การพัฒนาโปรแกรมระบบงานเชิงวัตถุโดยการใช้ด้วยภาษาจาวา 1

OBJECT ORIENTED SYSTEM DEVELOPMENT USING JAVA 1

นางสาวมนธิชา คงทวี

นางสาวอุทัยรัตน์ ดอกกุหลาบ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การพัฒนาโปรแกรมระบบงานเชิงวัตถุโดยการใช้ด้วยภาษาจาวา 1

OBJECT ORIENTED SYSTEM DEVELOPMENT USING JAVA 1

โดย

นางสาวมนธิชา คงทวี

นางสาวอุทัยรัตน์ ดอกกุหลาบ

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมระบบงานเชิงวัตถุโดยการใช้ด้วยภาษาจาวา 1

OBJECT ORIENTED SYSTEM DEVELOPMENT USING JAVA 1

ผู้จัดทำ

1. นางสาวมนธิชา คงทวี รหัสประจำตัว 40010583
2. นางสาวอุทัยรัตน์ ดอกกุหลาบ รหัสประจำตัว 40011019

อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การพัฒนาโปรแกรมระบบงานเชิงวัตถุโดยการใช้ด้วยภาษาจาวา 1

นางสาวมนธิชา คงทวี 40010583
นางสาวอุทัยรัตน์ ดอกกุหลาบ 40011019
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ภาษาจาวา (Java) เป็นภาษาหนึ่งที่มีลักษณะการโปรแกรมเป็นแบบเชิงวัตถุ (Object Oriented Programing,OOP) ซึ่งมีจุดเด่นมากตรงที่เราทำการเขียนโปรแกรมขึ้นมาเพียงครั้งเดียว แต่สามารถนำโปรแกรมนั้นไปใช้งานได้ในทุกระบบปฏิบัติการดังคำกล่าวที่ว่า " Write once run anywhere" และ จาวายังประกอบด้วยส่วนประกอบที่ใช้งานในระดับคอมพิวเตอร์สมัยใหม่ เช่น ส่วนของกราฟฟิกและระบบการติดต่อผู้ใช้ด้วยกราฟฟิกส์ (GUI) ส่วนประกอบของระบบเครือข่ายและ ระบบติดต่อกับฐานข้อมูล ด้วยเหตุนี้เองในปัจจุบันจึงมีคนนิยมใช้ภาษาจาวามากขึ้นเรื่อยๆทั่วโลก

ในปัจจุบันภาษาจาวาได้มีการใช้งาน Enterprise JavaBeans (EJB) ซึ่งมีสถาปัตยกรรมแบบคอมโพเน็นต์ที่สามารถติดต่อดาต้าเบส (Database) ข้ามเซิร์ฟเวอร์ (Server) ได้ดีเหมาะสำหรับนำมาใช้กับงานมิดเดิลแวร์ (Middleware) ได้ดีมาก นอกจากนี้ยังมีมาตรฐานที่ใช้สนับสนุนการใช้งานด้านมัลติเทียร์ โปรแกรมมิ่ง โมเดล (Multitiered programming model) ซึ่งก็คือเทคโนโลยีของ Java 2 Platform, Enterprise Edition (J2EE) ซึ่งได้ใช้ประโยชน์จาก Java 2 Platform, Standard Edition ซึ่งทำให้มีความเสถียร (stable) และความมั่นคงปลอดภัยมากยิ่งขึ้น

การใช้งาน J2EE นั้นสามารถใช้ได้บนหลายแพลตฟอร์ม (platform) ทำให้แอพพลิเคชันระดับเอ็นเตอร์ไพรส์ (Enterprise-class Application) สามารถทำงาน ณ ที่ใดก็ได้ และทำให้โปรแกรมมิ่งโมเดล (Programming model) สามารถศึกษาเข้าใจได้ง่ายขึ้น นอกจากนี้ J2EE ยังสนับสนุนเทคโนโลยีระดับเอ็นเตอร์ไพรส์ซึ่งมี EJB ประกอบอยู่ด้วย

Object Oriented System Development Using Java 1

Miss.Monticha Kongthawee 40010583

Miss.Uthairuth Dokkularb 40011019

Dr. Worawat Limpoka Advisor

ABSTRACT

Java language is an Object Oriented Programming language. It's main advantage are portability and hardware independency as the saying " Write once run anywhere" . Java has many components that use in the new computer programming such as Graphic User Interface (GUI) component, Network component and Database component. Many programmers in the world use Java.

Today ,Java has the Enterprise JavaBeans (EJB) which is a component architecture that allows for connect to the database across the server so that it good to use in middleware. Beside this it also has the Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications that takes advantage of many features of the Java 2 Platform, Standard Edition such as stability and security .

J2EE provides a platform that will allow enterprise-class application the ability to run anywhere and a single easy-to-learn blueprint programming model for J2EE. J2EE is a collection of enterprise technologies, of which EJB is an integral part. By understanding and using J2EE properly, you can build portable, object-oriented, enterprise-class applications in Java.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

มนธิชา คงทวี

อุทัยรัตน์ ดอกกุหลาบ

สารบัญ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันพัฒนาการทางด้านสถาปัตยกรรมคอมโพเนนต์บนฝั่งเซิร์ฟเวอร์ที่สามารถทำให้การสร้างแอพพลิเคชั่น(application)ในระดับเอนเตอร์ไพซ์ทำได้ง่ายขึ้นนอกจากนี้ยังทำให้แอพพลิเคชั่นมีความน่าเชื่อถือ (Reliable) มีระบบรักษาความปลอดภัย (Secure) มีความถูกต้องมากขึ้นโดยไม่ต้องเขียนโครงสร้างระบบที่มีความซับซ้อนทั้งหมดเอง ทำให้มีการพัฒนาผลิตภัณฑ์จากสถาปัตยกรรมบนฝั่งเซิร์ฟเวอร์ (Server-Side Component Architectuer Solutions) ออกโดยค่ายต่างๆดังนี้ เช่น

- Microsoft's Distributed internet Application Architecture (DNA)
- Sun Microsystem's Java 2 Platform,Enterprise Edition (J2EE)
- The Object Management Group's COBRA Standard

สำหรับในโครงงานที่จัดทำขึ้นนี้จะศึกษาและอิมพลีเมนต์ (implement)เฉพาะผลิตภัณฑ์ของค่าย SUN ซึ่งก็คือ Sun Microsystem's Java 2 Platform,Enterprise Edition หรือ J2EE โดยเป็นผลิตภัณฑ์ที่เกิดจากการได้ตระหนักถึงความจำเป็นและความสามารถของสถาปัตยกรรมแบบคอมโพเนนต์บนฝั่งซิร์ฟเวอร์ (Server -Side Component Architecture) ซึ่งจะต้องมีการองรับในการทำงานแบบ N เทียร์ที่มีจาวาเป็นฐานและมีลักษณะของการ โปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งนับว่าได้ก้าวหน้ามาก และนับวันแนวโน้มของการใช้งานจะมีเพิ่มมากขึ้นเรื่อย ๆ และจริง ๆ แล้วโดยตัวภาษาจาวาเองเหมาะสำหรับการใช้งานบนเซิร์ฟเวอร์วาจา ส่วนการใช้งานที่ฝั่งของไคลเอ็นต์จะมีปัญหาอยู่หลายเรื่อง เช่น แพลตฟอร์ม (platform) ของ User Interface เข้ากันไม่ได้ หรือ User Interface ช้า รวมไปถึงเรื่องของ Java Virtual Machine (JVM) ที่รันบน Client Machine ผิดเวอร์ชันกัน แต่สิ่งเหล่านี้จะไม่เป็นปัญหากับฝั่งเซิร์ฟเวอร์เพราะการ deploy บนฝั่งเซิร์ฟเวอร์จะอยู่ภายใต้การควบคุมของเอ็นไวรอนเมนต์ (Environment)

ภาษาจาวาจะเป็นภาษาที่สะดวกในการเขียนคอมโพเนนต์ใส่ไว้บนฝั่งเซิร์ฟเวอร์เพราะว่าผลิตภัณฑ์ของฝั่งเซิร์ฟเวอร์จะมียูนิกซ์ (UNIX) และเมนเฟรม (Mainframe) เป็นแกนหลักซึ่งหมายความว่าภาษาที่สามารถทำงานข้ามแพลตฟอร์มได้จะมีความสำคัญมากเพราะผู้พัฒนาจะต้องเขียนคอมโพเนนต์เพียงครั้งเดียวแต่สามารถนำมาใช้งานได้บนหลายเอ็นไวรอนเมนต์

ในการตอบสนองของจาวาบนฝั่งเซิร์ฟเวอร์นั้นบริษัท ซัน-ไมโครซิสเต็มส์ (Sun Microsystem) จึงได้ออกแพลตฟอร์มที่ชื่อว่า "Java 2 Platform, Enterprise Edition (J2EE)" โดย J2EE เป็นแพลตฟอร์มที่มีความอิสระ, สามารถนำไปใช้ได้อย่างมีประสิทธิภาพ, สนับสนุนกับผู้ใช้ได้ทีละหลาย ๆ คน (Multi-user), มีระบบรักษาความปลอดภัย (security) โดยแกนหลักในเทคโนโลยีของ J2EE นี้คือ Enterprise Javabeans หรือ EJB ซึ่งเป็นมาตรฐานในการสร้างคอมโพเนนต์บนฝั่งเซิร์ฟเวอร์โดยการใช้ภาษาจาวา

เทคโนโลยีของ J2EE (J2EE Technologies) ที่ได้ทำการศึกษาในโครงงานนั้นประกอบไปด้วยส่วนต่างๆดังนี้

- เอนเตอร์ไพซ์ จาวาบีนส์ หรือ Enterprise JavaBeans (EJB)
- ข้อกำหนดทางด้านการสื่อสารระหว่างเครื่องคอมพิวเตอร์ในรูปแบบของ Java Remote Method Invocation (RMI)
- การเชื่อมต่อกับฐานข้อมูลด้วยภาษาจาวาหรือ Java Database Connectivity (JDBC)
- มาตรฐานที่ใช้ในการติดต่อกับออบเจ็กต์ที่ทำงานร่วมกันภายใต้ระบบเครือข่ายหรือ Java Naming and Directories Interface (JNDI)
- การจัดการกับทรานส์แอ็กชันหรือ Java Transaction APIs (JTA), Java Transaction Services (JTS)

โดยในส่วนของแอพพลิเคชัน (application) ในโครงงานนั้นจะใช้ภาษาจาวา (Java) ในการอิมพลีเม้นท์ (implement) เนื่องจากภาษาจาวาเป็นหนึ่งในภาษาที่เป็นลักษณะเชิงวัตถุหรือออปเจ็กโอเรียนเต็ดโปรแกรมมิ่ง (OOP) ซึ่งคือ มุมมองในการออกแบบซอฟต์แวร์ (software) ที่มองเอนติตี้ (entity) ต่างๆในระบบเป็นออบเจ็กต์ซึ่งมีคุณสมบัติและความสามารถเฉพาะตัว ที่สามารถทำหน้าที่ของแต่ละออบเจ็กต์นั้นได้ เพื่อที่จะทำงานร่วมกันให้บรรลุจุดประสงค์ของระบบที่ตั้งไว้

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อให้สามารถวิเคราะห์ ออกแบบ และพัฒนาระบบงาน ตามแนวคิดของออบเจ็กต์

1.2.2 ศึกษาการพัฒนาโดยใช้สถาปัตยกรรมแบบ Server Side Component

1.2.3 ศึกษาการสร้างแอพพลิเคชันด้วยการใช้สถาปัตยกรรมแบบ Server Side Component ว่าจะต้องมีการออกแบบในส่วนใดบ้างและจะสามารถนำไปอิมพลีเมนต์ได้อย่างไร ซึ่งอาจจะนำความรู้ที่ได้รับนี้ไปประยุกต์ใช้ในงานต่างๆ ได้

1.2.4 เพื่อเป็นการนำเสนอแนวคิดและผลการปฏิบัติงานในการสร้างระบบด้วยการใช้สถาปัตยกรรมแบบ Server Side Component ซึ่งมีเอนเตอร์ไพซ์ จาวาบีนส์ Enterprise Javabeans เป็นซอฟต์แวร์ คอมโพเน้นต์ (software –component) บนฝั่งเซิร์ฟเวอร์ (server) ที่สามารถนำไปใช้งานใน distributed multi-tier environment ได้

1.3 ขอบเขตของโครงงาน

งานวิจัยนี้จะสร้างระบบการค้าเชิงอิเล็คทรอนิค (Electronic Commerce) ที่เกี่ยวกับการท่องเที่ยวขึ้นมา โดยการใช้ WebLogic Server และ Oracle Database ซึ่งมีการออกแบบส่วนติดต่อกับผู้ใช้ด้วย Browser ซึ่งเบื้องหลังนั้นจะมีการทำงานของ Enterprise JavaBeans ต่างๆ ที่บรรจุอยู่ใน EJB Container นอกจากนี้ยังมีส่วนการทำงานของ Java Servlets ซึ่งเป็นโปรแกรมที่ทำงานบนเครื่องคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์เป็นหลักโดย Java Servlets นั้นเป็นโปรแกรมที่สร้างจากภาษาจาวา และใช้ประมวลผลหรือทำงานบนเครื่องเซิร์ฟเวอร์ โดยจะทำงานควบคู่กับเครื่องของผู้ใช้ และเมื่อผู้ใช้ต้องการประมวลผล ก็สั่งให้เซิร์ฟเล็ต (servlet) ที่อยู่บนเซิร์ฟเวอร์ (Server) ทำงานได้จากเครื่องคอมพิวเตอร์ของผู้ใช้ แล้วส่งผล

ลัพธ์กลับมายังเครื่องผู้ใช้อีกทีซึ่งผลลัพธ์จากการทำงานนั้นจะอยู่ในรูปแบบของข้อมูลภาษา HTML โดยในงานวิจัยนี้จะต้องออกแบบสถาปัตยกรรมของระบบทั้งหมด

นอกจากนั้นในโครงงานนี้ยังถือว่าเป็นโครงการที่ทดลองสร้างแอพพลิเคชันเพื่อศึกษาความเป็นไปได้ในการใช้งานของระบบ ดังนั้นจึงต้องมีการจัดการด้านต่างๆ ของระบบ เช่น การจัดการด้าน Security, Reliability, Fail-over, Clustering และ Load balancing เพื่อให้มีการทำงานที่มีประสิทธิภาพและสามารถรองรับ user ปริมาณมาก ๆ ได้

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลักๆ คือ ภาษาจาวา (Java), สถาปัตยกรรมบนฝั่งเซิร์ฟเวอร์ (Server-Side Component Architecture) Enterprise JavaBeans, Enterprise Java Bean Container และ Web Container ซึ่งมีรายละเอียดดังในบทที่ 2, 3 ,4 และ 5 จากนั้นในบทที่ 6 จะเป็นการศึกษาเรื่องการติดต่อจากไคลเอนต์ และบทที่ 7 จะกล่าวถึงการจัดการด้านฐานข้อมูล เมื่อศึกษาถึงระบบทั้ง 3 เทียร์แล้ว จึงทำการศึกษาด้านความปลอดภัยในการใช้งานระบบจากไคลเอนต์เพื่อให้ระบบมีเสถียรภาพและมีความน่าเชื่อถือ รายละเอียดดังกล่าวจะถูกนำเสนอในบทที่ 8

จากความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบสถาปัตยกรรมของระบบจากนั้นจึงนำมาสร้างเป็นแอพพลิเคชัน ซึ่งมีรายละเอียดในบทที่ 9 และ บทที่ 10 ซึ่งจะกล่าวถึงองค์ประกอบโดยรวมของระบบที่พัฒนาขึ้นมาทั้งหมด และยังอธิบายไปถึงรูปแบบการติดต่อกับผู้ใช้ และการประมวลผลเบื้องต้น จากนั้นจะอธิบายการทำงานแต่ละส่วนของระบบซึ่งรวมทั้งการนำข้อมูลเข้าและออกจากฐานข้อมูลเพื่อแสดงผลต่อผู้ใช้ และบทที่ 11 ซึ่งเป็นบทสุดท้ายจากจะทำการวิเคราะห์และสรุปผลการทำงาน ผลที่ได้รับจากงานวิจัยชิ้นนี้ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
ภาษาจาวา

2.1 การเขียนโปรแกรมแบบอ้างอิงวัตถุ (Object Oriented Programming)

การเขียนโปรแกรมแบบอ้างอิงวัตถุ (OOP) คือ มุมมองในการออกแบบ software ที่มอง entity ต่างๆในระบบเป็น object ซึ่งมีคุณสมบัติและความสามารถเฉพาะตัว ที่สามารถทำหน้าที่ของแต่ละ object นั้นได้ เพื่อที่จะทำงานร่วมกันให้บรรลุจุดประสงค์ของระบบที่ตั้งไว้ ส่วน Object นั้นคือการรวบรวมข้อมูลและฟังก์ชันที่ดำเนินการต่อข้อมูลนั้นเข้าไว้ด้วยกันเป็นหน่วยเดียวกัน โดยในการทำงานร่วมกันของ object แต่ละ object นั้นจะสื่อสารกันโดยส่งข้อมูลข่าวสารจาก object หนึ่งไปยังอีก object หนึ่งได้โดยใช้การส่ง message

การเขียนโปรแกรมแบบ OOP ต่างจากการเขียนโปรแกรมแบบโครงสร้าง (Structure) พอสมควร เนื่องจากแบบ OOP ผู้เขียนต้องมองโปรแกรมออกเป็นชิ้นส่วนย่อยๆที่มีที่เก็บข้อมูล มีการทำงานภายในชิ้นส่วนย่อยๆให้ได้ แล้วนำเอาชิ้นส่วนย่อยๆเหล่านั้นมาผสมปนเปกันในอัตราที่พอเหมาะ จนเกิดเป็นโปรแกรมขึ้นใช้งาน แต่ถ้าเป็นแบบโครงสร้าง แล้ว ผู้เขียนต้องมองโปรแกรมออกเป็นลักษณะการทำงานแบบที่การทำงานต้องมีรูปแบบที่แน่นอน เช่น แบบเรียงลำดับ แบบมีเงื่อนไข หรือแบบซ้ำไปมาแต่เราสามารถนำเอาความคิดแบบโครงสร้าง ไปใช้ร่วมกับแบบ OOP ได้ในการทำให้ชิ้นส่วนย่อยๆ (Object) ทำงานแบบมีโครงสร้างมากขึ้น

2.1.1 คุณสมบัติหลักของ Object-Oriented

1. Encapsulation คือ คุณสมบัติในการห่อหุ้มข้อมูลที่อยู่ภายใน object ด้วย method ที่เตรียมไว้เพื่อให้จัดการกับข้อมูลนั้น เพื่อป้องกันไม่ให้ภายนอกสามารถเข้าถึงข้อมูลโดยตรง ทำให้มีความปลอดภัยและง่ายต่อการ maintenance

2. Inheritance คือ คุณสมบัติในการสืบทอดคุณสมบัติและความสามารถของ object หนึ่งไปยังอีก object หนึ่ง เพื่อแทนที่จะประกาศ class หลาย ๆ class ที่มีคุณสมบัติและความสามารถบางส่วนเหมือนกัน เราสามารถประกาศ Parent class แล้วประกาศ class อื่นๆมาสืบทอดคุณสมบัติและความสามารถของ class นี้ได้ ทำให้ software มีความ Reusable

3. Polymorphism คือ คุณสมบัติในการที่มี method หลายๆ method ที่ชื่อเดียวกันแต่มีความสามารถต่างกัน ซึ่งการเลือกจะใช้ method ไหนก็ขึ้นอยู่กับ parameter ที่รับเข้ามา คุณสมบัตินี้ทำให้ software มีความ Flexible

2.1.2 คลาส (class)

คลาสคือ ต้นแบบ (Template) ของออบเจกต์ โดยจะประกอบไปด้วยองค์ประกอบพื้นฐานที่สำคัญสองส่วนคือ ดาต้าเมมเบอร์ และเมมเบอร์ ฟังก์ชัน

ดาต้า เมมเบอร์ (Data member)หรือ คุณลักษณะ (Attribute) หมายถึง ดาต้าของออบเจ็กต์นั้นๆ เป็นลักษณะเฉพาะตัวที่แสดงให้เห็นถึงความแตกต่างกันของวัตถุแต่ละชนิด เช่น สี,ขนาด,จำนวน ส่วนเมมเบอร์ ฟังก์ชัน (Member function)หรือเมธอด (Method) จะหมายถึง processing code ของออบเจ็กต์

เป็นส่วนที่จะบอกว่า คลาสนั้นทำอะไรได้บ้าง สามารถทำงานร่วมกับคลาสอื่นๆได้อย่างไร และคลาสอื่นๆสามารถสั่งงานคลาสนั้นได้อย่างไร

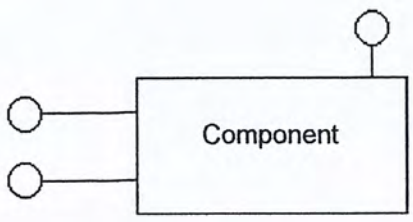
2.2 คอมโพเน้นท์ (Component)

การพัฒนาระบบงานในปัจจุบันมีหลายแนวทาง เช่น การพัฒนาระบบแบบโครงสร้าง การพัฒนาระบบแบบออบเจ็กต์โอเรียนเท็ต (Object Oriented) แต่ละแนวทางต่างเสนอวิธีการต่างๆ เพื่อทำให้การพัฒนาระบบเป็นขั้นตอน สามารถจัดระบบที่มีขนาดใหญ่และมีความซับซ้อน เพื่อให้ได้ระบบงานที่มีคุณภาพตรงกับความต้องการของผู้ใช้งาน

แต่อย่างไรก็ตามการพัฒนาระบบด้วยวิธีดังกล่าวก็ยังมีปัญหาต่างๆมากมาย ในกรณีของการพัฒนาระบบแบบโครงสร้างจะประสบปัญหาด้านโมเดลของระบบที่ไม่เชื่อมต่อกันตลอดวัฏจักรของการพัฒนา ในส่วนของการพัฒนาระบบแบบออบเจ็กต์โอเรียนเท็ตมีปัญหาที่เห็นได้ชัดคือ การที่ออบเจ็กต์ผูกติดกันมากเกินไป (Tightly Coupling) หากมีการแก้ไขออบเจ็กต์ใดอาจทำให้เกิดผลกระทบต่อออบเจ็กต์ลูกหลานได้ หรืออาจเกิดปัญหาด้านเวอร์ชันของซอฟต์แวร์ กล่าวคือ การเปลี่ยนแปลงไลบรารีเพื่อเพิ่มเติมฟังก์ชันใหม่จะทำให้ ซอฟต์แวร์เดิมอาจมีปัญหาในการทำงาน เป็นต้น

คอมโพเนนท์ออบเจ็กต์โมเดล (Component Object Model) เป็นมาตรฐานในการทำงานร่วมกันระหว่างออบเจ็กต์ที่ไม่ขึ้นกับภาษาที่ใช้พัฒนา โดยมีการติดต่อกันผ่านอินเตอร์เฟส (Interface) เท่านั้น อีกทั้งยังเพิ่มความสามารถในด้านการเรียกใช้แบบไม่ขึ้นกับรูปแบบ (Location Transparency) ทำให้ผู้พัฒนาสามารถเรียกใช้ออบเจ็กต์ได้ตั้งแต่ในรูปแบบพื้นฐาน คือ อยู่เครื่องเดียวกัน จนถึงในรูปแบบการเรียกใช้แบบข้ามเครื่อง จึงสามารถทำระบบแบบกระจาย (Distributed System) ซึ่งเป็นประโยชน์หลายประการ เช่นการทำ Load Balancing เพื่อเป็นการลดภาระของเครื่องเซิร์ฟเวอร์ และประเด็นที่สำคัญในออบเจ็กต์คือการนำกลับมาใช้ใหม่ (Reusability) ซึ่งจะช่วยแก้ปัญหาในการพัฒนาซอฟต์แวร์ต่างๆที่กล่าวมาแล้วได้

Component เป็นมาตรฐานในการทำงานร่วมกันระหว่างคอมโพเนนท์ ซึ่งมาตรฐานนี้ไม่ขึ้นกับตัวแอปพลิเคชันและภาษาที่ใช้พัฒนา โดยแอปพลิเคชันจะติดต่อกับแอปพลิเคชันอื่นๆ และกับระบบ โดยผ่านกลุ่มของฟังก์ชันหรืออินเตอร์เฟส ซึ่งอินเตอร์เฟส ก็คือ ข้อตกลงระหว่างซอฟต์แวร์คอมโพเนนท์กับผู้ใช้งาน ดังรูปที่ 2.1



รูปที่ 2.1 Component ซึ่งมี Interface 3 Interface

2.3 ภาษาจาวา

จาวาเป็นภาษาที่ใช้คอนเซ็ปของออบเจ็กต์โอเรียนเท็ต ในการเขียน หลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมคงจะทราบดีว่าออบเจ็กต์โอเรียนเท็ต สามารถลดความซับซ้อนโครงสร้างของโปรแกรม รวมไปถึงอำนวยความสะดวกในการรียูส(reuse)ส่วนประกอบต่าง ๆ ของโปรแกรมที่เขียนไว้แล้วเพียงใด นอกจากนี้จาวายังเป็นภาษาที่เป็น platform independent ซึ่งจะช่วยให้เราสามารถทำการพัฒนาระบบโดยใช้ environment อะไรก็ได้ซึ่งโดยทั่วไปมักจะเป็น Widows environment

ภาษาจาวาถูกพัฒนามาจากบริษัทซัน (Sun Microsystems) ซึ่งจัดให้เป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่ใช้หลักการออกแบบตัวภาษาด้วยวิธีเชิงวัตถุ และตัวภาษาถูกใช้เป็นเครื่องมือสำหรับพัฒนาโปรแกรมด้วยแนวคิดเชิงวัตถุ โดยตัวภาษามีลักษณะพิเศษดังนี้

Portability	สามารถในการใช้งานในสภาพวะแวดล้อมที่แตกต่าง โดยไม่ต้องมีการปรับแต่ง
Simple	ความง่ายในการเขียนโปรแกรม
Robust	ความคงสภาพในการทำงาน มีโอกาสเกิดข้อผิดพลาด ที่ไม่พึงประสงค์ได้น้อย
Secure	การรองรับมาตรฐานความปลอดภัยในการใช้งานรูปแบบต่างๆ
Distributed	มีความสามารถในการประมวลผลแบบกระจาย
Object-Oriented	มีหลักการของแนวคิดเชิงวัตถุ ในการสร้างโปรแกรม

ตารางที่ 2-1 แสดงลักษณะของภาษาจาวา

ลักษณะภาษาจาวา

การสร้าง : ต้องเขียนคำสั่งตามรูปแบบของภาษาจาวาแล้วเก็บไว้ในไฟล์ที่เรียกว่าซอร์สโค้ด จากนั้นนำไปคอมไฟล์ไป เป็นไฟล์คลาส

การใช้งาน : ถ้าภาษาจาวาที่ใช้สร้างไฟล์คลาสเขียนในรูปแบบจาวาแอปพลิเคชั่นให้ใช้งานบนเครื่องได้ทันทีผ่าน JVM เช่น ภาษาจาวาที่ใช้สร้างไฟล์คลาสเขียนในแบบจาวาแอปเพล็ต ก็ต้องสร้างเว็บมาเรียกใช้งานจาวาแอปเพล็ตอีกที (2 ขั้นตอน) และเวลาแอปเพล็ตทำงาน JVM ที่อยู่บนบราวเซอร์ก็จะรันแอปเพล็ตให้ทำงาน เป็นต้น

2.4 จาวาสคริปต์ (JavaScripts)

จาวาสคริปต์เป็นภาษาสคริปต์ที่เกิดขึ้นมาจากความนิยมในอินเตอร์เน็ต และเป็นหนึ่งในภาษาสำหรับอินเตอร์เน็ตที่มีการยอมรับ และได้รับความนิยมอย่างสูง ด้วยลักษณะของภาษาสคริปต์ที่ง่ายในการศึกษา เรียนรู้ และทำความเข้าใจ แต่ให้ศักยภาพและความสามารถเทียบเท่ากับภาษาระดับสูง

เว็บเพจที่เพิ่มความสามารถด้วยจาวาสคริปต์ สามารถสร้างความแตกต่างจากเว็บเพจปกติทั่วไป โดยในจาวาสคริปต์นั้นจะประกอบไปด้วยออบเจ็กต์ต่างๆที่สร้างไว้ในเอกสาร HTML เช่น วินโดว์ เฟรม ฟอร์ม ปุ่ม ภาพ ลิสต์บ็อกซ์ ปลั๊กอิน และ แอปเพล็ตที่ฝังตัวอยู่ ฯลฯ ซึ่งเราสามารถควบคุมออบเจ็กต์เหล่านี้ได้ดีกว่าการเขียนด้วยจาวาแอปเพล็ตเพราะเราสามารถโต้ตอบกับออบเจ็กต์บนเพจได้โดยตรง

การเขียนจาวาสคริปต์นั้นมีข้อดีที่สามารถใช้บนระบบปฏิบัติการใดๆก็ได้และยังสามารถกระโดดไปมาระหว่างระบบปฏิบัติการเหล่านี้ได้อีกด้วยซึ่งจะเป็นประโยชน์ในกรณีที่เราต้องการที่จะให้มีการทำงานของสคริปต์ภายใต้เบราเซอร์ในระบบปฏิบัติการต่าง ๆ โดยจาวาสคริปต์ทำงานได้ทั้งฝั่งไคลเอนต์ (สคริปต์จะทำงานหลังจากถูกโหลดลงหน่วยความจำในเครื่องของผู้ใช้แต่ละคน) และฝั่งเซิร์ฟเวอร์ (สคริปต์จะทำงานบนเว็บเซิร์ฟเวอร์ภายใต้สภาพแวดล้อม LiveWire ของ Netscape)

ถ้าจะเปรียบเทียบการใช้งานระหว่างจาวาสคริปต์กับภาษาจาวาแล้วอาจจะพิจารณาได้จาก เช่น ถ้าต้องการความง่ายและรวดเร็วก็ใช้จาวาสคริปต์ แต่ถ้าต้องการสร้างโปรแกรมที่ไม่ซับซ้อน และมีประ-สิทธิภาพก็จะใช้ภาษาจาวาซึ่งแน่นอนกว่า แต่ถึงอย่างไรมันก็มีหลายปัจจัยอื่นประกอบ เช่น เวลาในการใช้งานต้องการเร็วหรือช้า

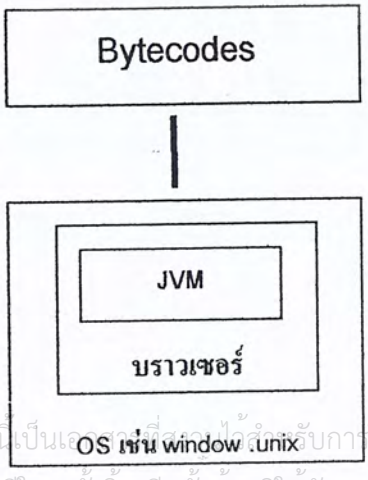
<u>ลักษณะของจาวาสคริปต์</u>

การสร้าง : ต้องเขียนร่วมกับภาษา HTML เพื่อใช้สร้างเว็บ

การใช้งาน : ต้องถูกโหลดใช้บนบราวเซอร์ และบราวเซอร์จะทำหน้าที่แปลคำสั่งจาวาสคริปต์ทีละคำสั่งเพื่อทำงานตามคำสั่ง

2.5 จาวาแอพเพล็ต (java Applet)

จาวาแอพเพล็ตคือโปรแกรมคอมพิวเตอร์ที่เขียนด้วยภาษาจาวา ที่ถูกแปลอยู่ในรูปของ Bytecodes ด้วยวิธีการคอมไพล์ และการทำงานของโปรแกรมจะทำงานผ่าน จาวาเวอร์ชวลแม็ชชีน (JVM) ที่อยู่บนบราวเซอร์ (Browser) เช่น Netscape, Internet Explorer, Opera ที่สนับสนุนการใช้งานจาวาแอพเพล็ต



รูปที่ 2-2 การแปลคำสั่งของของโปรแกรมที่เป็นแบบ Java Applet

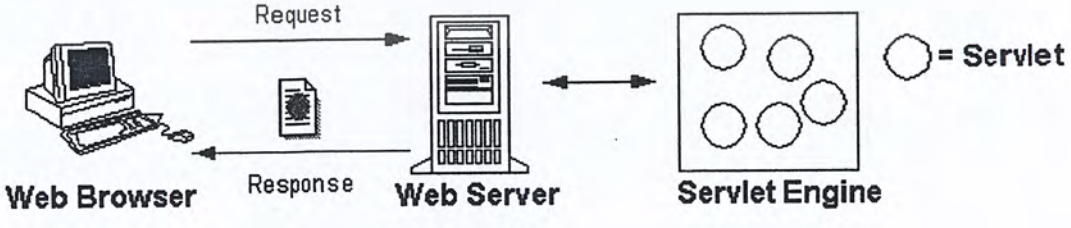
ข้อแตกต่างอีกประการของการสร้างแอพเพล็ตด้วยภาษาจาวา คือ แอพเพล็ตตัวนั้นต้องถูกสืบทอดมาจากคลาส Applet (ซึ่งอยู่ในรายละเอียดตอนเขียนโปรแกรม)

จาวาแอพเพล็ตยังคงคุณ สมบัติ Write Once, Run Anywhere เพราะยังทำงานผ่าน JVM บนบราวเซอร์ ซึ่งมีแพร่หลายอยู่ทุกๆแพลตฟอร์ม เช่น วินโดว์ แม็กอินทอส ยูนิกซ์ ลีนุกซ์ เป็นต้น

2.6 Javaservlet

javaservlet เป็นโปรแกรมที่ทำงานบนเครื่องคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์เป็นหลัก โดย javaservlet นั้นเป็นโปรแกรมที่สร้างจากภาษาจาวา และใช้ประมวลผลหรือทำงานบนเครื่องที่เรียกว่า server โดยจะทำงานควบคู่กับเครื่องของผู้ใช้ และเมื่อผู้ใช้ต้องการประมวลผล ก็สั่งให้ servlet ที่อยู่บน server ทำงานจากเครื่องคอมพิวเตอร์ของผู้ใช้ แล้วส่งผลลัพธ์กลับมายังเครื่องผู้ใช้อีกทีซึ่งผลลัพธ์จากการทำงานนั้นจะอยู่ในรูปแบบของข้อมูลภาษา HTML ทำให้เครื่องของผู้ใช้เป็นเครื่องที่ไม่ต้องมีประสิทธิภาพมากนัก ว่ากันไปแล้ว servlet จะช่วยให้ระบบคอมพิวเตอร์ขนาดใหญ่ทำงานได้ดีขึ้น โดยตัวแปรในการทำงานให้ดีขึ้นนั้นขึ้นอยู่กับประสิทธิภาพของตัวเซิร์ฟเวอร์เป็นหลัก ถ้าเซิร์ฟเวอร์ตัวใหญ่ก็ทำงานดี ถ้าตัวเล็กก็คงไม่เหมาะสมเท่าไหร่ นั่นคือ javaservlet นั้นต้องการที่จะใช้เครื่องเซิร์ฟเวอร์ที่มีประสิทธิภาพสูงมาคำนวน และส่งผลลัพธ์กลับไปที่ไคลแอนท์ ในขณะที่ Applet เป็นรูปแบบของการประมวลผล และแสดงผลที่ไคลแอนท์เอง (ไคลแอนท์ในที่นี้คือบราวเซอร์)

Servlet นั้นอ้างอิงหลักการของ CGI โดยข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาวานั่นเอง นอกจากนี้ Servlet ยังมีความเร็วที่เหนือกว่า CGI เพราะ Servlet ใช้หลักการของ thread โดยจะทำการสร้าง 1 thread ต่อหนึ่ง request ที่มาจาก client ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request ซึ่งจะทำให้เปลืองทรัพยากรมากกว่าและ process ในการรันก็จะช้ากว่าด้วย ถึงแม้ว่า Servlet จะอ้างอิงหลักการของ CGI อย่างไรก็ตามในการที่จะทำการรัน Servlet แล้ว ตัว web server จะไม่สามารถส่งข้อมูลไปให้ Servlet ได้โดยตรงเหมือนกับหลักการของ CGI แต่ตัว web server จะต้องเพิ่มอีกส่วนหนึ่งซึ่งเป็นส่วนที่ใช้เป็นเสมือนตัวห่อหุ้ม Servlet ต่าง ๆ ไว้โดยส่วนที่เพิ่มขึ้นมานี้เราเรียกว่า Servlet Engine หรือ Servlet Container



รูปที่ 2-3 Servlet Engine and its Servlets

2.7 จาวาบีนส์ (Javabeans)

จาวาบีนส์เป็นรูปแบบการใช้งานออปเจ็คที่ถูกสร้างไว้แล้ว และนำมาประกอบกันเป็นซอฟต์แวร์ ถ้าเทียบไปแล้วจะคล้ายกับ Active Control หรือ ActiveX ที่ใช้ใน VB หรือ VC แต่ Java Bean จะสร้างด้วยภาษาจาวา เพื่อให้ใช้งานได้ทุกๆ Platform ทำให้ Bean มีการสร้างเพื่อขายทางการค้าจากผู้ผลิต เช่น Third Party ด้วย นอกจากนี้จาวาบีนส์ยังมีลักษณะอื่นๆ อีก คือ

- เป็นคลาสในภาษาจาวาเพื่อเรียกใช้ได้ในโปรแกรมพัฒนาจำพวกวีชวล เช่น บีนเกี่ยวกับปุ่ม บีนเกี่ยวกับรูปภาพ บีนเกี่ยวกับเมนูเป็นต้น ถ้าเปรียบเทียบกับ vistual basic ก็คือสิ่งที่เรียกว่าคอนโทรลนั่นเอง
- เป็นส่วนประกอบทางกราฟฟิกที่สามารถนำมาประกอบกันเป็นโปรแกรม ที่ติดต่อกับผู้ใช้แบบ graphic user interface ได้ง่ายขึ้นสะดวกขึ้น เหมือนที่ใช้ใน visual basic, visual c++ หรือใน delphi เป็นต้น
- เป็นลักษณะคล้าย ๆ กล่องที่เราสามารถเปลี่ยนลักษณะการทำงานและคุณสมบัติต่าง ๆ ข้างในได้โดยเซ็ตที่ตัว property ของมัน (คล้าย ๆ พวก ActiveX ของ microsoft)

2.8 Enterprise JavaBeans (EJB)

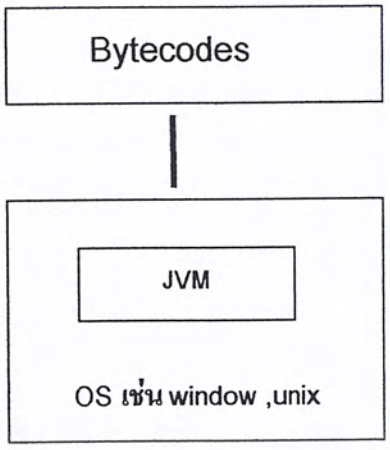
เป็นสถาปัตยกรรมคอมโพเนนท์บนฝั่งเซิร์ฟเวอร์ที่ทำให้การสร้าง application ในระดับเอนเตอร์ไพซ์ทำได้ง่ายขึ้นและมีความน่าเชื่อถือ (Reliable)เนื่องจากมีระบบรักษาความปลอดภัย (Secure)โดยไม่ต้องเขียนโครงสร้างระบบที่มีความซับซ้อนทั้งหมดเอง และ EJB จะถูกออกแบบให้สนับสนุนการนำแอพพลิเคชั่นเดิมมาใช้ใหม่ได้ (Reusability) EJB เป็นสถาปัตยกรรมคอมโพแนนท์ที่สามารถ deploy ได้โดยเราจะเรียกว่า "เอนเตอร์ไพซ์บีน" (Enterprise bean)

EJB เป็นแอพพลิเคชั่นคอมโพเนนท์ในระดับ coaser-grained สามารถ deploy เพียงลำพังหรือ deploy โดยรวมกับคอมโพเนนท์อื่นๆ (ประกอบกันขึ้นเป็นระบบแอพพลิเคชั่นที่มีขนาดใหญ่ขึ้น) โดยในการ deploy ต้องมีคอนเทนเนอร์ให้บริการ โดย Enterprise bean นั้นมีคอมโพเนนท์ของจาวาอยู่ 2 ชนิด ที่คล้ายมันมากคือ

1. Applet
2. Servlet

2.9 จาวาแอพพลิเคชัน (Java Application)

จาวาแอพพลิเคชันคือโปรแกรมคอมพิวเตอร์ที่เขียนด้วยภาษาจาวา ที่ถูกแปลอยู่ในรูปของ Bytecodes ด้วยวิธีการคอมไพล์ และการทำงานของโปรแกรมจะทำงานผ่าน จาวาเวอร์ชวลแม็ชชีน (JVM) ที่ทำหน้าที่แปลงคำสั่ง Bytecodes ให้อยู่ในรูปของคำสั่งที่สามารถประมวลผลได้ บนเครื่องคอมพิวเตอร์ที่ติดตั้ง OS ใดๆ



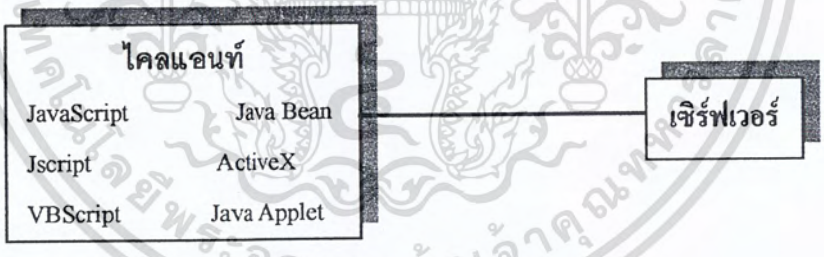
รูปที่ 2-4 การแปลคำสั่งของของโปรแกรมที่เป็นแบบ Java Application

การทำงานของจาวาแอพปลิเคชัน จึงเหมือนกับการทำงานของโปรแกรมทั่วไป ที่ถูกเขียนจากภาษาอื่นๆ เช่น C, VB, Delphi เป็นต้น ต่างกัน เพียงแต่จาวาแอพปลิเคชันต้องทำงานผ่าน JVM และด้วยเหตุผลนี้เอง ทำให้จาวาคงเป็นลักษณะของ Write Once, Run Anywhere คือเขียนครั้งเดียวใช้ได้ทุกที่

2.10 Client-Server

การทำงานหรือการประมวลผลของเว็บ ถูกแยกออกเป็นสองแบบ โดยพิจารณาที่การประมวลผลเป็นหลัก คือ

2.10.1 การประมวลผลที่ไคลแอนท์ (Client-Side Processing)



รูปที่ 2-5 การประมวลผลที่ไคลแอนท์

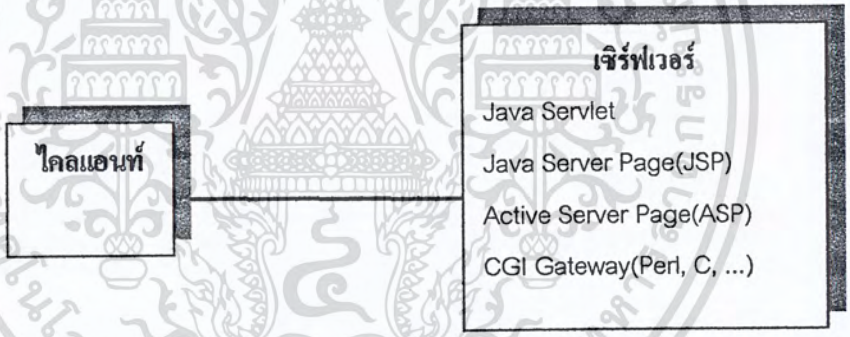
รูปแบบการทำงานลักษณะนี้ ไคลแอนท์ซึ่งก็คือ โปรแกรมบราวเซอร์ที่จำเป็นต้องทำงานหนัก เนื่องจากว่าต้องแปลงคำสั่งต้างๆที่บรรจุอยู่บนเว็บให้สามารถทำงานได้อย่างถูกต้อง ดังนั้นเครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรมไคลแอนท์ต้องใช้ทรัพยากรเพียงพอกับการทำงาน เทคนิคเหล่านี้ เป็นเทคโนโลยีที่ทำงานบนฝั่งไคลแอนท์

- **JavaScript** สคริปต์ภาษาจาวา ทำงานบนบราวเซอร์ทั่วไป โดยบราวเซอร์จะแปลงคำสั่งสคริปต์ก่อน แล้วจึงจะทำงานได้ตามคำสั่ง
- **JScript** สคริปภาษาจาวาของ Microsoft ทำงานได้เฉพาะ Internet Explorer ของ Microsoft เองเท่านั้น

- **VBScript** สคริปภาษาเบสิกของ Microsoft ทำงานได้เฉพาะ Internet Explorer ของ Microsoft เองเท่านั้นเช่นกัน
- **Java Applet** ไบต์โค้ดเขียนด้วยภาษาจาวา และถูกคอมไพล์แล้ว ทำงานบนบราวเซอร์คือ คำสั่งในไบต์โค้ดถูกแปลง แล้วจึงทำงาน
- **Java Bean** บีนภาษาจาวา เป็นคอมโพเน็นต์ที่ทำงานเหมือนแอปเพล็ต แต่ถูกสร้างขึ้นมาด้วยลักษณะของ JavaBean เขียนด้วยจาวาจึงทำงานได้ทุกแพล็ตฟอร์ม
- **ActiveX** คือคอมโพเน็นต์คล้ายกับ Java Bean แต่ต่างกันที่ภาษาที่ใช้สร้าง ส่วนใหญ่สร้างจากเครื่องมือ ที่ใช้งานบนวินโดว์ ดังนั้น ActiveX จึงทำงานได้บน Internet Explorer

2.10.2 การประมวลผลที่เซิร์ฟเวอร์(Server-Side Processing)

รูปแบบการทำงานลักษณะนี้ เซิร์ฟเวอร์คือเว็บเซิร์ฟเวอร์ จำเป็นต้องทำงานหนัก เนื่องจากเมื่อผู้ใช้เรียกใช้งาน ต้องทำการประมวลผลก่อน แล้วจึงค่อยส่งผลลัพธ์กลับไปให้ผู้ใช้ ดังนั้นเครื่องผู้ใช้ไม่จำเป็นต้องมีทรัพยากรมากมาย (ผู้ใช้ชอบ ไม่ต้องเสียตังค์มาก) และยังสามารถควบคุมให้ทำงานได้ ในกรณีที่ผู้ใช้ใช้บราวเซอร์แตกต่างกัน แต่ในทางกลับกันเซิร์ฟเวอร์ต้องทำงานหนักขึ้น และอาศัยทรัพยากรเพิ่มขึ้น เทคนิคเหล่านี้ทำงานบนเซิร์ฟเวอร์



รูปที่ 2-6 การประมวลผลที่เซิร์ฟเวอร์

- **Java Servlet** เป็นรูปแบบของโปรแกรมจาวาที่ทำงานบนเซิร์ฟเวอร์ โดยส่งผลลัพธ์ผ่านโพรโตคอล HTTP กลับไปยังบราวเซอร์ของผู้ใช้ เนื่องจากเขียนด้วยจาวา จึงทำงานได้กับเซิร์ฟเวอร์ ที่มีแพล็ตฟอร์มแตกต่างกันได้
- **Java Server Page(JSP)** เป็นผลพวงมาจาก Java Servlet สามารถสร้างเว็บด้วยภาษา HTML แล้วแทรกรูปแบบของภาษาจาวาสไตล์ JSP ลงไปในไฟล์เว็บ โดยไฟล์เว็บนี้ต้องถูกประมวลผลในส่วนที่เป็นคำสั่ง JSP ก่อนส่งข้อมูลที่เป็นภาษา HTML และ ผลลัพธ์จาก JSP กลับไป
- **Active Server Page(ASP)** เหมือนกับ JSP แต่ภาษาที่อยู่รวมกับ HTML เป็นสไตล์ของภาษา Basic

