南 華 大 學

資 訊 管 理 學 系

碩 士 論 文

以角色為基支援跨網域之網頁應用程式授權方法設計

Designing a RBAC-based Authorization Method for Supporting Web Applications across Domains

研 究 生：蔡政宇

指導教授：王昌斌 博士

中 華 民 國 九 十 七 年 六 月

南　華　大　學

資訊管理研究所

碩 士 學 位 論 文

以角色為基支援跨網域之網頁應用程式授權方法設計

研究生：蔡 政 宇

經考試合格特此證明

口試委員：吳鴻輝

陳象義

指導教授：

系主任(所長)：

口試日期：中華民國　97 年　6 月　11 日

誌　　謝

在研所兩年的日子裡，首先誠摯的感謝指導教授王昌斌博士，在研究過程中碰到的困惑，王老師總是悉心的指導並點出正確方向，使我在研究與學習的過程中獲益匪淺，王老師對學問所要求的嚴謹、態度是我引以學習的最佳典範；在研究實作的過程中，要感謝行政系統發展組的曾清義組長，與曾大哥多次訪談的過程中，我學習到很多更高一層次的實作技巧，以及如何有效的降低源碼撰寫成本的複雜度與重複性。

在這段期間裡感謝南華同儕們彼此的鼓勵與照應，以及實驗室的成員陳育銘學長、林育弘學長、藍裕凱學弟、張堯榮學弟在研究上的協助，與大家一同成長的時光非常愉快。另外要特別感謝於上屆畢業的張逸為學長，您的鼓勵及關心我銘感在心；以及曾於大學時代指導專題的楊吳泉老師，您的支持與鼓勵是我不懈往前的動力；以及所有曾幫助過我的貴人，在此，由衷的感謝你們。

最後要感謝的是我最親愛與敬愛的家人，有你們的勉勵與支持，讓我順利的完成這份學業，願與你們分享這份榮耀，謝謝你們！

蔡政宇　僅識

于　南華大學資管所

九十七年六月

以角色為基支援跨網域之網頁應用程式授權方法設計

學生：蔡政宇　　指導教授：王昌斌 博士

南 華 大 學　資訊管理學系碩士班

摘　　要

隨著資訊應用複雜性，及開發成本考量，虛擬團隊合作開發，已成為趨勢。現今許多網頁應用程式(Web Application)的設計，都是由虛擬團隊分工設計而成；而這些開發的網頁應用程式，都會建構會員管理機制，用以區分管理者(Admin)、一般使用者(User)、閱覽者(Guest)，使之控管部分的服務，不被非法的存取。在權限控管的部分，較多採用以角色為基礎的存取控制(Role-Based Access Control)，作為存取控制的基礎。

這個機制在實作上受到諸多的限制，使用者對網頁應用程式上的工作階段(Session)，無法直接地分享資訊給其它的網頁應用程式作存取。雖然可用 Cookie 將部分的資訊轉存於客戶端，但分享給其他網頁應用程式存取僅限定於相同網域之下(例如：*.yahoo.com)使用，另外 Cookie 本身也有儲存量的限制。為求簡單達到將資訊分享於多個網頁應用程式上，即將所有的網頁應用程式整合在同一台伺服器上，

以方便實作整個系統對於所設計的網頁應用程式施行存取控制。但將所有的服務全部集中在同一台伺服器上，將會造成伺服器嚴重的負擔;且開發者在本地端的建置環境,不一定與遠端伺服器的環境相符。

為解決上述問題，本研究以使用者對網頁應用程式上的工作階段(Session)，結合 XML 達成網頁應用程式權限存取跨網域的實作，使各個網頁應用程式可以在自己的伺服器中運作,不再需要統一彙整到同一伺服器中運行,且不再受限整個系統中網頁應用程式需以同一程式語言撰寫。

關鍵字：RBAC Access Control、Web Application、XML、Cross-Domain

Designing a RBAC-based Authorization Method for Supporting Web Applications across Domains

Student：Cheng-Yu, Tsai　　Advisors：Dr. Chang-Bin, Wang

Department of Information Management

The M.I.M. Program

Nan-Hua University

ABSTRACT

With the advancement of network technology and the consideration of R&D costs, collaboration among cooperation has become one of the trends. Using network application is the major tool of virtual teamwork and these developments of web applications will construct the membership management which includes the identification of administrators, users, and guests etc. We often use the role-base access control as the basis of the access control management.

But this method in operation actually leads to many restrictions. For example, Session can’t be applied to another Web Application Access. Besides, Cookie must be used in the same domains. Currently the general solution is to integrate all the Web Application in the same server in order to facilitate their access control. However, this solution will confuse the designer of the virtual team and make the remote server unable to meet

the need of a local designer perfectly.

In order to improve this problem, this study embedded Session in XML and made Web Application cross the borders between two different domain servers for operational convenience and access control. Each Web Application of the same system with different programming language can be run in their own server exactly.

Keywords ： RBAC Access Control, Web Application, XML, Cross-Domain

目錄

表目錄

圖目錄

第一章、緒論

第一節　研究動機

隨著網際網路蓬勃發展，企業入口網站所提供的服務多元化，從早期由單人可以維護的站台，逐漸發展為需要由團隊共同建置網頁應用程式。當企業入口網站開始提供新型態的服務時，單一團隊可能無法持續支撐企業入口網站的各項服務，而開始採用虛擬團隊共同分工來發展。虛擬團隊採用分工的方式來處理各項服務，雖然可應付各項服務的需求，但是在服務的整合上卻有技術上的瓶頸；雖然部分的問題只需要投資更大的成本即能解決，但另一部分依然是沒有得到良好的解決辦法。

早期企業入口網站所提供的某一項服務如：Web Mail Service 可以讓用戶由網頁上收發郵件，這個服務可以是由單一團隊所建置的網頁應用程式；接著該企業入口網站提供了另一項服務如：Homepage Service 可以讓用戶架設自己的網站。在單純的考量下，Web Mail Service 與 Homepage Service 可以整合為同一個網頁應用程式，使兩個服務間之資訊互通有無。但如果該企業入口網站持續的提供新服務，例如：Blog Service(部落格服務)、Space Service(網路硬碟服務)…等，若依舊將這些服務整合在同一個網頁應用程式中，將會造成網頁

應用程式在後續維護上的困難，以及增加伺服器乘載的負擔。改善方式是將所有的服務變成各個獨立的網頁應用程式，除了可以將部分的服務建置分派給其它的開發團隊處理，以減輕獨立開發的負擔，也可將各個網頁應用程式建置於其它的伺服器上，以降低原本單一伺服器所負荷的承載量。但此時各服務間資訊溝通就變的不容易了。

首先是使用者驗證問題，不同的 Web Application 與使用者工作階段(Session)的資訊是不能直接溝通，當使用者操作不同的服務時，需要重新登入驗證使用者身分，讓服務能獲取使用者相關資訊。這個部分可以使用單一簽入(Single Sign-On)系統來解決，但須花費一定的成本來建置單一簽入系統；目前部分的程式語言，已經支援建置單一簽入的方法在 Web Application 上，但若是開發團隊先前並非使用該支援的程式語言撰寫 Web Application，以至於無法提供單一簽入的能力，此時開發團隊則需要考慮是否更換程式語言，重新開發 Web Application，或額外撰寫可以支援單一簽入系統的函式庫(Function)。單一簽入系統雖然可以驗證使用者能否操作 Web Application 上的服務，但無法更低階進一步的去控管使用者對於 Web Application 上的各個物件的授權與否。虛擬團隊常碰到的另一種狀況，為了使每一個服務能有更好的貼合度，或是為了能更快速無礙的達到目的，都會決議使用同一種程式語言來撰寫 Web Application，以降低服務之間資訊

傳遞的非預期問題;但參與製作的虛擬團隊並非都熟悉使用決議的程式語言，或多或少都會因而影響開發效率。

本研究在不改變現有的限制環境下，提出不需額外花費成本，架構簡單且容易建置部屬的設計方式，使每一個負責建置服務的團隊，能依據慣用的程式語言開發 Web Application；當使用者登入服務時，透過 RBAC 授權機制可以取得用戶在每一個 Web Application 中詳細的授權資訊，並透過 XML 格式傳遞給每一個服務，包含位於不同伺服器與不同網域上的服務們。

第二節　研究目的

如何讓 Web Application 與使用者之間的工作執行階段(Session)分享給其它的 Web Application，一直是許多開發人員心中的問題。在網際網路上許多人提出不同的見解，包含使用 ASP.net 的單一簽入方法、利用 SQL 來製作大型的 Catch 機制、架設 Memcatched 伺服器來存放使用者與 Web Application 之間的工作執行階段(Session)以分享給其它的 Web Application、利用 PHP 中 Session 的特性…等諸多方法。

ASP.net 是目前最容易實作單一簽入的開發語言，但是很顯然的必須使用 ASP.net 來製作。利用 SQL 來建置一個暫存的資料表放置資訊，這是不錯的方法，但是若是要授權能取的資訊的 key 值，就必須

存放於客戶端上的 Cookie，使用 Cookie 就意味著必須在相同網域下才能運行。為了改善 SQL 殺雞用牛刀來存放臨時資訊，以及無法真正達到跨越網域，而有了 Memcatched 伺服器的發展，它的好處就是使用伺服器的記憶體來當作一個大型的 Session Catch，只要有該伺服器的 IP 位址與端口位置(Port)就能輕易撈取資訊；但這就意味著 Memcatched 的危險性，必須使用防火牆對 Memcatched 進行保護，並且要預防有心人士直接從端口中撈取資料。關於 PHP 的 Session 特性，即是利用 PHP 的運行環境作為一個更大的工作執行階段，讓每一個 Web Application 在這個工作階段之下能有權限取得共享的資訊，這表示著每一個 Web Application 必須是在同一伺服器底下，並且是屬於同一個 PHP 的執行工作階段裡。

本研究的目的，使程式設計人員可以不用改變開發習慣，也無須增資其它因應成本，即能達到位於不同網域與伺服器中的 Web Application 能共享資訊，並透過 RBAC 的授權機制，能更進一步的控制用戶對於每一個 Web Application 中各項元件的詳細設定。

第三節　研究範圍

本研究範圍包括跨網域處理機制及建置 RBAC 授權中心，讓用戶在登入系統後透過 RBAC 授權中心取得因應的可用權限。用戶可以

在不同的 Web Application 中操作可用的服務與資源，即便是 Web Application 位於不同的網域中，權限能即時的遞送至不同的網域中，使用戶可以無礙的使用這些服務。本研究並未對安全性議題進行設計，也未替資料傳遞中的訊息加密，因為資訊在傳遞的通道，可以由伺服器中的設定，改為較安全的 SSL 通道傳輸，增進資訊的傳遞安全，來達到較高安全層級的應用。

第四節　論文架構

本論文分為五個主題，其中包含緒論、文獻探討、架構流程、實作設計、結論。在緒論裡會提及研究動機與目的以及本研究的範圍；文獻探討中會談到目前常見的存取控制方法和 Web Services 中的相關知識；架構流程中包含本研究系統的基礎架構、系統完整的運行流程、RBAC 授權中心授權方式、跨越網域處理機制；實作設計即是把架構流程中提及的部分實際的製作出來，預期功能包含用戶能透過 RBAC 授權中心，取得完整的服務授權資訊，並且能在用戶位於不同網域上的 Web Application，進行合法的服務操作；結論即是把整體架構實作後之心得和探討。

第二章、文獻探討

第一節　存取控制相關研究

存取控制相關研究將針對存取控制的定義、存取控制方法、存取控制策略及以角色為基礎的存取控制作介紹與說明。

壹、存取控制的定義

存取控制(Access Control)[1,4,9,14,15,23]是當使用者請求一項服務時，判別使用者是否有該存取之能力，從而限制操作之能力，避免使用者超出自身的作業範圍，造成不當的資訊外泄或資源使用。存取控制系統[9,34]可分為存取控制機器與存取控制策略。存取控制機器是基礎的軟硬體，以執行存取控制方法來達成存取控制策略目標，目前常見的存取控制方法(Access Control Approaches)有存取控制矩陣(Access Control Matrix)、存取控制串列(Access Control List)、能力串列(Capabilities List)；存取控制策略(Access Control Policies)是高階的指導原則，用以決定存取如何控制及存取策略訂定，其中有自由裁量存取控制(Discretionary Access Control)、強制性存取控制(Mandatory Access Control)、以角色為基礎之存取控制(Role-Base Access Control)。

貳、存取控制方法

目前常見的存取控制方法有「存取控制矩陣(ACM)」、「存取控制串列(ACL)」、「能力串列(CL)」，下面將介紹這幾種存取控制方法。[20]

1. 存取控制矩陣(Access Control Matrix)

存取控制矩陣(ACM)[26]，是最簡單的存取控制方法，也是最早被用來控制資料存取的方法；它是由一個二維的陣列表現出主體(Subject)對於物件(Object)所擁有的權限。可以很清楚將彼此間的權限清楚表現出來，但若頻繁的更動主體(Subject)與物件(Object)，將會造成系統複雜且不易維護。如表 2.1 所示，User A 對於 Object 1 有 Read、Write、Modify 存取權限，對於 Object 2 無任何存取權限，對於 Object 3 有 Read、Write 存取權限。

表 2.1 存取控制矩陣

	Object 1	Object 2	Object 3
User A	Read、Write、Modify		Read、Write
User B	Read	Read、Write、Modify	
User C	Read		Read、Write

2. 存取控制串列(Access Control List)

存取控制串列(ACL)[34]是使用以存取控制矩陣為基礎的資料存取控制方法，每一個物件對應一個串列主體(Subject)。存取控制串列描述每一個物件各自的存取控制列表，並記錄可以對此物件進行存取的所有主體及每個主體可以存取的權限。因此從 ACL 中可以清楚的瞭解每一個物件的存取控制狀況，對於整體存取控制機制管理上也很簡單。因為 ACL 是透過以物件管理而形成的串列，因此由物件的角度來管理權限是非常方便。當多個使用者對於物件有相同的存取控制權限，可讓這些使用者形成一個群組，如此一來便可簡化整個存取控制機制。如圖 2.1 所示，User A 對 Object 1 有 Read、Write、Modify 的存取權限，User B 和 User C 對 Object 1 有 Read 的存取權限。

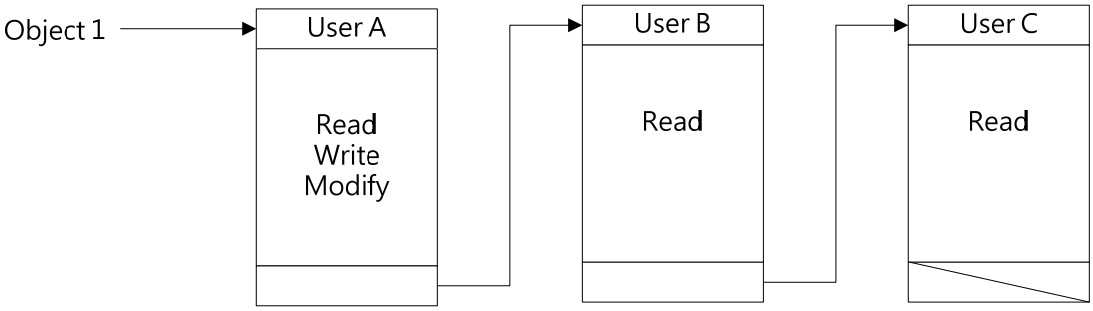

圖 2.1 存取控制串列

3. 能力串列(Capabilities List)

能力串列(CL)[27]運作模式類似於 ACL，也是以存取控制矩陣為基礎的存取控制機制，但它是以主體(Subject)的角度將每一個主體對應一連串的物件存取控制權限，描述每一個主體可存取的物件及權限，形成一個物件權限串列。由於主體對於某個物件擁有哪些存取權限不易檢閱，且對於存取權限的刪修也不方便，因此不常被使用。如圖 2.2 所示，User A 對於 Object 1 有 Read、Write、Modify 的存取權限，對於 Object 2 沒有任何權限，對於 Object 3 有 Read、Write 的存取權限。

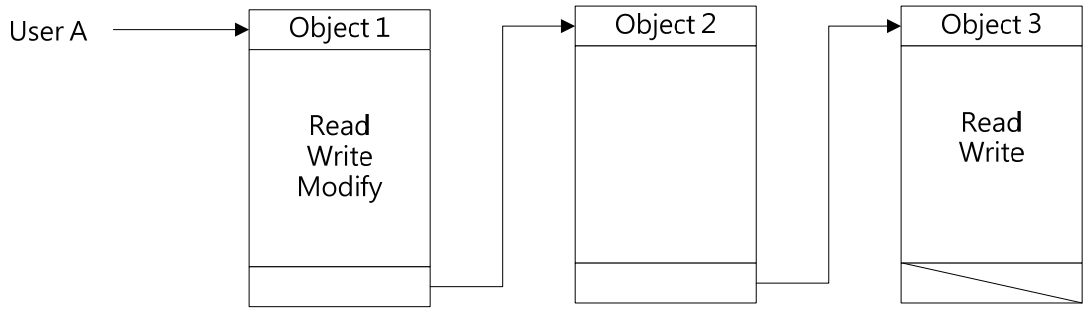

圖 2.2 能力串列

參、存取控制策略

存取控制策略[1,5-10,12,15,19]較常見的有三種，包括自由裁量存取控制(Discretionary Access Control)、強制存取控制(Mandatory Access Control)、以角色為基礎之存取控制(Role-Base Access

Control)；前兩種策略屬於傳統方式，在管理主體與物件的作法僅屬於最基礎的權限規範階段，所提供的彈性無法滿足企業實際的需求；以角色為基礎的存取控制是在近十年來逐漸發展的策略，利用角色(Role)為控制主體，將使用者(User)與權限(Permission)區分開來，以增進政策定義上的彈性和管理上的便利。

1. 自由裁量存取控制(Discretionary Access Control)

   自由裁量存取控制[1,5-9,12,20,34]，如圖 2.3 所示，是對於資訊或物件(Object)屬個人使用者所擁有，對於該資訊或物件的授予使用權，完全取決於個人的自由裁量方式管理，不會牽涉到其他共同擁有者之資訊。因此 DAC 又稱為基於個體的(Identity-Based)執行權管制機制；缺點是他對於資訊得傳遞及控制並不嚴謹，無法有效防止資訊擴散，而且無法保證系統中資訊流動的正確性，因此存取的限制可以容易的被跳過，面對現在快速變化的網際網路環境，它無法順應變化。

