行政院所屬各機關因公出國報告書

（出國類別：實習）

實習「J2EE 電腦技術之應用與實務」

服務機關：中華電信南區分公司
出國人　職　　稱：助理管理師(五)
　　姓　　名：黃榮璋

出國地點：美　國
出國期間：92 年 9 月 14 日至 92 年 9 月 27 日止
報告日期：92 年 12 月 8 日

系統識別號:C09203873

公 務 出 國 報 告 提 要

頁數: 47　含附件: 否

報告名稱:

J2EE電腦技術之應用與實務

主辦機關:

中華電信台灣南區電信分公司

聯絡人／電話:

李文志／07-3443121

出國人員:

黃榮璋　中華電信台灣南區電信分公司　資訊處　助理管理師

出國類別: 實習

出國地區: 美國

出國期間: 民國 92 年 09 月 14 日 -民國 92 年 09 月 27 日

報告日期: 民國 92 年 12 月 08 日

分類號/目: H6／電信　H6／電信

關鍵詞:　J2EE

內容摘要: J2EE是美國Sun公司所推出的企業級網路應用程式平台，是一種分散式多層式應用程式架構。它可完全支援EJB，具有良好的封裝與部署能力，並增加對目錄服務的支援、強化安全機制、提高執行效能，成爲目前國際級企業廣泛使用的應用平台。本實習報告主要先簡介J2EE之特性與優點及相關技術，接著介紹系統架構與架構的設計樣式、元件與容器，之後介紹J2EE的技術核心--EJB，及JavaServlet、JSP等、應用程式設計與部署、應用伺服器以及系統安全、整合與進階開發部份。

本文電子檔已上傳至出國報告資訊網

目　錄

壹、目　的

我們今日處在一個資訊的交流和貨品或服務的交流一樣重要的網路經濟環境中，在激烈競爭下，即使是傳統產業式的企業也發現他們必須發展新的技術，以管理、傳送和利用他們的資訊資源，公司需要透過網路來開發新客戶以及和廠商進行更密切的互動。

網路經濟主要即由 Internet 所推動，另外還有行動電話和 PDA 的無線網路、企業的 Intranet、區域網路、廣域網路等各種網路 。

在以往，資訊技術的重點是資料管理，企業藉著大型資料庫管理系統讓組織能夠搜集、分析並解釋資料，以便擬定公司的經營策略；而在網路經濟中，資訊技術的重點已然轉移到應用程式(Application)上：分散式電腦應用程式是重複使用現有資料和取得新資料的關鍵；應用程式則是和客戶、廠商以及企業夥伴建立安全且穩固連結的關鍵。因此，要有效的加強競爭力，在艱辛的環境中有更多機會存活茁壯，最重要就是擁有快速而有效的發展和佈署新應用程式的能力。

J2EE(Java 2 Platform，Enterprise Edition) 是美國 Sun 公司所推出的企業級網路應用程式平台，目的就是希望提供一個低成本、高可用性、高可靠性以及高可延展性的網路應用程式平台，讓企業在網路經濟時代取得優勢。透過這個統一的開發平台，J2EE 降低了開發多層式網路應用程式時所需之費用以及複雜性，同時也對企業內部現有的應用程式提供良好的支援，它可完全支援 EJB(Enterprise JavaBean)，具有良好的封裝與部署能力，並增加對目錄服務的支援、強化安全機制、提高執行效能，成爲目前國際級企業廣泛使用的應用平台。要開發 J2EE 應用程式，不僅必須瞭解多種 Java 程式技術，例如：EJB、JSP、Servlet...等等，還須具備系統架構的能力，包括：Architecture、UML、Design Pattern、OOAD 等等，其中任何一項技術都相當複雜。

本報告主要分三部分：

1. J2EE 簡介與系統架構

2. J2EE 程式設計與部署

3. 系統整合與進階開發

目前總公司正擬針對種類繁多的內部資訊系統進行整合工作，希望能讓資訊系統具備更有競爭力的新貌，本報告特就 J2EE 作簡單介紹，謹供參考。

貳、 過 程

本實習之行程如下：

9 月 14 日	搭長榮 BR32 前往紐華客
9 月 15 至 9 月 19 日	實習 J2EE 簡介
9 月 22 至 9 月 25 日	實習 Enterprise JavaBeans(EJB)架構
9 月 26/27 日	搭長榮 BR15 回國

參、 心 得

一、J2EE 簡介與系統架構

- J2EE 之優點：

J2EE 平台建立在 Java 程式語言與 Java 技術上，是一個最適用於企業分散式作業環境的應用程式架構。它是一套能爲資訊產業、應用程式開發者以及產品廠商帶來下列好處的標準。

1. 廠商所開發的產品，不須花費任何多餘代價即可以執行於任何支援 J2EE 標準的系統下。
2. 公司的資訊人員可以從可移植的元件技術下得以好處，應用程式可不受限於特定廠商的環境控制。
3. 公司的資訊人員能專注於滿足主要商業邏輯的各種需求，不必分心於建立提供基礎服務的架構諸如：多執行緒、同步作業、資源管理、交易管理、生命週期管理等。
4. 採用 J2EE 標準可使 Java 開發人員快速學到 J2EE API，大幅提昇其生產力。
5. 採用 J2EE 標準可使公司得以保障其投入成本，因爲 J2EE 爲業界公開支援的標準，非由某家廠商單獨掌握的架構。

6. 公司研發團隊可以更快地建立新的應用程式與系統，減少投入市場運轉的時間與成本。
7. 可確保應用程式可靠地在經過認證的平台上執行。
8. J2EE 平台提供可靠的應用程式，也劃分爲許多層以滿足多執行緒應用程式的需求。
9. 開發人員不只能夠建立自己的 J2EE 元件，還可以自廠商元件市場中取得。

● J2EE 之特性：

在今天，許多企業都需要擴展其業務範圍、降低經營成本、縮短回應時間，因此需要一些簡單又快速的服務，而且這些服務必須能夠與現有的**企業資訊系統**(Enterprise Information System，EIS)相結合，以提供新的產品或服務，而這些服務必需具備下列特點：

1. 可用性：以滿足全球化和全天候的商業環境需求
2. 安全性：以保護用戶隱私和企業資料的安全
3. 可依賴性：以確保商業交易的正確和迅速

這些服務通常是由分散式應用程式所組成的，包括前端資料端和後端資料來源以及它們之間的**中間層元件**(Component)，這些中間層元件提供了主要的商業功能和資料與 EIS 系統結合的能力，把用戶端從複雜的商業邏輯中分離出來，利用成熟的網路技術讓企業在管理上所花費的時間最少化，降低開發成本和複雜程度，使產品與服務可以快速推陳出新，從容面對企業競爭的壓力。 J2EE 就是一種利用 Java 語言的標準架構，讓使用它的企業可以在應用程式的中間層加速**分散式**部署，在開發過程中利用這種架構，讓開發人員可以集中精力在關鍵商業邏輯的設計和應用程式的表現上。

● J2EE 相關技術

1. Java2 Standard Edition(J2SE)

2.Enterprise JavaBean (EJB)

EJB 提供一個框架來開發和實施分散式商業邏輯，明顯地簡化具有可延伸性和高度複雜的企業級應用程式的開發；EJB 規範定義了 EJB 元件在何時如何與它們的容器(Container)進行交互作用，而容器則負責提供公用的服務，例如：目錄服務、交易管理、安全性、資料庫連線緩衝以及容錯性。

3.JavaServer Pages (JSP)

JSP 頁面由 HTML 原始碼和嵌入其中的 Java 程式碼所組成，它被設計來協助 Web 內容開發人員建立動態網頁，而且只需很少的程式碼。伺服器在頁面被用戶端請求後對這些 Java 程式碼進行處理，然後將產生的 HTML 頁面傳回給用戶端的瀏覽器。

4.Java Servlet：

Servlet 所提供的功能大多與 JSP 類似，不過表現的方式不同。JSP 通常是在 HTML 原始碼中嵌入少量的 Java 程式碼，而 Servlet 則全部由 Java 寫成並且產生 HTML 頁面，Servlet 是一種小型的 Java 程式，它延伸了 Web 伺服器的功能，作爲一種伺服器端的應用程式，當被請求時則開始執行。

5.Java Database Connectivity (JDBC)：

許多 Java 程式與元件都需要存取到某個資料庫內的資料，因此，Java 開發團隊發明了 JDBC API，讓程式能夠與任何商業的 DBMS 作連結以及資料存取異動。

6.Java Interface Definition Language(JavaIDL)：

讓 Java 平台與 Common Object Request Broker Architecture(CORBA)合作以使用 Object Management Group(OMG)所定義的標準介面定義語言(Interface Definition Language, IDL)的服務，在 JavaIDL 的支援下，開發人員可以將 Java

與 CORBA 整合在一起。

7.Remote Method Invocation-Internet Inter-ORB Protocol (RMI-IIOP)：

讓使用 RMI-API 和 CORBA IIOP 通訊協定與使用任何程式語言所開發的 CORBA 用戶端溝通的協定。

8.JavaMessage Server (JMS)：

JMS API 是在 Java 程式中建立元件之間用來傳送的連結管道，而這個連結具有容錯的功能，並且允許以非同步的模式來傳送與接收。

9.Java Naming and Directory Interface (JNDI)：

物件能夠放置於伺服器上的任何位置，而此伺服器與公司的基礎架構連結，爲此 Java 開發團隊需要一個方法來讓 Java 程式能夠很容易地放置這些物件。其解決方式就是建立標準化的命名規則與目錄以及 JNDI API，如此就能讓程式設計者從他們的 Java 程式內查詢物件。

