國立交通大學

電機學院與資訊學院　數位圖書資訊學程

碩士論文

RSS 技術應用於圖書館專題選粹服務之研究

—以個人化電子期刊訂閱為例

Application of RSS to Library Selective Dissemination of Information Service
-- A Study of Personalized Subscription of Electronic Journal

研 究 生：簡燕華

指導教授：黃明居　博士

中華民國九十五年二月

RSS 技術應用於圖書館專題選粹服務之研究

—以個人化電子期刊訂閱為例

Application of RSS to Library Selective Dissemination of Information Service
-- A Study of Personalized Subscription of Electronic Journal

研 究 生：簡燕華　　Student ：Yen-Hua Chien

指導教授：黃明居博士　　Advisor：Dr. Ming-Jiu Hwang

國 立 交 通 大 學

數位圖書資訊學程

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

In partial Fulfillment of the Requirements

for the Degree of

Master

in

Digital Library

February 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年二月

RSS 技術應用於圖書館專題選粹服務之研究
—以個人化電子期刊訂閱為例

學生：簡燕華　　指導教授：黃明居博士

電機學院與資訊學院　數位圖書資訊學程

國立交通大學電機學院與資訊學院　數位圖書資訊學程碩士班

摘　要

圖書館的專題選粹服務，是圖書館針對讀者的資訊需求，主動提供的服務之一。在自動化普及之後，多數電子期刊資料庫以電子郵件方式，定期寄送符合使用者需求之新資源。電子郵件在便利之餘，其質與量卻受到大量病毒和廣告郵件所影響，造成閱覽新資訊的困擾。

本研究利用近年來新興的資訊推播工具-RSS（Really Simple Syndication），在期刊資料庫和讀者之間，建置一個訂閱專題選粹服務的新平台-MJFB（My Journal Filter Broker）。MJFB 系統是採 MyLibrary 的設計理念，利用個人化的技術與方法，使用者能依個人背景與特性，精確定義期刊訂閱的需求，並在最後過濾搜尋到的期刊文章數量，減少不必要的新資訊；並且不需再透過其他軟體，在系統的同一網頁頁面中即可閱覽到所訂定的期刊及文章內容。

研究結果顯示，利用 RSS 傳遞新資訊可以避免現行電子郵件的問題；同時，靈活運用 XML tag 的彈性和 OPML 檔案規格，在訂閱和閱覽新資訊時可以更加便利，符合個人需求，很適合在數位化圖書館中導入此項新的專題選粹服務。

關鍵字：專題選粹服務、RSS、個人化、數位圖書館

Application of RSS to Library Selective Dissemination of Information Service
-- A Study of Personalized Subscription of Electronic Journal

Student：Yen-Hua Chien Advisor：Dr.Ming-Jiu Hwang

Degree Program of Electrical Engineering Computer Science

National Chiao Tung University

Abstract

Selective Dissemination of Information Service is one of the services provided actively by library and focuses on patron's personal information needs. With the automation of library services, most electronic journal system periodically send new information matching patron's needs by e-mail. E-mail is convenient nowadays. However, the great spread of viruses and junk mails effects the quality of e-mail and increases the amount of unessential e-mails。

This research uses RSS as an information dissemination pushing approach. We build a new web system–MJFB for SDI between electronic journal databases and patrons. MJFB system adopts the concepts of 「MyLibrary」and 「Personalization」 to be developed, and patrons can precisely subscribe their own electronic journals and articles based on backgrounds and features. Lastly, MJFB system filters articles searched by keywords to eliminate unnecessary information. Also, those subscribed new information can be viewed on the same web page as MJFB system without using other application.

The result of this research shows that RSS as an information delivering tool can avoid those problems occurring in e-mail. Meanwhile, using the flexibility of XML and OPML file can subscribe and browse new information much more conveniently and conforms to personal needs. In conclusion, MJFB can be a new aspect of SDI in digital library.

Keywords：SDI, RSS, Personalization, Digital Library

誌　　謝

當完成編排這篇論文時，心中真是百感交集。想起多年前大學畢業時，並沒有繼續唸書的打算；卻在工作多年後的機緣裡，認識圖資這個領域，興起重回學校的念頭，也因此特別珍惜這段重當學生的日子。二年多來，新竹台北間的往返來回雖然辛苦，但在同學和老師們的陪伴中，累積了許多回憶。

論文的撰寫實在是項自我挑戰的過程，也是耐力、時間規劃、生活安排的考驗；能夠完成，實要感謝許多人的幫忙。

感謝組上老師們對專業知識的教授，以及幫助學生的熱忱。懷念楊維邦前館長上課時溫煦的胸懷，希望有朝一日能親見「電腦與人生」一書的出版。感謝柯皓仁老師的教導，上課時的聲量和熱忱讓我始終精神飽滿。特別感謝指導教授黃明居老師，在撰寫論文這段期間的悉心幫助和指導，讓我一窺學術研究的初貌；尤其在離開職場之後，在時間安排上的指引和規劃，讓我得以順利完成學業。

組上的學長姐、同學、學弟妹們以及圖書館計畫室的人員，謝謝你們大家的陪伴、鼓勵、和幫忙。因為大家的包容、打氣加油、彼此對課業上的討論和分享，使我面對論文時，不致感到孤單；影印室、館合人員，謝謝你們的幫忙，讓我得以順利取得所需文件。

最後，由衷感謝我的家人和父母親，能支持我暫時離開職場完成學業，雖然有時因忙碌而焦頭爛額，仍提供溫暖的照顧，和安心休息的窩。謹將此篇論文獻給他們。

簡燕華　謹識

於交通大學數位圖書資訊組

九十五年二月

目　　錄

圖 目 錄

表 目 錄

第一章　緒論

1-1　研究動機與目的

圖書館除了提供豐富的圖書館藏外，讓知識工作者掌握到適當所需資訊也是責任之一。一個好的現代化圖書館，不應只是靜態的典藏知識，更應該動態的散播有用新知。而 Metcalfe 及 Paul 則指出，有用的資訊決定於人們的興趣焦點[1]。

長久以來，圖書館的專題選粹服務（Selective Dissemination of Information, SDI）針對讀者個別興趣，選擇最新資訊，以主動積極的方式，定時提供資訊服務。目的在節省讀者檢索資料的時間，加強掌握館藏的流通與利用[2]。

專題選粹服務是圖書館服務化被動為主動的一個最明顯例子和作法。從早期的手動人工方式，到現在圖書館自動化普及後，許多資料庫廠商免費提供該項服務。現行資料庫的 SDI 服務方式，有些要讀者先註冊基本資料，有些不需要；進入資料庫後，依各家廠商設計界面不同，勾選有興趣的資料類型或檢索條件，並將這些設定儲存起來。等到有符合勾選條件的新資料時，多以電子郵件方式通知讀者。

這樣的方式已是數位圖書館中個人服務的一種，即利用資訊系統提供針對個人需求的服務。如此，的確能幫讀者掌握新資訊及節省追蹤新資料的時間，帶來不少便利；另一方面的困擾，則是越來越多讀者開始抱怨這些新知通報郵件佔去信箱的大半空間，最後這些當初訂閱時深受期待的電子郵件落入被整批選取、刪除的下場，造成「過多的資訊等於沒有資訊」的困境[3]。

再者，近年來電腦病毒盛行，電子郵件也是散播管道之一，造成不敢輕易開啟來路不明郵件，影響電子郵件的品質；而垃圾郵件更是成為世界各國都想解決的問題之一。因此，電子郵件雖已成為現代人不可或缺的聯絡方式之一，但要在一堆電子郵件中區隔出真正所要的資訊，仍有困難和不便之處。

本研究動機之所在，即是欲尋求另一種專題選粹服務的新方式，保留原本之優點，又解決上述之問題。在此同時，RSS 這項被列為 2005 年 10 大電子商務發展趨勢第 2 名的新資訊技術正逐漸蓬勃興起與發展[4]，簡單易用的特性和傳遞新

資訊的功能，符合專題選粹服務的特性，且目前沒有病毒與垃圾郵件的問題。

也正因其發展迅速，RSS 的製作與訂閱數量日漸增多。如何找到符合個人需求的製作內容，控制訂閱數量，以達真正有效閱覽、獲取新資訊，又是進一步的課題。

因此，本研究期望以 RSS 技術替圖書館的專題選粹服務導入一項新方式，並提出明確的方式讓使用者訂定個人資訊需求，以達專題選粹服務原本個人化數位圖書館精神之實現。

1-2 研究內容

基於上述研究背景與動機，利用 RSS 解決現行圖書館 SDI 服務的問題，是本研究的重點目的所在。RSS 是一項新的資訊推播 (Push) 工具，主要用以傳遞更新訊息，也是本研究用以取代 SDI 電子郵件服務的主要意圖。

本研究企圖在以下內容的討論過程中，尋找出較佳的 SDI 服務方案：

1. 探討 SDI 服務的歷史演進，從以往人工的作業方式到現今的電子化服務，瞭解作業方式演變的優缺點，期能找出效率較佳的服務方式。
2. 分析個人化服務的精神和方法，以及在數位化圖書館的應用。
3. 研究 RSS 的技術沿革，包括各版本的演進和規格比較，現行的應用狀況和問題，以及 RSS 的相關名詞介紹。
4. 建置一個 SDI 電子期刊訂閱平台系統，提供 RSS 訂閱服務，以取代現行電子郵件的通報方式。

1-3 研究範圍與限制

本研究範圍在於以圖書館的電子資料庫為基礎，再以電子期刊為主要探討對象。現今電子期刊部份沒有提供 SDI 服務，部分提供免費的 SDI 服務；截至本研究時間，SDI 通報服務仍以電子郵件方式為多數，僅有少數圖書館已經開始使用 RSS 技術。

考量取得電子資源的便利性，本研究以交通大學圖書館電子期刊資源為基

礎。在期刊 RSS feed 取得及內容搜尋方面，以 Ingenta 期刊資料庫提供的搜尋介面為例，因其期刊涵蓋學科領域廣泛，且多數期刊已提供免費 RSS 訂閱的功能。

系統實作的期刊來源則是 2006 年交大圖書館西文電子期刊清單。依學院單位整理過後，含在 Ingenta 資料庫內，並且有 RSS 訂閱服務，共有 338 筆期刊。

1-4 研究方法與進行步驟

本研究在確定研究問題與方向後，分二方面進行。一方面開始著手收集相關文獻，主要分為三個層面：包括專題選粹服務、個人化服務以及 RSS 技術之相關規格和問題討論；另一方面則深入瞭解現行圖書館電子期刊專題選粹服務的系統狀況，觀察問題。

之後，嘗試幾種現行常用的開發工具和程式語言。在考慮使用性和普遍性之後，決定在 Windows 平台上，以 PHP 程式語言和 MySQL 資料庫軟體為本研究的開發工具。

接著，依以往專題選粹服務的精神和模式，以個人化服務的精神，使用 RSS 技術擬出解決問題的改良架構，建立系統雛型。最後進行系統驗證，提出本研究的結論和未來研究的建議。

本研究將 RSS 的應用延伸到圖書館電子資源，流程如圖 1-1 所示：

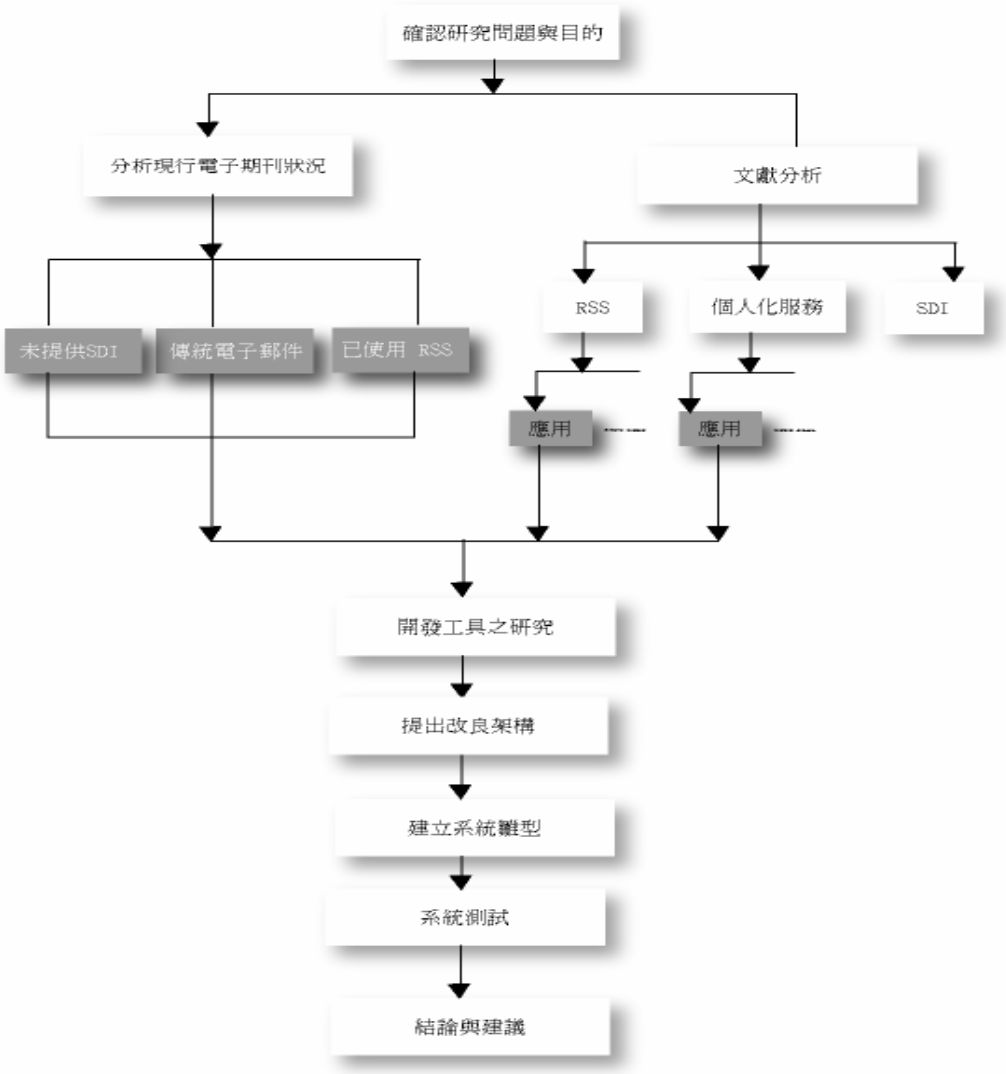

圖 1-1　研究流程

1-5　論文架構

本研究共分六章，各章重點簡述如下：

第一章、緒論

本章先說明此論文之研究動機與目的、研究內容、研究範圍與限制、研究方法與進行步驟、以及論文架構。

第二章、文獻探討

此部分探討 SDI 服務的歷史演進與改變,個人化服務和數位圖書館之介紹，以及 RSS 技術的簡介與應用狀況。

第三章、RSS 技術與 SDI 應用

本章探討 RSS 詳細的技術規格，以及和電子郵件之不同處。此外，觀察現今圖書館在 SDI 中使用 RSS 的狀況，整理出和使用電子郵件不同之處。。

第四章、MJFB 系統設計

在此提出本研究系統之架構，使用個人化服務方法，設計出期刊、以及以關鍵字搜尋新期刊內容二大主軸的 RSS Feed 訂閱。並且說明系統中動態產生 RSS Feed 和 OPML 檔案的過程。

第五章、系統實作與驗證

介紹系統實作環境和工具，展示系統執行結果，以及對系統設計做初步的結果驗證。

第六章、結論與未來研究建議

為本研究之總結，說明相關成果與貢獻，並提出未來可繼續研究之課題。

第二章　文獻探討

本章就研究內容所需文獻做一分析探討，共分為專題選粹服務、數位圖書館個人化服務，與 RSS 相關文獻等三大部分，其整體分佈如圖 2-1 所示。

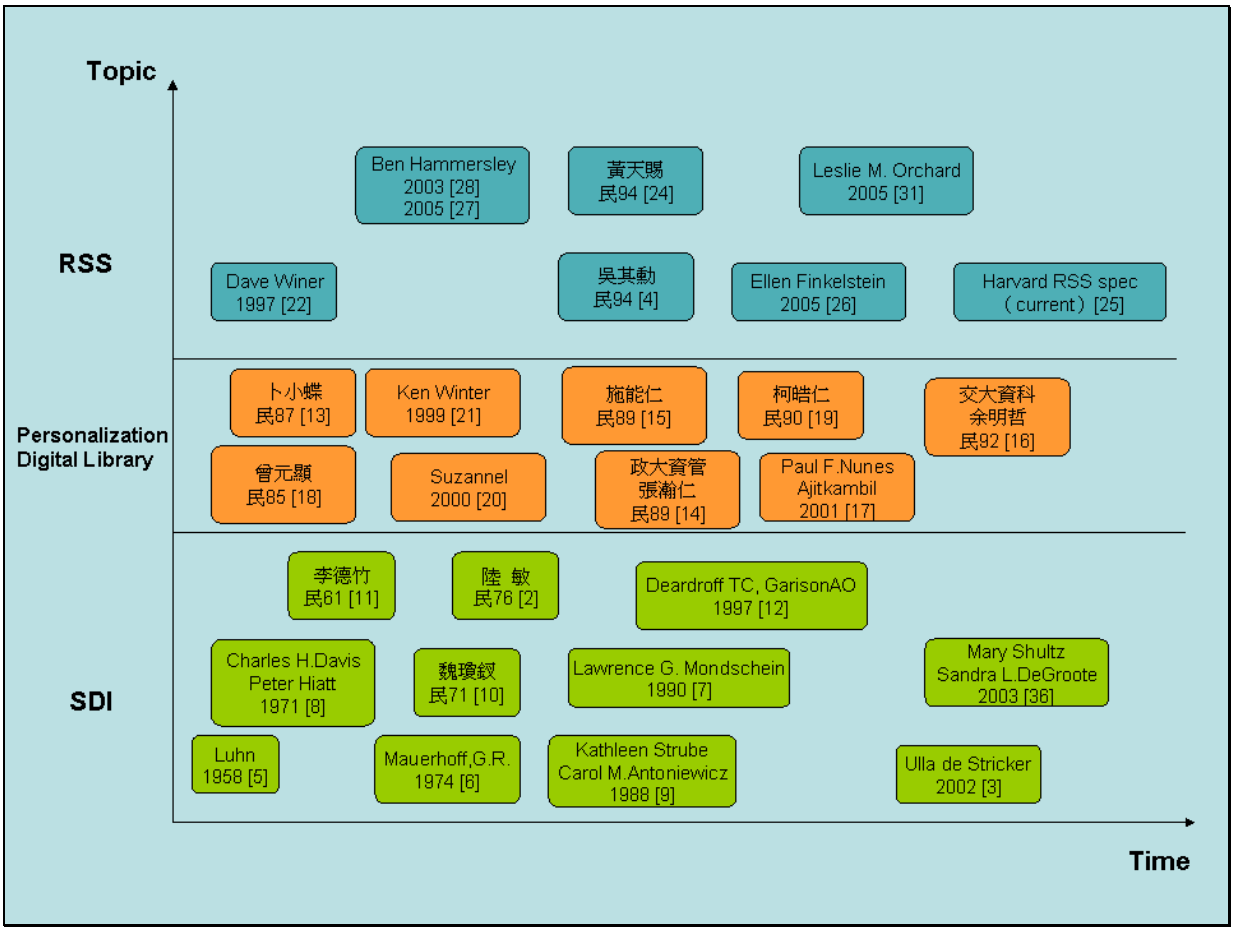

圖 2-1　文獻整理

2-1　專題選粹服務（Selective Dissemination of Information, SDI）

2-1-1　定義、目的與作業模式

1.定義與目的

陸敏清楚定義出：專題選粹服務是針對讀者個別興趣，選擇最新資訊，以主動積極的方式，定時提供資訊服務[2]。

Luhn 在 1961 年曾提出，SDI 是一項在機器（電腦）輔助下，傳播新資訊的服務，目的是要使組織內的每位成員能獲取新的資料，而組織內各項資料的使用率能達到最高點[5]。

Housman 和 Kaskela 在 1970 年說明，SDI 系統的目的並不是研究人員在緊急時，去找出有用的資料；而是持續通知研究人員在專業領域上的新資料，讓研究人員得以跟上最新發展。自動化的 SDI 並沒有完全取代獲取資訊的習慣，也沒有保證篩選出所有關於使用者核心興趣的文獻資料，但是 SDI 可以在一般搜尋能力和意願的範圍之外，幫助研究人員定期更新資料[6]。

