การพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์โดยใช้ EJB

WEB SERVICES & COMPONENT SOFTWARE DEVALOPMENT USING EJB

นายปิติ ปรัชญารุ่งโรจน์
นางสาวพนมพร สุขพึ่งธรรม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัญฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

การพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์โดยใช้ EJB

WEB SERVICES & COMPONENT SOFTWARE DEVALOPMENT USING EJB

โดย

นายปิติ ปรัชญารุ่งโรจน์

นางสาวพนมพร สุขพึ่งธรรม

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ดร.ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัญฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์โดยใช้ EJB

Web Services & Component Software Development Using J2EE

คณะผู้จัดทำ นายปิติ ปรัชญารุ่งโรจน์ รหัส 43010268

นางสาวพนมพร สุขพึ่งธรรม รหัส 43010281

..... อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

..... อาจารย์ที่ปรึกษา

(ดร.ชุติเมษฎ์ ศรีนิลทา)

การพัฒนาซอร์ฟแวร์เชิงคอมโพเนนต์โดยใช้ EJB

นายปิติ ปรัชญารุ่งโรจน์ 43010268
นางสาวพนมพร สุขพึ่งธรรม 43010281
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร.ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ (Component-based Software Development) เป็นกระบวนการในการสร้างซอฟต์แวร์ โดยนำคอมโพเน็นต์ที่มีอยู่มาใช้ (reuse) เพื่อช่วยลดเวลาและมูลค่าในการพัฒนา

เนื่องจาก ในปัจจุบันซอฟต์แวร์ระบบคอมพิวเตอร์สำหรับองค์กรขนาดใหญ่ มีความต้องการมากขึ้น ซึ่งต้องมีประสิทธิภาพสูง มีความน่าเชื่อถือ มีความปลอดภัย ต้องดูแลไม่ยาก ดังนั้นโครงงานนี้ใช้สถาปัตยกรรมของซันไมโครซิสเต็มส์ที่มี Enterprise JavaBeans (EJB) เป็นสถาปัตยกรรมคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์ (server-side component architecture) จะใช้ภาษา Java ในการพัฒนา จึงสามารถทำงานได้ทุกแพล็ตฟอร์ม (portability) พร้อมทั้งยังมี API อื่นที่ช่วยให้การพัฒนาเอ็นเตอร์ไพรส์แอพพลิเคชัน (enterprise application) ทำได้ง่าย ได้แก่ JSP, Servlet, JMS, JTA และ JDBC เป็นต้น

แนวคิดของเว็บเซอร์วิสทำให้สามารถทำงานข้ามแฟลตฟอร์มได้

วิทยานิพนธ์ฉบับนี้ศึกษาถึงวิธีในการพัฒนาเว็บเซอร์วิสเอนเตอร์ไพรส์แอพพลิเคชัน โดยใช้จาวาเทคโนโลยีในการพัฒนา มีการใช้งานข้ามแฟลตฟอร์มร่วมกันระหว่าง .NET เฟรมเวิร์ค

Web Services & Component Software Development Using J2EE

Piti	Pratyaroongrot	43010268
Panomporn	Sookpuengtham	43010281
Dr.Worawat	Limpoka	Advisor
Dr.Chutimate	Srinilta	Advisor

Academic Year 2003

ABSTRACT

Component-based software development is a software process that reused software components to reduce time and cost of development.

Nowadays, computer software for large Enterprise is required much more than in the past, this software should have more efficiency, more reliability, more security and easier to maintain. Home Services, this project use SUN Microsystems's architecture, which have server-side component architecture, named Enterprise JavaBeans (EJB). JAVA language is used, so this software can run anywhere, any platform (portability). Included other API to implement this project (enterprise application) easier such as JSP, Servlet, JMS, JTA, JDBC, etc.

Themes of Web Services can support the portability function.

This thesis present Web Services Enterprise Application Development component-based software development methodology using Java technology. Working across platform with .NET Framework.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลที่มีส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์วรวัฒน์ ลิ้มโภคา และอาจารย์ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา ที่ให้ความเอาใจใส่ ช่วยแนะนำ สั่งสอน และช่วยเหลือในเรื่องต่างๆ ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก และขอขอบคุณพี่ ประกิจ อิงคกิตติ ที่คอยให้คำปรึกษา และขอขอบคุณ เพื่อน ๆทุกคน ที่คอยช่วยเหลือ ให้คำปรึกษา และทำงานด้วยกันมา

สุดท้ายต้องขอขอบพระคุณ บิดา มารดา และญาติมิตร ที่อยู่เบื้องหลังความสำเร็จทั้งหมดนี้ จึงขอกราบขอบพระคุณมา ณ ที่นี้

นายปิติ ปรัชญารุ่งโรจน์
นางสาวพนมพร สุขพึ่งธรรม

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

การพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์ (Component-Based Software Development หรือ CBSD) มีข้อดี คือ สามารถนำคอมโพเนนต์ที่มีอยู่มาใช้ (reuse) เพื่อลดเวลาและค่าใช้จ่ายในการพัฒนา, น่าเชื่อถือ (Reliable) แต่สถาปัตยกรรมนี้มีข้อจำกัดที่ไม่สามารถนำคอมโพเนนต์ที่มีอยู่มาใช้ใหม่ข้ามสถาปัตยกรรม 2 สถาปัตยกรรมได้ คือ สถาปัตยกรรมของบริษัทไมโครซอฟท์ ซึ่งก็คือ COM+ และสถาปัตยกรรมของบริษัท ซันไมโครซิสเต็มส์ คือ EJB

แต่ภายหลังการเกิดของ XML , SOAP และแนวคิดของเว็ปเซอร์วิส ทำให้สามารถทำงานข้ามสถาปัตยกรรมได้โดยผ่านทาง SOAP ซึ่งเป็นโปรโตคอล (Protocol) ที่ใช้สำหรับ Remote Procedure Call (RPC) ผ่าน HTTP โปรโตคอล โดยรายละเอียดต่างๆ เกี่ยวกับทฤษฎีและตัวอย่างการสร้างเว็บเซอร์วิสโดยใช้จาวาเทคโนโลยีในการพัฒนาจะถูกนำเสนอในปริญญานิพนธ์การพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์โดยใช้ EJB1 ซึ่งปริญญานิพนธ์ฉบับนี้จะนำเสนอเทคโนโลยีของบริษัทซันไมโครซิสเต็มส์คือ Java 2 Platform Micro Edition (J2ME) เป็นแพล็ตฟอร์ม (Platform) ที่ออกแบบมาเพื่อตอบสนองความต้องการของผู้ใช้อุปกรณ์อิเล็กทรอนิกส์และอุปกรณ์ฝังตัวที่มีขนาดของหน่วยความจำ , ความสามารถในการประมวลผลที่จำกัด

และการพัฒนาสถาปัตยกรรมคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์ (Server-side component architecture) ยังช่วยให้การพัฒนาแอพพลิเคชันระดับเอ็นเตอร์ไพรส์ (Enterprise application) ทำได้ง่าย นอกจากนั้นยังมีความน่าเชื่อถือ (Reliable) , มีระบบรักษาความปลอดภัย (Security) ที่ถูกต้องมากขึ้น ทั้งยังช่วยลดความซับซ้อนเนื่องจากโครงสร้างของระบบ ในปัจจุบันได้มีบริษัทต่างๆ ได้พัฒนาสถาปัตยกรรมที่ทำงานบนฝั่งเซิร์ฟเวอร์ (Server-side architecture) 2 สถาปัตยกรรมที่ได้รับการยอมรับมากที่สุดคือ

- Microsoft's Distributed interNet Applications Architecture (DNA) เป็นสถาปัตยกรรมของบริษัทไมโครซอฟท์ (Microsoft) ที่นำเสนอสถาปัตยกรรมคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์คือ COM+ (Component Object Model plus)
- Sun Microsystem's Java 2 Platform, Enterprise Edition (J2EE) เป็นสถาปัตยกรรมของ บริษัท ซันไมโครซิสเต็มส์ (Sun Microsystems) ที่นำเสนอสถาปัตยกรรมคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์คือ EJB (Enterprise JavaBeans)

สถาปัตยกรรมของทั้ง 2 บริษัทมีความแตกต่างที่น่าสนใจคือ J2EE นำเสนอความสามารถในการทำงานข้ามแพล็ตฟอร์ม (portability) และ DNA นำเสนอความสามารถในการทำงานข้ามภาษา (interoperability) เนื่องจากทางบริษัทซันไมโครซิสเต็มส์ได้ออกแบบภาษาจาวา (Java) ให้สามารถทำงาน

ข้ามแพลตฟอร์ม การพัฒนาคอมโพเน็นต์ด้วยภาษาจาวาทำให้คอมโพเน็นต์มีความสามารถนี้อยู่ด้วย ในขณะที่ทางบริษัทไมโครซอฟท์ได้สนับสนุนการสร้างคอมโพเน็นต์จากภาษาใดก็ได้เช่น Visual Basic และ Visual C++ คอมโพเน็นต์ที่สร้างด้วย Visual Basic สามารถนำมาใช้กับ Visual C++ ได้ ความแตกต่างอีกอย่างหนึ่งคือ J2EE เป็น specification เท่านั้น แอพพลิเคชันเซิร์ฟเวอร์ (Application Server) จะถูกพัฒนาโดยผู้ผลิตรายอื่น เช่น BEA WebLogic Server , IBM WebSphere , Oracle 9i Application Server, SilverStream Application Server , iPlanet , Sybase Enterprise Application Server , Borland AppServer เป็นต้น

ทั้ง 2 สถาปัตยกรรมนี้มีข้อจำกัดที่ไม่สามารถนำกลับมาใช้ใหม่ (reuse) ข้ามสถาปัตยกรรมได้ แต่ภายหลังการเกิดของ XML , SOAP และแนวคิดของเว็บเซอร์วิส (Web Service) ทำให้สามารถทำงานร่วมกันได้ผ่านทาง SOAP ซึ่งเป็นโพรโตคอลที่ใช้สำหรับ Remote Procedure Call (RPC) ผ่าน HTTP โพรโตคอล

1.1 วัตถุประสงค์ของโครงงาน

ภายในโครงงานนี้ได้ทำการศึกษาถึงทฤษฎี การสร้าง และการนำคอมโพเนนต์ คือ EJB ไปใช้งานในการสร้างเว็บเซอร์วิสโดยมีจุดประสงค์ดังนี้

1. ศึกษาทฤษฎีพื้นฐานของคอมโพเนนต์
2. เพื่อศึกษาทำความเข้าใจ Enterprise JavaBean (EJB) ในจาวาเทคโนโลยี
3. ศึกษาเรื่องเว็บเซอร์วิส และเทคโนโลยีที่เกี่ยวข้อง ได้แก่ SOAP, XML, WSDL, UDDI เป็นต้น
4. ศึกษาการสร้างเว็บเซอร์วิส โดยใช้จาวาเทคโนโลยีลงลึกในเทคโนโลยีของเว็บเซอร์วิส เช่น ทรานแซกชั่น , การจัดการเสตท , ความปลอดภัยในการเรียกใช้เว็บเซอร์วิส
5. ทำการขยายการรองรับการใช้งานของระบบ (Scalability) โดยใช้เทคโนโลยี JMS
6. ศึกษาการเรียกใช้เว็บเซอร์วิสข้ามค่าย รวมทั้งเปรียบเทียบเทคโนโลยีของทั้งสองค่าย .NET และจาวา

1.2 หลักการและเหตุผล

เนื่องจากในปัจจุบันการค้าขายผ่านทางระบบ E-commerce เป็นธุรกิจมีการเติบโตค่อนข้างสูง แต่การจัดการการซื้อขายที่มีอยู่ในปัจจุบันนี้ ยังไม่สามารถสั่งซื้อสินค้าข้ามเวบได้ ต้องสั่งซื้อแค่ภายในเวบที่ขายสินค้าชนิดนั้นเท่านั้น ดังนั้น ซอฟต์แวร์ที่พัฒนาขึ้นมานี้ จึงสามารถเรียกการซื้อขายสินค้าข้ามเวบจากเวบอื่นได้ โดยใช้เทคโนโลยี เวบเซอร์วิสเข้ามาช่วย เพื่อที่จะอำนวยความสะดวกและประหยัดเวลาให้กับผู้ใช้งานบริการนั้น ซึ่งแอพลิเคชั่นจะเปิดเซอร์วิสของตัวเองให้ผู้อื่นมาเรียกใช้งาน

1.3 ขอบเขตของโครงงาน

1. ผู้ใช้สามารถเข้ามาเลือกชมอุปกรณ์ตกแต่งบ้านได้ ทั้งในกรณีที่มี user account หรือไม่มีก็ได้
2. ผู้ใช้สามารถสมัครสมาชิกกับทางระบบ เพื่อในกรณีที่ต้องการทำ กิจกรรม กับระบบ
3. การเลือกซื้ออุปกรณ์ตกแต่งบ้าน ต้องเป็นสมาชิกของระบบเท่านั้น
4. การจ่ายเงินผ่านทางระบบ จะมีรูปแบบให้เลือกคือ credit card , จ่ายผ่อน หรือ จ่ายเงินสด
5. furnitureและอุปกรณ์ต่างๆที่ระบบขาย จะถูกจัดแบ่งเป็นหมวดหมู่ ตามรูปแบบการใช้งานภายในบ้าน
6. มี Tips&Tricks เกี่ยวกับการดูแลบ้าน และ เทคนิคการตกแต่งและจัดบ้าน เพื่อเป็นความรู้กับให้ผู้ใช้ระบบซึ่ง จะ update ตลอด
7. มีการติดต่อกับธนาคารต่างๆ เพื่อดูอัตราดอกเบี้ยเงินกู้ และ แสดงอัตราเงินผ่อนให้ลูกค้าดูได้

1.4 วิธีการดำเนินงานของโครงงาน

1. ศึกษาทฤษฎีพื้นฐานของการพัฒนาเชิงคอมโพเนนต์ , J2EE ซึ่งเป็นเทคโนโลยีสถาปัตยกรรมบนฝั่งเซิร์ฟเวอร์ที่ใช้ภาษาจาวาในการพัฒนา ประกอบด้วย EJB , Servlet , JSP และ JDBC เป็นต้น
2. ศึกษาทฤษฎีของ XML และ SOAP และส่วนประกอบในเว็บเซอร์วิส
3. ศึกษาการพัฒนาคอมโพเน็นต์และเว็บเซอร์วิสในจาวาเทคโนโลยี
4. ออกแบบระบบแอพพลิเคชั่นในลักษณะงานเชิงคอมโพเน็นต์
5. ศึกษาการทำงานของ JMS ที่มีอยู่ในจาวาเทคโนโลยี
6. ศึกษาการปรับแต่งเพื่อให้เว็บเซอร์วิสมีความปลอดภัยในการสื่อสาร
7. ศึกษาการใช้งานร่วมกันรวมทั้งเปรียบเทียบเทคโนโลยีจากทั้งสองค่าย .NET และ จาวา

โครงงานนี้เป็นโครงงาน เรื่องการพัฒนาเชิงคอมโพเนนต์ โดยทำศึกษาร่วมกันทั้ง 3 กลุ่ม ประกอบด้วย กลุ่มที่พัฒนาบนแพลตฟอร์ม .NET 2 กลุ่ม และ EJB 1 กลุ่ม มีพื้นฐานอยู่บนการพัฒนาเชิงคอมโพเนนต์ ประกอบกับ เทคโนโลยี Web services

เนื่องจากเป็นการพัฒนาโครงงานร่วมกัน โดยต้องอาศัยพื้นฐานความรู้เดียวกัน ดังนั้นเนื้อหาและทฤษฎีบางส่วนจะมีความเหมือนกัน การแบ่งหัวข้อเพื่อแยกกันไปศึกษา จะทำให้เสียเวลา เพราะทุกกลุ่มก็ต้องศึกษาเนื้อหาเรื่องเดียวกัน ทำให้ผู้พัฒนาโครงงานได้แบ่งหัวข้อในการศึกษาให้แต่ละกลุ่มรับผิดชอบเพื่อนำความรู้ที่ได้มาใช้ร่วมกัน และ ถ่ายทอดให้กลุ่มที่เหลือได้เข้าใจตรงกัน เพื่อเป็นการประหยัดเวลา โดยหัวข้อที่แบ่ง เป็นดังนี้

กลุ่มการพัฒนาเชิงคอมโพเนนต์ ด้วย .NET กลุ่มที่ 1 (Land and Home) ศึกษาเรื่องดังต่อไปนี้

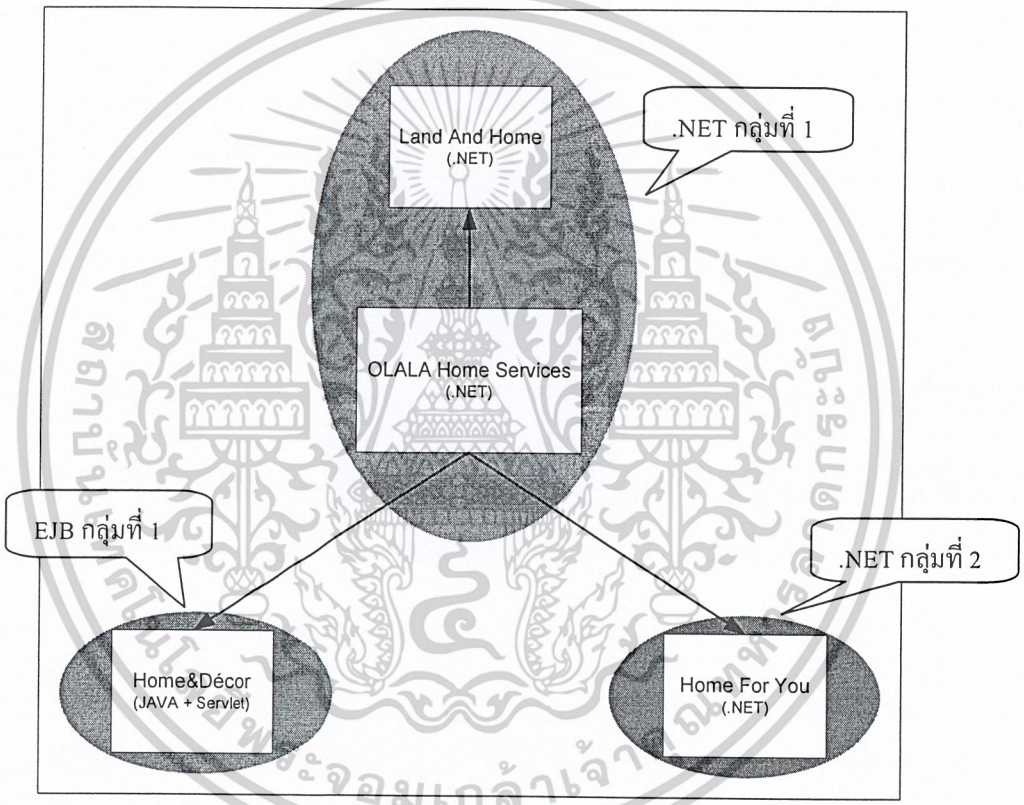
1. เว็บเซอร์วิส (Web Services)
2. ADO.NET
3. .NET Framework

กลุ่มการพัฒนาเชิงคอมโพเนนต์ ด้วย .NET กลุ่มที่ 2 (Home for you) ศึกษาเรื่องดังต่อไปนี้

1. ASP.NET
2. ASP.NET and Web services
3. VRML

กลุ่มการพัฒนาเชิงคอมโพเนนต์ ด้วย EJB กลุ่มที่ 1 (Home and Decor) ศึกษาเรื่องดังต่อไปนี้

1. ทฤษฎีคอมโพเน็นต์ (Component Theory)
2. สถาปัตยกรรม J2EE



บทที่ 2
ทฤษฎีคอมโพเน็นต์

คอมโพเน็นต์ คือ ส่วนย่อยของระบบที่ไม่ขึ้นกับส่วนอื่นๆ และ ซ่อนรายละเอียด การทำงานไว้ภายใน โดยมีส่วนติดต่อกับภายนอกเพื่อให้สามารถเรียกใช้งานได้ ความสามารถนี้ทำให้ คอมโพเน็นต์สามารถนำกลับมาใช้ใหม่ได้ (reuse)

2.1 การพัฒนาเชิงคอมโพเนนต์

การพัฒนาเชิงคอมโพเน็นต์นั้น มีความหมายกับคนแต่ละกลุ่มแตกต่างกันไปตามความต้องการ ซึ่งมีลักษณะเด่นบางอย่างที่นิยามความเป็นระบบคอมโพเน็นต์ดังต่อไปนี้

1. คอมโพเน็นต์มีลักษณะของออบเจ็กต์ คือ การซ่อนรายละเอียด (Encapsulated), โพลีมอร์ฟิค (polymorphic), การกำหนดหน้าที่ และการกำหนดอินเทอร์เฟซ
2. คอมโพเน็นต์ออกแบบภายใต้เฟรมเวิร์ก (Framework) ซึ่งได้สร้างข้อจำกัดบางอย่างไว้ เช่น ต้องไม่มีหลายเธรด, ไม่มีการติดต่อกับภายนอกโดยไม่ผ่านบริการของเฟรมเวิร์ก
3. คอมโพเน็นต์สามารถอยู่ได้ด้วยตัวเอง โดยไม่พึ่งพาคอมโพเน็นต์อื่นๆ ยกเว้นคอมโพเน็นต์ของเฟรมเวิร์กที่คอมโพเน็นต์ดังกล่าวใช้อยู่
4. ทุกคอมโพเน็นต์มีอินเทอร์เฟซสามัญที่แน่นอน (Fix and common) และอินเทอร์เฟซนี้เปลี่ยนแปลงไม่ได้5.คอมโพเน็นต์สามารถอธิบายตนเองได้ โดยอินเทอร์เฟซของคอมโพเน็นต์ จะต้องมีข้อมูลมากพอที่สามารถทำให้ไคลเอ็นต์ สามารถเข้าใจวิธีใช้คอมโพเน็นต์นั้นได้

2.1.1 เฟรมเวิร์ก

เฟรมเวิร์ก คือ สภาพแวดล้อมที่มีลักษณะดังต่อไปนี้

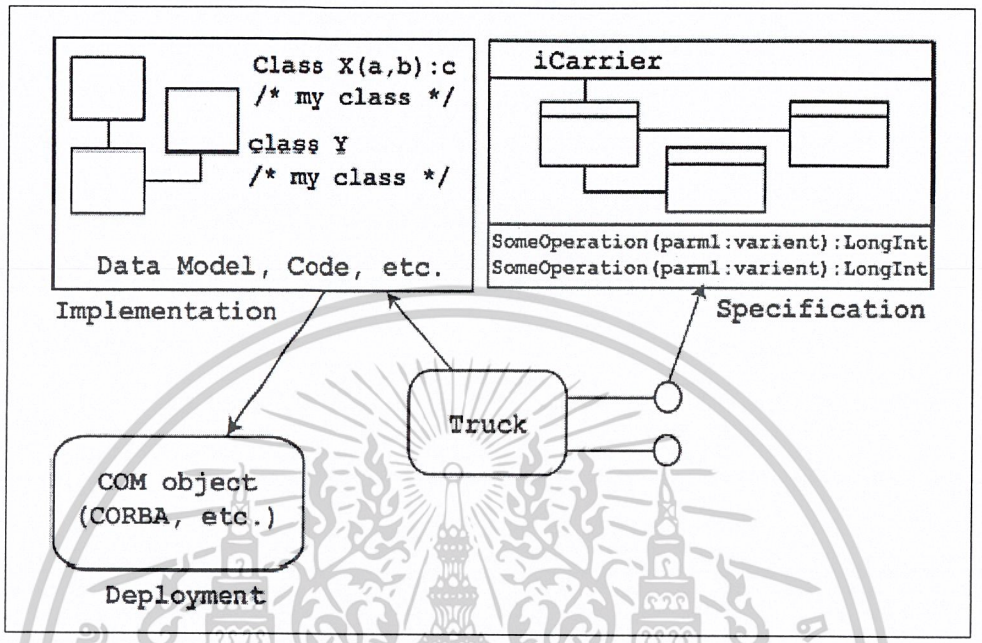
1. สร้างอินสแทนซ์ของคอมโพเน็นต์ตอนรันไทม์
2. ทำให้คอมโพเน็นต์ค้นพบคอมโพเน็นต์อื่นๆ ได้
3. ทำให้คอมโพเน็นต์ต่างๆ สามารถติดต่อกันได้
4. จัดหาบริการสามัญ เช่น เพอร์ซิสเทนซ์ (persistence), ทรานส์แอ็กชัน (transaction), ความไม่ขึ้นกับสถานที่, การรักษาความปลอดภัย, มอนิเตอริง (monitoring)

2.1.2 ส่วนประกอบของคอมโพเน็นต์

คอมโพเน็นต์จะประกอบด้วยส่วนสำคัญ 3 ส่วนดังนี้

1. อินเทอร์เฟซ (interface) - คอมโพเน็นต์จะถูกเรียกใช้งานผ่านทางอินเทอร์เฟซ
2. อิมพลีเมนต์เตชัน (implementation) - เป็นโค้ดที่กำหนดการทำงานของคอมโพเน็นต์

3. ดีพลอยเมนต์ (deployment) – เป็นเอ็กซิคิวต์ไฟล์จะใช้ในการทำให้คอมโพเน็นต์ทำงานได้ ทำหน้าที่จัดหารันไทม์เอ็นไวรอนเมนต์ ในการควบคุมการทำงานของคอมโพเน็นต์และจัดหาเซอร์วิสที่จำเป็น



รูปที่ 2-1 คอมโพเน็นต์และส่วนประกอบต่าง ๆ ของมัน

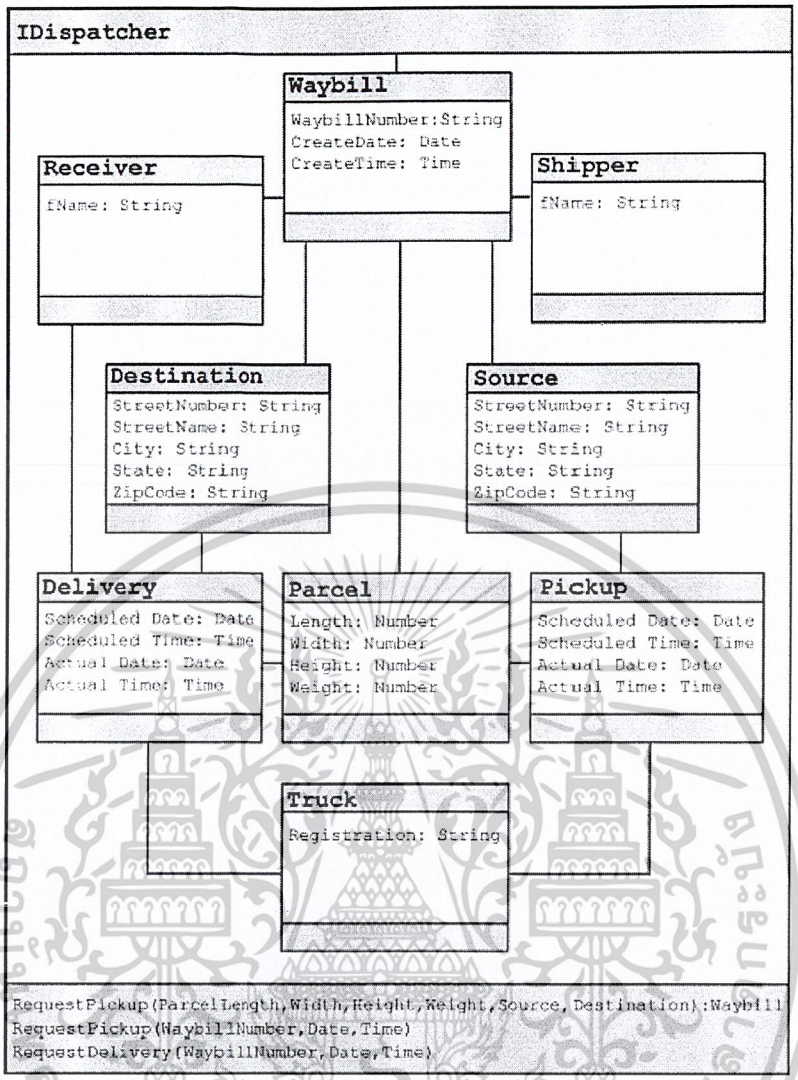
2.1.3 คุณสมบัติของคอมโพเน็นต์

1. **Encapsulated**

เป็นกระบวนการในการซ่อนโค้ดหรืออิมพลีเมนต์เตชันของคอมโพเน็นต์ ผู้ใช้จะรู้เพียงอินเทอร์เฟซและจะใช้งานผ่านมัน โดยไม่จำเป็นต้องรู้ถึงอิมพลีเมนต์เตชันหรือการทำงาน ทำให้การเปลี่ยนแปลงอิมพลีเมนต์เตชันไม่มีผลกระทบต่อผู้ใช้

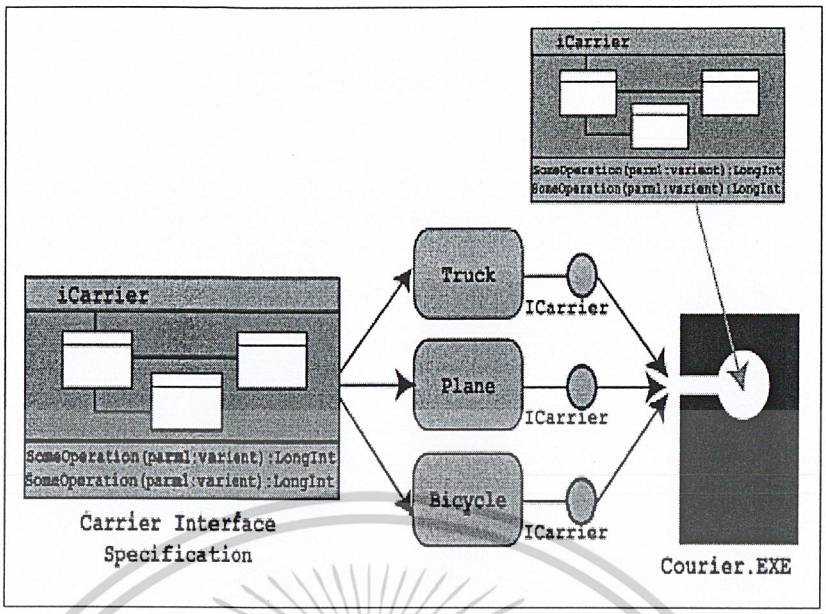
2. **Descriptive**

เนื่องจากคอมโพเน็นต์จะต้องติดต่อผ่านอินเทอร์เฟซเท่านั้น มันจะต้องมีอินฟอร์เมชันของมันเองที่ผู้ใช้สามารถจะเข้าใจได้ โดยอินฟอร์เมชันนั้นจะต้องอธิบายถึงอินเทอร์เฟซ, อิมพลีเมนต์เตชัน และดีพลอยเมนต์



รูปที่ 2-2 ตัวอย่างของอินเทอร์เฟซ

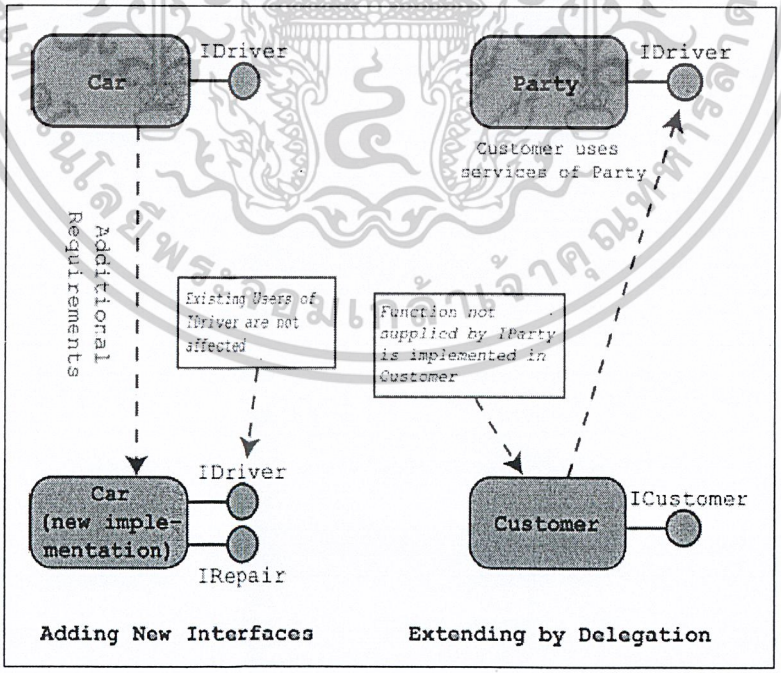
3. **Replaceable** คอมโพเน็นต์มีความสามารถในการเปลี่ยนคอมโพเน็นต์หนึ่งกับคอมโพเน็นต์ใด ๆ ที่มีอินเทอร์เฟซเหมือนกันได้



รูปที่ 2-3 คอมโพเน็นต์และการแทนที่

4. **Extensible**

สามารถที่จะเพิ่มความสามารถได้ มี 2 วิธีคือ การเพิ่มอินเทอร์เฟซและมอบหมายหน้าที่ (Delegating Responsibility) การสร้างคอมโพเน็นต์ใหม่สามารถมอบหมายหน้าที่ให้กับบริการที่มีอยู่ในคอมโพเน็นต์ที่มีอยู่แล้ว



รูปที่ 2-4 การเพิ่มเติมคอมโพเน็นต์

2.2 งานที่เหมาะสมกับการพัฒนาเชิงคอมโพเน็นต์

การพัฒนาเชิงคอมโพเน็นต์ เป็นวิธีการพัฒนาคอมโพเน็นต์โดยใช้คอมโพเน็นต์เฟรมเวิร์ก ซึ่งการคงหน้าที่ของคอมโพเน็นต์ไว้อย่างถูกต้อง (Well defined responsibility) กระทำได้โดยให้คอมโพเน็นต์เชิงธุรกิจ หลีกเลี่ยงโค้ดที่ไม่สามารถทำงานกับอินเทอร์เฟซที่แน่นอนได้

เฟรมเวิร์กคอมโพเน็นต์จะซ่อนบริการต่างๆ ของเอ็นไวรอนเม็นต์ ซึ่งบริการต่างๆ เหล่านี้เป็นมาตรฐานแล้ว เช่น message oriented middle-ware, Transaction Monitors, การรักษาความปลอดภัย และพิสูจน์สิทธิ์

2.3 ประโยชน์ของการพัฒนาเชิงคอมโพเน็นต์

การพัฒนาเชิงคอมโพเน็นต์มีประโยชน์อย่างมากด้านเกี่ยวกับเทคนิค และประโยชน์เหล่านี้ยังนำไปสู่ประโยชน์ด้านธุรกิจด้วย

<u>ประโยชน์ด้านเทคนิค</u>

1. จัดการความซับซ้อนได้ง่ายขึ้น ทำให้คุณภาพของการแก้ปัญหาซอฟต์แวร์ง่ายขึ้น
2. งานซับซ้อนที่ไม่เกี่ยวกับฟังก์ชันธุรกิจ ถูกรวมไว้ในเฟรมเวิร์ก
3. การออกแบบ, อิมพลีเมนต์, ทดสอบ ส่วนต่างๆอย่างอิสระ ทำให้สามารถพัฒนาส่วนต่างๆได้พร้อมกัน
4. การ ไม่ขึ้นต่อกันของคอมโพเน็นต์ลดผลกระทบของการเปลี่ยนแปลง requirement ไม่ให้กระจายไปทั้งระบบ

<u>ประโยชน์ด้านธุรกิจ</u>

1. ผลิตภัณฑ์มีคุณภาพมากขึ้น
2. เร่งเวลาออกสู่ตลาด
3. ใช้ทรัพยากรบุคคลได้คุ้มค่าขึ้น
4. ลดค่าใช้จ่าย
5. เพิ่มปริมาณการนำกลับมาใช้ ในโครงการต่อไป

บทที่ 3
เว็บเซอร์วิส (Web Services)

3.1 เว็บเซอร์วิส คืออะไร

เว็บเซอร์วิส คือ แอพพลิเคชั่น หรือ โปรแกรมที่ทำงานอย่างใดอย่างหนึ่ง ในลักษณะให้บริการ โดยจะถูกเรียกใช้งานจาก แอพพลิเคชั่น อื่นๆ ในรูปแบบ RPC (Remote Procedure Call) ซึ่งการเปลี่ยนคือ XML ทำให้เราสามารถเรียกใช้คอมโพเน็นต์ ใดๆ ก็ได้ ในแพลตฟอร์ม (platform) ใดๆ ก็ได้ บน โพรโตคอล HTTP ซึ่งเป็นโพรโตคอลสำหรับเวิร์ด ไวด์ เว็บ (World Wide Web) อันเป็นช่องทางที่ได้รับการยอมรับทั่วโลกในการติดต่อสื่อสารกันระหว่างแอพพลิเคชั่นกับแอพพลิเคชั่นในปัจจุบัน

3.2 เว็บเซอร์วิสกับแอพพลิเคชั่น

เทคโนโลยีในการกระจายข่าวสารข้อมูลทาง Internet ในปัจจุบันก็คือ เว็บเพจ แต่จากการที่มันมีความสามารถที่จะทำงานได้ด้วยการรวมภาษาทั้งไคล์เอ็นต์ (Client) และ เซิร์ฟเวอร์ สไลด์ สคริปท์ (Server Side Script) ไว้ในตัวเองเช่นภาษา VBScript, Java Script หรือ ASP, PHP, JSP นั้นทำให้เว็บเพจมีลักษณะคล้ายแอพพลิเคชั่น จึงถูกเรียกรวมกันว่า เว็บแอพพลิเคชั่น (Web Application)

เว็บแอพพลิเคชั่นสามารถตอบสนองความคิดดีสทริบิวเทต โปรแซสซิ่ง (Distributed Processing) ได้ในระดับหนึ่งซึ่งก็คือ การแบ่งการประมวลผลไว้ที่ฝั่งไคล์เอ็นต์ และฝั่งเซิร์ฟเวอร์ และมักจะมีการใช้ฐานข้อมูล (database) ควบคู่กับการทำเว็บแอพพลิเคชั่นไปด้วยตามความต้องการในการทำอี-บิสิเนส (E-Business) และ อี-คอมเมิร์ช (E-Commerce) ที่กำลังเป็นที่นิยมในปัจจุบัน และเกิดปัญหาที่ตามาคือ เรื่องของการจ่ายเงินหรือที่เรียกว่า อี-เพเมนต์ (E-payment) หรือ เพเมนต์-เกทเวย์ (Payment-Gateway) ซึ่งเว็บแอพพลิเคชั่นที่ทำอี-คอมเมิร์ช ต้องใช้บริการจากธนาคารออนไลน์ (Online) ในการจัดเก็บเงินกับลูกค้า เพราะด้วยเทคโนโลยีนี้การใช้บริการเก็บเงินจากธนาคารออนไลน์จำเป็นที่ผู้ค้าต้องไปทำการตกลงกับธนาคารและเขียนโปรแกรมให้ตรงตามมาตรฐานที่ธนาคารออนไลน์กำหนดไว้

ด้วยปัญหายุ่งยากในการค้นหา ติดต่อและตกลงในการขอใช้บริการเก็บเงินจากธนาคารออนไลน์ แนวคิด เว็บเซอร์วิสจึงดูเหมือนเป็นทางออกของปัญหานี้ ความเด่นของเทคโนโลยี เว็บเซอร์วิสนี้ก็คือ การทำให้ เว็บกับเว็บ สามารถติต่อสื่อสารกันได้ด้วยเอกสาร XML ที่ทั้งคนและคอมพิวเตอร์เข้าใจ และคอมพิวเตอร์ยังสามารถนำข้อมูลนั้นไปประมวลผลต่อได้ ด้วยเอกสาร XML นี้เองทำให้ Web สามารถส่งข้อมูลที่จำเป็นไปให้อีก Web หนึ่งทำงานบางอย่างให้หรือใช้บริการนั่นเอง ทำให้เป็นการง่ายที่จะเขียนโปรแกรมที่จะติดต่อสื่อสารหรือ ขอใช้บริการเก็บเงินจากธนาคารออนไลน์แต่สำหรับเว็บแอพพลิเคชั่นที่ใช้การส่งข้อมูลเป็น html ทำให้ข้อมูลนั้นไม่สามารถนำไปใช้ต่อได้ การเขียนโปรแกรมจึงยุ่งยากตามที่กล่าวด้านบน

แนวคิดของ เว็บเซอร์วิส ก็คือ เว็บที่สามารถทำงานอะไรบางอย่างหรือก็คือให้บริการบางอย่างจากการร้องขอจากต่าง Server ด้วยเหตุนี้ทำให้เทคโนโลยีเว็บเซอร์วิส เอื้อต่อแนวคิดดีสทริบิวเทตโปรเซส

ซึ่งมากกว่าเว็บแอพพลิเคชั่น และเมื่อประกอบกับการที่เว็บเซอร์วิสมี UDDI ทำให้ เว็บเซอร์วิส สามารถค้นหาบริการต่างๆ ที่ต้องการได้จากทั่วทุกมุมโลก

ในอนาคตอาจเป็นไปได้ว่าแอพพลิเคชั่น ก็อาจเป็นเพียงแค่การรวมเซอร์วิส ที่แต่ละ เว็บเซอร์วิสมีบริการมาให้ใช้เท่านั้นเอง

3.3 รูปแบบการเชื่อมต่อ

รูปแบบการเชื่อมต่อตอนนี้รองรับ 3 โปรโตคอล ได้แก่ HTTP GET, HTTP POST และ SOAP (Simple Object Access Protocol) เนื่องจากโปรโตคอลเหล่านี้เป็นมาตรฐานของเว็บอยู่แล้ว จึงง่ายที่ไคลเอนต์จะเรียกใช้

HTTP GET : query string จะรวมอยู่ใน URL ของเซิร์ฟเวอร์นั้น

HTTP POST : จะทำการรวม ชื่อ ค่าตัวแปรต่างๆ ไว้ในส่วนของบอดี้ ของ HTTP request Message

SOAP : SOAP จะแตกต่างจากทั้งสองวิธีข้างต้น มันจะใช้ XML ในการจัดวางรูปแบบเมสเสจที่ส่งไปจะมีโครงสร้างที่ดีกว่าและสามารถส่งผ่านข้อมูลที่ซับซ้อนได้ข้อแตกต่างอีกอย่างคือ SOAP สามารถใช้บนทรานสปอร์ส โปรโตคอล (transport protocols) อื่นๆได้ เช่น SMTP นอกเหนือจาก HTTP

3.4 WSDL

WSDL (Web Services Description Language) เกิดจากความร่วมมือระหว่าง IBM และ Microsoft WSDL เป็นภาษาที่ใช้อธิบายคุณลักษณะของเว็บเซอร์วิส และวิธีการติดต่อกับเว็บเซอร์วิสนั้นๆ โดยใช้ไวยากรณ์ของภาษา XML ซึ่ง WSDL อยู่ในความดูแลของ W3C เวอร์ชั่นล่าสุด คือ WSDL 1.1 ส่วนในทางปฏิบัติ หากเราต้องการ สร้างเว็บเซอร์วิสขึ้นมาเป็นของตนเอง ก็สามารถสร้างเอกสาร WSDL ได้โดยอัตโนมัติ เราจึงไม่ต้องไปกังวลในรายละเอียดในข้อกำหนดใน WSDL มากนักซึ่งทั้ง IDL และ WSDL นั้นอธิบาย อินเทอร์เฟซของ เมธอด ที่เรียกใช้งาน แล้ว แสดง พารามิเตอร์ที่ส่งเข้าส่งออกด้วย สิ่งแตกต่างกันคือ คำอธิบายต่างๆใน WSDL จะอยู่ในรูปแบบของ XML

