ระบบการจองบ้านออนไลน์

ONLINE HOME RESERVE SYSTEM

โดย

นายเมธี ทองทัพ

นายศุภชัย บุญสมเชื้อ

อาจารย์ที่ปรึกษา

ผศ.บรรจง ปิยธำรง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบการจองบ้านออนไลน์

ONLINE HOME RESERVE SYSTEM

ผู้จัดทำ

1. นาย เมธี ทองทัพ รหัสประจำตัว 41013542
2. นาย ศุภชัย บุญสมเชื้อ รหัสประจำตัว 41013552

อาจารย์ที่ปรึกษา

(ผศ. บรรจง ปิยธำรง)

ระบบการจองบ้านออนไลน์

นายเมธี ทองทัพ 41013542
นายศุภชัย บุญสมเชื้อ 41013552
ผศ. บรรจง ปียธำรง อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของโครงงานวิจัยและพัฒนาเพื่อออกแบบโปรแกรมประยุกต์ไคลเอนต์ เซิร์ฟเวอร์ เพื่อสร้างระบบการจัดการฐานข้อมูลโครงการจองบ้านผ่านเว็บไซต์ ให้มีความสะดวก, ถูกต้องและรวดเร็วมากขึ้นในการดูรายละเอียดต่างๆ ของโครงการบ้าน การค้นหาบ้าน และการจองบ้าน รวมทั้งให้ความสะดวกกับผู้ดูแลระบบในการแก้ไขและเพิ่มรายละเอียดต่างๆ ของโครงการ ข้อมูลบ้าน และการแก้ไขภาพแผนผังโครงการ

การพัฒนาจะเริ่มตั้งแต่ การกำหนดความต้องการของระบบ, วิเคราะห์และออกแบบระบบเชิงวัตถุ, ออกแบบและสร้างฐานข้อมูลของระบบโดยใช้ E-R Diagram โดยจะใช้ระบบการจัดการฐานข้อมูลของออราเคิล 8i

ปริญญานิพนธ์ฉบับนี้กล่าวถึงการพัฒนา ไคลเอนต์ เซิร์ฟเวอร์ โดยใช้ภาษาจาวาในการเขียนโปรแกรม และ ติดต่อกับฐานข้อมูล ด้วย Java Servlet ใช้มาตรฐาน JDBC และได้จำลองระบบการจองบ้านมาเป็นระบบตัวอย่างในการพัฒนา นำเสนอโมเดลตามสัญลักษณ์ของ UML และใช้ ออราเคิลแอพพลิเคชั่นเซิร์ฟเวอร์ (OAS) เป็นเว็บเซิร์ฟเวอร์

Online Home Reserve System

Metee Tongthup

Suphachai Boonsomchuer

Asst.Prof. Banjong Piyathamrong Advisor

ABSTRACT

This project is apart of research and develops, for design client server application program that implement DBMS village for reservation project via web site. It's able to be convenience, accuracy and rapidly in obtain information, search and home addition to be convenience with DBA that add details of home and modify project plan.

Initialization of home project development is requirement specification, system analysis and object – oriented design, design and implement database using E – R diagram with Oracle8i.

This project has mentioned client server development using Java in implementation and connection database. It's used Java servlet for JDBC standard. Case study in this project is home reservation system using UML model. And it's used Oracle Application Server (OAS) for web server.

กิตติกรรมประกาศ

โครงงานและวิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี เนื่องจากผู้จัดทำได้รับความร่วมมือและความช่วยเหลือจากบุคคลต่างๆ หลายฝ่าย ทั้งทางด้านวิชาการและกำลังใจ ผู้จัดทำขอขอบพระคุณ

อาจารย์ที่ปรึกษา ผศ. บรรจง ปิยธำรง ที่กรุณาชี้แนะให้คำปรึกษา และดูแลให้โครงงานนี้สำเร็จได้ด้วยดี

คณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกๆท่าน สำหรับความรู้และประสบการณ์ที่มีค่ายิ่ง ถึงแม้อาจารย์จะต้องรอส่งงานถึงตอนดึกหรือเข้ามาในวันหยุด อาจารย์ก็มีเวลาให้พวกเราเสมอ

เพื่อนๆ น้องๆ ห้อง P และ D ผู้ร่วมชะตากรรมทำโครงงานเช่นกัน และได้ให้คำแนะนำปรึกษาต่างๆ อันเป็นประโยชน์ต่อพวกเราเป็นอย่างยิ่ง

และบุคคลสำคัญที่สุดที่ทำให้เรามีวันนี้ คือ คุณพ่อ คุณแม่ และสมาชิกในครอบครัวทุกคน ที่คอยสนับสนุนและเป็นกำลังใจในทุกเรื่อง

นายเมธี ทองทัพ

นายศุภชัย บุญสมเชื้อ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

หน้า

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันความรู้และเทคโนโลยีในด้านคอมพิวเตอร์ ได้มีการพัฒนาอยู่ตลอดเวลา ไม่ว่าจะเป็นฮาร์ดแวร์หรือ ซอฟต์แวร์ ซึ่งถ้าจะกล่าวถึงในด้านซอฟต์แวร์แล้ว การออกแบบโปรแกรมก็ได้มีการพัฒนาไปสู่รูปแบบต่างๆ การออกแบบเชิงวัตถุ (Object-oriented) ก็เป็นการออกแบบโปรแกรมอีกรูปแบบหนึ่งที่ถูกพัฒนาขึ้นมา ที่มีคุณสมบัติดีกว่าการออกแบบโปรแกรมในรูปแบบเก่า ที่เป็นโพรซีเยอร์ (Procedure) อีกทั้งมีการพัฒนาภาษาต่างๆขึ้นมากมายเพื่อใช้ร่วมกับระบบที่ออกแบบมาเป็นออปเจ็กต์-โอเรียนเต็ด

สำหรับการพัฒนาด้านซอฟต์แวร์ในปัจจุบันนั้น ไม่ใช่แต่เพียงการออกแบบเชิงวัตถุเท่านั้นที่มีการพัฒนาขึ้นมา ปัจจุบันบริษัททางด้านซอฟต์แวร์คอมพิวเตอร์ต่างๆได้มีการพัฒนาเครื่องมือที่ใช้ในการพัฒนาภาษาต่างๆขึ้นมามากมาย (Software Development Tool) ซึ่งสำหรับภาษาจาวานั้นก็มีบริษัทต่างๆที่ทำการพัฒนาเครื่องมือที่ใช้ในการพัฒนาภาษาจาวา (Java Development Tool) ขึ้นมาเช่นกัน ไม่ว่าจะเป็น Jbuilder, J++, Visual Café และ VisualAge for Java เป็นต้น ซึ่ง Development Tool เหล่านี้จะช่วยทำให้เราสามารถพัฒนาโปรแกรมได้อย่างรวดเร็ว ไม่ว่าจะเป็นการช่วยในการสร้าง User Interface หรือว่าช่วยในการตรวจสอบ Syntax ทำให้โปรแกรมเมอร์ในปัจจุบันหันมาใช้เครื่องมือในการพัฒนาโปรแกรมเหล่านี้แทนการเขียนโปรแกรมที่อาศัยเอดิเตอร์ (Editor) ในการเขียนและนำมาคอมไพล์ (Compile) ด้วยคอมไพเลอร์ (Compiler) แบบในอดีต

โครงงาน การพัฒนาโปรแกรมประยุกต์ ไคลเอนต์ เซิร์ฟเวอร์ นี้จะเป็นการออกแบบและสร้างระบบการจอง,เช่า ของโครงการบ้านต่างๆ ด้วยระบบคอมพิวเตอร์ออนไลน์ผ่านเว็บ ซึ่งในปัจจุบันนี้อินเตอร์เน็ตได้เข้ามามีบทบาทและส่วนเกี่ยวข้องกับชีวิตประจำวันของเรามากขึ้น จำนวนผู้คนที่ใช้ก็เพิ่มสูงขึ้น ดังนั้น การจอง,ขาย/เช่าซื้อ ของโครงการบ้าน ด้วยระบบคอมพิวเตอร์ออนไลน์ผ่านเว็บ จึงเป็นอีกทางเลือกที่มีส่วนช่วยให้ผู้ที่สนใจกับโครงการบ้าน ในแบบต่างๆสามารถแวะเข้ามาดูรายละเอียดต่างๆ ได้อย่างรวดเร็วและไม่เสียเวลามากนัก

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษาการออกแบบโปรแกรมประยุกต์ด้วยแนวคิดการออกแบบเชิงวัตถุ ซึ่งเป็นแนวคิดใหม่ในการออกแบบระบบ
2. ศึกษาการใช้ระบบจัดการฐานข้อมูล (DBMS : Database Management System) ออราเคิล (Oracle) และการออกแบบฐานข้อมูลขึ้นมาเพื่อใช้งานร่วมกับระบบที่เป็นออบเจ็กต์-โอเรียนเต็ดได้

3. ศึกษาการเขียนโปรแกรมด้วยภาษาจาวา เพื่อใช้ในการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ ไม่ว่าจะเป็นการสร้าง User Interface, การควบคุมระบบ, การติดต่อกับรีเลชันนัลดาต้าเบส (Relational Database)
4. ศึกษาแนวทางการพัฒนาเชิงวัตถุโดยใช้เป็นส่วนในการออกแบบการทำงาน
5. ศึกษาการนำเสนอโมเดลการพัฒนาเชิงวัตถุด้วย Unified Modeling Language (UML)โดยใช้โปรแกรม Rational Rose ซึ่งเป็น CASE tool ช่วยในการสร้างโมเดล

1.3 ขอบเขตของงานวิจัย

โครงงานนี้จะออกแบบและสร้างระบบการจอง .เพื่อ ซื้อ/เช่าของโครงการบ้านต่างๆ ด้วยระบบคอมพิวเตอร์ออนไลน์ผ่านเวปโดยมีฟังก์ชันการทำงานและลักษณะของระบบตามที่กำหนดไว้ในรายละเอียดตามความต้องการของระบบที่ได้จากขั้นตอนการวิเคราะห์ แต่เป็นการจำลองฟังก์ชันการทำงานหลักมาเท่านั้น ยังไม่ครอบคลุมถึงฟังก์ชันการทำงาน,รายละเอียดและข้อกำหนดต่างๆที่เป็นจริงทั้งหมดในทางธุรกิจ ระบบนี้ใช้เพื่อเป็นตัวอย่างในการศึกษาแนวทางการพัฒนาและสร้างระบบเท่านั้นโดยระบบจะมีฐานข้อมูลกลางซึ่งเป็นออราเคิลเซิฟเวอร์ ไคลเอนต์ทุกเครื่องจะลงบนระบบปฏิบัติการ Windows NT 4.0 และจะติดต่อกับเซิร์ฟเวอร์ฐานข้อมูลผ่านทางเว็บไซต์โดยมีขอบเขตดังนี้

1. เพื่อให้สามารถให้บริการแก่ลูกค้าทั่วไป ที่ทำการลงทะเบียนเป็นสมาชิกไว้ให้สามารถดำเนินการในเรื่องของการจองบ้าน , การประกาศขายบ้าน และการประกาศให้เช่าที่พักอาศัย ในโครงการต่างๆ ด้วยระบบคอมพิวเตอร์ออนไลน์ผ่านเว็บไซต์
2. เพื่อให้สามารถสอบถามรายละเอียดของโครงการ ในทุกโครงการ ที่ดำเนินการจอง/เช่า ได้อย่างถูกต้องทันทีตลอดเวลาผ่านเว็บไซต์ โดยจะจัดส่งคำตอบที่ได้จากการสอบถามเจ้าของโครงการให้กับผู้ถามผ่านทางเมลล์
3. เพื่อเพิ่มประสิทธิภาพการให้บริการด้านการขาย
4. โครงงานนี้จะสามารถให้ผู้ที่ลงทะเบียน สามารถทำการจองบ้านในโครงการต่างๆได้เพียงหนึ่งหลังต่อ หนึ่งโครงการเท่านั้น เพื่อป้องกันการจองที่ซ้ำซ้อน
5. สามารถ แก้ไขเพิ่มเติมข้อมูลต่างๆ ที่เกี่ยวกับบ้านจัดสรรได้ เช่น รายละเอียดโครงการ ,รายละเอียดบ้าน และ รูปภาพ โดยผ่านเว็บไซต์เพื่อความสะดวกในการจัดการและจัดเก็บข้อมูลลงฐานข้อมูล

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ อยู่ 6 เรื่องด้วยกัน คือ การค้นหาและกำหนดความต้องการของระบบ, การออกแบบระบบเชิงวัตถุ, การเขียนโปรแกรมเชิงวัตถุด้วยภาษาจาวา , การใช้งานระบบจัดการฐานข้อมูล และ การออกแบบกับการติดต่อกับฐานข้อมูล ซึ่งมีรายละเอียดดังในบทที่ 2, 3, 4, 5 และ 6 จากนั้นก็จะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบระบบ ซึ่งมีรายละเอียดในบทที่ 7

เริ่มจากการศึกษาทฤษฎีการพัฒนาเชิงวัตถุ กำหนดโดเมนปัญหาของระบบ วิเคราะห์ความต้องการและออกแบบระบบ นำเสนอด้วยโมเดล UML จาก Class Diagram ที่ได้นำนำมาเขียนคลาสต่างๆ ของระบบด้วยภาษาจาวา และการออกแบบฐานข้อมูลแบบรีเลชันแนลด้วย E-R Diagram สำหรับใช้ในการสร้างตารางเพื่อเก็บข้อมูลลงสู่ฐานข้อมูล

ต่อไปจะทำการออกแบบหน้าจอ User Interface ที่ใช้ในการติดต่อกับผู้ใช้และทำการเขียนโปรแกรมเพิ่มเติมเพื่อให้ระบบนั้นทำงานได้ สุดท้ายก็จะทำการทดสอบโปรแกรมที่เขียนขึ้นมา พร้อมทั้งปรับปรุงให้สมบูรณ์

1.5 รายละเอียดของระบบ

1.5.1 ระบบงานต่างๆ

- ระบบงานการจัดการระบบทั่วไปได้แก่ การลงทะเบียน, การค้นหาข้อมูลบ้าน, การแสดงรายละเอียดของโครงการ, การประกาศ ขาย/เช่า บ้าน และ การสอบถามรายละเอียดของโครงการ
- ระบบงานการจอง เพื่อซื้อ/เช่า บ้าน
- ระบบการแก้ไขข้อมูลโครงการ และ ข้อมูลบ้าน

1.5.2 รายละเอียดระบบงานการจัดการทั่วไป

มีคุณสมบัติในการทำงาน สรุปได้ดังนี้

- ลงทะเบียนกับลูกค้าใหม่ที่เข้ามาใช้บริการ
- เชื่อมโยงฐานข้อมูล (Database) ที่ใช้งานขณะนั้น กำหนดผู้ใช้ (User) และรหัสผ่าน (Password) ในการใช้งาน
- กำหนดรหัสต่างๆที่จำเป็นต้องใช้ในการลงทะเบียนของลูกค้า, การจองบ้าน
- กำหนดและแก้ไข รายละเอียดโครงการ,ข้อมูลบ้าน และ รูปภาพต่างๆ
- ยกเลิกข้อมูลสำหรับโครงการที่ปิดแล้ว
- การค้นหาข้อมูลบ้านต่างๆที่ต้องการ และ แสดงรายละเอียดของข้อมูลโครงการที่ค้นหาได้
- แสดงรายละเอียดของโครงการต่างๆ ได้
- สามารถประกาศขายสำหรับบ้านมือสอง และ สามารถประกาศให้เช่าที่พักอาศัย สำหรับลูกค้าที่เป็นสมาชิกกับระบบ

1.5.3 รายละเอียดระบบงานการจอง

มีคุณสมบัติในการทำงาน สรุปได้ดังนี้

- สามารถทำการจองในลักษณะตาราง (2 มิติ)
- บันทึกข้อมูลลูกค้า พร้อมรายงานผลการจอง
- บันทึกการยกเลิกการจอง พร้อมรายงาน

- สามารถจองบ้านได้เพียงหนึ่งหลังต่อหนึ่งโครงการและ ถ้าไม่ปฏิบัติตามเงื่อนไขการจอง เช่น เมื่อทำการจองแล้วไม่มาติดต่อกับทางสำนักโครงการภายใน 7 วัน การจองนั้นก็จะถูกยกเลิกการจอง

1.5.4 รายละเอียดระบบการแก้ไขข้อมูลโครงการ และ ข้อมูลบ้าน

มีคุณสมบัติในการทำงาน สรุปได้ดังนี้

- สามารถแก้ไขรายละเอียดโครงการ รายละเอียดบ้าน และ รูปภาพแผนผังโครงการ โดยผู้ดูแลระบบได้
- สามารถกำหนดรายละเอียดข้อความโฆษณา หรือข้อความประกาศการขายบ้านสำหรับบ้านใหม่ และ รูปภาพบ้านที่จะแสดงบนเว็บไซต์ได้

บทที่ 2
พื้นฐานของออบเจ็กต์-โอเรียนเต็ด (Object - Oriented)

2.1 พื้นฐานของออบเจ็กต์-โอเรียนเต็ด (Object-Oriented)

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุ หรือที่เรียกกันว่า "ออบเจ็กต์" นั้นเป็นการมองโครงสร้างของโปรแกรมในรูปแบบใหม่ มองเปรียบเทียบกับสิ่งที่มีอยู่จริงในโลก ซึ่งการมองโครงสร้างของโปรแกรมในรูปแบบเก่านั้นจะมีการใช้ข้อมูลร่วมกัน แต่การมองแบบออบเจ็กต์ นั้นจะมองเสมือนว่าสิ่งต่าง ๆ นั้นเป็น ออบเจ็กต์ มันจะมีอิสระจากกัน ไม่ใช้ข้อมูลร่วมกัน มีการซ่อนของข้อมูลในตัวมันเอง ใช้ข้อมูลในส่วนของมัน มีการมองโครงสร้างเป็นลำดับชั้นลงไป มีการสืบทอดคุณสมบัติของแต่ละออบเจ็กต์ รวมไปถึงการนำสิ่งที่มีอยู่กลับมาใช้ใหม่ สิ่งต่างๆเหล่านี้ คือที่มาของการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ

2.2 เทคโนโลยีออบเจ็กต์-โอเรียนเต็ด (Object-Oriented Technology)

ในการพัฒนาระบบแบบโครงสร้างนั้น จะพยายามที่ให้นักพัฒนาระบบแยกปัญหาและแก้ปัญหาเป็นส่วน ๆ ซึ่งในการพัฒนาระบบแบบออบเจ็กต์โอเรียนเต็ด นั้นก็เช่นกันแต่จะมองปัญหาเป็นลักษณะของออบเจ็กต์ ซึ่งจะมีความใกล้ชิดกันน้อย ทำให้การดูแลและแก้ไขส่วนต่าง ๆ ของระบบ หรือออบเจ็กต์สามารถทำได้ง่าย และ มีผลกระทบต่อส่วนอื่น ๆ ของระบบน้อยที่สุด ด้วยการนำหลักการของ ออบเจ็กต์โอเรียนเต็ดมาใช้ จะทำให้ การออกแบบและพัฒนาซอฟต์แวร์สามารถตรวจสอบความผิดพลาดได้ง่าย , ช่วยเพิ่มประสิทธิภาพในการเอ็กซิคิวต์ (execute) คำสั่ง , ลดชนิดของคำสั่งของฮาร์ดแวร์ , มีการคอมไพล์จากภาษาระดับสูงเป็นภาษาระดับต่ำที่ไม่ซับซ้อนและยังช่วยลดพื้นที่ในการเก็บข้อมูลอีกด้วย

2.3 หลักการการพัฒนาระบบด้วยวิธีเชิงวัตถุ

หลักการของวัตถุ มีการแบ่งปัญหาออกเป็นวัตถุ ซึ่งมีความผูกพันกันน้อย ทำให้การดูแลและแก้ไขต่างๆ ของระบบหรือวัตถุ สามารถทำได้ง่าย และมีผลกระทบต่อส่วนอื่นๆ ของระบบน้อยที่สุด

กลยุทธหลักในการกำหนดวัตถุมีตัวอย่างดังนี้ เช่น

- ขีดเส้นใต้คำนาม จะง่ายและรวดเร็ว แต่มีความกำกวม
- กำหนดสิ่งที่มีอยู่ในโลกจริง
- กำหนดข้อมูลที่เก็บอยู่
- ประยุกต์ใช้ scenario

กำหนดความสัมพันธ์ระหว่างวัตถุ คือ ดูว่าวัตถุไหนสัมพันธ์กัน เช่น

- กำหนด Message เป็นสิ่งที่บ่งบอกถึงความสัมพันธ์ระหว่างวัตถุ
- กำหนดต้นทางของ message
- กำหนดวัตถุที่เก็บ message

- กำหนดวัตถุที่จัดการ message
- ประยุกต์ใช้ scenario

ในขั้นตอนของการวิเคราะห์ระบบนั้น ผู้พัฒนาจะกำหนดให้ระบบเป็นวัตถุใหญ่วัตถุหนึ่ง จากนั้นก็หา Use Case ของระบบใหญ่ที่มีต่อวัตถุภายนอก ซึ่งในขั้นตอนนี้จะได้แผนภาพ Context และ Use Case ออกมา ต่อจากนั้นก็กำหนด Scenario ภายในที่จำเป็นให้กับแต่ละ Use Case จากนั้นก็นำ Message มากำหนดเป็น Protocol ซึ่ง Protocol ก็จะประกอบด้วยหลายๆ Message

2.3.1 Use Case Model

Use case model ประกอบด้วยผู้กระทำ (Actor) และ Use Case ผู้กระทำจะอยู่นอกระบบ แสดงถึงบุคคลหรือสิ่งของที่มีความสัมพันธ์กับระบบ โดยผู้กระทำอาจจะ

- ให้ข้อมูลแก่ระบบเพียงอย่างเดียว
- รับข้อมูลจากระบบเพียงอย่างเดียว
- ทั้งให้และรับข้อมูลกับระบบ

ผู้กระทำอาจจะพบได้ในประโยคปัญหาและการสนทนาระหว่างลูกค้ากับผู้ดูแลระบบ

Use cases คือ บทความสนทนาระหว่างผู้กระทำกับระบบ ซึ่งแสดงถึงหน้าที่การทำงานที่ถูกจัดเตรียมขึ้นโดยระบบ การกำหนด Use Case ควรจะแสดงถึงงานหลักๆที่เกิดขึ้นตั้งแต่ต้นจนจบและต้องมีการส่งผลบางอย่างให้แก่ผู้กระทำ

ความสัมพันธ์ระหว่าง Use Case มีอยู่ 2 แบบ คือ *uses* และ *extends* ความสัมพันธ์แบบ *uses* เกิดจาก Use Case A ถูกเรียกใช้จาก Use Case B และ C นั่นคือ Use Case B และ C *uses* Use Case A

ส่วนความสัมพันธ์แบบ *extends* เกิดจาก Use Case B ถูกสร้างเริ่มต้นมาจาก Use Case A และเพิ่มรายละเอียดบางส่วน เราเรียกว่า Use Case B *extends* Use Case A

2.3.2 Scanario 2 แบบ

- Sequence Diagram ใช้แสดงลำดับขั้นตอนการทำงานในการแสดง message ระหว่างวัตถุพร้อมด้วยลำดับของ message ในระบบ
- Collaboration Diagram ใช้แสดงโครงสร้างของวัตถุที่ทำงานร่วมกันเป็นหลัก

2.3.3 แผนภาพของคลาส

สำคัญมากที่สุดในการวิเคราะห์และออกแบบโดยใช้วิธีเชิงวัตถุ จะเน้นที่การแสดงโครงสร้างของคลาสมากกว่าความสัมพันธ์ระหว่างคลาส

ขั้นตอนการวิเคราะห์เป็นขั้นตอนหนึ่งของการพัฒนาระบบ ผลลัพธ์ที่ได้จากขั้นตอนนี้คือการอธิบายว่าระบบถูกสร้างขึ้นมาเพื่อทำอะไร ไม่ได้เป็นการระบุว่าระบบควรจะทำงานอย่างไร ในเทคนิคเชิงวัตถุ ผลลัพธ์จากการวิเคราะห์นี้คือโมเดล ซึ่งโดยปกติจะเรียกขั้นตอนนี้ว่า Modeling Phase

2.3.4 ขั้นตอนของการวิเคราะห์

ขั้นตอนของการวิเคราะห์มีดังนี้

1. กำหนดประโยคปัญหา (Problem Statement) ของระบบ
2. กำหนดผู้กระทำ (Actor) และ Use Case จากประโยคปัญหา
3. ร่างภาพส่วนติดต่อผู้ใช้ของหน้าที่พื้นฐานที่ผู้ใช้ติดต่อกับระบบ
4. สร้าง Object Model
5. สร้าง Dynamic Model

2.3.4.1 ประโยคปัญหา (Problem Statement)

เป็นบทความที่บรรยายถึงลักษณะปัญหาของระบบ หน้าที่การทำงานต่างๆ ที่ระบบต้องมี ความสามารถในการติดต่อกันระหว่างผู้ใช้กับระบบ ขอบเขตการทำงานของระบบและรายละเอียดหน้าที่การทำงานที่มีในระบบ

2.3.4.2 ส่วนติดต่อผู้ใช้ (User Interface)

เพื่อความเข้าใจความต้องการของผู้ใช้ในการใช้งานโปรแกรม จึงควรมีการออกแบบส่วนการติดต่อระหว่างผู้ใช้กับระบบ โดยอาจจะเป็นการสร้างส่วนติดต่อของโปรแกรมในลักษณะคร่าวๆเพื่อให้ผู้ใช้สามารถเห็นภาพของการใช้งานโปรแกรมได้ดีขึ้น ซึ่งการสร้างส่วนติดต่อผู้ใช้นี้จะช่วยให้ผู้พัฒนาสามารถแสดงลักษณะการติดต่อภายใน Use Case ได้และเป็นการช่วยในการออกแบบโปรแกรมในขั้นตอนการออกแบบ

2.3.4.3 Object Model

Object Model เป็นแบบจำลองที่สำคัญที่ได้จากขั้นตอนการวิเคราะห์ระบบและเป็นจุดเริ่มต้นในส่วนของการออกแบบและการสร้างระบบ ในโครงงานนี้เราจะสร้าง Object Model ตามรูปแบบของ UML

ขั้นตอนการสร้าง Object Model มีดังนี้

1. กำหนดออบเจ็กต์ที่เป็นไปได้ทั้งหมดของระบบจากประโยคปัญหาและ Use Case
2. กำจัดออบเจ็กต์ที่ไม่ตรงกับปัญหาหรือมีความหมายที่ซ้ำกัน
3. จัดรวมออบเจ็กต์ที่มีลักษณะเหมือนกันเข้าเป็นคลาส
4. กำหนดแอททริบิวต์และโอเปอเรชั่นของแต่ละคลาส
5. กำหนดความสัมพันธ์ระหว่างออบเจ็กต์ของคลาส ซึ่งมีลักษณะความสัมพันธ์ได้แก่ Generalization , Association และ Aggregation

2.3.4.4 ออบเจ็กต์ (Object)

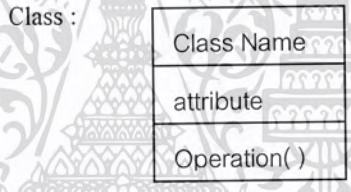
ออบเจ็กต์เป็นกลุ่มก้อนของเอนติตี้ (Entity) ที่มีคุณสมบัติ,พฤติกรรมและสถานะ ออบเจ็กต์แสดงถึงบางอย่างที่มีตัวตน เช่น รถยนต์,เครื่องคอมพิวเตอร์ หรือแสดงถึงสิ่งที่มีตัวตนในความคิด เช่น บัญชีธนาคาร, เครื่องหมายการค้า, พิธีแต่งงานหรือรายการสินค้า

สิ่งที่เราสนใจในตัวออบเจ็กต์ประกอบไปด้วย

- แอททริบิวต์ (Attributes) เป็นข้อมูลที่แสดงคุณลักษณะของวัตถุ เช่น รถยนต์ จะมีแอททริบิวต์คือ ความกว้าง, ความยาว,สี, น้ำหนัก
- โอเปอเรชั่น (Operations) คือการกระทำที่วัตถุนั้นสามารถกระทำได้ เช่น รถยนต์ จะมี operations คือ ออกตัว, เบรก, เลี้ยวขวา, เลี้ยวซ้าย
- สถานะ (State) สถานะของวัตถุคือหนึ่งในเงื่อนไขที่เป็นไปได้และอาจเกิดขึ้นได้จริง เช่น รถยนต์ จะมี state คือ กำลังวิ่ง, กำลังชะลอ
- เอกลักษณ์ (Identity) เป็นสิ่งที่ระบุว่าวัตถุนั้นมีอยู่เพียงหนึ่งเดียว เช่น รถยนต์มีเอกลักษณ์คือทะเบียนรถ
- หน้าที่ (Responsibility) คือ หน้าที่และบทบาทของวัตถุต่อระบบ

2.3.4.5 คลาส (Class)

คลาส คือการอธิบายถึงกลุ่มของออบเจ็กต์ซึ่งจะมีคุณสมบัติทั่วไป(attributes),พฤติกรรมทั่วไป (operations) และมีความสัมพันธ์ทั่วไปกับออบเจ็กต์อื่นๆในลักษณะเดียวกัน ดังนั้นคลาสจึงเป็นรูปแบบที่ใช้สร้างออบเจ็กต์



รูปที่ 2-1 แสดงสัญลักษณ์แสดงคลาส

2.3.4.6 สเตอริโอไทป์ (Stereoptype)

คือ การสร้างองค์ประกอบชนิดใหม่ของแบบจำลองหรือการสร้างคลาสใหม่ ซึ่งองค์ประกอบชนิดใหม่จะสืบทอดคุณสมบัติขององค์ประกอบเดิมมาทุกอย่าง(รวมทั้งความสัมพันธ์ต่างๆที่ใช้กับองค์ประกอบนั้นๆ ได้) โดยมีจุดประสงค์การใช้งานที่ต่างออกไป

รูปแบบของสเตอริโอไทป์ใน UML จะใช้เครื่องหมาย << และ >> คร่อมชื่อสเตริโอไทป์ เช่น << Entity >> แนวคิดสเตอริโอไทป์ของโครงงานนี้จะแบ่งประเภทของคลาส เป็น เอนติตี้คลาส (Entity Class) , คลาสขอบเขต (Boundary Class) และ คลาสควบคุม (Control Class)

- เอนติตี้คลาส สิ่งที่มีอยู่จริง หรือการกระทำภายในระบบ
- คลาสขอบเขต กล่าวถึงการติดต่อระหว่างสิ่งที่อยู่รอบๆระบบกับภายในระบบ (การติดต่อไปยังผู้กระทำ)
- คลาสควบคุม ส่งผลให้เกิดพฤติกรรมเฉพาะใน Use Case หรือเรียกว่าเป็นการ " running " Use Case

2.3.4.7 แพ็คเกจ (Packages)

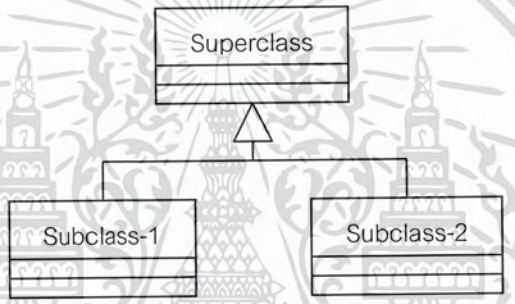
เป็นการรวมคลาสที่มีความสัมพันธ์ใกล้ชิดกันเข้าไว้ในแพ็คเกจ เพื่อประโยชน์สำหรับระบบที่มีคลาสเป็นจำนวนมาก ในระบบที่มีความซับซ้อนเราอาจจะสร้างแพ็คเกจขึ้นมาก่อน แล้วจึงบรรจุคลาสและความสัมพันธ์เข้าไว้ในแพ็คเกจ

2.4 ความสัมพันธ์

2.4.1 Generalization

เป็นความสัมพันธ์แบบการให้กำเนิดคลาสย่อย (Subclass) จากคลาสแม่ (Superclass) โดยคลาสย่อยจะสืบทอดแอททริบิวต์,โอเปอเรชั่นและความสัมพันธ์ทั้งหมดจากคลาสแม่ ทั้งของตัวเองและจากคลาสแม่ของคลาสแม่ชั้นต่อไปเรื่อยๆ คลาสย่อยอาจจะมีการเพิ่มแอททริบิวต์และโอเปอเรชั่นในคลาสของมันเอง ความสัมพันธ์แบบนี้อาจถูกเรียกได้เป็น is-a หรือ kind-of

Generalization :

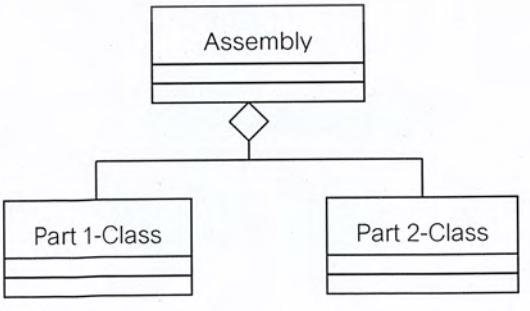


รูปที่ 2-2 แสดงความสัมพันธ์แบบ Generalization

2.4.2 Aggregation

เป็นความสัมพันธ์แบบเป็นส่วนประกอบ หรือที่เรียกว่า part-of คำว่าเป็นส่วนประกอบบางครั้งอาจจะมีความหมายที่ต่างกัน เช่น เครื่องยนต์และล้อเป็นส่วนประกอบของรถยนต์ในขณะที่เรากำลังขับรถอยู่ แต่ถ้ารถยนต์คันนั้นอยู่ในระหว่างการซ่อมที่อู่ซ่อมรถ นั่นคือเครื่องยนต์และล้อก็ไม่ได้เป็นส่วนประกอบของรถยนต์แล้ว

Aggregation :

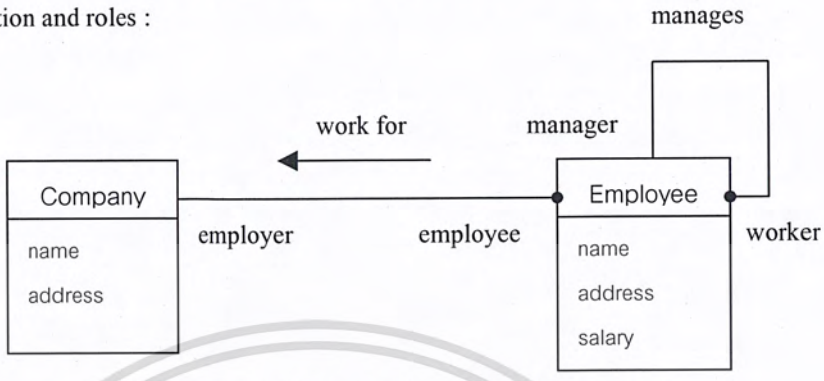


รูปที่ 2-3 แสดงความสัมพันธ์แบบ Aggregation

2.4.3 Association

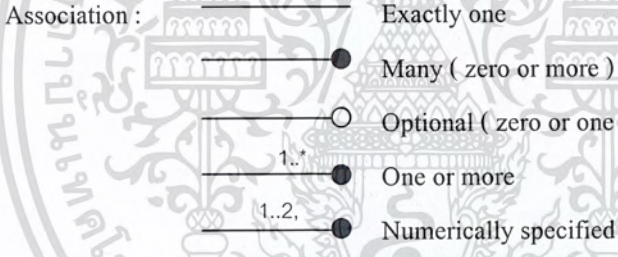
เป็นความสัมพันธ์แบบสองทิศทาง ซึ่งใช้ในการติดต่อกันระหว่างคลาส ความสัมพันธ์แบบ association ระหว่างคลาสหมายถึงมีการเชื่อมต่อระหว่างออบเจ็กต์ในคลาสทั้งสอง ที่ปลายของเส้นความสัมพันธ์จะมีการระบุชื่อบทบาทเพื่อบอกว่าคลาสนั้นถูกมองว่าเป็นอย่างไรจากคลาสอื่น

Association and roles :



รูปที่ 2-4 แสดง Association และบทบาท

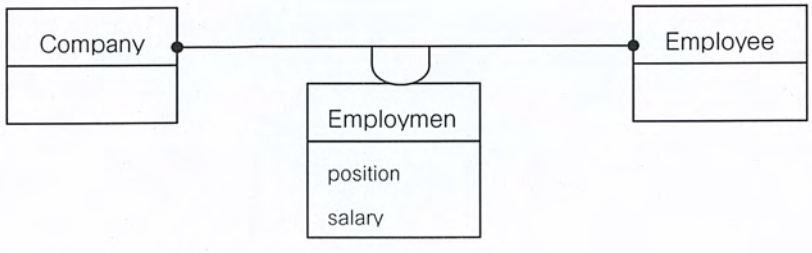
Multiplicity ของ Association เป็นการบอกถึงจำนวนของออบเจ็กต์ที่มีการติดต่อกับคลาสอื่น



รูปที่ 2-5 แสดง Multiplicity

Association Class ในบางครั้งเราอาจจะออกแบบ Association เป็นเหมือนกับคลาสได้ ถ้าหากมีแอททริบิวต์ที่ไม่ได้เป็นของคลาสใดๆที่มีความสัมพันธ์กัน แอททริบิวต์นั้นจะเป็นของ Association Class

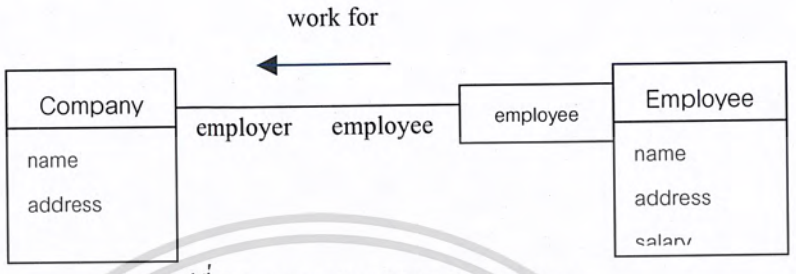
Modeling an association as a class :



รูปที่ 2-6 แสดง Association Class

Qualified Association คือ Association ที่มีแอททริบิวต์เรียกว่า Qualifier โดยค่าของแอททริบิวต์นั้นสามารถที่จะแบ่งแยกออบเจ็กต์ต่างๆของคลาสอีกด้านหนึ่งของความสัมพันธ์ได้ สัญลักษณ์ที่ใช้ก็คือรูปสี่เหลี่ยมขนาดเล็กที่ติดอยู่ระหว่างปลายของเส้น Association กับคลาสที่เชื่อมอยู่และจะมีแอททริบิวต์ที่มีลักษณะดังที่กล่าวมาอยู่ภายใน

Qualified Association :

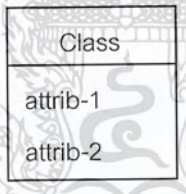


รูปที่ 2-7 แสดง Qualified Association

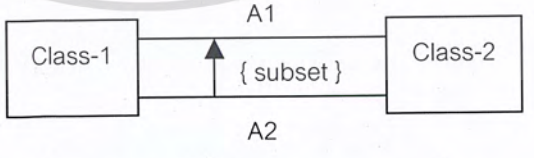
2.4.4 ข้อบังคับ (Constraints)

เป็นการกำหนดข้อบังคับของความสัมพันธ์ระหว่างเอนติตี้ โดยเอนติตี้อาจจะหมายถึงออบเจ็กต์, คลาส, บทบาท,แอททริบิวต์,การเชื่อมต่อ (link) และความสัมพันธ์ ข้อบังคับนี้จะเป็นการจำกัดค่าที่เอนติตี้สามารถเป็นไปได้

รูปแบบทั่วไปของข้อบังคับจะเป็นข้อความที่อยู่ในวงเล็บปีกกา เช่น



รูปที่ 2-8 แสดงข้อบังคับในออบเจ็กต์



รูปที่ 2-9 แสดงข้อบังคับระหว่าง Association

2.5 Dynamic Model

เป็นแบบจำลองที่อธิบายถึงพฤติกรรมของระบบและลำดับของการทำงาน ซึ่งแบ่งเป็น

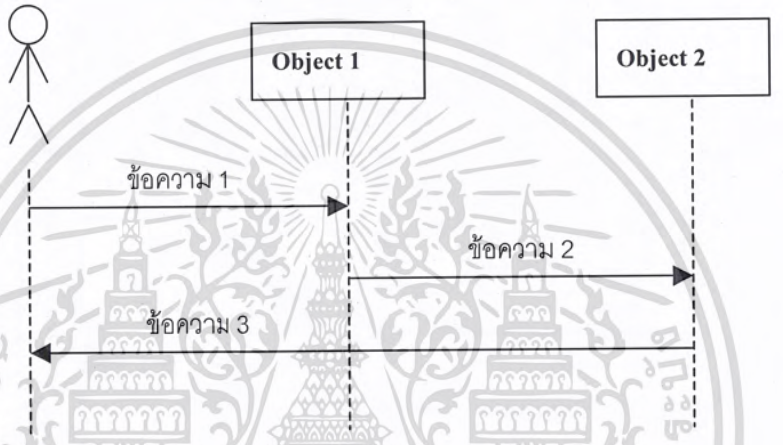
1. Sequence Diagram
2. Collaboration Diagram

3. State Diagram

2.5.1 Sequence Diagram

Sequence Diagram เป็นการแสดงข้อความและลำดับของการส่งข้อความระหว่างออบเจ็กต์ ข้อความ คือ หน่วยที่ออบเจ็กต์ใช้ในการสื่อสารกัน ตัวอย่างของข้อความที่เกิดขึ้น เช่น เมื่อผู้ใช้ร้องขอบริการ , ออบเจ็กต์ร้องขอบริการจากออบเจ็กต์อื่น หรือออบเจ็กต์ส่งคืนข้อมูลจากบริการที่ถูกร้องขอ

Sequence Diagram จะถูกพัฒนามาจาก Use Case โดยแสดงถึงลำดับของข้อความที่เกิดขึ้นของแต่ละ Use Case ตั้งแต่ต้นจนจบ Sequence Diagram มีจุดประสงค์เพื่อช่วยให้ผู้ใช้และผู้ออกแบบระบบสามารถเข้าใจการทำงานของระบบได้ง่ายขึ้น

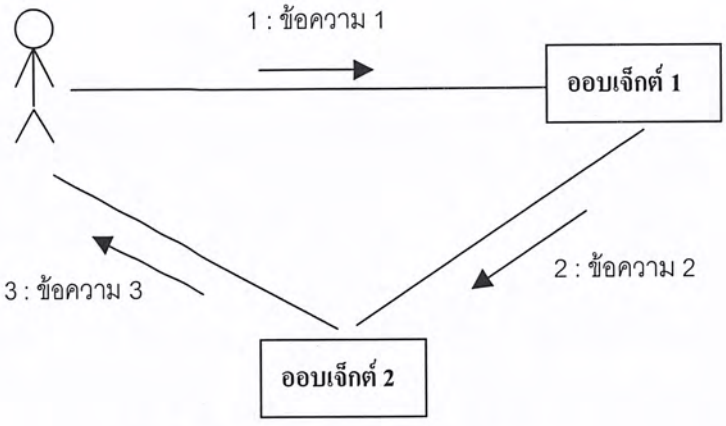


รูปที่ 2-10 แสดง Sequence Diagram

จากรูปไดอะแกรมข้างต้น เส้นในแนวตั้งจะหมายถึงออบเจ็กต์โดยมีชื่อของออบเจ็กต์เขียนอยู่ด้านบนหรือเป็นผู้กระทำที่ติดต่อกับระบบ ส่วนเส้นในแนวนอนนั้นจะแสดงถึงข้อความซึ่งมีชื่อของข้อความนั้นอยู่ ข้อความที่เกิดขึ้นจะเป็นลำดับตามแนวตั้งจากบนลงล่าง

2.5.2 Collaboration Diagram

Collaboration Diagram จะแสดงข้อมูลพื้นฐานเช่นเดียวกับ Sequence Diagram แต่ความแตกต่างอยู่ที่ Sequence Diagram จะแสดงลำดับของข้อความเป็นหลัก ส่วน Collaboration Diagram จะแสดงข้อความและโครงสร้างของออบเจ็กต์ที่ทำงานร่วมกันเป็นหลัก



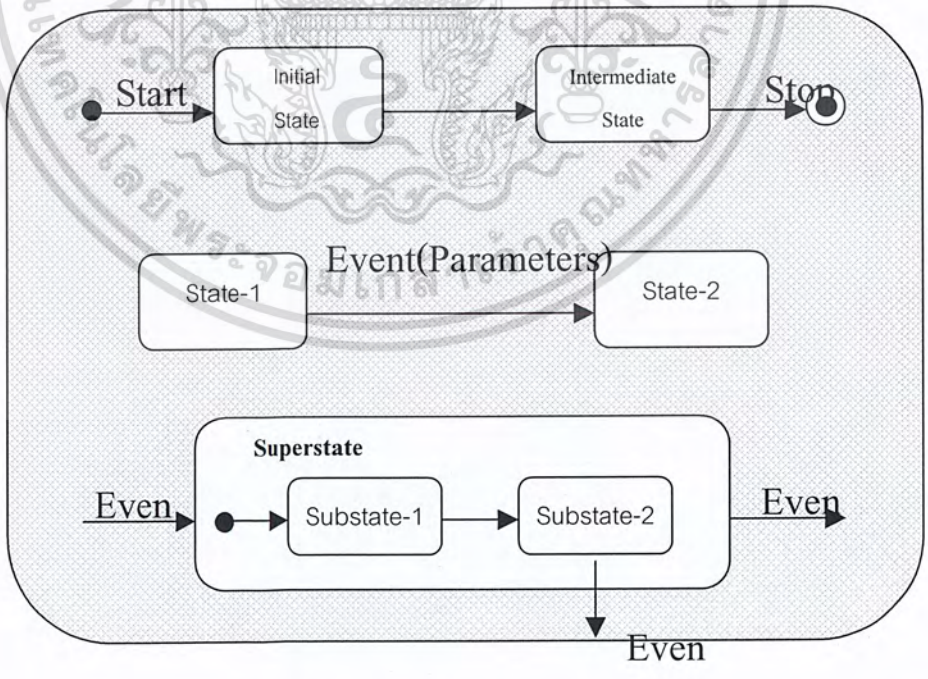
รูปที่ 2-11 แสดง Collaboration Diagram

จากรูปไดอะแกรมข้างต้น จะแสดงการไหลของข้อความระหว่างออบเจ็กต์ด้วยกันหรือระหว่างออบเจ็กต์กับผู้กระทำ โดยข้อความที่ใช้ในการติดต่อกันจะประกอบด้วยหมายเลขลำดับ,ชื่อข้อความและทิศทางของข้อความ

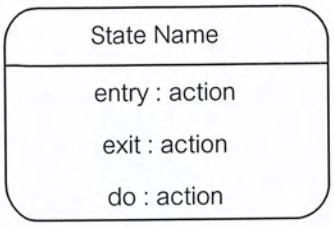
2.5.3 State Diagram

สถานะของออบเจ็กต์ปกติจะถูกกำหนดโดยค่าของแอททริบิวต์หรือค่าตัวแปรภายใน เช่น ออบเจ็กต์หลอดไฟจะมีสถานะคือเปิดและปิด ซึ่งสถานะเปิดและปิดนี้จะเป็นค่าของแอททริบิวต์สถานะของออบเจ็กต์หลอดไฟ

State Diagram แสดงถึงสถานะต่างๆของออบเจ็กต์และการส่งผ่าน (transition) ที่ทำให้เกิดการเปลี่ยนแปลงของสถานะเหล่านั้น และแสดงถึงกิจกรรมที่ออบเจ็กต์กระทำเมื่ออยู่ในสถานะหนึ่งๆ



รูปที่ 2-12 แสดง State Diagram



รูปที่ 2-13 แสดงรูปแบบของสถานะ

State Diagram สามารถมีสถานะซ้อนภายในสถานะอีกทีได้ ตัวสถานะที่อยู่ภายนอกนั้นเราจะเรียกว่า Superstate ส่วนสถานะที่อยู่ภายในเราจะเรียกว่า Substate ในแต่ละสถานะจะมีการระบุถึงการกระทำที่ออบเจ็กต์จะกระทำเมื่ออยู่ในสถานะนั้น โดยสามารถแบ่งการกระทำได้เป็น

- entry action จะทำงานเมื่อมีการเข้าสู่สถานะนั้น
- exit action จะทำงานเมื่อมีการออกจากสถานะนั้น
- do action จะทำงานเมื่อสถานะนั้นๆ แอกทีฟ

รูปแบบของการส่งผ่านคือ Event-name(Parameters)[Guard] / Action list ^Event list โดยมีรายละเอียดดังนี้

Field	คำอธิบาย
Event-name	เป็นชื่อของ event ที่ทำให้เกิดการเปลี่ยนสถานะ
Parameters	เป็นรายการข้อมูลพารามิเตอร์ ที่ต้องการส่งไปพร้อมกับ event
Guard	เป็นเงื่อนไขในรูปของประโยคบูลีน ซึ่งจะต้องมีค่าเป็นจริงจึงจะทำให้เกิดการเปลี่ยนสถานะได้
Action list	เป็นรายการของโอเปอเรชั่นที่จะกระทำเมื่อเกิดการเปลี่ยนสถานะโดยโอเปอเรชั่นเหล่านี้อาจจะเป็นของออบเจ็กต์เองหรือของออบเจ็กต์อื่นก็ได้
Event list	เป็นรายการของ events ที่ถูกสร้างขึ้นเมื่อมีการเปลี่ยนสถานะ โดย events เหล่านี้สามารถกระจายไปยัง state machines อื่นๆ เพื่อเป็นการทำงานแบบพร้อมกันได้

ตารางที่ 2-1 แสดงตารางรูปแบบการส่งผ่าน

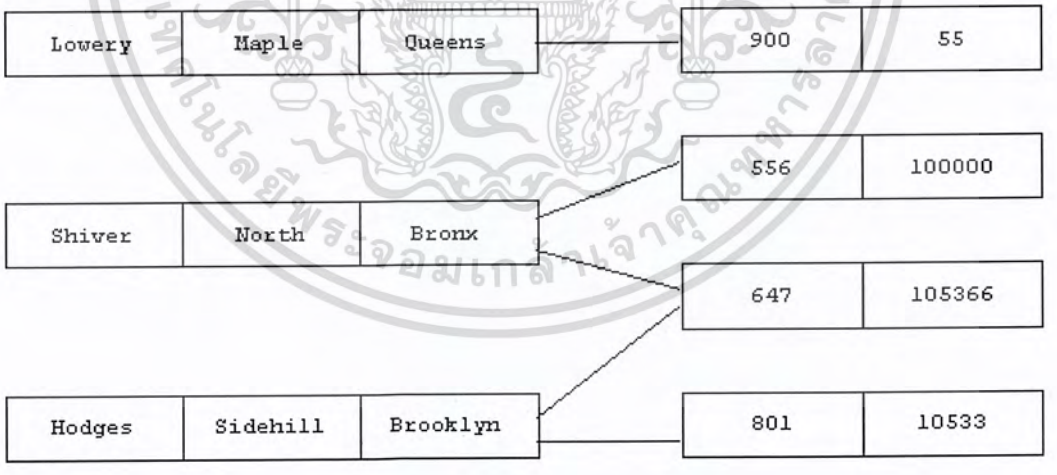
บทที่ 3
ความรู้และทฤษฎีเกี่ยวกับฐานข้อมูล

บทนี้จะอธิบายถึงหลักการและทฤษฎีของระบบฐานข้อมูลโดยอธิบายเป็นนัยวิเคราะห์ และออกแบบระบบฐานข้อมูล ซึ่งมีประโยชน์อย่างมาก สำหรับผู้ที่อยากออกแบบฐานข้อมูล สิ่งนี้จำเป็นอย่างยิ่งในรูปแบบฐานข้อมูล ที่เราเรียกว่าฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

3.1 ระบบฐานข้อมูล (Database System)

ฐานข้อมูล คือ แหล่งเก็บข้อมูลและความสัมพันธ์ระหว่างข้อมูลนั้น และได้รับการออกแบบและควบคุมพิเศษ ให้มีความซ้ำซ้อนของข้อมูลน้อยที่สุด และมีความถูกต้องของข้อมูลโครงสร้างของความสัมพันธ์ ระหว่างข้อมูลในฐานข้อมูลมีอยู่ด้วยกัน 3 รูปแบบดังนี้

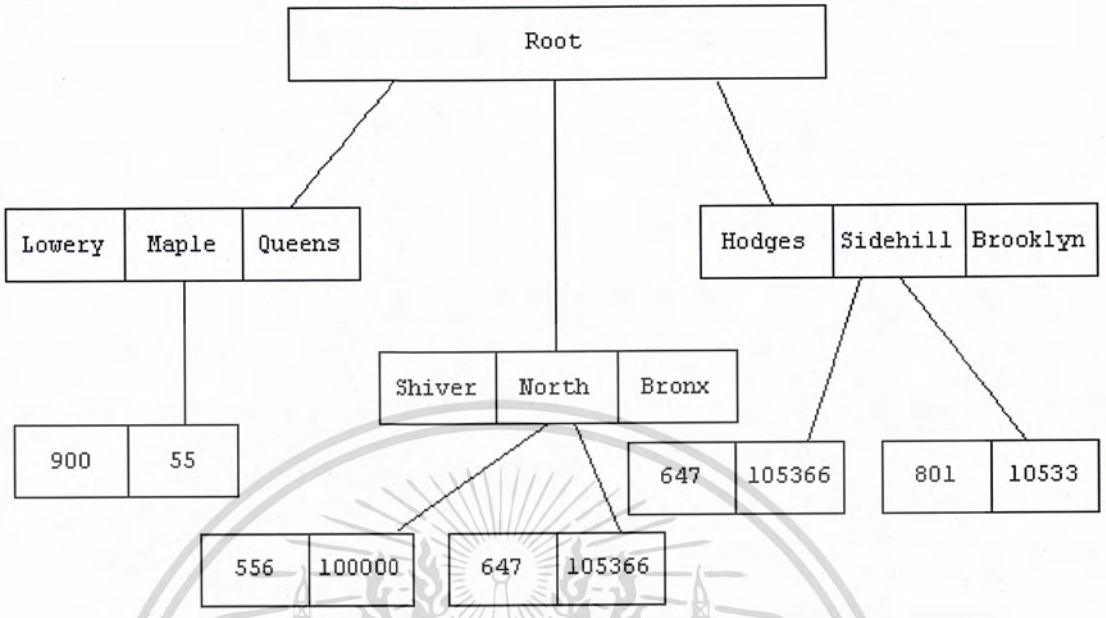
1. แบบโครงข่าย (Network Database Model) การจัดข้อมูลแบบโครงข่ายนี้จะแสดงด้วยกลุ่มของเรคอร์ด (Record) ที่มีส่วนเชื่อมต่อ (Link) หรือ ตัวชี้ (Pointer) แสดงความสัมพันธ์ของข้อมูล โดยที่โครงสร้างของข้อมูลเป็นแบบ มัลติลิสต์ (Multi-list) ซึ่งแสดงความสัมพันธ์ของเรคอร์ด ที่อยู่ในฐานข้อมูล จะมีสภาพเป็นกลุ่มของความสัมพันธ์ที่ไม่มีข้อกำหนดที่แน่นอน (Collection of arbitrary graph) ความสัมพันธ์เป็นแบบกลุ่มกับกลุ่ม (many to many) หรือ หนึ่งกับกลุ่ม (one to many) แต่มีความซับซ้อนใช้งานยาก ดังแสดงความสัมพันธ์ดังรูป



