

國立交通大學

資訊學院 數位圖書資訊學程

碩士論文

圖書館書後附件隨選系統之研究

A Study of Disc-on-Demand System in The Academic Libraries



研究生：張文雄

指導教授：黃明居 博士

中華民國九十九年二月

圖書館書後附件隨選系統研究與實作

A Study of Disc-on-Demand System in The Academic Libraries

研究生：張文雄

Student：Wen-Hsiung Chang

指導教授：黃明居

Advisor：Dr. Ming-Jiu Hwang

國立交通大學

資訊學院 數位圖書資訊學程



Submitted to College of Computer Science
Natural Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Digital Library
February 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年二月

圖書館書後附件隨選系統之研究

研究生：張文雄

指導教授：黃明居

國立交通大學資訊學院 數位圖書資訊學程碩士班

摘要

目前國內大學圖書館仍將管理紙本書籍的方式套用於書後附件(CD Attached with Books)的管理，因此衍生出陳列、借閱與遺失賠償方面的種種問題。本研究建置一套書後附件隨選系統(Disc-On-Demand System, DOD)解決圖書館在書後附件管理上面臨的難題。系統的核心為一組基於超文本傳輸協議(Hyper Text Transfer Protocol, HTTP)與可延伸標記語言(Extensible Markup Language, XML)的表象化狀態轉變(Representational State Transfer, REST)風格應用程式介面(Application Programming Interface, API)，結合用戶空間文件系統(File Space on User System, FUSE)技術達成以檔案總管(Explorer)為圖形化使用者介面(Graphical User Interface, GUI)的效果，讀者不需另外學習新的使用者介面操作方式。本研究所設計與研發的書後附件隨選系統不但解決圖書館書後附件相關問題，也解決現存書後附件管理系統的缺點，本系統登入即可使用，不需讀者自行搜尋書後附件、不需下載光碟映像檔、不需另外安裝虛擬光碟程式並可控管書後附件被取得的數量。經過測試使用本系統讀取書後附件速度甚至比使用光碟機讀取快七倍。

關鍵字：圖書館、書後附件、表象化狀態轉變、應用程式介面、設計模式

A Study of Disc-on-Demand System in The Academic Libraries

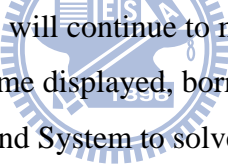
Student : Wen-Hsiung Chang

Advisor : Dr.Ming-Jiu Hwang

Degree Program of Computer Science

National Chiao Tung University

Abstract



At present, Academic libraries will continue to manage the way of books for the CD Attached with Books, thus derived some displayed, borrowed and lost compensation problems. This paper deploy a Disc-on-Demand System to solve the management of libraries in problems of CD Attached with Books. Core of the system is a Representational State Transfer (REST) style Application Programming Interface (API) based on the Hyper Text Transfer Protocol (HTTP) and Extensible Markup Language (XML), this paper also used the API and File Space on User System (FUSE) technology to make the Windows Explorer as GUI. Readers do not need to learn another user interface mode of operation. Disc-on-Demand System which can control the access amount to CD Attached with Books and can be used immediately when log in ,no need to search CD Attached with Books, download the image file and install a virtual CD-ROM application not only solve all above problems but also the exist system problems. Been tested using Disc-on-Demand System to read CD Attached with Books seven times faster than the use of CD-ROM drive.

Keyword: Library, CD Attached with Books, Representational State Transfer, Application Programming Interface, REST, API, Design Pattern

誌謝

如果試著將該感謝的人分類

第一位就是黃明居老師；其他的人就是其他，也只能是其他

黃老師謝啦！

當我毫無頭緒時，您總是能給我適當的方向，讓我少走了很多彎路

在我開始鑽牛角尖後，您總是能拉我一把，幫我再次回歸正途

有您的指導，寫論文的過程不再是辛苦與折磨，而是享受與成長

最後在您的督促與鼓勵下，果然有驚無險、一鼓作氣順利完成本論文

能當您的學生是我最大的福氣，感謝！真的謝謝！



還要感謝柯老師與林老師悉心地審查與建議，讓本論文更臻完美

感謝玉菱、莉池、媛媛協助測試系統

感謝圓淑在兩年求學過程中多次同組，患難與共

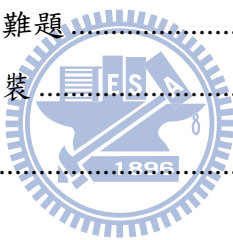
最後要感謝親愛的老婆怡萍，在我撰寫論文期間的體諒與陪伴

謝謝大家

目錄

中文摘要	I
英文摘要	II
誌謝	III
目錄	IV
圖目錄	VI
表目錄	VIII
一、 緒論	1
1.1 研究背景與動機	1
1.2 研究目的	4
1.3 研究範圍	5
1.4 研究流程	6
1.5 論文架構	8
二、 文獻探討與應用工具	9
2.1 文獻探討	9
2.1.1 圖書館書後附件之相關研究	9
2.1.2 應用程式介面	12
2.1.3 表述化狀態轉變	16
2.1.4 設計模式	18
2.1.5 用戶空間文件系統	22
2.1.6 全自動區分計算機和人類的圖靈測試	24
2.2 應用工具	25
2.1.1 Dokan Library	25
2.1.2 UrlRewriter.NET	26
三、 系統分析與實作	29
3.1 系統架構	29

3.2	應用程式介面實作	34
3.2.1	適配器模式	34
3.2.2	Token 機制	36
3.2.3	書後附件檔案儲存方式	37
3.2.4	應用程式介面提供的服務	38
3.2.5	應用程式介面設計難題	49
3.3	館員上傳程式實作	50
3.4	圖形化使用者介面實作	52
3.4.1	使用者介面的抉擇	53
3.4.2	登入畫面	55
3.4.3	查看目前借閱紀錄畫面	56
3.4.4	選擇書後附件畫面	57
3.4.5	無縫接成檔案總管	58
3.4.6	使用者介面設計難題	59
3.5	使用者介面發佈與安裝	60
四、	系統評估	63
4.1	系統整體表現	63
4.2	使用者介面 TEST CASE 測試	66
五、	結論與建議	69
5.1	結論與貢獻	69
5.2	未來研究方向	71
	參考文獻	72
	附錄一 TEST CASE 測試結果	76



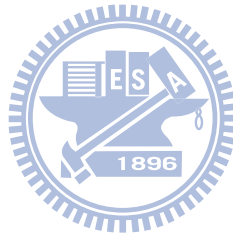
圖目錄

圖 1-1 浩然圖書館歷年書後附件累計館藏量	2
圖 1-2 書後附件借閱流程	2
圖 1-3 書後附件歸還流程	3
圖 1-4 書後附件隨選系統示意圖	5
圖 1-5 研究流程圖	7
圖 2-1 文獻整理	9
圖 2-2 中國石油大學隨書光盤管理系統	11
圖 2-3 eBay API 功能示意圖	13
圖 2-4 Web-based 系統	15
圖 2-5 使用者-伺服器系統	15
圖 2-6 以 API 實作 Web-Based 與使用者-伺服器兩種模式的系統	16
圖 2-7 現實世界中的 Adapter	19
圖 2-8 使用者介面—DOD API—圖書館自動化系統連接示意圖	20
圖 2-9 使用者介面—DOD API—多個圖書館自動化系統連接示意圖	20
圖 2-10 網際網路上 Proxy	21
圖 2-11 DOD 系統的 Proxy	22
圖 2-12 FUSE 示意圖	23
圖 2-13 Dokan Library 示意圖	26
圖 2-14 UriRewriter.NET 運作示意圖	28
圖 3-1 讀者與圖書館使用 DOD 系統必備條件	29
圖 3-2 DOD 系統提供的三個軟體	30
圖 3-3 DOD 系統架構圖	30
圖 3-4 分離伺服器示意圖	31
圖 3-5 GUI 與 DOD API 溝通腳本	32
圖 3-6 DOD 系統詳細架構示意圖	33
圖 3-7 NCTUAdapter 設計流程圖	34
圖 3-8 User、Record、List<Record>與 Rights 類別圖	35

圖 3-9 IAdapter 介面類別圖.....	35
圖 3-10 NCTUAdapter 類別圖.....	36
圖 3-11 Token 機制示意圖.....	37
圖 3-12 書後附件檔案儲存方式.....	38
圖 3-13 DOD API 組成圖.....	39
圖 3-14 DOD API 之帳號密碼認證服務流程圖.....	41
圖 3-15 Token 之 XML 格式示例與說明.....	42
圖 3-16 圖形驗證碼.....	43
圖 3-17 DOD API 之取得讀者目前借閱紀錄服務流程圖.....	44
圖 3-18 借閱紀錄 XML 格式示例與說明.....	45
圖 3-19 DOD API 之取得書後附件目錄與檔案結構服務流程圖.....	46
圖 3-20 書後附件檔案結構 XML 格式示例與說明.....	47
圖 3-21 下載書後附件檔案.....	48
圖 3-22 Uploader 運作流程圖.....	50
圖 3-23 Uploader 設定畫面.....	51
圖 3-24 Uploader 運行畫面.....	51
圖 3-25 HTML 內容示例.....	53
圖 3-26 第一次要求檔案.....	59
圖 3-27 僅下載需要的檔案區塊.....	59
圖 3-28 直接由緩存提供檔案.....	60
圖 3-29 GUI 安裝畫面.....	61
圖 3-30 GUI 安裝程式運作流程圖.....	62
圖 4-1 校內大檔案複製結果折線圖.....	64
圖 4-2 校內小檔案複製結果折線圖.....	65
圖 4-3 校外大檔案複製結果折線圖.....	65
圖 4-4 校外小檔案複製結果折線圖.....	66
圖 4-5 Test Case 方法流程圖.....	67

表目錄

表 1-1 圖書館書後附件管理與流通問題列表.....	3
表 3-1 DOD API 功能列表.....	52
表 3-2 登入畫面.....	55
表 3-3 查看目前借閱紀錄畫面.....	56
表 3-4 選擇書後附件畫面.....	57
表 3-5 無縫接成檔案總管.....	58
表 4-1 Test Case 測試項目表.....	68



一、緒論

本章描述本研究的動機、目的與希望解決的問題。1.1 節說明研究背景與動機，1.2 節為研究目的，以圖書館管理書後附件遭遇的困難點為主要考量因素，建構一套資訊系統以解決目前的問題，1.3 節為研究的問題與範圍，1.4 節則說明本研究的研究流程，1.5 節為本研究的論文架構。

1.1 研究背景與動機

隨著光碟機的普及與光碟製作成本降低，越來越多的書籍在販售時，除了提供紙本外也隨書附贈光碟。例如：程式設計相關書籍在光碟中提供程式原始碼；美工相關書籍在光碟中收錄圖片素材；語言學習相關書籍在光碟中則是影像與聲音檔等等。其目的在於補充紙本內容、節省讀者收集相關檔案時間，以及提供讀者除了紙本閱讀之外不同感受的加值服務，本文將此種光碟，稱為書後附件 (CD Attached with Books)。

書後附件因其容易被複製，體積小與易失竊等等特性，目前大學圖書館的作法均採用架式存放在流通櫃檯附近的空間的方式，以利館員在讀者要求借閱書後附件時能快速取得。然而隨著時間累積，書後附件館藏量增加，空間需求也同步增加。圖 1-1 為交通大學浩然圖書館自西元 1999 至 2008 年，每年書後附件館藏量累積數量。由圖中可知至 2008 年 12 月為止，全館書後附件館藏量已達 23047 片。平均以每年 2000 片的速度穩定地成長。以內插法 (Interpolation Method) 估算，2011 年書後附件館藏量即將突破 30000 大關，存放空間的問題亦日趨嚴重。

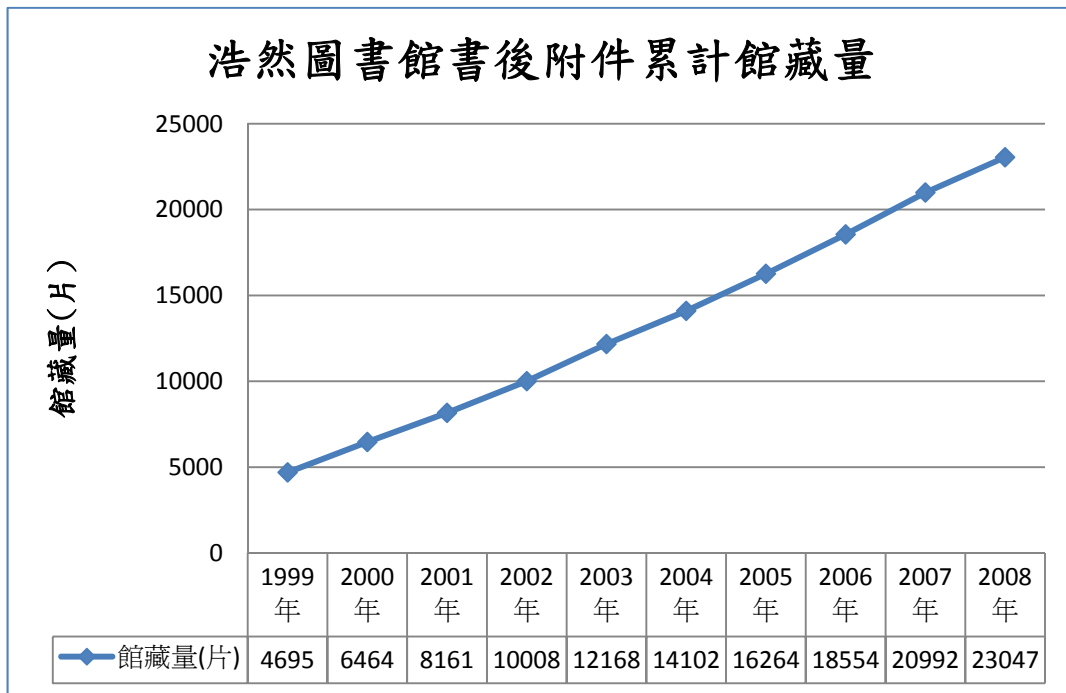


圖 1-1 浩然圖書館歷年書後附件累計館藏量

資料來源：交通大學浩然圖書館，本研究整理

另一方面，隨著書後附件數量增加，讀者於流通櫃檯借閱時，館員需要花費更長的時間找尋書後附件，往往造成讀者等待時間加長，如圖 1-2。

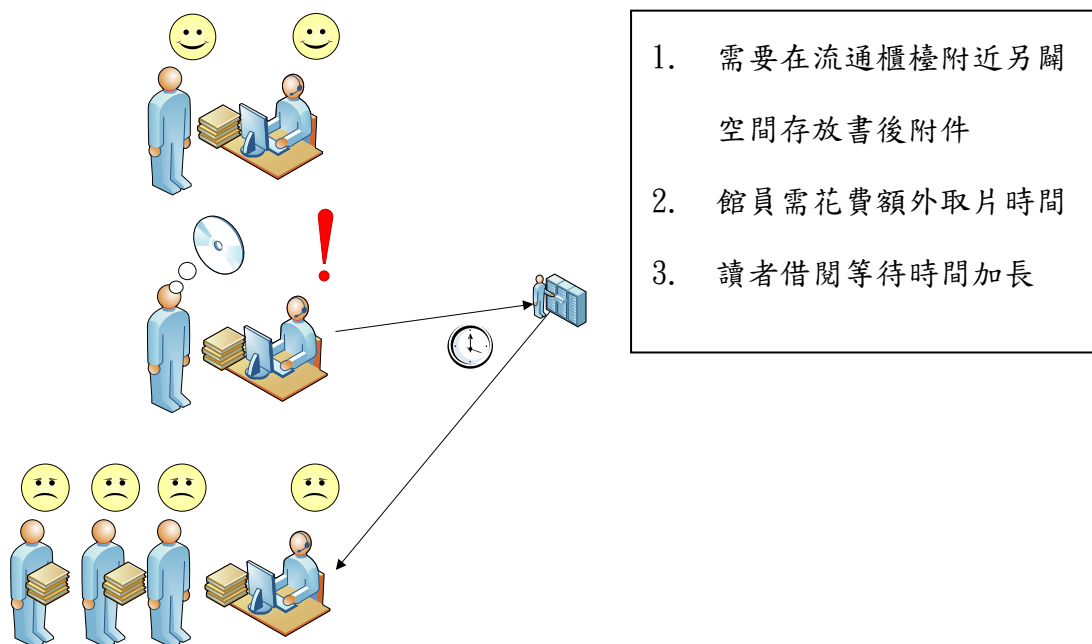


圖 1-2 書後附件借閱流程

除此之外，書後附件與紙本相比，在抗刮傷與耐磨擦部分相對地脆弱，而圖書館礙於人力與時間，不易在讀者歸還時詳細測試書後附件是否已遭損毀，讀者借閱書後附件後，除擔心自己使用時不慎刮傷之外，亦可能發生書後附件在使用前已經被前位借閱者刮傷，造成書後附件無法讀取與難以釐清賠償責任等問題，如圖 1-3。

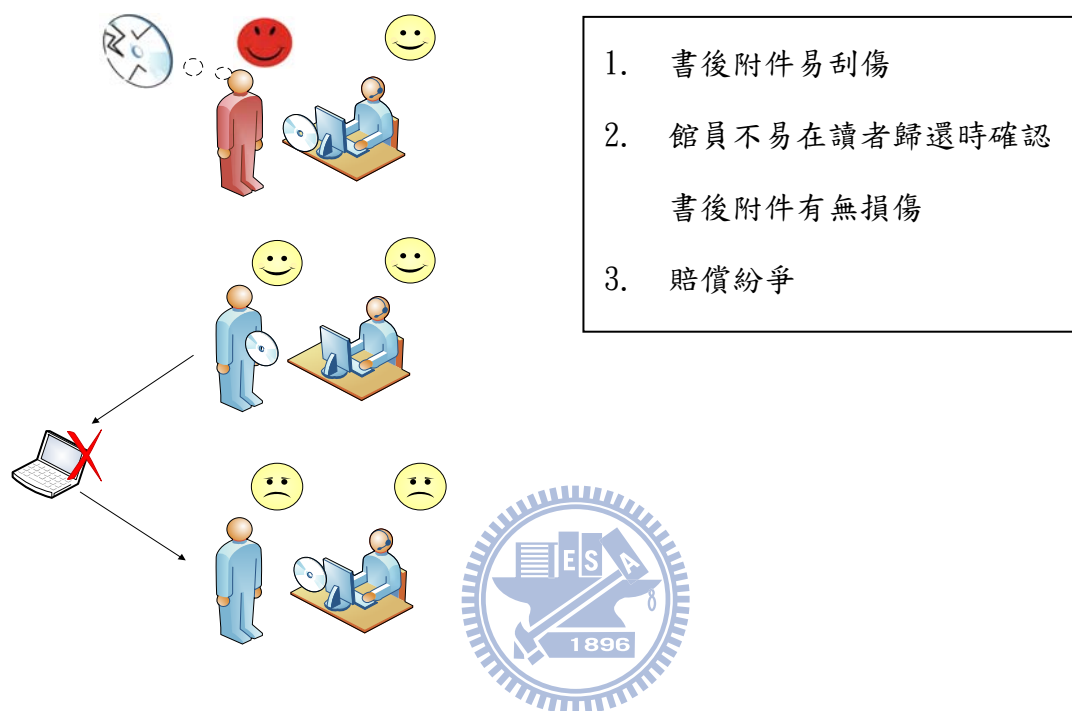


圖 1-3 書後附件歸還流程

書後附件為讀者帶來更佳的使用經驗與加值服務，也為圖書館帶來新的問題與挑戰，書後附件在保存、陳列與借還流通上，經過十幾年來，已逐漸出現上述問題，彙整如表 1-1 所示。

表 1-1 圖書館書後附件管理與流通問題列表

對象	問題點
圖書館	1. 需要在流通櫃檯附近另闢空間存放書後附件
	2. 館員需花費額外取片時間
	3. 館員不易在讀者歸還時確認書後附件有無損傷
	4. 賠償紛爭

對象	問題點
讀者	1. 書後附件易刮傷
	2. 讀者借閱等待時間加長
	3. 賠償紛爭

因此如何為書後附件產生的問題提出一套有效之解決方案，為本研究最主要的研究動機。近年來由於科技進步，網路的傳輸速度與普及率逐年提高，反之儲存媒體的價格卻每季降低，將書後附件儲存於伺服器並經由網路傳輸給讀者變得愈加可行。本研究擬建置一套資訊系統解決圖書館與讀者遭遇的問題，亦為本研究另一研究動機。

1.2 研究目的

1. 建置與研發資訊系統，讀者借閱書籍後到任何一台電腦使用帳號與密碼登入系統，即可取得目前借閱書籍之書後附件，由系統透過網路將書後附件遞送到讀者電腦。
2. 探討與分析建置系統相關技術，將系統與圖書館自動化系統連結，減少圖書館因導入系統付出的成本，並使圖書館之服務更加具有效率。
3. 模擬一本書籍對應一片或數片書後附件的特性，圖書館購買紙本書籍數量，即為允許書後附件被取得的數量。未來與書商討論書後附件著作權相關問題時，能解除書商疑慮。
4. 設計簡單且友善 (Friendly) 的圖形化使用者介面 (Graphic User Interface, GUI)，減少讀者學習操作使用者介面的時間與困難度，以減少推廣系統時的阻力。

圖 1-4 為系統希望達成目的之示意圖，本研究將此套系統稱為書後附件隨選系統 (Disc-on-Demand System)，簡稱 DOD 系統。其命名靈感來至視訊隨選系統 (Video-on-Demand System, VOD)，運作方式也類似。VOD 的運作方式為，使用者付費選擇想觀賞的節目，服務商將影片送到使用者的播放器中，並且影片有觀賞時間的限制。

DOD 系統的「附費動作」，為讀者至圖書館借閱紙本書籍，「時間限制」則是紙本書籍的借閱期限，且不需讀者自行蒐尋書後附件，而是由系統自動將借閱書籍相對應的書後附件送給讀者。

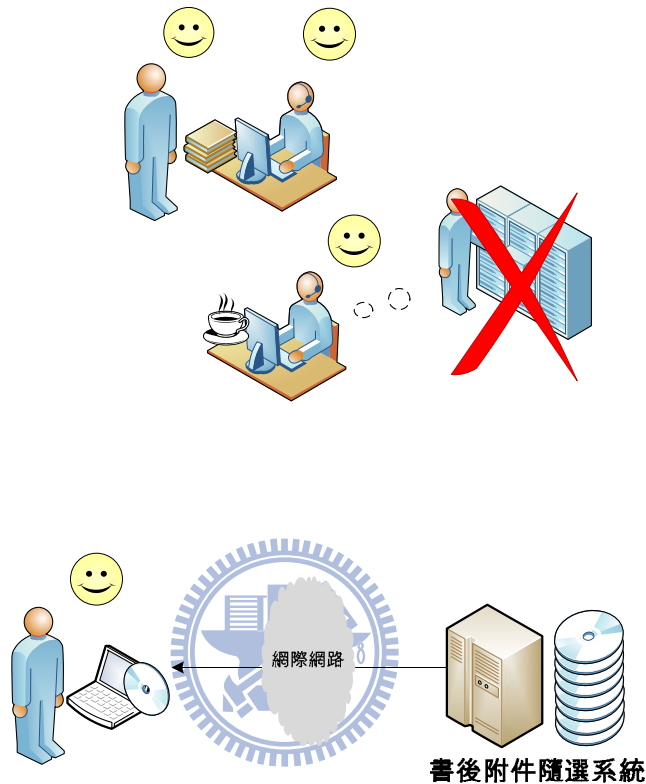


圖 1-4 書後附件隨選系統示意圖

1.3 研究範圍

1. 本研究假設圖書館已解決書後附件著作權相關問題，且讀者遵守國家法律不任意複製與傳播書後附件中檔案。本研究專注於提供具有讀者認證機制與連結圖書館自動化系統中借閱書籍與借閱期限之資訊系統，並非對書後附件中檔案內容進行數位版權保護。
2. 大容量、多數量與同時間的企業級儲存系統常常結合數種儲存技術，包括：磁碟陣列 (Redundant Array of Independent Disks, RAID)、磁帶櫃 (Tape Jukebox)、光碟櫃

