行政院及所屬各機關出國報告
(出國類別： 實習)

赴美國實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」出國報告

服務機關：中華電信研究所
出國人　職　稱：助理研究員
　　姓　名：葉 文 宏

出國地區：美國
出國期間：91 年 12 月 09 日 91 年 12 月 22 日
報告日期：92 年 02 月 22 日

H6/c09200965

系統識別號:C09200965

公 務 出 國 報 告 提 要

頁數: 24　含附件: 否

報告名稱:

實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」

主辦機關:

中華電信研究所

聯絡人／電話:

楊學文／03-4244218

出國人員:

葉文宏　中華電信研究所　91860專案研究計畫　助理研究員

出國類別: 實習

出國地區: 美國

出國期間: 民國 91 年 12 月 09 日 -民國 91 年 12 月 22 日

報告日期: 民國 92 年 02 月 22 日

分類號/目: H6／電信　／

關鍵詞: OSS,系統整合,基礎營運

內容摘要: 本次赴美國實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」，主要是到BEA Systems, Inc實習，爲了深入了解在Java平台上，開發企業級應用軟體的最新架構與技術。實習的內容涵蓋如何在J2EE平台上建構企業邏輯元件(Business Logic Components)，及如何運用J2EE架構來整合企業資訊系統。 報告內容將對J2EE(Java 2 Enterprise Edition)的架構作一個簡單的描述，並對EJB2.0規格的重要特性，作一些摘要的介紹。內容包括J2EE架構介紹，EJB 2.0規格簡介，Session Bean在EJB2.0中所扮演的角色，Entity Bean的特性，EJB2.0所定義的訊息處理架構，以及EJB Container對 Transaction所作的管理。 實習的過程中，讓職得以深入了解J2EE平台的優缺點，及EJB2.0規格中所定義的新的特性，並有機會與來自其它公司的工程師，互相交流系統開發的實務經驗，對往後在整合電信維運支援系統及開發基礎營運支援服務系統時，有很大的幫助。

本文電子檔已上傳至出國報告資訊網

摘要

本次赴美國實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」，主要是到 BEA Systems, Inc 實習，為了深入了解在 Java 平台上，開發企業級應用軟體的最新架構與技術。實習的內容涵蓋如何在 J2EE 平台上建構企業邏輯元件(Business Logic Components)，及如何運用 J2EE 架構來整合企業資訊系統。

報告內容將對 J2EE(Java 2 Enterprise Edition)的架構作一個簡單的描述，並對 EJB2.0 規格的重要特性，作一些摘要的介紹。內容包括 J2EE 架構介紹，EJB 2.0 規格簡介，Session Bean 在 EJB2.0 中所扮演的角色，Entity Bean 的特性，EJB2.0 所定義的訊息處理架構，以及 EJB Container 對 Transaction 所作的管理。

實習的過程中，讓職得以深入了解 J2EE 平台的優缺點，及 EJB2.0 規格中所定義的新的特性，並有機會與來自其它公司的工程師，互相交流系統開發的實務經驗，對往後在整合電信維運支援系統及開發基礎營運支援服務系統時，有很大的幫助。

目錄

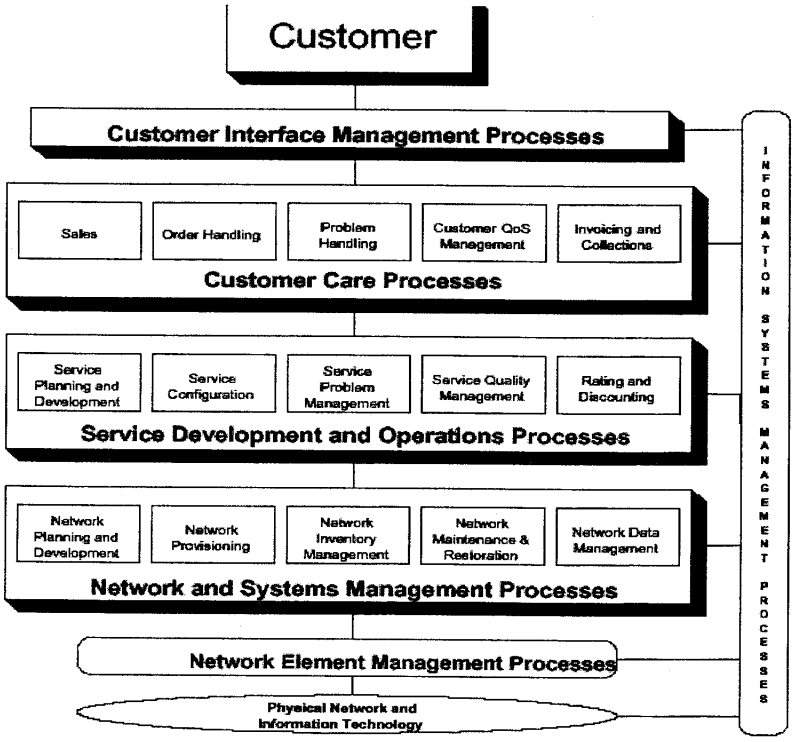
圖表

第一章　研習目的

中華電信為一歷史悠久的電信公司，隨著時代的變遷與資訊技術的演進，其服務型態，也從傳統以語音及專線電路為主的固網服務，演進到目前以行動通信、數據業務及寬頻服務為主。為了支援營運需求，各種不同的維運支援系統(OSS)及營運服務支援系統(BSS)也陸續被開發出來(如圖一)，這些資訊系統，由公司內不同的單位所開發、維運、掌管，使用不同的程式語言撰寫(例如：C、C++、Java 等等)，在不同的作業系統(例如：Windows、Linux、HP-UX、Sun Solaris、IBM AIX 等等)上執行。隨著電信市場開放，面對未來激烈的競爭，我們必須能快速提供新型態的服務，彈性的價格及費率，同時還要能整合相關的客戶資料(例如：銷售資料、服務資料、行銷資料等等)，培養快速的系統開發、建置能力，並對未來成長作預先的投資規劃，因此亟需尋求一個妥善的方式，以期能以最低的成本將相關系統整合起來，並預先規劃未來與新系統的整合模式，實現『及時滿足顧客與市場之需求』的品質政策，讓公司能在競爭的市場中脫穎而出。

