

自動化共通資訊模型之設計與實現

Automatic System for Common Information Model Realization

王建堯	劉志翔	楊偉楨
Chien-Yao Wang	Chih-Hsiang Liu	Wei-Jen Yang
資訊工業策進會	資訊工業策進會	資訊工業策進會
數據科技與應用研究所	數據科技與應用研究所	數據科技與應用研究所
工程師	工程師	工程師
jimmywang@iii.org.tw	ansonliu@iii.org.tw	wjyang@iii.org.tw

王致超
Chih-Chao Wang
資訊工業策進會
數據科技與應用研究所
工程師
chihchaowang@iii.org.tw

摘 要

本論文設計與實作一個自動化共通資訊模型系統。此系統透過結合共通應用程式介面(API)與持續整合技術，達到自動化建立資訊模型與即時更新之目的。將商業應用藉由流程物件(artifacts)分層與定義此資訊模型。透過流程物件的整合，此系統可運用持續整合技術快速並自動同步資訊模型的更新，達到與傳統建構資訊模型建構之技術差異並且能夠擁有更高之適應性。此外，此系統能夠快速並簡易的整合至雲端服務(cloud service)，達到應用實作自動化整合之可行性。

關鍵詞：流程物件、自動化、共通資訊模型、持續整合技術

Abstract

In this paper, we propose an automatic system for realizing the common information models. The proposed system, which optimally combines the application programming interface (API) framework and continuous integration techniques, can achieve the purpose of automatic common information model realization according to the defined artifacts. Different from the traditional information model realization systems, the proposed system can synchronously update the information models when the artifacts are modified due to the application of the continuous integration techniques, leading to the automation and high efficiency benefits. In addition, the proposed system can be easily applied to provide the cloud service. The implementation of the real applications demonstrates the practical applicability and feasibility the proposed automatic system.

Keywords: artifacts, automatic system, common information models, continuous integration technique

1. 前言

現今網路應用(web application)廣泛運用於各個領域，使用者的容易使用，已經成為現今不可或缺的使用模式。故有許多的網路應用設計模式相繼提出，其中以嵌入式命令語言(imperative embedding)[1]的網路應用設計模式為傳統大眾所廣泛使用。但透過此類設計模式，相網路應用開發者則可能必須花費許多的時間來設計與除錯網路應用程式，並且往後欲更新或新增功能與軟體時，又可能再花費相對應的開發時間。

因此，網路應用的設計模式強調的應該為如何有效率且容易的建立一個富有彈性的應用平台。其中以 Leff 等人所提出的 Web Relational Blocks (WebRB)[2]平台，一個容易檢視的虛擬資料流的設計模式，成為受歡迎的傳統開發平台。WebRB 是由以 Relational Blocks (RB)[3]為基礎衍生的開發模式，他定義了許多預先可供開發者使用的工具來快速的生成後端資料管理與設計使用者介面。此種設計方式，有效的解決嵌入式命令語言網路應用開發時程問題，但此類模式只有提供了基本的前端資料管理功能，在後端邏輯變更時卻無法免除大量的變更開發邏輯，故往往設計者變動前端資訊邏輯時，後端的對應服務卻無法快速對應更動。

因此，將後端服務邏輯單獨撰寫開放，為近年來為各網路應用所使用之開發模式，透過開放 API (Application Programming Interface)的快速增長與盛行，使得 API 的運用成為許多企業產生新的收益模式。API Economy[4]的低成本開發模式日漸成為產業開發網路應用的新模式[5][6]。其中 Janes 等人[7]提出透過了共通的 API 介面的易讀與建立，能夠找出一個共同的 API 模型架構定義。而因後端獨立之 API 開發模式盛行，將後端服務邏輯獨立建立已成為現今網路新興的開發模式，但要如何快速或更一步的自動化後端服務，並且能因應常態性變動又成為此開發模式的一個討論的議題。其中以 Loopback[8]架構，則為此目的所產生之開發平台，它以 Node.js[9]為基底之後端服務生成架構，稱為 API 實現層技術。透過物件的定義，產生相對應之 Node.js 後端服務。但是對於自動化的執行，此架構則並無聚焦於此。故本論文希冀透過持續整合(Continuous Integration, CI)與事件驅動(Event - Driven)之技術整合，達成此自動化開發平台之設計。

