私立東海大學資訊科學研究所

碩士論文

指導教授：周忠信

可應用在網際網路/企業內網路並以 XML 技術為基礎的瘦身型計算模式

Thin-Client Computing for Developing Internet/Intranet Applications Based on XML Technology

研究生：盧惠傑

中 華 民 國　八十九　年　六　月　十四　日

摘要

在本篇論文中，我們首先提出一個新的 thin-client-MVC 設計樣式。根據此設計樣式，我們進而利用 XML 技術與 Java2 EE 標準設計出一個新的瘦身型主從式計算架構。本篇論文所提出的瘦身型主從式計算模式，基本上是揉合傳統主從式計算模式、利用瀏覽器作為瘦身型客戶端之主從式計算模式、以及利用遠端顯示技術所發展的瘦身型客戶端之主從式計算模式於一體的新型態作法。其關鍵則在於利用 XML 來描述客戶端圖形使用者介面與伺服端對應之企業邏輯，客戶端基本上不再受各種不同應用需求而增大程式，但此架構下之客戶端又遠比傳統瀏覽器具有極大的人機介面彈性與效果，同時又能分擔原來遠端顯示技術必須透過伺服端執行的所有計算工作。因此本架構乃具備上述各模式的優點，卻又不受其缺點之影響。在實做設計上，為了讓整個軟體架構能夠支援狀態記憶模式的通訊效果，以及可以應用於企業內/外網路與網際網路或無線網路環境上，本論文另外提出一個專屬的通訊協定 TCTP(Thin-Client Transport Protocol)。此通訊協定不僅支援 TCP 連線作業同時也支援 UDP 的連線方式，以確保在連線品質較差的網路環境中也可以正常運作。本研究利用 XML 與 Java 技術設計而成，具有多項優點。除了具跨平台特性等 Java 原有優勢外，更重要的是由於資料以及應用程式是以 XML 文件形式傳輸，因此在本架構中我們可以進一步透過編、解密技術，保障企業資訊系統應用於網際網路上時，除了資料外，應用程式也無法被窺視的安全性。

關鍵詞： 主從式計算、網際網路、設計樣式、瘦身型客戶端、遠端顯示、XML、Java、TCTP

Abstract

In this thesis, a thin-client MVC design pattern is introduced. Based on the pattern, and the technology of XML and Java2 Enterprise Edition standard, we further propose a framework to support thin-client-server computing such that the framework can be used to develop thin-client applications under the Internet and/or Intranet environment with less effort. The model of thin-client-server computing proposed in this thesis is a trade-off between the traditional fat client-server architecture and the thin-client solution based on the technology of remote displaying. We use XML technology to define and describe the presentation logic of the Graphical User Interface and the mapping of the business logic stored in the server side. Following the proposed robust protocol TCTP (Thin-Client Transport Protocol), the presentation logic and the data are transferred as an XML document to the thin-client. After parsing the document, the user interface is shown on the thin-client. Furthermore, since the data and the application are encoded as an XML document, it is possible to protect both data and application transferred in the Internet by using corresponding encryption/decryption technology.

目次

摘要 ii

Abstract iii

圖目錄 v

表目錄 vii

第一章 緒論 1

1.1 **研究動機及目的** 1

1.2 **章節內容** 4

第二章 各式計算模式的 MVC **設計樣式、** XML **與** J2EE **介紹** 5

2.1 **應用在各式計算模式的** MVC **設計樣式** 5

2.2 **以瀏覽器為基礎的瘦身型客戶端** 7

2.3 **遠端顯示的瘦身型客戶端** 7

2.4 XML (eXtensible Markup Language) 8

2.5 Java 2 Platform Enterprise Edition (J2EE) 10

第三章 Thin-Client MVC **設計樣式** 12

3.1 **目的** 12

3.2 **動機** 12

3.3 **應用時機** 13

3.4 **架構圖** 13

3.5 **組成元件** 14

3.6 **合作關係** 14

3.7 **結果討論** 15

3.8 **範例程式** 16

第四章 以 XML **技術為基礎的瘦身型主從式計算軟體架構** 19

4.1 **軟體架構** 19

4.2 Thin-Client Markup Language (TCML) 21
4.3 Thin-Client Transport Protocol (TCTP) 24
第五章 範例系統實作 28
5.1 **範例劇情簡介** 28
5.2 **實做方法及範例** 29
第六章 結論及未來展望 49
6.1 **結論** 49
6.2 **未來方向** 49
參考文獻 50

圖目錄

圖 2-1 主從式計算模式..... 5
圖 2-2 以遠端顯示為架構的瘦身型計算模式..... 6
圖 2-3 Thin-Java client 瘦身型計算模式..... 6
圖 2-4 遠端顯示技術所發展的瘦身型客戶端技術說明圖..... 8
圖 2-5 XML DTD 範例..... 9
圖 2-6 XML 範例文件..... 10
圖 2-7 XML 樹狀結構範例圖..... 10
圖 2-8 J2EE 軟體架構圖..... 11
圖 3-1 本研究中之瘦身型客戶端技術示意圖..... 12
圖 3-2 Thin-client MVC 設計樣式..... 13
圖 4-1 以 XML 技術為基礎的瘦身型客戶端軟體架構..... 21
圖 4-2 TCML 之 DTD..... 22
圖 4-3 DSML 之 DTD..... 23
圖 4-4 metal.xsl..... 24
圖 4-5 motif.xsl..... 24
圖 4-6 windows.xsl..... 24
圖 5-1 系統流程圖(1)..... 29
圖 5-2 start_template.xml..... 30
圖 5-3 分析樣版文件的流程..... 30
圖 5-4 start.xml..... 31
圖 5-5 啟動人事資料庫管理系統的畫面..... 31
圖 5-6 觸發 logo 事件後的處理流程..... 32
圖 5-7 login_template.xml..... 33
圖 5-8 登入人事資料庫管理系統的畫面..... 34

圖 5-9 系統流程圖(2) 34
圖 5-10 登入系統主畫面前的認證畫面 35
圖 5-11 觸發 okbtn 事件後的處理流程 35
圖 5-12 init_template.xml 36
圖 5-13 人事資料庫管理系統主畫面 37
圖 5-14 系統流程圖(3) 37
圖 5-15 觸發 info 事件後的處理流程 38
圖 5-16 main_template.xml 39
圖 5-17 分析樣版文件及取得完整資料的流程 40
圖 5-18 main.xml 中 datamodel 的內容 40
圖 5-19 檢視人事資料的畫面 41
圖 5-20 新增人事資料的畫面 42
圖 5-21 create.xml 43
圖 5-22 填妥人事資料後的畫面 44
圖 5-23 觸發 okbtn 事件後的處理流程 44
圖 5-24 created_template.xml 45
圖 5-25 created.xml 中 datamodel 的內容 46
圖 5-26 新增人事資料後的畫面 47
圖 5-27 刪除人事資料的畫面 48
圖 5-28 刪除人事資料後的畫面 48

表目錄

表 4-1 TCTP **回應訊息的格式(伺服端至客戶端)** 26

表 4-2 **回應客戶端需求的服務** 26

表 4-3 SendTCML command **的參數** 26

表 4-4 TCTP **請求訊息的格式(客戶端至伺服端)** 27

表 4-5 **請求伺服端服務的** command 27

表 4-6 EventTrigger command **的參數** 27

第一章 緒論

1.1 研究動機及目的

近三十年來，電腦無疑已成為是企業用來提昇工作效率並確保工作品質所不可或缺的主要輔助工具之一。然而儘管它具有快速處理資料與計算的良好能力，但是對一般使用者而言，電腦仍是讓人望之生畏。而造成此種落差的主要關鍵因素之一，則在於電腦缺乏提供使用者舒適及易學易用的存取介面。因此當「圖形使用者介面」（Graphic User Interface, GUI）的出現，電腦不能為一般使用者順利操作的問題總算獲得部分解決。不過新的問題卻隨著圖形界面的出現而產生，因為一個具有圖形使用者介面的應用程式，將會耗費許多系統資源。所幸隨著「個人電腦」的蓬勃發展以及「主從式計算」（client-server computing）模式的出現，上述問題將不再是個困擾 [17, 18]。

一般而言，在主從式架構的應用程式中，「伺服端」（server side）的電腦主要在於負責處理與資料庫存取的相關工作，「客戶端」（client side）的個人電腦則負責處理大部分圖形使用者介面的顯示與控制等邏輯。由於客戶端的程式所佔空間較為龐大，因此我們稱此為「肥胖型的客戶端」（fat-client）[17, 18]。此種主從式計算的方式應用在區域網路（Local Area Network, LAN)環境中，由於頻寬足夠、個人電腦資源完整、同時距離近管理方便，因此較不會構成太大問題。但若將之移植至「網際網路」（Internet）或「無線網路」（wireless network）時，則受限於網路頻寬、客戶端縮小不再是個人電腦、以及廣域地理位置等因素，肥胖型的客戶端之主從式計算將很難適用。

隨著網際網路與無線網路的快速普及，目前許多網路應用系統的客戶端僅具少量記憶能力、低電力支援與簡易介面等效果，此類客戶端包括如 NC、PDA、IA 或 smart phone 等，因此所有服務皆需由網

路另一端強大的伺服器來提供。此種運作模式極為類似網際網路上「全球資訊網」（World Wide Web, WWW or web）的運作概念[4]，與上述介紹的主從式計算模式有所不同，我們稱之為「瘦身型計算」（thin-client computing）模式[3]。瘦身型計算模式主要是將客戶端應用程式降到最低，以便隨時隨地獲得並使用所需服務，以真正達到網路「無所不在」（ubiquitous）的最終目標。瘦身型計算模式一般而言擁有兩項類似「中央集權式計算」（centralized computing）的優點，分別是資料統一完整管理以及客戶端「零管理」（zero administration）。前者指的是所有應用程式相關的資料都會存放在同一環境上，較容易保證資料的一致性；後者的好處則有許多，其中例如應用在軟體升級時，可以不用理會客戶端，只需將伺服端的程式更新則會馬上反映至客戶端的使用上。這對需要大量客戶端使用者的大企業而言，可以因此而獲得相當明顯的效益。

目前有關瘦身型計算的研究很多[3, 19, 22-25, 28]，其中最主要的作法有兩種，它們分別是以全球資訊網之「瀏覽器」（web browser）作為客戶端，或是利用「遠端顯示」（remote displaying）技術另行發展客戶端[3, 22-24, 28]。有關利用網路瀏覽器作法，目前主要是透過網頁或 Java applet 進行，對於一般企業資訊系統（enterprise information system）需求而言頗為不足，因此在此不再贅言。至於遠端顯示技術的概念，主要是在於建立一種特殊的傳輸通訊協定，使其能夠傳送客戶端的輸入（如鍵盤及滑鼠）至伺服端，以及將從伺服端所擷取的工作畫面影像，經壓縮後傳至客戶端。此種作法，客戶端所見皆是由伺服端一一所執行之結果，因此客戶端就如同是一台多媒體化的終端機。這種計算模式的缺點，主要乃在於客戶端無法分擔伺服端的任何工作，因此也無法獲得如主從式計算模式中 down-sizing 的好處。而

更糟糕的是，利用遠端顯示技術所發展的瘦身型計算，當多人同時上線時，其伺服端必須具備有相當的計算能力，否則將無法負荷龐大的系統資源需求。目前此方面技術已有商業化產品，例如像昇陽（Sun）公司的 SunRay [22-24]、微軟（Microsoft）公司的 Window Terminal [28]等。

有鑑於上述主從式計算以及瘦身型計算兩種模式，應用在網際網路或無線網路時各有其缺點，因此在本論文中我們提出了一種新的瘦身型計算模式，此種計算模式類似瀏覽器技術，但仍然可以將客戶端的程式縮小，同時還可以有效地分擔伺服端的系統工作。換言之，此種瘦身型的計算模式揉合了主從式計算模式、以及瀏覽器與遠端顯示之瘦身型計算模式於一身的折衷方案。為了要達到此一目標，我們首先提出一個 thin-client-MVC 的「設計樣式」（design pattern）[9, 10]。根據此設計樣式，我們進而搭配 XML 技術與 Java 2 Platform Enterprise Edition（J2EE）所描繪的軟體架構標準 [20]，設計出新的「瘦身型計算架構」(thin-client-server framework)。在本架構下，企業邏輯（business logic）利用 EJB（Enterprise JavaBeans）實做並存於支援 J2EE 規格的「應用伺服器」（application server）上，應用系統則以 XML 所發展的 TCML（Thin-Client Markup Language）來分別描述圖形使用者介面以及伺服端所對應的 EJB 服務，並傳輸至客戶端。因此原來傳統以瀏覽器作為瘦身型客戶端所欠缺的介面彈性問題乃迎刃而解；利用 XML 所發展的瘦身型客戶端，基本上就是一個可以支援各式企業運算的泛用瀏覽器，因此它可以分擔圖形使用者介面的運作邏輯但又不會增加客戶端的程式大小，等於也具備瘦身型客戶端的優勢。此架構由於搭配 J2EE 作為其伺服端軟體平台，符合目前網路軟體的發展趨勢，因此本研究結果頗為適用於網際網路或無線網路

上的企業資訊應用系統。

本研究利用 XML 與 Java 技術設計而成，具有多項優點。除了具跨平台特性等 Java 原有優勢外，更重要的是由於資料以及應用程式是以 XML 文件形式傳輸，因此在本架構中我們可以進一步透過編、解密（encryption/decryption）技術，保障企業資訊系統應用於網際網路上時，除了資料外，應用程式也無法被窺視的安全性。

1.2 章節內容

本論文第二章將介紹各式計算模式的 MVC 設計樣式、以瀏覽器與遠端顯示作為瘦身型客戶端的方法以及 XML 與 J2EE 技術特性；第三章則介紹我們所提出的 thin-client-MVC 設計樣式；至於有關以 XML 技術與 J2EE 為基礎的瘦身型客戶端軟體架構則於第四章說明之；第五章將介紹本架構的實作範例系統；第六章為本論文之結論與未來發展方向介紹。

第二章 各式計算模式的 MVC 設計樣式、XML 與 J2EE 介紹

在本章中，我們首先利用 MVC 設計樣式來比較說明各類型計算模式的特性與差異。同時有關各種瘦身型客戶端的實做，包括瀏覽器客戶端與遠端顯示客戶端等，也將於本章中介紹之。最後，為了配合說明本研究所用到的技術，我們也將於本章中介紹 XML 與 J2EE 的基本概念。