(CD Data Storage Jukebox)、儲存區域網路 (Storage Area Network, SAN)、網路磁碟機 (Network Attached Storage, NAS)。本研究假設檔案儲存部分已有穩定可用的儲存裝置，並排除儲存裝置種類與效能面的研究課題。

3. 網路服務商面對大量使用者同時存取，可建置多個伺服器，並使用負載平衡 (Load Balancing) 的架構以滿足使用者需求。本研究也假設已有可正常使用的網路資源，並排除網路負載平衡與傳輸優化方面的研究課題。
4. 目前個人電腦作業系統種類繁多，本研究的讀者作業系統部分，專注於 Windows 系列作業系統 (Win98, Windows 2000, Windows XP, Vista, Windows 7)。根據 Net Applications¹公司報告指出，2008 年 12 月 Windows 作業系統市場佔有率為 88.63%，應可適用於大多數讀者的電腦環境。

1.4 研究流程

本研究肇始於筆者借閱書後附件時，館員協助筆者尋找書後附件，花費大量的時間，造成筆者後面讀者大排長龍。進一步詢問館員發現目前書後附件管理上的困難，同時筆者回想自己在借閱書後附件種種不便之處。經由文獻探討與分析，確定研究目的與範圍後，擬定系統功能需求，實際建置資訊系統，並對系統進行評估，最後撰寫研究報告提出結論與貢獻。研究流程如圖 1-5 所示。



¹ Net Application :知名網路調查公司，位於美國，網址：<http://www.netapplications.com>

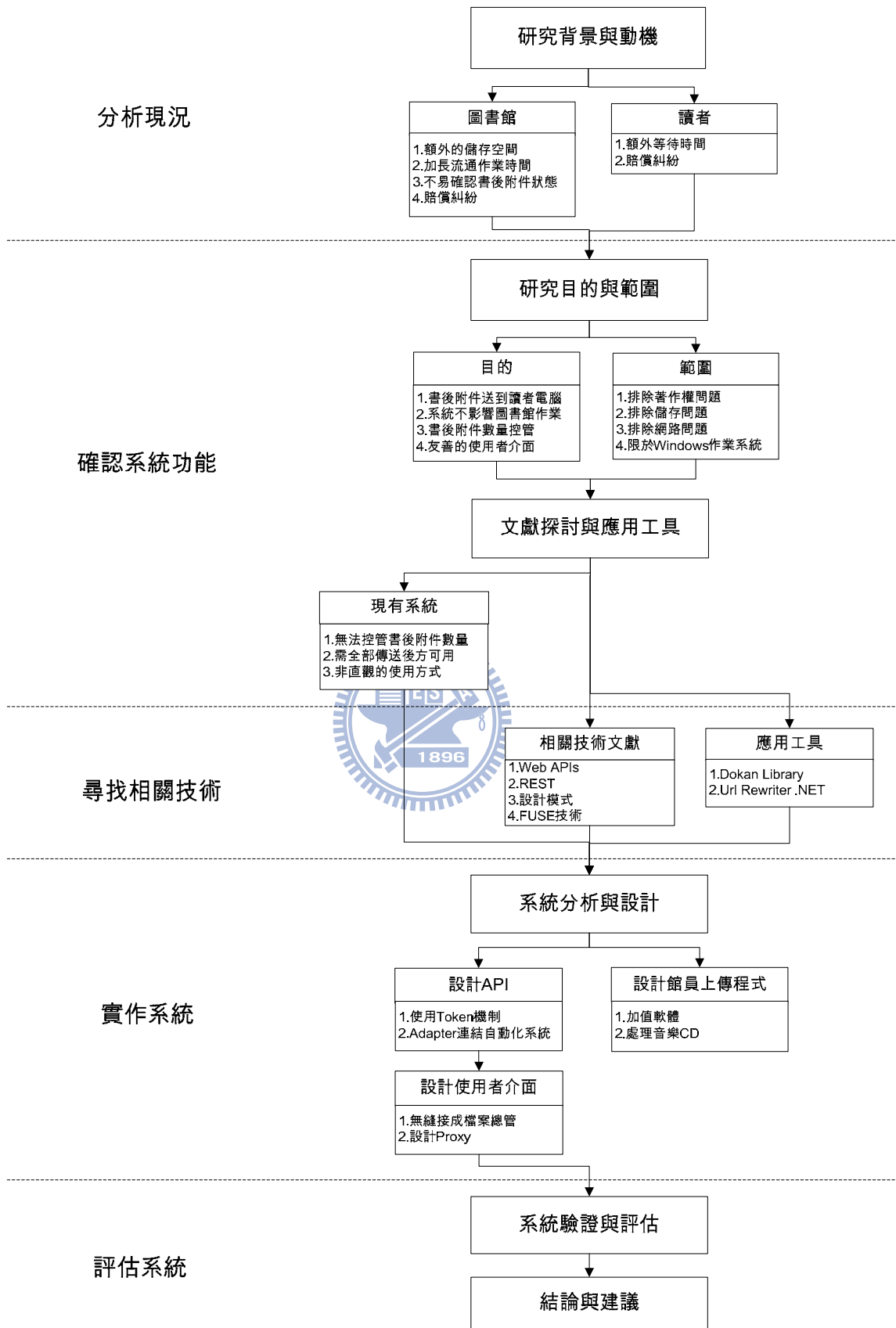


圖 1-5 研究流程圖

1.5 論文架構

本論文主要分為五個章節，整體架構如下：

第一章為緒論，透過研究動機與目的，說明目前書後附件在圖書館與讀者雙方所遭遇的問題，並訂定研究範圍與研究流程。

第二章為文獻探討與應用工具，以問題導向為出發點，介紹啟發本系統的研究與技術，並描述實際建置系統時所使用的工具。

第三章為系統分析與實作，詳細描述本系統架構、以及如何解決建置系統時所遭遇的問題。

第四章為系統評估，評估本系統整體性能與正確性。

第五章為結論與未來研究方向，為本研究之總結，說明本研究之貢獻，並提出未來可繼續研究之議題。



二、文獻探討與應用工具

本章內容分為兩個部份，2.1 節以問題導向方式說明啟發本研究之相關文獻；2.2 節則為建置本系統所需工具之介紹。

2.1 文獻探討

圖 2-1 為本研究文獻回顧整體分佈圖，共分為 REST 風格 API、設計模式與 FUSE 三大部份。

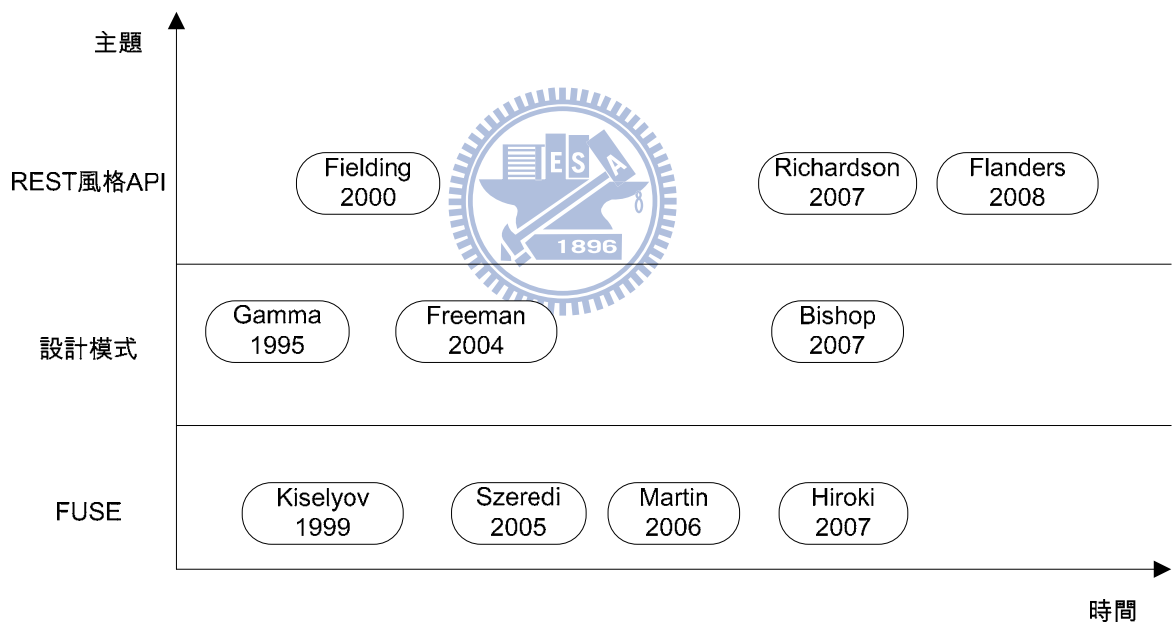


圖 2-1 文獻整理

2.1.1 圖書館書後附件之相關研究

圖書館與光碟 (CD-ROM) 的邂逅始於光碟資料庫 (CD-ROM Database)，早期由於線上資料庫成本遠高於光碟資料庫，西元 1987 年後歐美各國與我國各校皆陸續採購光

碟資料庫作為教授、研究生及學生重要的研究工具[1]，當時的研究主要聚焦於光碟資料庫的建置[2]與傳輸[3]方式。時至今日線上資料庫已成主流，圖書館與光碟卻再度相逢，只不過昔日的光碟資料庫換成今日本研究的主角—書後附件。

搜尋各大學術論文期刊資料庫²並未發現國內外與書後附件相關的研究，最後只在大陸地區期刊中找到相關文章。大陸地區因其著作權法允許隨書光盤³在校園內傳播[4]，有相當數量關於隨書光盤議題的研究。馬飛歸納目前大陸地區圖書館隨書光盤管理方式為書盤一體化、書盤分離式與網路管理式三種模式，並提出管理方法[5]，白永革與李新宇描述如何在編目時將紙本書籍與隨書光盤結合使其互相參照[6-7]，余元提出可用 ISBN 碼於隨書光盤館際合作的橋梁[8]，沈靜萍認為可直接由 MARC 導出隨書光盤訊息[9]，杜至波認為隨書光盤系統接口應該標準化，即支援 OAI 與 OpenURL[10]，徐勇進描述製作光碟映像檔⁴的一些技術細節[11]，沈靜萍、周月蓮、李禾與蔡立分別介紹他們以 ASP 或 ASP.NET 設計的隨書光盤管理系統[9, 12-14]，譚榕介紹使用開放原始碼軟體建置隨書光盤管理系統[15]。



大陸地區的相關研究對圖書館如何將隨書光盤納入編目中有頗多可借鏡之處，但在系統部分並未對本研究有任何的啟發，圖 2-2 為中國石油大學隨書光盤管理系統。系統有三個特點：

1. 將隨書光盤的光碟映像檔全部放置於網路上，限定使用校園 IP 的電腦下載。
2. 提供檢索功能讓讀者依關鍵字搜尋隨書光盤
3. 要求讀者下載整個光碟映像檔到電腦後，使用某種虛擬光碟程式開啟隨書光盤。

² 包括 IEEE Xplore、ACM Digital Library、ScienceDirect、CiteSeerX、本國碩博士論文與中文期刊索引系統。

³ 大陸地區稱書後附件為隨書光盤。

⁴ 將整片光碟存成一個檔案，此檔案稱為光碟映像檔。



圖 2-2 中國石油大學隨書光盤管理系統

資料來源：<http://202.194.153.246:8080/>

將隨書光盤光碟映像檔全部放置於網路上，造成的結果為讀者可以下載所有的隨書光盤，而非僅目前借閱書籍的隨書光盤；提供檢索功能，乃因所有的隨書光盤映像檔都被放到網路上，需要查詢介面幫助讀者找到需要的隨書光盤；要求讀者下載整個光碟映像檔到電腦最直接的後果就是讀者將光碟映像檔散佈給第三者，而且即使只需要隨書光盤中的某個檔案，也必須下載整個光碟映像檔。使用某種虛擬光碟程式開啟隨書光盤映像檔，則需要讀者另外安裝並學習某種虛擬光碟程式的使用方法。

整體而言，大陸地區的隨書光盤管理系統，類似早期圖書館的光碟資料庫，只是資料庫提供商改成圖書館自己。其使用方式為：

1. 搜尋隨書光盤映像檔
2. 下載隨書光盤映像檔
3. 安裝虛擬光碟程式
4. 使用虛擬光碟程式開啟隨書光盤映像檔

本研究認為其使用過程非常「曲折」並且對讀者很不友善。本研究的研究目的為將書後附件「送到」讀者電腦而非讀者自行上網尋找，只讓讀者「取用書後附件的檔案」而非下載整個書後附件映像檔，範圍僅為「目前借閱書籍」的書後附件而非全部書後附件，與目前大陸地區存在的系統均有顯著不同。

2.1.2 應用程式介面

應用程式介面 (Application Programming Interface, API)，原指應用軟體可以用於取得作業系統或其他服務的抽象化 (Abstraction) 介面。但本研究提及的 API 泛指網路上讓開發人員可以簡單地呼叫，做快速二次開發的服務，也稱為 Web APIs，一般習慣上會以 API 提供商的名稱為前綴來稱呼這一類的服務。例如：Google API⁵、Facebook API⁶、eBay API⁷。

圖 2-3 為 eBay API 的功能圖，清楚地表現了 API 的功能與定位。原本使用者登入 eBay 網站後只能使用 eBay 預先設計的網頁介面，進行列表與查詢等等動作。若第三方程式 (3rd Party Application) 欲提供更多的功能，則可透過網際網路 (Internet) 由 eBay

⁵ Google 提供的 API 服務。網址 <http://code.google.com/intl/zh-TW/>

⁶ Facebook 提供的 API 服務。網址 <http://developers.facebook.com/>

⁷ eBay 提供的 API 服務。網址 <http://developer.ebay.com/common/api/>

API 取得以可延伸標記語言 (Extensible Markup Language, XML)⁸表示的內容，第三方程式可解析 XML 取得 eBay API 提供的資料，並以自己的創意設計出不同的應用與服務。

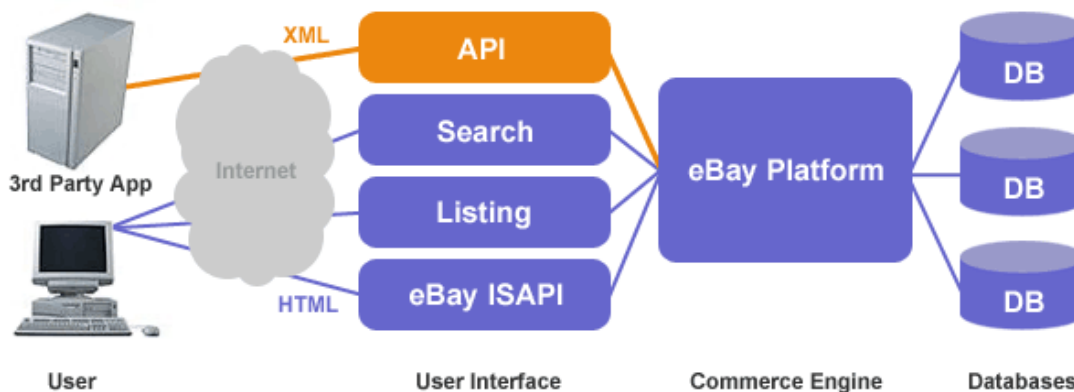


圖 2-3 eBay API 功能示意圖

資料來源：<http://developer.ebay.com/common/api/>

近兩年來網路上掀起一股聚合 (Mashup) 熱潮，所謂聚合就是從多個 API 獲取資訊，組合成新網絡應用的一種方法[16-17]。地圖日記 (atlaspost) 為應用 API 的一個成功案例，它使用 Google 及 Yahoo! 地圖 API 建立地圖社群。並且地圖日記本身所累積的內容，亦可透過地圖日記 API 再供其他網站使用。地圖日記於西元 2007 年年中上線，初期獲得經濟部「Web 2.0 創新點子大募集活動」入選首度曝光。西元 2008 年於 Demo Conference 獲得 People's Choice Award。更有名的成功案例是使用 Facebook API 的開心農場，根據創市際公司⁹統計，西元 2009 年 8 月台灣地區曾造訪 Facebook 的網友中，高達 62% 網友曾使用過 Facebook 應用程式，最多人使用的應用程式就是開心農場。開心農場利用 Facebook API 提供的好友列表，設計好友之間互相偷菜，贈禮等功能，充滿互動與趣味性，在台灣造成一股熱潮。

⁸ 一種標示語言，利用標籤來定義各種屬性，達到交換資料的目的。

⁹ 市場研究顧問公司，位於台灣。網址 <http://www.insightexplorer.com>

再回頭重新思考書後附件，本研究認為書後附件的重點在於其「內容」，即光碟中經過書籍作者適當安排的目錄與檔案，讀者依照書籍中提示的目錄與檔案名稱，定位到作者希望讀者參考的檔案，而光碟只是書後附件儲存檔案的媒介。但是目前隨書光盤管理系統最大的軟肋卻是代表光碟的光碟映像檔容易被複製與散播。本研究認為特別製造一個本來就不需要存在的東西(光碟映像檔)，提供給讀者下載，然後要求這個東西需要好好被管制，不能隨意散播，在系統設計上一開始就陷入誤區。是否有技術能跳過下載光碟映像檔，直接取得書後附件的內容，並能做到更精準的權限控管？這個問題的答案就是—API。

本研究在設計整個系統前，先完成一組基於超本文傳輸協定 (Hyper Text Transfer Protocol, HTTP)¹⁰ 與 XML 的 API，稱為 DOD API，DOD API 為聚合圖書館自動化系統提供的帳號密碼驗證、讀者目前借閱書籍列表與本研究提供的書後附件檔案傳輸機制而成。觀察目前網路上成功案例與 API 本身具備的特性，DOD API 有八個優點：

1. 不須讓讀者下載整個書後附件光碟映像檔，而是藉由 API 提供讀者所需檔案，避免被光碟映像檔被隨意複製與散播的缺點。
2. 按需求下載各別檔案，而非整個光碟映像檔，可節省傳輸時間，尤其對以 DVD¹¹ 製作，容量為 GB 級的書後附件助益最大。
3. 能精準地控制 DOD API 的使用者與書後附件同時被取得的數量，達成圖書館購買幾本書籍，就只允許幾份書後附件被取得的研究目的。
4. 未來實作使用者介面時，不管是使用如圖 2-4 的 Web-based 系統或是如圖 2-5 的使用者-伺服器(Client-Server)系統，都可以如圖 2-6 使用 DOD API 達成。

¹⁰ 網際網路上應用最為廣泛的一種網路協議。用於瀏覽器與網頁伺服器的溝通。

¹¹ 單面單層 DVD 容量約 4.7GB。

5. DOD API 提供跨平台的可能性，目前或未來推出面對一般使用者的作業系統毫無疑問地都必需支援網路功能，即可利用 DOD API 設計出基於各種作業系統的使用者介面。
6. DOD API 可動態調整提供給讀者的服務，如果未來圖書館考慮將非書資料，例如：磁片與錄影音資料等等也提供給讀者，可在 DOD API 另外增加服務；同樣地，如果只想停止某部分服務，只要從源頭，即 DOD API 關閉相對應功能即可。
7. DOD API 更加容易與圖書館其它系統整合。
8. DOD API 的開放精神，更適合大學校園。

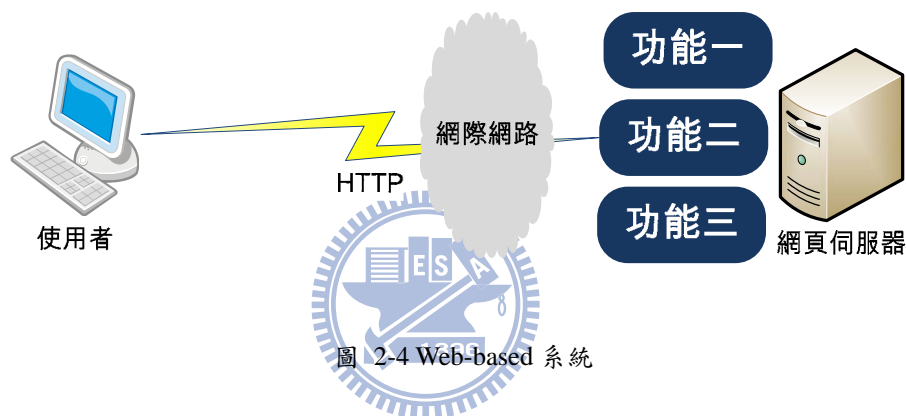


圖 2-5 使用者-伺服器系統

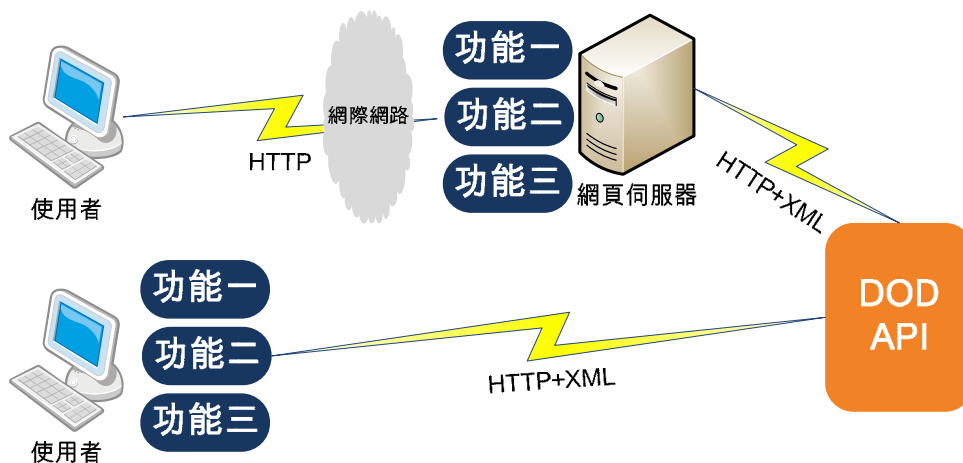


圖 2-6 以 API 實作 Web-Based 與使用者-伺服器兩種模式的系統

2.1.3 表述化狀態轉變

表述化狀態轉變 (Representational State Transfer, REST) 是 Fielding¹² 研究 Web 的需求與面臨的問題之後，在他的博士論文中提出適用於分散式超媒體系統 (Distributed Hypermedia Systems) 的一種軟體系統架構風格 (Architectural Style)。Fielding 由空風格 (Null Style) 開始推導到按需代碼 (Code-on-Demand) 提出了數項架構約束 (Architectural Constraints)，並證明了這些約束能達成組件交互運作的可伸縮性 (Scalability of Component Interactions)、介面的通用性 (Generality of Interfaces)、組件的獨立部署性 (Independent Deployment of Components)[18]。

Fielding 對於本研究的啟發為提供 DOD API 一種架構風格。設計 DOD API 的細節部分，則參考 RESTful Web Services[19] 一書。實際編碼 (Coding) 因本系統使用 ASP.NET¹³ 語言，故參考書籍 RESTful .NET : Build and Consume RESTful Web Services

