การสร้างโปรแกรมประยุกต์เชิงวัตถุบน แอพพลิเคชันเซิร์ฟเวอร์ ด้วย JSP และ JAVABEANS

OBJECT-ORIENTED APPLICATION IMPLEMENT ON APPLICATION SERVER USING JSP AND JAVABEANS

นายณัฐวุฒิ ตาทอง
นายพิเชษฐ์ ยาวิราช

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

การสร้างโปรแกรมประยุกต์เชิงวัตถุบน แอพพลิเคชันเซิร์ฟเวอร์ ด้วย JSP และ JAVABEANS

OBJECT-ORIENTED APPLICATION IMPLEMENT ON APPLICATION SERVER USING JSP AND JAVABEANS

โดย

นายณัฐวุฒิ ตาทอง

นายพิเชษฐ์ ยาวิราช

อาจารย์ที่ปรึกษา

รศ. บรรจง ปิยธำรง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสร้างโปรแกรมประยุกต์เชิงวัตถุบนแอพพลิเคชันเซิร์ฟเวอร์ ด้วย JSP และ JAVABEANS

OBJECT-ORIENTED APPLICATION IMPLEMENT ON APPLICATION SERVER USING JSP AND JAVABEANS

ผู้จัดทำ

1. นายณัฐวุฒิ ตาทอง รหัสประจำตัว 41013530
2. นายพิเชษฐ์ ยาวิราช รหัสประจำตัว 41013541

_____ อาจารย์ที่ปรึกษา

(รศ. บรรจง ปิยธำรง)

การสร้างโปรแกรมประยุกต์เชิงวัตถุบนแอพพลิเคชันเซิร์ฟเวอร์ ด้วย JSP และ JAVABEANS

นายณัฐวุฒิ ตาทอง 41013530
นายพิเชษฐ์ ยาวิราช 41013541
รศ. บรรจง ปิยธำรง อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ในปัจจุบันเทคโนโลยีทางด้านเว็บไซต์ได้พัฒนาไปมาก เพื่อรองรับการทำธุรกิจผ่านทางอินเทอร์เน็ต ได้มีการพัฒนาในด้านของประสิทธิภาพและความสามารถให้มากขึ้น สำหรับระบบที่มีการนำเสนอข้อมูลซึ่งมีการเปลี่ยนแปลงบ่อยครั้งนั้น ไดนามิกเว็บไซต์เป็นแนวทางหนึ่งซึ่งเหมาะสมกับการนำมาใช้งานทั้งนี้เนื่องจากมีคุณสมบัติที่สะดวกรวดเร็วในการปรับปรุงแก้ไขต่อผู้พัฒนา

การพัฒนาเชิงวัตถุเป็นรูปแบบการพัฒนาที่ข้อดีเหนือกว่าการพัฒนารูปแบบเดิมหลายประการ มีโมเดลหลายชนิดที่นำเสนอรูปแบบพัฒนาเชิงวัตถุแต่ UML ได้เป็นโมเดลมาตรฐานรวมเอาข้อดีของโมเดลต่าง ๆ เอาไว้และนำเสนอในรูปแบบที่เข้าใจง่ายทั้งผู้พัฒนาและผู้ใช้ระบบทำให้ได้ระบบที่ตรงกับความต้องการมากยิ่งขึ้น

การจัดการเกี่ยวกับโครงสร้างการทำงานและการใช้ทรัพยากรในเว็บเซิร์ฟเวอร์ เป็นสิ่งที่ควรทำควบคู่ไปกับการสร้างเว็บไซต์ด้วย ทั้งนี้เพื่อปรับปรุงประสิทธิภาพของเว็บเซิร์ฟเวอร์ให้ตอบสนองตามความต้องการของผู้ใช้ได้ สถาปัตยกรรมแอพพลิเคชันเซิร์ฟเวอร์ได้เข้ามามีบทบาทในการกระทำดังกล่าว

วิทยานิพนธ์ฉบับนี้ได้เป็นอีกแนวทางหนึ่งในการสร้างเว็บไซต์แบบแอพพลิเคชันเซิร์ฟเวอร์ โดยจำลองระบบทำสัญญาซื้อขายบ้านจัดสรรมาเป็นระบบตัวอย่างในการพัฒนาและนำเสนอโมเดลตามสัญลักษณ์ของ UML มาช่วยในการวิเคราะห์ความต้องการและออกแบบระบบ จากนั้นนำผลลัพธ์ที่ได้ไปสร้างเป็นโค้ดด้วย JSP และ Java Beans

OBJECT-ORIENTED APPLICATION IMPLEMENT ON APPLICATION SERVER USING JSP AND JAVABEANS

Nuttawut Tartong

Pichet Yawirach

Assoc.Prof. Banjong Piyathamrong Advisor

ABSTRACT

Recently, Web site technology was developed rapidly both performance and ability to make business via internet. For propose changing data system Dynamic Web site has properties about easy to use and good maintenance for developer that is appropriate solution for implement to business system.

Object - Oriented Development has many advantages over other development methods. Many models present Object - Oriented Development. But UML which contains the good things of all models was used as the standard model. UML also presents in easy - understanding way, helping the system developers and users get a system that meets their requirements.

About management the structure of system and using resource in server. Them should be work with construct Web site simultaneously. For improve performance of Web Server to response requirement of users. Application Server architecture has the role of this.

This thesis is concerned with the way to implement Application Server web site and shows The Contract Project Home System as an example and representing models with UML notation in analysis requirements and design phase. Then, take the result to coding by JSP and Java Beans.

กิตติกรรมประกาศ

โครงงานและวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงเป็นรูปเล่มขึ้นมาได้ด้วยดี เนื่องจากผู้จัดทำได้รับความร่วมมือและความช่วยเหลือจากบุคคลต่าง ๆ หลายฝ่าย ทั้งทางด้านวิชาการและกำลังใจ ผู้จัดทำขอขอบพระคุณ

อาจารย์ที่ปรึกษา รศ. บรรจง ปิยธำรง ที่กรุณาชี้แนะให้คำปรึกษาและดูแลให้โครงงานนี้สำเร็จได้

คณาจารย์ภาควิชาคอมพิวเตอร์ทุกท่านที่คอยให้ความรู้ในหลายๆ ด้านแก่เราตลอด 3 ปีที่ผ่านมา

ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่คอยให้แสงสว่างท้าทายยามราตรี ให้ความร่มเย็นในยามรุ่มร้อน

เหล่าผองเพื่อนห้อง P,D ที่ช่วยกันแต่งแต้มสีสันให้กับชีวิตวัยหนุ่มสาว ด้วยรอยยิ้มและอาทร

บุคคลที่ให้ลมหายใจคอยเลี้ยงดูเรามา พ่อ แม่ ผู้ซึ่งให้ความรักและหวังดี คอยปลอบใจเราเสมอมา

ช่วงเวลาที่ผันผ่านกับชีวิตที่ต้องก้าวเดิน มาจนถึงวันนี้ข้าพเจ้านึกไม่ออกเลยว่าจะบอกสิ่งที่ข้าพเจ้ารู้สึกได้อย่างไรแต่มีสิ่งหนึ่งที่ข้าพเจ้าเชื่อเสมอมา ครั้นหนึ่งเราเคยได้อยู่บ้านแคแสดหลังนี้ บ้านอันเป็นที่รักของเรา พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ด้วยหัวใจและยืนยัน

นายณัฐวุฒิ ตาทอง

นายพิเชษฐ์ ยาวิราช

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 7 สถาปัตยกรรมของ แอพพลิเคชันเซิร์ฟเวอร์

บทที่ 8 การสร้างโปรแกรมประยุกต์ด้วย JSP และ JavaBeans

บทที่ 9 บทสรุปโปรแกรมทำสัญญาซื้อขายบ้านจัดสรร

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

การพัฒนาโครงสร้างของเว็บเซิร์ฟเวอร์นั้นได้มีส่วนช่วยในด้าน การตอบสนองความต้องการของผู้ใช้ในโลกของอินเทอร์เน็ต ให้เป็นไปอย่างรวดเร็วและทั่วถึงเราจึงจำเป็นต้องมีการจัดการระบบโครงสร้างของเว็บเซิร์ฟเวอร์ให้เหมาะสมเพื่อรองรับการให้บริการผู้ใช้เหล่านั้น การจัดการในด้านของการนำเสนอข้อมูลให้ทันสมัยทันและง่ายต่อการปรับปรุงข้อมูลของเว็บไซต์ ก็เป็นเรื่องที่ต้องให้ความสำคัญอย่างหนึ่ง สำหรับเว็บไซต์เล็ก ๆ แล้วก็มักจะให้บริการเพียงพอต่อความต้องการของผู้ใช้ทั่วไป อย่างไรก็ตามสำหรับเว็บไซต์ใหญ่ ๆ ที่มีผู้ใช้จำนวนมาก ส่วนประกอบที่สำคัญอีกอย่างหนึ่งที่มักจะถูกเพิ่มเข้าไปมักจะเป็นส่วนที่เรียกว่า แอพพลิเคชันเซิร์ฟเวอร์ (Application Server)

แอพพลิเคชันเซิร์ฟเวอร์เป็นส่วนประกอบอีกส่วนหนึ่งที่ถูกสร้างขึ้นมาเพื่อสนองความต้องการที่ว่า ในการสร้างระบบที่ดีระบบหนึ่งขึ้นมานั้น เราควรจะแยกส่วนที่ทำหน้าที่เกี่ยวกับการบริการต่าง ๆ รวมไปถึงการจัดสรรข้อมูลออกมาเป็นอีกส่วนต่างหาก ทั้งนี้เพื่อเพิ่มความคล่องตัวในการเปลี่ยนแปลงของระบบ รวมไปถึงการจัดระบบให้แบ่งออกเป็นส่วน ๆ อย่างชัดเจนมากยิ่งขึ้น บทบาทของแอพพลิเคชันเซิร์ฟเวอร์ที่ผ่านมาตั้งแต่อดีตจนถึงปัจจุบันเราอาจจะพูดได้ว่า แอพพลิเคชันเซิร์ฟเวอร์ก็คือส่วนที่ช่วยอำนวยความสะดวกให้กับโปรแกรมทางฝั่งเซิร์ฟเวอร์ (Server Side Application) โดยมักจะเป็นตัวช่วยในการคำนวณข้อมูล, ช่วยเป็นตัวกลางในการจัดส่งข้อมูล, ช่วยในการควบคุมการจัดเก็บข้อมูลลงไปในดาต้าเบสอย่างถูกต้อง ตลอดจนช่วยในการเพิ่มความเร็วให้กับระบบโดยการเก็บออบเจ็กต์ต่าง ๆ ที่ใช้แล้วหรือไม่ใช้นำกลับมาใช้ใหม่ เป็นต้น

ในการออกแบบระบบซอฟต์แวร์ที่จะใช้เป็นระบบจำลองนั้นจะใช้วิธีการพัฒนาเชิงวัตถุ (Object-Oriented Development) ได้เป็นแนวทางในการพัฒนาที่ดีและได้รับความนิยมแพร่หลายในปัจจุบัน โดยใช้มุมมองจากโลกแห่งความเป็นจริงทำให้สามารถเข้าใจในตัวระบบที่ทำการพัฒนาได้ชัดเจนยิ่งขึ้น โดยมีข้อดีดังต่อไปนี้

- โมเดลมีความสอดคล้องกันในทุกขั้นตอนในการพัฒนา ทั้งในขั้นตอนการวิเคราะห์,การออกแบบ,ไปจนถึงการสร้าง
- โมเดลเชิงวัตถุมีการแยกแยะเอกลักษณ์ (Abstraction) โดยเป็นการแยกเอาเฉพาะสิ่งที่สนใจออกมา แสดงคุณลักษณ์ที่สำคัญของออบเจ็กต์และการซ่อนรายละเอียด (Encapsulation) ซึ่งเป็นการซ่อนข้อมูลและความซับซ้อนภายในออบเจ็กต์
- รองรับการเปลี่ยนแปลงของระบบได้ และ การปรับเปลี่ยนขนาดของระบบ
- สนับสนุนการนำกลับมาใช้ใหม่ เชื่อถือได้ปลอดภัย
- สนับสนุนการทำงานพร้อม ๆ กัน (Concurrency)

วิธีการพัฒนาเชิงวัตถุจะประกอบไปด้วยขั้นตอนต่าง ๆ คือขั้นตอนการวิเคราะห์ออกแบบโดยจะมองภาพรวมของฟังก์ชันการทำงานของระบบเป็นหลัก จากนั้นกำหนดรายละเอียดของ ออบเจ็กต์,คลาส, ความสัมพันธ์และคุณลักษณะที่จะมีในระบบและจะเข้าสู่ขั้นตอนการพัฒนาโดยใช้ภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (Object - Oriented Programming Language)

จากทั้งหมดที่กล่าวมาขั้นต้นนั้น โครงงานนี้จึงได้ใช้แนวทางการพัฒนาเชิงวัตถุและการพัฒนาโครงสร้างของเว็บเซิร์ฟเวอร์แบบแอพพลิเคชันเซิร์ฟเวอร์ โดยจะกำหนดโดเมนปัญหามาจากระบบทำสัญญาซื้อขายบ้านจัดสรรเป็นระบบจำลอง เพื่อทำการวิเคราะห์ความต้องการของระบบและออกแบบโดยใช้ CASE tool สนับสนุนโมเดลเชิงวัตถุ สร้างโปรแกรมด้วยภาษาจาวาและมีการจัดการฐานข้อมูลแบบรีเลชันแนล

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาการโครงสร้างและการจัดระบบการทำงานของเว็บเซิร์ฟเวอร์ แบบแอพพลิเคชันเซิร์ฟเวอร์ เพื่อเป็นแนวทางในการพัฒนาประสิทธิภาพของเว็บเซิร์ฟเวอร์ได้

1.2.2 ศึกษาการนำเสนอโมเดลการพัฒนาเชิงวัตถุด้วย UML (Unified Modeling Language) ด้วยโปรแกรม Rational Rose ซึ่งเป็น CASE tool ช่วยในการสร้างโมเดล

1.2.3 ศึกษาการทำงานของ JSP (Java Server Page) และ JavaBeans

1.2.4 ศึกษาการสร้างฐานข้อมูลรีเลชันแนลจากออบเจ็กต์ที่ได้จากการพัฒนาเชิงวัตถุและทำการสร้างฐานข้อมูลแบบรีเลชันแนลด้วย ออราเคิล (Oracle 8.1.7)

1.2.5 นำความรู้ที่ได้นำมาออกแบบและพัฒนาโปรแกรมประยุกต์เพื่อให้สอดคล้องกับ โมเดลของแอพพลิเคชันเซิร์ฟเวอร์ได้

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะออกแบบและสร้างโปรแกรมประยุกต์สำหรับ การทำสัญญาซื้อขายบ้านจัดสรร โดยมีฟังก์ชันการทำงานและลักษณะของระบบตามที่กำหนดไว้ในรายละเอียดความต้องการของระบบที่ได้จากขั้นตอนการวิเคราะห์ แต่เป็นการจำลองฟังก์ชันการทำงานหลักมาเท่านั้น ยังไม่ครอบคลุมถึงรายละเอียดต่าง ๆ ที่เป็นจริงทั้งหมดในทางธุรกิจ ระบบนี้เป็นระบบตัวอย่างเพื่อใช้เป็นแนวทางในการศึกษาการพัฒนาระบบเท่านั้น

โดยจะมีระบบเว็บเซิร์ฟเวอร์ของ OAS (Oracle Application Server 4.0.8.2.1) และระบบฐานข้อมูลนั้นจะเป็นออราเคิล โดยจะรันภายใต้ระบบปฏิบัติการ Windows NT Server 4.0 ส่วนไคลเอ็นต์จะติดต่อกับเซิร์ฟเวอร์ฐานข้อมูลผ่านทาง อินเทอร์เน็ต (Internet)

1.4 วิธีดำเนินงาน

เริ่มจากการศึกษาทฤษฎีการพัฒนาเชิงวัตถุ กำหนดโดเมนปัญหาของระบบ วิเคราะห์ความต้องการและออกแบบระบบนำเสนอด้วยโมเดล UML กำหนดรายละเอียดระบบทำสัญญาซื้อขายบ้านจัดสรร แสดงโมเดลต่าง ๆ และทำการออกแบบฐานข้อมูล ต่อมาก็จะเป็นการติดตั้งเว็บเซิร์ฟเวอร์และดาต้าเบสเซิร์ฟเวอร์ ในบทต่อมาก็จะเป็นวิธีการสร้างฐานข้อมูลแบบรีเลชันแนลด้วยออราเคิล

บทสุดท้ายก็จะเป็นสรุปผลการทดลอง การทำงานของโปรแกรมประยุกต์ที่สร้างขึ้น และแนวทางในการพัฒนาโครงงานนี้เพิ่มเติม

บทที่ 2
การพัฒนาระบบด้วยวิธีเชิงวัตถุ

2.1 บทนำ

Visual Modeling Technique (VMT) คือแนวทางหนึ่งของการพัฒนาเชิงวัตถุ โดยการออกแบบระบบด้วยโมเดลตามแนวความคิดของโลกแห่งความเป็นจริง โมเดลที่ถูกออกแบบมานั้น จะมีประโยชน์ช่วยในการเข้าใจถึงปัญหาของระบบ, การติดต่อกันของผู้ที่จัดทำโครงงาน (เช่น ลูกค้า , ผู้ดูแลระบบ , นักวิเคราะห์ และนักออกแบบ เป็นต้น) , การเตรียมการทำเอกสาร , การออกแบบโปรแกรมและดาต้าเบส

โมเดลเป็นนามธรรมที่แสดงถึงลักษณะที่สำคัญของปัญหาหรือโครงสร้างของระบบ โดยการกลั่นกรองรายละเอียดที่ไม่สำคัญออก ดังนั้นจึงเป็นการทำให้ปัญหานั้นง่ายต่อการเข้าใจ เราจึงควรสร้างโมเดลสำหรับระบบที่ซับซ้อน เพราะความสามารถของคนที่จะเข้าใจระบบที่ซับซ้อนนั้นมีจำกัด การสร้างโมเดลจะทำให้นักออกแบบสามารถเห็นภาพกว้างของโครงงานได้ โมเดลที่ดีควรจะใช้สัญลักษณ์ที่ถูกต้องและมีการตรวจสอบโมเดลให้ตรงตามความต้องการของระบบ

Visual Modeling Technique จะมีส่วนเกี่ยวข้องกับทุกขั้นตอนของการพัฒนาโปรแกรมเชิงวัตถุ ทั้งการรวบรวมความต้องการ (requirement gathering), การวิเคราะห์ (analysis), การออกแบบ (design), การเขียนโปรแกรม (coding) และการทดสอบ (testing)

2.2 แบบจำลองเชิงวัตถุแบบ The Unified Modeling Language (UML)

โมเดลที่ปรับปรุงข้อดีของวิธีการต่างๆ ของ โมเดล OMT จะมีจุดแข็งที่การวิเคราะห์ แต่มีจุดอ่อนที่การออกแบบ,โมเดล Booch จะมีจุดแข็งที่การออกแบบและมีจุดอ่อนที่การวิเคราะห์ ,โมเดลแบบ OOSE จะมีจุดแข็งที่วิธีการทำการวิเคราะห์ แต่มีจุดอ่อนในส่วนอื่นๆ มารวมกันเป็น Unified Modeling Language (UML) โดยมีนิยามว่า "UML คือภาษามาตรฐานที่ใช้ในการระบุรายละเอียด สื่อที่สามารถมองเห็นได้และการสร้างวัตถุของระบบเชิงวัตถุภายใต้การพัฒนา โดยแสดงถึงการรวมกันของวิธีการแบบ Booch , OMT และเครื่องหมายที่ใช้แทนออบเจ็กต์ ซึ่งเป็นแนวความคิดที่ดีที่สุดจากจำนวนวิธีการต่างๆ ทั้งหลาย"

UML มีการจัดเตรียมสัญลักษณ์และเครื่องหมายเป็นจำนวนมาก จึงทำให้สามารถสร้างโมเดลและไดอะแกรมได้อย่างยืดหยุ่นและครอบคลุมปัญหาของระบบได้ทั้งหมด

ในโครงงานนี้เราจะใช้เครื่องมือช่วยในการออกแบบโปรแกรมเชิงวัตถุตามหลักการ UML ด้วยโปรแกรมที่ชื่อ Rational Rose

2.3 ขั้นตอนการวิเคราะห์ตาม Visual Modeling Technique (VMT)

ขั้นตอนการวิเคราะห์เป็นขั้นตอนหนึ่งของการพัฒนาระบบผลลัพธ์ที่ได้ ขั้นตอนนี้คือการอธิบายว่าระบบถูกสร้างขึ้นมาเพื่อทำอะไร ไม่ได้เป็นการระบุว่าระบบควรจะทำงานอย่างไร ผลลัพธ์จากการวิเคราะห์คือโมเดลเชิงวัตถุ ซึ่งโดยปกติจะเรียกขั้นตอนนี้ว่า Modeling Phase

ขั้นตอนของการวิเคราะห์มีดังนี้

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

2.3.1 ประโยคปัญหา (Problem Statement)

เป็นบทความที่บรรยายถึงลักษณะปัญหาของระบบ หน้าที่การทำงานต่างๆ ที่ระบบต้องมี ความสามารถในการติดต่อกันระหว่างผู้ใช้กับระบบขอบเขตการทำงานของระบบและรายละเอียดหน้าที่การทำงานที่มีในระบบ

2.3.2 Use Case Model

Use case model ประกอบด้วยผู้กระทำ (Actor) และ Use Case ผู้กระทำจะอยู่นอกระบบ แสดงถึงบุคคลหรือสิ่งของที่มีความสัมพันธ์กับระบบ โดยผู้กระทำอาจจะให้ข้อมูลแก่ระบบเพียงอย่างเดียว,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 นั่นคือ Use Case B *uses* Use Case A
- ความสัมพันธ์แบบ *extends* เกิดจาก Use Case B ถูกสร้างเริ่มต้นมาจาก Use Case A และเพิ่มรายละเอียดบางส่วน เราเรียกว่า Use Case B *extends* Use Case A

2.3.3 ส่วนติดต่อผู้ใช้ (User Interface)

เพื่อความเข้าใจความต้องการของผู้ใช้ในการใช้งานโปรแกรม จึงควรมีการออกแบบส่วนการติดต่อระหว่างผู้ใช้กับระบบ โดยอาจจะเป็นการสร้างส่วนติดต่อของโปรแกรมในลักษณะคร่าวๆเพื่อให้ผู้ใช้สามารถเห็นภาพของการใช้งานโปรแกรมได้ดีขึ้น ซึ่งการสร้างส่วนติดต่อผู้ใช้นี้จะช่วยให้ผู้พัฒนาสามารถแสดงลักษณะการติดต่อภายใน Use Case ได้และเป็นการช่วยในการออกแบบโปรแกรมในขั้นตอนการออกแบบ

2.3.4 Object Model

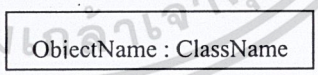
Object Model เป็นแบบจำลองที่สำคัญที่ได้จากขั้นตอนการวิเคราะห์ระบบและเป็นจุดเริ่มต้นในส่วนของการออกแบบและการสร้างระบบ ในโครงงานนี้เราจะสร้าง Object Model ตามรูปแบบของ UML

ขั้นตอนการสร้าง Object Model มีดังนี้

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

2.3.4.1 ออบเจ็กต์ (Object)

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

Object : 

รูปที่ 2-1 สัญลักษณ์แสดงออบเจ็กต์

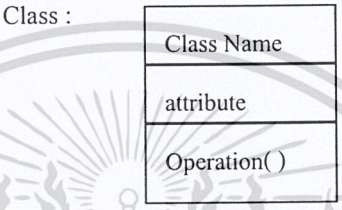
สิ่งที่เราสนใจในตัวออบเจ็กต์ประกอบไปด้วย

- แอตทริบิวต์ (Attributes) เป็นข้อมูลที่แสดงคุณลักษณะของวัตถุ เช่น รถยนต์ จะมีแอตทริบิวต์คือ ความกว้าง, ความยาว, สี, น้ำหนัก
- โอเปอเรชั่น (Operations) คือการกระทำที่วัตถุนั้นสามารถกระทำได้ เช่น รถยนต์ จะมี operations คือ ออกตัว, เบรก, เลี้ยวขวา, เลี้ยวซ้าย
- สถานะ (State) สถานะของวัตถุคือหนึ่งในเงื่อนไขที่เป็นไปได้และอาจเกิดขึ้นได้จริง เช่น รถ-ยนต์ จะมี state คือ กำลังวิ่ง, กำลังชะลอ,หยุด

- เอกลักษณ์ (Identity) เป็นสิ่งที่ระบุว่าวัตถุนั้นมีอยู่เพียงหนึ่งเดียว เช่น รถยนต์มีเอกลักษณ์คือทะเบียนรถ
- หน้าที่ (Responsibility) คือ หน้าที่และบทบาทของวัตถุต่อระบบ เช่น รถยนต์มีหน้าที่เป็นพาหนะในการเดินทางไปให้ถึงจุดหมายได้

2.3.4.2 คลาส (Class)

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



รูปที่ 2-2 สัญลักษณ์แสดงคลาส

2.3.4.3 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 Use Case ส่งผลให้เกิดพฤติกรรมเฉพาะใน Use Case หรือเรียกว่าเป็นการ “ running ”

2.3.4.4 แพ็กเกจ (Packages)

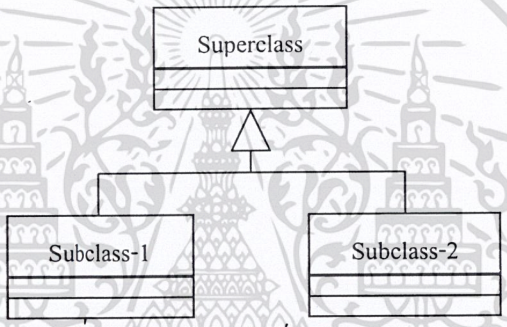
เป็นการรวมคลาสที่มีความสัมพันธ์ใกล้ชิดกันเข้าไว้ในแพ็กเกจ เพื่อประโยชน์สำหรับระบบที่มีคลาสเป็นจำนวนมาก ในระบบที่มีความซับซ้อนเราอาจจะสร้างแพ็กเกจขึ้นมาก่อน แล้วจึงบรรจุคลาสและความสัมพันธ์เข้าไว้ในแพ็กเกจ

2.3.4.5 ความสัมพันธ์

2.3.4.5.1 Generalization

เป็นความสัมพันธ์แบบการให้กำเนิดคลาสย่อย (Subclass) จากคลาสแม่ (Superclass) โดยคลาสย่อยจะสืบทอดแอตทริบิวต์ ,โอเปอเรชั่นและความสัมพันธ์ทั้งหมดจากคลาสแม่ ทั้งของตัวเองและจากคลาสแม่ของคลาสแม่ชั้นต่อไปเรื่อย ๆ คลาสย่อยอาจจะมีการเพิ่มแอตทริบิวต์และโอเปอเรชั่นในคลาสของมันเอง ความสัมพันธ์แบบนี้อาจถูกเรียกได้เป็น is-a หรือ kind-of

Generalization :

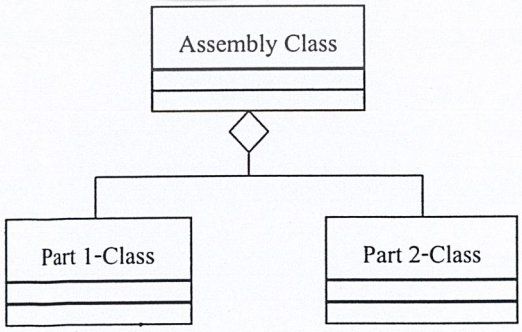


รูปที่ 2-3 ความสัมพันธ์แบบ Generalization

2.3.4.5.2 Aggregation

เป็นความสัมพันธ์แบบเป็นส่วนประกอบ หรือที่เรียกว่า part-of คำว่าเป็นส่วนประกอบบางครั้งอาจจะมีความหมายที่ต่างกัน เช่น เครื่องยนต์และล้อเป็นส่วนประกอบของรถยนต์ในขณะที่เรากำลังขับรถอยู่ แต่ถ้ารถยนต์คันนั้นอยู่ในระหว่างการซ่อมที่อู่ซ่อมรถ นั่นคือเครื่องยนต์และล้อก็ไม่ได้เป็นส่วนประกอบของรถยนต์แล้ว

Aggregation :

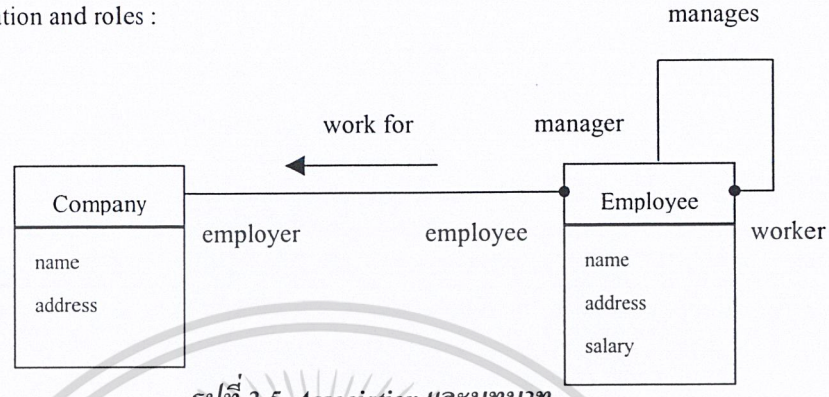


รูปที่ 2-4 ความสัมพันธ์แบบ Aggregation

2.3.4.5.3 Association

เป็นความสัมพันธ์แบบสองทิศทาง ซึ่งใช้ในการติดต่อกันระหว่างคลาส ความสัมพันธ์แบบ association ระหว่างคลาสหมายถึงมีการเชื่อมต่อระหว่างออบเจ็กต์ในคลาสทั้งสอง ที่ปลายของเส้นความสัมพันธ์จะมีการระบุชื่อบทบาทเพื่อบอกว่าคลาสนั้นถูกมองว่าเป็นอย่างไรจากคลาสอื่น

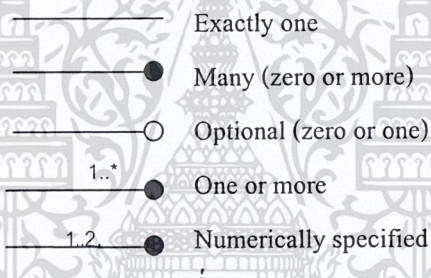
Association and roles :



รูปที่ 2-5 Association และบทบาท

Multiplicity ของ Association เป็นการบอกถึงจำนวนของออบเจ็กต์ที่มีการติดต่อกับคลาสอื่น

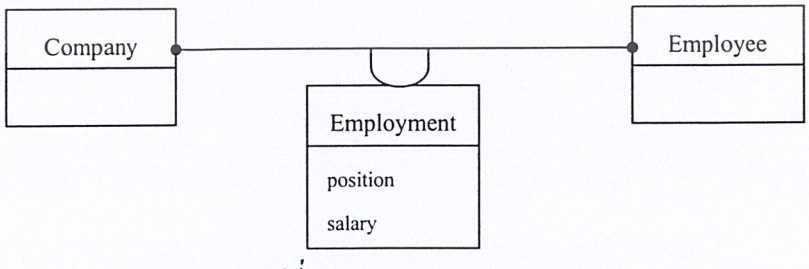
Association :



รูปที่ 2-6 Multiplicity

Association Class ในบางครั้งเราอาจจะออกแบบ Association เป็นเหมือนกับคลาสได้ ถ้าหากมีแอตทริบิวต์ ที่ไม่ได้เป็นของคลาสใดๆที่มีความสัมพันธ์กันแอตทริบิวต์ นั้นจะเป็นของ Association Class

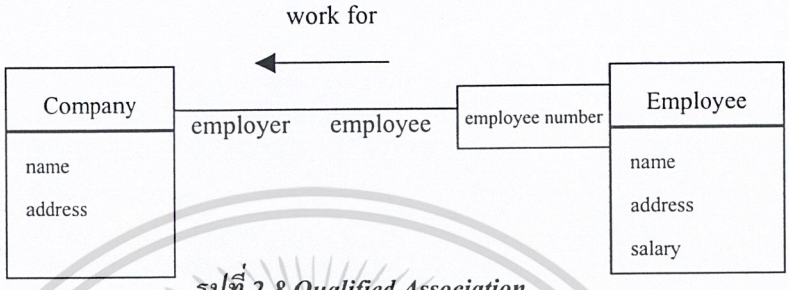
Modeling an association as a class :



รูปที่ 2-7 Association Class

Qualified Association คือ Association ที่มีแอตทริบิวต์ เรียกว่า Qualifier โดยค่าของแอตทริบิวต์ นั้นสามารถที่จะแบ่งแยกออบเจ็กต์ต่างๆของคลาสอีกด้านหนึ่งของความสัมพันธ์ได้ สัญลักษณ์ที่ใช้ก็คือ รูปสี่เหลี่ยมขนาดเล็กที่ติดอยู่ระหว่างปลายของเส้น Association กับคลาสที่เชื่อมอยู่และจะมีแอตทริบิวต์ ที่มีลักษณะดังที่กล่าวมาอยู่ภายใน

Qualified Association :

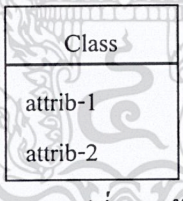


รูปที่ 2-8 Qualified Association

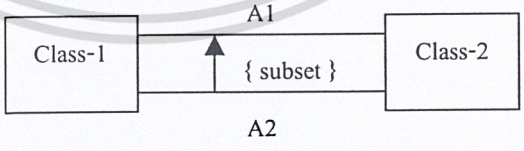
2.3.4.6 ข้อบังคับ (Constraints)

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

รูปแบบทั่วไปของข้อบังคับจะเป็นข้อความที่อยู่ในวงเล็บปีกกา เช่น



รูปที่ 2-9 ข้อบังคับในออบเจ็กต์



รูปที่ 2-10 ข้อบังคับระหว่าง Association

2.3.5 Dynamic Model

เป็นแบบจำลองที่อธิบายถึงพฤติกรรมของระบบและลำดับของการทำงาน ซึ่งแบ่งเป็น

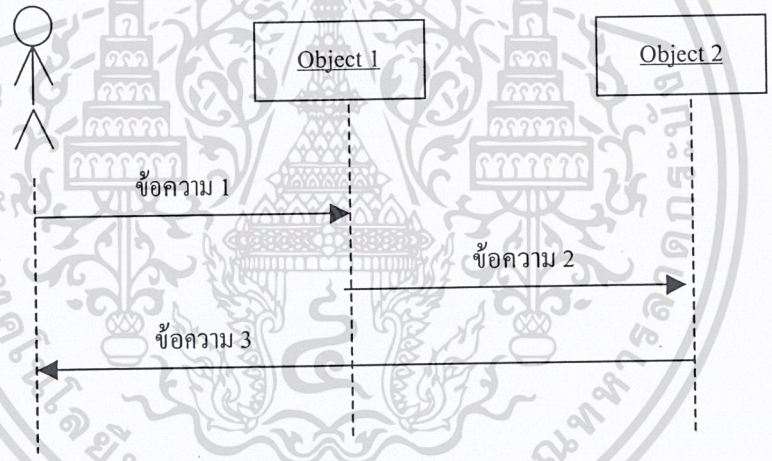
1. Sequence Diagram
2. Collaboration Diagram
3. State Diagram

2.3.5.1 Sequence Diagram

Sequence Diagram เป็นการแสดงข้อความและลำดับของการส่งข้อความระหว่างออบเจ็กต์

ข้อความ คือ หน่วยที่ออบเจ็กต์ใช้ในการสื่อสารกัน ตัวอย่างของข้อความที่เกิดขึ้น เช่น เมื่อผู้ใช้ร้องขอบริการ, ออบเจ็กต์ร้องขอบริการจากออบเจ็กต์อื่นหรือออบเจ็กต์ส่งคืนข้อมูลจากบริการที่ถูกร้องขอ

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

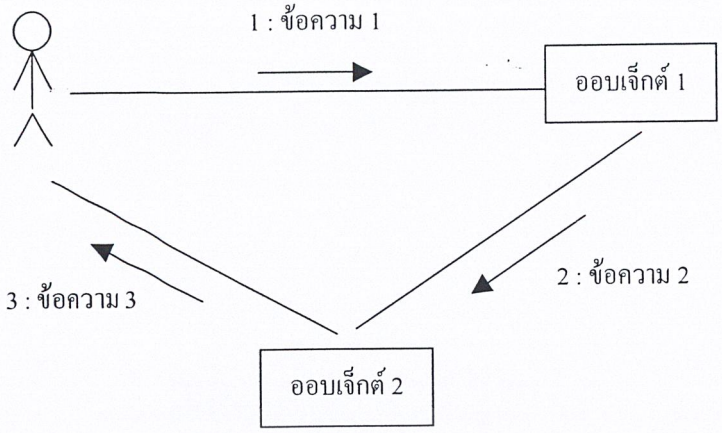


รูปที่ 2-11 Sequence Diagram

จากรูปไดอะแกรมข้างต้น เส้นในแนวตั้งจะหมายถึงออบเจ็กต์โดยมีชื่อของออบเจ็กต์เขียนอยู่ด้านบนหรือเป็นผู้กระทำที่ติดต่อกับระบบ ส่วนเส้นในแนวนอนนั้นจะแสดงถึงข้อความซึ่งมีชื่อของข้อความนั้นอยู่ ข้อความที่เกิดขึ้นจะเป็นลำดับตามแนวตั้งจากบนลงล่าง

2.3.5.2 Collaboration Diagram

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



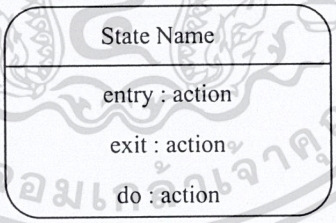
รูปที่ 2-12 Collaboration Diagram

จากรูปไดอะแกรมข้างต้น จะแสดงการไหลของข้อความระหว่างออบเจ็กต์ด้วยกันหรือระหว่างออบเจ็กต์กับผู้กระทำ โดยข้อความที่ใช้ในการติดต่อกันจะประกอบด้วยหมายเลขลำดับ,ชื่อข้อความและทิศทางของข้อความ

2.3.5.3 State Diagram

สถานะของออบเจ็กต์ปกติจะถูกกำหนดโดยค่าของแอตทริบิวต์ หรือค่าตัวแปรภายใน เช่น ออบเจ็กต์หลอดไฟจะมีสถานะคือเปิดและปิด ซึ่งสถานะเปิดและปิดนี้จะเป็นค่าของแอตทริบิวต์ สถานะของออบเจ็กต์หลอดไฟ

State Diagram แสดงถึงสถานะต่างๆของออบเจ็กต์และการส่งผ่าน (transition) ที่ทำให้เกิดการเปลี่ยนแปลงของสถานะเหล่านั้น และแสดงถึงกิจกรรมที่ออบเจ็กต์กระทำเมื่ออยู่ในสถานะหนึ่งๆ



รูปที่ 2-13 รูปแบบของสถานะ

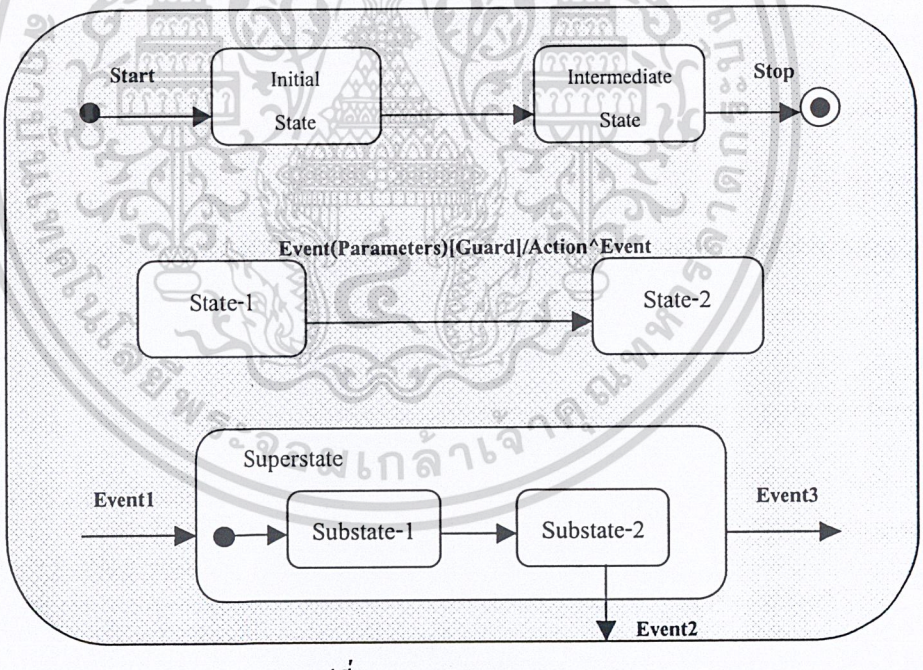
State Diagram สามารถมีสถานะซ้อนภายในสถานะอีกทีได้ ตัวสถานะที่อยู่ภายนอกนั้นเราจะเรียกว่า Superstate ส่วนสถานะที่อยู่ภายในเราจะเรียกว่า Substate ในแต่ละสถานะจะมีการระบุถึงการกระทำที่ออบเจ็กต์จะกระทำเมื่ออยู่ในสถานะนั้น โดยสามารถแบ่งการกระทำได้เป็น

- 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	เป็นรายการของโอเปอเรชั่นที่จะ กระทำเมื่อเกิดการเปลี่ยนสถานะ โดยโอเปอเรชั่นเหล่านี้อาจจะเป็นของออบเจ็กต์เองหรือของออบเจ็กต์อื่นก็ได้
Event list	เป็นรายการของ events ที่ถูกสร้างขึ้นเมื่อมีการเปลี่ยนสถานะ โดย events เหล่านี้สามารถกระจายไปยัง state machines อื่นๆ เพื่อเป็นการทำงานแบบพร้อมกันได้

ตารางที่ 2-1 รูปแบบการส่งผ่าน



รูปที่ 2-14 State Diagram

บทที่ 3
การออกแบบด้วยวิธีเชิงวัตถุ

จุดมุ่งหมายของการออกแบบ Visual Modeling Technique (VMT) คือการคิดหาทางแก้ปัญหาและสร้างสถาปัตยกรรมทั้งหมดที่เป็นพื้นฐานสำหรับการสร้างเพื่อนำไปแก้ปัญหาระบบ จากนั้นเราจะทำการสำรวจวิธีการ และสร้างยุทธวิธีในการตัดสินใจเพื่อแปลงการออกแบบไปเป็นองค์ประกอบของการแก้ปัญหา

ในขั้นตอนของการออกแบบ จะแบ่งการออกแบบเป็นส่วนต่างๆ ดังนี้

- การออกแบบระบบ (System Design)

  เป็นการออกแบบระบบในระดับสูง โดยกำหนดสถาปัตยกรรมของซอฟต์แวร์และจัดโครงสร้างของระบบซึ่งจะแบ่งระบบที่มีขนาดใหญ่ออกให้เป็นระบบขนาดย่อย

- การออกแบบออบเจ็กต์ (Object design)

  เป็นการกลั่นกรองโมเดลที่ได้จากการวิเคราะห์ระบบและปรับให้เข้ากับการทำงานในสิ่งแวดล้อมที่มีอยู่ โดยการกำหนดรายละเอียดของของคลาสของออบเจ็กต์เพิ่มขึ้นและสร้างเป็น Design Model ของระบบ

- การออกแบบออบเจ็กต์คงอยู่ (Designing for object persistence)

  เป็นการออกแบบการเก็บรักษาออบเจ็กต์โดยเกี่ยวข้องกับการสร้างโมเดลและการออกแบบเซิฟเวอร์ฐานข้อมูล รวมถึงการแปลงคลาสของออบเจ็กต์ไปเป็นตารางฐานข้อมูลและกำหนดนโยบายการเข้าถึงข้อมูลเพื่อทำการแก้ไข หรือเปลี่ยนแปลงค่าของข้อมูลที่ใช้งานร่วมกัน

3.1 การออกแบบระบบ (System Design)

การออกแบบระบบเป็นการออกแบบสถาปัตยกรรมในระดับสูง (high-level) ของวิธีการแก้ปัญหาที่วางแผนไว้ โดยสถาปัตยกรรมของระบบจะกำหนดกลุ่มงานหลักๆของระบบและการเชื่อมต่อกันในระดับสูง กลุ่มของระบบนี้จะแสดงถึงฟังก์ชันการทำงานของระบบ สถาปัตยกรรมของแอพพลิเคชัน (Application Architecture) กำหนดโครงสร้างของวิธีการแก้ปัญหาโดยแบ่งเป็นระบบย่อย (subsystem) แสดงกลุ่มงาน โดยทั้งสถาปัตยกรรมระบบและสถาปัตยกรรมแอพพลิเคชันแบ่งรายละเอียดเป็น 3 ระดับ คือ แนวคิด (conceptual) , ลอจิคอล (logical) และ กายภาพ (physical)

Use Case Model , Object Model และ Dynamic Model จะใช้แสดงสถาปัตยกรรมทางแนวคิด (Conceptual Architecture)

การออกแบบระบบเป็นการเริ่มต้นพิจารณาในขั้นลอจิคอล ในขั้นนี้จำเป็นต้องมีการตัดสินใจสำหรับการจัดหาข้อมูล, กระบวนการ , แพลตฟอร์ม (platform) ที่ใช้ในการสร้างระบบ โดยพิจารณาจาก

เทคโนโลยีที่มีอยู่และสภาพแวดล้อมปัจจุบัน เช่น ระบบเก่า การตัดสินใจเป็นการเลือกเทคโนโลยีและผลิตภัณฑ์ที่ใช้ในการสร้างกลุ่มงานหลักๆของระบบ

อย่างไรก็ตามผู้ออกแบบจำเป็นที่จะต้องสร้างสถาปัตยกรรมทางกายภาพ (Physical Architecture) ซึ่งแสดงการจัดตั้งของระบบตามลอจิคอลและคอมโพเนนท์ของแอพพลิเคชันตามวิธีแก้ปัญหาไปเป็นแพลตฟอร์มการสร้างเช่นเดียวกับหน่วยย่อยของโปรแกรม (programming module)

การออกแบบสถาปัตยกรรมของระบบเป็นการสร้างแนวความคิดจากโมเดลต่างๆ ที่ได้มาจากขั้นตอนการวิเคราะห์ระบบ เช่น Use Case Model , Object Model และ Dynamic Model เพื่อหาทางแก้ปัญหาของระบบ โดยระบบสามารถแบ่งได้เป็นสถาปัตยกรรมทางกายภาพและสถาปัตยกรรมซอฟต์แวร์ ซึ่งในการออกแบบสถาปัตยกรรมระบบนี้ เราจะแปลงซอฟต์แวร์คอมโพเนนท์ขนาดใหญ่ เช่น ระบบย่อย, แพ็กเกจ (packages) และงาน (tasks) ไปยังหลายๆตัวประมวลผลและอุปกรณ์ต่างๆของระบบ สำหรับระบบที่มีขนาดไม่ใหญ่มากนัก เราอาจจะข้ามขั้นตอนของการออกแบบในระดับของสถาปัตยกรรมนี้ได้ แต่สำหรับระบบที่มีขนาดใหญ่ที่มีการทำงานแบบกระจายนั้น ผู้พัฒนาระบบจำเป็นต้องมีการออกแบบระบบในระดับของสถาปัตยกรรมด้วย

3.2 การออกแบบออบเจ็กต์ (Object Design)