- **CGI Gateway** คือรูปแบบการสร้างโปรแกรมที่ถูกเรียกใช้งานบนเว็บ แต่สามารถสื่อสารกับผู้ใช้ผ่านโพรโตคอล HTTP ในตัวแปรสถาพแวดล้อม (Environment Variables) ดังนั้น โปรแกรมที่ว่าจะเขียนด้วยภาษาอะไรก็ได้ เช่น Perl, C, Visual C++, VB, Delphi, ... (ยังมีอีกเยอะ) แต่โปรแกรมที่ว่าต้องเข้าใจ ตัวแปรสภาพแวดล้อมของ HTTP

2.10.3 ข้อเปรียบเทียบของการประมวลผลที่ไคลแอนท์และเซิร์ฟเวอร์

การทำงาน	ไคลแอนท์	เซิร์ฟเวอร์
อินเทอร์เร็กทีฟ	ดีกว่า	จัดการยุ่งยาก และซับซ้อน
ความต้องการทรัพยากรที่บราวเซอร์	จำเป็นอย่างยิ่ง	-
ความต้องการทรัพยากรที่เซิร์ฟเวอร์	-	จำเป็นอย่างที่สุด เพราะต้องรองรับผู้ใช้จำนวนมากๆ
การติดตั้งและคอนฟิกที่บราวเซอร์	จำเป็น เมื่อต้องการทำงาน ในลักษณะเฉพาะ เช่นฐานข้อมูล	-
เวลาในการทำงาน	ขึ้นอยู่กับทรัพยากรไคลแอนท์และเทคนิค	ขึ้นอยู่กับเซิร์ฟเวอร์ และเทคนิคการใช้

ตารางที่ 2-2 ข้อเปรียบเทียบของการประมวลผลที่ไคลแอนท์และเซิร์ฟเวอร์

10.2.4 การ deploy แต่ละคอมโพเนนท์

- **Enterprise bean**

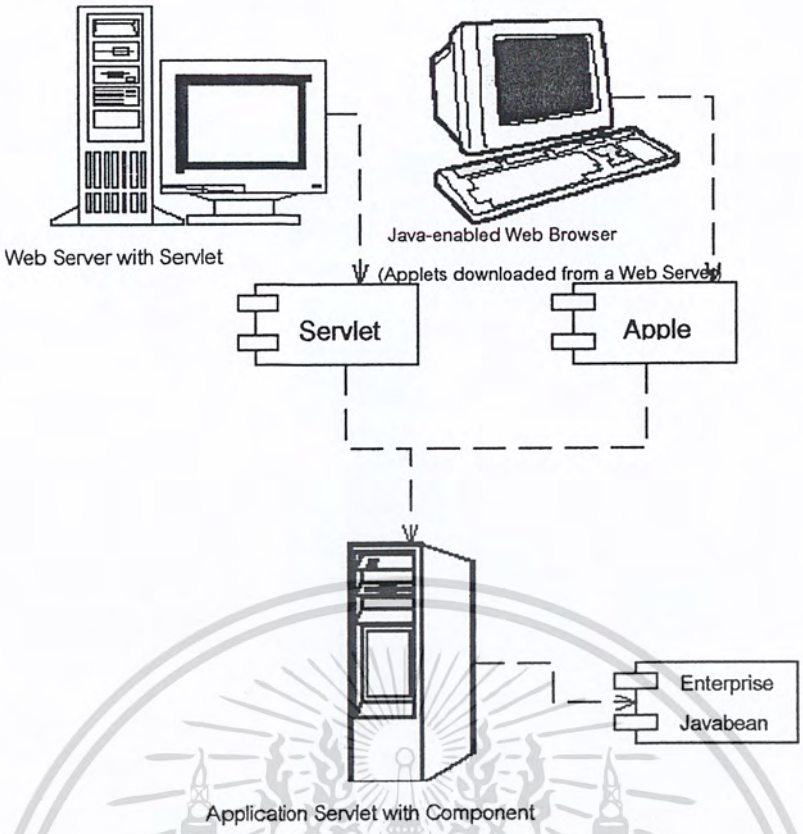
สามารถ deploy ได้ใน Application Server โดย Application Server จะจัดการคอนเทนเนอร์ในขณะเวลารัน (run time)

- **Applet**

สามารถ deploy ได้บนหน้าเว็บ โดยจะมี Appletveiwer ที่อยู่ในเบาว์เซอร์จัดการคอนเทนเนอร์ในเวลารัน

- **Servlet**

สามารถ deploy ได้บนเว็บเซิร์ฟเวอร์ โดยจะมี Servlet engine ที่อยู่ในเว็บเซิร์ฟเวอร์จัดการคอเทนเนอร์ในเวลารัน ดังรูป



รูปที่ 2-7 Applet, servlet และ Enterprise JavaBeans

จุดเด่นที่สุดของ enterprise java bean คือ bean สามารถติดต่อ database ข้าม server ได้ดี และมีลักษณะ float (คือลอย) อยู่ตลอดไม่เหมือนกับ ภาษา script อื่น จึงทำให้ EJB (Enterprise Java Bean) เหมาะ สำหรับนำมาใช้กับงาน middleware ได้ดีมาก คาดว่าคงจะแทน corba ได้ไม่ยากเพราะความเด่นตรงที่เป็น java นั่นเอง คือคุยกันด้วยภาษาเดียวไม่เหมือนกับ coba

2.10.5 ความแตกต่างระหว่าง Enterprise bean,Applet และ Servlet

- **Applet**

เป็นโปรแกรมจาวาที่สามารถดาวน์โหลดและเอ็กซีคิว (Execute) ได้ เช่น สามารถดาวน์โหลดจากเว็บเซิร์ฟเวอร์ลงในเว็บเบาว์เซอร์และจากเว็บเบาว์เซอร์แสดงผลผ่าน User Interface ไปยังผู้ใช้ได้

- **Servlet**

เป็นคอมโพเนนท์ที่สามารถใช้ในการขยายความสามารถของเว็บเซิร์ฟเวอร์(Web server)ได้ สามารถร้องขอบริการ (Request) และตอบสนองบริการ(Response)ได้ เช่น เมื่อมีไคลเอ็นท์โฮสท์(Client Host)เช่น เว็บเบาว์เซอร์มาร้องขอบริการก็จะตอบสนองบริการกลับไป ลักษณะการทำงานเช่นนี้ทำให้ Servlet เหมาะกับการทำงานเกี่ยวกับเว็บ เช่น render ไฟล์ HTML ติดต่อไปยังอี-คอมเมอร์ส แคตตาล็อค (E-Commerce Catalog)

ทั้ง Applet และ Servlet เหมาะกับการทำงานบนฝั่งไคล์เอนท์ เช่น render GUIs(Graphic User Interfaces), ลอจิกที่เกี่ยวกับการแสดงผล (Presentation-relate Logic)

- **Enterprise Bean**

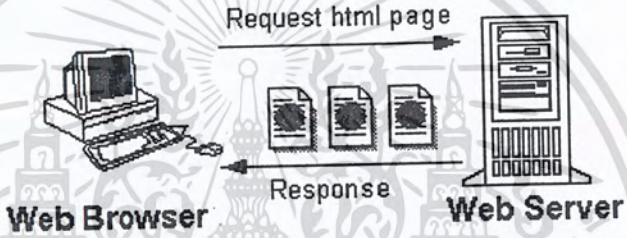
เป็นคอมโพเนนท์ที่ทำงานบนฝั่งเซิร์ฟเวอร์ เช่น เอ็กซีคิวอัลกอริทึมที่ซับซ้อน หรือ ทำงานเกี่ยวกับ Business Transaction ที่มีปริมาณมากๆ โดยจะต้องมีทรัพยากรในการเอ็กซีคิวที่สูง, การทำงานที่เป็นทรานแซคชั่น, มีระบบรักษาความปลอดภัยรองรับสำหรับผู้ใช้หลายคน เซิฟเวอร์ที่ใช้เช่น แอพพลิเคชั่นเซิฟเวอร์ (Application Server) จะเป็นตัวจัดการคอมโพเนนท์บนฝั่งเซิฟเวอร์สำหรับ Enterprise Bean

สรุปแล้วก็คือ Applets, Servlets และ Enterprise Bean เป็นเทคโนโลยีที่มีความสำคัญทั้งหมด เช่น เราสามารถใช้ Javabean ในการสร้างประกอบกันขึ้นมาเป็น Enterprise Bean ที่สามารถ deploy ได้และสามารถจัดการติดต่อผู้ใช้ไปยัง Enterprise Bean ได้โดยใช้ Servlet หรือ Applet

บทที่ 3
สถาปัตยกรรมบนฝั่งเซิร์ฟเวอร์

3.1 วิวัฒนาการของ Server Side Application

Server Side Application ที่ถูกใช้บนเวปตั้งแต่สมัยแรก ๆ นั้นส่วนประกอบหลักของเวปไม่ค่อยจะมีอะไรซักเท่าไหร่ โดยใช้หลักการพื้นฐานของ client/server architecture ซึ่งส่วนที่เป็น client ในเวปก็คือเว็บบราวเซอร์ (web browser) ยกตัวอย่างเช่น NCSA Mosaic, Lynx, Netscape และส่วนที่เป็น server ซึ่งก็คือ web server และโดยทั่วไป web server หนึ่ง ๆ จะสามารถบริการส่งข้อมูลให้ตัวเว็บบราวเซอร์ได้หลายตัว ซึ่งจะเป็นลักษณะที่คนหลายคน (multiple clients) สามารถที่จะเข้าไปดูข้อมูลต่าง ๆ จากเวปไซค์เดียวกัน (1 server) ได้ในเวลาเดียวกันนั่นเอง



รูปที่ 3-1 เบสิค client/server architecture เริ่มแรกสุดที่ใช้กันบนเวป

ต่อมาทั้งคนทำเวปและคนดูเวปก็คงจะเริ่มเบื่อจึงช่วยกันคิดค้นหาวิธีที่ทำให้เวปมีสีสันมากขึ้น เทคโนโลยีที่เกิดขึ้นตามมาก็คือ CGI (Common Gateway Interface) หลักการของ CGI ก็คือแทนที่ข้อมูลที่เราส่งไปให้ web server (โดยผ่านทาง web browser) ก็จะถูก process โดยตัว web server อย่างเดียว ข้อมูลนี้ก็อาจจะถูกส่งไปให้อีกโปรแกรมหนึ่งซึ่งตัวโปรแกรมนี้จะสามารถถูกเรียกได้โดย web server เพื่อทำการประมวลผลข้อมูลที่มาจากเรา โดยทั้งนี้ข้อมูลของเราจะถูก process ที่ใดก็ขึ้นอยู่กับ Setting ต่าง ๆ ที่ทางเวปไซค์ที่เราเข้าไปดูกำหนดขึ้นมา ยกตัวอย่างเช่น ถ้าเราต้องการดูเพจธรรมดา ๆ หลักจากที่เราคลิกไปที่ตัวลิงค์ตัว web server ก็จะไปอ่านเพจ (html ไฟล์) ที่เราต้องการแล้วส่งกลับมาเป็น text stream ให้เรา แต่ถ้าเราต้องการที่จะ post ข้อความลงไปในเวป ทาง web server ก็จะส่งข้อความที่เราพิมพ์ไปให้โปรแกรมที่ทำหน้าที่ดูแลและเก็บข้อความที่เราส่งเข้าไป

หลักจากที่ CGI กลายเป็นที่นิยมจนเป็นส่วนหนึ่งของ WWW Standard แล้ว Server Side Application ที่เป็นลูกหลานก็เกิดขึ้นกันมาอย่างมากมาย ยกตัวอย่างเช่น ASP, PHP, ColdFusion, Servlet, JSPและอื่น ๆ (อย่างไรก็ตามในโครงงานนี้จะกล่าวถึงเทคโนโลยีที่เกี่ยวข้องกับ Server Side Application ที่ใช้จาว่า platform เป็นหลักเท่านั้น)

3.2 Multi-tier Architecture

เซิร์ฟเวอร์ที่ดีนั้นจะต้องสามารถทำงานได้อย่างมีเสถียรภาพ, มีความน่าเชื่อถือ, มีประสิทธิภาพ และสามารถสนับสนุนการทำงานเมื่อมีผู้ใช้ทำงานพร้อมกันหลาย ๆ คนได้ รวมทั้งสนับสนุนการทำงานของไคลเอ็นท์ทั้งหมดได้เป็นอย่างดี เพราะว่าการทำงานของไคลเอ็นท์นั้นจะขึ้นอยู่กับเซิร์ฟเวอร์กลางเป็นสำคัญ ถ้าเกิดเซิร์ฟเวอร์กลางเกิดดาวน์ขึ้นมาจะทำให้ยากที่จะกู้ขึ้นใหม่ได้ และอาจมีส่วนที่เป็นอันตรายเข้าไปในระบบได้ ดังนั้นการที่จะทำการ deploy บนฝั่งเซิร์ฟเวอร์จะต้องทดสอบทีละขั้น

ในการทำ deployment ที่ดีนั้นในทางลอจิกจะต้องแบ่งออกเป็นเลเยอร์ (Layer) โดยแต่ละเลเยอร์จะตอบสนองการทำงานที่ต่างกันและสามารถมีได้มากกว่า 1 คอมโพเนนท์ดังนี้

- **Presentation Layer**

เป็นคอมโพเนนท์ที่เกี่ยวกับการติดต่อกับผู้ใช้ เช่น Presentation Layer ของแอพพลิเคชันสามารถเขียนโดยใช้ภาษา Visual Basic, Presentation Layer ในทาง Web-based deployment สามารถใช้ Java Servlet, Java server pages หรือ Java applets

- **Business Logic Layer**

เป็นคอมโพเนนท์ที่แก้ปัญหาเกี่ยวกับ Business problem ต้องมีเอนจิน (Engine) ที่มีประสิทธิภาพสูง เช่น Catalog Engine, Pricing Engine โดยทั่วไปแล้วคอมโพเนนท์นี้ต้องเขียนด้วยภาษาที่มีระบบความปลอดภัยที่ดี เช่น Java หรือ C++

- **Data Layer**

คอมโพเนนท์นี้จะถูกใช้โดย Business Logic Layer โดยศูนย์กลางที่ Data Layer จะติดต่อไปถึงได้แก่ ฐานข้อมูลซึ่งสามารถติดต่อไปถึงได้หลายฐานข้อมูล

ข้อดีของการแบ่งเป็นแต่ละเลเยอร์คือ ทำให้สามารถเปลี่ยนแปลงเลเยอร์แต่ละเลเยอร์ได้โดยแทบไม่มีผลกระทบต่อเลเยอร์อื่น เพราะว่าแต่ละส่วนแยกเป็นอิสระจากกัน เช่น การเปลี่ยนแปลงฐานข้อมูลในส่วน Data Layer

แต่ในการแบ่งการ deploy ออกเป็นแต่ละเลเยอร์นั้นเป็นการแบ่งในทาง Logical ซึ่งจะแตกต่างจากการแบ่งในเชิง Physical คือการแบ่งจะแบ่งเป็นหน่วยที่เรียกว่า "เทียร์ " (Tier) โดยจะมีสถาปัตยกรรมที่เป็น 2 เทียร์, 3 เทียร์ และแบบหลายเทียร์

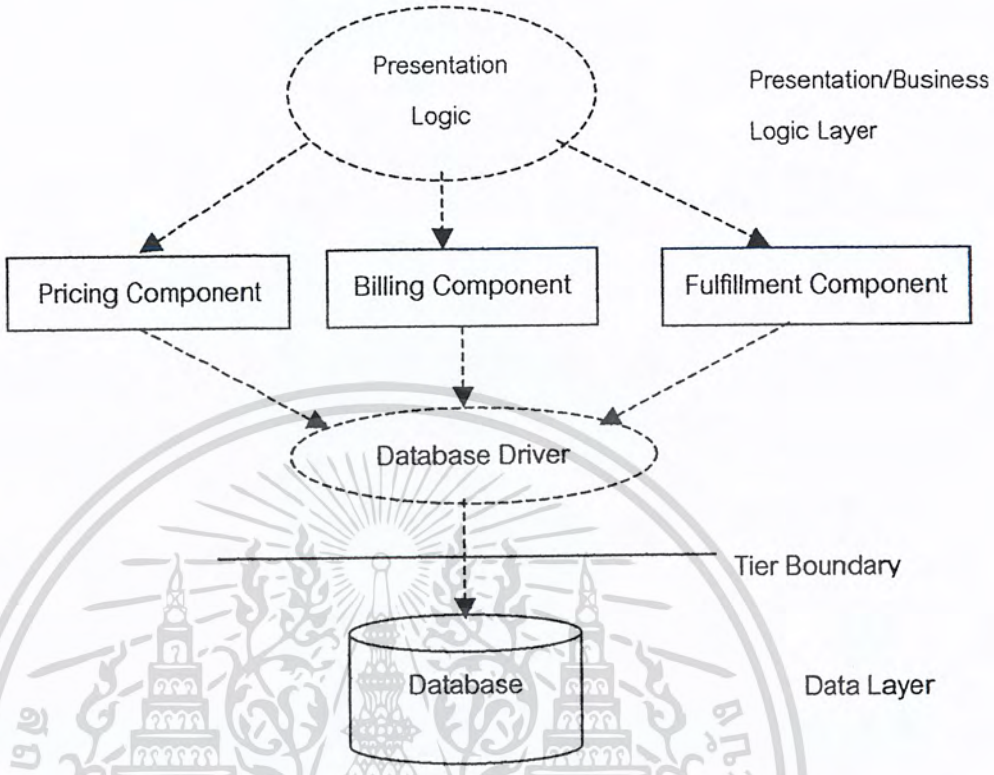
3.2.1 สถาปัตยกรรม 2 เทียร์ (2 - Tier Architecture)

ในสถาปัตยกรรมนี้จะเป็นการนำส่วน Business Logic Layer เข้าไปรวมกับเลเยอร์อื่นจะมี 2 แบบที่เป็นไปได้คือ การรวม Business Logic Layer เข้ากับ Presentation Layer และ รวม Business Logic Layer เข้ากับ Data Layer

- การรวมส่วน Business Logic Layer เข้ากับ Presentation Layer

คือการรวม Business Logic Layer กับ Presentation Layer เอาไว้เป็นเทียร์แรก และเอาส่วน Data Layer เอาไว้เป็นเทียร์ที่สอง ดังรูป 3-2 โดยเทียร์แรกนั้นจะเป็นไคลเอ็นต์ (Client) และเทียร์ที่สองเป็นเซิร์ฟเวอร์ (Server) ทำให้สถาปัตยกรรมนี้มีลักษณะที่ไคลเอ็นต์นั้นจะอ้วน (Fat) ส่วนเซิร์ฟเวอร์นั้นจะผอม (Thin) ในสถาปัตยกรรมนี้แอพพลิเคชั่นจากไคลเอ็นต์จะติดต่อกับส่วน Data Layer โดยผ่าน

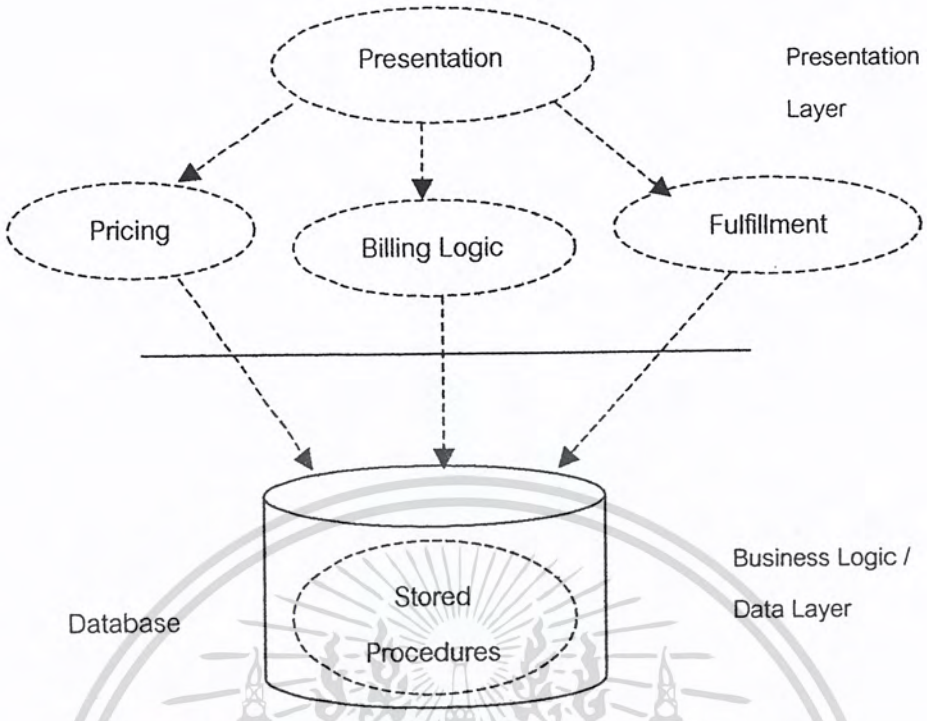
Database bridge API เช่น Open Database Connectivity (ODBC) หรือ Java Database Connectivity (JDBC)



รูปที่ 3-2 การรวมส่วน Business Logic Layer เข้ากับ Presentation Layer

- การรวมส่วน Business Logic Layer เข้ากับ Data Layer

คือ การรวม Business Logic Layer กับ Data Layer เอาไว้เป็นเทียร์แรก และเอาส่วน Presentation Layer เอาไว้เป็นเทียร์ที่สอง ดังรูป 3.3 โดยคิดว่าเทียร์แรกเป็นเซิร์ฟเวอร์ (Server) และเทียร์ที่สองเป็นไคลเอ็นต์ (Client) ทำให้สถาปัตยกรรมนี้มีลักษณะที่เซิร์ฟเวอร์นั้นจะอ้วน (Fat) ส่วนไคลเอ็นต์นั้นจะผอม (Thin) ในสถาปัตยกรรมนี้จะใส่ Business Logic เอาไว้ในฐานข้อมูล โดยฐานข้อมูลจะอนุญาตให้สามารถเอ็กซีคิว (Execute) Context ที่อยู่ในฐานข้อมูลได้เมื่อสร้างโมดูลขึ้นมาเรียกใช้เรียกว่า "Stored Procedures" โดยการใส่ส่วนของลอจิกเข้าไปใน Stored Procedures ซึ่งจะช่วยทำให้การทำงานมีประสิทธิภาพมากขึ้น เพราะว่าในการใส่ลอจิกเข้าไปใน Stored Procedures ทำให้ระบบเครือข่าย (Network) ลดเวลาในการวิ่งกลับไปกลับมาจากลอจิกไปยังฐานข้อมูล แทนที่จะต้อง Query ข้อมูลจากฐานข้อมูล N ครั้ง เป็นการลด Traffic ภายในเครือข่าย ช่วยแก้ปัญหาที่มีในการรวมส่วน Business Logic Layer เข้ากับ Presentation Layer แต่ว่าสถาปัตยกรรมนี้ก็ไม่สามารถแก้ปัญหาได้ทั้งหมดดัง เช่น ภาษาของ Stored Procedures จะผูกมัดทำให้ไคลเอ็นต์เป็นส่วนหนึ่งของฐานข้อมูล แต่จุดประสงค์ของการที่มี Database Bridge ก็เพื่อต้องการทำให้สามารถเปลี่ยนแปลงแก้ไขฐานข้อมูลได้ง่าย



รูปที่ 3-3 การรวมส่วน Business Logic Layer เข้ากับ Data Layer

3.2.2 ลักษณะของสถาปัตยกรรม 2 เทียร์

ลักษณะของสถาปัตกรรมแบบ 2 เทียร์ คือ

1.ค่าใช้จ่ายสูง เพราะในไดเวอร์ของฐานข้อมูล (Database driver) จะต้องติดตั้งไคลเอ็นต์เทียร์แต่ละเทียร์ไว้ด้วย

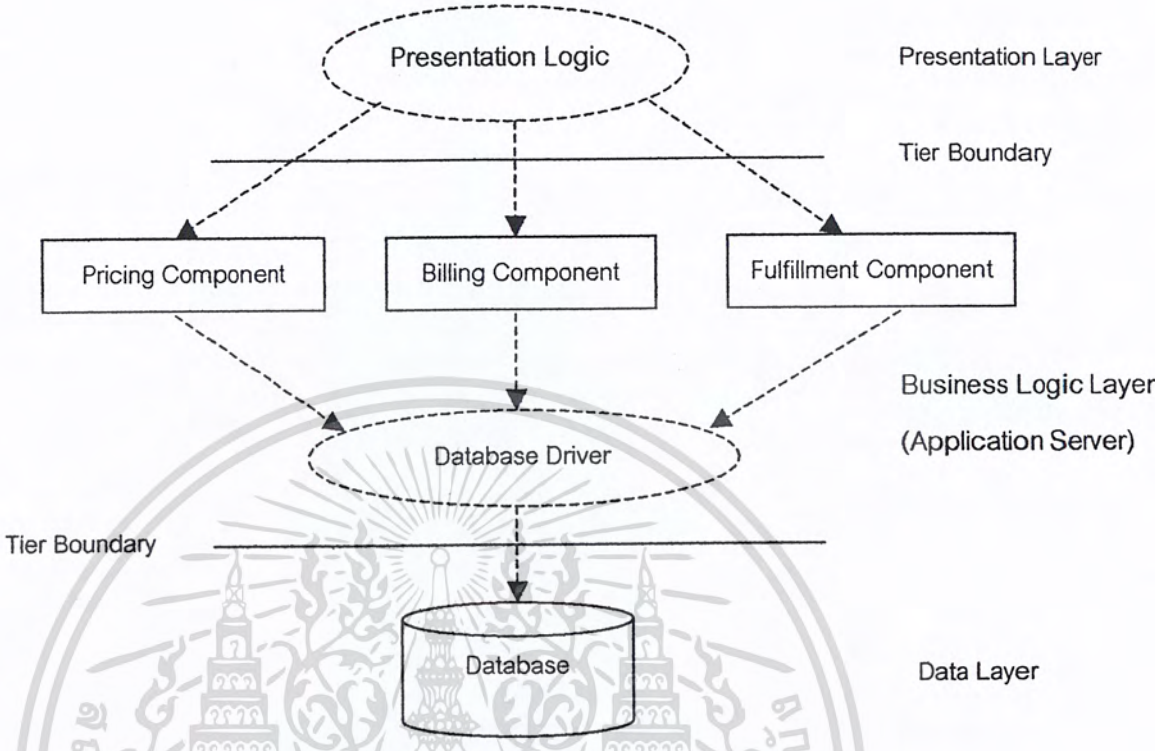
2.ค่าใช้จ่ายเมื่อต้องการเปลี่ยนแปลงในการไดเวอร์ของฐานข้อมูลมีสูง โดยจะต้องติดตั้งไดเวอร์ของฐานข้อมูลภายในแต่ละไคลเอ็นต์ใหม่ทั้งหมดซึ่งเป็นค่าใช้ในการซ่อมบำรุง (Maintenance) ที่สูงมาก

3.ค่าใช้จ่ายในการเปลี่ยนแปลง Database schema มีสูง เนื่องจากเทียร์ที่เป็นไคลเอ็นต์นั้นจะต้องเข้าถึงฐานข้อมูลโดยตรงโดยผ่าน JDBC หรือ ODBC หรือ SQL/J ซึ่งนั้นหมายถึงว่าไคลเอ็นต์จะต้องติดต่อกับ Database schema ถ้าต้องการที่จะเปลี่ยนแปลงกระบวนการทางธุรกิจ (Business Process) ใหม่จะต้อง deploy แต่ละไคลเอ็นต์ใหม่ด้วย

4.ค่าใช้จ่ายเมื่อต้องการเปลี่ยนแปลงชนิดของฐานข้อมูลใหม่มีสูง เช่น เมื่อต้องการเปลี่ยนจากฐานข้อมูลที่เป็น Relational มาเป็นฐานข้อมูลที่เป็น Object ซึ่งไม่เพียงแต่จะต้อง deploy แต่ละไคลเอ็นต์ใหม่เท่านั้นแต่จะต้องแก้โค้ดของไคลเอ็นต์ให้เหมาะสมกับชนิดของฐานข้อมูลด้วย

5. ในแต่ละครั้งที่ Business Logic Layer ทำการติดต่อกับ Database Layer จะต้องมีการวิ่งกลับไปกลับมาหลายรอบเพราะถูกแยกออกเป็นคนละส่วนกัน ทำให้เสียเวลาเป็นการจำกัดความสามารถของฐานข้อมูล และทำให้ระบบเครือข่าย (Network) ให้ทำงานช้าลงและเป็นการลด bandwidth การทำงานของผู้ใช้ด้วย

3.2.3 สถาปัตยกรรม N เทียร์



รูปที่ 3-4 แสดง 3-Tier Web-Based Deployment

เป็นสถาปัตยกรรมที่เพิ่มเทียร์ให้เพิ่มขึ้นจากที่มีอยู่แล้วจากแบบ 2 เทียร์โดยจะแยก Presentation Layer, Business Logic Layer และ Data Layer กันออกมาเป็นแต่ละเทียร์ในทาง Physical ถ้าเป็นสถาปัตยกรรมที่มีมากกว่า 3 เทียร์จะแยกแต่ละเลเยอร์ออกจากกันอิสระ จากตัวอย่างดังรูป 3-4 เป็น 3-Tier Web-Based Deployment โดยเทียร์จะแตกออกเป็นดังนี้

- Presentation Tier

สามารถรันได้บนหลายเว็บเซิร์ฟเวอร์ (Web-Server) ภายในจะประกอบไปด้วย Java servlet, Scripts (เช่น Activer Server Pages, Java Server Pages) และลอจิกที่เชื่อมการทำงานของแต่ละส่วน

- Business Logic Tier

สามารถรันได้บนหลายแอพพลิเคชั่นเซิร์ฟเวอร์ (Application Server) โดยแอพพลิเคชั่นเซิร์ฟเวอร์จะต้องมีคอนเทนเนอร์ (Container) ที่ใช้ในการรัน Business Logic และมีการบริการ (Service) เพื่อรองรับการจัดการคอมโพเนนท์ที่มีประสิทธิภาพ เช่น แอพพลิเคชั่นเซิร์ฟเวอร์จะจัดการการเข้าถึงข้อมูลภายในฐานข้อมูลให้กับ Business Component

- Data Tier

ประกอบด้วยฐานข้อมูลซึ่งสามารถมีได้มากกว่า 1 ฐานข้อมูลและ Stored Procedures

3.2.4 ลักษณะของสถาปัตยกรรม N เทียร์

1. ค่าใช้จ่ายในการ deploy ต่ำ เพราะว่าไดรเวอร์ฐานข้อมูลจะถูกติดตั้งบนฝั่งเซิร์ฟเวอร์มากกว่าที่จะติดตั้งบนฝั่งไคลเอนต์

2. ค่าใช้จ่ายเมื่อมีการเปลี่ยนแปลงฐานข้อมูลต่ำ เพราะว่าไคลเอนต์ไม่ได้เข้าถึงฐานข้อมูลโดยตรง แต่จะเข้าถึงโดยผ่านเทียร์ตรงกลาง ทำให้เมื่อมีการเปลี่ยนแปลง Database Schema จะเปลี่ยนเพียงไดรเวอร์ของฐานข้อมูลไม่ต้องทำการ deploy ไคลเอนต์ใหม่

3. ค่าใช้จ่ายเมื่อมีการเปลี่ยนแปลง Business Logic ต่ำ เพราะไม่ต้องคอมไพล์และ deploy เทียร์ของไคลเอนต์ใหม่

4. มี Filewalls เป็นตัวรักษาตวามปลอดภัย คือ ป้องกันการเข้าถึงข้อมูลภายในแอพพลิเคชั่นเซิร์ฟเวอร์

5. มีการจัดการที่มีประสิทธิภาพเมื่อมีการเรียกใช้ทรัพยากรภายนอก (External Resources) เช่น พรินเตอร์ และเมื่อขณะที่ไคลเอ็นต์ทำงานไม่ต้องใช้ทรัพยากร (Resource) เช่น การ render การแสดงผลที่ต้องส่งไปยังผู้ใช้ ซึ่งขณะนั้นไม่จำเป็นต้องใช้ทรัพยากร ทรัพยากรจะถูกจัดสรรไปให้ไคลเอ็นต์อื่นที่ต้องการที่ต้องทำเช่นนี้เพราะว่า

- ความสิ้นเปลืองในการเข้าถึงฐานข้อมูลนั้นมีสูงจึงต้องจำกัดจำนวนของผู้ที่ต้องการเข้าถึงฐานข้อมูลในขณะใดขณะหนึ่ง
- ไม่จำเป็นที่ต้องติดต่อ ไปยังทรัพยากรตลอดเวลา
- เป็นการเพิ่มประสิทธิภาพของแอพพลิเคชั่น

6. ในแต่ละเทียร์มีการทำงานอย่างอิสระ เช่น สามารถเพิ่มหรือแก้ไขฐานข้อมูลได้โดยแก้ไขโค้ดน้อยมากและต้องคอมไพล์ใหม่เฉพาะเทียร์ที่แก้ไขเท่านั้น

7. เมื่อมีเทียร์ใดเกิดโอเวอร์โหลด (Overload) เทียร์อื่นก็ยังคงมีคุณสมบัติครบเหมือนเดิม

8. ถ้ามีเทียร์ใดเกิด Error ขึ้นมาก็จะเป็นเฉพาะเทียร์นั้นไม่เกี่ยวกับเทียร์อื่น เช่น ถ้าแอพพลิเคชั่นเซิร์ฟเวอร์เกิดเครช (Crashed) เว็บเซิร์ฟเวอร์จะรายงานไปยังบราวเซอร์ของไคลเอ็นต์ว่า “Site Down”

9. ค่าใช้จ่ายในการซ่อมบำรุงสูง

3.3 Java 2 Platform, Enterprise Editon (J2EE)

ผลิตภัณฑ์จากสถาปัตยกรรมบนฝั่งเซิร์ฟเวอร์ (Server-Side Component Architectuer Solutions) นั้นได้มีออกมาโดยค่ายต่างๆดังนี้

1. Microsoft’s Distributed interNet Application Architecture (DNA)
2. Sun Microsystem’s Java 2 Platform,Enterprise Edition (J2EE)
3. The Object Management Group’s COBRA Standard

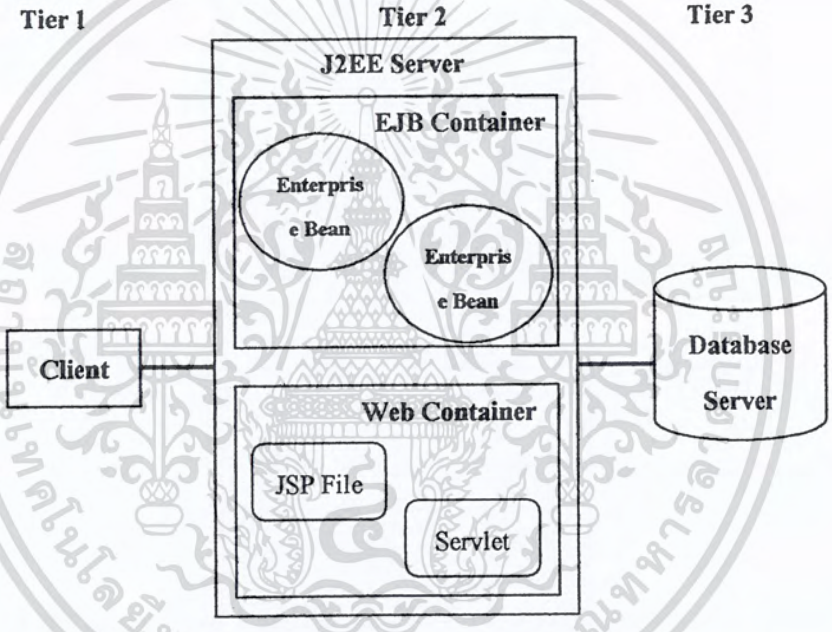
แต่ในที่นี้จะขออธิบายเฉพาะค่ายของ SUN ซึ่งคือ J2EE

3.3.1 Sun Microsystem’s Java 2 Platform,Enterprise Edition (J2EE)

ภาษาจาวาจะเป็นภาษาที่สะดวกในการเขียนคอมโพเนนท์ใส่ไว้บนฝั่งเซิร์ฟเวอร์เพราะว่าตลาดของฝั่งเซิร์ฟเวอร์จะมี UNIX และเมนเฟรม (Mainframe) เป็นแกนหลักซึ่งหมายความว่าภาษาที่สามารถข้ามแพลตฟอร์มได้จะมีความสำคัญมากเพราะผู้พัฒนาจะต้องเขียนคอมโพเนนท์เพียงครั้งเดียวแต่สามารถ deploy ได้บนหลาย Evironment

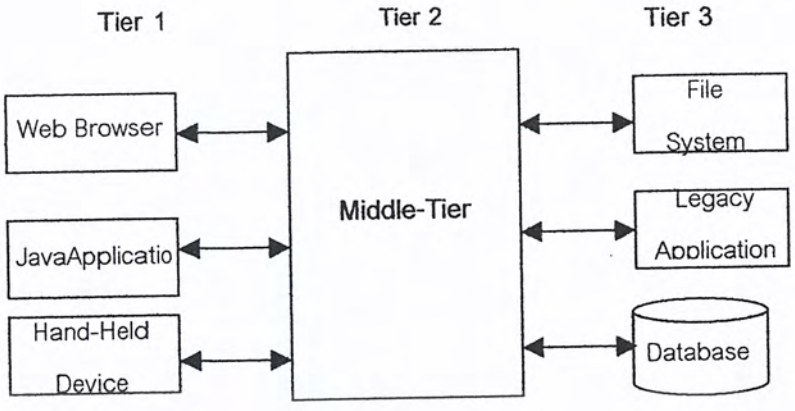
ในการตองสนองของจาวาบนฝั่งเซิร์ฟเวอร์ SUN จึงได้ออกแพลตฟอร์มที่ชื่อว่า "Java 2 Platform, Enterprise Edition (J2EE) ซึ่งเป็นแพลตฟอร์มที่มีความอิสระ (independent), สนับสนุนกับผู้ใช้ได้หลาย ๆ คน (Multi-user) และยังมีระบบรักษาความปลอดภัย โดยแกนหลักของ J2EE คือ EJB (Enterprise Javabeans) ซึ่งเป็นมาตรฐานในการสร้างคอมโพเนนท์บนฝั่งเซิร์ฟเวอร์โดยใช้ภาษาจาวา

3.3.2 สถาปัตยกรรม J2EE



รูปที่ 3-5 แสดงโครงสร้างสถาปัตยกรรมของ J2EE ในระบบ 3-Tier Server

จากรูป 3-5 จะเห็นว่า J2EE นั้นเป็น Middle-Tier Servers ซึ่งเป็นส่วนสำคัญหลักในระบบ 3-tier Server โดยมีหน้าที่บริการการร้องขอจาก Client แบ่งการทำงานในการติดต่อกับระบบ back office และฐานข้อมูลออกเป็นส่วนๆเพื่อลดความซับซ้อน นอกจากนี้ยังช่วยในการจัดการกับ resource ต่าง ๆ ของระบบรวมทั้งการรักษาความปลอดภัย โดย Middle-tier server นี้จะสนับสนุนการทำงานหลายอย่างของ client เช่น Web browsers, Java application และ Client จะเป็นตัวคอยดูแลเกี่ยวกับการติดต่อกับผู้ใช้เท่านั้น การที่ยกหน้าที่ต่างเหล่านี้ให้เป็นของ Middle-Tier เลยทำให้ client มีขนาดเล็ก,ไม่ซับซ้อนและพัฒนาไปได้ช้าทำให้เราสามารถรวม Client ใหม่เข้ากับแอปพลิเคชันและฐานข้อมูลที่มีอยู่ได้ Middle-Tier servers ทำให้คุณสามารถสร้างแอปพลิเคชันแต่ละ แอปพลิเคชัน ที่มีขนาดใหญ่ได้ และสถาปัยกรรม J2EE จึงเป็นอีกทางเลือกในการพัฒนา middle-tier servers



รูปที่ 3-6 แสดงโครงสร้างของระบบ 3-Tier Client/Server Application

จากรูป3-6 นั้น Tier1 จะประกอบด้วยหลาย ๆ client ที่ร้องขอการบริการไปที่ middle-tier server ซึ่งอยู่ใน Tier2 โดยใน Tier2 จะเข้าถึงข้อมูลจากระบบใน Tier3 โดยจะทำการ Process Rule ที่เข้ามา และส่งผลลัพธ์กลับไปยัง Client ใน Tier1

J2EE เป็นแพลตฟอร์ม ที่สร้างขึ้นโดยมีเป้าหมายเพื่อพัฒนาและจัดการแอพพลิเคชันแบบ Multi-Tier Server โดย J2EE Servers มีความสามารถด้านต่าง ๆ ดังนี้

- สามารถใช้หลักการของ naming และ directory ซึ่งจะอนุญาตให้โปรแกรมจองการใช้งานและคอมโพเน้นต์ ผ่าน Java Naming and Directoryl Interface (JNDI)
- มีความสามารถที่จะสร้างความปลอดภัยให้กับระบบโดยให้มีการ Authentication ซึ่งจะต้องให้ผู้ใช้ล็อคอินเข้าสู่ระบบ
- สนับสนุนการใช้ HTTP ซึ่งทำให้เว็บบราวเซอร์เข้าถึง servlet และไฟล์ JavaServer Page (JSP)
- สนับสนุนการทำงานของ EJB โดยอนุญาตให้ client เรียกเมทอดของ enterprise beans ได้
- เป็นแพลตฟอร์มที่รวบรวมความสามารถในการสร้างและจัดการซอฟแวร์ในระดับ Enterprise
- อนุญาตให้แอพพลิเคชันในระดับ Enterprise สามารถรันได้ทุกที่ และรวบรวมความสามารถในการบริการระดับ Enterprise ไว้อย่างสมบูรณ์

3.3.3 เทคโนโลยีของ J2EE (J2EE Technologies)

เทคโนโลยีของ J2EE ประกอบด้วยส่วนต่างๆดังนี้

- เอนเตอร์ไพซ์ จาวาบีนส์ หรือ Enterprise JavaBean (EJB)
- ข้อกำหนดทางด้านการสื่อสารระหว่างเครื่องคอมพิวเตอร์ในรูปแบบของ Java Remote Method Invocation (RMI)
- การเชื่อมต่อกับฐานข้อมูลด้วยภาษาจาวาหรือ Java Database Connectivity (JDBC)
- มาตรฐานที่ใช้ในการติดต่อกับออบเจ็กต์ที่ทำงานร่วมกันภายใต้ระบบเครือข่ายหรือ Java Naming and Directories Interface (JNDI)

- การจัดการกับทรานส์แอ็กชันหรือ Java Transaction APIs (JTA), Java Transaction Services (JTS)
- การให้บริการด้านการส่ง message โดยภาษาจาวาหรือ Java Messaging Services (JMS)
- จาวาเซิร์ฟเล็ต (Java Servlets) และ จาวาเซิร์ฟเวอร์เพจ (Java Server Pages :JSP)
- Java IDL
- จาวาเมล์ (JavaMail)
- Connectors
- ความสามารถในการแยกส่วนของข้อมูลเพื่อประโยชน์ในการแสดงผลโดยใช้ Extensible Markup Language (XML)

3.4 เอนเตอร์ไพซ์ จาวาบีนส์ (Enterprise JavaBean : EJB)

จุดเด่นที่สุดของ Enterprise Javabeans คือ bean ที่สามารถติดต่อ database ข้าม server ได้ดี และมีลักษณะ float หรือลอยอยู่ตลอดไม่เหมือนกับ ภาษา script อื่น จึงทำให้ EJB (Enterprise Javabeans) เหมาะสำหรับนำมาใช้กับงาน middleware ได้ดีมาก คาดว่าคงจะแทน Corba ได้ไม่ยากเพราะความเด่นตรงที่เป็น java นั่นเอง คือคุยกันด้วยภาษาเดียวไม่เหมือนกับ corba นอกจากนี้ Enterprise beans ยังเป็นส่วนประกอบส่วนหนึ่งของ server ที่เขียนด้วยภาษา Java มี 2 ชนิด คือ

3.4.1 Session Beans

Session Beans เป็นเสมือนไคลเอ็นต์ใน J2EE Server ซึ่งไคลเอ็นต์จะทำการติดต่อกับ J2EE Server โดยการเรียกเมทอดของ Enterprise bean เช่น Online shopping ไคลเอ็นต์จะเรียกเมทอด enterOrder ของ session bean ให้สร้างลำดับไคลเอ็นต์สามารถเปลี่ยนแปลง session bean ได้ สามารถคิดได้ว่า session bean เป็นส่วนขยายของไคลเอ็นต์ในแต่ละ session bean สามารถมีได้เพียงไคลเอ็นต์เดียว และอยู่ไม่ถาวรเมื่อไคลเอ็นต์ยกเลิกการติดต่อจะทำให้ session bean ยกเลิกไปด้วย นอกจากนี้ session bean ยังแบ่งออกเป็น 2 ประเภท คือ stateful session beans และ stateless session beans ซึ่งจะกล่าวถึงรายละเอียดในบทที่ 5

3.4.2 Entity Bean

Entity bean เป็นเสมือนอ๊อบเจ็คที่เรากำหนดขึ้นถูกเก็บไว้ในฐานข้อมูล เช่น Entity bean อาจแทนลูกค้า ซึ่งต้องเก็บเป็น row ในตารางลูกค้าของ relational database ข้อมูลของ entity bean จะไม่ถูกเก็บใน relational database สามารถเก็บไว้ในออบเจ็กต์-ดาต้าเบส อาจเก็บเป็นไฟล์, แอพพลิเคชัน โดยชนิดการเก็บนั้นจะขึ้นอยู่กับเทคโนโลยีของ EJB

Entity bean เป็น EJB ที่อยู่ถาวร สามารถถูกจัดการโดย entity bean ด้วยกันหรือ EJB Container โดย Entity bean ต้องการให้เขียนโค้ดเข้าถึงข้อมูลภายในบีนเพราะตัว Entity bean จริง ๆ จะอยู่ใน Database Server เช่น ภายใน entity bean ลูกค้าจะมีคำสั่ง SQL ในการเข้าถึง relational database ผ่าน JDBC โดย entity bean ยังแบ่งย่อยได้เป็น 2 ประเภท คือ bean-managed persistent และ container-

managed persistent ซึ่งจะกล่าวถึงในบทที่ 5 เช่นกัน

3.5 Remote Method Invocation (RMI) and RMI-IIOP

RMI หรือ Remote Method Invocation เป็นข้อกำหนดทางด้านการสื่อสารระหว่างเครื่องคอมพิวเตอร์ข้อหนึ่งที่สามารถนำไปใช้ประโยชน์ได้ในลักษณะเดียวกันกับ RPC (Remote Procedure Call) ในขณะที่ RPC ถูกนำไปใช้กับระบบซอฟต์แวร์ที่มีการโปรแกรมแบบ Structural model แต่ RMI จะใช้กับระบบซอฟต์แวร์ที่มีการโปรแกรมแบบ Object-Oriented model