รูปที่ 3-1 แสดงตัวอย่างข้อมูลที่ความสัมพันธ์ข้อมูลเป็นแบบโครงข่าย

2. แบบแผนภูมิต้นไม้ (Hierachical Database Model) แบบแผนภูมิต้นไม้นั้นจะมีลักษณะใกล้เคียงกับแบบโครงข่าย แต่จะแตกต่างกัน ที่โครงสร้างความสัมพันธ์ของเรคคอร์ด ที่อยู่ในฐานข้อมูล ซึ่งจะมีความสัมพันธ์เป็นแบบหนึ่งกับกลุ่ม (one ot many) ซึ่งจะมีลักษณะคล้ายต้นไม้กลับหัว

และการค้นหาข้อมูลที่ต้องการจะต้องเริ่มจากตัวแม่ (Root) และใส่ความสัมพันธ์ ลงมาตามตัวลูก (Child) แสดงความสัมพันธ์ดังรูป



รูปที่ 3-2 แสดงตัวอย่างข้อมูลที่ความสัมพันธ์ระหว่างข้อมูลเป็นแผนภูมิต้นไม้

3. แบบเชิงสัมพันธ์ (Relational Model) รูปแบบนี้จะแสดงรายละเอียดของข้อมูลและความสัมพันธ์ระหว่างข้อมูล อยู่ในรูปกลุ่มของตารางซึ่งในแต่ละตาราง จะประกอบด้วยคอลัมน์ (Column) ต่างๆ โดยชื่อของคอลัมน์เหล่านั้น จะต้องมีชื่อไม่ซ้ำกันและสามารถแสดงความสัมพันธ์ระหว่างข้อมูล อยู่ในรูปของตารางได้ โดยไม่มีตัวชี้หรือลิงก์ลิสต์มาเกี่ยวข้อง ในการแสดงความสัมพันธ์นี้ แต่สามารถมีตัวชี้ (idex) มาเกี่ยวข้องได้เพื่อประโยชน์ในการเพิ่มความเร็ว ในการจัดการข้อมูลเท่านั้นซึ่งจะไม่เกี่ยวข้องกับความสมพันธ์ระหว่างข้อมูล สามารถแสดงความสัมพันธ์ดังรูป

name	street	city	number
Lower	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

number	balance
900	55
556	100000
647	105366
801	10533

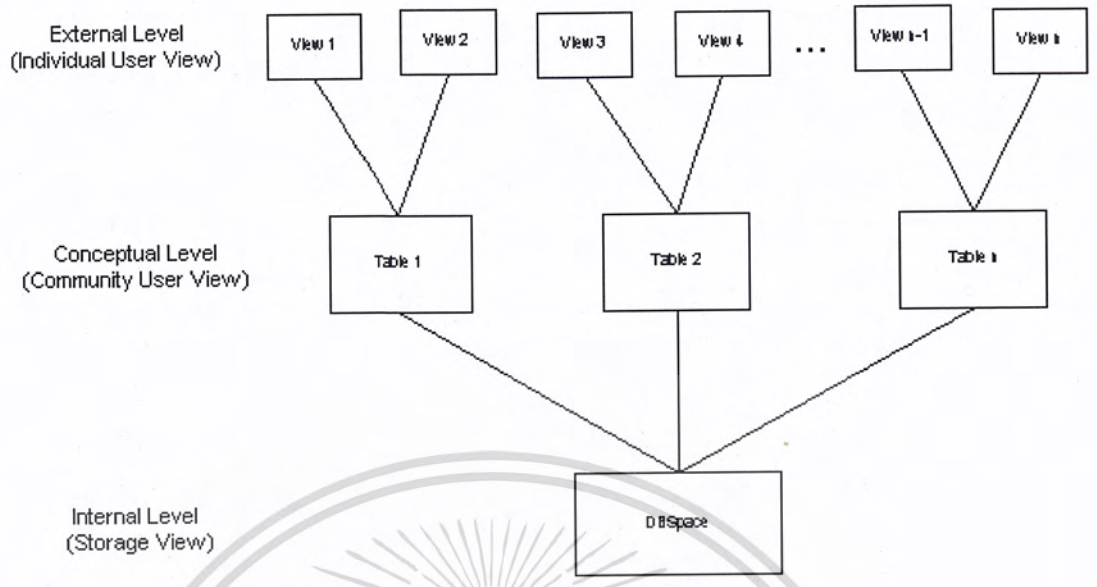
รูปที่ 3-3 แสดงตัวอย่างข้อมูลที่มีความสัมพันธ์ระหว่างข้อมูลเป็นแบบเชิงสัมพันธ์

3.2 สถาปัตยกรรมของระบบฐานข้อมูล (C.J. Date 1986:29)

แบ่งได้เป็น 3 ระดับดังนี้

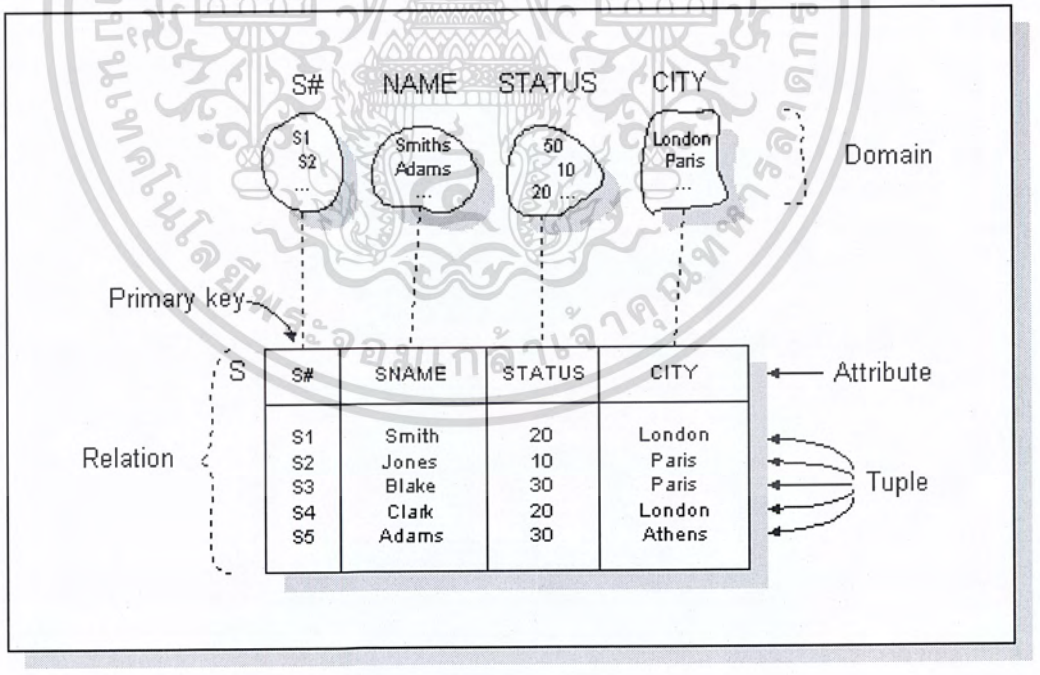
1. นิยามข้อมูลระดับภายนอก (External Schema) จะเป็นการกำหนดโครงสร้างข้อมูล ที่ให้ผู้ใช้เห็น ซึ่งอาจเป็นบางส่วนของนิยามข้อมูล ระดับแนวคิด เช่น ในระบบฐานข้อมูลเชิงสัมพันธ์นั้น ผู้ใช้บางคนอาจต้องใช้ข้อมูลผ่านวิว (View) ซึ่งจะมีสิทธิใช้ข้อมูลบางแถว (ROW) หรือบางคอลัมน์ (Column) ของตารางเท่านั้น ดังนั้นผู้ใช้เหล่านั้น จะมองเป็นเฉพาะข้อมูลที่ผู้ดูแลและควบคุมฐานข้อมูล (DBA/Database Administration) หรือผู้ที่มีอำนาจกำหนดสิทธิของตารางนั้นๆ กำหนดขอบเขตการใช้ข้อมูลในตารางต่างๆ ให้เท่านั้น
2. นิยามข้อมูลระดับแนวคิด (Conceptual Database Schema) จะเป็นการกำหนดลักษณะรูปแบบข้อมูล ขนาดของข้อมูลและความสัมพันธ์ของข้อมูลทั้งหมดในระบบงาน นั่นคือไม่ว่าฐานข้อมูลจะมีความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบใดๆ ก็ตาม จะต้องกำหนดการแทนรูปแบบข้อมูลในนิยามข้อมูลระดับแนวคิดนี้ เช่น ถ้าความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบเชิงสัมพันธ์ (Relation Model) ในระดับจะแสดงชื่อตาราง ชื่อคอลัมน์ ชนิดข้อมูลในแต่ละคอลัมน์ตลอดจนชื่อของคีย์หลัก (Primary Key) และชื่อคีย์นอก (Foreign Key)
3. นิยามข้อมูลระดับภายใน (Internal DatabaseSchema) จะเป็นการกำหนดลักษณะโครงสร้างข้อมูลที่ถูกจัดเก็บจริงในอุปกรณ์เก็บข้อมูล เช่น ถ้าความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบเชิงสัมพันธ์ ซึ่งในระดับแนวคิดและระดับภายนอกจะแสดงอยู่ในรูปแบบตาราง แต่เมื่อข้อมูลของตารางนั้นถูกจัดเก็บจริงในฮาร์ดดิสก์ (Harddisk) ข้อมูลอาจจะถูกจัดเก็บด้วยรูปแบบของบีทรี (B-tree) หรือลิงค์ลิสต์ (Link List) จะจัดเก็บข้อมูล ในระดับนี้ระบบการจัดการฐานข้อมูล (DBMS:Database Management System) จะจัดการให้โดยที่ผู้ใช้ไม่ต้องจัดการเอง

สถาปัตยกรรมทั้ง 3 ระดับนี้ จะแสดงได้ดังรูป



รูปที่ 3-4 แสดงสถาปัตยกรรม 3 ระดับ

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System)



รูปที่ 3-5 แสดงโครงสร้างข้อมูลของฐานข้อมูลเชิงสัมพันธ์

3.3 ระบบจัดการฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลแบบเชิงสัมพันธ์จะแสดงการเก็บข้อมูลในรูปของตารางดังแสดงไว้ในภาพ ซึ่งเป็นรูปแบบที่ง่ายต่อการทำความเข้าใจ และสามารถตอบสนองต่อคำถามที่ซับซ้อนได้ดีกว่าโมเดลฐานข้อมูลชนิดอื่น องค์ประกอบสำคัญของระบบจัดการฐานข้อมูลเชิงสัมพันธ์แบ่งออกเป็น 3 ส่วนคือ

1. โครงสร้างของมูล (Data Structure)

โครงสร้างข้อมูลในระดับ Logical Structure ที่ใช้ต้องเป็นแบบ Relational เท่านั้น นั่นหมายความว่า จะไม่มี Pointer หรือ Index มาใช้ร่วมด้วย ทั้งนี้สามารถใช้ตาราง (Table) มาแสดงความสัมพันธ์ระหว่างข้อมูลได้องค์ประกอบต่างๆ ของโครงสร้างข้อมูลมีดังนี้

- แอททริบิวต์ (Attributes) หมายถึงชื่อคอลัมน์ ของตารางความสัมพันธ์ ที่ใช้บ่งชี้ถึงคุณสมบัติของสิ่งต่างๆ ที่มีความสัมพันธ์กันประกอบขึ้นมาเป็นตารางฐานข้อมูล เช่น แอททริบิวต์ S# หมายถึง รหัสของพนักงาน, แอททริบิวต์ SNAME หมายถึงชื่อของพนักงานและแอททริบิวต์ CITY หมายถึงชื่อของเมือง โดยที่แอททริบิวต์ดังกล่าวมีความสัมพันธ์กันดังนี้คือ พนักงานที่มีรหัส (#S) QC1203 มีชื่อ (SNAME) ว่า 'John' และอาศียอยู่ในเมือง (CITY) 'New Yoek'
- โดเมน (Domains) หมายถึงขอบเขตของข้อมูลในแต่ละแอททริบิวต์ เช่น ข้อมูลที่เก็บลงแอททริบิวต์ CITY จะต้องเป็นชื่อของเมืองที่อยู่ในทวีปยุโรปเท่านั้น
- ทัปเปิล (Tuples) หมายถึงแถวของข้อมูลภายในตารางข้อมูล
- ไพรมารี่คีย์ (Primary Key) หมายถึง แอททริบิวต์หรือกลุ่มของแอททริบิวต์ที่ทำหน้าที่เป็นตัวชี้เอกลักษณ์ของข้อมูลในแต่ละทัปเปิลของตารางฐานข้อมูล
- รีเลชั่น (Relation) หมายถึงผลคูณคาร์ทีเซียนของโดเมนที่สนใจในแอพพลิเคชั่นประกอบขึ้นด้วยองค์ประกอบ 2 ส่วนคือ เฮดดิ้ง (Heading) และบอดี้ (Body)

เฮดดิ้ง (Heading) ประกอบขึ้นมาจากเซ็ตที่มีขนาดคงที่ของแอททริบิวต์ A1,A2,…,An โดย Ai คือ แอททริบิวต์ที่แทนข้อมูลในโดเมน Di(I = 1,2,…,n)

บอดี้ (Body) ประกอบขึ้นจากเซ็ตของทัปเปิลที่มีขนาดแปรผันตามเวลาของรีเลชั่น โดยแต่ละทัปเปิลประกอบด้วยค่าข้อมูลของแอททริบิวต์ต่างๆที่อ้างอิงกับโดเมนดังจะแสดงต่อไปนี้

(S# : 'S1')

(SNAME : 'Smith')

(STATUS : 20)

(CITY : 'London')

1. ความถูกต้องของข้อมูล (Integrity Constraints)

การบังคับความถูกต้องของข้อมูลบนระบบฐานข้อมูลเชิงสัมพันธ์ อย่างน้อยต้องสามารถบังคับกฎข้อบังคับความถูกต้องของข้อมูลต่อไปนี้ได้คือ

- กฎข้อบังคับความถูกต้องของข้อมูลชนิดเอนติตี้ (Entity Integrity) กล่าวว่าแอตทริบิวที่เป็นส่วนของ Primary Key จะต้องไม่เป็น null
- กฎข้อบังคับความถูกต้องของข้อมูลในการอ้างอิง (Referential Integrity) กล่าวว่าแอตทริบิวที่เป็น Foreign Key ต้องมีค่าตรงกับ Primary Key หรือไม่ก็มีค่าเป็น Null Value

2. การจัดการข้อมูล (Data Manipulation)

ข้อมูลภายในฐานข้อมูลเชิงสัมพันธ์ จะต้องถูกจัดการโดยภาษาจัดการข้อมูลของระบบจัดการฐานข้อมูลเท่านั้นโดยภาษาดังกล่าวจะต้องมีคุณสมบัติตามนิยามของ "ความสัมพันธ์ที่สมบูรณ์" (Relational complete) กล่าวคือต้องสนับสนุนกลุ่มปฏิบัติการพื้นฐาน ที่นิยามไว้ภาษาคณิตสัมพันธ์ (Relational Algebra) หรือ (Relational Calculus) ซึ่งประกอบด้วยปฏิบัติการ 8 อย่างที่แบ่งออกได้เป็น 2 กลุ่มดังนี้

- กลุ่มปฏิบัติการพื้นฐาน (Trasitional Set Operators) ได้แก่ ผลคูณคาร์ทีเชียน, ยูเนียน, อินเทอร์เซ็คชั่น และดิฟเฟอร์เรนซ์
- กลุ่มปฏิบัติการพิเศษ (Special Relation Operators) ได้แก่ ซีเล็คท์, โปรเจ็คท์, จอยน์ และดิไวด์

รีเลชั่นนอลโมเดล (Relationsl model)

รีเลชั่นนอลโมเดลเป็นโมเดลที่ใช้ในการอธิบายความสัมพันธ์ ของข้อมูลที่ถูกเก็บด้วยระบบจัดการฐานข้อมูลแบบรีเลชั่นนอล (Relational Database Management System : RDBMS) ในปัจจุบันนี้มีการใช้งานแพร่หลายไปมาก มีการนำไปใช้งานกับเครื่องระดับตั้งแต่เมนเฟรมลงไปจนถึงเครื่องระดับไมโคร และเป็นที่ยอมรับกันว่าผู้ใช้ระบบฐานข้อมูลส่วนใหญ่ (โดยเฉพาะผู้ที่ทำงานด้วยเครื่องระดับมินิ และระดับไมโคร) จะมีความคุ้นเคยกับรีเลชั่นนอลโมเดลมากกว่าอีก 2 โมเดล คือ ไฮเออร์ราร์ชิเคิลโมเดล (Hierarchical model) และเน็ทเวอร์คโมเดล (Network model)

3.4 Entity-Relationship Model

ในการออกแบบฐานข้อมูลขึ้นใช้งานในระบบงานสารสนเทศใดๆ จะต้องอาศัย Data Model เพื่อนำเสนอรายละเอียดต่างๆ ที่เกี่ยวข้องกับข้อมูลในฐานข้อมูลที่ออกแบบ เนื่องจาก Data Model เป็นแบบจำลองที่มีรูปแบบในการนำเสนอรายละเอียดต่างๆ ที่เกี่ยวข้องกับฐานข้อมูล ที่เป็นมาตรฐาน จึงทำให้สามารถนำเสนอต่อผู้ใช้ในต่ละระดับที่มีมุมมองที่แตกต่างกันได้เป็นอย่างดี สำหรับ Data Model ที่นิยมใช้ได้ก่ Entity-Relationship Model หรือที่นิยมเรียกกันสั้นๆ ว่า E-R Model เนื่องจาก E-R Model เป็นแบบจำลองที่มีรูปภาพที่ใช้แทนโครงสร้างทางด้าน Abstraction ต่างๆ ได้เป็นอย่างดี

Entity

เป็นรูปภาพที่ใช้แทน Class ของสิ่งต่างๆ ที่สามารถระบุได้ในความเป็นจริง ซึ่งอาจเป็นสิ่งที่จับต้องได้เช่น พนักงานของบริษัท นักศึกษาที่ลงทะเบียนเรียน หนังสือของห้องสมุด ฯลฯ เป็นต้น หรือ

อาจเป็นเพียงสิ่งที่อยู่ในรูปนามธรรม ที่ไม่สามารถจับต้องได้ เช่น วันหยุดทำการของธนาคาร จำนวนวันลาพักร้อนของพนักงาน ฯลฯ เป็นต้น ซึ่งใน E-R Model จะบ่งออกเป็น 2 ประเภทดังนี้

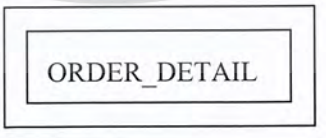
Regular Entity

หรือบางครั้งเรียกว่า Strong Entity ได้แก่ Entity ที่ประกอบด้วยสมาชิกที่มี คุณสมบัติ ซึ่งบ่งบอกถึงเอกลักษณ์ของแต่ละสมาชิกนั้น เช่น Entity "POPULAR" ซึ่งสมาชิกภายใน Entity นี้ ได้กา ประชากรแต่ละคนในประเทศไทยที่มีหมายเลขบัตรประชาชนไม่ซ้ำกันเลย เป็นต้น สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ ได้แก่ รูปภาพสี่เหลี่ยมผืนผ้า โดยมีชื่อของ Entity นั้นอยู่ภายใน ดังรูป

รูปที่ 3-6 แสดง Regular Entity

Weak Entity

เป็น Entity ที่มีลักษณะตรงข้ามกับ Regular Entity กล่าวคือ สมาชิกของ Entity ประเภทนี้ จะสามารถมีคุณสมบัติที่บ่งบอกถึงเอกลักษณ์ของแต่ละสมาชิกได้นั้น จะต้องอาศัยคุณสมบัติหนึ่งของ Regular Entity มาประกอบกับคุณสมบัติของตัวมันเอง เช่น Entity "ORSER_DETAIL" ซึ่งสมาชิกของ Entity นี้ได้แก่รายละเอียดของสินค้าที่สั่งซื้อภายใต้ใบสั่งซื้อแต่ละใบ ซึ่งเมื่อพิจารณาดูจะพบว่า สินค้า ก อาจถูกสั่งซื้อมนใบสั่งซื้อได้หลายใบ ดังนั้น ถ้าระบุเพียงต้องการทราบจำนวนของสินค้า ก ก็จะไม่สามารถทราบได้ว่าต้องการทราบจำนวนสินค้าของสินค้า ก ในใบสั่งซื้อใด แต่ถ้ามีการระบุเลขที่ใบสั่งซื้อประกอบกับสินค้า ก แล้ว ก็จะสามารถทราบได้ทันทีว่าหมายถึงจำนวนของสินค้า ก ในใบสั่งซื้อใด ซึ่งเลขที่ใบสั่งซื้อนี้ ได้แก่คุณสมบัติของ Regular Entity ที่นำมาประกิบกับคุณสมบัติของ Weak Entity "ORDER_DETAIL" เพื่อทำให้สมาชิกของ Entity นี้สามารถมีคุณสมบัติที่บ่งบอกถึงเอกลักษณ์ของแต่ละสมาชิกได้ สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ได้แก่รูปสี่เหลี่ยมผืนผ้า 2 รูปซ้อนกัน โดยมีชื่อของ Entity นั้นอยู่ภายใน ดังรูป



รูปที่ 3-7 แสดง Weak Entity

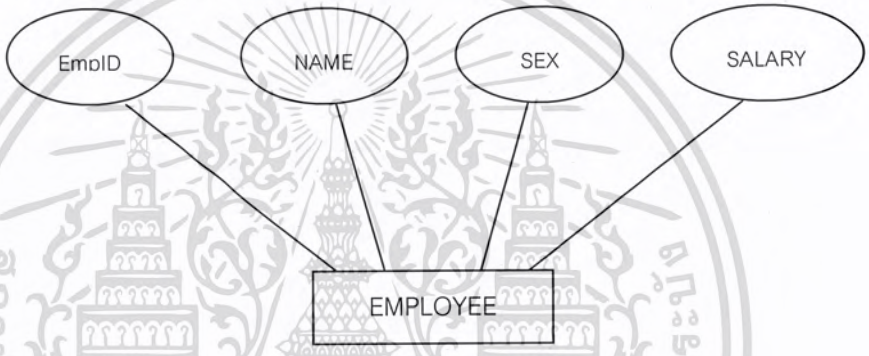
Attribute

อาจใช้คำว่า Property แทน เนื่องจากเป็นการกล่าวถึง E-R Model ในการออกแบบฐานข้อมูลในระดับ Conceptual แต่เนื่องจากจะนำเอา E-R Model ไปใช้ในการออกแบบฐานข้อมูลทั้งในระดับ Conceptual และ Logical จึงจะใช้คำว่า "Attribute" แทน

Attribute ได้แก่ Class ของคุณสมบัติต่างๆ ที่นำมารวมกันแบบ Aggregation Abstraction เพื่อเป็น Entity หรือ Relationship เช่น หมายเลขบัตรประชาชน ชื่อสกุล วันเดือนปีเกิด ภูมิลำเนา วันที่ออกบัตร วันที่บัตรหมดอายุ ที่รวมกันเป็น Entity "บัตรประชาชน" เป็นต้น สำหรับ Attribute ใน E-R Model สามารถแบ่งออกได้เป็น 6 ประเภทดังนี้

- Simple Attribute

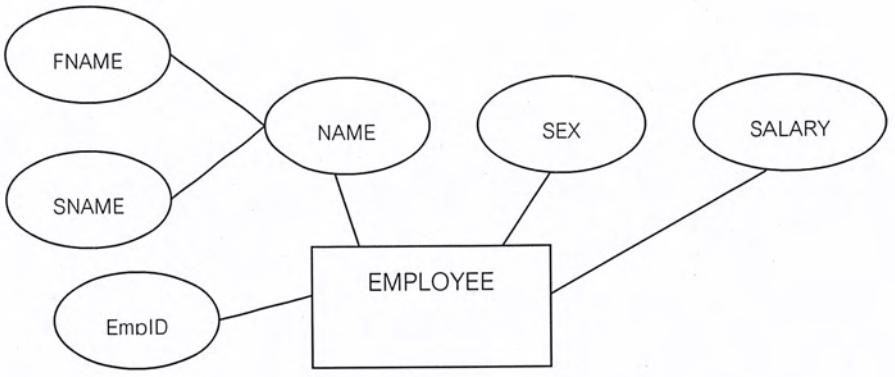
ได้แก่ Attribute ที่ค่าภายใน Attribute นั้น ไม่สามารถแบ่งย่อยได้อีก เช่น เพศ เงินเดือน อายุ จังหวัด ฯลฯ เป็นต้น สำหรับรูปภาพที่ใช้แทน Attribute ประเภทนี้ได้แก่ วงรีที่มีเส้นเชื่อมต่อไปยัง Entity ที่เป็นเจ้าของ Attribute นั้น โดยมีชื่อของ Attribute นั้นอยู่ภายใน เช่น Attribute "EmpID", "NAME", "SEX" และ "SALARY" ของ Entity "EMPLOYEE" ดังรูป



รูปที่ 3-8 แสดง Simple Attribute

- Composite Attribuey

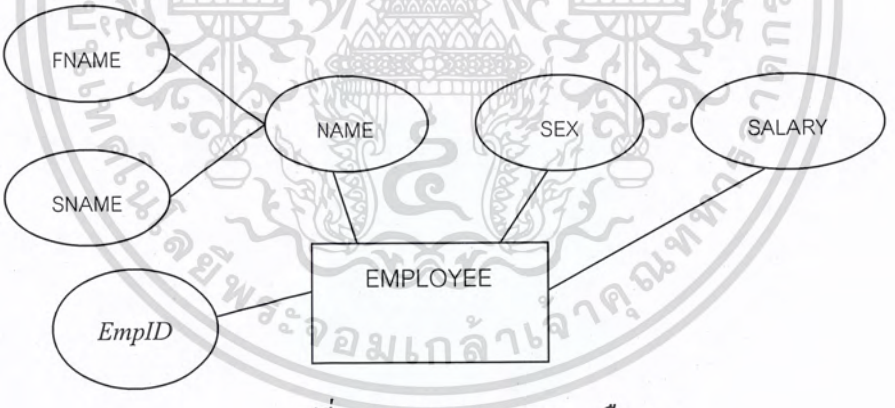
เป็น Attribute ที่มีลักษณะตรงข้ามกับ Simple Attribute กล่าวคือ จะเป็น Attribute ที่ค่าภายใน Attribute นั้น สามารถแยกเป็น Attribute ย่อยได้อีก เช่น "ชื่อ" ที่สามารถแบ่งย่อยออกเป็น "คำนำหน้าชื่อ" "ชื่อ" และ "นามสกุล" หรือ "ที่อยู่" ที่สามารถแบ่งย่อยออกเป็น "เลขที่บ้าน" " ซอย" "ถนน" "หมู่ที่" "แขวง/ตำบล" "เขต/อำเภอ" "จังหวัด" เป็นต้น สำหรับรูปภาพที่ใช้แทน Attribute ประเภทนี้จะใช้วงรีเช่นเดียวกับ Simple Attribute แต่จะเป็นวงรีที่ต่อเชื่อมกับวงรีของ Simple Attribute ที่เป็นเจ้าของ Composite Attribute นั้น เช่น Attribute "FNAME" และ "SNAME" ของ Entity "EMPLOYEE" ดังรูป



รูปที่ 3-9 แสดง Composite Attribute

- Identifier หรือ Key

อาจใช้คำว่า "Identifier" แทน Identifier ได้แก่ Attribute หรือกลุ่มของ Attribute ที่มีค่าในแต่ละสมาชิกของ Entity ไม่ซ้ำกันเลย ซึ่งถูกนำมาใช้กำหนดความเป็นเอกลักษณ์ให้กับแต่ละสมาชิกใน Entity เช่น Attribute "EmpID" ของ Entity "EMPLOYEE" ซึ่งใช้แทนรหัสประจำตัวพนักงาน สำหรับรูปภาพที่ใช้แทน Key ของ Entity จะใช้รูปวงรีเช่นเดียวกับ Attribute แต่จะมีเส้นขีดอยู่ใต้ Attribute ที่เป็น Key ดังรูป



รูปที่ 3-10 แสดง Identifier หรือ Key

- Single-valued Attribute

เป็น Attribute ที่มีค่าของข้อมูลภายใต้ค่าของ Attribute ใด Attribute หนึ่ง เพียงค่าเดียว เช่น Attribute "SALARY" ซึ่งที่ใช้เก็บเงินเดือนของพนักงาน ซึ่งพนักงานแต่ละคนจะมีเงินเดือนเพียงค่าเดียว ดังตัวอย่างข้อมูลต่อไปนี้

EMPLOYEE = {(สมชาย,20000),(สมบูรณ์,15000),(เจนจิรา,6000),(มยุรฉัตร,35000)}

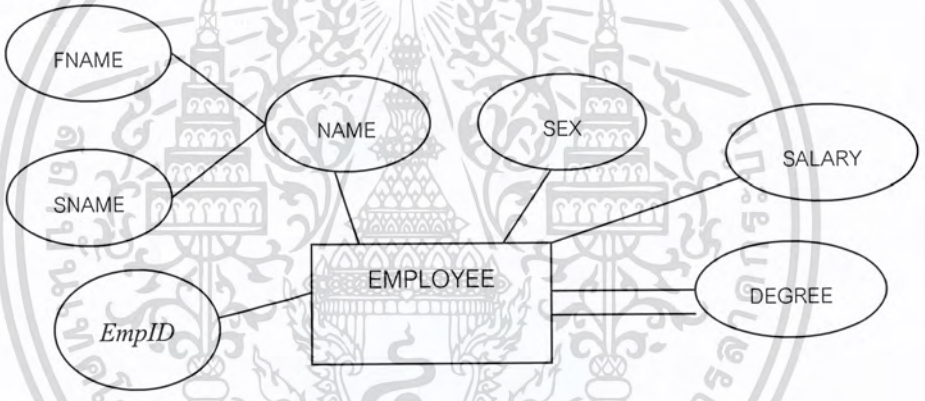
สำหรับรูปภาพที่ใช้แทน Attribute แระเภทนี้ จะใช้รูปภาพเช่นเดียวกับ Simple Attribute

- Multi-valued Attribute

เป็น Attribute ที่มีลักษณะตรงข้ามกับ Attribute แบบ Single-valued กล่าวคือ เป็น Attribute ที่มีค่าของข้อมูลได้หลายค่าภายใต้ค่าของ Attribute ใด Attribute หนึ่ง เช่น Attribute "DEGREE" ที่ใช้ระบุระดับการศึกษาของพนักงานแต่ละคน ซึ่งพนักงานแต่ละคน จะมีระดับการศึกษาได้หลายระดับ ดังตัวอย่างข้อมูลต่อไปนี้

EMPLOYEE = {(สมชาย,ม.6,ปริญญาตรี), (สมบูรณ์,ม.6,ปริญญาตรี,ปริญญาโท),
(เจนจิรา,ม.6), (มยุรฉัตร,ม.6,ปริญญาตรี,ปริญญาโท,ปริญญาเอก)}

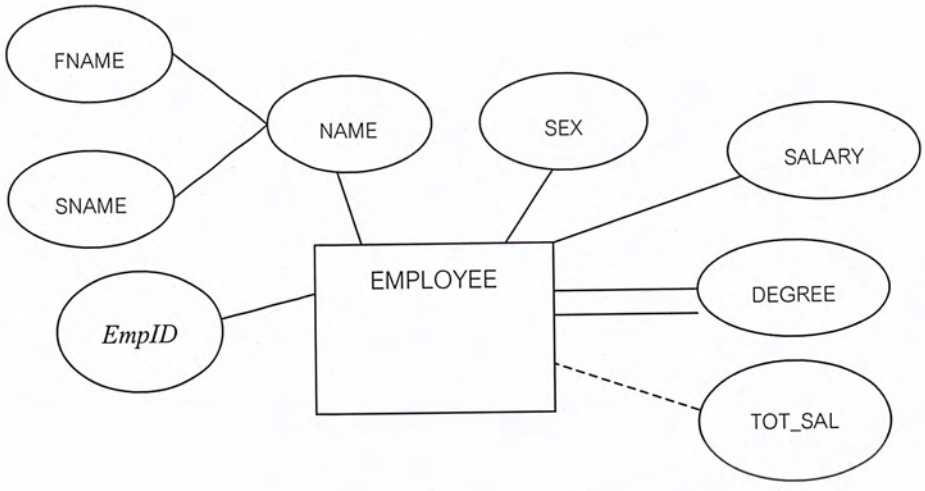
สำหรับรูปภาพที่ใช้แทน Attribute ประเภทนี้ จะใช้รูปภาพเช่นเดียวกับ Simple Attribute แต่เส้นที่ใช้เชื่อมระหว่างรูปภาพของ Attribute ประเภทนี้กับรูปภาพของ Entity หรือ Relationship จะใช้เส้น 2 เส้น แทนดังรูป



รูปที่ 3-11 แสดง Multi-valued Attribute

- Derived Attribute

เป็น Attribute ที่ค่าของข้อมูลได้มาจากการนำเอาค่าของ Attribute อื่น มาทำการคำนวณ ซึ่งค่าของ Attribute ประเภทนี้ จะต้องเปลี่ยนแปลงทุกครั้ง เมื่อมีการเปลี่ยนแปลงค่าของ Attribute ที่ถูกนำค่ามาคำนวณเช่น Attribute "TOT_SAL" ของ Entity "EMPLOYEE" ที่ใช้เก็บเงินเดือนสะสมของพนักงานแต่ละคนเพื่อนำไปคำนวณภาษี ซึ่งได้มาจากผลรวมของค่าใน Attribute "INCOME" ของ Entity "MTHLY_SALARY" ซึ่งเป็นเงินเดือนที่พนักงานแต่ละคนได้รับในแต่ละเดือน สำหรับรูปภาพที่ใช้แทน Attribute ประเภทนี้ จะใช้รูปภาพเดียวกับ Simple Attribute แต่เส้นที่ใช้เชื่อมระหว่างรูปภาพของ Attribute ประเภทนี้กับรูปภาพของ Entity หรือ Relationship จะใช้เส้นปะแทน ดังรูป



รูปที่ 3-12 แสดง Derived Attribute

3.5 Relationship

ได้แก่ การนำเอา Entity มารวมกันแบบ Aggregation Abstraction ดังนั้น สมาชิกของ Relationship จึงเกิดจากการจับคู่กันระหว่างสมาชิกของ Entity ที่มารวมกันภายใต้ Relationship นั้น เช่น

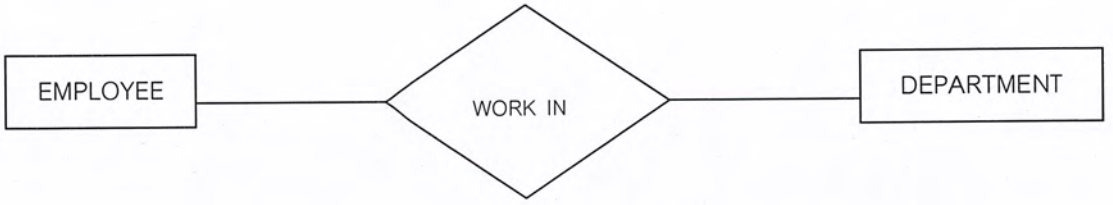
EMPLOYEE = {E1,E2,E3,E4}

DEPARTMENT = {D1,D2,D3}

WORK_IN = {(E1,D2), (E2,D1), (E3,D3), (E4,D1)}

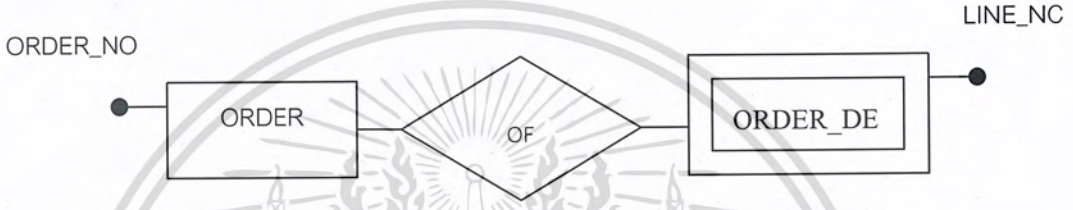
จะสังเกตเห็นว่า สมาชิกของ Relationship "WORK_IN" ก็คือ คู่ลำดับที่เกิดจากการจับคู่กันของสมาชิกของ Entity "EMPLOYEE" และ "DEPARTMENT" เช่น Employee "E1" ที่จับคู่กับ Department "D2" เป็นต้น

Relationship ที่สร้างขึ้นนี้ จะใช้แทนความสัมพันธ์ใดความสัมพันธ์หนึ่งระหว่างสมาชิกของ Entity ที่มารวมกันภายใต้ Relationship นั้น เช่น Relationship "WORK_IN" ที่ใช้แสดงฝ่ายที่พนักงานแต่ละคนสังกัด เป็นต้น ดังนั้นในการตั้งชื่อให้กับ Relationship จึงควรที่จะตั้งชื่อที่แสดงถึงความสัมพันธ์นั้นๆ สำหรับรูปภาพที่ใช้แทน Relationship ใน E-R Model จะได้แก่ รูปสี่เหลี่ยมข้าวหลามตัดที่มีชื่อของ Relationship นั้นอยู่ภายในรูปภาพของ Relationship นี้ จะไม่สามารถปรากฏอยู่เดี่ยวๆ ได้ แต่จะต้องปรากฏอยู่คู่กับ Entity เสมอ ดังรูป



รูปที่ 3-13 แสดง Relationship ที่เกิดจากการจับคู่กันระหว่างสมาชิกของ Entity

สำหรับ Relationship ที่ใช้กับ Weak Entity จะใช้รูปภาพเช่นเดียวกับรูปภาพของ Relationship โดยทั่วไป เช่น Entity "ORDER_DETAIL" ซึ่งเป็น Weak Entity ที่มีความสัมพันธ์กับ Entity "ORDER" ผ่านทาง Relation "OF" ดังรูป



รูปที่ 3-14 แสดง Relationship ที่ใช้กับ Weak Entity

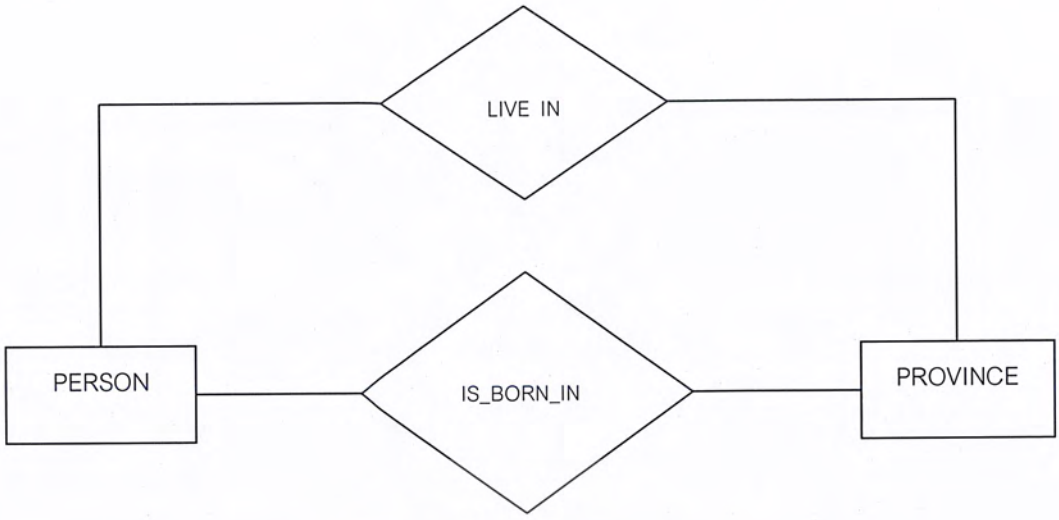
ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

ORDER = {O1,O2,O3}

ORDER_DETAIL = {P1,P2,P3,P4}

OF = {(O1,P1), (O1,P3), (O2,P1), (O2,P2), (O3,P3), (O3,P4)}

Relationship ระหว่าง Entity ใดๆ ไม่จำเห็นที่จะต้องมีเพียง Relationship เดียว ถ้าความสัมพันธ์ระหว่างสมาชิกใน Entity เหล่านั้น มีมากกว่า 1 ความสัมพันธ์ เช่น ความสัมพันธ์ระหว่างสมาชิกของ Entity "PERSON" ซึ่งเป็นข้อมูลของบุคคลต่างๆ และสมาชิกของ Entity "PROVINCE" ซึ่งเป็นข้อมูลของจังหวัดนอกเหนือจากจะแสดงถึงภูมิลำเนาของบุคคลต่างๆ ด้วย Relationship "LIVES_IN" แล้วยังสามารถแสดงถึงสถานที่เกิดของบุคคลด้วย Relationship "IS_BORN_IN" ได้ดังรูป



รูปที่ 3-15 แสดงสมาชิกใน Entity ที่มีมากกว่า 1 ความสัมพันธ์

ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

PERSON = {P1,P2,P3}

PROVINCE = {C1,C2,C3}

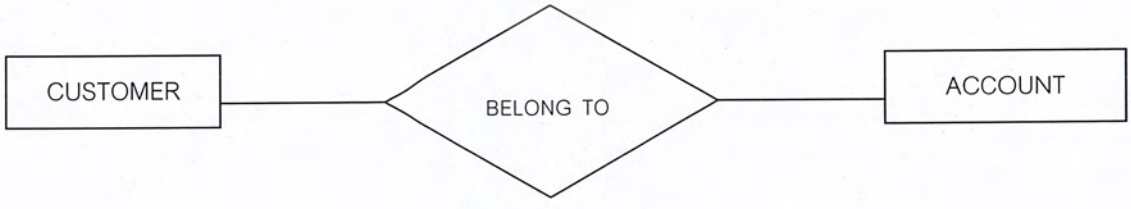
IS_BORN_IN = {(P1,C1), (P2,C2), (P3,C3)}

LIVES_IN = {(P1,C2), (P2,C1), (P3,C3)}

สมาชิกใน Entity ที่เกี่ยวข้องกับ Relationship จะถูกเรียกว่า Participant ซึ่งจำนวนของ Participant นี้ จะถูกนำไปใช้กำหนดประเภทของ Relationship ได้ดังนี้

One – to – One Relationship

เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่ง จะมีความสัมพันธ์กับอีก Participant ของอีก Entity หนึ่งเพียง Participant เดียว เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้เพียงบัญชีเดียว และแต่ละบัญชีเงินฝากจะมีเจ้าของบัญชีได้เพียงคนเดียว ดังรูป



รูปที่ 3-16 แสดง One – to – One Relationship

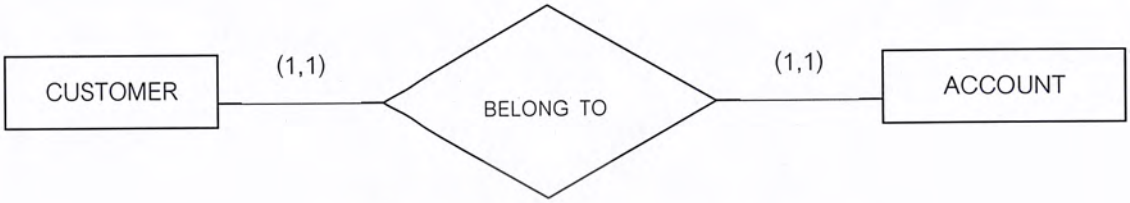
ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

CUSTOMER = {C1,C2,C3}

ACCOUNT = {A1,A2,A3}

BELONG_TO = {(C1,A1), (C2,A2), (C3,A3)}

หรืออาจใช้การระบุ Mapping Cardinality แทน



รูปที่ 3-17 แสดงการระบุ Mapping Cardinality

One – to – Many Relationship

เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่งมีความสัมพันธ์กับ Participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากจะต้องมีเจ้าของบัญชีเพียงคนเดียว ดังรูป

รูปที่ 3-18 แสดง One – to – Many Relationship

ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

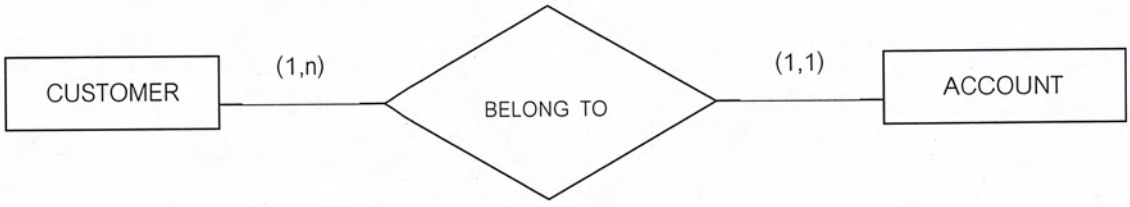
CUSTOMER = {C1,C2,C3}

ACCOUNT = {A1,A2,A3,A4}

BELONG_TO = {(C1,A1), (C2,A3), (C1,A4), (C3,A2)}

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Customer "C1" เป็นเจ้าของบัญชีเงินฝากถึง 2 บัญชี คือ บัญชีเงินฝาก "A1" และ "A4" แต่สำหรับแต่ละบัญชี จะมีเจ้าของบัญชีได้เพียงคนเดียว สำหรับสัญลักษณ์ที่ใช้กับ Relationship ประเภทนี้ ได้แก่ ตัวเลข 1 และ ตัวอักษร M โดยตัวเลข 1 จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน Participant ที่เดี่ยวข้องกับ Relationship เพียง Participant เดียว ส่วนตัวอักษร M จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ซึ่งจากรูป ด้านเดียว ส่วนด้าน Entity "ACCOUNT" จะเป็น Entity ที่มีจำนวน Participant ที่

เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ดังนั้น จึงปรากฏตัวเลข 1 ไว้ทางด้าน Entity "CUSTOMER" และ ตัวอักษร M ทางด้าน Entity "ACCOUNT" อาจจะใช้การระบุด้วย Mapping Cardinality แทน ดังรูป



รูปที่ 3-19 แสดงการระบุด้วย Mapping Cardinality

Many – to – Many Relationship

เป็น Relationship ที่ Participant มากกว่า 1 Participant ของ Entity หนึ่ง มีความสัมพันธ์กับ Participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากสามารถมีเจ้าของบัญชีได้มากกว่า 1 คน ดังรูป

รูปที่ 3-20 แสดง Many – to – Many Relationship

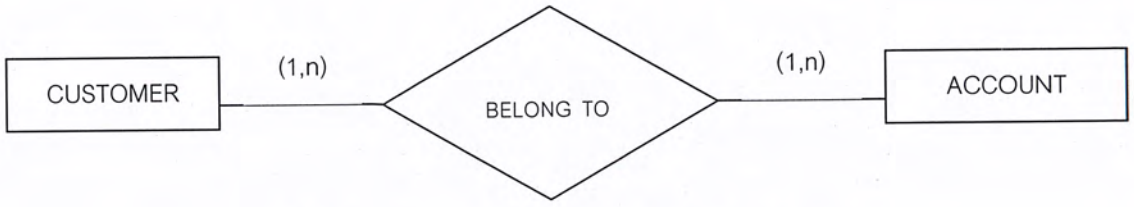
ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

CUSTOMER = {C1,C2,C3}

ACCOUNT = {A1,A2,A3}

BELONG_TO = {(C1,A1), (C2,A2), (C1,A3), (C3,A1)}

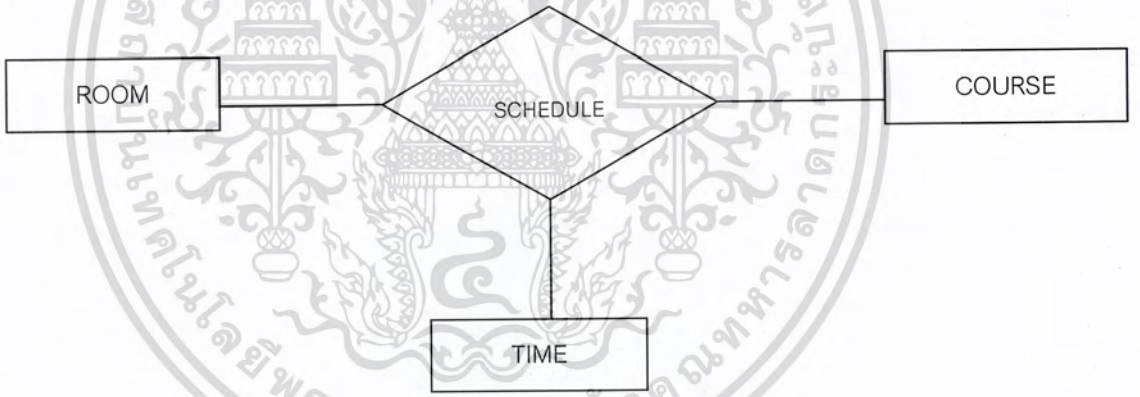
จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Customer "C1" เป็นเจ้าของบัญชีเงินฝากถึง 2 บัญชี ในขณะเดียวกันบัญชี "A1" ก็มีเจ้าของบัญชี 2 คนเช่นเดียวกัน สำหรับสัญลักษณ์ที่ใช้กับ Relationship ประเภทนี้ ได้แก่ ตัวอักษร M โดยจะกำหนดไว้ทั้ง 2ด้านของ Entity แต่อาจใช้การระบุด้วย Mapping Cardinality แทนดังรูป



รูปที่ 3-21 แสดงการระบุด้วย Mapping Cardinality

นอกเหนือจากการใช้จำนวนของ Participant ในการจัดประเภทของ Relationship แล้ว ยังสามารถใช้จำนวนของ Entity ที่มีความสัมพันธ์กับแต่ละ Relationship มากำหนดประเภทของ Relationship ได้ดังนี้

1. Binary Relationship เป็น Relationship ที่พบมากที่สุดใน E-R Diagram โดยเป็น Relationship ที่เกิดขึ้นระหว่าง 2 Entity ใดๆ เช่น E-R Diagram ที่แสดงความสัมพันธ์ระหว่างลูกค้ากับบัญชีเงินฝาก เป็นต้น
2. N-ary Relationship เป็น Relationship ที่เกิดขึ้นระหว่าง Entity มากกว่า 2 Entity ขึ้นไป เช่น Relationship "SCHEDULE" ซึ่งใช้แสดงตารางเรียนของวิชาต่างๆ ดังรูป



รูปที่ 3-22 แสดง N-ary Relationship

ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

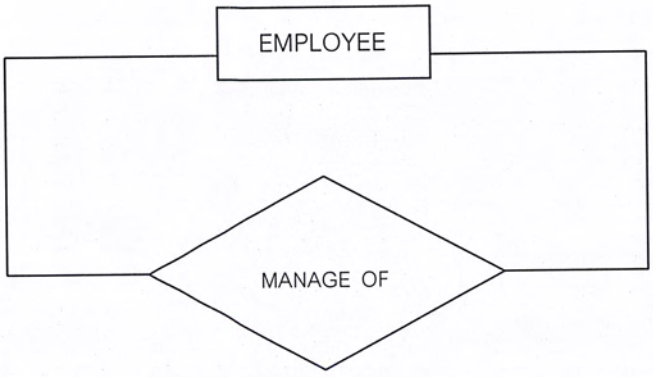
ROOM = {R1,R2,R3}

TIME = {T1,T2,T3}

COURSE = {C1,C2,C3}

SCHEDULE = {(R1,T1,C1), (R2,T3,C2), (R3,T2,C3)}

3. Recursive Relationship เป็น Relationship ที่เกิดขึ้นกับ Entity เดียว ในกรณีที่ Attribute ของ Entity นั้น สามารถสร้างความสัมพันธ์กับอีก Attribute หนึ่งภายใน Entity เดียวกัน เช่น Relationship "MANAGE_OF" ซึ่งใช้แสดงชื่อหัวหน้างานของพนักงานแต่ละคนดังรูป



รูปที่ 3-23 แสดง Recursive Relationship

ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

EMPLOYEE = {E1,E2,E3,E4}

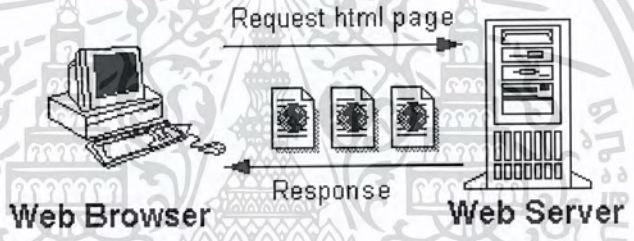
MANAGE_OF = {(E1,E4), (E2,E4), (E3,E4), (E4,-)}

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Relationship "MANAGE_OF" จะใช้สมาชิกของ Entity "EMPLOTYEE" เองมาสร้างความสัมพันธ์ขึ้น

บทที่ 4
สถาปัตยกรรมของระบบ

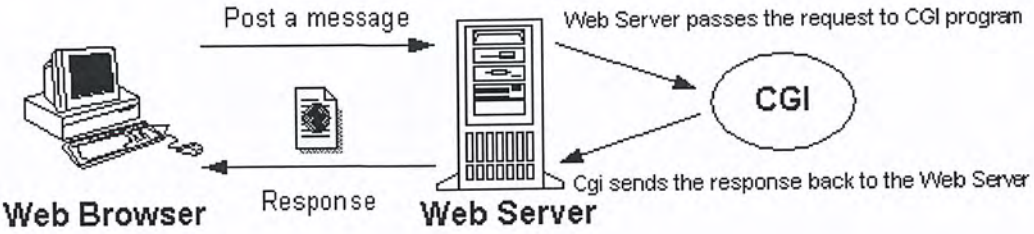
4.1 สถาปัตยกรรมของจาวาบนเว็ป (Java Architecture On the Web)

วิวัฒนาการของ Server Side Application ที่ถูกใช้บนเวปตั้งแต่สมัยแรก ๆ จนถึงปัจจุบัน สมัยแรกสุดตอนที่มีเวปเกิดขึ้นมาใหม่ ๆ ส่วนประกอบหลักของเวปไม่ค่อยจะมีอะไรซักเท่าไหร่ เริ่มแรกสุดก็ทำกันอย่างง่าย ๆ โดยใช้หลักการพื้นฐานของ client/server architecture ซึ่งส่วนที่เป็น client ในเวปก็คือ web browser ยกตัวอย่างเช่น NCSA Mosaic, Lynx, Netscape และส่วนที่เป็น server ซึ่งก็คือ web server (ตอนนั้นจะมีแค่ NCSA web server ซึ่งเป็นต้นแบบของ Apache web server) นั่นเอง โดยทั่วไป web server หนึ่ง ๆ จะสามารถบริการส่งข้อมูลให้ตัว web browser ได้หลายตัว ซึ่งถ้าพูดกันก็คงจะเป็นลักษณะที่คนหลายคน (multiple clients) สามารถที่จะเข้าไปดูข้อมูลต่าง ๆ จากเวปไซต์เดียวกัน (1 server) ได้ในเวลาเดียวกันนั่นเอง



