的年代。一個idea是否能證明自己有價值，很大程度上左右了提出這個idea的人或團隊，日後發展的可能性。不論是公司的內部專案，或是正在尋找資金來源的新創團隊都可能面臨這個問題。

　　在資訊領域也是如此，新的產品與idea能否從競爭激烈的市場環境或資源緊張的公司內部存活下來，在很長的時間內可能都是需要被迫切關注的議題之一。

　　不幸的是，以商務邏輯的發源地，後端（Back-end）來說。在這個本該要最專注於產品的核心競爭力功能、核心商務邏輯的地方，卻也時常發現有開發人員花費很多寶貴的注意力，去重新製造輪子的憾事出現。

　　在較底層，例如接收與回覆http請求、JSON的格式化、連接資料庫、甚至是單元測試，大部分的後端程式框架都能發揮應有的作用。協助將接收、回覆、格式化、連接等等的瑣碎實作細節包裝成各種一目了然的抽象介面，讓開發者得以從這些零碎的泥沼中抽身，真正的開始寫自己的程式。

　　但在較高級的功能中，各類框架能提供的協助大多戛然而止。如會員、文章、檔案、權限管理等各類常見功能，多數框架對這部分幾乎很難有較多的著墨。最多可能也只是提供ORM（Object-relational mapping）讓開發者快速建立與使用資料庫物件（如Java中的Hibernate、Python中的Django）或是提供樣板程式碼來快速建立最基礎的相關系統（如PHP中的Laravel就尤其擅長此道），剩下的就得慢慢的自己搞定。

　　會有這種狀況，其實箇中源由不難理解，就是每個人、每個產品對於這些基礎功能都有自己不同的需求。所以很難量身打造，卻又不失其彈性。否則使用Wordpress等一站式解決方案即可，也不會需要勞駕開發者開新專案來寫了。

　　因此，我們（雖然小組目前只有一個人）也在思考，是否能針對上述程式框架與一站式方案之間的需求真空，來設計後端程式框架的解決方案。以全力協助開發人員在最短的時間內將專注力移至其專案的核心價值、商務邏輯，而不要花過多時間在基礎、常見的功能上。

# 2.系統特色

● 具備快速開發產品原型能力的框架

　　本系統將主要面向需要快速開發產品原型的使用者，使其能夠在相對短的時間內開發出實際可用且具備擴充性的產品後端原型，來驗證該概念（此泛指產品的核心理念）是否具備發展的潛力和價值。

　　值得一提的是，本系統從設計、程式碼品質到壓力測試，皆擁有希望以正式生產環境為標準的野心。開發人員在開發完原型後亦可選擇以本系統做為未來上線到生產環境的基礎框架。

● 彈性較高的開發與測試環境

　　由於本系統將所有除初始化以外的商務邏輯都抽象為一個個步驟與步驟的組合，所以在開發階段時能夠提供許多相當彈性的測試功能。

　　例如在本系統的執行期間，從本系統的後台介面以自訂參數動態的呼叫某個步驟 / 步驟組合，並且觀察其每個階段的輸出與運作數據等，或甚至在後台直接新增並組裝步驟組合。

　　讓開發人員得以專注於產品的抽象邏輯，而不需要面對瑣碎重複的具體實作。

● 具備自己的生態體系以支持開箱即用

　　開發人員或組織在開發專案時，常常會因需求的細節不同而將類型相近的功能作為特例重複撰寫，例如帳戶、權限、文章、檔案管理或通知推播等，容易造成許多時間上無謂的浪費。

　　本系統從設計上將以模組化為依歸，並且積極開發和維護相容於本系統的通用功能模組，盡可能地讓開發人員可以用設定代替開發這些同質性極高的功能。並且擁抱開源社群的精神，本系統也會對於第三方開發的功能模組抱持開放和支持的態度。