對於網路應用服務的開發，此平台設計開發運用智慧系統服務技術(Smart System Service)[10]為基底，進行網路應用需求分析與目標設定，進而定義出此網路應用之事業流程(Business Process)，事業流程為整體網路應用服務之功能需求與結構性事業活動的拓撲關係。而經由事業流程之定義，可具體引導出此網路應用之流程物件清單(Artifact List)。透過流程物件清單的定義輸入，結合前述之持續整合與事件驅動之技術，設計與實作此自動化

共通資訊模組開發平台。

2. 平台設計

本章說明此自動化系統架構設計與詳細模組，此平台透過持續整合伺服器(Continuous Integration Server)並結合 API 實現層技術設計此平台基底架構。透過前述網路應用需求分析方法所導出之流程物件清單(Artifact List)，進而轉化為此自動化平台系統之輸入物件與前期需求，而經由此平台之實現與產出，網路應用服務設計者可專注於商業邏輯與流程物件定義的開發，將繁複的後端服務更新與版本控管交由此平台管控。

此平台依據功能設計可分為四個模組，分別為流程物件轉譯模組、持續整合伺服器模組、自動化串接模組與後端服務建構模組，如圖一平台設計架構圖所示，以下分段詳述各模組功能。

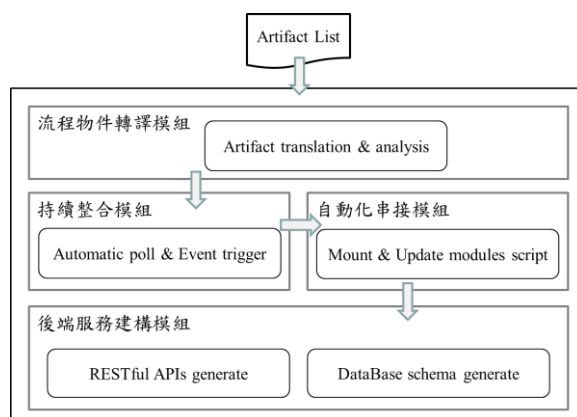


圖 1、平台設計架構圖

2.1 流程物件轉譯模組

經由服務設計人員設計與輸入流程物件至此模組，當服務設計人員新增或後續更新流程物件時，透過此模組分析流程物件之差異化並同步轉譯流程物件中的參數，其中擷取流程物件中之名稱與型別，導向至後端服務建構。

2.2 持續整合伺服器模組

透過軟體工程中之持續整合 (Continuous Integration, CI) 技術為基礎，架構自動偵測與事件觸發機制。經由此技術架設持續整合伺服器，透過此伺服器中的定期自動輪詢(Automatic poll)機制，決定自動化串接模組需觸發與掛載的程式化腳本，並能夠依據優先順序管控整體平台狀態與執行過程。藉由此模組之整合與排程機制，有效達到平台效能優化與自動化機制。

2.3 自動化串接模組

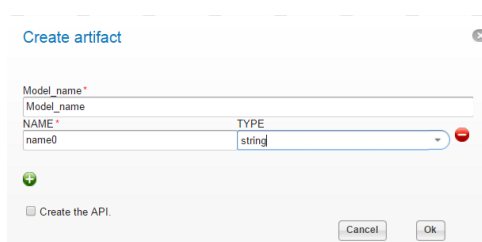
由持續整合模組之輪循與排程機制進入此自動化串接模組，此模組為許多串接程式碼腳本之集合。依據不同的指令更新與起始對應之程式碼腳本。

2.4 後端服務建構模組

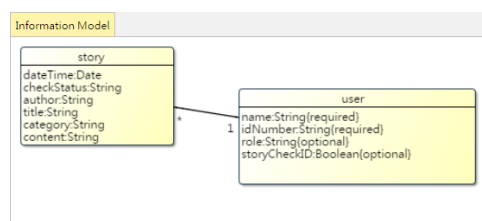
此模組經由自動化串接模組之呼叫執行對應之程式化腳本。此後端服務建構包含資料庫的自動串接與建立與此網路應用資訊模組的基本共通 API 建立，其中包含新增、查詢、更新、刪除與搜尋特定(create、read、update、delete and find)等介面方法。

2. 實作

此章節描述與說明各模組開發介面與實作方式，並驗證此資訊模型自動開發平台之實際運用與可行性。應用服務設計者將需求轉換為企業流程與其對應之流程物件，透過此平台所開發之設計介面，定義與輸入欲開發之網路應用需求流程物件，如圖二所示，並可依照需求定義各流程物件之關聯與對應關係，關聯編輯介面如圖三所示。