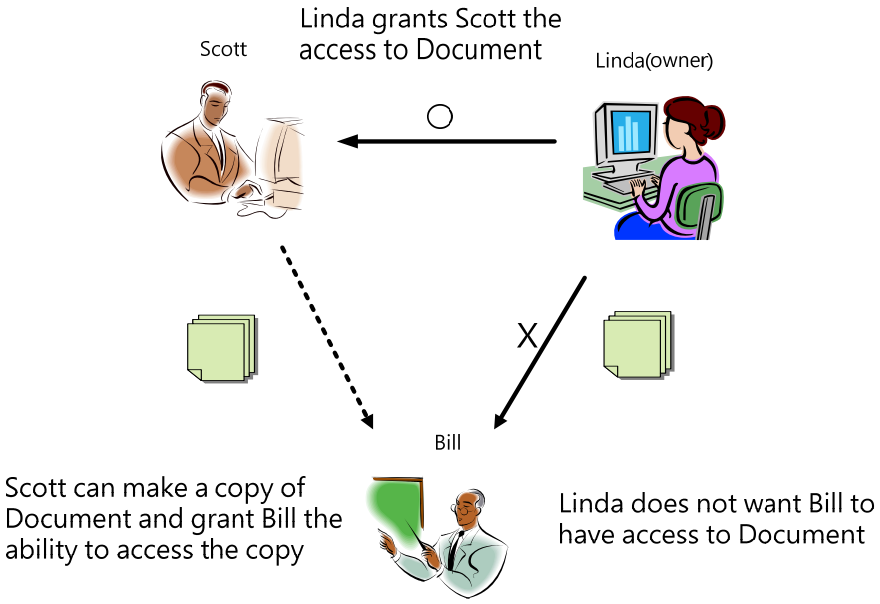

圖 2.3 自由裁量存取控制

2. 強制性存取控制(Mandatory Access Control)

強制存取控制[1,5-9,12,20,32-34]，如圖 2.4 所示，對於被使用的資訊或物件執行要求，是以使用者與該資訊或物件之間的安全等級作為依據基礎，並不可因個人因素而隨意改變。適用於機密性需求高的單位，例如國防單位。由於強制行的執行權管制是採用資訊的機密等級作為執行管制權的依據，因此 MAC 又稱為以規則為基礎(Rule-Based)執行權管制；缺點是不提供對於一般的網路認證動作。

MAC Information Flow

					Object
	R/W	W	W	W	TS
	R	R/W	W	W	S
	R	R	R/W	W	C
	R	R	R	R/W	U
Subject	TS	S	C	U	

Secret level
Top Secret(TS)
Secret(S)
Classified(C)
Unclassified(U)

圖 2.4 強制性存取控制

3. 以角色為基礎之存取控制(Role-Base Access Control)[20,33]

有鑑於 MAC 在群組權限控制上無法重複將使用者設定為不同群組的限制，加上為了反應企業組織與資訊系統間的使用關係、讓資訊系統更能貼合企業的需求，以角色為基礎存取控制機制(RBAC)在 1992 年被正式提出。以角色為基礎之存取控制主要的觀念是在使用者及資訊資源權限之間加入角色(職務)的概念，將所需的權限指派該角色，再將使用者指派至所屬的角色上，以角色來決定使用者是否執行某個物件的權限，是一種非隨意性的存取控制(non-discretionary access control)。Sandhu 等人提出 RBAC 模型，主要包括了四個元件，分別為使用者(User, U)、角色(Roles, R)、權限(Permissions, P)、工作階段(Session ,S)，各元件關係圖如 2.5 所示，主要的特色包含了權責區分(Separation of Duty)、職務階層(Role Hierarchy)、限制(Constraint)、最小權限(Least Privilege)。

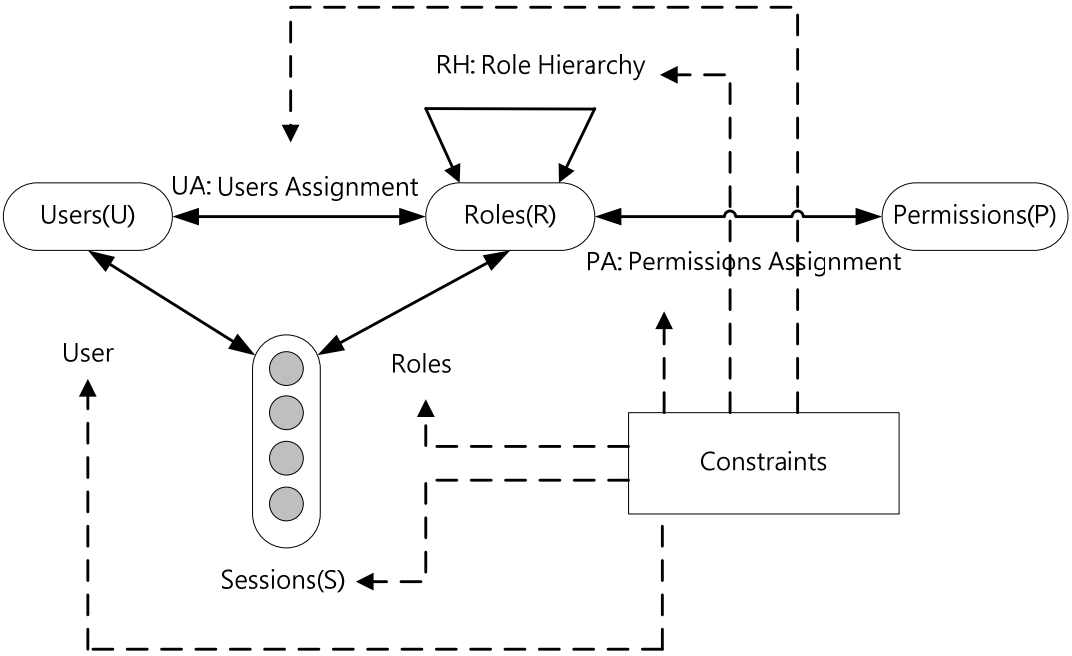

圖 2.5 以角色為基礎之存取控制

表 2.2 存取控制策略比較

Criteria	DAC	MAC	RBAC
權限管理基本單位	個人	個人	群組
可分類管理	否	否	可
權限設定複雜度	極高	高	低
可歸屬於單一群組	否	否	是
可歸屬於多群組	否	否	是
整體管理複雜度	高	高	低

各種存取控制策略各有其優缺點，如表 2.2 所示，由於 RBAC 相對於其它權限控制機制，具有可預先制定角色權限，再將使用者設定給某個角色而使之取得預設之使用權限，加上能夠重複地將不同的角色設定給同一個使用者，具有相當的使用彈性，並且能有效地減輕系統管理者的負擔。

肆、以角色為基礎的存取控制

以角色為基礎之存取控制，下面將針對角色概念、RBAC 發展歷程、RBAC 模型、RBAC 特色進行介紹。[15]

1. 角色概念

角色(Role)[9]的意涵有三種：(1)角色是使用者的集合，主要在說明角色所含因子。(2)角色是權限的集合，主要再說明角色所賦予的行動。(3)角色是使用者結合權限的集合，角色會列舉符合的使用者與對應之權限集合。以角色為基礎的存取控制[5,6,8,9]，是新一代的存取控制策略，在使用者與權限之間加入角色關係；利用角色介於使用者與權限之間的好處來簡化管理程序，且變得具有彈性。例如：將權限賦予給角色，並指向給使用者來執行，當使用者的職務或任務有所變動，只需要調整使用者的角色，即可更動使用者的執行權限，讓

權限的設定與管理更為簡單快速。

2. RBAC 發展歷程

以角色為基礎之存取控制[1,9]，可視為以群組或職位為基礎的執行管制權，是一套相當完善的存取控制機制。RBAC 起源於 1992 年，D. Ferraiolo 和 R. Kuhn 在美國國家技術標準局(NIST)所舉辦的電腦安全研討會上，共同發表了”Role-Based Access Control”[24]，在 1996 年 R. Sandhu 等學者在 IEEE Computer 期刊所發表一篇”Role-Based Access Control Models”[33]，此為探討 RBAC 概念之經典文獻。在 2000 年美國國家技術標準局(NIST)的 D. F. Ferraiolo、R. Sandhu、S. Gavrial 共同發表”A Proposed Standard for Role-Based Access Control”，此為 RBAC 訂定最為完整的標準架構[25]。在 2001 年 D. Ferraiolo、R. Sandhu、S. Gavrila、R. Kuhn、R. Chandramouli 在 ACM Transactions on Information and System Security 共同發表”Proposed NIST Standard for Role-Based Access Control” [21]，主要制訂了 NIST 的 RBAC 標準，並修正部分的 RBAC 模組，包括刪除 Symmetric RBAC、權限分成操作與物件兩個相依物件、增加對會期的描述與訂定

RBAC 的功能說明。

3. RBAC 模型

1996 年由 R. Sandhu 等學者所建構 RBAC 的雛形，包括使用者(Users)、角色(Roles)、權限(Permissions)、會期(Sessions)等四個元素，也另外提出四個基於 RBAC 模式，包括 $RBAC_0$、$RBAC_1$、$RBAC_2$、$RBAC_3$，四者彼此之間的架構請參閱圖 2.6 所示。

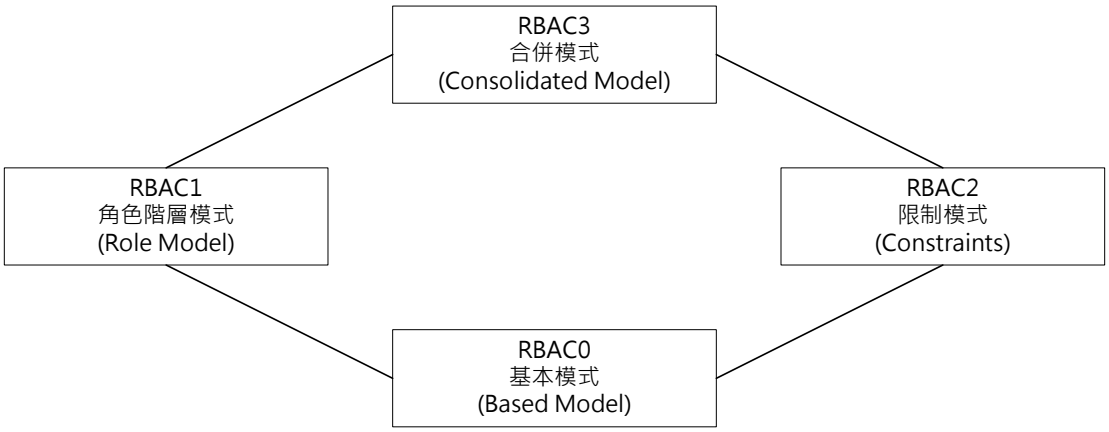

圖 2.6 RBAC 架構關係圖[2,8,10,13,19]

(1) $RBAC_0$

$RBAC_0$[7,8,10,13,16,17,19]為基本模式(Based Model)，也是 RBAC 的基礎模式，定義三個主要的元素，包括使用者(Users)、角色(Roles)、權限(Permissions)，其中使用者可被指派一至多個角色，反之一個角色也可以由多個使用者來

授予；角色可以對應到一至多個權限，反之一個權限也可對應多個角色。因此使用者、角色、權限之間同時存在多對多的關係，透過角色的分派，讓使用者取得因應的權限。另外會期(Session)的部分，主要是建立使用者與角色之間的關聯性，使用者可以同時建立多個會期，而每個會期必屬於某一個使用者，一個使用者的會期將會對應至該會期所被授權的角色集合。關於 $RBAC_0$ 基本模式圖，請參閱圖 2.7 所示。

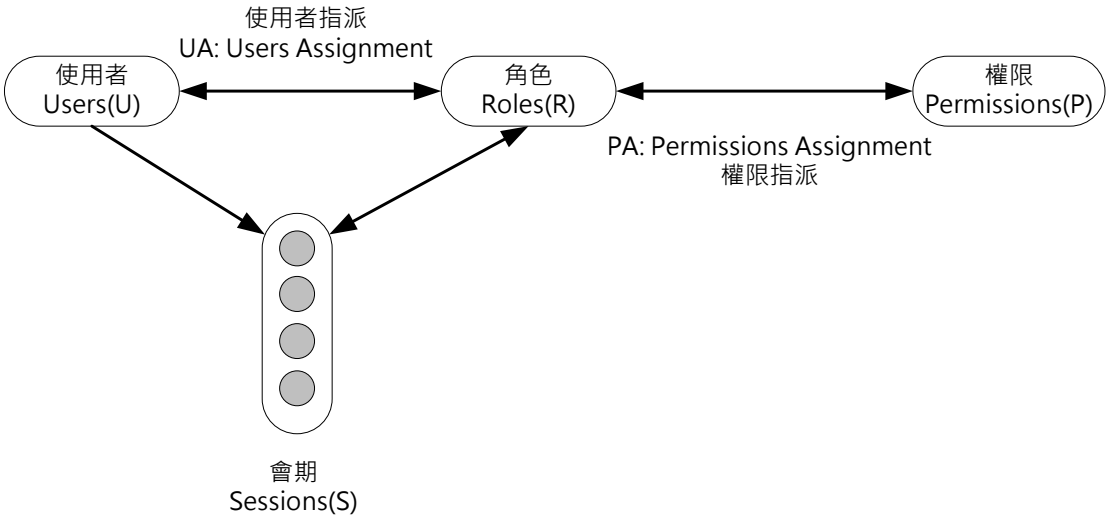

圖 2.7 $RBAC_0$ 基本模式圖[13,16,17,19]

(2) $RBAC_1$

$RBAC_1$[7,8,10,13,16,17,19] 為角色階層模式(Hierarchy RBAC)，是 $RBAC_0$ 的衍生模式，主要在 $RBAC_0$ 的特性加

上角色繼承與階層概念(Role Hierarchy)。角色與角色之間具有多對多的關係，即為一個角色可以與多個角色形成層級上的關係。另外角色階層可區分為一般性角色階層模式(General Hierarchical RBAC)和限制性角色階層模式(Limited Hierarchical RBAC)；一般性角色階層模式係指高層角色(Senior Role)可以完全繼承低層角色(Junior Role)的所有權限，且不受任何條件的限制；限制性角色階層模式係指高層角色(Senior Role)可以有條件的繼承低層角色(Junior Role)，並非完全擁有。在企業組織中，角色是具有階層性的架構，高職位的角色能繼層低職位角色之權限，因此 $RBAC_1$ 符合企業架構的需求。關於 $RBAC_1$ 角色階層模式圖，請參閱圖 2.8 所示。

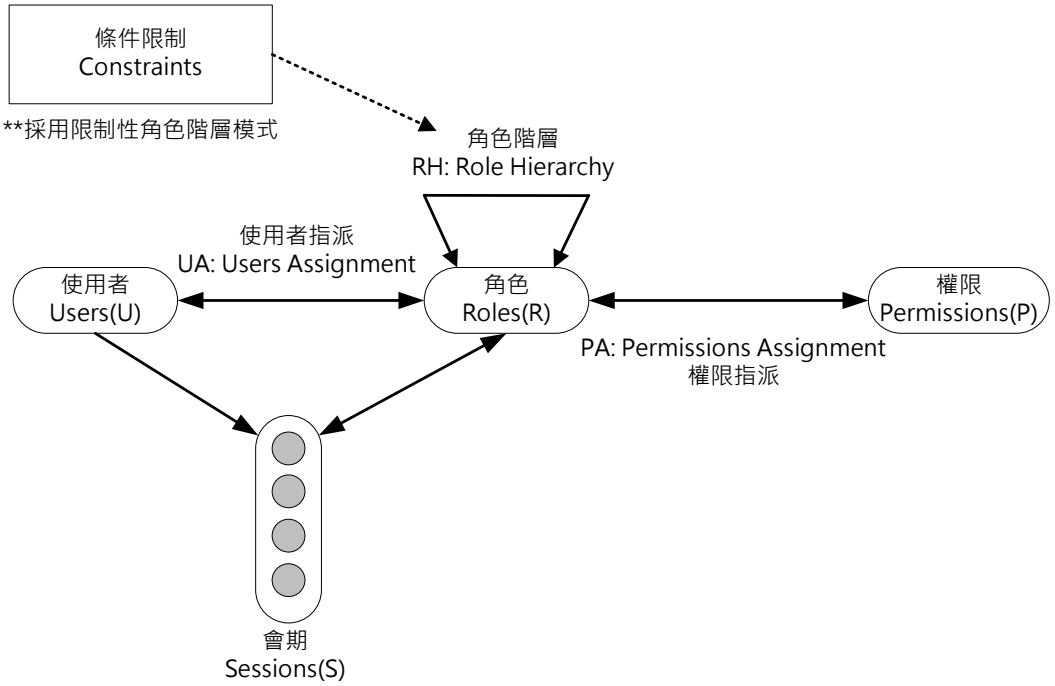

圖 2.8 $RBAC_1$ 角色階層模式圖[13,16,19]

(3) $RBAC_2$

$RBAC_2$[7,8,10,16] 為限制模式(Constraints)，主要是在 $RBAC_0$ 模式中加入限制條件(Constraints)，其範圍包括使用者與會期的建立(Role Activation)、角色授權(Role Authorization)、操作執行(Operation Execution)。限制條件的形式主要包含權責區分(Separation of Duties)、先決角色(Prerequisite Role)、角色數量限制(Cardinality)。權責區分是一個安全機制，主要用來將具有互斥性(Mutually Exclusive)的角色分派給不同的使用者來負責，期又可細分為強互斥(Strong Exclusive)與弱互斥(Weak Exclusive)。先決角色主要用來限制，當使用者被指派為角色 A 時，必須

曾經擁有過角色 B。角色數量限制主要用來限制，某個角色最少需由幾個人負責擔當或最多需要由多少人來擔任。關於 $RBAC_2$ 角色階層模式圖，請參閱圖 2.9 所示。