3.5 Web Services Discovery

เป็นกระบวนการในการค้นหาเซอร์วิสและทำการตรวจสอบ WSDL ของเว็บเซอร์วิสว่ามีอยู่จริงไหม แบ่งเป็น 2 แบบ

3.5.1 Static discovery

เราต้องทำการบอกชัดเจนในไฟล์ .disco ว่า wsdl อยู่ที่ไหน

```
<?xml version="1.0" ?>
<disco:discovery xmlns:disco="http://schemas.xmlsoap.org/disco">
</disco:discovery>
```

3.5.2 Dynamic discovery

เราสามารถตั้งค่าให้เป็นแบบไดนามิก ดิสคอพเวอรี่ (dynamic discovery) ได้ ซึ่งจะทำให้เว็บเซอร์วิสทั้งหมดที่อยู่ภายใต้ URL ที่เรากำหนดจะถูกลิสต์ขึ้นมาโดยอัตโนมัติ

3.6 UDDI

UDDI (Universal Description, Discovery and Integration) เป็นมาตรฐานที่จัดตั้งขึ้นโดยบริษัท IBM, Microsoft และบริษัทยักษ์ใหญ่ทางธุรกิจ B2B (Business-to-Business) อื่น ๆ UDDI ถูกสร้างขึ้นมาเป็นมาตรฐาน ในการค้นหาบริการเว็บเซอร์วิสสำหรับคู่ค้าทางธุรกิจ ซึ่งเปรียบได้กับฐานข้อมูลขนาดใหญ่ ซึ่งมีข้อมูลของเว็บเซอร์วิสที่เปิดให้บริการ โดยที่เว็บไซต์ สำหรับค้นหาเว็บเซอร์วิสมีอยู่หลายที่อย่างเช่น

http://uddi.microsoft.com/search.aspx

http://www-3.ibm.com/services/uddi/testregistry/find

http://uddi.org

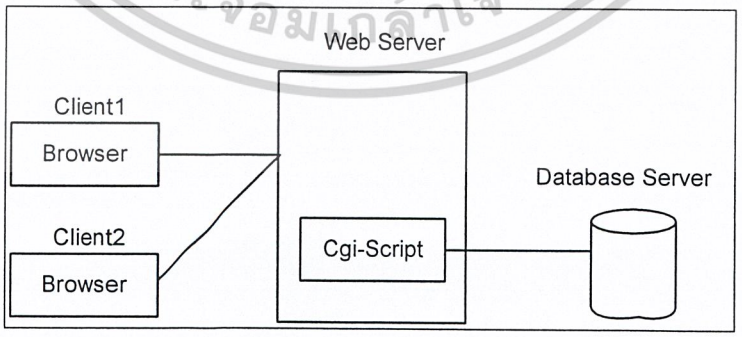
บทที่ 4

สถาปัตยกรรม J2EE (J2EE Architecture)

เอนเตอร์ไพรส์จาวาบีน (EJB) เป็นสถาปัตยกรรมคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์ (server-side component architecture) ซึ่งช่วยให้กระบวนการพัฒนาแอพพลิเคชันออบเจ็กต์แบบกระจาย (distributed object application) ในระดับเอ็นเตอร์ไพรส์ด้วยภาษาจาวานั้นทำได้ง่ายขึ้น ด้วย EJB เราสามารถสร้างแอพพลิเคชันที่สามารถเพิ่มขยายได้ (scalable) , มีความน่าเชื่อถือ (reliable) และมีความปลอดภัย (secure) โดยไม่ต้องเขียนโค้ดเกี่ยวกับการใช้งานออบเจ็กต์แบบกระจายซึ่งมีความซับซ้อนสูงด้วยตนเอง เราสามารถสร้างคอมโพเน็นต์บนฝั่งเซิร์ฟเวอร์ด้วยภาษาจาวาได้อย่างง่ายดายและรวดเร็ว นอกจากนี้ EJB ยังถูกออกแบบขึ้นมาเพื่อสนับสนุนความสามารถในการเคลื่อนย้ายได้ (portability) และความสามารถนำกลับมาใช้ใหม่ได้ (reusability) ของ แอพพลิเคชันในแต่ละมิดเดิลแวร์ของแต่ละผู้ผลิต

EJB นั้นเป็นการทำงานต่างๆบนฝั่งเซิร์ฟเวอร์ทั้งสิ้น เช่น วิธีการทำงานที่สลับซับซ้อน หรือการทำธุรกรรมทางธุรกิจที่ต้องใช้ทรัพยากรมากๆ การทำงานบนฝั่งเซิร์ฟเวอร์นั้นจะมีความต้องการที่ต่างออกไปจากสิ่งแวดล้อมที่ใช้งาน GUI ทั่วๆไป การทำงานบนฝั่งเซิร์ฟเวอร์โดยปกติต้องทำงานหนักและตลอดเวลา, ทนต่อความผิดพลาดร้ายแรง, ความสามารถในการทำทรานแซคชั่น และ ความปลอดภัยของผู้ใช้งาน ซึ่งเวบปกติที่ทำด้วยแอพเพล็ต(applets), เซิร์ฟเลต(servlets) นั้นไม่รองรับการทำงานในลักษณะนี้ เนื่องจากทั้งเซิร์ฟเลตและแอพเพลตนั้นเหมาะที่จะทำงานที่ฝั่งไคลเอนต์(client-side) อย่างเดียวเท่านั้น

ในยุคแรกๆนั้น ผู้ใช้งานระบบสามารถใช้เบราเซอร์(browser) ติดต่อกับอินเตร์เนทในแบบกราฟฟิก แอพพลิเคชันบนฝั่งเซิร์ฟเวอร์ ส่วนประกอบของเว็บยังไม่มีความซับซ้อนเช่นในปัจจุบัน โดยใช้หลักการพื้นฐานของสถาปัตยกรรมแบบไคลเอ็นต์/เซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์ในที่นี้ก็คือเว็บเซิร์ฟเวอร์ และไคลเอ็นต์ก็คือ เว็บบราวเซอร์ เช่น Netscape, Mosaic, Lynx , หรือ NCSA โดยเว็บเซิร์ฟเวอร์หนึ่ง ๆ จะให้บริการกับเว็บบราวเซอร์หลาย ๆ ตัวได้พร้อมกัน



รูปที่ 4-1 รูปแบบการใช้งานในยุคแรกๆ

แต่ข้อเสียของเว็บไซต์ในตอนนั้นก็คือ หน้าเว็บนั้นเป็น HTML ที่ตายตัว (static) ข้อมูลที่ปรากฏอยู่ไม่สามารถเปลี่ยนแปลงได้เลย เว็บเซิร์ฟเวอร์จะทำหน้าที่แค่เพียงส่งโค้ด HTML กลับมาเท่านั้น เทคโนโลยีที่เกิดขึ้นตามมาก็คือ CGI (Common Gateway Interface) ซึ่งใช้ในการสร้างหน้าเพจแบบไดนามิก เนื่องจากมันมีการประมวลผลก่อนที่จะส่งกลับไป

หลังจากที่ CGI ได้รับความนิยมจนกลายมาเป็นส่วนหนึ่งในมาตรฐานของเว็บแล้ว ก็ได้มีเทคโนโลยีที่ใกล้เคียงกันตามออกมามากมาย เช่น ASP , PHP , JSP , Servlet และอื่น ๆ อย่างไรก็ตามในโครงงานนี้จะกล่าวถึงเฉพาะเทคโนโลยีแอพพลิเคชันทางฝั่งเซิร์ฟเวอร์ของจาวาเท่านั้น

ดีพลอยเมนต์บนฝั่งเซิร์ฟเวอร์ (server-side deployment) คือซอฟต์แวร์ที่ถูกเขียนขึ้นมาให้สามารถรองรับการใช้งานจากผู้ใช้ได้หลาย ๆ คนพร้อมกัน , มีความปลอดภัย , เชื่อถือได้ และมีประสิทธิภาพ ตัวอย่างของดีพลอยเมนต์บนฝั่งเซิร์ฟเวอร์ได้แก่

- ธนาคาร – เครื่องเอทีเอ็มจำนวนมากจะต้องติดต่อเข้าไปยังเครื่องเซิร์ฟเวอร์กลางของธนาคาร
- สาขาร้านค้า – เช่นร้านค้าในเครือข่ายของ Wal-Mart โดยร้านค้าแต่ละร้านจะส่งรายละเอียดการซื้อ สินค้าของลูกค้าแต่ละรายไปยังเซิร์ฟเวอร์กลางของ Wal-Mart
- ฝ่ายสนับสนุนลูกค้า – วิศวกรหลาย ๆ คนสามารถใช้เครื่องเทอร์มินอลเชื่อมต่อเข้ามาเซิร์ฟเวอร์กลางและนำข้อมูลเกี่ยวกับลูกค้าที่ถูกเก็บที่เซิร์ฟเวอร์ไปใช้งานได้
- เว็บไซต์ – ในเวลาใดเวลาหนึ่งจะมีผู้ใช้จำนวนมากทำการติดต่อเข้ามายังเว็บเซิร์ฟเวอร์ และเว็บเซิร์ฟเวอร์เหล่านี้จะต้องติดต่อไปยังเซิร์ฟเวอร์กลาง เพื่อดึงเอาข้อมูลและเรียกใช้การประมวลผลที่จำเป็น

ในการทำดีพลอยเมนต์ที่ดีนั้น จะแบ่งซอฟต์แวร์ออกเป็นเลเยอร์ (layer) โดยในแต่ละเลเยอร์จะมีความรับผิดชอบในส่วนของดีพลอยเมนต์ที่แตกต่างกันออกไป และอาจประกอบด้วยหนึ่งคอมโพเน็นต์หรือมากกว่าก็ได้ เลเยอร์เหล่านี้เป็นเพียงการแบ่งเลเยอร์ในทางความคิด (logical) เท่านั้น ซึ่งอาจแตกต่างจากการแบ่งหน้าที่ความรับผิดชอบของเครื่องคอมพิวเตอร์แต่ละเครื่องในทางกายภาพ (physical) ได้

ระบบที่แบ่งการทำงานเป็นเลเยอร์เป็นระบบที่ถูกออกแบบมาอย่างดี เนื่องจากแต่ละเลเยอร์นั้นจะมีหน้าที่รับผิดชอบที่แตกต่างกันออกไป การแบ่งเลเยอร์ทั่วไปจะแบ่งเลเยอร์เป็นดังนี้

พรีเซนต์เทชัน เลเยอร์ – ประกอบด้วยคอมโพเน็นต์ที่จัดการกับ การติดต่อกับผู้ใช้ (user interface) และการโต้ตอบกับผู้ใช้ (user interaction) พรีเซนเทชันเลเยอร์ของแอพพลิเคชันแบบ stand-alone อาจเขียนด้วยภาษาวิชวลเบสิค ส่วนพรีเซนเทชันเลเยอร์ของดีพลอยเมนต์แบบ Web-base นั้น จะใช้ Java Servlet , JSP , หรือ Java Applet

บิซิเนส ลอจิก เลเยอร์ -- ประกอบด้วยคอมโพเน็นต์ที่ใช้ในการแก้ปัญหาทางธุรกิจ คอมโพเน็นต์เหล่านี้อาจเป็น engine ที่มีความสามารถสูง เช่น engine ในการจัดการแคตาล็อก หรือ คำนวณราคาสินค้า โดย

ปกติแล้วคอมโพเน็นต์เหล่านี้จะต้องถูกเขียนขึ้นโดยภาษาที่มีคุณสมบัติ type-safe เช่น ภาษาจาวา หรือ ภาษา C++

ดาต้า เลเยอร์ – ถูกเรียกใช้โดยบิซิเนสลอจิกเลเยอร์เพื่อเก็บข้อมูลแบบถาวร โดยทั่วไปดาต้า เลเยอร์จะเป็นระบบฐานข้อมูล

ข้อดีของการแบ่งคอมโพเน็นต์เป็นเลเยอร์คือ ทำให้สามารถเปลี่ยนแปลงแต่ละเลเยอร์ได้โดยจะไม่มีผลกระทบต่อเลเยอร์อื่น ๆ เช่น เราสามารถเปลี่ยนแปลงหน้าจออินเทอร์เฟซในพรีเซนเทชันเลเยอร์ได้โดยไม่มีผลกระทบต่อเลเยอร์อื่น

แต่การแบ่งลักษณะการทำงานออกเป็นเลเยอร์นี้ เป็นการแบ่งในทางความคิดเท่านั้น ส่วนในทางกายภาพเราจะแบ่งออกเป็นหน่วยที่เรียกว่า "เทียร์" (tier) โดยสถาปัตยกรรมแบบหนึ่งเทียร์ คือการรวมเลเยอร์ทั้งสามเป็นส่วนเดียวทางกายภาพ , สถาปัตยกรรมแบบสองเทียร์ คือการแบ่งเลเยอร์ทั้งสามออกเป็นสองส่วนในทางกายภาพ และสถาปัตยกรรมแบบสามเทียร์ คือการแบ่งเลเยอร์ทั้งสามออกเป็นสามส่วนในทางกายภาพ

4.1 สถาปัตยกรรมแบบหนึ่งเทียร์ (1-Tier Architecture)

ในสถาปัตยกรรมนี้ ทั้งสามเลเยอร์จะรวมกันเป็นคอมโพเน็นต์เดียวซึ่งโดยปกติจะเป็นโปรแกรมทำงานในเครื่อง เช่น เวิร์คโปรเซสเซอร์ หรือสเปรดชีท ซึ่งหากมีการแก้ไขเปลี่ยนแปลงที่เลเยอร์ใดๆ ก็ต้องแก้ไขโค้ดทั้งหมดทุกครั้ง

4.2 สถาปัตยกรรมแบบสองเทียร์ (2-Tier Architecture)

ประกอบด้วย 2 ส่วนหลักๆ คือ โปรแกรมในส่วน พรีเซนต์เทชัน เลเยอร์ ซึ่งเป็นส่วนที่กำหนดรูปแบบการติดต่อระหว่างผู้ใช้กับแอพพลิเคชั่น และ โปรแกรมส่วนของ บิซิเนส ลอจิก เลเยอร์ ซึ่งกำหนดว่าข้อมูลจะถูกจัดการอย่างไรในการทำธุรกิจนั้นๆ โปรแกรมทั้ง 2 ส่วนจะถูกติดตั้งและทำงานบนเครื่องไคลเอนต์

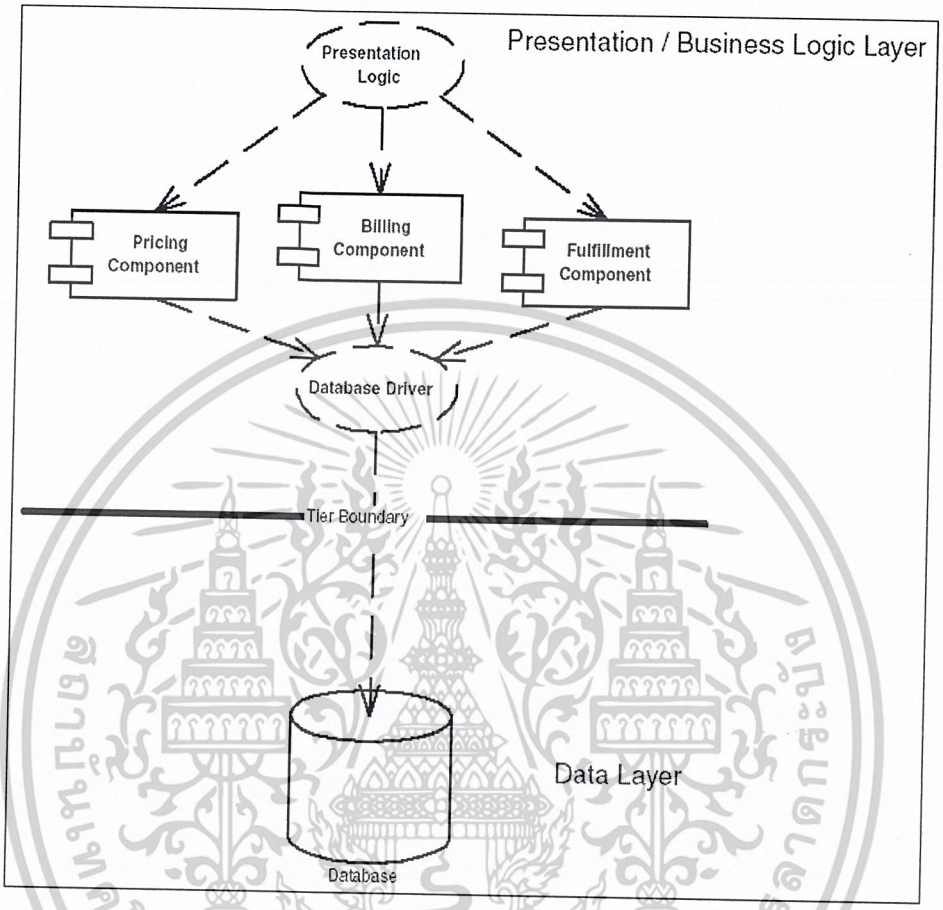
สถาปัตยกรรมนี้จะรวมบิซิเนสลอจิกเลเยอร์เข้ากับอีกหนึ่งในสองเลเยอร์ที่เหลือ ซึ่งทางเลือกที่เป็นไปได้คือรวมบิซิเนสลอจิกเลเยอร์เข้ากับพรีเซนเทชันเลเยอร์ และรวมบิซิเนสลอจิกเลเยอร์เข้ากับดาต้าเลเยอร์

4.2.1 การรวมบิซิเนสลอจิกเลเยอร์เข้ากับพรีเซนเทชันเลเยอร์

พรีเซนเทชันเลเยอร์ และบิซิเนสลอจิกเลเยอร์นั้น สามารถนำมารวมกันเป็นเทียร์เดียวได้ โดยแยกดาต้าแอ็กเซสเลเยอร์เป็นอีกหนึ่งเทียร์ต่างหาก ซึ่งเสมือนกับเป็นการสร้างกำแพงกั้นระหว่างบิซิเนสลอจิกเลเยอร์และดาต้าแอกเซสเลเยอร์ ดังรูปที่ 4-2

ถ้าหากเรามองว่าเทียร์แรกเป็นเทียร์ของไคลเอ็นต์ และเทียร์ที่สองเป็นเซิร์ฟเวอร์ สถาปัตยกรรมนี้จะทำให้ทางฝั่งไคลเอ็นต์มีขนาดใหญ่ (fat) ส่วนทางฝั่งเซิร์ฟเวอร์จะมีขนาดเล็ก (thin)

ในสถาปัตยกรรมแบบ 2 เทียร์ ไคลเอ็นต์แอพพลิเคชันจะติดต่อกับดาต้าเลเยอร์โดยผ่านทาง database bridge API เช่น Open Database Connectivity (ODBC) หรือ Java Database Connectivity (JDBC)



รูปที่ 4-2 ลักษณะของสถาปัตยกรรมแบบสองเทียร์

1) มีค่าใช้จ่ายในการดีพลอยสูง

ไดรเวอร์ของระบบฐานข้อมูลจะต้องถูกติดตั้งลงในไคลเอ็นต์ทุกเครื่อง ที่อยู่ในไคลเอ็นต์เทียร์ ซึ่งอาจเป็นเครื่องคอมพิวเตอร์จำนวนหลายร้อย หรือหลายพันเครื่อง

2) มีค่าใช้จ่ายในการเปลี่ยนไดรเวอร์ของระบบฐานข้อมูลสูง

การเปลี่ยนแปลงไดรเวอร์ของระบบฐานข้อมูล จะต้องการการเปลี่ยนแปลงที่เครื่องไคลเอ็นต์ทุกเครื่อง ซึ่งเป็นขั้นตอนที่สิ้นเปลืองมาก และเกือบจะเป็นไปไม่ได้ในความเป็นจริงสำหรับระบบงานที่มีขนาดใหญ่

3) การเปลี่ยนแปลงโครงสร้างของฐานข้อมูลมีค่าใช้จ่ายที่สูง

เนื่องจากไคลเอ็นต์มีขนาดใหญ่ การเข้าถึงฐานข้อมูลเป็นการเข้าถึงโดยตรงโดยผ่านทาง JDBC , SQL/J หรือ ODBC ซึ่งหมายความว่า การเปลี่ยนแปลงโครงสร้างของฐานข้อมูลให้รองรับบิซิเนสลอจิกใหม่ ๆ จะต้องเปลี่ยนแปลงที่ไคลเอ็นต์ทุกเครื่องด้วย

4) การเปลี่ยนแปลงรูปแบบของฐานข้อมูลมีค่าใช้จ่ายที่สูง

ไคลเอ็นต์ที่มีขนาดใหญ่จะยึดติดอยู่กับ API ของระบบฐานข้อมูล เช่น API ของระบบฐานข้อมูล เชิงสัมพันธ์ (relational database) หรือ API ของระบบฐานข้อมูลแบบออบเจ็กต์ (object database) ถ้าหากเราตัดสินใจที่จะเปลี่ยนแปลงรูปแบบของฐานข้อมูล เช่นเปลี่ยนจากการใช้ฐานข้อมูลเชิงสัมพันธ์มาเป็นการใช้ฐานข้อมูลแบบออบเจ็กต์ นอกจากเราจะต้องทำการดีพลอยไคลเอ็นต์ทั้งหมดใหม่แล้ว ยังต้องมีการเปลี่ยนแปลงโค้ดภายในตัวไคลเอ็นต์ให้เข้ากับรูปแบบฐานข้อมูลที่เปลี่ยนแปลงไปอีกด้วย

5) การเปลี่ยนแปลงบิซิเนสลอจิกเลเยอร์มีค่าใช้จ่ายที่สูง

หากมีการเปลี่ยนแปลงที่เกิดขึ้นกับบิซิเนสลอจิกเลเยอร์ ไคลเอ็นต์จะต้องคอมไพล์ใหม่ และทำการดีพลอยใหม่ทั้งหมด

6) การเชื่อมต่อกับฐานข้อมูลมีค่าใช้จ่ายที่สูง

ไคลเอ็นต์แต่ละเครื่องจะติดต่อกับฐานข้อมูลโดยใช้ connection ของมันเอง ในขณะที่ connection มีจำนวนได้จำกัดและมีค่าใช้จ่ายในการสร้างสูง เมื่อไคลเอ็นต์ไม่ได้ใช้ฐานข้อมูล connection จะยังอยู่และ ไคลเอ็นต์อื่นไม่สามารถนำมันไปใช้ได้

7) ประสิทธิภาพของระบบเครือข่ายลดลง

ในแต่ละครั้งที่ไคลเอ็นต์ทำการติดต่อกับฐานข้อมูล จะมีการส่งข้อมูลจำนวนมากไปมาระหว่าง บิซิเนสลอจิกเลเยอร์ และดาต้าเลเยอร์ ซึ่งหากว่าตัวกลางระหว่างเลเยอร์ทั้งสองนี้เป็นระบบเครือข่าย อาจทำให้ข้อมูลในระบบเครือข่ายนั้นมากจนเกิดการติดขัด ซึ่งจะมีผลให้การติดต่อกับฐานข้อมูลนั้นทำได้ช้าลง

4.2.2 การรวมส่วนบิซิเนสลอจิกเลเยอร์บางส่วน เข้ากับดาต้าเลเยอร์

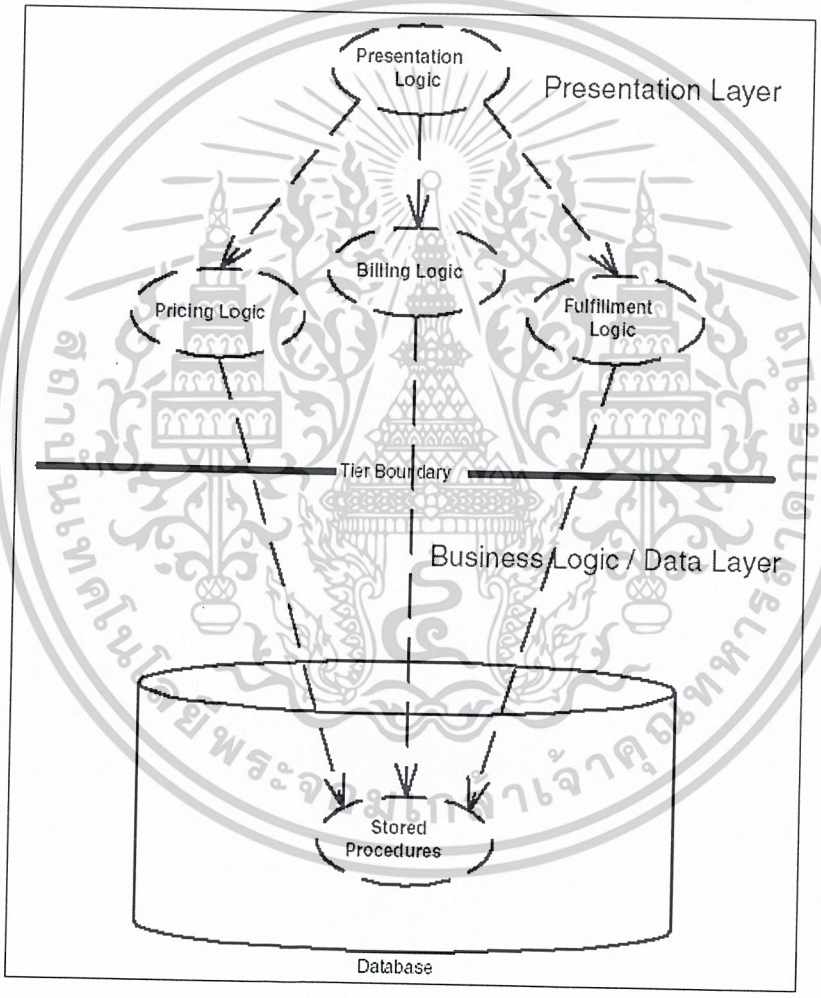
ในระยะหลัง ๆ มานี้ ดีพลอยเมนต์ต่าง ๆ จะนำบิซิเนสลอจิกเลเยอร์มารวมกับดาต้าเลเยอร์ และแยกไว้เป็นเทียร์หนึ่ง ซึ่งแสดงให้เห็นในรูปที่ 4-3 โดยหากมองว่าเทียร์ที่หนึ่งเป็นไคลเอ็นต์ ส่วนเทียร์ที่สองเป็นเซิร์ฟเวอร์ สถาปัตยกรรมแบบนี้จะให้ทำไคลเอ็นต์นั้นมีขนาดเล็ก ส่วนเซิร์ฟเวอร์นั้นจะมีขนาดใหญ่

ในสถาปัตยกรรมดังกล่าว จะนำบิซิเนสลอจิกบางส่วนเข้าไปไว้ในฐานข้อมูล ซึ่งโดยทั่วไปจะเป็นลอจิกที่เกี่ยวข้องกับข้อมูล ฐานข้อมูลจะอนุญาตให้เอ็กซิคิวต์ (execute) ลอจิกภายในฐานข้อมูลได้โดยการเขียนโมดูลที่เรียกว่า store procedure

การรวมลอจิกเข้าไปเป็น store procedure จะเป็นการเพิ่ม scalability และประสิทธิภาพในการทำงาน เนื่องจากว่าลอจิกบางส่วนถูกรวมเข้ากับฐานข้อมูล ดังนั้นการใช้งานเครือข่ายเพื่อติดต่อระหว่าง

ลอจิกกับฐานข้อมูลจะลดลง แทนที่จะทำการ query ข้อมูลในฐานข้อมูลจำนวน n ครั้ง เราสามารถที่จะเรียก store procedure ซึ่งอยู่ภายในฐานข้อมูลให้ทำการ query n ครั้งแทน ซึ่งจะลดปริมาณการใช้เครือข่ายไปได้มากทีเดียว และยังทำให้การประมวลผลฐานข้อมูลเร็วขึ้นอีกด้วย

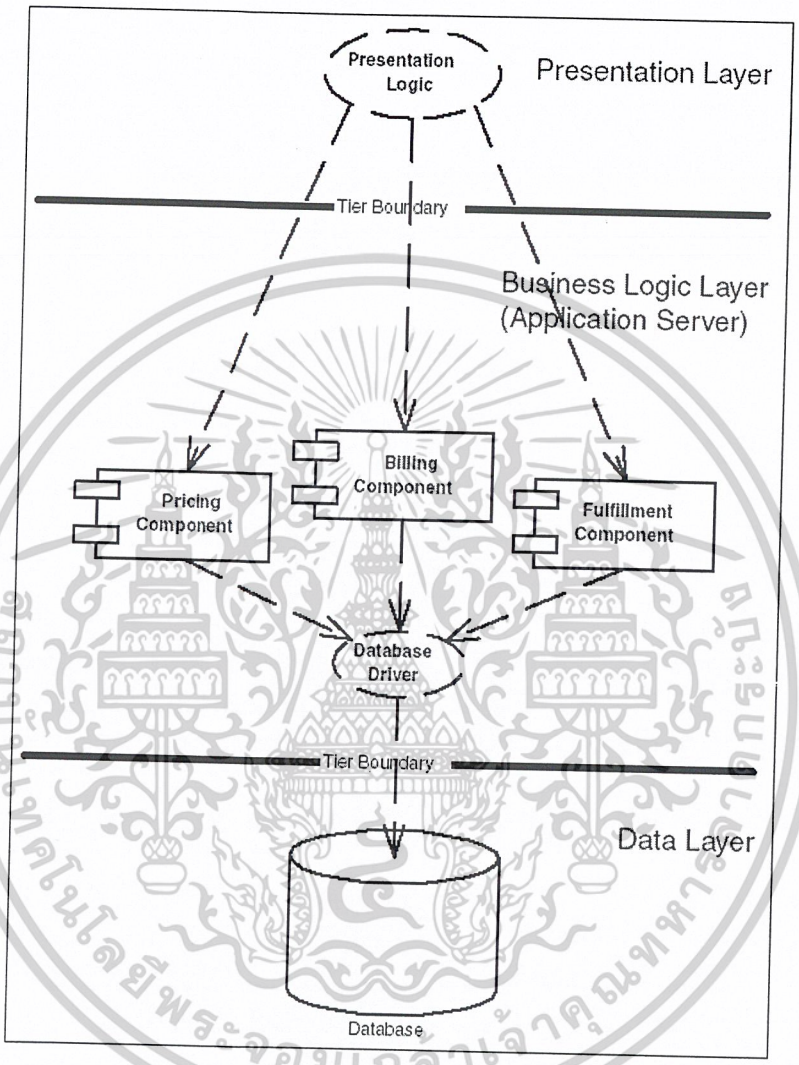
จะเห็นว่าการรวมเอาลอจิกที่เกี่ยวข้องกับฐานข้อมูลเข้าไปไว้ในฐานข้อมูล ช่วยเพิ่มประสิทธิภาพโดยรวมของดีพลอยเมนต์ขึ้นอย่างมาก แต่บางปัญหาซึ่งเกิดกับสถาปัตยกรรมแบบ 2 เทียร์ก็ยังคงมีอยู่ในขณะที่การสร้าง store procedure นั้นเป็นวิธีการที่สามารถแก้ปัญหาได้หลายอย่าง แต่ตัวมันเองก็ทำให้เกิดปัญหาใหม่ ๆ ขึ้นมาเช่นกัน โดยภาษาที่ใช้ในการเขียน store procedure นั้น จะผูกอยู่กับฐานข้อมูลที่ใช้ ทำให้การเปลี่ยนแปลงระบบฐานข้อมูลที่ใช้อยู่นั้นทำได้ยาก แต่อย่างไรก็ตาม ภาษาจาวาได้ถูกนำมาใช้เขียน store procedure มากขึ้นเรื่อย ๆ ซึ่งจะช่วยเพิ่มความสามารถในการโยกย้าย (portability)



รูปที่ 4-3 สถาปัตยกรรมแบบ 2 เทียร์

4.3 สถาปัตยกรรมแบบสามเทียร์ (3-Tier Architecture)

สถาปัตยกรรมแบบการแยกเลเยอร์ทั้งสามออกจากกันไว้ในแต่ละเทียร์ ตัวอย่างที่ดีของสถาปัตยกรรมแบบสามเทียร์ได้แก่ web-base ดีพลอยเมนต์แบบ 3 เทียร์ ดังแสดงในรูปที่ 4-4 ซึ่งเทียร์ทั้งสามนั้น ถูกแบ่งออกดังนี้



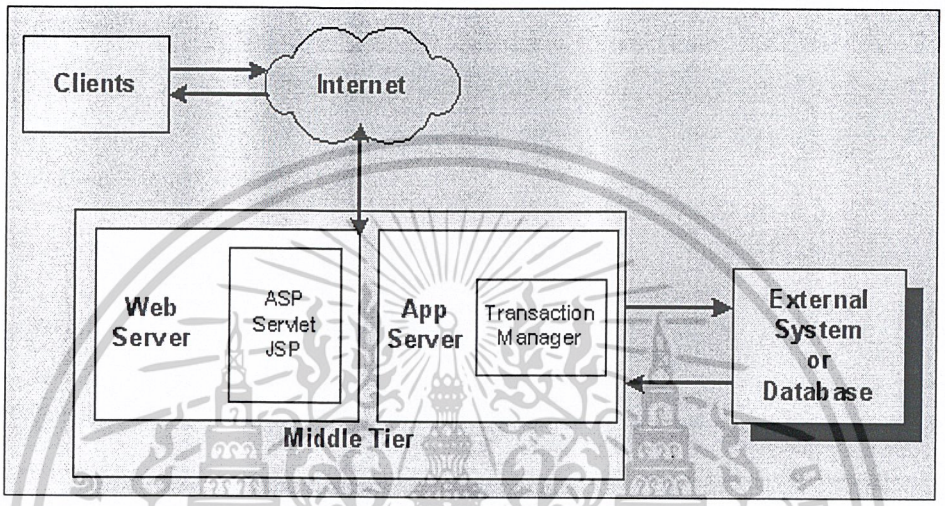
รูปที่ 4-4 Web-base ดีพลอยเมนต์แบบ 3 เทียร์

ระบบสามเทียร์ แยกส่วนโปรแกรมพรีเซนเทชั่น เลเยอร์ ออกจากส่วนโปรแกรมบิสซิเนส เลเยอร์ คือ ให้การทำงานในส่วนบิสสิเนสลอจิกนั้นอยู่บนเครื่องมิดเดิ้ลเทียร์เซิร์ฟเวอร์ (middle-tier server) ส่วนโปรแกรมของพรีเซนเทชั่น ลอจิกอยู่บนเครื่องไคลเอนต์ โปรแกรมของทางฝั่งไคลเอนต์จึงมีขนาดเล็กลง เรียกว่า ทินไคลเอนต์(thin client)

4.4 สถาปัตยกรรมแบบ n เทียร์ (N-Tier Architecture)

สถาปัตยกรรม N เทียร์ เป็นการเพิ่มเทียร์ลงในสถาปัตยกรรมสองเทียร์เพียงหนึ่งเทียร์ หรือมากกว่า การติดตั้งสถาปัตยกรรมแบบนี้ พรีเซนเทชั่นเลเยอร์ บิสสิเนสลอจิกเลเยอร์ และ ดาต้าเลเยอร์จะถูกแยกออกจากกันไปตามเทียร์ทางกายภาพ

ตัวอย่างของสถาปัตยกรรมแบบ n เทียร์ ที่เห็นชัดเจนก็คือ การจัดระบบดีพลอยเมนต์แบบสามเทียร์แบบ เวบเบสท์ (three-tier Web-based) ดังรูปที่ 4-5



รูปที่ 4-5 ระบบไคลเอ็นต์/เซิร์ฟเวอร์ดีพลอยเมนต์แบบ 4 เทียร์

โดยแบ่งเทียร์ ออกเป็นดังนี้

1. พรีเซนเทชันเทียร์ (Presentation Tier)

อาจประกอบไปด้วย Java Servlet , สคริปต์พวก Active Server Page หรือ Java Server Page และลอจิกที่เชื่อมการทำงานของแต่ละส่วนเข้าด้วยกัน

2. บิซิเนสลอจิกเทียร์ (Business Logic Tier)

แอพพลิเคชันเซิร์ฟเวอร์เป็นสิ่งจำเป็นในการให้สภาพแวดล้อมในการทำงานที่เหมาะสมกับคอมโพเน็นต์ของบิซิเนสลอจิก แอพพลิเคชันเซิร์ฟเวอร์จะจัดหาเอ็นไวรอนเมนต์ให้กับบิซิเนสลอจิกคอมโพเน็นต์ ทั้งยังทำหน้าที่จัดการคอมโพเน็นต์ และให้บริการที่จำเป็นแก่คอมโพเน็นต์เหล่านี้ ยกตัวอย่างเช่น แอพพลิเคชันเซิร์ฟเวอร์สามารถให้บริการติดต่อฐานข้อมูลกับ บิซิเนสคอมโพเน็นต์ ซึ่งทำให้บิซิเนสคอมโพเน็นต์สามารถเก็บข้อมูลแบบถาวร (persistent) ลงไป และดึงข้อมูลขึ้นมาจากดาต้าเทียร์ได้ แอพพลิเคชันเซิร์ฟเวอร์ยังมีหน้าที่ในการจัดการคอมโพเน็นต์โดยการสร้าง อินสแทนซ์ของคอมโพเนนต์เมื่อจำเป็น

3. ดาต้าเทียร์ (Data Tier)

อาจมีลอจิกเกี่ยวกับข้อมูลซึ่งถูกเก็บไว้ในรูปของ store procedure

4.4.1 ลักษณะโดยทั่วไปของสถาปัตยกรรม N เทียร์

1) ค่าใช้จ่ายในการดีพลอยต่ำ

เนื่องจากไดรเวอร์ของระบบฐานข้อมูลนั้นถูกติดตั้งอยู่ที่ฝั่งเซิร์ฟเวอร์ แทนที่จะเป็นเครื่องของไคลเอ็นต์ ซึ่งเป็นการสะดวกที่จะทำการติดตั้งซอฟต์แวร์บนฝั่งเซิร์ฟเวอร์ในสภาพแวดล้อมที่ควบคุมได้ มากกว่าการติดตั้งบนเครื่องไคลเอ็นต์จำนวนมากมาย

2) ค่าใช้จ่ายในการเปลี่ยนแปลงระบบฐานข้อมูลต่ำ

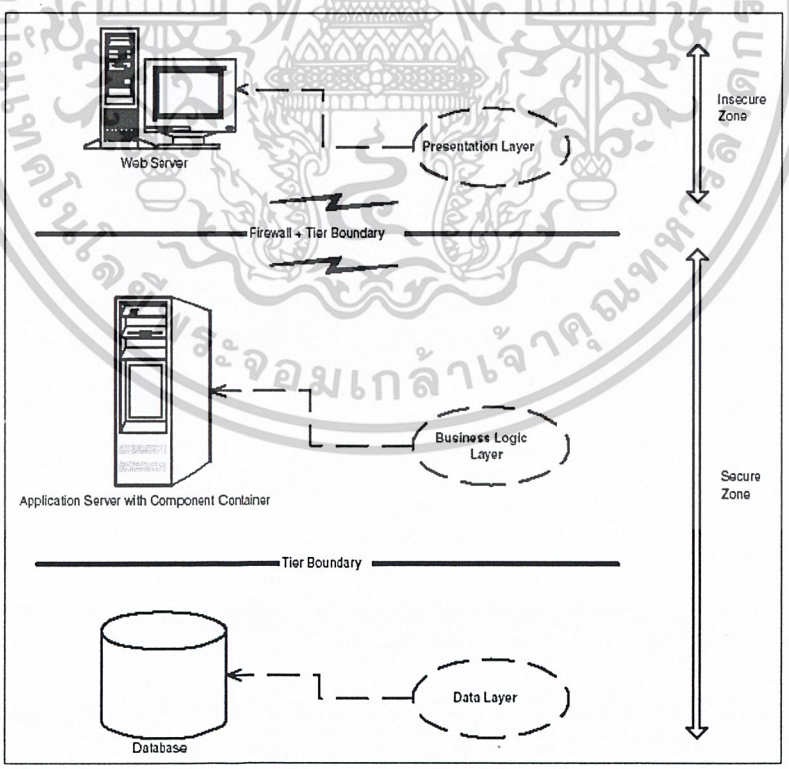
ไคลเอ็นต์จะทำการติดต่อกับฐานข้อมูลโดยผ่านทางเทียร์ซึ่งอยู่ตรงกลาง ทำให้เราสามารถเปลี่ยนแปลงโครงสร้างของฐานข้อมูล , ไดรเวอร์ของฐานข้อมูล แม้กระทั่งฐานข้อมูลโดยไม่จำเป็นต้องดีพลอยไคลเอ็นต์ทั้งหมดใหม่อีกครั้ง

3) ค่าใช้จ่ายในการเปลี่ยนแปลงบิซิเนสลอจิกต่ำ

เพราะการเปลี่ยนแปลงในบิซิเนสลอจิกเลเยอร์ไม่จำเป็นที่จะต้องมีการดีพลอยไคลเอ็นต์เทียร์ใหม่

4) ความปลอดภัย

สามารถสร้างเขตความปลอดภัยให้กับส่วนใดส่วนหนึ่งของดีพลอยเมนต์ได้ด้วยไฟล์วอลล์ ใน web-base ดีพลอยเมนต์ เราอาจไม่ต้องการให้บุคคลภายนอกเข้ามาเห็นข้อมูลในบิซิเนสลอจิกเลเยอร์ แต่อนุญาตให้บุคคลภายนอกเข้าถึงพรีเซนเทชันเลเยอร์ได้ ซึ่งสามารถแก้ปัญหานี้ได้โดยการวางไฟร์วอลล์ไว้ระหว่างส่วนพรีเซนเทชัน และบิซิเนสลอจิกเทียร์ ดังแสดงในรูปที่ 4-6