2.1 應用在各式計算模式的 MVC 設計樣式

為了說明瘦身型與肥胖型客戶端間的特性與差異，我們首先藉由 Model-View-Controller(MVC)這個設計樣式來比較之[10, 11]。MVC 設計樣式的概念，基本上可將一般應用程式分成三類組成元件來設計之，它們分別是 model, view 及 controller。一般而言，一個應用系統的企業邏輯及程式主要邏輯都是實作在 model 元件中，而圖形使用者介面及其事件處理(event-handling)的邏輯則分別實作於 view 及 controller 元件中。如圖 2-1 所示，在傳統的主從式計算模式中，絕大部分的 model 元件以及 view 與 controller 元件都是實作在客戶端的程式中，而此種作法基本上乃會造就主從式計算模式中肥胖型的客戶端。

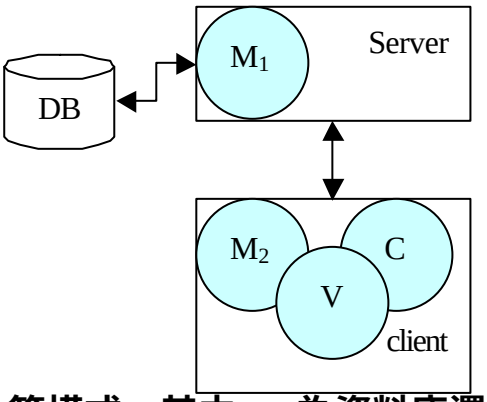

圖 2-1. 主從式計算模式，其中 M_1 為資料庫邏輯(database logic)而 M_2 為程式主邏輯

為了能夠有效地降低肥胖型客戶端的大小，我們可試圖並盡其所能地將 MVC 元件從客戶端移至伺服端。圖 2-2 就是其中一個瘦身型客戶端的極端作法，事實上這也正是遠端顯示的概念。

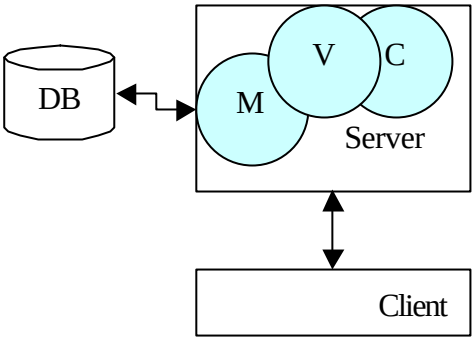

圖 2-2. 以遠端顯示為架構的瘦身型計算模式

除了以上所介紹的模式外，尚有其他兩種瘦身型計算模式的設計，其概念介紹如下。

其中第一種稱為 Thin-Java client，如圖 2-3 所示是將 model 及 controller 完全置於伺服端，而只留 view 在客戶端運作 [19]。

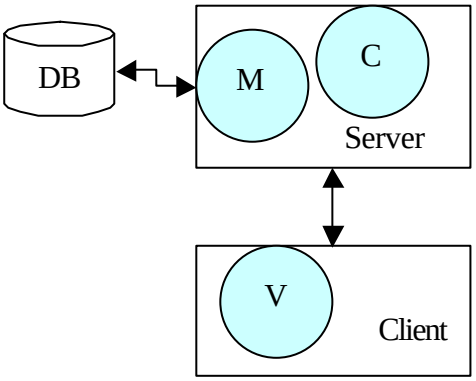

圖 2-3. Thin-Java client 瘦身型計算模式

第二種稱為 Distributed Observable/Observer 設計樣式，強調應用於建立一個即時（real-time）分散式主從架構的應用系統。在其架構中, model 置於伺服端而 view 與 controller 兩元件則留在客戶端 [25]。

當 distributed observable（model）狀態改變時，它會送訊息通知相對應的 distributed observers（view 及 controller）去做相關的處理。為了能夠簡化且實現上述功能，在此設計樣式中還包含另外兩種元件，分別是在伺服端的 Multiplexer 及在客戶端的 Demultiplexer。這兩種元件可以看成是遠端代理者(remote proxy)，其功能是用來降低 distributed observable 及 observer 透過網路通訊時的程式複雜度。雖然此種作法能夠適度降低客戶端的大小，但是在客戶端還需實作 view 及 controller 兩種元件。

2.2 以瀏覽器為基礎的瘦身型客戶端

瘦身型客戶端在網際網路上的應用發展，目前最廣為接受的作法是以「全球資訊網瀏覽器」（world wide web browser）做為基礎的應用程式客戶端。一般而言，此種設計概念又分為網頁及Java applet兩種作法。前者是利用類似HTML form-CGI以及DHTML的作法來做為客戶端的人機介面，其優點是客戶端乃以網頁方式展現，較適用於廣域網路（Wide Area Network, WAN）上，但是致命的缺點卻在於此種技術無法提供豐富的使用者介面，因而無法全面滿足企業資訊系統的應用需求。至於後者利用Java applet作為客戶端，則與前者的情況恰好相反。其優點在於可提供充分且豐富的使用者介面供開發者設計，但是這種作法必須在每次執行時，同時下載所需applet，由於其較接近主從式計算架構，在實際企業應用中所需之客戶端將較為肥大，因此比較適用於區域網路（Local Area Network, LAN）傳輸的環境中。

2.3 遠端顯示的瘦身型客戶端

圖2-4為以遠端顯示技術所發展的瘦身型客戶端技術架構說明

圖。由圖2-2中可知model、view及controller都被放置在遠端顯示之伺服端，所以客戶端在整個應用程式中所佔的比例幾近為零。再進一步由圖2-4窺究其作法可以清楚發現整個程式完全是在遠端顯示之伺服端執行，而客戶端只是負責顯示由伺服端所執行而得之畫面，並將其輸入裝置所接收到的訊號傳回給伺服端。明顯可看出利用此種技術的優點是可以造就出一個極端瘦小的客戶端，但是其缺點是其伺服端的負荷將增加不少，並且客戶端與伺服端間必須維持較高的網路品質，以確保客戶端運作的即時性。若以Sun公司之SunRay為例[22-24]，一台Ultra 10的伺服器最多同時只能服務15個客戶端，並且需配合100Mbps的高速乙太網路環境，因此在整個硬體及網路環境建置上將造成費用大量增加之困擾。

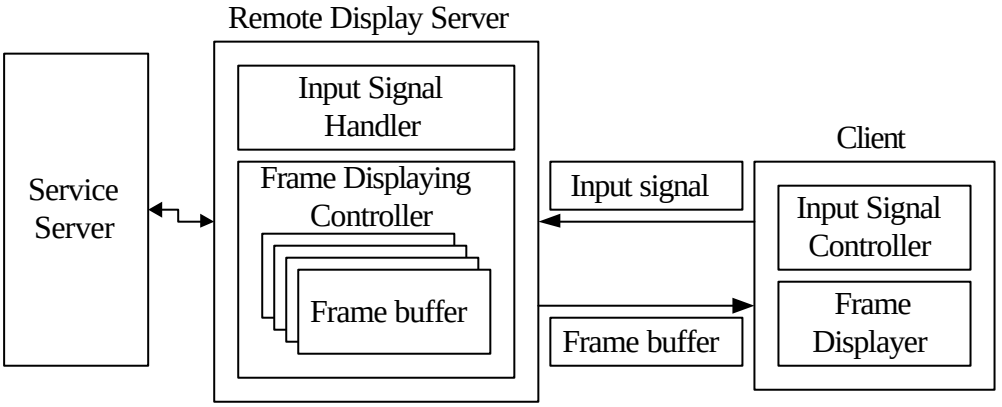

圖2-4. 遠端顯示技術所發展的瘦身型客戶端技術說明圖

2.4 XML（eXtensible Markup Language）

XML是由World Wide Web Consortium(W3C)所制訂的一個用來表現資料及文件中內含意義的語言[7, 27]。XML是屬於SGML(Standard Generalized Markup Language)內的一個子集合，使用此技術的一個很重要的優點，是其可以允許使用者自行定義一些標籤(tags)以闡述文件或資料中的「語意」(semantic)，同時XML的文件也

是一種結構化的文件格式。XML與HTML雖然同歸類於markup language，但是XML可以自行定義標籤比之HTML只能使用已定義好的標籤有相當大差異，其更具彈性與延伸性，所以XML被稱作meta-markup language更為適當。XML是一個可以表現文件結構及語意的語言，與HTML只強調在表現文件的顯示格式(format)上有所不同。一般而言，一份XML文件並不會包含其顯示效果的資訊，需另行配合使用XSL(eXtensible Style Language)[26]及CSS(Cascading Style Sheet)[6]來定義文件的顯示格式。

XML可透過定義文件的DTD(Document Type Definition)以自行創造出符合需求的標籤及其屬性(attribute)。DTD所代表的是XML文件中之語法(syntax)，若純以資料而言也可以將之比擬為資料庫(database)的綱要(schema)[7]。在使用上，每一份XML文件都必須有其對應的DTD，而且撰寫文件時也需遵循此DTD所定義的語法。從另一角度觀之，我們也可以透過解析DTD來驗證XML文件結構的正確性，並於parsing後，利用如DOM(Document Object Model)[14]技術進而產生該文件的樹狀結構圖(tree structure)，以便為其他程式所使用。圖2-5、2-6與2-7分別為一個XML文件的DTD、範例文件與其樹狀結構範例圖。由圖2-6不難看出這個範例是關於歌曲相關資訊的XML文件，而整個編輯語法則需遵循圖2-5的DTD。其中我們定義了一些標籤(曲目、作曲人、製作人、出版商、歌曲長度、出版年份及演唱人)及整份文件的結構關係，以表現一首歌曲中相關資訊所隱含的語意。

```
<!ELEMENT SONG (TITLE,COMPOSER*,PRODUCER,PUBLISHER,LENGTH,YEAR,ARTIST)>
   <!ELEMENT TITLE (#PCDATA)>
   <!ELEMENT COMPOSER (#PCDATA)>
   <!ELEMENT PRODUCER (#PCDATA)>
   <!ELEMENT PUBLISHER (#PCDATA)>
   <!ELEMENT LENGTH (#PCDATA)>
   <!ELEMENT YEAR (#PCDATA)>
   <!ELEMENT ARTIST (#PCDATA)>
```

圖2-5. XML DTD範例

```
<!DOCTYPE SONG SYSTEM "song.dtd">
<SONG>
  <TITLE>Hot Cop</TITLE>
  <COMPOSER>Jacques Morali</COMPOSER>
  <COMPOSER>Henri Belolo</COMPOSER>
  <COMPOSER>Victor Willis</COMPOSER>
  <PRODUCER>Jacques Morali</PRODUCER>
  <PUBLISHER>PolyGram Records</PUBLISHER>
  <LENGTH>6:20</LENGTH>
  <YEAR>1978</YEAR>
  <ARTIST>Village People</ARTIST>
</SONG>
```

圖2-6. XML範例文件

圖2-7. XML樹狀結構範例圖

2.5 Java 2 Platform Enterprise Edition (J2EE)

J2EE是Sun Microsystems所提出適用於發展電子化企業（e-Business）的軟體規格，圖2-8則是利用J2EE所定義的一個系統架構圖。在整個架構中，主要可分為三個部分，分別是client tier、EIS（Enterprise Information System）tier及middle tier。Client tier泛指使用此應用系統的客戶端類型，如專屬的Java application或是web Browser等。EIS tier則是指一般企業內的資訊系統，如ERP系統或資料庫等。而整個軟體架構最重要的部分乃在於middle tier，因為其除了提供client tier所需服務外，同時扮演了client tier與EIS tier兩部分結合的橋樑，所以必須提供許多整合能力強大的API。在middle tier中，大致上可分為三個部分，Web Container、EJB Container及其他相關API(如JNDI、JMS與JavaMail等)。在Web Container中包含Java Servlet與

JSP(JavaServer Pages)兩個元件，其提供了client tier HTTP請求的服務，可以動態產生HTML或XML的網頁，並可存取EJB Container中的EJB（Enterprise JavaBeans）[21]。EJB是middle tier中最為重要的規格之一，透過實作企業邏輯於EJB元件中，client tier乃能輕易地與EIS tier裡的服務相整合。而EJB元件透過EJB Container所提供的軟體服務，例如像transaction service等，可以有效地縮短軟體開發時程。最後其提供了許多可整合其他相關服務的API，以擴大其應用的範圍。其中如JNDI(Java Naming and Directory Interface)可整合名稱(naming)及目錄(directory)的服務;JMS(Java Message Service)可提供點對點(point-to-point)及發行-訂閱(publish-subscribe)的訊息傳遞模式;JavaMail則提供了電子郵件(email)收發的服務等。目前大部分商業用途的應用伺服器(application server)，除微軟（Microsoft）平台外，幾乎皆已遵循J2EE的軟體架構規格。

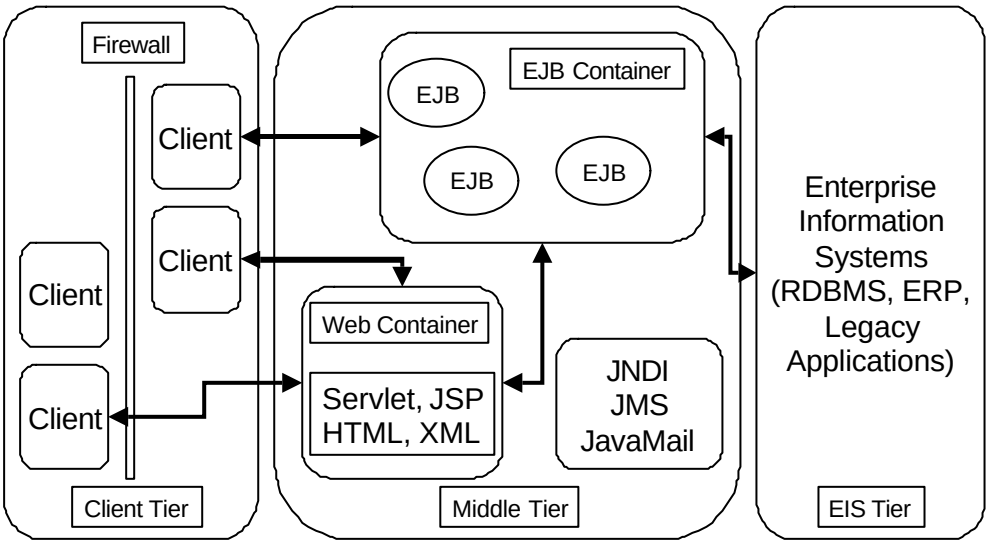

圖2-8. J2EE軟體架構圖

第三章 Thin-Client MVC 設計樣式

在本章中為支援發展本研究裡的瘦身型客戶端技術，我們乃提出一個新的設計樣式（design pattern），叫做thin-client MVC。設計樣式可以記錄設計軟體時所累積的經驗，若應用於軟體設計時，可被重複應用於解決同類型問題。對每一個設計樣式而言，其必須包括描述問題本身及解決該問題的方法[9, 10]。Thin-client MVC是改變自原來的MVC 設計樣式，此設計樣式將成為建構本研究之瘦身型主從式計算軟體架構的設計依據。