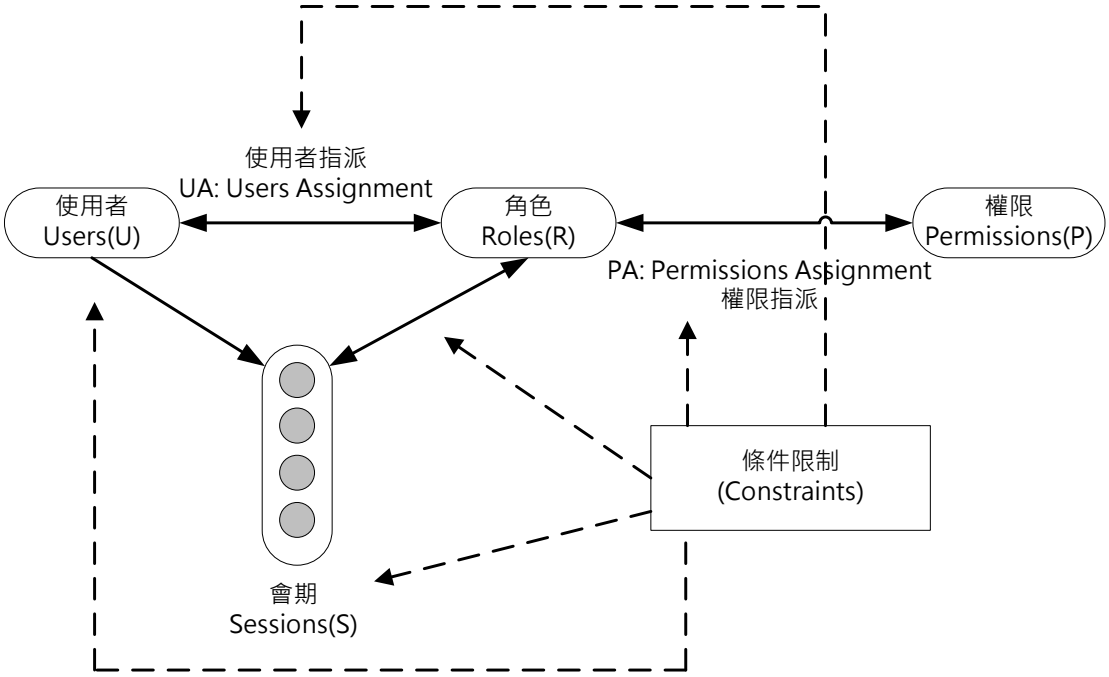

圖 2.9 $RBAC_2$ 限制模式圖[16]

(4) $RBAC_3$

$RBAC_3$[2,6,7,8,10,13,16] 為合併模式 (Consolidated Model)，主要包含了 $RBAC_0$ 模式、$RBAC_1$ 模式、$RBAC_2$ 模式的組合，有關 $RBAC_3$ 合併模式圖，請參閱圖 2.10 所示。

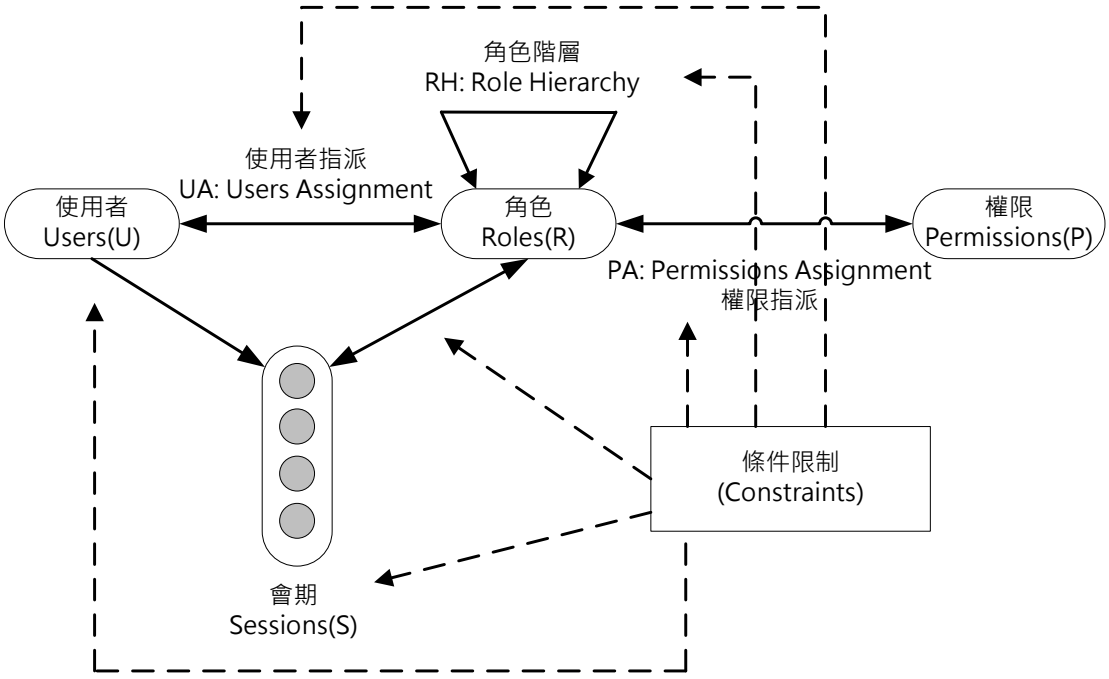

圖 2.10 $RBAC_3$ 合併模式圖[2,6,7,8,10,13,16]

4. RBAC 特色

RBAC 具備五個主要特色[1,6,8,9,14,21,25,33]：

(1) 最小基本權限(Least Privilege)

一個角色可能會擁有多個不同權限，一個使用者可能擁有多重角色。在執行特定職務的需求時，特定角色的權限才會授予給特定職務。例如：使用者在執行特定職務時，該使用者需具備特定的角色且該角色已經被授予相關權限，另外擁有多重角色的使用者無法利用其它角色來進行特定角色的職務，以避免權限遭受濫用或發生衝突。

(2) 責任區分(Separation of Duties)

責任區分是針對互斥性質的角色職務予以訂定規範與限制，目的在於防止舞弊或衝突的狀況發生，其可分為靜態權責區分(Static Separation of Duty Relations)與動態權責區分(Dynamic Separation of Duty Relations)。前者用來避免使用者取得互斥的角色職務，例如：一個員工不能同時擁有申請與審核的職務。後者用來限制使用者於同一特定時間點僅能執行某一個互斥角色的職務，例如：一個員工不能在同一時間內執行授權與放行的職務。

(3) 抽象資料(Data Abstraction)

傳統的存取控制系統採用讀取(Read)、寫入(Write)、執行(Execute)、擁有權(Owner)等指令方式實際操作存取動作，而 RBAC 可以透過較為人性化的方式設定存取動作，例如：申請、審核、通過。

(4) 角色階層與繼承(Role Hierarchical)

角色之間的階層關係可以存在有權限的繼承關係，根據一般化(Generalization)與特殊化(Specialization)的原則，前者越往上層的角色越特殊化，亦擁有較高的權限，後者越下層的角色越一般化，亦擁有較基本的權限，透過此作法可

以簡化授權管理[1]。

(5) 容易管理(Ease of Administration)

以角色為基礎的存取控制(RBAC)是將角色賦予給使用者，權限賦予給角色來執行，當使用者的職務有了變動時，只需要調整使用者的角色，就可以變更使用者所具有的權限，讓權限的設定與管理更為簡單快速。

第二節　Web Services

Web Services[5,22]是十分常見的名詞，但是對於什麼是 Web Services，各方人士各說其詞，每個人回答的版本不盡相同。Web Services 有個標準定義[11]：「Web Services 是一些應用在網路上的標準技術來傳送的鬆散偶合軟體元件」，簡而言之，Web Services 是利用 XML、SOAP、WDSL 等技術作為溝通橋樑來提供軟體元件服務。例如：程式開發者在完成一支計算機程式後，可透過 Web Services 的相關技術放置在 Server 上，讓遠端的使用者透過 Web Services 來使用這支計算機程式。根據它的定義，Web Services 有以下的特點[11]：

(1) Web Services 提供服務的對象並非一般使用者，而是能透過網路呼叫的電腦程式。

(2) Web Services 是透過現存的網路通訊協定來提供服務，也就是

只要能使用 HTTP、SMTP 等通訊協定的程式工具或程式語言，就能夠使用 Web Services。

(3) Web Services 提供了標準的、不限於技術平台的存取介面來提供服務。

(4) Web Services 利用 XML 來做為溝通工具。

(5) Web Services 提供整合的服務項目查詢服務，讓準備使用 Web Services 的程式能夠找尋到自己想要使用的服務項目。

下面將針對運行於 Web Services 上的網頁應用程式(Web Application)、網頁應用程式執行工作階段(Session)、Cookies、XML進行介紹。

壹、Web Application

Web Application[3]是一種架設在 Web 伺服器(Server)上，接受使用者資訊、處理並回傳網頁資訊的應用程式，基本的架構是Client/Server，Server 只負責處理資料，並且和資料儲存區間溝通與交換資料，在處理完成後產生網頁指令，並輸出到用戶端。用戶端(Client)是一種實作網路通訊的本機應用程式(Local Application)通常是瀏覽器(Internet Explorer/ Firefox)，或是實作網路通訊用戶端的伺服器程式，負責處理/解譯由伺服器傳輸而來的網頁標記資料(tagged

data)，然後繪製(rendering)成一個完整的網頁畫面，並且處理存在標記資料中的指令程式碼(scripting language)，讓用戶端具有快速處理與回應的能力。Web Application 使用的標記資料是一種以標記(tag)定義各項資料的格式，藉以讓用戶端處理的資料流(stream)，網頁使用的標記資料格式稱為 HTML，或者是使用更嚴謹的 XML 格式，裡面包含了文字資料，超連結或是加密的二進位資料(編碼為 Base64 格式的字串)。常見的 Web Application 包含 Webmail(電子郵件收發)、Online Retail Sales(線上零售)、Discussion Boards(討論區)、Weblogs(部落格)等。

貳、網頁應用程式執行工作階段(Session)

網頁用程式執行工作階段狀態[30]，讓使用者在使用者巡覽不同的網頁時，儲存和擷取使用者的數值。HTTP 是沒有狀態 (Stateless)的通訊協定，表示 Web 伺服器會將頁面的每個 HTTP 要求視為獨立要求；伺服器不會保留先前要求所使用的變數值。網頁應用程式工作階段狀態會在限制時間間隔內，將來自相同瀏覽器的要求識別為一個工作階段，並且提供保存這個工作階段期間內之變數值的功能。工作階段是由唯一的工作階段識別項所識別，當啟用網頁應用程式的工作階段狀態時，應用程式中對頁面的每個要求，都會檢查瀏覽器送出的

Session ID 值。如果並未提供任何 Session ID 值，網頁應用程式會啟動新的工作階段，然後搭配回應將該工作階段的 Session ID 傳送給瀏覽器。

根據預設，Session ID 值會存放在 Cookie 中，但是也可以設定應用程式將 Session ID 值存放在工作階段的 URL 中。只要使用相同的 Session ID 值持續產生要求，工作階段就會視為使用中。如果特定工作階段的要求間隔超過指定的逾時值 (通常為 30 分鐘)，則工作階段會視為已過期。以過期的 Session ID 值產生要求，會導致啟動新的工作階段。

參、Cookie

Cookie[18,28,31]是網際網路上最常見的一種儲存少量資訊的機制，網站伺服器透過 Cookie 將少量的資訊儲存至使用者的瀏覽器內，主要是因為網頁傳輸協定(Hypertext Transfer Protocol)是一種無狀態的通訊協定(Stateless Protocol)，因此大多數的網站都會利用 Cookie 來維持與使用者之間的部分資訊，或儲存網頁應用程式所需要的資訊。基本上 Cookie 可分為持續性 Cookie(Persistent Cookie)與暫時性 Cookie(Transient Cookie or Session Cookie)兩種。

1. 持續性 Cookie

持續性 Cookie(Persistent Cookie)主要儲存於客戶端的硬碟裡，不同的瀏覽器對於 Cookie 管理方式略有不同；Microsoft Internet Explorer 是將每一個網址的 Cookies 獨立存放於 \%StstemRoot%\COOKIES 的目錄下，Mozilla Firefox 則是將 Cookie 存放於 Cookie.txt 的檔案中。但它們處理 Cookie 的基本結構是不變的，持續性 Cookie 基本結構請參閱表 2.3。

表 2.3 持續性 Cookie 基本結構

參數	說明
NAME=[Value]	用來設定 Cookie 的名稱與內容。一般而言 Cookie 的內容不允許使用特殊符號,若必須包含特殊符號時,必須先經過網址編碼(URL Encode)才可存入 Cookie 裡，另外對於字串長度也有限制。
Expire=[Date]	用來設定 Cookie 的到期日，日期格式必須符合

	RFC882 、 RFC850 、 RFC1036、RFC1123 等所定義格式。當系統時間超過所設定的日期時，該 Cookie 的內容就會自動失效。
Domain=[Domain_Name]	用來設定 Cookie 所能存取的網域範圍，預設空值表示僅有原本的域名可以存取，若設定為其它域名，只有相符合的域名才可存取（例如：*.yahoo.com）。當 Cookie 的內容被搜尋時，首先必須檢查 Cookie 內容所定義之網域是否符合允許範圍。
Path=[Path_Name]	用來設定 Cookie 的存放路徑。如果設定了路徑，Cookie 被搜尋時，必須為相同的路徑目錄下的網頁才能被存取。
Secure	用來設定傳輸協定是否以安

	全通道加密傳輸。

持續性的 Cookie 限制，會因不同的瀏覽器而略有不同，大致上包含下列項目：

(1) Cookie 總數不可超過 300 個

(2) 單一網域不可超過 20 個

(3) 每一個 Cookie 內容不可大於 4096 個位元組

2. 暫時性 Cookie

暫時性 Cookie(Transient Cookie)又稱為 Session Cookie，係指在瀏覽器工作階段，將暫時需要的資訊存放於客戶端的記憶體中，當使用者關閉瀏覽器後，該 Cookie 隨之消失。

肆、XML(Extensible Markup Language)

XML[5]是由 W3C(Word Wide Web Consortium)[35]所制定，延伸自 SGML 的標記語言，由於具有開放的架構，加上不被技術平台所限制的特性，在 XML 推出後隨即廣受業界所接受。XML 的應用範圍相當廣泛，只要透過標準化的標籤制訂方法，系統開發者即可利用它來開發企業間通訊的資料格式，甚至是通訊協定。XML 為 Web Services 其它技術的基礎，也由於 XML 本身跨平台的特性，方使 Web

Services 得以突破技術平台的限制。

伍、Memcached

Memcached[29] 是一套分散式的快取系統，最初是 Danga Interactive 為 LiveJournal 所發展的。Memcached 缺乏認證及安全機制，在使用上僅需得知位址(IP)及端口(Port)即可取出資料，由於企業在建置 Web Services 時已加入具高安全層級之防火牆，Memcached 於防火牆之內受到保護，因此該技術在企業界仍被廣泛使用。

第三章、架構流程

第一節　系統架構

本研究機制包括兩個主要架構：(1)RBAC 授權機制，(2)跨網域處理機制。本章節以「輕度障礙生數位學習平台」建置為例，在這個平台上其中有兩個團隊分別負責製作「概念學習」及「問題導向學習」兩個區塊。製作「概念學習」的團隊慣用 ASP.net 作為開發語言，製作「問題導向學習」的團隊慣用 JSP 作為開發語言。在以往的慣例都會協調以什麼樣的語言開發為主軸，非必要時不使用其它的程式語言做為輔佐；假設位於遠端的伺服器是 Linux 的平台，ASP.net 目前無法完全的相容於 Apache 中，因此「概念學習」的團隊就必須放棄慣用的 ASP.net 改由 JSP 或是其他 Apache 可以運行的開發語言；假設「問題導向學習」的團隊改用 JSP 以符合在遠端伺服器中 Apache 的要求，但遠端伺服器不一定能完全的配合「問題導向學習」團隊在建置上的所有需求，當中必須來來回回不斷的協調與確認，才能在最後上線時所碰到的問題降至最低。透過本系統的架構，無論是「概念學習」或「問題導向學習」的設計團隊，不再有這種困擾，他們可以將服務建置在自己的伺服器中，不需依賴遠端伺服器上的需求及標準，如此般「概念學習」的開發團隊，即可使用慣用的開發語言建置服務。

本系統架構是將設計團隊的每一個網頁，都視為每一個物件

(Object)，將這些物件的位址全部導入 RBAC 授權中心，並運用 RBAC 的運作原理配置角色，最後再將角色指向給用戶。用戶在登入後取得對應角色即可取得一張可用的物件表，當用戶在操作 Web Application 時，便是驗證這張物件表來判斷使用者是否有操作 Web Application 中部分網頁的權限。當用戶移動到不同的 Web Application 時，系統會優先判斷用戶是否曾經登入過，曾經登入過的用戶都會取得一個由 RBAC 授權中心所頒予的 Session Client；當系統得知用戶有 Session Client 時，將會透過這個進入 RBAC 授權中心要求取得完整的授權資料，RBAC 授權中心在確定用戶身分後，將會以 XML 的格式將用戶的相關資訊傳遞給 Web Application，由 Web Application 重新建立使用者工作執行階段，讓用戶的權限如同 Web Application 轉移之前一樣。本研究機制基礎架構圖如 3.1 所示。