本次研習，選擇以 Java 平台作為我們評估「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」的應用平台，主要是著眼於 Java 具有跨平台的優越性，用 Java 開發出來的應用可以在 Windows、Linux、HP-UX、Sun Solaris、IBM AIX 等不同的作業系統上執行；而且，Java 技術的標準，也是由領域內的各領導廠商所共同制定並接受，包括 IBM、HP、Sun、Oracle、BEA 等等；且受到許多 Open Source 組織的支持，例如 http://jakarta.apache.org/ 等；為一開放的技術，且目前已臻成熟，執行效能及穩定度均已受肯定，而 J2EE 標準更是針對企業級應用系統的特性所制定。如 Sun 的網站所描述的：

『*J2EE™ technology and its component-based model simplifies enterprise development and deployment. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications.*』(http://java.sun.com/j2ee/)



圖表 一、Telecom Operations Map, Business Process Model

資料來源：eTOM, [3]

而選擇以 BEA System, Inc. 作為研習的地點，主要是因為 BEA System, Inc. 為 Java 平台上應用伺服器產品的領導廠商，該應用程式伺服器目前在多項效能評比中均為第一，且在受到世界各地多家公司採用。其 Know-How 及 Know-Why

均是值得我們學習的地方。

『*BEA is the world's leading application infrastructure software company with more than 13,000 customers around the world, including the majority of the Fortune Global 500. Companies turn to BEA to help them evolve their existing enterprise software applications from inflexible, redundant, legacy client/server architectures to highly responsive, mature Web infrastructures. Companies built on BEA software are able to use IT to effect rapid change within their organizations and achieve breakthrough levels of efficiency and responsiveness.* 』(http://www.bea.com/about/index.shtml)

職為電信研究所 860 專案之成員，負責 IOIS/eTRIS(工務資訊系統整合計畫/障礙資訊系統)的系統開發工作。本系統在 Java 平台的開發，並必須與其它相關的資訊系統作資訊整合，受派前往美國實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」，對專案的開發，有直接且顯著得幫助。職於 91 年 12 月 09 日搭乘長榮班機由桃園中正國際機場出發，於美國當地時間 12 月 09 日下午抵達，並於次日前往位於加州聖荷西(San Jose, CA)的 BEA System, Inc.，展開為期 11 天的研習。研習結束後，於美國當地時間 12 月 21 日搭乘長榮班機返回台灣， 91 年 12 月 22 日抵達桃園中正國際機場。

在報告的第二章，將對研習的內容作一些摘要的報告，而第三章則是本次研習的心得及一些建議。

第二章　研習內容摘要

本此研習，選擇 BEA Systems, Inc. 作為這一次赴美國實習「電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術」的地點，研習的重點放在如何運用 J2EE 平台上來建構企業邏輯元件(Business Logic Components)，及了解如何運用 J2EE 架構來整合企業資訊系統。以下便利用幾個章節來摘要介紹這次研習的內容。

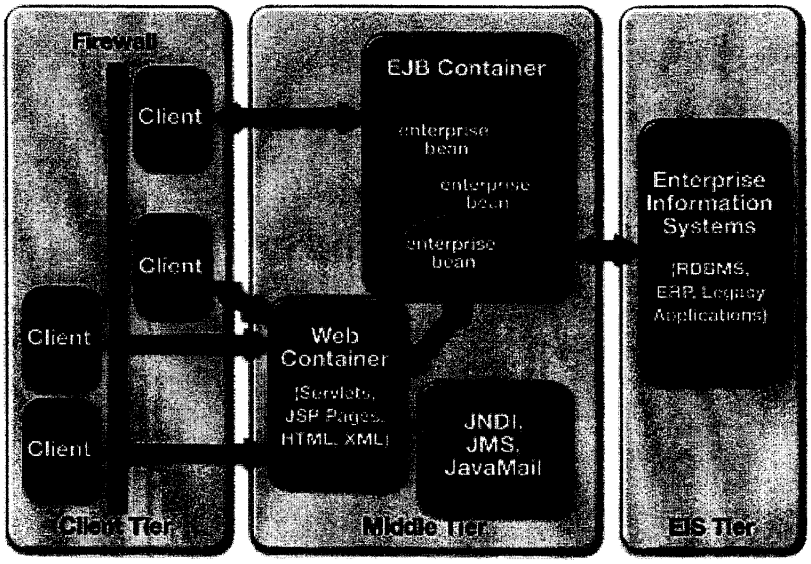
第一節　J2EE 架構簡介

J2EE 平台乃在定義一個開發及建置企業應用軟體的標準，提供企業一個開放的平台，使企業得以用最低的成本、最快的速度，開發建置應用軟體，維持競爭力。由於，標準的制定過程開放，且邀集領域內的領導廠商及專家來共同制定，而使其今日能夠成功，為大家所接受。