10.Java Transaction API (JTA)：

一系列關於交易(Transaction)管理的 API，應用程式可以透過 JTA API 以建立元件的處理流程來管理這些交易。

11.JavaMail：

用於存取郵件伺服器的 API，它能夠讓顧客與電子商務網站作有效率的資訊交換與溝通。

12.XML 部署描述檔：

許多企業已經開始採用 XML 來存取、處理。與交換文件內顯示出來的文字資訊。Java 開發團隊已經將這些描述符號包含在 J2EE 中，以便讓程式設計者能夠建立和 XML 文件互動的工具與元件。當元件被部署到 J2EE 的容器時，XML 部署描述檔就定義了元件的環境與功能。J2EE 的容器會藉由讀取部署描述檔來知道如何部署元件，以及該部署到何處，只是元件本身並不需要與部署描述檔作互動。以下是一些常見的 XML 部署描述檔：

1. 管理容器與 Enterprise Java Bean 之間的交易。

2. 註冊一個 Message-Driven Bean 到佇列(Queue)中。

3. 定義 JNDI 的查詢命名方式。

4. 管理 Stateful 與 Stateless Session Bean。

● J2EE 軟體架構(Architecture)

在傳統軟體設計中的軟體架構，主要分爲模組(Module)的架構、模組間的依存性、作業分工等；而在現在軟體設計中的軟體架構中，尙須考慮網路架構、集線器、伺服器、終端機的配置，應用程式的分配執行、相互溝通、交換資料等，即須滿足系統的「**非功能性**」需求。

非功能性需求可分爲下列數項：

1. 延展性(Scalability)

   當系統負載增加時，仍需維持原有服務品質。

2. 可維護性(Maintainability)

   在現有功能上作修改而不影響其它元件或系統功能。

3. 可靠性(Reliability)

   應用程式與其執行之交易及所提供服務的正確性與一致性

4. 可用性(Availability)

   確保資源或服務可以長久使用。

5. 可延伸性(Extensibility)

   修改或新增功能而不影響現有的功能。

6. 容錯能力(Fault tolerance)

   能處理系統錯誤並回復正常的能力。

對企業級應用系統而言，軟體功能之完善與否固然重要，但系統架構更形重要，其重要性猶如一棟大廈之建築師；前者以微觀的角度來看軟體系統，後者則以宏觀的角度切入系統架構，系統架構設計師的領域即涵括：

1. 需求分析
2. 硬體與軟體架構
3. 系統分析、設計與開發
4. 產品支援
5. 效能調校、安全規劃、網路規劃..等

網路經濟中的應用程式通常是多層式的、以伺服器爲基礎的應用程式，並且能夠和各種系統互動。這些應用程式是分散式的(Distributed)，也就是分在一些不同的裝置上執行。

分散式應用程式架構的演進：

1. 純粹的主從式應用程式架構：
   分爲 Client Tier(終端機)與 Server Tier(大型主機)
2. 以 PC 爲主的主從式應用程式架構：
   分爲 PC Client Tier(個人電腦)與 Server/Data Repository Tier(大型主機)
3. 使用分散式交易的多層式應用程式架構：
   分爲 PC Client Tier(個人電腦)、Middeleware/Transaction Processing Tier(中介軟體)與 Database Tier(資料庫)
4. 利用多伺服器與 CORBA 互通性建構多層式應用程式架構：
   分爲 PC Client Tier(個人電腦)、Web Server Tier(網路伺服器)、Business Logic Tier(商業邏輯)與 Distributed Database Tier(分散式資料庫)

J2EE 是一個四層式(Four-Tier)架構。由用戶端層(也稱展示層或應用程式層)、Web 層、Enterprise JavaBean 層(也稱商業邏輯層)、以及企業資訊系統(Enterprise Information System, EIS)層。每一層的任務在於提供特定型態的功能給應用程式使用。用戶端是向元件要求服務的程式；資源是元件所提供

的任何服務；元件是層級的一部分，是由大量類別或執行函式以提供服務的程式所組成；容器是管理元件以及提供元件系統服務的軟體, 元件與容器之

1. 用戶端層級由使用者互動程式所組成。這些程式會提示使用者輸入資料，再將這些資料轉換成請求(Request)傳送給元件中的軟體處理，然後將處理後的結果回傳給用戶端程式。
2. Web 層級提供 J2EE 應用程式 Internet 的功能。在 Web 層級運作的元件使用 HTTP 接收請求與傳送回應資訊給位於任意層級的用戶端，用戶端可以是任何發出請求的元件。
3. Enterprise JavaBeans 層包含 J2EE 應用程式所需的商業邏輯操作。此層級中擁有一或多個 Enterprise JavaBean，每個 EJB 內含商業邏輯操作，再由用戶端來呼叫。**EJB 層可說是 J2EE 應用程式的基礎**，在此層運作的 EJB 允許多個應用程式實體同時存取商業邏輯與資料，並不會妨礙到效能的表現。而儘管 EJB 可以存取任何一層的元件，不過，一般而言 EJB 會存取在企業資訊系統(EIS)層的元件與資源如 DBMS。總之，此層與 Web 層構成了 J2EE 平台中最重要的中間層。
4. EIS 使得 J2EE 應用程式能對企業網路上可以取得的資源與舊式系統作連結。J2EE 應用程式以 EIS 爲介面直接或間接使用各式各樣技術資源，包括 DBMS 與維持企業運作、扮演關鍵任務系統的大型電腦。

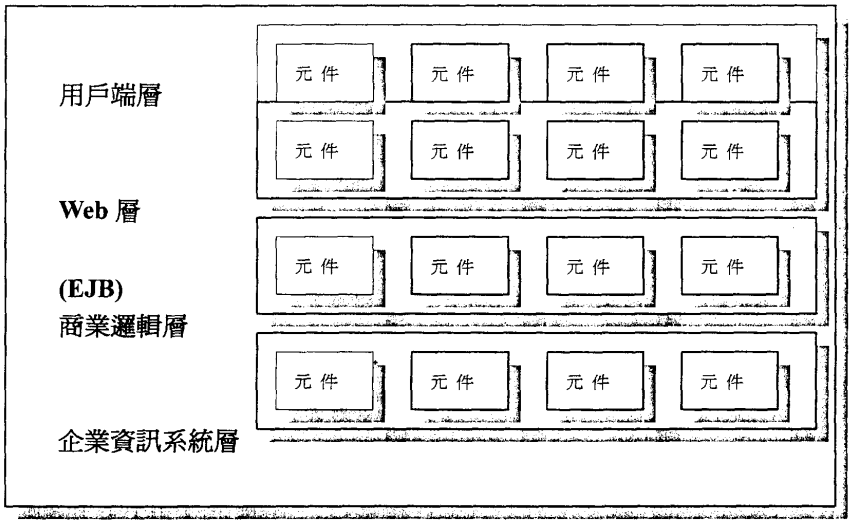

圖 1. J2EE 由四個層級組成，每一層皆具備特定的功能供應用程式使用

● J2EE 設計樣式

樣式(Pattern)是一項經過證實認可的技術，是以前一些專家解決問題後所留下來的經驗，經過整理後成爲可用來協助初入門者快速入門的捷徑。在設計與建立 J2EE 應用程式時，加入適當的樣式，可以獲得專業 J2EE 開發者的經驗，以避免常見的錯誤又發生。總之，樣式是一個已經解決某種問題的解法與技巧，未來可以利用此樣式來解決相同的問題。

● 樣式目錄

樣式通常被聚集成一個稱爲樣式目錄（Pattern Catalog）的群集，以供設計 J2EE 應用程式時參考用。以下簡述一些既有之樣式。

1. Handle-Forward 樣式

用來建立「處理」或「轉送」的模型機制，透過它，將請求從某一

個元件傳送到另一個元件，直到抵達可處理此請求的元件爲止。

2. Translator 樣式

提供一種方式讓應用程式能夠將某種型式的訊息轉換成另一種型式。訊息可以是服務的請求，其中包含處理此請求所需要的資訊。而請求的型式也許和處理此請求的元件所要求之型式不同，本樣式將會適當地重新編組這些訊息，將請求重新傳送到合適的元件，以便進行處理。

3. Distributor 樣式

管理一或多個應用程式所使用元件之間的通訊。本樣式接收到從某個應用程式或另一個元件所送來的訊息，再根據訊息的內容，將訊息轉送到適當的元件。其角色類似於網路上的路由器(Router)。

4. Broadcaster 樣式

從一個以上的應用程式或元件接收訊息，再將這些訊息分派到其它對這些訊息感興趣的元件。類似於 Translator 與 Distributor 樣式，它們都是先接收訊息，再將訊息轉送給其他元件，不相同的是：在 Broadcaster 樣式，元件若要能接收到轉送訊息，必須先向 Broadcast 元件提出申請。

5. Zero Sum 樣式

將一群獨立的程序當成單一的處理單元(Processing Unit)，只有處理單元中所有的程序都成功地終止，此單元才算成功；如果有一個程序發生錯誤，則代表此單元失敗。