การออกแบบออบเจ็กต์เป็นการกลั่นกรองรายละเอียดของออบเจ็กต์ใหม่อีกครั้ง จากขั้นตอนของการออกแบบระบบที่ได้แบ่งโปรแกรมออกเป็นระบบย่อย เราจะกำหนดงานในแต่ละระบบย่อยให้แก่กลุ่มของนักพัฒนา โดยที่แต่ละกลุ่มนั้นจะมีการทำงานที่อิสระต่อกัน จนกว่าจะถึงขั้นตอนของการรวมระบบ

ในระหว่างขั้นตอนการวิเคราะห์ สำหรับแต่ละคลาสที่อยู่ใน Object Model เราจะกำหนดแอตทริบิวต์ (ซึ่งได้มาจากข้อมูลในตัวออบเจ็กต์ สำหรับการดูแลรักษาระบบ) และโอเปอเรชั่น สำหรับออบเจ็กต์ที่ถูกเรียกใช้บริการจากคลาสอื่นๆ และ Object Model จะมีการแสดงความสัมพันธ์แบบ associations ระหว่างคลาส

จากข้อมูลเหล่านี้ เราจะทำการออกแบบต่อโดยการกำหนดคลาสและกำหนดรายละเอียดของคลาสในแต่ละระบบย่อย

3.2.1 การสร้างโมเดลในขั้นตอนการออกแบบออบเจ็กต์

การสร้างโมเดล ในขั้นตอนการออกแบบออบเจ็กต์ใช้เทคนิคเดียวกับในขั้นตอนการวิเคราะห์ เทคนิคนี้รวมไปถึงการทำ CRC card , Event Trace Diagram , State Diagram และ Object Model แต่รายละเอียดจะเน้นไปในวิธีการแก้ปัญหามากกว่าตัวปัญหา

Design CRC แสดงหน้าที่และความขึ้นต่อกันของแต่ละออบเจ็กต์ แสดงแอตทริบิวต์,ข้อความและเมธอด ช่วยให้เข้าใจหน้าที่ของแต่ละออบเจ็กต์ได้ถูกต้องมากขึ้น

Design Event Trace Diagram หน้าที่ของออบเจ็กต์แสดงในเทอมของการให้บริการซึ่งกระทำผ่านโอเปอเรชั่นของออบเจ็กต์ รายละเอียดที่แสดงในไดอะแกรมควรมีคำอธิบายหน้าที่ของโอเปอเรชั่นว่าโอเปอเรชั่นนี้ทำอะไร , โอเปอเรชั่นนี้จะเริ่มทำงานได้อย่างไร , ชื่อโอเปอเรชั่น , อินพุต/เอาท์พุตพารา

มิเตอร์และชนิดของอินพุต/เอาท์พุต ในขั้นตอนการวิเคราะห์เน้นที่เหตุการณ์ภายนอกที่สร้างจากผู้กระทำ แต่ในขั้นตอนการออกแบบจะเน้นที่เหตุการณ์และการทำงานร่วมกันของออบเจ็กต์ภายในระบบ เพื่อทราบรายละเอียดของการส่งข้อความระหว่าง solution - domain object

Design State Diagram ในขั้นตอนนี้จะเน้นที่การเปลี่ยนสถานะภายในออบเจ็กต์มากกว่าสถานะรวมที่เน้นในขั้นตอนการวิเคราะห์

Design Object Model จากขั้นตอนการวิเคราห์ นำ Object Diagram ที่ได้มาทำการขัดเกลาขึ้นโดยการ

- เพิ่ม/ลบคลาส โดยเพิ่มคลาสที่จำเป็นต่อแอพพลิเคชันเข้าไป ลบคลาสที่ไม่เหมาะสมออก อาจสร้างคลาสแม่ (superclass) หรือ Abstract Class เพื่อให้สนับสนุนการนำกลับมาใช้ใหม่ หรือการ genelization
- ปรับปรุงลำดับการสืบทอด (inheritance)
- กำหนดชนิด , ช่วงค่าที่ใช้ , ค่าเริ่มต้น ของแอตทริบิวต์เพื่อเตรียมพร้อมนำไปสร้าง
- พิจารณาความสัมพันธ์ระหว่างคลาสให้เหมาะสมยิ่งขึ้น
- กำหนด multiplicity สำหรับ association

อย่างไรก็ตามการออกแบบที่ถูกต้องไม่ได้มาจากการวนพัฒนาเพียงครั้งเดียว บางครั้งข้อผิดพลาดอาจจะยังไม่พบจนกว่าจะทำเป็นต้นแบบ (prototype) ออกมา ในทางปฏิบัติการออกแบบและการทำต้นแบบมักมีการวนพัฒนาหลายครั้งจนกระทั่งได้การออกแบบที่พอใจ เพื่อให้การออกแบบดีขึ้นผู้พัฒนาอาจใช้แพทเทิร์น (pattern) หรือเฟรมเวิร์ค (framework)

Design pattern คือการจัดรูปแบบการแก้ปัญหาให้อยู่ในแนวทางเดียวกัน เช่น design pattern แบบ Model - View - Controller , Master - Slave , Microkernal , Proxy , Observer , เป็นต้น

Framework เป็นกลุ่มของคลาสที่ได้ออกแบบมาเพื่อการนำกลับไปใช้ใหม่ในแอพพลิเคชันเฉพาะประเภทเท่านั้น เฟรมเวิร์คจะวางโครงสร้างของระบบโดยแบ่งออกเป็นออบเจ็กต์, คลาส, หน้าที่รับผิดชอบหลัก, วิธีทำงานร่วมกันของคลาสและออบเจ็กต์ รวมทั้งกระบวนการควบคุมการทำงานทั้งหมดของระบบ เฟรมเวิร์คจะกำหนดรูปแบบในการออกแบบแอพพลิเคชัน ทำให้ผู้พัฒนาสามารถสร้างระบบขึ้นมาจากส่วนประกอบหลักๆที่เฟรมเวิร์คมีให้ ช่วยให้ผู้พัฒนาสามารถพัฒนาแอพพลิเคชันได้เร็วขึ้นและยังสามารถดูแลรักษาซอฟต์แวร์ได้ง่ายขึ้น การเปลี่ยนแปลงโครงสร้างของเฟรมเวิร์คจะมีผลกระทบต่อโครงสร้างของแอพพลิเคชันด้วย ดังนั้นในการออกแบบเฟรมเวิร์คจึงควรมีการนำแพทเทิร์นไปประยุกต์ใช้หลาย ๆ แบบเพื่อให้เฟรมเวิร์คมีความยืดหยุ่นและสามารถขยายฟังก์ชั่นเพิ่มเติมได้ ความแตกต่างระหว่างแพทเทิร์นและ เฟรมเวิร์คมีดังนี้

- แพทเทิร์นเป็นสิ่งที่อยู่ในทางความคิดมากกว่าเฟรมเวิร์ค เนื่องจากเฟรมเวิร์คจะมีโค้ดมาให้ใช้ทำให้มีความพร้อมในการนำไปใช้หรือการนำกลับไปใช้ แต่แพทเทิร์นผู้พัฒนาจะต้องสร้างโค้ดขึ้นมาเองเมื่อต้องการใช้
- แพทเทิร์นมีส่วนประกอบเล็กกว่าเฟรมเวิร์ค โดยทั่วไปเฟรมเวิร์คจะประกอบด้วยหลายๆ แพทเทิร์น

- แพทเทิร์นมีความเจาะจงน้อยกว่าเฟรมเวิร์ค โดยทั่วไปเฟรมเวิร์คนั้นได้รับการออกแบบและพัฒนาขึ้นมาสำหรับงานเฉพาะประเภทเท่านั้น แต่แพทเทิร์นสามารถนำไปใช้กับแอพพลิเคชันได้อย่างกว้างขวาง

3.2.2 MVC Design Pattern

MVC ประกอบด้วยออบเจ็กต์ 3 ชนิด : Model ประกอบด้วยออบเจ็กต์หลักของแอพพลิเคชัน , View เป็นการนำเสนอข้อมูลผ่านหน้าต่างส่วนติดต่อผู้ใช้และ Controller กำหนดวิธีการทำงานร่วมกันระหว่างส่วนติดต่อผู้ใช้กับอินพุตจากผู้ใช้

แนวคิดหลักของ MVC ก็คือการแยก Model (ออบเจ็กต์หลักของแอพพลิเคชัน) ซึ่งมีความสัมพันธ์ใกล้ชิดกับโดเมนธุรกิจออกจาก View โดยใช้ Controller ดูแลการใช้งานจากผู้ใช้ และเป็นตัวกลางระหว่างออบเจ็กต์ส่วนติดต่อกับออบเจ็กต์ของโมเดล เหตุผลหลักของการแยกคือในขณะที่ออบเจ็กต์ของโมเดลมีการเปลี่ยนแปลงน้อยแต่ View อาจมีการเปลี่ยนแปลงได้บ่อยตามความต้องการของผู้ใช้และออบเจ็กต์ของโมเดลหนึ่งสามารถสัมพันธ์กับ View ได้หลาย View ระบบที่สนับสนุน MVC จำเป็นต้องมีเฟรมเวิร์คซึ่งรับบทบาทเป็น Controller คอยอำนวยความสะดวกการสื่อสารระหว่าง View และ Model

ในการวิเคราะห์ระบบ เราควรจะมีความอิสระจากขั้นตอนการสร้าง เช่น ภาษาในการเขียนโปรแกรมและรูปแบบการทำงาน แต่ในความเป็นจริงแล้วออบเจ็กต์ที่ถูกออกแบบมานั้นจะขึ้นอยู่กับโครงสร้างของภาษา ดังนั้นการออกแบบระบบจึงต้องพิจารณาถึงเป้าหมายของสิ่งแวดล้อมในการทำงานของระบบด้วย

3.3 การออกแบบออบเจ็กต์คงอยู่ (Designing for object persistence)

หากต้องการเก็บออบเจ็กต์ที่ใช้ในการเอ็กซีคิวต์โปรแกรม ในอนาคตจะต้องทำการเก็บไว้ในตัวกลางถาวรซึ่งอาจมีวิธีการจัดการที่แตกต่างออกไป ออบเจ็กต์เหล่านี้เรียกว่าออบเจ็กต์คงอยู่ (Persistent Object)

ผู้พัฒนาสามารถพบทางเลือกในการเก็บและเรียกดูข้อมูลได้หลายวิธี โดยมีระบบจัดการฐานข้อมูล (Database Management Systems) หลายชนิดที่สามารถนำมาใช้ได้ ในโครงงานนี้จะเลือกใช้ฐานข้อมูลแบบรีเลชันแนล (Relational Database) ของออราเคิล

3.3.1 Normal Form

Normal Form เป็นแนวทางการออกแบบฐานข้อมูลรีเลชันแนล ช่วยให้ข้อมูลมีความสอดคล้องกันมากขึ้น

First Normal Form มีข้อกำหนดว่าข้อมูลที่อยู่ในแต่ละฟิลด์ (field) ของตารางจะต้องมีเพียงค่าเดียว ไม่มี repeating group ของ multiple value

Second Normal Form ต้องผ่าน First Normal Form และแอตทริบิวต์ทุกตัวต้องขึ้นอยู่กับ primary key

Third Normal Form ตารางต้องผ่าน Second Normal Form และต้องไม่มีแอตทริบิวต์ใดขึ้นอยู่กับแอตทริบิวต์ที่ไม่ใช่ primary key

เราสามารถออกแบบตารางให้ได้ถึง Fifth Normal Form แต่โดยทั่วไปแล้วถึงขั้น Third Normal Form ก็เพียงพอแล้ว

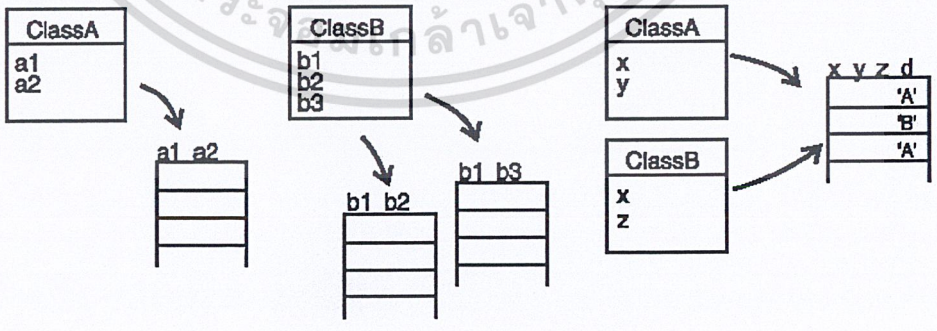
3.3.2 การแมปคลาสเป็นตาราง (Mapping Class to Table)

หลักการแปลงคลาสไปเป็นตารางคือเราจะแปลงแต่ละคลาสไปเป็นหนึ่ง(หรืออาจมากกว่าหนึ่ง)ตารางโดยแต่ละแอตทริบิวต์ของออบเจ็กต์จะแปลงไปเป็นคอลัมน์ของตาราง ซึ่งชนิดของแอตทริบิวต์นั้นจะต้องสามารถเก็บไว้อยู่ในฐานข้อมูลแบบรีเลชันแนลได้

ในแต่ละตารางจะมี primary key ซึ่งเหมือนกับเป็นตัว unique identifier ของออบเจ็กต์ที่เก็บอยู่ในตาราง การแปลงคลาสไปเป็นตารางนั้นเราสามารถจะทำได้โดย แมปจากแอตทริบิวต์ของคลาส, แมปจากความสัมพันธ์ของคลาส ซึ่งจะมีวิธีการแมปดังนี้

3.3.2.1 แอตทริบิวต์ (Attribute mapping)

แอตทริบิวต์ต่างๆของคลาส สามารถแมปไปเป็น 1 หรือหลายคอลัมน์ได้ใน 1 ตารางหรือ หลายตารางด้วยก็ได้



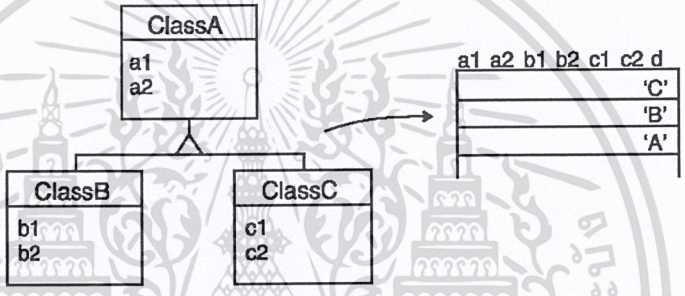
รูปที่ 3-1 แสดง การแมปจากคลาส เป็นตาราง

จากรูปซ้ายไปขวาเป็นการแสดงให้เห็นการแมปจาก คลาส A ที่ประกอบไปด้วยแอตทริบิวต์ a1 และ a2 เป็น 1 ตาราง ส่วนรูปถัดมาก็เป็นการแมปจาก คลาส B คลาส เดียวมี 3 แอตทริบิวต์ b1, b2 และ b3 แยกออกมาเป็น 2 ตาราง b1 กับ b2 และ b1 กับ b3 ส่วนรูปสุดท้ายจาก คลาส A และ คลาส B ที่มี แอตทริบิวต์เหมือนกันคือ x สามารถรวมกันเป็นตารางเดียวได้ รูปแบบของการแมป แอตทริบิวต์ นั้นเราจะต้องพิจารณาความสัมพันธ์ของ แอตทริบิวต์ ด้วยว่าควรแมปออกมาเป็นแบบใด

3.3.2.2 การแมปโดยมองที่การสืบทอดของคลาส (Inheritance mapping)

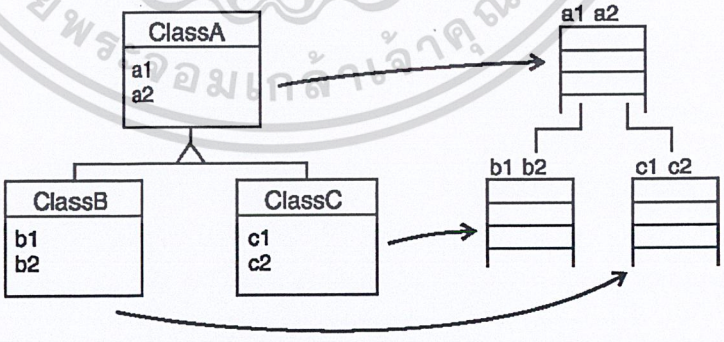
ความสัมพันธ์ของ คลาส เราสามารถแมปออกมาเป็นตารางเดียวหรือหลายตารางก็ได้แบ่งออกเป็น 2 ประเภท

- Typed partitioning เป็นความสัมพันธ์แบบลำดับชั้นของคลาส (Class hierarchy) ที่แมปออกมาเป็น ฐานข้อมูล 1 ตาราง



รูปที่ 3-2 แสดง การแมปจาก คลาส ทั้ง 3 ที่สืบทอดกันเป็น 1 ตาราง

- Vertical partitioning เป็นความสัมพันธ์แบบลำดับชั้นของคลาส ที่แมปออกมาเป็นตาราง Root และ ตาราง Leaf



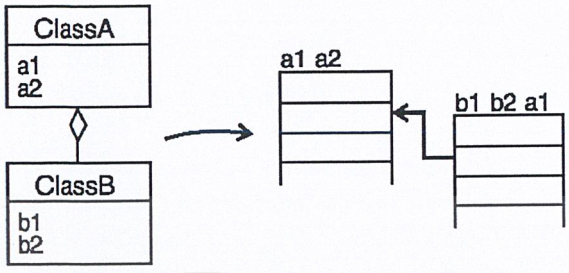
รูปที่ 3-3 แสดง การแมปจาก คลาส ทั้ง 3 ที่สืบทอดกันเป็น ตาราง Root และLeaf

3.3.2.3 การแมปโดยการมองที่ความสัมพันธ์ของคลาส (Relationship mapping)

ออบเจ็กต์ สามารถแมปเป็น one-to-one หรือ one-to-many relationships ได้

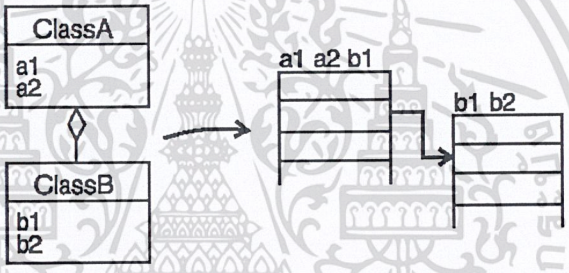
- One-to-one relationships สามารถแมปได้เป็น 2 ทางคือ

1. มองจากล่างขึ้นบน (Backward pointing)



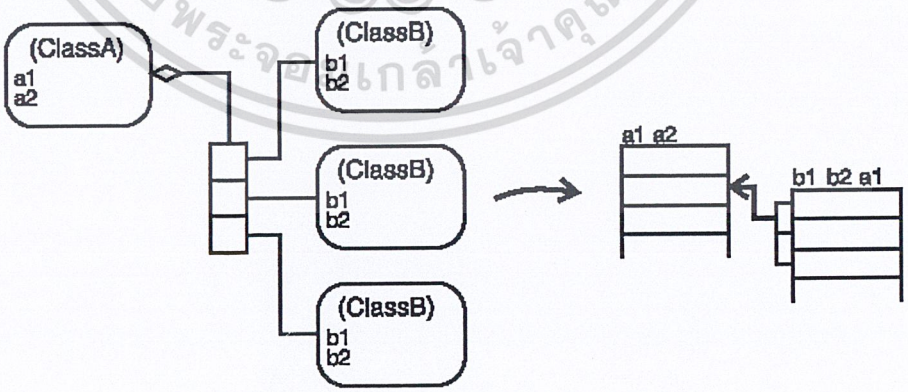
รูปที่ 3-4 แสดงการแมปคลาส ที่มีความสัมพันธ์กัน มาเป็นตารางโดยการมองจากล่างขึ้นบน

2. มองจากบนลงล่าง (Forward pointing)



รูปที่ 3-5 แสดงการแมปคลาส ที่มีความสัมพันธ์กัน มาเป็นตารางโดยการมองจากบนลงล่าง

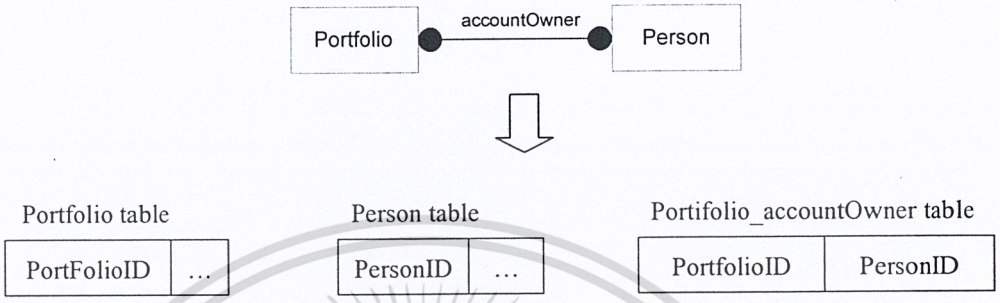
- One-to-many relationships



รูปที่ 3-6 แสดง ความสัมพันธ์แบบ 1 to many ที่แมป ออกมาเป็นตาราง

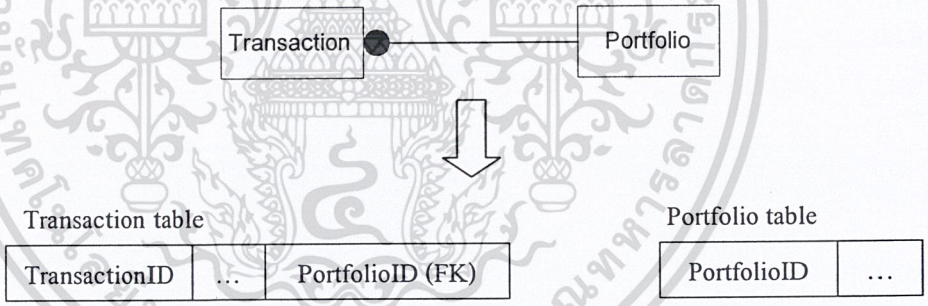
3.3.2.4 การแปลง Simple Associations

⊙ *แยกตารางสำหรับ many - to - many association* ควรแปลงแต่ละ many - to - many association เป็นตารางต่างหาก primary key ของความสัมพันธ์นี้ได้จากการรวม primary key จากแต่ละตาราง โดยลำดับของคลาสไม่มีผลกับ primary key และอาจมี non key ที่เป็นแอตทริบิวต์ของความสัมพันธ์มาประกอบด้วย



รูป 3-7 วิธีที่แนะนำในการแปลง many – to – many association

⊙ *รวม one - to - many association* นำ primary key ของ class ที่มี multiplicity เป็น 1 ไปเป็น foreign key ในตารางของคลาสที่มี multiplicity เป็น many



รูป 3-8 วิธีที่แนะนำในการแปลง one – to – many association

⊙ *รวม zero - or - one - exactly one association* นำ primary key ของ 1 ไปเป็น foreign key รวมกับตารางของคลาสที่เป็น 0 หรือ 1 โดยกำหนดให้ foreign key นั้นห้ามเป็น null

⊙ *รวม one - to - one association* นำ primary key ของคลาสใดคลาสหนึ่งไปเป็น foreign key ในตารางของอีกคลาสหนึ่ง foreign key จะเป็น null หรือไม่นั้นขึ้นอยู่กับ minimum multiplicity ตามที่กำหนดไว้

บทที่ 4
ระบบทำสัญญาซื้อขายบ้านจัดสรร

4.1 ประโยคปัญหา (Problem Statement)

ระบบทำสัญญาซื้อขายบ้านจัดสรรเป็นโปรแกรมที่เป็นตัวกลางในการพบปะระหว่างลูกค้ากับเจ้าของโครงการบ้านโดยทำสัญญาซื้อขายบ้านผ่านทางอินเทอร์เน็ต โดยมีฟังก์ชันการทำงานที่เกี่ยวกับการจอง ระบบสามารถคำนวณเงินผ่อนที่จะชำระในแต่ละงวดเพื่อช่วยในการตัดสินใจของลูกค้า,สามารถเก็บข้อมูลประวัติของลูกค้า,เก็บรายละเอียดโครงการบ้าน,ค้นหาข้อมูลโครงการที่ต้องการ,รายงานผลการนัดหมายวันมาทำสัญญาของลูกค้า,รายงานเกี่ยวกับโครงการบ้านที่มีอยู่ในระบบและเพิ่มข่าวสารในการซื้อขายได้

เมื่อลูกค้าทำการลงทะเบียนเพื่อจะมีสิทธิการจองบ้านภายในระบบได้ ระบบก็จะทำการเก็บข้อมูลประวัติลูกค้าเอาไว้

ถ้าลูกค้าตัดสินใจที่จะจองบ้านโดยคำนวณเงินผ่อนแต่ละงวดแล้ว ซึ่งการคำนวณเงินผ่อนในแต่ละงวดนั้นเราจะพิจารณาจากรายได้ของครอบครัวต่อเดือนและระยะเวลาที่ผ่อนชำระ โดยอ้างอิงกับข้อกำหนดทางธนาคารกรุงไทยจำกัด เมื่อได้ทำการจองบ้านแล้วต่อมาก็จะเป็นขั้นตอนของการนัดหมายทำสัญญา โดยระบบจะเป็นตัวกำหนดวันดังกล่าวจนลูกค้าพอใจและจะทำการส่งอี-เมล์ (E-mail) เพื่อจะการแจ้งผลการทำสัญญาและวันนัดหมาย บ้านที่ถูกจองแล้วลูกค้าคนอื่นจะไม่สามารถมาจองได้อีก

หากลูกค้าไม่มาทำสัญญาตามที่ได้นัดหมายเอาไว้โดยไม่ได้แจ้งเลื่อนล่วงหน้ามาก่อน ระบบจะทำการตัดสิทธิในจองของลูกค้าคนนั้นไปแล้วก็จะเปลี่ยนสถานะบ้านที่ถูกจองเป็นปกติเหมือนเดิม

4.2 รายละเอียดข้อมูล, การคำนวณเงินผ่อนในระบบการทำซื้อขายบ้านจัดสรร

ตารางเทียบอัตราส่วนที่ใช้ในการคำนวณเงินผ่อนในแต่ตละงวด โดยพิจารณาจาอัตราดอกเบี้ยเงินกู้และระยะเวลาการผ่อนชำระ จากเงินต้น 1 บาท

	7 %	7.5 %	8 %	8.5 %	9 %
5 ปี	0.019801	0.020038	0.020276	0.020517	0.020758
10 ปี	0.011611	0.011870	0.012133	0.012399	0.012668
15 ปี	0.008988	0.009270	0.009557	0.009847	0.010143
20 ปี	0.007753	0.008056	0.008364	0.008678	0.008997

ตารางที่ 4-1 ตารางเทียบอัตราส่วนที่ใช้ในการคำนวณเงินผ่อนคงที่ต่องวด

การคำนวณเงินผ่อนที่ต้องชำระในแต่ละเดือนจะหาได้จาก = ราคาบ้าน*สัดส่วนในช่องตาราง(ดูจากอัตราดอกเบี้ยและระยะเวลาที่ขอกู้)

ตัวอย่าง - ราคาบ้าน 1,000,000 ล้านบาท

- อัตราดอกเบี้ย 7.0 %
- กำหนดระยะเวลาผ่อนชำระ 10 ปี
- สัดส่วนในตาราง 0.011611

วิธีคิด จำนวนเงินผ่อนที่ต้องชำระในแต่ละเดือน = 1,000,000 * (0.011611)

= 11,611

ดังนั้น เงินที่ต้องชำระในแต่ละเดือน = 11,611 บาท

4.3 Use Case Model

4.3.1 กำหนดผู้กระทำ (Actors)

ผู้กระทำของระบบงานนี้คือ ลูกค้า,เจ้าของโครงการ,ผู้ดูแลระบบ เพราะเป็นผู้ที่ใช้งานและเกี่ยวข้องกับระบบโดยตรง

4.3.2 กำหนด Use cases

ลงทะเบียน ระบบจะทำการเก็บข้อมูลประวัติลูกค้าเพื่อให้มีสิทธิในการจองบ้านได้

ค้นหาโครงการ ลูกค้าสามารถสอบถามข้อมูลรายละเอียดที่เกี่ยวกับโครงการบ้านได้

คำนวณเงินผ่อน ประมาณเงินที่ต้องชำระในแต่ละงวดตามอัตราดอกเบี้ยและระยะเวลาผ่อนชำระเพื่อช่วยในการตัดสินใจของลูกค้าในการเลือกบ้าน

ทำสัญญา เมื่อลูกค้าได้ตัดสินใจในการเลือกบ้านแล้ว ลูกค้าจะต้องทำการจองบ้านกับระบบ โดยเมื่อเสร็จสิ้นการจองแล้วก็จะไปทำสัญญา ระบบจะทำการสรุปผลของการทำสัญญาและแจ้งวันที่นัดหมายให้ทราบผ่านทาง อี-เมล์ ซึ่งจะเป็นการนัดหมายในการพบปะซื้อขายบ้านกับเจ้าของโครงการโดยตรง

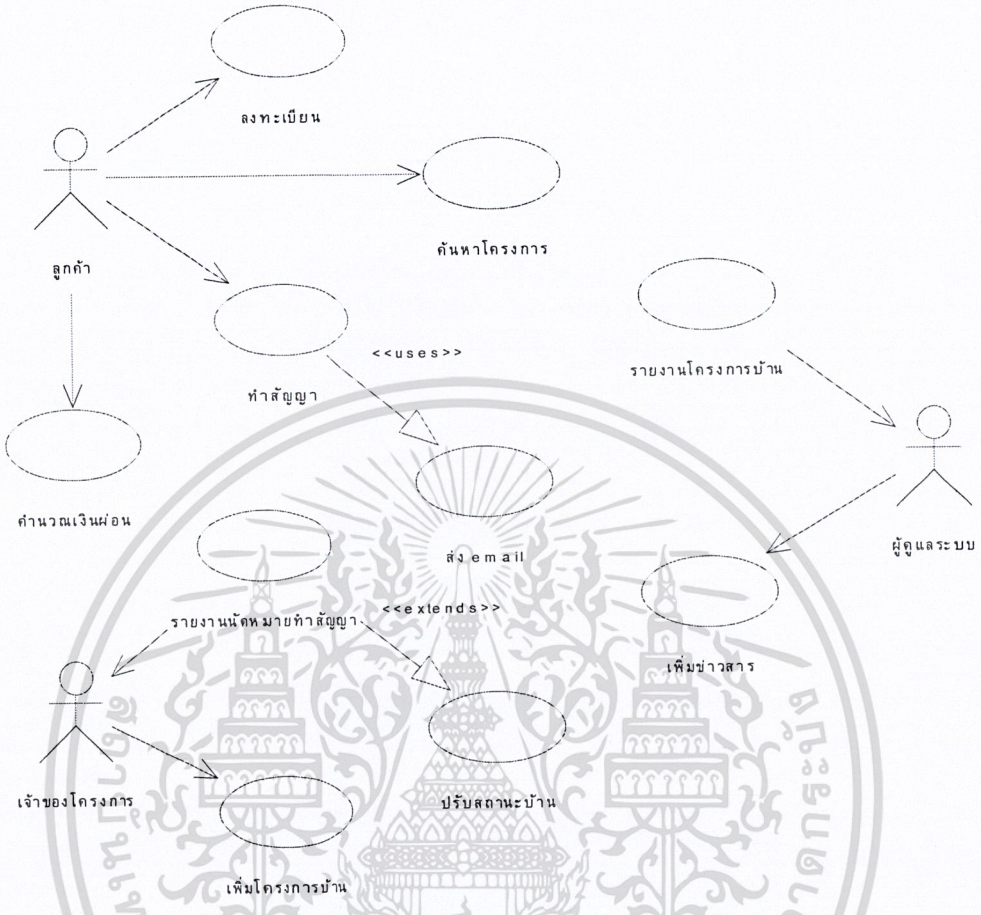
รายงานนัดหมายทำสัญญา แสดงให้เจ้าโครงการบ้านแต่ละโครงการได้ทราบถึง จำนวนลูกค้าที่เข้ามาจองโครงการกับวันที่ได้นัดหมายในการทำสัญญา

เพิ่มโครงการบ้าน เจ้าของโครงการสามารถเพิ่มรายละเอียดของบ้านในแต่ละโครงการของตนได้

รายงานโครงการบ้าน ผู้ดูแลระบบจำทราบถึงจำนวนและข้อมูลรายละเอียดต่าง ๆ ของโครงการบ้านที่อยู่ในระบบของตน

เพิ่มข่าวสาร จะเป็นการประกาศถึงข้อมูลข่าวสารให้แก่ลูกค้าได้รับทราบ ทั้งนี้เพื่อเป็นการโฆษณาโครงการใหม่ ๆ ซึ่งกำลังเกิดขึ้นไปด้วย

4.3.3 Use Case Diagram ของระบบทำสัญญาซื้อขายบ้านจัดสรร



รูป 4-1 Use Case Model ของระบบทำสัญญาซื้อขายบ้านจัดสรร

4.4 ต้นแบบส่วนติดต่อผู้ใช้

การสร้างต้นแบบของระบบโดยให้ผู้ใช้ทดลองใช้งานจากต้นแบบที่สร้างขึ้นมานี้ เพื่อใช้เป็นเครื่องมือช่วยในการหาความต้องของระบบได้อย่างถูกต้องมากยิ่งขึ้น และยังเป็นการหาส่วนที่เข้าใจผิดพลาดไปหรือส่วนที่ยังหาไม่พบในตอนแรก

4.5 Object Model

4.5.1 กำหนดออบเจ็กต์

ออบเจ็กต์ที่ได้จากประโยคปัญหาและจาก Use Cases มีดังนี้

ลูกค้า, เจ้าของโครงการ, ผู้ดูแลระบบ,อินเทอร์เน็ต,อี-เมล์,รายได้ครอบครัวต่อเดือน,อัตราดอกเบี้ย,ระยะเวลาการผ่อนชำระ,โครงการบ้าน,บ้าน,สัญญา,ประวัติลูกค้า,ธนาคารกรุงไทย

4.5.2 กำจัดออบเจ็กต์ที่ไม่จำเป็นและไม่ถูกต้อง

จากออบเจ็กต์ที่ได้ข้างต้น เราจะกำจัดคลาสที่ไม่จำเป็นและไม่ถูกต้องออก ดังนี้

4.5.2.1 ออบเจ็กต์ที่ซ้ำซ้อนกัน (Redundant Objects) กำจัดออบเจ็กต์ที่เกินความจำเป็น ซึ่งได้แก่

ลูกค้า,ประวัติลูกค้า,รายได้ครอบครัวต่อเดือน เป็น ลูกค้า

สัญญา,อัตราดอกเบี้ย,ระยะเวลาการผ่อนชำระ เป็น สัญญา

4.5.2.2 ออบเจ็กต์ที่ไม่เกี่ยวข้องกับระบบ (Irrelevant Objects) กำจัดออบเจ็กต์ที่อยู่นอกเหนือระบบได้แก่ ธนาคารกรุงไทย

4.5.2.3 ออบเจ็กต์ที่คลุมเครือ (Vague Object) กำจัดออบเจ็กต์ที่มีความหมายไม่ชัดเจน ซึ่งได้แก่ อินเทอร์เน็ต, อี-เมล์

4.5.2.4 ออบเจ็กต์ที่เป็นแอตทริบิวต์ของคลาส กำจัดออบเจ็กต์ที่เป็นคุณลักษณะของคลาส ซึ่งได้แก่ รายได้ครอบครัวต่อเดือน,อัตราดอกเบี้ย,ระยะเวลาการผ่อนชำระ,ประวัติลูกค้า

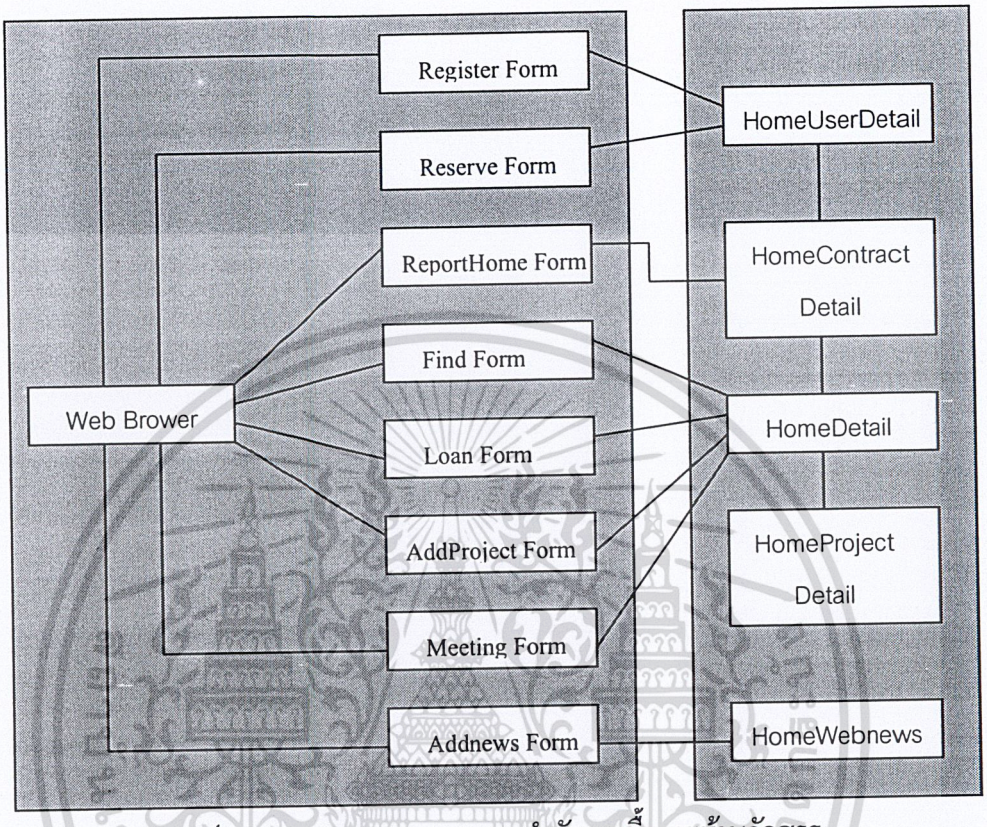
4.5.2.5 ออบเจ็กต์ที่เป็นโอเปอเรชั่นของคลาส กำจัดออบเจ็กต์ที่เป็นพฤติกรรมของคลาส

4.5.2.6 ออบเจ็กต์ที่เป็นหน้าที่ของคลาส กำจัดออบเจ็กต์ที่แสดงหน้าที่บางอย่างของคลาส

4.5.2.7 ออบเจ็กต์ที่สร้างจากระบบ (Implementation Construct) กำจัดออบเจ็กต์ที่เป็นสิ่งที่ถูกทำขึ้นจากระบบ

4.5.3 Object Model ของระบบทำสัญญาซื้อขายบ้านจัดสรร

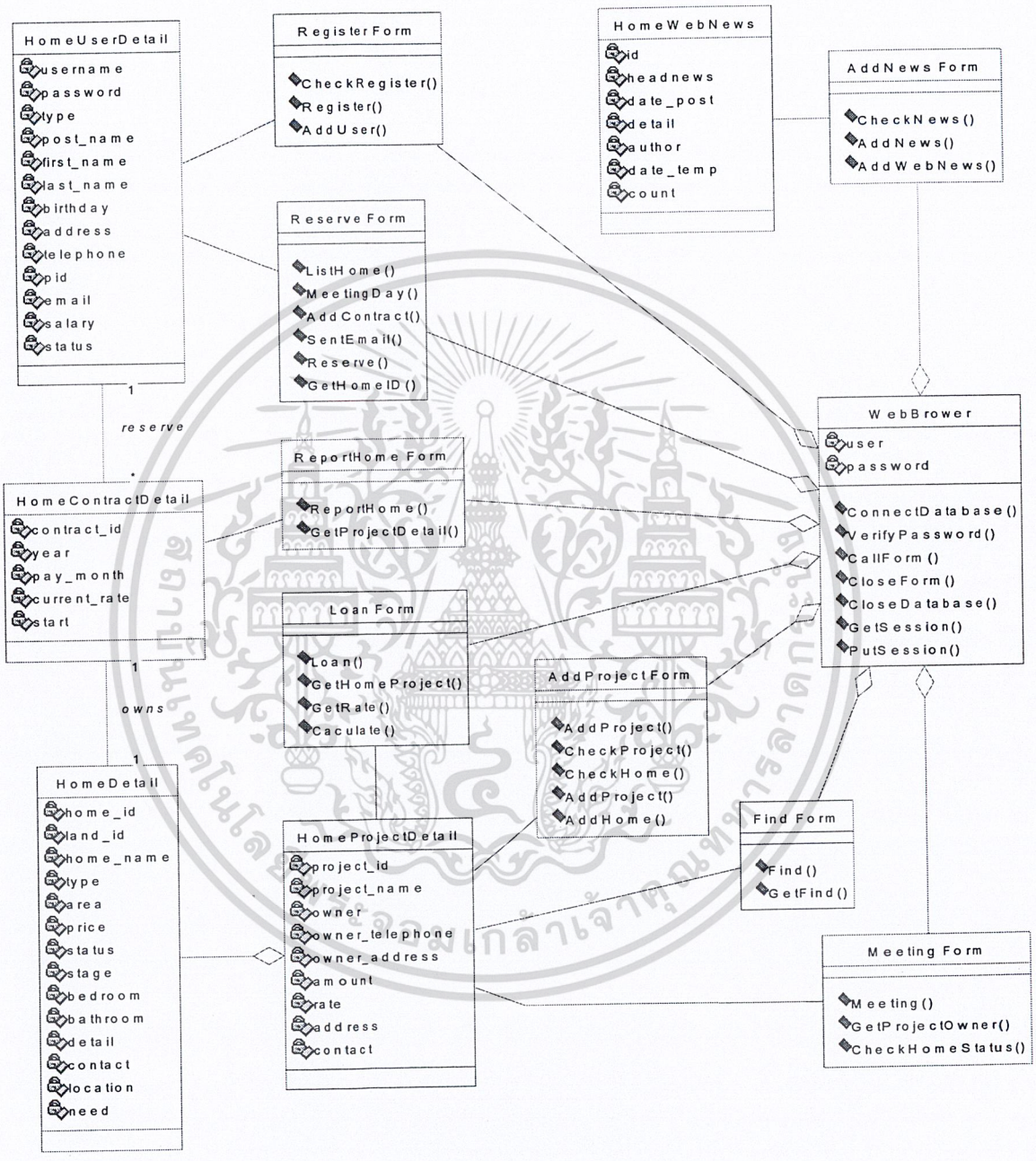
จากออบเจ็กต์ที่ได้นำมาเขียน Object Model ของระบบทำสัญญาซื้อขายบ้านจัดสรรโดยได้มีการพัฒนาแก้ไขโมเดลหลายครั้งเพื่อให้ได้โมเดลที่ตรงตามความต้องการของระบบมากที่สุด



รูป 4-2 Object Model ของระบบทำสัญญาซื้อขายบ้านจัดสรร

4.6 Class Diagram

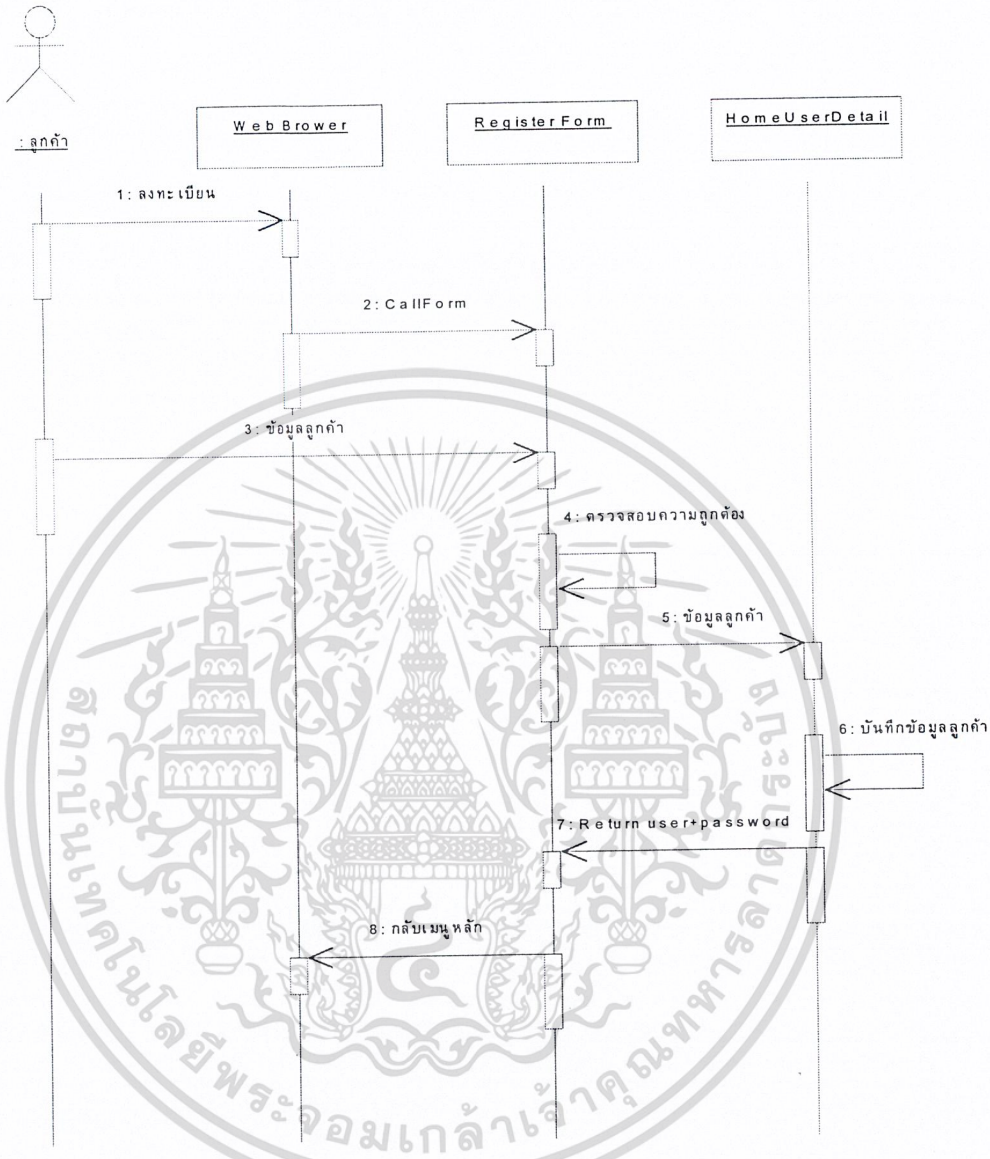
เมื่อทำการวิเคราะห์เพิ่มก็สามารถกำหนดคลาสของออบเจ็กต์ที่ได้ กำหนดแอตทริบิวต์,เมธอดและความสัมพันธ์ระหว่างคลาสต่าง ๆ เขียนเป็น Class Diagram ได้ดังต่อไปนี้