更廣義的解釋是，不論採用人工或自動的方式，只要針對個別使用者，提供現況通知服務，選擇有關研究興趣的新資料並加以傳遞，就算是 SDI 服務。

2.作業模式

無論是人工手動或電子自動化系統，在篩選資料時，專題選粹服務之一般模式，如圖 2-2。

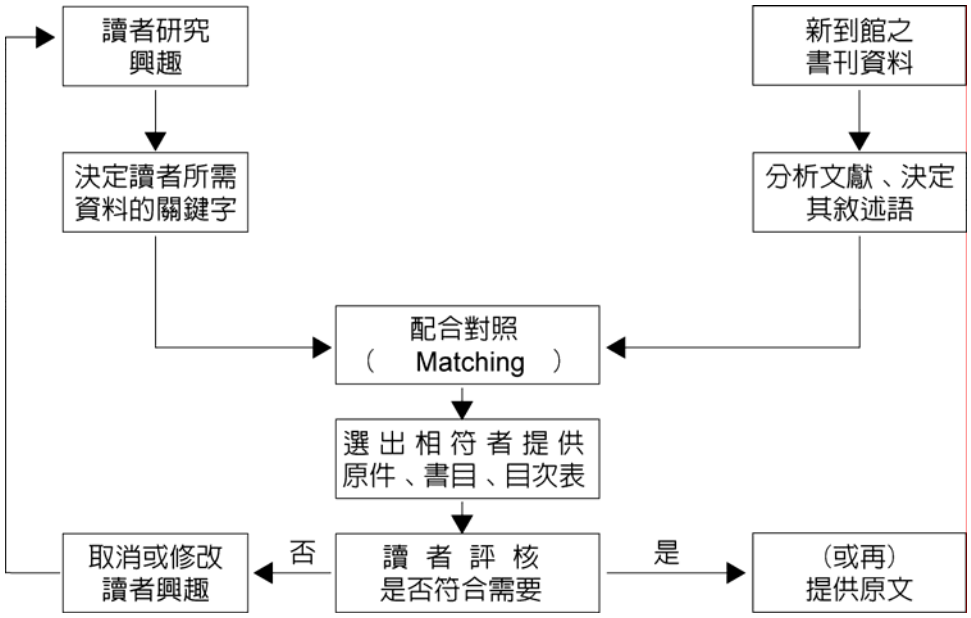

圖 2-2　專題選粹服務的一般模式[2]

圖 2-2 作業流程如下：

（一）讀者研究興趣與資料蒐集

- 讀者興趣研究

專題選粹服務的對象可以是個人、團體、或學科主題。常用的方法有問卷法、面談法、閱讀服務對象最近出版的文章、參觀實驗室等。工作人員根據調查結果加以整理，再製成讀者興趣檔，並決定讀者研究的關鍵字。現行電子化系統，通常由讀者自行勾選建立。

此作業目的在瞭解讀者個別興趣，建立興趣檔資料，以供往後比對使用。

- 資料蒐集

除了圖書館藏書、還有西文文獻、書目資料、書刊及期刊的目次表、以及一些報導性的消息短文。現在網路資訊發達，網路上的資料亦不少，平日可多觀察、累積一些資訊。

此作業主要在累積、蒐集多方面的新資訊，做為往後新資訊比對、

篩選的來源。

（二）資料篩選對照

將讀者的興趣檔、關鍵字和所蒐集的資料配合比對，選出相符部分的資料，依讀者需求，提供書目、目次表或全文。若有不相符的部分，則依狀況修改原先的興趣檔和關鍵字。現行的電子化系統，每家廠商有不同的演算法設計配對，以找出相符合的資料。

（三）讀者資料檔備份

專題選粹服務所提供的資料，應留存一份，載明資料標題、篇名、作者、資料來源、序列號及讀者姓名。以供後續服務，編印專題書目，及給其他讀者查詢。現行電子化系統，大都可將這些資料儲存於系統，供進一步使用。

（四）資料寄發

為爭取時效，在讀者要求的時間內，將資料傳送給讀者；通常有一星期一次，半個月或一個月一次。現行電子化系統，多以電子郵件方式寄送。

SDI 服務和其他服務最大不同的特點，在於個別讀者興趣檔的發展維護，而且可以隨資訊需求改變而更改。同時，研究顯示，那些規律使用 SDI 服務的研究者，其研究生產力比起那些不常使用或沒有訂閱的同事們要高（研究生產力乃是以所出版的文章數量來評量）[7]。

2-1-2 歷史發展

溯及歷史，SDI 一開始是以人工服務的方式。1920 年代，紐約公共圖書館請讀者填寫興趣卡（即 Profile 檔），當有新資料進館時，圖書館就依此興趣卡來寄發相關資料。但當時尚未有「專題選粹服務 SDI」的正式名稱，這項服務雖然相當受到讚賞，但需要大量的人力，因此沒有擴大服務[8]。

到 1940 年，哥倫比亞大學開始使用一個稱為「書目服務」的程式，這是在 MEDLINE 和 Library Literature 二個資料庫中查詢到最早的自動化 SDI 紀錄[8]。

正式最早的專題選粹服務是 Luhn 於 1961 年實施。他創先致力於通知研究者新資訊，在 IBM 公司用當時的電腦建立一套「商業智慧系統（Business Intelligence System）」，整理新到文件，經由關鍵字統計分析，篩選出文章摘要，然後將摘要傳送給組織內有興趣的人；這些看過摘要的人接下來可能便會索取全文[1]。

有趣的是，SDI 的發展並非從圖書館開始。Luhn 的動機是要修補原有人工方式的不足之處，但因為當時電腦設備昂貴，以及 Luhn 個人背景的推廣範圍因素，使 SDI 從一些商業公司和研究機構開始發展[6]。

這種自動化風潮在 1960 年代及 1970 年代早期逐漸受到重視；1970 年代中期因維護系統的成本增加，提供 SDI 服務的廠商減少，也使圖書館減少 SDI 的推廣活動，重新評估付出成本與維護 SDI 讀者興趣檔的利益是否值得。到了 1980 年代，科學研究者仍使用 SDI 服務，但多數資訊專家們並不積極推廣[7]。

不過，在 1960 及 1970 年代，人工手動的 SDI 服務在醫學圖書館還是相當普遍的。直到 1972 年，美國國家醫學圖書館（National Library of Medicine, NLM）引進 SDILINE 系統，自動化的 SDI 才廣泛取代了人工 SDI[9]。

這些 SDI 的詳細發展，整理成表 2-1。

表 2-1　SDI 大事年表

年　份	重 要 事 件
1958 年 10 月	IBM 公司構想利用機器輔助處理資料，促進資訊的傳播
1959 年 5 月	IBM 公司測試 SDI-1 系統
1960 年 4 月 ~1964 年	IBM 公司陸續完成 SDI-2，SDI-3，SDI-4，SDI-5 系統
1962 年	美國化學學會（American Chemical Society）以磁帶方式供應化學文獻的篇名（Chemical Titles）
1963 年	艾美斯實驗室（Ames Lab）將機器可讀的輸入與回饋設備，裝置在其 SDI 電腦系統
1965 年	ISI 發展一套商業性的 SDI 系統-ASCA（Automatic Subject Citation Alert）
1968 年 2 月	雪爾研究公司（Share Research Corporation）出版 SDI 書目
1968 年 9 月	美國印第安那大學（Indiana University）利用 LC 的 MARC Tapes 提供 SDI 服務
1968 年 9 月	ASIDIC（Association of Scientific Information Dissemination Centers）利用磁帶提供 SDI 及回溯性服務

1968 年 11 月	“化學資訊會議”在荷蘭舉行，討論利用電腦來處理資訊
1968 年 11 月	美國國家醫學圖書館介紹 MEDLARS 系統（Medical Literature Retrieval System）
1969 年 4 月	威斯康斯大學（University of Wisconsin）舉行有關 SDI 的研討會議，評鑑 SDI 的發展與作業
1969 年 6 月	ASIS-SIG/SDI（American Society of Information Science –Special Interest Group）調查研究 SDI 系統
1969 年 10 月	ASIS-SIG/SDI 正式成立
1970 年 4 月	EUSIDI（European Association of Scientific Information Centers）提供電腦的化學資訊服務
1971 年 6 月	ASLIB（Association of Special Libraries and Information Bureaux）開會討論資訊服務問題
1973 年 1 月	ISI 提供社會科學引用文獻索引的磁帶服務

資料來源：[10]

而國內 1972 年則是由李德竹首先引進 SDI 之定義及運作方式[11]。其後，中山科學研究院為了解決閱讀資料量持續增長的問題，於民國 65 年 8 月開始籌備「圖書館提供資訊服務計畫」，成立資訊服務組。因試行效果良好，於民國 66 年 8 月，擴大推廣，是國內最先推行專題選粹服務的單位[10]。

2-1-3 特色和貢獻

1.特色

每個 SDI 系統都有它各自的特點，在此則歸納提出一般 SDI 系統共有的特色[5]。

（一）明確的挑選

SDI 系統中，資訊挑選的條件是以能描述讀者搜尋型態的興趣檔為基礎。若系統中興趣檔的建立能反映讀者以往的搜尋習慣，系統接受度和滿意度會較高。

（二）簡便的通知

使用 SDI 系統的最後結果，就是要獲得和興趣檔相符合的新書目摘要通知及期刊全文。以往是用紙本列印傳遞，現今多以電子郵件通知。

（三）全面的回饋

為了評估系統效能，讀者常被詢問對於通知內容的反應，而多以文獻的查全率（Recall）及精確率（Precision）來衡量。但 Schneider（1971）認為，與讀者的互動不應只侷限在讀者感興趣的文獻，還應該包含讀者引用該文獻後對其研究工作的影響。對於 SDI 系統及興趣檔的效率，其他學者還提出有以文獻訂購數量、付費意願、個人訪談、及問卷調查等來衡量。

（四）興趣檔的更新

若讀者長期持續相同的研究，其文獻搜尋型態一般會穩定不變。如果因為興趣改變，或其他因素，造成搜尋重點突然轉變，就必須根據轉變調整興趣檔。

SDI 系統管理者應該全面策略性地考量這四項特色，其中任何一項特色的改進，都會增加系統的使用敏感度。然而，在許多文獻報告中，對於 SDI 系統的重點都放在系統如何設計、硬體設備的設定、軟體的包裝，而較少注意到這些特色與使用者間的互動。管理者、研究人員、和電腦工程師們會注意到 SDI 系統的存在，但卻很少進一步瞭解可以如何利用，真正解決資訊處理的問題。

故此，為提升 SDI 使用率，必須調整思考方向，對於潛在使用者應先告知這四項特色及優缺點。而且，系統功能應和電腦硬體設備分開，因為使用者只在意所提供的功能，所以這些功能必須符合使用者喜好及定位。

2. 貢獻

新知通報（Current Awareness Service, CAS）是比專題選粹服務更為廣泛的概念。不局限在期刊書目上，只要有新的公告或圖書期刊到館，就會通知讀者。

對圖書館而言，CAS 和 SDI 都可增進和讀者的積極合作關係。並且在圖書館實體建築之外，還能持續讓讀者方便獲得從圖書館散播出來的資訊。相對其他圖書館服務需要讀者積極追蹤所要的資料，CAS 和 SDI 則是積極的追著讀者。此種服務形式也幫圖書館打廣告，每當讀者接到一份通知，就等於提醒一次讀者圖書館存在的事實，也提醒讀者圖書館正為他們服務。而讀者興趣檔則對典藏發展提供有價值的資訊，因為其資料庫可作為採購決策的指標。最後，

類似期刊目次這樣的電子服務，連結到文獻傳遞，也減少期刊因較少使用而被取消購買的機會[12]。

2-2 個人化服務

個人化資訊服務一直是圖書館努力的方向之一，拜近年來科技發展之賜，個人化資訊服務的可行性及影響已逐漸受到重視。這項新的服務空間可能是對傳統圖書館服務的一項挑戰，也可能替圖書館開創新的契機[13]。

圖書館向來提供的 SDI 服務也可說是個人化服務的一種。以下便就個人化服務及本研究之相關設計做一概括介紹。

2-2-1 定義

個人化（Personalization）是針對不同的個人，給予不同的產品或服務，以滿足屬於個人的需求。此概念最早出現於製造業，但稱為「大量客製化（Mass-Customization）」，意指依客戶的需求去生產。更早期的製造業為降低生產成本，達到經濟規模效益，採取「大量生產（Mass-Production）」的方法；但隨著生活水準的提升，個人需求日增，而有「大量客製化」的概念產生。將此概念應用到服務業，就是「個人化」。

「個人化」比「客製化」更強調每個個人，Mittal & Lassar（1996）提出二者間的差異：

- 客製化不需透過人員的接觸即可完成，但個人化必須和顧客接觸。
- 個人化不是以任務（Task）為導向，而是利用互動的方式瞭解需求。
- 負責個人化的人員必須隨時提供顧客所需要的需求。

在網路上，「個人化」與「客製化」常代表同樣的涵義，但兩者定義有所不同。個人化由網站主導，藉由紀錄使用者的個人資料或瀏覽行為，網站即可提供滿足使用者個人需求的內容；而客製化則由使用者主導，自行決定網頁的畫面或資料呈現方式，例如在 MyYahoo 中，使用者可能把天氣資訊移到首頁最上方，以備經常查詢之需[14]。

從商用行銷的角度來看，個人化服務可說是區隔目標市場的一種行銷方法。透過和顧客間瞭解(Understanding)、溝通(Communication)、與謙恭友善(Courtesy)的行為，找出「因人而異」的因素和最終商品，讓每位顧客感覺是為他「量身訂作」的商品，從而提升整體的服務。將此觀點應用到圖書館的資訊服務，則是透過對讀者個別差異的了解，提供滿足個人需求的服務，提高整體讀者服務的滿意度。

2-2-2 個人化的方法

個人化在技術上有許多不同的分類與作法，有些方法需要使用者主動參與，例如選填表單或回答一系列設計好的問題；另外則是採取隱藏的方式，直接追蹤使用者的資料與行為，不需要使用者輸入資料。整體而言，目前網路上使用的方法可分為下列五種[15]：

1.以規則為主的過濾法（Rule-Based Filtering）

此法需要使用者先回答些問題，然後依據回應的資料來傳送內容。對網站而言，需事先將企業規則（Business Rules）邏輯化。如本研究系統中，根據所輸入的登入帳號，判別所屬學院，再列出該學院背景相關的期刊供選擇。

2.學習性代理人技術（Learning-Agent Technology）

這是最不干擾使用者的個人化方式，透過系統自動追蹤使用者在網站上瀏覽行為來達成。而且這些系統通常也有學習能力，能根據使用者實際點選了網頁中哪些項目，調整網頁下次展示的內容。如本研究中，藉由圖書館自動化系統中對讀者瀏覽期刊的歷史紀錄，推測讀者對期刊的喜好和需求，而供選擇訂閱。

3.以內容為主的過濾法（Content-Based Filtering）

以文件內容作為過濾的對象並加以分析比較，系統會將新進資訊和興趣檔做比對達成資訊過濾。通常以關鍵字詞比對方式對文件內容(Content) 加以分析比對，著名的商用系統如 Verity、CompassWare、IBM InfoSage 及 InfoSeek Personalized News 等皆是採用此類技術。

4.以合作為主的過濾法（Collaborative-Based Filtering）

此法不但會根據使用者興趣來判斷新進資訊，還會考慮使用者的背景、系統推導等因素。如圖書館的 SDI 服務；亞馬遜網路書店，依購買紀錄推薦新進書籍。又如若有讀者借閱過鄭愁予、楊牧等書籍，就可推斷該讀者對新詩有興趣，進而推薦其他新進的新詩作品。

進一步還可利用其他技術，如資料探勘技術（Data Mining）加以輔助，讀者和系統同心協力來做資訊過濾。如可找出讀者通常會同時一起借閱那些書，當有讀者對其中一本有興趣時，便可推薦其他相關藏書給讀者。但這些技術較

為複雜，因此採用的系統較少。

5.以限制條件為主的過濾法（Constraint-Based Filtering）

和以內容為主的過濾法相似，只是先輸入查詢的條件，以縮小查詢的範圍，例如查詢多少價錢內的書籍、一段時間內出版的書籍。本研究中以關鍵字及輸入的過濾、檢索條件，查詢出精確所需的期刊文章。

因為使用的推薦方法不同或混合使用上述幾種方法，所以產生個人化的程度也不同。程度可分為三種：第一種是無個人化（Non-Personalized），如只用一些統計摘要方法所產生，每個人的推薦內容是一樣的。第二種是短期的個人化（Ephemeral Personalization），主要依據顧客最近的網頁瀏覽記錄或選取了哪些商品，來推薦可能的商品；再利用以內容為主的過濾法，或資料探勘找出商品間的關係來推薦。第三種則是長期的個人化（Persistent Personalization），此種方式會長期追蹤顧客興趣，主要利用顧客的歷史紀錄檔找出使用者和使用者間的關係，再利用合作式的過濾方法來推薦[16]。

個人化的技術固然在資訊過濾上帶來不少好處，但隱私權的議題是它被質疑的缺點所在。因為這些技術或多或少都需要使用者的個人資料，或由系統推斷出進一步的個人喜好資訊，造成部份人士覺得隱私受到侵犯[17]。此外，越是個人化的設計，可能導致系統操作方式越難理解，或造成伺服器的運算處理負擔；因此，如何以最少的資訊，達成簡單又滿足使用者需求的個人化服務設計，將是系統設計上的一項重要議題。

2-2-3 個人化數位圖書館（MyLibrary）

近年來，數位圖書館已漸為趨勢；完整、理想的數位圖書館，不需要館員的實際協助就能順利使用圖書館資源；或者可以利用新技術建立虛擬館員給予讀者協助，以實現個人資訊空間（Personal Information Space, PIS）的模式。在這個人資訊空間內，讀者可將常用的資訊匯集在一起，以便隨手取得，隨時運用。此模式的重要特性如下[18]：

- 讀者根據自己的需求在資訊網路中找資料。
- 對於匯入 PIS 的資源，依讀者適合的方式各自處理。

- 讀者透過瀏覽器即可進入自己的 PIS，直接接觸資訊。
- 讀者專業的參考問題，可以透過公開的討論空間獲得解答或指引。
- PIS 裡有讀者的資訊需求與檢索行為的紀錄，透過分析這些資料，數位圖書館可以主動提供讀者更好的服務。

然而，數位圖書館不是一套單一完整的數位系統，更不可能讓世界各地的使用者從這單一的數位系統中就可取得任何學科領域的所有資訊。相反的，數位圖書館比較像是一群異質系統或資源的組合，每一個個別的系統或資源都是依據特定讀者群的需求而建立[19]。

如何找出讀者最關切、有用的資訊成了一項研究課題。個人化服務的方法便是因應此研究課題而生。隨著網路上個人化的風潮，圖書資訊技術協會（Library and Information Technology Association, LITA）在 1999 年提出，My-Library 的模式是最值得關注的服務方式。讀者希望客製化、互動性、以及能充分支援的服務；著重圖書館本身，而非針對讀者的服務，將逐漸不受重視[20]。

在圖書館中落實個人化服務有二方面助益。首先，提供個人化服務的網頁讓使用者選擇最相關的資訊來呈現，最後所呈現的是使用者和館員共同控制下的結果，也更接近使用者的需求。因此，省去館員製作、維護重複網頁的困擾。其次，圖書館資源及館員服務可依據讀者個別興趣、最常使用的資源類型、及學術研究領域等要素，針對目標作充分的行銷，傳遞新資訊給正確的讀者將會容易許多[21]。

2-3 RSS（Really Simple Syndication）

2-3-1 RSS 簡介

RSS 最早是 1997 年由 Dave Winer 在 UserLand 公司時設計出的一種規格，由於應用了 XML，使得撰寫網頁不用像以往一樣編輯 HTML 檔，只要編輯好一個個的 item 內容，便可轉換成 HTML 的網頁。ScriptingNews 是他近似自傳的網站（http://www.scripting.com），也可說是個人部落格網站的先鋒[22]。

RSS 可視為一個小型資料庫，內含網站標題及內容、和網站本身的描述，起先主要用來發佈和聚集網頁內容(例如新聞標題)。對網頁內容供應方而言，可以利用 RSS 半主動地將該網站的某些訊息、狀態等進行整理後，透過標題列的顯示方式提供給讀者閱讀，免去讀者要在眾多文字、圖片的網頁中尋找標題位置，也不必再一一通知訂閱戶內容有所更新；對網頁訂閱方而言，只要訂閱 RSS feed，不必開瀏覽器，就可透過 RSS Reader 自動獲取網頁更新的內容；如此可省卻許多訊息傳遞的時間和成本，達到內容聯合（Content Syndication）的自動化。

故此，同樣一份文件，可以經由許多入口（各處的 RSS Reader）連結取閱，而不必有許多相同的複本散落網路各角落。就網站管理者而言，可以使網站獲得最多回溯連結次數；就網頁瀏覽者而言，得以瀏覽網站最新內容；RSS 可說是創造此雙贏局面的利器[23]。

RSS 發展至今有多種版本，各版本雖具有相同英文名稱縮寫，但其英文字母所代表的意義卻完全不同，表 2-2 是簡單的整理。

表 2-2　RSS 版本比較表

版本	名稱	發佈時間	發佈單位	RDF 支援	XML Namespace
0.90	RDF Site Summary	1999.3	Netscape	有	有
0.91	Rich Site Summary	1999.7	Netscape（RSS 0.90）+ Userland（ScriptingNews）	無	無
0.92（0.9x）	Rich Site Summary	2000.12	UserLand	無	無
1.0	RDF Site Summary	2000	RSS-DEV	有	有
2.0	Really Simple Syndication	2002.8	UserLand	無	有

資料來源：[24]

由於版本發展混亂，對 RSS 的控制權也成為一項議題。在 2003 年 7 月 15 日，UserLand 軟體公司將 RSS 2.0 的所有權移轉給哈佛的伯克曼網際網路與社會中心（Berkman Center for Internet & Society at Harvard Law School），成立獨立的諮詢委員會，推廣 RSS 使用，及給予 RSS 開發人員建議。Dave Winer 的用意即在停止各大廠商對 RSS 的繼續開發和壟斷，以維持 RSS 簡單易用的特性；也終止對 RSS 控制權的疑慮[25]。

2-3-2　RSS 優缺點

一項新技術的產生，固然解決一些問題，帶來許多好處；在便利與普及背後，也相對產生其他問題。本小節針對 RSS 技術所遇到的各項狀況及優缺點加以分析記錄。

1.RSS 的優點

（一）新資訊的推播工具

對於網路上浩瀚無窮的資訊，除了搜尋引擎工具可以快速找到所要的資

訊外，新的「推播（Push）」技術則將所要的資訊帶到使用者面前。透過 RSS，使用者無須在多個網站間尋覓，造成無謂的時間和精力浪費；一旦訂閱後，可自動追蹤更新資訊。

