การพัฒนาระบบงานเชิงวัตถุโดยใช้คอมโพเนนต์สำหรับสถาปัตยกรรมเอ็นเทียร์ส

N-Tiers Systerm Development Using Component

นายร่มธรรม สินธุประสิทธิ์

นายวัลลภ วรกิจวิบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การพัฒนาระบบงานเชิงวัตถุโดยใช้คอมโพเนนต์สำหรับสถาปัตยกรรมเอ็นเทียร์ส

N-Tiers Systerm Development Using Component

โดย

นายร่มธรรม สินธุประสิทธิ์

นายวัลลภ วรกิจวิบูลย์

อาจารย์ที่ปรึษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบงานเชิงวัตถุโดยใช้คอมโพเนนท์สำหรับสถาปัตยกรรมเอ็นเทียร์ส
(N-Tiers System Development Using Component)

ผู้จัดทำ

1. นายร่มธรรม สินธุประสิทธิ์ รหัสประจำตัว 40010644
2. นายวัลลภ วรกิจวิบูลย์ รหัสประจำตัว 40010718

อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การพัฒนาระบบงานเชิงวัตถุโดยใช้คอมโพเนนต์สำหรับสถาปัตยกรรมเอ็นเทียร์ส

นายร่มธรรม สินธุประสิทธิ์ รหัส 40010644
นายวัลลภ วรกิจวิบูลย์ รหัส 40010718
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

คอมโพเนนต์ออบเจ็กต์โมเดล(COM) เป็นโมเดลในการพัฒนาระบบแบบใหม่ที่ช่วยแก้ปัญหาต่าง ๆ ที่เกิดจากการพัฒนาระบบงานเชิงออบเจ็กต์ โดย COM ออบเจ็กต์ลักษณะเป็นไบนารีที่เป็นมาตรฐานในการทำงานร่วมกันระหว่างออบเจ็กต์ มีลักษณะไม่ขึ้นกับภาษาที่ใช้พัฒนา ไม่ขึ้นกับระบบปฏิบัติการที่ใช้ เป็นอิสระต่อตำแหน่งที่ตั้ง และสามารถนำกลับมาใช้ใหม่ได้ โมเดลนี้ช่วยแก้ปัญหาด้านเวอร์ชันของซอฟต์แวร์ การพัฒนาและปรับปรุงส่วนต่าง ๆ ของระบบงานไม่ส่งผลกระทบต่อส่วนอื่น ๆ ของระบบ

โครงงานนี้ได้ศึกษาทฤษฎีคอมโพเนนต์ออบเจ็กต์โมเดล และพัฒนาคอมโพเนนต์เพื่อใช้ในระบบงานเชิงกระจายแบบหลายเทียร์ พัฒนาไปสู่เทคโนโลยีคอมพลัสซึ่งทำให้คอมโพเนนต์มีประสิทธิภาพในด้านต่าง ๆ เช่น ความสามารถในการจัดการทรานแซคชั่น เป็นต้น และพัฒนาระบบงานทั้งหมดโดยอ้างอิงสถาปัตยกรรมวินโดวส์ดีเอ็นเอ

เมื่อนำเทคโนโลยีต่าง ๆ มารวมกันจะทำให้สามารถออกแบบระบบงานที่มีความซับซ้อนให้มีประสิทธิภาพ มีความยืดหยุ่น สามารถแก้ปรับปรุงและบำรุงรักษาได้ง่าย โดยโครงงานนี้ได้ออกแบบและพัฒนาระบบพานิชย์อิเล็กทรอนิกส์ที่ใช้แนวความคิดและเทคโนโลยีทั้งหมดที่กล่าวมา

N-Tiers Systerm Development Using Component

Mr.Romtham Sinthurprahsith

Mr.Wanlop Worakitviboon

Dr.Worawat Limpoka Advisor

Abstract

Component object models (COM) are new model for develop newly system, which solve any problem that found in object oriented development. COM Components are standard binary for work between objects make them not depend on programming language, operation system, location and reuseable. This model been solve software version problem and development in each part of system don't affect to other parts of system.

This project is study in COM theory and develop COM components for multi-tiers distributed system, COM+ technology that make COM components more powerful in a lot of features such as transaction management and develop system with DNA architecture.

Integrating all technology make complex system but has powerful but flexible for develop and maintenance. In this project include design and develop E-Commerce system by all technology that been describe.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จลงได้ หากไม่ได้รับความช่วยเหลือและความร่วมมือจากบุคคลหลาย ๆ ฝ่าย บุคคลท่านแรกคือ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา ที่คอยเอาใจใส่ให้คำปรึกษา แนะแนวทาง ผลักดัน ให้ความรู้ทั้งทางด้านวิชาการและเรื่องอื่น ๆ ทำให้โครงงานสำเร็จลุล่วงด้วยดี ต้องขอขอบพระคุณเป็นอย่างมาก

ขอบคุณเพื่อน ๆ ทุกคนโดยเฉพาะที่ห้องปฏิบัติงาน OLALA ที่ช่วยเหลือและอยู่ร่วมทุกข์ ร่วมสุข ร่วมกิน ร่วมเล่น(เกม) กันมาตั้งแต่เริ่มทำโครงงานจนเสร็จ

สุดท้ายขอขอบพระคุณบุพการี บิดา มารดา และญาติพี่น้อง ที่อบรมสั่งสอน เลี้ยงดู ให้กำลังใจ และกำลังทรัพย์ในการศึกษาจนกระทั่งลูกน้อยมีวันนี้ ลูกน้อยรำลึกถึงพระคุณสุดประมาณ ขอ กราบขอบพระคุณมา ณ ที่นี้

ร่มธรรม สินธุประสิทธิ์
วัลลภ วรกิจวิบูลย์

สารบัญ

หน้าที่

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

การพัฒนาโปรแกรมคอมพิวเตอร์นั้นได้มีการพัฒนาจาก การใช้การพัฒนาโดยใช้รูปแบบโครงสร้าง (Structural Development) ไปเป็นการพัฒนาเชิงวัตถุ (Object Oriented Development) ซึ่งก็มีข้อดีที่สามารถจัดการกับระบบที่มีความซับซ้อนสูงได้อย่างมีประสิทธิภาพมากขึ้น แต่ในขณะเดียวกันตัวการพัฒนาแบบวัตถุ ก็มีจุดบกพร่องอยู่เป็นจำนวนมากเช่น จุดบกพร่องของหลักการสืบทอดคุณสมบัติ (Inheritance) การนำกลับมาใช้ใหม่ซึ่งยังไม่มีประสิทธิภาพเพียงพอเป็นต้น ซึ่งทำให้เกิดแนวความคิดใหม่ของการ นำกลับมาใช้ใหม่ซึ่งมีประสิทธิภาพสูงกว่า และเชื่อถือได้มากกว่ารูปแบบเดิม

การพัฒนาโดยสถาปัตยกรรม COM เป็นพื้นฐาน(COM base Architecture) เป็นอีกแนวทางหนึ่งซึ่งได้ถูกนำมาใช้ โดยมีจุดเด่นดังนี้

- มีความเป็นอิสระในการพัฒนาไม่ขึ้นกับเครื่องพัฒนาใด ๆ เช่น COM ออบเจ็กต์ที่ถูกสร้างจาก Borland Delphi 5.0 ก็สามารถเรียกใช้โดย โปรแกรมที่พัฒนาโดย Microsoft Visual Basic ได้
- มีความสมบูร์ภายในตัวเองไม่ต้องคอมไพล์ใหม่เมื่อมีการเรียกใช้จากไคล์เอนต์
- มีความสะดวกในการเรียกใช้ การเรียกใช้ไม่ว่าตัว COM ออบเจ็กต์นั้นจะอยู่ที่ไหนก็ไม่มีความแตกต่างกันมากนัก
- เป็นหลักการพื้นฐานของ OLE และ ActiveX
- เนื่องจากมีการใช้อินเตอร์เฟสเป็นตัวติดต่อกับไคลเอนต์ซึ่งอินเตอร์เฟสนั้นจะไม่มีการเปลียนแปลงการ
- เปลี่ยนแปลงภายในตัวเซิร์ฟเวอร์จะไม่มีผลต่อตัวไคลเอนต์โดยตรง

เมื่อระบบงานนั้นต้องการจะมีเพิ่มความสามารถในการเชื่อมต่อกับ อินเตอร์เน็ต ทางบริษัทไมโครซอฟท์ก็ได้เสนอสถาปัตยกรรมที่เป็นสภาพแวดล้อมที่เหมาะสม กับ แพลตฟอร์มของไมโครซอฟท์ ซึ่งเรียกว่า DNA องค์ประกอบหนึ่งที่สำคัญของ DNA คือ สภาพแวดล้อม COM+ รันไทม์ หรือ MTS เวอร์ชัน 3.0 นั่นเอง ซึ่งมีบริการใหม่ๆ เพิ่มเติมจาก MTS เวอร์ชันก่อนหน้าร่วมทั้งยังใช้งานง่ายยิ่งกว่า

ปริญานิพนธ์ฉบับนี้ได้นำเสนอทั้งทฤษฎี และตัวอย่างการสร้าง ทั้ง COM คอมโพเนนท์ และ การออกแบบระบบงานขนาดใหญ่ โดยใช้หลักการคอมโพเนนท์ ซึ่งมีความละเอียดพอสมควร

วัตถุประสงค์

ภายในโครงงานนี้ได้ทำการศึกษาถึงทฤษฎี ,การสร้าง และการนำ COM ออบเจ็กต์ไปใช้งานโดยมีจุดประสงค์ดังนี้

1. เข้าใจถึงทฤษฎี พื้นฐานของ COM ลักษณะภายในตัว COM ออบเจ็กต์ และองค์ประกอบที่เกี่ยวข้อง
2. รู้วิธีการสร้าง COM ออบเจ็กต์ ทั้งแบบ in-process, out-of-process และแบบ remote server

3. เข้าใจถึงทฤษฎีพื้นฐานของสภาพแวดล้อม COM+ รันไทม์และบริการต่าง ๆ ที่มาพร้อมกันกับ DNA
4. สามารถออกแบบระบบงานเชิงคอมโพเนนท์ และสร้างระบบงานที่ออกแบบในลักษณะคอมโพเนนท์ โดยใช้เทคโนโลยี COM + ร่วมกับ DNA

บทที่ 2
การวิเคราะห์ ออกแบบและการพัฒนาระบบเชิงวัตถุ

2.1 ความหมายของการโปรแกรมเชิงออบเจ็กต์ (Object-Oriented Programming)

โอโอพี หรือ OOP เป็นคำย่อของ Object-Oriented Programming ซึ่งหมายถึง วิธีการเขียนโปรแกรมแบบออบเจ็กต์ โดยหลักการสำคัญของโอโอพี คือ การสร้างออบเจ็กต์ขึ้นมาแทนระบบซึ่งสามารถประมวลผลออบเจ็กต์นั้นเพื่อให้ได้ผลลัพธ์ออกมา

การเขียนโปรแกรมแบบออบเจ็กต์ คือ วิธีการเขียนโปรแกรมคอมพิวเตอร์ที่ใช้แนวความคิดในการจำลองวัตถุในโลกแห่งความเป็นจริงด้วยออบเจ็กต์ซึ่งทำให้ได้โครงสร้างของโปรแกรมเป็นระเบียบมากขึ้นและเอื้ออำนวยต่อการพัฒนาโปรแกรมในรุ่นต่อๆไป

2.2 คุณสมบัติของการโปรแกรมเชิงออบเจ็กต์

การโปรแกรมเชิงออบเจ็กต์มีแนวคิดพื้นฐานที่ควรทำความเข้าใจดังต่อไปนี้

2.2.1 เอกลักษณ์ (Abstraction)

แอบสแตร็คชั่นเป็นความสามารถในการสร้างเอกลักษณ์ของออบเจ็กต์ ตัวอย่างเช่น ออบเจ็กต์รถจะมีคุณสมบัติต่าง ๆ เช่น ความเร็วสูงสุด ความจุและสี ซึ่งจะแตกต่างจากออบเจ็กต์คนที่มีคุณสมบัติต่าง ๆ เช่น ความสูงและน้ำหนัก เป็นต้น

2.2.2 การซ่อนรายละเอียด (Encapsulation)

เอ็นแคปซูลเลชั่นเป็นการรวบรวมข้อมูลและโพรซีเยอร์หรือฟังก์ชั่นเข้าด้วยกันเป็นออบเจ็กต์ โดยรายระเอียดเกี่ยวกับข้อมูล ส่วนประกอบ โครงสร้างและการทำงานภายในของออบเจ็กต์จะถูกซ่อนอยู่จากผู้ใช้ (Client) โดยผู้ใช้จะเห็นเพียงอินเตอร์เฟสของออบเจ็กต์ซึ่งการทำงานภายในของออบเจ็กต์จะซ่อนอยู่ภายใต้อินเตอร์เฟส ตัวอย่างเช่น ในการพิมพ์งานจะมีคีย์บอร์ดเป็นอินเตอร์เฟสซึ่งใช้ในการสร้างตัวอักษรในโปรแกรมเวิร์ดโพรเซสเซอร์ ส่วนการทำงานภายในที่ทำการแปลงจากการกดคีย์บอร์ดไปเป็นตัวอักษรบนหน้าจอจะถูกซ่อนอยู่ภายใต้อินเตอร์เฟส

2.2.3 การสืบทอด (Inheritance)

การสืบทอดเป็นวิธีการในการสืบทอดจากคลาสต้นแบบ (Base class) ให้กับคลาสสืบทอด (Derived class) ที่จะถูกสร้างขึ้น โดยคลาสสืบทอดจะทำการสืบทอดทั้งคุณสมบัติ (Property) และคุณลักษณะ (Characteristic) ทุกอย่างจากคลาสต้นแบบ ในขณะเดียวกันคลาสสืบทอดก็สามารถเพิ่มคุณสมบัติและคุณลักษณะให้กับตัวเองได้ ตัวอย่างเช่น คลาสต้นแบบ คือ คลาสเครื่องเขียนและทำการสืบทอดคลาสไปเป็นคลาสสืบทอด คือ คลาสดินสอ คลาสปากกาลูกลื่นและคลาสปากกาหมึกซึม เป็นต้น

2.2.4 โพลิเมอร์ฟีซึ่ม (Polymorphism)

โพลิเมอร์ฟีซึ่มเป็นความสามารถของออบเจ็กต์ในการทำงานเมื่อได้รับคำสั่งเดียวกันจากโปรแกรม โดยออบเจ็กต์จะทำงานตามแบบของตัวเอง ตัวอย่างเช่น ออบเจ็กต์คนกับออบเจ็กต์สุนัขสามารถที่จะเดินได้แต่วิธีการเดินของทั้งสองออบเจ็กต์จะแตกต่างกัน เป็นต้น การที่ออบเจ็กต์จะสามารถทำงานได้ในลักษณะนี้ คลาสต้นแบบจะต้องประกาศเมธอด (Method) ที่จะใช้เป็นเวอร์ชวลเมธอด

2.2.4.1 เมธอดแบบสแตติก (Static Method)

เมธอดแบบสแตติกมีลักษณะการทำงานคล้ายกับการทำงานของโพรซีเยอร์และฟังก์ชั่นทั่วไปในภาษาปาสคาล กล่าวคือ เมื่อเมธอดเหล่านี้เรียกใช้เมธอดอื่น ก็จะมีการเรียกใช้เมธอดเดิมนั้นเสมอ ที่เป็นเช่นนี้ก็เพราะได้มีการผูกติดกันตั้งแต่ตอนคอมไพล์โปรแกรมซึ่งเรียกว่า เออร์ลีไบน์ดิงก์ (Early binding) ผลของเออร์ลีไบน์ดิงก์ก็คือ เมธอดที่ทำการเรียกเมธอดนั้นจากที่ใดและตำแหน่งการเรียกเมธอดนั้นจะคงที่ไม่สามารถเปลี่ยนแปลงได้

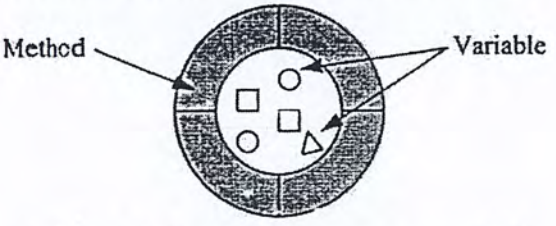
2.2.4.2 เมธอดแบบเวอร์ชวล (Virtual Method)

เมธอดแบบนี้จะไม่มีการผูกติดกับเมธอดอื่นตั้งแต่ตอนคอมไพล์โปรแกรม แต่จะมีการผูกติดกันตอนรันโปรแกรม ซึ่งเรียนกว่า เลตไบน์ดิงก์ (Late binding) ผลของเลตไบน์ดิงก์ก็คือ เมธอดที่ทำการเรียกเมธอดอื่นที่ไม่รู้ว่าจะเรียกเมธอดนั้นจากที่ไหนแต่จะรู้ในตอนรันโปรแกรม

2.3 ออบเจ็กต์ (Object)

ออบเจ็กต์คือสิ่งที่เกิดจากการรวมกันของข้อมูลและวิธีการจัดการกับข้อมูลนั้น ออบเจ็กต์เป็นส่วนสำคัญในการเขียนโปรแกรมแบบออบเจ็กต์ โดยตัวอย่างของออบเจ็กต์ในโลกแห่งความจริง เช่น คนเป็นสิ่งมีชีวิตที่มีคุณสมบัติ เช่น ความสูงและน้ำหนัก เป็นต้น และมีพฤติกรรม เช่น สามารถเดินได้และสามารถวิ่งได้ เป็นต้น สำหรับการโปรแกรมเชิงออบเจ็กต์นั้นเราสามารถที่จะแทนคุณสมบัติด้วยตัวแปร (Variable) และแทนพฤติกรรมด้วยเมธอด

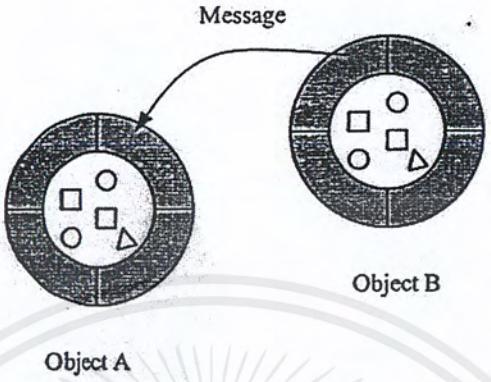
จากรูปที่ 2-1 เห็นได้ว่าตัวแปรของออบเจ็กต์ถูกจัดให้อยู่ตรงกลางของวงกลมออบเจ็กต์และมีเมธอดล้อมรอบเพื่อทำหน้าที่ซ่อนส่วนกลางของออบเจ็กต์ไว้ ประโยชน์ของการซ่อนรายละเอียดก็คือ ถ้าหากต้องการเปลี่ยนแปลงข้อมูลภายในออบเจ็กต์ ก็สามารถกระทำได้โดยไม่ส่งผลกระทบต่อออบเจ็กต์อื่นๆ และประโยชน์อีกข้อหนึ่ง ก็คือ สามารถจักการกับข้อมูลภายในออบเจ็กต์ได้ว่าข้อมูลส่วนไหนต้องการที่จะเปิดเผยหรือไม่ ซึ่งสามารถกำหนดได้โดยประกาศคุณสมบัติให้กับออบเจ็กต์



รูปที่ 2-1 การจำลองส่วนประกอบของออบเจ็กต์

2.4 ข่าวสาร (Massage)

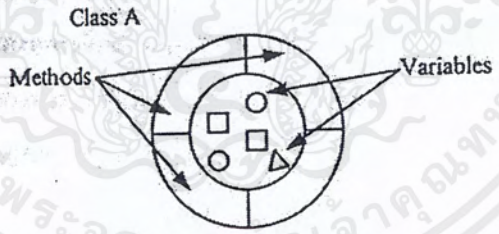
การติดต่อสื่อสารกันระหว่างออบเจ็กต์สามารถทำได้โดยการรับส่งข่าวสาร ตัวอย่างเช่น เมื่อออบเจ็กต์ A ต้องการทำงานกับออบเจ็กต์ B ออบเจ็กต์ A ก็จะส่งข่าวสารไปบอกแก่ออบเจ็กต์ B ดังรูปที่ 2-2



รูปที่ 2-2 การส่งข่าวสารระหว่างออบเจ็กต์

2.5 คลาสและอินสแตนซ์ (Class and Instance)

คลาสคือชนิดของออบเจ็กต์ซึ่งจะแสดงถึงคุณสมบัติและคุณลักษณะของออบเจ็กต์ คลาสจะถูกสร้างขึ้นโดยการเขียนรายละเอียดของคลาส ซึ่งคลาสหนึ่งสามารถที่จะเป็นต้นแบบให้แก่ออบเจ็กต์ได้หลายออบเจ็กต์ การที่เราสร้างออบเจ็กต์ขึ้นมานั้นเราเรียกว่าอินสแตนซ์ (Instance) โดยเราต้องทำการสร้างอินสแตนซ์ของคลาสขึ้นมาก่อนที่จะมีการใช้ตัวแปรหรือเมธอดของคลาสนั้น ซึ่งจะเรียกอินสแตนซ์ว่าเป็นออบเจ็กต์เมื่อมีการเปลี่ยนแปลงข้อมูลหรือเรียกใช้เมธอดของอินสแตนซ์นั้นเกิดขึ้น



รูปที่ 2-3 รายระเอียดของคลาส

2.6 ตัวอย่างการออกแบบออบเจ็กต์

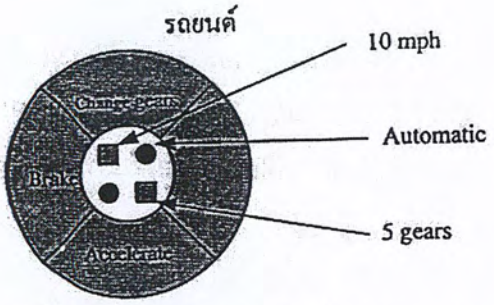
ในตัวอย่างนี้จะทำการออกแบออบเจ็กต์รถยนต์ที่มีการกำหนดตัวแปรดังต่อไปนี้

สถานะของรถยนต์

1. รถยนต์วิ่งด้วยความเร็ว 10 เมตรต่อวินาที
2. พวงมาลัยรถยนต์เป็นแบบออโตเมติก
3. มีเกียร์ทั้งหมด 5 เกียร์

เมธอดของรถยนต์

1. การเบรก
2. การเร่งความเร็ว
3. การเปลี่ยนเกียร์



รูปที่ 2-4 ไดอะแกรมของตัวอย่างการออกแบบ

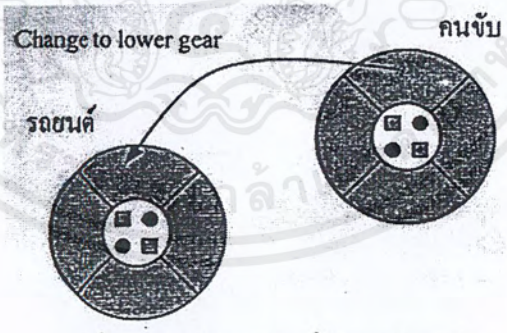
จากรูปที่ 2-4 ตัวแปรของออบเจ็กต์จะอยู่ตรงส่วนกลางของออบเจ็กต์ โดยเมธอดจะอยู่รอบ ๆ ตัวแปรของออบเจ็กต์และซ่อนตัวแปรของออบเจ็กต์จากออบเจ็กต์อื่นๆ เช่น เมื่อเราต้องการที่จะเปลี่ยนเกียร์ของรถยนต์ เราก็ไม่จำเป็นต้องรู้ระบบของเกียร์ทำงานอย่างไรบ้าง เพียงแต่รู้วิธีการที่จะเลื่อนคันโยกสำหรับเปลี่ยนเกียร์ก็พอแล้ว

เมสเซจของรถยนต์

ในการทำงานของออบเจ็กต์ ออบเจ็กต์จะต้องได้รับข้อมูลที่เพียงพอเพื่อที่ออบเจ็กต์จะรู้ว่าต้องทำอะไร จากตัวอย่างของการออกแบบ เมื่อต้องการเปลี่ยนเกียร์จะต้องทราบว่าเปลี่ยนเกียร์เป็นเกียร์อะไร โดยข้อมูลเหล่านี้จะถูกส่งไปกับเมสเซจเหมือนพารามิเตอร์

ส่วนประกอบของเมสเซจประกอบด้วย

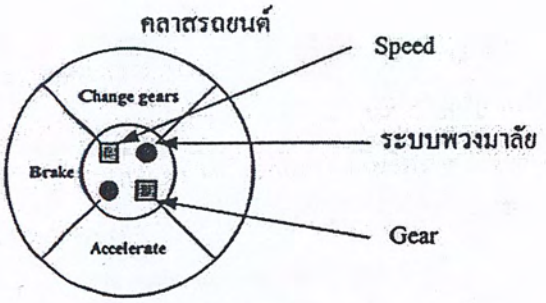
1. ออบเจ็กต์ที่จะส่งเมสเซจไปให้ (รถยนต์)
2. ชื่อของเมธอดที่จะทำ (เปลี่ยนเกียร์)
3. พารามิเตอร์ต่าง ๆ ที่เมธอดต้องใช้



รูปที่ 2-5 การส่งเมสเสจที่มีพารามิเตอร์

คลาสของรถยนต์

เราต้องสร้างคลาสของรถยนต์โดยประกาศตัวแปรเพื่อใช้เก็บสถานะต่าง ๆ เช่น ความเร็วของรถยนต์ สถานะของเกียร์ปัจจุบันและระบบของพวงมาลัย เป็นต้น และสร้างเมธอดต่าง ๆ ที่ใช้ในการทำงานของออบเจ็กต์ เช่น การเบรก การเร่งความเร็ว และการเปลี่ยนเกียร์ เป็นต้น

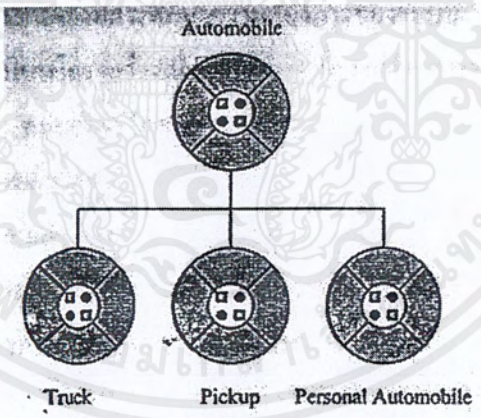


รูปที่ 2-6 คลาสไดอะแกรมของรถยนต์

ค่าของตัวแปรจะถูกกำหนดโดยอินสแตนซ์ของคลาส ดังนั้นหลังจากที่ทำการสร้างคลาสของรถยนต์แล้วจะต้องทำการสร้างอินสแตนซ์ของคลาสด้วย ถ้าทำการสร้างอินสแตนซ์ของคลาสแล้วตัวแปรที่ประกาศไว้โดยคลาสจะต้องเก็บไว้ในหน่วยความจำ จากนั้นจะสามารถใช้เมธอดเพื่อกำหนดค่าให้ตัวแปรได้

2.7 การสืบทอด

ในการเขียนโปรแกรมแบบออบเจ็กต์จะมีการสืบทอดคลาสอย่างเป็นลำดับชั้นและกำหนดให้คลาสหนึ่งเป็นคลาสต้นแบบของคลาสอื่นๆได้ เช่น รถกระบะ รถบรรทุกและรถยนต์ส่วนบุคคลเป็นคลาสที่ทำการสืบทอดมาจากคลาสรถยนต์ โดยจะเรียนคลาสทั้งสามแบบเป็นสับคลาส (Sub Class) และเรียกคลาสรถยนต์ว่าเป็นซุปเปอร์คลาส (Super class)



รูปที่ 2-7 การสืบทอดคลาสรถยนต์

บทที่ 3

Component Object Model (COM)

3.1 ซอฟต์แวร์คอมโพเนนท์

เป็นกลุ่มของ คลาสซึ่งอยู่ร่วมกันและอยู่ในรูปแบบที่ใช้งานได้โดยไม่ต้องทำการคอมไพล์ใหม่ (อาจจะเป็นไบนารี หรือเป็น ไฟล์ Dll) ติดต่อกับระบบภายนอกโดยทางอินเตอร์เฟส ซึ่งมีลักษณะคล้าย แอปแทรคคลาส (Abstract Class) ซึ่งเป็นคลาสที่ถูกประกาศเพียงแต่เมธอดไม่มีการอิมพลีเมนต์ เมธอดภายใน และไม่มีการประกาศคุณสมบัติ (Property) ต่างๆ ของคลาสนั้นด้วย

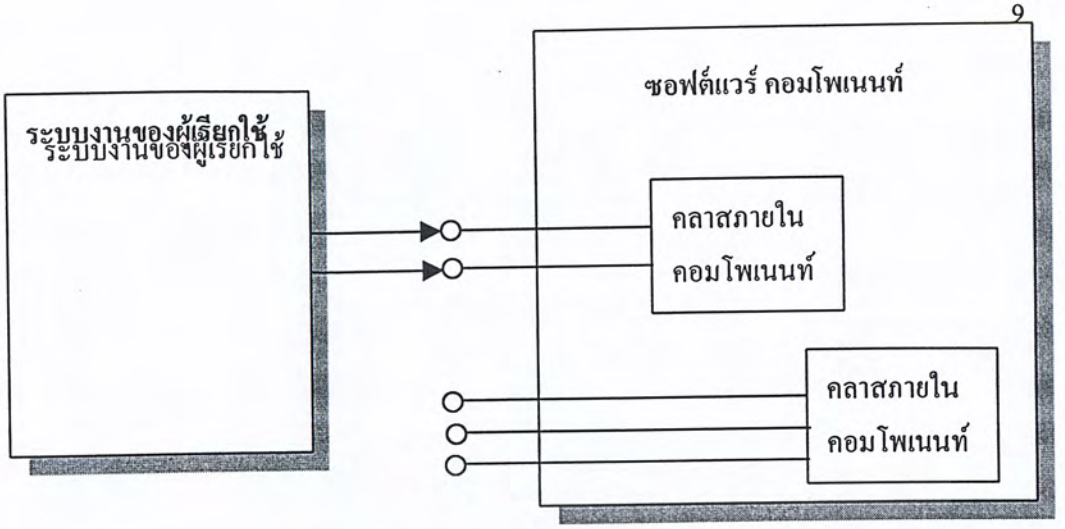
สำหรับ COM แล้ว อินเตอร์เฟสนั้นเป็นองค์ประกอบสำคัญในการสร้างความเป็นอิสระระหว่างผู้เรียกใช้กับคอมโพเนนท์ ตัวอินเตอร์เฟสนั้นจะถูกประกาศโดยใช้ภาษาเป็นมาตรฐานในการสร้างอินเทอร์ ซึ่งเรียกว่า Interface Definition Language (IDL) ซึ่งทำให้การพัฒนา COM คอมโพเนนท์นั้นสามารถพัฒนาจากภาษาโปรแกรมภาษาใด ๆ ก็ได้ อย่างไรก็ดีเนื่องจากโครงสร้างพื้นฐานของ COM คอมโพเนนท์ที่จำเป็นต้องมีการเรียกใช้ผ่านทางพอยต์เตอร์ ดังนั้นภาษาที่จะสร้าง และเรียกใช้งานก็ควรจะมีความสามารถที่จะเรียกใช้พอยต์เตอร์เหล่านั้นได้ ซึ่งไม่ได้หมายความว่าภาษาเหล่านั้นจะต้องมีการเรียกใช้ตัวแปรพอยต์เตอร์ได้โดยตรง เช่น Visual Basic นั้นไม่มีความสามารถที่จะเรียกใช้ตัวแปร พอยต์เตอร์ได้โดยตรง แต่ก็สามารถที่จะสร้างและเรียกใช้ COM คอมโพเนนท์ ได้ เนื่องจากตัวภาษานั้นสามารถเรียกใช้ได้เพียงแต่ ไม่สามารถเรียกใช้ได้โดยผ่านทางตัวแปรพอยต์เตอร์เท่านั้น

ในการพัฒนาคอมโพเนนท์ ก็จะให้คลาสภายในคอมโพเนนท์นั้นทำการอิมพลีเมนต์ แอปแทรคคลาสหรือ อินเตอร์เฟสนั้น และฝ่ายผู้ใช้คอมโพเนนท์ นั้นก็จะติดต่อกับคลาสภายในคอมโพเนนท์ผ่านทางอินเตอร์เฟส ดังนั้นเมื่อมีการเปลี่ยนแปลงลักษณะภายในของคลาสในคอมโพเนนท์ก็จะไม่มีผลกระทบโดยตรงกับผู้เรียกใช้ เนื่องจาก การเรียกใช้คอมโพเนนท์ก็ยังเรียกผ่านทางอินเทอร์เฟสที่ยังไม่เปลี่ยนแปลง ซึ่งก็เป็นเหตุผลสำคัญที่ทำให้การกำหนดอินเตอร์เฟสนั้นไม่ควรมีการเปลี่ยนแปลง (immutable)

เมื่อผู้เรียกใช้ต้องการเปลี่ยนคอมโพเนนท์ จากตัวเดิมที่ใช้อยู่ไปเป็นตัวใหม่ก็สามารถเปลี่ยนได้โดยไม่ต้องแก้ซอร์สโค้ด เพียงแต่คอมโพเนนท์ตัวใหม่นั้นจะต้องมีอินเตอร์เฟสเหมือนคอมโพเนนท์ตัวเก่าเท่านั้น ซึ่งทำให้การพัฒนาโปรแกรมนั้นเป็นอิสระต่อกันมากกว่าการพัฒนาโปรแกรมแบบเดิม

แต่ละคลาสนั้นสามารถอิมพลีเมนต์อินเตอร์เฟสได้มากกว่าหนึ่งอินเตอร์เฟส ซึ่งจะแตกต่างจากที่คลาสต่าง ๆ นั้นสามารถสืบทอดคุณสมบัติ ได้จากคลาสใดคลาสหนึ่ง เนื่องจากหลักการ เขียนโปรแกรมเชิงคอมโพเนนท์นั้นไม่สนับสนุนการสืบทอดคุณสมบัติมากกว่าหนึ่ง (Multiple Inheritance) แต่สนับสนุนหลักการการมีหลายอินเตอร์เฟส (Multiple Interface)

จากที่ได้กล่าวมาเบื้องต้น ลักษณะอย่างคร่าว ๆ ของซอฟต์แวร์คอมโพเนนท์จะมีลักษณะดังรูปที่



รูปที่ 3-1 แสดงลักษณะของซอฟต์แวร์คอมโพเนนท์

3.2 COM คอมโพเนนท์

COM เป็นซอฟต์แวร์คอมโพเนนท์โมเดลที่เป็นอิสระทางภาษาไม่ขึ้นกับภาษาที่ใช้พัฒนา เราสามารถสร้างออบเจ็กต์ (Object) แล้วใช้โดยภาษาอื่นๆ ได้หลายภาษา สามารถใช้ภายในเครื่องเดียวกันหรือใช้ผ่านเครือข่ายมาจากเครื่องอื่นก็ได้

ลักษณะที่เห็นได้ชัดของ COM ก็คือการสื่อสารระหว่างคอมโพเนนท์, ระหว่างแอปพลิเคชั่น และระหว่างไคลเอนต์กับเซิร์ฟเวอร์ ผ่านทางอินเตอร์เฟส (Interface) แอปพลิเคชั่นต่างๆ สามารถเข้าถึง COM คอมโพเนนท์และอินเตอร์เฟสของมันที่มีได้ในเครื่องคอมพิวเตอร์เครื่องเดียวกันหรือเครื่องคอมพิวเตอร์ที่มีอยู่ในเครือข่ายได้ ในกรณีที่ใช้ผ่านเครือข่ายนั้นจะใช้ Distribute COM (DCOM)

ในการใช้งาน COM มีส่วนประกอบอยู่ 3 ส่วนคือ

1. COM เซิร์ฟเวอร์ เป็นตัวให้บริการ ในการทำงานไคลเอนต์จะเรียกของการทำงานจาก COM เซิร์ฟเวอร์โดยผ่านทางอินเตอร์เฟส
2. COM อินเตอร์เฟส เป็นตัวกลางในการติดต่อระหว่างไคลเอนต์ และเซิร์ฟเวอร์
3. COM ไคลเอนต์ เป็นตัวเรียกใช้การทำงานของ COM เซิร์ฟเวอร์

3.2.1 COM เซิร์ฟเวอร์

COM เซิร์ฟเวอร์ เป็นแอปพลิเคชั่น หรือไลบรารี (Library) ที่เตรียมบริการให้แก่ไคลเอนต์ ใน COM เซิร์ฟเวอร์หนึ่งตัวประกอบไปด้วยหนึ่ง หรือหลายๆ COM ออบเจ็กต์

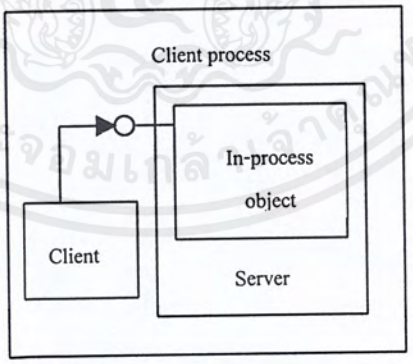
ไคลเอนต์ไม่จำเป็นต้องรู้วิธีในการให้บริการ COM ออบเจ็กต์ และไม่ต้องรู้ว่า COM ออบเจ็กต์อยู่ที่ใด ตัว COM เองจะจัดการในการเข้าถึงออบเจ็กต์

เมื่อไคลเอนต์ร้องขอบริการจาก COM ออบเจ็กต์ ตัวไคลเอนต์จะส่ง Class identifier (CLSID) ไปยัง COM จากนั้น COM จะใช้ CLSID ในการเตรียมที่ที่เหมาะสมในการนำโค้ด (Code) เข้ามาไว้ในหน่วยความจำ แล้วทำการกำหนดค่าเริ่มต้นให้แก่ออบเจ็กต์อินสแตนซ์เพื่อไคลเอนต์ในการนำไปใช้ ส่วนตัว

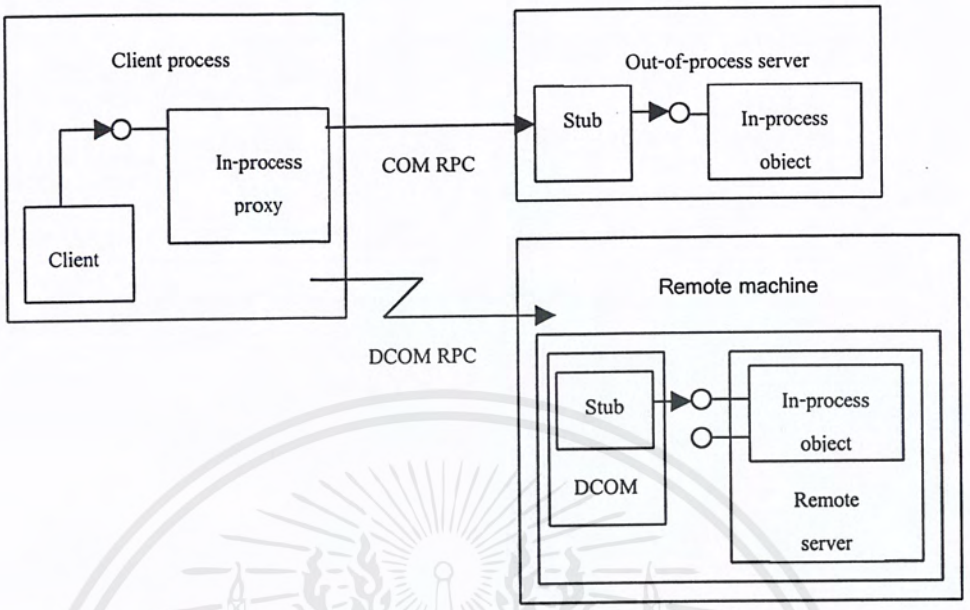
COM เซิร์ฟเวอร์ต้องทำการลงทะเบียน (Register) ด้วย CLSID เข้าในระบบ แล้วเตรียม Class factory ออบเจ็กต์ (IClassFactory) เพื่อการสร้างอินสแตนซ์ที่ต้องการตาม CLSID และเตรียมวิธีในการนำอินสแตนซ์ออกจากหน่วยความจำเมื่อไคลเอนต์ไม่ใช้อินสแตนซ์นั้นแล้ว

ประเภทของ COM เซิร์ฟเวอร์มี 3 แบบดังนี้

- **In-process server** เป็นไลบรารี (DLL) จะรัน (Run) ภายในโพรเซส (Process) เดียวกันกับไคลเอนต์ เช่น แอกทีฟเอ็กซ์คอนโทรล (ActiveX control) ที่ฝังในเว็บเพจที่สามารถแสดงผลทางอินเตอร์เน็ตเอ็กซ์โพลเรอร์ (Internet explorer) หรือ เน็ตส์เคป (Netscape) โดยแอกทีฟเอ็กซ์คอนโทรลจะถูกดาวน์โหลดมาไว้ที่ไคลเอนต์และทำงานในโพรเซสเดียวกันกับเว็บบราวเซอร์ ในการติดต่อของไคลเอนต์กับ In-process server จะเรียกใช้ไปโดยตรงไปยัง COM อินเตอร์เฟส
- **Out-of-process server** หรือ Local server เป็นแอปพลิเคชั่น (EXE) ทำการรันในคนละโพรเซส แต่อยู่ในเครื่องเดียวกัน เช่น เวิร์ด มี เอ็กเซล ฝังอยู่ภายในทั้งสองแอปพลิเคชั่นรันอยู่ในเครื่องเดียวกันแต่คนละโพรเซส Out-of-process server จะใช้ COM ในการติดต่อกับไคลเอนต์
- **Remote server** เป็นไลบรารี หรือแอปพลิเคชั่นก็ได้ รันอยู่บนคนละเครื่องกับไคลเอนต์ เช่น ในการติดต่อของ ดาตาเบสแอปพลิเคชั่น กับ แอปพลิเคชั่นเซิร์ฟเวอร์ที่อยู่กันคนละเครื่องในเครือข่าย Remote server จะใช้ Distribute COM (DCOM) ในการติดต่อกับไคลเอนต์



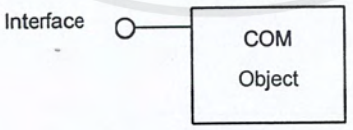
รูปที่ 3-2 แสดง In-process server



รูปที่ 3-3 แสดง Out-of-process server และ Remote server

ทั้งในแบบ Out-of-Process Server และ Remote Server จะต้องมีวิธีการพิเศษซึ่งเรียกว่า RPC (Remote Procedure Call) ซึ่งใช้ในการเรียกใช้โพรซีเยอร์ โดยผ่านทางพรอกซี ซึ่งอยู่ภายนอกโพรเซส แต่เป็น in–process ในมุมมองของ ไคลเอนต์ ความแตกต่างระหว่างตำแหน่งทั้งสองคือ ใน Out-of -Process นั้นจะใช้ COM เป็นตัวกลางในการติดต่อ ส่วน ในแบบ Remote machine จะใช้ DCOM (Distributed COM) เป็นตัวส่งผ่าน RPC นั้นให้กับตัวเซิร์ฟเวอร์

3.2.2 COM อินเตอร์เฟส



รูปที่ 3-4 แสดง ลักษณะของ COM

COM ไคลเอนต์สื่อสารกับออบเจ็กต์ผ่านทาง COM อินเตอร์เฟส ซึ่งเป็นกลุ่มของรูทีนเอจ (Routine) ที่ช่วยในการติดต่อระหว่างผู้ให้บริการ (Server Object) และผู้ขอใช้บริการ (Client) ภายในอินเตอร์เฟส จะไม่มีโค้ด แต่จะมีเพียง รายเอียดอย่างคร่าวๆ ของ แต่ละโพรเซส ซึ่งได้แก่ ชื่อ เมธอด และออบเจ็กต์ แต่ละตัวสามารถมี อินเตอร์เฟส ได้หลายตัวซึ่ง ทุกตัวต้องประกอบด้วย อินเตอร์เฟส อย่างน้อย 2 ตัว หนึ่งคือ อิน

เตอร์เฟส พื้นฐานซึ่งเป็นตัวที่ ออบเจ็กต์ทุกตัวต้องมีคือ IUnknown ซึ่งมี เมธอด พื้นฐานคือ QueryInterface (), AddRef () และ Release( ) และอินเตอร์เฟส ที่เหลือคือ อินเตอร์เฟสซึ่งผู้พัฒนาพัฒนาให้อยู่ในรูปแบบซึ่งตัวเองต้องการ อินเตอร์เฟส เหล่านี้จะสืบทอดมาจาก IUnknown

โดยอินเตอร์เฟสมีลักษณะดังนี้

- สร้างขึ้นมาแล้วจะแก้ไขหรือเปลี่ยนอินเตอร์เฟสไม่ได้ ในแต่ละอินเตอร์เฟสจะเตรียมเซตของฟังก์ชันให้เราเรียกใช้ได้ การจะเพิ่มฟังก์ชันทำได้โดยการเพิ่มอินเตอร์เฟสเข้าไปใหม่
- ชื่อของ COM อินเตอร์เฟสจะขึ้นต้นด้วย " I " ตัวใหญ่แล้วตามด้วยชื่อของอินเตอร์เฟสนั้น
- ในแต่ละอินเตอร์เฟสนั้นจะมีหมายเลขที่รับรองว่าไม่มีการซ้ำเรียกว่า Globally Unique Identifier (GUID) มีขนาด 128 บิต หรือ 32 ไบต์
- อินเตอร์เฟสเป็นอิสระทางภาษาคือจะใช้ภาษาใดก็ได้ในการสร้างแต่ภาษาที่ใช้ต้องสนับสนุนการใช้พอยน์เตอร์และสามาเรียกฟังก์ชันผ่านทางพอยน์เตอร์ได้
- อินเตอร์เฟสไม่ใช่ออบเจ็กต์ แต่เป็นทางที่จะเข้าถึงออบเจ็กต์ได้ ตัวไคลเอนต์ไม่สามารถเข้าถึงข้อมูลได้โดยตรงแต่จะเข้าถึงผ่านอินเตอร์เฟสพอยน์เตอร์
- อินเตอร์เฟสต้องได้รับการสืบทอดมาจาก IUknown เสมอ

3.2.1.1 ลักษณะของ IUnknown

ทุกๆ COM ออบเจ็กต์ต้องมีอินเตอร์เฟสพื้นฐานที่ชื่อ IUnknown ซึ่งภายในมีเมธอดดังนี้

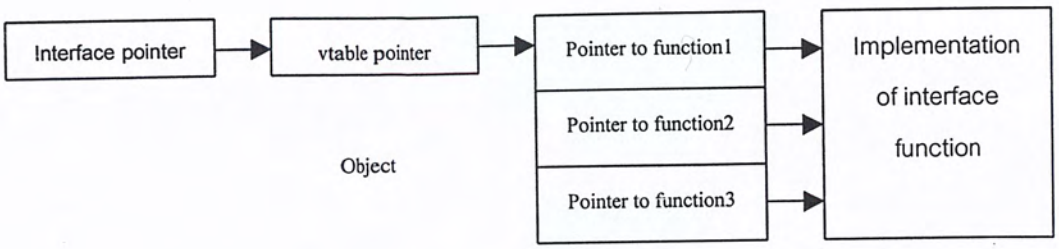
- **QueryInterface** ทำหน้าที่เตรียมพอยน์เตอร์ชี้ไปยังอินเตอร์เฟสที่ออปเจ็คนั้นสนับสนุนอยู่
- **AddRef และ Release** ทำหน้าที่ในการนับโดยจะเก็บช่วงชีวิตของออบเจ็กต์แล้วคอยดูว่าออบเจ็กต์ตัวใดสามารถถูกลบออกไปได้ถ้าไม่ได้ถูกใช้เป็นเวลานานๆ

ในการทำงานไคลเอนต์ได้รับพอยน์เตอร์ไปยังอินเตอร์เฟสต่างๆ ผ่าน IUnknown โดยใช้เมธอด QueryInterface ตัว QueryInterface นั้นจะรู้ข้อมูลเกี่ยวกับทุกๆ อินเตอร์เฟสในเซิร์ฟเวอร์แล้วจะให้พอยน์เตอร์แก่ไคลเอนต์ในการเรียกใช้เมธอดจากอินเตอร์เฟส

ในการนับช่วงชีวิตของออบเจ็กต์ก็จะใช้ AddRef และ Release ในการนับออบเจ็กต์ที่มีค่าอ้างอิงไม่เป็นศูนย์จะอยู่ในหน่วยความจำและถ้าค่าอ้างอิงเป็นศูนย์เมื่อไหร่ออบเจ็กต์นั้นก็จะถูกนำออกไป

พอยน์เตอร์ของอินเตอร์เฟสจะมีขนาด 32 บิต ชี้ไปยังอินสแตนซ์ (Instance) ของออบเจ็กต์ เพื่อการใช้เมธอดที่มีในอินเตอร์เฟส โดยการทำงานจะใช้อาเรย์ของพอยน์เตอร์ชี้ไปยังเมธอด เรียกว่า vtable

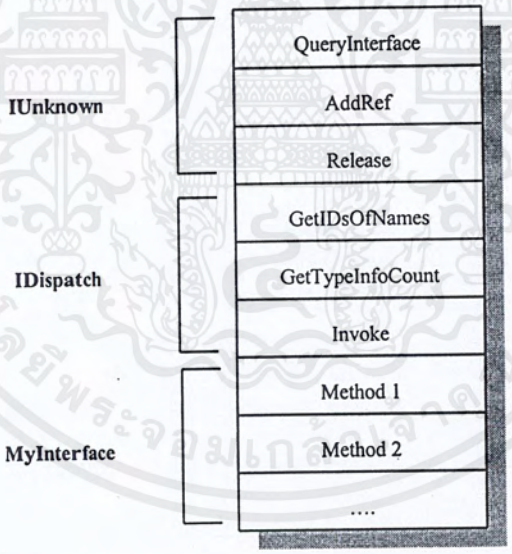
vtable จะถูกใช้ร่วมกันโดยทุกๆ อินสแตนซ์ของออบเจ็กต์คลาส และพอยน์เตอร์ของอินเตอร์เฟสจะชี้ไปยัง vtable ดังรูปที่ 3-5



รูปที่ 3-5 แสดง ลักษณะของ vtable

3.2.3 ดูอัลอินเตอร์เฟส (Dual Interface)

เป็นอีกรูปแบบหนึ่งของการเรียกใช้อินเตอร์เฟส สำหรับโดยคอมโพเนนท์ที่จะมีลักษณะเป็น Dual Interface นี้จะเป็นอินเตอร์เฟสที่มีอินเตอร์เฟส IDispatch เพิ่มขึ้นมาอีกอินเตอร์เฟสหนึ่งอินเตอร์เฟส ดังรูปที่ 3-6 จะแสดงลักษณะอินเตอร์เฟส IMyInterface ซึ่งมีลักษณะเป็น Dual Interface ของ COM คอมโพเนนท์หนึ่ง จะเป็นว่า 3 เมธอดแรกนั้นเป็นเมธอดของอินเตอร์เฟส IUnknown ส่วนอีก 4 เมธอดถัดไปนั้นเป็นของอินเตอร์เฟส IDispatch หลังจากนั้นจึงเป็นเมธอดของอินเตอร์เฟส IMyInterface



รูปที่ 3-6 แสดงลักษณะของอินเตอร์เฟสที่เป็น ดูอัลอินเตอร์เฟส

ในการเรียกใช้งาน COM คอมโพเนนท์ประเภทนี้นั้นจะสามารถเรียกใช้งานได้ใน 2 รูปแบบเนื่องจากอินเตอร์เฟสที่เป็นดูอัลอินเตอร์เฟสนั้นจะมีลักษณะทั้งสองอย่างอยู่ในตัวมัน ซึ่งได้แก่

3.2.3.1)เรียกใช้งานแบบเออลีบาย (Early Binding) ที่เรียกใช้ตอนคอมไพล์ไทม์โดยผ่านการเรียกไปยังเมธอดของ COM คอมโพเนนท์โดยตรงผ่านทาง VTable หรือคัสตอมอินเตอร์เฟส (Custom Interface) ซึ่งเป็นลักษณะพื้นฐานของ COM คอมโพเนนท์ทั่วไปซึ่งมีข้อดีกว่าการเรียกอีกรูปแบบหนึ่งดังนี้

- การเรียกใช้ผ่าน VTable นั้นจะมีการตรวจสอบชื่อเมธอด และพารามิเตอร์ซึ่งทำให้สามารถควบคุมการเกิดความผิดพลาดเบื้องต้นได้
- เข้าถึง COM คอมโพเนนท์ได้เร็วกว่าสำหรับเซิร์ฟเวอร์ชนิดอินโพรเซสเซิร์ฟเวอร์นั้น

นอกจากสำหรับ COM คอมโพเนนท์ทั่วไปหากไม่สนับสนุนอินเตอร์เฟส IDispatch การเรียกใช้งานจะเรียกโดยใช้วิธีโดยอัตโนมัติ

3.2.3.2)เรียกใช้งานแบบเลทบาย (Late Binding) ซึ่งจะทำการเรียกใช้ตอนรันไทม์โดยผ่านทางอินเตอร์เฟส IDispatch ในการเรียกใช้ผู้เรียกใช้ต้องทำการสร้างออบเจ็กต์ที่เรียกอินเตอร์เฟส IUnknown เสียก่อนหลังจากนั้นจึงเรียกไปยังอินเตอร์เฟส IDispatch อีกที IDispatch จะเก็บคุณสมบัติและเมธอดต่าง ๆ ของมันไว้ภายในโดยใช้ ตัวบ่งชี้ dispatch (dispID) ซึ่งเป็น GUID การจะเข้าถึงชื่อของเมธอด และคุณสมบัติต่าง ๆของอินเตอร์เหล่านั้นได้โดยผ่านตัวบ่งชี้ ซึ่งจะถูกสร้างเมื่ออยู่ในขณะรันไทม์ ร่วมกับเมธอด GetIDsOfNames ส่วนการเรียกใช้จะใช้เมธอด Invoke ข้อดีของการใช้วิธีนี้ก็คือตัวไคล์เอนต์จะเป็นอิสระต่อตัวคอมโพเนนท์เนื่องจาก จะเรียกใช้จริง ๆ ตอนรันไทม์เท่านั้น ในการทำงานจริงหากมีการเปลี่ยนแปลง ตัวคอมโพเนนท์จึงไม่จำเป็นต้องคอมไพล์ตัวไคลเอนต์ใหม่ (อย่างไรก็ดีต้องไม่มีการเปลี่ยนแปลงอินเตอร์เฟสเก่า แต่สามารถเพิ่มเมธอดใหม่ เข้าไปในคอมโพเนนท์ใหม่ได้) นอกจากนั้นการเรียกใช้ COM คอมโพเนนท์ในผลิตภัณฑ์ Microsoft Visual Basic จะเป็นการเรียกโดยวิธีนี้เท่านั้น

3.2.4 โคคลาส (Coclasses) และ คลาสแฟคทอรี่ (Class factory)

COM ออบเจ็กต์ เป็นอินสแตนซ์ของโคคลาส ตัวโคคลาสเองจะมีออบเจ็กต์ที่เรียกคลาสแฟคทอรี่ เมื่อไรที่มีการร้องขอบริการจากไคลเอนต์ตัวคลาสแฟคทอรี่ จะสร้าง และลงทะเบียนออบเจ็กต์อินสแตนซ์ให้กับไคลเอนต์ที่ทำการร้องขอมา และถ้ามีไคลเอนต์ตัวอื่นต้องการบริการอีกคลาสแฟคทอรี่ ก็จะทำการสร้างให้อีก

3.2.5 มาร์แชลลิ่งแมคนานิซึม (Marshaling mechanism)

เป็นวิธีการที่ไคลเอนต์ทำการเรียกไปยังรีโมตออบเจ็กต์ในโพรเซสอื่น หรือบนคอมพิวเตอร์คนละเครื่องวิธีการมาร์แชลิงมีดังนี้

- ใช้อินเตอร์เฟสพอยน์เตอร์ในโพรเซสของเซิร์ฟเวอร์ และสร้างพรอกซีพอยน์เตอร์ชี้ไปยังโค้ดในไคลเอนต์โพรเซส
- ส่งอาร์กิวเมนต์ (Argument) ของการเรียกอินเตอร์เฟสจากไคลเอนต์ และวางอาร์กิวเมนต์ลงในโพรเซสของรีโมตออบเจ็กต์

สำหรับการเรียกอินเตอร์เฟส ตัวไคลเอนต์จะทำการวาง (Push) อาร์กิวเมนต์ลงในสแตก และทำการเรียกฟังก์ชันผ่านทางอินเตอร์เฟสพอยน์เตอร์ ถ้าการเรียกออบเจ็กต์ไม่เป็นแบบ In-process server การเรียกจะ

ผ่านไปยังพรอกซี ตัวพรอกซีจะทำการห่อ (Packs) อาร์กิวเมนต์ให้เป็น มาร์แชลิงแพ็กเกจ แล้วทำการส่งไปยังรีโมตออบเจ็กต์ หลังจากนั้น Stub จะทำการแก้ห่อ (Unpacks) แล้ววางอาร์กิวเมนต์ลงในสแตกแล้วจึงเรียกใช้ออบเจ็กต์

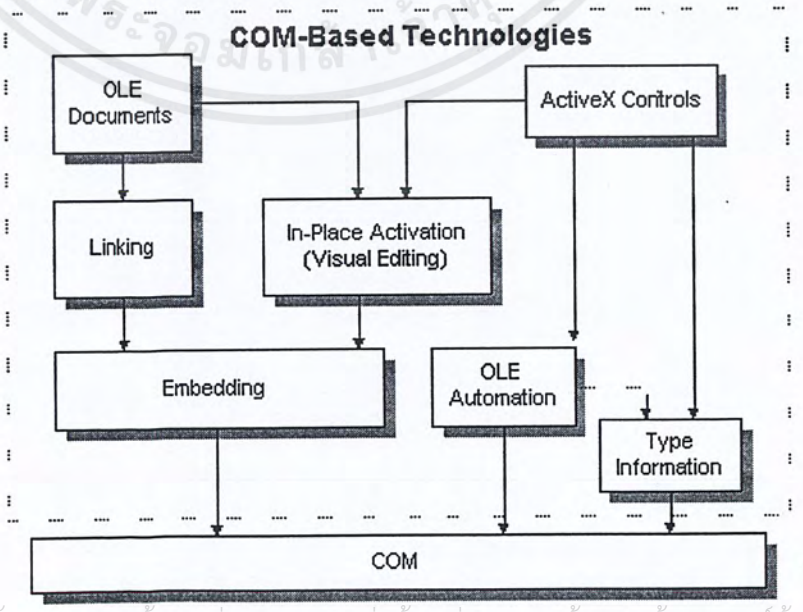
3.3 COM ไคลเอนต์

เป็นผู้ขอการบริการจาก COM เซิร์ฟเวอร์ ผ่านทาง COM อินเตอร์เฟส นับเป็นส่วนที่สำคัญส่วนหนึ่งในการออกแบบ COM แอปพลิเคชั่น สิ่งทีสำคัญของส่วนนี้คือ ไคลเอนต์นั้นไม่ต้องการรู้ว่า COM เซิร์ฟเวอร์ นั้นทำงานอย่างไร และอยู่ที่ไหนขั้นตอนเหล่านั้น ทั้งอินเตอร์เฟส และ เซิร์ฟเวอร์ต้องปกปิดเพื่อให้ไคลเอนต์ นั้นสะดวกในการนำไปใช้

3.4 ส่วยขยาย COM

COM ได้ออกแบบให้มีการติดต่อกันของฟังก์ชัน และขยายการติดต่อกับฟังก์ชันออกไปในวงกว้าง ตัว COM เองได้มีการขยายออกไปเฉพาะทางตามความต้องการในการใช้งาน เช่น อย่างแรกที่ต้องพูดถึงคือเทคโนโลยี OLE และบริการหลายๆ อย่างของวินโดวส์

- **ออโตเมชั่น เซิร์ฟเวอร์** เป็นความสามารถของแอปพลิเคชั่นในการควบคุมออบเจ็กต์ในแอปพลิเคชั่นตัวอื่น ในทางโปรแกรมมิ่งพูดได้ว่า ออโตเมชั่น เซิร์ฟเวอร์ เป็นออบเจ็กต์ที่สามารถควบคุมโดยเอ็กซีคิวเทเบิ้ล (EXE) ในเวลารัน
- **ออโตเมชั่น คอนโทรลเลอร์ (COM clients)** เป็นไคลเอนต์ของ ออโตเมชั่น เซิร์ฟเวอร์ ผู้ใช้สามารถเขียนสคริปต์ทำการควบคุม ออโตเมชั่น เซิร์ฟเวอร์ ได้
- **ActiveX controls** เป็นแบบเฉพาะของ In-process COM เซิร์ฟเวอร์ โดยจะฝังในแอปพลิเคชั่นของไคลเอนต์
- **Type libraries** เป็นโครงสร้างข้อมูลแบบหนึ่งใช้เก็บทรัพยากร (Resource) บอกรายละเอียดของรูปแบบเกี่ยวกับออบเจ็กต์และอินเตอร์เฟสต่างๆ



รูปที่ 3-7 แสดง แสดงส่วนขยาย COM

- **Active Server Pages** เป็น ActiveX คอมโพเนนท์ในการสร้าง ไดนามิคเว็บเพจ
- **Active Document** เป็นออบเจ็กต์ที่สนับสนุน Linking และ Embedding , Drag และ Drop , Visual editing ตัวอย่างเช่น ผลิตภัณฑ์ไมโครซอฟต์เวิร์ด และ ไมโครซอฟท์เอ็กเซล
- **Visual Cross-Process Object** เป็นออบเจ็กต์ซึ่งสามารถถ่ายโอนข้ามโพรเซส

3.4.1 Automation sever and controller

ออโตเมชั่น อ้างถึงความสามารถของแอปพลิเคชั่นในการควบคุมแอปพลิเคชั่นอื่น เหมือนกับมาโครที่สามารถใช้กับแอปพลิเคชั่นหลายๆ แอปพลิเคชั่นในเวลาเดียวกัน ออโตเมชั่น สามารถใช้ได้ทั้งบน in-process, local และ remote เซิร์ฟเวอร์

ออโตเมชั่น มีลักษณะที่เด่นๆ 2จุดคือ

- ออโตเมชั่น ออบเจ็กต์ต้องสามารถกำหนดเซตของคุณสมบัติและ คำสั่ง แล้วต้องสามารถเตรียมข้อมูลเกี่ยวกับอินเตอร์เฟสของออบเจ็กต์เช่น เมธอดของอินเตอร์เฟส และ อาร์กิวเมนต์ของออบเจ็กต์ ธรรมดาแล้วจะสามารถหาข้อมูลได้ในไทพ์ไลบรารี ตัว ออโตเมชั่น เซิร์ฟเวอร์ สามารถสร้างข้อมูลของรูปแบบได้โดยอัตโนมัติ
- ออโตเมชั่น ออบเจ็กต์ต้องสร้างเมธอดที่สามารถให้แอปพลิเคชั่นอื่นๆ มาใช้ได้ ในกรณีนี้คือต้องมี IDispatch เราสามารถดูเมธอดทั้งหมดและ คุณสมบัติได้ผ่านทางอินเตอร์เฟสนี้

ผู้พัฒนาจะใช้ ออโตเมชั่น ในการสร้างและใช้ OLE ออบเจ็กต์ที่รันได้ในทุกๆ โพรเซสเพราะอินเตอร์เฟส IDisplatch สามารถใช้มาร์แชร์ลิง

3.4.2 ActiveX controls

ActiveX controls เป็นตัวคอนโทรลที่มองเห็นได้สามารถรันใน in-process เซิร์ฟเวอร์ ได้อย่างเดียว และสามารถต่อกับแอปพลิเคชั่นที่เป็นคอนเทนเนอร์ ของ ActiveX control ตัวมันเองนั้นไม่ได้เป็นแอปพลิเคชั่นแต่เป็นไลบรารีที่สามารถถูกเรียกใช้ได้ในแอปพลิเคชั่นต่างๆ และสามารถถูกเรียกใช้งานจากหลายๆ แอปพลิเคชั่น พร้อม ๆ กันได้ งานที่ใช้ ActiveX control กันมากก็คืองานประเภทเว็บไซต์ ต่าง ๆ

3.4.3 Type libraries

เป็นทางที่จะรู้ถึง ข้อมูลเกี่ยวกับรูปแบบ (Type information) ของออบเจ็กต์ ข้อมูลเกี่ยวกับรูปแบบจะเก็บอยู่ใน Type libraries ในนั้นจะมีข้อมูลเกี่ยวกับออบเจ็กต์ และอินเตอร์เฟส เช่น มีอินเตอร์เฟสอะไรบ้างในออบเจ็กต์ มีฟังก์ชันอะไรบ้างในอินเตอร์เฟส และบอกถึงชนิดและจำนวนอาร์กิวเมนต์อะไรบ้างที่จะต้องส่งให้ฟังก์ชันรวมถึง หมายเลขที่ไม่ซ้ำสำหรับ Coclasses (CLSID) และ IIDs, รูทีนที่ส่งออกมาโดย ออโตเมชั่น หรือ ActiveX เซิร์ฟเวอร์ แต่ไม่ใช่เมธอดของอินเตอร์เฟส, ข้อมูลเกี่ยวกับ enumeration, record, union, alias

การสร้าง Type libraries สามารถสร้างโดยการใช้ทูล (Tool) โดยเขียนสคริปต์ใน Interface Definition Language (IDL) หรือ Object Description Language (ODL) แล้วรันสคริปต์ผ่านคอมไพเลอร์ แต่อาจใช้ Type Library editor มาช่วยในการสร้างก็ได้ทำให้ง่ายขึ้น

Type libraries เป็นสิ่งที่สำคัญสำหรับแต่ละเซตของออบเจ็กต์ในการทำให้ผู้ใช้เข้าใจ ตัวอย่างความสำคัญที่ต้องใช้ Type libraries เช่น ActiveX controls ต้องการ Type libraries ที่ต้องรวมทรัพยากรใน DLL ที่เก็บ ActiveX controls, ออบเจ็กต์ที่สนับสนุน vtable ของอินเตอร์เฟสที่สร้างขึ้นต้องอธิบายใน Type libraries เพราะ vtable จะต้องอ้างถึงในเวลาคอมไพล์, แอปพลิเคชั่นที่ใช้ ออโตเมชั่น เซิร์ฟเวอร์ ต้องมี Type libraries สำหรับไคลเอนต์ ฯลฯ

ข้อได้เปรียบในการใช้ไทพ์ไลบรารีคือ ช่วยตรวจสอบรูปแบบขณะคอมไพล์, สามารถดูไลบรารีได้เมื่อไคลเอนต์ต้องการรู้ลักษณะของออบเจ็กต์ ฯลฯ

3.4.4 Active Server Page (ASP)

เป็นเทคโนโลยีที่ใช้ในการสร้างเว็บเพจโดยใช้คอมโพเนนท์ของ ActiveX เซิร์ฟเวอร์ ใน Active server page เราสามารถฝัง ActiveX control ลงในเว็บเพจได้แล้วเรียกได้ตลอดเวลาที่เซิร์ฟเวอร์โหลดเว็บเพจ

ทางด้านเซิร์ฟเวอร์ Active server page เป็น ActiveX คอมโพเนนท์ที่สามารถพัฒนาได้ด้วยภาษาต่างๆ เช่น Delphi, C++, Java, VB ส่วนทางด้านไคลเอนต์ ASP จะถูกมองเป็นเอกสาร HTML และสามารถดูได้ด้วยเว็บบราวเซอร์ต่างๆ

3.4.5 Active Document

แอคทีฟดอคคิวเมนต์ อ้างถึง OLE ดอคคิวเมนต์ เป็นเซตของบริการของ COMเกี่ยวกับการ linking และembedding, drag และ drop และ visual editing ตัว แอคทีฟดอคคิวเมนต์ นั้นสามารถรวมข้อมูลหรือออบเจ็กต์ที่มีรูปแบบที่ต่างกันเช่น เสียง , ตัวหนังสือ, รูปภาพ

ออบเจ็กต์ของ แอคทีฟดอคคิวเมนต์ สามารถเป็นคอนเทเนอร์ของเอกสารหรือเซิร์ฟเวอร์ของเอกสาร เราสามารถใช้ TOleContainer ซึ่งเป็นพื้นฐานของคอนเทเนอร์ของ แอคทีฟดอคคิวเมนต์ ในการสร้างออบเจ็กต์ของ แอคทีฟดอคคิวเมนต์ เซิร์ฟเวอร์ ตัว แอคทีฟดอคคิวเมนต์ เองนั้นสนับสนุนการใช้มาร์แชร์ลิงสำหรับการข้ามโพรเซสของแอปพลิเคชั่น แต่ไม่สามารถรันในรีโมท เซิร์ฟเวอร์ได้

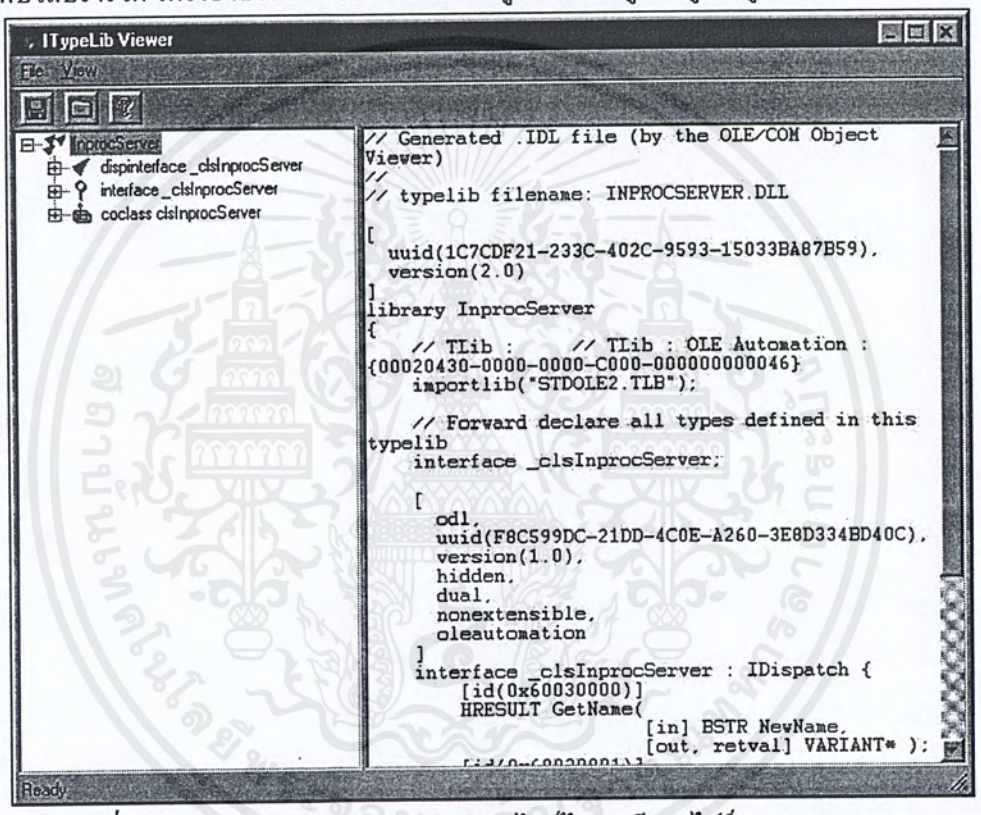
3.4.6 Visual cross-process objects

ออโตเมชันออบเจ็กต์, แอคทีฟดอคคิวเมนต์, ActiveX control เป็นการใช้ออบเจ็กต์ทั่วๆ ไปอย่างน้อย OLE และ ActiveX ออบเจ็กต์ ก็เป็นสิ่งที่มองเห็นและจัดการแบบข้ามโพรเซส รูปแบบของออบเจ็กต์เป็นการยากในการสร้างเพราะโพรโตคอลที่ใช้ในการจัดการออบเจ็กต์แบบข้ามโพรเซสจะใช้อินเตอร์เฟสของ แอคทีฟดอคคิวเมนต์ ดังนั้นการสร้างอินเตอร์เฟสและการทำมาร์แชลิงต่างๆ ต้องทำเอง

ที่นี้สามารถทำได้ด้วยการใช้ ดูอัลอินเตอร์เฟส IDispatch จะเตรียมมาร์แชลิงให้อัตโนมัติซึ่งเป็นวิธีที่แนะนำเพราะง่ายในการสร้าง อีกวิธีคือการเขียนมาร์แชลิงคลาสโดยใช้ IMarshal

บทที่ 4
การพัฒนาคอมโพเนนต์โดยใช้วิชวลเบสิก

ในวิชวลเบสิก เราสามารถสร้างคอมโพเนนต์ด้วย ActiveX control, ActiveX Document, ActiveX EXE และ ActiveX DLL โดยวิชวลเบสิกได้ถูกออกแบบให้ซ่อนความซับซ้อนในการสร้างคอมโพเนนต์ โดยจะสร้างไทป์ไลบรารี (Type Library) ให้โดยอัตโนมัติเมื่อคอมไพล์คอมโพเนนต์ ดังนั้นเราจึงไม่สามารถกำหนด GUID หรือรายละเอียดของอินเตอร์เฟสด้วยตนเองได้ แต่สามารถดูรายละเอียดของไทป์ไลบรารีได้ โดยใช้โปรแกรม OLE View ซึ่งอยู่ในวิชวลสตูดิโอทูลดังรูปที่ 4-1



รูปที่ 4-1 การใช้โปรแกรม OLE View แสดงไทป์ไลบรารีของไฟล์ InprocServer.dll

4.1 ชนิดของคอมโพเนนต์ในวิชวลเบสิก

วิชวลเบสิกสร้างคอมโพเนนต์ประเภทต่าง ๆ ดังนี้

1. อินโพรเซส ใช้ ActiveX DLL
2. เอาท์ออฟโพรเซส ใช้ ActiveX EXE
3. รีโมทเซิร์ฟเวอร์ ใช้ ActiveX DLL และ AxtiveX EXE

ส่วน ActiveX Control และ ActiveX Document ใช้ในการสร้างคอมโพเนนต์ที่ใช้งานบนเว็บ

4.2 อินเตอร์เฟสกับวิชวลเบสิก

ชื่ออินเตอร์เฟสของออบเจ็กต์ที่สร้างจากวิชวลเบสิก คือชื่อของคลาสโมดูลโดยมีเครื่องหมาย _ นำหน้า เช่น หากในโปรเจ็กต์มีคลาสโมดูลชื่อ clsInprocServer ชื่ออินเตอร์เฟสก็คือ _clsInprocServer

เป็นต้น ส่วนชื่อเมธอดภายในอินเตอร์เฟสก็คือฟังก์ชั่นหรือโพรซีเดอร์ต่าง ๆ ที่กำหนดไว้ภายในคลาสโมดูลที่ประกาศไว้เป็น Public

4.3การสร้าง COM ออบเจ็กต์ในวิชวลเบสิก

การสร้าง COM ออบเจ็กต์ในวิชวลเบสิกสามารถทำได้ 2 วิธีคือ

1. ใช้คำสั่ง New ซึ่งมีรูปแบบ

```
Dim [ชื่อออบเจ็กต์] As New [ชื่อคลาสโมดูล]
```

หรือใช้คำสั่ง

```
Dim [ชื่อออบเจ็กต์] As [ชื่อคลาสโมดูล]
Set [ชื่อออบเจ็กต์] = New [ชื่อคลาสโมดูล]
```

ซึ่งวิธีนี้หลังนี้สามารถทำลายออบเจ็กต์ได้เมื่อไม่ต้องการใช้ด้วยคำสั่ง

```
Set [ชื่อออบเจ็กต์] = Nothing
```

2. ใช้คำสั่ง CreateObject ซึ่งมีรูปแบบ

```
Dim [ชื่อออบเจ็กต์] As Object
Set [ชื่อออบเจ็กต์] = CreateObject("[ชื่อไฟล์ .dll หรือ .exe].[ชื่อคลาสโมดูล]")
```

4.4 ตัวอย่างการพัฒนาคอมโพเนนต์ซอร์ฟแวร์

ตัวอย่างที่ 4-1 พัฒนาคอมโพเนนต์ซอฟต์แวร์ประเภทอินโพรเซส (In-process)

จุดประสงค์

1. สามารถสร้างคอมโพเนนต์ประเภทอินโพรเซสได้
2. ทดสอบคุณสมบัติต่าง ๆ ของเทคโนโลยี COM อันได้แก่
   - การแก้ปัญหาด้านเวอร์ชันของซอฟต์แวร์
   - การไม่ขึ้นกับภาษาที่ใช้พัฒนา