รูปที่ 4-3 Class Diagram ของระบบทำสัญญาซื้อขายบ้านจัดสรร

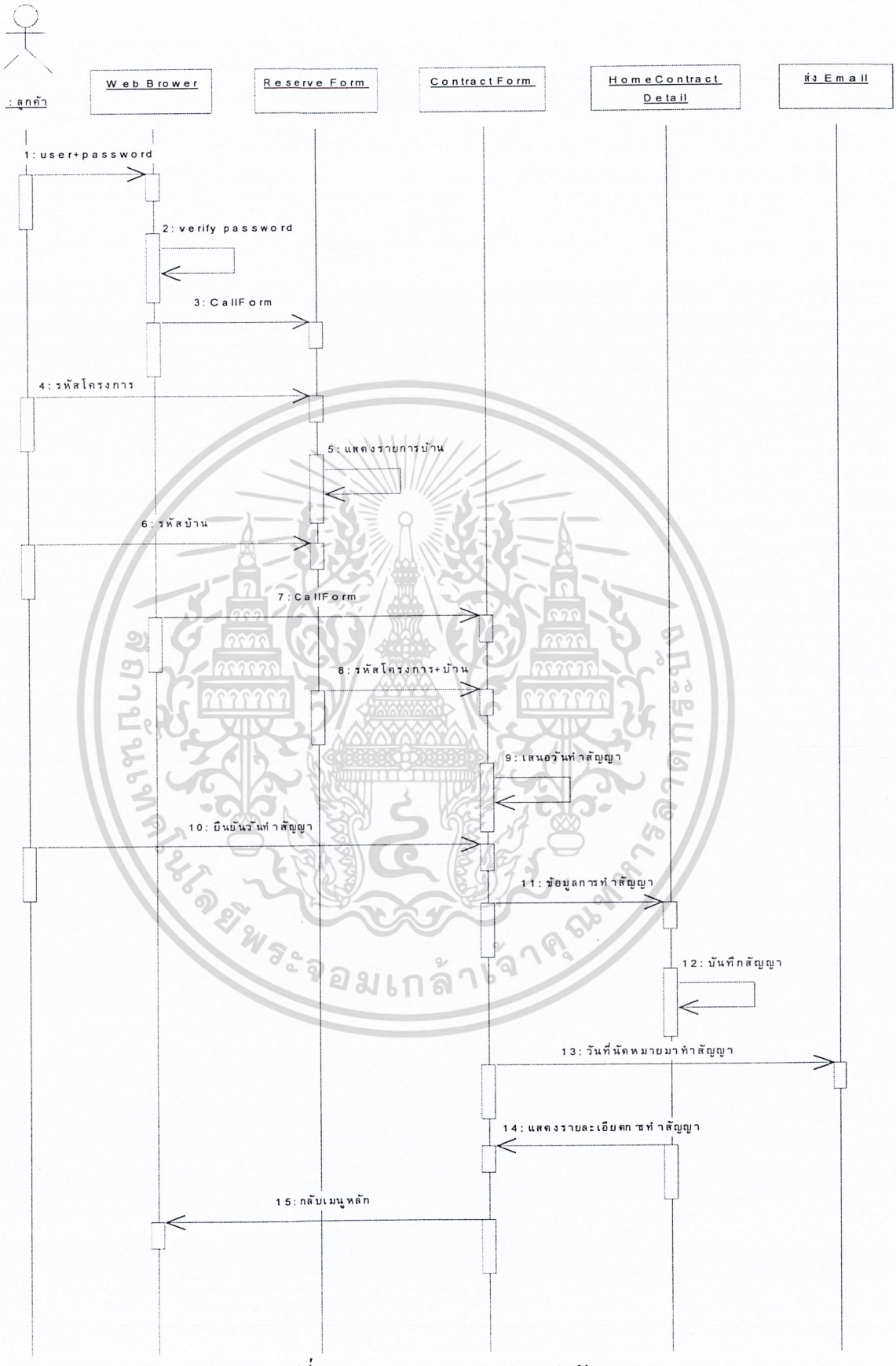
4.7 Dynamic Model

4.7.1 Sequence Diagram ลงทะเบียน



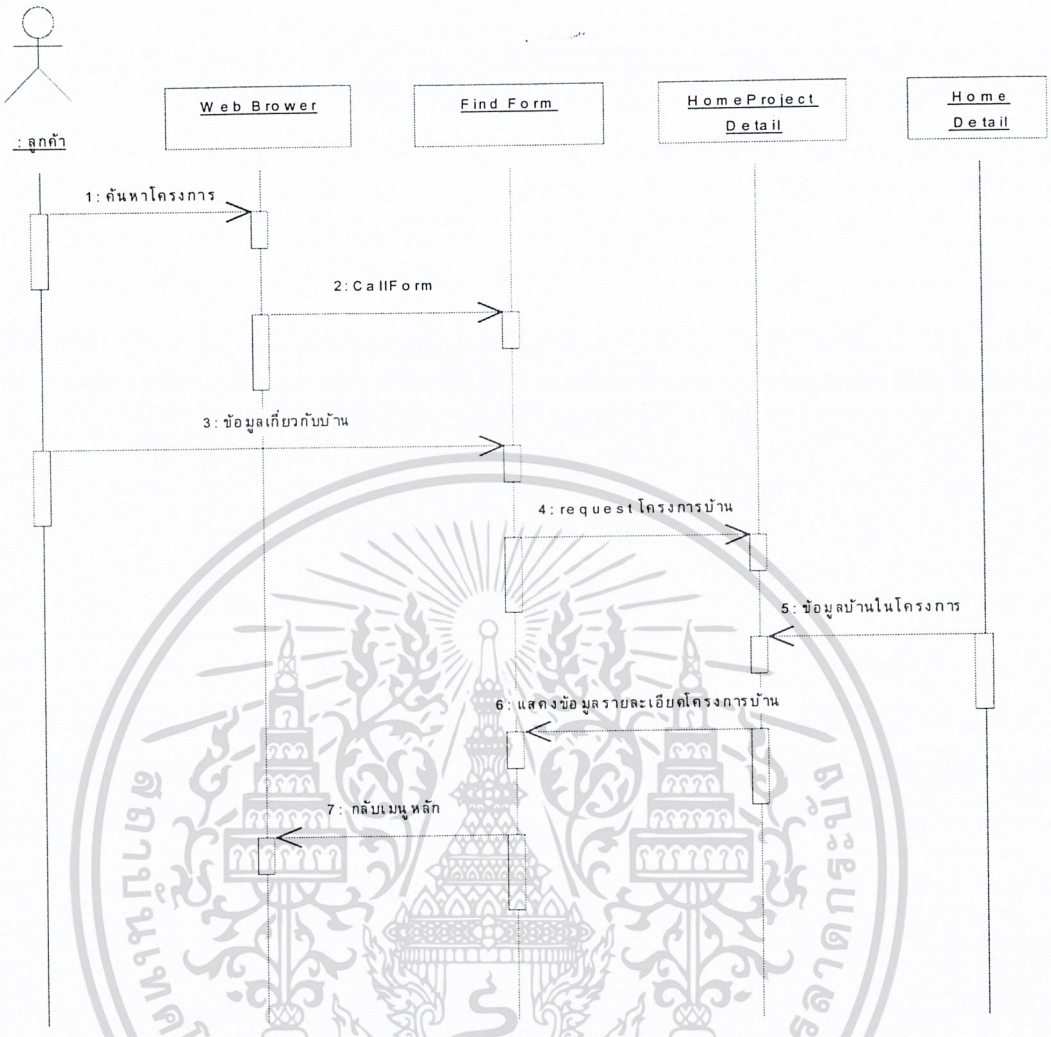
รูปที่ 4-4 Sequence Diagram ลงทะเบียน

4.7.2 Sequence Diagram ทำสัญญา



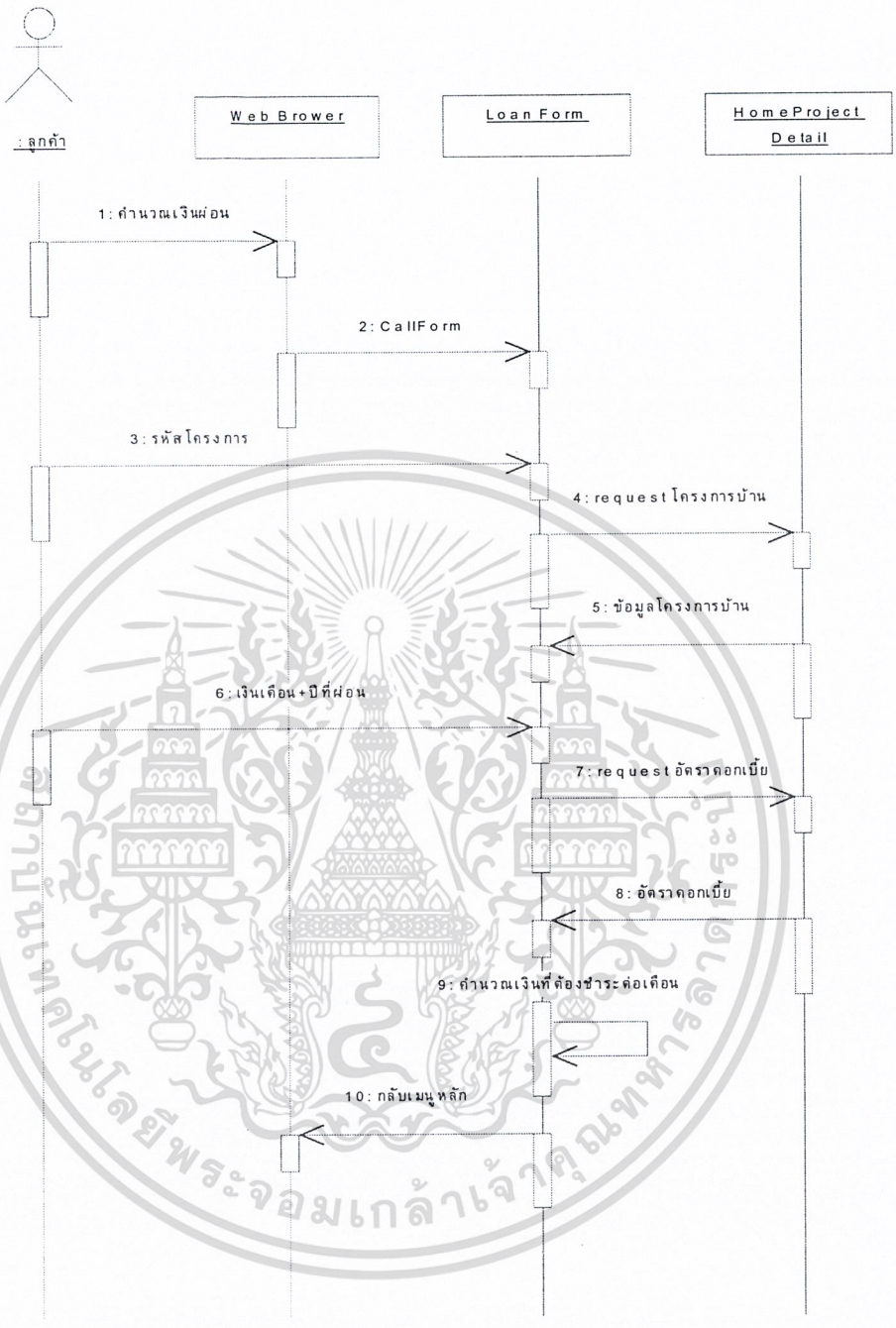
รูปที่ 4-5 Sequence Diagram การทำสัญญา

4.7.3 Sequence Diagram ค้นหาโครงการ



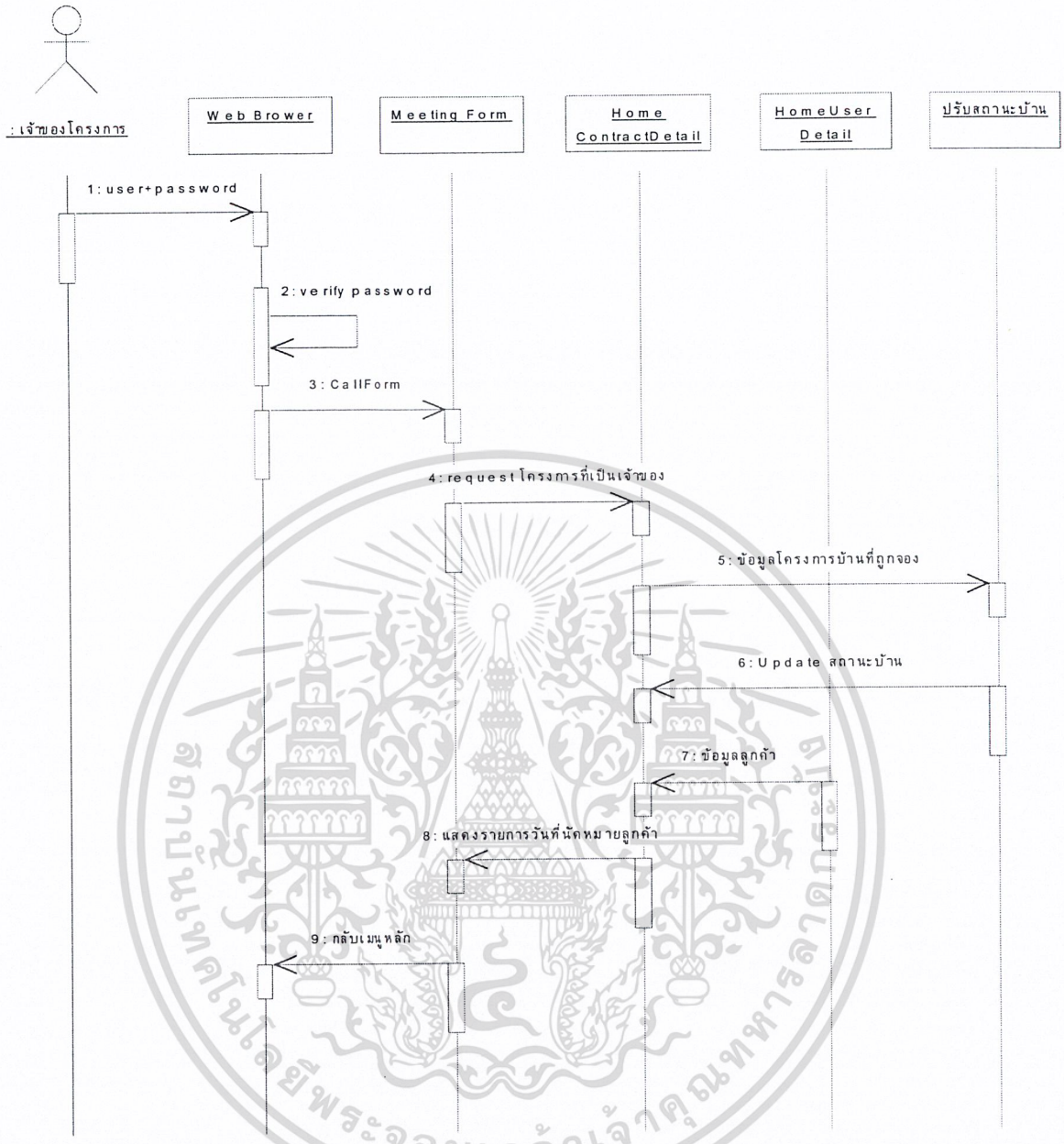
รูปที่ 4-6 Sequence Diagram ค้นหาโครงการ

4.7.4 Sequence Diagram คำนวณเงินผ่อน



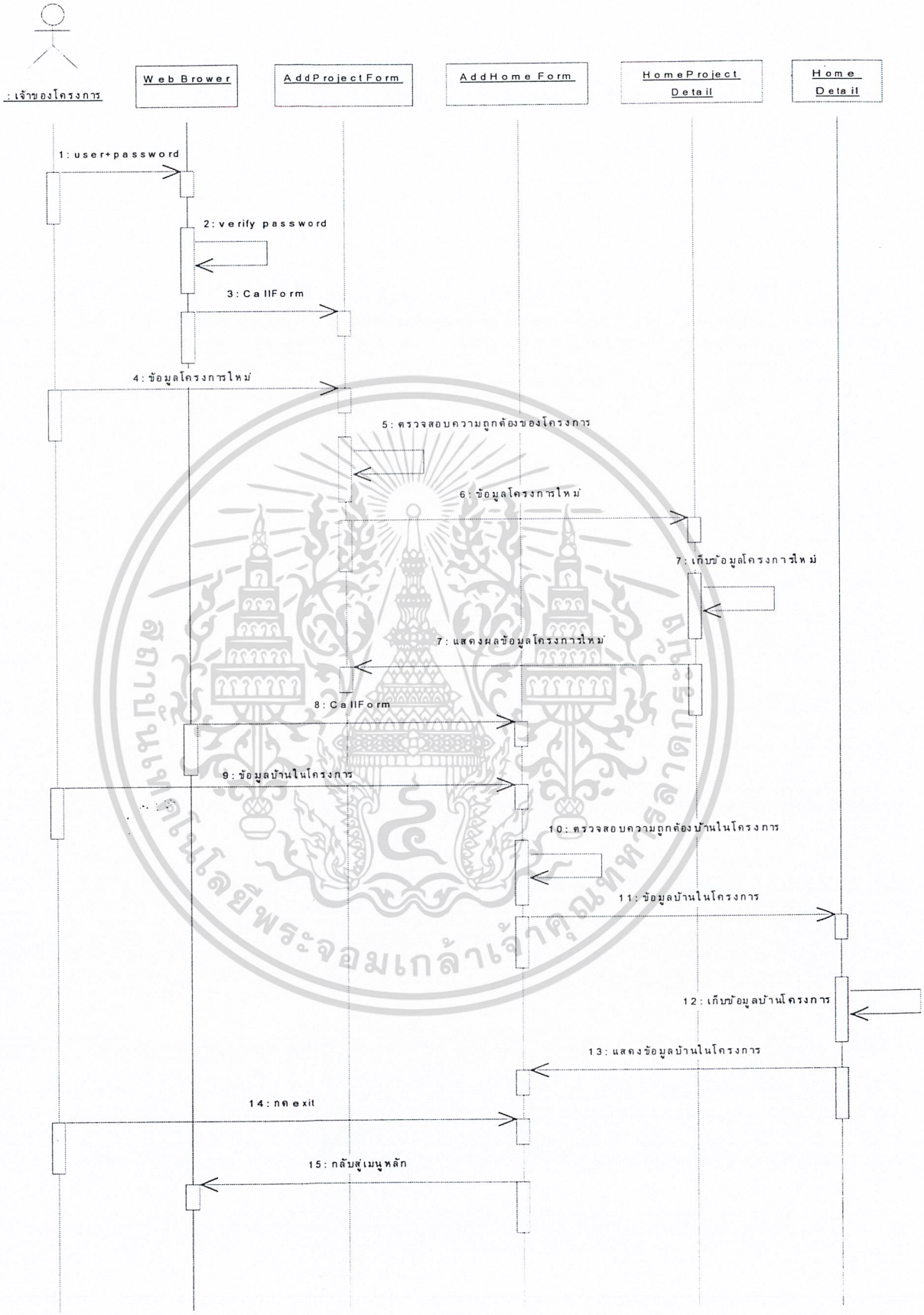
รูปที่ 4-7 Sequence Diagram คำนวณเงินผ่อน

4.7.5 Sequence Diagram รายงานนัดหมายทำสัญญา



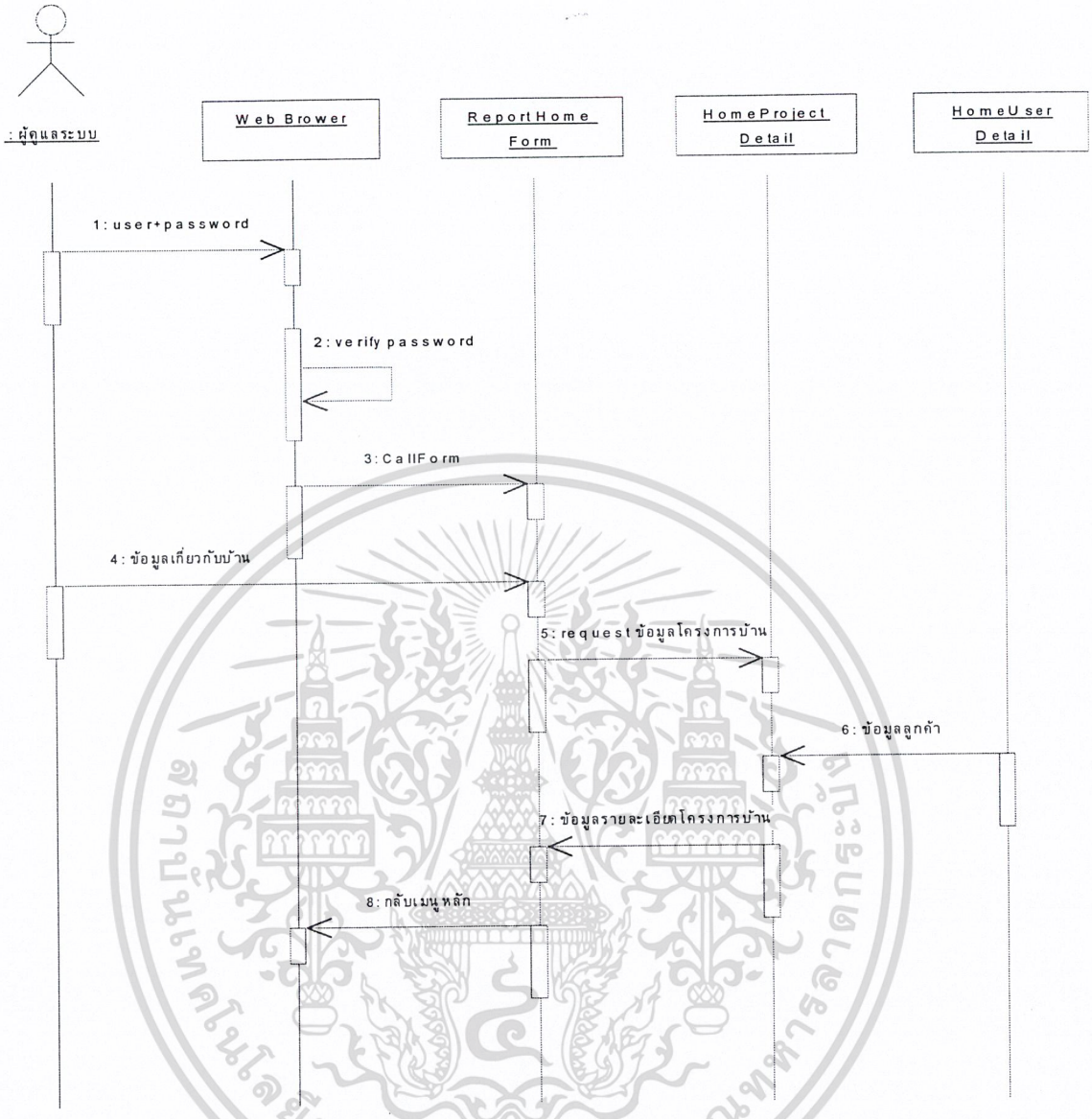
รูปที่ 4-8 Sequence Diagram รายงานนัดหมายทำสัญญา

4.7.6 Sequence Diagram เพิ่มโครงการบ้าน



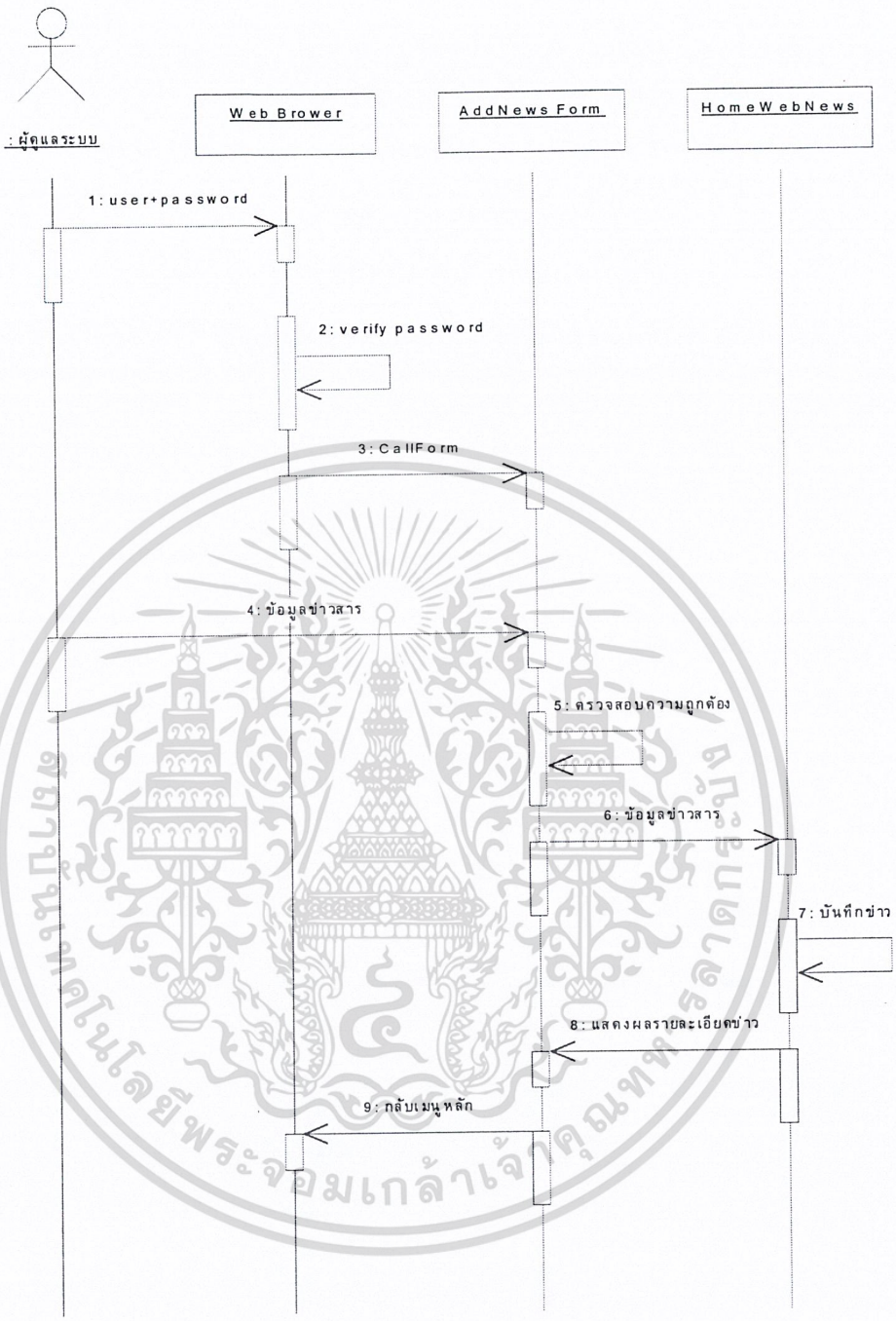
รูปที่ 4-9 Sequence Diagram เพิ่มโครงการบ้าน

4.7.7 Sequence Diagram รายงานโครงการบ้าน



รูปที่ 4-10 Sequence Diagram รายงานโครงการบ้าน

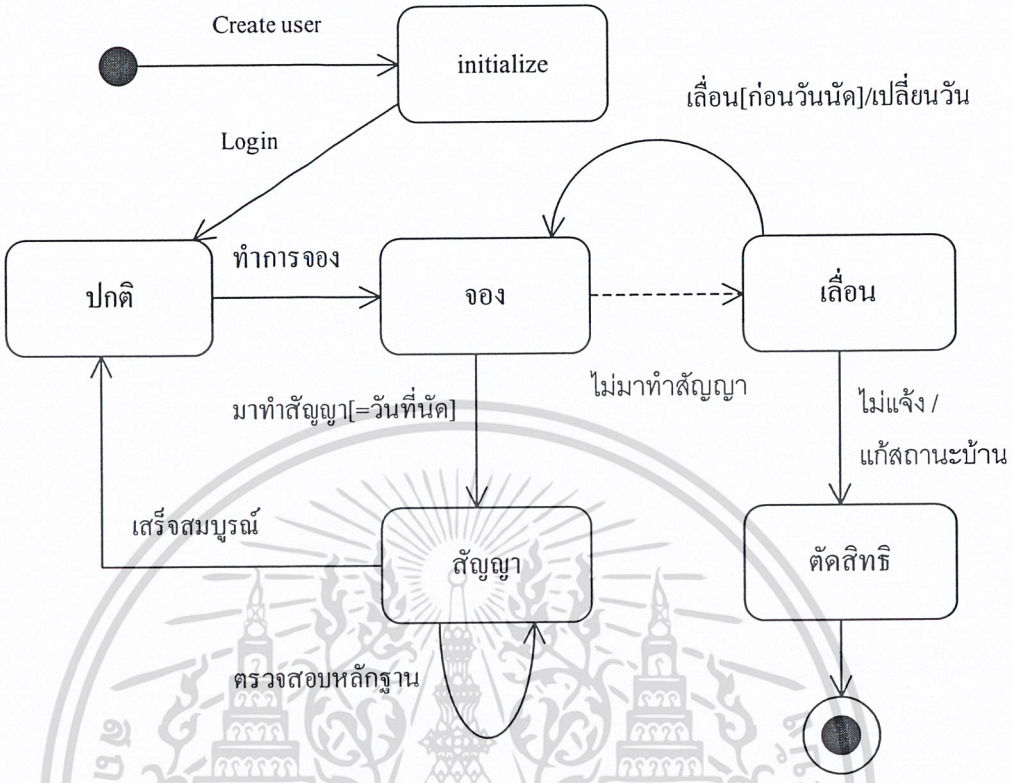
4.7.8 Sequence Diagram เพิ่มข่าวสาร



รูปที่ 4-11 Sequence Diagram เพิ่มข่าวสาร

4.8 State Diagram

State Diagram ของคลาส HomeUserDetail แสดงดังต่อไปนี้



รูปที่ 4-12 State Diagram ของคลาส HomeUserDetail

4.9 การสร้างตารางฐานข้อมูลจาก Class Diagram

จาก Class Diagram ที่ได้นำออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์มาสร้างเป็นตารางตามวิธีที่ได้กล่าวบทที่ 3

ตาราง HomeContractDetail

คอลัมน์	ชนิดข้อมูล	ข้อบังคับ
CONTRACT_ID	NUMBER(10)	Primary Key
PROJECT_ID	NUMBER(10)	Foreign Key อ้างอิง HomeDetail
HOME_ID	NUMBER(10)	Foreign Key อ้างอิง HomeDetail
USERNAME	VARCHAR2(10)	Foreign Key อ้างอิง HomeUserDetail
YEAR	NUMBER(10)	Check [5,10,15,20]
PAY_MONTH	NUMBER(10)	Not Null
CURRENT_RATE	FLOAT(2)	Check [7,7.5,8,8.5,9]
START	DATE	Not Null

ตาราง 4-2 ตารางฐานข้อมูลคลาส HomeContractDetail

ตาราง HomeUserDetail

คอลัมน์	ชนิดข้อมูล	ข้อบังคับ
USERNAME	VARCHAR2(10)	Primary Key
PASSWORD	VARCHAR2(10)	Not Null
TYPE	VARCHAR2(10)	Check [user, dba, supply]
POST_NAME	VARCHAR2(10)	Not Null
FIRST_NAME	VARCHAR2(30)	Not Null
LAST_NAME	VARCHAR2(30)	Not Null
BIRTHDAY	DATE	Not Null
ADDRESS	VARCHAR2(50)	Not Null
TELEPHONE	VARCHAR2(10)	Not Null
PID	VARCHAR2(10)	Not Null
EMAIL	VARCHAR2(20)	Not Null
SALARY	NUMBER(10)	Not Null
STATUS	VARCHAR2(10)	Check [enable , disable]

ตาราง 4-3 ตารางฐานข้อมูลคลาส HomeUserDetail

ตาราง HomeProjectDetail

คอลัมน์	ชนิดข้อมูล	ข้อบังคับ
PROJECT_ID	NUMBER(10)	Primary Key
PROJECT_NAME	VARCHAR2(30)	Unique, Not Null
OWNER	VARCHAR2(50)	Not Null
OWNER_TELEPHONE	VARCHAR2(10)	Not Null
OWNER_ADDRESS	VARCHAR2(50)	Not Null
AMOUNT	VARCHAR2(10)	Not Null
RATE	FLOAT(2)	Check [7,7.5,8,8.5,9]
ADDRESS	VARCHAR2(50)	Not Null
CONTACT	VARCHAR2(30)	Not Null

ตาราง 4-4 ตารางฐานข้อมูลคลาส HomeProjectDetail

ตาราง HomeDetail

คอลัมน์	ชนิดข้อมูล	ข้อบังคับ
HOME_ID	NUMBER(10)	Primary Key (concatenated key)
PROJECT_ID	NUMBER(10)	Primary Key (concatenated key),Foreign Key อ้างอิง HomeProjectDetail
LAND_ID	VARCHAR2(10)	Unique, Not Null
HOME_NAME	VARCHAR2(50)	Not Null
TYPE	VARCHAR2(10)	Not Null
AREA	VARCHAR2(10)	Not Null
PRICE	NUMBER(10)	Check [<=100,000,000]
STATUS	VARCHAR2(10)	Check [empty ,reserve ,sell]
STAGE	VARCHAR2(10)	Not Null
BEDROOM	VARCHAR2(10)	Not Null
BATHROOM	VARCHAR2(10)	Not Null
DETAIL	VARCHAR2(100)	Not Null
CONTACT	VARCHAR2(30)	Not Null
LOCATION	VARCHAR2(50)	Not Null
NEED	VARCHAR2(10)	Not Null

ตาราง 4-5 ตารางฐานข้อมูลคลาส HomeDetail

ตาราง HomeWebNews

คอลัมน์	ชนิดข้อมูล	ข้อบังคับ
ID	NUMBER(10)	Primary Key
HEADNEWS	VARCHAR2(500)	Not Null
DATE_POST	VARCHAR2(50)	Not Null
DETAIL	VARCHAR2(4000)	Not Null
AUTHOR	VARCHAR2(100)	Not Null
DATE_TEMP	DATE	Not Null
COUNT	ROWID	Not Null

ตาราง 4-6 ตารางฐานข้อมูลคลาส HomeWebNews

บทที่ 5
ทฤษฎีและความรู้พื้นฐานเกี่ยวกับระบบฐานข้อมูล

5.1 ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database Management System : ORDBMS) เป็นระบบจัดการฐานข้อมูลที่มีความสามารถของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System : RDBMS) โดยในโครงงานนี้ได้เลือกใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ของบริษัทออราเคิลที่มีชื่อว่า ออราเคิล8i (Oracle8i) เวอร์ชัน8.1.7ในโครงงานนี้ได้เลือกใช้เครื่องมือที่ DBA Studio และความสามารถของระบบจัดการฐานข้อมูลแบบออบเจ็กต์ (Object-Oriented Database Management System) ไว้ด้วยกันซึ่งแยกอธิบายได้ดังนี้

5.1.1 ความสามารถของระบบจัดการฐานข้อมูลเชิงสัมพันธ์

มีความสามารถในการเก็บและจัดการ ข้อมูลที่อยู่ในลักษณะความสัมพันธ์แบบรีเลชันซึ่งตัวอย่างหนึ่งของความสัมพันธ์แบบรีเลชันนี้คือ ตาราง โดยมีความสัมพันธ์กันระหว่างแถว (Row) และสดมภ์ (Column) ตัวอย่างของระบบจัดการฐานข้อมูลแบบนี้คือ ระบบจัดการฐานข้อมูลที่ใช้ภาษา SQL เป็นหลัก (SQL-Base Database Management System) ซึ่งมีคุณสมบัติดังนี้

1. โครงสร้างข้อมูล (Data Structure หรือ Logical data structure) เป็นในรูปแบบของความสัมพันธ์โดยแสดงออกมาในรูปแบบของตารางซึ่งข้อมูลในตารางจะต้องมีคุณสมบัติดังนี้
   - ข้อมูลในแต่ละแถวไม่ซ้ำกัน
   - ไม่จำเป็นต้องมีการเรียงลำดับของแถวต่างๆ
   - ข้อมูลในแต่ละสดมภ์จะไม่สามารถแบ่งแยกอีกได้
2. กฎบังคับความถูกต้องของข้อมูล (Data Integrity) เป็นกฎบังคับว่าข้อมูลที่ทำการใส่ลงฐานข้อมูลนั้นต้องมีความถูกต้องตามกฎข้อบังคับ (Constraint) ที่ตั้งไว้
3. ภาษาที่ใช้ในการจัดการข้อมูล (Data Manipulation)

5.1.2 ความสามารถของระบบจัดการฐานข้อมูลแบบออบเจ็กต์

มีการเก็บข้อมูล (Data หรือ Property) และการใช้ข้อมูล (Procedure หรือ Method) เข้าด้วยกันอยู่ในลักษณะของออบเจ็กต์ ซึ่งมีคุณสมบัติทางออบเจ็กต์ดังนี้

1. ความสามารถในการซ่อนรายละเอียด (Encapsulation) เป็นการซ่อนการทำงานภายในของออบเจ็กต์ไว้ ซึ่งออบเจ็กต์อื่นจะรู้แค่อินเตอร์เฟส (Interface) เท่านั้น
2. คุณสมบัติการโพลิมอร์ฟิซึม (Polymorphism)
3. ความสัมพันธ์แบบออบเจ็กต์ คือ
   - ความสัมพันธ์แบบแอคกรีเกชั่น (Aggregation)
   - ความสัมพันธ์แบบสืบทอด (Generalization)

ในระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ที่ทำการศึกษาและนำมาใช้งาน คือระบบจัดการฐานข้อมูลออราเคิล ซึ่งมีความสามารถดังนี้

1. สามารถใช้ชนิดข้อมูลมาตรฐาน (Built-In Datatypes)
2. สามารถใช้ชนิดข้อมูลแบบกำหนดเอง (User-Defined Datatypes) ซึ่งประกอบด้วย
   2.1 ชนิดข้อมูลแบบออบเจ็กต์ (Object Types)
   2.2 ชนิดข้อมูลแบบกลุ่ม (Collection Types)

และทั้งสองหัวข้อนี้จะถูกนำเสนอถึงรายละเอียดทั้งหมดภายในบทนี้

5.2 ชนิดข้อมูลมาตรฐาน

ใช้ในการระบุชนิดของข้อมูลที่ต้องการเก็บลงฐานข้อมูล ซึ่งมีชื่อและลักษณะที่แตกต่างกันในระบบจัดการฐานข้อมูลที่มีอยู่มากมายในปัจจุบัน โดยชนิดข้อมูลมาตรฐานที่จะนำเสนอนี้จะเป็นชนิดข้อมูลมาตรฐานของระบบจัดการฐานข้อมูลออราเคิล

5.2.1 ชนิดข้อมูลแบบตัวอักษร (Character Datatypes)

ใช้สำหรับการเก็บข้อมูลที่เป็นตัวอักษร ซึ่งอยู่ในรูปแบบของการเข้ารหัสตัวอักษร (Character set) ในลักษณะต่างๆ เช่น ASCII, EBCDIC, etc. ซึ่งมีการแบ่งประเภทของชนิดข้อมูลแบบตัวอักษรนี้ย่อยลงไปอีกได้ดังนี้

5.2.1.1 CHAR Datatype

ใช้เก็บข้อความที่มีกำหนดความยาวของข้อความไว้แน่นอน ซึ่งมีความยาวข้อมลอยู่ระหว่าง 1 ถึง 200 ไบต์ ซึ่งเมื่อทำการเพิ่มหรือเปลี่ยนแปลงข้อมูลที่มีความยาวน้อยกว่าที่กำหนดข้อมูลชุดนั้นจะถูกแทนที่ด้วยช่องว่างซึ่งเป็นรหัสแอสกีตัวหนึ่ง

5.2.1.2 VARCHAR2 Datatype

ใช้เก็บข้อความที่มีจำนวนของตัวอักษรในข้อความนั้นเปลี่ยนแปลงได้ สามารถกำหนดจำนวนของตัวอักษรมากที่สุดที่อยู่ในหนึ่งข้อความได้ โดยมีค่าได้ระหว่าง 1 ถึง 4000

5.2.1.3 VARCHAR Datatype

เป็นชนิดข้อมูลที่ใช้ในออราเคิลรุ่นเก่าซึ่งยังคงไว้เพื่อสามารถ ใช้เก็บข้อมูลที่มีอยู่ในฐานข้อมูลออราเคิลรุ่นเก่นได้ โดยมีคุณสมบัติคล้ายกับ Varchar2

5.2.1.4 NCHAR และ NVARCHAR2 Datatype

ใช้เก็บข้อมูลที่มีการเข้ารหัสแบบ NLS โดย NCHAR จะมีคุณสมบัติเหมือน CHAR และ NVARCHAR2 จะมีคุณสมบัติเหมือน VARCHAR2

5.2.1.5 LONG Datatype

ใช้เก็บข้อความที่มีจำนวนของตัวอักษรในข้อความนั้นเปลี่ยนแปลงได้ และสามารถเก็บจำนวนตัวอักษรสูงสุดได้มากกว่า VARCHAR2

5.2.2 ชนิดข้อมูลแบบตัวเลข (Number Datatype)

ใช้เก็บข้อมูลที่เป็นตัวเลข ซึ่งประกอบด้วยเลขจำนวนเต็ม และเลขจำนวนจริง ซึ่งมีขนาดข้อมูลสูงสุด 21 ไบต์

5.2.3 ชนิดข้อมูลที่เป็นวันที่ (Date Datatype)

ใช้เก็บข้อมูลประเภทเวลา ประกอบด้วยปี, เดือน, วัน, ชั่วโมง, นาที, วินาที ซึ่งมีขนาดคงที่ 7 ไบต์ สำหรับรูปแบบในการเก็บนั้นสามารถกำหนดได้ ซึ่งจะนำเสนอในส่วนของการเปลี่ยนชนิดข้อมูล

5.2.4 ชนิดข้อมูลที่มีขนาดใหญ่ (LOB Datatypes)

ใช้เก็บข้อมูลขนาดใหญ่ที่ไม่มีโครงสร้างข้อมูลที่แน่นอน เช่น ข้อมูลรูปภาพ ข้อมูลมัลติมีเดีย ซึ่งการเก็บข้อมูลด้วยชนิดของการเก็บข้อมูลนี้ทำให้สามารถเข้าถึงข้อมูลแบบสุ่มได้ (Random access) ในขณะที่การเก็บข้อมูลชนิด LONG จะเข้าถึงข้อมูลแบบเป็นลำดับเท่านั้น (Sequential access) ชนิดข้อมูลประเภทนี้ประกอบด้วย

5.2.4.1 BLOB Datatype

ใช้เก็บข้อมูลไบนารีขนาดใหญ่ที่ไม่มีโครงสร้าง (Unstructured Binary Data)

5.2.4.2 CLOB และ NCLOB Datatype

ใช้เก็บข้อมูลข้อความที่มีจำนวนตัวอักษรมากๆ

5.2.4.3 BFILE Datatype

ใช้เก็บข้อมูลไบนารีขนาดใหญ่โดยทำการเก็บข้อมูลนั้นไว้นอกฐานข้อมูล และเก็บเพียงตัวชี้ข้อมูลนั้นไว้ (Pointer) ซึ่งจะเป็นการเก็บข้อมูลเพื่ออ่านอย่างเดียว

5.2.5 RAW และ LONG RAW Datatypes

ใช้เก็บข้อมูลที่มีขนาดใหญ่โดย RAW มีคุณสมบัติเหมือน VARCHAR2 ส่วน LONG RAW นั้นเหมือน LONG สามารถใช้เก็บข้อมูลที่เป็นมัลติมีเดียได้แต่ประสิทธิภาพจะไม่ดีเท่าใช้ LOB และไม่สามารถทำการเอคซ์พอร์ท (Export)

5.2.6 ROWID Datatype

ใช้เก็บข้อมูลที่เป็นตำแหน่งจริงของแถวในฐานข้อมูล

5.3 ชนิดข้อมูลแบบกำหนดเอง

เกิดจากการสร้างชนิดข้อมูลขึ้นมาจากชนิดข้อมูลมาตรฐาน และชนิดข้อมูลกำหนดเองอื่นๆ ซึ่งได้ถูกแบ่งออกมาสองลักษณะดังนี้

5.3.1 ชนิดข้อมูลแบบออบเจ็กต์

เป็นชนิดของข้อมูลที่ผู้ใช้สามารถทำการสร้างขึ้นมาเอง โดยทำการกำหนดลักษณะของข้อมูลให้มีความใกล้เคียงกับความเป็นจริงมากที่สุด ซึ่งเป็นการใช้ลักษณะเด่นของข้อมูลนั้นเป็นตัวระบุ ซึ่งชนิดข้อมูลแบบออบเจ็กต์นี้ประกอบด้วยส่วนประกอบ 3 ส่วนดังนี้

1. ชื่อ (Name) ใช้ในการระบุถึงชนิดข้อมูลแบบออบเจ็กต์ ซึ่งต้องไม่ซ้ำกันในโครงสร้างที่เก็บข้อมูลแบบออบเจ็กต์เดียวกัน (Schema)
2. ลักษณะต่างๆ (Attributes) คือข้อมูลที่ออบเจ็กต์ควรมีเพื่อทำให้ออบเจ็กต์นั้นมีความใกล้เคียงกับความเป็นจริงให้มากที่สุด ซึ่งจะเป็นชนิดข้อมูลแบบมาตรฐาน หรือชนิดข้อมูลแบบกำหนดเองก็ได้
3. การกระทำ (Method) เป็นฟังก์ชันหรือโพรซีเดอร์ ที่สร้างจากภาษา PL/SQL เพื่อสร้างการกระทำให้กับออบเจ็กต์

ตัวอย่าง การสร้างข้อมูลชนิดออบเจ็กต์

```
CREATE OR REPLACE TYPE person_object AS OBJECT (
ssn             NUMBER,
last_name       VARCHAR2(50),
first_name      VARCHAR2(50) ;
```

5.3.2 ชนิดข้อมูลแบบกลุ่ม

เป็นลักษณะของข้อมูลที่มีชนิดของข้อมูลเหมือนกันและมีจำนวนที่ไม่สามารถระบุได้ว่ามีเท่าไร ซึ่งจะแบ่งเป็น ชนิดข้อมูลแบบอาร์เรย์ (Array Types) หรือชนิดข้อมูลแบบ วีอาร์เรย์ (VArray) และ ชนิดข้อมูลแบบ ตาราง (Table Types) หรือชนิดข้อมูลแบบตารางซ้อนตาราง (Nested Table)

- ชนิดข้อมูลแบบวีอาร์เรย์

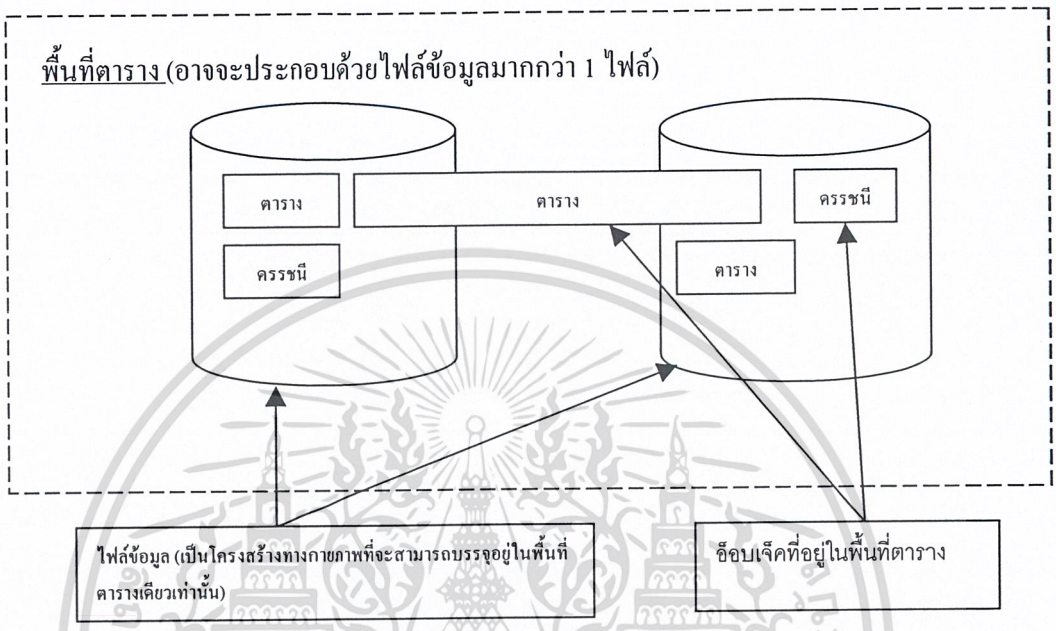
เป็นการเก็บข้อมูลที่มีชนิดของข้อมูลเหมือนกัน และข้อมูลที่อยู่ในแต่ละช่อง ของอาร์เรย์จะถูกระบุด้วยดัชนี (Index) ซึ่งเป็นลักษณะของเซตของข้อมูลแบบมีลำดับ