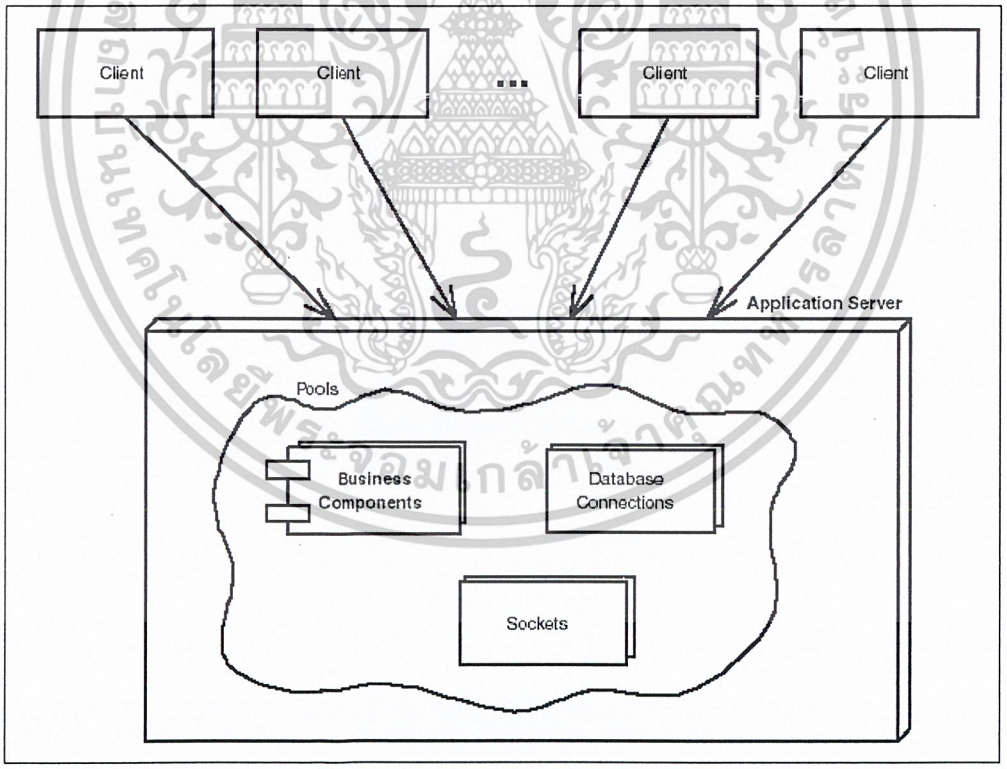


รูปที่ 4-6 การรักษาความปลอดภัยในสถาปัตยกรรมแบบ 3 เทียร์

5) ทรัพยากรสามารถถูกพูล (pool) และนำกลับมาใช้ใหม่ได้อย่างมีประสิทธิภาพ

ในสถาปัตยกรรมแบบสามเทียร์ การเชื่อมต่อไปยังทรัพยากรภายนอกสามารถถูกจัดการได้อย่างมีประสิทธิภาพ การพูลทรัพยากร (resource pooling) นั้นใช้หลักความเป็นจริงที่ว่า ไคลเอ็นต์มักจะทำงานอื่นไปพร้อม ๆ กับการใช้ทรัพยากร เช่น การสร้างส่วนติดต่อกับผู้ใช้ แทนที่บิซิเนสคอมโพเน็นต์จะทำการเชื่อมต่อ และปล่อยการติดต่อกับทรัพยากร (เช่น ฐานข้อมูล) ทรัพยากรนั้นสามารถที่จะพูล และนำไปให้บริการแก่ไคลเอ็นต์หลาย ๆ เครื่องได้ ทำให้จำนวนของการเชื่อมต่อฐานข้อมูล (database connection) นั้น จะมีจำนวนน้อยกว่าจำนวนของคอมโพเน็นต์ทั้งหมดมาก

เนื่องจากการติดต่อกับฐานข้อมูลนั้นเป็นวิธีการที่แพง (expensive) วิธีการนี้จะช่วยเพิ่ม scalability ของดีพลอยเมนต์ นอกจากนี้แล้วการเชื่อมต่อไปยังทรัพยากรนั้นไม่จำเป็นที่จะต้องถูกสร้างขึ้นใหม่เรื่อย ๆ ซึ่งจะช่วยเพิ่มประสิทธิภาพการทำงานของแอพพลิเคชัน วิธีการพูลทรัพยากรนี้สามารถใช้กับทรัพยากรประเภทอื่น ๆ นอกเหนือจากฐานข้อมูลได้อีกด้วย เช่น การเชื่อมต่อซ็อกเก็ต (socket connection) หรือเธรด (thread) จริง ๆ แล้วในสถาปัตยกรรมแบบสามเทียร์นั้น บิซิเนสคอมโพเน็นต์เองสามารถที่จะพูลและให้บริการแก่ไคลเอ็นต์หลาย ๆ เครื่องได้ การพูลคอมโพเน็นต์นั้นหมายความว่าเราไม่จำเป็นที่จะต้องมีคอมโพเน็นต์ส่วนตัวสำหรับไคลเอ็นต์แต่ละเครื่อง การพูลทรัพยากรนั้นถูกแสดงไว้ในรูปที่ 4-7



รูปที่ 4-7 การพูลทรัพยากร (Resource Pooling) ในสถาปัตยกรรมแบบสามเทียร์

6) ประสิทธิภาพถูกแบ่งเป็นส่วน ๆ

หากเทียร์ใดเทียร์หนึ่งเกิดการโอเวอร์โหลด (overload) ขึ้น เทียร์อื่น ๆ จะยังคงทำงานต่อไปได้อย่างปกติ เช่นในกรณีของเว็บดีพลอยเมนต์ ผู้ใช้สามารถที่จะเข้ามายังหน้าแรกของเว็บไซต์มาก ถึงแม้ว่าที่แอพพลิเคชันเซิร์ฟเวอร์จะเกิดการโอเวอร์โหลดก็ตาม

7) ความผิดพลาดในการทำงานถูกแบ่งเป็นส่วน ๆ

หากเกิดความผิดพลาดในการทำงานอย่างรุนแรงขึ้นที่เทียร์ใด ๆ ความผิดพลาดก็จะถูกจำกัดวงอยู่เพียงในเทียร์นั้นโดยไม่มีผลกระทบต่อเทียร์อื่น ๆ ยกตัวอย่างเช่น หากแอพพลิเคชันเซิร์ฟเวอร์เกิดความผิดพลาดในการทำงาน หรือแครช (crash) เว็บเซิร์ฟเวอร์สามารถทำการรายงานความผิดพลาดให้กับผู้ใช้ทราบได้

8) มีค่าใช้จ่ายในการบำรุงรักษาสูง

เนื่องจากเราต้องทำการดีพลอยระบบถึงสามเทียร์ขึ้นไปในทางกายภาพ ทำให้ค่าใช้จ่ายในการติดตั้งซอฟต์แวร์ , การอัพเกรดซอฟต์แวร์ , การดีพลอยใหม่ และค่าใช้จ่ายในการบำรุงรักษาระบบสูงขึ้น

4.4.2 ตัวอย่างของระบบ N เทียร์

ระบบ N เทียร์ ก็คือระบบที่บิสสิเนสลอจิกถูกแบ่งออกเป็นระบบใหญ่หลายๆระบบ ดังแสดงในรูปที่ 3-9 เช่น ระบบอาจจะแบ่งออกเป็น

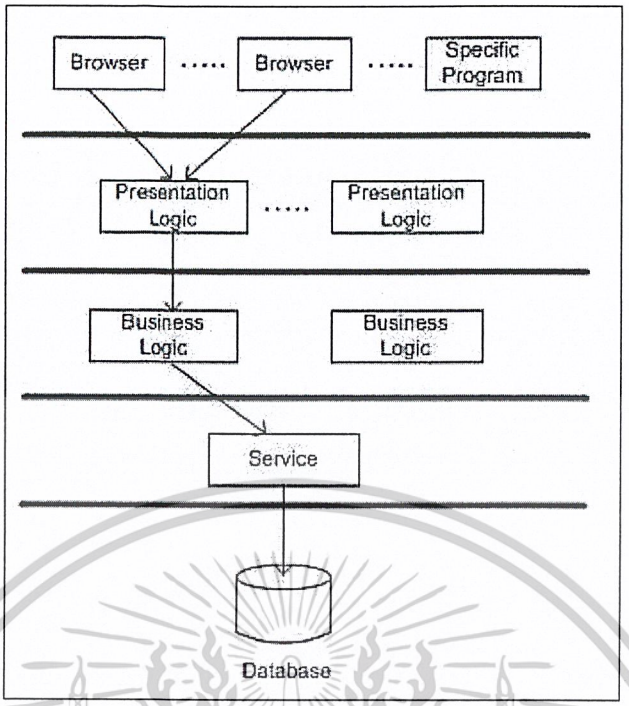
User Interface ทำหน้าที่แสดงข้อมูลให้แก่ผู้ใช้ดูและรับข้อมูลจากผู้ใช้ อาจจะเป็นโปรแกรมที่สร้างขึ้นสำหรับหน้าที่นี้โดยเฉพาะ หรือ เบราเซอร์ หรือแม้แต่อุปกรณ์สื่อสารไร้สาย

พรีเซนเทชั่นลอจิก ทำหน้าที่ควบคุมการแสดงและรับข้อมูลจากส่วน user interface โดยปกติระบบ n เทียร์หนึ่งๆ จะมีหลายพรีเซนเทชั่นลอจิก สำหรับ user interface แต่ละประเภท และสามารถเพิ่มได้เมื่อมี user interface ชนิดใหม่ๆเพิ่มขึ้น ทำให้บิสสิเนสลอจิก ไม่จำกัดอยู่บน user interface หนึ่ง

บิสสิเนสลอจิก กำหนดหน้าที่ของแอพพลิเคชั่น

เซอร์วิส เป็นระบบที่ให้บริการแก่บิสสิเนสลอจิก เช่น การให้บริการเมลล์ การให้บริการทรานแซคชั่น เป็นต้น

ดาต้า เลเยอร์ คือระบบฐานข้อมูล อาจเป็นข้อมูลทางธุรกิจ หรือเอกสารอย่างเช่น HTML และ XML



รูปที่ 4-8 ตัวอย่างของระบบ N เทียร์

บทที่ 5
พื้นฐานของ EJB (EJB Fundamental)

Enterprise JavaBeans (EJB) ได้นำหลักการในการแบ่งการทำงานเป็นส่วน ๆ (Divide-and-Conquer) มาใช้ในการพัฒนาโปรแกรมที่ประมวลผลบนฝั่งเซิร์ฟเวอร์ EJB จะแบ่งความรับผิดชอบในการพัฒนาระบบทั้งหมดออกเป็นส่วน ๆ โดยในแต่ละส่วนนั้นจะรับผิดชอบโดยผู้ที่มีความเชี่ยวชาญในด้านของตนเอง เวลาที่ใช้ในการดีพลอยระบบ EJB จึงลดลงมากเมื่อเปรียบเทียบกับการพัฒนาระบบงานทั่ว ๆ ไป

หน้าที่ที่เกี่ยวข้องกับการพัฒนาระบบด้วย EJB มีดังนี้

- Bean Provider

เป็นผู้สร้างคอมโพเน็นต์ทางธุรกิจที่สามารถนำกลับมาใช้ใหม่ได้ (Reusable) ซึ่งธุรกิจอื่น ๆ สามารถหาซื้อคอมโพเน็นต์ไปใช้ในธุรกิจของตนเองได้ bean นั้นไม่ใช่แอพพลิเคชันที่สมบูรณ์ในตัวเอง แต่เป็นคอมโพเน็นต์ซึ่งสามารถนำไปดีพลอย และประกอบกันขึ้นมาเป็นแอพพลิเคชันที่สมบูรณ์ได้

- Container Provider

เป็นผู้พัฒนาคอนเทนเนอร์ที่จะทำหน้าที่ในกาจัดการกับเอ็นไวรอนเมนต์ ในการทำงานให้กับ EJB

- Server Provider

เป็นผู้พัฒนาแอพพลิเคชันเซิร์ฟเวอร์ แอพพลิเคชันเซิร์ฟเวอร์คือเซิร์ฟเวอร์ในมิดเดิลเทียร์ ทำหน้าที่ในการจัดการและดีพลอยคอมโพเน็นต์ต่าง ๆ โดยในขณะนี้ยังไม่มีข้อแตกต่างที่เห็นได้ชัดเจนระหว่าง Container Provider และ Server Provider

- Application Assembler

เป็นผู้มีหน้าที่รับผิดชอบในการนำคอมโพเน็นต์ต่าง ๆ มาประกอบกัน และเขียนแอพพลิเคชันที่ใช้คอมโพเน็นต์เหล่านั้น โดยที่อาจจะต้องเขียนคอมโพเน็นต์บางตัวขึ้นมาเอง Application Assembler อาจเป็นบริษัทภายนอกที่รับทำงานนี้โดยเฉพาะ หรืออาจจะเป็นโปรแกรมเมอร์ภายในบริษัทเองก็ได้

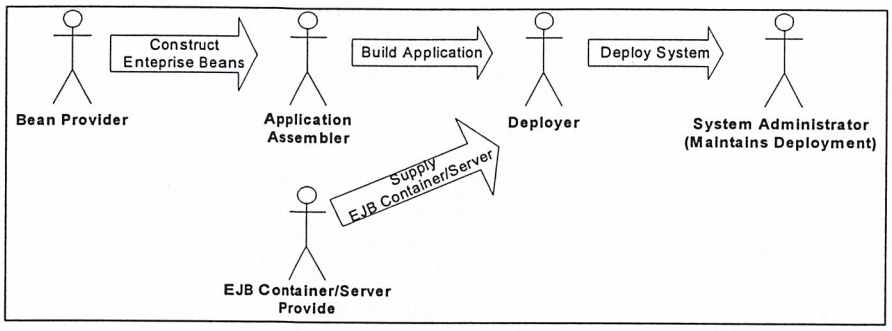
- Deployer

เป็นผู้นำเอาคอมโพเน็นต์ต่าง ๆ ที่ได้สร้างไว้แล้วมาทำการติดตั้งลงในแอพพลิเคชันเซิร์ฟเวอร์เพื่อใช้งาน

- System Administrator

มีหน้าที่ดูแลระบบที่ถูกดีพลอยเรียบร้อยแล้วให้ทำงานเป็นปกติ

ความสัมพันธ์ในการทำงานร่วมกันของแต่ละส่วน แสดงในรูปที่ 5-1



รูปที่ 5-1 ความสัมพันธ์ของแต่ละหน้าที่ในการพัฒนา EJB

5.1 EJB เซิร์ฟเวอร์ และ EJB คอนเทนเนอร์

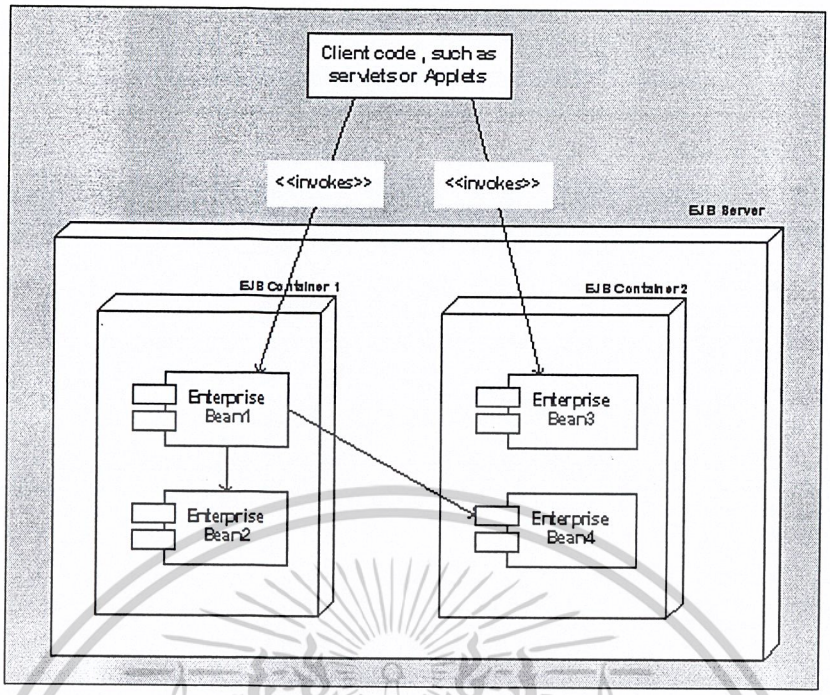
แอพพลิเคชันเซิร์ฟเวอร์จะจัดหาบริการมิดเดิลแวร์ (middleware service) ให้กับแอพพลิเคชัน เช่น ทรานแซ็กชัน , ความปลอดภัย และอื่น ๆ บริการเหล่านี้จำเป็นสำหรับแอพพลิเคชันที่ต้องการความสามารถในการเพิ่มขยาย , มีประสิทธิภาพ และมีความปลอดภัย ในการให้บริการกับผู้ใช้จำนวนมากพร้อม ๆ กัน แอพพลิเคชันเซิร์ฟเวอร์จะแบ่งเป็น 2 ส่วนคือ

5.1.1 EJB คอนเทนเนอร์

เป็นที่อยู่ที่เอนเตอร์ไพรส์บีนสามารถทำงานได้ ภายในคอนเทนเนอร์หนึ่งสามารถมีบีนจำนวนมากทำงานอยู่ คอนเทนเนอร์มีหน้าที่ในการควบคุมบีนที่รันอยู่ภายในตัวมัน

5.1.2 EJB Server

EJB เซิร์ฟเวอร์จะจัดหารันไทม์เอ็นไวรอนเมนต์ให้กับคอนเทนเนอร์ โดยภายใน EJB เซิร์ฟเวอร์สามารถมีคอนเทนเนอร์ได้มากกว่า 1 คอนเทนเนอร์ EJB เซิร์ฟเวอร์จะควบคุมทรัพยากรระบบในระดับล่าง และจัดสรรทรัพยากรให้กับคอนเทนเนอร์ ความสัมพันธ์ระหว่าง EJB เซิร์ฟเวอร์และคอนเทนเนอร์เป็นดังรูปที่ 5-2



รูปที่ 5-2 ความสัมพันธ์ระหว่าง EJB เซิร์ฟเวอร์และคอนเทนเนอร์

5.2 เอนเตอร์ไพรส์บีน

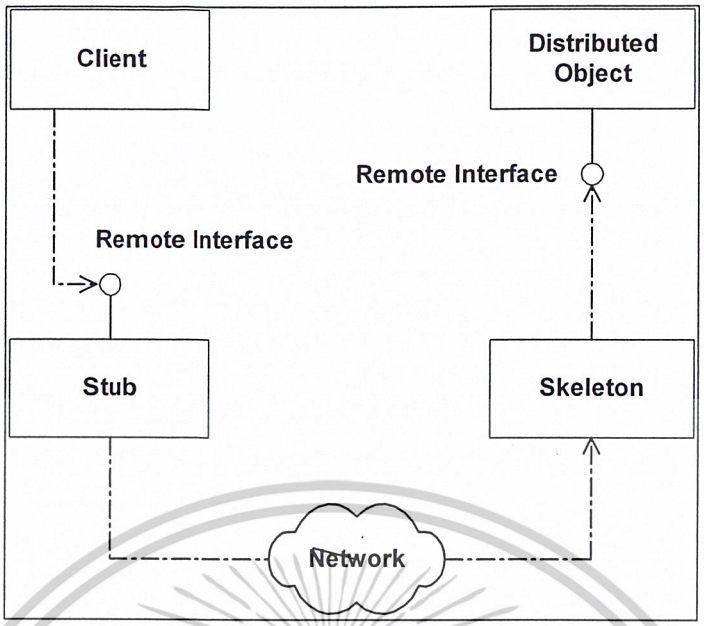
เอนเตอร์ไพรส์บีน เป็นคอมโพเน็นต์ที่ทำงานบนฝั่งเซิร์ฟเวอร์ (Server-side software component) ซึ่งสามารถนำไปดีพลอยในเอ็นไวรอนเมนต์มัลติเทียร์แบบกระจาย (Distributed multi-tier environment) ได้ ภายในหนึ่งเอนเตอร์ไพรส์บีนสามารถประกอบไปด้วยจาวาออบเจ็กต์มากกว่าหนึ่งชิ้นได้ เนื่องจากคอมโพเนนต์นั้นไม่จำเป็นจะต้องเป็นออบเจ็กต์ชิ้นเดียว

ไม่ว่าเอนเตอร์ไพรส์บีนนั้นจะมีองค์ประกอบภายในอย่างไรก็ตาม ไคลเอ็นต์ที่มาเรียกใช้เอนเตอร์ไพรส์บีนนั้นต้องติดต่อผ่านทางอินเทอร์เฟซของเอนเตอร์ไพรส์บีนเท่านั้น อินเทอร์เฟซเหล่านี้รวมทั้ง เอนเตอร์ไพรส์บีนจะต้องเป็นไปตามข้อกำหนดของ Enterprise JavaBeans ข้อกำหนดเหล่านี้จะระบุถึงเมธอดที่เป็นเมธอดบังคับจำนวนหนึ่ง ซึ่งเมธอดเหล่านี้จะทำให้ EJB คอนเทนเนอร์ควบคุมเอนเตอร์ไพรส์บีนได้ในแนวทางเดียวกัน โดยไม่ขึ้นอยู่กับคอนเทนเนอร์ที่เอนเตอร์ไพรส์บีนทำงานอยู่

ไคลเอ็นต์ของเอนเตอร์ไพรส์บีนนั้นจะเป็นอะไรก็ได้ เช่น Servlet, Applet หรือเอนเตอร์ไพรส์บีนอื่น โดยในกรณีหลัง ไคลเอ็นต์ที่ทำการร้องขอไปยังบีนจะทำให้เกิดการร้องขอไปยังบีนตัวอื่น ๆ ต่อไปเรื่อย ๆ ซึ่งเป็นแนวคิดที่ดี เนื่องจากเราสามารถแบ่งบีนที่ทำงานที่ซับซ้อนเป็นส่วนย่อย ๆ ได้

5.2.1 ออบเจ็กต์แบบกระจาย

EJB คอมโพเน็นต์มีพื้นฐานอยู่บน RMI ซึ่งเป็นเทคโนโลยีออบเจ็กต์แบบกระจาย รูปที่ 5-3 จะแสดงการเรียกออบเจ็กต์แบบกระจายของไคลเอ็นต์



รูปที่ 5-3 ออบเจ็กต์แบบกระจาย

- ไคลเอ็นต์ เรียก stub ซึ่งเป็น proxy ออบเจ็กต์ทางฝั่งไคลเอ็นต์ (client-side proxy object) stub จะซ่อนการติดต่อผ่านเน็ตเวิร์ก (network communication) จากไคลเอ็นต์ stub จะรู้ว่าจะเรียกผ่านเน็ตเวิร์ก (call over network) อย่างไรผ่านทางซ็อกเก็ต
- stub จะเรียกผ่านเน็ตเวิร์กไปยัง skeleton ซึ่งเป็น proxy ออบเจ็กต์ทางฝั่งเซิร์ฟเวอร์ (server-side proxy object) skeleton จะซ่อนการติดต่อผ่านเน็ตเวิร์กจากออบเจ็กต์แบบกระจาย skeleton จะรู้ว่าจะรับการเรียกอย่างไรผ่านทางซ็อกเก็ต
- skeleton จะเป็นตัวแทนในการเรียกออบเจ็กต์แบบกระจาย ออบเจ็กต์แบบกระจายจะทำงานของมันและจะคืนผลลัพธ์ให้กับ skeleton กลับไปยัง stub ซึ่งจะคืนให้กับไคลเอ็นต์ต่อไป

จุดสำคัญคือทั้ง stub และออบเจ็กต์แบบกระจายจะใช้อินเทอร์เฟซเดียวกัน (ที่เรียกว่ารีโมตอินเทอร์เฟซ) ไคลเอ็นต์ที่เรียกเมธอดผ่าน stub จะคิดว่าเรียกออบเจ็กต์แบบกระจายโดยตรง แต่ในความจริงแล้วไคลเอ็นต์ได้เรียก stub ที่รู้ว่าจะทำมันผ่านเน็ตเวิร์กได้อย่างไร สิ่งนี้เรียกว่า local/remote tranparency

5.2.2 ออบเจ็กต์แบบกระจายและมิดเดิลแวร์

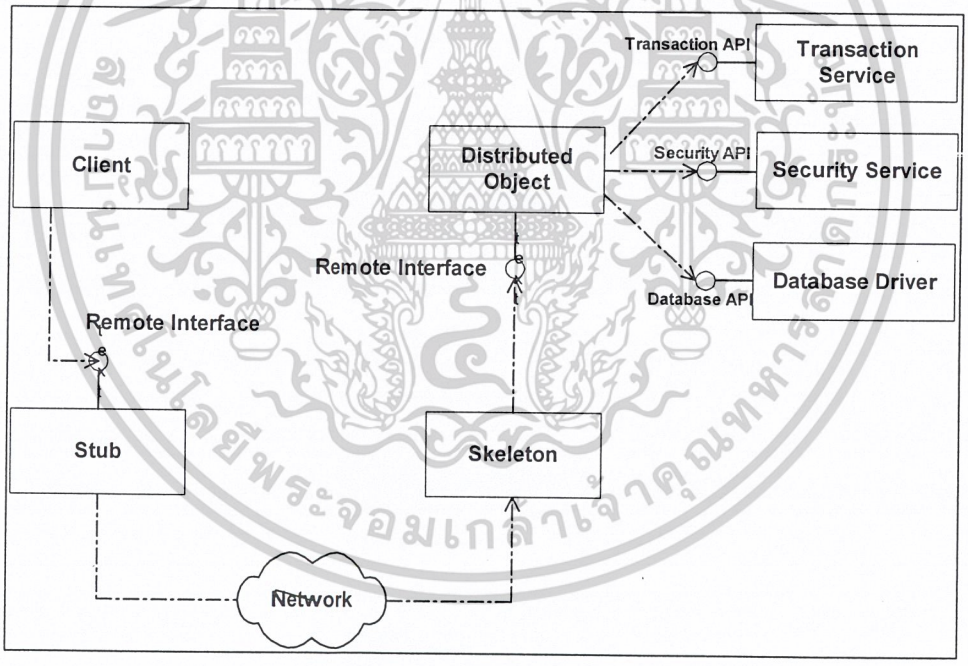
ออบเจ็กต์แบบกระจายนั้นมีประโยชน์มาก ทำให้สามารถเรียกใช้คอมโพเน็นต์บนเน็ตเวิร์กได้ แต่มันก็ยังต้องการบริการจากมิดเดิลแวร์ เช่น ทรานแซ็กชันและความปลอดภัย

5.2.2.1 Explicit Middleware

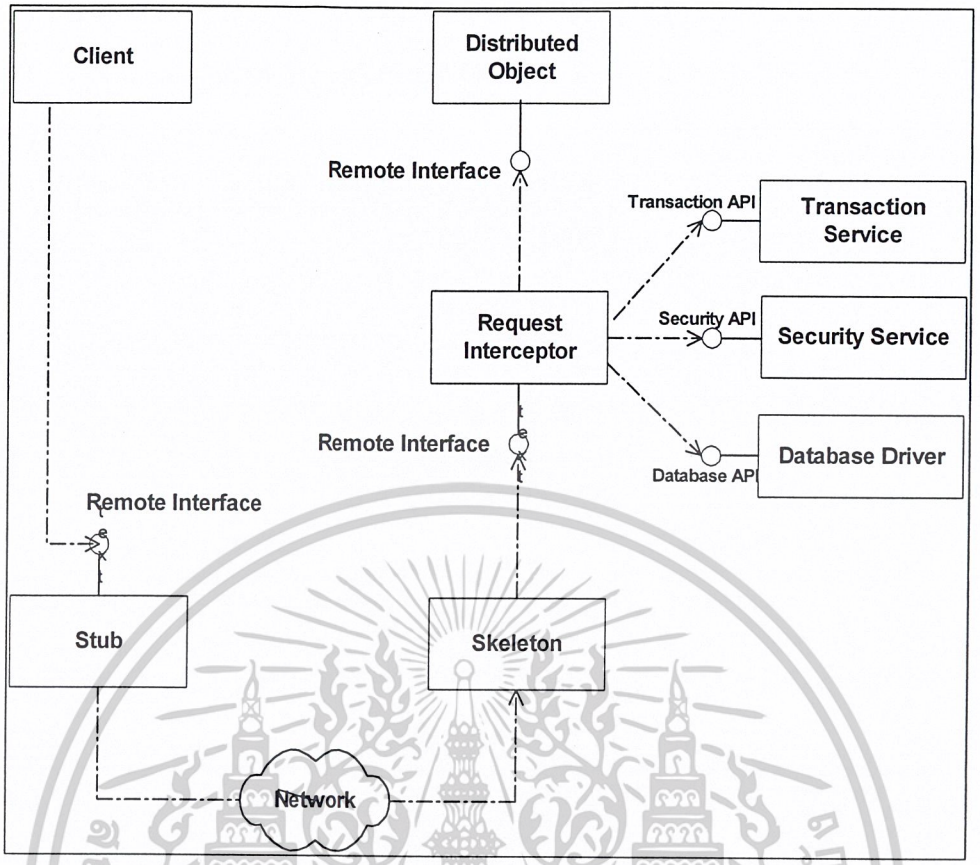
โดยปกติออบเจ็กต์แบบกระจายจะติดต่อกับมิดเดิลแวร์โดยเขียนโปรแกรมติดต่อกับ API ของมิดเดิลแวร์ เช่นหากต้องการการจัดการทางด้านทรานแซ็กชันจะต้องเขียนโค้ดติดต่อกับ transaction API ดังรูปที่ 5-4 มิดเดิลแวร์ประเภทนี้มีข้อเสียคือต้องเขียนโปรแกรมติดต่อกับ API ของมิดเดิลแวร์และหากต้องการเปลี่ยนมิดเดิลแวร์ที่ใช้ จะต้องเขียนโค้ดใหม่

5.2.2.2 Implicit Middleware

ความแตกต่างที่สำคัญของระบบในอดีต (transaction processing monitors เช่น TUXEDO หรือ CICS หรือ เทคโนโลยีออบเจ็กต์แบบกระจายแบบเดิม ๆ เช่น CORBA , DCOM และ RMI) และแบบใหม่ที่เป็นเทคโนโลยีที่มีพื้นฐานจากคอมโพเน็นต์ (EJB , CORBA Component Model และ Microsoft .NET) แบบนี้ไม่จำเป็นที่จะต้องเขียนมิดเดิลแวร์ API แต่จะใช้วิธีการประกาศ (declare) เรียกได้อีกอย่างว่า declarative middleware ซึ่งจะง่ายในการเขียนโปรแกรมและดูแลรักษา โดยใน EJB จะถูกประกาศใน deployment descriptor ซึ่งเป็นไฟล์ XML ที่กำหนดแท็กสำหรับการประกาศค่าต่าง ๆ ไว้



รูปที่ 5-4 Explicit Middleware



รูปที่ 5-5 Implicit Middleware

5.3 ชนิดของบีน (Types of Bean)

5.3.1 Session Bean

Session bean เป็นรูปแบบทางธุรกรรมในระบบธุรกิจ ลักษณะที่บ่งบอกว่าเป็น session bean คือ เป็นกริยา (Verb) เพราะว่ามันเป็นการกระทำ ซึ่งการกระทำเหล่านี้จะเป็นอะไรก็ได้ เช่น การบวกเลข, การติดต่อกับฐานข้อมูล หรือ การเรียกใช้งานเอนเตอร์ไพรส์บีนตัวอื่นๆ

เหตุผลที่มันถูกเรียกว่า session bean เนื่องจากว่ามันมีอายุเท่ากับ session ของไคลเอ็นต์ที่เรียกใช้มัน ตัวอย่างเช่น ถ้ามีไคลเอ็นต์ติดต่อเข้ามายัง session bean เพื่อสั่งสินค้า แอพพลิเคชันเซิร์ฟเวอร์จะมีหน้าที่ในการสร้างอินสแทนซ์ของคอมโพเน็นต์ session bean ขึ้นมา เมื่อไคลเอ็นต์ที่เรียกใช้ยกเลิกการติดต่อ (Disconnect) แอพพลิเคชันเซิร์ฟเวอร์สามารถทำลายอินสแทนซ์ของ session bean นั้นได้

ณ เวลาใดเวลาหนึ่ง session bean หนึ่งสามารถถูกเรียกใช้ได้จากไคลเอ็นต์เพียงหนึ่งตัวเท่านั้น นั่นคือจะไม่มีการแบ่งใช้ session bean ร่วมกันระหว่างไคลเอ็นต์ที่เข้ามาขอใช้บริการ ซึ่งเป็นจุดที่แตกต่างจาก entity bean โดย entity bean นั้นจะถูกใช้ร่วมกันหลาย ๆ ไคลเอ็นต์

EJB เซิร์ฟเวอร์มีหน้าที่จัดการและควบคุมวงจรชีวิต (Life Cycle) ของทุก bean นั่นคือ ไคลเอ็นต์จะไม่ได้ทำการสร้างอินสแทนซ์ของ bean ขึ้นมาเองโดยตรง แต่ EJB คอนเทนเนอร์จะเป็นผู้สร้างอินสแทนซ์ของ bean ขึ้นมาให้โดยอัตโนมัติ ในทำนองเดียวกัน EJB คอนเทนเนอร์จะเป็นผู้ทำลายอินสแทนซ์

ของ bean ในเวลาที่เหมาะสม ซึ่งทำให้ bean สามารถพูลและนำกลับมาใช้ใหม่กับหลาย ๆ ไคลเอ็นต์ที่ทำการร้องขอเข้ามาได้

session bean มี 2 ชนิดคือ stateful session bean และ stateless session bean

5.3.1.1 Stateful Session Bean

กระบวนการทางธุรกิจ (Business Process) บางอย่าง สามารถทำให้เสร็จสิ้นได้ในการเรียกใช้เมธอดเพียงครั้งเดียว เช่นการคำนวณราคาสินค้า หรือการตรวจสอบยอดบัตรเครดิต แต่กระบวนการทางธุรกิจบางอย่างไม่สามารถทำให้เสร็จสิ้นได้ในการเรียกใช้งานเมธอดเพียงครั้งเดียว

stateful session bean เป็น session bean ซึ่งถูกออกแบบมาเพื่อให้บริการกับกระบวนการทางธุรกิจที่ต้องมีการเรียกใช้หลายเมธอด หรือต้องมีการใช้ทรานแซ็กชันโดย stateful session bean จะเก็บสถานะต่าง ๆ ของไคลเอ็นต์ไว้ในตัวมัน ถ้าสถานะของ stateful session bean มีการเปลี่ยนแปลงในระหว่างการเรียกใช้เมธอด ค่าสถานะนั้นจะยังคงอยู่เมื่อไคลเอ็นต์เดิมทำการเรียกใช้เมธอดครั้งถัดไป

5.3.1.2 Stateless Session Bean

กระบวนการทางธุรกิจบางชนิดมีรูปแบบการร้องขอบริการไปยัง bean ด้วยการร้องขอเมธอดเพียงครั้งเดียว เนื่องจากงานของกระบวนการทางธุรกิจประเภทนี้ สามารถถูกทำให้เสร็จสิ้นได้โดยการเรียกใช้เมธอดเพียงครั้งเดียว จึงไม่จำเป็นต้องมีการคงสถานะของไคลเอ็นต์เอาไว้ในการเรียกใช้เมธอดครั้งต่อไป โดย stateless session bean นั้นจะให้บริการกับไคลเอ็นต์ที่ร้องขอเข้ามาโดยไม่แบ่งแยก และจะไม่เก็บสถานะเดิมของไคลเอ็นต์ที่ทำการติดต่อด้วยไว้

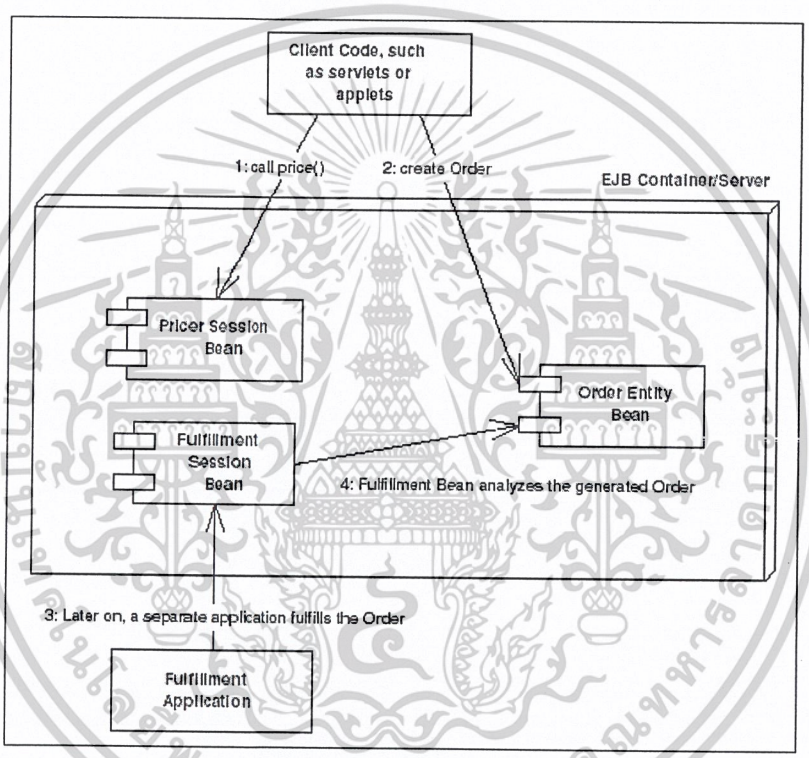
ตัวอย่างหนึ่งของ stateless session bean ได้แก่ คอมโพเน็นต์ที่ทำหน้าที่ตรวจสอบความถูกต้องของหมายเลขบัตรเครดิต bean ที่ทำหน้าที่นี้ จะรับข้อมูลหมายเลขบัตรเครดิต , วันหมดอายุ , ชื่อผู้ถือบัตร และจำนวนเงินเข้ามา หลังจากนั้น bean จะส่งข้อมูลตอบกลับไปว่า ข้อมูลนี้ถูกต้องหรือไม่โดยประมวลผลจากข้อมูลที่รับเข้ามา

5.3.2 Entity Bean

Entity bean เป็นรูปแบบของข้อมูลในระบบธุรกิจ ลักษณะที่บ่งบอกว่าเป็น entity bean คือเป็นคำนาม (noun) เพราะว่ามันคือหน่วยของข้อมูล ซึ่งก็คือออบเจ็กต์ของจาวาในการดึงข้อมูลจากฐานข้อมูลนั่นเอง ตัวอย่างเช่น ข้อมูลสินค้า, ข้อมูลการสั่งซื้อ, ข้อมูลพนักงาน, ข้อมูลในคลังสินค้า ปกติแล้ว session bean จะเป็นตัวควบคุม entity bean เพื่อให้บรรลุเป้าหมายในทางธุรกิจ เช่น ส่วนควบคุมคลังสินค้า (session bean) ซึ่งจะจัดการเกี่ยวกับส่วนคลังสินค้า (entity bean) ดังตัวอย่างที่แสดงในตาราง

SESSION BEAN	ENTITY BEAN
Bank teller	Bank account
Credit card authorizer	Credit card
DNA sequencer	DNA strand
Order entry system	Order, Line item
Catalog engine	Product
Auction broker	Bid, Item
Purchase order Approval router	Purchase order

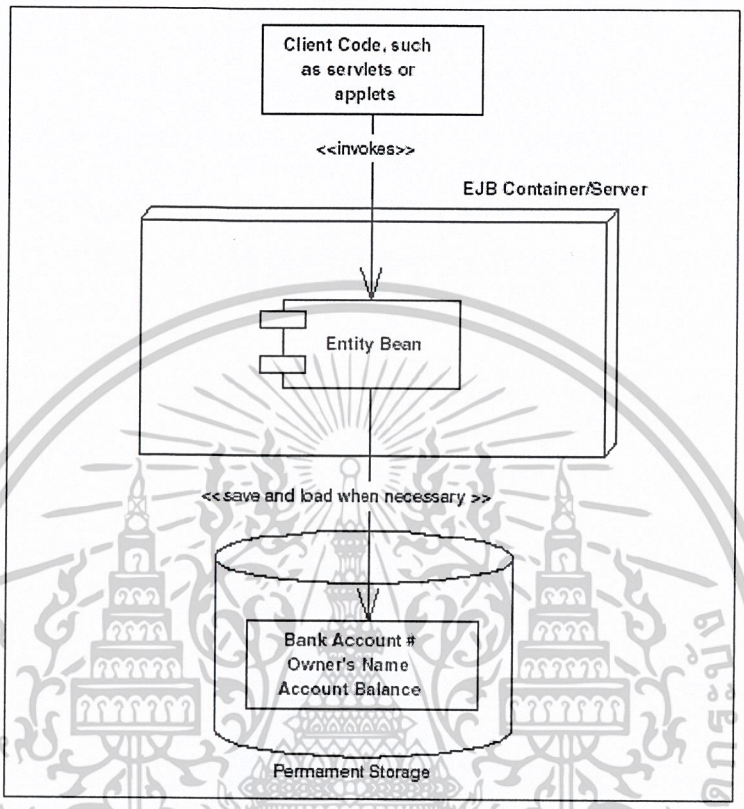
รูปที่ 5-6 Session bean เรียกใช้งาน Entity bean



รูปที่ 5-7 การนำ Session Bean และ Entity Bean มาใช้ร่วมกันในแอพลิเคชันการคิดราคาของสินค้าที่สั่งซื้อ

ประโยชน์ของ entity bean คือ มันจะเป็นมุมมอง (View) แบบออบเจ็กต์ในหน่วยความจำไปยังแหล่งเก็บข้อมูล เราสามารถอ่านเซตของข้อมูลจากฐานข้อมูลมาไว้ใน entity bean ซึ่งอยู่ในหน่วยความจำ และเราสามารถจัดการกับ entity bean ซึ่งอยู่ในหน่วยความจำนี้ได้โดยการเรียกใช้เมธอดของมัน โดยข้อมูลจริง ๆ ซึ่งอยู่ในฐานข้อมูลจะถูกเปลี่ยนแปลงตามไปด้วยโดยอัตโนมัติ

entity bean จะช่วยให้เราใช้ความสามารถของข้อมูลซึ่งมีความถาวร (Persistant) และยังมีคุณสมบัติ encapsulation ในขณะเดียวกัน entity bean จะเป็นเลเยอร์ของการเรียกใช้ข้อมูล (Data Access Logic) ในสถาปัตยกรรมแบบมัลติเทียร์ ดังแสดงในรูปที่ 5-8



รูปที่ 5-8 Entity Bean เป็นมุมมองแบบออบเจ็กต์ไปยังแหล่งเก็บข้อมูล

entity bean สามารถแบ่งออกได้เป็นอีกสองประเภท ได้แก่ entity bean แบบ bean-managed persistent และแบบ container-managed persistent

5.3.2.1 Bean-managed Persistent Entity Bean

จากที่กล่าวมาแล้ว entity bean เป็นคอมโพเน็นต์ที่มีคุณสมบัติ persistent เพราะว่าสถานะของมันนั้น จะถูกบันทึกลงในแหล่งบันทึกข้อมูล (Secondary Storage) เช่นในฐานข้อมูลเชิงสัมพันธ์ ตัวอย่างเช่น ด้วยเทคโนโลยี Object-relational Mapping เราสามารถแมปออบเจ็กต์ที่อยู่ในหน่วยความจำไปเป็นเรคอร์ดในฐานข้อมูลเชิงสัมพันธ์ซึ่งมีคุณสมบัติ persistent ได้ และเราสามารถนำข้อมูลนี้กลับมาใช้ได้ทุกเมื่อโดยการดึงข้อมูลในฐานข้อมูลกลับมาสร้างเป็นออบเจ็กต์ในหน่วยความจำอีกครั้ง

bean-managed persistent entity bean เป็น entity bean ที่ต้องทำการเก็บข้อมูลต่าง ๆ แบบ persistent ด้วยตัวเอง นั่นคือ ผู้พัฒนาคอมโพเน็นต์จะต้องเขียนโค้ดในการแปลงฟิลด์ต่าง ๆ ของ

entity bean ซึ่งอยู่ในหน่วยความจำให้กลายเป็นข้อมูลซึ่งเก็บอยู่ในแหล่งบันทึกข้อมูลเช่นฐานข้อมูลเชิงสัมพันธ์ ซึ่งจะต้องเขียนโค้ดสำหรับการเขียน การอ่าน และการค้นหาข้อมูลเข้าไปเป็นส่วนหนึ่งของ entity bean โดยอาจทำผ่านทาง API แบบ persistent ต่าง ๆ เช่น JDBC หรือ SQL/J