6. Status Flag 樣式

指出一個狀態，其值決定了一或多個獨立元件所進行的行爲。

7. Sequencer 樣式

連續地從應用程式或元件中存取獨立的元件。Sequencer 元件觸發一連串的程序，每個程序的回應將會被回傳到 Sequencer，其再依

序呼叫下一個程序；一或多個程序的失敗並不會影響請求者。

8. Consolidator 樣式

可以用它簡易地存取散佈在多個資源上的資料。需要資料的應用程式或元件將傳送一個請求給 Consolidator 元件，此元件中包含了如何從多個資源擷取資料並將它們合併在一起的邏輯運算，並將合併後的資料回傳給請求者。

9. Simplicity 樣式

簡化複雜處理常式的實作過程，其要求需要呼叫這些常式的應用程式或元件傳送一個簡單的請求給某個元件，該元件將此簡單的請求轉換成可以滿足請求的適當元件呼叫。

● J2EE 資料庫

J2EE 應用程式是藉由元件的組合所建立，每個元件都會提供一個網路服務給 J2EE 應用程式，而其中一個元件可以存取一個或更多的資料庫。資料庫是資料的群集，由資料庫管理系統(DBMS)來管理。提供資料庫存取的元件，使用了 Java Database Connection(JDBC)規格書中所定義的 Java 資料物件，並且結合 SQL，使得 J2EE 應用程式能夠存取商業上的 DBMS。當使用 SQL 語言來建構 DBMS 連線時，Java 資料物件會與 DBMS 形成一個通訊連結，進行資料的查詢、更正、刪除、以及其他在 DBMS 上的操作。

● JDBC 概念

Sun 公司於 1996 年建立了 JDBC 驅動程式與 JDBC 應用程式介面(API)，使得 Java 程式設計師可以使用 JDBC API 所定義的高階 Java 資料物件，撰寫與 DBMS 互動的程序。Java 資料物件遵循 JDBC 驅動程式規格書，將程序轉換成低階的訊息並將之傳送給 JDBC 驅動程式。JDBC 驅動程式將程序翻譯成 DBMS 看得懂的低階訊息。JDBC 驅動程式讓 J2EE 元件資料庫成獨

立狀態，這讓 Java 平台獨立的哲學更加強化。今日幾乎所有的商業 DBMS 都有 JDBC 驅動程式。

● JDBC 處理行程

1.載入 JDBC 驅動程式

2.連線到 DBMS

3.建立並執行 Statement 物件

4.處理由 DBMS 回傳的資料

5.終止與 DBMS 的連線

J2EE 元件並未直接與 DBMS 連線，而是連線到與 DBMS 聯繫的 JDBC 驅動程式。在建立連線前，JDBC 驅動程式必須先載入，並且利用 DriverManager 來註冊。

● Statement 物件

一但開啓資料庫連線，J2EE 元件會建立並傳送一個查詢(Statement)來存取資料庫內的資料。Statement 物件可分爲三種類型：

1.Statement 物件

每當 J2EE 元件需要直接執行查詢，而毋須事先編譯過，則會使用 Statement 物件。Statement 物件包含 executeQuery()方法，此方法將查詢作爲參數，接著查詢會傳送給 DBMS 處理。executeQuery()方法會回傳 ResultSet 物件，此物件包含了查詢所要求的資料以及中繼資料如欄位名稱、欄位大小以及欄的資料型態。

2.PreparedStatement 物件

在 DBMS 處理查詢之前，必須先編譯 SQL 查詢，在呼叫 Statement 物件的 execute()方法後就會開始進行編譯動作。編譯查詢是一項負擔，

而藉由使用 PreparedStatement 物件，我們可以預先編譯與執行 SQL 查詢，並且可以依!照需要呼叫 setxxx()方法來改變查詢中特定的值，而毋需重新編譯查詢。

3.CallableStatement 物件

用來從 J2EE 物件裏呼叫預存的程序(Stored Procedures)。預存程序是一個程式碼區塊，由唯一的名稱來辨識，藉由呼叫預存程序的名稱，我們可以執行此預存程序。當呼叫預存程序時，物件使用三種參數型態：IN、OUT、INOUT。其中 IN 參數包含要傳入預存程序的任何資料，OUT 參數包含預存程序所回傳的資料，INOUT 參數可以用來傳送資訊給預存程序，也可以從預存程序中接收資訊。

二、J2EE 程式設計與部署

● EJB(Enterprise JavaBean)

JavaBean 是一種以 Java 語言開發的元件，提供一個標準的開發環境，主要是用來定義事件與屬性，讓所有符合 JavaBean 規範的物件都可以在不同的 IDE 開發工具中使用。EJB 是在企業伺服器端執行的 JavaBean，其目的是用來發展「企業級物件」，而企業級物件最重要的功能則是交易與分散式處理，此外還有安全性、一致性、延展性、多用戶端.....等特性。EJB 必須在 EJB 容器(Container)的環境中執行，EJB 容器會自動管理 EJB 的安全、交易與一致性以保障伺服器的安全性，提高資源的有效運用，並提供 EJB 之間的溝通管道。

Sun 對 EJB 的定義：「EJB 是用於開發和部署多層式、分散式、物件導向 Java 應用系統的跨平台架構」，使用 EJB 可以使商業應用系統的開發變得更容易，應用系統可以在任何支援 EJB 的環境下開發，開發完成後再部署在其他的環境中，而且隨著需求的改變，應用系統也可以在不須修改的前提下轉移到其他功能更強、更複雜的伺服器上執行。

*客戶端如何與 EJB 互動

位於 EJB 環境中央的 EJB 本身包含一些爲客戶端提供服務所需要的處理邏輯。EJB 被 EJB 容器包著，EJB 容器將處理 EJB 的生命週期需求，EJB 容器利用主介面(Home Interface)和遠端介面(Remote Interface)來處理 EJB 與 EJB 環境中其他元件之間的訊息溝通。

客戶端利用主介面和遠端介面，能夠同時以遠端方式(位於 EJB 所在容器之外)和本機端的方式存取相同容器中的 EJB。如圖 2。

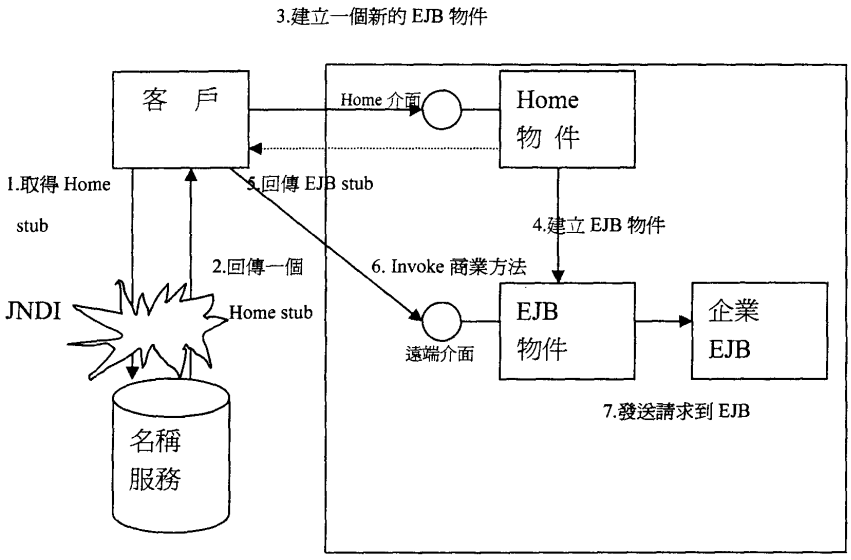

圖 2. 用戶端呼叫 EJB 之流程

✱ 開發人士的角色分配

EJB 程式設計的過程可以分爲六種不同角色，以簡化複雜的應用程式：

1.EJB 元件提供者(EJB Component Provider)

負責定義 EJB 的遠端和主介面、撰寫執行商業邏輯的 EJB 類別、提供部署 EJB 的部署描述檔(Deployment Descriptor)，所開發出的 EJB 元件會被封裝成一個 JAR 檔。

2.應用程式組合者(Application Assembled)

負責利用各種 EJB 元件組成一個完整的系統。

3.部署者(Deployer)

負責將 JAR 檔部署到用戶端的系統環境中，這個系統環境包含 EJB 伺服器和 EJB 容器，部署者必須保證所有由 EJB 元件開發者在部署描述檔中宣告的資源都可以使用。

4.EJB 伺服器提供者(EJB Server Provider)

通常這是系統領域的專家，精通分散式交易管理、分散式物件管理及其他系統層級的服務，一般由作業系統、中間元件或資料庫的供應商所提供。

5.EJB 容器提供者(EJB Container Provider)

主要提供一組標準、簡單的 API 以存取 EJB 容器，讓 EJB 元件開發者不需要瞭解 EJB 伺服器中的各種技術細節;同時也提供系統監測工具用來監測 EJB 容器和執行中的 EJB 元件狀態。

6.系統管理員(System Administrator)