¹² HTTP 和 URI 等 Web 架構標準的主要設計者，Apache HTTP 伺服器的主要開發者。

¹³ 一種程式語言，主要支援者為微軟。

with .NET 3.5 [20]。本研究以 Tilkov¹⁴提出 REST 風格 API 進行設計時應該遵守的五項原則[21]為主，整理敘述 REST 風格 API 實作時的要點如下：

1. 為所有資源 (Resource) 定義一個唯一的通用資源標誌符 (Uniform Resource Identifier, URI)。
2. 通過 URI 得到一個資源的表述 (Representation)，表述通常以 XML 表示，但也可以是其它 MIME (Multipurpose Internet Mail Extensions) 類型的資源。
3. 使用 HTTP 協議的 GET 方法取得一個資源，POST 方法創建一個新資源，PUT 方法更新資源，DELETE 方法刪除一個資源。
4. 使用 HTTP 協議回傳的狀態碼 (Status Code)，代表要求一個資源的結果，例如：200 代表成功、400 代表參數錯誤、401 代表未認證、403 代表資源不存在等等。

設計 API 時 REST 常被與 Web Services¹⁵比較，Pautasso 與 EFOAGUI 的研究皆指出 REST 與 Web Service 風格的 API 皆可滿足實際作業上的需求。其中 REST 實作簡單且對於 Web 環境變化的適應力較佳，而 Web Service 適合用於需要高可靠度 (Reliability) 與交易異動 (Transactions) 的商業系統[22-23]。

DOD API 無交易異動，而且 Web Service 使用的簡單對象訪問協議 (Simple Object Access Protocol, SOAP) 協議，無法滿足 DOD API 需要傳送書後附件中二進位檔 (Binary File) 的要求，需使用其他方式實作。因此以 REST 實作 DOD API 成為一項自然而然的選擇。

¹⁴ InfoQ SOA 社區的首席編輯，並且是位於德國和瑞士的 innoQ 公司的共同創始人、首席顧問。

¹⁵ Web Services 是一種軟體元件，它透過 Web 通訊協定及資料格式的開放式標準(例如 HTTP、XML 及 SOAP 等)來為其他的應用程式提供服務。

2.1.4 設計模式

Gamma 等四人收集二十三個在物件導向程式設計 (Object Oriented Programming) 中常見的問題與解法，名之為設計模式 (Design Patterns)，並記錄描述後集結成書[24]。書中對設計模式的說明為：

設計模式使人們可以更加簡單方便地複用成功的設計和體系結構。將已證實的技術表述成設計模式也會使新系統開發者更加容易理解其設計思路。設計模式說明你做出有利於系統複用的選擇，避免設計損害了系統複用性。通過提供一個類和物件作用關係以及它們之間潛在聯繫的說明規範，設計模式甚至能夠提高已有系統的文檔管理和系統維護的有效性。簡而言之，設計模式可以讓設計者更快更好地完成系統設計。

值得一提的是，設計模式並不是一種發明。在設計模式一詞未被用於軟體設計領域時，程式設計師使用物件導向程式設計的方法，仍然能設計出功能良好且穩定運行的程式。但是設計模式被提出後，其優點不斷被證實，Lange 研究認為設計模式可以幫助應用程式框架 (Framework)¹⁶ 的使用者，更容易使用框架設計出更好的系統[25]。Broecke 運用 Observer 模式建立了多媒體網路系統框架[26]，Schmidt 敘述設計模式如何被運用於大型的商業系統與其帶來的優點[27]。

基於上述研究提出的優點，本研究參考 Head First Design Patterns[28]與 C# 3.0 design patterns[29]兩本書籍中的講解與示例，發現其中三種模式可適用於本系統。

¹⁶一組精心設計的函式庫集合，幫助程式設計者更容易設計程式。例如：Java Development Kit(JDK) , .Net Framework。

1. 適配器模式 (Adapter)：Gamma 將其定義為「將某個類 (Class) 的介面轉換成另一個介面。適配器模式可以消除由於介面不匹配所造成的兼容性問題。」
2. 抽象工廠模式 (Abstract Factory)：Gamma 將其定義為「為一系列類提供統一的創建介面。當需要這個系列類的某一個對象 (Entity)時，可以從抽象工廠中選出相應的類創建一個對象。」
3. 代理模式 (Proxy)：Gamma 將其定義為「替其他對象提供一個代理以控制對這個對象的訪問。」

三種模式於如何與為何應用於 DOD 系統，說明於下：

1. Adapter

Adapter 其實大量存在現實世界中，圖 2-7 代表將 USB 介面轉成 PS2，Print Port 與 mini-USB 介面的三種產品。Adapter 的用途是將各種不同的介面，轉成另一種共同的介面以利後續使用。通常使用 Adapter 是對現況的一種妥協，以 USB-PS2 Adapter 為例，電腦只有 USB 且電腦運作良好；PS2 鍵盤也已存在且未損壞。此時使用 Adapter 將兩者串接，就可以達到繼續使用兩者的目的。



圖 2-7 現實世界中的 Adapter

本研究在連結圖書館自動化系統的時候也遭遇相同的問題，如圖 2-8，整個 DOD 系統能正常運作，需使用者介面、DOD API 與圖書館自動化系統緊密配合。但現況是圖

書館的自動化系統已經存在並運作良好，不可能特別為本系統修改。所以在 DOD API 與圖書館自動化系統溝通的部分，本研究使用 Adapter 模式將圖書館自動化系統的功能轉換成 DOD API 需要的功能。

以更宏觀的觀點來看 DOD 系統，整個 DOD API 部分都是 Adapter，DOD API 提供了一種 Adapter 讓讀者能由 DOD API 查詢自動化系統中自己的借閱紀錄，並隱藏了圖書館自動化系統大部分不該由讀者取得的資訊。



圖 2-8 使用者介面—DOD API—圖書館自動化系統連接示意圖

2. Abstract Factory



Adapter 模式用於連結 DOD API 與圖書館自動化系統，而如何讓系統如圖 2-9，轉換到不同圖書館或是圖書館更換自動化系統時不需要修改程式碼。就是利用 Abstract Factory 模式達成。DOD API 並不將所有不同圖書館的 Adapter 全部都寫在程式碼裡面(事實上也不可能)而是針對不同圖書館提供不同的動態連結函式庫 (Dynamic Link Library)，並在執行時動態載入。

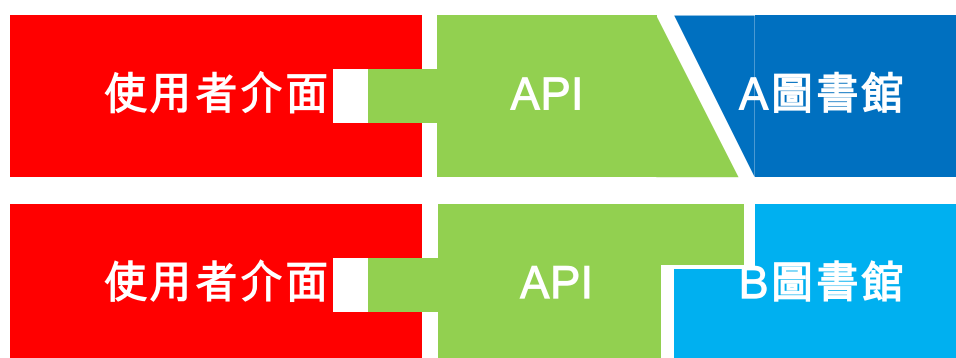


圖 2-9 使用者介面—DOD API—多個圖書館自動化系統連接示意圖

本系統若要應用於連結不同的自動化系統，只需將新的動態連結函式庫複製到系統中，並修改 ASP.NET 設定檔 web.config 中的 <appSetting> 區段即可。

```
<configSections>

  <appSettings>

    <add key="Factory" value="XXX"/>  //(XXX 為圖書館 Adapter 的名稱)

  </appSettings>

</configSections>
```

3. Proxy

圖 2-10 為網路上的 Proxy 機制，第一位使用者要求資料後，Proxy 伺服器會將資料緩存 (Cache)，之後第二、第三位使用者要求相同資料時，則不需再到原網頁伺服器要求資料，而是由 Proxy 伺服器將緩存中的資料傳回給第二、第三位使用者。

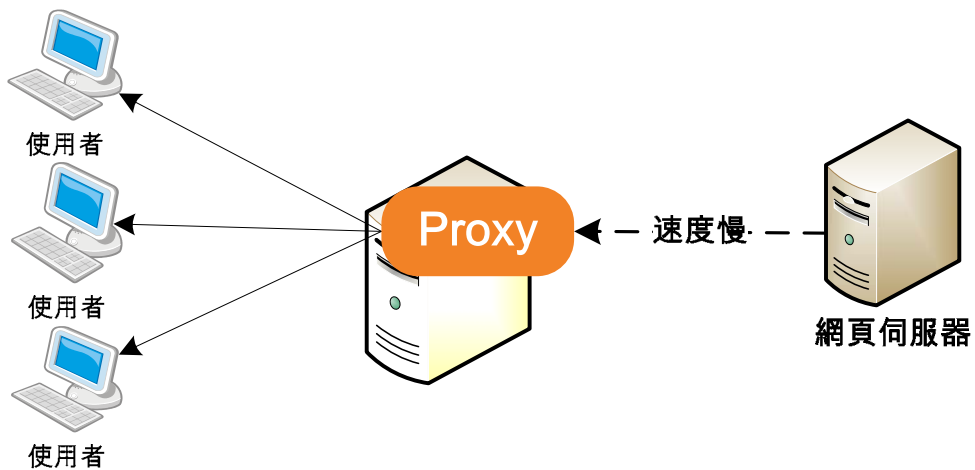


圖 2-10 網際網路上 Proxy

DOD 系統的 Proxy 機制稍有不同，如圖 2-11。考慮到讀者可能會向書後附件檔案伺服器要求多次相同的當檔案，所以 DOD 系統設計一個執行在讀者電腦上的 Proxy，

當讀者第一次向書後附件檔案伺服器要求資料後，Proxy 會將檔案儲存於使用者硬碟，並於第二，第三次要求相同檔案時，Proxy 將直接從硬碟取出檔案並提供給使用者。

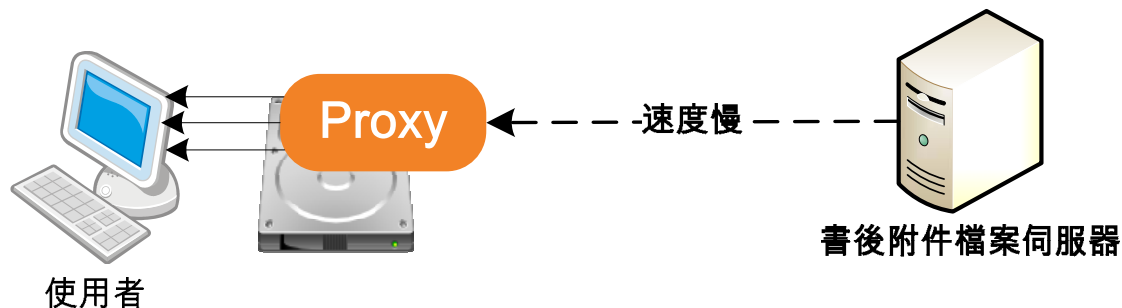


圖 2-11 DOD 系統的 Proxy

2.1.5 用戶空間文件系統

用戶空間文件系統 (Filesystem on User Space, FUSE) 是 Szeredi 在 SourceForge¹⁷上成立的一個專案 (Project)，主要目的為替 Linux 作業系統提供一個非基於核心層 (Kernel-Level) 的檔案系統 (File System)¹⁸，其示意圖如圖 2-12 所示，FUSE 主要包含兩部份，

1. 一個 Kernel-Level 的檔案系統

Kernel-Level 的檔案系統，負責將使用者對檔案系統的操作，例如：打開目錄、讀取與寫入等等動作傳遞給使用者層 (User-Level) 的監聽器。

¹⁷ 開放原始碼的作者進行開發管理的集中式場所，也是全球最大開放原始碼開發平台和倉庫

¹⁸ 檔案系統是一套實作了資料的儲存、分級組織、存取和獲取等操作的抽象資料型式。

2. 一組函式庫 (Library)

接受到 Kernel-Level 傳遞給 User-Level 監聽器的訊息之後，可以藉由 FUSE 提供的函式庫，得知使用者對檔案系統的操作並回應。

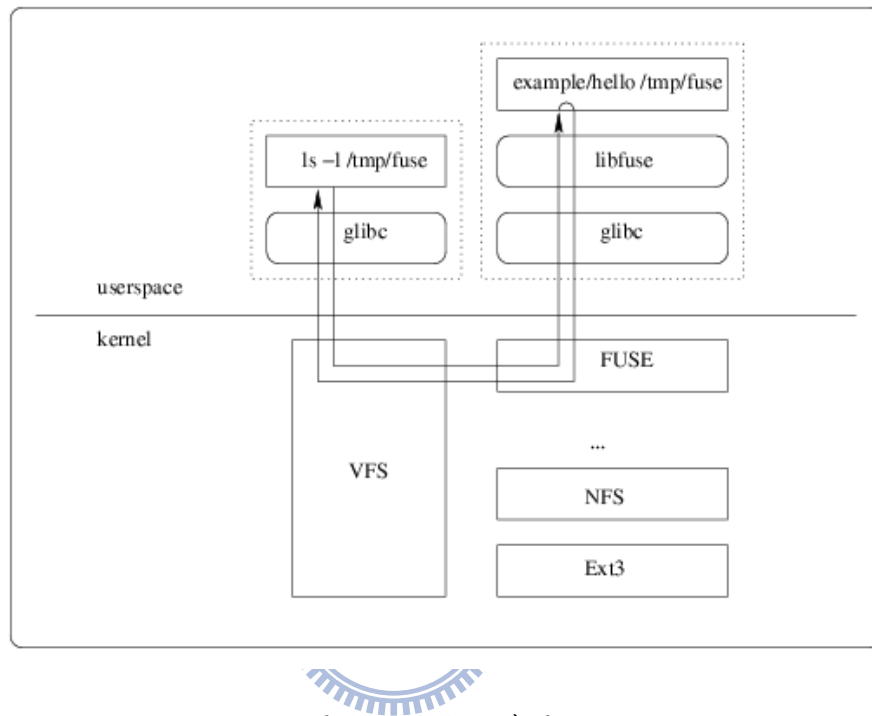


圖 2-12 FUSE 示意圖

資料來源：<http://fuse.sourceforge.net>

FUSE 揭櫫一種概念，檔案系統中的內容並非就是儲存在磁碟中的真實目錄與檔案，只要能「騙」過作業系統，在作業系統試圖開啟一個目錄時告訴作業系統目錄下有哪些檔案；在作業系統試圖讀取一個檔案時告訴作業系統檔案的內容為何，完全可以實作一個設計者自己定義的檔案系統。FUSE 的研究與應用主要可以分為兩種，第一種是使用不同的通訊協議，例如：SSH[30]、FTP 與 HTTP 等，另一個部分則是將不同的資料來源表示成檔案系統，例如：關聯式資料庫[31]與 Gmail[32]等。

啟發 DOD 系統的研究為 Kiselyov 設計的一個基於 HTTP 協議的網路檔案系統，Kiselyov 將作業系統對檔案存取動作對應到 HTTP 協議的 GET、HEAD、PUT、DELETE 方法，並假設作業系統就是瀏覽器，將作業系統對檔案的操作經由 HTTP 協議映射

(Mapping) 到遠端伺服器上，完成了一個檔案系統[33]；Martin 認為 XML 可以表現檔案系統的目錄結構並使用 FUSE 技術在 Linux 上展示[34]。

本研究由 DOD API 提供以 XML 表示的書後附件檔案結構，然後利用 FUSE 的概念將其映射到讀者電腦成為一個虛擬磁碟，並使用 HTTP 協議傳輸書後附件檔案，讓讀者可以如同將書後附件光碟放入光碟機般地使用書後附件。

2.1.6 全自動區分計算機和人類的圖靈測試

全自動區分計算機和人類的圖靈測試 (Completely Automated Public Turing Test to Tell Computers and Humans Apart, CAPTCHA)，是一種區分用戶是電腦還是人類的機制[35]。在一個 CAPTCHA 挑戰 (CAPTCHA Challenge) 中，服務器會自動生成一個「問題」由客戶端來解答。這個問題必須只有人類才能解答，由於電腦無法解答 CAPTCHA 的問題，所以能回答出問題的客戶端就被認為是人類。CAPTCHA 主要用之於防止網路上的機器人程式，使用暴力破解法 (Brute Force Attack)，不停地送出登錄驗證以竊取使用者的密碼。

很多人可能在還不知道什麼是 CAPTCHA 挑戰之前，就已經遭遇過 CAPTCHA 挑戰了。當註冊成為一個網站的會員，或是發表文章。最後網頁會出現一個扭曲且帶有雜訊的圖片，要求使用者輸入圖片所代表的字母與阿拉伯數字，這就是 CAPTCHA 挑戰所謂的「問題」，CAPTCHA 的普及率也由此可見一般。Yan 將 CAPTCHA 分為文字 (Text-Base)、聲音 (Sound-Base) 與圖形 (Image-Base) 三種模式[36]。其中文字模式相對於其他兩類，有容易實作與使用率較高的優點，因此本研究也使用文字模式的 CAPTCHA 挑戰以防止有心人利用 DOD API 竊取讀者帳號密碼。

2.2 應用工具

2.1.1 Dokan Library

Dokan Library 為 Windows 作業系統提供類似 FUSE 的功能[37]。作者在官方網站對 Dokan Library 的介紹如下：

當你要在 Windows 作業系統上建立一個新的檔案系統，例如 FAT 或是 NTFS，你需要設計一個檔案系統的設備驅動程式。為 Windows 作業系統設計一個 Kernel Mode 的檔案系統是極端困難的一件工作，但是使用 Doakn Library 你可以非常容易地建立自己的檔案系統，而不需撰寫設備驅動程式，Doakn Library 類似 FUSE 不過是運作在 Windows 作業系統上。

Dokan Library 也包含兩個部分：



1. 一個使用者端 (User-Level) 的動態連結檔—dokan.dll。
2. 一個系統端 (Kernel-Level) 的設備驅動程式—dokan.sys。

安裝 Dokan Library 之後，可以在作業系統上創造一個虛擬磁碟 (Virtual Disk)，對這個磁碟上檔案系統的動作，會送到 dokan.sys，然後 dokan.sys 再轉送給 dokan.dll。例如：在磁碟機圖示點兩下，dokan.sys 就會通知 doakn.dll， dokan.dll 被通知後會引發 OpenDirectory 事件，設計者處理這個事件告訴 dokan.dll 磁碟機根目錄下應該有哪些檔案與這些檔案的名稱、屬性、大小、建立時間等等，dokan.dll 回傳給 dokan.sys，然後 dokan.sys 再傳遞給作業系統，作業系統就將使用者提供的檔案顯示在檔案總管裡，Dokan Library 運作的示意圖如圖 2-13。

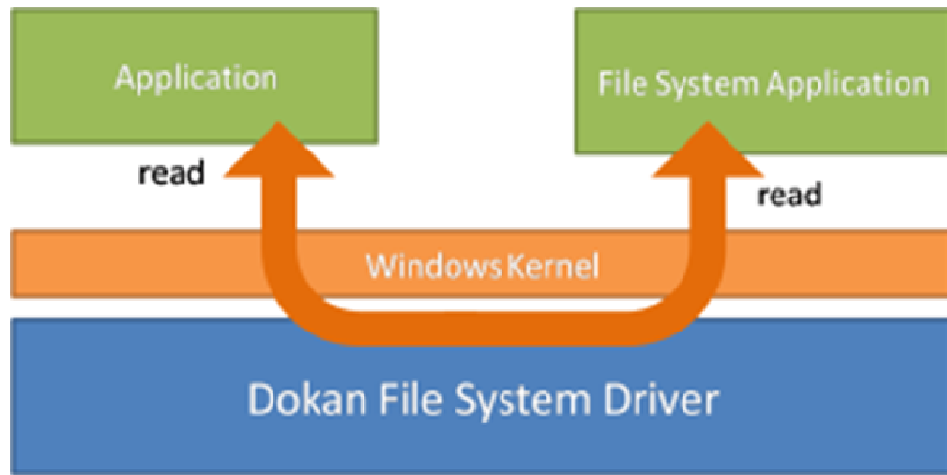


圖 2-13 Dokan Library 示意圖

資料來源：<http://dokan-dev.net/en/>

本研究使用 Dokan Library 的方式與 Dokan Library 原本被設計的原意不符，Dokan Library 目的為讓設計者建立自己設計的檔案系統；而本研究觀察目前書後附件的使用方式，發現書後附件的使用者介面即為檔案總管。因此想設計一個「很像」檔案總管，免去讀者重新學習操作方式的使用者介面。無論如何，最後本研究利用 Dokan Library 設計的使用者介面不但像檔案總管，它「就是」檔案總管。

使用檔案總管為使用者介面的另一個好處是，檔案總管為作業系統必備的一個應用程式，不管未來 Windows 出了幾個版本，只要讀者選擇安裝新版本 Windows 作業系統，就需學習新的檔案總管操作方式，自然而然也學會 DOD 系統的使用者介面操作方式。

2.1.2 UrlRewriter.NET

UrlRewriter.NET[38]主要被用於搜尋引擎優化(Search Engine Optimization, SEO)，它的功能是利用正則表達式 (Regular Expression)¹⁹，將某個網址轉變成另外一種型態。

¹⁹ 一個用來描述或者匹配一系列符合某個句法規則的字符串。

一個網址通常代表一個靜態的資源，打開瀏覽器，在網址列輸入 `http://ServerIP/index.htm` 並按下 Enter 鍵，代表要求 `index.htm` 這個檔案，可以僅輸入 `http://ServerIP` 此時則是由網頁伺服器提供預設的檔案給瀏覽器，通常預設的檔案為 `index.htm` 或是 `index.html`，瀏覽器其實還是參照至某個實際存在的檔案。

此種方式顯然不符合 DOD API 需要的 URI 表現方式，DOD API 的 URI 並非真正存在而是動態產生。於是本研究試著以網址加上參數的方式，達到動態網頁的目的。例如：`http://ServerIP/Default.aspx?discid=X000001`。但又遇到另一個難題，這種格式的 URI 並不符合 REST 風格。最後本研究使用了 `UrlRewriter.NET` 達成產生 DOD API 中各符合 REST 風格 URI 目的，如