『This success is largely due to the fact that the J2EE platform has been designed through an open process, engaging a range of enterprise computing vendors to ensure that it meets the widest possible range of enterprise application requirements.』 [2]

整個 J2EE 標準是由一系列相關的規格集合而成，其中包括了 Java Server Page、Java Servlet、EJB 等等，目的在提供企業一個開發分散式(distributed)、多層式(multi-tier)應用軟體的架構及方法，其架構圖如圖二所示。其特性如下：

- Multi-tier Model
- Container-Based Component Management
- Support for Client Components
- Support for Business Logic Components



圖表 二、J2EE 架構圖

資料來源：Design Enterprise Application with J2EE, [2]

在這個分散式、多層式的架構下，具有許多優點[2]，例如：

- 架構簡單，應用軟體開發可以分工
  - Maps easily to application functionality
  - Enables assembly- and deploy-time behaviors
  - Supports division of labor
- 容易與存在的資訊系統整合
  - The J2EE Connector architecture is the infrastructure for interacting with a variety of Enterprise Information System types, including ERP, CRM, and other legacy systems
  - The JDBC™ API is used for accessing relational data from the Java programming language
  - The Java Transaction API (JTA) is the API for managing and

coordinating transactions across heterogeneous enterprise information systems

- The Java Naming and Directory Interface™(JNDI) is the API for accessing information in enterprise name and directory services
- The Java Message Service (JMS) is the API for sending and receiving messages via enterprise messaging systems such as IBM MQ Series and TIBCO Rendezvous. In the J2EE platform version 1.3, message-driven beans provide a component-based approach to encapsulating messaging functionality
- The JavaMail™API is used for sending and receiving e-mail
- Java IDL provides the mechanism for calling CORBA services
- Java APIs for XML provide support for integration with legacy systems and applications, and for implementing Web services in the J2EE platform

- 企業可以從市場上自由選擇不同的應用伺服器及元件，並加以組合
- 有很好的擴充性

  J2EE containers provide components with transaction support, database connections, life cycle management, and other features that influence performance, they can be designed to provide scalability in these areas.
- 具彈性的、一致的安全模型

  The J2EE security model is designed to support single sign on access to application services. Component developers can specify the security requirements of a component at the method level to ensure that only users with appropriate permissions can access specific data operations. It also provide programmatic security control, the basic role-based security

mechanism (where groups of users share specific permissions) is specified entirely at application deployment time. This provides both greater flexibility and better security control。

第二節 EJB (Enterprise JavaBeans)簡介

EJB 是這一次研習的重點，從 J2EE 的架構圖，我們不難發現，EJB 其實是定義伺服端(server-side)的元件模型及架構，他扮演一個中介者(middleware)的角色，提供企業服務的商業邏輯，都封裝在 EJB 的元件中。當我們要開發一個大型的企業資訊系統時，我們可能會考慮到下列問題：

- Client 如何跟 Server 連接，Server 如何同時服務很多 clients，
- 如何平衡 Server 的負擔 (load balancing) ，
- 當網路或 Server 有問題時，如何繼續提供服務給使用者 (transparent fail-over)，
- 如何與後端系統整合，
- 如何管理 transaction，
- 如何建立 clustering servers，
- 如何動態部署企業邏輯元件，
- 企業邏輯元件的生命週期，
- 如何做 logging 及 auditing，
- 如何做好系統管理機制，
- 如何共用系統資源 (resource pooling)，
- 安全機制如何管理，
- 如何 caching 資料，等等等。

而 EJB 的規格[4]，恰好為我們解決這些惱人的問題，它定義了開發和部署

伺服端 Java 元件的標準，同時也明確界定出 EJB 元件和 EJB Container 之間的互動關係、EJB Container 和 AP Server 的關係、以及，EJB Container 和 EJB client 之間的關係，其優點在於[5]：

- Develop scalable, accessible, robust, and highly secure applications rapidly relying on the J2EE platform to handle complex system-level issues，
- Construct server-side components quickly and easily by leveraging a prewritten infrastructure provided by industry，
- EJB is designed to support applications portability and reusability across any vendor's enterprise middleware service。

因此，遵循這個架構，很多複雜的問題，都將由 EJB Container 及 AP Server 來接手管理。企業只需要專心開發跟企業邏輯相關的元件(EJB Components)，因此可以很快的發展出具擴充性、高可靠度的應用系統。

此外，EJB 規格中也描述了六個不同的角色(role)，每個角色都由相當專業的員工或廠商來扮演，不同的角色互相分工合作，迅速開發並部署企業應用系統，這六個不同的角色分別是：

- EJB Bean Provider，負責開發企業邏輯元件，
- Application Assembler，相當於企業應用系統的建築師，能針對問題的所在，去整合相關的元件，解決企業問題，
- EJB Deployer，負責部署企業應用系統，
- System Administrator，經由 Java Management Extension(JMX)標準介面，監督及管理 AP Server 的運行狀態，
- EJB Container/ Server Provider，EJB Server Provider 通常也是 EJB Container Provider，提供 EJB Components 的執行環境，通常為業界領