รูปที่ 4-1 แสดงพื้นฐาน client/server architecture เริ่มแรกสุดที่ใช้กันบนเวป

ต่อมาทั้งคนทำเวปและคนดูเวปก็เริ่มเบื่อจึงคิดค้นหาวิธีที่ทำให้เวปมีสีสันมากขึ้น เทคโนโลยีที่เกิดขึ้นตามมาก็คือ **CGI (Common Gateway Interface)** หลักการของ CGI ก็คือแทนที่ข้อมูลที่เราส่งไปให้ web server (โดยผ่านทาง web browser) จะถูก process โดยตัว web server อย่างเดียว ข้อมูลนี้ก็อาจจะถูกส่งไปให้อีกโปรแกรมหนึ่งซึ่งตัวโปรแกรมนี้จะสามารถถูกเรียกได้โดย web server เพื่อทำการประมวลผลข้อมูลที่มาจากเรา โดยทั้งนี้ข้อมูลของเราจะถูก process ที่ใดก็ขึ้นอยู่กับ Setting ต่าง ๆ ที่ทางเวปไซต์ที่เราเข้าไปดูกำหนดขึ้นมา ยกตัวอย่างเช่น ถ้าเราต้องการดูเพจธรรมดา ๆ หลักจากที่เราคลิกไปที่ตัวลิงค์ตัว web server ก็จะไปอ่านเพจ (html ไฟล์) ที่เราต้องการแล้วส่งกลับมาเป็น text stream ให้เรา แต่ถ้าเราต้องการที่จะ post ข้อความลงไปในเวป ทาง web server ก็จะส่งข้อความที่เราพิมพ์ไปให้โปรแกรมที่ทำหน้าที่ดูแลและเก็บข้อความที่เราส่งเข้าไป ยกตัวอย่างเช่นโปรแกรม message board เป็นต้น เทคโนโลยี CGI นี้สามารถเขียนขึ้นโดยภาษาอะไรก็ได้แต่ภาษาที่นิยมใช้กันอย่างกว้างขวางก็มักจะเป็น Perl และ C ตัวอย่างของ CGI Application ที่เราพบเห็นกันอยู่ในปัจจุบันก็อาจจะเป็น Counter, Guestbook, Message Board, Send Mail หรือแม้กระทั่งโปรแกรมส่งเพจ (Paging) เป็นต้น



รูปที่ 4-2 การ post ข้อความไปยัง CGI โปรแกรม

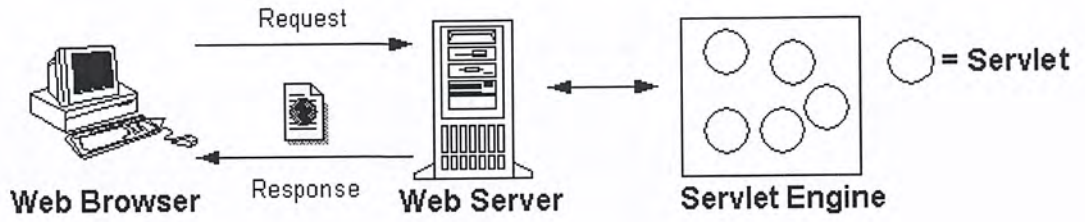
หลักจากที่ CGI กลายเป็นที่นิยมจนเป็นส่วนหนึ่งของ WWW Standard แล้ว Server Side Application ที่เป็นลูกหลานก็เกิดขึ้นกันมาอย่างมากมาย ยกตัวอย่างเช่น ASP, PHP, ColdFusion, Servlet และอื่น ๆ (อย่างไรก็ตามบทความนี้จะกล่าวถึงเทคโนโลยีที่เกี่ยวข้องกับ Server Side Application ที่ใช้จา ว่า platform เป็นหลักเท่านั้น)

4.2 Java Servlet

จาวามีสโลแกนอันหนึ่งที่กล่าวไว้ว่า *"สิ่งใดก็ตามที่เป็นที่นิยมใช้กันอย่างกว้างขวาง ท้ายที่สุดสิ่งนั้นจะกลายเป็นจาวา library"* โดยผลพวงหนึ่งที่มาจากสโลแกนอันนี้ก็คือ Servlet นั่นเอง

Servlet อ้างอิงหลักการของ CGI โดยข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาว่านั่น จาว่าเป็นภาษาที่ใช้คอนเซ็ปของ Object Oriented ในการเขียน หลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมคงจะทราบดีว่า Object Oriented สามารถลดความซับซ้อนโครงสร้างของโปรแกรมรวมไปถึงอำนวยความสะดวกในการ reuse ส่วนประกอบต่าง ๆ ของโปรแกรมที่เขียนไว้แล้วเพียงใด นอกจากนี้จาว่ายังเป็นภาษาที่เป็น platform independent ซึ่งจะช่วยให้เราสามารถทำการพัฒนาระบบโดยใช้ environment อะไรก็ได้ซึ่งโดยทั่วไปมักจะเป็น Widows environment โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบน Unix environment เพื่อเพิ่มความเสถียรภาพของโปรแกรมอีกทีหนึ่ง นอกจากนี้ Servlet ยังมีความเร็วที่เหนือกว่า CGI เพราะ Servlet ใช้หลักการของ thread โดยจะทำการสร้าง 1 thread ต่อหนึ่ง request ที่มาจาก client ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request ซึ่งจะทำให้เปลืองทรัพยากรมากกว่าและ process ในการรันก็จะช้ากว่าด้วย

ถึงแม้ว่า Servlet จะอ้างอิงหลักการของ CGI อย่างไรก็ตามในการที่จะทำการรัน Servlet แล้ว ตัว web server จะไม่สามารถส่งข้อมูลไปให้ Servlet ได้โดยตรงเหมือนกับหลักการของ CGI แต่ตัว web server จะต้องเพิ่มอีกส่วนหนึ่งซึ่งเป็นส่วนที่ใช้เป็นเสมือนตัวห่อหุ้ม Servlet ต่าง ๆ ไว้โดยส่วนที่เพิ่มขึ้นมานี้เราเรียกว่า Servlet Engine หรือ Servlet Container



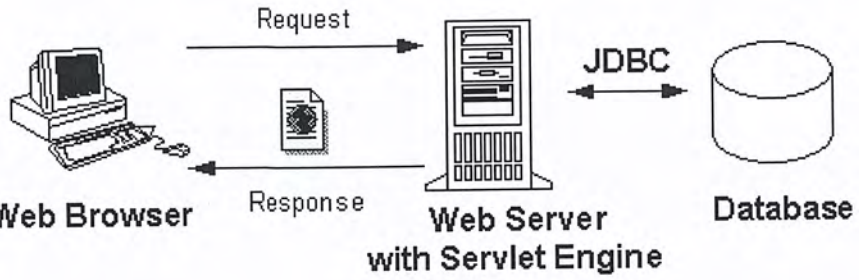
รูปที่ 4-3 แสดง Servlet Engine and its Servlets

โดยทั่วไป Servlet Engine จะเป็นส่วนที่มี Java Virtual Machine (JVM) อยู่ในตัวเองโดย Servlet Engine นี้จะมีหน้าที่รับ request จาก web server (ซึ่งมาจาก web browser) แล้วทำการเลือกตัว Servlet ขึ้นมาทำการประมวลผล request นั้นภายใต้ JVM ของมัน โดยผลที่ได้จากการประมวลผลของ Servlet ที่ถูกเลือกจะถูกส่งกลับไปยัง web server โดย web server นี้จะส่งผลกลับไปยัง web browser ในท้ายที่สุด ตัวอย่างของ Servlet Engine ที่ใช้กันอยู่ในปัจจุบันก็อาจจะเป็น Apache Jserv, Apache Tomcat, Allaire Jrun, IBM Websphere, BEA Weblogic, Servlet Exec เป็นต้น

4.3 Relational Database

ช่วงที่มี Server Side Application ใหม่ ๆ ข้อมูลต่าง ๆ มักจะถูกเก็บไว้ในไฟล์โดยโปรแกรมจะทำการอ่านไฟล์ทุกครั้งที่มีการโหลดข้อมูลดังกล่าวขึ้นมาใช้ ตัวอย่างที่เราพบเห็นกันโดยทั่วไปสำหรับโปรแกรมที่ต้องอ่านหรือเก็บข้อมูลต่าง ๆ ลงไปในไฟล์ยกตัวอย่างเช่น เวปบอร์ด, guest book หรือแม้กระทั่ง counter เป็นต้น สำหรับเวปบอร์ดแล้วข้อมูลที่จะถูกจัดเก็บอาจจะเป็นลิสของข้อความที่มีอยู่ทั้งหมดในแต่ละกระทู้หรืออาจจะเป็นข้อมูลที่เกี่ยวกับ login และ password ของผู้ใช้แต่ละคนในกรณีที่เวปบอร์ดดังกล่าวจะอนุญาตให้ใช้กับเฉพาะบุคคลที่อยู่ในกลุ่มเท่านั้น การจัดเก็บข้อมูลในไฟล์มีผลเสียในแง่ของการยากต่อการดูแลและจัดเก็บ ตลอดจนประสิทธิภาพและความเร็วที่ลดลงของโปรแกรมในแง่ของการตอบสนองกับผู้ใช้ปลายทางเนื่องจากที่โปรแกรมมีกิจกรรมที่เกี่ยวข้องกับ I/O เป็นหลักใหญ่ ซึ่งในท้ายสุดการจัดเก็บข้อมูลในไฟล์ก็ถูกแทนที่ด้วยการจัดเก็บข้อมูลในเดต้าเบสแทน

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

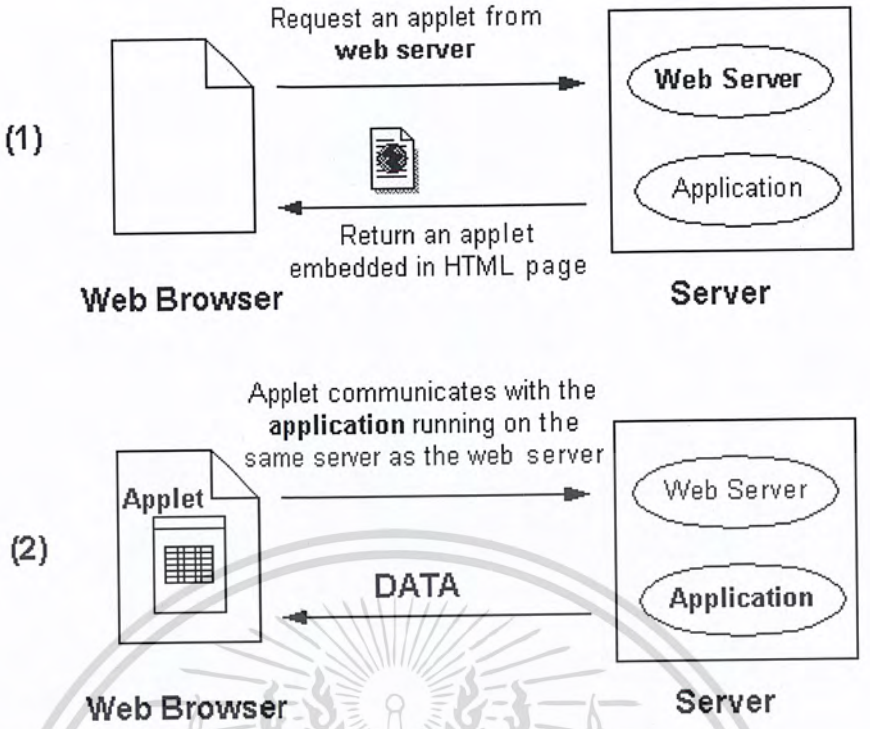


รูปที่ 4-4 แสดง Servlets(in Servlet Engine) connecting to Relational Database via JDBC

4.4 Applet (on Web Browser) to Application

ทุกคนคงจะรู้จัก Applet กันเป็นอย่างดี Applet เป็นโปรแกรมที่เขียนขึ้นโดยใช้จาว่าซึ่งจริงๆแล้วจะสามารถรันที่ไหนก็ได้ที่มี JVM อยู่ แต่โดยทั่วไปตัว Applet นี้มักจะถูกรันภายใต้ JVM ที่อยู่ใน Web Browser หรือไม่ก็ AppletViewer เสียมากกว่า ในช่วงแรก ๆ Applet ที่ใช้รันบนเวปมักจะถูกเขียนขึ้นเพื่อใช้ตกแต่งเพจต่าง ๆ ให้มีความสวยงามแต่ด้วยความสามารถที่ว่า Applet สามารถสร้าง connection กลับไปยังเซฟเวอร์ที่เป็นตัวเก็บ Applet ดังกล่าวได้ Applet จึงถูกใส่ความสามารถอื่น ๆ เข้าไปเกินกว่าที่จะเป็นเพียง client side application แต่เพียงอย่างเดียว วิธีการแรกที่นิยมใช้กันก็คือการให้ Applet ติดต่อกับ Application ที่รันอยู่ที่ตัวเซฟเวอร์ที่เก็บ Applet นั้นไว้โดยหลักการก็คือการใช้ web server เป็นแค่เพียงตัวเก็บ Applet เพื่อให้ web browser สามารถโหลด Applet มาจาก web server ได้โดยหลักจากที่ web browser ทำการโหลด Applet เสร็จแล้ว ตัว Applet ก็จะทำการสร้าง connection ไปยัง Application ที่รันอยู่บน Server เดียวกับที่รัน web server แทน หลักการดังกล่าวกลายเป็นจุดเริ่มต้นของคำ ๆ หนึ่งคือ Application Service Provider (SPP) ซึ่งก็หมายถึงบริษัทที่ให้บริการแก่ลูกค้าในการใช้ application ต่าง ๆ ผ่านทางเวปโดยไม่ต้องทำการ install ตัว application นั้นลงไปที่เครื่องของลูกค้าแต่จะทำการโหลด application นั้นมาใช้เมื่อถึงเวลาที่ต้องการนั่นเอง (บางคนจะเรียกว่า apps-on-tap โดยจะเปรียบ applications เป็นเหมือนกับน้ำที่เราใช้ดื่มใช้กินโดยได้มาจากการเปิดก๊อกน้ำ[tap])

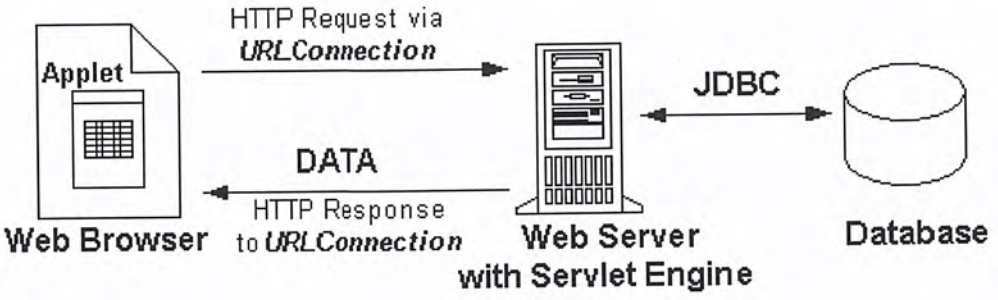
ตัวอย่างของ Applet พวกนี้ก็อาจจะเป็นพวก Stock Monitor ที่ใช้ Applet ในการดูดัชนีของหุ้นต่าง ๆ ในแบบ real-time หรืออาจจะเป็น online Banking ซึ่งเป็น Applet ที่ธนาคารใช้เพื่อให้บริการต่าง ๆ กับลูกค้าเช่น การฝาก/ถอน การโอนเงิน การตรวจเช็คยอดบัญชีต่าง ๆ เป็นต้น



รูปที่ 4-5 แสดง Applet to Application Communication

4.5 Applet (on Web Browser) to Servlet

ในกรณีที่เวปไซด์ต้องการความสวยงามหรือความ interactive กับผู้ใช้อย่างสูงจนกระทั่งการใช้ HTML และ Widget ต่าง ๆ (เช่นปุ่มหรือพวก html form) ที่ทาง web browser จัดเตรียมไว้ให้ไม่สามารถที่จะให้ความสวยงามตลอดจนความยืดหยุ่นเพียงพอต่อความต้องการของผู้ที่ออกแบบเวปไซด์นั้นได้ Applet ก็มักเป็นอีกทางเลือกหนึ่งที่หลาย ๆ คนนิยมใช้เพื่อลดช่องว่างในส่วนนี้ ลักษณะของ Applet ที่ติดต่อกับ Servlet จะต่างจาก Applet ที่ใช้ติดต่อกับ Application ตรงที่ว่า Applet ที่ใช้ติดต่อกับ Servlet จะยังคงใช้ HTTP protocol ในการติดต่อสื่อสารกับ Server โดยถ้าเปรียบเทียบก็คือแทนที่เราจะให้ user ใช้ HTML Form ต่าง ๆ ของ web browser ในการติดต่อสื่อสารกับ Servlet (ที่อยู่ที่ web server) เราก็ใช้ Applet เป็นตัวแทนของ HTML Form แทน ซึ่งนอกจากที่ Applet จะสามารถแทนที่ HTML Form ได้แล้ว เราก็ยังสามารถเพิ่มความสวยงามและความ interactive เข้าไปใน Applet ได้ตามแต่จินตนาการของเราอีกด้วย หลายคนอาจจะสงสัยว่าแล้ว Applet จะติดต่อกับ Servlet ได้อย่างไร วิธีการที่นิยมใช้กัน*ก็คือ แทนที่จะให้ web browser สร้าง HTTP Request ไปยัง Servlet เหมือนอย่างแต่ก่อนเราก็ให้ Applet ที่รันอยู่บน web browser สร้าง HTTP Request ติดต่อไปยัง Servlet แทนโดยวิธีการเช่นนี้เราเรียกว่า HTTP Tunnel ซึ่งสามารถทำได้ง่าย ๆ โดยให้ Applet สร้าง HTTP Connection ไปยัง Servlet ผ่านทางคลาส java.net.URLConnection



รูปที่ 4-6 แสดง Applet to Servlet communication via java.net.URLConnection class

เมื่อไรก็ตามที่มีการเปลี่ยนแปลงเกิดขึ้นกับ Applet ซึ่งเป็นผลมาจาก activitity ต่าง ๆ ของผู้ใช้ ตัว Applet จะทำการส่งค่าที่เปลี่ยนแปลงนี้ไปยัง Servlet เพื่อทำการประมวลผล โดยหลังจากที่ Applet ได้รับผลที่ส่งไปกลับมาจาก Servlet แล้ว Applet จะทำการตีความผลที่ได้เพื่อทำการเปลี่ยนแปลงตัวมันเองโดยผลที่ได้สำหรับผู้ใช้ก็คือการเปลี่ยนแปลงของค่าตัวเลขต่าง ๆ ที่อยู่บน Applet ในกรณีที่ Applet นั้นถูกใช้สำหรับแสดงค่าจำนวนตัวเลข ตลอดจนการเปลี่ยนแปลงรูปร่างหน้าตาของ Applet ไปยัง state อื่น ๆ ในกรณีของ Applet ที่ใช้สำหรับปฏิทิน เป็นต้น

วิธีการ Applet to Servlet นี้นิยมใช้กันมากเพราะเป็นการลดโหลดหรือการคำนวณที่หนักหน่วงจากส่วนของ client (Applet) ไปยังส่วนของ server (Servlet) แทน โดยมักจะมีศัพท์คำหนึ่งที่ใช้แทนลักษณะ architecture แบบนี้ซึ่งเราเรียกว่า thin client

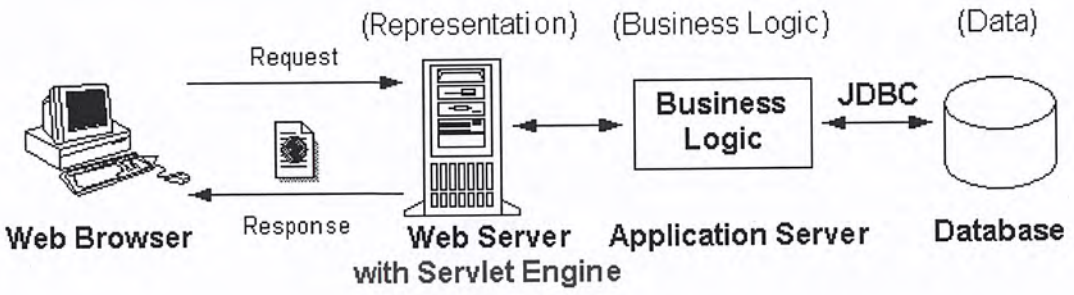
นอกจาก HTTP โปรโตคอลแล้ว Applet อาจจะใช้ RMI หรือ CORBA ในการติดต่อสื่อสารกับ Servlet ที่สนับสนุน Middleware สองอันนี้ก็ได้

4.6 Application Server

สำหรับเวปไซด์เล็ก ๆ แล้วโครงสร้างของ Server Side Application ที่กล่าวถึงข้างต้นก็มักจะเพียงพอต่อความต้องการสำหรับประโยชน์ใช้สอย อย่างไรก็ตามสำหรับเวปไซด์ใหญ่ ๆ ที่มีผู้ใช้จำนวนมหาศาล ส่วนประกอบที่สำคัญอีกอย่างหนึ่งที่มักจะถูกเพิ่มเข้าไปมักจะเป็นส่วนที่เรียกว่า Application Server

Application Server เป็นส่วนประกอบอีกส่วนหนึ่งที่ถูกสร้างขึ้นมาเพื่อสนองความต้องการที่ว่า *ในการสร้างระบบที่ดีระบบหนึ่งขึ้นมานั้น เราควรจะแยกส่วนที่ทำหน้าที่เกี่ยวกับการบริการต่าง ๆ รวมไปถึงการจัดสรรข้อมูลออกมาเป็นอีกส่วนต่างหาก(Business Logic) ทั้งนี้เพื่อเพิ่มความคล่องตัวในการเปลี่ยนแปลงของระบบ รวมไปถึงการจัดระบบให้แบ่งออกเป็นส่วน ๆ อย่างชัดเจนมากยิ่งขึ้น* โดยถ้าดูจากบทบาทของ Application Server ที่ผ่านตั้งแต่อดีตจนถึงปัจจุบันเราอาจจะพูดได้ว่า Application Server ก็คือส่วนที่ช่วยอำนวยความสะดวกให้กับ Server Side Application โดยมักจะเป็นตัวช่วยในการคำนวณข้อมูล (Services), ช่วยเป็นตัวกลางในการจัดส่งข้อมูล(Messaging Services), ช่วยในการควบคุมการจัดเก็บข้อมูลลงไปในเดต้าเบสอย่างถูกต้อง(Transaction) ตลอดจนช่วยในการเพิ่มความเร็วให้กับระบบโดยการเก็บ

ออฟเจคต่าง ๆ ที่ใช้หรือไม่ใช้แล้วไว้เพื่อนำมาใช้อีก(Caching) เป็นต้น (ในแง่ของการออกแบบระบบ Application Server มักจะเป็นส่วนที่เราใช้ในส่วนที่เรียกว่า Services Layer ซึ่งในจาว่าแล้วส่วนนี้ก็มักจะเกี่ยวข้องกับ Enterprise Java ต่าง ๆ เช่น EJB, JTS, JMS, JNDI นั่นเอง)



รูปที่ 4-7 แสดง Application Server

ประโยชน์อย่างหนึ่งของ Application Server ที่เรามักพบเห็นโดยทั่วไปก็คือการนำมาใช้สำหรับรัน EJB (Enterprise Java Bean) ถ้าพูดกันในเชิงนิยามแล้ว EJB ไม่ได้ถูกรันที่ Application Server แต่จะถูกรันกับสิ่งหนึ่งที่เรียกว่า EJB Server อย่างไรก็ตาม Application Server ส่วนมากมักจะสนับสนุน EJB ซึ่งเป็นส่วนหนึ่งของ J2EE ดังนั้น Application Server ส่วนมากจึงมีส่วนหนึ่งที่ทำหน้าที่เป็น EJB Server เพื่อรัน EJB ต่าง ๆ ได้. EJB แบ่งออกอย่างคร่าว ๆ เป็นสามแบบคือ

1. Session Bean เป็น EJB ที่มักจะถูกใช้เป็นตัวเก็บรายละเอียดของผู้ใช้ที่กำลังทำกิจกรรมบางอย่างกับระบบหรืออาจจะถูกใช้เป็น Object Pool สำหรับ object ต่าง ๆ ที่ JSP หรือ Servlet สามารถเรียกใช้เพื่ออำนวยความสะดวกในการประมวลผลตลอดจนใช้ในการเปลี่ยนแปลงสถานะต่าง ๆ ของระบบโดยเรามักเรียกส่วนนี้ว่า Services Layer นั่นเอง
2. Entity Bean เป็น EJB ที่ใช้สำหรับเก็บเดต้าต่าง ๆ ที่เราทำการแมปมาจากเดต้าที่อยู่ในเดต้าเบส, ไฟล์, LDAP objects หรืออื่น ๆ ซึ่งจุดหลักของการนำเดต้าต่าง ๆ มาใส่ไว้ใน Entity Bean ก็คือการเพิ่มความง่ายในการเปลี่ยนแปลงเดต้าโดยผ่านทาง EJB Object, การง่ายต่อการควบคุมการอ่านและเขียนเดต้าต่าง ๆ ได้อย่างถูกต้องในกรณีที่ระบบมีมากกว่า 1 thread เข้ามาเกี่ยวข้องในการอ่านเขียนหรือเปลี่ยนแปลงเดต้าตัวเดียวกันในเวลาเดียวกัน รวมไปถึงความเร็วที่เพิ่มขึ้นในการเก็บเดต้าต่าง ๆ ไว้ในหน่วยความจำ (Bean Caching) แทนที่จะต้องทำการอ่านเดต้าใหม่จากเดต้าเบสทุกครั้งที่เราต้องการทำกิจกรรมบางอย่างกับเดต้าเหล่านั้น โดยข้อดีต่าง ๆ ที่กล่าวมาข้างต้นเหล่านี้จะเกิดขึ้นได้จากการทำงานร่วมกันระหว่าง Entity Bean, EJB Container และ EJB Server อย่างไรก็ตามนักพัฒนาระบบก็ยังคงถกเถียงกันอยู่ถึง performance ที่เกิดขึ้นจากการใช้ Entity Bean เพื่อเก็บเดต้าต่าง ๆ ไว้ โดยผลกระทบนี้มาจากการที่แต่ละ thread จะต้องแย่งกันเข้าไปใช้ Entity Bean ตัวเดียวกันเพื่อทำการเขียนอ่าน

หรือแปลงแปลงเดต้า(Bean Contention) ซึ่งผลที่เกิดขึ้นอาจจะทำให้ระบบช้าลงกว่าการใช้ JDBC หรือ Object Database ธรรมดาทั่ว ๆ ไปก็ได้

3. Message-driven Bean(2.0 Proposed Final Draft) เป็น EJB ชนิดใหม่ที่ถูกเพิ่มเข้ามาเพื่อทำการสนันสนุน JMS (Java Message Service) โดย EJB นี้จะถูกควบคุมได้ใช้ event ต่าง ๆ ที่เกิดขึ้นจาก JMS Message

B2B (Business to Business)

นอกเหนือจากติดต่อสื่อสารระหว่าง web browser กับ Server Side Application แล้วข้างหลังฉากของ Server Side Application จากที่ ๆ หนึ่งอาจมีการติดต่อสื่อสารเพื่อแลกเปลี่ยนข้อมูลหรือทำการซื้อขายต่าง ๆ กับ Server Side Application ของอีกที่หนึ่งก็ได้ เรามักจะเรียกการติดต่อสื่อสารแบบนี้ว่า B2B (Business to Business) เทคโนโลยีในจาวาที่นิยมใช้กันใน B2B มักจะเป็น JMS (Java Message Service), Socket ที่นักพัฒนานิยาม application protocol ขึ้นมาเอง, CORBA (Common Object Request Broker Architecture) หรืออาจเป็น Servlet to Servlet communication โดยมักนิยมใช้ SSL (Secure Sockets Layer) สำหรับ encrypt ข้อมูลและอาจใช้ XML เป็นเสมือน message สำหรับใช้รับส่งข้อมูลของทั้งสองฝ่ายก็ได้

บทที่ 5
การเชื่อมต่อกับฐานข้อมูล

5.1 Java DataBase Connectivity

ภาษาจาวาเป็นภาษาคอมพิวเตอร์ที่ได้รับการพัฒนาอย่างรวดเร็วและต่อเนื่องนอกเหนือจากการสร้างกราฟิกหรือองค์ประกอบที่ใช้ในการประดับโฮมเพจ (Homepage) ยังมีการพัฒนาองค์แระกอบใหม่ๆ ในการปฏิบัติงานในด้านต่างๆ ได้ถูกเพิ่มเติมประสิทธิภาพให้แก่จาวาคือ JDBC (Java Database Connectivity) ซึ่งทำหน้าที่อำนวยความสะดวกในการเชื่อมต่อและสื่อสารกับระบบฐานข้อมูล โดยพื้นฐานของภาษาจาวาที่ได้รับการออกแบบมาเพื่อปฏิบัติงานกับระบบเครือข่ายคอมพิวเตอร์โดยตรง องค์ประกอบนี้จึงทำให้จาวาเหมาะสมเป็นอย่างยิ่งในการเป็นเครื่องมือสำหรับสร้างโปรแกรมประยุกต์เพื่อเข้าถึงข้อมูลแบบไคล์เอนท์เซอร์เวอร์ โดยมี JDBC เป็นองค์ประกอบในการเชื่อม ในการถ่ายทอดคำสั่งการเรียกค้นข้อมูลจากไคล์เอนท์ ไปสู่เซอร์เวอร์ และรับข้อมูลผลลัพธ์จากเซอร์เวอร์ กลับคืนไปสู่ไคล์เอนท์

ระบบไคล์เอนท์เซอร์เวอร์ (Client-Server) ในปัจจุบันมีการเพิ่มขึ้นของค่าใช้จ่ายของบุคลากรที่มาทำหน้าที่ดูแลบำรุงรักษา Database Server ซึ่งได้แก่บุคลากรด้านการจัดการและด้านบริการสนับสนุน สำหรับ Network ขนาดเล็ก (โดยทั่วไปน้อยกว่า 20 ผู้ใช้) โดยปกติผู้จัดการ Network เพียงคนเดียวสามารถรับภาระหน้าที่ดูแลบำรุงรักษา Database Server ควบคุมการเข้าใช้ Database Server จากบรรดาผู้ใช้ทั้งหลาย และให้บริการสนับสนุน Front End เมื่อจำนวนผู้ใช้ฐานข้อมูลเพิ่มจำนวนขึ้น หรือฐานข้อมูลตัวมันเองมีขนาดใหญ่ขึ้น โดยปกติแล้วก็เริ่มจำเป็นที่ต้องจ้างผู้จัดการฐานข้อมูลเพิ่มอีกคนเพื่อมาทำหน้าที่รันระบบจัดการฐานข้อมูลและให้บริการสนับสนุนในส่วน Front End ด้วย นอกจากนี้แล้วในกรณีที่ระบบจัดการฐานข้อมูลรันอยู่บนระบบปฏิบัติการที่บุคลากรด้านบริการสนับสนุนไม่คุ้นเคยนั้นก็ต้องมีการอบรม ซึ่งเป็นค่าใช้จ่ายที่ต้องเพิ่มเข้าไปในค่าใช้จ่ายเริ่มแรกด้วย

นอกจานนี้แล้วค่าใช้จ่ายด้านฮาร์ดแวร์ก็เพิ่มขึ้นด้วย ในขณะที่มีฐานข้อมูล ไคล์เอนท์เซอร์เวอร์เป็นจำนวนมากที่รันอยู่ภายใต้ระบบปฏิบัติการที่นิยมใช้กันเช่น Netware OS2 และ Unix เป็นต้น และบริษัทผู้ขายส่วนใหญ่ก็ประกาศว่าระบบจัดการฐานข้อมูลของตนเองนั้นสามารถรันอยู่บนฮาร์ดแวร์เดียวกับที่ File Server Software รันอยู่ได้ก็ตาม แต่ว่าโดยปกติแล้วถ้าต้องการสมรรถนะที่สูงและความถูกต้องของข้อมูลที่สูงก็สมควรให้ Database Server แยกไปรันคนละคอมพิวเตอร์เฉพาะกิจของ Database เอง สิ่งนี้หมายความว่าต้องทำการซื้อคอมพิวเตอร์เพิ่มอีกเครื่อง เพื่อมาทำเป็น Database Server โดยเฉพาะคอมพิวเตอร์นี้ต้องมีกำลังสนับสนุนเช่น เครื่องป้องกันไฟฟ้า (UPS) เพื่อป้องกัน เซอร์เวอร์ เสียหายจากเหตุการณ์ไฟตกหรือไฟดับ

ค่าใช้จ่ายโดยรวมของ Software ของระบบไคล์เอนท์เซอร์เวอร์ ปกติแล้วสูงกว่าระบบจัดการฐานข้อมูลแบบหลายผู้ใช้บนพีซี ค่าใช้จ่ายต่อหนึ่งเซอร์เวอร์ สำหรับฐานข้อมูลไคล์เอนท์เซอร์เวอร์ ที่สามารถบริการผู้ใช้ได้แบบไม่จำกัดจำนวน มีราคาสูง บวกกับค่าใช้จ่ายแยกออกไปของ front-end หรือ

เครื่องมือพัฒนา software พร้อมทั้งค่าใช้จ่ายบุคลากรสำหรับการฝึกอบรมโปรแกรมเมอร์ในระบบใหม่ กล่าวได้ว่าค่าใช้จ่ายของระบบไคล์เอนท์เซอร์เวอร์สูงกว่าของระบบจัดการฐานข้อมูลบนพีซีมาก

ยังมีเรื่องที่น่ายุ่งยากอีกเรื่องหนึ่งคือ ระบบไคล์เอนท์เซอร์เวอร์ นั้นมีส่วยประกอบต่างๆ เป็นจำนวนมากตามหลักทั่วไปที่กล่าวว่า ยิ่งระบบใดมีส่วนประกอบเป็นจำนวนมากโอกาสที่ส่วนประกอบต่างๆ บนระบบนั้นจะเสียก็มีมากตามไปด้วย เมื่อเกิดเหตุการณ์ร้ายแรงและระบบเกิดพังเสียหายก็เป็นการยากที่จะชี้ชัดถึงปัญหาที่แท้จริงได้ ระยะเวลาที่ใช้ในการทำให้สิ่งทุกอย่างกลับสู่สภาพปกติและเริ่มทำงานได้อีกก็เป็นเวลาที่ยาวนาน สิ่งต่างๆ ที่เกิดขึ้นเช่นนี้มาจากการขาดประสบการณ์และความชำนาญของบุคบากรด้านบริการสนับสนุนและโปรแกรมเมอร์ ทั้งนี้เนื่องจากไคล์เอนท์เซอร์เวอร์ เป็นเทคโนโลยีใหม่ เมื่อเวลาผ่านไปและระบบไคล์เอนท์เซอร์เวอร์เริ่มเป็นที่นิยมใช้มากขึ้นปัญหาที่ได้กล่าวมาแล้วก็จะลดน้อยลง

องค์ประกอบทางด้าน Hardware และ Software รวมกัน ซึ่งระบบจัดการฐานข้อมูลไคล์เอนท์เซอร์เวอร์รันอยู่ใน Platform ต่างๆ เช่น พีซี, Unix Workstation, Super Server, Minicomputer, Mainframe ซึ่งพีซีเป็น Platform ที่นิยมใช้กันมากที่สุดสำหรับไคล์เอนท์เซอร์เวอร์

บรรดาระบบฮาร์ดแวร์ต่างๆ ทั้งหลายก็มีความแตกต่างทั้งในเรื่องคุณลักษณะและความสามารถ อย่างไรก็ตามก็มีคุณลักษณะร่วมที่จำเป็นสำหรับระบบปฏิบัติการ โดยปกติแล้ว Application ถูกเขียนให้รันภายใต้ระบบปฏิบัติการใดระบบหนึ่งโดยเฉพาะ ซอฟต์แวร์ระบบปฏิบัติการที่นิยมใช้กัน เช่น MS Dos , PC-Dos, OS/2, Unix, VMS ของ Digital, MVS/XA ที่รันบน IBM Mainframe เป็นต้น

คุณลักษณะเบื้องต้นของระบบปฏิบัตอการที่จำเป็นสำหรับระบบจัดการฐานข้อมูลไคล์เอนท์เซอร์เวอร์ คือ ความสามารถในการทำ Multitasking หรือการรัน Application หลาย Application พร้อมกัน แบบ Multitasking ทำให้ซอฟต์แวร์ระบบจัดการฐานข้อมูลสามารถบริการการสิบถามต่างๆ และคำขอต่างๆของบรรดาผู้ใช้ทั้งหลายได้อย่างถูกต้อง โดยปราศจากการรบกวนซึ่งกันและกัน multitasking ทำงานโดยแบ่งเวลาประมวลผลของซีพียูออกเป็นช่วงๆ (time Slices) ให้แก่บรรดา Task ต่างๆ และบรรดา Process ต่างๆ ระบบปฏิบัติการ Multitasking ทั้งที่เป็นแบบ Preemptive และแบบ Non-Preemptive

นอกจากนี้บางระบบปฏิบัติการยังสามารถเป็นระบบ Multi-user ได้ (คือยอมให้มีผู้ใช้หลายคนทำงานที่แตกต่างกันได้พร้อมกัน) โดยเฉพาะเมื่อมีการใช้เทอร์มินอลธรรมดาทั้งหลายเป็นทางเข้าสู่ฐานข้อมูลโอเอสแบบ multi-user ไม่ได้ ก่อให้เกิดข้อดีหรือข้อเสียโดยเฉพาะเมื่อถูกใช้เป็น Platform สำหรับระบบจัดการฐานข้อมูลไคล์เอนท์เซอร์เวอร์

การเพิ่มความสามารถด้าน Multithread เข้าไปในซอฟต์แวร์โอเอส Multitask แบบ Preemptive ความสามารถ Multithread ทำให้ Application ทำ Multitask ได้ภายในตัวเองได้ ตัวอย่างเช่น ระบบจัดการฐานข้อมูลผู้ใช้คนเดียวแลล Multithread สามารถทำการ Start ใหม่ (Process หรือ Task) ให้ทำงานออกรายงานที่ซับซ้อนโดยทำอยู่ใน Background ในขณะที่ผู้ใช้กำลังทำการสอบถาม ซึ่งทำอยู่ใน foreground Multithread เป็นปัจจัยที่สำคัญสำหรับการออกแบบระบบจัดการฐานข้อมูลไคล์เอนท์เซอร์เวอร์ที่ซับซ้อน Multithread ทำให้ Application สามารถควบคุมให้ Task ใหม่เริ่มทำงานหรือหยุดทำงานได้ Application สามารถถูกออกแบบให้มีความชาญฉลาดภายในที่ตัดสินได้ว่า Process ใดหรือ Task ใดมีลำดับความ

สำคัญและควรได้รับเวลา ซีพียูมาก ตัวอย่างเช่น ระบบจัดการฐานข้อมูลให้การแก้ไขข้อมูลมีลำดับความสำคัญสูงกว่าการสอบถามข้อมูล เป็นต้น

เนื่องจากจุดด้อยในเรื่องการไม่มี Platform ใดที่สามารถสนองความต้องการได้ทุกประการหรือเหมาะสมในทุกสถานการณ์ในระบบไคล์เอนท์เซอร์เวอร์ ในปัจจุบันมีการประยุกต์ใช้ระบบที่เรียกว่า "Network Computing" เพื่อช่วยแก้ไขในปัญหาของระบบไคล์เอนท์เซอร์เวอร์ ในเรื่องการใช้บุคลากรในการดูแลระบบน้อยลงคือจะไม่ต้องปดูแลในส่วนของไคล์เอนท์ ที่ในอนาคตจะมีจำนวนเพิ่มขึ้นเรื่อยๆ การง่ายในการปรับปรุงระบบไม่ต้องคอยตามแก้ไขที่ทุกเครื่องไคล์เอนท์ และจากปัญหาการทำงานไม่ครบทุก Platform ในปัจจุบันมีภาษาคอมพิวเตอร์หนึ่งที่สามารถเขียนโปรแกรมคอมพิวเตอร์โดยไม่คำนึงถึง Platform แระกอบกับความสามารถทางด้านระบบ Network Compution ที่ภาษานี้สามารถนำมาประยุกต์ใช้ได้อย่างดี ภาษาดังกล่าวคือ ภาษาจาวา

5.2 ความหมายของ JDBC

JDBC (Java DataBase Connectivity) ถูกพัฒนาโดย JavaSoft Department ของบริษัท Sun MicroSystems ซึ่งก็คือฟังก์ชั่นมาตรฐานหรือ Java Application Programming Interface (API) สำหรับการเชื่อมต่อกับระบบฐานข้อมูล นักพัฒนาสามารถใช้ JDBC API และยังประกอบด้วย packages อื่นๆ ด้วย ซึ่งนำเสนอในรูปแบบ function พิเศษหรือ higher level API เพื่อเขียนโปรแกรมอิสระในการกำหนด DBMS หรือ database connectivity mechanism ในการเริ่มใช้งาน โดยทั่วไปการใช้ SQL database ในการติดต่อ (Interface) กับ framework เพื่อที่จะจัดมาตรฐานในการติดต่อในส่วนบนสุดของชนิดต่างๆ ของ database connectivity modules ซึ่งก็คือมาตรฐานของ ANSI SQL-2 Entry level database เพราะว่า relational databases เกือบจะทั้งหมดในปัจจุบันใช้มาตรฐานของ SQL-2 Entry level

JDBC สร้างระดับการเชื่อมต่อเพื่อสื่อสารกับฐานข้อมูลในรูปแบบที่คล้ายคลึงกับ ODBC Data Base Connectivity ของบริษัท Microsift) ซึ่งในปัจจุบันได้ถือว่าเป็นมาตรฐานของการเชื่อมต่อกับฐานข้อมูลสำหรับเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) และระบบเครือข่ายท้องถิ่น (LAN) แต่ตามหลักการทำงานของทั้ง JDBC และ ODBC ตั้งอยู่บนมาตรฐานเดียวกันคือ X/Open SQL Call-Level Interface ของระบบ X-Windows และ JDBC driver ต้องเข้ากันได้กับระดับมาตรฐานในการเข้าถึง SQL (ANSI SQL Entry Level Standard) และต้องผ่าน Conformance test ซึ่ง JavaSoft เป็นผู้กำหนดขึ้น

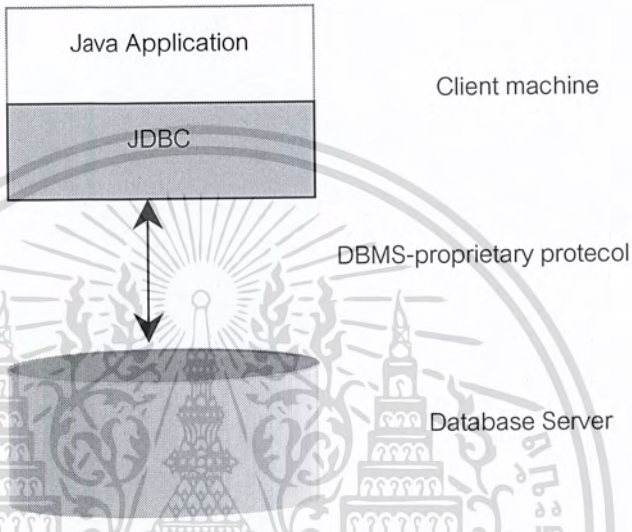
รูปแบบการเชื่อมต่อฐานข้อมูลของ JDBC

JDBC API สนับสนุนรูปแบบการเชื่อมต่อฐานข้อมูลทั้งแลล Two-Tier model และ Three-Tier model

Two-Tier model

จาวาแอพเพล็ท (Applet) หรือจาวาแอพพลิเคชั่น (Application) จะติดต่อกัฐานข้อมูลโดยตรงจึงมีความจำเป็นที่โปรแกรมจาวาต้องการ JDBC ไดร์ฟเวอร์พิเศษที่สามารถสื่อสารกับระบบจัดการฐานข้อมูลชนิดนั้นได้ คำสั่งในการเรียกค้นข้อมูลในรูปของภาษา SQL (Structured Language) จะถูกส่งจากผู้ใช้ไปสู่

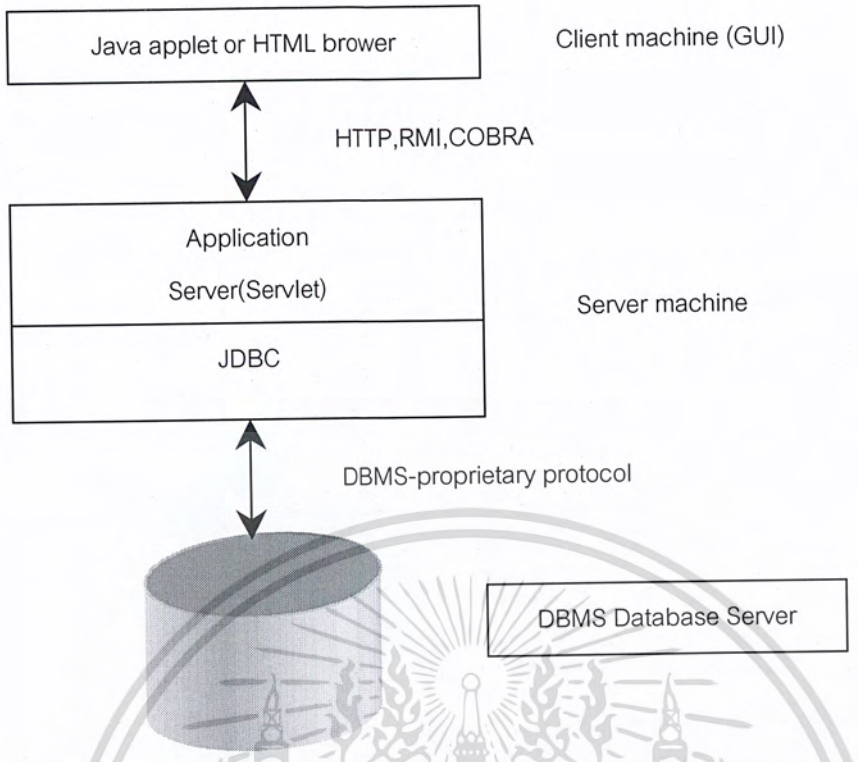
ฐานข้อมูล หลังจากนั้นผลจากการประมวลผลของระบบจัดการฐานข้อมูล ก็จะถูกส่งกลับมาสู่ผู้ใช้,นข้อมูลดังกล่าวนี้ส่วนมากจะติดตั้งอยู่ต่างเครื่องคอมพิวเตอร์ โดยผู้ใช้สามารถเชื่อมต่อ ผ่านระบบเน็ตเวิร์ค (Network) รูปแบบ Two Tier นี้ใช้หลักการทำงานเช่นเดียวกันกับรูปแบบไคล์เอนท์/เซอร์เวอร์ที่เรารู้จักกันดี โดยที่เครื่องคอมพิวเตอร์ของผู้ใช้คือ ไคล์เอ็นท์และเครื่องคอมพิวเตอร์ ที่ให้บริการฐานข้อมูลคือ เซิร์ฟเวอร์ เครือข่ายคอมพิวเตอร์ที่นิยมใช้รูปแบบ Two-Tier มักจะเป็นเครือข่ายอินทราเน็ต (Intranet) สำหรับดำเนินธุรกรรมภายในองค์กร



รูปที่ 5-1 แสดง Two-Tier Model

Three-Tier model

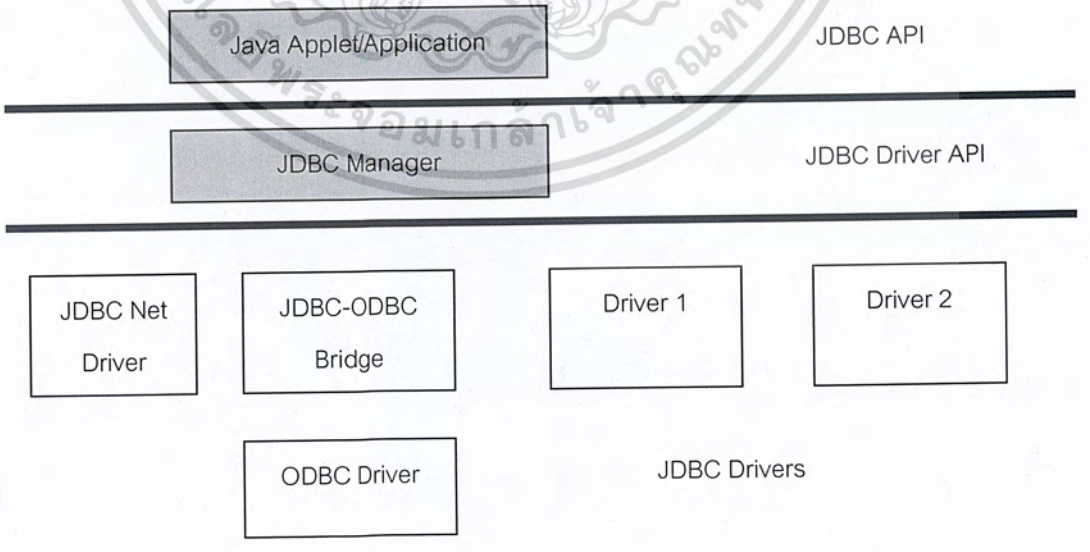
คำสั่งเรียกค้นต่างๆ จากผู้ใช้จะถูกส่งไปให้กับ Middle Tier หรือส่วนกลางของการบริการเสียก่อน หลังจากนั้น Middle Tier จะแปลงคำสั่งเหล่านี้ให้เป็นภาษา SQL เพื่อส่งไปให้กับระบบจัดการฐานข้อมูลเพื่อทำการประมวลผล ข้อมูลผลลัพธ์ที่ได้ก็จะถูกส่งกลับคืนไปให้กับ Middle Tier และส่งต่อไปให้ผู้ใช้ในที่สุด หลักการทำงานเช่นนี้มักจะพบในเครือข่ายอินเทอร์เน็ตซึ่งประกอบด้วยเครื่องคอมพิวเตอร์หลากหลายชนิด และเว็บเซิร์ฟเวอร์ (Web-Server) ซึ่งทำหน้าที่เป็น Middle Tier ก็จะเป็นตัวกลางในการจัดการให้คอมพิวเตอร์ทั้งไคล์เอ็นท์และเซิร์หเวอร์ฐานข้อมูลสามารถพูดคุยกันได้ การปรับเปลี่ยนระบบคอมพิวเตอร์ของไคล์เอ็นท์และเซิร์ฟเวอร์ หรือแม้กระทั่งการเปลี่ยนฐานข้อมูลตัวใหม่จะไม่มีผลกระทบซึ่งกันและกันเกิดขึ้น



รูปที่ 5-2 แสดง Three-Tier Model

5.3 โครงสร้างของ JDBC

โครงสร้างของการเชื่อมต่อภายใน JDBC ประกอบด้วย 3 ระดับหลักคือ JDBC API,JDBC Driver API และ JDBC Driver ดังรูป ข้างล่าง ระดับบน JDBC API เป็นระดับของฟังก์ชั่น API ที่อำนวยความสะดวกให้แก่โปรแกรมประยุกต์ ระดับกลาง JDBC Driver (มีไดร์ฟเวอร์ที่ต่างกันอยู่ 4 ชนิด) ที่เหมาะสม

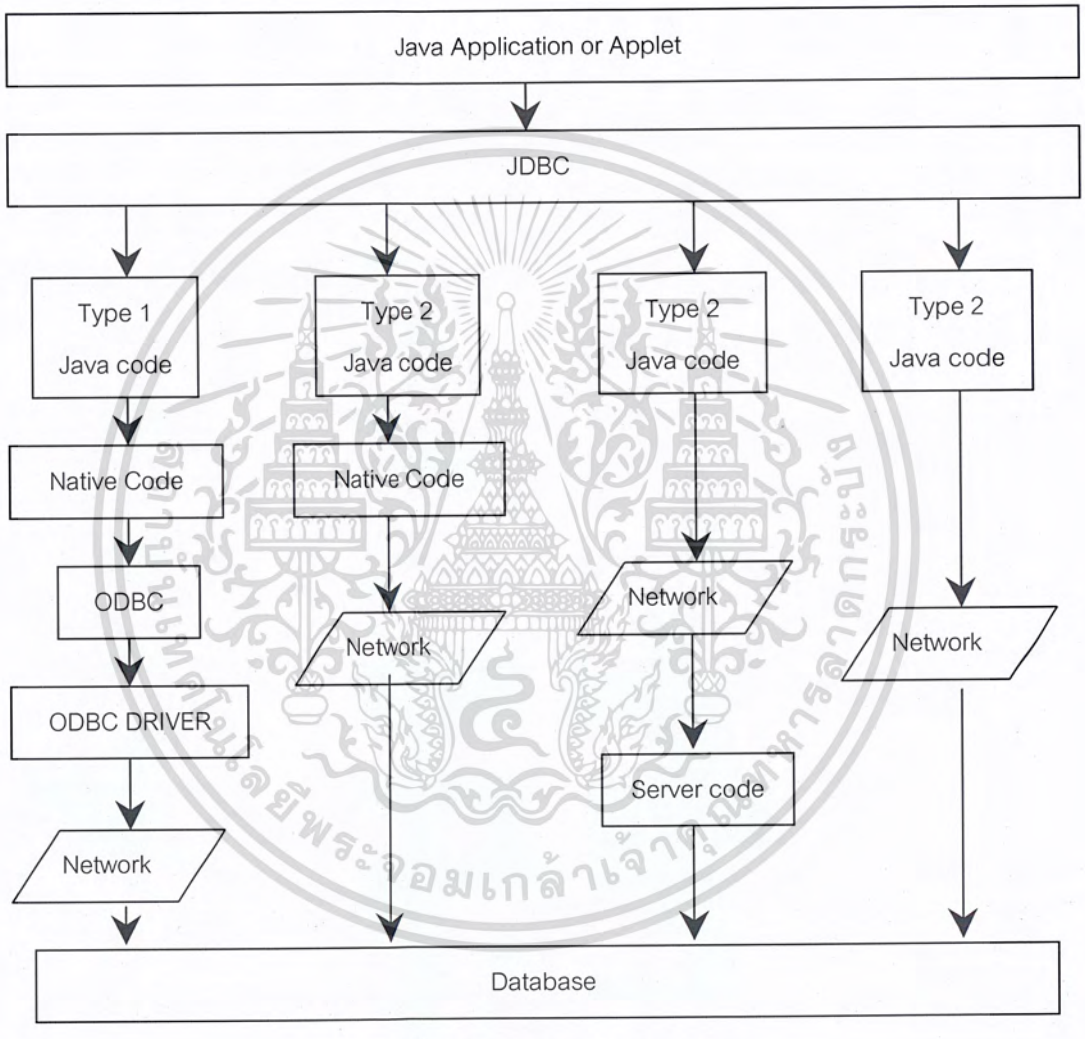


รูปที่ 5-3 แสดงระดับการเชื่อมต่อของ JDBC API

5.4 รูปแบบของ JDBC ไดร์ฟเวอร์ (Driver)

JDBC ไดร์ฟเวอร์หรือส่วนที่อยู่เบื้องหลังการทำงานของฟังก์ชัน API ต่างๆ ของ JDBC ถูกจัดแบ่งออกเป็นทั้งหมด 4 ชนิดคือ

1. JDBC-ODBC Bridge
2. Native-API (Partly-Java) Driver
3. Net-Protocol (All-Java) Drivers
4. Native-Protocol (All-Java) Drivers