Distributed system หรือ ระบบการประมวลผลแบบกระจาย มีความต้องการประสิทธิภาพที่สูงในการติดต่อสื่อสารระหว่าง Process ซึ่งอยู่ต่างเครื่องกัน ภาษา Java สนับสนุนการใช้ Socket ซึ่งเป็นกลไกการติดต่อสื่อสารระหว่าง Process แบบพื้นฐานที่อยู่ในระดับดี เนื่องจากมีความยืดหยุ่น และมีประสิทธิภาพเพียงพอสำหรับการติดต่อสื่อสารที่ใช้กันทั่วไป แต่ Socket มีความต้องการ Protocol ซึ่งมีการทำงานในลักษณะ Client/Server ที่จะจัดการการติดต่อสื่อสารในระดับแอปพลิเคชันเพื่อที่จะเข้ารหัส และถอดรหัส Message ที่จะใช้เพื่อการแลกเปลี่ยน ซึ่งการออกแบบ Protocol นี้ย่อมเป็นภาระของผู้พัฒนาระบบซอฟต์แวร์ และ Protocol ที่ต้องพัฒนาขึ้นเองที่อาจเกิดข้อผิดพลาดขึ้น อันจะส่งผลถึงประสิทธิภาพของระบบที่นำ Protocol นี้มาใช้ด้วย

RPC เป็นข้อกำหนดตัวหนึ่งที่มีความสามารถดังความต้องการข้างต้น ซึ่งจะสร้างการติดต่อสื่อสารให้เกิดขึ้นได้ในลักษณะของการเรียกใช้ Procedure (Procedure call) แทนการส่งข้อมูลหรือ Message ผ่าน Socket โดยตรงเมื่อเกิดความจำเป็นที่ต้องมีการติดต่อสื่อสารระหว่าง Object ในระดับโปรแกรม ซึ่ง Object เหล่านั้นมีอยู่ในเครื่องที่ต่างเครื่องกัน แต่เราไม่สามารถนำ RPC มาดัดแปลงใช้กับ Distributed system ที่เป็น Object model ได้ ฉะนั้น Distributed system ที่เป็น Object model จึงต้องการข้อกำหนดตัวใหม่อย่าง RMI

Java RMI ถูกพัฒนาขึ้นโดยทีมงานพัฒนาซอฟต์แวร์ของ Sun Microsystem ซึ่งออกแบบมาให้ทำงานได้ในสภาพแวดล้อมของ Java เท่านั้น ในขณะที่ RMI ตัวอื่นๆ สามารถนำมาใช้กับ Object ของ Java ได้ แต่ไม่มีความสะดวก ไม่มีความราบรื่น ในการติดต่อสื่อสารระหว่าง Object ของระบบเหล่านั้นกับ Object ของ Java เท่าที่ควร (อย่างเช่น CORBA ที่สามารถจัดการกับสภาพแวดล้อมที่ต่างระบบกันได้ ซึ่ง Object ของ CORBA จำเป็นต้องมีรูปแบบของภาษาที่เป็นกลาง) ในทางกลับกัน RMI ที่เป็นของ Java (Java RMI) โดยเฉพาะ จะทำให้ดึงความสามารถของ Object ของ Java มาใช้ได้อย่างเต็มที่

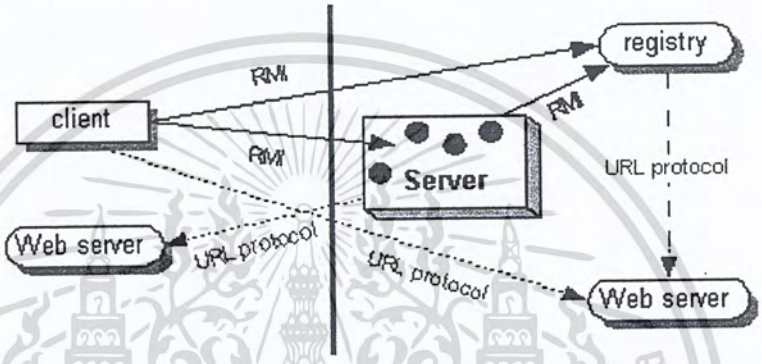
โปรแกรมที่ถูกพัฒนาขึ้นมาโดยใช้เทคโนโลยีของ Java สามารถที่จะเรียกใช้ Object ที่อยู่ต่างเครื่องได้โดยอาศัยค่า Reference ของ Object เหล่านั้นซึ่งมันได้รับมา โดยการได้มาของค่า Reference นี้อาจได้มาจาก

- เข้าไปค้นหาจาก Bootstrap naming service ที่ระบบ RMI ได้จัดเตรียมไว้ให้
- ได้จาก Argument หรือ Return value ที่ได้มาโดยการเรียกหา Method

เครื่องทุกเครื่องสามารถเรียกใช้ Object จากต่างเครื่องได้ โดยไม่มีการกำหนดสถานะไว้ว่าเครื่องนี้ต้องเป็น Server หรือ Client แต่เพียงอย่างเดียวตลอดเวลา และ RMI ยังใช้ Object serialization เพื่อทำ

การ Marshal และ Unmarshal ตัว Parameter โดยไม่ตัดข้อมูลที่จะบอกชนิดของ Object ทำให้สามารถจัดการกับ Object ชนิดที่ผู้ใช้ (ผู้พัฒนา Java application) กำหนดขึ้นเองได้

จากรูป 3-7 นี้จะอธิบายถึง RMI distributed application ที่มีการใช้ registry เพื่อที่จะสามารถเข้าไปอ้างถึง remote object โดยเซิร์ฟเวอร์จะเป็นผู้เรียกตัว registry เพื่อที่จะกำหนดชื่อให้กับ remote object ส่วนตัว client นั้นจะมองหา remote object โดยใช้ชื่อของตัว remote object นั้นที่อยู่ใน server's registry จากนั้นก็จะทำการเรียกขอ method ที่มีอยู่ใน remote object นั้น นอกจากนั้นในรูปยังแสดงถึง RMI system ที่มีการใช้ Web server ในการโหลด bytecodes จากเซิร์ฟเวอร์ไปยัง client และจาก client ไปยัง server เมื่อมีการใช้ object นั้นอีกด้วย



รูปที่ 3-7 แสดง RMI distributed application

3.6 Java Naming and Directory Interface (JNDI)

JNDI นั้นถือว่าเป็นคอมโพเน้นท์ (Component) ที่สำคัญของ Java 2 Platfrom, Enterprise Edition (J2EE) โดยจะมี interface มาตรฐานที่ใช้ในการติดต่อกับ user , machine, network, objects และ service ต่างๆ เช่น เราสามารถใช้ JNDI ในการเข้าติดต่อกับ printer ที่อยู่ในระบบ intranet ที่ทำงานร่วมกันอยู่ได้ นอกจากนี้ยังสามารถใช้ในการเข้าถึง java object หรือในการเชื่อมต่อกับ database ได้อีกด้วย นอกจากนี้ใน EJB, JNDI ยังถูกใช้อย่างมากมายในการเข้าถึง object ข้ามระบบเครือข่าย

3.6.1 Naming and Directory Services

Naming service นั้นเหมือนกับโอเปอเรเตอร์ที่ทำงานกับโทรศัพท์ (telephone operator) ที่เมื่อเราต้องการเรียกไปยังอีกคนหนึ่งแต่เราไม่รู้เบอร์โทรของคนคนนั้น ดังนั้นเราจึงต้องโทรไปยังศูนย์บริการลูกค้า เพื่อที่จะขอข้อมูลเบอร์โทรศัพท์ของคนคนนั้นโดยการบอกชื่อของคนที่ต้องการติดต่อ จากนั้นโอเปอเรเตอร์จะทำการตรวจหาและทำการเชื่อมต่อไปยังบุคคลนั้นให้แก่เรา ดังนั้น naming service จึงมีหน้าที่ดังนี้

- ทำการกำหนดชื่อให้กับออปเจ็กต์ซึ่งเรียกว่าการ binding name ให้กับออปเจ็กต์
- ให้บริการในการหาออปเจ็กต์เมื่อทราบชื่อของออปเจ็กต์นั้น หรือเรียกว่าการ looking up หรือ searching object

ตัวอย่างการ naming อื่น ๆ เช่น การเชื่อมต่อไปยัง machine อื่น ๆ บนเน็ตเวิร์ค (network) นั้น โดย Domain Name System (DNS) จะถูกใช้ในการแปลง machine name ไปเป็น IP address

โดยปกติแล้ว naming service จะสามารถหาออปเจ็กต์ที่อยู่ทั่วไปได้ แต่จะไม่สามารถหาออปเจ็กต์บางชนิดได้ซึ่งก็คือ directory object หรือ directory entity ซึ่งมีความแตกต่างจากออปเจ็กต์ทั่วไปคือ การที่สามารถเก็บแอตทริบิวต์ (attribute) ต่าง ๆ ไว้กับ directory object ได้เพื่อใช้ในวัตถุประสงค์ต่าง ๆ เช่น สามารถใช้ directory object ในการเก็บข้อมูลของ user เช่น password ซึ่งถ้าเรามีแอปพลิเคชัน (application) ที่จะต้องมีการ authentication เราก็สามารเก็บ login และ password ของ user ไว้ในส่วนของแอตทริบิวต์ใน directory object ได้และเมื่อไคลเอ็นต์ทำการขอการเชื่อมต่อมายังแอปพลิเคชัน (application) นั้น ไคลเอ็นต์ก็ทำการป้อน login และ password ซึ่งเราสามารถนำมาเปรียบเทียบกับ login name และ password ที่เก็บไว้เป็นแอตทริบิวต์ของ directory object ได้เพื่อดูว่าข้อมูลนั้นถูกต้องหรือไม่

Directory service เป็น naming service ที่ให้บริการในการหา directory object ที่เก็บแอตทริบิวต์ต่าง ๆ เอาไว้ภายใน

3.6.2 ข้อดีของ JNDI

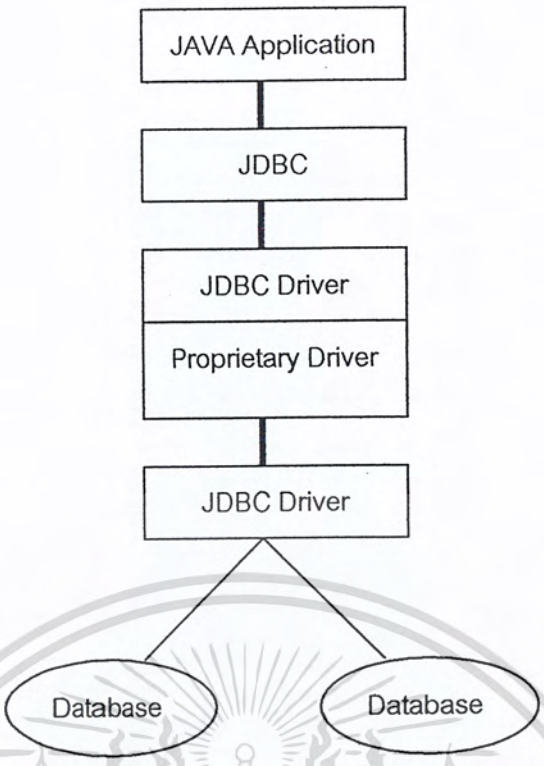
ข้อดีของ JNDI นั้นเป็นดังนี้

- JNDI นั้นเป็นระบบที่ช่วยในการเข้าถึงข้อมูลทุกชนิดที่อยู่ภายใน directory object ได้ เช่น login , password, หมายเลขโทรศัพท์, email address, ที่อยู่ตามทะเบียนบ้าน หรือ network address
- JNDI เป็น single API ที่ใช้ในการเข้าถึง directory ต่าง ๆ ด้วยโปรโตคอล (protocal) ต่าง ๆ กัน
- JNDI นั้นมีการทำงานที่ยืดหยุ่นโดยสามารถนำไปใช้ร่วมกับ directory ต่างๆได้โดยไม่มีผลกระทบต่อส่วนของโค้ดที่ฝั่งไคลเอ็นต์ (client code)

การใช้ JNDI นั้นทำให้เราสามารถทำการ อ่านหรือเขียน java object ทั้งหมดที่อยู่ใน directory ต่าง ๆ ได้

3.7 Java Database Connectivity (JDBC)

ในภาษาโปรแกรมสมัยใหม่จะมีช่องทางสำหรับเชื่อมต่อกับดาต้าเบส (database) อยู่ ในภาษาจาวา (Java) ก็เช่นเดียวกัน ซึ่งสามารถทำได้โดยการใช้ JDBC (Java Database Connectivity) ซึ่งมีโมดูลการเชื่อมต่ออย่างง่าย ดังรูป 3-12



รูปที่ 3-8 แสดงโมดูลการเชื่อมต่อ JDBC อย่างง่าย

3.7.1 ลักษณะของ JDBC

- เป็นมิดเดิลแวร์ (middleware) ที่สร้างมาจากภาษาจาวา
- ทำให้ JDBC ทำงานได้หลายแพลตฟอร์ม เนื่องมาจากข้อกำหนด Write once,run anywhere
- JDBC เป็นรูปแบบในการติดต่อกับฐานข้อมูลมีการแบ่งวิธีการในการทำงานออกเป็น 4 Type คือ Type1, Type2, Type3 และ Type4
- JDBC driver คือ ไดร์เวอร์ส่วนที่จะนำมาประกอบกับรูปแบบ JDBC ให้เข้าใจฐานข้อมูล เช่น Oracle JDBC Driver คือ ไดร์เวอร์ที่เข้าใจการเก็บข้อมูลของ Oracle

3.7.1.1 JDBC Type1 (JDBC -- ODBC Bridge)

- คือ JDBC ที่ทำงานบน ODBC อีกที
- ช่วยให้สามารถนำเอาฐานข้อมูลเดิมที่ทำงานบนพื้นฐานของวินโดว์มาใช้งานจาวา
- เพื่อหลีกเลี่ยงปัญหาจากความจำเป็นต้องปรับเปลี่ยนระบบทั้งระบบเป็นการปรับเปลี่ยนเพียงบางส่วนเท่านั้น
- ไม่เหมาะกับการนำไปใช้งานบนระบบงานที่มีขนาดใหญ่ๆ เนื่องจากทำให้เกิดความช้าในการทำงานและประสิทธิภาพในการทำงานที่ไม่ดี
- มีข้อมูลในส่วนโอเวอร์เฮดสูงเนื่องจากต้องมีส่วนในการติดต่อระหว่าง JDBC และ ODBC เพิ่มเติม

- ไม่สนับสนุนความสามารถทั้งหมดของมาตรฐาน JDBC เนื่องจากข้อจำกัดของ ODBC

3.7.1.2 JDBC Type2 (Partial Java Driver)

- มีประสิทธิภาพในการทำงานดีกว่าประเภทแรกเมื่อเปรียบเทียบกัน
- คำสั่งในการติดต่อกับเซิร์ฟเวอร์จะเป็นคำสั่งที่ดีที่สุดสำหรับเซิร์ฟเวอร์นั้น ๆ โดยเฉพาะทำให้การทำงานโดยรวมดีกว่า
- ผู้ใช้โปรแกรมในส่วนไคลเอ็นต์ยังต้องการใช้ไดรเวอร์สำหรับเซิร์ฟเวอร์โดยเฉพาะ
- เมื่อมีการเปลี่ยนแปลงเซิร์ฟเวอร์เป็นตัวอื่นโปรแกรมในส่วนไคลเอ็นต์ต้องมีการเปลี่ยนแปลงและ คอมไพล์ใหม่เสมอ

3.7.1.3 JDBC Type3 (Pure Java Driver)

- มีประสิทธิภาพในการทำงานดีกว่าประเภทที่ 1 และ 2 เมื่อเปรียบเทียบกัน
- เหมาะสำหรับองค์กรที่เซิร์ฟเวอร์ทางด้านฐานข้อมูลที่หลากหลายรูปแบบ
- การทำงานของไคลเอ็นต์ไม่จำเป็นต้องติดตั้ง JDBC ไว้ในทุกตัว
- การติดตั้งและการดูแลระบบไคลเอ็นต์สามารถทำได้ง่ายและสะดวก
- ยังต้องการไดรเวอร์สำหรับแต่ละผลิตภัณฑ์เพื่อติดตั้งไว้ที่ Application Server

3.7.1.4 JDBC Type4 (Direct-to-DB/Native to protocol)

- มีประสิทธิภาพในการทำงานดีที่สุด
- มีความยุ่งยากในการพัฒนาเพราะผู้พัฒนาต้องมีการเรียนรู้การทำงานของ server มาก่อน

สำหรับในโครงงานนี้เราเลือกใช้ JDBC Type2 (Partial Java Driver) สำหรับการติดต่อกับ Oracle Database ในการเลือกใช้ JDBC driver นั้นจะขึ้นอยู่กับตัวคาต้าเบสแต่ละตัวว่าสนับสนุน JDBC Driver ชนิดใด

บทที่ 4

EJB Container

4.1 การทำงานของ EJB Container

EJB Container นี้จะเป็นตัวควบคุม enterprise beans เนื่องด้วย ออบเจ็กต์ของ Enterprise bean จะรันอยู่ใน EJB Container นอกจากจะควบคุมขณะที่รันแล้ว ยังทำหน้าที่เป็นตัวให้บริการกระบวนการทำงานต่าง ๆ ด้วย EJB Container มีหน้าที่ให้บริการกับ enterprise beans ในด้านต่าง ๆ คือ

- การบริหารการใช้งานทรัพยากร และอายุในการใช้งานบีนที่อยู่ในระบบ (Resource Management and Bean Life Cycle Management)
- การจัดการกับทรานส์แอ็กชัน (Transaction Management)
- การสนับสนุนในการสร้างความปลอดภัยให้กับระบบ (Security)
- ความสามารถในการคงสถานะเดิมของบีน (Persistance)
- ความสามารถในการเรียกใช้งานบีนแบบ Remote Accessibility

4.1.1 การบริหารการใช้งานทรัพยากร และอายุในการใช้งานบีนที่อยู่ใน

EJB container นั้นจะรับผิดชอบในการให้บริการด้านการบริหารมรัพยากรของระบบ (resource management) โดยทรัพยากร (resource) นั้นอาจจะเป็น threads, socket connection, database connection ซึ่งอาจจะถูก pool โดย application server และนอกจากการจัดการด้าน resource management แล้ว EJB container ยังรับผิดชอบในด้านการควบคุมวงจรและอายุในการใช้งานของคอมโพเน้นต์ที่เป็นแบบ enterprise bean ซึ่งถูก deploy ไว้โดยการ instantiates, destroys หรือ การ reuse เช่น เมื่อไคลเอ็นต์ร้องขอบีนซึ่งไม่ได้อยู่ใน memory ในขณะนั้น ดังนั้น EJB container ก็จะทำการ instanctiate instance ตัวใหม่ขึ้นมาใน memory เพื่อให้ ไคลเอ็นต์ สามารถใช้งานได้ หรืออีกนัยหนึ่งในกรณีที่บีนนั้นมีอยู่ใน memory อยู่แล้ว EJB container ก็จะไม่ต้องทำการ instanctiate บีนนั้นขึ้นมาใหม่ หรือในกรณีที่ถ้าในระบบนั้นเหลือ memory อยู่น้อยแล้ว EJB container นั้นก็จะทำการ reassign บีนจากไคลเอ็นต์ตัวหนึ่งให้ไคลเอ็นต์อีกตัวหนึ่ง หรืออาจจะทำลายบีนตัวอื่นที่ไม่ค่อยได้มีการใช้งาน โดยกระบวนการนี้เรียกว่า instance pooling

4.2 Instance Pooling

4.2.1 การ pooling และ reuse ของ container ใน stateless session bean

เนื่องจาก method ของ ejbcreate() ใน stateless session bean นั้นไม่ต้องผ่าน parameter ทำให้ไคลเอ็นต์ จึงไม่ต้องส่ง critical information ที่ bean instance ต้องใช้ในการ start up ไปดังนั้น EJB container จึงใช้สิ่งนี้ให้เกิดประโยชน์ในการที่จะทำ Precreate instance ของ stateless session bean ดังนั้น EJB container จึงต้องสามารถทำการ pool stateless session bean instance ก่อนที่ไคลเอ็นต์ จะทำการ connect มา เมื่อไคลเอ็นต์ทำการเรียกใช้ method ใด ๆ แล้ว container ก็จะสามารถได้ instance นั้นจากการ pooling และให้ service กับไคลเอ็นต์นั้น และหลังจากนั้นก็จะส่งค่า instance นั้นกลับไปยัง bean pool นั้น วิธีการนี้ทำให้ container สามารถกำหนด instance ของ bean ให้กับไคลเอ็นต์ ต่าง ๆ ได้อย่าง dynamic

ผลของการทำงานแบบนี้ก็คือ ทุก ๆ ครั้งที่มีการ request เข้ามาจากไคลเอ็นต์จะมี instance ของ stateless session bean หลายตัวที่สามารถให้บริการต่อการ request นั้นได้ เนื่องจาก stateless session bean นั้นจะให้บริการ method ที่ถูก request มาเพียง 1 การ request เท่านั้น และก็จะเป็นอิสระจากไคลเอ็นต์ specific หลังจากการ call method นั้น นั่นคือ stateless session bean ทุกตัวจะมีสถานะเหมือนกัน หลังจาก method call แต่ละครั้ง นอกจากนี้มันจะไม่รู้ว่ามีการเกิด method call ขึ้นก่อนหน้านั้นหรือไม่ ดังนั้น container จึงสามารถทำการ reassign beans ต่าง ๆ ให้กับการ request ของไคลเอ็นต์ได้ใน per – method level ซึ่งก็คือการทำ container – specific

ข้อดีของการทำ bean instance pooling ก็คือบีนต่าง ๆ ที่ใช้ในการให้บริการจะมีจำนวนน้อยกว่าจำนวนของไคลเอ็นต์ connecting ซึ่งเกิดจากการทำงานของไคลเอ็นต์ แบบ "think time" เช่น การเกิด network lag หรือ human decision ในฝั่งของไคลเอ็นต์โดยในขณะที่ไคลเอ็นต์เกิดการ think time นี้ container ก็สามารถใช้ bean instance เพื่อให้เซิร์ฟเวอร์กับไคลเอ็นต์ตัวอื่นได้เพื่อเป็นการประหยัด systems resource ได้อีกทางหนึ่ง

บีนที่ใช้ในกระบวนการ pooling (bean pool) นั้นจะมีขนาดแตกต่างกันไปขึ้นอยู่กับคอนเทนเนอร์ (container) ว่าจะสามารถเปลี่ยนขนาด pool size ได้หรือไม่เมื่อโหลดมีการเปลี่ยนแปลงอยู่เสมอ ๆ เช่น ถ้ามีไคลเอ็นต์ connect อยู่กับการ deployment ในเวลากลางวันมากกว่าเวลากลาคืนกับคอนเทนเนอร์ก็จะมีขนาดของการ pool ในเวลากลางวันมากกว่าเวลากลางคืนเช่นกัน วิธีการนี้จะทำให้ free system resource สามารถถูกนำไปใช้ได้ในงานอื่น ๆ ในขณะ off-peak hour ได้

4.2.2 การ pooling ของ container ใน statefull session bean

statefull session bean จำนวนมากที่ถูกใช้โดยไคลเอ็นต์ หลาย ๆ ตัวที่มาขอการ connect นั้น แต่ละตัวจะมีการเก็บ conversation state ในการทำงานที่มันทำกับไคลเอ็นต์ นั้น ๆ และเนื่องจาก EJB container/ Server มีจำนวน resource จำกัด เช่น memory , database connection และ socket connection ดังนั้นถ้า conversation state นั้นมีขนาดใหญ่ ก็จะทำให้ container / server นั้นมี resource ในการทำงานไม่เพียงพอ แต่สิ่งนี้จะไม่เป็นปัญหากับ stateless session bean เพราะว่า container นั้นสามารถ pool bean ได้จำนวนจำกัดในการให้บริการกับ ไคลเอ็นต์ หลายๆตัว

การ pooling ใน statefull session bean นั้นจะต่างกับใน stateless session bean คือเมื่อไคลเอ็นต์ทำการเรียก method ใน bean มันก็จะเริ่ม conversation กับ bean นั้นพร้อมทั้งทำการเก็บ conversation state ใน bean นั้นด้วย ซึ่ง state นี้ จะยังคงอยู่สำหรับไคลเอ็นต์ ตัวนั้นที่มันติดต่ออยู่เพื่อใน method request ครั้งต่อไป ดังนั้นกับคอนเทนเนอร์จึงไม่สามารถทำการ pooling bean และ assign bean ให้กับไคลเอ็นต์ ตัวอื่นๆ ที่มา request อย่าง dynamic ได้ เพราะแต่ละ bean จะเก็บ state ของไคลเอ็นต์ ที่มันทำงานอยู่เท่านั้น ดังนั้นเพื่อให้การทำงานแบบนี้เป็นไปด้วยดีจึงต้องใช้ resource อย่างประหยัด และเพิ่มประสิทธิภาพในการทำงานของระบบให้สูงขึ้น

ทุกครั้งที่มีการ run application บนเครื่องคอมพิวเตอร์จะมี physical memory ที่จำเป็นในการ run นั้นเป็นจำนวนจำกัด operating system จึงต้องหาวิธีในการจัดการเมื่อมี application หลาย ๆ ตัวเข้ามา run ถึงแม้ว่า application เหล่านั้น จะต้องการ memory รวมมากกว่า physical memory ที่มีอยู่ เพื่อที่จะแก้ไข

ปัญหานี้จึงมีการใช้ hard disk มาเป็น extension physical memory ซึ่งทำให้เกิดการขยายส่วนของ virtual memory ที่ระบบมีอยู่ เมื่อ application อยู่ในภาวะ idle memory ส่วนของ application นี้จะถูก swaped out ออกจาก physical memory ไปยัง hard disk และเมื่อ application กลับเข้ามา active อีกครั้ง ข้อมูลที่ต้องการใช้ก็จะถูก swapped in จาก hard disk ไปยัง physical memory โดยการ swap นี้จะเกิดขึ้นบ่อยเมื่อมีการ switching ระหว่าง application (เรียกว่า context switch)

EJB container ใช้ประโยชน์จากหลักการนี้เพื่อจัดการกับการใช้ resource ใน statefull session bean ในการที่จะจำกัดจำนวนของ statefull session bean instance ใน memory นั้น container สามารถที่จะ swap out statefull session bean โดยการ save conversation state ลง hard disk หรือ แหล่งเก็บข้อมูลอื่น ๆ เรียกว่า passivation หลังจากการ passivation ของ statefull session bean จะทำให้ resource เช่น memory สามารถถูกนำไปใช้ให้บริการกับไคลเอ็นต์ ตัวอื่นได้ และเมื่อไคลเอ็นต์เดิมทำการเรียก method เข้ามาอีกครั้ง state ที่ถูกนำไปเก็บไว้ก็จะถูก swap in เข้ามาใน bean นั้นอีกครั้ง เรียกว่า activation

EJB นั้น support กับการ pooling ของ statefull session bean นั่นคือจะมี instance เพียงจำนวนหนึ่งเท่านั้นที่มีอยู่ใน memory เมื่อมีการติดต่อกับ ไคลเอ็นต์จำนวนมาก แต่การ pooling ที่นำเทคนิค passivation / activation มาใช้นี้ อาจทำให้เกิด bottleneck ได้ ซึ่งไม่เกิดใน stateless session bean

การตัดสินว่า bean ไหนที่จะทำการ activation หรือ passivation นั้นจะขึ้นอยู่กับ container โดย container ส่วนมากนั้นจะใช้หลักการของ Least Recently used (LRU) นั่นคือจะทำการ passivate bean ที่มีการ call น้อยที่สุด วิธีนี้เหมาะในกรณีของremote ไคลเอ็นต์ที่มักเกิดการ disconnect จากระบบ network ทำให้ bean นั้น ตกค้างอยู่โดยไม่มีไคลเอ็นต์จึงควรทำการ passivate ออกไป โดยการ Passivation นั้นสามารถเกิดได้ตลอดเวลา เมื่อ bean นั้นไม่เกี่ยวข้องกับ method call แล้ว โดยการตัดสินใจจะขึ้นอยู่กับ container แต่มีข้อยกเว้น คือ bean ที่เกี่ยวข้องกับ transaction ทุกตัวจะไม่สามารถทำการ passivate ได้จนกระทั่ง transaction นั้นเสร็จสมบูรณ์

การ Activation นั้น container ส่วนมากจะใช้ algorithm just – in – time นั่นคือ bean ต่าง ๆ จะถูก activate ขึ้นมาตามความต้องการในขณะนั้น เช่นเมื่อมีการ request จากไคลเอ็นต์ เข้ามา ซึ่งถ้าในขณะนั้น state ของมันถูก passivation อยู่ container ก็จะทำการ activate bean นั้นเข้ามาสู่ memory เพื่อใช้งานต่อไป

ปกติแล้ว passivation และ activation นั้นจะไม่เหมาะกับ stateless session bean เพราะ bean ชนิดนี้ไม่มี state ดังนั้น stateless session bean จึงสามารถ destroy ได้ง่ายโดย container นอกจากนี้การ passivation/activation ยังเหมาะสำหรับ entity bean อีกด้วย

4.2.3 การ pooling ของ container entity bean instance

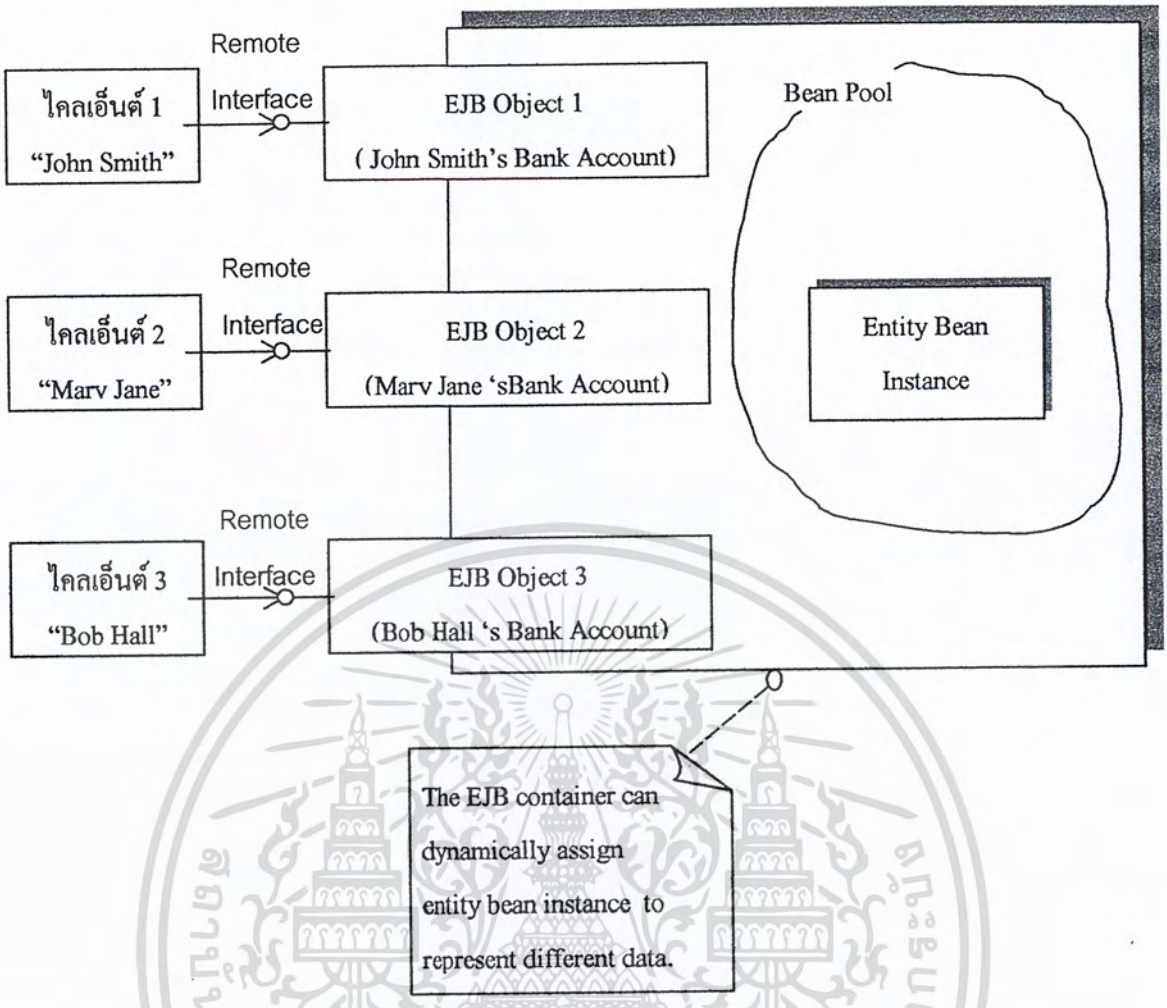
ในการสร้าง EJB cointainer / server ที่ดีนั้นจะต้องสามารถทำการ instantiate entity ได้โดย entity bean นั้นคือข้อมูลที่เก็บอยู่ในฐานข้อมูล ที่เกิดขึ้นเมื่อไคลเอ็นต์ มีการ connect และ disconnect นั่นคือ จะมีการสร้าง และทำลาย bean เพื่อให้บริการแก่ไคลเอ็นต์ นั้นๆ ซึ่งวิธีแบบนี้จะไม่เหมาะสมในการสร้าง application server เพราะการสร้างและทำลายของ object จะเป็นการสิ้นเปลืองมากโดยเฉพาะถ้าไคลเอ็นต์นั้นมีการ request บ่อยๆ

entity bean class นั้นจะอธิบายถึง field และ rule ต่างๆ สำหรับ entity bean นั้น แต่ไม่ได้ระบุข้อมูลที่แน่นอน เช่น entity bean class ที่ระบุ bank account แต่ละตัวก็จะมี field ต่าง ๆ คือ

- ชื่อของเจ้าของบัญชี
- Account ID
- จำนวนเงินในบัญชีในขณะนั้น

ใน bean class นั้นจะแสดงถึง instance ของข้อมูลใน database ที่แตกต่างกันออกไป เช่นข้อมูลใน bank account ใน 1 record ก็จะมี field ต่างๆ ที่มีข้อมูลต่างกันออกไปซึ่งใน bean class นั้นไม่ได้กำหนดข้อมูลที่ตายตัวไว้ภายใน ดังนั้นเพื่อที่จะลดเวลาในการ instantiate object ใดๆนั้น instance ของ entity bean จะสามารถนำกลับมาใช้ได้อีกครั้งโดยการ pooling ซึ่งในการทำนั้นจะขึ้นอยู่กับ container policy , container สามารถทำการ pool และ reuse instance entity bean นั้น เพื่อที่จะแสดงถึง instance ที่แตกต่างกันของข้อมูลชนิดเดียวกันที่อยู่ใน storage เช่น container สามารถใช้ bank account entity bean instance เพื่อที่จะแสดงถึง bank account record ต่างๆ ดังนั้นเมื่อมีการใช้ entity bean instance ,instance นั้นอาจใช้ในการ assign ให้กับ ไคลเอ็นต์ ต่างๆ ที่มีการ request เข้ามา และอาจจะใช้แสดงถึงข้อมูลในรูปแบบต่างๆ โดยการทำงานของ container นั้นจะเป็นแบบ dynamic

วิธีนี้จะช่วยให้ container ไม่จำเป็นต้องทำ instantiate bean instance ทุกครั้งที่มีการ request ของไคลเอ็นต์ แล้วยังช่วยลดการใช้ resource ของระบบได้ ดังรูป 4-1



รูปที่ 4-1 EJB Container pooling of entity bean

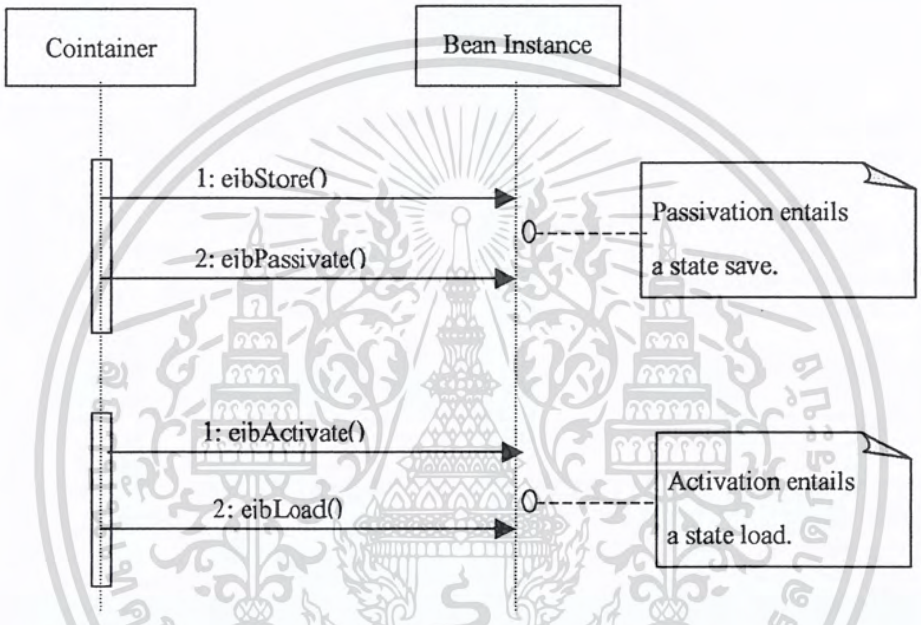
การทำ instance pooling นั้น container จะเป็นตัวจัดการ ซึ่งนอกจากจะใช้กับ entity bean แล้วยัง สามารถใช้กับ stateless session bean แต่ถ้าใช้กับ EJBobject เมื่อ entity bean ถูก assign ให้กับ EJBobject ตัวหนึ่งแล้วมันอาจต้องใช้ resource บางอย่าง เช่น socket connection แต่เมื่อมันอยู่ใน pool มันก็จะไม่ต้องใช้ socket ดังนั้นเพื่อที่จะให้ bean สามารถเข้าครอบครองและปลดปล่อย resource ได้นั้น entity bean class จะต้องมี callback method คือ

ejbactivate() : เป็น callback method ที่ container จะทำการเรียก bean instance เมื่อมีการเปลี่ยน bean นั้นออกจาก instance pool โดยกระบวนการนี้เรียกว่า activation และจะเป็นตัวบ่งชี้ว่า container ได้เข้าไปมีส่วนในการจัดการกับ bean ด้วย specific EJBobject และ specific primary key โดย ejbactivate() method นั้น จะต้องใช้ resource ต่างๆ เช่น sockets ที่ต้องใช้เมื่อทำการ assign ไปยัง particular EJBobject

ejbpassivate() เป็น callback method ที่ container เรียกใช้เมื่อมีการเปลี่ยน bean เข้าไปยัง instance pool เรียกกระบวนการนี้ว่า passivation และจะเป็นตัวบ่งชี้ว่า container นั้นจะไม่มีส่วนเกี่ยวข้องกับ bean

จาก specific EJBobject และ specific primary key โดย ejbpassivate() method นั้นจะปลดปล่อย resource เช่น socket ที่ได้มีการใช้ในภายหลังการทำงานของ ejbactivate()

เมื่อ entity bean instance นั้นถูก passivate แล้วนอกจากจะต้องปลดปล่อย resource ต่าง ๆ แล้ว ยังต้อง save state ของมันลง storage อีกด้วยทำให้ storage นั้นถูก update ให้ถูกต้องตาม state ในขณะนั้น และในการ save field ของ instance นั้นลง database ,container จะทำการเรียก ejbStore() method ก่อนที่จะทำการ passivation ในทำนองเดียวกัน เมื่อ entity bean ถูก activate นอกจากจะต้องการ resource แล้วยังจะต้อง load data ที่ต้องใช้มาจาก database ด้วย โดยในการ load data ไปยัง bean instance นั้น container จะเรียก ejbLoad() method หลังจากการทำ activatation ดังรูป 4-2



รูปที่ 4-2 State save และ Sate load ในการทำ Passivation และ Activation ของ entity bean

จริง ๆ แล้ว entity bean จะค่อนข้างคล้ายกับ stateful session beans โดยทั้ง 2 bean นี้สามารถทำ passivate/activate แต่จะมีข้อแตกต่างอยู่ คือ entity bean จะต้องเรียก ejbStore() callback ในการ save state ในระหว่างการทำ passivate และ ejbLoad() สำหรับ load data ในระหว่างการทำ activation แต่ใน stateful session bean นั้นไม่ต้องใช้ เพราะว่า container จะใช้ object serializable (equivalent) ในการรักษา field ต่าง ๆ ของ statefull session bean ไว้ ซึ่งการทำแบบนี้จะมีความซับซ้อนน้อยกว่าแบบของ entity bean.