● 設計上保留未來開發本系統微服務版本的可能

　　本系統會盡可能避開日後阻礙開發微服務版本可能的設計，例如模組之間過於低耦合高內聚、或不當共享全域變數等等。

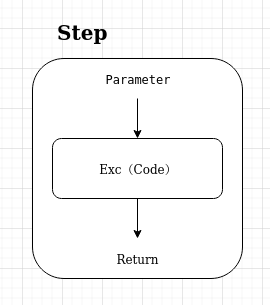
# 3. 系統功能

　　承前所述，本系統的核心價值之一是讓使用者能夠快速靈活的開發產品原型，並且在驗證價值的過程中，能夠快速對於新問題和新需求產生回饋。

　　本系統將由Step（步驟）、Job（工作）與Module（模組）組成，以下為其大致的運作原理：

● Step（步驟）

　　具有輸入與輸出定義的Class，為框架中的基本執行零件。

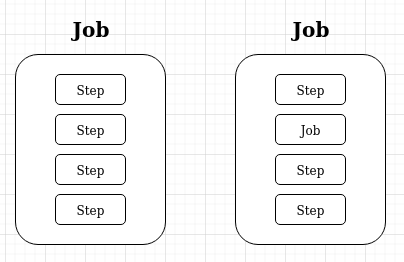


以下為一個Step的簡易程式碼範例：



● Job（工作）

　　定義一組由Step或Job組成的工作流程。



● Module（模組）

　　負責成體系的Step與Job建立與管理。如帳戶模組中就會包含所有與帳戶相關的功能的所有Step與Job，並負責與框架註冊這些元件。

　　通常會以外部Jar的形式放入框架執行目錄底下的Module資料夾中，並由框架於執行時（Runtime）讀取與初始化。

　　以下是使用標註（annotation）註冊Module，並且在其中註冊Job的簡易例子：

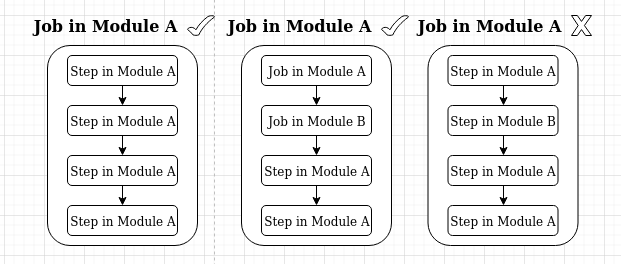


希望採取此架構是為了建立以下優勢：

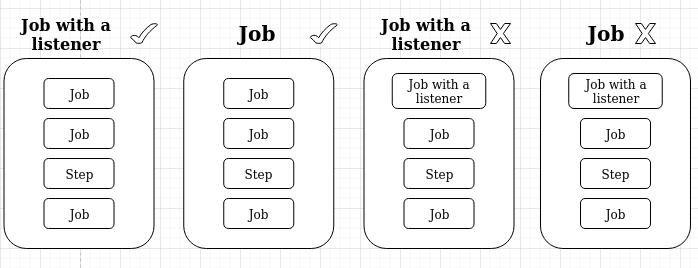
1. 統一規格，增加各個元件（Job、Step、Module）之間的重複利用性
2. 在框架後台（為網頁介面）可以訪問各個元件的運作效率、輸入輸出、統計圖表及各項開發人員功能
3. 在框架執行時（Runtime）動態調用個別Job或Step，並可為其手動給予輸入，來測試運作是否符合預期。此功能於生產環境中預設關閉。
4. 在框架執行時（Runtime）修改應用程序的運作邏輯。例如在某個Job中刪除權限檢查的環節，並檢視其運作結果。此功能於生產環境中預設關閉。

● 其他運作細節

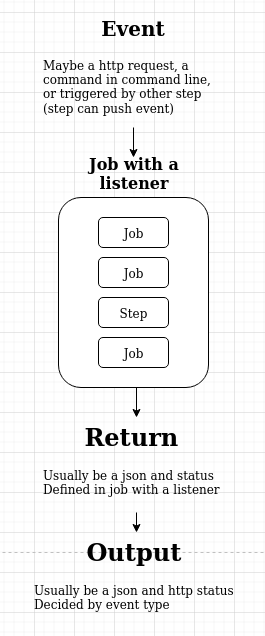
● Job可以設定為public來跨Module使用，但Step不行  
 　（根據低耦合高內聚的原則，Step等級的細節不應該暴露於Module之外）