ตัวอย่าง การสร้างชนิดข้อมูลวีอาร์เรย์

```
CREATE TYPE Project AS OBJECT(
  project_no NUMBER(2),
  title      VARCHAR2(35),
  cost       NUMBER(7,2))
CREATE TYPE ProjectList AS VARRAY(50) OF Project
```

5.4 ฐานข้อมูล(Database) พื้นที่ตาราง(Tablespaces)และไฟล์ข้อมูล(Datafiles)

ข้อมูลของออราเคิลที่ทำการเก็บลงในฐานข้อมูลนั้นสามารถพิจารณาได้ 2 มุมมองคือในทางด้านความคิดหรือทางลอจิกนั้นจะเก็บข้อมูลลงในพื้นที่ตาราง (Tablespaces) และในมุมมองทางด้านกายภาพนั้นจะทำการเก็บข้อมูลลงในไฟล์ข้อมูลที่กำหนดในพื้นที่ตารางดังแสดงในรูปที่ 5-1



รูปที่ 5-1 พื้นที่ตารางและไฟล์ข้อมูล

แม้ว่าฐานข้อมูล พื้นที่ตารางและไฟล์ข้อมูลจะมีความสัมพันธ์กันแต่ก็มีความแตกต่างกันอยู่บ้าง คือ ในฐานข้อมูลของออราเคิลจะมีหน่วยทางตรรกที่จัดเก็บข้อมูลทั้งหมดคือพื้นที่ตารางและในพื้นที่ตารางนั้นจะมีไฟล์ที่ทำการเก็บข้อมูลในทางกายภาพเรียกว่าไฟล์ข้อมูล

เมื่อเราสามารถสร้างพื้นที่ตารางแล้วเราสามารถสร้างตารางขึ้นมาใช้งานได้ โดยไม่เกี่ยวข้องกับพื้นที่ตารางระบบของทางออราเคิลที่ได้สร้างขึ้นมาเมื่อเราได้ติดตั้งระบบ

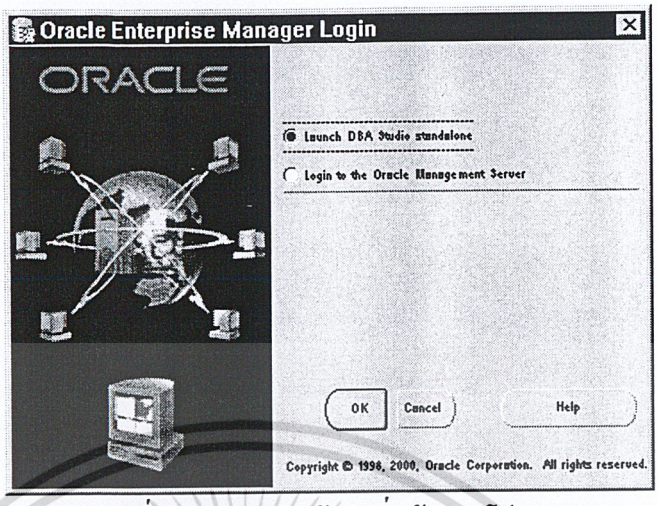
5.5 การใช้งาน DBA Studio

โปรแกรม DBA Studio เป็นโปรแกรมที่ช่วยให้ผู้ดูแลระบบสามารถที่บริหารระบบฐานข้อมูลของออราเคิลได้สะดวกและง่ายเนื่องจากมีการติดต่อกับผู้ใช้ด้วยรูปภาพ โดยลักษณะของไอคอนดังแสดงในรูปที่ 5-2

รูปที่ 5-2 แสดงรูปไอคอนของโปรแกรม DBA Studio

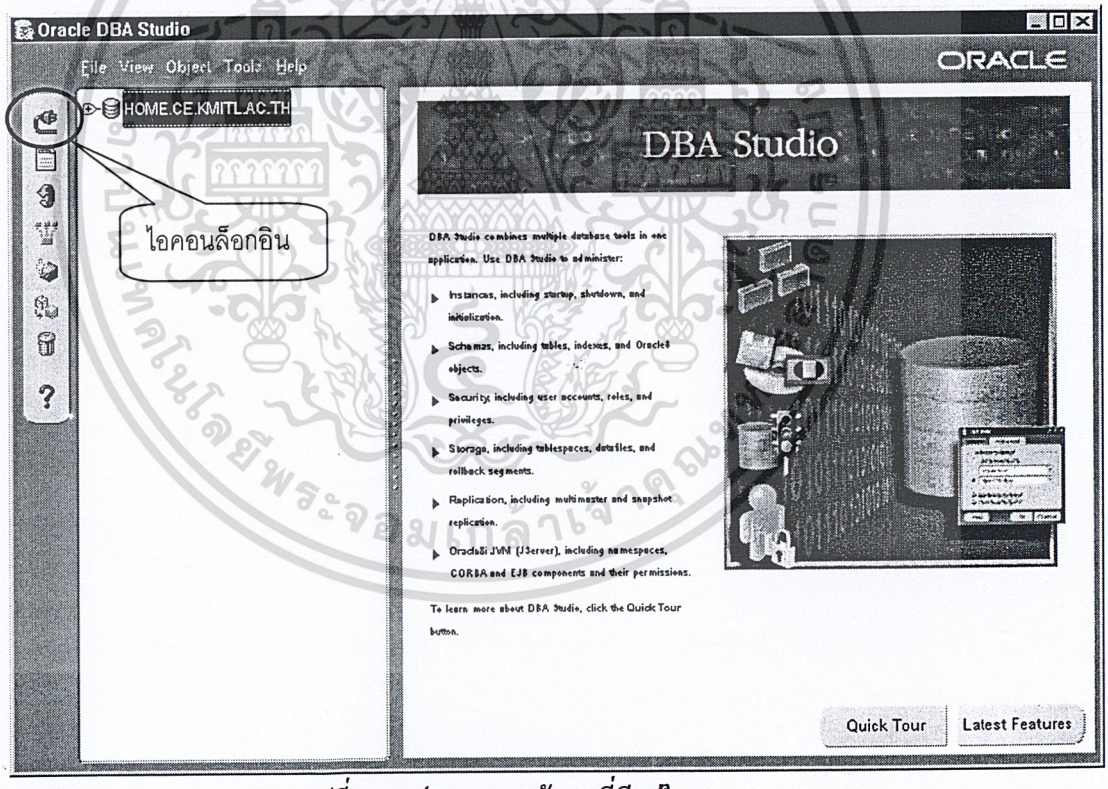
5.5.1 เชื่อมต่อฐานข้อมูลด้วย DBA Studio

เมื่อทำการเรียกใช้โปรแกรมเพื่อทำการบริหารระบบฐานข้อมูลดังแสดงดังรูปที่ 5-3



รูปที่ 5-3 รูปแสดงหน้าจอเริ่มต้นของโปรแกรม

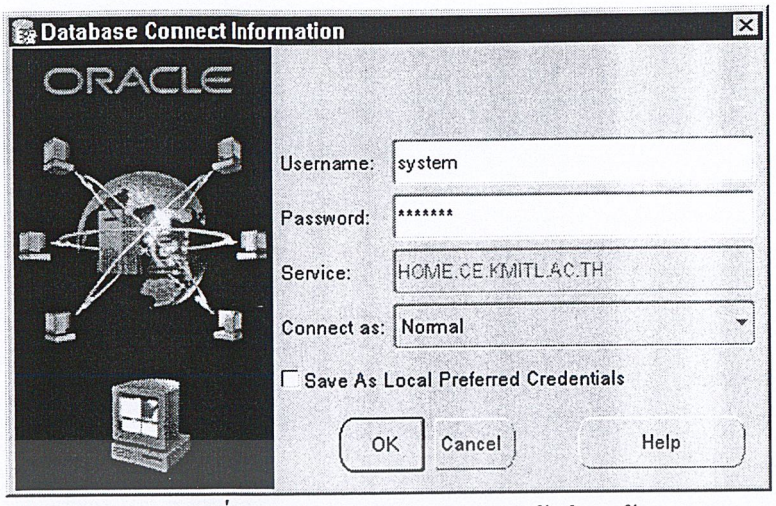
และเมื่อทำการคลิ้ก OK ก็จะเข้าสู่หน้าจอหลักของโปรแกรมที่แสดงฐานข้อมูลที่มีในระบบดังแสดงในรูปที่ 5-4



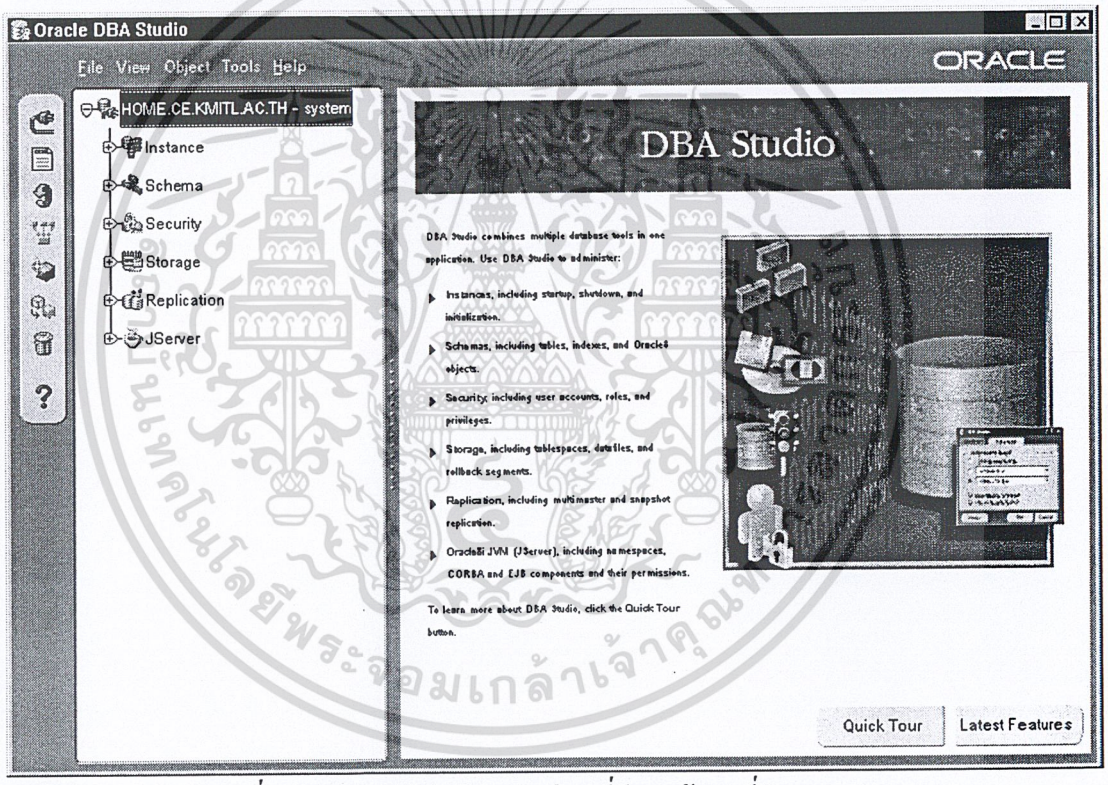
รูปที่ 5-4 รูปแสดงฐานข้อมูลที่มีอยู่ในระบบ

จากนั้นให้ทำการเลือกฐานข้อมูลที่ต้องการแล้วทำการดับเบิลคลิ้กที่ฐานข้อมูลหรือที่ไอคอน Login และจะเข้าสู่หน้าจอการ Login ในรูปที่ 5-5

และเมื่อ Login เข้าสู่ฐานข้อมูลได้แล้วส่วนประกอบต่างๆของฐานข้อมูลจะแสดงในรูปที่ 5-6



รูปที่ 5-5 รูปแสดงหน้าจอ Login เข้าสู่ฐานข้อมูล

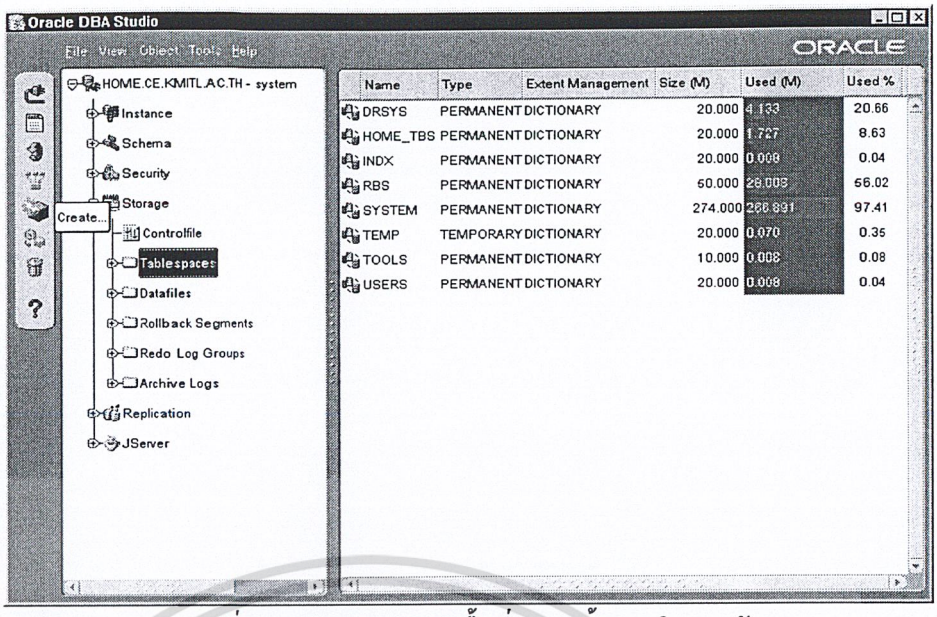


รูปที่ 5-6 รูปแสดงส่วนประกอบต่างๆที่มีฐานข้อมูลที่ทำการ Login

5.5.2 การสร้างพื้นที่ตารางและไฟล์ข้อมูล

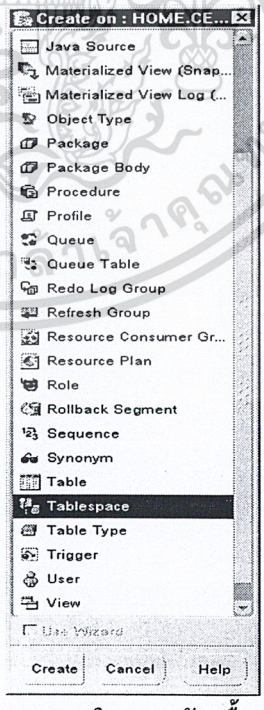
ในขั้นตอนของการพื้นที่ตารางและไฟล์ข้อมูลนั้นจะมีการสร้างมีขั้นตอนดังต่อไปนี้

1. คลิ้กที่ไอคอน “Storage” ในเฟรมทางด้านซ้ายเพื่อแสดงส่วนประกอบในการเก็บข้อมูลแล้วเลือกที่ไอคอน ”Tablespaces” ดังแสดงในรูปที่ 5-7

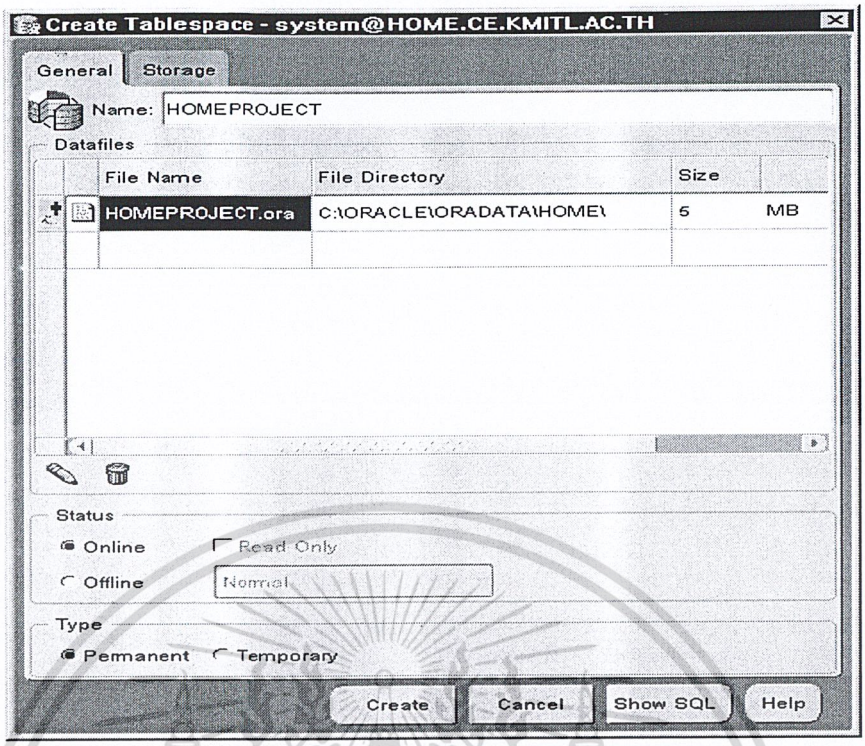


ปที่ 5-7 รูปแสดงจำนวนพื้นที่ตารางทั้งหมดในฐานข้อมูล

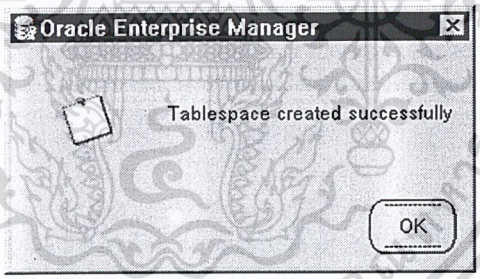
2. ในการสร้างพื้นที่คลิ้กที่ไอคอน "Create" ซึ่งอยู่เมนูในแถบทางด้านซ้ายมือสุด(กล่องสี่เหลี่ยมลูกบาศก์) ซึ่งจะทำให้ได้หน้าจอการสร้างพื้นที่ตารางในรูปที่ 5-8 เพื่อแสดงเมนูในการสร้าง "Tablespaces"
3. ในขั้นตอนต่อไปเราต้องทำการกำหนดชื่อของพื้นที่ตารางและในเราต้องกำหนดไฟล์ข้อมูลในพื้นที่ตารางด้วยซึ่งแสดงในรูปที่ 5-9
4. เมื่อการกำหนดค่าของพื้นที่ตารางถูกต้องการสร้างพื้นที่ตารางถูกต้องจะแสดงรูปในหน้าจอที่ 5-10



รูปที่ 5-8 รูปแสดงเมนูในการสร้างพื้นที่ตาราง



รูปที่ 5-9 รูปแสดงการกำหนดของพื้นที่ตารางและไฟล์ข้อมูลที่จะทำการสร้าง

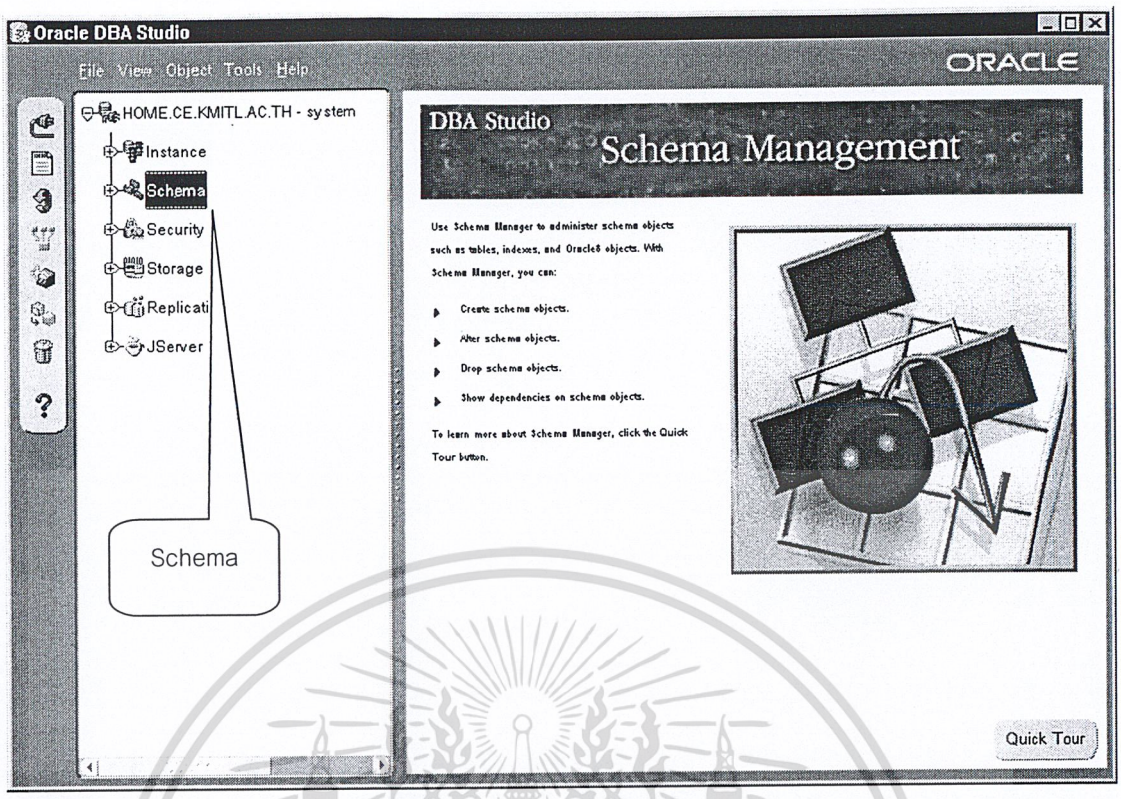


รูปที่ 5-10 รูปแสดงหน้าจอเมื่อมีการสร้างพื้นที่ตารางและไฟล์ข้อมูลถูกต้อง

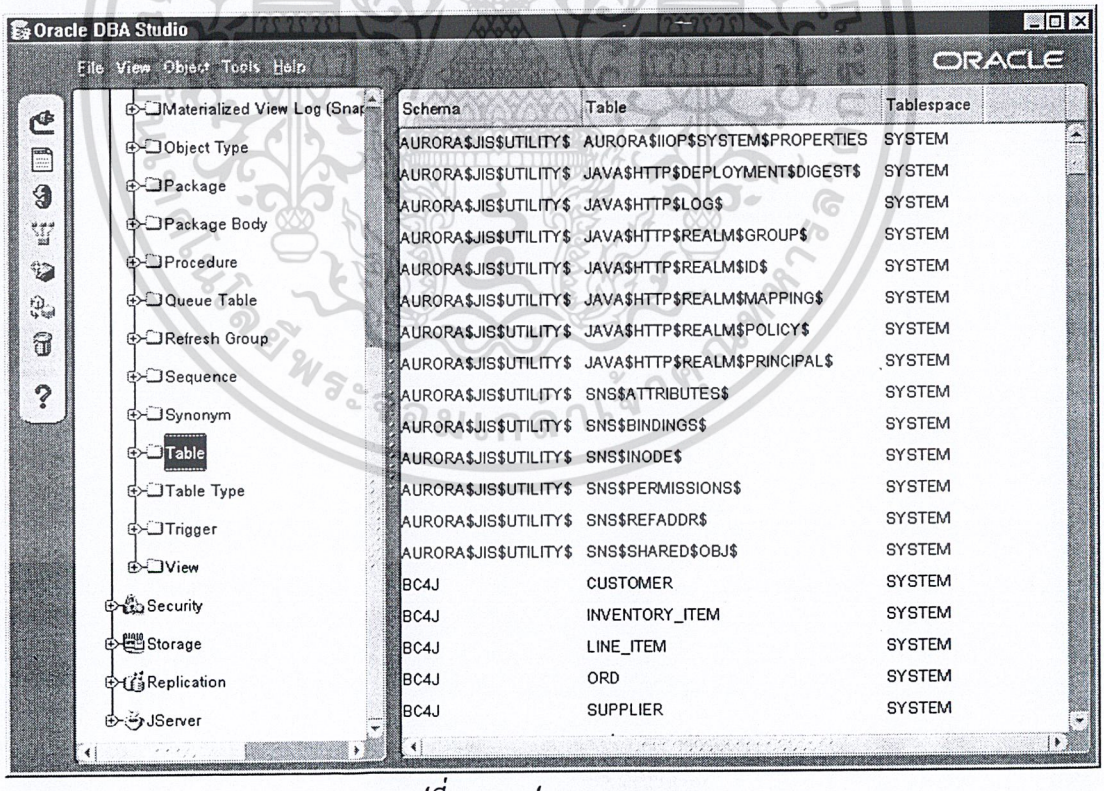
5.5.3 การสร้างตาราง

เมื่อเราได้พื้นที่ตารางและไฟล์ข้อมูลที่ไม่เกี่ยวข้องกับพื้นที่ตารางและไฟล์ข้อมูลของระบบของทางออราเคิล ขั้นตอนต่อไปจะเป็นขั้นตอนของการสร้างตารางที่ใช้เก็บข้อมูลโดยขั้นตอนของการสร้างตารางนั้นประกอบไปด้วย

1. เข้าสู่เมนู “Schemas” ในเฟรมเมนูทางด้านซ้ายดังรูปที่ 5-11 จากนั้นดับเบิลคลิ้กไอคอนจะแสดงรายการของ Schemas ทั้งหมดของระบบ แล้วคลิ้กที่ไอคอน “Table” ดังรูปที่ 5-12

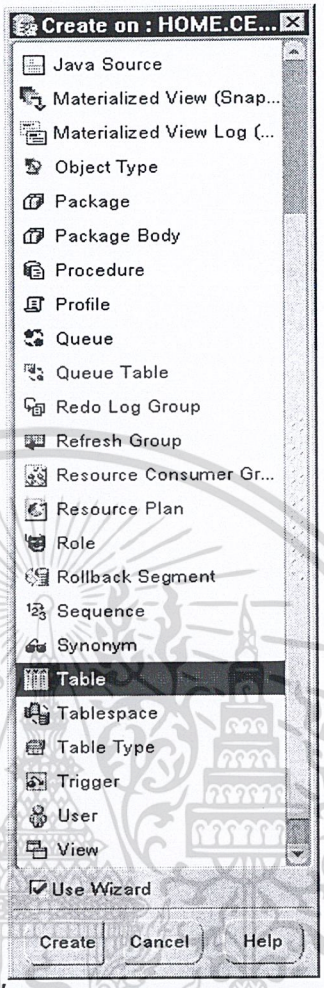


รูปที่ 5-11 รูปแสดงหน้าจอของเมนู Schema



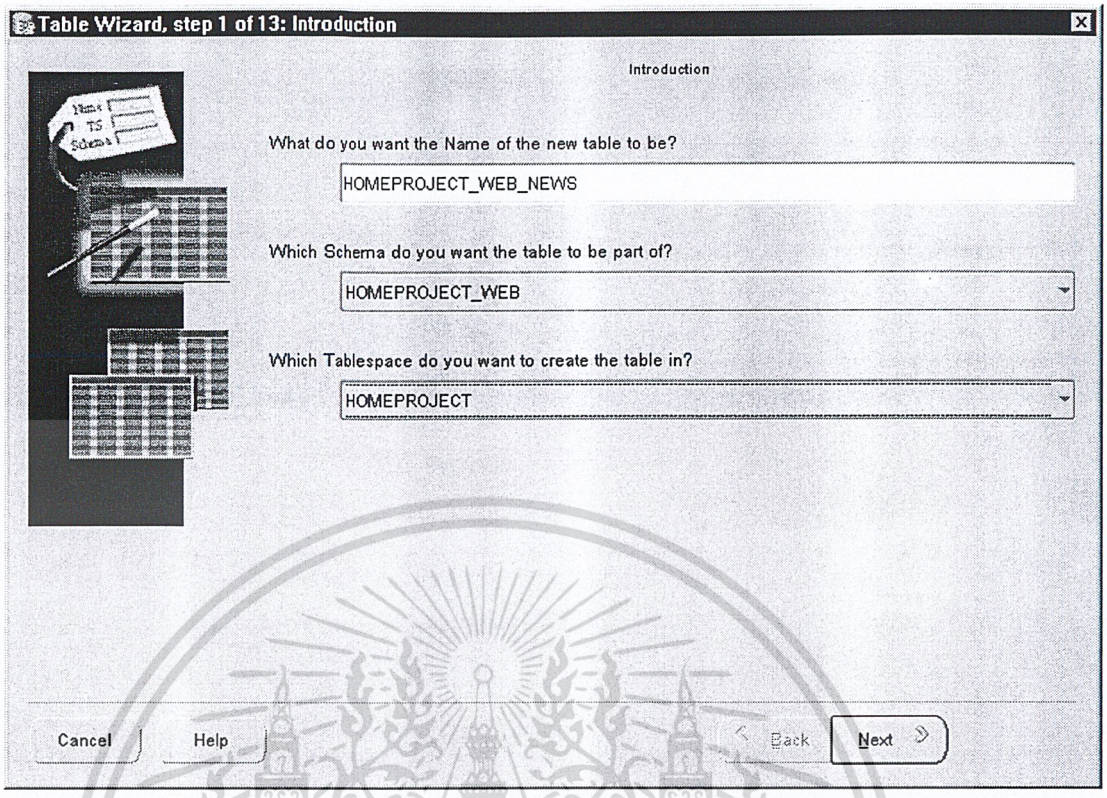
รูปที่ 5-12 รูปแสดงรายการของ Schema

2. คลิ้กที่ไอคอน "Create" ที่อยู่บนแถบเครื่องมือทางด้านซ้ายสุด(กล่องสี่เหลี่ยมลูกบาศก์) ดังแสดงในรูปที่ 5-13

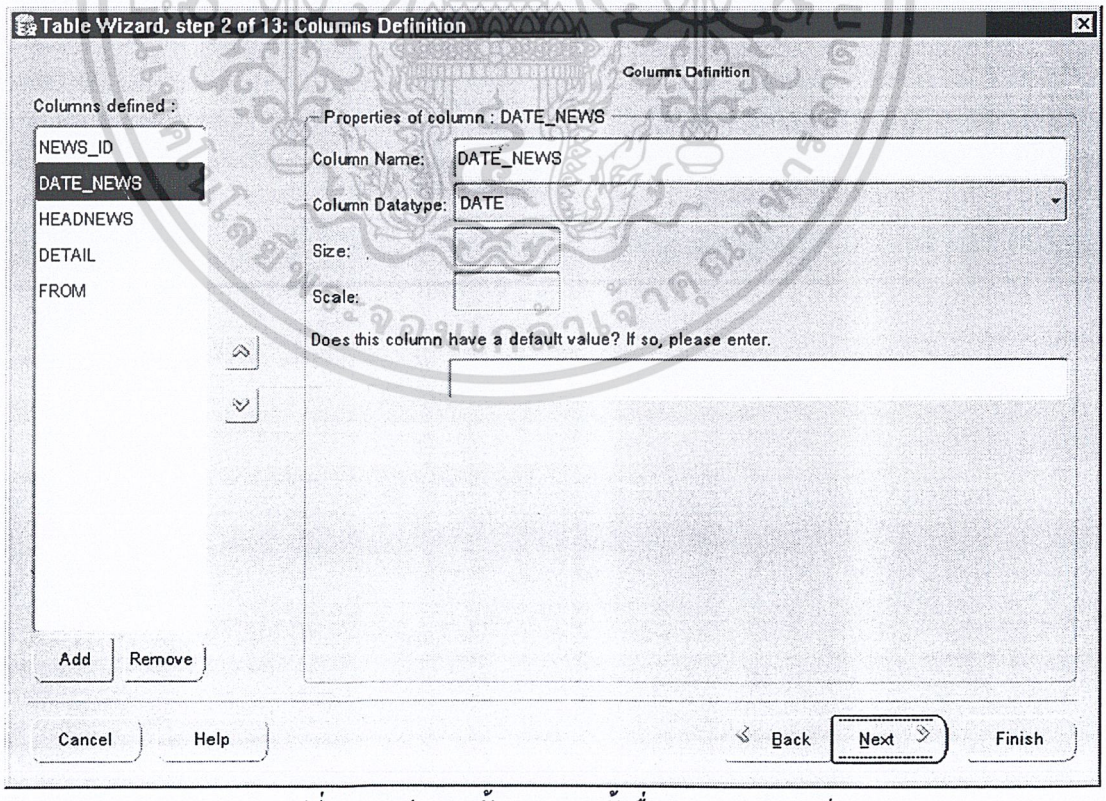


รูปที่ 5-13 รูปแสดงเมนูในการสร้างตาราง

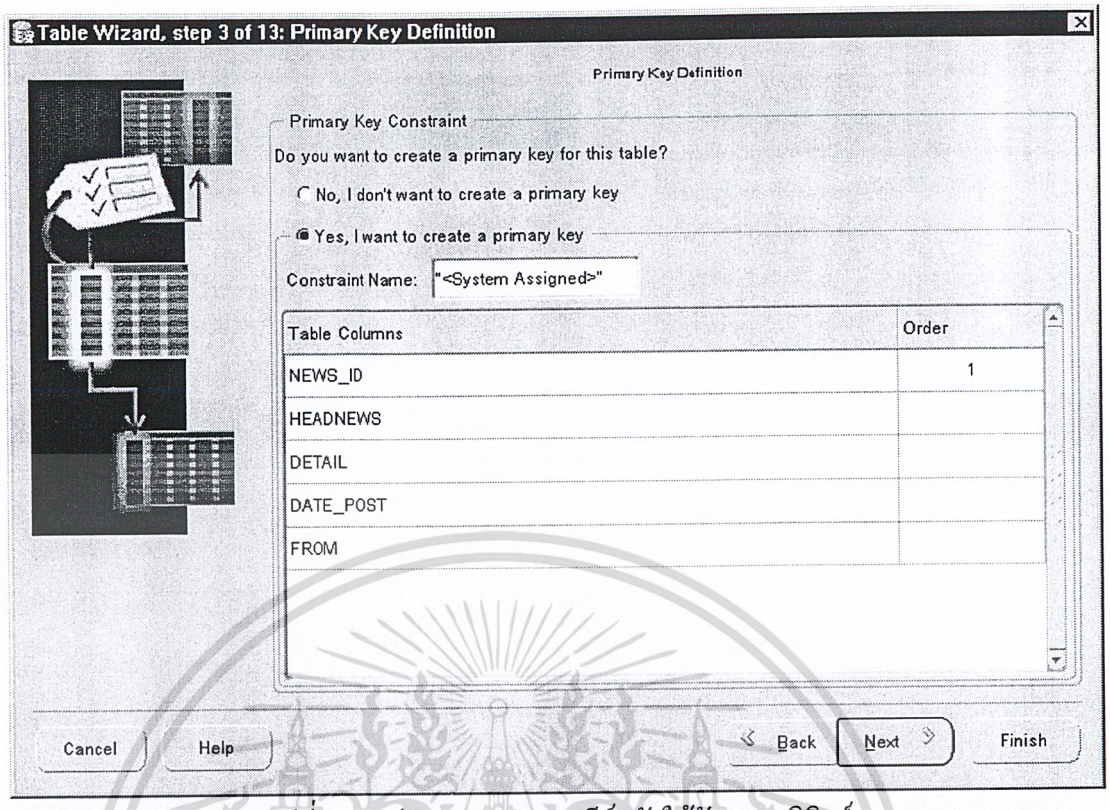
3. กำหนดชื่อของตาราง และกำหนดค่า Schema (ชื่อของผู้ใช้) และในพื้นที่ตาราง ดังแสดงในรูปที่ 5-14
4. กำหนดชื่อของของแอตทริบิวต์ และชนิดของข้อมูล ดังแสดงในรูปที่ 5-15
5. เมื่อป้อนแอตทริบิวต์ ตามต้องการและคลิ้ก "Next" เพื่อกำหนดคีย์หลักดังแสดงในรูปที่ 5-16
6. หากต้องการเพิ่มข้อมูลของแต่ละแอตทริบิวต์ ก็ให้ทำการคลิ้ก "Next" หรือไม่ต้องการก็ให้คลิ้กที่ "Finish" ดังแสดงในรูปที่ 5-17
7. จากนั้นคลิ้กที่ "Finish" อีกครั้งเพื่อยืนยันการสร้างตารางแล้วจะมีการคืนค่าผลลัพธ์ออกมาดังรูปที่ 5-18
8. เมื่อมีการสร้างตารางแล้วลองตรวจสอบรายการของตารางได้โดยการเข้าไปที่หน้าจอ Schema ที่เรากำหนดขึ้นมาจะดังแสดงในรูปที่ 5-19



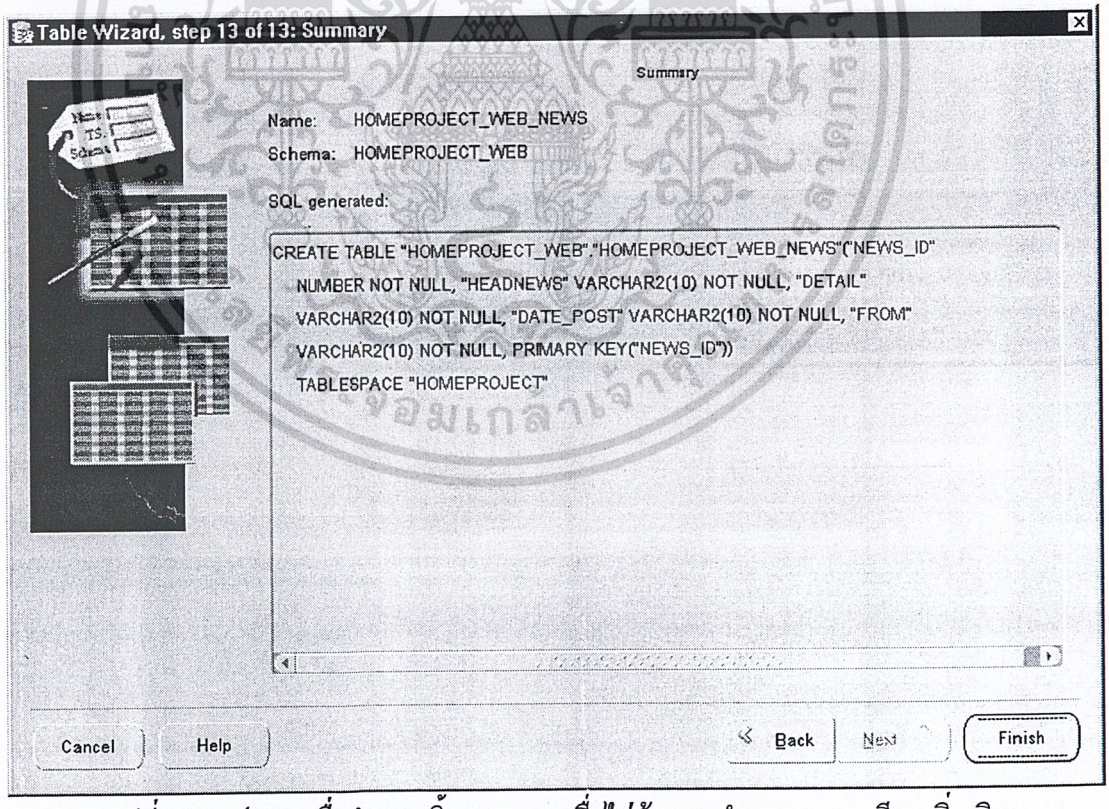
รูปที่ 5-14 รูปแสดงขั้นตอนการตั้งชื่อของตารางและการกำหนดพื้นที่ตารางและไฟล์ข้อมูล



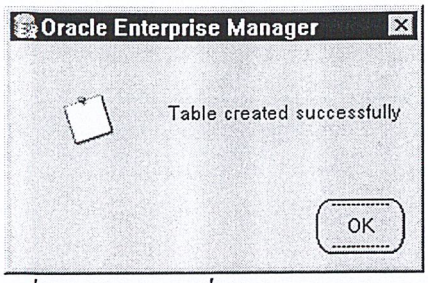
รูปที่ 5-15 รูปแสดงขั้นตอนการตั้งชื่อของแอตทริบิวต์



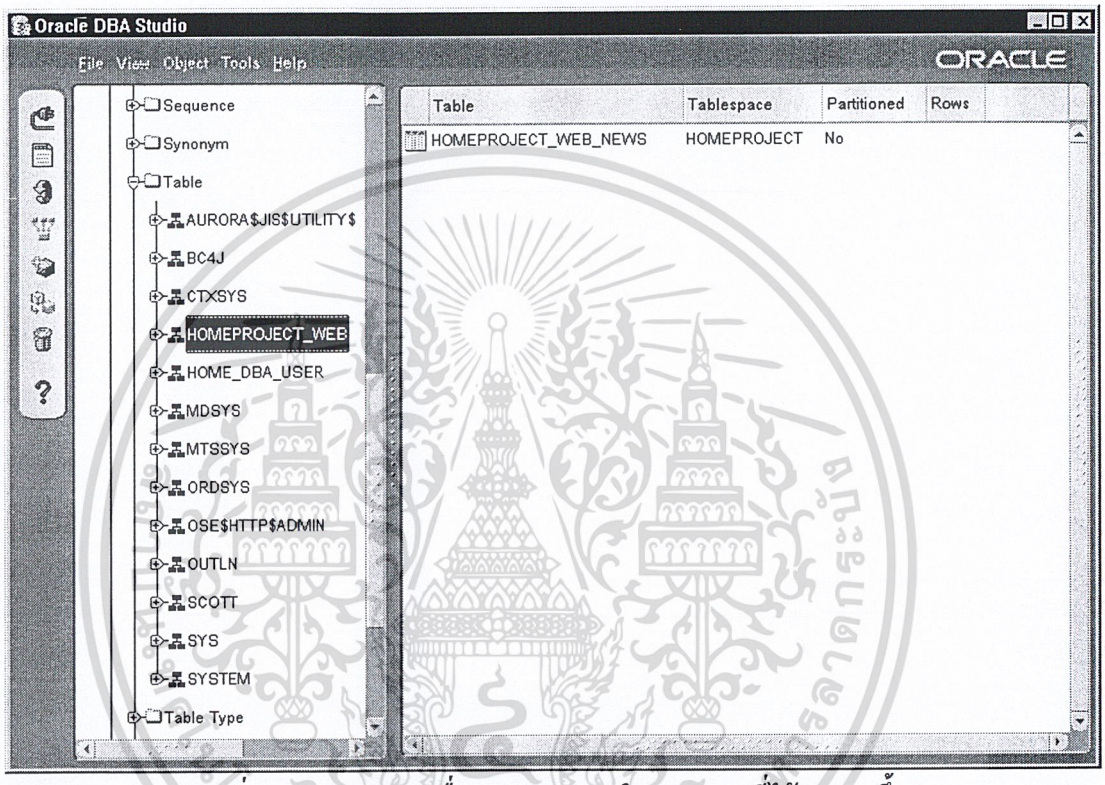
รูปที่ 5-16 รูปแสดงการกำหนดคีย์หลักให้กับแอตทริบิวต์



รูปที่ 5-17 รูปแสดงเมื่อทำการคลิ๊ก "Finish"เมื่อไม่ต้องการกำหนดรายละเอียดเพิ่มเติม



รูปที่ 5-18 รูปแสดงเมื่อยืนยันการสร้างตาราง

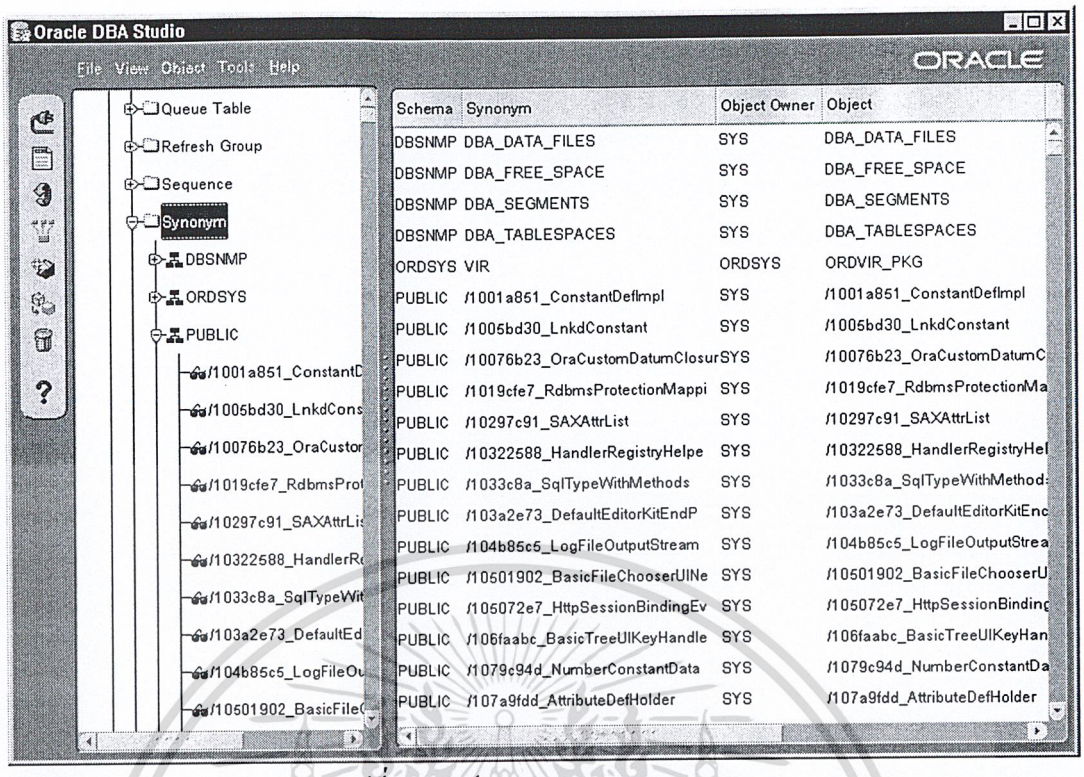


รูปที่ 5-19 รูปแสดงรายชื่อของตารางภายใน Schema ที่ได้กำหนดขึ้น

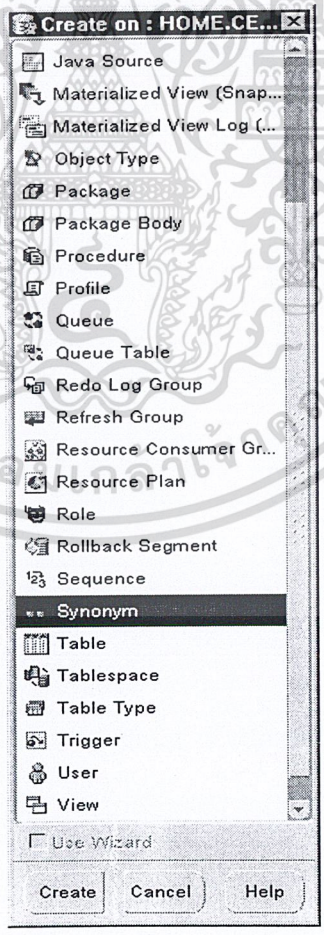
5.5.4 การสร้างชื่อเรียก(Synonym)

เมื่อเราได้สร้างตารางขึ้นมาใน Schema ใด ๆ ก็ตามในการที่ผู้ใช้คนอื่นๆจะสามารถที่จะเรียกใช้ตารางของ Schema นั้นเราจะต้องทำการกำหนดชื่อของผู้ใช้ที่สร้างก่อนทุกครั้งแต่เมื่อเราได้ทำการสร้างชื่อเรียก(Synonym) เราก็สามารถที่จะเรียกใช้ตาราง(หรืออาจใช้กับวิว)นั้นได้โดยตรง โดยขั้นตอนของการสร้างชื่อเรียกมีขั้นตอนดังต่อไปนี้