4.3 Database Connection Pooling

การติดต่อกับฐานข้อมูลนั้นจะมีความจำกัดในด้านเวลาและจำนวนการติดต่อ เพื่อแก้ไขปัญหานี้ จึงให้ Container เป็นตัวจัดการ Pool ของการติดต่อฐานข้อมูล โดย Enterprise bean จะได้รับการติดต่อจาก Pool โดย เมื่อ bean หนึ่งยกเลิกการติดต่อแล้ว bean อื่นจะสามารถติดต่อแทนได้ทำให้ใช้งาน Database อย่างมีประสิทธิภาพ

4.4 Scalability

EJB server นั้นอาจจะสามารถวัดจำนวน resource ที่ใช้ไปในขณะใด ๆ ได้ โดยถ้าเราเพิ่มกลไกพิเศษที่มีความสามารถด้านต่าง ๆ ที่เท่ากันลงไป เช่น memory , processor power, disk space หรือ network bandwidth กลไกที่เพิ่มเข้าไปนี้จะทำให้จำนวนของ user ที่ฝั่ง server สามารถรองรับได้ และจำนวนของ transaction ที่ system สามารถ execute ได้ต่อวินาทีนั้นเพิ่มขึ้นอย่างเป็นเชิงเส้น

4.5 ความปลอดภัยให้กับระบบ

ในระบบของ EJB นั้นจะมีระบบรักษาความปลอดภัยที่ไคลเอ็นต์จะต้องผ่านก่อนที่จะขอใช้บริการจาก EJB ได้ คือ

- ไคลเอ็นต์จะต้องถูก authenticate โดยการ authenticate นั้นจะเป็นการเป็นการพิสูจน์ว่าไคลเอ็นต์นั้นเป็นคนที่ถูกระบุไว้ว่าตนเองนั้นเป็นบุคคลที่ถูกต้อง เช่น ไคลเอ็นต์อาจจะใส่ username/password ใน web browser เพื่อตรวจสอบกับ profile ของไคลเอ็นต์ที่เก็บในดาต้าเบสหรือ LDAP server และเมื่อใดที่ไคลเอ็นต์ถูก authenticate แล้ว หลังจากนั้นมันจะมี security identity ไปตลอด session ของมัน
- ไคลเอ็นต์จะต้องถูก authorize หลังจากที่ไคลเอ็นต์นั้นถูก authenticate แล้ว นอกจากนี้ยังจะต้องได้รับอนุญาตในการทำโอเปอเรชันต่างๆ เช่น ในการสั่งซื้อสินค้า ใครก็สามารถสั่งซื้อสินค้าได้ แต่เฉพาะ supervisor เท่านั้นที่จะสามารถเข้าไปตรวจสอบรายการสั่งซื้อสินค้าได้

บทที่ 5

Enterprise JavaBeans

Enterprise JavaBeans ใช้หลักการ devide-and-conquer ในการแบ่งความรับผิดชอบในการสร้างระบบทั้งหมดออกเป็นส่วน ๆ โดยแบ่งเป็น 6 ส่วนโดยทั้ง 6 ส่วนนี้รับผิดชอบโดยผู้ที่มีความเชี่ยวชาญในด้านของตนเองซึ่งการทำงานร่วมกันของฝ่ายต่าง ๆ ทั้ง 6 ฝ่ายซึ่งจะเป็นกุญแจสำคัญที่จะนำไปสู่ความสำเร็จของ Enterprise JavaBeans Deployment เพราะความเป็นมืออาชีพของทุก ๆ ฝ่าย เวลาที่ใช้ในการสร้าง Enterprise-class deployment จึงต่ำมากเมื่อเทียบกับโดยทั่ว ๆ ไป โดยสามารถดูความสัมพันธ์ในการทำงานร่วมกันของส่วนต่าง ๆ ทั้ง 6 ส่วนได้ดังรูป 5-1

1. Bean provider เป็นผู้สร้าง reusable business components ที่สามารถหาซื้อ และนำไปใช้ในธุรกิจของตนได้ ซึ่งในความเป็นจริงแล้วบีนไม่ใช่ application โดยสมบูรณ์ในตัวเอง แต่เป็น deployable component ที่สามารถนำไปประกอบกันเป็น application ที่สมบูรณ์พร้อมใช้ในธุรกิจต่าง ๆ ตัวอย่างของ bean provider ในปัจจุบัน เช่น บริษัท enterprise software ซึ่งประกอบด้วย SAP และ Trilogy จะออกผลิตภัณฑ์ของตัวเองเป็น enterprise bean หรือไม่ก็ออก connectors ที่จะมาเชื่อมต่อกับ software ของตนเอง

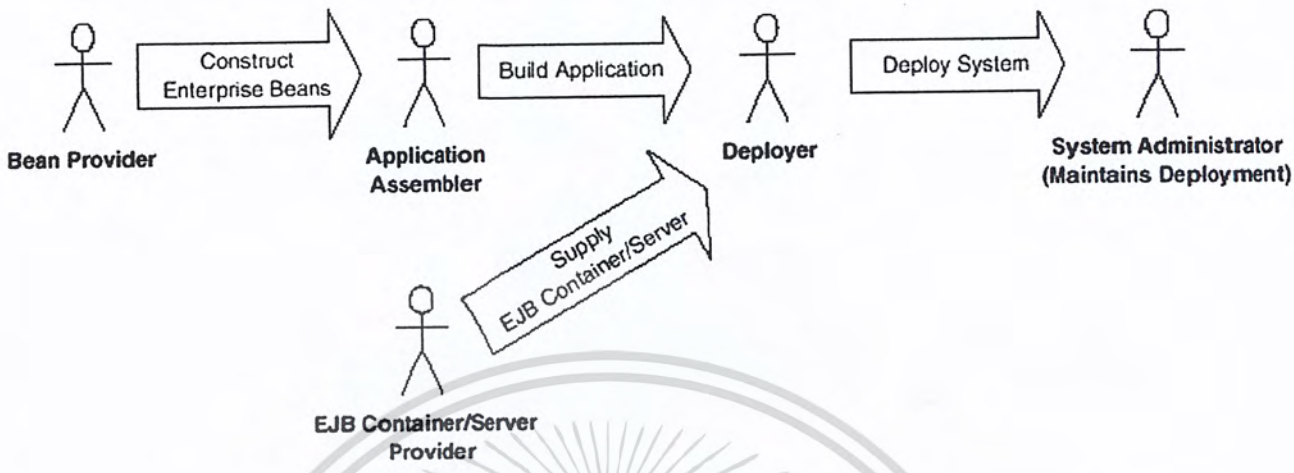
2. Container provider เป็นผู้รับผิดชอบจัดเตรียมสภาวะแวดล้อมระดับต่ำที่ EJB application ต้องการในขณะทำงาน

3. Server provider เป็นผู้รับผิดชอบ application server ที่จะเป็นที่บรรจุ บริหารจัดการคอมโพเน้นต์ต่าง ๆ โดยในขณะนี้ยังไม่มีข้อแตกต่างระหว่าง EJB container provider และ EJB server provider ตัวอย่างของ EJB container/server เช่น Weblogic จาก BEA Inc, NetDynamics จาก Sun Microsystems, Oracle 8i จาก Oracle และ Inprise Application Server จาก Inprise

4. Application assembler เป็นผู้มีความรับผิดชอบที่จะนำคอมโพเน้นต์ต่าง ๆ มาประกอบกันและเขียน application ที่ใช้ componenet เหล่านั้น โดยที่อาจจะต้องเขียนบีนบางบีนขึ้นมาเอง ตัวอย่างของ application assembler ก็เช่น system integrator หรืออาจจะเป็นโปรแกรมเมอร์ก็ได้

5. Deployer เป็นผู้นำคอมโพเน้นต์ที่ถูกเลือกจาก application assembler มาใช้งานประกอบกับ Application server แต่ละตัวที่มีอยู่ เพื่อช่วยในการติดตั้งคอมโพเน้นต์ให้สามารถทำงานในสภาวะแวดล้อมจริงได้

6. System administrator เป็นผู้ดูแลให้ระบบงานทั้งหมดทำงานร่วมกันไปได้อย่างราบรื่น



รูปที่ 5-1 ความรับผิดชอบในการสร้างระบบที่รองรับการใช้งานของ Enterprise Javabeans

5.1 ชนิดของบีนส์ (Type of Beans)

ใน EJB 1.0 และ 1.1 นั้นได้กำหนด enterprise beans ไว้ 2 ชนิดซึ่งประกอบด้วย session beans และ entity beans

5.1.1 Session Beans

session bean แสดงงานที่ถูกกระทำจาก code ของไคลเอ็นต์ที่เรียกใช้มัน session bean คือ ออบเจ็กต์ที่ใช้ในการประมวลผล โดยการทำ business logic, business rule, และ workflow ตัวอย่างเช่น การแจ้งราคาสินค้า, สั่งสินค้า, การฝากถอนเงิน, และอีกมากมาย ดังนั้นมันจึงเป็นคอมโพเน้นต์ที่บรรจุการทำงานต่าง ๆ ไว้ นอกจากนี้ Session bean ยังสามารถแบ่งออกเป็น 2 ประเภทซึ่งประกอบด้วย session bean แบบ stateful และ session bean แบบ stateless โดยจะกล่าวถึงในภายหลัง

sessioin bean ถูกเรียกว่า sessioin bean เพราะว่ามันมีอายุเท่ากับ session ของไคลเอ็นต์ที่เรียกใช้มัน ยกตัวอย่างเช่น ถ้าส่วน code ของไคลเอ็นต์ติดต่อกับ session bean เพื่อสั่งสินค้าซึ่งจะมี application server หรือ (EJB container/server) ในการรับผิดชอบการ instantiate session bean component นั้นขึ้นมาและเมื่อไคลเอ็นต์ ยกเลิกการติดต่อ ตัว application server ก็จะรับหน้าที่ทำลาย instance นั้น

ณ เวลาหนึ่ง session bean สามารถถูกเรียกใช้จากไคลเอ็นต์เพียงหนึ่งตัวเท่านั้น นั่นคือจะไม่มีการใช้ session bean ร่วมกันระหว่างไคลเอ็นต์ต่าง ๆ ที่เข้ามาขอบริการ

EJB server เป็นผู้รับผิดชอบอายุการใช้งานของบีนทุกบีนนั่นคือไคลเอ็นต์ไม่ได้ intantiate bean โดยตรงแต่ EJB container จะเป็นผู้ทำให้โดยอัตโนมัติและ EJB container จะเป็นผู้ทำลาย session bean ด้วยเมื่อถึงเวลาที่เหมาะสมจะทำให้บีนถูกจัดสรรและนำกลับมาใช้ใหม่กับหลาย ๆ ไคลเอ็นต์ที่เข้ามาได้

5.1.1.1 Stateful Session Beans

business process บางอย่างสามารถทำได้ใน method เดียว เช่น การคำนวณราคาสินค้า, การตรวจสอบยอดบัตรเครดิต แต่ในงานบางอย่างต้องใช้หลาย method ร่วมกัน และบางทีอาจเกี่ยวกับ transaction ด้วย

ร้านค้าบน web เป็นตัวอย่างหนึ่งของ business process ที่ไม่สามารถทำงานให้เสร็จภายใน 1 method ได้ลูกค้าเป็นผู้เลือกสินค้าใส่ในรถเข็นออนไลน์ซึ่งในการเพิ่มหรือลดสินค้าในรถเข็นนั้นจะต้องขึ้นอยู่กับ สถานะปัจจุบันของรถเข็นของลูกค้าแต่ละคนด้วย

อีกตัวอย่างหนึ่ง ก็คืองานธนาคาร โดยในงานธนาคารนั้นเราอาจจะมี code ที่ทำหน้าที่เป็นพนักงานธนาคารที่ติดต่อกับลูกค้าแต่ละคนในระยะเวลาหนึ่ง ๆ โดยที่ code นั้นอาจจะทำ transaction ให้กับลูกค้าแต่ละคน เช่น เช็คยอดเงินฝาก, โอนเงิน, ถอนเงิน

stateful session bean เป็นบีนที่ถูกออกแบบมาสำหรับให้บริการ business process ที่ต้องทำหลาย ๆ method หรือหลาย ๆ transaction โดยที่ stateful session bean จะคงสถานะของลูกค้าแต่ละคนไว้ตลอดเวลาที่ถูกใช้งาน

5.1.1.2 Stateless Session Beans

business process บาง business process ก็มีรูปแบบเป็นการขอบริการจากลูกค้าไปยังผู้ให้บริการเพียงครั้งเดียวและเนื่องจากมีการขอบริการเพียงครั้งเดียวจึงไม่จำเป็นจะต้องคงสถานะไว้เพื่อไว้ใช้ในการขอบริการครั้งต่อไป ดังนั้น stateless sesstion bean จึงเป็นคอมโพเน้นต์ที่เหมาะสมกับการใช้งานใน business process ประเภทนี้ โดย stateless session bean นั้นจะให้บริการกับผู้ที่ร้องขอเข้ามาโดยไม่สนใจสถานะเดิมใด ๆ ทั้งสิ้น

ตัวอย่างของ stateless session bean อาจจะเป็นบีนที่ไว้ใช้ในการแก้ไขปัญหาทางคณิตศาสตร์ที่มีความซับซ้อนจากอินพุทที่รับเข้ามาเช่น การบีบอัดข้อมูลภาพและเสียงบีนจะรับข้อมูลดิบ และอัตราการบีบอัดเข้ามาทำการบีบอัดแล้วส่งผลลัพท์ที่ได้กลับไปให้ผู้เรียกใช้ แล้วก็ไปให้บริการคนอื่นต่อ business process ทั้งหมดอยู่ภายใน method เดียวบีนไม่ต้องเก็บสถานะใด ๆ จากการเรียกใช้ครั้งที่แล้วไว้

ตัวอย่างอื่นก็เช่น stateless session bean ที่เป็นคอมโพเน้นต์ที่ทำหน้าที่ตรวจสอบบัตรเครดิต โดยบีนจะรับเอาหมายเลขบัตรเครดิต, วันหมดอายุ, ชื่อผู้ใช้ และยอดเงิน เมื่อบีนตรวจสอบข้อมูลทั้งหมดแล้วก็จะส่งผลลัพท์กลับไปให้ผู้ที่เรียกใช้ แล้วก็ไปให้บริการแก่คนอื่นต่อโดยที่ไม่ต้องเก็บข้อมูลของลูกค้าคนที่แล้วไว้ภายในบีนนั้นเลย

5.1.2 Entity Beans

ส่วนหลัก ๆ อีกส่วนหนึ่งที่ระบบงานส่วนใหญ่มีกันก็คือ ส่วนของข้อมูลทางธุรกิจที่จะถูกนำไปใช้ใน business process ดังตัวอย่างเช่น

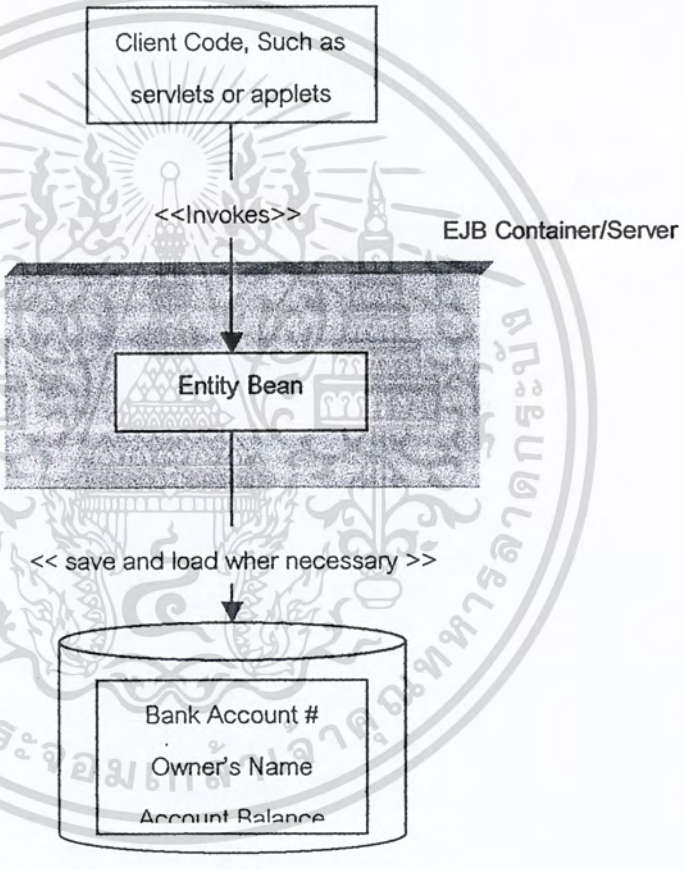
- คอมโพเน้นต์ที่ทำหน้าที่พนักงานธนาคาร ทำการฝากเงิน, ถอนเงิน, โอนเงิน แต่ข้อมูลทั้งหมดเป็นข้อมูลที่มาจากบัญชีธนาคาร

-คอมโพเน้นต์ที่ทำหน้าที่รับรายการสั่งซื้อสินค้าทำหน้าที่ submit รายการสินค้าที่ลูกค้าสั่งเช่น การสั่งซื้อเครื่องคอมพิวเตอร์ ข้อมูลที่ถูกสร้างขึ้นมาจากการรับสั่งสินค้าจะต้องประกอบด้วยจำนวนชิ้น

ส่วนต่างๆที่จะนำมาประกอบเป็นเครื่องคอมพิวเตอร์เพื่อที่จะส่งต่อไปให้คอมโพเน้นต์อื่นต่อไป

ในทุก ๆ สถานะการณ์ที่กล่าวมานี้จะมีบีนที่ทำหน้าที่เปลี่ยนแปลงข้อมูลที่อยู่ใน data storage เช่น ในฐานข้อมูล entity bean ก็คือคอมโพเน้นต์ที่เป็นตัวแทนของข้อมูลเหล่านั้น เช่น entity bean บัญชี, entity bean การสั่งสินค้า entity bean จะเป็นตัวแทนของ dataออบเจ็กต์ที่อยู่ในโลกของความต่าง

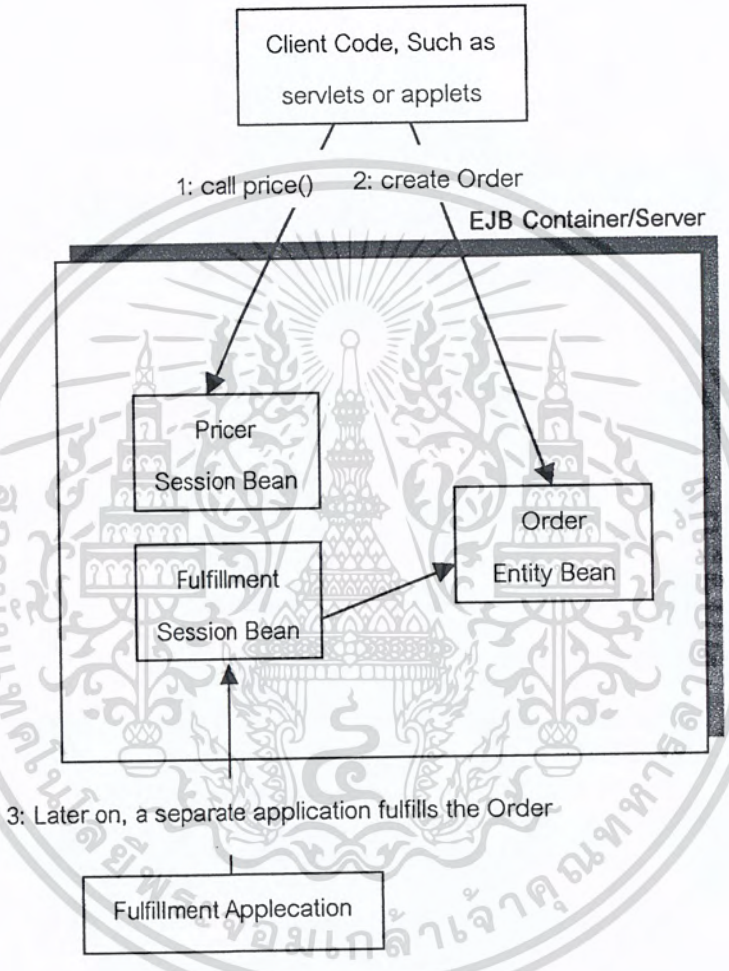
ในการสร้าง entity bean นั้นเราจะไม่ใส่ business process ไว้ใน entity bean เนื่องจากมันเหมาะสมที่จะเอาไว้ใส่ข้อมูลมากกว่าส่วนตัว session bean นั้นจะเป็นตัวบรรจุ business process ไว้ แล้วอาจจะมีการใช้ entity bean เป็นตัวแทนข้อมูลที่มันติดต่อด้วย เหมือนกับพนักงานธนาคารใช้ข้อมูลจากบัญชีธนาคาร



รูปที่ 5-2 entity bean จะใช้ data access logic layer ใน multi-tier architectures

ประโยชน์ของ entity bean ก็คือ ทำให้เราสามารถมองข้อมูลที่อยู่ใน data store เป็นข้อมูลแบบ object-oriented (ในเวลาที่มันถูก load ขึ้น memory) ซึ่งถ้าเป็นเมื่อก่อนขณะที่ application จะทำการเรียกใช้ข้อมูลก็ต้องลงไปยุ่งกับตารางในฐานข้อมูลในการสั่งให้อ่านข้อมูล, เขียนข้อมูลแต่ entity bean ทำให้เราสามารถมองข้อมูลทั้งหมดเป็นออบเจ็กต์เราสามารถทำงานกับข้อมูลที่อยู่ในฐานข้อมูลได้เหมือนเป็นออบเจ็กต์ตัว

อย่างเช่น เราสามารถอ่านข้อมูลบัญชีที่ต้องการออกมาจากฐานข้อมูลแล้วนำไปใส่ไว้ใน entity bean เราสามารถเรียกใช้ method withdraw (ถอนเงิน) เพื่อที่จะลดค่าของตัวแปร balance ซึ่งเป็นชนิด private ที่อยู่ภายใน entity bean ได้โดยที่ entity bean จะมีคุณสมบัติ persistent คือรับประกันว่าข้อมูลในฐานข้อมูลจะถูกเปลี่ยนแปลงแล้ว แน่ ๆ และจะอยู่ตลอดไป ดังนั้นจึงเป็นการสะดวกต่อนักพัฒนาอย่างมากที่จะได้คุณสมบัติ persistent และ encapsulation ในขณะเดียวกัน entity bean จะใช้ data access logic layer ใน multi-tier architectures



รูปที่ 5-3 แสดงการใช้งานร่วมกันของ session bean และ entity bean ใน Pricing Application

ความแตกต่างอย่างชัดเจนของ entity bean กับ session bean อีกอย่างก็คือ entity bean สามารถให้บริการไคลเอ็นต์หลาย ๆ ไคลเอ็นต์ได้พร้อมกันดังเช่นการที่ไคลเอ็นต์หลาย ๆ ไคลเอ็นต์เข้าถึงข้อมูลในฐานข้อมูลพร้อมกันโดยที่เราสามารถจะทำให้ไคลเอ็นต์แต่ละไคลเอ็นต์ทำงานได้โดยเหมือนกับทำงานอยู่เพียงคนเดียวโดยใช้ transaction

ในความเป็นจริงแล้วเราอาจจะมีข้อมูลในฐานข้อมูลอยู่ก่อนแล้วก็ได้ ซึ่งจะเป็นประโยชน์มากสำหรับEJB เพราะทำให้เราสามารถนำข้อมูลในฐานข้อมูลไปใช้เป็นออบเจ็กต์ในภาษาจาวา (Java) ได้โดย

ที่การแปลงข้อมูลนี้อาจเกิดขึ้นในบีนหรือ อาจให้ EJB container ทำให้โดยอัตโนมัติก็ได้

entity bean แบ่งย่อยได้เป็น 2 ประเภท คือ bean-managed persistent และ container-managed persistent โดยมีรายละเอียดดังนี้

5.1.2.1 Bean-Managed Persistent Entity Beans

entity bean เป็นคอมโพเน้นต์ที่มีคุณสมบัติ persistent เพราะว่าสถานะของบีนจะถูกบันทึกลงใน secondary storage เช่น ฐานข้อมูลแบบ relational ตัวอย่างเช่น เมื่อทำการ object-relational mapping แล้ว เราจะสามารถ map ออบเจ็กต์ที่อยู่ในหน่วยความจำ (memory) ไปเก็บเป็น record ในฐานข้อมูลได้ และในทางกลับกันเราสามารถนำ record เหล่านั้นมาสร้างออบเจ็กต์นั้น ๆ ขึ้นมาใหม่ เราอาจจะสามารถใช้ออบเจ็กต์–ดาต้าเบส (object database) ได้ โดยจะทำให้เราสามารถเก็บออบเจ็กต์ลงในฐานข้อมูลจริง ๆ ได้แทนที่จะเป็น record

entity bean แบบ bean-managed persistent เป็น entity bean ที่ผู้พัฒนาต้องทำการเก็บข้อมูลด้วยตนเอง หมายความว่าใน code ของบีนของเราจะต้องมีส่วนที่ทำการ map field ต่าง ๆ ในออบเจ็กต์ไปเป็น field ต่าง ๆ ใน record ในฐานข้อมูลซึ่งจะต้อมีการเตรียม code สำหรับการอ่าน, การเขียน หรือการค้นหาข้อมูล โดยผ่านทาง persistent API เช่น JDBC หรือ SQL/J

5.1.2.2 Container-Managed Persistent Entity Beans

EJB ทำให้นักพัฒนามีความสะดวกมากขึ้นโดยไม่ต้องพะวงกับการเขียน code ที่เกี่ยวกับการ persist ของข้อมูล EJB 1.1 จะทำให้ entity bean ของเราสามารถ persist ได้โดยอัตโนมัติเมื่อเราต้องการ

โดยที่ container/server จะทำการเขียน, การอ่าน หรือการค้นหาข้อมูลของคอมโพเน้นต์ให้ทุกอย่าง ทำให้เราไม่ต้องเขียน code ที่มี API ข้อฐานข้อมูลแต่ละยี่ห้อที่จะทำให้ประหยัดเวลาในการเขียนบีนและทำให้บีนนั้นไม่ยึดติดกับฐานข้อมูล เป็นการเพิ่มความ reusable เราเพียงแค่กำหนดว่าเราต้องการให้ field ไหน persist ต่อ container เท่านั้นหลังจากนั้น container ก็จะจัดการให้โดยอัตโนมัติ

ในตลาดปัจจุบัน container/server มีการ map หลากหลายรูปแบบและ tools ที่ใช้ในการ map ก็มีหลายตัว บางตัวก็เป็นการ map แบบง่าย ๆ บางตัวก็มีฟังก์ชันการ map ที่ซับซ้อน เช่น Weblogic server จาก BEA ซึ่งสามารถทำการ map object/relational แบบง่าย ๆ ได้

5.2 Clustered EJB

EJB ทุกตัวนั้นสามารถ run ได้กับทุกเซิร์ฟเวอร์ที่อยู่ภายใน cluster โดยถึงแม้ว่า EJB ถูก deploy บน เซิร์ฟเวอร์หลาย ๆ ตัวแต่ละ EJB นั้นก็สามารถอยู่บนเซิร์ฟเวอร์ได้ทั้งหมด

5.2.1 EJB Homes

Home ของบีนทุก ๆ บีนก็สามารถ run ได้ในทุกเซิร์ฟเวอร์เช่นกัน โดย home ที่อยู่ในแต่ละเซิร์ฟเวอร์ ซึ่งอยู่ใน cluster นั้นจะใช้ชื่อเดียวกันเมื่อไคลเอ็นต์ทำการ look up home , stub จะทำการอ้างอิงไปยัง home ตัวใดตัวหนึ่งบนเซิร์ฟเวอร์ที่ได้ทำการ deploy บีนนั้นไว้

5.2.2 Stateless EJB

เมื่อ home ทำการสร้าง (create) stateless bean ขึ้นมาแล้ว home จะรีเทิร์น stub ที่สามารถใช้ได้กับ ทุก ๆ เซิร์ฟเวอร์ที่มีบีนนั้น deploy อยู่ได้เนื่องจาก stateless bean จะไม่เก็บ state ของไคลเอ็นต์ไว้

stub ของ stateless EJB นี้ไม่เป็น idempotent นอกจากเราจะต้องกำหนด idempotent method เอาไว้ใน deployment descriptor ไว้เอง มันจึงจะจัดการให้

5.2.3 Stateful EJB

เมื่อ home ทำการ create stateful bean ขึ้นมันก็จะทำการ return stub กลับไปยังไคลเอ็นต์ที่เป็นคนเรียกนั้นเนื่องจาก stateful EJB จะเก็บ state ของไคลเอ็นต์ไว้ดังนั้นในทุก ๆ การ call ที่เข้ามาจะต้องทำการหาเส้นทางไปยัง instance ที่อยู่บนเซิร์ฟเวอร์ตัวเดียวกันทุกครั้ง

ในกรณีของ stateful bean การเกิด load balancing และการ fail สามารถเกิดขึ้นได้เพียงขณะทำการ call ไปยัง home ซึ่งสำหรับในการร้องขอของบีนแบบ stateful นั้นจะต้องร้องขอไปยังเซิร์ฟเวอร์ตัวเดียวเท่านั้นถ้ามีการ fail เกิดขึ้น ไคลเอ็นต์จะต้อง catch remote exception เอาไว้ เมื่อต้องการติดต่อใหม่ต้องทำการ create instance ตัวใหม่โดยใช้ home ที่มีอยู่เดิม

5.2.4 Entity EJB

Entity ที่นิยมใช้มีอยู่ 2 ชนิดคือ read-write entity และ read-only entity เมื่อ home ทำการค้นหาหรือ create entity read-write ก็จะมี instance เกิดขึ้นและจะ return stub กลับไป การ load balancing และการ fail จะเกิดขึ้นเพียงในขณะติดต่อกับ home เท่านั้นเนื่องจากว่าภายใน cluster สามารถมี entity ได้หลายตัว และแต่ละ instance จะต้องทำการข้อมูลจากฐานข้อมูลก่อนที่จะมีการ commit transaction เพราะจะทำให้ข้อมูลมีการเปลี่ยนแปลง

และเมื่อ home ทำการค้นหาหรือ create read-only entity bean จะ return stub กลับไป stub นี้จะทำหน้าที่กระจายงานให้เซิร์ฟเวอร์แต่ละเท่า ๆ กัน และเมื่อมีการ fail เกิดขึ้น read-only bean จะทำการ cache เซิร์ฟเวอร์ทุกตัวไว้เพื่อป้องกันไม่ให้มีการอ่านข้อมูลจากฐานข้อมูล

บทที่ 6

JavaServerPage Component

6.1 จาวาเซิร์ฟเวอร์เพจ (JavaServer Pages : JSP)

เป็นเทคโนโลยีในการสร้างหน้าเวบ (Web Pages) โดยถูกออกแบบมาเพื่อให้ง่ายและเร็วต่อการสร้างแอพพลิเคชั่นที่เป็น Web-based ที่สามารถทำงานได้กับหลากหลายเว็บเซิร์ฟเวอร์ (WebServers), แอพพลิเคชั่นเซิร์ฟเวอร์(ApplicationServers), บราวเซอร์(Browser) หรือเครื่องมือในการพัฒนาอื่นๆ (Developement Tools)

6.2 ประวัติการพัฒนาแอพพลิเคชั่นที่เป็น Web-based

วิวัฒนาการของ World Wide Web มีวิวัฒนาการมาจากการแสดงข้อมูลเพื่อการค้าหรือก็คือ "E-Commerce" แอพพลิเคชั่นที่สามารถใช้ browser-based Clients มีข้อดีหลายอย่าง เช่น Client/Server-based application จะไม่จำกัดการเข้าถึงของไคลเอ็นต์, การจัดการและการ deploy ที่สามารถทำได้ง่ายขึ้น ในการอัพเดต (update) แอพพลิเคชั่นผู้พัฒนาเพียงแต่แก้ไขโปรแกรมบนเซิร์ฟเวอร์ โดยไม่จำเป็นต้องติดตั้งแอพพลิเคชั่นบนไคลเอ็นต์ใหม่ ซึ่งทำให้อุตสาหกรรมซอฟแวร์สามารถพัฒนาไปได้อย่างรวดเร็วในการสร้างแอพพลิเคชั่นที่เป็นหลายเทียร์โดยใช้ browser-based client

เมื่อมีแอพพลิเคชั่นที่เป็น Web-based เพิ่มมากขึ้นก็ต้องการการแปลี่ยนแปลงเทคโนโลยีในการพัฒนา HTML จึงเป็นทางเลือกในการแสดงรายละเอียดของเอกสาร จึงได้มีความพยายามในการพัฒนาให้หน้าของเวบนั้นขึ้นอยู่กับความต้องการของผู้ใช้หรือสถานะของระบบ

ต่อมาจึงได้เกิด CGI ขึ้นโดยผู้พัฒนาจะเป็นผู้เขียนโปรแกรมขึ้นและ Web-based application จะเรียกโปรแกรมผ่านเว็บเซิร์ฟเวอร์ โดยการทำเช่นนี้จะทำให้เกิดความยืดหยุ่นมากขึ้นคือในการเรียก CGI แต่ละตัวจะทำให้เกิด Process ใหม่บนเซิร์ฟเวอร์ ถ้าผู้ใช้หลายๆคนเข้าถึงโปรแกรมพร้อมๆกันจะมีการแย่งทรัพยากร (Resource) กันทำให้การทำงานช้าลง

ในผู้ขายเว็บเซิร์ฟเวอร์แต่ละรายก็ได้มีความพยายามที่จะพัฒนาแอพพลิเคชั้นบนเว็บให้ทำได้ง่ายขึ้นโดยการสร้าง "Plug-ins" และ APIs ขึ้นบนเซิร์ฟเวอร์ แต่ปัญหาก็คือไม่สามารถใช้ผลิตภัณฑ์ข้ามระหว่างแต่ละผู้ขายได้ เช่น เทคโนโลยี Active Server Pages ของ Microsoft ทำให้การสร้างหน้าเว็บทำได้ง่ายขึ้นแต่ต้องทำงานร่วมกับ Personal Web Server หรือ Microsof Iist เท่านั้น

เทคโนโลยีที่เกิดขึ้นอีกก็คือ Java Servlets ทำให้การเขียนโค้ดบนฝั่งเซิร์ฟเวอร์สามารถทำได้ง่ายขึ้น สามารถติดต่อ Servlet ได้โดยการสร้างแอพพลิเคชั่นด้วยภาษาจาวาติดต่อโดยจะเป็นโปรแกรมที่รันบนฝั่งเซิร์ฟเวอร์ ตรงข้ามกับ applet ที่รันบนบราวเซอร์ ผู้พัฒนาสามารถเขียน Servlets เพื่อรองรับ HTTP ที่ส่งมาจากเว็บบราวเซอร์ได้ (อาจต้อง query จากฐานข้อมูล) และ Servlet จะตอบสนองกลับไปโดยการส่งในรูปของเอกสาร HTML หรือว่า XML กลับไปยังบราวเซอร์

เมื่อลำดับการทำงานเป็นดังที่กล่าวมาแล้ว หน้าของเว็บเพจทุกหน้าจะถูกประกอบขึ้นสมบูรณ์ภายใน Servlet ถ้าต้องการที่แก้ไขโครงสร้างของหน้าเวบเพจจะต้องไป edit และคอมไพล์ Servlet ใหม่ ดังนั้นคุณสมบัติที่สำคัญของ servlet ในการสร้างหน้าเว็บเพจมีดังนี้

- Servlet สามารถทำงานได้กับเว็บเซิร์ฟเวอร์และแอพพลิเคชั่นเซิร์ฟเวอร์หลายยี่ห้อ
- จะต้องแยกหน้าเว็บในส่วนแอพพลิเคชั่นลอจิกออกจากหน้าเว็บเพจที่เห็นได้
- Process สามารถติดต่อด้วยได้ง่าย เช่น Web-base Application

6.3 เทคโนโลยีของ JavaServer Pages ที่เป็นการพัฒนาแอพพลิเคชั่นบนเว็บ

บริษัทซันร่วมกับผู้นำทางด้านเวบเซิร์ฟเวอร์ (Web Server), แอพพลิเคชั่นเซิร์ฟเวอร์ (Application Server) ได้ร่วมมือกันพัฒนาจาวาเวบเซิร์ฟเวอร์ขึ้น โดยเทคโนโลยีของ JavaServer Pages จะเป็นพัฒนาหน้าเว็บเพจในลักษณะที่เป็นไดนามิก คือมีลักษณะดังนี้คือ

1. แยกโปรแกรม (Content) ออกจากส่วนแสดงผล (Presentation)

ในเทคโนโลยีของ JSP ผู้พัฒนาจะใช้ tag HTML หรือ XML ในการออกแบบหน้าเวบ (ส่วน Presentation) และใช้ tag JSP หรือ Scriptlet ในการสร้างส่วนของโปรแกรม (Dynamic Content) คือ การแยกส่วนแสดงผลกับส่วนของโปรแกรมแยกออกจากกัน จะทำให้เราสามารถแสดงผลข้อมูลที่ต้องการได้โดยไม่ต้องสร้างหน้าเวบใหม่ทั้งหมดเพราะจะเขียนโปรแกรมเป็นตัวประมวลผลและส่งไปแสดงผลในส่วน Presentation แทน โดยลอจิกต่างๆของโปรแกรมที่เราเขียนนั้นจะถูกซ่อน (Encapsulate) ไว้ใน tag และทั้งส่วน Presentation และส่วนโปรแกรมทั้งหมดนี้จะถูกรวมกันเรียกว่า “สคริปต์” (Script) สคริปต์ทั้งหมดนี้จะทำการประมวลผลที่ฝั่งเซิร์ฟเวอร์ และเวบมาสเตอร์ (Webmaster) จะสามารถทำการแก้ไขหน้าเวบ (ส่วน Presentation) ได้โดยจะไม่กระทบต่อส่วนของโปรแกรม

บนฝั่งเซิร์ฟเวอร์ JSP เอนจิน (JSP engine) จะตีความ tag JSP และสคริปต์ เมื่อทำการประมวลผลตามลอจิกในโปรแกรมแล้วก็จะส่งเอาพุตท์กลับไปยังบราวเซอร์ในรูปของ HTML หรือ XML ในการทำงานเช่นนี้จะเป็นการป้องกันการเข้าถึงโค้ดและยังสามารถติดต่อไปยังเวบบราวเซอร์ได้หลายตัวอีกด้วย

2. สามารถนำคอมโพเนนท์แต่ละส่วนกลับมาใช้ใหม่ได้

ความสามารถนี้เป็นความสามารถที่สำคัญของ JSP และในการประมวลที่ซับซ้อนสามารถข้ามแพลตฟอร์มกันได้ ทำให้ในระบบที่มีผู้ใช้จำนวนมาก ผู้พัฒนาสามารถใช้คอมโพเนนท์ร่วมกันหรือว่าแลกเปลี่ยนคอมโพเนนท์กันได้

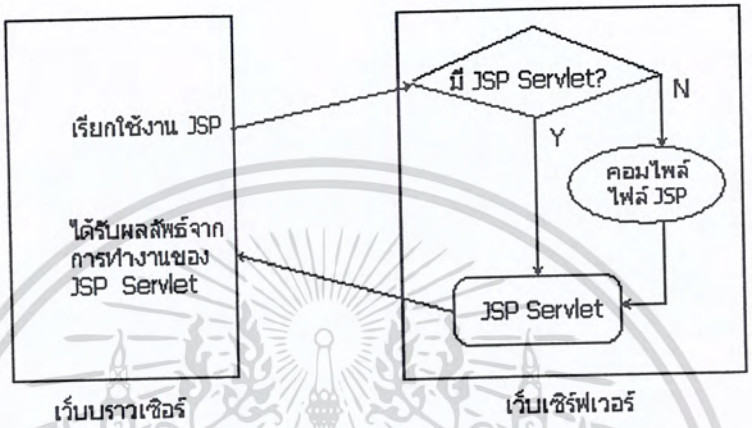
3. ทำให้การพัฒนาเวบสามารถทำได้ง่ายขึ้น

เนื่องจากผู้พัฒนาเวบส่วนใหญ่นั้นจะไม่ใช่โปรแกรมเมอร์ที่คุ้นเคยกับการใช้ภาษาสคริปต์มากนัก JSP จะสร้างฟังก์ชั่นที่จำเป็นในการใช้งานไว้พร้อมให้ใช้ ทำให้สามารถใช้งานได้ง่าย นอกจากนี้ JSP ยังอนุญาตให้ผู้ใช้สามารถสร้าง tag library ขึ้นได้เอง และยังสนับสนุนการทำงานของแอพพลิเคชั่นในระดับเอนเทอร์ไพซ์ด้วย

JSP ได้รวมความสามารถต่างๆของเทคโนโลยีจาวาเอาไว้ได้แก่ มีการจัดการหน่วยความจำที่ดี, มีระบบรักษาความปลอดภัยที่ดี เพราะภาษาสคริปต์จะมีพื้นฐานมาจากภาษาจาวาอยู่แล้วและ JSP นั้นเวลา

คอมไพล์จะคอมไพล์ใน Servlets และ JSP ยังมีลักษณะที่เรียกว่า Write Once Run Anywhere คือสร้างขึ้นเพียงครั้งเดียวแต่สามารถรันได้ในทุกที่ซึ่งเป็นคุณสมบัติของภาษาจาวาอีกด้วยโดยลักษณะของ JSP นั้นจะคล้ายกับ HTML หรือ XML

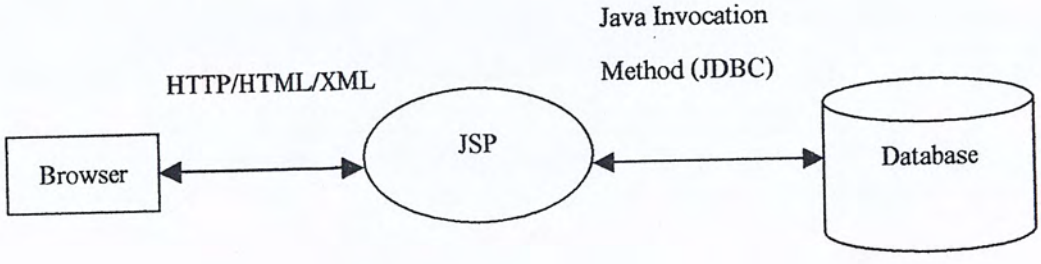
6.4 การทำงานของ JSP



รูปที่ 6-1 แสดงการทำงานของ JSP

เมื่อเว็บบราวเซอร์ มีการร้องขอโดยผู้ใช้เพื่อเรียกข้อมูล JSP ที่อยู่บนเซิร์ฟเวอร์ ซึ่งก็หมายถึงไฟล์ที่มีนามสกุล JSP เว็บเซิร์ฟเวอร์จะทำการตรวจสอบว่า ไฟล์นั้นมีการคอมไพล์ไปเป็น JSP Servlet แล้วหรือยัง ถ้ายังไฟล์ JSP จะถูกคอมไฟล์ โดยตัวเว็บเซิร์ฟเวอร์ซึ่งมี JSP Engine ไปเป็น JSP Servlet แต่ถ้าไฟล์ถูกคอมไพล์แล้ว (ซึ่งเกิดขึ้นจากการใช้งานในครั้งแรก) เว็บเซิร์ฟเวอร์จะนำ JSP Servlet มาใช้งานโดยการเรียกเพื่อประมวลผล ได้ผลลัพธ์อย่างไรก็จะเกิดเป็นข้อมูล ซึ่งเว็บเซิร์ฟเวอร์ จะส่งกลับไปให้ผู้เรียกใช้งาน และการประมวลผลของ JSP Servlet นี้เอง ที่ก่อให้เกิด ข้อมูลเว็บในแบบไดนามิก

บราวเซอร์ (Browser) จะเรียกใช้ JSP ซึ่งจะสร้างหน้าเว็บที่บราวเซอร์ต้องการ โดย JSP อาจจะต้องทำการร้องขอข้อมูลไปยังฐานข้อมูลโดยตรง JSP สามารถเรียกไปยังฐานข้อมูลโดยผ่าน JDBC หรือ Java Blend component เพื่อนำผลลัพท์มาจัดการแสดงผลเป็น HTML ส่งกลับไปให้บราวเซอร์



รูปที่ 6-2 แสดงการติดต่อกับฐานข้อมูลของ JSP

จากรูป 6-2 นี้เป็นการแทนที่แนวคิด (concept) ของ CGI-bin ด้วย JSP ซึ่งมีข้อได้เปรียบเหนือกว่า คือ ง่ายในการเขียนและใช้เวลาพัฒนาน้อย โดยโมเดลนี้ทำงานได้ดีกับหลาย ๆ แอปพลิเคชัน (Application) แต่ก็ยังมีข้อจำกัดสำหรับงานที่มีไคลเอ็นต์เรียกใช้ทรัพยากรพร้อมกันเป็นจำนวนมากๆ เช่น ใช้ข้อมูลจากฐานข้อมูล จะทำให้ต้องสร้างแต่ละการเชื่อมต่อสำหรับแต่ละไคลเอ็นต์ทำให้เป็นการลดประสิทธิภาพการทำงานของฐานข้อมูลอย่างมาก

6.4.21 การทำงานของ JSP ร่วมกับ Enterprise JavaBeans

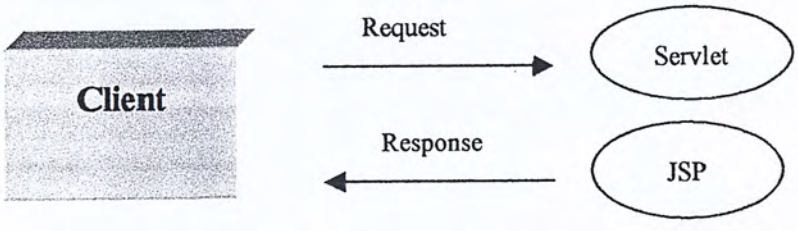
JSP สามารถเป็นมิดเดิลเทียร์ (middle tier) ให้กับ Enterprise JavaBeans ได้โดย JSP จะเป็นตัวแทนติดต่อ back-end resources ผ่านคอมโพเน้นต์ของ Enterprise JavaBeans ได้ดังรูป 6-3

รูปที่ 6-3 การทำงานของ JSP ร่วมกับ Enterprise JavaBeans

6.4.2 การทำงานของ JSP ร่วมกับ Java Servlet

JSP (Java Server Page) กับ Servlet นั้นจะมีความแตกต่างกันตรงวิธีการเขียน และการนำไปใช้เท่านั้นเอง ส่วนประสิทธิภาพของทั้งคู่จะมีความใกล้เคียงกันมาก สำหรับ JSP เมื่อเราเขียนเสร็จ มันจะเรียก JSP Container ซึ่งจะมีการนำ JSP engine มาใช้ในการคอมไพล์ให้เป็นแบบ servlet (ซึ่งสุดท้าย JSP ก็ต้องกลาย เป็น servlet อยู่ดี) แต่สำหรับ servlet เราต้องจัดการคอมไพล์เองโดยต้องมี servlet engine มาใช้ในการคอมไพล์โดยโปรแกรมจะไม่ต้องถูกคอมไพล์ซ้ำเหมือน JSP

รายละเอียดในการทำงานร่วมกันนั้นไคลเอ็นต์อาจจะขอบริการโดยตรงไปยัง Java Servlet ซึ่งจะเป็นตัวสร้างไดนามิก-คอนเท็นต์ (dynamic content) และให้ผลลัพท์เป็นบีน (bean) แล้วเรียกไปยัง JSP ซึ่ง JSP จะนำไดนามิก-คอนเท็นต์ จากบีนนั้นมาจัดให้เหมาะแก่การแสดงผล (เป็น HTML) แล้วส่งไปยังบราวเซอร์



รูปที่ 6-4 แสดงการร้องขอบริการจากไคลเอ็นต์ไปยังเซิร์ฟเล็ต (Servlet)

วิธีนี้เป็นการสร้างคอมโพเน้นต์ (component) ที่สามารถนำกลับมาใช้ได้ในหลาย ๆ แอฟพลิเคชัน(Application) ซึ่งอาจนำไปใช้เป็นส่วนหนึ่งของแอฟพลิเคชันขนาดใหญ่ได้ และยังเป็นการทำให้สามารถจัดการกับทรัพยากรได้ดีขึ้นด้วย

บทที่ 7
การเชื่อมต่อกับฐานข้อมูล

7.1 การเชื่อมต่อกับฐานข้อมูล

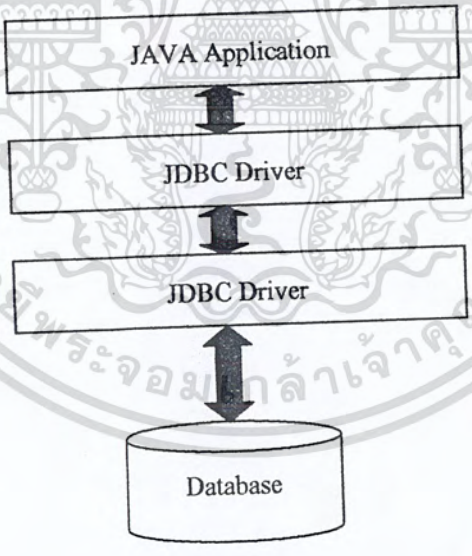
การเชื่อมต่อกับฐานข้อมูลด้วย JDBC จะมีรูปแบบหลักการในการทำงานดังนี้

Jdbc:<subprotocol>:<subname>

ในการเชื่อมต่อนั้นจะมีโปรโตคอล (protocol) หลักในการทำงานเป็น JDBC ซึ่งมีการทำงานแบบเดียวในการเชื่อมต่อกับตัว JAVA แต่จะแตกต่างกันที่ Sub Protocol ซึ่งเป็นตัวบอกชนิดของการเชื่อมต่อกับตัวแม่ข่ายของฐานข้อมูล และ Sub Name จะเป็นตัวบอกข้อมูลของการเข้าถึงฐานข้อมูลนั้น เช่น หมายเลขพอร์ท หรือ ชื่อของเครื่องแม่ข่าย สำหรับ Sub Protocol มีได้หลายแบบตามชนิดของการเชื่อมต่อ ซึ่งในโครงงานนี้จะใช้ Thin driver ในการเชื่อมต่อ

Thin driver เป็นรูปแบบดั้งเดิมของการเชื่อมต่อด้วย JDBC เป็นการเชื่อมต่อที่ทั้งด้านเครื่องลูกข่ายและเครื่องแม่ข่ายทำการติดต่อกันผ่านทางโปรแกรมที่พัฒนาโดย JAVA ทั้งหมด (Pure Java) โดยฐานข้อมูลส่วนใหญ่ที่สนับสนุน Java ก็จะสนับสนุนการเชื่อมต่อแบบนี้ ตัวอย่างการขอเชื่อมต่อโดยใช้ Thin driver เช่น

DriverManager.getConnection("jdbc:oracle:thin:@myhost:1521:orcl","scott","tiger");



รูปที่ 7-1 แสดงการเชื่อมต่อแบบ Thin-client

7.2 การจัดการกับทรานส์แอ็กชัน (Java Transaction)

ใน Distributed object operations นั้น การทำทรานส์แอ็กชัน (transaction) เป็นวิธีที่ปลอดภัยสำหรับการทำโอเปอเรชัน (operation) ที่เป็นแบบ multiple components สำหรับทรานส์แอ็กชันนั้นก็คือลำดับของคำสั่งซึ่งปรากฏในกระบวนการ execute operation ในการทำทรานส์แอ็กชันนั้นจะยินยอมให้ผู้

ใช้หลายคนสามารถใช้ข้อมูลร่วมกันได้ และยังสามารถยืนยันได้ว่าข้อมูลเหล่านั้นจะสมบูรณ์ตามที่ได้แก้ไขไว้ทั้งหมดหรือไม่ โดยข้อมูลดังกล่าวนั้นจะไม่ถูกสอดแทรกด้วยข้อมูลของผู้ใช้คนอื่น ๆ เลย

การทำทรานส์แอ็กชันจะยืนยันได้ว่า การทำโอเปอเรชัน 2 โอเปอเรชันจะเป็นอิสระต่อกัน และจะสามารถป้องกันการเสียหายที่เกิดจากดาต้าเบสเสียหายหรือ Crash ได้ ซึ่งหากไม่มีการทำทรานส์แอ็กชันแล้วการจัดการฐานข้อมูลอาจจะไม่มีประสิทธิภาพได้

7.3 รูปแบบของทรานส์แอ็กชัน (Transaction Model)

วิธีการต่าง ๆ ในการทำทรานส์แอ็กชันแต่ละโมเดลนั้นจะมีความซับซ้อนและมีลักษณะของแต่ละโมเดล โดยในการทำทรานส์แอ็กชันสำหรับโมเดลที่นิยมจะมี 2 รูปแบบก็คือ