3.1 目的

為了將傳統MVC設計樣式中的view組成元件分離成另外兩個組成元件 – ViewDescriptor 及 ViewDisplayer，以達到兩個元件間可以各自獨立運作的好處。

3.2 動機

發展如圖 3-1 中之瘦身型計算模式，原則上必須將 view 的展現邏輯（VD_1）與 view 的顯示計算（VD_2）分別獨立運作並置於伺服端及客戶端。而客戶端中的 VD_2 元件與伺服端的其他元件間溝通，可透過網路進行之。

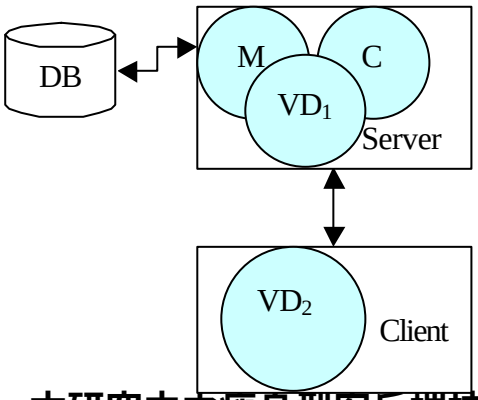

圖 3-1. 本研究中之瘦身型客戶端技術示意圖

3.3 應用時機

- 當想要實作一個極瘦小的客戶端時。
- 當一個物件狀態被改變並且其他物件的狀態也需要被改變時，但被改變物件的個數並不需要被知道。
- 當一個物件需要通知(notify)其他物件訊息，但其與被通知物件間的耦合(coupling)程度很低。

3.4 架構圖

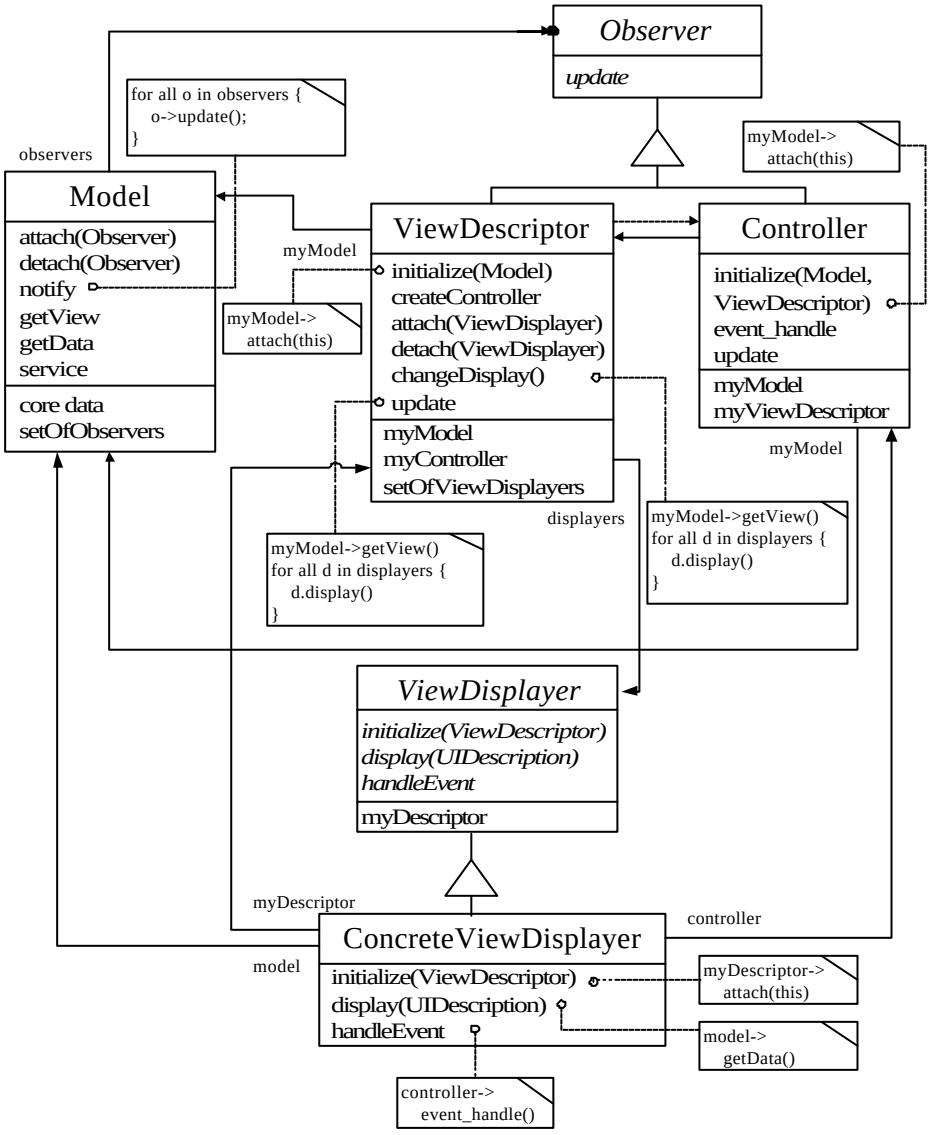

圖 3-2. Thin-client MVC **設計樣式**

3.5 組成元件

- ViewDescriptor
  - 可以索引(reference)到 model 元件。
  - 能夠產生相對應的 controller 元件。
  - 提供 ViewDisplayer 註冊的機制。一個 ViewDescriptor 只能擁有一個 ViewDisplayer。
  - 提供 ViewDisplayer 元件註冊及移除的介面。
  - 提供可以改變 ViewDisplayer 狀態的介面。
  - 需實作 Observer 元件中 update 的介面以維持與 model 元件狀態的一致性。
- ViewDisplayer
  - 可以分別索引到 model 及 controller 元件。
  - 提供可以顯示在 ViewDescriptor 中所定義圖形使用者介面的介面。
  - 提供可以呼叫其對應 controller 元件中事件處理方法的介面。

3.6 合作關係

- 當 ConcreteViewDisplayer 事件被觸發時，會呼叫 controller 中 event_handle 方法(method)，而後 controller 可能會要求 model 的服務或者是透過呼叫 ViewDescriptor 中 changeDisaply 方法切換成另一個顯示畫面。
- 當 Model 被要求服務時，會通知 Observers 更新成最新的狀態。而 ViewDescriptor 會呼叫 model 中 getView 方法以取得其最新的圖形使用者介面資料。最後再呼叫 ViewDisplayer 中 display 方法以顯示畫面到客戶端。

- 若 controller 需要切換目前的顯示畫面，則會直接呼叫 model 中 getView 方法以獲得圖形使用者介面的資料。然後透過呼叫 changeDisplay 方法以傳遞這些資料到 ViewDescriptor。最後在實作 changeDisplay 方法時，會呼叫 ViewDisplay 中的 display 方法而顯示到客戶端畫面上。

3.7 結果討論

Thin-client MVC 設計樣式擁有下列幾項優點：

1. 物件間的耦合度降低了。本設計樣式將 view 物件切分成 ViewDescriptor 與 ViewDisplayer 物件，使得原有物件間的耦合度降低了。
2. 可動態抽換 (pluggable) 的 ViewDisplayer。由於 ViewDisplayer 與其他元件間的耦合度相當低，因此在實做 ViewDisplayer 時變得相當簡易，甚至可以達到在執行程式間(run-time)動態地更換 Concrete ViewDisplayer。

Thin-client MVC 設計樣式缺點條列如下：

1. 程式複雜度增加。將 view 元件分成 ViewDescriptor 及 ViewDisplayer 後，整體程式實作的複雜度增加了不少。雖然降低了元件的耦合度，但是組成元件間的通訊量卻增大了。
2. 更新物件狀態時，所需更新的物件增加了。在傳統 MVC 設計樣式中，當 view 狀態改變時，所需更新的物件狀態是其相關的 model 物件及所有與這 model 物件相關的其他 view 物件。但在 thin-client MVC 設計樣式中，當 ViewDisplayer 物件狀態改變時，所需更新的物件有其所屬

的 ViewDescriptor 物件狀態外，還有與其相關的 model 物件，乃至到與這 model 物件相關的其他 ViewDescriptor 物件及其所屬的 ViewDisplayer 物件狀態。

3.8 範例程式

Model 類別的虛擬碼定義如下：

```
public class Model {
   Vector observers;
   public void attach(Observer o) {
      observers.add(o);
   }
   public void detach(Observer o) {
      observers.remove(o);
   }
   public void notify() {
      for (int i=0;i<observers.size();i++) {
         observers.get(i).update();
      }
   }
   public UIDescription getView() {
      ...
   }
   public String getData() {
      ...
   }
   public void service() {
      ...
   }
}
```

其中 notify()啟動後會致使相關的 Observer 物件更新其狀態，而 getView()則是用來取得目前最新的顯示畫面的描述物件 – UIDescription。

Observer 抽象化類別及其實體子類別 ViewDescriptor 及 Controller 定義如下：

```
public abstract class Observer {
   public abstract update();
}

public class Contorller extends Observer {
   public void initialize(Model m, ViewDescriptor vd) {
      ...
   }
   public void event_handle() {
      ...
   }
   public void update() {
      ...
   }
}
```

在 Controller 中 event_handle()是用來處理 ViewDisplayer 所

觸發的事件。

```
public class ViewDescriptor extends Observer {
   ViewDisplayer displayer;
   public void initialize(Model m) {
      ...
   }
   public void createController() {
      ...
   }
   public void attach(ViewDisplayer vd) {
      displayer = vd;
   }
   public void detach(ViewDisplayer vd) {
     if (displayer == vd) {
       displayer = null;
     }
   }
   public void changeDisplay() {
      UIDescription ui = m.getView();
      for (int i=0;i<displayers.size();i++) {
        displayers.get(i).display(ui);
      }
   }
   public void update() {
      ...
      UIDescription ui = m.getView();
      for (int i=0;i<displayers.size();i++) {
        displayers.get(i).display(ui);
      }
   }
}
```

在 ViewDescriptor **中** changeDisaply()**及** update()**都會驅使其** ViewDisplayer **更新其目前顯示的畫面。**

ViewDisplayer **抽象化類別及其實體子類別** AppletDisplayer **及** HtmlDisplayer **定義如下：**

```
public abstract ViewDisplayer {
   public initialize(ViewDescriptor vd) {
      ...
   }
   public abstract void display(UIDescription ui);
   public abstract void handleEvent();
}
```

ViewDisplayer **只定義了抽象化的介面，所有的介面需由其子類別實作之。**

```
public class AppletDisplayer extends ViewDisplayer {
   public void display(UIDescription ui) {
      /* use applet power GUI implementation */
      ...
   }
   public void handleEvent() {
     controller.event_handle();
   }
}
```

在 AppletDisplayer **中我們可以在** display(UIDecription)**實作** Java applet **的顯示介面，其可以提供相當豐富的圖形使用者介面。**

```
public class HtmlDisplayer extends ViewDisplayer {
   public void display(UIDescription ui) {
      /* use HTML poor form implementation */
      ...
   }
   public void handleEvent() {
      controller.event_handle();
   }
}
```

HtmlDisplayer **不同於** AppletDisplayer **在於其** display(UIDecription)**中的實作方法，其利用了** HTML **所提供的圖形使用者介面，所能顯示的介面較為有限。**

第四章 以 XML 技術為基礎的瘦身型主從式計算軟體架構

在本章中，我們利用 thin-client MVC 設計樣式發展出一個新的瘦身型主從式計算軟體架構(thin-client-server framework)。另外此架構中我們搭配了 XML 的技術發展出一個可以表現圖形使用者介面及結合 J2EE 軟體架構的 XML 規格 – TCML。最後，我們還定義了一個適用於此架構下客戶端與伺服端間的通訊協定 – TCTP。

4.1 軟體架構

根據上一節所介紹的 thin-client MVC 設計樣式，我們發現若客戶端能夠完全實作出 ViewDisplayer 這個部分，則一個瘦身型的客戶端系統將可以被實現。更進一步的來看，若我們可以將一個應用系統中所需使用到的圖形使用者介面元件歸類出來，並且能夠制訂一個描述這些圖形使用者介面元件的描述語言，則要實作出一個可應用的系統將是可達成的目標 圖 4-1 就是利用上述想法所建構出的一個以 XML 技術為基礎的瘦身型客戶端軟體架構。

在此軟體架構中，客戶端包含兩個部分，分別是 *Thin-Client Markup Language*(TCML) displayer 及 En/De-cryption Handler，其中 TCML 是一種遵照 XML 標準所訂定的文件並將在本章後段有較詳盡的描述, 而 En/De-cryption Handler 的功能則將在介紹 thin-client server 組成元件時會有詳盡的說明。另外 TCML displayer 的功能是負責解譯 *Thin-Client Transport Protocol*(TCTP)訊息中所含的 TCML 文件, 使其顯示成客戶端的顯示畫面。當客戶端觸發某個事件時，TCML displayer 會將此事件的相關資料送給 En/De-cryption Handler 編密而後包裝成 TCTP 的訊息格式再透過網路送至 thin-client server 處理之。其中，TCTP 是本架構中客戶端和伺服端傳輸訊息時所使用的通訊協定，也將在本章後段有詳盡的介紹。

thin-client server 中包含了四個組成元件，分別是 En/De-cryption Handler、Adapter、TCML Template DB 及 EJB Server。而 Adapter 元件又是由兩個子元件所組成，分別為 TCML Adapter 及 EJB Adapter，所有 thin-client server 的組成元件分別條列並說明如下：