5.3.2.2 Container-managed Persistent Entity Bean

Enterprise JavaBeans ช่วยให้นักพัฒนาสามารถพัฒนา entity bean ได้อย่างสะดวกมากขึ้น โดยไม่ต้องเสียเวลายุ่งเกี่ยวกับโค้ดในการทำ persistent ให้กับข้อมูล ตามข้อกำหนดของ EJB 1.1 คอนเทนเนอร์จะต้องสนับสนุนการ persistent ข้อมูลโดยอัตโนมัติสำหรับ entity bean

คอนเทนเนอร์จะทำฟังก์ชันในการเข้าถึงข้อมูลโดยอัตโนมัติ รวมถึงการเก็บข้อมูล การอ่านข้อมูล และการค้นหาข้อมูลของคอมโพเน็นต์ โดยที่ผู้พัฒนาไม่ต้องเขียนโค้ดในการติดต่อกับฐานข้อมูลด้วยตัวเองซึ่งจะช่วยประหยัดเวลาในขั้นตอนการพัฒนาลงไปได้มาก สิ่งที่ผู้พัฒนาต้องทำก็คือการระบุว่าต้องการให้ฟิลด์ใดบ้างที่ต้อง persistent หลังจากนั้น คอนเทนเนอร์ก็จะดูแลการ persistent ข้อมูลให้ตามที่กำหนด

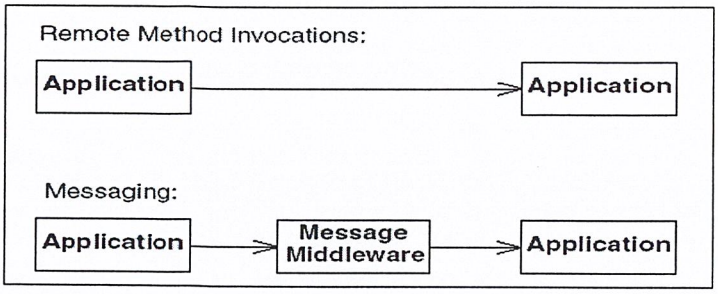
ด้วยวิธีการดังกล่าว จะทำให้เราได้โค้ดที่ไม่ขึ้นอยู่กับระบบฐานข้อมูล ทำให้เราสามารถเปลี่ยนแปลงแหล่งเก็บข้อมูลได้อย่างอิสระ เนื่องจากใน Container-managed ไม่มีการเขียนโค้ดติดต่อกับ API ของระบบฐานข้อมูล

5.3.3 Message-Driven Bean

Message-Driven Bean จะเหมือนกับ session bean ตรงที่เป็นการกระทำเหมือนกัน แต่จะต่างกันตรงที่เราสามารถเรียกใช้งาน Message-Driven Bean โดยการส่งเมสเสจ (Message) ไปเรียกใช้งานบีนตัวนั้นเท่านั้น ตัวอย่างเช่น บีนที่รวมการทำงานโดยการรอรับเมสเสจระหว่าง คลังสินค้า, การยืนยันตัวผู้ใช้บัตรเครดิต, หรือเมสเสจจากลำดับการทำงานต่างๆ Message-Driven Bean สามารถเรียกใช้งานบีนอื่นๆ ได้เช่นกัน

เป็นอีกแนวทางของรีโมตเมธอด โดยใช้เมสเซจซึ่งเป็นการติดต่อแบบ asynchronous ภายหลังจากการส่งเมสเซจออกไป ผู้สร้างเมสเซจสามารถที่จะประมวลผลงานอื่น ๆ ได้ ภายหลังที่ผู้รับเมสเซจประมวลผลเสร็จก็อาจกำหนดให้ส่งผลลัพธ์กลับมาให้กับผู้สร้างเมสเซจได้

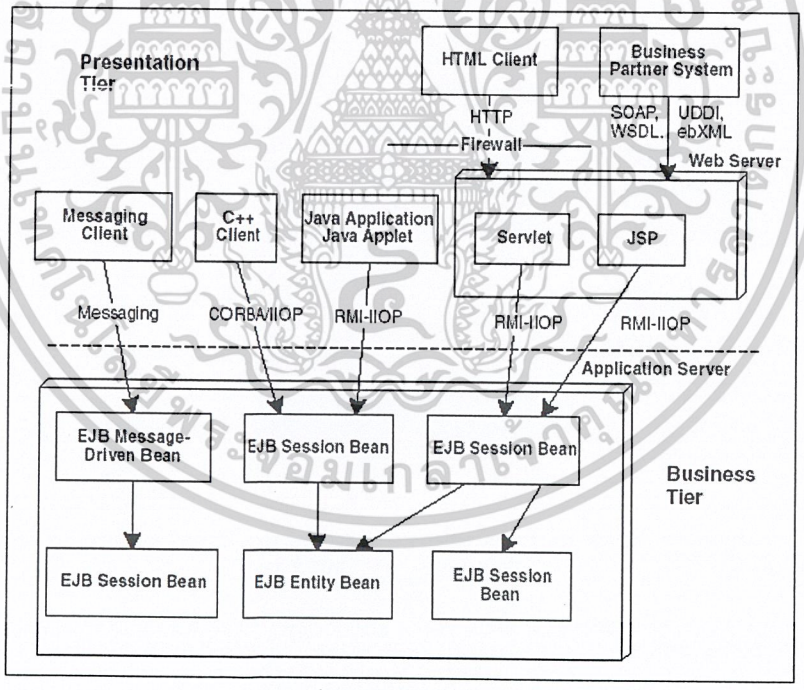
โครงสร้างพื้นฐานที่รองรับการใช้เมสเซจนี้จะเรียกว่า Message-oriented middleware (MOM) มีผลิตภัณฑ์หลายชนิดที่ใช้โครงสร้างพื้นฐานของ MOM เช่น IBM MQSeries , BEA Tuxedo/Q และ Microsoft MSMQ เป็นต้น



รูปที่ 5-9 เปรียบเทียบรีโมตเมธอดกับเมสเซจ

ผู้พัฒนาโปรแกรมบางคนอาจจะใช้งานบีนแต่ละประเภทไปอย่างผิดวิธีโดยไม่เจตนา แต่เพื่อให้การทำงานสามารถทำงานได้มีประสิทธิภาพเพิ่มขึ้นจะมีวิธีการบ่งบอกอยู่ ว่าบีนชนิดนี้ใช้ทำอะไร โดยการรวม session bean ไปกับรูปแบบทางธุรกิจ(business model) จะมีกระบวนการในการกระจายบีนนี้ออกไปเป็นสถาปัตยกรรมแบบหลายเทียร์ การรวม entity bean ไปในรายละเอียดของ EJB เลย ซึ่งเป็นก้าวแรกของคุณสมบัติ persistent, การกระจายออบเจกต์ซึ่งจะถูกใช้โดยกระบวนการทางบิสสิเนสส่วน message-driven bean นั้นสามารถใช้งานเมสเสจในการเข้าถึงระบบ EJB ต่างๆได้

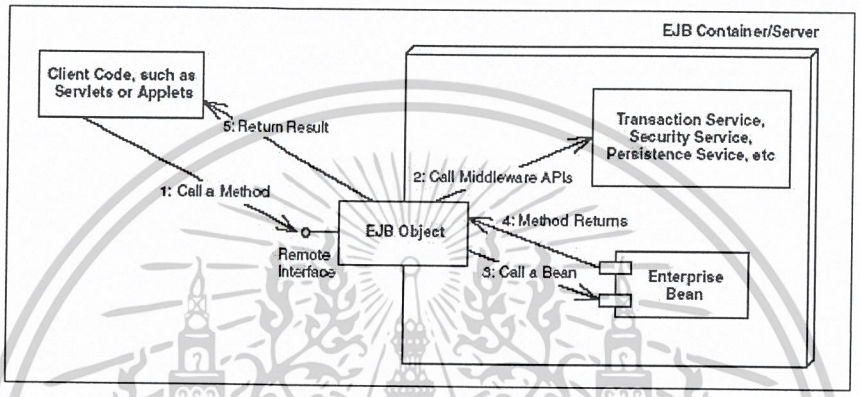
ตัวอย่างการใช้งานของไคลเอนต์ต่อระบบ EJB คอมโพเนนต์



รูปที่ 5-10 ไคลเอนต์ทำการติดต่อกับระบบ EJB คอมโพเนนต์

5.4 EJB Object

เนื่องจาก EJB คอนเทนเนอร์ เหมือนกับเป็นเลเยอร์ที่ทำให้ติดต่อกับไม่ได้โดยตรงระหว่าง โค้ดของไคลเอนต์กับบีนในระบบ เลเยอร์นี้ เหมือนกับเป็น network-aware ออบเจกต์นึง ซึ่งถูกเรียกว่า EJB object EJB object เหมือนกับเป็นตัวกลางที่รู้เกียวกับ ระบบเครือข่าย ทรานแซคชั่น ความปลอดภัย และอื่นๆ ซึ่ง EJB คอนเทนเนอร์จะต้องเรียกใช้งานก่อนที่จะไปให้บริการเมทธอดนั้นๆโดยอินสแทนซ์ของบีน EJB object จะรองรับการทำงานทุกๆกระบวนการทางบิสสิเนส ซึ่งจะมีบีนของมันมารับอีกทีหนึ่ง EJB object เป็นตัวแทนของทุกๆไคลเอนต์รีเควสท์ที่จะใช้งานบีน ดังรูป



รูปที่ 5-11 EJB object

5.5 Remote Interface

Interface นี้จะสร้างมาจากบิสสิเนสลอจิกทุกๆตัว ซึ่งเกี่ยวข้องโดยตรงกับบีน โดย interface นี้จะถูกเรียกว่า Remote Interface

Remote Interface จะต้องยินยอมโดยกฎพิเศษที่ถูกระบุอยู่ในรายละเอียดของ EJB เช่น Remote Interface จะต้อง deriveมาจากInterface พื้นฐานซึ่งได้มีการสร้างไว้โดยSUN ซึ่งก็คือ javax.ejb.EJBObject

```
public interface javax.ejb.EJBObject
extends java.rmi.Remote
{
    public javax.ejb.EJBHome getEJBHome()
        throws java.rmi.RemoteException;

    public java.lang.Object getPrimaryKey()
        throws java.rmi.RemoteException;

    public void remove()
        throws java.rmi.RemoteException,
        javax.ejb.RemoveException;

    public javax.ejb.Handle getHandle()
        throws java.rmi.RemoteException;

    public boolean isIdentical(javax.ejb.EJBObject)
        throws java.rmi.RemoteException;
}
```

รูปที่ 5-12 ตัวอย่างส่วนหนึ่งของ javax.ejb.EJBObject interface

ไคลเอนต์โค้ดที่เรียกใช้งานบีน จะต้องเรียกใช้งาน เมทธอดใน javax.ejb.EJBObject โดยที่ไคลเอนต์โค้ดนี่จะต้องเป็นโค้ดที่ทำงานในตัวเองได้, applets, servlets หรืออย่างอื่นแม้แต่ enterprise bean อื่นๆก็ได้

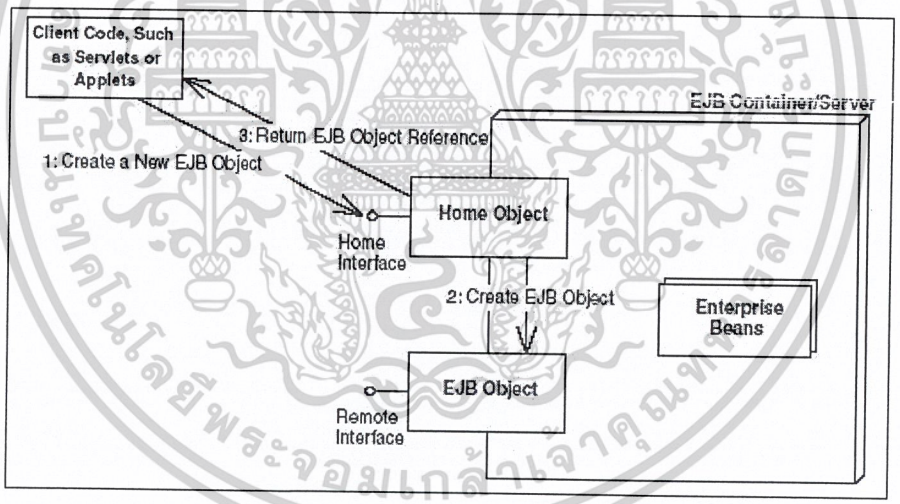
5.6 The Home Object

ไคลเอนต์ไม่สามารถสร้างอินสแทนต์ของ EJB Object ได้โดยตรง ดังนั้นในการเรียกใช้งาน EJB Object นั้น จะต้องไปถามมาจาก EJB object factory ซึ่ง factory นี้จะรับหน้าที่ในการสร้างอินสแทนซ์ (และทำงายอินสแทนซ์)ของ EJB Object ให้ ขั้นตอนการทำงานก็คือ

- สร้าง Ejb object
- ค้นหา Ejb object ที่มีอยู่ (จาก entity bean)
- ทำงาย Ejb object

5.7 Home Interface

Home Interface เป็นตัวระบุเมทธอดสำหรับสร้าง ทำลายและค้นหา EJB object โดย Home Object ของคอนเทนเนอร์จะต้องอิมพลีเมนต์ interface นี้ ดังรูป

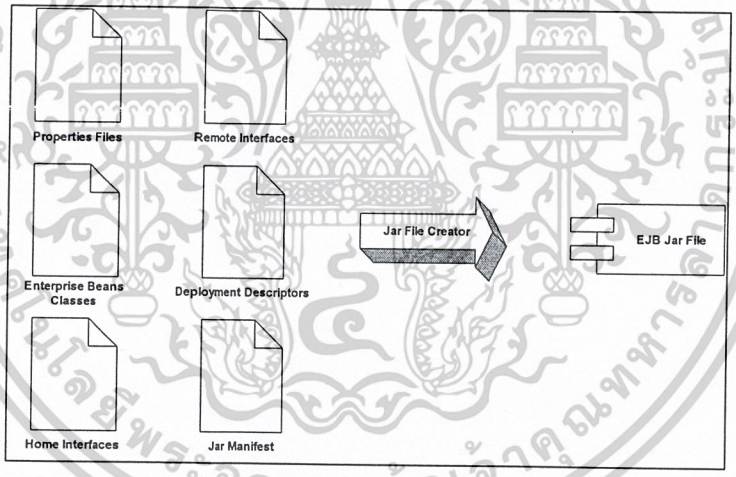


รูปที่ 5-13 Home interfaces and objects

EJB จะเรียกใช้งานเมทธอดบางตัวซึ่ง home interface จะต้องสนับสนุน โดยเมทธอดเหล่านี้จะถูกประกาศอยู่ใน javax.ejb.EJBHome interface ซึ่งเป็น interface ที่home interface ของเราจะต้อง extend

```
public interface javax.ejb.EJBHome extends java.rmi.Remote
{
public EJBMetaData getEJBMetaData()
throws java.rmi.RemoteException;
public javax.ejb.HomeHandle getHomeHandle()
throws java.rmi.RemoteException;
public void remove(javax.ejb.Handle handle)
throws java.rmi.RemoteException,
javax.ejb.RemoveException;
public void remove(Object primaryKey)
throws java.rmi.RemoteException,
javax.ejb.RemoveException;
}
```

รูปที่ 5-14 javax.ejb.EJBHome interface



รูปที่ 5-15 ขั้นตอนการสร้างไฟล์ Ejb-jar

หลังจากสร้างไฟล์ ejb-jar แล้ว ถือว่าเอนเตอร์ไพรส์บีนสมบูรณ์ สามารถนำไปดีพลอยในแอพพลิเคชันเซิร์ฟเวอร์ได้ เมื่อนำไปดีพลอย เครื่องมือของผู้พัฒนาคอนเทนเนอร์จะทำหน้าที่ขยายไฟล์ ejb-jar และนำข้อมูลต่าง ๆ ในไฟล์มาใช้งาน

บทที่ 6

Session Bean

session bean เป็น enterprise bean ตัวแทนงานที่ประมวลผลให้กับไคลเอ็นต์หนึ่ง ๆ มีจุดประสงค์เพื่อให้เป็นตัวแทนของบิซิเนสโพรเซส อาจเป็นงานเกี่ยวข้องกับลอจิก , อัลกอริทึม , หรือเวิร์คโฟล์ว ตัวอย่างของบิซิเนสโพรเซส เช่น การตรวจสอบบัตรเครดิต การรับรายการสั่งซื้อสินค้า การคำนวณต่าง ๆ หรือการติดตามยอดสินค้าคงคลัง บิซิเนสโพรเซสเหล่านี้สามารถแทนได้ด้วย session bean

การติดต่อระหว่างไคลเอนต์ กับ เซิร์ฟเวอร์ในช่วงเวลาหนึ่ง ว่า หนึ่งเซสชั่น โดยใน 1 เซสชั่น นั้นจะประกอบด้วย 1 request และ response หรือมากกว่านั้น

session bean จะไม่คงสภาวะของเสตทข้ามช่วงชีวิต คือ ข้อมูลจะไม่ถูกเก็บลงฐานข้อมูล เมื่อไคลเอนต์สิ้นสุดการทำงานลง session bean ก็จะสิ้นสุดการทำงานลงด้วยเช่นกัน และ จะไม่มีความเกี่ยวข้องกับไคลเอนต์นั้นอีก

6.1 ลักษณะของ Session Bean

6.1.1 ช่วงชีวิตของ Session Bean (Session Bean Lifetime)

ข้อแตกต่างหลักระหว่าง session bean กับ entity bean คือช่วงชีวิตของมัน session bean เป็นคอมโพเน็นต์ที่มีช่วงชีวิตสั้น โดยมีช่วงชีวิตเท่ากับ session ของไคลเอ็นต์เท่านั้น เป็นที่มาของชื่อ session bean ระยะเวลาของเซสชันของไคลเอ็นต์นั้น อาจยาวนานเท่ากับระยะเวลาที่เว็บบราวเซอร์เปิดอยู่ , ระยะเวลาที่ Applet ทำงาน , ระยะเวลาที่แอพพลิเคชันแบบ stand-alone กำลังทำงาน หรือระยะเวลาที่ bean อื่น ๆ เข้ามาเรียกใช้ bean นั้นก็ได้

โดยทั่วไประยะเวลาของ session ของไคลเอ็นต์จะเป็นตัวกำหนดช่วงชีวิตของ session bean โดย EJB คอนเทนเนอร์จะทำลาย session bean เมื่อไคลเอ็นต์จบการทำงานไป ถ้าหากแอพพลิเคชัน เซิร์ฟเวอร์หรือเครื่องคอมพิวเตอร์ที่ session bean อยู่เกิดแครช session bean จะถูกทำลายไปด้วย เพราะ session bean เป็นออบเจ็กต์ที่อยู่ในหน่วยความจำ ต้องอาศัยสภาวะแวดล้อมในการทำงาน

ในทางกลับกัน entity bean มีช่วงชีวิตยาวนาน เนื่องจากเป็นออบเจ็กต์ที่ persistent ซึ่งถือเป็นส่วนหนึ่งของแหล่งบันทึกข้อมูลถาวร เช่น ฐานข้อมูล โดย entity bean ถูกสร้างขึ้นในหน่วยความจำจากข้อมูลในฐานข้อมูล

session bean ไม่มีคุณลักษณะ persistent ไม่มีการบันทึกลงในแหล่งเก็บข้อมูลเหมือนกับ entity bean สามารถทำงานเกี่ยวกับฐานข้อมูลได้ แต่ตัว session bean เองไม่ใช่ออบเจ็กต์ที่ persistent

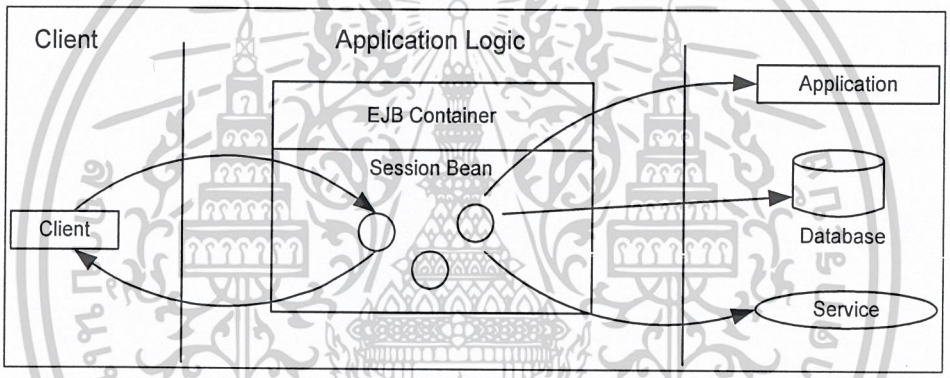
6.1.2 Session Bean ที่มีการเก็บสถานะและไม่มีการเก็บสถานะ

session bean มี 2 ชนิดคือ stateless session bean และ stateful session bean ข้อแตกต่างระหว่าง bean สองชนิดนี้คือ stateless session bean จะไม่มีการเก็บข้อมูลสถานะของไคลเอ็นต์ไว้ ในขณะที่

stateful session bean จะมีการเก็บสถานะของไคลเอ็นต์ไว้ เช่น ในระบบอี-คอมเมิร์ช รถเข็นแบบออนไลน์จะต้องเก็บสถานะของสินค้าที่สั่งไว้ เป็นต้น

6.1.3 เมธอดทั้งหมดของ Session Bean จะต้องถูก Serialize

เมื่อเรียกเมธอดกับอินสแทนซ์ของ session bean EJB คอนเทนเนอร์จะรับประกันว่าจะไม่มีไคลเอ็นต์อื่นใช้อินสแทนซ์นั้น คอนเทนเนอร์จะป้องกันอินสแทนซ์ของ bean นั้นไว้และให้ไคลเอ็นต์ที่ทำงานอยู่พร้อม ๆ กันไปใช้อินสแทนซ์อื่นหรือคอยจนกว่าอินสแทนซ์ที่มีการใช้งานจะทำเสร็จแล้ว และถ้าหลาย ๆ ไคลเอ็นต์มีการเรียกใช้เมธอดกับ session bean พร้อม ๆ กัน การเรียกนั้นจะถูก serialize หรือทำใน lock-step ซึ่งหมายความว่าคอนเทนเนอร์จะจัดการลำดับการเรียกให้โดยอัตโนมัติ ทำให้อินสแทนซ์หนึ่งจะถูกใช้งานโดยไคลเอ็นต์เดียวเท่านั้น และเนื่องจากการร้องขอของไคลเอ็นต์จะถูก serialize ทำให้เราไม่ต้องเขียนโค้ดให้ bean มีคุณสมบัติ thread-safe จะมีเพียงเธรดเดียวของไคลเอ็นต์ที่จะทำงานกับ bean ในเวลาหนึ่ง ๆ



รูปที่ 6-1 การทำงานของ session bean

โดยทั่วไปแล้ว จะใช้ session bean เมื่อ

- มีไคลเอนต์เพียงตัวเดียวที่ใช้งานบีน ในช่วงเวลาหนึ่งๆ
- สภาวะ (state) ของบีนไม่คงอยู่อย่างต่อเนื่อง โดยสภาวะของบีนจะคงอยู่แค่ช่วงเวลาสั้นๆ

6.2 ชนิดของ Session Bean

6.2.1 Stateless Session Bean

6.2.1.1 ลักษณะของ Stateless Session Bean

6.2.1.1.1 ไม่มีการเก็บสถานะ

stateless session bean ไม่เก็บสถานะการทำงานกับไคลเอ็นต์ใด ๆ ไว้ แม้จะสามารถเก็บสถานะภายในได้ แต่ไม่ได้เป็นสถานะเฉพาะสำหรับไคลเอ็นต์ตัวใดตัวหนึ่ง ในมุมมองของไคลเอ็นต์ stateless session bean ทุกตัวเหมือนกันหมด โดยไคลเอ็นต์ไม่สามารถแยกความ

แตกต่างได้ ในการใช้งานไคลเอ็นต์จะต้องส่งข้อมูลของไคลเอ็นต์ทั้งหมดที่ bean ต้องใช้ในการทำงานไปเป็นพารามิเตอร์ให้กับบิซิเนสเมธอด หรือ bean อาจดึงข้อมูลที่จำเป็นมาจากแหล่งภายนอก เช่นฐานข้อมูลก็ได้

6.2.1.1.2 สามารถกำหนดค่าเริ่มต้นให้กับ Stateless Session Bean ได้เพียงวิธีเดียว

session bean ถูกสร้างและกำหนดค่าเริ่มต้นจากเมธอด ejbCreate() แต่เนื่องจาก stateless session bean ไม่คงค่าสถานะเอาไว้ในการเรียกเมธอดแต่ละครั้ง ดังนั้นจึงไม่สามารถคงค่าใด ๆ ที่ไคลเอ็นต์ส่งเป็นพารามิเตอร์ให้กับเมธอด ejbCreate() ได้ stateless session bean ไม่จำเป็นที่จะต้องรองรับการเรียกใช้เมธอด ejbCreate() ได้หลายรูปแบบ เพราะในการเรียกใช้อินสแทนซ์ของ bean ครั้งต่อ ๆ ไป bean จะไม่มีข้อมูลของการเรียกใช้เมธอด ejbCreate() ในครั้งก่อนหน้าอยู่เลย

จากเหตุผลดังกล่าว stateless session bean จึงมีเมธอด ejbCreate() ได้เพียงตัวเดียว ที่ไม่รับค่าพารามิเตอร์ใด ๆ และใน home object จะมีเพียงเมธอด create() เข้าคู่กันซึ่งไม่รับพารามิเตอร์ใด ๆ เช่นกัน

6.2.1.1.3 คอนเทนเนอร์สามารถพูล และนำ Stateless Session Bean กลับมาใช้ใหม่ได้

เนื่องจากเมธอด ejbCreate() ของ stateless session bean ไม่รับพารามิเตอร์ใด ๆ ไคลเอ็นต์จึงไม่มีการส่งข้อมูลเพื่อเริ่มต้นการทำงานของอินสแทนซ์ของ bean ดังนั้นคอนเทนเนอร์สามารถสร้างอินสแทนซ์ของ stateless session bean ไว้ก่อนที่ไคลเอ็นต์จะติดต่อเข้ามาได้ เมื่อไคลเอ็นต์เรียกใช้งานเมธอดคอนเทนเนอร์ก็ดึงอินสแทนซ์จากในพูลไปให้บริการกับไคลเอ็นต์แล้วนำอินสแทนซ์นั้นกลับเข้าไปเก็บไว้ในพูลเช่นเดิมเมื่อไคลเอ็นต์เลิกใช้งาน ทำให้คอนเทนเนอร์สามารถนำอินสแทนซ์ของ bean มาให้บริการกับไคลเอ็นต์ได้อย่างทันที

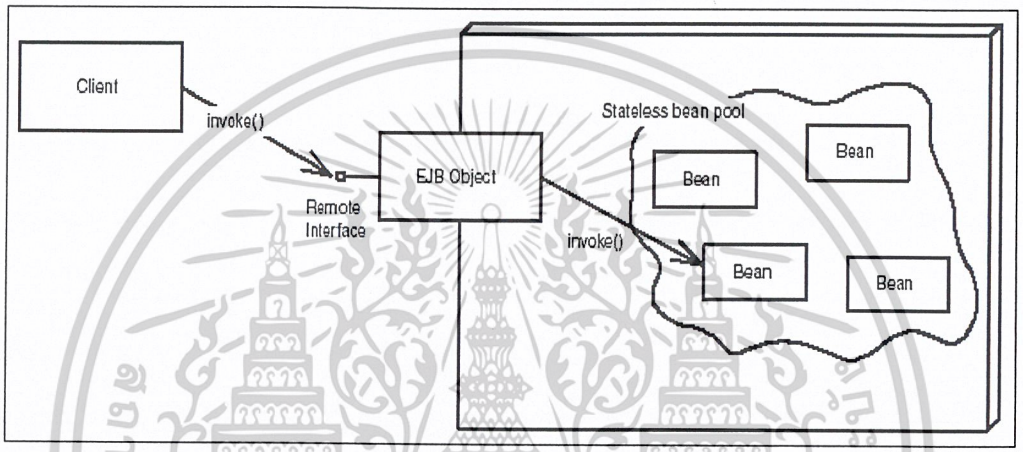
ผลที่ได้คือในแต่ละครั้งที่มีการร้องขอเข้ามาจากไคลเอ็นต์ อินสแทนซ์ของ session bean ใด ๆ สามารถให้บริการก็ได้ เนื่องจาก stateless session bean เก็บสถานะการทำงานระหว่างการเรียกใช้งานเมธอดในแต่ละครั้งเท่านั้น และไม่เก็บสถานะไคลเอ็นต์เอาไว้อีกหลังจากการเรียกใช้งานเมธอดเสร็จสิ้น stateless session bean แต่ละตัวจะอยู่ในสถานะเดียวกันเสมอหลังจากการเรียกใช้งานเมธอดในแต่ละครั้ง ดังนั้น คอนเทนเนอร์สามารถนำ stateless session bean ใด ๆ ไปให้บริการกับไคลเอ็นต์ได้ในทุก ๆ การเรียกใช้เมธอดแต่ละครั้ง stateless session bean หลายๆ ตัวสามารถให้บริการการเรียกใช้เมธอดแต่ละครั้งจากไคลเอ็นต์เดียวกันได้ ซึ่งการอิมพลีเมนต์วิธีการเลือก stateless session bean ให้กับไคลเอ็นต์จะแตกต่างกันไปตามคอนเทนเนอร์แต่ละผลิตภัณฑ์

ประโยชน์ของการพูลอินสแทนซ์คือ พูลของ bean สามารถมีจำนวนน้อยกว่าจำนวนไคลเอ็นต์ที่ติดต่อเข้ามาได้มาก เนื่องจากเวลาที่ใช้ในการคิดของไคลเอ็นต์ อาจเป็นเวลาที่เสียไประหว่างส่งข้อมูลผ่านเครือข่ายหรือเวลาที่มนุษย์ใช้ในการตัดสินใจบนฝั่งไคลเอ็นต์ ในระหว่า

นั้นคอนเทนเนอร์อาจนำอินสแทนซ์ของ bean ไปให้บริการกับไคลเอ็นต์ตัวอื่นได้เพื่อประหยัดทรัพยากรของระบบ

ขนาดพูลของ bean ไม่จำเป็นต้องกำหนดไว้ตายตัว คอนเทนเนอร์ที่มีความสามารถสูงสามารถปรับขนาดของพูลได้อย่างไดนามิก โดยอัตโนมัติขณะโหลดเปลี่ยนแปลง เช่น ถ้าหากมีไคลเอ็นต์เข้ามาใช้งานตอนกลางวันมากกว่าตอนกลางคืน คอนเทนเนอร์อาจจะสร้างพูลขนาดใหญ่ในตอนกลางวัน และขนาดเล็กกว่าในตอนกลางคืน ช่วยให้ระบบมีทรัพยากรเหลือมากขึ้นสำหรับทำงานอย่างอื่นในช่วงเวลาที่ระบบมีโหลดไม่มาก

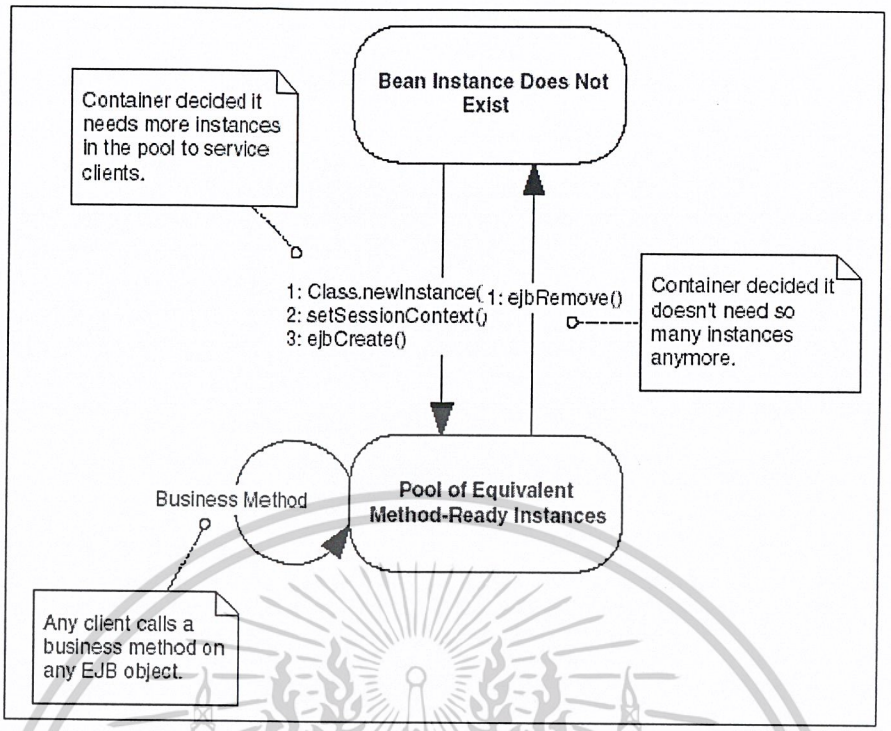
การพูลอินสแทนซ์ของ stateless session bean แสดงในรูปที่ 6-2



รูปที่ 6-2 การพูล Stateless Session Bean

6.2.1.2 วงจรชีวิตของ Stateless Session Bean

วงจรชีวิตของ stateless session bean ดังรูปที่ 6-3



รูปที่ 6-3 วงจรชีวิตของ Stateless Session Bean

6.2.1.3 ตัวอย่างโค้ด Stateless Session Bean

เป็น session bean ที่ทำหน้าที่เก็บสถานะของผู้ใช้งานระบบ ที่ login กับทางระบบแล้ว โดยโค้ดในโปรแกรมนี้ จะประกอบด้วยส่วนต่างๆ ดังนี้

Remote Interface

จะต้องมีบิซิเนสเมธอดทั้งหมดที่มีอยู่ใน bean คอนเทนเนอร์จะอิมพลีเมนต์ remote interface อิมพลีเมนต์นั้นคือ EJB object และ EJB object จะส่ง request ของไคลเอ็นต์ต่อไปยัง bean ที่แท้จริง ดังโค้ดต่อไปนี้

```
import javax.ejb.*;
import java.util.*;
import java.rmi.*;
 public interface SessionCustomer extends javax.ejb.EJBObject {
 public void setName(String name) throws RemoteException;
 public void setPassword(String password) throws RemoteException;;
}
```

รูปที่ 6-4 ตัวอย่างโค้ด SessionCustomer.java

remote interface จะต้องขยายมาจาก javax.ejb.EJBObject ซึ่งหมายความว่าคอนเทนเนอร์จะสร้าง EJB object ให้ โดยจะอิมพลีเมนต์จาก remote interface และจะมีทุกเมธอดที่มีใน remote interface ในตัวอย่างนี้ มีบิสสิเนสเมทธอด 2 เมทธอด คือ setName() และ ViewKeepSession(String iiD, String password)) โดยจะต้องอิมพลีเมนต์เมธอดนี้ใน enterprise bean class และเนื่องจาก remote interface ขยายมาจาก java.rmi.Remote มันจะโยน remote exception ด้วย ซึ่งเป็นข้อแตกต่างของบิซิเนสเมธอดในอินเทอร์เฟซและ enterpise bean class

Enterprise Bean Class

bean จะต้องขยายมาจากอินเทอร์เฟซ javax.ejb.SessionBean ซึ่งจะต้องนำโค้ดในส่วนของ Remote Interface มา implement ตัวอย่างโค้ดจะเป็นดังนี้

```
import javax.ejb.*;
import javax.naming.*;
import javax.rmi.PortableRemoteObject;
public class SessionCustomerBean implements SessionBean {
 public void setName(java.lang.String name) {
  this.name = name;
 }
 public void setPassword(java.lang.String password) {
  this.password = password;
 }
}
```

รูปที่ 6-5 ตัวอย่างโค้ด SessionCustomerBean.java

Home Interface

home interface จะใช้ในการระบุกลไกในการสร้างและทำลาย EJB object ตัวอย่างของโค้ดมีดังนี้

```
import javax.ejb.*;
import java.util.*;
import java.rmi.*;
```

```
public interface SessionCustomerHome extends javax.ejb.EJBHome {
 public SessionCustomer create() throws CreateException, RemoteException;
}
```

รูปที่ 6-6 ตัวอย่างโค้ด SessionCustomerHome.java

home interface จะต้องขยายมาจาก javax.ejb.EJBHome EJBHome จะกำหนดวิธีในการทำลาย EJB object เอง จึงไม่ต้องเขียนเมธอดนี้ home interface จะต้องมีเมธอดที่ใช้สร้าง EJB object โดยไม่ต้องมีพารามิเตอร์ใด ๆ และเมธอด create() จะต้องโยน java.rmi.RemoteException และ javax.ejb.CreateException

Deployment Descriptor

จะต้องประกอบด้วย ejb-xml.jar และ weblogic-ejb-xml.jar โดยภายใน descriptor สามารถมีรายละเอียดของ bean ได้มากกว่า 1 bean

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <display-name>SessionCustomer</display-name>
      <ejb-name>SessionCustomer</ejb-name>
      <home>home.SessionCustomerHome</home>
      <remote>home.SessionCustomer</remote>
      <ejb-class>home.SessionCustomerBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
      .....
```

รูปที่ 6-7 ตัวอย่างโค้ด ejb-xml.jar ของโปรแกรมที่เราสร้าง (SessionCustomer.java)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD WebLogic 7.0.0 EJB//EN' 'http://www.bea.com/servers/wls700/dtd/weblogic-ejb-jar.dtd'>
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>SessionCustomer</ejb-name>
    <jndi-name>SessionCustomer</jndi-name>
  </weblogic-enterprise-bean>
.....
```

รูปที่ 6-8 ตัวอย่างโค้ด weblogic-ejb-xml.jar ของโปรแกรมที่เราสร้าง (SessionCustomer.java)

ไคลเอนต์

ส่วนของโปรแกรมที่ทำการเรียกใช้งาน SessionCustomer.java ในส่วนของการสร้างอินสแทนซ์ของ class SessionCustomer.java คือ

```
Object objsession = jndiContext.lookup("SessionCustomer");
SessionCustomerHome chome = (SessionCustomerHome)
        PortableRemoteObject.narrow(objsession, SessionCustomerHome.class);
HttpSession session = request.getSession(true);
SessionCustomer customer =
        (SessionCustomer)session.getValue("SessionCustomer");

if (customer == null) {
        customer = chome.create();
        session.putValue("SessionCustomer", customer);
}
.....
```

รูปที่ 6-9 ตัวอย่างโค้ด ส่วนของไคลเอนต์

6.2.2 Stateful Session Bean

stateful session bean เป็น bean ที่เก็บสถานะการทำงานกับไคลเอ็นต์หนึ่ง ๆ ครอบคลุมระยะเวลาการเรียกเมธอดหลายครั้ง

6.2.2.1 ลักษณะของ Stateful Session Bean

6.2.2.1.1 การพูล Stateful Session Bean

ในสถานการณ์ที่ไคลเอ็นต์จำนวนมากกำลังทำงานกับ stateful session bean จำนวนหนึ่งในคอนเทนเนอร์ โดยปกติไคลเอ็นต์ต้องมีระยะเวลาในการคิด หรืออยู่ในสถานที่ห่างไกลออกไปมาก และระบบเครือข่ายสามารถส่งข้อมูลได้ช้า แต่หมายความว่าจะต้องมี stateful session bean จำนวนเท่า ๆ กันกับไคลเอ็นต์ในคอนเทนเนอร์ โดย stateful session bean แต่ละตัวจะเก็บสถานะการทำงานของไคลเอ็นต์แต่ละตัวเอาไว้ แต่คอนเทนเนอร์มีทรัพยากรที่จำกัด เช่น หน่วยความจำ ช่องทางติดต่อกับฐานข้อมูล (Socket Connection) หากว่าสถานะของการทำงานที่เก็บนั้นมีขนาดใหญ่ คอนเทนเนอร์ก็จะมีทรัพยากรเหลือไม่เพียงพอแก่การทำงานได้ ซึ่งปัญหานี้จะไม่เกิดกับ stateless session bean เพราะคอนเทนเนอร์สามารถพูล bean จำนวนน้อย เพื่อให้บริการกับไคลเอ็นต์จำนวนมากได้

การพูล stateful session bean ทำได้ไม่ง่าย เมื่อไคลเอ็นต์ติดต่อเข้ามายัง bean ไคลเอ็นต์นั้นจะเริ่มการทำงานกับ bean ทันที และสถานะการทำงานที่เก็บอยู่ใน bean จะต้องคงอยู่ในการเรียกใช้เมธอดครั้งต่อไปของไคลเอ็นต์ตัวเดิมนั้น ดังนั้นคอนเทนเนอร์จะไม่สามารถพูล bean ขึ้นมาและนำไปให้บริการการเรียกใช้งานเมธอดต่าง ๆ แบบไดนามิกได้ เนื่องจาก bean แต่ละตัวนั้น เก็บสถานะโดยเจาะจงกับไคลเอ็นต์แต่ละตัวเอาไว้

ปัญหาดังกล่าว ใกล้เคียงกับปัญหาที่เกิดขึ้นในระบบปฏิบัติการ เมื่อเรียกใช้แอพพลิเคชันบนเครื่องคอมพิวเตอร์ จะมีหน่วยความจำแบบฟิสิคอลที่จำกัดสำหรับให้โปรแกรมทำงาน ระบบปฏิบัติการต้องทำให้แอพพลิเคชันทำงานอยู่ได้ แม้ว่าแอพพลิเคชันจะต้องการปริมาณหน่วยความจำมากกว่าหน่วยความจำจริง ๆ ทั้งหมดที่มีอยู่ โดยระบบปฏิบัติการจะใช้เนื้อที่ในฮาร์ดดิสก์เป็นส่วนเพิ่มขยายของหน่วยความจำ เป็นการเพิ่มหน่วยความจำเสมือน (virtual memory) ของระบบ เมื่อแอพพลิเคชันใด ๆ ที่ยังไม่มีการทำงาน ข้อมูลในหน่วยความจำที่มันใช้งานอยู่ก็จะถูกย้าย (swap) ออกจากหน่วยความจำลงไปยังฮาร์ดดิสก์ เมื่อแอพพลิเคชันนั้นมีการทำงานอีกครั้ง ข้อมูลที่จำเป็นก็จะถูกย้ายกลับเข้ามาในหน่วยความจำ การสลับข้อมูลไปมาจะเกิดขึ้นบ่อยครั้งเมื่อมีการสลับการทำงานไปมาระหว่างแอพพลิเคชัน (context switching)

EJB คอนเทนเนอร์ใช้วิธีแก้ปัญหาที่เกิดขึ้นกับ stateful session bean ในลักษณะเดียวกัน เพื่อจำกัดจำนวนอินสแทนซ์ของ stateful session bean ในหน่วยความจำ คอนเทนเนอร์สามารถ swap stateful session bean ออกไปได้ โดยอาจเก็บสถานะของ bean ไว้ในฮาร์ดดิสก์หรือแหล่งเก็บข้อมูลอื่น การ swap stateful session bean ออกไปเรียกว่าการทำ passivation ทำให้ได้หน่วยความจำและทรัพยากรต่าง ๆ กลับคืนมา เมื่อไคลเอ็นต์ตัวเดิมเรียกใช้เมธอดของ bean