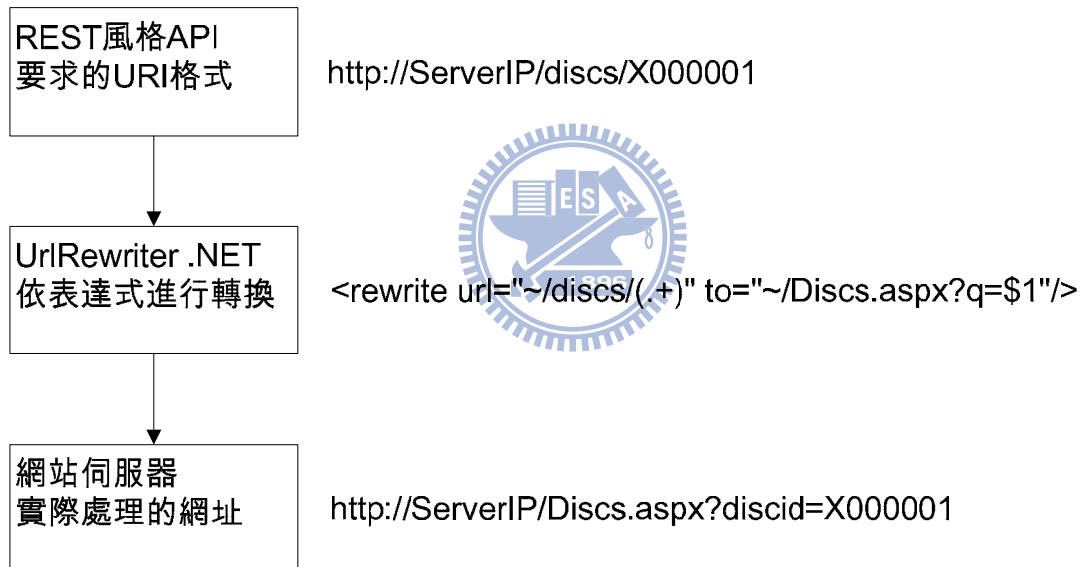


圖 2-14。

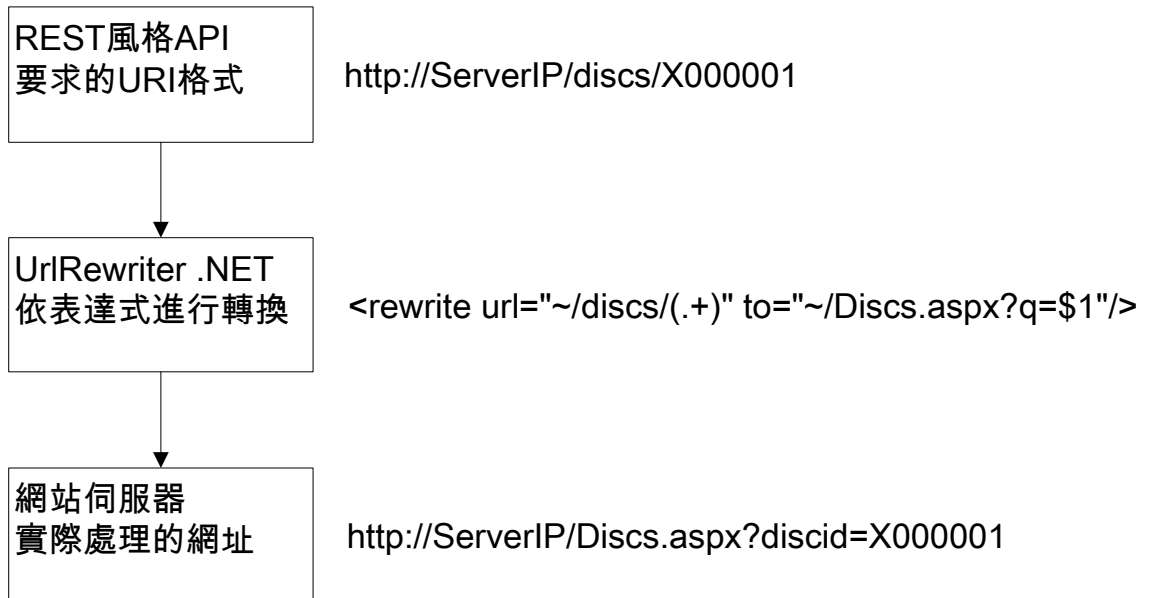


圖 2-14 UrlRewriter.NET 運作示意圖



三、系統分析與實作

本章以讀者與圖書館使用本系統必須具備的條件開始，一步步描述整個系統架構並說明如何建置 DOD 系統與解決所遭遇的難題。

3.1 系統架構

如圖 3-1，使用 DOD 系統讀者必需自行準備已安裝 Windows 系列作業系統的個人電腦，並確保可連上網際網路；圖書館須準備一台可以與自動化系統連結的伺服器，並指派一或數位館員進行將書後附件上傳的工作。

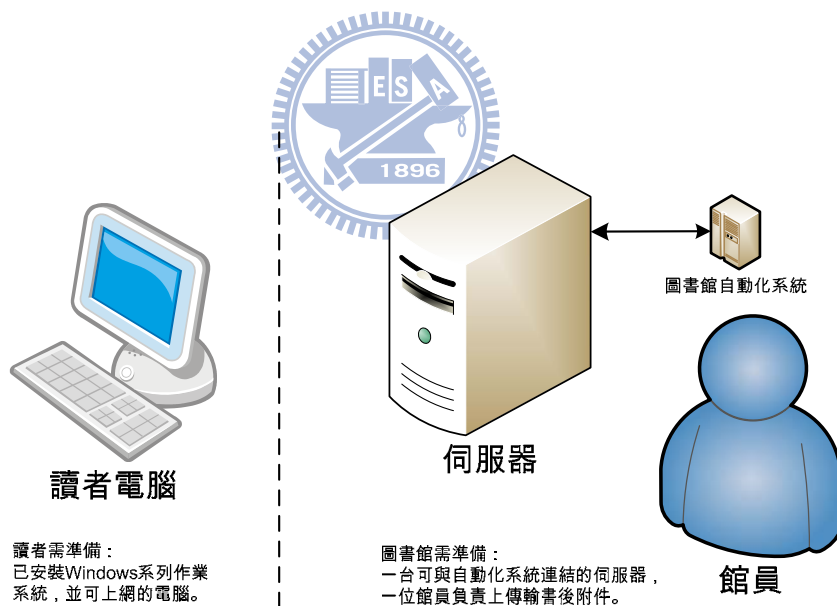


圖 3-1 讀者與圖書館使用 DOD 系統必備條件

本研究為便利館員上傳書後附件尚提供一個上傳書後附件檔案的應用程式，稱之為 Uploader。圖 3-2 說明 DOD 系統提供的三個應用程式，其中 GUI 與 API 之間需交換資料，而 Uploader 與整個系統較無關聯。使用坊間的應用軟體仍可達成 Uploader 所有的功能，Uploader 可以將其視為一個加值 (Value Added) 軟體。

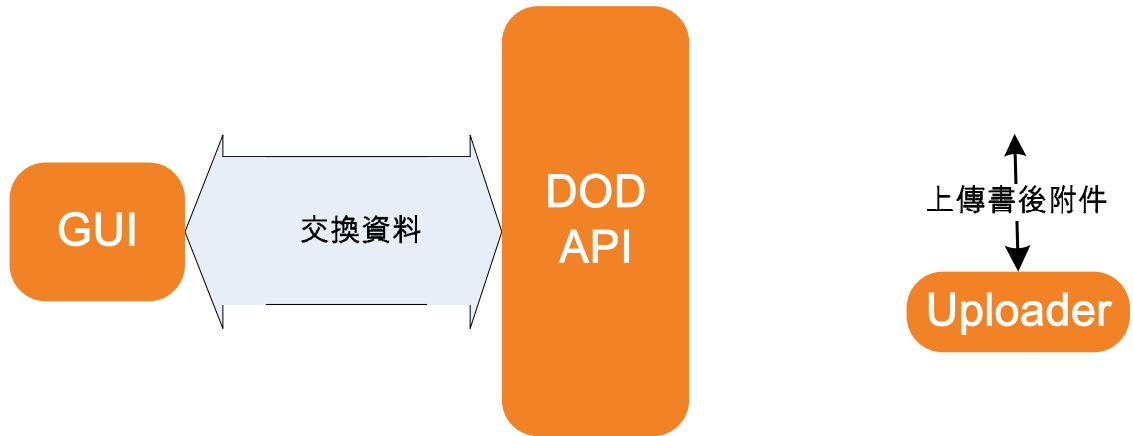


圖 3-2 DOD 系統提供的三個軟體

將圖 3-1 與圖 3-2 組合即可得到圖 3-3，其中 GUI 運行於讀者電腦，API 運行於伺服器，館員使用 Uploader 上傳書後附件檔案至伺服器。

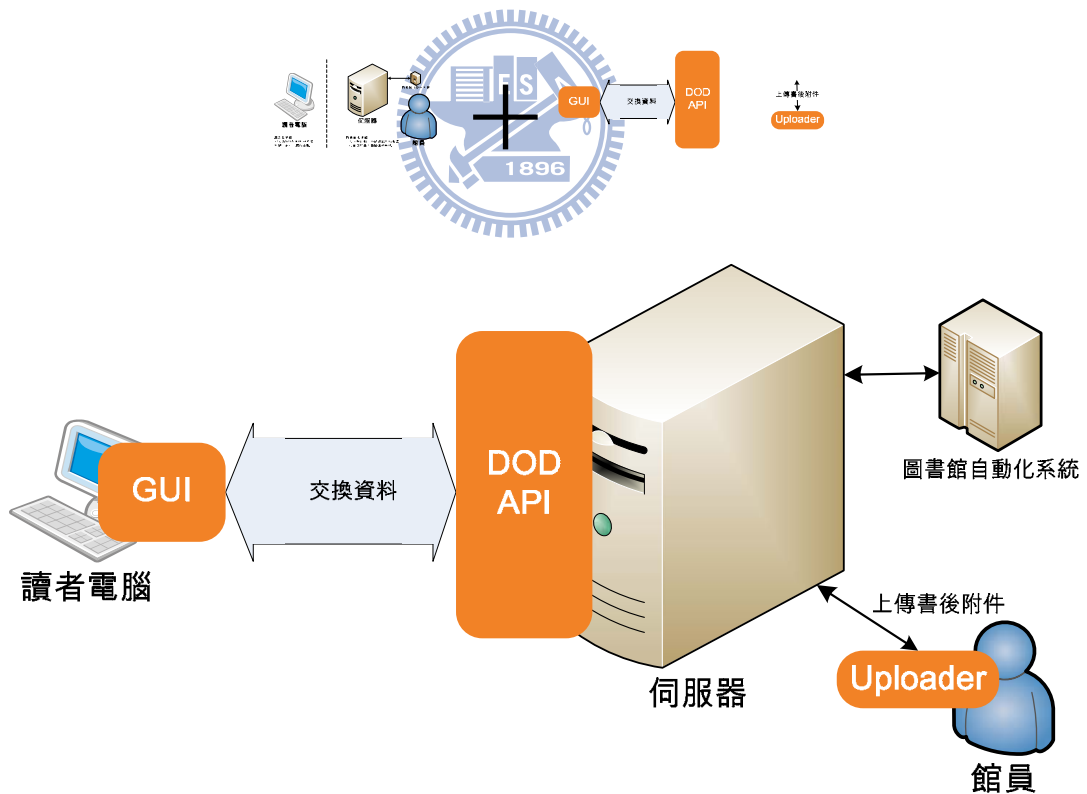


圖 3-3 DOD 系統架構圖

本研究建議再將伺服器分為如圖 3-4 的兩個伺服器

1. 認證伺服器

認證伺服器需與自動化系統溝通，進行帳號密碼認證與取得讀者目前借閱紀錄。本研究建議認證伺服器需置放於防火牆之後，只開啟特定通訊埠 (Port)，不應直接暴露於網際網路中以確保安全。

2. 檔案伺服器

檔案伺服器負責將書後附件的檔案經由網路傳送給讀者，因此會佔用大量頻寬，並不適合放在防火牆之後。此外檔案伺服器需提供讓 Uploader 上傳書後附件檔案的機制，本研究使用免費文件傳輸協議 (File Transfer Protocol, FTP) 伺服器—FileZilla Server[39] 並設定密碼與鎖定 IP，僅允許執行 Uploader 電腦的 IP 存取 FTP 伺服器。

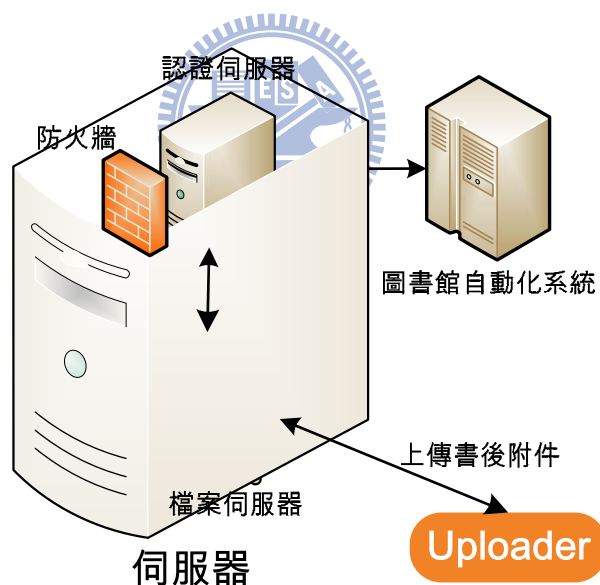


圖 3-4 分離伺服器示意圖

如圖 3-5，GUI 與 DOD API 交換資料的劇本(Scenarios)也可細分為八個步驟：

1. 要求認證

讀者藉由個人電腦上的 GUI 向 DOD API 提出認證要求。

2. 回答認證結果

DOD API 將監測讀者的登入行為，決定是否需要讀者進行 CAPTCHA 挑戰，若需 CAPTCHA 挑戰，則讀者除提供帳號密碼之外尚須提供驗證碼進行認證。

3. 要求借閱紀錄

GUI 認證成功後，向 DOD API 要求借閱紀錄。

4. 回答借閱紀錄

DOD API 向圖書館自動化系統取得讀者借閱紀錄後回傳給 GUI。

5. 要求書後附件檔案結構

GUI 向 DOD API 要求書後附件檔案結構，用以建立虛擬磁碟。

6. 回答書後附件檔案結構

DOD API 回答書後附件檔案結構。

7. 要求下載書後附件檔案

GUI 依照借閱紀錄向 DOD API 要求書後附件的檔案。

8. 回傳書後附件檔案

DOD API 回傳要求的檔案給 GUI。

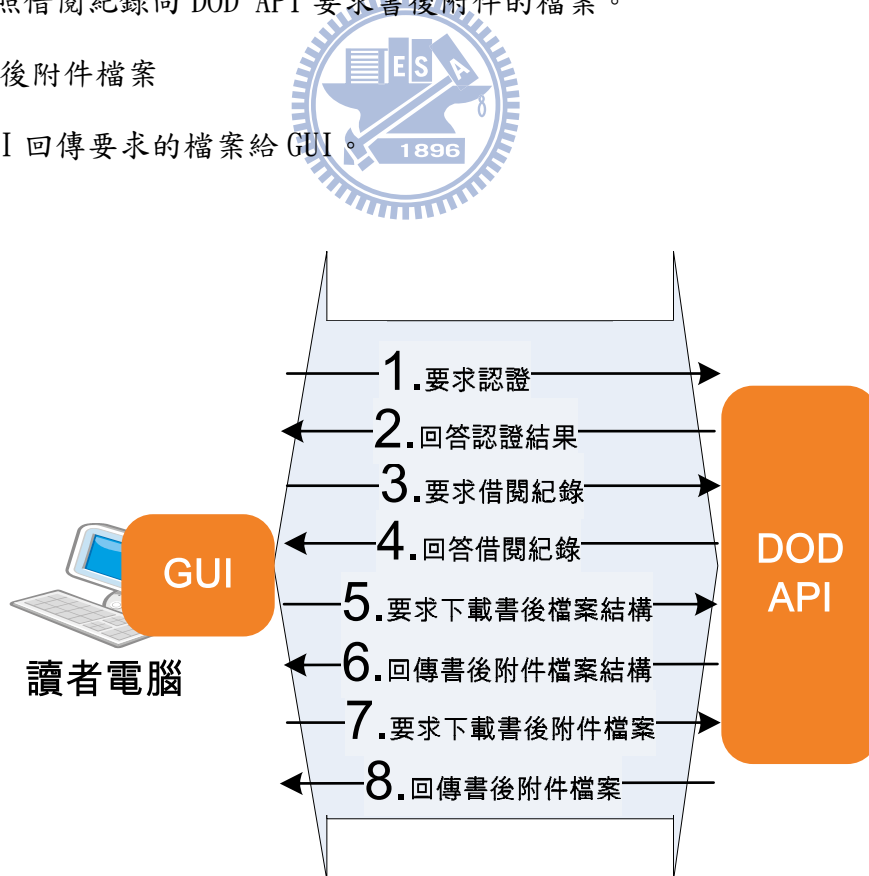


圖 3-5 GUI 與 DOD API 溝通腳本

再將

圖 3-3、圖 3-4 與圖 3-5 結合，得到 DOD 系統的詳細架構示意圖如圖 3-6 所示。
本研究將繼續於 3.2、3.3、3.4、3.5 節，詳述 DOD API、Uploader 與 GUI 的實作方式。

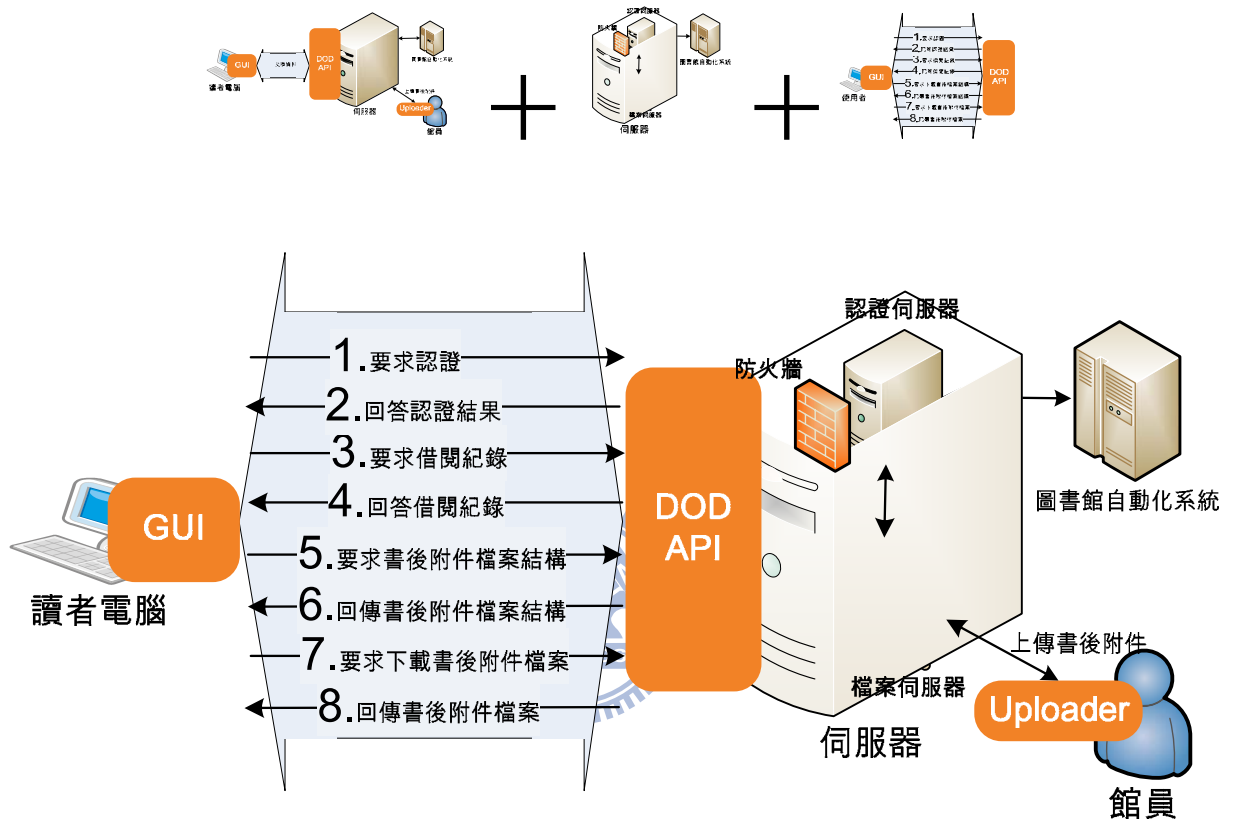


圖 3-6 DOD 系統詳細架構示意圖

3.2 應用程式介面實作

3.2.1 適配器模式

在 2.3 節中曾經說明本研究使用 Adapter 模式連結圖書館自動化系統。Adapter 模式設計流程如圖 3-7。

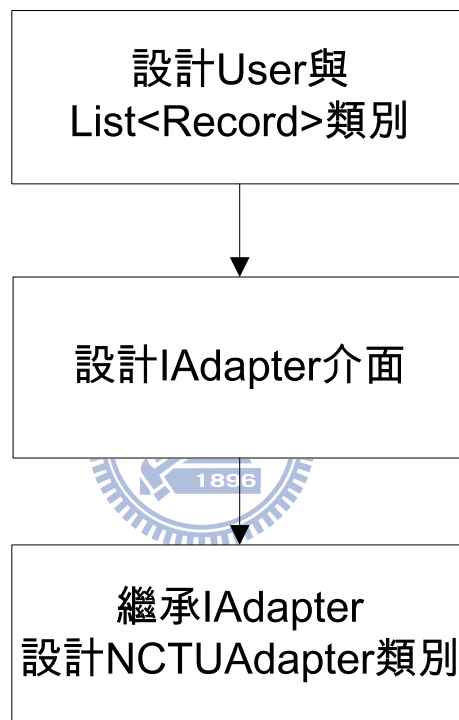


圖 3-7 NCTUAdapter 設計流程圖

首先將讀者與借閱紀錄抽象化 (Abstraction) 並設計成物件。所謂抽象化是將事物與 DOD API 無關的屬性 (Property) 忽略，僅留下對 DOD API 有用的屬性。例如：讀者的籍貫、人種與喜好都是讀者的屬性但是與 DOD API 無關，故忽略之；讀者在自動化系統中的帳號與密碼需使用於 DOD API 中，因此保留。各物件類別圖如

圖 3-8。User 物件代表讀者，Record 物件代表借閱紀錄，List<Record>代表借閱紀錄列表，Rights 代表借閱紀錄的開始與結束時間。其中 List<Record>還需一個 ToXml 方法，返回本身序列化(Serialize)後的 XML 字串。

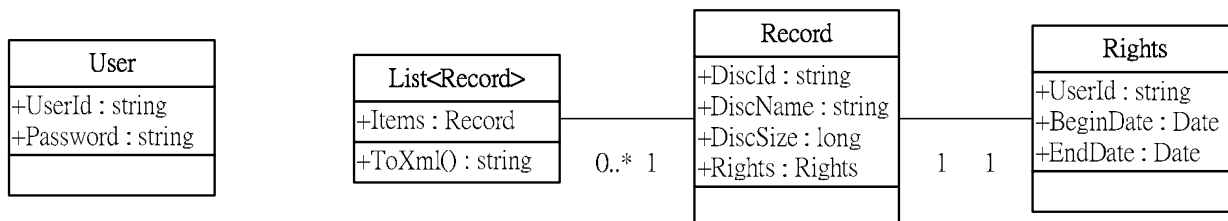


圖 3-8 User、Record、List<Record>與 Rights 類別圖

接著設計 IAdapter 介面 (Interface)，IAdapter 有兩個方法(Method)，類別圖如圖 3-9

1. GetUser 方法，接受使用者帳號 (userid) 與密碼 (password) 為參數，並傳回 User。
2. GetRecordsBuUserId 方法，接受使用者帳號 (userid) 與密碼 (password) 為參數，並傳回該 User 目前的 List<Record>。

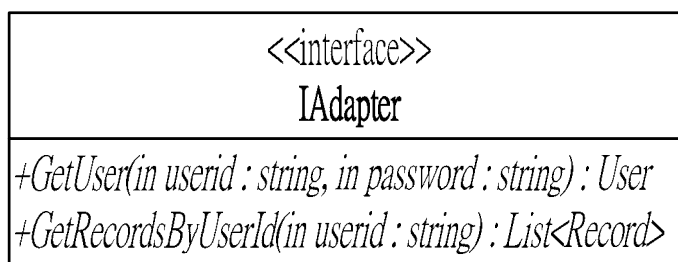


圖 3-9 IAdapter 介面類別圖

最後以浩然圖書館為例，本研究設計一個繼承自 IAdapter 介面實作 NCTUAdapter 類別，如圖 3-10。

1. GerUser 方法以讀者的帳號與密碼為參數，呼叫自動化系統 X Service²⁰提供的 bor-auth 服務，如果認證成功就傳回 User，反之傳回空值 (NULL)。
2. GetRecordsByUserId 則呼叫 bor-info 服務，成功後 X Service 會傳回 XML 表示的讀者借閱紀錄，解析 XML 得到 List<Record>。