- En/De-cryption Handler：主要的功能有兩個，分別是產生/解譯 TCTP messages與編密/解密TCTP messages。在此架構中客戶端與伺服端傳遞訊息需皆遵守TCTP通訊協定, 所以此組成元件有個功能就是產生及解譯TCTP messages。另外還有個很重要的功能就是，為維持系統的資料安全性，此架構下的客戶端與伺服端間傳送的TCTP messages都必需經過編密處理，當另一方收到後再解密處理以取得明文(plain text)。
- TCML Adapter：當TCML displayer有事件被觸發時，其會將一些相關資料傳遞給TCML Adapter處理。而TCML Adapter會根據資料中的描述處理之。其處理方式共分為兩種，分別是取得新的TCML及透過EJB Adapter去取得回應。在取得新的TCML方面，TCML Adapter會至TCML Template DB去取出目前所需TCML文件的樣版文件(template document)，經處理後再將含有完整圖形使用者介面資料的TCML文件送至客戶端顯示之。在某些情況下樣版文件中圖形使用者介面所需呈現的資料並不是事先定義好的靜態資料，而是來自於Enterprise JavaBeans(EJB)動態產生的即時資料，這時TCML Adapter就需透過EJB Adapter向EJB取得對應的資料。
- EJB Adapter：EJB Adapter的工作主要是負責連線至適當的EJB Server呼叫對應的EJB，若有回傳資料則會傳遞給TCML Adapter。
- TCML Template DB：其為一個專門存放TCML樣版文件的資料

庫，也可以說是存放所有應用程式圖形使用者介面的儲存庫。所有在客戶端所顯示的畫面都是以樣版文件的方式存放在資料庫中，而此樣版文件可以看成是Thin-Client MVC中ViewDescriptor的角色，待其真正要送至客戶端顯示時才會轉換成含有完整資料的TCML文件。

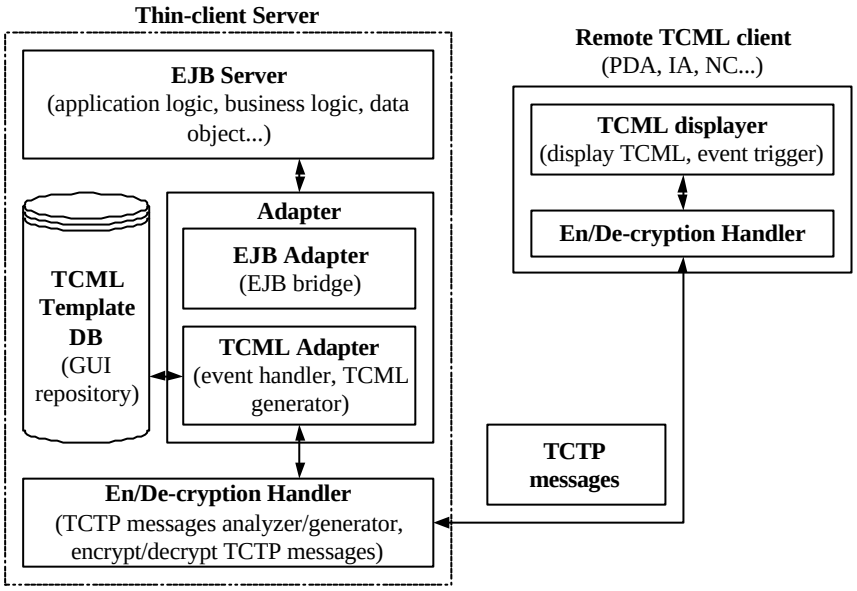

圖 4-1. 以 XML 技術為基礎的瘦身型客戶端軟體架構

- EJB Server：這裡的EJB Server必須是遵照J2EE規格書中所訂之標準。目前大部分的應用伺服器(application server)都是根據EJB Server標準所開發的產品，如NAS, Web Logic等。在其中可以定義企業本身的企業邏輯、系統本身的應用程式主邏輯及資料庫存取相關的data object等。

4.2 Thin-Client Markup Language(TCML)

TCML 是遵循 XML 標準所定義的文件格式，其 DTD 如圖 4-2 所示。TCML 最初發展的目的除了要將圖形使用者介面利用 XML 文件描述外，另外一個目的就是要支援瘦身型客戶端的系統運作架構。所以目前已知的一些描述使用者介面的 XML 文件，如 UIML[1]及

```
<!ELEMENT tcml (view,datamodel,events)>
<!ATTLIST tcml name CDATA #REQUIRED>
 <!ELEMENT view (frame)>
  <!ELEMENT frame (menu+,panel)>
  <!ATTLIST frame name ID #REQUIRED
                 width CDATA #IMPLIED
                 height CDATA #IMPLIED>
   <!ELEMENT menu (menuitem|menuitems)*>
   <!ATTLIST menu name ID #REQUIRED
                 enabled (true|false) "true"
                 mnemonic CDATA #IMPLIED
                 accelerator CDATA #IMPLIED>
    <!ELEMENT menuitems (menuitem|menuitems)*>
    <!ATTLIST menuitems name ID #REQUIRED
                      enabled (true|false) "true"
                      mnemonic CDATA #IMPLIED
                      accelerator CDATA #IMPLIED>
    <!ELEMENT menuitem EMPTY>
    <!ATTLIST menuitem name ID #REQUIRED
                     enabled (true|false) "true"
                     mnemonic CDATA #IMPLIED
                     accelerator CDATA #IMPLIED>
   <!ELEMENT panel (border|flow|grid)>
   <!ATTLIST panel name ID #REQUIRED>
   <!ELEMENT border (component)*>
   <!ELEMENT component (button|canvas|combobox|checkbox|list|menu|table|textarea|textfield|tree|panel)*>
   <!ATTLIST component position (east|west|north|south|center) "center">
   <!ELEMENT flow (button|canvas|combobox|checkbox|list|menu|table|textarea|textfield|tree|panel)*>
   <!ATTLIST flow alignment (left|right|center) "center">
   <!ELEMENT grid (button|canvas|combobox|checkbox|list|menu|table|textarea|textfield|tree|panel)*>
   <!ATTLIST grid row CDATA #IMPLIED
                 column CDATA #IMPLIED>
    <!ELEMENT button EMPTY>
    <!ATTLIST button name ID #REQUIRED
                   enabled (true|false) "true">
    <!ELEMENT canvas EMPTY>
    <!ATTLIST canvas name ID #REQUIRED>
    <!ELEMENT combobox EMPTY>
    <!ATTLIST combobox name ID #REQUIRED
                     enabled (true|false) "true">
    <!ELEMENT checkbox EMPTY>
    <!ATTLIST checkbox name ID #REQUIRED
                     enabled (true|false) "true"
                     selected (true|false) "false">
    <!ELEMENT list EMPTY>
    <!ATTLIST list name ID #REQUIRED
                  enabled (true|false) "true"
                  mode (single|multiple) "single">
    <!ELEMENT table EMPTY>
    <!ATTLIST table name ID #REQUIRED
                   enabled (true|false) "true"
                   mode (row|column|cell|none) "row">
    <!ELEMENT textarea EMPTY>
    <!ATTLIST textarea name ID #REQUIRED
                     enabled (true|false) "true"
                     editable (true|false) "true"
                     allownull (true|false) "true">
    <!ELEMENT textfield EMPTY>
    <!ATTLIST textfield name ID #REQUIRED
                      enabled (true|false) "true"
                      editable (true|false) "true"
                      allownull (true|false) "true">
    <!ELEMENT tree EMPTY>
    <!ATTLIST tree name ID #REQUIRED
                  enabled (true|false) "true">
    <!ELEMENT passwordfield EMPTY>
    <!ATTLIST passwordfield name ID #REQUIRED
                            allownull (true|false) "true">
<!ELEMENT datamodel (content)*>
 <!ELEMENT content (#PCDATA)>
 <!ATTLIST content view IDREF #REQUIRED
                  type (text|dsml|image) #REQUIRED>
<!ELEMENT events (event)*>
 <!ELEMENT event ((if|choose|action)?,params?)>
 <!ATTLIST event name ID #REQUIRED
                target CDATA #REQUIRED
                dialog CDATA #IMPLIED>
  <!ELEMENT if (action?)>
  <!ATTLIST if condition CDATA #REQUIRED>
  <!ELEMENT choose (when+,otherwise?)>
   <!ELEMENT when (action?)>
   <!ATTLIST when condition CDATA #REQUIRED>
       <!ELEMENT otherwise (action?)>
     <!ELEMENT action (#PCDATA)>
     <!ATTLIST action preprocess CDATA #IMPLIED>
     <!ELEMENT params (param)*>
       <!ELEMENT param (dsml)>
       <!ATTLIST param name ID #REQUIRED>
```

圖 4-2. TCML 之 DTD

TCML 是一個能夠描述圖形使用者介面及事件觸發時之處理方式的文件，所以整個文件大致上的結構可分成三大部分，分別是 view、datamodel 及 events。其中 view 是描述圖形使用者介面，而在圖形使用者介面的元件中我們提供了視窗(frame)、選單(menu and menuitem)、按鈕(button)、畫布(canvas)、combobox、checkbox、list、table、文字輸入框(textarea)、文字輸入列(textfield)、樹狀結構圖(tree)及密碼輸入列(password field)等可供使用。在 datamodel 中則是描述了圖形使用者介面中所需顯示的資料，而其支援了三種資料格式分別為一般文字、DSML、及影像檔。其中 DSML(Data Structure Markup Language)也是一種 XML 格式的文件，其提供了一種結構化的方式來描述所需展現的資料，DSML DTD 如圖 4-3 所示。最後，events 則是描述了整個圖形使用者介面中事件觸發的部分。若使用者觸發了 TCML 中所定義的事件時, TCML 客戶端就會將與此事件相關的資料（定義在 event element 中）送至 thin-client server 處理。

```
<!ELEMENT dsml (list|multilist|aggregate|table|tree)>
  <!ELEMENT list (element*)>
  <!ATTLIST list size CDATA #REQUIRED>
    <!ELEMENT element (#PCDATA)>
    <!ATTLIST element index CDATA #REQUIRED>
  <!ELEMENT multilist (object*)>
  <!ATTLIST multilist size CDATA #REQUIRED>
    <!ELEMENT object (dsml|#PCDATA)>
    <!ATTLIST object index CDATA #REQUIRED>
  <!ELEMENT aggregate (entry*)>
  <!ATTLIST aggregate size CDATA #REQUIRED
                   default CDATA #REQUIRED>
    <!ELEMENT entry (#PCDATA)>
    <!ATTLIST entry index CDATA #REQUIRED>
  <!ELEMENT table (columnname, tuple*)>
  <!ATTLIST table columnsize CDATA #REQUIRED>
    <!ELEMENT columnname (list)>
    <!ELEMENT tuple (list|multilist)>
    <!ATTLIST tuple index CDATA #REQUIRED>
  <!ELEMENT tree (root)>
    <!ELEMENT root (data,node*)>
    <!ATTLIST root name ID #REQUIRED
                childcount CDATA #REQUIRED>
      <!ELEMENT node (data,node*)>
      <!ATTLIST node name ID #REQUIRED
                  parent IDREF #REQUIRED
                  childcount CDATA #REQUIRED
                  childindex CDATA #REQUIRED>
      <!ELEMENT data (dsml|#PCDATA)>
```

圖4-3. DSML之DTD

針對相同的 TCML 而言，我們可以搭配不同的 XSL 使其有不同的顯示外觀。目前我們提供了三種不同的顯示外觀，分別對應到 Metal(圖 4-4)、Motif(圖 4-5)及 Windows(圖 4-6)三種不同的圖形使用者介面樣式。在 TCML 客戶端的使用者就可依據個人喜好任意更換不同的 XSL，以產生不同的顯示外觀。

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <tcml>
    <xsl:attribute name="name">{@name}</xsl:attribute>
    <xsl:attribute name="laf">metal</xsl:attribute>
      <xsl:value-of select="."/>
    </tcml>
  </xsl:template>
</xsl:stylesheet>
```

圖4-4. metal.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <tcml>
    <xsl:attribute name="name">{@name}</xsl:attribute>
    <xsl:attribute name="laf">motif</xsl:attribute>
      <xsl:value-of select="."/>
    </tcml>
  </xsl:template>
</xsl:stylesheet>
```