- Flat Transactions
- Nested Transactions

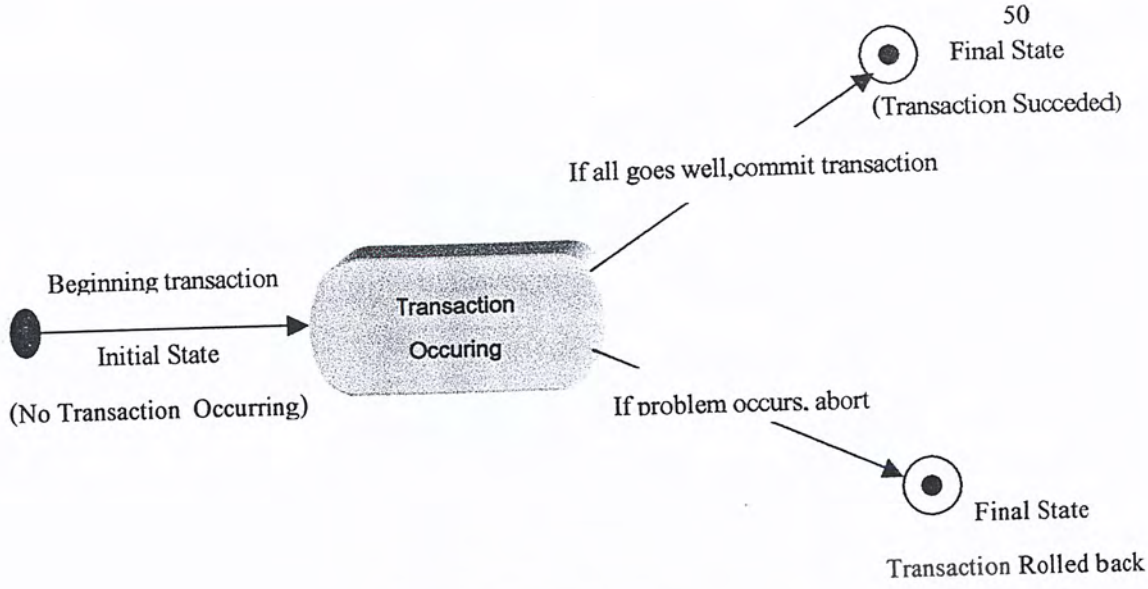
ในการใช้โมเดลดังกล่าวนั้น transaction service ต้องรองรับกับโมเดลแบบนั้นด้วย แต่ในปัจจุบันนี้ ไม่ใช่ผู้ขายทั้งหมดที่จะกำหนด EJB Specification ให้สามารถใช้ Nested transactions ได้ดังนั้น Enterprise JavaBeans ที่ประกาศเป็น Flat transaction ก็สามารถไม่รองรับการทำ Nested transaction ได้ แต่อาจมีการเปลี่ยนแปลงในอนาคตตามความต้องการที่เพิ่มมากขึ้น

7.3.1 Flat Transactions

Flat transaction เป็นโมเดลของทรานส์แอ็กชันที่ง่ายต่อการทำความเข้าใจ โดย Flat transactions ก็คือ ลำดับของโอเปอเรชันที่ถูกจัดการอย่างอัตโนมัติ แบบ Single unit of work หลังจากเริ่มต้นการทำ Flat transaction แล้วแอบพลิเคชัน (Application) ก็จะเริ่มทำโอเปอเรชันนั้น ๆ โดยบางโอเปอเรชันจะเป็น persistent operations และบางโอเปอเรชันก็ไม่ใช่ จนกระทั่งเมื่อจบการทำทรานส์แอ็กชัน ก็จะมีผลลัพธ์ออกมาคือ อาจจะสำเร็จหรือผิดพลาด โดยทรานส์แอ็กชันที่สำเร็จก็คือทรานส์แอ็กชันที่ committed แล้ว ขณะที่ทรานส์แอ็กชัน ที่ผิดพลาดก็คือทรานส์แอ็กชันที่ Aborted

เมื่อทรานส์แอ็กชันนั้น committed แล้วการ update ทั้งหมดที่ updates ไปยัง resource ต่าง ๆ เช่น ดาต้าเบสก็จะถูกเก็บอย่างถาวรเพียงแค่ทรานส์แอ็กชันนั้นจบด้วยการ commit แต่ถ้าทรานส์แอ็กชันนั้น aborted ก็จะไม่กระทำการใด ๆ กับ resource ที่ได้ทำการ update ไป และก็จะทำการ rolled back ในส่วนของ persistent operations ทั้งหมดที่แอปพลิเคชันจะต้องปฏิบัติ ก็จะไม่มีการทำอะไรทั้งสิ้น ซึ่งจะขึ้นอยู่กับระบบด้วย แอปพลิเคชันนั้นสามารถแจ้งกรณีของการเกิด Abort ได้ ดังนั้นแอปพลิเคชันจะไม่ทำการเปลี่ยนแปลงสิ่งใด ๆ ในหน่วยความจำขณะที่ทำทรานส์แอ็กชัน เรียกวิธีนี้ว่า all-or-nothing

"All-or-nothing" หากดูตามตัวอย่างของการฝากถอนเงินจากธนาคาร ก็จะหมายความว่าเราสามารถถอนเงินจากธนาคารหนึ่งได้และฝากในธนาคารอื่นได้โดยโอเปอเรชันเหล่านี้จะไม่เป็นทั้ง commit และ roll back ซึ่งสามารถอธิบายกระบวนการของ Flat transactions ได้ดังรูป 3-14



รูปที่ 7-2 The Flat Transaction

7.3.1.1 การ Rolled Back ของการทำทรานส์แอ็กชัน

ในกรณีที่เป็น Flat transaction model ซึ่งจะมีการรวบรวมโอเปอเรชันต่าง ๆ ไว้บน physical permanent resource เช่น ดาต้าเบส โดยหลังจากเริ่มทำทรานส์แอ็กชันแล้ว business component ก็จะเริ่มต้องการการติดต่อไปยังดาต้าเบส ดังนั้น database connection ก็จะถูกร้องขออย่างอัตโนมัติในทรานส์แอ็กชันที่คอมโพเน้นต์นั้นเกี่ยวข้อง จากนั้นคอมโพเน้นต์จะกระทำในส่วนของ persistent operation เช่น การ update ข้อมูลในดาต้าเบสแต่ถ้าการจัดการ resource นั้นไม่สามารถทำได้ อาจจะต้องรอจนกระทั่งมีการ commit ก่อน ซึ่งจะมีการ commit ได้เมื่อ business component นั้น ๆ ได้มีการกระทำโอเปอเรชันต่าง ๆ ที่อยู่ภายใต้ทรานส์แอ็กชันเสร็จเรียบร้อยแล้วเท่านั้น

ในอีกประเด็นหนึ่งก็คือการที่ businness component นั้น ตามธรรมดาแล้วจะไม่ทำการ rollback persistent state กล่าวคือเมื่อมี abort เกิดขึ้น resource นั้นก็จะไม่ถูก update ดังนั้น ก็จะไม่มีการ undo ข้อมูลที่เป็น permanent data ในคอมโพเน้นต์เหล่านั้น ซึ่งทั้งนี้ก็ขึ้นอยู่กับระบบด้วย โดยคอมโพเน้นต์จะควบคุมการทำทรานส์แอ็กชันและสั่งให้ทรานส์แอ็กชันนั้น Abort แต่การทำการ rollback กับ persistent state นั้น จะเกิดขึ้นโดยอัตโนมัติ ดังนั้นเมื่อ business component ทำโอเปอเรชันภายใต้ทรานส์แอ็กชัน แต่ละคอมโพเน้นต์จะจัดการ persistent operation ทั้งหมดโดยถือว่าทรานส์แอ็กชันทั้งหมดจะเสร็จสิ้นอย่างเรียบร้อย

7.3.2 Nested Transactions

หากเราต้องการที่จะเขียนแอปพลิเคชันที่สามารถออกแบบเส้นทางท่องเที่ยวสำหรับใช้กับองค์กรหรือบริษัทที่เกี่ยวกับการท่องเที่ยว แอปพลิเคชันที่ต้องการนั้นต้องเขียนโค้ดที่ออกแบบเส้นทางรอบโลก

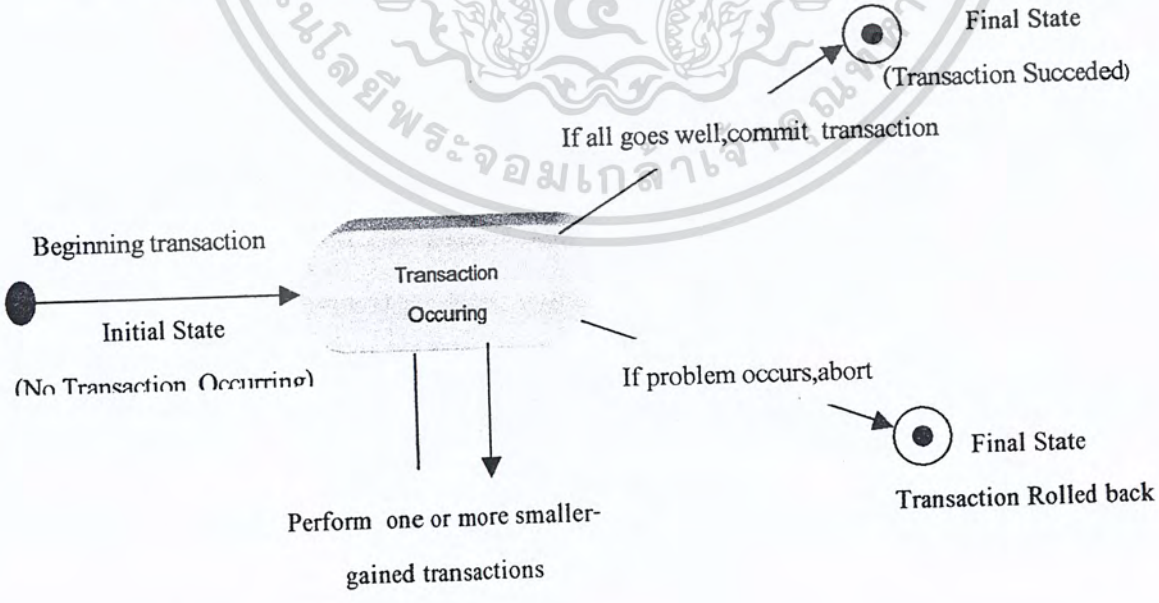
และเป็นแอปพลิเคชันที่สามารถบริการการซื้อตั๋วที่ใช้ในการเดินทางได้ ซึ่งจะทำให้แอปพลิเคชันจะต้องมีโอเปอเรชัน ต่าง ๆ ดังนี้

1. แอปพลิเคชันนี้ต้องสามารถให้จองตั๋ว จาก Boston, USA ไป New York, USA
2. แอปพลิเคชันต้องสามารถจองตั๋วเครื่องบินจาก New York, USA ไป London England
3. แอปพลิเคชันต้องสามารถจองตั๋วบอลลูนจาก London, England ไปยัง Parisประเทศฝรั่งเศส
4. แอปพลิเคชันต้องสามารถหาว่าเที่ยวบินไหนที่ไม่ได้ออกจากประเทศฝรั่งเศส

ในการจองตั๋วนั้นหากใช้ Flat Transactions แล้ว แอปพลิเคชัน นั้น จะต้องการโอเปอเรชันเพื่อที่จะทำการ roll back transaction เพราะว่าไม่มีเที่ยวบินใดที่ออกจากปารีสดังนั้นแอปพลิเคชันก็จะไม่มีข้อมูลการจองเที่ยวบินที่ไปปารีสแต่อาจจะมีการแทนการเดินทางเป็นอย่างอื่น เช่น รถไฟหรือเครื่องบิน ดังนั้น Flat transaction จึงไม่เพียงพอ ดังนั้น Nested Transaction สามารถที่จะแก้ปัญหาเหล่านี้ได้ โดย nested Transaction ยอมให้ทำการฝัง atomic units of work ลงใน units of work อื่น ๆ กล่าวคือ Unit of work จะถูกซ่อนอยู่ใน unit of work อื่น ๆ ที่สามารถทำการ roll back โดยไม่มีการบังคับทรานส์แอ็กชันทั้งหมดให้ roll back

ดังนั้น unit ที่มีขนาดใหญ่กว่าจะพยายามทำส่วนของ Embedded unit of work ซ้ำ ๆ ถ้า embedded unit of work นั้นเสร็จสมบูรณ์แล้ว unit ที่มีขนาดใหญ่กว่าก็จะเสร็จด้วย แต่ถ้า unit of work ไม่สามารถทำงานได้ มันก็จะบังคับให้ unit นั้น ๆ fail

เราสามารถเปรียบเทียบ Nested Transaction ว่าเหมือนกับเป็น Tree ของทรานส์แอ็กชันซึ่งจะมีการแตกย่อยลงมาจาก root หรือ top-level transaction โดย root transaction จะเป็นทรานส์แอ็กชันหลัก จากตัวอย่างของ trip-planning example นั้น Root transaction เป็นโพรเซสที่รวมเกี่ยวกับการจองตั๋วทั้งหมดทุก ๆ ทรานส์แอ็กชันใน tree transaction จะเรียกว่า subtransaction ตามรูป 7-3



รูปที่ 7-3 The Nested Teansactions

จุดที่พิเศษของ nested transaction ก็คือ การที่ subtransaction สามารถจะทำการ roll back ได้อย่างอิสระ โดยไม่มีผลกระทบต่อทรานส์แอ็กชันอื่น ๆ ที่อยู่ในระดับที่สูงกว่า ซึ่งนอกจากจะเป็นแนวความคิดที่ดีแล้วยังสามารถช่วยแก้ปัญหาที่กล่าวมาในข้างต้นได้กล่าวคือ ถ้าการจองตั๋วนั้นเป็น Nested transactions ก็จะสามารถทำการ roll back การจองใด ๆ ก็ได้โดยไม่เกี่ยวข้องกับเงื่อนไขอื่น ๆ

บทที่ 8
การรักษาความปลอดภัยของระบบ

8.1 การรักษาความปลอดภัย

ลักษณะของการจัดการด้านการรักษาความปลอดภัยนั้นเพื่อสร้างการเชื่อมต่อการรักษาความปลอดภัยจาก Web browser, Java client และ EJB server โดยจะมีลักษณะดังนี้

1. ขอบเขตในการรักษาความปลอดภัยนั้นจะมีการกำหนดเป็นกลุ่มต่างๆ ของ User โดยมีจุดประสงค์เพื่อป้องกันทรัพยากรของ server เราอาจไม่ได้ทำตามของเขตเหล่านี้ หรืออาจจะมีทางเลือกอื่น ๆ ก็เป็นไปได้ เช่น การใช้ Window NT , UNIX, LDAP security store

2. การอนุญาตการร้องขอเพื่อเข้าใช้ทรัพยากรของ Server ซึ่งการให้อนุญาตนี้สามารถทำได้โดยการใช้ Username กับ เพื่อให้อนุญาตกับ Client

3. การใช้ Java Authentication and Authorization Service (JAAS) application program interface (API) สำหรับ Authentication

4. Data Intrigrity และ confidentially ผ่าน SSL Protocal Client สามารถที่จะสร้าง SSL Session ด้วย Web Logic Server โดยการใช้ Hypertext Transfer Protocol (HTTP) หรือ Remote Mothod Tnvocation (RMI) บน Inter –ORB (IIOP) Protocol.

5. การตรวจสอบ event เช่น การพยายามเข้า login หลังจากที่ login ไม่สำเร็จ (Failed login attemt), การร้องขอให้มีการรับรอง (Authentication request)

6. การตรวจสอบการติดต่อเข้ามาของ Client เพื่อการตอบรับ หรือการปฏิเสธการร้องขอจาก Client จาก จุดเริ่มต้นซึ่งอาจจะเป็น hostname หรือ network address หรือ protocol ของ Client

จากสถาปัตยกรรมของการรักษาความปลอดภัยนั้นส่วนที่อยู่ในชั้นแรกก็คือ Authentication ซึ่ง Authentication จะป้องกัน บุคคลที่จะเข้ามาติดต่อ Server Environment

ในการ Authentication จะเป็นการตรวจสอบว่าไคลเอ็นต์คือคนที่เขาอ้างว่าเขาเป็นหรือเปล่าและ authorization ซึ่งคือการตรวจสอบผู้ที่ถูก authentication แล้วว่า มีสิทธิ์ที่จะทำงานนั้น ๆ หรือไม่ซึ่งการ authentication จะต้องถูกกระทำในเวลาใดเวลาหนึ่งก่อนที่จะเรียกใช้ method ของ EJB ส่วนการ Authorization จะถูกกระทำระหว่าง การเรียกใช้ method ของ EJB โดยมีรายละเอียดดังนี้

ขั้นตอนที่ 1: การ Authentication

EJB ไม่ได้บังคับวิธีการ authentication ไว้ ดังนั้นการที่จะทำการ authentication จึงเป็นหน้าที่ของ application ของผู้ใช้กับ container ซึ่งหมายความว่าแต่ละ container จะจัดการกับเรื่องนี้แตกต่างกัน เช่น BEA's Weblogic นั้นเราสามารถระบุ username/password ใน code ของ ไคลเอ็นต์เมื่อมันใช้ JNDI เพื่อมองหา home object

```
Properties props = System.getProperties();

props.put(Context.SECUTIRY_PRINCIPAL, "EmployeeA");
props.put(context.SECURITY_CREDENTIALS, "myPassword1");

Context ctx = new InitialContext(props);
```

เนื่องจาก EJB specification ไม่ได้ระบุไว้ถึงวิธีการ authentication ทำให้ code เหล่านี้ไม่สามารถนำไปใช้บน application server อื่นๆ ได้

เมื่อผู้ใช้ run code นี้ application server จะ map username/password ของผู้ใช้เข้ากับ security identity ขั้นตอนนี้ก็ขึ้นอยู่กับแต่ละ application server เช่นกัน บาง application server อนุญาตให้ผู้ใช้ใส่ username/password ไว้ใน property file ที่ application server จะอ่านตอน runtime บาง server จะ support ระบบ security ที่ขณะนี้ใช้กันอยู่แพร่หลายทั่วไป เช่น username/password ที่อยู่ใน LDAP server

ขั้นตอนที่ 2 : การ Authorization

หลังจากไคลเอ็นต์ถูก authenticate แล้ว มันจะต้องผ่านการ authorization เพื่อตรวจสอบสิทธิ์ในการเรียกใช้ method ของ EJB โดยจะมีแนวทางอยู่ 2 แนวทางในการ authorization คือ declaratively และ programmatically

ถ้าผู้ใช้เลือกการทำงานแบบ declarative authorization, container จะจัดการตรวจสอบ authorization ทั้งหมดให้เป็นวิธีที่สะดวก และง่ายดาย ช่วยให้ผู้ใช้สามารถเขียน bean ได้โดยไม่ต้องไปพะวงกับเรื่อง security

แต่ถ้าผู้ใช้เลือกการทำงานแบบ programmatic authorization, ผู้ใช้จะต้องเขียนการตรวจสอบ security ไว้ใน bean ของผู้ใช้เอง ทำให้ bean นั้นมี business logic ปนอยู่กับ security ทำให้ bean มีขนาดใหญ่โตโดยไม่จำเป็น

8.2 Security Roles

Authorizationn ใน EJB ใช้ concept ของ security roles ซึ่งคือกลุ่มของ client identity ในการที่ไคลเอ็นต์ทำ operation ต่าง ๆ ไคลเอ็นต์จะถูกตรวจสอบว่า security identity ของมันอยู่ใน security roles ที่ถูกต้องรึเปล่า

ข้อดีของการใช้ security roles คือผู้ใช้ไม่ต้องระบุ identity ลงไปใน bean ของผู้ใช้ นี่เป็นสิ่งที่จำเป็นในการพัฒนา bean เพื่อที่จะไป deploy ในหลาย ๆ ที่ซึ่งมีระบบ security ที่ต่างกัน ทำให้เราสามารถเปลี่ยนแปลงสิทธิในการใช้ method ได้โดยไม่ต้อง compile bean ใหม่

8.3 Programmatic Authorization

ในการตรวจสอบ authorization ด้วย bean ของผู้ใช้เองนั้นควรจะต้อง query EJB context เพื่อเรียกข้อมูลเกี่ยวกับไคลเอ็นต์ที่ติดต่ออยู่ในขณะนั้นซึ่งจะมี method ที่เกี่ยวข้องอยู่ 2 method คือ isCallerInRole() และ getCallerIdentity()

- **isCallerInRole()**

isCallerInRole(Identity role) จะทำการตรวจสอบว่าผู้ที่เรียกใช้ bean อยู่ใน role ที่กำหนดหรือไม่ เมื่อผู้ใช้เรียก method นี้ ผู้ใช้จะต้องส่ง role ที่ต้องการเปรียบเทียบกัน, ผู้ที่เรียกใช้ bean จะเป็นถึงความแตกต่างระหว่าง isCallerInRole(Identity role) และ declarative security คือ isCallerInRole(Identity role) จะไม่ตรวจสอบ role ที่คุณระบุไว้ใน deployment descriptor แต่จะตรวจสอบกับ role ที่ระบุไว้ใน code ของผู้ใช้เท่านั้น

```
import java.security.Identity;
...
public class MyBean implements SessionBean {
    private SessionContext ctx;
    ....
    public void foo() {
        Identity id = new MyIdentity("administrators");
        If (ctx.isCallerInRole(id)) {
            System.out.println("An admin called me");
            return;
        }
    System.out.println("A non-admin called me");
    }
}
```

โค้ดข้างบนเป็นตัวอย่างในการทำงานต่างกันถ้า role ของผู้เรียกใช้ต่างกัน เราจะต้องสร้าง class MyIdenity ซึ่ง inherit มาจาก java.security.Identity ซึ่งเป็น abstract class ซึ่งตัวอย่างของ class สามารถแสดงได้ดังต่อไปนี้

```
import java.security.Identity;
public class MyIdentity extends Identity {
```

```
        public MyIdentity (String id) {
            super(id);
        }
    }
```

- **getCallerIdentity()**

getCallerIdentity() จะส่งค่า security identiy ของผู้ที่เรียกใช้ bean ในขณะนั้นกลับมา โดยสามารถที่จะ นำ identity นั้นไปใช้ ทำงานต่าง ๆ ต่อได้ดังเช่นตัวอย่างโค้ดข้างล่าง

```
import java.security.Identity;
...
public class MyBean implements SessionBean {
    private SessionContext ctx;
...
    public void bar() {
        Identity id = ctx.getCallerIdentity();
        String name = id.getName();
        System.out.println("The caller's name is " + name);
    }
}
```

8.4 SSL Protocol

การนำ SSL Protocol มาใช้ การสร้าง security connection นั้น เมื่อ SSL protocol บน Server ถูกกำหนดโดยการใช้การ authentication ร่วมกันแล้ว การติดต่อของ client ก็จะต้องอาศัย digital certificate ที่ถูกระบุไว้อย่างถูกต้อง

SSL (Secure Sockets Layer) Handshake Protocol ได้รับการพัฒนามโดย Netscape Communications Corporation เพื่อจัดการด้านการรักษาความปลอดภัยและการใช้งานอย่างอิสระบนระบบอินเตอร์เนต SSL Protocol นี้สนับสนุนการทำ Authentication ของสถาปัตยกรรมแบบ Server - Client SSL protocol นั้นเป็น application independent ที่ให้อนุญาตกับโปรโตคอลต่าง ๆ เช่น HTTP (HyperText Transfer Protocol) FTP (File Transfer Protocol) รวมไปถึง Telnet ด้วย

SSL protocol สามารถติดต่อด้วย encryption เหมือนกับการขอ authenticate กับ server ก่อนที่จะทำการแลกเปลี่ยนข้อมูลในระดับ higher-level application SSL protocol จะดูแลเรื่องของการรักษา

ความปลอดภัย และ Integrity ของ Transmission channel ด้วยวิธีการ Encryption Authentication และ Message authentication codes.

SSL Handshake Protocol จะประกอบไปด้วย 2 เฟส คือ

- Server authentication
- Optional client authentication

สำหรับใน Server authentication ซึ่งเป็นเฟสแรกนั้น server จะตอบสนองการร้องขอจาก Client ด้วยการส่ง certificate และ รหัส(ciphers) ไปให้ จากนั้นด้านไคลเอนต์จะสร้าง master key ซึ่งจะเกิดจากการเข้ารหัสกับ public key ของ server และทำการส่ง encrypted master key ที่ผ่านการเข้ารหัสแล้วไปยัง server เมื่อ server ได้รับ master key แล้วจะทำการแยกรหัสออกมา จากนั้นก็จะทำการ Authenticates ไปยัง client ด้วยการส่ง messageที่ระบุการ Authenticate ไปกับ master key

ในเฟสที่สอง ซึ่งก็คือ Optional second phase นั้น server จะส่งการสอบถามไปยังไคลเอนต์ เพื่อให้ไคลเอนต์ทำการ Authenticates ตัวเองมายัง server ก่อน ด้วยการส่ง client's digital signature กลับมาให้ หรือที่เราเรียกกันว่า public-key certificate นั่นเอง

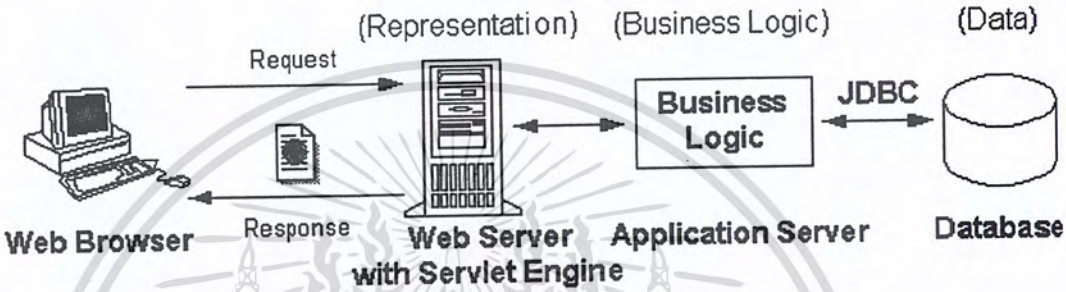
วิธีการในการเข้ารหัสมีมากมายหลายวิธีที่ SSL สามารถใช้ได้ ในระหว่างการทำ ``handshaking" นั้น จะใช้ RSA public-key cryptosystem หลังจากนั้น เมื่อมีการแลกเปลี่ยน key กันแล้ว เลขรหัส (ciphers) ก็จะถูกนำมาใช้งาน ซึ่งจะประกอบด้วย RC2 ,RC4, IDEA, DES และ triple-DES สำหรับ Public-key certificates จะมีรูปแบบตาม X.509 syntax

8.5 Application Server

สำหรับเวปไซต์เล็ก ๆ แล้วโครงสร้างของ Server Side Application ก็เพียงพอต่อความต้องการสำหรับประโยชน์ใช้สอย แต่สำหรับเวปไซต์ใหญ่ ๆ ที่มีผู้ใช้จำนวมหาศาลส่วนประกอบที่สำคัญอีกอย่างหนึ่งที่มักจะถูกเพิ่มเข้าไปมักจะเป็นส่วนที่เรียกว่า Application Server

Application Server เป็นส่วนประกอบอีกส่วนหนึ่งที่ถูกสร้างขึ้นมาเพื่อสนองความต้องการที่ว่าในการสร้างระบบที่ดีระบบหนึ่งขึ้นมานั้น เราควรจะแยกส่วนที่ทำหน้าที่เกี่ยวกับการบริการต่าง ๆ รวมไปถึงการจัดสรรข้อมูลออกมาเป็นอีกส่วนต่างหาก (Business Logic) ทั้งนี้เพื่อเพิ่มความคล่องตัวในการเปลี่ยนแปลงของระบบ รวมไปถึงการจัดระบบให้แบ่งออกเป็นส่วน ๆ อย่างชัดเจนมากยิ่งขึ้น โดยถ้าดูจากบทบาทของ Application Server ที่ผ่านตั้งแต่อดีตจนถึงปัจจุบันเราอาจจะพูดได้ว่า Application Server ก็คือส่วนที่ช่วยอำนวยความสะดวกให้กับ Server Side Application โดยมักจะเป็นตัวช่วยในการคำนวณข้อมูล (Services), ช่วยเป็นตัวกลางในการจัดส่งข้อมูล(Messaging Services), ช่วยในการควบคุมการจัดเก็บข้อมูลลงไปในเดต้าเบสอย่างถูกต้อง (Transaction) ตลอดจนช่วยในการเพิ่มความเร็วให้กับระบบโดยการ

เก็บอ๊อปเจคต่าง ๆ ที่ใช้หรือไม่ใช้แล้วไว้เพื่อนำมาใช้อีก (Caching) เป็นต้น (ในแง่ของการออกแบบระบบ Application Server มักจะเป็นส่วนที่เราใช้ในส่วนที่เรียกว่า Services Layer ซึ่งในจาวาแล้วส่วนนี้ก็มักจะเกี่ยวข้องกับ Enterprise Java ต่าง ๆ เช่น EJB, JTS, JMS, JNDI นั่นเอง) โดยประโยชน์อย่างหนึ่งของ Application Server ที่เรามักพบเห็นโดยทั่วไปก็คือการนำมาใช้สำหรับรัน EJB



รูปที่ 8-1 แสดงหลักการของ Application Server

8.6 Enterprise Java Bean

ถ้าพูดกันในเชิงนิยามแล้ว EJB ไม่ได้ถูกรันที่ Application Server แต่จะถูกรันกับส่งหนึ่งที่เรียกว่า EJB Server อย่างไรก็ตาม Application Server ส่วนมากมักจะสนับสนุน EJB ซึ่งเป็นส่วนหนึ่งของ J2EE ดังนั้น Application Server ส่วนมากจึงมีส่วนหนึ่งที่ทำหน้าที่เป็น EJB Server เพื่อรัน EJB ต่าง ๆ ได้โดย EJB สามารถแบ่งออกอย่างคร่าว ๆ เป็นสามแบบคือ

8.6.1 Session Bean

เป็น EJB ที่มักจะถูกใช้เป็นตัวเก็บรายละเอียดของผู้ใช้ที่กำลังทำกิจกรรมบางอย่างกับระบบหรืออาจจะถูกใช้เป็น Object Pool สำหรับ object ต่าง ๆ ที่ JSP หรือ Servlet สามารถเรียกใช้เพื่ออำนวยความสะดวกในการประมวลผลตลอดจนใช้ในการเปลี่ยนแปลงสถานะต่าง ๆ ของระบบโดยเรามักเรียกส่วนนี้ว่า Services Layer นั่นเอง

8.6.2 Entity Bean

เป็น EJB ที่ใช้สำหรับเก็บเคต้าต่าง ๆ ที่เราทำการแมปมาจากเคต้าที่อยู่ในเคต้าเบส, ไฟล์, LDAP objects หรืออื่น ๆ ซึ่งจุดหลักของการนำเคต้าต่าง ๆ มาใส่ไว้ใน Entity Bean ก็คือการเพิ่มความง่ายในการเปลี่ยนแปลงเคต้าโดยผ่านทาง EJB Object, การง่ายต่อการควบคุมการอ่านและเขียนเคต้าต่าง ๆ ได้อย่างถูกต้องในกรณีที่ระบบมีมากกว่า 1 thread เข้ามาเกี่ยวข้องในการอ่านเขียนหรือเปลี่ยนแปลงเคต้าตัวเดียวกันในเวลาเดียวกัน รวมไปถึงความเร็วที่เพิ่มขึ้นในการเก็บเคต้าต่าง ๆ ไว้ในหน่วยความจำ (Bean Caching) แทนที่จะต้องทำการอ่านเคต้าใหม่จากเคต้าเบสทุกครั้งที่เราต้องการทำกิจกรรมบางอย่างกับเคต้า

เหล่านั้น โดยข้อดีต่าง ๆ ที่กล่าวมาข้างต้นเหล่านี้จะเกิดขึ้นได้จากการทำงานร่วมกันระหว่าง Entity Bean, EJB Container และ EJB Server อย่างไรก็ตามนักพัฒนาระบบก็ยังคงถกเถียงกันอยู่ถึง performance ที่เกิดขึ้นจากการใช้ Entity Bean เพื่อเก็บเดต้าต่าง ๆ ไว้ โดยผลกระทบนี้มาจากการที่แต่ละ thread จะต้องแย่งกันเข้าไปใช้ Entity Bean ตัวเดียวกันเพื่อทำการเขียนอ่านหรือแปลงแปลงเดต้า(Bean Contention) ซึ่งผลที่เกิดขึ้นอาจจะทำให้ระบบช้าลงกว่าการใช้ JDBC หรือ Object Database ธรรมดาทั่ว ๆ ไปก็ได้

บทที่ 9
หลักการออกแบบเพื่อการสร้างโปรแกรมในโครงงาน

9.1 UML (Unified Modeling Language)

UML หรือ Unified Modeling Language เป็นภาษาในการโมเดลมาตรฐาน UML เกิดจากการพัฒนาร่วมกันของผู้นำเทคโนโลยีทางด้านออบเจ็กต์ 3 คน คือ Grady Booch Ivar Jacobson และ Jim Rumbaugh โดยก่อนที่จะเป็น UML นั้น ผู้นำทั้ง 3 คนนี้ต่างก็มีโมเดลทางด้านออบเจ็กต์เป็นของตัวเอง ต่อมาทางบริษัท Ration Software จึงได้ร่วมมือกับผู้นำทั้ง 3 คนนี้ ในการพัฒนาโมเดลร่วมกัน เพื่อให้เกิดความเป็นหนึ่งเดียวสำหรับการพัฒนาระบบด้วยออบเจ็กต์โอเรียนเท็ด (Object Oriented)

UML นั้นได้รวมแนวความคิดของวิธีการต่าง ๆ เข้าไว้ด้วยกัน จุดประสงค์ของ UML ก็คือ ต้องการสร้างโมเดลในการพัฒนาที่เข้าใจและสามารถสร้างได้ง่าย แต่สามารถนำไปใช้ได้กับทุกระบบ โดยตัว UML เองนั้นได้รับการสนับสนุนจากผู้นำในอุตสาหกรรมทางด้านคอมพิวเตอร์หลาย ๆ ราย เช่น HP, IBM, Microsoft หรือ Oracle เป็นต้น

ใน UML นั้นมีโมเดลที่มองเห็นได้สำหรับระบบหลาย ๆ โมเดล โดยแต่ละโมเดลก็จะแสดงมุมมองต่อระบบที่ไม่เหมือนกัน ซึ่งประกอบไปด้วย

- ยูสเคสไดอะแกรม (Use-Case Diagram) : ใช้แสดงการติดต่อระหว่างระบบกับผู้ใช้
- คลาสไดอะแกรม (Class Diagram) : ใช้แสดงโครงสร้างทางลอจิกของระบบ
- ออบเจ็กต์ไดอะแกรม (Object Diagram) : ใช้แสดงออบเจ็กต์และความสัมพันธ์
- สเตทไดอะแกรม (State Diagram) : ใช้แสดงใช้แสดงพฤติการณ์ของระบบ
- คอมโพเนนท์ไดอะแกรม (Component Diagram) : ใช้แสดง ใช้แสดงโครงสร้างทางกายภาพของซอร์ฟแวร์
- ดีพลอยเม้นท์ไดอะแกรม (Deployment Diagram) : ใช้แสดงการติดต่อระหว่างซอร์ฟแวร์กับฮาร์ดแวร์
- ซีเควนซ์ไดอะแกรม (Sequence Diagram) : ใช้แสดงลำดับของเมสเสจระหว่างออบเจ็กต์
- คอลลาบอเลชันไดอะแกรม (Collaboration Diagram) : ใช้แสดงข้อมูลพื้นฐานเช่นเดียวกับซีเควนซ์ไดอะแกรม

9.1.1 ยูสเคสไดอะแกรม (Use-Case Diagram)

แผนภาพนี้จะแสดงมุมมองต่อฟังก์ชันการทำงานหลัก ๆ ของระบบซึ่งผู้พัฒนาระบบสามารถใช้แผนภาพ Use Case นี้ในการสื่อสารกับผู้ใช้ระบบได้ด้วย ตัวแผนภาพ Use Case อาจจะเป็นแผนภาพหลักในการพัฒนาระบบเนื่องจากตัวแผนภาพ Use Case นั้นได้รวมความต้องการ และรายละเอียดในการทำงานหรือ Scenario ของระบบไว้ภายในแล้ว สำหรับตัว Use Case เองนั้นก็สามารถที่จะแบ่งเป็น Use

Case ย่อย ๆ ได้อีก หรืออาจจะใช้ในการสร้าง Scenario ซึ่งจะแสดงรายละเอียดลำดับในการติดต่อสื่อสารระหว่างวัตถุในระบบ เนื่องจากแผนภาพ Use Case นั้นเป็นมุมมองในระดับสูง และสามารถอธิบายระบบด้วยวิธีง่าย ๆ ทำให้ผู้ใช้ในทุก ๆ ระดับสามารถตรวจสอบแผนภาพ Use Case ได้ง่ายขึ้น โดยสามารถใช้แผนภาพนี้ในการสื่อสารระหว่างทีมผู้พัฒนาเองเนื่องจากว่าสัญลักษณ์ต่าง ๆ ที่ใช้ในแผนภาพนี้เป็นมาตรฐานที่ทุกคนรู้จัก และแผนภาพก็มีรายละเอียดของระบบที่ครบถ้วน

9.1.2 คลาสไดอะแกรม (Class Diagram)

คลาสไดอะแกรมเป็นไดอะแกรมที่มีความสำคัญมากที่สุด ในการวิเคราะห์และออกแบบโดยใช้วิธีออบเจ็กต์โอเรียนเท็ด เนื่องจากคลาสไดอะแกรมแสดงโครงสร้างของออบเจ็กต์และคลาสที่มีในระบบ และยังใช้ในการแยกย่อยรายละเอียด (Decomposite) ด้วยวิธีออบเจ็กต์โอเรียนเท็ด อย่างไรก็ดีคลาสไดอะแกรมนั้น อาจจะมีหน้าตาที่คล้ายกับออบเจ็กต์ไดอะแกรม แต่ว่าคลาสไดอะแกรมจะแสดงเฉพาะคลาสโดยไม่ได้แสดงอินสแตนซ์ หรือออบเจ็กต์ทั้งหมด

9.1.3 ความสัมพันธ์ระหว่างคลาส

สามารถแบ่งหลัก ๆ ได้ คือ การแอสโซซิเอชัน (Association) การถ่ายทอด (Inheritance) และการอะกรีเกชัน (Aggregration) ซึ่งมีรายละเอียดดังนี้

- **แอสโซซิเอชัน (Association)**

เมื่อออบเจ็กต์หนึ่งใช้บริการของอีกออบเจ็กต์ แต่ว่าไม่ได้เป็นเจ้าของออบเจ็กต์นั้นเราเรียกความสัมพันธ์นี้ว่า แอสโซซิเอชัน ซึ่งความสัมพันธ์แบบนี้จะใช้ได้อย่างเหมาะสมก็ต่อเมื่อ

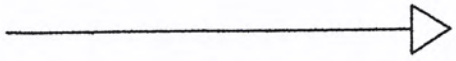
- ออบเจ็กต์หนึ่งใช้บริการของอีกออบเจ็กต์ แต่ไม่ได้มีความสัมพันธ์แบบอะกรีเกชัน
- วงจรชีวิตของคลาสที่ใช้ ไม่ได้เป็นหน่ที่ความรับผิดชอบของคลาสที่เรียกใช้ ทั้งการสร้างและการทำลายออบเจ็กต์
- ความสัมพันธ์ระหว่างออบเจ็กต์นั้นน้อยกว่าอะกรีเกชัน
- ความสัมพันธ์เป็นคุณลักษณะของไคลเอ็นต์-เซิร์ฟเวอร์
- ออบเจ็กต์ที่ถูกเรียกมีการแชร์และถูกใช้มากเท่า ๆ กับที่ออบเจ็กต์อื่น ๆ เรียกใช้

รูปที่ 9-1 สัญลักษณ์ของแอสโซซิเอชัน

- **การถ่ายทอด (Inheritance)**

เมื่อคลาสหนึ่งเป็นความเฉพาะของอีกคลาสหนึ่งนั้น เราจะใช้ความสัมพันธ์ที่เรียกว่า เจเนอรอลไลเซชัน (Generation) หรือ การถ่ายทอด (Inheritance) หมายความว่า คลาสลูกจะมีคุณสมบัติทุกอย่างที่

คลาสพ่อ แม่มี ถึงแม้ว่าคลาสลูกจะมีความเฉพาะมากกว่าก็ตาม คลาสลูกอาจจะเพิ่มคุณสมบัติของคลาสพ่อแม่ โดยการเพิ่มแอตทริบิวต์เมธอดเข้าไป เราสามารถเรียกความสัมพันธ์นี้ว่า "เป็นชนิดหนึ่งของ" เช่น สัตว์เลี้ยงลูกด้วยนมเป็นชนิดหนึ่งของสัตว์ หรือ อินฟาเรดเซนเซอร์เป็นชนิดหนึ่งของเซนเซอร์ เป็นต้น



รูปที่ 9-2 สัญลักษณ์ของการถ่ายทอด

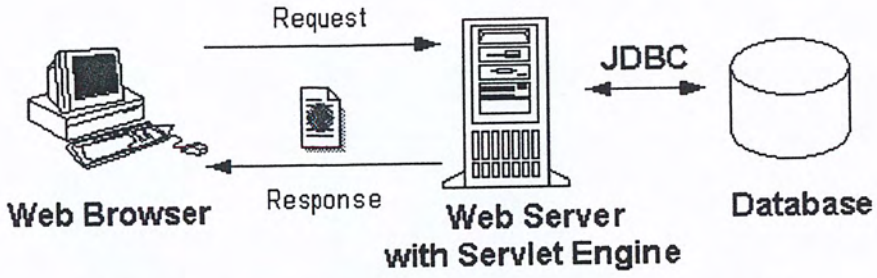
- อะกรีเกชัน (Aggregration)

ความสัมพันธ์แบบอะกรีเกชันนี้ จะใช้ก็ต่อเมื่อมีออบเจ็กต์หนึ่งบรรจุอีดออบเจ็กต์หนึ่งไว้ ทั้งในทางความคิดและกายภาพ คลาสที่ใหญ่กว่า เราเรียกว่าเจ้าของหรือ Owner หรือ whole ซึ่งเป็นคลาสที่มีหัวลูกศรรูปสี่เหลี่ยมอยู่ ส่วนคลาสที่เล็กกว่านั้น เราเรียกว่า คลาสคอมโพเน็นต์ (Component) หรือ owned หรือ part โดยทั่วไป คลาสเจ้าของมีหน้าที่สร้างและทำลายคลาสคอมโพเน็นต์ ใน UML ยอมให้คลาสคอมโพเน็นต์สามารถมีคลาสเจ้าของได้หลายคลาส และเมื่อออบเจ็กต์มีการแชร์ความเป็นเจ้าของ เราก็จะต้องใช้กฎของ ad hoc ในการตัดสินใจว่าคลาสใดจะทำหน้าที่สร้างและทำลายคลาสคอมโพเน็นต์นั้น

รูปที่ 9-3 สัญลักษณ์อะกรีเกชัน

9.2 การออกแบบฐานข้อมูล (Database Design)

เดต้าเบสเป็นส่วนที่ถูกเพิ่มขึ้นมาใน Server Side Application โดยมักจะเป็นส่วนที่อยู่ทางท้ายสุดของระบบ ส่วนที่จะเป็นตัวเรียกใช้เดต้าเบสมักจะเป็นส่วนที่ทำหน้าที่ประมวลผล request ที่มาจากผู้ใช้ซึ่งโดยทั่วไปก็คือ Servlet Engine นั่นเอง [ในกรณีของ JSP (Java Server Pages) ตัว JSP Container จะเป็นส่วนที่ทำการติดต่อกับเดต้าเบสโดย JSP เพจจะทำการเรียกใช้ข้อมูลต่าง ๆ โดยผ่านทางจาว่าคลาสที่อยู่ในรูปของ JavaBean หรือ Tag Library] ตัวอย่างของ Server Side Application ที่มีเดต้าเบสมาเกี่ยวข้องอาจจะเป็นดังรูปที่ 9-4



รูปที่ 9-4 Servlets connecting to Relational Database via JDBC

9.2.1 เอ็นทิติรีเลชันชิพโมเดล (Entity Relationship Model)

เอ็นทิติรีเลชันชิพโมเดล เป็นวิธีในการออกแบบฐานข้อมูลอีกรูปแแบหนึ่งซึ่งเกิดภายหลังรีเลชันโมเดล (relational model) และเป็น conceptual schema model หรือ semantic data model ซึ่งคิดโดยศาสตราจารย์ชาวจีนชื่อ P.Chen ปี ค.ศ.1975 สำหรับ 3 data model เดิมนั้น ขึ้นอยู่กับ record structure จะเห็นว่าให้ความหมายของข้อมูลไม่ชัดเจนแต่ Entity Relationship model ถือเป็น Object Oriented อันแรกของโลกซึ่งเป็น model ใหม่ มีโปรแกรมวิ่งบนตารางซึ่ง ER จะเป็นตัวเก็บ semantic และใช้เป็นเครื่องมือในการออกแบบแต่ใช้เก็บข้อมูลไม่ได้ ในเรื่องของ ER model ส่วนใหญ่จะเกี่ยวข้องกับ Entity และ Attribute

- Entity (ปรากฏการณ์)

หมายถึง สิ่งที่เราสนใจจะเก็บ (Object of Instance) ซึ่งอาจจะสัมผัสแตะต้องได้ (tangible) เป็นรูปธรรม เช่น พนักงาน หรืออาจจะสัมผัสแตะต้องไม่ได้ (intangible) เป็นนามธรรม เช่น บริษัท หรือแผนก เป็นต้น

- แอตทริบิวต์ (Attribute)

หมายถึง คุณสมบัติ (Characteristic) ของ entity เช่น attribute ของ entity พนักงาน ได้แก่ ชื่อพนักงาน เพศ เงินเดือน ฯลฯ เป็นต้น หรือ แอตทริบิวท์ (Attribute) ของ เอนติตี้ (entity) รถยนต์ ได้แก่ ยี่ห้อ รุ่น สี น้ำหนัก เป็นต้น Value (Attribute Value) จะหมายถึงค่าคงที่ (Constant) ของแอตทริบิวต์ เช่น แอตทริบิวต์ชื่อพนักงาน มี value = นายยิ่งใหญ่ เป็นต้น

พนักงาน

	รหัสพนักงาน	ชื่อพนักงาน	เงินเดือน	
				Schema (entity type)
e1	8675	นายยิ่งใหญ่	31000	
e2	8972	นายเอก	27000	Instance (entity)
e3	8522	นายมงคล	42000	

ตาราง 9 –1 ตารางพนักงาน

จากตารางพนักงาน (Employee) ซึ่งเป็นวัตถุที่เราสนใจจะเก็บ ดังนั้นพนักงานแต่ละคนจะเป็นเอ็นติตี้ (entity) ที่เราสนใจ เราจึงให้ e1 เป็นพนักงานคนที่ 1, e2 เป็นพนักงานคนที่ 2 และ e3 เป็นพนักงานคนที่ 3 พนักงานจะมีแอตทริบิวต์ ที่สนใจคือ รหัสพนักงาน เงินเดือน เป็นต้น โดย value หรือ attribute value ของ e1, e2 และ e3 เป็นดังนี้

e1	(8675	นายยิ่งใหญ่	31000)
e2	(8972	นายเอก	27000)
e3	(8522	นายมงคล	42000)

9.3 สัญลักษณ์ที่นิยมใช้ในการเขียน ER Model สำหรับ Case Tool

1. กรอบสี่เหลี่ยม ใช้แทน entity ภายในบรรจุ attribute ของ entity โดยนิยมเขียนชื่อ entity ไว้ด้านบนสุดภายในกรอบ สำหรับ attribute ที่เป็น primary key และ attribute ที่เป็น Foreign Key จะเขียน PK และ FK ตามลำดับกำกับไว้ข้าง ๆ attribute นั้นดังรูป 6-10 กรณีที่ Primary Key ประกอบไปด้วย 2 attribute ขึ้นไป จะถือว่าเป็น Composit Primary Key (CPK)

กรณีที่เป็น weak entity (เป็น entity ที่ไม่มี key ไม่สามารถระบุ key attribute ได้จำเป็นต้องมี partial key) เช่น employee เป็น entity type ปกติ แต่ dependent เป็น weaken entity type สัญลักษณ์ของ weak entity type จะใช้เป็นรูปสี่เหลี่ยมหัวมน

2. เส้นประ ---------- ใช้แทนความสัมพันธ์ซึ่งเป็น optional (may be)

3. เส้นทึบ _____ ใช้แทนความสัมพันธ์ซึ่งเป็น mandatory (must be)