อีกครั้ง สถานะการทำงานกับไคลเอ็นต์ที่ถูก passivate ออกไปจะถูก swap กลับเข้ามาใน stateful session bean อีกครั้ง วิธีการนี้เรียกว่า activation ทำให้ bean สามารถทำงานกับไคลเอ็นต์ต่อไปได้

bean ที่ได้รับสถานะโดยการ activate เข้ามาอาจจะไม่ใช่อินสแทนซ์ของ bean ตัวเดิมที่เคยติดต่อกับไคลเอ็นต์ตัวนั้นก็ได้ ซึ่งไม่ทำให้เกิดผลใด ๆ เนื่องจากอินสแทนซ์ของ bean ตัวใหม่นั้น จะทำงานต่อไปจากจุดเดิมที่สถานะของการทำงานได้ถูก passivate เอาไว้

ดังนั้นอินสแทนซ์จำนวนจำกัดเท่านั้นที่จะอยู่ในหน่วยความจำ ขณะไคลเอ็นต์จำนวนมากมายติดต่อเข้ามา แต่การพูลวิธีนี้ก็มีข้อเสีย เพราะการทำ passivation และ activation ต้องมีการใช้ I/O ด้วย อาจทำให้เกิดกรณีคอขวด (bottleneck) ขึ้นได้ ซึ่งแตกต่างจาก stateless session bean ซึ่งสามารถทำพูลได้ง่าย เนื่องจากไม่ต้องเก็บค่าสถานะการทำงานเอาไว้

การทำ passivation อาจเกิดขึ้นได้ทุกขณะ เมื่อ bean นั้นไม่ได้เกี่ยวข้องกับการเรียกใช้งานเมธอดในขณะนั้น การตัดสินใจว่าจะ passivate bean เมื่อไหร่ขึ้นอยู่กับคอนเทนเนอร์ โดยมีข้อยกเว้นอยู่ว่า bean ซึ่งทำงานอยู่ในทรานแซ็กชัน จะไม่สามารถ passivate ได้จนกว่าทรานแซ็กชันนั้นจะเสร็จสมบูรณ์

ส่วนการactivation bean จะเกิดขึ้นเมื่อ bean นั้นได้รับการร้องขอเข้ามา หากไคลเอ็นต์ใด ๆ ทำการร้องขอเข้ามา แต่สถานะการทำงานของมันถูก passivate อยู่ คอนเทนเนอร์จะ activate bean และอ่านข้อมูลที่ถูก passivate อยู่กลับเข้ามาในหน่วยความจำ

โดยทั่วไป การทำ passivation และ activation ไม่มีประโยชน์สำหรับ stateless session bean เนื่องจากไม่มีสถานะที่จะทำการ passivate และ activate และการทำ passivation และ activation นั้นจะถูกนำไปใช้กับ entity bean ด้วย จะได้กล่าวถึงต่อไป

6.2.1.1.2 ข้อบังคับในการเก็บสถานะของไคลเอ็นต์

สถานะการทำงานของ bean เป็นไปตามกฎที่วางไว้โดย java object Serialization เมื่อคอนเทนเนอร์passivate bean จะใช้ object serialization ในการแปลงสถานะการทำงานของ bean ไปเป็นบิตข้อมูล หลังจากนั้นจึงเขียนบิตข้อมูลเหล่านี้ลงไปยังแหล่งเก็บข้อมูล เมื่อ bean ถูกเขียนลงไปยังแหล่งเก็บข้อมูลแล้ว หน่วยความจำที่มันใช้อยู่ก็จะถูกปล่อยโดย garbage collector ได้ ส่วนการ activate ก็เป็นขั้นตอนที่ตรงข้ามกัน โดยบิตข้อมูลที่ถูก serialize ที่เก็บอยู่ในแหล่งเก็บข้อมูลจะถูกอ่านกลับขึ้นมาในหน่วยความจำและแปลงไปเป็นข้อมูลของ bean ซึ่งอยู่ในหน่วยความจำ สิ่งที่ทำให้วิธีการเหล่านี้สามารถทำได้ คือการที่อินเทอร์เฟซ javax.ejb.EnterpriseBean สืบทอดมาจาก java.io.Serializable และทุก ๆ enterprise bean อิมพลีเมนต์อินเทอร์เฟซนี้

สำหรับจาวาออบเจ็กต์ที่เป็นส่วนหนึ่งของสถานะของ bean ก็ใช้หลักการเดียวกันคือออบเจ็กต์นั้นจะต้องอิมพลีเมนต์ java.io.Serializable ด้วย แม้ว่า bean จะต้องเป็นไปตามกฎของ object serialization แต่ EJB คอนเทนเนอร์ไม่จำเป็นต้องใช้โพรโตคอลมาตรฐานในการทำ

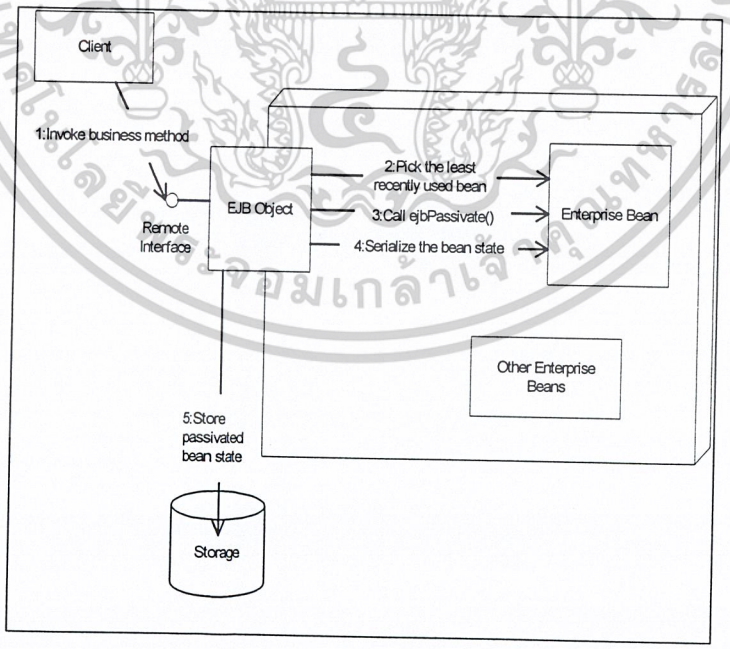
serialization เสมอไป อาจใช้โพรโตคอลของตัวเองซึ่งสามารถช่วยเพิ่มความยืดหยุ่นและความแตกต่างระหว่างคอนเทนเนอร์ของแต่ละผู้ผลิต

6.2.1.1.3 ขั้นตอนการทำ Passivation และ Activation

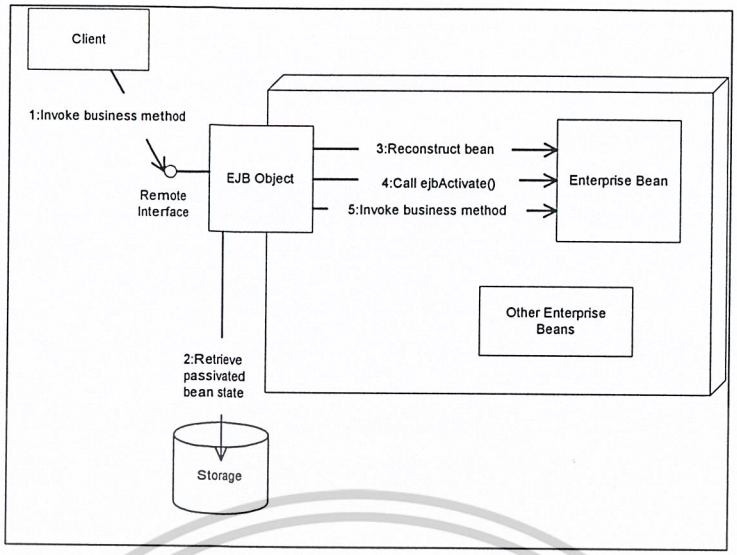
เมื่อ EJB คอนเทนเนอร์ passivate bean คอนเทนเนอร์จะเก็บสถานะการทำงานของ bean นั้นลงไปในแหล่งเก็บข้อมูล เช่นไฟล์ หรือฐานข้อมูล โดยคอนเทนเนอร์จะบอกให้ bean ทราบว่า มันกำลังจะทำการ passivate ด้วยการเรียกใช้เมธอด ejbPassivate() เมธอดนี้เป็นการเตือน bean ว่า สถานะการทำงานของมันกำลังจะถูก swap ออกไป

การที่คอนเทนเนอร์บอกให้ bean ทราบด้วยการเรียกใช้เมธอด ejbPassivate() นั้นเป็นสิ่งจำเป็น ทำให้ bean สามารถปล่อยทรัพยากรที่มันกำลังถือครองอยู่ได้ ทรัพยากรในที่นี้อาจเป็นช่องทางการติดต่อฐานข้อมูล , ซ็อกเก็ต , ไฟล์ที่เปิดอยู่ หรือทรัพยากรอื่น ๆ ซึ่งไม่สามารถเก็บลงในดิสก์ได้ หรือไม่มีความจำเป็นที่จะต้องเก็บไว้ เมื่อการเรียกเมธอด ejbPassivate() ของ bean เสร็จสมบูรณ์ bean ก็จะอยู่ในสถานะพร้อมถูก passivate การทำ passivation แสดงในรูปที่ 6-10

ขั้นตอนตรงกันข้ามจะเกิดขึ้นในกรณีของการ activate bean ซึ่งสถานะของการทำงานที่ถูก serialize จะถูกอ่านกลับเข้ามาในหน่วยความจำ และคอนเทนเนอร์จะสร้างสถานะขึ้นมาใหม่ในหน่วยความจำ โดยใช้ Object Serialization หรือสิ่งที่เหมือนกัน (equivalent) ในการแปลง หลังจากนั้นคอนเทนเนอร์จะเรียกเมธอด ejbActivate() ทำให้ bean สามารถเรียกใช้ทรัพยากรต่าง ๆ ที่จำเป็น ซึ่งถูกปล่อยไปขณะถูก passivate ด้วยเมธอด ejbPassivate() กลับมา ขั้นตอนการทำ activation แสดงในรูปที่ 6-11



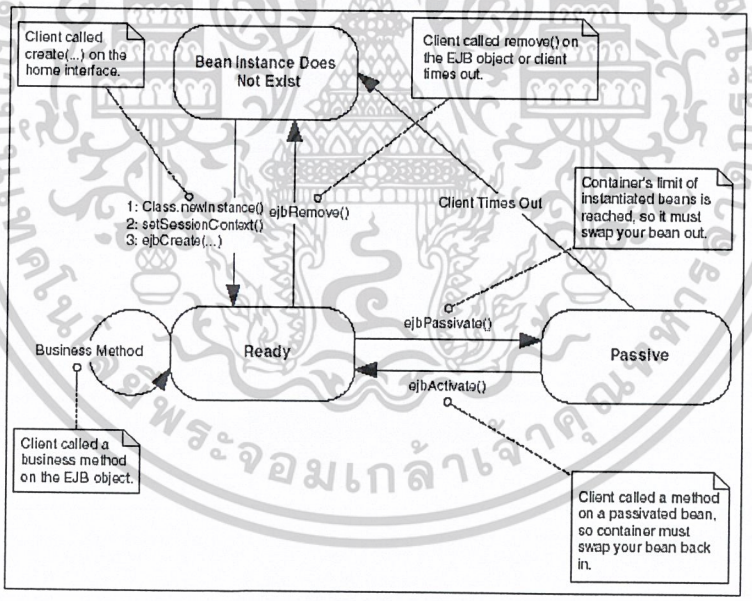
รูปที่ 6-10 การ Passivate Stateful Session Bean



รูปที่ 6-11 การ Activate Stateful Session Bean

6.2.2.2 วงจรชีวิตของ Stateful Session Bean

วงจรชีวิตของ stateful session bean ดังรูปที่ 6-12



รูปที่ 6-12 วงจรชีวิตของ Stateful Session Bean

6.2.2.3 การเขียนโค้ด Stateful session bean

วิธีการเขียนโค้ดในส่วนของ Stateful Session Bean นั้นเหมือนกับของ Stateless Session Bean แต่จะต่างกันตรงที่ xml file ที่จะบ่งบอกว่าเมทธอดนั้นเป็น Stateless หรือ Stateful

บทที่ 7

Entity Bean

7.1 แนวคิด Persistent concept

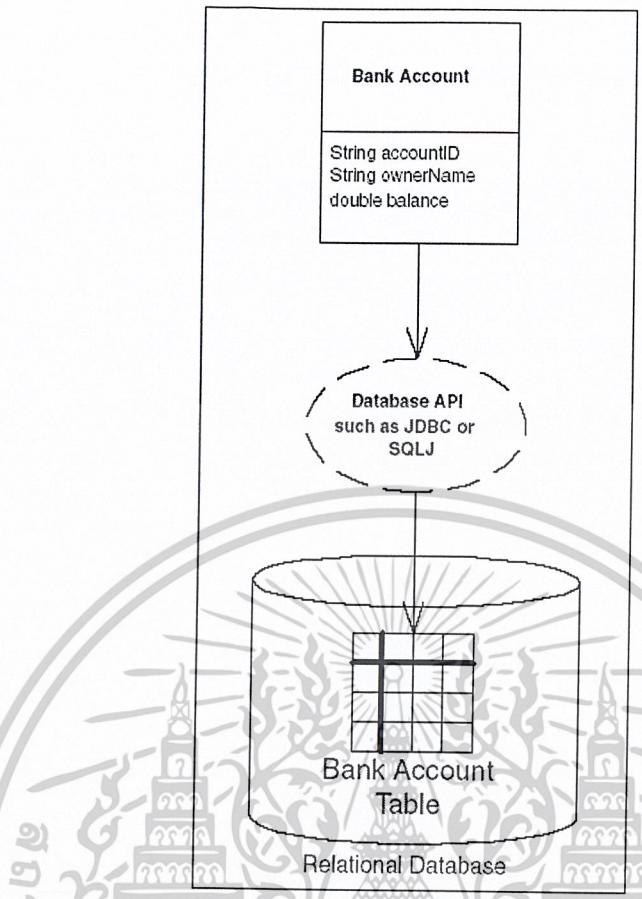
7.1.1 Java Object Serialization

ในการทำงานกับออบเจ็กต์ที่เขียนด้วยภาษาจาวา อาจต้องการเก็บสถานะ (state) ของออบเจ็กต์ลงใน permanent storage วิธีหนึ่งที่ทำได้คือใช้วิธีการ object serialization ซึ่งเป็นการแปลงออบเจ็กต์ให้อยู่ในรูป สตรีมของไบต์ข้อมูล (byte stream) หลังจากนั้น สามารถจะทำอะไรกับไบต์สตรีมนี้ก็ได้ เช่น ส่งข้ามระบบเครือข่าย หรือเก็บลง storage เช่น ไฟล์ , ฐานข้อมูล , JNDI ทรี

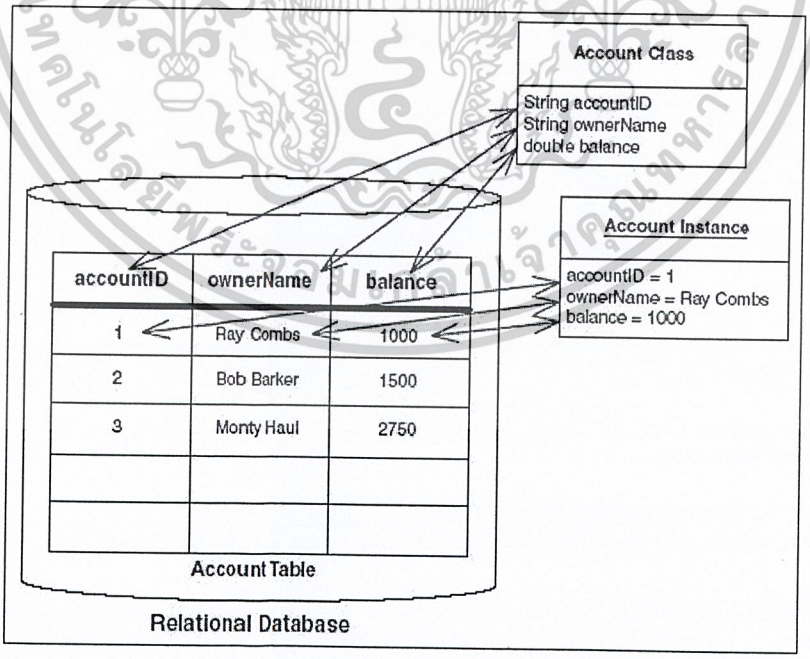
เมื่อต้องการเรียกกลับขึ้นมาใช้งานใหม่ จะต้องทำในกระบวนการย้อนกลับ คือ อ่านไบต์สตรีมขึ้นมา แล้วนำไปสร้างเป็นออบเจ็กต์ใหม่อีกครั้ง อย่างไรก็ตาม วิธีนี้ไม่เหมาะในงานที่มีการ query บ่อย เนื่องจาก operation ทุกอย่างทำกับออบเจ็กต์ จึงต้องโหลดทุกออบเจ็กต์กลับขึ้นมาเพื่อดูว่าเงื่อนไขตรงกับที่ query หรือไม่

7.1.2 Object-Relational Mapping

อีกวิธีหนึ่งที่ได้รับความนิยมในการเก็บจาวาออบเจ็กต์ก็คือ การใช้ฐานข้อมูลเชิงสัมพันธ์ (relational database) เช่น Oracle หรือ Microsoft SQL Server แทนที่จะทำการ serialize ออบเจ็กต์ การบันทึกจาวาออบเจ็กต์ใช้ JDBC หรือ SQL/J เพื่อแมปข้อมูลของออบเจ็กต์ไปยังฐานข้อมูลเชิงสัมพันธ์ นอกจากนี้ยังสามารถบันทึกชื่อของจาวาคลาสที่สัมพันธ์กับข้อมูลนี้ เพื่อที่จะสร้าง (instantiate) คลาสที่ถูกต้องตามออบเจ็กต์ที่ต้องการอ่านกลับขึ้นมาในหน่วยความจำ ในการโหลดออบเจ็กต์จากฐานข้อมูล จะต้องสร้างออบเจ็กต์จากคลาสนั้น อ่านข้อมูลขึ้นมาจากฐานข้อมูล และเก็บค่าเหล่านั้นกับฟิลด์ของอินสแทนซ์ของออบเจ็กต์ที่อยู่ในหน่วยความจำ ดังแสดงในรูปที่ 7-1 และ 7-2



รูปที่ 7-1 Object-Relational Mapping



รูปที่ 7-2 ตัวอย่างการแมประหว่างออบเจ็กต์และฐานข้อมูลเชิงสัมพันธ์

7.1.3 ฐานข้อมูลเชิงวัตถุ (Object Database)

Object Database Management System (ODBMS) สามารถเก็บออบเจ็กต์ไว้ในฐานข้อมูลได้โดยไม่ต้องผ่านกระบวนการ O/R mapping และทำให้การเขียนโค้ดเพื่อเข้าถึงข้อมูลทำได้ง่ายขึ้น โดยการเขียนโปรแกรมติดต่อกับ API ของฐานข้อมูลเชิงวัตถุแทนการเขียนโค้ดติดต่อกับ API ของฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลเชิงวัตถุส่วนใหญ่ จะให้เครื่องมือในการ query ข้อมูลที่เรียกว่า Object Query Language (OQL) ซึ่งเป็นการ query ในระดับสูงกว่าฐานข้อมูลเชิงสัมพันธ์

นอกจากนี้ ระบบฐานข้อมูลเชิงวัตถุยังมีความถูกต้อง (integrity) และความปลอดภัย (security) สูงกว่าระบบฐานข้อมูลเชิงสัมพันธ์ ในกรณี persistent object มีความซับซ้อนมาก ๆ เช่น การเก็บฐานข้อมูลของ CAD/CAM แต่ถ้าออบเจ็กต์ไม่ซับซ้อนและสามารถแมปไปเป็นฐานข้อมูลเชิงสัมพันธ์ได้ง่าย เช่น แอพพลิเคชันทางธุรกิจต่าง ๆ ซึ่งเน้นที่ปริมาณงานจำนวนมากแล้ว ฐานข้อมูลเชิงสัมพันธ์จะมีประสิทธิภาพที่สูงกว่า

7.2 แนวคิดของ Entity Bean

7.2.1 อินสแทนซ์ของ Entity Bean

คอมโพเน็นต์ที่ถูกดีพลอยแบ่งแยกความแตกต่างได้ดังนี้

- Application Logic Components

  เป็นคอมโพเน็นต์ที่ให้บริการสำหรับงานทั่ว ๆ ไปเช่น

- การคำนวณราคาคำสั่งซื้อ
- การออกบิลบัตรเครดิตของลูกค้า
- การหาอินเวิร์สของเมตริก

โดยส่วนใหญ่คอมโพเน็นต์เหล่านี้จะแทนการกระทำ เหมาะกับการใช้กับบิซิเนสโพรเซส

- Persistent Data Components

  เป็นออบเจ็กต์ที่รู้ถึงวิธีการแปลงตัวมันเองลงใน persistent storage โดยใช้กลไกต่าง ๆ เช่น serialization , O/R mapping ไปเป็นฐานข้อมูลเชิงสัมพันธ์หรือฐานข้อมูลเชิงวัตถุ ออบเจ็กต์ชนิดนี้เป็นตัวแทนของข้อมูล (data) เช่น

- ข้อมูลในบัญชีธนาคาร เช่น หมายเลขบัญชี และ ยอดคงเหลือ
- ข้อมูลของฝ่ายบุคคล เช่น ชื่อ แผนก เงินเดือน

เรียกได้ว่าคอมโพเน็นต์ชนิดนี้ใช้แสดงถึงสิ่งที่เป็นคำนาม เช่น คน สถานที่ สิ่งของ

การมองข้อมูลเหล่านี้เป็นออบเจ็กต์ แทนที่จะจัดการในระดับข้อมูลดิบ (raw data) นั้น ทำให้จัดการได้ง่ายขึ้น สามารถรวมข้อมูลที่สัมพันธ์กันให้อยู่ในออบเจ็กต์เดียว สามารถใช้เมธอดเพื่อจัดการกับกลุ่มข้อมูลเหล่านี้ และได้รับบริการจากมิดเดิลแวร์ เช่น ทรานแซกชัน , การเข้าถึงระบบเครือข่าย , และการรักษาความปลอดภัย

entity bean เป็นคอมโพเน็นต์ประเภท persistent data component ที่รู้วิธีการแปลงตัวมันเองลงในแหล่งเก็บข้อมูล เช่น ฐานข้อมูล โดยเก็บอยู่ในรูปแบบของฟิลด์ (field) หลาย ๆ ฟิลด์ เช่น ฟิลด์ของหมายเลขบัญชี และฟิลด์ของยอดคงเหลือ ซึ่งจะมีเมธอดที่สัมพันธ์กันอยู่ภายในคอมโพเน็นต์ด้วย เช่น getBankAccountNumber() และ getBankAccountBalance()

entity bean อาจเปรียบเสมือน serializable java object ที่สามารถเก็บตัวเองในรูปกลุ่มของบิตลงในแหล่งบันทึกข้อมูลที่ persistent entity bean สามารถทำ persistent ได้หลายวิธี เช่น serialization , O/R mapping และ object database persistence เนื่องจากข้อกำหนดของ Enterprise JavaBeans ไม่ได้บังคับวิธีการทำ persistent ไว้

ข้อแตกต่างระหว่าง entity bean กับ session bean คือ session bean แสดงถึงกระบวนการทำงานที่เกิดขึ้นโดยผู้ใช้ และจบลงเมื่อผู้ใช้จบการทำงานไป แต่ entity bean จะเก็บตัวข้อมูลจริง ๆ เช่น ข้อมูลของสินค้า , บัญชีธนาคาร , คำสั่งซื้อ หรือ ข้อมูลลูกค้า entity bean ไม่ทำงานที่ซับซ้อนมาก เช่น การออกใบเสร็จให้ลูกค้า แต่ตัวมันจะแทนลูกค้าเองเลย และ entity bean เป็นออบเจ็กต์ที่มีสถานะคงทน (persistent state object) แม้ว่าผู้ใช้จะจบการทำงานไปแล้วก็ตาม

อินสแทนซ์ของ entity bean จะมีลักษณะดังนี้

- เป็นสิ่งที่แทนข้อมูลที่ persistent ในลักษณะของภาษาจาวาที่อยู่ในหน่วยความจำ
- รู้วิธีอ่านค่าตัวเองเข้ามาจากแหล่งบันทึกข้อมูล และสามารถแมปข้อมูลจากในฐานข้อมูลเข้ากับแต่ละฟิลด์ของออบเจ็กต์ได้
- สามารถแก้ไขค่าได้ในหน่วยความจำ เพื่อที่จะเปลี่ยนแปลงค่าข้อมูลจริง
- มีความคงทน สามารถเก็บลงในแหล่งบันทึกข้อมูลอีกครั้งได้เพื่อเป็นการอัพเดตข้อมูล

อินสแทนซ์ของ entity bean คือมุมมองที่มองลงไปยังข้อมูลที่อยู่ในหน่วยความจำเป็นอินสแทนซ์ของ entity bean class

ข้อมูลของ entity bean (entity bean data หรือ data instance) คือกลุ่มของข้อมูลในฐานข้อมูลในระดับกายภาพ เช่น ระเบียนของบัญชีธนาคาร

7.2.2 ส่วนประกอบของ Entity Bean

7.2.2.1 Entity Bean Class

คือคลาสในภาษาจาวาที่จำลองข้อมูลที่ persistent โดย entity bean class จะแมปเข้ากับหนึ่ง entity definition ใน database schema เช่น entity bean class หนึ่งสามารถแมปเข้าได้กับ definition

ของตารางเชิงสัมพันธ์ ในกรณีนี้อินสแทนซ์ของ entity bean ของคลาสนั้นจะถูกแมปเข้ากับแถวหนึ่งในตาราง entity bean class สามารถมีเมธอดพื้นฐานเพื่อใช้จัดการหรือเข้าถึงข้อมูล เช่น เมธอดสำหรับลดยอดเงินในบัญชีธนาคาร เป็นต้น เช่นเดียวกับ session bean class EJB กำหนดเมธอดบังคับสำหรับ entity bean class ซึ่งคอนเทนเนอร์จะเรียกใช้เมธอดเหล่านี้โดยอัตโนมัติ

7.2.2.2 Remote Interface

เป็นอินเทอร์เฟซอ้างอิงไปยัง bean ซึ่งไคลเอ็นต์จะเข้ามาเรียกใช้ ภายใน remote interface จะต้องระบุถึงบิซิเนสเมธอดทั้งหมดที่ entity bean มีให้ใช้งาน ผู้พัฒนา EJB คอนเทนเนอร์จะให้เครื่องมือสำหรับอิมพลีเมนท์ remote interface มาให้ โดยส่วนอิมพลีเมนต์ของ remote interface ก็คือ EJB ออบเจ็กต์ ซึ่งทำหน้าที่เป็นชั้นกั้นกลางระหว่างไคลเอ็นต์กับ bean ไคลเอ็นต์จะร้องขอการใช้งาน bean มายัง EJB ออบเจ็กต์แทนที่จะร้องขอกับ bean โดยตรง

เนื่องจาก EJB ออบเจ็กต์เป็นส่วนหนึ่งของคอนเทนเนอร์ มันจะประกอบด้วยลอจิกในการดักจับการเรียกใช้เมธอด และจัดการอินสแทนซ์ของ bean ซึ่งเป็นหลักการเดียวกับ session bean

7.2.2.3 Home Interface

เป็นอินเทอร์เฟซที่ไคลเอ็นต์จะต้องเรียกใช้เพื่อสร้าง , ค้นหา , และทำลาย EJB ออบเจ็กต์ของ entity bean ภายใน home interface ต้องระบุเมธอดบอกถึงวิธีที่มีทั้งหมดในการสร้าง EJB ออบเจ็กต์ใหม่ขึ้นมา , การค้นหาและการทำลาย EJB ออบเจ็กต์เดิมที่มีอยู่แล้ว ผู้พัฒนา EJB คอนเทนเนอร์ จะให้เครื่องมือสำหรับ อิมพลีเมนต์ home interface นี้มาด้วย ส่วนอิมพลีเมนต์ของ home interface ก็คือ home object ซึ่งเปรียบเสมือนเป็นโรงงานที่ใช้ในการสร้าง , ค้นหา และทำลาย EJB ออบเจ็กต์ ในการค้นหา home object จะต้องเรียกใช้บริการ look up ของ JNDI เช่นเดียวกับหลักการของ session bean

7.2.2.4 Primary Key Class

เป็น unique identifier สำหรับ entity bean โดย primary key จะแยกความแตกต่างระหว่างแต่ละ entity bean เช่น ถ้ามี entity bean ประกอบด้วยบัญชีธนาคารทั้งหมดหนึ่งล้านบัญชี แต่ละบัญชีต้องมี unique ID (เช่น หมายเลขบัญชีธนาคาร) ซึ่งจะต้องไม่ซ้ำกันเลยในบัญชีทั้งหมด primary key เป็นออบเจ็กต์ที่สามารถประกอบด้วยหลาย attribute ได้ อาจเป็นข้อมูลที่จำเป็นในการแยกแยะอินสแทนซ์ของ entity bean ในกรณีที่ entity bean เป็นตัวแทนของความสัมพันธ์ที่ซับซ้อน primary key อาจเป็นทั้งออบเจ็กต์เลยก็ได้ EJB ให้ความยืดหยุ่นในการระบุว่าจะใช้อะไรเป็น unique ID โดยการรวม primary key class เข้ากับ entity bean มีกฎอยู่เพียงข้อเดียวว่า primary key class ต้องสามารถ serialize ได้ และเป็นไปตามกฎของ java object serialization

7.2.2.5 Deployment Descriptor

เก็บรายการคุณสมบัติ (properties) ต่าง ๆ ที่คอนเทนเนอร์ใช้ในการดีพลอย deployment descriptor ใช้แจ้งรายละเอียดเกี่ยวกับ bean ให้กับคอนเทนเนอร์ เมื่อจะนำ entity bean ไปใช้งาน ต้องรวมไฟล์เหล่านี้เข้าไว้ในไฟล์ ejb-jar รวมทั้ง manifest file เพื่อใช้ค้นหาตำแหน่งของ bean ในไฟล์ ejb-jar ด้วย

7.2.3 ลักษณะของ Entity Bean

7.2.3.1 Entity Bean มีช่วงชีวิตยาวนาน

session bean มีอายุเท่ากับ session ของไคลเอ็นต์เท่านั้น เมื่อไคลเอ็นต์จบการติดต่อก็สามารถทำลาย session bean ได้ทันที ส่วน entity bean จะมีช่วงชีวิตยาวนานเท่ากับคอนเทนเนอร์ โดยเริ่มและจบการทำงานพร้อมกับ คอนเทนเนอร์ เช่น ข้อมูลบัญชีธนาคาร

7.2.3.2 Entity Bean มีความคงทนต่อความผิดพลาด

session bean มีช่วงชีวิตที่สั้น เพียงแค่ session ของไคลเอ็นต์เท่านั้น และเมื่อมีเหตุการณ์ผิดพลาด เช่น JVM แครชเกิดขึ้น session bean จะถูกทำลายทันที แต่ entity bean เป็นส่วนหนึ่งของ persistent storage ดังนั้นการแครชของ JVM หรือฐานข้อมูล จะไม่มีผลกระทบต่อ entity bean เมื่อระบบกลับมาทำงานได้อีกครั้ง อินสแทนซ์ของ entity bean ก็สามารถสร้างขึ้นมาใหม่ได้ โดยการอ่านข้อมูลจากฐานข้อมูล และนำมาสร้างเป็นอินสแทนซ์ของ entity bean เพื่อเป็นตัวแทนของข้อมูลนั้นในหน่วยความจำได้

7.2.3.3 อินสแทนซ์ของ Entity Bean เป็นมุมมองไปยังฐานข้อมูล

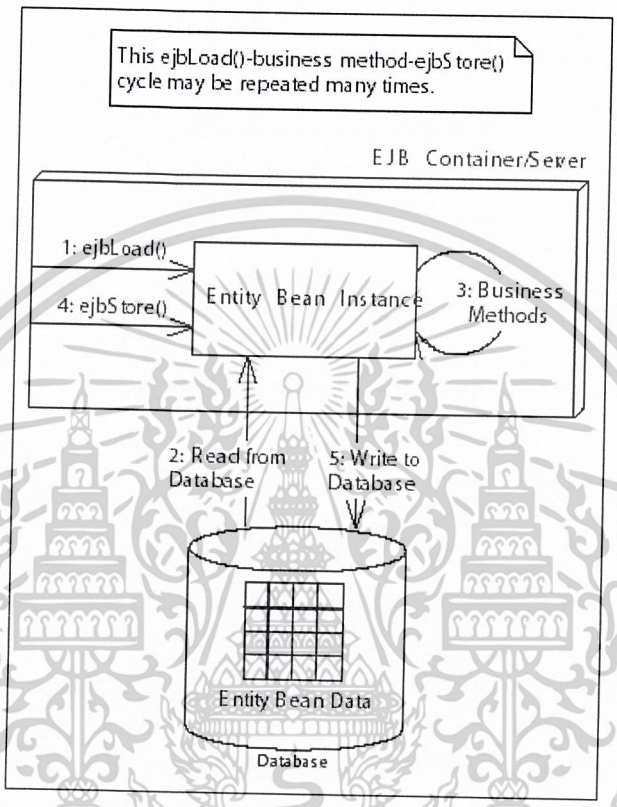
ในการโหลด entity bean เข้ามาเป็นอินสแทนซ์ในหน่วยความจำ จะต้องอ่านข้อมูลที่เก็บอยู่ในฐานข้อมูลแล้วนำมาจัดการใน java virtual machine โดยมองว่าออบเจ็กต์ที่อยู่ในหน่วยความจำและฐานข้อมูลนั้นเป็นสิ่งเดียวกัน หมายความว่า เมื่อแก้ไขอินสแทนซ์ของ entity bean ซึ่งอยู่ในหน่วยความจำ ข้อมูลในฐานข้อมูลจะต้องถูกเปลี่ยนแปลงด้วยโดยอัตโนมัติ entity bean ที่อยู่ในหน่วยความจำนั้น เสมือนเป็นมุมมองไปยังฐานข้อมูล

ในความเป็นจริงแล้ว ข้อมูลหนึ่ง ๆ จะมีอยู่สองชุดด้วยกัน คืออินสแทนซ์ของ entity bean ซึ่งอยู่ในหน่วยความจำ และตัว entity bean จริง ๆ ที่ถูกเก็บอยู่ในฐานข้อมูล ดังนั้นจึงต้องมีวิธีการส่งผ่านข้อมูลไปมาระหว่างจาวาออบเจ็กต์และฐานข้อมูล การส่งผ่านข้อมูลนี้สามารถทำได้โดยเมธอดที่ทุก entity bean class ต้องอิมพลีเมนต์ นั่นคือ ejbLoad() และ ejbStore()

ejbLoad() อ่านข้อมูลจาก persistent storage เข้าไปยังฟิลด์ของ entity bean ในหน่วยความจำ

ejbStore() บันทึกค่าในฟิลด์ปัจจุบันของอินสแทนซ์ของ bean ลงไปยังแหล่งเก็บข้อมูล เป็นการทำงานที่ตรงข้ามกับ ejbLoad()

เมธอดทั้งสองเป็น callback method ที่คอนเทนเนอร์จะเรียกใช้งานโดยอัตโนมัติ เป็นเมธอดการจัดการที่ EJB บังคับให้ต้องมี คอนเทนเนอร์จะพิจารณาเองว่า จะเรียกใช้งานเมธอดเหล่านี้เมื่อใดขึ้นอยู่กับความสามารถของคอนเทนเนอร์ bean อาจถูกเรียกใช้งานเมธอดเหล่านี้เมื่อใดก็ได้ โดยคอนเทนเนอร์จะพิจารณาจากสถานะของทรานแซ็กชัน และ synchronize ข้อมูลให้โดยอัตโนมัติ โดยปกติแล้วลำดับจะเป็นดังรูปที่ 7-3



รูปที่ 7-3 การโหลดและการเก็บข้อมูลของ Entity Bean

แต่ในบางคอนเทนเนอร์สามารถกำหนดคุณสมบัติให้ enity bean เป็น read-only เท่านั้น ทำให้ไม่จำเป็นต้องเรียก ejbStore()

7.2.3.4 อินสแทนซ์ของ Entity Bean หลายตัวอาจแสดงข้อมูลเดียวกัน

เมื่อมีไคลเอ็นต์หลายตัวเข้ามาใช้งานข้อมูลต่าง ๆ ในเวลาเดียวกัน จำเป็นต้องออกแบบวิธีการเข้าถึงข้อมูลที่มีประสิทธิภาพสูงให้กับ entity bean โดย entity bean ไม่อนุญาตให้ไคลเอ็นต์หลายตัวใช้อินสแทนซ์ของ entity bean ตัวเดียวพร้อม ๆ กัน เนื่องจากอินสแทนซ์ของ bean ที่ทำเช่นนี้ได้ จะต้องเขียนแบบ thread-safe ซึ่งมีความซับซ้อนและผิดพลาดได้ง่าย ตรงข้ามกับวัตถุประสงค์ของ EJB ที่ต้องการให้การพัฒนาแอพพลิเคชันทำได้ง่ายและรวดเร็ว ปัญหาอีกข้อ

หนึ่งคือ ในขณะที่มีหลาย ๆ เทรดทำงานพร้อมกันนั้น การควบคุมทรานแซกชันโดยระบบจัดการทรานแซกชันจะทำได้ยากมาก

จากปัญหาดังกล่าว EJB จึงอนุญาตให้มีเพียงเธรดเดียวเท่านั้น ที่สามารถทำงานอยู่ในอินสแทนซ์ของ bean ไม่ว่าจะเป็นอินสแทนซ์ของ session bean หรือ entity bean ก็ตาม

เพื่อเพิ่มประสิทธิภาพในการทำงาน คอนเทนเนอร์สามารถสร้างอินสแทนซ์ของ entity bean หนึ่ง ๆ ขึ้นมาได้หลายชุด ทำให้ไคลเอ็นต์หลายตัวสามารถทำงานพร้อมกันได้ โดยแต่ละไคลเอ็นต์จะทำงานกับ อินสแทนซ์หนึ่งตัว โดยที่อินสแทนซ์ของ entity bean เหล่านั้นสามารถแทนข้อมูลชุดเดียวกัน

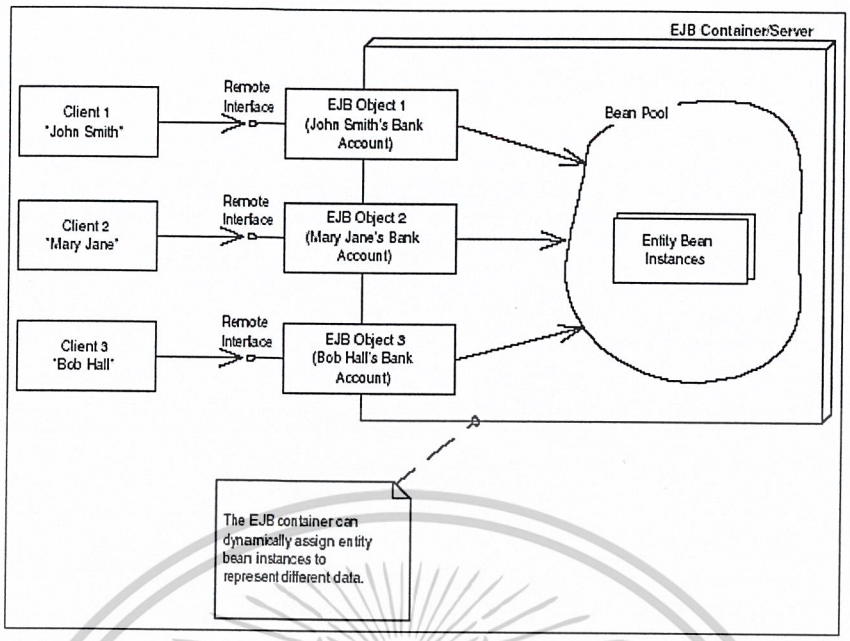
การมีอินสแทนซ์ของ bean หลาย ๆ ชุดแทนข้อมูลชุดเดียวกันนั้น ทำให้เกิดปัญหาใหม่ตามมา คือปัญหาเรื่องความถูกต้องของข้อมูล เมื่อไคลเอ็นต์หลายตัวพยายามจะแก้ไขข้อมูลพร้อม ๆ กัน และอาจทำให้มีไคลเอ็นต์ที่ได้ข้อมูลผิดพลาดไป เพื่อแก้ปัญหานี้คอนเทนเนอร์จะทำการ synchronize ข้อมูลของ bean กับข้อมูลที่อยู่ในฐานข้อมูลด้วยการเรียกใช้งานเมธอด ejbLoad() และ ejbStore()

ความถี่ของการ synchronize ข้อมูล คอนเทนเนอร์จะพิจารณาจากทรานแซกชัน ทรานแซกชันทำให้ไคลเอ็นต์รู้สึกเหมือนกับว่าทำงานกับข้อมูลนั้นเพียงคนเดียว ทั้งที่ความเป็นจริงมีไคลเอ็นต์อีกหลายตัวกำลังทำงานกับข้อมูลชุดเดียวกันนั้นอยู่

7.2.3.5 อินสแทนซ์ของ Entity Bean สามารถพูลได้

เพื่อประหยัดเวลาในการสร้างและทำลายอินสแทนซ์ของ bean ทุกครั้งที่มีไคลเอ็นต์ติดต่อเข้ามา และเพื่อเพิ่มประสิทธิภาพในการทำงาน อินสแทนซ์ของ entity bean เป็นออบเจ็กต์ที่สามารถนำกลับมาใช้ใหม่และพูลได้ คอนเทนเนอร์สามารถพูลและนำอินสแทนซ์ของ entity bean กลับไปใช้ใหม่เพื่อแทนอินสแทนซ์ของข้อมูลประเภทเดียวกันได้ เช่น คอนเทนเนอร์สามารถใช้อินสแทนซ์ของ entity bean บัญชีธนาคาร เพื่อแทนบัญชีหนึ่ง ๆ ในระบบบัญชีธนาคาร เมื่อใช้งานอินสแทนซ์นี้เสร็จแล้วก็สามารถนำไปให้บริการกับ ไคลเอ็นต์ตัวอื่น ๆ ได้ต่อไปและอาจแทนข้อมูลคนละชุดกับของเดิม คอนเทนเนอร์จัดการโดยการกำหนด อินสแทนซ์ของ entity bean ให้กับ EJB ออบเจ็กต์ที่แตกต่างกันออกไปในแต่ละไคลเอ็นต์

นอกจากจะช่วยประหยัดเวลาในการสร้างและทำลายอินสแทนซ์ของ bean โดยไม่จำเป็นแล้ว วิธีการนี้ยังช่วยลดทรัพยากรที่ระบบจำเป็นต้องใช้อีกด้วย ดังแสดงในรูปที่ 7-4