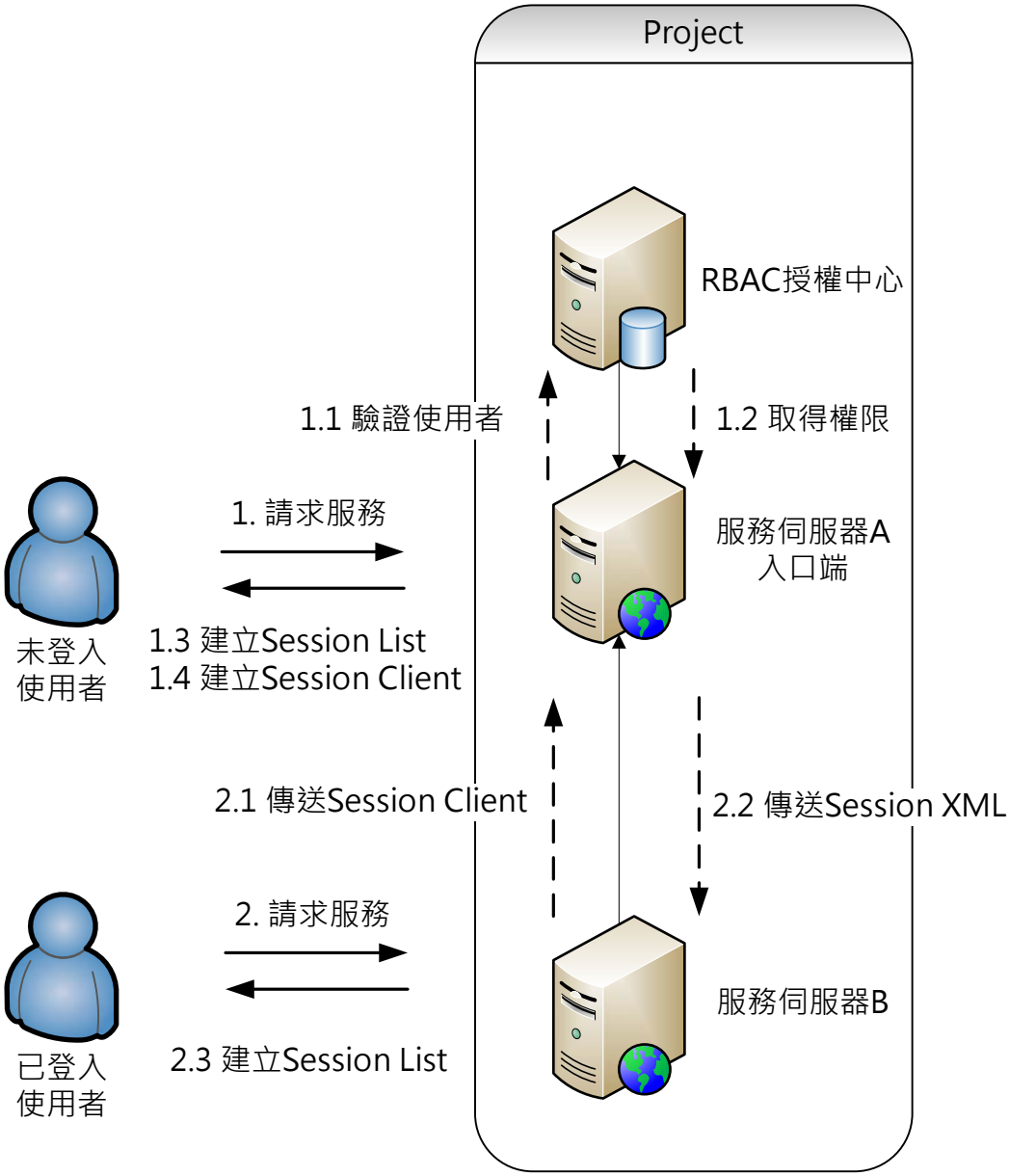

圖 3.1 系統基礎架構

第二節 RBAC授權中心/機制

一個合法的使用者，在登入之後，必須經過 RBAC 授權中心取的自己的授權關係表(Session List)，這個關係表將會儲存於使用者與 Web Application 的工作執行階段(Session)中。RBAC 授權中心是利用

SQL 資料表與預存程序來完成 RBAC 的授權機制。

在使用者首次進入系統時，會將用戶 ID 傳送至 SQL 中，透過預存程序取得關聯的角色 ID，由於一個使用者可能會有多個角色，因此建立一組角色陣列以儲放這些可用的角色 ID；經由預存程序的控制，可以優先過濾出使用者是否在核定的時間內、核定的地點內擁有存取該角色之能力，若正確無誤則將此角色 ID 記錄下來，反之則拒絕存取該角色 ID，直到過濾完全部角色陣列成為一個可用的角色陣列清單，該部分詳細流程請參閱圖 3.2。經由預存程序過濾出的可用角色 ID 後，便進入另一個預存程序，將可用的角色 ID 取得關聯的物件 ID，由於一個角色可能會包含多個可用物件，因此必須建立一組物件陣列以存放這些可用的物件 ID；經由預存程序的控制，將不在核定時間內、核定地點內的物件予以剃除。最後將產生一張完整的用戶授權關係表(Session List)，並將之存放於使用者與 Web Application 的工作執行階段(Session)中。該部分的詳細流程圖請參閱圖 3.3。最後產生的關係授權表(Session List)存放於使用者與 Web Application 工作執行階段(Session)中的資訊請參閱表 3.1。

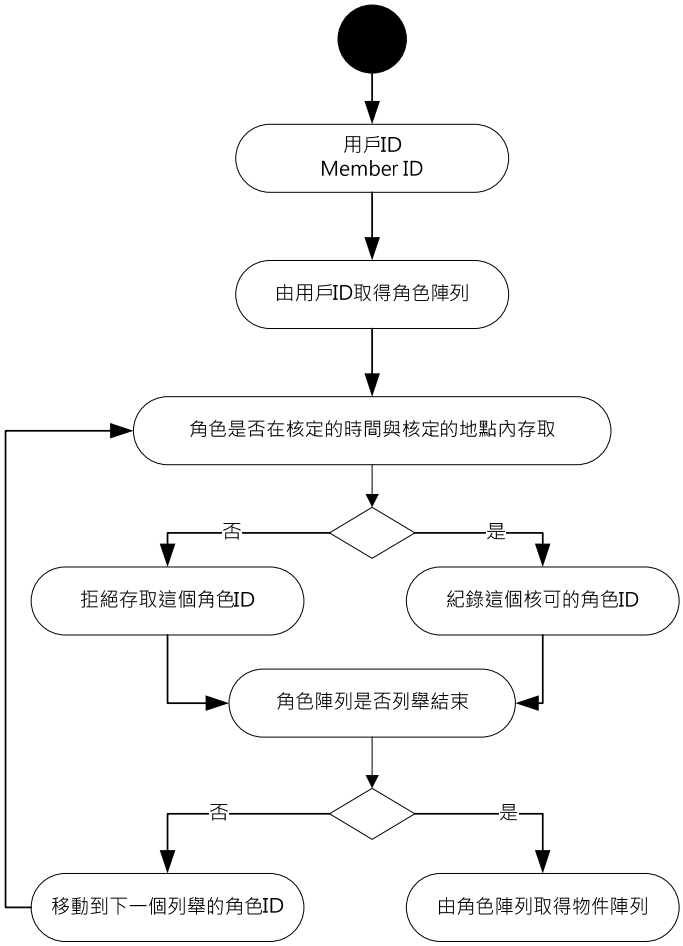

圖 3.2 可用角色 ID 取得流程

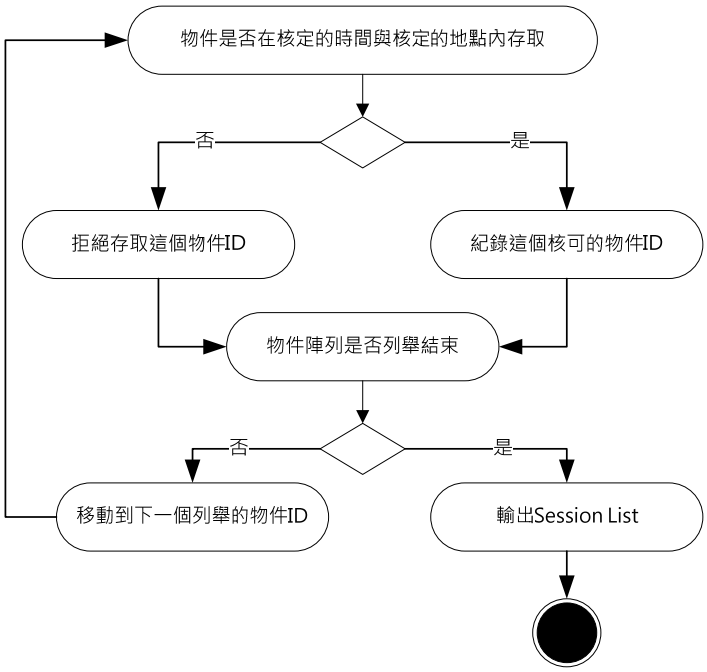

圖 3.3 可用物件 ID 取得流程

表 3.1 Session List

Session key	Session Value
ID	ayu
Name	亞由
Email	sfw.sakana@gmail.com
R_Count	2
R_ID_0	Browser01
R_ID_1	Sysadmin
O_Count	14
O_ID_0	Admin_O2R
O_ID_1	Admin_Objects
O_ID_2	Admin_R2O
O_ID_3	Admin_R2U
O_ID_4	Admin_Roles
O_ID_5	Admin_U2R
O_ID_6	Admin_Users
O_ID_7	Logout
O_ID_8	O_List
O_ID_9	Radmin_EX01
O_ID_10	Session_List
O_ID_11	Session_XML
O_ID_12	Session_XML_Show
O_ID_13	Index

第三節　跨網域處理機制

使用者與 Web Application 執行階段(Session)的信息特性，它無法直接的轉交或授予其他的 Web Application 進行存取。目前 PHP 的 Web Application 執行工作階段，可以透過一個參數設定，使所有在 PHP 執行階段內的所有 Web Application 進行存取與分享，但這個方法也只限定在這個 PHP 執行階段內所擁有的 Web Application，並不能真正跨出這個伺服器以及其它的網域下。因此要跨過網域傳遞 Session List 到其他的 Web Application 與使用者的執行階段，必須透過其他的方法。大部分開發者優先想到的便是利用 Cookies 來儲存 Session List。但多數的瀏覽器(如：IE 或 Firefox)對於 Cookies 有幾個條件上的限制：

(1)只允許每一個網站(Web Application)儲存 20 個 Cookies。

(2)部分瀏覽器對 Cookies 的數量加上絕對限制，總數不超過 300 個。

(3)Cookie 內容不可超過 4096 個位元組。

經由 RBAC 授權中心所產生的 Session List，無法預估一個用戶會有多少個角色以及相依可用元件。單一個網站只能使用 20 個 Cookies 顯然是不敷使用，雖然部分瀏覽器提供用戶可自行修改部分參數，但這就必須改變使用者經驗來達到系統需求。另外 Cookie 的安全性原

則，僅能存取自身網站或是允許的相依網域之下(例如：*.yahoo.com)，當使用者跨出域名之後，Cookie 就無權存取。因此使用 Cookie 作為跨網域的傳遞媒介不在此考慮，取而代之的是由伺服器建立一組唯一的雜湊碼(Session Client)，在 RBAC 授權中心建立起 Session List 時一併將 Session Client 遞送給使用者。

當使用者進入服務時，將檢驗使用者與 Web Application 工作執行階段(Session)是否擁有 Session List，並依據 Session List 來控制使用者是否有權限操作 Web Application 上的部分服務；當使用者與 Web Application 工作執行階段(Session)並不存在由 RBAC 授權中心提供的 Session List 時，將會檢查使用者是否存在著 Session Client，若使用者不存在 Session Client，表示使用者未曾登入過系統，或閒置時間過長以至於 Session Client 失效，此時將會引導使用者回到登入畫面。反之使用者不存在 Session List，卻擁有 RBAC 授權中心頒予的 Session Client，表示使用者已經跨離原本的 Web Application 工作執行階段，進入了另一個新的 Web Application 工作執行階段，因此 Web Application 將自行透過 Session Client 向 RBAC 授權中心重新取得使用者的授權關係表，RBAC 授權中心在取得 Web Application 的請求後，將會驗證 Session Client，並透過 XML 格式將完整的使用者關係授權表(Session XML)回傳給發出請求的 Web Application，讓 Web

Application 重新建立起與使用者的工作執行階段(Session)。跨網域請求使用者授權關係表流程圖請參閱圖 3.4，由 RBAC 授權中心回傳的 Session XML 請參閱表 3.2。