รูปที่ 5-4 แสดง JDBC driver types

รายละเอียดการทำงานของไดร์ฟเวอร์แต่ละชนิดอธิบายได้ดังนี้

5.4.1 JDBC/ODBC Bridge

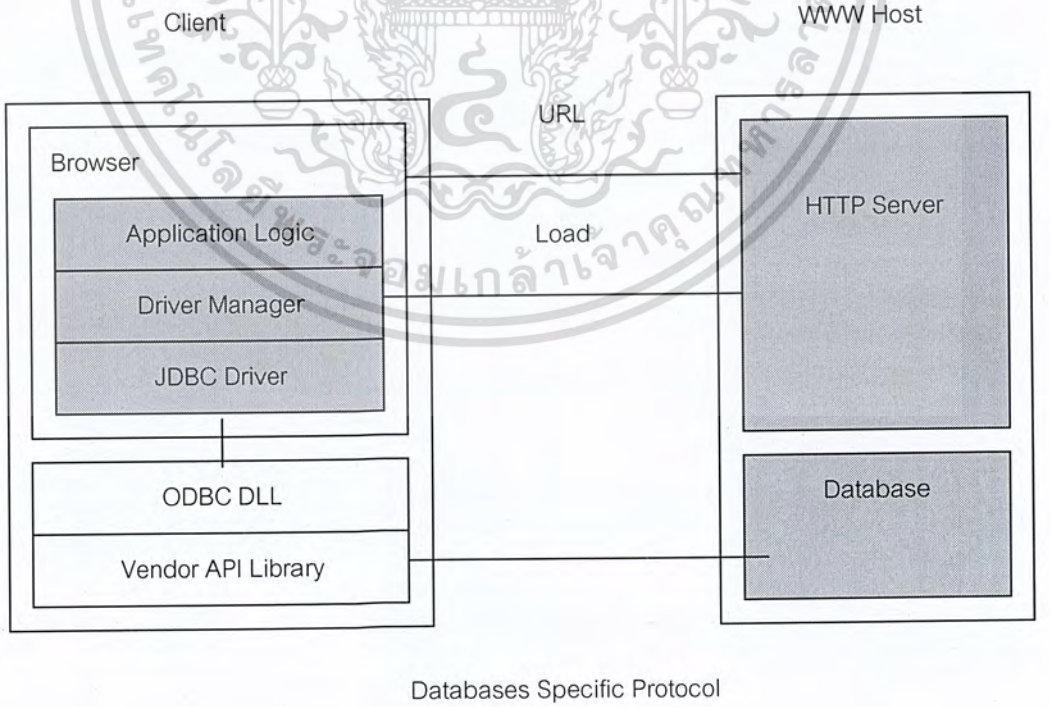
JDBC/ODBC bridge ถูกพัฒนาโดย JavaSoft และ Intersolv ทำหน้าที่เป็นตัวกลางในการเข้าถึงฐานข้อมูลได้โดยผ่านการทำงานของ ODBC โดยนำข้อดีของ ODBC – enabled data sources ที่มีอยู่โดย

ทั่วไปอย่างมากมาย , ฝั่งไคล์เอนท์ Java applet หรือ application จะถูกเขียนโดยใช้ JDBC API, Bridge จะทำการแปลงการเรียกใช้ JDBC ไปยัง ODBC และส่งค่า ODBC driver ที่เหมาะสมสำหรับ back-end database

ข้อดีของ bridge ทำให้ application สามารถติดต่อกับ databases ได้อย่างง่ายดายโดยจาก vendors ที่มากมายโดยเลือก ODBC driver ที่เหมาะสม อย่างไรก็ตามการติดต่อ database ประเภทนี้ต้องพิจารณาค่าใช้จ่าย (Overhead) และความซับซ้อน (Complexity) เพราะว่าการเรียกใช้จะมีลำดับการเรียกใช้ดังนี้คือ

JDBC ⟷ bridge ⟷ ODBC ⟷ Native client-API ⟷ database

การติดต่อโดยใช้ประเภทนี้ชนิดของ Driver จะไม่อนุญาตสำหรับการย้ายข้อมูลแบบ just-in-time ของ Java applet Native code ต้องมีการเตรียมการติดตั้งบนเครื่องไคล์เอนท์โดยตรงเพราะว่า JDBC-ODBC Bridge เป็นไดร์ฟเวอร์ที่ถูกเขียนขึ้นมาจากภาษา C/C++ ในบางส่วนจึงทำให้ไม่สามารถดาวน์โหลด (Download) ข้ามเครือข่ายอินเทอร์เน็ตและแปล (Interpret) บนเว็บบราวเซอร์ (Webbrowser) ได้ เพราะภาษา C/C++ ต้องรันเฉพาะแต่ละระบบเท่านี้น ดังนั้น ODBC ไดร์ฟเวอร์ในรูปแลลของไลบรารีไฟล์หรือ DLL (Dynamic Link Library) และ JDBC-ODBC Bridge ไดร์ฟเวอร์ต้องได้รับการติดตั้งบนไคล์เอ็นท์แต่ละเครื่องเพื่อให้สามารถติดต่อกับฐานข้อมูลที่รู้จักได้ มีความต้องการในการเตรียมการติดตั้ง software โดย Administrative รูปแบบเหมือน client-server application ดังนั้น JDBC/ODBC Bridge จะไม่สามารถแก้ไขปัญหาของไคล์เอนท์ program ดังกล่าวได้ จึงเหมาะสำหรับการใช้งานกับระบบ Three-Tier โดยไดร์ฟเวอร์ทั้งสองจะได้รับการติดตั้งอยู่ที่ Middle-Tier หรือเครื่องคอมพิวเตอร์ตัวกลาง เช่น Web-Server เป็นต้น



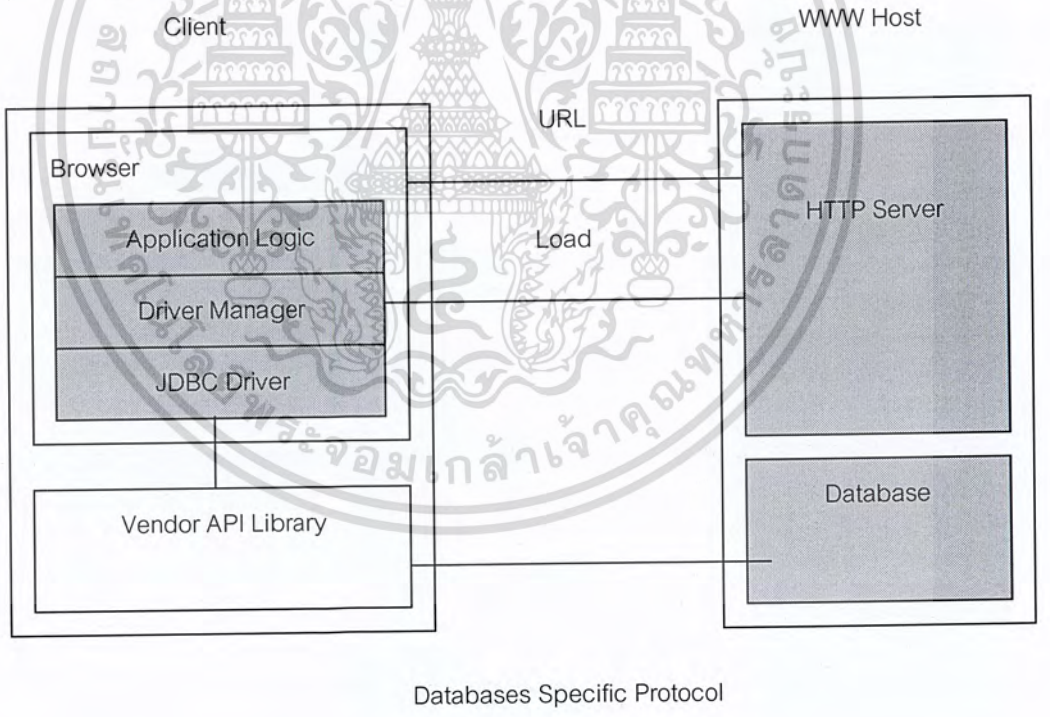
รูปที่ 5-5 แสดงการติดต่อ JDBC โดยรูปแบบที่ 1 JDBC-ODBC Bridge

5.4.2 Native-API , party Java driver

Native-API , partly Java driver เป็น two-tier ที่ซึ่ง JDBC driver ต้องการ Vendor Library ในการแปลง JDBC function ไปยังคุณลักษณะของการใช้ภาษาในการ Query เรียกใช้ DBMS ยกตัวอย่างเช่น library สำหรับ Sybase คือ dblib , สำหรับ Oracle คือ ocilib ซึ่ง driver เหล่านี้จะถูกใช้เพื่อเขียนในการรวมของภาษา Java และ C/C++ การติดต่อโดย JDBC ประเภทนี้จะเหมือนกับ JDBC/ODBC Bridge คือต้องการ code ซึ่งก็คือ vendor library เพื่อที่จะใช้ในการติดตั้งบนเครื่องไคล์เอนท์ ดังนั้นต้องมีการบำรุงรักษา (maintenance) ปัญหาเหมือนกับ bridge อย่างไรก็ตาม Driver ประเภทนี้จะเร็วกว่าประเภทแรก เพราะว่ามี Layer พิเศษของการแปลงเป็น ODBC ถูกกำจัดออกไป

JDBC ⟷ Native vlient-API ⟷ database

ข้อดีของไดร์ฟเวอร์ชนิดนี้คือ สามารถใช้ Native-API Driver ติดต่อกับฐานข้อมูลนั้นๆ โดยตรงโดยผ่านโปรโตคอลเดิมที่ใช้อยู่ก่อนแล้ว ทำให้เหมาะสมกับการเชื่อมต่อกับฐานข้อมูลแบบ Two-Tier Client/Server แต่ก็มีข้อเสียที่คล้ายคลึงกับ JDBC-ODBC Bridge คือต้องติดตั้งไดร์ฟเวอร์ (DLL) และโปรโตคอลไดร์ฟเวอร์เช่น SQL-NET , INET , Open Client , ฯลฯ บนเครื่องคอมพิวเตอร์นั้นๆ ที่สำคัญส่วนหนึ่งของ Native-API Driver ถูกเขียนขึ้นจากภาษา C++(Partly-Java) จึงไม่สามารถดาวน์โหลดผ่านเครือข่ายอินเทอร์เน็ตได้และไม่สามารถเชื่อมต่อฐานข้อมูลข้ามชนิดกันได้



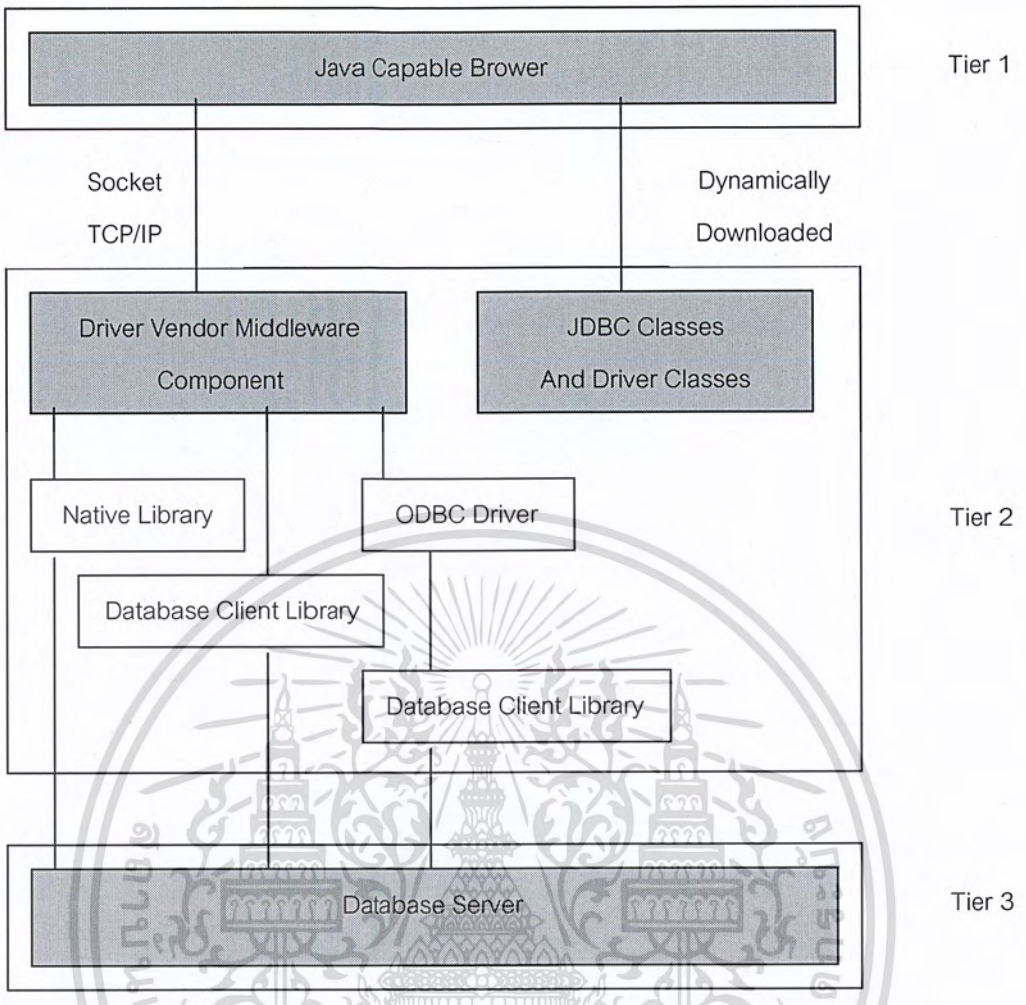
รูปที่ 5-6 แสดงการติดต่อ JDBC โดยรูปแบบที่ 2 Native-API, partly Java driver

5.4.3 Network-protocol, all- Java driver

Network-protocol, all-Java ประเภทนี้ driver จะทำการแปลงการเรียกใช้ JDBC ให้อยู่ในรูปของเน็ตเวิร์คโปรโตคอลร่วม (DBMS-independent network Protocol) ซึ่งหลังจากนั้นจะถูกแปลงให้อยู่ในรูปเฉพาะของแต่ละฐานข้อมูล (database-specific API) บนเซิร์ฟเวอร์นั้นๆ Middle-tier server รูปแบบการเชื่อมต่อฐานข้อมูลจะเป็นลักษณะ three tier :JDBC driver (ปรกติ 200 KB หรือน้อยหว่านั้น) ทำการ execute บนไคล์เอนท์ และถูกใช้เครื่องมือทาง logic ส่งคำสั่ง SQL ไปยัง Network และ ไปยัง JDBC เซอร์เวอร์ , เมื่อได้รับข้อมูลกลับมาจากเซอร์เวอร์ และ ถูกจัดการ connection Driver ประเภทนี้อนุญาตสำหรับการใช้ Just-in-time บนเครื่องไคล์เอนท์

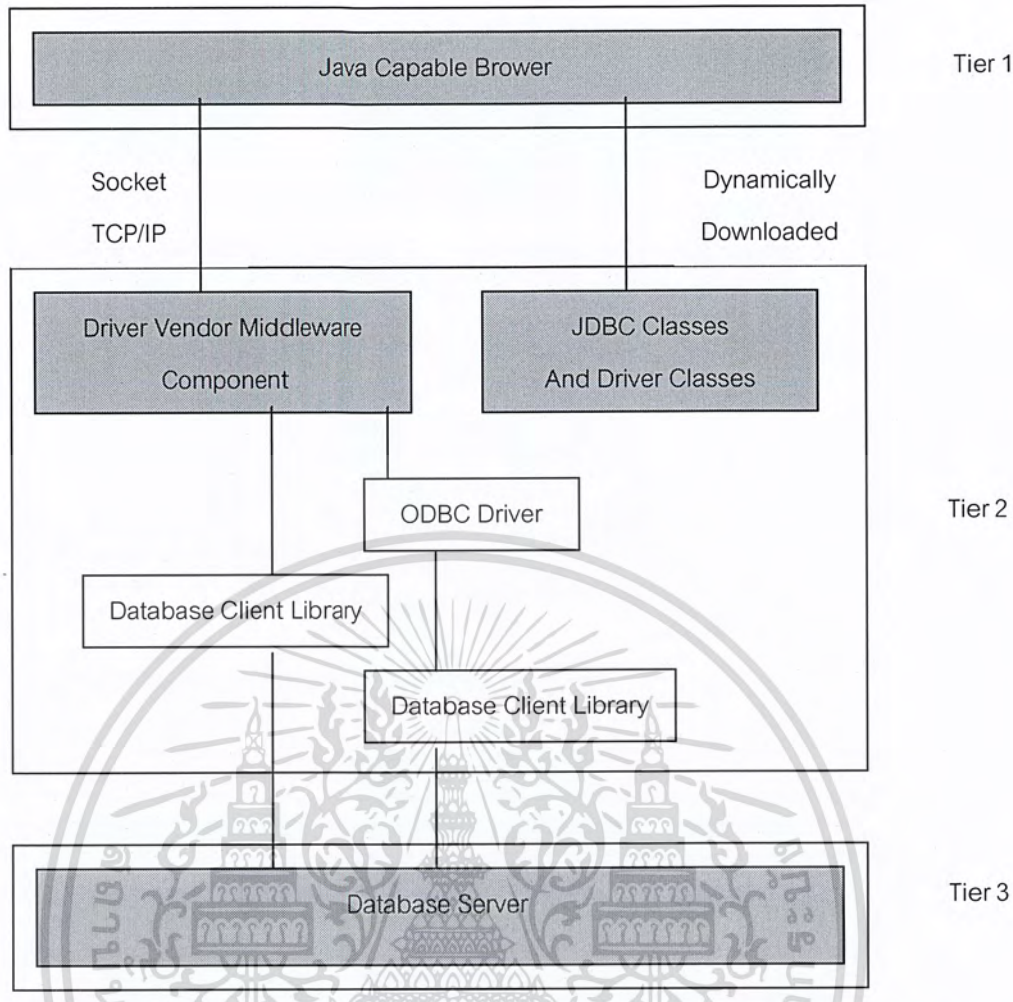
JDBC เซอร์เวอร์จะจัดการ connection ที่มีมาไปยัง database ซึ่งดีเหมือน exception และ status events resulting จาก SQL excution Packages ของ data สำหรับการติดต่อข้าม network ไปยัง JDBC ไคล์เอนท์

Middleware server component สามารถอธิบายเหมือน native component หรือถูกเขียนด้วย Java การอธิบาย Native ในการ connect ไปยัง Database server โดยใช้ library ของ vendor หรือ ODBC ซึ่ง Symantec's db Anywhere และ Intersolv's SequeLink ทั้งคู่ถูกจัดให้อยู่ในประเภทนี้ , SequeLink ไม่ต้องการ database client libraries ที่จะใช้ในการติดตั้งลงบน เซอร์เวอร์ แต่จะถูกใช้ Libraries ของตนเอง เซอร์เวอร์ต้องมีการติดตั้ง Configuration สำหรับ database ที่จะใช้ติดต่อเช่น ติดตั้งหมายเลขของ Port, ตัวแปรของ database specific environment สำหรับ instance, DSQuery กับ Sybase, Database specific parameters เช่น logging , translation และ parameter อื่นๆ ที่ซึ่งเซอร์เวอร์ อาจจะต้องการถ้า Middleware server ถูกเขียนด้วยภาษา Java สามารถใช้ JDBC - compliant driver ในการติดต่อกับ DBMS โดย Database Proprietary Protocol ของ vendor ไม่ต้องการที่จะ configure ที่เซอร์เวอร์ สำหรับ database



รูปที่ 5-7 แสดงการติดต่อ JDBC โดยรูปแบบที่ 3 Network-protocol, all-Java driver ยกตัวอย่างของ dbAnywhere, SequelLink

สำหรับคุณสมบัติเหมือนการติดต่อกับ Caching, จำนวนของ listener threads, และ อื่นๆ, ความต้องการค่าคงที่ต้องมีการติดตั้ง Weblogic's T3 server ถูกจัดให้อยู่ในประเภทนี้ JDBC T3 driver จะถูก executing บนไคล์เอนท์โดยจะนำชื่อของ driver ใช้ในการติดต่อสื่อสารกับ DBMS ซึ่งดีเหมือนกับคุณสมบัติที่ต้องการโดย driver เช่น username, password, database name , และ อื่นๆ ดังนั้นการส่งข้อมูลไปยัง T3 server จะ Load ผ่านไปยัง driver เช่น driver class และใช้ในการติดต่อกับ database , Server ยังต้องมีการ load ด้วย database vendor libraries และ/หรือ ODBC driver ในกรณีนี้การส่งค่า driver class name ในประเภทที่1 หรือ ประเภทที่ 2



รูปที่ 5-8 แสดงการติดต่อ JDBC โดยรูปแบบที่ 3 Network-protocol , all-Java driver ยกตัวอย่างของ WebLogic

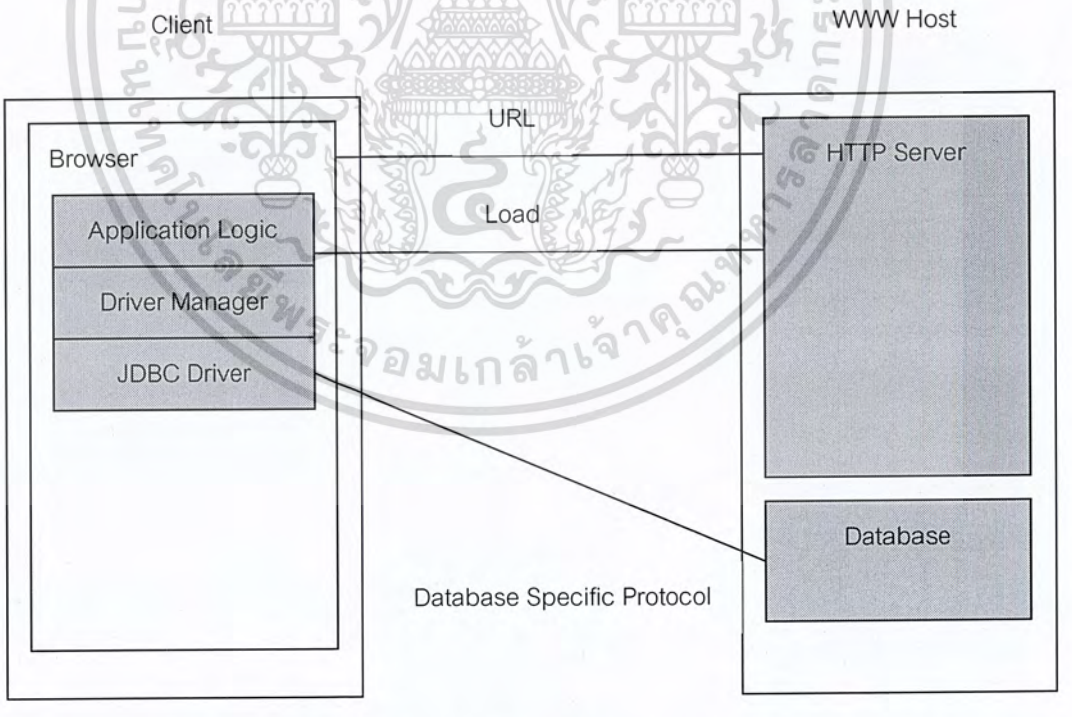
Driver ประเภทนี้ถือว่าเหมาะสมมากที่สุดสำหรับ Internet/intranet – based, multi – user data – intensive application, ที่ซึ่งมีจำนวนของ concurrent data operations มากๆ เพราะมีความยืดหยุ่นคล่องตัวที่สุด เพราะว่าเขียนขึ้นจากภาษาจาวาทั้งหมด ทำให้ไม่จำเป็นต้องมีไดร์ฟเวอร์ร่วมที่เขียนขึ้นจากภาษาอื่น ซึ่งต้องติดตั้งเฉพาะบนไคล์เอ็นท์นั้น และสามารถรันบนระบบใดก็ได้ที่สนับสนุนสภาพแวดล้อมเสมือนของจาวาหรือ JVM (Java Virtual Machine) การเฏิบัติงานทุกรูปแบบทั้งดาวน์โหลดข้ามเครือข่ายหรือรันบนเครื่องเดี่ยวๆ สามารถกระทำได้หมดเช่นการ queries searches และอื่นๆ และยังมีความสามารถทางด้าน scalability และประสิทธิภาพซึ่งถือเป็นส่วนประกอบหลัก เซอร์เวอร์สามารถจัดการ multiplexing management ระหว่าง database ที่หลากหลาย, สามารถจัดหา logging และ administration facilities สามารถ load balancing features, และยังสามารถรองรับ catalog และ การ Query caches Three-tier web database application ส่วนใหญ่มี security, firewalls, และ proxies

ข้อดีของ network – centric drivers คือองค์ประกอบของ เซอร์เวอร์ คือ proprietary middleware; ตัวแทนจำหน่ายแต่ละรายใช้ middleware ของตนเองสำหรับการติดต่อส่อสารข้ามผ่าน network ดังนั้น enterprise users จึงมีความสามารถที่จะจัดการได้โดย CORBA produsts และการ upgrade paths ในอนาคต

Openjdbc จาก I-Kinetics คือ Network-centric driver ที่ใช้ CORBA คือ middleware จะเชื่อม JDBC ไคล์เอนท์ และ เซอร์เวอร์ เครื่องมือชนิดนี้มีข้อดีของ CORBA's distributed computing infrastructure คือ วงการอุตสาหกรรมยอมรับเป็นมาตรฐานในการ Open ที่จะติดต่อไปยังการสื่อสารอื่นๆ Driver มาเป็น assembly ของ cooperating objects ที่ค่อนข้างจะมากกว่าหนึ่ง เซอร์เวอร์

5.4.4 Native – protocol , all – Java driver

Native – protocol , all – Java drivers จะแปลงการเรียกใช้คำสั่งของ JDBC ให้อยู่ในรูปแบบของเน็ตเวิร์คโปรโตคอลเฉพาะของฐานข้อมูลนั้นโดยตรงไป โดยคุณสมบัติของตัวแทนจำหน่าย database Driver เหล่านี้สามารถเขียนในภาษา Java และสามารถติดต่อกับ applet แบบ just – in – time เพราะว่า drivers เหล่านี้จะแปลง JDBC ตรงไปยัง native protocol โดยปราศจากการใช้ ODBC หนือ Native APIs ซึ่งสามารถตัดหามาสำหรับการติดต่อ database ที่มีประสิทธิภาพสูง Driver เหล่านี้สามารถสร้างจากตัวแทนจำหน่าย DBMS เท่านั้นเนื่องจากความจริงที่ซึ่งความรู้ของ Protocol lies กับตัวแทนจำหน่าย , การนำการติดต่อประเภทนี้มาใช้งานมีอยู่เล็กน้อยมากในปัจจุบัน แต่ในอนาคตคาดว่าจะเพิ่มจำนวนขึ้นเรื่อยๆ



รูปที่ 5-9 แสดงการติดต่อ JDBC โดยรูปแบบที่ 4 Native – protocol, all – Java driver

ข้อดีของไดร์ฟเวอร์ชนิดนี้คือไม่ต้องมีการปรับเปลี่ยนระบบฐานข้อมูลเดิมที่ใช้งานอยู่แล้วในแต่ละองค์กร และไม่มีความจำเป็นที่ต้องติดตั้งไดร์ฟเวอร์ตัวกลางเช่น ODBC บนไคล์เอนท์แต่ละเครื่อง จึงเหมาะสำหรับการพัฒนาการเชื่อมต่อฐานข้อมูลแบบ Two – Tier สำหรับเครือข่ายอินทราเน็ตภายในองค์กรไดร์ฟเวอร์ชนิดนี้มีข้อเสียคือไม่เหมาะที่จะนำไปใช้บนเครือข่ายอินเทอร์เน็ต เพราะปัญหาเรื่องความปลอดภัย การเชื่อมต่อฐานข้อมูลไม่สามารถกระทำข้ามระบบที่แตกต่างกันได้ ต้องใช้ไดร์ฟเวอร์เฉพาะสำหรับฐานข้อมูลตัวนั้นๆ

JDBC ไดร์เวอร์ชนิดที่ 3 และ 4 คือ ไดร์ฟเวอร์ที่คาดว่าจะเป็นสิ่งที่ต้องการสำหรับการเชื่อมต่อกับฐานข้อมูลในอนาคตเพราะถูกเขียนขึ้นจากภาษาจาวาทั้งหมดซึ่งมีความปลอดภัยและความคล่องมากกว่า เช่นมีการติดตั้งไดร์ฟเวอร์โดยอัตโนมัติเมื่อมีการดาวน์โหลดจาวาแอพเพร็ท สำหรับไดร์ฟเวอร์ชนิดที่ 1 และ 2 คือไดร์ฟเวอร์ เพื่อพัฒนาขึ้นมาเพื่อใช้งานร่วมกับไดร์ฟเวอร์ดั้งเดิม (ODBC) หรือเน็ตเวิร์คโปรโตคอลเฉพาะฐานข้อมูล ที่มีใช้อยู่แล้วในปัจจุบันไปพลางๆ ก่อน ซึ่งนานอนว่าไดร์ฟเวอร์ชนิดที่ 1 และ 2 ชนิดนี้มีข้อจำกัดในการใช้งานมากกว่า 2 ชนิดแรกโดยเฉพาะเรื่องความปลอดภัยและ Bridge ร่วมกับ ODBC ไดร์ฟเวอร์บน Window 95/NT ผลปรากฏว่าไม่ยุ่งยากและให้ประสิทธิภาพดีพอสมควรเหมาะสมในการเป็นเครื่องมือในการทดลองใช้ปฏิบัติงานได้จริง

5.5 ขั้นตอนพื้นฐานสำหรับการเชื่อมต่อกับฐานข้อมูล

1. เปิดการเชื่อมต่อกับฐานข้อมูล
2. ส่งคำสั่ง SQL ไปให้แก่ฐานข้อมูล
3. จัดการกับผลลัพธ์ที่ได้รับ
4. ปิดการเชื่อมต่อ

และเพื่อให้เข้าในการทำงานของ JDBC ยิ่งขึ้นตัวอย่างโปรแกรมบางส่วนต่อไปนี้จะแสดงให้เห็นถึงขั้นตอนทั้ง 4 ในการติดต่อกับฐานข้อมูลผ่าน ODBC ไดร์ฟเวอร์โดยใช้ JDBC – ODBC Bridge (ฟังก์ชัน API ต่างๆ ของ JDBC ที่มาพร้อมกับ JDK เวอร์ชัน 1.1 ขึ้นไปถูกเก็บรวบรวมไว้ในแพ็กเกจ java.sql.*)

```
//(1)เปิดการเชื่อมต่อกับฐานข้อมูล
Class.forName("sun.jdbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:myDB",login,password);
//(2) ส่งคำสั่ง SQL ไปให้แก่ฐานข้อมูล
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT a,b,c  FROM Table1");
//(3) จัดการกับผลลัพธ์ที่ได้
while(rs.next()){
int I=rs.getInt("a");
```

```
String s=rs.getString("b");
Float f=rs.getFloat("c");
}
//(4) ปิดการเชื่อมต่อ
stmt.close();
con.close();
```

จากตัวอย่างสิ่งแรกที่ทำคือเลือกชนิดของ JDBC ไดร์ฟเวอร์ (JDBC-ODBC Bridge) ระบุชื่อฐานข้อมูล (DSN : Data Source Name) รวมถึงชื่อผู้ใช้ (Login) และรหัสผ่าน (Password) จากนั้นเป็นการสร้างอ็อปเจ็คท์ stmt (คลาส Statement) สำหรับการเรียกค้นข้อมูล และจัดส่งคำสั่ง SQL ไปในรูปแบบสตริง (String) executeQuery() ผลลัพธ์ที่ได้จากเม็ทธอดนี้คือผลลัพธ์จากการประมวลผลของฐานข้อมูลซึ่งจะเก็บอยู่ในอ็อปเจ็คท์ rs (คลาส ResultSet) จากนั้นข้อมูลที่ได้รับนี้จะถูกถ่ายทอด (Mapping) ให้อยู่ในชนิดข้อมูลที่จาวารู้จัก เช่น int,String , float, ฯลฯ เพื่อนำไปใช้ประโยชน์อื่นต่อไป สุดท้ายคือการปิดการเชื่อมต่อกับฐานข้อมูล

ภาษาจาวาและ JDBC คือเครื่องมือที่จะทำให้การเชื่อมต่อสื่อสารกับระบบฐานข้อมูลเป็นไปอย่างสะดวกรวดเร็วและเป็นหนึ่งเดียว (Database-neutal Communication) อย่างไรก็ตามพื้นฐานที่จะขาดเสียไม่ได้สำหรับการสร้างระบบเชื่อมต่อฐานข้อมูลนั้นก็คือความรู้ความเข้าใจเกี่ยวกับหลักการทำงาน หลักการออกแบบของฐานข้อมูลชนิดสัมพันธ์ (Relational Database) ระบบจัดการฐานข้อมูล (DBMS : Database Management System) และภาษา SQL

จากชนิดของ JDBC Driver สามารถแสดงตัวแทนจำหน่ายและฐานข้อมูลที่สามารถปฏิบัติการแสดงดังตารางได้ดังนี้

Vendor	Type	DBMSs supported
Agave Software Design	3	Oracle, Sybase, Informix, others vis ODBC
Asgard Software	3	Unisys A series DMSII database
Borland	4	InterBase 4.0
Caribou Lake Software	3	Ingres
Connect Software	4	Sybase, MS SQL Server
DataRamp	3	Several sozen through ODBC drivers.*
Ensodex,Inc.	3	Several through ODBC.*
IBM	2,3	IBM DB2 Bersion 2
IBM	4	DB2 DOR OS/400

Vendor	Type	DBMSs supported
GWE Technologies	4	mysql
IDS Software	3	Oracle, Sybase, MS SQL Server, MS Access, Informix,Watsom, and others vis ODBC
I-Kineties, Inc.	3	Oracle, Informix, Sybase, and others via ODBC
Imaginary	4	mSQL
InterSoft	3	Essentia
Intersolv	3	DB2,Ingres,Informix,Oracle,Microsoft SQL Server,Sybase 10/11
JavaSoft	1	Several dozen through ODBC drivers.*
KonaSoft, Inc.	3,4	Sybase,Oracle,Informix
NetAway	3	Oracle,Informix,Sybase,MS SQLServer , SB2 , others vis ODBC
OpenLink	3	Oracle,Informix,Sybase,MS SQL Server, CA-Ingres,Progress,Unify,PostgreSQL,Solid,and others via ODBC.
Oracle Corporation	2	Oracle
SAS Institute Inc.	3,4	SAS,and via SAS/ACCESS,Oracle,Informix, Ingres,and ADABAS
SCO	3	Informix,Oracle,Ingres,Sybase,Interbase
StormCloud Development	3	Any DBMS that is accessibke via ODBC
Sybase, Inc	3,4	Sybase SQL Server,SQL Anywhere,Sybase IQ,Replication SERVER and more than 25 enterprise and legacy datase servers vis Sybase Omni CONNECT
Symantec	3	Oracle,Sybase,MS SQL Server, MS Access, Watcom and others vis ODBC
Trifox, Inc.	3	ADABAS,DB2,Informix,Ingres,Oracle,Rdb,SQL Server,Sybase,and legacy systems via GENESIS.
Visigenic	3	Several dozen through ODBC drivers.*
WebLogic	2	Oracle,Sybase,MS SQLServer
WebLogic	3	Several dozen through ODBC drivers.*
XDB Systems, Inc.	1,3	Many databases through ODBC.*
Yard Software GmbH	4	YARD-SQL Database

ตารางที่ 5-1 แสดงตารางเปรียบเทียบ JDBC Driver

- รายการของ ODBC drivers สำหรับการใช้การเชื่อมต่อด้วย JDBC-ODBC Bridge หรือการเชื่อมต่อด้วยชนิดที่ 3 drivers

  สำหรับตารางแสดงให้เห็นถึงผู้แทนจำหน่าย DBMS หรือ ODBC vendors

5.6 การ Connect DBMS โดยใช้ Servlet

จะมี 7 steps สำหรับการ connect database ของ java

Step 1. ต้อง import JDBC API มาใช้โดยจะใช้ package ของ java.sql

Step 2. ต้อง Load Driver มาใช้ (ขึ้นอยู่กับว่าเราใช้ DBMS อะไร ที่สำคัญต้อง support JDBC)

Step 3. สร้าง connection กับ DBMS ที่เราต้องการ connect ด้วย

Step 4. สร้าง Statement ขึ้นมา

Step 5. ดึงข้อมูลจาก DBMS มาใส่ Result Set

Step 6. นำข้อมูลจาก ResultSet (ที่เราดึงจาก DBMS มาใส่ไว้) มาใช้

Step 7. ต้องมีการปิด connection กับ statement ทุกๆ ครั้ง เพื่อลดการ lag ของ memory

ต่อไปจะมาดูการ connect database ของ Servlets โดยเป็นการ connect ของ JDBC - JDBC

```
//Step 1. มีการ import api ที่เราจะใช้
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;
public class Data extends HttpServlet {

//Step 2.
Connection c;
Statement   stmt;
int  count = 0;
public void init(ServletConfig config) throws ServletException {
super.init(config);
try{

//Step 2.
Class.forName("com.informix.jdbc.IfxDriver");
String str = "";
```

```
// Step 3.
str="jdbc:informix-sqli://168.120.30.2:2222/ "
      +" aiesec:informixserver=netnes;"
      +" user=.....;password=.....";
c = DriverManager.getConnection(str);
}
catch(Exception e) { }
}
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
ResultSet rs;
res.setContentType("text/html");
PrintWriter out = res.getWriter();
try {
out.print("<html>");
out.print("<body>");

//Step 4.
stmt = c.createStatement();

// Step 5.
sql = "Select * From question";
rs = stmt.executeQuery(sql);

// Step 6.
while (rs.next()) {
out.print(rs.getString("name")+"<br>");
out.print(rs.getString("address")+"<br>");
out.print(rs.getInt("telephone")+"<br>");
}
out.print("</body>");
out.print("</html>");
} catch(Exception e) { out.print(e) ); }
```

```
out.print("</body></html>");
}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
doGet(request, response);
}
public void destroy() {
try {
if ( c!=null){

// Step 7.
stmt.close();
c.close();
}

} catch (Exception ex) { }
}
}
```

บทที่ 6

Java Servlet

6.1 Introduction to Java Servlet (in depth)

แม้ว่าโลกของ Internet จะเพิ่งเกิดขึ้นเพียงไม่กี่ปีก็ตาม เทคโนโลยีที่ใช้กับ Internet กลับมีการเปลี่ยนแปลงไปอย่างรวดเร็วมาก ในสมัยแรก ๆ เพจต่าง ๆ ที่อยู่ในเวปจะเป็นลักษณะของ static page หรือเพจที่ไม่มีการเปลี่ยนแปลงเนื้อหาที่ไม่ว่าจะนานเท่าไหร่ นอกเสียจากว่าผู้ดูแลเพจนั้นจะทำการอัพเดทเพจดังกล่าว เพจลักษณะนี้เป็นเพจที่นิยมใช้กันทั่วไปใน Internet สมัยแรกเพราะอินเทอร์เนทยังนิยมกันอยู่ในวงแคบ โดยกลุ่มผู้ใช้จะเป็นกลุ่มบุคคลที่อยู่ในวงการศึกษาเท่านั้น ต่อมาจากนั้นไม่นานทางผู้ผลิตBrowser ได้ทำการเพิ่มความสามารถให้กับเพจโดยอนุญาตให้เพจสามารถแทรก Script เล็ก ๆ ลงไปรวมกับส่วนที่เป็น HTML ได้ซึ่งจุดนี้ก็คือจุดเริ่มต้นของ Client Side JavaScript นั่นเอง แม้ว่าเพจจะเริ่มมีความสามารถในการโต้ตอบกับผู้ใช้โดยอิงความสามารถจาก JavaScript แล้วก็ตาม ถ้าพูดถึงในแง่ของส่วนเนื้อหาของตัวเพจจริง ๆ แล้วตัวเพจเองก็ยังคงเป็น static page อยู่เช่นเดิม เมื่อกลุ่มผู้ใช้อินเทอร์เนทเริ่มมีมากขึ้นความต้องการที่จะให้เพจสามารถทำการรับส่งข้อมูล รวมไปถึงเปลี่ยนแปลงเนื้อหาได้โดยอัตโนมัติก็เกิดขึ้น เทคโนโลยีที่เกิดขึ้นเพื่อรองรับความต้องการเหล่านี้ก็คือ Server Side Application นั่นเอง

Server Side Application ในระยะแรก ๆ มักถูกเขียนขึ้นด้วยคอนเซ็ปของ CGI (Common Gateway Interface) โดยหลักการทำงานง่าย ๆ ก็คือ Web Browser จะทำการส่งเดต้าที่เกิดจาก Action ของ User เช่น การคลิกลิงค์หรือการกรอกแบบสอบถามไปยัง Web Server โดยแทนที่ Web Server จะทำการส่งเพจที่เป็น static page กลับมา Web Server จะทำการ forward เดต้าดังกล่าวไปยังโปรแกรมซึ่งถูกจัดไว้ โปรแกรมดังกล่าวจะทำการประมวลผลเดต้าที่ได้แล้วจะส่งผลกลับไปยัง Web Server ซึ่งทาง Web Server ก็จะส่งผลที่ได้นี้กลับไปยัง Web Browser อีกทีหนึ่ง หลาย ๆ คนคงเคยเห็น Server Side Application ที่เป็นผลิตผลของ CGI ในรุ่นแรก ๆ ที่ยังหลงเหลืออยู่ในปัจจุบันนี้ ยกตัวอย่างเช่น Counter, Image Map, Guessbook, SendMail เป็นต้น จริง ๆ แล้วตัว CGI เองสามารถเขียนด้วยภาษาอะไรก็ได้ แต่ที่นิยมมากที่สุดเห็นจะเป็น C และ Perl อาจจะเป็นเพราะว่า CGI เป็นสิ่งที่มีมากับ Internet ตั้งแต่ช่วงแรก ๆ ดังนั้นไม่ว่าจะเป็น Web Server ไหนก็ตาม Server Side Application พื้นฐานที่ทาง Web Server เหล่านั้นจะต้องสนันสนุนก็คือ CGI ซึ่งจุดนี้เองที่เป็นจุดเด่นทำให้ CGI เป็นที่นิยมใช้กันอย่างกว้างขวางจนกระทั่งปัจจุบันนี้

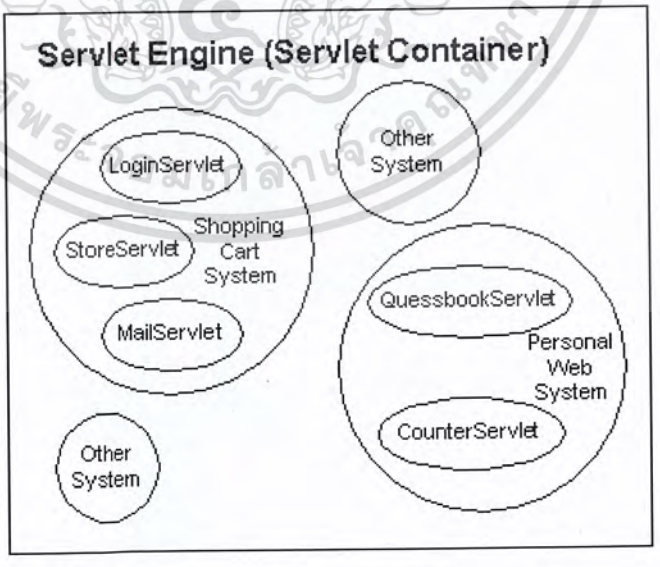
6.2 Servlet คืออะไร

Servlet เป็น Server Side Application แบบหนึ่งซึ่งอ้างอิงคอนเซ็ปมาจาก CGI ข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาวานั่นเอง จาวาเป็นภาษาที่ใช้คอนเซ็ปของ Object Oriented ในการเขียน หลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมสำหรับโปรเจคใหญ่ ๆ จะทราบดีว่า Object Oriented สามารถลดความซับซ้อนของโครงสร้างโปรแกรมรวมไปถึงการอำนวยความสะดวกในการ reuse ส่วนของโปรแกรมที่เขียนไว้แล้วเพียงไร นอกจากนี้จาวายังเป็นภาษาที่เป็นลักษณะแบบ

platform independent ซึ่งจะช่วยให้เราสามารถที่จะทำการพัฒนาระบบโดยใช้ Environment อะไรก็ได้ซึ่งโดยทั่วไปมักนิยมใช้ Window Environment โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบน Unix Environment เพื่อเพิ่มความเสถียรภาพของโปรแกรมแทน นอกจากนี้ Servlet ยังมีความเร็วที่สูงกว่า CGI เพราะ Servlet ใช้หลักการของ thread โดยจะทำการสร้าง 1 thread ต่อหนึ่ง request ที่มาจาก client ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request* ซึ่งจะทำให้เปลืองทรัพยากรมากกว่าและ process ในการรันก็จะช้ากว่าด้วย ท้ายที่สุดจุดเด่นที่สำคัญของ Servlet ก็คือ API (Application Programming Interface) โดยระบบที่ทำการพัฒนาโดยใช้คอนเซ็ปของ Servlet จะสามารถเรียกใช้ API ที่ทางจาวามีมาให้ (javax.servlet.*, javax.servlet.http.*) ซึ่งจะช่วยทำให้การพัฒนาระบบดังกล่าวง่ายและเร็วยิ่งขึ้น

6.3 Servlet Engine

ในการรันระบบที่เขียนขึ้นโดยใช้หลักการของ Servlet เราจะต้องนำระบบดังกล่าวมาบรรจุอยู่ในสิ่ง ๆ หนึ่งที่เรียกว่า Servlet Engine ให้นึกถึง Servlet Engine คล้าย ๆ กับกล่อง ๆ หนึ่งที่ใส่ลูกปิงปองไว้หลายลูก โดยลูกปิงปองแต่ละลูกก็คือระบบ ๆ หนึ่งนั่นเอง อาจสงสัยว่าทำไมถึงใช้คำว่าระบบ โดยทั่วไป Server Side Application หนึ่ง ๆ ที่ถูกเขียนขึ้นโดยใช้ Servlet API จะถูกเรียกว่า Servlet ในหนึ่งระบบอาจประกอบด้วย Servlet หลายอัน ยกตัวอย่างเช่น ระบบที่เกี่ยวกับ Shopping Cart อาจจะประกอบด้วย Servlet ที่ทำหน้าที่ในการเช็คล็อกอิน, Servlet ที่ทำหน้าที่ในการเก็บข้อมูลสินค้า, Servlet ที่ทำหน้าที่ในการส่งเมล์กลับไปยังลูกค้าเพื่อบอกว่าได้ทำการส่งของไปให้แล้ว เป็นต้น ดังนั้นถ้ามองโดยรวมแล้ว Servlet Engine ก็คือที่รวมของระบบตั้งแต่หนึ่งระบบถึงหลายระบบ โดยแต่ละระบบจะประกอบด้วย Servlet หนึ่งอันหรือมากกว่า ดังรูป ระบบ ในที่นี้อาจจะหมายถึง Zone (Apache JServ) หรือ Web Application (Tomcat) ก็ได้



รูปที่ 6-1 แสดง Servlet Engine and its Servlets

Servlet Engine เป็นเพียงกล่อง ๆ หนึ่งที่ใช้บรรจุและรันกลุ่มของ Servlet เท่านั้น ในการที่จะทำการติดต่อสื่อสารกับ Client ตัว Servlet Engine นี้จะต้องทำงานร่วมกับ Web Server ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับ Client อีกทีหนึ่ง เมื่อใดก็ตามที่มี request ส่งมาจาก Client ถ้า request นั้นเจาะจงมาที่ตัว Servlet ทาง Web Server ก็จะทำการ forward ตัว request นั้นมาให้ Servlet Engine ซึ่งทาง Servlet Engine ก็จะทำการเรียก Servlet ที่ Client ต้องการขึ้นมาทำการประมวลผล request นั้น โดยท้ายสุด Servlet จะส่งผลกลับไปให้ Servlet Engine, Servlet Engine ก็จะ forward ผลที่ได้กลับไปให้ Web Server ซึ่ง Web Server ก็จะส่งผลกลับไปให้ Client

Sevlet Engine อาจจะเป็นส่วนที่ติดมากับ Web Server อยู่แล้วยกตัวอย่างเช่น Servlet Engine ที่อยู่ใน Netscape Enterprise Server, IBM WebSphere หรืออาจจะเป็นส่วนที่เป็น Add-on ให้กับ Web Server ก็ได้เช่น Apache Jserv, Tomcat, JRun หรือแม้กระทั่งเป็นส่วนหนึ่งที่อยู่ใน Web Application เช่น BEA Weblogic เป็นต้น ทั้งนี้การเลือกใช้ Servlet Engine แต่ละชนิดก็มักขึ้นอยู่กับปัจจัยหลายอย่างเช่น ความสะดวกในการรวมระบบที่จะสร้างขึ้นมาใหม่กับระบบที่มีอยู่แล้ว, งบประมาณที่มีอยู่สำหรับโครงการหรืออาจจะรวมไปถึงทักษะและประสบการณ์ส่วนตัวของนักพัฒนาแต่ละคน

6.4 Servlet ทำงานได้อย่างไร

สมมุติว่าเรามี interface อันหนึ่งชื่อ Servlet โดย interface นี้ประกอบไปด้วยฟังก์ชันสองฟังก์ชัน ดัง code snippet ข้างล่างนี้

```
class MyServlet implements Servlet {
 public void init() {
  System.out.println("Calling MyServlet init()…");
 }
 public void service() {
  System.out.println("I'm MyServlet");
 }
}
```

เราจะสังเกตเห็นว่าคลาส MyServlet ทำการ implement ฟังก์ชั่น init() และ service() ซึ่งเป็นฟังก์ชันที่ถูก define อยู่ใน Servlet interface

Servlet Engine ใช้ประโยชน์จากคอนเซ็ปของ interface ในการโหลด Servlet ต่าง ๆ ในช่วง Runtime โดย Servlet ต่าง ๆ จะถูกโหลดและ cast กลับไปเป็นอยู่ในรูปของ interface แทน

ถ้าดูจากตัวอย่างของเราหลังจาก MyServlet ถูก cast กลับไปเป็น instance ของ Servlet interface แล้วทาง Servlet Engine ก็จะสามารถเรียกฟังก์ชันอะไรก็ได้ที่ถูก define อยู่ใน Servlet interface (ในที่นี้ก็คือ init() และ service())

ตัวอย่าง code ง่าย ๆ ที่ Servlet Engine ใช้ในการโหลด Servlet ในช่วง Runtime อาจจะเป็นอย่างข้างล่างนี้

```
public class SimpleServletEnging {
 public static void main(String args[])  throws Exception {
  if (args.length <= 0) {
   System.out.println("Usage: java SimpleServletEngine javaClassName");
   System.exit(0);
  }
  System.err.println("loading class " + args[0] + "…");
  Class cls = Class.forName (args[0]);
  Servlet servlet = (Servlet) cls.newInstance();  //casting
  servlet.init();
  servlet.service();
 }
}
```

ตัว Servlet ที่ Servlet Engine โหลดจะถูกอ้างอิงมาจาก argument ที่ชื่อ javaClassName โดย arguement นี้จะถูกส่งผ่านไปให้ Class.forName() เพื่อเรียก Class object ที่จะใช้สร้าง Servlet instance ขึ้นมาโดยใช้คำสั่ง Class.newInstance() ซึ่งหลังจากนั้น Servlet instance ที่ได้จะถูก cast กลับให้กลายเป็น instance ของ Servlet interface เพื่อ Servlet Engine จะได้ใช้ในการเรียกฟังก์ชั่น init() และ service()

อาจพูดง่าย ๆ ว่าหลักการทำงานของ Servlet Engine ก็คือการโหลดคลาสใดคลาสหนึ่งขึ้นมาโดยคลาสนั้นจะต้อง implement ตัว interface ที่ชื่อ Servlet ซึ่งทาง Servlet Engine ทราบอยู่แล้วว่ามีฟังก์ชันอะไรประกอบอยู่ใน Servlet interface บ้างโดยถ้า Service Engine ทำการ cast คลาสที่ implement Servlet interface (ในที่นี้ก็คือ MyServlet) กลับไปเป็น instace ของ Servlet interface แล้วตัว Servlet Engine ก็จะสามารถเรียกฟังก์ชันที่ชื่อ init() และ service() ที่ define อยู่ใน Servlet interface จากคลาสดังกล่าวได้

ในกรณีที่เราไม่ต้องการ implement ฟังก์ชันทุกฟังก์ชันที่ถูก define อยู่ใน Servlet interface เราก็อาจจะสร้างคลาสที่เป็น abstract ขึ้นมาคลาสหนึ่งก่อนโดยคลาสนี้จะทำการ implement ฟังก์ชันบางฟังก์ชันที่อยู่ใน Servlet interface แล้วให้ subclass ของ abstract คลาสนี้ทำการ implement ฟังก์ชันที่เหลือ ยกตัวอย่างเช่น

```
abstract  class HttpServlet implements Servlet
{
 public void init() {
  System.out.println("Calling HttpServlet init()…");
 }
 //not implement yet
```

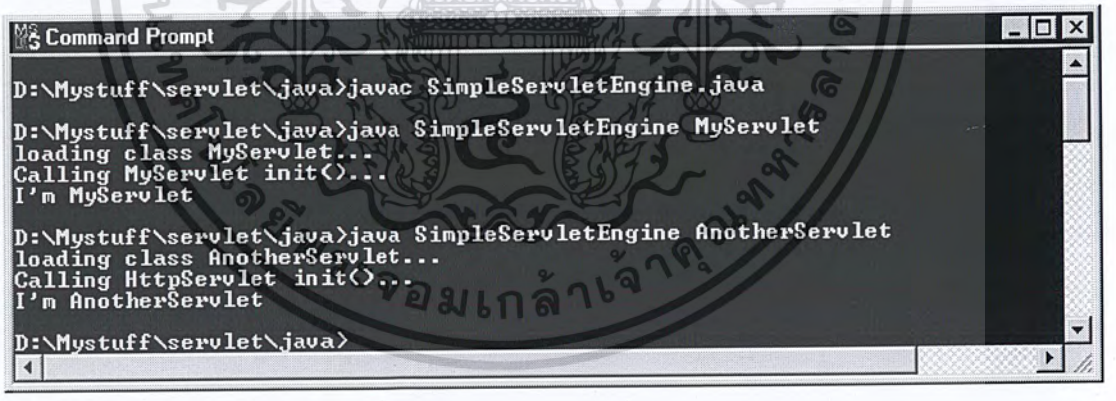
```
 public abstract void service();
}
```

ถ้าเราต้องการสร้าง Servlet ขึ้นมาโดยจะ implement แค่ฟังก์ชัน service() อย่างเดียว เราก็เพียง subclass คลาส HttpServlet เท่านั้นโดยคลาส Servlet ของเราอาจจะเป็นดังตัวอย่างข้างล่างนี้

ถ้าเราต้องการสร้าง Servlet ขึ้นมาโดยจะ implement แค่ฟังก์ชัน service() อย่างเดียว เราก็เพียง subclass คลาส HttpServlet เท่านั้นโดยคลาส Servlet ของเราอาจจะเป็นดังตัวอย่างข้างล่างนี้

```
class AnotherServlet extends HttpServlet {
 public void service() {
  System.out.println("I'm AnotherServlet");
 }
}
```

เราจะสังเกตเห็นว่าคลาส AnotherServlet จะไม่ต้องทำการ implement ฟังก์ชัน init() เพราะตัว parent คลาสของมันซึ่งก็คือ HttpServlet ได้ทำการ implement ไปแล้ว ในการโหลดคลาส AnotherServlet เข้าไปยัง Servlet Engine ก็ยังใช้หลักการเดียวกันกับคลาส MyServlet ข้างต้นคือโหลดคลาส AnotherServletแล้ว cast กลับให้กลายเป็น instance ของ Servlet interface ซึ่งผลจากการรันโปรแกรม SimpleServletEngine โดยโหลด Servlet (หรือคลาส)ที่ชื่อ MyServlet และ AnotherServlet ก็จะออกมาเป็นดังรูป



รูปที่ 6-2 แสดงการโหลด Servlet ต่าง ๆ ของ SimpleServletEngine

ในการใช้งานจริงเมื่อไรก็ตามที่เราต้องการติดตั้ง Servlet เข้าไปยัง Servlet Engine เราจะต้องทำการใส่รายละเอียดต่าง ๆ ของ Servlet เช่น ชื่อคลาส, parameters ต่าง ๆ ลงไปยัง properties file ของ Servlet Engine (อาจอยู่ในรูปของ xxx.properties ไฟล์หรือ xxx.xml ไฟล์) เพื่อที่จะบอก Servlet Engine ให้รับรู้และโหลด Servlet ของเราได้อย่างถูกต้อง