เครื่องมือที่ใช้พัฒนา

1. วิชวลเบสิก 6.0 เอนเตอร์ไพรส์เอดิชัน
2. บอร์แลนด์เดลไฟ 5.0 เอนเตอร์ไพรส์

คำอธิบายโปรแกรม

1. ใช้วิชวลเบสิกสร้างคอมโพเนนต์ประเภทอินโพรเซส ซึ่งมีเมธอดทำหน้าที่รับข้อความและส่งคำทักทายพร้อมข้อความที่ได้รับ
2. สร้างไคลเอนท์แอปพลิเคชั่นด้วยวิชวลเบสิกที่มีการเรียกใช้คอมโพเนนต์ดังกล่าวโดยให้ผู้ใช้พิมพ์ชื่อลงในเท็กซ์บ็อกซ์ (Textbox) แล้วคลิกปุ่มเพื่อแสดงข้อความทักทาย

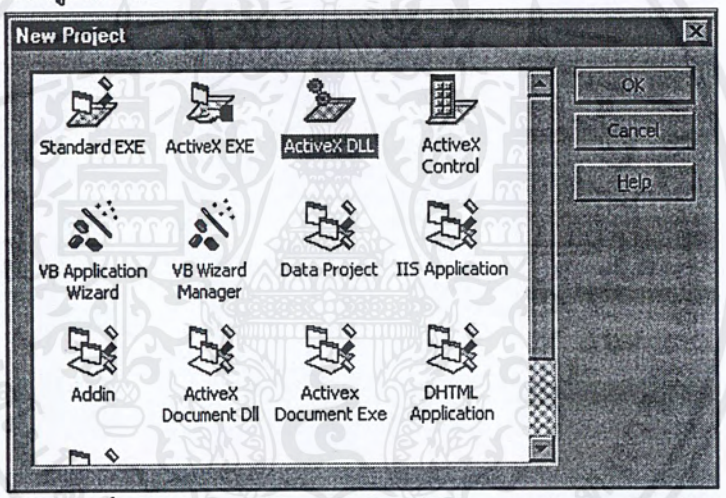
3. สร้างไคลเอนท์แอปพลิเคชั่นด้วยเคลไฟให้มีการทำงานเช่นเดียวกับไคลเอนท์แอปพลิเคชั่นที่เขียนด้วยวิชวลเบสิก
4. เปลี่ยนเวอร์ชันของคอมโพเนนต์ และใช้แอปพลิเคชั่นเดิมเรียกใช้คอมโพเนนต์เวอร์ชันใหม่ เพื่อแสดงให้เห็นว่าการเปลี่ยนเวอร์ชันไม่ทำให้แอปพลิเคชั่นเกิดปัญหาเรื่องการไม่เข้ากันระหว่างคอมโพเนนต์กับแอปพลิเคชั่น

ขั้นตอนการพัฒนา

1. ออกแบบอินเตอร์เฟสที่จำเป็น ในที่นี้มีเพียงอินเตอร์เฟสเดียวคือ InprocServer ซึ่ง InprocServer ประกอบด้วยเมธอด 2 เมธอดคือ เมธอด GetName มีหน้าที่รับค่าสตริงมาเก็บไว้ในตัวแปร และเมธอด SayHello มีหน้าที่ส่งสตริง "Hello, " ตามด้วยค่าของตัวแปรนั้น

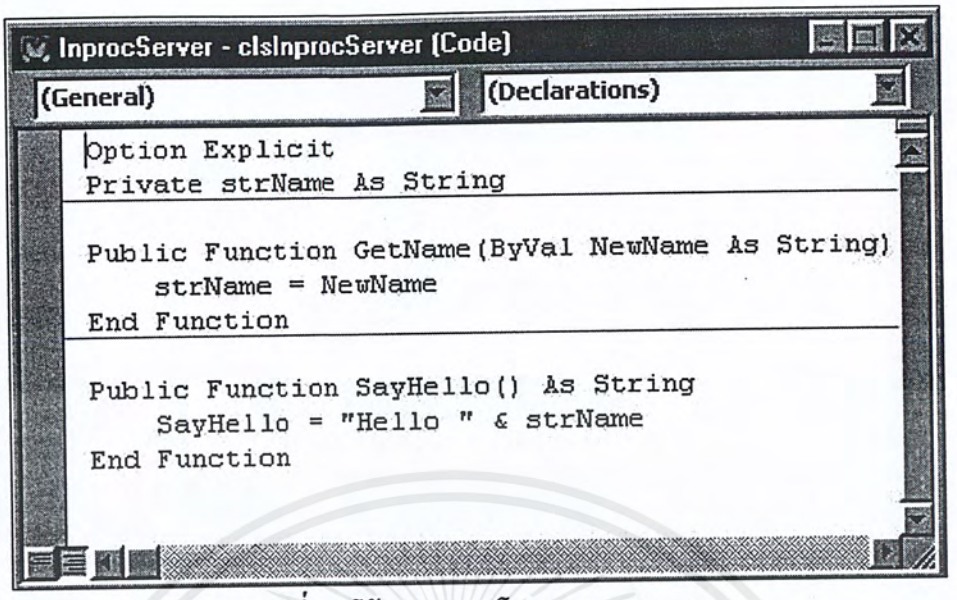
2. ทำการสร้างคอมโพเนนต์โดยใช้วิชวลเบสิกดังนี้

2.1 เปิดโปรแกรมวิชวลเบสิก 6.0 แล้วเลือก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ ActiveX DLL ดังรูปที่ 4-2



รูปที่ 4-2 หน้าจอเริ่มต้นในการสร้างคอมโพเนนต์ประเภทอินโพรเซสโดยใช้วิชวลเบสิก

2.2 ตั้งชื่อคลาสโมดูลชื่อ InprocServer เขียนโค้ดฟังก์ชั่น GetName และฟังก์ชัน SayHello ดังรูปที่ 4-3



รูปที่ 4-3 โค้ดของคลาสโมดูล InprocServer

2.3 ตั้งชื่อโปรเจ็กต์เป็น InprocServer แล้วทำการคอมไพล์โดยเลือก ***File > Make InprocServer.dll*** จะได้ไฟล์ InprocServer.dll ซึ่งเป็นคอมโพเนนต์ที่สามารถเรียกใช้ได้

2.4 บันทึกโปรเจ็กต์และปิดโปรแกรม

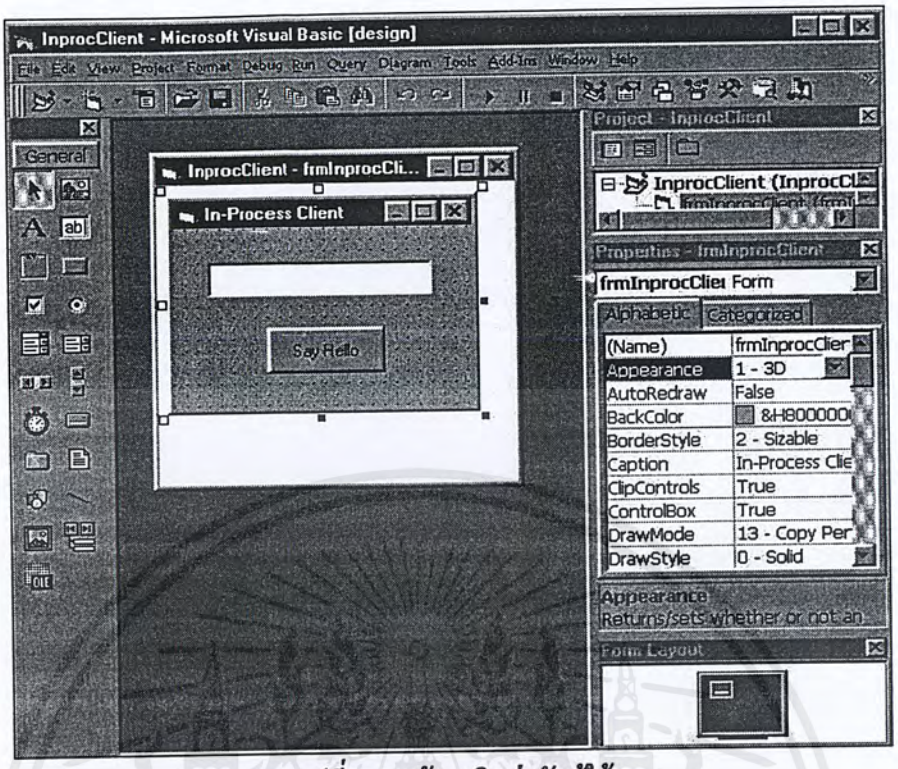
3. สร้างโปรแกรมทั่วไปที่จะเรียกใช้คอมโพเนนต์โดยใช้วิชวลเบสิก มีขั้นตอนต่อไปนี้

3.1 เปิดโปรแกรมวิชวลเบสิก 6.0 แล้วเลือก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ Standard EXE

3.2 ออกแบบหน้าจอส่วนติดต่อกับผู้ใช้ดังรูปที่ 4-4 โดยรายละเอียดของคอนโทรลดังตารางที่ 4-1

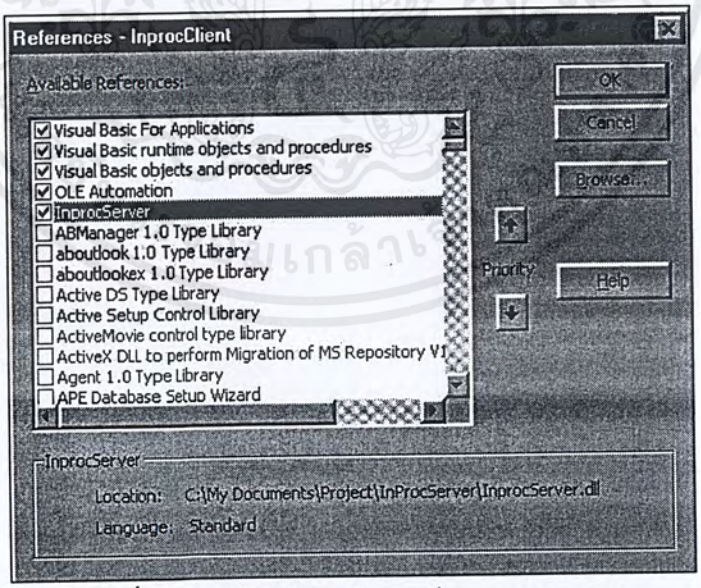
คอนโทรล	คุณสมบัติ	ค่า
Text1	Name	TxtName
	Text	<Blank>
Label1	Name	LblName
	Caption	Name:
Command1	Name	CmdSayHello
	Caption	Say Hello

ตารางที่ 4-1 รายละเอียดของคอลโทรล



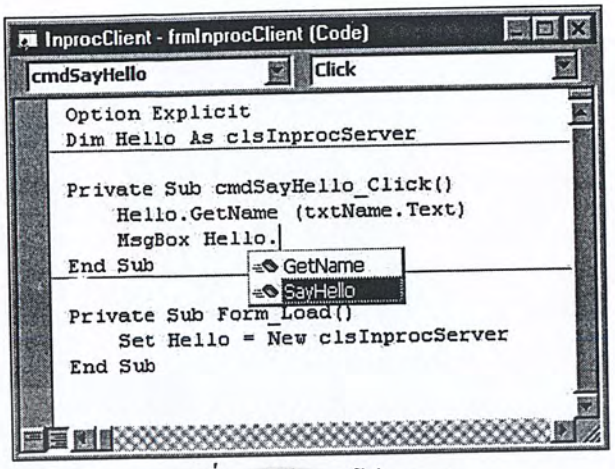
รูปที่ 4-4 หน้าจอติดต่อกับผู้ใช้

3.3 เรียกใช้คอมโพเนนต์ InprocServer1.dll โดยเลือก ***Project > References...*** จากนั้นเลือกไฟล์ InprocServer1.dll จะได้ผลดังรูปที่ 4-5



รูปที่ 4-5 การเรียกใช้คอมโพเนนต์ InprocServer.dll

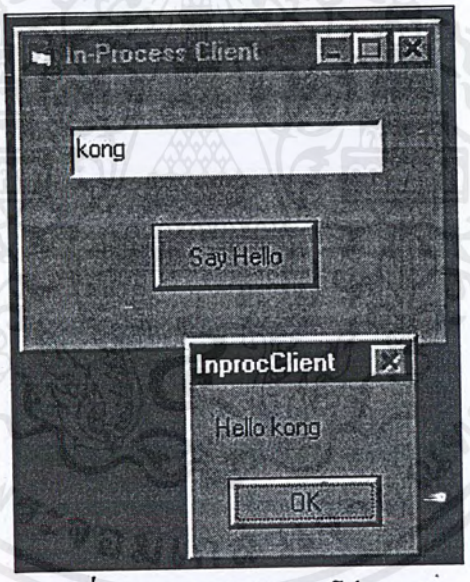
3.4 เขียนโค้ดเพื่อเรียกใช้บริการต่าง ๆ ของคอมโพเนนต์ดังรูปที่ 4-6



รูปที่ 4-6 โค้ดของโปรแกรม

3.5 ทดลองรันโปรแกรม โดยเลือก ***run > start*** หรือกดปุ่ม F5

3.6 ใส่ชื่อและคลิกปุ่ม Say Hello จะได้ผลลัพธ์ดังรูปที่ 4-7



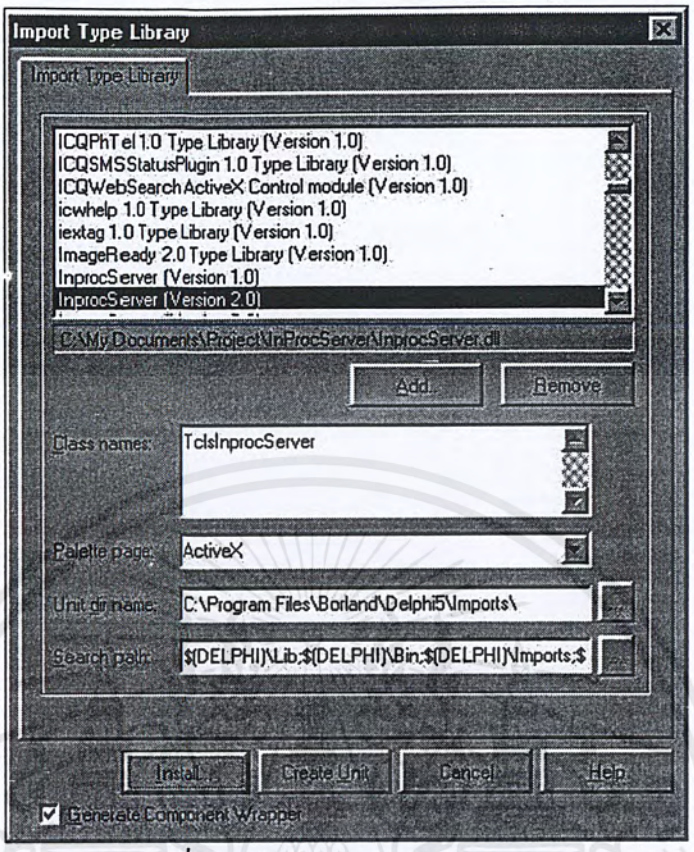
รูปที่ 4-7 ผลการทำงานของโปรแกรม

4. ทดสอบคุณสมบัติการไม่ขึ้นกับภาษาที่ใช้พัฒนา โดยใช้เดลไฟสร้างไคลเอนท์แอปพลิเคชั่น มีรายละเอียดดังนี้

4.1 เปิดโปรแกรมเดลไฟ คลิก ***File > New Application***

4.2 ก่อนเรียกใช้คอมโพเนนต์ต้องทำการอิมพอร์ตไทป์ไลบรารี โดยคลิก ***Project > Import Type Library*** จะปรากฏหน้าต่าง Import Type Library คลิกปุ่ม Add เพื่อเลือกไฟล์ .dll ที่ต้องการ ในที่นี้คือ ไฟล์ InprocServer.dll จากนั้นคลิกปุ่ม Install ดังรูปที่ 4-8

4.3 เรียกใช้คอมโพเนนต์โดยเพิ่มชื่อของไฟล์ .dll ตามด้วย _TLB ที่ส่วน uses เช่นในไคลเอนท์แอปพลิเคชันนี้ จะเพิ่มข้อความ InprocServer_TLB ลงไปดังรูปที่ 4-9



รูปที่ 4-8 หน้าต่าง Import Type Library

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  InprocServer_TLB, StdCtrls;
```

รูปที่ 4-9 การเรียกใช้คอมโพเนนต์ InprocServer.dll

4.4 การสร้างออบเจ็กต์ทำได้ดังนี้

- ประกาศชื่อตัวแปรออบเจ็กต์โดยมีรูปแบบ

  [ชื่อตัวแปร] : _[ชื่ออินเตอร์เฟส]; เช่น Hello : _clsInprocServer;

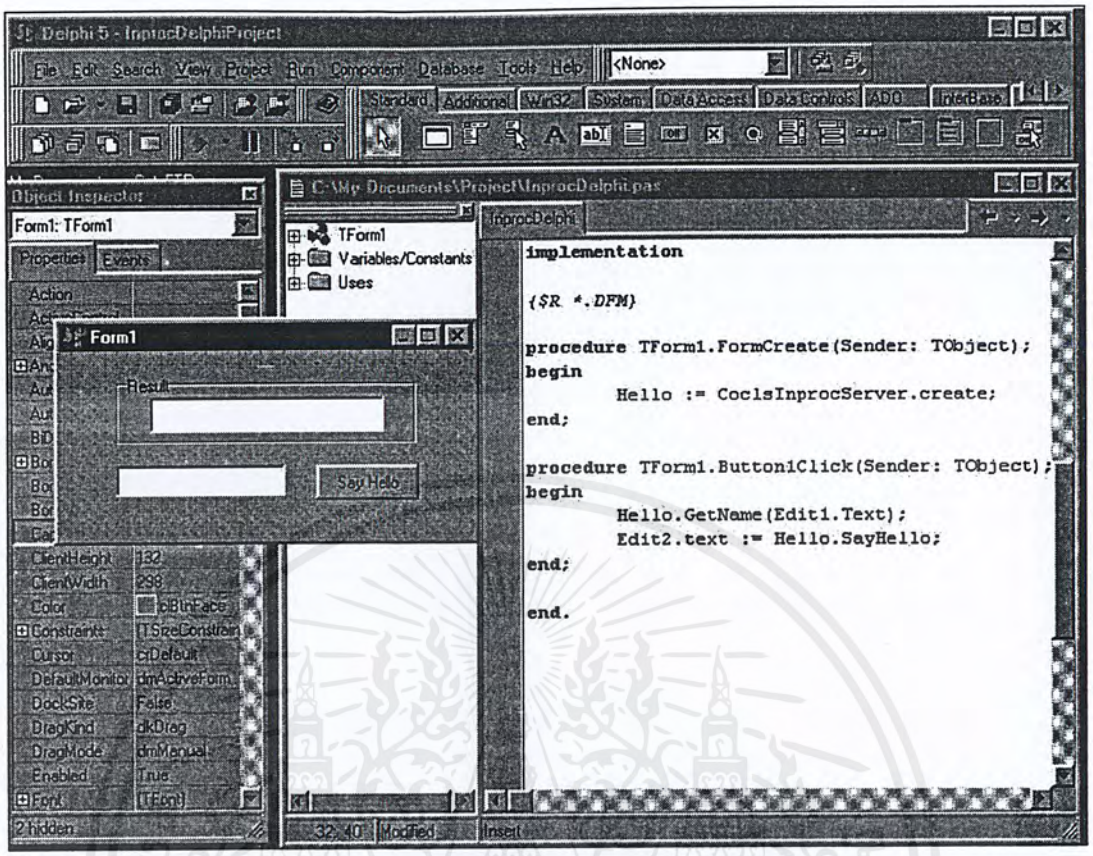
- การสร้างออบเจ็กต์โดยมีรูปแบบ

  [ชื่อตัวแปร] := Co[ชื่อคลาสโมดูล].create; เช่น Hello := CoclsInprocServer.create

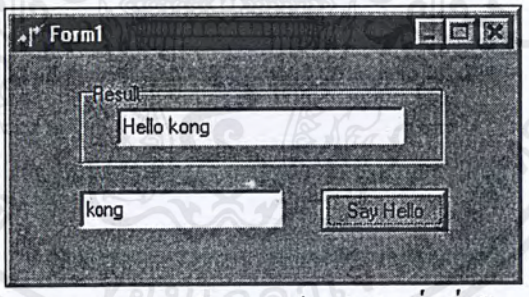
เมื่อสร้างออบเจ็กต์แล้วก็สามารถเรียกใช้เมธอดต่าง ๆ ได้ตามที่คอมโพเนนต์ประกาศ

4.5 ออกแบบหน้าจอส่วนติดต่อกับผู้ใช้ และเขียนโค้ดจัดการดังรูปที่ 4-10

4.6 ทดสอบโปรแกรม โดยคลิก ***Run*** > ***Run*** หรือกด F9 ทดลองป้อนข้อความแล้วคลิก Say Hello จะได้ผลการทำงานเหมือนไคลเอนท์แอปพลิเคชั่นที่สร้างจากวิชวลเบสิกดังรูปที่ 4-11



รูปที่ 4-10 หน้าจอส่วนติดต่อกับผู้ใช้และโค้ด



รูปที่ 4-11 ผลการทำงานของไคลเอนท์แอปพลิเคชั่นที่สร้างจากเดลไฟ

<u>ตัวอย่างที่ 4-2</u> **การพัฒนาคอมโพเนนต์ซอร์ฟแวร์ประเภทเอาท์ออฟโพรเซส (Out-of-process)**

จุดประสงค์

1. สามารถสร้างคอมโพเนนต์ประเภทเอาท์ออฟโพรเซสได้
2. ทดสอบคุณสมบัติการไม่ขึ้นกับภาษาที่ใช้พัฒนา

เครื่องมือที่ใช้ในการพัฒนา

1. วิชวลเบสิก 6.0 เอนเตอร์ไพรส์เอดิชัน
2. บอร์แลนด์เดลไฟ 5 เอนเตอร์ไพรส์

คำอธิบายโปรแกรม

1. ใช้วิชวลเบสิกสร้างคอมโพเนนต์ประเภทเอาท์ออฟโพรเซส เป็นเซิร์ฟเวอร์ทำหน้าที่รับข้อความและส่งข้อความที่ได้รับออกทางหน้าต่างของโปรแกรม

2. สร้างแอปพลิเคชั่นทำหน้าที่เป็นไคลเอนท์โดยใช้วิชวลเบสิก ทำการเรียกใช้เมธอดต่าง ๆ ของเซิร์ฟเวอร์ โดยรับสตริงจากเซิร์ฟเวอร์ และสามารถแสดงสตริงใหม่ออกทางหน้าต่างของโปรแกรมทั้งฝั่งไคลเอนท์และเซิร์ฟเวอร์ได้

3. สร้างแอปพลิเคชั่นทำหน้าที่เป็นไคลเอนท์โดยใช้เดลไฟมีการทำงานเหมือนกับไคลเอนท์ในข้อที่ 2

ขั้นตอนการพัฒนา

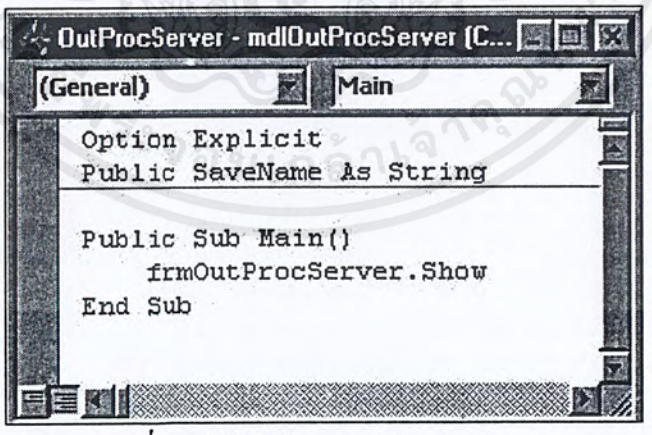
1. ออกแบบอินเตอร์เฟสที่จำเป็น ในที่นี้มีเพียงอินเตอร์เฟสเดียวคือ clsOutProcServer ซึ่งประกอบด้วย 2 เมธอดคือ เมธอด GetText ทำการรับสติงมาเก็บไว้ในตัวแปร และเมธอด SendText ทำการส่งค่าตัวแปรนั้น

2. ทำการสร้างคอมโพเนนต์โดยใช้วิชวลเบสิกดังนี้

2.1 เปิดโปรแกรมวิชวลเบสิก 6.0 เลือก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ ActiveX EXE และตั้งชื่อโปรเจ็กต์เป็น prjOutProcServer.vbp

2.2 สร้างโมดูล mdlOutProcServer โดยคลิก ***Project > Add Module*** การสร้างโมดูลเพื่อใช้ในการประกาศตัวแปรที่มีลักษณะเป็นสแตติกคลาสดาต้า (Static Class Data) คือเป็นตัวแปรที่ทุกอินสแตนทิเอตของคลาสสามารถใช้ร่วมกันได้ ซึ่งในที่นี้คือตัวแปร SaveName

2.3 เขียนโค้ดภายในโมดูลดังรูปที่ 4-12

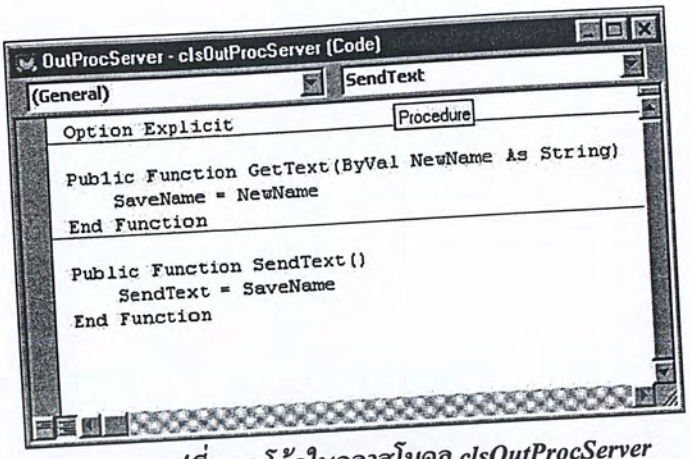


รูปที่ 4-12 โค้ดในโมดูล mdlOutProcServer

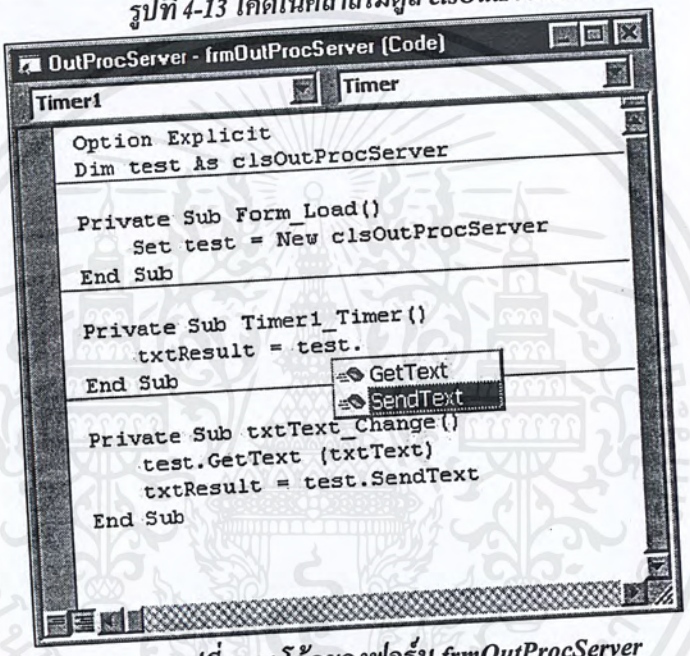
2.4 สร้างคลาสโมดูล clsOutProcServer โดยคลิก ***Project > Add Class Module***

2.5 เขียนโค้ดภายในคลาสโมดูลดังรูปที่ 4-13

2.6 สร้างฟอร์ม โดยคลิก ***Project > Add Form*** เพื่อเป็นส่วนโต้ตอบกับผู้ใช้ เนื่องจากคอมโพเนนต์แบบเอาท์ออฟโพรเซสสามารถทำงานได้ด้วยตนเอง โดยมีการอ้างฟังก์ชันต่าง ๆ ในคลาสโมดูล clsOutProcServer ดังรูปที่ 4-14



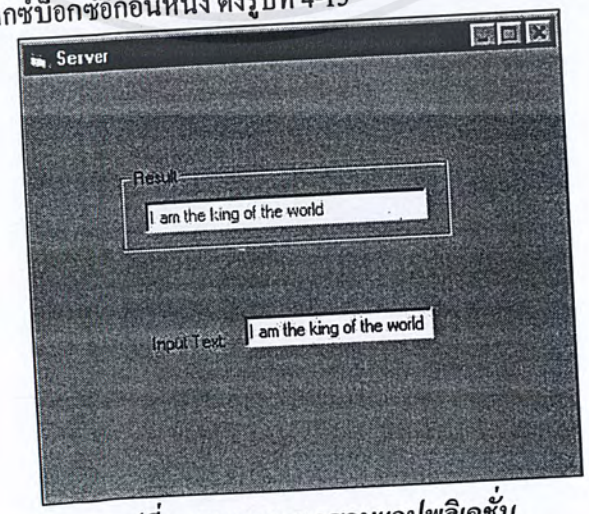
รูปที่ 4-13 โค้ดในคลาสโมดูล clsOutProcServer



รูปที่ 4-14 โค้ดของฟอร์ม frmOutProcServer

2.7 ทดสอบคอมโพเนนต์โดยคลิก ***Run > Start*** หรือกด ***F5***

2.8 ทดสอบโดยพิมพ์ข้อความลงในเท็กซ์บ็อกซ์ เซิร์ฟเวอร์แอพพลิเคชั่นจะทำงานโดยรับข้อความนั้นไปแสดงที่เท็กซ์บ็อกซ์อีกอันหนึ่ง ดังรูปที่ 4-15



รูปที่ 4-15 ผลการทดสอบแอปพลิเคชั่น

2.9 บันทึกโปรเจ็กต์และคอมไพล์โปรเจ็กต์โดยคลิก ***File > Make prjOutProcSercer.exe*** จะได้ไฟล์ prjOutProcServer ซึ่งเป็นคอมโพเนนต์แบบเอาท์ออฟโพรเซสที่ต้องการ

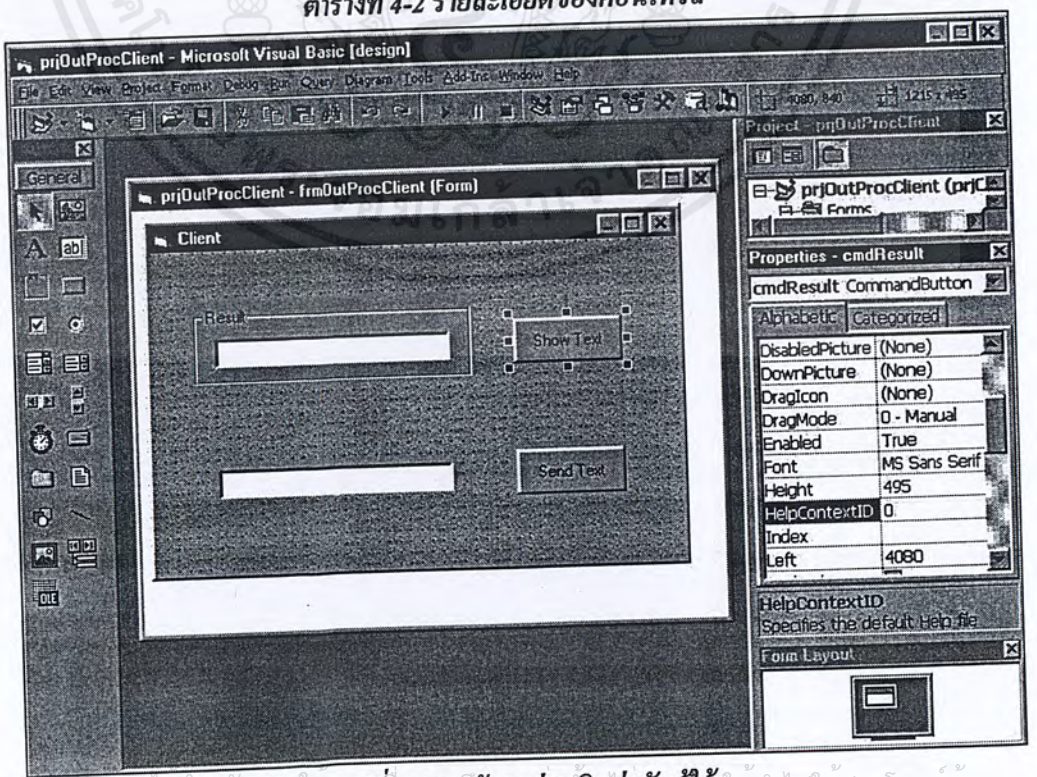
3. สร้างไคลเอนท์แอปพลิเคชั่นโดยใช้วิชวลเบสิก มีรายละเอียดดังต่อไปนี้

3.1 เปิดโปรแกรมวิชวลเบสิก 6.0 แล้วเลือก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ Standard EXE และตั้งชื่อโปรเจ็กต์เป็น prjOutProcClient.vbp

3.2 ออกแบบหน้าจอส่วนติดต่อกับผู้ใช้ดังรูปที่ 4-16 โดยรายละเอียดของคอนโทรลดังตารางที่ 4-2

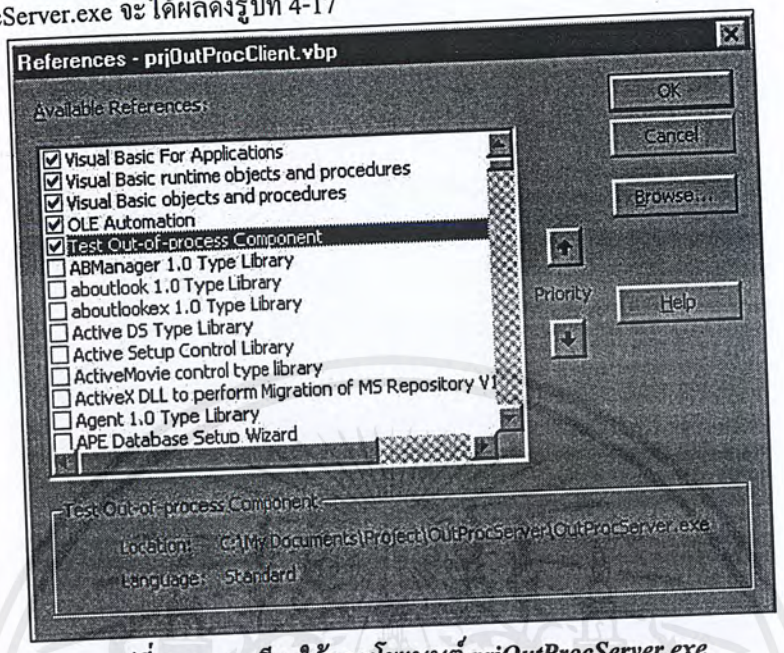
คอนโทรล	คุณสมบัติ	ค่า
Text1	Name	txtText
	Text	<Blank>
Text2	Name	TxtResult
	Text	<Blank>
Frame1	Name	FraResult
	Caption	Result
Command1	Name	CmdResult
	Caption	Show Text
Command2	Name	cmdSendText
	Caption	Send Text

ตารางที่ 4-2 รายละเอียดของคอนโทรล



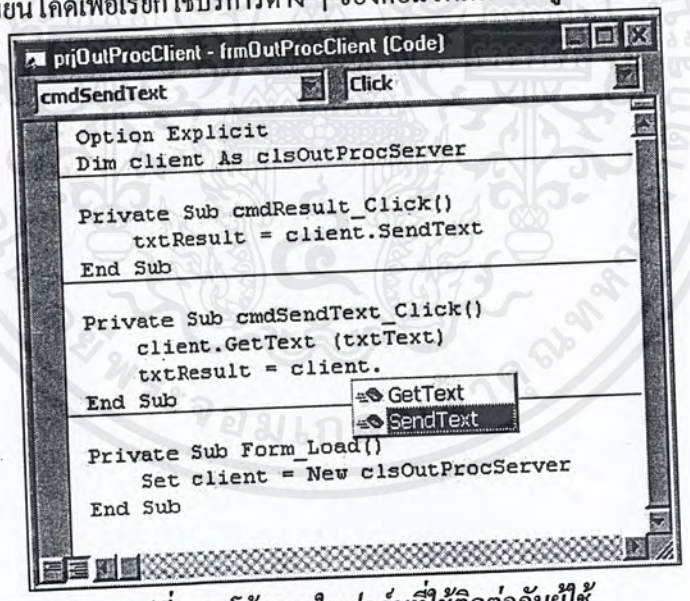
รูปที่ 4-16 หน้าจอส่วนติดต่อกับผู้ใช้

3.3 เรียกใช้คอมโพเนนต์ prjOutProcServer.exe โดยเลือก ***Project > References...*** เลือกไฟล์ prjOutProcServer.exe จะได้ผลดังรูปที่ 4-17



รูปที่ 4-17 การเรียกใช้คอมโพเนนต์ prjOutProcServer.exe

3.4 เขียนโค้ดเพื่อเรียกใช้บริการต่าง ๆ ของคอมโพเนนต์ดังรูปที่ 4-18

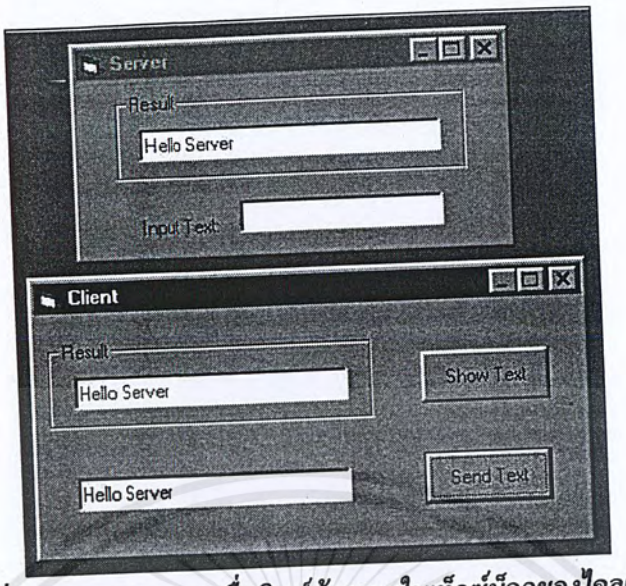


รูปที่ 4-18 โค้ดภายในฟอร์มที่ใช้ติดต่อกับผู้ใช้

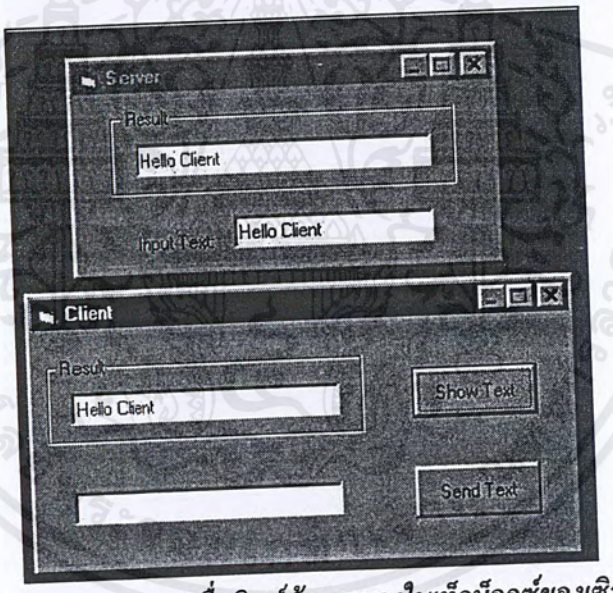
3.5 ทดสอบไคลเอนท์แอปพลิเคชั่น โดยคลิก ***Run > Start*** หรือกด ***F5***

3.6 เมื่อรันไคลเอนท์แอปพลิเคชั่น ส่วนของเซิร์ฟเวอร์แอปพลิเคชันจะปรากฏขึ้นมาทันที จากนั้นลองพิมพ์ข้อความลงในเท็กซ์บ็อกซ์ของไคลเอนท์ แล้วคลิก Sent Text ข้อความที่พิมพ์จะปรากฏที่เท็กซ์บ็อกซ์ทั้งไคลเอนท์และเซิร์ฟเวอร์ ดังรูปที่ 4-19

3.7 ทดลองพิมพ์ข้อความในเท็กซ์บ็อกซ์ของเซิร์ฟเวอร์ จากนั้นกลับมาคลิกปุ่ม Show Text ของไคลเอนท์ ข้อความในเท็กซ์บ็อกซ์ของเซิร์ฟเวอร์จะปรากฏที่เท็กซ์บ็อกซ์ของไคลเอนท์ดังรูปที่ 4-20

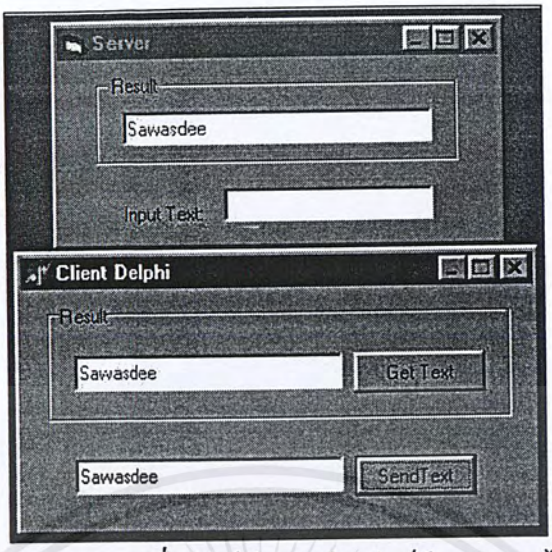


รูปที่ 4-19 ผลการทดลองเมื่อพิมพ์ข้อความในเท็กซ์บ็อกของไคลเอนท์ แล้วคลิก Send Text ข้อความจะไปปรากฏที่เท็กบ็อกซ์ของเซิร์ฟเวอร์

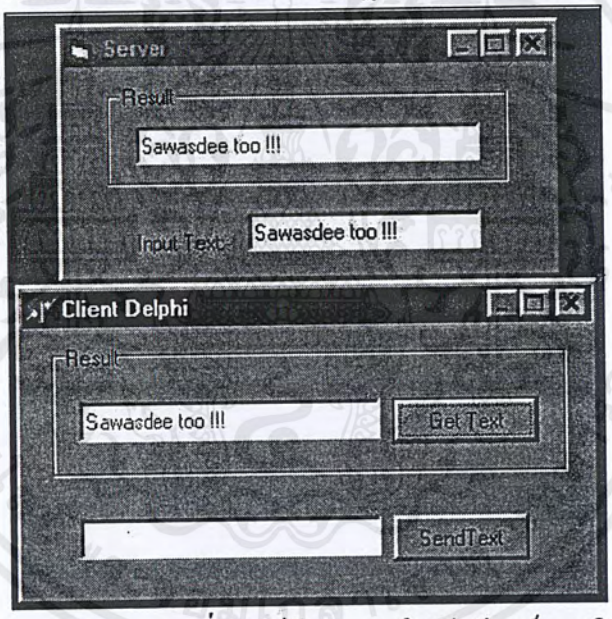


รูปที่ 4-20 ผลการทดลองเมื่อพิมพ์ข้อความลงในเท็กบ็อกซ์ของเซิร์ฟเวอร์ เมื่อไคลเอนท์คลิก Show Text ข้อความจะไปปรากฏที่เท็กซ์บ็อกซ์ในเฟรม Result

4. ทดสอบคุณสมบัติการไม่ขึ้นกับภาษาที่ใช้พัฒนา โดยใช้เดลไฟสร้างไคลเอนท์แอปพลิเคชั่น การสร้างมีขั้นตอนเช่นเดียวกับการสร้างไคลเอนท์แอปพลิเคชั่นด้วยเดลไฟในการทดลองที่ 1 ทดลองรันไคลเอนท์จะได้ผลการทดลองดังรูปที่ 4-21 และรูปที่ 4-22 ซึ่งผลการทดลองเหมือนกับการการทำงานของไคลเอนท์แอปพลิเคชันที่เขียนด้วยวิชวลเบสิก



รูปที่ 4-21 ผลการทดลองเมื่อพิมพ์ข้อความในเท็กซ์บ็อกของเดลไฟไคลเอนท์ แล้วคลิก Send Text ข้อความจะไปปรากฏที่เท็กบ็อกซ์ของเซิร์ฟเวอร์



รูปที่ 4-22 ผลการทดลองเมื่อพิมพ์ข้อความลงในเท็กบ็อกซ์ของเซิร์ฟเวอร์ เมื่อเดลไฟไคลเอนท์คลิก Show Text ข้อความจะไปปรากฏที่เท็กซ์บ็อกซ์ในเฟรม Result

<u>ตัวอย่างที่ 3</u> การพัฒนาคอมโพเนนต์ประเภทรีโมทเซิร์ฟเวอร์ (Remote Server)

จุดประสงค์

1. สามารถสร้างรีโมทคอมโพเนนต์ที่สามารถเรียกใช้ระหว่างเครื่องได้

เครื่องมือที่ใช้พัฒนา

1. วิชวลเบสิก 6.0 เอนเตอร์ไพรส์เอดิชัน

คำอธิบายโปรแกรม

1. ใช้วิชวลเบสิกสร้างรีโมทคอมโพเนนต์ เป็นเซิร์ฟเวอร์ซึ่งมีเมธอดทำหน้าที่รับข้อความและแสดงข้อความ "Welcome" ตามด้วยข้อความนั้นและเครื่องหมาย "!!!" โดยทำงานอยู่บนคอมพิวเตอร์ระบบปฏิบัติการวินโดว์เอ็นที 4.0 เซิร์ฟเวอร์

2. สร้างไคลเอนท์แอปพลิเคชั่นซึ่งทำงานอยู่คนละเครื่องกับคอมโพเนนต์ที่สร้าง ทำการติดต่อกับคอมโพเนนต์โดยใช้ DCOM และเรียกใช้เมธอดต่าง ๆ ของคอมโพเนนต์

ขั้นตอนการพัฒนาโปรแกรม

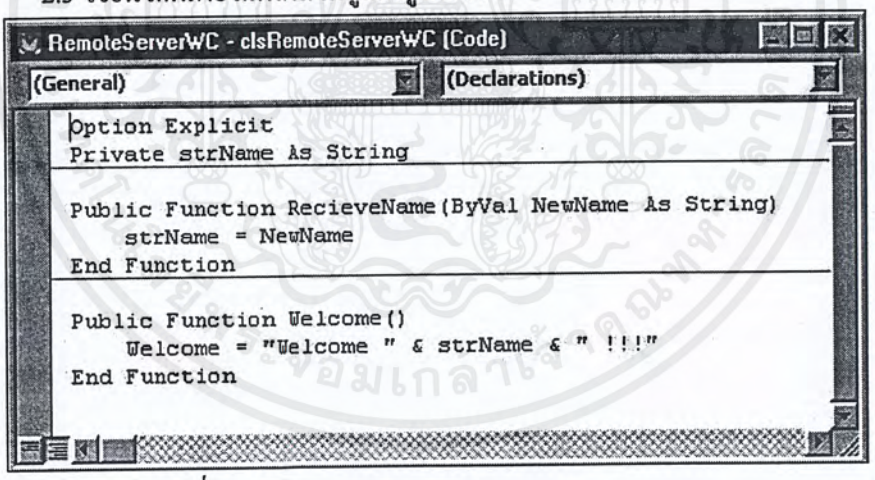
1. ออกแบบอินเตอร์เฟส ซึ่งมีอินเตอร์เฟสเดียวคือ clsRemoteServerWC มีเมธอด 2 เมธอด คือ RecieveName() ทำหน้าที่รับค่าพารามิเตอร์มาเก็บไว้ในตัวแปรภายในคอมโพเนนต์ และเมธอด Welcome ทำหน้าที่ส่งข้อความ "Welcome" ตามด้วยค่าตัวแปรที่ได้จากเมธอด RecieveName() และต่อท้ายด้วยข้อความ "!!!"

2. ทำการสร้างคอมโพเนนต์บนระบบปฏิบัติการวินโดว์เอ็นที 4.0 เซิร์ฟเวอร์ดังนี้

2.1 เปิดโปรแกรมวิชวลเบสิก 6.0 เลือก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ ActiveX EXE ตั้งชื่อโปรเจ็กต์เป็น RemoteServerWC

2.2 สร้างคลาสโมดูล clsRemoteServerWC โดยคลิก ***Project > Add Class Module***

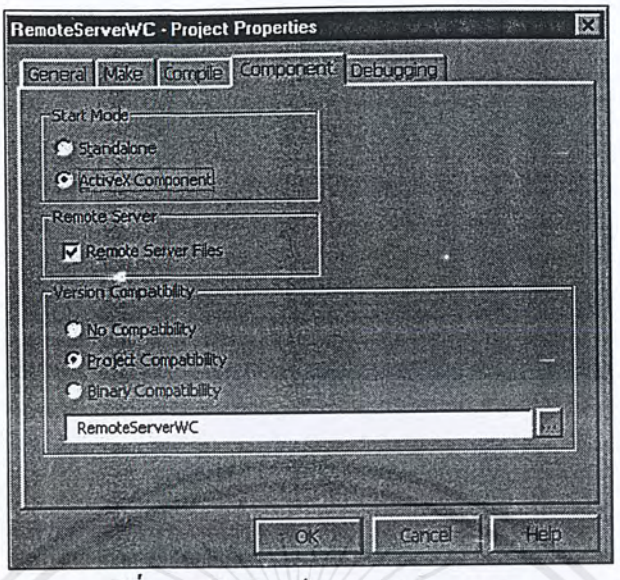
2.3 เขียนโค้ดภายในคลาสโมดูลดังรูปที่ 4-23



รูปที่ 4-23 โค้ดภายในคลาสโมดูล clsRemoteServerWC

2.4 คลิก ***Project > RemoteServerWC Properties...*** จะมีหน้าต่าง Project Properties ปรากฏขึ้นมา เลือกแท็บ Component และทำเครื่องหมายที่เช็กบ็อกซ์ Remote Server Files ดังรูปที่ 4-24 จุดประสงค์เพื่อเมื่อทำการคอมไพล์โปรเจ็กต์ วิชวลเบสิกจะทำการสร้างไฟล์ .vbr และ .tlb ซึ่งมีชื่อเดียวกับโปรเจ็กต์ ไฟล์ .vbr มีข้อมูลที่วินโดว์รีจิสทรีต้องการในการรันคอมโพเนนต์จากเครื่องอื่น

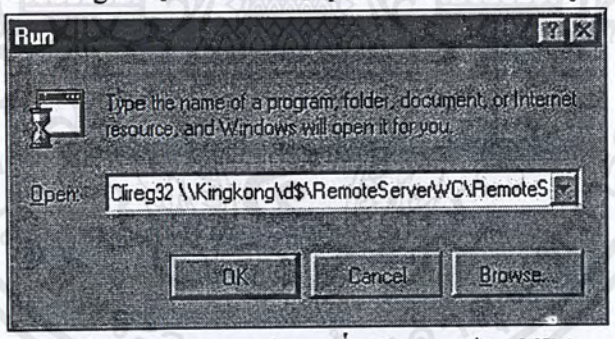
2.5 คลิก ***File > Make RemoteServerWC.exe*** เพื่อทำการสร้างไฟล์ RemoteServerWC.exe พร้อมกันนั้นจะได้ไฟล์ RemoteServerWC.tlb และ RemoteServerWC.vbr



รูปที่ 4-24 เช็กบ็อกซ์ Remote Server Files

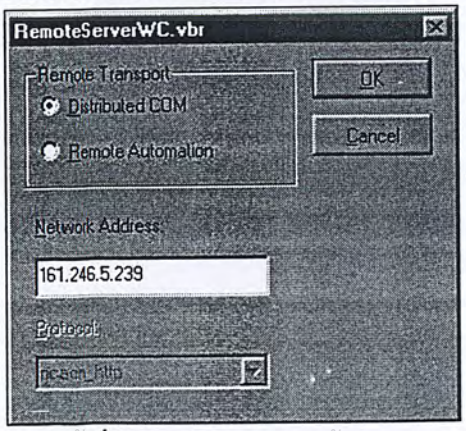
3. สร้างไคลเอนท์แอปพลิเคชั่นบนเครื่องอื่น โดยในการทดลองนี้ใช้เครื่องระบบปฏิบัติการวินโดว์ 98

3.1 รีจิสเตอร์รีโมทคอมโพเนนต์บนเครื่องไคลเอนต์ โดยใช้โปรแกรมClireg32.exe บนเดสก์ท็อปของวินโดว์ 98 คลิกปุ่ม ***Start > Run...*** และพิมพ์คำสั่ง Clireg32 ตามด้วย path ของไพล์ .vbr ที่อยู่บนเครื่องเซิร์ฟเวอร์ เช่น clireg32 \\[ชื่อเซิร์ฟเวอร์]\[path ของไฟล์ .vbr] ดังรูปที่ 4-25



รูปที่ 4-25 การรีจิสเตอร์รีโมทคอมโพเนนต์บนเครื่องไคลเอนท์โดยใช้โปรแกรม Clireg32.exe

3.2 เมื่อคลิกปุ่ม OK จะปรากฏหน้าต่างดังรูปที่ 4-26 เลือก Distributed COM และในเท็กซ์บ็อกซ์ Network Address พิมพ์ IP แอดเดรสของเครื่องเซิร์ฟเวอร์ จากนั้นคลิกปุ่ม OK



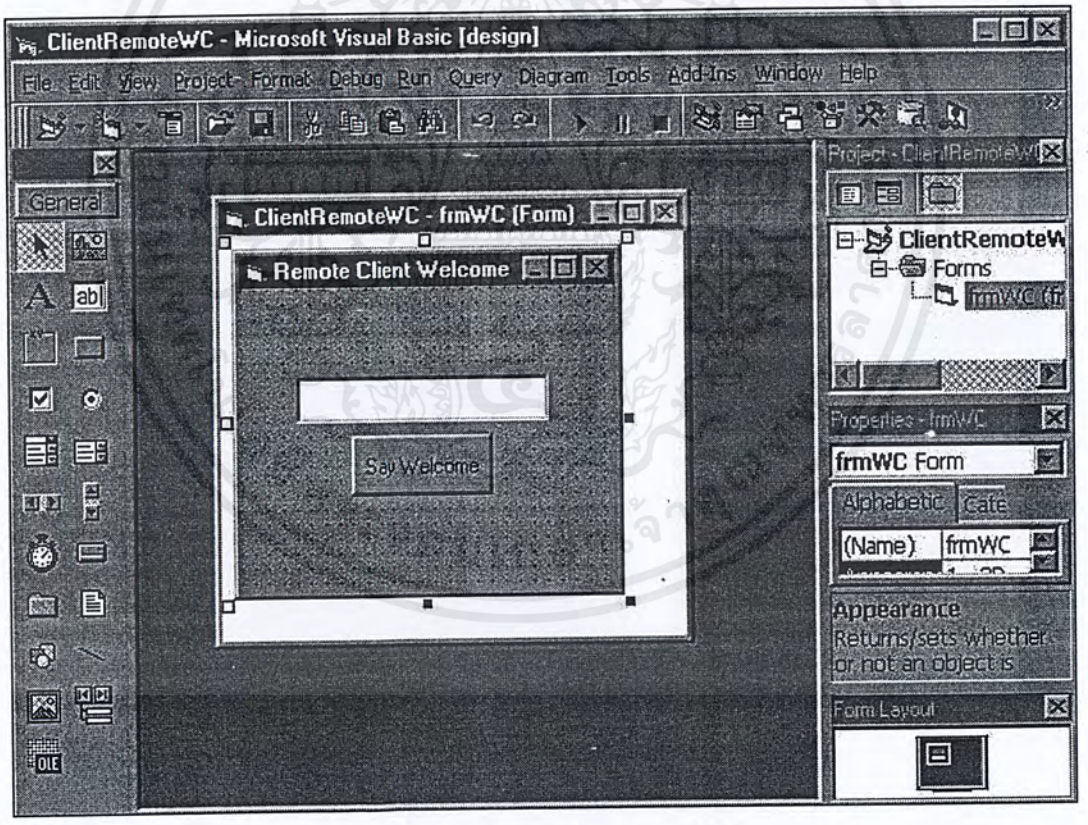
รูปที่ 4-26 โปรแกรม Clireg32.exe

3.3 สร้างไคลเอนท์แอปพลิเคชั่น โดยเปิดโปรแกรมวิชวลเบสิก 6.0 คลิก ***File > New Project*** และเลือกประเภทโปรเจ็กต์ Standard EXE ตั้งชื่อโปรเจ็กต์เป็น ClientRemoteWC

3.4 ออกแบบหน้าจอส่วนติดต่อกับผู้ใช้ดังรูปที่ 4-27 โดยรายละเอียดของคอนโทรลแสดงในตารางที่ 4-3

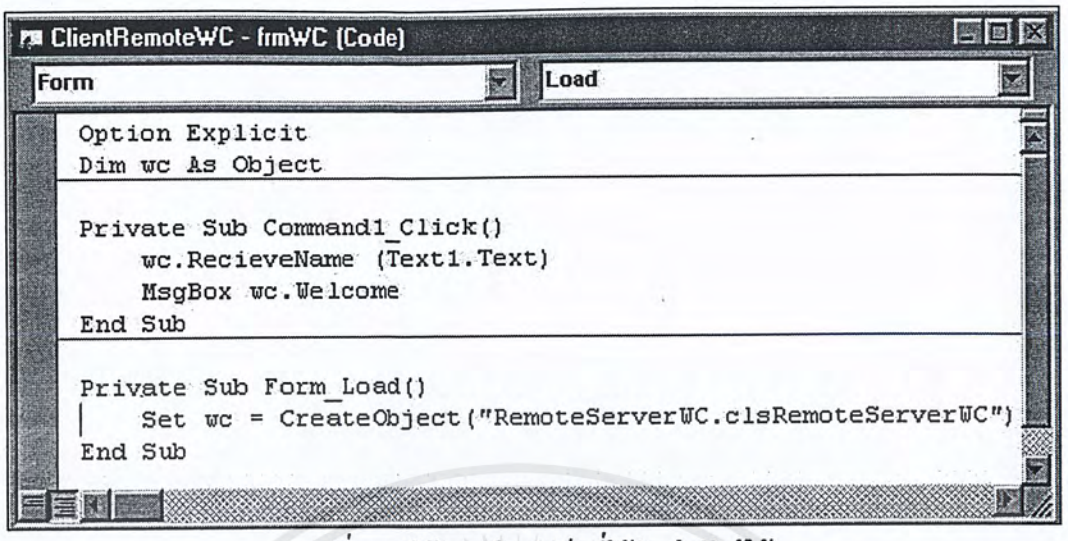
คอนโทรล	คุณสมบัติ	ค่า
Text1	Name	TxtName
	Text	<Blank>
Command1	Name	CmdSayWelcome
	Caption	Say Welcome
Form1	Name	FrmWC
	Caption	Remote Client Welcome

ตารางที่ 4-3 รายละเอียดของคอนโทรล



รูปที่ 4-27 หน้าจอส่วนติดต่อกับผู้ใช้

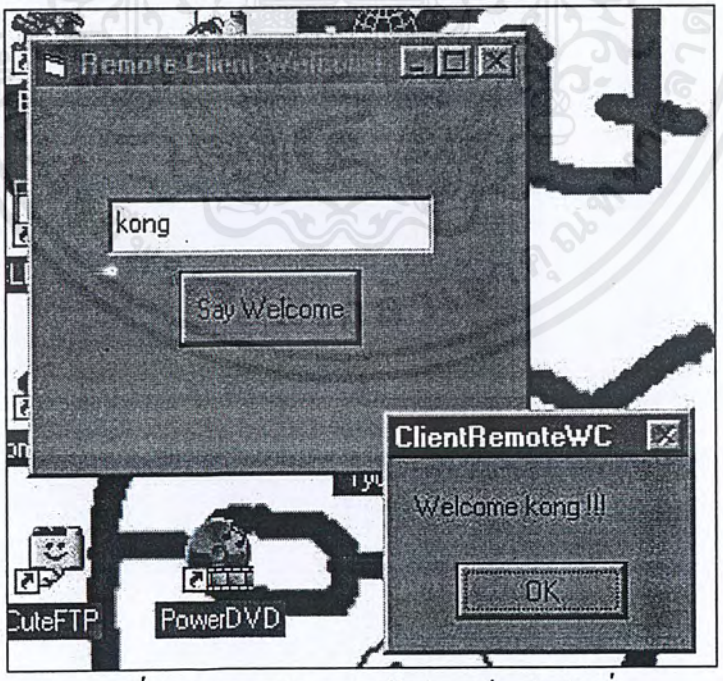
3.5 เขียนโค้ดดังรูปที่ 4-28



รูปที่ 4-28 โค้ดภายในฟอร์มที่ใช้ติดต่อกับผู้ใช้

จากรูปที่ 4-28 คำสั่งที่ใช้สร้างออบเจ็กต์คือ CreateObject() โดยพารามิเตอร์ภายในเป็นชื่อของคอมโพเนนต์ตามด้วยคลาสโมดูลที่ต้องการใช้งาน เมื่อสร้างออบเจ็กต์แล้ว จะสามารถเรียกใช้เมธอดต่าง ๆ ได้ ซึ่งในโปรแกรมนี้มี 2 เมธอดคือ RecieveName และ Welcome

3.6 ทดสอบการทำงานของโปรแกรมโดยคลิก ***Run > Start*** หรือพิมพ์ F5 ทดลองพิมพ์ข้อความลงในเท็กซ์บ็อกซ์แล้วคลิก Say Welcome จะได้ผลการทำงานดังรูปที่ 4-29



รูปที่ 4-29 ผลการทำงานของไคลเอนท์แอปพลิเคชั่น

บทที่ 5
การพัฒนา COM โดยใช้ Borland Delphi 5.0

5.1 กรณีศึกษาการพัฒนา คอมโพเนนท์ ออบเจ็กต์ ประเภท อิน – โพรเซส (In-Process COM Object)

จุดประสงค์

1. เข้าใจถึงวิธีการในการพัฒนาคอมโพเนนท์ซอฟต์แวร์ประเภท อิน – โพรเซส
2. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภท อิน – โพรเซส จาก เครื่องมือพัฒนาตัวเดียวกัน
3. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภท อิน – โพรเซส จาก เครื่องมือพัฒนาอื่น

เครื่องมือที่ใช้พัฒนา

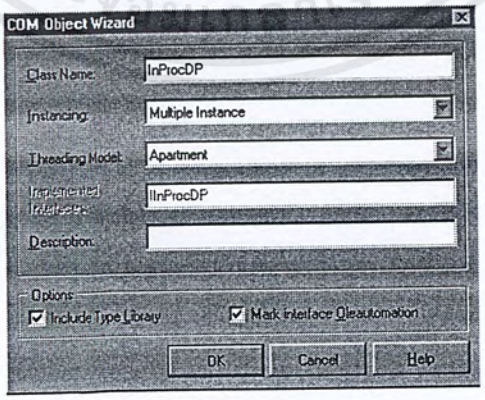
- Borland Delphi 5.0
- Microsoft Visual Basic 6.0

คำอธิบายโปรแกรม

- เป็นคอมโพเนนท์ออบเจ็กต์ ประเภท อิน-โพรเซส ซึ่งมี อินเตอร์เฟส 1 อินเตอร์เฟส
- ภายในอินเตอร์เฟสจะมี เมธอด 2 เมธอด ซึ่งได้แก่ในการรับค่าสตริงจากไคลเอนต์ และเมธอดในการส่งค่าสตริงคืนไคลเอนต์

5.1.1 ขั้นตอนการพัฒนา

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File* > *New* แล้วเลื่อนไปที่หน้า ActiveX แล้วเลือก ActiveX Library
2. สร้างคอมโพเนนท์ออบเจ็กต์ใหม่ โดยเลือก *File* > *New* แล้วเลื่อนไปที่หน้า ActiveX แล้วเลือก COM Object จะขึ้น COM Object Wizard ไดอะล็อก ให้ใส่ข้อมูลจนมีลักษณะดัง รูปที่ 5-1

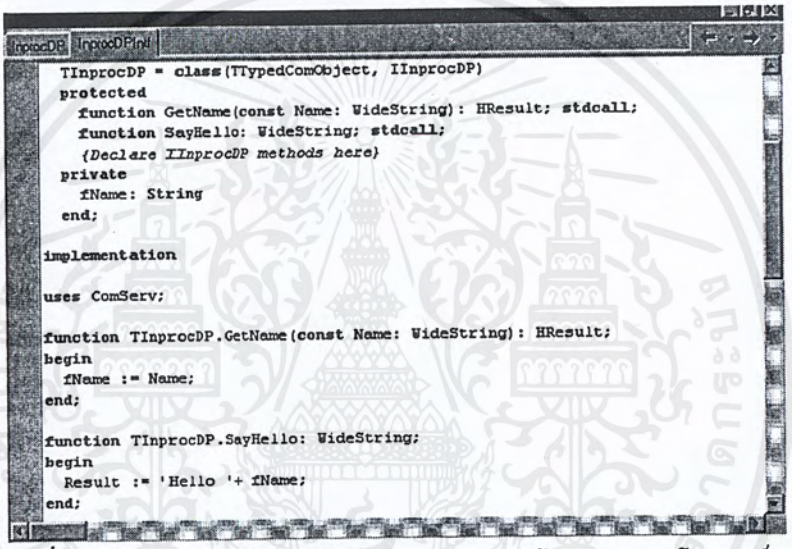


รูปที่ 5-1 แสดงหน้าจอของ COM Object Wizard ในการสร้างอิน-โพรเซส คอมโพเนนท์

3. กด OK จะทำให้เกิดการเรียกใช้ type library editor ให้เพิ่มเมธอด ลงไปใน IInprocDP ดังนี้

- เมธอด GetName มี return value เป็น HResult, มีพารามิเตอร์ ชื่อ Name มีชนิดเป็น BSTR ซึ่งมี Modifier เป็น [in]
- เมธอด SayHello มี return value เป็น HResult, มีพารามิเตอร์ ชื่อ Result มีชนิดเป็น BSTR * ซึ่งมี Modifier เป็น [out, retval]

4. กด Refresh เพื่อทำการเปลี่ยนข้อมูลที่กำหนดไว้ใน type library editor ให้เป็น ซอร์สโค้ด
5. เพิ่ม field เข้าไปใน Class TinprocDP 1 field โดยให้มีชนิดเป็นสตริง
6. เพิ่มซอร์สโค้ดในส่วนของเมธอด SayHello และGetName ซึ่งลักษณะของซอร์สโค้ดที่ได้ถูกแก้ไขเรียบร้อยแล้วจะมีลักษณะดังรูปที่ 5-2



รูปที่ 5-2 แสดงซอร์สโค้ดของตัวโปรแกรมของ อิน-โพรเซส คอมโพเนนท์

7. ทำการบันทึกไฟล์ทั้งหมดของโปรเจ็กต์โดย บันทึกไฟล์ของโปรเจ็กต์ไว้ที่ InprocDP.dpr และ ตัวซอร์สโค้ดจะบันทึกไว้ที่ InprocIntf.pas
8. ทำการรีจิสเตอร์อินเตอร์เฟสโดยเลือก *Run > Register ActiveX Server* หากไม่มีข้อผิดพลาดใด ๆ จะมีการแสดงไดอะลอกที่บอกว่าการรีจิสเตอร์นั้นได้เสร็จสมบูรณ์แล้ว และจะพบว่าได้มีการสร้างไฟล์ที่ชื่อว่า InprocDP_TLB.pas ขึ้นมาด้วย
9. ทำการสร้างไฟล์ InprocDP.dll โดยเลือก *Project > Build InprocDP*
10. เสร็จสิ้นการพัฒนาออบเจ็กต์ คอมโพเนนท์ประเภท อิน-โพรเซส

5.1.2 ขั้นตอนการเรียกใช้งาน ด้วย Borland Delphi 5.0

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File > New* แล้วเลือก Application จากหน้า New
2. อิมพอร์ตไฟล์ที่เป็น TLB ซึ่งในที่นี้คือ InprocIntf_TLB.pas โดยเลือก *Project > Add to Project* แล้วเลือกไฟล์ InprocIntf_TLB.pas จากตำแหน่งไดเรกทอรี่ที่บันทึกไว้หรือในไดเรกเทอรี่ที่ได้ copy เฉพาะส่วนที่เป็น ไฟล์ TLB อย่างเดียวก็ได้

3. เพิ่มการเรียกใช้ InprocIntf_TLB และ COMObj ในส่วนของ uses เช่น

```
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    COMObj, InprocDP_TLB, StdCtrls;
```

4. ประกาศตัวแปรประเภท IInprocDP ในส่วน Private
5. เมื่อจะทำการเรียกใช้ตัวแปรที่เป็นคอมโพเนนท์ จะต้องมีการสร้างตัวแปรที่เป็นคอมโพเนนท์โดยใช้คำสั่ง

```
a := CreateCOMObject(Class_xxxx) as IntfName;
```

โดย a จะเป็นชื่ออินแสตนซ์ของคอมโพเนนท์

xxxx จะแทนชื่อของคอมโพเนนท์ออบเจ็กต์เช่นในการทดลองนี้จะเป็น Class_InprocDP

IntfName จะแทนชื่อของอินเตอร์เฟส เช่น

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    a := CreateCOMObject(Class_InprocDP) as IInprocDP;
    a.GetName(Edit1.Text);
    Label1.Caption := a.SayHello;
end;
```

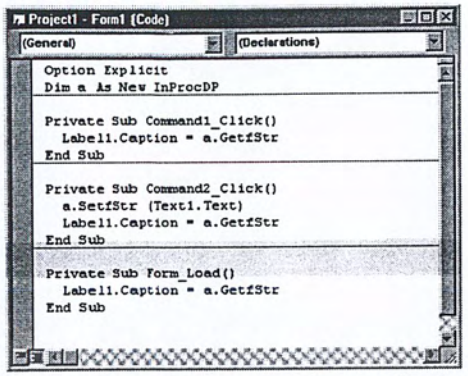
6. หลังจากนั้นจะสามารถใช้งานอินแสตนซ์ของ คอมโพเนนท์ได้เหมือนกับออบเจ็กต์ทั่วไป

5.1.3 ขั้นตอนการเรียกใช้งาน ด้วย Microsoft Visual Basic 6.0

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File* > *New Project* แล้วเลือก Standard EXE
2. ทำการอ้างอิงไปยังไฟล์ที่เป็นอิน-โพรเซส โดยทำการเลือก *Project* > *References* จากนั้นให้ใช้ปุ่ม Browse เพื่อหาไฟล์ DLL ของอิน-โพรเซส เซิร์ฟเวอร์
3. ทำการปรับแต่งและเพิ่มเติม องค์ประกอบของโปรแกรมใหม่นี้ โดยการเรียกใช้อิน-โพรเซสจะใช้คำสั่ง ในลักษณะ

```
Dim a As New InProcDP เพื่อทำการสร้างอินแสตนซ์ใหม่
Label1.Caption = a.GetfStr 'เพื่อเรียกใช้เมธอดของอินเตอร์เฟส
```

ซึ่งการนำไปใช้งานอาจ เป็นดังในรูปที่ 5-3



รูปที่ 5-3 แสดงการเรียกใช้ อิน-โพรเซส คอมโพเนนท์โดยใช้ Microsoft Visual Basic 6.0

4. ทำการเรียกใช้งานได้เหมือนกับออบเจ็กต์ทั่วไป

5.2 กรณีศึกษาการพัฒนา คอมโพเนนท์ ออบเจ็กต์ ประเภท เอาท์– โพรเซส (Out-Process COM Object)

จุดประสงค์

1. เข้าใจถึงวิธีการในการพัฒนาคอมโพเนนท์ซอฟต์แวร์ประเภท เอาท์ – โพรเซส
2. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภท เอาท์ – โพรเซส จาก เครื่องมือพัฒนาตัวเดียวกัน
3. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภท เอาท์ – โพรเซส จาก เครื่องมือพัฒนาอื่น

เครื่องมือที่ใช้พัฒนา

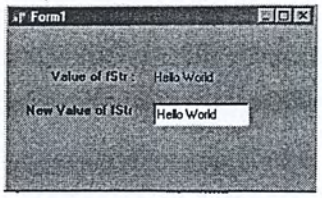
- Borland Delphi 5.0
- Microsoft Visual Basic 6.0

คำอธิบายโปรแกรม

- เป็นคอมโพเนนท์ออบเจ็กต์ ประเภท เอาท์-โพรเซส ซึ่งมี ซึ่งติดต่อกับโปรแกรมประยุกต์ที่เก็บค่าตัวแปรชนิดสตริงเอาไว้โดยจะมีอินเตอร์เฟส ที่ติดต่อกับโปรแกรมนี้ภายในอินเตอร์เฟสจะมี เมธอด 2 เมธอด ซึ่งได้แก่ในการรับค่าสตริงจากไคลเอนต์ และเมธอดในการส่งค่าสตริงคืนไคลเอนต์

5.2.1 ขั้นตอนการพัฒนา

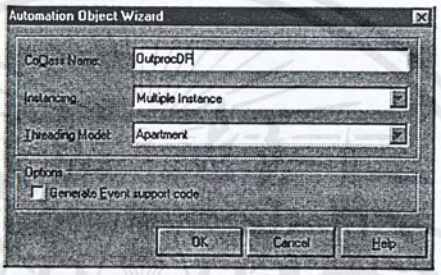
1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File* > *New* แล้วเลื่อนไปที่หน้า Application จากหน้า New
2. แต่งให้โปรแกรมประยุกต์ที่สร้างใหม่มี ลักษณะดังรูปโดยมีฟิลด์ภายใน fStr เป็นชนิดสตริงโดยมีอนุญาติให้มีการเปลี่ยนแปลงค่าโดยมีค่าเท่าสตริงที่อยู่ในอิดิทบอกซ์ ดังรูปที่ 5-4



รูปที่ 5-4 แสดงรายละเอียดของตัวเอาท์-โพรเซส เซิร์ฟเวอร์

3. บันทึกโปรแกรมประยุกต์นี้ไว้ในไฟล์ OutprocDPSrvMainFrm.pas และไฟล์ของโปรเจ็กต์เก็บไว้ที่ OutprocDPSrv.dpr

4. สร้างคอมโพเนนท์ออบเจ็กต์ใหม่ โดยเลือก *File* > *New* แล้วเลื่อนไปที่หน้า ActiveX แล้วเลือก Automation Object จะขึ้น Automation Object Wizard ไดอะล็อกใส่ข้อมูลจนมีลักษณะดังรูปที่ 5-5



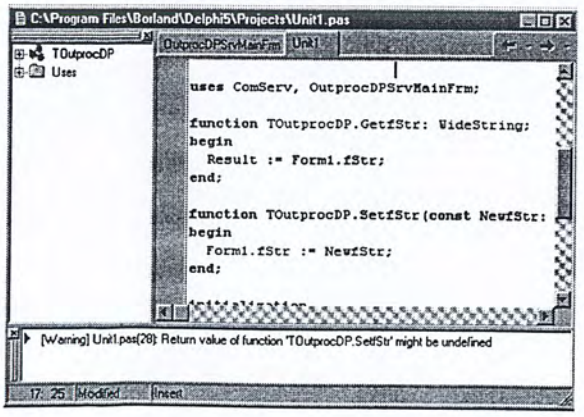
รูปที่ 5-5 แสดงหน้าจอของ Automation Object Wizard ในการสร้าง เอาท์-โพรเซส คอมโพเนนท์

5. กด OK จะทำให้เกิดการเรียกใช้ type library editor ให้เพิ่มเมธอด ลงไปใน IOutprocDP ดังนี้

- เมธอด Setf Str มี return value เป็น HResult, มีพารามิเตอร์ ชื่อ NewfStr มีชนิดเป็น BSTR ซึ่งมี Modifier เป็น [in]
- เมธอด Getf Str มี return value เป็น HResult, มีพารามิเตอร์ชื่อ Result มีชนิดเป็น BSTR * ซึ่งมี Modifier เป็น [out, retval]

6. กด Refresh เพื่อทำการเปลี่ยนข้อมูลที่กำหนดไว้ใน type library editor ให้เป็น ซอร์สโค้ด

7. ทำการ เพิ่มชื่อ OutprocDPSrvMainFrm ลงในส่วนการประกาศ uses ของส่วน interface และใส่ซอร์สโค้ดสำหรับเมธอดทั้งสองของ IOutprocDP ดังรูปที่ 5-6