圖4-5. motif.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <tcml>
    <xsl:attribute name="name">{@name}</xsl:attribute>
    <xsl:attribute name="laf">windows</xsl:attribute>
      <xsl:value-of select="."/>
    </tcml>
  </xsl:template>
</xsl:stylesheet>
```

圖4-6. windows.xsl

4.3 Thin-Client Transport Protocol (TCTP)

為了能夠確保此軟體架構能同時適用於網際網路、無線網路及企業內網路，所以特別制訂了一個適用此瘦身型客戶端應用的通訊協定 – TCTP。一般而言，企業內網路所屬的環境網路品質較佳，所以

在系統的運作上較不會有網路方面的問題，但在網際網路或無線網路的環境中就必須要考慮到網路品質較差時的連線問題。基於這個原因，我們設計了一套連線的規則，以增加網路通訊的強韌性（robutsness），其規則如下：

- 伺服端與客戶端間預設以 TCP 連線。
- 當 TCP timeout 時，客戶端與伺服端則全部改以 UDP 連線。
- 當使用 UDP 連線時，所有的 UDP packet 都必須被接收者確認，而回復確認的訊息給發送者。
- 在 UDP 連線時，發送訊息者每隔 10 秒就會重新送出 packet 直到收到確認訊息。
- 在 UDP 連線時，若連續傳送訊息六次後仍無法獲得確認訊息（即一分鐘後），則發送訊息者會送出一個 Message_Fail_Ack 的訊息，此時伺服端與客戶端自行清除此次連線狀態並斷絕連線。

以下將介紹 TCTP 連線傳送訊息的格式，並將其格式分成兩部分來討論，茲分述如下：

(1) TCTP 回應訊息格式(從伺服端傳至客戶端)：

表 4-1 是 TCTP 回應訊息的格式，每個從伺服端送至客戶端的 TCTP message 都必須符合此格式。以下是針對每個欄位的介紹：

- Version：這個欄位包含在所有的 TCTP 訊息中，用來識別這是一個 TCTP 訊息。
- Command：此欄位是用來描述伺服端回應客戶端需求的服務，詳細的服務項目條列在表 4-2。
- SeqNum：此欄位是使用在 UDP 通訊協定時，每個封包所編出的連續且唯一(除非封包重送)的號碼，避免 UDP 封包遺失或重複造成的混淆。

表 4-1. TCTP 回應訊息的格式(伺服端至客戶端)

Length	Name	Definition
2 bytes	Version	確認此為 TCTP messages
2 bytes	Command	回應客戶端需求的服務(續參考表 4-2)
2 bytes	SeqNum	訊息序列號碼

表 4-2. 回應客戶端需求的服務

Name	Definition
Ack	用以確認收到訊息的 command
Message_Fail_Ack	UDP message 傳送六次失敗後所送出的 command
SendTCML（參數）	送 TCML 文件至客戶端(續參考表 4-3)
SendSessionID	告知客戶端其 Session ID

表 4-3. SendTCML command 的參數

Length	Name	Definition
2 bytes	Length	TCML 的長度
variable	TCML	要傳送的 TCML

(2) TCTP 請求訊息格式(從客戶端傳至伺服端)：

表 4-4 是 TCTP 請求訊息的格式，每個從客戶端送至伺服端的 TCTP message 都必須符合此格式。由於 Version 及 SeqNum 與上述同，故以下是針對 Command 及 SessionID 欄位的詳述：

- Command：此欄位是用來描述客戶端請求伺服端的服務，詳細的請求服務的項目條列在表 4-5。
- SessionID：此欄位是客戶端用來告知伺服端其 Session ID，以辨識其身份。

表 4-4. TCTP 請求訊息的格式(客戶端至伺服端)

Length	Name	Definition
2 bytes	Version	確認此為 TCTP messages
2 bytes	Command	請求伺服端的服務(續參考表 4-6)
2 bytes	SeqNum	訊息序列號碼
2 bytes	SessionID	客戶端的 Session ID

表 4-5. 請求伺服端服務的 command

Name	Definition
Ack	用以確認收到訊息的 command
Message_Fail_Ack	UDP message 傳送六次失敗後所送出的 command
EventTrigger（參數）	將事件觸發的相關資料送至伺服端(續參考表 4-6)
KeepAlive	每隔 2 分鐘傳送一次，確認保持連線狀態的 command

表 4-6. EventTrigger command 的參數

Length	Name	Definition
2 bytes	Length	EventTrigger 資料的長度
variable	Event	要傳送的事件觸發的相關資料

第五章 範例系統實做

在本章中為了驗證前一章所提出軟體架構之可行性，故實做一個在企業界中常常會被應用到的範例系統。在此我們將先介紹此範例系統可應用的劇情，並帶出此範例系統所提供的功能。最後再針對部分功能的實做過程說明整個軟體架構的運作流程及方法，並附上對應的 TCML 及 DSML 範例文件以闡明設計理念及應用方法。

5.1 範例劇情簡介

在本章中，我們將實做一個非常簡單的人事資料庫管理系統。在一般的企業中，為了可以很方便地管理其員工的基本資料，所以通常會有一個人事資料庫管理系統管理之。在這個系統中提供了人事資料的「檢視所有人事資料」、「新增」、「刪除」、「修改」、「登入系統」時身份查驗與「登出系統」的功能，以下將假設一個劇情以模擬整個系統的運作。根據前一章所介紹的軟體架構，系統中每一個顯示畫面必須以 TCML 文件表現之，所以在劇情中也將分析出系統中所需的所有顯示畫面。假設某 A 公司來了一位新進員工王大明並且有位員工王小明正要離職，這時管理公司人事資料的人事部員工必須將王大明個人資料加入公司人事資料庫中，並且將王小明的人事資料從資料庫中刪除。第一步，此人事部員工必須先啟動此系統的 TCML 客戶端，所以會先進入此系統的「啟動畫面」。第二步，點一下此畫面後接著會進入需要身份驗證的「登入畫面」。第三步，在身份驗證無誤後，則進入此系統的「初始畫面」，此時的畫面將有登出系統及檢視所有人事資料的功能。第四步，按下檢視的按鈕後即產生「檢視所有人事資料畫面」，在此畫面中會列出目前人事資料庫中全部員工的資料列表，並且可針對人事資料做新增、修改及刪除的動作。第五步，為了增加王大明的資料進資料庫，所以按下新增按鈕則帶出「新增人事資

料畫面」。第六步，在新增資料之後隨即回到檢視所有人事資料畫面，但這時卻發現剛才輸入的王大明基本資料有誤，所以隨即按下修改按鈕而帶出「修改人事資料畫面」。第七步，修改確認無誤後再將離職的王小明資料刪除，而完成整個人事資料的管理作業。第八步，按下登出系統按鈕以結束此系統的運作。

5.2 實做方法及範例

為了要根據上一章所介紹之軟體架構實作出一個瘦身型客戶端應用系統，我們利用 Java 技術實做出一個十分瘦小的 TCML 客戶端，並且透過網路 socket 連線(同時支援 TCP 或 UDP 兩種連線方式)作為其與 thin-client server 的通訊方式。以下則是系統運作的流程，並且搭配軟體架構及其 TCML 文件的說明。

- 啟動人事資料庫管理系統，系統流程圖如圖5-1所示：

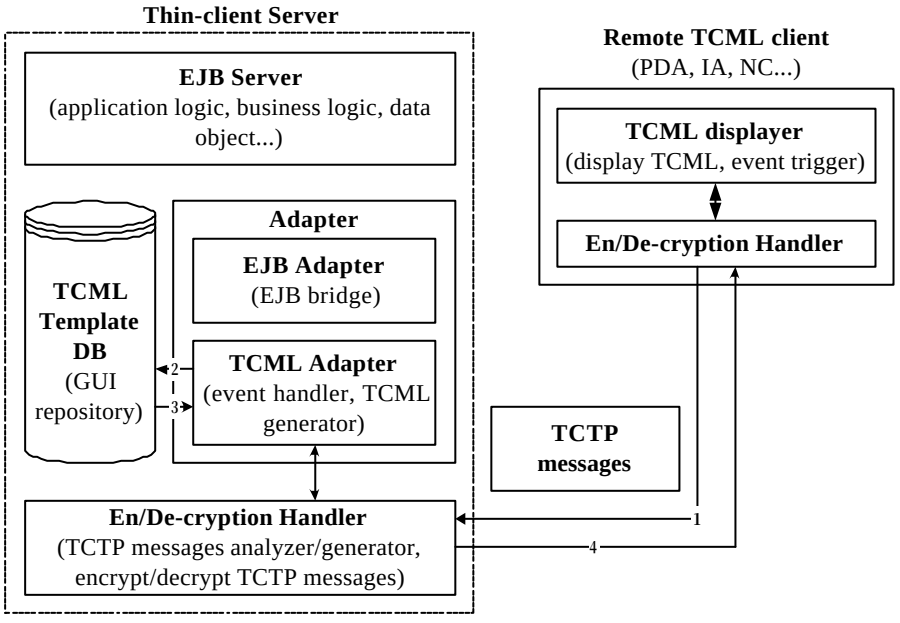

圖5-1. 系統流程圖(1)

1. TCML client透過Web Server向thin-client server請求一個為TCML格式的start.xml。

2. Thin-client server中的TCML Adapter會將此請求轉變為start_template.xml，並根據此名稱XML Template DB中query一個名稱為start_template.xml TCML樣版文件（圖5-2）。

```
<?xml version="1.0" encoding="Big5"?>
<?xml-stylesheet type="text/xml" href="metal.xsl"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="start_template">
  <view>
    <frame name="mainframe" width="500" height="400">
      <panel name="mainpanel">
       <border>
         <component position="center">
          <canvas name="logo"/>
         </component>
        </border>
      </panel>
    </frame>
  </view>
  <datamodel>
    <content view="logo" type="image">>bc.jpg</content>
  </datamodel>
  <events>
    <event name="logo_event" target="logo">
      <action>login.xml</action>
    </event>
  </events>
</tcml>
```

圖5-2. start_template.xml

3. XML Template DB則將尋找到的TCML樣版文件傳回給TCML Adapter，並透過其分析(parse)後會產生一個包含完整資料的TCML文件-start.xml（圖5-4），其流程如圖5-3所示。

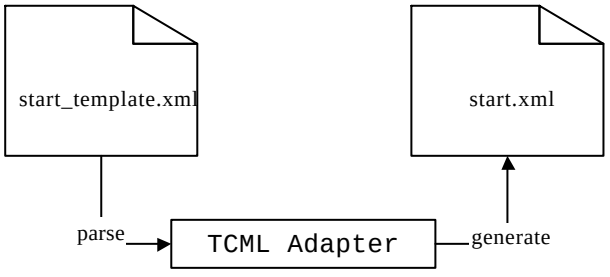

圖5-3. 分析樣版文件的流程

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="start">
  <view>
    <frame name="mainframe" width="500" height="400">
      <panel name="mainpanel">
        <border>
         <component position="center">
          <canvas name="logo"/>
         </component>
        </border>
      </panel>
    </frame>
  </view>
  <datamodel>
    <content view="logo" type="image">bc.jpg</content>
  </datamodel>
  <events>
    <event name="logo_event" target="logo">
      <action>login.xml</action>
    </event>
  </events>
</tcml>
```

圖5-4. start.xml

4. TCML Adapter**將結果傳給**TCML client**，顯示畫面如下圖**5-5。

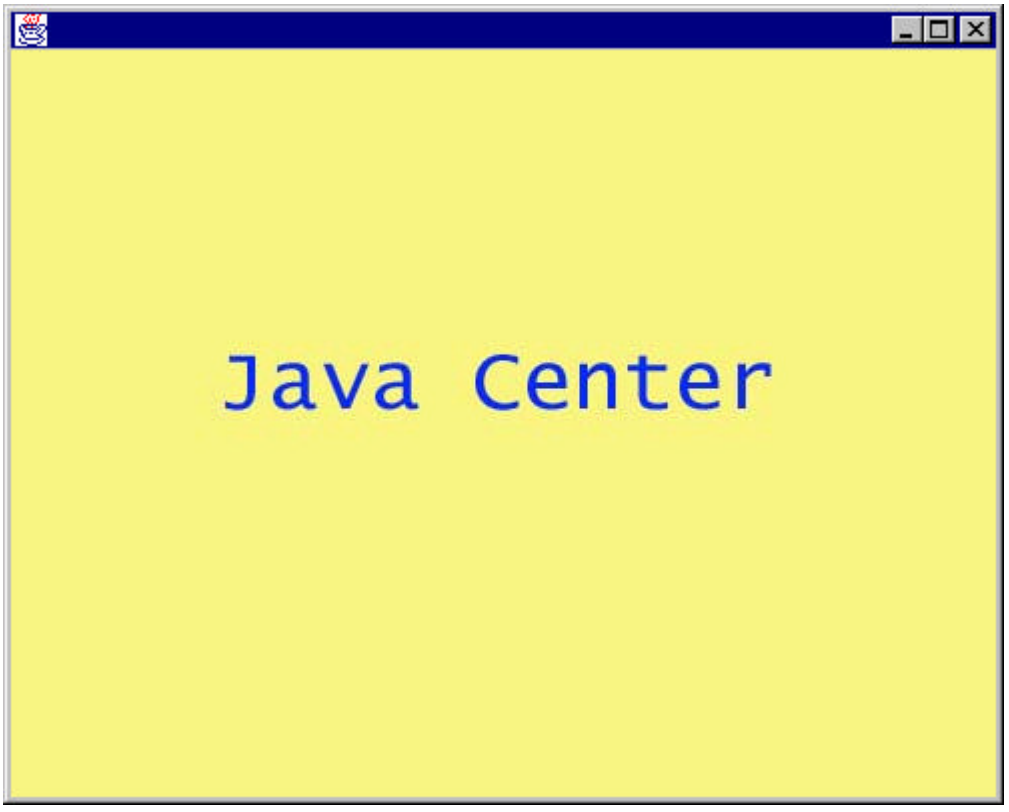

圖5-5. **啟動人事資料庫管理系統的畫面**

- **登入人事資料庫管理系統，其系統流程圖請參考圖**5-1**：**

1. **看到啟動畫面後，可利用滑鼠在畫面上點選。此時會觸發在**start.xml**中所定義的**logo**事件，**TCML client**則將**target**屬性**(attribute)**之值為**logo**的**event element**傳至**TCML Adapter**處理之，其流程如圖**5-6**所示。**

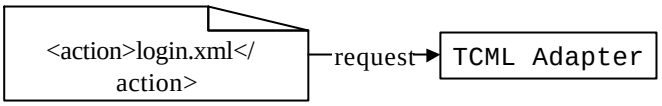

圖5-6. **觸發**logo**事件後的處理流程**

2. TCML Adapter**會根據**action element**中的請求並轉變為**login_template.xml**，而去**XML Template DB**中**query**一個名稱為**login_template.xml TCML**樣版文件（圖**5-7**）。**
3. XML Template DB**則將尋找到的**TCML**樣版文件傳回給**TCML Adapter**，並透過其分析後會產生一個包含完整資料的**TCML**文件**- login.xml**。由於在本例中的**login.xml**並無資料需要附加，故分析後的文件內容將與**login_template.xml**同，可參考圖**5-7**。**
4. TCML Adapter**將結果傳給**TCML client**，顯示畫面如下圖**5-8**。**