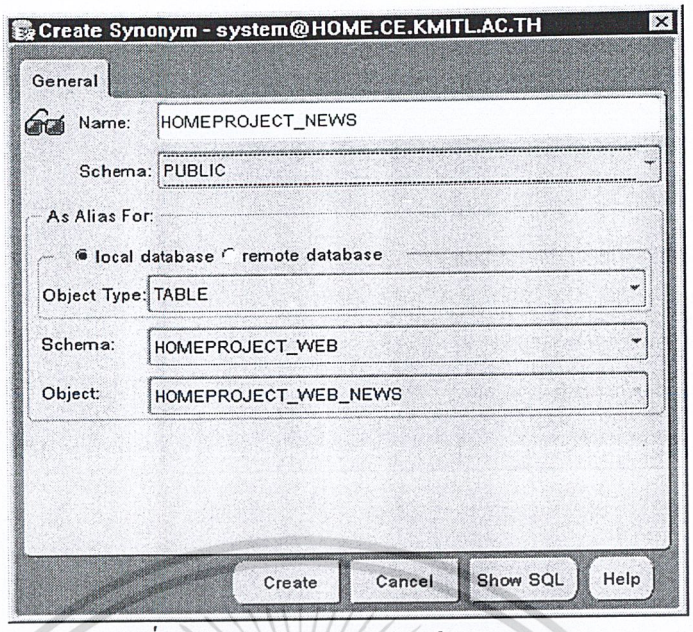
1. คลิ้กที่ “Synonym” ในส่วนของ Schema แล้วทำการคลิ้กที่ไอคอน “Create” ในแถบเครื่องมือด้านซ้ายมือสุด ดังแสดงในรูปที่ 5-20 และ 5-21
2. กำหนดขอบเขตของการเห็นชื่อเรียกที่เราสร้างขึ้นโดยกำหนดที่ Schema ให้เป็น “Public” เมื่อต้องการให้ทุกคนเห็นชื่อเรียกของเราที่ได้ทำการสร้างขึ้นมาดังรูปที่ 5-22
3. เมื่อยืนยันการสร้างชื่อเรียกจะแสดงดังรูปที่ 5-23



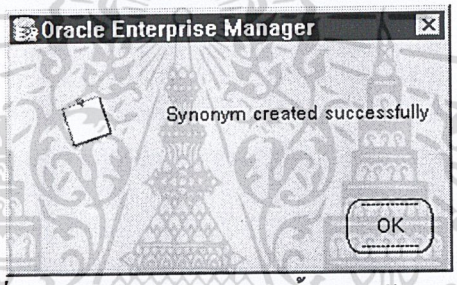
รูปที่ 5-20 รูปแสดงรายการของ Synonym



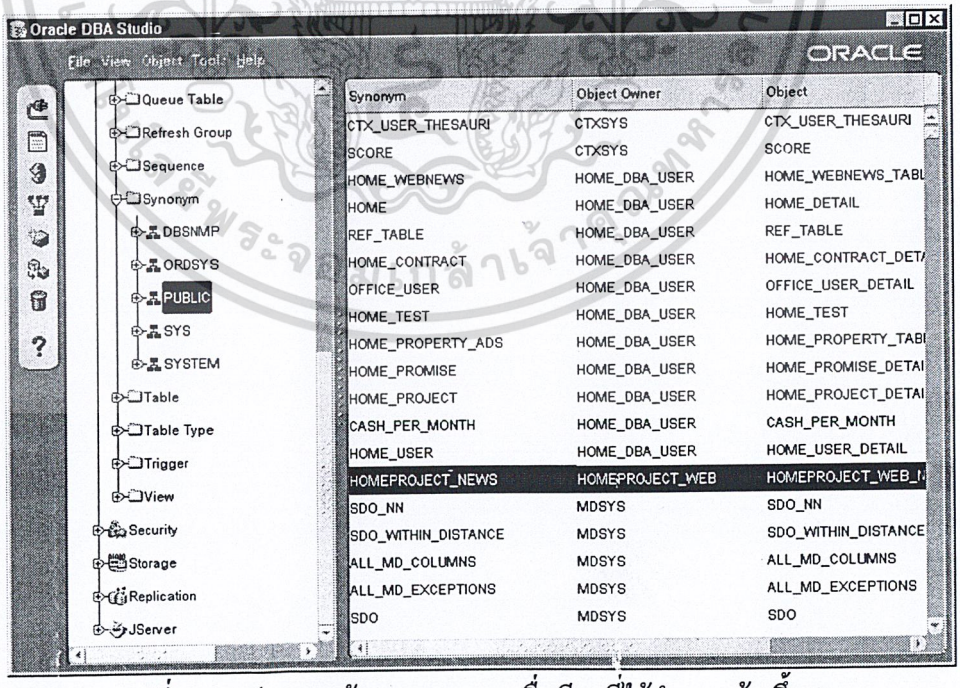
รูปที่ 5-21 รูปแสดงหน้าจอการสร้าง Synonym



รูปที่ 5-22 รูปแสดงการกำหนดค่าของ Synonym



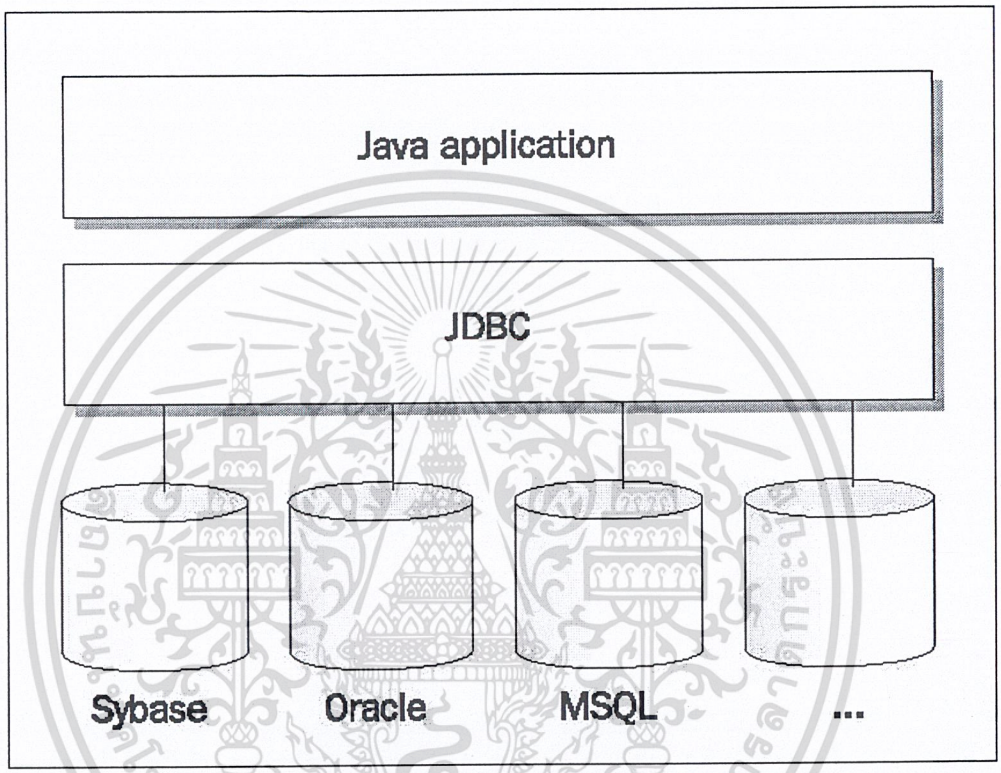
รูปที่ 5-23 รูปแสดงหน้าจอเสร็จสิ้นการสร้าง Synonym



รูปที่ 5-24 รูปแสดงหน้าจอรายการของชื่อเรียกที่ได้ทำการสร้างขึ้นมา

5.6 การเชื่อมต่อเข้ากับฐานข้อมูลโดยใช้ JDBC

JDBC(Java Database Connectivity) ถูกพัฒนาโดย JavaSoft Department ของบริษัท Sun Microsystems ซึ่งก็คือฟังก์ชันมาตรฐานหรือ Java Application Programming Interface(API) สำหรับการเชื่อมต่อกับระบบฐานข้อมูลสัมพันธ์(Relation Database) ใน library ของ JDBC ได้มีส่วนที่ติดต่อกับฐานข้อมูลและได้นำคำสั่ง SQL ไปประมวลผลและควบคุมการทำงานของฐานข้อมูลเชิงสัมพันธ์ ดังแสดงในรูปที่ 5-25



รูปที่5-25 แสดงลักษณะการทำงานของ JDBC จะเป็นตัวกลางระหว่างฐานข้อมูลกับโปรแกรมภาษาจาวา

JDBC ถูกสร้างขึ้นมาในระดับของการเชื่อมต่อกับฐานข้อมูลในรูปแบบที่คล้ายกับ ODBC (ของทางบริษัทไมรโครซอฟ์) หลักการของทั้งสองมาตรฐานตั้งอยู่บนฐานเดียวกันคือ X/Open SQL Call-Level Interface ของระบบ X-windows และ JDBC driver ต้องเข้ากับระดับมาตรฐานในการเข้าถึง SQL (ANSI SQL Entry Level Standards) และต้องผ่าน Conformance test ซึ่ง JavaSoft เป็นผู้กำหนดขึ้น

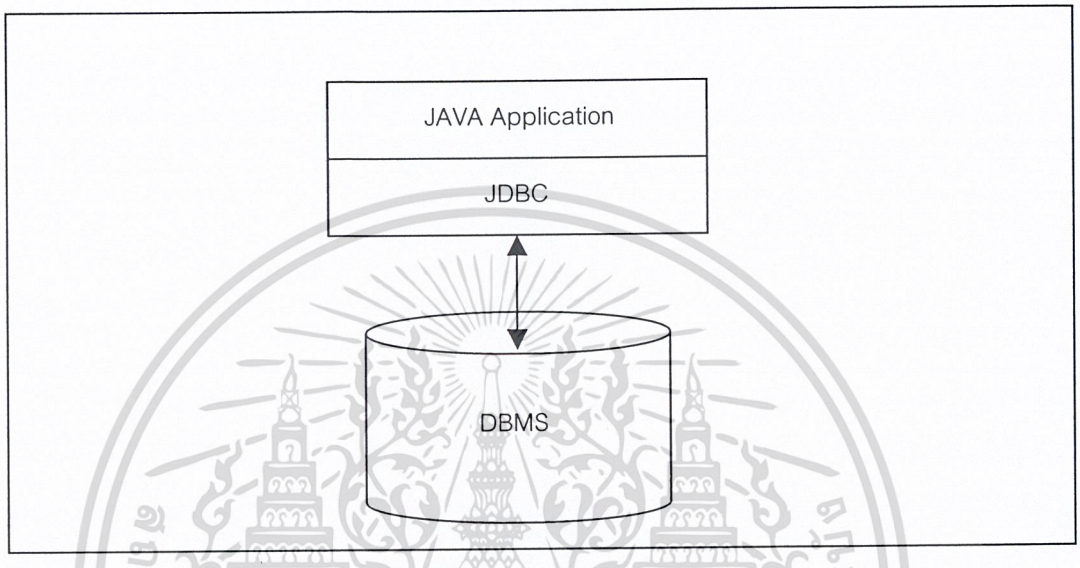
5.6.1 รูปแบบการเชื่อมต่อฐานข้อมูลของ JDBC

JDBC API นั้น สนับสนุนการเชื่อมต่อฐานข้อมูลแบบ 2-Tier และ 3-Teir

- รูปแบบ 2-tier

จาวาแอพเพล็ตหรือจาวาแอพพลิเคชันจะทำการติดต่อกับฐานข้อมูลโดยตรงจึงมีความจำเป็นที่โปรแกรมจาวาต้องการ JDBC Driver พิเศษที่สามารถสื่อสารกับระบบจัดการฐานข้อมูลชนิดนั้นได้ รูปแบบการเชื่อมต่อแสดงรูปที่ 5-26 คำสั่งในการเรียกค้นข้อมูลในรูปแบบของภาษา SQL จะถูกส่งจากผู้ใช้

ไปสู่ฐานข้อมูลหลังจากนั้นการประมวลผลของระบบจัดการฐานข้อมูลก็จะถูกส่งกลับมาที่ผู้ใช้ ฐานข้อมูลดังกล่าวนี้ส่วนมากจะถูกติดตั้งอยู่บนคอมพิวเตอร์อีกเครื่อง โดยผู้ใช้สามารถเชื่อมต่อผ่านระบบเน็ตเวิร์ค รูปแบบ 2-tier นี้ใช้หลักการทำงานเช่นเดียวกันกับรูปแบบไคลเอ็นต์/เซิร์ฟเวอร์ที่เรารู้จักกันดี โดยที่เครื่องคอมพิวเตอร์ของผู้ใช้คือไคลเอ็นต์และเครื่องคอมพิวเตอร์ที่ให้บริการฐานข้อมูลคือเซิร์ฟเวอร์ เครือข่ายคอมพิวเตอร์ที่นิยมใช้รูปแบบ 2-Tier มักจะเป็นเครือข่ายอินทราเน็ต(Intranet) สำหรับดำเนินธุรกรรมภายในองค์กร



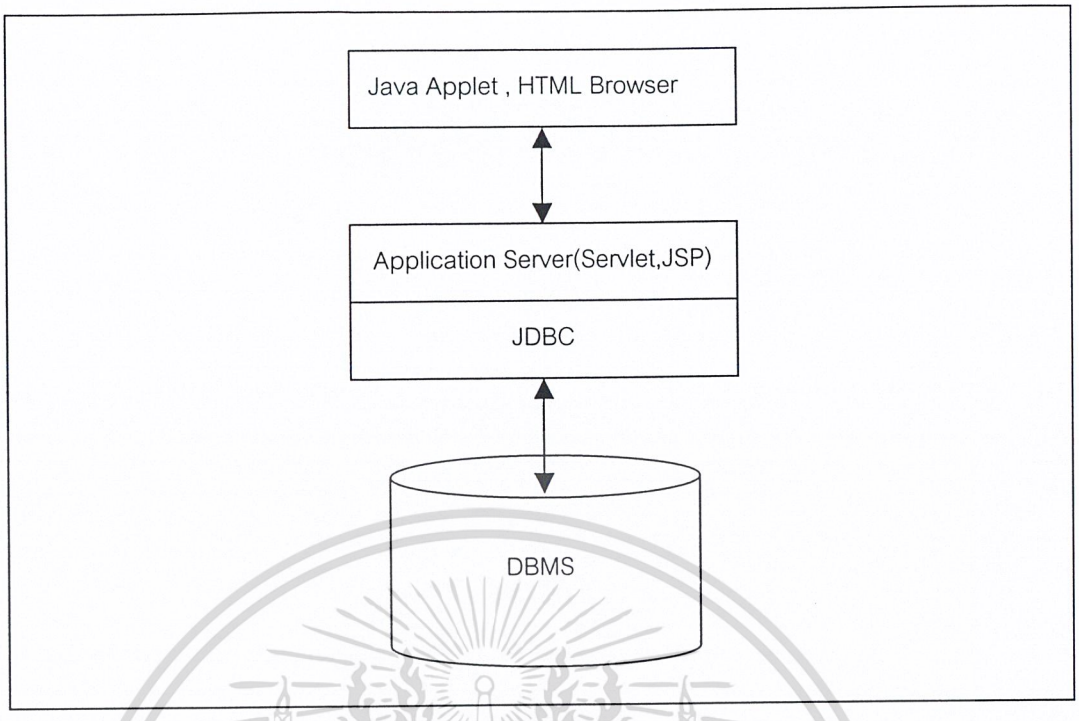
รูปที่ 5-26 รูปแบบ 2-Tier

- รูปแบบ 3-tier

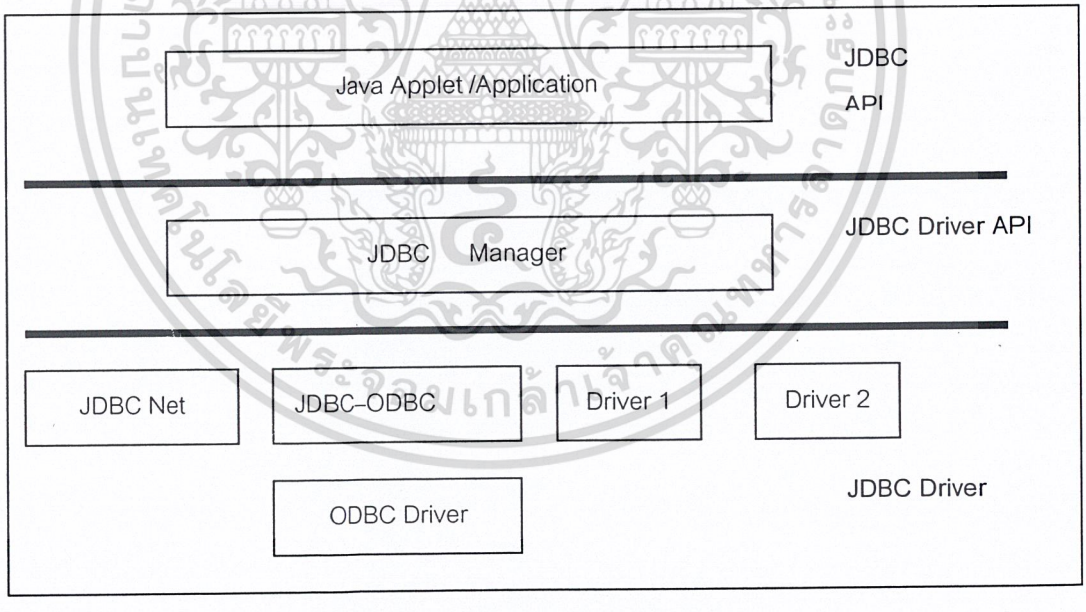
คำสั่งต่างๆจากผู้ใช้จะถูกส่งไปให้กับ Tier กลางหรือตัวกลาง หลังจากนั้น Tier กลางจะแปลงคำสั่งเหล่านั้นให้เป็นภาษา SQL เพื่อส่งให้กับระบบฐานข้อมูลเพื่อทำการประมวลผลและทำการส่งผลลัพธ์คืนให้กับ Tier กลางและส่งให้ผู้ใช้ในที่สุด หลักการทำงานเช่นนี้มักจะพบในเครือข่ายอินเทอร์เน็ตซึ่งประกอบด้วยเครื่องคอมพิวเตอร์หลากหลายชนิดและเว็บเซิร์ฟเวอร์(Web Server)ซึ่งทำหน้าที่เป็น Tier กลางในการจัดการให้ระบบคอมพิวเตอร์ทั้งไคลเอ็นต์และเซิร์ฟเวอร์ฐานข้อมูลสามารถติดต่อกันได้ การปรับเปลี่ยนระบบไคลเอ็นต์และเซิร์ฟเวอร์ หรือแม้กระทั่งการเปลี่ยนตัวฐานข้อมูลตัวใหม่จะไม่มีผลกระทบซึ่งกันและกันเกิดขึ้น แสดงในรูป 5-27

5.6.2 โครงสร้างของ JDBC

โครงสร้างการเชื่อมต่อภายใน JDBC ประกอบด้วย 3 ระดับคือ JDBC API , JDBC Driver API และ JDBC Driver ดังรูปที่ 5-28 ระดับบนสุด เป็น JDBC API เป็นระดับฟังก์ชัน API ที่อำนวยความสะดวกให้แก่โปรแกรมประยุกต์ ระดับกลาง JDBC Driver(มี Driver ที่ต่างกันอยู่ 4 ชนิด) ที่เหมาะสม



รูปที่ 5-27 รูปแบบ 3 Tier



รูปที่ 5-28 ระดับการเชื่อมต่อของ JDBC API

5.6.3 รูปแบบของ JDBC Driver

JDBC Driver หรือส่วนที่อยู่เบื้องหลังการทำงานของฟังก์ชัน API ต่างๆ ของ JDBC ถูกแบ่งออกเป็นทั้งหมด 4 ชนิดคือ

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

- JDBC/ODBC Bridge

ถูกพัฒนาโดย JavaSoft และ InterSolv ทำหน้าที่เป็นตัวกลางในการเข้าถึงฐานข้อมูลได้โดยผ่านทางการทำงานของODBC โดยนำข้อดีของ ODBC-enabled data source ที่มีใช้อยู่โดยทั่วไปอย่างมากมาย ฝั่งไคลเอ็นต์ที่เป็นจาวาแอพเพล็ตหรือจาวาแอพพลิเคชันจะถูกเขียนโดยใช้ JDBC API Bridge จะทำการแปลงการเรียกใช้ JDBC ไปยัง ODBC Driver ที่เหมาะสมสำหรับฐานข้อมูล

ข้อดีของ Bridge ทำให้แอพพลิเคชันสามารถติดต่อกับฐานข้อมูลได้ง่ายจากผู้ผลิตที่มากโดยสามารถเลือกได้จาก ODBC driver ที่เหมาะสม

- Native –API ,partly Java Driver

เป็นลักษณะ Driver แบบ 2-tier ที่ JDBC-driver ต้องการ Vendor library ในการแปลง JDBC function ไปยังคุณลักษณะของการใช้ภาษาในการสอบถามข้อมูล (Query) จาก DBMS เช่น library สำหรับ Sysbase คือ dblib และของ Oracle คือ ocilib ซึ่ง Driver เหล่านี้จะถูกเรียกใช้เพื่อเขียนในการรวมของภาษา Java และ C/C++ การติดต่อโดยใช้ JDBC ประเภทนี้จะเหมือนกับ JDBC/ODBC bridge คือต้องการคำสั่งจาก vendor library เพื่อที่จะใช้ในการติดตั้งบนเครื่องไคลเอ็นต์ ดังนั้นจึงต้องมีการบำรุงรักษาเหมือนกับ Bridge อย่างไรก็ตาม Driver ประเภทนี้จะเร็วกว่าประเภทแรกเพราะว่าส่วนที่มีการแปลงเป็น ODBC ถูกตัดออกไป

- Network-Protocol ,all Java Driver

Driver จะทำการแปลงการเรียกใช้ JDBC ให้อยู่ในรูปของเน็ตเวิร์คโพรโตคอลร่วม (DBMS-independent network protocol) ซึ่งหลังจากนั้นจะถูกแปลงให้อยู่ในรูปเฉพาะของแต่ละฐานข้อมูล (database-specific API)บนเซิร์ฟเวอร์นั้นๆ โดยจะมี Tier กลางในการเชื่อมต่อข้อมูลจะเป็นลักษณะ 3-tier:JDBC driver(ปรกติ 200kb หรือน้อยกว่านั้น) ที่ทำการประมวลผลบนฝั่งไคลเอ็นต์และถูกใช้เครื่องมือทางลอจิก ในการได้รับข้อมูลกลับมาจากเซิร์ฟเวอร์และถูกจัดการ connection.Driver ประเภทนี้ได้รับอนุญาติสำหรับการใช้ Just-in-time บนเครื่องไคลเอ็นต์

JDBC เซิร์ฟเวอร์จะจัดการ connection ที่มีมายังฐานข้อมูลซึ่งดีเหมือน exception และ status event resulting จาก SQL execution แพ็กเกจของข้อมูลสำหรับการติดต่อข้ามระบบเครือข่ายไปยัง JDBC ในฝั่งไคลเอ็นต์

Driverประเภทนี้เหมาะสมมากที่สุดสำหรับ Internet/Intranet-base , multi-user data-intensive application ที่ที่มีจำนวนของ concurrent data operation มากๆ เพราะมีความยืดหยุ่นคล่องตัวมากที่สุด เพราะว่าเขียนขึ้นมาจากภาษาจาวาทั้งหมด ทำให้ไม่จำเป็นต้องมี Driver ร่วมที่เขียนขึ้นมาจากภาษาอื่นซึ่งต้องติดตั้งเฉพาะบนไคลเอ็นต์เท่านั้น และสามารถทำงานบนระบบใดก็ได้ที่สนับสนุนจาวาเวอร์ชวลแมชชีน(Java Virtual Machine) การปฏิบัติงานทุกรูปแบบทั้งดาวน์โหลดข้ามเครือข่ายหรือทำงานบนเครื่องเดียวสามารถกระทำได้หมดเช่นการสอบถาม(Query) และยังมีความสามารถทางด้าน scalability และประสิทธิภาพซึ่งถือเป็นส่วนประกอบหลัก เซิร์ฟเวอร์สามารถ multiplex management ระหว่างฐานข้อมูลจำนวนมาก สามารถจัดหา logging และ administration facility สามารถทำการ load balancing feature และยังสามารถรองรับ catalog และการ Query caches

ซึ่งในโครงงานนี้ได้นำ Driver ประเภทนี้มาใช้งานโดยใช้คลาสของทางออราเคิลซึ่งจะมีตัวอย่างในหัวข้อต่อไป

- Native-protocol ,all-Java Driver

ใน Driver ประเภทนี้จะทำการแปลงการเรียกใช้คำสั่งของ JDBC ให้อยู่ในรูปแบบของโพรโตคอลของระบบเครือข่ายเฉพาะของฐานข้อมูลนั้นโดยตรงไป โดยคุณสมบัติของตัวแทนจำหน่าย Driver ของฐานข้อมูลเหล่านี้สามารถเขียนภาษาจาวาและสามารถติดต่อกับ Applet แบบ just-in- time เพราะว่า Driver เหล่านี้จะแปลง JDBC ตรงไปยัง native protocol โดยปราศจากการใช้ ODBC หรือ native API ซึ่งสามารถจัดหามาสำหรับการติดต่อฐานข้อมูลที่มีประสิทธิภาพสูง Driver เหล่านี้สามารถสร้างจากตัวแทนจำหน่าย DBMS เท่านั้น การติดต่อประเภทนี้มาใช้งานมีอยู่น้อยมากในปัจจุบันแต่คาดว่าในอนาคตจะมีการนำมาใช้มากขึ้น

5.6.4 ขั้นต่อการติดต่อกับระบบจัดการฐานข้อมูล(DBMS) ด้วยภาษาจาวา

สำหรับการเชื่อมต่อกับระบบจัดการฐานข้อมูลด้วยภาษาจาวานั้นจะต้องประกอบด้วยขั้นตอนที่มากพอสมควรโดยจะเห็นได้จากรูปที่ 5-29 โดยแสดงเป็นขั้นตอนดังนี้

1. เรียกใช้คลาสที่จำเป็นเช่น

   import java.sql.;*

2. เลือกใช้ Driver

   DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

3. กำหนดแหล่งของข้อมูลภายในระบบจัดการฐานข้อมูล

   DriverManager.getConnection("jdbc:oracle:thin:@bom:1521:home","home_dba_user", "popbom");

4. สร้างออบเจ็กต์ Connection

   Connection conn =

   DriverManager.getConnection("jdbc:oracle:thin:@bom:1521:home","home_dba_user", "popbom");

5. สร้างออบเจ็กต์Statement

   Statement stmt= conn.createStatement();

6. ประมวลผลคำสั่ง SQL โดยใช้ออบเจ็กต์ Statement

   stmt.executeQuery("insert into table1(attribute1,attribute2) values('test1','test2')");

7. รับผลรับจากประมวลผลคำสั่งด้วยออบเจ็กต์ ResultSet

   ResultSet rset=stmt.executeQuery(query);

8. ปิดออบเจ็กต์ ResultSet

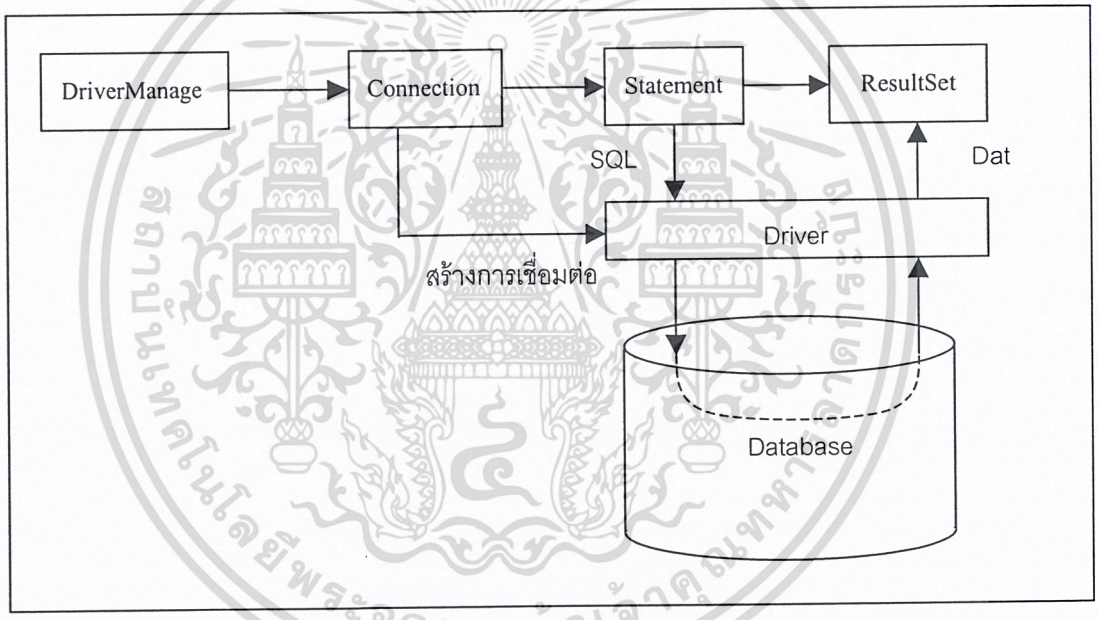
   rset.close();

9. ปิดออบเจ็กต์ Statement

   stmt.close();

10. ปิดออบเจ็กต์ Connection

    conn.close();



รูปที่ 5-29 ลำดับการเชื่อมต่อเข้าฐานข้อมูลโดยใช้ JDBC

ตัวอย่างโปรแกรมการเชื่อมต่อเข้ากับฐานข้อมูลของออราเคิลด้วยภาษาจาวา

```
import java.sql.*;
import java.util.*;
import java.io.*;
public class NameBean {
  Connection conn = null;
  Statement stmt = null;
  ResultSet rset = null;
  String userStatus,status;

  //Construct the application
   public NameBean()
   {
   }
   public void init()
   {
          try {
     DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
              conn    =
     DriverManager.getConnection("jdbc:oracle:thin:@bom:1521:home","home_dba_user", "popbom");
         stmt= conn.createStatement();
         status="Statement is created";
                } catch (SQLException e)
       {     this.status="error";   }
   }
```

```
public String setLogin(String username,String password)//throws SQLException
  {  ResultSet rset2; String login="login_fail";
     try{
     setStmt();
     rset2=setRset("select","SELECT username,password,type FROM home_user WHERE
username='"+username+"' AND password='"+password+"'");
     while(rset2.next())
     {userStatus=rset2.getString(3);
     }
     }catch(SQLException e)
      {userStatus="login_fail";
      }
     return login;
 }

  public voidt  setRset(String type,String query)
  {
   try {
     if(type.equals("select"))
     {rset = stmt.executeQuery(query);
      status="Query are processed";
     }
     else if(type.equals("insert"))
     {stmt.executeUpdate(query);
      status="Insertion are processed";
     }
           } catch (SQLException e)
      {    this.status="error";   }
     return rset;
   }
 }
```

บทที่ 6
หลักการทำงานเบื้องต้นของ Oracle Application Server

ในบทนี้จะขอกล่าวถึงส่วนประกอบของ OAS (Oracle Application Server) ของบริษัทออราเคิล ซึ่งทำหน้าที่เป็นหรับเป็นแอพพลิเคชันเซิร์ฟเวอร์ และหน้าที่การทำงานของ Oracle Application Server Manager และ Oracle Application Server Utilities

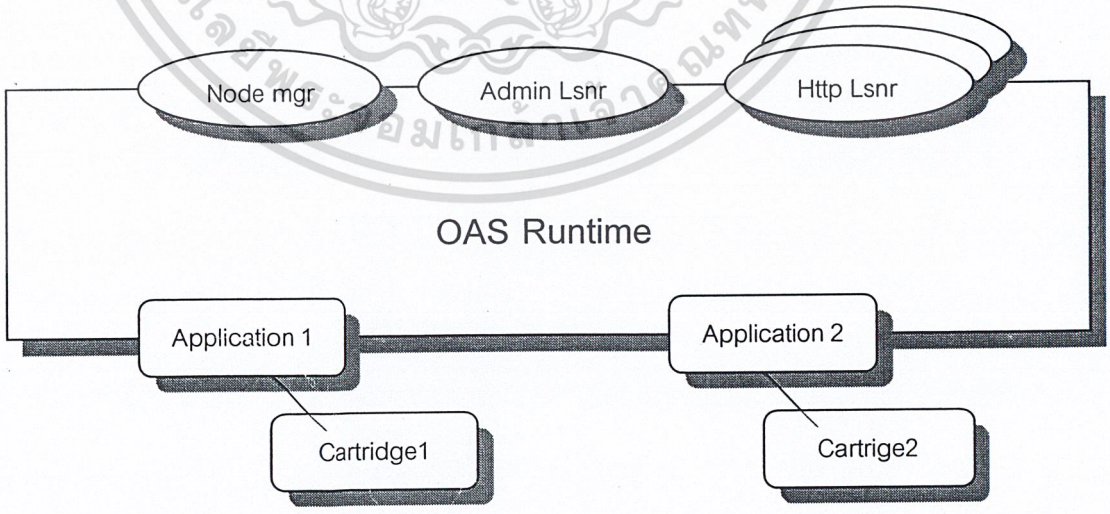
6.1 ส่วนประกอบของ OAS

แนวความคิดของเว็บไซต์คือ กลุ่มของเครื่องคอมพิวเตอร์ที่นำมาเชื่อมต่อกันและมีการใช้งาน Web Request Broker (WRB) ร่วมกัน โดย Web Request Broker คือโพรเซสของออราเคิลแอพพลิเคชันเซิร์ฟเวอร์ที่ทำงานเพื่อทำการจัดการกับทรัพยากรของระบบ

ในแต่ละโหนดในเว็บไซต์นั้นประกอบไปด้วยส่วนต่างๆดังนี้

- listeners ทำหน้าที่รับการร้องขอที่มาจากไคลเอ็นต์
- runtime processes เป็นส่วนที่กำหนดค่าเริ่มต้นที่จำเป็นสำหรับการร้องขอ
- Object Request Broker(ORB) ทำหน้าที่ในการจัดการกับการเชื่อมต่อระหว่างโหนดต่างๆ
- แอพพลิเคชันมีหน้าที่เตรียมส่วนประกอบในการทำงานให้กับ Cartridge เพื่อที่จะนำไปประมวลผล
- Cartridge เป็นโปรแกรมที่จะนำไปประมวลผล

ส่วนประกอบต่างๆ ที่ได้กล่าวมานี้สามารถทำงานอยู่ในเครื่องคอมพิวเตอร์เพียงเครื่องเดียวหรือกระจายไปอยู่บนเครื่องต่างๆซึ่งอยู่ที่การกำหนดองค์ประกอบการทำงานให้ระบบ



รูปที่ 6-1 ส่วนประกอบในเว็บไซต์ของ OAS

HTTP Listener เป็นส่วนประกอบภายในแอพพลิเคชันเซิร์ฟเวอร์ที่ทำหน้าที่รับการร้องขอการใช้งานแอพพลิเคชัน เมื่อได้รับการร้องขอ HTTP Listener จะทำการส่งการร้องขอไปให้กับโพรเซสที่ทำงานอยู่ ขั้นตอนต่อไปโพรเซสนั้นจะทำการกำหนดค่าเริ่มต้นให้กับแอพพลิเคชันและอาจมีการสร้าง Cartridge ขึ้นมา จากนั้นแอพพลิเคชันจะถูกนำไปประมวลผลซึ่งได้มีการเตรียมส่วนประกอบในการทำงานให้กับ Cartridge

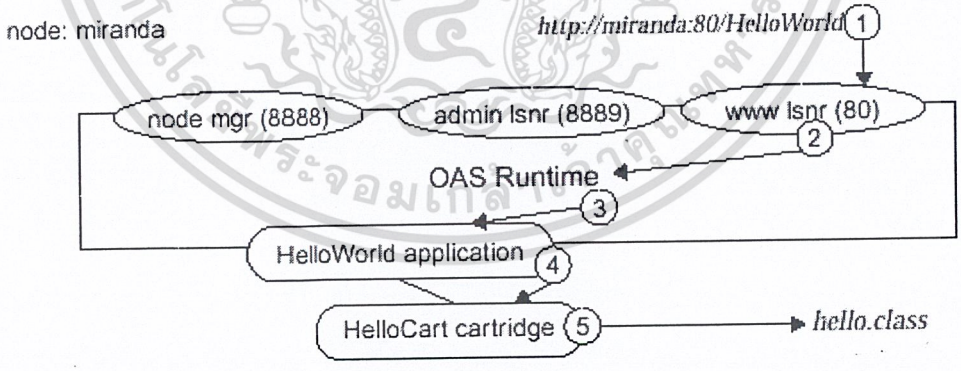
- ตัวอย่างของการร้องขอใช้บริการ

ในรูปที่ 6-2 ได้นำเสนอลำดับเส้นทางการร้องขอใช้งานแอพพลิเคชัน ในตัวอย่างนี้เว็บไซต์นั้นทำงานอยู่บนเครื่องที่มีชื่อว่า miranda และได้มีการใช้หมายเลขพอร์ทปกติสำหรับตัว listener ดังได้แสดงในตารางที่ 6-1

Listener	Port
Node manager	8888
Admin	8889
www	80

ตารางที่ 6-1 หมายเลขพอร์ทปกติที่ใช้ใน OAS

ในเว็บไซต์นี้ได้มีแอพพลิเคชันที่ชื่อว่า Hello World ซึ่งทำการประมวลผล Cartridgeที่เป็น JServlet (เป็นชื่อที่ทางออราเคิลเรียก Cartridge Servlet) ที่มีชื่อว่า HelloCart และภายใน Cartridgeนี้ได้มีการบรรจุคลาสไฟล์จาวา(Servlet)ที่มีชื่อว่า hello.class



รูปที่6-2 ตัวอย่างเส้นทางการร้องขอ

จากรูปที่ 6-2 ได้แสดงตัวอย่างการร้องขอโดยมีลำดับดังต่อไปนี้

1. ไคลเอ็นต์ได้เรียกใช้บริการทาง http://miranda:80/HelloWorld และ www Listener มีหน้าที่รับการร้องขอผ่านทางพอร์ทหมายเลข 80

2. Listener จะส่งค่าการร้องขอไปให้กับโพรเซสรันไทม์และโพรเซสจะทำหน้าที่หาว่าใน HelloWorld แอพพลิเคชันมี Cartridgeที่มีชื่อว่า HelloCart จากนั้นโพรเซสจะหาว่ามีออบเจ็กต์ของ HelloCart ที่ว่างอยู่หรือไม่ ถ้ามีไม่พบก็จะทำการสร้างออบเจ็กต์นั้นขึ้นมา
3. OAS จะเริ่มต้นการทำงานให้กับแอพพลิเคชัน HelloWorld โดยแอพพลิเคชันจะต้องเตรียมส่วนประกอบในการทำงานให้กับ Cartridge ที่มีชื่อว่า HelloCart เนื่องจาก HelloCart นี้เป็น Cartridge ชนิด Jservlet ดังนั้นจึงต้องทำการเริ่มต้นการทำงานให้กับจาวาเวอร์ชวลแมชชีน
4. ออบเจ็กต์ของ HelloCart ถูกประมวลผลด้วยแอพพลิเคชัน HelloWorld
5. ออบเจ็กต์ HelloCart ประมวลผลคำสั่งภายในโปรแกรมภายในคลาสที่ชื่อ hello.class และทำการคืนค่าผลลัพธ์ผ่านตัวโพรเซสรันไทม์และlistener ให้กับผู้ใช้งาน

6.2 หลักการเบื้องต้นของแอพพลิเคชันในมุมมองของOAS

การทำงานหลักของ OAS คือให้บริการแอพพลิเคชันที่ทำงานอยู่ในฝั่งเซิร์ฟเวอร์ ซึ่งเราสามารถทำการพัฒนาแอพพลิเคชันด้วยการเขียนโปรแกรมด้วยภาษาต่างๆ และนอกจากนั้นเราสามารถเลือกโพรโตคอลที่ใช้ในการติดต่อสื่อสาร ระหว่างไคลเอ็นต์กับแอพลิเคชันที่เราได้ทำการสร้างขึ้นมาซึ่งประกอบไปด้วย HTTP หรือ CORBA/IIOP ในลักษณะของไคลเอ็นต์ของแอพพลิเคชันนั้นอาจจะเป็นเว็บบราวเซอร์เช่น Netscape Navigator หรือ Microsoft Internet Explorer หรือจาวาแอพเพล็ตที่ทำงานอยู่ในเว็บบราวเซอร์หรือแม้กระทั่งแอพพลิเคชันที่ทำงานด้วยตัวเอง (Standalone Application) ในหัวข้อนี้จะได้กล่าวถึงหลักการเบื้องต้นของแอพพลิเคชันและ Cartridge รูปแบบการให้บริการของแอพพลิเคชันทั่วไป

6.2.1 Cartridgeและแอพพลิเคชัน

ในการพัฒนาแอพพลิเคชันในฝั่งเซิร์ฟเวอร์ของออราเคิลเว็บเซิร์ฟเวอร์นั้น สามารถแบ่งออกได้เป็น 2 ลักษณะคือ

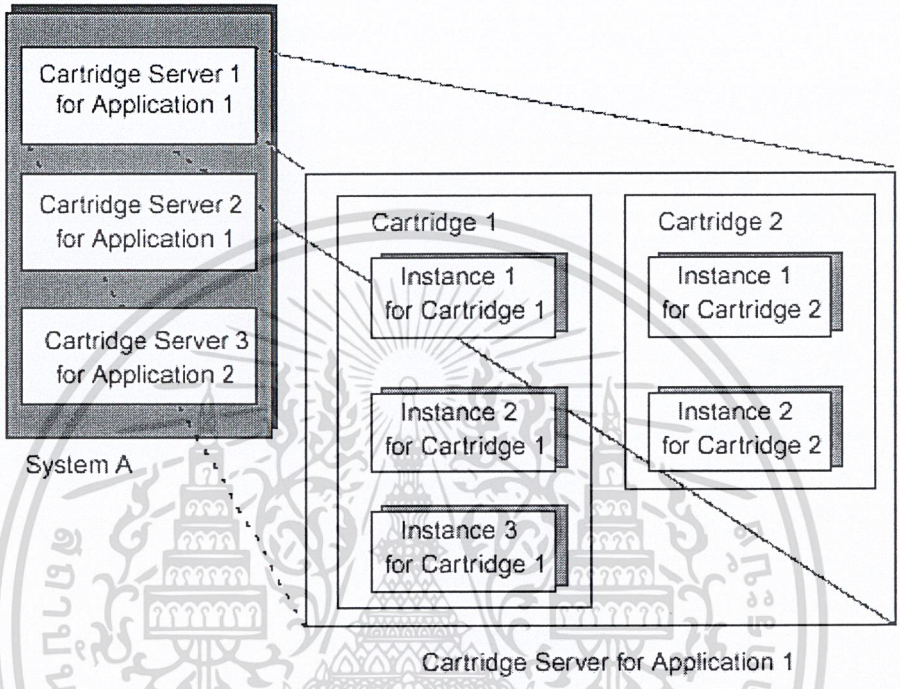
- แอพพลิเคชันที่ทำงานด้วย Cartridge (Cartridge-Based Applications)
- แอพพลิเคชันที่ทำงานด้วย IIOP (IIOP Based Applications)

- แอพพลิเคชันที่ทำงานด้วย Cartridge

OAS นั้นประกอบไปด้วย Cartridgeเซิร์ฟเวอร์ที่สามารถให้บริการแอพพลิเคชันที่ทำงานฝั่งเซิร์ฟเวอร์ โดย Cartridge เซิร์ฟเวอร์นั้นก็คือโพรเซสที่ให้ Cartridgeที่สร้างขึ้นมาทำการประมวลผลกลุ่มคำสั่งและ Instance Cartridge ประกอบไปด้วยกลุ่มของคำสั่งและข้อมูลที่นำไปใช้ในการกำหนดพื้นที่ให้กับ Cartridge และการควบคุม

เมื่อเราสร้างแอพพลิเคชันเราสามารถที่จะกำหนดองค์ประกอบโดยใช้เครื่องมือที่มีชื่อว่าออราเคิลแอพพลิเคชันเซิร์ฟเวอร์เมเนเจอร์(Oracle Application Server Manager)โดยเราสามารถที่จะกำหนดองค์-ประกอบในสองระดับคือในระดับของแอพพลิเคชันเป็นการกำหนดค่าพื้นฐานให้กับกลุ่มของ Cartridge ที่เป็นประเภทเดียวกันในแต่ละแอพพลิเคชัน(แสดงในตารางที่ 6-2) และในระดับของ Cartridge เป็นการกำหนดองค์ประกอบเฉพาะสำหรับแต่ละ Cartridge ซึ่งจะนำไปใช้กับ Instance Cartridge ที่สร้างมาจาก Cartridge ชนิดนั้น

ตัวอย่างในรูปที่ 6-3 ในระบบ A ประกอบไปด้วย Cartridge เซิร์ฟเวอร์ 3 ตัวซึ่งเป็นของแอพพลิเคชัน 1 สองตัวและสำหรับ แอพพลิเคชัน 2 หนึ่งตัว Cartridge เซิร์ฟเวอร์คือโพรเซสที่ถูกควบคุมด้วยแอพพลิเคชันเซริฟเวอร์ที่ทำงานอยู่ในระบบ A เมื่อพิจารณาที่ Cartridge เซิร์ฟเวอร์ที่อยู่ในแอพพลิเคชัน 1 นั้นมีอยู่สอง Cartridge ที่ทำงานอยู่ ในCartridgeแรกมีอินสแทนซ์ของ Cartridge1 สามอินสแทนซ์และอินสแทนซ์ของ Cartridge2 สองอินสแทนซ์



รูปที่ 6-3 ตัวอย่างแอพพลิเคชันและ Cartridge

ในตารางที่ 6-2 ได้นำเสนอ Cartridge ที่OASได้เตรียมให้ โดยแอพพลิเคชันที่ทำงานด้วย Cartridge นั้นจะใช้โพรโตคอล HTTP ในการติดต่อสื่อสารเพื่อรับการร้องขอจากไคลเอ็นต์

ชนิดของ Cartridge	ลักษณะการทำงานของโปรแกรม
C Cartridge	มีการใช้ Library(UNIX) หรือ DLL(NT)ร่วมกัน
Jservlt, JWeb Cartridge	Java Class file
JSP Cartridge	มีการนำ directive และ tagไปรวม ในไฟล์JSP
LiveHTML Cartridge	มีการนำ Script อยู่ในไฟล์ HTML
Perl Cartridge	Perl Scripts
PL/SQL	Stored Procedures ในฐานข้อมูลหรือไฟล์ SQL script

ตารางที่ 6-2 แสดงลักษณะต่างของ Cartridge

- แอพพลิเคชันที่ทำงานด้วย IIOP

ในOAS สามารถให้เราทำการพัฒนาแอพพลิเคชันที่ใช้หลักการกระจายออบเจ็กต์ด้วยมาตรฐานของCORBA ในการติดต่อสื่อสารด้วยโพรโตคอล IIOP และแอพพลิเคชันนั้นสามารถรองรับรูปแบบของแอพพลิเคชันได้ดังนี้

- แอพพลิเคชันในรูปแบบ ECO/Java
- แอพพลิเคชันในรูปแบบ Enterprise JavaBean(EJB)
- แอพพลิเคชันในรูปแบบ C++ CORBA Cartridge

ECO/Java และ EJB คือการสร้างออบเจ็กต์ของ CORBA ที่สามารถเข้าใช้งานได้จากไคลเอ็นต์ชนิดต่างๆ เช่น จาวาแอพเพล็ต, จาวาแอพพลิเคชันและ CORBA Application ส่วน C++, CORBA Cartridge นั้นเป็นการกำหนดสถาปัตยกรรมสำหรับสร้างแอพลิเคชัน ในลักษณะของออบเจ็กต์แบบกระจายในทางธุรกิจ ด้วยโปรแกรมภาษาC++