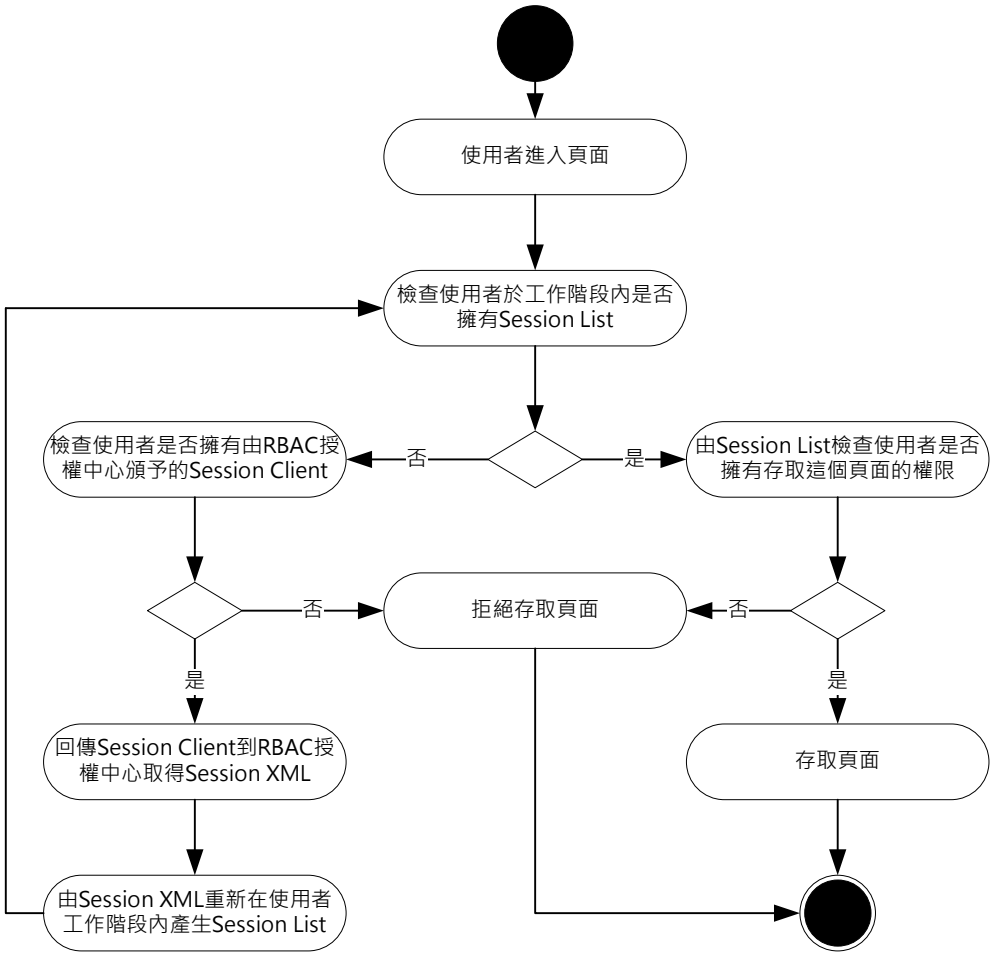

圖 3.4 跨網域請求使用者授權關係表流程圖

表 3.2 Session XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<PageAdmin>
    <User ID="ayu">
        <Name>亞由</Name>
        <Email>sfw.sakana@gmail.com</Email>
    </User>
    <Role Count="2">
        <RID>browser01</RID>
        <RID>sysadmin</RID>
    </Role>
    <Object Count="14">
        <OID>Admin_O2R</OID>
        <OID>Admin_Objects</OID>
        <OID>Admin_R2O</OID>
        <OID>Admin_R2U</OID>
        <OID>Admin_Roles</OID>
        <OID>Admin_U2R</OID>
        <OID>Admin_Users</OID>
        <OID>Logout</OID>
        <OID>O_List</OID>
        <OID>Radmin_EX01</OID>
        <OID>Session_List</OID>
        <OID>Session_XML</OID>
        <OID>Session_XML_Show</OID>
        <OID>Index</OID>
    </Object>
</PageAdmin>
```

第四節　其它跨網域機制比較

我們的目的是將 Web Application 中的資訊傳遞給其它的 Web Application 使用。除了本研究所提之跨網域處理機制外，目前亦有其它的跨網域機制處理辦法，其方法有三：

(1) Cookie：是儲存少量資訊於客戶端的機制，但由於 Cookie 本身的限制使得，欲遞送大量資訊顯得困難；若將資訊以批次的方式遞送，將使傳遞效率變差；直接修改客戶端瀏覽器軟體，將必須改變使用者經驗，且增加部分轉製成本。另外使用 Cookie 機制在跨網域部分僅限定於相同域名之下。

(2) Memcached：是分散式的快取系統，在實作的部分類似於 Session 的存取方式。使用 Memched 技術須架設 Memcached 伺服器以及改變部分系統原始碼，在建構上必須付出一定成本，且 Memcached 機制在安全上仍有顧忌及疑慮，由於企業界在建構 Web Services 已建置相關高安全性防火牆，因此該方法在企業界仍被廣泛使用。

(3) Other Agent：是基於兩種不同系統進行異質傳輸，而設計的代理程序。在運用上是將相關資訊傳遞給代理人程式，經由代理人程式與其它代理人溝通將資訊順利輸出。在建置上必須付出代理人程序製作之成本。

本研究機制，資訊的傳遞位於伺服器之間，無須改變使用者的使用經驗；在系統架構的部分，資訊的存取位於 Web Application 中的 Session，不需另外指向於其它存取位置上(Memcached 機制須將存取位置指向於 Memcached 伺服器)，由於資訊在傳遞時是使用 HTTP 通道，因此不需要透過第三方軟體代理人來傳輸，在傳輸的期間可透過 SSL 加密通道進一步的增強安全性，亦可對傳輸的內文進行加密。本研究機制與其它三種機制比較請參閱表 3.3。

表 3.3 相關機制比較表

	Cookie 方法	Memcached 技術	Other Agent	本研究機制
使用者經驗是否改變	是	否	否	否
是否需改變資訊存取方式	否	是	是	否
是否需另外開發第三方軟體	是	否	是	否
資料傳遞安全性	視遞送方式而定	須藉由其它方式加強	視 Agent 設計架構而定	視遞送方式而定
跨網域能力	僅於相同域名之下	可跨網域	可跨網域	可跨網域

第四章、實作設計

在這個章節裡，將透過 SQL 實作一個簡單的 RBAC 授權中心，以及基本的 RBAC 授權控制服務，另外也將實作跨網域存取物件的部分。

第一節　資料庫格式

從 RBAC 的基本模型中，可以知道它是由基本的三個部件所組成的：使用者(User)、角色(Role)、物件(Object)，因此在資料庫裡優先建立起這三張資料表，建立 RBAC 最低需求資料表可參閱表 4.1、4.2、4.3，實際運用可視實際需求調整資料表內的資訊規格，如表 4.1 的使用者資料表，可視需求增加信箱資訊(Email)、地址資訊(Address)、電話資訊(TEL)…等。

表 4.1 使用者資料表

名稱	說明
ID	唯一值索引鍵，作為用戶帳號使用
Password	密碼，作為用戶登入驗證之用
Name	易於辨識的使用者名稱

表 4.2 角色資料表

名稱	說明
ID	唯一值索引鍵，作為角色 ID 使用
Name	易於辨識的角色名稱

表 4.3 物件資料表

名稱	說明
ID	唯一值索引鍵，作為物件 ID 使用
Name	易於辨識的物件名稱

建立好 RBAC 的基本元素後，接著建立兩兩相依的關係表，分別是使用者與角色之間的關係表，角色與物件之間的關係表，這兩張關聯資料表請參閱表 4.4 與 4.5。

表 4.4 使用者與角色關係資料表

名稱	說明
U_ID	用戶 ID。與 R_ID 成為雙索引
R_ID	角色 ID。與 U_ID 成為雙索引

表 4.5 角色與物件關係資料表

名稱	說明
R_ID	角色 ID。與 O_ID 成為雙索引
O_ID	物件 ID。與 R_ID 成為雙索引

使用者帳戶 ID 與「使用者與角色關係資料表」中的 U_ID 互相關聯，角色資料表中的 ID 與「使用者與角色關係資料表」的 R_ID 互相關聯。角色資料表的 ID 與「角色與物件關係資料表」中的 R_ID 互相關聯，物件資料表中的 ID 與「角色與物件關係資料表」的 O_ID 互相關聯；如此般使用者便能透過帳戶 ID 透過「使用者與角色關係資料表」與「角色與物件關係資料表」取得相應的物件 ID，詳細的資料表關連圖請參閱圖 4.1 所示。

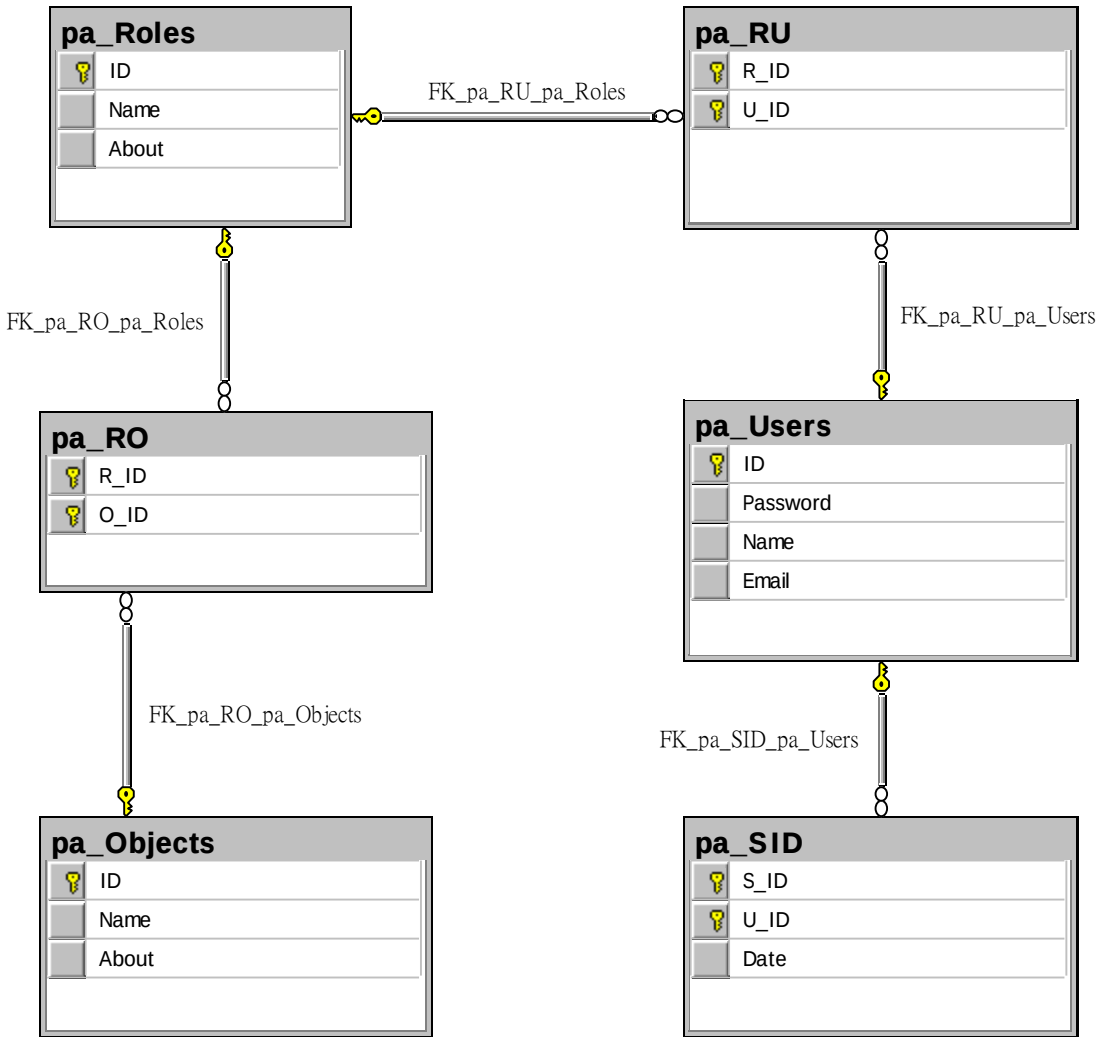

圖 4.1 資料庫關聯圖表

第二節　預存程序

預存程序是 MS-SQL 特有的，其程式語言又稱為 T-SQL，它的好處是將複雜的資料搜尋、處理、過濾，可以優先的從 SQL 中處理好，再展示給前端的開發語言；並非由前端的開發語言逐步得處理資料庫內的檢索、過濾、搜尋等步驟。屆時系統發完成後，若因為有特殊需求改變資料庫內容，僅需修正部分的預存程序即可完成欲更變的內

容，而無需一併修改前端的開發程式。在此舉出一個簡單的情境，當非使用預存程序所設計的系統在完成之後，想要增加一個具有紀錄使用者所有操作紀錄的資料表；雖然資料庫製表容易，但是前端所有的應用程序必須重新改寫增加一個具有 Log 能力的函式。使用者有多少的操作能力，就必須補充多少的相關程式碼。若使用預存程序，只需要預存程序中加入對資料表 Log 的能力就能解決此狀況，而無須再修正前端的應用程式。

在本系統添入幾個基本的預存程序，包含：使用者登入(User_Login)、角色列表(User_RoleList)、物件列表(User_ObjectList)、Session Client 增加(SID_Add)、Session Client 驗證(SID_Check)、Session Client 刪除(SID_Del)。請分別參閱表 4.6、表 4.7、表 4.8、表 4.9、表 4.10、表 4.11。其它如，增刪修使用者、增刪修角色、增刪修物件…皆可撰寫於預存程序中，由前端程式直接呼叫使用。

表 4.6 使用者登入(User_Login)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[User_Login]  指令碼日期: 03/17/2008 13:19:08
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[User_Login]
        -- Add the parameters for the stored procedure here
        @ID varchar(10),
        @Password varchar(50)
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

                SELECT          ID, Name, Email
                FROM            pa_Users
                WHERE       (ID = @ID) AND (Password = @Password)
END
```

表 4.7 角色列表(User_RoleList)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[User_RoleList]  指令碼日期: 03/17/2008 13:19:33
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[User_RoleList]
      -- Add the parameters for the stored procedure here
      @U_ID varchar(10)
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;

     -- Insert statements for procedure here
      SELECT       R_ID
      FROM            pa_RU
      WHERE       (U_ID = @U_ID)
END
```

表 4.8 物件列表(User_ObjectList)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[User_ObjectList]    指令碼日期: 03/17/2008 13:47:30
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[User_ObjectList]
      -- Add the parameters for the stored procedure here
@U_ID varchar(10)
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;

     -- Insert statements for procedure here
      SELECT DISTINCT pa_RO.O_ID, pa_Objects.Name
      FROM            pa_RO INNER JOIN
                              pa_RU ON pa_RO.R_ID = pa_RU.R_ID INNER JOIN
                              pa_Users ON pa_RU.U_ID = pa_Users.ID INNER JOIN
                              pa_Objects ON pa_RO.O_ID = pa_Objects.ID
      WHERE       (pa_Users.ID = @U_ID)
END
```

表 4.9 Session Client 增加(SID_Add)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[SID_Add]    指令碼日期: 03/17/2008 13:48:22 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[SID_Add]
        -- Add the parameters for the stored procedure here
        @U_ID varchar(50),
        @S_ID varchar(50)
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- 刪除
        EXECUTE SID_Del

        -- 刪除重複登入
        DELETE FROM pa_SID
        WHERE        (U_ID = @U_ID)

      -- Insert statements for procedure here
        INSERT INTO pa_SID(U_ID,S_ID)
                VALUES(@U_ID,@S_ID)

END
```

表 4.10 Session Client 驗證(SID_Check)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[SID_Check]  指令碼日期: 03/17/2008 13:49:22
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[SID_Check]
      -- Add the parameters for the stored procedure here
      @S_ID varchar(50)
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;

     -- Insert statements for procedure here
      SELECT       S_ID, U_ID
            FROM            pa_SID
            WHERE       (S_ID = @S_ID)

      -- 刪除
      EXECUTE SID_Del
END
```

表 4.11 Session Client 刪除(SID_Del)預存程序

```
USE [PageAdmin_02]
GO
/****** 物件:  StoredProcedure [dbo].[SID_Del]    指令碼日期: 03/17/2008 13:50:08 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[SID_Del]
      -- Add the parameters for the stored procedure here

AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;

     -- Insert statements for procedure here
      DELETE FROM pa_SID
      WHERE       (DATEDIFF(n, Date, { fn NOW() }) >= 30)
END
```

第三節　前端後台介面

前端的後台介面功能包含：新增使用者、刪除使用者、新增角色、刪除角色、新增物件、刪除物件、使用者與角色關連、角色與物件關連。在物件的部分，是將 Web Application 裡的每一個網頁都是為物件，透過角色來區分，哪些網頁(物件)可以被哪些角色所存取。使用者登入介面，請參閱圖 4.2，使用者登入介面除了可以由此頁進行登入外，也可以從其它設計較為美觀的介面傳遞參數進入。增刪使用者介面請參閱圖 4.3，增加使用者除了可以由此後台操作，亦可由較為美觀的介面傳遞參數操作。增刪角色介面請參閱圖 4.4。增刪物件介面請參閱圖 4.5；增刪角色與物件關係的方式可由兩個面向進行，一是由角色為基準加入到物件，該部分請參閱圖 4.6，另一是由物件為基準加入到角色，該部分請參閱圖 4.7。增刪用戶與角色關係的方式也由兩個面向進行，一是由角色為基準配置到指定的用戶上，該部分請參閱圖 4.8，另一是由用戶為基準指向到特定的角色上，該部分請參閱圖 4.9。