主要負責爲 EJB 伺服器和容器提供一個企業級的運算與網路環境，並利用 EJB 伺服器和容器提供的監測管理工具來監測 EJB 元件的執行狀況。

✱ EJB 的組成

EJB 是以分散式處理爲基礎的企業級應用程式元件，其組成包含：

1. 遠端介面

延伸自 javax.ejb.EJBObject，爲用戶端使用 EJB 時所呼叫的介面,主要用來定義 Bean 的商業邏輯函式，以 Proxy 樣式運作。

2. 主介面

延伸自 javax.ejb.EJBHome，以 Factory 樣式建立 EJB 的實例，用戶端可以透過主介面取得 EJB 物件，主要用來管理 Bean 的生命週期以及查詢 Bean 的屬性。

3. Bean 類別

存在 EJB 伺服器上，包含遠端介面中所定義的函式實作，負責實際處理用戶端要求的類別，是真正執行商業邏輯的地方。延伸自 javax.ejb.Session 或 javax.ejb.Entity。

4. Primary Key

5. 部署描述檔(Deployment Descriptor)

通常是由開發工具或 EJB 伺服器所產生，負責 EJB 與伺服器之間的溝通，包含交易控制、安全管理與 Bean 的資料永續(Persistence)。

6. 環境特性

也就是 EJB 的屬性，負責提供 EJB 特性的改變。

7. JAR 檔

負責將 EJB 封裝起來，一個 EJB 的 JAR 檔中可以部署一個或多個 EJB 元件，其內容有必要的 Java 類別檔。

✱ EJB 的架構

EJB 上層的分散式應用程式是以物件元件模型爲基礎，底層的是件服務則使用 API 技術，它簡化了使用 Java 語言來撰寫企業應用系統時的開發、配置與執行過程，以下是一個典型的 EJB 架構。

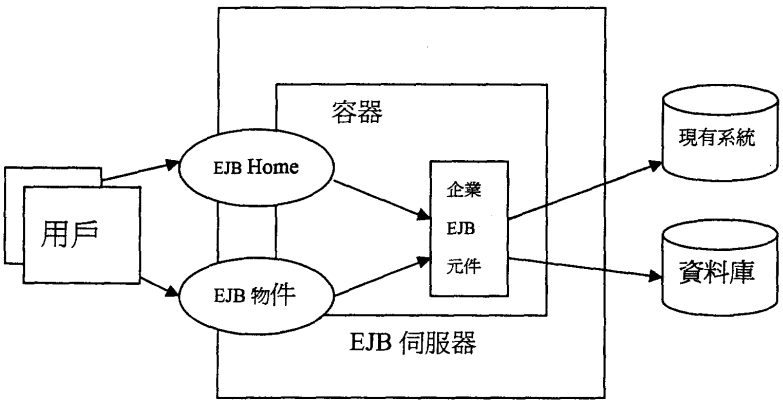

圖 3　EJB 架構

EJB 容器是一個管理一或多個 EJB 類別/實例的抽象層，透過定義的介

面使 EJB 類別存取所需的類別服務，而 EJB 容器供應商通常也會在容器或伺服器中提供額外的服務介面諸如：交易管理(Transaction Management)、安全管理(Security Management)、遠端用戶連接(Remote Client Connectivity)、生命週期管理(Life Cycle Management)與資料庫連線緩衝區(Database Connection Pooling)等。

用戶端程式要與 EJB 溝通，必須透過主介面建立 EJBobject，並獲得遠端介面以呼叫該物件所提供之各種函式，待 EJB 服務完成後，再透過主介面來移除此物件。

✱ EJB 的限制

1. 無法使用靜態函式
2. 沒有執行緒控制
3. 無法直接存取系統 I/O

✱ EJB 的命名原則

語　法	項　目	範　例
<*name*>EJB	EJB 名稱	BuyingEJB
<*name*>JAR	EJBJAR 名稱	BuyingJAR
<*name*>Bean	EJB 類別	BuyingBean
<*name*>Home	主介面	BuyingHome
<*name*>	遠端介面	Buying

Local<*name*>Home	Local 主介面	LocalBuyingHome
Local<*name*>	Local 介面	LocalBuying
<*name*>	Abstract schema	Buying

✱ EJB 的種類

從性質上來看，EJB 可分爲三類:

1. Entity Beans 是以元件方式代表儲存於資料庫的資料，可被多個用戶端使用，以節省系統資源。

2. Session Beans 用來主導一個工作流程，會與其他 Beans 進行互動，以完成用戶端所需的服務。

3. Message Driven Bean 是在新的 EJB 2.0 的規格中才有的一種新型 EJB。

類　　型		適　用　工　作　範　圍
Entity Beans	CMP	較簡單的資料庫存取
	BMP	較複雜的資料庫存取
Session Beans	Stateless	無狀態，工作時間較短的工作流程控制元件
	Stateful	有狀態，工作時間較長的工作流程控制元件
Message Driven Bean	Java Message Service API 用來處理非同步訊息的 Listener 元件	

✱ Entity Bean:

Entity Bean 可以用來代表底層的物件，簡單的 Entity Bean 可以定義代表資料表的記錄，亦即：每一個實例都代表一個特殊的記錄；而複雜的 Entity

Bean 更可以代表資料間的關係圖。Entity Beans 的生命週期較長，而且狀態是持續性的，只要資料庫中的資料存在，Entity Beans 就會一直存在，不會因爲應用程式離開而消失；此外，多個用戶端能夠同時存取同一個 Entity Bean，是所有 EJB 中最耗系統資源的。

Entity Bean 又可分爲 BMP 與 CMP；此兩種 Entity Bean 的型態不同，但目的都是用於提供資料，其主要的差別在於維護資料的角色：

BMP (Bean-Managed Persistence)是由 Bean 自行維護資料的一致性，而

CMP (Container-Managed Persistence)則是由 EJB 容器來維護資料的一致性。

✱ Session Bean:

Session Bean 的生命週期相對於 Entity Bean 較短，只有當用戶端保持 Session 時，Session Bean 才會存在，一旦用戶端離開，Session Bean 就不再與用戶端連結。Session Bean 主要的目地是讓程式開發者將商業邏輯層獨立，複雜的商業邏輯可以放在 Session Bean 中。

Session Bean 又可分爲 Stateful Session Bean 與 Stateless Session Bean 兩種；Stateful Session Bean 可以記錄使用者的狀態，一般當使用者呼叫某個 Stateful Session Bean 的函式時，EJB 容器會清楚的知道哪一個 EJB 實體是屬於哪一個使用者的，Stateless Session Bean 則不會記錄使用者的狀態，亦即：當使用者呼叫 Stateless Session Bean 時，EJB 容器並不會去尋找特定的 Stateless Session Bean 實體以執行函式。因此，在一般設計上，不會讓兩個使用者同時使用同一個 Stateful Session Bean，而當數個使用者同時使用同一個 Stateless Session Bean 時，其實是使用同一個實體。

✱ Message Driven Bean

主要是回應 Message Queue 中的事件，亦即：當有訊息傳入 Message Queue 時，Message Driven Bean 就會被主動觸發，作出相對的回應。

✱ 交易(Transaction)

交易必須遵循 ACID 原則：即 Atomic、Consistent、Isolated 與 Durable。Atomic 表示交易不能「部份完成」，若非全部完成，就是完全回復到起始狀態，這是交易的「不可分割性」；Consistent 指的是資料必須保持前後一致的狀態，不會因交易失敗而產生前後不一致的情況，User 存取資料時才能確保其準確性，這是交易的「一貫性」；Isolated 是指各次交易間的互相獨立性，交易的完成不會依賴其它我們無法控制的交易，這是交易的「隔離性」；Durable 則指明交易應能排除系統的錯誤，萬一系統在交易途中出錯，則交易必須依照 Atomic 與 Consistent 的原則退出來，這是交易的「耐久性」。

交易由 Session Bean 或 EJB 容器來管理。Session Bean 使用 JTA(Java Transaction API)的 javax.transaction.UserTransaction 類別來管理交易，包括啓動交易與確認(Commit)或回復(Rollback)；而 EJB 容器的交易管理則可透過 EJBContext.setRollbackOnly 來控制，交易會因系統例外而自動回復，但不包含應用程式的例外。

✱EJB 之效能調校方式：

a. 使用 Local 界面

b. 利用交易控制 Entity Bean

c. 減少 JNDI 搜尋

d. 使用 CMP 代替 BMP

e. 選擇適當的 EJB 種類

f. 擴充系統架構

g. 調整系統參數

h. 模擬環境測試

i. 利用工具檢查效能的瓶頸

✱在 J2EE 應用程式中使用 EJB

a. 建立 J2EE 應用程式

b. 撰寫程式碼

1. 建立遠端介面

2. 建立主介面

3. 建立 Bean 類別

4. 程式碼編譯

c. 封裝 EJB

●Java Servlet 與 JSP

前面提到元件(Component)，在 J2EE，每個元件提供應用程式一種特定的服務；例如使用者介面(User Interface)，是最高階且最顯而易見的元件，它接收 J2EE 應用程式使用者的服務請求(Request)，並將此請求轉給另一個元件來處理。處理服務請求的元件是用 Java Servlet 或 JavaServer Page(JSP)撰寫而成。Servlet 由使用者介面或另一個 J2EE 元件所呼叫，乃屬於伺服端之程式，其中包含一些用來處理服務請求的商業邏輯。JSP 同樣是伺服端之程式，所執行的功能與 Servlet 類似，只是所用之技術不同。兩者都是呼叫其他的元件來協助處理細節部分。

