應用 Web 框架工具建立通用資料庫系統 RDAA：以政治大學社會科學研究資料庫之整合為例

吳信輝、林祐德、廖峻鋒、李蔡彥
國立政治大學電子計算機中心
{elviscat, yutelin, try, li}@nccu.edu.tw

摘要

在軟體工程領域之中，以架構及元件導向的方式開發軟體是未來發展的趨勢。軟體開發所面臨的兩大問題為「變化」與「複雜度」，軟體開發者利用重用性(reuse)處理「變化」，利用元件化(component)處理「複雜度」。針對相同領域的應用系統，如何透過標準化的開發過程，加速軟體開發時程，是近年來非常熱門的研究課題。本研究以政治大學社會科學研究資料庫群為對象，利用軟體架構觀點來分析這些系統之架構與特性，並利用 Java Web 端之相關框架工具：Spring、JavaServer Faces、Hibernate，重新設計其系統架構。經過實作驗證後，我們發現此作法可以有效縮短了開發新的研究資料庫系統所需的時程。

關鍵詞：軟體開發、框架、Spring、JavaServer Faces、Hibernate

Abstract

In the field of software engineering, *framework* and *component* are the trend of software development, where *variant* and *complexity* are two key issues. Software engineer use the concept of "reuse" to deal with variant and "component" to deal with complexity. For some software applications in the same domain, it is a popular research topic on how to reduce the development time by using standard software development process. This paper takes the application of research databases for social sciences in National Chengchi University as an example to analyze the application from the point of view of software architecture. We adopt the Java Web framework: *Spring*, *JavaServer Faces* and *Hibernate* to redesign the general-purpose database application. Our experiments show that this methodology is adequate for our application and does speed up the process of building a new database application.

Keywords: Software Development, Software Framework, Spring, JavaServer Faces, Hibernate

一、前言

隨著網際網路的發展，以網路為載具的各式資料庫系統，如雨後春筍般的開發成長，成了知識經濟時代的重要資產。以政治大學為例，社會科學研究單位之研究成果，多以網頁資料庫建置為其成果目標。這些資料庫包括中國大陸研究中心之中國大陸政治菁英資料庫、中國大陸財經資料庫、中國大陸台商研究資料庫、中國大陸對外關係及兩岸衝突解決資料庫[1]；第三部門研究中心之大陸第三部門學術資料庫[2]；選舉研究中心之中國大陸選舉與基層治理資料庫[3]；以及傳播學院之中國傳播資料庫[4]。回顧以上之資料庫特性，皆具有相當類似之功能需求，即一般之瀏覽、查詢以及管理者資料管理等功能。但是在實際開發過程中，由於資料內容的差異性，組織層面上的開發人力來源多元化，程式設計人員仍舊必須利用人工重新產生相類似的原始碼。在查詢相關之資料庫時，也必須經過不同的資料查詢介面，造成相互資料庫間之整合問題。以上這些現象，不僅延誤資料庫開發時程，也有著產出效率低落及維護不易的問題。

另外，由於這一類的社會科學研究資料庫，最終成果也必須對外展示，因此資料庫多以網頁方式呈現。回顧網頁資料庫之開發歷史，從傳統 cgi 程式至伺服器端的網頁腳本語言(Server-side scripting language)，例如：ASP、JSP 與 PHP 等。我們可以發現其發展趨勢已經漸漸從主從式架構轉變為多層式架構。在多層式架構中，功能導向的設計與表現層與邏輯層之區隔，已經漸漸成為開發之主流。各類相關程式語言，也都漸漸支援相關概念之發展；例如 Java 所提出的 J2EE 架構，其中之 MVC 架構方法，便是以多層式架構為核心，而此類技術的發展正是框架理論的應用[16]。

在軟體工程領域之中，以架構及元件導向的

[1] http://ics.nccu.edu.tw/chinaleaders
[2] http://333.nccu.edu.tw/db/China/
[3] http:// ssrc.sinica.edu.tw/
[4] http://commdb.nccu.edu.tw/

方式開發軟體是未來發展的趨勢[8]。軟體開發所面臨的兩大問題為「變化(variant)」與「複雜度(complexity)」，軟體開發者利用重用性(reuse)處理「變化」，利用元件化(component)處理「複雜度」。針對相同領域的應用系統，如何透過標準化的開發過程，加速軟體開發時程，是近年來熱門的研究課題[2]。在本研究中，我們提出一個可客製化(customizable)的應用程式架構 RDAA(Research Database Application Architecture)；利用此應用程式架構，開發者可以在標準的開發流程上快速的建立與管理不同資料特性的資料庫系統。本研究利用軟體架構的概念，以及 Java Web 端之框架工具：Spring、JavaServer Faces(JSF)、Hibernate，將表現層與邏輯層分離，使得不同特性之資料庫系統，可以利用相同的軟體架構產生，並且經由元件化的設計，加強程式之維護性。本研究整合相關的框架工具，以快速建立可用的資料庫系統，大幅縮短開發流程，提高整合度，並降低未來維護困難度。