6.5 Interface javax.servlet.Servlet

หัวใจของ Servlet จริง ๆ อยู่ที่ interface ที่ชื่อ javax.servlet.Servlet* โดยทุก Servlet ที่ถูกเขียนขึ้นจะต้องทำการ implement ตัว interface นี้ไม่ทางตรงก็ทางอ้อม (ทางตรงก็คือการ implement ตัว interface นี้เลย ส่วนทางอ้อมก็คือการให้ Servlet ทำการ subclass คลาสบางคลาสที่ได้ทำการ implement ตัว interface นี้ไว้แล้ว) เหตุผลว่าทำไมเราต้อง implement ตัว interface นี้เพราะว่าเมื่อไรก็ตามที่มี request จาก client เข้ามายัง Servlet Engine ตัว Servlet Engine จะทำการหา Servlet ที่ request ดังกล่าวอ้างถึง หลังจากนั้น Servlet Engine จะทำการเรียกฟังก์ชันต่าง ๆ ที่อยู่ใน Servlet เพื่อทำการประมวลผล request ของ client โดยฟังก์ชันที่ Servlet Engine จะทำการเรียกก็คือฟังก์ชันที่ Servlet ได้ทำการ implement ซึ่งเป็นฟังก์ชันที่ถูก define อยู่ใน javax.servlet.Servlet interface นั่นเอง หลายคนอาจจะถามว่าทำไม Servlet Engine ถึงเรียกฟังก์ชันที่ถูก define อยู่ใน interface นี้ เหตุผลง่าย ๆ ก็คือ Servlet เป็นส่วนที่ถูกโหลดเข้าไปใน Servlet Engine ในช่วง Runtime ตัว Servlet Engine เองไม่สามารถทราบได้ว่า Servlet ต่าง ๆ มีฟังก์ชันอะไรประกอบอยู่บ้างนอกเสียจากว่า Servlet นั้นได้ทำการ implement ฟังก์ชันที่เป็นมาตรฐานที่ Servlet Engine รับรู้ซึ่งนี่ก็คือเหตุผลว่าทำไมทุก Servlet จะต้องทำการ implement ตัว javax.servlet.Servlet interface อย่างไรก็ตามเราสามารถให้ Servlet Engine เรียกฟังก์ชันอื่น ๆ ที่อยู่ใน Servlet ได้ซึ่งวิธีการก็คือการใส่ฟังก์ชันดังกล่าวเข้าไปในส่วน implementation ของฟังก์ชันต่าง ๆ ที่ถูก define อยู่ใน javax.servlet.Servlet interface นั่นเอง

สำหรับคนที่ยังใหม่กับจาว่าอาจจะสงสัยว่า javax คืออะไร? javax คือ standard extention package ของ Sun ที่อยู่นอกเหนือจาก core API ที่อยู่ใน JDK ตัว Servlet ที่เราพูดถึงกันอยู่นี้ก็เป็นหนึ่งใน standard extension ของ Sun เช่นเดียวกัน ดังนั้น API ทุกอย่างที่เกี่ยวกับ Servlet ก็จะเป็น javax.servlet.XXX.YYY.ZZZ

ดังนั้นหลักการง่าย ๆ ในการสร้าง Servlet ก็คือการสร้างคลาสขึ้นมาคลาสหนึ่งโดยคลาสนั้นจะต้องทำการ implement ตัว interface ที่ชื่อ javax.servlet.Servlet เพียงเท่านี้เราก็ได้ Servlet เป็นของตัวเองแล้ว อย่างที่กล่าวมาข้างต้นว่าในการเขียน Servlet เราอาจจะทำการ implement ตัว javax.servlet.Servlet interface ไม่ทางตรงก็ทางอ้อม เพื่อที่จะอำนวยความสะดวกให้กับนักพัฒนา ทางจาว่าจึงได้มีการสร้างคลาสพื้นฐานที่ได้ทำการ implement ตัว javax.servlet.Servlet interface ขึ้นมาสองคลาสคือคลาส javax.servlet.GenericServlet และคลาส javax.servlet.http.HttpServlet (ซึ่งเป็น subclass ของ javax.servlet.GenericServlet อีกทีหนึ่ง) ดังนั้นสิ่งที่นักพัฒนาจะต้องทำก็คือการ subclass คลาสใดคลาสหนึ่งในสองคลาสนี้แล้ว override ฟังก์ชั่นที่ต้องการซึ่งโดยทั่วไปก็คือฟังก์ชั่นที่ชื่อ service() นั่นเอง

ถ้าเราดูความสัมพันธ์ระหว่าง javax.servlet.Servlet (interface), javax.servlet.http.GenericServlet (abstract class) และ javax.servlet.http.HttpServlet (abstract class) เราจะเห็นว่าเริ่มแรก javax.servlet.Servlet จะถูก define ด้วย 5 ฟังก์ชันพื้นฐานที่ทุก Servlet จะต้องทำการ implement ดังลิสของ API ที่ปรากฏอยู่ข้างล่างนี้

public interface Servlet

```
public void init (ServletConfig config) throws ServletException;
public ServletConfig getServletConfig();
public void servine (ServletRequest req, ServletRespinse res) throws IOException, ServletException;
public String getServletInfo();
public void destroy();
```

หลังจากนั้นคลาส javax.servlet.http.GenericServlet จะทำการ implement ฟังก์ชั่นต่าง ๆ ที่อยู่ใน javax.servlet.Servlet ซึ่งนอกจากนี้คลาส GenericServlet ยังเพิ่มเติมฟังก์ชันเสริมอีกส่วนหนึ่งเพื่อช่วยให้นักพัฒนาทำการพัฒนา Servlet ได้อย่างรวดเร็วและมีประสิทธิภาพมากขึ้น

```
public abstract class GenericServlet implemints Servlet
public GenericServlet();
public String getInitParameter();
public Enumeration getInitParameterNames();
public ServletConfig getServletConfig();
public ServletContext getServletContext();
public String getServletInfo();
public void init();
public void init(Servletconfig config) throws ServletException;
public void log(String message);
public void log(String message, Throwable cause);
public abstract void service(ServletRequest req, ServletResponse res) throws ServletException,
                    IOException;
public void destroy();
```

เราจะสังเกตว่าคลาส GenericServlet เป็น abstract คลาสเพราะฟังก์ชัน public void service (ServletRequest req,...) ของ javax.servlet.Servlet interface จะไม่ถูก implement โดยคลาสนี้ ท้ายที่สุดคลาส GenericServlet จะถูก subclass อีกทีหนึ่งโดยคลาส HttpServlet ซึ่งเป็นคลาสที่ถูกสร้างขึ้นเพื่อรองรับ Servlet ที่ถูกเขียนขึ้นสำหรับใช้ในการติดต่อสื่อสารกับ Client ที่ใช้ Http โปรโตคอล ดังนั้นในการเขียน Servlet ของเรา ๆ ก็เพียงทำการ subclass คลาส HttpServlet แล้วทำการ override ฟังก์ชันที่เราต้องการเท่านั้น

```
public abstract class HttpServlet extends GenericServlet implements Serializable
```

```
public HttpServlet();
protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void dePost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void doOptions(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void doTrace(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
protected void service(HttpServletRequest req, HttpServletResponse res) throws ServletException,
                    IOException;
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;
protected long getLastModefed(HttpServletRequest req);
```

คลาส HttpServlet ถูกกำหนดให้เป็น abstract คลาสไม่ใช่เพราะว่ามีบางฟังก์ชันในตัวมันที่ไม่ได้มีการ implement แต่เพราะว่าผู้ที่เขียนคลาสนี้ขึ้นมาไม่ต้องการให้เราทำการเรียกใช้คลาสนี้โดยตรงแต่ต้องการให้เราสร้าง Servlet ขึ้นมาจากการ subclass คลาสนี้อีกทีหนึ่ง

6.6 Hello World

ในการเรียนรู้ภาษาอะไรก็ตาม โปรแกรมแรกที่เรามักจะเขียนก็คือโปรแกรมที่พิมพ์คำว่า Hello World ออกมาดังนั้นในการเขียน Servlet เราก็จะทำเช่นเดียวกัน โปรแกรม Hello World ที่เขียนแบบ Servlet อาจจะเป็นอย่างข้างล่างนี้ (Hello World.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HeddlWorld extends GenericServlet
{
 public void service (ServletRequest request, ServletResponse response) throws IOException,
                    ServletException
```

```
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("<head>");
    out.println("<title>Hello World!</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Hello world!</h1>");
    out.println("</body>");
    out.println("<h1>Hello World!</h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

คลาส Hello World เป็นตัวอย่างพื้นฐานที่อธิบายหลักการทำงานของ Servlet ได้เป็นอย่างดี คลาสนี้เป็นคลาสที่ subclass คลาส `GenericServlet` โดยทำการ override ฟังก์ชัน `public void service(ServletRequest request, ServletResponse response)...` ซึ่งเป็นฟังก์ชันที่อยู่ใน Servlet interface ฟังก์ชันนี้เป็นฟังก์ชันที่สำคัญที่สุดสำหรับทุก Servlet เพราะว่าเมื่อใดก็ตามที่มี request เข้ามายัง Servlet ฟังก์ชัน service ที่อยู่ใน Servlet ดังกล่าวจะถูกเรียกโดย Servlet Engine เพื่อทำการประมวลผล request ที่เข้ามาเสมอ

GenericServlet เป็น abstract คลาสที่ทำการ implement ตัว Servlet interface โดยทิ้งฟังก์ชัน service ให้เป็น abstract (ไม่มีส่วนที่เป็น implementation) ซึ่งทางคลาสที่เป็น subclass ของ GenericServlet จะต้องมีหน้าที่ในการ implement ฟังก์ชันนี้ยกตัวอย่างเช่นคลาส HelloWorld ของเรา เป็นต้น หันกลับมาดู code ที่อยู่ใน HelloWorld กันบ้าง ในสามบรรทัดแรกเราจะเห็นว่าจะมีสาม package ที่ถูก import เข้ามาซึ่งก็คือ java.io.*, javax.servlet.* และ javax.servlet.http.* เราเรียก package java.io เข้ามาเพราะว่าเราต้องการใช้คลาส IOException และ คลาส PrintWriter ส่วน package javax.servlet และ javax.servlet.http เราจะใช้สำหรับเรียกคลาสที่เกี่ยวข้องกับ Servlet ทั้งหมด

ดู code ที่อยู่ในฟังก์ชัน service กันบ้าง ตัวฟังก์ชัน service จะมี argument อยู่ด้วยกันสองตัวคือ ServletRequest และ ServletResponse โดยเมื่อใดก็ตามที่ฟังก์ชัน service ถูกเรียกโดย Servlet Engine ตัว Servlet Engine จะทำการส่งผ่านออฟเจคมายัง argument เหล่านี้ซึ่งก็คือ ServletRequest และ ServletResponse ออฟเจคนั่นเอง ServletResponse ออฟเจคเป็นตัวที่เก็บรายละเอียดของ Request ที่ส่งมา

โดย client ทั้งหมดโดย Servlet สามารถใช้ออฟเจคนี้เพื่ออ่าน parameters ต่าง ๆ ที่อยู่ใน Request เข้ามาประมวลผลได้ หลังจากที่ Servlet ทำการประมวลผลเสร็จแล้ว Servlet ก็สามารถส่งผลที่ได้กลับออกไปให้ client ได้โดยใช้ ServletResponse ออฟเจค อย่างไรก็ตามฟังก์ชัน service ที่อยู่ใน Hello World Servlet ของเราไม่ได้มีการประมวลผลใด ๆ เพียงแต่จะส่ง HTML Stream ที่เขียนคำว่า "Hello World" กลับไปให้ Client เท่านั้น

ในการส่งผลต่าง ๆ กลับไปให้ Client ขั้นตอนแรกที่ Servlet ต้องทำคือการเซ็ต Header โดยในตัวอย่างของเรา Header ที่ถูกส่งกลับไปก็คือ Content Type (MIME Type) Header ซึ่งเป็นตัวบอก Client ว่า Stream ที่กำลังจะได้รับเป็น Stream ชนิดไหน (ให้นึกถึง MIME Type เป็นลักษณะคล้าย ๆ กับไฟล์ extension) ยกตัวอย่างเช่น text/html จะบอกว่า Stream ที่กำลังส่งออกไปเป็น HTML Stream หรือ image/gif จะบอกว่า Stream ที่ส่งออกไปเป็น Stream ของ gif image เป็นต้น ตัว Servlet สามารถทำการเซ็ต Content Type ได้โดยใช้ฟังก์ชัน setContentType() ซึ่งเป็นฟังก์ชันที่อยู่ใน ServletResponse นั่นเอง ในการส่ง Stream ต่าง ๆ ไปให้ Client ตัว Servlet สามารถส่ง Stream ออกไปได้สองแบบคือ text Stream และ byte Stream ในกรณีของเรา ๆ จะส่ง Stream ออกไปเป็นแบบ text Stream ดังนั้นคลาสที่เราจะใช้ในการเขียนลงไปที่ ServletResponse ก็คือคลาส ProntWriter นั่นเอง (สำหรับรายละเอียดของ Stream เราจะพูดถึงในบทความต่อ ๆ ไป)

ในการเรียกใช้คลาส PrintWriter สำหรับเขียนลงไปที่ ServletResponse เราก็สามารถทำได้ง่าย ๆ โดยการเรียกฟังก์ชัน ServletResponse.getWriter() ซึ่งจะ return ตัว PrintWriter instance ที่จะใช้ในการเขียน output ต่าง ๆ ลงไปยังตัว ServletResponse ที่เราใช้สำหรับเรียก instance ดังกล่าวขึ้นมา ท้ายสุด HelloWorld Servlet จะใช้ PrintWriter ทำการพิมพ์ text ต่าง ๆ ที่ใช้ในการสร้าง HTML Output Stream ที่เขียนคำว่า "Hello World" ออกไปให้ Client

6.7 การรัน Servlet

Servlet เป็นคอนเซ็ปหนึ่งในจาว่าที่เราเขียนขึ้นเพื่อใช้งานสำหรับ Server Side Application ถ้าใครเคยเขียน application โดยใช้จาว่ามาก่อนจะเห็นว่าตัว entry point ของ application จะเป็น static ฟังก์ชันที่ชื่อ main() ยกตัวอย่างเช่น

```
public class HelloWorld {
 public static void main(String args[]) {
  System.out.println("Hello World");
 }
}
```

ตัว Servlet เองจะมีลักษณะคล้าย ๆ Applet ตรงที่ว่าในการรัน ฟังก์ชันที่ใช้เป็น entry point จะไม่ใช่ฟังก์ชัน `main()` แต่จะกลายเป็นฟังก์ชันอื่นที่ถูก define ไว้ใช้สำหรับตัวมันเองโดยเฉพาะยกตัวอย่างเช่นใน Applet ฟังก์ชันที่ถูกใช้เป็น entry point ก็จะเป็นฟังก์ชันพวก `init()`, `start()` ซึ่งสำหรับ

Servlet แล้วฟังก์ชันที่ถูกใช้เป็น entry point ก็คือ `service()` ซึ่งก็คือฟังก์ชันที่ถูก define อยู่ใน `javax.servlet.Servlet` interface นั่นเอง

6.8 การรับส่งข้อมูลระหว่าง เว็บและServlet

สำหรับการใช้งานเว็บผ่านบราวเซอร์ โดยทั่วไปแล้ว ปกติในช่องที่ใช้กำหนดข้อมูล URL จะประกอบด้วยชื่อเซิร์ฟเวอร์ ชื่อ ไดเร็คทรอรี่ และ ชื่อไฟล์เว็บเพจที่ต้องการดู เช่น หากข้อมูล URL คือ http://www.javacentrix.com/tutorials/apu/index.htm นั่นหมายถึงต้องการอ่านเว็บเพจที่ชื่อ index.htm จากไดเร็กทรอรี่ tutorials/api จากเว็บเซิร์ฟเวอร์ที่ชื่อ www.javacentrix.com

สำหรับ URL นั้นนอกจาก จะใช้ชี้ตำแหน่งไฟล์ที่ต้องการเรียกดูแล้ว ยังสามารถใช้สำหรับส่งของมูลให้กับระบบเซิร์ฟเวอร์ เพื่อใช้ในการทำงาน หรือดำเนินการอย่างใดอย่างหนึ่ง ลักษณะ URL ประเภทนี้หลังจากชื่อไฟล์แล้ว จะตามด้วยเครื่องหมายคำถาม ? และข้อมูลที่ต้องการส่งไปให้เซิร์ฟเวอร์ ซึ่งเรียกโดยทั่วไปว่าการส่งข้อมูลแบบ GET โดยมีรูปแบบคือ

URL?Parameter1 = Data1&Parameter2=Data2

จากรูปแบบการส่งข้อมูลผ่าน URL โดย Parameter1 และ Parameter2 คือชื่อพารามิเตอร์ที่ถูกเซิร์ฟเวอร์อ้างถึงได้ในขณะที่ Data1 และ Data2 คือข้อมูลที่อยู่ในพารามิเตอร์แต่ละตัว หากต้องการส่งข้อมูลมากกว่า 2 ตัว ก็จะเพิ่มต่อท้ายไปได้เรื่อยๆ เช่น

URL?Parameter1=Data&Parameter2=Data2&Parameter3=Data3&Parameter4=Data4

ตัวอย่างเช่น

http://www.javacentrix.com/index.htm?Name=JAVACENTRIX&Description=ThaiJavaWebSite

http://www.javacentrix.com/index.htm คือข้อมูล URL ในขณะที่ ?Name=JAVACENTRIX &Description=ThaiJavaWebSite คือข้อมูลที่ส่งให้กับเซิร์ฟเวอร์ อธิบายได้ดังนี้คือ เป็นการเรียกใช้งานไฟล์ index.htm โดยที่มีการส่งข้อมูลไปให้กับเซิร์ฟเวอร์ 2 ตัวคือ Name ที่มีข้อมูล “JAVACENTRIX” และ Description ที่มีข้อมูล “Thai Java Web Site” เซิร์ฟเวอร์จะนำข้อมูลเหล่านี้ไปใช้ประกอบการทำงานกับไฟล์ index.htm ซึ่งในที่นี้หมายถึง Servlet นั่นเอง

หากเคยใช้เว็บที่เป็นระบบค้นหา (Search Engine) และพิจารณาในช่อง URL ก็จะสังเกตเห็นข้อมูลที่เป็นเช่นอธิบายไว้ หลักจากคลิกที่ปุ่มค้นหา แต่ข้อเสียของการส่งข้อมูลรูปแบบนี้คือ ผู้ใช้ต้องพิมพ์ชื่อพารามิเตอร์ให้ถูก และพิมพ์ข้อมูลให้ถูกรูปแบบด้วย อีกทั้งการใช้งานยังไม่เป็นความลับ ซึ่งถูกมองเห็นได้จากคนทั่วไป

นอกจากผู้ใช้เว็บจะต้องพิมพ์ข้อมูลที่ URL เองแล้ว การส่งข้อมูลแบบ **GET** นี้ยังสามารถใช้แท็ก FORM ใน HTML เพื่อสร้างรูปแบบการกรอกข้อมูลแก่ผู้ใช้ ที่เรียกว่าฟอร์ม เมื่อผู้ใช้ข้อมูลกรอกข้อมูลผ่านฟอร์ม และกดปุ่ม Submit ข้อมูลเหล่านั้นจะถูกบราวเซอร์ ประกอบเป็น URL ในรูปแบบ GET ให้อย่างอัตโนมัติ โดยที่ผู้อ่านไม่จำเป็น ต้องนั่งพิมพ์ URL เองให้เสียเวลาเลย

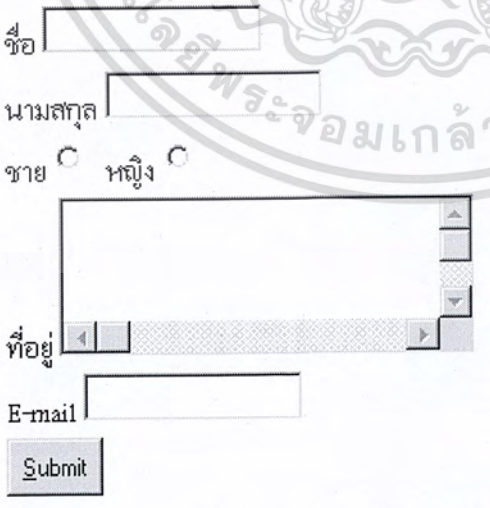
ในทำนองเดียวกัน การส่งข้อมูลจากผู้ใช้นอกจากจะส่งในรูปแบบ **GET** แล้วยังมีอีกรูปแบบหนึ่งคือรูปแบบ **POST** ซึ่งแทนที่จะส่งไปที่ตัว URL โดยตรง แต่เป็นการส่งด้วยโปรโตคอล HTTP ที่ใช้ในเว็บเป็น

ตัวจัดการ การส่งแบบนี้มีข้อดีที่ผู้อื่น ไม่สามารถล่วงรู้ได้ว่าลักษณะการส่งข้อมูลไปใช้เซิร์ฟเวอร์ ส่งข้อมูลอะไรไป และส่งโดยการใช้พารามิเตอร์ชื่ออะไร การส่งข้อมูลในแบบ POST ทำได้โดยการใช้แท็ก FORM แต่กำหนดวิธีการส่งเป็น POST นอกนั้นการสร้างฟอร์มก็ใช้การสร้างยังส่งข้อมูลแบบ GET ได้ด้วย หากสร้างฟอร์มด้วยการใช้แท็ก FORM แต่กำหนดวิธีการส่งเป็น GET

การสร้างฟอร์ม แบบ Get และ Post

```
<HTML>
<HEAD><TITLE>Test Form </TITLE><./HEAD>
<BODY>
 <FORM method=>"get"action="myPatameter">
  ชื่อ <input type="text" name="name"> <BR>
  นามสกุล <input type= "text" name="surname"> <BR>
  ชาย<input type="radeo"name="sex" value="M">
  หญิง<input type="radio"name="sex" value="F"> <BR>
  ที่อยู่ <textarea name="add"></textarea><br>
  E-mail <input type="text" name="mail"> <BR>
   <input type="submit" name="Submit" value="Submit">
 </FORM>
</BODY>
</HTML>
```

ซึ่งไฟล์ดังกล่าวให้ผลลัพท์เมื่อแสดงที่หน้าเว็บดังนี้ (myForm.htm) E-mail



รูปที่ 6-3 แสดงหน้าเว็บที่สร้างฟอร์มแบบ Get และ Post

จากคำสั่ง HTML ที่เห็นจากตัวอย่าง มีช่องสำหรับให้ผู้ใช้กรอก 3 ช่องคือเท็กซ์บล็อกที่ชื่อ name, surname และ mail (สังเกตจากคำสั่งในแท็ก FORM ของ HTML) และมีการเรียกใช้รูปแบบการส่งข้อมูลแบบ GET (สังเหตจากในแท็กฟอร์ม แอททริบิวต์ method="get") ผู้อ่านสามารถทดลองเปลี่ยนคำว่า get เป็นคำว่า post ได้หากต้องการส่งข้อมูรูปแบบ POST จำไว้ว่าการส่งแบบ GET ข้อมูลจะถูกร่วมกับ URL ในขณะทีส่งแบบ POST ข้อมูลถูกส่งด้วยรูปแบบคำสั่งโปรโตคอล HTTP แทน

6.9 สร้าง Servlet อ่านข้อมูลจากฟอร์ม

คำสั่งที่ใช้สำหรับอ่านข้อมูลจากผู้ใช้คือ getParameter ที่อยู่ในคลาส HTTPServlet…….

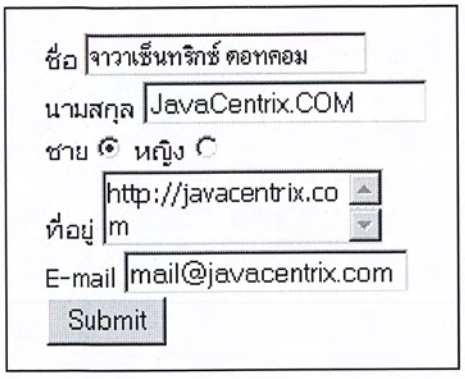
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class ShowFormData extends HttpServlet
{ public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
     response.setContentType("text/html");
     String title = "Retriving Information from Web-Form";
     PrintWriter out = response.getWriter();
     out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">\n"
+
                 "<HTML>\n" +
                 "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
                 "<BODY>\n" +
                 "<H1>" + title + "</H1>\n");
     out.print("<TABLE BGCOLOR=\"#EEEEEE\"?>\n");
     out.print("<TR><TD align=\"center\">Parameter Name</TD>\n");
     out.print("        <TD align=\"center\">Parameter Value(s)</TD></TR>\n");
     Enumeration paramNames = request.getParameterNames();
     while(paramNames.hasMoreElements()) {
        String paramName = (String) paramNames.nextElement();
        out.print("<TR><TD valign=\"top\">" + paramName + "</TD>\n");
        String[] paramValues = request.getParameterValues(paramName);
```