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="login_template">
 <view>
  <frame name="mainframe" width="500" height="400">
   <panel name="mainpanel">
    <border>
     <component position="north">
      <panel name="panel1">
       <grid row="3" column="1">
         <panel name="panel1_1">
          <flow alignment="center">
           <label name="acc_label"/>
           <textfield name="acc_text" allownull="false"/>
          </flow>
         </panel>
         <panel name="panel1_2">
          <flow alignment="center">
           <label name="paswd_label"/>
           <passwordfield name="paswd_text" allownull="false"/>
          </flow>
         </panel>
        <panel name="panel1_3">
          <flow alignment="center">
           <combobox name="sys_choice"/>
          </flow>
         </panel>
        </grid>
      </panel>
     </component>
    </border>
    <panel name="panel2">
     <border>
        <component position="south">
         <panel name="panel2_1">
        <flow alignment="right">
         <button name="okbtn"/>
           <button name="cancelbtn"/>
         </flow>
        </panel>
       </component>
     </border>
    </panel>
   </panel>
  </frame>
 </view>
 <datamodel>
  <content view="acc_label" type="text">帳號：</content>
  <content view="pswd_label" type="text">密碼：</content>
  <content view="sys_choice" type="dsml">
   <dsml:dsml xmlns:dsml=".">
    <dsml:list size="3">
     <dsml:element>人事資料系統</dsml:element>
     <dsml:element>財務金融系統</dsml:element>
     <dsml:element>遠距教學系統</dsml:element>
    </dsml:list>
   </dsml:dsml>
  </content>
  <content view="okbtn" type="text">確定</content>
  <content view="cancelbtn" type="text">取消</content>
 </datamodel>
 <events>
  <event name="ok_event" target="okbtn">
   <choose>
    <when condition="ejb://140.128.104.23/Employee.loginCheck(acc_text.getText(),
pswd_text.getPassword())">
     <choose>
      <when condition="sys_choice.getSelected()=&amp;quot;人事資料系統&amp;quot;">
       <action>main.xml</action>
      </when>
      <when condition="sys_choice.getSelected()=&amp;quot;財務金融系統&amp;quot;">
      </when>
      <when condition="sys_choice.getSelected()=&amp;quot;遠距教學系統&amp;quot;">
      </when>
     </choose>
    </when>
    <otherwise>
     <action>login.xml</action>
    </otherwise>
   </choose>
  </event>
  <event name="canel_event" target="cancelbtn">
   <action>none</action>
  </event>
 </events>
</tcml>
```

圖5-7. login_template.xml

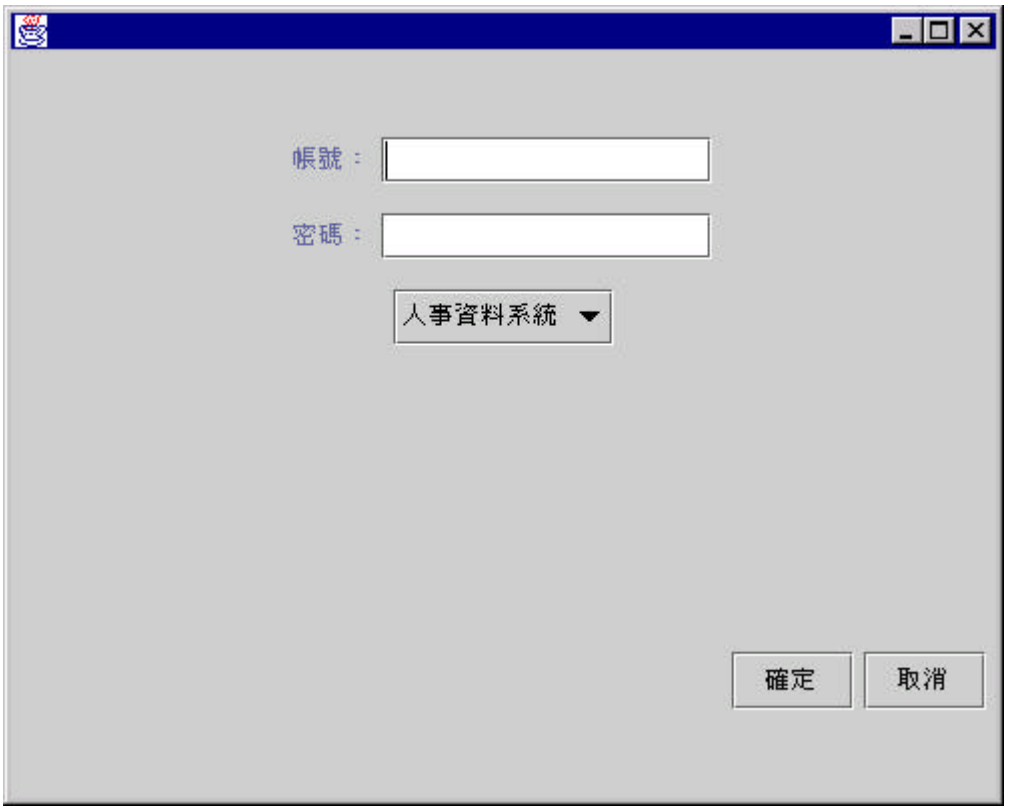

圖5-8. 登入人事資料庫管理系統的畫面

- 進入人事資料庫管理系統主畫面，系統流程圖如圖5-9所示：

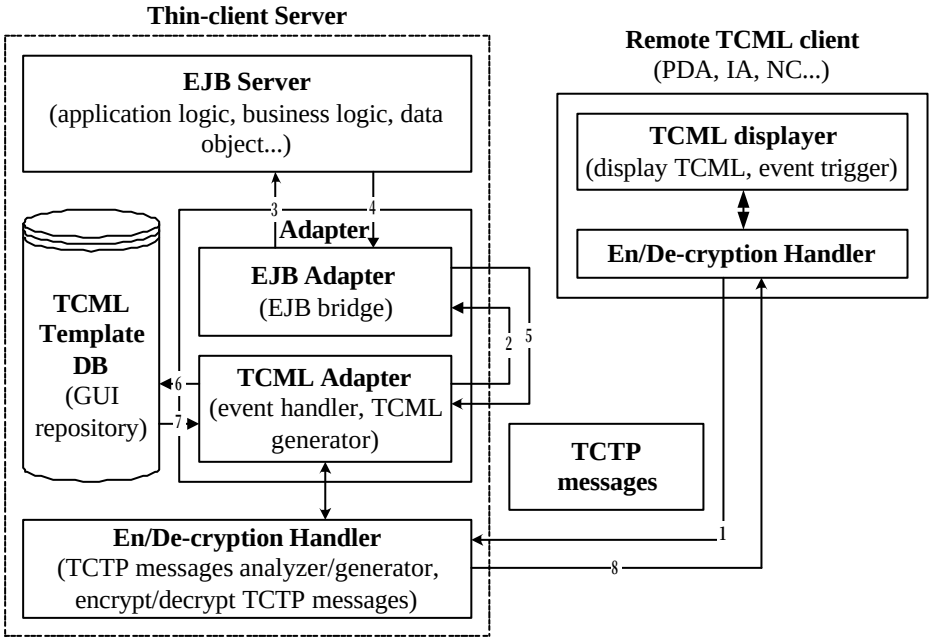

圖5-9. 系統流程圖(2)

1. 在圖5-8中，若帳號及密碼輸入正確並按下確定按鈕後（圖5-10），則可進入系統主畫面。此時會觸發在login.xml中所定義的okbtn事件，TCML client則將target屬性(attribute)之值為okbtn的event element傳至TCML Adapter處理之，其流程如圖5-11所示。

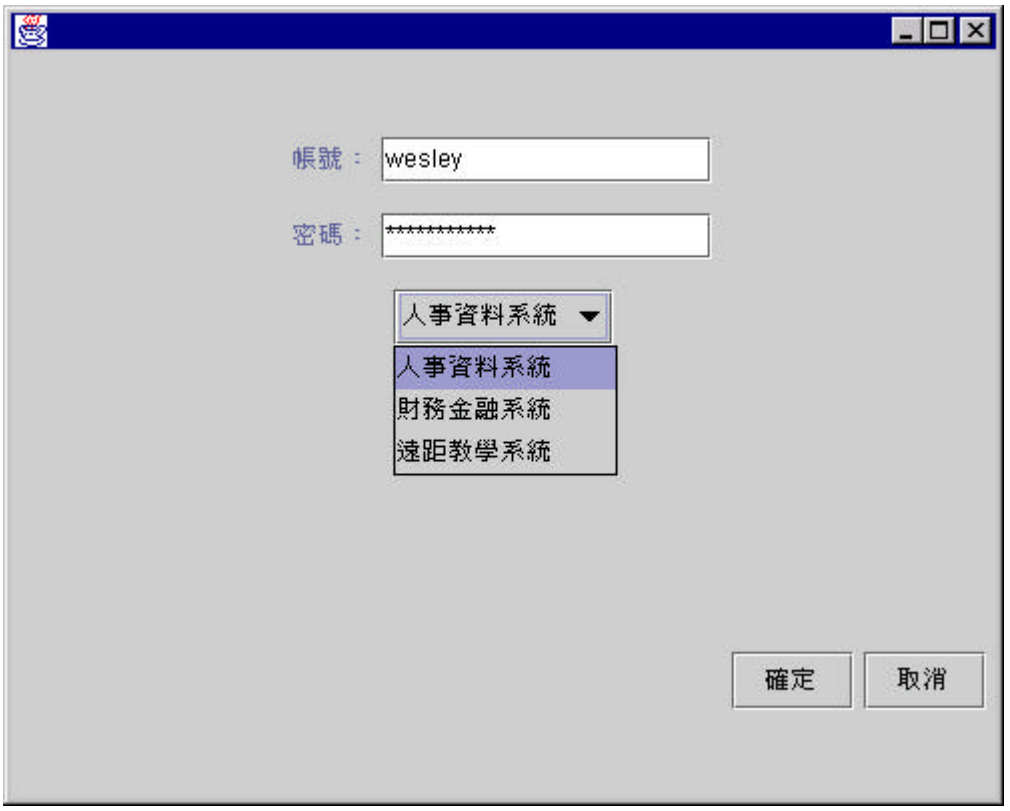

圖5-10. 登入系統主畫面前的認證畫面

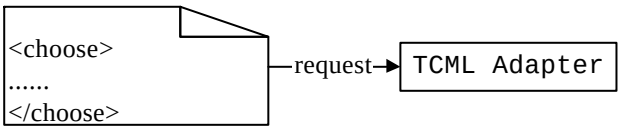

圖5-11. 觸發okbtn事件後的處理流程

2. 當TCML Adapter處理到when element之condition屬性所使用到的ejb時，其會將此請求交由EJB Adapter處理。EJB Adapter會根據其所記錄的URL尋找到EJB Server中對應的EJB，然後

再負責將結果回傳給TCML Adapter處理之。

3. 若此EJB回傳的結果為真，則會進行下一層choose element的判斷。最後若使用者在圖5-10的combobox中選擇了『人事資料系統』選項，則將根據action element中的請求並轉變為init_template.xml（圖5-12），而去XML Template DB中query一個名稱為init_template.xml的TCML樣版文件。若EJB回傳的結果為否，則會去query一個名稱為login_template.xml的TCML樣版文件。
4. XML Template DB則將尋找到的TCML樣版文件傳回給TCML Adapter，並透過其分析後會產生一個包含完整資料的TCML文件-init.xml（同圖5-12）。
5. TCML Adapter將結果傳給TCML client，顯示畫面如下圖5-13。

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="init_template">
  <view>
    <frame name="mainframe" width="500" height="400">
      <menu name="system"></menu>
      <menu name="view">
        <menuitem name="info"/>
      </menu>
      <menu name="search"></menu>
      <menu name="exit"></menu>
    </frame>
  </view>
  <datamodel>
    <content view="system" type="text">系統</content>
    <content view="view" type="text">檢視</content>
    <content view="info" type="text">人事資料</content>
    <content view="search" type="text">搜尋</content>
    <content view="exit" type="text">離開</content>
  </datamodel>
  <events>
    <event name="info_event" target="info">
      <action>main.xml</action>
    </event>
    <event name="exit_event" target="exit">
      <action>login.xml</action>
    </event>
  </events>
</tcml>
```

圖5-12. init_template.xml

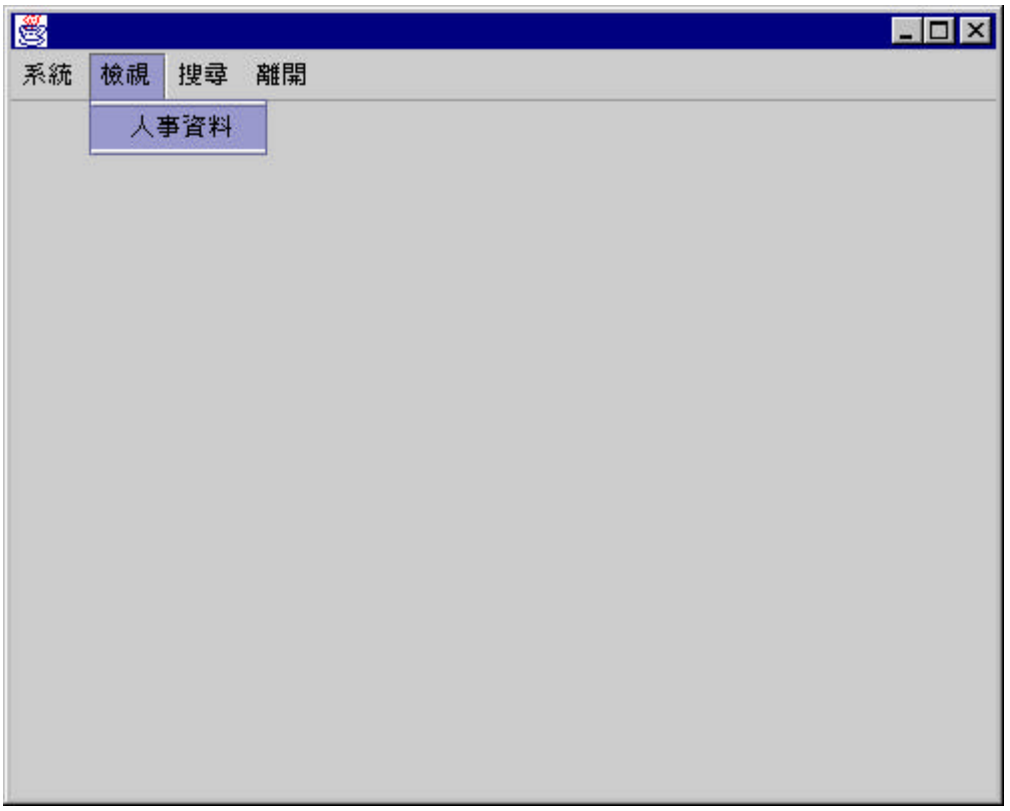

圖5-13. **人事資料庫管理系統主畫面**

- **檢視人事資料，系統流程圖如圖5-14所示：**

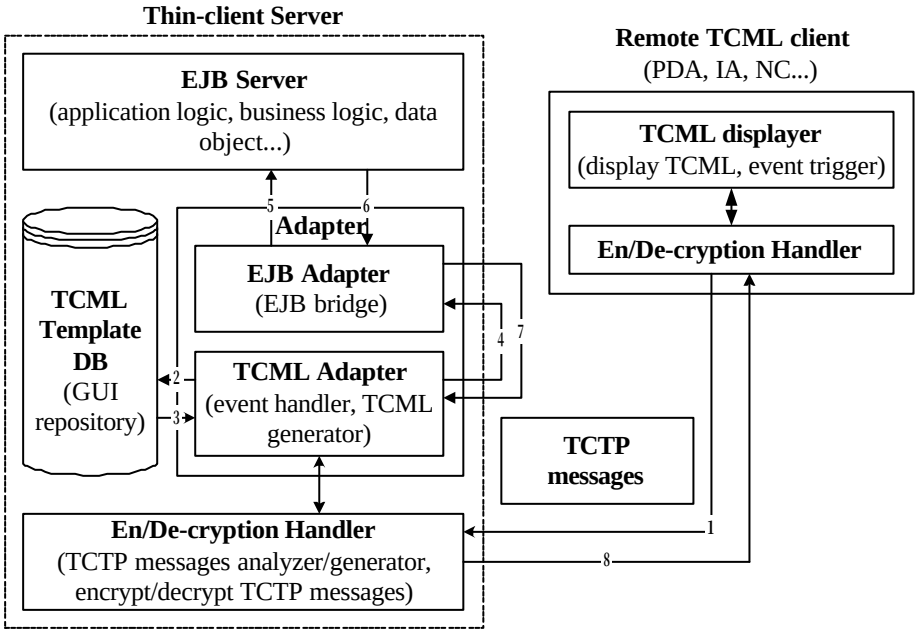

圖5-14. **系統流程圖**(3)