9.4 สถาปัตยกรรมที่ใช้ในการออกแบบโปรแกรมของระบบ Olala Tour

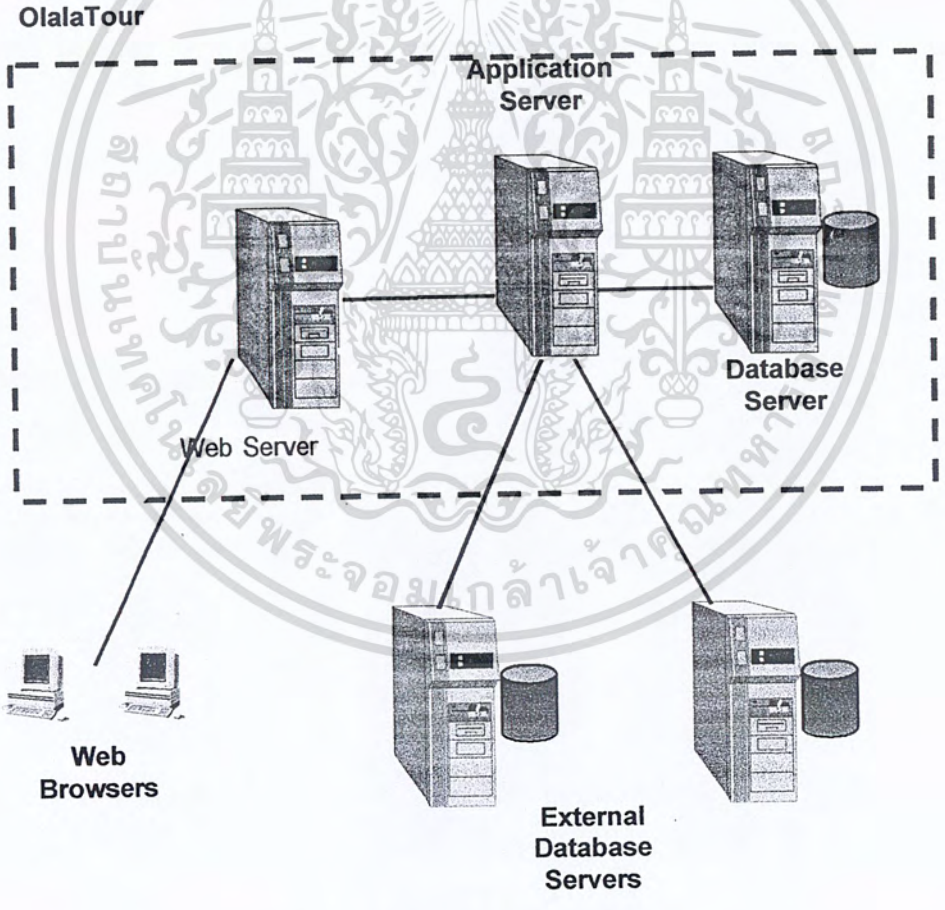
การค้าขายบนอินเทอร์เน็ตเป็นลักษณะของการทำธุรกิจเชิงพาณิชย์ที่ทำผ่านระบบเครือข่ายอินเทอร์เน็ต โดยที่ผู้ทำการค้าขายได้จัดตั้งร้านค้าโดยใช้โฮมเพจ(Homepage) เป็นสื่อในการแสดงระบบค้าขายของตน ที่สามารถใช้แสดงลักษณะสินค้าและบริการต่างๆไว้บนเซิร์ฟเวอร์ ซึ่งลูกค้าสามารถเข้าไปเลือกชมสินค้าหรือใช้บริการต่างๆ ผ่านเว็บบราวเซอร์ (Web Browser) ที่สามารถเข้าใช้บริการได้ตลอดเวลา เมื่อลูกค้าตกลงใจซื้อสินค้า หรือบริการแล้ว ก็จะตกลงทำสัญญาออนไลน์ เพื่อใช้เป็นหลักฐานในการบอกถึงการซื้อสินค้าหรือการใช้บริการต่างๆ แล้วจึงทำการชำระเงินผ่านสื่อต่างๆ เช่น เครดิตการ์ด

การสั่งจ่ายทางธนานัติ หรือ การจ่ายเงินผ่านบัญชีธนาคารเป็นต้น เมื่อผู้ให้บริการได้รับหลักฐานทุกอย่างครบถ้วนแล้ว ก็จะทำการจัดส่งสินค้าหรือ ให้บริการด้านนั้นๆ แก่ลูกค้าก็เป็นอันสิ้นสุดขบวนการทำงาน

ในโครงงานนี้ได้พัฒนาและออกแบบระบบงานเชิงคอมโพเนนต์ สร้างบริษัทชื่อ OlalaTour ให้บริการนำเที่ยว จองที่พักและตั๋วเดินทางโดยให้บริการผ่านอินเตอร์เน็ต ระบบประกอบด้วย 4 ส่วนหลักคือ

1. ส่วนกลาง มีหน้าที่จัดการควบคุมระบบ จัดการรายละเอียดของลูกค้า สร้างชุดการเดินทาง
2. ส่วนจองการเดินทาง ให้บริการจัดซื้อ และสำรองที่นั่งรถโดยสารและเครื่องบิน รวมถึงการเช่ารถบัส
3. ส่วนจองที่พัก ให้บริการจองห้องพักของโรงแรมต่าง ๆ
4. ส่วนจองชุดการเดินทาง (แพ็กเกจทัวร์) ให้บริการจองชุดการท่องเที่ยวของบริษัท OlalaTour

จากการทำงานของระบบต่าง ๆ ที่เกี่ยวข้องกับข้อมูลทั้งภายนอกและภายใน บริษัท Olala Tour จึงได้ออกแบบระบบเป็นสถาปัตยกรรมแบบมัลติเทียร์โดยมี 3 เทียร์ส (Three Tiers) ด้วยกัน



รูปที่ 9-5 โมเดลของระบบงาน

ตามรูปที่ 9-5 ที่แบ่งออกเป็น 3 เทียร์นั้นได้แก่

1. **พรีเซ็นเตชันเทียร์** คือเว็บบราวเซอร์ไคลเอนต์โดยการใช้เว็บบราวเซอร์ให้บริการกับผู้ใช้ทั่วไป และใช้แอปพลิเคชั่นในการสร้างแพ็กเกจทัวร์ ผู้ที่ใช้แอปพลิเคชันจะเป็นผู้ดูแลระบบเท่านั้น
2. **บิสซิเนสเทียร์** ใช้ J2EE เป็นเว็บเซิร์ฟเวอร์ และปีนแอปพลิเคชั่นเซิร์ฟเวอร์ เซอร์วิสทั้งสองเป็นเซอร์วิสที่สามารถทำงานได้บนระบบปฏิบัติการวินโดวส์ 98 หรือ วินโดวส์ 2000 ได้โดยคอมโพเนนต์ทั้งหมดของระบบจะถูกควบคุมและบรรจุอยู่ใน EJB Cointainer ส่วนเอกสารเอชทีเอ็มแอลและเอกสารเจเอสพีถูกจัดการโดย WEB Container ซึ่ง Container ทั้ง 2 นี้อยู่ใน J2EE Server
3. **ดาต้าเทียร์** ใช้ Cloudscape เป็นระบบจัดการฐานข้อมูลโดยเป็นเครื่องมือที่มาพร้อมกับ J2EE Server ซึ่งทำให้ง่ายในการนำไปใช้งาน

9.5 คุณสมบัติในด้านต่าง ๆ ของระบบ

1. การแก้ไขรายละเอียดของระบบกลางและระบบงานให้บริการด้านต่าง ๆ ได้ตามต้องการ

รายละเอียดทุกอย่างเกี่ยวกับระบบส่วนกลาง หรือ Olala Tour หรือ ระบบงานให้บริการส่วนที่เกี่ยวข้องนั้นสามารถทำการแปลี่ยนแปลงแก้ไขได้ตลอดเวลา

2. ระบบความปลอดภัย

มีระบบชื่อผู้ใช้งานและรหัสผ่าน ที่ให้สิทธิ์การเปลี่ยนแปลงรายละเอียดทุกอย่างของระบบส่วนกลางซึ่งสามารถทำได้เพียงผู้ดูแลระบบเท่านั้น

3. การสมัครเป็นสมาชิก

ในกรณีที่ลูกค้าต้องการซื้อหรือขอใช้บริการจากผู้ให้บริการนั้นจำเป็นต้องสมัครเป็นสมาชิกก่อน โดยการทำงานคือให้ลูกค้ากรอกรายละเอียดต่างๆเพื่อเข้าเป็นสมาชิก และลูกค้าจะมีรหัสผ่านในการใช้ในครั้งต่อไป

4. ความสามารถของผู้ดูแลระบบหลัก(Admin)

โดยหลักการระบบงานนี้น่าจะดำเนินการได้ตลอดเวลาแต่ก็ไม่สามารถหลีกเลี่ยง ผู้ดูแลระบบไม่ได้ เพราะต้องสามารถแก้ไขข้อทำงานที่ผิดพลาดในทันที ในระบบนี้ได้มีส่วนของผู้ดูแลหลัก ที่จะจัดการในเรื่องการดูแลฐานข้อมูลภายในระบบ

เมื่อได้ออกแบบสถาปัตยกรรมและกำหนดซอร์ฟแวร์ที่ใช้ใน แต่ละส่วนเรียบร้อยแล้ว จึงทำการออกแบบในส่วนของการสร้างโค้ดของโปรแกรมต่อไป

บทที่ 10

การสร้างโปรแกรมของระบบงานในส่วนที่รับผิดชอบ

ระบบ Olala Tour ในโปรเจคนี้จะมีลักษณะแตกต่างจากระบบการค้าอื่นๆทั่วไปที่เราพบบนอินเทอร์เน็ต คือโดยทั่วไปที่แต่ละระบบการค้าจะให้บริการเพียงส่วนของตนเอง หรือบางแห่งก็รวบรวมเอาบริการหลายๆ ชนิดจากหลายๆ ที่มาเก็บไว้ที่ฐานข้อมูลของตน เมื่อระบบเหล่านี้ต้องการจะขายสินค้าหรือบริการก็นำเอารายการการให้บริการของตนมาฝากไว้กับระบบงานนั้น แต่ระบบงานให้บริการในโปรเจคของเราจะมีคุณสมบัติคือ ตัวระบบงานส่วนกลางสามารถทำการติดต่อเชื่อมโยงกับระบบการให้บริการอื่นๆ ที่ทำการค้าร่วมกันอยู่ได้ เช่นการที่ลูกค้าสามารถเข้าไปค้นหาข้อมูลต่างๆ ของระบบการให้บริการหนึ่งๆ ได้โดยตรง เช่น ระบบโรงแรม ระบบงานบริการของรถบัส และสายการบิน โดยระบบส่วนกลาง หรือ Olala Tour นั้นจะเป็นผู้รับผิดชอบในเรื่อง ต่าง ๆ เช่น การเก็บประวัติและสถานะต่าง ๆ ของลูกค้า การดูแลในเรื่องความปลอดภัยในการเข้าใช้งานระบบของลูกค้าแต่ละคน รวมถึงความปลอดภัยในการเข้าถึงข้อมูลในส่วนต่าง ๆ อีกด้วย

ด้วยเนื่องจาก Olala Tour เป็นระบบที่ใหญ่มาก เพราะทำงานที่เกี่ยวข้องกับระบบภายนอกต่าง ๆ มาก ตามที่ได้กล่าวไว้ในข้างต้น ซึ่งในรายงานฉบับนี้จะนำเสนอเฉพาะส่วนที่ได้รับผิดชอบ คือ

- ส่วนระบบรถโดยสารและสายการบิน
- ส่วนระบบการทำงานของธนาคาร

10.1 ข้อมูลการให้บริการต่าง ๆ ของระบบ

- ส่วนการบริการของระบบรถโดยสารและสายการบิน
  - บริการค้นหาเส้นทางการเดินทาง
  - จองการเดินทาง
  - ยืนยันจองการเดินทาง
  - ยกเลิกจองการเดินทาง
- ส่วนการบริการของระบบธนาคาร
  - บริการการจ่ายเงินโดยการใช้บัตรเครดิต
  - บริการการจ่ายเงินโดยการใช้บัตรเดบิต
  - บริการการจ่ายเงินโดยการโอน

10.1.1 ส่วนการบริการของระบบรถโดยสารและสายการบิน

การค้นหาข้อมูลของสายการบิน

ขั้นตอนแรกผู้ใช้จะเริ่มใช้บริการจากระบบงาน จะทำการกรอกรายละเอียดที่ต้องการหาผลลัพธ์ที่ออกมาจะเป็นรายละเอียดต่าง ๆ ของเที่ยวการเดินทางที่เกี่ยวข้องกับคีย์เวิร์ดที่ผู้ใช้ป้อนเข้าไป ผู้ใช้สามารถเลือกในแต่ละเที่ยวการเดินทางเพื่อดูรายละเอียดเพิ่มเติมได้ โดยการค้นหาข้อมูลเที่ยวบินมีข้อมูลที่สามารถใช้เป็นเงื่อนไขได้ดังนี้

- สถานที่ต้นทาง

- สถานที่ปลายทาง
- ตัวเลือกว่าต้องการเดินทางไปอย่างเดียว หรือ ไป-กลับ
- วันที่ เวลา ของการออกเดินทางไป
- หากเลือกการเดินทางไปกลับสามารถระบุ วันที่ เวลา ของการเดินทาง
- ชนิดของที่นั่ง
- สายการบินที่ต้องการใช้บริการ

การเรียงลำดับจากตัวเลือกต่างๆที่ค้นหาเจอ เช่น เรียงจากราคาที่ต่ำสุด ใช้เวลาการเดินทางน้อยที่สุด โดยผลลัพธ์จากการค้นหาจะมีรายละเอียดของสายการบิน ดังนี้

ชื่อเที่ยวบิน

ชื่อสายการบิน

สถานีต้นทาง – ปลายทาง

เวลาขึ้นเครื่อง/เวลาลงเครื่อง

ราคา

10.1.2 การค้นหาข้อมูลรถโดยสาร

ขั้นตอนนี้จะทำงานเมื่อผู้ใช้เลือกรายการค้นหาสายการเดินรถ โดยผู้ใช้จะป้อนคุณสมบัติโดยคร่าว ๆ ของสายการเดินรถที่ตนต้องการจะใช้งาน โดยรายละเอียดข้อมูลในการค้นหารถโดยสารมีดังนี้

- สถานที่ต้นทาง
- สถานที่ปลายทาง
- ตัวเลือกว่าต้องการเดินทางไปอย่างเดียว หรือ ไป-กลับ
- วันที่ เวลา ของการออกเดินทางไป
- หากเลือกการเดินทางไปกลับสามารถระบุ วันที่ เวลา ของการเดินทาง
- ชนิดของที่นั่ง
- ชื่อบริษัทที่ต้องการใช้บริการ

การเรียงลำดับจากตัวเลือกต่างๆที่ค้นหาเจอ เช่น เรียงจากราคาที่ต่ำสุด ใช้เวลาการเดินทางน้อยที่สุด โดยผลลัพธ์จากการค้นหาจะมีรายละเอียดรถโดยสาร มีดังนี้

ชื่อบริษัทขนส่ง

ชื่อสายการเดินรถ

สถานีต้นทาง

สถานีปลายทาง

เวลารถออก

เวลาถึงที่หมาย

ราคา

10.1.3 ส่วนการบริการของระบบธนาคาร

ในส่วนนี้ผู้ใช้สามารถเข้ามาใช้บริการด้านการชำระเงินค่าบริการด้านต่าง ๆ ของระบบ Olala Tour ซึ่งสามารถเลือกวิธีการชำระเงินได้ คือ

- จ่ายเงินด้วยบัตรเครดิต วิธีนี้ผู้ใช้ต้องป้อนข้อมูลที่จำเป็นคือ
- หมายเลขบัตรเครดิตของผู้ใช้บริการเอง
- หมายเลขบัญชีที่ผู้ใช้ต้องการจะจ่ายเงินให้
- จำนวนเงินที่ต้องการจะจ่าย
- จ่ายเงินด้วยบัตรเดบิต ข้อมูลที่จำเป็นคือ
- หมายเลขบัตรเดบิตของผู้ใช้บริการเอง
- หมายเลขบัญชีที่ผู้ใช้ต้องการจะจ่ายเงินให้
- จำนวนเงินที่ต้องการจะจ่าย
- จ่ายเงินด้วยวิธีการโอน
- หมายเลขบัญชีของผู้ใช้บริการ
- พาสเวิร์ดของผู้เข้าใช้บริการเพื่อที่จะทำการโอนเงินนั้นเพื่อเป็นการรักษาความปลอดภัย หมายเลขบัญชีของผู้ให้บริการหรือ หมายเลขบัญชีที่ต้องการจะโอนเงินไปให้
- จำนวนเงินที่ต้องการจะโอน

โดยหลังจากที่ผู้ใช้เข้าไปทำการจ่ายเงินโดยผ่านระบบนี้แล้ว ระบบธนาคารนี้จะทำการสร้างรหัสเฉพาะของทรานแซคชันนั้นหรือ Transaction ID เพื่อให้ผู้ใช้นำรหัสนี้ไปใช้ในการรับรองการขอใช้บริการ หรือการ Confirm กับระบบ ซึ่งระบบ Olala นั้นก็จะสามารถนำรหัสนี้ไปตรวจสอบกับธนาคารว่าผู้ใช้นี้ได้ทำการจ่ายเงินแล้วจริง

9.2 การออกแบบระบบงาน

ในโครงงานนี้จะนำเสนอการออกแบบระบบเฉพาะส่วนของระบบรถโดยสาร,ระบบสายการบิน และระบบการทำงานของธนาคารซึ่งระบบงานบริษัท Olala Tour จะนำไปใช้โดยมีรายละเอียดดังนี้

10.2 สร้าง Use Case

เรานำรูปแบบระบบงานมาแยกออกเป็นฟังก์ชันการทำงานโดยรวมของระบบ ในรูปแบบของยูสเคส ซึ่งมียูสเคสดังนี้

10.2.1 ยูสเคสของระบบรถโดยสาร

ค้นหาข้อมูล

ค้นหาข้อมูลข้อมูลการจอง

ค้นหาเส้นทางเดินรถ

ค้นหาสถานีโดยสาร

ค้นหาบริษัทเดินรถ

ค้นหารถโดยสาร

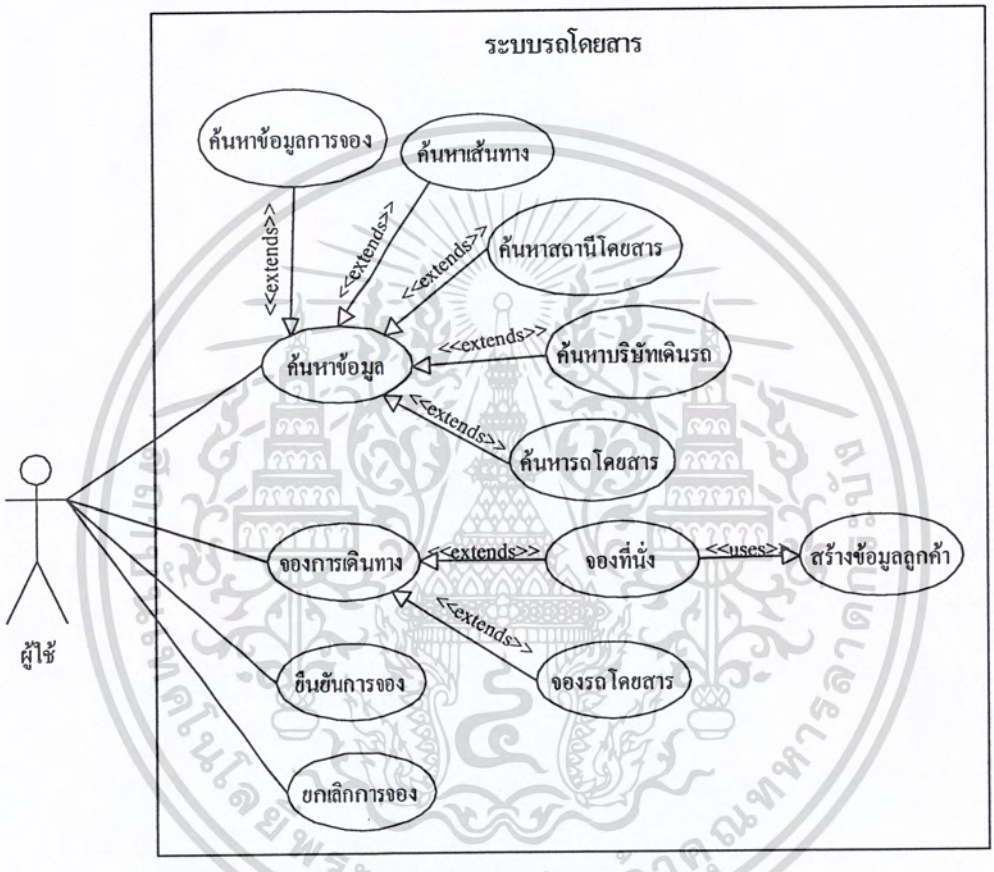
จองการเดินทาง

จองที่นั่ง

จองรถโดยสาร (เพื่อจัดทำแพกเกจทัวร์)

ยืนยันการจอง

ยกเลิกการจอง



รูปที่ 10-1 ยูสเคสของระบบรถโดยสาร

10.2.2 ยูสเคสของระบบสายการบิน

ค้นหาข้อมูล

ค้นหาข้อมูลข้อมูลการจอง

ค้นหาเส้นทางบิน

ค้นหาสถานีสนามบิน

ค้นหาบริษัทสายการบิน

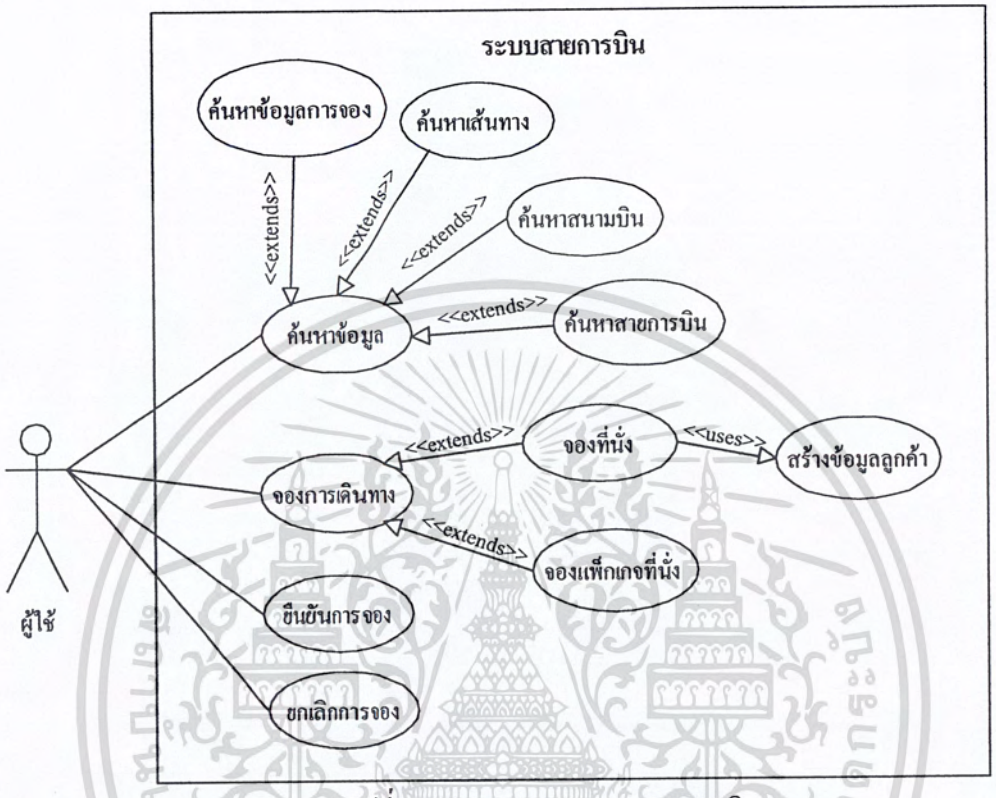
จองการเดินทาง

จองที่นั่ง

จองชุดที่นั่ง (เพื่อจัดทำแพกเกจทัวร์)

ยืนยันการจอง

ยกเลิกการจอง



รูปที่ 10-2 ยูสเคสของระบบสายการบิน

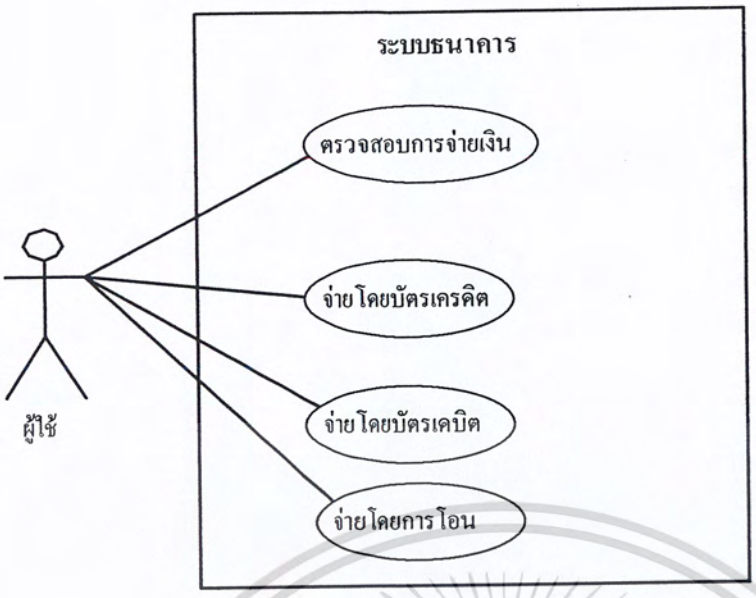
10.2.3 ยูสเคสของระบบธนาคาร

จ่ายเงินด้วยบัตรเครดิต

จ่ายเงินด้วยบัตรเดบิต

จ่ายเงินด้วยวิธีการโอน

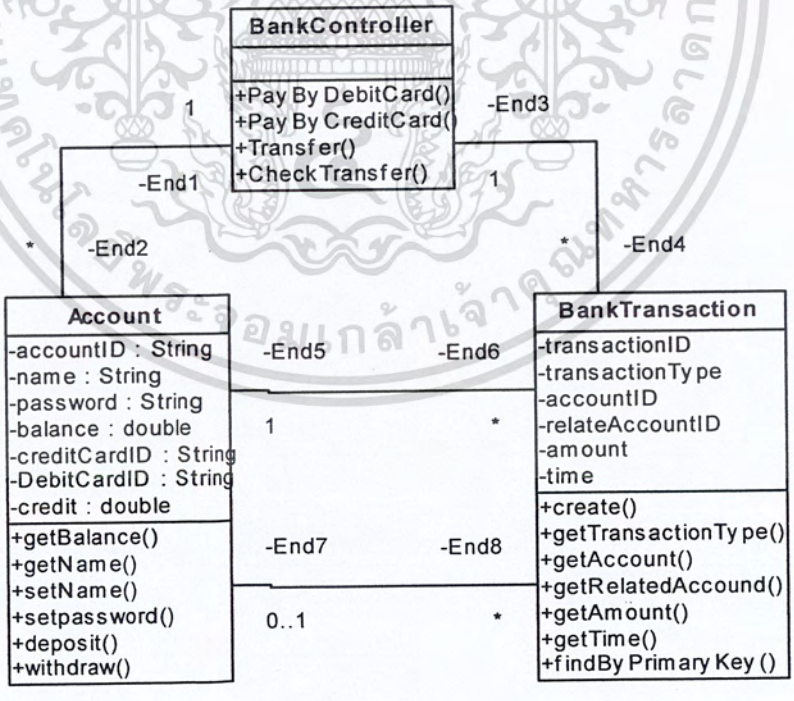
การตรวจสอบการจ่ายเงิน



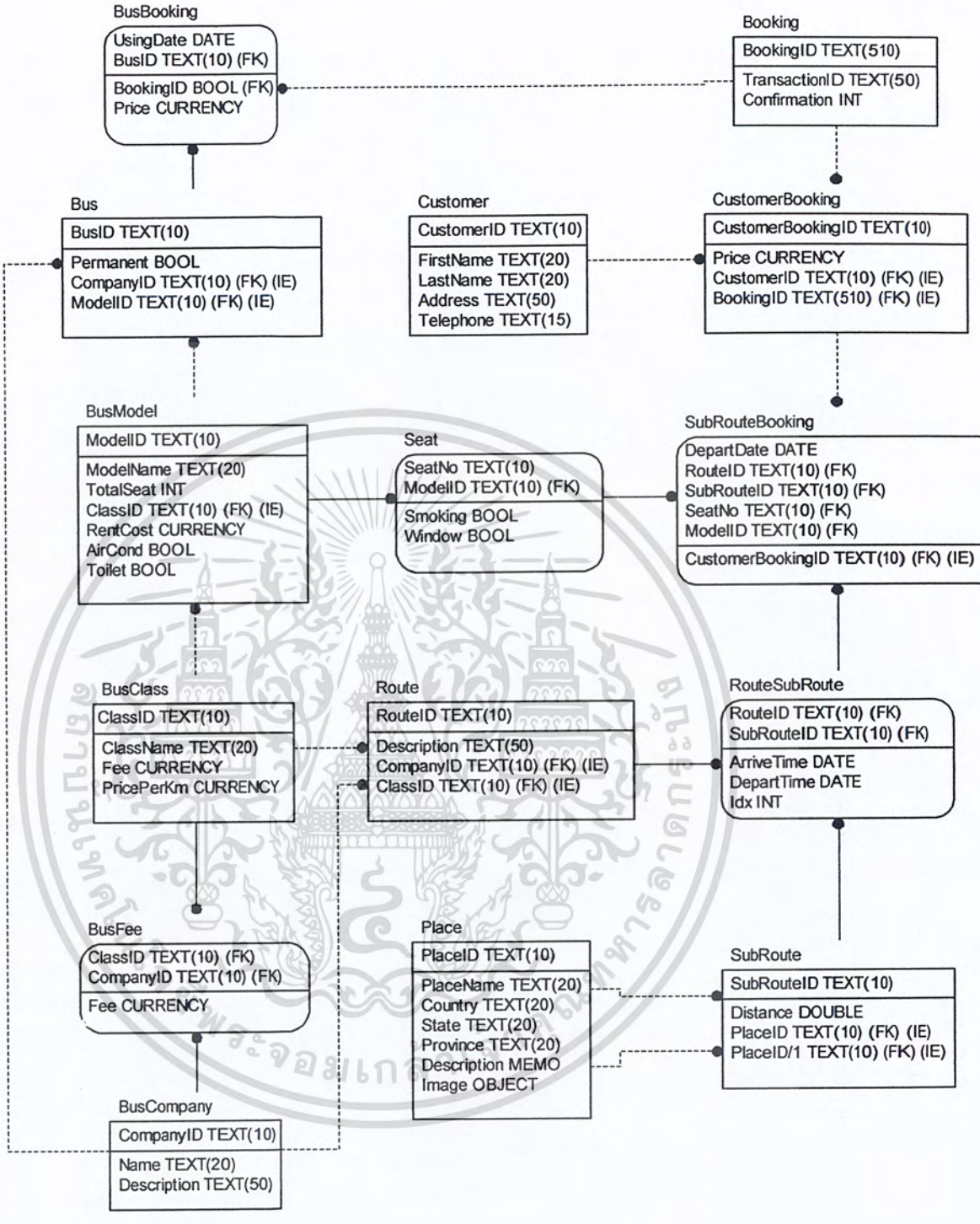
รูปที่ 10-3 ยูสเคสของระบบธนาคาร

10.3 ออกแบบฐานข้อมูล

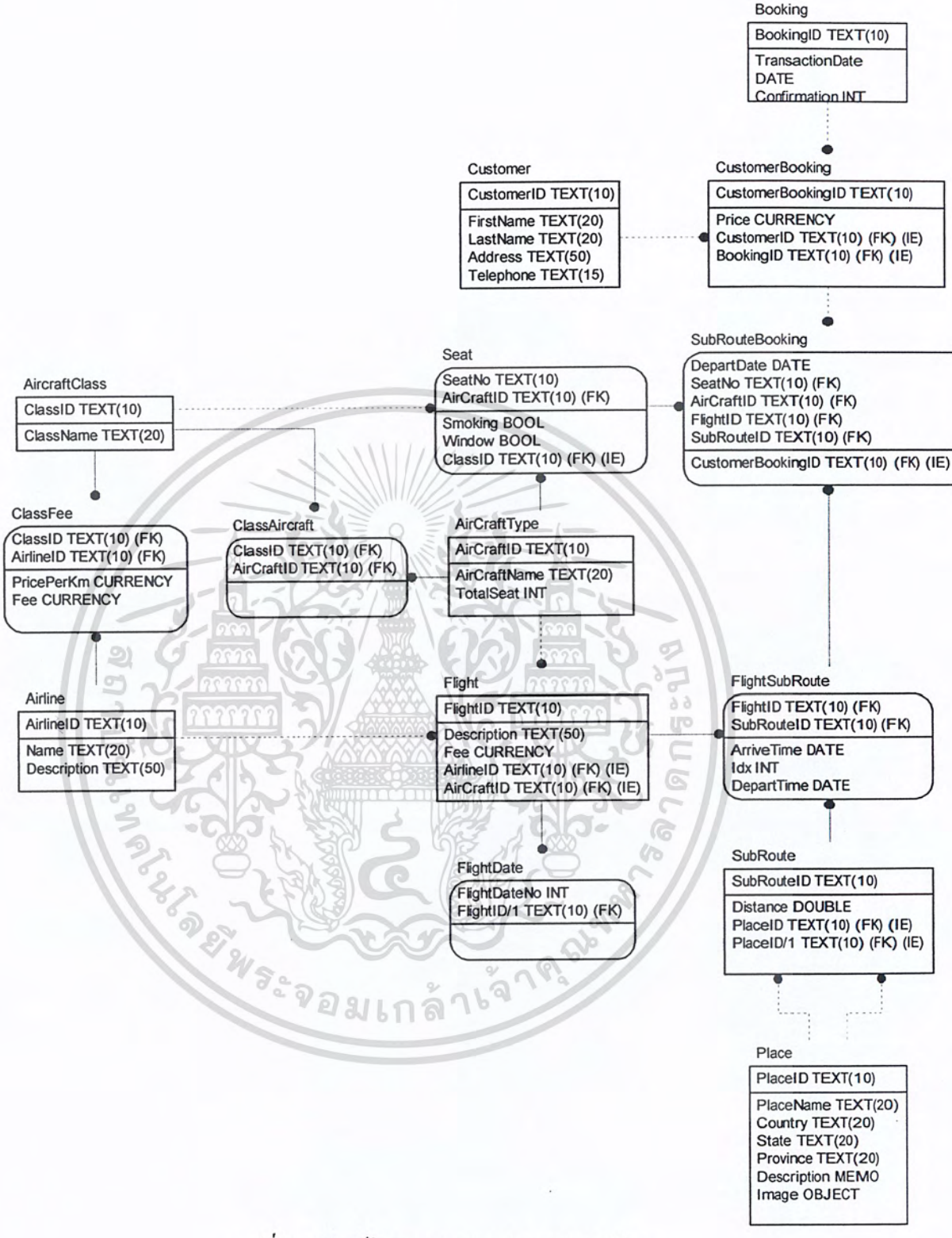
การออกแบบลักษณะของข้อมูลของระบบงานมีความสำคัญมาก เราใช้อีอาร์ไดอะแกรม เป็นสัญลักษณ์แสดงความสัมพันธ์ระหว่างข้อมูลดังนี้



รูปที่ 10-4 อีอาร์ไดอะแกรมของระบบธนาคาร



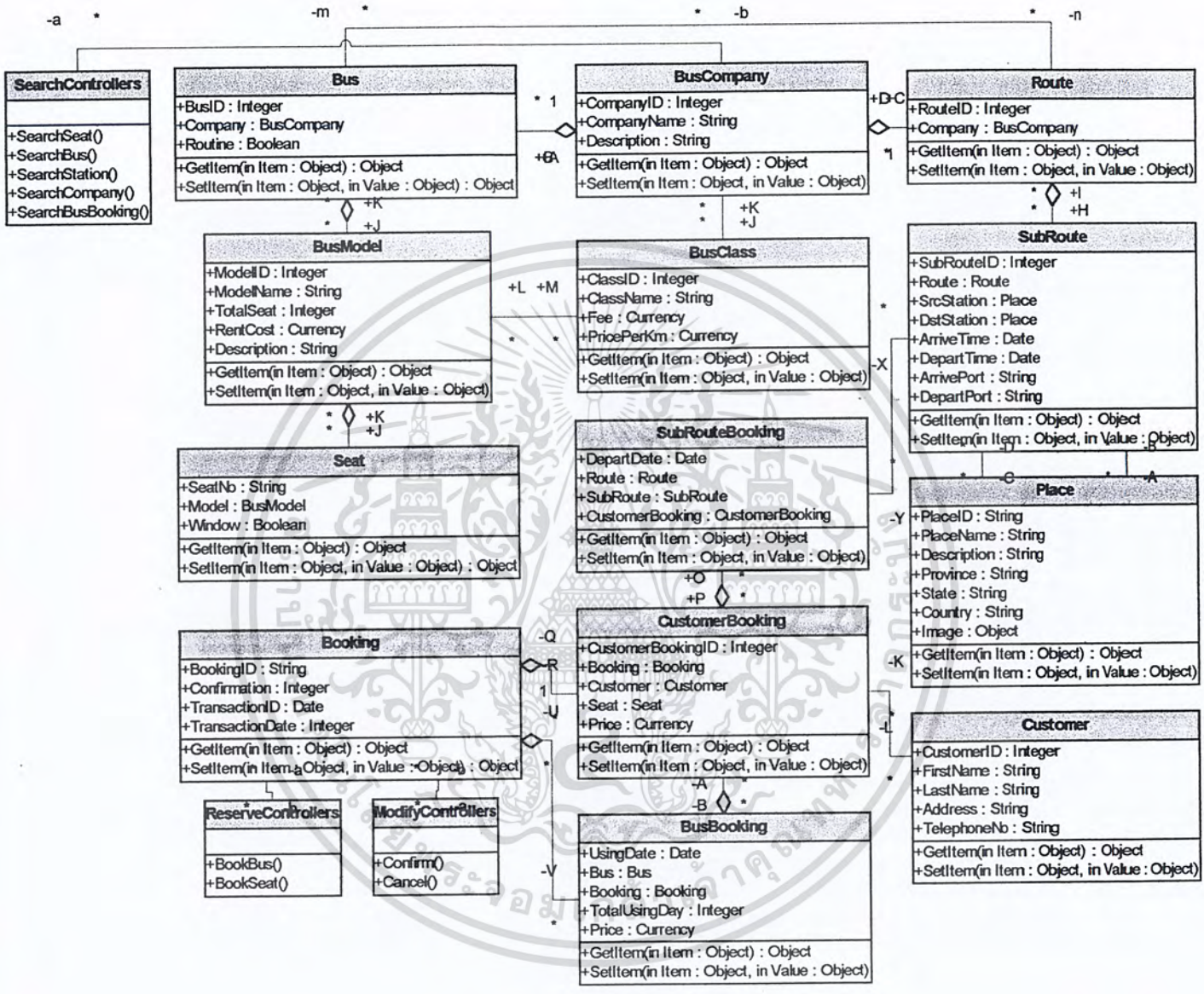
รูปที่ 10-5 อีอาร์ไดอะแกรมของระบบรถโดยสาร



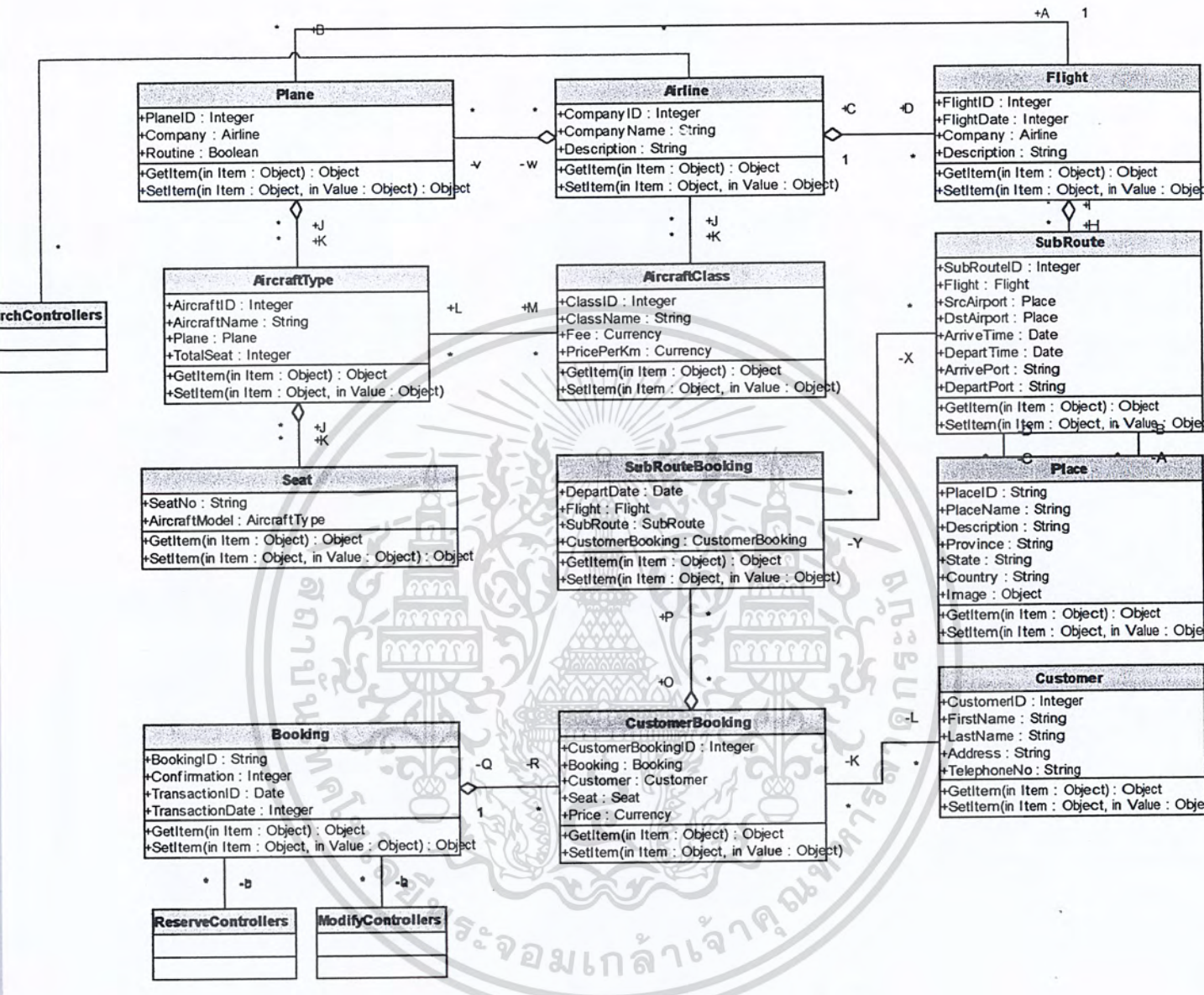
รูปที่ 10-6 อีอาร์ไดอะแกรมของระบบสายการบิน

10.4 ออกแบบคลาสไดอะแกรม

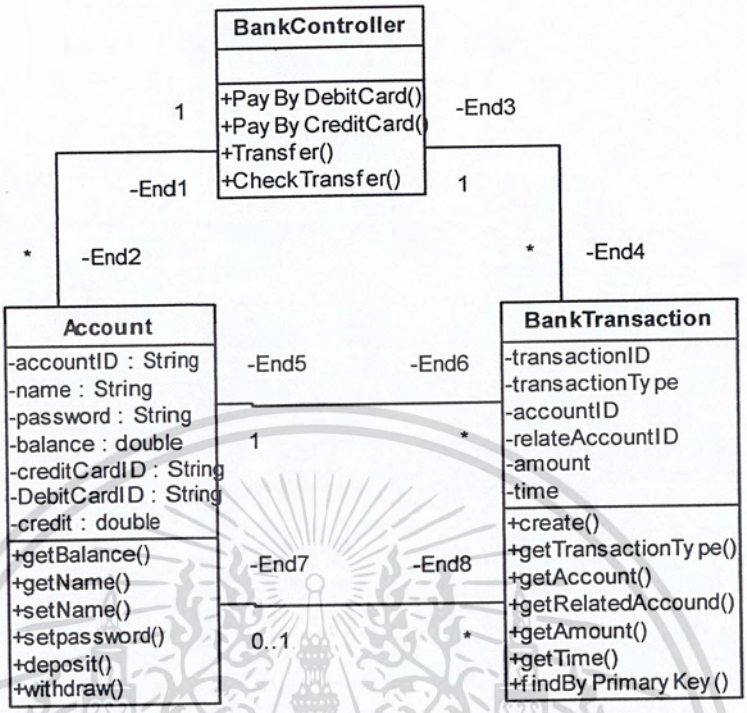
เรานำอีอาร์ไดอะแกรมมาเป็นหาความสัมพันธ์ของข้อมูลในรูปแบบของคลาสไดอะแกรม ซึ่งโดยส่วนใหญ่เราสามารถแม็บเอ็นติตีมากกว่า 1 เอ็นติตีไปเป็นคลาส 1 คลาส



รูปที่ 10-7 คลาสไดอะแกรมของระบบรถโดยสาร



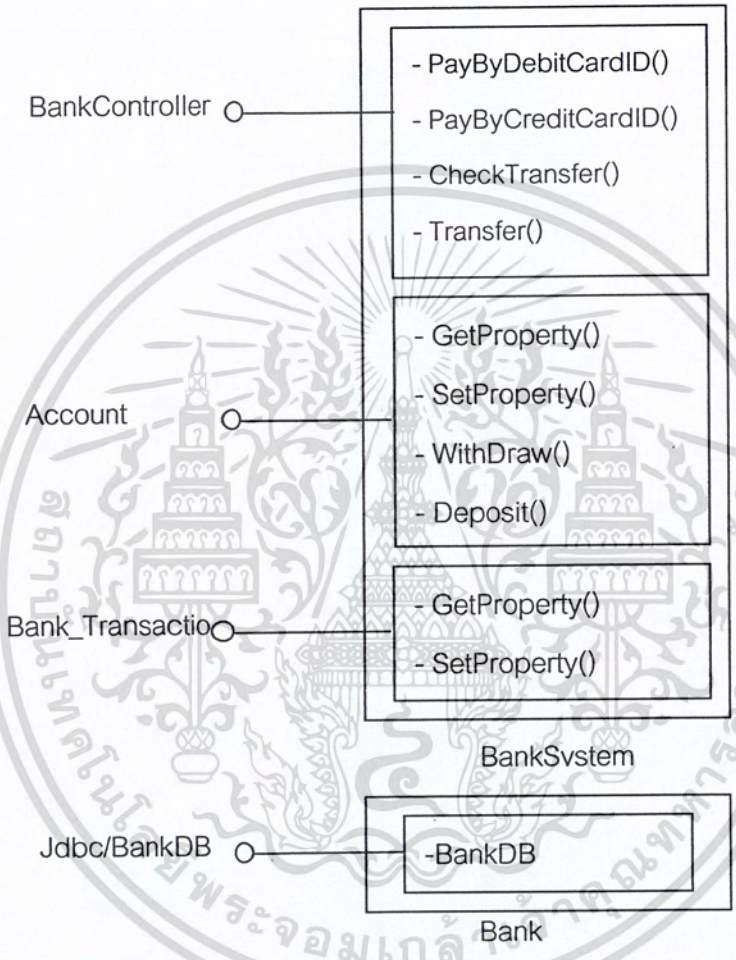
รูปที่ 10-8 คลาสไดอะแกรมของระบบสายการบิน