二、文獻回顧

我們針對軟體架構、框架以及 Java Web 端的相關工具之內容與應用進行評析。

(一) 軟體架構(software architecture)

軟體架構是一個較高層次的軟體設計方法，以元件的角度來描述整個應用系統的架構及其相互關連性 [8]。由上述軟體架構的定義，我們發現軟體架構的基本元素為元件，元件依據應用系統需求個別建立，軟體架構的目標則在重複利用。因此其運作過程在於將元件套用在客製化的架構裝中，當此架構設計完成後，即可在相同領域的其他應用系統中重複使用。應用系統內容與需求如有更改，元件可以重新設計，並只需針對元件做更換或是依據需求重新組合元件即可。

軟體架構的概念在提供一個結構化與系統化的方法來增加軟體開發時的可用性(reuse)以及降低軟體開發複雜度(complexity)。一個有效率的軟體架構開發流程，需要一個實用的軟體元件塑模方法以及具有軟體架構觀念的工程師才能達成[14]。但由於其結構化與系統化方法的概念與傳統軟體開發方法不同，造成學習曲線較高。一個具有軟體架構觀念的工程師不易在短時間內養成，而以軟體架構概念為基礎的軟體開發公司仍佔少數。

(二) 框架理論

框架是一種軟體重用(resue)的技術，由一組抽象類別及這群抽象類別的相對應動作組成。框架本身是一個軟體半成品，軟體設計人員將可重用的程式製成框架，再由程式開發人員將這個軟體半成品客製化成符合該專案需求之應用程式[4][13]。框架、一般系統軟體之函式庫(Library)與軟體樣式(Software Patterns)[6]皆屬於軟體重用的技術。

框架設計的目標是辨認領域實體(Entity)，將其建立為領域模型(Domain Model)，讓領域邏輯可以被重覆使用。框架設計的第一個程序是由軟體設計人員將目標應用系統，切分成若干個抽象類別，並定義各個類別的責任(responsibility)與其相互間的合作關係，作為新系統之架構設計。開發者可繼承這些類別並建立類別之間的關連來建構新系統[3]。

建構框架方法論有很多種，依據專案特色將其區分為由舊系統重構(Refactoring Approach)及重新建構(Priori Approach)二類[4]。由舊系統重構適用於既存的系統，由開發人員將可重用的部份抽象化再建立框架。以社會科學研究資料庫系統為例，建構框架前，即存在許多正在使用中的系統，故較適合由舊系統重構方式。若想將原本不存在的軟體建立框架，則較適合重新建構的方法。依一般開發流程，由使用案例(Use-case)開始，再將使用案例抽象化(Use-case assortment)，找出系統變異點(hot-spot)，這些變異點便成為建置抽象類別的候選對象[4,15]。框架技術將程式分割為具有關連性、階層性的物件，利用其互動關係來架構整個系統，對於未來在系統的架構與維護方面，皆具有相當程度的幫助。

(三) 展示層相關框架工具

Web 應用程式與傳統 Client-Server 應用程式最大的不同在於 HTTP 協定本身無法保留客端狀態(stateless)，因些 Web 開發人員必須自行手動撰寫保留客端狀態的程式碼，增加了不少負擔；再加上所有客端傳送的資訊都必須由開發人員自行處理並轉換成領域模型(Domain Model)，因此造成 Web 應用程式的開發十分煩瑣且不易偵錯。

有許多研究人員應用 2.1 所提到的框架理論來解決這些問題，例如 Craig R.McClanahan 在 2001 年所提出的 Struts 應用程式框架[18]，Struts 是一個 MVC 框架，採用 Servlet，JSP 和 custom tag library，是開發 MVC 系統的底層技術。Struts 對 Model、View 和 Controller 都提供了對應的實作元件，Controller：控制器的作用是從客端接收 request，並且選擇執行對應的商業邏輯，然後把結果送回到用戶端。在 Struts 中 Controller 功能由 ActionServlet 和 ActionMapping 物件構成，由 ActionServlet 接收 request 並經由其子類別 ActionMapping 產生動態的相對應映射。