（二）新資訊的簡單發佈工具

RSS 降低了發佈新資訊的技術門檻，內容提供者，無論企業或個人，都可用簡單的步驟在網路上發表訊息。訊息發布管道可避免被傳統的「中間者」過濾和延遲。

2. RSS 的缺點

（一）版本眾多混亂

RSS 是 XML、RDF（Resource Description Framework）標準的延伸應用，自發展以來，已有好幾個版本，但尚未成為 W3C 組織（World Wide Web Consortium, http://www.w3.org）和 IETF（Internet Engineering Task Force）組織之 RFC（Request For Comments）正式標準。版本的混亂不一，沒有統一標準，讓 RSS 撰寫者、以及軟體解析上，遇到不同的問題。這些問題整理如下：

- 1.0 版因採用 RDF 和模組化，造成和其他版本差異最大；而 2.0 版的模組化並不沿襲 RDF。
- 各版本的的「必要」、「非必要」元素不太相同，造成<item>項目的表達有好幾種方式
- RSS 2.0 的<pubDate>元素，其日期格式應照 RFC 822 規格表示，但網路上的 RSS feed 卻有多種寫法，如：
  - 2003-03-21T16:28:40
  - 2003-04-03T07:45:57-08:00
  - Fri, 04 Apr 2003 05:04:39 GMT（RFC 822）
  - Fri, 28 Mar 2003 05:18:59 -0800
  - 1049379042.0

  在 RSS 1.0 中，相對的<dc:date>元素也出現多種寫法：
  - 2003-03-21T16:28:40
  - 2003-01-17T13:03:00+00:00

- 2003-03-27T19:41:49-06:00

- guid 與 link 元素的區別

RSS 2.0 版裡，<guid>是<item>元素下新增的子元素，也是 2.0 版的特色。但因其功能定義和<link>子元素有相似之處，引起使用者的困惑，也常被拿來討論。

<link>元素指的是該<item>元素的 URL 網址。<guid>元素全名為 Globally Unique Identifier，內容為一單一獨立字串，代表該<item>，藉以區別其他<item>，避免有相同名稱和內容的<item>出現。因此，此字串不能更改，也避免 RSS Reader 軟體將同一<item>呈現多次。

<guid>元素的非必要屬性：isPermalink，則是造成困惑的原因。若該屬性值為預設的 false，即代表該<item>；若該屬性值為 true，則<guid>元素的內容則是該<item>元素的永久參考 URL 網址。假設該<item>內容因更新版本，舊版內容被移往另一永久儲存位置，<link>元素指的是新版內容的現行 URL 網址，而<item isPermalink =”true”>指的是舊版內容的永久儲存位置[26]。

（二）RSS feed 的流量管理

隨著 RSS 的普及盛行，提供 RSS feed 訂閱的網站伺服器負擔也越來越重。雖然 RSS feed 內容通常只包含標題和摘要，一旦訂閱數量過高，要保持伺服器上的 RSS 內容更新，並且避免訂閱者過於頻繁地向伺服器索取更新內容，成為所有 RSS 伺服器管理者的問題。如 CNET 網站就曾經因承受不住 RSS 巨大流量，而減少 RSS 提供內容，但卻失去部分網站訪客，只好再恢復 RSS 提供內容。

要解決此現象，除了基本上控制發佈與訂閱的數量外，目前一些 RSS Reader 軟體，如 SharpReader 可以自行設定更新下載頻率。在 RSS 2.0 版中，藉由幾個元素調整更新頻率（Refresh Rate），整理如表 2-2。

表 2-3　RSS 2.0 頻率更新元素

RSS feed 元素	意義
<channel>→ <pubDate>	以 RFC 822 格式，表示 feed 內容的出現在此時間之後，可以是未來的時間；只少部分 reader 軟體可回應此元素。 ex：<pubDate>Sun，12 Sep 2004 19：00：40 GMT</pubDate>
<channel>→ <lastBuildDate>	以 RFC 822 格式，表示 feed 上一次更新的時間，一定是過去的時間，reader 軟體以此為主。
<channel>→ <ttl>	Time-to-live，最小分鐘數，即 reader 軟體在 feed 來源更新前必須等待的時間。Feed 作者應調整此數字，以反應 feed 更新和被讀取的時間，以及希望訂閱者跟上更新的程度。 ex：<ttl>60</ttl>
<channel>→ <skipDays> / <skipHours>	用以控制訂閱者何時讀取 feed，訂閱者不該在所列時間來讀取 feed。 ex：<skipDays><day>Monday</day></skipDays> <skipHours><hour>20</hour></skipHours> 表示告訴訂閱者不要在星期一或每晚 8 點來讀取 feed
<channel>→ <item>→<pubDate>	和<channel>→ <pubDate>相同

資料來源：[27]

雖然 Feed 內容常更新，若在 Reader 軟體中設定一小時更新一次，就表示每小時要去下載一次。是否有這樣的需求值得事先考量。

另外，由 RSS 版本規格探討中得知，RSS 0.91 版是屬於 Pull-Based，也就是完全依賴訂閱方向伺服器索取新內容；RSS 0.92 和 RSS 2.0 依靠相關元素達成發佈和訂閱的協定；而 RSS 1.0 則是以模組化的方式來達成。

RSS 的發佈與訂閱，固然技術設計上已經有所考量；對伺服器管理者和出版者而言，如何利用這些設計，在內容更新速度與網路流量負擔間達到有效、平衡的控制，將是一項考驗。

2-3-3 RSS 相關名詞

在瞭解 RSS 發展及詳細規格之外，以下將實際應用 RSS 時會遇到的相關名詞做一簡單整理，分述如下：

RSS Reader

剖析、接收 RSS Feed 檔案的軟體，有各種形式，包括單機安裝、瀏覽器內掛、和電子郵件方式；支援各作業平台，如 Windows、Mac OS、及 PDA。使用者只要找到 RSS Feed 檔案的 URL 網址並訂閱，就可定期收到更新內容。

RSS Feed

放置資料內容的檔案，資料內容可能是一則新聞或網誌，檔案則是一份 Well-Formed 的 XML 檔案，依循上述版本規格而有不同的寫法。相對的，現在很多 RSS Reader 軟體也同時支援不同的 RSS feed 版本。

RSS Aggregator

RSS Aggregator 為 RSS Feed 提供多一層的使用方法，其將 RSS feed 收集在同一介面，分類、更新並過濾標題，成為訂閱、搜尋 RSS feed 的來源，如 O'reilly Meerkat 網站（http://www.oreillynet.com/meerkat）。有些網站甚至提供公開的 Service API 呼叫，供訂閱戶運用，以分析 RSS Feed 的點選率及連結數。現在有些人把 RSS Aggregator 和 RSS Reader 軟體視為相同，其實是不大一樣的。

OPML（Outline Processor Markup Language）

是一個 XML 檔案，起源由 Radio UserLand 公司發展而成，用在 Outliner 的文字編輯程式中。Outliner 是將文字資料以階層式結構集合整理的編輯程式。因為具有結構性，OPML 檔可以讓許多 RSS Feed 清單有次序地匯集成單一檔案，以供 RSS Aggregator、RSS Reader 軟體匯入、匯出互相交換使用；這是應用最普遍的地方。

第三章　RSS 技術與 SDI 應用

本章說明 RSS 技術與 SDI 的應用狀況。第一節詳細探討 RSS 版本規格；第二節整理出電子郵件與 RSS 技術的差異；第三節整理出 SDI 使用 RSS 技術和使用電子郵件二種方式不同之處。

3-1　RSS 版本介紹

3-1-1　RSS 版本規格

目前 RSS 有多種版本規格，而最早由 Netscape 主導的 0.90 版已較少使用，在此僅介紹 0.91、0.92、1.0、及 2.0 版的詳細規格文件。

1.RSS 0.91 版

此版作者 Dave Winer 根據 Netscape 的 0.90 版再編輯修改，文件結構如圖 3-1。UserLand 的 0.91 版和 Netscape 的 0.91 版最大不同，在於少了 DTD 宣告；而 Netscape 的 0.91 版是所有 RSS 版本中唯一有用到 DTD 宣告的版本。因此，至於是否使用 DTD 宣告可由個人決定。

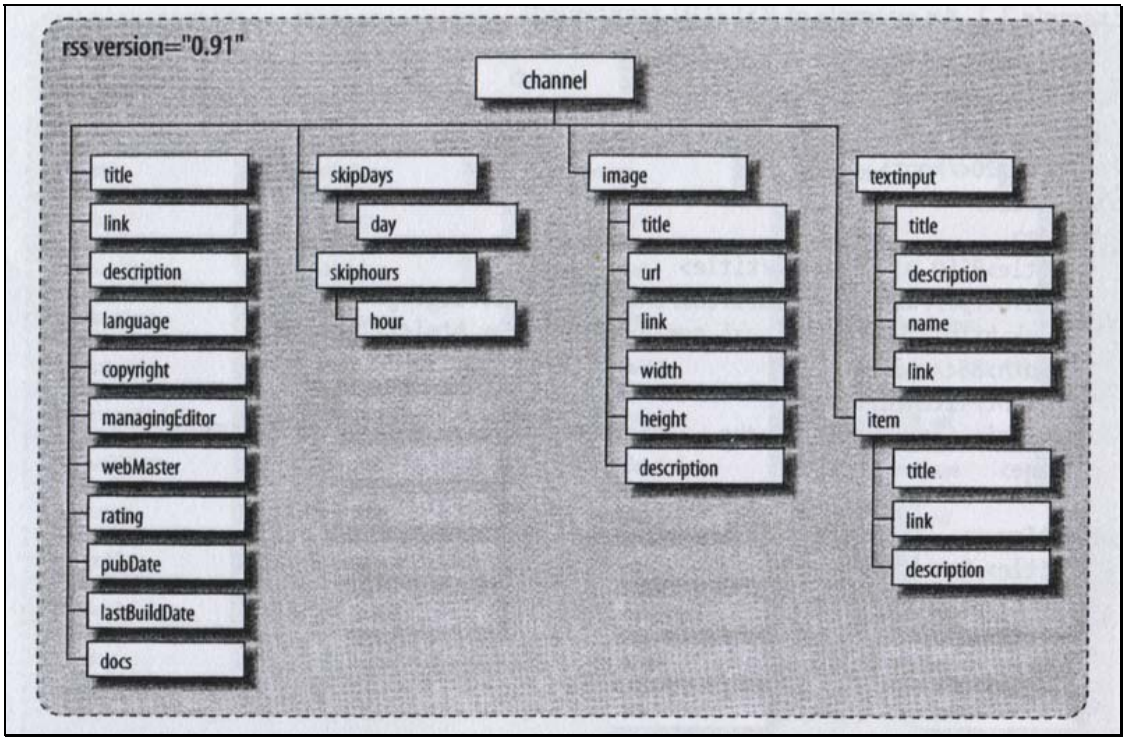

圖 3-1　RSS 0.91 版元素列表[28]

圖 3-1 規格摘要如下：

- 以 XML 為基礎，根元素是 rss，接著包含一個 channel 元素。
- 每個 channel 元素其下最多 15 個 item 子元素，每個 itme 元素有 title、description、及 URL 子元素。
- metadata 資料較其他版本少。
- 訂閱者需要主動讀取 RSS Feed，屬於 Pull-Based 的技術。

圖 3-2 是一 RSS 0.91 版範例：

```
<?xml version=”1.0” encoding=”ISO-8859-1” ?>
<rss version=”0.91”>
<channel>
   <title>RSS0.91 example</example>
   <link>http://www.exampleurl.com/example/index.html</link>
   <description>this is an example 0.91 feed</description>
   <skipDays><day>Monday</day></skipDays>
```

```
    <image>
      <title>RSS 0.91 example</title>
      <url> http://www.exampleurl.com/example/images/logo.gif</url>
<link>http://www.exampleurl.com/example/index.html</link>
<width>88</width>
<height>31</height>
    </image>

    <item>
      <title>the first item</item>
      <link> http://www.exampleurl.com/example/001.html</link>
      <description>this is the first item</description>
    </item>
</channel>
</rss>
```

圖 3-2　RSS 0.91 版範例

2.RSS 0.92 版

2000 年時，Netscape 對 RSS 失去興趣，因此 0.92 版由 Dave Winer 主導，並以之前 0.91 版為基礎修改延展而成。圖 3-3 是其文件結構：

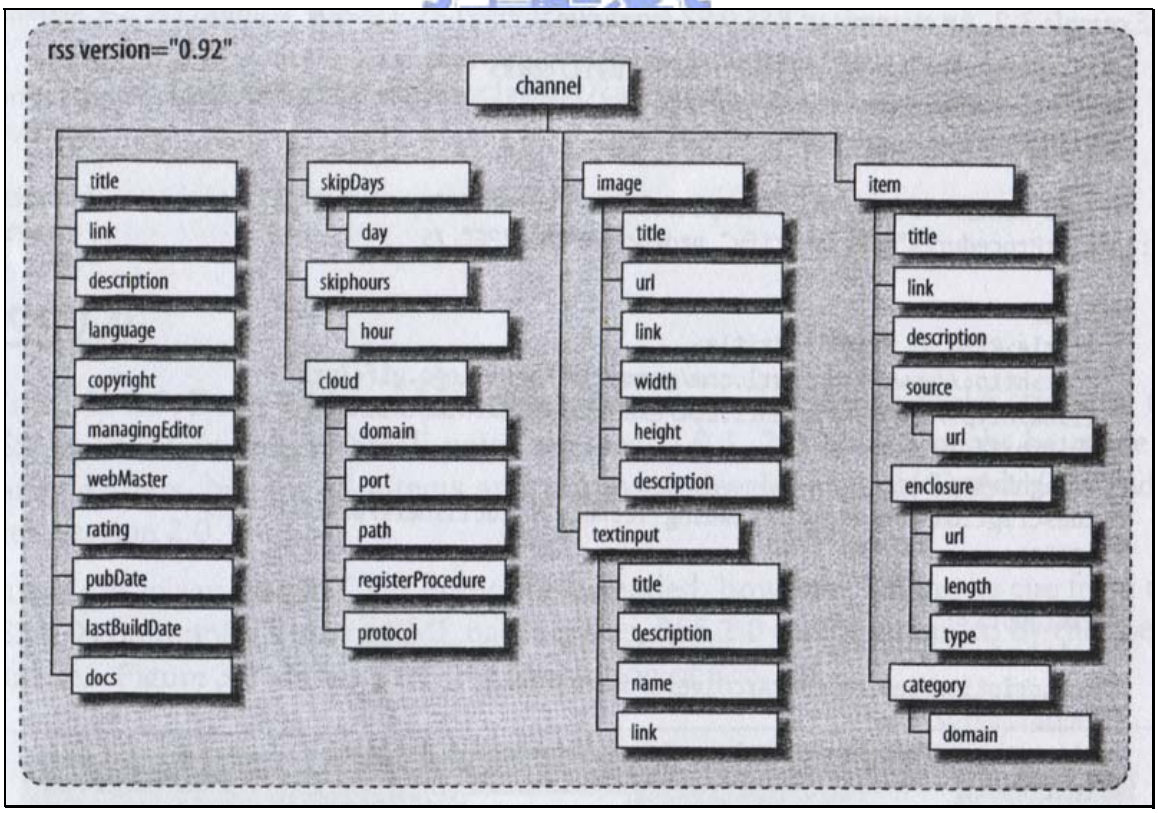

圖 3-3　RSS 0.92 版元素列表[28]

圖 3-3 規格摘要如下：

- 以 XML 為基礎，根元素是 rss，包含一個 channel 元素，其下 item 元素及子元素數量則不限制，元素的字元長度不受限制。
- 比 0.91 版擁有更多 metadata 資訊。
- 較 0.91 版增加四個元素：source、enclosure、category、cloud。其中 cloud 元素的屬性有 domain、port、path、registerProcedure、及 protocol；用以說明 RSS Feed 的 Publish 及 Subscription 程序。
- 主要是 Pull-Based 技術，但有 Publish 及 Subscription 的功能。

圖 3-4 是一 RSS 0.92 版範例：

```
<?xml version=”1.0” encoding=”ISO-8859-1” ?>
<rss version=”0.92”>
<channel>
   <title>RSS0.92 example</example>
   <link>http://www.exampleurl.com/example/index.html</link>
   <description>this is an example 0.92 feed</description>
   <skipDays><day>Monday</day></skipDays>
   <! - - 新增的 cloud 元素，表示 Publish/Subscription 功能的相關設定 - - >
   <cloud domain = “http://www.oreilly.com “ port=”80”   path=”/RPC2” registerProcedure=”pleaseNotify”
protocol=”xml-rpc” / >

   <image>
      <title>RSS 0.92 example</title>
      <url> http://www.exampleurl.com/example/images/logo.gif</url>
      <link>http://www.exampleurl.com/example/index.html</link>
      <width>88</width>
      <height>31</height>
   </image>

   <item>
      <title>the first item</item>
      <link> http://www.exampleurl.com/example/001.html</link>
      <description>this is the first item</description>
      <! - - 新增的 source、enclosure、category 子元素 - - >
      <source url =” http://www.anothersite.com/index.xml” >Another Site</source>
       - -  item 由此網址取得 - -
      <enclosure url = “http://www.oreilly.com/001.mp3 “  length=” 54321” type” audio/mpeg” />
       - - 和 item 相關的檔案描述，長度單位爲 bytes，type 爲 MIME 格式 - -
      <category domain = “http://www.dmoz.org” >Business/Industries/Publishing/Publishers/Notifiction/</category>
       - - item 在 domain 屬性值內的分類，以階層表示 - -
```

```
  </item>

</channel>
</rss>
```

圖 3-4　RSS 0.92 版範例

3.RSS 1.0 版

此版本由網路社群 RSS-DEV 工作小組發展而成,和以往版本最大不同改變，在於重新導入 RDF 和 XML Namespace 的使用。導入 RDF 使此版本在語法上有重大改變，也因此可將 RSS item 和屬性間的關係以圖表示；使用 Namespace 則讓 1.0 版具有模組化（Modularization）的特色。因為模組化，開發人員在增加新功能時，不需更動核心規格；此外，這些模組一直在增加修訂中，並陸續提為標準。

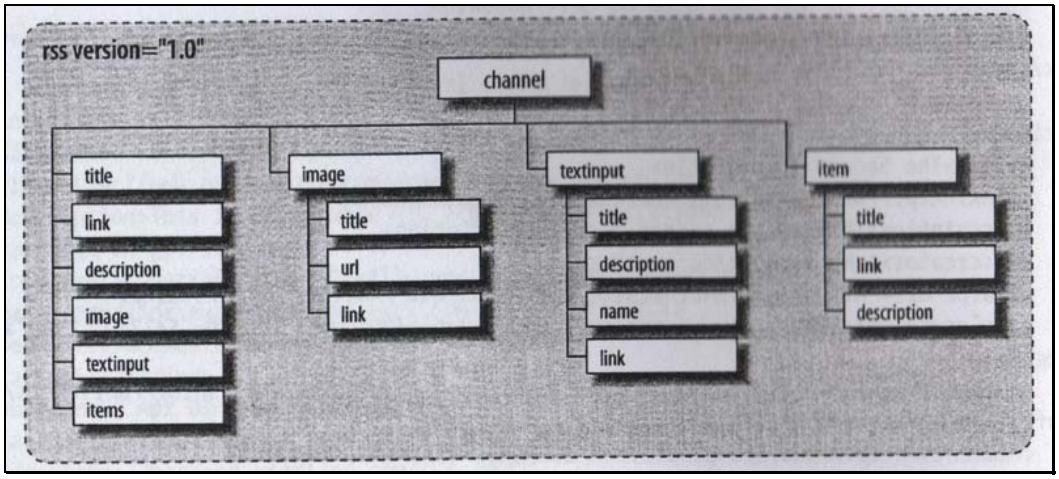

圖 3-5　RSS 1.0 版元素列表[28]

圖 3-5 規格摘要如下：

- 以 XML、RDF 為基礎，較以往版本複雜，也擁有較多 metadata 資訊。
- 模組化，有較大的延展性。
- 主要是 Pull-Based 技術，但有 Publish 及 Subscription 的功能。
- 根元素是 rdf，並且一定以下列的名稱宣告為開頭：

<rdf:RDF　xmlns=”http：//purl.org/rss/1.0/”

xmlrdf=”http：//www.w3.org/1999/02/22-rdf-syntax-ns#”>

其他的名稱宣告則插入在這二行內，目前有 dc（Dublin Core）、sy（Syndication）、co（Company）、ti（TextInput）等標準模組。