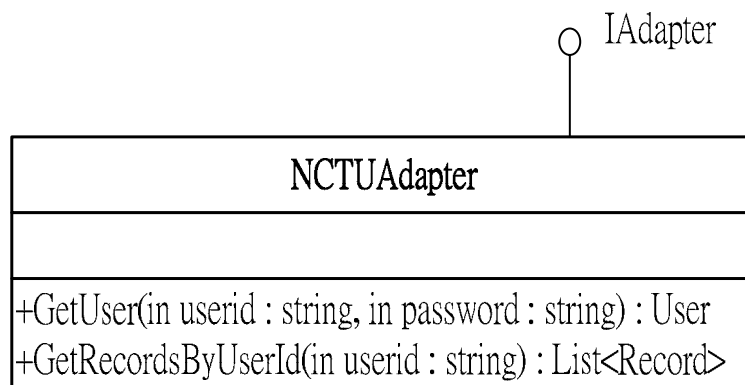


圖 3-10 NCTUAdapter 類別圖

完成 NCTUAdapter 後，即可以在 API 需要與自動化系統溝通時實例化 NCTUAdapter，並呼叫 GetUser 與 GetRecordsByUserID 方法。

3.2.2 Token 機制

HTTP 協議本身是一種無狀態 (Statelessness) 通訊方式。當使用者發出請求，伺服器回應之後，兩者的關係就此結束。即使是同一台電腦，同一個 IP，在極短時間內再發出請求，伺服器會將其當成一個全新的對象，DOD API 為了維持狀態使用 Token 機制。

DOD API 接收讀者送出帳號密碼後，轉送給自動化系統判斷帳號密碼是否正確，如果正確會返回一個 Token，Token 並不含使用者帳號密碼，也非帳號密碼加密結果，而

²⁰ Ex Libris 公司提供的 REST 風格 API 介面。

是由 DOD API 隨機產生一組對整個系統而言唯一的字串，並以此 Token 為索引在記憶體中記錄相關資訊，例如：讀者帳號與借閱紀錄。之後對 DOD API 發出要求都需包含此 Token，DOD API 收到 Token 後，用 Token 反查存放在記憶體中的相關資訊，即可得知該 Token 所代表的讀者是誰，借閱紀錄有哪些。

此外為了控管書後附件檔案同時被取得的數量還需要另外一塊記憶體，以讀者帳號為索引，儲存該名讀者最近 (Last) 一次登入取得的 Token，不論讀者在幾台電腦登入，都可以用 Token 取得目前借閱紀錄列表，但只有最近一次的 Token 才能取得目錄與檔案結構與下載檔案。Token 機制示意圖如圖 3-11。

使用 Token 的另一個目的是，如果每一次讀者發出要求 DOD API 就向自動化系統查詢一次資訊，將大大拖垮 DOD API 自己與自動化系統的效能。使用 Token 後，只有讀者第一次要求時必須向圖書館自動化系統查詢資訊並記錄於記憶體，之後的要求直接從記憶體取出，可大大提高速度。

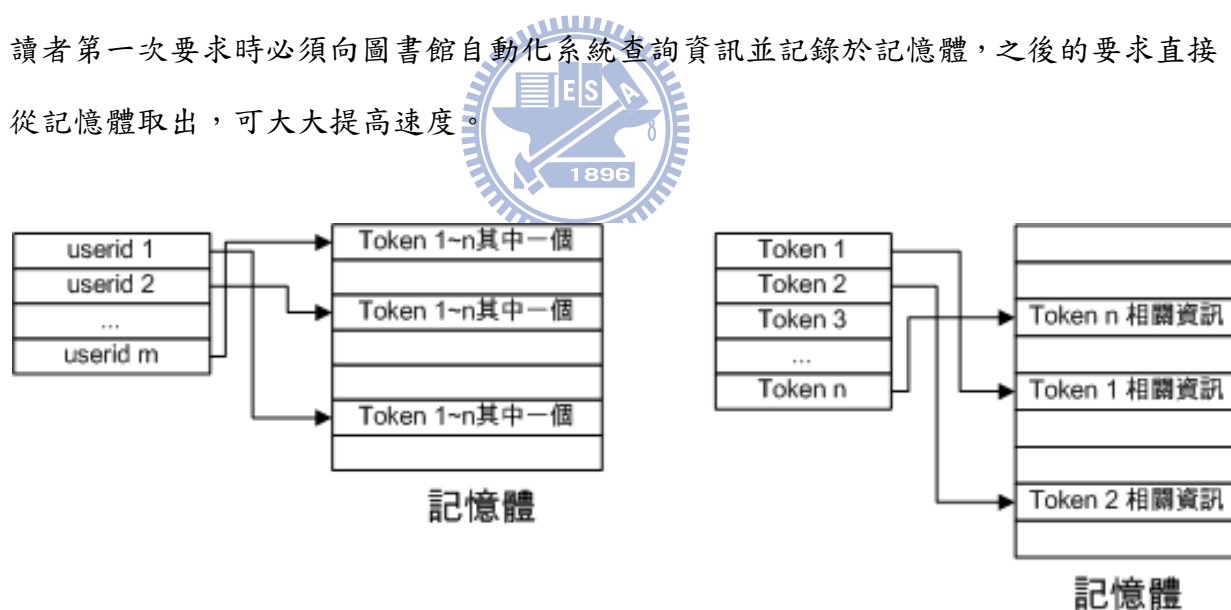


圖 3-11 Token 機制示意圖

3.2.3 書後附件檔案儲存方式

書後附件檔案在伺服器的儲存方式為，以書後附件的條碼號為目錄名稱，並將書後附件實體光碟的目錄與檔案結構完全複製到目錄中。如圖 3-12。但是有部分書後附件

並非以檔案的方式儲存，例如：音樂 CD(Audio CD)，音樂 CD 中的聲音以音軌的方式儲存，此時需要擷取音軌並轉換成 MP3 格式的檔案後，依序以 Track01.mp3、Track02.mp3 等等的名稱儲存於伺服器中。

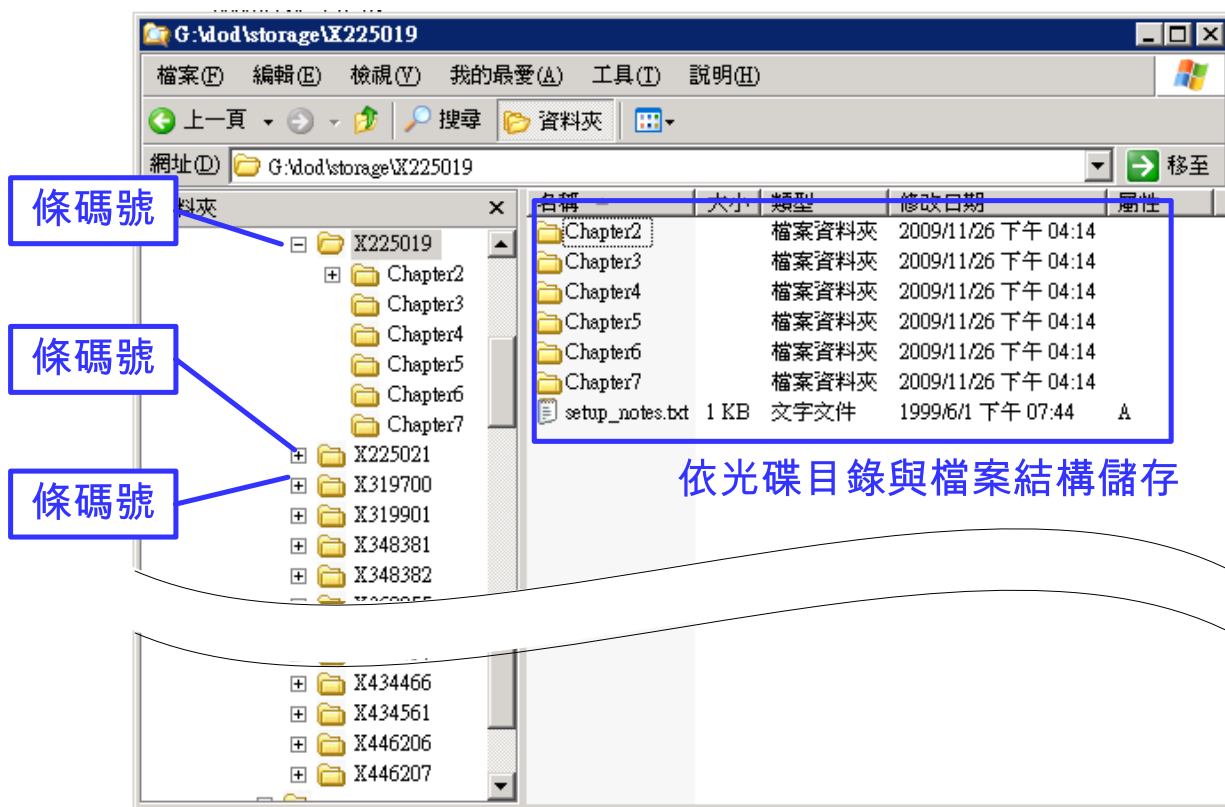


圖 3-12 書後附件檔案儲存方式

3.2.4 應用程式介面提供的服務

如圖 3-13，DOD API 由 NCTUAdapter、Token 機制與書後附件檔案三個部分組成，並藉由這三部分交互作用提供 DOD API 的各項服務 (Services)，服務的要求方式為以 URI 搭配 HTTP 方法 (GET、POST 等等)。

其中 NCTUAdapter 可能因更換自動化系統而被改寫；Token 可使用資料庫而非記憶體儲存；書後附件檔案可以被存成 ISO 9660 格式或是壓縮檔，收到要求時再經過解壓

縮後傳回，甚至存在雲儲存 (Cloud Storage) 中。不管內部的如何變化，DOD API 需保證對外開放的服務其要求方式不變。

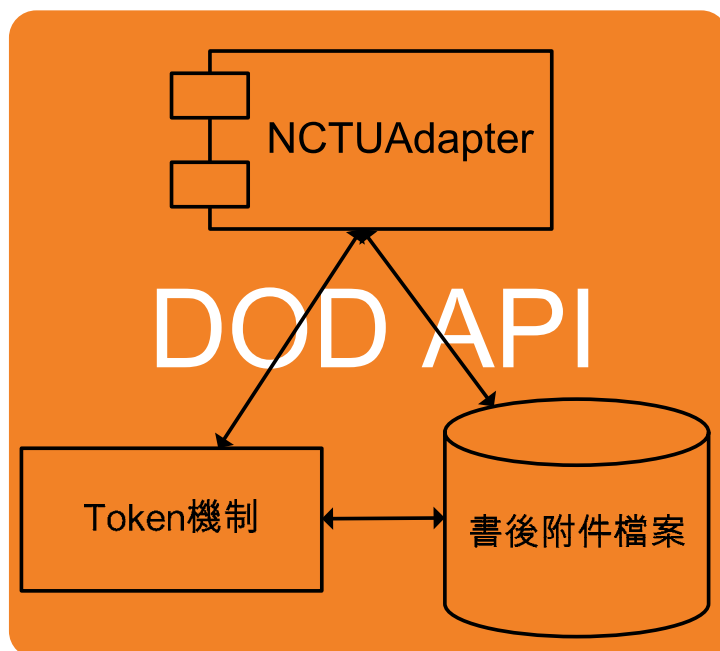


圖 3-13 DOD API 組成圖

DOD API 總共包含了五個服務：

1. 驗證帳號密碼 <http://ServerIP/accounts/login>
2. 取得驗證圖片 <http://ServerIP/accounts/captcha>
3. 取得讀者目前借閱紀錄 <http://ServerIP/records>
4. 取得書後附件檔案結構 [http://ServerIP/discs/\[條碼號\]](http://ServerIP/discs/[條碼號])
5. 下載書後附件檔案 [http://ServerIP/discs/\[條碼號\]/\[檔案名稱\]](http://ServerIP/discs/[條碼號]/[檔案名稱])

將五個服務分述於下：

1. 驗證帳號密碼 `http://ServerIP/accounts/login`

此服務最主要目的為檢查帳號密碼並發給一個 Token，使用者之後每次要求其它 API 服務時，必需將此 Token 附加於 HTTP 要求中，詳細流程圖如圖 3-14。回傳的結果為圖 3-15。此功能設計上較為複雜，需考慮下列三點：

- (1). Token 產生後不能永遠存在記憶體中，否則將耗盡所有記憶體資源，必須有機制對不再使用的 Token 進行刪除並將記憶體回收，本研究的作法是如果在六十分鐘內此 Token 未被使用就刪除 Token。
- (2). 由帳號密碼的錯誤次數，送出要求的頻率等等特徵，判斷要求此服務的是人還是機器，決定要求 CAPTCHA 挑戰。本研究的作法是只要帳號密碼錯誤一次就要求 CAPTCHA 挑戰。
- (3). 與自動化系統連結認證帳號密碼則是使用 NCTUAdapter 以傳入的帳號 (userid) 與密碼(password)為參數呼叫 GetUser 方法，如果傳回的 User 物件不為 NULL 即代表認證成功並發給 Token。



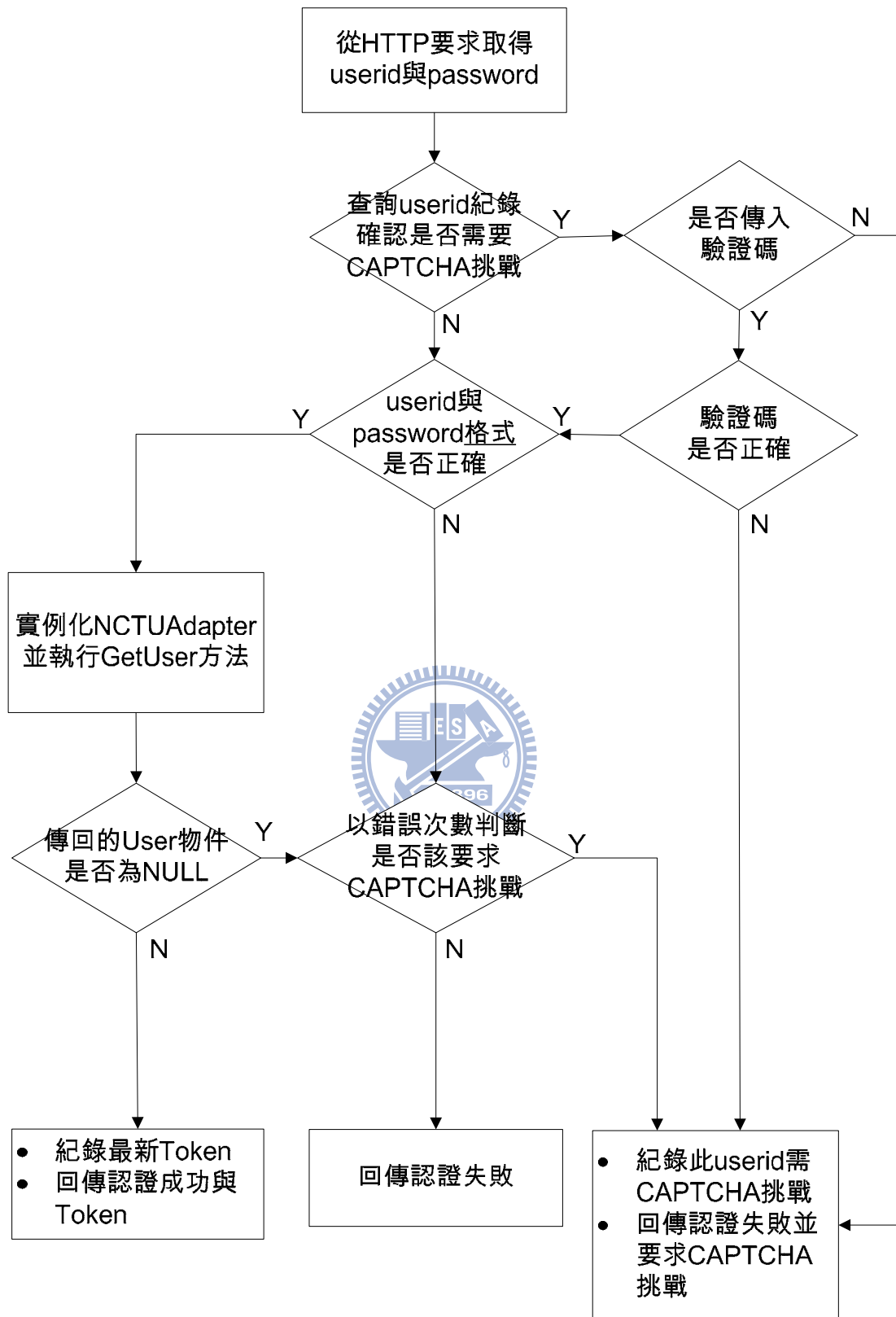



圖 3-14 DOD API 之帳號密碼認證服務流程圖

<p>認證成功回傳 Token</p> <pre><?xml version="1.0" encoding="utf-8"?> <Token> <Id>ASP.NET_SessionId=ipstt345ohytw255rxwgnkfs</Id> </Token></pre>	<p><Id>：即為 token</p>
<p>認證失敗</p> <pre><?xml version="1.0" encoding="utf-8"?> <Error> <Message> UnAuthentication</Message> <Token/> </Error></pre>	
<p>認證失敗要求進行 CAPTCHA 挑戰</p> <pre><?xml version="1.0" encoding="utf-8"?> <Error> <Message>CaptchaRequired</Message> <Token>ASP.NET_SessionId=knwvmq332om0ee3o3r3dvhbn</Token> </Error></pre>	<p><Token>：用以取得驗證圖片的 token，並非系統 token</p>

圖 3-15 Token 之 XML 格式示例與說明

2. 取得驗證圖片 <http://ServerIP/accounts/captcha>

當 DOD API 判斷需要 CAPTCHA 挑戰時，會回傳使用者如圖 3-15 第三部分格式的 XML，使用者者解析 XML 得到 Token (並非認證成功後傳回的 Token)，並將其包含在 HTTP 表頭中，對 <http://ServerIP/accounts/captcha> 發出 HTTP 請求，即可得到一個如圖 3-16 的圖形驗證碼。

HTTP 要求(Request)
GET /accounts/captcha HTTP/1.1 Host : http://ServerIP Cookie : ASP.NET_SessionId=knwvmq332om0ee3o3r3dvhbn
HTTP 回應(Response)

圖 3-16 圖形驗證碼



3. 取得讀者目前借閱紀錄 <http://ServerIP/records>

此服務功能為傳回讀者的目前借閱紀錄，使用者發出的 HTTP 請求需包含認證成功後取得的 Token，DOD API 以 Token 判斷讀者是否登入成功，之後以 Token 為索引取得帳號(userid)，並實例化 NCTUAdapter 使用 GetRecordsByUserId 方法取得讀者目前借閱紀錄回傳給讀者，詳細的流程圖如圖 3-17。回傳的結果為圖 3-18。

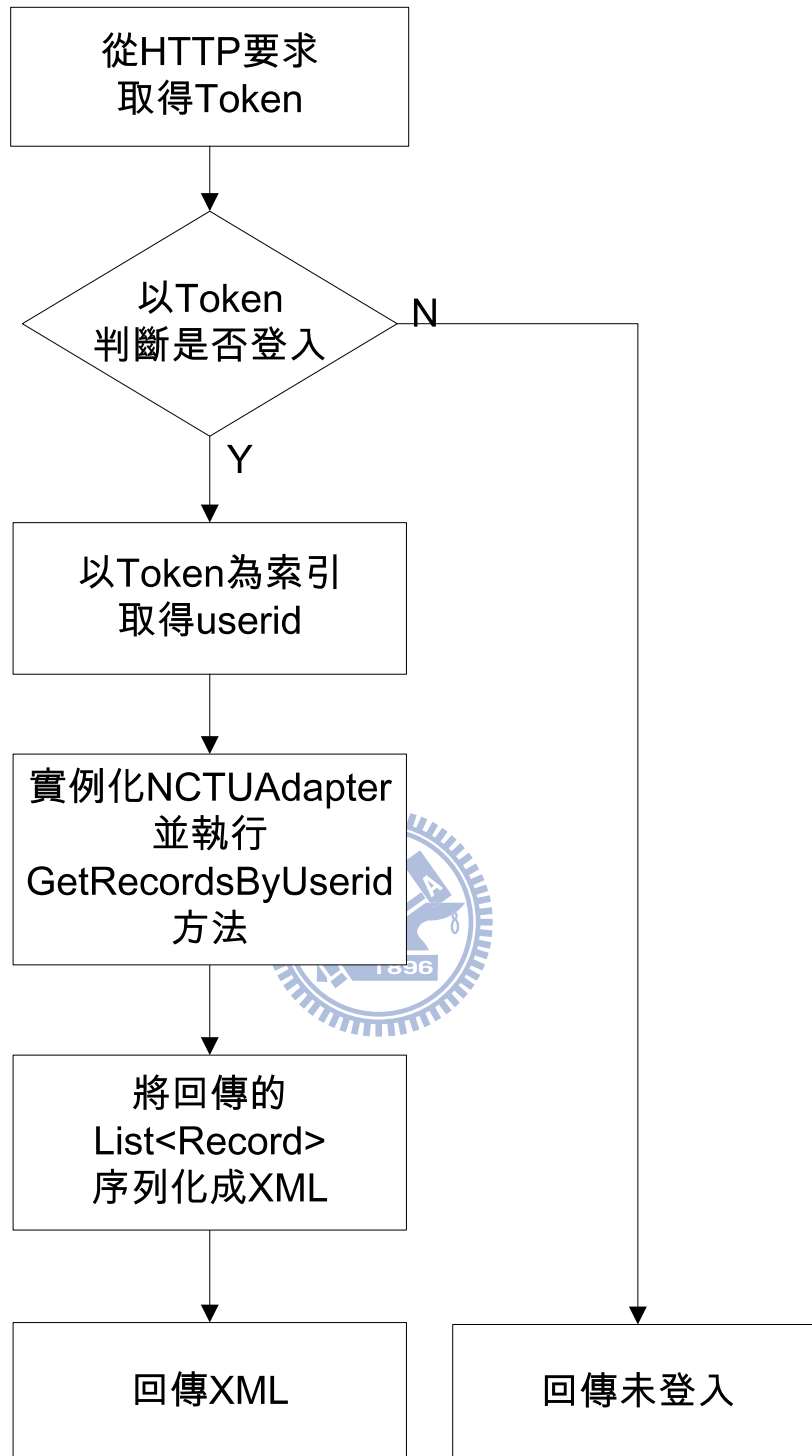


圖 3-17 DOD API 之取得讀者目前借閱紀錄服務流程圖

```

<?xml version="1.0" encoding="utf-8"?>
<List>
  <item>
    <value>
      <Record>
        <DiscId>X225019</DiscId>
        <DiscName>[BCD]網路應用程式設計指南 /</DiscName>
        <DiscSize>640000000</DiscSize>
        <Rights>
          <UserId>0796795451</UserId>
          <BeginDate>2009-11-02T12:12:00</BeginDate>
          <EndDate>2010-01-02T23:59:00</EndDate>
        </Rights>
      </Record>
    </value>
  </item>
  <item>
    <value>
      <Record>
        <DiscId>X319700</DiscId>
        <DiscName>[BCD]Office 2003 論文寫作實務 /</DiscName>
        <DiscSize>640000000</DiscSize>
        <Rights>
          <UserId>0796795451</UserId>
          <BeginDate>2009-11-02T12:12:00</BeginDate>
          <EndDate>2010-01-02T23:59:00</EndDate>
        </Rights>
      </Record>
    </value>
  </item>
  .....
</List>