導廠商，如 BEA's WebLogic、IBM's WebSphere、開放原始碼的 Jboss 等等，

- Tool Provider，提供便利的整合性工具，方便來開發、管理、部署 EJB 元件及 EJB 應用程式。

對於 EJB 元件的規範，在 EJB2.0 規格中總共定義了四種不同類型的元件，分別是 Stateless Session Bean、Stateful Session Bean、Entity Bean 以及 Message-Driven Bean，這些不同類型的 EJB 元件，其生命週期及 transaction 均交由 EJB Container 來管理。在後續的章節中，將依序對 Stateless/stateful Session Bean、Entity Bean 以及 Message-Driven Bean 對介紹，並介紹 EJB Container 的 transaction management。

第三節 Stateless/Stateful Session Bean 介紹

EJB2.0 規格中定義的 session bean 有兩種，一種為 stateless，另一種為 stateful，一般我們將所謂的企業邏輯封裝在 session bean 之中。對 stateless session bean 而言，它可以被重複使用並提供單一次的服務，也不會去儲存 client 端的資訊，例如，可以用來查詢股票目前的價格、計算保險的費率等等。其特性如下[1]：

- provide independent services
- do not maintain state on behalf of the client across multiple calls
- are interchangeable (calls can be handled by any instance of the same type with the same results)
- are synchronous
- are only maintained in memory
- do not survive EJB server crash

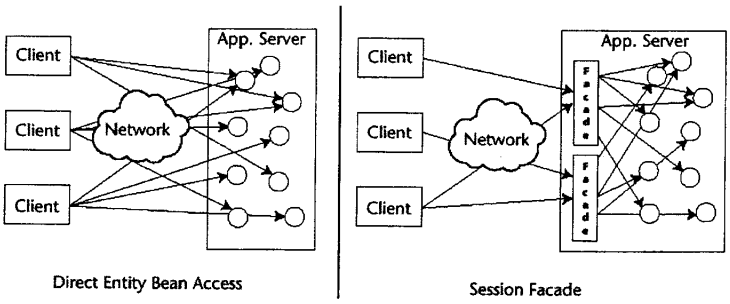
而 stateful session bean 則可以用來儲存整個交易過程中 client 端的資訊，例如可

以用來儲存購物車的內容，其特性不同於 stateless session bean，說明如下[1]：

- provide conversional interaction
- store state on behalf of the client
- are not interchangeable (each instance is associated to a single client)
- are synchronous
- are maintained in memory
- do not survive EJB server crash, except when fail-over is configure

對於 stateless session bean，從實務的經驗來分析，一般我們有三種不同的使用情境：

- 作為 client 端及 entity bean 的介面，用來封裝多次的 entity bean 存取，我們亦稱之為 session facade。Entity bean 通常用來直接面對後端的資料儲存裝置，例如資料庫或檔案系統，有時 client 端的程式在執行一個功能時，會需要操作不同的 entity bean 多次，如此會增加 client 端及 EJB Server 之間網路通訊的次數，及物件序列化(serialize)的次數。若使用 stateless session bean 來封裝多次的 entity bean 存取，則 stateless session bean 在接受 client 端請求後，在同一 EJB Server 中存取相關的 entity bean，最後把結果傳回 client 端，僅需一次網路通訊即可完成。



圖表 三、The architectural benefits of session façade

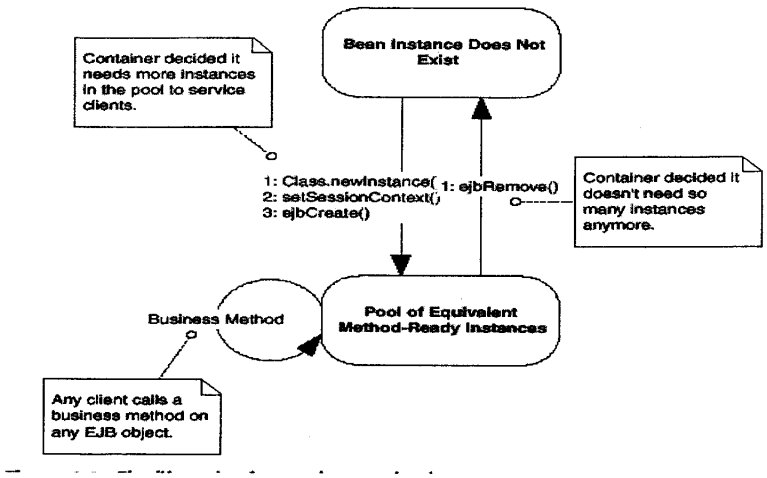
資料來源：EJB Design Patterns, [6]

- 作為 client 端存取資料庫的介面。即 client 端不直接透過 JDBC 去存取資料庫，而將存取資料庫的方式及邏輯封裝在 stateless session bean 之中。
- 把需要多個步驟才能完成的企業流程封裝在一個 stateless session bean 中。每一個步驟在 session bean 中被實作成一個 method，client 端僅需一次呼叫就可以完成企業流程的多個步驟，而不同的 client，也可以重複運用這些企業流程。