圖 3-6 是一 RSS 1.0 版之範例：

```
<?xml version=”1.0” encoding=”utf-8”?>

<! - - 空間名稱宣告 - - >
<rdf:RDF　xmlns=”http：//purl.org/rss/1.0/”
xmlns：dc =”http://purl.org/dc/elements/1.1”
xmlns：sy =”http://purl.org/rss/1.0/modules/syndication”
xmlns：co =”http://purl.org/rss/1.0/modules/company”
 xmlns：ti =”http://purl.org/rss/1.0/modules/textinput”
 xmlrdf =”http：//www.w3.org/1999/02/22-rdf-syntax-ns#”>

<! - - 標明 channel 元素資源在 RDF 中的 URI - - >
<channel rdf：about=”http://meerkat.oreillynet.com/?_fl=rss1.0”>
<title>Meerkat</title>
<link>http://meerkat. oreillynet.com</link>
<description>Meerkat：An Open Wire Service</description>
<! - -　空間名稱模組化的應用　- - >
<dc：publisher>The O’reilly Network</dc：publisher>
<dc：creator>Rael dornfest</creator>
<sy：updatePeriod>hourly</ sy：updatePeriod>
<sy：updateFrequency>2</ sy：updateFrequency>
 :
 :
<textinput rdf：resource =”http://meerkat.oreillynet.com”>

<items>　- - 文件中所有 items 的 RDF 清單 - -
  <rdf：Seq>
<rdf：li rdf：resource =“http://c.moreover.com/click/here.pl?r123”>
</rdf：Seq>
 </items>
</channel> -- channel 元素在此結束 --

<! - - 標明 textinput 元素資源在 RDF 中的 URI - - >
<textinput rdf：about=”http://meerkat.oreillynet.com”>
      <title>Search Meerkat</title>
      <description>Search Meerkat’s RSS Database</description>
      <name>s</name>
      <link>http://meerkat.oreillynet.com</link>
<! - - 空間名稱模組化應用　- - >
          <ti：function>search</ti：function>
      <ti：inputTpye>regex</ ti：inputTpye>
</textinput>
 :
 :
<! - - 標明 item 元素資源在 RDF 中的 URI - - >
<item rdf：about= “http://c.moreover.com/click/here.pl?r123”>
```

```
        <title>XML:A Disruptive Technology</title>
        <link>http://c.moreover.com/click/here.pl?r123</link>
<! - - 空間名稱模組化應用  - - >
        <co：name>XML.com</ co：name>
        <co:market>NASDAQ</ co:market>
        <co:symbol>XML</ co:symbol>
</item>
</rdf:RDF>
```

圖 3-6　RSS 1.0 版範例

4.RSS 2.0 版

RSS 開發小組為了改善並簡化以往的 RSS 版本，在 2002 年 8 月發表 RSS 2.0 版。此版本以 0.92 版為基礎，加入模組化的概念，並新增少數新元素，如下：

- generator：channel 元素內非必要的子元素，表示產生此 RSS Feed 檔案的程式。
- ttl：channel 元素內非必要的子元素，表示 reader 軟體等待更新的時間。
- author：item 元素內非必要的子元素，可以是該 item 作者的 e-mail 地址。
- comments：item 元素內非必要的子元素，包含對該 item 評論的 URL 網址，主要用於部落格網站。
- pubDate：item 元素內非必要的子元素，item 內所指物件的原始發佈時間。
- guid：item 元素內非必要的子元素，item 元素的唯一識別字串。不同於標準化的 GUID（Globally Unique Identifier），RSS 2.0 的 guid 可以採任何形式表示，由 RSS Feed 出版者決定。

相較於 0.92 版，RSS 2.0 版主要改變在於可以包含新模組的名稱空間宣告；且 link 元素可以是任何已註冊 URI 的 URL 網址，因此，如 https://、javascript://、aim:// 都可以使用。

圖 3-7 是 RSS 2.0 版之文件結構：

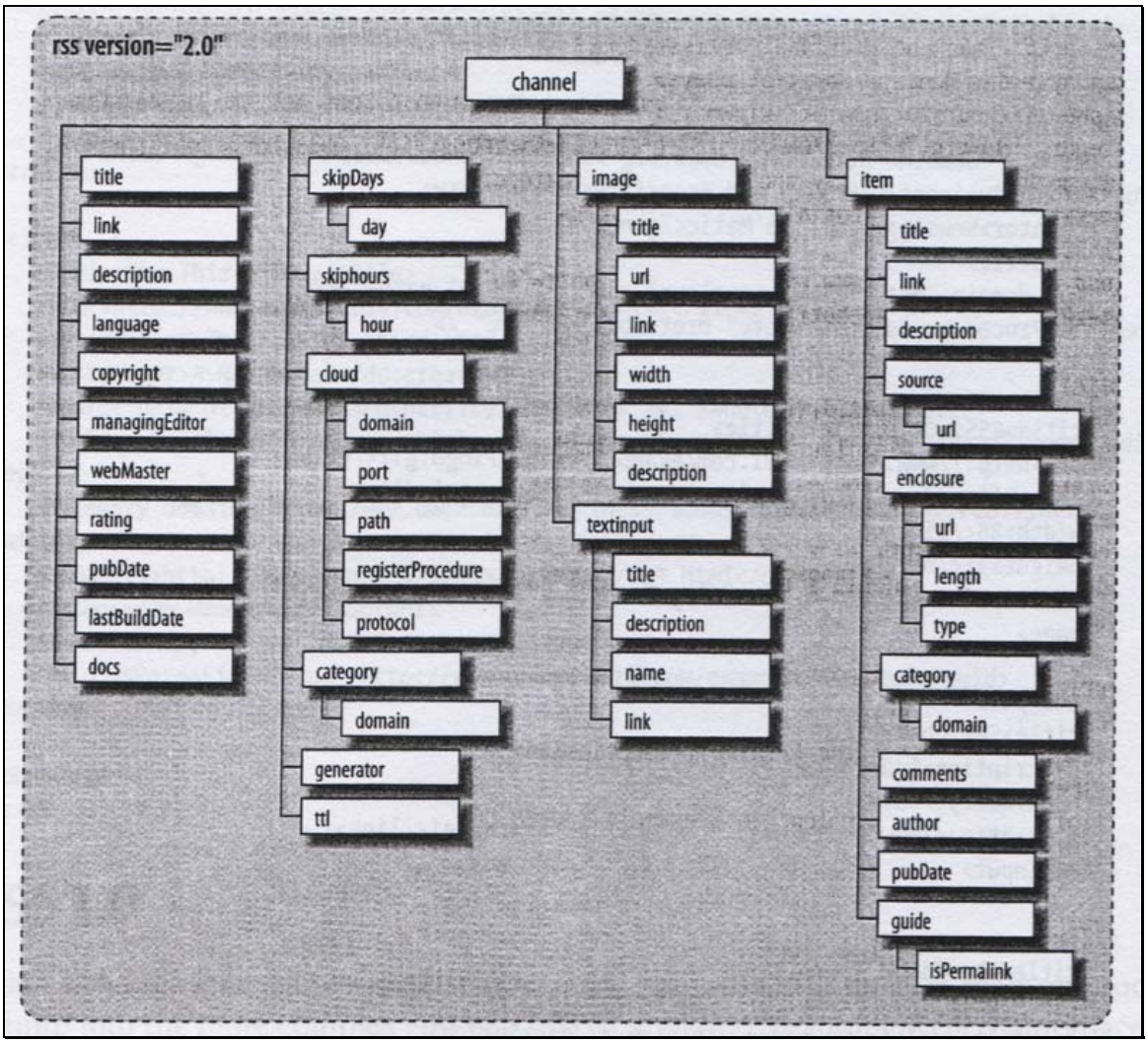

圖 3-7　RSS 2.0 版元素列表[28]

圖 3-8 是 RSS 2.0 版，沒有模組化之範例：

```
<? xml version=”1.0” ?>
<rss version=”2.0”>
<channel>
    <title>RSS 2.0 example</title>
    <link>http://www.exampleurl.com/example/index.html</link>
    <description>This is an example RSS 2.0 feed</description>
    <language>en-gb</language>
    <pubDate>02 Apr 02 1500 GMT</pubDate>
    <lastBuildDate>02 Apr 02 1500 GMT</lastBuildDate>
    <docs>http://backend.userland.com/rss</docs>
    <skipDays><day>Monday</day></skipDays>
    <skipHours><hour>20</hour></skipHours>
```

```
    <! - - 新增的元素：generator、ttl - - >
    <generator>NewsAggregator'o Matic</generator>
    <ttl>30</ttl>
     <cloud domain="http://www.exampleurl.com "    port="80"    path="/RPC2"
                    registerProcedure="pleaseNotify"    protocol="XML-RPC" />

<item>
<title>the second item</title>
<link>http://www.exampleurl.com/example/002.html</link>
<description>this is second item</description>
<source url= "http://www.anothersite.com/index.xml"> another site</source>
<! - - 新增的元素：comments、author、guid - - >
<comments>http://www.exampleurl.com/comments/002.html</comments>
<author>Ben Hammersley</author>
<pubDate>Sun, 02 Jan 2002 0:00:01 GMT</pubDate>
<guid isPermaLink="true"> http://www.exampleurl.com/example/002.html</guid>
</item>

</channel>
</rss>
```

圖 3-8　RSS 2.0 版範例

模組化是 RSS 2.0 最重要的改變。不同於 1.0 版，2.0 版的模組化在 Dave Winer 的堅持下，並不採用 RDF。但 2.0 版可以使用 1.0 版的模組。

RSS 2.0 版主要應用於部落格程式，部落格使用的模組為 BlogChannel Module。此模組包含 channel 元素下的三個非必要元素，並有下列的名稱宣告：

xmlns:blogChannel=" http://backend.userland.com/blogChannelModule"

三個元素如下：

- blogChannel：blogRoll

  OPML 檔的 URL 網址，OPML 檔包含此網站的 blogRoll，blogRoll 是此部落格網站作者習慣閱覽的部落格網站清單。

- blogChannel：blink

  部落格網站作者建議閱覽的網站網址。

- blogChannel：mySubscriptions

OPML 檔網址，此檔包含部落格網站作者的 RSS Feed 訂閱清單。

圖 3-9 是個含有 BlogChannel Module 的範例：

```
<? Xml version=”1.0” ?>
<! -- 模組化空間名稱宣告 -- >
<rss version=”2.0” xmlns:blogChannel=”http://backend.userland.com/blogChannelModule”>
<channel>
   <title>RSS2.0 example</title>
   <link>http://www.exampleurl.com/example/index.html</link>
   <description>This is an example RSS 2.0 feed</description>
    <! - - 模組化內容 -- >
   <blogChannel:blogRoll>http://www.exampleurl.com/blogroll.opml</blogChannel:blogRoll>
   <blogChannel:blink>http://www.benhammersley.com</blogChannel:blink>
   <blogChannel:mySubscriptions>http://www.exampleurl.com/mySubscritipns.opml
                  </blogChannel:mySubscriptions>
 :
 :
</channel>
</rss>
```

圖 3-9　RSS 2.0 版 BlogChannel 模組範例

3-1-2　RSS 版本比較

1. 元素規格的比較

RSS 最令人困惑的問題，在於版本眾多造成使用者與開發人員混亂不清的狀況。雖然現在遍採用 2.0 版的規格，但部分軟體或其他因素所需，仍會摻雜或共用其他版本。表 3-1 將各版本元素及屬性按階層性做一整理，由左至右依 RSS 版本發展排列，元素及屬性分別按字母順序排列，粗體字表示該元素或屬性在此版本首次出現，@ 符號表示為元素的屬性。

表 3-1　RSS 各版本元素及屬性整理[29]

RSS	**RSS**	**RSS**	**RSS**	**RSS**
"RDF Site Summary"	"Rich Site Summary"	"RDF Site Summary"	"no acronym"	"Really Simple Syndication"
0.9	0.91	1.0	0.92	2.0
channel **description** **link** **title** **image** **title** **url** **link** **item** **link** **title** **textinput** **description** **link** **name** **title**	**@version** channel **copyright** description **docs** image **height** link title url **width** item **description** link title **language** **lastBuildDate** link **managingEditor** **pubDate** **rating** textinput description link name title **skipHours** **skipDays** title **webMaster**	channel **@about** description link title **items** image **@about** link title url item **@about** description link title textinput **@about** description link name title	@version channel **cloud** copyright description docs image height link title url width item **category** **@domain** description **enclosure** **@length** **@type** **@url** link **source** **@url** title language lastBuildDate link managingEditor pubDate rating textInput description link name title title skipHours skipDays webMaster	@version channel **category** cloud copyright description docs **generator** image height link title url width item **author** category **comments** @domain description **@type** enclosure @length @type @url **guid** **@isPermaLink** link **pubDate** source @url title language lastBuildDate link managingEditor pubDate rating skipHours skipDays textInput description link name title title **ttl** webMaster

2. RSS 1.0 版的差異

由於 1.0 版採取模組化的方式，造成和其他版本差異頗大，在此以不同角度分析其差異。

（一）以元素的意義而言

就具有相同意義元素來看，RSS 1.0 版元素的表示法，和其他版本相對應元素的表示法，頗為不同，如表 3-2 所列。

表 3-2　RSS 1.0 版的差異[29]

0.9x/2.0	category	cloud	comments	copyright	docs
1.0	dc:subject	cp:server	annotate:reference	dc:rights	xmlns

0.9x/2.0	generator	guid	language	lastBuildDate
1.0	admin:generatorAgent	rdf:about	dc:language	dcterms:modified

0.9x/2.0	managingEditor	pubDate	ttl	webMaster
1.0	dc:creator	dcterms:available		dc:publisher

（二）以版本的結構而言

過去 RSS 0.9x 及 2.0 版，結構如下[27]：

```
<rss>
  <channel>
        <image/>
        <textinput/>
        <item/>
        <item/>
        <item/>
  </channel>
</rss>
```

RSS 1.0 的文件基本結構是：

```
<rdf>
      <channel/>
      <image/>
      <textinput/>
      <item/>
```

```
        <item/>
        <item/>
</rdf>
```

在 RSS 0.9x 及 2.0 版中，channel 元素包含 image、textinput、item 等子元素；而 RSS 1.0 中，image、textinput、item 等元素是和 channel 元素同階層。如圖 3-6 中，channel 元素內含有 image、textinput、item 等元素的 RDF 指標，而後每個 image、textinput、item 元素的第一行有個 rdf：about 的屬性，標明這些元素資源在 RDF 中的 URI。在這些元素內，應用已宣告的名稱空間模組，如 dc、sy、co、ti 等，來描述子元素。

3-2 RSS 與電子郵件的比較

現行電子化 SDI 服務多以電子郵件來傳遞訊息，而電子郵件已成為不可或缺的聯絡工具；方便之餘，最大問題莫過於垃圾郵件氾濫與病毒侵襲。本研究重點在於以 RSS 技術來改善以電子郵件傳遞訊息的缺點，茲將二項技術方法的優缺點彙整如表 3-3。

表 3-3　RSS 與電子郵件的比較

	RSS	電子郵件
使用軟體	各平台、單機版或 Web 版都有	各平台、單機版或 Web 版都有
新訊息處理	只傳遞更新內容，且主動通知	傳遞內容不限，部分軟體有通知新郵件的功能
安全性機制	目前無	PGP（Pretty Good Privacy）郵件加密軟體[28]
夾帶病毒	目前無	許多病毒種類
非自願性廣告	目前無	許多垃圾郵件
處理事項	主要用於傳達新訊息	主要與人聯絡、溝通；也可以傳達訊息
分類整理	可自訂訂閱種類	可自訂收件匣種類
回覆機制	目前無	依需要，使用者可自行回信
訂閱動作	使用者直接訂閱	（電子報、SDI 服務） 有些需身份認證
取消訂閱動作	使用者直接刪除	（電子報、SDI 服務）

		有時步驟繁複，且功能不全

從表 3-3 看出，RSS 技術目前沒有電子郵件的主要問題：病毒和垃圾郵件。要訂閱或取消訂閱也比較容易。

從應用的面向來看，因為本質上的不同，所以主要應用方向也不同。電子郵件的本質在於與人的聯絡溝通上，使用者可以主動編寫寄出一封新的郵件，收信者雖可依需要回覆對方，但卻無從選擇的接收郵件，造成今天垃圾郵件氾濫充斥。而 RSS 的本質是從簡化發佈新資訊程序的角度出發，資訊提供者必須依規格製作 RSS feed 檔案，將新資訊內容放進 RSS Feed 檔案內，大大降低垃圾郵件的產生機會；訂閱者依需求訂閱 RSS Feed，雖然沒有回覆的機制去回應，但達到在提供者和訂閱者間傳遞新資訊的功能。

從技術功能來看，RSS Feed 元素與電子郵件訊息的規格設計有異曲同工之處，現將二者比較整理成表 3-4。

表 3-4　RSS 與電子郵件規格之比較

電子郵件訊息	RSS Feed 元素
Date：	<rss>→<channel>→<item>→<pubdate>
To：	None，feed is for all subscribers，like mailing list
From：	<rss>→<channel>→<webmaster>
Subject：	<rss>→<channel>→<item>→<title>
Message body	<rss>→<channel>→<item>→<description>

資料來源：[31]

在電子郵件中，這些訊息包含在標頭 Header 中，如雙方電子郵件地址、寄信與收信時間、及信件主旨。在 RSS Feed 檔案中，也有相對應的元素，除了接收者的定義，因為 RSS Feed 的接收對象是非固定的訂閱者。

3-3　以 RSS 訂閱 SDI 之流程

目前一些圖書館的期刊資料庫廠商，開始提供以 RSS 方式傳遞 SDI 服務訊息，這些訊息包括新期刊的目次和文章摘要，如 MIT 的 Oxford 資料庫 http://www.oxfordjournals.org/our_journals/index.html，其流程如圖 3-10。

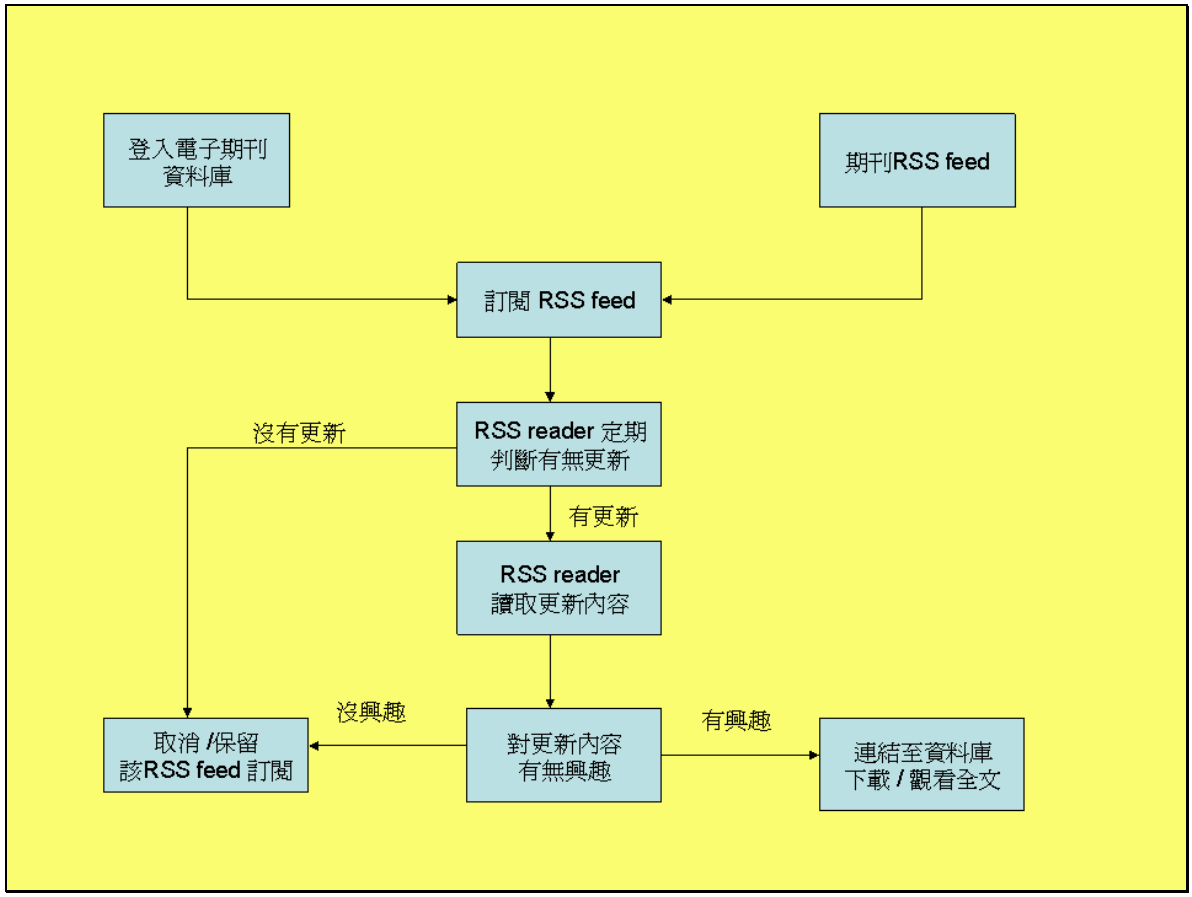

圖 3-10 以 RSS 技術訂閱電子期刊的模式

圖 3-10 中，與使用電子郵件的 SDI 服務模式比較，值得注意的地方如下：

1. RSS Feed 是個符合 XML 規格的檔案，元素內容可視製作者需求而定。目前 RSS Feed 的製作者以資料庫廠商為主。每則 RSS Feed 所傳遞的訊息多是期刊目次，訊息安排較規律；而郵件通知只是告知新到期刊，詳細目次還要進一步連結，且每家資料庫的郵件訊息內容不一。

2. 以 RSS 訂閱電子期刊不需建立興趣檔，使用者進入系統後，就按需求直接一一訂閱；資料庫系統也不儲存訂閱紀錄。

3. 在使用電子郵件的 SDI 服務模式中，資料庫系統根據讀者興趣檔和新資料做比對，把符合的新資料以電子郵件傳送給讀者；在使用 RSS 的 SDI 中，只要判別有無新資料，不需比對，且判別及傳送更新內容的動作可由分散世界各地的 RSS Reader 分攤處理，如此可減輕期刊資料庫集中處理的負擔，並分散傳送資料的網路頻寬資源。

4. 若要取消或修改訂閱內容，在使用電子郵件的 SDI 服務模式中，必須修改興趣檔，或依步驟取消訂閱；以 RSS 的方式則是直接刪除 RSS Feed 訂閱。

第四章　系統設計

本章節主要提出 MJFB（My Journal Filter Broker）系統之設計概念與架構，內容包括系統設計的目標與規劃、系統架構、系統後置處理、以及 RSS Feed 和 OPML 檔案的產生過程。