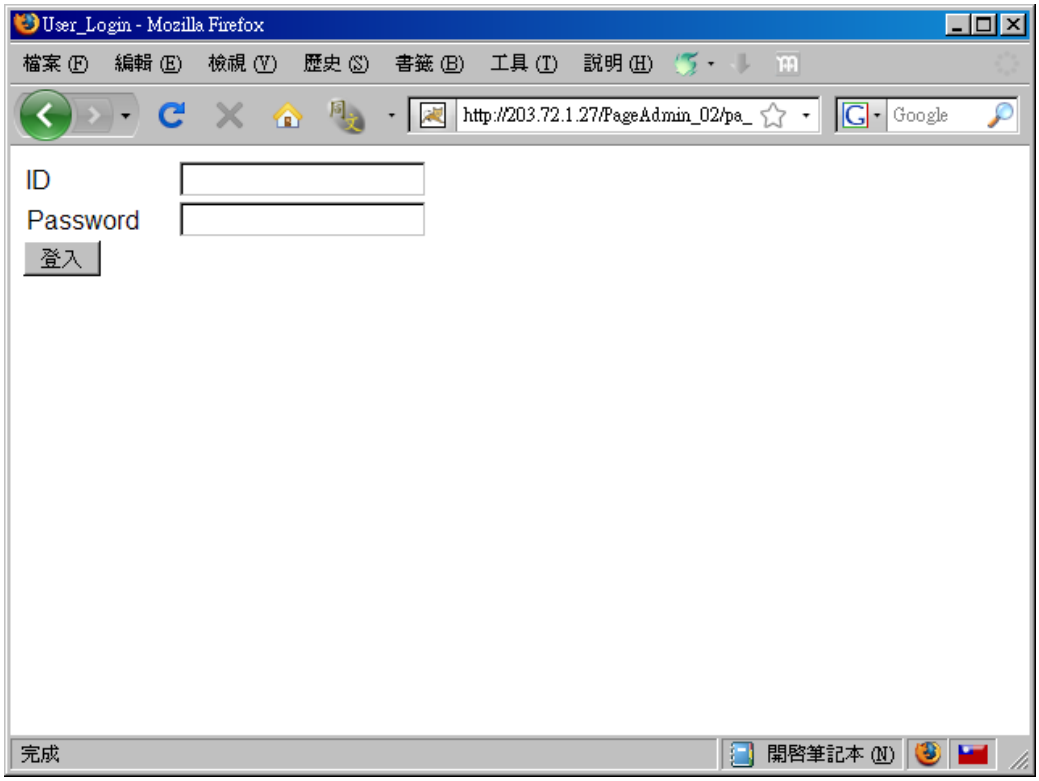

圖 4.2 使用者後台登入介面

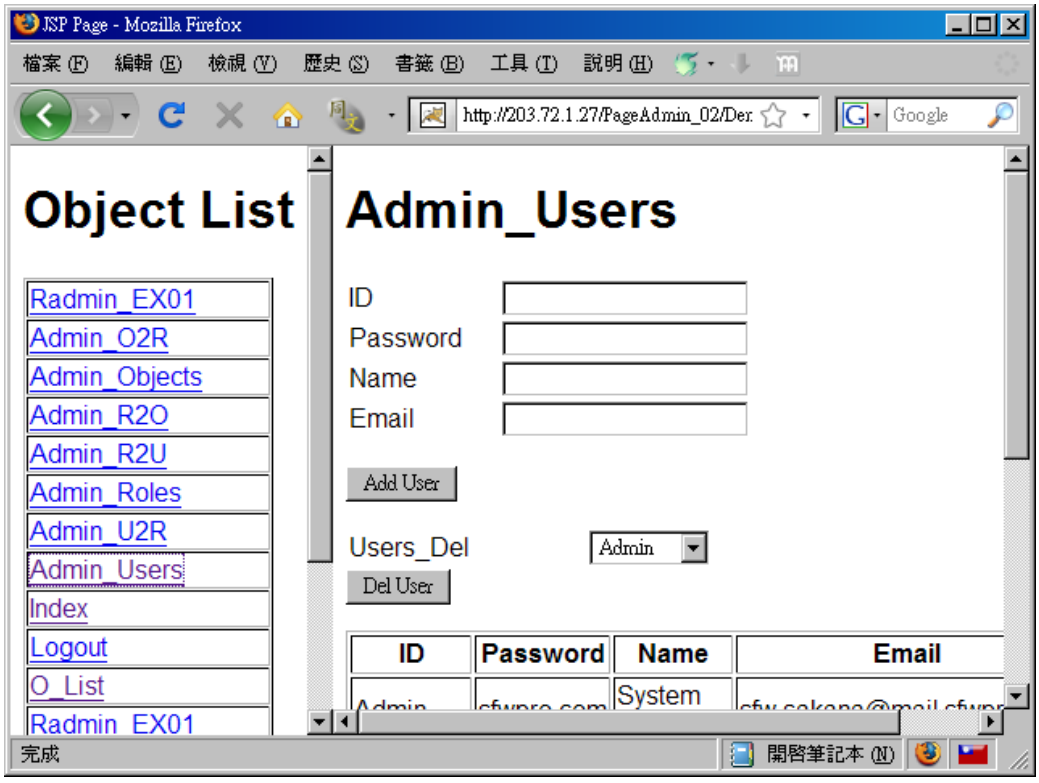

圖 4.3 增刪使用者後台介面

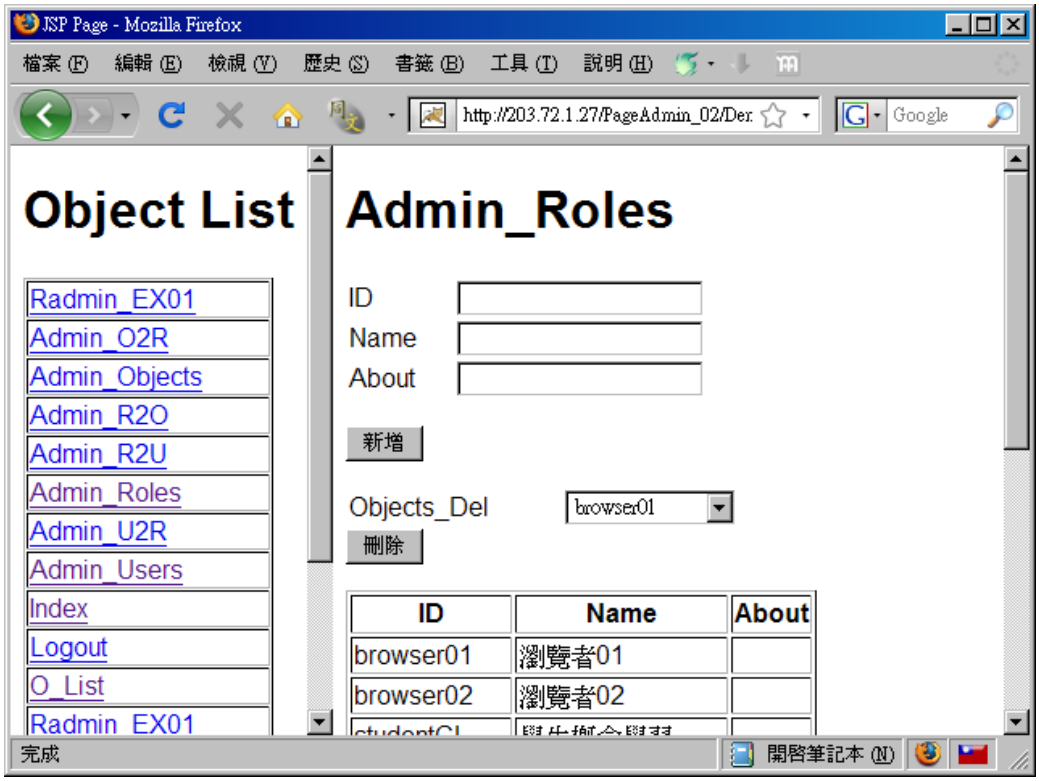

圖 4.4 增刪角色後台介面

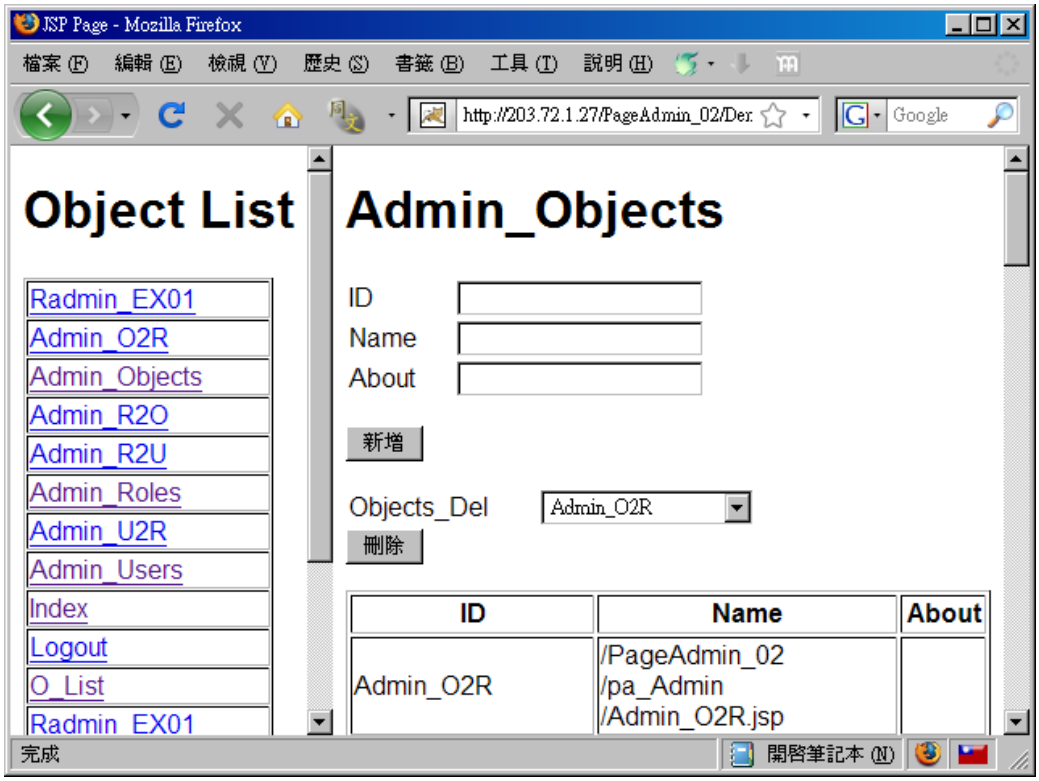

圖 4.5 增刪物件後台介面

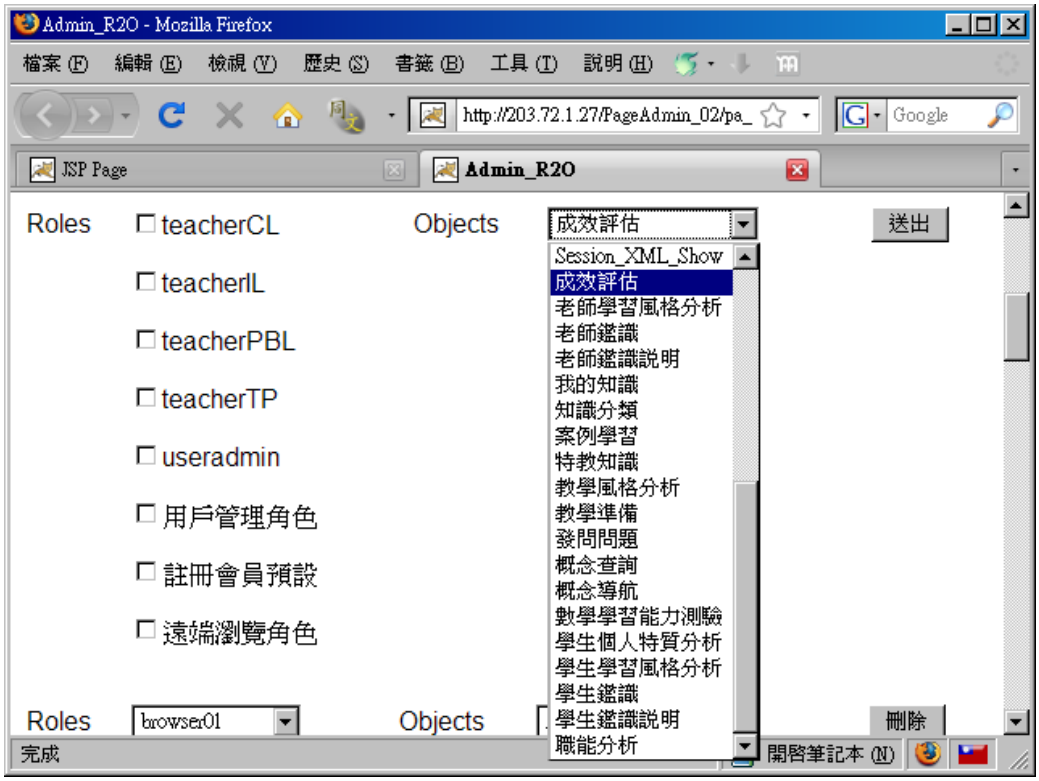

圖 4.6 由角色為基準加入到物件

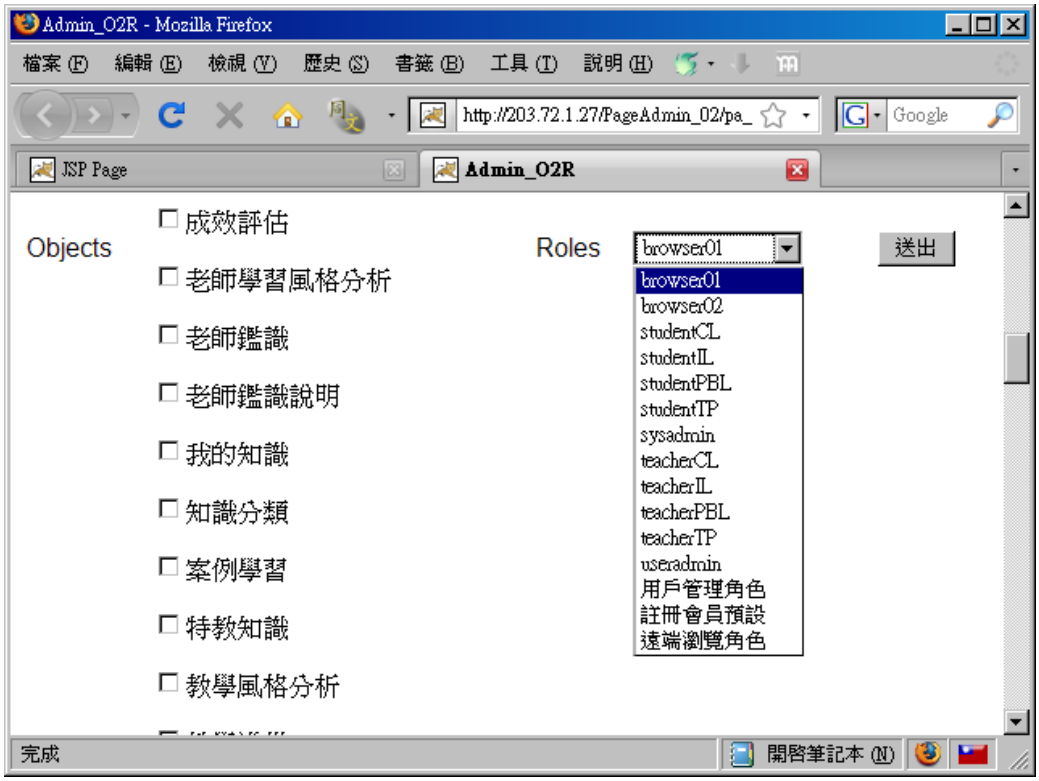

圖 4.7 由物件為基準加入到角色

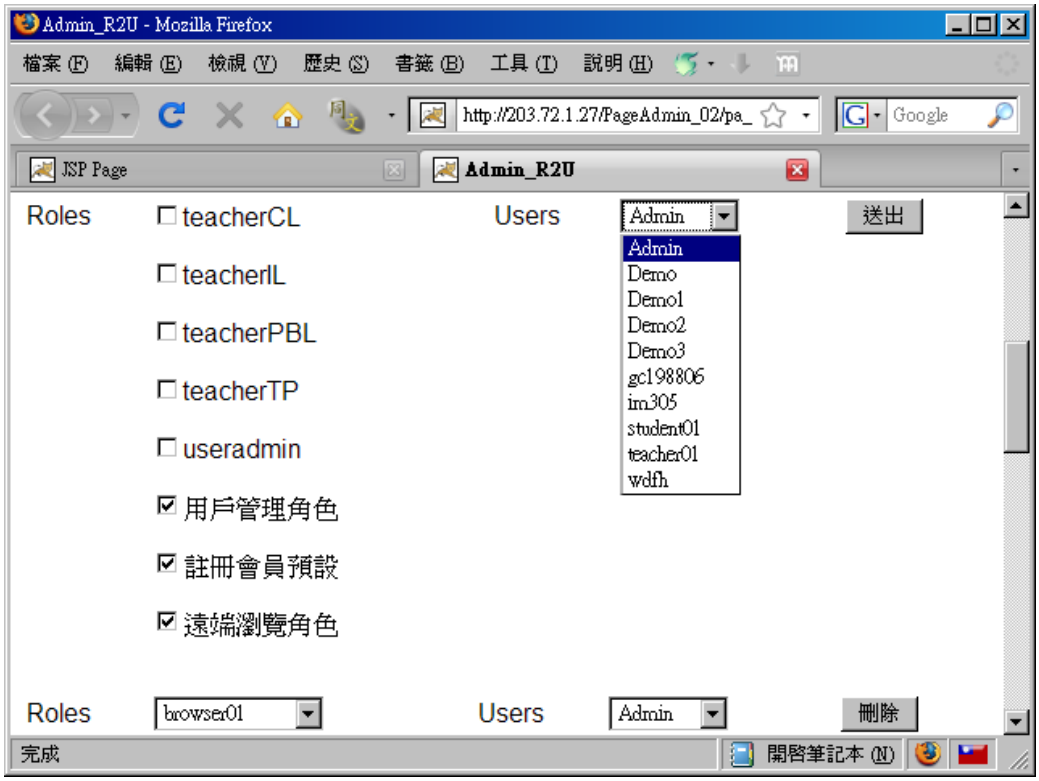

圖 4.8 由角色為基準配置到用戶

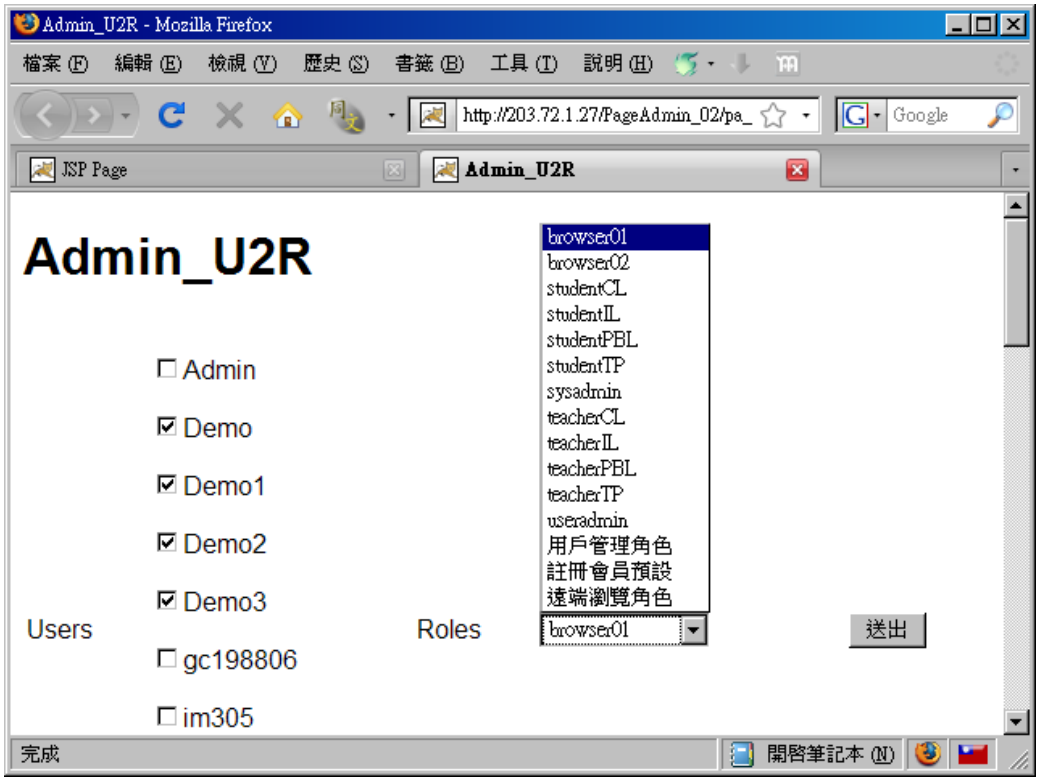

圖 4.9 由用戶為基準指向到角色

第四節　實際建置

本章節實際建置環境包含一個 RBAC 授權中心與兩個 Web Application 環境。

(1) RBAC 授權中心：

位於 203.72.1.127，透過標準通訊協定與 MS-SQL 溝通，用以驗證使用者帳戶及列舉用戶合法權限。

(2) Web Application A：

位於 203.72.1.27，這裡放置後台服務介面，用以操作 RBAC 權限配置等服務。

(3) Web Application B：

位於 203.72.1.59，建置一個 Web Application，並可由遠端 RBAC 授權控制用戶是否有權讀取。

以上列舉三個服務，皆位於不同的 IP 上，且未設置域名，可視為不同域名。首先將所有的 Web Application 的網頁作為物件，依序加入 RBAC 授權中心之物件表內。建立三個角色分別為系統管理員(sysadmin)、註冊會員(users)、遠端服務瀏覽(browser01)，「系統管理員角色」即為可以操作後台服務，包含控制 RBAC 授權調配；「註冊會員角色」即為可進一步操作瀏覽系統所建置的服務，在使用者帳戶

建置時即可預先配置該權限於預存程序內；「遠端瀏覽角色」即是位於其它位置上的 Web Application 是否授予存取之能力。建立兩組可用的合法帳戶分別為 Demo1 與 Demo2，「Demo1」帳戶配置「用戶管理角色」權限，「Demo2」帳戶配置「用戶管理角色」與「遠端瀏覽角色」兩權限，分別測試登入系統所能操作之能力。已知下列三個連結，分別為後台登入首頁、會員管理介面、遠端測試介面。

(A) 後台登入介面：

http://203.72.1.27/PageAdmin_02/pa_index.jsp

(B) 會員管理介面：

http://203.72.1.27/PageAdmin_02/pa_Admin/Admin_Users.jsp

(C) 遠端測試介面：

http://203.72.1.59:8080/EX01/pa_Radmin/EX01.jsp

在未登入狀況下進入「會員管理介面」與「遠端測試介面」，由於使用者與 Web Application 工作執行階段內並沒有 Session List，且沒有 RBAC 授權中心頒予的 Session Client，因此會被轉送於錯誤警示畫面，該畫面請分別參閱圖 4.10、圖 4.11。

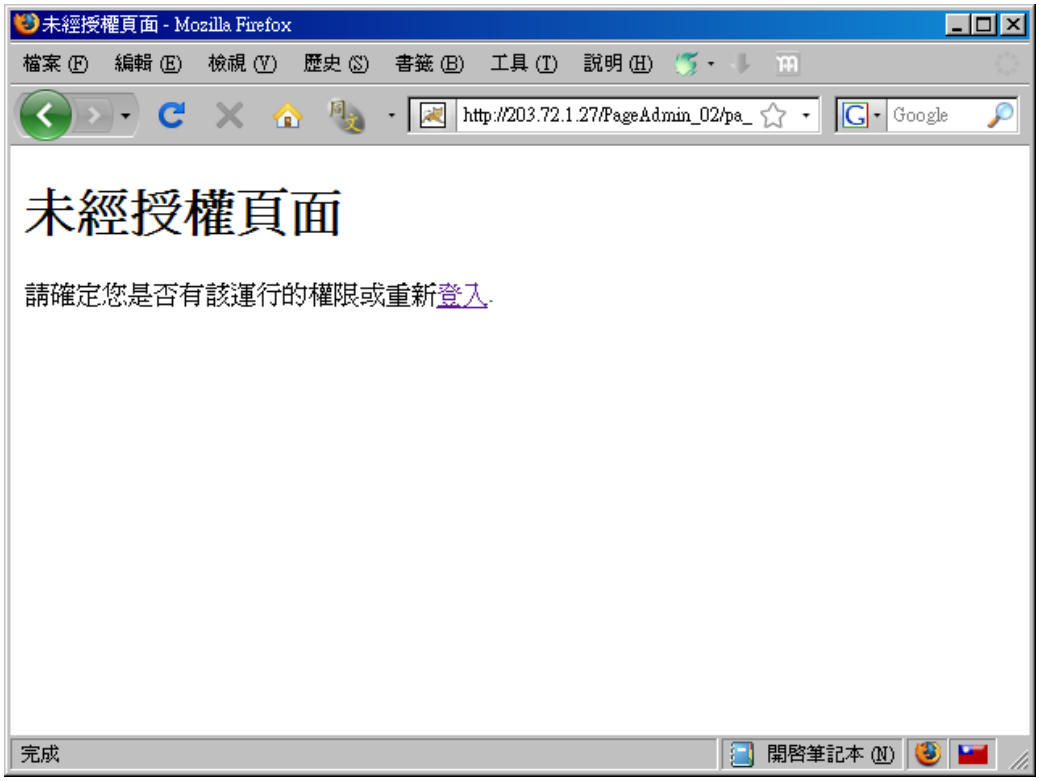

圖 4.10 未授權轉入頁面於 203.72.1.27 伺服器

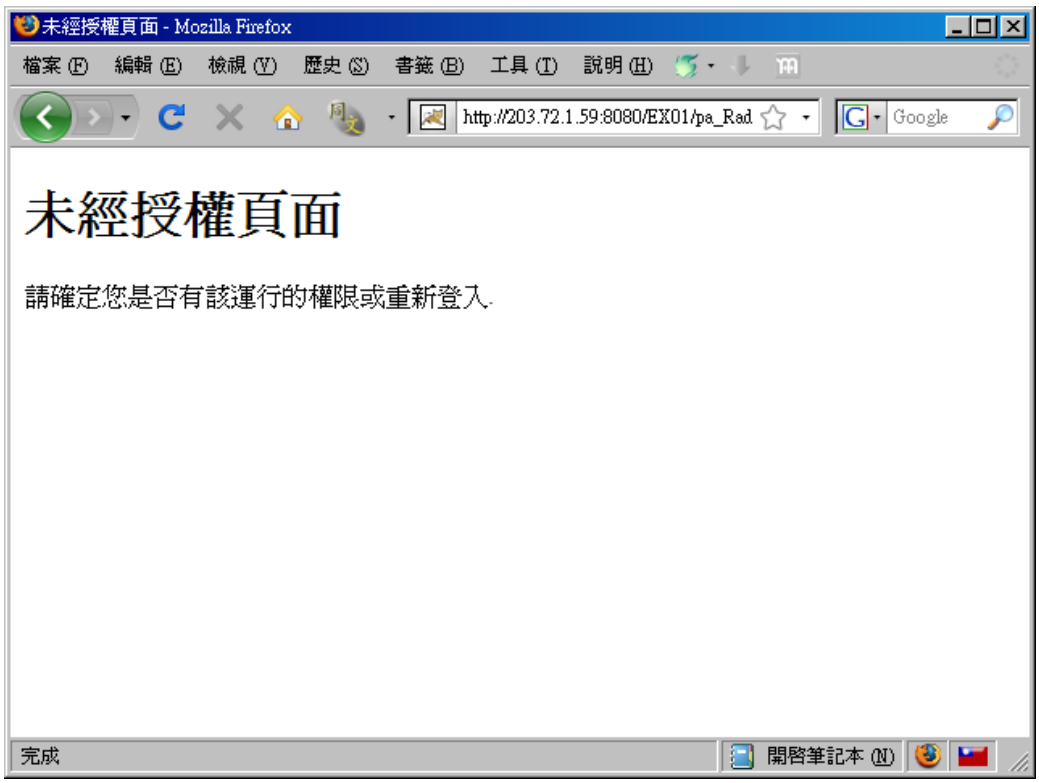

圖 4.11 未授權轉入頁面於 203.72.1.59 伺服器

使用「Demo1」帳戶登入後，請參閱圖 4.12 在左邊的頁框可以看到該用戶所能操作的權限，包含管理使用者帳戶(Admin_Users)、登出後台(Logout)、列出可用物件(O_List)等三種權限。點擊「Admin_Users」將進入帳戶管理介面，請參閱圖 4.13。