```

<DiscId>：條碼號(唯一 ID) <DiscName>：書後附件名稱 <DiscSize>：書後附件光碟容量 <UserId>：讀者 ID <BeginDate>：借閱日期 <EndDate>：歸還日期

圖 3-18 借閱紀錄 XML 格式示例與說明

4. 取得書後附件檔案結構 [http://ServerIP/discs/\[條碼號\]](http://ServerIP/discs/[條碼號])

此服務功能為取得指定條碼號的書後附件檔案結構，使用者除傳入 Token 外還需提供書後附件條碼號(discid)，DOD API 將書後附件的目錄與檔案結構表示成 XML 格式，並回傳給使用者，詳細的流程圖如圖 3-19。回傳的結果為圖 3-20。

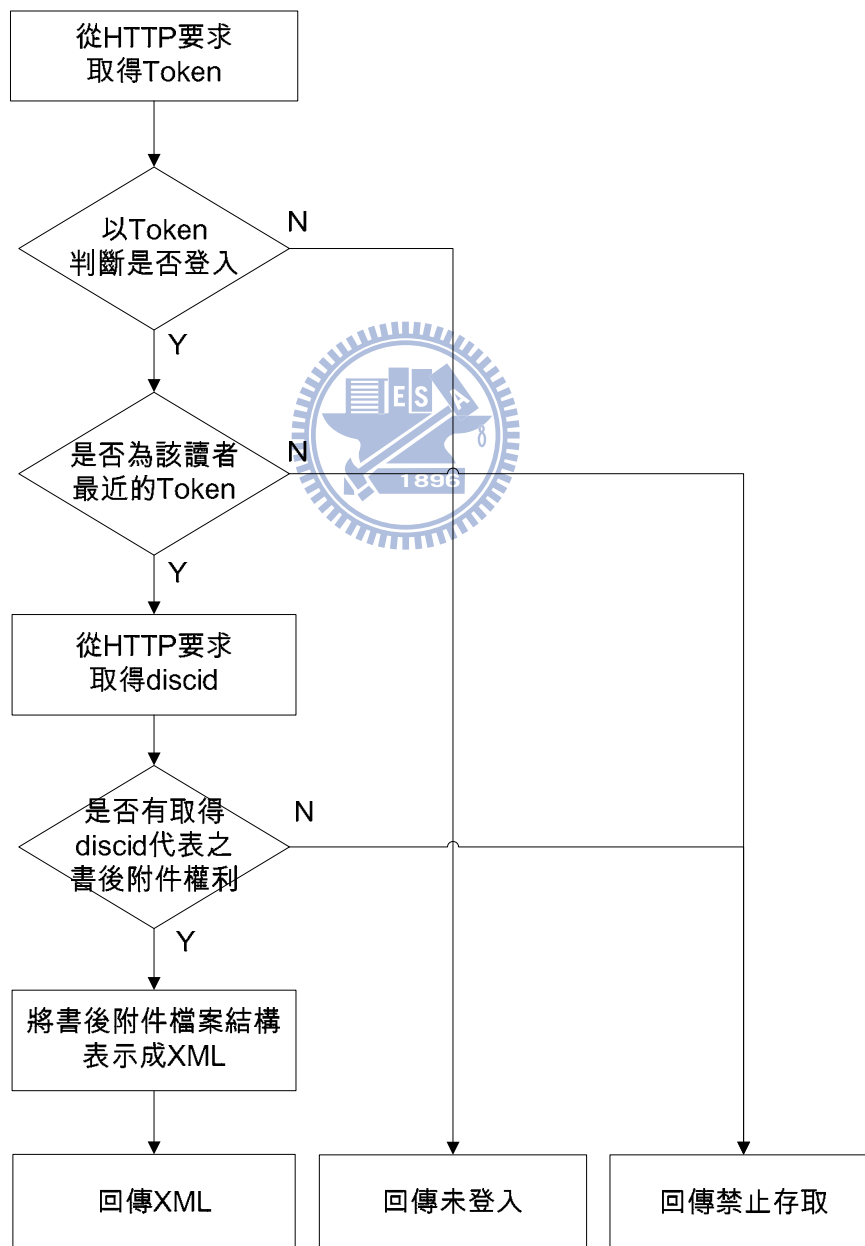


圖 3-19 DOD API 之取得書後附件目錄與檔案結構服務流程圖


```

<?xml version="1.0" encoding="utf-8"?>
<Dictionary>
  <item>
    <key>
      <string>\setup_notes.txt</string>
    </key>
    <value>
      <FileInformation>
        <Attributes>Archive</Attributes>
        <CreationTime>2009-11-18T19:06:00.84375+08:00</CreationTime>
        <LastAccessTime>2009-11-20T04:51:42.9375+08:00</LastAccessTime>
        <LastWriteTime>1999-06-01T19:44:08+08:00</LastWriteTime>
        <Length>825</Length>
        <FileName>setup_notes.txt</FileName>
      </FileInformation>
    </value>
  </item>
  <item>
    <key>
      <string>\Chapter2</string>
    </key>
    <value>
      <FileInformation>
        <Attributes>Directory</Attributes>
        <CreationTime>2009-11-18T19:06:00.84375+08:00</CreationTime>
        <LastAccessTime>2009-11-25T15:51:42.03125+08:00</LastAccessTime>
        <LastWriteTime>2009-11-18T19:06:00.96875+08:00</LastWriteTime>
        <Length>0</Length>
        <FileName>Chapter2</FileName>
      </FileInformation>
    </value>
  </item>
  .....
</Dictionary>

```

<string>：包含路徑的檔名 <Attributes>：檔案的屬性 <CreationTime>：建立時間 <LastAccessTime>：最後存取時間 <LastWriteTime>：最後修改時間 <Length>：檔案大小 <FileName>：檔名，不含路徑

圖 3-20 書後附件檔案結構 XML 格式示例與說明

5. 下載書後附件檔案 [http://ServerIP/discs/\[條碼號\]/\[檔案名稱\]](http://ServerIP/discs/[條碼號]/[檔案名稱])

此服務功能為取得書後附件檔案，使用者提供 Token，條碼號與檔案名稱，DOD API 回傳二進位檔案，詳細的流程圖如圖 3-21。

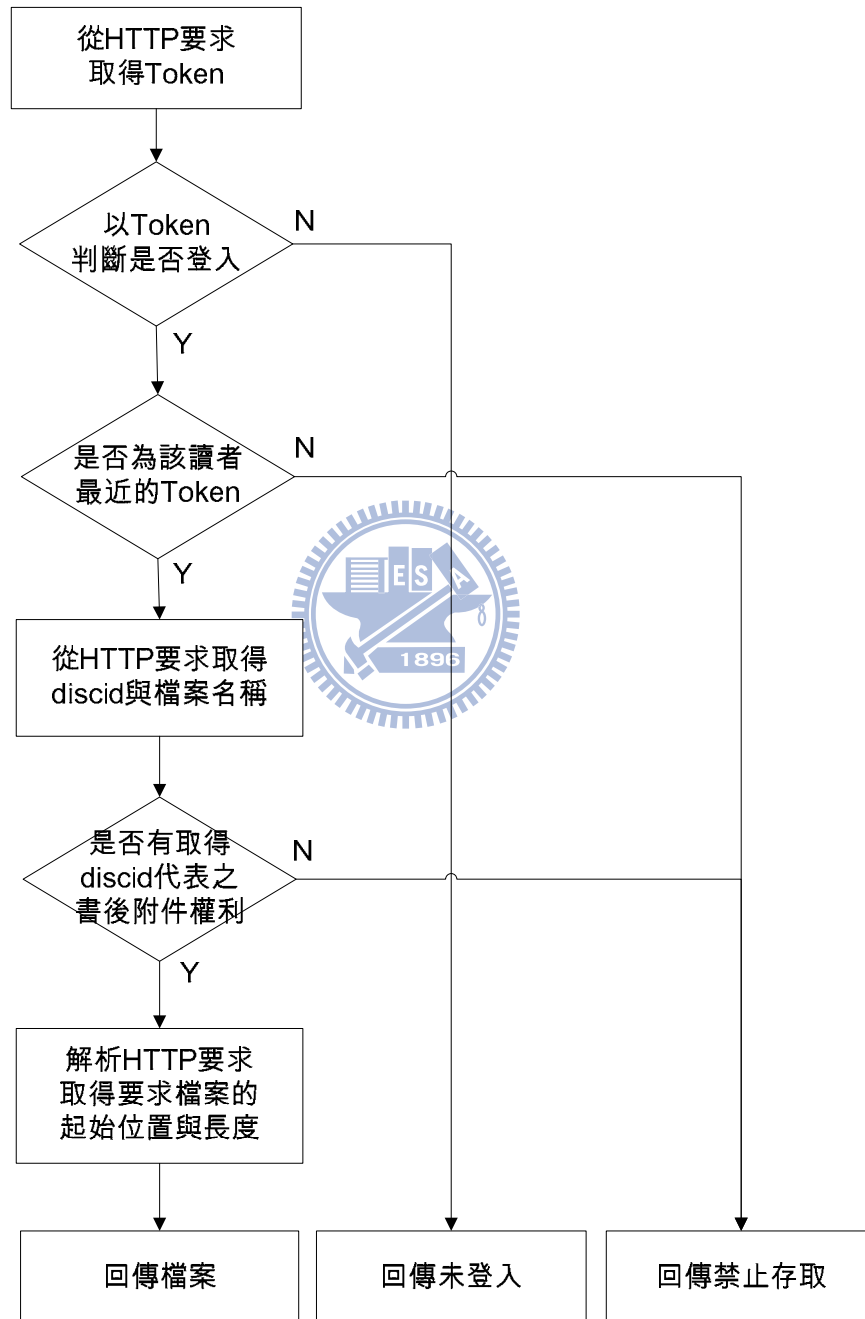


圖 3-21 下載書後附件檔案

3.2.5 應用程式介面設計難題

DOD API 設計的難題為，在 URI 中某些特殊字元與中文傳送到伺服器時會遭到改變，例如：空白字元會被轉換成 %20， / 會被轉換成 %2F 等等。本研究參考目前業界比較常用的做法，要求使用者呼叫 DOD API 下載書後附件檔案服務時，需先以 Base64²¹ 將檔案名稱編碼，

以取得 X000001 條碼號書後附件中 \C01\資料庫備份\TestProvider_bak 檔案為例：

1. 「\C01\資料庫備份\TestProvider.bak」經過 Base64 編碼後變成

「L0MwMS-os4fmlpnluqvlgpknku70vVGVzdFByb3ZpZGVyX2Jhaw==」

2. http://ServerIP/discs/X000001\C01\資料庫備份\TestProvider_bak 需改成

http://ServerIP/discs/X000001/L0MwMS-os4fmlpnluqvlgpknku70vVGVzdFByb3ZpZGVyX2Jhaw== 才是 DOD API 真正接受的格式。



²¹ Base64 僅用可列印的 62+2 個字元編碼，因此可以將中文字或二進位檔(Binary File)編碼後以純文字的方式進行傳輸。

3.3 館員上傳程式實作

Uploader 提供館員一鍵 (One-Click) 上傳書後附件的功能。其運作流程如圖 3-22。設定畫面如圖 3-23。實際運作畫面如圖 3-24。

Uploader 的誕生起因於館員抱怨上傳音樂 CD 格式的書後附件流程過於複雜。上傳音樂 CD 格式的書後附件需擷取音樂 CD 中的音軌成.wav 檔，再轉檔成.mp3 檔，然後上傳至檔案伺服器，以人工方式處理容易出錯，故設計 Uploader 完成這些工作。程式可以設定放入書後附件的光碟機代號，存放書後附件檔案之 FTP 伺服器的 IP 與帳號密碼。暫存路徑為暫存.wav 檔案的位置，為了處理 DVD 光碟，要求需有 4.7GB 以上的空間。

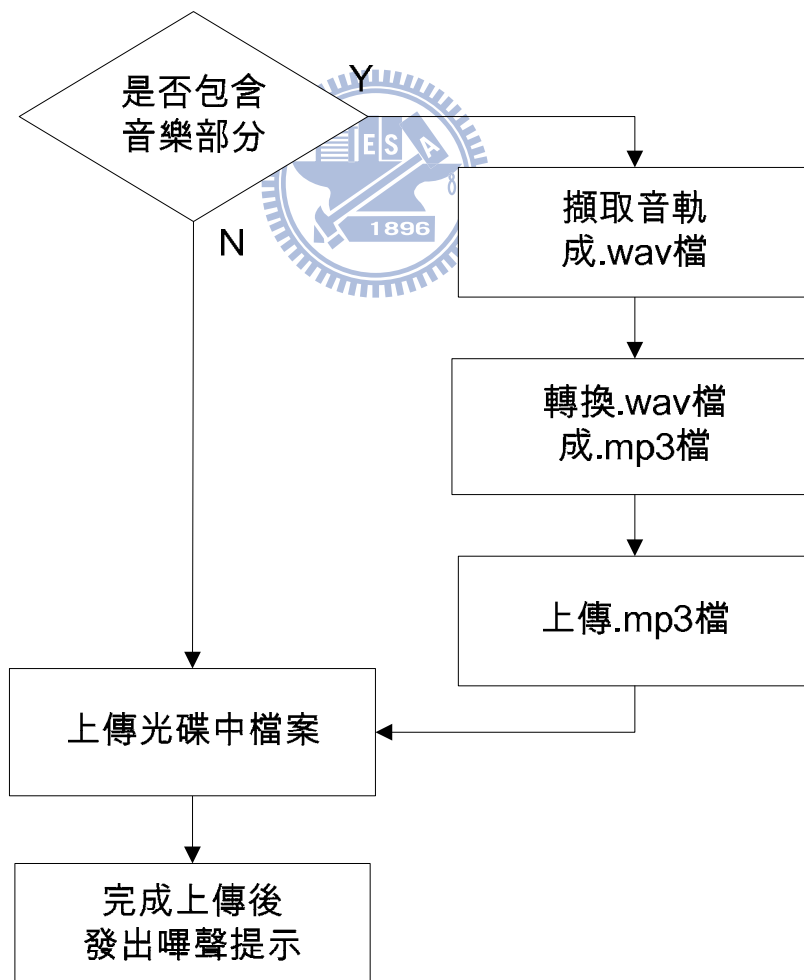


圖 3-22 Uploader 運作流程圖



圖 3-23 Uploder 設定畫面



圖 3-24 Uploder 運行畫面

3.4 圖形化使用者介面實作

在本研究中由同一個作者設計 DOD API 與 GUI 應是一種特例，現實世界中 API 與 GUI 通常由不同的兩個作者(組織)設計，兩者可能使用不同的作業平台、以不同的程式語言開發並且位於不同的國家，兩者之間唯一的共同語言就是—HTTP 與 XML。

API 的作者戮力於提供 API 更多的功能，將 API 功能劃分的更合理、使用更好的伺服器與提供更穩固的服務等等；而 GUI 的作者則是以自己獨特的創意，設計出更吸引使用者與更友善的介面。

在設計本研究的 GUI 之前，先將前文所有的分析研究與探討都「忘記」，甚至連圖 3-5 所謂 GUI 兩 DOD API 溝通的八個步驟也拋棄，這八個步驟只是 DOD API 的設計者認為 GUI 與 DOD API 之間的劇本，不一定是 GUI 設計者想要的方式。然後將 3.3 節 DOD API 的服務整理於表 3-1，表中的五個服務就是設計 GUI 時，取得所需資訊的唯一途徑。

表 3-1 DOD API 服務列表

服務說明	URI	接收參數	傳回結果
帳號密碼認證	http://ServerIP/accounts/login	userid password (captcha)	XML 字串 圖 3-15
取得 CAPTCHA 驗證圖片	http://ServerIP/accounts/captcha	Token	圖片
取得讀者目前 借閱紀錄	http://ServerIP/records	Token	XML 字串 圖 3-18
取得書後附件 檔案結構	http://ServerIP/discs/[條碼號]	Token discid	XML 字串 圖 3-20
下載書後附件 檔案	http://ServerIP/discs/[條碼號]/[檔案名稱]	Token discid filename	檔案

3.4.1 使用者介面的抉擇

Web-based 介面因其跨平台的優點，本研究曾以瀏覽器讓讀者下載書後附件檔案的方式實作使用者介面。但實際使用時發現，除了使用者需要下載檔案至硬碟與原有使用流程不同之外，最大的問題在於光碟中檔案存在相互參照的特性。以開啟一個最基本的 HTML 檔案為例，如圖 3-25：

```
<html>
  <head>
    <title>首頁</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class=header>
      
    </div>
    <div class=body>
      歡迎來到 XXX 網站
    </div>
    <div class=footer>
      
    </div>
  </body>
</html>
```

檔案目錄結構為

- index.htm
- style.css
- images\
 - header.png
 - footer.png

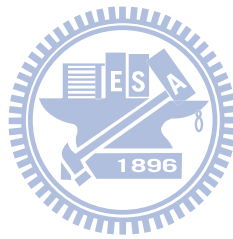
圖 3-25 HTML 內容示例

瀏覽器載入 index.htm 後將會：

1. 在與 index.htm 同一層目錄尋找 style.css 並載入用以配置版面格式，
2. 到下一層的 images 目錄中，尋找 header.png 與 footer.png 檔案並載入，以顯示標題圖片與頁尾圖片。

讀者使用書後附件實體光碟放入光碟機後，開啟 index.htm 一切正常。但是如果只下載 index.htm 至硬碟後開啟，會發現圖片都消失了，網頁版型也不對。因為讀者並不知道還需要下載 style.css，然後在電腦建立一個 images 目錄，之後再下載 banner.png 與 footer.png 到 image 目錄才能正常運作。


光碟中檔案相互參照的特性，屬於無法以技術克服的問題。除非將整個書後附件映像檔下載到電腦後，才能保證在開啟某個檔案時，其他關聯檔案都已經存在。但下載整個映像檔，又會遇到如 2.1 節提及書後附件映像檔被隨意傳播的問題，所以本研究最後並未以 Web-based 介面設計使用者介面，取而代之的是以 FUSE 技術在讀者電腦上建立虛擬磁碟的方式實作使用者介面，解決了檔案相互參照的問題。



3.4.2 登入畫面

需要登入畫面的原因為 DOD API 需要 Token 才能使用其它服務，而從 DOD API 取得 Token 的唯一方法為提供讀者帳號密碼，所以 GUI 設計一個轉送機制，「利用」讀者輸入帳號密碼獲取 Token，讓 GUI 能繼續運作。若 DOD API 回應需 CAPTCHA 挑戰，則先取得驗證碼圖片後顯示於程式畫面中，讀者除輸入帳號密碼外，尚需輸入驗證碼。實際登入畫面和使用 DOD API 時發出的 HTTP 要求與回應如表 3-2。

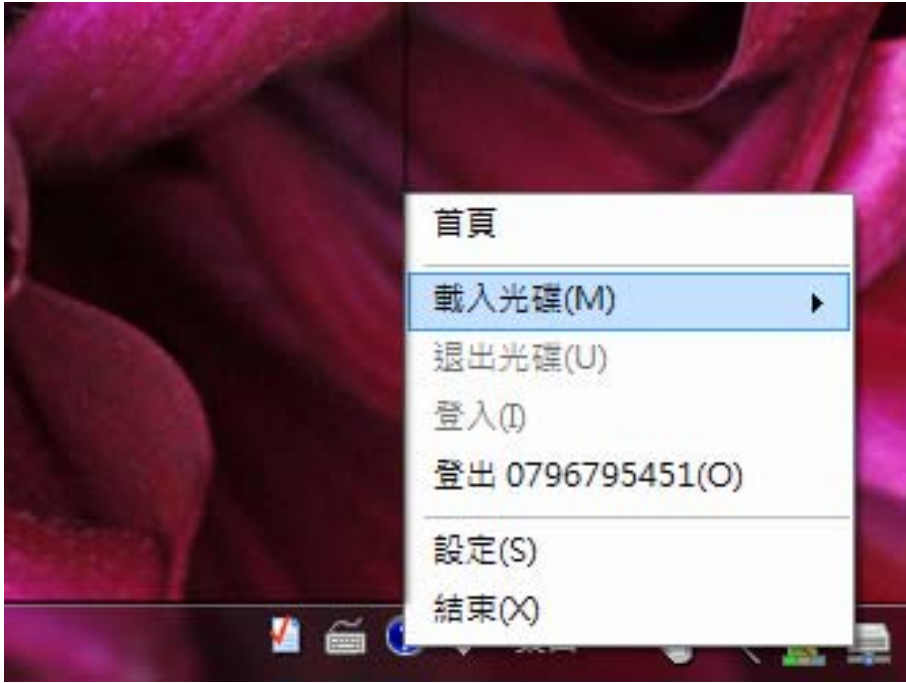
表 3-2 登入畫面

畫面	
	
HTTP 要求	
<p>帳號密碼認證</p> <p>POST /accounts/login HTTP/1.1</p> <p>Host: http://ServerIP</p> <p>-----</p> <p>userid=0796795451&password=xxxxx</p>	<p>取得驗證碼圖片</p> <p>GET /accounts/captcha HTTP/1.1</p> <p>Host: http://ServerIP</p> <p>帳號密碼認證</p> <p>POST /accounts/login HTTP/1.1</p> <p>Host: http://ServerIP</p> <p>-----</p> <p>userid=0796795451&password=xxxxx &captcha=RKZSC</p>
HTTP 回應	
<p>以 XML 表示的 Token，或是需要 CAPTCHA 的訊息，如圖 3-15。</p>	

3.4.3 查看目前借閱紀錄畫面

如表 3-3，將滑鼠停在 [載入光碟(M)]，GUI 會向 DOD API 查詢目前讀者借閱紀錄，並使用彈出式列表 (Popup Menu) 顯示。在畫面中可以看到 [退出光碟(U)] 與 [登入(I)] 被關閉 (Disable)，此為設計 GUI 的基本原則，與其提供說明書告訴使用者，哪裡能按哪裡不能按，不如在設計時就「限制」使用者於各種狀態下能操作的功能。