4-1　系統設計的目標與規劃

本研究針對 SDI 服務，以 RSS 技術為基礎，設計出 SDI 服務的新方式。在此提出一個符合個人化需求，電子期刊中間過濾機制的系統（MJFB），以期實現 SDI 個人化數位圖書館服務之精神。

系統旨意在於以網頁瀏覽器成為讀者與網路期刊資源的中介者介面。為了讓使用者能明確定義需求，借助個人化服務的概念，設計出挑選、過濾訂閱期刊的功能。而關鍵字對衡量一網站的價值和重要性，佔有頗高的依據[32]；因此系統設計以關鍵字去尋找新發表的相關期刊文章；在最後以 RSS 作為傳遞新資訊的工具之時，為了減少不明確、非必要的 RSS Feed item 數量，系統提供進一步檢索 RSS Feed 的功能；並且為了顧及瀏覽 RSS Feed 的便利，系統提供瀏覽排序的選項。

早期閱覽訂閱的 RSS Feed，大多需要安裝額外的 RSS Reader 軟體；現在則有 web 介面和電子郵件的方式可以選擇。本研究為了增加閱覽 RSS Feed 便利，將所有處理過程結束後的 RSS Feed 同時顯示在系統頁面上，每位使用者在登入系統之後，不需額外軟體，就可隨時看到屬於自己的所有訂閱內容。

利用 RSS 的 OPML 檔案，可把所有訂閱紀錄彙整輸出成為「My Journal」檔案。只要在 RSS Reader 軟體一次匯入，就可閱覽所有訂閱的內容。

4-2 系統架構

就軟體設計而言，系統以三層次的概念來實現，如圖 4-1。

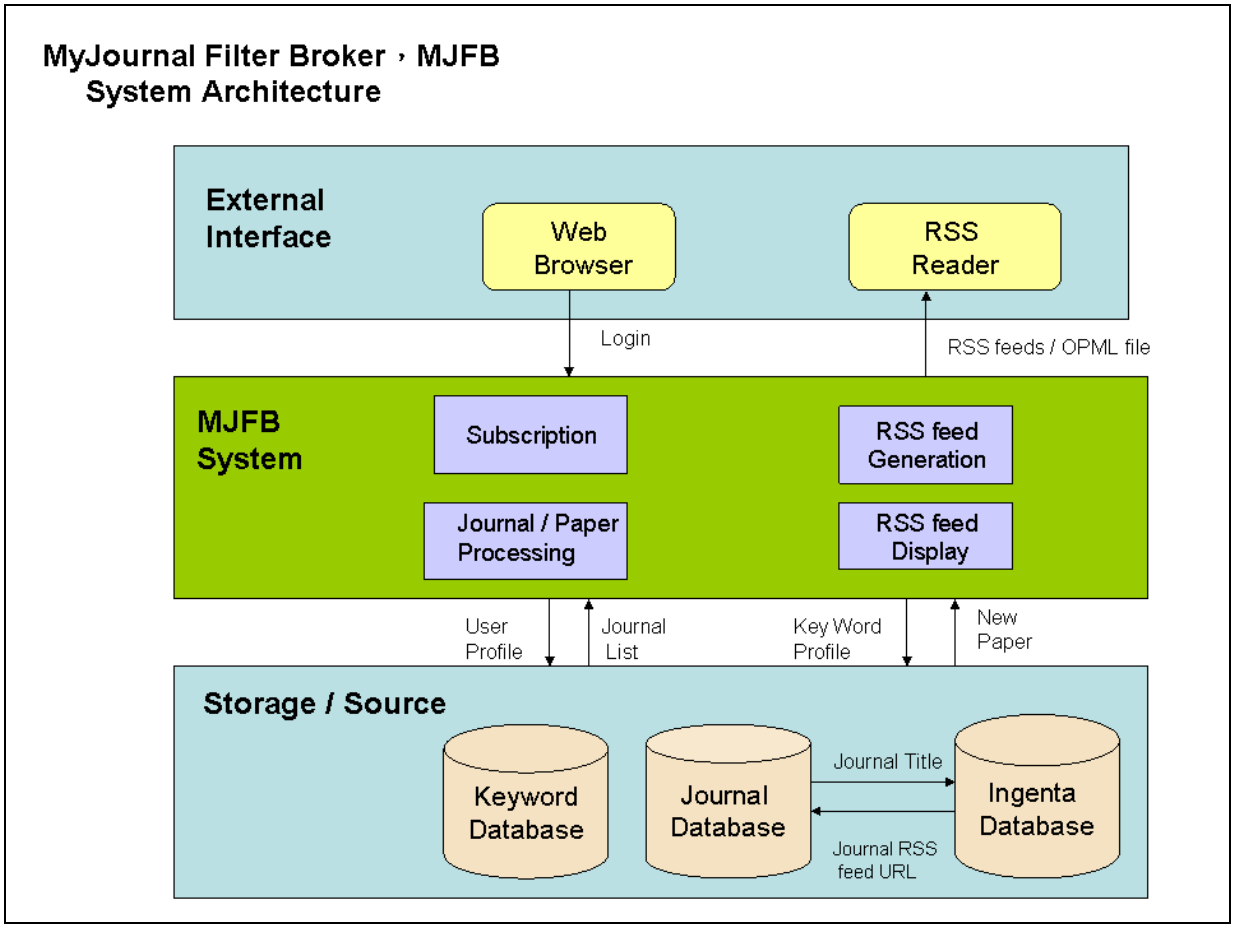

圖 4-1 系統架構圖

以下就系統架構的三層次概念分別說明：

1.外部界面（External Interface）：

系統初始，透過瀏覽器做個人帳號的登入，以達個人化服務的第一步身份驗證，進入系統。另外，使用者透過外部一般 RSS Reader 軟體，訂閱經系統處理過後的 RSS Feed 及 OPML 檔案。

2.系統處理（MJFB System）：

此為系統運作的主要部分。在使用者登入之後，依據個人 Profile 資料，

系統做了四個部分的處理：進一步增加或刪除訂閱、訂閱後的處理、產生 RSS Feed、並且讓所訂閱之 RSS 能同時在系統中顯示，而無須再透過額外的 RSS Reader 軟體和介面，增加系統的整體便利性。

系統處理層次的對象，主要分為期刊以及以關鍵字搜尋新期刊文章，二大主軸，本小節先描述 MJFB 內這二主軸的訂閱處理。

（一）期刊訂閱：

為了讓使用者明確定義需求，同時便利尋找所需期刊，系統設計以下的方法來訂閱期刊。

- 背景學院期刊

以 Rule_Based Filtering 的方法，在使用者登入系統後，判別其所歸屬學院，然後列出該使用者在學院中尚未訂閱之專業期刊，以供訂閱及後續處理之用。

- 歷史性期刊

以 Learning-Agent Technology 的方法，將圖書館自動化系統中的網頁瀏覽紀錄，推測作為讀者訂閱期刊的喜好、需求之依據，因此列出該記錄中尚未訂閱之期刊。

以此二種方法作為使用者個人的期刊訂閱依據，明確定義篩選出使用者的個人需求。

（二）關鍵字訂閱：

在期刊訂閱之外，系統提供以關鍵字搜尋特定內容的新期刊文章。在輸入欲查詢之關鍵字後，由於期刊文章更新的速度並非如新聞般時時在增加，系統設計以追蹤頻率時間定期搜尋新期刊文章。追蹤頻率以一星期一次為起始、接著二星期一次、一個月一次、以及二個月一次。並以 Constraint-Based Filtering 的方法，增加特定檢索條件（Profile）以減少最後 RSS Feed item 的數量。然後，設定瀏覽排序的條件，讓使用者在瀏覽 RSS Feed item 時，能更加便利順暢。

3.資料儲存和資料來源

依據使用者及所建關鍵字的 Profile 資料，儲存訂閱紀錄在系統資料庫，同時，藉助 Ingenta 資料庫提供相關資訊給系統做進一步處理。

（一）期刊和關鍵字資料庫

期刊資料庫儲存每位使用者的期刊訂閱紀錄。全部期刊依學院分類儲存，並儲存期刊的 ISSN 及 RSS Feed 的 URL 網址，詳細期刊資料則透過 URL 連結。歷史查詢期刊則從圖書館自動化系統中取出紀錄。

關鍵字資料庫儲存每位使用者所建立的關鍵字組，以及搜尋回來的新期刊文章相關資訊。

（二）Ingenta 資料庫

交通大學圖書館所購買的 Ingenta 資料庫包含了數千種各領域的電子期刊，多數期刊已經推行 RSS 訂閱服務；在搜尋介面上，除了期刊搜尋外，還提供以關鍵字搜尋期刊內容的功能。因此，本研究利用其搜尋介面及免費的 RSS Feed 為系統實作之內容。

4-3 訂閱後置處理

本小節描述圖 4-1 中，系統處理的訂閱後置處理功能部分，分為期刊和關鍵字二主軸。

1.期刊

在期刊訂閱中，使用者登入之後，學院期刊部分則需先做背景身分的判斷，得知使用者所屬學院。接著，從訂閱資料庫查詢已訂閱清單，系統分別與該學院及歷史查詢清單比對，若訂閱清單中並無紀錄，則從歷史查詢及學院期刊資料庫中列出全部期刊；若已有訂閱紀錄，則列出尚未勾選訂閱過的期刊。使用者的 profile 資料包括帳號所屬學院、以及之前訂閱的所有期刊。其程序以圖 4-2 表示。

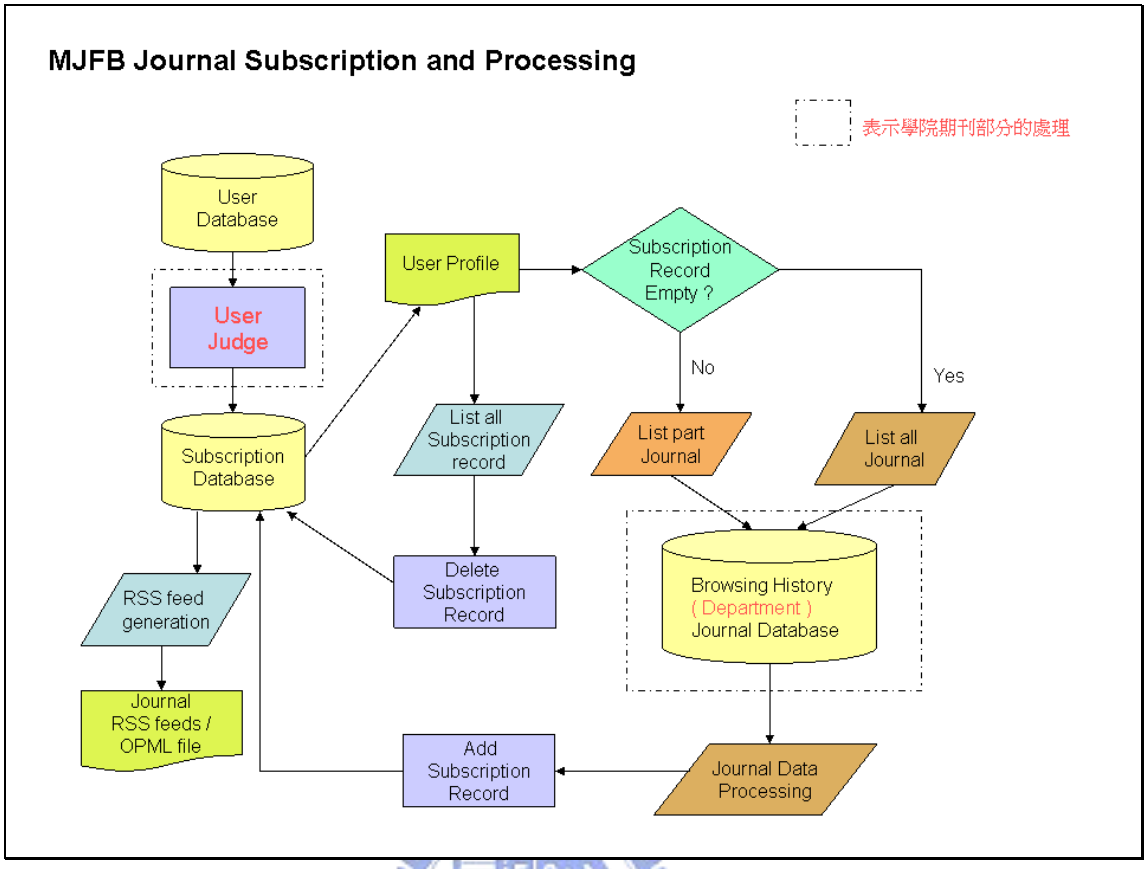

圖 4-2　期刊訂閱處理程序

對使用者而言，可從”觀看已訂閱之清單”的功能，得知已經訂閱過的期刊清單；也可在此刪除不想再訂閱的期刊。

訂閱資料庫僅紀錄期刊 ISSN（International Standard Serial Number）編號，詳細的期刊資料，包括 RSS Feed 網址，則以 ISSN 值在期刊資料庫（jurs）中查詢。在列出尚未訂閱的期刊時，使用者便可依此網址單筆在 RSS Reader 中訂閱。然而，本研究重點在於讓使用者能一次多筆勾選訂閱，並將每位使用者的訂閱紀錄存在資料庫中，最後，經由「一次全部訂閱」的功能，得到屬於自己個人的 OPML 檔。表 4-1 是儲存訂閱紀錄的資料庫表格結構。

表 4-1　資料庫表格 myjur 結構

表格名稱	myjur		說明	紀錄使用者訂閱的全部期刊	
欄位名稱	型態	大小	鍵值	Null 值	說明
No	int	100	Y	N	表格主鍵，自動編號
id	varchar	10			使用者登入身份證號
issn	varchar	9			訂閱期刊之 issn

另一資料庫表格紀錄期刊的詳細資訊，表格結構如表 4-2。

表 4-2　資料庫表格 jurs 結構

表格名稱	jurs		說明	紀錄每種期刊的詳細資訊	
欄位名稱	型態	大小	鍵值	Null 值	說明
No	int	100	Y	N	表格主鍵，自動編號
issn	varchar	9			期刊之 issn
name	varchar	50			期刊之名稱
jrss	varchar	100			期刊之 RSS Feed 網址

2.關鍵字處理

以關鍵字搜尋最新期刊文章的處理程序如圖 4-3，各步驟分述如下：

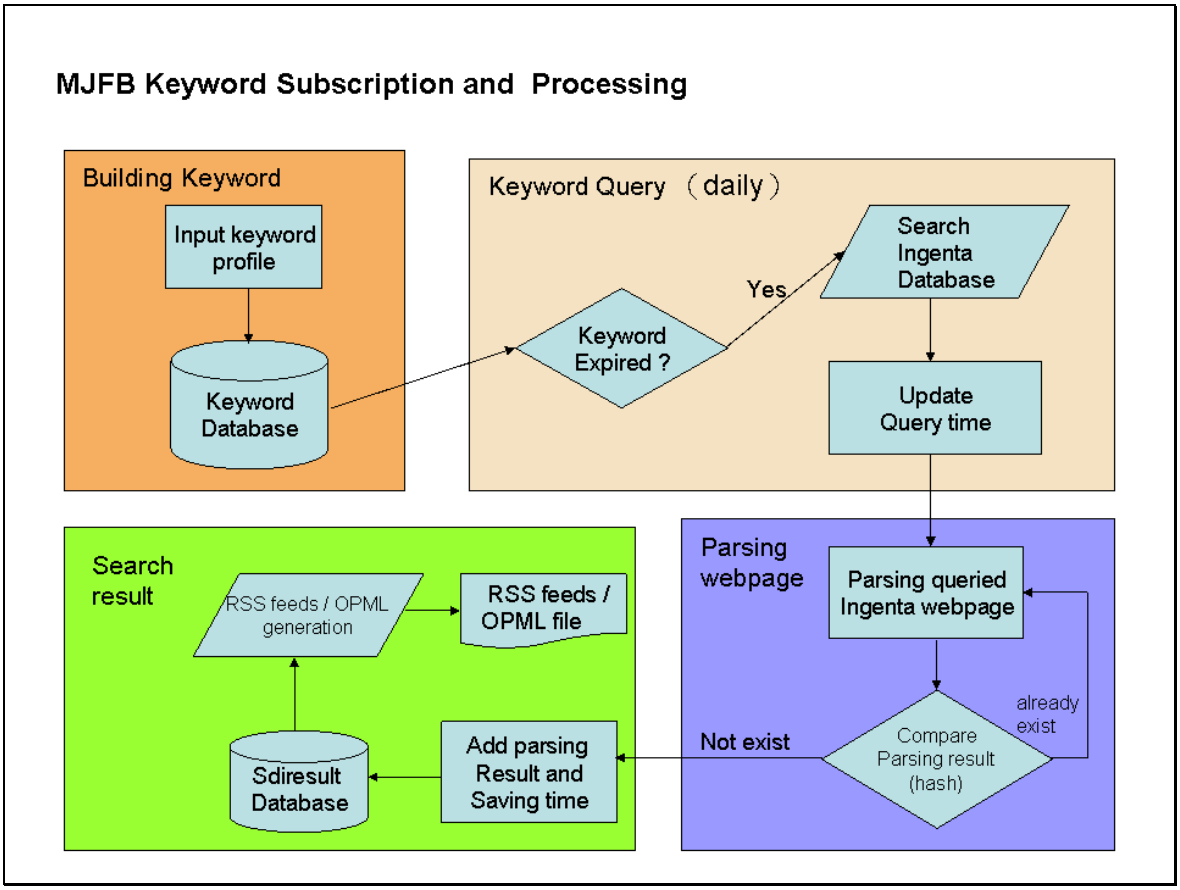

圖 4-3　關鍵字訂閱處理程序

（一）關鍵字建立

在系統介面中，輸入關鍵字內容、追蹤頻率、以及最後的排序和檢索條件。

（二）關鍵字查詢

系統每天自動判斷有哪些關鍵字已到達追蹤時間，將已到達追蹤時間的關鍵字搜尋 Ingenta 資料庫期刊內容，同時更新該關鍵字之最新時間；並紀錄此更新時間存進資料庫，以此更新時間為計算下次是否已達追蹤時間的依據。

所謂已到達追蹤時間，就是指現在系統時間減去此更新時間之差距，大於或等於使用者設定的追蹤時間間隔，亦即符合追蹤頻率時間。

搜尋期刊內容的程序每天執行一次，因此若關鍵字的建立時間在該日程序執行之後，要等到次日程序再執行之後，才會有結果。

（三）解析查詢結果頁面

搜尋完 Ingenta 資料庫之後，接著做網頁解析（Parsing）的動作，也就是分析查詢結果頁面上的內容該如何放入資料庫欄位中。由於本研究採用 Ingenta 資料庫的搜尋界面，而 Ingenta 搜尋結果頁面上的 export(plain text) 功能中，則以每頁 50 筆紀錄的規則詳列了搜尋結果的文字，成為網頁解析的依據。

因為搜尋結果頁面可能不只一頁，也就是說，判斷搜尋結果頁面中的查詢結果，若少於 50 筆，或現在頁面有 50 筆紀錄但下一頁之頁面格式不合搜尋結果之頁面格式，就表示已到達最後搜尋結果。

在每個搜尋結果的頁面中，很有規律的一筆筆紀錄了搜尋到的期刊文章，並且期刊文章之 metadata 資料也規則的列了出來。因此，可以利用程式解析期刊文章之 metadata 資料，並將之存入資料庫。

（四）儲存搜尋結果

由於 RSS 2.0 版本身的 item 數量並沒有限制，也沒有定義判別 item 的 tag；在儲存解析結果進入資料庫之前，為避免儲存到重複的結果，先將該筆紀錄內容結合成一長字串，再計算此長字串之雜湊值[33]，並以此雜湊值和資料庫中已存紀錄之雜湊值比較。若無碰撞（Collision）發生，表示資料庫中並無該筆紀錄，就存進資料庫，同時存進該筆紀錄之雜湊值和此刻的系統時間；若發生碰撞，表示資料庫中已有該筆紀錄，就繼續解析下筆紀錄並計算、比較其雜湊值。

此雜湊值是以安全雜湊演算法（Secure Hash Algorithm）計算出來，在 PHP 程式語言中，使用 sha1（）函數將字串轉換成一組 40 個 16 進位的字元，也就是上述的雜湊值[34]。儘管 sha1（）所使用的演算法之碰撞機率已經相當微小，若往後資料越加龐大而考慮到碰撞機率會發生，如 sha1（A）

=sha1（B），but A!=B，則可增加計算雜湊值的字串欄位，如期刊篇名。

4-4 RSS Feed 與 OPML 檔案的產生與呈現

本小節描述圖 4-1 中，第二層次系統處理內，RSS Feed 和 OPML 檔如何產生及呈現，分別期刊和關鍵字二主軸來介紹。

依 syndic8 aggregator 網站統計調查，目前 RSS 的使用分佈以 2.0 版數量及比例佔最多[35]，且 RSS Reader 軟體普遍支援，因此本研究在 RSS Feed 的製作上，皆以 2.0 版為主。