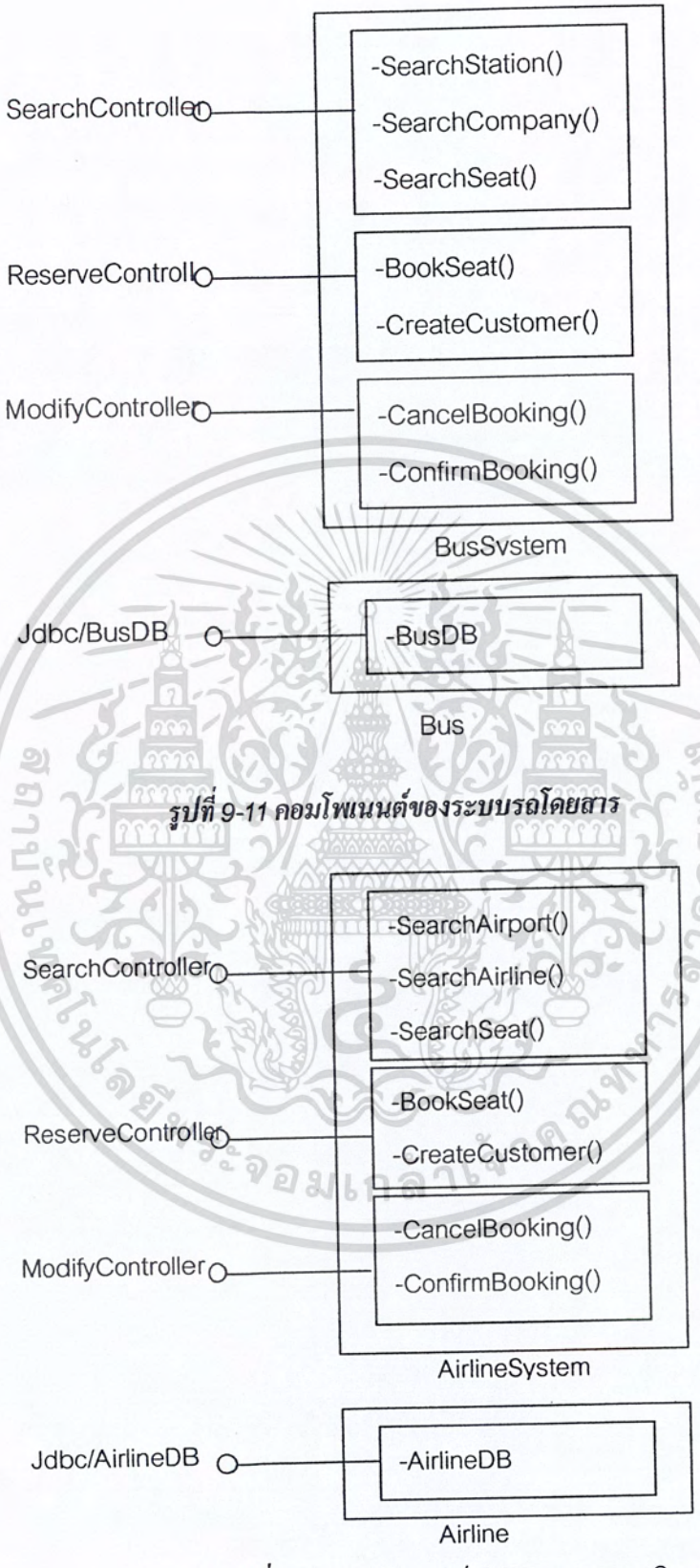
รูปที่ 10-9 คลาสไดอะแกรมของระบบธนาคาร

10.5 **ออกแบบคอมโพเนนต์ไดอะแกรม**

จากคลาสไดอะแกรมที่ได้ เรานำมาออกแบบเป็นคอมโพเนนต์ไดอะแกรม ที่เป็นคอมโพเนนต์ของของระบบงานในลักษณะของ Enterprise Java Bean



รูปที่ 10-10 คอมโพเนนต์ของระบบธนาคาร



รูปที่ 9-11 คอมโพเนนต์ของระบบรถโดยสาร

รูปที่ 9-12 คอมโพเนนต์ของระบบสายการบิน

10.6 การสร้างโปรแกรม (Coding)

10.6.1 ระบบธนาคาร

ในส่วนการทำงานของธนาคารนั้นประกอบไปด้วย 2 เอนติตี้บีนส์ และ 1 เซสชั่นบีนส์ ดังนี้

เอนติตี้บีนส์ (entity bean) ประกอบด้วย

- Account Entity Bean
- Bank_Transaction bean

การทำงานของทั้ง 2 ตัวนี้จะเป็นในลักษณะการติดต่อกับฐานข้อมูล ซึ่งรวมถึงการทำงานที่เกี่ยวข้องกับการ Select, Update, Delete ของข้อมูล โดยอาจมาจากการเรียกใช้ของบีนส์ตัวอื่น ๆ เช่น Session Bean และโดยส่วนมากแล้วใน เอนติตี้บีนส์นั้นจะประกอบไปด้วย method get /set attribute ต่าง ๆ ของข้อมูลที่อยู่ในฐานข้อมูล

เซสชั่นบีนส์ (Session Bean)

ประกอบด้วยส่วนของการควบคุมการทำงานและเรียกใช้ method ที่อยู่ในตัวเอนติตี้บีนส์ต่าง ๆ โดยตัวมันจะไม่เข้าถึงฐานข้อมูลโดยตรง ซึ่งในระบบนี้ก็คือส่วนของ BankControllerBean ที่เป็นตัวที่มีอินเตอร์เฟสต่าง ๆ ที่ผู้ใช้สามารถเรียกใช้ได้ คือ

เมธอด payByDebitCardID()

เมธอด payByCreditCardID()

เมธอด checkTransfer()

เมธอด transfer()

นอกจากนี้ยังประกอบด้วยเซสชั่นบีนส์ที่ทำหน้าที่ในการสร้างยูนิคคีย์ เพื่อสร้างรหัสเฉพาะต่างๆ คือ UUIDGenerator Session Bean

10.6.2 การสร้างและการออกแบบโปรแกรมในส่วนของระบบสายการบิน

ในระบบ Airline System นั้นจะประกอบไปด้วย Enterprise Java Bean ที่เป็น Session Bean ทั้งหมด 4 Bean คือ

1.SearchControllerBean

2.ReserveControllerBean

3.ModifyControllerBean

4.UUIDGeneratorBean

การทำงานของ session ทั้ง 4 นี้จะมีการอ้างถึงฐานข้อมูลโดยตรง นั่นคือ จะไม่มีการใช้ entity bean ดังนั้นเมื่อจะทำการอ้างถึงข้อมูลต่าง ๆ ที่ต้องการจากฐานข้อมูล ก็จะสามารถทำได้โดยใช้คำสั่ง SQL Command ได้เลย ซึ่งการทำงานใน Bean ต่าง ๆ นั้นสามารถอธิบายได้ดังนี้

SearchControllerBean

Session Bean นี้มีหน้าที่ในการจัดการ ควบคุมการเรียกใช้ interface ต่าง ๆ ที่ใช้ในการค้นหาข้อมูลเกี่ยวกับสายการบิน เที่ยวบิน หรือที่นั่งที่จะจองที่ผู้ใช้ต้องการจะจอง ดังนั้นภายใน Bean นี้ จะประกอบด้วย interface ต่าง ๆที่สามารถเรียกใช้ได้คือ

1. SearchAirline(String keyword) ใน interface นี้จะทำหน้าที่ในการค้นหาสายการบินทั้งหมดในฐานข้อมูลที่ตรงกับข้อมูลหรือ keyword ที่ใส่ผู้ใช้เข้าไปเพื่อทำการค้นหา โดยหลังจากการค้นหาแล้วนั้นผลลัพธ์ที่ส่งกลับมาแสดงยังผู้ใช้นั้นจะอยู่ในรูปของ AirlineID และ AirlineName ซึ่งในการคันหาข้อมูลนี้จากฐานข้อมูลนั้นสามารถเขียนเป็นคำสั่ง SQL Command ได้คือ

```
SELECT AIRLINEID, AIRLINENAME
FROM AIRLINE
WHERE DESCRIPTION LIKE '%keyword%'
```

2. SearchAirport(String keyword) ใน interface นี้นั้นจะทำหน้าที่ในการค้นหาท่าอากาศยานหรือ Airport ทั้งหมดที่อยู่ในฐานข้อมูลที่ตรงกับข้อมูลหรือ keyword ที่ใส่ผู้ใช้เข้าไปเพื่อทำการค้นหา โดยหลังจากการค้นหาแล้วนั้นผลลัพธ์ที่ส่งกลับมาแสดงยังผู้ใช้นั้นจะอยู่ในรูปของ AirportID และ AirportName ซึ่งในการคันหาข้อมูลนี้จากฐานข้อมูลนั้นสามารถเขียนเป็นคำสั่ง SQL Command ได้คือ

```
SELECT AIRPORTID, AIRPORTNAME
FROM PLACE
WHERE DESCRIPTION LIKE '%keyword%'
```

3 SearchSeat( ) ใน interface นี้นั้นจะทำหน้าที่ในการค้นหาที่นั่งของเที่ยวบินที่ต้องการ โดยผู้ใช้ต้องใส่ข้อมูลที่จำเป็นต้องใช้ในการค้นหา ซึ่งในการค้นหานั้น ระบบจะพยายามค้นหาที่นั่งที่ตรงตามความต้องการของผู้ใช้บริการมากที่สุด ถ้าการค้นหาสำเร็จก็จะแสดงผลการค้นหาทั้งหมดให้แก่ผู้ใช้ เพื่อใช้ในการพิจารณาและตัดสินใจต่อไป

ReserveControllerBean

หลังจากที่ผู้ใช้ได้ทำการค้นหาเที่ยวบินและที่นั่งที่ต้องการแล้ว ผู้ใช้ก็สามารถเข้ามาทำการจองเที่ยวบินที่ต้องการนั้นได้ โดยในการจองนั้นผู้ใช้จะต้องระบุ CustomerID ของผู้ใช้นั้นด้วย ซึ่งถ้าหากผู้ใช้นั้นยังไม่มี CustomerID ทางระบบ Olala Tour ก็จะต้องทำการออก CustomerID ใหม่ให้แก่ผู้ใช้บริการรายใหม่ นั่นคือใน ReserveController Session Bean นั้นจะต้องมีหน้าที่ในการจัดการ ควบคุมการเรียกใช้ interface ต่าง ๆ ที่ใช้ในการจองเที่ยวบิน ยังต้องมี interface ในการสร้างส่วนเก็บข้อมูลของลูกค้ารายใหม่ด้วย ดังนั้นภายใน Bean นี้ จะประกอบด้วย interface ต่าง ๆที่สามารถเรียกใช้ได้คือ

1. CreateCustomer( ) Interface นี้จะถูกเรียกเมื่อต้องการที่จะเก็บข้อมูลของลูกค้ารายใหม่ที่เข้ามาขอใช้บริการกับระบบ Olala Tour โดยข้อมูลที่จำเป็นที่ผู้ใช้จะต้องให้กับระบบเพื่อทำการบันทึกนั้นจะประกอบไปด้วย ชื่อ, นามสกุล, ที่อยู่, เบอร์โทรศัพท์ เป็นต้น โดยหลังจากที่ระบบได้ทำการบันทึกข้อ

มูลผู้ใช้เรียบร้อยแล้ว ระบบก็จะสร้าง CustomerID เพื่อให้ลูกค้านั้นสามารถนำไปใช้ในการขอบริการต่าง ๆ ของระบบต่อไปได้

2. BookSeat ( ) Interface นี้จะถูกเรียกใช้เมื่อผู้ใช้ต้องการที่จะเข้ามาจองที่นั่ง หรือเที่ยวการบินที่ต้องการ โดยการใส่ข้อมูลของที่นั่งที่จะทำการจองให้แก่ระบบ ซึ่งข้อมูลที่ใส่เข้ามานั้นอาจเป็นการจองทีละหลาย ๆ ที่นั่ง และในแต่ละที่นั่งอาจมีข้อมูลการเดินทางที่แตกต่างกันได้ โดยระบบจะเริ่มการจองทีละ 1 ที่นั่งจนสามารถทำการจองได้จนครบทั้งหมด หลังจากนั้นระบบก็จะเริ่มการทำงานโดยมีขั้นตอน ดังนี้ หาหมายเลยที่นั่งที่สามารถจองได้และตรงกับความต้องการของผู้ใช้มากที่สุด เมื่อได้หมายเลขที่นั่งที่จะจองแล้ว ระบบจะทำการหาว่าเที่ยวบินที่จะไปนั้นจะมีการจอดระหว่างทางที่ไหนบ้าง เพื่อที่จะสามารถนำข้อมูลการจองไปบันทึกเก็บไว้ใน SubrouteBooking ซึ่งมีประโยชน์ในการดูว่า Route ไหนมีผู้จองแล้วเท่าไหร่ และจะสามารถจองได้อีกหรือไม่ หลังจากทำการบันทึกลงใน SubrouteBooking หมดแล้ว ระบบก็จะสร้าง CustomerBookingID เพื่อนำข้อมูลการจองที่นั่งทีนี้ไปบันทึกลงใน CustomerBooking ระบบจะตรวจสอบว่าข้อมูลที่ผู้ใช้ส่งมาจองนั้นได้ทำการจองครบแล้วหรือยัง ถ้ายังไม่ครบระบบก็จะเข้าไปทำการหาหมายเลขที่นั่งอีกครั้งสำหรับการจองอีกที่หนึ่ง ทำเช่นนี้จนสามารถจองได้ครบทุกที่นั่ง เมื่อจองได้ครบแล้ว ระบบจะสร้าง BookingID ขึ้นมาเพื่อนำข้อมูลจากการจองครั้งนี้ไปบันทึกเก็บไว้ นั่นคือการเข้ามาจองครั้งหนึ่ง สามารถจองกี่ที่ก็ได้ แต่การจองในแต่ละครั้งนั้นจะต้องทำการบันทึกข้อมูลต่าง ๆ ลงใน 1 Bookinhg เสมอ

ModifyControllerBean

หลังจากที่ผู้ใช้เข้าไปทำการจองในระบบเรียบร้อยแล้วนั้น ผู้ใช้มีสิทธิที่จะเข้าไปทำการแก้ไขหรือเปลี่ยนแปลงข้อมูลการจองเดิมที่ได้จองไว้ แต่ทั้งนี้ต้องอยู่ในขอบเขตและเงื่อนไขที่ทางระบบได้กำหนดไว้เช่น การที่จะสามารถเข้าไปทำการยกเลิกเที่ยวบินที่ได้จองไว้นั้น ผู้ใช้สามารถทำได้เมื่อวันที่เข้าไปยกเลิกนั้นเป็นวันก่อนกำหนดการเดินทางจริงอย่างน้อย 3 วัน เป็นต้น โดย interface ที่สามารถเรียกใช้ได้ใน Bean นี้คือ

1. CancleBooking(String BookingID) โดยปกติแล้วในตารางฐานข้อมูลนั้นจะเก็บสถานะการจองของผู้ใช้ไว้โดยแสดงด้วยตัวเลขอินทิเจอร์ ดังนี้

0 หมายถึง ผู้ใช้ทำการจองไว้แล้วแต่ยังไม่ได้ทำการยืนยัน (Confirm)

หมายถึง ผู้ใช้ทำการจองไว้และได้ทำการยืนยันไว้เรียบร้อยแล้ว

หมายถึง ผู้ใช้เคยทำการจองไว้และได้เข้าไปทำการยกเลิกการจองนั้นแล้ว หรือทำการ Cancel Booking นั้นแล้ว

ดังนั้นการที่ผู้ใช้จะสามารถเข้าไปยกเลิกการจองได้นั้น ระบบจะต้องไปตรวจสอบสถานะการจองของผู้ใช้ก่อนว่าอยู่ในสถานะที่สามารถยกเลิกได้หรือไม่นั่นคือต้องตรวจสอบว่าสถานะเป็น 0 หรือไม่ ถ้าใช่ทางระบบก็จะสามารถให้ทำการยกเลิกได้โดยการเปลี่ยนสถานะจาก 0 เป็น 2 นั่น เอง แต่ก่อนหน้า

นั้นทางระบบต้องตรวจสอบในเรื่องของเวลาที่ผู้ใช้เข้ามาทำการยกเลิกก่อนว่าอยู่ในเงื่อนไขที่กำหนดหรือไม่ด้วย ซึ่งหากไม่เป็นไปตามที่กำหนด ระบบก็จะไม่อนุญาติให้ทำการยกเลิก Booking นั้น

2. ConfirmBooking(String BookingID) ในกรณีของการที่ผู้ใช้จะเข้าไปทำการยืนยันหรือ Confirm นั้นก็มีกี่ทำงานที่คล้ายกับการเข้าไปยกเลิกนั่นคือทางระบบจะต้องไปตรวจสอบสถานะการจองของผู้ใช้ก่อนว่าอยู่ในสถานะที่สามารถยืนยันได้หรือไม่ นั่นคือต้องตรวจสอบว่าสถานะนั้นเป็น 0 หรือไม่ และเวลาที่เข้าไปทำการยืนยันนั้นอยู่ในขอบเขตที่ถูกต้องหรือไม่ ถ้าเงื่อนไขเหล่านี้เป็นจริงผู้ใช้ก็จะสามารถเข้าไปทำการยืนยันการจองนั้นได้ แต่ถ้าเงื่อนไขไม่ถูกต้องผู้ใช้จะไม่ถูกอนุญาติให้เข้าไปทำการยืนยันได้

4.UUIDGeneratorBean

Session Bean นี้ไม่ได้มีการทำงานที่เกี่ยวข้องกับฐานข้อมูลของระบบ และไม่ได้เป็น Bean ที่มี interface ให้ผู้ใช้เรียกใช้งานได้ แต่บีนส์นี้มีไว้เพื่อใช้ในการสร้างหรือ generate รหัสเฉพาะหรือ Unique key เพื่อให้แต่ละรหัสนั้นไม่ซ้ำกันเช่นการสร้าง CustomerID, CustomerBookingID หรือ BookingID เพื่อใช้อ้างอิงในการเข้าถึงฐานข้อมูลนั้นได้อย่างถูกต้อง

10.6.3 การสร้างและการออกแบบโปรแกรมในส่วนของระบบรถโดยสาร

ในระบบ Bus System นั้นจะประกอบไปด้วย Enterprise Java Bean ที่เป็น Session Bean ทั้งหมด เช่นเดียวกับในระบบ Airline System ซึ่งประกอบด้วย 4 Bean คือ

1.SearchControllerBean

2.ReserveControllerBean

3.ModifyControllerBean

4.UUIDGeneratorBean

การทำงานของ session ทั้ง 4 นี้จะมีการทำงานที่คล้ายคลึงกับในส่วนการทำงานของ Airline System ทั้งในเรื่องของการอ้างถึงฐานข้อมูลโดยใช้คำสั่ง SQL Command ได้เลย โดยในการทำงานของ Bean ต่าง ๆ นั้นสามารถอธิบายได้ดังนี้

SearchControllerBean

Session Bean นี้มีหน้าที่ในการจัดการ ควบคุมการเรียกใช้ interface ต่าง ๆ ที่ใช้ในการค้นหาข้อมูลเกี่ยวกับสถานีรถบัส บริษัทรถบัส หรือที่นั่งว่างที่สามารถจองได้ ดังนั้นภายใน Bean นี้ จะประกอบด้วย interface ต่าง ๆที่สามารถเรียกใช้ได้คือ

1. SearchBusCompany(String keyword) ใน interface นี้จะทำหน้าที่ในการค้นหาบริษัทรถบัสทั้งหมดในฐานข้อมูลที่ตรงกับข้อมูลหรือ keyword ที่ใส่ผู้ใช้เข้าไปเพื่อทำการค้นหา โดยหลังจากการค้นหาแล้วนั้นผลลัพธ์ที่ส่งกลับมาแสดงยังผู้ใช้นั้นจะอยู่ในรูปของ CompanyID และ CompanyName ซึ่งในการค้นหาข้อมูลนี้จากฐานข้อมูลนั้นสามารถเขียนเป็นคำสั่ง SQL Command ได้คือ

```
SELECT COMPANYID, COMPANYNAME
FROM COMPANY
WHERE DESCRIPTION LIKE '%keyword%'
```

2. SearchStation(String keyword)

ใน interface นี้นั้นจะทำหน้าที่ในการค้นสถานีรถบัสหรือ Bus station ทั้งหมดที่อยู่ในฐานข้อมูลที่ตรงกับข้อมูลหรือ keyword ที่ใส่ผู้ใช้เข้าไปเพื่อทำการค้นหา โดยหลังจากการค้นหาแล้วนั้นผลลัพธ์ที่ส่งกลับมาแสดงยังผู้ใช้นั้นจะอยู่ในรูปของ PlaceID และ PlaceName ซึ่งในการดันหาข้อมูลนี้จากฐานข้อมูลนั้นสามารถเขียนเป็นคำสั่ง SQL Command ได้คือ

```
SELECT PLACEID, PLACENAME
FROM PLACE
WHERE DESCRIPTION LIKE '%keyword%'
```

3. SearchSeat( )

ใน interface นี้นั้นจะทำหน้าที่ในการค้นหาที่นั่งของรถบัสที่ต้องการจะไป โดยผู้ใช้ต้องใส่ข้อมูลที่จำเป็นต้องใช้ในการค้นหา ซึ่งในการค้นหานั้น ระบบจะพยายามค้นหาที่นั่งที่ตรงตามความต้องการของผู้ใช้บริการมากที่สุด ถ้าการค้นหาสำเร็จก็จะแสดงผลการค้นหาทั้งหมดให้แก่ผู้ใช้ เพื่อใช้ในการพิจารณาและตัดสินใจต่อไป

ReserveControllerBean

หลังจากที่ผู้ใช้ได้ทำการค้นหารถบัสและที่นั่งที่ต้องการแล้ว ผู้ใช้ก็สามารถเข้ามาทำการจองที่นั่งในรถบัสที่ต้องการนั้นได้ โดยในการจองนั้นผู้ใช้จะต้องระบุ CustomerID ของผู้ใช้นั้นด้วย ซึ่งถ้าหากผู้ใช้นั้นยังไม่มี CustomerID ทางระบบ Olala Tour ก็จะต้องทำการออก CustomerID ใหม่ให้แก่ผู้ใช้บริการรายใหม่ นั่นคือใน ReserveController Session Bean นั้นจะต้องมีหน้าที่ในการจัดการ ควบคุมการเรียกใช้ interface ต่าง ๆ ที่ใช้ในการจองที่นั่งของรถบัส ยังต้องมี interface ในการสร้างส่วนเก็บข้อมูลของลูกค้ารายใหม่ด้วย ดังนั้นภายใน Bean นี้ จะประกอบด้วย interface ต่าง ๆที่สามารถเรียกใช้ได้คือ

1. CreateCustomer( ) Interface นี้จะถูกเรียกเมื่อต้องการที่จะเก็บข้อมูลของลูกค้ารายใหม่ที่เข้ามาขอใช้บริการกับระบบ Olala Tour โดยข้อมูลที่จำเป็นที่ผู้ใช้จะต้องให้กับระบบเพื่อทำการบันทึกนั้นจะประกอบไปด้วย ชื่อ, นามสกุล, ที่อยู่, เบอร์โทรศัพท์ เป็นต้น โดยหลังจากที่ระบบได้ทำการบันทึกข้อมูลผู้ใช้เรียบร้อยแล้ว ระบบก็จะสร้าง CustomerID เพื่อให้ลูกค้านั้นสามารถนำไปใช้ในการขอบริการต่าง ๆ ของระบบต่อไปได้ในภายหลัง

2. BookSeat ( ) Interface นี้จะถูกเรียกใช้เมื่อผู้ใช้ต้องการที่จะเข้ามาจองที่นั่งในรถบัสที่ต้องการ โดยการใส่ข้อมูลของที่นั่งที่จะทำการจองให้แก่ระบบ ซึ่งข้อมูลที่ใส่เข้ามานั้นอาจเป็นการจองทีละหลาย ๆ ที่นั่ง และในแต่ละที่นั่งอาจมีข้อมูลการเดินทางที่แตกต่างกันได้ โดยระบบจะเริ่มการจองทีละ 1 ที่

นั่งจนสามารถทำการจองได้จนครบทั้งหมด หลังจากนั้นระบบก็จะเริ่มการทำงานโดยมีขั้นตอนเหมือนกับในระบบการจองที่นั่งของ Airline System

ModifyControllerBean

หลังจากที่ผู้ใช้เข้าไปทำการจองในระบบเรียบร้อยแล้วนั้น ผู้ใช้มีสิทธิที่จะเข้าไปทำการแก้ไขหรือเปลี่ยนแปลงข้อมูลการจองเดิมที่ได้จองไว้ แต่ทั้งนี้ต้องอยู่ในขอบเขตและเงื่อนไขที่ทางระบบได้กำหนดไว้เช่น การที่จะสามารถเข้าไปทำการยกเลิกเที่ยวบินที่ได้จองไว้นั้น ผู้ใช้สามารถทำได้เมื่อวันที่เข้าไปยกเลิกนั้นเป็นวันก่อนกำหนดการเดินทางจริงอย่างน้อย 3 วัน เป็นต้น โดย interface ที่สามารถเรียกใช้ได้ใน Bean นี้คือ

CancleBooking(String BookingID)

โดยปกติแล้วในตารางฐานข้อมูลนั้นจะเก็บสถานะการจองของผู้ใช้ไว้โดยแสดงด้วยตัวเลขอินทิเจอร์ ดังนี้

0 หมายถึง ผู้ใช้ทำการจองไว้แล้วแต่ยังไม่ได้ทำการยืนยัน (Confirm)

1 หมายถึง ผู้ใช้ทำการจองไว้และได้ทำการยืนยันไว้เรียบร้อยแล้ว

2 หมายถึง ผู้ใช้เคยทำการจองไว้และได้เข้าไปทำการยกเลิกการจองนั้นแล้ว หรือทำการ Cancel Booking นั้นแล้ว

ดังนั้นการที่ผู้ใช้จะสามารถเข้าไปยกเลิกการจองได้นั้น ระบบจะต้องไปตรวจสอบสถานะการจองของผู้ใช้ก่อนว่าอยู่ในสถานะที่สามารถยกเลิกได้หรือไม่นั่นคือต้องตรวจสอบว่าสถานะเป็น 0 หรือไม่ ถ้าใช่ทางระบบก็จะสามารถให้ทำการยกเลิกได้โดยการเปลี่ยนสถานะจาก 0 เป็น 2 นั่น เอง แต่ก่อนหน้านั้นทางระบบต้องตรวจสอบในเรื่องของเวลาที่ผู้ใช้เข้ามาทำการยกเลิกก่อนว่าอยู่ในเงื่อนไขที่กำหนดหรือไม่ด้วย ซึ่งหากไม่เป็นไปตามที่กำหนด ระบบก็จะไม่อนุญาติให้ทำการยกเลิก Booking นั้น

1. ConfirmBooking(String BookingID)

ในกรณีของการที่ผู้ใช้จะเข้าไปทำการยืนยันหรือ Confirm นั้นก็มีกี่ทำงานที่คล้ายกับการเข้าไปยกเลิกนั่นคือทางระบบจะต้องไปตรวจสอบสถานะการจองของผู้ใช้ก่อนว่าอยู่ในสถานะที่สามารถยืนยันได้หรือไม่ นั่นคือต้องตรวจสอบว่าสถานะนั้นเป็น 0 หรือไม่ และเวลาที่เข้าไปทำการยืนยันนั้นอยู่ในขอบเขตที่ถูกต้องหรือไม่ ถ้าเงื่อนไขเหล่านี้เป็นจริงผู้ใช้ก็จะสามารถเข้าไปทำการยืนยันการจองนั้นได้ แต่ถ้าเงื่อนไขไม่ถูกต้องผู้ใช้จะไม่ถูกอนุญาติให้เข้าไปทำการยืนยันได้

UUIDGeneratorBean

Session Bean นี้มีการทำงานเหมือนกับ Bean ที่อยู่ในระบบ Airline System

บทที่ 11
บทวิจารณ์และสรุปผล

11.1 สรุปผลการดำเนินงาน

จากที่ได้ทำการออกแบบและวางโครงสร้างให้กับโปรแกรมที่ใช้งานในระบบ ซึ่งประกอบด้วยการทำงานที่เห็นได้ชัด คือ ส่วนของการจัดการและเรียกใช้ EJB การทำงานของเซิร์ฟเล็ต พบว่าการพัฒนาจัดสร้างโปรแกรมได้ทำไปทุกกการทำงานที่ได้วางไว้ แม้ว่าโครงสร้างปลีกย่อยจะได้ถูกเปลี่ยนแปลงเพื่อให้เหมาะสมกับการทำงานจริงในขณะทำการพัฒนาโปรเจคไปบ้าง โดยสำหรับผลการทำงานของโปรแกรมหลังจากได้นำไปทดลองใช้จรองนั้น พบว่าในระยะแรกของการใช้งานจะมีข้อผิดพลาดของโปรแกรมบางส่วนถูกแสดงออกมา การพัฒนาในระยะหลังจึงเกี่ยวข้องกับการทำให้ข้อผิดพลาดของโปรแกรมเหล่านี้หมดไป เนื่องจากจุดประสงค์ของการสร้างโปรแกรม คือ การทำให้การทำงานที่ได้ออกแบบไว้สามารถทำงานที่ได้ออกแบบไว้ออกมาถูกต้อง เพราะฉะนั้นการนำโปรแกรมออกทดลองใช้งานจึงเป็นส่วนสำคัญที่ทำให้พบข้อผิดพลาดที่อาจจะไม่สามารถพบได้ขณะเขียนโปรแกรม โดยหลังจากทำการแก้ไขข้อผิดพลาดที่พบเจอแล้ว โปรแกรมที่สร้างขึ้นในโครงงานก็สามารถทำงานตามที่ได้ออกแบบไว้อย่างถูกต้อง

สำหรับปัญหาที่พบในขณะทำการพัฒนาโปรแกรมนั้น นอกเหนือจากข้อผิดพลาดของการทำงานซึ่งเป็นเรื่องปกติที่ต้องพบเจอในขณะทำการเขียนโปรแกรมแล้ว มีปัญหาหลักอีกส่วนที่ผู้คิดจะจัดสร้างโปรแกรมเพื่อทำธุรกรรมอิเล็กทรอนิกส์ควรคำนึงถึง นั่นก็คือ การออกแบบโครงสร้างของระบบงานให้ชัดเจนก่อนลงมือทำ โดยควรจะรวบรวมข้อมูลของฟังก์ชันการทำงานต่าง ๆ ที่จำเป็นทั้งหมด และออกแบบโครงสร้างของการทำงานเหล่านี้ให้ชัดเจน แต่ก็มีความยืดหยุ่นพอที่หลังจากทำการเขียนโปรแกรมได้ง่ายโดยไม่ส่งผลกระทบกับโปรแกรมส่วนอื่นที่ไม่เกี่ยวข้อง ซึ่งถ้าโครงสร้างของโปรแกรมมีความชัดเจนมากเท่าไร ก็จะส่วผลให้การพัฒนาโปรแกรมมีความชัดเจนมากขึ้น และมีข้อผิดพลาดลดลง

การนำโปรแกรมตามที่ได้จัดสร้างขึ้นไปใช้งานนั้น ผู้นำไปใช้สามารถใช้งานได้เลย ตามฟังก์ชันการทำงานที่ได้อธิบายไว้ โดยไม่จำเป็นต้องเปลี่ยนแปลงตัวโปรแกรมแต่อย่างใด เนื่องจากการออกแบบโปรแกรมให้สามารถติดตั้งได้สะดวก โดยข้อดีของโปรแกรมนี้ก็คือแนวคิดที่แตกต่างจากการทำธุรกรรมทางอิเล็กทรอนิกส์รูปแบบอื่นออกไป นั่นคือ การนำเทคโนโลยีของ J2EE เข้ามาใช้งาน ซึ่งเป็นเทคโนโลยีที่ยังไม่ค่อยมีคนนำไปใช้ในการพัฒนาระบบ ส่วนนี้จึงนับว่าเป็นจุดที่น่าสนใจที่จะสามารถนำมาทดสอบการทำงานและประสิทธิภาพของระบบที่พัฒนาขึ้นด้วยเทคโนโลยีแบบนี้

11.2 แนวทางการพัฒนาต่อ

จากโปรแกรมการทำงานที่ได้พัฒนาขึ้น สามารถถูกนำไปพัฒนาต่อได้อีกหลายทาง สุดแล้วแต่แนวความคิดของผู้พัฒนาต่อไป เนื่องจากตัวโปรแกรมนั้นเปรียบเสมือนส่วนหลักที่มีการทำงานที่จำเป็นไว้ครบถ้วนแล้ว การทำให้โปรแกรมมีความสมบูรณ์และสามารถก้าวทันกับเทคโนโลยีของ J2EE จึงสามารถทำได้โดยง่าย ผู้พัฒนาจำเป็นต้องศึกษาโครงสร้างของโปรแกรมที่มีอยู่โดยละเอียด แล้วจึงเพิ่มการทำงานส่วน

อื่นลงไปให้เหมาะสม ซึ่งการทำงานเหล่านี้มีตัวอย่าง เช่น การเพิ่มการทำงานในส่วนของ JavaMail หรือในส่วนของ Extensive Markup Language (XML) ซึ่งเป็นเทคโนโลยีที่สามารถรองรับการทำงานได้ใน J2EE และมีประโยชน์ในด้านทำให้ผู้ใช้สามารถสร้างและดูแลเอกสารที่มีโครงสร้าง (structured documents) ที่บรรจุตัวอักษร (plain text) โดยทำให้สามารถ rendered หรือปรับเปลี่ยนการแสดงผลในรูปแบบที่หลากหลาย จุดประสงค์หลักของ XML คือการแยกส่วน ข้อมูลเพื่อประโยชน์ในการแสดงผลได้ นอกเหนือจากนี้ยังสามารถนำเอาเทคโนโลยีทางด้านของ Application Server หรือ Web Server ที่มีการพัฒนาออกมาใหม่ ๆ เพื่อนำมาทดลองใช้งานกับโปรแกรมของระบบที่ได้สร้างขึ้นได้เนื่องจาก โปรแกรมในโครงงานนั้นใช้หลักการเชิงคอมโพเน้นต์ทำให้ง่ายต่อการนำไปใช้เป็นอย่างมาก ซึ่งหลังจากการทดลองนั้นแล้วเราอาจจะเปรียบเทียบผลการทำงานกับระบบเดิมที่เคยใช้ได้ เพื่อเป็นแนวทางในการที่ผู้พัฒนาต่อสามารถนำไปใช้ได้ ทำให้จุดประสงค์ในการทำงานนี้มีความถูกต้องและสมบูรณ์มากยิ่งขึ้นไปอีก

ภาคผนวก ก ความต้องการด้านทรัพยากรของระบบ

Server

ระบบปฏิบัติการ วินโดว์ส 2000 แอดวานซ์ เซิร์ฟเวอร์ (Windows 2000 Advance Server)

- ซีพียู (CPU) ในขั้นต่ำคือ รุ่น Pentium 166 หรือสูงกว่านั้น ซึ่งที่แนะนำควรเป็น Pentium II ขึ้นไปจะดีกว่า
- แรม (RAM) ในขั้นต่ำอยู่ที่ 128 MBหรือสูงกว่า เพื่อให้ทำงานในระบบเครือข่ายได้อย่างมีประสิทธิภาพ
- เนื้อที่ฮาร์ดดิสก์ อย่างน้อย 2 GB โดยประมาณและมีเนื้อที่ว่างอยู่อย่างน้อย 900 MB – 1 GB หรือมากกว่า

Client

ระบบปฏิบัติการ วินโดว์ส 2000 โปรเฟสชันนัล (Windows 2000 professional)

- ซีพียู (CPU) ในขั้นต่ำคือ รุ่น Pentium ของ Intel หรือ ซีพียูของ AMD เช่น AMD K6
- แรม (RAM) ในขั้นต่ำอยู่ที่ 64 MB
- เนื้อที่ฮาร์ดดิสก์ อย่างน้อย 1 GB โดยประมาณและมีเนื้อที่ว่างอยู่อย่างน้อย 700-800 MB

ระบบปฏิบัติการ วินโดว์ส 98 (Windows 98)

- ซีพียู (CPU) ในขั้นต่ำคือ Pentium 166 ขึ้นไปและถ้าเป็นรุ่น MMX ก็จะดี
- แรม (RAM) ในขั้นต่ำอยู่ที่ 32 MB
- เนื้อที่ฮาร์ดดิสก์ตั้งแต่ 1 GBขึ้นไปและมีเนื้อที่ว่าง 120 MB ขึ้นไป

Software Requirement

1. ระบบปฏิบัติการ วินโดส์ว 2000 (Windows 2000)
   - ระบบปฏิบัติการ วินโดว์ส 2000 แอดวานซ์ เซิร์ฟเวอร์ (Windows 2000 Advance server)
   - ระบบปฏิบัติการ วินโดว์ส 2000 โปรเฟสชันนัล (Windows 2000 professional)
   - ระบบปฏิบัติการ วินโดว์ส 98(Window 98)
2. Java 2 SDK Enterprise Edition
3. Java 2 SDK 1.3 Standard Edition
4. Jbuilder 4

ภาคผนวก ข การติดตั้งและการกำหนดค่าเริ่มต้นของซอร์ฟแวร์เพื่อการสร้างและพัฒนาโปรแกรม

1. ระบบปฏิบัติการวินโดว์ส

1.1 ระบบปฏิบัติการวินโดว์ส 2000 (Windows 2000)

วินโดว์ส 2000 เป็นระบบปฏิบัติการที่พัฒนามาจาก วินโดว์ส เอ็นที 4.0 (Windows NT 4.0) โดยหัวใจหลักของระบบปฏิบัติการนั้นเป็นส่วนที่ได้รับอิทธิพลมาจาก วินโดว์ส เอ็นที 4.0 โดยที่มีส่วนที่ใช้ติดต่อกับผู้ใช้หรือ ยูสเซอร์ อินเตอร์เฟส (User Interface) ของวินโดว์ส 2000 นั้นจะใกล้เคียงกับ ยูสเซอร์ อินเตอร์เฟส ของ วินโดว์ส 98 (Windows 98)มากกว่า การออกแบบวินโดว์ส 2000 นั้นจะยึดตามแนวทางดังนี้

- พัฒนาต่อจากรากฐานของความมีเสถียรภาพของและศักยภาพต่าง ๆ ของ วินโดว์ส เอ็นที 4.0
- ผสมผสานลักษณะที่ดีของ วินโดว์ส 98
- ใช้งานได้ง่ายและสะดวกมากกว่าวินโดว์สเวอร์ชันก่อน
- บริหารระบบง่ายและค่าใช้จ่ายในการดูแลระบบต่ำ
- สามารถรองรับเทคโนโลยีใหม่ๆ ได้เป็นอย่างดีทั้งในด้านระบบเครือข่ายอินเทอร์เน็ต ฐานข้อมูล และการพัฒนาแอปพลิเคชั่น

<u>การตรวจสอบก่อนการติดตั้ง</u>

1. การตรวจสอบลิสต์รายชื่อ HCL(Hardware Compatibility List) ซึ่งเป็นรายการชื่ออุปกรณ์ฮาร์ดแวร์ต่าง ๆ ที่ วินโดว์ส 2000 รู้จักและสามารถทำงานอยู่ภายใต้ระบบปฏิบัติการวินโดว์ส 2000 ได้
2. การกำหนดดิสก์พาร์ติชันและระบบไฟล์
   - FAT 32

     เป็นระบบไฟล์ที่ใช้สำหรับวินโดว์ส 95 โอเอสอาร์ทู (Windows 95 OSR2) วินโดว์ส 98 และ วินโดว์ส 2000
   - NTFS เวอร์ชัน 5

     ระบบไฟล์ NTFS ของวินโดว์ส 2000 นั้นไม่เหมือนกับ NTFS ของ วินโดว์ส เอ็นที 4.0 โดย NTFS ของวินโดว์ส 2000 จะได้รับการพัฒนาและปรับปรุงประสิทธิภาพให้ดีกว่า อย่างเช่น Dynamic Volume Management , File and Folder Security , Disk Quota รวมถึง Encryption File System
3. การเป็นสมาชิกของเวิร์กกรุ๊ป หรือของโดเมน

   3.1 วินโดว์ส 2000 เซิร์ฟเวอร์

   วินโดว์ส 2000 เซิร์ฟเวอร์นั้น มันสามารถทำงานได้หลายหน้าที่ในระบบเครือข่ายหนึ่ง ๆ ทั้งนี้จะขึ้นอยู่กับความต้องการของผู้ติดตั้งหรือผู้ดูแลระบบ ซึ่งหน้าที่ดังกล่าวก็ได้แก่

• วินโดว์ 2000 เซิร์ฟเวอร์ที่ทำหน้าที่เป็นโดเมนคอนโทรลเลอร์

• วินโดว์ 2000 เซิร์ฟเวอร์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์ธรรมดาโดยเข้าเป็นสมาชิกของโดเมน เรียกว่า เมมเบอร์ เซิร์ฟเวอร์ (Member server)

• วินโดว์ 2000 เซิร์ฟเวอร์ ที่ทำหน้าที่เป็นเซิร์ฟเวอร์ธรรมดาภายใต้เวิร์กกรุ๊ปเล็ก ๆ โดยไม่ยุ่งเกี่ยวกับสมาชิกของโดเมนใด เรียกว่า สแตน อโลน เซิร์ฟเวอร์ (Stand Alone Server) สำหรับในโครงงานนี้ เราใช้ วินโดว์ 2000 เซิร์ฟเวอร์ ทำหน้าที่เป็นสแตน อโลน เซิร์ฟเวอร์ เพื่อเพื่อให้มันทำงานเฉพาะทางได้อย่างเต็มที่

3.2 วินโดว์ส 2000 โปรเฟสชันนัล

วินโดว์ส 2000 โปรเฟสชันนัล นั้น ได้ถูกจัดวางตำแหน่งให้ทำหน้าที่เป็นเครื่องไคลเอนต์ในระบบเครือข่ายอยู่แล้ว ซึ่งเราเพียงแค่ระบุเชื่อเวิร์กกรุ๊ปของเครื่องขณะติดตั้งเท่านั้น ในกรณีที่ติดตั้ง วินโดว์ส 2000 โปรเฟสชันนัล บนเครื่องคอมพิวเตอร์ธรรมดาที่บ้านหรือเครื่องใช้ส่วนตัวที่ไม่มีเน็ตเวิร์กการ์ดเสียบอยู่ โปรแกรมติดตั้งก็จะแสดงไดอะล็อกบ็อกพร้อมทั้งข้อความแจ้งให้ทราบว่ามันกำลังติดตั้งองค์ประกอบต่าง ๆ ของเน็ตเวิร์กอยู่ เผื่อเอาไว้สำหรับให้ผู้ใช้เชื่อมต่อกับเครือข่ายทางเน็ตเวิร์กในภายหลัง

1.2 ระบบปฏิบัติการวินโดว์ส 98 (Windows 98)

วินโดว์ส 98 เป็นระบบปฏิบัติการกึ่ง 32 บิต มีเสถียรภาพในการทำงานสูงกว่าวินโดว์ส 95 การจัดระบบไฟล์ในวินโดว์ส 98 แบบ FAT 32 จะทำให้คลัสเตอร์ มีขนาดเล็กกว่าการจัดระบบไฟล์แบบ FAT16 ที่ใช้ในวินโดว์ส 95 นอกจากนี้วินโดว์ส 98 ยังสนับสนุนการใช้งานพอร์ตยูเอสบี (USB) ซึ่งเป็นพอร์ตที่สนับสนุนในด้านการเชื่อมต่ออุปกรณ์ต่าง ๆ เข้ากับพีซี เมื่อใช้งานความสามารถด้านมัลติมีเดียบนเครื่องที่ใช้ซีพียู MMX ก็จะได้ภาพที่สวยงามและสมจริงมากขึ้น แต่ในส่วนของการรักษาความปลอดภัยนับว่ายังด้อยอยู่มากเมื่อเทียบกับ วินโดว์ส 2000 โปรเฟสชันนัล ซึ่งถูกใช้เพื่อเป็นไคลเอนต์เหมือนกัน เนื่องจาก วินโดว์ส 2000 โปรเฟสชันนัล มีการพัฒนามาจาก วินโดว์ส เอ็นที แต่ วินโดว์ส 98 นั้นพัฒนามาจากวินโดว์ส 95 หรือกล่าวได้ว่าเป็น วินโดว์ส 95 ที่แก้บั๊คแล้วนั่นเอง

2. Java 2 SDK Enterprise Edition

ทำการดาน์วโหลดโปรแกรม Java 2 SDK Enterprise Edition ได้จากเว็บไซต์

http://java.sun.com/j2ee/j2sdkee/

ทำการติดตั้งโปรแกรมจากไฟล์ที่ดาน์วโหลดเรียบร้อยแล้วให้ทำการเซ็ท PATH และ CLASSPATH ดังนี้

PATH= %j2ee_Home%\bin

CLASSPATH= %j2ee_Home%\lib\j2ee.jar

3. Java 2 SDK 1.3 Standard Edition

ทำการดาน์วโหลดโปรแกรม Java 2 SDK 1.3 Standard Edition ได้จากเว็บไซต์

http://java.sun.com/j2se/1.3/download-windows.html

ทำการติดตั้งโปรแกรมจากไฟล์ที่ดาน์วโหลดเรียบร้อยแล้วให้ทำการเซ็ท PATH และ CLASSPATH ดังนี้

PATH= %j2sdk_Home%\bin

CLASSPATH= %j2sdk_Home%\lib\dt.jar; %j2sdk_Home%\lib\tools.jar

ในการเซ็ต PATH และ CLASSPATH นั้นสามารถเซ็ตจากการแก้ไขที่ไฟล์ Autoexec.BAT หรือเซ็ตค่าที่ Environment Variables ซึ่งอยู่ใน แถบ Advanced ใน System Properties ก็ได้เช่นกันโดยสามารถเซ็ตจาก Autoexec.BAT หรือเซ็ตที่ Environment Variables ซึ่งอยู่ใน แถบ Advanced ใน System Properties ก็ได้เช่นกัน

4. Jbuilder เวอร์ชัน 4.0

Jbuilder เป็นโปรแกรมที่ช่วยในการพัฒนาแอพลิเคชั่นที่เขียนด้วยภาษาจาวา ไม่ว่าจะเป็น Applet JSP/Servlets ,JavaBean และ Enterprise Java Bean รวมไปถึง J2EE Applications ที่เขียนสำหรับ The Java 2 Platform ด้วย

เมื่อติดตั้ง Jbuilder 4.0 จากแผ่นโปรแกรมเรียบร้อยแล้วให้ทำการเซ็ต PATH ดังนี้

PATH= C:\JBuilder4\bin

จากตัวอย่างนั้น โปรแกรม Jbuilder 4.0 นั้นได้ ติดตั้งที่ Drive C ที่ไดเรคทอรี่ Jbuilder4 หากผู้ติดตั้งหรือผู้ดูแลระบบได้ติดตั้งที่ Drive อื่นก็เซ็ตให้ถูกต้อง