รูปที่ 5-6 แสดงรายละเอียดของเมธอดในตัวเอาท์-โพรเซสคอมโพเนนท์

8. ทำการบันทึกไฟล์ของ IOutprocDP ไว้ที่ OutprocIntf.pas
9. ทำการรีจิสเตอร์อินเตอร์เฟสโดยเลือก Run โปรเจ็กต์และจะสังเกตพบว่ามีการสร้างไฟล์ที่ชื่อว่า OutprocDPSrv_TLB.pas ขึ้นมาด้วย
10. ทำการสร้าง OutprocDPSrv.exe โดยเลือก *Project > Build OutprocDPSrv*
11. เสร็จสิ้นการพัฒนาออบเจ็กต์ คอมโพเนนท์ประเภท เอาท์-โพรเซส

5.2.2 ขั้นตอนการเรียกใช้งาน ด้วย Borland Delphi 5.0

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File > New* แล้วเลือก Application จากหน้า New
2. อิมพอร์ตไฟล์ที่เป็น TLB ซึ่งในที่นี้คือ OutprocIntf_TLB.pas โดยเลือก *Project > Add to Project* แล้วเลือกไฟล์ OutprocIntf_TLB.pas จากตำแหน่งไดเรกทอรี่ที่บันทึกไว้หรือในไดเรกเทอรี่ที่ได้ copy เฉพาะส่วนที่เป็น ไฟล์ TLB อย่างเดียวก็ได้
3. เพิ่มการเรียกใช้ OutprocIntf_TLB และ COMObj ในส่วนของ uses เช่น

```
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    COMObj, OutprocDP_TLB, StdCtrls;
```

4. ประกาศตัวแปรประเภท IOutprocDP ในส่วน Private
5. เมื่อจะทำการเรียกใช้ตัวแปรที่เป็นคอมโพเนนท์ จะต้องมีการสร้างตัวแปรที่เป็นคอมโพเนนท์โดยใช้คำสั่ง

```
a := CreateCOMObject(Class_xxxx) as IntfName;
```

โดยมีรายละเอียดเหมือนในการเรียกใช้ คอมโพเนนท์ออบเจ็กต์ชนิด อิน-โพรเซส เช่น

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    a := CreateCOMObject(Class_OutprocDP) as IOutprocDP;
end;
```

6. หลังจากนั้นจะสามารถใช้งานอินแสตนซ์ของ คอมโพเนนท์ได้เหมือนกับ ออบเจ็กต์ทั่วๆไป

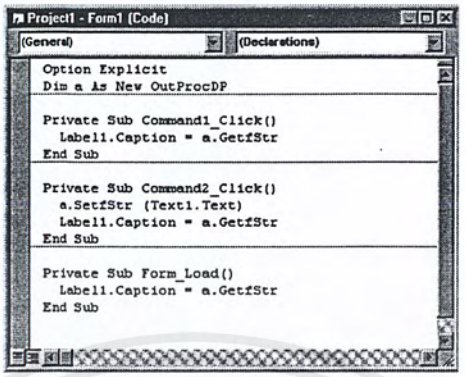
5.2.3 ขั้นตอนการเรียกใช้งาน ด้วย Microsoft Visual Basic 6.0

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File > New Project* แล้วเลือก Standard EXE
2. ทำการอ้างอิงไปยังไฟล์ที่เป็นเอาท์-โพรเซส โดยทำการเลือก *Project > References* จากนั้นให้ใช้ปุ่ม Browse เพื่อหาไฟล์ EXE ของเอาท์-โพรเซส เซริฟเวอร์
3. ทำการปรับแต่งและเพิ่มเติม องค์ประกอบของโปรแกรมใหม่นี้ โดยการเรียกใช้เอาท์-โพรเซสจะใช้คำสั่ง ในลักษณะ

```
Dim a As New OutProcDP เพื่อทำการสร้างอินแสตนซ์ใหม่
```

Label1.Caption = a.GetfStr เพื่อเรียกใช้เมธอดของอินเตอร์เฟส

ซึ่งการอาจเป็นดังในรูปที่ 5-7



รูปที่ 5-7 แสดงตัวอย่างการเรียกใช้งานเอาท์-โพรเซสคอมโพเนนท์

4. ทำการเรียกใช้งานได้เหมือนกับออบเจ็กต์ทั่วๆ ไป

ข้อสังเกต

1. ในการใช้ เอาท์โพรเซสเซิร์ฟเวอร์นั้นเมื่อตัวไคลเอนต์ทำการรันโปรแกรมจะส่งผลให้ตัวเซิร์ฟเวอร์นั้นรันโดยอัตโนมัติ
2. การเรียกใช้คอมโพเนนท์ทั้งในรูปแบบ อิน-โพรเซสและ เอาท์-โพรเซส จะมีลักษณะการอ้างอิง และการเรียกใช้เมธอดที่อยู่ภายในคอมโพเนนท์เหมือนกัน

5.3 กรณีศึกษาการพัฒนา คอมโพเนนท์ ออบเจ็กต์ ประเภท รีโมท เซิร์ฟเวอร์ (Remote Server)

จุดประสงค์

1. เข้าใจถึงวิธีการในการพัฒนาคอมโพเนนท์ซอฟต์แวร์ประเภท รีโมทเซิร์ฟเวอร์
2. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภท รีโมท เซิร์ฟเวอร์ จาก เครื่องมือพัฒนาตัวเดียวกัน
3. เข้าใจถึงวิธีการเรียกใช้คอมโพเนนท์ซอฟต์แวร์ประเภทรีโมท เซิร์ฟเวอร์ จาก เครื่องมือพัฒนาอื่น

เครื่องมือที่ใช้พัฒนา

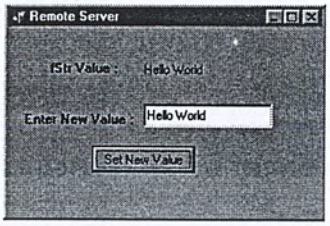
- Borland Delphi 5.0
- Microsoft Visual Basic 6.0

คำอธิบายโปรแกรม

- เป็นคอมโพเนนท์ออบเจ็กต์ ประเภท รีโมท เซิร์ฟเวอร์ ซึ่งมี อินเตอร์เฟส 1 อินเตอร์เฟส
- ภายในอินเตอร์เฟสจะมี เมธอด 2 เมธอด ซึ่งได้แก่ในการรับค่าสตริงจากไคลเอนต์ และเมธอดในการส่งค่าสตริงคืนไคลเอนต์

5.3.1 ขั้นตอนการพัฒนา

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File* > *New* แล้วเลื่อนไปที่หน้า Application จากหน้า New
2. แต่งให้โปรแกรมประยุกต์ที่สร้างใหม่มี ลักษณะดังรูปโดยมีฟิลด์ภายใน fStr เป็นชนิดสตริงโดยมีอนุญาติให้มีการเปลี่ยนแปลงค่าโดยมีค่าเท่าสตริงที่อยู่ในอิดิทบอกซ์ ดังรูปที่ 5-8

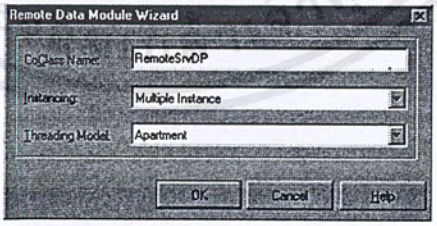


รูปที่ 5-8 แสดงลักษณะของรีโมท เซิร์ฟเวอร์

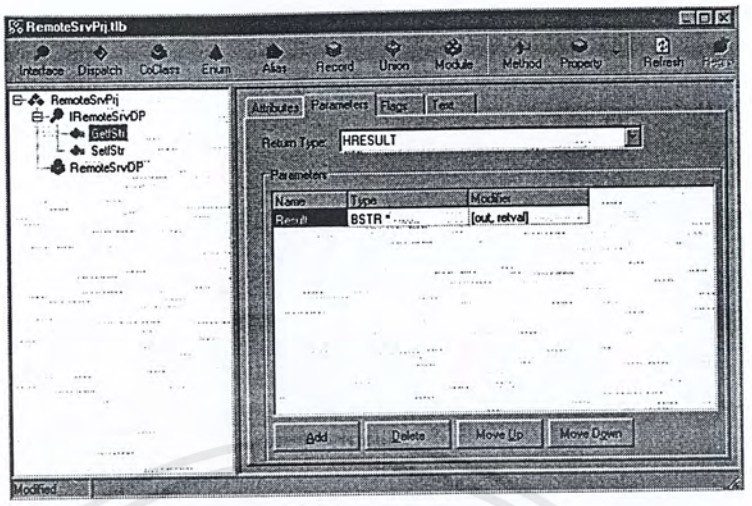
3. บันทึกโปรแกรมประยุกต์นี้ไว้ในไฟล์ RemoteSrvMainFrm.pas และไฟล์ของโปรเจ็กต์เก็บไว้ที่ RemoteSrv.dpr
4. เพิ่ม Remote Data Module โดยเลือก *File* > *New* หลังจากนั้นเลือก Remote Data Module จากหน้า Multitier ตัว Remote Data Module ที่สร้างขึ้นนั้นเป็นตัวอินเตอร์เฟสด้วย มันจะถูกเรียกใช้โดยตัวไคลเอนต์ตัวคอมโพเนนท์เซิร์ฟเวอร์ตัวนี้ เมื่อเลือกสร้าง Remote Data Module จะขึ้นหน้าจอของ Remote Data Module Wizard ให้ใส่รายละเอียดที่ต้องการไป ดังรูปที่ 5-9 หลังจากทำการปรับแต่งจนเหมาะสมแล้วให้กดปุ่ม OK เพื่อทำงานต่อ
5. กด OK จะทำให้เกิดการเรียกใช้ type library editor ให้เพิ่มเมธอด ลงไปใน IOutprocDP ดังนี้

- เมธอด Setf Str มี return value เป็น HResult, มีพารามิเตอร์ ชื่อ NewfStr มีชนิดเป็น BSTR ซึ่งมี Modifierเป็น [in]

  เมธอด Getf Str มี return value เป็น HResult, มีพารามิเตอร์ ชื่อ Result มีชนิดเป็น BSTR * ซึ่งมี Modifierเป็น [out, retval]ดังรูปที่ 5-10

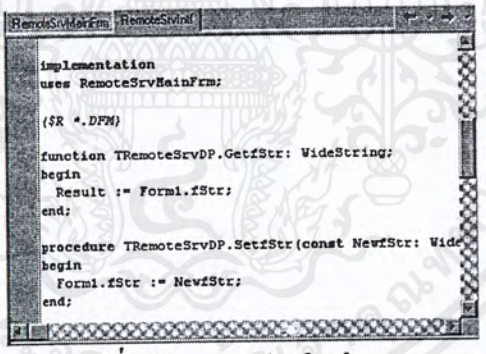


รูปที่ 5-9 แสดงหน้าจอของ Remote Data Module ในการสร้างรีโมท เซิร์ฟเวอร์



รูปที่ 5-10 แสดงหน้าจอของ type library editor ในการเพิ่ม เมธอดและคุณสมบัติต่างๆ ให้กับอินเตอร์เฟส

6. กด Refresh เพื่อทำการเปลี่ยนข้อมูลที่กำหนดไว้ใน type library editor ให้เป็น ซอร์สโค้ด
7. ทำการ เพิ่มชื่อ RemoteSrvMainFrm ลงในส่วนการประกาศ uses ของส่วน interface และใส่ซอร์สโค้ดสำหรับเมธอดทั้งสองของ IRemoteSrvDP ดังรูปที่ 5-11



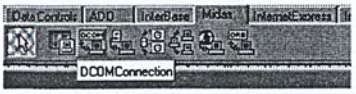
รูปที่ 5-11 แสดงการเพิ่มเติมรายละเอียดในส่วน Remote Data Module

8. ทำการบันทึกไฟล์ของ IRemoteSrvDP ไว้ที่ RemoteSrvDPIntf.pas
9. ทำการรีจิสเตอร์อินเตอร์เฟสโดยเลือก Run โปรเจ็กต์และจะสังเกตพบว่ามีการสร้างไฟล์ที่ชื่อว่า RemoteDPSrvPrj_TLB.pas ขึ้นมาด้วย
10. ทำการสร้างไฟล์ RemoteDPSrvPrj_TLB.pas โดยการเลือก *Project > Build RemoteSrvPrj*
11. เสร็จสิ้นการพัฒนาออบเจ็กต์ คอมโพเนนท์ประเภท รีโมท เซิร์ฟเวอร์

5.3.2 ขั้นตอนการเรียกใช้งาน ด้วย Borland Delphi 5.0

1. สร้างโปรเจ็กต์ใหม่โดยเลือก *File > New* แล้วเลือก Application จากหน้า New
2. ทำการรีจิสเตอร์ตัวเซิร์ฟเวอร์ ไว้ในเครื่องของไคลเอนต์โดยการ รันตัวโปรแกรมเซิร์ฟเวอร์แล้วต่อท้ายด้วย / regserver เช่น \\Chead\c\Project\Report\RemoteDP\RemoteSrvPrj.exe /regserver

3. ทำการเพิ่มออบเจ็กต์ชนิด DCOMConnection จากหน้า Midas ลงในโปรเจ็กต์ ดังรูปที่ 4-12 ออบเจ็กต์ตัวนี้จะทำหน้าที่เป็นตัวกลางในการติดต่อระหว่างเซิร์ฟเวอร์ และไคลเอนต์ดังจะได้อธิบายต่อไป



รูปที่ 5-12 แสดงการเลือก DCOMConnect คอมโพเนนท์

4. ทำการเลือกตัวเซิร์ฟเวอร์ที่ต้องการจะติดต่อผ่านตัว DCOMConnection โดยการตั้งค่าคุณสมบัติ ServerName ดังรูปที่ 5-13

รูปที่ 5-13 แสดงการเลือกตัวรีโมท เซิร์ฟเวอร์ในคุณสมบัติ ServerName

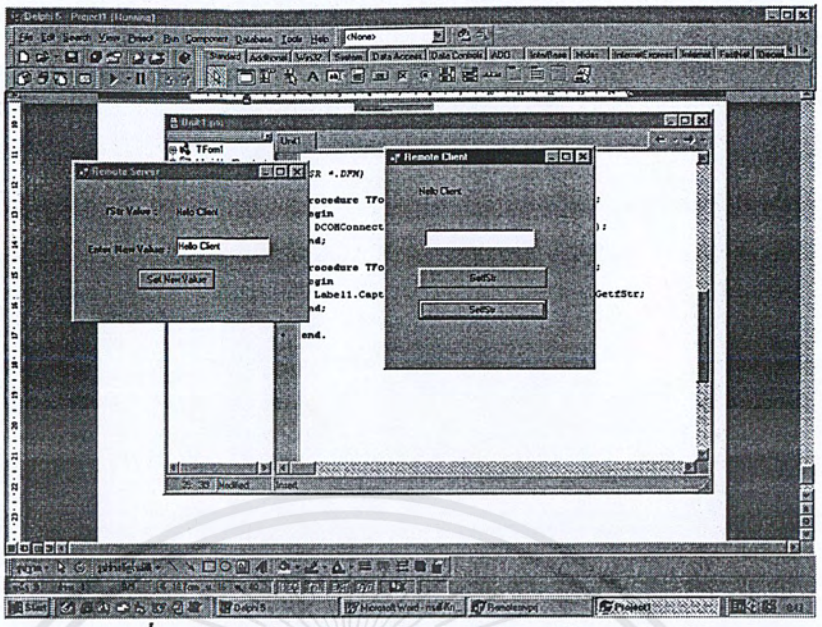
5. ทำการตั้งค่า Connection ให้เป็น true จะสังเกตได้ว่าตัวเซิร์ฟเวอร์นั้นจะถูกเรียกใช้งานโดยอัตโนมัติ
6. ในการเรียกใช้เซิร์ฟเวอร์นั้นไม่จำเป็นต้องสร้างออบเจ็กต์ใหม่แต่ อย่างใดแต่จะเรียกใช้ผ่านทางตัว DCOMConnection.AppServer แทน เช่นหากตัว DCOMConnection มีชื่อว่า DCOMConnection1 ก็สามารถเรียกใช้เมธอด GetfStr ได้ดังนี้

```
DCOMConnection1.AppServer.SetfStr('Hello World');
```

7. หลังจากนั้นจะสามารถใช้งานอินแสตนซ์ของ คอมโพเนนท์ได้เหมือนกับ ออบเจ็กต์ทั่วๆไป

<u>ข้อสังเกต</u>

1. เมื่อมีการเรียกตัวไคล์เอนต์ทำงานตัวเซิร์ฟจะถูกเรียกใช้งานโดยอัตโนมัติ ดังรูปที่ 4-14
2. หากก่อนการกำหนดค่าของ DCOMConnection นั้นไม่ได้มีการรีจิสเตอร์ตัวเซิร์ฟเวอร์ในเครื่องไคลเอนต์ จะต้องกำหนดค่า ComputerName ของเครื่องServer เสียก่อนจึงจะเรียกใช้ได้ เนื่องจากการใส่ค่า ComputerName จะมีความหมายว่าตัวเซิร์ฟเวอร์นั้นอยู่ในเครื่องไคลเอนต์ หรือ ตัวเซิร์ฟเวอร์นั้นได้ถูกรีจิสเตอร์ไว้ในเครื่องไคลเอนต์แล้ว
3. ในการใช้งานเมธอดของรีโมทเซิร์ฟเวอร์บนเครื่องไคลเอนต์โดยผ่าน DCOMConnection นั้นจะไม่สามารถใช้ type library ผ่านทางปุ่ม Ctrl + Space เพื่อดูรายละเอียดของเมธอดภายในตัวเซิร์ฟเวอร์ได้



รูปที่ 5-14 แสดงการรันไคลเอนต์จะให้ตัวเซิร์ฟเวอร์รันด้วย

บทที่ 6

ActiveX Data Object (ADO)

ActiveX Data Object เป็นเซตของคอมโพเนนต์ใน Universal Data Access ช่วยให้โปรแกรมเมอร์เข้าถึงข้อมูลได้ ทั้งแบบที่เป็น Relational Database และ โครงสร้างข้อมูลแบบอื่นๆ เช่น เมล์

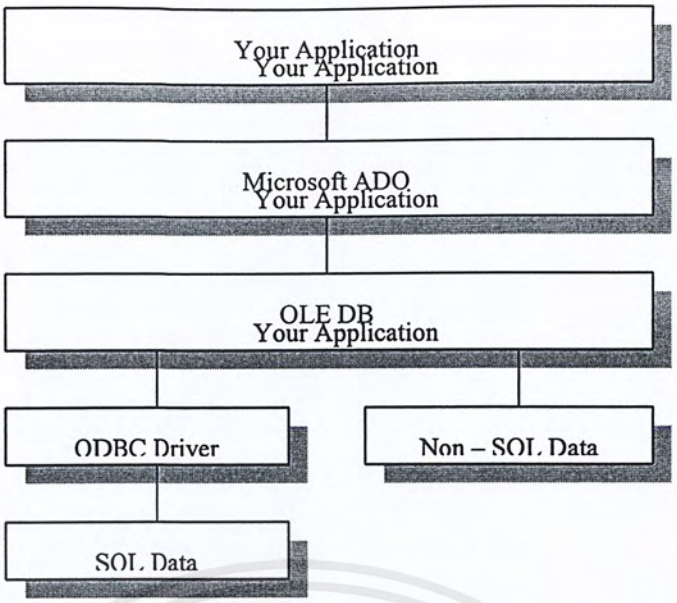
ในการใช้งานเราจะใช้ ADO ติดต่อกับ OLE DB เพื่อให้การทำงานมีประสิทธิภาพสูงขึ้น เพราะ OLE DB มีความสามารถสูงแต่การใช้งานนั้นใช้ได้ยาก จึงใช้ผ่าน ADO ซึ่งการใช้งาน ADO นั้นใช้ได้ง่ายกว่ามาก การทำงานของ ADO นั้นใช้แนวคิดของ Provider ในการเข้าถึงข้อมูล เช่น ถ้าเราใช้ ODBC ตัว Provider ก็เหมือนกับ ODBC Driver ตัวที่เป็น Provider จะต้องเผยให้เห็นถึงเซตของอินเตอร์เฟสให้ ADO สามารถเข้าถึง ทำให้ ADO สามารถเข้าถึงข้อมูลที่แตกต่างกันได้อย่างถูกต้อง

6.1 ADO และ OLE DB

ADO เตรียม API ให้โปรแกรมเมอร์ใช้ในการทำงานได้อย่างมีประสิทธิภาพ และมีความเป็นไปได้ว่าการพัฒนาในอนาคตจะหันมาใช้ ADO กันอย่างแพร่หลาย

การใช้ ADO นั้นมีข้อได้เปรียบมากมาย เช่น ความเร็วสูง, ใช้หน่วยความจำน้อย, ประหยัดพื้นที่, ใช้เครือข่ายอย่างประหยัด, มีอินเตอร์เฟสที่มีประสิทธิภาพสูง จึงเป็นทางเลือกหนึ่งที่ดี ในการทำงาน ADO จะไปเรียก OLE DB ตัว OLE DB นี้ช่วยให้การเข้าถึงข้อมูลได้หลากหลายมากขึ้น แม้ว่าการใช้ ODBC จะมีความสามารถในการเข้าถึงข้อมูลได้หลายรูปแบบในระบบของ ดาตาเบส ต่างๆ แต่ก็มีข้อจำกัดเนื่องจาก ODBC ไม่ได้เตรียมการเข้าถึงของมูลไว้ทุกรูปแบบ จึงมีการใช้ OLE DB มาช่วยในการทำงานทำให้การเข้าถึงข้อมูลทำได้มากขึ้นคือช่วยให้สามาเข้าถึงข้อมูลแบบที่ไม่ใช่ Relation ได้ด้วย แต่การใช้ OLE DB ก็มีปัญหาเหมือนกันคือการใช้งาน OLE DB นั้นทำได้ยาก มีความซับซ้อนในการใช้งานสูง และเพื่อการใช้งาน OLE DB ให้ได้มีประสิทธิภาพที่ดี และใช้ได้ง่ายขึ้นจึงใช้ ADO มาติดต่อกับ OLE DB อีกที

ADO จึงเป็นทางเลือกที่ดีสำหรับการเขียนโปรแกรมระดับสูงโดยผู้ใช้ไม่จำเป็นต้องเข้าใจการใช้ COM อินเตอร์เฟส ในระดับล่างๆ ทำให้การใช้งายเป็นไปได้ง่าย



รูปที่ 6-1 แสดงการทำงานของ ADO

6.2 ออบเจ็กต์ ADO

สถาปัตยกรรมของ ADO ประกอบไปด้วยออบเจ็กต์เพียงไม่กี่ตัวที่สำคัญ ดังนี้

- Connection เป็นออบเจ็กต์ที่ใช้ในการติดต่อกับแหล่งข้อมูล โดยใช้ Connection strings มาเป็นตัวบอกว่าจะติดต่อกับข้อมูลชนิดใด ช่วยจัดการในการเชื่อมต่อและดูแลทรานแซคชั่น
- Command จัดการกับแหล่งข้อมูล เช่นการ เพิ่ม, แก้ไข, ลบ ข้อมูล พูดง่ายๆ คือจัดการกับข้อมูลโดยไม่ต้องการค่าส่งคืนกลับ
- Recordset เป็นออบเจ็กต์ที่เก็บค่าส่งคืนมาจากการ Query

ADO ยังมีออบเจ็กต์อื่นๆ อีก คือ Error เป็นออบเจ็กต์ที่จัดการเกี่ยวกับความผิดพลาด, Property เป็นออบเจ็กต์ที่บอกถึงคุณสมบัติของตัวมันเอง เป็นต้น

6.3 คอมโพเนนต์ของ ADO ใน Borland Delphi 5.0

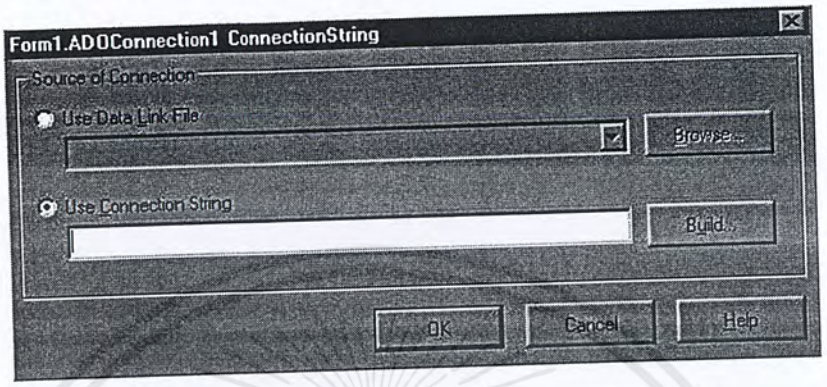
คอมโพเนนต์ ของ ADO ใน Delphi สืบทอดมาจากคลาส TDataset มีอยู่สามคอมโพเนนต์ที่สามารถใช้ในการเขียนแอปพลิเคชั่นที่ทำงานกับดาตาเบส ดังนี้

- TADOConnection เป็นคอมโพเนนต์ที่หุ้มออบเจ็กต์ ADO Connection ไว้ ช่วยในการสร้าง Connection strings, การ login, ทรานแซคชั่น คล้ายๆ กับ TDatabase
- TADOCommand เป็นคอมโพเนนต์ที่หุ้มออบเจ็กต์ ADO Command ไว้ ทำงานแบบที่ไม่ต้องการค่าส่งคืนกลับ
- TADODataSet เป็นคอมโพเนนต์ที่หุ้มออบเจ็กต์ สองตัวด้วยกันคือ ADO Command และ ADO Recordset ไว้ด้วยกันใช้ในการออกคำสั่งจัดการกับตารางหนึ่งตาราง หรือหลายๆ ตารางก็ได้ และรับเซตของเรคอร์ดที่ส่งคืนกลับมา

ทั้งสามคอมโพเนนต์จำเป็นในการสร้างแอปพลิเคชั่นด้วย Delphi แต่ก็มีคอมโพเนนต์อื่นๆ ให้ใช้ได้อีกนอกจากสามตัวที่ได้กล่าวไป คือ TADOTable, TADOQuery, TADOStoredProc

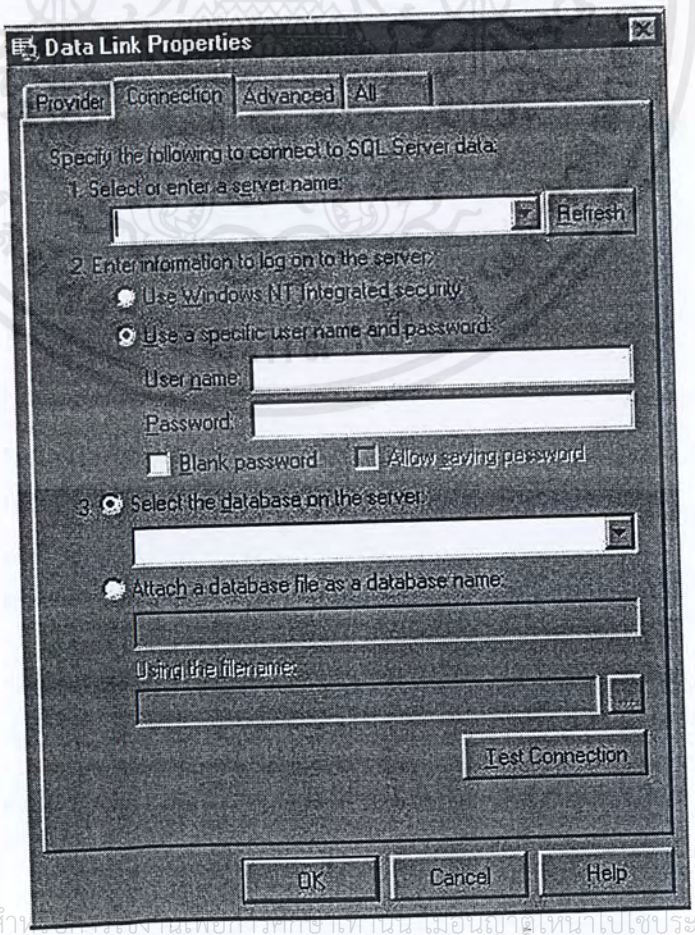
6.4 การใช้คอมโพเนนต์ ADO ใน Borland Delphi 5.0

ได้กล่าวไปแล้วถึงแนวคิดของ ADO ต่อจากนี้เป็นการใช้งานคอมโพเนนต์ในการสร้างแอปพลิเคชั่นขึ้นมาให้ติดต่อกับดาตาเบสได้เริ่มจากการใช้ TADOConnection ในการติดต่อไปยังดาตาเบสที่เราต้องการโดยการเลือก Connection strings ดังรูปที่ 6-2



รูปที่ 6-2 แสดงหน้าจอในการเลือกสร้าง Connection String

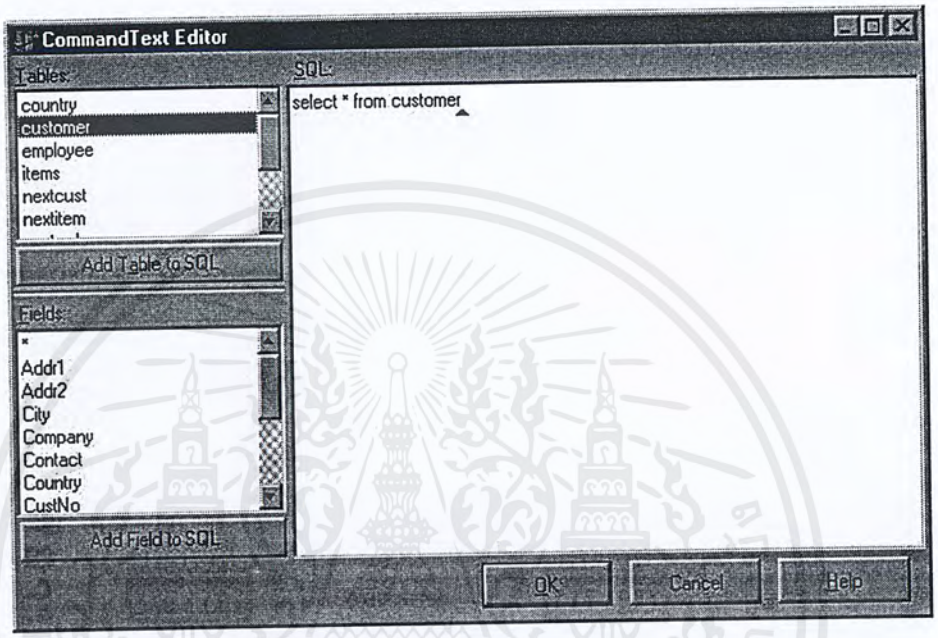
เลือก Use Connection string จะมีกล่องข้อความใหม่ปรากฏขึ้นมาใหม่ให้เลือก OLE DB Provider ที่เราต้องการใช้งาน เช่น Provider สำหรับ Oracle, Provider สำหรับ SQL Server เป็นต้น ต่อไปก็เป็นการกำหนดเงื่อนไขในการติดต่อ เช่น ชื่อเซิร์ฟเวอร์, ชื่อผู้ใช้, ชื่อดาตาเบส ดังรูปที่ 6-3



รูปที่ 6-3 แสดงหน้าจอในการสร้าง Connection String

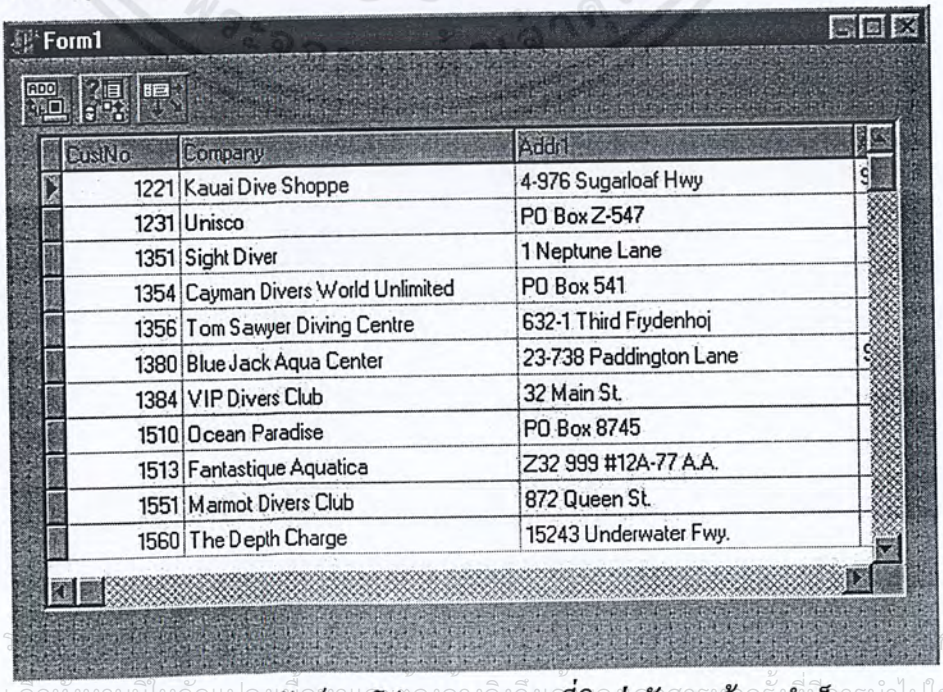
หลังจากตั้งค่าของ ADOConnection เสร็จเรียบร้อยแล้วเราสามารถเพิ่ม TADODataSet เข้ามาใช้งานโดยตั้งค่า Connection เป็น ADOConnection ตัวคอมโพเนนต์ ADODataSet ก็จะติดต่อดาตาเบสผ่านทาง ADOConnection ได้โดยกำหนดค่า CommandText เมื่อเลือกไปจะเห็น CommandText Editor ปรากฏขึ้นมา ดังรูปที่ 6-4

เราสามารถใส่คำสั่ง SQL ลงไปได้ตามต้องการต่อจากนั้น ก็ใช้ DataSource และ DBGrid ในการ



รูปที่ 6-4 แสดงหน้าจอในการสร้างคำสั่งภาษา SQL

แสดงเซตของข้อมูลที่ได้จากการ Query โดยกำหนดค่า DataSet ของ DataSource เป็น ADODataSet และกำหนดค่า DataSource ของ DBGrid เป็น DataSource เท่านี้ก็จะเห็นเซตของข้อมูลที่เรา Query ออกมาดังรูปที่ 6-5



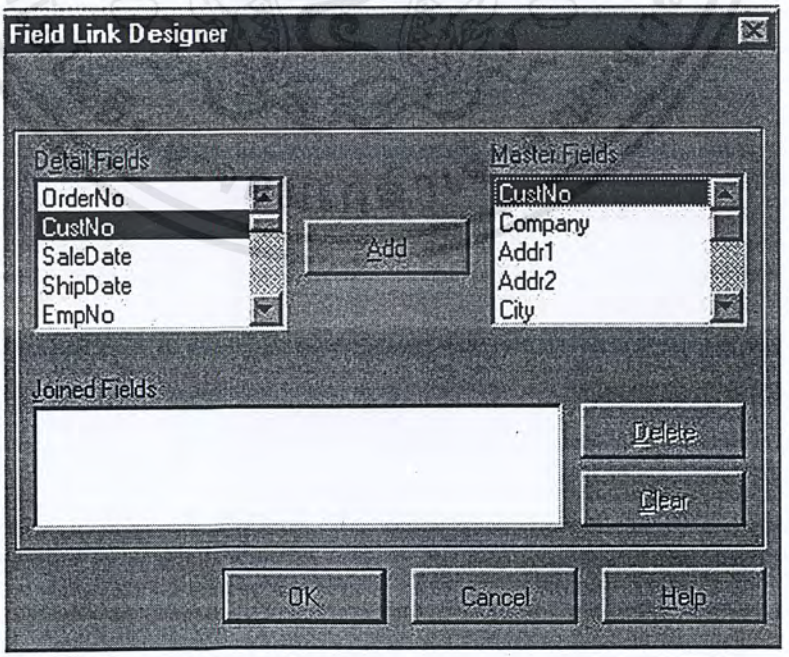
รูป 6-5 แสดงผลลัพธ์ จากโปรแกรมทดสอบที่ติดต่อกับฐานข้อมูลสำเร็จ

6.5 โครงสร้างแบบ Master/Detail

คอมโพเนนต์ ADODataset สนับสนุนการทำงานให้มีความสัมพันธ์แบบ Master/Detail รูปแบบการทำงานแบบ Master/Detail ก็คือการที่มีตารางหนึ่งเป็นตาราง Master มีความสัมพันธ์กับตารางอื่นที่เป็นตาราง Detail ซึ่งอาจมีได้มากกว่าหนึ่งตารางโดยที่ตารางที่เป็น Detail นั้นต้องมี Foreign key ที่เป็น Primary key ของตารางที่เป็น Master อยู่ในตารางด้วยทุกตาราง เพื่อเป็นตัวอ้างอิงในการรวมตาราง (Join) เข้าด้วยกันได้ถูกต้อง ตัวอย่างของการแสดงข้อมูลที่สามารถเป็นแบบ Master/Detail ได้ เช่น การเก็บข้อมูลของนักศึกษา มีตาราง Master เป็นข้อมูลเกี่ยวกับตัวนักศึกษาเอง รูปแบบของตาราง Master อาจมี รหัสนักศึกษา ชื่อ นามสกุล วัน/เดือน/ปี เกิด ที่อยู่ คณะ ภาควิชา เป็นต้น ส่วนตารางที่เป็น Detail ก็อาจมีหลายตาราง อย่างเช่น ตารางของการลงทะเบียนเรียน ตารางของผลการเรียน เป็นต้น เมื่อใช้โครงสร้างแบบ Master/Detail มาแสดงก็จะเห็นข้อมูลของนักศึกษาแต่ละคนได้อย่างชัดเจน คือเมื่อตารางที่เป็น Master มีการเลือกไปที่ข้อมูลในตาราง จะทำให้ตารางที่เป็น Detail เกิดการนำข้อมูลที่เกี่ยวกับนักศึกษาคนที่ถูกเลือกนำขึ้นมาแสดง โดยการรวมตาราง (Join) ตารางทุกตารางที่เป็น Detail ทำให้การแสดงข้อมูลเป็นไปได้ง่าย การทำงานแบบ Master/Detail นั้นเราสามารถใช้คำสั่งในการรวมตารางเองก็ได้

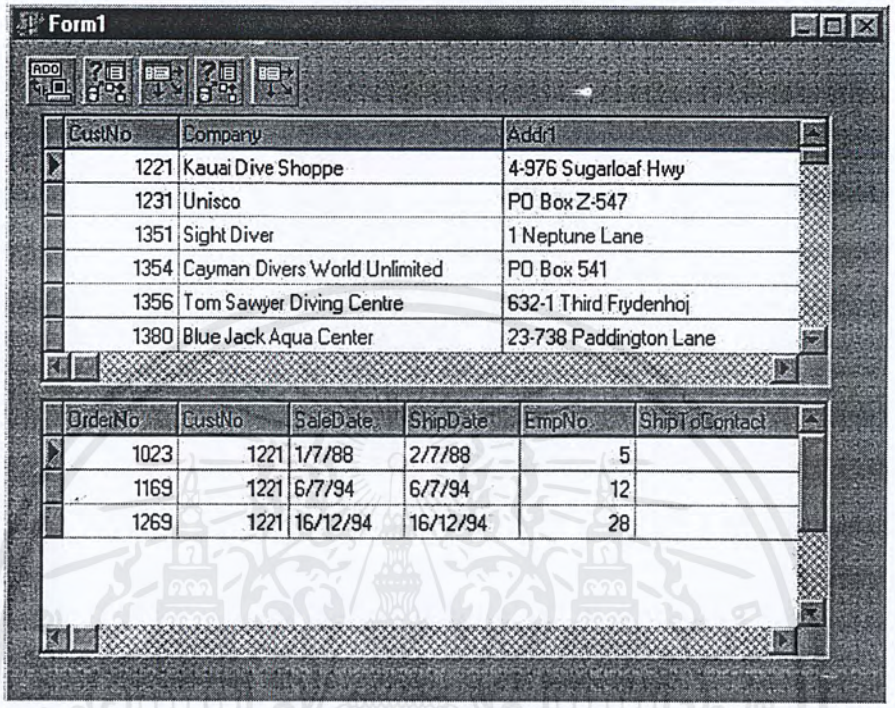
6.6 การใช้โครงสร้างแบบ Master/Detail ใน Borland Delphi 5.0

เริ่มแรกต้องมีตารางที่เป็น Master และ Detail กำหนดค่า CommandType ของ ADODataSet ของทั้งสองตารางให้เป็น cmdTable และกำหนด CommandText เป็นชื่อตารางที่ต้องการ ตารางที่เป็น Master กำหนดเสร็จเรียบร้อยแล้ว ที่เหลือตั้งค่า DataSource ของ ADODataSet ที่เป็น Detail ให้เป็น DataSource ของตาราง Master เสร็จแล้วกำหนดค่า MasterFields จะปรากฏกล่องข้อความดังรูปที่ 6-6



รูปที่ 6-6 แสดง หน้าจอของโปรแกรมตัวอย่าง

เลือก Key ที่จะนำมาใช้เป็นตัวกลางในการรวมตารางเข้าด้วยกัน กด Add เป็นการเสร็จการกำหนดค่าของตาราง Detail เสร็จแล้วจะได้แอปพลิเคชั่นง่าย ดังรูปที่ 6-7



รูปที่ 6-7 แสดงผลลัพธ์ของการทดสอบลักษณะ Master/Detail

จะเห็นได้ว่าตารางที่เป็น Detail (ด้านล่าง) จะขึ้นกับตารางที่เป็น Master (ด้านบน) ทั้งสองตารางสัมพันธ์กันด้วย CustNo

6.7 การจัดการทรานแซคชันด้วย ADO

ในการทำงานเกี่ยวกับดาตาเบสเราอาจจะใช้ทรานแซคชั่นเข้ามาช่วย ทรานแซคชั่นเป็นการทำงานที่เป็นหนึ่งเดียวจะแบ่งแยกไม่ได้ พูดง่ายๆ ก็คือ การทำงานที่เป็นทรานแซคชั่นถ้าทำแล้วต้องทำให้เสร็จทั้งหมดถ้าทำไม่เสร็จต้องเหมือนไม่เคยทำเลยยกเลิกที่ทำไม่เสร็จให้หมด

ใน ADO เองมีการจัดการการทำทรานแซคชันให้ได้ คือสามารถทำการ Start Transaction, Commit Transaction, Rollback Transaction ได้ช่วยให้การทำงานมีความถูกต้อง และความปลอดภัยมากขึ้นในการใช้งานคำสั่งเกี่ยวกับการจัดการทรานแซคชั่นนั้นมีเมธอดอยู่ในคอมโพเนนต์ ADOConnection ให้เราใช้คือ BeginTrans, CommitTrans, RollbackTrans ทั้งสามตัวใช้ในการกำหนด ทรานแซคชัน

บทที่ 7
อีเวนต์ และ คอลแบ็คของ COM

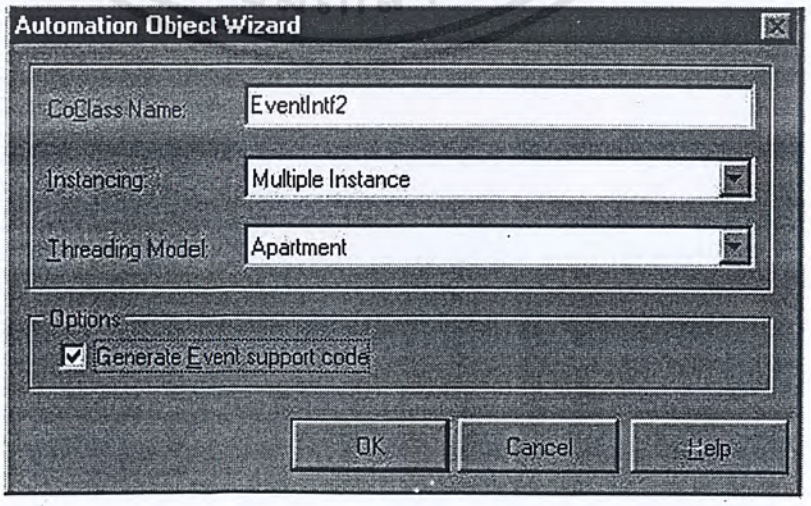
ในรูปแบบของโมเดลแบบไคลเอนต์/เซิร์ฟเวอร์ ตัวไคลเอนต์จะเรียกไปยังเซิร์ฟเวอร์เพื่อทำงานบางอย่างหรือเรียกดูข้อมูล แล้วตัวเซิร์ฟเวอร์จะทำการประมวลงานนั้น หรือส่งข้อมูลไปให้ไคลเอนต์ซึ่งเป็นรูปแบบที่ดี แต่มีปัญหาเมื่อมีอีเวนต์ในฝั่งไคลเอนต์เป็นแบบอะซิงโครนัส หรือทำงานตามการใช้งานของผู้ใช้ ตัวอย่างเช่น ไคลเอนต์ร้องขอไปยังเซิร์ฟเวอร์เพื่อดาวน์โหลดไฟล์ตัวไคลเอนต์นั้นไม่ต้องการรอจึงมีรูปแบบในการใช้อีเวนต์เกิดขึ้นโดยตัวไคลเอนต์จะทำการร้องขอไปที่เซิร์ฟเวอร์แล้วกลับมาทำงานต่อได้จนกระทั่งเซิร์ฟเวอร์แจ้งกลับมาที่ไคลเอนต์ว่าการดาวน์โหลดเสร็จแล้ว หรืออีกตัวอย่างหนึ่งคือการใช้งานของผู้ใช้ เช่นการกดปุ่มจะเกิดอีเวนต์ขึ้นบอกไปยังเซิร์ฟเวอร์แล้วเซิร์ฟเวอร์ทำการเรียกเมธอดที่สนับสนุนอีเวนต์นั้นให้

ลักษณะของออโตเมชันอีเวนต์นั้นเป็นโมเดลที่มีประสิทธิภาพแต่ซับซ้อนการใช้อีเวนต์นั้นจะเรียกผ่านอินเตอร์เฟส โดยอินเตอร์เฟสนี้เรียกว่า อีเวนต์อินเตอร์เฟส หรือ เอาท์โกอิงอินเตอร์เฟส (Outgoing Interface) ที่เรียกว่า เอาท์โกอิงอินเตอร์เฟสเพราะว่าการเขียนเมธอดไม่ได้ทำที่เซิร์ฟเวอร์เหมือนอินเตอร์เฟสอื่นๆ แต่จะทำที่ไคลเอนต์ของเซิร์ฟเวอร์นั้นๆ โดยอินเตอร์เฟสนี้จะเหมือนกับอินเตอร์เฟสทั่วๆไปเช่นมี IID เป็นต้น

ใน Delphi เราจะใช้ DispInterface ในการสร้างอีเวนต์ เพราะสามารถเข้ากันได้กับ Visual Basic ได้ ถ้าเราใช้อินเตอร์เฟสธรรมดาในการสร้างอีเวนต์จะทำงานได้เร็วขึ้นกว่า DispInterface แต่ไม่สามารถเข้ากับ Visual Basic ได้ และต้องเขียนโค้ดมากขึ้น

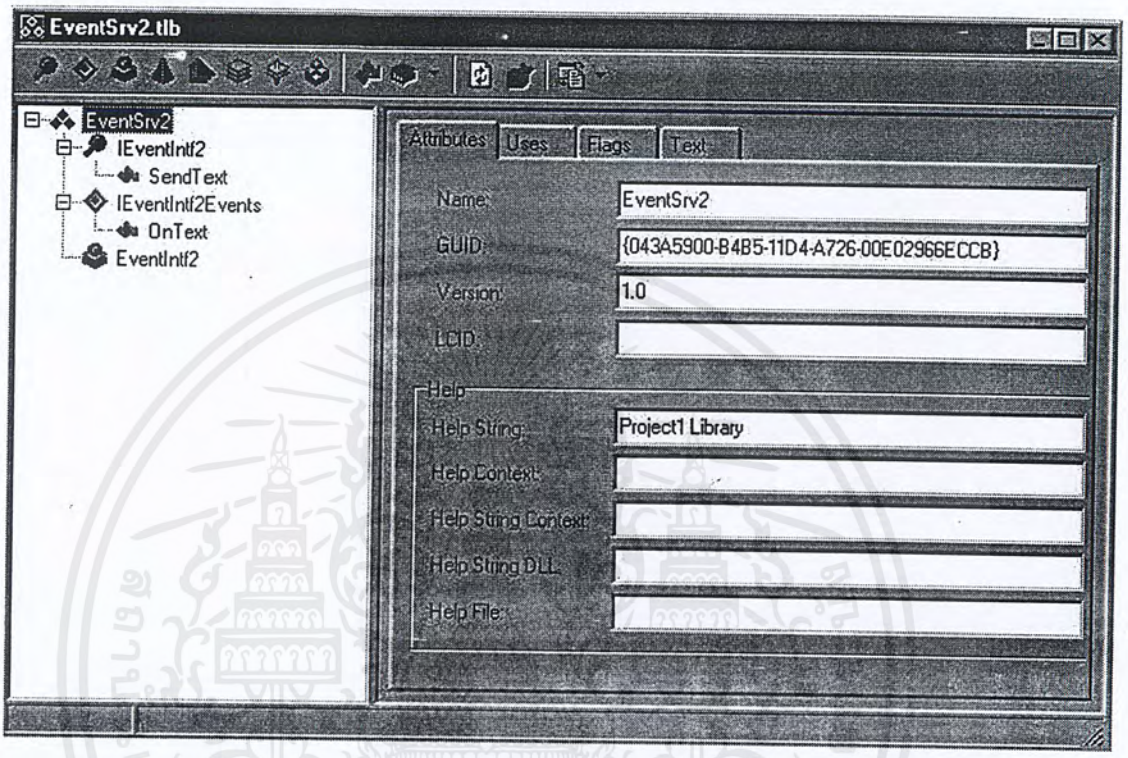
7.1 การสร้างออโตเมชันเซิร์ฟเวอร์ในการใช้งานอีเวนต์

เริ่มจากการสร้างแอปพลิเคชั่นง่ายๆ มาหนึ่งแอปพลิเคชั่นแล้วมำการเพิ่มออโตเมชั่นออบเจ็กต์เข้าไปตั้งชื่อแล้วเช็คที่เช็คบ๊อก Generate Event support code ด้วย ดังรูปที่ 7-1.



รูปที่ 7-1 แสดงหน้าจอในเพิ่มคุณสมบัติอีเวนต์เข้ากับคอมโพเนนท์ใหม่ใน Borland Delphi 5.0

โดย Borland Delphi 5.0 จะสร้างอินเตอร์เฟสให้สองอินเตอร์เฟสสำหรับ COM ออบเจ็กหนึ่ง และสำหรับอีเวนต์อีกหนึ่งในไทพ์ไลบรารีแล้วทำการเพิ่มเมธอดไปยัง IEventIntf2 ชื่อ SendText ให้มีพารามิเตอร์ชื่อ Text เป็นชนิด WideString ต่อไปทำการเพิ่มอีเวนต์ใน IEventIntf2Events ชื่อ OnText มีพารามิเตอร์ชื่อ Text ชนิด WideString แล้วกดปุ่ม Refresh จะเห็นไทป์ไลบรารี ดังรูปที่ 7-2



รูปที่ 7-2 แสดงการกำหนดเมธอดของ อินเตอร์เฟส IeventIntf2 และ IEventIntf2Events

ทำการเขียนโค้ดในเมธอด SendText ดังนี้

```
Procedure TeventIntf2.SendText(const Text : WideString)
Begin
        FEvents.OnText(Text);
End;
```

ทำการบันทึกเป็นการเสร็จสิ้นการสร้างเซิร์ฟเวอร์

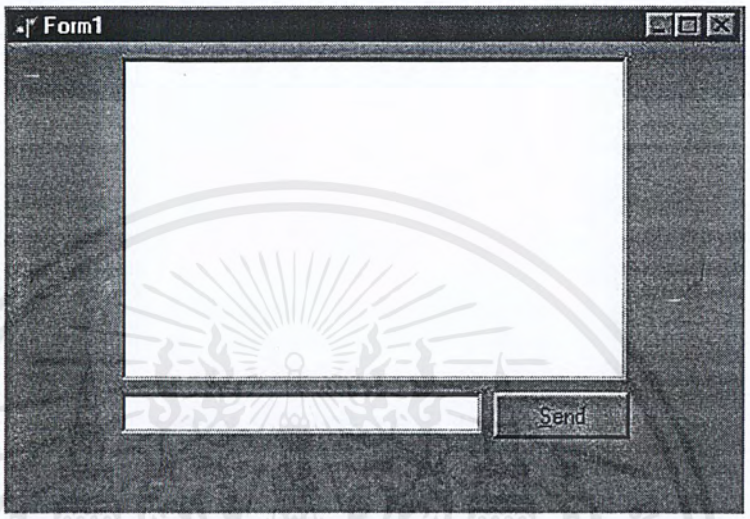
7.2 การสร้างไคลเอนต์ในการใช้งานอีเวนต์

การสร้างไคลเอนต์ทำได้โดยสร้างแอปพลิเคชั่นใหม่ขึ้นมาแล้วทำการอิมพอร์ตไทพ์ไลบรารีเข้ามา จะได้คอมโพเนนต์ชื่อ TEventIntf2 ออกมาบนคอมโพเนนต์พาเลตทำการเรียกใช้ได้เหมือนคอมโพเนนต์ทั่วๆ ไป

ตัว TEventIntf2 ที่เราสร้างขึ้นจะมีคุณสมบัติที่น่าสนใจอยู่ 3 ตัว คือ AutoConnect เป็นการกำหนดให้แอปพลิเคชั่นทำการติดต่อกับเซิร์ฟเวอร์โดยอัตโนมัติเมื่อถูกเรียกใช้งาน ตัวที่สองคือ

ConnectKind ใช้บอกว่าจะติดต่อกับเซิร์เวอร์อย่างไร เช่น ติดต่อกับรีโมตเซิร์ฟเวอร์, ติดต่อกับอินสแตนซ์ที่มีอยู่แล้ว หรือทำการสร้างอินสแตนซ์ ไม่ เป็นต้นส่วนตัวที่สามคือ RemoteMachineName ใช้ในการติดต่อกับรีโมตเซิร์ฟเวอร์

เราจะสร้างแอปพลิเคชั่นของไคลเอนต์ขึ้นมาง่ายๆ โดยเรียกใช้ตัว TEventIntf2 วางบนฟอร์มแล้ววางคอมโพเนนต์ต่างๆให้เป็น แอปพลิเคชั่นอย่างง่ายๆ ดังรูปที่ 7-3



รูปที่ 7-3 แสดงโปรแกรมทดสอบการเป็นไคล์เอนต์ ของ คอมโพเนนท์TEventIntf2

ภายในคอมโพเนนต์ TEventIntf2 จะมีอีเวนต์ให้เราใช้ได้คือ อีเวนต์ OnText ที่เราสร้างขึ้นทำการใช้ได้เหมือนกับการใช้อีเวนต์ของคอมโพเนนต์ทั่วไป โดยดับเบิลคลิกที่ชื่ออีเวนต์ OnText แล้วเขียนโค้ดดังนี้

```
Procedure TForm1.ButtonClick(Sender : TObject)
Begin
        EventIntf21.SenderText(Edit1.Text)
End;

Procedure TForm1.EventIntf2Event1Text(Sender : TObject)
Begin
        Memo1.Lines.Add(Text);
End;
```

การทำงานของไคลเอนต์เป็นการทำงานแบบง่ายๆ โดยเรียกเมธอด SendText ของเซิร์ฟเวอร์แล้วเซิร์ฟเวอร์จะทำการ Callback กลับมาที่ไคลเอนต์ทำให้เกิดการเขียนตัวอักษรขึ้นบนเมโม

7.3 การสร้างเซิร์ฟเวอร์แบบหลายไคลเอนต์

ตัวอย่างข้างต้นที่แสดงไปเป็นเซิร์ฟเวอร์ที่จัดการได้กับไคลเอนต์ได้ตัวเดียว ต่อไปเราจะดัดแปลงเซิร์ฟเวอร์ให้สามารถรองรับไคลเอนต์ให้เข้ามาใช้งานออบเจ็กต์ที่เซิร์ฟเวอร์ได้ทีละหลายๆ ตัวเป็นตัวอย่างการทำ Chat เซิร์ฟเวอร์อย่างง่ายๆ

ขั้นตอนการแปลงทำดังนี้ แก้ไขเมธอด Initialize จาก ckSingle เป็น ckMulti และทำการเรียกเมธอด RegiaterActiveXObject ดังนี้

```
Procedure TEventMultSrv.Initialize;
Begin
        Inherited Initialize;
        FconnectionPoints := TconnectionPoints.Create(Self);
        If AutoFactory.EventTypeInfo <> nil then
                FConnectionPoint := FconnectionPoints.CreateConnectionPoint(
                        AutoFactory.EventIID, ckMulti, EventConnect )
        Else FConnectionPoint := nil

        RegisterActiveObject ( Self as IUnknown, CLASS_EventMultSrv,
                ACTIVEOBJECT_WEAK, FObjectID );
End;
```

เนื่องจากเซิร์ฟเวอร์ต้องจัดการกับไคลเอนต์หลายตัวจึงต้องรู้ว่ามีเส้นทางการติดต่อมายังเซิร์ฟเวอร์อย่างไร โดยการใช้เมธอด GetEnumerator และต้องมีการทำลายออบเจ็กต์เมื่อเลิกใช้ด้วยตัวทำลาย Destroy การเขียนโค้ดภายในเป็นดังนี้

```
Destructor TeventMultSrv.Destroy;
Begin
        RevokeActiveObject ( FObjectID, nil) ;
        Inherited Destroy;
End;

Function TEventMultSrv.GetEnumerator : IEnumConnections;
Var
        Containner : IConnectionPointContainner;
        ConnectionPoint : IConnectionPoint;
Begin
        OleCheck(QueryInterface(IConnectionPointContainner, Containner));
        OleCheck(Containner.FindConnectionPoint(AutoFactory.EventIID, ConnectionPoint));
        ConnectionPoint.EnumConnections(Result);
End;
```

มาถึงเมธอด SendText ก็ต้องแก้ไขให้รองรับกับการ กระจายข้อความให้กับทุกๆ ไคลเอนต์ที่ต่อต่อกับออบเจ็กต์ของเซิร์ฟเวอร์ ดังนี้

```
Function TEventMultSrv.SendText ( const Text: WideString);
var
        Enum : IEnumConnections;
        ConnectData : TConnectData
```