Model 功能，MVC 中的 Model 部分從概念上可以分為兩類：系統的內部狀態，和改變系統狀態的動作。在改變系統狀態的動作方面，Struts 為 Model 部分提供了 Action 和 ActionForm 物件處理改變系統狀態的動作：Action 處理器物件將處理邏輯封裝後另行處理，並且對應到合適的 View 元件。ActionForm 物件，透過屬性定義的方式來描述客端 Form 欄位資料，利用 ActionForm 物件和 Struts 提供的自定義標籤庫，可以簡化對客端的表單資料的設計作業，也不再需要和 request、response 物件進行資料交換。在內部狀態處理上，Struts 建議使用 JavaBean 表示系統的內部狀態，如果系統複雜度較高，也可以使用像 Entity EJB 和 Session EJB 等元件來實作系統狀態。在呈現部分 View，Struts 則是透過 JSP 技術，提供自定義的標籤庫，與 ActionForm 的建立相對應的映射。

Tapstry[10]是一個元件型態的框架，使用獨立的 IEngine 提供 IEngineService，此 engine 代理 Servlet 的 request，並負責 Render 頁面，其狀態變化資訊皆先暫存於 Pool 中，只有在 render 時才把 Instantiation Object 從 Pool 中取出與設定好的 Properties 結合，透過繪製器(render)送至客端。但由於其由元件組成的特性，使得學習曲線較其他框架為高，一般程式開發者不易在短時間中瞭解。

Spring Framework 是一個以「觀點導向程式設計(AOP, Aspect-Oriented Programming)」與「控制權倒置(IOC, Inversion of Control)」為基礎的框架[17]，其詳細特性我們將於第五節中詳述。

JSF(JavaServer Faces)則是一個建構網路應用程式(web application)的使用者介面(user interfaces)的框架工具，其內容包括兩項。第一項是一組 APIs(應用程式介面, Application Programming Interface)，用以展示使用者介面的元件、使用者介面狀態管理元件、網頁應用程式事件管理元件、輸入資料的驗證管理元件、頁面導航管理元件、國際化語言管理元件以及存取管理元件等。第二項是利用利用 JavaServer Pages (JSP)中的 custom tag library，用來表示 JSF 的介面，使 JSF 可以應用在 JSP 之中。

JSF 的設計具有許多優點。例如，從簡單的輸入欄位(input)到捲軸資料表格(scrollable data table)的元件等，這大部分的元件皆已經定義狀態(state)與行為(behavior)的處理方式，利用這些標準元件應用程式介面(API)，開發者可以自行建立符合自己需求的元件，也可以引用其他已經公布的第三方團體所製作元件函式庫。另外，利用分離式的繪製器模式(rendering model)使得在資料呈現的時候可以採取不同的方式；例如，選擇一個項目的元件我們可以利用選單(menu)或是一組單一選擇按鈕(radio button)呈現。再者，利用事件與監督模式(event and listener model)來管理事件；例如，我們想製作使用者行為記錄的時候，我們可以建立一個啟動元件(activating a component)對應到該按鈕，當使用者按了該按鈕，即可對應到啟動元件，並進行事件的後續處理動作。最後，可以透過轉換及驗證模式來轉換及驗證元件中的資料[11]。

每個框架皆有其使用特性，在考量各類框架之優缺點、社會科學研究資料庫之特性、框架程式對系統效能影響程度與系統複雜度之後，我們決定以 JSF 做為展現層部分框架。

(四) 資料存取層相關框架工具

在資料庫的連結介面上，各大軟體開發廠商紛紛推出相關的的中介連接技術，例如 Java 的 JDBC、Microsoft 的 ADO.NET 等。以上的應用程式介面(API)，多以物件關係映射(O/R Mapping)來達成與資料庫連接的目標；例如，JDO, Apache OJB、SQLMap 與 Hibernate 等。其中，Hibernate 可以將管理業務邏輯(business domain logic)直接實作在 Hibernate 的物件中，而無須另外實作持久層(persistence layer)來容納業務邏輯[9]。透過 Hibernate 的映射文件 xxx.hbm.xml，我們只需要更改其中的設定即可轉換不同資料庫之間的瀏覽等相關功能，加上其可以設計動態的查詢功能，因此在考量系統架構特性、成本、開發時程等相關因素後，我們採取 Hibernate 為資料存取層的框架工具。