6.3 แอพพลิเคชันในลักษณะ JServlet Cartridge

ภายใน Jservlet Cartridge คือ จาวาเวอร์ชวลแมชชินและจาวาคลาส Library มีหน้าที่ในการเตรียมส่วนประกอบในการทำงานของแอพพลิเคชันที่เขียนด้วยข้อกำหนดทางภาษาจาวา Servlet และทำงานอยู่ในฝั่งเซิร์ฟเวอร์ ข้อดีของการใช้งาน JServlet Cartridge พอที่จะสรุปได้ดังนี้

- มีประสิทธิภาพที่ดีกว่าซีจีไอ(CGI) เพราะว่าไม่ต้องมีการเริ่มต้นและหยุดการทำงานของจาวาเวอร์ชวลแมชชีนเมื่อมีการร้องขอเข้ามาในแต่ละครั้งตัวอย่างเช่นการเชื่อมต่อกับฐานข้อมูลสามารถสร้างการเชื่อมต่อเพียงครั้งเดียวสำหรับในแต่ละการเชื่อมต่อ (session) ทำให้ไม่ต้องสามารถเชื่อมต่อฐานข้อมูลทุกครั้งที่มีการร้องขอจากไคลเอ็นต์
- สามารถใช้ข้อดีของออราเคิลแอพพลิเคชันเซิร์ฟเวอร์ในการจัดการด้านความสมดุลย์ การขยายขอบเขต, การตรวจสอบ, การติดตามการใช้งานและความสามารถในการบริหารการติดต่อ
- มีการการใช้งานทรัพยากรของระบบน้อยเพราะมีการทำงานของ JServlet Cartridge หลายๆ ตัวสามารถทำงานอยู่บนจาวาเวอร์ชวลแมชชีนเดียวกันได้และนอกจากนั้นอินสแทนซ์ที่ว่างสามารถนำกลับมาใช้ได้ใหม่
- สามารถใช้ชุดเครื่องมือของทาง JServlet ซึ่งประกอบด้วยชุดของคลาสไฟล์ ที่สามารถสร้างเอกสาร HTML และสามารถเชื่อมต่อกับฐานข้อมูล

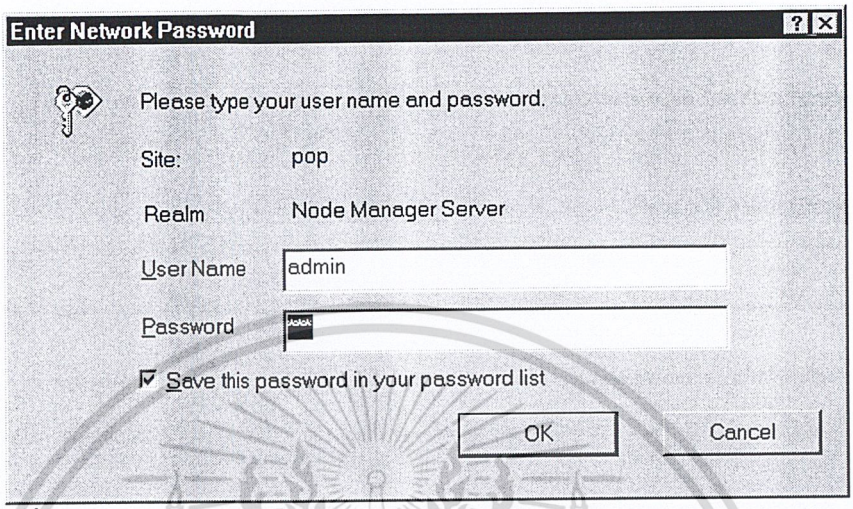
การใช้งาน JServlet นั้นจะทำงานอยู่ภายในฝั่งเซิร์ฟเวอร์ที่แตกต่างกับแอพแพล็ตที่ต้องโหลดและไปทำงานอยู่ในฝั่งไคลเอ็นต์

6.4 การสร้างแอพพลิเคชันและ Cartridge แบบ JServlet

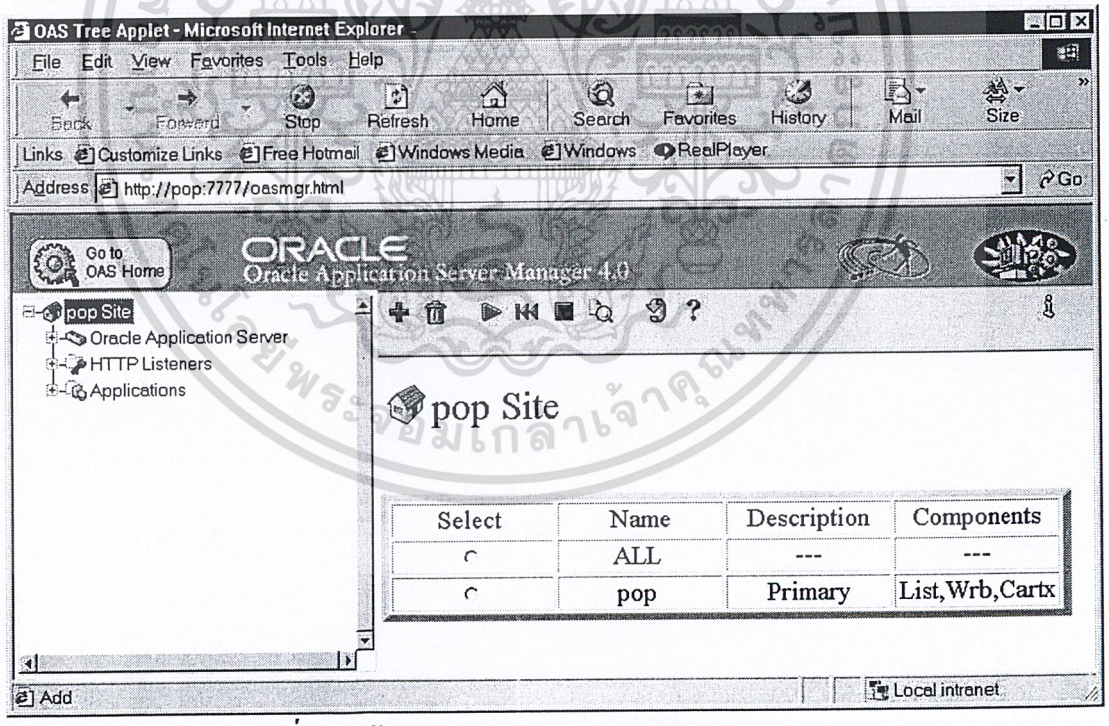
1.เข้าสู่โปรแกรม Oracle Application Server → Oracle Application Server Manager จะปรากฏหน้าจอเพื่อป้อนรหัสผ่าน(กำหนดเมื่อติดตั้ง)ดังรูป ที่ 6-4

2. เมื่อป้อนรหัสผ่านแล้วจะเข้าสู่โปรแกรม Oracle Application Server Manager ดังรูปที่ 6-5

3. คลิกที่เครื่องหมาย ⊞ ในเฟรมด้านซ้ายหน้าชื่อไซต์เพื่อขยายเมนูการทำงาน จากนั้นคลิกที่ แอพพลิเคชัน เพื่อแสดงรายการของแอพพลิเคชันทางด้านขวาเท่านั้น



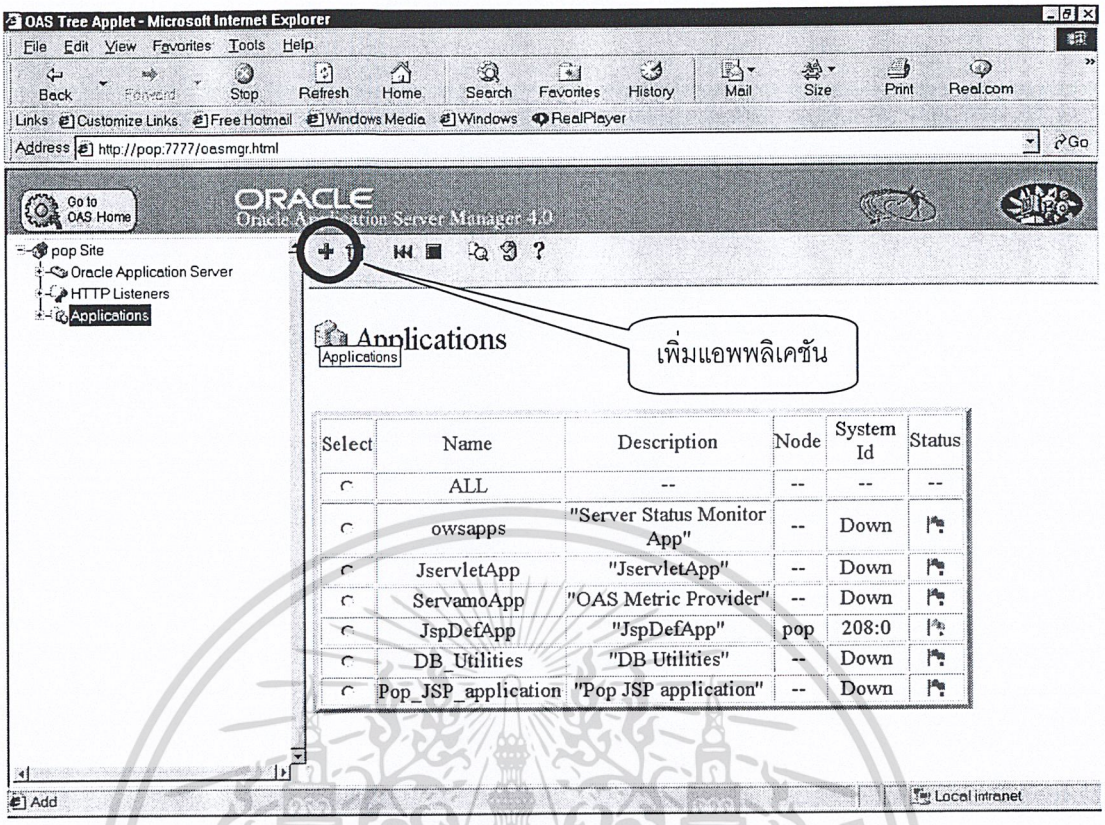
รูปที่ 6-4 เมื่อเรียกใช้งานโปรแกรม Oracle Application Server Manager จะต้องมีการป้อนรหัสผ่านสำหรับผู้ดูแลระบบ



รูปที่ 6-5 หน้าจอของ Oracle Application Server Manager

4. ในเฟรมด้านขวาที่แสดงหน้าจอแอพพลิเคชัน ที่แสดงในรูปที่ 6-6 ให้คลิกที่ปุ่มเพิ่มแอพพลิเคชัน

5.ในหน้าจอการเพิ่มแอพพลิเคชันให้ทำการใส่ข้อมูลดังรูปที่ 6-7 เพื่อกำหนดชนิดของแอพพลิเคชันและควรกำหนดการกำหนดองค์ประกอบการทำงาน (Configure Mode) เป็นกำหนดเอง (Manual) จากนั้นคลิกปุ่ม “Apply”



รูปที่6-6 แสดงหน้าจอแอพพลิเคชัน

6. ขั้นตอนต่อไปจะเป็นการกำหนดชื่อให้กับแอพพลิเคชันโดยจะต้องมีค่าต่างๆดังต่อไปนี้

Application Name ใช้ในการระบุชื่อของแอพพลิเคชัน

Display Name เป็นชื่อที่ใช้ในหน้าจอของการบริหาร

Application Version เป็นการกำหนดรุ่นของแอพพลิเคชัน

ดังแสดงในรูปที่ 6-8 จากนั้นคลิกที่ปุ่ม “Apply” จะปรากฏหน้าจอที่แจ้งว่าการเพิ่มแอพพลิเคชันนั้นเสร็จสิ้นและจะเข้าสู่หน้าจอการเพิ่ม Cartridge ให้กับแอพพลิเคชัน

7. เมื่อขึ้นหน้าจอเสร็จสิ้นการเพิ่มแอพพลิเคชันคลิกที่ “Add Cartridge to this Application” ดังรูปที่ 6-9

8. จากนั้นจะเป็นขั้นตอนในการเพิ่ม JServlet Cartridge ดังรูปที่ 6-10 โดยจะต้องทำการป้อนข้อมูลต่างๆ ได้แก่

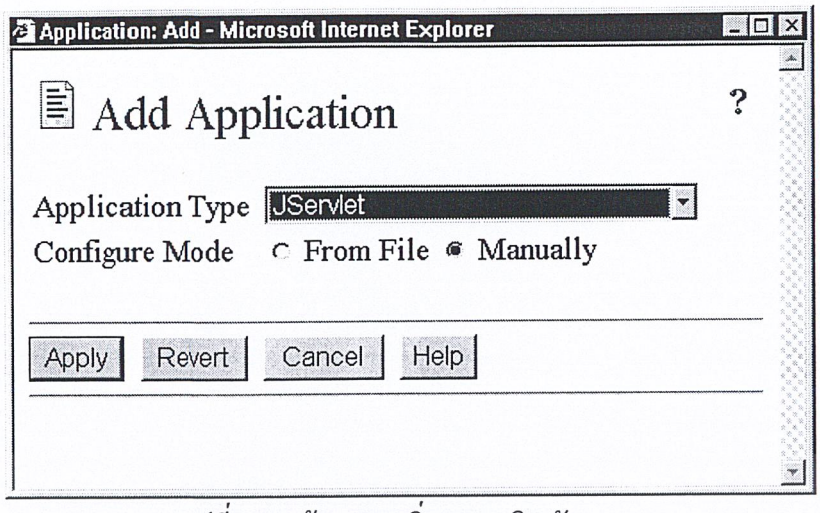
Cartridge Name ใช้ในการระบุชื่อ Cartridgeในแอพพลิเคชัน

Display Name เป็นชื่อที่ใช้ในหน้าจอของการบริหาร

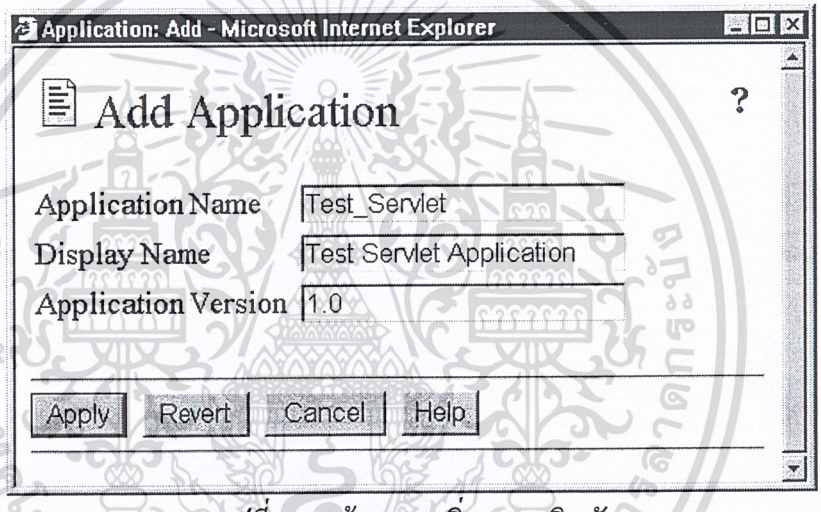
Virtual Path คือชื่อที่สามารถเรียกจากไคลเอ็นต์

Physical Path เป็นตำแหน่งของไดเรกทอรีที่ทำการบรรจุคลาสไฟล์จาวา

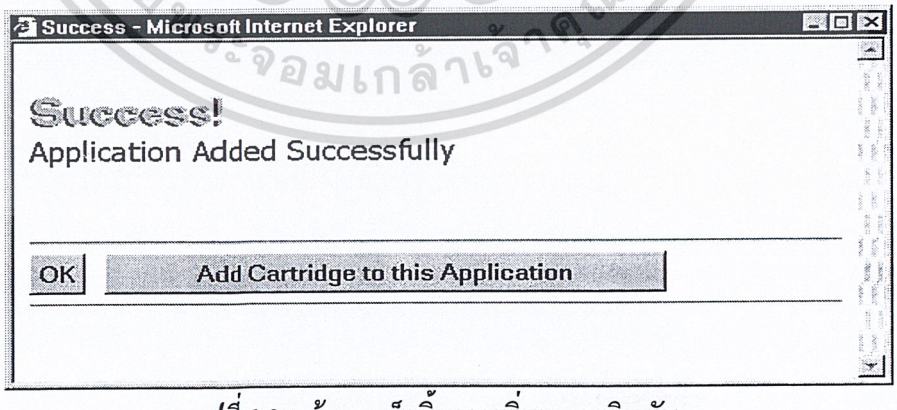
จากนั้นเราจะได้ Cartridgeสำหรับแอพพลิเคชันของ Jservlet



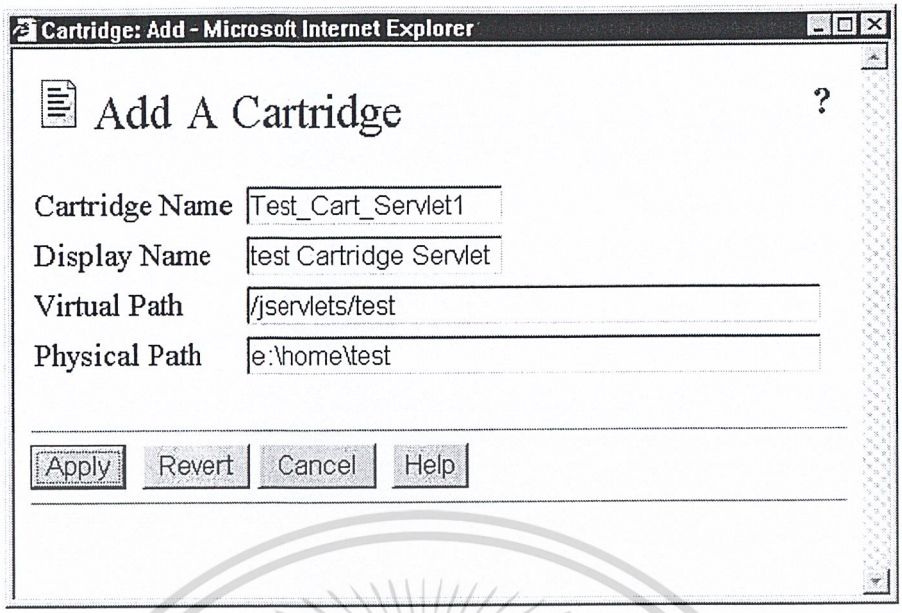
รูปที่ 6-7 หน้าจอการเพิ่มแอพพลิเคชัน



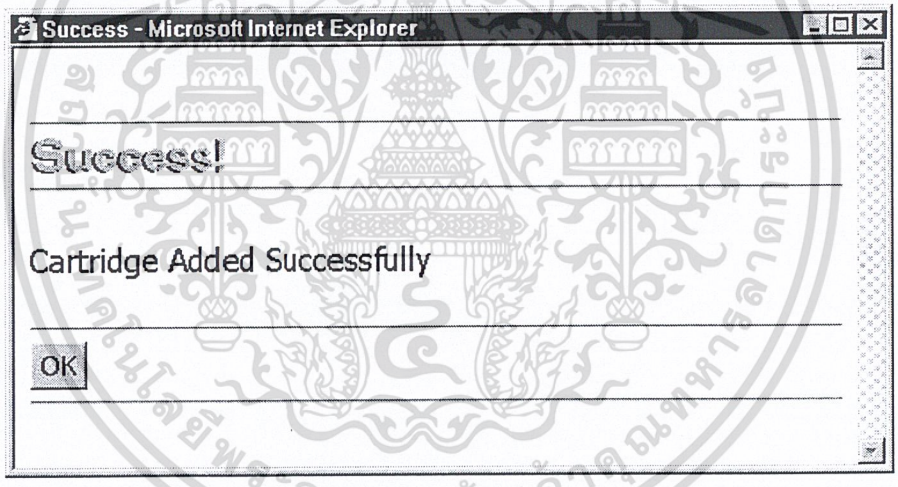
รูปที่ 6-8 หน้าจอการเพิ่มแอพพลิเคชัน 2



รูปที่ 6-9 หน้าจอเสร็จสิ้นการเพิ่มแอพพลิเคชัน



รูปที่ 6-10 หน้าจอการเพิ่ม Cartridge ของแอพพลิเคชันชนิด JServlet

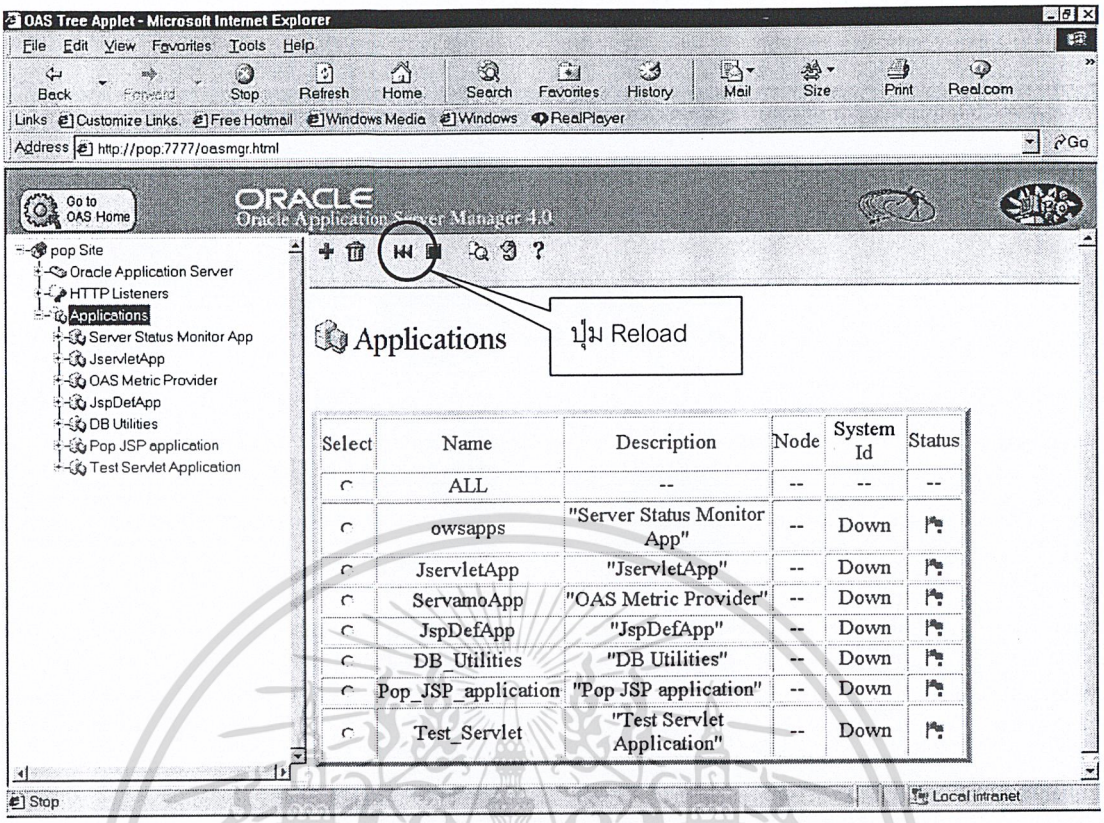


รูปที่ 6-11 หน้าจอเมื่อการเพิ่ม Cartridge เสร็จสิ้น

6.5 การเริ่มต้นการทำงานใหม่ของแอพพลิเคชัน

เมื่อเราได้ทำการเพิ่มแอพลิเคชันเราจะต้องมีการ เริ่มต้นการทำงานใหม่ให้กับแอพพลิเคชันที่ได้ทำการสร้างขึ้นมาหรือแอพลลิเคชันที่ได้ทำการสร้างมาแล้ว ด้วยขั้นตอนดังต่อไปนี้

1. ในหน้าจอ Oracle Application Server Manager ไปยังส่วนของแอพพลิเคชันดังแสดงในรูปที่ 6-12
2. เลือกแอพพลิเคชันที่ต้องการ หรือ “ALL” จากนั้นคลิกปุ่ม Reload ที่อยู่ในเฟรมทางด้านขวา



รูปที่ 6-12 หน้าจอการเริ่มต้นการทำงานใหม่ให้กับแอพพลิเคชัน

6.6 การเรียกใช้งาน Cartridge JServlet

เราสามารถเรียกใช้งาน Cartridgeแบบ JServlet ได้โดยการพิมพ์ URL ในบราวเซอร์เช่น

http://<ชื่อเครื่อง>:<หมายเลขพอร์ท>/JServlets/test/HelloServlet

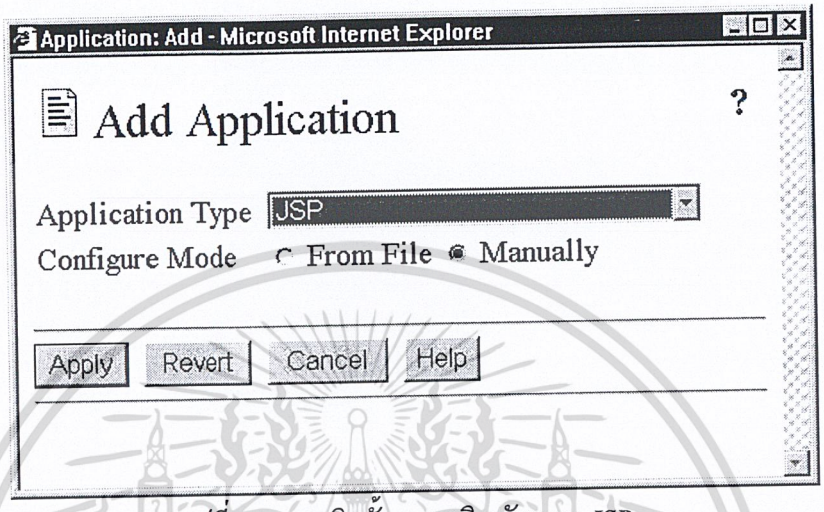
โดย*<ชื่อเครื่อง>*และ*<หมายเลขพอร์ท>* สามารถกำหนดได้จากตัวที่คอยรับฟังการเรียกใช้ (Listener)

และอีกวิธีหนึ่งคือการเรียกใช้ Cartridge โดยผ่านทางหน้าจอ HTML ในการสร้างหน้าจอด้วยภาษา HTML ที่ทำการเรียกใช้ Cartridge JServlet โดยจะต้องมีส่วนที่เรียกไปยัง URL ของ Cartridge ตัวอย่างเช่น

```
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>My First JServlet</H1>
<P><A HREF=" http:// <host>:<port>/jservlets/test/HelloServlet">Run my Hello
World class</a>
</BODY>
</HTML>
```

6.7 การติดตั้งและใช้งาน Cartridge ชนิด JSP

ในOAS เวอร์เวอร์ชัน 4.0.8.2.1 ได้ทำการติดตั้ง Cartridge แบบ JSP เพื่ออำนวยความสะดวกในการใช้งานโปรแกรมที่เขียนด้วย JSP และการเพิ่มแอพพลิเคชันที่เป็นแบบ JSP นั้นก็เหมือนกับการเพิ่มแอพพลิเคชันที่เป็นแบบ JServlet แต่จะแตกต่างกันที่การกำหนดชนิดของแอพพลิเคชันให้เป็นแบบ JSP ดังแสดงในรูป 6-13

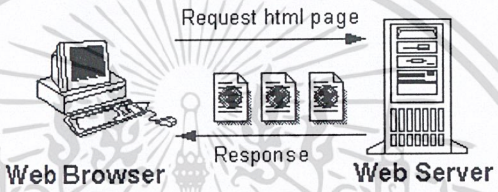


รูปที่ 6-13 การติดตั้งแอพพลิเคชันแบบ JSP

บทที่ 7
สถาปัตยกรรมของ แอพพลิเคชันเซิร์ฟเวอร์

7.1 สถาปัตยกรรมไคลเอ็นต์/เซิร์ฟเวอร์

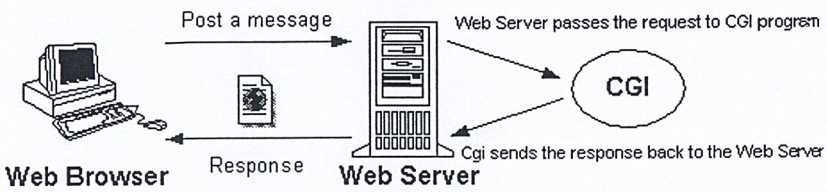
วิวัฒนาการของ Server Side Application ที่ถูกใช้บนเว็บไซต์ตั้งแต่สมัยแรก ๆ จนถึงปัจจุบันนั้น จะเห็นว่าสมัยแรกสุดตอนที่มีเว็บไซต์เกิดขึ้นมาใหม่ ๆ ส่วนประกอบหลักของเว็บไซต์ไม่ค่อยจะมีอะไรซักเท่าไหร่เริ่มแรกสุดก็ทำกันอย่างง่าย ๆ โดยใช้หลักการพื้นฐานของสถาปัตยกรรมไคลเอ็นต์/เซิร์ฟเวอร์ (Client/Server Architecture) ซึ่งส่วนที่เป็นไคลเอ็นต์ในเว็บไซต์ก็คือเว็บบราวเซอร์และส่วนที่เป็นเซิร์ฟเวอร์ซึ่งก็คือเว็บเซิร์ฟเวอร์ โดยทั่วไป เว็บเซิร์ฟเวอร์หนึ่ง ๆ จะสามารถบริการส่งข้อมูลให้ตัวเว็บบราวเซอร์ได้หลายตัว



รูปที่ 7-1 สถาปัตยกรรมไคลเอ็นต์/เซิร์ฟเวอร์ เริ่มแรกสุดที่ใช้กันบนเว็บไซต์

7.2 CGI

ต่อมาเทคโนโลยีที่เกิดขึ้นตามมาก็คือ CGI (Common Gateway Interface) หลักการของ CGI คือแทนที่ข้อมูลที่เราส่งไปให้ เว็บเซิร์ฟเวอร์ (โดยผ่านทางเว็บบราวเซอร์) จะถูกโพรเซสโดยตัว เว็บเซิร์ฟเวอร์ อย่างเดียว ข้อมูลนี้ก็อาจจะถูกส่งไปให้อีกโปรแกรมหนึ่งซึ่งตัวโปรแกรมนี้จะสามารถถูกเรียกได้โดย เว็บเซิร์ฟเวอร์ เพื่อทำการประมวลผลข้อมูลที่ร้องขอจากเรา โดยทั้งนี้ข้อมูลของเราจะถูกโพรเซสที่ใดก็ขึ้นอยู่กับเซตค่าต่าง ๆ ที่ทางเว็บไซต์ที่เราเข้าไปดูกำหนดขึ้นมา ยกตัวอย่างเช่น ถ้าเราต้องการดูเพจธรรมดา ๆ หลักจากที่เราคลิกไปที่ตัวลิงค์ตัว เว็บเซิร์ฟเวอร์ ก็จะไปอ่านเพจ (html ไฟล์) ที่เราต้องการแล้วส่งกลับมาเป็น text stream ให้เรา



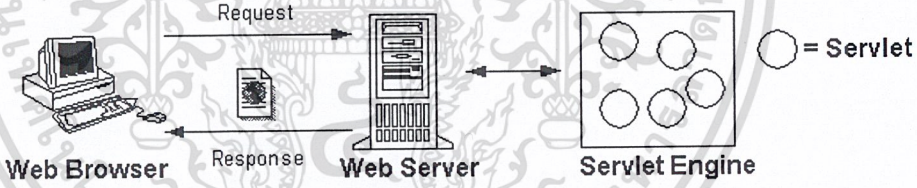
รูปที่ 7-2 การ post ข้อความไปยัง CGI โปรแกรม

7.3 Servlet

จาวามีสโลแกนอันหนึ่งที่กล่าวไว้ว่า "สิ่งใดก็ตามที่เป็นที่นิยมใช้กันอย่างกว้างขวาง ท้ายที่สุดสิ่งนั้นจะกลายเป็นจาว่า library" โดยผลพวงหนึ่งที่มาจากสโลแกนอันนี้ก็คือ Servlet นั่นเอง

Servlet อ้างอิงหลักการของ CGI โดยข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาวานั่น จาว่าเป็นภาษาที่ใช้คอนเซ็ปต์ของ 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 ไคลเอ็นต์ ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request ซึ่งจะทำให้เปลืองทรัพยาการมากกว่าและ process ในการรันก็จะช้ากว่าด้วย

ถึงแม้ว่า Servlet จะอ้างอิงหลักการของ CGI อย่างไรก็ตามในการที่จะทำการรัน Servlet แล้วตัวเว็บเซิร์ฟเวอร์ จะไม่สามารถส่งข้อมูลไปให้ Servlet ได้โดยตรงเหมือนกับหลักการของ CGI แต่ตัว เว็บเซิร์ฟเวอร์ จะต้องเพิ่มอีกส่วนหนึ่งซึ่งเป็นส่วนที่ใช้เป็นเสมือนตัวห่อหุ้ม Servlet ต่าง ๆ ไว้โดยส่วนที่เพิ่มขึ้นมานี้เราเรียกว่า Servlet Engine หรือ Servlet Container



รูปที่ 7-3 Servlet Engine and its Servlets

โดยทั่วไป Servlet Engine จะเป็นส่วนที่มี JVM (Java Virtual Machine) อยู่ในตัวเองโดย Servlet Engine นี้จะมีหน้าที่รับ request จาก เว็บเซิร์ฟเวอร์ (ซึ่งมาจาก เว็บบราวเซอร์) แล้วทำการเลือกตัว Servlet ขึ้นมาทำการประมวลผล request นั้นภายใต้ JVM ของมัน โดยผลที่ได้จากการประมวลผลของ Servlet ที่ถูกเลือกจะถูกส่งกลับไปยัง เว็บเซิร์ฟเวอร์ โดย เว็บเซิร์ฟเวอร์ นี้จะส่งผลกลับไปยัง เว็บบราวเซอร์ ในท้ายที่สุด

7.3.1โครงสร้างของ Servlet

การรับส่งข้อมูลโดยใช้เว็บ ผ่านทางแท็กฟอร์ม <FORM ...> มีรูปแบบการส่งอยู่ 2 ลักษณะคือ GET และ POST โดยที่การส่งแบบ GET เป็นรูปแบบการส่งข้อมูล ผ่านทาง URL (ช่องพิมพ์ชื่อเว็บหรือเว็บแอดเดรส) ซึ่งผู้ใช้อาจจะพิมพ์ได้เองโดยตรง เช่น

http://www.javacentrix.com?**user=1245&pwd=4444** หรือฟอร์มที่แท็กกำหนดแอตทริบิวต์ METHOD เป็น GET เช่น <FORM **METHOD="GET"** ... เป็นต้น (ตัวอย่างที่เป็นตัวหนาคือการส่งข้อมูลผ่านฟอร์มในแบบ GET) สำหรับการส่งข้อมูลผ่านฟอร์มในแบบ POST คือการกำหนดแท็กฟอร์มให้มีแอตทริบิวต์ METHOD เป็น POST เช่น <FORM **METHOD="POST"** ... เป็นต้น

7.3.2 Servlet ส่งข้อมูลรูปแบบ HTML

การสร้าง Servlet ขึ้นมาใช้งานความต้องการคือ ให้สามารถส่งข้อมูลเว็บ ซึ่งก็คือข้อมูลในรูปแบบ HTML นั่นเอง ดังนั้นการส่งข้อมูล HTML ที่สำคัญ จำเป็นต้องส่งข้อมูลบอกลักษณะของรูปแบบ ให้กับผู้ใช้ (ซึ่งเป็นข้อมูลที่บราวเซอร์ได้รับ) ว่าเป็นข้อมูลรูปแบบ HTML ซึ่งเรียกกันว่า Content-Type แบบ text/html การส่งข้อมูลบอกบราวเซอร์นี้ ทำได้โดยการเรียกใช้เมธรอด setContentType ของคลาส HttpServletResponse แล้วส่งข้อความ text/html เป็นอาร์กิวเมนต์ไป

7.3.3 การอ่านข้อมูลจากเว็บ

สำหรับการใช้งานเว็บผ่านบราวเซอร์ โดยทั่วไปแล้ว ปกติในช่องที่ใช้กำหนดข้อมูล URL จะประกอบด้วยชื่อเซิร์ฟเวอร์ ชื่อไดเรกทรอรีและชื่อไฟล์เว็บเพจที่ต้องการดู เช่น หากข้อมูล URL คือ http://www.javacentrix.com/tutorials/api/index.htm นั่นหมายถึงต้องการอ่านเว็บเพจที่ชื่อ index.htm จากไดเรกทรอรี tutorials/api จากเว็บเซิร์ฟเวอร์ที่ชื่อ www.javacentrix.com

สำหรับ URL นั้นนอกจาก จะใช้ชี้ตำแหน่งไฟล์ที่ต้องการเรียกดูแล้ว ยังสามารถใช้สำหรับส่งข้อมูลให้กับระบบเซิร์ฟเวอร์ เพื่อใช้ในการทำงาน หรือดำเนินการอย่างใดอย่างหนึ่ง ลักษณะ URL ประเภทนี้ หลังจากชื่อไฟล์แล้ว จะตามด้วยเครื่องหมายคำถามและข้อมูลที่ต้องการส่งไปให้เซิร์ฟเวอร์ ซึ่งเรียกโดยทั่วไปว่าการส่งข้อมูลแบบ GET โดยมีรูปแบบคือ

URL?Parameter1=Data1&Parameter2=Data2

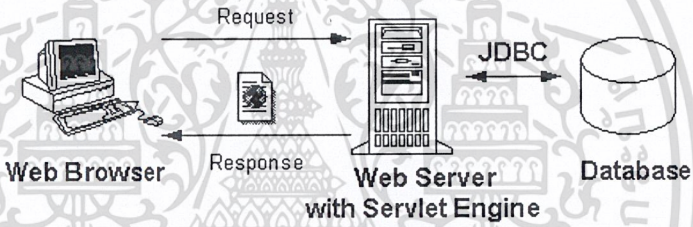
จากรูปแบบการส่งข้อมูลผ่าน URL โดย Parameter1 และ Parameter2 คือชื่อพารามิเตอร์ที่ถูกเซิร์ฟเวอร์อ้างถึงได้ ในขณะที่ Data1 และ Data2 คือข้อมูลที่อยู่ในพารามิเตอร์แต่ละตัว หากต้องการส่งข้อมูลมากกว่า 2 ตัว ก็จะเพิ่มต่อท้ายไปได้เรื่อยๆ เช่น

URL?Parameter1=Data1&Parameter2=Data2&Parameter3=Data3&Parameter4=Data4

7.4 Relational Database

ช่วงที่มี Server Side Application ใหม่ ๆ ข้อมูลต่าง ๆ มักจะถูกเก็บไว้ในไฟล์โดยโปรแกรมจะทำการอ่านไฟล์ทุกครั้งที่มีการโหลดข้อมูลดังกล่าวขึ้นมาใช้ ตัวอย่างที่เราพบเห็นกันโดยทั่วไปสำหรับโปรแกรมที่ต้องอ่านหรือเก็บข้อมูลต่าง ๆ ลงไปในไฟล์ยกตัวอย่างเช่น เว็บบอร์ด, guest book เป็นต้น การจัดเก็บข้อมูลในไฟล์มีผลเสียในแง่ของการยากต่อการดูแลและจัดเก็บ ตลอดจนประสิทธิภาพและความเร็วที่ลดลงของโปรแกรมในแง่ของการตอบสนองกับผู้ใช้ปลายทางเนื่องจากที่โปรแกรมมีกิจกรรมที่เกี่ยวข้องกับอินพุต-เอาต์พุตเป็นหลัก ซึ่งในท้ายสุดการจัดเก็บข้อมูลในไฟล์ก็ถูกแทนที่ด้วยการจัดเก็บข้อมูลในดาต้าเบสแทน

ดาต้าเบสเป็นส่วนที่ถูกเพิ่มขึ้นมาใน Server Side Application โดยมักจะเป็นส่วนที่อยู่ทางท้ายสุดของระบบ ส่วนที่จะเป็นตัวเรียกใช้ดาต้าเบสมักจะเป็นส่วนที่ทำหน้าที่ประมวลผล request ที่มาจากผู้ใช้ซึ่งโดยทั่วไปก็คือ Servlet Engine นั่นเอง ในกรณีของ JSP ตัว JSP Container จะเป็นส่วนที่ทำการติดต่อกับดาต้าเบสโดย JSP เพจจะทำการเรียกใช้ข้อมูลต่าง ๆ โดยผ่านทางจาว่าคลาสที่อยู่ในรูปของ JavaBeans ตัวอย่างของ Server Side Application ที่มีดาต้าเบสมาเกี่ยวข้องก็จะเป็นดังรูปข้างล่างนี้

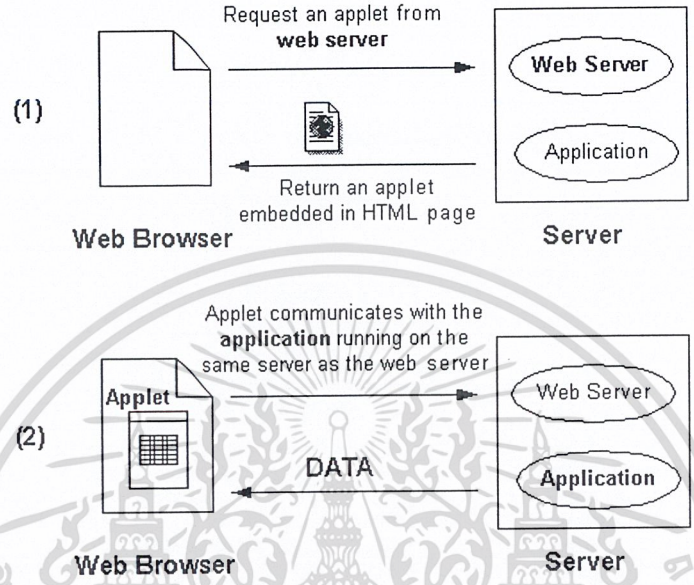


รูปที่ 7-4 Servlets(in Servlet Engine) ติดต่อกับ Relational Database ผ่าน JDBC

7.5 Applet (On Web Browser) to Application

Applet เป็นโปรแกรมที่เขียนขึ้นโดยใช้จาว่าซึ่งจริงๆ แล้วจะสามารถรันที่ไหนก็ได้ที่มี JVM (Java Visual Machine) อยู่ แต่โดยทั่วไปตัว Applet นี้มักจะถูกรันภายใต้ JVM ที่อยู่ใน เว็บบราวเซอร์ หรือไม่ก็ AppletViewer เสียมากกว่า ในช่วงแรก ๆ Applet ที่ใช้รันบนเว็บไซต์มักจะถูกเขียนขึ้นเพื่อใช้ตกแต่งเพจต่าง ๆ ให้มีความสวยงามแต่ด้วยความสามารถที่ว่า Applet สามารถสร้างการติดต่อกลับไปยังเว็บเซิร์ฟเวอร์ที่เป็นตัวเก็บ Applet ดังกล่าวได้ Applet จึงถูกใส่ความสามารถอื่น ๆ เข้าไปเกินกว่าที่จะเป็นเพียงไคลเอ็นต์ไซต์แอพพลิเคชันแต่เพียงอย่างเดียว วิธีการแรกที่นิยมใช้กันก็คือการให้ Applet ติดต่อกับแอพพลิเคชันที่รันอยู่ที่ตัวเว็บเซิร์ฟเวอร์ที่เก็บ Applet นั้นไว้โดยหลักการก็คือการใช้ เว็บเซิร์ฟเวอร์ เป็นแค่เพียงตัวเก็บ Applet เพื่อให้เว็บบราวเซอร์สามารถโหลด Applet มาจาก เว็บเซิร์ฟเวอร์ ได้โดยหลักจากที่เว็บบราวเซอร์ทำการโหลด Applet เสร็จแล้ว ตัว Applet ก็จะทำการสร้างการติดต่อไปยังแอพพลิเคชันซึ่งรันอยู่บนเซิร์ฟเวอร์เดียวกับที่รันเว็บเซิร์ฟเวอร์แทน หลักการดังกล่าวกลายเป็นจุดเริ่มต้นของคำ ๆ หนึ่งคือ Application Service Provider (ASP) ซึ่งก็หมายถึงบริษัทที่ให้บริการแก่ลูกค้าในการใช้แอพพลิเคชันต่าง ๆ ผ่านทางเว็บไซต์โดยไม่ต้องทำการติดตั้งแอพพิลเคชันนั้นลงไปที่เครื่องของลูกค้าแต่จะทำการโหลด แอพ-

พลิเคชันนั้นมาใช้เมื่อถึงเวลาที่ต้องการนั่นเอง 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

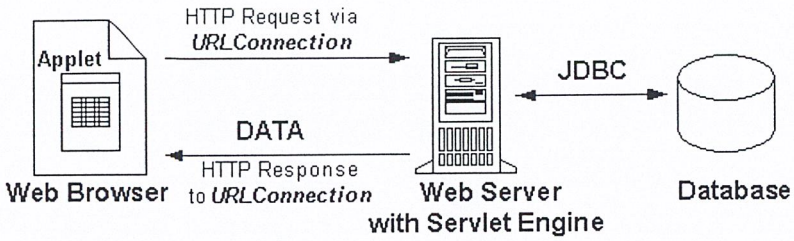


รูปที่ 7-5 Applet to Application Communication

7.6 Applet (on Web Browser) to Servlet

ในกรณีที่เว็บไซต์ต้องการความสวยงามให้กับผู้ใช้อย่างสูงจนกระทั่งการใช้ HTML และ Widget ต่าง ๆ (เช่นปุ่มหรือพวก html form) ที่ทางเว็บบราวเซอร์มีให้นั้นยังไม่ยืดหยุ่นเพียงพอต่อความต้องการของผู้ที่ออกแบบเว็บไซต์นั้นละก็ Applet มักเป็นอีกทางเลือกหนึ่งที่หลาย ๆ คนนิยมใช้เพื่อลดช่องว่างในส่วนนี้ ลักษณะของ Applet ที่ติดต่อกับ Servlet จะต่างจาก Applet ที่ใช้ติดต่อกับแอพพลิเคชันตรงที่ว่า Applet ที่ใช้ติดต่อกับ Servlet จะยังคงใช้ HTTP protocol ในการติดต่อสื่อสารกับเซิร์ฟเวอร์ โดยถ้าเปรียบเทียบก็คือแทนที่เราจะให้ผู้ใช้ใช้งาน HTML Form ต่าง ๆ ของเว็บบราวเซอร์ในการติดต่อสื่อสารกับ Servlet (ที่อยู่ในเว็บเซิร์ฟเวอร์) เราก็ใช้ Applet เป็นตัวแทนของ HTML Form ซึ่งนอกจากที่ Applet จะสามารถแทนที่ HTML Form ได้แล้วเราก็ยังสามารถเพิ่มความสวยงามเข้าไปใน Applet ได้ตามแต่จินตนาการของเราอีกด้วย

หลายคนอาจจะสงสัยว่าแล้ว Applet จะติดต่อกับ Servlet ได้อย่างไร วิธีการที่นิยมใช้กันก็คือแทนที่จะให้ เว็บบราวเซอร์สร้าง HTTP Request ไปยัง Servlet เหมือนอย่างแต่ก่อนเราก็ให้ Applet ที่รันอยู่บน เว็บบราวเซอร์สร้าง HTTP Request ติดต่อไปยัง Servlet แทนโดยวิธีการเช่นนี้เราเรียกว่า HTTP Tunnel ซึ่งสามารถทำได้ง่าย ๆ โดยให้ Applet สร้าง HTTP Connection ไปยัง Servlet ผ่านทางคลาส java.net.URLConnection