```
        Fetched : Cardinal;
Begin
        Enum := GetEnumerator;
        If Enum <> nil then
        Begin
                While Enum.Next(1, ConnectData, @fetched) = s_ok do
                        If ConnectData.pUnk <> nil then
                                (ConnectData.pUnk as IeventMultSrvEvents).OnText(Text);
        End;
End;
```

เป็นการเสร็จสิ้นการสร้าง Chat เซิร์ฟเวอร์อย่างง่ายๆ ส่วนไคลเอนต์ไม่ต้องแก้ไขสามารถนำมาใช้กับเซิร์ฟเวอร์นี้ได้เลย

บทที่ 8
ไมโครซอฟต์รานแซคชันเซิร์ฟเวอร์

การพัฒนาระบบงานโดยใช้สถาปัตยกรรมเอ็นเทียร์ส(N-Tiers Architecture) เราจะให้คอมโพเนนต์แอปพลิเคชั่นต่าง ๆ ทำงานอยู่บนมิดเดิลเทียร์เซิร์ฟเวอร์ (Middle-Tier Server) ซึ่งเป็นอิสระจากส่วนงานพรีเซ็นเตชั่นและงานด้านฐานข้อมูล คอมโพเนนต์ที่เกี่ยวกับงานพรีเซ็นเตชั่นจะทำงานอยู่ที่เดสก์ท็อปไคลเอนต์หรือเว็บบราวเซอร์ ส่วนคอมโพเนนที่เกี่ยวกับข้อมูลก็จะทำงานอยู่บนดาตาเบสเซิร์ฟเวอร์ ข้อดีในการแยกแอปพลิเคชั่นคอมโพเนนต์เป็นอสระจากส่วนพรีเซ็นเตชั่นและงานฐานข้อมูลคือ

- สามารถนำแอปพลิเคชั่นคอมโพเนนต์กกลับมาใช้ใหม่ (Reusable)ได้
- แอปพลิเคชั่นคอมโพเนนต์สามารถทำงานบนหลาย ๆ เครื่องพร้อมกันได้
- แอปพลิเคชั่นคอมโพเนนต์สามารถใช้การเชื่อมต่อดาตาเบส (Database Connection) ร่วมกันได้

8.1 การพัฒนาระบบงานเอ็นเทียร์โดยใช้ไมโครซอร์ฟทรานแซคชันเซิร์ฟเวอร์

เอ็มทีเอสเป็นแอปพลิเคชั่นที่ใช้ในการพัฒนาระบบงานเชิงคอมโพเนนต์ มีรันไทม์เอนไวรอนเมนต์ (Run-time environment) ใช้ควบคุมแอปพลิเคชั่นในการเรียกใช้คอมโพเนนต์ทั้งที่มีลักษณะเป็นทรานแซคชันหรือไม่ใช่ทรานแซคชันก็ตาม เอ็มทีเอสทำให้การพัฒนาและจัดการระบบงานเอ็นเทียร์ที่สร้างบนพื้นฐานของคอมโพเนนต์ ออบเจ็กต์โมเดลมีความสะดวกขึ้น แอปพลิเคชันคอมโพเนนต์จะทำงานภายใต้การควบคุมของ MTS คอมโพเนนต์เหล่านี้จะถูกเรียกใช้จากไคลเอนต์หลายประเภท เช่น เว็บบราวเซอร์ เอเอสพีสคริปท์ที่ทำงานอยู่ภายในไอไอเอส หรือแม้กระทั่งแอปพลิเคชั่นทั่วไปที่เขียนโดยวิชวลเบสิก เป็นต้น

8.2 ข้อดีในการใช้เอ็มทีเอส

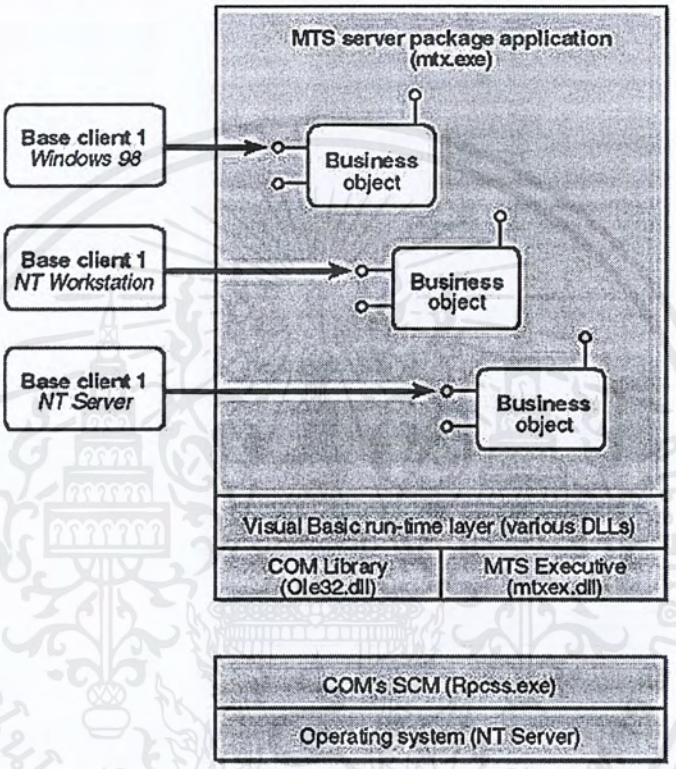
เอ็มทีเอสทำให้การพัฒนาระบบงานมีประสิทธิภาพ ฟังก์ชันในงานธุรกิจต่าง ๆ สามารถรองรับการใช้งานจากผู้ใช้ที่มีจำนวนมากได้ ช่วยลดความซ้ำซ้อนและค่าใช้จ่ายในการพัฒนาแอปพลิเคชั่น โดยเราสามารถพัฒนาคอมโพเนนต์ประเภทอินโพรเซสทำงานบนมิดเดิลเทียร์เซิร์ฟเวอร์ภายใต้การควบคุมของเอ็มทีเอส เมื่อไคลเอนต์แอปพลิเคชั่นต้องการเรียกใช้คอมโพเนนต์ก็จะติดต่อกับเอ็มทีเอสโดยใช้ดีซีโอ-เอ็มข้อดีอื่น ๆ ในการใช้ MTS คือ

- สามารถติดต่อกับฐานข้อมูลและทรัพยากรอื่น ๆ เช่น เมนเฟรมแอปพลิเคชั่น ในลักษณะที่เป็นทรานแซคชันได้
- จัดการคอมโพเนนต์ได้ง่าย สามารถใช้การแดรกแอนด์ดรอป (drag-and-drop) คอมโพเนนต์ผ่านทางเอ็มทีเอสเอ็กซ์พลอเรอร์ และมียูทิลิตี้ต่าง ๆ ที่ช่วยในการรีจิสเตอร์คอมโพเนนต์บนไคลเอนต์ได้โดยอัตโนมัติ

8.3 เอ็มทีเอสคอมโพเนนต์

เอ็มทีเอสมีรันไทม์เอนไวรอนเมนต์ที่ใช้รันออบเจ็กต์ รูปที่ 8-1 แสดงความสัมพันธ์ระหว่างไคลเอนต์แอปพลิเคชันและออบเจ็กต์ที่รันอยู่ในรันไทม์เอนไวรอนเมนต์

ถึงแม้ว่าคอมโพเนนต์ประเภทอินโพรเซสสามารถใช้เป็น เอ็มทีเอสคอมโพเนนต์ได้ แต่จะไม่สามารถใช้เซอร์วิสต่าง ๆ ของเอ็มทีเอส ได้อย่างมีประสิทธิภาพ ดังนั้นเราจึงควรปรับปรุงคอมโพเนนต์ให้ใช้คุณสมบัติของเอ็มทีเอสอย่างเต็มที่ ซึ่งเอ็มทีเอสต้องการคอมโพเนนต์ที่มีลักษณะดังนี้



รูปที่ 8-1 ความสัมพันธ์ระหว่างไคลเอนต์แอปพลิเคชันและออบเจ็กต์ที่รันอยู่ในรันไทม์เอนไวรอนเมนต์

- คอมโพเนนต์จะต้องมีออบเจ็กต์คลาสมาตรฐานที่อิมพลีเมนต์มาจากอินเตอร์เฟส IclassFactory
- เมธอด IClassFactory::CreateInstance ของออบเจ็กต์คลาสจะต้องรีเทิร์นอินสแตนท์ของออบเจ็กต์ ใหม่ทุกครั้งเมื่อเมธอดนี้ถูกเรียกใช้
- คอมโพเนนต์จะต้องเปิดเผยออบเจ็กต์คลาสโดยใช้ฟังก์ชัน DllGetClassObject
- คอมโพเนนต์ที่รันในเอ็มทีเอส ต้องไม่อะกรีเกตกับคอมโพเนนต์ที่ไม่ได้รันในเอ็มทีเอส
- คอมโพเนนต์ต้องใช้ฟังก์ชัน DllRegisterServer ในการรีจิสเตอร์ ProgID, ClsID, อินเตอร์เฟส และไทป์ไลบรารี ไว้ที่ HKEY_CLASSES_ROOT ของวินโดว์รีจิสทรี
- อินเตอร์เฟสของคอทโพเนนท์ และ ซีโอเอ็มคลาสทั้งหมดจะต้องอธิบายอยู่ในไทป์ไลบรารี

8.4 บริการต่าง ๆ ของ MTS

คอมโพเนนต์ทรานแซคชัน (Component Transaction)

คอมโพเนนต์ทรานแซคชันมีลักษณะคล้ายกับดาตาเบสทรานแซคชัน คอมโพเนนต์ทรานแซคชันประกอบด้วยวิธีการในการรันคอมโพเนนต์ตั้งแต่หนึ่งคอมโพเนนต์ขึ้นไป เอ็มทีเอสจะควบคุมการทำงานคอมโพเนนต์ในการติดต่อกับทรัพยากรที่ทำงานแบบทรานแซคชัน เช่น ติดต่อกับฐานข้อมูล หากคอมโพเนนต์ยกเลิกการทำทรานแซคชัน เอ็มทีเอสจะโรลแบค (Rollback) สิ่งที่คอมโพเนนต์ได้ปฏิบัติไป

ความปลอดภัย

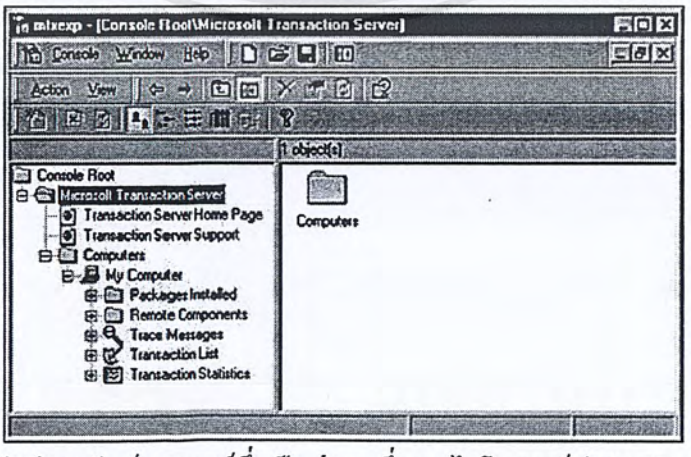
เอ็มทีเอสใช้ระบบการรักษาความปลอดภัยของวินโดว์เอ็นทีในการตรวจสอบคุณสมบัติผู้ใช้ระบบ ส่วนเอ็มทีเอสเองมีโมเดลในการรักษาความปลอดภัย 2 แบบคือ Declarative Security และ Programmatic Security

การจัดการ (Administration)

เอ็มทีเอสมีเอ็มทีเอสเอ็กซ์พลอเรอร์ (MTS Explorer) เป็นยูสเซอร์อินเตอร์เฟสในการจัดการควบคุมเพจเกจ ปรับเปลี่ยนคุณสมบัติคอมโพเนนต์ในด้านความปลอดภัยและลักษณะการทำทรานแซคชัน ติดตามการทำงานของเซิร์ฟเวอร์ นอกจากนั้นเอ็มทีเอสเอ็กซ์พลอเรอร์ยังมีคุณสมบัติอื่น ๆ ดังนี้

- ใช้งานง่าย สามารถใช้การแดรกแอนด์ดรอปคอมโพเนนต์จากภายนอกนำเข้ามาติดตั้งในเอ็นไวรอนเมนต์ของเอ็มทีเอสได้ทันที
- สามารถล็อกแพกเกจเพื่อป้องกันการเปลี่ยนแปลงแพกเกจขณะใช้งาน สามารถดูสถานะขณะทำงาน เปลี่ยนชื่อแพกเกจ และเพิ่มเงื่อนไขด้านความปลอดภัย
- มีความสามารถในการจัดการไอไอเอสแอปพลิเคชั่น (IIS Application) ที่ทำงานภายใต้เอ็มทีเอส และจัดการแพกเกจที่ถูกสร้างโดยเอสเอ็นเอเซิร์เวอร์ 4.0 (SNA Server 4.0)

เอ็มทีเอสเอ็กซ์พลอเรอร์เป็นส่วนหนึ่งของไมโครซอร์ฟเมเนจเมนต์คอนโซล (Microsoft Management Console, MMC) มีลักษณะคล้ายกับวินโดว์สเอ็กซ์พลอเรอร์ ดังรูปที่ 8-2



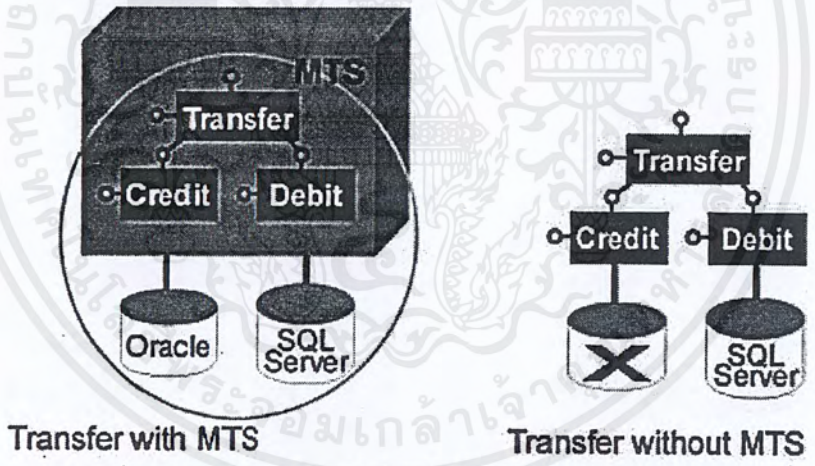
รูปที่ 8-2 เอ็มทีเอสเอ็กซ์พลอเรอร์ซึ่งเป็นส่วนหนึ่งของไมโครซอร์ฟเมเนจเมนต์คอนโซล

นอกจากนี้ยังมีปริการอื่น ๆ เช่น การทำเธรดพูลลิ่งโดยอัตโนมัติ (Automatic Thread Pooling) ออบเจ็กต์โบรกเกอร์ (Object Broker) รีซอร์สพูลลิ่ง (Resource Pooling) โพรเซสไอโซเลชัน (Process Isolation) สนับสนุนคอมโพเนนต์แพจเกจ จัสอินไทม์แอคติเวชัน (Just-in-Time Activation) เป็นต้น

8.5 ตัวอย่างของเอ็มทีเอสแอปพลิเคชัน

แอปพลิเคชั่นระบบธนาคารประกอบด้วย 3 คอมโพเนนต์ได้แก่ การฝาก การถอน และการโอนเงิน ซึ่งคอมโพเนนต์แต่ละอันถูกพัฒนาโดยนักพัฒนาต่างกัน และการโอนเงินทำงานโดยเรียกใช้คอมโพเนนต์การถอนและการฝาก

สมมุติว่ามีการโอนเงิน เริ่มต้นจะต้องถอนเงินจากบัญชีก่อน แล้วจึงทำการฝาก หากการฝากเกิดบัญหาไม่สามารถทำงานได้ หากไม่ได้ทำงานแบบทรานแซคชันผู้พัฒนาคอมโพเนนต์การโอนเงินจะต้องเขียนโปรแกรมจัดการปัญหาให้บัญชีมียอดเงินเท่าเดิม ในกรณีตัวอย่างนี้การจัดการแก้ปัญหายังสามารถจัดการได้ง่ายเนื่องจากมีคอมโพเนนต์ในระบบไม่มากนัก หากระบบงานประกอบด้วยคอมโพเนนต์จำนวนมากที่ทำงานสัมพันธ์กัน หากการทำงานของคอมโพเนนต์ใดมีปัญหา การจัดการแก้ไขจะทำได้ยากและสิ้นเปลืองค่าใช้จ่าย รูปที่ 8-3 แสดงให้เห็นถึงการโอนเงินที่ทำงานในลักษณะเป็นทรานแซคชัน (โดยใช้เอ็มทีเอส) และไม่ได้เป็นทรานแซคชัน (ไม่ใช้เอ็มทีเอส)



รูปที่ 8-3 กระบวนการโอนเงินโดยใช้เอ็มทีเอสและไม่ใช้เอ็มทีเอส

เอ็มทีเอสจะควบคุมการทำทรานแซคชันให้โดยผู้พัฒนาโปรแกรมไม่จำเป็นต้องเขียน Begin Transaction และ End Transaction ในโค้ดของโปรแกรมเลย ผู้พัฒนาโปรแกรมสามารถใช้เครื่องมือและภาษาต่าง ๆ สร้างคอมโพเนนต์และให้ทำงานเป็นแบบทรานแซคชันได้โดยใช้เอ็มทีเอสควบคุม เมื่อคอมโพเนนต์เริ่มทำงาน เอ็มทีเอสก็จะเริ่มต้นทรานแซคชันให้หากคอมโพเนนต์ต้องการ

เมื่อคอมโพเนนต์มีการติดต่อทรัพยากรต่าง ๆ เช่น ฐานข้อมูลหรือเมนเฟรมแอปพลิเคชั่น เอ็มทีเอสจะทำงานเป็นแบบทรานแซคชันโดยอัตโนมัติ หากคอมโพเนนต์ที่ทำงานแบบทรานแซคชันมีการเรียกใช้คอมโพเนนต์อื่น ๆ คอมโพเนนต์ที่ถูกเรียกใช้ก็จะอยู่ในทรานแซคชันเช่นกัน และ เอ็มทีเอสทำการควบคุมทรานแซคชันแบบทูเฟส-คอมมิท (Two-phase commit)

สรุปแล้วเอ็มทีเอสมีความสามารถหลักที่สำคัญ 3 ประการคือ

1. สนับสนุนการทำทรานแซคชัน
2. สนับสนุนการใช้ข้อมูลร่วมกัน
3. สนับสนุนด้านความปลอดภัย

บทที่ 9
วินโดวส์ดีเอ็นเอ

9.1 ความหมาย

วินโดวส์ดีเอ็นเอ (Windows DNA) มาจากคำว่า Windows Distributed interNet Architecture เป็นแแนวทางในการสร้างระบบงานของไมโครซอฟต์ หรือเป็นข้อแนะนำในการสร้างระบบงานซึ่งเชื่อมต่อกับอินเทอร์เน็ต วินโดวส์ดีเอ็นเอไม่ใช่ข้อกำหนดดังนั้นในการพัฒนาระบบงาน ใด ๆ จึงอาจจะใช้เพียงบางส่วนของวินโดวส์ดีเอ็นเอ เพื่อพัฒนาระบบงานร่วมกับองค์ประกอบอื่น ด้วยก็ได้

9.2 องค์ประกอบของวินโดวส์ดีเอ็นเอ

องค์ประกอบของวินโดวส์ดีเอ็นเอประกอบด้วย

- ส่วนซอฟต์แวร์คอมโพเนนต์ ในส่วนนี้ทางไมโครซอฟต์นั้นเสนอให้ใช้คอมพลัสคอมโพเนนต์ เพื่อให้สามารถใช้บริการต่าง ๆ ของ สภาพแวดล้อมคอมพลัสรันไทม์เช่นมี เธรดโมเดลเป็นลักษณะ Both ทำให้สามารถเพิ่มประสิทธิภาพในการประมวลผล (Scalability) เป็นต้น

- มีแอปพลิเคชั่นเซิร์ฟเวอร์ หรือมีคอนเทนเนอร์ (Container) ของคอมพลัสคอมโพเนนต์เป็นสภาพแวดล้อม คอมพลัสรันไทม์มีมาพร้อมกับระบบปฏิบัติการ Windows 2000 ฎ Server, Windows 2000 ฎ Advance Server และ Windows 2000 ฎ Data Center เนื่องจากตัวสภาพแวดล้อมคอมพลัสนั้นสนับสนุนตัวคอมโพเนนต์ซึ่งมีลักษณะเหมาะสมกับระบบสภาพแวดล้อมนี้ นอกจากนั้นยังมีบริการเพิ่มเติม เช่น การจัดการทรานแซคชัน หรือระบบความปลอดภัยแบบแบ่งลำดับชั้น (Fine-Gain Security) เป็นต้น

- มีเว็บเซิร์ฟเวอร์ เป็น Internet Information Server (IIS) เวอร์ชัน 5.0 ซึ่งมาพร้อมกับผลิตภัณฑ์ Microsoft Windows 2000 Server, Microsoft Windows 2000 ฎ Advance Server และ Microsoft Windows 2000 ฎ Data Center

- เครื่องมือในการพัฒนาคอมคอมโพเนนต์นั้นทางไมโครซอฟต์ได้เสนอผลิตภัณฑ์ Microsoft Visual Studio ฎ ซึ่งประกอบด้วยโปรแกรม Visual Basic, Visual C++ และ Visual InterDev อยู่นอกจากนั้นยังมีโปรแกรมและเครื่องมือต่าง ๆ ที่จะช่วยให้การพัฒนาคอมคอมโพเนนต์สะดวกยิ่งขึ้น และไม่เกิดปัญหาจากความเข้ากันได้ของคอมโพเนนต์และส่วนประกอบอื่น ๆ ของระบบ

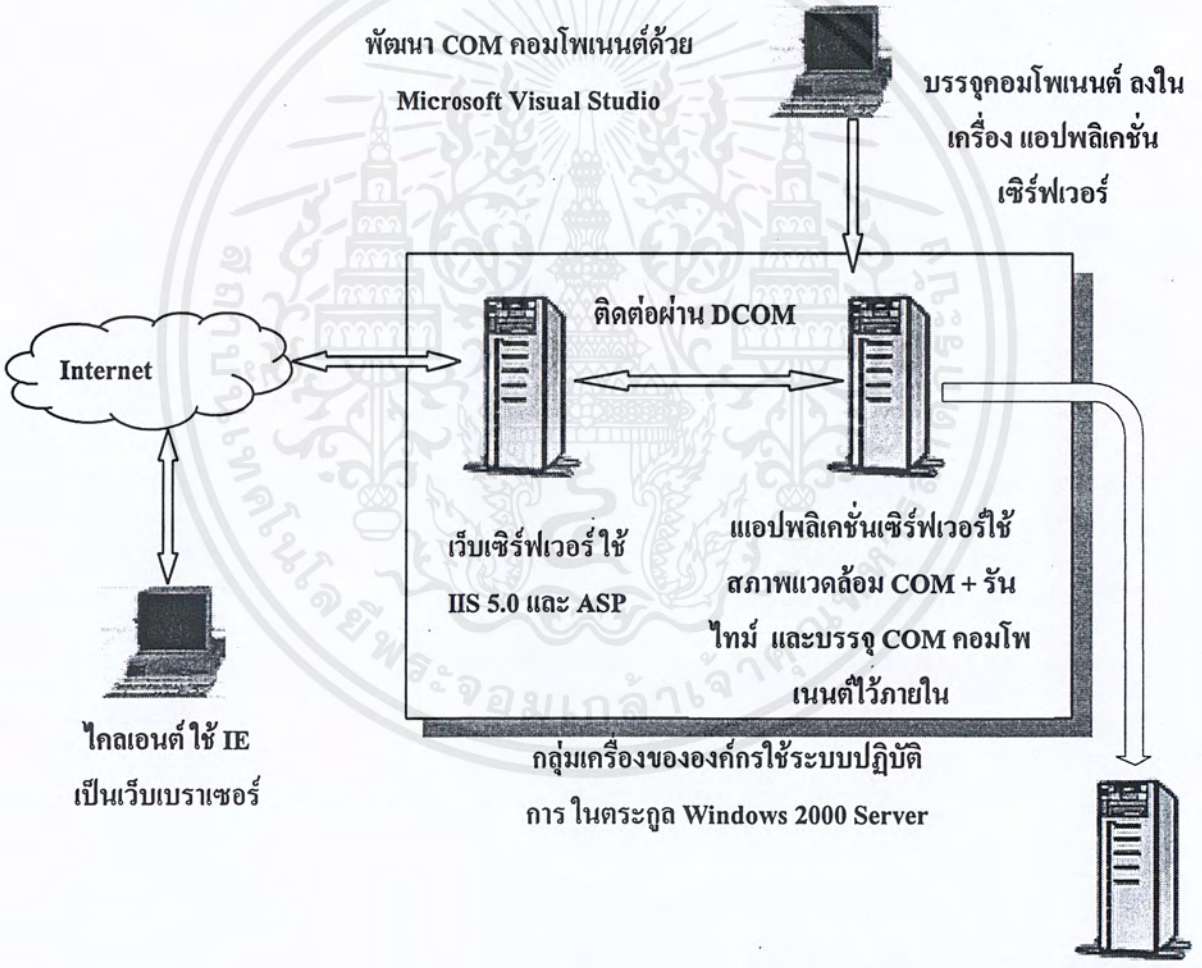
- สำหรับภาษาสคริปต์ในฝั่งเซิร์ฟเวอร์ ไมโครซอฟต์เสนอ Active Server Page (ASP) ซึ่งใช้งานได้ไม่ยากนักเนื่องจากมีความคล้ายภาษา VBScript ซึ่งก็มีลักษณะและไวยกรณ์ของภาษาไม่ต่างจาก Visual Basic ซึ่งอยู่ในชุดผลิตภัณฑ์ Microsoft Visual Studio ฎ เท่าใดนัก เพื่อใช้ในการเรียกตัวคอมคอมโพเนนต์ซึ่งอยู่ภายในตัว แอปพลิเคชั่นเซิร์ฟเวอร์ นอกจากนี้ยังสามารถพัฒนาได้ด้วยโปรแกรม Microsoft Visual InterDev ซึ่งอยู่ในชุดผลิตภัณฑ์ Microsoft Visual Studio อีกด้วย

- ในส่วนการจัดการฐานข้อมูลนั้นก็จะเป็นตัวผลิตภัณฑ์ Microsoft SQL Server 2000 ซึ่งเข้ากันได้กับคอมโพเนนต์ และยังมีความสามารถใหม่ ๆ ซึ่งได้เพิ่มมากจาก เวอร์ชันก่อนหน้า คือตัวผลิตภัณฑ์ Microsoft SQL Server 7.0 อยู่พอสมควร และใช้งานได้ง่ายกว่าผลิตภัณฑ์ ระบบจัดการบริหาร ฐานข้อมูลตัวอื่นอยู่พอสมควร

- ในการเชื่อมต่อและส่งผ่านข้อมูล นั้นก็จะใช้ดีคอม เป็นตัวกลางในการติดต่อกับคอมโพเนนต์ และ ส่วน ต่าง ๆ ภายในระบบ

- ตัวเว็บบราวเซอร์ ที่อยู่ใน DNA นั้นก็จะเป็น Internet Explorer เนื่องจากมีความเข้ากันได้กับตัวคอมโพเนนต์ นอกจากนั้นตัว Internet Explorer นั้นก็มีเวอร์ชันที่ทำงานอยู่ในหลาย ๆ แพลตฟอร์ม และยังเป็นผลิตภัณฑ์ ที่แถมมากับระบบปฏิบัติการ Microsoft Windows เวอร์ชันใหม่ ๆ อยู่แล้ว

จากที่ได้กล่าวมาเบื้องต้น ภาพรวมของระบบ Windows DNA จะมีลักษณะดังรูปที่ 9-1



รูปที่ 9-1 สถาปัตยกรรมของวินโดวส์ดีเอ็นเอ

บทที่ 10
สภาพแวดล้อม COM+ รันไทม์ และบริการของ COM+

เป็นกลุ่มของบริการจากระบบ ที่มีบริการต่าง ๆ สำหรับ COM ออบเจ็กต์ ดังนี้ fine-grained security, distributed transactions, thread synchronization, load balancing, asynchronous store และ forward method invocation นอกจากนั้นแล้วยังมีการพัฒนาการขยายขนาด (scalability) โดยใช้ Just In Time (JIT) Activation และ การ pooling ของ ออบเจ็กต์ และ database connection

10.1 สถาปัตยกรรม ของ COM+

10.1.1 Attributed-Based Programming

หลักการของวิธีนี้คือ การที่สามารถทำการปรับแต่งคุณสมบัติของ COM ออบเจ็กต์ได้จากการปรับแต่งค่าของแอททริบิวต์ โดยที่รายละเอียดนอกจากนั้น COM+ รันไทม์ จะเป็นส่วนที่จัดการเพิ่มเติมเอง สำหรับในวินโดว์ 2000 นั้นค่าแอททริบิวต์สำหรับ COM ออบเจ็กต์ต่างนั้นไม่ได้เก็บอยู่ในรีจิสต์ทรีของระบบ แต่จะถูกเก็บไว้ใน configuration database ซึ่งเรียกว่า COM+ catalog นอกจากนั้นแล้ว Windows 2000 นั้นยังได้เตรียม Catalog Manager ไว้เพื่อความสะดวกในการใช้งานเรียกว่า Component Service Explorer

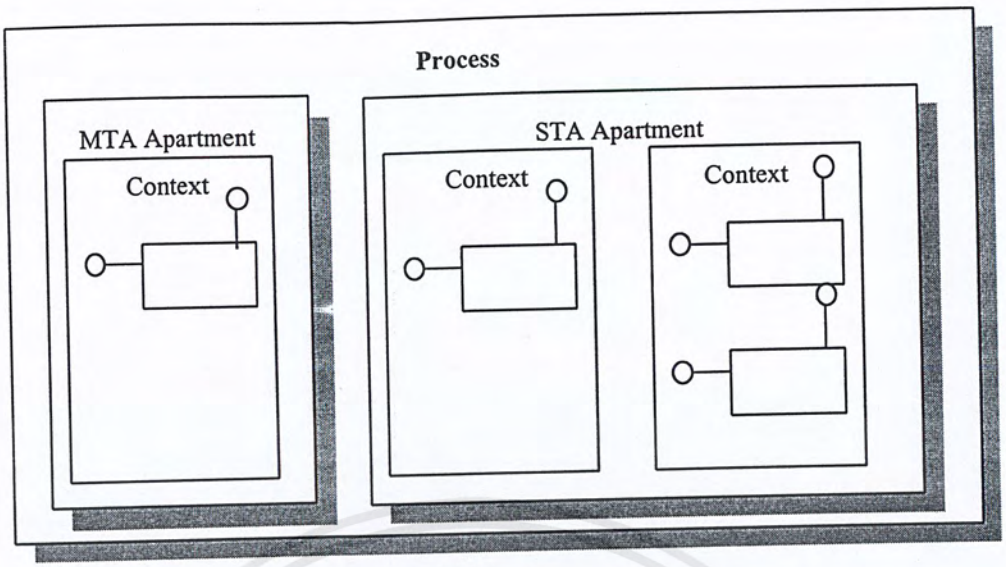
การเรียกใช้งาน Component Service Explorer สามารถเรียกใช้ได้โดย

1. กดปุ่ม Start
2. เลือก Programs \ Administrative Tools \ Component Services

10.1.2 คอนเทค (Context)

COM+ คอนเทค คือ สภาพแวดล้อมรันไทม์ ซึ่ง COM+ ออบเจ็กต์ ที่มีความเข้ากันได้ใน โพรเซสที่เหมาะสมทำการประมวลผล (สำหรับความเข้ากันได้นั้นหมายถึง ออบเจ็กต์ที่ใช้ runtime requirement ร่วมกัน; หากมีการปรับแต่งค่า สำหรับ ออบเจ็กต์ตัวใดที่แตกต่างกัน ก็ต้องอยู่ต่าง คอนเทค กัน)

Context จะอยู่ภายใน อพาร์ตเมนท์ (Apartment) ซึ่งจะอยู่ภายใน โพรเซส อีกที เฉพาะ เธรด ที่อยู่ใน Apartment นั้นเท่านั้นที่จะสามารถเรียกใช้โดยผ่าน คอนเทค ได้โดยตรง (นอกจากจะเป็น thread-neutral apartment) ในแต่ละ อพาร์ตเมนท์ นั้นสามารถมีได้หลาย คอนเทค และ ภายในแต่ละ คอนเทค ก็มีได้หลายออบเจ็กต์ ดังรูปที่ 10-1



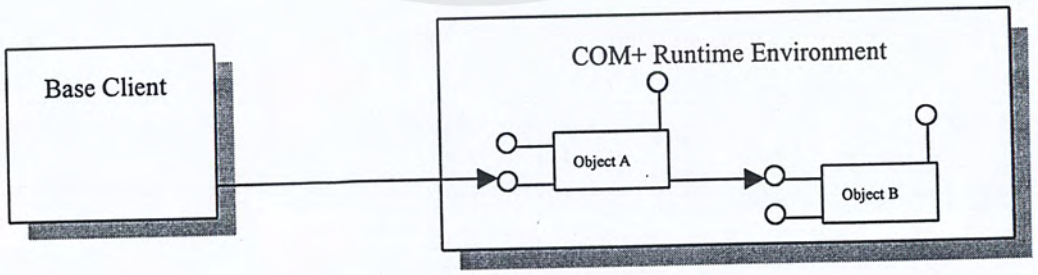
รูปที่ 10-1 แสดงโพรเซส อพาร์ตเมนท์ และ คอนเทค

10.1.3 COM+ ไคลเอนต์ กับ Context

COM+ ออบเจ็กต์สามารถมีไคลเอนต์ได้ 2 ประเภทคือ 1) COM+ ออบเจ็กต์ตัวอื่นแต่อยู่ในสภาพแวดล้อม COM+ runtime หรือ 2) ไคลเอนต์ซึ่งรันอยู่นอกสภาพแวดล้อม COM+ runtime ดังรูปที่ 10-2

รูปแบบแรกนั้นจะมีลักษณะเดียวกับ ออบเจ็กต์ A ในรูปที่ x ซึ่งเรียกใช้ออบเจ็กต์ B และอยู่ในสภาพแวดล้อม COM+ รันไทม์ ส่วนในรูปแบบหลังนั้นเรียกว่า Base Client ซึ่งเป็นส่วนของซอฟต์แวร์ที่เรียกใช้ COM+ ออบเจ็กต์แต่ตัวมันจะอยู่นอกสภาพแวดล้อม COM+ รันไทม์ โดยมากแล้วไคลเอนต์ประเภทนี้มักจะเป็นพวก User Interface แต่ก็ไม่เสมอไป

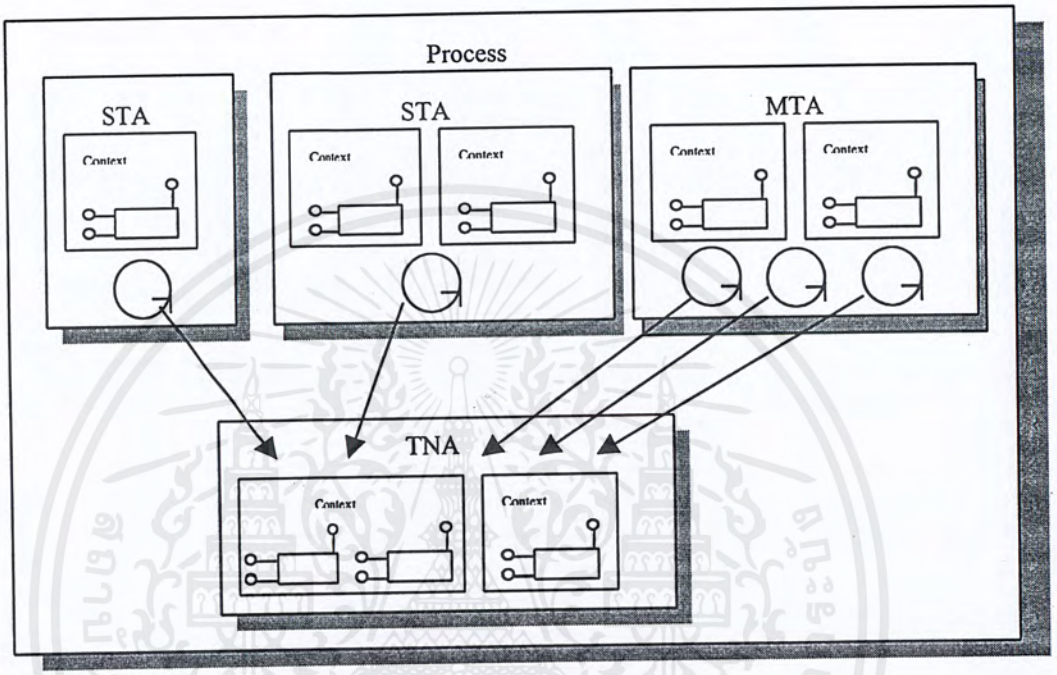
ส่วนใหญ่แล้วตัวไคลเอนต์กับตัว COM+ ออบเจ็กต์มักจะอยู่ต่าง Processหรือ อย่างน้อยก็จะอยู่ต่าง Context การติดต่อสื่อสารระหว่างส่วนต่างๆจึงจำเป็นต้องมีกระบวนการที่ใช้จัดการนี้ซึ่งเรียกว่า Interception (ซึ่งจะได้อธิบายต่อไป) อย่างไรก็ดีในบางกรณีเราจะออกแบบให้ตัวไคลเอนต์กับตัว COM+ ออบเจ็กต์อยู่ใน Context เดียวกันก็ได้



รูปที่ 10-2 แสดงCOM+ ไคลเอนต์ทั้งสองประเภท

10.1.4 คอนเทค และ อพาร์ตเมนท์

ในมุมมองของ COM+ อพาร์ตเมนท์ จะเป็นกลุ่มของ คอนเทค ซึ่งอยู่ใน โพรเซส ดังรูปที่ 10-3 Context ทำหน้าที่เป็นสภาพแวดล้อมที่ใช้ในการประมวลผลชั้นในสุดแทน อพาร์ตเมนท์ โดยที่ อพาร์ตเมนท์ จะทำหน้าที่พิจารณาว่า Thread ใดมีสิทธิที่จะเข้าไป ติดต่อกับออบเจ็กต์ได้โดยตรงใน คอนเทค ที่เหมาะสม



รูปที่ 10-3 แสดงอพาร์ตเมนท์ และ คอนเทค

คอนเทค สามารถควบคุม การทำเธรดซิงโครไนซ์สำหรับ COM+ ออบเจ็กต์ซึ่งการให้บริการนี้เป็นของ สภาพแวดล้อม COM+ รันไทม์

วิธีการทำ การทำเธรดซิงโครไนซ์ที่ COM+ ให้บริการโดย COM+ รันไทม์ ทำโดย เมื่อมี เธรดใด เธรด หนึ่ง อ้างถึงเมธอดที่อยู่ในออบเจ็กต์ ทุกๆ เธรด ที่เหลือจะถูกล็อคไว้จนกว่า เธรด แรกจะออกจากเมธอดและยกเลิกการล็อคซึ่งวิธีการนี้ก็ใช้ได้อย่างมี ประสิทธิภาพกับ Object Pooling ซึ่งกำหนดให้ออบเจ็กต์นั้นสามารถถูกเรียกใช้จาก Thread ใดๆ ก็ได้ อย่างไรก็ดี การทำเธรดซิงโครไนซ์ นั้นไม่สามารถใช้งานได้ในระบบงานที่เป็น Thread-affinity เนื่องจากระบบงานแบบนี้นั้นจำเพาะเจาะจงไปเลยว่าต้องการ เธรด ใน เธรด หนึ่งอย่างเดียว

สำหรับ STA นั้นไม่จำเป็นต้องมีการล็อคเนื่องจากมีแค่ Thread เดียวในตัวมันแต่อย่างไรก็ตาม STA นั้นไม่เหมาะกับ Object Pooling สรุปก็คือ คอนเทค จะใช้ในการควบคุมว่า เธรด ต่างๆ จะสามารถเรียกใช้ออบเจ็กต์ได้ เมื่อไร ส่วน Apartment นั้นจะใช้ในการควบคุมว่า เธรด ไหน ที่จะถูกอนุญาติให้เข้ามาเรียกใช้ออบเจ็กต์ นอกจากนั้นยังมีข้อแนะนำใหม่อีก 2 ข้อคือ

- COM+ ออบเจ็กต์ควรจะใช้ TNA และใช้ การทำเธรดซิงโครไนซ์ซึ่งให้บริการโดย COM+
- COM+ ออบเจ็กต์ที่เป็น UI หรือที่เป็น Thread-affinity ควรใช้ STA ต่อไป

เพื่อให้สามารถเข้ากันได้กับระบบเก่า ๆ COM อพาร์ตเมนท์ ก็ยังรวมถึง คอนเทค ที่เป็นดีฟอลต์ ซึ่งจะไม่มีบริการต่างของ COM+

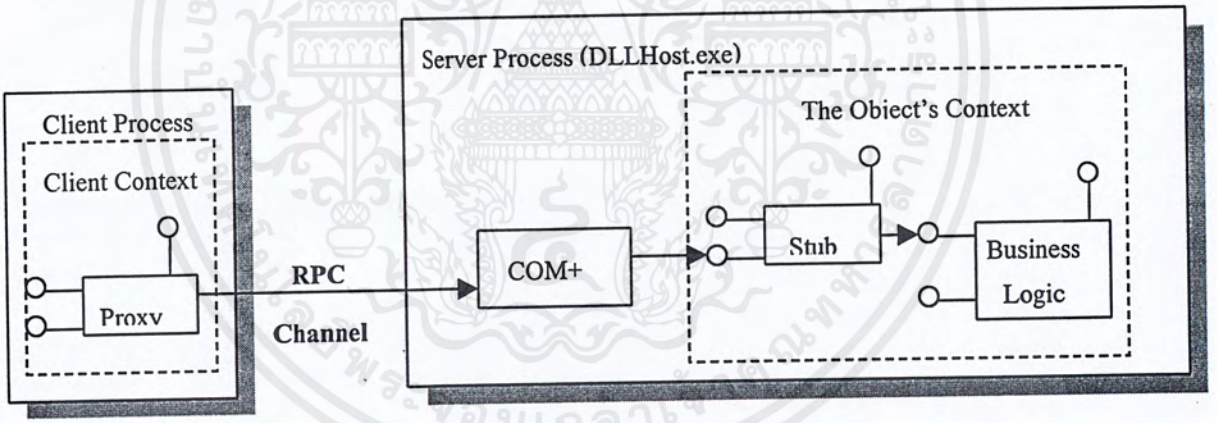
การทำเธรดชิงโครไนซ์ เป็นทางเลือกหนึ่ง ซึ่งสามารถเลือกได้ว่าจะใช้บริการนี้หรือไม่

10.1.5 Interception

ในขณะที่ ไคลเอนต์เรียกใช้ COM ออบเจ็กต์ ถ้าตัวไคลเอนต์นั้นอยู่ใน คอนเทค เดียวกับ COM ออบเจ็กต์ ก็จะสามารถเรียกใช้ได้โดยตรงเนื่องจากตัวไคลเอนต์นั้นอยู่ในสภาพแวดล้อมแบบเดียวกับที่ตัว COM ออบเจ็กต์ต้องการ แต่ในการใช้งานจริงนั้นส่วนใหญ่แล้วตัวไคลเอนต์ และ COM ออบเจ็กต์มักจะอยู่ต่าง คอนเทค กันดังนั้นจึงต้องมีตัวกลางในการจัดการ ซึ่งตัวกลางนี้เรียกว่า Interceptor ซึ่งมีลักษณะเป็น พร็อกซี่ ที่มีขนาดเล็ก

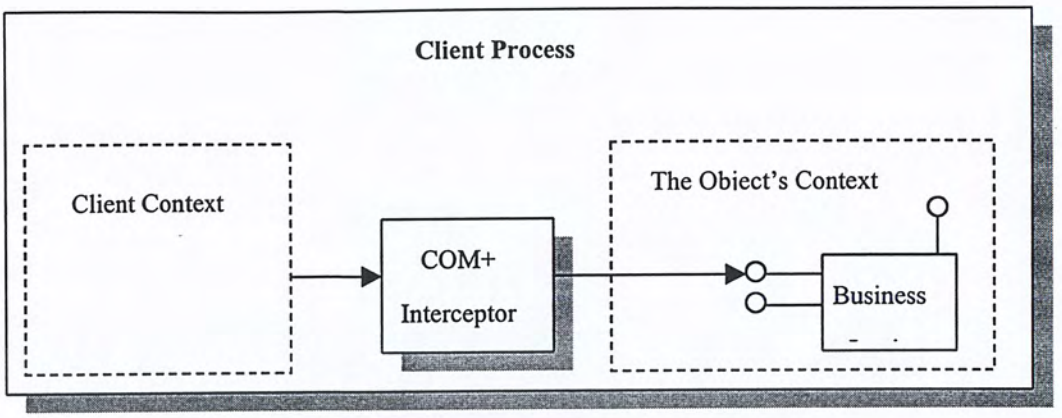
Interceptor มีหน้าที่ทำต่างๆ ซึ่งจำเป็นต่อการส่งคำสั่งจากไคลเอนต์ไปยังตัว COM ออบเจ็กต์ เช่นถ้าหากตัวออบเจ็กต์ ต้องการทรานแซคชันนั่นก็คือ ทรานแซคชัน ต้องถูกเริ่มต้น และจะบังคับให้มีการควบคุมการเข้าถึงแบบ fine-grained โดยการตรวจสอบว่าผู้ใช้นั้นมีสิทธิ์ที่จะใช้บริการนั้นๆ หรือไม่

หากตัวไคลเอนต์ และ COM ออบเจ็กต์นั้นอยู่คนละโพรเซสตัว COM+ Interceptor ก็จะอยู่ระหว่าง RPC Channel กับ Marshalling Stub ดังรูปที่ 10-4



รูปที่ 10-4 แสดงInterception ที่ใช้กับ COM+ ออบเจ็กต์ชนิด Out-of-Process

แต่ถ้าตัวไคลเอนต์ และตัว COM ออบเจ็กต์อยู่ในโพรเซสเดียวกัน ตัว COM+ Interceptor ก็จะอยู่ระหว่างตัวไคลเอนต์ และตัวออบเจ็กต์ ดังรูปที่ 10-5



รูปที่ 10-5 แสดงInterception ที่ใช้กับ COM+ ออบเจ็กต์ชนิด In-Process

การที่มีตัว Interceptor อยู่ระหว่างการเรียกใช้ของไคลเอนต์นั้นทำให้เกิดข้อดีที่เราสามารถที่จะควบคุมการทำงาน และการเรียกใช้งาน ระหว่างตัวไคลเอนต์และตัวออบเจ็กต์ได้โดยที่การทำงานนั้นจะเกิดขึ้นที่ตัว Interceptor ซึ่งวิธีการนี้จะเรียกว่า Interception และ COM+ จะนำไปใช้เพื่อ เริ่มต้นทรานแซคชัน, บังคับให้เกิด fine-grained security ฯลฯ ซึ่งเป็นการให้บริการต่าง ๆของ COM+ Runtime Environment โดยที่ COM ออบเจ็กต์นั้นไม่จำเป็นต้องถูกคอมไพล์ใหม่ หรือต้องเพิ่มโค้ด เพื่อให้สามารถใช้ได้ในระบบ COM+ แต่อย่างใด

10.2 การปรับแต่ง ออบเจ็กต์เพื่อใช้กับ สภาพแวดล้อม COM+ ไทม์

หลังจากที่ได้ศึกษาถึงลักษณะ และการให้บริการต่าง ๆที่ COM+ แล้วก็เป็นส่วนของการนำ COM ออบเจ็กต์ที่ได้พัฒนาไว้แล้วมาปรับแต่งเพื่อให้สามารถใช้กับ COM+ ได้อย่างมีประสิทธิภาพ ดังที่ได้กล่าวไว้แล้วว่า การปรับแต่งคุณสมบัติของ COM ออบเจ็กต์นั้นสามารถทำได้โดยโปรแกรม Component Service Explorer คุณสมบัติส่วนใหญ่นั้นจะมีผลในระดับคลาส แต่ในบางคุณสมบัติ ที่เป็นการแบ่งระดับในการรักษาความปลอดภัย(Fine - Grained Security) ก็สามารถจัดการในระดับย่อยลงไปเป็นระดับ อินเตอร์เฟส หรือเมธอดได้

ข้อจำกัด

1) COM ออบเจ็กต์ทุกตัวที่จะใช้ COM+ นั้นจะต้องเป็น In-Process Server
2) COM+ ออบเจ็กต์ควรใช้ TNA ออบเจ็กต์ที่ใช้ STA และ MTA สามารถทำงานได้แต่ก็มีข้อจำกัด เช่น ออบเจ็กต์ที่เป็น STA และ Primary STA นั้นจะไม่สามารถใช้ใน Object Pooling ได้
3) คอมโพเนนต์ทั้งหลายนั้นจะต้องใช้ การมาร์แชล แบบมาตรฐาน (ไม่สามารถใช้กับ การมาร์แชล แบบกำหนดเอง (Custom Marshaling))

10.3 COM+ แอปพลิเคชั่น

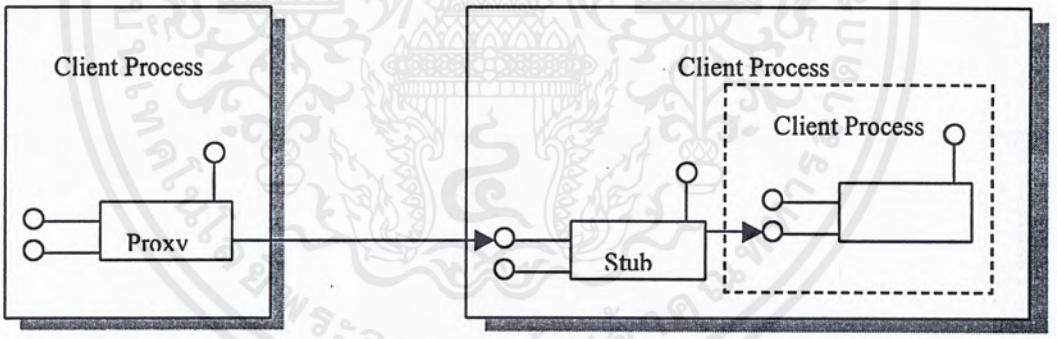
เป็นกลุ่มของ COM+ คอมโพเนนต์ซึ่งถูกจัดการเหมือนกับเป็น กลุ่มหนึ่งกลุ่ม และจำทำการรันในโพรเซส เดียวกันทั้งหมด แอปพลิเคชั่น หนึ่ง ๆ นั้นสามารถจะมีได้หลายเซิร์ฟเวอร์ (ไฟล์ DLL) และในการปรับแต่ง COM+ คอมโพเนนต์ นั้นจำเป็นต้องมีการสร้าง COM+ Application ขึ้นมาเสียก่อน

COM+ แอปพลิเคชั่น นั้นมาแทนที่ MTS Package โดยCOM+ แอปพลิเคชั่น จะมีสองประเภทคือ เซิร์ฟเวอร์แอปพลิเคชั่น และ ไลบรารีแอปพลิเคชั่น

10.3.1 เซิร์ฟเวอร์แอปพลิเคชั่น (Server Application)

แอปพลิเคชั่น ประเภทนี้จะ รันอยู่ในโพรเซสของมัน ซึ่งในความเป็นจริงแล้วมันจะรันอยู่ใน Surrogate Process ซึ่ง COM+ เป็นผู้ให้บริการนี้ Surrogate Process จะเป็นอินแสตนซ์ของ DLLHost.Exe ซึ่งไฟล์นี้จะอยู่ที่ \WinNT System32\

Surrogate Process คือวิธีการที่ทำให้ In-Process Server รันอยู่ในโพรเซสเฉพาะของมันซึ่งอยู่แยกกับโพรเซสของไคลเอนต์โดยการฝังตัว In-Process ที่ทำหน้าที่เป็น Proxy และ Stub ไว้ระหว่าง เซิร์ฟเวอร์และไคลเอนต์ ดังรูปที่ 10-6 ส่วนมากแล้วเรามักจะตั้งให้ แอปพลิเคชั่น เป็น เซิร์ฟเวอร์แอปพลิเคชั่น และเป็นค่าดีฟลอต์ของ COM+ ด้วย



รูปที่ 10-6 แสดง Surrogate Process

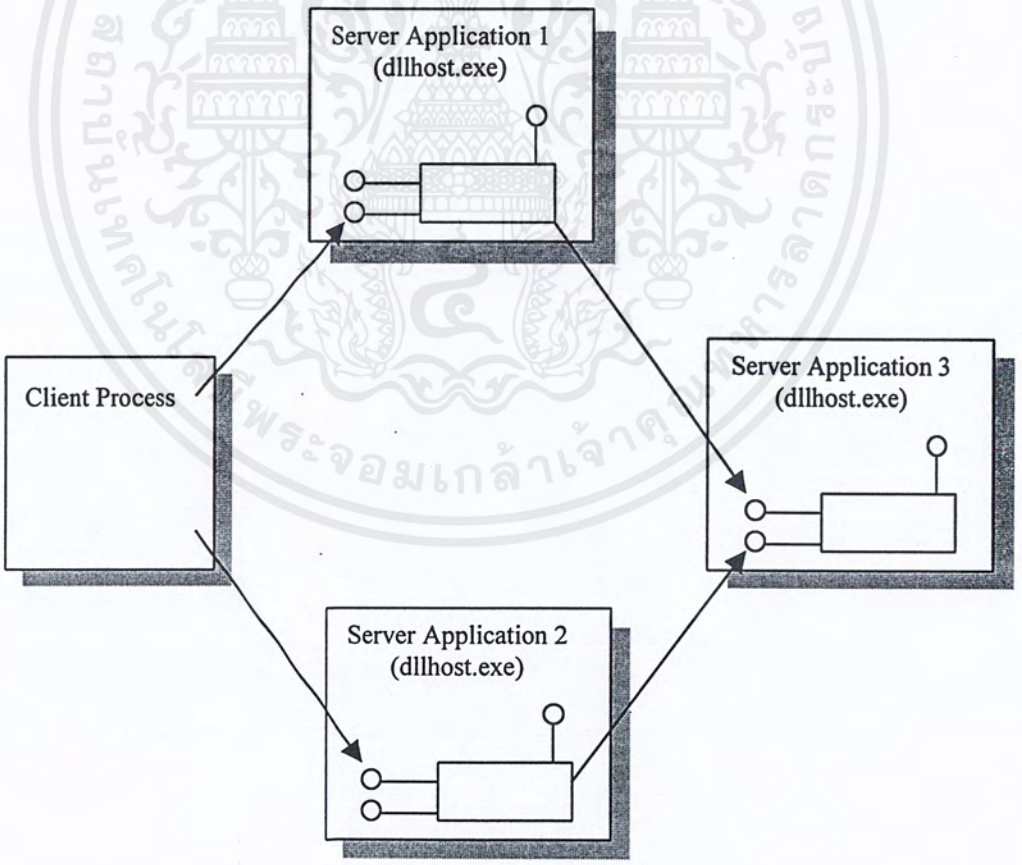
10.3.2 ไลบรารีแอปพลิเคชั่น (Library Application)

แอปพลิเคชั่น ประเภทนี้จะอยู่ในโพรเซส เดียวกับไคลเอนต์ของมัน จุดประสงค์หลักในการเลือกใช้ แอปพลิเคชั่น ประเภทนี้ก็คือสำหรับ COM+ ออบเจ็กต์ ซึ่งถูกเรียกใช้ จาก COM+ ออบเจ็กต์ ด้วยกันหลายๆตัว โดยที่ COM+ ออบเจ็กต์ที่เรียกใช้นั้นอยู่ต่างโพรเซสกันการแบ่งกันใช้ COM+ ออบเจ็กต์ดังรูปที่ 10-7 แต่ถ้าเรานำตัว COM+ ออบเจ็กต์ไปไว้ในโพรเซสเดียวกับ ออบเจ็กต์ที่เรียกใช้จะมีประสิทธิภาพสูงกว่า เนื่องจากรันอยู่ในโพรเซสเดียวกัน ซึ่งจะมีลักษณะดังรูปที่ 10-8

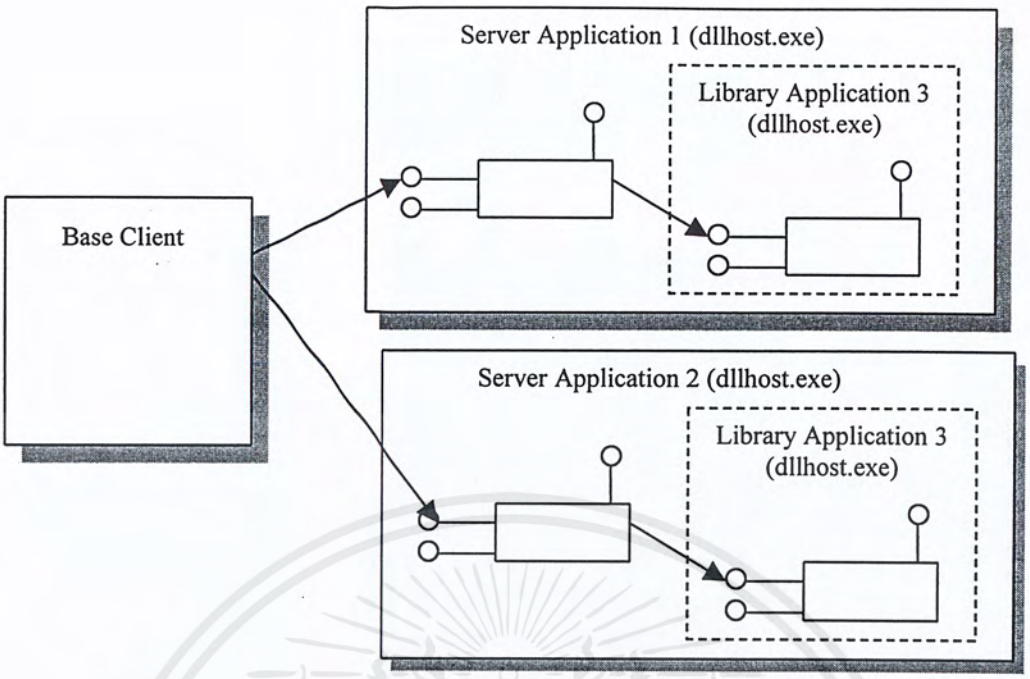
ในทางปฏิบัติเราอาจจะให้ Base Client ติดต่อกับ ไลบรารีแอปพลิเคชั่น ได้ก็จริงแต่หากมองถึงประสิทธิภาพแล้วควรจะหลีกเลี่ยงวิธีนี้

หลักการในการพิจารณาว่าควรจะจัดการกับการแบ่งกลุ่มออบเจ็กต์ต่าง ๆ ว่าจะอยู่ที่แอปพลิเคชั่น ไหนอย่างไรนั้นมีรายละเอียดดังนี้

- ส่วนใหญ่แล้วควรจะมีแค่ แอปพลิเคชั่น เดียวซึ่งรวมคอมโพเนนต์ที่มีทั้งไว้ภายใน
- คอมโพเนนต์ที่อยู่ต่าง แอปพลิเคชั่น กันสามารถรันอยู่ในโพรเซสเดียวกันโดยใช้ COM+ ไลบรารีแอปพลิเคชั่นคอมโพเนนต์ที่อยู่ใน แอปพลิเคชั่น กันจะรันที่โพรเซสเดียวกันตลอด
- คอมโพเนนต์ที่มีความสัมพันธ์กันมาก ควรจะอยู่ใน แอปพลิเคชั่น เดียวกันเพื่อประสิทธิภาพที่ดีกว่า
- สำหรับ COM+ ไลบรารีแอปพลิเคชั่น นั้นเหมาะกับงานที่ต้องการให้ แอปพลิเคชั่น อยู่ในโพรเซสเดียวกับไคลเอนต์ของมันเพื่อประสิทธิภาพ
- คอมโพเนนต์ที่อยู่ใน แอปพลิเคชั่น เดียวกันจะถูกจัดการด้วยกัน เราสามารถที่จะจัดระดับการรักษาความปลอดภัยของออบเจ็กต์ต่าง ๆให้อยู่ในระดับ คลาส หรือ เมธอด ก็ได้ แต่ผลของการจัดระดับนี้จะส่งผลกระทบกับทุก ๆ ออบเจ็กต์ใน แอปพลิเคชั่น นั้นดังนั้น หากมีความจำเป็นที่จะต้องจัดระดับต่างกันในออบเจ็กต์คนละตัว ควรแยกออบเจ็กต์ที่จัดระดับแตกต่างกันนั้นให้อยู่คนละ แอปพลิเคชั่น กัน



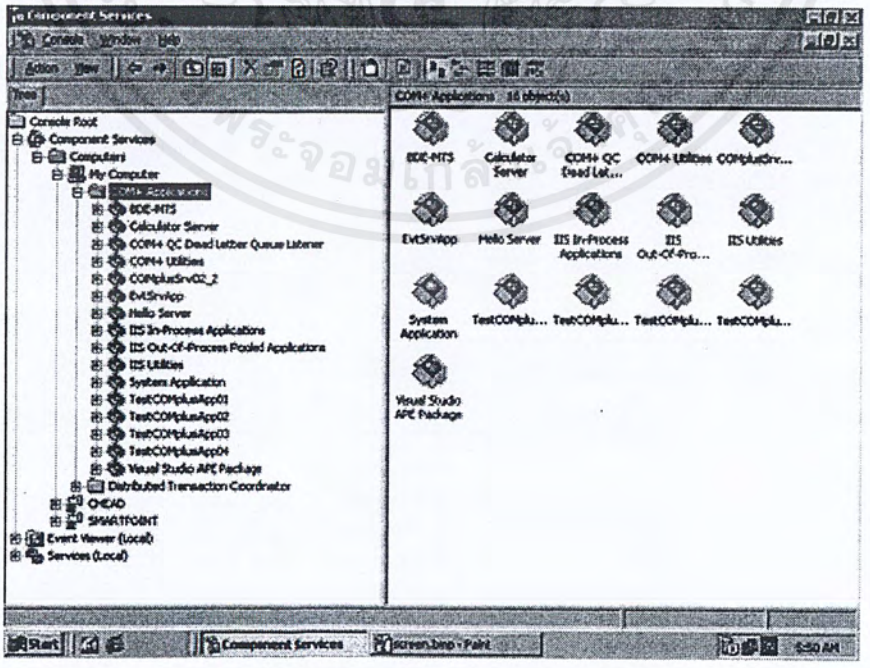
รูปที่ 10-7 แสดงCOM+ Server Application 2 ตัว ใช้ Server Application ตัวที่สามร่วมกัน



รูปที่ 10-8 แสดงCOM+ Server Application 2 ตัวใช้ Library Application ร่วมกัน

10.4 การสร้าง COM+ แอปพลิเคชั่น

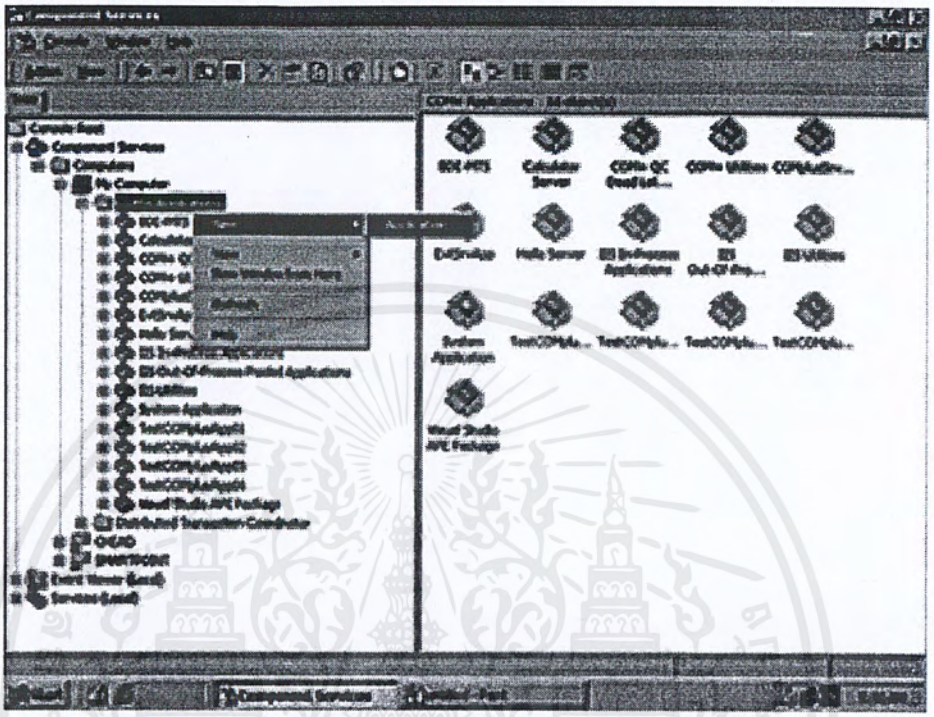
ในการสร้างและปรับแต่ง COM+ ออบเจ็กต์ และ COM+ แอปพลิเคชั่น นั้นจะใช้ Component Service Explorer โดยที่สามารถเรียกใช้ได้จากปุ่ม Start \ Programs \ Administrative Tools \ Component Services \ ซึ่งจะมีลักษณะหน้าจอดังรูปที่ 10-9



รูปที่ 10-9 แสดงComponent Service Explorer

ขั้นตอนการสร้าง COM+ Application ทำได้ดังนี้

1) คลิกขวาที่ Console Root \ Component Services \ Computers \ My Computer \ COM+ Application ดังรูปที่ 10-10

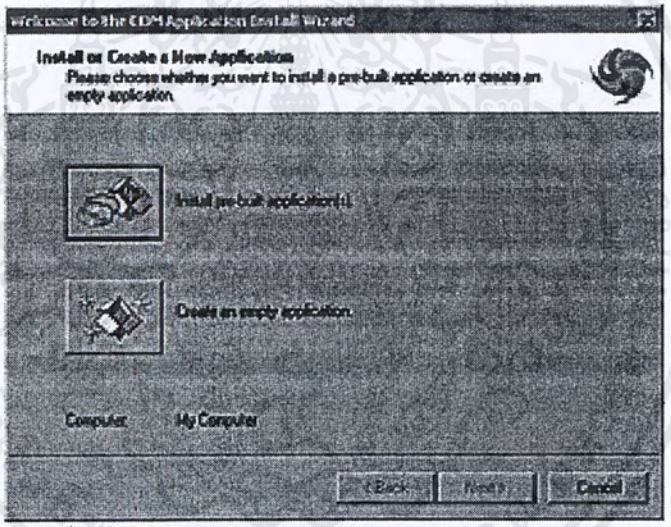


รูปที่ 10-10 แสดงขั้นตอนแรกในการสร้าง COM+ Application

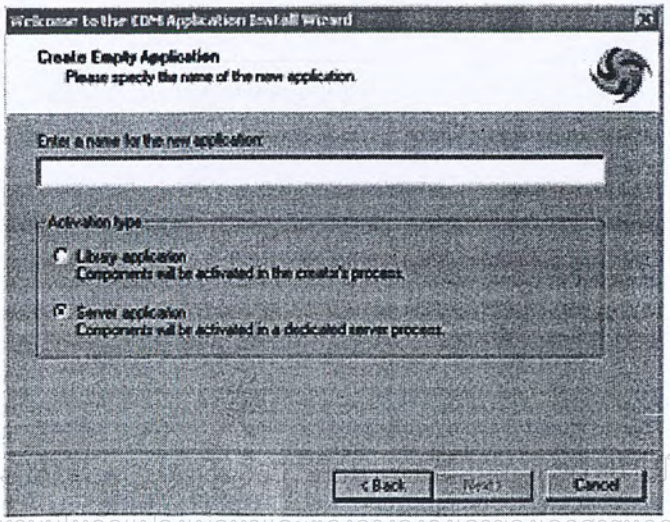
2) จากรูปที่ 7-10 ทำการเลือก New \ Application จะปรากฏหน้าจอดังรูปที่ 10-11
3) หลังจากกดปุ่ม Next จะปรากฏหน้าจอดังรูปที่ 10-12
4) กดปุ่ม Create an empty application แล้วจะเกิดหน้าจอดังรูปที่ 10-13
5) ใส่ชื่อ Application และเลือกชนิดการ Activation จากนั้นกดปุ่ม Next จะเกิดหน้าจอดังรูปที่ 10-14
6) เลือกลักษณะของ Identity
7) กดปุ่ม Next เพื่อสิ้นสุดการสร้าง COM+ Application

เมื่อทำการสร้าง Application ใหม่เสร็จแล้วก็จะมีไอคอนใต้ COM+ Application ใน Component Service ดังรูปที่ 10-15

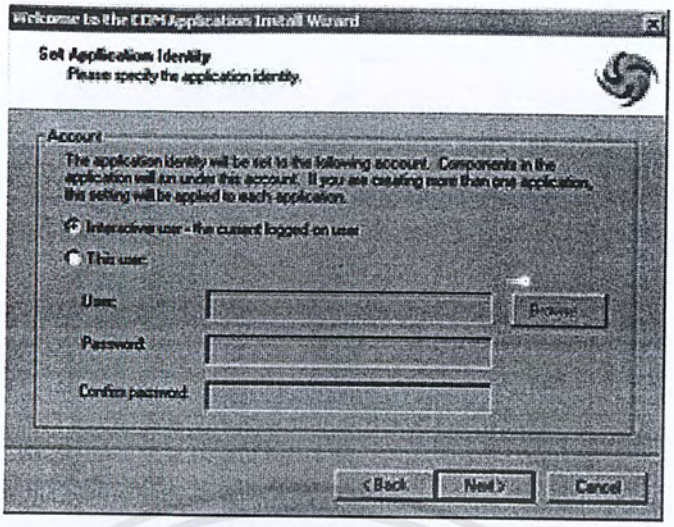
รูปที่ 10-11 แสดงหน้าจอแรกของ COM+ Application Install Wizard



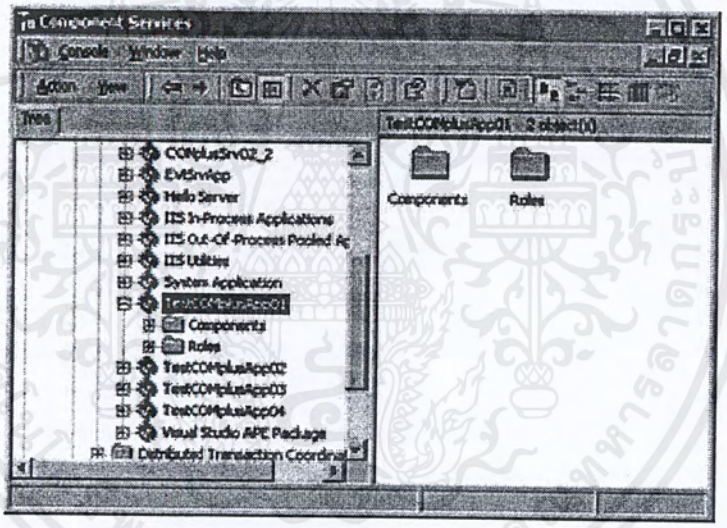
รูปที่ 10-12 แสดงหน้า Install Pre-Built COM+ Application หรือ สร้างใหม่



รูปที่ 10-13 แสดงการกำหนดชื่อของ Application และ ชนิดการ Activate



รูปที่ 10-14 แสดงการเลือก Identity ของ COM+ Application



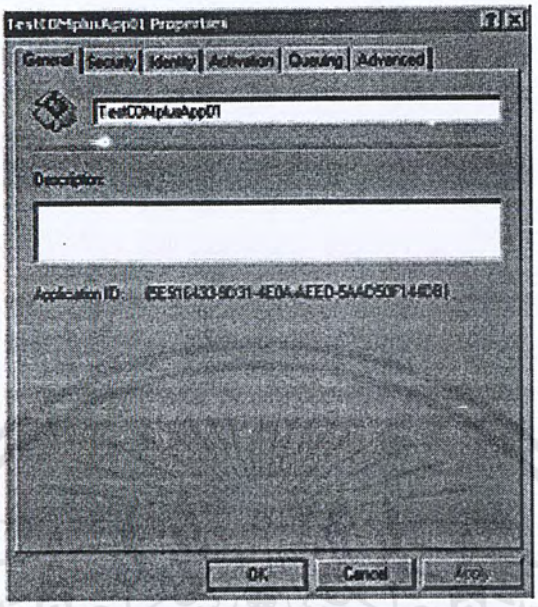
รูปที่ 10-15 แสดงComponent Service Explorer ที่แสดง Application ใหม่

เมื่อเราสร้าง COM+ Application เสร็จดังรูปที่ 10-15 จะมีไอคอนใหม่ที่เกิดขึ้นใน Application นั้นซึ่งได้แก่ Components และ Roles เราจะใส่คอมโพเนนต์ที่จะอยู่ใน Application ไว้ใน Components ส่วนการควบคุม Fine – Grained Security นั้นจะใช้ Role ซึ่งจะได้อธิบายต่อไป

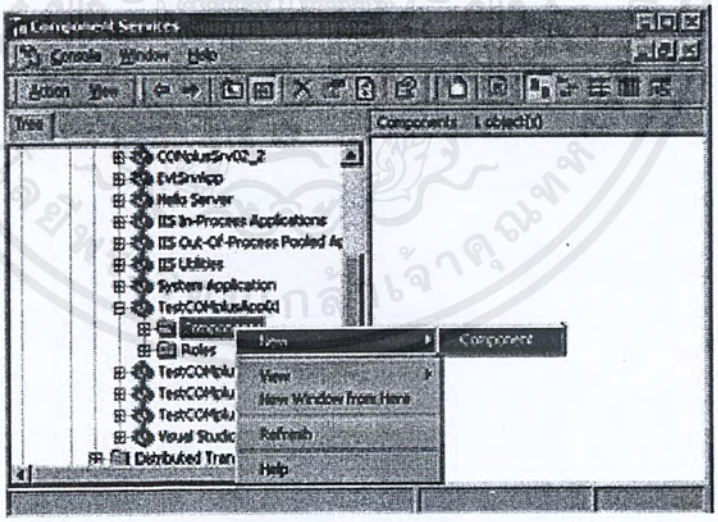
สำหรับขั้นตอนในการดูรายละเอียดในระดับ Application นั้นดูได้โดย

1) คลิกขวาที่ไอคอนของ COM+ Application ที่ต้องการจะดูรายละเอียด

2) เลือก Properties จาก เมนู Context (คุณสมบัติของ COM+ Application จะแสดงดังรูปที่ 10-16 ซึ่งจะใช้ในการปรับแต่งคุณสมบัติต่าง ๆของ COM+ ในระดับ Application ซึ่งจะได้อธิบายต่อไป



รูปที่ 10-16 แสดงGeneral Tab ของ COM+ Application Properties Window



รูปที่ 10-17 แสดงคอมโพเนนต์โฟลเดอร์สำหรับ COM+ Application

ในการใส่คอมโพเนนต์ให้กับ COM+ Application จะใช้ COM ออบเจ็กต์ธรรมดา ซึ่งเป็น In-Process ตัวใด ๆ ก็ได้ใส่ลงใน Application ซึ่งมีขั้นตอนดังนี้

1) คลิกขวาที่โฟลเดอร์ Components ซึ่งอยู่ใต้ COM+ Application ดังรูปที่ 10-17

2) เลือก New \ Component จาก เมนู Context (จะเกิด Component Install Wizard ดังรูปที่ 10-18)

3) กดปุ่ม Next (ขั้นตอนที่ 2 ของ Component Install Wizard จะแสดงดังรูปที่ 10-19)

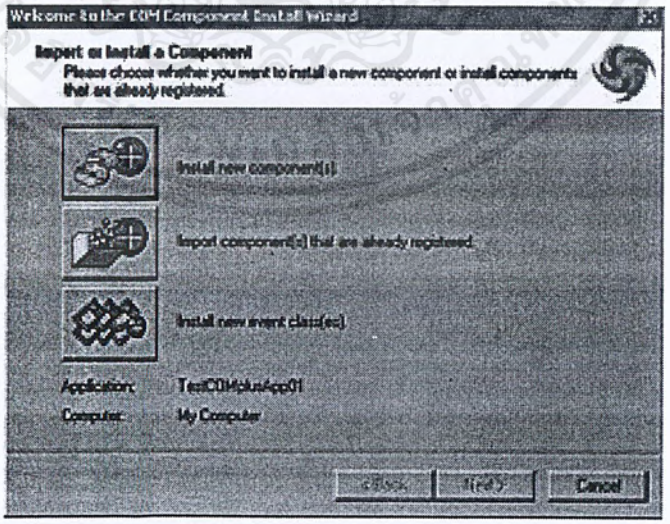
4) กดปุ่ม Install New Components (ขั้นตอนที่ 3 ของ Component Install Wizard จะแสดงดังรูปที่ 10-20)

5) กดปุ่ม Add.. (จะปรากฏไดอะลอก Select Files to Install)

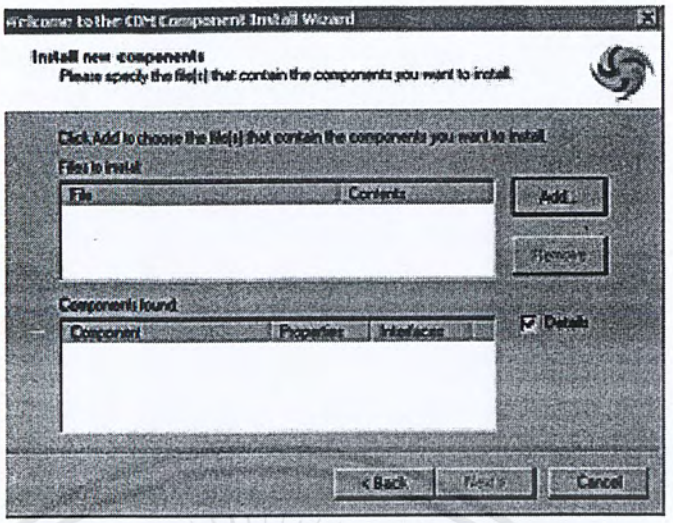
6) เลือกไฟล์ DLL แล้วกดปุ่ม Open

7) กดปุ่ม Next และ Finished

รูปที่ 10-18 แสดงหน้าจอแรกของ Component Install Wizard



รูปที่ 10-19 แสดงการเลือก Install ระหว่างคอมโพเนนต์ใหม่ กับคอมโพเนนต์ที่ลงทะเบียนเรียบร้อยแล้ว

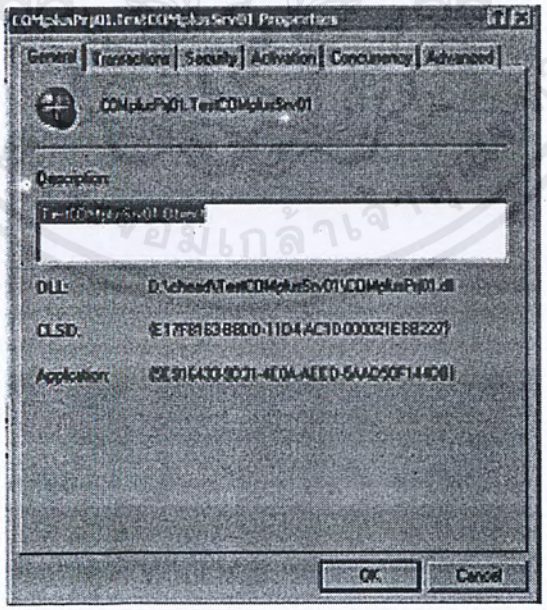


รูปที่ 10-20 แสดงการเลือก คอมโพเนนต์ที่ต้องการจะติดตั้ง

สำหรับการ ดูและปรับแต่งค่าคุณสมบัติต่างของ COM+ ออบเจ็กต์ในระดับคลาสสามารถทำได้โดยมีขั้นตอนดังนี้

1) คลิกขวาที่คลาสใน Component tab

2) เลือก Properties จากเมนู Context (คุณสมบัติ COM+ คอมโพเนนต์จะแสดงดังรูปที่ 10-21)



รูปที่ 10-21 แสดงไดอะลอก COM+ Component Properties

บทที่ 11
เธรดโมเดลของ COM+

11.1 โพรเซส (Process)

อินแสตนซ์ ของโปรแกรม ในแต่ละ โพรเซส จะถือครองทรัพยากรต่าง ๆ เช่น ไฟล์ และ หน่วยความจำ ทรัพยากรทั้งหมดนี้จะถูกทำลายเมื่อ โพรเซส นั้นถูกทำลาย อย่างไรก็ดีเพียงแต่ โพรเซส อย่างเดียวก็ไม่สามารถทำงานอะไรได้ และแต่ละ โพรเซส จะต้องมีอย่างน้อย 1 เธรด

11.2 เธรด (Thread)

ส่วนของการประมวลผลโดยผ่านโค้ดที่อยู่ใน โพรเซส ทุก ๆ Thread จะมี stack และ CPU-state เป็นของตัวเอง และเนื่องจากมี Stack เป็นของตัวเองดังนั้นตัวแปรของแต่ละThread จึงมีความเป็นส่วนตัวจากเธรด อื่นๆ

11.3 มัลติเธรด (Multi-Thread)

ทุก ๆ Thread จะต้องมีอย่างน้อย 1 เธรด ซึ่งเรียกว่า ไพมารีเธรด (Primary Thread) ในการเขียนโปรแกรมให้สามารถใช้ได้เพียงแค่ เธรด เดียวนั้นจะง่าย เนื่องจากไม่ต้องกังวลในเรื่องการที่มีหลาย เธรด อ่านค่าเดียวกันพร้อมกัน แต่ข้อดีของการที่สามารถทำงานได้หลาย ๆ เธรด พร้อมกันนั้นคือสามารทำงานได้อย่างมีประสิทธิภาพกว่า เนื่องจาก เธรด ที่ช้าเนื่องจากต้องติดต่อกับอุปกรณ์ภายนอก เช่น แผ่นดิสก์, เครื่องพิมพ์ เป็นต้น ก็จะไม่ขวางการทำงานของ เธรด อื่น เนื่องจาก เธรด อื่นสามารถทำงานได้

ในระบบปฏิบัติการสมัยใหม่ทั้งหลาย เช่น Windows 98 และ Windows 2000 นั้นสามารถประมวลผล เธรด ได้หลาย ๆ ตัวได้อย่างต่อเนื่อง ระบบปฏิบัติการจะจัดการกับงานเหล่านี้โดยการตั้งช่วงเวลาในการประมวลผลให้กับแต่ละ เธรด เมื่อแต่ละ เธรด ทำงานจนหมดช่วงเวลาของตัวเองแล้ว ระบบปฏิบัติการก็จะทำการบันทึกสถานะของ ซีพียู และ เธรด นั้นไว้แล้วทำการเปลี่ยนให้ เธรด อื่นมาทำงานแทน จนกว่าจะถึงเวลาที่มันจะทำการประมวลผลใหม่ก็จะทำการโหลดข้อมูลเหล่านั้นมาเพื่อทำงานต่อไป

11.4 Thread-Safe

ซอร์สโค้ดที่สามารถทำงานในแบบ มัลติเธรด(Multi-Thread) ได้อย่างปลอดภัย ซึ่งมักจะใช้วิธี thread-synchronization primitive เช่น Critical Section ในขณะที่จะประมวลผลซึ่งไม่ปลอดภัย

11.5 Multi-Threading และ COM

จุดประสงค์หลักจุดประสงค์หนึ่งของ COM คือการสร้างออบเจ็กต์ให้ง่ายต่อการนำกลับมาใช้ใหม่ โดยไม่ขึ้นกับภาษาที่ใช้พัฒนา และตำแหน่งของตัวออบเจ็กต์ อย่างไรก็ดีปัญหาของระบบซึ่งเป็นแบบ มัลติเธรด นั้นจะต้องพิจารณาใหม่ เนื่องจากเราไม่สามารถนำระบบที่ไม่เป็นแบบThread-Safeมาทำงานในระบบงานที่เป็นแบบ Multi-Thread ได้

ดังนั้นนอกจากที่ต้องการให้ คอมโพเนนต์นั้นสามารถทำงานได้โดยไม่ขึ้นกับภาษาที่ใช้พัฒนา และ ตำแหน่งของตัวคอมโพเนนต์ แล้ว เรายังต้องการให้มีความเป็นอิสระต่อลักษณะที่แตกต่างกันของ เธรด ด้วย

ในระบบที่เป็น COM นั้นทำได้โดยการพิจารณาว่าออบเจ็กต์ที่จะเรียกใช้งานนั้นเป็นแบบ Thread-Safe หรือไม่ค่าที่เป็นตัวบ่งชี้ว่าเป็นหรือไม่นั้นสามารถดูได้จากรีจิสทรีของออบเจ็กต์นั้นเมื่อระบบงานที่เป็นแบบMulti-Thread จะพยายามเรียกใช้ออบเจ็กต์ซึ่งมีลักษณะไม่เป็น Thread-Safe COM+รันไทม์จะทำการ Serialize การเรียกใช้ต่างนั้นให้โดยอัตโนมัติ

จากหลักการที่ได้กล่าวมาทำให้เกิดปัญหาที่ตามมาหลายประการดังนี้

- จะทำให้ COM รันไทม์ รู้ได้อย่างไรว่าแอปพลิเคชั่นนั้นเป็น มัลติเธรด
- จะทำให้ COM ออบเจ็กต์บ่งชี้ว่าตัวเองนั้นเป็น Thread-Safe
- COM+ รันไทม์ จะทำการ Serialize การเรียกเมธอดได้อย่างไรเมื่อมันทำการเรียกใช้ COM ออบเจ็กต์ที่ไม่เป็น Thread-Safe จากแอปพลิเคชั่นที่เป็น มัลติเธรด

ก่อนที่จะศึกษาถึงวิธีการที่จะแก้ไขปัญหาเหล่านี้ เราต้องเข้าใจถึงหลักการพื้นฐานที่อยู่ภายในแต่ละโพรเซส เสียก่อน ซึ่งหลักการพื้นฐานนี้คือ อพาร์ตเมนท์ นอกจากนั้นปัญหาในของวิธีการ Serialize ของ COM รันไทม์ นั้นก็จะใช้ อพาร์ตเมนท์จัดการด้วย

11.6 อพาร์ตเมนท์

เป็น สถานที่ทางจินตภาพ (logical construct) ที่มีลักษณะคล้ายกับสภาพแวดล้อมของ เธรด ซึ่ง COM ออบเจ็กต์ หนึ่งหรือมากกว่า ออบเจ็กต์สามารถประมวลผลได้ภายใน และ COM ออบเจ็กต์ทุกตัวที่อยู่ใน โพรเซส จะต้องอยู่ใน อพาร์ตเมนท์ใด อพาร์ตเมนท์หนึ่ง เนื่องจากในแต่ละโพรเซส ก็สามารถมีได้หลาย อพาร์ตเมนท์

เฉพาะ เธรด ที่เข้าถึง อพาร์ตเมนท์ ของออบเจ็กต์นั้นๆ ถึงจะสามารถเรียกใช้เมธอดของออบเจ็กต์นั้นได้โดยตรง สำหรับ เธรด อื่นนั้นจะสามารถเรียกใช้ได้โดยผ่านทาง การส่งแมสเสจไปยัง เธรด ใด ๆที่อยู่ภายใน อพาร์ตเมนท์ นั้นแทน

อพาร์ตเมนท์ มีประเภทต่าง ๆ ดังนี้

1) แบบที่สามารถมีได้เพียง Thread เดียวซึ่งเรียกว่า STA (Single - Threaded Apartment) ซึ่งทุก ๆ ออบเจ็กต์ใน อพาร์ตเมนท์ นั้นจะถูกเข้าถึงได้จากเธรดเพียงเธรดเดียว ซึ่งจะมีปัญหาเกี่ยวกับการใช้ทรัพยากรเมื่อต้องทำงานในแบบที่เป็น มัลติเธรด
2) แบบทีสามารถมีได้หลายThread หรือ MTA (Multi-Threaded Apartment) ซึ่งสามารถให้บริการ เมธอดต่าง ๆ ที่อยู่ใน อพาร์ตเมนท์ ให้กับ Thread ต่าง ๆ ได้อย่างต่อเนื่อง แต่ออบเจ็กต์ที่อยู่ใน MTA ได้นั้นต้องเป็น Thread-Safe

แต่ละ โพรเซส นั้นสามารถมีได้หลาย STA แต่จะมีเพียงแค่ MTA เดียวเท่านั้น

ออบเจ็กต์ทุกตัวจะระบุว่าตัวมันนั้นต้องการ สภาพแวดล้อมที่มันต้องการโดยระบุไว้ที่ โมเดลของเธรด ซึ่งสามารถดูได้จาก รีจิสทรีของระบบที่ตำแหน่ง

HKEY_CLASSES_ROOT\ CLSID\ CLSID_FOR_A_COM_CLASS \ ThreadingModel

เช่น

HKEY_CLASSES_ROOT\ CLSID\ {06637330-314D-11D3-998A-E0EC08C10000}\ ThreadingModel = Apartment

ค่าที่เป็นไปได้ของ โมเดลเธรด มีทั้งหมด 5 ค่าได้แก่

1) Single มีชนิดของ อพาร์ตเมนท์ เป็น STA ออบเจ็กต์ประเภทนี้จะไม่เป็น Thread-Safe ไม่ใช่แค่ตัวออบเจ็กต์เท่านั้นแต่ยังรวมถึง Class Factory สำหรับคลาส และฟังก์ชัน DllGetClassObject และ DllCanUnloadNow ก็เช่นกัน Single-Thread สามารถทำการเรียกใช้ทุก ๆ เมธอดที่อยู่ในเซิร์ฟเวอร์

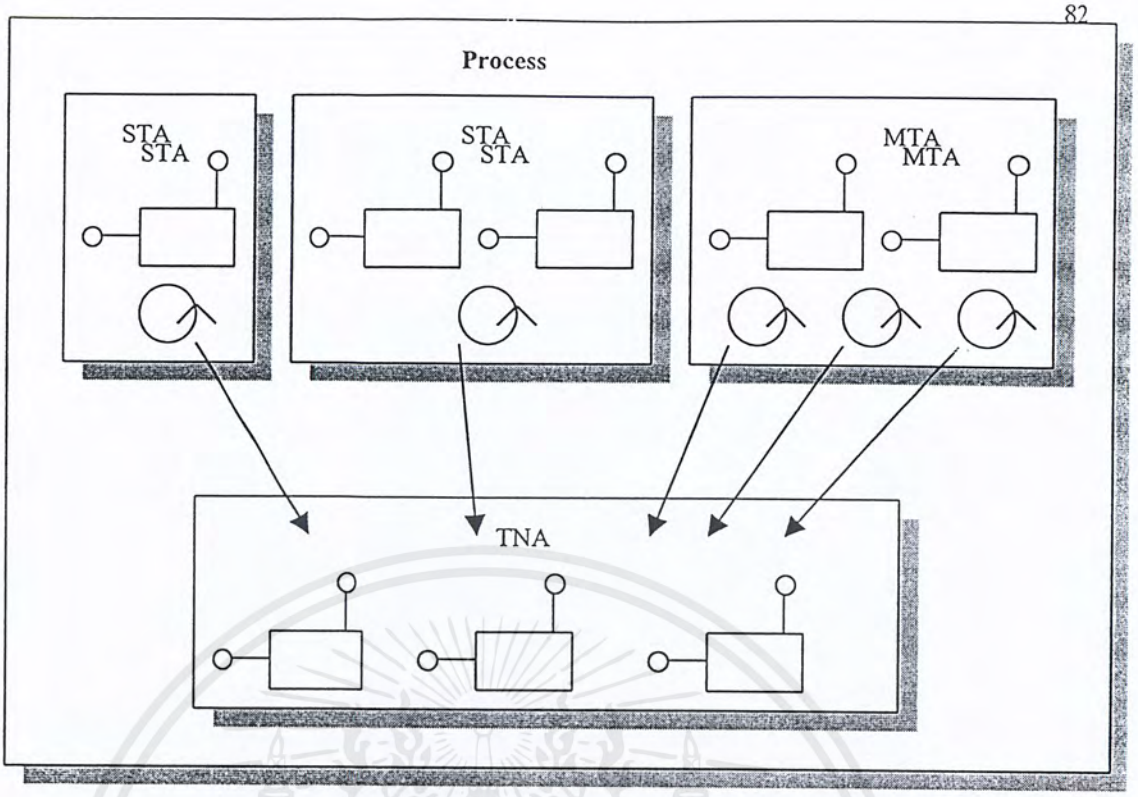
2) Apartment มีชนิดของอพาร์ตเมนท์เป็น STA ออบเจ็กต์ประเภทนี้จะสามารถป้องกันปัญหาของระบบ Multi-Thread ได้เฉพาะในระบบดับคลาสเท่านั้น เนื่องจาก Class Factory และ ฟังก์ชัน DllGetClassObject และ DllCanUnloadNow ในคลาสของมันเป็น Thread-Safe แต่ในระดับ อินแสตนซ์จะถูกจำกัด เนื่องจากการเข้าถึงแต่ละอินแสตนซ์จะถูกกำหนดให้เป็น Single-Thread เท่านั้น

3) Free มีชนิดของ อพาร์ตเมนท์ เป็น MTA ออบเจ็กต์ประเภทนี้สามารถป้องกันปัญหาของระบบ มัลติเธรด ทั้งสถานะต่ออินแสตนซ์ และสถานะต่อคลาสโดยใช้ Thread Synchronization Primitives เช่น Critical Section หรือ Mutex ออบเจ็กต์เหล่านี้สามารถทำงานในสถานะที่เป็น มัลติเธรดได้อย่างปลอดภัย

4) Both สามารถมีชนิดของอพาร์ตเมนท์ ได้ทั้ง STA และ MTA ออบเจ็กต์ประเภทนี้จะสร้างตัวเองอยู่ในอพาร์ตเมนท์เดียวกับไคลเอนต์ของมันซึ่งทำให้มีประสิทธิภาพสูงสุด และทำงานในเธรดเดียวกับผู้ที่เรียกมัน และเนื่องจากมันสามารถทำงานได้ในระบบที่เป็นมัลติเธรดมันจึงป้องกับปัญหาที่จะเกิดในระดับคลาส และ ระดับอินแสตนซ์

5) NeuTral มีชนิดของอพาร์ตเมนท์เป็น TNA (Thread Neutral Apartment) ออบเจ็กต์ประเภทนี้จะทำงานในเธรดของผู้เรียกแต่อยู่ในอพาร์ตเมนท์ของมันเองแทนที่จะเป็นอพาร์ตเมนท์ ของมันเนื่องจากเธรดที่อยู่ในอพาร์ตเมนท์ ประเภทอื่นในโพรเซสเดียวกันสามารถเข้าถึง TNA ได้ทุกเวลา และออกจากอพาร์ตเมนท์เมื่อมันทำงานเมื่อการทำงานของเมธอดนั้นเสร็จ เนื่องจากอพาร์ตเมนท์ประเภทนี้สามารถทำการ Thread-Synchronization primitives ได้ภายในตัวของมันเอง หรือสามารถ Thread-Synchronize ได้ด้วยบริการของ COM+ ซึ่งจะได้อธิบายโดยละเอียดในภายหลัง

ความสัมพันธ์ระหว่าง โพรเซส, อพาร์ตเมนท์, เธรด และ COM ออบเจ็กต์อธิบายได้ดังรูปที่11-1



รูปที่ 11-1 แสดงProcess, Apartment, Thread และ COM ออบเจ็กต์

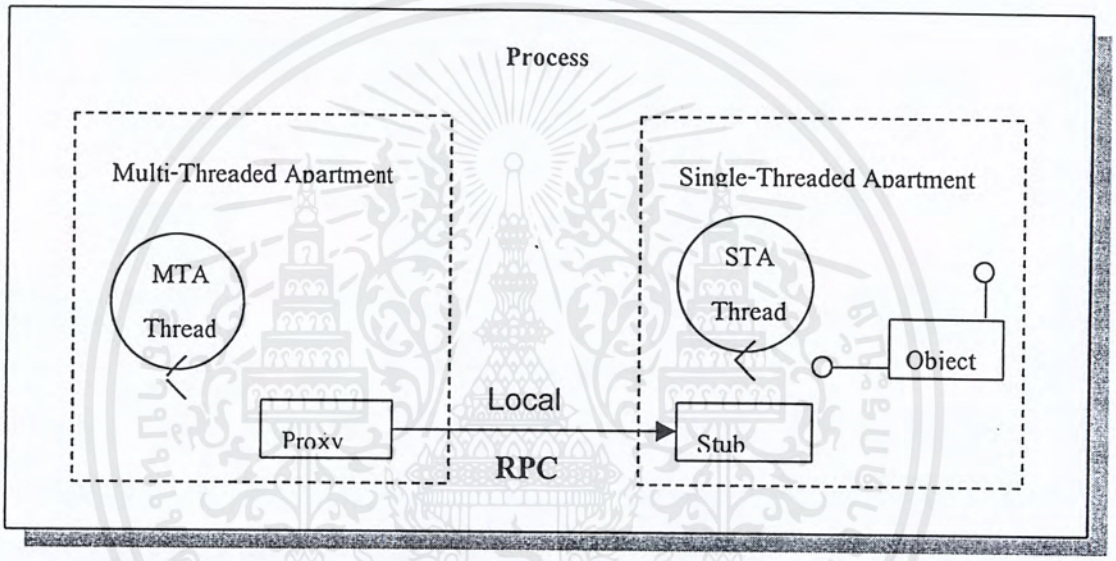
11.7 วิธีที่COM+ runtime จัดการกับ User Apartment

ถ้า COM ออบเจ็กต์ระบุว่าตัวมันนั้นเป็น Thread-Safe (ThreadingModel = Free) ตัว COM + รันไทม์ จะมองตัวออบเจ็กต์นั้นรันอยู่ใน MTA หากออบเจ็กต์นั้นถูกสร้างใน MTA Thread (ตัวไคลเอนต์เป็นมัลติเธรด) แล้วCOM+ เธรด ก็ไม่จำเป็นต้องทำอะไรเพิ่มเติม แต่หากตัวออบเจ็กต์นั้นถูกสร้างในSTA เธรด (ตัวไคลเอนต์เป็น Single-Thread) ทาง COM+ รันไทม์ จะทำการสร้าง MTA เธรด ขึ้นใหม่การสร้าง STA เธรด จะได้รับ Marshaled Interface Pointer (ซึ่งจะได้อธิบายต่อไป) หลาย ๆ Thread สามารถเข้าถึง MTA และทุก ๆ เธรด ที่เข้าถึง MTA สามารถเรียกใช้ทุก ๆ เมธอดจากทุก ๆ ออบเจ็กต์ที่อยู่ใน MTA การที่สามารถทำได้ดังที่กล่าวมาแล้ว ก็เนื่องเธรดโมเดล แบบ Free นั้นมีส่วนจัดการ การ Synchronization ให้โดยอัตโนมัติจึงไม่ต้องกังเกี่ยวกับปัญหาที่เกิดขึ้นในระบบที่เป็น Single-Thread หรือ Multi-Thread

แต่ถ้า COM ออบเจ็กต์นั้นไม่เป็น Thread-Safe (เธรดโมเดล เป็น Apartment หรือ Single) COM+ รันไทม์ จะมองว่าออบเจ็กต์นั้นรันอยู่ที่ เธรด STA ถ้าเธรดโมเดล ของออบเจ็กต์เป็น อพาร์ตเมนท์ และออบเจ็กต์ถูกสร้างจากเธรด STA ออบเจ็กต์นั้นจะถูกสร้างอยู่ในอพาร์ตเมนท์ เดียวกับเธรด ที่สร้างมัน ถ้าออบเจ็กต์นั้นถูกสร้างจาก MTA Thread เมื่อ COM+รันไทม์ สร้างเธรด และSTA ของออบเจ็กต์ ให้มัน COM+ รันไทม์ จะส่ง Marshaled Interface กลับมาให้กับเธรด MTA ด้วย และถ้าเธรดโมเดล ของออบเจ็กต์เป็น Single ออบเจ็กต์นั้นจะอยู่ใน Primary STA ของ Process (Primary STA เป็น STA แรกที่ถูกสร้างในโพรเซส) นอกจากนี้ COM+ รันไทม์ จะสร้าง ออบเจ็กต์ไว้ใน primary STA และส่ง Marshaled Interface คืนไปยัง อพาร์ตเมนท์ ที่เรียกมัน

11.8 การเรียกใช้เมธอดแบบ Inter-Apartment

ถึงแม้ว่าเราจะอนุญาติให้เธรดเดียวสามารถใช้งานเมธอดที่อยู่ใน STA ได้โดยตรง แต่ก็ไม่ได้หมายความว่าจะไม่อนุญาติให้เธรดอื่นๆเข้าไปใช้งานเมธอดนั้นแต่จะต้องเรียกใช้เมธอดนั้นโดยอ้อม (Indirect Method Call) หากมีเธรดที่อยู่ใน อพาร์ตเมนท์หนึ่งเรียกใช้เมธอดที่อยู่ในออบเจ็กต์ที่อยู่ใน อพาร์ตเมนท์อื่น COM+ รันไทม์จะตรวจสอบว่าการเรียกใช้เมธอดถูกต้องหรือไม่โดยดูจากว่าการเรียกใช้เมธอดนั้นจะเกิดจากเธรด ที่อยู่ในอพาร์ตเมนท์ของออบเจ็กต์ที่ถูกเรียกใช้ สำหรับลักษณะในการติดต่อนั้นจะเหมือนกับมาแชลลิ่งซึ่งใช้ในการเรียกใช้เมธอดผ่านโพรเซสและผ่านเครื่องซึ่งเรียกวิธีนี้ว่า Intra-Process Marshalingหรือ Thread Switch ดังรูปที่ 11-2



รูปที่ 11-2 แสดงIntra-Process Marshaling

อย่างไรก็ดีข้อวิธีการ Intra-Process Marshaling ก็มีจุดด้อยที่ค่อนค้างช้ามากเมื่อเทียบกับการเรียกใช้ เมธอดภายใน Apartment เดียวกัน

11.9 Marshalling Interface Pointer ระหว่าง อพาร์ตเมนท์

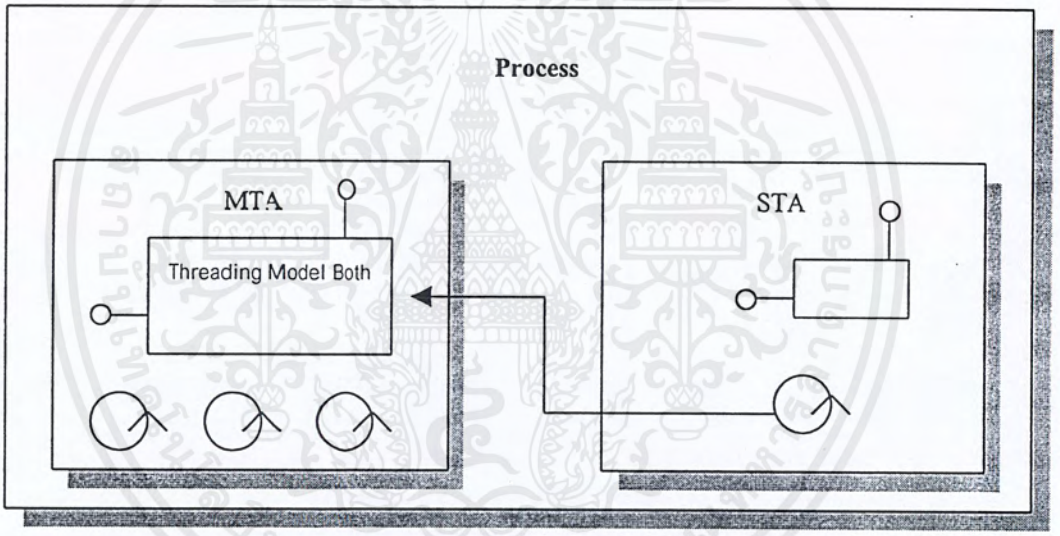
เมื่อ เธรด ใดๆ ซึ่งรันอยู่ใน อพาร์ตเมนท์ หนึ่ง เรียกใช้เมธอดซึ่งอยู่ในออบเจ็กต์ที่อยู่ในอีก อพาร์ตเมนท์ หนึ่งจะทำใช้ Interface Pointer จัดการติดต่อสื่อสารระหว่าง อพาร์ตเมนท์ โดยที่จะติดตั้งตัวพร็อกซี่อยู่ที่ฝั่งไคลเอนต์ และติดตั้งตัวพร็อกซี่ ที่ฝั่งของตัวออบเจ็กต์

11.10 Threading Model แบบ Both และ Neutral

รายละเอียดของเธรดโมเดลทั้งสองนี้มีจุดที่เหมือนและแตกต่างกันอยู่พอสมควร จุดที่เหมือนกันก็คือเธรดโมเดลทั้งสองถูกออกแบบมาเพื่อหลีกเลี่ยงการใช้ Thread-Switch จากการเรียกใช้เมธอดซึ่งอยู่ต่างอพาร์ตเมนท์โดยการให้ออบเจ็กต์ซึ่งมีเธรดโมเดลแบบนี้ทำงานในเธรดของไคลเอนต์ ส่วนความแตกต่างระหว่างเธรดโมเดล ทั้งสองนั้นสามารถอธิบายได้จากรายละเอียดของมันดังนี้

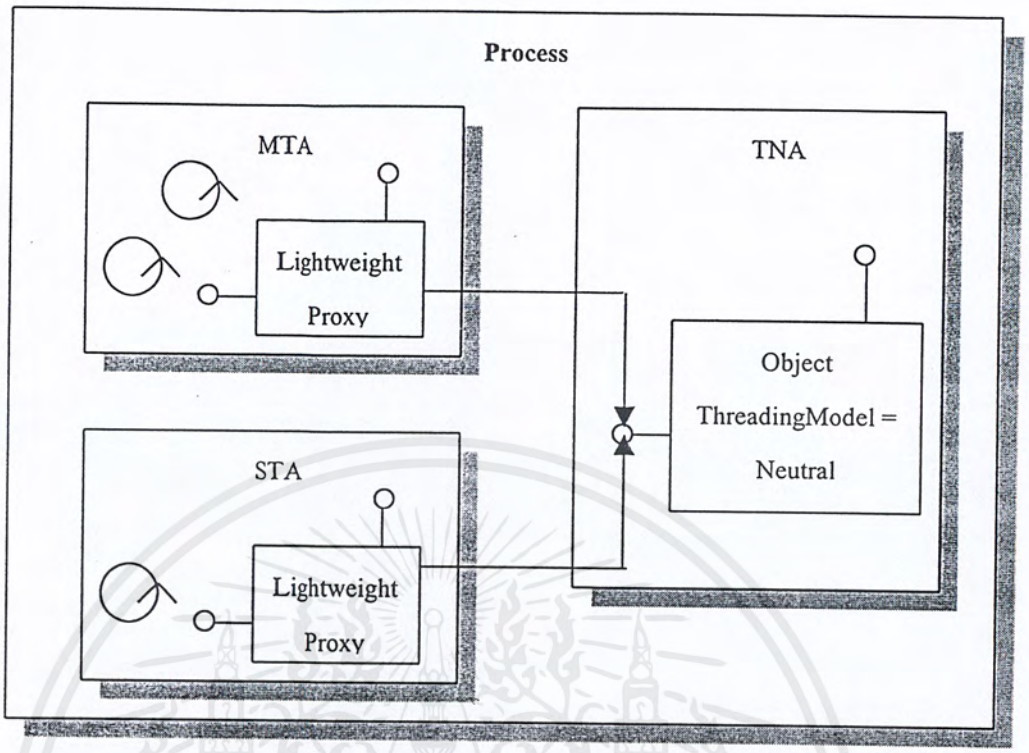
- Both เป็นชื่อที่ตั้งได้ไม่ค่อยดีนัก เนื่องจากในสมัยที่ออกแบบ เธรดโมเดล ชนิดนี้ยังมีอพาร์ตเมนท์ แค่สองแบบคือ STA และ MTA แต่ในความเป็นจริงแล้วมันก็สามารถทำงานให้กับออบเจ็กต์ที่มีเธรดโมเดลเป็นแบบ TNA ได้เช่นเดียวกัน และเนื่องจากออบเจ็กต์ที่สามารถใช้เธรดโมเดล ชนิดนี้ได้นั้นอาจเป็น MTA หรือ TNA ก็ได้ดังนั้นออบเจ็กต์ทุกตัวที่จะมี เธรดโมเดล ชนิดนี้ต้องมีความเป็น Thread-Safe ด้วย

  จุดด้อยของเธรดโมเดลชนิดนี้ก็คือไม่สามารถที่จะจำกัดการเรียกใช้ Thread-Switch ได้ เช่น ในกรณีที่มี MTA Thread เรียกใช้ ออบเจ็กต์ตัวหนึ่งที่มี เธรดโมเดลเป็น Both ตัวออบเจ็กต์นั้นก็จะถูกสร้างไว้ใน อพาร์ตเมนท์MTA นั้น หากมี STAเธรดอีก เธรดหนึ่งต้องการเรียกใช้ออบเจ็กต์ตัวนั้น ดังรูปที่ 11-3 ในกรณีนี้ก็ยังจำเป็นต้องเรียกใช้ Thread-Switch หรือ หากจะหลีกเลี่ยงจริง ๆ ก็ต้องใช้ FTM (Free-Threaded Marshaler) ร่วมกับ GIT ซึ่งค่อนค้างยุ่งยากเมื่อเทียบกับการใช้ Neutral



รูปที่ 11-3 แสดงเธรด ที่เป็น MTA และ STA และออบเจ็กต์ที่มี เธรดโมเดลชนิด Both

- Neutral เป็น Threading Model ที่ถูกเพิ่มเข้ามาใน Windows 2000 ซึ่งถูกออกแบบมาในแนวทางเดียวกันกับ Both โดยปราศจากความซับซ้อนในการใช้ FTM และ GIT ความแตกต่างระหว่างทั้งสองแบบนี้ก็คือ ทุก ๆ Thread นั้นไม่มีขอบเขตที่แบ่งแยกกันอย่างชัดเจน Thread ใน STA และ MTA สามารถเรียกใช้ออบเจ็กต์ที่อยู่ใน TNA โดยผ่านทาง Lightweight Proxy ดังรูปที่ 11-4



รูปที่ 11-4 แสดงอพาร์ตเมนท์ชนิด STA และ MTAเรียกใช้ออบเจ็กต์ที่รันอยู่ใน TNA

Lightweight Proxy นั้นจะไม่ทำการ Thread Switch แต่กฎข้ออื่นของอพาร์ตเมนท์ก็ยังใช้ได้และเช่นเดียวกับเธรดโมเดลแบบ Both ทุกๆ ออบเจ็กต์จะเป็น Thread-Safe การใช้การเธรดโมเดลชนิดนี้ไม่จำเป็นต้องคีย์โปรแกรมลงไปเองเนื่องจากเป็นส่วนบริการของ สภาพแวดล้อม COM+ รันไทม์อยู่แล้วมันจะทำการ Thread-Synchronization โดยไม่จำเป็นต้องเขียนโค้ดลงไปเอง สำหรับรายละเอียดของการทำ Thread-Synchronization จะได้อธิบายภายหลัง

อย่างไรก็ตามเธรดโมเดลชนิด TNA นั้นก็ไม่สามารถที่ใช้ได้ คอมโพเนนต์ที่ถูกออกแบบมาเพื่อให้มีเพียงเธรดที่เหมาะสมเธรดเดียวเท่านั้นที่สามารถเรียกใช้มันได้ (เราเรียกเธรด ประเภทนี้ว่า Thread affinity) ซึ่งได้คอมโพเนนต์ ประเภทที่ต้องแสดงผล (Visual Component) เป็นต้น สำหรับออบเจ็กต์พวกนี้จำเป็นต้องใช้เธรดโมเดลที่เป็นอพาร์ตเมนท์เป็นชนิด STA อย่างเดียว เท่านั้น

11.11 วิธีพิจารณาว่าแอปพลิเคชั่นเป็นมัลติเธรด

จากที่ได้กล่าวถึงเรื่องอพาร์ตเมนท์คำถามนี้จึงเป็นคำถามที่ผิด คำถามที่ถูกต้องควรเป็นว่า COM+ รันไทม์ จะพิจารณาว่าเธรดที่เข้ามาเรียกใช้ ออบเจ็กต์นั้นเป็นเธรดที่อยู่ในส่วนของสภาพแวดล้อมที่มี เธรดเดียวหรือ อยู่ในสภาพแวดล้อมที่มีหลายเธรด สำหรับคำตอบก็คือแต่ละเธรดจะต้องประกาศว่าตัวมันเป็น Single – Thread หรือ Multi-Thread โดยดูจากว่า เธรดเหล่านั้นทำงานอยู่ในอพาร์ตเมนท์ที่เป็น STA หรือ MTA นั่นเอง

11.12 วิธีการที่ออบเจ็กต์จะบอกว่าเป็น Thread-Safe

ในเรื่อง Apartment ที่ได้พูดกล่าว ชนิดของ Apartment และ ค่าต่าง ๆ ของคีย์ เธรดโมเดล ซึ่งจะพบว่าชนิด STA ซึ่งจะมีค่าคีย์ เธรดโมเดล เป็น Single และ Apartment นั้นไม่เป็น Thread-Safe ดังนั้นจึงไม่สามารถนำออบเจ็กต์ซึ่งอยู่ในอพาร์ตเมนท์ ชนิดนี้ไปใช้ในลักษณะมัลติเธรดได้ ซึ่งค่าของคีย์นี้ สามารถดูได้จาก registry ของระบบซึ่งได้กล่าวไปแล้วในหัวข้ออพาร์ตเมนท์หาก COM คลาสไม่ระบุค่า เธรดโมเดลจะกำหนดโดยอัตโนมัติให้เป็น Single

บทที่ 12
ทรานแซคชัน

ทรานแซคชัน คือ หน่วยของงาน หรือกลุ่มคำสั่งระดับลอจิคอล เช่น SQL โดยในกลุ่มของคำสั่งเหล่านี้อนุญาตให้มีการละเมิด Integrity Constraints (Integrity Rule) ได้ ตัวอย่างเช่น การโอนเงินประกอบด้วยงานสองอย่าง คือ การถอนเงินออกจากบัญชีหนึ่งและนำไปฝากที่อีกบัญชีหนึ่ง Integrity Rule คือ "R:ในการโอนเงินจะไม่มียอดเงินหายไปจากระบบ" ถ้าต้องการโอนเงินจากบัญชี A ไปบัญชี B เป็นเงิน 1,000 บาท ทำได้ดังนี้

A $\xrightarrow{1{,}000\ Baht}$ B

1 ทรานแซคชัน
- begin transaction (Disable R)
- ถอนเงินจากบัญชี A 1,000 Bath
- ฝากเงินที่บัญชี B 1,000 Bath → R ถูกละเมิด
- end transaction (Restore R)

จะเห็นว่าในขณะทำการถอนหรือฝาก Integrity Rule จะถูกละเมิด จนกว่าจะทำงานทั้งสองอย่างเสร็จสิ้น กลุ่มขอองงานดังกล่าวจึงเป็น 1 ทรานแซคชัน

12.1 คุณสมบัติของทรานแซคชัน

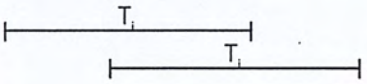
เป็นที่รู้จักกันในชื่อคุณสมบัติ ACID ประกอบด้วย

A (Atomic) - ทรานแซคชันต้องถูกมองเป็นหน่วยเดียว แบ่งย่อยไม่ได้ (ถ้าระบบทำงานล้มเหลว ต้องยกเลิกทั้งทรานแซคชัน ถ้าสำเร็จต้องสำเร็จหมดทั้งทรานแซคชัน)

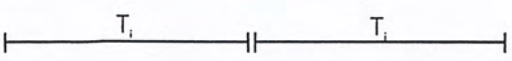
C (Correctness หรือ Consistency) - ทรานแซคชันต้องรักษาความถูกต้องตาม Integrity Rule เมื่อเสร็จทรานแซคชันแล้ว

I (Isolation) - ถ้ามีทรานแซคชันอื่นปฏิบัติพร้อมๆกับทรานแซคชันของเราในช่วงเวลาเดียวกัน ทรานแซคชันของเราต้องไม่ได้รับผลกระทบจากทรานแซคชันอื่นๆเหล่านั้น

เช่น



ต้องทำงานเหมือนกับ



หรือ

D (Durability) - เมื่อมีทรานแซคชันสำเร็จ (commit) ทรานแซคชันนั้นต้องอยู่ถาวรเช่น เมื่อทรานแซคชันเสร็จแล้ว system fail เมื่อ recovery กลับมาต้องเห็นทรานแซคชันอยู่

ระบบจัดการฐานข้อมูล (Database Management System, DBMS) จะมีโครงสร้างภายในที่สนับสนุนการเริ่มต้น, คอมมิท และโรลแบ็กทรานแซคชัน ขั้นตอนพื้นฐานในการทำทรานแซคชันแสดงได้ดังนี้