對於 stateful session bean，因為可以儲存整個交易過程中的資訊，所以在實務上的應用，一般我們也有幾種不同的使用情境：

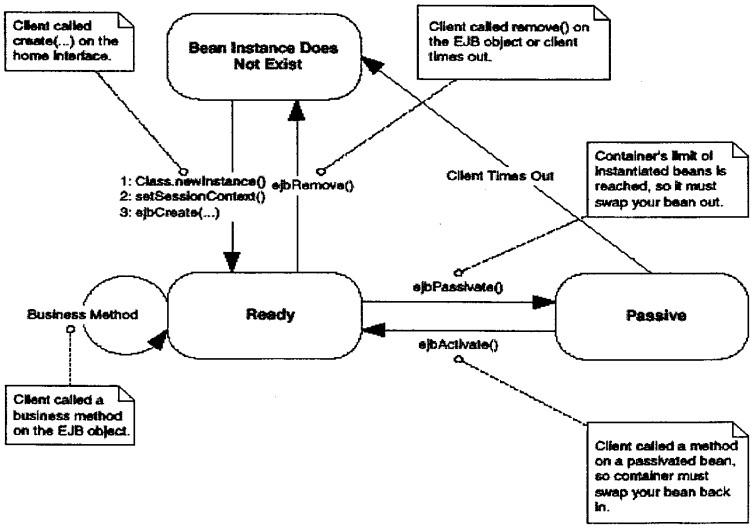
- 作為多個步驟的工作流程(workflow)及企業流程(business process)的管理。
- 取代 Web Server 的 HttpSession 物件，提供另一種儲存資料的方法。

在實作時，stateless 及 stateful session bean 的程式寫法實際上差不多，最大的差別在於 EJB Container 對待他們的方式截然不同。依據 EJB 2.0 的規格，只要在部署 EJB 元件時，作適當的宣告即可。其在 EJB Container 內的生命週期如圖四、圖五所示。



圖表 四、The life cycle of a stateless session bean

資料來源：Mastering EJB II, [5]



圖表 五、The life cycle of a stateful session bean

資料來源：Mastering EJB II, [5]

第四節 Entity Bean 介紹

Entity bean 具有『永久性(persistent)』的特性，也就是說不管 entity bean 目前是否正在記憶體中執行，它所處理的資料都會永久存在於資料庫或檔案或其它的儲存裝置中，其行為特性如下[1]：

- they are representations of persistent data and therefore survive a crash，
- multiple clients can be using entity beans that represent the same data，
- they are synchronous，
- the EJB manages an in-memory "copy" of the data。

EJB 2.0 規格定義的 entity bean 有兩種：BMP(Bean-Managed Persistence)

entity bean，及 CMP(Container-Managed Persistence) entity bean。BMP，顧名思義就是 entity bean 本身必須有程式碼負責處理 data persistence 的邏輯，因此，能夠允許開發者在其中加入 debug 的程式碼，相對的，也能夠處理較複雜的 data persistence 邏輯；CMP 顧名思義就是 data persistence 的邏輯完全交由 EJB Container 來處理，因此 EJB Container 能作一些 Caching 的機制，來提高其效能，而開發者僅需要再部署 CMP 時，作適當的宣告即可，因此相當容易開發。

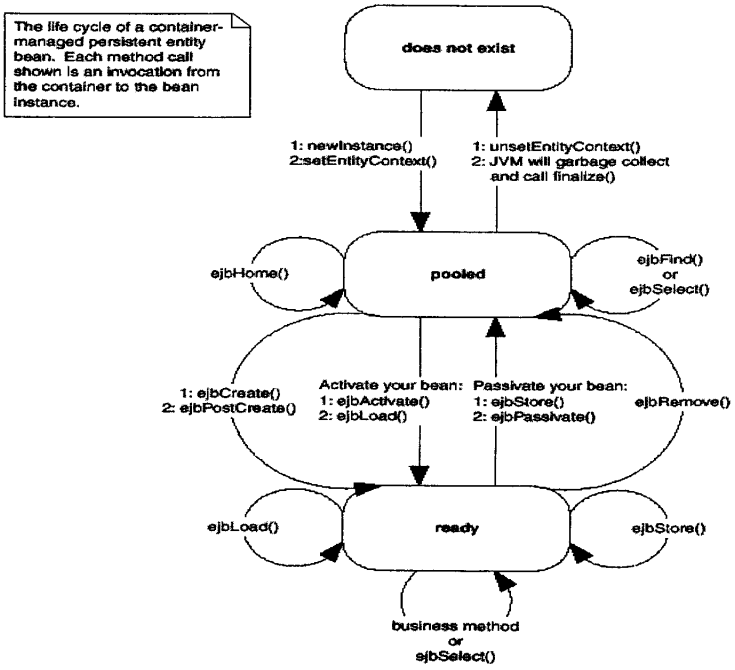
實務上，我們通常使用資料庫作為資料的儲存裝置，而 entity bean 在定義上，也具有一些資料庫的特性，譬如說，它一定要定義 primary key 的欄位。因此，在開發上，最簡單的方式就是做 1-to-1 object-relation mapping，也就是說，用一個 entity bean 去對應資料庫中的一個 table，當然，EJB 2.0 也允許我們用一個 entity bean 去對應資料庫中的多個 tables，尤其當資料庫的設計採用正規化的觀念時，table 和 table 之間的關係有時是密不可分的，必須設計一個 entity bean 去操作他們。

相較於 EJB 1.1 的規格，EJB 2.0 除了加強 entity bean 和資料庫 table 的關係對應外，另外也定義了一套查詢語言，稱之為 EJB-QL，其語法類似一般的資料庫查詢語言，方便開發者在部署 entity bean 時，只要作適當的宣告，就可以從資料庫中擷取資料，而不需撰寫操作資料庫的程式碼，而且，當透過 entity bean 去查詢資料時，也可以感受到執行效能有大大的改善。此外，EJB 2.0 也為 entity bean 新增一個 Local Interface 的介面，來改變過去 EJB Container 呼叫 entity bean 的方式，提高執行效能。