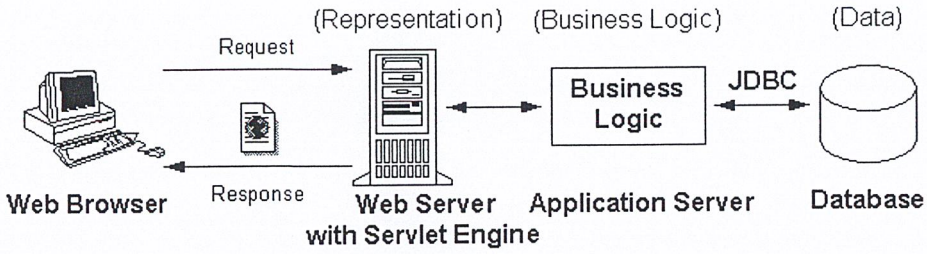
รูปที่ 7-6 Applet to Servlet ติดต่อสื่อสารผ่าน java.net.URLConnection class

เมื่อไรก็ตามที่มีการเปลี่ยนแปลงเกิดขึ้นกับ Applet ซึ่งเป็นผลมาจากกิจกรรมต่าง ๆ ของผู้ใช้ ตัว Applet จะทำการส่งค่าที่เปลี่ยนแปลงนี้ไปยัง Servlet เพื่อทำการประมวลผล โดยหลังจากที่ Applet ได้รับผลที่ส่งไปกลับมาจาก Servlet แล้ว Applet จะทำการตีความผลที่ได้เพื่อทำการเปลี่ยนแปลงตัวมันเองโดยผลที่ได้สำหรับผู้ใช้ก็คือการเปลี่ยนแปลงของค่าตัวเลขต่าง ๆ ที่อยู่บน Applet ในกรณีที่ Applet นั้นถูกใช้สำหรับแสดงค่าจำนวนตัวเลข ตลอดจนการเปลี่ยนแปลงรูปร่างหน้าตาของ Applet ไปยังสถานะ อื่น ๆ ในกรณีของ Applet ที่ใช้สำหรับปฏิทิน เป็นต้น วิธีการ Applet to Servlet นี้นิยมใช้กันมากเพราะเป็นการลดโหลดหรือการคำนวณที่หนักหน่วงจากส่วนของ ไคลเอ็นต์ (Applet) ไปยังส่วนของเซิร์ฟเวอร์ (Servlet) แทน โดยมักจะมีศัพท์คำหนึ่งที่ใช้แทนลักษณะสถาปัตยกรรมแบบนี้ซึ่งเราเรียกว่า Thin Client

7.7 แอพพลิเคชันเซิร์ฟเวอร์ (Application Server)

สำหรับเว็บไซต์เล็ก ๆ แล้วโครงสร้างของ Server Side Application ที่กล่าวถึงข้างต้นก็มักจะเพียงพอต่อความต้องการสำหรับประโยชน์ใช้สอย อย่างไรก็ตามสำหรับเว็บไซต์ไซด์ใหญ่ ๆ ที่มีผู้ใช้จำนวนมหาศาล ส่วนประกอบที่สำคัญอีกอย่างหนึ่งที่มักจะถูกเพิ่มเข้าไปมักจะเป็นส่วนที่เรียกว่า แอพพลิเคชันเซิร์ฟเวอร์

แอพพลิเคชันเซิร์ฟเวอร์เป็นส่วนประกอบอีกส่วนหนึ่งที่ถูกสร้างขึ้นมาเพื่อสนองความต้องการที่ว่าในการสร้างระบบที่ดีระบบหนึ่งขึ้นมานั้น เราควรจะแยกส่วนที่ทำหน้าที่เกี่ยวกับการบริการต่าง ๆ รวมไปถึงการจัดสรรข้อมูลออกมาเป็นอีกส่วนต่างหาก (Business Logic) ทั้งนี้เพื่อเพิ่มความคล่องตัวในการเปลี่ยนแปลงของระบบ รวมไปถึงการจัดระบบให้แบ่งออกเป็นส่วน ๆ อย่างชัดเจนมากยิ่งขึ้น โดยถ้าดูจากบทบาทของแอพพลิเคชันเซิร์ฟเวอร์ที่ผ่านมา ตั้งแต่อดีตจนถึงปัจจุบันเราอาจจะพูดได้ว่า แอพพลิเคชันเซิร์ฟเวอร์ก็คือส่วนที่ช่วยอำนวยความสะดวกให้กับ Server Side Application โดยมักจะเป็นตัวช่วยในการคำนวณข้อมูล, ช่วยเป็นตัวกลางในการจัดส่งข้อมูล, ช่วยในการควบคุมการจัดเก็บข้อมูลลงไปในดาต้าเบสอย่างถูกต้อง ตลอดจนช่วยในการเพิ่มความเร็วให้กับระบบโดยการเก็บออบเจ็กต์ต่าง ๆ ที่ใช้แล้วหรือไม่ใช้นำกลับมาใช้อีก



รูปที่ 7-7 โครงสร้างของแอพพลิเคชันเซิร์ฟเวอร์

ประโยชน์อย่างหนึ่งของ แอพพลิเคชันเซิร์ฟเวอร์ที่เรามักพบเห็นโดยทั่วไปก็คือการนำมาใช้สำหรับรัน EJB (Enterprise Java Bean) ถ้าพูดกันในเชิงนิยามแล้ว EJB ไม่ได้ถูกรันที่แอพพลิเคชันเซิร์ฟเวอร์ แต่จะถูกรันกับส่งหนึ่งที่เรียกว่า EJB Server อย่างไรก็ตาม แอพพลิเคชันเซิร์ฟเวอร์ส่วนมากมักจะสนับสนุน EJB ซึ่งเป็นส่วนหนึ่งของ J2EE ดังนั้นแอพพลิเคชันเซิร์ฟเวอร์ส่วนมากจึงมีส่วนหนึ่งที่ทำหน้าที่เป็น EJB Server เพื่อรัน EJB ต่าง ๆ ได้

บทที่ 8
การสร้างโปรแกรมประยุกต์ด้วย JSP และ JavaBeans

8.1 JSP

Java Server Page หรือที่เรียกกันง่ายๆว่า JSP เป็นเทคโนโลยีที่เป็นผลรวม ของการนำเอาหลักการสร้างเว็บแบบสแตติกด้วย HTML มารวมกับการสร้างเว็บแบบไดนามิก ก่อให้เกิดการแสดงผลเว็บ ที่มีการเปลี่ยนแปลงข้อมูลได้ตามการใช้งานของผู้ใช้ JSP เป็นเทคโนโลยีคล้ายกับ ASP (Microsoft Active Server Pages) แต่ผิดกันตรงที่หัวใจของ JSP คือ Java ซึ่งเป็นภาษาที่มีคอนเซ็ปต์หลักอยู่ที่ออบเจ็กต์ ซึ่งช่วยทำให้ง่ายต่อการพัฒนาในโปรเจ็กต์ใหญ่ ๆ ตลอดจนสามารถนำส่วนประกอบต่าง ๆ กลับมาใช้ได้อีก จุดเด่นที่สำคัญของ JSP คือสามารถทำงานได้โดยไม่ขึ้นอยู่กับผู้ผลิตซอฟต์แวร์รายใดรายหนึ่งโดยเฉพาะ ซึ่งโดยทั่วไปเทคโนโลยีต่าง ๆ มักจะออกมาในลักษณะของผลิตภัณฑ์จากบริษัทผู้ผลิตแห่งใดแห่งหนึ่ง แต่ JSP ใช้ลักษณะของ Specification ซึ่งกำหนดโดย Sun Microsystems ดังนั้นผู้ผลิตซอฟต์แวร์จึงสามารถอ้างอิง Specification ที่กำหนดขึ้นได้

8.2 เปรียบเทียบความแตกต่างของ JSP เมื่อเทียบกับเทคโนโลยีอื่นๆ

JSP ไม่ได้เป็นเทคโนโลยีเดียว ในปัจจุบันที่สามารถทำให้เว็บแสดงข้อมูลในแบบไดนามิกได้ โดยมีเทคโนโลยีอื่น ๆ ดังต่อไปนี้

8.2.1 Active Server Page(ASP) ASP เป็นเทคโนโลยีที่เหมือนกับ JSP แต่เป็นเทคโนโลยีที่พัฒนามาจาก บริษัทไมโครซอฟต์ ผู้ผลิตระบบปฏิบัติการวินโดว์ที่รู้จักกัน โดยทั่วไป ลักษณะ JSP มีรูปแบบที่แตกต่างกันเด่นชัด 2 ประการคือ

- ประการแรก JSP สามารถสร้างได้จาก ภาษาจาวา ต่างจากภาษา Visual Basic หรือภาษาใดๆ ซึ่งเป็นข้อกำหนดเฉพาะของ ไมโครซอฟต์
- ประการที่สอง JSP ซึ่งเป็นผลพ่วงมาจากประการแรกคือ สามารถโยกย้ายการทำงาน ไปใช้งานระบบปฏิบัติการใดๆก็ได้ ทีมีการใช้งานกันอยู่ในปัจจุบัน โดยไม่ต้องแก้ไขหรือเปลี่ยนแปลงใดๆ ในขณะที่ ASP ก็โยกย้ายได้เช่นเดียวกัน แต่ต้องอยู่ในระบบปฏิบัติการที่ ไมโครซอฟต์กำหนดขึ้นเท่านั้น

8.2.2 Servlet JSP สามารถทำงานได้เช่นเดียวกับที่ Sevlet ทำได้ แต่มีจุดเด่นมากกว่าที่ JSP มีความสะดวกในการสร้าง และเปลี่ยนแปลง มากกว่า เพราะทำโดยตรงที่ไฟล์ HTML มากกว่าการที่ต้องลงมือเขียนคำสั่งภาษาจาวา โดยตรงเหมือนกันสร้าง Servlet ซึ่งต้องมีการนำไปคอมไพล์ ก่อนการนำไปใช้งาน

8.2.3 **JavaScript** เป็นการทำเนื้อหาไดนามิกให้กับเว็บ แต่รูปแบบการทำงานเกิดขึ้นของ JavaScript เกิดจากการประมวลผลและดึงข้อมูล ที่มีอยู่บนเครื่องผู้ใช้หรือไคลเอ็นต์ เท่านั้น มารวบรวมเนื้อหาในเว็บ ต่างกับ JSP ที่เป็นการทำไดนามิก แต่ข้อมูลถูกสร้างและดึงมาจากระบบเว็บเซิร์ฟเวอร์ ที่มีหลากหลายมากกว่า และทำให้เนื้อข้อมูลเป็นเนื้อเดียวกัน เมื่อผู้ใช้เรียกดู แต่ JavaScript เนื้อข้อมูลเป็นของเครื่องผู้ใช้เอง ซึ่งผู้ใช้ต่างคน(ต่างเครื่อง) ก็จะให้ข้อมูลที่คล้ายกัน คือไม่เหมือนกันทั้งหมด

8.2.4 **Static HTML** หรือสแตติก HTML แน่นอนข้อนี้ คงเห็นได้ชัดเจนที่ว่า สแตติก HTML ให้เนื้อข้อมูลที่คงที่ตลอดเวลา ไม่ว่าจะถูกเรียกดูในเวลาใดๆ (ยกเว้นมีคนมาแก้ไขคำสั่ง HTML) ในขณะที่ JSP ทำให้เนื้อข้อมูลมีการเปลี่ยนแปลงตามการใช้งาน หรือเครื่องผู้ใช้โดยไม่ต้องเปลี่ยนแปลงแก้ไขคำสั่ง JSP

	CGI/Perl	Mod_Perl	ASP	JSP
เว็บเซิร์ฟเวอร์	Any Web server	Apache Web Server	Microsoft IIS or Personal Web Server	Any Web server, including Apache, Netscape, IIS today
โยกย้ายการทำงาน	No	No	No	Yes
ลักษณะนำกลับมาใช้ (Reusable, modular code)	No	No	No	Yes
สคริปต์หรือภาษา	C, Perl	Perl	VBScript, JScript	Java
ปกป้องการใช้งานหน่วยความจำ	Yes	No	No	Yes
สนับสนุนการทำงานร่วมในหลายโปรเซส	No	Yes	Yes	Yes

ข้อมูลจาก เว็บ http://www.javasoft.com/products/jsp/jspguide-wp.html

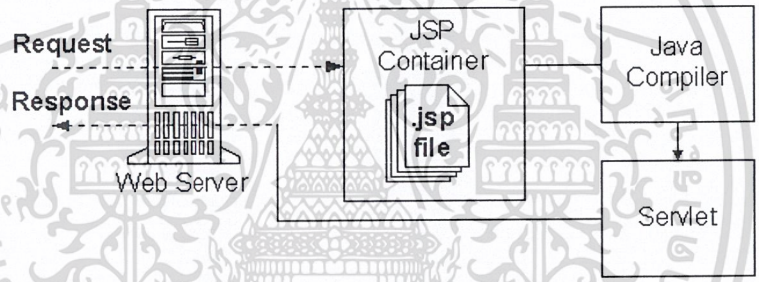
ตารางที่ 8-1 เปรียบเทียบความแตกต่างของ JSP เมื่อเทียบกับเทคโนโลยีอื่นๆ

8.3 JSP Containers

JSP page (ไฟล์ที่เขียนขึ้นโดยใช้ JSP script และลงท้ายด้วย .jsp) จะถูกรันโดย JSP Container ซึ่งมักจะเป็นส่วนประกอบที่อยู่ในเว็บเซิร์ฟเวอร์หรือเป็นตัวแอทออนใน Application Server โดยทั่วไป JSP Container จะเป็นตัวรับ request จากไคลเอ็นต์ส่งผ่านไปยัง JSP page และส่งค่าที่ได้จากการประมวลผลโดย JSP page กลับไปยังไคลเอ็นต์

ถ้าดูเผิน ๆ จากทางเว็บบราวเซอร์การเรียกไฟล์ JSP คงดูคล้าย ๆ กับการเรียกไฟล์ HTML ธรรมดา ๆ แต่ความจริงแล้ว หลังจาก request จากไคลเอ็นต์ส่งมาถึงเว็บเซิร์ฟเวอร์, JSP Container จะทำการแปลงไฟล์ JSP ให้กลายเป็น Servlet source ไฟล์ (ในกรณีที่ JSP ไฟล์ดังกล่าวถูกเรียกเป็นครั้งแรก) ซึ่งไฟล์ Servlet source ที่ได้จะถูกคอมไพล์เป็น .class เพื่อใช้ในการประมวลผล request ของไคลเอ็นต์แล้วส่งกลับไปให้เว็บเซิร์ฟเวอร์ในรูปของ OutputStream ซึ่งถูกส่งไปที่ไคลเอ็นต์ในท้ายสุด

หลังจากนั้น ถ้า JSP ไฟล์ดังกล่าวถูกเรียกอีก การแปลงไฟล์หรือคอมไพล์จะไม่เกิดขึ้น เพราะ JSP Container จะใช้ไฟล์ .class ที่เก็บไว้แล้วประมวลผลแทน ข้อสังเกตคือ การเรียกไฟล์ JSP ครั้งแรกจะรู้สึกว่าช้า แต่ครั้งถัดไป ๆ จะเห็นได้ว่าเร็วขึ้นมาก เพราะได้มีการลดขั้นตอนต่าง ๆ ที่เสียเวลาไป



รูปที่ 8-1 การแปลงและคอมไพล์ JSP ไฟล์

จากข้อความที่อธิบายข้างต้น เราสามารถแบ่งระยะเวลาของไฟล์ JSP หนึ่ง ๆ ออกเป็นสองช่วงคือ

1) Translation Time คือช่วงเวลาที่ JSP ไฟล์ถูกแปลงให้กลายเป็น Servlet ไฟล์และถูกคอมไพล์ให้กลายเป็น .class ไฟล์ ซึ่งจะเกิดก่อนการรับ request จากไคลเอ็นต์ครั้งแรก

2) Client Request Time คือช่วงเวลาที่ .class ของ JSP ไฟล์ทำการรับ request จากแต่ละไคลเอ็นต์แล้วทำการประมวลผล

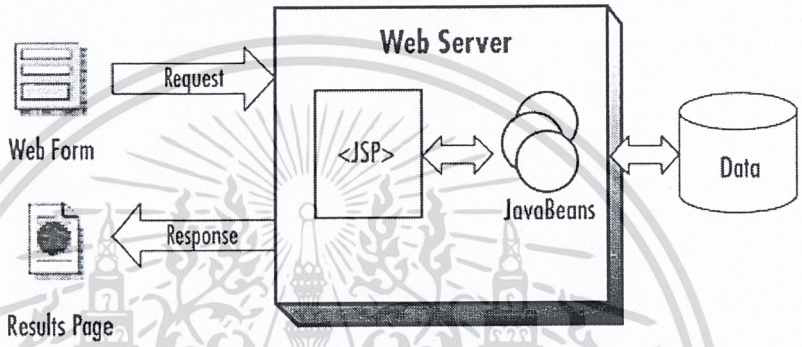
8.4 โมเดลของ JSP

JSP นับเป็นแนวทางออกที่ดีสำหรับการพัฒนาและออกแบบเว็บไซต์โดยมี สถาปัตยกรรมสำหรับการสร้าง Sever-Side Web Application โดยมีอยู่กัน 2 แบบดังนี้คือ

- Model 1 JSP Architecture จะเป็นการโปรแกรมด้วย JSP
- Model 2 JSP Architecture จะเป็นการโปรแกรม Servlet ร่วมกับ JSP

ทั้งสองโมเดลนี้จะมีความสอดคล้องกับการทำงานของกับ Application Server โดยจัดให้ JSP ซึ่งเป็นส่วนของ Representation และ JavaBeans ซึ่งเป็นส่วนของ Business Logic

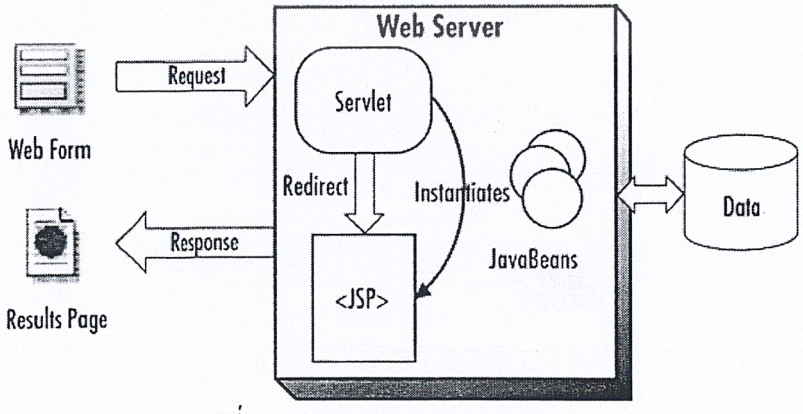
8.4.1 Model 1 JSP Architecture



รูปที่ 8-2 Model 1 JSP Architecture

โมเดลนี้จะจัดให้ JSP ซึ่งทำหน้าที่เป็น Representation กับ JavaBeans ซึ่งเป็น Business Logic ทำงานร่วมกันเพื่อให้ง่ายต่อการ โปรแกรม การดีบัก และการทดสอบ เมื่อมีการแก้ไขส่วน Representation ก็จะไม่กระทบต่อส่วน Business Logic โดยการที่จะไปหน้าถัดไปนั้นต้องทำการตัดสินใจจากหน้าปัจจุบัน ซึ่งการทำแบบนี้เหมาะสำหรับระบบขนาดเล็ก ๆ ข้อเสียของโมเดลนี้คือใช้กับระบบที่มีความซับซ้อนมากไม่ได้

8.4.2 Model 2 JSP Architecture



รูปที่ 8-3 Model 2 JSP Architecture

ส่วนโมเดลสองนี้จะมี Servlet ทำหน้าที่เป็นตัวคอยรับ request จากไคลเอ็นต์ โดยจะทำหน้าที่คอยควบคุมการทำงานในการส่งผลลัพธ์ให้กับ JSP เพื่อนำไปแสดงผลโดย JSP นั้นสามารถเรียกใช้ข้อมูลจาก JavaBeans ได้ ซึ่งโมเดลนี้จะเป็น Model แบบ MVC (Model View Controller) Design Pattern การที่จะไปหน้าถัดไปนั้นสำหรับโมเดลนี้ เหมาะกับงานที่ต้องใช้หลายเงื่อนไขในการตัดสินใจว่าหน้าไหนจะเป็นหน้าถัดไป

โมเดลนี้มีข้อเสียคือจะมี Servlet เพียงตัวเดียวเป็นตัวกลางควบคุมการทำงานของระบบเช่นการ Login,การให้สิทธิ,การติดต่อกับฐานข้อมูล ถ้าหากระบบมีขนาดใหญ่ขึ้นย่อมเป็นการยากต่อการโปรแกรม ซึ่งปัญหาดังกล่าวควรแก้โดย ควรมี Sevlet หลายตัวที่รองรับฟังก์ชันการทำงานทางธุรกิจที่ต่างกันออกไป และมี Master Servlet เป็นตัวควบคุมอีกทีหนึ่ง

ระบบหนึ่งระบบนั้นเราสามารถรวมสองโมเดลนี้ทำงานเข้าด้วยกันได้ โดยแต่ละโมเดลจะทำแต่ละฟังก์ชันของระบบก็ได้ ในโครงงานระบบทำสัญญาซื้อขายบ้านจัดสรรนี้จะใช้โมเดลแบบผสมเพื่อ ให้เหมาะกับการทำงานของแต่หละฟังก์ชัน และจะใช้ OAS เป็นเว็บเซิร์ฟเวอร์ซึ่งสนับสนุน Servlet Engine และ JSP Containers

8.5 JSP Standard Syntax and Semantics

JSP Directives and Scripting Elements	
Directives	<%@ directive %>
Declarations	<%! Declaration %>
Expressions	<%= expression %>
Code Fragment/Scriptlet	<% code fragment %>
Comments	<%-- comment --%>

ตารางที่ 8-2 JSP Standard Syntax and Semantics

8.5.1 Directives

JSP Directives เป็นส่วนที่เรียกว่า "message" ที่ใช้ส่งไปถึง JSP Container ในลักษณะของ <%@ ... %> โดย message นี้จะเป็นตัวบอกว่า Container ควรจะทำอะไรกับ JSP page ต่อไป ซึ่ง Directives ในส่วนนี้จะไม่มีผลต่อหน้าตาที่ออกมาของ JSP page เมื่อถูกแปลงออกมาในรูปของ HTML แล้ว Directives แบ่งออกเป็นสามส่วน ส่วนที่เราใช้ทั่ว ๆ ไปคือ page และ include กับส่วนที่สามซึ่งเป็น taglib (เราจะไม่กล่าวถึงในที่นี้) ซึ่งใช้สำหรับสร้าง tag ของเราเอง

- JSP page directive เป็นส่วนที่ปกติเราจะเห็นอยู่บนสุดของ JSP page ซึ่งใช้ในการกำหนดค่าต่าง ๆ ที่เกี่ยวข้องกับ JSP page นั้น ๆ หรือเกี่ยวข้องกับการติดต่อสื่อสารกับ JSP Container ยกตัวอย่างเช่น ถ้าเราอยากเรียกใช้คลาสที่ชื่อ java.util.Date ก็สามารถที่จะ import คลาสนี้ได้โดยใช้

```
<%@ page import="java.util.Date" %>
```

หรือเวลาที่เกิดข้อผิดพลาดใน JSP page ของเรา แล้วเราอยากให้โชว์ข้อผิดพลาดที่เกิดขึ้นที่ error page ที่ชื่อ errorPage.jsp เราก็ใช้

<%@ page errorPage="errorPage.jsp" %>

หรือแม้กระทั่งเวลาที่เราอยากจะเก็บสถานะของ ผู้ใช้ ในรูปของ session เราก็สามารถใช้

<%@ page session="true" %>

เพื่อบอก Container ว่า JSP page นี้สามารถเรียกใช้ HttpSession ในการเปลี่ยนแปลงและเก็บสถานะของผู้ใช้ได้ เป็นต้น

- JSP include directive เป็นส่วนที่ช่วยให้เราสามารถนำไฟล์ JSP อื่น ๆ มาเป็นส่วนประกอบของไฟล์ JSP ปัจจุบัน โดยใช้

<%@ include file="filename.jsp" %>

ประโยชน์ที่เห็นได้ชัดอย่างหนึ่งของการใช้ include directive คือ การง่ายต่อการเปลี่ยนแปลงและบำรุงรักษา ยกตัวอย่างเช่น สมมุติว่าทุกหน้าของเราต้องการ header หรือ footer ที่เหมือนกัน เราก็แค่ใช้ <%@ include file="header.jsp" %> ใส่ลงไปในทุก ๆ JSP ไฟล์ หลังจากนั้น ทุกครั้งที่เราต้องการเปลี่ยนแปลง header นี้ เราก็แค่เปลี่ยนไฟล์ที่ชื่อ header.jsp ซึ่งไฟล์ทุกไฟล์ที่ include ไฟล์นี้เข้าไป ก็จะเปลี่ยนไปด้วยโดยอัตโนมัติ

8.5.2 Declarations

Declarations ใช้ในการประกาศค่าตัวแปร (variable) หรือสร้างฟังก์ชั่นต่าง ๆ (method) เพื่อจะใช้ใน JSP page นั้น Declarations เป็นส่วนที่ถูกประมวลผลในช่วง Translation time

โดยทั่วไป Declarations จะถูก initialize เมื่อ JSP page ถูก initialize ซึ่งจะทำให้ตัวแปร หรือฟังก์ชันใน Declarations พร้อมใช้งานได้ทันที ตัวอย่างเช่น

```
<%! int i = 0; %>
<%! public String f(int i) {
        if (i<3) return "...";
        ...
} %>
```

8.5.3 Expressions

โดยการใช้ Expressions สิ่งต่าง ๆ ที่อยู่ใน <%= ... %> จะถูกประมวลผลแล้วเปลี่ยนให้อยู่ในรูปของ String และส่วนที่ได้นี้จะถูกรวมเข้าไปอยู่ใน output page (HTML) โดยตรง Expressions เป็นส่วนที่ถูกประมวลผลในช่วง Translation time เช่น

<%= i %>

JSP Container จะดึงค่า i ออกมา แล้วเปลี่ยนเป็นสตริงโดยใช้ Integer.toString(i)

```
<%= "Hello" %>
```

JSP Container จะนำค่า Hello ใส่เข้าไปใน output page

ข้อควรจำอย่างหนึ่งคือ ห้ามทำการใส่ semicolon (;) ลงไปใน <%= ... %> ยกเว้นถ้า semicolon นั้นเป็นส่วนหนึ่งของสตริง เช่น

```
<%= "Hello semicolon ;" %>
```

8.5.4 Code Fragments/Scriptlets

เราสามารถใส่โค้ด หรือส่วนหนึ่งของโค้ดเข้าไปยัง JSP page ได้โดยใช้ <% ... %> โค้ดที่ใส่เข้าไปนี้จะไปอยู่ในส่วนของ service() ฟังก์ชันของ Servlet ซึ่งจะถูกเรียกใช้เมื่อมี request จาก ไคลเอ็นต์ยก ตัวอย่างเช่น

```
<% int userId = request.getParameter("userId"); %>
```

8.5.5 Comments

comments ใน JSP มีอยู่สองแบบคือ

1) comment ซึ่งปกติถูกใช้ใน HTML ไฟล์นั้นจะไม่ถูกละเว้นโดย JSP Container เมื่อไฟล์ JSP ถูกแปลงเป็น .class แล้ว (จะยังคงปรากฏที่ client side ในส่วนของ HTML source code) ตัวอย่างเช่น

```
<!-- This is my comment -->
```

บางทีเราอาจเปลี่ยนแปลง comment ที่อยู่ข้างใน <!-- ... --> ได้ โดยใช้ expression เช่น

```
<!-- <%= expression %>  more comment -->
```

2) comments ที่ผู้เขียน JSP ใช้สำหรับ comment ไฟล์ JSP ที่เขียนอยู่ซึ่งจะถูกละเว้นโดย JSP Container ในช่วงของ Translation time ตัวอย่างเช่น

```
<%-- comment for server side only --%>
```

8.6 JSP Implicit Objects

เมื่อไรก็ตามที่เราต้องการเรียกใช้คลาสบางคลาส เราจะต้องทำการ import โดยใช้ page direction แล้วทำการเรียกใน Scriptlet ยกตัวอย่างเช่น

```
<%@ page import="java.util.Date" %>
<% Date rightNow = new Date() %>
```

บางครั้งในการเขียน JSP จะมีคลาสบางคลาสที่เราใช้บ่อย เช่น HttpSession หรือคลาสบางคลาสที่เราไม่สามารถเรียกใช้ได้โดยตรงจากการ import เช่น HttpServletRequest,HttpServletResponse ดังนั้น JSP จึงช่วยอำนวยความสะดวกโดยให้เราสามารถเรียกออบเจ็กต์ของคลาสดังกล่าวออกมาใช้ได้เลย โดยไม่ต้องมีการประกาศตัวแปรก่อน (ใช้ได้ใน Scriptlet และ Expression) คลาสพวกนี้เราเรียกว่า predifined variables หรือ Implicit Objects

ใน JSP 1.1 Specification มี Implicit Objects ที่เราสามารถเรียกใช้ได้คือ

Implicit Objects	
Request	Javax.servlet.ServletRequest คือ request ออบเจ็กต์ที่เก็บรายละเอียดต่าง ๆ ที่ถูกส่งมาจากไคลเอ็นต์ซึ่งรวมไปถึง parameters ที่มาจาก GET/POST ในกรณีของ Http ด้วย
Response	Javax.servlet.ServletResponse คือ ออบเจ็กต์ที่ใช้ส่งผลกลับไปยังไคลเอ็นต์
PageContext	Javax.servlet.jsp.PageContext คุณสมบัติต่าง ๆ ของ JSP page จะถูกเปลี่ยนแปลงได้โดยใช้ออบเจ็กต์นี้
Session	Javax.servlet.http.HttpSession คือ session ออบเจ็กต์ของไคลเอ็นต์ที่ส่ง request มา
Application	Javax.servlet.ServletContext คือ ออบเจ็กต์ที่เก็บรายละเอียดที่เกี่ยวกับ environment ที่ JSP page อยู่
Out	Javax.servlet.jsp.JspWriter คือ ออบเจ็กต์ OutputStream ที่ JSP page ใช้ส่งผลกลับไปยัง ไคลเอ็นต์(จะเชื่อมต่อเข้ากับ response ออบเจ็กต์อีกทีหนึ่ง) ซึ่งอาจอยู่ในรูปของ PrintWriter หรือ JspWriter
Config	Javax.servlet.ServletConfig คือ ออบเจ็กต์ที่เก็บค่า setting ต่าง ๆ ของ JSP page
Page	คือ ตัว JSP page ออบเจ็กต์ ซึ่งก็คือ this คีย์เวิร์ดในจาว่านั่นเอง
Exception	Java.lang.Throwable คือ uncaught exception ออบเจ็กต์ที่ใช้กับ error page ที่ถูกใส่ไว้ใน page directive เช่น <@page errorPage="myErrorPage.jsp" %>

ตารางที่ 8-3 Implicit Objects

1) request คือ ออบเจ็กต์ที่เป็นอินสแทนซ์ของ ServletRequest ซึ่งถ้าใช้ใน Http ก็จะเป็น HttpServletRequest เราสามารถใช้ request ในการอ่านค่า parameters ต่าง ๆ ที่ส่งมาโดย GET หรือ POST จาก ไคลเอ็นต์โดยการใช้ request.getParameter(...) หรืออ่านค่าอื่น ๆ เช่น Header โดยใช้ API ที่มีอยู่ใน HttpServletRequest.

2) response คือ ServletResponse ที่ใช้ในการส่งค่าต่าง ๆ กลับไปที่ไคลเอ็นต์โดยผ่านทาง OutputStream ซึ่งอยู่ในรูปของPrintWriter หรือ JspWriter

3) pageContext คือ ออบเจ็กต์ที่เก็บฟังก์ชันที่ใช้ในการเรียกคุณสมบัติต่าง ๆ ของ JSP Container ซึ่งจะไม่เหมือนกันในแต่ละตัวมาเชื่อมกับ JSP ไฟล์ เพื่อเพิ่มประสิทธิภาพในการทำงาน

4) session คือ HttpSession ของไคลเอ็นต์ที่ส่ง request มาซึ่งจริง ๆ แล้วเราสามารถเรียกออบเจ็กต์นี้มา โดยผ่านทาง request.getSession(...) ก็ได้

5) applicaion คือ ServletContext ที่เก็บค่าต่าง ๆ ของ environment ที่ไฟล์ JSP นี้อยู่

6) out คือ PrintWriter หรือ JspWriter (Buffered version of PrintWriter) ที่ใช้ในการพิมพ์ text output ออกไปยังไคลเอ็นต์โดยผ่านทาง response อีกทีหนึ่ง

7) config คือ ServletConfig ที่เก็บค่าที่ setting ต่าง ๆ สำหรับไฟล์ JSP

8) page คือ *this* ที่ใช้แทนตัวของไฟล์ JSP เอง

9) exception คือ Throwable interface ที่ใช้ส่งค่า error ต่าง ๆ ไปยัง error page

สำหรับรายละเอียดในการใช้ implicit object แต่ละตัว ให้หาอ่านเพิ่มเติมจาก API ของ ออบเจ็กต์นั้น ๆ

8.7 JSP Actions

JSP Actions สามารถใช้ในการควบคุมและเปลี่ยนแปลงการทำงานของไฟล์ JSP ที่เราเขียน ซึ่งอาจจะเป็นในลักษณะของ

1) insert ไฟล์ต่าง ๆเข้ามาเป็นส่วนหนึ่งของไฟล์ JSP ปัจจุบัน โดยใช้

```
<jsp:include page="exfile.jsp" .../>
```

2) ใช้สำหรับควบคุมการทำงานของ JavaBeans ที่เป็นส่วนประกอบหนึ่งของไฟล์ JSP นั้นโดยใช้

```
<jsp:useBean .../>, <jsp:setProperty .../>, <jsp:getProperty .../>
```

3) ใช้ในการส่งผ่าน (forward) request ของ ไคลเอ็นต์ไปยังไฟล์ JSP อื่น ๆ โดยใช้

```
<jsp:forward .../>
```

4) ใช้ <jsp:plugin type=... /> tag เพื่อทำการสร้างออบเจ็กต์หรือ EMBED code ที่ใช้สำหรับเรียก (หรือโหลด) Java plugin เพื่อทำการรัน Applet หรือ JavaBeans ในตัว เว็บบราวเซอร์ที่กำลังเปิดไฟล์ JSP ดังกล่าวอยู่

8.7.1 <jsp:include> Action

อย่างที่กล่าวมาข้างต้น action นี้ ถูกใช้สำหรับ insert ไฟล์ต่าง ๆ เข้ามาที่ไฟล์ JSP ปัจจุบัน โดยใช้ <jsp:include page="relativeURL" flush="true"/> relativeURL คือ ที่อยู่ของไฟล์ที่เราจะทำการ insert ซึ่งจะต้องอยู่ในรูปของ relative URL กับไฟล์ JSP ปัจจุบัน flush คือ ตัวที่บอก JSP Container ให้ทำการ flush สิ่งต่าง ๆ ที่อยู่ใน buffer ของ page output ออกไปที่ OutputStream ก่อนที่จะทำการ insert ไฟล์นี้เข้าไป โดยใช้ค่า flush="true"

ข้อแตกต่างของ <%@ include file="relativeURL" %> กับ <jsp:include page="relativeURL"/> คือ

- <%@ **include ...%>** ไฟล์ relativeURL จะถูกใส่เข้าไปเป็นส่วนหนึ่งของไฟล์ JSP ที่มี include directive นี้อยู่ ในช่วงของการแปลงไฟล์ JSP ให้กลายเป็น Servlet ข้อดีก็คือจะเร็ว แต่ข้อเสียคือไฟล์ relativeURL ที่ถูก insert เข้าไปอาจจะไม่อัพเดต ซึ่งจะเกิดขึ้นในกรณีที่ไฟล์ relativeURL ดังกล่าวถูกเปลี่ยนแปลงไป แต่เกิดขึ้นหลังจากที่ไฟล์ JSP ที่บรรจุไฟล์ relativeURL นี้อยู่ถูกแปลงเป็น Servlet แล้ว
- **<jsp:include .../>** ตัว JSP Container จะทำการ insert ไฟล์ relativeURL นี้เข้าไปในไฟล์ JSP ทุก ๆ ครั้ง ที่ไฟล์ JSP ดังกล่าวถูก request โดยไคลเอ็นต์ซึ่งข้อดีก็คือ เราจะได้ไฟล์ relativeURL ที่เป็นเวอร์ชั่นอัพเดตอยู่เสมอ แต่ข้อเสียคือจะเสียเวลาในการโหลดและ insert ไฟล์ relativeURL นี้เข้ามาอยู่ด้วยทุกครั้ง

ประโยชน์ของ <jsp:include .../> ที่ใช้กันโดยทั่วไป ก็เช่นการโหลดหรือซ่อนส่วนใดส่วนหนึ่งของเพจโดยใช้แอตทริบิวต์ ของผู้ใช้ที่ Login เข้ามาในระบบเป็นตัวตัดสิน ยกตัวอย่างเช่น

```
<HTML>
<BODY> ...
<%
int userRole = userInfo.getUserRoleFromDB();
if (userRole == UserType.ADMIN) {
%>
<jsp:include page="jsp/AdminMenu.jsp" flush="true"/>
<% } else { %>
<jsp:include page="template/UserMenu.html" flush="true"/>
<% } %>
... </BODY>
</HTML>
```

8.7.2 JavaBeans

ในการเขียน JSP สำหรับระบบใหญ่ ๆ นักพัฒนาจะไม่นิยมเขียนจาวาโค้ดลงไปในไฟล์ JSP มากนัก สิ่งที่อยู่ในไฟล์มักจะเป็นเพียง HTML code และค่าของ variables ต่าง ๆ ที่ได้มาจาก JavaBeans เท่านั้น ข้อดีของการใช้ JavaBeans คือการง่ายต่อการเปลี่ยนแปลงลักษณะของไฟล์ JSP ที่เป็นเช่นนี้เพราะส่วนที่เป็นดาต้า (variables) กับส่วนที่เป็น Representation (HTML) จะอยู่ด้วยกันอย่างหลวม ๆ ดังนี้เราจึงสามารถเปลี่ยนตัว Representation เมื่อไหร่ก็ได้ โดยตัว JavaBeans ที่ใช้เก็บค่าของ variables ต่าง ๆ ยังคงเหมือนเดิม

สำหรับ JavaBeans จะมีลักษณะคล้ายกล่องซึ่งมีส่วนที่ใช้ติดต่อกับโลกภายนอกอยู่สองส่วน คือ

1. ส่วนที่ใช้สำหรับรับข้อมูลต่าง ๆ โดยข้อมูลเหล่านี้จะถูกใช้เป็นตัวแปรในการควบคุมหรือเปลี่ยนแปลงคุณสมบัติและการทำงานของกล่อง (Setter)
2. ส่วนที่ใช้สำหรับอ่านคุณสมบัติของกล่อง (Getter) ซึ่งโดยส่วนมาก(แต่ไม่ทั้งหมด) ก็คือค่าของตัวแปรต่าง ๆ ที่บรรจุอยู่ในกล่อง

ในการใช้งานกล่อง JavaBeans ดังกล่าวร่วมกับ JSP เราสามารถที่จะสร้างกล่องขึ้นมาโดยใช้ <jsp:useBean .../>, ส่งค่าเข้าไปในกล่องโดยใช้ <jsp:setProperty .../> และอ่านค่าต่าง ๆ ที่อยู่ในกล่อง โดยใช้ <jsp:getProperty .../>

8.7.2.1 Bean Conventions

สำหรับ application ที่เกี่ยวกับ JavaServlet แต่ละคลาสที่เราเขียนขึ้น อาจจะต้องมีการ extend คลาสใดคลาสหนึ่งก่อน เช่น HttpServlet (ในกรณีของการเขียน JavaServlet) แต่สำหรับการเขียนคลาสให้กลายเป็น JavaBeans แล้ว เราไม่จำเป็นต้องทำสิ่งต่าง ๆ ที่กล่าวมาข้างต้นเลย JavaBeans จริงๆ แล้วก็คือจาว่าคลาสธรรมดา ๆ แต่ถูกเขียนขึ้นโดยใช้ Naming และออกแบบ Conventions ที่อยู่ใน JavaBeans Specification เท่านั้น Conventions หลัก ๆ ที่เราจะพูดสำหรับการเขียน JavaBeans เพื่อใช้ใน JSP มีดังต่อไปนี้คือ

1) *The Bean Constructor* ในการเขียน Bean ให้สามารถใช้ได้กับ <jsp:useBean .../> สิ่งหนึ่งที่ Bean ของเราจะต้องมีคือ ตัว constructor ที่ไม่ต้องการอาร์กิวเมนต์สำหรับการ initialize ของ Bean (* JavaBeans และ Bean ในที่นี้จะหมายถึงสิ่งเดียวกัน) ยกตัวอย่างเช่น

```
public class TimeBean {
private int hours;
private int minutes;
public TimeBean() {
                java.util.Date rightNow = new java.util.Date();
                hours = rightNow.getHours();
                minutes = rightNow.getMinutes();
        }
}
```

คลาส TimeBean เขียนถูกต้องตาม Bean Conventions ทุกอย่าง คือมี constructor ที่ไม่ต้องการอาร์กิวเมนต์สำหรับการ initialize ดังนั้นเราสามารถเรียกคลาสนี้ว่า JavaBeans

2) *Bean's Properties* Bean's Properties คือตัวที่บอกว่า เราสามารถทำอะไรกับ JavaBeans นั้นได้บ้าง ซึ่งผลที่ได้ก็คือการควบคุมและเปลี่ยนแปลงคุณสมบัติต่าง ๆ ของ Bean อย่างไรก็ตามบางที properties ที่เรากำหนดขึ้น อาจจะไม่ได้ทำอะไรที่เกี่ยวข้องกับ instance variables ที่อยู่ข้างใน Bean นั้นเลย

JSP Container จะสามารถทราบว่า JavaBeans ของเรามี properties อะไรบ้าง ได้จากวิธีการที่เรียกว่า *introspection* หลักการก็คือ JSP Container จะทำการวิเคราะห์ฟังก์ชันที่เป็น public ของ Bean นั้น ซึ่งจะต้องเขียนขึ้นอย่างถูกต้องตามหลักของ JavaBeans API โดยจะถือว่า property ใด ๆ ก็ตามจะเกิดขึ้นได้ก็ต่อเมื่อ Bean คลาสนั้นมี

1. public ฟังก์ชันที่ใช้สำหรับควบคุมหรือเปลี่ยนแปลงค่าของ property นั้น (Setter)
2. public ฟังก์ชันที่ใช้สำหรับอ่านค่าของ property นั้น (Getter)
3. มีทั้งสองฟังก์ชันข้างต้น

การกำหนด property ขึ้นมาอันหนึ่ง เราจะต้องสร้าง public ฟังก์ชันที่ขึ้นต้นด้วยคำว่า set (Setter) ซึ่งใช้สำหรับควบคุมหรือเปลี่ยนแปลงค่าของ property นั้น และหรือ public ฟังก์ชันที่ขึ้นต้นด้วยคำว่า get (Getter) ซึ่งใช้สำหรับอ่านค่าของ property นั้น โดยรูปแบบโดยทั่วไปก็คือ

```
public void setPropertyName(PropertyType value);
public PropertyType getPropertyName();
```

ย้อนกลับมาที่ TimeBean โดยเพิ่มส่วนที่เป็น Setter และ Getter เข้าไป เราจะได้

```
public class TimeBean {
private int hours;
private int minutes;
public TimeBean() {
        java.util.Date rightNow = new java.util.Date();
        hours = rightNow.getHours();
        minutes = rightNow.getMinutes();
}
public int getHours() { return hours; }
public void setHours(int hours) { this.hours = hours; }
}
```

ใน JSP ไฟล์ เราสามารถเรียกใช้ TimeBean ได้ดังตัวอย่างข้างล่าง

```
<jsp:useBean id="time" class="TimeBean" scope="request"/>
<HTML>
<BODY>
Time is <jsp:getProperty name="time" property="hours"/>
...
```

8.7.2.2 Property Name Conventions

เราจะสังเกตเห็นว่า ชื่อของ properties และตัว setter, getter ของ properties เหล่านั้นจะต่างกันที่ตัวแรก หลักจากคำว่า set หรือ get เท่านั้น ในการตีความ setter, getter เพื่อหา properties ต่าง ๆ ของ JavaBeans โดย JSP Container, JSP Container จะทำการแปลง case ต่าง ๆ ให้เข้ากับชื่อของ properties ที่อยู่ใน <jsp:getProperties .../>, <jsp:setProperty .../> โดยอัตโนมัติ (สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับ JavaBeans สามารถหาอ่านได้จาก http://java.sun.com/beans/)

8.7.2.3 Loading a Bean - <jsp:useBean>

ก่อนที่เราจะสามารถใช้ JavaBeans ใน JSP ได้ สิ่งแรกที่เราต้องทำคือ การโหลดอินสแทนซ์ของ Bean เราสามารถโหลดอินสแทนซ์ของ Bean เข้ามาในระบบได้โดยใช้ <jsp:useBean .../> โดยเราจะต้อง

1. กำหนดชื่อที่จะใช้ในการอ้างถึงอินสแทนซ์ของ Bean นี้ (id)
2. กำหนดคลาสของ Bean ที่จะโหลด (class)
3. ทำการเซต scope (หรือช่วงชีวิต) ของ Bean ว่าจะยาวนานแค่ไหน (scope)

ดังรูปแบบข้างล่าง

```
<jsp:useBean id="myBeanInstance" class="com.myPackage.BeanClass"
scope="page|request|session|application">
...body...
</jsp:useBean>
```

เพราะว่า <jsp:useBean> เขียนในรูปแบบของ XML(Extensible Markup Language) ดังนั้นจึงต้องมีตัวปิด tag </jsp:useBean> จากตัวอย่างจะเห็นว่าใน <jsp:useBean> จะมีส่วนที่เป็น ...body... อยู่ โดยส่วนนี้จะถูกเรียกเพียงครั้งเดียวในช่วงที่อินสแทนซ์ของ Bean ถูกโหลดในกรณีที่ไม่มี ...body... เราสามารถเขียน <jsp:useBean> ให้อยู่ในรูปของ Empty tag ได้ โดยใช้

```
<jsp:useBean id="myBeanInstance" class="com.myPackage.BeanClass"
scope="page|request|session|application"/>
```

บางครั้ง <jsp:useBean> อาจจะไม่ทำการโหลด new instance ของ Bean ขึ้นมา ถ้าหากว่าอินส-แทนซ์ดังกล่าวถูกโหลดขึ้นมาก่อนแล้ว โดยบรรทัดฐานของการโหลด Bean จะขึ้นอยู่กับ scope ที่เซตไว้ให้กับอินสแทนซ์ของ Bean นั้น

8.7.2.4 Initializing a Bean - <jsp:setProperty>

ใน Bean Conventions เรากล่าวถึง Bean Constructor ที่ไม่ต้องการอาร์กิวเมนต์ซึ่งถูกใช้สำหรับ <jsp:useBean> อย่างไรก็ตามบางครั้ง Bean ที่เราใช้อาจต้องการค่าเริ่มต้นในการ initialize ซึ่งเราสามารถทำการเซตค่าต่าง ๆ ให้ Bean ได้โดยใช้ <jsp:setProperty .../>

ค่าที่เราจะเซตให้ Bean บางครั้งอาจมาจากค่าที่เรากำหนดขึ้น หรืออาจจะเป็นค่าที่ได้จาก request ของไคลเอ็นต์ซึ่งมาจาก element ต่าง ๆ ใน <FORM> tag ของ HTML ดังนั้นการเซตค่า Bean จึงถูกแบ่งออกเป็นสองแบบคือ