```
try {
    Transaction.Begin
    ถอนเงิน 1000 บาท จากบัญชี A
    ฝากเงิน  1000 บาท จากบัญชี B
    Transaction.Commit
  }
 catch (Exception) {
   // ถ้าเกิดเหตุการณ์ผิดพลาด ให้โรลแบ็กการทำงานทั้งหมด
    Transaction.Rollback
  }
```

เริ่มต้นทรานแซคชันโดยใช้ฟังก์ชัน Begin เมื่อทำงานเสร็จสมบูรณ์จึงใช้ฟังก์ชัน Commit เพื่อคอมมิท ทรานแซคชัน แต่ถ้าเกิดผิดพลาดระหว่างการทำงานเราสามารถใช้ฟังก์ชัน Rollback เพื่อยกเลิกการทำงานทั้งหมดนับตั้งแต่เรียกใช้ฟังก์ชัน Begin จนถึงตำแหน่งที่เกิดความผิดพลาด

12.2 ทรานแซคชันแบบกระจาย (Distribute Transaction)

ในกรณีที่ข้อมูลถูกจัดเก็บอยู่ในดาตาเบสตัวเดียว เราสามารถจัดการทรานแซคชันโดยใช้ฟังก์ชันของระบบจัดการฐานข้อมูลได้ แต่โดยทั่วไปแล้วข้อมูลต่าง ๆ ถูกจัดเก็บไว้ที่ดาตาเบสซึ่งอยู่ในหลาย ๆ เซิร์ฟเวอร์ หากการทำทรานแซคชันเสร็จสมบูรณ์ รีซอร์สเมเนเจอร์ทั้งหมดจะต้องคอมมิทส่วนของทรานแซคชันที่ตัวเองทำอยู่ หากไม่สามารถคอมมิทได้ก็จะต้องโรลแบ็กทั้งหมดเช่นกัน

รีซอร์สเมเนเจอร์เป็นชื่อในเทอมของกระบวนการทรานแซคชัน หมายถึงระบบที่ทำงานกับออบเจ็กต์ใด ๆ ที่มันถือครองอยู่โดยรักษาคุณสมบัติ ACID ตัวอย่างรีซอร์สเมเนเจอร์ที่เห็นได้ชัดคือ RDBMS เช่น SQL Server หรือ Oracle

12.3 โพรโตคอลแบบทู-เฟสคอมมิท (2-Phase Commit Protocal)

สิ่งสำคัญในการทำทรานแซคชันแบบกระจายคือโพรโตคอลแบบทู-เฟสคอมมิท ในโพรโตคอลนี้กิจกรรมต่างๆของรีซอร์สเมเนเจอร์จะถูกควบคุมโดยส่วนของซอฟแวร์ซึ่งเรียกว่า ทรานแซคชันเมเนเจอร์ (Transaction Manager) หรือ ทรานแซคชันโคออร์ดิเนเตอร์ (Transaction Coordinator) ขั้นตอนของโปรโตคอลมีดังนี้

1. แอปพลิเคชั่นเรียกใช้เมธอดคอมมิทในทรานแซคชันโคออร์ดิเนเตอร์

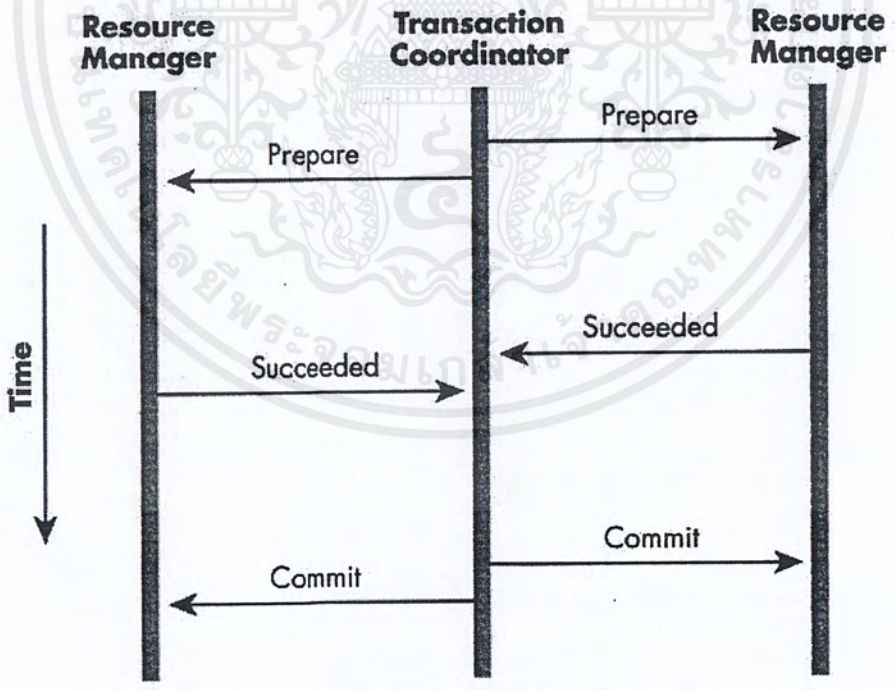
2. ทรานแซคชันโคออร์ดิเนเตอร์ติดต่อกับรีซอร์สเมเนเจอร์แต่ละตัวเพื่อให้จัดการทรานแซคชันและบอกให้เตรียมคอมมิททรานแซคชัน (เป็นจุดเริ่มต้นของเฟสที่หนึ่ง)

3. รีซอร์สเมเนเจอร์แต่ละตัวจะวางตัวอยู่ในสถานะที่จะการันตีกับทรานแซคชันโคออร์ดิเนเตอร์ว่าจะคอมมิทหรือโรลแบ็กทรานแซคชัน โดยรีซอร์สเมเนเจอร์ทั้งหมดจะส่งไฟล์ที่ระบุสิ่งที่มันจะปฏิบัติในกรณีที่พร้อมจะคอมมิท แต่ถ้าไม่สามารถคอมมิทได้มันจะตอบสนองกับทรานแซคชันโคออร์ดิเนเตอร์ว่าไม่สามารถคอมมิทได้

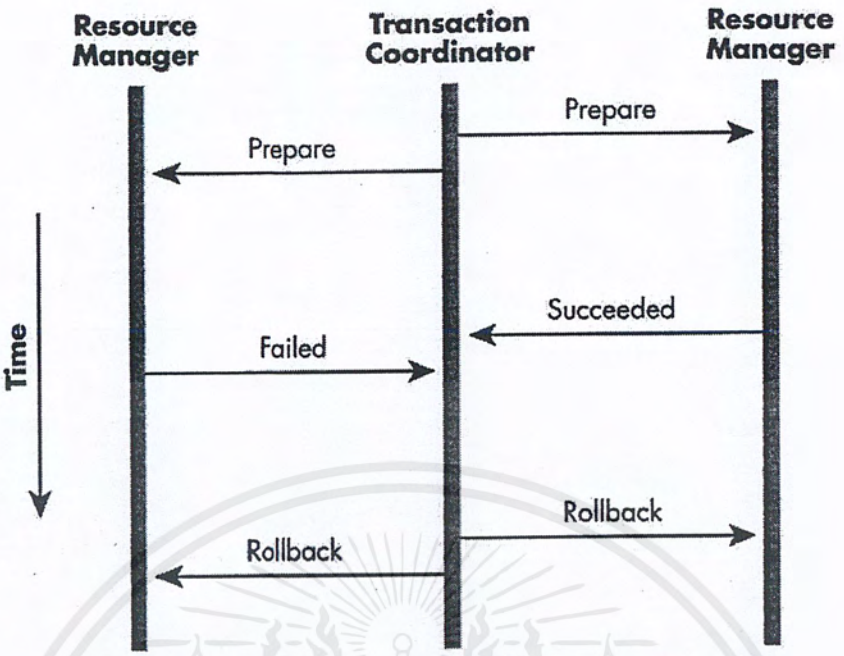
4. ทรานแซคชันโคออร์ดิเนเตอร์ รวบรวมการตอบสนองทั้งหมดของรีซอร์สเมเนเจอร์

5. เฟสที่สอง ทรานแซคชันโคออร์ดิเนเตอร์แจ้งผลลัพธ์ไปยังรีซอร์สเมเนเจอร์แต่ละตัวเกี่ยวกับทรานแซคชันนั้น โดยถ้าในเฟสที่หนึ่งหากมีรีซอร์สเมเนเจอร์ตัวใดตัวหนึ่งส่งคำสั่งตอบสนองว่าไม่สามารถคอมมิทได้ ทรานแซคชันโคออร์ดิเนเตอร์ก็จะส่งคำสั่งโรลแบ็กไปยังรีซอร์สเมเนเจอร์ทั้งหมดให้โรลแบ็กทรานแซคชันนั้น แต่ถ้ารีซอร์สเมเนเจอร์ทั้งหมดตอบสนองในเฟสที่หนึ่งว่าสามารถคอมมิทได้ ทรานแซคชันโคออร์ดิเนเตอร์ก็จะส่งคำสั่งคอมมิทไปยังรีซอร์สเมเนเจอร์ทุกตัว เพื่อให้คอมมิททรานแซคชัน

6. โพรโตคอลแบบทู-เฟสคอมมิท แสดงเป็นแผนภาพลำดับขั้นได้ดังรูปที่ 12-1 และ 12-2 โดยรูปที่ 12-1 เป็นกรณีที่ทรานแซคชันคอมมิท ส่วนรูปที่ 12-2 เป็นกรณีที่ทรานแซคชันโรลแบ็ก



รูปที่ 12-1 การคอมมิททรานแซคชันโดยใช้โพรโตคอลทู-เฟสคอมมิท



รูปที่ 12-2 การโรลแบ็กทรานแซคชันโดยใช้โพรโตคอลทู-เฟสคอมมิท

12.4 ทรานแซคชันกับ COM+

COM+ สนับสนุนการทำทรานแซคชันแบบกระจายสำหรับ COM คอมโพเนนต์ โดยเราไม่ต้องเขียนโค้ดที่ซับซ้อน กระบวนการต่างๆ เป็นดังนี้

1. ปรับแต่งออบเจ็กต์ที่ทำงานภายใต้ COM+
2. ตั้งค่าแอตทริบิวต์ของออบเจ็กต์ว่าจะให้เป็นแบบ Requires a Transaction, Requires a New Transaction หรือ Supports Transactions
3. ใช้เมธอด SetComplete ของออบเจ็กต์คอนเท็กซ์ (Object Context) เมื่อพร้อมที่จะคอมมิททรานแซคชัน
4. ใช้เมธอด SetAbort ของออบเจ็กต์คอนเท็กซ์ (Object Context) เมื่อพร้อมที่จะโรลแบ็กทรานแซคชัน

ตารางที่ 12-1 สรุปความหมายของค่าต่างๆ ของทรานแซคชันแอตทริบิวต์ทั้งหมดของ COM+ ออบเจ็กต์ดังนี้

ทรานแซคชันแอตทริบิวต์	คำอธิบาย
Requires a Transaction	ออบเจ็กต์นี้จะต้องทำงานภายใต้ทรานแซคชัน COM+ จะเริ่มต้นทรานแซคชันใหม่สำหรับออบเจ็กต์นี้ก็ต่อเมื่อ ไม่มีออบเจ็กต์อื่น ๆ ที่อยู่ในคอนเท็กซ์เดียวกัน เริ่มต้นทรานแซคชันเลย
Requires a New Transaction	ออบเจ็กต์นี้จะต้องทำงานภายใต้ทรานแซคชัน COM+ จะเริ่มต้นทรานแซคชันใหม่ให้เมื่อออบเจ็กต์นี้ถูกสร้างขึ้น
Supports Transaction	ออบเจ็กต์นี้จะทำงานภายใต้ทรานแซคชันก็ต่อเมื่อ ผู้สร้าง (Creator) ได้สร้างออบเจ็กต์ที่เริ่มต้นทรานแซคชันไว้ก่อนแล้ว ไม่เช่นนั้นออบเจ็กต์จะทำงานโดยปราศจากทรานแซคชัน
Does Not Support Transactions	ออปเจ็กต์นี้ไม่ต้องทำงานแบบทรานแซคชัน และ COM+ จะไม่รับออบเจ็กต์นี้ภายใต้ทรานแซคชันใหม่ (เป็นค่าดีฟอลต์)
Disabled	ออบเจ็กต์ไม่สนใจที่จะเลือกค่าใดๆ ในทรานแซคชันแอตทริบิวต์ ตัวเลือกนี้ทำให้คอมโพเนนต์มีลักษณะเป็นคอมโพเนนต์ที่ยังไม่ได้ตั้งค่าอะไรเลย

ตารางที่ 12-1 ค่าทรานแซคชันแอตทริบิวต์

การเริ่มต้นทรานแซคชัน

หลังจากที่ตั้งค่าทรานแซคชันให้กับคอมโพเนนต์ COM+ รันไทม์จะอ่านค่าทรานแซคชันแอตทริบิวต์เมื่อเริ่มสร้างออบเจ็กต์ หลังจากนั้นจะติดต่อกับทรานแซคชันโคออร์ดิเนเตอร์ที่เรียกว่า Distribute Transaction Coordinator (DTC) โดยจะบอกให้ DTC เริ่มต้นทรานแซคชันสำหรับออบเจ็กต์นี้ DTCจะเริ่มต้นทรานแซคชันให้กับออบเจ็กต์แล้วส่งค่า GUID ของทรานแซคชันกลับมา (DTC จะกำหนด GUID ให้กับทุกทรานแซคชัน) GUID นี้จะเก็บอยู่ในคอนเท็กซ์ของออบเจ็กต์นั้น เราสามารถใช้เมธอด IsInTransaction และ GetTransactionId ของอินเทอร์เฟส IObjectContextInfo เพื่อดูว่าออบเจ็กต์นี้ทำงานอยู่ภายใต้ทรานแซคชันหรือไม่ และมี GUID ทรานแซคชันอะไร

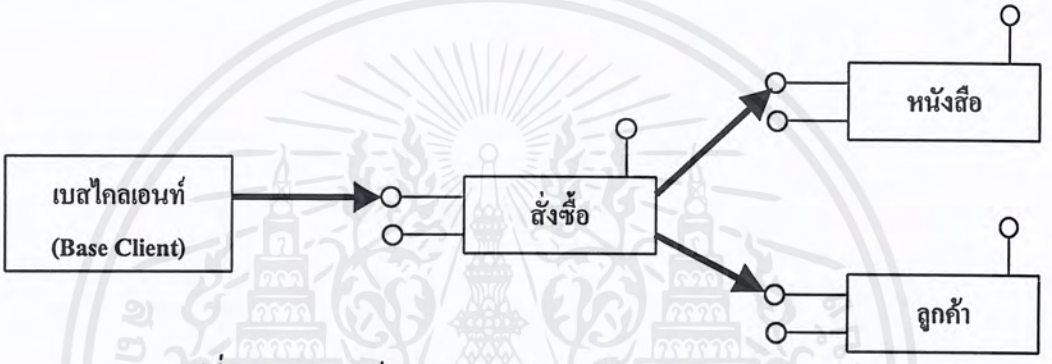
ออบเจ็กต์ที่เป็นตัวเริ่มต้นทรานแซคชันจะเรียกว่าออบเจ็กต์ราก (Root Object) ของทรานแซคชัน ออบเจ็กต์รากมีความสำคัญมากเนื่องจากว่า COM+ รันไทม์จะไม่คอมมิททรานแซคชัน จนกว่าออบเจ็กต์รากจะถูกดีแอกติเวท (Deactived) ออบเจ็กต์อื่น ๆ ที่อยู่ภายใต้ทรานแซคชันเดียวกันจะเป็นเพียงตัวลงความเห็นว่าควรจะคอมมิททรานแซคชันหรือไม่ แต่ออบเจ็กต์รากเป็นตัวกำหนดว่าเมื่อไหร่ที่จะคอมมิททรานแซคชัน

เมื่อ COM+ ออบเจ็กต์ที่อยู่ภายใต้ทรานแซคชันทำการติดต่อกับรีซอร์สเมเนเจอร์ รีซอร์สเมเนเจอร์นั้นก็จะอยู่ในทรานแซคชันด้วย รีซอร์สเมเนเจอร์จะติดต่อกับ DTC เพื่อบอกว่าขณะนี้มันเป็นส่วนหนึ่งของทรานแซคชัน ถ้าออบเจ็กต์ติดต่อกับรีซอร์สเมเนเจอร์หลายตัว รีซอร์สเมเนเจอร์เหล่านั้นทั้งหมดก็จะอยู่ในทรานแซคชันเช่นกัน

เมธอด SetComplete เป็นการโหวตเพื่อคอมมิททรานแซคชัน ออบเจ็กต์รากเป็นตัวบ่งบอกว่าขณะนี้พร้อมคอมมิทแล้ว DTC จะทำงานโดยใช้โปรโตคอลทู-เฟสคอมมิท เพื่อคอมมิทหรือโรลแบ็กทรานแซคชันตามขั้นตอนที่ได้กล่าวมาแล้วขั้นต้น

12.5 ทรานแซคชันที่เกี่ยวข้องกับออบเจ็กต์มากกว่า 1 ตัว

ก่อนหน้านี้เราได้กล่าวถึง COM+ ออบเจ็กต์ตัวเดียวที่ทำงานในลักษณะที่เป็นทรานแซคชันและมีการติดต่อกับรีซอร์ทเมเนเจอร์ตั้งแต่สองตัวขึ้นไป แต่ในหลาย ๆ กรณีทรานแซคชันมักจะเกี่ยวข้องหรือใช้ออบเจ็กต์หลาย ๆ ตัว ตัวอย่างเช่นการสั่งหนังสือออนไลน์ สมมติว่าให้ออบเจ็กต์การสั่งซื้อเป็นออบเจ็กต์รากของทรานแซคชัน ออบเจ็กต์การสั่งซื้อมีการสร้างและเรียกใช้ออบเจ็กต์ลูกค้ากับออบเจ็กต์หนังสือดังรูปที่ 12-3



รูปที่ 12-3 ออบเจ็กต์หลาย ๆ ตัวทำงานอยู่ในทรานแซคชันเดียวกัน

เมื่อ COM+ ออบเจ็กต์ถูกสร้างโดย COM+ ออบเจ็กต์ด้วยกัน ออบเจ็กที่ถูกสร้างขึ้นใหม่จะมีลักษณะการทำงานแบ่งได้เป็น 3 ทางคือ (1) อยู่ในทรานแซคชันเดียวกับออบเจ็กที่สร้างมัน (2) สร้างทรานแซคชันของตนเองขึ้นมาใหม่ (3) ไม่ทำงานเป็นทรานแซคชันเลย กรณีทั้งสามจะเกิดขึ้นอยู่กับการตั้งค่าทรานแซคชันแอตทริบิวต์ให้กับออบเจ็กต์ต่าง ๆ

ถ้าออบเจ็กต์สั่งซื้อและออบเจ็กต์ลูกค้าต่างมีทรานแซคชันเป็นของตัวเอง ออบเจ็กต์สั่งซื้ออาจคอมมิทงานของตน โดยที่ออบเจ็กต์ลูกค้าอาจทำงานแล้วเกิดปัญหาทำให้ต้องโรลแบ็กงานของตน จะเห็นได้ว่าออบเจ็กต์ทั้งสองอาจทำงานขัดแย้งกันได้ซึ่งเป็นเหตุการณ์ที่เราไม่ต้องการ หากออบเจ็กต์ทั้งคู่ทำงานอยู่ในทรานแซคชันเดียวกัน การคอมมิทหรือโรลแบ็กทรานแซคชันจะขึ้นอยู่กับออบเจ็กต์ทั้งหมดที่อยู่ในทรานแซคชันนั้น ออบเจ็กต์แต่ละตัวจะเสนอการคอมมิทโดยเรียกใช้เมธอด SetComplete ของอินเทอร์เฟส IObjectContext หรืออาจเสนอให้โรลแบ็กโดยใช้เมธอด SetAbort ของอินเทอร์เฟส IObjectContext เช่นเดียวกัน ถ้าออบเจ็กต์ทุกตัวเสนอให้คอมมิท ทรานแซคชันจึงจะคอมมิท หากมีตัวใดตัวหนึ่งเสนอให้โรลแบ็ก ทรานแซคชันจะต้องโรลแบ็ก สิ่งสำคัญประการหนึ่งคือ การเรียกใช้เมธอด SetComplete เป็นเพียงการเสนอว่าออบเจ็กต์นั้นยอมรับและพร้อมคอมมิท ทรานแซคชัน ออบเจ็กต์จะยังไม่ทราบว่าทรานแซคชันคอมมิทแล้วจริง ๆ หรือไม่ ดังนั้น COM+ ออบเจ็กต์ที่ทำงานในลักษณะที่เป็นทรานแซคชันไม่ควรสรุปว่าสถานะของตนหลังจากใช้เมธอด SetComplete ไปแล้วจะเป็นสถานะสุดท้ายที่ถูกต้องเสมอไปเมื่อทรานแซคชันคอมมิท ตัวอย่างเช่น ถ้าออบเจ็กต์ลูกค้าทำงานโดยหักเงินของบัญชีลูกค้าในฐานข้อมูล

ออบเจ็กต์ลูกค้ายังไม่ควรเปลี่ยนแปลงสถานะยอดเงินจนกว่าทรานแซคชันจะคอมมิท เพราะหากเปลี่ยนแปลงแล้วออบเจ็กต์อื่น ๆ ที่ทำงานในทรานแซคชันเดียวเสนอให้โรลแบ็กทรานแซคชัน จะทำให้ออบเจ็กต์ลูกค้าอยู่ในสถานะที่ไม่ถูกต้องโดยเงินในบัญชีลูกค้าถูกหักออกไปโดยที่การสั่งหนังสือทำงานไม่สำเร็จ ในการป้องกันปัญหาดังกล่าว COM+ จะดีแอกติเวทออบเจ็กต์ทั้งหมดที่เกี่ยวข้องกับทรานแซคชันเมื่อทรานแซคชันคอมมิท การดีแอกติเวทออบเจ็กต์เป็นการเพิ่ม scalability ของระบบ แต่เหตุผลของในเรื่องของ ทรานแซคชันคือเพื่อความความถูกต้องของงานที่ปฏิบัติ

บทที่ 13
Scalability

Scalability ซึ่งในที่นี้หมายถึงความสามารถที่แอปพลิเคชั่นสามารถจัดการกับงานที่มีจำนวนมากขึ้นเนื่องจากในระบบงานที่ต้องมีการติดต่อกับไคลเอนต์เป็นจำนวนมาก ทุก ๆ ไคลเอนต์ก็จะมีออบเจ็กต์ของมันเอง และจะถือครองออบเจ็กต์เหล่านั้นตลอดเวลา เมื่อมีไคลเอนต์เป็นจำนวนมากๆ จึงต้องเสียทรัพยากรของเซิร์ฟเวอร์เป็นจำนวนมากสำหรับออบเจ็กต์เหล่านั้นตลอดเวลา และจะทำให้ถึงขีดจำกัดของเซิร์ฟเวอร์ได้ง่ายซึ่งทำให้ไม่สามารถสร้างระบบงานที่มีการติดต่อกับไคลเอนต์ที่มีจำนวนมากขึ้นได้ นอกจากนั้นการที่ครองออบเจ็กต์ตลอดเวลานั้นไม่คุ้มค่าเนื่องจากไคลเอนต์นั้นจะเรียกใช้ออบเจ็กต์เพียงช่วงเวลาสั้น

ในการแก้ปัญหาที่เกิดขึ้นนั้น COM+ ได้เตรียมบริการ ไว้สองตัวคือ JIT (Just-In Time) Activation และ Object Pooling ซึ่งจะร่วมกันในการเพิ่มความสามารถในการขยายขนาด (Scalability)

13.1 JIT Activation

จากที่ได้กล่าวมาแล้วว่าในระบบงานที่มีการติดต่อกับไคลเอนต์นั้น ตัวไคลเอนต์จะทำการสร้างออบเจ็กต์ไว้ตั้งแต่เริ่มโปรแกรม และจะครองไว้จนกว่าจะจบการทำงาน สำหรับในเหตุผลก็คือ ในการสร้างออบเจ็กต์บ่อยถือเป็นการทำงานที่เปลืองทรัพยากรโดยไม่จำเป็นเมื่อนึกถึง ไคลเอนต์ที่อยู่ต่างเครื่องยิ่งต้องเสียเวลาติดต่อผ่านเน็ตเวิร์คและการทำงานอย่างอื่นอีกมากกว่าที่จะสามารถสร้างออบเจ็กต์ไว้ที่ฝั่งเซิร์ฟเวอร์ได้ อย่างไรก็ดังที่ได้กล่าวมาแล้วว่าการถือครองออบเจ็กต์ไว้ตลอดเวลาก็ไม่เป็นผลดีเช่นกัน ดังนั้นการแก้ปัญหาก็คือใช้หลักการที่จะให้ไคลเอนต์นั้นถือครองอินสแตนซ์ของออบเจ็กต์เมื่อเวลาที่มันต้องการเรียกใช้งานทรานแซคชั่นเท่านั้นและ รีบคืนออบเจ็กต์นั้นให้กับระบบโดยเร็วที่สุดหลักจากที่ทรานแซคชั่นนั้นคอมมิท (Commit)หรือไม่ก็โรลแบ็ค (Roll back)

จากหลักการที่ได้กล่าวมาแล้วนั้นเป็นวิธีที่ยากที่จะให้ผู้พัฒนาโปรแกรมเขียนโค้ดเพื่อทำงานนี้ให้ใช้งานง่ายดังนั้น COM+ จึงได้จัดเตรียมฟังก์ชันเหล่านี้ไว้ให้โดยเรียกว่า JIT (Just-In Time) Activation โดยที่สามารถควบคุมได้โดยเป็นลักษณะแอททริบิวต์เช่นเดียวกับบริการอื่นๆ ของ COM+ Runtime โดยมีขั้นตอนการทำงานคือ

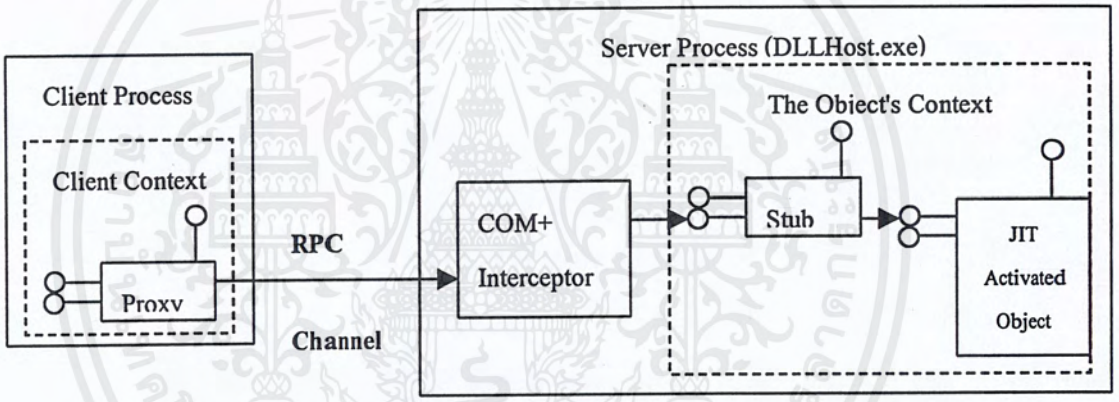
เมื่อมีการสร้างออบเจ็กต์ขึ้นใหม่ COM+ Runtime จะสร้าง Proxy และ Stub มาตรฐานพร้อมด้วย RPC Channel ระหว่าง ออบเจ็กต์และไคลเอนต์ของมันดังรูปที่ 13-1

เมื่อออบเจ็กต์เซตค่าบิต Deactivate-on-return ของมัน COM+ Runtime จะทำการ Deactivate ตัวออบเจ็กต์นั้น ซึ่งออบเจ็กต์นั้นจะถูกทำลาย หรือไม่ก็จะถูกนำกลับไปไว้ที่ Object Pool (ขึ้นกับค่าที่ถูกตั้งไว้ของ แอททริบิวต์ Object Pooling) แต่ตัวไคลเอนต์ก็จะยังถือครอง Proxy ไปยังตัวออบเจ็กต์อยู่ RPC และ Stub ก็ยังคงอยู่ ดังรูปที่ 13-2 ในขณะที่ตัวไคลเอนต์จะยังคงรับรู้ว่าตัวออบเจ็กต์ยังอยู่ จนกระทั่งเมื่อไคลเอนต์จะเรียกใช้เมธอดที่ของออบเจ็กต์อีกครั้งหนึ่ง ตัว Interceptor ที่อยู่ระหว่าง RPC Channel และ

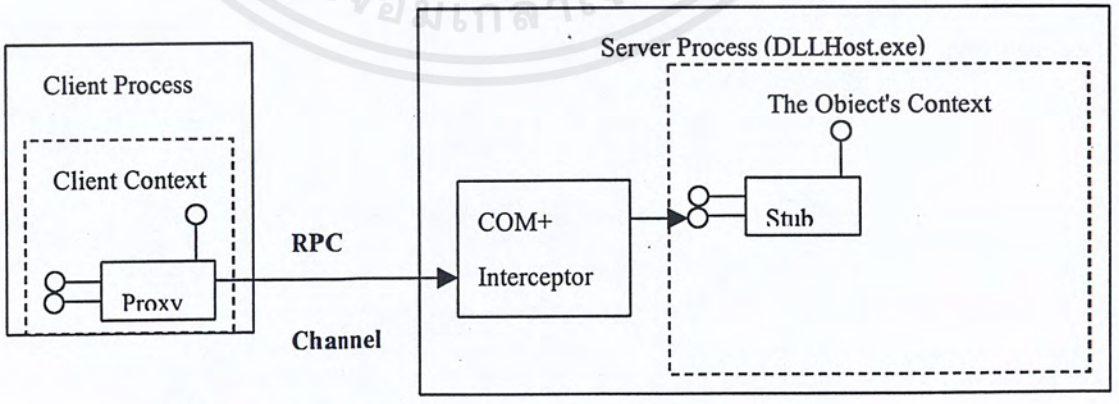
Sub จะทำการสร้าง อินสแตนซ์ใหม่อื่นของออบเจ็กต์เดิม (หรือ ดึงอินแสตนซ์นั้นมาจาก Object Pool) และนำมาเชื่อมเข้ากับ Stub และใช้งานอย่างปรกติ

นอกจากจะช่วยลดจำนวนออบเจ็กต์ที่จะต้อง Activate ในแต่ละเวลาแล้ว สำหรับออบเจ็กต์ทุกๆ ตัวที่มีลักษณะเป็นทรานแซคชั่นของ COM+ นั้นต้องใช้ JIT Activation ซึ่งจะช่วยบังคับให้ทุก ๆ ทรานแซคชั่น นั้นมีความถูกต้อง เราสามารถเซต JIT Activation ได้ในระดับคลาส โดยใช้ Component Services Explorer โดยมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเนนต์ใน โฟลเดอร์ Components ของแอปพลิเคชั่น
2. เลือก Properties... จาก เมนู Context (ไดอะลอก Component Properties จะแสดงดังรูปที่ 13-3)
3. เลือกหน้า Activation
4. เซตเช็คบ็อกซ์ Enable Just In Time Activation
5. กด OK



รูปที่ 13-1 แสดงออบเจ็กต์ที่สนับสนุน JIT Activation ในสถานะ Activated



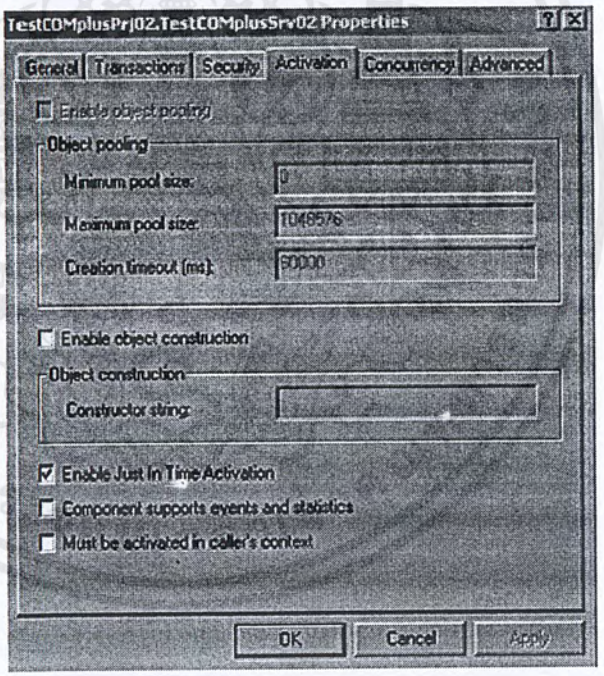
รูปที่ 13-2 แสดงออบเจ็กต์ที่สนับสนุน JIT Activation ในสถานะ Deactivated

13.2 Object Pooling

สำหรับแอปพลิเคชั่นที่ใช้ JIT Activation ในการ ActivateและDeactivate ออบเจ็กต์เป็นจำนวนมาก ๆ นั้นค่อนข้างจะเป็นวิธีที่สิ้นเปลืองที่จะต้องสร้างออบเจ็กต์ใหม่ ทุก ๆครั้งที่มีการเริ่มต้น ทรานแซคชั่น Object Pool เป็นวิธีการที่ถูกนำมาใช้ในการแก้ปัญหาดังกล่าวโดยเมื่อไคลเอนต์จะทำการ Activate ออบเจ็กต์นั้น COM+ Runtime จะส่งอินแสตนซ์จาก Object Pool มาให้ถ้าหากมีออบเจ็กต์ชนิดนั้นอยู่ใน Object Pool แต่หากไม่มีก็จะทำการสร้างอินแสตนซ์ใหม่ขึ้นมา ไคลเอนต์สามารถใช้ออบเจ็กต์ได้นานเท่าที่มันต้องการและเมื่อไคลเอนต์ปล่อยออบเจ็กต์นั้น ออบเจ็กต์ก็จะไม่ถูกทำลายแต่จะถูกนำกลับไปไว้ที่ Object Pool เพื่อรอการเรียกใช้ครั้งต่อไป ซึ่งจะมีลักษณะดังรูปที่ 13-4

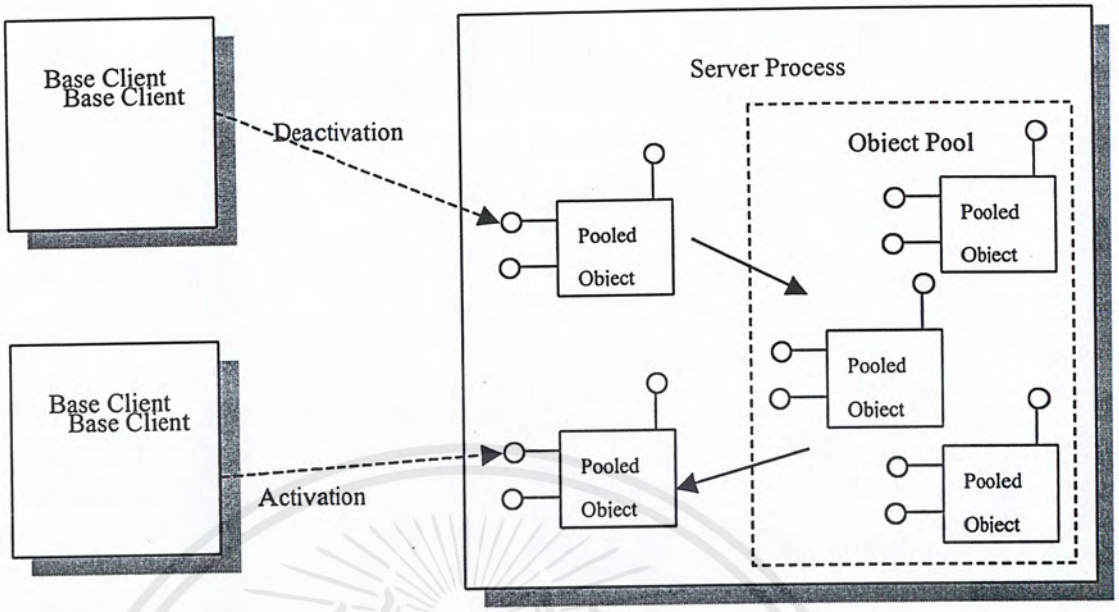
สำหรับการที่จะทำให้ออบเจ็กต์มีความสามารถที่เป็นออบเจ็กต์ที่สามารถใช้ใน Object Pool ได้นั้นก็สามารถจะเซตแอททริบิวต์ได้เหมือนกับบริการอื่น ๆ ของ COM+ Runtime โดยมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเนนต์ในโฟลเดอร์ Components ของแอปพลิเคชั่น
2. เลือก Properties... จากเมนู Context (ไดอะลอก Component Properties จะปรากฏดังรูปที่ 13-3)



รูปที่ 13-3 ไดอะลอก Component Properties

3. เลือกหน้า Activation
4. เซตเชคบ็อกซ์ Enable object pooling
5. กด OK



รูปที่ 13-4 แสดง Object Pooling

จากวิธีที่ได้กล่าวมานั้นเป็นวิธีที่จะเซตให้ออบเจ็กต์สามารถนำมาใช้ร่วมกัน (Pool) ได้แต่ก่อนที่เซตค่านั้นได้ตัวออบเจ็กต์จะต้องมีลักษณะดังนี้ด้วยคือ

- ต้องอยู่ในอพารต์เมนต์ชนิด Thread Neutral หรือ Free-Thread
- ต้องไม่มีสถานะ
- ต้องไม่มี Thread-affinity
- ต้องสามารถ รวมกันได้ (aggregateable)

นอกจากจะเซตให้ออบเจ็กต์นั้นสามารถใช้งานร่วมกันได้แล้วยังสามารถกำหนดจำนวนอินแสตนซ์ของออบเจ็กต์ที่ต้องมีอย่างน้อยที่สุด และมากที่สุดเท่าที่จะมีได้ โดยหากออบเจ็กต์ที่มีอยู่ใน Object Pool ยังน้อยกว่าจำนวนที่ต้องมีอย่างน้อยที่สุด เมื่อมีการเรียกใช้จากไคลเอนต์ตัวใหม่ COM+ Runtime ก็จะสร้างอินแสตนซ์ใหม่ให้ทันที แต่หากมีอยู่เกินกว่าจำนวนที่น้อยที่สุดแล้วแต่ถูกใช้อยู่ทุกตัว และยังมีจำนวนที่น้อยกว่าจำนวนที่มากที่สุดเท่าที่จะมีได้แล้ว ก็จะสร้างอินแสตนซ์ใหม่ให้กับไคลเอนต์นั้น แต่ถ้าเกินขอบเขตที่มากที่สุดแล้ว ไคลเอนต์จะต้องรอจนกว่าจะมีออบเจ็กต์ว่าง แล้วจึงนำเอาออบเจ็กต์นั้นไปใช้งานได้

สำหรับ COM+ ออบเจ็กต์ที่ต้องการแสดงสถานะว่าสามารถที่จะถูกนำกลับไปที่ Pool ได้ หรือให้ทำการ Activate และ Deactivate เองต้องทำการอิมพลีเมนต์อินเตอร์เฟส IObjectControl โดยมีเมธอด Activate และ Deactivate ที่ใช้ในการ Activate และ Deactivate ออบเจ็กต์ตามลำดับ และ เมธอด CanBePooled ที่ใช้ในการบอก COM+ Runtime ว่าสามารถนำกลับไปใช้ได้หรือไม่ในขณะนั้น

13.3 Synchronization

ในหัวข้อ COM+ Thread ได้กล่าวถึงคุณสมบัติ และข้อเด่นของ เธรดโมเดล ชนิด STA และ MTA โดยที่ MTA นั้นจะทำงานได้อย่างมีประสิทธิภาพในสภาพแวดล้อมที่เป็น Multi-Thread แต่ตัวออบเจ็กต์นั้นจะต้องมีลักษณะเป็น Thread Safe ถ้าหากออบเจ็กต์นั้นไม่เป็น Thread-Safe แล้วจะต้องทำงานภายใน STA

นอกจากนั้น STA ยังไม่เหมาะกับ Object Pool เนื่องจากมันเจาะจงว่ามีเพียงเธรด เธรดเดียวเท่านั้นที่จะใช้งานมันได้ดังนั้นถึงแม้ว่าจะนำออบเจ็กต์นี้ไปไว้ใน Object Pool ก็ไม่สามารถถูกเรียกใช้จากออบเจ็กต์ตัวอื่น ๆ ได้และจะต้องสร้างอินแสตนซ์ใหม่ให้กับแต่ละไคลเอนต์ทุกครั้งอยู่ดี

TNA นั้นเป็นวิธีที่ถูกนำมาใช้ในการแก้ปัญหาสำหรับทั้ง MTA และ STA เพื่อทำให้ COM+ Runtime สามารถทำงานได้อย่างมีประสิทธิภาพสูงสุดเมื่อทำงานกับ Object Pool เนื่องจากอนุญาติให้เธรดที่อยู่ในเธรดโมเดลชนิดต่าง ๆ เข้าทำงานภายในตัวมันได้ เช่นเดียวกับบริการอื่น ๆ ของ COM+ Runtime เราสามารถให้ COM+ ออบเจ็กต์ของเรานั้นเป็น Thread-Safe ได้โดยไม่ต้องใส่โค้ดเพิ่มเติม จากการใช้บริการ Activity-Based Synchronization Support

สำหรับ Thread-Safe สำหรับ COM+ ออบเจ็กต์นั้นสับซ้อนกว่าแค่การ ล็อกออบเจ็กต์หนึ่ง ๆ ไว้แล้วอนุญาติให้เธรดเพียงเธรดเดียวทำงานกับ เมธอดของเธรดได้เท่านั้น เนื่องจากออบเจ็กต์เหล่านั้นอาจจะเรียกเมธอดจากออบเจ็กต์ตัวอื่นซึ่งอยู่ต่างโพรเซส หรือต่างเครื่อง แล้วทำการแก้ไขหลังจากนั้นก็การยกเลิก ซึ่งอาจทำให้เกิดปัญหาในเรื่องทรานแซคชั่นต่าง ๆ ได้ ในการใช้ Activity-Based Synchronization Support นั้นจะช่วยป้องกันปัญหาของการทำทรานแซคชั่นหลาย ๆ ทรานแซคชั่นพร้อมกันได้ โดยใช้หลักการของ Activity

13.4 Activity

คือเธรดในทางจินตภาพซึ่งจะทำงานขึ้นกับยูสเซอร์เพียงยูสเซอร์เดียว และสามารถทำงานข้ามเครื่องได้ ยูสเซอร์แต่ละยูสเซอร์นั้นจะมี Activity ที่เป็นอิสระจากกัน ภายใน Activity นั้นก็สามารถมีได้หลาย ออบเจ็กต์ สภาพแวดล้อม COM+ รันไทม์จะจัดการรายละเอียดเหล่านี้โดยการสร้าง Activity ใหม่ขึ้นมาทุก ครั้งเมื่อยูสเซอร์ทำการสร้าง COM+ ออบเจ็กต์ขึ้นมาใหม่

เมื่อมีการสร้าง Activity ใหม่ขึ้นมานั้น COM+ รันไทม์จะทำการสร้าง STA ใหม่ขึ้นมาด้วย แต่หากจำนวน STA ที่สร้างขึ้นมาได้ถึงขีดจำกัดของระบบแล้ว COM+ รันไทม์ก็จะทำสร้าง Activity ขึ้นมาแบ่งกันใช้งานภายใน STA เดียวกัน เมื่อมีเหตุการณ์นี้ COM+ รันไทม์จะคอยจัดการ Multiplexing STA (เธรดจริง) และ การเข้าออกของ Activity (เธรดในทางจินตภาพ)

ในทางปฏิบัติเมื่อมี Activity มากกว่าหนึ่ง Activity ใช้ STA ร่วมกัน การทำงานของ ยูสเซอร์หนึ่งก็จะต้องบล๊อกการทำงานของยูสเซอร์อื่นอีกเช่นกัน แต่เราไม่จำเป็นต้องสนใจในเรื่องนั้น เนื่องจากสิ่งที่สำคัญของวิธีการนี้คือการที่ในการทำงาน ในไม่มีปัญหาในเรื่องของการ Synchronization และมีรูปแบบ Threading Model ที่เข้าใจได้ง่าย ๆ ว่าเป็นเธรดทางจินตภาพที่ COM+ รันไทม์คอยจัดการให้

อีกสิ่งที่สำคัญของการทำงานใน COM+ รันไทม์ซึ่งจัดการ Activity ให้นั่นคือการสร้างออบเจ็กต์ใหม่ เนื่องจาก หากมีการสร้าง COM+ ออบเจ็กต์ใหม่นั่นก็จะทำให้เกิดการสร้าง Activity ใหม่ซึ่งเมื่องานของยูสเซอร์ต้องเรียกใช้ออบเจ็กต์ผ่าน Activity ก็จะทำให้เปลืองทรัพยากรของระบบ และทำให้ความเร็วลดลง เนื่องจากต้องติดต่อระหว่าง Activity 2 Activity

ในทางปฏิบัตินั้นเราจะใช้คำสั่ง *CreateInstance* เพื่อสร้างออบเจ็กต์ใหม่ให้อยู่ใน Activity เดิม โดยคำสั่งนี้จะมีลักษณะการเรียกใช้งาน เป็นซอร์สโค้ดของ วิชวลเบสิก ได้ดังนี้