表 3-3 查看目前借閱紀錄畫面

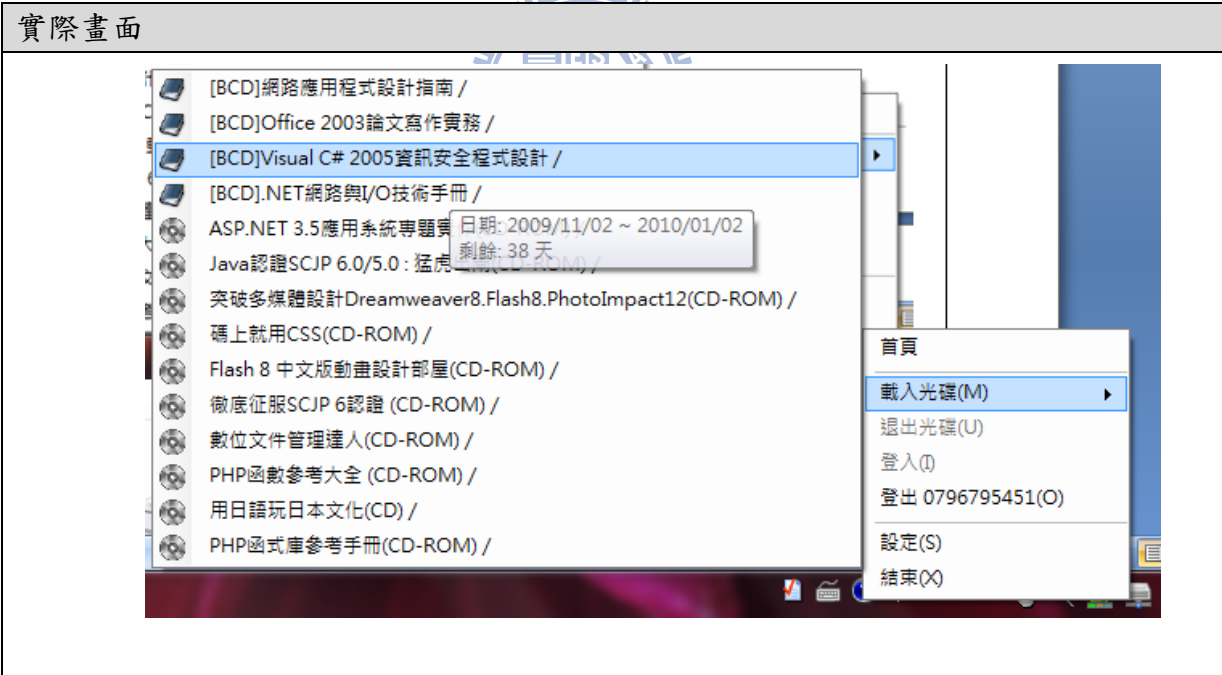
畫面

HTTP 要求
GET /records HTTP/1.1 Host: http://ServerIP Cookie: ASP.NET_SessionId=ipstt345ohytw255rxwgnkfs
HTTP 回應
以 XML 表示的讀者目前借閱紀錄，如圖 3-18。

3.4.4 選擇書後附件畫面

讀者目前的借閱紀錄在 GUI 裡，被設計成一個彈出式的列表，滑鼠移至列表上會顯示簡短提示。列表其實也是一種限制，在列表彈出後鼠標 (Cursor) 必須在列表中才能保持列表開啟，因為鼠標只有一個，所以列表開啟時保證 GUI 其他位置不可能被操作，讀者被「強迫」在移動滑鼠瀏覽列表、使用滑鼠點擊(Click)某個項目與將滑鼠移出列表三個動作之間作出選擇。

直到讀者使用滑鼠點擊列表上代表某個書後附件的項目後，GUI 以被點擊項目代表的書後附件條碼號，向 DOD API 要求該條碼號書後附件目錄與檔案結構。


表 3-4 選擇書後附件畫面

<p>實際畫面</p> 
<p>HTTP 要求</p>
<p>GET /discs/X225019 HTTP/1.1 Host: http://ServerIP Cookie: ASP.NET_SessionId=ipstt345ohytw255rxwgnkfs</p>
<p>HTTP 回應</p>
<p>以 XML 表示的目錄與檔案結構，如圖 3-20。</p>

3.4.5 無縫接成檔案總管

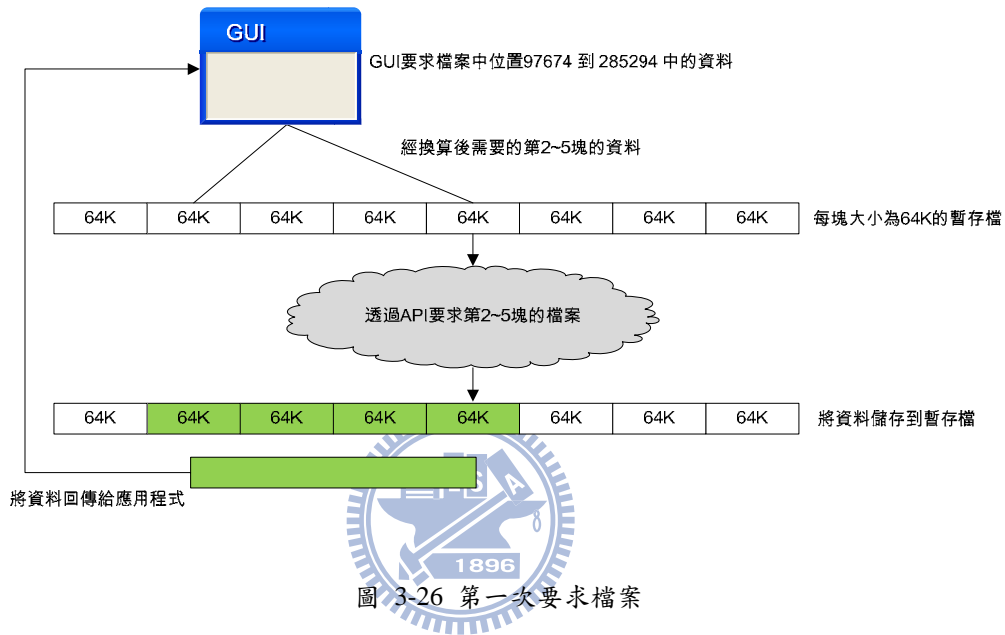
3.4.4 節取得目錄與檔案結構後，即可使用 Dokan Library 在作業系統虛擬出一顆磁碟，如表 3-5，爾後操作介面即由檔案總管接手，讀者如同使用實體書後附件光碟般，以檔案總管瀏覽光碟內容、執行程式與撥放影片等等。GUI 使用 Dokan Library 監視讀者對虛擬磁碟目錄的瀏覽動作並告訴作業系統，讀者切換至的目錄應該有哪些檔案。並且只有在應用程式真正需要開啟檔案(滑鼠點兩下，或是按右鍵選擇開啟程式)時，才向 DOD API 取得需要的檔案內容。

表 3-5 無縫接成檔案總管

<p>實際畫面</p>
 <p>The screenshot shows a Windows File Explorer window with a network drive mapped to G:. The drive contains a file named "[BCD]網路應用程式設計指南 / (G:)". The drive is labeled as "硬碟 (3)".</p>
<p>HTTP 要求</p>
<p>GET /discs/X225019/[Base64 編碼後的檔案名稱] HTTP/1.1 Host: http://ServerIP Cookie: ASP.NET_SessionId=ipstt345ohytw255rxwgnkfs</p>
<p>HTTP 回應</p>
<p>整個或部分檔案。</p>

3.4.6 使用者介面設計難題

GUI 難題為緩存的設計，本研究以設計模式的 Proxy 模式解決這個問題，先將檔案切成每塊固定大小的檔案區塊。圖 3-26 代表第一次要求檔案中內容時，GUI 透過網路取得所有要求的檔案區塊。



GUI 繼續要求檔案內容發現第四與第五塊已經在暫存檔中，所以僅透過網路取得第六與第七塊的檔案區塊，如圖 3-27。

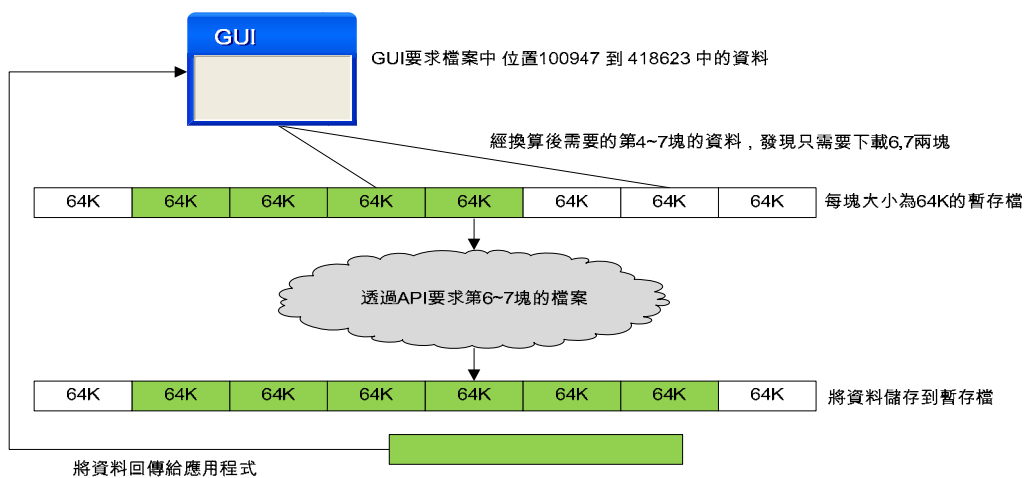


圖 3-28 說明當所有檔案區塊都在暫存檔中，則全部檔案皆直接由暫存檔中取出，不需另外下載。

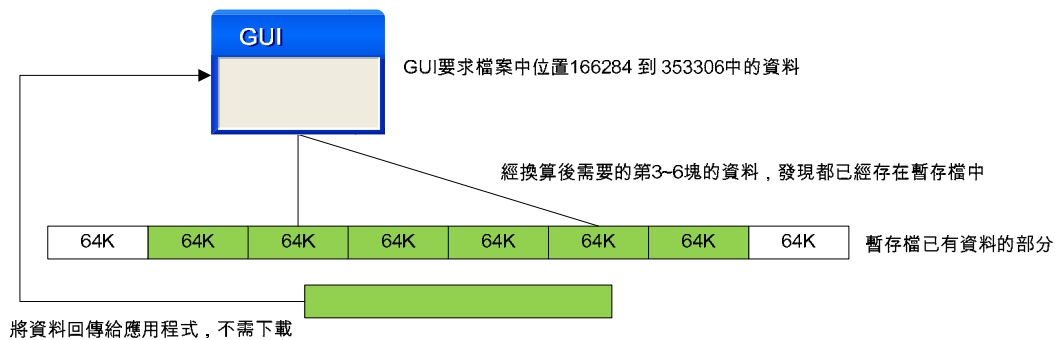


圖 3-28 直接由緩存提供檔案

GUI 實作的緩存系統，由 4.2 節測試結果顯示，第二次存取資料時，速度甚至快於由光碟機讀取，同時緩存也降低了伺服器的負擔並減少網路流量。

3.5 使用者介面發佈與安裝

本研究採用 Inno Setup[40]為 GUI 包裝 (Package) 程式，Inno Setup 具有強大的自訂指令碼(Script)，可以藉由指令碼完成產生安裝檔、註冊動態連結程式、創建安裝目錄與重新開機等等動作。Inno Setup 主要協助本研究完成五項工作：

1. 將 GUI 與所需相關資源包裝成一個安裝檔。
2. 將 Dokan Library 註冊到作業系統。
3. 檢查讀者電腦是否已安裝 .Net Framework 2.0，若未安裝則協助讀者自網路下載 .Net Framework 2.0 安裝檔。
4. 創建桌面或工作列捷徑。

5. 重新啟動作業系統，僅 Windows XP 之前的作業系統需重新啟動，Windows Vista 與 Windows 7 不需重啟即可使用。

實際安裝介面如圖 3-29，安裝程式運作流程如圖 3-30，GUI 的安裝過程與市面上一般應用軟體的安裝過程類似，下一步，下一步...完成，安裝程式以互動的介面提示讀者接下來該進行的步驟。

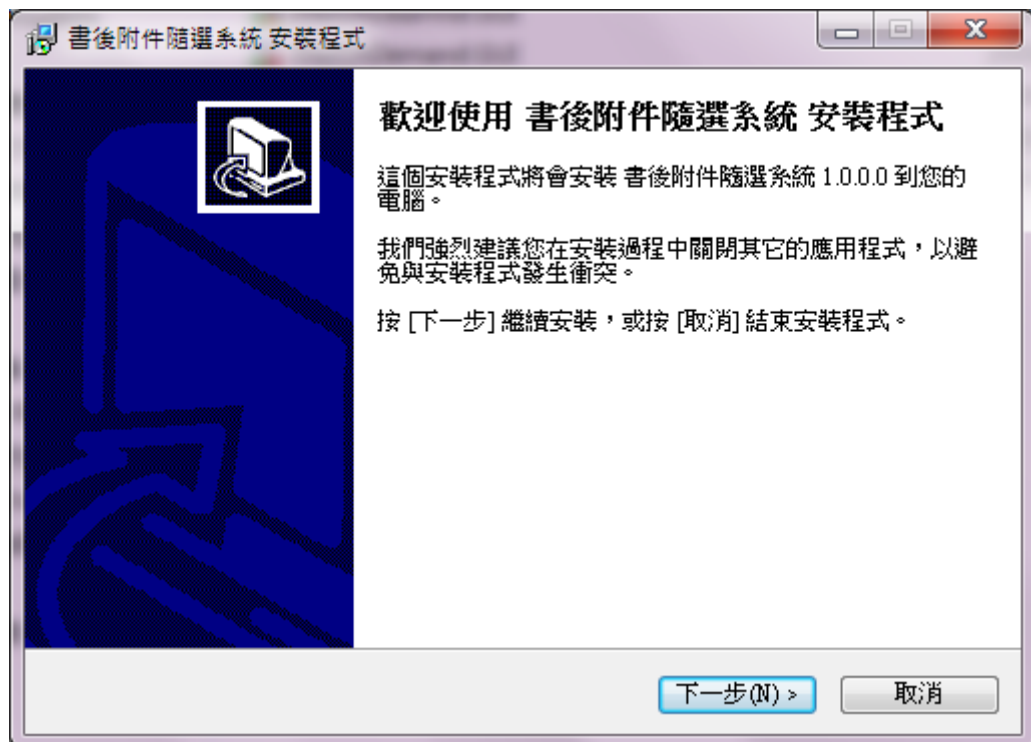


圖 3-29 GUI 安裝畫面

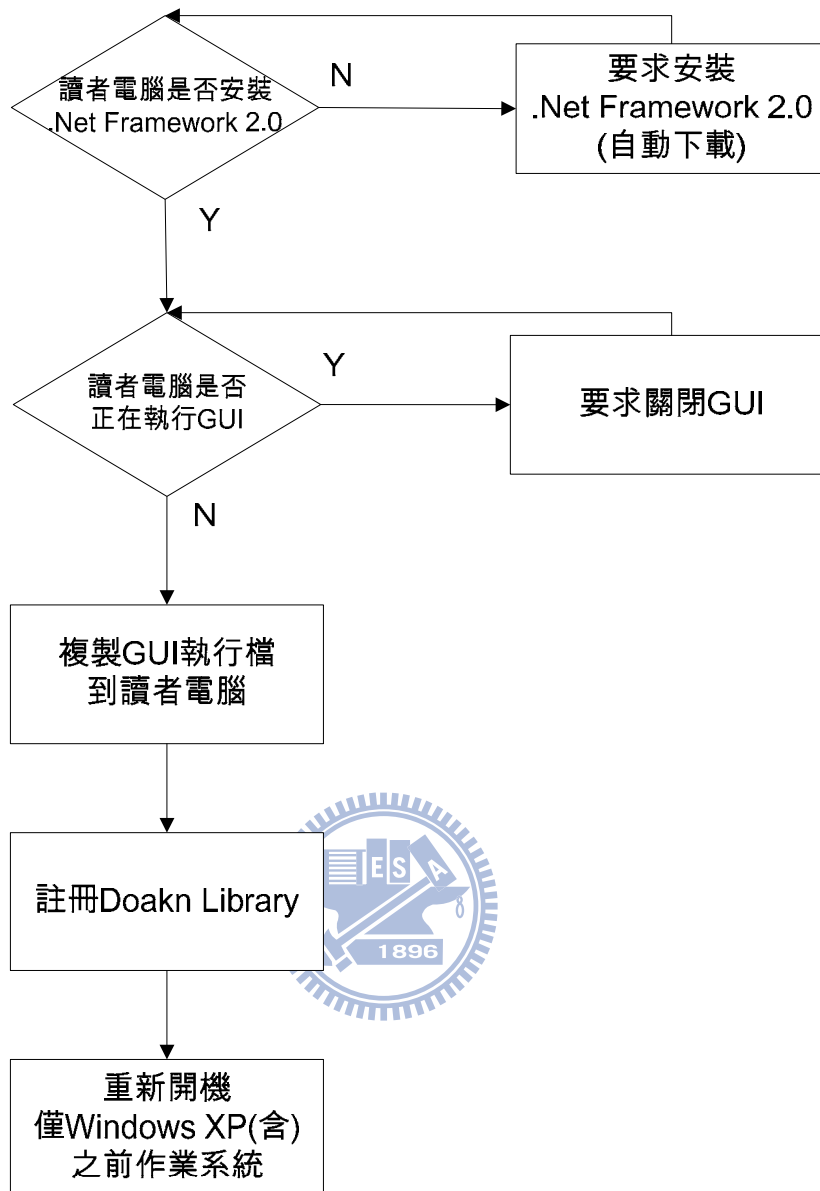


圖 3-30 GUI 安裝程式運作流程圖

四、系統評估

本章分兩個部分評估 DOD 系統。4.1 節測試系統整體表現；4.2 節以 Test Case 方法驗證圖形化使用者介面正確性。

4.1 系統整體表現

此項測試的目的為，取得書後附件隨選系統一個總合 (Over-all) 的數據，希望藉由此測試，明確地告訴圖書館與讀者，使用本系統與使用光碟機讀取書後附件時兩者讀取速度的倍數關係。此外從系統的設計方式，本研究也預期系統將有兩個特點：

1. 因為建立 HTTP 連接需要額外的開銷，讀取數量眾多的小檔案速度將會較慢。
2. 第二次讀取檔案時由於緩存機制啟動，讀取速度將會大幅提升。

測試方法為先以程式產生 1 個 100MB 的檔案(大檔案)與 10240 個 10KB 檔案(小檔案)燒錄成光碟並上傳到 DOD 系統，用光碟機與 DOD 系統各複製十次，紀錄時間。

測試伺服器端配備為：

CPU：Intel Xeon E5420, 2000 MHz (6 x 333)

RAM：4GB (3328 MB 可用)

網路卡：1000Mps

作業系統：Windows Server 2003 Service Pack 2

網頁伺服器：IIS 6

測試客戶端配備為：

CPU：Intel Core 2 Duo T5500, 1666 MHz (10 x 166)

RAM：2048MB

網路卡：1000Mps (校內使用校園網路，校外使用 10M/2M ADSL)

作業系統： Windows XP

測試結果如圖 4-1、圖 4-2、圖 4-3、圖 4-4，經由測試數據可以得到三點結論：

1. 網路為影響本系統最大原因，在校內本系統第一次讀取資料的速度已快於光碟機，最多為一點八倍；校外使用時本系統時，受限於網路速度則比光碟機慢九倍。
2. 讀取數量眾多的小檔案時速度確實降低，同樣的情況也發生在光碟機。本系統讀取速度剩六分之一；光碟機更低，僅剩九分之一。
3. 緩存機制確實有效，不管在校外還是校內，緩存啟動後讀取速度皆大幅提升。歸功於緩存機制，本系統讀取速度最多可以比光碟機快七倍。