4-4-1 RSS Feed 的產生

1.期刊

圖 4-4 表示期刊 RSS Feed 的產生過程。分二個方向說明，一個以一般 RSS Reader 訂閱，另一個是在系統內部的 RSS Reader 顯示。

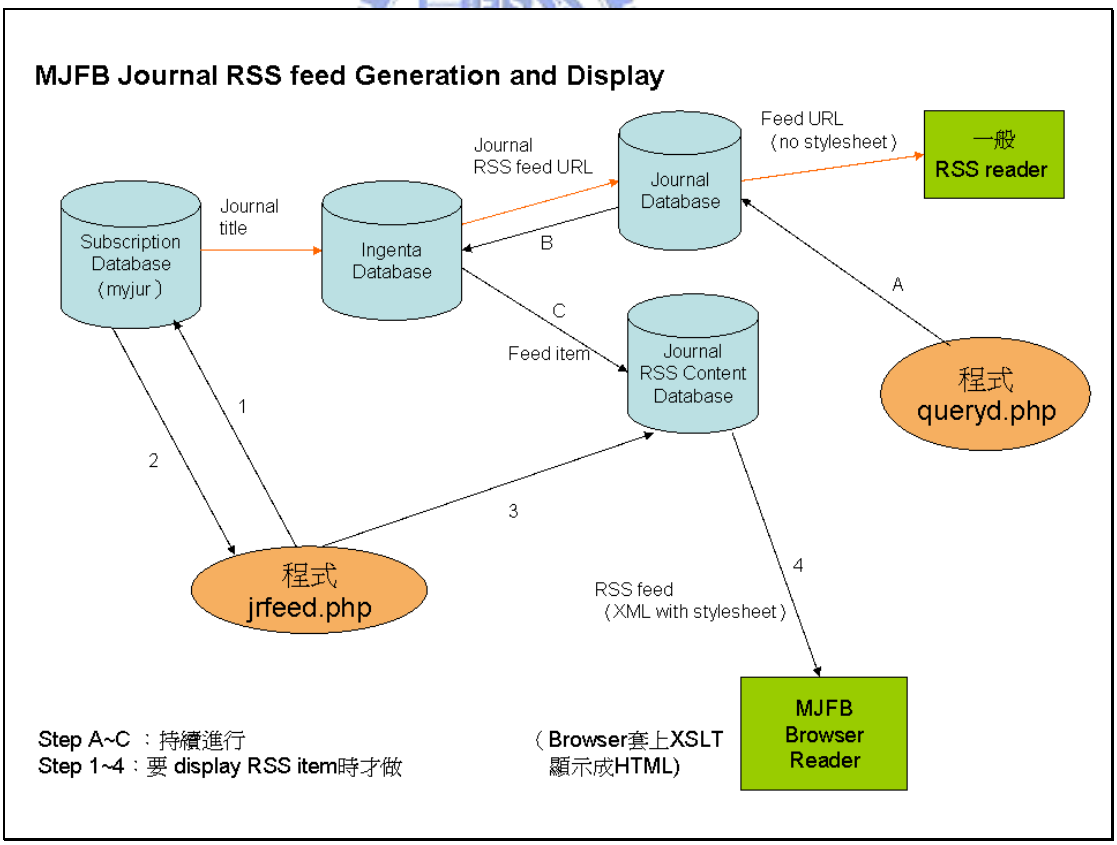

圖 4-4 期刊 RSS Feed 產生過程

（一）以一般 RSS Reader 訂閱

系統事先的處理會把期刊的 RSS URL 網址紀錄在期刊的詳細資料庫中。一登入系統，列出個人所有尚未訂閱之期刊，同時查詢相對應之期刊 RSS URL 網址，若要以一般 RSS Reader 軟體的方式來訂閱，就以此 URL 網址訂閱。

（二）在系統中以 Browser Reader 瀏覽

在圖 4-4 的步驟 A-C 中，系統程式每天執行一次，先找出所有已被訂閱的期刊 ISSN，依據資料庫儲存的期刊 RSS Feed URL 網址，向 Ingenta 資料庫下載期刊 RSS Feed items，並透過 PHP 公用免費程式庫 lastRSS（附錄一），解析找到的 RSS Feed items 內容，存進資料庫。

為避免儲存到重複的內容，在儲存之前會先將同本期刊之前的 Feed item 刪除。在資料庫處理上，新增儲存的資料可能會被放在這些資料被刪除的空間內，而造成同本期刊的 Feed item 散落在資料庫各處的可能。因此，每次下載、儲存 Feed item 內容時，同時依其在網頁上的次序從 0 開始給予編號，並將編號紀錄在資料庫中。如此，Feed item 的顯示順序就可以和 Ingenta 資料庫上原始順序相同。並且，資料庫在存取同一期刊之 Feed item 時，速度會較快。

當從系統頁面去點選所訂閱期刊時，就執行圖 4-4 中步驟 1-4。首先查詢個人訂閱了哪些期刊，再去期刊的 RSS 內容資料庫索取 RSS Feed 內容。最後，顯示在系統頁面上。

2. 關鍵字

圖 4-5 描述將關鍵字搜尋新文章的結果，製成 RSS Feed 的過程。

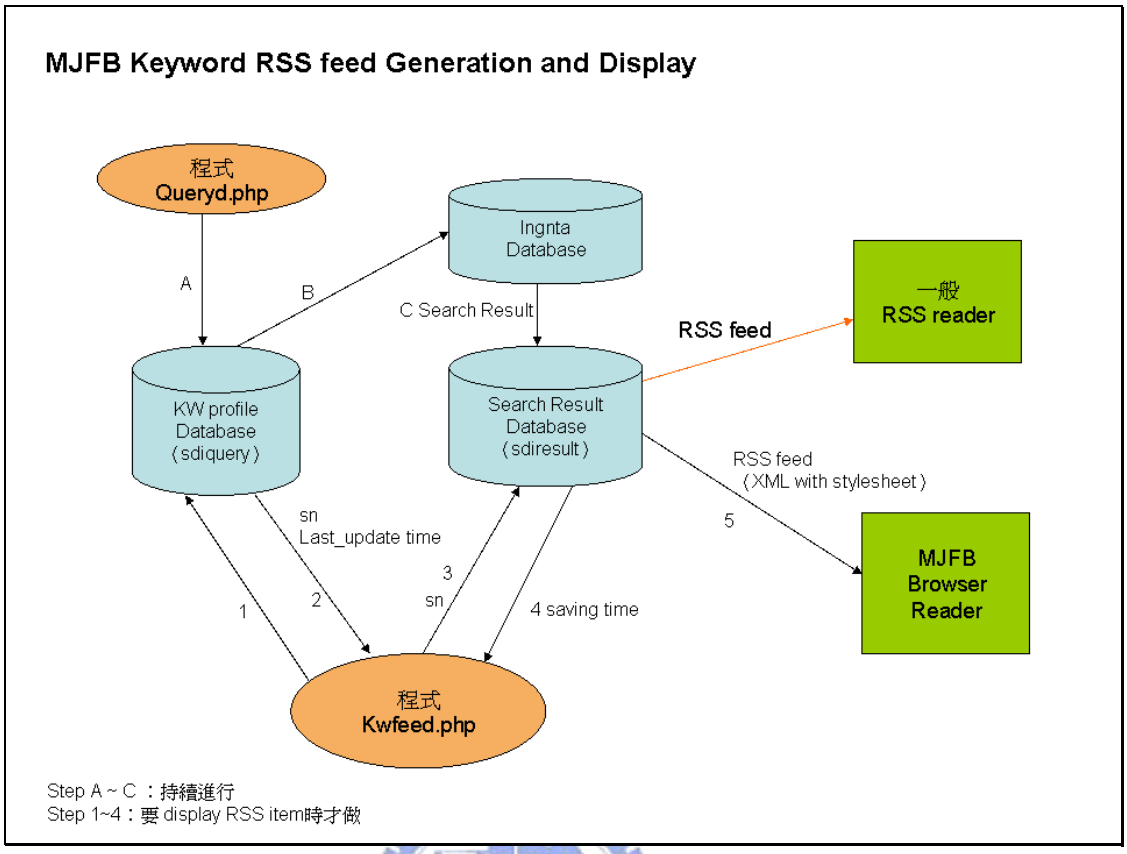

圖 4-5　關鍵字 RSS Feed 產生過程

關鍵字的 RSS Feed URL 網址生成規則如下:
<網站的網址 (程式自動偵測)> / kwfeed.php / <關鍵字的 sn 序號>，也就是在確認增加該關鍵字後，系統資料庫將每位使用者的每組關鍵字會給予一 sn 序號。

圖 4-5 中，步驟 A-C 代表系統程式對關鍵字的判斷和搜尋每天執行一次，只要有關鍵字到達追蹤時間就會有新的搜尋結果。

當使用者登入要產生該關鍵字之 RSS Feed 時，執行步驟 1-4，先找出此組關鍵字之 sn 編號。因為 RSS 是要傳遞最新的資源，所以依 sn 在資料庫中找出搜尋結果之儲存時間大於（後於）關鍵字上次更新時間的紀錄，就是表示自上次追蹤時間後所更新的紀錄。

若無更新的期刊文章，或尚未到達追蹤時間，則該關鍵字之 RSS Feed 便無 item 顯示。

4-4-2 RSS Feed 的顯示

如圖 4-4 與圖 4-5 中，在系統頁面顯示期刊的 Feed item 時，因為 RSS Feed 檔案其實也就是 XML 檔案，因此可以利用 XSLT 轉換技術，先將資料庫各欄位內容轉換為帶有 stylesheet 樣式訊息的 XML 檔案。當瀏覽器讀到此檔案，便會根據 XSLT 的樣式內容，轉換該 XML 內容成 HTML，然後呈現在系統網頁上。

XML 檔案（也就是 RSS Feed 檔案）中的樣式訊息，就是指在 XML 宣告後要加入一行 stylesheet 的指引：<?xml-stylesheet type="text/xsl" href="MY-STYLE-SHEET.xsl"?>，在本研究中，此訊息指引如下：<?xml-stylesheet type="text/xsl" href="'.$baseurl.'/rss-stylesheet.xsl"?>，rss-stylesheet.xsl 檔案如附錄二。

4-4-3 OPML 檔

OPML 檔表示個人使用者全部訂閱紀錄。當選按「全部一次訂閱」的功能時，系統去抓取該使用者所有訂閱期刊和關鍵字的 RSS Feed URL 網址，依 OPML 規格動態產生一文字檔，使用者可以將之儲存檔名為「all-帳號.opml」形式的檔案。也就是每個使用者都可以得到一個屬於自己的 OPML 檔案。圖 4-6 為本研究產生的一 OPML 檔，內容分為 journal 與 keyword 二部分，每個 outline 標籤表示一個 RSS Feed 的訂閱網址。

此 OPML 檔案是動態產生，當選按此功能時，程式察看使用者訂了哪些期刊和關鍵字組。期刊的 RSS URL 網址可直接從資料庫取得，關鍵字的 RSS URL 則依 sn 值及系統環境變數動態產生。

```
<?xml version="1.0" encoding="UTF-8"?>
<opml version="1.1">
        <head>
                <title>My Journal and Keywords</title>
                <dateCreated>Tue, 03 Jan 2006 06:58:27 +0000</dateCreated>
                <dateModified>Tue, 03 Jan 2006 06:58:27 +0000</dateModified>
        </head>
        <body>
    <!-- journal -->
        <outline title="Journal">
            <outline text="Advances in Computational Mathematics" type="rss"
                xmlUrl="http://api.ingentaconnect.com/content/klu/acom/latest?format=rss" />
            <outline text="Applied Artificial Intelligence" type="rss"
                xmlUrl="http://api.ingentaconnect.com/content/tandf/uaai/latest?format=rss" />
            <outline text="Archives and Museum Informatics" type="rss“
                xmlUrl="http://api.ingentaconnect.com/content/klu/armu/latest?format=rss" />
            <outline text="Health Libraries Review" type="rss“
                xmlUrl="http://api.ingentaconnect.com/content/bsc/hlr/latest?format=rss" />
            <outline text="Information &amp;#38; Communications Technology Law" type="rss“
                xmlUrl="http://api.ingentaconnect.com/content/routledg/cict/latest?format=rss" />
            <outline text="Quaerendo" type="rss“
                xmlUrl="http://api.ingentaconnect.com/content/brill/qua/latest?format=rss" />
        </outline>
    <!-- keyword -->
            <outline title="Keyword">
              <outline text="Keyword: digital library" type="rss"
                xmlUrl="http://localhost:8080/rss/rss/kwfeed.php/3" />
            </outline>
        </body>
</opml>
```

圖 4-6　OPML 檔案範例

第五章　系統實作與驗證

本章節描述研究系統的實作結果，先介紹系統實作環境與開發工具，其次敘述系統功能，並且針對之前章節的討論，展示實作的結果驗證。

5-1　系統環境

本研究系統的實作環境如圖 5-1，系統作業平台為 Windows 2000 Server SP4，開發工具採用免費的 AppServ 套件，版本是 2.2.5 版，包含 Apache 伺服器 2.0.55 版、MySQL 資料庫 5.0.16-nt 版、以及 PHP 程式語言 5.5.1 版。管理工具 phpMyAdmin，版本為 2.6.4-pl4 版。

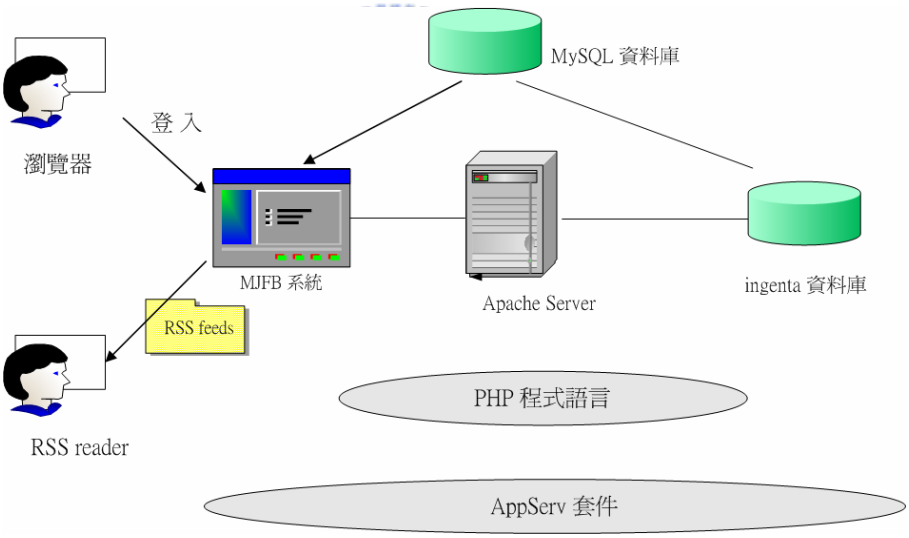

圖 5-1　系統實作環境

圖 5-1 中，使用者以瀏覽器登入 MJFB 系統，系統則持續在後端向 Ingenta 資料庫搜尋、更新資料，並存進 MySQL 資料庫。所訂閱的 RSS Feed 除可在系統同一頁面上顯示外，也可透過使用者自己的 RSS Reader 軟體閱覽。而測試本系統的 RSS Reader 軟體，採用免費的 SharpReader 軟體，版本為 0.9.5.1。

5-2　系統功能

本小節針對個人化需求與 RSS 技術本身的特性，設計出如圖 5-2 的系統功能，使用者在遠端透過瀏覽器登入、身份驗證過後，進入 MJFB 系統。在提供訂閱與刪除訂閱功能之外，系統同時也是一個 RSS Reader 軟體，使用者無須再另外透過額外的 RSS Reader 軟體來閱覽，增加便利；系統輸出的部分，除了個別期刊、關鍵字的訂閱外，利用 RSS 本身的 OPML 格式，產生每位使用者自己的 OPML 檔，達到個人化的服務效能。

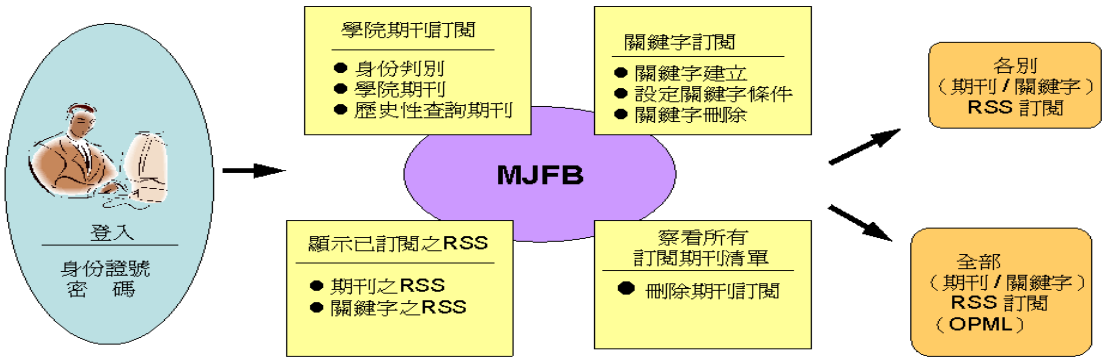

圖 5-2　系統功能圖

圖 5-3 是登入系統後的執行畫面，在判別身份之後，畫面右邊首先是期刊訂閱區域，列出了個人所屬學院與歷史性查詢期刊中尚未訂閱之所有期刊，每頁 4 筆期刊供挑選，並可選按每筆期刊後面的 RSS 圖案單筆訂閱；畫面右邊下方則是關鍵字組的訂閱區域，提供關鍵字內容的輸入，以及對該關鍵字的進一步條件設定，可選按每個關鍵字組後面的 RSS 圖案單筆訂閱，也可以刪除不想再訂閱之關鍵字組。

在「觀看所有訂閱期刊清單」的功能中，除了列出目前已經訂閱之所有期刊，

也可刪除不想再訂閱之期刊。「全部一次訂閱」的功能則是將所有訂閱之 RSS Feeds，以 OPML 檔的形式儲存。

畫面左邊上下二個頁框則是系統本身提供的 RSS Reader 功能，在此可以立即顯示所訂閱的 RSS Feed 內容。

圖 5-3　系統執行畫面

5-3　實作結果

本小節針對系統研究開發，展示執行結果；並且對前面章節的問題討論，以實作結果來達成驗證的目的。

5-3-1　RSS Reader 驗証

1.單筆期刊訂閱

系統畫面上所列出的全部期刊，不論是否已經勾選訂閱，都可一筆一筆選按訂閱。圖 5-4 是單一訂閱「Computer Journal」的結果。

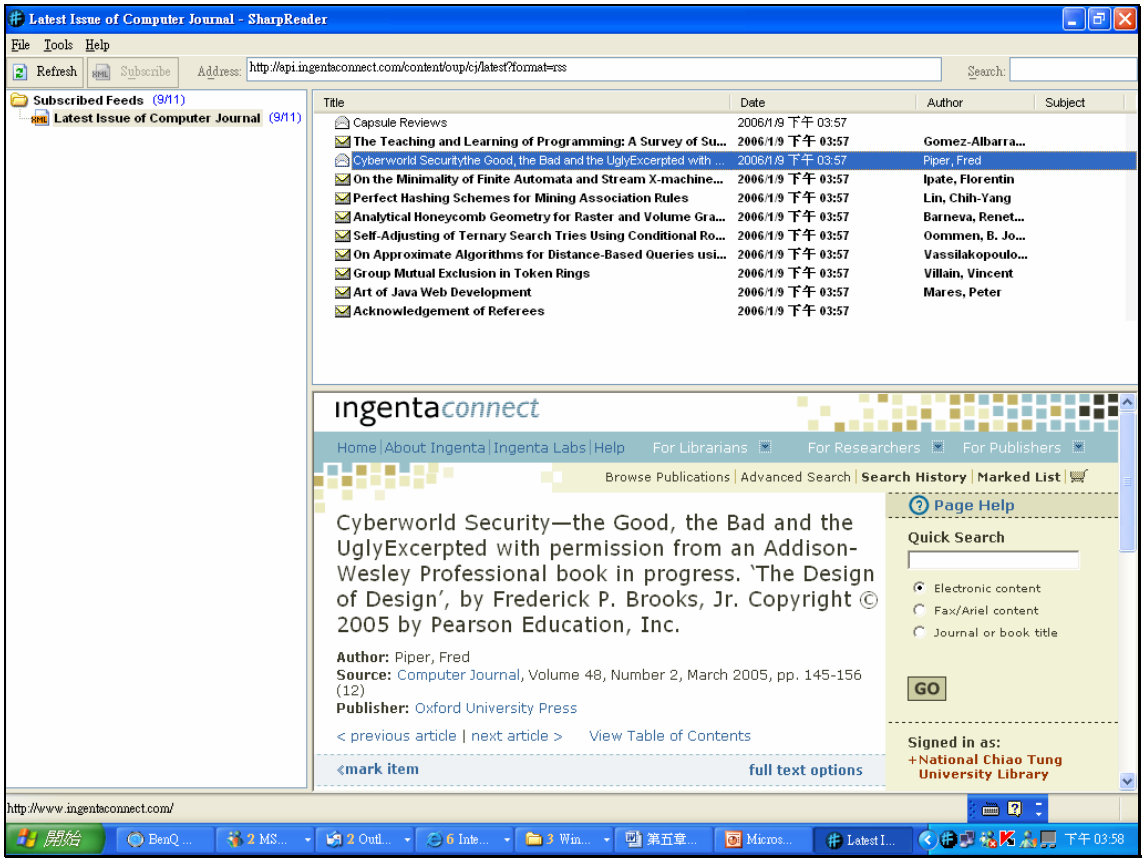

圖 5-4　各別期刊訂閱結果

2.關鍵字單筆訂閱

所建立的關鍵字組，可單筆訂閱。圖 5-5 為關鍵字「RSS」的訂閱結果，在資料庫中該組關鍵字之 querysn 為 11，因此 RSS Feed 網址為 http://140.113.39.69:8080/rss/kwfeed.php/11，瀏覽次序則設定以日期為順序。