1. **進入系統主畫面後，按下『檢視』選單中的『人事資料』選項，則會顯示所有的人事資料（圖**5-13**），此時則會觸發在**init.xml**中所定義的**info**事件，** TCML client**則將**target**屬性**(attribute)**之值為**info**的**event element**傳至**TCML Adapter**處理之，其流程如圖**5-15**所示。**

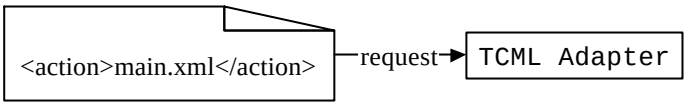

圖5-15. **觸發**info**事件後的處理流程**

2. TCML Adapter**會根據**action element**中的請求並轉變為**main_template.xml**（圖**5-16**），而去**XML Template DB**中**query**一個名稱為**main_template.xml**的**TCML**樣版文件。**
3. XML Template DB**則將尋找到的**TCML**樣版文件傳回給**TCML Adapter**，當其分析到文件中**content element**中所記錄的**ejb url**時，則會將此請求交由**EJB Adapter**處理。**EJB Adapter**會根據此**url**尋找到**EJB Server**中對應的**EJB**，然後再將結果回傳至**TCML Adapter**處理之。最後再由**XML Adatper**產生一個含有完整資料的**TCML**文件**-main.xml**。其流程如圖**5-17**所示，而圖**5-18**為**main.xml**文件的部分內容。**
4. TCML Adapter**將結果傳給**TCML client**，顯示畫面如下圖**5-19**。**

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="main_template">
  <view>
    <frame name="mainframe" width="500" height="400">
      <menu name="system"></menu>
      <menu name="view">
        <menuitem name="info"/>
      </menu>
      <menu name="search"></menu>
      <menu name="exit"></menu>
      <panel name="mainpanel">
      <border>
       <component position="center">
        <table name="infotable"/>
       </component>
       <component position="south">
        <panel name="option">
         <flow alignment="right">
          <button name="createinfo"/>
           <button name="updateinfo"/>
           <button name="delinfo"/>
           <button name="cancelbtn"/>
         </flow>
        </panel>
       </component>
      </border>
     </panel>
    </frame>
  </view>
  <datamodel>
    <content view="system" type="text">系統</content>
    <content view="view" type="text">檢視</content>
    <content view="info" type="text">人事資料</content>
    <content view="search" type="text">搜尋</content>
    <content view="exit" type="text">離開</content>
    <content view="infotable" type="dsml"><?invoke
ejb://localhost/Employee.getInfo()?></content>
    <content view="createinfo" type="text">新增</content>
    <content view="updateinfo" type="text">修改</content>
    <content view="delinfo" type="text">刪除</content>
    <content view="cancelbtn" type="text">取消</content>
  </datamodel>
  <events>
    <event name="info_event" target="info">
      <action>main.xml</action>
    </event>
    <event name="create_event" target="createinfo">
      <action>create.xml</action>
    </event>
    <event name="update_event" target="updateinfo">
      <action>update.xml</action>
      <params>
        <param name="row">infotable.getSelectedRow()</param>
      </params>
    </event>
    <event name="del_event" target="delinfo" dialog="本資料將被刪除！">
      <action>delete.xml</action>
      <params>
        <param>infotable.getSelectedRow()</param>
      </params>
    </event>
  </events>
</tcml>
```

圖5-16. main_template.xml

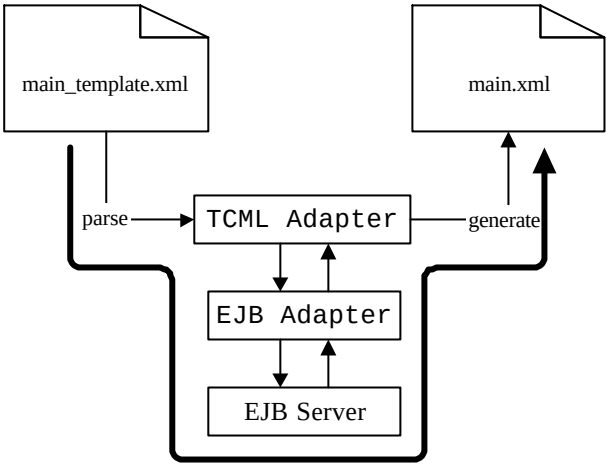

圖5-17. **分析樣版文件及取得完整資料的流程**

```
<datamodel>
  <content view="system" type="text">系統</content>
  <content view="view" type="text">檢視</content>
  <content view="info" type="text">人事資料</content>
  <content view="search" type="text">搜尋</content>
  <content view="exit" type="text">離開</content>
  <content view="infotable" type="dsml">
   <dsml:dsml xmlns:dsml=".">
    <dsml:table columnsize="4">
     <dsml:columnname>
      <dsml:list size="4">
       <dsml:element>姓名</dsml:element>
       <dsml:element>E-mail</dsml:element>
       <dsml:element>部門</dsml:element>
       <dsml:element>電話</dsml:element>
      </dsml:list>
     </dsml:columnname>
     <dsml:tuple>
      <dsml:list size="4">
       <dsml:element>王小明</dsml:element>
       <dsml:element>ming@javacenter.cis.thu.edu.tw</dsml:element>
       <dsml:element>CIS</dsml:element>
       <dsml:element>04-3590121</dsml:element>
      </dsml:list>
     </dsml:tuple>
    </dsml:table>
   </dsml:dsml>
  </content>
  <content view="createinfo" type="text">新增</content>
  <content view="updateinfo" type="text">修改</content>
  <content view="delinfo" type="text">刪除</content>
  <content view="cancelbtn" type="text">取消</content>
</datamodel>
```

圖5-18. main.xml**中**datamodel**的內容**

圖5-19. 檢視人事資料的畫面

- **新增人事資料，其系統流程圖請參考圖**5-14**：**
  1. **當使用者按下圖**5-19**中『新增』按鈕時則將出現新增人事資料的畫面（圖**5-20**），其**TCML**文件為**create.xml（圖**5-21**）。而填妥需新增的資料欄位後（圖**5-22**），按下『新增』按鈕則會將資料加入人事資料庫中。此時則會觸發在**create.xml**中所定義的**okbtn**事件，** TCML client**則將**target**屬性**(attribute)**之值為** okbtn**的**event element**傳至**TCML Adapter**處理之，其流程如圖** 5-23**所示。**
  2. TCML Adapter**會根據** action element**中的請求並轉變為** created_template.xml**（圖**5-24**），而去**XML Template DB**中**query**一個名稱為**created_template.xml**的**TCML**樣版文件。**
  3. XML Template DB**則將尋找到的**TCML**樣版文件傳回給**

TCML Adapter，當其分析到文件中content element中所記錄的ejb url時，則會將由TCML client所送來params element中記錄的資料當作參數而一併交由EJB Adapter處理。EJB Adapter會根據此URL尋找到EJB Server中對應的EJB，然後再將結果回傳至TCML Adapter處理之。最後再由XML Adatper產生一個含有完整資料的TCML文件-created.xml，圖5-25為created.xml文件的部分內容。

4. TCML Adapter將結果傳給TCML client，顯示畫面如圖5-26。

圖5-20. 新增人事資料的畫面

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="create">
  <view>
    <frame name="createframe" width="500" height="400">
      <menu name="system"></menu>
      <menu name="view">
        <menuitem name="info"/>
      </menu>
      <menu name="search"></menu>
      <menu name="exit"></menu>
      <panel name="mainpanel">
        <border>
         <component position="north">
          <panel name="panel1">
           <grid row="4" column="1">
            <panel name="panel1_1">
             <flow alignment="center">
              <label name="namel"/>
            <textfield name="nametxt" allownull="false"/>
             </flow>
            </panel>
            <panel name="panel1_2">
             <flow alignment="center">
              <label name="emaill"/>
            <textfield name="emailtxt" allownull="false"/>
             </flow>
            </panel>
            <panel name="panel1_3">
             <flow alignment="center">
              <label name="depl"/>
            <textfield name="deptxt" allownull="false"/>
             </flow>
            </panel>
            <panel name="panel1_4">
             <flow alignment="center">
              <label name="tell"/>
            <textfield name="teltxt" allownull="false"/>
             </flow>
            </panel>
           </grid>
          </panel>
         </component>
         <component position="south">
          <panel name="option">
           <flow alignment="right">
          <button name="okbtn"/>
          <button name="cancelbtn"/>
           </flow>
          </panel>
         </component>
        </border>
      </panel>
    </frame>
  </view>
  <datamodel>
    <content view="namel" type="text">姓名：</content>
    <content view="emaill" type="text">信箱：</content>
    <content view="depl" type="text">部門：</content>
    <content view="tell" type="text">電話：</content>
    <content view="okbtn" type="text">新增</content>
    <content view="cancelbtn" type="text">取消</content>
  </datamodel>
  <events>
    <event name="ok_event" target="okbtn">
      <action>created.xml</action>
      <params>
        <param name="name_param">nametxt.getText()</param>
        <param name="email_param">emailtxt.getText()</param>
        <param name="dep_param">deptxt.getText()</param>
        <param name="tel_param">teltxt.getText()</param>
      </params>
    </event>
    <event name="canel_event" target="cancelbtn">
      <action>main.xml</action>
    </event>
  </events>
</tcml>
```

圖5-21. create.xml

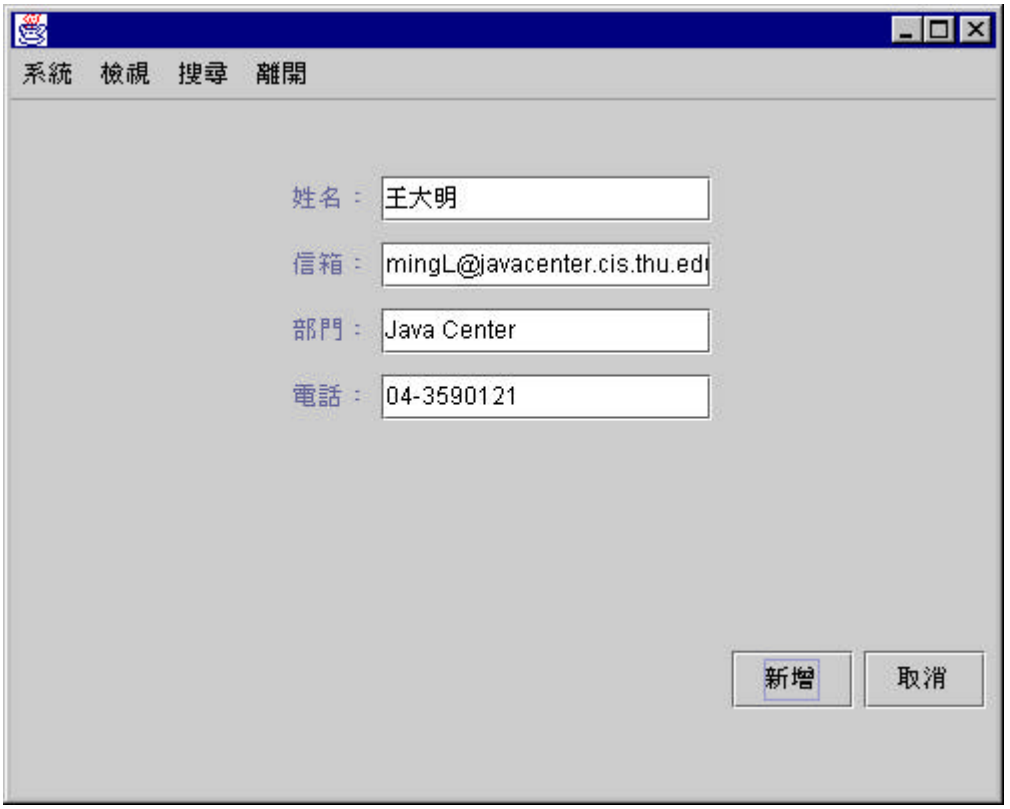

圖5-22. 填妥人事資料後的畫面

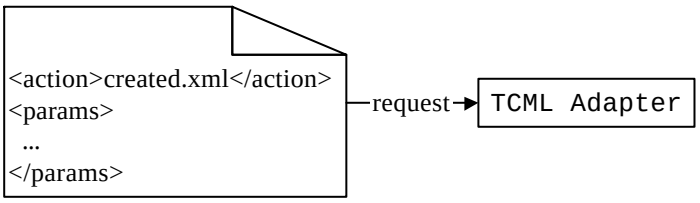

圖5-23. 觸發okbtn事件後的處理流程

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE tcml SYSTEM "tcml.dtd">
<tcml name="created_template">
  <view>
    <frame name="mainframe" width="500" height="400">
      <menu name="system"></menu>
      <menu name="view">
        <menuitem name="info"/>
      </menu>
      <menu name="search"></menu>
      <menu name="exit"></menu>
      <panel name="mainpanel">
      <border>
       <component position="center">
        <table name="infotable"/>
       </component>
       <component position="south">
        <panel name="option">
         <flow alignment="right">
          <button name="createinfo"/>
           <button name="updateinfo"/>
           <button name="delinfo"/>
           <button name="cancelbtn"/>
         </flow>
        </panel>
       </component>
      </border>
     </panel>
    </frame>
  </view>
  <datamodel>
    <content view="system" type="text">系統</content>
    <content view="view" type="text">檢視</content>
    <content view="info" type="text">人事資料</content>
    <content view="search" type="text">搜尋</content>
    <content view="exit" type="text">離開</content>
    <content view="infotable" type="dsml"><?invoke
ejb://localhost/Employee.createRowData(param.name_param,param.email_
,param.dep_param,param.tel_param)?></content>
    <content view="createinfo" type="text">新增</content>
    <content view="updateinfo" type="text">修改</content>
    <content view="delinfo" type="text">刪除</content>
    <content view="cancelbtn" type="text">取消</content>
  </datamodel>
  <events>
    <event name="info_event" target="info">
      <action>main.xml</action>
    </event>
    <event name="create_event" target="createinfo">
      <action>create.xml</action>
    </event>
    <event name="update_event" target="updateinfo">
      <action>update.xml</action>
      <params>
        <param name="row">infotable.getSelectedRow()</param>
      </params>
    </event>
    <event name="del_event" target="delinfo" dialog="本資料將被刪除！">
      <action>delete.xml</action>
      <params>
        <param>infotable.getSelectedRow()</param>
      </params>
    </event>
  </events>
</tcml>
```

圖5-24. created_template.xml