✱ Servlet 優於 CGI

比起 CGI(Common Gateway Interface)，Servlet 技術要更有效率且更節省環境資源的使用。因爲：

a.不論同時有多少服務請求，只有一份 Servlet 會載入 JVM 中。

b.每次遇有服務請求，會產生一個執行緒(Thread)而非一新的處理程序，因此節省了伺服器上記憶體的使用。

c.由於記憶體裏只有一份 Servlet，反應時間也因此變快。

d.當服務請求已完成，即使有新的請求進來，Servlet 所使用過的資料也能保留下來，此即它的永續性(Persistence)。

✱ Java Servlet 簡介

Java Servlet 是個 Java 類別，其功能爲讀取客戶端送來的服務請求，並將回應資訊回傳給客戶端。此 Java 類別必須繼承自 Httpservlet 類別，並覆寫 Httpservlet 的 doGet()和 doPost()方法。doGet()和 doPost()兩個方法都需要兩個引數，其中第一個引數是 HttpservletRequest 物件，第二個是 HttpservletResponse 物件。HttpservletRequest 物件是用來存放傳入的資訊，而 HttpservletResponse 物件則用來存放 Servlet 要回應給客戶端的資訊。傳入的資訊包括顯性資料和隱性資料。顯性資料由使用者所提供，由詢問字串(Query String) 所組成，此字串包含由網頁上的表格所輸入的資料；隱性資料爲 HTTP 資訊。要送出的資料爲 Servlet 所產生的顯性及隱性資料，資料由 PrintWriter 物件使用 println()方法回應給發出請求的客戶端，回應的資料依客戶端而定，若客戶端接收者爲瀏覽器，則回傳的資料格式爲 HTML 或 XML 網頁。Println() 方法和 PrintWriter 一起使用以送出顯性資料，例如顯示在網頁上的文字；HttpservletResponse 物件裏的方法則用來傳送隱性資料。

Java Servlet 至少包括四個方法，分別代表 Servlet 程式生命週期裏的四個階段：init()、service()、適當的 request 方法、及 destroy()。

a. init

在 Servlet 的生命週期中，只會執行一次 init()函式，它是在伺服器載入 Servlet 時被執行，主要用來配置伺服器，以便在啓動伺服器或用戶端首次提出請求時載入 Servlet，無論有多少用戶端提出請求，只要是同一個 Servlet，這個函式都不會被重複執行。

b. service

service()函式可說是 Servlet 的核心，每當用戶端請求一個 HttpService 物件

時，都必須呼叫該物件的 service()函式，而且必須傳遞一個 ServiceRequest 物件和一個 ServiceResponse 物件作爲參數。

而 Servlet 的回應則有兩種類型：

1. 一個輸出串流，瀏覽器會依據它的內容類型(例如: TEXT/HTML)來進行解譯
2. 一個 HTTP 錯誤回應，重新導向到另一個 URL、Servlet 或 JSP。

c. destroy

當伺服器停止或卸載 Servlet 時會執行 destroy()函式，而且只執行一次。伺服器會在所有 service()函式呼叫完成後，或是在指定的逾時時間過後呼叫 destroy()函式；通常在呼叫 destroy()函式之前，我們必須確定所有執行緒都已經終止或完成。

Java Servlet 的優點：

1. 效能高
2. 使用方便
3. 功能強大
4. 可攜性高
5. 節省投資

✱ JavaServer Pages(JSP)

JSP 是一個建立在 Java Servlet 模型上的技術，讓撰寫 HTML 頁面變得更簡單，它允許使用者將靜態 HTML 內容與伺服器端的腳本程式(例如 JSP、ASP 與 PHP)整合以產生動態的 HTML 內容。爲了兼顧 Web 應用與簡單性，JSP 提供了大量的伺服器端標籤(Tag)，讓開發者可以使用最少的程式碼來完成大部分的動態內容操作，讓那些只熟悉網頁編排的人員可以使用 JSP 標籤建立簡單的 HTML 內容而不必去學習 Java 語言；然而，若要在 JSP 頁面進行更高階的操作，還是必須使用 Java 語言。

JSP 技術是 Java 爲了建立動態內容的 Web 網頁所提供的方法之一，簡言之，JSP 是一個純文字格式的文件，它包含了可以表現文件格式的靜態樣版資料(例如: HTML、XML)與構成動態內容的 JSP 標籤。

✱ JSP 與 Java Servlet 的區別

Servlet 發展在先，其功能較強，架構設計也較完整，但在輸出 HTML 頁面時仍採用類似 CGI 的方式(即一行一行輸出)，因此在撰寫和修改 HTML 內容就極不方便，後來 Sun 公司乃推出類似 ASP 的嵌入式腳本語言 JSP，把 JSP 標籤內嵌到 HTML 頁面中，如此一來，即可大幅簡化網頁的設計和修改。

✱ JSP 的特點：

1. "Write Once，Run Anywhere"可跨平台編寫執行
2. 實現元件的再利用
3. 簡化網頁設計
4. 可與 XML 整合

✱ JSP 系統架構：

來產生顯示層的內容；Servlet 則負責底層的運算處理，它扮演控制者角色，負責管理對請求的處理，建立 JSP 頁面所需要使用到的 EJB 和物件，再根據用戶端的動作決定傳回那一個 JSP 頁面。在 JSP 頁面內則無任何處理邏輯，只負責檢索由 Servlet 建立的物件或 EJB，從 Servlet 中取得動態內容並插入到靜態的網頁樣版中。

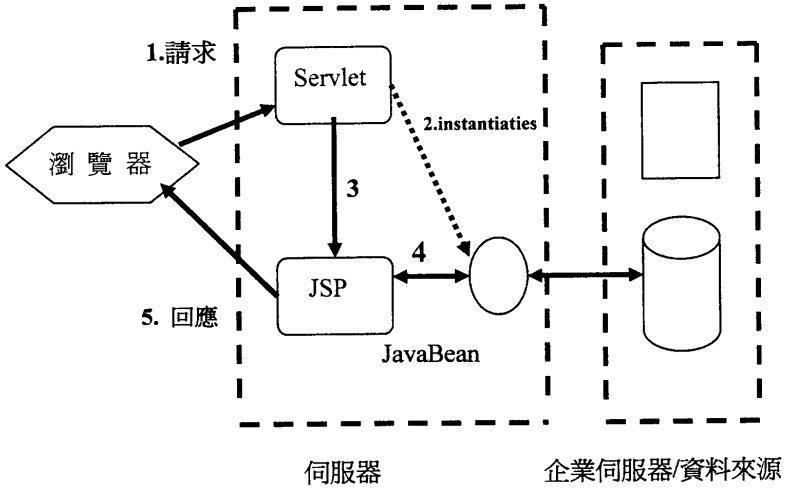

圖 4. JSP 系統架構

✱ JSP 標籤庫

標籤庫的特色

a. 容易使用與維護

b. 可擴充 JSP 功能

c. 快速開發

標籤庫的組成

1. EJB
2. 標籤處理器(Tag Handler)
3. 標籤庫描述檔(TLD 檔)
4. web.xml 文件
5. 部署檔(WAR 或 JAR 檔)
6. 標籤庫宣告

✱ JSP + Servlet + EJB 架構

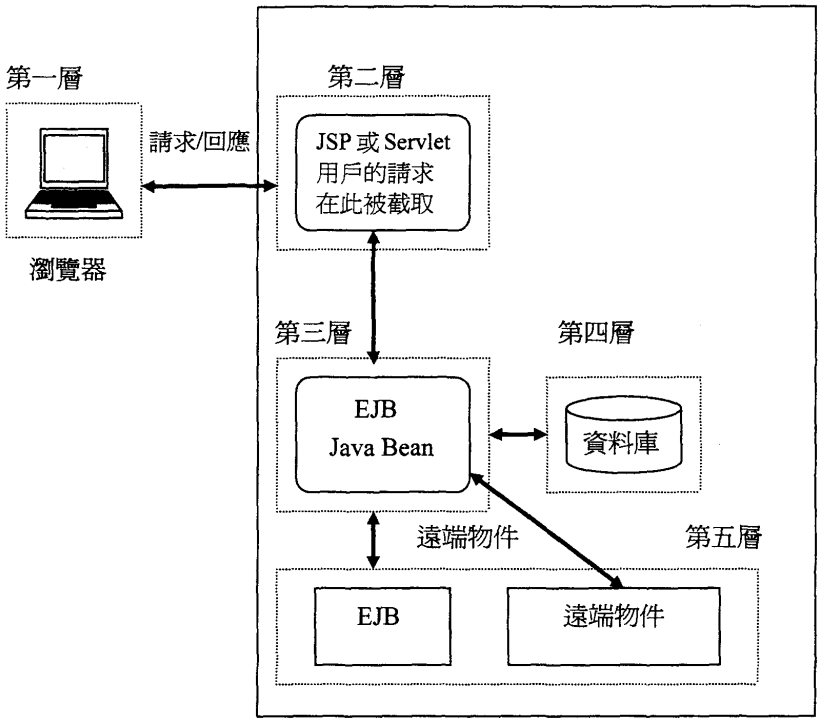

圖 5. JSP + Servlet + EJB 系統架構

● J2EE 應用伺服器

✱ J2EE 應用伺服器簡介

所謂應用伺服器(Application Server)，是在網際網路迅速發展之下，爲提供企業級應用與電子商務應用所產生的一種技術，透過它，可以將企業的商業活動在網際網路上安全且有效地運作，它並非傳統的電腦軟體，而是一種可以提供透過網際網路以執行電子商務的平台，具有極高的穩定性、可延伸性和安全性，可謂：「網際網路上的作業系統」。