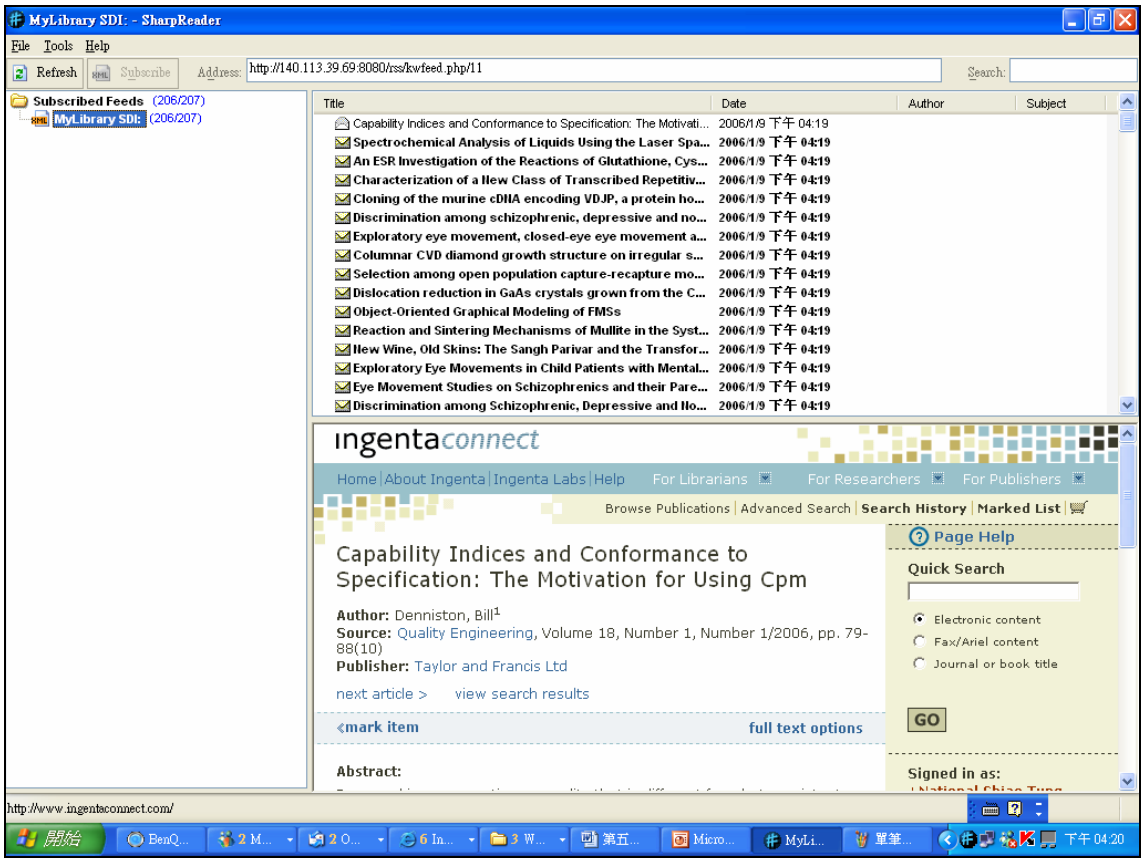

圖 5-5　單筆關鍵字訂閱

3.OPML 匯入

圖 5-6 是 OPML 檔的匯入結果，包括期刊和關鍵字的所有訂閱，並且可以看出 OPML 檔名為「all-帳號.opml」的形式，代表該帳號個人的全部訂閱紀錄。

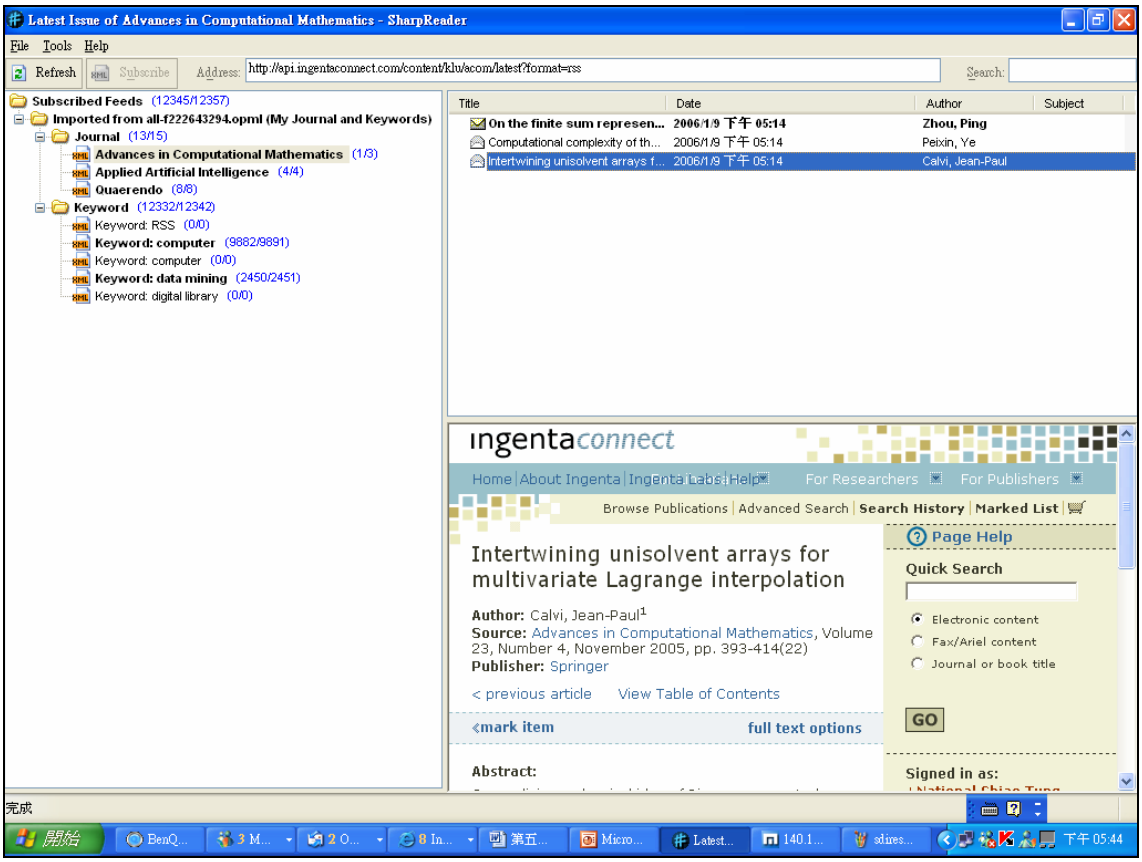

圖 5-6　OPML 檔匯入

5-3-2　關鍵字查詢結果

圖 5-7 是將關鍵字「digital library」的查詢結果儲存在資料庫中的結構，該組關鍵字 querysn = 3，每筆查詢結果有獨一的 hash 值，「install time」欄位表示該筆查詢結果儲存時的系統時間，其餘則是所查詢到的該篇期刊文章之相關資訊。

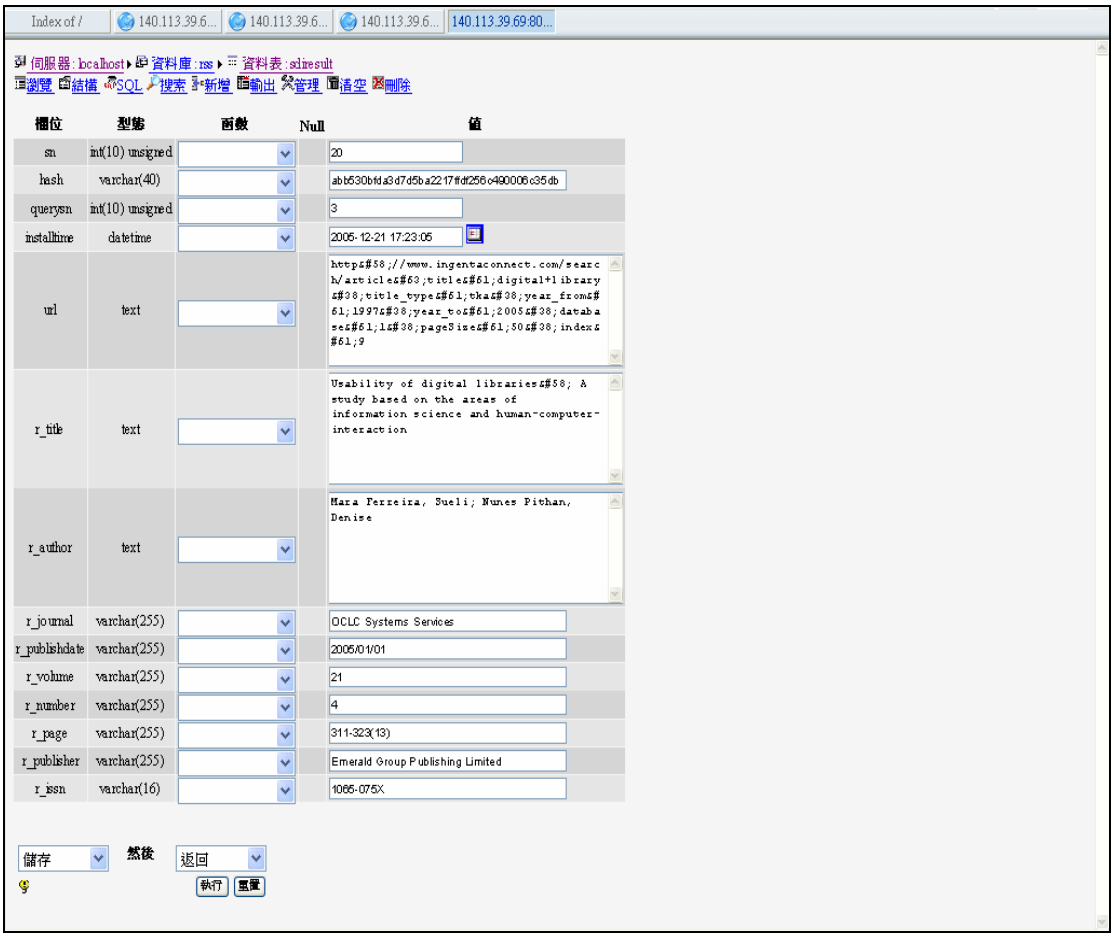

圖 5-7　關鍵字查詢結果儲存內容

實際執行之後，以關鍵字「data mining」為例，原先搜尋到 2451 筆，以期刊名稱「Plant Mocular Biology」為最後檢索條件時，則減少到 14 筆，如圖 5-8。

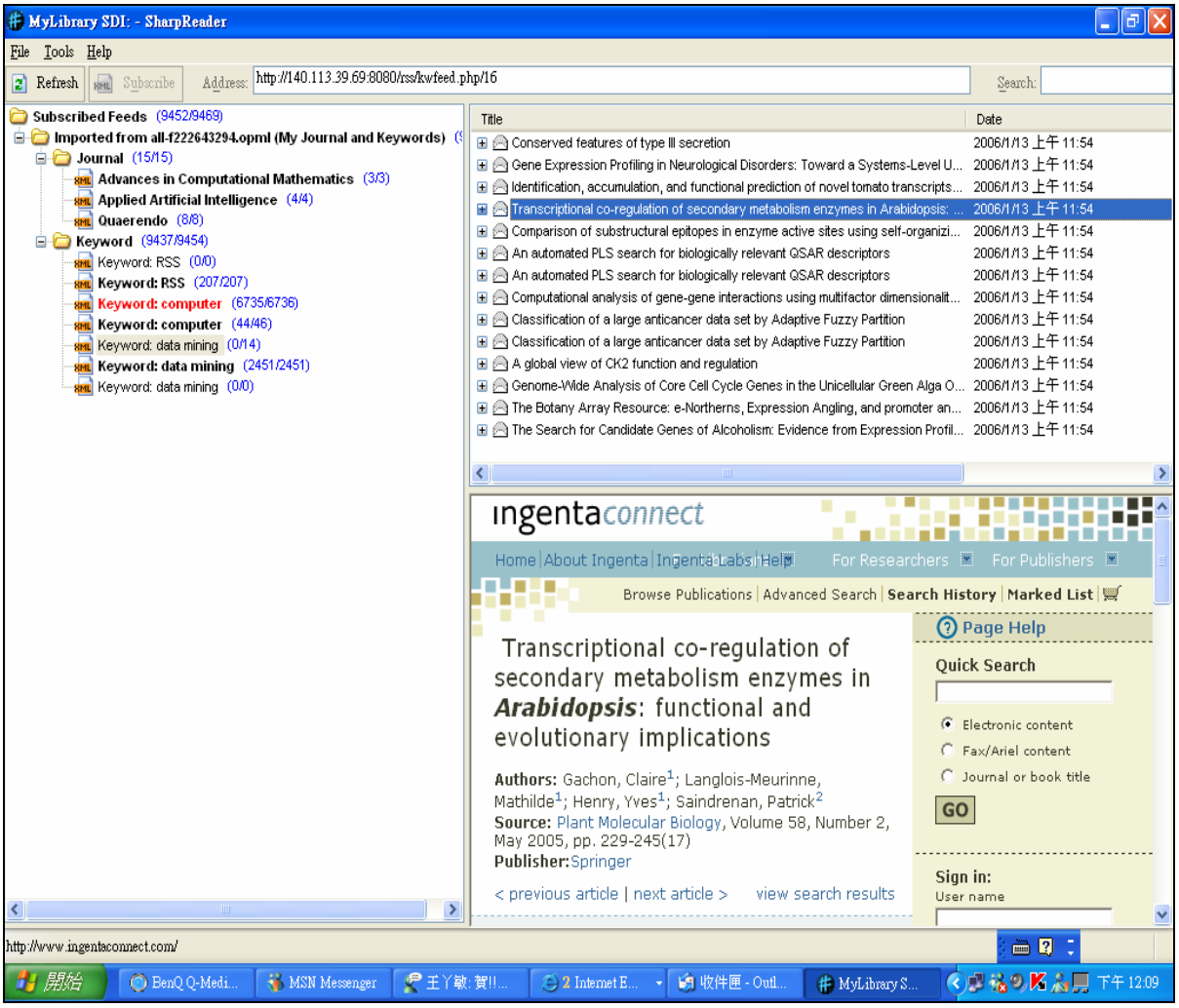

圖 5-8　關鍵字搭配檢索條件的搜尋結果

圖 5-9 則是以關鍵字「computer」搜尋之後，將期刊文章依日期排序，以方便閱覽。

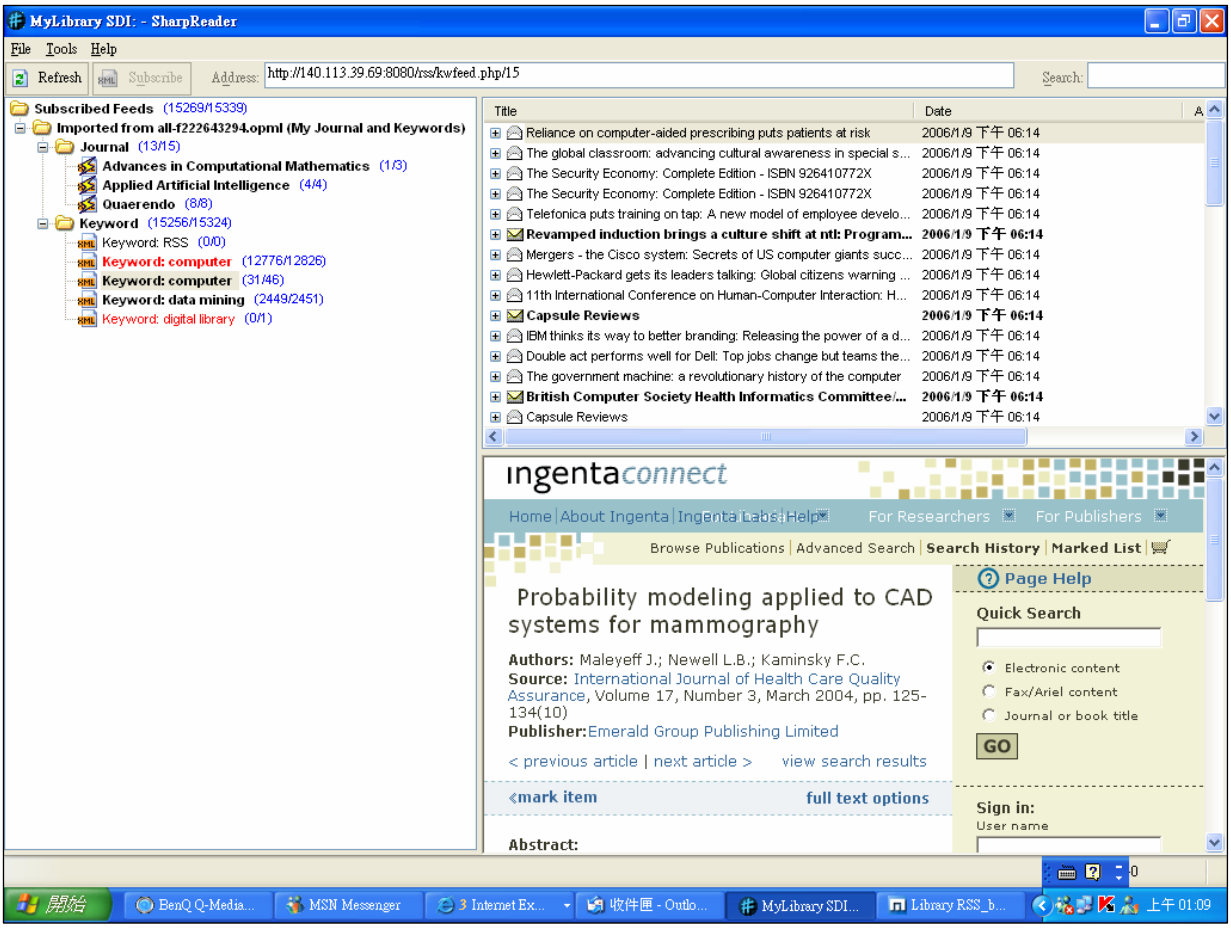

圖 5-9　關鍵字搜尋結果按日期排序

5-3-3　系統內部的 RSS Reader

圖 5-10 畫面左邊二個頁框是系統本身的 RSS Reader，上頁框是所有訂閱期刊和關鍵字的 RSS Feed 清單，點按任一 RSS Feed 後，該 Feed 內容列於下頁框。若再點按任一 Feed 內容，則直接連結至該篇文章之網頁，並以另一新視窗呈現，如圖 5-11。

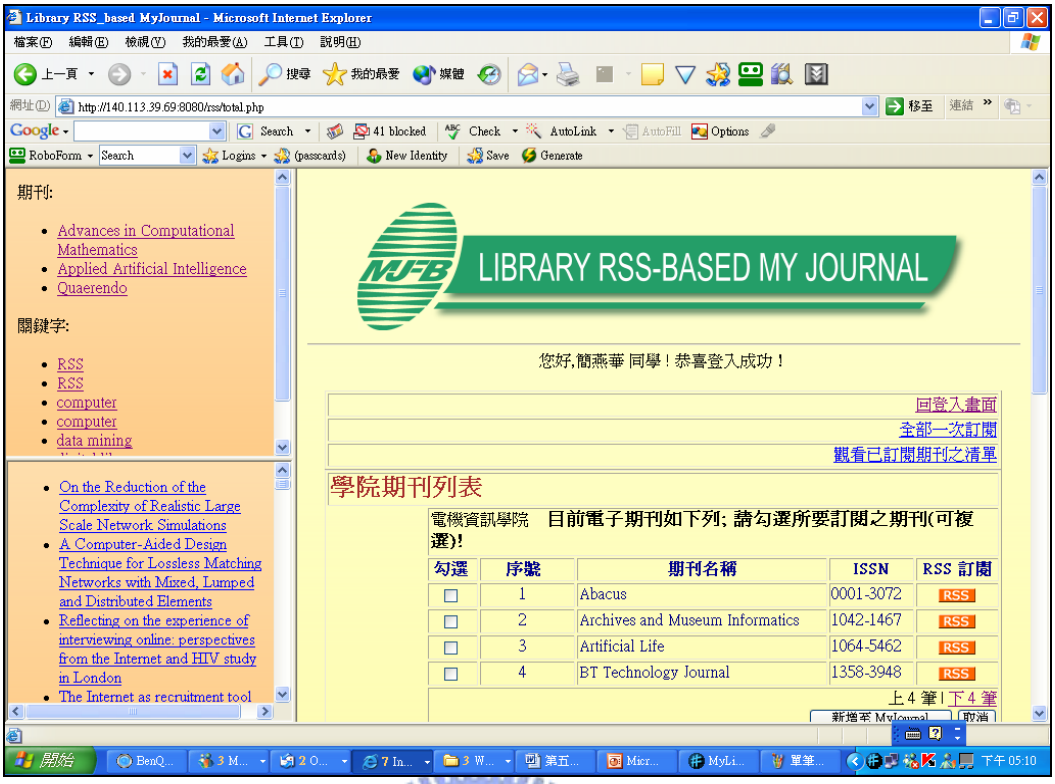

圖 5-10　系統的 RSS Reader

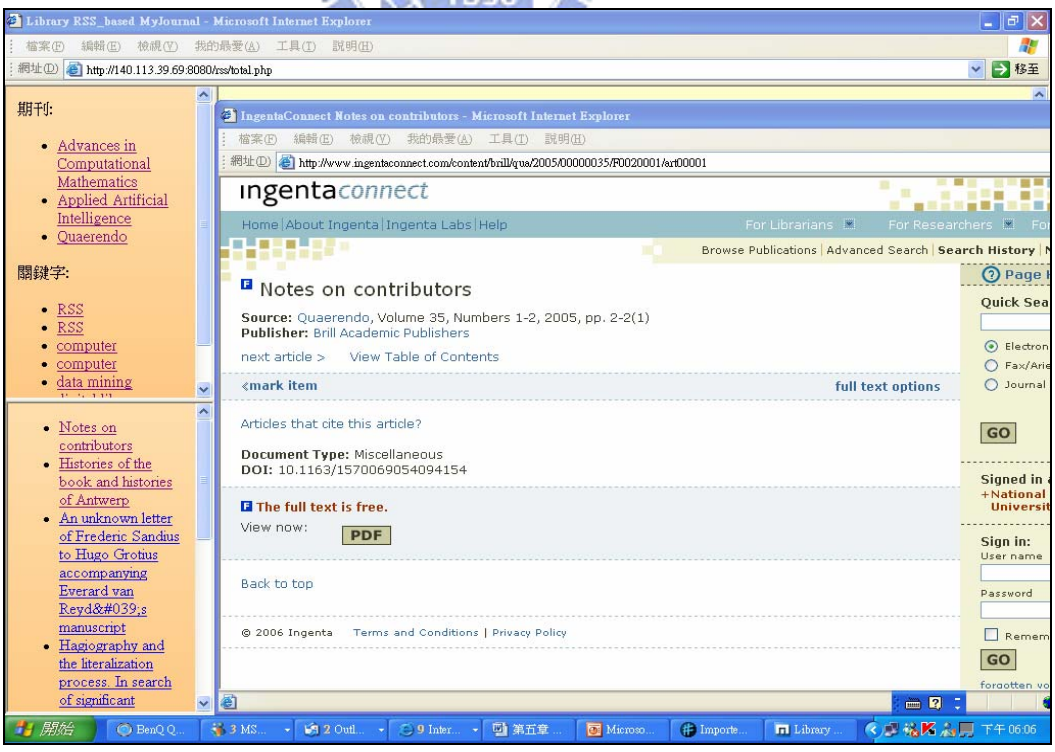

圖 5-11　系統 RSS Reader 中 Feed 的連結

5-3-4 實作小結

本章針對之前分析所設計的系統功能，作一結果驗證。MJFB 作為 SDI 的新模式，經由實作驗證，可以解決現行電子化 SDI 系統的問題，實現個人化數位圖書館的一部份。