(五) 元件容器相關工具

元件容器主要的目的主要是作為展示層與資料存取層之間的橋樑，常見的元件容器相關工具有：EJB、Spring、Pico 與 Avalon 等，元件容器依據應用系統的複雜度而不同的選擇；例如 EJB 就是適合擁有複雜架構與業務邏輯的應用系統所使用。但有些應用系統並不會用到複雜的架構與業務邏輯，因此有輕量級的元件容器工具(Lightweight Container, LWC)的設計，如 Spring、Pico 與 Avalon 等。Spring 除了是一個框架工具外，也是一個元件容器工具[5]。由兩個核心技術組成：「觀點導向程式設計(AOP, Aspect-Oriented Programming)」與「控制權倒置(IOC, Inversion of Control)」。Spring 所構成的體系非常的龐大，其架構圖如圖 1 所示。例如有針對 AOP 所對應的相關模組支援，也有針對資料存取層的相關模組支援。Spring 與其他框架工具最為不同的地方，就是在不同的軟體架構層級裡面，提供了一個彈性的整合機制來符合開發者的的需求；例如在資料存取層上，我們不採用 JDBC，而是利用 Hibernate，Spring 可以利用 Spring DAO 來進行彈性的調整。另外，我們也可以利用 Spring

的 AOP 與 IOC 觀念，實作軟體架架構[17]。

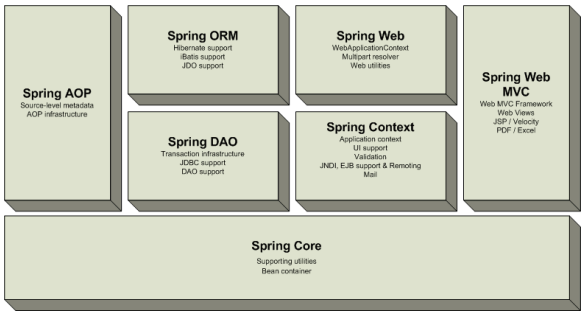

圖 1： Spring 架構圖[17]

三、社會科學研究資料庫的需求分析

根據現有之政治大學社會科學研究資料庫之分析，我們歸納分析相關的問題與需求如下。

(一) 現有之問題

- **分散管理安全性問題：**現有的政治大學社會科學研究資料庫之系統，依據研究計畫之單位分開建置，在系統管理上受限於該單位所開發程式之完整性，在管理與安全性上，仍有改進之空間。
- **各資料庫無法相互整合：**各資料庫所採用的網頁應用系統不盡相同，因此在整合方面（例如進行綜合文獻查詢等功能需求），較難達成理想目標。
- **系統建置流程較長：**由各研究單位委外或是自行開發，缺乏專職系統分析與程式設計團隊，對研究型資料庫之資料特性之瞭解程度不一，開發期程也叫無法明確掌握。
- **新系統開發必須重新進行規劃設計：**各研究單位在設計新資料庫時，皆重新針對資料庫作分析、設計與系統分析等相關流程。

(二) 未來之需求

根據我們以物件導向方法分析目前社會科學研究資料庫之應用特性及系統架構，以及目前各研究單位對於未來的資料庫系統期待，我們歸納社會科學研究資料庫應用系統之未來需求如下：

- **相類似之功能需求：**社會科學研究資料庫應用系統之最主要目標為其研究成果的呈現，因此經過物件導向分析後，可以將所需要之功能列出，例如一般使用者之瀏覽頁面、查詢功能、資料排序等功能，以及管理者之資料編修功能。
- **單一登入機制：**利用單一登入機制整合管理功能，讓在同一應用系統上管理不同之資料庫之管理者可以簡化權限管理的複雜度。

四、RDAA 研究資料庫應用架構

依據上述之問題與需求分析，我們歸納 RDAA 系統架構如圖 2 所示。在圖 2 中，我們分別利用 JSF、Spring 以及 Hibernate 來處理系統架構中的 Presentation Tier、Bussiness Logic Tier 以及 Integration Tier。採用此類架構後，我們只需要修改相關的資料庫映射(mapping)與組態設定檔，即可馬上映射(mapping)至不同類型之資料庫中。以此系統架構為基礎，便可以達成我們建立適用於不同資料庫的資料庫管理系統之目標。