✱ 應用伺服器的定義

所謂應用伺服器的定義是：採用具有分散式計算能力的集合架構，支援用戶端的軟體伺服器產品，其基本用途包括：管理用戶 Session、管理商業邏輯、

管理與後端資源(包括資料、事件和內容)的連接，同時它也是三層/多層式架構的一部分，屬於中間層，運作在瀏覽器和資料來源之間；在企業運用中，應用伺服器是位於企業資源和存取企業資源的用戶端之間的中間元件，提供商業程式碼存放和執行的環境，把商業邏輯與用戶端和資料來源分隔開，以提供商業系統快速開發和部署的環境。

一般應用伺服器可分三類：

1. 執行 Web 的應用伺服器(Pure-Play Web Application Server)

   例如：IBM WebSphere、BEA Weblogic，是以 Java 爲中心，幫助開發者使用 EJB 元件和 Servlet 來建立系統，並可使用喜歡的 Java IDE 開發工具來建立元件並載入伺服器中。

2. 開發及部署伺服器(Develop-and-Deploy Server)

   將開發環境與應用伺服器緊密結合，讓開發者迅速開發應用系統並立刻部署，例如：SilverStream、Borland Appserver 和 Sun NetDynamics。

3. 來自 Client/Server 供應商的應用伺服器(Application Servers from Client/Server Vendors)

   例如：Microsoft MTS 和 Sybase Enterprise Application Server 3.0。

   此類伺服器最主要的好處在於對原有程式碼的再利用。

應用伺服器的功能可分爲：

1. 安全管理

   爲了獲得資源存取的權限，用戶端必須通過伺服器的驗證，由伺服器決定是否允許或禁止該用戶端的元件存取以及資料連線。

2. Session 管理

   伺服器必須使用 Session 物件來保持用戶的資料，以簡化應用程式的開發，避免在程式中使用 Cookies 或隱藏欄位來傳遞狀態，以提高系統的安

全性。

3. 負載平衡

傳送到伺服器的請求由分配器(Dispatcher)處理，以便將請求轉送給伺服器群當中最空閒的伺服器；隨著用戶端數目的增加，可以加入更多的伺服器主機以分擔負載。

4. 失敗復原

提供系統的容錯性，並複製狀態和 Session 資料到其他伺服器或存在資料庫中，可在某伺服器當機時，新的請求會被重新分派給其他伺服器，用戶資料也可以繼續使用。

5. 商業和處理邏輯

商業邏輯集中在應用伺服器的核心部分，與應用程式相關的邏輯可由可再利用的元件組成，並載入到應用伺服器中，分配適當的安全設定並執行。

6. 產生 HTML 文件

應用伺服器解析 URL 以決定該執行那一個元件，元件則存取資料庫或其他元件，再將結果包裝成 HTML 文件後傳回瀏覽器。

7. 資料存取

應用伺服器可提供管理與關連式資料庫的連線機制，一旦資料來源被加入，元件就可進行呼叫，執行 SQL 請求。

8. 交易管理

在應用伺服器中，SQL 的回復(Rollback)或確認(Commit)動作是由伺服器來管理的。

9. 連線緩衝

爲避免耗用過多系統資源，應用伺服器都設計有連線緩衝來處理資料庫連線的工作。

J2EE 應用伺服器採用許多先進的開發理念與技術，而且能夠在原有的標準下不斷延伸，符合在網際網路上執行電子商務的需求：

1. 多層式架構：適合網路環境，系統有很好的可延伸性與可管理性。
2. 分散式環境：確保系統的穩定性，同時性能也較佳。
3. 物件導向的模組化元件設計：有效提高開發速度，降低開發成本。
4. 採用 Java 技術：完全跨平台，符合網際網路的需要，而且得到大多數廠商的支援，可保障用戶的投資。

✱ 選擇適合的 J2EE 應用伺服器

1. 成本

除了考慮主機 CPU 或伺服器數量，J2EE 應用伺服器尚需考慮：

1. 特殊平台授權
2. 機器或平台轉移授權
3. 系統支援費用
4. 開放費用(包括：開發工具授權、測試工具等)
5. .教育訓練

2.效率

考慮伺服器之效能。

3.需求

大部分系統事實上都只使用應用伺服器中的 JSP 與 Servlet 而已，甚至連 EJB 都用不到，故須明確界定系統真正所需要的功能。

4.擴充性

即使目前只需用到 JSP 與 Servlet，仍需考慮到日後是否有擴充的需求。

5.服務

雖然都符合 J2EE 規範，但每家廠商所推出的 J2EE 應用伺服器仍互有許多差異，所以日後的系統支援、教育訓練...等服務，也是選擇 J2EE 應用

伺服器時必須考慮的因素之一。

✱ J2EE 應用伺服器之安裝：

1. 取得軟體
2. 開始安裝
3. 環境設定
4. 啓動 J2EE 伺服器
5. 啓動部署工具
6. J2EE SDK 進階設定：
   a. 設定 JDBC 驅動程式
   b. 交易設定
   c. 通訊埠設定日誌設定
   d. 安全設定

三、系統整合與進階開發

● J2EE 應用程式部署與執行

前面各段已介紹了各個元件設計封裝的基本知識，以下將對應用程式的配置與部署作介紹。

✱ J2EE 應用程式的配置步驟：

1.建構各個元件，包括 EJB、JSP、Servlet 和 HTML....等。

2.將元件封裝成 J2EE 模組(JAR 檔或 WAR 檔)，同時提供部署描述檔。

3 結合一或多個 J2EE 模組以建立 EAR 檔，同時提供部署描述檔以建構出 J2EE 應用程式。

4 將 J2EE 應用程式部署到 J2EE 應用伺服器上。

J2EE 企業應用程式部署檔是由一或多個 J2EE 模組以及一個 META-INF\目錄下名爲 application.xml 之部署描述檔所構成，以 JAR 檔格式儲存在一個副檔名爲.ear 的文件中。 EAR 檔可以用部署工具來建立，也可以用 JDK 所提供的 JAR 工具來建立，步驟如下：

1. 建立一個用來存放 EAR 檔內容的目錄。
2. .把所有 J2EE 模組放入此目錄中，建立 META-INF\目錄。
3. 在 META-INF\目錄中建立 application.xml 部署描述檔。
4. 之後，進入該目錄，執行 jar 工具以建立 EAR 檔。

接著就可開始進行部署，所建立好的 EAR 檔即可拿到任何一台 J2EE 應用程式伺服器上進行部署，當然我們的程式需先符合 J2EE 規範才行。當元件被封裝成 J2EE 模組時必須要先有一個 EAR 部署描述檔，一般都可透過部署工具來自動建立，也可手動建立或修改成。在部署描述檔中，有兩種主要的標籤可用，其作用爲設定應用程式之內容與使用權限：

1.<module>標籤→是用來設定應用程式內容的標籤，其值爲 EAR 檔

根目錄的 URI，指向一個部署描述檔。

2.<security-role>標籤→用來指定應用程式的安全層級，這些角色將用於 EAR 檔中所包含的所有 J2EE 模組。

✱ 指定 JNDI 名稱：

當所有 J2EE 模組完成建立與封裝後，必須爲 EJB 指定一個正確的 JNDI 名稱，如此 Web 應用程式或用戶端程式執行時才能夠找到所需的 EJB。

✱ 部署 J2EE 應用程式：

1. 啓動部署精靈。

   產生 CalculateAppClient.jar 檔，它會包含所有遠端存取 CalculateEJ 時所需要的類別。

2. 設定 JNDI 名稱。
3. 設定 WAR 祖件的目錄。
4. 確認 OK，部署工具開始進行應用程式的部署。

✱ 執行 J2EE 應用程式：可分三種方式→

1. 透過用戶端程式執行
2. 透過 JSP 執行
3. 透過 Servlet 執行

● J2EE 提供的技術與服務

✱ JDBC(Java Database Connectivity)

是由 Java 語言所撰寫的類別與介面組成，是 Java 資料庫/工具開發人員的標準 API，他們因此可以使用純 Java API 來撰寫資料庫應用程式；JDBC 的概念其實可說是用一個單一的窗口來對各種資料庫作存取，和 ODBC 一樣，JDBC 爲開發人員處理不同資料庫間的差異性。

JDBC 定義了四種不同的驅動程式：

1. JDBC-ODBC Bridge(JDBC-ODBC 橋接器)
2. JDBC-Native Driver Bridge(JDBC 本地端驅動程式橋接器)
3. JDBC-Network Bridge(JDBC 網路橋接器)
4. Pure Java Driver(純 Java 資料庫驅動程式)

使用 JDBC 可以容易地將 SQL 與法傳送到任何關聯式資料庫中，不再需要爲每一個關聯式資料庫撰寫程式；用 JDBC API 寫出的程式能夠將 SQL 語法送到任何廠牌的資料庫；Java 與 JDBC 之結合，也讓程式設計師可以只撰寫一次資料庫應用程式，再跨平台執行，完全發揮 Java 語言的優點。