```
        if (paramValues.length == 1) {
          String paramValue = paramValues[0];
          if (paramValue.length() == 0)
            out.print("<TD>No Value</TD>");
          else
            out.print("<TD>" + paramValue + "</TD>");
        } else {
            out.println("<TD>");
            for(int i=0; i<paramValues.length; i++)
              out.println(paramValues[i] + "<BR>");
            out.println("</TD>");
          }
        out.print("</TR>\n");
      }
      out.println("</TABLE>\n" +
                  "</BODY></HTML>");
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      doGet(request, response);
    }
}
```

ผลการทำงาน

เมื่อทำการคอมไพล์ Servlet และย้ายไฟล์ myForm.htm ไปไว้ในไดเร็กทรอรี่เพื่อให้งานแล้ว ลองทดสอบด้วยการป้อนข้อมูลลงในช่อง ต่างๆที่อยู่ในฟอร์ม และกดปุ่ม Submit จะสังเกตเห็นว่า URL ที่เกิดขึ้นมีข้อมูลพารามิเตอร์ประกอบ พร้อมกับการอ้างไปยังชื่อ Servlet ที่มีอยู่บนเซิร์ฟเวอร์ และที่หน้าเว็บเพจ จะแสดงข้อมูลที่เป็นทั้งชื่อ และข้อมูลพารามิเตอร์ออกมาดังนี้



รูปที่ 6-4 แสดงแบบฟอร์มเมื่อปรากฏบนเว็บบราวเซอร์

Retriving Information from Web-Form

Parameter Name	Parameter Value(s)
add	http://javacextrix.com
surname	JavaCentrix.COM
sex	M
name	จาวาเซ็นทริกซ์ ดอทคอม
mail	mail@javacentrix.com

ลองทดสอบโดยการเปลี่ยนคำว่า method="get" ไปเป็น method="post" ในไฟล์ myForm.htm และทดสอบการใช้งาน จะเห็นผลลัพท์เว็บเพจ แสดงหน้าตาเหมือนกัน แต่ต่างกันที่ URL ไม่มีชื่อและข้อมูลพารามิเตอร์ประกอบอยู่เหมือนกับที่ใช้แบบ get

Servlet จะนำข้อมูลที่ได้รับเหล่านี้ไปประมวลผลกิจกรรมต่างๆ เช่นหากฟอร์มเป็นการกรอกข้อมูล การสั่งซื้อสินค้า Servlet ก็จะนำข้อมูลไปทำรายการสั่งซื้อสินค้า หรือหากฟอร์มเป็นการรับข้อมูลเพื่อค้นหาชื่อสินค้า Servlet ที่รับชื่อสินค้าเข้ามาก ก็จะนำไปค้นหาสินค้าและแสดงผลลัพท์ออกมานั่นเอง เหล่านี้คือประโยชน์ส่วนหนึ่งที่จะนำมาใช้สำหรับเทคโนโลยีจาวา ที่เรียกว่า Java Servlet

บทที่ 7
การออกแบบคลาสของการทำงาน

การออกแบบการทำงานของระบบจะใช้โมเดลในการออกแบบระบบคือ ยูสเคสไดอะแกรม , ซีเควนส์ไดอะแกรม และ คลาสไดอะแกรม เพื่อช่วยอธิบายการทำงานของระบบ

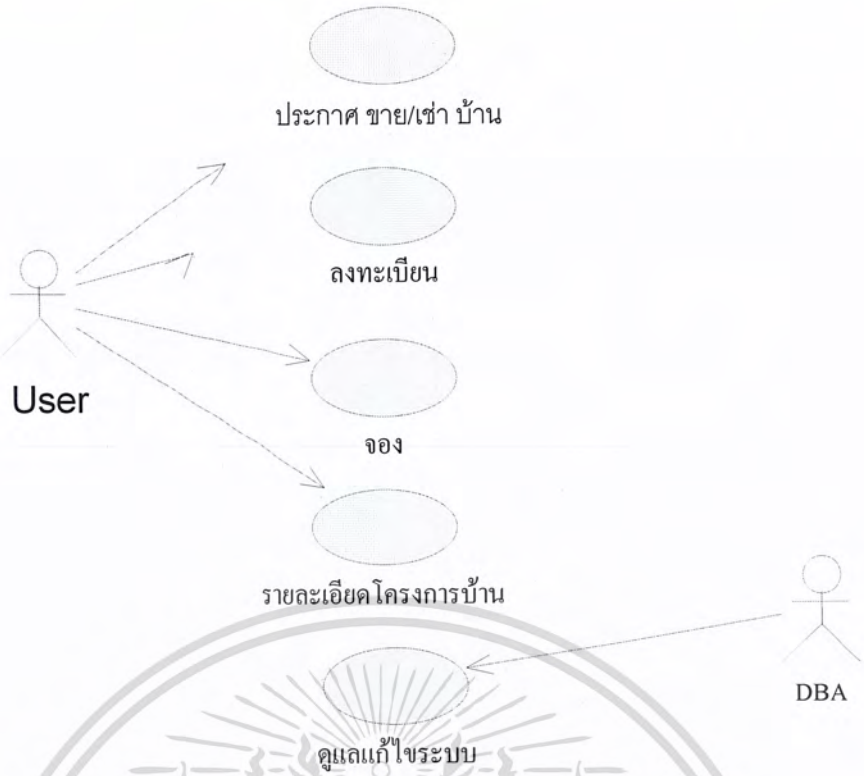
7.1 ยูสเคสไดอะแกรม

แผนภาพ ยูสเคส ที่ได้จากการวิเคราะห์นั้นจะบอกถึงการติดต่อระหว่างสิ่งแวดล้อมภายนอกกับระบบและสำหรับในการวิเคราะห์ระบบนั้นผู้พัฒนาสามารถสร้างแผนภาพยูสเคสซึ่งเป็นแผนภาพที่เพิ่มกรณีสำหรับการติดต่อระหว่างระบบกับแอคเตอร์ภายนอก แผนภาพยูสเคสนั้นจะแสดงมุมมองต่อฟังก์ชันการทำงานหลักๆ ของระบบซึ่งผู้พัฒนาระบบสามารถใช้แผนภาพ ยูสเคส นี้ในการสื่อสารกับผู้ใช้ระบบได้ด้วย ตัวแผนภาพยูสเคส อาจจะเป็นแผนที่หลักในการพัฒนาระบบเนื่องจากตัวแผนภาพยูสเคสนั้นได้รวมความต้องการ และรายละเอียดการทำงานของระบบไว้ภายในแล้ว สำหรับตัวยูสเคสเองนั้นก็สามารถที่จะแบ่งเป็น ยูสเคสย่อยๆ ลงไปได้อีก ซึ่งจะแสดงรายละเอียดลำดับในการติดต่อสื่อสารระหว่างวัตถุในระบบ เนื่องจากแผนภาพยูสเคสนั้นเป็นมุมมองต่อระบบในระดับสูงและมีการใช้ภาษาอธิบายระบบทั่วๆไป ทำให้ผู้ใช้ในทุกๆ ระดับสามารถตรวจสอบแผนภาพยูสเคสได้ง่ายขึ้นโดยสามารถใช้แผนภาพยูสเคสในการสื่อสารระหว่างทีมผู้พัฒนาเองเนื่องจากว่าสัญลักษณ์ต่างๆ ที่ใช้ในแผนภาพนี้เป็นมาตรฐานที่ทุกคนรู้จัก และแผนภาพก็มีรายละเอียดของระบบที่ครบถ้วน

การใช้ยูสเคสไดอะแกรมในการอธิบายภาพโดยรวม โดยผู้พัฒนาจะใช้ ยูสเคส เป็นเครื่องมือหลักในการพัฒนาโครงการ ทั้งนี้เนื่องจากสามารถใช้ ยูสเคส สื่อสารกันได้โดยง่าย นอกจากนั้น ยูสเคสยังรวมฟังก์ชันการทำงานหลักๆ ของระบบไว้ด้วย โดยในการออกแบบระบบการจองบ้านออนไลน์นั้น จะมีส่วนประกอบการทำงานหลักๆของระบบดังนี้

1. การลงทะเบียน
2. การดูรายละเอียดของโครงการบ้าน
3. การจอง
4. การประกาศขาย/เช่าบ้าน
5. การดูแลและแก้ไขข้อมูลของระบบ

ซึ่งการทำงานหลักๆของระบบนี้จะมีผู้ที่เกี่ยวข้องกับระบบได้แก่ User และ DBA โดย User จะเกี่ยวข้องกับระบบได้แก่ การลงทะเบียน การจอง การดูรายละเอียดของโครงการประกาศขาย/เช่าบ้าน และ DBA ได้แก่ การดูแลแก้ไขข้อมูลของระบบ ซึ่งจะแสดงยูสเคสดังในรูปที่ 7.1



รูปที่ 7-1 แสดง Use Case Diagram การทำงานของ E-HomeProject

7.2 แผนภาพแสดงลำดับขั้นตอนการทำงาน

แผนภาพแสดงลำดับขั้นตอนการทำงานจะแสดงลำดับของแมสเซจระหว่างวัตถุ ซึ่งผู้พัฒนาโครงงานจะใช้แผนภาพแสดงลำดับขั้นตอนการทำงานในการอธิบายการทำงานของระบบ โดยมีส่วนประกอบสำหรับแผนภาพแสดงลำดับขั้นตอนการทำงานโดยเส้นในแนวจังจะหมายถึงวัตถุ โดยจะมีชื่อของวัตถุเขียนอยู่ด้านบน หรือด้านล่างของเส้น ส่วนเส้นในแนวนอนนั้นแสดงแมสเซจ เส้นตรงที่แทนเมสเซจทุกเส้นจะต้องมีวัตถุต้นกำเนิดของแมสเซจส่วนจุดที่สิ้นสุดของแมสเซจก็จะเป็นวัตถุปลายทางโดยแต่ละเส้นของแมสเซจจะมีชื่อของแมสเซจกำกับอยู่โดยชื่อของแมสเซจนี้ในขั้นตอนต่อไปอาจจะมีการเพิ่มรายการของพารามิเตอร์ และค่าที่คืนกลับมา (Return Value) ด้วย สำหรับเวลานั้นจะนับจากข้างบนลงล่าง เช่น ลูกค้าจะต้องล็อกอินก่อนจึงจะสามารถจองบ้านได้ เป็นต้น โดยสเกลแกนตั้งนั้นจะบอกเพียงลำดับของแมสเซจเท่านั้นไม่สามารถใช้เป็นอัตราส่วนในการคำนวณหาเวลาระหว่างแมสเซจได้

การอธิบายการทำงานของระบบส่วนใหญ่จะเป็นแผนภาพแสดงลำดับขั้นตอนการทำงานซึ่งจะมีการออกแบบการทำงานของโครงงานการจองบ้านออนไลน์ดังต่อไปนี้คือ

1. การลงทะเบียน
2. การเข้าสู่ระบบ
3. การดูข้อมูลโครงการบ้าน
4. การค้นหาบ้าน
5. การจองบ้าน
6. การประกาศ ขาย/เช่า บ้าน

7. การเพิ่มข้อมูลโครงการ, ภาพแผนผังโครงการ , ข้อมูลบ้าน
8. การแก้ไขข้อมูลโครงการ , ภาพแผนผังโครงการ , ข้อมูลบ้าน
9. การแก้ไขและเพิ่มรายละเอียดของบ้านใหม่

7.3 แผนภาพของคลาส

แผนภาพของคลาสเป็นแผนภาพที่มีความสำคัญมากที่สุดในการวิเคราะห์และออกแบบ เนื่องจากแผนภาพของคลาสจะแสดงโครงสร้างของวัตถุและคลาสที่มีในระบบรวมทั้งแสดงความสัมพันธ์ด้วย แผนภาพของคลาสจะเป็นโครงสร้างหลักของระบบและยังใช้ในการแยกย่อยรายละเอียดด้วย อย่างไรก็ดี แผนภาพของคลาสจะแสดงเฉพาะคลาส โดยไม่ได้แสดงอินสแทนซ์ หรือวัตถุทั้งหมด และจะเน้นที่การแสดงโครงสร้างของคลาสมากกว่าความสัมพันธ์ระหว่างคลาส

และเช่นเดียวกัน การอธิบายการเขียนโปรแกรมจะอ้างอิงกับแผนภาพของคลาสเป็นหลักซึ่งจะมีการออกแบบถึงความสัมพันธ์ของคลาสต่างๆซึ่งมีแผนภาพของคลาสดังนี้คือ

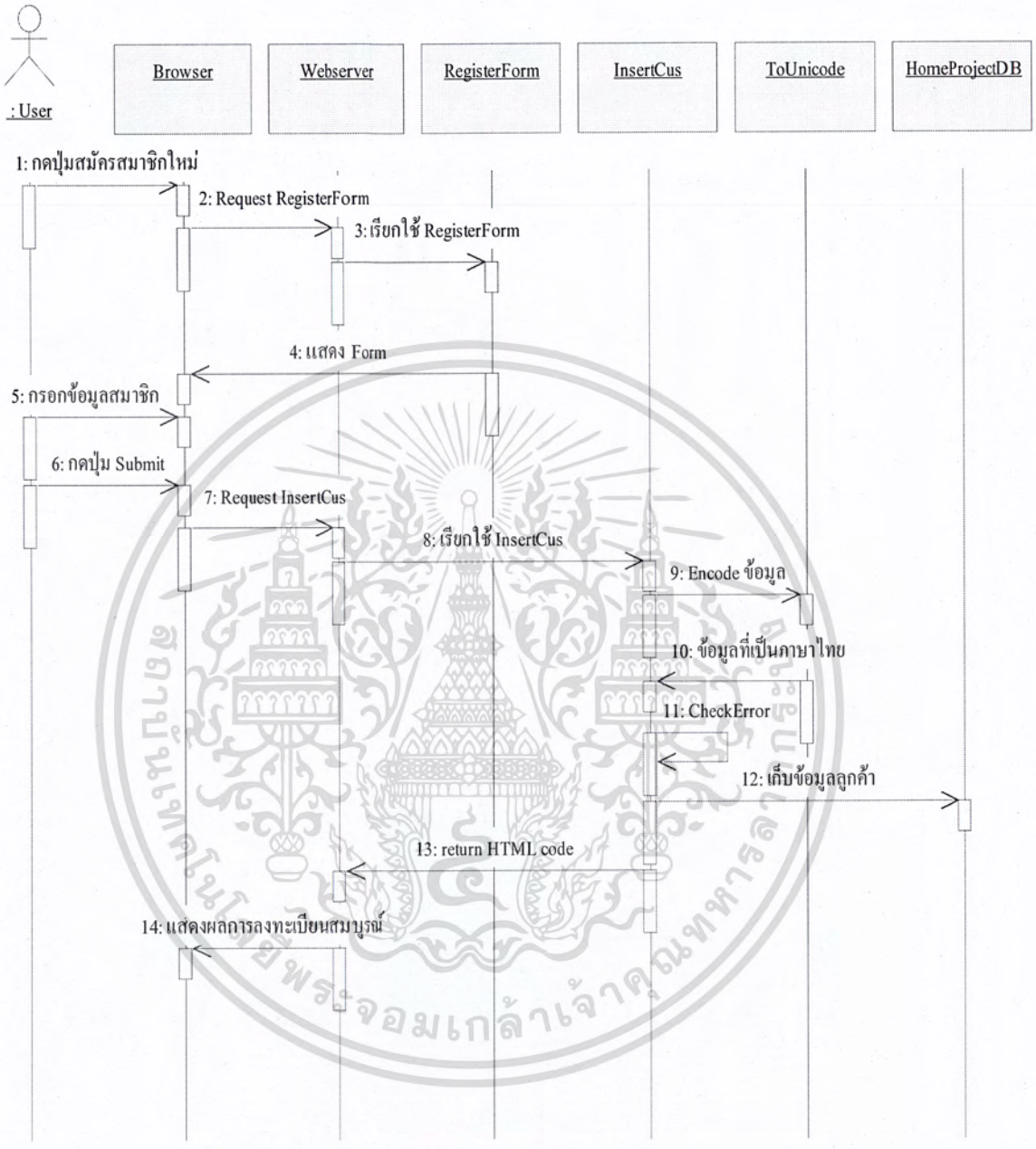
1. การลงทะเบียน
2. การเข้าสู่ระบบ
3. การดูรายละเอียดโครงการบ้าน
4. การค้นหาบ้าน
5. การจองบ้าน
6. การประกาศ ขาย/เช่า บ้าน
7. การเพิ่มข้อมูลโครงการ, ภาพแผนผังโครงการ , ข้อมูลบ้าน
8. การแก้ไขข้อมูลโครงการ , ภาพแผนผังโครงการ , ข้อมูลบ้าน
9. การแก้ไขและเพิ่มรายละเอียดของบ้านใหม่

7.4 การลงทะเบียน

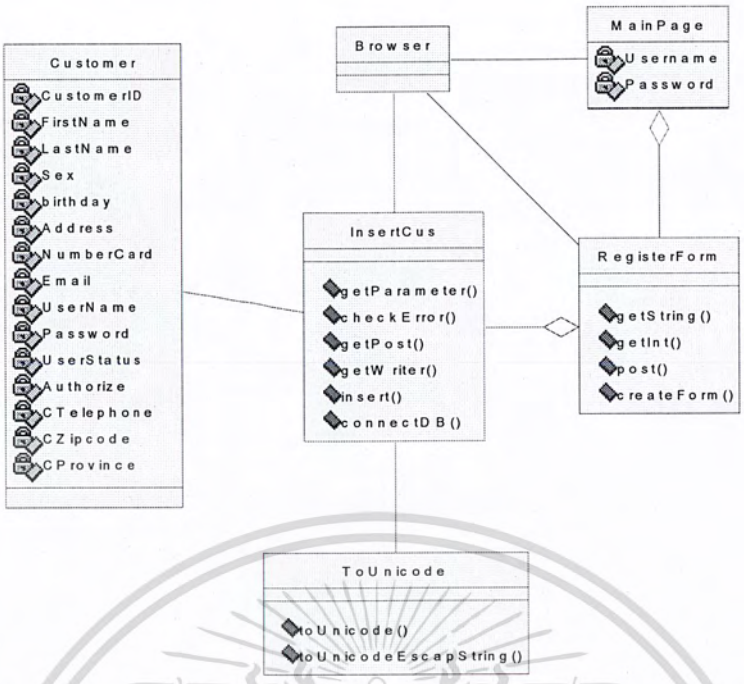
ในการที่จะใช้บริการต่าง ๆ เช่นการจองบ้าน,การประกาศขาย/เช่าบ้าน ของเว็บไซต์นั้น ลูกค้าจะต้องทำการลงทะเบียนกับระบบก่อนเพราะเนื่องจากการบริการบางอย่างนั้นจะมีการเก็บขอ้มูลบางส่วนลูกค้าไว้เพื่อประโยชน์และความสะดวกแก่ลูกค้าและผู้พัฒนาระบบดังนั้นขั้นตอนและลำดับของการลงทะเบียนมีการทำงานดังนี้

เมื่อลูกค้าเข้าสู่หน้าหลักของเว็บ จะมีปุ่มสมัครสมาชิกใหม่ ให้ลูกค้ากดปุ่ม "**สมัครสมาชิกใหม่** " ซึ่งจะทำให้ลูกค้าเข้าสู่หน้าจอของแบบฟอร์มการลงทะเบียน ซึ่งจะมีแบบฟอร์มให้ลูกค้ากรอกข้อมูล เมื่อกรอกข้อมูลเสร็จแล้ว ให้ลูกค้ากดที่ปุ่ม Submit ก็จะแสดงหน้าจอแสดงว่าทำการลงทะเบียนเรียบร้อยแล้ว หรือเลือกกดปุ่ม Clear เพื่อลบข้อความเดิมและทำการกรอกข้อมูลใหม่ เมื่อกดปุ่ม Submit แล้วบราวเซอร์จะร้องขอให้เว็บเซิร์ฟเวอร์เรียกใช้ InsertCus ซึ่งเป็นไฟล์ servlet ให้ติดต่อกับฐานข้อมูลเพื่อทำการเก็บข้อมูลของลูกค้าลงฐานข้อมูลโดยก่อนจะเก็บลงฐานข้อมูลต้องทำการเปลี่ยนเป็นข้อมูลที่เป็นภาษาไทยเสีย

ก่อน โดยการเรียกใช้คลาส ToUnicode เมื่อเก็บข้อมูลลูกค้าลงสู่ฐานข้อมูลเรียบร้อยแล้วก็จะแสดงหน้าเว็บว่าลงทะเบียนสมบูรณ์แล้ว



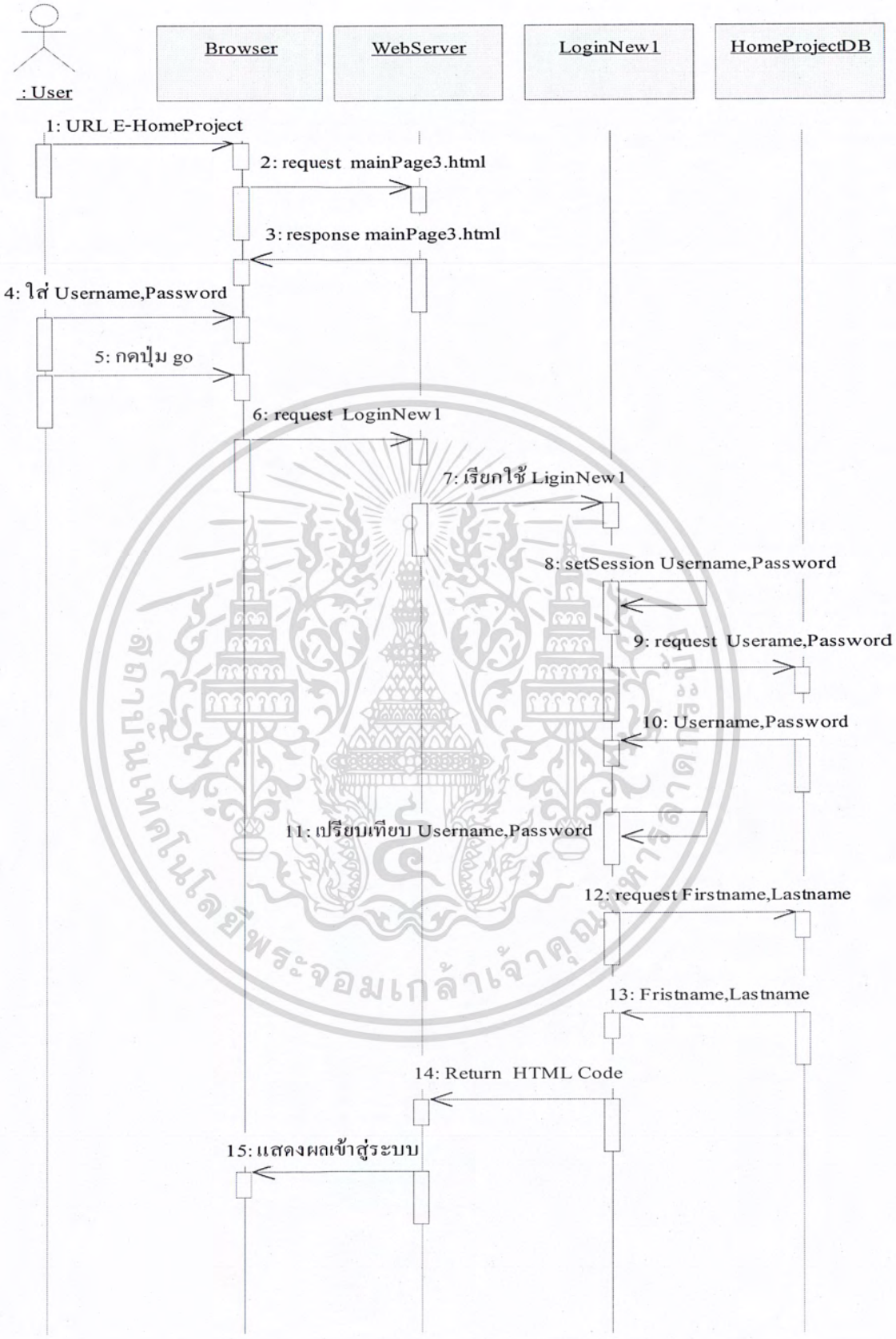
รูปที่ 7-2 แสดง Sequence Diagram การลงทะเบียนของลูกค้า



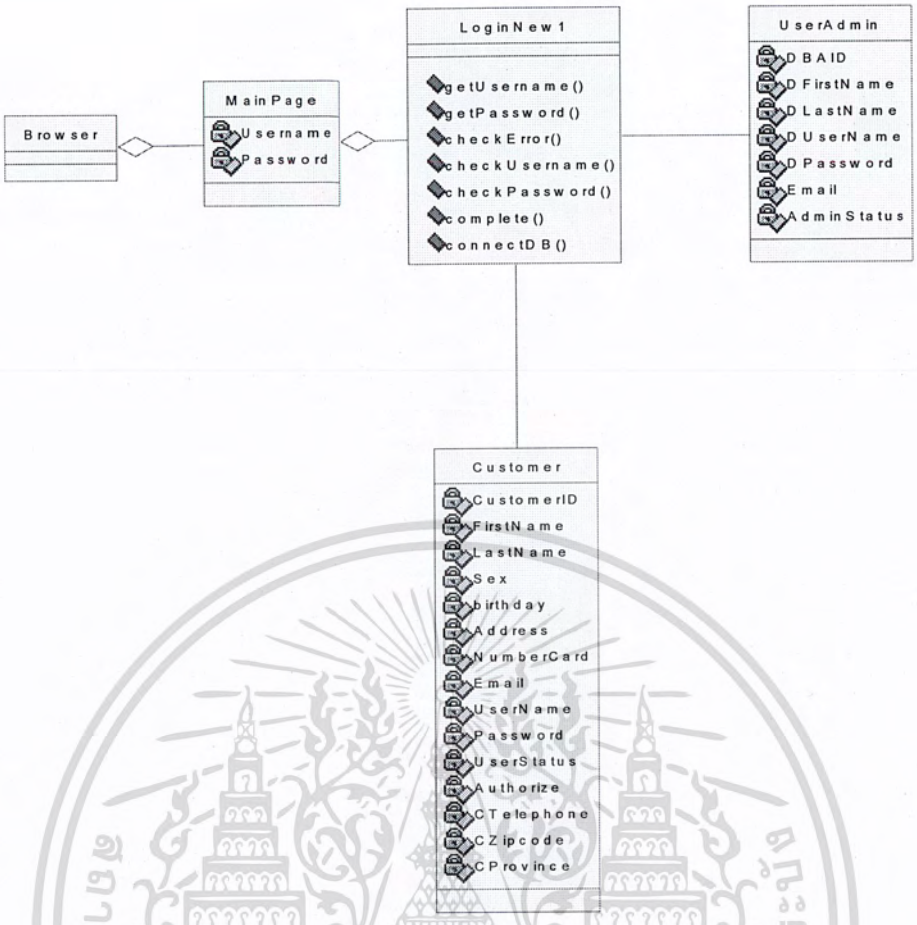
รูปที่ 7-3 แสดง Class Diagram การลงทะเบียนของลูกค้า

7.5 การเข้าสู่ระบบ

เมื่อลูกค้าเข้าสู่หน้าหลักของเว็บ จะมีช่องให้กรอก UserName และ Password เมื่อกรอกเสร็จแล้วก็จะต้องทำการกดปุ่ม Go จากนั้นจะไปทำการเรียกไฟล์ LoginNew1 ซึ่งเป็น servlet ที่ทำหน้าที่เก็บ UserName และ Password ใน session (session คือ การเก็บค่าตัวแปรที่มีลักษณะคล้ายกับตัวแปรแบบ Globol ซึ่งจะใช้ในการอ้างอิงค่าต่าง ๆ ที่เก็บไว้เช่น การเก็บค่า Username และ Password ใน session ตลอดจนการทำงานเมื่อมีการเข้ามาสู่ระบบโดยผ่านการ ล็อกอิน)ไว้แล้วนำ UserName และ Password ที่รับเข้ามาไปเปรียบเทียบกับUserName และ Password ในฐานข้อมูล ถ้าถูกต้องจะแสดงผลการเข้าสู่ระบบ ถ้าไม่ถูกต้องจะแสดงผลข้อผิดพลาดออกมา



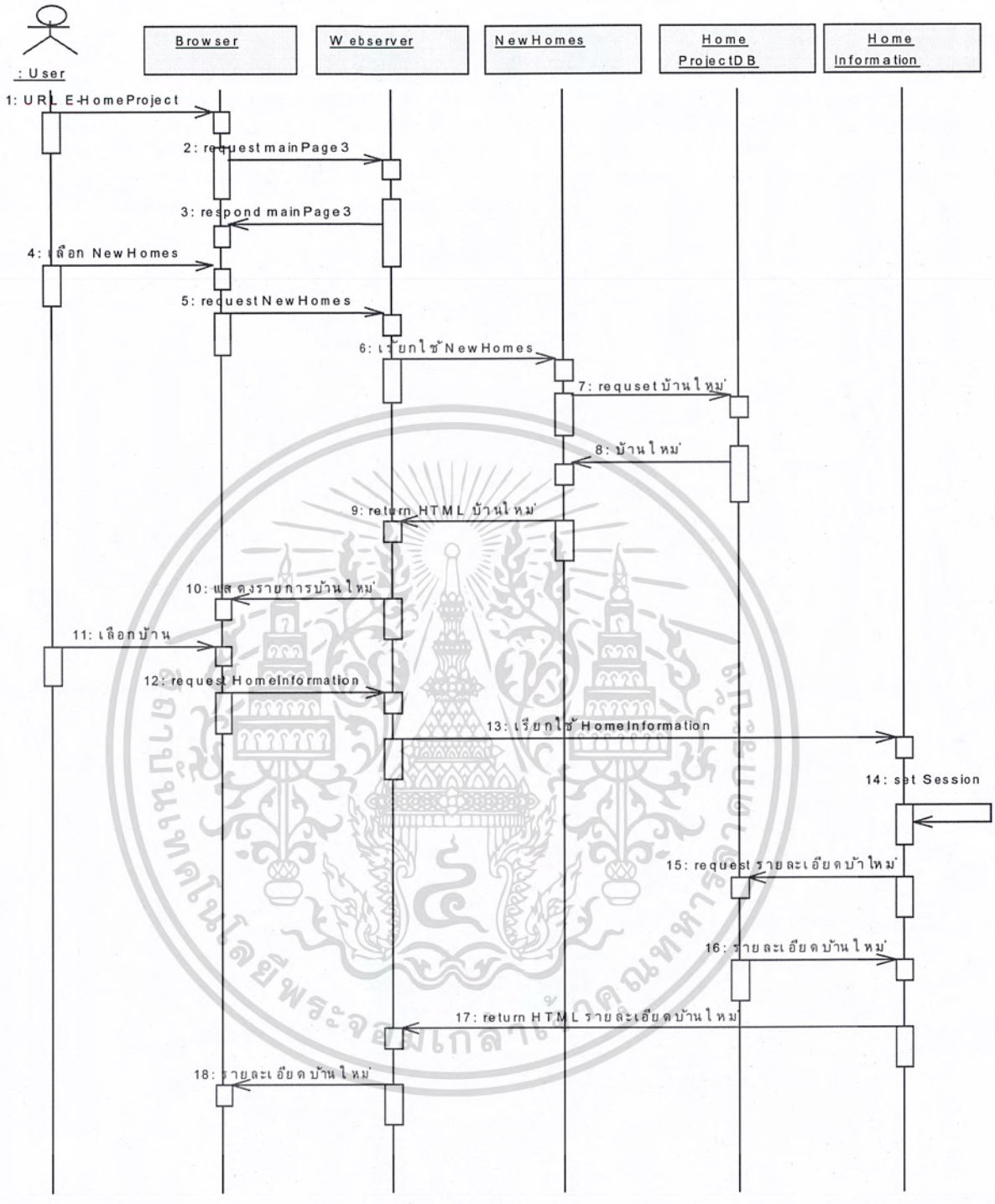
รูปที่ 7-4 แสดง Sequence Diagram การเข้าสู่ระบบ



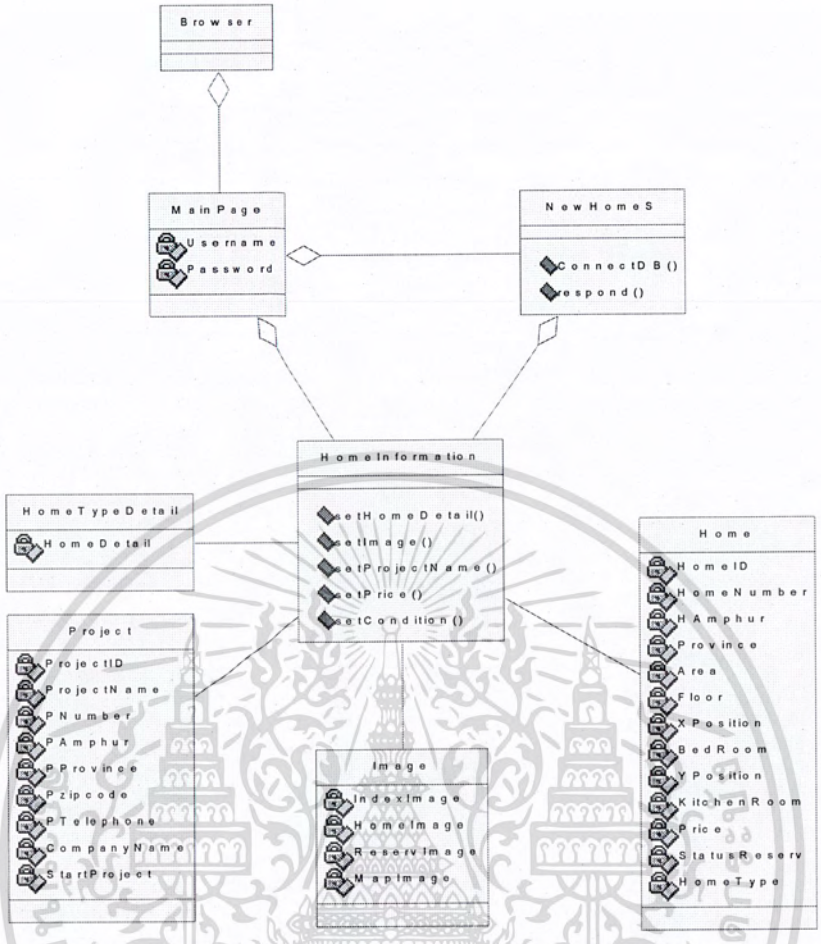
รูปที่ 7-5 แสดง Class Diagram การเข้าสู่ระบบ

7.6 การดูรายละเอียดโครงการ

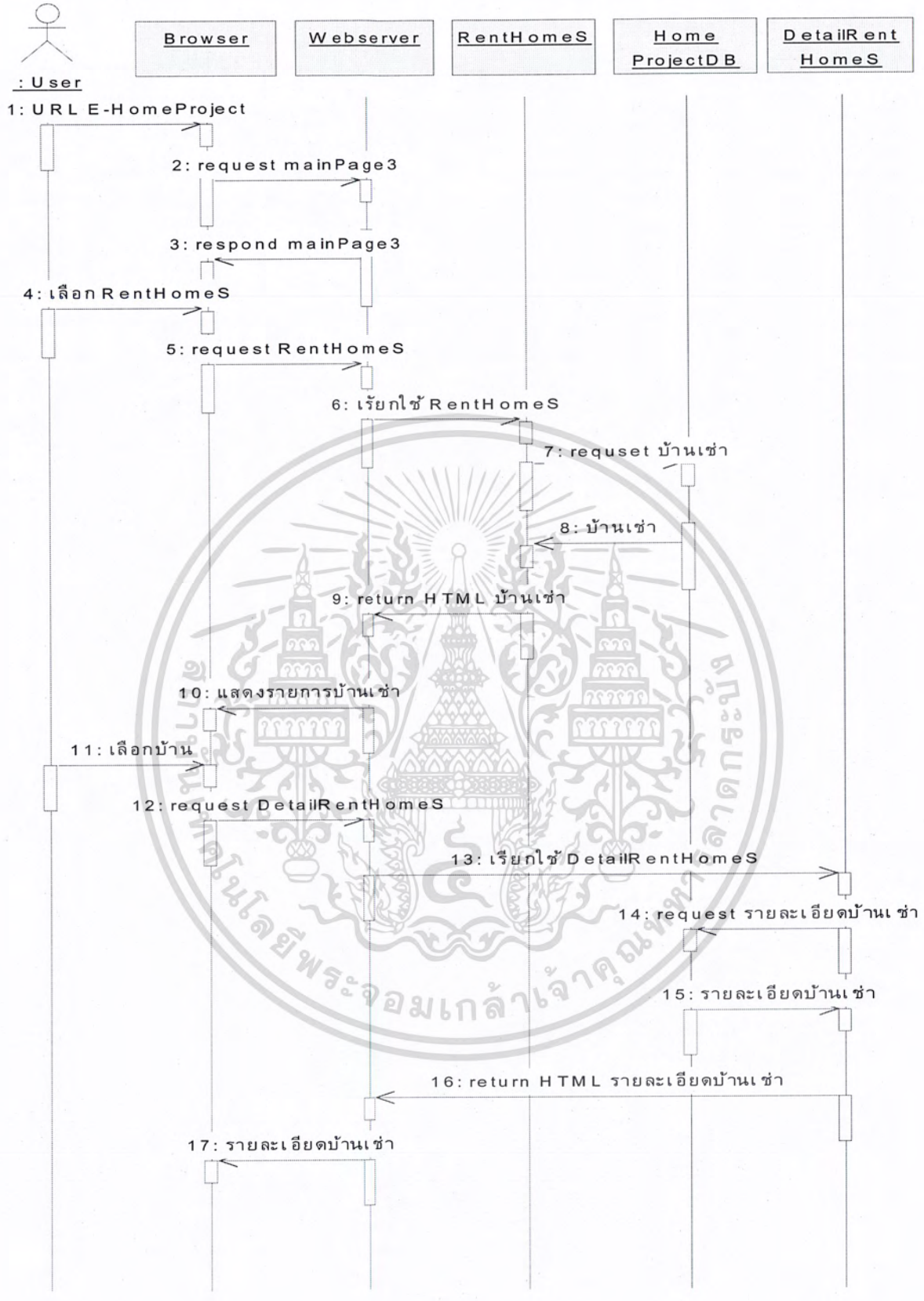
ในการดูรายละเอียดโครงการบ้านนั้น ลูกค้าจะสามารถเข้าไปดูรายละเอียดโครงการได้โดยไม่จำเป็นต้องผ่านการ ล็อกอิน เมื่อลูกค้าเข้าสู่หน้าหลักของเว็บไซต์จะมีส่วนที่แสดง แนะนำบ้านใหม่, New Homes, Home Rentals และ Home Secondhand ซึ่งลูกค้าสามารถที่จะเลือกดูประเภทของได้ตามความต้องการ โดยจะขั้นตอนจาก Sequence Diagram การดูรายละเอียดโครงการบ้านได้ ดังนี้



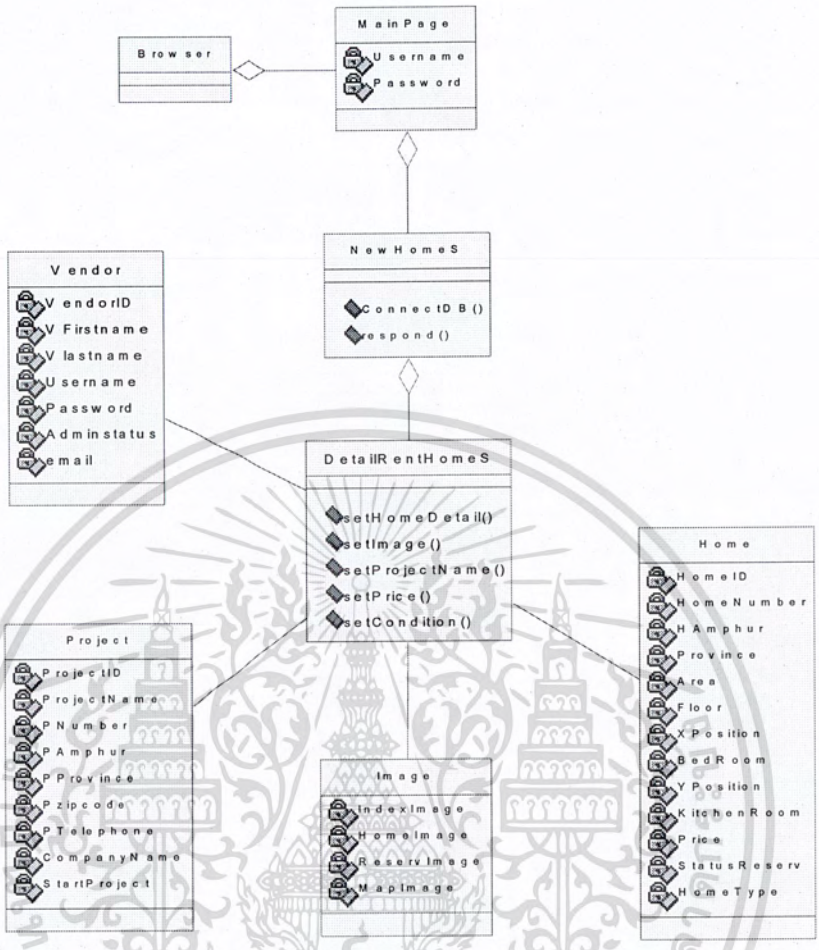
รูปที่ 7-6 แสดง Sequence Diagram ดูรายละเอียดบ้านใหม่



รูปที่ 7-7 แสดง Class Diagram การดูรายละเอียดบ้านใหม่

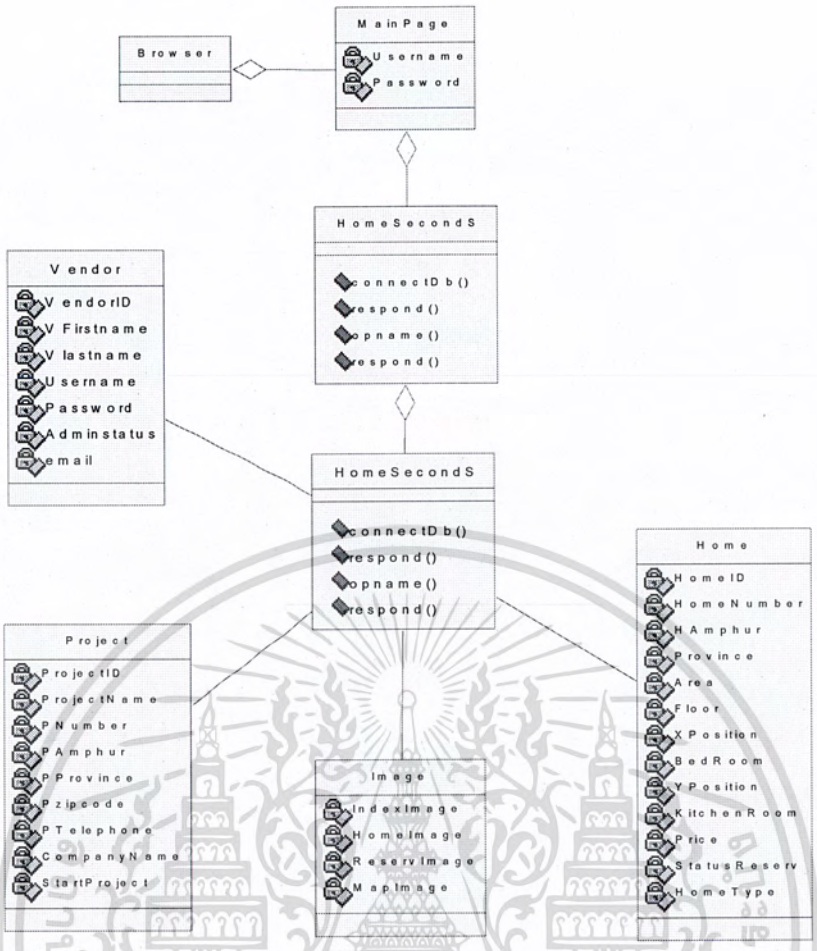


รูปที่ 7-8 แสดง Sequence Diagram การดูรายละเอียดบ้านเช่า



รูปที่ 7-9 แสดง Class Diagram การดูรายละเอียดบ้านเช่า

รูปที่ 7-10 แสดง Sequence Diagram การดูรายละเอียดบ้านมือสอง

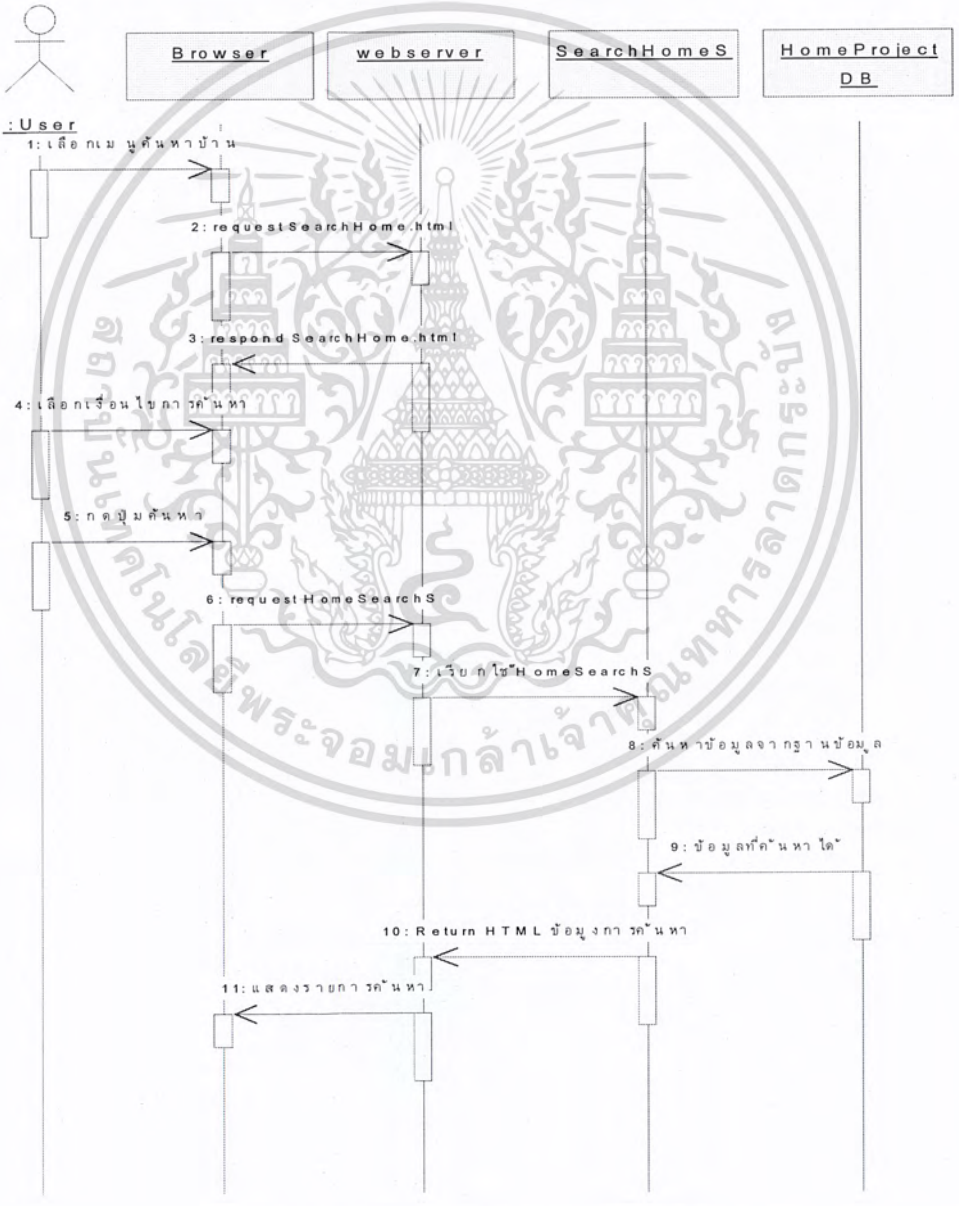


รูปที่ 7-11 แสดง Class Diagram การดูรายละเอียดบ้านมือสอง

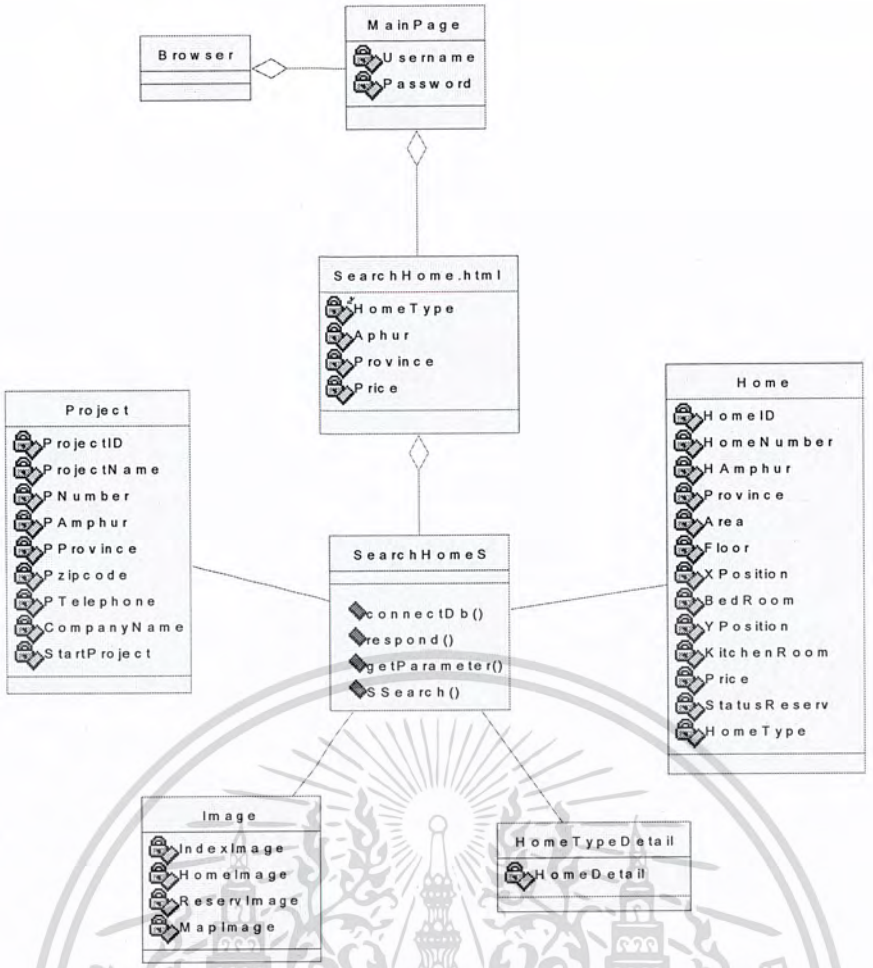
7.7 การค้นหาบ้าน

ในการค้นหาบ้านนั้น ลูกค้าจะสามารถค้นหารายละเอียดที่ต้องการได้โดยไม่จำเป็นต้องมีการเข้าสู่ระบบ ลูกค้าสามารถดูข้อมูลโครงการต่างๆโดยการค้นหา จากลักษณะบ้านที่ต้องการ จากทำเลที่ตั้ง จากงบประมาณ

จากรูปแสดงของ Sequence Diagram การค้นหาบ้านนั้น เมื่อลูกค้าเข้าสู่หน้าหลักของเว็บ จะมีเมนู ค้นหาบ้าน เมื่อลูกค้ากดที่เมนู ค้นหาบ้าน ก็จะลิงค์ไปยัง HomeSearch.html ซึ่งจะแสดงหน้าจอการค้นหาบ้านออกมาให้ลูกค้าทำการใส่รายละเอียดของการค้าหาแล้วจึงทำการกดปุ่มค้นหา จากนั้นจะทำการเรียกใช้คลาส HomeSearchS ซึ่งทำหน้าที่นำข้อมูลที่ลูกค้ากำหนดรายละเอียดไปค้นหาข้อมูลที่ต้องการจากฐานข้อมูล จากนั้นก็แสดงข้อมูลการค้นหาทั้งหมดที่ได้จากฐานข้อมูลให้ลูกค้าได้ทราบ



รูปที่ 7-12 แสดง Sequence Diagram การค้นหาบ้าน

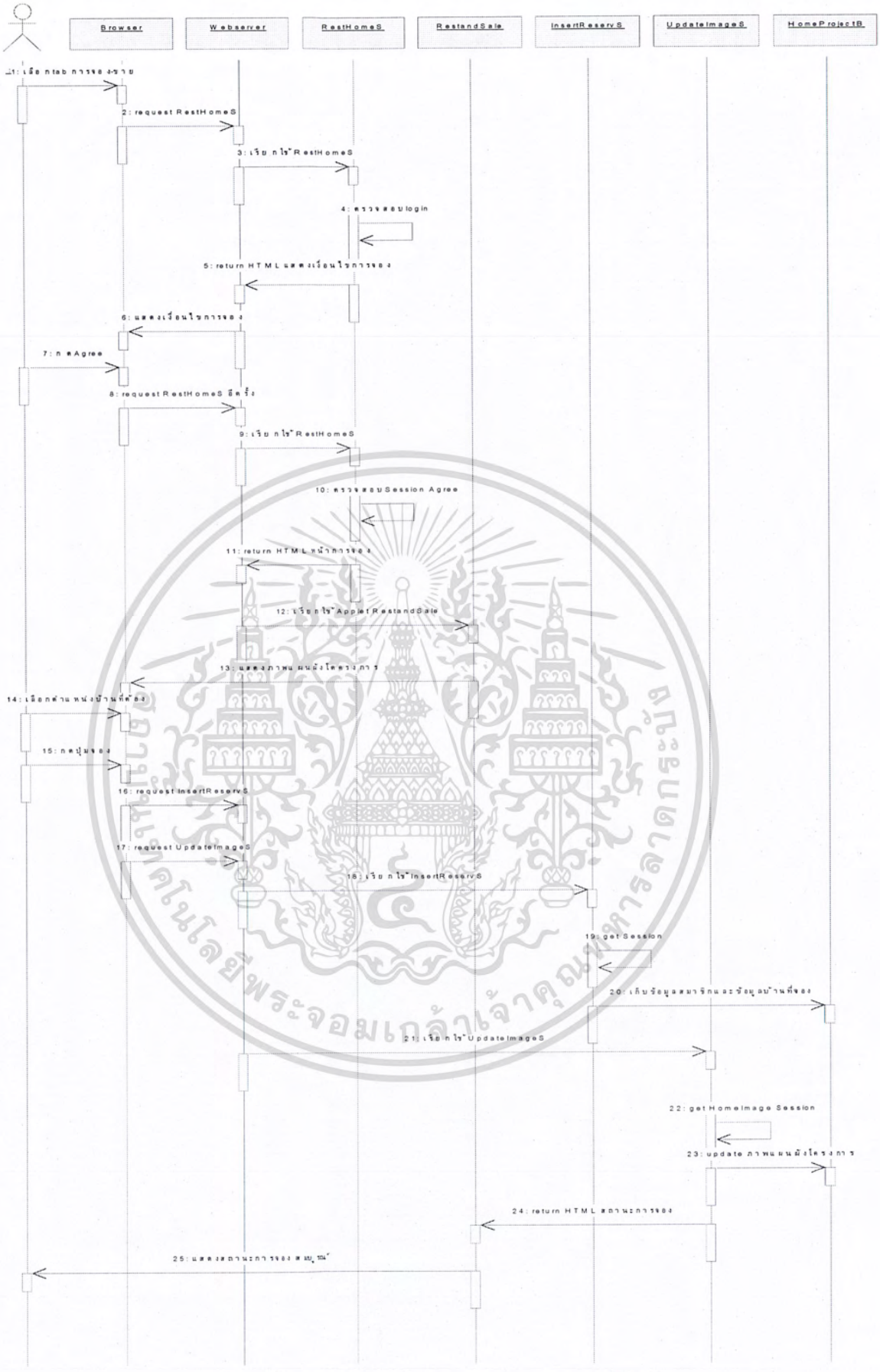


รูปที่ 7-13 แสดง Class Diagram การค้นหาบ้าน

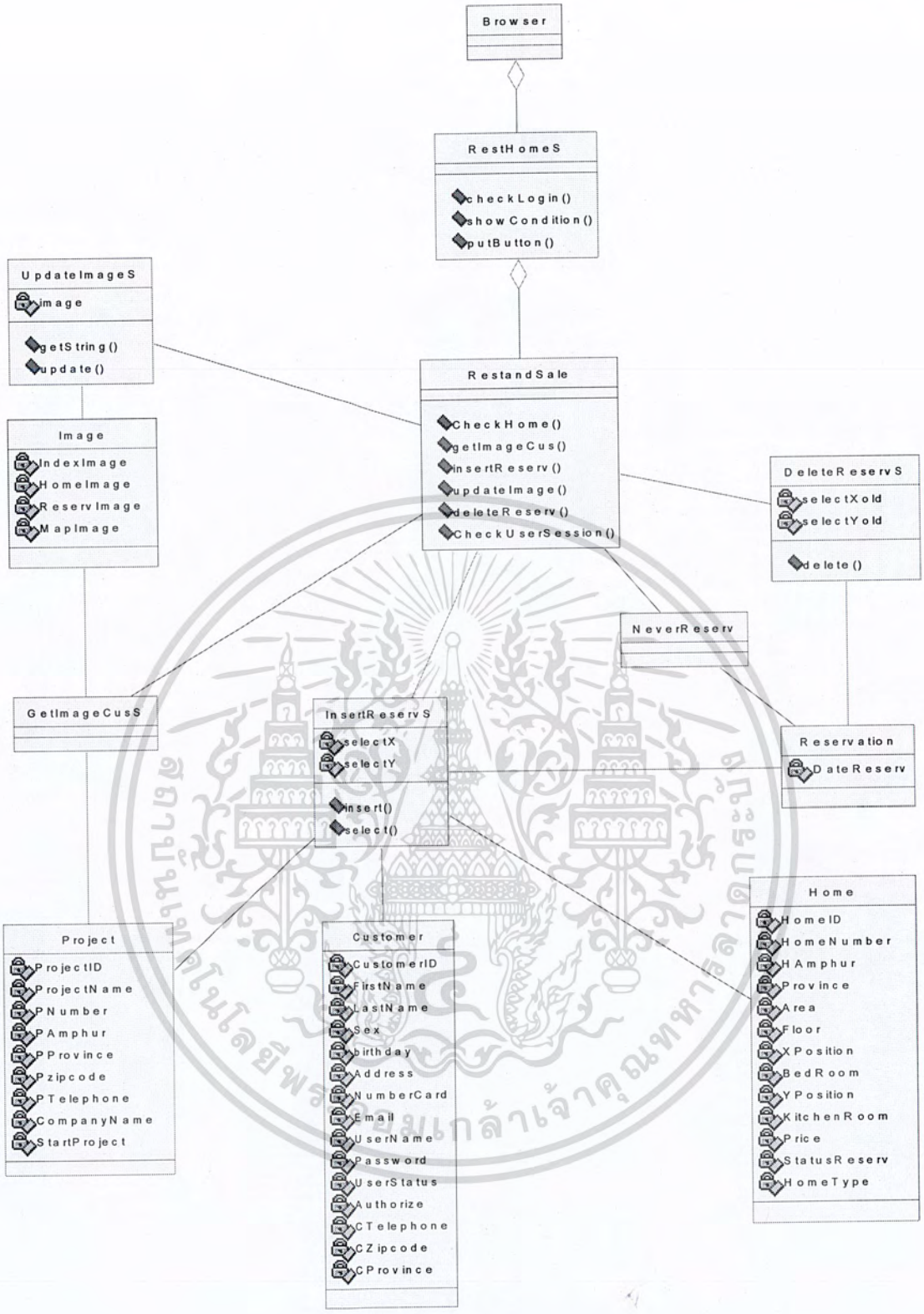
7.8 การจองบ้าน

ในการจองบ้าน ลูกค้าจะสามารถจองบ้านที่ต้องการได้โดยการ ล็อกอิน เพื่อเข้าสู่ระบบก่อน จึงจะทำการจองได้โดยการจองบ้านนั้นลูกค้าสามารถจองได้ครั้งละ 1 หลังต่อ 1 โครงการเท่านั้น จากนั้นลูกค้าจะต้องมาติดต่อยังสำนักงานขาย เพื่อที่จะได้ดำเนินการตามขั้นตอนการทำสัญญาต่อไป

จากรูปแสดงของ Sequence Diagram การจองบ้านนั้น เมื่อลูกค้าเข้าสู่หน้าหลักของเว็บ จะต้องเข้าไปดูรายละเอียดของโครงการและรายละเอียดของบ้านก่อนที่จะตัดสินใจทำการจอง เมื่อต้องการจองก็จะเลือกแท็บ จอง – ขาย เมื่อลูกค้ากดไปที่ แท็บจอง – ขาย จะไปทำการเรียกใช้คลาส RestHomeS ซึ่งจะทำการตรวจสอบ ล็อกอิน ว่าได้ลงทะเบียนไว้แล้วหรือไม่ ถ้ายังไม่ได้ลงทะเบียนไว้ก็จะแจ้งให้ทำการลงทะเบียนให้เรียบร้อยก่อนที่จะทำการจองบ้าน จากนั้นจะแสดงเงื่อนไขของการจองให้ลูกค้าได้ทราบ ถ้าลูกค้ายอมรับก็จะต้องกดปุ่มเพื่อการยอมรับ แล้วจะทำการเรียกใช้คลาส RestHomeS อีกครั้งหนึ่งก่อนจะทำไปสู่หน้าของการจอง โดยจะเป็นแผนภาพของโครงการซึ่งจะมีบ้านให้เลือกจอง โดยดูจากสถานะว่าจองได้หรือไม่ ถ้าจะทำการจองก็ต้องเลือกตำแหน่งบ้านที่ต้องการ แล้วกดปุ่มจอง จากนั้นจะไปเรียกใช้คลาส InsertReservS ซึ่งจะทำการเก็บข้อมูลผู้ที่ทำการจองและข้อมูลของบ้านที่จองลงสู่ฐานข้อมูล และอัพเดทสถานะของบ้านในฐานข้อมูล เรียกใช้ UpdateImageS อัพเดทภาพแผนผังโครงการเพื่อให้ข้อมูลลงสู่ฐานข้อมูล เมื่อเรียบร้อยแล้วก็จะทำการแสดงให้ลูกค้าได้ทราบว่าได้ทำการจองเรียบร้อยแล้ว



รูปที่ 7-14 แสดง Sequence Diagram การจองบ้าน



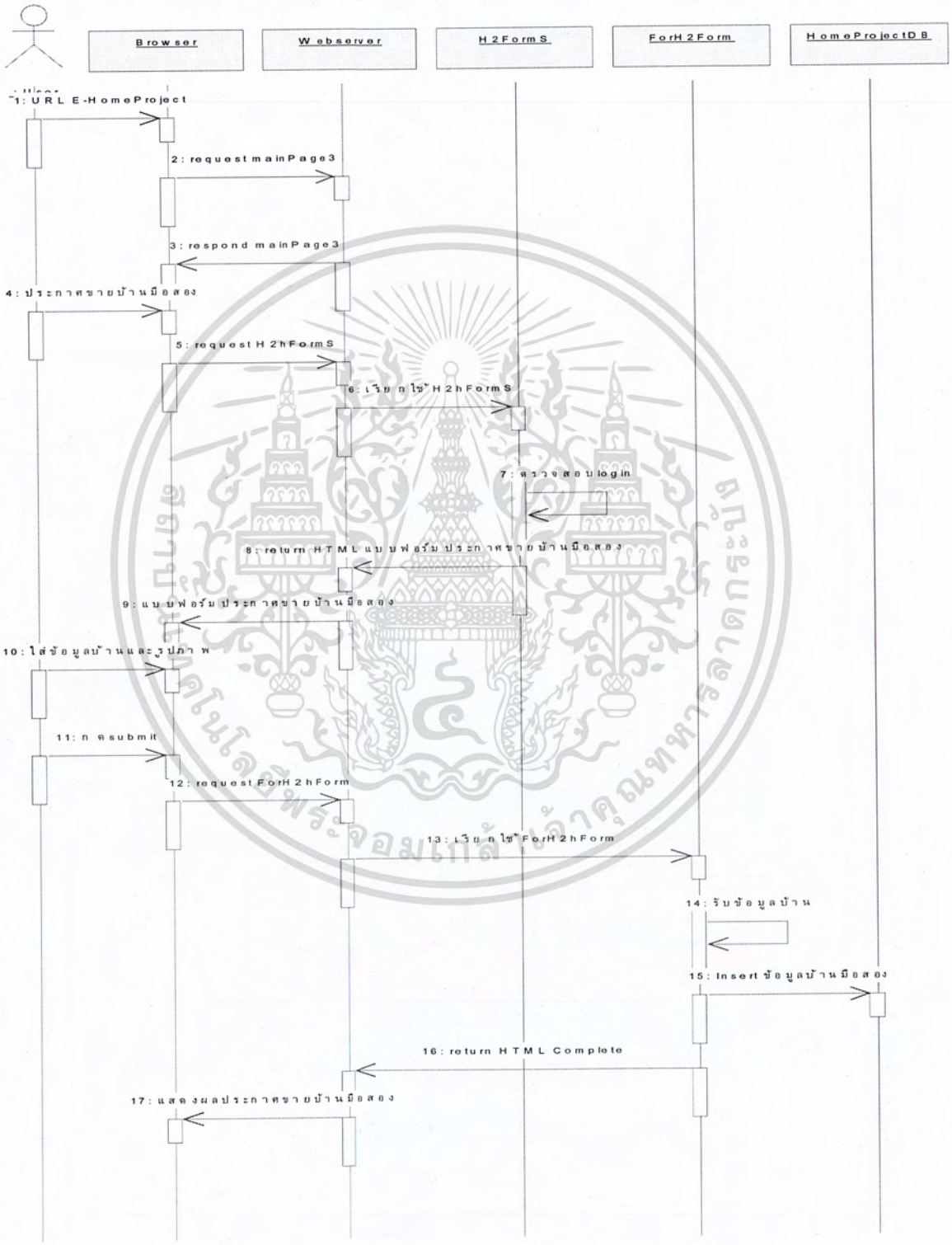
รูปที่ 7-15 แสดง Class Diagram การจองบ้าน

7.9 การประกาศ ขาย/เช่า บ้าน

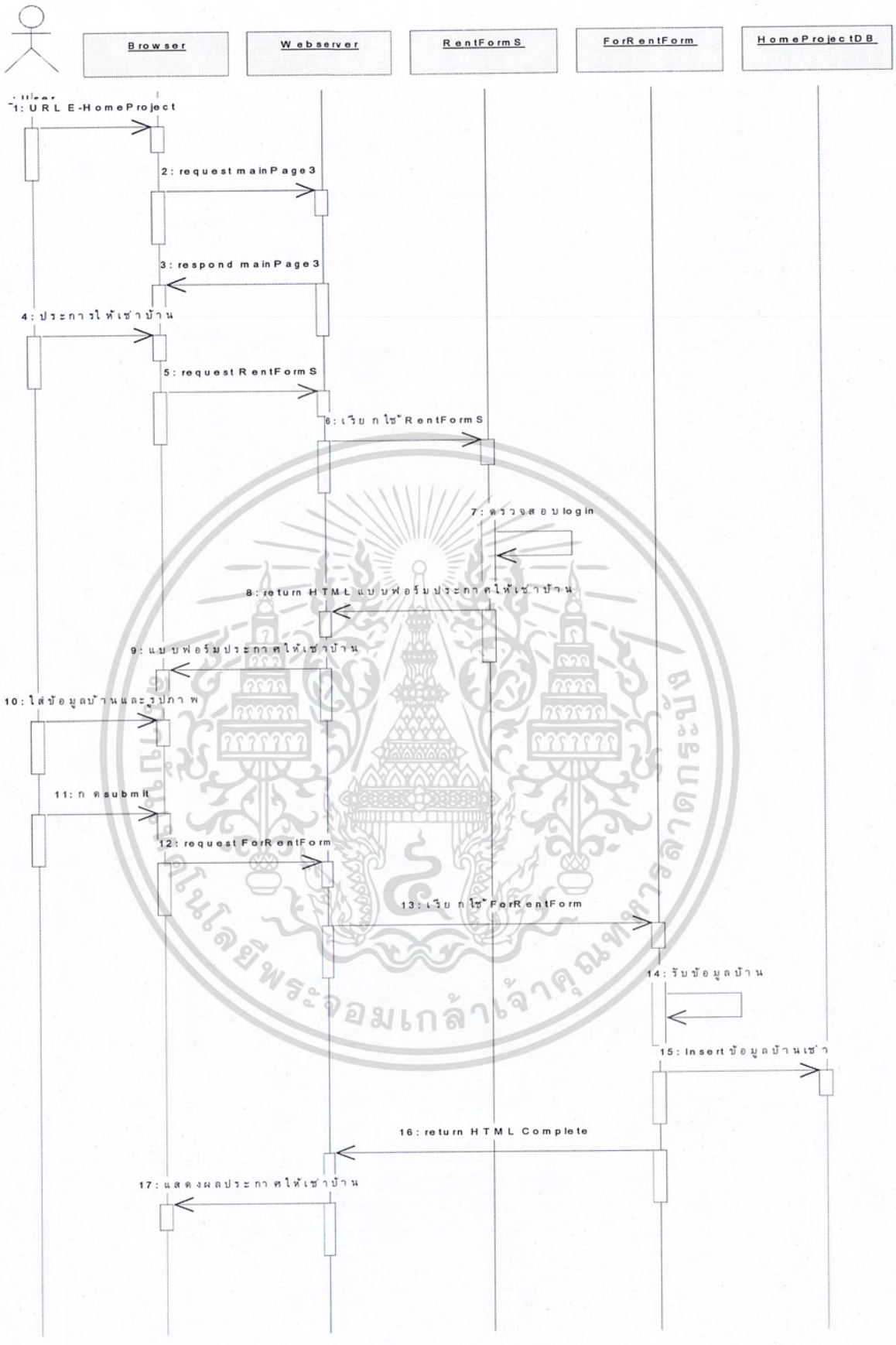
การประกาศ ขาย/เช่า บ้านเป็นการให้บริการฟรีสำหรับลูกค้า โดยการประกาศนั้นจะแบ่งออกเป็น 2 ประเภท คือ

1. การประกาศขายบ้านมือสอง
2. การประกาศให้เช่าที่พักอาศัย

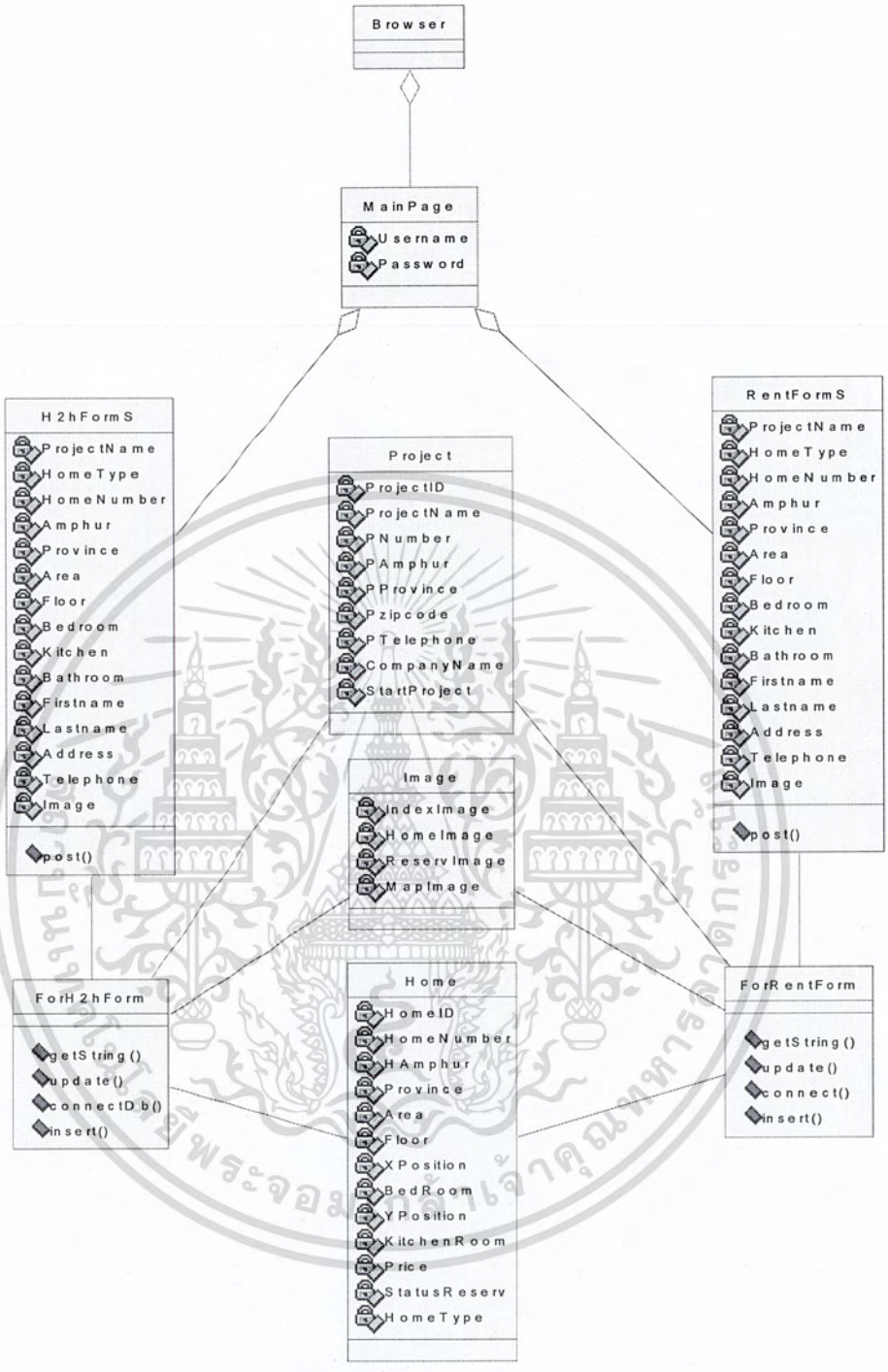