依據現行電子化 SDI 系統的使用狀況，觀察出下列的現象：

1. 根據醫學資料庫 MEDLINE 的測試結果，即使同一電子資料庫中使用相同的檢索策略，並非所有 MEDLINE SDI 服務會獲得相同的結果。每家系統提供各自的服務功能，而有不同的特色[36]。

2. 在電子化系統發展中，便利性和資訊過量二者間，存在難以平衡的狀況。一方面要自動通報新資訊，一方面要花時間去處理這些新資訊。而沒有人願意為了跟上新發展要每天做複雜的事情。因此，普遍而言，現今每人的電子郵件信箱內都充滿（未閱讀的）郵件[3]。

3. 對於新資訊的量，使用者想要的是「更少」，而不是「更多」。

4. 對通報服務系統廠商而言，提供完整、精確、彈性的解決方式，是項艱鉅的任務。

5. 早期圖書館員的價值，在於協助調整 SDI 的檢索策略。隨著越來越多的電子化系統出現，SDI 從需要館員中介支援，轉變到使用者可以自行掌控的服務[36]。設定檢索策略若缺乏資訊專家支援，就要花相當的努力來調整。使用者不會樂於調整他們的興趣檔；而廠商也很少主動提供個人化的興趣檔管理工具。因此，興趣檔的管理與設定是否恰當，值得注意。

在 MJFB 系統中，期刊通報廠商只需負責提供 RSS Feed 內容，不但以完整、精確的方式，提供新資訊通報的功能，也沒有各家服務功能比較不同的問題。對使用者而言，能獲得更少、更精確的新資訊的量；也不需去調整維護興趣檔。不但減少電子郵件信箱中郵件的量，MJFB 本身提供的 Web_Based RSS Reader 功能，也讓使用者在閱覽新資訊時更加便利。

第六章　結論與建議

6-1　研究結論

本篇論文以個人化服務的精神，提出以 RSS 技術導入圖書館 SDI 服務，成為數位圖書館中 MyLibrary 的一種服務，期望改進現行電子期刊採用電子郵件通報新知的缺點。

系統實作結論如下：

1. RSS 能取代電子郵件傳遞新訊息的功能，並且目前沒有電子郵件的重大問題，如病毒和垃圾郵件。

2. RSS 提供者或訂閱者，都可設定內容更新的時間頻率，如何維持內容同步更新，又不影響網路流量，將視經驗與需求而定。

3. 以個人化的技術概念，可依不同需求篩選 RSS 的訂閱資源，加快搜尋訂閱 RSS 的速度；配合檢索條件，能減少 RSS Feed 中 item 的數量；同時，設定排序條件，增加閱覽時的便利。

4. 利用 OPML 檔，可一次將所有 RSS 訂閱清單匯入 Reader，節省單筆訂閱的時間；而每位讀者可從 MJFB 系統獲得自己的 OPML 檔，達到個人化的目的。

5. 建置 Web-Based 的 RSS Reader，可省去再透過其他軟體閱覽的步驟。

6-2　未來研究建議

對圖書館而言，只要有尋求新資訊以跟上研究領域發展的需求存在，就會產生大量 SDI 服務，以便能清楚定義新資訊，並提供符合定義的新資訊[5]。本研究提出改良的 SDI 服務方式，保留 SDI 的服務精神，又避免以往 SDI 的問題。

在研究與系統設計過程中，我們考慮過一些更深入的問題，在此整理條列出來以供往後研究參考之用：

1.增加篩選期刊訂閱的方法

除了本研究提出的篩選訂閱期刊方法外，還可利用其他方法來做篩選，例如整理出熱門期刊的排行、建立推薦期刊清單等不具個人意義的方法，可避免個人隱私外洩的風險；又如由使用者紀錄自己個人喜好的期刊，也可以先過濾出使用者的需求。

2.關鍵字搜尋範圍與頻率

在關鍵字查詢上，本研究目前是透過 Ingenta 資料庫所提供的期刊內容搜尋界面所找到的結果來驗證實作。在往後的應用或研究中，可以擴大搜尋界面，如 Google、Yahoo、MSN 等網站，都已經提供 RSS 搜尋訂閱服務，並且內容也不局限在圖書館期刊上[37]。

由於每份期刊的出刊時間不一，因此，本研究設計中，關鍵字的追蹤時間頻率以二星期為起始。若是在其他內容的 RSS 搜尋訂閱服務上，可依需求和內容更改追蹤時間頻率，例如以一天為起始單位。

3.追蹤讀者興趣

本研究中的歷史查詢期刊紀錄主要來自圖書館自動化系統中的資料庫。另外，還可以利用系統中 Profile 的資料，以類似資料探勘的方法，找出讀者進一步的相關資訊。

RSS 技術從簡化新聞發佈的角度出發，到現在發展成為推播新資訊的一種方法，歷程不過幾年的時間。而一個完整的數位圖書館需要全面性的考量和許多資訊技術的應用配合。本研究提出 RSS 在圖書館電子期刊的應用，為 SDI 提出一個新的方式，解決以往 SDI 電子化系統的問題。如此，只是推動數位圖書館的一小步，相信往後 RSS 技術不但在數位圖書館，還有其他更多面向，都能有所應用。可以預見，控制 RSS 訂閱數量與過濾內容，減少不必要的閱覽時間，將是值得研究的課題。

參考文獻

[1] Paul Martin and Mike Metcalfe, “Informing the knowledge workers”, Reference Services Review , 29, 4, pp. 267, 2001.

[2] 陸敏，「專題選粹服務簡介」，中國圖書館學會會務通訊，55 期，頁 14-15、5，民國 73 年 3 月。

[3] Ulla de Stricker, “Keep Me Posted…but not too Much：Challenges and Opportunities for STM Current-awareness Provider”, Searcher, 10:1, pp. 52, 2002.

[4] 吳其勳，「RSS 下一代網路內容傳遞關鍵技術」，iTHome，202 期，頁 6，民國 94 年。

[5] Luhn,H.P., “A Business Intelligent System”, IBM Journal of Research and Development, 2, pp. 314-419, 1958.

[6] Mauerhoff,G.R., “Selective Dissemination of Information”, Advances in Librarianship, 4, pp. 25-, 1974.

[7] Lawrence G. Mondschein, “SDI Use and Productivity in the Corporate Research Environment”, Spec Libr, 81(4), pp. 265-79, 1990 Fall.

[8] Charles H.Davis, and Peter Hiatt, “SDI is for people”, Library Journal, 96, pp. 3573, 1971.

[9] Kathleen Strube and Carol M. Antoniewicz, “Manual Selective Dissemination of Information from Journal Holdings of an Academic Medical Library”, Medical Reference Services Quarterly, vol.7（1）, 1988.

[10] 魏瓊釵，「資訊服務人工系統之探討」，中國圖書館學會會報，34 期，頁 33-79，民國 71 年 12 月。

[11] 李德竹，「美國的科學報導」，美國圖書館業務，遠東圖書公司，台北，頁 152-153，民國 61 年。

[12] Deardorff TC、Garrison AO., “Developing an Automated Current Awareness Program Using Microcomputers and Electronic Mail”, Technical Services Quartly, 14, 4, pp.1-12, 1997.

[13] 卜小蝶，「淺析個人化服務技術的發展趨勢對圖書館的影響」，成功大學圖書館館刊，2 期，頁 63-73，民國 87 年 10 月。

[14] 張瀚仁，「個人化技術對虛擬社群發展之影響」，國立政治大學，碩士論文，民國 89 年。

[15] 施能仁等，「個人化技術應用於網路書店」，社會科教育研究，第五期，頁 227-250，民國 89 年。

[16] 余明哲，「圖書館個人化館藏推薦系統」，國立交通大學，碩士論文，民國 92 年。

[17] Paul F. Nunes and Ajit Kambil, “Personalization ? No Thanks”, Harvard Business

Review, 79, 4, pp. 32-34, 2001 April 01.
[18] 曾元顯，「數位圖書館中的個人資訊空間之構想」，中國圖書館學會會報，56 期，頁 69-75，民國 85 年 6 月。
[19] 柯皓仁，「電子圖書館實施經驗談—以交通大學圖書館為例」，書苑，47 期，頁 10-33，民國 90 年 1 月。
[20] Suzanne Cohen,et al., “Personalized Electronic Services in the Cornell University Library”, D-Lib Magazine, 6, 4, 2000 April.
[21] Ken Winter, “MyLibrary Can Help Your Library”, American Libraries, 30, pp. 65-67, 1999 August.
[22] http://blogs.law.harvard.edu/tech/rssVersionHistory, RSS History, RSS at Harvard Law, available at 2006.01.07.
[23] http://www.webreference.com/authoring/languages/xml/rss/intro /2.html, RSS Syndication and Aggregation, available at 2006.01.07.
[24] 黃天賜，「交錯雜綜的 RSS 演進史」，iTHome，202 期，頁 66，民國 94 年。
[25] http://blogs.law.harvard.edu/tech/advisoryBoard, Advisory Board, RSS at Harvard Law, available at 2006.01.07.
[26] Ellen Finkelstein,Syndicating Web Sites with RSS Feeds for Dummies, Wiley Publishing, Inc., 2005.
[27] Ben Hammersley, Developing Feeds with RSS and ATOM, O’reilly, 2005.
[28] Ben Hammersley, Content Syndication with RSS, O’reilly, 2003.
[29] http://www.intertwingly.net/slides/2003/rssQuickSummary.html, RSS Quick Summary, available at 2006.01.07.
[30] http://www.pgp.net/pgpnet/pgp-faq/pgp-faq-general-questions.html# pgp-what,what is PGP ? The comp.security.pgp FAQ, available at 2006.01.07.
[31] Leslie M. Orchard, Hacking RSS and ATOM, Wiley Publishing, Inc., 2005.
[32] Dr. Ramesh R. Sarukkai, “How much is a Keyword worth ?”, ACM, pp. 890-891, May 10-14, 2005.
[33] http://zh.wikipedia.org/w/index.php?title=SHA_%E5%AE%B6%E6%97%8F&variant=zh，SHA 家族，available at 2006.01.07.
[34] http://tw2.php.net/sha1，PHP sha1 函數，available at 2006.01.07
[35] http://www.syndic8.com/stats.php?Section=rss#tabtable, Distribution of RSS version, available at 2006.01.07.
[36] Mary Shultz; Sandra L De Groote, “MEDLINE SDI service-how do they compare ?”, Journal of the Medical Library Association, Academic Research Library, pp. 460-467, Oct 2003.
[37] http://uckan.info/depot/monitorthis/, MonitorThis, available at 2006.01.07.

附 錄 一

<lastRSS.php file>, page1

```
<?php
/*
 ================================================================================
 lastRSS 0.9.1
 Simple yet powerfull PHP class to parse RSS files. by Vojtech Semecky, webmaster @ webdot . cz
 Latest version, features, manual and examples: http://lastrss.webdot.cz/
 ---------------------------------------------------------------------------------------------------------------
 LICENSE
 This program is free software; you can redistribute it and/or  modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
 This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
 To read the license please visit http://www.gnu.org/copyleft/gpl.html
 ================================================================================
*/
/**
* lastRSS
* Simple yet powerfull PHP class to parse RSS files.
*/
class lastRSS {
        // -------------------------------------------------------------------
        // Public properties
        // -------------------------------------------------------------------
        var $default_cp = 'UTF-8';
        var $CDATA = 'nochange';
        var $cp = '';
        var $items_limit = 0;
        var $stripHTML = False;
        var $date_format = '';
        // -------------------------------------------------------------------
        // Private variables
        // -------------------------------------------------------------------
        var $channeltags = array ('title', 'link', 'description', 'language', 'copyright', 'managingEditor', 'webMaster', 'lastBuildDate', 'rating', 'docs');
        var $itemtags = array('title', 'link', 'description', 'author', 'category', 'comments', 'enclosure', 'guid', 'pubDate', 'source');
        var $imagetags = array('title', 'url', 'link', 'width', 'height');
        var $textinputtags = array('title', 'description', 'name', 'link');
        // -------------------------------------------------------------------
        // Parse RSS file and returns associative array.
        // -------------------------------------------------------------------
        function Get ($rss_url) {
                // If CACHE DISABLED >> load and parse the file directly
```

<lastRSS.php file>, page 2

```
                $result = $this->Parse($rss_url);
                if ($result) $result['cached'] = 0;
        // return result
        return $result;
    }           // --------------------------------------------------------------------

// Modification of preg_match(); return trimed field with index 1
    // from 'classic' preg_match() array output
    // --------------------------------------------------------------------
    function my_preg_match ($pattern, $subject) {
        // start regullar expression
        preg_match($pattern, $subject, $out);

        // if there is some result... process it and return it
        if(isset($out[1])) {
            // Process CDATA (if present)
            if ($this->CDATA == 'content') { // Get CDATA content (without CDATA tag)
                $out[1] = strtr($out[1], array('<![CDATA['=>'', ']]>'=>''));
            } elseif ($this->CDATA == 'strip') { // Strip CDATA
                $out[1] = strtr($out[1], array('<![CDATA['=>'', ']]>'=>''));
            }

            // If code page is set convert character encoding to required
            if ($this->cp != '')
                //$out[1] = $this->MyConvertEncoding($this->rsscp, $this->cp, $out[1]);
                $out[1] = iconv($this->rsscp, $this->cp.'//TRANSLIT', $out[1]);
            // Return result
            return trim($out[1]);
        } else {
        // if there is NO result, return empty string
            return '';
        }
    }
    // --------------------------------------------------------------------
    // Replace HTML entities &something; by real characters
    // --------------------------------------------------------------------
    function unhtmlentities ($string) {
        // Get HTML entities table
        $trans_tbl = get_html_translation_table (HTML_ENTITIES, ENT_QUOTES);
        // Flip keys<==>values
        $trans_tbl = array_flip ($trans_tbl);
        // Add support for &apos; entity (missing in HTML_ENTITIES)
        $trans_tbl += array('&apos;' => "'");
        // Replace entities by values
        return strtr ($string, $trans_tbl);
    }
```

<lastRS.php file>, page 3

```
// --------------------------------------------------------------------
        // Parse() is private method used by Get() to load and parse RSS file.
        // Don't use Parse() in your scripts - use Get($rss_file) instead.
        // --------------------------------------------------------------------
        function Parse ($rss_url) {
                // Open and load RSS file
                if ($f = @fopen($rss_url, 'r')) {

                        $rss_content = '';
                        while (!feof($f)) {
                                $rss_content .= fgets($f, 4096);
                        }
                        fclose($f);

                        // Parse document encoding
                        $result['encoding'] = $this->my_preg_match("'encoding=[\'\"](.*?)[\'\"]'si", $rss_content);
                        // if document codepage is specified, use it
                        if ($result['encoding'] != '')
                                { $this->rsscp = $result['encoding']; } // This is used in my_preg_match()
                        // otherwise use the default codepage
                        else
                                { $this->rsscp = $this->default_cp; } // This is used in my_preg_match()

                        // Parse CHANNEL info
                        preg_match("'<channel.*?>(.*?)</channel>'si", $rss_content, $out_channel);
                        foreach($this->channeltags as $channeltag)
                        {
                                $temp = $this->my_preg_match("'<$channeltag.*?>(.*?)</$channeltag>'si", $out_channel[1]);
                                if ($temp != '') $result[$channeltag] = $temp; // Set only if not empty
                        }
                        // If date_format is specified and lastBuildDate is valid
                        if ($this->date_format != '' && ($timestamp = strtotime($result['lastBuildDate'])) !==-1) {
                                        // convert lastBuildDate to specified date format
                                        $result['lastBuildDate'] = date($this->date_format, $timestamp);
                        }

                        // Parse TEXTINPUT info
                        preg_match("'<textinput([^>]*[^/])>(.*?)</textinput>'si", $rss_content, $out_textinfo);
                                // This a little strange regexp means:
                                // Look for tag <textinput> with or without any attributes, but skip truncated version <textinput /> (it's not
beggining tag)
                        if (isset($out_textinfo[2])) {
                                foreach($this->textinputtags as $textinputtag) {
                                        $temp = $this->my_preg_match("'<$textinputtag.*?>(.*?)</$textinputtag>'si", $out_textinfo[2]);
                                        if ($temp != '') $result['textinput_'.$textinputtag] = $temp; // Set only if not empty
                                }
                        }
```

<lastRSS.php file>, page 4

```
                // Parse IMAGE info
                preg_match("'<image.*?>(.*?)</image>'si", $rss_content, $out_imageinfo);
                if (isset($out_imageinfo[1])) {
                        foreach($this->imagetags as $imagetag) {
                                $temp = $this->my_preg_match("'<$imagetag.*?>(.*?)</$imagetag>'si", $out_imageinfo[1]);
                                if ($temp != '') $result['image_'.$imagetag] = $temp; // Set only if not empty
                        }
                }

                // Parse ITEMS
                preg_match_all("'<item(| .*?)>(.*?)</item>'si", $rss_content, $items);
                $rss_items = $items[2];
                $i = 0;
                $result['items'] = array(); // create array even if there are no items
                foreach($rss_items as $rss_item) {
                        // If number of items is lower then limit: Parse one item
                        if ($i < $this->items_limit || $this->items_limit == 0) {
                                foreach($this->itemtags as $itemtag) {
                                        $temp = $this->my_preg_match("'<$itemtag.*?>(.*?)</$itemtag>'si", $rss_item);
                                        if ($temp != '') $result['items'][$i][$itemtag] = $temp; // Set only if not empty
                                }
                                // Strip HTML tags and other bullshit from DESCRIPTION
                                if ($this->stripHTML && $result['items'][$i]['description'])
                                        $result['items'][$i]['description'] =
strip_tags($this->unhtmlentities(strip_tags($result['items'][$i]['description'])));
                                // Strip HTML tags and other bullshit from TITLE
                                if ($this->stripHTML && $result['items'][$i]['title'])
                                        $result['items'][$i]['title'] =
strip_tags($this->unhtmlentities(strip_tags($result['items'][$i]['title'])));
                                // If date_format is specified and pubDate is valid
                                if ($this->date_format != '' && ($timestamp = strtotime($result['items'][$i]['pubDate'])) !==-1) {
                                        // convert pubDate to specified date format
                                        $result['items'][$i]['pubDate'] = date($this->date_format, $timestamp);
                                }
                                // Item counter
                                $i++;
                        }
                }

                $result['items_count'] = $i;
                return $result;
        }
        else // Error in opening return False
        {
                return False;
        }
```

<lastRSS.php file>, page 5

```
        }
}
?>
```

附 錄 二

<rss-stylesheet.xsl file>

```
<?xml version="1.0" encoding="utf-8" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />
- <xsl:template match="rss/channel">
- <html>
- <head>
- <title>
  <xsl:value-of select="title" />
    </title>
    </head>
- <body>
- <xsl:if test="0 = count(item)">
  <p>這個 feed 目前沒有資料。</p>
    </xsl:if>
- <xsl:if test="0 < count(item)">
- <ul>
  <xsl:apply-templates select="item" />
    </ul>
    </xsl:if>
    </body>
    </html>
    </xsl:template>
- <xsl:template match="title">
- <div>
  <xsl:value-of select="text()" />
    </div>
    </xsl:template>
- <xsl:template match="item">
- <li>
- <a href="{link}" target="_new">
  <xsl:value-of select="title" />
    </a>
    </li>
    </xsl:template>
    </xsl:stylesheet>
```