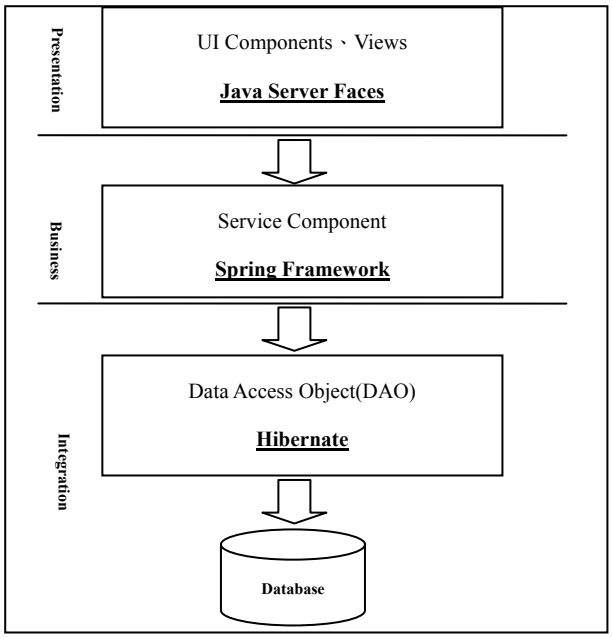

圖 2： RDAA 系統架構圖

依據此系統架構我們分別開發了五個功能模組，如表 1 所示。

表 1： RDBF 功能模組簡介

功能模組名稱	功能簡介
資料呈現模組	呈現資料列表、排序、分頁以及全文查詢功能。
全文檢索模組	以全文檢索方式查詢資料庫中之資料。
管理者上傳模組	利用 CSV 檔上傳批次之資料檔。
權限管控模組	依據角色原則(Role)建立管理者、資料維護者等不同的角色權限。
單一登入模組	利用 SSO server 建立單一登入功能。

(一) 資料呈現模組

我們以開發 Service Beans 的方式來處理資料庫呈現的問題，並設計分頁之功能來處理 JSF 中所存在的分頁問題。我們利用程式將資料筆數處理成頁數之後，將單一之資料利用符號轉換成可以展示分頁的功能，其運作流程如圖 3 所示。

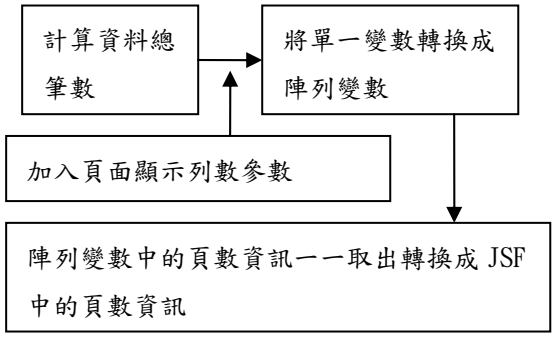

圖 3： 分頁功能程式運作流程圖

(二) 全文檢索模組

本研究利用 Java 的搜尋引擎 Lucene[5]建構了全文檢索機制，Lucene 是以 Java 為基礎的搜尋引擎，其特色包含提供高效能的索引機制、提供布林運算查詢、提供查詢結果的排名（ranking）與評分、跨平台、開放易用的 API、極易擴充及客製等[12]。此外，Lucene 利用 Java 的特性，並擁有以下的優點。

- 支援遞增式的索引：毋需每次重建索引，即使只增加一點資料。
- 不侷限在特定型態的資料來源：例如 HTML，可自行剖析不同的資料來源，轉化成可供 Lucene 處理的文件類別。
- 文件的多欄索引控制：可將文件劃分為多個欄位，每個欄位都可進行不同的索引控制，例如斷詞與否。
- 提供通用的文件分析能力：可自訂文件分析的方式，標準支援 CJK 與阿拉伯語言等非拉丁語系的語言。
- 提供通用的查詢分析能力：可自訂查詢的語法。[1]

(三) 管理者上傳檔案模組

本研究利用 JSF 自訂標籤(tag)的方式建立上傳檔案模組；JSF 自訂標籤(tag)之設定語法如圖 4 所示。我們將自訂標籤對應到相對的上傳檔案處理程式 Doucument_Validation 中。另外，我們再利用 filter 程式檢查上傳檔案之正確性。filter 程式主要是針對資料庫的空值、是否符合格式等驗證項目作檢查，其特性是利用映對的方式去對應該資料庫的設定而產生動態的 filter 程式內容，程式流程如圖 5 所示。