ภาคผนวก ค การเปรียบเทียบเทคโนโลยีระหว่าง COM Plus กับ EJB

1. Component Type Supported

- EJB จะมีคอมโพเนนท์ 2 ชนิดใน EJB1.1 Specification คือ Entity Bean และ Session Bean โดย Entity Bean จะเป็นคอมโพเนนท์ที่คอยจัดการข้อมูลในตารางในฐานข้อมูล และ Session Bean เป็นคอมโพเนนท์ที่ทำ business logic ให้ไคล์เอนท์คอยเรียกใช้บริการและติดต่อ Entity Bean เพื่อทำการใดๆกับข้อมูล
- COM+ จะมีคอมโพเนนท์เพียงแบบเดียวและจะแนะนำให้เขียนเป็น Stateless เพื่อไม่ต้องจัดการเกี่ยวกับ state และเพื่อ scalability

2. State Management

- EJB Session Bean จะแบ่งเป็น Stateful และ Stateless ซึ่ง Stateless Session Bean จะมีความ scalability ที่ดีกว่าและสามารถถูกเรียกใช้ได้จากหลายไคล์เอ็นท์พร้อมกัน และ Stateful Session Bean Container จะเป็นคนจัดการ state ให้โดยจะให้ไคล์เอ็นท์ ติอต่อกับคอมโพเนนท์ที่มี state และไคล์เอ็นท์ตัวนั้นเก็บอยู่เป็นคนให้บริการไคล์เอ็นท์คนนั้นต่อ แต่เนื่องจากอาจจะทีการ passivate state ของไคล์เอ็นท์ไปเก็บใน temporary storage จึงต้องมีการนำ state ของไคล์เอ็นท์ขึ้นมาจาก storage ไปใส่ในคอมโพเนนท์ก่อนจะให้บริการแก่ไคล์เอ็นท์
- COM+ จะมีการจัดการ state ของคอมโพเนนท์ผ่าน share property

3. Declarative Persistence

- EJB จะมี Entity Bean ประเภทหนึ่งเรียกว่า Container-Manage Persistance Entity Bean ซึ่งเราสามารถกำหนดว่า attribute ไหนบ้างของคอมโพเนนท์นั้นๆที่จะกำหนดให้ persist ลงฐานข้อมูล โดยไม่ต้องเขียนโค้ด SQL หรือเขียนเองก็ได้
- COM+ ต้องเขียน SQL ติดต่อกับตารางในฐานข้อมูลเองอย่างเดียว

4. Speed

- EJB จากการทดลองพบว่าการทำงานค่อนข้างช้าเมื่อเทียบกับ COM+ ในแอพพลิเคชันเดียวกัน เนื่องจากเป็นภาษา JAVA ซึ่งช้าทั้งการคอมไพล์และการรัน
- COM+ เป็น Native Codeb ล้วนจึงสามารทำงานได้เร็ว

5.Thread

- EJB คอนเทนเนอร์จะเป็นผู้จัดการ thread ของทุกๆคอมโพเนนท์เพียงผู้เดียวในการเขียนโค้ด ห้ามเขียนโค้ดที่ยุ่งเกี่ยวกับ thread โดยเด็ดขาดตาม Specification เพราะจะทำให้คอนเทนเนอร์ไม่สามารถจัดการ thread ได้อย่างอิสระ
- COM+ จะมีหลาย thread model ให้ใช้โดย programmer จะต้องเป็นผู้จัดการในเรื่องของการทำงาน thread เอง ซึ่งมีความยืดหยุ่นมากกว่า แต่ก็ทำให้ผิดพลาดได้ง่ายกว่า

6.Open

- EJB เป็นเพียง specification เท่านั้น ใครก็สามารถทำ product ออกมาขายตาม spec นี้ได้ และคอมโพเนนท์ที่เขียนขึ้นมาสามารถนำไป deploy บนทุกๆคอนเทนเนอร์ได้เหมือนกัน ทำให้เราสามารถเลือกคอนเทนเนอร์ได้อย่างอิสระ ไม่ติดกับ product หนึ่ง และบาง product เป็นแบบ open source และแจกฟรีทำให้นักพัฒนาสามารถรู้กลไลการทำงานทั้งหมดได้
- COM+ เป็นของ Microsoft เพียงผู้เดียว ตัวคอนเทนเนอร์มีมาให้พร้อม window2000 ทำให้เมื่อจะใช้ COM+ ต้องใช้ OS เป็น windows2000 เท่านั้น และมีคอนเทนเนอร์ของ Microsoft เพียงบริษัทเดียวแต่ภาษาที่ใช้ในการเขียนคอมโพเนนท์มีได้หลายภาษาเช่น VB, Delphi,VC แต่ EJB ใช้ Java ได้เพียงอย่างเดียว

ภาคผนวก ง Getting Started

ในส่วนนี้จะเป็นการอธิบายในขั้นตอนการพัฒนา ,การดีพลอย และการรันแอพพลิเคชั่นที่เป็นไคล์เอ็นท์-เซิฟเวอร์ ที่ใช้เอ็นเตอร์ไพร์จาวาบีน โดยเริ่มต้นจะกำหนดให้ไคล์เอ็นท์ชื่อว่า ConverterClient และให้เอ็นเตอร์ไพร์จาวาบีนชื่อว่า ConverterEJB

โดยมีขั้นตอนการทำงานตามลำดับขั้นดังนี้

1. ทำการเขียนโค้ดเอ็นเตอร์ไพร์จาวาบีน
2. สร้าง J2EE แอพพลิเคชัน
3. แพ็คเก็ตเอ็นเตอร์ไพร์จาวาบีน
4. ดีพลอย J2EE แอพพลิเคชัน
5. สร้างไคล์เอ็นท์
6. รันไคล์เอ็นท์

1. ทำการเขียนโค้ดเอ็นเตอร์ไพร์จาวาบีน

เอ็นเตอร์ไพร์จาวาบีนทุกๆบีนต้องประกอบไปด้วย

- Remote Interface
- Home Interface
- Enterprise bean class

1.1 การเขียนโค้ดรีโมตอินเตอร์เฟส (Coding the Remote Interface)

ภายในรีโมตอินเตอร์เฟสจะเป็นการประกาศเมทอดที่อนุญาตให้ไคล์เอ็นท์เรียกใช้ได้ โดยเอ็นเตอร์ไพร์จาวาบีนจะต้องทำการ implement เมทอดที่ประกาศในรีโมตอินเตอร์เฟสนี้ โค้ดภายในรีโมตอินเตอร์เฟสมีดังนี้

```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Converter extends EJBObject {

  public double dollarToYen(double dollars) throws RemoteException;
  public double yenToEuro(double yen) throws RemoteException;
```

```
}
```

1.2 การเขียนโค้ดโฮมอินเตอร์เฟส (Home the Home Interface)

โฮมอินเตอร์เฟสจะทำการกำหนดเมทอดเพื่อให้ไคล์เอ็นท์ทำการ create, find หรือ remove เอ็นเตอร์ไพร์จาวาบีน โดยภายในจะมีเพียงเมทอด create เพียงเมทอดเดียว โดยจะรีเทอร์นเป็นอ๊อบเจ็คที่เป็น type เดียวกับรีโมตอินเตอร์เฟส โค้ดภายในโฮมอินเตอร์เฟสมีดังนี้

```
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface ConverterHome extends EJBHome {

   Converter create() throws RemoteException, CreateException;
}
```

1.3 การเขียนโค้ดเอ็นเตอร์ไพร์จาวาบีนคลาส

ภายในเอ็นเตอร์ไพร์จาวาบีนตัวอย่างที่ยกมานี้จะเป็น stateless session bean ที่ชื่อว่า ConverEJB โดยภายในบีนนี้จะมีบีซิเนสเมทอดอยู่ 2 เมทอดที่ประกาศไว้ใน Converter รีโมตอินเตอร์เฟส คือ dollerToTYen และ YenToEuro โดยโค้ดภายใน ConverterEJB มีดังนี้

```
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class ConverterEJB implements SessionBean {

   public double dollarToYen(double dollars) {
```

```
    return dollars * 121.6000;
  }

  public double yenToEuro(double yen) {

    return yen * 0.0077;
  }

  public ConverterEJB() {}
  public void ejbCreate() {}
  public void ejbRemove() {}
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void setSessionContext(SessionContext sc) {}
}
```

1.4 ทำการคอมไพล์ซอร์สโค้ดของเอ็นเตอร์ไพร์จาวาบีน

ตอนนี้เมื่อสร้างทั้ง 3 ไฟล์ได้แล้วก็พร้อมที่จะทำการคอมไพล์ คือ ไฟล์รีโมตอินเตอร์เฟส (Converter.java) ,ไฟล์โฮมอินเตอร์เฟส (ConverterHome.java) และ ไฟล์เอ็นเตอร์ไพร์บีนคลาส (ConverterEJB.java) โดยทำการวิธีการคอมไพล์ทำดังนี้

1. สร้างไฟล์ compilerEJB.bat ขึ้นมาให้ไปอยู่ในไดเร็กทอรีที่ติดตั้ง J2EE SDK โดยภายในเขียนดังนี้

```
set J2EE_HOME=<installation-location>
set CPATH=.;%J2EE_HOME%\lib\j2ee.jar

javac -classpath %CPATH% ConverterEJB.java ConverterHome.java Converter.java
```

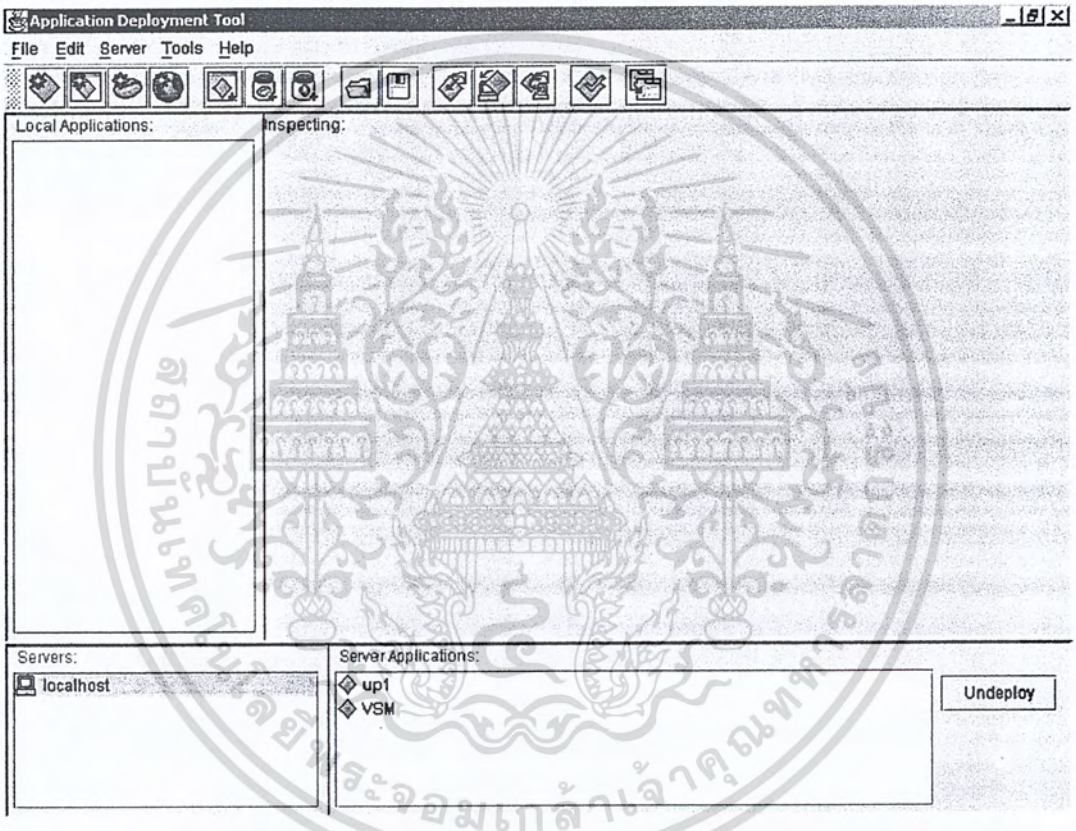
2 ทำการรันไฟล์ compilerEJB.bat

2. สร้าง J2EE แอพพลิเคชัน

คุณจะไม่สามารถดีพล็อยเอ็นเตอร์ไพส์บีนได้โดยตรงใน J2EE Server ต้องทำการ add บีนเข้าไปไว้ใน J2EE แอพพลิเคชัน จึงจะสามารถดีพล็อยได้ โดยเราจะทำการสร้างแอพพลิเคชันใหม่ที่ชื่อว่า ConverterApp และเก็บไฟล์นี้ไว้ใน ConverterApp.ear

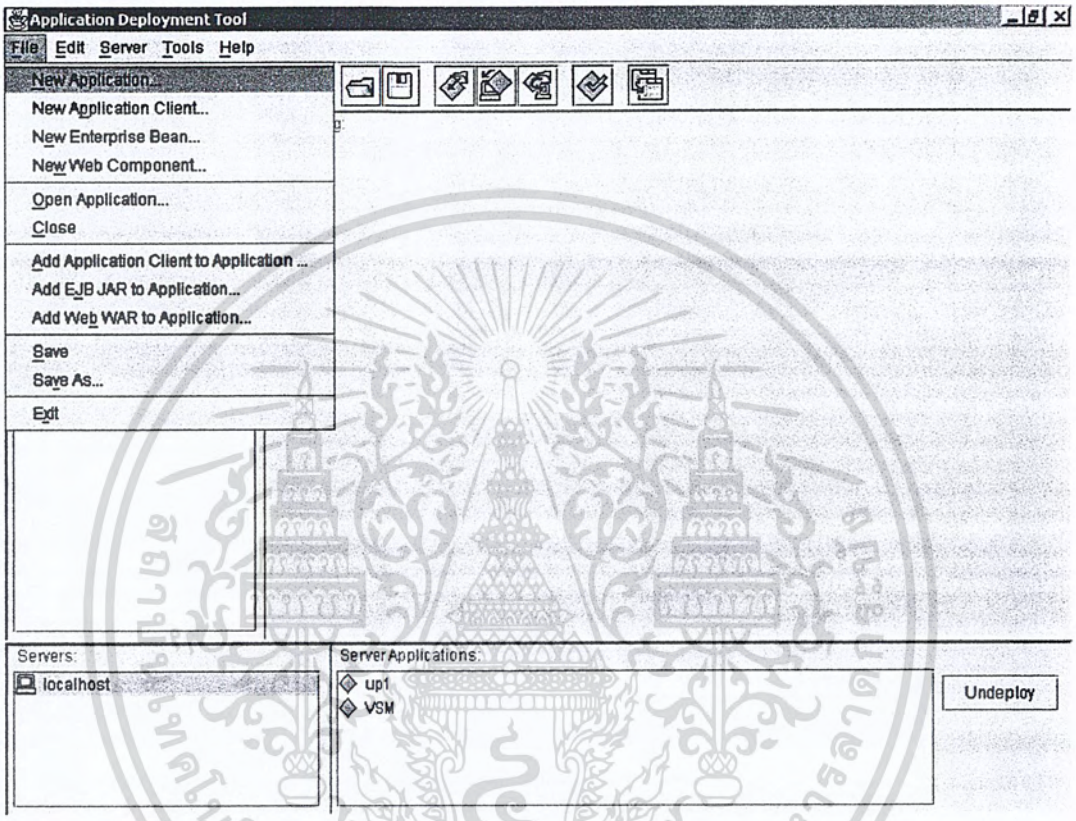
2.1 ทำการเปิดหน้าต่าง command prompt และ start J2EE server โดยพิมพ์ j2ee –verbose

2.2 เปิด command prompt ขึ้นมาอีกหน้าต่างหนึ่งและพิมพ์ deploytool จะปรากฏหน้าจอดังนี้

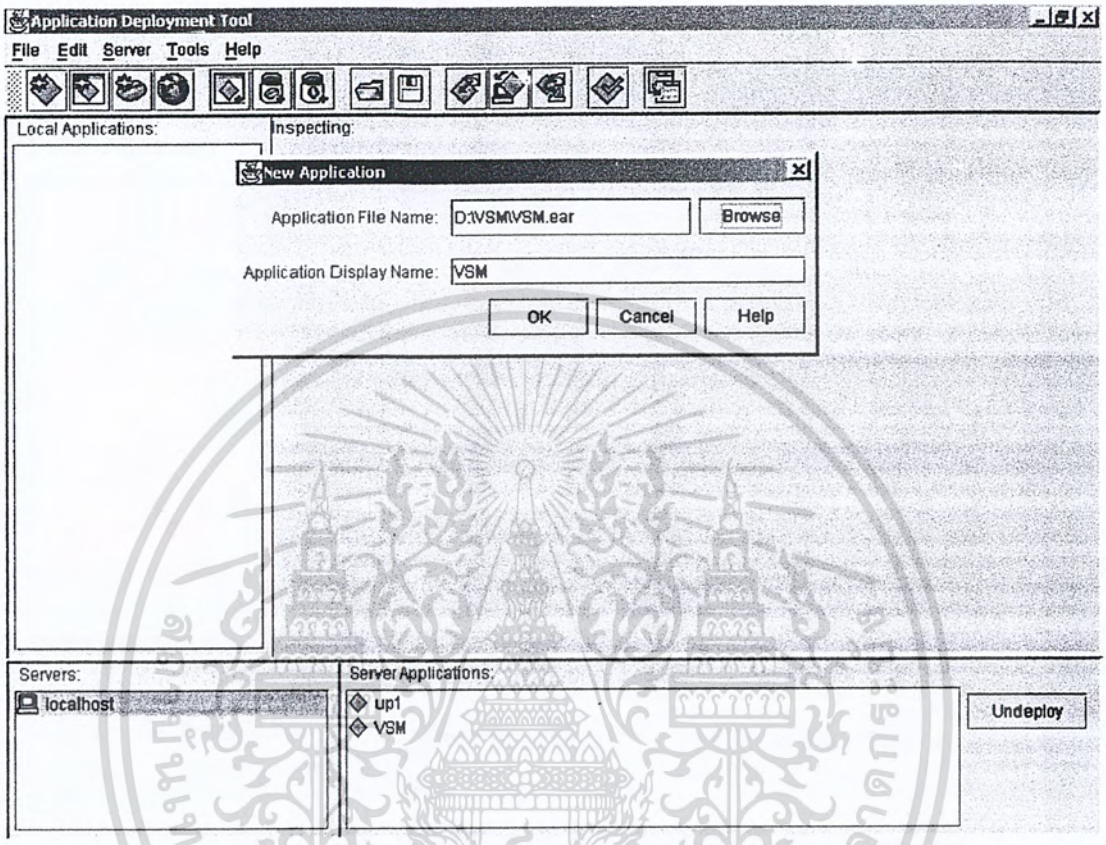


2.3 การเริ่มสร้างแอพพลิเคชั่น

1. ไปที่ File>New Application

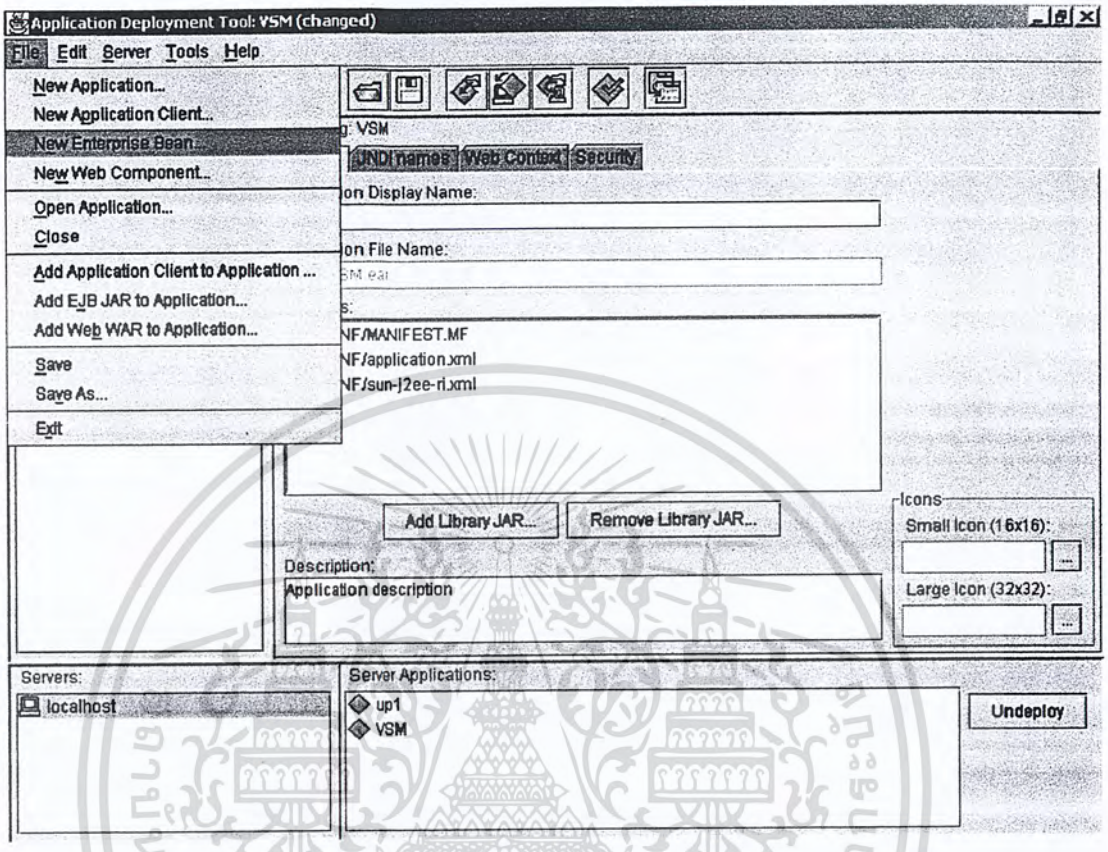


2. กำหนดที่อยู่และตั้งชื่อแอพพลิเคชั่น ที่ต้องการเอาลงเซิฟเวอร์ (ในที่นี้ให้ใส่ ConverterApp.ear)

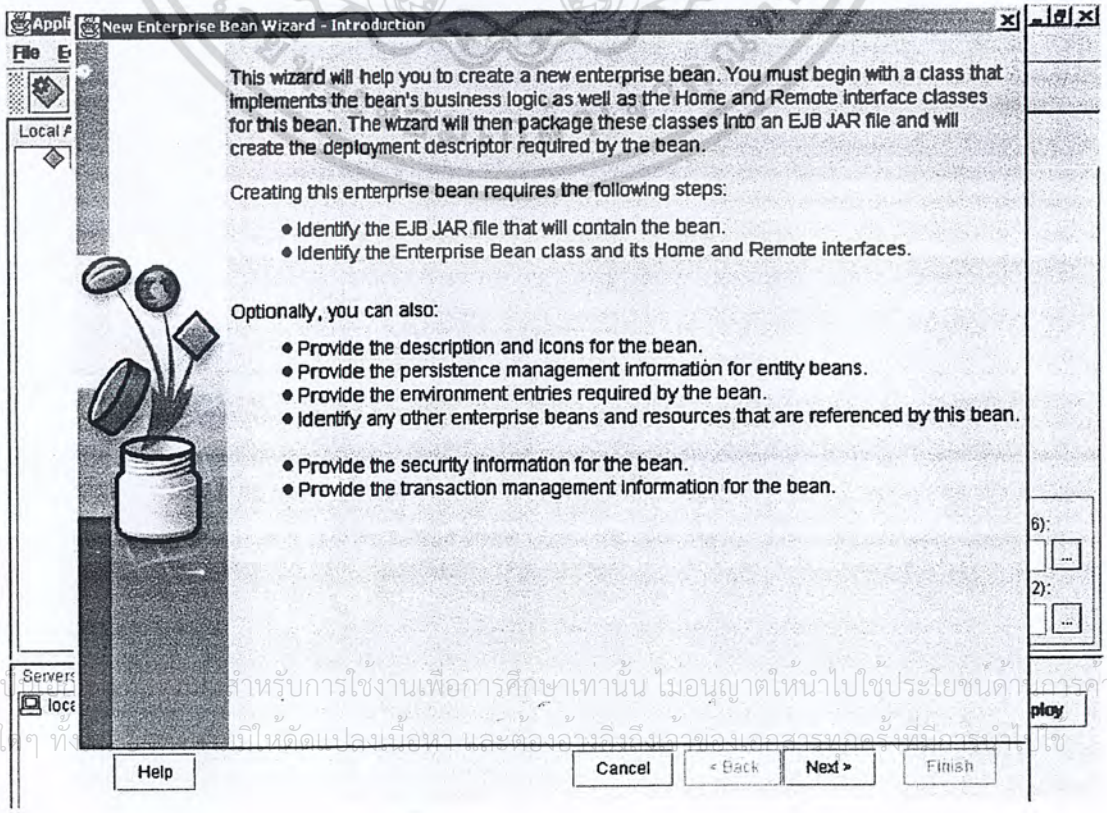


3. แพ็คเก็ตเอ็นเตอร์ไพร์จาวาบีน

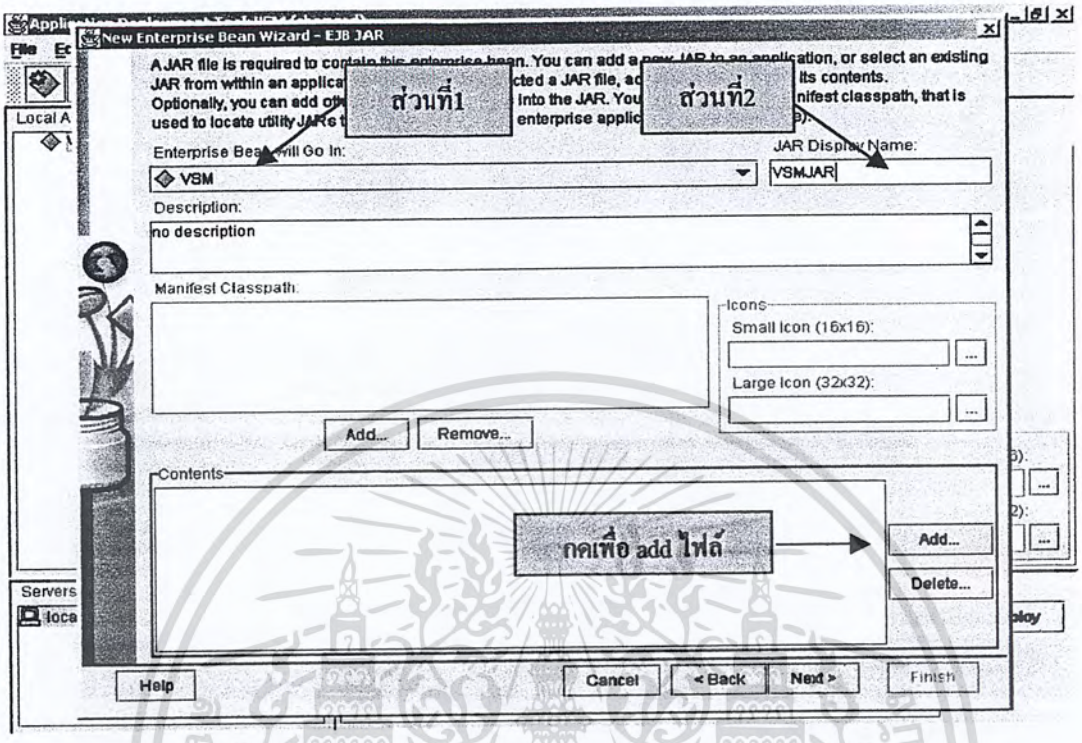
3.1 ไปที่ File>New Enterprise bean



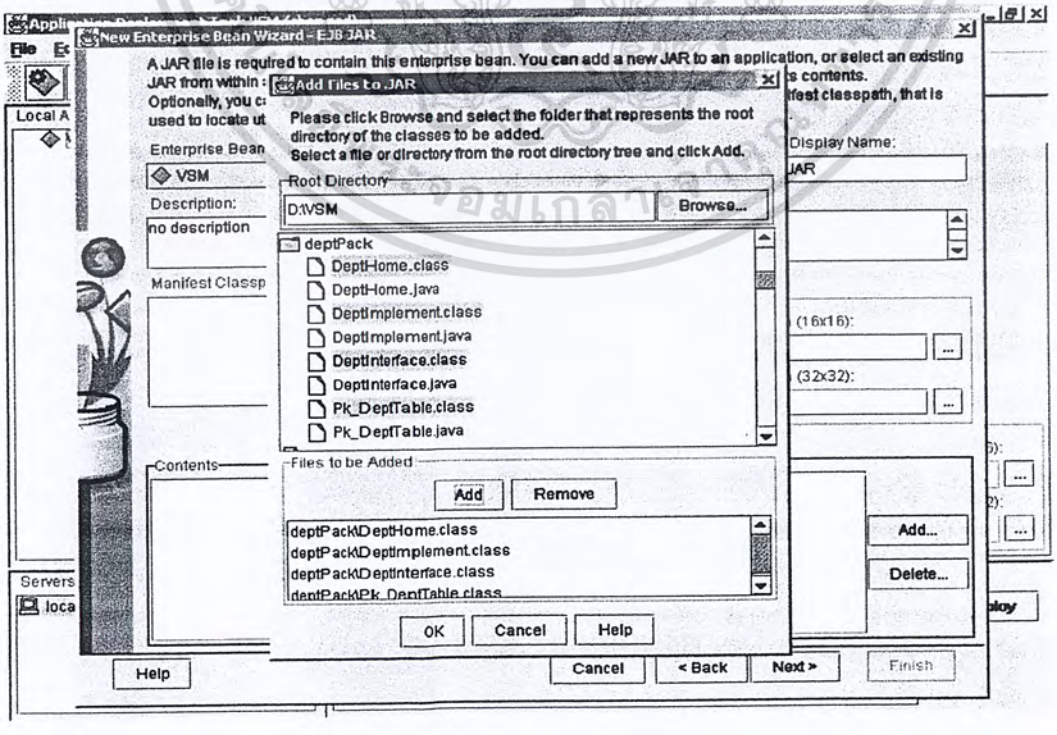
3.2 หน้าแรกจะแสดงรายละเอียดในการสร้างคอมโพเนนท์ เมื่ออ่านเสร็จกดปุ่ม OK



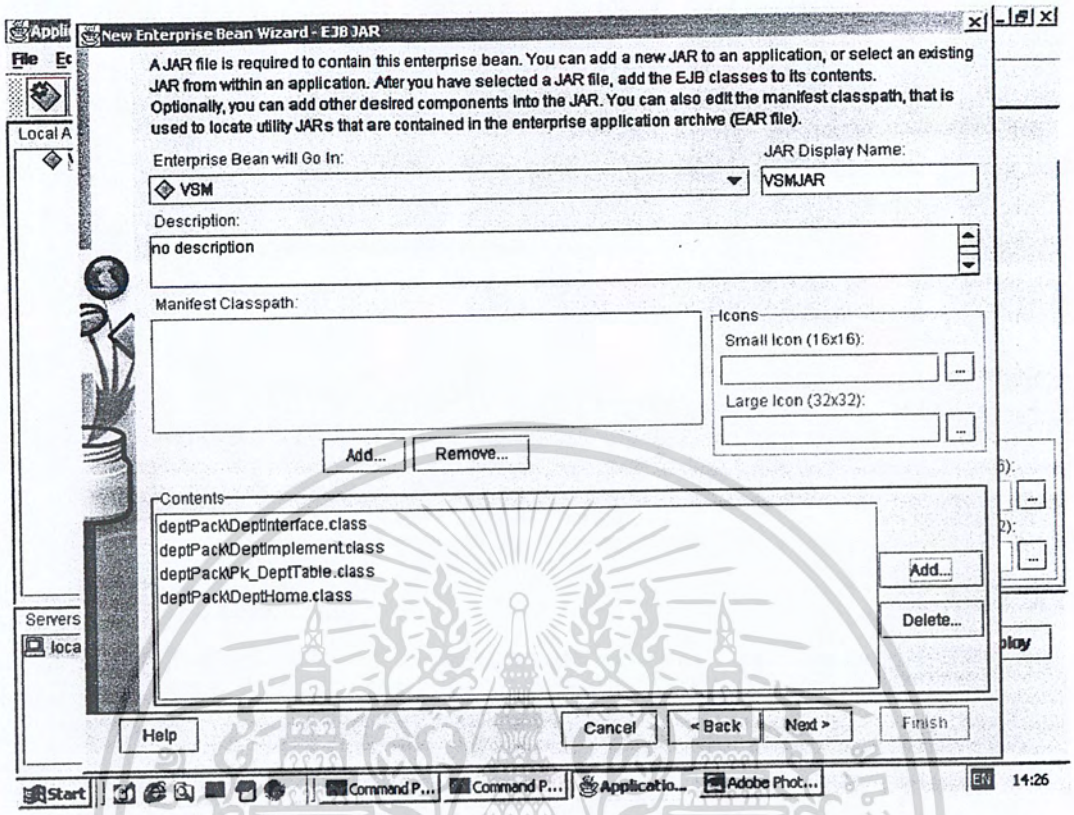
3.3 กำหนดแอพพลิเคชั่นที่จะให้คอมโพเนนท์เข้าไปอยู่ (ส่วนที่1) กำหนดชื่อจาร์ไฟล์ (ส่วนที่2) เมื่อกำหนดชื่อเรียบร้อยแล้ว ต่อไปต้องทำการ add ด้านล่าง



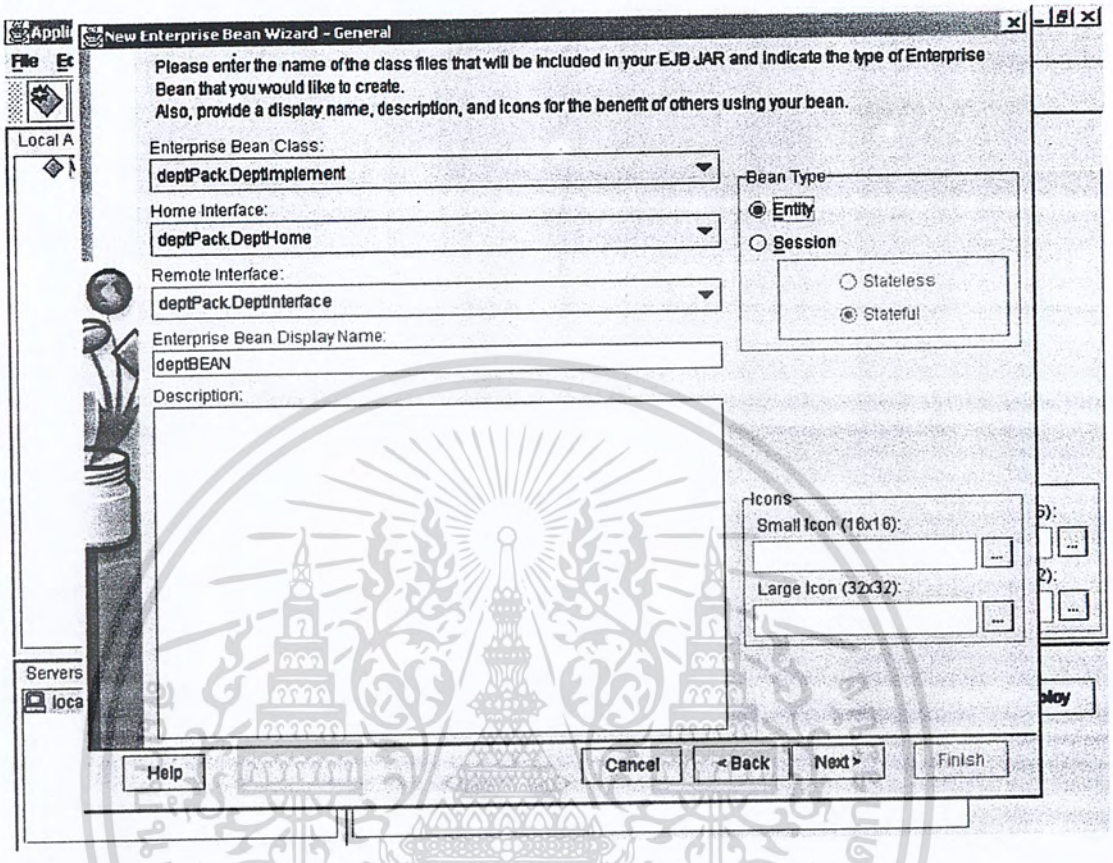
3.4 เลือกไฟล์ที่จะ add โดยถ้าเป็นแพ็คเก็ตอยู่ด้วย (ในที่นี้ให้เลือกไฟล์ Converter.class,ConverterEJB.class,ConverterHome.class) เลือกเสร็จกด add และ ok



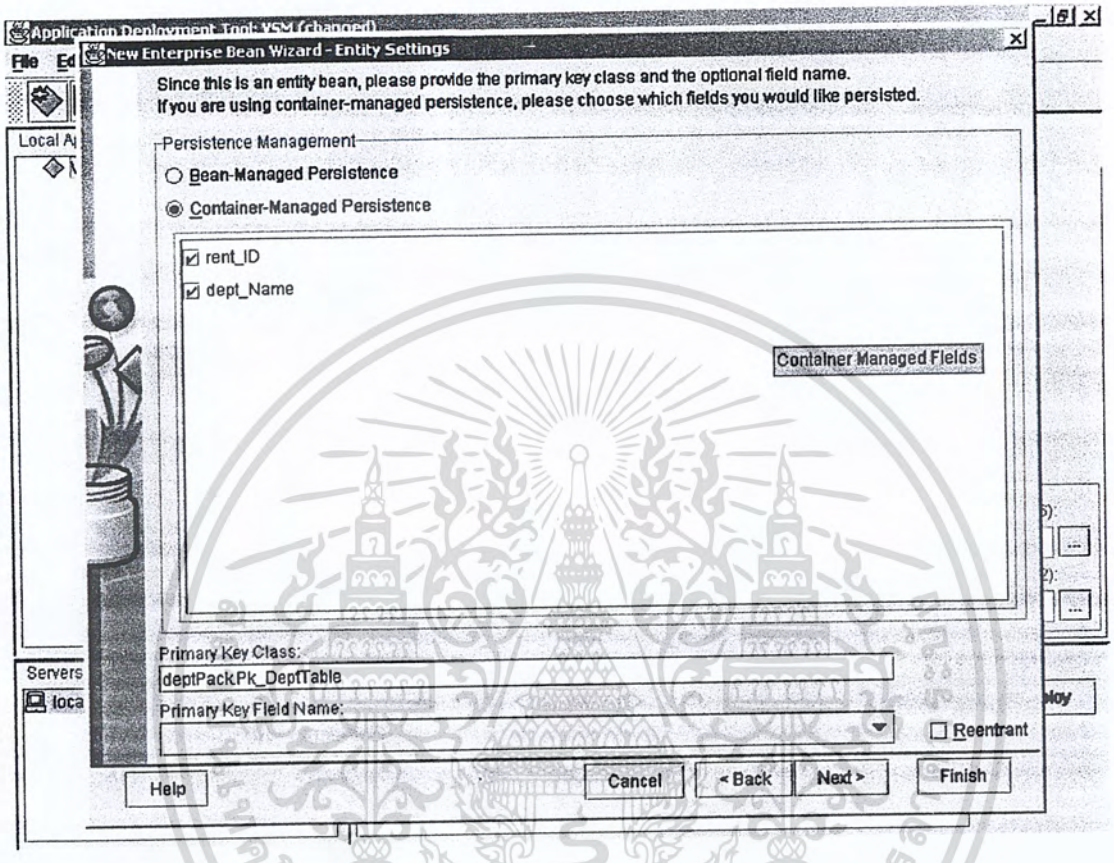
3.5 เมื่อ add เสร็จแล้วจะขึ้นรายชื่อไฟล์ในจาร์ ในส่วนคอนเทนท์กด Next



3.6 เลือกไฟล์ที่เป็นเอ็นเตอร์ไพรส์บีน โฮมอินเตอร์เฟส รีโมทอินเตอร์เฟส กำหนดชื่อเอ็นเตอร์ไพรส์บีน และกำหนดชนิดของเอ็นเตอร์ไพรส์บีน เป็น เอ็นติตี้



3.7 กำหนดชนิดของเอ็นติตี้บีนว่าเป็นแบบ bean management หรือ container management ในโครงงานนี้จะใช้แบบ container management จึงเลือกแบบที่สอง เลือกฟิวส์ที่เกี่ยวข้องกับฐานข้อมูล และกำหนดคลาสที่เป็นไพรมารี่คีย์



3.8 คลิกปุ่ม finish

4. **ดีพลอย J2EE แอพพลิเคชัน**

ตอนนี้ J2EE แอพพลิเคชันนี้พร้อมที่จะดีพล็อยแล้ว โดยการดีพล็อยมีขั้นตอนดังนี้

การใส่ JNDI Name ของเอ็นเตอร์ไพส์บีน

4.1 ในหน้าต่าง Application Deploy Tool ทำการเลือก ConverterApp ใน tree view

4.2 เลือก JNDI Name tab

4.3 ในช่อง JNDI Name ใส่ My Converter และกดปุ่ม Return

การดีพล็อยแอพพลิเคชัน

4.4 ใน เมนู Tools ให้เลือก Deploy Application

4.5 ทำการตรวจสอบ Target Server โดยการใส่ชื่อของโฮสต์ที่จะรัน J2EE Server หรือ ใสเป็น localhost

4.6 เลือก checkbox labelled "Return Client Jar"

4.7 คลิก Next

4.8 ใน dialog box ทำการตรวจสอบ JNDI Name ของ ConverterBean ว่าคือ MyConverter

4.9 คลิก Next

4.10คลิก Finish

4.11จะปรากฏหน้าต่าง Deployment Progress dialog box ให้คลิก OK เมื่อดีพล็อยเสร็จ

5. **สร้างไคล์เอ็นท์**

ทำการเขียนโค้ดไคล์เอ็นท์ประกอบด้วย 2 ส่วนคือ

5.1 การเขียนโค้ดไคล์เอ็นท์

โค้ดของไคล์เอ็นท์ในที่นี้จะให้ชื่อว่า ConverterClient.java ประกอบด้วยพื้นฐาน 3 ส่วนคือ

5.1.1 Locate Home Interface คือ การสร้างการติดต่อโดยเริ่มจากการอินสแตนทิเอทอ๊อบเจ็ค ของโฮมอินเตอร์เฟส

5.1.2 Create an Enterprise Bean Instance คือ การสร้างอินสแตนท์ของเอ็นเตอร์ไพส์บีน

5.1.3 Invoke a Business Method คือ การเรียกบีซิเนสเมทอดที่ประกาศในรีโมทอินเตอร์เฟส

โดยแต่ละขั้นตอนมีรายละเอียดดังนี้

5.1.1 Locate Home Interface คือการสร้างการติดต่อโดยเริ่มจากการอินสแตนทิเอทอ๊อบเจ็ค ของโฮมอินเตอร์เฟส

ก่อนที่ ConverterClient จะทำการเรียกเมทอด create() ที่อยู่ใน ConverterHome จะต้องทำการอินสแตนทิเอทอ็อบเจ็คที่มี type เป็น ConverterHome ก่อนโดยขั้นตอนการได้อ็อบเจ็คของ ConvertHome มี 3 ขั้นตอนดังนี้

สร้าง JNDI naming context

```
Context initial = new InitialContext();
```

2. เมื่อได้ naming context แล้วนำ JNDI naming context ไปทำการ lookup ไปยัง JNDI name

```
java.lang.Object objref = initial.lookup("MyConverter");
```

2. Narrow เพื่ออ้างอิงไปยังอ็อบเจ็คของ ConverterHome

```
ConverterHome home = (ConverterHome) PortabeRemoteObject.narrow
(objref,ConverterHome.class);
```

5.1.2 Create an Enterprise Bean Instance คือ การสร้างอินสแตนท์ของเอ็นเตอร์ไพส์บีน

การสร้างอินสแตนท์ของเอ็นเตอร์ไพส์บีนทำได้โดยการที่ไคล์เอ็นท์ทำการเรียกเมทอด create() บนอ็อบเจ็คConverterHome โดยเมทอด create จะทำการรีเทอร์นอ็อบเจ็คที่มี type เป็น Converter ภายในอินเตอร์เฟส Converter จะประกาศชื่อของบิซิเนสเมทอดที่ต้องการให้ไคล์เอ็นท์เรียกโดยมีเมทอดจริงกำหนดไว้ใน ConverterEJB เมื่อไคล์เอ็นท์ทำการเรียกเมทอด create() คอนเทนเนอร์จะอินสแตนทิเอท ConverterEJB ขึ้นมาและทำการเรียกเมทอด ConverterEJB.ejbCreate()

```
Converter currencyConverter = home.create();
```

5.1.3 Invoke a Business Method คือ การเรียกบีซิเนสเมทอดที่ประกาศในรีโมทอินเตอร์เฟส

การเรียกบิซิเนสเมทอดทำได้โดยเรียกจากอ็อบเจ็ค Converter โดยคอนเทนเนอร์จะทำการเรียกเมทอดใน ConverterEJB ซึ่งรันอยู่บนเซิฟเวอร์ เช่น ไคล์เอ็นท์ทำการเรียกเมทอด dollarToYen() ดังนี้

```
double amount = currencyConverter.dollarToYen(100.00);
```

5.2 การคอมไพล์ไคล์เอ็นท์

การคอมไพล์ไคล์เอ็นท์ทำได้โดย

1. การสร้างไฟล์ compileClient.bat ดังนี้

set J2EE_HOME = <installation-location>

set CPATH = .;%J2EE_HOME%\lib\j2ee.jar

javac – classpath %CPATH% ConverterClient.java

หมายเหตุ <installation-location> **คือ ไดเรกทอรีที่ติดตั้ง** J2EE SDK

2. **การรันไฟล์** compileClient.bat

6. **รันไคล์เอ็นท์**

การรันไคล์เอ็นท์ต้องมีไฟล์ ConverterAppClient.jar โดยไฟล์ jar นี้ภายในจะใส่ stub classes **ที่อนุญาตให้ไคล์เอ็นท์ทำการติดต่อกับอินสแตนท์ของเอนเตอร์ไพส์บีนที่รันอยู่ในคอนเทนเนอร์ดังนี้**

1. การสร้างไฟล์ testClient.bat ดังนี้

set J2EE_HOME = <installation-location>

set CPATH = .;%J2EE_HOME%\lib\j2ee.jar;ConverterAppClient.jar

javac – classpath %CPATH% ConverterClient

หมายเหตุ <installation-location> **คือ ไดเรกทอรีที่ติดตั้ง** J2EE SDK

2. **การรันไฟล์** testClient.bat

บรรณานุกรม

ชื่อผู้แต่ง ชื่อหนังสือ สำนักพิมพ์ ปีที่พิมพ์ หน้าที่

[1] Tsichritzis D.C. and Klug A. (1978) : "*The ANSI/X3/SPARC DBMS Frame Work : Report of the Study Group on Data Base Management Systems*", "Information System" ,3 1978.

[1] Ed Roman (1999) :"*Mastering Enterprise JavaBeans and the Java 2 Platform Enterprise Edition*" :Wiley Computer Publishing

[3] Sun MicroSystems,INC (2000) : "*The Java 2 Enterprise Edition Developing Guide*"

[4] David Harel (1998) :"*Real-Time UML Developing Efficient Objects for Embedded System"* : ADDISON-WESLEY

[5] Joseph O'Neil Edited by Herb Schildt (1998) :"*JavaBeans™ Programming from the Ground Up*" : McGraw-Hill

[5] http://www.java.sun.com

[6] http://www.javacentrix.com

[7] http://www.oracle.com

[9] http://www2.theserverside.com

[12] www.bea.com