因為要操作資料庫，所以一定會有 transaction management 的問題，對 entity bean 而言，不論是 BMP 或 CMP，一律交由 EJB Container 來管理，也就是所謂

的 Container-Managed Transaction。這一點和 session bean 不同，在 session bean 中，是允許開發者自行撰寫程式碼去控制 transaction 的，也就是所謂的 Bean-Managed Transaction。當 entity bean 丟出系統層級的例外時(throw system exception)，EJB Container 最後會自動 roll back transaction，開發者不需去擔心資料 roll back 及 commit 的問題。

Entity bean 的生命週期也是由 EJB Container 來管理，總共有三個狀態，如圖六：(1) does not exist、(2) polled、(3) ready。第一個和第三個狀態，顧名思義我們及可以了解，至於第二種狀態，我們可以把它想像成是一個『bare naked』物件，也就是說，在這個狀態下，它並未和資料庫中的資料結合在一起，它雖然已經存在於記憶體，但卻尚未被賦予一識別標記，例如 primary key，而且它的欄位資料也尚未被初始化。



圖表 六、The life cycle of a CMP entity bean

資料來源：Mastering EJB II, [5]

第五節 Message-Driven Bean 介紹

訊息(message)是一種溝通的資訊，而 messaging，則是一種溝通的技術，它提供一種非同步的概念，一般我們用 TCP/IP Sockets 或 shared memory 來作為程式和程式間溝通的技術。由於是非同步，一般我們可以對系統的資源作較佳的管理，也能提高系統的執行效能，甚至可以對 message 的處理，作優先權的管理。在 Java 的標準裡，定義了一組 API，提供 Java Message Service(JMS) ，而在 EJB 2.0 規格，則把 Message-Driven Bean 納入，成為 EJB Server 所必須提供的基本配備。因此，符合 EJB 2.0 規格的 EJB Server，必須提供像這樣的 message-oriented middleware，讓 EJB application 可以和其它的 EJB application 作訊息的交換。

在 JMS 的定義裡，所謂的 JMS clients 指的是我們所撰寫的 Java 程式，這些 Java 程式會產生 Java message 或接收並處理 message；所謂的 destination，指的是 JMS Server 上的某一塊位置，用來儲存 Java 程式產生 message，並等待其它的 message consumer 程式來處理。同時，JMS 的定義也包含兩種應用模式：point-to-point 及 publish/subscribe。在 point-to-point 的應用模式下[1]：

- multiple producers can place messages onto the queue，
- multiple receivers can take messages off the queue，
- the messages come off in a first-in-first-out order，
- a message can only be delivered to a single receiver。

這些特性很像客服中心的應用，同時會有很多電話打進來，同時也有很多客服人員可以在線上服務，而電話分派系統會自動將電話轉給目前空閒的客服人員。而在 publish/subscribe 的應用模式下[1]：

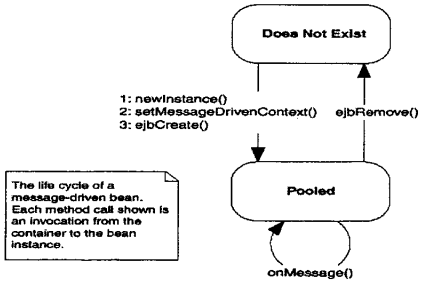
- publishers publish message to a topic and no longer have to worry about the number of subscribers，

- publishers do not have to worry about the delivery of guaranteed messages，
- messages can be delivered to multiple subscribers。

這些特性很像即時報價系統的應用，subscriber 可以登記自己有興趣的標的物，當標的物的價格有變動時，馬上可以收到訊息。

而 Message-Driven Bean 呢？它是 EJB 2.0 所新定義的 EJB 元件，扮演一個訊息接收者和處理者的角色，其特性如下[1][5]：

- consume message from Queues or Topics，
- are not visible to the client，therefore they do not have home or remote interface，
- execute as stateless services，
- are controlled by the container，其生命週期如圖七所示，
- have a single, weakly typed business method，
- do not have any returned value，
- cannot send exceptions back to the client，
- can be durable or nondurable subscribers，所謂的 durable subscriber，是指這個 subscriber 一定可以收到所有要傳送給他的訊息，當他暫時沒辦法去接收訊息時，JMS Server 會為它保留訊息，等到它可以接收時再傳送給他，
- can also send messages。



圖表 七、The life cycle of a message-driven bean

資料來源：Mastering EJB II, [5]

第六節 Transaction 管理

Transaction 是一種將一連串的資料操作視為一個資料操作的技術，這一連串的資料操作要不就全部成功，要不就全部失敗。一般，我們要達到這樣的目的，就是把這一連串的資料操作用一對 BEGIN TRANSACTION 及 END TRANSACTION 包起來，並寫在同一個 function 之中，但是在 EJB 的環境下，這一連串的資料操作可能散佈在不同的 EJB 元件中，甚至操作不同的資料庫，因此 EJB Container/Server 必須有能力對這些 EJB 元件作 transaction 的管理。本節將介紹 transaction 的特性及 EJB Container/ Server 如何提供 transaction 管理的機制。

Transaction 具有所謂的 ACID 的特性[1]：

- ***Atomicity*** – All or nothing; all operations involved in the transaction are implemented or none are；
- ***Consistency*** – The database must be modified from one consistent state to another. In the event the system or database fails during the transaction, the original state is restored (rolled back) ；
- ***Isolation*** – An executing transaction is isolated from other executing transaction in turns of the database records it is accessing；
- ***Durability*** – Once a transaction is committed it can be restored back to this state in the event of a system or database failure。