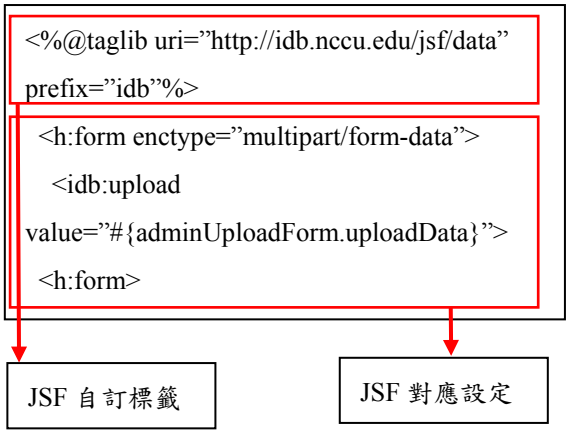

圖 4： JSF 自訂標籤建立上傳功能模組程式節錄

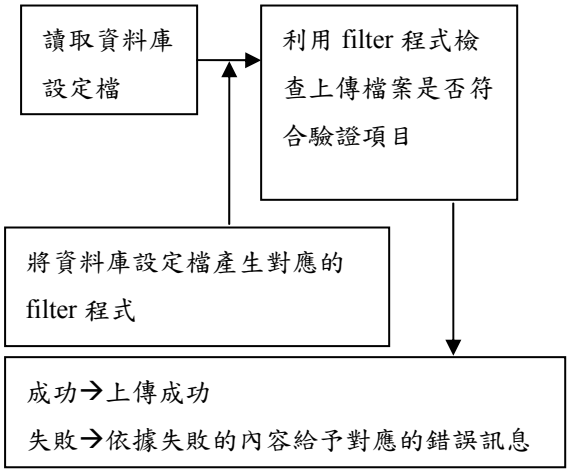

圖 5： 上傳檢查功能程式流程

(四) 權限控管模組

本研究利用角色管理原則 (Role-based Access Control) 來建立權限管控模組。使用者主要以角色來控管其使用權限，而不是以使用者名稱為依據來建立權限控管機制。在角色為依據的權限控管機制中，管理者管理不同角色所需的權限及每個使用者所擁有角色。當某使用者被刪除時，管理者無須額外處理權限變換相關事宜，可減少管理者管理成本。本研究利用 Acegi+CAS 軟體，建立權限控管模組[7]，其中密碼的提取的方法是利用 spring 的 singleton config 方法至 user 資料庫中找尋相關資料，再利用 mapping 的機制來檢查名稱與角色，相關程式流程如圖 6 所示。

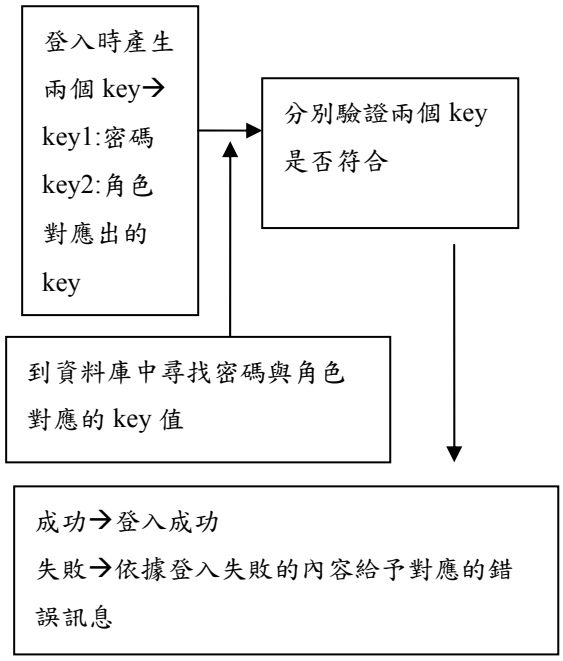

圖 6： 權限管控模組程式流程

(五) 單一登入模組

本研究利用 Acegi+CAS 軟體建立單一登入機制[7]，其 Spring 設定內容如圖 7 所示。我們利用 Open-source Single Sign-On with CAS 軟體建立 SSO Server，此 server 可以自動驗證使用者，而驗證過的 username 會存在 session 中。當同一個使用者登入不同的機器時，皆會至此台 SSO server 中尋找是否驗證過；如有驗證過則無須檢查，如未驗證過則出現登入畫面重新登入，程式流程如圖 8 所示。

```
<bean id="keyClassMapping"
class="edu.nccu.idb.auth.KeyClassMapping
" abstract="false" singleton="true"
lazy-init="default" autowire="default"
dependency-check="default">
  <property name="map">
    <props>
      <prop
key="ROLE_FIRSTTESTAP_ADMIN">
edu.nccu.idb.action.LoginAction</prop>
    </props>
  </property>
</bean>
```

Spring 標籤連結至 SSO 單一登入模組

圖 7： 單一登入中的 Spring 的設定方式

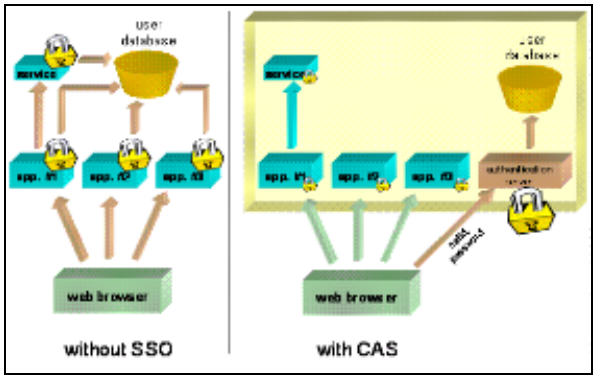

圖 8： 單一登入流程[7]