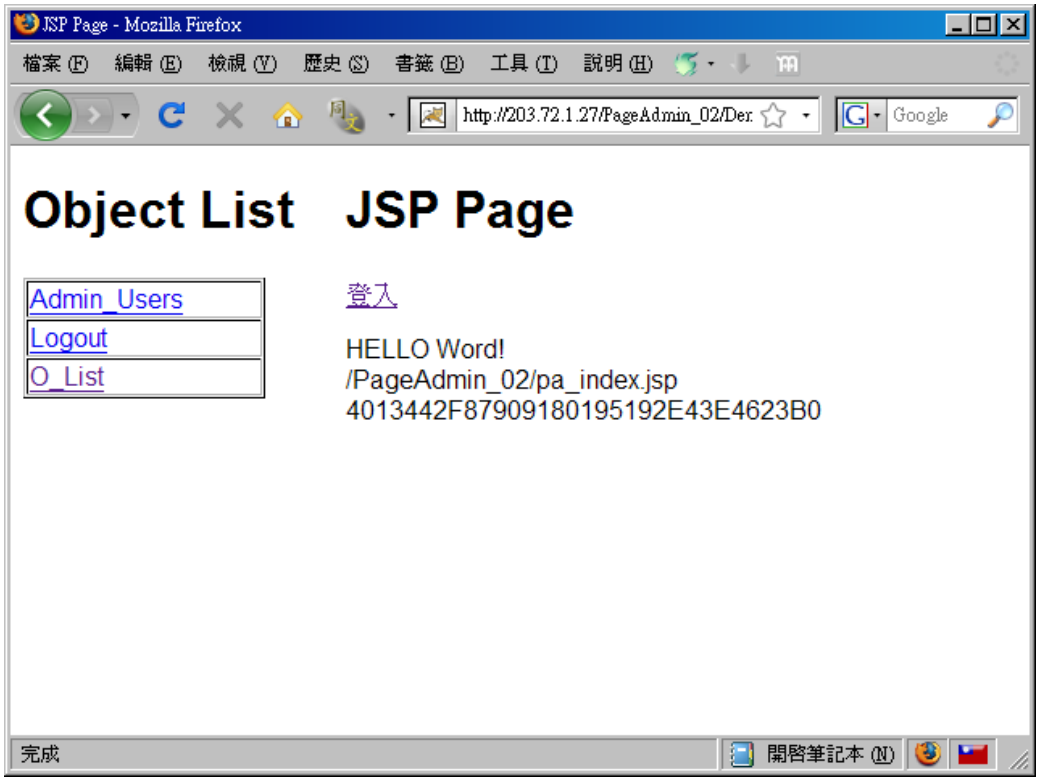

圖 4.12 Demo1 帳戶登入畫面

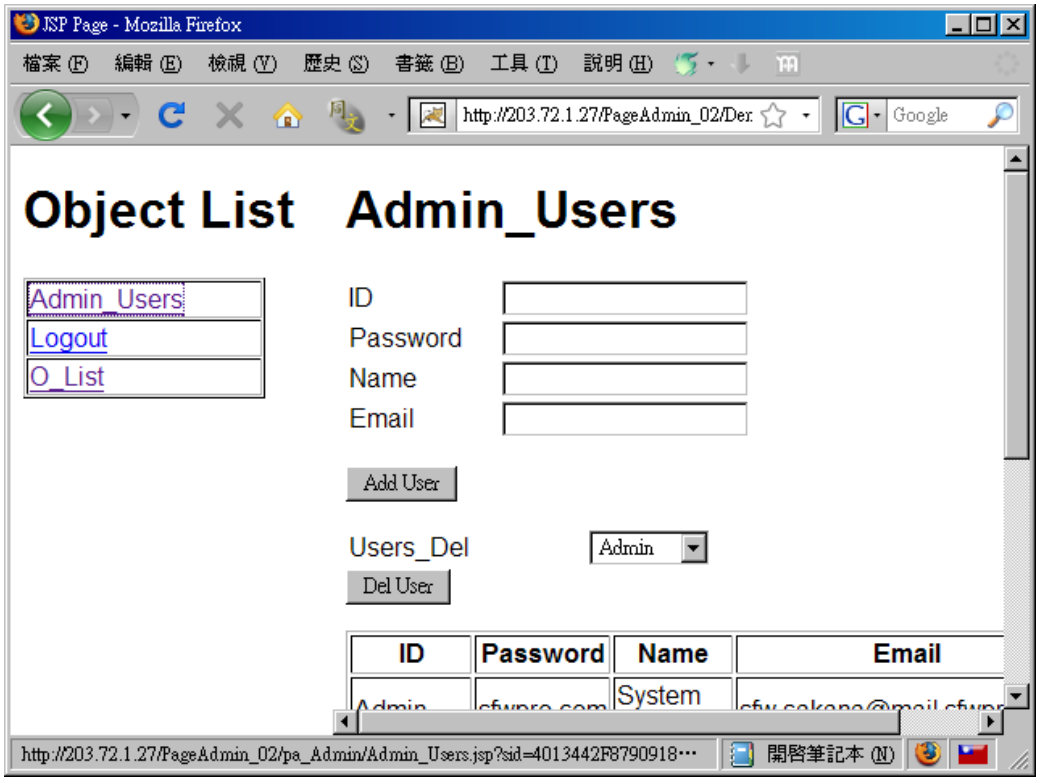

圖 4.13 操作帳戶管理介面

由 Object List 這個頁框可以得知，Demo1 帳戶並沒有遠端瀏覽的角色權限，但是已知遠端測試介面在 http://203.72.1.59:8080/EX01/pa_Radmin/EX01.jsp；在網址列輸入嘗試連線後，會發現網頁被重新導入至未授權介面，請參閱圖 4.11，因為遠端測試介面透過 Session Client 與 RBAC 授權中心取得 Session XML，並重新建置一份使用者與 Web Application 工作階段的 Session List 後，並未檢測到使用者含有操作遠端測試介面之權限，所以轉向於未授權警示介面。

將「Demo1」帳戶登出後改由「Demo2」登入，請參閱圖 4.14，由 Object List 頁框上可以看到多了遠端測試介面(Radmin_EX01)，點擊後將會轉入 http://203.72.1.59:8080/EX01/pa_Radmin/EX01.jsp 的位置，請參閱圖 4.15。遠端測試介面獨立開啟請參閱圖 4.16。

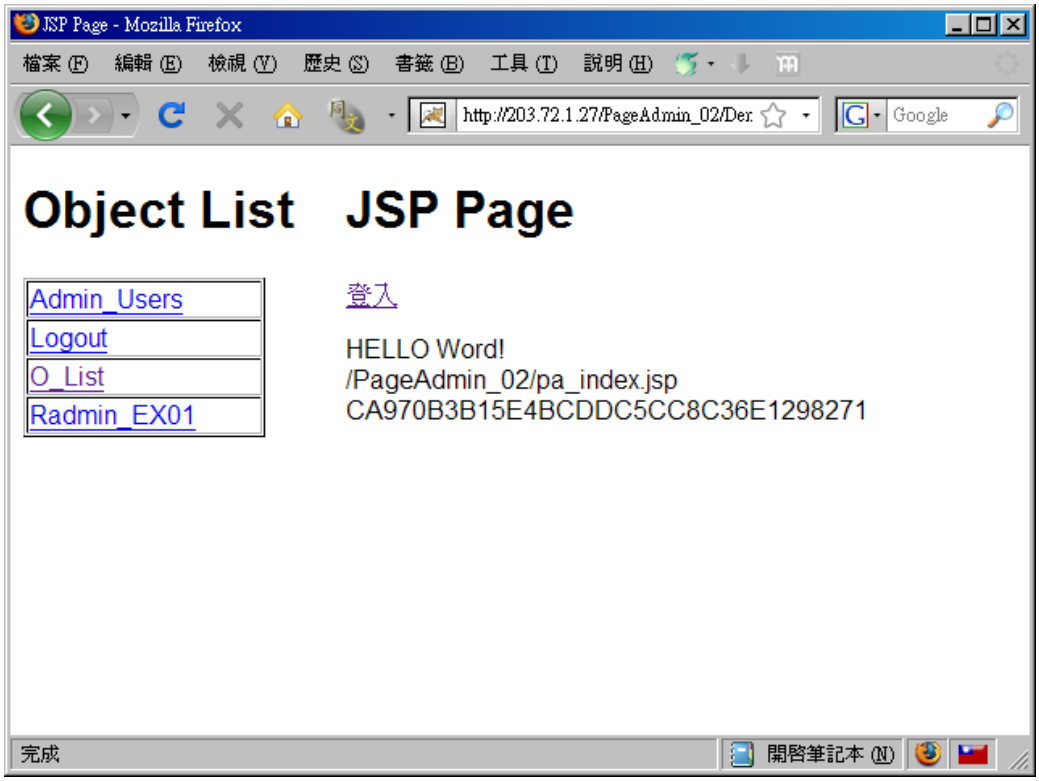

圖 4.14 Demo2 帳戶登入畫面

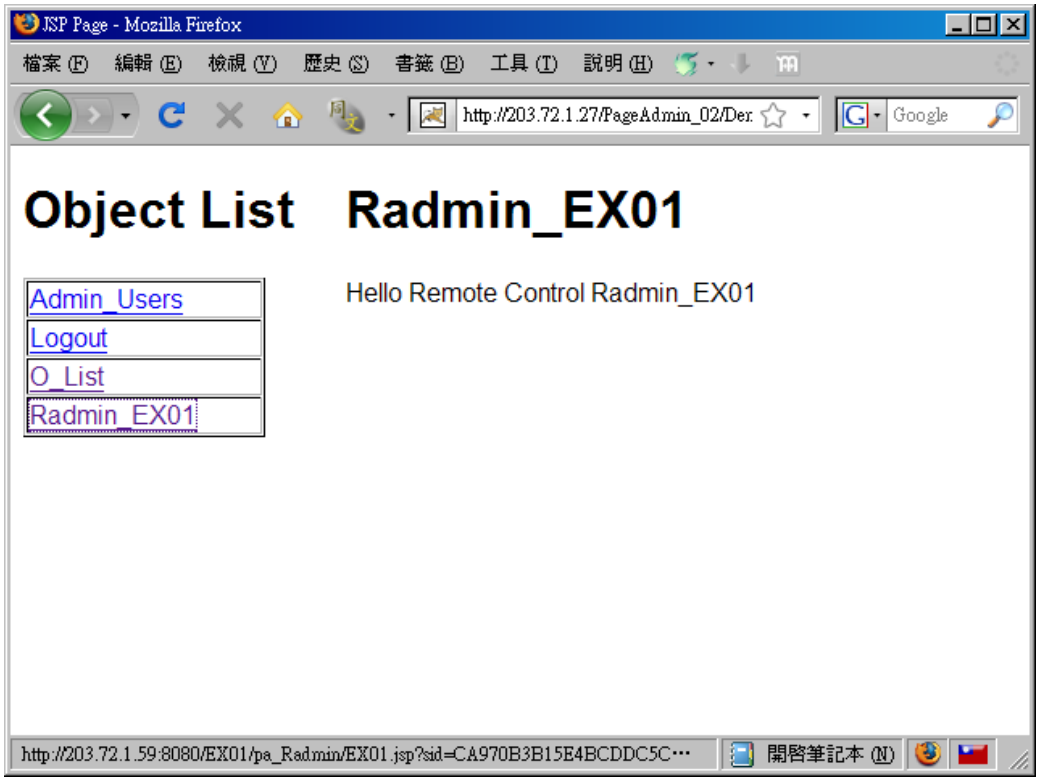

圖 4.15 遠端測試介面

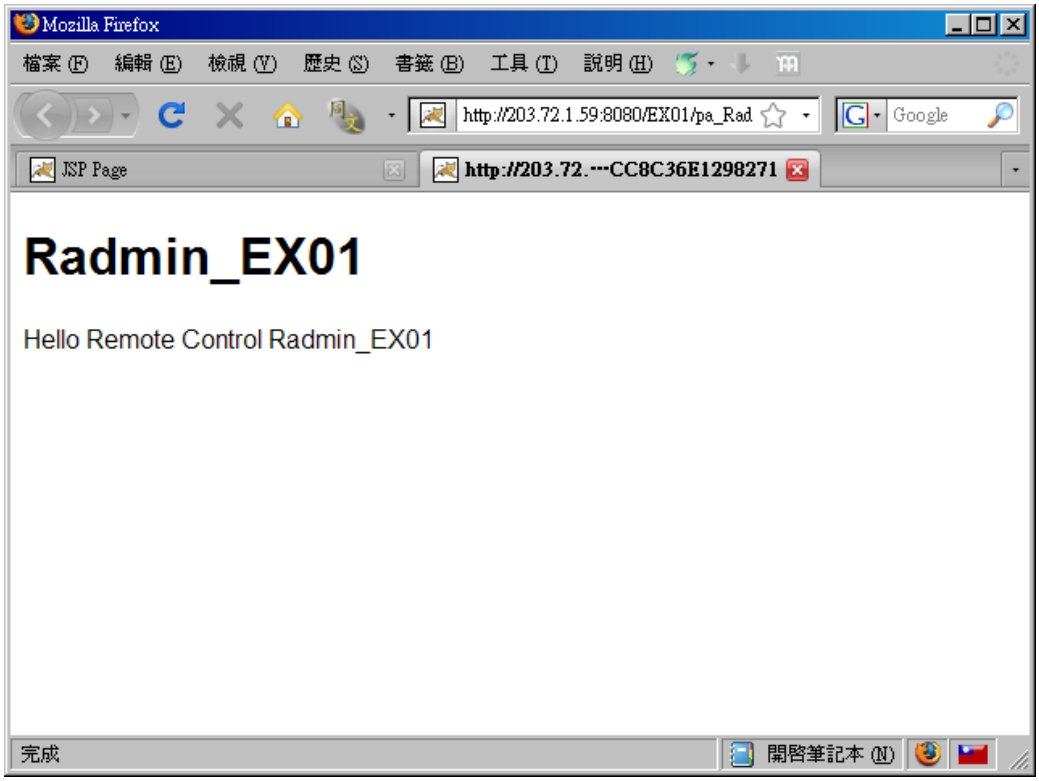

圖 4.16 遠端測試介面獨立頁面

第五節　演算流程

本研究機制演算流程可區分為兩大部分，(1)使用者在成功登入後，由 RBAC 授權中心取得因應的角色和權限，以及(2)使用者進入網頁時判別使用者是否具有權能操作，這個部分涵蓋至跨越網域識別使用者權限之能力。

壹、由 RBAC 授權中心資訊取得演算流程

這個部分的演算流程，係指使用者經過身分確認完成後，由 RBAC 授權中心傳遞因應的角色和權限給使用者。其中包含的副程式有取得角色、取得物件、建立 Session List。

1. 副程式：由 RBAC 授權中心取得可用的角色資訊

   由伺服器向 RBAC 授權中心取得使用者可用角色，透過使用者帳戶進入預存程序可取得一角色陣列。在函式過程中帶入使用者帳號與密碼可加強安全性驗證；該部分請參閱表 4.12。

表 4.12 副程式：Get_RBAC_Role

```
Function Get_RBAC_Role[](User_ID, User_Password) as String {
    If RBAC(User_ID, User_Password) = False then
        Print “User_ID or User_Password is Error.”
        Exit Function
    End if
    Return RBAC(User_ID).getRole
}
```

2. 副程式：由 RBAC 授權中心取得可用的物件資訊

由伺服器向 RBAC 授權中心取得使用者可用物件，透過使用者帳戶進入預存程序，將會取得使用者角色陣列，再由使用者角色陣列取得使用者可用物件；該部分請參閱表 4.13。

表 4.13 副程式：Get_RBAC_Object

```
Function Get_RBAC_Object[](User_ID, User_Password) as String {
    If RBAC(User_ID, User_Password) = False then
        Print “User_ID or User_Password is Error.”
        Exit Function
    End if
    Return RBAC(User_ID).getObject
}
```

3. 副程式：使用者登入後建立 Session List

使用者在通過身分確認後，將透過 RBAC 授權中心取得可用角色及物件，將這些資訊放入 Web Application 中的執行工作階段中，以方便進行可用權限的比對；該部分請參閱表 4.14。

表 4.14 副程式：Set_Session_List

```
Function Set_Session_List(){
    Dim Role[] <- Get_RBAC_Role(User_ID, User_Password)
    Dim Object[] <- Get_RBAC_Object(User_ID, User_Password)
    If Role.Count or Object.Count = 0 then
        Print “User Role or Object can’t get.”
        End Function
    End if
    Session <- RBAC(User_ID, User_Password).getUserProFile
    Session <- Role.Count
    Session <- Role[]
    Session <- Object.Count
    Session <- Object[]
}
```