```
Dim ObjCtx As ObjectContext
Set ObjCtx = GetObjectContext()
Dim Object2 As CMyClass2
Set Object2 = ObjCtx.CreateInstance("MyServer.CMyClass2")
```

แต่ในกรณีที่ต้องการสร้างออบเจ็กต์ที่ไม่ใช่ COM+ ออบเจ็กต์ (ไม่ได้ติดตั้งลงบน COM+ รันไทม์ เช่น ออบเจ็กต์ ADO) ซึ่งออบเจ็กต์เหล่านั้นไม่จำเป็นต้องมีการจัดการสภาพแวดล้อมจาก COM+ รันไทม์ ซึ่งทำให้ไม่มีการสร้าง Activity ใหม่ดังนั้นจึงไม่จำเป็นต้องเรียกใช้ผ่านทางเมธอด *CreateInstance*

เราสามารถกำหนดลักษณะการสนับสนุนการ ซิงโครไนซ์ ของออบเจ็กต์ได้โดยใช้ Component Service Explorer โดยมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเนนต์ในโฟลเดอร์ Components ของแอปพลิเคชั่น
2. เลือก Properties... จากเมนู Context (ไดอะลอก Component Properties จะปรากฏดังรูปที่ 13-3)
3. เลือกหน้า Concurrency
4. เลือกปุ่มเรดิโอสำหรับค่าที่ต้องการ
5. กด OK

อย่างไรก็ดีออบเจ็กต์บางรูปแบบก็มีลักษณะการ ซิงโครไนซ์ ที่เหมาะสมและเป็นข้อบังคับดังนี้

- COM+ คอมโพเนนต์ที่ทำงานในรูปแบบทรานแซคชันต้องอยู่ใน Activity ซึ่งค่าที่จะเลือกได้จะเป็นได้แค่ Required หรือ Required New เท่านั้น
- STA คอมโพเนนต์ต้องอยู่ใน Activity เสมอไม่ว่ามันจะทำงานในรูปแบบทรานแซคชันหรือไม่ก็ตาม ค่าที่เลือกได้ก็คือ Required หรือ Required New
- ออบเจ็กต์ที่สนับสนุน JIT Activation ไม่ว่ามันจะทำงานในรูปแบบทรานแซคชันหรือไม่ก็ต้องอยู่ใน Activity เช่นกัน และค่าที่จะเลือกได้ก็จะมีแค่ Required หรือ Required New สำหรับรายละเอียดของค่าต่าง ๆ ที่จะเซตได้จะแสดงในตารางที่ 13-1

ค่าการ ซิงโครไนซ์	รายละเอียด
Not Support	ออบเจ็กต์จะไม่รันใน Activity
Supported	ออบเจ็กต์จะรันใน Activity เดียวกับผู้สร้าง ถ้าผู้สร้างมันรันใน Activity
Required	ออบเจ็กต์จะรันใน Activity เดียวกับผู้สร้าง ถ้าผู้สร้างมันรันใน Activity และจะสร้าง Activity ของมันเองหากผู้สร้างไม่ได้รันใน Activity
Required New	จะทำการสร้าง Activity ของมันเองเสมอ
Disable	ไม่สนในการทำ ซิงโครไนซ์ ทั้งหมด

ตารางที่ 13-1 ค่าที่เป็นไปได้ทั้งหมดของการ ซิงโครไนซ์

ออบเจ็กต์ที่รันอยู่ใน Activity ก็จะมี Activity ID ซึ่งสามารถดูได้โดยเมธอด GetActivityId ซึ่งอยู่ใน IObjectContextInfo ซึ่งสามารถอธิบายการทำงานของ Activity ได้ดังนี้ (สมมติให้ออบเจ็กต์มีค่าการซิงโครไนซ์ เป็น Required และผู้สร้างยังไม่ได้รันใน Activity) เมื่อออบเจ็กต์ถูกสร้างมันก็จะสร้าง Activity ของมันเองซึ่งแต่ละ Activity จะมี ล็อกซิงโครไนซ์ระหว่างโพรเซส (Process-Wide Synchronization Lock) เมื่อมีเธรดมาขอเรียกใช้ออบเจ็กต์นี้ เธรดนั้นก็จะนำล็อกตัวนี้ไปด้วย และจะไม่คืนจนกว่าจะเธรดจะออกจากเมธอดนั้น เมื่อมีเธรดอื่นมาขอเรียกใช้ออบเจ็กต์นี้ก็จะยังไม่สามารถใช้งานได้หากเธรดแรกนั้นยังไม่ออกจากเมธอดของออบเจ็กต์ เนื่องจากไม่มีล็อก

หากเธรดที่อยู่ในเมธอดของออบเจ็กต์พยายามสร้างออบเจ็กต์ใหม่ออบเจ็กต์ใหม่จะมีลักษณะที่จะอยู่กับ Activityในรูปแบบใดก็จะขึ้นกับคุณสมบัติในการซิงโครไนซ์ของมันเองซึ่งเป็นตามตารางที่ 13-2

ค่าการ ซิงโครไนซ์	สถานะ Activity
Not Supported	ไม่รันใน Activity
Support	รันใน Activity ของผู้สร้าง
Required	รันใน Activity ของผู้สร้าง
Required New	รันใน Activity ของมันเอง

ตารางที่ 13-2 สถานะของ Activity ของออบเจ็กต์ซึ่งผู้สร้างอยู่ใน Activity

หากออบเจ็กต์ตัวใหม่มีค่าในการซิงโครไนซ์เป็น Required มันจะทำการใช้ Activity ร่วมกันออบเจ็กต์แรก และเมื่อมีเธรดใหม่เรียกใช้ออบเจ็กต์ใหม่ในขณะที่ออบเจ็กต์แรกทำงานอยู่ นั้นจะไม่สามารถทำได้เนื่องจาก เธรดใหม่นี้ก็ต้องการล็อกซิงโครไนซ์ระหว่างโพรเซส ซึ่งไม่มีเนื่องจากเธรดที่ทำงานกับออบเจ็กต์แรกนั้น ใช้อยู่ ดังนั้นทั้งสองออบเจ็กต์ก็จะไม่สามารถทำงานได้อย่างต่อเนื่อง แต่ถ้า

ออบเจ็กต์มีค่าในการชิงโครไนซ์เป็น Not Support หรือ Required New ก็จะทำให้ออบเจ็กต์ใหม่นั้นสามารถทำงานพร้อมกับออบเจ็กต์แรกได้

เมื่อออบเจ็กต์ที่อยู่ภายใน Activity เดียวกัน แต่อยู่ต่าง โพรเซส หรือต่างเครื่องกัน มันจะไม่สามารถใช้เธรดร่วมกันได้แต่ COM+ Runtime จะคอยจัดการตัวล็อกชิงโครไนซ์ระหว่างโพรเซสนี้ให้

หากมีออบเจ็กต์หนึ่งเรียกเมธอดที่อยู่ในออบเจ็กต์หนึ่ง และ ออบเจ็กต์ตัวที่สองก็เรียกใช้เมธอดในออบเจ็กต์แรกเช่นกันออบเจ็กต์แรกก็จะล็อกและรอจนกว่าออบเจ็กต์ตัวที่สองจะว่าง เช่นเดียวกับออบเจ็กต์ตัวที่สองที่จะล็อก และรอจนกว่าออบเจ็กต์จะว่าง(ซึ่งจะทำให้เกิด Dead Lock) ปัญหานี้จะแก้ไขโดยใช้หลักการ Causality (ซึ่งเป็นหน้าที่ของ DCOM) Causality เป็นชื่อที่แทนกลุ่มองการเรียกใช้ฟังก์ชันที่ซ้อนกัน Causality ID (ซึ่งเป็น GUID) จะถูกส่งผ่านทาง DCOM ทุกครั้งที่มีการเรียกใช้เมธอด ออบเจ็กต์แรกจะตรวจสอบว่า Causality ID และพิจารณาว่าตรงกับ Causality ID ตรงกับที่มันรออยู่หรือไม่ ออบเจ็กต์แรกจะทำการป้องกันไม่ให้เกิด Dead Lock โดยไม่อนุญาติให้ออบเจ็กต์ตัวที่สองเรียกใช้เมธอด

บทที่ 14
ความปลอดภัยบนอินเตอร์เน็ต (Internet security)

การติดต่อสื่อสารบนอินเตอร์เน็ตนั้นจะใช้ Transmission Control Protocol/Internet Protocol (TCP/IP) โดย TCP/IP นั้นอนุญาตให้สามารถส่งข้อมูลระหว่าง คอมพิวเตอร์ ได้ ทำให้เป็นไปได้ว่าอาจมีมือที่สามเข้ามาก่อกวนระหว่างที่สื่อสารกันอยู่ก็เป็นได้ ลักษณะที่จะถูกมือที่สามเข้ามาก่อกวนมีอยู่ 3 ลักษณะ ดังนี้

1. การลอบฟัง (Eavesdropping) ข้อมูลอาจถูกผู้อื่นเข้ามาลอบฟังได้ระหว่างทางที่มีการขนส่งข้อมูล เช่น มีการลอบฟังเอาหมายเลขบัตรเครดิตของเราขณะที่ทำการซื้อสินค้าผ่านอินเตอร์เน็ต
2. การลักลอบเปลี่ยนแปลงข้อมูล (Tampering) การเข้ามาเปลี่ยนแปลงข้อมูลขณะที่ทำการส่งข้อมูลทำให้ข้อมูลบิดเบือนไปทำให้การรับข้อมูลเป็นไปอย่างผิดๆ เช่น ข้อมูลการส่งสินค้าถูกเปลี่ยนแปลงจำนวนไปทำให้เกิดความเสียหายได้
3. การปลอมตัวเข้ามา (Impersonation) การปลอมตัวเข้ามาหลอกเราว่าเป็นคนๆ นั้น หรือ เว็บไซต์นั้นทำให้เราหลงเชื่อและทำการติดต่อด้วย อาจเกิดความเสียหายได้

ดังที่ได้กล่าวมาแล้วถึงปัญหาความปลอดภัยบนอินเตอร์เน็ต จึงมีการคิดวิธีการแก้ปัญหาข้างต้น ซึ่งจะกล่าวดังต่อไปนี้

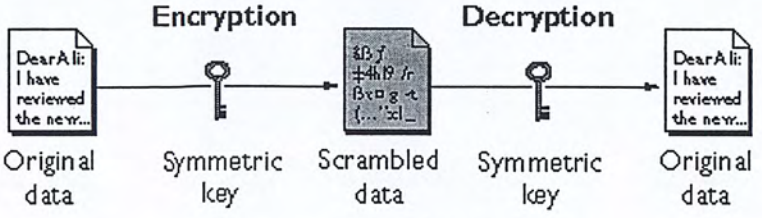
14.1 การแก้ปัญหาการลอบฟัง จะใช้การเข้ารหัส และ การถอดรหัส ข้อมูล มีรายละเอียดดังนี้

14.1.1 การเข้ารหัสข้อมูลและถอดรหัสข้อมูล (Encription and Decription)

การเข้ารหัสข้อมูลเป็นวิธีการที่ทำให้ข้อมูลที่ถูกเข้ารหัสไม่สามารถอ่านเข้าใจได้ และการถอดรหัสข้อมูลเป็นการทำให้ข้อมูลที่ถูกเข้ารหัสกลับมาเป็นข้อมูลที่สามารถอ่านเข้าใจได้ กระบวนการเข้ารหัส(Cryptographic Algorithm) มักเรียกอีกอย่าง ว่า Cipher เป็นกระบวนการทางคณิตศาสตร์ที่ใช้ทำการเข้ารหัสและถอดรหัส

ในที่นี้จะกล่าวถึงการเข้ารหัสและถอดรหัส โดยใช้ Symmetric-Key Encryption และ Public-Key Encription

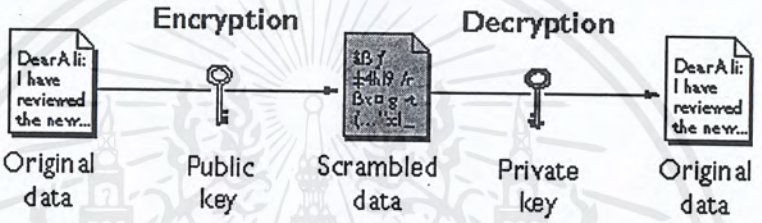
14.1.2 การเข้ารหัสแบบสมมาตร (Symmetric-Key Encryption) คือ ใช้คีย์ตัวเดียวกันทั้งการเข้ารหัสและถอดรหัส



รูปที่14-1 แสดงการเข้ารหัสแบบสมมาตร

การเข้ารหัสแบบสมมาตรมีประสิทธิภาพสูง มีความเร็วในการเข้ารหัสและถอดรหัสสูง แต่ คีย์ที่ใช้ในการเข้ารหัสและถอดรหัสต้องถูกเก็บไว้เป็นความลับ ถ้าถูกมือที่สามเอาไปได้จะทำให้ข้อมูลถูกถอดรหัสได้ การเข้ารหัสแบบสมมาตรมีบทบาทสำคัญสำหรับ SSL Protocol ที่จะกล่าวถึงในบทนี้

14.1.3 การเข้ารหัสโดยพับลิคคีย์ (Public-Key Encription) หรือ เรียกอีกอย่างว่า การเข้ารหัสแบบอสมมาตร (Asymmetric Encryption) เป็นวิธที่ที่ใช้กันมาก โดยการเข้ารหัสและถอดรหัสจะใช้คีย์ สองตัว คือ พับลิคคีย์ (Public key) และ ไพรเวท คีย์ (Private key) โดย พับลิคคีย์เป็นคีย์ที่สามารถให้ผู้อื่นรู้ได้ ส่วน ไพรเวทคีย์เป็นคีย์ที่ต้องเก็บไว้เป็นความลับห้ามให้ผู้อื่นรู้ โดยการเข้ารหัสถ้าเข้าโดยพับลิคคีย์จะต้องถอดรหัสด้วยไพรเวทคีย์ และในทางตรงกันข้าม ถ้าเข้ารหัสด้วยไพรเวทคีย์จะต้องถอดรหัสด้วยพับลิคคีย์



รูปที่ 14-2 การเข้ารหัสแบบอสมมาตร

โดยการส่งข้อมูลถ้าเราจะส่งข้อมูลให้กับใครนั้น จะต้องทำการเข้ารหัสข้อมูลด้วยพับลิคคีย์ของคนๆ นั้น แล้วทำการส่งข้อมูลไปให้ แล้วจากนั้นคนๆ นั้นจะทำการถอดรหัสด้วยไพรเวทคีย์ของตนเอง บางทีมีการใช้การเข้ารหัสแบบพับลิคคีย์ในการส่ง คีย์แบบสมมาตร ซึ่งเป็นวิธีการที่ใช้ใน SSL Protocol

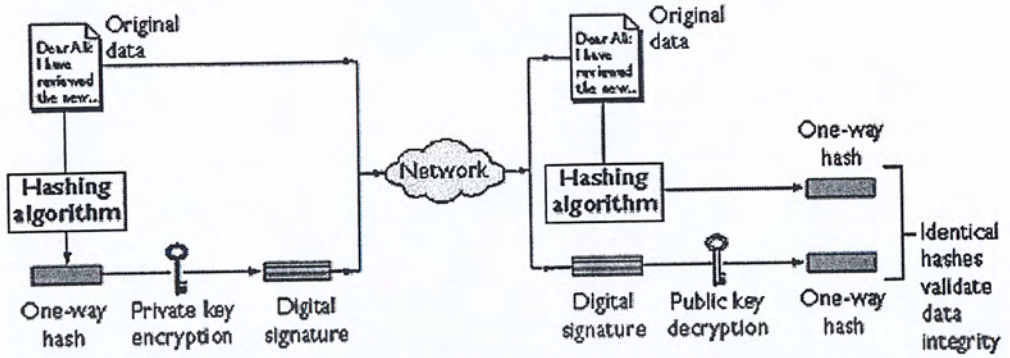
การเข้ารหัสโดยใช้ ไพรเวทคีย์นั้นไม่ควรใช้กับข้อมูลที่ต้องการความปลอดภัยสูง แต่ก็มีประโยชน์มากในการใช้เป็นลายเซ็นดิจิตอล (Digital signature)

14.2 การแก้ปัญหาการลักลอบเปลี่ยนข้อมูล ทำได้โดยการใช้ ลายเซ็นอิเล็กทรอนิก มีรายละเอียดดังนี้

14.2.1 ลายเซ็นดิจิตอล (Digital signature)

ใช้วิธีการเข้ารหัสแบบพับลิคคีย์เข้ามาช่วย โดยการตรวจสอบการเปลี่ยนแปลงข้อมูลทำได้โดยการใช้กระบวนการทางคณิตศาสตร์ เรียกว่า One-way hash หรือ Message digest ซึ่งมีคุณสมบัติสองอย่าง คือ 1. ข้อมูลที่ต่างกันเพียงอักขระตัวเดียวก็จะได้ ผลของการ Hash ไม่เหมือนกัน 2. การ Hash เมื่อทำกับข้อมูลแล้วไม่สามารถทำให้กลับมาเหมือนเดิมได้

ดังที่ได้กล่าวมาแล้วว่า เราสามารถใช้ ไพรเวทคย์ในการเข้ารหัส และถอดรหัสด้วย พับลิคคีย์ได้ แม้วิธีนี้จะไม่ดีต่อการส่งข้อมูลที่เป็นความลับ แต่กลับเป็นประโยชน์ในการทำลายเซ็นดิจิตอล



รูปที่ 14-3 แสดงการลงลายเซ็นดิจิตอล

ขั้นตอนการลงลายเซ็นนั้นจะมีการส่งข้อมูลไป สองชุด คือ ชุดที่ 1 เป็นข้อมูลต้นฉบับ ที่ไม่ผ่านกระบวนการใดๆ ทั้งสิ้น ชุดที่ 2 เป็นข้อมูลต้นฉบับที่ผ่านการทำ Hash แล้วทำการ ลงลายเซ็นด้วยไพรเวทคีย์ เสร็จแล้วทำการส่งข้อมูลทั้งสองไปพร้อมกัน เมื่อส่งมาถึงทางด้านผู้รับแล้วผู้รับจะทำการ ตรวจสอบโดยการทำสองขั้นตอนดังนี้ ขั้นที่ 1 นำข้อมูลต้นฉบับที่ได้มาทำการ Hash ขั้นที่ 2 นำลายเซ็นอิเล็กทรอนิกมาถอดรหัสด้วย พับลิคคีย์ เมื่อทำทั้งสองขั้นตอนแล้วนำข้อมูลที่ได้มาเปรียบเทียบกันถ้าเหมือนกันหมายความว่าข้อมูลไม่ถูกแก้ไขระหว่างทาง แต่ถ้าเปรียบเทียบแล้วไม่เหมือนกันหมายความว่าข้อมูลถูกแก้ไขระหว่างทาง

14.3 การแก้ปัญหาการปลอมตัวเข้ามา ทำโดยการใช้ Certificate และ Authentication มีรายละเอียดดังนี้

14.3.1 Certificate เป็นเหมือนใบรับรองที่ใช้เฉพาะบุคคล เซิร์ฟเวอร์ หรือ บริษัทต่างๆ เป็นเหมือนใบอนุญาต พาสปอร์ต หรือบัตรประจำตัวประชาชน บอกถึงลักษณะของตัวบุคคล ช่วยแก้ปัญหาการปลอมตัวเข้ามาได้

การทำงานของ Certificate ใช้ในการบ่งชี้ถึงบุคคล โดย Certificate Authorities (CAs) จะออกให้โดยบุคคลที่สามที่เชื่อถือได้ หรือ องค์กรที่มี Certificate เซิร์ฟเวอร์ ตัว Certificate ที่ออกให้มานั้นจะใช้ดูได้โดย พับลิคคีย์ที่เข้าคู่กับไพรเวทคีย์ที่เข้าระหัส Certificate เท่านั้น เพื่อเป็นการยืนยันว่าผู้ที่ถือ Certificate เป็นตัวจริง

ตัวอย่างของ Certificate

-----BEGIN CERTIFICATE-----
MIICKzCCAZSgAwIBAgIBAzANBgkqhkiG9w0BAQQFADA3MQswCQYDVQQGEwJVUzER
MA8GA1UEChMITmV0c2NhcGUxFTATBgNVBAsTDFN1cHJpeWEncyBDQTAeFw05NzEw
MTgwMTM2MjVaFw05OTEwMTgwMTM2MjVaMEgxCzAJBgNVBAYTAlVTMREwDwYDVQQK
EwhOZXRzY2FwZTENMAsGA1UECxMEUHViczEXMBUGA1UEAxMOU3VwcmI5YSBTaGV0
dHkwgZ8wDQYJKoZIhvcNAQEFBQADgY0AMIGJAoGBAMr6eZiPGfjX3uRJgEjmKiqG
7SdATYazBcABu1AVyd7chRkiQ31FbXFOGD3wNktbf6hRo6EAmM5/R1AskzZ8AW7L
iQZBcrXpc0k4du+2Q6xJu2MPm/8WKuMOnTuvzpo+SGXelmHVChEqooCwfdiZywyZ
NMmrJgaoMa2MS6pUkfQVAgMBAAGjNjA0MBEGCWCGSAGG+EIBAQQEAwIAgDAfBgNV
HSMEGDAWgBTy8gZZkBhHUfWJM1oxeuZc+zYmyTANBgkqhkiG9w0BAQQFAAOBgQBt
I6/z07Z635DfzX4XbAFpj1R1/AYwQzTSYx8GfcNAqCqCwaSDKvsuj/vwbf91o3j3
UkdGYpcd2cYRCgKi4MwqdWyLtpuHAH18hHZ5uvi00mJYw8W2wUOsY0RC/a/IDy84
hW3WWehBUqVK5SY4/zJ4oTjx7dwNMdGwbWfpRqjd1A==
-----END CERTIFICATE-----

14.3.2 Authentication เป็นการยอมรับบุคคลที่เข้ามาให้เข้ามาได้หรือไม่ มีทั้งการ Authentication ที่เซิร์ฟเวอร์ และการ Authentication ที่ไคลเอนต์ซึ่งการ Authentication ทั้งสองแบบสามารถแบ่งย่อยได้อีกตามนี้คือ การ Authentication ที่ไคลเอนต์ มีทั้ง Password Based Authentication ซึ่งเป็นการ ใช้ชื่อและรหัสผ่าน ในการบ่งชี้ตัวไคลเอนต์ และ Certificate Based Authentication เป็นการใช้ Certificate ในการ Authentication ซึ่งจะอยู่ในส่วนของ SSL Protocol ในการลงลายเซ็นดิจิตอล

14.4 SSL Protocol

Secure Sockets Layer (SSL) protocol เป็นวิธีการในการทำ Authentication ที่เซิร์ฟเวอร์ และ การทำ Authentication ทีไคลเอนต์ รวมไปถึงการเข้ารหัสระหว่างการส่งข้อมูลของเซิร์ฟเวอร์และไคลเอนต์ SSL ใช้กันอย่างกว้างขวางในอินเตอร์เน็ตโดยเฉพาะอย่างยิ่งการส่ง หมายเลข เครดิตการ์ด

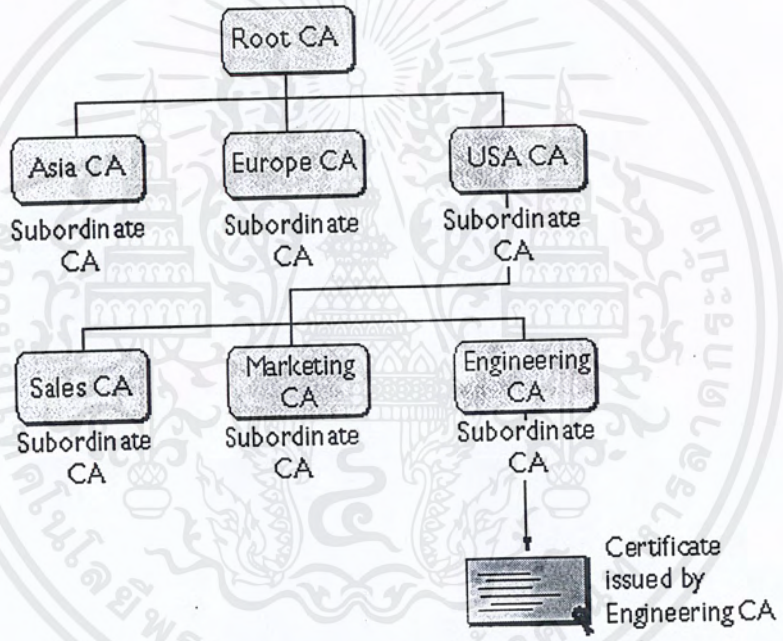
SSL ต้องการ Certificate เป็นอย่างน้อยในการทำ Handshake ขณะเริ่มต้นการส่งข้อมูล โดยเซิร์ฟเวอร์จะส่ง Certificate ให้ไคลเอนต์ในการทำ Authentication ซึ่งในกระบวนการนั้นมีทั้งการใช้ การเข้ารหัสแบบพับลิคคีย์ การใช้ลายเซ็นดิจิตอล ในการยอมรับเซิร์ฟเวอร์ หลังจากการ Authentiction เสร็จแล้ว ไคลเอนต์และเซิร์ฟเวอร์จะใช้ การเข้ารหัสแบบสมมาตร ซึ่งมีความเร็ว ในการคุยกันทั้งสองฝ่าย

ทางฝั่งเซิร์ฟเวอร์สามารถเลือกได้ว่าจะให้มีการ Authentication ที่ฝั่งไคลเอนต์ได้ด้วยหลังจากการที่ทำการ Authentication เซิร์ฟเวอร์เสร็จแล้ว จึงจะทำการ Authentication ไคลเอนต์ ก่อนที่จะเกิด SSL Connection

Certificate hierarchy และ Certificate chain เป็นการบอกถึงลำดับการเชื่อถือ (Trust) ของ Certificate

14.5 Certificate hierarchy

ในองค์กรขนาดใหญ่อาจแบ่งหน้าที่ในการออก Certificate เป็นหลายๆ ส่วน เช่น มีคนจำนวนมากที่ต้องการ Certificate เกินที่ CA เดียวจะรับได้ เป็นไปได้ว่า ตัว CA จะมอบหน้าที่ให้ลูกของมันทำแทนได้

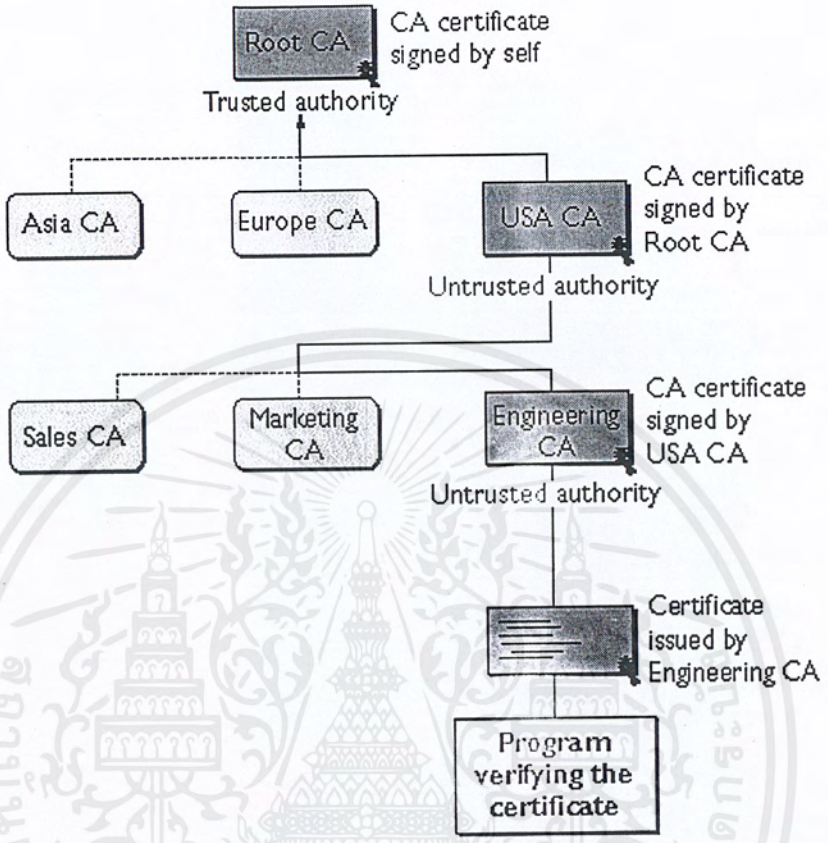


รูปที่ 14-4 แสดงลำดับขั้นของ CA

CA ที่อยู่ด้านล่างจะถูก Sign ด้วย CA ที่อยู่ด้านบน ทำให้การกระจายการออก CA เป็นไปได้ทั่วถึง โดย ตัวที่อยู่บนสุด Root CA จะ เป็น Self-signed certificate ทำ การ Sign ลูกที่อยู่ต่ำลงมา

14.6 Certificate chains

ตัว CA hierarchy มีผลต่อมายัง Certificate chains ซึ่งเป็นชุดของการออก Certificate โดย CA ให้กับลูกที่อยู่ต่ำกว่า



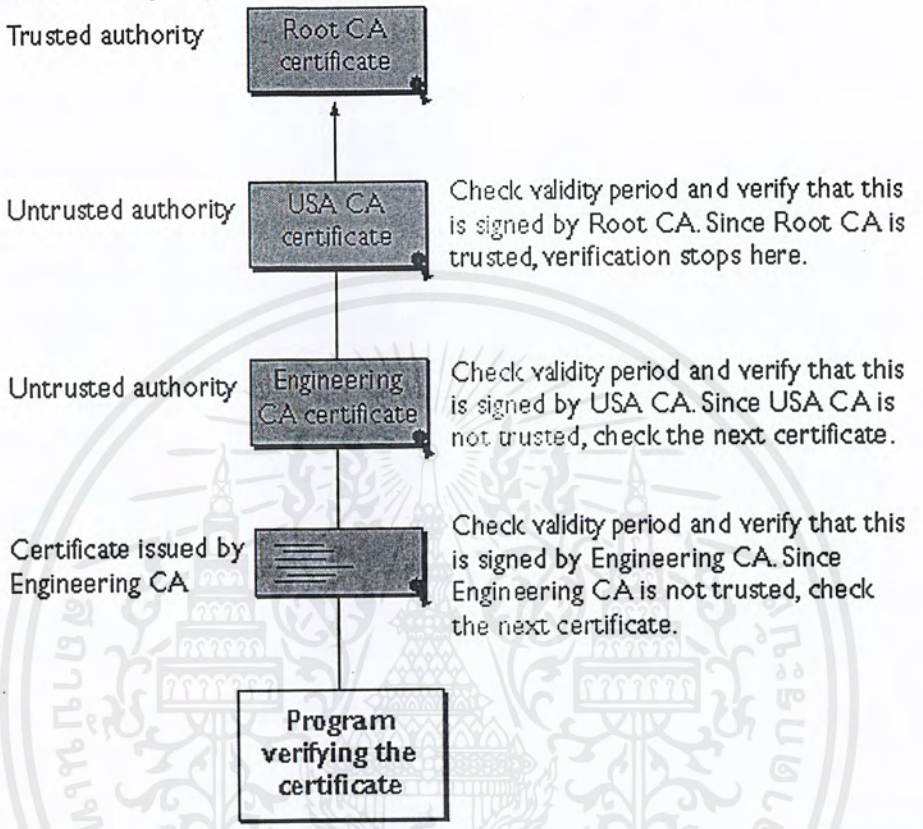
รูปที่ 14-5 แสดง Certificate chains

ใน Certificate chains มีลักษณะเป็นดังนี้

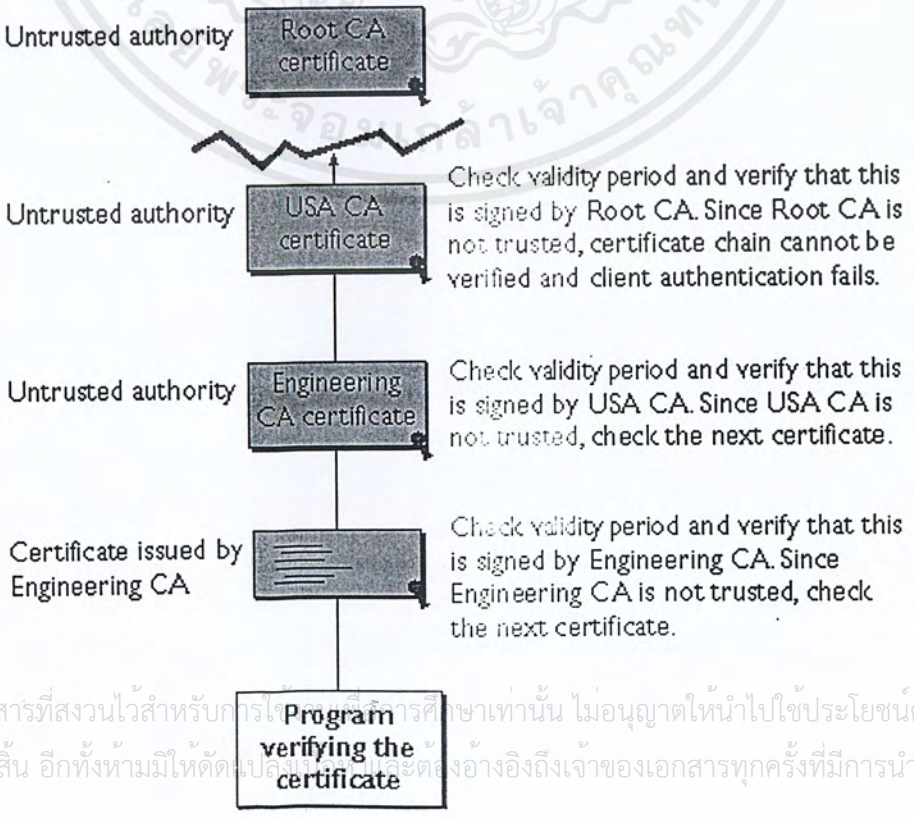
- แต่ละ Certificate จะออกมาจาก Chain ที่อยู่ด้านบน
- แต่ละ Certificate จะเก็บข้อมูล ของผู้ที่ออกให้ ตามรูป Engineering CA จะเก็บข้อมูล (DN) ของ USA CA เอาไว้ และ USA CA ก็จะเก็บข้อมูลของ CA ที่อยู่ถัดขึ้นไป
- แต่ละ Certificate จะถูก เซ็น ด้วย ไพรเวทคีย์ของผู้ที่ออกให้ และเป็นอย่างนี้ขึ้นไปเรื่อยๆ ตามรูป USA CA จะเซ็น Certificate ให้ Engineering CA

14.7 การตรวจสอบ Certificate chains

ถ้ามี Certificate Chains ต่อขึ้นไปเรื่อยๆ หลายชั้น การตรวจสอบจะตรวจสอบกับ CA ที่ออก Certificate ให้ว่าเชื่อได้หรือไม่ ถ้าไม่ก็จะ ตรวจสอบขึ้นไปเรื่อยๆ จนกว่าจะมี CA ที่ เชื่อถือได้ ถ้าหากไม่มีที่เชื่อถือได้เลย ก็จะขึ้นอยู่กับ ผู้ใช้ว่าจะเชื่อหรือไม่ แต่ถ้ามี CA ที่เชื่อได้ก็จะทำการเชื่อให้เลย

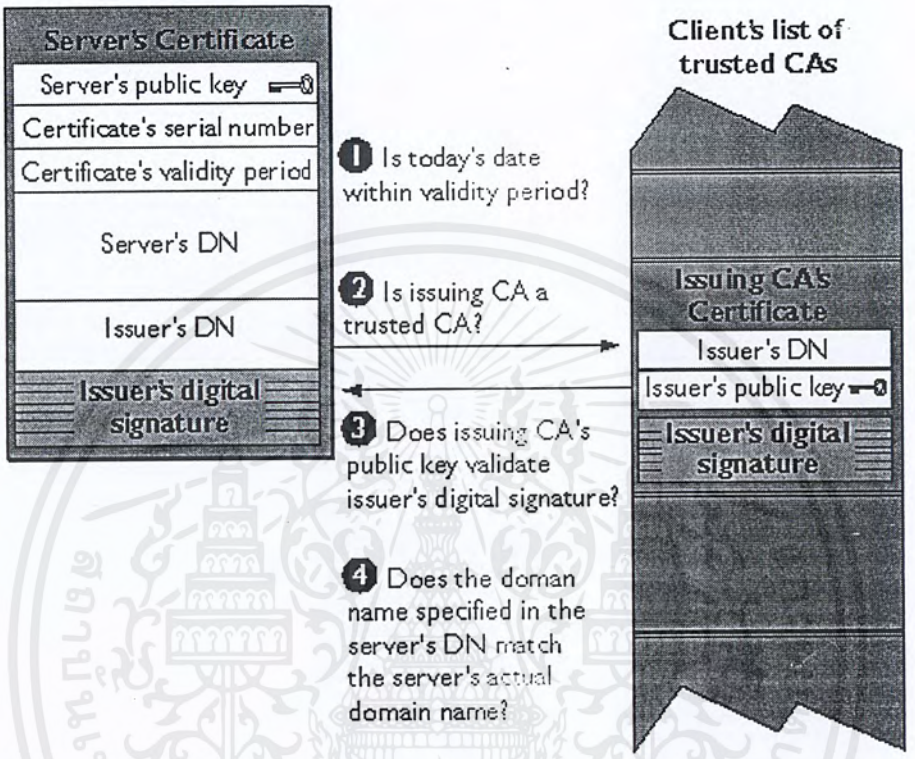


รูปที่ 14-6 แสดงการ Trust ของ Certificate chains



14.8 SSL Server authentication

เป็นการพิสูจน์ตนของ เซิร์ฟเวอร์ โดย SSL สามารถให้ไคลเอนต์ใช้การเข้ารหัสแบบพับลิคคีย์มาตรฐาน ในการตรวจสอบ Certificate ของ เซิร์ฟเวอร์ ว่าเชื่อถือได้ตามที่ CA ได้ออกให้มาหรือไม่ กระบวนการนี้สำคัญมาก เช่น การส่งหมายเลขเครดิตการ์ด



รูปที่ 14-8 แสดงการ Authentication ของเซิร์ฟเวอร์

ขั้นตอนการ ระหว่างการ Authentication เป็นดังนี้

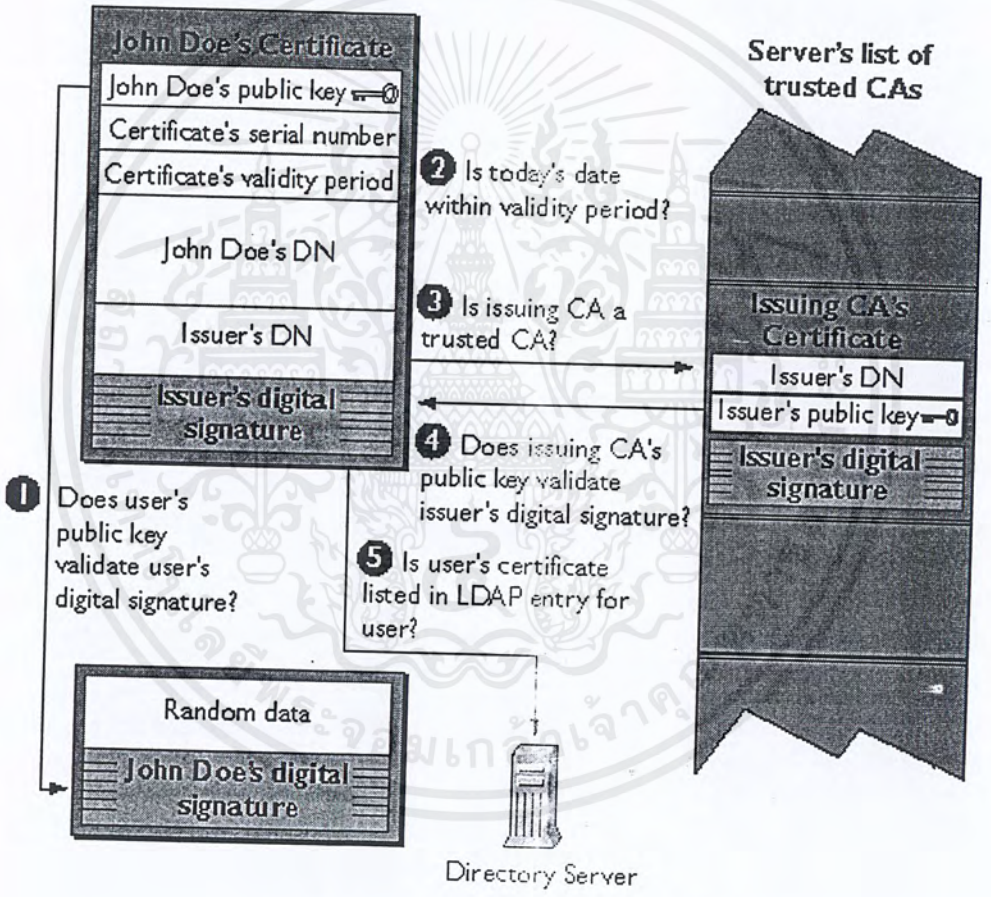
1. ไคลเอนต์ตรวจสอบ ว่า Certificate ของเซิร์ฟเวอร์นั้นเทียบกับเวลาปัจจุบันว่าหมดอายุหรือยัง ถ้าหมดอายุแล้วการ Authentication จะหยุดทันที แต่ถ้ายังไม่หมดอายุจะไปทำข้อสอง
2. ไคลเอนต์ทำการตรวจสอบว่า CA ที่ออก Certificate นั้นอยู่ในบัญชีที่เชื่อถือได้หรือไม่ถ้าอยู่ในบัญชีที่เชื่อถือได้ก็จะผ่านไปทำขั้นที่สามต่อไป แต่ถ้าไม่อยู่ในบัญชีที่เชื่อได้ก็จะทำการตรวจขึ้นไปตาม Certificate chains ตามที่ได้กล่าวไปแล้วข้างต้น
3. ไคลเอนต์ใช้พับลิคคีย์ที่มากับ Certificate ทำการตรวจสอบลายเซ็นดิจิตอลว่าตรงกันหรือไม่ ถ้าข้อมูลของ Certificate ถูกเปลี่ยนแปลงจะทำให้ ข้อมูลที่ถูกถอดรหัสโดย พับลิคคีย์จะไม่ตรงกับข้อมูลใน Certificate จะหยุดทำการ Authentication แต่ถ้าตรงกันจะไปทำขั้นต่อไป
4. ตรวจดูว่า โดเมนเนมใน Certificate ตรงกับ โดเมนเนมของเซิร์ฟเวอร์หรือไม่ ถ้าตรงกัน จึงไปทำขั้นต่อไป

5. เซิร์ฟเวอร์ผ่านการ Authentication จะใช้ Protocol SSL Handshake ซึ่งจะกล่าวต่อไปทำการสร้างการติดต่อระหว่างเซิร์ฟเวอร์ กับไคลเอนต์

14.9 Client Authentication

SSI สามารถให้มี Authentication ของไคลเอนต์ได้ โดยไคลเอนต์จะส่ง Certificate และ ลายเซ็นดิจิตอล ไปให้เซิร์ฟเวอร์ เพื่อทำการ Authentication แล้วเซิร์ฟเวอร์จะใช้ลายเซ็นดิจิตอลในการ ตรวจ Certificate

SSL Protocol ให้ไคลเอนต์ทำการสร้างลายเซ็นดิจิตอลโดยใช้ One way hash ในการ hash ข้อมูลที่สุ่มขึ้นมาระหว่างการ handshake โดยจะรู้กันเพียงสองฝ่าย คือ ไคลเอนต์กับเซิร์ฟเวอร์เท่านั้น โดยข้อมูลที่ถูก hash จะถูกเข้ารหัสด้วย ไพรเวทคีย์ของไคลเอนต์ ที่เข้ากันได้กับพับลิคคีย์ที่ส่งไปให้เซิร์ฟเวอร์



รูปที่ 14-9 แสดงการ Authentication ของไคลเอนต์

ขั้นตอนการ ระหว่างการ Authentication เป็นดังนี้

1. ตรวจสอบลายเซ็นดิจิตอล ของไคลเอนต์ ว่าถูกต้องหรือไม่
2. เซิร์ฟเวอร์ตรวจสอบ ว่า Certificate ของไคลเอนต์นั้นเทียบกับเวลาปัจจุบันว่าหมดอายุหรือยัง ถ้าหมดอายุแล้วการ Authentication จะยกเลิกทันที แต่ถ้ายังไม่หมดอายุจะไปทำข้อสาม

3. เซิร์ฟเวอร์ทำการตรวจสอบว่า CA ที่ออก Certificate นั้นอยู่ในบัญชีที่เชื่อถือได้หรือไม่ถ้าอยู่ในบัญชีที่เชื่อถือได้ก็จะผ่านไปทำขั้นที่สี่ต่อไป แต่ถ้าไม่อยู่ในบัญชีที่เชื่อได้ก็จะทำการตรวจขึ้นไปตาม Certificate chains ตามที่ได้กล่าวไปแล้วข้างต้น
4. เซิร์ฟเวอร์ใช้พับลิคคีย์ที่มากับ Certificate ทำการตรวจสอบลายเซ็นดิจิตอลของ Certificate ว่าตรงกันหรือไม่ ถ้าข้อมูลของ Certificate ถูกเปลี่ยนแปลงจะทำให้ ข้อมูลที่ถูกถอดรหัสโดย พับลิคคีย์จะไม่ตรงกับข้อมูลใน Certificate จะหยุดทำการ Authentication แต่ถ้าตรงกันจะไปทำขั้นต่อไป
5. ถ้ามีการใช้ไดเรคทอรีเซอร์วิสก็จะทำการตรวจว่าผู้ใช้คนนี้อยู่ในบัญชีหรือไม่ ถ้าไม่มีก็จะเข้าใช้เซิร์ฟเวอร์ไม่ได้
6. เซิร์ฟเวอร์ตรวจดูว่าไคลเอนต์จะใช้ทรัพยากรใด โดยดูว่ามีสิทธิใช้ได้หรือไม่ ถ้าไม่มีสิทธิก็จะเข้าใช้ไม่ได้

14.10 SSL Handshake

ใน SSL Protocol ใช้ทั้ง การเข้ารหัสทั้งสองแบบคือ การเข้ารหัสแบบพับลิคคีย์ และการเข้ารหัสแบบสมมาตร โดยการเข้ารหัสแบบสมมาตรจะทำได้เร็วกว่าการเข้ารหัสแบบพับลิคคีย์ แต่การเข้ารหัสแบบพับลิคคีย์ทำมาเพื่อการ Authentication ตัว SSL session จะเริ่มการติดต่อกันด้วยการทำ SSL Handshake ในการ handshake นั้นทำการ Authenticate ตัวเซิร์ฟเวอร์ด้วยวิธีพับลิคคีย์ และก็ให้ไคลเอนต์ติดต่อกับเซิร์ฟเวอร์ด้วยการเข้ารหัสแบบสมมาตร ขั้นตอนการทำ SSL Handshake มีดังนี้

1. ไคลเอนต์ส่ง เวอร์ชันของ SSL, Cipher setting, ข้อมูลที่สุ่มขึ้นมา และข้อมูลที่เซิร์ฟเวอร์ต้องการให้เซิร์ฟเวอร์
2. เซิร์ฟเวอร์ส่ง เวอร์ชันของ SSL, Cipher setting, ข้อมูลที่สุ่มขึ้นมา และข้อมูลที่ไคลเอนต์ต้องการให้ไคลเอนต์ โดยส่วนมากเซิร์ฟเวอร์จะส่ง Certificate มาให้ และถ้าไคลเอนต์ต้องการทรัพยากรจะต้องมีการ Authenticate ฝั่งไคลเอนต์ด้วย
3. ไคลเอนต์ใช้ข้อมูลที่ส่งมาจากเซิร์ฟเวอร์ในการ Authentication เซิร์ฟเวอร์
4. ไคลเอนต์ทำการใช้ข้อมูลที่สร้างจากการ Handshake สร้าง Premaster secrect แล้วเข้ารหัสด้วยพับลิคคีย์ของเซิร์ฟเวอร์ส่งไปยังเซิร์ฟเวอร์
5. ถ้าเซิร์ฟเวอร์ต้องการการ Authentication ของไคลเอนต์ ตัวไคลเอนต์เองต้องนำข้อมูลที่ถูกเซ็นและ Certificate ส่งมาให้เซิร์ฟเวอร์เพื่อทำการ Authenticate
6. เซิร์ฟเวอร์ทำการถอดรหัสด้วยไพรเวทคีย์แล้วนำ Premaster secrect มาสร้าง Master secret
7. ทั้ง เซิร์ฟเวอร์และไคลเอนต์ใช้ Master secret ในการสร้าง Session keys ซึ่งเป็นคีย์แบบสมมาตรจะใช้ในการคุยกันใน SSL session
8. ไคลเอนต์ส่งข้อความไปบอกเซิร์ฟเวอร์ว่า เสร็จการ Handshake แล้ว
9. เซิร์ฟเวอร์ส่งข้อความไปบอกไคลเอนต์ว่า เสร็จการ Handshake แล้ว

10. SSL Handshake เสร็จแล้วจะเกิด SSL session ขึ้นแล้วการติดต่อกันระหว่างเซิร์ฟเวอร์และไคลเอนต์จะใช้ Session key ในการเข้ารหัสและถอดรหัส

14.11 โมเดลความปลอดภัยของ COM+

จุดประสงค์ของโมเดลมี 4 ประการคือ

1. Activation Control
2. Access Control
3. Authentication Control
4. Identify Control

Activation Control เป็นการควบคุมว่าใครสามารถ launch คอมโพเนนต์ได้ Access Control เป็นการจำกัดการเข้าถึงออบเจ็กต์ของคอมโพเนนต์ ในบางกรณีผู้ใช้อาจได้รับอนุญาตและไม่อนุญาตให้เข้าถึงฟังก์ชันบางส่วนของคอมโพเนนต์ ตัวอย่างเช่น ผู้ใช้ทำการติดต่อผ่านทาง WWW ได้รับอนุญาตให้เรียกใช้บางเมธอดของคอมโพเนนต์ และมีบางส่วนของคอมโพเนนต์ที่สงวนให้ใช้ได้เฉพาะผู้ดูแลระบบ ดังนั้นจึงมี Authentication Control เพื่อจำกัดการเข้าถึงทรัพยากรจากผู้ใช้ประเภทต่าง ๆ นอกจากนั้นระดับของ Authentication Control ยังเป็นการกำหนดการเข้ารหัสข้อมูลก่อนส่งผ่านเครือข่าย Identify Control เป็นการกำหนดหลักฐานด้านความปลอดภัยของคอมโพเนนต์ที่ทำงาน ซึ่งมีรายละเอียดของยูสเซอร์แอคเคานต์หรือรายละเอียดของไคลเอนต์

การกำหนดความปลอดภัยของ COM+ คอมโพเนนต์ สามารถทำได้ 2 วิธีคือ

1. Declarative Security
2. Programmatic Security

Declarative Security เป็นการกำหนดความปลอดภัยลงในรีจิสทรีซึ่งกระทำโดยผู้ดูแลระบบ Programmatic Security เป็นการเขียนโปรแกรมเพื่อควบคุมความปลอดภัย ทำโดยผู้พัฒนาคอมโพเนนต์ โมเดลความปลอดภัยทั้ง 4 แบบสามารถใช้ Declarative Security กำหนดได้ทั้งหมด

14.11.1 Declarative Security

ก่อนหน้าที่จะมี COM+ เราสามารถกำหนดความปลอดภัยแบบ Declarative Security โดยใช้แอปพลิเคชั่น dcomcnfg ปัญหาหลักของ dcomcnfg คือค่าความปลอดภัยจะครอบคลุมทั้งหมดของ COM เซิร์ฟเวอร์, COM คลาสทั้งหมดที่อยู่ใน COM เซิร์ฟเวอร์เดียวกันมีค่าความปลอดภัยเหมือนกันหมด ซึ่งไม่เหมาะสมหาก COM เซิร์ฟเวอร์ประกอบด้วยคลาสหรืออินเตอร์เฟสที่ต้องการค่าความปลอดภัยแตกต่างกัน ตัวอย่างเช่น เรามีคอมโพเนนต์ที่ประกอบด้วย 2 อินเตอร์เฟส อินเตอร์เฟสแรกประกอบด้วยเมธอดต่าง ๆ ที่จัดเตรียมไว้ให้ผู้ใช้ทั่วไปใช้งาน ส่วนอินเตอร์เฟสที่สองเป็นของผู้ดูแลระบบเพื่อใช้ในการจัดการควบคุมคอมโพเนนต์โดยไม่อนุญาตให้ผู้ใช้ทั่วไปใช้อินเตอร์เฟสนี้ ในกรณีตัวอย่างที่กล่าวมา dcomcnfg ไม่

สามารถทำได้ วิธีเดียวที่ทำได้คือใช้ API ของ COM Security อย่างไรก็ดีในการใช้ COM+ รันไทม์ ก็ยังสามารถใช้ประโยชน์จากวิธีนี้ได้เมื่อต้องการให้ Role ของผู้ใช้เป็นเงื่อนไขสำหรับการทำงานของเมธอดในคอมโพเนนท์ เช่นให้บริการที่แตกต่างกันจากเมธอด เมธอดเดียวกัน เมื่อผู้ใช้อยู่ใน Role ที่แตกต่างกัน เป็นต้น

สำหรับ COM+ เราสามารถควบคุมการใช้เซิร์ฟเวอร์ได้ตั้งแต่ระดับคลาส อินเตอร์เฟส และเมธอดซึ่งเรียกว่า Fine-Grained Security โดยที่ไม่ต้องเขียนโปรแกรมเพิ่มเติม การใช้ Fine-Grained Security เราต้องทำความเข้าใจเกี่ยวกับโรล (Role) ซึ่งเป็นสิ่งสำคัญในการกำหนดความปลอดภัยบน COM+

14.11.2 Role

COM+ ใช้โมเดลความปลอดภัยแบบใหม่โดยแสดงอยู่ในรูปแบบของโรล โรลคือกลุ่มของผู้ใช้ COM+ แอปพลิเคชั่นที่มีค่าความปลดภัยเหมือนกัน การกำหนดโรลมีลักษณะเหมือนชีวิตจริงที่มีการกำหนดบทบาทให้บุคคลว่ามีบทบาทอะไรในสังคม สามารถทำหรือไม่อนุญาตให้ทำอะไรได้บ้าง ตัวอย่าง เช่น COM+ แอปพลิเคชั่นที่ออกแบบสำหรับธนาคารแบ่งโรลออกเป็น Teller role, Manager role และ Loan Office role ผู้ใช้ที่อยู่ในโรลเดียวกันมีค่าความปลอดภัยเหมือนกัน แต่ละโรลอนุญาตและไม่อนุญาตให้ทำอะไรเหมือนกัน เมื่อกำหนดโรลให้กับ COM+ แอปพลิเคชั่น เราสามารถเพิ่มผู้ใช้หรือกลุ่มของผู้ใช้เข้าไปในโรลได้ 2 วิธี คือใช้คอมโพเนนต์เซอร์วิสเอ็กซ์พลอเรอร์ (Declarative Security) หรือเขียนโปรแกรมโดยเรียกข้อมูงจากคอนเท็กซ์เพื่อดูว่าผู้ใช้มีโรลอะไร หากเป็นโรลที่ไม่อนุญาตก็สามารถหยุดการใช้งานหรือแจ้งความผิดพลาดที่เกิดขึ้น (Programmatic Security)

14.11.3 การสร้าง Role

เราสามารถเพิ่มโรลให้ COM+ Application โดยมีขั้นตอนดังนี้

1. คลิกขวาที่โฟลเดอร์ Roles ของแอปพลิเคชั่นที่ต้องการเพิ่มโรล
2. เลือก New > Role จะปรากฏไดอะลอก Role
3. ตั้งชื่อโรลที่ต้องการแล้วคลิก OK

เมื่อสร้างโรลแล้วต่อมาคือเพิ่มผู้ใช้เข้าไปในโรลและกำหนดส่วนต่าง ๆ ของแอปพลิเคชั่น (คอมโพเนนต์, อินเตอร์เฟส, เมธอด) ว่าต้องการให้โรลใดเรียกใช้ได้

14.11.4 การเพิ่มผู้ใช้เข้าไปในโรล

มีขั้นตอนดังนี้

1. คลิกขวาที่โฟลเดอร์ Users ที่อยู่ในไดเรกทอรีของโรลที่ต้องการเพิ่มผู้ใช้
2. เลือก New > Users จะปรากฏไดอะลอกให้เลือกผู้ใช้ (ไดอะลอกที่ปรากฏขึ้นอยู่ว่ามีใช้ Active Directory หรือไม่)
3. เลือกผู้ใช้ที่ต้องการแล้วคลิก OK

14.11.5 การกำหนดความปลอดให้แต่ละส่วนของแอปพลิเคชั่น

ก่อนอื่นเราต้องเลือกออบชั่น Access Control จึงจะใช้งานได้ โดยค่าดีฟอลต์แล้ว Access Control จะไม่ได้เลือกไว้ การเลือกออบชั่น Access Control มีขั้นตอนดังนี้

1. เปิดคอมโพเนนต์เซอร์วิสเอ็กซพลอเรอร์
2. คลิกขวาที่แอปพลิเคชั่นที่ต้องการ เลือก Properties
3. คลิกที่แท็บ Security ของไดอะลอกคอมโพเนนต์ที่ปรากฏขึ้นมา
4. เลือกเช็คบ็อกซ์ Enforce access checks for this application แล้วคลิก Apply หรือ OK

การเปลี่ยนแปลงค่าความปลอดภัยของ COM+ จะยังไม่เกิดผลทันทีจนกว่าเราจะรีสตาร์ทแอปพลิเคชันใหม่

การกำหนดความปลอดให้แต่ละส่วนของแอปพลิเคชั่นมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเนนต์ อินเตอร์เฟส หรือเมธอดที่ต้องการ เลือก Properties
2. คลิกที่แท็บ Security ของไดอะลอกที่ปรากฏขึ้นมา
3. เลือกโรลที่ต้องการที่อยู่ Role explicit set for selected item(s):
4. คลิก Appy หรือ OK

บทที่ 15
แอกทิฟเซิร์ฟเวอร์เพจและอินเตอร์เน็ตอินฟอร์เมชั่นเซอร์วิส

15.1 แนวคิดการทำงานของแอกทิฟเซิร์ฟเวอร์เพจ

แอกทิฟเซิร์ฟเวอร์เพจ (Active Server Page) หรือเอเอสพี (ASP) เป็นเทคโนโลยีการเขียนภาษาคริปต์บนฝั่งเซิร์ฟเวอร์ (Server-side Script) ทำให้เราสามารถสร้างเว็บเพจแบบไดนามิค สามารถโต้ตอบกับผู้ใช้เว็บเพจได้ ไฟล์เอเอสพีเป็นแอสกีไฟล์ที่มีนามสกุล .asp มีขั้นตอนการทำงานคือ เมื่อไอไอเอสได้รับการร้องขอไฟล์เอเอสพี มันจะประมวลผลสคริปต์ในไฟล์ที่ฝั่งเซิร์ฟเวอร์ และสร้างเอกสารเอชทีเอ็มแอลส่งกลับไปยังบราวเซอร์ที่ร้องขอ เอเอสพีรองรับภาษาสคริปต์ทั้งวิชวลเบสิกสคริปต์และจาวาสริปต์

15.2 ออบเจ็กต์พื้นฐานของเอเอสพี

เอเอสพีมีออบเจ็กต์พื้นฐานทั้งหมด 6 ชนิด ได้แก่ออบเจ็กต์รีเควส (Request Object) ออบเจ็กต์เรสพอนซ์ (Response Object) ออบเจ็กต์แอปพลิเคชั่น (Application Object) ออบเจ็กต์เสสชั่น (Session Object) ออบเจ็กต์เซิร์ฟเวอร์ (Server Object) และออบเจ็กต์เอเอสพีเออเรอร์ (ASPError Object) ซึ่งทั้ง 6 ชนิด เราสามารถเขียนสคริปต์เพื่อควบคุม และเรียกใช้งานได้ทันที

ออบเจ็กต์รีเควสและออบเจ็กต์เรสพอนซ์จะทำหน้าที่ติดต่อกับผู้ใช้งาน โดยออบเจ็กต์รีเควสทำหน้าที่รับข้อมูลจากผู้ใช้งานเข้ามา ส่วนออบเจ็กต์เรสพอนซ์ทำหน้าที่ส่งข้อมูลจากเว็บเซิร์ฟเวอร์กลับไปให้ผู้ใช้งานโดยตรง ออบเจ็กต์ที่อยู่ถัดมาเป็นออบเจ็กต์เสสชั่นซึ่งทำหน้าที่เก็บข้อมูลของผู้ใช้งานแต่ละรายที่เข้ามาใช้แอพพลิเคชันเอเอสพี โดยสร้างเสสชั่นเป็นจำนวนเท่ากับจำนวนผู้ใช้งานที่เรียกใช้สำหรับออบเจ็กต์แอปพลิเคชั่นทำหน้าที่ติดต่อ ควบคุมการทำงานของแอปพลิเคชั่นชันเอเอสพี และ ออบเจ็กต์เซิร์ฟเวอร์ทำหน้าที่ติดต่อระหว่างแอพพลิเคชันเอเอสพีกับเว็บเซิร์ฟเวอร์

ออบเจ็กต์เอเอสพีเออเรอร์ทำหน้าที่เก็บรายละเอียดการทำงานที่ผิดพลาด โดยเราจะตรวจสอบรายละเอียดความผิดพลาดของแอปพลิเคชั่นเอเอสพีที่เกิดขึ้น ได้จากรายละเอียดต่างๆ ที่เก็บอยู่ภายใน แล้วแก้ไขให้การทำงานต่างๆเป็นไปอย่างถูกต้อง

รายละเอียดของออบเจ็กต์พื้นฐานต่าง ๆ มีดังนี้

ออบเจ็กต์	คำอธิบาย
Request	ออบเจ็กต์ที่ใช้จัดการข้อมูลที่ได้รับเข้ามาจากผู้ใช้งาน
Response	ออบเจ็กต์ที่ใช้จัดการข้อมูลที่ส่งกลับไปยังผู้ใช้งาน
Session	ออบเจ็กต์ที่เก็บรายละเอียดของผู้ใช้งานแอพพลิเคชันเอเอสพีแต่ละคน
Application	ออบเจ็กต์ที่ใช้แลกเปลี่ยนข้อมูลกับแอพพลิเคชันเอเอสพี
Server	ออบเจ็กต์ที่ทำหน้าที่จัดการและบริหารทรัพยากรของเว็บเซิร์ฟเวอร์
ASPError	ออบเจ็กต์ที่ทำหน้าที่จัดการเมื่อเกิดความผิดพลาดจากการทำงานของแอพพลิเคชันเอเอสพี

ตารางที่ 15-1 ออบเจ็กต์พื้นฐานของเอเอสพี

15.2.1 ออบเจ็กต์รีเควส

ออบเจ็กต์รีเควสทำหน้าที่รับเอาข้อมูลจากบราวเซอร์ที่เรียกเข้ามา แล้วส่งข้อมูลไปให้กับออบเจ็กต์อื่นที่เกี่ยวข้อง ซึ่งข้อมูลที่ได้รับเข้ามาจากผู้ใช้งานอาจเป็นการร้องขอธรรมดาหรืออาจเป็นชุดข้อมูลที่ส่งผ่านฟอร์มก็ได้

ในบางครั้งข้อมูลที่ได้รับเข้ามาอาจมีการเข้ารหัสด้วยการใช้งานร่วมกับโปรโตคอลเอสเอสแอล (Secure Sockets Layers) เพื่อความปลอดภัยได้ด้วย

คุณสมบัติของออบเจ็กต์รีเควส

TotalBytes แสดงจำนวนไบต์ของข้อมูลที่อ่านเข้ามา

เมธอดของออบเจ็กต์รีเควส

BinaryRead เป็นการอ่านข้อมูลที่ผู้ใช้งานส่งเข้ามาด้วยวิธีโพสต์ (Post) ซึ่งการส่งด้วยวิธีการโพสต์จะเหมาะในการส่งข้อมูลขนาดใหญ่ และข้อมูลที่ไม่ใช่ตัวอักษร เช่น ภาพกราฟิก ไฟล์เสียง วิดีโอ เป็นต้น

คอลเล็คชันของออบเจ็กต์รีเควส

Form เป็นคอลเล็คชันที่รับเอาข้อมูลที่ผู้ใช้งานกรอกเข้ามาทางฟอร์ม ข้อมูลที่ป้อนเข้ามาภายใต้แท็ก < FORM >

QueryString เป็นคอลเล็คชันที่รับข้อมูลที่ส่งมาโดยแปะท้ายมากับยูอาร์แอล (URL)

Cookies เป็นคอลเล็คชั่นที่รับข้อมูลในส่วนที่เป็นคุกกี้ (Cookie) ซึ่งเก็บไว้ในเครื่องคอมพิวเตอร์ของผู้ใช้งานนำมาประมวลผล

ClientCertificate เป็นคอลเล็คชันที่เก็บรายละเอียดเกี่ยวกับสิทธิในการใช้งาน โดยใช้กับแอปพลิเคชั่นที่ต้องการความมั่นใจเกี่ยวกับความปลอดภัยของข้อมูล

ServerVariables เป็นคอลเล็คชันที่เก็บค่าตัวแปรของเซิร์ฟเวอร์ โดยเก็บรายละเอียดด้านเทคนิคและสภาพการทำงานของเว็บเซิร์ฟเวอร์เอาไว้ซึ่งสามารถนำมาประกอบการใช้งานในแอพพลิเคชันเอเอสพีได้ ตัวอย่างของ ServerVariables ที่สำคัญ ๆ เช่น

ตัวแปร	คำอธิบาย
REQUEST_PROTOCOL	ชื่อของโพรโตคอล และเวอร์ชันที่รองรับรีเควสที่เข้ามาเช่น HTTP/1.0
REQUEST_METHOD	วิธีการส่งข้อมูลไปให้กับเซิร์ฟเวอร์ (เช่นวิธีเกทหรือวิธีโพสต์)
EXECUTABLE_PATH	เป็นพาธที่เราใช้เก็บโปรแกรมที่จะประมวลผล (เป็นพาธเสมือน (Virtual Path) เซิร์ฟเวอร์ต้องนำมาแปลงเป็นพาธจริง (Physical Path))
QUERY_STRING	เป็นข้อมูลขนาดเล็กที่ส่งไปพ่วงกับยูอาร์แอลเพื่อใช้เป็นอินพุตให้แอปพลิเคชั่นเอเอสพี
SERVER_SOFTWARE	ชื่อของซอฟต์แวร์ที่เซิร์ฟเวอร์รันอยู่ (ขึ้นอยู่กับโพรโตคอล) เช่น ชื่อของเว็บเซิร์ฟเวอร์
SERVER_NAME	ชื่อของเซิร์ฟเวอร์หรือไอพีแอดเดรส

ตารางที่ 15-2 ตัวอย่างของ ServerVariables

SERVER_PORT	หมายเลขพอร์ตที่ใช้ (ปกติจะเป็นพอร์ต 80)
SERVER_ADMIN	อีเมล์แอดเดรสของผู้ที่ดูแลเซิร์ฟเวอร์
REMOTE_HOST	เป็นชื่อคอมพิวเตอร์ของไคลเอนต์หรือโดเมนเนมของไคลเอนต์
CONTENT_TYPE	ประเภทข้อมูลที่ส่งไปยังเซิร์ฟเวอร์ด้วยเมธอดโพสต์ จะเป็นชนิดข้อมูลที่แบ่งไว้ใน MINE (Multipurpose Internet Mail Extensions)
CONTENT_LENGTH	ขนาดของข้อมูลหน่วยไบต์ ที่ถูกส่งไปยังเซิร์ฟเวอร์ด้วยเมธอดโพสต์

ตารางที่ 15-2(ต่อ) ตัวอย่างของ ServerVariables

15.2.2 ออบเจ็กต์เรสพอนซ์

ออบเจ็กต์เรสพอนซ์ทำหน้าที่ส่งผลการทำงานที่ได้กลับไปยังผู้ใช้งานที่ได้ร้องขอมา ซึ่งจะส่งออกไปในรูปของเอกสารเอชทีเอ็มแอล ข้อความธรรมดา หรือคุกกี้

คุณสมบัติของออบเจ็กต์เรสพอนซ์

Buffer เป็นการเลือกว่าจะกักข้อมูลไว้ ก่อนจะส่งไปให้กับไคลเอนต์หรือไม่ ถ้ากักไว้คือจะรอให้ข้อมูลครบถ้วนทั้งหมดแล้วส่งไปทีเดียว ซึ่งการใช้บัฟเฟอร์จะมีผลตั้งแต่บรรทัดแรกที่เรากำหนดว่าให้มีการใช้บัฟเฟอร์ (ถ้าต้องการใช้เรามักกำหนดในบรรทัดแรก)

CharSet เป็นการกำหนดรูปแบบการแสดงผลประเภทข้อความที่แสดงที่บราวเซอร์ ซึ่งจะมีการตีความแตกต่างกันเมื่อใช้งานกันคนละภาษา

ContentType เป็นชนิดของข้อมูลที่ส่งกลับไป ซึ่งเราจะแยกความแตกต่างด้วยชนิดข้อมูลที่แบ่งไว้ใน MINE (Multipurpose Internet Mail Extensions)

Expires เป็นการกำหนดเวลาในหน่วยวินาที ที่เว็บเพจหมดอายุจากแคชของบราวเซอร์ เพราะว่าทุกครั้งที่เรียกแอปพลิเคชั่นชันเอเอสพี มันต้องถูกประมวลผลทุกครั้ง ซึ่งบางครั้งการเรียกใช้ในระยะเวลาใกล้เคียงกันอาจไม่จำเป็นต้องประมวลผลให้เสียเวลาก็ได้ เพราะฉะนั้นจึงมีการกำหนดเวลาที่ข้อมูลพอจะมีความถูกต้องเอาไว้ อีกเหตุผลหนึ่งก็เพื่อลดภาระของเว็บเซิร์ฟเวอร์

ExpiresAbsolute เป็นการระบุวันเวลาที่ชัดเจนที่เว็บเพจหมดอายุในแคชของบราวเซอร์

IsClientConnected เป็นการเพิ่มความสามารถของออบเจ็กต์เสสชั่น โดยจะถามว่าบราวเซอร์ยังเชื่อมต่อกับเซิร์ฟเวอร์หรือไม่ เราจะใช้ตรวจสอบว่าไคลเอนท์ที่ร้องขอข้อมูลจากเว็บเซิร์ฟเวอร์นั้นยังเชื่อมต่ออยู่หรือไม่ ถ้าขาดการเชื่อมต่อไปก็สามารถยกเลิกการส่งข้อมูลได้

Status เป็นสถานะที่เป็นผลลัพธ์ของการตอบสนองของเว็บเซิร์ฟเวอร์ ต่อการร้องขอของไคลเอนต์ ซึ่งมักถูกนำไปใช้บ่อย ๆ ในการบังคับให้ผู้ใช้งานป้อนรหัสผ่านก่อนเข้าใช้เว็บเพจ

เมธอดของออบเจ็กต์เรสพอนซ์

Write	เป็นการเขียนข้อความไปยังไคลเอนต์
BinaryWrite	เป็นการเขียนข้อมูลที่ไม่ใช่ข้อความ เช่น รูปภาพ มีเดียไฟล์ ไปยังไคลเอนท์
Clear	เป็นการล้างข้อมูลที่อยู่ในบัฟเฟอร์ออกไปให้หมด
End	เป็นการสั่งให้นำข้อมูลในบัฟเฟอร์ส่งไปให้กับไคลเอนต์ (ซึ่งเต็มบัฟเฟอร์แล้ว)
Flush	เป็นการสั่งให้นำข้อมูลในที่มีอยู่ในขณะนั้นส่งไปให้ไคลเอนท์ทันที
Redirect	เป็นการสั่งให้บราวเซอร์ย้ายไปอ่านหรือส่งต่อการทำงานไปยังเว็บเพจอื่นที่ได้ระบุไว้
AddHeader	เป็นการเขียนสตริงไปที่เอชทีทีพีเฮดเดอร์ (HTTP Header) ซึ่งเป็นส่วนที่เราไม่ใช่แสดงผล แต่ใช้เพื่อส่งข้อมูลพิเศษอื่นๆ เพื่อใช้ควบคุมการทำงานของแอปพลิเคชั่น
AppendToLog	เป็นการเขียนข้อมูลลงไปที่ล็อกของเซิร์ฟเวอร์

คอลเล็คชันของออบเจ็กต์เรสพอนซ์

ออบเจ็กต์เรสพอนซ์และออบเจ็กต์รีเควสมีคอลเล็คชันที่สำคัญคือคุกกี้ คุกกี้คือเท็กซ์ไฟล์ขนาดเล็กที่เว็บเซิร์ฟเวอร์ส่งเข้าไปเก็บในคอมพิวเตอร์ของผู้ใช้งานทั่วไป โดยจะเป็นข้อมูลให้กับเว็บเซิร์ฟเวอร์เรียกมาใช้งานเมื่อผู้ใช้งานเข้ามาใช้งานอีกครั้ง

เราอาจเห็นการใช้งานคุกกี้ในเว็บไซต์ที่มีการให้บริการสมาชิก ซึ่งการตรวจสอบการเป็นสมาชิกอาจทำได้ด้วยการใช้คุกกี้หรือการที่เก็บข้อมูลของผู้เข้าเยี่ยมชมเอาไว้ในคุกกี้ เมื่อเขากลับมาเยี่ยมซ้ำอีกครั้ง เราสามารถนำเสนอเนื้อหาที่สัมพันธ์กับการเยี่ยมชมคราวก่อนโดยอ่านจากคุกกี้ที่มีอยู่ก็ได้

สำหรับการเก็บข้อมูลในคอลเล็คชันคุกกี้มีรูปแบบดังนี้