(六) 組態設定

我們利用組態設定的方式來描述 Web 應用程式的內容。利用組態設定檔的好處在於將 data 與 logic 分開，讓程式的可維護性增高；此外我們也可以利用這一個特性，將不同的資料庫映對描述，動態產生於組態設定檔中，簡化系統開發的流程與縮短開發時程。相關組態設定在 faces-config.xml、hibernate-config.xml、及 spring-config.xml 三個檔案中。以 JSF 為例，相關設定內容如圖 9 所示。利用 managed-bean 標籤建立相關映射的 java bean 檔，並且設定相對應的程式設定位置等。

```
<faces-config
xmlns="http://java.sun.com/JSF/Configuration">
<application>
  <variable-resolver>
de.mindmatters.faces.spring.FacesSpringVariable
Resolver
  </variable-resolver>
</application>
<managed-bean>
  <managed-bean-name>應用資料庫名稱
  </managed-bean-name>
  <managed-bean-class>應用程式 package 與
class 位置
  </managed-bean-class>
  <managed-bean-scope> session(程式型態)
  </managed-bean-scope>
</managed-bean>
```

圖 9： faces-config.xml 之相關設定方式

在連結資料庫方面，有關於資料庫的連結位置的設定在 db-config.xml 檔中；相對應的資料庫對應欄位，則是定義在 db.xml 中。db.xml 之設定內容如圖 10 所示，利用組態設定檔中的資料庫映射(mapping)設定，即可對應至不同之社會科學研究資料庫中。

```
<?xml version="1.0" encoding="UTF-8" ?>
<database>
  <table name="article" type="atomic">
  <column name="NUMBER"(資料庫對應名稱)
type="varchar"(資料庫欄位型態)
length="10"(資料庫欄位長度) key="true"(資料
庫主鍵設定) label="編號"(資料庫呈現名稱)
showLabel="true"(資料庫呈現名稱設定) list-
Label="依編號排序"(資料庫欄位排序名稱)
showList="true"(資料庫欄位排序設定) arti-
cleLink="true"(資料庫連結設定) />
…((第三欄資料欄位設定)
…(第三欄資料欄位設定)以此類推
  </table>
</database>
```

圖 10：db.xml 相關設定圖

在網頁的呈現上，我們利用 include 的方式來建立呈現頁面。我們將整個呈現頁面以 header.jsp、main.jsp 以及 footer.jsp 三個區塊組成整體之頁面。在 header.jsp 與 footer.jsp 中，可以依據不同之社會科學研究資料庫，建立具有本身特色之標題與 logo 圖檔。另外，在 main.jsp 中，我們利用 JSF 中之 CSS 相關設定功能，進行網頁呈現風格之設定。

(七) 系統架構小結

依據上述社會科學研究資料庫管理系統之功能模組之介紹，我們可以總結此社會科學研究資料庫管理系統之幾點特性如下：

- **跨平台安裝：**利用 Java 跨平台之特性，可安裝在不同的 Application Server 上。如需針對特定 AP Server 及 Database Server，則只需更改相關之設定並利用 Ant 工具在安裝前重新組態。
- **單一登入：**在同一台 AP Server 可利用 Ant 客製化不同社會科學研究資料庫管理系統，但不同的資料庫管理系統可共用一個帳號。
- **模組可以重複利用：**利用組態檔案方式映射不同之資料庫內容，使資料庫管理系統能夠依據不同資料庫的需求進行客製化。
- **角色對應之權限管理機制：**角色的主要以總系統管理者、資料庫管理系統管理者、資料庫管理系統資料維護者以及一般使用者四類為主，各個角色的系統管理權限如下：
  1. **總系統管理者：**管理所有之使用者帳號資料、資料庫管理系統及資料庫描述、及 SSO server。
  2. **資料庫管理系統管理者：**管理資料庫管理系統內容客製化設定、資料庫管理系統介面之客製化設定、及一般使用者帳號管理。
  3. **資料庫管理系統資料維護者：**建置及維護資料。
  4. **一般使用者：**使用者登入及資料查詢。

五、系統實作成果

依據第四節所建立之系統架構，本研究實作了社會科學研究資料庫之管理系統。系統呈現主要區分為兩類，分別針對系統管理者與一般使用者。以下分別簡述這兩部分的功能及實作成果。