ในขั้นตอนของการประกาศนั้น ลูกค้าจะเข้าสู่หน้าของแบบฟอร์มการประกาศทั้ง 2 ประเภทได้ต้องผ่านการ ล็อกอิน เพื่อเข้าสู่ระบบก่อนถึงจะสามารถทำการกรอกแบบฟอร์มของการประกาศได้ ในแบบฟอร์มนั้นจะมีช่องว่างให้ใส่ข้อมูลที่เกี่ยวกับบ้านรูปภาพของบ้านได้



รูปที่ 7-16 แสดง Sequence Diagram การประกาศขายบ้านมือสอง



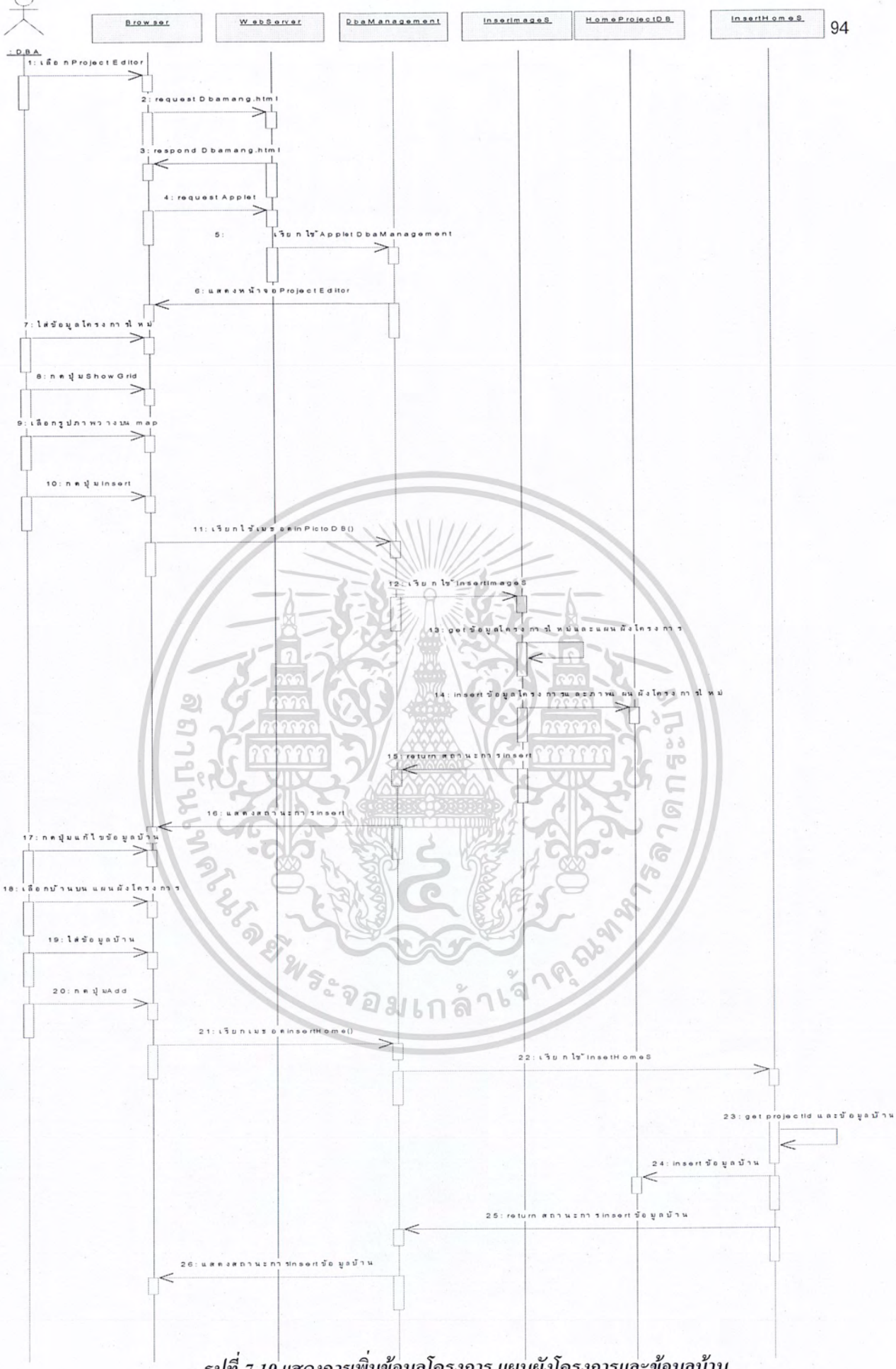
รูปที่ 7-17 แสดง Sequence Diagram การประกาศให้เช่าที่พักอาศัย



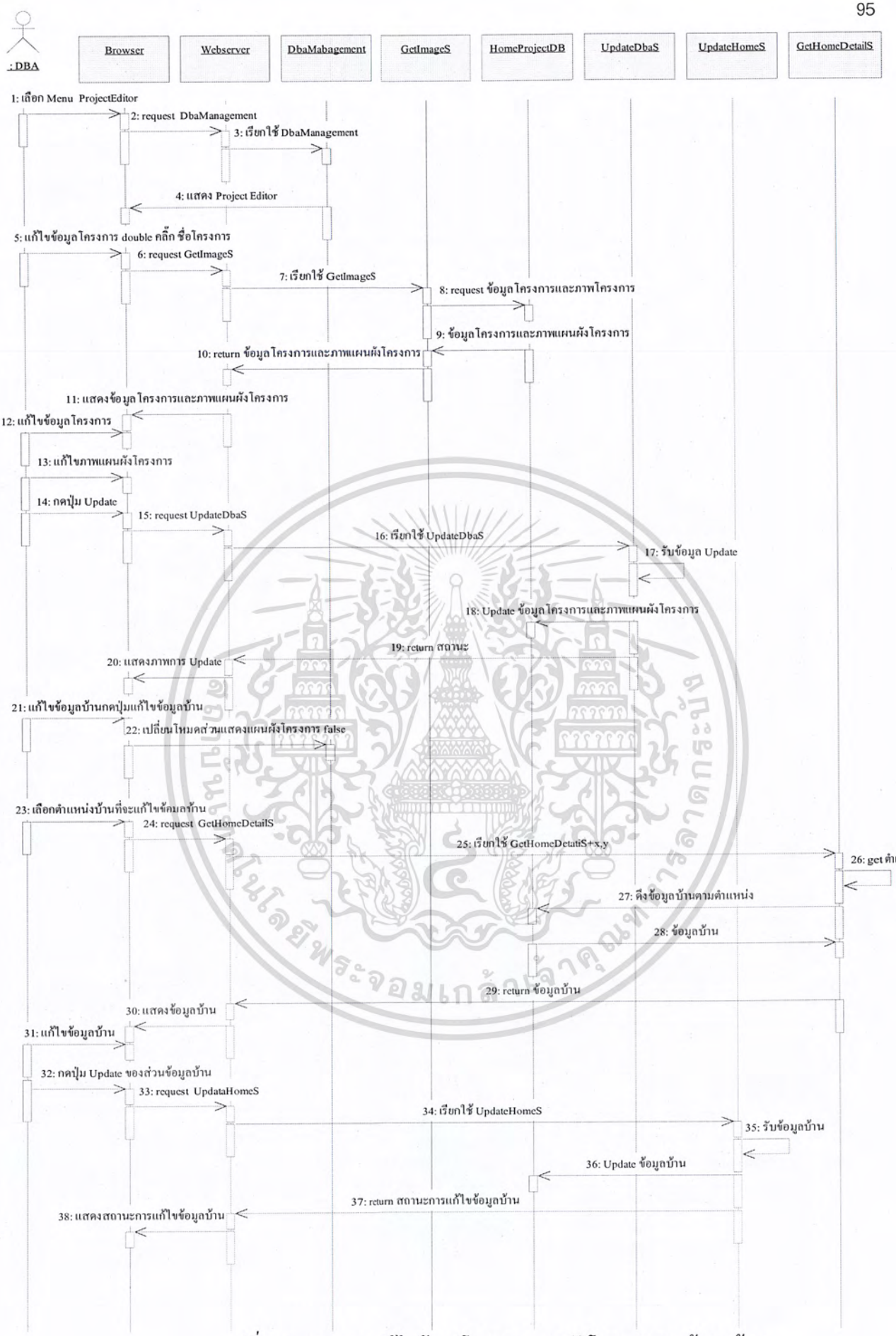
รูปที่ 7-18 แสดง Class Diagram การประกาศ ขาย/เช่า บ้าน

7.10 ผู้ดูแลระบบ

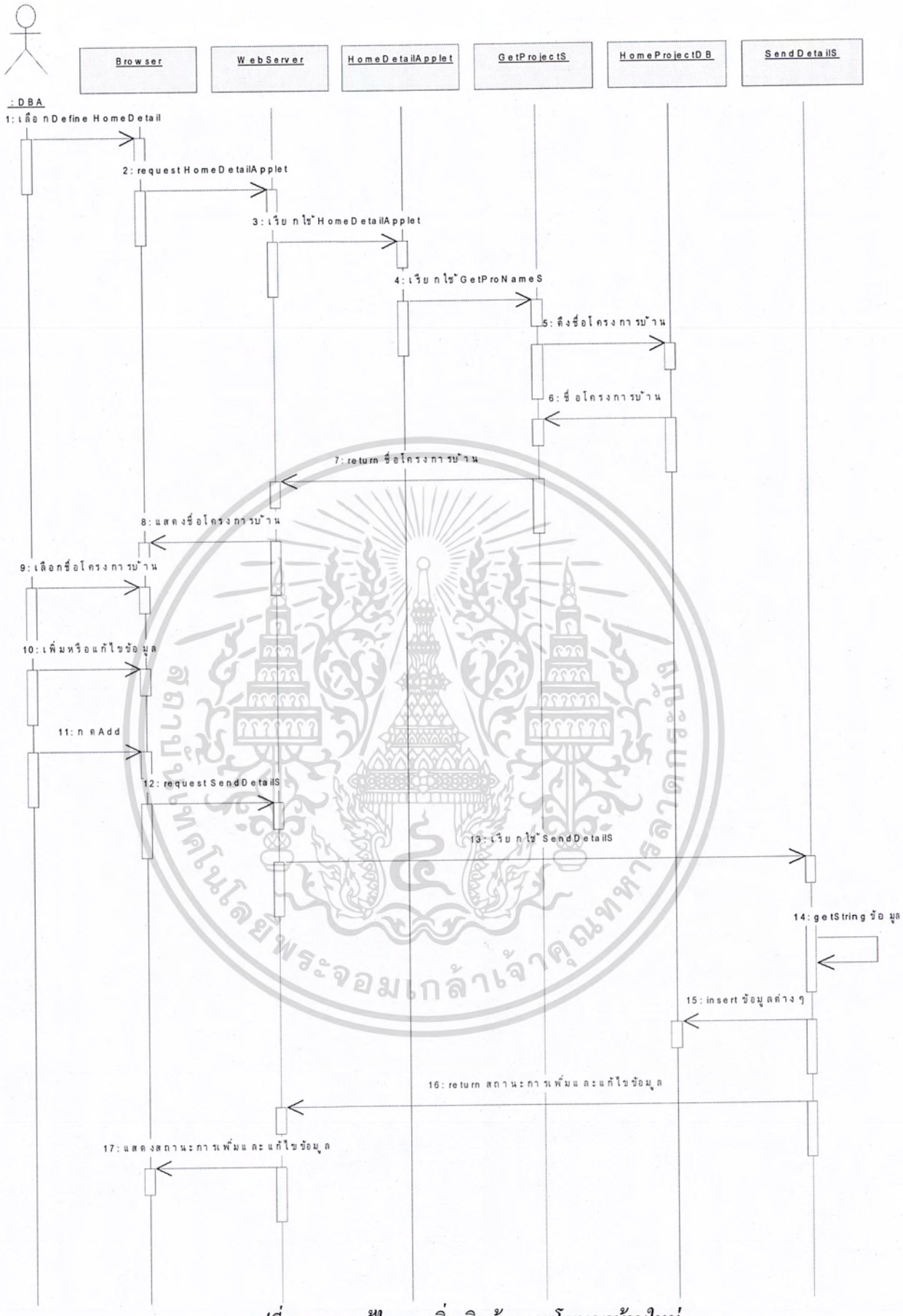
ในการเพิ่มและแก้ไขข้อมูลโครงการต่างๆ จะเป็นหน้าที่ของผู้ดูแลระบบ โดยจะต้องล็อกอินเข้าสู่ระบบที่หน้าจอหลัก เมื่อระบบตรวจสอบแล้วว่าเป็น UserName และ Password ของผู้ดูแลระบบแล้วก็จะเข้าสู่หน้าเมนูของผู้ดูแลระบบซึ่งมีอยู่ 2 ตัวเลือกคือ ถ้าต้องการแก้ไขหรือเพิ่มเติมข้อมูลโครงการ,ข้อมูลบ้านและรูปภาพแผนผังโครงการจะเลือก Project Editor แต่ถ้าต้องการแก้ไขหรือเพิ่มรายละเอียดบ้านใหม่ซึ่งใช้เป็นข้อความในการประกาศโฆษณาสำหรับบ้านใหม่ซึ่งจะประกอบไปด้วย การกำหมดรูปภาพบ้าน, รูปภาพแผนที่โครงการและ เงื่อนไขโครงการ จะเลือก Define HomeDetail



รูปที่ 7-19 แสดงการเพิ่มข้อมูลโครงการ,แผนผังโครงการและข้อมูลบ้าน



รูปที่ 7-20 แสดงการแก้ไขข้อมูลโครงการ,แผนผังโครงการและข้อมูลบ้าน



รูปที่ 7-21 การแก้ไขและเพิ่มเติมข้อความโฆษณาบ้านใหม่

บทที่ 8

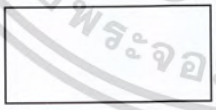
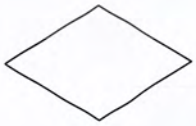
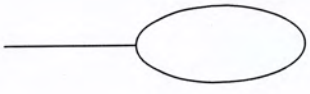
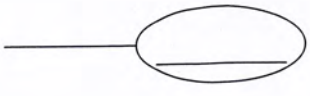
การออกแบบฐานข้อมูลโดยใช้ E-R Diagram

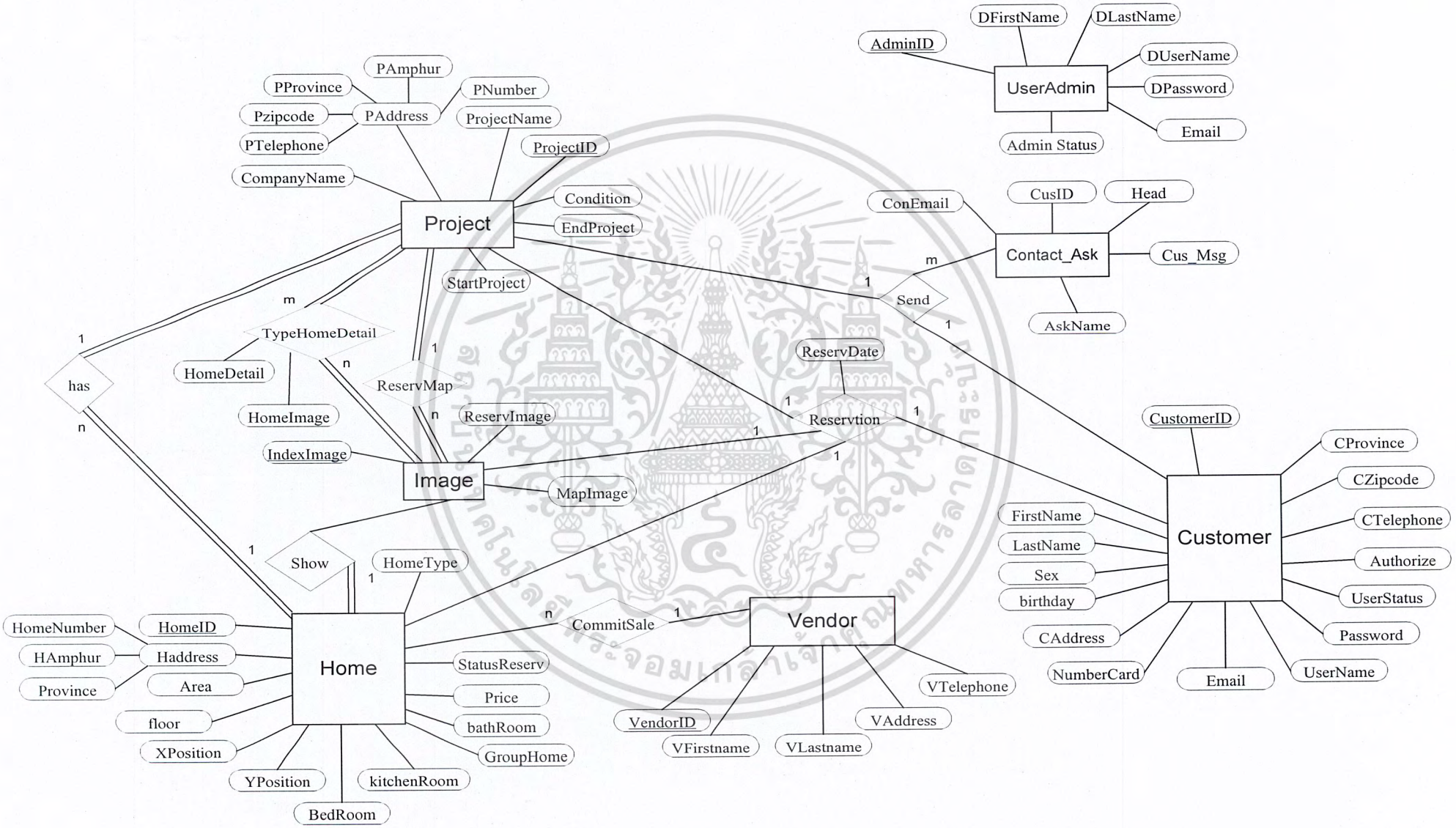
8.1 การ Mapping จาก E-R ไปเป็น Relational

1. Map Regular Entity Type เป็นหนึ่งตารางโดย Attribute ที่ map ไปจะเป็น Attribute ทุกชนิด ยกเว้น Multivalue attribute
2. Map Weak Entity Type เป็นหนึ่งตารางในลักษณะเดียวกับ ข้อ 1 primary key จะเป็น partial key combice กับ primary key ของ parents.
3. 1:1 Relationship ให้ Map 1:1 Relationship เป็น primary key เป็น F.K. โดยไม่ต้องสร้างตารางใหม่ ถ้ามีฝั่งใดฝั่งหนึ่งเป็น total participation ให้ใช้ฝั่ง total เป็นหลัก ยก primary key อีกฝั่งหนึ่งมาเกาะเป็น F.K. พร้อมด้วย Attribute ของความสัมพันธ์นั้น
4. 1:n ไม่ต้องการตารางใหม่ ใช้ Primary key และ F.K. ได้ โดยใช้ฝั่ง many เป็นหลัก ยกฝั่ง one มาเกาะ พร้อมด้วย Attribute เป็น F.K.
5. n:n map เห็นหนึ่งตารางโดย primary key จะเป็น combine key ของ primarykey ที่เกี่ยวข้อง
6. Multivalue Attribute ใช้สร้างตารางใหม่ สำหรับ Multivalue Attribute โดยตารางใหม่นี้จะประกอบด้วย Multivalue Attribute combine กับ privary key เอง Entitytype เป็น combine primary key
7. n-ary relationship เมื่อ n เท่ากับ 3 ขึ้นไป map เป็น 1 ตาราง ในลักษณะเดียวกับ step 5

8.2 แสดงการออก ER Diagram ของระบบการจองบ้านออนไลน์

เครื่องหมายสัญลักษณ์ที่ใช้ในการออกแบบ ER Diagram

สัญลักษณ์	ความหมาย
	- Entity Type
	- Relationship Type
	- Attribute
	- Key Attribute



รูปที่ 8-1 แสดง ER Diagram ของระบบการจองบ้านออนไลน์

8.3 ตารางข้อมูลที่ได้จาก E-R Diagram

1. ตาราง Project

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	ProjectID	Number	เลขที่ของโครงการเป็น P.K Not null
2	ProjectName	varchar2(100)	ชื่อโครงการ
3	Pnumber	varchar2(10)	เลขที่อยู่โครงการ
4	Pamphur	varchar2(20)	อำเภอ
5	Pprovince	varchar2(20)	จังหวัด
6	Pzipcode	varchar2(10)	รหัสไปรษณีย์
7	Ptelephone	varchar2(15)	เบอร์โทรศัพท์
8	Company	varchar2(100)	บริษัท
9	DateStart	varchar2(30)	วันที่เริ่มโครงการ
10	DateEnd	varchar2(30)	วันสิ้นสุดโครงการ
11	Condition	varchar2(100)	เงื่อนไขของการจอง

ตารางที่ 8-1 แสดงตารางที่ใช้เก็บข้อมูลของโครงการที่ชื่อว่า Project

2. ตาราง Customer

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	CustomerID	Number	เลขที่ของลูกค้าเป็น P.K. Not Null
2	FirstName	varchar2(20)	ชื่อของลูกค้า
3	LastName	varchar2(20)	นามสกุลของลูกค้า
4	Sex	varchar2(1)	เพศของลูกค้า
5	Birthday	varchar2(30)	วันเกิดของลูกค้า
6	Address	varchar2(50)	ที่อยู่ของลูกค้า
7	Province	varchar2(20)	จังหวัด
8	Zipcode	varchar2(10)	รหัสไปรษณีย์
9	Telephone	varchar2(15)	เบอร์โทรศัพท์
10	NumberCard	varchar2(20)	หมายเลขบัตรประชาชน

11	Email	varchar2(30)	E-Mail Address
12	UserName	varchar2(10)	UserName
13	Password	varchar2(8)	Password
14	UserStatus	varchar2(10)	สถานะลูกค้า

ตารางที่ 8-2 แสดงตารางที่ใช้เก็บข้อมูลของลูกค้าที่ชื่อว่า Customer

3. ตาราง Image

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	IndexImage	Number	หมายเลขของภาพเป็น P.K. Not null
2	ProjectID	Number	เลขที่ของโครงการเป็น F.K Not null
4	ReservImage	varchar2(2000)	รูปการจองของบ้าน
5	MapImage	varchar2(20)	แผนภาพของบ้าน

ตารางที่ 8-3 แสดงตารางที่ใช้เก็บข้อมูลของรูปภาพที่ชื่อว่า Image

4. ตาราง UserAdmin

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	AdminID	Number	หมายเลขของผู้ควบคุมระบบเป็น P.K. Not null
2	DfirstName	varchar2(20)	ชื่อของผู้ควบคุมระบบ
3	DlastName	varchar2(10)	นามสกุลของผู้ควบคุมระบบ
4	DuserName	varchar2(10)	UserName ของผู้ควบคุมระบบ
5	Dpassword	varchar2(8)	Password ของผู้ควบคุมระบบ
6	AdminStatus	varchar2(10)	สถานะของผู้ควบคุมระบบ
7	Email	varchar2(30)	E-mail ของผู้ควบคุมระบบ

ตารางที่ 8-4 แสดงตารางที่ใช้เก็บข้อมูลของผู้ควบคุมระบบที่ชื่อว่า UserAdmin

5. ตาราง Reservation

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	CustomerID	Number	หมายเลขของลูกค้าเป็น P.K. Not null
2	HomeID	Number	หมายเลขบ้านที่ลูกค้าจอง
3	ProjectID	Number	หมายเลขโครงการที่ลูกค้าจอง
4	IndexImage	Number	หมายเลขของภาพ
5	ReservDate	varchar2(30)	วันที่การจอง

ตารางที่ 8-5 แสดงตารางที่ใช้เก็บข้อมูลของการจองที่ชื่อว่า Reservation

6. ตาราง Vendor

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	VendorID	Number	หมายเลขของผู้ขายฝากเป็น P.K. Not null
2	VfirstName	varchar2(20)	ชื่อของผู้ขายฝาก
3	VLastName	varchar2(20)	นามสกุลของผู้ขายฝาก
4	Address	varchar2(150)	ที่อยู่ของผู้ขายฝาก
5	VTelephone	varchar2(30)	เบอร์โทรศัพท์ของผู้ขายฝาก

ตารางที่ 8-6 แสดงตารางที่ใช้เก็บข้อมูลของผู้ฝากขายที่ชื่อว่า Vendor

7. ตาราง Contact_Ask

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	ConID	Number	หมายเลขของการสอบถามเป็น P.K. Not null
2	CustomerID	Number	หมายเลขของลูกค้าเป็น F.K. Not null
3	Head	varchar2(100)	หัวข้อคำถามลูกค้า
4	Message	varchar2(2000)	ข้อความของลูกค้า
5	Email	varchar2(30)	อีเมล์ของผู้ลูกค้าที่ต้องให้ส่งคำตอบไปให้

ตารางที่ 8-7 แสดงตารางที่ใช้เก็บข้อมูลของการสอบถามที่ชื่อว่า Contact_Ask

8. ตาราง Home

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	HomeID	Number	เลขที่ของบ้านเป็น P.K. Not Null
2	HomeNumber	varchar2(10)	บ้านเลขที่
3	HAmphur	varchar2(20)	อำเภอ
4	HProvince	varchar2(20)	จังหวัด
5	Area	varchar2(20)	พื้นที่ขนาดของบ้าน (ตารางวา)
6	Floor	varchar2(10)	จำนวนชั้น
7	BedRoom	varchar2(10)	จำนวนห้องนอน
8	KitchenRoom	varchar2(10)	จำนวนห้องครัว
9	bathRoom	varchar2(10)	จำนวนห้องน้ำ
10	Price	Number	ราคาของบ้าน
11	XPosition	Number	ตำแหน่ง X ของบ้านในแผนภาพ
12	YPosition	Number	ตำแหน่ง Y ของบ้านในแผนภาพ
13	StatusReserv	varchar2(10)	สถานะการจอง
14	HomeType	varchar2(20)	ชนิดของบ้าน
15	GroupHome	varchar2(10)	กลุ่มของบ้าน
16	HomeImage	varchar2(20)	รูปภาพของบ้าน
17	ProjectID	Number	เลขที่ของโครงการเป็น F.K Not Null
18	IndexImage	Number	หมายเลขของภาพเป็น F.K. Not Null
19	Ven	Number	หมายเลขของผู้ฝากขายเป็น F.K. Not Null

ตารางที่ 8-8 แสดงตารางที่ใช้เก็บข้อมูลของการจองที่ชื่อว่า Home

9. ตาราง HomeTypeDetail

ลำดับ	คอลัมน์	ชนิดข้อมูล	อธิบาย
1	ProjectID	Number	เลขที่ของโครงการเป็น F.K Not Null
2	IndexImage	Number	หมายเลขของภาพเป็น F.K. Not Null
3	HomeImage	varchar2(20)	รูปภาพของบ้าน
4	HomeDetail	varchar2(4000)	เป็นข้อความประกาศหรือข้อความโฆษณาบ้านใหม่

ตารางที่ 8-9 แสดงตารางที่ใช้เก็บข้อมูลชนิดของบ้านที่ชื่อว่า HomeTypeDetail

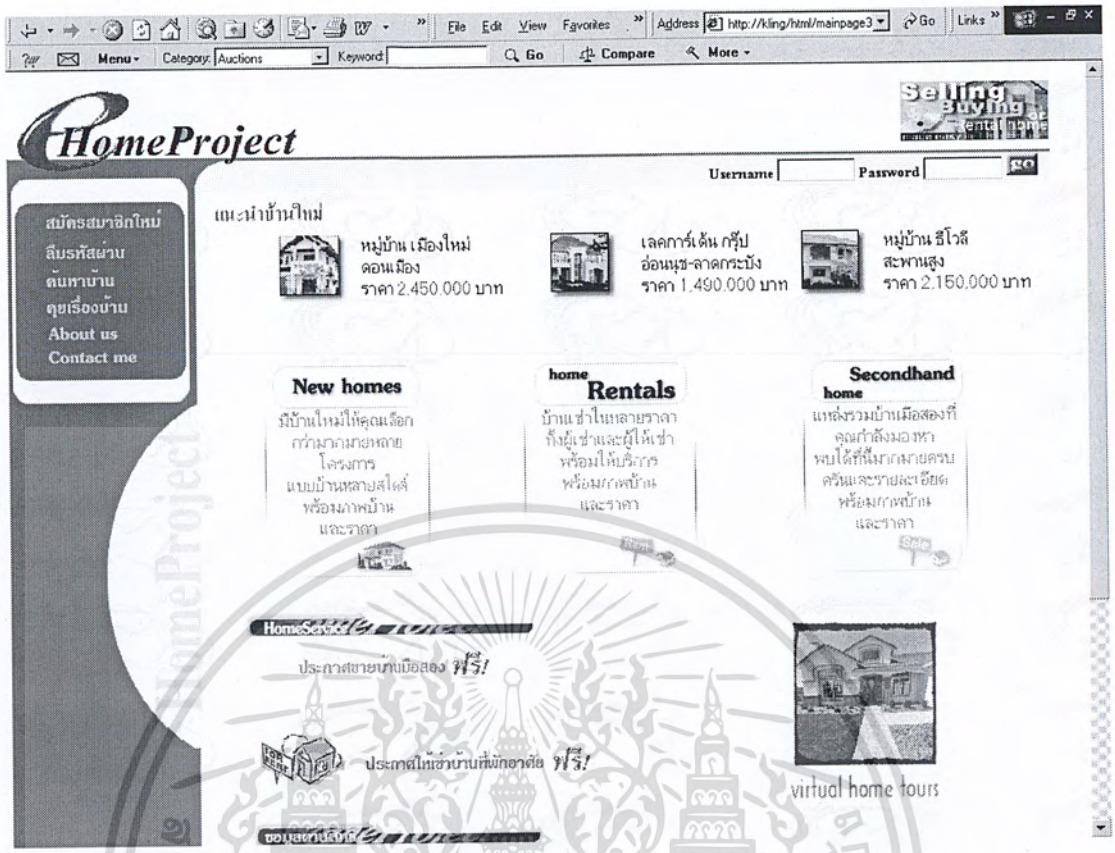
บทที่ 9

การทดสอบและผลการทดลอง

9.1 การทดสอบหน้าจอและการทำงานของโปรแกรม

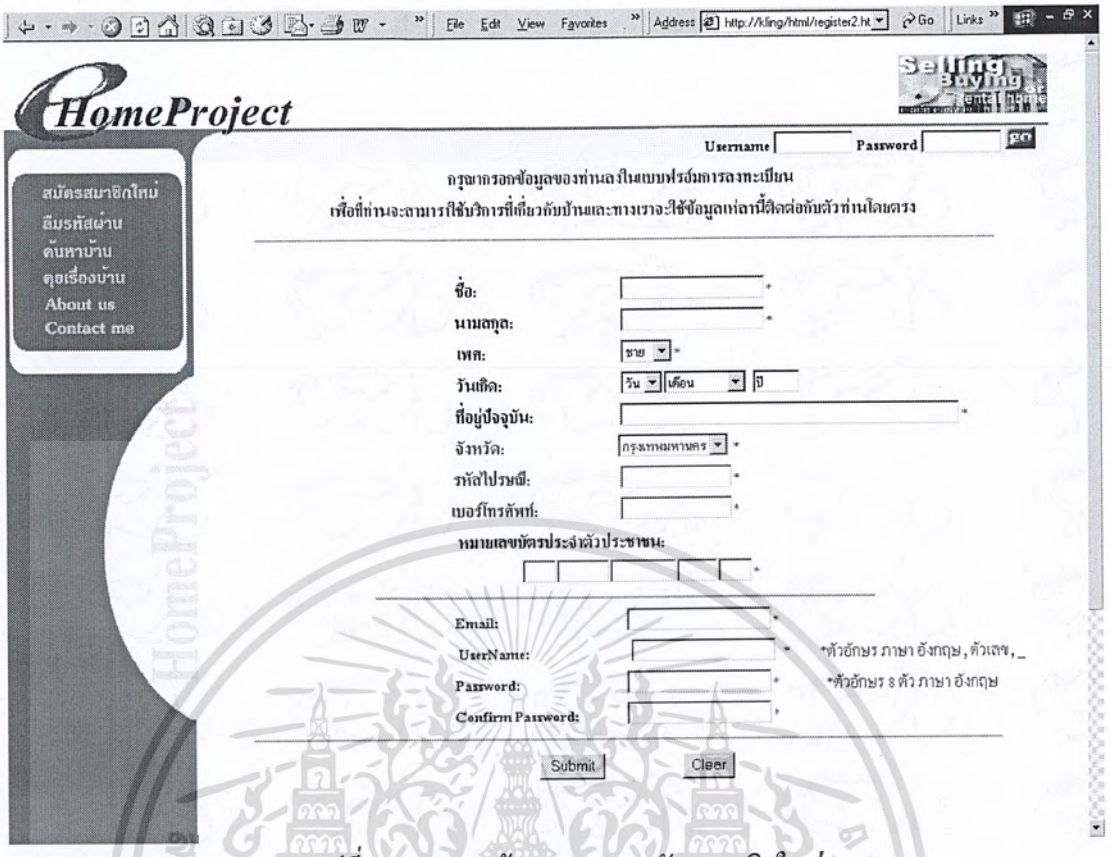
หลังจากที่ทำการออกแบบหน้าจอ ,แก้ไข เพิ่มเติมโค้ดและ สร้างคลาสที่ใช้ใน ระบบทั้งหมดเสร็จเรียบร้อยแล้ว เราก็จะได้การทำงานของระบบการจองบ้านออนไลน์ออกมา โดยในแต่ละหน้าจอของการทำงาน เราได้มีการทดสอบการทำงานของหน้าจอ หรือว่าคลาสนั้นๆ เป็นที่เรียบร้อยแล้วว่าสามารถทำงานได้ถูกต้อง และ ไม่มีข้อผิดพลาด เราก็จะทำการทดสอบระบบรวม โดยนำคลาสหรือหน้าจอต่างๆ มาใช้งานร่วมกันโดยการเรียกผ่านเครื่องไคลเอนต์ เพื่อเรียกใช้งาน Servlet ผ่านตัว เซิร์ฟเวอร์ เพื่อเป็นการรับรองว่าเมื่อนำคลาสต่างๆมารวมกันแล้ว จะไม่เกิดข้อผิดพลาดขึ้น โดยทั่วไปแล้วการออกแบบโดยใช้ ออบเจ็กต์โอเรียนเต็ด ทั้งการออกแบบและการเขียนโปรแกรม เมื่อนำคลาสหรือ หน้าจอการทำงานต่างๆ มารวมกันหรือว่าใช้งานรวมกันแล้ว ก็มักจะไม่มีปัญหาเกิดขึ้น เพราะแต่ละคลาสนั้นมีแยกตัวแปรที่ชัดเจน ไม่มีการใช้ตัวแปรร่วมกัน ซึ่งเป็นข้อดีข้อหนึ่งของ ออบเจ็กต์-โอเรียนเต็ด ต่อไปก็จะแสดงหน้าจอตัวอย่างของการทำงานหลักๆของโปรแกรม ดังนี้

- หน้าจอหลัก
- หน้าจอของการสมัครสมาชิกใหม่
- หน้าจอแสดงรายการบ้านใหม่
- หน้าจอแสดงรายการบ้านเช่า
- หน้าจอแสดงรายการบ้านมือสอง
- หน้าจอแสดงรายละเอียดของโครงการและบ้านใหม่
- หน้าจอแสดงการจองบ้าน
- หน้าจอของการค้นหาบ้าน
- หน้าจอรายการการค้นหาบ้านที่ได้
- หน้าจอของรายละเอียดบ้านทั่วไป
- หน้าจอของแบบฟอร์มประกาศขายบ้านมือสอง
- หน้าจอของการแก้ไขข้อมูลโครงการ,ข้อมูลบ้าน และ แผนผังโครงการ
- หน้าจอของการเพิ่มและแก้ไขรายละเอียดต่างๆ ของบ้านใหม่



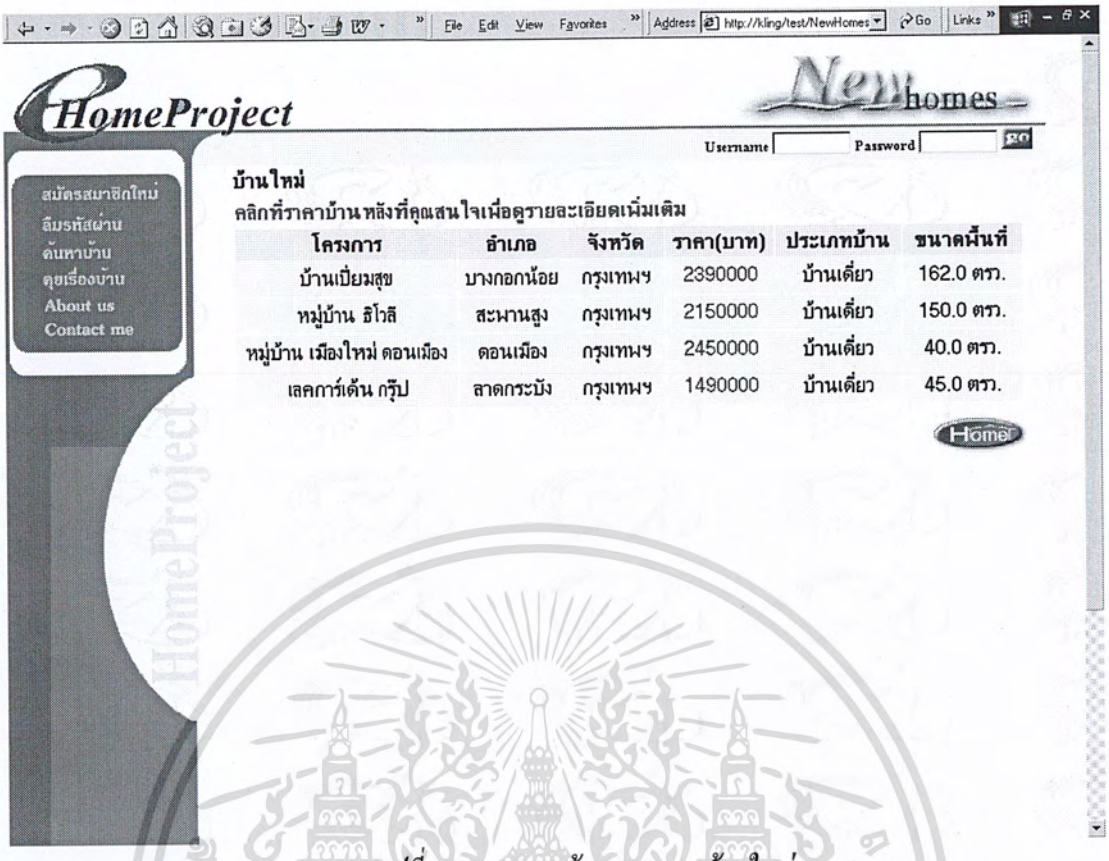
รูปที่ 9-1 แสดงหน้าจอหลัก

จากรูปที่ 9-1 แสดงหน้าจอหลักโดยผู้ใช้จะเข้ามาสู่หน้าแรกของเว็บเพจ ในหน้าหลักนี้ประกอบด้วยฟังก์ชันหลักๆ 6 ฟังก์ชัน คือ สมัครสมาชิกใหม่, ลืมรหัสผ่าน, ค้นหาบ้าน, คุยเรื่องบ้าน, About us ,Contact me และ ส่วนประกอบอื่นๆ ส่วนล็อกอินเพื่อเข้าสู่ระบบ ส่วนแนะนำบ้านใหม่ ส่วนแสดงรายการบ้านใหม่ บ้านเช่า บ้านมือสอง ส่วนบริการเกี่ยวกับบ้าน เช่น การประกาศ ขาย/เช่า บ้าน และส่วนที่เป็นสาระความรู้ข้อมูลด้านสิทธิ ในส่วนต่างๆ ที่กล่าวมานั้นก็มีการลิงก์ไปยังส่วนที่แสดงผลรายละเอียดต่อไป



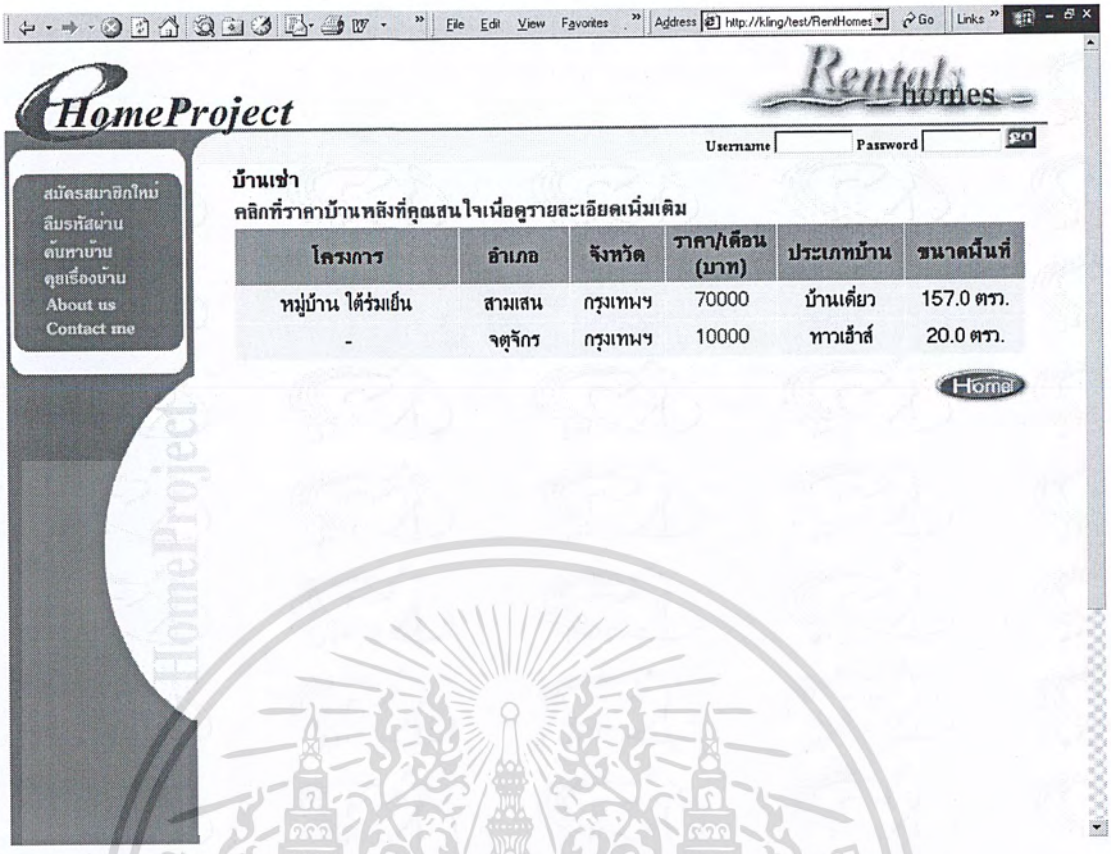
รูปที่ 9-2 แสดงหน้าจอของการสมัครสมาชิกใหม่

จากรูปที่ 9-2 แสดงหน้าจอของการสมัครสมาชิกใหม่ ถ้าต้องใช้บริการและสิทธิพิเศษของเว็บไซต์นี้ผู้ใช้ก็ต้องทำการลงทะเบียนกับระบบก่อน โดยการเลือกไปที่เมนู "สมัครสมาชิกใหม่" ก็จะปรากฏแบบฟอร์มสำหรับการลงทะเบียนซึ่งจะมีรายละเอียดที่เกี่ยวกับผู้ใช้ให้ทำการกรอกลงไป และ กำหนด Username และ Password เพื่อที่ใช้ในการเข้าสู่ระบบ ให้ลูกค้าทำการกรอกให้ครบ แล้วจึงกดปุ่ม submit ก็เป็นการเสร็จเรียบร้อย ข้อมูลต่างๆ ของลูกค้าก็จะถูกจัดเก็บลงฐานมูลของระบบ



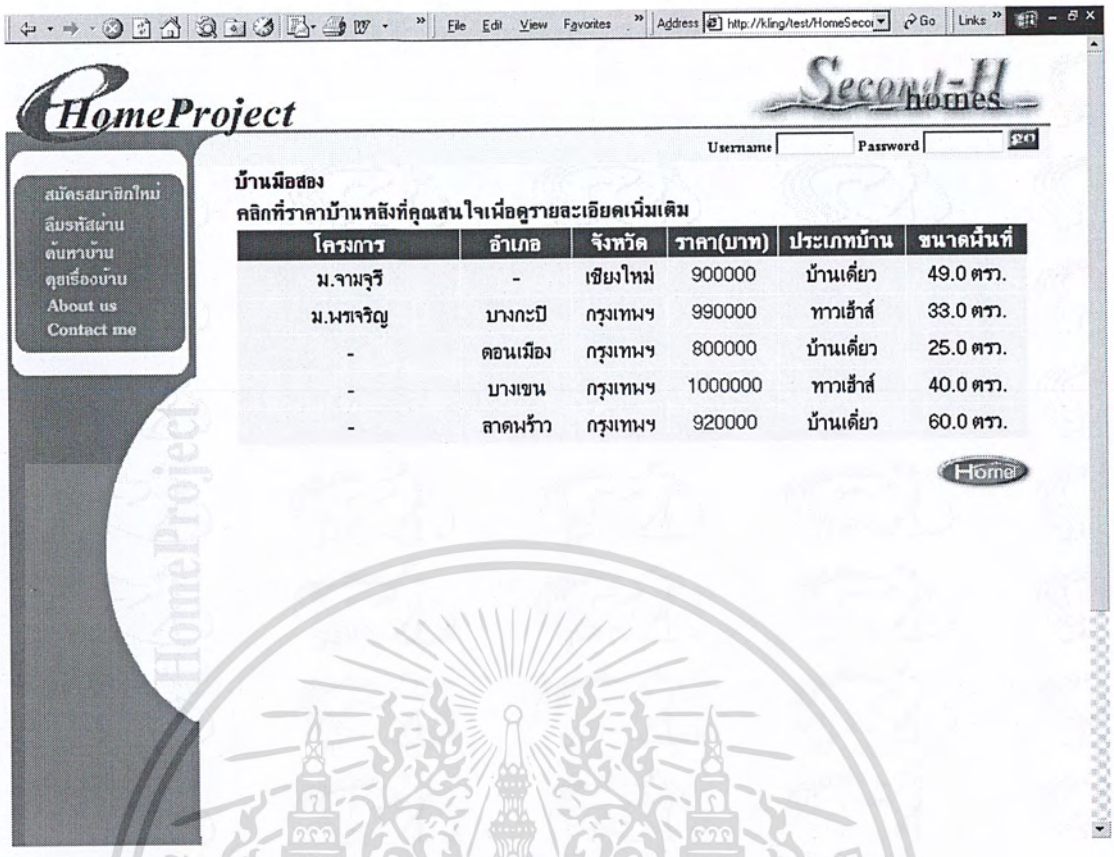
รูปที่ 9-3 แสดงหน้าจอรายการบ้านใหม่

จากรูปที่ 9-3 แสดงหน้าจอรายการบ้านใหม่ โดยเลือกไปกดที่ New Homes ก็จะเข้ามาสู่หน้าจอของการแสดงรายการของบ้านใหม่ทั้งหมดที่มี และสามารถเลือกดูรายละเอียดต่างๆ ได้อีกโดยทำการคลิ๊กที่ราคาของ รายการโครงการนั้นๆ ก็จะสามารถแสดงรายละเอียดของ บ้านใหม่ที่เลือกได้อีกด้วย



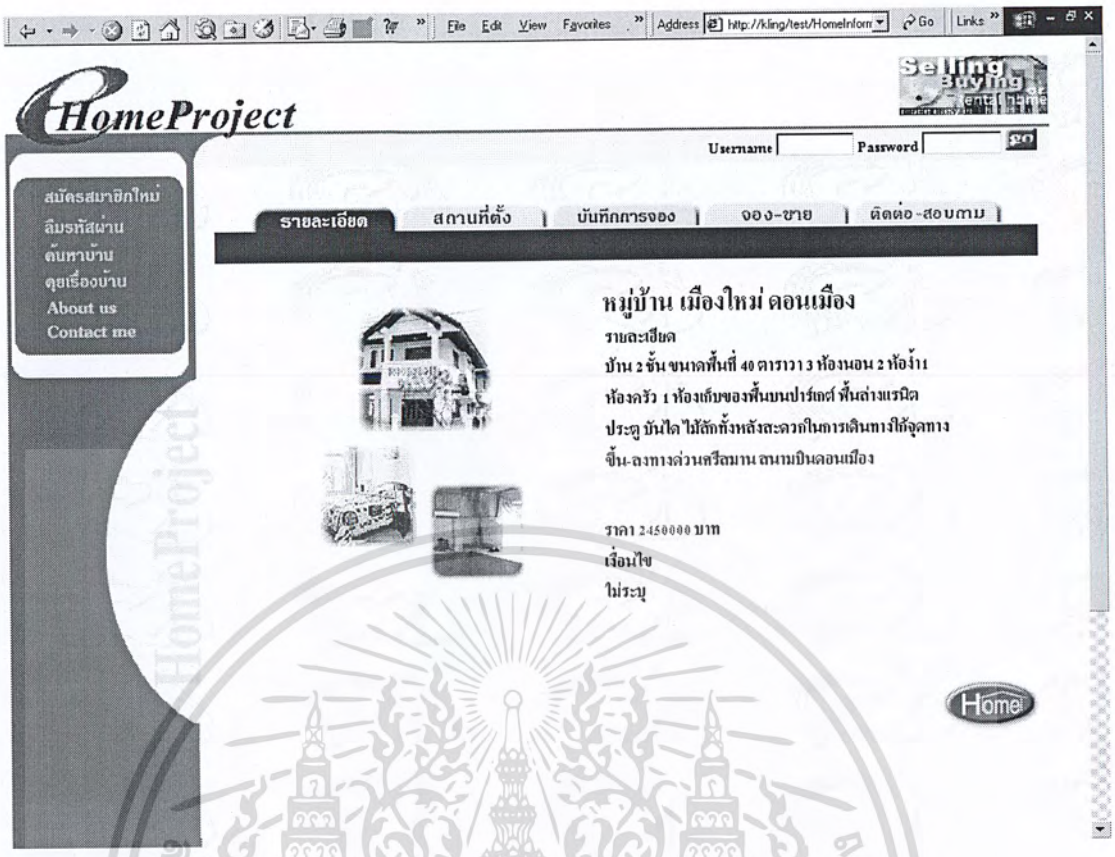
รูปที่ 9-4 แสดงหน้าจอรายการบ้านเช่า

จากรูปที่ 9-4 แสดงหน้าจอรายการบ้านเช่า โดยเลือกไปกดที่ home rentals ก็จะเข้ามาสู่หน้าจอของการแสดงรายการของบ้านเช่าทั้งหมดที่มี และสามารถเลือกดูรายละเอียดต่างๆ ได้อีกโดยทำการคลิ๊กที่ราคาของรายการโครงการนั้นๆ ก็จะสามารถแสดงรายละเอียดของบ้านเช่าที่เลือกได้อีกด้วย



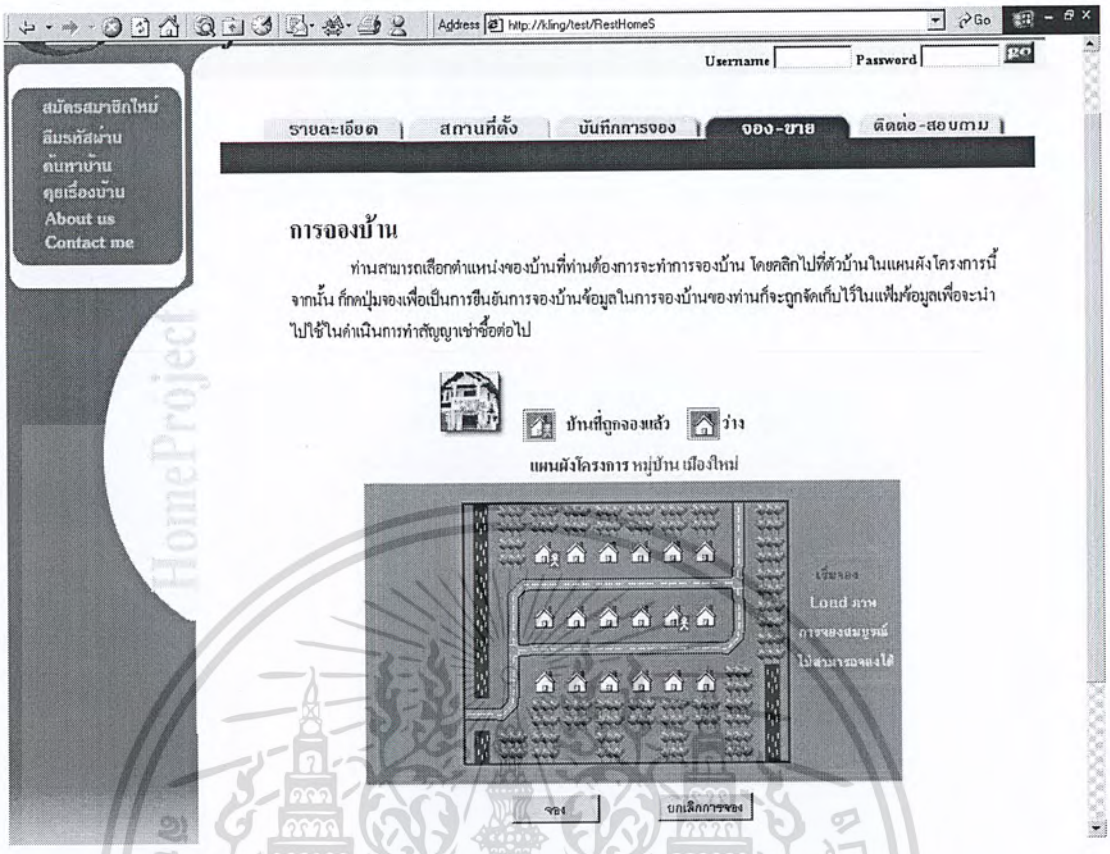
รูปที่ 9-5 แสดงหน้าจอรายการบ้านมือสอง

จากรูปที่ 9-5 แสดงหน้าจอรายการบ้านมือสอง โดยเลือกไปกดที่ SecondHamd Home ก็จะเข้ามาสู่หน้าจอของการแสดงรายการของบ้านมือสองทั้งหมดที่มี และสามารถเลือกดูรายละเอียดต่างๆ ได้อีก โดยทำการคลิ๊กที่ราคาของรายการโครงการนั้นๆ ก็จะสามารถแสดงรายละเอียดของบ้านมือสองที่เลือกได้อีกด้วย



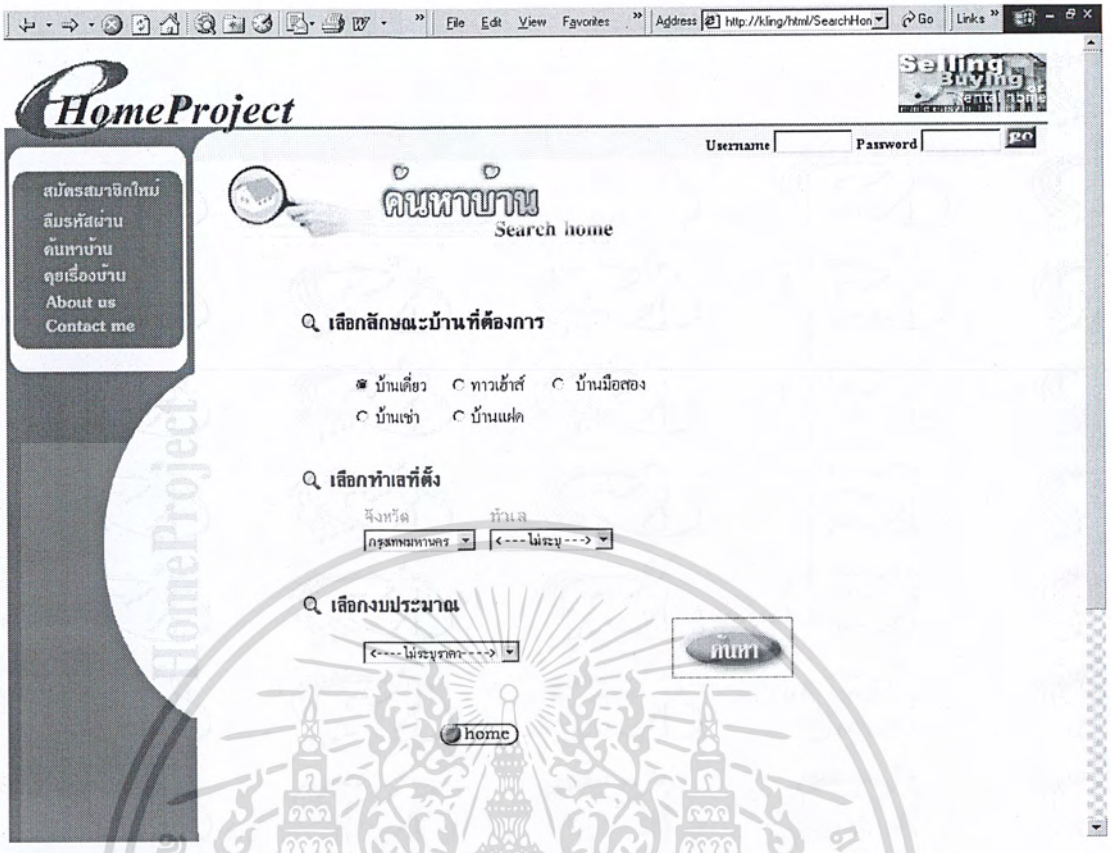
รูปที่ 9-6 แสดงหน้าจอแสดงรายละเอียดของโครงการและบ้านใหม่

จากรูปที่ 9-6 แสดงหน้าจอแสดงรายละเอียดของโครงการและบ้านใหม่ ซึ่งจะประกอบไปด้วยส่วนที่แสดงรายละเอียดของโครงการ มีรูปภาพประกอบข้อมูลบ้านราคาของบ้านและเงื่อนไขของโครงการ, สถานที่ตั้งโครงการก็จะแสดงรูปภาพแผนที่ตั้งของโครงการนั้น, บันทึกการจองก็จะแสดงรายการของโครงการต่างๆและรายละเอียดเกี่ยวกับบ้านที่ลูกค้าได้ทำการจองไปแล้ว, จอง-ขายแสดงหน้าสำหรับทำการจอง และติดต่อสอบถามจะแสดงที่อยู่เบอร์โทรศัพท์ที่จะสามารถใช้ในการติดต่อกับโครงการบ้านนั้นๆและยังมีแบบฟอร์มที่ใช้กรอกสำหรับการสอบถามข้อมูลของโครงการ ผู้ที่เข้ามาเยี่ยมชมสามารถเข้ามาที่หน้าเพจนี้ได้โดยการคลิ๊กที่รูปบ้านที่หน้าหลัก หรือ ส่วนแสดงรายการของบ้านใหม่ใน New Homes ก็จะสามารถเข้ามาอ่านรายละเอียดของบ้านได้ โดยไม่ต้องทำการล็อกอินก่อนก็ได้