```
Response.Cookies(ชื่อคุกกี้)(คีย์ที่ใช้).แอตทริบิวต์ที่ใช้ = ค่าที่เก็บ
```

ชื่อคุกกี้เราจะกำหนดเป็นอะไรก็ได้ ส่วนคีย์ที่ใช้เป็นชื่อข้อมูลที่เก็บ ส่วนค่าที่เก็บจะสัมพันธ์กับชื่อของคีย์ สำหรับแอตทริบิวต์ (Attribute) ซึ่งจะถูกใช้ในการควบคุมการทำงานของคุกกี้ได้แก่

Expires	เป็นการกำหนดวันที่คุกกี้หมดอายุ
Domain	เป็นการจำกัดว่าคุกกี้มาจากเว็บเซิร์ฟเวอร์ตัวใด (เพราะบางทีอาจตั้งชื่อคุกกี้เหมือนกันก็ได้)
Path	เป็นชื่อพาธในเว็บเซิร์ฟเวอร์จะถูกส่งไปในช่องทางที่ปลอดภัยหรือไม่ ถ้าเป็นข้อมูลที่สำคัญ เช่น ข้อมูลเกี่ยวกับการเงินอาจจำเป็นต้องใช้
HasKeys	เป็นข้อมูลที่ใช้บอกถึงแอตทริบิวต์ตัวอื่นๆในที่เก็บไว้ในคุกกี้

15.2.3 ออบเจ็กต์เสสชั่น

ออบเจ็กต์เสสชั่นทำหน้าที่ดูแลและเก็บข้อมูลของแต่ละไคลเอนท์ที่ส่งการร้องขอเข้ามา โดยกำหนดหมายเลขของผู้ใช้บริการ (SessionID) ซึ่งจะทำให้เว็บเซิร์ฟเวอร์สามารถแยกแยะแต่ละไคลเอนต์เพื่อให้บริการได้อย่างเหมาะสม

คุณสมบัติของออบเจ็กต์เสสชั่น

SessionID เป็นหมายเลขที่ออบเจ็กต์เสสชั่นกำหนดให้กับไคลเอนต์ ที่เข้าใช้แอปพลิเคชั่นเอเอสพี

TimeOut เป็นเวลาที่เว็บเซิร์ฟเวอร์จะตัดการติดต่อกับไคลเอนต์ หากไม่สามารถติดต่อกับไคลเอนต์ได้ หรือไคลเอนต์ไม่ตอบสนองตามเวลาที่กำหนดไว้ (มีหน่วยเป็นนาที ค่าดีฟอลต์คือ 20 นาที)

LCID LCID หรือโลคอลไอเดนทิไฟเออร์ (Locate Identifier) เป็นข้อมูลในลักษณะตัวย่อ ที่บอกให้บราวเซอร์ทราบว่าบราวเซอร์ควรแสดงผลให้เข้ากับท้องถิ่น เช่น เว็บเซิร์ฟเวอร์นั้นต้องการแสดงหน่วยเงินตราของอังกฤษ ก็จะกำหนด LCID เป็นเลขฐานสิบหก คือ 0809 หรือถ้าเป็นเมืองไทย ต้องกำหนดเป็น 041E เป็นต้น (ดูข้อมูลเพิ่มเติมได้จากเว็บไซต์ www.w3c.org)

เมธอดของออบเจ็กต์เสสชั่น

Abandon เป็นการสั่งให้จบการทำงานของไคลเอนต์

คอลเล็คชันของออบเจ็กต์เสสชั่น

นอกจากจะอ่านและกำหนดค่าจากคุณสมบัติต่างๆของออบเจ็กต์เสสชั่น แล้วเรายังสามารถเก็บข้อมูลต่างๆของแต่ละเสสชั่น ให้เหมาะกับลักษณะของแอปพลิเคชั่นได้ เช่น เราสร้างเว็บไซต์ขายสินค้า เราอาจต้องการเก็บข้อมูลของแต่ละไคลเอนท์ที่เข้ามาเป็นชื่อลูกค้า หรือสินค้าที่ได้เลือกซื้อเอาไว้ เป็นต้น

เราสามารถเก็บข้อมูลข้างต้นของแต่ละเสสชั่น ด้วยตัวแปรที่สร้างขึ้นมาพิเศษ แล้วเก็บตัวแปรทั้งหมดไว้ในคอลเล็กชันคอนเท็นต์ส (Contents)

ตัวแปรที่อยู่ในแต่ละเสสชั่นจะมีอายุจนกว่าผู้ใช้จะสิ้นสุดการทำงานแอพพลิเคชัน ซึ่งก็ทำให้ SessionID นั้นหมดอายุไป

อีเว็นต์ของออบเจ็กต์เสสชั่น

Session_OnStart เป็นเหตุการณ์ที่เกิดขึ้นเมื่อไคลเอนต์เริ่มใช้งานแอปพลิเคชั่นเอเอสพี

Session_OnEnd เป็นเหตุการณ์ที่เกิดขึ้นเมื่อไคลเอนต์สิ้นสุดการทำงานแอปพลิเคชั่นเอเอสพี

15.2.4 ออบเจ็กต์แอปพลิเคชั่น

ออบเจ็กต์แอปพลิเคชั่นทำหน้าที่จัดการติดต่อเกี่ยวกับแอพพลิเคชันที่ทำงานอยู่บนเซิร์ฟเวอร์ ซึ่งทำให้เราสามารถใช้ข้อมูลร่วมกันระหว่างแอพพลิเคชันเอเอสพีกับข้อมูลของไคลเอนต์ที่เรียกใช้งาน

ในการใช้ข้อมูลร่วมกันระหว่างแอปพลิเคชั่นเอเอสพีกับไคลเอนต์ เราจะสร้างตัวแปรขึ้นมา ซึ่งตัวแปรที่สร้างขึ้นมานั้นจะเก็บข้อมูลไปจนกว่าแอปพลิเคชั่นเอเอสพีนั้นจะหยุดทำงานหรือปิดเซิร์ฟเวอร์

เมธอดของออบเจ็กต์แอปพลิเคชั่น

Lock เป็นเมธอดที่ใช้ป้องกันไม่ให้ไคลเอนต์เปลี่ยนแปลงข้อมูลใดๆ ที่เก็บไว้ในออบเจ็กต์แอปพลิเคชั่น ซึ่งมักจะเป็นช่วงที่เราต้องการอัพเดตข้อมูลบางอย่างในออบเจ็กต์แอปพลิเคชั่น

UnLock จะทำหน้าที่ตรงกันข้ามกับเมธอด Lock ซึ่งจะยอมให้ออบเจ็กต์แอปพลิเคชั่นสามารถถูกเปลี่ยนแปลงข้อมูลได้จากไคลเอนต์ที่ใช้งาน

คอลเล็กชันของออบเจ็กต์แอปพลิเคชั่น

ออบเจ็กต์แอปพลิเคชั่นมีคอลเล็กชันคอนเท็นต์ส ซึ่งมีแนวคิดการทำงานเหมือนกับออบเจ็กต์เสสชั่น เพียงแต่จะมีขอบเขตการทำงานทั้งแอปพลิเคชั่น คือ ตัวแปรต่างๆที่อยู่ในคอลเล็คชันคอนเท็นต์สของออบเจ็กต์แอปพลิเคชั่นจะมีอายุจนกว่าแอพพลิเคชั่นจะสิ้นสุดการทำงาน (มักจะเป็นเวลาที่ปิดเว็บเซิร์ฟเวอร์) หรือมีการคอมไพล์ไฟล์ Global.asa ใหม่

อีเว็นต์ของออบเจ็กต์แอปพลิเคชั่น

ทั้งสองอีเวนต์ของออบเจ็กต์แอปพลิเคชั่นจะเป็นเหตุการณ์ที่เกิดขึ้นเมื่อแอปพลิเคชั่นเริ่ม และสิ้นสุดการใช้งานซึ่งเรามักจะเพิ่มเติมโค้ดเพื่อจัดการแอปพลิเคชั่นเอาไว้ทั้งสองอีเว็นต์ ได้แก่

Application_OnStart เป็นเหตุการณ์ที่เกิดขึ้นเมื่อแอพพลิเคชันเอเสพี เริ่มทำงานบนเซิร์ฟเวอร์ เราจะใช้อีเว็นต์นี้ในการกำหนดค่าเริ่มต้นให้กับตัวแปร คุณสมบัติและคอลเล็คชันต่าง ๆ ที่ต้องใช้งานกันในแอปพลิเคชั่น

Application_OnEnd เป็นเหตุการณ์ที่เกิดขึ้นเมื่อแอพพลิเคชันเอเอสพี จบการทำงาน

15.2.5 ออบเจ็กต์เซิร์ฟเวอร์

ออบเจ็กต์เซิร์ฟเวอร์ทำหน้าที่ติดต่อ และจัดการกับตัวเซิร์ฟเวอร์ที่ทำงานอยู่

คุณสมบัติของออบเจ็กต์เซิร์ฟเวอร์

ScriptTimeOut เป็นเวลาสูงสุดที่ยอมให้แอปพลิเคชั่นเอเอสพีประมวลผล (ปกติกำหนดไว้ 90 วินาที) เพราะถ้าหากใช้เวลามากกว่านี้ จะทำให้การตอบสนองผู้ใช้อย่างไม่มีประสิทธิภาพ เป็นการใช้ทรัพยากรของระบบมากเกินไปแล้ว ดังนั้นถ้าเกินเวลาที่กำหนดให้ก็ให้จบการทำงาน และรายงานข้อผิดพลาดให้ทราบ

เมธอดของออบเจ็กต์เซิร์ฟเวอร์

MapPath("url") เป็นเมธอดที่ใช้ขอพาธจริงพร้อมชื่อไฟล์ โดยเราจะต้องระบุพารามิเตอร์เป็นยูอาร์แอล

Execute("url") เป็นการหยุดการทำงานในเว็บเพจปัจจุบัน แล้วย้ายการทำงานไปยังเว็บเพจอื่นที่ได้ระบุไว้ในพารามิเตอร์ยูอาร์แอล

GetLastError เป็นการอ้างอิงถึงออบเจ็กต์เอเอสพีเออเรอร์ ซึ่งจะเป็นการขอรายละเอียดของข้อผิดพลาดครั้งล่าสุดนำไปใช้งาน

CreateObject	เป็นเมธอดที่สำคัญมาก ซึ่งทำหน้าที่สร้างอินสแตนซ์ของออบเจ็กต์เอเอสพีขึ้นมาใช้งาน
HTMLEncode	เป็นการเข้ารหัสเอกสารเอชทีเอ็มแอล ทำให้เราสามารถนำเอาแท็กของเอชทีเอ็มแอลไปแสดงผลที่บราวเซอร์ได้อย่างถูกต้อง
URLEncode	เป็นการเข้ารหัสข้อความที่ต้องการให้เป็นคิวรีสตริง (QueryString) ซึ่งไปต่อท้ายยูอาร์แอล

15.2.6 ออบเจ็กต์เอเอสพีเออเรอร์

ออบเจ็กต์เอเอสพีเออเรอร์ เป็นออบเจ็กต์ตัวใหม่ที่พบในเอเอสพี 3.0 ซึ่งเริ่มทำงานเมื่อเรียกใช้งานเมธอด GetLastError ของออบเจ็กต์เซิร์ฟเวอร์

เมธอดของออบเจ็กต์เอเอสพีเออร์

ASPCode	เป็นหมายเลขของข้อผิดพลาด (สร้างโดยเว็บเซิร์ฟเวอร์)
ASPDescription	เป็นคำอธิบายของหมายเลขข้อผิดพลาดนั้น
File	เป็นชื่อของไฟล์ที่เป็นตัวกำเนิดของข้อผิดพลาด
Line	เป็นเลขที่บรรทัดที่อยู่ในไฟล์ที่เป็นต้นกำเนิดของข้อผิดพลาด
Source	เป็นส่วนของซอร์สโค้ด ที่ทำให้เกิดข้อผิดพลาด
Category	เป็นประเภทของความผิดพลาดว่ามาจากเอเอสพีเอง มาจากภาษาสคริปต์ หรือมาจากตัวออบเจ็กต์
Description	เป็นคำอธิบายสั้น ๆ ของความผิดพลาดนั้น

นอกเหนือจากชุดออบเจ็กต์พื้นฐานทั้ง 6 ชนิดแล้ว บางครั้งเราอาจต้องการออบเจ็กต์ที่มีคุณสมบัติพิเศษเพื่อเพิ่มเติมให้กับแอพพลิเคชัน หรือ ออบเจ็กต์ที่ใช้จัดการกับเว็บเซิร์ฟเวอร์ เราก็สามารถทำได้ โดยมี 3 ทางเลือก คือ

- ออบเจ็กต์แอกทิฟเซิร์ฟเวอร์ (Active Server Object) เป็นชุดออบเจ็กต์ที่เว็บเซิร์ฟเวอร์เตรียมไว้ให้ใช้นอกเหนือจากออบเจ็กต์พื้นฐาน 6 ชนิดที่เราได้เรียนรู้ไปแล้ว โดยออบเจ็กต์แต่ละตัวจะมีความสามารถในการทำงานเฉพาะด้านที่แตกต่างกัน
- คอมโพเนนต์อื่น ๆ (Third Party Component) เป็นออบเจ็กต์ที่มีบริษัทซอฟต์แวร์อื่นๆสร้างไว้ให้โดยมีทั้งของฟรีและที่ต้องเสียเงินซื้อ
- ออบเจ็กต์ที่สร้างขึ้นมาเองซึ่งสามารถใช้ Visual Basic หรือ Visual C++ เขียนคอมโพเนนต์มาให้ใช้งานเฉพาะก็ได้ ซึ่งคอมโพแนนต์เหล่านั้นจะอยู่ในรูปของไฟล์ .dll ที่ทำงานบนเว็บเซิร์ฟเวอร์

15.3 อินเตอร์เน็ตอินฟอร์เมชั่นเซอร์วิส (ไอไอเอส)

การให้บริการข้อมูลข่าวสารทางอินเตอร์เน็ตหรืออินเตอร์เน็ตอินฟอร์เมชั่นเซอร์วิสนับเป็นหัวใจสำคัญของการให้บริการด้านเว็บของไมโครซอฟต์ ซึ่งเป็นหนึ่งในแอปพลิเคชั่นที่มีวิวัฒนาการเร็วที่สุดเท่า

ที่เคยมีมา ไอไอเอสมีความยืดหยุ่น สมรรถภาพ และเสถียรภาพ โดยเฉพาะไมโครซอฟต์วินโดวส์ 2000 เซิร์ฟเวอร์ ซึ่งนำไอไอเอสเวอร์ชัน 5 มารวมไว้เป็นคุณลักษณะใหม่ที่ช่วยเพิ่มเสถียรภาพและความปลอดภัยให้กับเว็บเซิร์ฟเวอร์ มากขึ้น การจัดการสภาพแวดล้อมของการให้บริการเว็บ ออกแบบมาให้ลูกค้าหลายๆคนสามารถใช้งานบนเครื่องเดียวได้ ซึ่งทำให้ได้ประโยชน์สูงสุด

ไอไอเอส 5.0 ได้เพิ่มวิซาร์ดเข้ามารองรับงานบางอย่างที่ซับซ้อนยุ่งยากของไอไอเอส 4.0 ให้สามารถทำงานได้ตามกระบวนการสำเร็จรูปที่เตรียมไว้ให้ โดยนำเซอร์ทิฟีเคทวิซาร์ด (Certificate Wizard) มาแทนที่คีย์เมเนเจอร์ (Key Manager) เพื่อสร้างการรับรองแบบเอสเอสแอล และมีเพอร์มิชชั่นวิซาร์ด (Permission Wizard) ทำให้คุณสามารถใช้วิธีง่าย ๆ มาปรับแต่งรายการ ควบคุมการเข้าถึงบนไฟล์ได้ วิซาร์ดเหล่านี้ทำให้ไอไอเอสใช้งานได้ง่ายขึ้นและกระจายความนิยมภายในหมู่ผู้ใช้บริการในวงกว้างรวมถึงลดปัญหาที่ผู้ใช้ได้รับมาก่อนลง

เซอร์ทิฟีเคทวิซาร์ดเป็นคุณลักษณะตัวใหม่ที่เข้ามาทดแทนคีย์เมเนเจอร์ โดยเฉพาะเรื่องของการใช้งานง่ายและอินเทอร์เฟซที่รวดเร็ว อนุญาตให้ผู้ดูแลระบบแจ้งการร้องขอการรับรองจากเซอร์ทิฟีเคชั่นออร์เธอริที้ (Certification Authorities) หรือซีเอ (CA) ได้ เช่นจากเวอริไซน์ (Verisign) และร้องขอการรับรองจากเซอร์ทิฟีเคทเซอร์วิส (Certificate Services) ซึ่งอาจรันอยู่บนเครือข่ายท้องถิ่น

เอเอสพีเป็นเทคโนโลยีหนึ่งที่ใช้ไอไอเอสประมวลผลสคริปต์และส่งผลที่ได้จากการทำงานของสคริปต์ออกมาเป็นไฟล์เอชทีเอ็มแอล นักพัฒนาสามารถสร้างโค้ดเอเอสพีให้มีประสิทธิภาพเพิ่มขึ้นและส่งข้อความแจ้งเตือนความผิดพลาดกลับมาได้

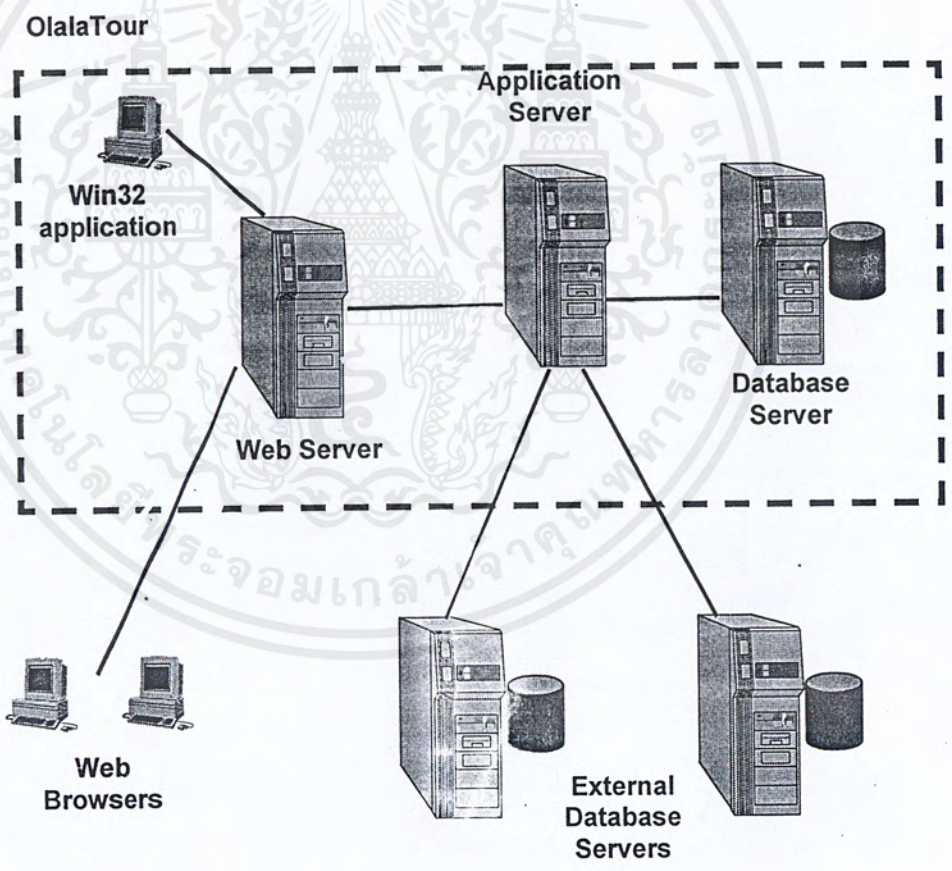
ขีดความสามารถโดยรวมของเอเอสพีได้รับการปรับปรุงให้ดีขึ้น สามารถพลิกแพลงตัวมันเองเพื่อหลีกเลี่ยงการถูกปิดกั้น โดยโปรเซสเพียงตัวเดียว ซึ่งเวอร์ชันที่แล้วจะทำอยู่ในส่วนของโปรเซสไฟล์เอเอสพี ไม่เก็บโค้ดของสคริปต์เอเอสพีที่ถูกโปรเซสเหมือนกับไฟล์เอชทีเอ็มแอล แต่อนุญาตให้นักพัฒนาสร้างเอกสารทั้งหมดโดยมีส่วนขยายเป็น .asp ได้โดยไม่ทำให้ประสิทธิภาพของระบบเสียหายหรือลดลง

บทที่ 16
การวิเคราะห์และออกแบบระบบงาน

16.1 การวิเคราะห์ระบบงาน

ในโครงงานนี้ได้พัฒนาและออกแบบระบบงานเชิงคอมโพเนนต์ สร้างบริษัทชื่อ OlalaTour ให้บริการนำเที่ยว จองที่พักและตั๋วเดินทางโดยให้บริการผ่านอินเตอร์เน็ต ระบบประกอบด้วย 4 ส่วนหลักคือ

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



รูปที่ 16-1 โมเดลขอ[illegible]ระบบงาน

โมเดลของระบบงานแสดงดังรูปที่ 16-1 เป็นสถาปัตยกรรมแบบทรีเทียร์ส (Three Tiers) ประกอบด้วย

1. **พรีเซ็นเตชันเทียร์** มีไคลเอนต์ 2 แบบ คือเว็บบราวเซอร์ไคลเอนต์ และวิน32แอปพลิเคชั่น (Win32Application) โดยใช้เว็บบราวเซอร์ให้บริการกับผู้ใช้ทั่วไป และใช้แอปพลิเคชั่นในการสร้างแพ็กเกจทัวร์ ผู้ที่ใช้แอปพลิเคชันจะเป็นผู้ดูแลระบบเท่านั้น
2. **บิสซิเนสเทียร์** ใช้อินเตอร์เน็ตอินฟอร์เมชันเซอร์วิส 5.0 (Internet Information Service 5.0) เป็นเว็บเซิร์ฟเวอร์ และคอมโพเนนต์เซอร์วิสเป็นแอปพลิเคชั่นเซิร์ฟเวอร์ เซอร์วิสทั้งสองเป็นเซอร์วิสของระบบปฏิบัติการวินโดวส์ 2000 แอดวานซ์เซิร์ฟเวอร์ คอมโพเนนต์ทั้งหมดของระบบจะถูกควบคุมโดยคอมโพเนนต์เซอร์วิส เอกสารเอชทีเอ็มแอลและเอกสารเอเอสพีถูกจัดการโดยอินเตอร์เน็ตอินฟอร์เมชันเซอร์วิส
3. **ดาต้าเทียร์** ใช้ไมโครซอฟต์เอสคิวแอลเซิร์ฟเวอร์ 2000 เป็นระบบจัดการฐานข้อมูล ฐานข้อมูลของบริษัท OlalaTour จะอยู่ในเครือข่ายเดียวกันกับคอมโพเนนต์ ส่วนฐานข้อมูลของสายการบิน รถประจำทาง โรงแรม จะอยู่คนละเครือข่าย

ในเอกสารนี้จะกล่าวถึงเฉพาะโครงสร้างของส่วนจองการเดินทาง ระบบจองการเดินทาง ซึ่งประกอบด้วยบริการต่าง ๆ ของระบบรถโดยสารและสายการบินต่าง ๆ ดังต่อไปนี้

1. บริการค้นหาเส้นทางการเดินทาง
2. จองการเดินทาง
3. ยืนยันจองการเดินทาง
4. ยกเลิกจองการเดินทาง

การค้นหาข้อมูลรถโดยสาร

ขั้นตอนแรกผู้ใช้จะเริ่มใช้บริการจากระบบงาน จะทำการกรอกรายละเอียดที่ต้องการหา ผลลัพธ์ที่ออกมาจะเป็นรายละเอียดต่าง ๆ ของเที่ยวการเดินทางที่เกี่ยวข้องกับคีย์เวิร์ดที่ผู้ใช้ป้อนเข้าไป ผู้ใช้สามารถเลือกในแต่ละเที่ยวการเดินทางเพื่อดูรายละเอียดเพิ่มเติมได้

การค้นหาข้อมูลเที่ยวบินมีข้อมูลที่สามารถใช้เป็นเงื่อนไขได้ดังนี้

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

- การเรียงลำดับจากตัวเลือกต่างๆที่ค้นหาเจอ เช่น เรียงจากราคาที่ต่ำสุด ใช้เวลาการเดินทางน้อยที่สุด

โดยผลลัพธ์จากการค้นหาจะมีรายละเอียดของสายการบิน ดังนี้

- ชื่อเที่ยวบิน
- ชื่อสายการบิน
- สถานีต้นทาง – ปลายทาง
- เวลาขึ้นเครื่อง/เวลาลงเครื่อง
- ราคา

การค้นหาข้อมูลรถโดยสาร

ขั้นตอนนี้จะทำงานเมื่อผู้ใช้เลือกรายการค้นหาสายการเดินรถ โดยผู้ใช้จะป้อนคุณสมบัติโดยคร่าว ๆ ของสายการเดินรถที่ตนต้องการจะใช้งาน

โดยรายละเอียดข้อมูลในการค้นหารถโดยสารมีดังนี้

- สถานที่ต้นทาง
- สถานที่ปลายทาง
- ตัวเลือกว่าต้องการเดินทางไปอย่างเดียว หรือ ไป-กลับ
- วันที่ เวลา ของการออกเดินทางไป
- หากเลือกการเดินทางไปกลับสามารถระบุ วันที่ เวลา ของการเดินทาง
- ชนิดของที่นั่ง
- ชื่อบริษัทที่ต้องการใช้บริการ
- การเรียงลำดับจากตัวเลือกต่างๆที่ค้นหาเจอ เช่น เรียงจากราคาที่ต่ำสุด,ใช้เวลาการเดินทางน้อยที่สุด

ผลลัพธ์จากการค้นหาจะมีรายละเอียดรถบัส มีดังนี้

- ชื่อบริษัทขนส่ง
- ชื่อสายการเดินรถ
- สถานีต้นทาง
- สถานีปลายทาง
- เวลารถออก
- เวลาถึงที่หมาย
- ราคา

16.2 การออกแบบระบบงาน

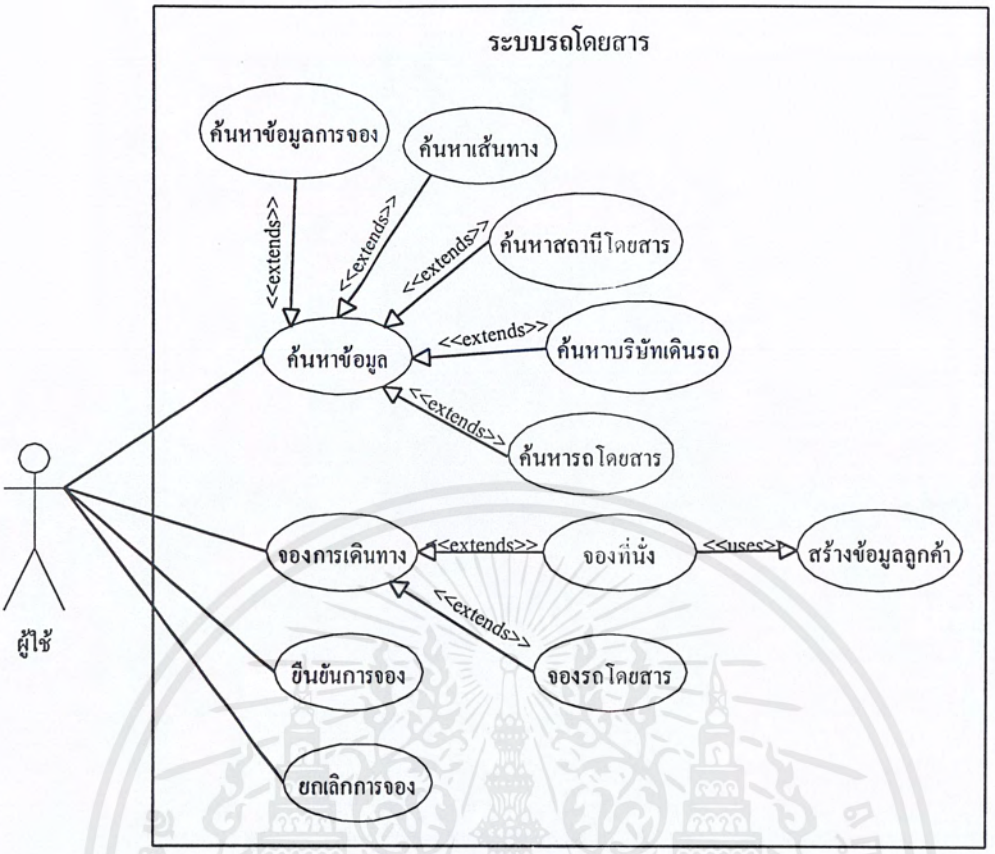
ในโครงงานนี้จะนำเสนอการออกแบบระบบเฉพาะส่วนของระบบรถโดยสารและระบบสายการบิน ซึ่งระบบงานบริษัท Olala Tour จะนำไปใช้โดยมีรายละเอียดดังนี้

16.2.1 สร้าง Use Case

เรานำรูปแบบระบบงานมาแยกออกเป็นฟังก์ชันการทำงานโดยรวมของระบบ ในรูปแบบของยูสเคส ซึ่งมียูสเคสดังนี้

ยูสเคสของระบบรถโดยสาร

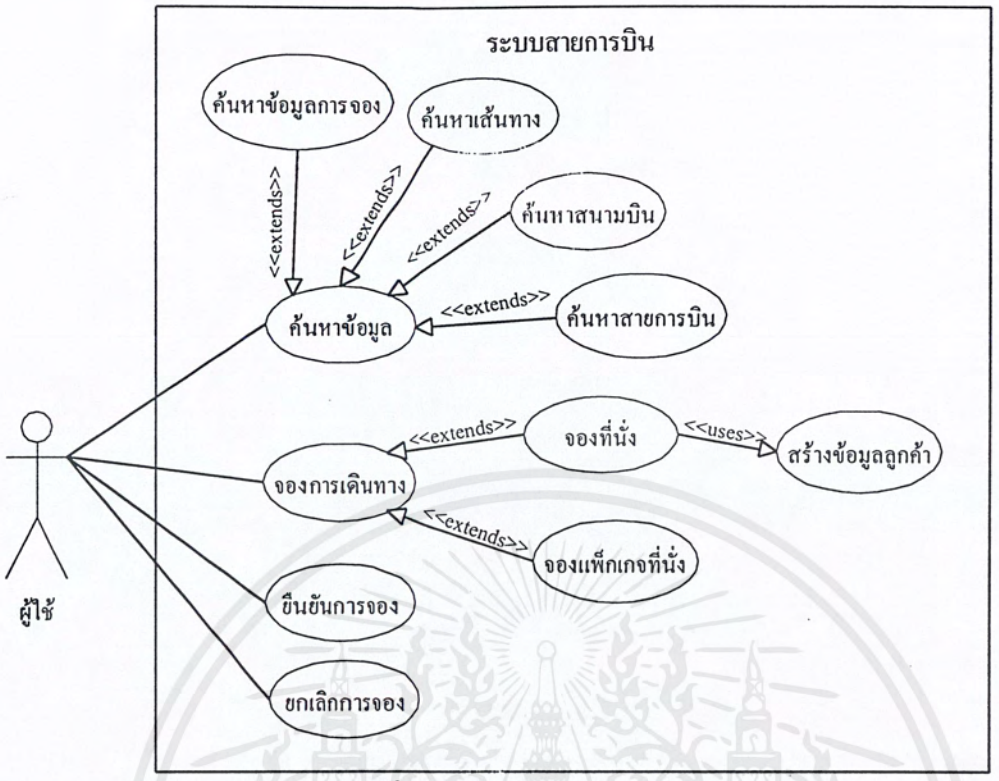
- ค้นหาข้อมูล
  - ค้นหาข้อมูลข้อมูลการจอง
  - ค้นหาเส้นทางเดินรถ
  - ค้นหาสถานีโดยสาร
  - ค้นหาบริษัทเดินรถ
  - ค้นหารถโดยสาร
- จองการเดินทาง
  - จองที่นั่ง
  - จองรถโดยสาร (เพื่อจัดทำแพกเกจทัวร์)
- ยืนยันการจอง
- ยกเลิกการจอง



รูปที่ 16-2 ยูสเคสของระบบรถโดยสาร

ยูสเคสของระบบสายการบิน

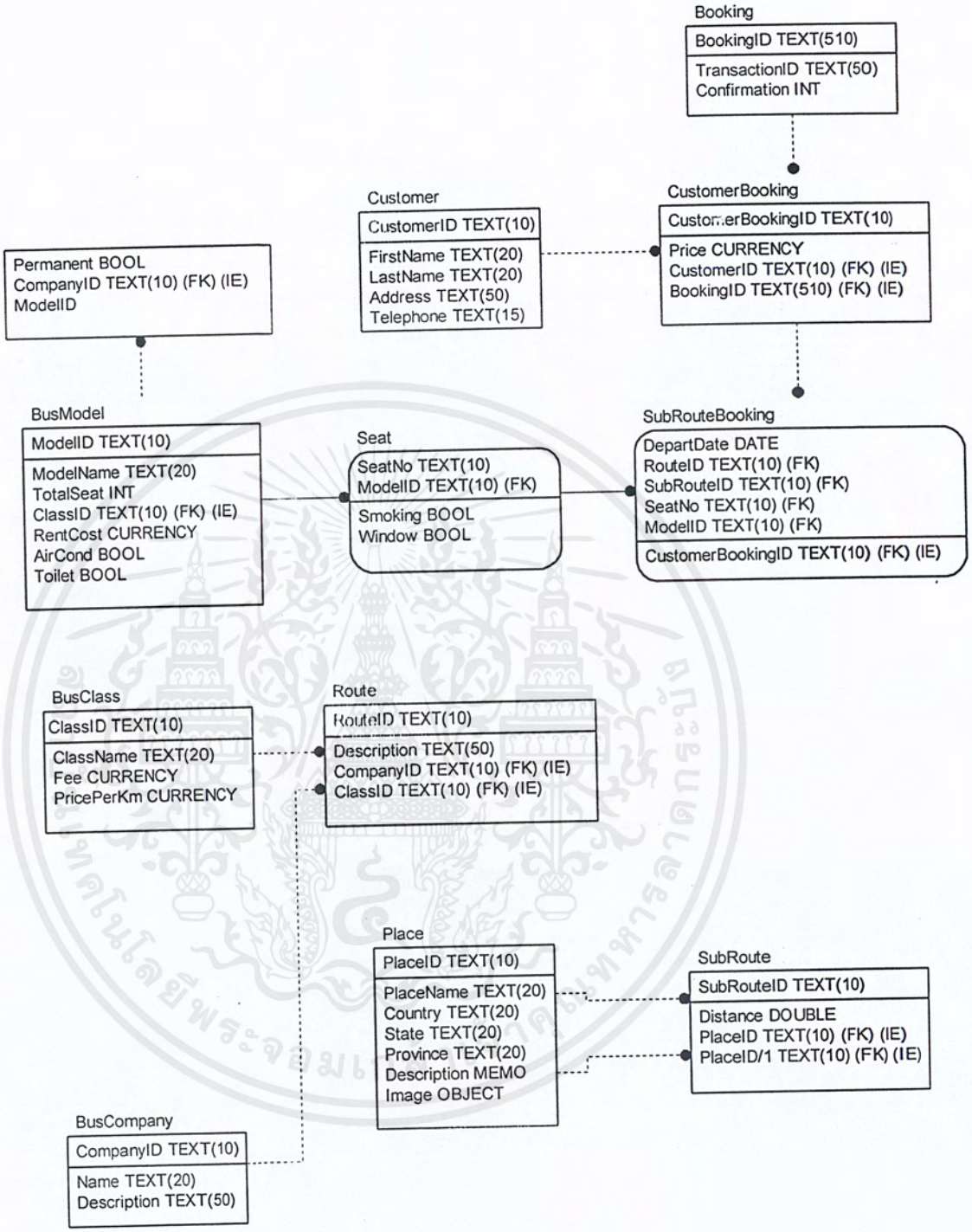
- ค้นหาข้อมูล
  - ค้นหาข้อมูลข้อมูลการจอง
  - ค้นหาเส้นทางบิน
  - ค้นหาสถานีสนามบิน
  - ค้นหาบริษัทสายการบิน
- จองการเดินทาง
  - จองที่นั่ง
  - จองชุดที่นั่ง (เพื่อจัดทำแพกเกจทัวร์)
- ยืนยันการจอง
- ยกเลิกการจอง

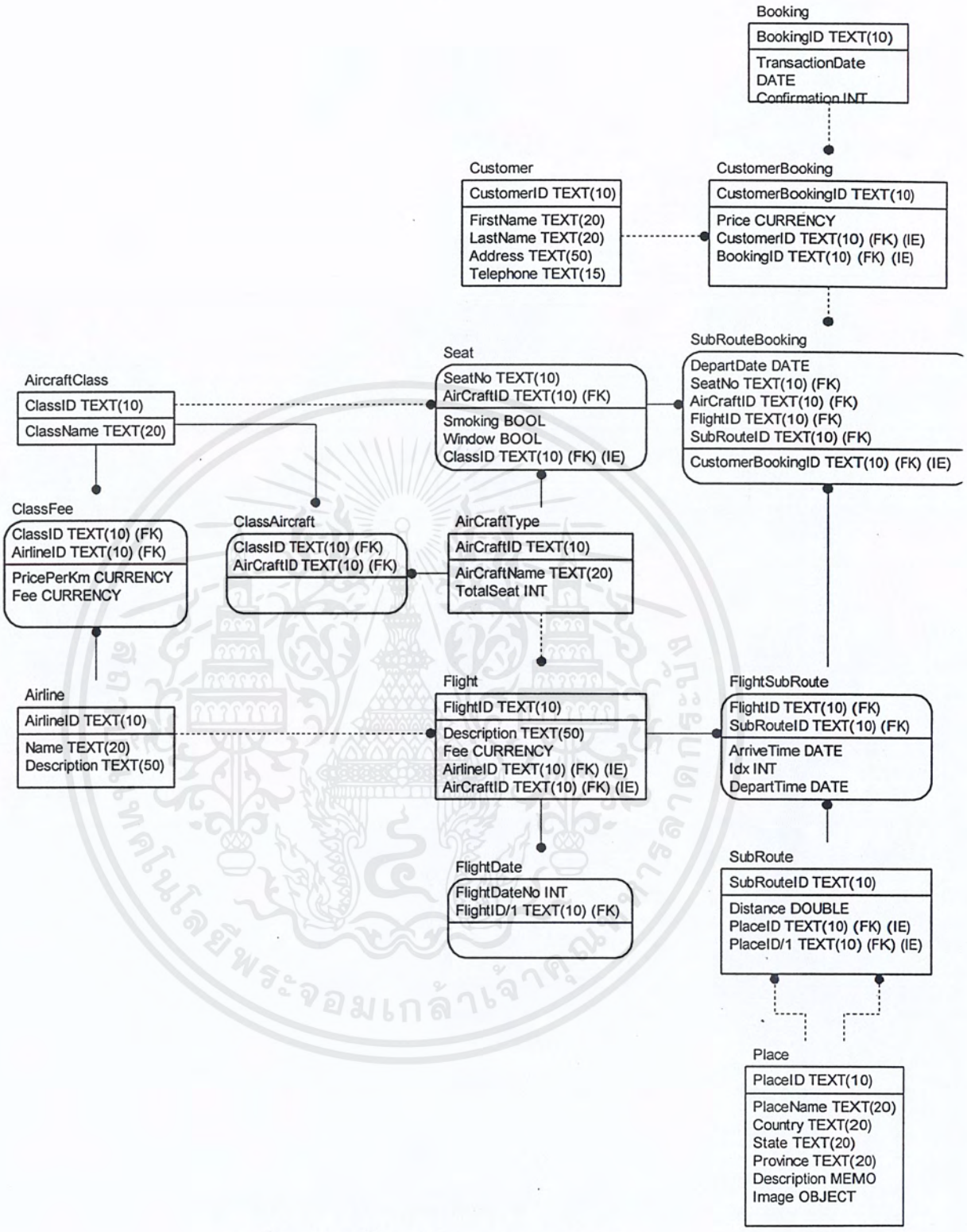


รูปที่ 16-3 ยูสเคสของระบบสายการบิน

16.2.2 ออกแบบฐานข้อมูล

การออกแบบลักษณะของข้อมูลของระบบงานมีความสำคัญมาก เราใช้อีอาร์ไดอะแกรม เป็นสัญลักษณ์แสดงความสัมพันธ์ระหว่างข้อมูลดังนี้

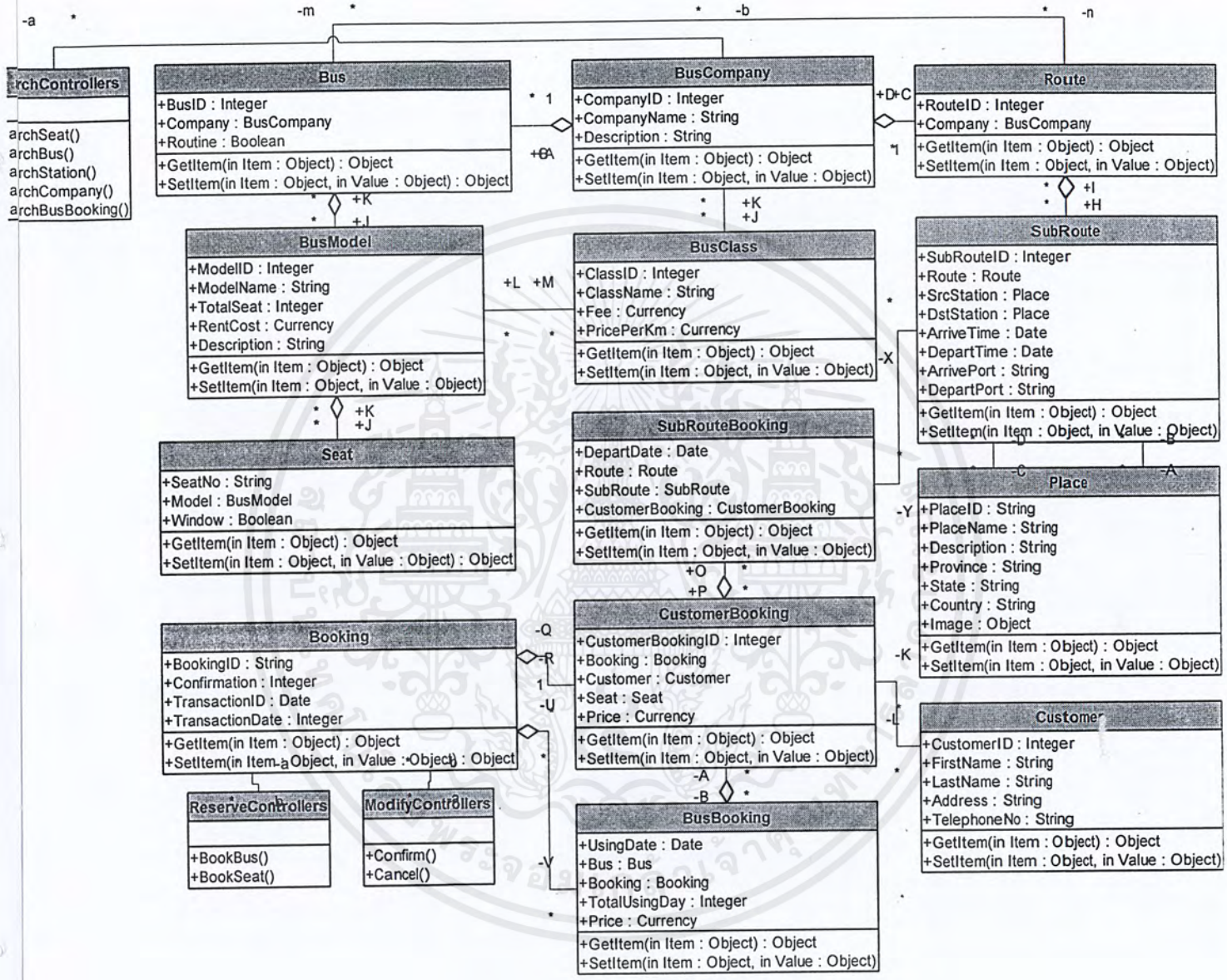




รูปที่ 16-5 อีอาร์ไดอะแกรมของระบบสายการบิน

16.2.3 ออกแบบคลาสไดอะแกรม

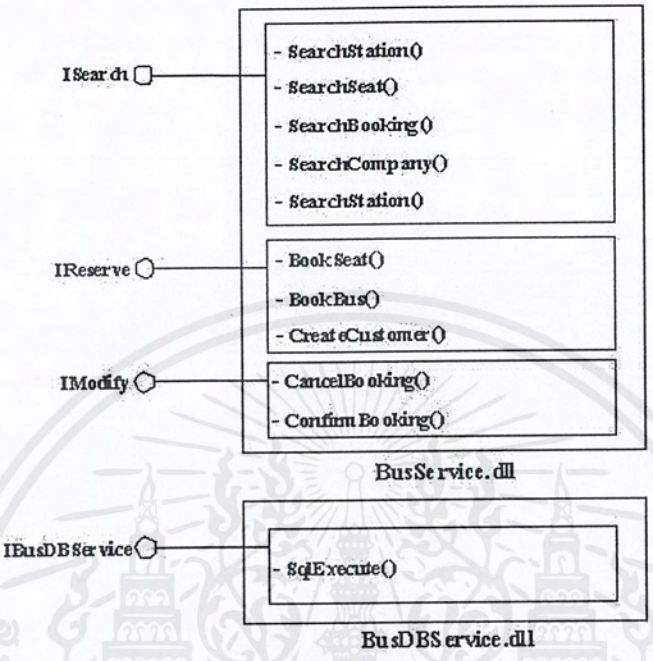