當一個 transaction 中所包含的資料操作必須橫跨多個資料庫時，我們稱之為 distributed transaction，以下，先介紹一些名詞的意義[1]：

- ***Resource Manager (RM)*** – A resource, like a database, is controlled through software called resource manager，

- ***Local Transaction*** – A transaction that deals with a single resource manager，
- ***Distributed Transaction*** – A transaction that coordinates or spans multiple resource managers，
- ***Transaction Manager (TM)*** – A coordinator of multiple resource manager，
- ***Two-Phase Commit Protocol*** – A method of coordinating a single transaction across two or more resource manager. It guarantees data integrity by ensuring that transactional updates are committed in all of the participating resources, or are fully rolled back out of all resources, reverting to the state prior to the start of the transaction。

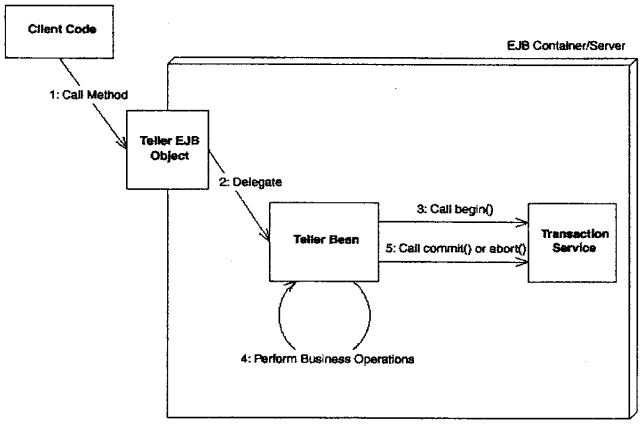
在 EJB 2.0 的規格裡，EJB Container/Server 是必須支援 distributed transaction 的，也就是說，必須能扮演好一個 ***Transaction Manager*** 的角色，但前提是，所使用的 ***Resource Manager*** 必須支援 XA(Extended Architecture)的架構。一般，我們常使用的資料庫，如 Oracle、SyBase、Informix、DB2 等等，均提供支援 XA 架構的驅動程式(JDBC Driver) 。因此，所有的 EJB 元件(Stateless/Stateful Session Bean、Entity Bean 、Message-Driven Bean)在本質上都是能夠參予一個 transaction 的，如書上[5]所描述的：

> 『*This means they can fully leverage the ACID properties to perform reliable, robust, server-side operations. Thus, enterprise bean are ideal modules for performing mission-critical tasks.* 』

在開發 EJB 元件時，事實上撰寫的程式碼不會直接跟 ***Transaction Manager*** 或 ***Resource Manager*** 有所接觸，整個 transaction 的管理都交由 EJB Container 去管控了。由 EJB Container 的角度看 transaction，可分成三種不同的情境：programmatic transaction、declarative transaction 及 client-initiated transaction，主

要的分別在於引發 transaction 的『人』不同。

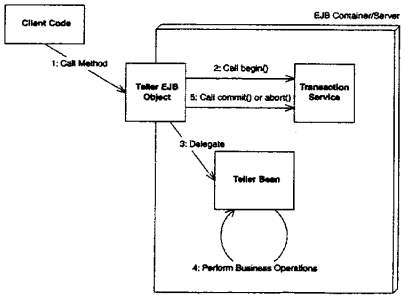
- Programmatic transaction – 現存的系統，程式的寫法多屬於此類。EJB 元件開發者必須自行處理 transaction 的邏輯，決定什麼條件下可以 commit，什麼條件下必須 roll back，其流程如圖八所示；



圖表 八、Bean with programmatic transaction

資料來源：Mastering EJB II, [5]

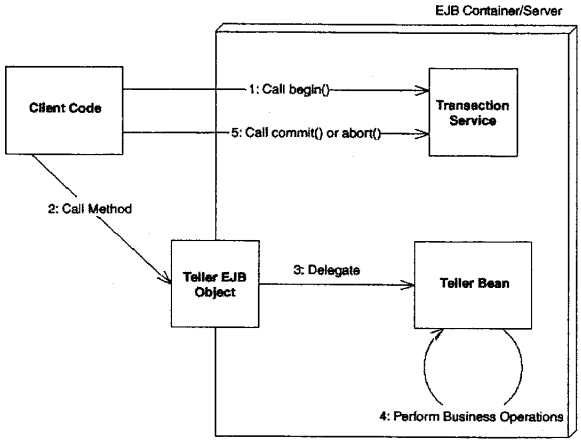
- Declarative transaction - EJB 元件開發者只需專注於商業邏輯的開發，而在部署 EJB 元件時利用宣告的方式，EJB Container 即可是需要將此元件納入 transaction 管理，其流程如圖九所示；



圖表 九、Bean with declarative transaction

資料來源：Mastering EJB II, [5]

- Client-initiated transaction – Transaction 是由呼叫者所啟動的，其流程如圖十所示；



圖表 十、Bean with client-initiated transaction

資料來源：Mastering EJB II, [5]

EJB Container 在管理 transaction 時，提供六種不同的 transaction 類別供部署 EJB 元件時使用，對 EJB 元件中所 implement 的每一個 method，我們都可以去設定其 transaction 類別：