圖二、流程物件定義介面



圖三、流程物件關聯設計介面

服務設計者完成流程物件清單(Artifact List)編輯後，經由此平台之自動輪詢(Automatic poll)與自動化串接機制，定期將各專案之流程物件清單變更之處做新增與更新。其中輪詢排程圖像化介面如圖四所示。而透過持續整合伺服器可排程控制各流程物件的後端服務建置，持續整合模組依據流程物件清單的定義與與其他流程物件之關聯，擇定需排程進入後端服務模組建置的相關程式化腳本，後端服務建置模組自動於資料庫創建各流程物件之資料綱要(Data schema)與相關設定。並一併於平台中自動創建此網路應用對應之共通資訊模組(create、read、update、delete 和 find)方法，如圖五所示。



圖四、輪詢排程圖



stories	
POST	/stories
GET	/stories
PUT	/stories
PUT	/stories/{id}
HEAD	/stories/{id}
GET	/stories/{id}
DELETE	/stories/{id}
GET	/stories/{id}/exists
GET	/stories/count
GET	/stories/findOne
POST	/stories/update

圖五、共通資訊模組存取介面

網路應用設計者只需定義流程物件，即可透過此平台快速建構此網路應用之共通資訊模組，並包含應用程式介面與後端資料存取介面。快速且靈活的加速網路應用開發。而假如需要修正此網路應用相關流程物件內容或關聯時，網路應用設計者也只需透過平台介面修正相關內容，平台即可同步且快速建立與佈署，修正更新之後端服務介面。故網路應用設計者可減少與後端服務開發者之繁複溝通，專注於商業邏輯與物件的定義設計。

4. 結論

透過此自動化平台可有效改善服務設計人員之與後端 IT 服務開發之溝通方式。傳統式開發如需修正單一參數或新增資訊物件時，往往需經由層層修正與重新佈署，造成溝通繁複與可能之經常性人員失誤。

經由此平台串接，服務設計人員可專注於服務邏輯與對應之資訊物件之設計。其中繁複之後端服務的實現，則由此自動化平台介入自動建立。設計者不需反覆重新修正對應服務串接程式，達到靈活且快速之開發模式。

致 謝

本研究依經濟部補助財團法人資訊工業策進會「104 年度系統服務與事業轉型計畫(3/4)」辦理。感謝經濟部技術處支持資訊工業策進會數據科技與應用研究所進行自動化共通資訊模型平台發展之研發。

參考文獻

- [1] Avraham Leff, James T. Rayfield, “WebRB: a different way to write web applications,” IEEE Internet Computing, vol. 12, no. 3, 2008, pp. 52–61.
- [2] Avraham Leff, James T. Rayfield, “WebRB: evaluating a visual domainspecific language for building relational web-applications,” ACM SIGPLAN Notices, vol. 42, no. 10, 2007, pp. 281–300
- [3] Daniel D. Hils, “Visual languages and computing survey: dataflow visual programming languages,” J. of Visual Languages and Computing, vol. 3, no. 1, 1992, pp. 69–101.
- [4] Israel Gat ,Agile, Giancarlo Succi “A Survey of the API Economy,” Agile Product & Project Management , Executive Update Vol. 14, No. 6., 2013
- [5] Gat, Israel, Remencius, Tadas, Sillitti, Alberto, Succi, Giancarlo, Vlasenko, Jelena, “API Economy: Playing the Devil’s Advocate,” Cutter IT Journal 26(9), 6–11, 2013
- [6] Remencius Tadas, Succi Giancarlo, “Tailoring ITIL for the Management of APIs,” Cutter IT Journal 26 (9), 22-29, 2013
- [7] Andrea Janes, Tadas Remencius, Alberto Sillitti, Giancarlo Succi “Towards Understanding of Structural Attributes of Web APIs Using Metrics Based on API Call Responses,” OSS 2014: 83-92.
- [8] Loopback, The Node.js API Framework, <http://loopback.io/>
- [9] Node.js, JavaScript runtime built on Chrome's V8 JavaScript engine, <https://nodejs.org/en/>
- [10] 王可言、林蔚君、周志勳、蔡其杭、余承歡、徐熙妙, “以智慧系統服務實現智慧綠建築管理解決方案”, Symposium of Enterprise Architecture and Information Technology(SEAIT), Taipei, Taiwan, Oct 27, 2012.