4. 主程式：使用者登入

使用者登入進行身分確認，係將使用者帳號與密碼傳遞至 RBAC 授權中心，由資料中的預存程序進行身分確認；驗證

成功回傳 True，驗證失敗回傳 False。在通過身分確認完畢後，即開始建立 Session List，其過中包含向 RBAC 授權中心取得該使用者之角色與物件資訊；該部分請參閱表 4.15。

表 4.15 主程式：使用者登入

```
<%
    If RBAC(User_ID, User_Password) = False then
        Print “User_ID or User_Password is Error.”
        Goto ErrorPage()
    Else
        Goto Set_Session_List()
        Goto NextPage()
    End if
%>
```

由 RBAC 授權中心取的使用者角色與物件相關資訊，主程式與副程式之間互動過程，完整演算流程請參閱圖 4.17。

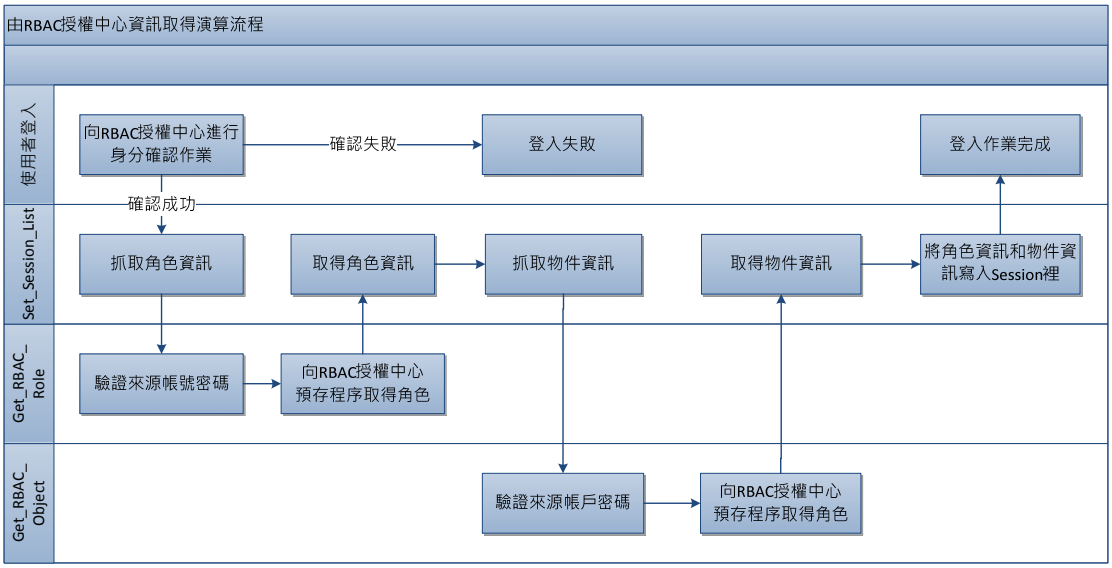

圖 4.17 由 RBAC 授權中心資訊取得演算流程

貳、權限判斷能力演算流程

使用者在進入一個網頁必須經過權限的判斷，若使用者的 Session List 帶有該網頁的 Title 名稱，即表示使用者擁有讀取這個網頁的能力。其中所包含的副程式有檢查 Session List 是否存在、檢查 Session Client 是否存在、以及借由 Session Client 取得 Session XML 轉換為 Session List。

1. 副程式：檢查 Session List 是否存在

   Session List 是在使用者登入系統後，由 RBAC 授權中心回傳之角色與物件資訊儲存於使用者與伺服器執行工作階段的清單。我們可以藉由讀取特定的 Session 以判定使用者是否曾經順利的完成整個登入作業過程。由於無法確認每個使用者會

取得的特定角色與物件，因此藉由登入作業時將使用者的部分資訊一併寫入 Session List 中，除了可以作為 Session List 是否存在外，也能方便其它的 Web Application 進行更人性化的介面撰寫 (例如：Session List 中存在使用者名稱(yuuchilyann)，在使用者操作某一服務，該服務可以由 Session List 取得使用者名稱並於畫面中顯示：歡迎 yuuchilyann 使用者操作本服務)；該部分請參閱表 4.16。

表 4.16 副程式：Check_SessionList

```
Function Check_SessionList() as Boolean {
    If Session(ID)="" or Session(ID) Is Nothing then
        Return False
    Else
        Return True
    End if
}
```

2. 副程式：檢查 Session Client 是否存在

Session Client 是在使用者登入系統後，在 RBAC 授權中心回傳角色與物件資訊的同時，透過預存程序調配一組雜湊值一併地回傳給使用者，並記錄於使用者的客戶端軟體上。當 Web

Application 讀取 Session List 失敗時，將會確認 Session Client 是否存在，若存在將可使用 Session Client 來取得 Session XML；該部分請參閱表 4.17。

表 4.17 副程式：Check_SessionClient

```
Function Check_SessionClient() as Boolean {
    If Client.Software(Session_Client)="" or _
        Client.Software(Session_Client) Is Nothing
        Return False
    Else
        Return True
    End if
}
```

3. 副程式：由 Session XML 轉換為 Session List

當使用者進入某一服務時，Web Application 無法檢測到 Session List 的存在，意味者使用者可能尚未進行登入作業過程，或使用者跨越至別的伺服器，以至於伺服器與使用者執行工作階段內並無 Session List 的存在。因此必須透過 Session Client 的存在來識別使用者是否曾經進行過登入作業過程；若使用者擁有 Session Client 即可透過該雜湊值傳遞給 RBAC 授權中心，以取得完整的 Session XML，該 Session XML 內容

包含完整的使用者角色、物件、相關資訊等。在順利取得 Session XML 之後即能重新的轉換為 Session List；該部分請參閱表 4.18。

表 4.18 副程式：Get_Session_XML

```
Functuin Get_Session_XML(Session_Client){
    Dim XML = RBAC(Session_Client).getXML
    Session <- XML.Role.Count
    Session <- XML.Role[]
    Session <- XML.Object.Count
    Session <- XML.Object[]
}
```

4. 主程式：判斷使用者是擁有操作權限

當使用者進入一項服務介面時，主要的目標任務即是判斷使用者，是否具備該服務的操作權限。使用者所有的角色資訊與物件資訊皆存放於 Session List 中，若 Session List 中並無該服務的名稱，即視為使用者無此服務的操作能力。當中延伸出判別使用者是否曾進行過登入作業過程，以及使用者是否跨越至其它伺服器操作服務；該部分請參閱表 4.19。

表 4.19 主程式：判斷使用者是否擁有操作權限

```
<%
    If Check_SessionList = False then
        If Check_SessionClient = False then
            Goto Login()
        Else
            Goto Get_Session_XML(Session_Client)
            Reload.Page()
        End if
    End if
    If Page.Title != Session then
        Goto ErrorPage()
    End if
%>
```

使用者進入服務時進行權限判斷能力，主程式與副程式之間互動演算流程請參閱圖 4.18。

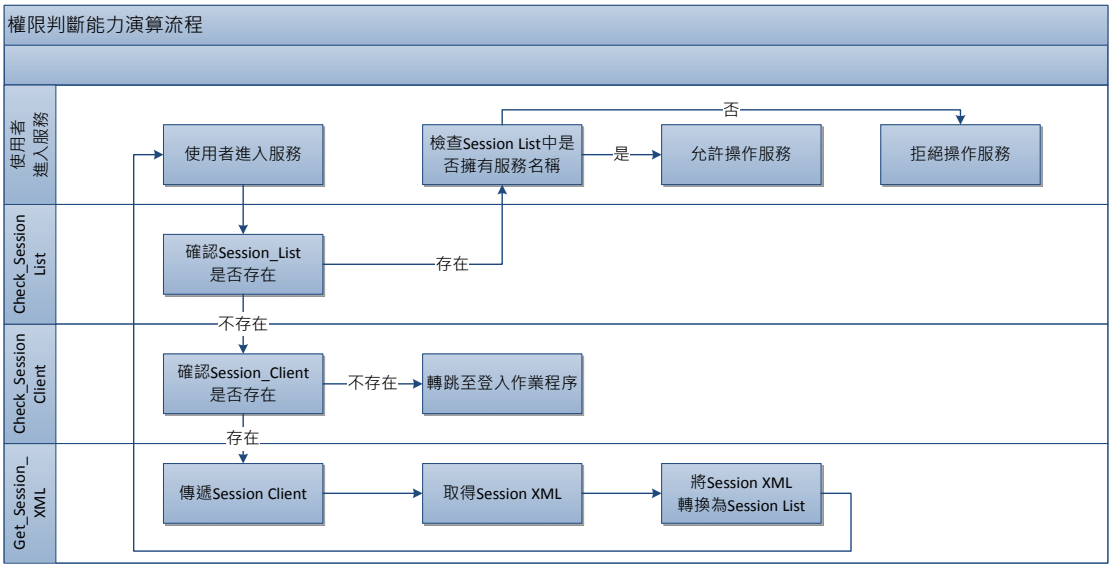

圖 4.18 權限判斷能力演算流程

第五章、結論

支援跨網域之網頁應用程式，在以往的設計可以透過下面幾種方法達成：

(1)利用 Cookie 來攜帶資訊，但它只能攜帶少量的資訊，雖然可以藉由修改客戶端瀏覽器來達到更大量的資訊攜帶，但這必須改變使用者經驗。除此之外，Cookie 的存取被限制只能在某一設定的域名內。

(2)利用 Memcached 與 Other Agent 的方式，藉由改變網頁應用程式存取資訊的位置，指向到一特定位置上進行存取。這個方法必須修改原本程序對於資訊存取的方式，如果原先開發某個物件的團隊已不存在，或某物件經過封裝後，無法再進行程式碼的修訂，則開發團隊群必須面臨重新開發該物件的窘境。

(3)以使用 ASP.net 所提供的單一簽入技術為例，這裡先忽略當前的單一簽入技術，還未能對每一個物件進行低階控管的問題。採用這個方法達到支援跨網的網頁應用程式的先決條件，是所有的開發團隊必須統一使用 ASP.net。假設部分的開發團隊並不熟悉 ASP.net 的開發語言，則必須花費許多成本來解決這個問題，亦可能降低開發效率。

為避免因上述所造成的困擾，本研究機制提出一個有效的解決辦法。

多數的網頁應用程式在撰寫「判讀使用者權限」程序時，皆是把使用者權限存放在 Session 內；本研究機制在客戶端與伺服器間的權限判讀，仍舊使用 Session 作為儲存和擷取使用者權限的方法，不需重新指向到某一特地位置上進行存取。當使用者存取其它伺服器時，將會由跨網域機制透過 RBAC 授權中心，取得以 XML 為格式的使用者權限授權表，重新為使用者與伺服器執行工作階段，建置完整的使用者權限授權關係；由鑑於此，伺服器之間的網頁應用程式，不再需要由同一種開發語言撰寫，無論是何種語言的網頁應用程式，僅需將 XML 格式的授權內容轉換於 Session 內，即可使網頁應用程式進行「判讀使用者權限」之能力。

本研究並未設計安全性議題，首先是由 RBAC 授權中心所產生的 Session List，它僅存在使用者與網頁應用程式間的執行工作階段內，排除伺服器與使用者系統已被植入後門，Session List 是無法用其它方式讀出，即使讀出也無法由客戶端發動將可用的 Session List 轉植入於執行工作階段中，其次是 RBAC 授權中心頒予的 Session Client 是由 128 位元的雜湊碼任意組成，並於閒置後 30 分鐘後自動失效，因此要由此來猜測可用的 Session Client 並不容易，即使得知有效的 Session Client 也僅能讀出完整的 Session List，並不能由客戶端發動轉植入使用者與網頁應用程式之間的執行工作階段中。由於網頁應用程

式自身的諸多安全性限制，使本研究方法無須刻意針對安全性問題加以解決，若要加強本研究的安全性依然有因應對策。首先本研究的資訊都是經由標準 HTTP 通道進行，可由伺服器調整改為 SSL 安全通道傳輸，所有的服務、文本、參數、包含 Session XML 都能經由 SSL 安全通道傳輸。至於 Session XML 的資訊傳遞，僅在於伺服器與伺服器之間，並未經過客戶端，因此 Session XML 的傳遞可由伺服器限定僅允許核定的 IP 或域名進行溝通。若擔憂有效的 Session Client 被獲取，且是在核定允許的 IP 內，並破解 SSL 傳輸安全通道取得了 Session XML，那麼還可以針對 XML 文件內的文本進行額外加密，進一步的提高安全層級。

本研究所提出的機制僅解決透過 RBAC 授權機制，可以對所有的物件(網頁)進行詳細的角色配置，並跨越網域將完整的授權資訊傳達給所有的網頁應用程式；但並未對傳達於網頁應用程式之間的授權資訊進行最佳化。假設有五個分佈於不同網域上的網頁應用程式，每一個網頁應用程式恰好有 40 個註冊會員預設可用物件(網頁)，藉由 Session XML 發佈到每一個網頁應用程式產生使用者與網頁應用程式執行工作階段中的 Session List 將會有 200 筆，但對於單一網頁應用程式有效資料僅有 40 筆，其於 160 筆是其它網頁應用程式所需要的。未來針對本研究的持續進行，可對於 Session XML 進行最佳化，將對

應於正確的網頁應用程式傳遞所需的資訊即可，避免傳遞過於完整的 Session XML，使網頁應用程式在建置 Session List 時造成額外的伺服器負擔。

參考文獻

1. 方俊富，「以 LDAP 提昇協同合作平台的安全控管措施之案例研究」，台灣科技大學自動化及控制系碩士論文，2004 年。
2. 王香齡，「以 XACML 標準為基礎之 RBAC 系統建置研究」，清華大學工業工程與工程管理學系碩士論文，2004 年。
3. 朱明中，「Web Application 首部曲」，http://www.microsoft.com/taiwan/msdn/columns/jhu_ming_jhong/A-ASP.NET_Architecture.htm
4. 余俊賢，「以 RBAC 為基礎建構網頁存取控管機制」，交通大學資訊管理學研究所碩士論文，2002 年。
5. 李長庚，「一個開放的 Web-Based Single Sign-On 服務架構」，交通大學資訊管理學研究所碩士論文，2002 年。
6. 李俊傑，「整合角色和工作為基礎的動態存取控制架構研究」，南華大學資訊管理學系暨研究所碩士論文，2002 年。
7. 汪志君，「以物件導向方式設計 RBAC 系統之研究 - 以保險業為例」，交通大學資訊管理學研究所碩士論文，2000 年。
8. 林千代，「可攜性 RBAC 資訊系統架構之研究」，朝陽科技大學資訊管理學系碩士論文，2003 年。
9. 林忠賢，「以角色為基礎之數位權力管理模式之研究」，世新大學資訊管理學系碩士論文，2003 年。

10. 邵曉薇，「配合 RBAC 的稽核機制之設計 - 以製造業的採購流程為例」，交通大學資訊管理學研究所碩士論文，2000 年。
11. 張思源，「Web Services 之概論」，資訊與電腦 2003 年四月號，台北，2003 年。
12. 張益嘉、林金龍、何建明，「數位典藏系統之權限控管」，中央研究院資訊科學研究所，2003 年。
13. 張清文，「利用 XML Security 與 RBAC 設計企業文件控管系統」，中央大學資訊管理學系碩士論文，2003 年。
14. 許育嘉、林世明、蘇偉仁，「以服務導向架構為基礎建構多維度分析資料權限控管服務」，中央研究院資訊科學研究所，2003 年。
15. 郭貞伶，「網際網路單一簽入系統結合 RBAC 授權機制之研究」世新大學資訊管理學系碩士論文，2006 年。
16. 郭素馨，「應用智慧卡實現 RBAC 執行權管制」，交通大學資訊管理學研究所碩士論文，2000 年。
17. 陳星吏，「架構一個以角色為金鑰管理基礎的企業數位版權管理系統雛型」，交通大學資訊管理學研究所碩士論文，2003 年。
18. 廖英彥，「網際網路單一簽入系統應用」，世新大學資訊管理學系碩士論文，2005 年。
19. 劉義漢，「以 RBAC 架構設計 XML-Based 電子金融服務入口之存

取權控管」，交通大學資訊管理學研究所碩士論文，2002 年。

20. 蘇彩妤，「以本體論為基之虛擬企業知識存取控制研究」，成功大學製造工程研究所碩士論文，2006 年。
21. Chandramouli R., Ferraiolo D., Gavrila S., Kuhn R., Sandhu R., "Proposed NIST standard for role-based access control", ACM Transactions on Information and Systems Security, Vol. 4, No. 3, pp. 224-274, August 2001.
22. Dave Happell, Java Web Services, O'Reilly, USA, 2002.
23. Denning D. J., Graham G. S., "Protection-Principles and Practice", Proc. AFIPS, Vol. 40, pp. 40-48, 1972.
24. Ferraiolo D., Kuhn R., "Role-Based Access Control", Proceedings of the 15th NIST-NCSC National Computer Security Conference, pp. 554-563, October 1992.
25. Ferraiolo D., Kuhn R., Sandhu R. S., "The NIST Model for Role-Based Access Control: Towards a Unified Standard", In Proceedings of the 5th ACM Workshop on Role-Based Access Control, pp. 47-63, 2000.
26. G. Edward Amoroso, "Fundamentals of computer security technlolgy", Prentice-Hall International, Inc., 1994.
27. Handy, "Trust and Virtual Organization: How Do You Manage People Whom You DO Not See", Harvard Business Review, Vol. 73, No. 2, pp. 40-50, 1995.
28. Kristol D., "Http Cookie: Standards, Privacy, and Politics", ACM Transactions on Internet Technology, Vol. 1, No. 2, pp. 151-198,

2001.

29. Memcached, http://www.danga.com/memcached
30. MSDN，「工作階段狀態概觀」，

    http://msdn2.microsoft.com/zh-tw/library/ms178581(VS.80).aspx
31. PERSISTENT CLIENT STATE HTTP COOKIES,

    http://wp.netscape.com/newsref/std/cookie_spec.html
32. Sandhu R. S., "Lattice-Based Access Control Models", IEEE Computer, Vol. 26, pp. 9-19, 1993.
33. Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E., "Role-Base Access Control Models", IEEE Computer, pp. 38-47, February 1996.
34. Sandhu R. S., Samarati P., "Access Control: Principles and Practice", IEEE Communications Magazine, pp. 40-48, 1994.
35. W3C, http://www.w3c.org