- Required – 這是一個相當常用的選項，一般，如果我們希望一個 method 是在有 transaction 的環境下執行時，就把它設定成這一個類別。當這樣的一個 method 被執行時，若已經是在一個有 transaction 的環境下，則 EJB Container 會把它放入已經存在的 transaction 環境下執行；若 transaction 的環境不存在，則 EJB Container 會啟動一個新的 transaction，然後在執行這個 method；
- Require-New – 當 EJB 元件的 method 被呼叫時，EJB Container 永遠會啟動一個新的 transaction 環境來執行這個 method，而不管原先是否已存在其它的 transaction 環境；

- Supports -當 EJB 元件的 method 被呼叫時，如果呼叫者已經啟動了一個 transaction 環境，則 EJB Container 會把它放入這個 transaction 環境下執行；若呼叫者並沒有在一個 transaction 環境下執行，則這個 method 也就不在任何的 transaction 環境下執行；
- Mandatory –當 EJB 元件的 method 被呼叫時，強制要在已經存在的 transaction 環境下執行，若呼叫者沒有啟動一個 transaction 環境，則 EJB Container 會不允許此 method 得執行，並產生一 exception；
- Not-Supported -這也是一個相當常用的選項，譬如當 EJB 元件的 method 只是用來查詢一些資料時。如果一個 method 是不需在有 transaction 的環境下執行時，就把它設定成這一個類別；
- Never – 相對於 Mandatory，當 EJB 元件的 method 被呼叫時，強制不可以在 transaction 環境下執行，若呼叫者已經啟動一個 transaction 環境，則 EJB Container 會不允許此 method 執行，並產生一 exception。

雖然，使用 programmatic transaction 可以獲得最大的彈性，意即 EJB 元件的開發者可以完全掌控 transaction 的範圍，但一般而言，我們使用 container-managed transaction 就足夠了，意即使用 declarative 的方式，EJB 元件的開發者只要專注在企業邏輯的 implement 上，並妥善規劃每一個 method 的 transaction 類別，將 transaction 的管理放心交給 EJB container/server，便可以做出 *reliable, robust, scalable* 的元件。

第三章　研習心得及建議

此次奉派出國實習『電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術』，可以說是獲益匪淺。職於出國前即開始接觸 J2EE 的平台及 EJB 的技術，並運用在 IOIS/eTRIS 專案的開發上，但由於接觸的時間太短，經驗不足，且專案的時程緊迫，因此無法對整個 EJB 的規格有深入的了解，且當初 EJB 的規格為 EJB 1.1。經過這次出國實習的洗鍊，由於 BEA System, Inc. 本身參予 EJB 規格的制定，產品在市場上佔有領導地位，其人員亦具有豐富的實務經驗，經過他們深入淺出的全盤介紹 EJB 2.0 的特性後，讓職得以更清楚的一窺 EJB 的全貌。之前，職在觀念上仍有些模糊甚至混淆，經過對一些問題的反覆討論後，總算茅塞頓開，特別是在討論的過程，他們往往能抓住重點，在關鍵處給予一針見血的答案。

目前，J2EE 的技術已臻成熟，市場上也是百家爭鳴，同時，Java 社群標準討論會(Java Community Process, JSP)的組織[7]，也開始重視電信的應用範疇，已經著手討論有關 OSS/BSS 的標準，希望對 OSS 的整合及部署制定一套一致性、相容性的 API[8][9]，宣揚用 J2EE 作為電信 OSS 的開發平台，發展元件化的 OSS 解決方案。OSS/J 目前的成員有 BEA Systems, Inc.、Sun、Borland、Telecordia Technologies、Ericsson、Nokia、Nortel、NEC 等等，包含了 J2EE 領域與電信界的領導大廠，共同為『OSS through Java』的目標來努力，期望將 network equipment providers (NEPs)，independent software vendors (ISVs)，enterprise middleware vendors (EMVs)，和 service providers (SPs)集合在一起，在合作的氣氛下，共同制定 OSS 解決方案的標準。

中華電信公司，為國內電信服務業的龍頭，而中華電信研究所更肩負著開

發與整合電信維運支援系統(OSS)與營運支援系統(BSS)的重責大任，值得投入人力，去評估 J2EE 平台的技術及未來的發展，密切了解 OSS/J 的發展趨勢，然後將一些新的觀念，導入公司內部，應用在系統的開發與整合上。

致謝

承蒙 860 詹 專案副經理凱統推薦，梁 專案經理冠雄遴派，出國實習『電信維運支援系統(OSS)系統整合及基礎營運支援服務系統運用之最新應用技術』，除由衷感謝外，並希望研習的心得能對專案的開發與企業資訊系統的整合有所幫助。

參考資料

1. Course books of "Developing Business Logic Components with Enterprise JavaBeans using BEA Weblogic Server 7.0", BEA Systems, Inc., 2002.
2. "Designing Enterprise Applications with the J2EE™ Platform, Second Edition", http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/DEA2eTOC.html
3. "enhanced Telecommunication Operations Map™ (eTOM)", TM Forum GB921, June 2002.
4. "Enterprise JavaBeans™ 2.0 Specification Final Release", http://java.sun.com/products/ejb/docs.html.
5. Ed Roman, Scott Ambler, Tyler Jewell, "Mastering Enterprise JavaBeans, Second Edition", Wiley Computer Publishing, John Wiley & Sons, Inc., 2002.
6. Floyd Marinescu, "EJB Design Patterns", Wiley Computer Publishing, John Wiley & Sons, Inc., 2002.
7. JCP Web site, http://jcp.org.
8. OSS/J Web site, http://java.sun.com/products/oss .
9. OSS/J API Web site, http://jcp.org/jsr/tech/oss.jsp .