● 最外層的Job可以附加監聽器來監聽事件，但也可以都不加



● 最後，以下是一整串請求的完整示意圖。另外，在Event進入Job以及Return到Output之間都會經過包裝或格式化，讓同一個Job可以同時處理不同來源但性質一樣的工作。

****

# 4. 使用對象

任何具有基礎Java程式設計能力的開發者，可參考本框架文件進行開發。

# 5. 使用環境

※開發

1. Gradle或Maven（用以取得框架依賴）
2. JDK 8 以上

※運作

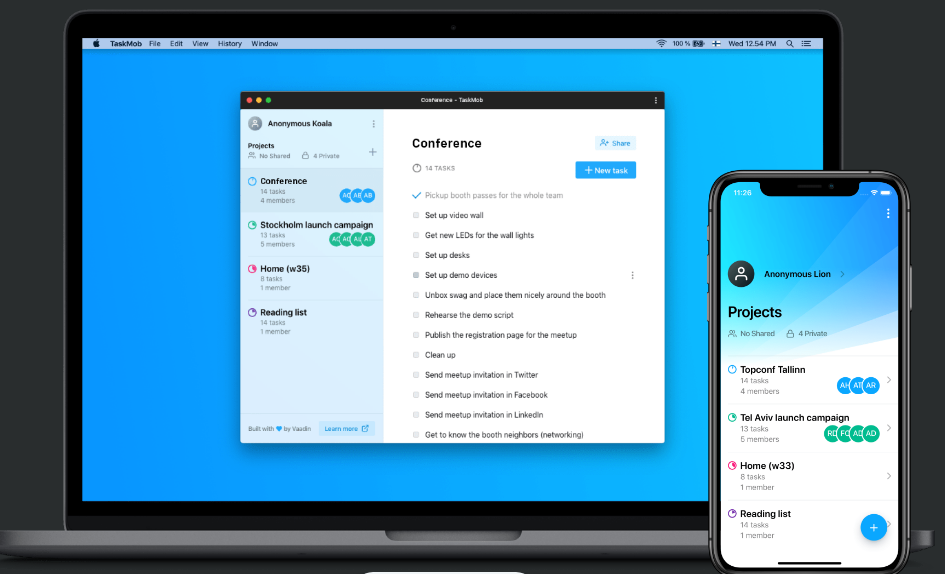
1. JRE 8 以上

# 6. 開發工具

* 工具
  + JDK 8
  + Git（版本控制系統）
  + Vim（文字編輯器）
  + Gradle（依賴管理與自動化）
  + IntelliJ IDEA（整合開發工具）
* 平台
  + Proxmox Virtual Environment（虛擬化環境）
  + Ubuntu desktop & server（框架的開發環境）
  + Docker（虛擬化容器）

# 7. 系統畫面

暫無。但本框架後台預計使用Vaadin框架進行開發，以下為其官網提供的示意圖：



# 8. 成本分析

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 項 目 名 稱 | 說 明 | 單位 | 數量 | 單 價 | 小 計 | 備 註 |
| 臺幣(元) | 臺幣(元) |
| 筆記型電腦 | 遠端開發與資料查詢 | 部 | 1 | 40000 | 40000 | 自行負擔 |
| 雷射印表機 | 文件整理及列印等 | 部 | 1 | 10000 | 10000 | 由系上實驗室提供 |
| 個人主機 | 開發環境建構與框架測試 | 部 | 2 | 10000 | 20000 | 由系上實驗室提供 |
| 個人主機 | 開發環境建構與框架測試 | 部 | 2 | 10000 | 20000 | 自行負擔 |
| NAS主機硬碟 | 供NAS 主機使用的硬碟 | 顆 | 3 | 2000 | 6000 | 自行負擔 |
| 網域註冊 | 註冊網域供服務使用 | 年 | 1 | 709 | 709 | 自行負擔 |
| 雜支費 | 所有可能的雜支 | 批 | 1 | 500 | 500 | 自行負擔 |
| 共 計 | | | | | 83209 |  |