รูปที่ 7-4 การพูลอินสแทนซ์ของ Entity Bean โดยคอนเทนเนอร์

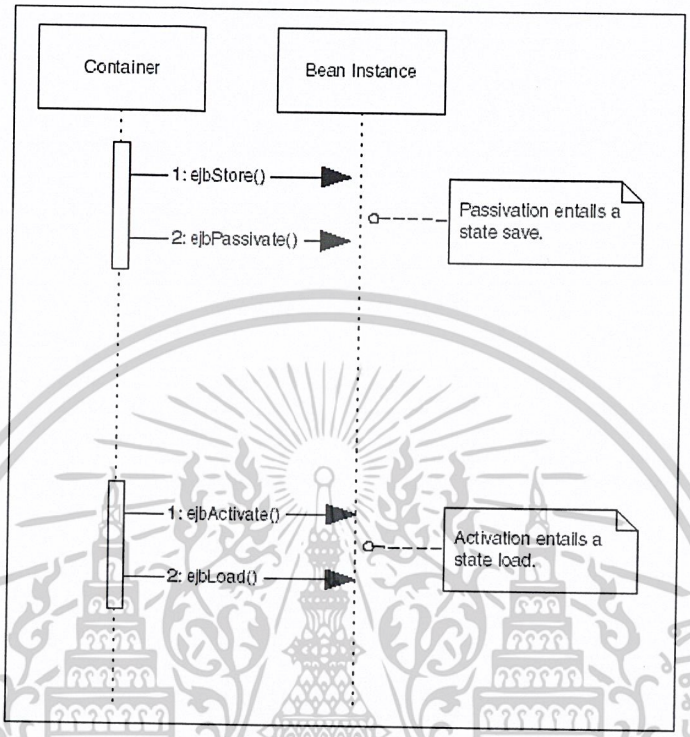
ในทำนองเดียวกับ stateful session bean การกำหนดอินสแทนซ์ของ entity bean ให้กับอีก EJB ออบเจ็กต์หนึ่งนั้นมีความซับซ้อน เมื่อ entity bean ถูกกำหนดให้กับ EJB ออบเจ็กต์ใด ๆ มันอาจมีการใช้งานทรัพยากร เช่น socket connection เป็นต้น แต่ในเวลาที่อยู่ในพูล ไม่มีความจำเป็นที่จะต้องใช้ซ็อกเก็ต ดังนั้นเพื่อให้ bean สามารถปล่อย และกลับมาใช้งานทรัพยากรต่าง ๆ entity bean จะต้องอิมพลีเมนต์ callback method ทั้งสองนี้

ejbActivate() เป็นเมธอดที่คอนเทนเนอร์เรียกใช้งานเมื่อจะนำอินสแทนซ์ของ bean ออกจากพูล กระบวนการนี้เรียกว่า activation บอกให้รู้ว่า คอนเทนเนอร์กำลังกำหนด bean ให้กับ EJB ออบเจ็กต์และ primary key หนึ่ง เมธอด ejbActivate() ควรจะร้องขอทรัพยากรที่ต้องใช้งานเมื่อถูกกำหนดให้กับ EJB ออบเจ็กต์แล้ว เช่นซ็อกเก็ต หลักการนี้เป็นหลักการเดียวกับการทำ activation ของ stateful session bean

ejbPassivate() เป็นเมธอดที่คอนเทนเนอร์เรียกใช้งานเมื่อจะนำอินสแทนซ์ของ bean เก็บเข้าไปในพูล กระบวนการนี้เรียกว่า passivation เพื่อบอกให้รู้ว่า คอนเทนเนอร์กำลังถอน bean ออกจาก EJBออบเจ็กต์ เมธอด ejbPassivate() ควรทำการปล่อยทรัพยากรที่ถือครองอยู่ทั้งหมด เช่นซ็อกเก็ตซึ่ง bean ได้ถือครองไว้ขณะ activation หลักการนี้เป็นหลักการเดียวกับการทำ passivation ของ stateful session bean

เมื่ออินสแทนซ์ของ entity bean ถูก passivate ไม่เพียงแต่จะต้องปล่อยทรัพยากรที่ถือครองเท่านั้น แต่ต้องบันทึกสถานะของมันลงไปแหล่งเก็บข้อมูลด้วย เพื่อให้ข้อมูลในแหล่งเก็บข้อมูลถูกอัพเดตให้ตรงกับสถานะล่าสุดของอินสแทนซ์ของ entity bean ในการบันทึกฟิลด์ของอินสแทนซ์ลงในฐานข้อมูล คอนเทนเนอร์จะเรียกใช้งานเมธอด ejbStore() ก่อนทำ passivation ทำนองเดียวกัน

เมื่ออินสแทนซ์ของ entity bean ถูก activate ไม่เพียงแต่จะต้องร้องขอการใช้งานทรัพยากรที่จำเป็น แต่จะต้องโหลดข้อมูลล่าสุดขึ้นมาจากฐานข้อมูลด้วย ในการโหลดข้อมูลล่าสุดจากฐานข้อมูลนี้ คอนเทนเนอร์จะเรียกใช้เมธอด ejbLoad() หลังจากทำ activation วิธีการนี้แสดงอยู่ในรูปที่ 7-5



รูปที่ 7-5 การทำ passivation และ activation ของ Entity Bean

entity bean มีความคล้ายคลึงกับ stateful session bean ตรงที่สามารถทำ activation และ passivation ได้ แต่ข้อแตกต่างที่สำคัญคือ entity bean มีเมธอด ejbLoad() เพื่อโหลดค่าสถานะขณะทำ activation และ เมธอด ejbStore() เพื่อบันทึกสถานะขณะทำ passivation entity bean เป็นออบเจ็กต์ที่มีความซับซ้อนกว่า stateful session bean มาก และต้องการวิธีการที่ซับซ้อนมากกว่า object serialization อินสแทนซ์ของ entity bean อาจเก็บข้อมูลของตัวเองลงในฐานข้อมูลแบบออบเจ็กต์โดยใช้ API ของฐานข้อมูลแบบออบเจ็กต์ ทำให้ entity bean ต้องการเมธอด ejbLoad() และ ejbStore() เพื่อจัดการเกี่ยวกับการเก็บค่าสถานะ ขณะที่ stateful session bean ไม่ต้องการเมธอดเหล่านี้ แต่จะใช้วิธีการ object serialization ในการบันทึกสถานะของ stateful session bean

7.2.3.6 Entity Bean สามารถทำ persistent ได้สองวิธี

entity bean เป็น persistent object ที่รู้ถึงวิธีการแปลงตัวมันเองลงในแหล่งบันทึกข้อมูล เช่น ฐานข้อมูลเชิงสัมพันธ์ โดยใช้หลักการ O/R mapping หรือฐานข้อมูลเชิงวัตถุ มีวิธีการทำ persistent อยู่ 2 วิธี

1) Bean-Managed Persistent (BMP)

วิธีนี้ bean เป็นผู้จัดการเกี่ยวกับการติดต่อกับแหล่งบันทึกข้อมูลเอง โดยอิมพลีเมนต์การเรียกใช้ฐานข้อมูลลงไปใน bean เช่น เมื่อทำงานกับฐานข้อมูลเชิงสัมพันธ์ entity bean สามารถทำคำสั่ง SQL INSERT ผ่านทาง JDBC เพื่อบันทึกข้อมูลลงไปในฐานข้อมูลเชิงสัมพันธ์ หรืออาจทำคำสั่ง SQL DELETE ผ่านทาง JDBC เพื่อลบข้อมูลออกจากฐานข้อมูล

2) Container-Managed Persistent (CMP)

วิธีนี้ให้คอนเทนเนอร์เป็นผู้จัดการเกี่ยวกับ persistent ให้ ทำให้สามารถตัดลอจิกที่เกี่ยวข้องกับการ persistent ข้อมูลออกจาก bean ได้ ไม่จำเป็นต้องเขียนโค้ดลงไปใน bean คลาส เพียงแต่บอก EJB คอนเทนเนอร์ว่า ฟิลด์ใดเป็น persistent ฟิลด์ โดยระบุไว้ใน deployment descriptor เมื่อคอนเทนเนอร์ทราบแล้วว่า ฟิลด์ใดบ้างที่ต้อง persistent คอนเทนเนอร์จะจัดการลอจิกเกี่ยวกับการเข้าถึงข้อมูลให้โดยอัตโนมัติ

การจัดการเกี่ยวกับ persistent โดยอัตโนมัตินี้ เป็นหน้าที่ของผู้ดีพลอยที่จะระบุว่า entity bean จะแมปลงไปยังแหล่งเก็บข้อมูลอย่างไร เช่น ฟิลด์ของจาวาออบเจ็กต์ชื่อ customerID จะต้องแมปเข้ากับคอลัมน์ชื่อ id ในตาราง customer ของฐานข้อมูล เป็นต้น สิ่งนี้เป็นจุดเด่นที่สำคัญของ entity bean เพราะสามารถเขียนออบเจ็กต์ของข้อมูลที่ไม่ขึ้นกับแหล่งบันทึกข้อมูลได้ และสามารถใช้ได้ในสภาพแวดล้อมที่หลากหลาย

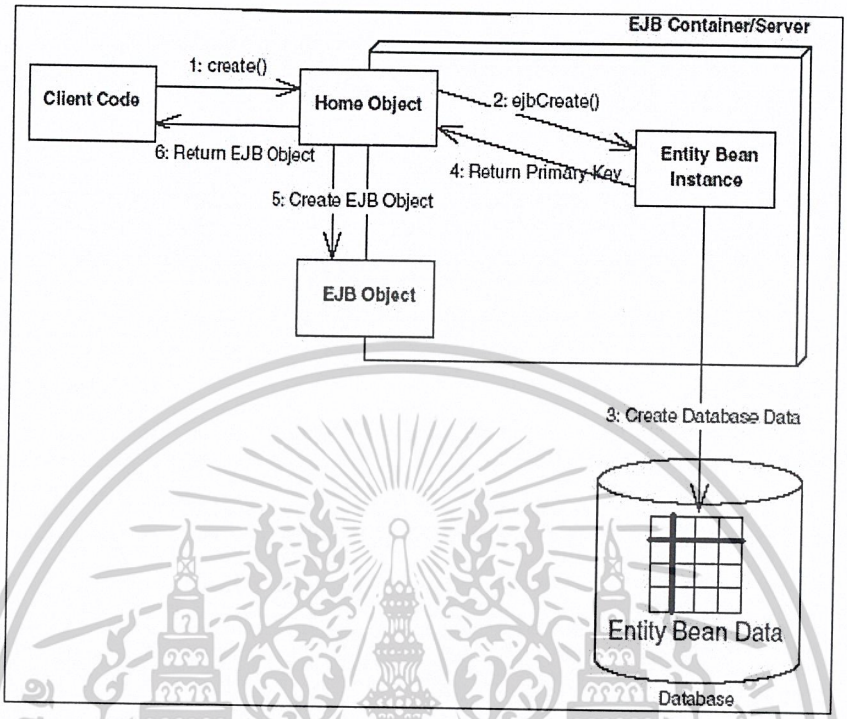
7.2.3.7 Entity Bean สามารถสร้าง ,ทำลาย หรือค้นหาได้

การกำหนดค่าเริ่มต้น และการทำลาย entity bean แตกต่างออกไปจาก session bean เล็กน้อย entity bean เป็นมุมมองลงไปยังฐานข้อมูล และมองอินสแทนซ์ของ entity bean และข้อมูลที่อยู่ในฐานข้อมูลเป็นสิ่งเดียวกัน เพราะข้อมูลจะถูก synchronize กันอยู่เสมอ การสร้างและกำหนดค่าเริ่มต้นให้กับอินสแทนซ์ของ entity bean จึงเป็นการสร้างและกำหนดค่าเริ่มต้นให้กับข้อมูลในฐานข้อมูลด้วย ดังนั้นเมื่อ entity bean ถูกสร้างและกำหนดค่าเริ่มต้นในหน่วยความจำด้วยเมธอด ejbCreate() เมธอดนี้จะสร้างข้อมูลใหม่ในฐานข้อมูลซึ่งสัมพันธ์กับอินสแทนซ์ในหน่วยความจำ ในทำนองเดียวกัน เมื่อเมธอด ejbRemove() ถูกเรียกใช้งาน ข้อมูลในฐานข้อมูลที่สัมพันธ์กันก็จะถูกลบทิ้ง ถ้าเป็นแบบ container-managed persistence คอนเทนเนอร์จะแก้ไขข้อมูลในฐานข้อมูลให้อัตโนมัติ จึงไม่ต้องอิมพลีเมนต์เมธอดทั้งสองนี้

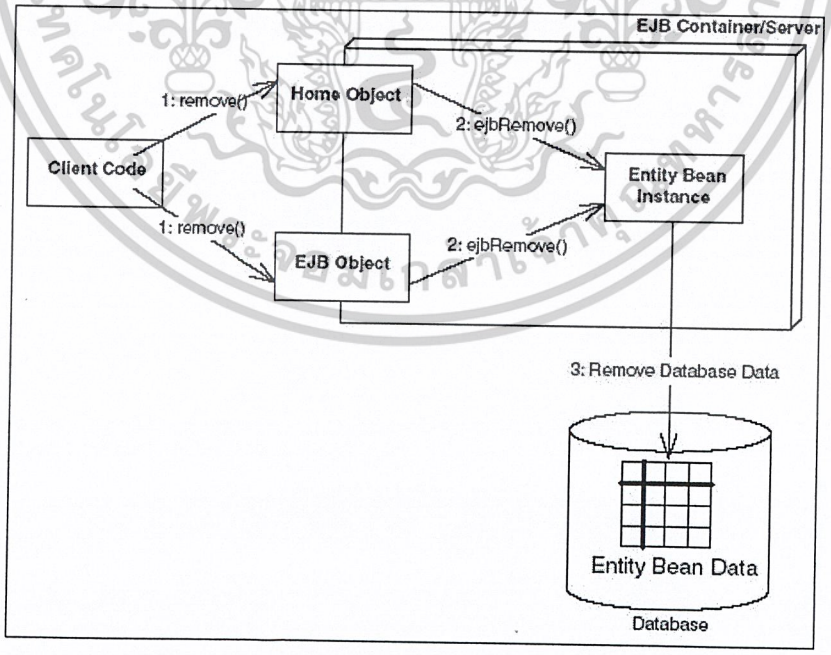
ข้อมูลของ entity bean สามารถระบุชี้เฉพาะตัวได้ จึงสามารถใช้การค้นหา entity bean แทนที่จะต้องสร้างขึ้นมาใหม่ การค้นหา entity bean เหมือนกับการใช้คำสั่ง SQL SELECT การค้นหา entity bean ก็คือการค้นหาข้อมูลในแหล่งบันทึกข้อมูล แตกต่างจาก session bean ที่ไม่สามารถค้นหาได้ เพราะ session bean นั้นไม่ได้เป็นออบเจ็กต์ที่ถาวร

วิธีค้นหา entity bean มีได้หลายวิธีด้วยกัน โดยจะระบุวิธีการเหล่านี้เป็นเมธอดไว้ใน home interface ของ entity bean เรียกว่า finder method นอกเหนือจากเมธอดที่ใช้ในการสร้างและทำลาย

entity bean จุดนี้เป็นจุดที่ home interface ของ session bean และ entity bean แตกต่างกัน เพราะใน home interface ของ session bean ไม่มี finder method



รูปที่ 7-6 การสร้าง Entity Bean และ EJB Object ด้วยเมทธอด ejbCreate()



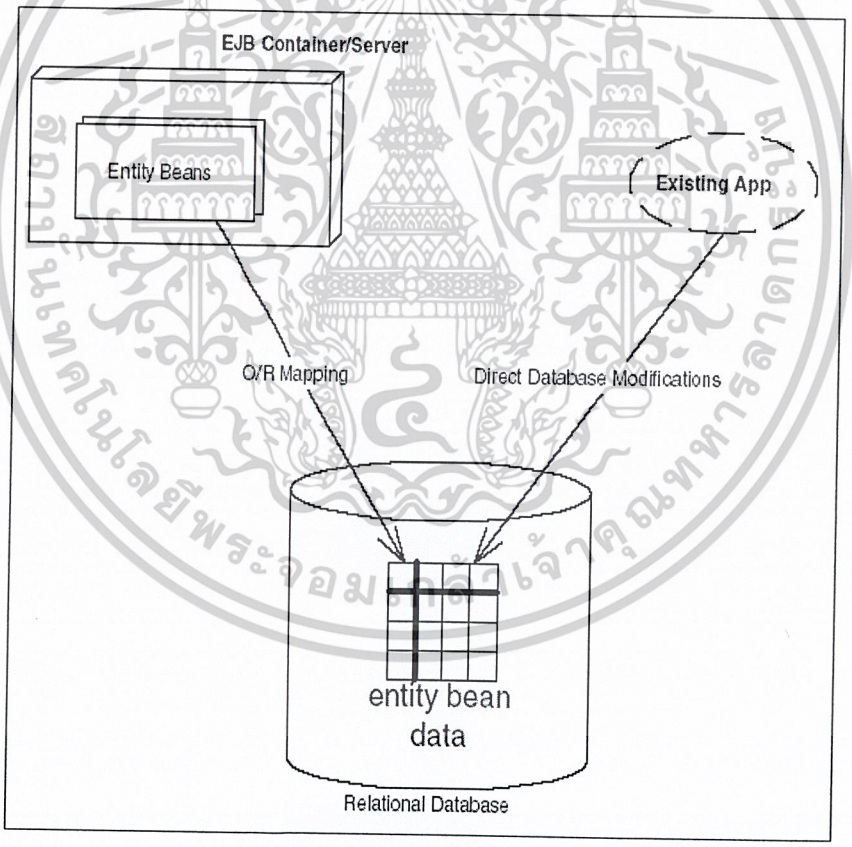
รูปที่7-7 การทำลาย Entity Bean ที่เป็นตัวแทนของข้อมูลในฐานข้อมูล ด้วยเมธอด ejbRemove()

7.2.3.8 Entity Bean สามารถใช้เป็นตัวแทน Legacy Data และ Legacy Systems ได้

entity bean สามารถเกิดก่อนที่จะมีการนำ EJB มาใช้ก็ได้ เพราะ entity bean ก็คือข้อมูล ดังนั้นสามารถนำระบบฐานข้อมูลเก่า (legacy database) หรือระบบเก่า (legacy system) และใช้ entity bean เป็นตัวแทนข้อมูลเหล่านี้ได้

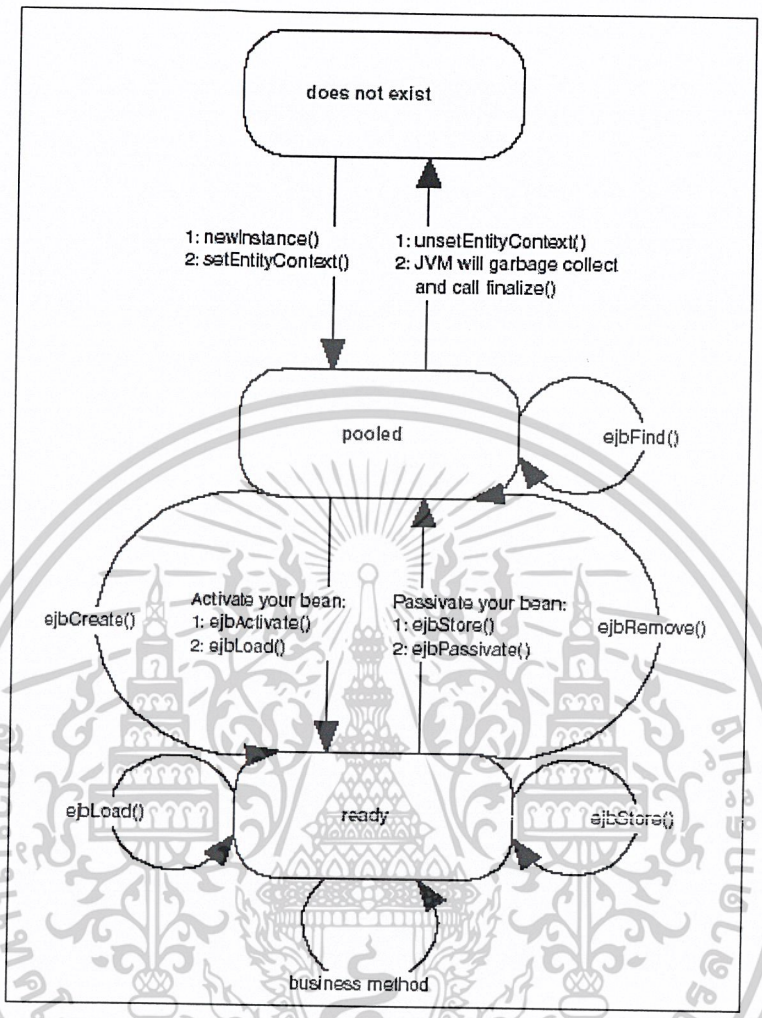
7.2.3.9 Entity Bean สามารถแก้ไขได้โดยไม่ต้องผ่าน EJB

ปกติ การสร้าง , ทำลาย และค้นหาข้อมูลของ entity bean ทำโดยใช้ home interface แต่สามารถติดต่อกับ entity bean ได้อีกวิธีหนึ่ง คือการแก้ไขฐานข้อมูลซึ่งบันทึกข้อมูลนั้นอยู่โดยตรง เช่น ถ้าอินสแทนซ์ของ bean ถูกแมปเข้ากับฐานข้อมูลเชิงสัมพันธ์ จึงสามารถลบแถวออกจากฐานข้อมูลซึ่งสัมพันธ์กับอินสแทนซ์ของ entity bean ได้โดยตรง (รูป 7-8) , สามารถสร้างข้อมูล entity bean ใหม่ และแก้ไขข้อมูลที่มีอยู่แล้ว โดยการทำงานกับฐานข้อมูลโดยตรง การทำเช่นนี้อาจมีความจำเป็นหากระบบงานส่วนหนึ่งเป็นระบบงานเก่าที่แก้ไขข้อมูล โดยการติดต่อกับฐานข้อมูลโดยตรง



รูปที่ 7-8 การแก้ไขฐานข้อมูลที่สัมพันธ์กับ Entity Bean โดยตรง

7.2.4 วงจรชีวิตของ Entity Bean



รูปที่ 7-9 วงจรชีวิตของ Entity Bean

วงจรชีวิตของ entity bean ดังรูปที่ 7-9 ซึ่งจะเห็นว่ามีเมธอดที่เพิ่มเติมขึ้นมาคือ ejbFind() , ejbLoad() และ ejbCreate()

7.3 Entity Beans สามารถถูกค้นหาได้

เนื่องจากว่า ข้อมูลใน entity bean นั้น จะเป็นการระบุอย่างเป็นเอกเทศน์โดยตรงต่อฐานข้อมูล entity bean สามารถถูกค้นหาได้ง่ายพอๆกับการสร้าง การค้นหา entity bean จะคล้ายคลึงกับการทำงานคำสั่ง SELECT ในภาษา SQL จากคำสั่ง SELECT นั้นจะทำการค้นหาข้อมูลทั้งหมดจากฐานข้อมูลเชิงสัมพันธ์(relational database) เมื่อต้องการค้นหา entity bean นั้นก็คือการข้อมูลในฐานข้อมูลจริงๆ ซึ่งจะแตกต่างจาก session bean ตรงที่ session bean นั้นเราจะทำการค้นหาเองไม่ได้ เพราะว่ามันเป็นออบเจกต์แบบถาวร ซึ่งจะคงอยู่และตายไปพร้อมกับการทำงานของไคลเอนต์หนึ่งๆ

การค้นหา entity bean นั้นสามารถทำได้หลายวิธี โดยจะทำการประกาศวิธีเหล่านี้ไว้ใน home interface ซึ่งเมทธอดเหล่านี้จะถูกเรียกว่า finder home interface นั้นจะทำการประกาศเมทธอด finder และรวมไปถึงเมทธอดในการสร้างและทำลาย entity bean นั้นๆ ซึ่งนี่ก็คือข้อแตกต่างที่สำคัญระหว่าง interface ของ entity bean กับ bean ชนิดอื่น เนื่องจากว่า bean ชนิดอื่นๆนั้น จะไม่มี finder เมทธอด

7.4 ตัวอย่างโค้ด entity bean

ในการเขียน entity bean class จะต้องอิมพลีเมนต์อินเทอร์เฟซ javax.ejb.EntityBean อินเทอร์เฟซจะกำหนดเมธอดที่ entity bean class จะต้องอิมพลีเมนต์ อินเทอร์เฟซ javax.ejb.EnterpriseBean ไม่ได้กำหนดเมธอดใด ๆ ไว้ ดังนี้

```
public interface javax.ejb.EnterpriseBean implements
{
}
```

รูปที่ 7-10 อินเทอร์เฟซ javax.ejb.EnterpriseBean

อินเทอร์เฟซ javax.ejb.EntityBean จะกำหนด callback method ที่ bean ต้องอิมพลีเมนต์ ดังนี้

```
public interface javax.ejb.EntityBean implements javax.ejb.EnterpriseBean{
        public abstract void setEntityContext(javax.ejb.EntityContext);
        public abstract void unsetEntityContext();
        public abstract void ejbRemove();
        public abstract void ejbActivate();
        public abstract void ejbPassivate();
        public abstract void ejbLoad();
        public abstract void ejbStore();
}
```

รูปที่ 7-11 การอิมพลีเมนต์ อินเทอร์เฟซ javax.ejb.EnterpriseBean

นอกเหนือจากเมธอดที่เห็น bean จำเป็นที่จะต้องกำหนดเมธอด ejbCreate() ที่ใช้ในการสร้างข้อมูลของ entity bean ใหม่และ ejbFind() ที่ใช้ในการค้นหาข้อมูลของ bean ที่มีอยู่

- ejbCreate()

ejbCreate() ใช้ในการสร้างข้อมูลใหม่ในฐานข้อมูล เมทธอดนี้จะมีหรือไม่มีก็ได้ หากไม่ต้องการสร้างข้อมูลใหม่ขึ้นมาก็ไม่จำเป็นที่จะต้องมีเมธอดนี้ ไม่เหมือนกับ session bean ที่จะบังคับให้มีอย่างน้อย 1 เมทธอด พารามิเตอร์ของเมธอดนี้มีได้หลากหลายขึ้นอยู่กับข้อมูลในฐานข้อมูลที่ต้องการสร้าง และ ejbCreate() จะต้องมีเมธอด create() ที่มีพารามิเตอร์ตรงกับ ejbCreate() นั้นใน home interface

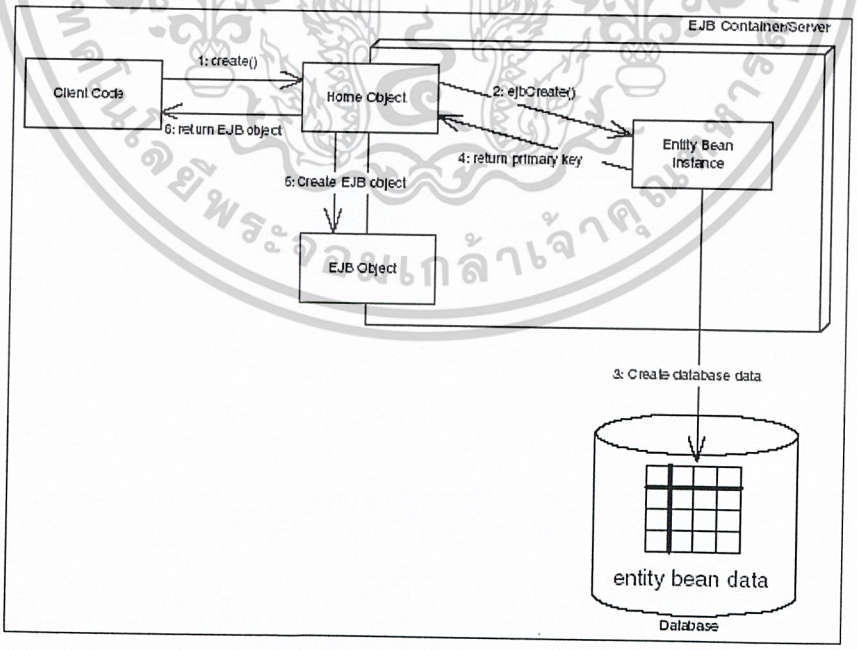
ตัวอย่างเช่น bank account entity bean class ที่ชื่อว่า AccountBean กับ remote interface ที่ชื่อ Account , home interface ชื่อ AccountHome และ primary key class ชื่อ AccountPK โดยให้เมธอด ejbCreate() ใน AccountBean เป็นดังนี้

```
public AccountPK ejbCreate(String accountId,String owner) …
```

จะต้องมีเมธอด create() ใน home interface ดังนี้

```
public Account create(String accountId,String owner) …
```

สังเกตข้อแตกต่างระหว่างค่าที่คืนมาใน 2 เมธอดนี้ จะพบว่าอินสแทนซ์ของ bean จะคืนค่า primary key ขณะที่ home object จะคืนค่า EJB object ที่เป็นเช่นนี้เนื่องจากคอนเทนเนอร์จะคืนค่า primary key ให้กับ home object และ home object จะนำไปสร้าง EJB object แล้วคืนค่า EJB object ให้กับไคลเอ็นต์ ดังรูปที่ 7-12



รูปที่ 7-12 ความสัมพันธ์ระหว่าง ejbCreate() และ create()

- ejbFind()

finder method ใช้ในการหา entity bean ที่อยู่ใน storage โดยสามารถมีได้หลายเมธอดดังนี้

```
public AccountPK ejbFindByPrimaryKey(AccountPK key)
public Enumeration ejbFindAllProducts()
public Enumeration ejbFindBigAccounts(int minimum)
public OrderPK ejbFindMostRecentOrder()
```

finder method มีกฎคือ จะต้องขึ้นด้วย ejbFind , ต้องมีอย่างน้อย 1 เมธอดคือ ejbFindByPrimaryKey , ค่าที่คืนจะต้องเป็น primary key ของ entity bean หรือ enumeration ของ primary key และเหมือนกับ ejbCreate() คือต้องมีเมธอดที่ตอบสนองคู่กับ ejbFind ดังตัวอย่างนี้ finder method ใน AccountBean ดังนี้

```
public Enumeration ejbFindAllProducts() ...
public AccountPK ejbFindBigAccounts(int minimum) ...
```

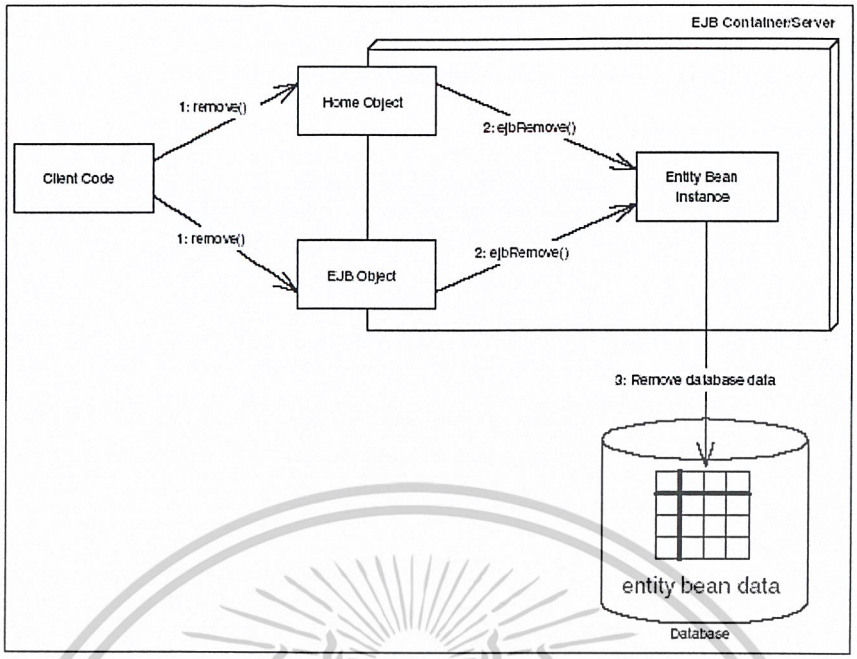
แล้วจะต้องมีเมธอดใน home interface ดังนี้

```
public Enumeration findAllProducts() ...
public Account ejbFindBigAccounts(int minimum) ...
```

สังเกตว่าในกรณีที่ไม่ใช่ enumeration ค่าที่คืนจะต่างกันด้วยเหตุผลเดียวกับ ejbCreate() และมีกฎพิเศษอีกอย่างหนึ่งคือในกรณีที่เป็น container-managed persistence EJB คอนเทนเนอร์จะอิมพลีเมนต์ finder method ใน EJB object ให้โดยอัตโนมัติ

- ejbRemove()

ไคลเอ็นต์จะใช้เมธอด remove() ในการลบข้อมูลของ entity bean ในฐานข้อมูล โดยเรียกเมธอด remove() กับ EJB object หรือ home object แล้วคอนเทนเนอร์จะเรียกเมธอด ejbRemove() กับ bean ดัง รูปที่ 7-13 เมธอดนี้จะไม่ทำลายอินสแทนซ์ของ entity bean แต่จะลบข้อมูลในฐานข้อมูลเท่านั้น



รูปที่ 7-13 การลบข้อมูลของ Entity Bean

7.5 Bean-Managed Persistent Entity Bean (BMP)

entity bean ทั้ง 2 ชนิดจะต้องอิมพลีเมนต์อินเทอร์เฟซ javax.ejb.EntityBean โดยอินเทอร์เฟซจะกำหนด callback method ที่คอนเทนเนอร์จะเรียกใช้กับ bean เมธอดที่ต้องอิมพลีเมนต์นั้นขึ้นอยู่กับชนิดของ entity bean ตารางที่ 7-1 จะอธิบายถึงเมธอดสำหรับ bean-managed persistent

เมธอด	คำอธิบาย
setEntityContext()	เมื่อคอนเทนเนอร์สร้างอินสแทนซ์ของ bean แล้วจะเรียกเมธอดนี้
ejbFind<...>(<...>)	finder method ใช้ในการค้นหาอินสแทนซ์ของ entity bean ที่มีข้อมูลที่ต้องการ โดย bean จะต้องมีเมธอดนี้อย่างน้อยหนึ่งเมธอดคือ ejbFindByPrimaryKey()
EjbCreate (<...>)	เมื่อไคลเอ็นต์เรียกเมธอด create() กับ home interface คอนเทนเนอร์จะเรียกเมธอด ejbCreate() กับอินสแทนซ์ของ bean ที่พูลอยู่ ejbCreate() จะทำหน้าที่ในการสร้างข้อมูลใหม่ในฐานข้อมูลให้
ejbPostCreate(<...>)	bean class จะต้องกำหนดเมธอด ejbPostCreate() ให้เข้าคู่กับ ejbCreate() คือต้องมีพารามิเตอร์เหมือนกัน คอนเทนเนอร์จะเรียกเมธอดนี้หลังจากเรียก ejbCreate()
ejbLoad()	คอนเทนเนอร์จะเรียกเมธอดนี้เพื่อโหลดข้อมูลจากฐานข้อมูลไปยังอินสแทนซ์ของ bean

ตารางที่ 7-1 คำอธิบายเมธอดใน Bean-Managed Persistent Entity Bean

ejbStore()	คอนเทนเนอร์จะเรียกเมธอดนี้เพื่ออัพเดตข้อมูลจากอินสแทนซ์ของ bean ลงฐานข้อมูล
ejbPassivate()	คอนเทนเนอร์จะเรียกเมธอดนี้เมื่อนำ entity bean เก็บลงในพูล
ejbRemove()	จะลบข้อมูลในฐานข้อมูล
unsetEntityContext()	เมธอดนี้ใช้ในการลบ bean ออกจากเอ็นไวรอนเมนต์ คอนเทนเนอร์จะเรียกเมธอดนี้ก่อนที่อินสแทนซ์ของ entity bean จะถูกทำลาย

ตารางที่ 7-1 คำอธิบายเมธอดใน Bean-Managed Persistent Entity Bean (ต่อ)

7.6 Container-Managed Persistent Entity Bean (CMP)

เมธอดที่ต้องอิมพลีเมนต์ของ container-managed persistent ต่างจาก bean-managed persistent ดังตารางที่ 7-2

เมธอด	คำอธิบาย
setEntityContext() (เหมือนกับ bean-managed persistent)	เมื่อคอนเทนเนอร์สร้างอินสแทนซ์ของ bean แล้วจะเรียกเมธอดนี้
ejbFind<...>(<...>)	ไม่จำเป็นต้องอิมพลีเมนต์ finder method คอนเทนเนอร์จะจัดการให้ โดยวิธีในการระบุเพื่อให้คอนเทนเนอร์รู้วิธีแมปนั้นขึ้นอยู่กับผลิตภัณฑ์
EjbCreate(<...>)	เมื่อไคลเอ็นต์เรียกเมธอด create() กับ home interface คอนเทนเนอร์จะเรียกเมธอด ejbCreate() กับอินสแทนซ์ของ bean ที่พูลอยู่ ejbCreate() จะทำหน้าที่ในการสร้างข้อมูลใหม่ในฐานข้อมูลให้ เมธอดนี้เหมือนกับ bean-managed persistent แต่กับ container-managed persistent ไม่จำเป็นที่จะต้องอิมพลีเมนต์เมธอดนี้
ejbPostCreate(<...>) (เหมือนกับ bean-managed persistent)	bean class จะต้องกำหนดเมธอด ejbPostCreate() ให้เข้าคู่กับ ejbCreate() คือต้องมีพารามิเตอร์เหมือนกัน คอนเทนเนอร์จะเรียกเมธอดนี้หลังจากเรียก ejbCreate()
ejbActivate() (เหมือนกับ bean-managed persistent)	เมื่อไคลเอ็นต์ต้องการเรียกบิซิเนสเมธอดกับ EJB object แต่ไม่มีอินสแทนซ์ของ entity bean นั้นที่ bind กับ EJB object คอนเทนเนอร์จะนำ bean จากพูลและเปลี่ยนสถานะเป็น ready โดยเมธอดนี้จะไม่ถูกเรียกระหว่างทรานแซ็กชัน
ejbLoad()	คอนเทนเนอร์จะเรียกเมธอดนี้เพื่อโหลดข้อมูลจากฐานข้อมูลไปยังอินสแทนซ์ของ bean ไม่ต้องอิมพลีเมนต์เมธอดนี้

ตารางที่ 7-2 คำอธิบายเมธอดใน Container-Managed Persistent Entity Bean

ejbStore()	คอนเทนเนอร์จะเรียกเมธอดนี้เพื่ออัพเดตข้อมูลจากอินสแทนซ์ของ bean ลงฐานข้อมูล ไม่ต้องอิมพลีเมนต์เมธอดนี้
ejbPassivate() (เหมือนกับ bean-managed persistent)	คอนเทนเนอร์จะเรียกเมธอดนี้เมื่อนำ entity bean เก็บลงในพูล
ejbRemove()	จะลบข้อมูลในฐานข้อมูล ไม่ต้องอิมพลีเมนต์เมธอดนี้
unsetEntityContext() (เหมือนกับ bean-managed persistent)	เมธอดนี้ใช้ในการลบ bean ออกจากเอ็นไวรอนเมนต์ คอนเทนเนอร์จะเรียกเมธอดนี้ก่อนที่อินสแทนซ์ของ entity bean จะถูกทำลาย

ตารางที่ 7-2 คำอธิบายเมธอดใน Container-Managed Persistent Entity Bean (ต่อ)

7.7 เปรียบเทียบ BMP กับ CMP Entity Bean

การเลือกระหว่าง entity bean 2 ชนิดนี้ขึ้นอยู่กับงานที่จะทำ เนื่องจากทั้ง 2 ชนิดนี้มีข้อดีและข้อเสียต่างกัน โดยจะเปรียบเทียบในหัวข้อต่าง ๆ ดังนี้

1) จำนวนโค้ดและเวลาในการพัฒนา

container-managed persistent มีจำนวนโค้ดน้อยกว่าและพัฒนาได้เร็วกว่า เนื่องจากไม่ต้องเขียนโค้ดแมปข้อมูลด้วย JDBC

2) Bugs

การดีบัก container-managed persistent ยากกว่า bean-managed persistent เนื่องจาก bean-managed persistent เป็นการควบคุมโค้ด JDBC ทั้งหมดทำให้ง่ายต่อการตรวจสอบโค้ดเหล่านั้น แต่ container-managed persistent จะช่วยลดการเกิดบักได้มากกว่า bean-managed persistent ที่ต้องเขียนโค้ดเองทั้งหมด

3) การควบคุม

bean-managed persistent ควบคุมผ่านทาง JDBC จึงมีความยืดหยุ่นในการแมปออบเจ็กต์ไปยังฐานข้อมูล สำหรับ container-managed persistent ขึ้นอยู่กับว่าคอนเทนเนอร์รองรับการแมปที่ซับซ้อนแค่ไหน

4) ความเป็นอิสระจากแอพพลิเคชันเซิร์ฟเวอร์และฐานข้อมูล

container-managed persistent เป็นอิสระจากฐานข้อมูล เนื่องจากไม่มีการโค้ด JDBC สามารถจะเปลี่ยนแปลงได้ง่าย โดยอาจเปลี่ยนจากฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุโดยไม่ต้องแก้ไขโค้ดใด ๆ แต่การแมปจากออบเจ็กต์ไปยังฐานข้อมูลไม่มีมาตรฐานที่แน่นอน จึงไม่ portable กับแอพพลิเคชันเซิร์ฟเวอร์ใด ๆ ส่วน bean-managed persistent นั้น portable กับแอพพลิเคชันเซิร์ฟเวอร์และฐานข้อมูลที่เป็นกลางคือรองรับ SQL มาตรฐาน

5) การเรียนรู้และค่าใช้จ่าย

นักพัฒนาส่วนใหญ่จะมีความเข้าใจในการทำงานกับฐานข้อมูลเชิงสัมพันธ์ ดังนั้นการใช้ bean-managed persistent จึงไม่ต้องการการเรียนรู้ แต่กับ container-managed persistent จะต้องมีการเรียนรู้วิธีการแมปของผลิตภัณฑ์ใด ๆ ที่ใช้ รวมถึงค่าใช้จ่ายในการเรียนรู้วิธีการแมปในแต่ละผลิตภัณฑ์ของแอพพลิเคชันเซิร์ฟเวอร์

บทที่ 8
แอพพลิเคชั่น Home&Décor

8.1 ภาพรวมของระบบ Olala Home Services

Olala Home Services เป็นระบบจัดการอำนวยความสะดวกในการซื้อบ้าน ที่ดิน และเฟอร์นิเจอร์ตกแต่งบ้าน ซึ่งเป็นเวบเซอร์วิส ที่ให้บริการ จัดหาบ้าน คอนโดมิเนียมที่สร้างเสร็จแล้ว หรือที่ดิน ตามแหล่งที่อยู่อาศัยที่ผู้ใช้ต้องการ การให้บริการเลือกแบบบ้านและก่อสร้างบ้านตามแบบที่ผู้ใช้ระบบต้องการ อีกทั้งยังอำนวยความสะดวกแก่ผู้ใช้งานระบบด้วยเซอร์วิสที่ให้บริการ การเลือกซื้อเฟอร์นิเจอร์ตกแต่งบ้าน โดยการกรอกรายละเอียดที่จำเป็นในการค้นหาลงไป เช่น ผู้ใช้ระบบที่ต้องการซื้อบ้านตามแหล่งที่อยู่ที่ต้องการ คือ แหล่งที่อยู่อาศัยจะระบุโดย จังหวัด เขต หรือ ซอยที่ต้องการ หรือหากผู้ใช้งานระบบมีที่ดินอยู่แล้ว แต่ต้องการจะสร้างบ้าน ก็สามารถเข้ามาเลือกแบบบ้านที่ต้องการได้ เมื่อได้แบบบ้าน หรือ ที่อยู่ที่ต้องการแล้ว ผู้ใช้ยังสามารถเลือกซื้อเครื่องใช้ตกแต่งบ้านได้ผ่านทางระบบ จากนั้นระบบจะทำการไปเรียก ใช้เซอร์วิสต่างๆ ที่เปิดให้บริการไว้