1. การเซตค่าให้ Bean โดยใช้ค่าที่เรากำหนดขึ้น เช่น

```
<jsp:useBean id="myBeanInstance" class="com.myPackage.BeanClass" scope="request">
<jsp:setProperty name="myBeanInstance" property="myProperty" value="definedValue"/>
</jsp:useBean>
```

name คือ ชื่อของ Bean อินสแทนซ์ที่เราต้องการอ้างถึง
property คือ ชื่อของ property ที่เราต้องการเซตค่า
value คือ ค่าที่เราจะใส่เข้าไปให้ property นั้น

2. การเซตค่าให้ Bean โดยใช้ค่าที่ได้จาก element ต่าง ๆ ของ <FORM> tag เช่น

```
<jsp:useBean id="myBeanInstance" class="com.myPackage.BeanClass" scope="request">
<jsp:setProperty name="myBeanInstance" property="myProperty"
param="myFormElementName"/>
</jsp:useBean>
```

param คือ ชื่อของ element ที่อยู่ใน <FORM> tag ที่ส่ง request มาที่ไฟล์ JSP ที่มี <jsp:useBean>

เราไม่สามารถใช้ value และ param พร้อมกันได้เพราะ JSP Container จะไม่ทราบว่าเราจะใช้ค่าไหนในการเซตให้ Bean

ในกรณีที่ค่าทุกค่าใน Bean ขึ้นอยู่กับค่าของ element ต่าง ๆ ใน <FORM > tag เราสามารถบอกให้ JSP Container แมปค่าของ element เข้ากับ ค่า property ต่าง ๆ ของ Bean ได้โดยอัตโนมัติ โดยการใช้ property="*" ดังตัวอย่างข้างล่าง

```
<jsp:useBean id="myBeanInstance" class="com.myPackage.BeanClass" scope="request">
<jsp:setProperty name="myBeanInstance" property="*"/>
</jsp:useBean>
```

เพิ่มเติมอีกนิดหนึ่งคือ <jsp:setProperty> ไม่จำเป็นจะต้องอยู่ใน <jsp:useBean>... </jsp:useBean> เสมอไปเราสามารถเรียกใช้ tag นี้เมื่อไหร่ก็ได้ที่เราต้องการเปลี่ยนหรือเซตค่าของ property ของ Bean

8.7.2.5 Displaying Dynamic Content - <jsp:getProperty>

หลังจากที่เราทำการโหลด และเซตค่าของ property ต่าง ๆ ใน Bean แล้ว เราสามารถดูผลที่เกิดขึ้นได้ โดยใช้

```
<jsp:getProperty name="myBeanInstance" property="myProperty"/>
```

ค่าที่ได้จาก <jsp:getProperty> อาจจะเป็นค่าของ instance variables ของ Bean ที่เปลี่ยนไปหลังจากที่เราใช้ <jsp:setProperty> หรือบางครั้ง อาจจะเป็นลิสต์ของตารางรายชื่อ ในกรณีที่ค่า property ที่เราเซตผ่าน <jsp:setProperty> เป็นตัวกระตุ้นให้ Bean ส่ง query ไปที่ดาต้าเบส เป็นต้น

8.7.2.6 The scope Attribute

โดยทั่วไปในการสร้างอินสแทนซ์ของ Bean แต่ละตัว เราจะต้องกำหนดว่าช่วงชีวิตของ อินสแทนซ์ นั้นจะยาวนานแค่ไหน scope เป็นตัวที่ถูกใช้ในการกำหนดช่วงชีวิตของอินสแทนซ์เหล่านั้น โดยเราสามารถเซต scope ได้เป็นสี่แบบคือ

Scope in JSP	
Page	ออบเจ็กต์จะถูกใช้งานจากไคลเอ็นต์เพียงตัวเดียวจากเพจที่ถูกสร้างขึ้น
Request	ออบเจ็กต์จะถูกใช้งานจากไคลเอ็นต์เพียงตัวเดียวต่อเมื่อมีการร้องขอจากไคลเอ็นต์
Session	ออบเจ็กต์จะถูกใช้งานจากไคลเอ็นต์จนกว่าจะมีการจบการทำงานของ Session
Application	ออบเจ็กต์จะถูกใช้งานโดยไคลเอ็นต์จนกว่าจะมีการจบการทำงานของ Application

ตารางที่ 8-4 The scope Attribute

- **Page** อินสแทนซ์ ของ Bean ที่ถูกเซต scope="page" จะมีช่วงชีวิตอยู่ตั้งแต่ JSP page ที่มีอินสแทนซ์ดังกล่าวอยู่ทำการประมวลผล ไปจนถึงระยะเวลาที่เพจดังกล่าวเสร็จสิ้นการประมวลผลแล้ว
- **Request** ถ้า scope="request" อินสแทนซ์ของ Bean จะมีช่วงชีวิตอยู่ตั้งแต่ request ของไคลเอ็นต์ถูกส่งมาถึง JSP Container จนถึงช่วงที่ ไคลเอ็นต์ได้รับ page output ทั้งหมดกลับไปแล้ว (ซึ่งส่งกลับไปโดย ServletResponse)

ข้อแตกต่างระหว่างสอง scope นี้คือ ในหนึ่ง HttpRequest อาจจะมีการเกี่ยวข้องกับ JSP page หลาย ๆ เพจยกตัวอย่างเช่น การใช้ include directive เพื่อประกอบ JSP page หลาย ๆ เพจเข้าด้วยกันหรือแม้กระทั่งการใช้ <jsp:forward> ซึ่งจะทำการส่ง ServletRequest จาก JSP page หนึ่งไปยังอีกเพจหนึ่ง ดังนั้นเราอาจพูดได้ว่า ในหนึ่ง scope ของ request(ServletRequest) จะประกอบด้วยหลาย ๆ scope ที่เป็นเพจ

- **Session** กรณีของ scope="session", อินสแทนซ์ของ Bean จะมีช่วงชีวิตยาวนานเท่ากับ HttpSession ของไคลเอ็นต์นั้น ๆ ซึ่งก็คือตั้งแต่ไคลเอ็นต์เริ่มทำการติดต่อกับ server ที่มี JSP เพจดังกล่าวอยู่จนถึงเวลาที่ไคลเอ็นต์นั้นหายไปซึ่งอาจเกิดขึ้นจากการปิดเว็บบราวเซอร์หรือการขาดการติดต่อกับ server เป็นเวลานาน ๆ (timeout)
- **Application** scope แบบสุดท้ายก็คือ scope="application" ซึ่งจะยาวนานตั้งแต่อินสแทนซ์ของ Bean นั้นถูกสร้างขึ้นจนถึงระยะเวลาที่ server ที่มี Bean นั้นอยู่ถูก shutdown หรือ restart

ในส่วนของ <jsp:useBean> แล้ว scope จะเป็นตัวกำหนดว่า JSP Container จะต้องทำการโหลดอินสแทนซ์ของ Bean ขึ้นมาใหม่หรือไม่ยกตัวอย่างเช่น ถ้าอินสแทนซ์ของ Bean ถูกเซต scope ให้เป็น request, JSP Container จะทำการสร้าง Bean อินสแทนซ์ขึ้นมาใหม่ ในทุกครั้งที่มี request ใหม่ถูกส่งมาจากไคลเอ็นต์โดยหลังจากที่ request ดังกล่าวเสร็จสิ้นลง Bean อินสแทนซ์ที่ถูกสร้างขึ้นโดย request นั้นจะถูกทำลาย โดย JSP Container จะทำการสร้างขึ้นมาใหม่อีกครั้งสำหรับ request อันถัดไป

ในกรณีของ Bean อินสแทนซ์ที่ถูกเซต scope ให้เป็น session, JSP Container จะทำการสร้างอินสแทนซ์ดังกล่าวขึ้นในครั้งแรกที่อินสแทนซ์นั้นถูกเรียกใช้เท่านั้น ถ้าอินสแทนซ์นี้ถูกเรียกใช้อีกในอนาคต, JSP Container จะไม่ทำการสร้าง Bean อินสแทนซ์ขึ้นมาอีก แต่จะทำการโหลดอินสแทนซ์ที่สร้างขึ้นไว้แล้วมาใช้แทน

8.7.3 <jsp:forward> Action

Action นี้อนุญาติให้เราส่งผ่าน forward ตัว request ของไคลเอ็นต์จาก JSP page ปัจจุบันไปยังเพจอื่น ๆ ซึ่งอาจจะเป็น JSP หรือ Servlet (ที่อยู่ใน ServletContext) เดียวกันก็ได้ซึ่งตัว <jsp:forward> จะทำลาย content ของเพจปัจจุบันที่จะส่งให้กับ ไคลเอ็นต์โดยจะแทนด้วย content ของเพจที่ถูก forward ไปแทนลักษณะการใช้ก็คือ

```
<jsp:forward page="urlSpec" />
```

urlSpec คือ relative URL ของเพจที่เราส่งผ่าน request ไปให้

ตัวอย่างที่ใช้กัน ก็อาจจะเป็น

```
<jsp:forward page="/utils/errorPage.jsp" />
<jsp:forward page="<%= whereTo %>" />
```

จะสังเกตเห็นว่า ในส่วนของ page attribute เราสามารถใส่ค่าที่เป็นไดนามิกก็ได้

8.7.4 <jsp:plugin> Action

เราสามารถใช้ <jsp:plugin> ซึ่งจะทำการตรวจชนิดของเว็บบราวเซอร์และใส่ค่า parameter ต่าง ๆ ที่จำเป็นสำหรับ <APPLET> เข้าไปใน <jsp:plugin> ด้วย ยกตัวอย่างเช่น

```
<jsp:plugin type="applet" code="Molecule.class" codebase="/html">
<jsp:params>
<jsp:param name="molecule" value="molecules/benzene.mol"/>
</jsp:params>
<jsp:fallback>
<p> unable to start plugin </p>
</jsp:fallback>
</jsp:plugin>
```

<jsp:param> เป็นตัวบ่งบอกถึง parameter ต่าง ๆ ที่ใช้สำหรับ Applet

<jsp:fallback> เป็นตัวที่ใช้ใส่ข้อความเพื่อบอกว่าเกิดอะไรขึ้น ในกรณีที่ Java Plugin ไม่สามารถ start หรือเกิดข้อผิดพลาดบางอย่างขึ้น

บทที่ 9
บทสรุปโปรแกรมทำสัญญาซื้อขายบ้านจัดสรร

9.1 ลักษณะของระบบ

ระบบทำสัญญาซื้อขายบ้านจัดสรรจัดทำขึ้นมาเพื่อเป็นตัวกลาง ในการนัดหมายทำสัญญาซื้อขายบ้านระหว่างลูกค้ากับเจ้าของโครงการบ้านจัดสรรได้ โดยมีการติดต่อกันผ่านทางระบบอินเทอร์เน็ต

ระบบจะมีฐานข้อมูลที่ติดต่อกับเว็บเซิร์ฟเวอร์เพื่อให้บริการแก่ ผู้ดูแลระบบ,ลูกค้า,เจ้าของโครงการ โดยใช้ระบบจัดการฐานข้อมูลแบบรีเลชันแนลของออราเคิลและใช้เว็บเซิร์ฟเวอร์ของ OAS (Oracle Application Server) โดยใช้ JDBC Driver เป็นตัวเชื่อมต่อระหว่างเว็บเซิร์ฟเวอร์และฐานข้อมูล เครื่องไคลเอ็นต์ที่ติดต่อกับระบบจะใช้เว็บบราวเซอร์ในการติดต่อกับระบบ

ก่อนการใช้งานโปรแกรมนั้นจะมีการตรวจสอบสิทธิ ในการเข้าไปใช้ระบบซึ่งแบ่งเป็น ผู้ดูแลระบบจะมีสิทธิในการเพิ่มข่าวสารและดูข้อมูลเกี่ยวกับโครงการบ้าน ลูกค้าจะมีสิทธิในการลงทะเบียน, คำนวณเงินผ่อน, ค้นหาข้อมูลโครงการบ้านและทำสัญญาจอง ส่วนเจ้าของโครงการนั้นะมีสิทธิในการเพิ่มข้อมูลของบ้านซึ่งมีอยู่ในโครงการและดูรายงานผลการนัดหมายในการมาทำสัญญาได้

9.2 ลักษณะของโปรแกรมและการใช้งาน

โปรแกรมระบบทำสัญญาซื้อขายบ้านจัดสรร เป็นโปรแกรมตัวกลางที่ใช้สำหรับจองบ้านผ่านทางอินเทอร์เน็ตเพื่อให้ลูกค้าสามารถทำการทำสัญญากับผู้ขายบ้านในภายหลังได้

ลักษณะหน้าตาของโปรแกรมมีรูปแบบใกล้เคียงกับส่วนต้นแบบส่วนติดต่อผู้ใช้ที่ได้ออกแบบเอาไว้โดยจะมีรูปแบบของโปรแกรมและการใช้งานต่าง ๆ ดังนี้

9.2.1 เริ่มต้นการใช้งานโปรแกรม หน้าจอแรกจะเป็นการประกาศข่าวเกี่ยวกับการซื้อขายบ้านจัดสรร ผู้ใช้สามารถทราบถึงข้อมูลเกี่ยวกับบ้านและทราบข้อมูลข่าวสารเกี่ยวกับการซื้อขายได้ดังแสดงในรูปที่ 9-1 และรูป 9-2

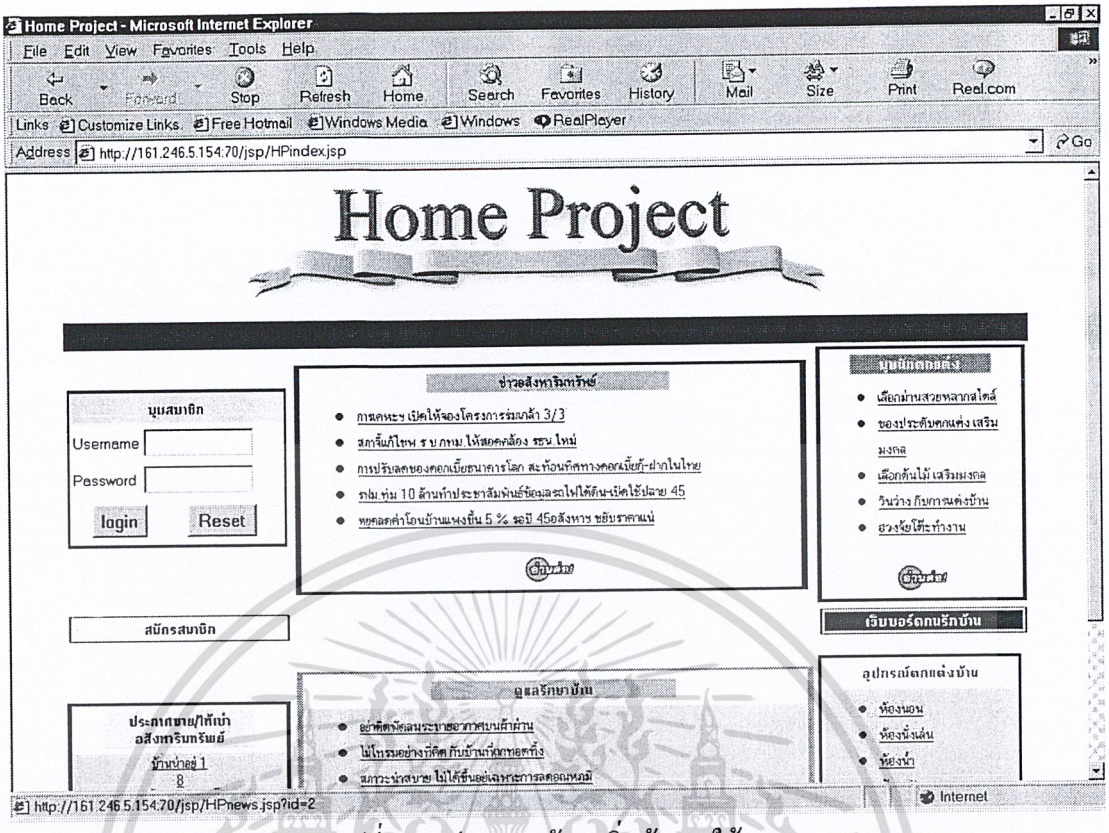
9.2.2 การเพิ่ม ลบและปรับปรุงข่าวสาร ในส่วนการเพิ่มและลบข่าวสารนี้ผู้ดูแลระบบเท่านั้นที่จะทำได้โดยการใส่ชื่อและรหัสผ่าน นอกจากนี้ผู้ดูแลระบบยังทำการปรับปรุงข่าวสารได้ดังแสดงในรูปที่ 9-3

9.2.3 การค้นหาข้อมูลบ้าน ในส่วนการค้นหาข้อมูลจะเป็นส่วนที่ใช้สำหรับผู้ใช้หรือผู้ที่มีความสนใจทั่วไป เพื่อทำการดูรายละเอียดเกี่ยวกับบ้านที่ตนเองต้องการได้ดังแสดงในรูปที่ 9-4

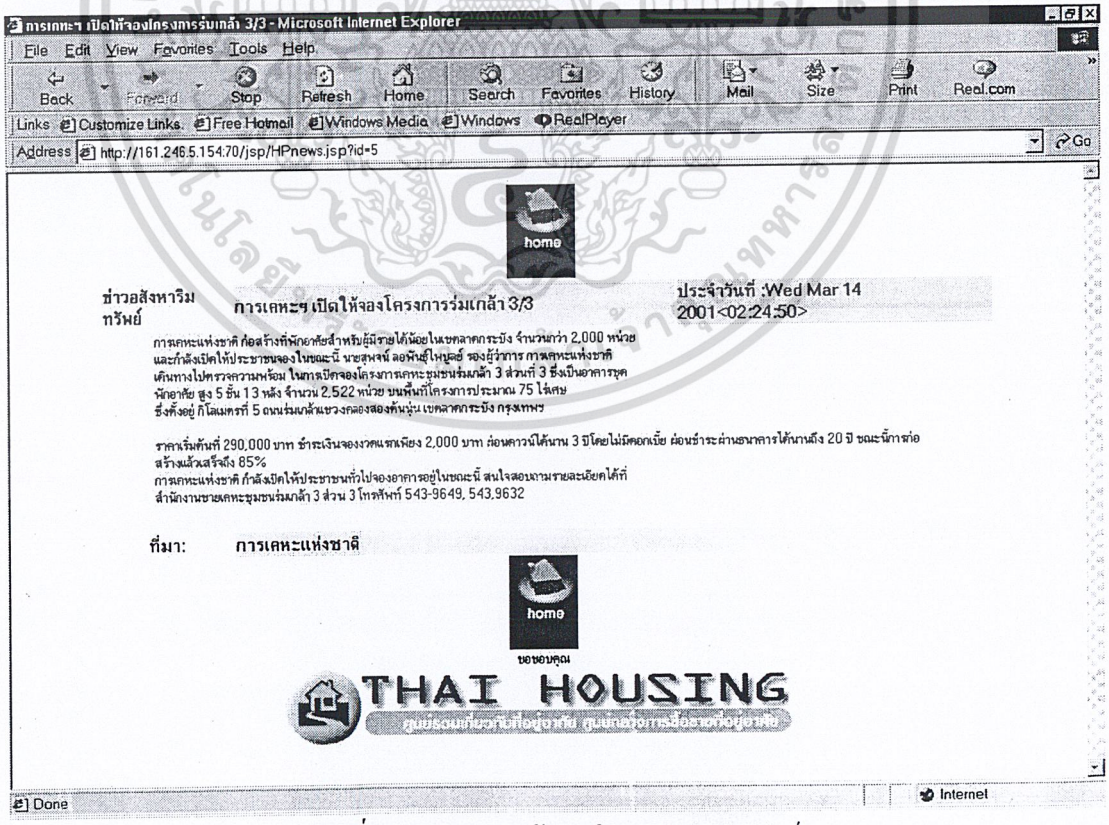
9.2.4 การลงทะเบียน ผู้ที่มีความสนใจที่จะทำการจองบ้าน สามารถลงทะเบียนและกรอกรายละเอียดส่วนตัวเพื่อเป็นสมาชิกของระบบได้ดังแสดงในรูปที่ 9-5

9.2.5 การทำสัญญาจอง เมื่อลูกค้าทำการลงทะเบียนแล้วก็จะเลือกบ้านเพื่อทำการจองแล้วนัดหมายวันที่มาทำสัญญาซื้อขายกับเจ้าของโครงการบ้านได้ดังแสดงในรูปที่ 9-6

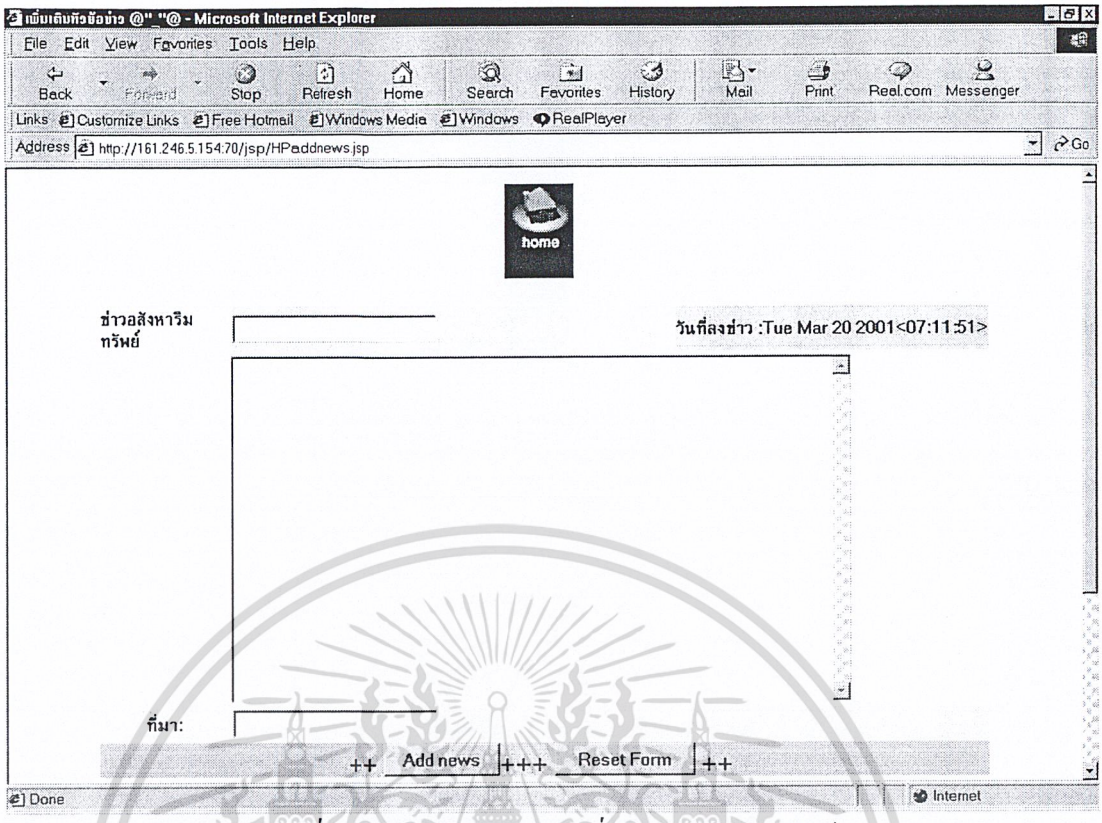
9.2.6 ตรวจสอบวันนัดหมายทำสัญญา เจ้าของโครงการบ้านสามารถตรวจสอบวันที่นัดหมายในการทำสัญญาซื้อขายบ้านได้ดังแสดงในรูปที่ 9-7



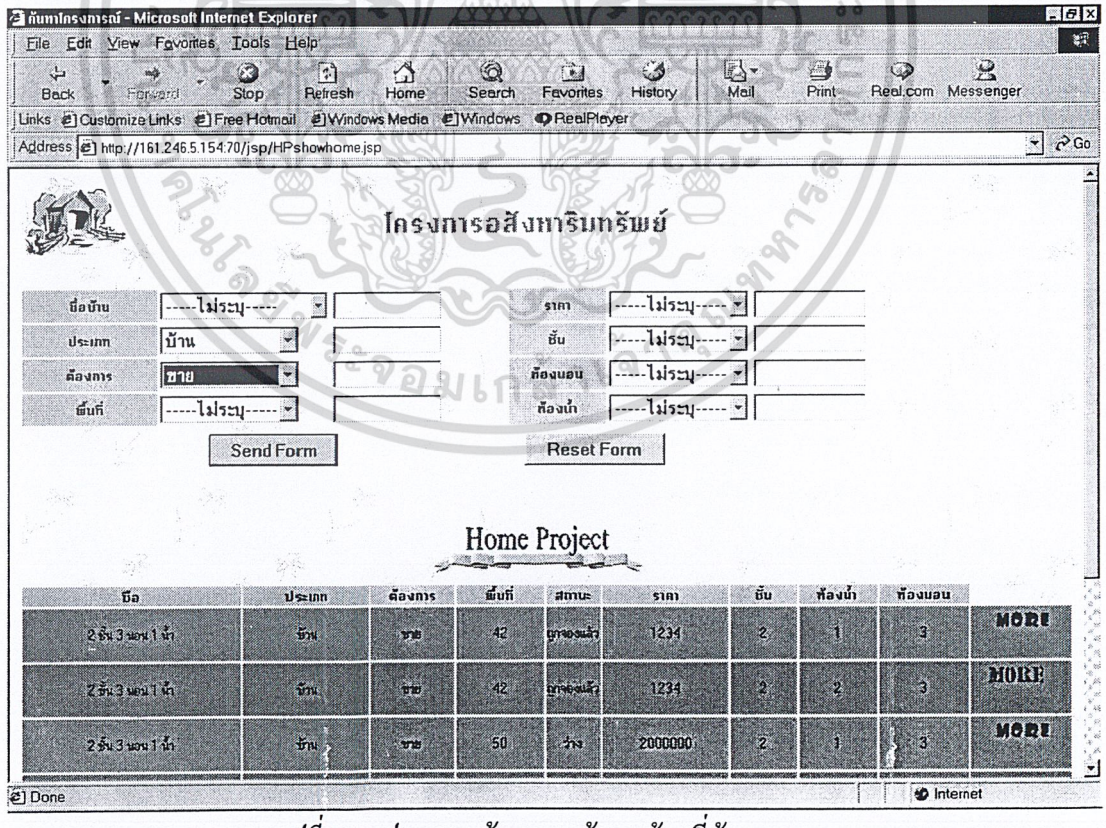
รูปที่ 9-1 รูปแสดงหน้าจอเริ่มต้นการใช้งาน



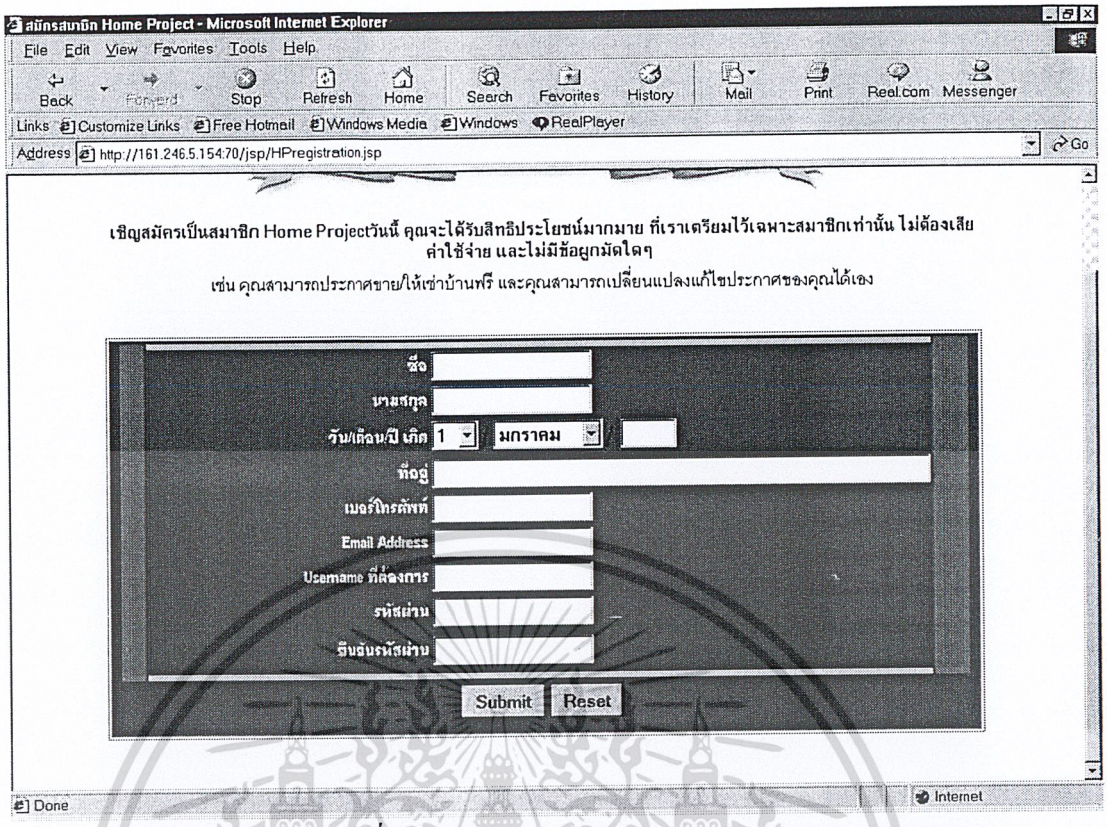
รูปที่ 9-2 รูปแสดงหน้าจอข่าวอสังหาริมทรัพย์



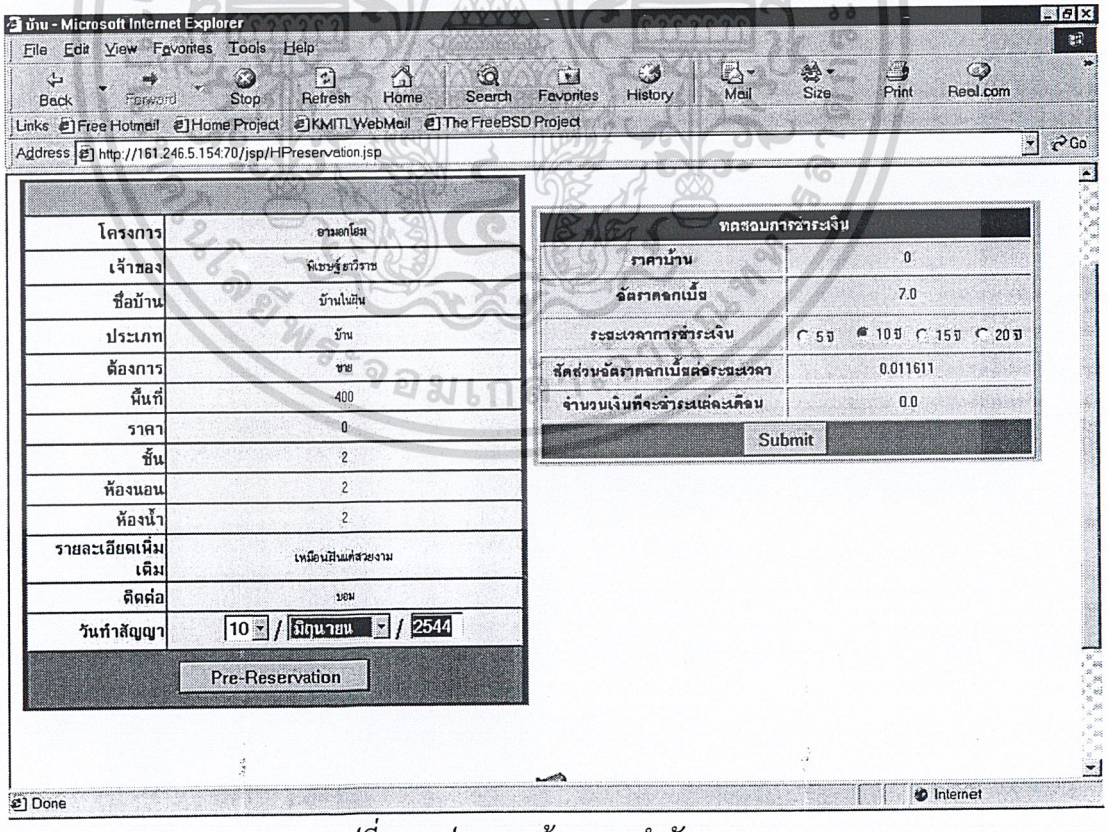
รูปที่ 9-3 รูปแสดงหน้าจอการเพิ่มข่าวอสังหาริมทรัพย์



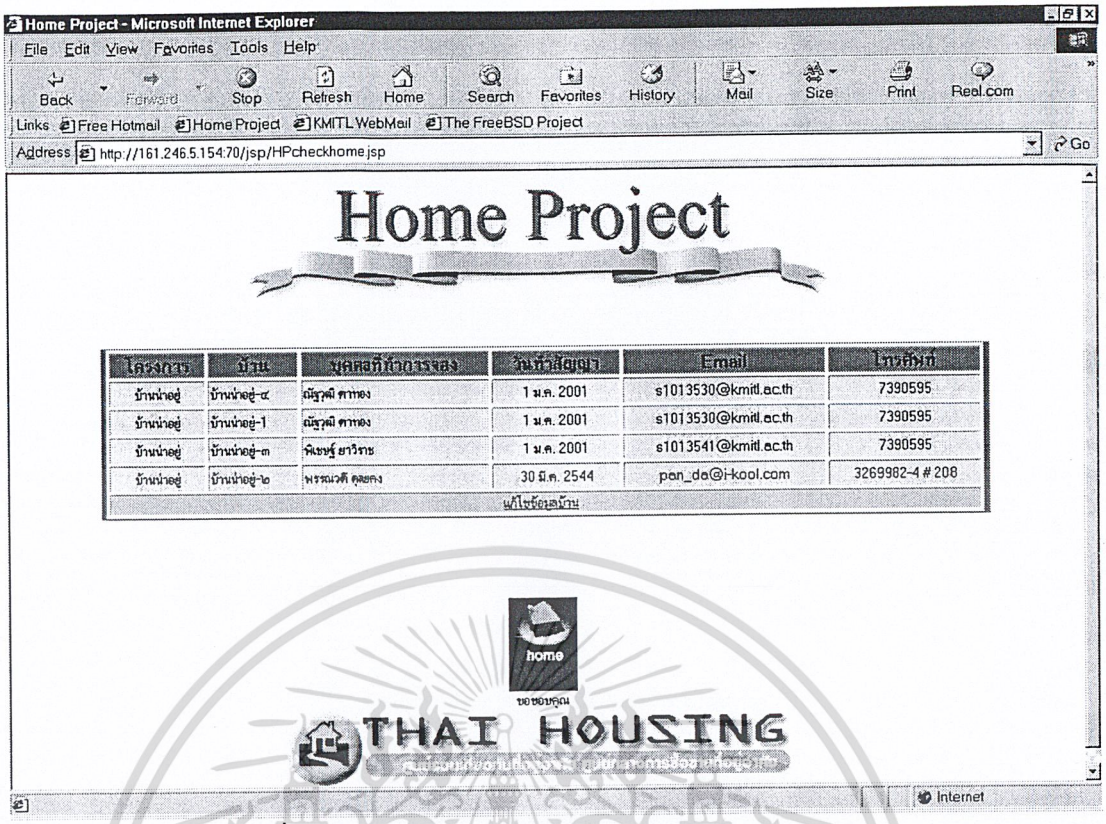
รูปที่ 9-4 รูปแสดงหน้าจอการค้นหาบ้านที่ต้องการขาย



รูปที่ 9-5 รูปแสดงหน้าจอการสมัครสมาชิก



รูปที่ 9-6 รูปแสดงหน้าจอการทำสัญญาจอง



รูปที่ 9-7 รูปแสดงหน้าจอการตรวจสอบวันนัดหมายทำสัญญา

9.3 การทดสอบการทำงานของโปรแกรม

การทดสอบการทำงานของโปรแกรม จะแบ่งแนวทางการทดสอบเป็น 2 ประเภท คือการทดสอบการทำงานทั่วๆไปเพื่อให้ระบบสามารถทำงานได้ถูกต้องตามที่คาดหวังไว้ และการทดสอบเพื่อหาข้อผิดพลาดของโปรแกรมซึ่งทำได้โดยการป้อนข้อมูลที่คาดว่าจะทำให้เกิดข้อผิดพลาดขึ้น เช่น ข้อมูลที่เกินช่วงขอบเขตที่ระบุไว้ในโปรแกรม

9.3.1 การทดสอบการทำงานทั่วๆไปของโปรแกรม

เราจะทำการทดสอบการทำงานต่างๆ ของโปรแกรมให้เป็นไปตามโดเมนของปัญหาที่ได้กล่าวไว้ในบทที่ 4 ซึ่งสามารถจำแนกหัวข้อการทดสอบต่างๆ ได้ดังนี้

- การคำนวณเงินผ่อนชำระต่องวดเพื่อแสดงเป็นตัวอย่าง
- การเก็บบันทึกประวัติลูกค้าและรายละเอียดของโครงการบ้าน
- การทำสัญญาซื้อขายบ้านจัดสรร
- การแสดงตารางการนัดหมายในการทำสัญญา
- การแสดงรายละเอียดโครงการบ้านและเพิ่มข่าวสาร
- การค้นหาข้อมูลเกี่ยวกับโครงการบ้านที่ต้องการ

9.3.2 การทดสอบเพื่อหาข้อผิดพลาดของโปรแกรม

ในการทดสอบเพื่อหาข้อผิดพลาดของโปรแกรม เราจะทดสอบระบบเพื่อหาข้อผิดพลาดที่เกิดจากเหตุการณ์ต่างๆ เพื่อหาทางแก้ไขข้อผิดพลาดเหล่านั้นให้เหลือน้อยที่สุด โดยจะมีหัวข้อในการทดสอบการทำงานต่างๆ ดังนี้

- ทำการป้อนข้อมูลที่เกินช่วงที่กำหนดไว้ของระบบ
- ป้อนข้อมูลผิดประเภท เช่น ป้อนข้อมูลที่เป็นสตริงในช่องจำนวนเงิน
- ป้อนข้อมูลลูกค้าซ้ำ เพื่อทดสอบการบันทึกข้อมูลลูกค้าที่ซ้ำซ้อน
- ทดสอบการทำงานของเครือข่าย เช่น การทำงานเมื่อการติดต่อผ่านเครือข่ายมีปัญหา
- ทดสอบการทำงานเมื่อมีผู้ใช้ระบบหลายรายพร้อมกัน

หัวข้อการทดสอบต่างๆข้างต้น เป็นหัวข้อการทดสอบเบื้องต้นสำหรับจำลองการใช้งาน เนื่องจากโปรแกรมที่ได้สร้างขึ้นมานั้นเป็นเพียงการจำลองขึ้นเพื่อศึกษาถึงวิธีการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ ไม่ใช่โปรแกรมที่ใช้สำหรับการทำงานจริง

9.4 บทสรุปการพัฒนาโปรแกรมทำสัญญาซื้อขายบ้านจัดสรร

แอพพลิเคชันเซิร์ฟเวอร์ได้เป็นอีกแนวทางหนึ่งในการสร้างและออกแบบโปรแกรมประยุกต์บนเว็บไซต์ซึ่งมีจะแบ่งส่วนของการทำเว็บไซต์กับการเขียนโปรแกรมออกจากกัน ทั้งนี้เพื่อให้การพัฒนาและการปรับปรุงให้เป็นไปอย่างรวดเร็ว โปรแกรมประยุกต์สำหรับระบบทำสัญญาซื้อขายบ้านจัดสรร ได้ใช้แนวทางการพัฒนาเชิงวัตถุทำให้โมเดลที่ได้มีความสอดคล้องกันทุกขั้นตอนการพัฒนา ทำให้ได้ระบบที่ตรงกับความต้องการของผู้ใช้มากที่สุด โดยเครื่องมือที่ช่วยในการจัดทำโครงการได้แก่

- โปรแกรม Rational Rose เป็น CASE tool ที่มีการนำเสนอโมเดลในรูปแบบสัญลักษณ์ของ UML ทำให้การวิเคราะห์และออกแบบระบบได้ดีขึ้น
- โปรแกรมออราเคิล เป็นโปรแกรมระบบจัดการฐานข้อมูล ช่วยในการสร้างตารางฐานข้อมูลแบบรีเลชันแนล มีประสิทธิภาพในการจัดการฐานข้อมูลสูง
- โปรแกรม OAS เป็นเว็บเซิร์ฟเวอร์ที่คอยให้บริการของระบบให้แก่ผู้ใช้ทาง อินเทอร์เน็ตซึ่งสนับสนุนการทำงานของ Servlet Engine และ JSP Container เพื่อเป็นตัวคอมไพล์ JSP ไฟล์ให้ทำงานได้
- โปรแกรม Borland J Builder 3.0 เป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมภาษาจาวา ซึ่งเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ โดยเราจะเขียนโปรแกรม JavaBeans แล้วคอมไพล์ผ่านเครื่องมือนี้

9.5 แนวทางในการพัฒนาเพิ่มเติม

9.5.1 เพิ่มส่วนของการปรับสถานะของลูกค้าให้มีความเหมาะสมต่อสถานการณ์จริงมากขึ้น เพิ่มส่วนการแก้ไขวันนัดหมายทำสัญญาซื้อขายบ้านเพื่อให้เจ้าโครงการบ้านสามารถทำการปรับเปลี่ยนได้

9.5.2 เพิ่มเติมฟังก์ชันและข้อกำหนดรายละเอียดการทำงานให้เหมือนจริง ขอบเขตของฟังก์ชันการทำงานของระบบอยู่ในโดเมนที่จำลองมาและกำหนดความต้องการของระบบขึ้นมาเอง หากต้องการนำระบบไปใช้งานทางธุรกิจจริงจะต้องแก้ไขหรือเพิ่มเติมให้เป็นไปตามโลกแห่งความเป็นจริงทางธุรกิจ

9.5.3 ปรับปรุงส่วนของเว็บไซต์โดยใช้ JSP ติดต่อกับ XML (Extensible Markup Language) เพื่อการอัพเดตข้อมูลข่าวสารในระบบที่มีการเปลี่ยนแปลงได้ดียิ่งขึ้น

ภาคผนวก ก
การติดตั้ง Java Mail API 1.2

การติดตั้ง JavaMail(TM) API 1.2

ระบบการทำสัญญาซื้อขายบ้านจัดสรรนั้นต้องมี ฟังก์ชันหน้าที่การทำงานของการส่งอี-เมล์ ฉะนั้นเราจึงเลือก JavaMail API 1.2 เนื่องจากมีคุณสมบัติต่าง ๆ ที่น่าสนใจดังต่อไปนี้

Welcome to the JavaMail API 1.2 release! This release includes versions of the JavaMail API implementation, IMAP, SMTP, and POP3 service providers, some examples, and documentation for the JavaMail API. Please see the FAQ at http://java.sun.com/products/javamail/FAQ.html

JDK Version notes

The JavaMail API supports JDK 1.1.x or higher, including the Java 2 platform, Standard Edition (J2SE) v1.2.x. Note that we have currently tested this implementation only with JDK 1.1.7 and J2SE 1.2.2. Note also that JAF is known to have problems with JDK versions before 1.1.6.

Protocols supported

This release supports the following mail protocols:

IMAP - a message Store protocol, for reading messages from a server

SMTP - a message Transport protocol, for sending messages to a server

POP3 - a message Store protocol, for reading messages from a server

See our web page at http://java.sun.com/products/javamail for the latest information on third party protocol providers.

Contents

Included in this release are the following:

README.txt	this file
LICENSE.txt	Software license
NOTES.txt	Notes, issues and known bugs
CHANGES.txt	Changes since the previous release
mail.jar	The JavaMail API and all service providers
mailapi.jar	The JavaMail API with no service providers
imap.jar	The IMAP service provider
smtp.jar	The SMTP service provider
pop3.jar	The POP3 service provider

Requirements

Note that the JavaMail API requires the JavaBeans(TM) Activation Framework package to be installed as well. Download the latest version of the JavaBeans Activation Framework from http://java.sun.com/beans/glasgow/jaf.html and install it in a suitable location.

Installation in Windows NT/95

1. Unzip the javamail-1_2.zip archive. (you may have already done this)

2. Set your CLASSPATH to include the "mail.jar" file obtained from the download, as well as the current directory.

Assuming you unzipped javamail-1_2.zip in c:\download the following would work:

```
set CLASSPATH=%CLASSPATH%;c:\download\javamail-1.2\mail.jar;.
```

Also include the "activation.jar" file that you obtained from downloading the JavaBeans Activation Framework, in your CLASSPATH.

```
set CLASSPATH=%CLASSPATH%;c:\download\activation\activation.jar
```

3. Go to the demo directory

4. Compile any demo using your java compiler. For example:

```
javac msgshow.java
```

5. Run the demo. The '-' option lists the required and optional command-line options to successfully run any demo. For example:

java msgshow - lists the available options

And

java msgshow -T imap -H <mailserver> -U <username> -P <passwd> -f INBOX 5

uses the IMAP protocol to display message number 5 from your INBOX. (Additional instructions on how to run the simple mail reader demo and servlet demo are provided in demo/client/README.txt and demo/servlet/README.txt, respectively.)

Problems

Our web page at http://java.sun.com/products/javamail has a pointer to the FAQ that includes information on protocols supported, installation problems, etc. You'll also find information on how to subscribe to our discussion-list or announcement-list for announcements of future JavaMail API releases. See the NOTES.txt file for information on how to report bugs.

ภาคผนวก ข

การติดตั้ง JavaBeans(tm) Activation Framework 1.0.1

การติดตั้ง JavaBeans(TM) Activation Framework 1.0.1

ในการติดตั้ง JavaMail API 1.2 นั้นจำเป็นต้องมี JavaBeans™ Activation Framwork 1.0.1 มาช่วยทำงานด้วยซึ่งมีขั้นตอนการติดตั้งดังต่อไปนี้

Welcome to the JavaBeans(tm) Activation Framework! A great deal of time and effort was put into making this standard extension functional and usable. We are however, realistic in anticipating that there exist scenarios and applications where this implementation could be improved upon. We welcome your feedback in this area to the email address listed below.

System Requirements

The JavaBeans(tm) Activation Framework (JAF) was developed and tested against Sun's JDK 1.1.6 on the Solaris SPARC platform, and Windows 32 (both Windows 95, and Windows NT 4.0). This version of the JAF is written in Java (with no native code). It will run on any JDK(tm) 1.1 compatible virtual machine so don't hesitate to try it and please report failures.

Installation

There is effectively *no* installation of the JAF. The classes that make up the JAF are contained in the included Java(tm) Archive (JAR) file,"activation.jar". This file can be placed anywhere accessible to the Java virtual machine running on your system. The only requirement is that the activation.jar be included in your system's class path so Java can find the JAF classes.

Related Web Sites

There are a number of web sites you might find useful if you haven't already run across them:

http://java.sun.com -- This is Sun's Java Software External Web Site.

http://java.sun.com/beans -- This is the JavaBeans web site which is full of the latest beans information.

The JAF spec is available in the 'Glasgow' area.

http://java.sun.com/products/javamail -- The JavaMail(tm) API provides a set of abstract classes that model a mail system.It's implementation is dependent on the JAF.

บรรณานุกรม

[1] Sanjay Singh (2000) : *"Oracle Application Server Developer's Guide:.IServlet and JSP Application Release 4.0.8.2"* ,Oracle Corporation

[2] Scott McPherson (2000):*"JavaServer Pages: A Developer's Perspective"*, http://developer.java.sun.com

[3] jGuru: *"JavaServer Pages Fundamentals"*, http://developer.java.sun.com

[4] Julie Basu: *"Database access from JavaServer Pages"*, Oracle Corporation

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

[6] Meghraj Thakkar:*"SAMS Teach Yourself Oracle 8I on Windows NT in 24 Hours"*, Sams publishing 1999

[7] Michael Ault: *"Oracle Administration and Management"* ,Wiley Computer publishing 1997

[8] G.M Nyssen and E.D. Falkenberg :*"Introduction IBM SQL"*, Nyssen Data Bases Pty.Ltd. 1984