下面談到 JDBC 在企業應用程式中的應用：

1. 連線緩衝→在多層式企業應用程式中，通常會以一 EJB 來跟用戶端進行溝通，並藉以建立資料庫的連線，而爲了確保系統效能與延展性，多數 J2EE 應用伺服器都會提供對**連線緩衝** (Connection Pooling) 之支援，其功能爲減少建立和釋放資料庫連線所佔用的系統資源，當系統啓動後即可建立緩衝，此後若有任何對資料庫之請求，伺服器即可直接從緩衝區提取資料。
2. 設定 DataSources →用戶端若欲建資料庫連線，可透過查詢 JNDI 中的 DataSource 介面(JDBC 2.0)或 DriveManager 介面(JDBC 1.0)來取得對應的資料庫連線。
3. JDBC 的交易管理→其目的爲確保資料的完整性，在預設的情況下 JDBC 會使用 Auto-Commit 模式，也可以透過使用 Connection 類別的 setAutoCommit() 函式來實現。
4. 本地端交易→java.sql.Connection 介面可以控制交易範圍，在交易開始時呼叫 setAutoCommit(false)，而在交易結束時呼叫 rollback()或 commit()方法。
5. 分散式交易→其定義爲當同時有多個用戶端參與一個交易，或用戶端在同一個交易中執行跨多個資料庫的操作。

✱ JNDI

JNDI 是 Java 的名稱與目錄服務介面，名稱服務是一種應用程式，包含一個物件或物件引用的集合，並將每個物件關聯到一個用戶名稱。目錄服務則是名稱服務所提供的一個延伸功能。

實際上執行名稱與目錄服務的 JNDI API，提供一致的模型來存取和操作企業資源如 DNS、LDAP 或本地端系統。JNDI 的目錄結構中的每一個節點都稱爲 Context，而每一個 JNDI 名稱都是相對於 Context 的，在此並無絕對名稱的概念。

✱ JCA(Java Connector Architecture)

這是一個將應用伺服器和企業的一些應用資訊系統以最有效的方式整合起來的方案。由於 J2EE 本身對企業級應用程式整合的支援很少，所以 Sun 已經把 J2EE Connector Architecture 納入 J2EE 1.3 規範內。它定義了標準的資源轉接器(Resource Adapter)和相關的連線、交易與安全管理，使 J2EE 伺服器可用標準化和統一的方式使用各種企業資訊系統諸如 ERP(Enterprise Resource Planning)、CRM(Customer Relationship Management)..等。基本上 JCA 是一組協定，讓企業資訊系統以插入的方式連接到 J2EE 伺服器上，這種架構定義了負責連結的資源調配器，而此 SPI(The Connector Service Provider Interface, SPI)，正好可與 JDBC 介面緊密結合。所有透過 JCA 整合的應用程式是具有跨平台性的，而且遵守 JCA 規範的資源調配器均可支援兩套標準界面：一是應用程式用來與其他資訊系統溝通，另一則是用戶端與企業資訊系統溝通。

使用資源調配器有以下好處：

1. 可讓一個應用程式伺服器存取很多需要使用到的資訊系統。
2. 可讓一個應用程式伺服器在多個資源間使用交易管理。
3. 提供應用程式環境對安全性的支援，降低資訊系統潛在的安全威脅，以保護企業的資料資源。

JCA 另外還定義了一個用戶端與資訊系統溝通的介面，它會呼叫此用戶端

介面(Common Client Interface, CCI)以連線和存取後端系統的程式介面，此介面是一個類似 JDBC 的低階 API，而 CCI 主要任務在處理應用程式與系統之間的資料流程，其優點是非常容易使用、具高延伸性且可跨不同廠牌的 EIS。

✱ JTA(Java Transaction Architecture)與 JTS(Java Transaction Service)

JTA 由兩部分所組成：高階的用戶端交易介面和低階的 XA 介面，EJB 可直接存取用戶端交易介面，XA 介面則由 EJB 伺服器和 EJB 容器使用以協調交易和資源管理。簡言之，JTA 定義了一個標準的 API，使應用系統可以藉此存取各種交易管理。

JTS 亦稱元件交易監視器(Component Transaction Monitor, CTM)，它規定了交易管理實現的方式，並在高層支援 JTA 規範，在底層實現 OMG 物件交易服務(Object Transaction Service, OTS)規範的 Java 對應，爲應用伺服器、資源管理器、獨立應用程式與通訊資源管理器提供交易管理服務。

✱ JMS(Java Message Service)

這是一個通訊應用程式介面，可提供點對點的訊息發佈與訂閱服務，並支援經過認可的訊息傳遞、交易模式訊息傳遞、訊息的一致性與長時間的訂閱，並能讓獨立應用程式之間可藉由非同步通訊機制來溝通。非同步通訊機制可分爲發佈/訂閱(Publish/Subscribe)與點對點(Peer to Peer)兩種模式。

✱ JavaMail

這是一組用來存取郵件伺服器的 API，提供郵件伺服器的抽象類別，支援 SMTP 伺服器與 IMTP 伺服器；它利用 JAF(JavaBeans Activation Framework)來處理 MIME 編碼的郵件附加檔案，使 MIME 的位元組串流可以直接被轉換成 Java 物件，或是直接從 Java 物件轉換成 MIME 的位元組串流，毋須直接使用 JAF。

以下是 JavaMail 的幾個主要類別：

1. javax.mail.Session
2. javax.mail.Message

3. javax.mail.Address

4. javax.mail.Authenticator

5. javax.mail.Transport

6. javax.mail.Store

7. javax.mail.Folder

● J2EE 的安全性

由於企業應用程式關係到各種商業機密，其安全的議題比一般應用程式更爲重要，因此當應用程式被部署時，必須符合對它在安全上的要求。

J2EE 應用程式可分許多層級：從使用類別的 Java Statement 到關於 JVM 的低階程式設計，任何一層級都有安全性的考慮與措施。安全性會在完全保護(Utimate Protection)和效能(Utilization)間求取平衡，如果安全機制愈傾向完全保護，則效能自然會降低些, 反之亦然。適當的應用程式安全性取決於工作層級/效能因素，在高階環境下，使用者接受嚴格的安全檢查，然後犧牲一程式效能；在低階環境下，使用者接受的安全檢查較鬆散，而程式效能則較佳。

✱ JVM 的安全性

JVM 會防止 Java 程式執行一些容易妨害到作業環境的動作，因爲在執行類別之前 JVM 會先驗證類別，如此可避免諸如系統資源(CPU 等)被某一個 Java 程式獨佔。

JVM 並非依賴 Java 程式語言來保證執行的類別是安全的，而是在執行類別之前先檢驗包含類別的位元組碼(bytecode)來保證安全性。

安全管理員(security manager)可以使用在設定於 JVM 裏的安全特性以管理一些安全行爲，其目的就在控制管理 JVM 執行時期程式的動作，且讓 J2EE 程式建立自己的安全原則來限制程式的動作。

安全管理員在防止違反安全行爲上非常重要，例如防止一個遠端載入的 Java applet 存取本機端檔案系統；同時安全管理員也會限制一個遠端 applet 只能連結到產生此 applet 的伺服器。

安全管理員包含一些預設的守則以強制那些在 JVM 上執行的程式遵守，這些守則名爲 permission，並且被定義在 Java 安裝目錄/jre/lib/security/java.-policy 裏。

✱ Java API 的安全性

在 J2EE 應用程式中，有數個套件被用來加入額外的安全保護，例如在 java.security.*套件裏的 Java Cryptography Architecture(JCA)和 javax.crypto.*套件裏的 Java Cryptography Extension(JCE)，其中包含 cryptographic algorithm、secure stream、key generation、certificate、digital fingerprint 和 signature 等技術。

✱ 瀏覽器的安全性

由於 Java applet 的遠端電腦載入特性，其安全性是很薄弱的。Java applet 的安全管理員就會限制 applet 的動作，例如：不允許 applet 存取本機端電腦的資料；不允許 applet 對一個 IP 位址開啓 socket 以避免駭客藉由使用本機端電腦來攻擊其他的遠端電腦；不允許 applet 存取取本機端電腦上管理收取訊息的連接埠以免被 applet 攔截並重新導向此接收訊息。

✱ Web Service 的安全性

對於在 Web Service 環境下的 J2EE 應用程式而言，其安全性絕對重要！因爲對於服務所發出的請求和元件的回應在透過網路傳送中極易遭入侵。

Web Service 是一個多層式架構，其元件分佈於各層次之間，彼此可以互相通訊連結。WSDP(Java Web Services Deployment pack)被設計爲用來透過在多層式架構下的 Web Service 來滿足安全需求，不論元件位於架構下的那一層，WSDP 都爲元件提供一個安全需求。

＊Web Services 安全規則裏的存取權限可分爲：

1. 使用者(User)
2. 角色(Role)
3. 群組(Group)

＊Web Services 各層之安全性

Web Server 是一個多層架構，每一層都包含一個網路資源集(Web Resource Collection)，是 URL 符號列表和 HTTP 方法列表，而安全性條件約(Security Constraint)可用來允許或禁止對網路資源集的存取。網路容器(Web Container)使用安全限制來驗證要求網路資源集服務的使用者。當使用者被驗證通過後，就不會再有網路容器對使用者要求驗證，亦即使用者(J2EE 程式)可以存取任何資源，不再需要額外的認證。