เมื่อได้รับผลลัพธ์กลับมาแล้ว ระบบจะต้องรอการทำงานจากผู้ใช้งานต่อว่า จะทำการจอง หรือสั่งซื้อสินค้า แต่หากผู้ใช้งานระบบเลือกทำการผิดพลาด ก็สามารถแก้ไขรายการที่ทำไปได้ เมื่อทำการสั่งซื้อสินค้าแล้ว ระบบจะทำการออกใบเสร็จมาให้ผู้ใช้งานระบบ หลังจากนี้ผู้ใช้งานจะไม่สามารถแก้ไขรายการใดๆกับทางระบบ Olala Home Services ได้อีก

ปริญญานิพนธ์ฉบับนี้ ออกแบบการให้บริการออกเป็น 2 เทคโนโลยี คือ เทคโนโลยีของ .NET เฟรมเวิร์ค และ เทคโนโลยีของจาวา เพื่อทำการศึกษาเรื่องการเรียกใช้งานคอมโพเนนต์ข้ามสถาปัตยกรรม โดยทำการติดต่อสื่อสารกันผ่านโปรโตคอล SOAP

ระบบที่ให้บริการโดยใช้เทคโนโลยีจาวา ได้แก่

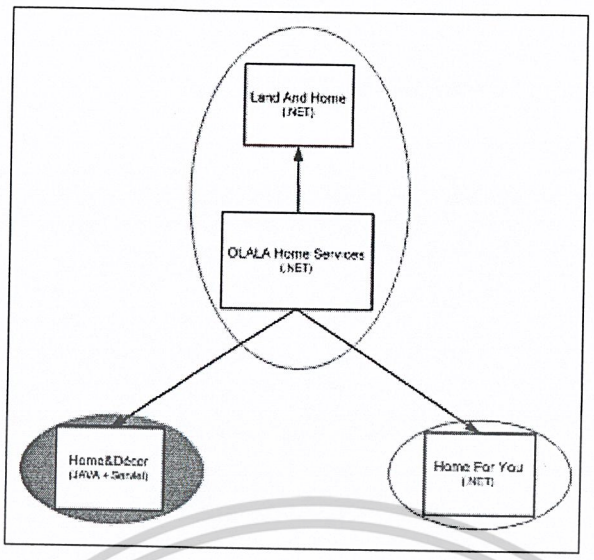
Home&Décor Service

ธนาคาร

ระบบที่ให้บริการโดยใช้เทคโนโลยี .NET เฟรมเวิร์ค ได้แก่

Home For You Service

Land And House Service



รูปที่ 8-1 ภาพรวมของระบบ Olala Home Services

8.2 รายละเอียดของแอพพลิเคชั่น Home&Décor

แอพพลิเคชั่น Home&Décor เป็นระบบที่ให้บริการเกี่ยวกับการเลือกซื้อสินค้าตกแต่งบ้าน ทั้งผ่านทางหน้าเวบโดยตรง และผ่านทางเวบเซอร์วิส โดยสินค้าที่นำเสนอผ่านทางเวบนั้น จะมีการนำความรู้ในด้านเทคโนโลยี VRML มาประกอบด้วย ดังนั้น สินค้าที่แสดงจะมีทั้งที่เป็นรูปสองมิติ และ ที่เป็นสามมิติ โดยใช้เทคโนโลยี VRML ซึ่งรายการสินค้าที่แสดงในแต่ละหมวดหมู่นั้นจะเป็นการสุ่มขึ้นมาแสดง นั่นคือการเข้ามาชมสินค้าแต่ละครั้ง จะเห็นสินค้าที่แสดงเหมือนเดิม หรือ ต่างจากเดิมก็ได้

บริการของระบบ Home&Décor ประกอบด้วย

1. การลงทะเบียน
2. การค้นหาข้อมูลสินค้า ดังหัวข้อต่อไปนี้
   2.1. ค้นหาเครื่องใช้ตามหมวดหมู่(แผนก)สินค้า
   2.2. ค้นหาเครื่องใช้ตามคำที่ต้องการ
3. การดูรายการสินค้าที่สั่งซื้อ
4. การแก้ไขรายการสั่งซื้อที่ทำกับระบบก่อนมีการชำระเงิน
5. การสั่งซื้อสินค้าผ่านทางระบบ
6. เกร็ดความรู้ในการตกแต่งบ้าน เป็นการดูเกร็ดความรู้ผ่านทางเวบเพจ Home&Décor เท่านั้น ไม่ให้บริการผ่านเวบเซอร์วิส
7. การทำงานในส่วนของผู้ดูแลระบบ

1. การลงทะเบียน

เมื่อผู้ใช้งานระบบเป็นครั้งแรก ที่ต้องการสั่งซื้อสินค้าผ่านทางระบบ จะต้องสมัครสมาชิกกับทางระบบก่อน หากต้องการเพียงแค่ค้นหาสินค้า หรือ อ่านเกร็ดความรู้ในการตกแต่งบ้าน ไม่จำเป็นจะต้องเป็นสมาชิกกับทางระบบ

การลงทะเบียนนี้ สนับสนุนทั้งการลงทะเบียนผ่านทางหน้าเวบเพจโดยตรง และ ลงทะเบียนผ่านการใช้งานเวบเซอร์วิส โดยระบบกลาง(Olala Home Services)

2. การค้นหาข้อมูลสินค้า

ผ่านทางเวบเซอร์วิส

ผู้ใช้ระบบจะเรียกใช้การค้นหาผ่านทางเวบกลาง โดยการค้นหาสินค้าเฟอร์นิเจอร์ตามคำที่ผู้ใช้งานระบบต้องการ แล้วทางระบบกลางจะเรียกใช้งานเซอร์วิสของระบบ Home&Décor ในการค้นหาสินค้า

ผ่านทางเวบเพจ Home&Décor

ผู้ใช้ระบบสามารถค้นหารายการสินค้าที่ต้องการได้ ทั้งผ่านทางวิธีการค้นหา (Search) และ ผ่านการค้นหาเอง โดยการเข้าไปดูสินค้าในแต่ละหมวด ตามที่ผู้ใช้งานต้องการ

3. การดูรายการสินค้าที่สั่งซื้อ

เมื่อมีการเลือกสินค้าที่ต้องการสั่งซื้อแล้ว ผู้ใช้งานระบบต้องการตรวจสอบรายการสินค้าทั้งหมดก่อนที่จะทำการสั่งซื้อ หากไม่ถูกต้องตรงตามที่ต้องการ ผู้ใช้งานระบบสามารถแก้ไขรายการที่ได้เลือกไปแล้วได้

4. การแก้ไขรายการสั่งซื้อที่ทำกับระบบก่อนมีการชำระเงิน

หากมีการตกลงชำระเงินไปแล้ว แต่มาพบทีหลังว่ารายการผิดพลาด จะไม่สามารถแก้ไขรายการผ่านทางเวบเพจได้ ต้องติดต่อมายังผู้ดูแลระบบโดยตรง แต่หากยังไม่มีการตกลงชำระเงิน ผู้ใช้งานระบบสามารถแก้ไขรายละเอียดการสั่งซื้อได้ตลอด ทั้งจำนวนสินค้าที่ต้องการ และ เลือกรายการเพิ่มเตอม หรือลบรายการใดทิ้ง

5. การสั่งซื้อสินค้าผ่านทางระบบ

ทำเมื่อมีการสั่งซื้อสินค้ากับทางระบบ แล้วระบบจะทำการออกใบชำระเงิน และ ใบรับสินค้าให้กับผู้ใช้งานระบบ เมื่อผู้ใช้งานระบบติดต่อผ่านทาง Olala Home Services หรือผ่านทางเวบเพจโดยตรงก็ได้

รูปแบบของการชำระเงิน จะเป็นการชำระผ่านบัตรเครดิต โดยผู้ใช้ระบบสามารถเลือกได้ว่า จะชำระเงินด้วยบัตรใด และสามารถเลือกที่อยู่ที่ส่งสินค้าได้

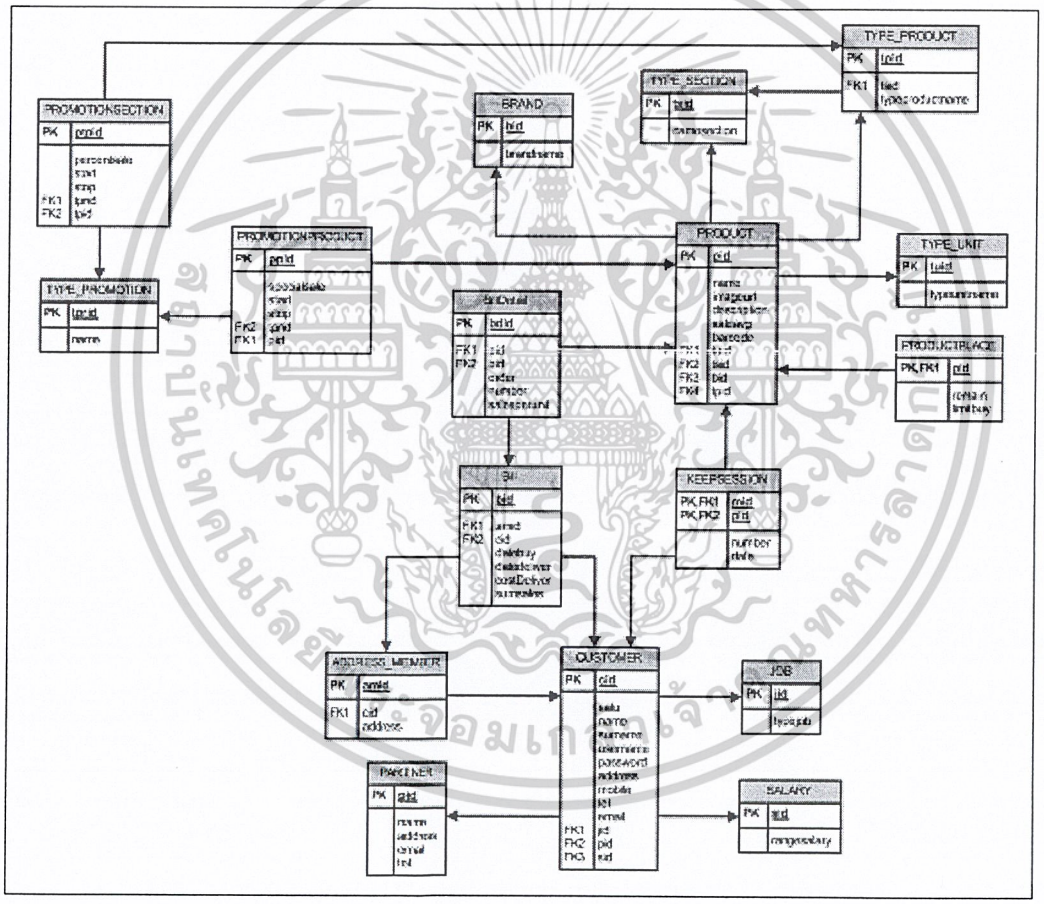
6. เกร็ดความรู้ในการตกแต่งบ้าน

เป็นบริการที่มีเฉพาะ ผู้ใช้ที่ติดต่อผ่านทางเวบเพจ Home&Décor เท่านั้น หากติดต่อผ่านเวบกลาง จะไม่สามารถใช้บริการนี้ได้ จะทำได้เพียงค้นหาสินค้า และสั่งซื้อสินค้า

7. การทำงานในส่วนของผู้ดูแลระบบ

การทำงานในส่วนของผู้ดูแล จะเป็นการจัดการเกี่ยวกับสินค้าในระบบทั้งสินค้าใหม่ และ ข้อมูลต่างๆที่เกี่ยวกับสินค้า, เกร็ดความรู้ต่างๆ รวมไปถึง การจัดการเกี่ยวกับผู้ดูแล ที่มีทั้งสิทธิ์ปืนผู้ดูแลทั่วไป และ Administrator

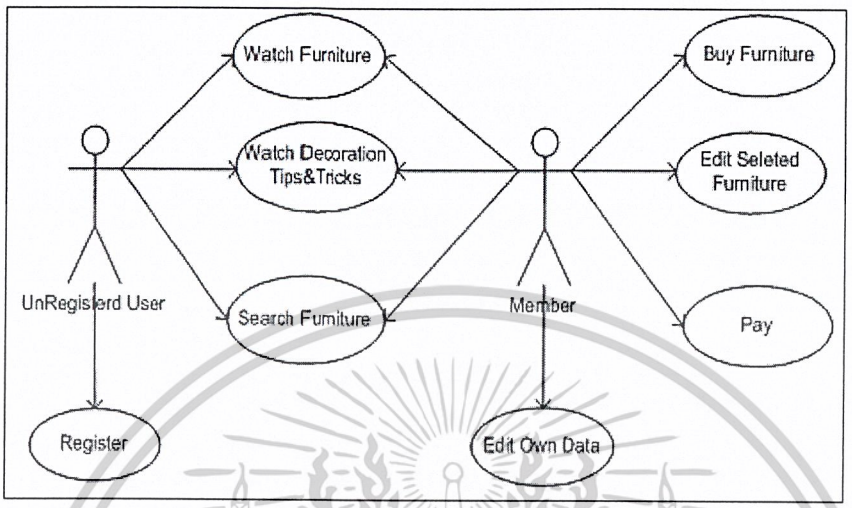
8.3 การออกแบบฐานข้อมูล (Database Design)



รูปที่ 8-2 ฐานข้อมูลของระบบ Home&Decor

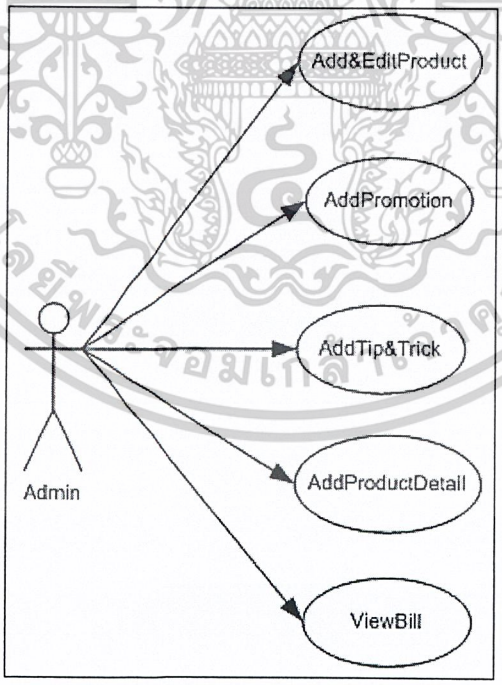
8.4 การออกแบบ USE CASE Diagram

8.4.1 USE CASE ของผู้ใช้งานระบบทั่วไป รวมไปถึงผู้ใช้งานผ่านทางเวบเซอร์วิสด้วย



รูปที่ 8-3 เซอร์วิสที่ระบบให้บริการแก่ผู้ใช้งานระบบ

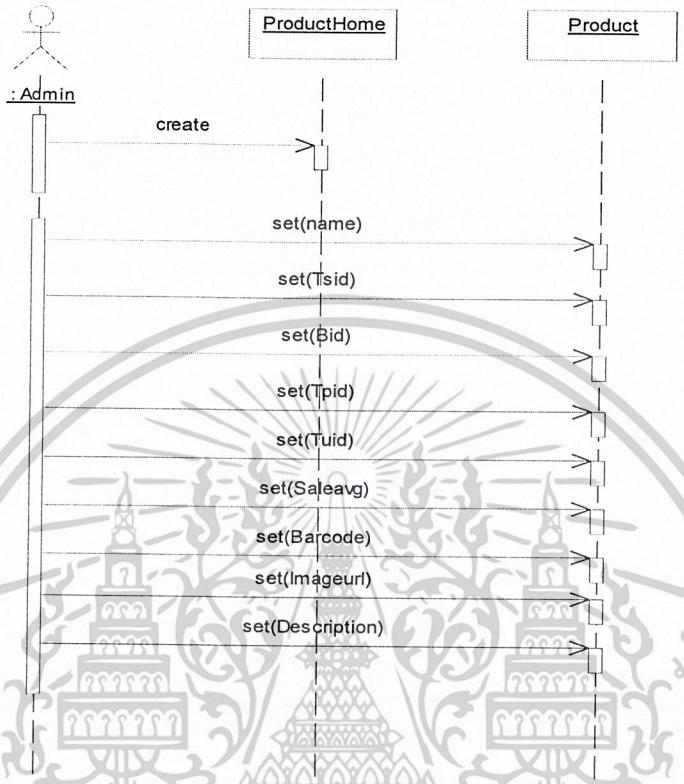
8.4.2 USE CASE ของผู้ดูแลระบบ



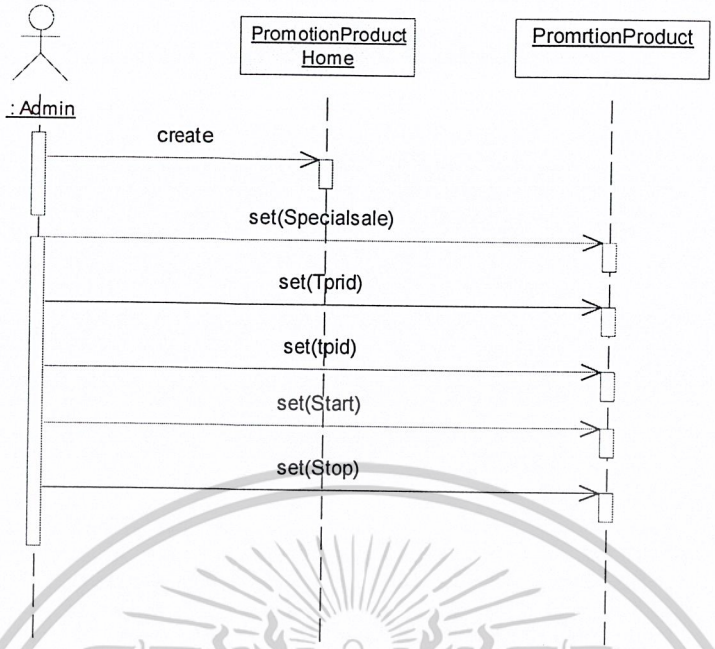
รูปที่ 8-4 การทำงานของผู้ดูแลระบบ

8.5 Sequence Diagram

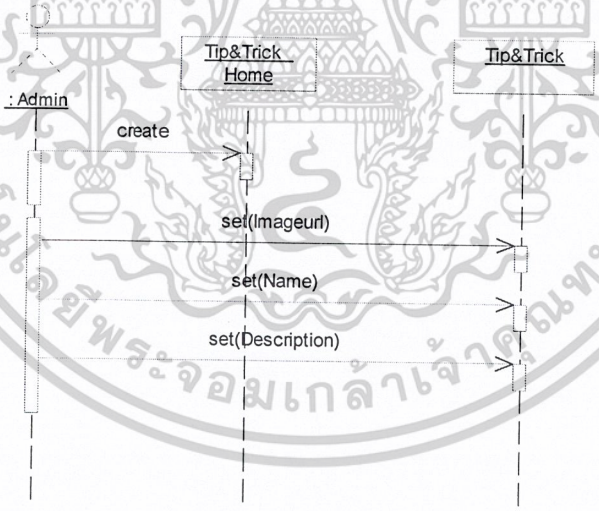
8.5.1 ผู้ดูแลระบบ



รูปที่ 8-5 ลำดับการทำงานของการเพิ่มเติมสินค้าในระบบผ่านทางเวบโดยผู้ดูแลระบบ

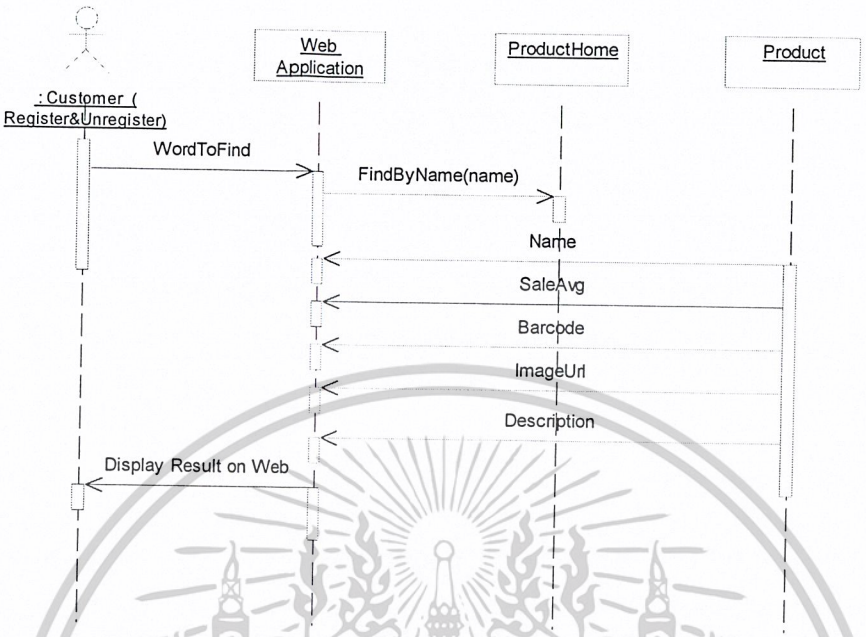


รูปที่ 8-6 การเพิ่มเติมโปรโมชั่นของสินค้าในระบบ

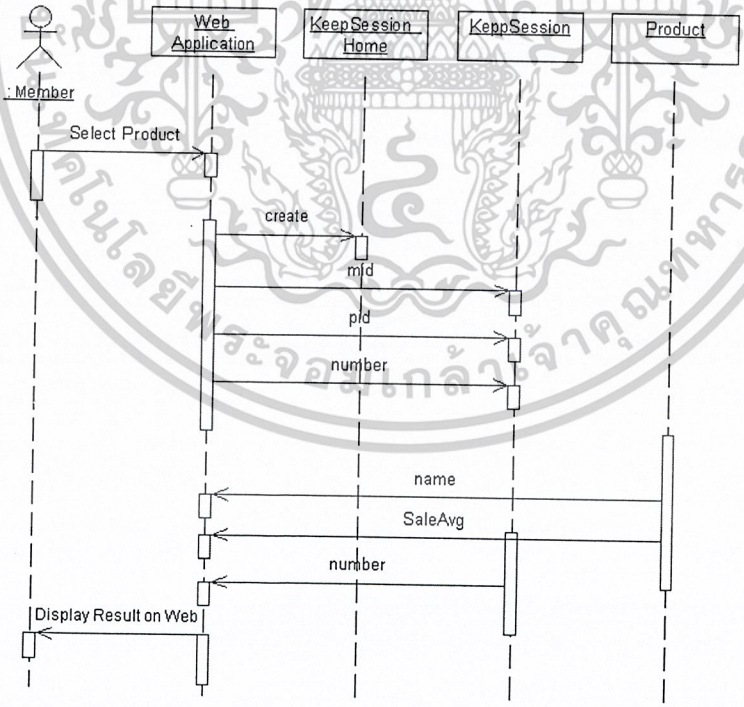


รูปที่ 8-7 การเพิ่มเติมเกร็ดความรู้ในการตกแต่งบ้าน

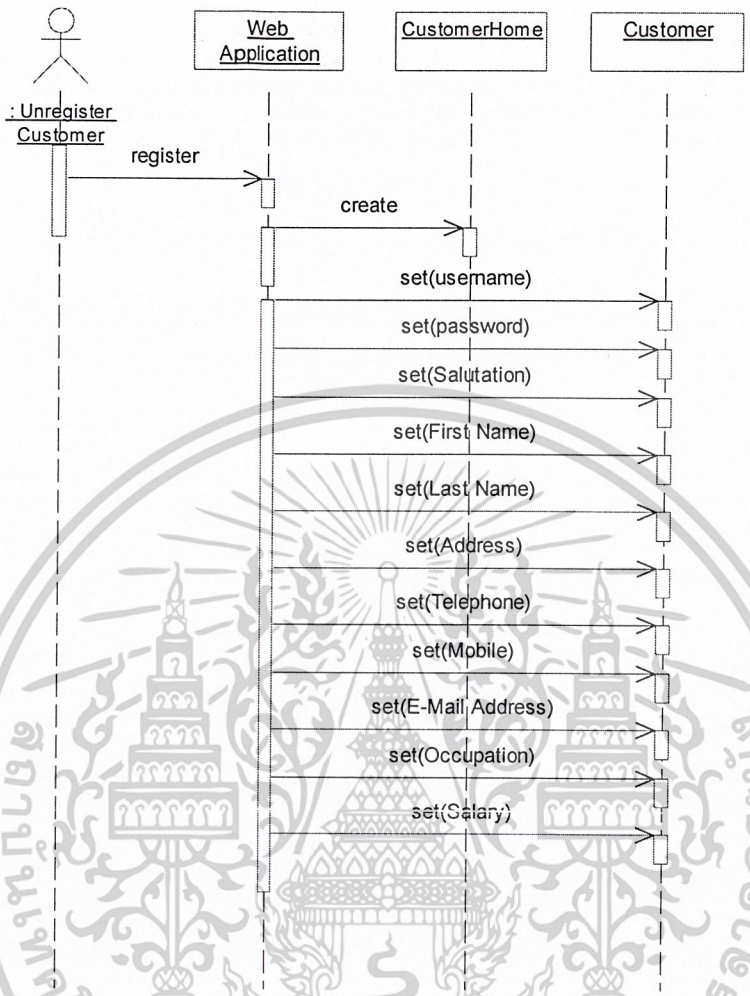
8.5.2 ผู้ใช้งานระบบ



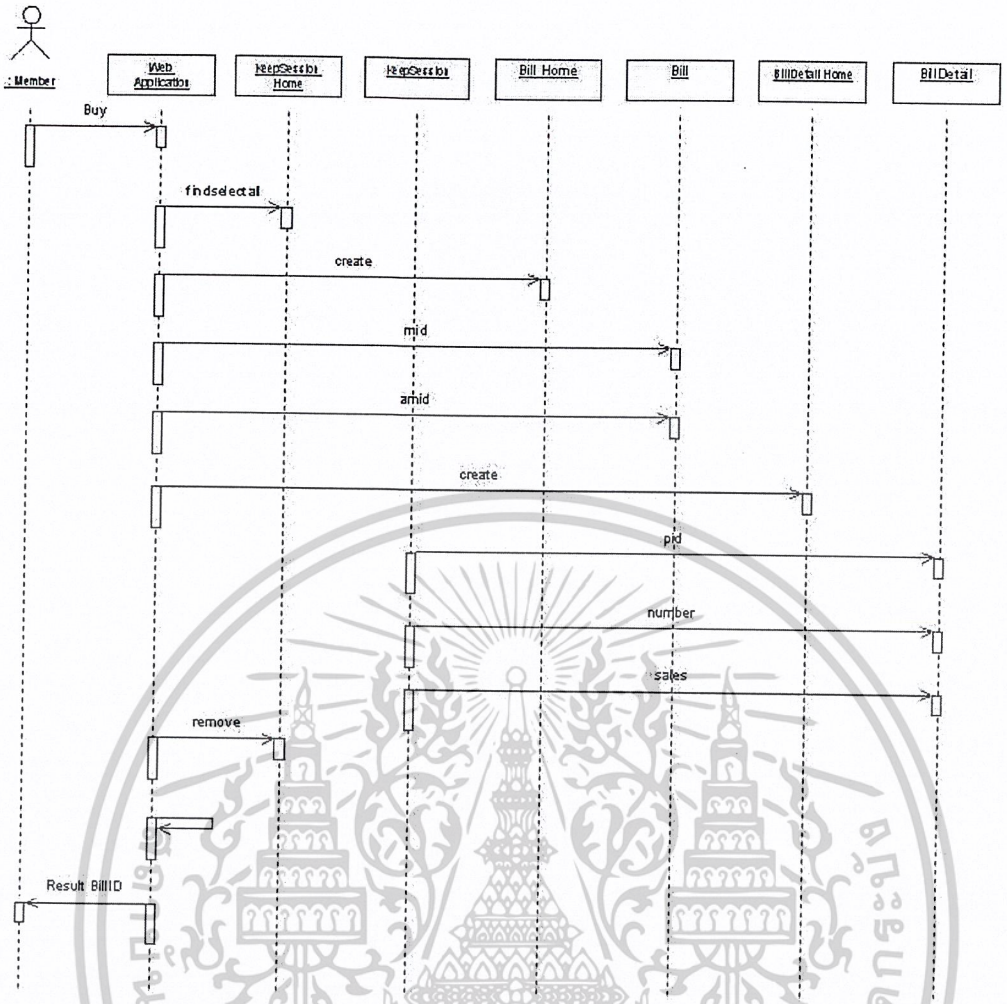
รูปที่ 8-8 ลำดับการค้นหารายการสินค้า



รูปที่ 8-9 แสดงรายการสินค้าที่เลือกซื้อ



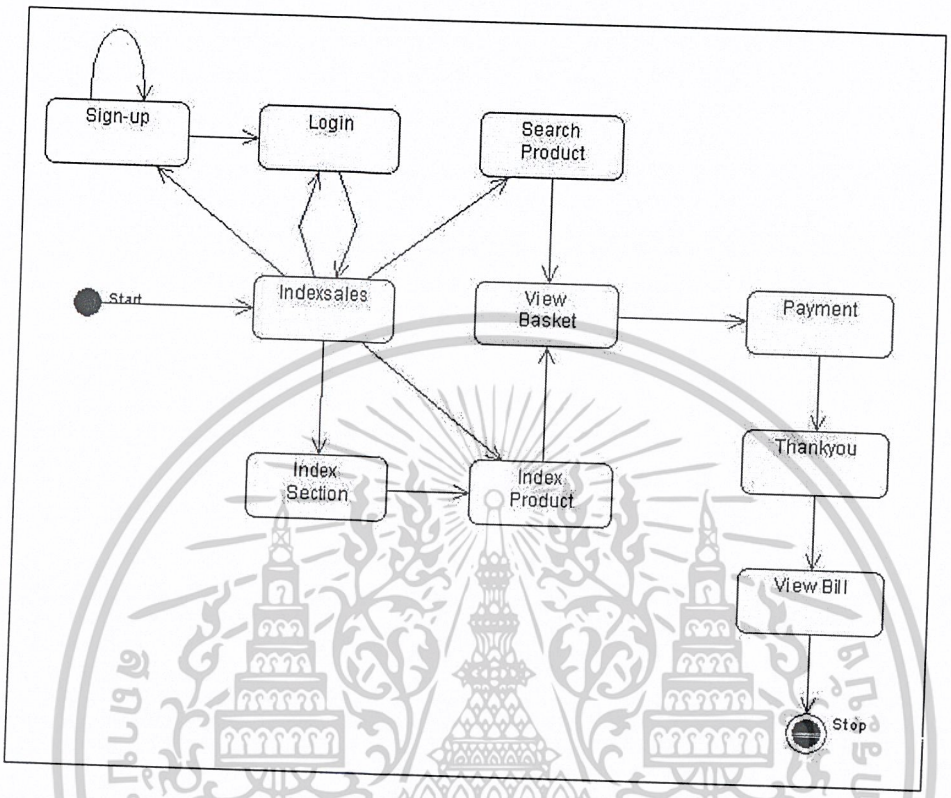
รูปที่ 8-10 การสมัครสมาชิกกับทางระบบ



รูปที่ 8-11 ลำดับการซื้อสินค้า

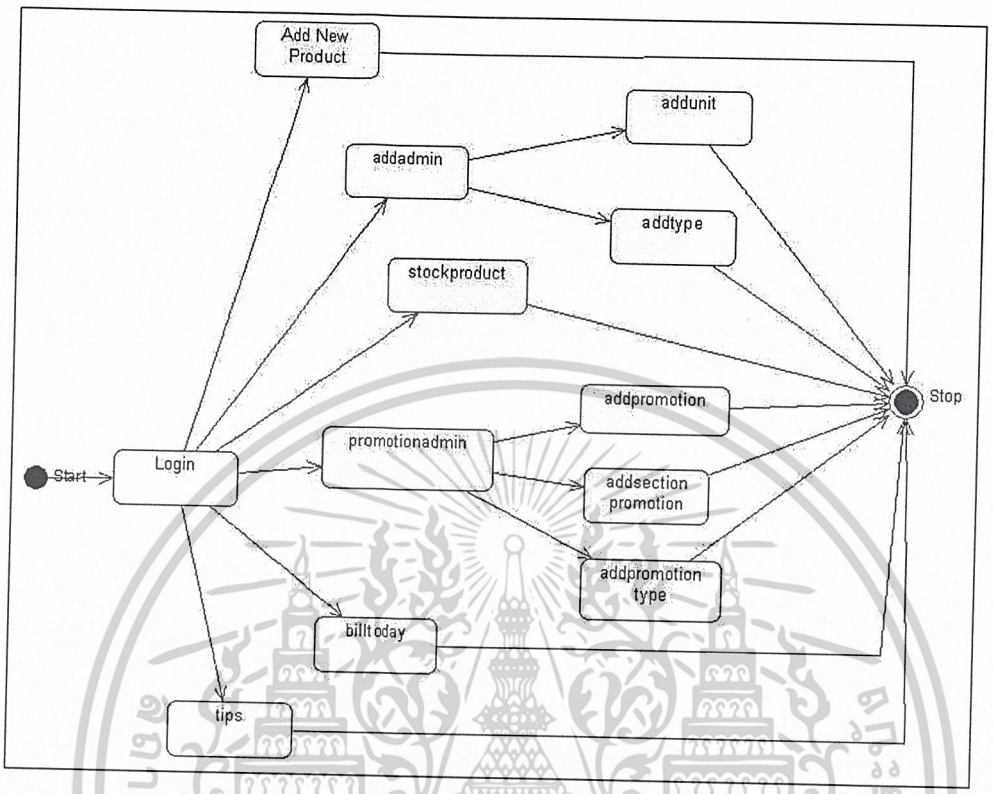
8.6 State Diagram

8.6.1 ผู้ใช้งานระบบ



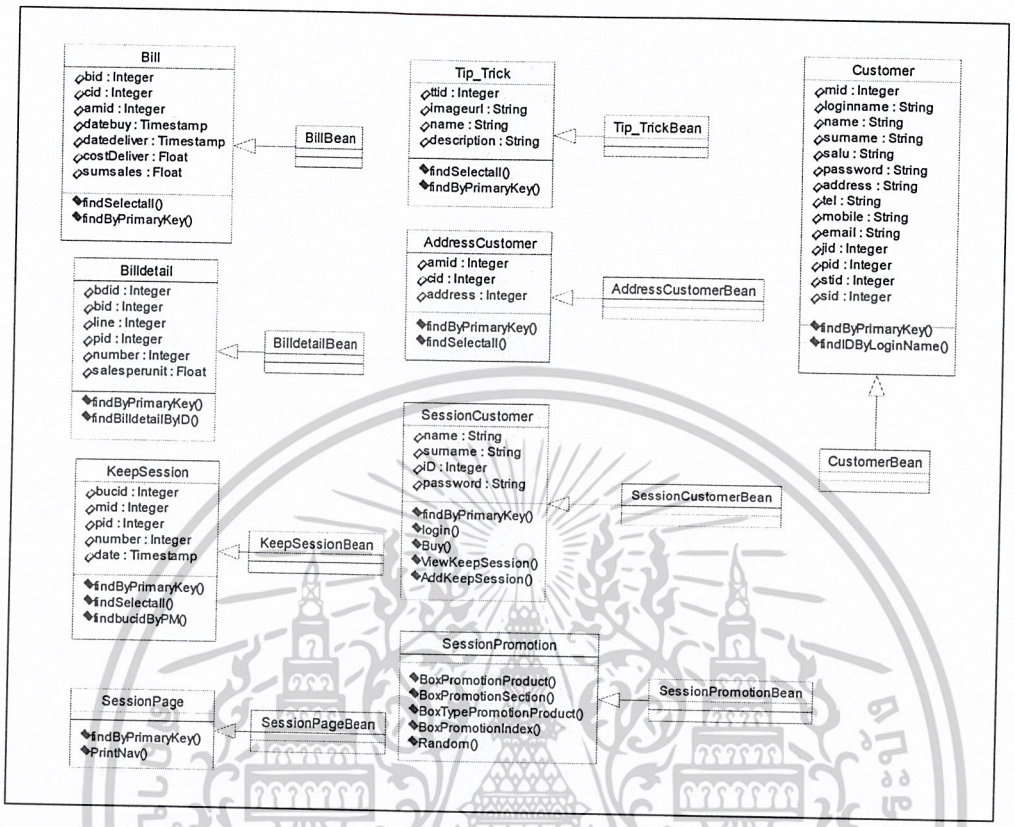
รูปที่ 8-12 State Diagram ของผู้ใช้งานระบบ

8.6.2 ผู้ดูแลระบบ

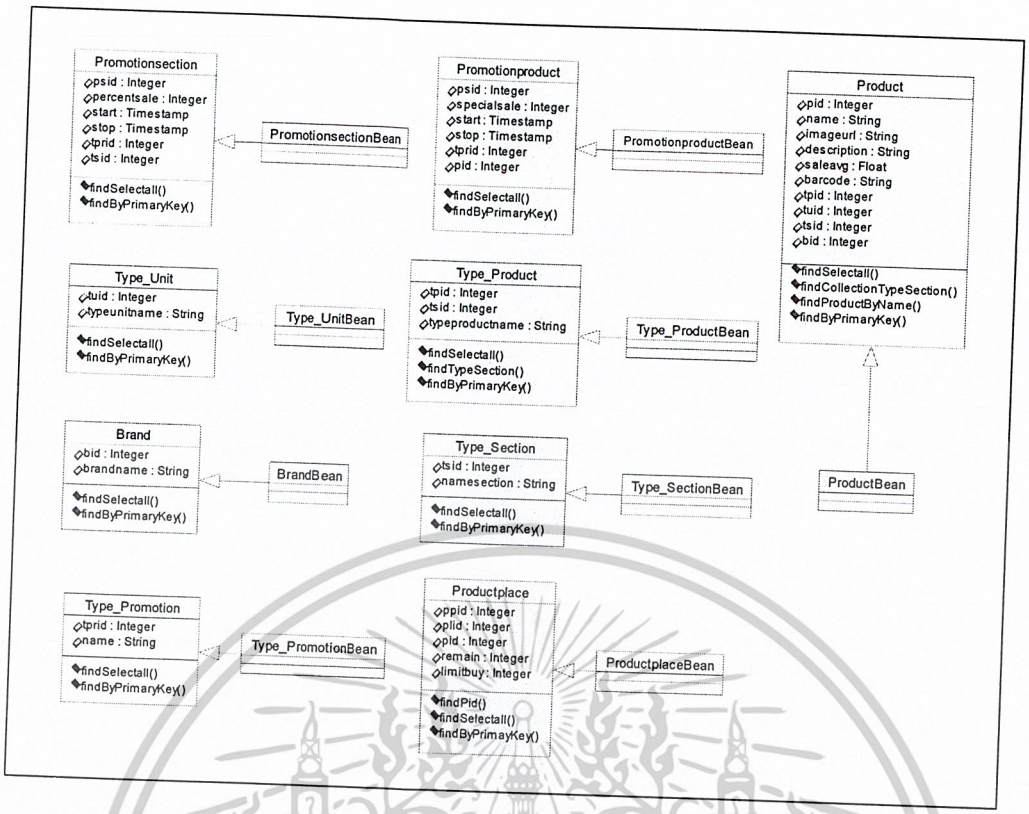


รูปที่ 8-13 State Diagram ของผู้ดูแลระบบ

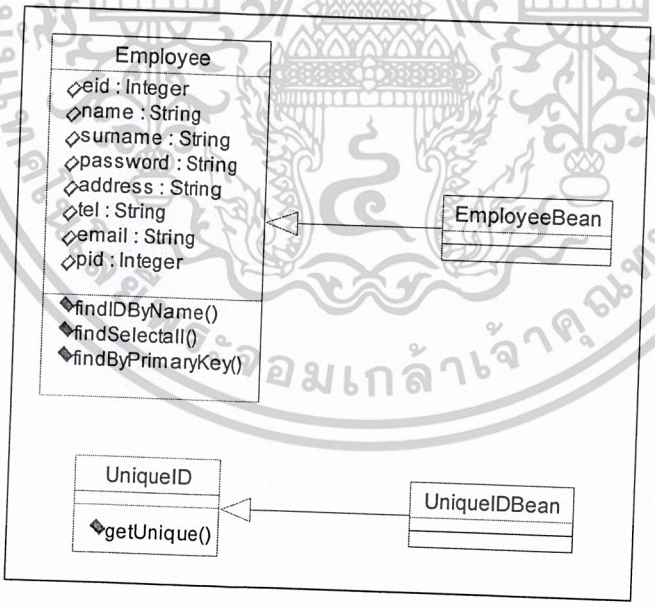
8.7 Class Diagram



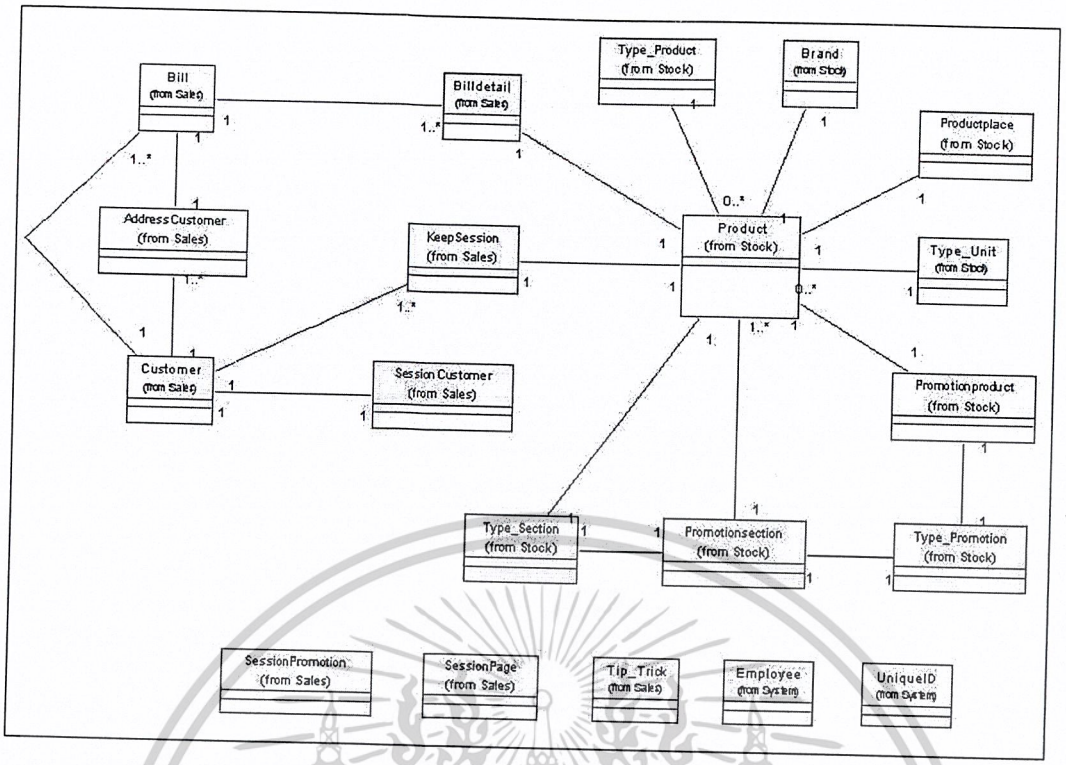
รูปที่ 8-14 Class Diagram ของแอพพลิเคชั่น



รูปที่ 8-15 Class Diagram ของแอพพลิเคชั่น



รูปที่ 8-16 Class Diagram ของส่วนผู้ดูแลระบบ



รูปที่ 8-17 Class Diagram

8.8 รายละเอียดฟังก์ชัน Home&Décor

8.8.1 การสมัครสมาชิกของลูกค้าใหม่

Function : Boolean Signup(String salu, String name, String surname, String loginname, String password, String address, String mobile)

Output : ถ้าสำเร็จ return true
ถ้าไม่สำเร็จ return false

8.8.2 เป็นการตรวจสอบว่า ชื่อและพาสเวิร์ดได้มีคนสมัครไปแล้วหรือยัง

Function : Boolean Login (String username, String password)

Output : ถ้าสำเร็จ return true
ถ้าไม่สำเร็จ return false

8.8.3 การหารหัสของสินค้าที่ชื่อที่เราต้องการทราบ

Function : String[] SearchProduct (String productname)

Output : ถ้าสำเร็จ return Array of IDProduct
ถ้าไม่สำเร็จ return null

8.8.4 ดูรายละเอียดของสินค้าที่ต้องการ

Function : String[] ViewDetailProduct (String productid)

Output : ถ้าสำเร็จ return output[0] = String.valueOf(premote.getPid())
output[1] = premote.getName()
output[2] = String.valueOf(premote.getSaleavg())
output[3] = premote.getBarcode()
output[4] = premote.getImageurl()
output[5] = premote.getDescription()
...