```
<datamodel>
 <content view="system" type="text">系統</content>
 <content view="view" type="text">檢視</content>
 <content view="info" type="text">人事資料</content>
 <content view="search" type="text">搜尋</content>
 <content view="exit" type="text">離開</content>
 <content view="infotable" type="dsml">
  <dsml:dsml xmlns:dsml=".">
   <dsml:table columnsize="4">
    <dsml:columnname>
      <dsml:list size="4">
       <dsml:element>姓名</dsml:element>
       <dsml:element>E-mail</dsml:element>
       <dsml:element>部門</dsml:element>
       <dsml:element>電話</dsml:element>
      </dsml:list>
    </dsml:columnname>
    <dsml:tuple>
      <dsml:list size="4">
       <dsml:element>王小明</dsml:element>
       <dsml:element>ming@javacenter.cis.thu.edu.tw</dsml:element>
       <dsml:element>CIS</dsml:element>
       <dsml:element>04-3590121</dsml:element>
      </dsml:list>
    </dsml:tuple>
    <dsml:tuple>
      <dsml:list size="4">
       <dsml:element>王大明</dsml:element>
       <dsml:element>mingL@javacenter.cis.thu.edu.tw</dsml:element>
       <dsml:element>Java Center</dsml:element>
       <dsml:element>04-3590121</dsml:element>
      </dsml:list>
    </dsml:tuple>
   </dsml:table>
  </dsml:dsml>
 </content>
 <content view="createinfo" type="text">新增</content>
 <content view="updateinfo" type="text">修改</content>
 <content view="delinfo" type="text">刪除</content>
 <content view="cancelbtn" type="text">取消</content>
</datamodel>
```

圖5-25. created.xml**中**datamodel**的內容**

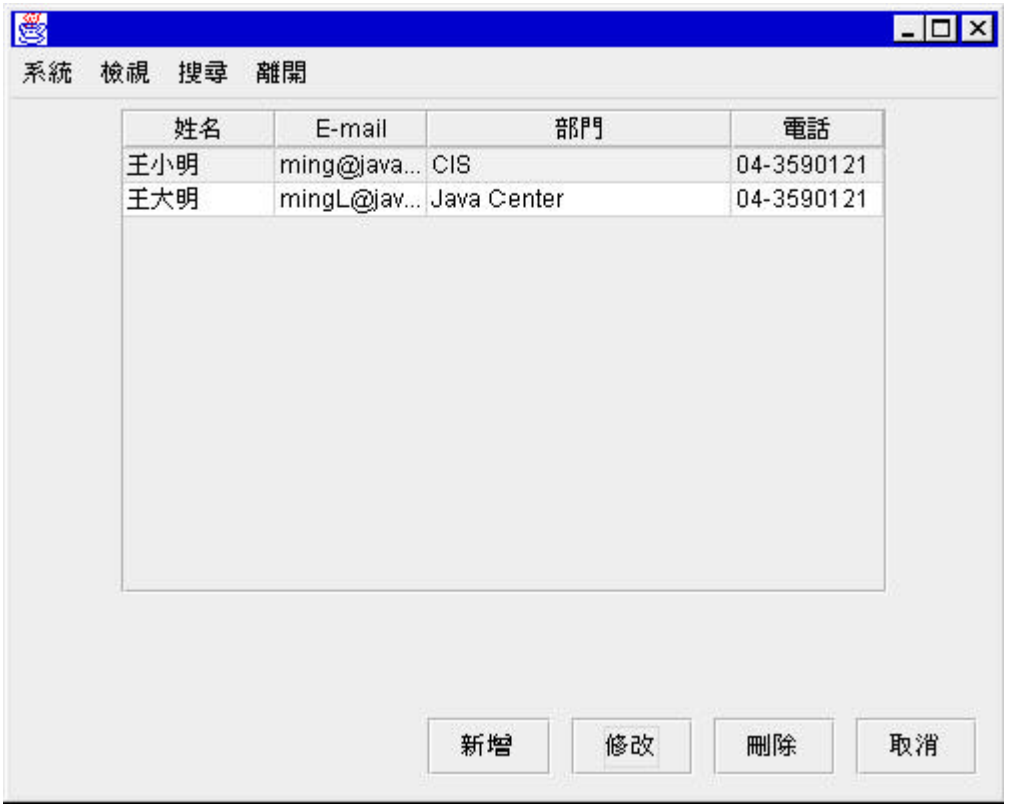

圖5-26. 新增人事資料後的畫面

- **刪除人事資料：**
  1. **當選定圖**5-27**中任一列並按下刪除按鈕時，此時則會觸發在**main.xml**中所定義的**delinfo**事件，並根據**dialog**屬性設定而跳出一個對話框。**
  2. **按下對話框中的**”OK”**按鈕後，**TCML client**才會將**target**屬性**(attribute)**之值為**delinfo**的**event element**傳至**TCML Adapter**處理。**
  3. TCML Adapter**將處理後的結果**-main.xml**傳回至**TCML client**，顯示畫面如下圖**5-28**。**

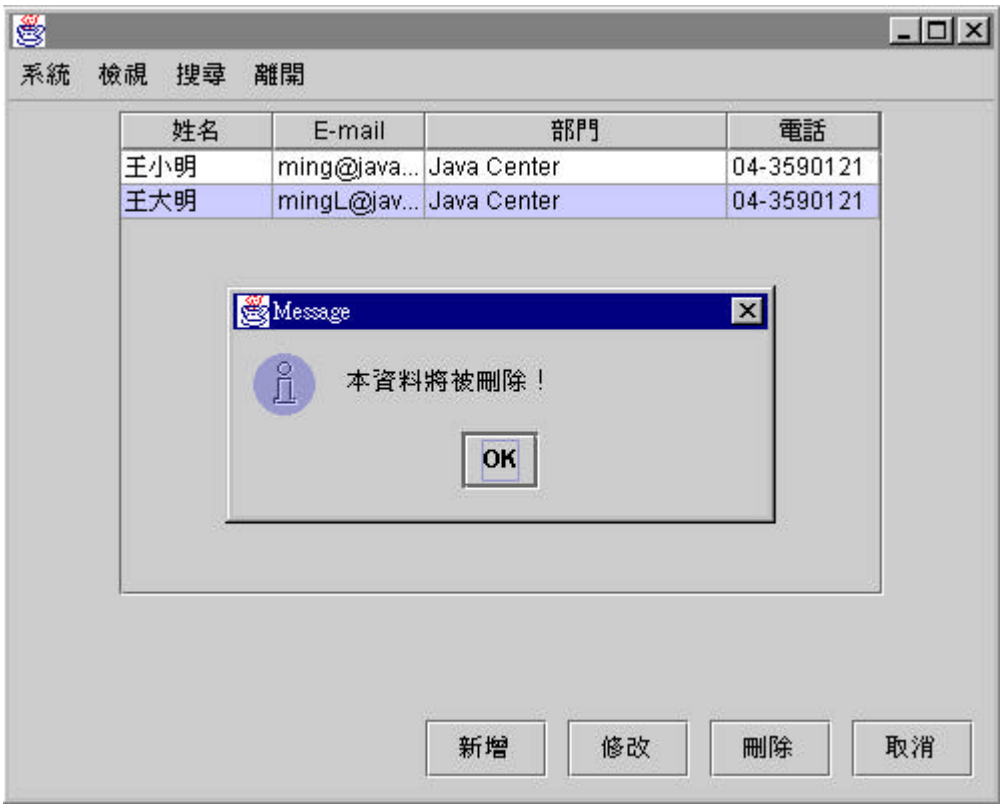

圖5-27. 刪除人事資料的畫面

圖5-28. 刪除人事資料後的畫面

第六章 結論及未來展望

6.1 結論

在這篇論文中我們利用物件導向設計中的設計樣式觀念設計出新的 thin-client MVC 設計樣式。根據此設計樣式，我們進一步利用軟體工程中軟體架構的觀念，規劃出一個新的瘦身型主從式計算架構。最後我們利過 XML 技術定義圖形使用者介面以及伺服端所對應之企業邏輯，以解決傳統以瀏覽器作為瘦身型客戶端所欠缺的介面彈性問題，並且搭配 J2EE 作為其伺服端軟體平台。此種軟體架構可以分擔圖形使用者介面的運作邏輯但又不會增加客戶端的程式大小。總結而言，本研究中之瘦身型主從式計算揉合了主從式計算、瀏覽器與遠端顯示之瘦身型計算於一身的折衷方案。本研究利用 XML 與 Java 技術設計而成，具有多項優點。除了具跨平台特性等 Java 原有優勢外，更重要的是由於資料以及應用程式是以 XML 文件形式傳輸，因此在本架構中我們可以進一步透過編、解密（encryption/decryption）技術，保障企業資訊系統應用於網際網路上時，除了資料外，應用程式也無法被窺視的安全性。

6.2 未來方向

在目前的研究中，我們比較著重在設計整體架構，有關 TCML 中圖形使用者介面的種類定義尚稱不足，因此為實際應用本研究結果於工業界，支援更進階的使用者介面是未來發展的必要方向之一。在伺服端架構方面，增加其容錯（fault tolerance）及工作平衡（load balancing）的能力，也是未來的研究主要目標。最後，架構在更多的通訊協定上，如 HTTP 及 WAP 等，將能夠使本研究成果擴展至各式客戶端。

參考文獻

[1] A. L. Batongbacal, C. Phanouriou, J. E. Shuster, M. Abrams, S. M. Williams, "UIML: an appliance-independent XML user interface language," *Computer Networks*, 1999, pp. 1695-1708.

[2] A. Comils, E. Agerbo, "How to Preserve the Benefits of Design Patterns," *Proceedings of OPSLA'98*, Vancouver, Germany, 1998.

[3] A. Hopper, K. R. Wood, Quentin Stafford-Fraser, and Tristan Richardson, "Virtual Network Computing," *IEEE Internet Computing*, January·February, 1998, pp. 33-38.

[4] Berners-Lee, T., R. Cailliau, A. Luotonen, H.F. Nielsen, and A. Secret, "The World Wide Web, " *Communications of the ACM*, 37(8), 87-96, 1994.

[5] B. Shneiderman, *Designing the User Interface*, Addison-Wesley, Reading, MA, 1998.

[6] B. Bos, H.W. Lie, C. Lilley, I. Jacobs, *Cascading Style Sheets, level 2, CSS2 Specification*, W3C Recommendation 1998, http://www.w3.org/TR/REC-CSS2/.

[7] E. R. Harold, *XML Bible*, IDG Books Worldwide, Inc., 1999.

[8] E. Schuman, "Mainframes live on, study says," *Communications Week*, 15 January 1996, p.19.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[10] F. Buschmann, H. Rohnert, M. Stal, P. Sommerlad and R. Meunier, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons, 1996.

[11] G. Blair et al. *Object-Oriented Languages, Systems, and Applications*, John & Sons Inc., New York, 1991.

[12] G. S. Hura, "Internet: State-of-the-art," *Computer Communications*, 1998, pp. 1391-1396.

[13] G. S. Hura, "The Internet: global information superhighway for the future," *Computer Communications*, 1998, pp. 1412-1430.

[14]L. Wood et al., Document Object Model (DOM) Level 1 Specification, W3C Recommendation, 1 October 1998, http://www.w3.org/TR/REC-DOM-Level-1/.

[15]Mozilla, *XUL Language Spec*, 2nd draft, 25 February 1999, http://www.mozilla.org/xpfe/languageSpec.html.

[16]R. Bernard, *The Corporate Intranet: Create and Manage an Internal Web for your Organization*, Wiley, 1996.

[17]R.A. Schultheis and D.B. Bock, "Benefits and barriers to client server computing," *Journal of Systems Management*, February 1994, pp. 12-41.

[18]R. Orfali, D. Harkey and J. Edwards, *The Essential Client/Server Survival Guide*, John Wiley, New York, 1996.

[19]S. Willcox, "Designing Thin Java Client Applications for Network Computers," *Java Report*, June, 1998, pp. 19-28.

[20]Sun Microsystems, Inc., *Java 2 Platform Enterprise Edition Specification, v1.2*, 1999, http://www.sun.com.

[21]Sun Microsystems, Inc., *Enterprise JavaBeans Specification, v1.1*, 1999, http://www.sun.com.

[22]Sun Microsystems, Inc., *Sun Ray 1 Enterprise Appliance Overview and Technical Brief*, 1999, http://www.sun.com.

[23]Sun Microsystems, Inc., *Deploying the Sun Ray Hot Desk Architecture*, 1999, http://www.sun.com.

[24]Sun Microsystems, Inc., *Sun Ray Hot Desk Architecture: A New Appliance Model Ushers In the Services-Driven Network*, 1999, http://www.sun.com.

[25]T. Minoura and V. Gundoju, "Distributed Observable/Observer: A Distributed Real-Time Object-Communication Mechanism," *Object-oriented Real-time Distributed Computing*, 1998, pp. 358-362.

[26]World-Wide Web Consortium, *An introduction to XSL*, 1998, http://www.w3c.org/Style/XSL.

[27]World-Wide Web Consortium, *XML 1.0*, February 1998, http://www.w3.org/XML.

[28]*Microsoft Windows NT 'Hydra' and Windows-Based Terminals*, 1999, http://microsoft.com/ntserver/guide/hydrapapers.asp.