認證的機制有下列三種：

1. 基本認證(Basic Authentication)

亦稱 HTTP 基本認證(HTTP Basic Authentication)，網路伺服器需要使用者名稱和密碼來認證使用者。

2. 表單認證(Form-Based Authentication)

使用 HTTP 瀏覽器顯示一個登入畫面，並以之擷取使用者名稱和密碼來認證。

3. 用戶端憑證認證(Client-Certificate Authentication)

較上述兩種方式安全，使用 HTTPS 通訊協定，伺服器與用戶端有一個公開金鑰，而 SSL(Secure Socket Layer)則提供資料的加密，以進行 TCP/IP 伺服器認證、傳遞資料的完整性與用戶端認證。

基本上表單認證與基本認證都不是非常安全，表單認證時使用者所輸入的資料是以存文字方式傳送，而且目標伺服器並未經過認證；基本認證雖已將資料編碼，卻也沒有加密，如果被有心人擷取網路封包，使用者的名稱與密碼等重要資料還是無法保護，所以解決之道還是要使用 SSL 來提高表單認

證與基本認證的安全性。

＊ EJB 之安全性

由於所有的商業邏輯都在 EJB 中，EJB 的安全性更顯重要。EJB 資源的保護可透過宣告函式權限與對應角色到 J2EE 使用者群組來實現。當我們定義好應用程式角色後，可以定義一個 EJB 函式的權限，權限決定那些角色可以呼叫函式，再使用部署工具 Deploytool 對應角色與函式來指定函式的權限。此外，EJB 程式碼中的安全管理主要是由 getCallerPrincipal()與 isCallerInRole()兩函式組成，其中 getCallerPrincipal()判斷該 EJB 元件的呼叫者，isCallerInRole()則可以判斷該呼叫者的角色以決定其是否可以呼叫 EJB 的函式。

＊ 用戶端程式之安全性

J2EE 應用程式用戶端可以使用 Java 認證和授權服務(Java Authentication and Authorization Service, JAAS)來取得操作權限，在 JAAS 下，可以賦予用戶端或服務特定的權限來執行 Java 類別中的程式碼。而可插拔的認證模組(Pluggable Authentication Module,PAM)則允應用程式在使用認證技術的同時保留其獨立性，程式開發者可以隨時插入或更新應用程式的認證技術而不必更改程式。

＊ EIS 之安全性

在企業資訊系統(EIS)的整合上，J2EE 應用程式元件會對 EIS 資源提出連線的請求，通常 EIS 都會要求以一定的安全方式進行登錄，在應用程式元件開發者則以下面兩種方式進行登錄：

1. Container-Managed 登錄模式 → 由應用程式元件的容器負責設定與管理 EIS 的登錄，確認使用者名稱與密碼以建立 EIS 連線。
2. Component-Managed 登錄模式 → 由應用程式元件在程式中管理 EIS 資源的登錄。

＊ 伺服器憑證

資料加密是安全的核心，Java 平台有兩組提供安全和加密服務的 API：JCA 和 JCE。JCA(Java Cryptography Architecture) 提供基本的加密框架，諸如：數位簽章、訊息摘要和金鑰產生器等；JCE(Java Cryptography Extension)則是在 JCA 的基礎上作延伸，包含：加密演算法、金鑰交換、金鑰產生和訊息確認服務等介面。一般而言，JCA 或 JCE 並不會實際執行各種演算法，它們只會連結應用程式和演算程式的一組介面，軟體開發廠商依據 JCE 介面，將各種演算法實現後，封裝成一個 Provider，然後動態地加到 Java 執行環境中。

大部份 J2EE 應用伺服器中都會包含一組完整的 JCE Provider，以提供數位簽章、訊息識別、序列密碼加密、私密金鑰...等金鑰產生或交換的演算法，以實現各種安全架構，使電子商務或企業的機密資料更加安全可靠。

肆、 感想與建議

企業電腦化的腳步隨時都是進行式，而且只會愈來愈快速，尤其是網路的出現與演進，使得各家企業不只須將生產自動化、業務電腦化，對於各種應用程式的需求更急速增加，因此，具備改進與整合既有應用程式的能力更形重要。

企業會成長，在過程中幾乎都會面臨整合不同應用程式與資料庫系統的難題。此外，企業組織必須經常變革，當企業亟需爭取競爭優勢時，便產生變革的需求。

「滿足現狀，就是落伍；維持現狀，就是退步。」企業要維持在業界的優勢地位，變革應是唯一的路。變革的方式有許多種：整頓內部組織、採用新技術與新平台、合併或結盟其他企業、採用電子商務策略...等。

電子商務模式在企業管理採購與供應問題、管理客戶關係、提供客戶服務、提供網路架構的應用程式與服務方面特別實用。由於企業需要適應業務與技術的變革，因此更需要以電子商務模式讓既有的業務流程、應用程式與企業系統適應這種變動。然而，要企業脫離既有的應用程式，全面改掉已經有固定架構且行之有年的業務流程殊非易事，像這類的變更會所費不貲，會耗費大量的人力物力資源，許多企業還是無法進行這樣的變革或是擺脫原有的系統。因此，靈活運用既有的企業基層架構與對於應用程式的投資，是另外一條可行之路。

企業應用程式整合(Enterprise Application Intergration, EAI)在此具有相當重要性且值得考慮運用，它讓企業的資訊技術部門得以整合既有的應用程式與資料資源，並在其中加入新技術與新應用程式，而在此網路世紀更須及於極端重要的 Web 導向之架構服務，因此，企業應用程式需要配置在被廣泛採用、標準的應用程式平台上；應用程式伺服器已被公認是企業舊有資訊系統與 Web 架構應用程式之間整合的關鍵，J2EE 應用程式平台已成爲企業與應用程式供應商的熱門選擇。J2EE 的基礎是 Java 語言，Java 是沒有平台依賴興的電腦語言，專爲 Web 而設計，而且是非常成功，廣受採用的企業應用程式開發平台。

中華電信長久以來在通訊業界獨佔龍頭地位，我們是以電話服務供應商的角色起家，在成長過程中逐步加入許多新的業務與新的資訊系統，也歷經幾次重大的組織調整變革，目前更著眼於即將面臨的民營化挑戰，成長與變革永遠是公司不變的方向，因此許多措施，諸如整合各種服務(包括訂單、技術、帳單、後續客服等)以及整合存取提供這些服務的既有應用程式，就顯然有急迫需要。J2EE 所提供的 JCA 技術，對於 EAI 既然有其明確方案，應可爲經營階層參考之方向。此外，目前已陸續在各區分公司上線運轉的 E-Tris 與 E-Leamis 系統，亦爲 Java 平台下發展而成之應用程式架構，足見此一新型技術亦逐漸在公司內獲重視並據以改善舊系統，以新面貌及更完善之系統架構與功能延續該項服務，深信未來持續對 J2EE 技術之引用，必能有助於公司體質之改善與服務之加強。

專 用 術 語

Application Program Interface (API)：應用程式設計介面

Application Component Provider：應用程式元件提供者

Application Assember：應用程式組合者

ACID(Atomic、Consistent、Isolated、Durable)

Authentication：身份驗證

Authorization：授權(權限管控)

Bean-Managed Persistence(BMP)：Bean 管理的永續性

Business Logic：商業邏輯

Commit：提交

Component：元件

Connector Architecture：連接器架構

Container：容器

Container-Managed Persistence(CMP)：容器管理的永續性

Common Object Request Broker Architecture(CORBA)：共同物件需求中介架構

Deployment：佈署

Deployment Descriptor：佈署描述檔

Distributed Application：分散式應用程式

Enterprise JavaBeans (EJB)

EJB Server Provider：EJB 伺服器提供者

Enterprise Information System(EIS)：企業界資訊系統

Extention Markup Language(XML)：可擴充標記語言

Home Interface：主介面

HyperText Markup Language (HTML)：超文字標記語言

HyperText Transfer Protopol (HTTP)：超文字傳輸協定

Interface Definition Language(IDL)：介面定義語言

Internet Inter-ORB Protocol(IIOP)：網際網路物件需求中介通訊協定

Java 2 Platform, Enterprise Edition (J2EE)

Java 2 Platform, Standard Edition (J2SE)

Java Authentication and Authorization Service(JAAS)： Java 認證和授權服務

Java Database Connectivity (JDBC)：Java 資料庫連接介面

Java Developers Kit (JDK)：Java 開發者工具

Java Message Service(JMS)：Java 訊息服務

Java Naming and Directory Interface (JNDI)：Java 命名和目錄介面

Java Server Pages(JSP)

Java Transaction API(JTA)：Java 交易 API

Java Transaction Service(JTS)：Java 交易服務

Miltipurpose Internet Mail Extensions (MIME)：多用途網際網路延伸標準

Object Management Group(OMG)：物件管理組織

Object Transaction Service(OTS)：物件交易服務

Pattern：樣式

Persistence：永續性

Remote Interface：遠端介面

Resource Adapter：資源轉接器

Remote Method Invocation(RMI)：遠端方法引用

Rollback：復原

Security Constraint：安全性條件約束

Security Role：安全性角色

Session：連線

Secure Socket Layer(SSL)：網路安全協定

Transaction：交易

Uniform Resource Identifier(URI)：全球資源識別元

Uniform Resource Locator(URL)：全球資源定位器

Web Resource Collection：網路資源集