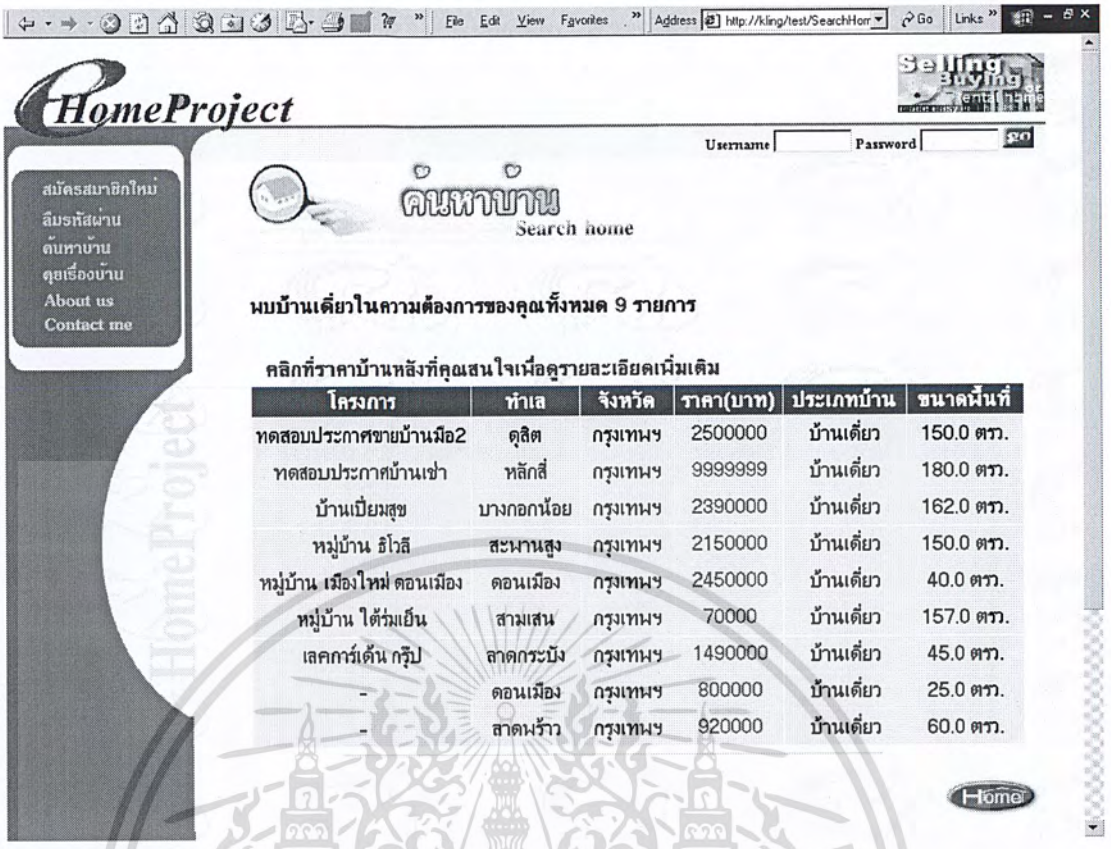
รูปที่ 9-7 แสดงหน้าจอของการจองบ้าน

จากรูปที่ 9-7 แสดงหน้าจอของการจองบ้าน การจะเข้ามาสู่ในหน้าของการจองได้นั้นจะต้องผ่านการล็อกอินเพื่อเข้าสู่ระบบเสียก่อน และเมื่อเข้ามาสู่ในหน้าของการจองแล้ว ถ้าลูกค้าที่ยังไม่เคยทำการจองบ้านในโครงการที่เข้ามาก่อนเลย สถานะของการจองของลูกค้าก็จะอยู่ในสถานะที่สามารถทำการจองได้ โดยลูกค้าสามารถที่จะเลือกยังตำแหน่งของบ้านในโครงการนั้นที่สนใจ ภายในแแผนผังโครงการนั้นจะมีสัญลักษณ์ของบ้านอยู่สองชนิด คือ บ้านที่ถูกจองแล้วกับบ้านที่ว่างอยู่ เมื่อเลือกบ้านในตำแหน่งใดที่ว่างสัญลักษณ์ของบ้านก็จะเปลี่ยนเป็นถูกจองแต่การจองจะยังไม่สมบรูณ์จนกว่าจะกดปุ่มจองเพื่อยืนยันอีกครั้งหนึ่ง เมื่อกดปุ่มจองข้อมูลต่างๆของลูกค้าและข้อมูลที่เกี่ยวกับบ้านหลังนั้นก็จะถูกจัดเก็บลงฐานข้อมูลเป็นบันทึกการจองของลูกค้า และในขณะเวลาเดียวกันกับที่ลูกค้าทำการจองบ้านไปแล้วก็สามารถที่จะทำการยกเลิกการจองในขณะนั้นได้ เมื่อเสร็จสิ้นการจอง สถานะการจองก็จะอยู่ที่การจองสมบูรณ์



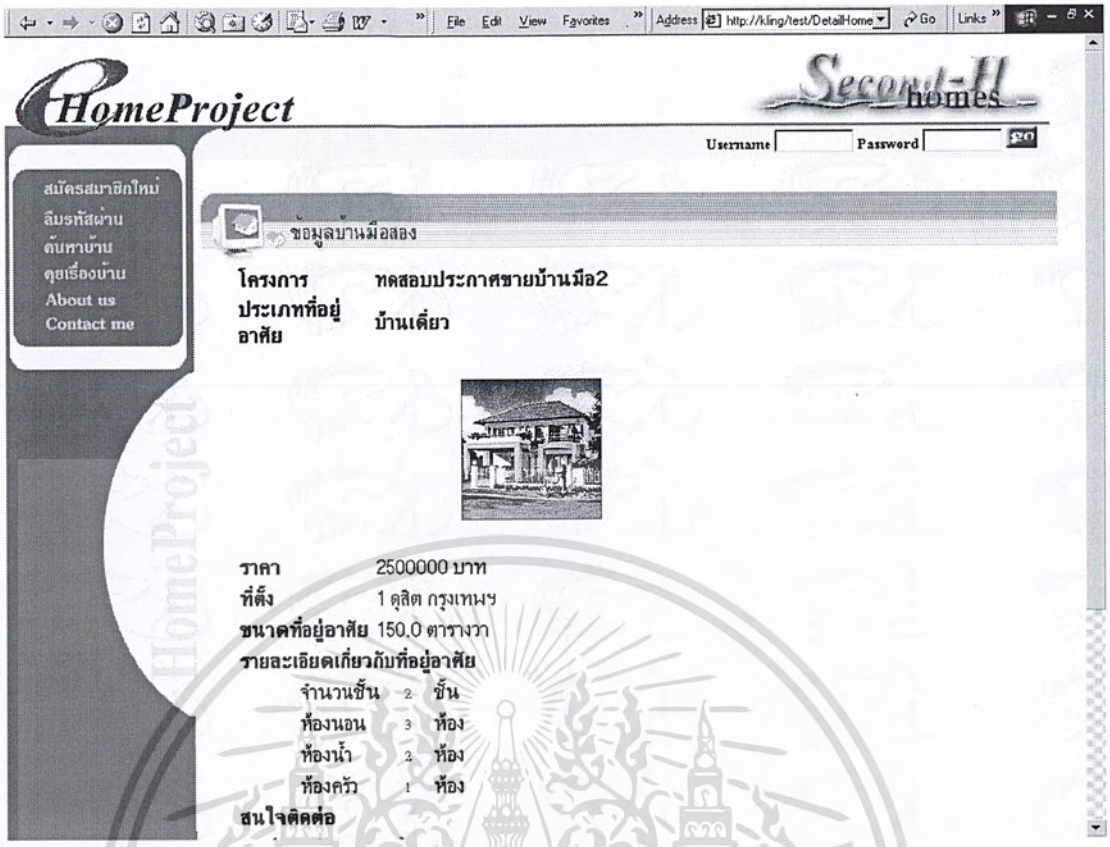
รูปที่ 9-8 แสดงหน้าจอของการค้นหาบ้าน

จากรูปที่ 9-8 แสดงหน้าจอของการค้นหาบ้าน ผู้ใช้สามารถที่จะลือกเงื่อนไขสำหรับการค้นหาบ้านที่ต้องการได้ โดยมีเงื่อนไขอยู่ 3 ลักษณะ คือ เลือกลักษณะบ้านที่ต้องการ เลือกทำเลที่ตั้ง เลือกงบประมาณ เมื่อเลือกได้ตามต้องการแล้วก็กดปุ่ม “ค้นหาบ้าน” ระบบก็จะนำเงื่อนไขที่ผู้ใช้กำหนดไปทำการค้นข้อมูลที่อยู่ในฐานข้อมูล แล้วก็จะแสดงผลลัพธ์ออกดังจะแสดงในรูปถัดไป



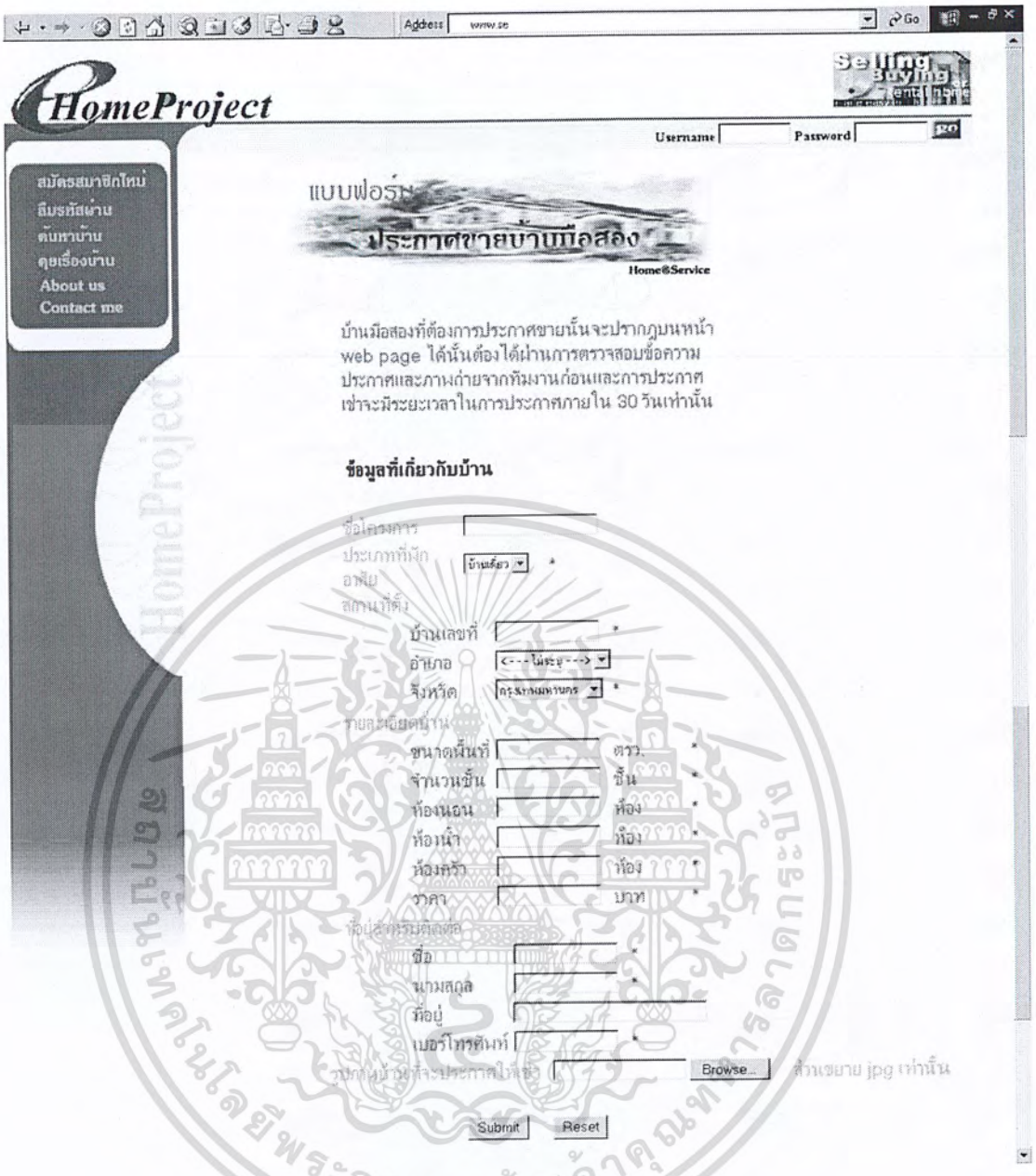
รูปที่ 9-9 แสดงหน้าจอของรายการการค้นหาบ้านที่ได้

จากรูปที่ 9-9 แสดงหน้าจอรายการการค้นหาบ้านที่ได้ จากการกำหนดเงื่อนไขในการค้นหาบ้านแล้วทำการค้นหา ก็จะได้ผลลัพธ์จากการค้น โดยจะแสดงออกมาเป็นรายการของโครงการบ้านต่างๆ ดังรูป ซึ่งผู้ใช้สามารถที่เข้าไปดูรายละเอียดเพิ่มเติมที่เกียวกับบ้านในโครงการนั้นๆ ได้โดยคลิ๊กที่ราคาของของบ้านในโครงการ ก็จะแสดงรายละเอียดของบ้านออกมา



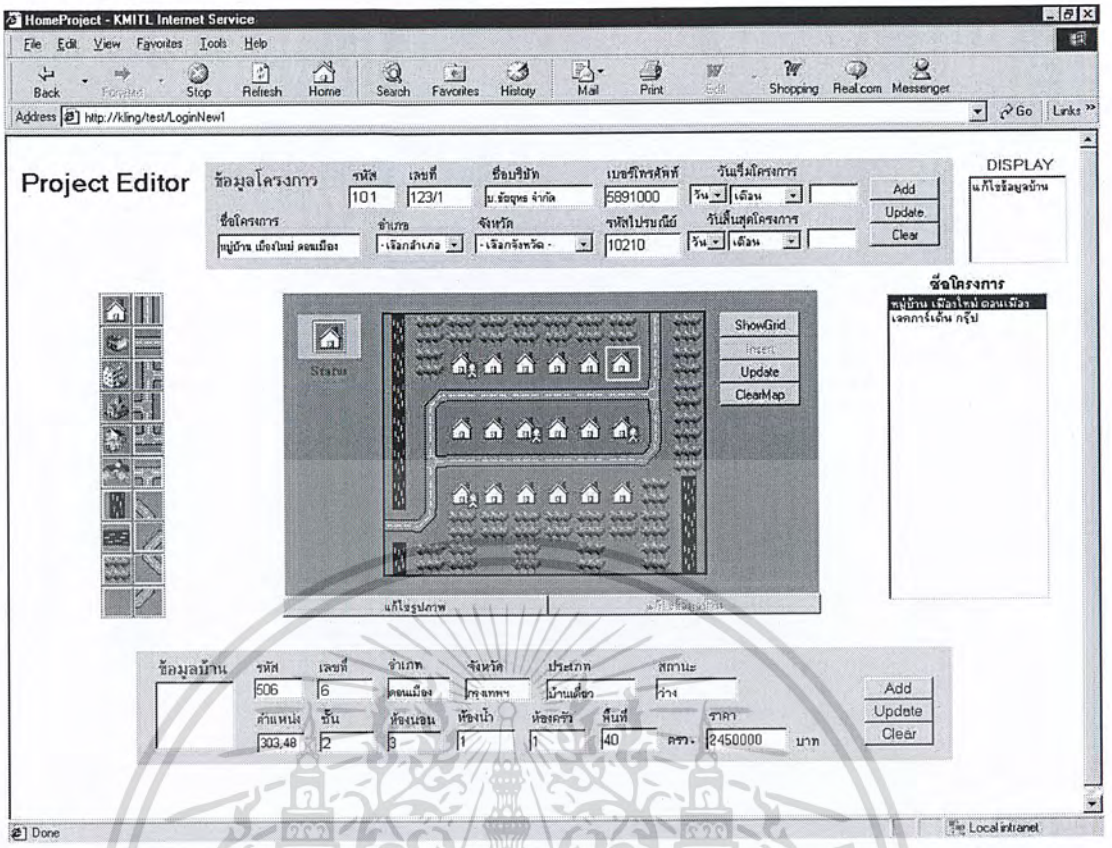
รูปที่ 9-10 แสดงหน้าจอของรายละเอียดบ้านทั่วไป

จากรูปที่ 9-10 แสดงหน้าจอรายละเอียดบ้านทั่วไป ซึ่งจะมีด้วยกัน 2 ลักษณะ คือ รายละเอียดของบ้านมือสองดังรูปและรายละเอียดของบ้านเช่าซึ่งจะมีลักษณะการแสดงรายละเอียดที่เหมือนกัน การที่จะเข้ามาสู่ในหน้าของการแสดงรายละเอียดของบ้านทั้ง 2 ลักษณะนี้ สารามถที่จะเข้ามาจากการเลือกราคาบ้านที่อยู่ในส่วนแสดงรายการที่ได้จากการค้นหาบ้าน หรือ จากในส่วนแสดงรายการบ้านเช่า Home Rentals และ ส่วนแสดงรายการบ้านมือสอง Home Secondhand ที่อยู่ในส่วนของหน้าหลัก



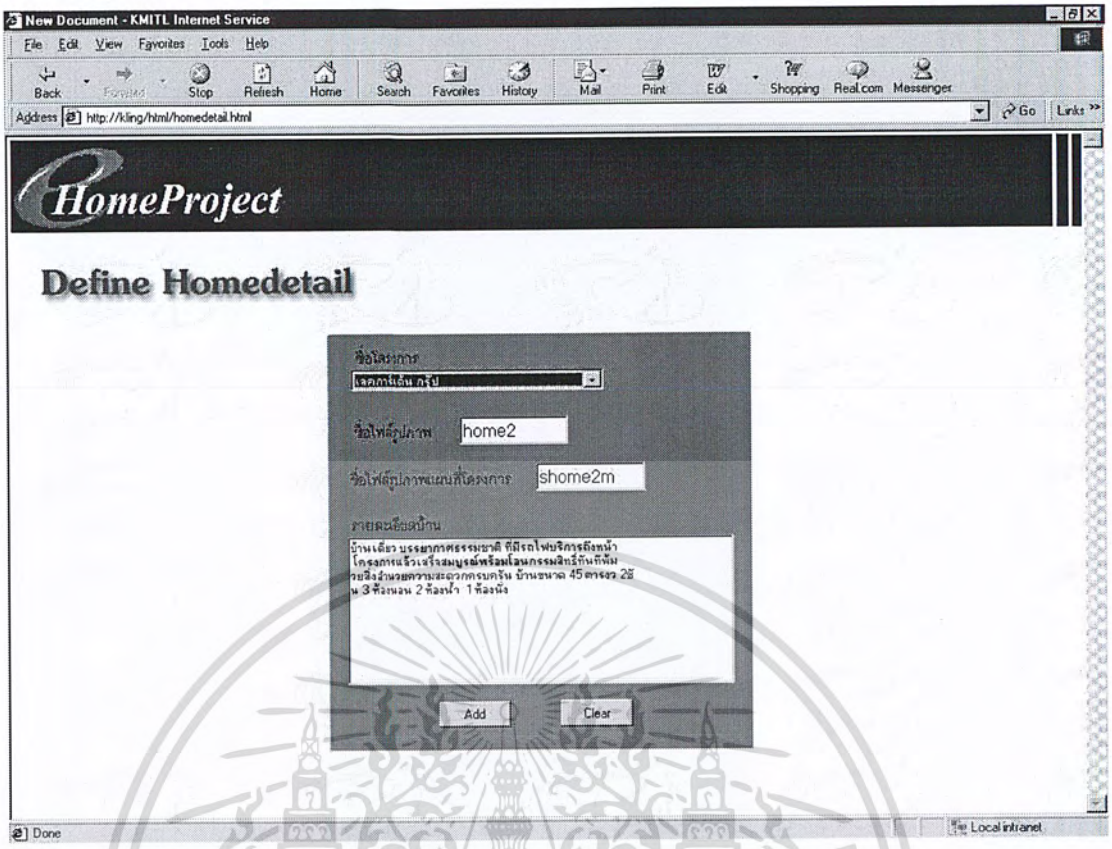
รูปที่ 9-11 แสดงหน้าจอแบบฟอร์มประกาศขายบ้านมือสอง

จากรูปที่ 9-11 แสดงหน้าจอแบบฟอร์มประกาศขายบ้านมือสอง (หน้าจอแบบฟอร์มประกาศให้เช่าที่พักอาศัยมีลักษณะเหมือนกัน) การที่จะเข้ามาสู่หน้านี้ได้ลูกค้าต้องผ่านการล็อกอินมาก่อน แบบฟอร์มประกาศขายบ้านมือสองจะช่องให้ใส่รายละเอียดที่เกี่ยวกับบ้านที่จะประกาศขาย พร้อมกับให้ใส่ไฟล์รูปภาพบ้านที่เป็นส่วนขยายเป็น JPG เท่านั้น เมื่อใส่รายละเอียดครบหมดแล้วก็กดปุ่ม Submit ข้อมูลทั้งหมดจากรายการจะถูกส่งไปเก็บลงฐานข้อมูล และสามารถตรวจสอบการประกาศขายบ้านได้ โดยเข้าไปที่ส่วนของ Secondhand homes ในหน้าจอหลักของเว็บไซต์ ก็จะแสดงรายการของบ้านมือสองที่เราประกาศขายออกมา



รูปที่ 9-12 แสดงหน้าจอของการแก้ไขและเพิ่มข้อมูลโครงการ,ข้อมูลบ้าน,แผนผังโครงการ

จากรูปที่ 9-12 แสดงหน้าจอของการแก้ไขและเพิ่มข้อมูลโครงการ,ข้อมูลบ้าน และแผนผังโครงการ โดยทำการใส่ username และ password ของผู้ดูแลระบบก็จะเข้าสู่หน้าจอนี้ ถ้าต้องการแก้ไขข้อมูลโครงการที่มีอยู่แล้วก็ดับเบิ้ลคลิกที่ชื่อโครงการในลิส ก็จะปรากฏข้อมูลโครงการขึ้นมาที่ส่วนข้อมูลโครงการและรูปภาพแผนผังโครงการให้ทำการแก้ไขต่อไปแล้วกดปุ่ม update แต่ถ้าต้องการเพิ่มโครงการใหม่ก็เพียงแต่กดปุ่ม Clear ในทุกส่วน แล้วก็ใส่ข้อมูลโครงการแล้วสร้างภาพแผนผังโครงการมาใหม่แล้วกดปุ่ม insert และถ้าจะใส่ข้อมูลบ้านก็กดปุ่มแก้ไขข้อมูลบ้านเพื่อเปลี่ยนโหมด คลิ๊กที่รูปบ้านในแผนผังโครงการ ใส่ข้อมูลบ้าน กด Add ข้อมูลที่ได้จากการแก้ไขหรือเพิ่มเติมก็จะถูกจัดเก็บลงฐานข้อมูล



รูปที่ 9-13 แสดงหน้าจอของการเพิ่มและแก้ไขรายละเอียดต่างๆ ของบ้านใหม่

จากรูปที่ 9-13 แสดงหน้าจอของการเพิ่มและแก้ไขรายละเอียดต่างๆ ของบ้านใหม่ ในส่วนของชื่อโครงการจะถูกดึงออกมาจากฐานข้อมูลโดยอัตโนมัติเมื่อเข้ามาสู่หน้านี้ และชื่อโครงการจะเป็นโครงการของบ้านใหม่ทั้งหมด ถ้าเลือกชื่อโครงการใดโครงการหนึ่งก็จะปรากฏรายละเอียดต่างๆ ที่เกี่ยวกับบ้านและโครงการออกมา ซึ่งเราสามารถที่ลบหรือทำการแก้ไขรายละเอียดต่างๆ ได้แล้วกดปุ่ม Add ข้อมูลต่าง ๆก็จะถูกอัพเดทลงฐานข้อมูล

บทที่ 10
บทสรุปและวิจารณ์

10.1 บทสรุปและวิจารณ์

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุ หรือ ออปเจ็กต์-โอเรียนเต็ด แอพพลิเคชัน เราได้พัฒนาเพื่อสร้างระบบจัดการฐานข้อมูลโครงการจองบ้านผ่านเว็บไซต์ ให้มีความสะดวก ถูกต้อง และรวดเร็วมากขึ้นในการดูรายละเอียดของบ้าน การค้นหาบ้าน การจองบ้าน การประกาศ ขาย/เช่า รวมไปถึงการเพิ่มส่วนที่ช่วยในการเพิ่ม และ เพื่อให้ง่ายต่อการจัดการฐานข้อมูล ในการพัฒนาระบบเราได้มีการนำเครื่องมือหลายตัวเข้ามาช่วยในการสร้างระบบ ได้แก่

- โปรแกรม Rational Rose เป็น CASE tool ที่มีการนำเสนอโมเดลในรูปแบบของสัญลักษณ์ทำให้การวิเคราะห์และออดแบบระบบดูเป็นมาตราฐานและเป็นระบบที่ดูดีขึ้น
- โปรแกรมออราเคิล เป็นโปรแกรมระบบจดการฐานข้อมูล ที่ช่วยในการสร้างตารางฐานข้อมูล แบบรีเลชันแนล ที่มีประสิทธิภาพในการจัดฐานข้อมูลสูง
- โปรแกรม OAS เป็นเว็บเซิร์ฟเวอร์ที่คอยให้บริการของระบบแก่ผู้ใช้ทางอินเตอร์เน็ต ซึ่งสนับสนุนการทำงานของ Servlet Engine
- โปรแกรม Broland J Builder 3.0 เป็นเครื่องมือที่ช่วยในการเขียนโปรแกรม ภาษาจาวา ซึ่งเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ

10.2 แนวทางในการพัฒนาเพิ่มเติม

- เพิ่มส่วนที่เป็นข้อมูลเกี่ยวกับอหังสาริมทรัพย์ที่ครบวงจรและสามารถที่จะทำสัญญาซื้อขายผ่านระบบอินเตอร์เน็ตได้
- เพิ่มส่วนที่สามารถแก้ไขข้อมูลลูกค้าโดยให้ลูกค้าสามารถที่จะแก้ไขข้อมูลของตนเองได้
- การแสดงรูปตัวอย่างบ้านน่าจะสามารถมองได้หลายมุม โดยสามารถหมุนภาพมองได้ 360°
- ในส่วนภาพแผนผังโครงการควรมีการขยายขนาดได้และ รูปแบบควรมีหลายๆ แบบ

ภาคผนวก

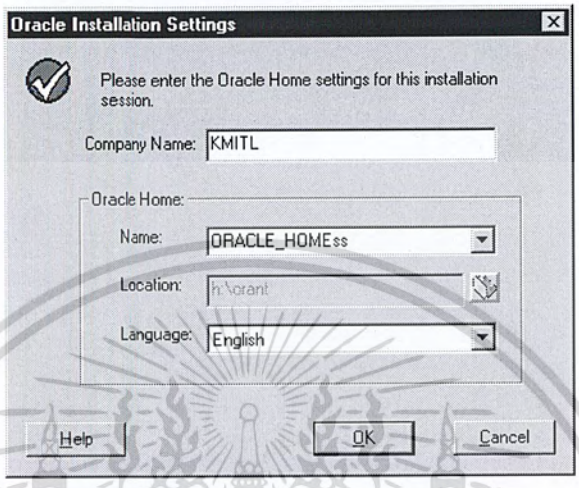
ภาคผนวก ก

การติดตั้ง OAS(Oracle Application Server)

1. ขั้นตอนการติดตั้ง OAS (Oracle Application Server)

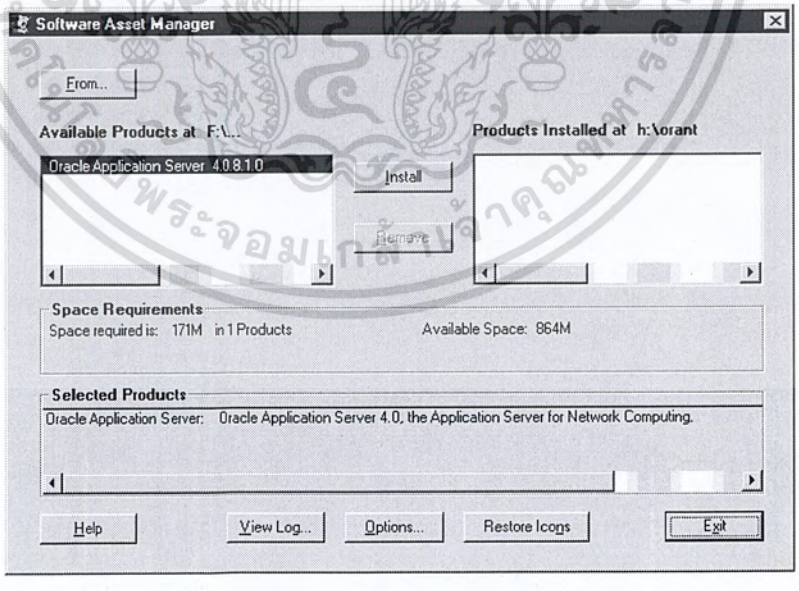
1. ติดตั้ง Windows NT 4.0 Server โดยใช้ชื่อเครื่องว่า kling

2. Run Setup จากแผ่น CD จากนั้นกำหนด Directory ,Name และ ภาษา ดังแสดงในรูปที่ 1



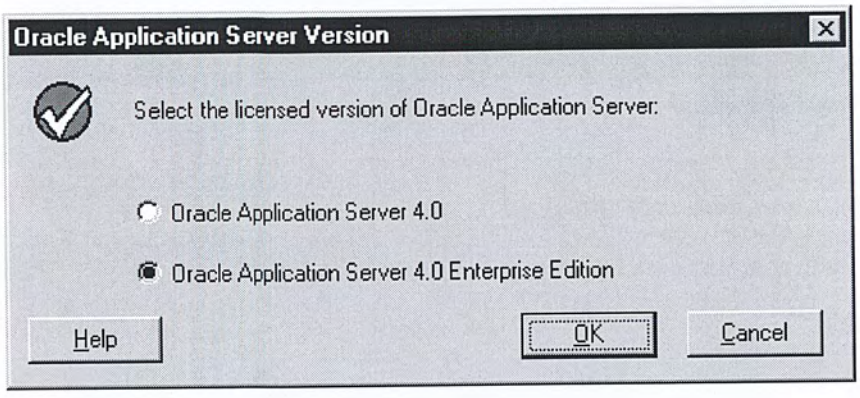
รูปที่ 1 แสดงการกำหนด Directory

3. เลือก Product Oracle Application Server 4.0 แล้วกด Install



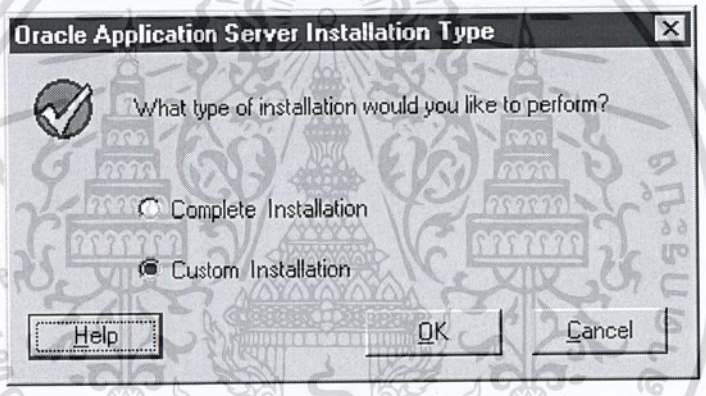
รูปที่ 2 แสดงการเลือก Product

4. การเลือก Version โดยให้เลือก Enterprise Edition



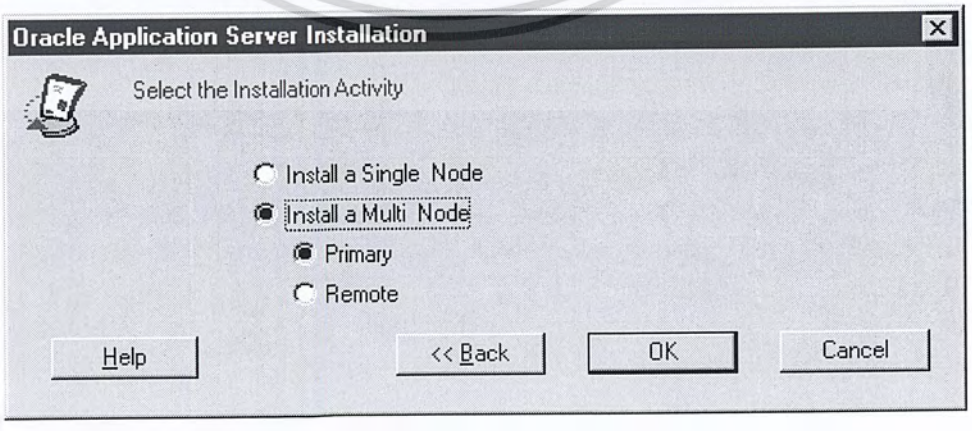
รูปที่ 3 แสดงการเลือก Version

5. การเลือก Installation Type โดยให้เลือกแบบ Custom



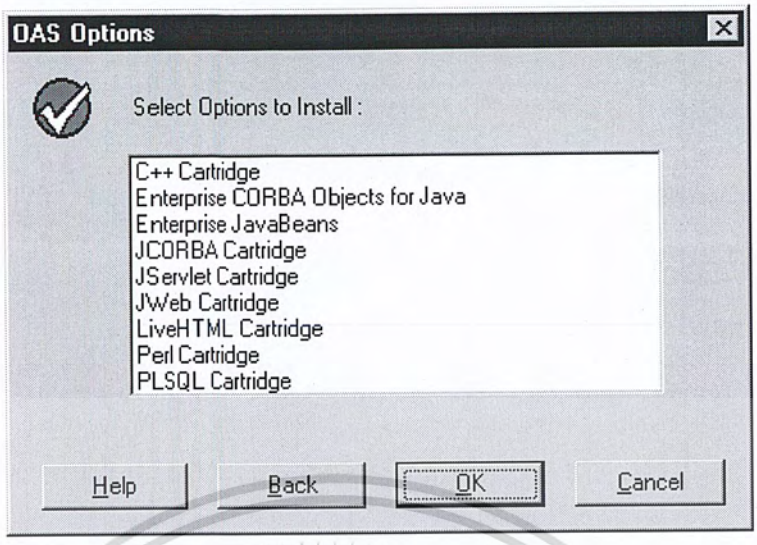
รูปที่ 4 แสดงการเลือก Installation Type

6. กำหนด Installation Activity ให้เลือก Multi Node แบบ Primary



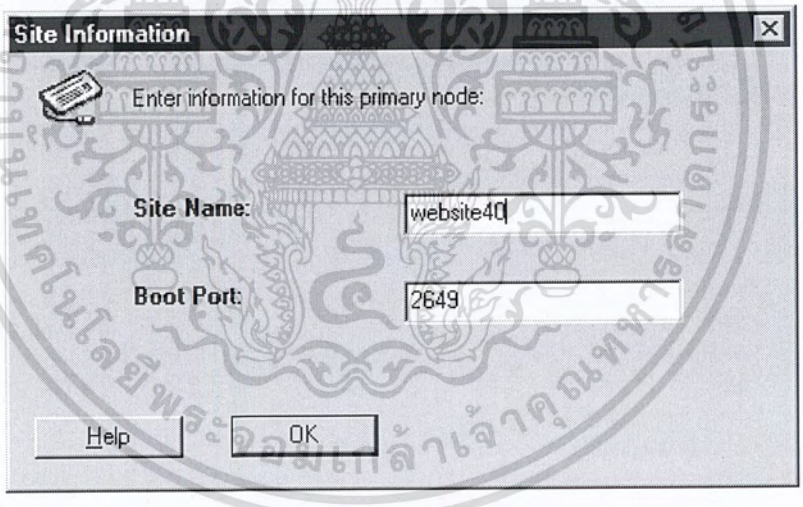
รูปที่ 5 แสดงการกำหนด Installation Activity

7. การเลือก Option ที่ต้องการให้เลือกทุกตัวเลือก



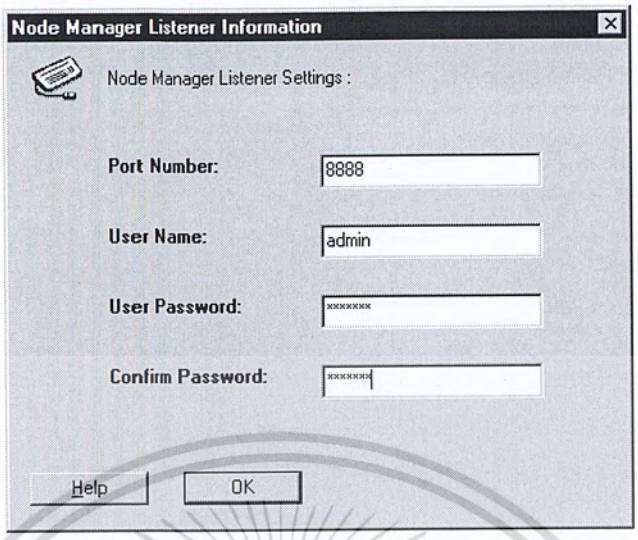
รูปที่ 6 แสดงการเลือก Option

8. การกำหนด Site Name และ Boot Port ให้กำหนดตามรูปที่ 7



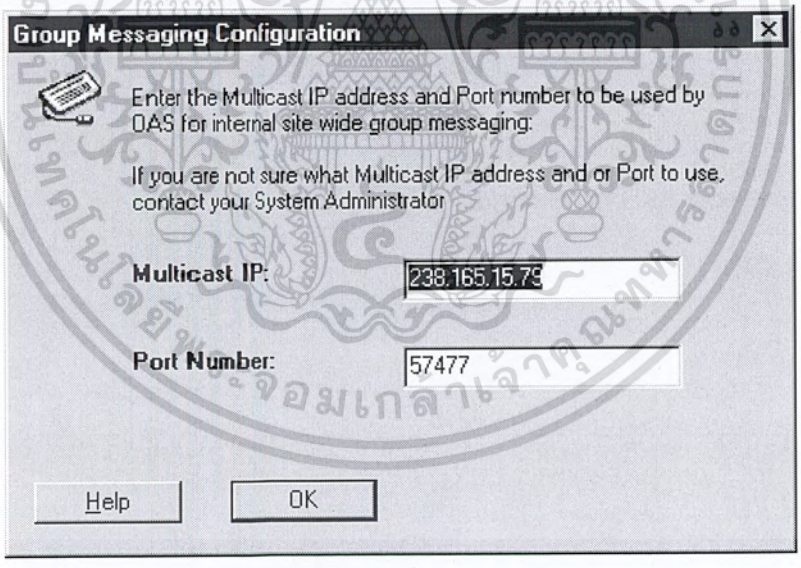
รูปที่ 7 แสดงการกำหนด Site Name และ Boot Port

9. กำหนดข้อมูลเกี่ยวกับการเข้ามาจัดการ Server ของ Administrator ให้ใช้ Port 8888



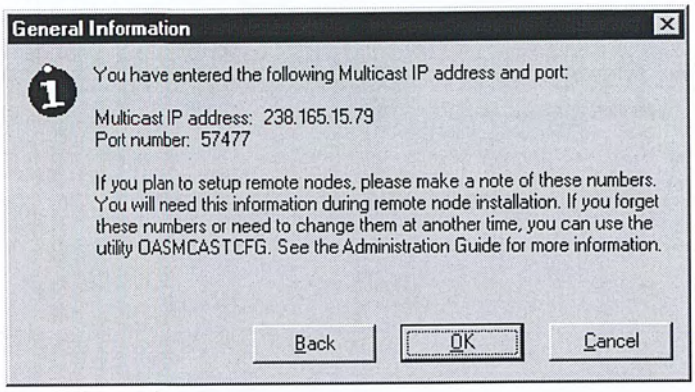
รูปที่ 8 การกำหนด User Name และ Password

10. การกำหนด Multicast IP และ Port number ให้กำหนดตาม Default



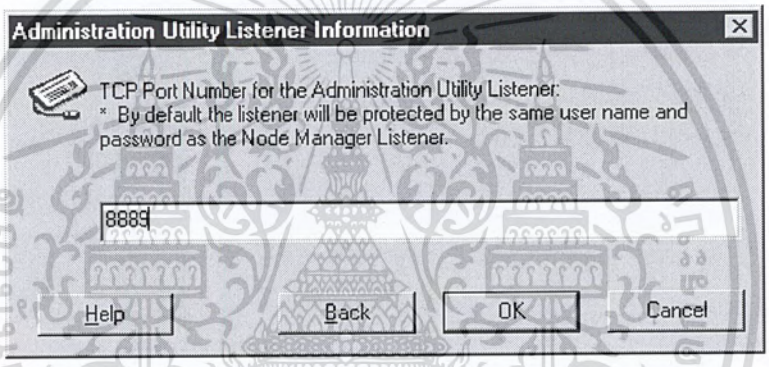
รูปที่ 9 การกำหนด Multicast IP และ Port number

11. แสดง General Information ให้กด OK



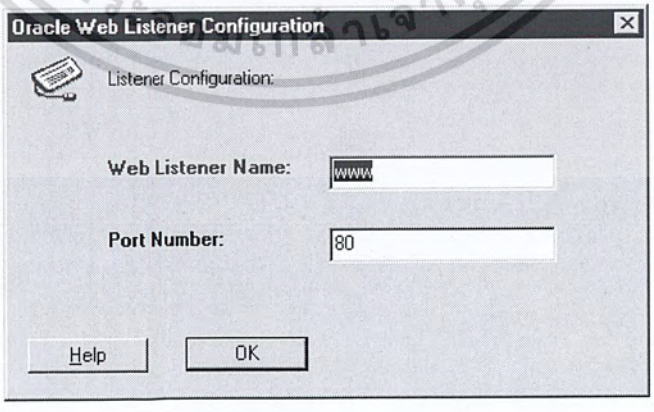
รูปที่ 10 แสดง General Information

12. กำหนด TCP Port Number ให้กำหนดตาม Default



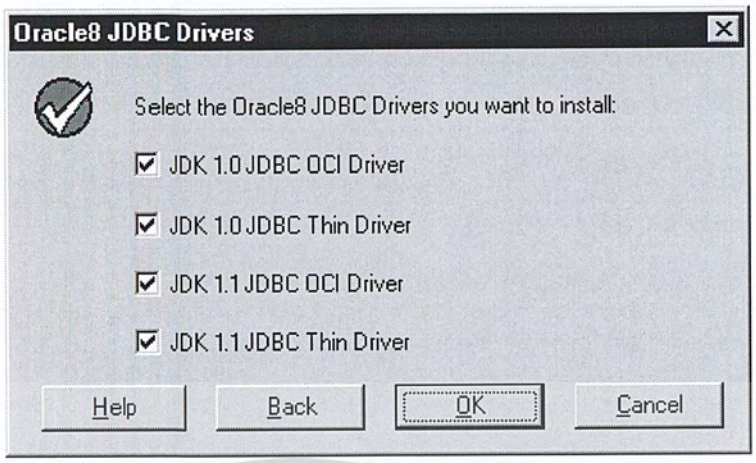
รูปที่ 11 แสงการกำหนด TCP Port Number

13. กำหนด Port Number เพื่อใช้ในการเรียก Web ให้กำหนดตาม Default



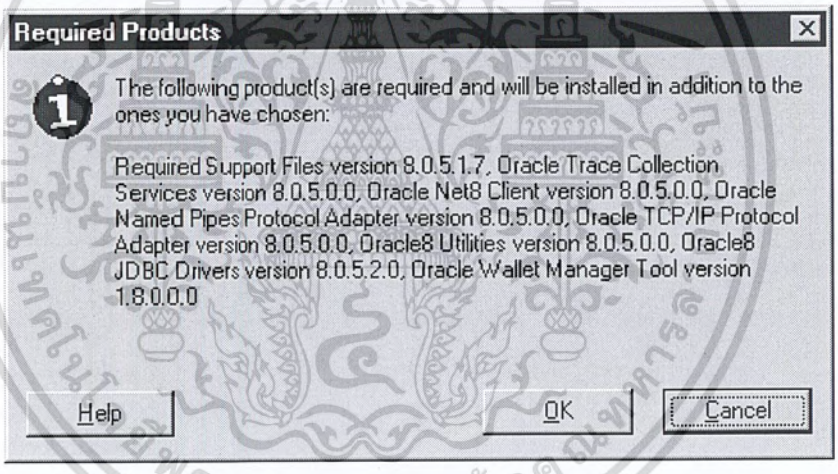
รูปที่ 12 การกำหนด Port Number เพื่อใช้ในการเรียก Web

14. เลือก JDBC เพื่อใช้ในการติดต่อกับฐานข้อมูล



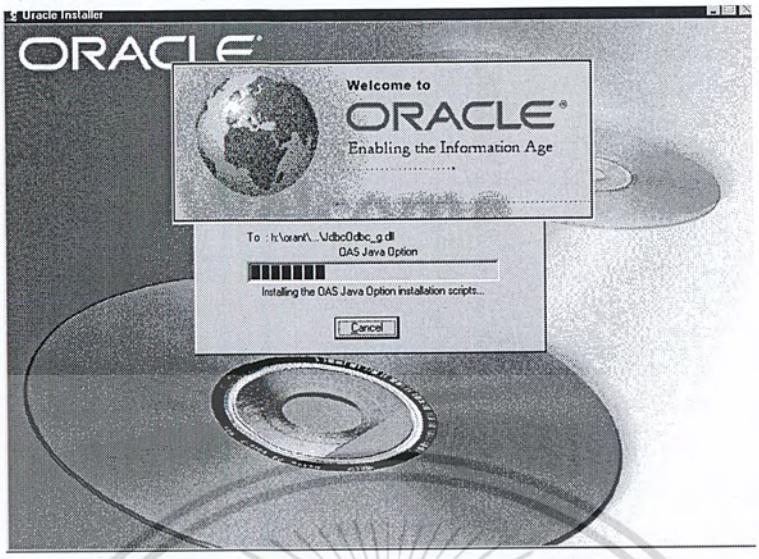
รูปที่ 13 เลือก JDBC เพื่อใช้ในการติดต่อกับฐานข้อมูล

15. แสดง Product ที่จะทำการ Install



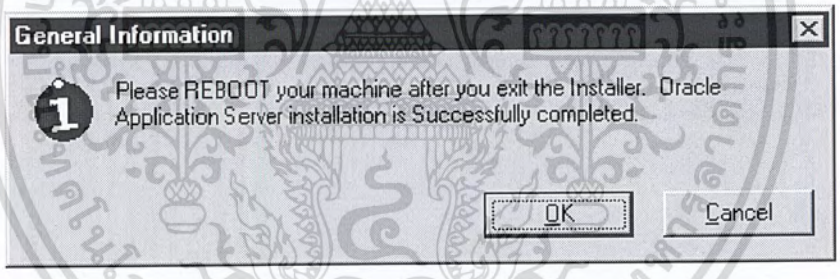
รูปที่ 14 แสดง Product ที่จะทำการ Install

16. แสดง Install Progress



รูปที่ 15 แสดง Install Progress

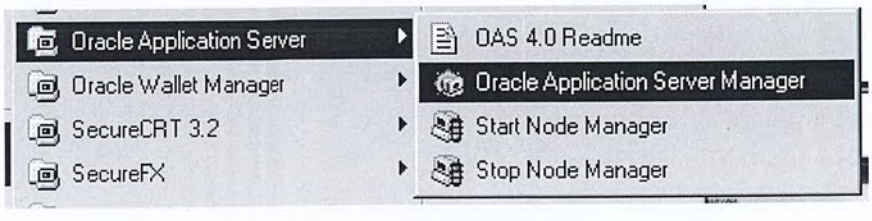
17. แสดงการแนะนำให้ Reboot



รูปที่ 16 แสดงการแนะนำให้ Reboot

2. ขั้นตอนการติดตั้ง Cartridge

1. เลือก Menu Oracle Application Server Manager

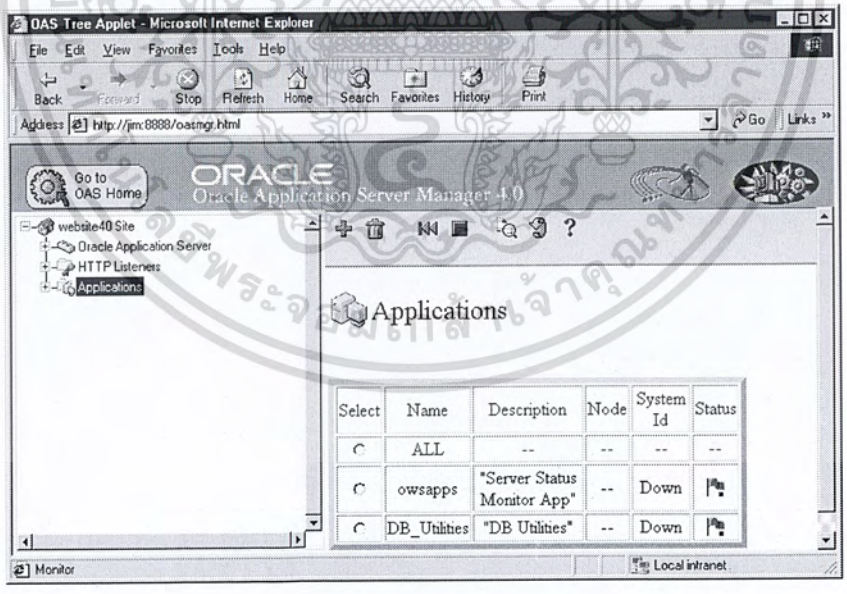


รูปที่ 17 แสดงการเลือก Menu Oracle Application Server Manager

2. ใส่ UserPassword ที่กำหนดไว้ตอน Install

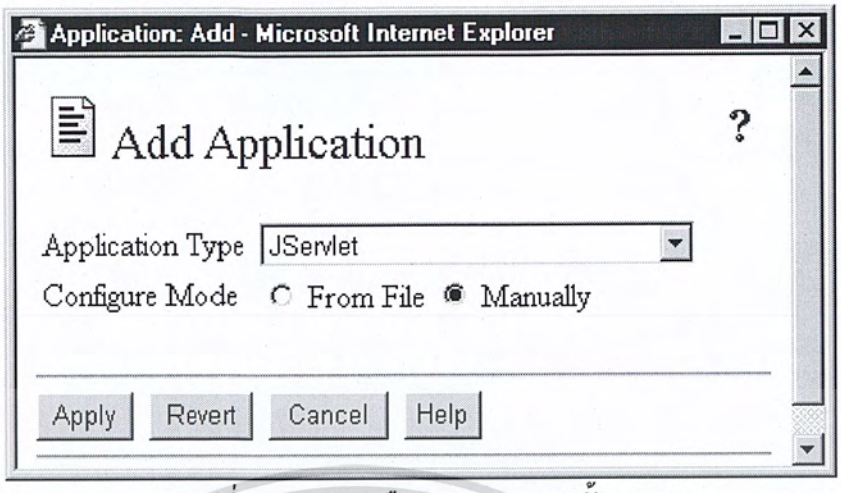
รูปที่ 18 การใส่ UserPassword ที่กำหนดไว้ตอน Install

3. เลือก Menu Applications แล้วกด  เพื่อทำการ Add Application ใหม่



รูปที่ 19 การเลือก Menu Applications

4. เลือก JServlet จากนั้นกด Apply

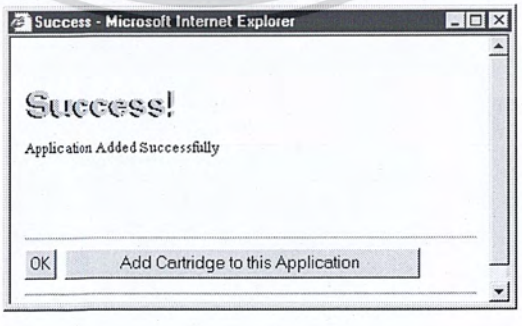


รูปที่ 20 แสดงการเลือก JServlet จากนั้นกด Apply

5. กำหนดรายละเอียดของ Application จากนั้นกด Apply (เป็นเพียงการตั้งชื่อเท่านั้น)

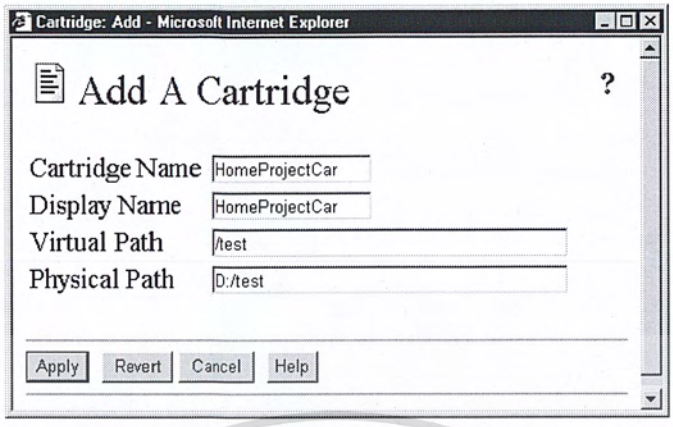
รูปที่ 21 กำหนดรายละเอียดของ Application

6. แสดงการ Add Application Success ต่อไปให้กด Add Cartridge to this Application



รูปที่ 22 แสดงการ Add Application Succes

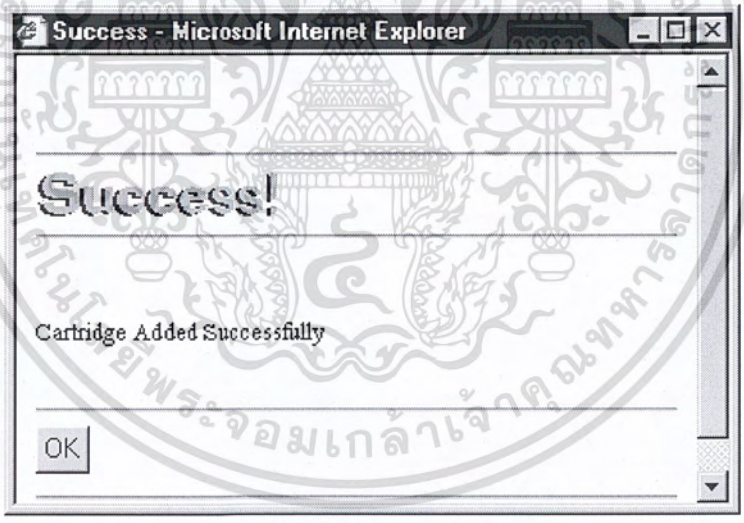
7. กำหนดรายละเอียดของ Cartridge Virtual Path : /test ,
Physical Path : D:\test จากนั้นกด Apply



รูปที่ 23 การกำหนดรายละเอียดของ Cartridge

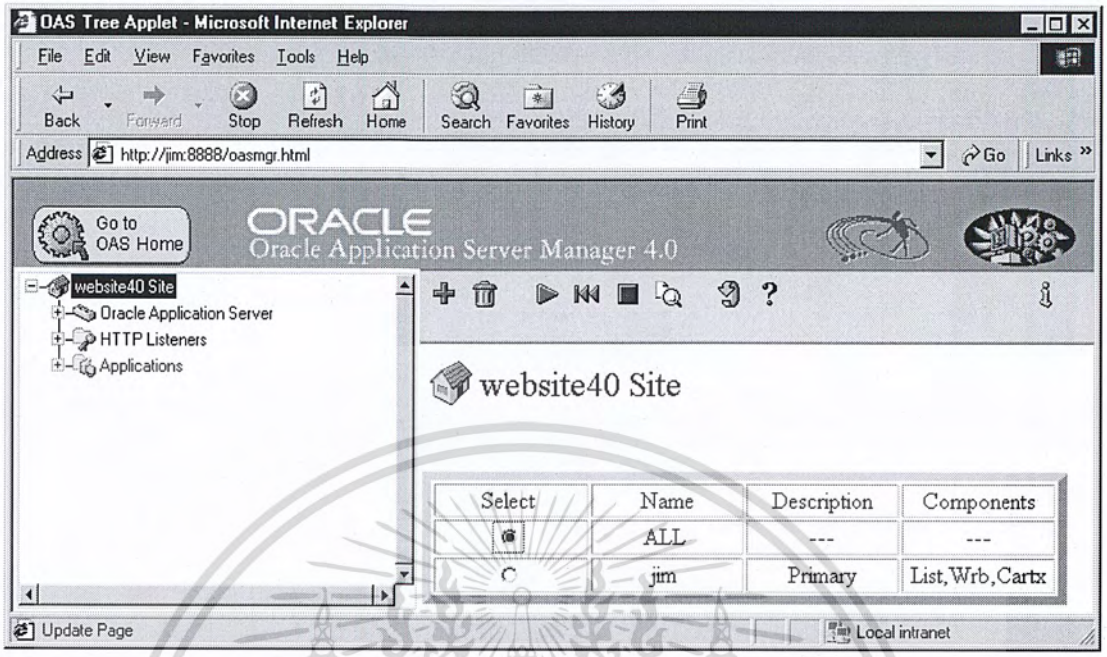
หมายเหตุ คลาสไฟล์ servlets ทุกคลาสจะต้องเก็บในไดเรคทรอรี่ D:/test ที่กำหนดใน Physical Path

8. แสดงการ Add Cartridge Success



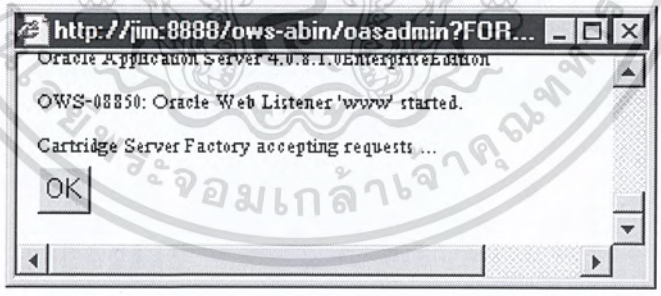
รูปที่ 24 แสดงการ Add Cartridge Success

9. เลือก Menu website4.0 Site และ เลือก ALL จากนั้นกดปุ่ม Reload (|<<)



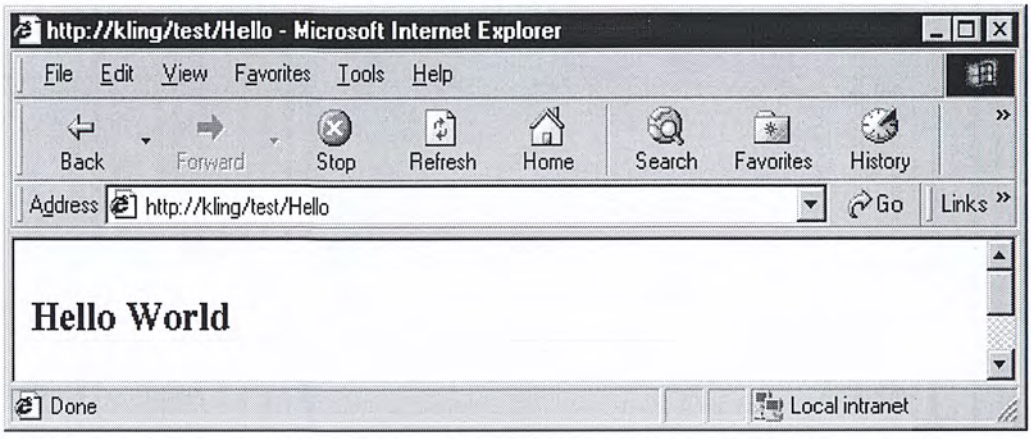
รูปที่ 25 แสดงการเลือก Menu website4.0 Site

10. แสดงการ Start เสร็จสมบูรณ์และตรวจดูว่ามี Error หรือไม่ถ้ามีแสดงว่าการติดตั้ง OAS ผิดพลาดให้ตรวจสอบการติดตั้งอีกครั้ง



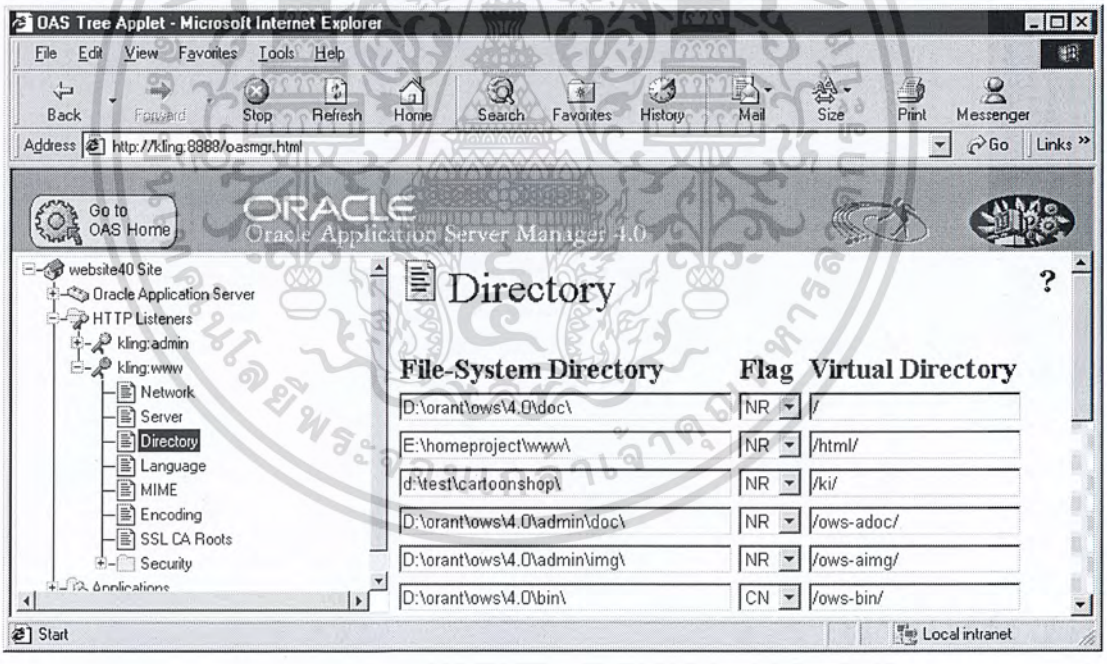
รูปที่ 26 แสดงการ Start เสร็จสมบูรณ์

3. ทำการทดสอบโดยใช้ Browser เรียก Hello



รูปที่ 27 ผลการทดสอบ Hello

4. กำหนด Directory ที่เก็บ files HTML ทั้งหมด และ Virtual Directory ที่ใช้ในการอ้างอิงสำหรับเรียก file HTML จาก Browser ดังนี้ E:\homeproject\www และ /html/ ตามลำดับ



รูปที่ 28 กำหนด Directory และVirtual Directory สำหรับ HTML

หมายเหตุ ไฟล์ HTML ทุกไฟล์ และ ไฟล์ภาพทั้งหมด image ,images จะต้องเก็บในไดเรคทรอรี่ E:\homeproject\www

บรรณานุกรม

หนังสืออ้างอิง

[1] Jason Hunter with William Crawford : " JAVA Servlet Programming ",O'REILLY,1998

[2] Maghraj Thakkar : " Oracle8I on Windows NT " , SAMS , 1999

[3] Karl Moss : " Java Servlets " , McGraw-Hill , 1999

[4] Merlin Hughes , Michael Shokkner , Derek Hamner : " Java Network Programming Second Edition " , MANNING , 1997

[5] Bill McCarty and Luke Cassady – Dorion : " JAVA Distributed Objects " , SAMS , 1999

[6] พ.อ. เจนวิทย์ เหลืองอร่าม : " การเขียนโปรแกรมสำหรับ Applications และ Applets ด้วย JAVA" , พิมพ์ที่ บริษัท ธรรมสาร จำกัด , 2542

[7] ดร. วีระศักดิ์ ซึงถาวร : " Findamental of Java Programming Volumn 2 " , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2543

[8] อาจารย์อภิเนตร อูนากูล : " Object – Oriented Analysis And Design " , แผนกตำรา คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2543

เว็บไซต์อ้างอิง

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

[2] http://www.servlet.com

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

[4] http://www.javasoft.com