(一) 系統管理者

我們實做出的系統包含三種系統管理者：總系統管理者、資料庫管理系統管理者及資料庫管理系統資料維護者。總系統管理者可利用 Ant 建立社會科學研究資料庫所需之環境，包括 SSO Application Package、以及客製化之社會科學研究資料庫管理系統。另外，總系統管理者也管理所有使用者之使用權限。資料庫管理系統管理者則利用組態設定介面檔設定資料庫樣式、欄位名稱以及欄位限制。在表現層部分，可以設定複合查詢欄位、每頁資料數目、及 CSS 變更等功能。另外，此管理者可利用 Ant 建構客製化社會科學研究資料庫管理系統，並可設定該社會科學研究資料庫管理系統使用者之使用權限。最後，資料庫管理系統資料維護者可以單筆新增資料、搜尋資料並可更新、下載批次上傳格式、批次上傳資料、及在資料上傳後系統自動建立全文檢索索引。管理者使用介面如圖 11 所示。

圖 11： 管理者使用介面

(二) 一般使用者

使用者可自行決定登入與否，並在登入後點選列表選項或進階查詢選項。使用者可以從列表選項直接列出瀏覽結果，或從進階查詢選項列出查詢選項及欄位以供輸入。如使用者角色錯誤則自動轉

圖 12： 使用者查詢畫面

向 SSO Server 處理。使用者查詢畫面如圖 12 所示。

六、結論

本研究利用軟體架構與框架的概念，以及 Java Web 應用程式之框架工具 Spring、JSF、Hibernate 實作可以快速應用於不同資料庫之資料庫管理系統架構。本研究將表現層與邏輯層分離，使得不同特性之資料庫可以在相同之軟體架構上呈現，並且經由模組化的設計，加強程式之維護性。經實作驗證後，確實符合相關之需求。基於此軟體架構，本研究利用 Spring、JSF、Hibernate 等框架工具建立的 RDAA，未來可依據不同的框架工具重新組合架構。

在以軟體架構為基礎的軟體開發過程中，本研究成果著重在軟體架構層。對上層的概念層而言，我們可以利用本研究的成果，探討與抽離出更上層之架構概念，以符合更大之「變化」需求。對下層的應用層而言，在未來增加新功能時，以本研究的元件化設計為基礎，可探討元件的管理機制，以符合「複雜性」的需求。在資料庫查詢的整合方面，我們建議未來可以透過 Open URL 的技術，來達到整合查詢之目標。

六、參考文獻

[1] 王建興, “Java 的開放原碼全文搜尋技術-Lucene”, Java two 2004, available at URL<http://www.javaworld.com.tw/javatwo2004PDF/王建興/b16.pdf>

[2] 吳仁和、曾光輝, “軟體元件塑模方法之研究,” 第八屆『國際資訊管理研究暨實務研討會』論文集, 2002.

[3] 陳泓志, “物件導向企業框架之研究,” 國立政治大學資訊管理學系, 碩士論文, 1990.

[4] 廖峻鋒、李蔡彥, “SAWA:支援單一登入及 MVC 樣式的校務行政系統應用程式框架,” TANET2003 研討會論文集, 2003。

[5] B. Tate, “Secrets of lightweight development success, Part 2: How to lighten up your containers”, available at URL<http://www-128.ibm.com/developerworks/opensource/library/os-lightweight2/>

[6] D. Govoni., *Java Application Frameworks*, Mississauga: John Wiley & Sons, 1999.

[7] A. Petro, “Examining the Central Authentication Service,” available at URL<http://zoo.cs.yale.edu/classes/cs490/03-04b/andrew.petro/tex/caspaper.pdf>

[8] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*, Boston: Addison-Wesley Professional, 2004.

[9] Hibernate home, URL<http://www.hibernate.org/>

[10] Jakarta Tapestry, URL<http://jakarta.apache.org/tapestry/>

[11] JavaServer Faces home URL<http://incubator.apache.org/beehive/pageflow/jsf.html >

[12] Lucene, URL<http://lucene.apache.org/>

[13] M. Fowler, *Patterns of Enterprise Application Architecture*, Boston: Addison-Wesley Professional, 2002.

[14] M. Shaw, D. Garlan, *Software Architecture: Perspectives on an emerging discipline*, Upper Saddle River : Prentice Hall, 1996.

[15] P. Wolfgang, *Hot-Spot-Driven Development”, Building Application Frameworks - Object-Oriented Foundations of Framework Design*, Wiley Computer Publishing, Chap. 16, pp.379-393, 1999.

[16] S. Burbeck, “Applications Programming in Smalltalk-80(TM): How to use (MVC) Model-View-Controller”, available at URL<http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>

[17] Spring Framework home URL<http://www.springframework.org/>

[18] Struts, URL<http://struts.apache.org/>