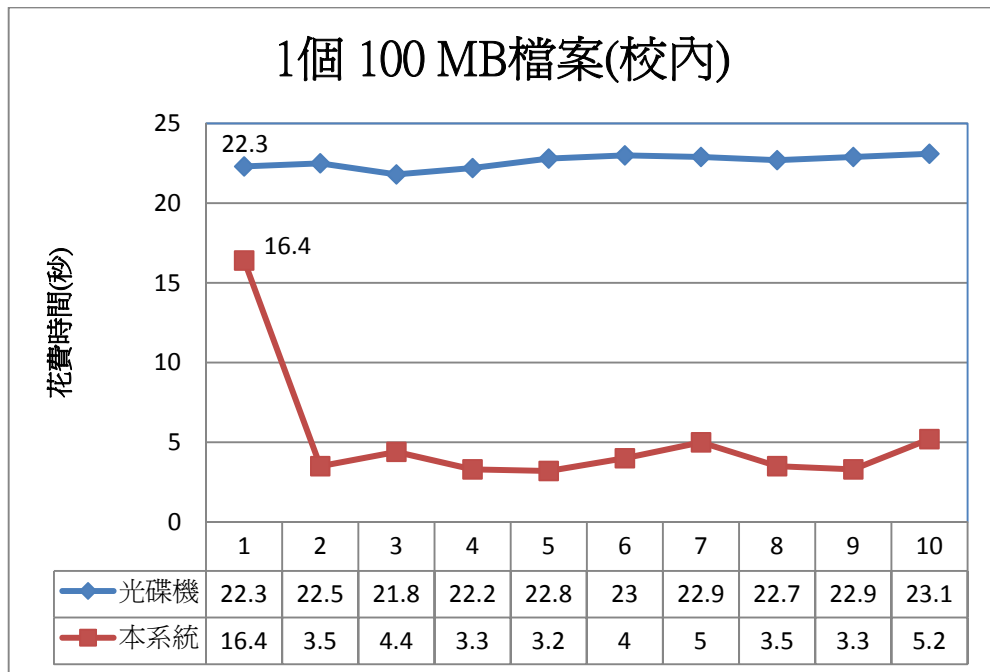


圖 4-1 校內大檔案複製結果折線圖

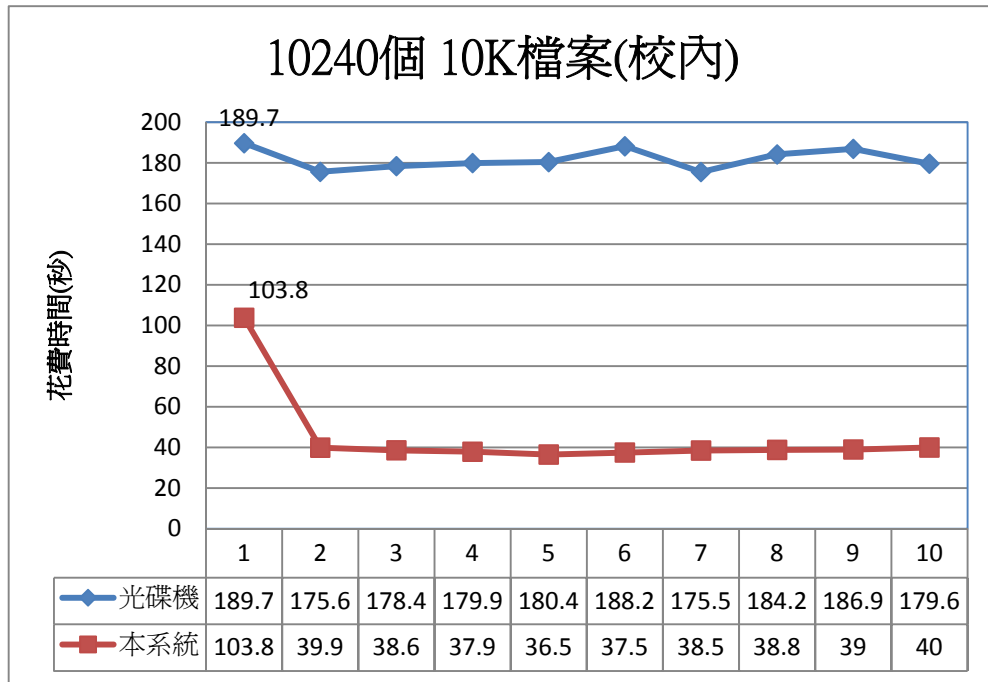


圖 4-2 校內小檔案複製結果折線圖

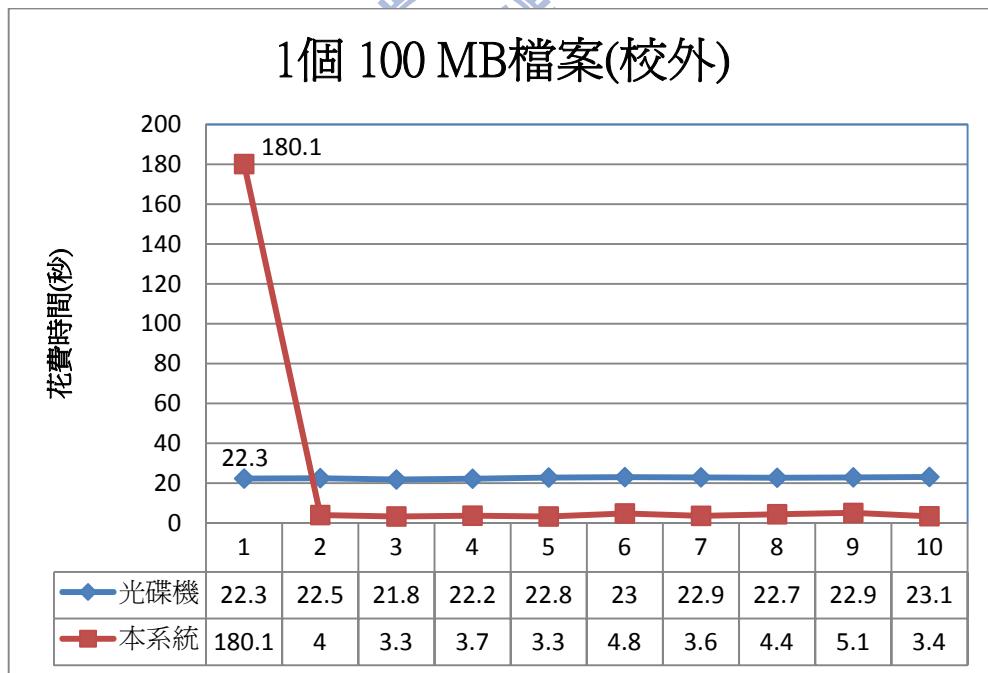


圖 4-3 校外大檔案複製結果折線圖

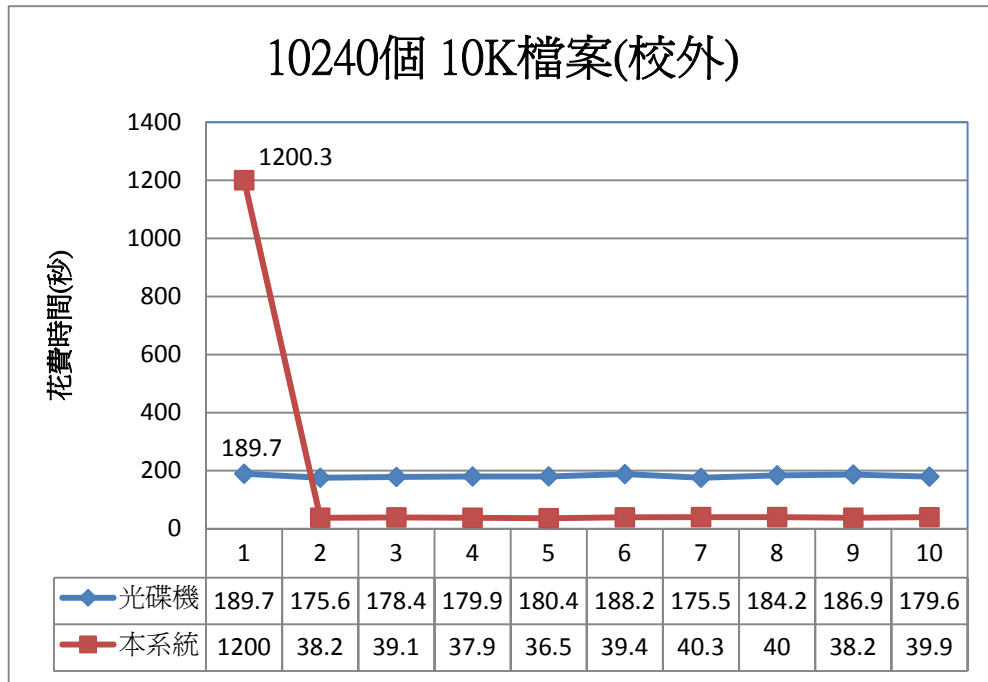


圖 4-4 校外小檔案複製結果折線圖

4.2 使用者介面 Test Case 測試



本研究採用業界常用的 Test Case 測試方法對使用者介面進行測試，先經由腦力激盪 (Brainstorming) 找出 Test Case 的項目。召開會議 (本研究為與指導教授討論) 確認 Test Case 項目，由測試人員進行測試 (本研究為請求三位讀者協助測試)，若有新規格或是發現 BUG，則改寫程式後，再度腦力激盪決定 Test Case，反覆循環。Test Case 測試進行的流程，如圖 4-5 所示。

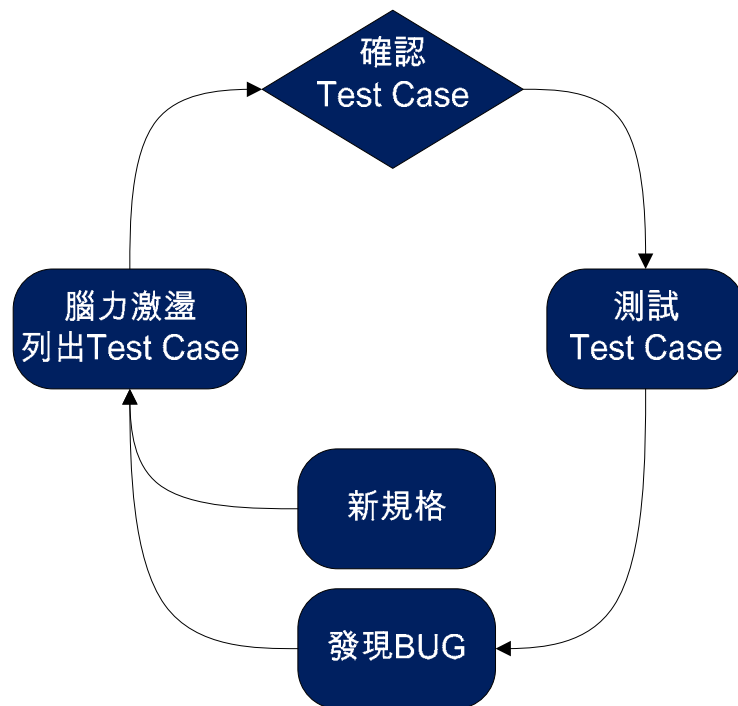


圖 4-5 Test Case 方法流程圖

整個 Test Case 測試進行的原則是，在腦力激盪列出 Test Case 項目與確定 Test Case 項目時設計者可以參加，確定 Test Case 後旋即凍結 (Freeze) 程式碼，不再對程式碼做任何修改，實際測試 Test Case 項目時則嚴禁設計者參加，以避免設計者忍不住修改程式碼。

最後本研究找出六個 Test Case 項目，如表 4-1，三位測試人員測試結果收錄於附錄一。測試伺服器與 4.2 節相同，客戶端則未限定以符合未來讀者實際使用狀況。結果為圖形化使用者介面通過所有 Test Case 項目測試。

表 4-1 Test Case 測試項目表

NO.	目的	步驟	結果
1	驗證登入機制	1. 使用正確的浩然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
2		2. 使用錯誤的浩然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
2	驗證 CAPTCHA 機制	1. 使用錯誤的浩然圖書館帳號密碼登入。 2. 輸入正確的浩然圖書館帳號密碼與驗證碼登入。	<input type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
3	比對目前借閱紀錄正確性	1. 登入浩然圖書館 WebPAC 2. 比對借閱紀錄與 GUI 顯示的借閱紀錄。	<input type="checkbox"/> 相同 <input type="checkbox"/> 不同
4	比對檔案結構正確性	1. 將實體光碟放入光碟機。 2. 比對實體光碟的全部檔案大小與虛擬磁碟的全部檔案大小。	<input type="checkbox"/> 相同 <input type="checkbox"/> 不同
5		1. 將實體光碟放入光碟機。 2. 抽測實體光碟十個檔案，尋找虛擬磁碟中相同路徑是否存在檔案。	<input type="checkbox"/> 存在 <input type="checkbox"/> 不存在
6	驗證虛擬磁碟中檔案。	1. 實際開啟十個虛擬磁碟中的檔案。	<input type="checkbox"/> 開啟正常 <input type="checkbox"/> 開啟異常
附註：			
		測試者(簽名)：	測試日期：98.12.

五、結論與建議

章魚和人是兩種無法建立任何進化關係的不同生物，而兩者的眼睛在結構與功能上卻極為相似。就算他們的眼睛很相似，進化論者也不能聲稱人和章魚具有共同祖先。

— 哈榮·雅荷雅 [41]

5.1 結論與貢獻

書後附件隨選系統與大陸地區現有的隨書光盤管理系統，如果僅看系統最後達成的效果而不論其它缺點，都解決了圖書館在書後附件管理上的問題，讀者也都是藉由網路取得書後附件。但是在所謂的「進化」過程，即本研究觀察圖書館遭遇的問題後，查閱相關文獻、設計系統架構最後實作整個系統的研究過程，就像人跟章魚一樣有著完全不同的進化過程。書後附件隨選系統並非現有隨書光盤管理系統的改進或更進一步研究，而是為了滿足研究動機與目的，巧妙運用 API 與 FUSE 技術，模擬讀者借出書後附件光碟放入光碟機讀取的實際狀況，並達成不須下載整個光碟映像檔、可控管書後附件同時被存取數量等優點。此乃因整個書後附件隨選系統以 DOD API 為核心，以提供書後附件「服務」而非網站為起點，是一種做對的事情 (Do right things) 之創見；而隨書光盤管理系統一開始已被下載光碟映像檔的想法限制，提供更好的檢索介面，更精美的網站畫面，只是一種將事情做對 (Do things right) 的努力。

書後附件隨選系統不需借閱實體光碟給讀者，而是在讀者登入後，提供可使用書後附件列表；在讀者選擇欲使用的書後附件後，於作業系統虛擬一個內含書後附件完整檔案內容的磁碟機，整個使用過程中並未涉及實體光碟，所以如表 1-1 因書後附件實體光碟的借出、歸還、陳列與管理而衍生的問題將不復存在。

書後附件隨選系統藉由 NCTUAdapter 連結圖書館自動化系統，由系統自動搜尋圖書館自動化系統中的借閱紀錄與借閱期限，提供相對應書後附件。圖書館除了不須為此系統於書籍借出或歸還進行任何額外動作外，更可簡化目前作業流程，不再需要館員協助讀者進行找片與取片等等作業，因此提升圖書館的效率。

書後附件隨選系統可模擬一本書籍對應一片或數片書後附件的特性，例如：圖書館購買兩本相同書籍，第一本借給讀者 A，第二本借給讀者 B。就只有讀者 A 與 B 可以取得書後附件。並沒有第三本可以借給其他讀者，因此也不會有第三份書後附件可被取得。此做法與目前實體光碟借閱狀況類似，本研究希望藉此控管方式能解除書商疑慮，更容易與書商協商著作權問題。

最後在書後附件隨選系統的圖形化使用者介面部分，FUSE 技術協助本研究將檔案總管與使用者介面無縫接成，讓讀者繼續使用原本書後附件的使用者介面，不需讀者額外學習新使用者介面的操作方式，還體貼地以 Inno Setup 製作安裝檔，一步一步引導使用者安裝減少讀者學習操作使用者介面的時間與困難度，以減少推廣系統時的阻力。

總結本研究最主要有四點貢獻：

1. 本研究為國內第一個提出圖書館書後附件面臨問題與解決方案的研究。
2. 提供一套開放的 DOD API，可匯集其它設計者的創意，發展出更多的應用。
3. 以 FUSE 技術將使用者介面與檔案總管無縫接成，免去讀者學習操作使用者介面的困擾，減少系統推廣時阻力。
4. 達成書後附件取得數量上的控管，圖書館購買的紙本書籍數量，即為允許書後附件被取得的數量。

5.2 未來研究方向

本研究認為未來可以從兩個方向進行深入研究：

1. 橫向拓展圖書館服務項目

觀察目前網路應用發展的趨勢，越來越多公司將本身的服務以 API 的方式提供給使用者，並獲得成功。圖書館目前已有許多項協定²²用於館與館之間的溝通。未來可研究是否可將其它資源藉由 API 提供給讀者。

2. 進行館際合作

研究如何解決各館讀者帳號與條碼號不同的問題。將 DOD 系統運用於館際合作，免除需郵政系統運送實體書後附件的成本，進行館際合作還可減輕館員上傳書後附件檔案的負擔。



²² MARC、Z39.50、OAI、OpenURL、DOI 等等

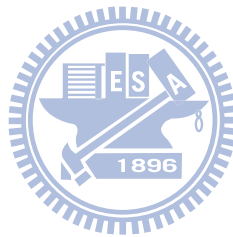
參考文獻

- [1] 石美玉，「光碟唯讀體(CD-ROM)在臺灣各大學圖書館之應用」，教育資料與圖書館學，第 29 卷第 3 期，331~344 頁，1991 年。
- [2] Mercado H., “Library Instruction and Online Database Searching”, Reference Services Review, Vol. 27, No. 3, pp. 259-265, 1999.
- [3] 陳瑞順等，「經由校園網路存取圖書館光碟資料庫之研究」，教育資料與圖書館學，第 29 卷第 4 期，376~387 頁，1992 年。
- [4] 龔舒野，「隨書光盤的著作權問題初探」，南通航運職業技術學院學報，第 6 卷第 3 期，121~122 頁，2007 年 9 月。
- [5] 馬飛，「高校圖書館隨書光盤科學化管理模式探討」，北京理工大學學報社會科學版，第 9 卷第 3 期，118~120 頁，2007 年 6 月。
- [6] 白永革，「隨書光盤管理與利用的一種新模式」，圖書館理論與實踐，第 2005 卷第 5 期，117~119 頁，2005 年 10 月。
- [7] 李新宇，「隨書光盤的著錄與管理模式研究」，安徽教育學院學報，第 24 卷第 3 期，120 ~122 頁，2006 年 5 月。
- [8] 於源，周慶林，「ISBN 作為隨書光盤館際共用橋引的研究」，大連交通大學學報，第 30 卷第 1 期，107 ~109 頁，2009 年 2 月。
- [9] 沈靜萍，「一種與圖書管理系統整合的隨書光盤管理系統」，圖書館工作與研究，第 2009 卷第 8 期，38 -41 頁，2009 年 8 月。
- [10] 杜治波，「隨書光盤工作標準化探析」，雲南財貿學院學報社會科學版，第 22 卷第 6 期，134~135 頁，2007 年 12 月。
- [11] 徐勇進，「隨書光盤的深層開發利用新探」，情報探索，第 2008 卷第 2 期，110~111 頁，2008 年 2 月。

- [12] 周月蓮，李波，「CNMARC856 欄位與隨書電子文獻檢索系統設計」，圖書館工作與研究，第 2007 卷第 2 期，55~57 頁，2007 年 3 月。
- [13] 李禾，馮濤，「基於 ASP.NET2.0 平台的隨書光盤動態發佈模式」，圖書館工作與研究，第 2009 卷第 2 期，42~45 頁，2009 年 2 月。
- [14] 蔡立，蘇建英，「基於 ASP.NET 的隨書光盤管理系統的設計與實現」，農業網絡信息，第 2008 卷第 8 期，48~49 頁，2008 年 8 月。
- [15] 譚榕，「開源軟件在隨書光盤系統中的應用」，現代情報，第 29 卷第 7 期，151~156 頁，2009 年 7 月。
- [16] Jin Y. et al., Understanding Mashup Development. Internet Computing, Vol.12, No. 5, p. 44-52, 2008.
- [17] Benslimane D. et al., Services Mashups: The New Generation of Web Applications, Internet Computing, Vol.12, No. 5, p.13-15, 2008.
- [18] Fielding R.T., "Architectural Styles and The Design of Network-based Software Architectures", University of California, Irvine, pp.162, 2000.
- [19] Richardson L. and S. Ruby, RESTful Web Services, O'Reilly, 2007.
- [20] Flanders J., RESTful.NET: Build and Consume RESTful Web Services with .NET 3.5, O'Reilly, 2008.
- [21] Tilkov, A Brief Introduction to REST, <http://www.infoq.com/articles/rest-introduction> , Available at 2009/12/1.
- [22] Pautasso C. et al., "Restful Web Services vs Big Web Services: Making The Right Architectural Decision", Proceeding of the 17th international conference on World Wide Web, pp. 805-814, Beijing, China, 2008.
- [23] EFOAGUI, An Investigation into The Differences between RESTful and SOAP Web Services.
http://www.mindworksworldwide.com/docs/EFOAGUI_CHINEDU_For_public.pdf, Available at 2009/12/1.

- [24] Gamma E., et al., Design patterns: elements of reusable object-oriented software, Addison-Wesley, 1995.
- [25] Danny B. and Lange Y.N., “Interactive Visualization of Design Patterns Can Help in Framework Understanding”, Proceedings of the tenth annual conference on Object-oriented programming systems languages and applications, pp. 342-357, Austin, Texas, United States, 1995.
- [26] Broecke J.A.v.d. and J.O. Coplien, Using Design Patterns to Build a Framework for Multimedia Networking, Cambridge University Press, 2001.
- [27] Schmidt D.C., “Using design patterns to develop reusable object-oriented communication software”, Communications of the ACM, Vol.38, No.10, pp.65-74, 1995.
- [28] Freeman E., et al., Head First Design Patterns, O' Reilly, 2004.
- [29] Bishop J., C# 3.0 Design Patterns, O' Reilly, 2007.
- [30] Hoskins M.E., SSHFS: Super Easy File Access over SSH, Linux Journal, Vol.2006, No 146, p. 4, 2006.
- [31] Holupirek A. and M.H. Scholl, Implementing Filesystems by Tree-aware DBMSs. Proceedings of VLDB Endow. Vol.1, No. 2, p. 1623-1630, 2008.
- [32] Richard, GmailFS,
<http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html>, Available at 2009/12/1.
- [33] Kiselyov O., “A Network File System over HTTP: Remote Access and Modification of Files and Files”, Proceedings of the annual conference on USENIX Annual Technical Conference, pp.31-31, California, 1999.
- [34] Martin B., “The World Is A Libferris Filesystem”, Linux Journal, Vol.2006, No.146, p. 4, 2006.
- [35] Luis von Ahn, et al., Completely Automatic Public Turing Test to Tell Computers and Humans Apart, <http://www.captcha.net>, Available at 2009/12/1.


- [36] Yan J. and A.S.E. Ahmad, Usability of CAPTCHAs or Usability Issues in CAPTCHA Design, Proceedings of the 4th symposium on Usable privacy and security, p. 44-52, Pennsylvania, 2008.
- [37] Hiroki, Dokan Library, <http://dokan-dev.net/en/>, Available at 2009/12/1.
- [38] Syates, UrlRewriter.NET, <http://urlrewriter.net>, Available at 2009/12/1.
- [39] Kosse T., FileZilla Server, <http://filezilla-project.org/index.php>, Available at 2009/12/1.
- [40] Russell J., Inno Setup, <http://www.jrsoftware.org/isinfo.php>, Available at 2009/12/1.
- [41] Harun Yahya, Evolution Deceit, <http://www.evolutiondeceit.com/cn/index.html>, Available at 2009/12/1.



附錄一 Test Case 測試結果

NO.	目的	步驟	結果
1	驗證登入機制	1. 使用正確的浩然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
2		2. 使用錯誤的浩然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input checked="" type="checkbox"/> 不可登入
2	驗證 CAPTCHA 機制	1. 使用錯誤的浩然圖書館帳號密碼登入。 2. 輸入正確的浩然圖書館帳號密碼與驗證碼登入。	<input checked="" type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
3	比對目前借閱紀錄正確性	1. 登入浩然圖書館 WebPAC 2. 比對借閱紀錄與 GUI 顯示的借閱紀錄。	<input type="checkbox"/> 相同 <input type="checkbox"/> 不同
4	比對檔案結構正確性	1. 將實體光碟放入光碟機。 2. 比對實體光碟的全部檔案大小與虛擬磁碟的全部檔案大小。	<input checked="" type="checkbox"/> 相同 <input type="checkbox"/> 不同
5		1. 將實體光碟放入光碟機。 2. 抽測實體光碟十個檔案，尋找虛擬磁碟中相同路徑是否存在檔案。	<input checked="" type="checkbox"/> 存在 <input type="checkbox"/> 不存在
6	驗證虛擬磁碟中檔案。	1. 實際開啟十個虛擬磁碟中的檔案。	<input checked="" type="checkbox"/> 開啟正常 <input type="checkbox"/> 開啟異常
附註：			
測試者(簽名): <u>吳曉偉</u> 測試日期: 98.12.			

NO.	目的	步驟	結果
1	驗證登入機制	1. 使用正確的活然圖書館帳號密碼登入。	<input checked="" type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
2		2. 使用錯誤的活然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input checked="" type="checkbox"/> 不可登入
2	驗證 CAPTCHA 機制	1. 使用錯誤的活然圖書館帳號密碼登入。 2. 輸入正確的活然圖書館帳號密碼與驗證碼登入。	<input checked="" type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
3	比對三宗借閱紀錄正確性	1. 登入活然圖書館 WebPAC 2. 比對借閱紀錄與 GUI 顯示的借閱紀錄。	<input checked="" type="checkbox"/> 相同 <input type="checkbox"/> 不同
4	比對檔案格式正確性	1. 將實體光碟放入光碟機。 2. 比對實體光碟的全部檔案大小與虛擬磁碟的全部檔案大小。	<input checked="" type="checkbox"/> 相同 <input type="checkbox"/> 不同
5		1. 將實體光碟放入光碟機。 2. 比對實體光碟十個檔案，尋找虛擬磁碟中的同路徑是否存在檔案。	<input checked="" type="checkbox"/> 存在 <input type="checkbox"/> 不存在
6	驗證虛擬磁碟中檔案。	1. 實際開讀十個虛擬磁碟中的檔案。	<input checked="" type="checkbox"/> 開啟正常 <input type="checkbox"/> 開啟異常
附註：			
測試者(簽名)： 鄧新地 測試日期：98.12.1			

NO.	目的	步驟	結果
1	驗證登入機制	1. 使用正確的浩然圖書館帳號密碼登入。	<input checked="" type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
2		2. 使用錯誤的浩然圖書館帳號密碼登入。	<input type="checkbox"/> 可登入 <input checked="" type="checkbox"/> 不可登入
3	驗證 CAPTCHA 機制	1. 使用錯誤的浩然圖書館帳號密碼登入。 2. 輸入正確的浩然圖書館帳號密碼與驗證碼登入。	<input checked="" type="checkbox"/> 可登入 <input type="checkbox"/> 不可登入
3	比對目前借閱紀錄正確性	1. 登入浩然圖書館 WebPAC 2. 比對借閱紀錄與 GUI 顯示的借閱紀錄。	<input checked="" type="checkbox"/> 相同 <input type="checkbox"/> 不同
4	比對檔案結構正確性	1. 將實體光碟放入光碟機。 2. 比對實體光碟的封面檔案大小與虛擬磁碟的全部檔案大小。	<input checked="" type="checkbox"/> 相同 <input type="checkbox"/> 不同
5		1. 將實體光碟放入光碟機。 2. 遍測實體光碟十個檔案，尋找虛擬磁碟中相同路徑是否存在檔案。	<input checked="" type="checkbox"/> 存在 <input type="checkbox"/> 不存在
6	驗證虛擬磁碟中檔案。	1. 實際開出十個虛擬磁碟中的檔案。	<input checked="" type="checkbox"/> 開啟正常 <input type="checkbox"/> 開啟異常
附件： <div style="text-align: right; margin-top: 20px;"> 測試者(簽名):  測試日期: 98.12.2 </div>			