ถ้าไม่สำเร็จ return null

8.8.5 เพิ่มรายการสินค้าในรายการสินค้า

Function : Boolean AddKeepSession(String username, String password, String pid)

Output : ถ้าสำเร็จ return true

ถ้าไม่สำเร็จ return false

8.8.6 ดูรายการสินค้าก่อนจะสั่งซื้อสินค้า

Function : String[][] ViewKeepSession(String username, String password)

Output : ถ้าสำเร็จ return output[0][0] = ProductID
output[0][1] = ProductName
output[0][2] = Number of Buy
output[0][3] = Price
... ...

ถ้าไม่สำเร็จ output[0][0] = "NO!"

8.8.7 ทำการสั่งซื้อสินค้าตามรายการ

Function : Boolean AddBuyProduct(String username, String password, String costdeliver, String amid)

Output : ถ้าสำเร็จ return true

ถ้าไม่สำเร็จ return false

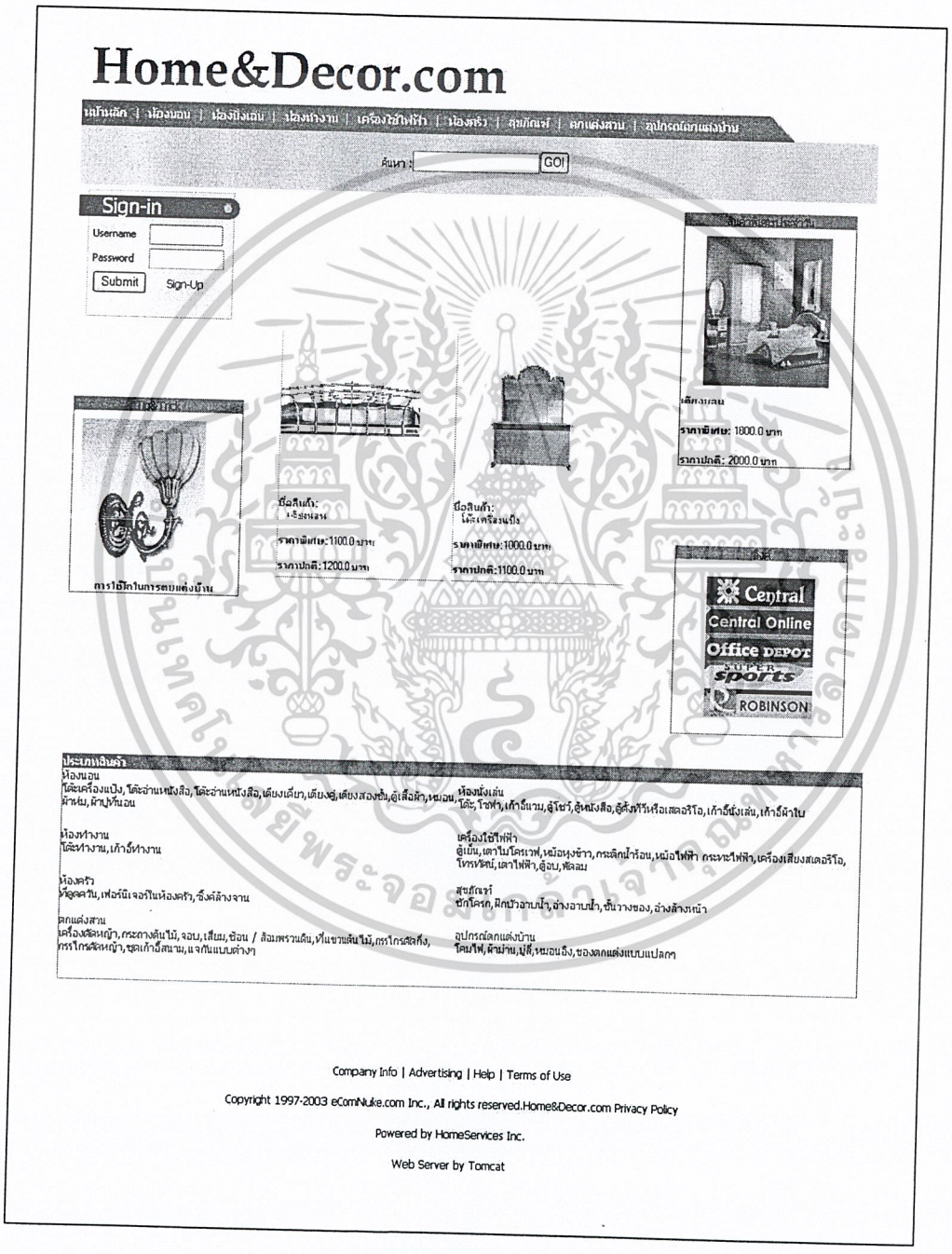
8.8.8 ดูใบเสร็จสินค้าที่ต้องชำระเงิน

Function : String[] ViewBill(String username, String password)

Output : ถ้าสำเร็จ return Link to Bill of Home and Décor

ถ้าไม่สำเร็จ return null

8.9 คู่มือการใช้งาน Application Home&Décor

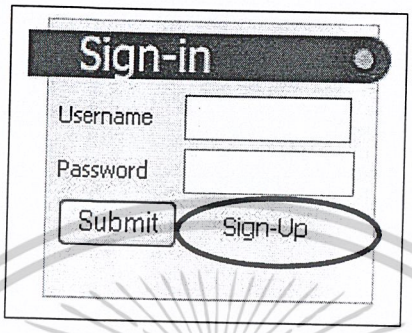


รูปที่ 8-18 หน้าแรกในการติดต่อกับ Home&Decor com

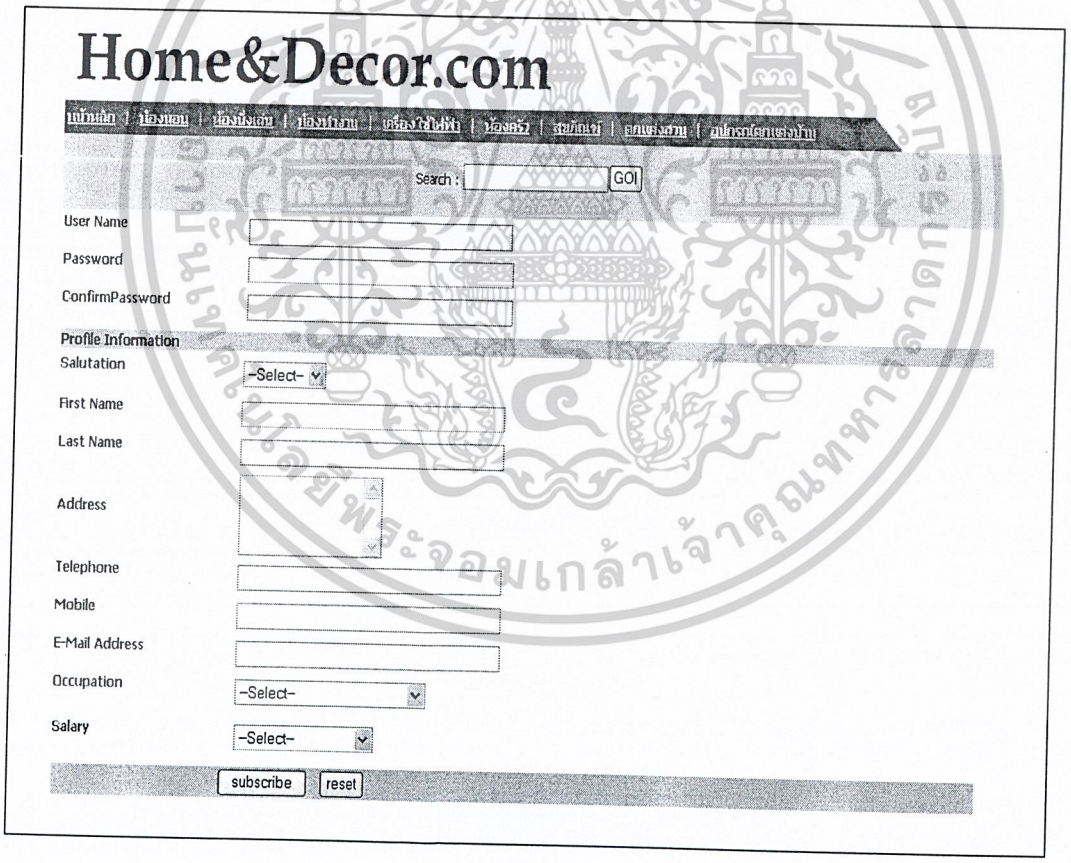
8.9.1 ผู้ใช้งานระบบ

1. การลงทะเบียนกับทางระบบ

กรณีที่ผู้ใช้งานระบบยังไม่เป็นสามชิกกับทางระบบ จะต้องลงทะเบียนเป็นสมาชิกก่อน จึงสามารถทำธุรกรรมต่างๆกับทางระบบได้



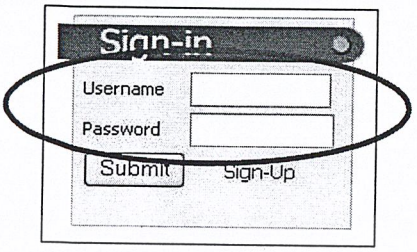
รูปที่ 8-19 การลงทะเบียนกับทางระบบ



รูปที่ 8-20 แบบฟอร์มการลงทะเบียน

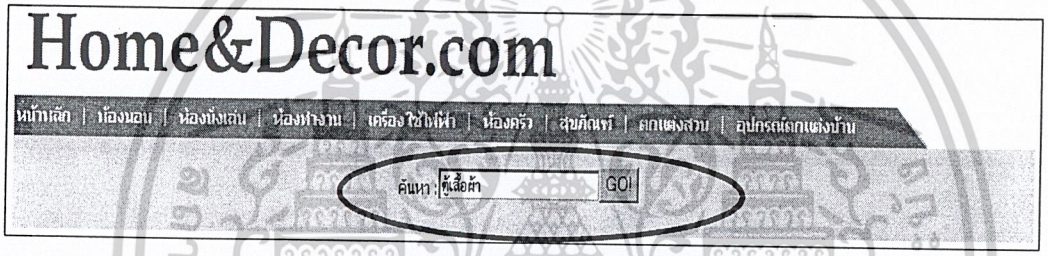
2. การล็อกอินเข้าสู่ระบบ

หากเป็นสมาชิกกับทางระบบแล้ว ผู้ใช้งานระบบจะต้อง ล็อกอินเข้าสู่ระบบก่อน จึงสามารถเข้าทำธุรกรรมกับทางระบบ



รูปที่ 8-21 การล็อกอินเข้าสู่ระบบ

3. การค้นหารายการสินค้าที่ต้องการ



รูปที่ 8-22 ป้อนคำที่ต้องการค้นหา

รูปที่ 8-23 ผลลัพธ์จากการค้นหา

4. การเลือกดูสินค้าตามแผนก

รูปที่ 8-24 การเลือกดูสินค้าหมวดห้องนั่งเล่น

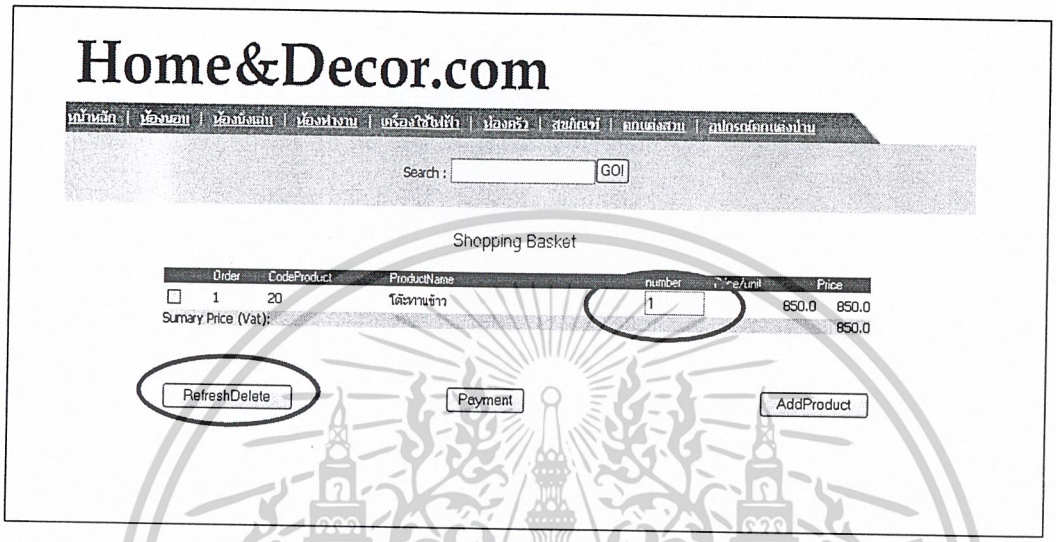
5. การเลือกซื้อสินค้าที่ต้องการ

เมื่อต้องการซื้อสินค้าชิ้นใด ผู้ใช้งานระบบคลิกที่ชื่อของสินค้าชิ้นนั้น แล้วจะต้องคลิกที่ปุ่ม Buy อีกทีหนึ่ง เหมือนเป็นการ ตกลงซื้อ

รูปที่ 8-25 การเลือกซื้อสินค้าชิ้นที่ต้องการ

การเพิ่ม-ลบ จำนวนสินค้าที่เลือกซื้อ

หากผู้ใช้งานระบบต้องการสินค้าชิ้นนี้เพิ่มเติม หรือแก้ไขจำนวนสินค้าที่ต้องการ สามารถแก้ไขได้ที่ช่องที่ระบุจำนวนสินค้าเอง โดยใส่จำนวนที่ต้องการลงไป และกดที่ปุ่ม " Refresh Delete " เพื่ออำนวยความสะดวกแก่ผู้ใช้งาน ที่ไม่ต้องย้อนกลับไปเลือกสินค้าชิ้นเดียวกันซ้ำๆหลายครั้ง



รูปที่ 8-26 ผู้ใช้ต้องการเลือกซื้อสินค้ากับทางระบบ

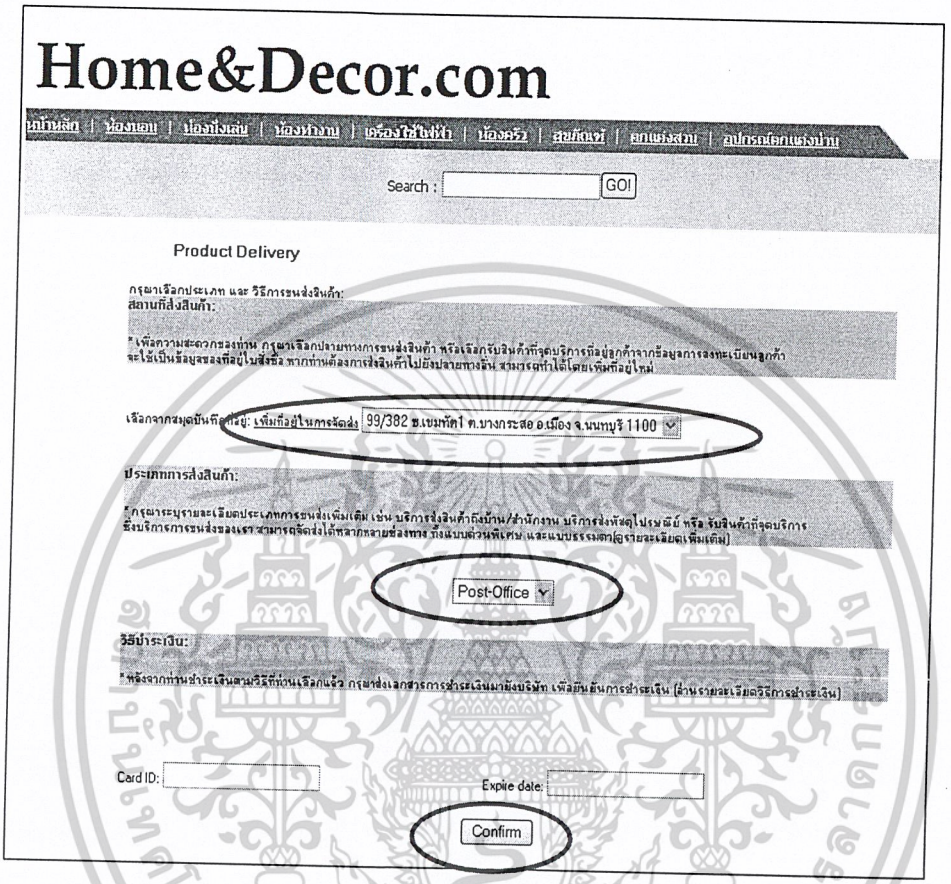
การลบสินค้าที่ไม่ต้องการ

กรณีที่ผู้ใช้งานระบบเลือกสินค้าผิดชิ้น หรือ ไม่ต้องการสินค้าชิ้นนั้นๆแล้ว ก็สามารถลบสินค้าออกจากรายการได้ โดยทำการกดที่ check box หน้าสินค้าชิ้นนั้น และกดที่ปุ่ม " Refresh Delete "

รูปที่ 8-27 การเลือกลบรายการสินค้าที่ไม่ต้องการ

6. การชำระเงิน

ผู้ใช้สามารถเลือกที่อยู่ที่ให้ไปส่งสินค้าได้ โดยอาจจะไม่ใช่ที่อยู่ที่ผู้ใช้งานระบบลงทะเบียนไว้ และสามารถเลือกประเภทของการส่งสินค้าได้ หากถูกต้องตามที่ผู้ใช้งานต้องการแล้วกดปุ่ม “ Confirm ”

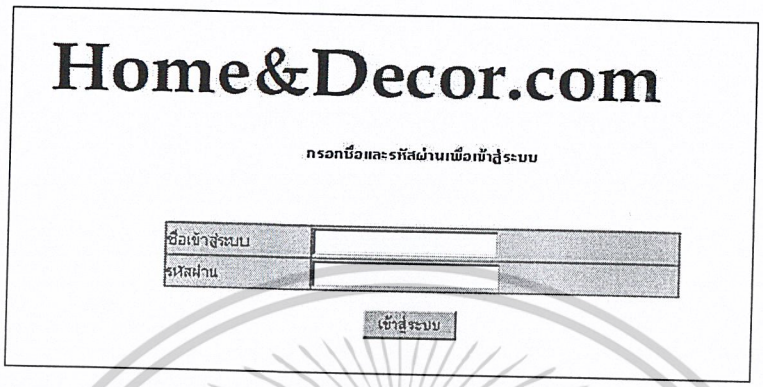


รูปที่ 8-28 แสดงหน้าจอการชำระเงิน

8.9.2 ผู้ดูแลระบบทั่วไป

1. การล็อกอินเข้าสู่ระบบ

เมื่อผู้ดูแลระบบล็อกอินเข้าสู่ระบบแล้ว จะแสดงสิทธ์ในการจัดการภายในระบบของผู้ใช้งานขึ้นมา เพื่อบอกแก่ผู้ใช้งานว่า สิทธิ์ในการจัดการระบบของผู้นั้นมีแค่ไหน

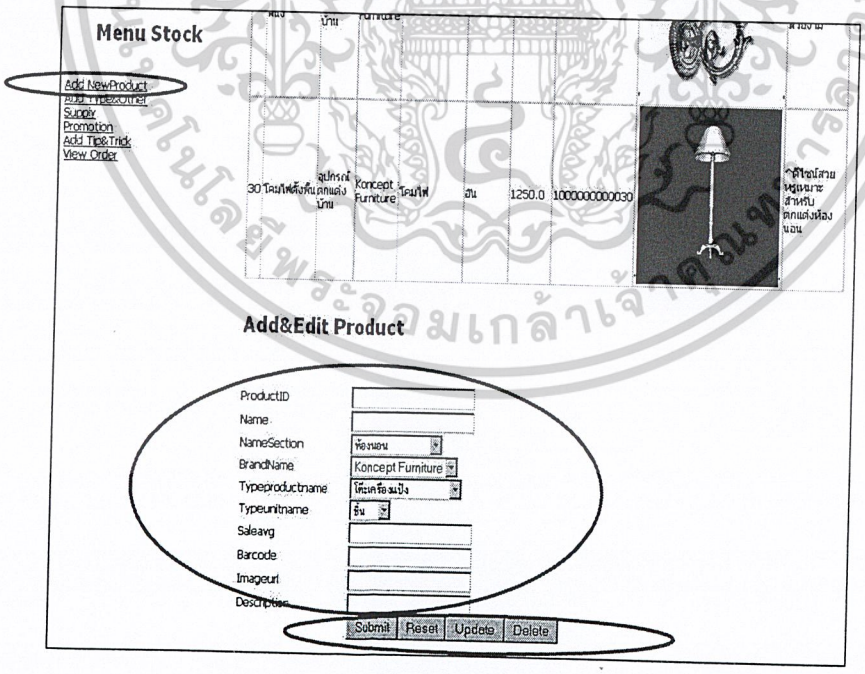


รูปที่ 8-29 หน้าจอการล็อกอินเข้าสู่ระบบของผู้ดูแลระบบ

2. การเพิ่มเติมข้อมูลของสินค้าใหม่ และแก้ไขรายละเอียดเดิมของสินค้า

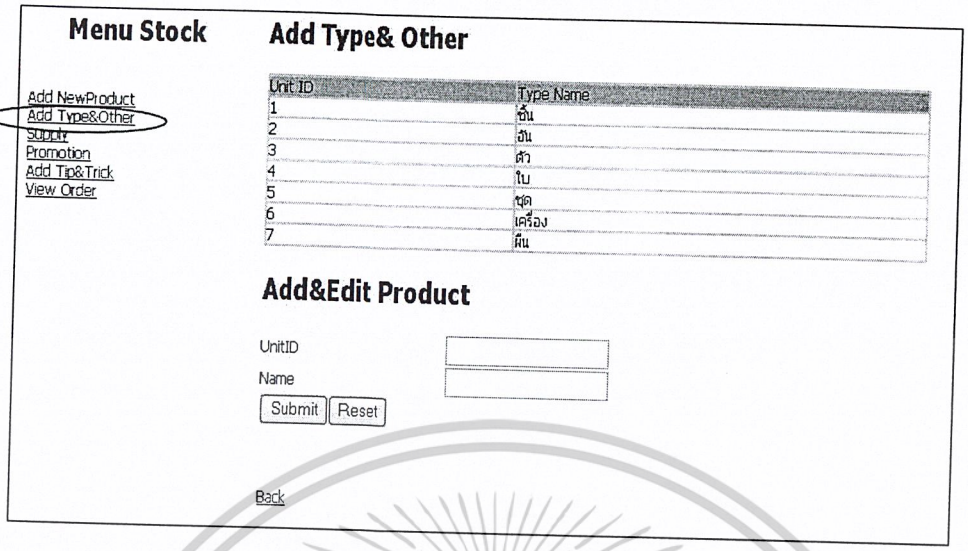
การเพิ่มเติมข้อมูลสินค้าจะอยู่ด้านล่างของหน้า โดยที่ภายในหน้าเพจนี้จะมีการแสดงรายการสินค้าทั้งหมดที่มีในระบบ พร้อมกับแสดงรูปภาพของสินค้านั้นๆด้วย

โดยหน้าเพิ่มเติมสินค้าใหม่ให้กับระบบนี้ สามารถเลือกจาก แถบเมนูทางด้านซ้ายมือ



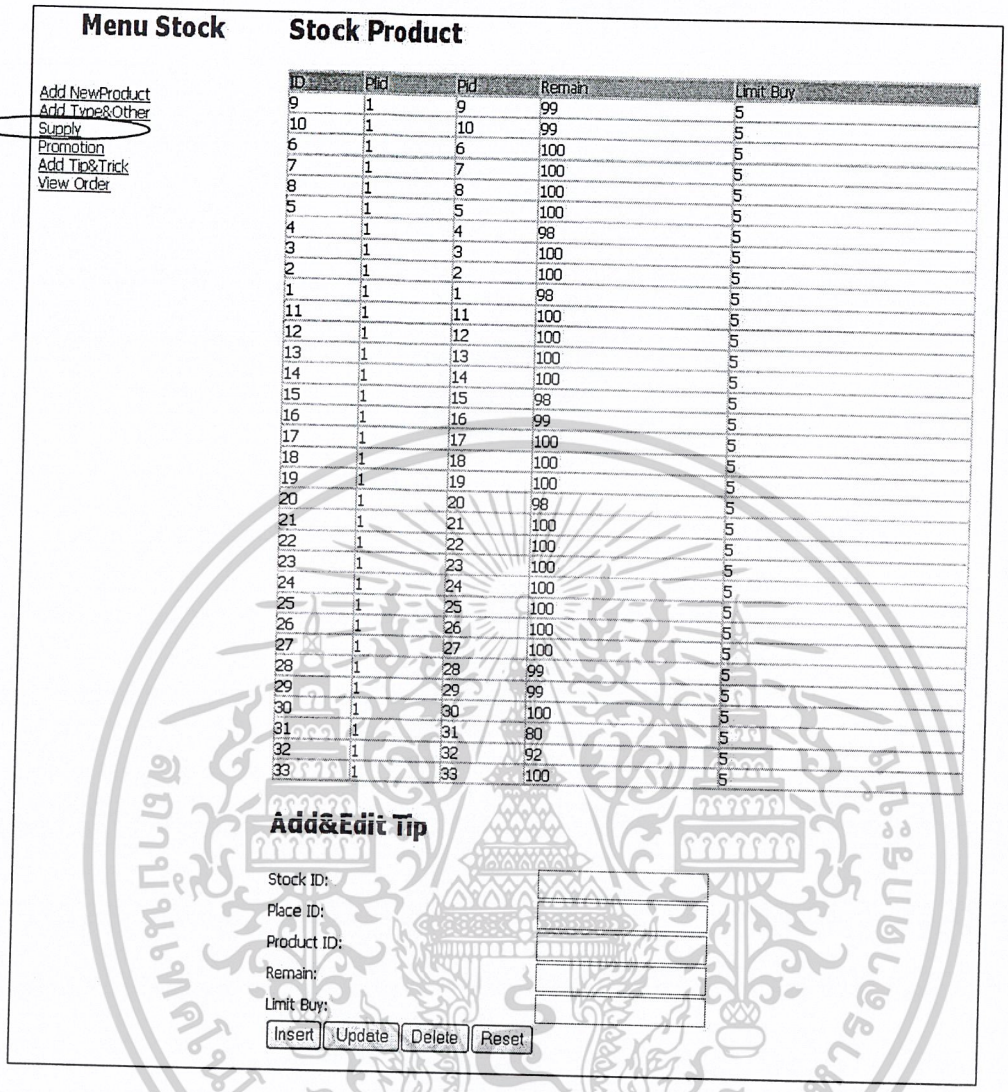
รูปที่ 8-30 การเพิ่มเติมรายการสินค้าใหม่

3. การเพิ่มเติมข้อมูลประเภทของหน่วยของสินค้า



รูปที่ 8-31 ข้อมูลหน่วยสินค้า

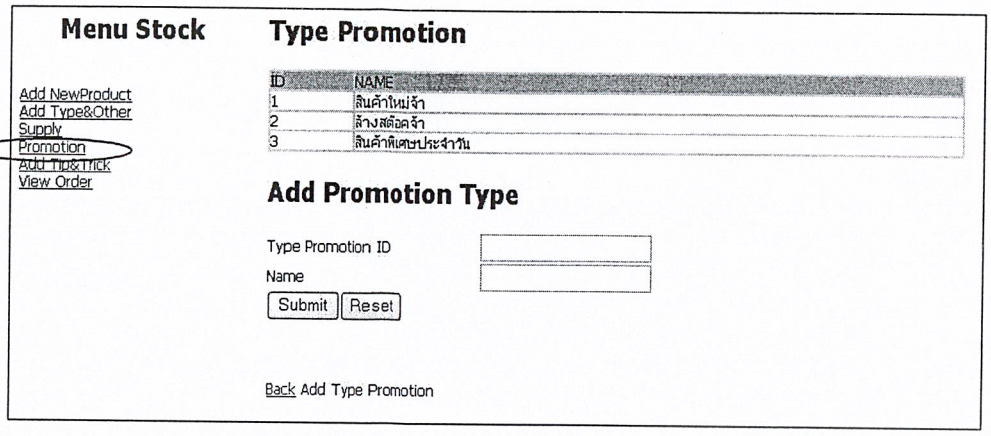
4. การแก้ไขเพิ่มเติม และ จัดการเกี่ยวกับปริมาณสินค้าในคลังสินค้า



รูปที่ 8-32 การจัดการคลังสินค้า

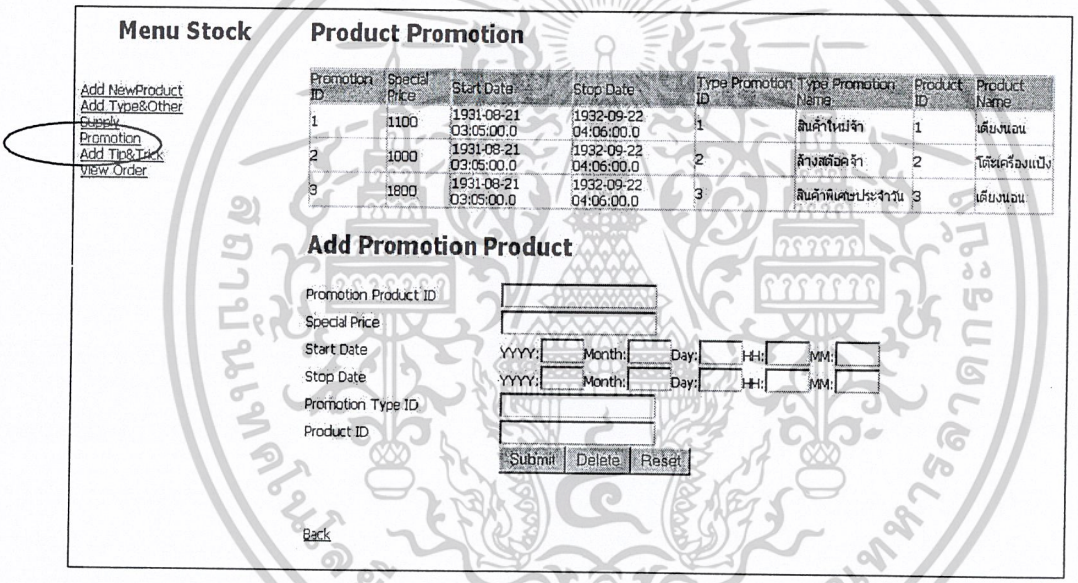
5. การเพิ่มเติมและจัดการเกี่ยวกับโปรโมชั่นแต่ละแบบ

โดยโปรโมชั่นของสินค้าทั้ง 2 รูปแบบนั้น จะต้องอ้างอิงกับประเภทของสินค้า ว่าอยู่ในโปรโมชั่นประเภทใด



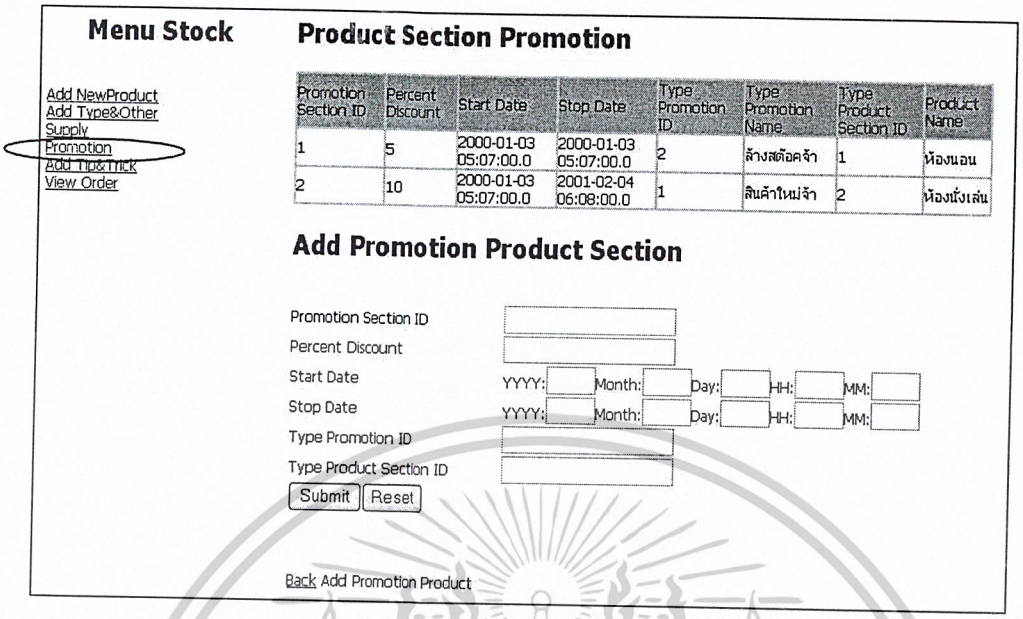
รูปที่ 8-33 ประเภทของโปรโมชั่น

- โปรโมชั่นของสินค้าแต่ละชิ้น



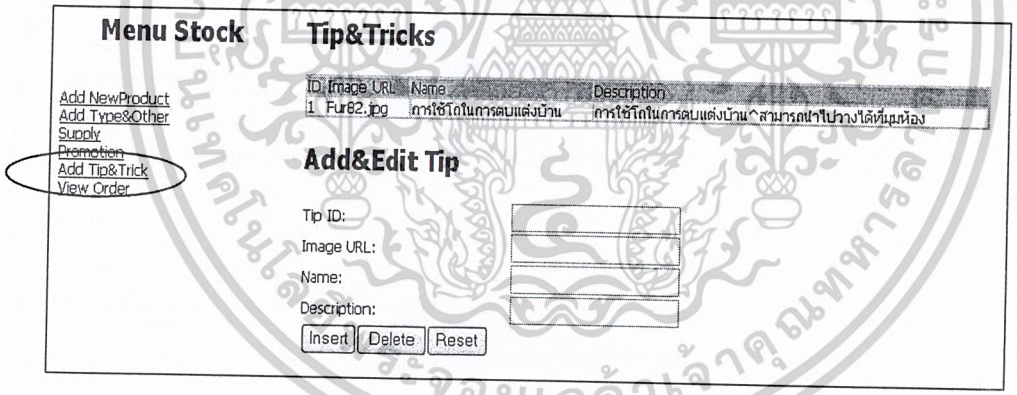
รูปที่ 8-34 โปรโมชั่นสินค้าแต่ละชิ้น

- โปรโมชั่นของสินค้าแต่ละแผนก



รูปที่ 8-35 โปรโมชั่นสินค้าแต่ละแผนก

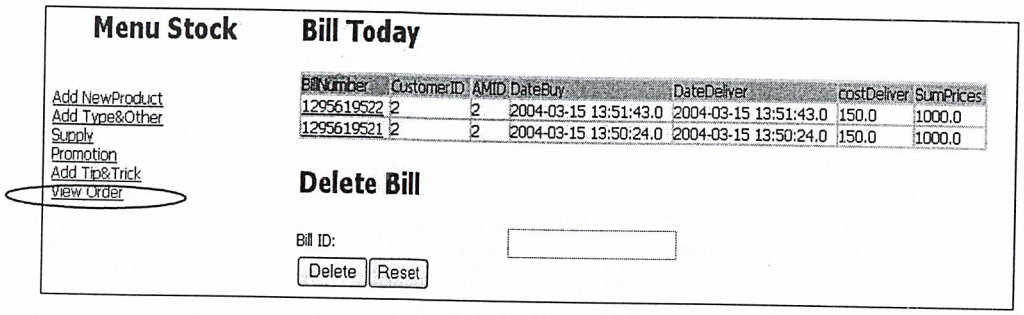
6. การแก้ไขและเพิ่มเติมเกร็ดความรู้ให้กับผู้ใช้งานระบบ



รูปที่ 8-36 การกรอกเกร็ดความรู้ในระบบ

7. การตรวจสอบรายการขายในแต่ละวัน

โดยจะดูว่าในแต่ละวัน มีใบชำระค่าสินค้ากี่ใบและมีรายละเอียดอย่างไรบ้าง โดยเลือกที่เมนู View Order

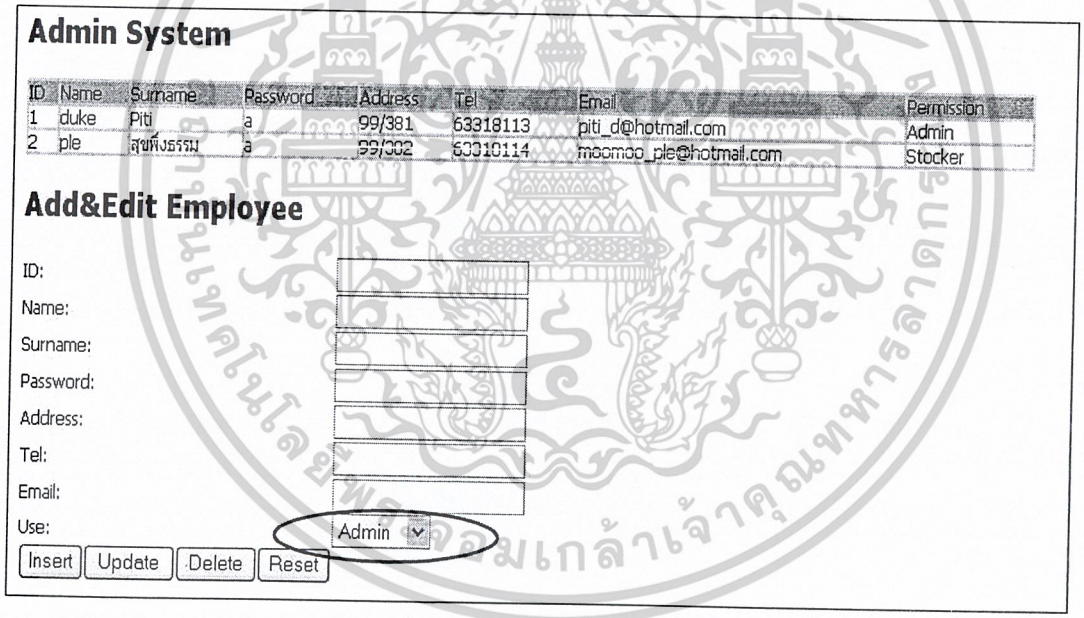


รูปที่ 8-37 การเลือกดูใบรายการสั่งสินค้าในแต่ละวัน

8.9.3 ผู้ดูแลระบบที่เป็น Administrator ของระบบ

1. เพิ่ม และ แก้ไขข้อมูลของผู้ดูแลระบบ

ทั้งในส่วนของผู้ดูแลที่เป็น administrator และ ผู้ดูแลระบบที่เป็นผู้ดูแลระบบทั่วไป หากสิทธิ์การจัดการในระบบเป็น System นั่นก็คือมีสิทธิ์เป็น administrator ของระบบ ทำหน้าที่จัดการดูแล เกี่ยวกับผู้ดูแลระบบอื่นๆ



รูปที่ 8-38 การเพิ่มผู้ดูแลระบบ

บทที่ 9
บทวิจารณ์ และสรุป

9.1 สรุปผลการดำเนินงาน

ในโครงงานนี้ได้ศึกษาการพัฒนาเชิงคอมโพแนนต์ ซึ่งมีหัวข้อในการศึกษาดังนี้

1. ในเรื่องทฤษฎีอ็อบเจ็กต์โอเรียนเต็ด เพื่อการออกแบบ
2. การพัฒนาโปรแกรมด้วย Enterprise Java Beans
3. การพัฒนาโปรแกรมข้ามแพลตฟอร์มด้วย Web services

9.2 แนวทางการพัฒนาต่อ

โครงงานนี้เป็นโครงงานต่อเนื่องในส่วนของการพัฒนาแอพพลิชั่น ซึ่งทำการพัฒนาเพื่อให้สามารถใช้งานได้จริง ดังนั้น จึงแนะนำให้พัฒนาในส่วนเชิงเทคนิคมากขึ้น ซึ่งได้แบ่งหัวข้อย่อยได้ดังต่อไปนี้ เพื่อเป็นการศึกษาในเชิงลึกต่อไป

1. การพัฒนาโปรแกรม EJB ในเรื่องของ Design Pattern ต่างๆ เพื่อเพิ่มประสิทธิภาพของระบบ
2. ในเรื่องการทำเสตทฟูลบนเว็บเซอร์วิส
3. การแลกเปลี่ยนข้อมูลและเรียกใช้เซอร์วิสบริการ ในลักษณะชนิดข้อมูลแบบต่างๆ
4. EJB ในการทำเรื่อง Transaction, Message Driven Beans เพื่อนำไปประยุกต์ใช้

บรรณานุกรม

[1] Ed Roman : "*Mastering Enterprise JavaBeans*" , Wiley Computer Publishing , 1999

[2] Ed Roman , Scott Ambler , Tyler Jewell : "*Mastering Enterprise JavaBeans Second Edition*" , Wiley Computer Publishing , 2002

[3] Bill Burke and Sacha Labourey : "*JBoss 3.2 Workbook for Enterprise JavaBeans*" , www.orelly.com , 2003

[4] Deepak Alur , John Crupi , Dan Malks : "*core J2EE Patterns Best Practices and Design Strategies*" , Prentice Hall PTR , 2001

[5] BEA Systems , Inc : "*Programming WebLogic Enterprise JavaBeans*" , 2001

[6] BEA Systems , Inc : "*BEA WebLogic Server Administration Guide*" , 2001

[7] ดร.วีระศักดิ์ ชิงถาวร และคณะ : "*Enterprise JavaBeans*" , ซีเอ็ด , 2003