เรานำอีอาร์ไดอะแกรมมาเป็นหาความสัมพันธ์ของข้อมูลในรูปแบบของคลาสไดอะแกรม ซึ่งโดยส่วนใหญ่เราสามารถแม็บเอ็นติตีมากกว่า 1 เอ็นติตีไปเป็นคลาส 1 คลาส



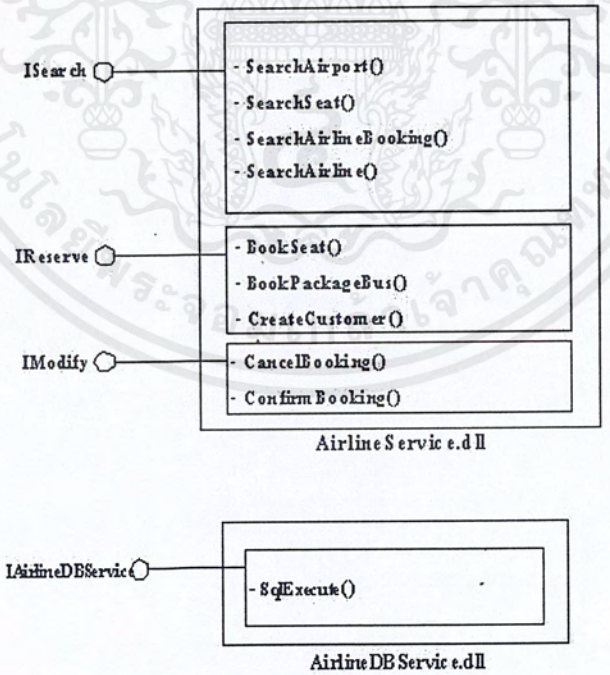
รูปที่ 16-6 คลาสไดอะแกรมของระบบรถโดยสาร

16.2.4 ออกแบบคอมโพเนนต์ไดอะแกรม

จากคลาสไดอะแกรมที่ได้ เรานำมาออกแบบเป็นคอมโพเนนต์ไดอะแกรม สิ่งที่ได้จากคอมโพเนนต์ไดอะแกรมจะเป็นไฟล์ DLL ที่เป็นคอมโพเนนต์ของของระบบงาน



รูปที่ 16-7 คอมโพเนนต์ของระบบรถโดยสาร



รูปที่ 16-8 คอมโพเนนต์ของระบบสายการบิน

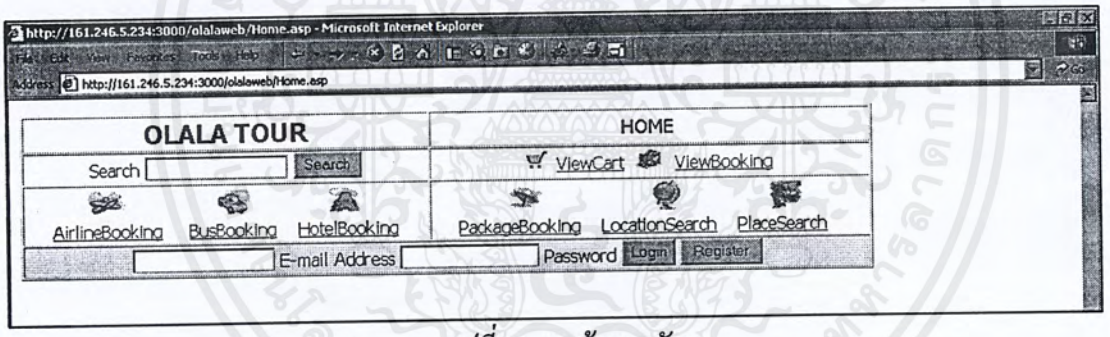
บทที่ 17
ลักษณะแอปพลิเคชั่น

17.1 ระบบงาน Olala Tour บนอินเตอร์เน็ต

Olala Tour บนอินเตอร์เน็ตเป็นลักษณะการทำธุรกิจเชิงพาณิชย์ที่ทำผ่านระบบเครือข่ายอินเตอร์เน็ต โดยที่ลูกค้าสามารถเข้าไปเลือกชมสินค้าและบริการต่างๆผ่านเว็บบราวเซอร์(Web Browser) ที่สามารถเข้าใช้บริการได้ตลอดเวลา เมื่อลูกค้าตกลงใจเลือกจองสินค้าแล้ว ก็ทำการตกลงชำระเงินผ่านสื่อต่างๆ เช่น เครดิตการ์ด เดบิตการ์ด การชำระเงินผ่านบัญชีธนาคาร เป็นต้น เมื่อผู้ให้บริการได้รับหลักฐานทุกอย่างครบถ้วนแล้วก็จะทำการจัดส่งสินค้าหรือให้บริการด้านนั้นๆแก่ลูกค้าก็เป็นอันสิ้นสุดขบวนการทำงาน

ในปัจจุบันเครือข่ายอินเตอร์เน็ตได้มีใช้กันอย่างแพร่หลายมากขึ้น โดยเพื่อให้เกิดการทำงานนี้เกิดขึ้นต้องอาศัยเทคนิคในหลายรูปแบบทั้งในด้านฮาร์ดแวร์และซอฟต์แวร์ ซึ่งโปรเจคนี้จะเกี่ยวกับการโปรแกรมโดยอาศัยเทคโนโลยีด้าน COM ,DCOM ,DNS ,DNA ,MTS ,COM+ ,Active Directory ,IIS ,ASP เป็นหลักสำคัญในการประมวลผลในการทำงานของฟังก์ชันต่างๆและในการติดต่อฐานข้อมูลของทั้ง ระบบOlala Tour ระบบสายการบิน ระบบรถบัส ระบบโรงแรมและลูกค้า

17.1.1 หน้าจอหลัก

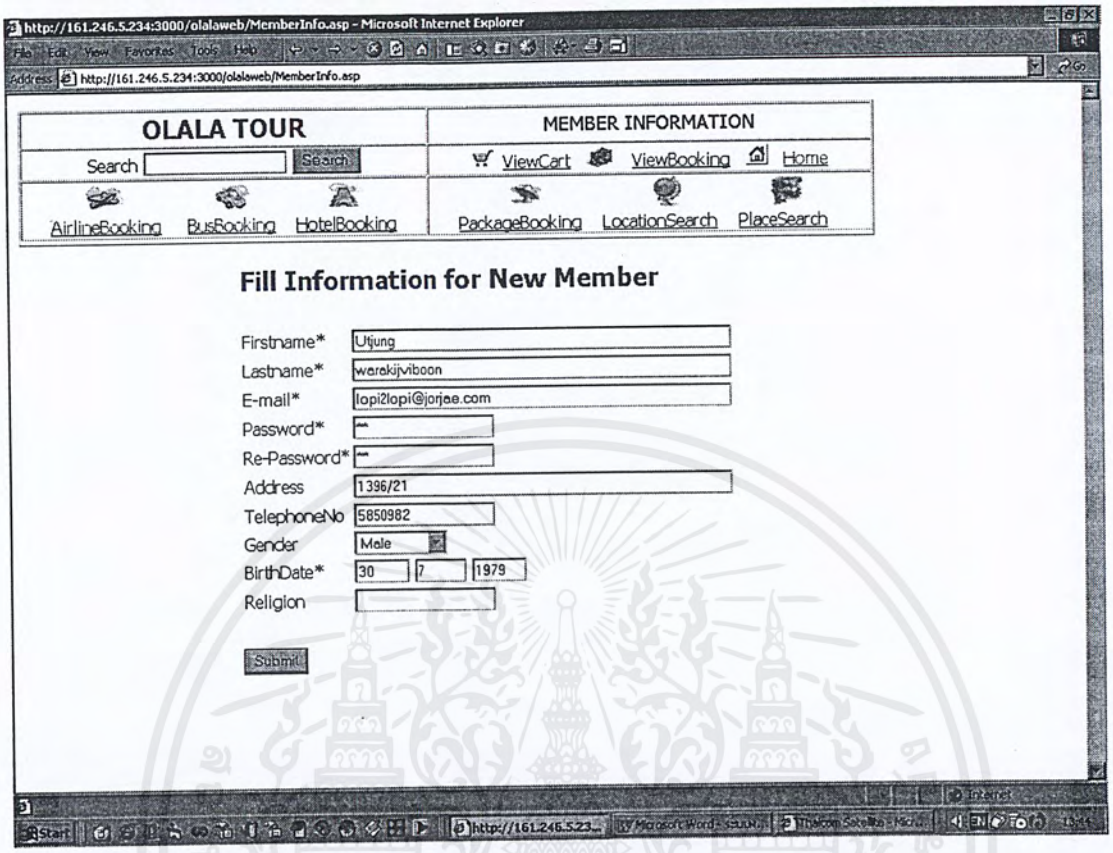


รูปที่ 17-1 หน้าจอหลัก

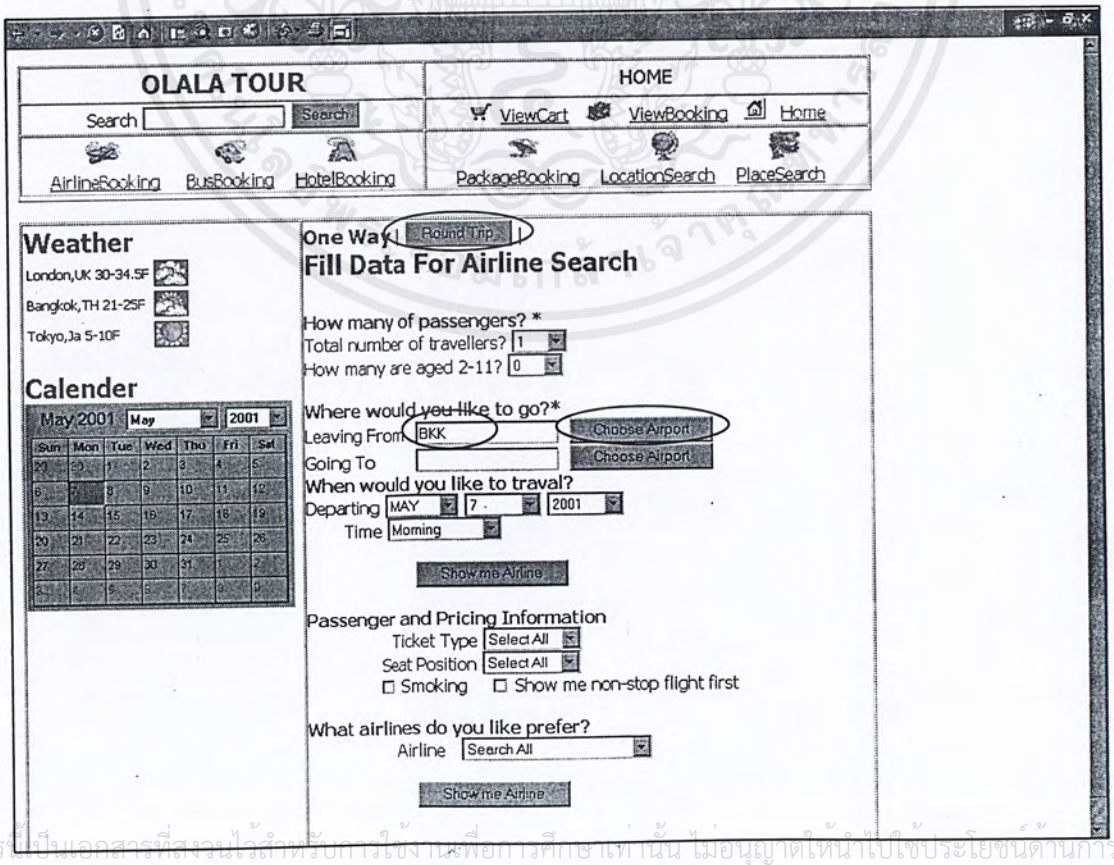
ที่หน้าจอหลักจะมี link ไปที่การให้บริการอื่นๆคือ

- AirlineBooking เป็นการค้นหาสายการบินที่เราต้องการ
- BusBooking เป็นการค้นหาเที่ยวการเดินรถที่เราต้องการ
- HotelBooking เป็นการค้นหาโรงแรมที่เราต้องการ
- PackageBooking เป็นการค้นหาชุดการท่องเที่ยวที่เราต้องการ
- LocationSearch และ PlaceSearch เป็นการค้นหาสถานที่ที่เราต้องการ
- ViewCart เป็นการตรวจดูรถเข็นที่เราใส่สินค้าไว้
- ViewBooking เป็นการตรวจดูสินค้าที่เราทำการจองไว้
- Login เป็นการเข้าสู่ระบบ
- Register เป็นการสมัครสมาชิก

17.1.2 การสมัครเป็นสมาชิก



รูปที่ 17-2 การสมัครสมาชิก



รูปที่ 17-3 การค้นหาสายการบิน

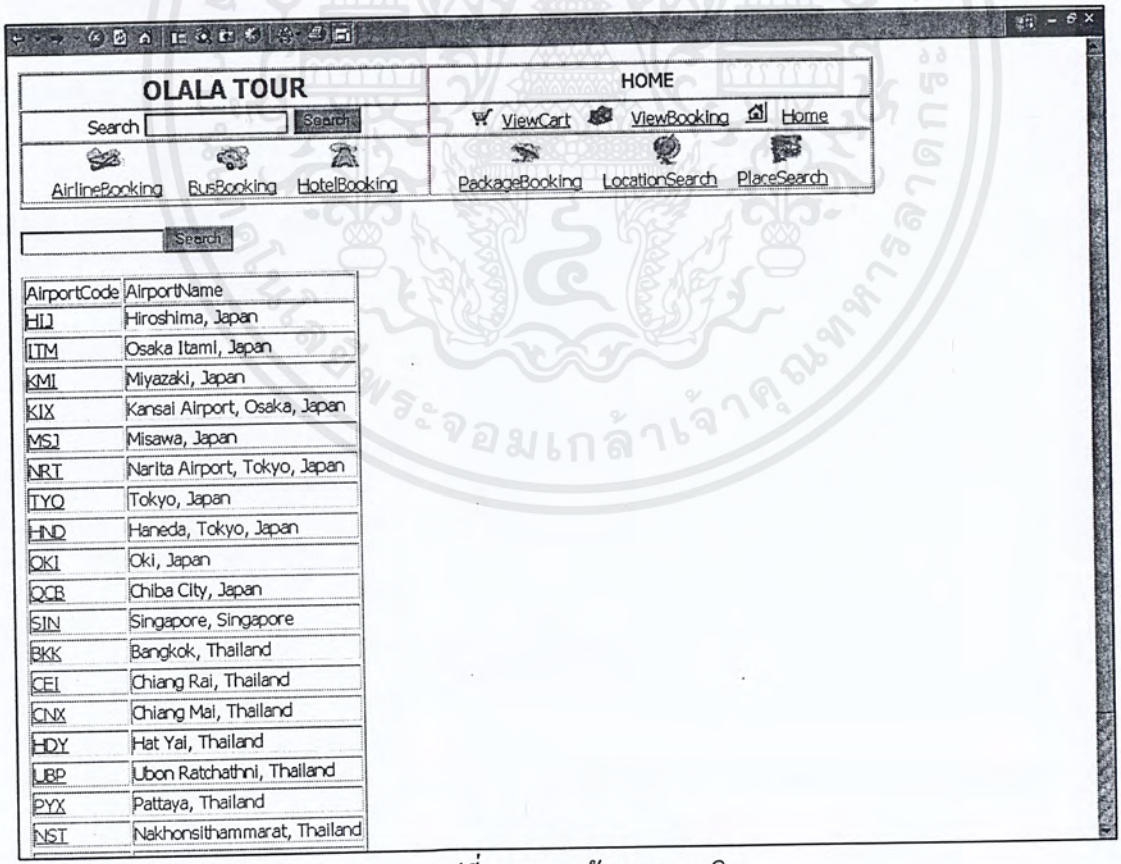
ในกรณีที่ลูกค้าต้องการซื้อหรือจองสินค้าจำเป็นต้องสมัครเป็นสมาชิกก่อน โดยการทำงานคือให้ลูกค้ากรอกรายละเอียดต่างๆเพื่อเข้าเป็นสมาชิก และลูกค้าสามารถที่จะใช้ชื่ออีเมล์และรหัสผ่านในการเข้าใช้งานในครั้งต่อไป

แต่ในกรณีที่ลูกค้าต้องการค้นหาสินค้าและทำการเลือกสินค้าใส่รถเข็นลูกค้ายังไม่จำเป็นต้องสมัครเป็นสมาชิกก่อน เมื่อจะทำการจองสินค้าจึงทำการเข้าระบบหรือสมัครสมาชิกใหม่

17.1.3 การค้นหาสินค้าและการเลือกสินค้าใส่รถเข็น

1. การค้นหาตั๋วเครื่องบินมีข้อมูลที่ใส่ในการค้นหาคือ

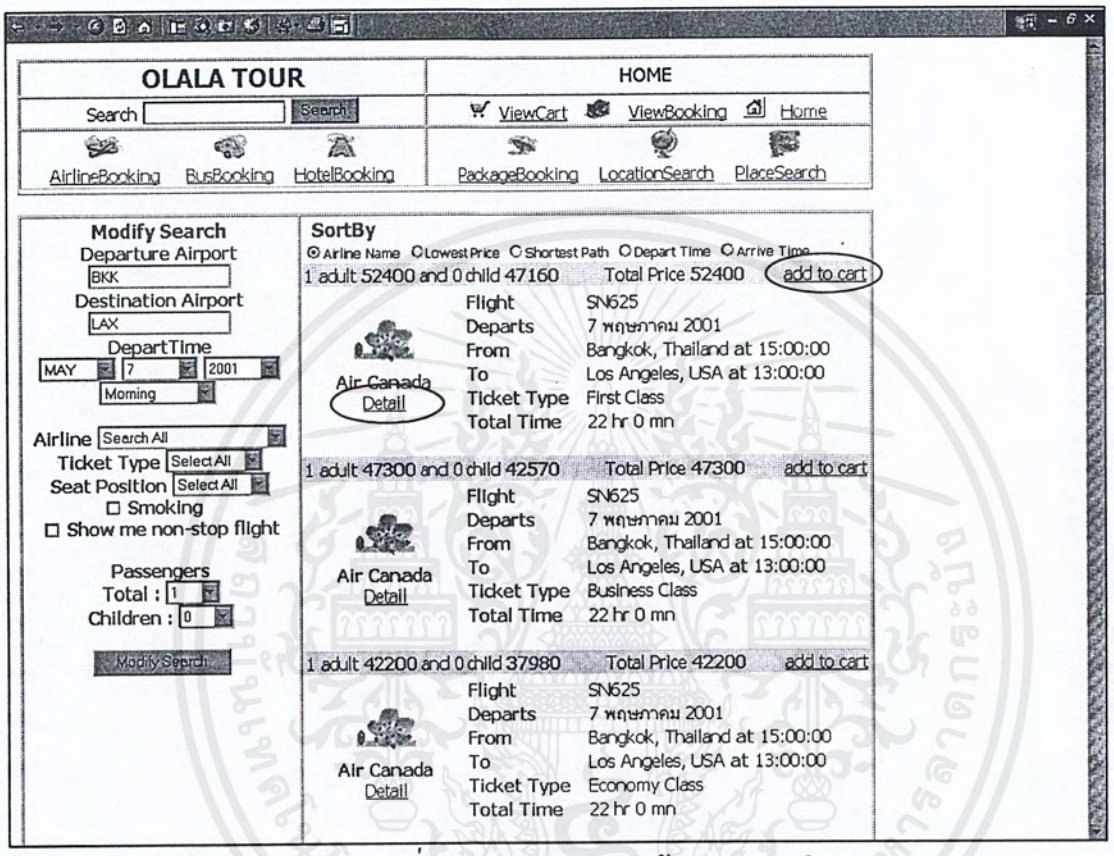
- ชนิดตั๋วไปอย่างเดียวหรือตั๋วไป-กลับ โดยกดที่ปุ่ม **Round Trip**
- จำนวนผู้โดยสารโดยจะให้เลือกจำนวนผู้โดยสารทั้งหมดและจำนวนเด็กอายุระหว่าง 2-11 ปี เพื่อคำนวณราคาที่ต่างกัน
- สถานที่ต้นทางและปลายทางที่จะไป สามารถที่จะเลือกสนามบินได้โดยกด **Choose Airport**
- วันที่จะเดินทาง
- ชนิดของตั๋วและตำแหน่งที่นั่ง
- บริษัทสายการบิน



รูปที่ 17-4 การค้นหาสนามบิน

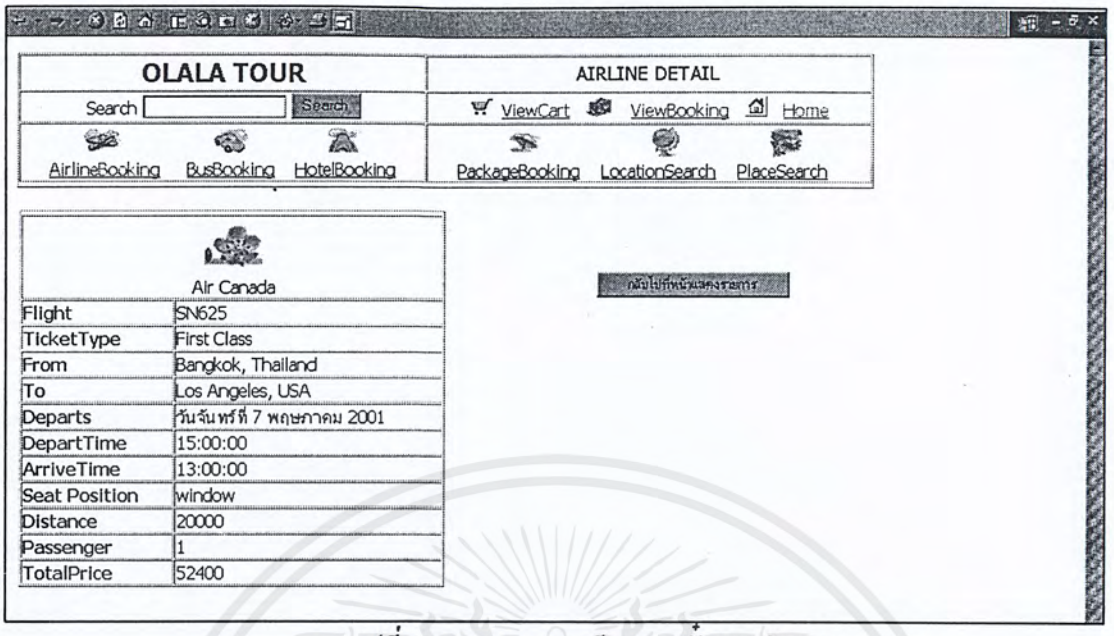
เมื่อกดปุ่ม **show me airline** จะแสดงตั๋วการเดินทางที่เป็นไปได้ โดยที่หน้าจอนี้สามารถที่จะเปลี่ยนแปลงการค้นหาได้และสามารถเรียงลำดับการค้นหาตาม

- ชื่อสายการบิน
- ราคาตั๋ว
- เวลาในการเดินทาง
- เวลาที่ออกเดินทาง
- เวลาที่ถึงปลายทาง



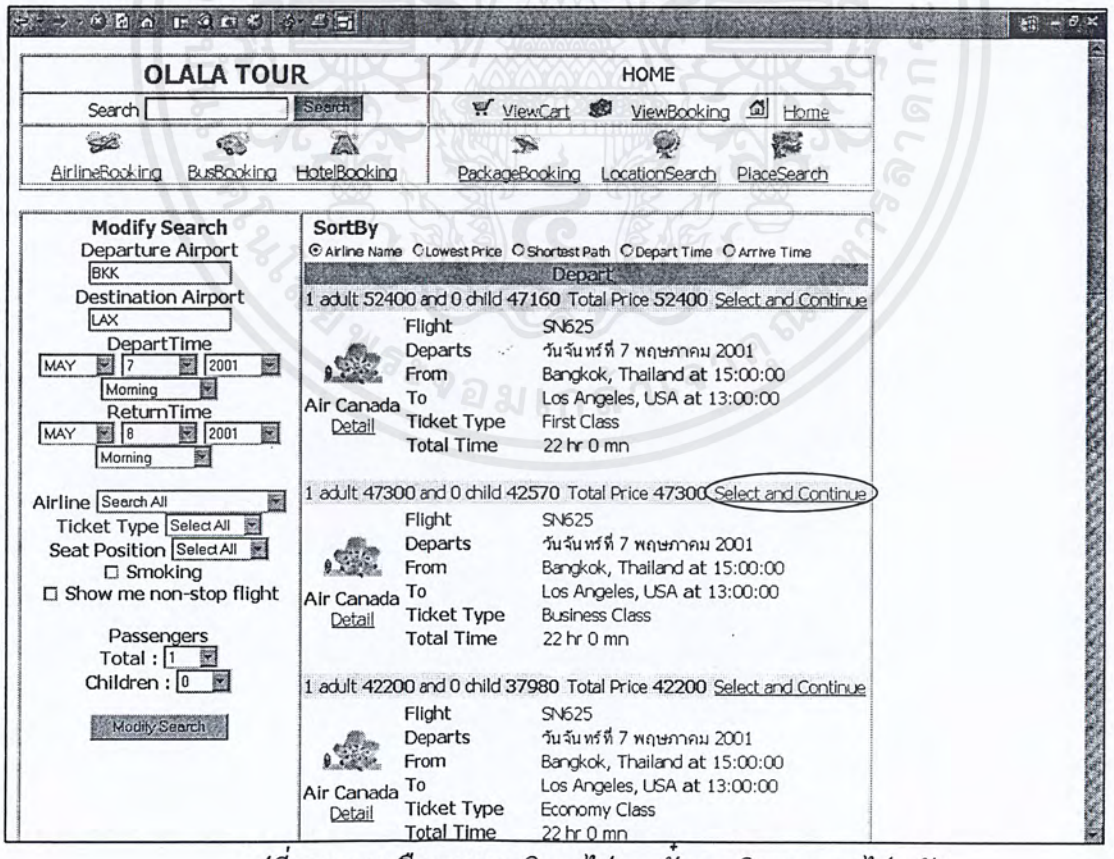
รูปที่ 17-5 การแสดงผลการค้นหาสายการบิน

สามารถดูรายละเอียดของตั๋วได้จากการกดปุ่ม **Detail** และหยิบรายการนั้นใส่รถเข็นโดยกดปุ่ม **add to cart**

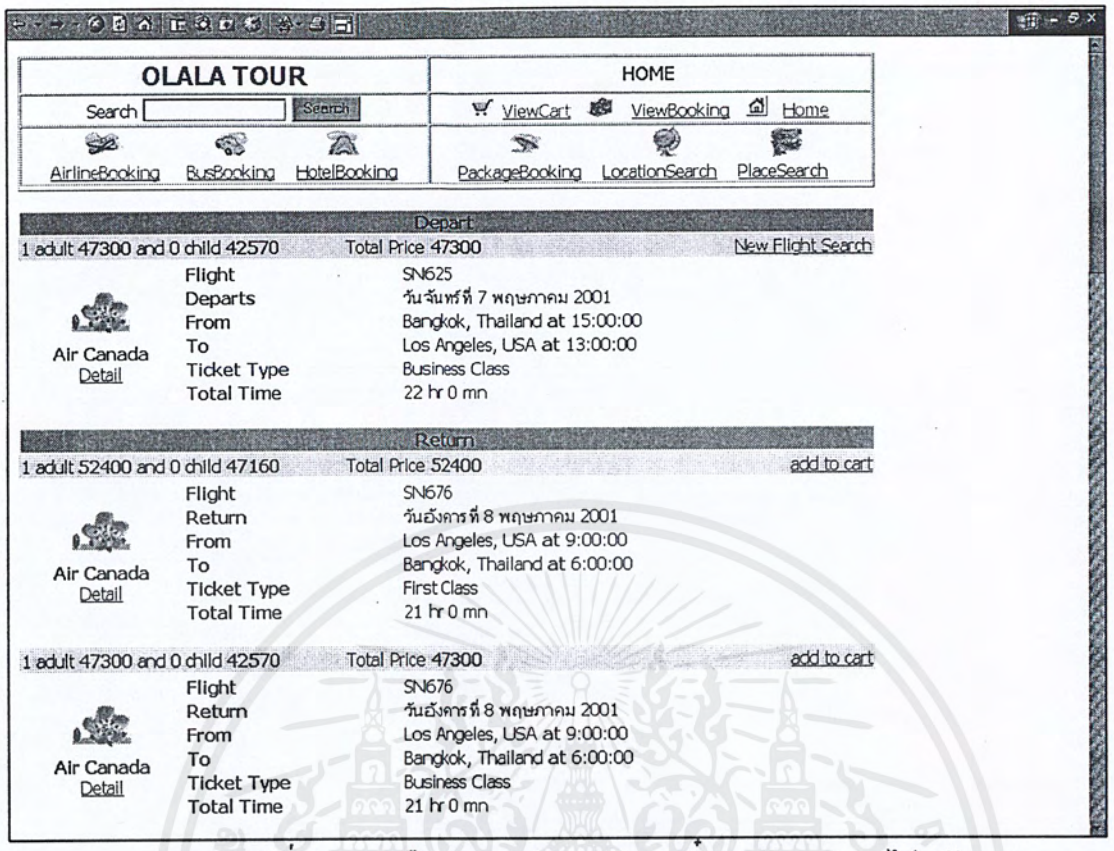


รูปที่ 17-6 การดูรายละเอียดของตั๋วการเดินทาง

การค้นหาตั๋วเครื่องบินชนิดไป-กลับ โดยการกดปุ่ม Round Trip ที่หน้าการค้นหาตั๋วเครื่องบินจะมีตัวเลือกเวลาการเดินทางกลับให้ใส่และเมื่อทำการค้นหาจะแสดงรายการตั๋วเครื่องบินเที่ยวไปที่เป็นไปได้ตามข้อมูลในการค้นหาก่อนและเมื่อเลือกรายการนั้นแล้วจึงทำการเลือกตั๋วเครื่องบินเที่ยวกลับ



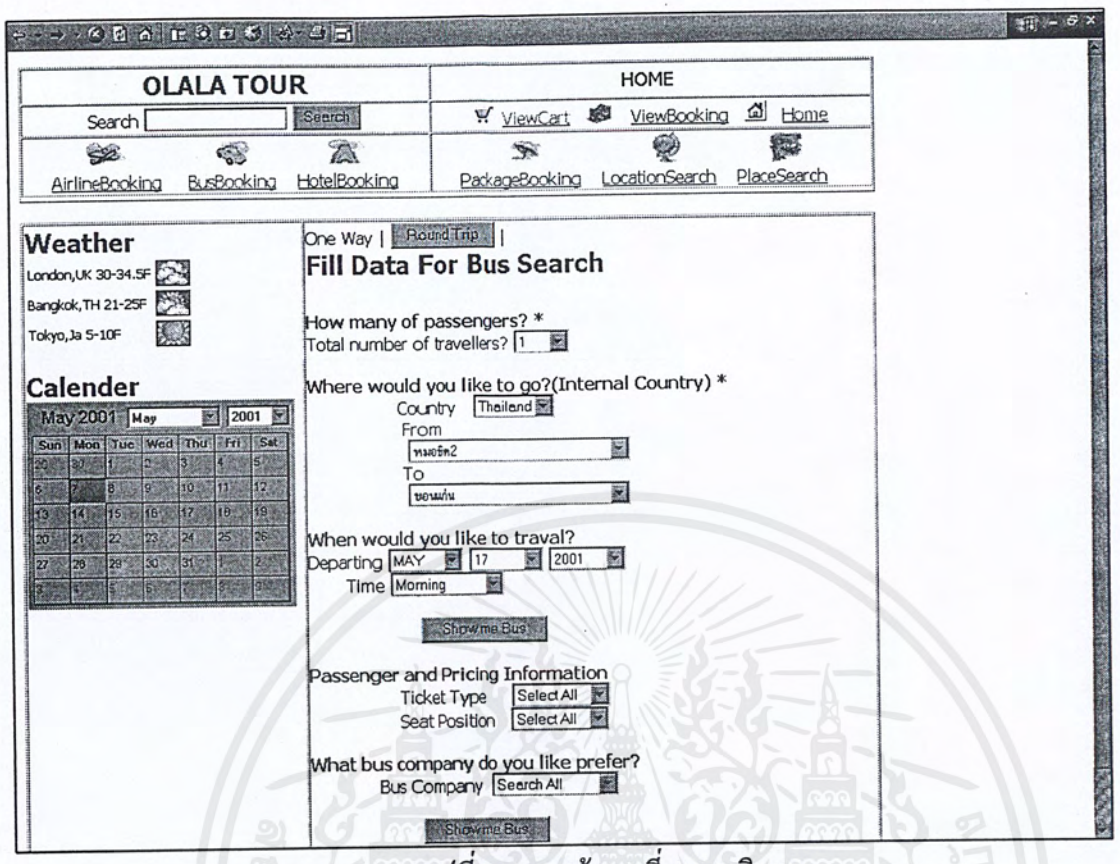
รูปที่ 17-7 การเลือกสายการบินขาไปของตั๋วการเดินทางแบบไป-กลับ



รูปที่ 17-8 การเลือกสายการบินขากลับของตั๋วการเดินทางแบบไป-กลับ

2. การค้นหาตั๋วการเดินรถมีข้อมูลที่ใส่ในการค้นหาคือ

- ชนิดตั๋วไปอย่างเดียวหรือตั๋วไป-กลับ โดยกดที่ปุ่ม **Round Trip**
- จำนวนผู้โดยสาร โดยจะให้เลือกจำนวนผู้โดยสารทั้งหมด
- สถานที่ต้นทางและปลายทางที่จะไป
- วันที่จะเดินทาง
- ชนิดของตั๋วและตำแหน่งที่นั่ง
- บริษัทสายการเดินรถ

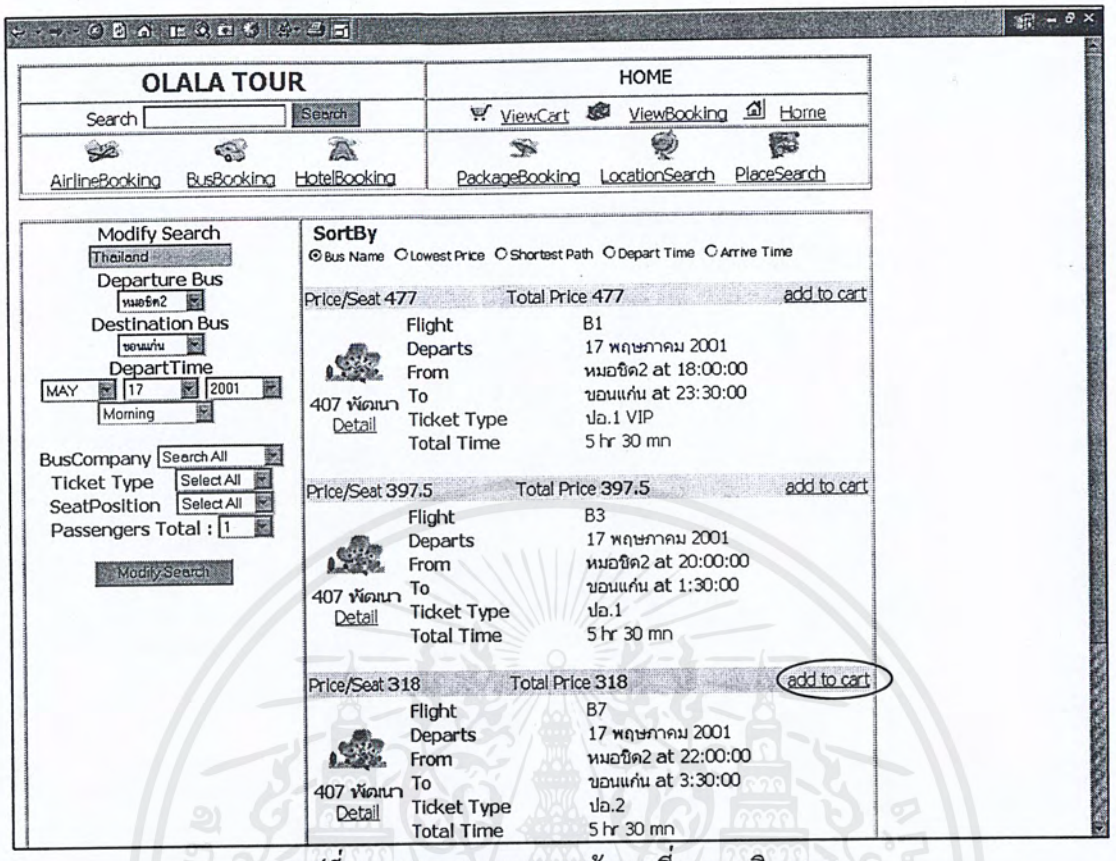


รูปที่ 17-9 การค้นหาเที่ยวการเดินรถ

เมื่อกดปุ่ม **show me bus** จะแสดงตั๋วการเดินทางที่เป็นไปได้ โดยที่หน้าจอนี้สามารถที่จะเปลี่ยนแปลงการค้นหาได้และสามารถเรียงลำดับการค้นหาตาม

- ชื่อสายการเดินรถ
- ราคาตั๋ว
- เวลาในการเดินทาง
- เวลาที่ออกเดินทาง
- เวลาที่ถึงปลายทาง

สามารถดูรายละเอียดของตั๋วได้จากการกดปุ่ม **Detail** และหยิบรายการนั้นใส่รถเข็นโดยกดปุ่ม **add to cart**

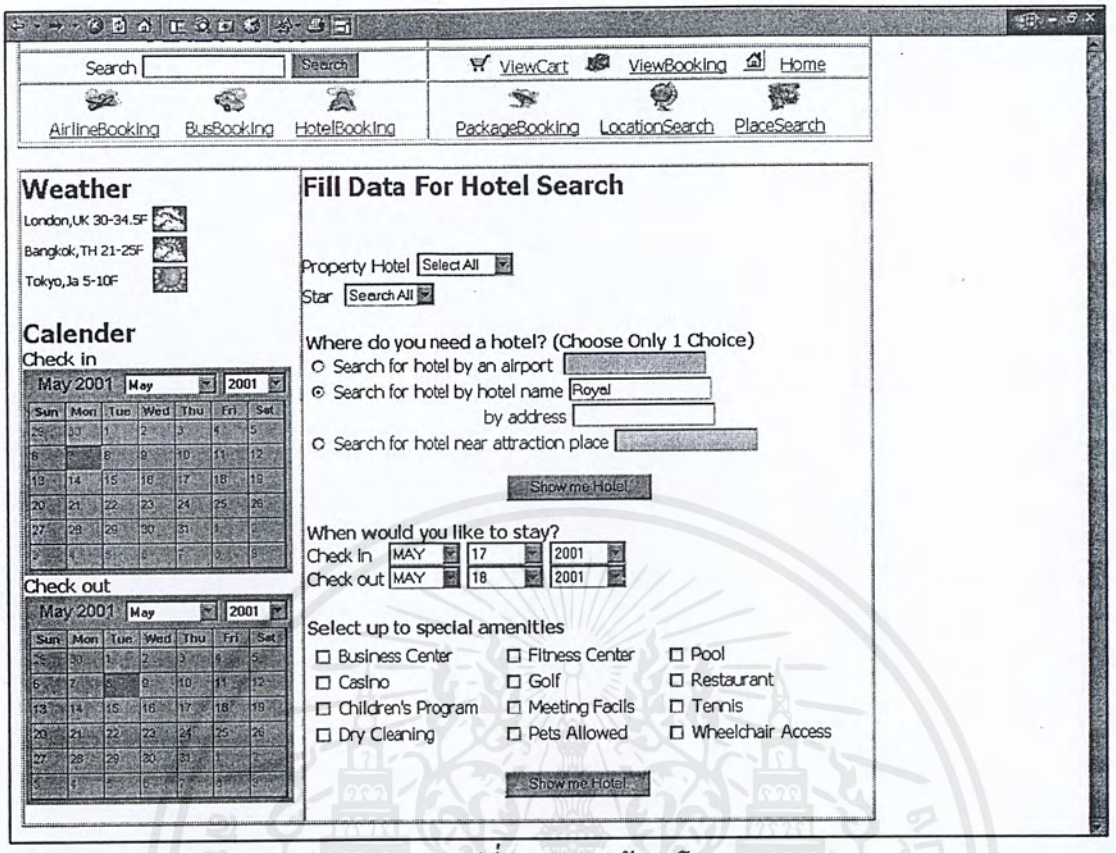


รูปที่ 17-10 การแสดงผลการค้นหาเที่ยวการเดินรถ

การค้นหาตั๋วการเดินรถชนิดไป-กลับ โดยการกดปุ่ม Round Trip ที่หน้าการค้นหาตั๋วการเดินรถ จะมีตัวเลือกเวลาการเดินทางกลับให้ใส่และเมื่อทำการค้นหาจะแสดงรายการตั๋วการเดินรถเที่ยวไปที่เป็นไปได้ตามข้อมูลในการค้นหาก่อนและเมื่อเลือกรายการนั้นแล้วจึงทำการเลือกตั๋วเการเดินรถเที่ยวกลับ

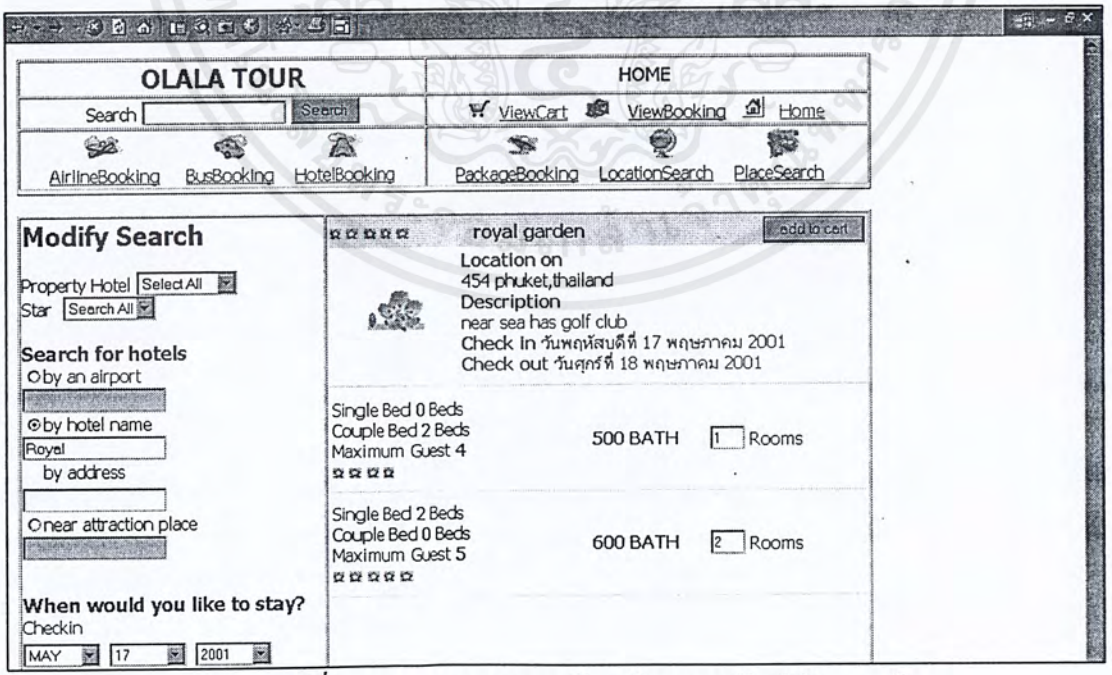
3. การค้นหาโรงแรมมีข้อมูลที่ใส่ในการค้นหาคือ

- ชนิดของโรงแรม
- ระดับของโรงแรม
- เลือกโรงแรมได้จากใกล้สนามบินหรือชื่อที่อยู่ของโรงแรมหรือใกล้สถานที่สำคัญอะไร
- วันที่จะเข้าพักและออก
- ตัวเลือกอื่นๆของโรงแรม



รูปที่ 17-11 การค้นหาโรงแรม

เมื่อกดปุ่ม **show me hotel** จะแสดงโรงแรมที่เป็นไปได้ โดยที่หน้าจอนี้สามารถที่จะเปลี่ยนแปลงการค้นหาได้ และหยิบรายการนั้นใส่รถเข็นโดยกดปุ่ม **add to cart**

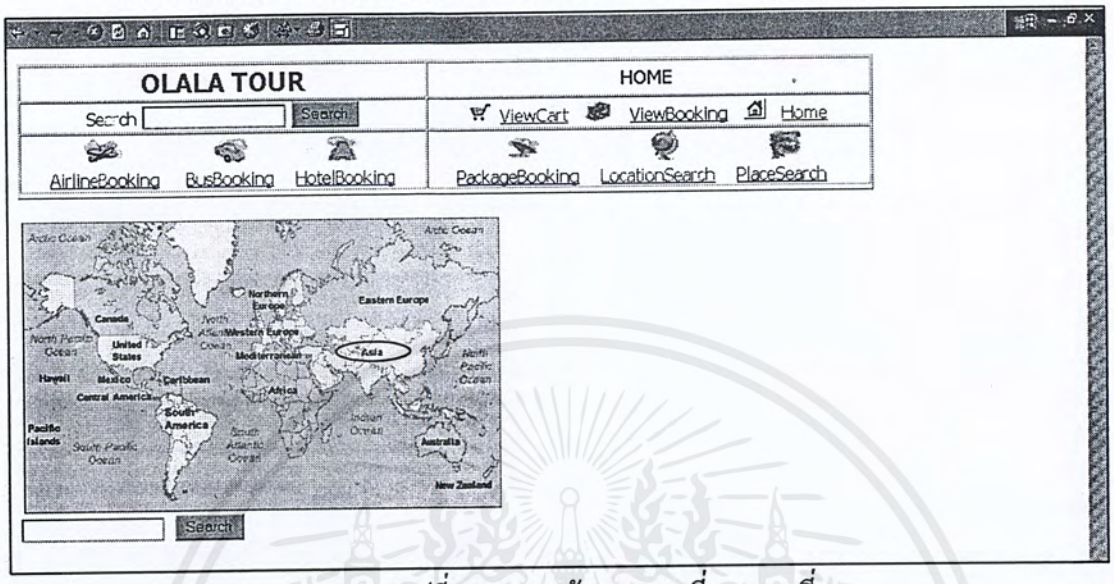


รูปที่ 17-12 การแสดงผลการค้นหาโรงแรมพร้อมใส่จำนวนห้อง

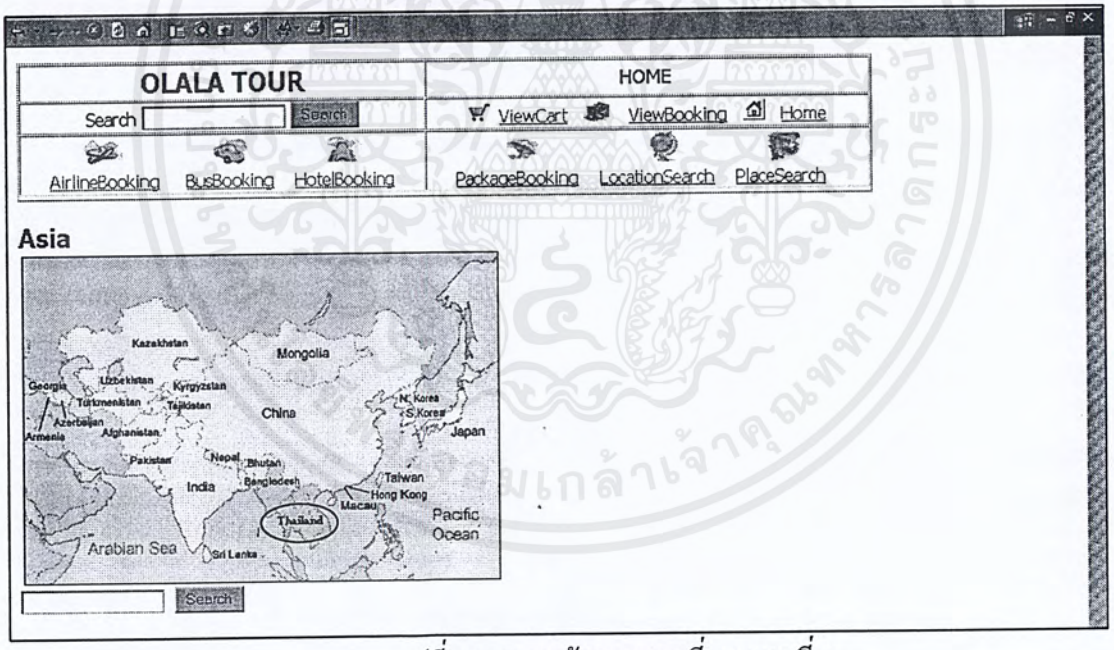
4. การค้นหาสถานที่

- Location Search

สามารถเลือกที่ชื่อประเทศหรือจะใส่ข้อมูลในการค้นหาแล้วกดปุ่ม **search**

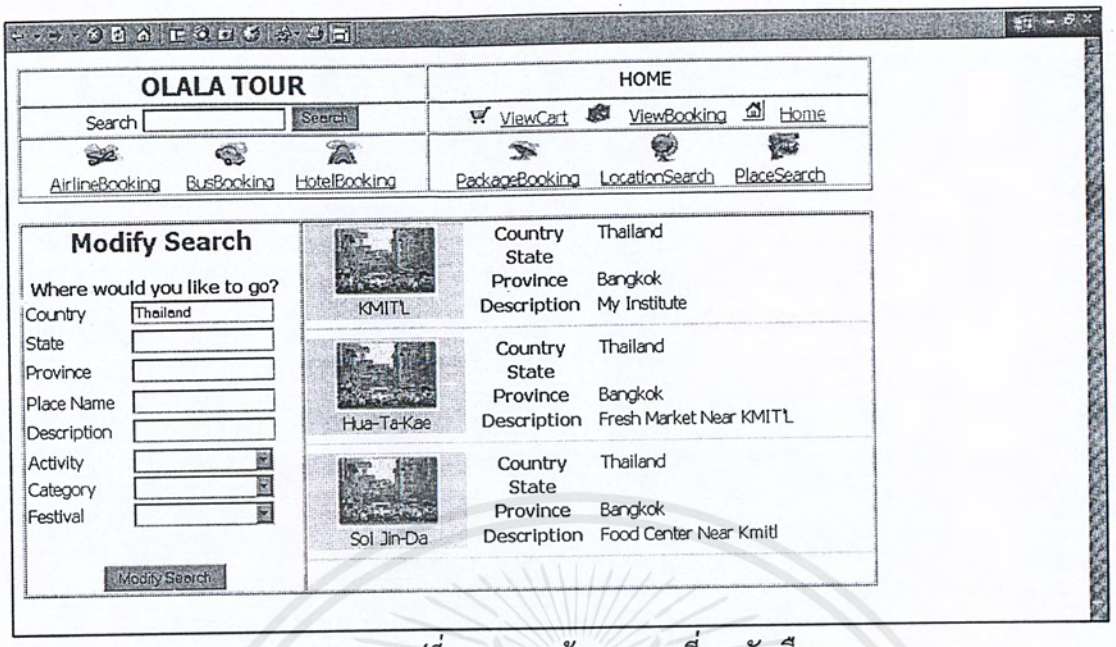


รูปที่ 17-13 การค้นหาสถานที่จากแผนที่ 1



รูปที่ 17-14 การค้นหาสถานที่จากแผนที่ 2

เมื่อกดเลือกที่ Asia และ Thailand จะแสดงข้อมูลในการค้นหาเหมือนกับการค้นหาจากPlace Search ที่ใส่ชื่อประเทศเป็น Thailand



รูปที่ 17-15 การค้นหาสถานที่จากตัวเลือก

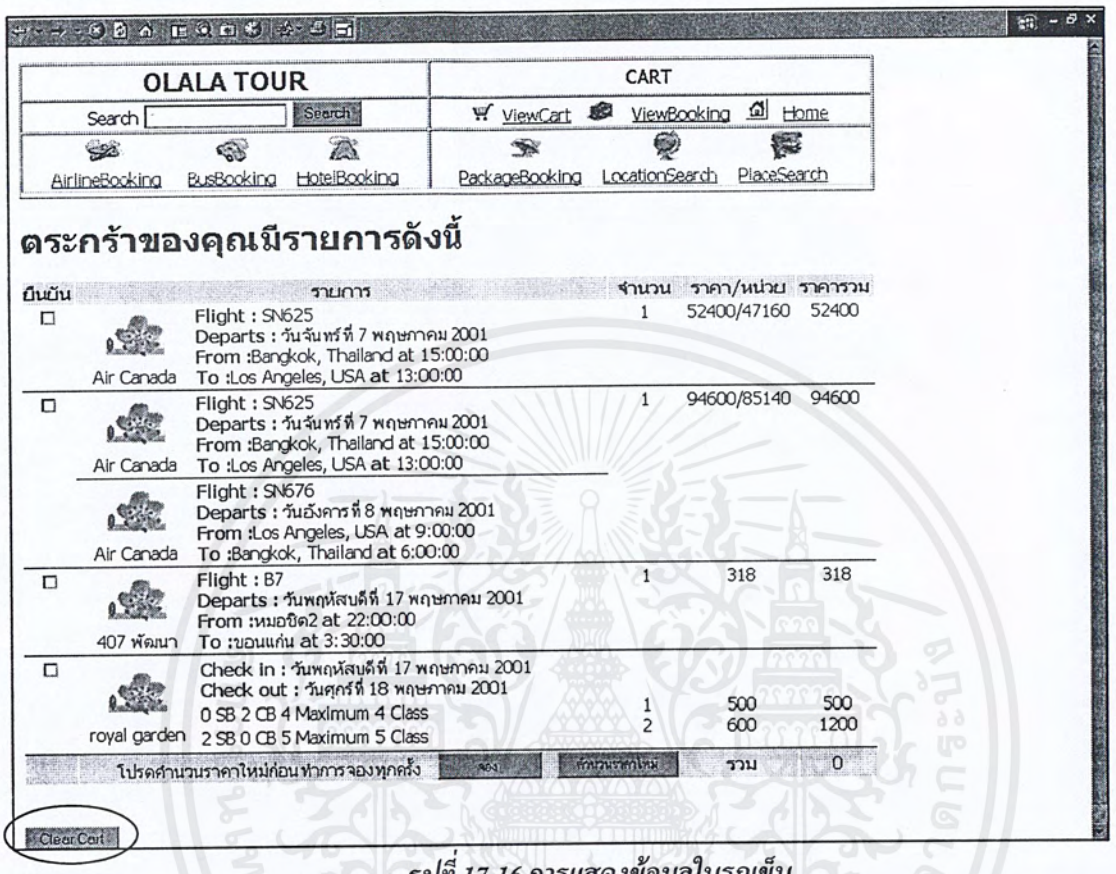
- Place Search

การค้นหาสถานที่มีข้อมูลที่ใส่ในการค้นหาคือ

- ประเทศ รัฐ และจังหวัด
- ชื่อสถานที่
- กิจกรรมที่สถานที่นั้นมี
- ประเภทของสถานที่นั้น
- งานเทศกาล

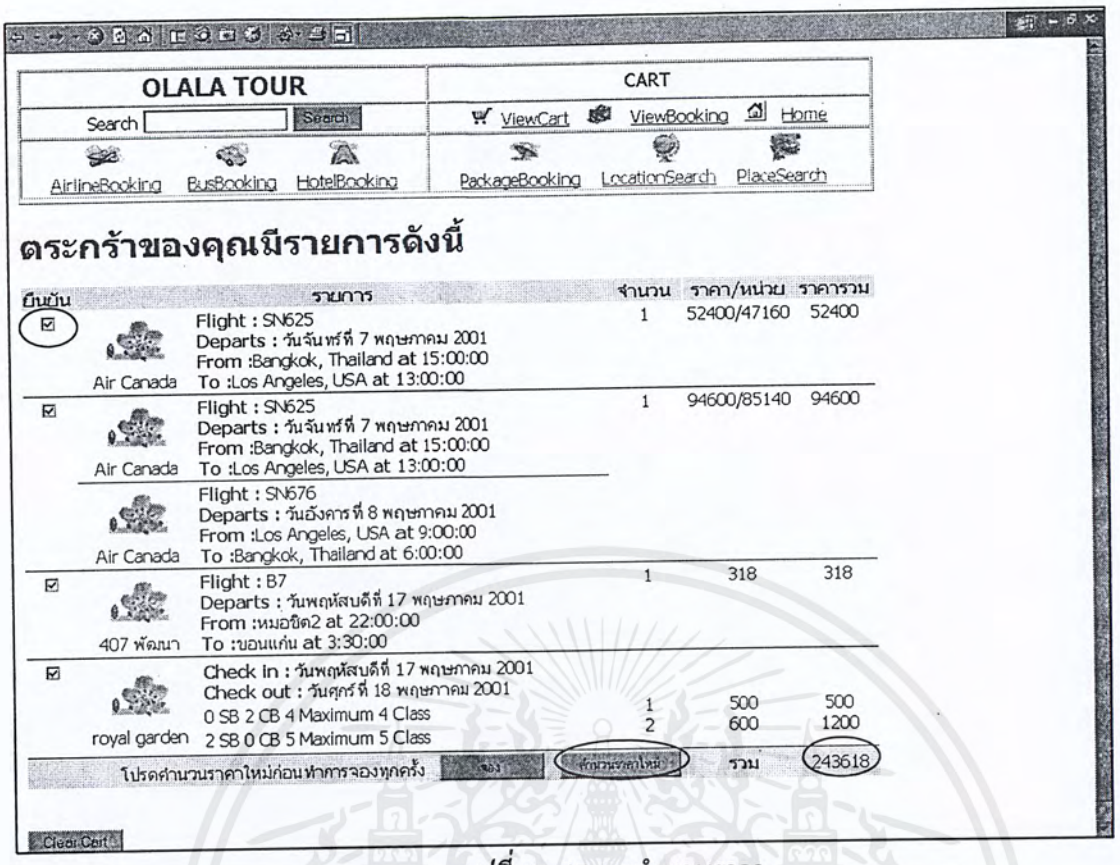
17.1.4 การดูสินค้าในรถเข็น

กดที่ปุ่ม **View Cart** จะเป็นการแสดงรายการที่เราทำการเลือกสินค้าใส่รถเข็นเอาไว้ โดยจะแสดงรายละเอียดของรายการนั้น จำนวน ราคาต่อหน่วย ราคารวมของรายการนั้น



รูปที่ 17-16 การแสดงข้อมูลในรถเข็น

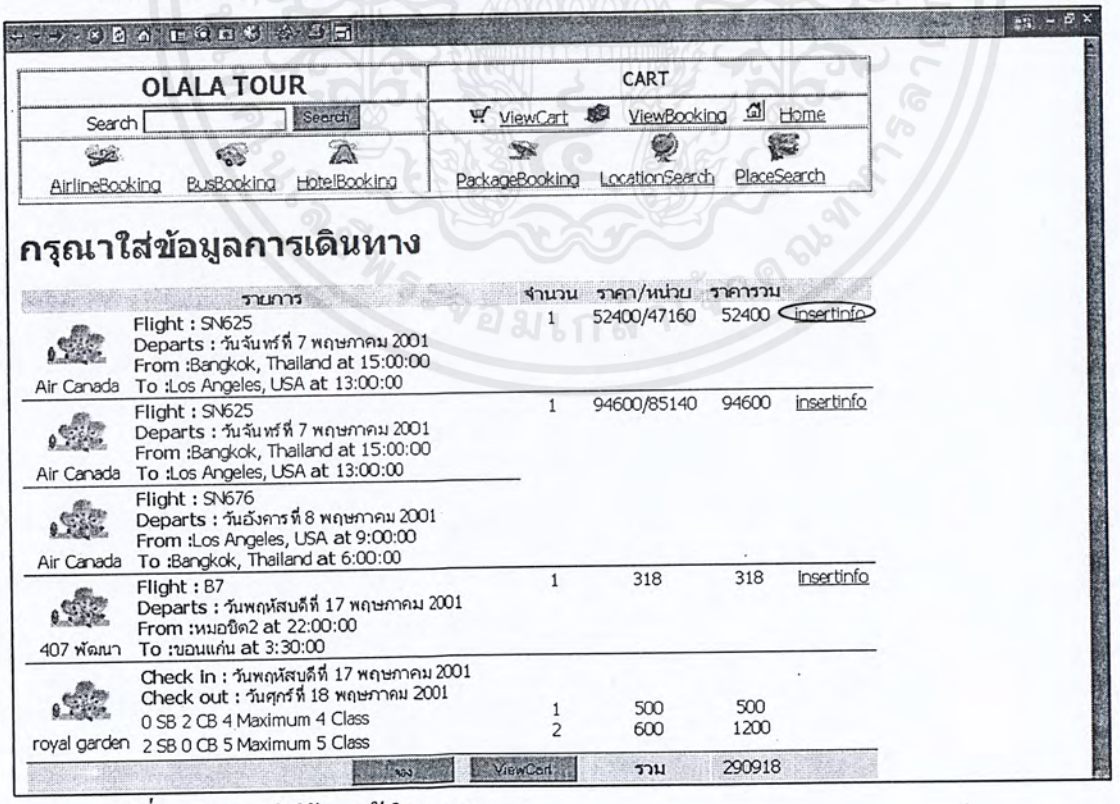
เมื่อจะทำการจองต้องยืนยันและคำนวนราคาก่อนจะได้ราคารวมทั้งหมดแล้วจึงทำการจองต่อไป ซึ่งถ้าไม่ต้องการรายการที่อยู่ในรถเข็นก็ไม่ต้องทำการยืนยัน หรือสามารถที่จะลบรายการทั้งหมดที่อยู่ในรถเข็นโดยการกดปุ่ม **Clear Cart**



รูปที่ 17- 17 การคำนวนราคา

17.1.5 การซื้อหรือจองสินค้าจากรถเข็น

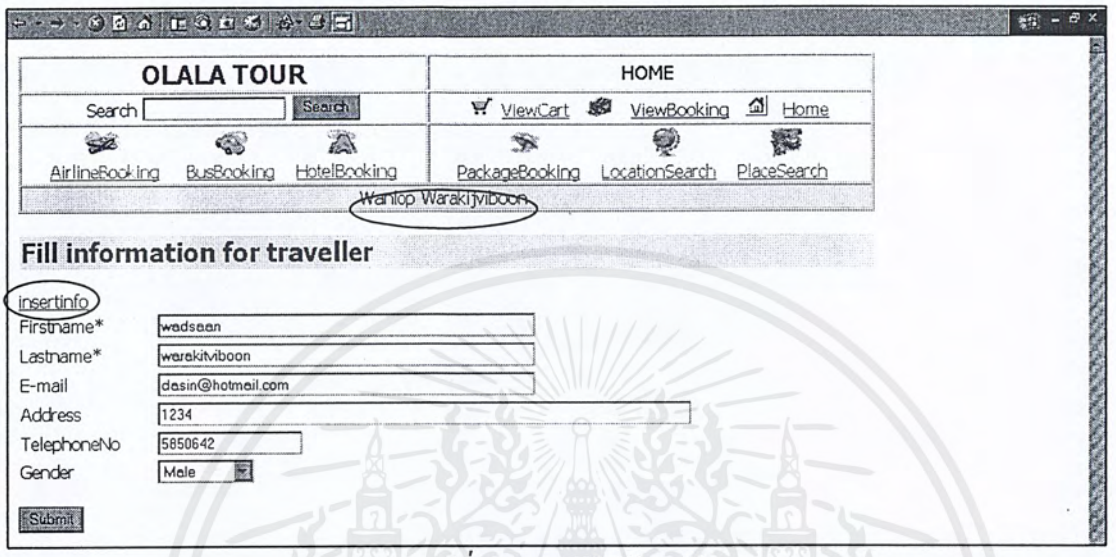
เมื่อต้องการซื้อหรือจองสินค้าก็กดปุ่ม จอง ที่หน้า View Cart



รูปที่ 17- 18 การใส่ข้อมูลผู้เดินทาง 1

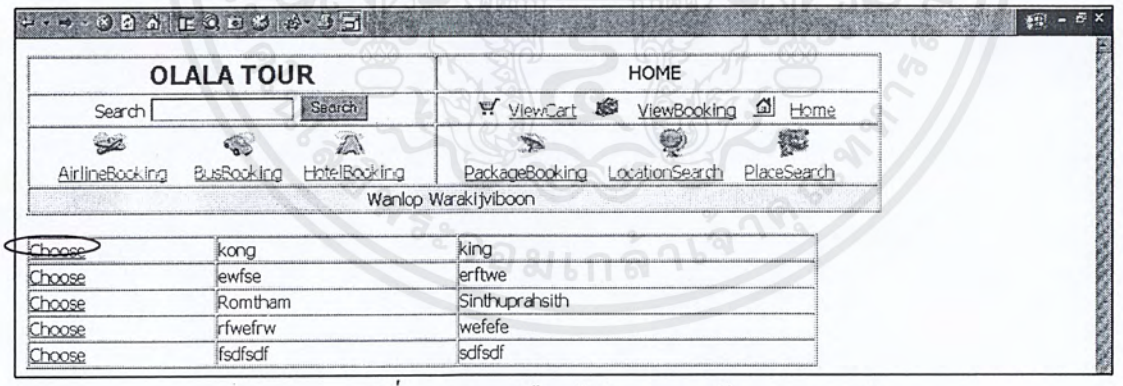
ทำการใส่ข้อมูลของผู้เดินทางได้โดยการกดปุ่ม **insertinfo**

ถ้ายังไม่ทำการเข้าสู่ระบบจะมีหน้าจอให้ทำการเข้าสู่ระบบหรือทำการสมัครสมาชิกใหม่และจะไปยังหน้าจอในกรอกข้อมูลสำหรับผู้เดินทางโดยที่การจองโรงแรมไม่จำเป็นต้องใส่ข้อมูลในส่วนนี้เพราะสามารถอ้างอิงได้จากข้อมูลของสมาชิก

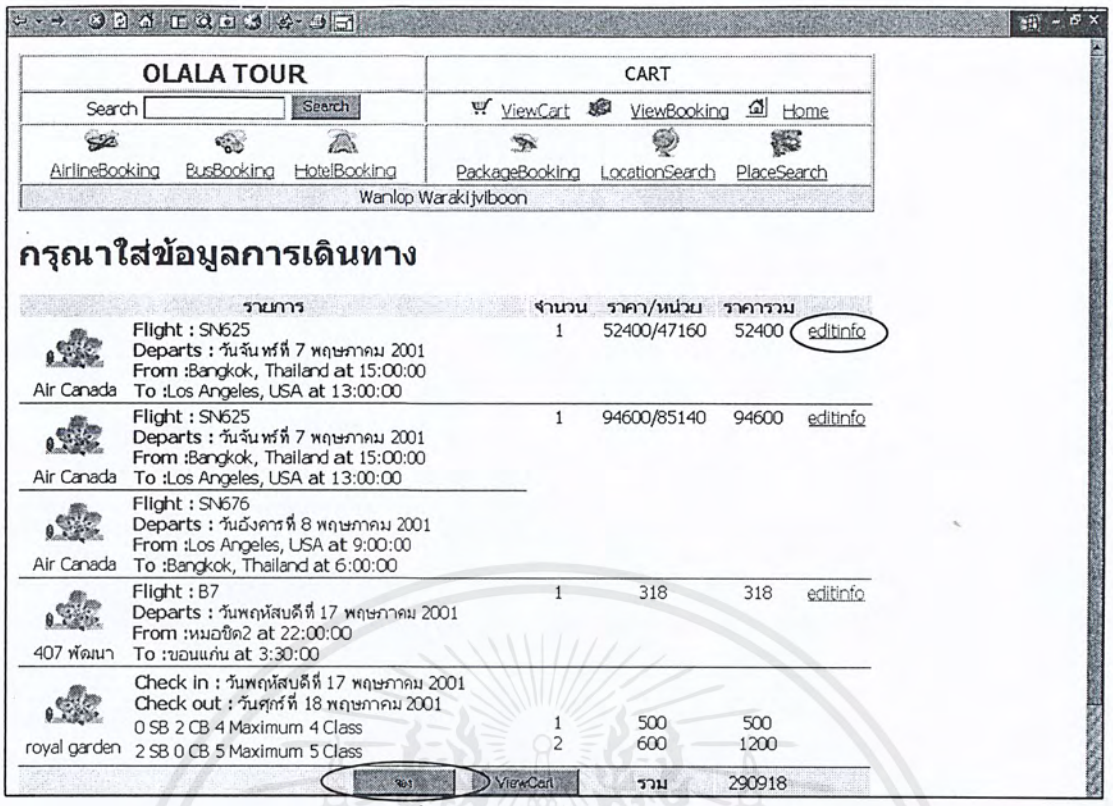


รูปที่ 17- 19 การใส่ข้อมูลผู้เดินทาง 2

สามารถที่จะเลือกผู้เดินทางที่อยู่ในข้อมูลของสมาชิกได้โดยกดที่ปุ่ม **insertinfo** จะมีรายชื่อของคนที่เคยเดินทางโดยใช้ชื่อผู้จองเป็นเรามาให้เลือกและทำการ **Submit** เป็นอันใส่ข้อมูลสำหรับผู้เดินทางเรียบร้อยและจะต้องใส่ข้อมูลสำหรับผู้เดินทางให้ครบทุกรายการ

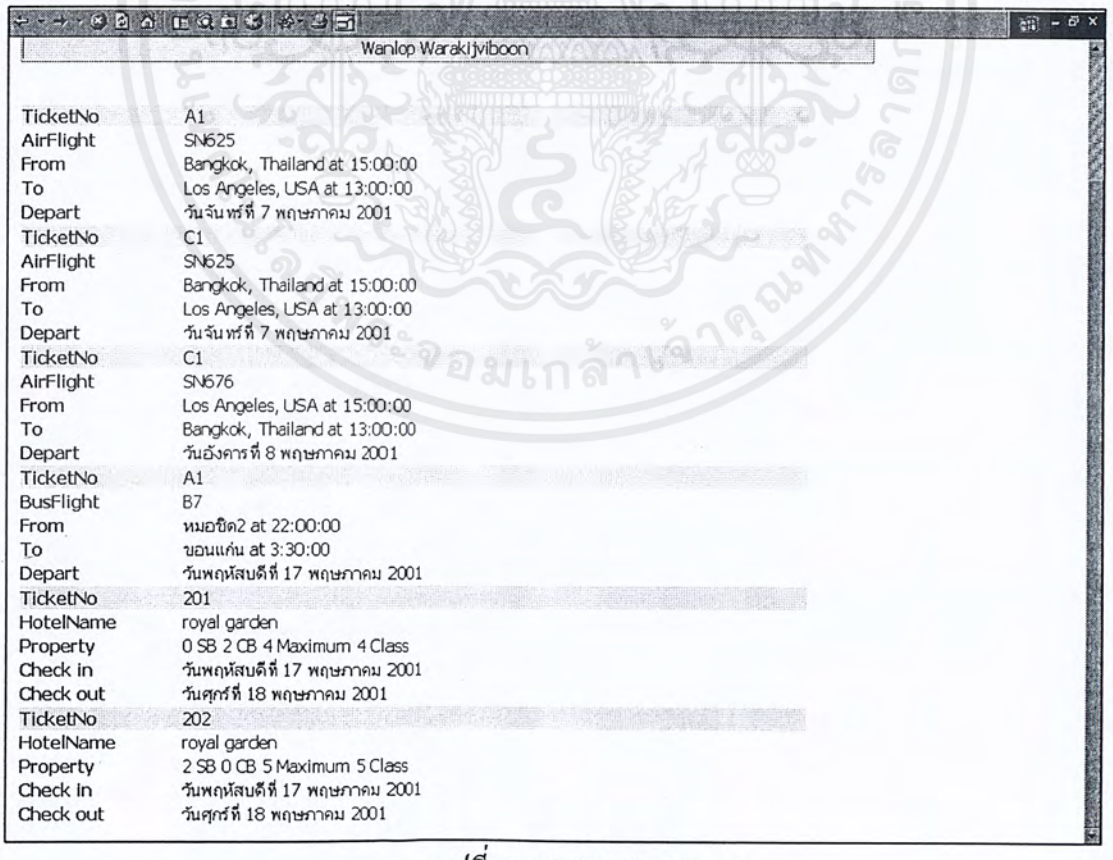


รูปที่ 17- 20 การเลือกผู้เดินทางจากข้อมูลของสมาชิก



รูปที่ 17-21 การแก้ไขข้อมูลของผู้เดินทาง

เมื่อใส่ข้อมูลสำหรับผู้เดินทางแล้วสามารถทำการแก้ไขข้อมูลสำหรับผู้เดินทางโดยกดปุ่ม **editinfo** เมื่อทำครบทุกรายการแล้วก็กดปุ่ม **จอง** อีกครั้ง

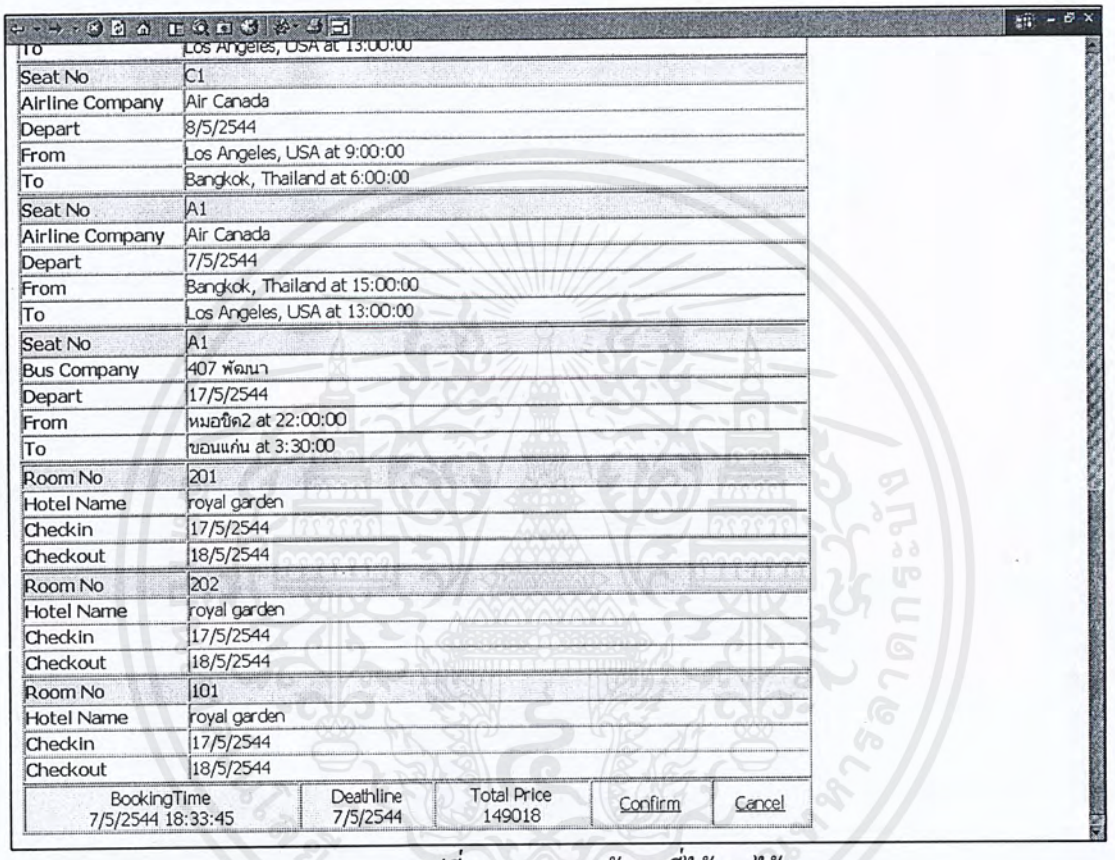


รูปที่ 17-22 แสดงผลการจอง

เมื่อทำการจองเรียบร้อยแล้วจะแสดงหมายเลขที่นั่งหรือหมายเลขห้องที่ได้

17.1.6 การชำระเงิน

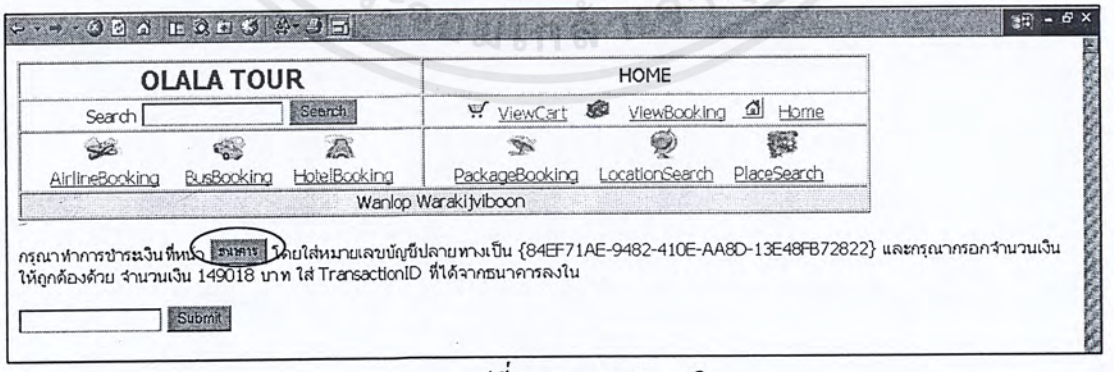
ระบบจะเก็บรายการต่างๆที่ได้จองเอาไว้อยู่ในข้อมูลของสมาชิกเมื่อจะทำการชำระเงินต้องทำการเข้าสู่ระบบก่อนโดยกดปุ่ม View Booking จะแสดงรายการพร้อมทั้งรายละเอียดต่างๆที่ได้จองเอาไว้ในชื่อของสมาชิกนั้น สามารถทำการชำระเงินได้โดยการกด **Confirm** และทำการยกเลิกการจองนั้นได้โดยการกด **Cancel**



To	Los Angeles, USA at 13:00:00
Seat No	C1
Airline Company	Air Canada
Depart	8/5/2544
From	Los Angeles, USA at 9:00:00
To	Bangkok, Thailand at 6:00:00
Seat No	A1
Airline Company	Air Canada
Depart	7/5/2544
From	Bangkok, Thailand at 15:00:00
To	Los Angeles, USA at 13:00:00
Seat No	A1
Bus Company	407 พัฒนา
Depart	17/5/2544
From	หมอชิด2 at 22:00:00
To	ขอนแก่น at 3:30:00
Room No	201
Hotel Name	royal garden
Checkin	17/5/2544
Checkout	18/5/2544
Room No	202
Hotel Name	royal garden
Checkin	17/5/2544
Checkout	18/5/2544
Room No	101
Hotel Name	royal garden
Checkin	17/5/2544
Checkout	18/5/2544

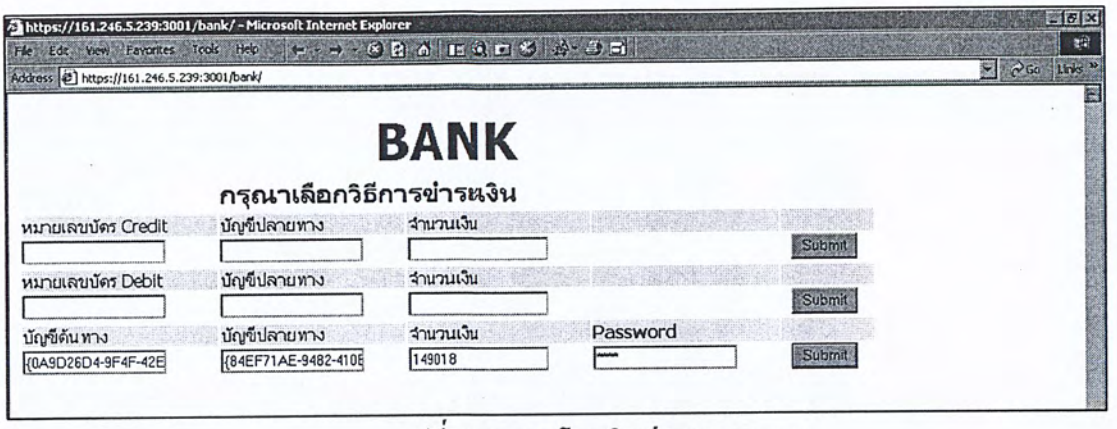
BookingTime	Deathline	Total Price	Confirm	Cancel
7/5/2544 18:33:45	7/5/2544	149018		

รูปที่ 17- 23 การดูข้อมูลที่ได้จองไว้



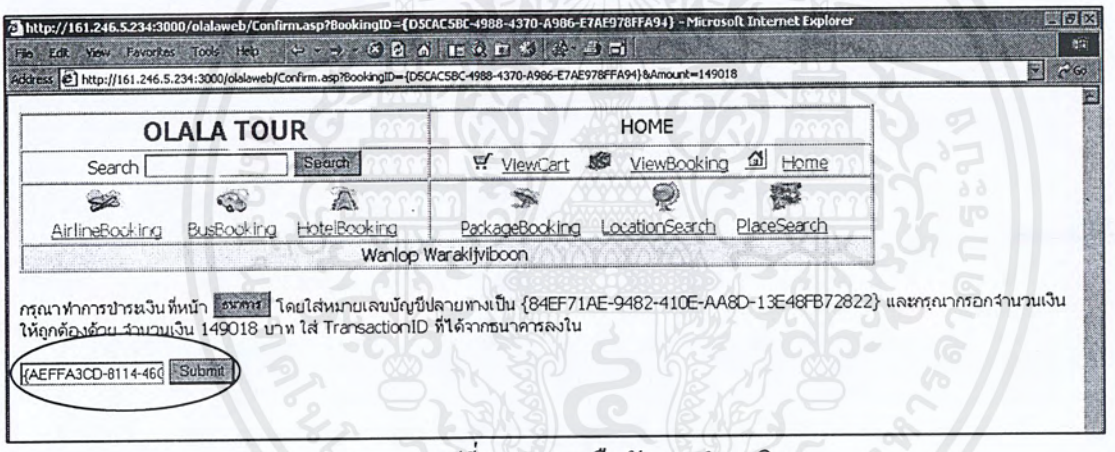
รูปที่ 17- 24 การชำระเงิน

เมื่อกด **Confirm** แสดงหน้าจอให้ไปที่ธนาคารเพื่อทำการโอนเงินเมื่อโอนเงินเรียบร้อยจะได้ TransactionID จากหน้าธนาคารนำมาใส่ใน **textbox** และกดปุ่ม **submit**



รูปที่ 17-25 การโอนเงินผ่านทางธนาคาร

ที่หน้าจอของธนาคารจะมีให้เลือกวิธีการชำระเงินโดยจะต้องใส่บัญชีปลายทางเป็น {84EF71AE-9482-410E-AA8D-13E48FB72822} ใส่จำนวนเงินให้ตรงกับที่ระบบแจ้งให้ทราบและกดปุ่ม **submit** ธนาคารจะให้หมายเลขมาหนึ่งชุดนำกลับไปใส่ที่หน้าจอของระบบ Olala Tourและกด **Submit** ก็จะทำการชำระเงินเรียบร้อย

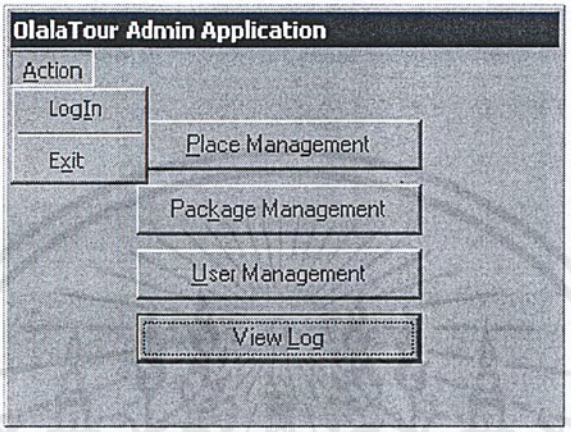


รูปที่ 17-26 การยืนยันการชำระเงิน

17.2 Admin application

เป็นแอปพลิเคชั่นสำหรับ ผู้ดูแลระบบในการจัดการเกี่ยวกับสถานที่ท่องเที่ยว, แพคเกจท่องเที่ยว, การจัดการกับผู้ใช้ของระบบ, การดูประวัติการทำงานของระบบ

17.2.1 เข้าสู่การใช้งาน Admin application



รูปที่ 17-27 แสดงการเริ่มทำงาน

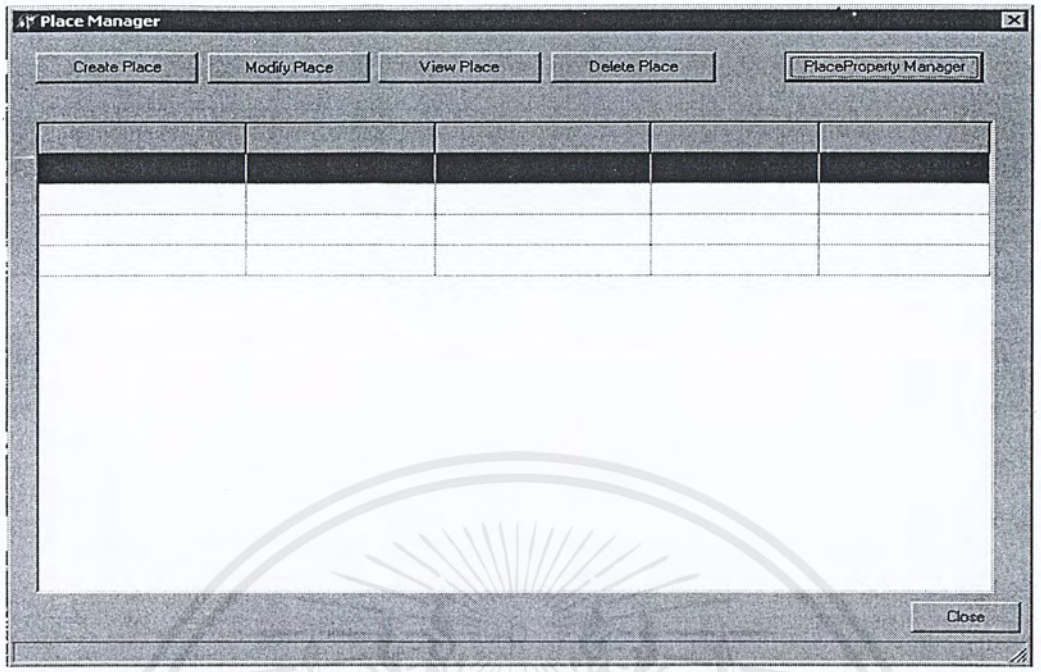
ทำการ Login เข้าสู่ระบบดังรูป

รูปที่ 17-28 แสดงการ Login

17.2.2 การจัดการเกี่ยวกับสถานที่ท่องเที่ยว

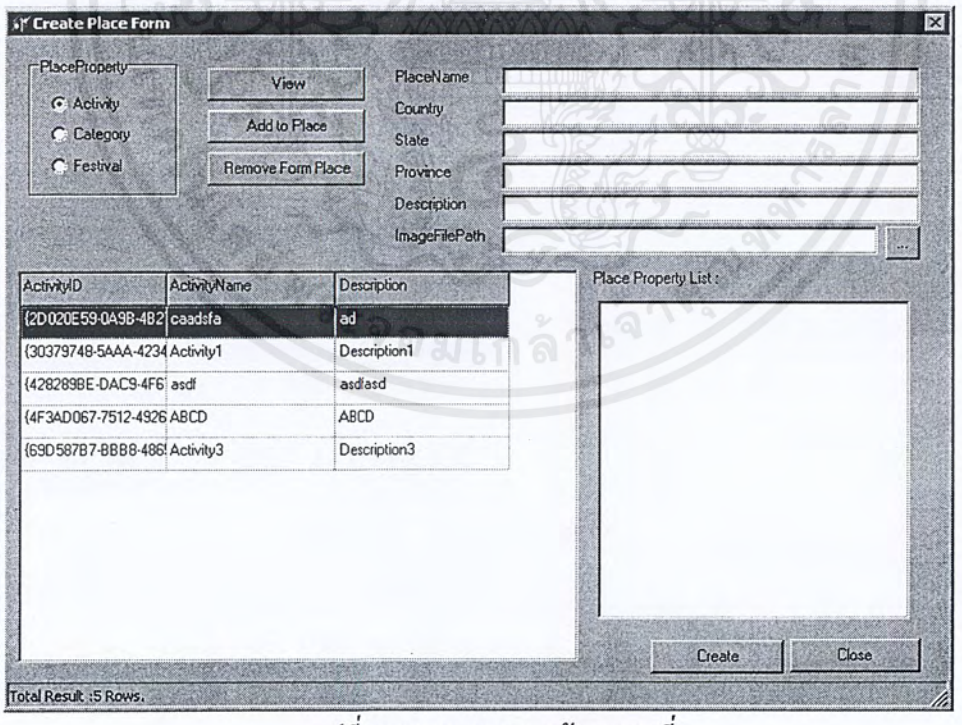
โดยการจัดการกับสถานที่เที่ยวสามารถทำการ สร้าง, แก้ไข, ดูข้อมูล, ลบ, จัดการคุณสมบัติของสถานที่ ซึ่งการทำงานจะอธิบาย ดังนี้

เริ่มเข้าสู่การจัดการสถานที่โดย Admin application ทำการเลือก Place Management



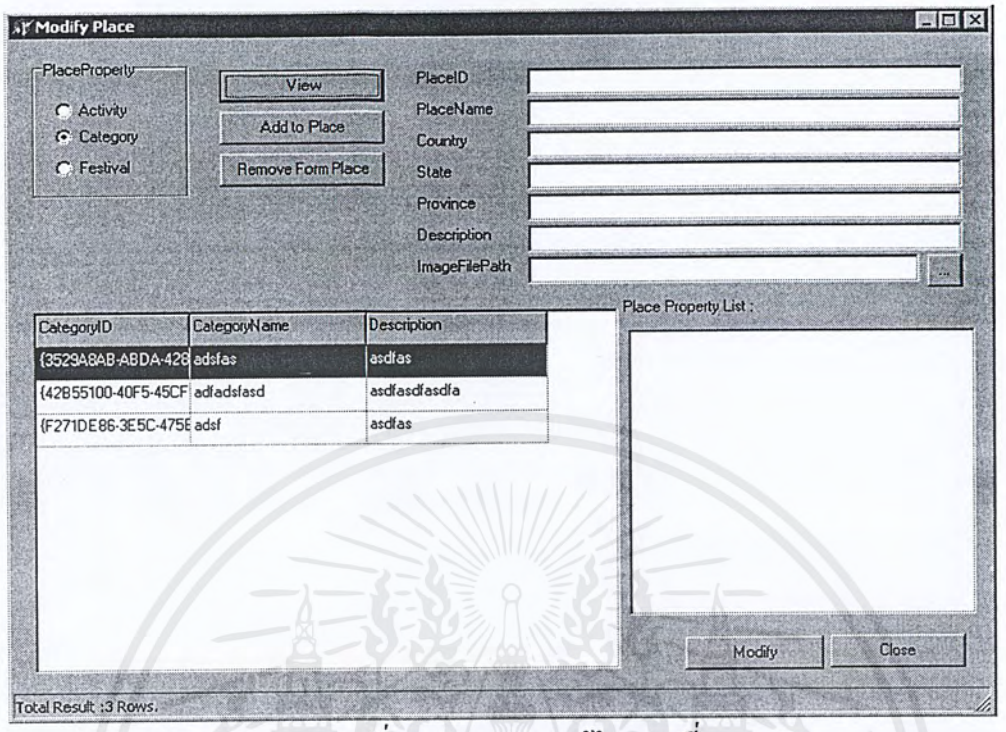
รูปที่ 17-29 แสดงหน้าจอการจัดการสถานที่

หลังจากเข้าสู่การจัดการสถานที่แล้วสามารถทำการ สร้างสถานที่ให้ระบบรู้จัก โดยกดปุ่ม Create place



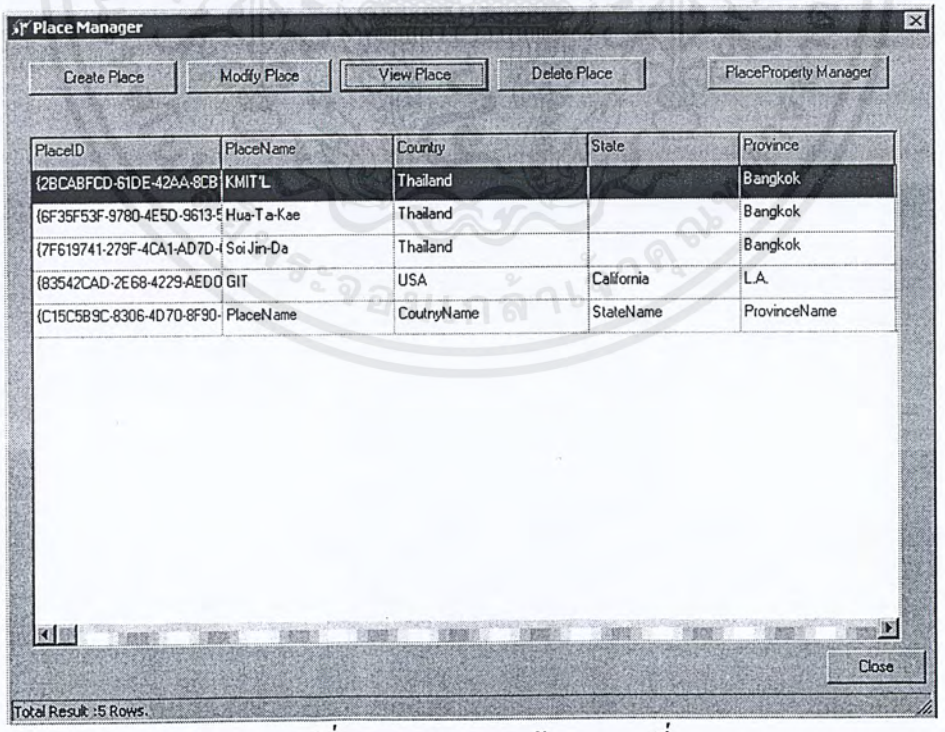
รูปที่ 17-30 แสดงการสร้างสถานที่

การทำการแก้ไขสถานที่ทำได้โดยการกดปุ่ม Modify place จะได้หน้าจอดังรูป



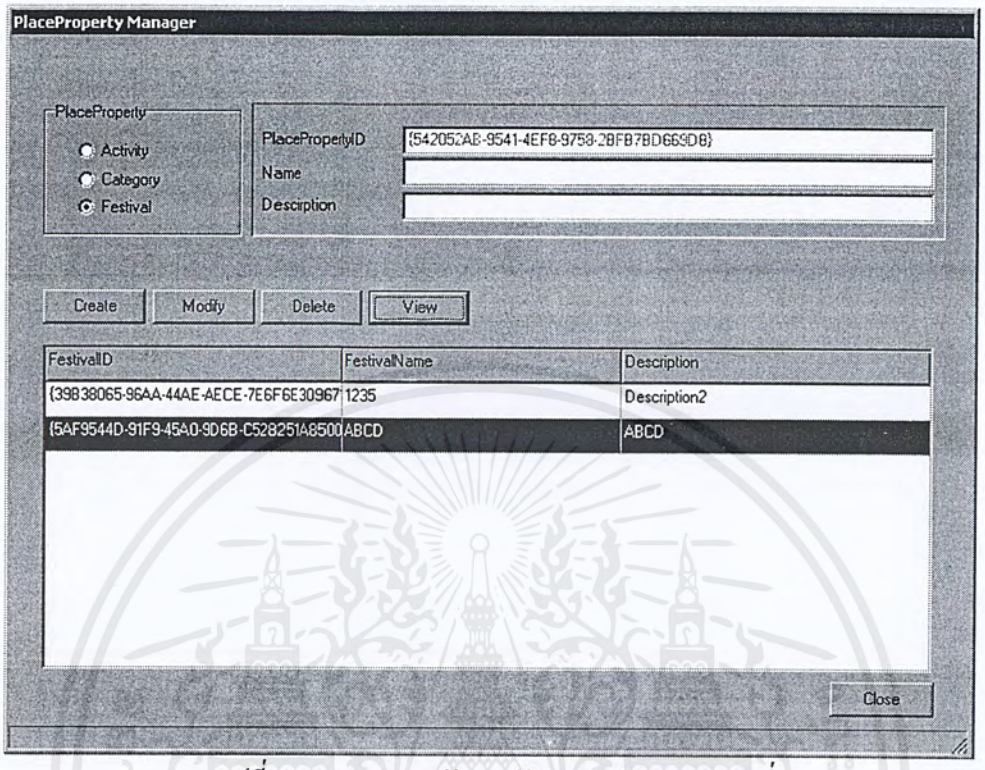
รูปที่ 17-31 แสดงการแก้ไขสถานที่

เมื่อแก้ไขเรียบร้อยแล้วสามารถดูข้อมูลของสถานที่ได้โดยกดปุ่ม View place



รูปที่ 17-32 แสดงการดูข้อมูลสถานที่

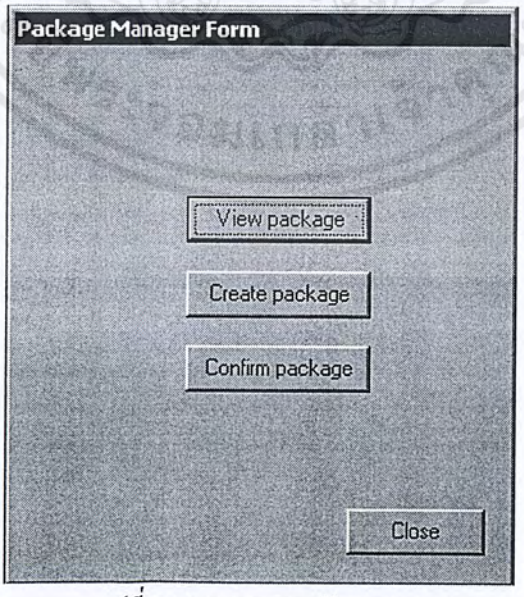
สามารถทำการลบสถานที่ออกจากระบบ โดยกดปุ่ม Delete place
และทำการจัดการกับคุณสมบัติของสถานที่ได้โดยกดปุ่ม PlaceProperty Manager



รูปที่ 17-33 แสดงการจัดการกับคุณสมบัติของสถานที่

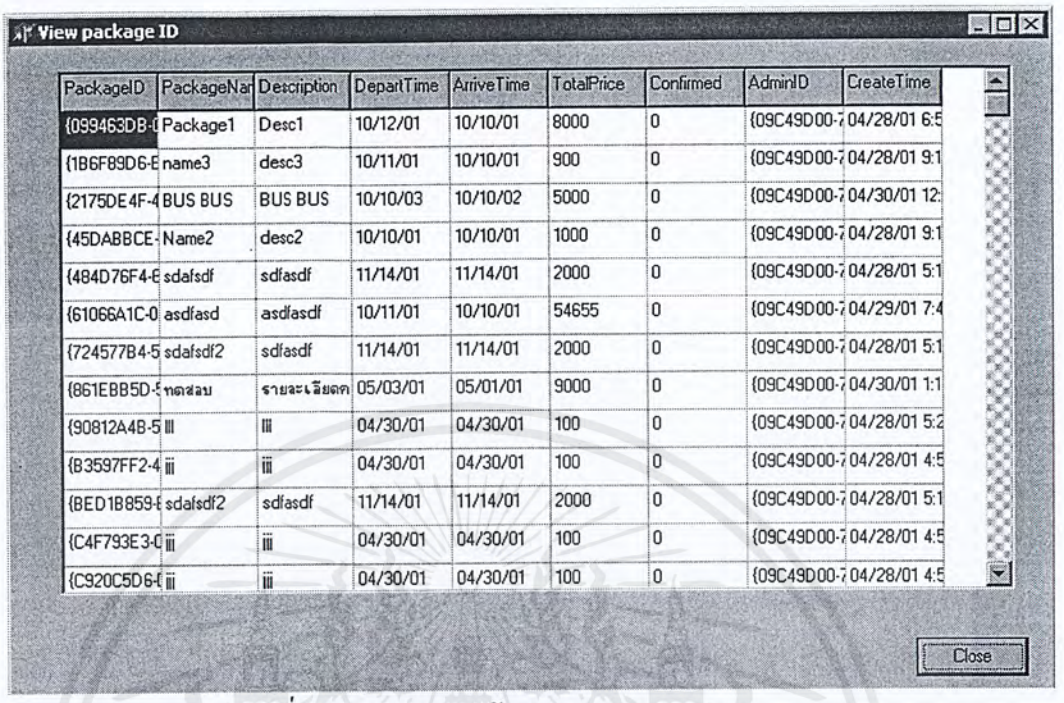
17.2.3 การจัดการแพคเกจการท่องเที่ยว

สามารถทำการสร้าง, ดูข้อมูลการจอง, ตกลงหรือปฏิเสธการจองได้ซึ่งการทำงานจะอธิบาย ดังนี้
ที่หน้าจอแรกของ Admin application ทำการเลือก Package management



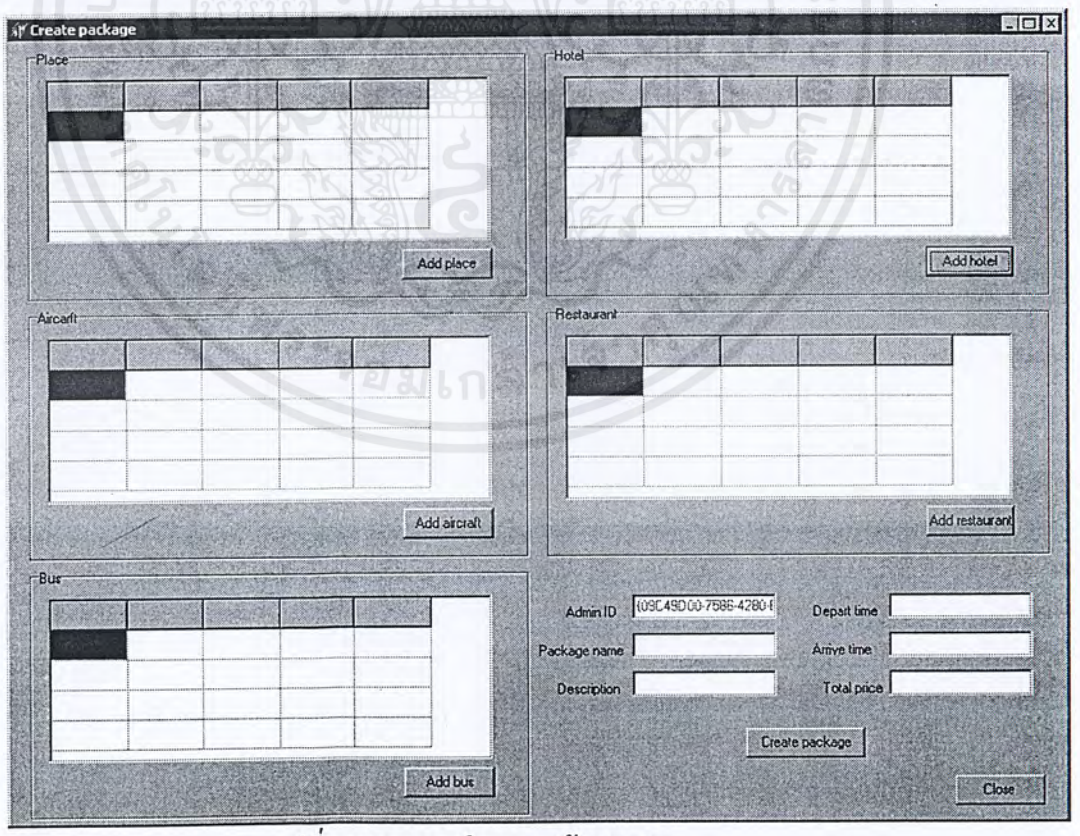
รูปที่ 17-34 แสดงการจัดการแพคเกจ

เลือกการดูแพคเกจโดยกดปุ่ม View package



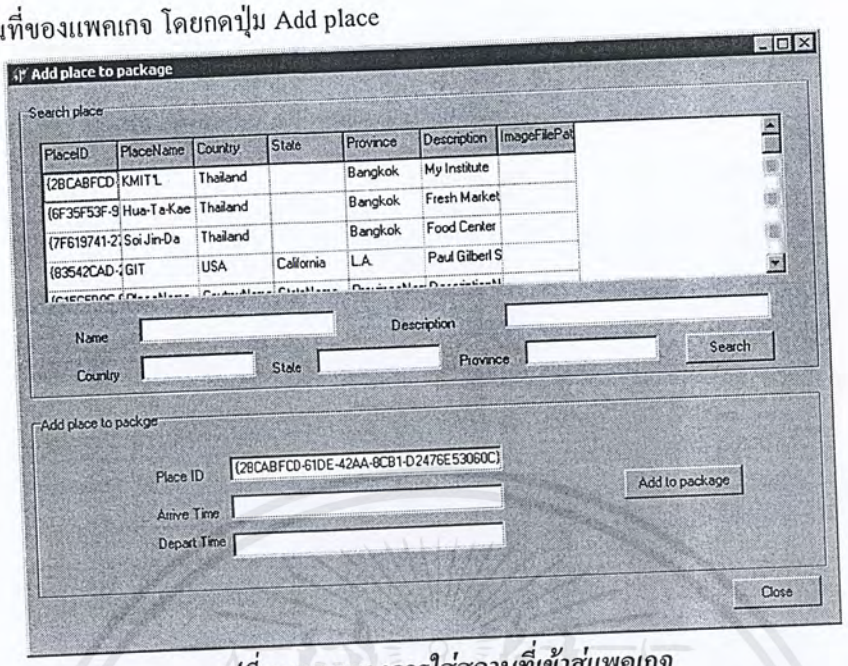
รูปที่ 17-35 แสดงการดูข้อมูลการจองแพคเกจ

ทำการสร้างแพคเกจโดยการเลือก Create package จะเห็นหน้าจอดังรูป



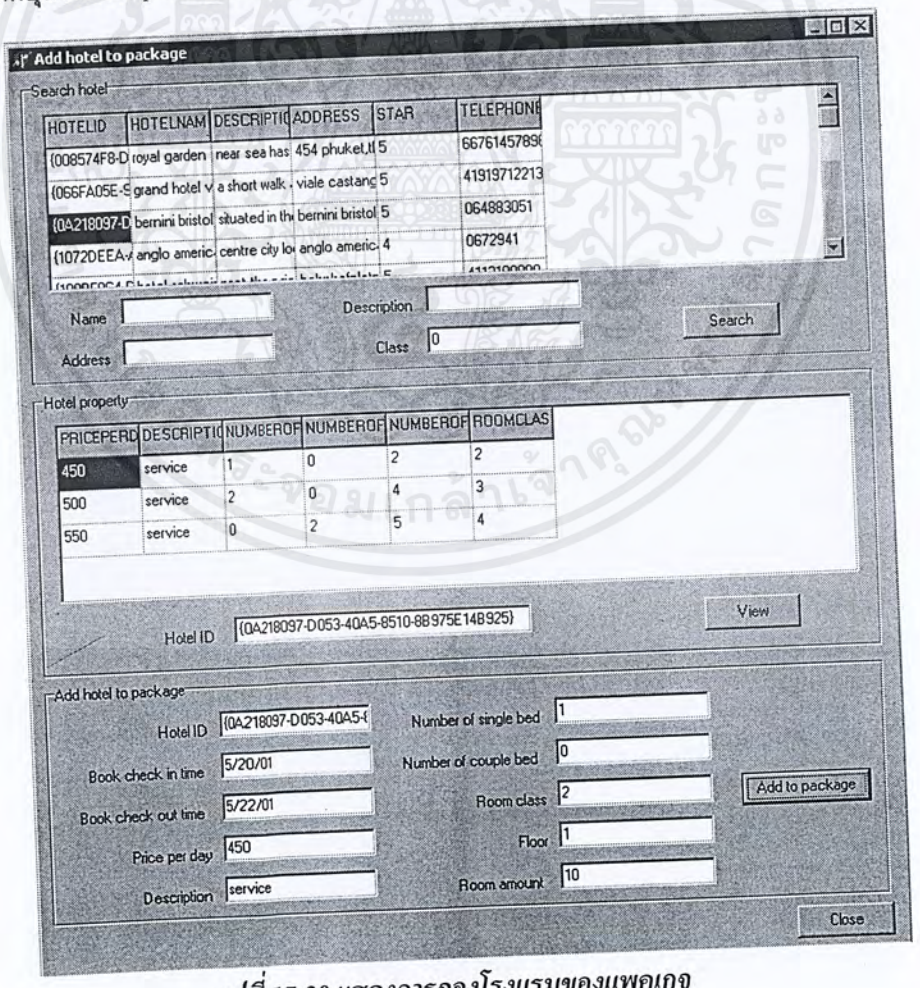
รูปที่ 17-36 แสดงก่อนการสร้างแพคเกจ

ทำการเลือกสถานที่ของแพคเกจ โดยกดปุ่ม Add place



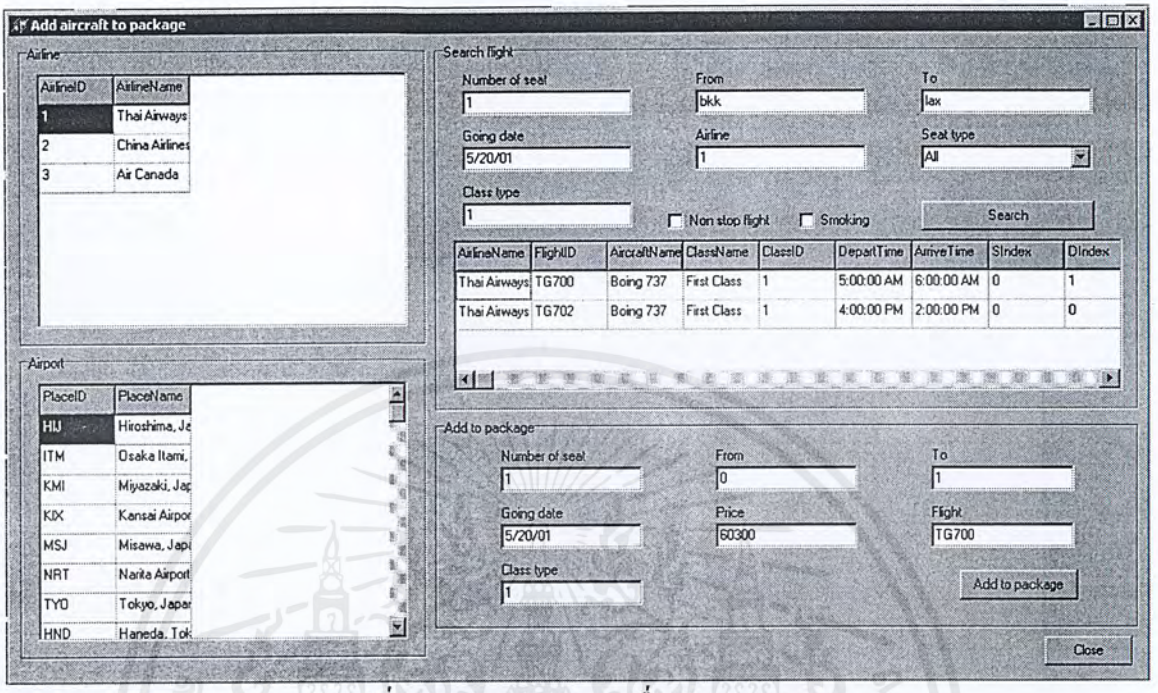
รูปที่ 17-37 แสดงการใส่สถานที่เข้าสู่แพคเกจ

เสร็จแล้วทำการ กดปุ่ม Add to package แล้วปิดหน้าจอนี้กลับไปทำการจองโรงแรมโดยกดปุ่ม Add hotel



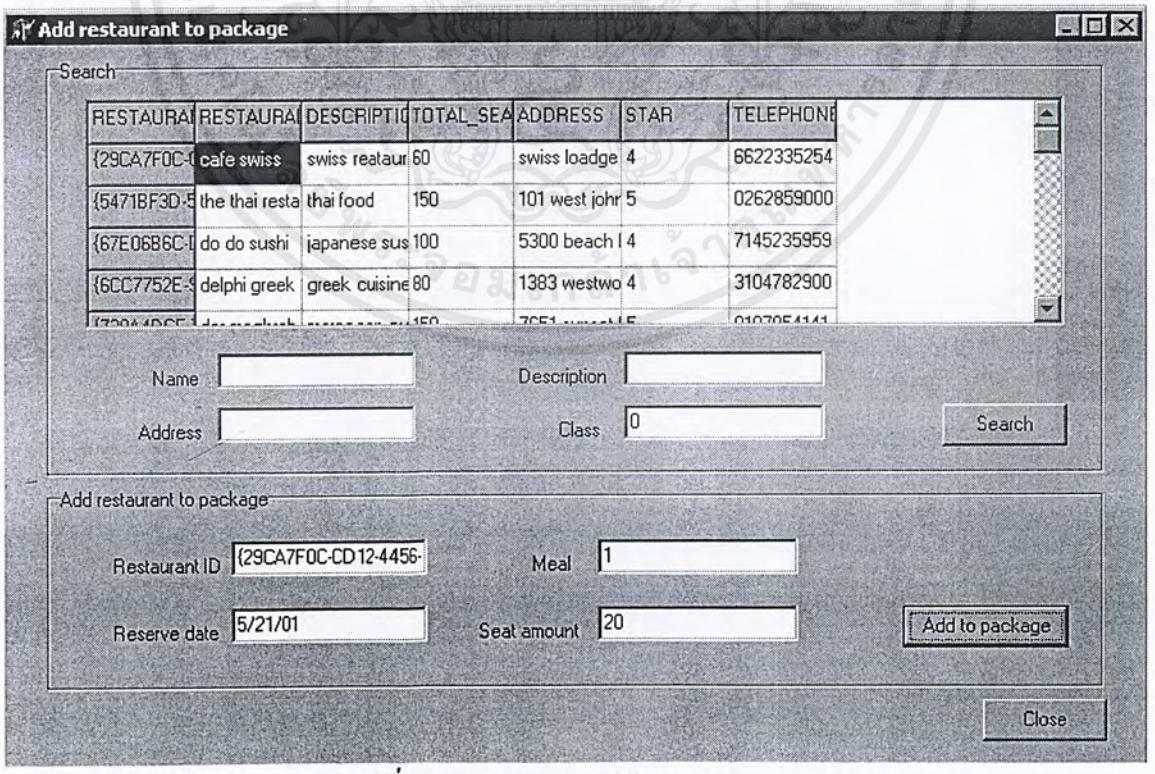
รูปที่ 17-38 แสดงการจองโรงแรมของแพคเกจ

เสร็จแล้วทำการ กดปุ่ม Add to package แล้วปิดหน้าจอนี้กลับไปทำการจองเครื่องบินโดยกดปุ่ม Add aircraft



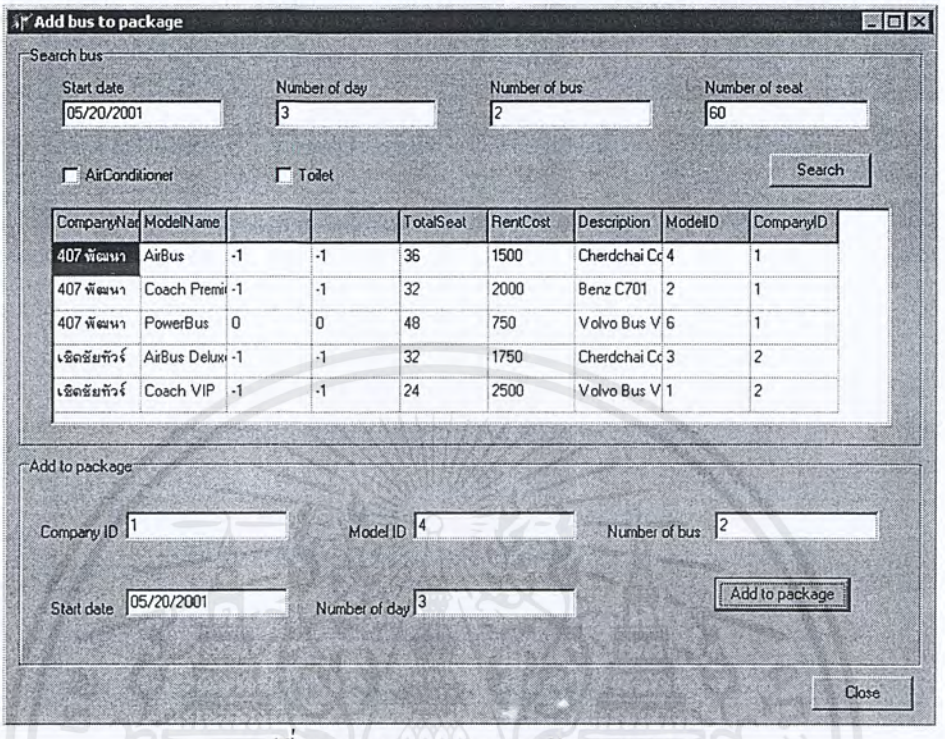
รูปที่ 17-39 แสดงการจองเครื่องบินของแพคเกจ

เสร็จแล้วทำการ กดปุ่ม Add to package แล้วปิดหน้าจอนี้กลับไปทำการจองร้านอาหาร โดยกดปุ่ม Add restaurant



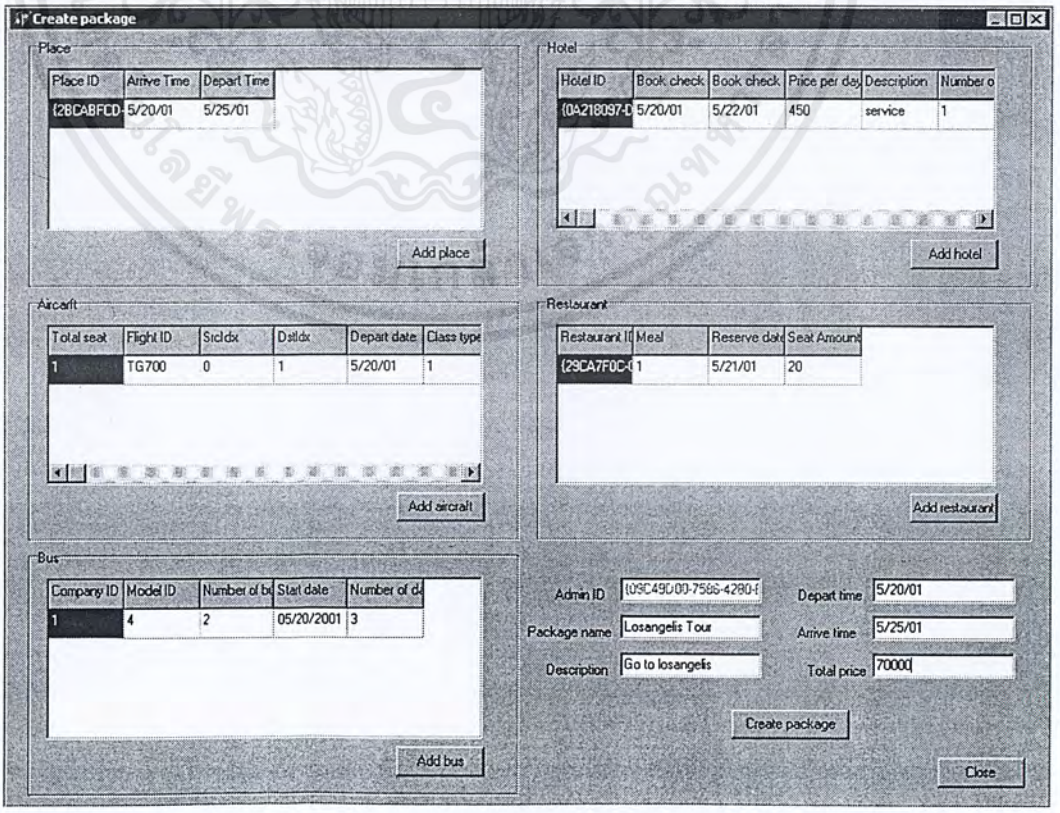
รูปที่ 17-40 แสดงการจองร้านอาหารของแพคเกจ

เสร็จแล้วทำการ กดปุ่ม Add to package แล้วปิดหน้าจอนี้กลับไปทำการจองรถบัสโดยกดปุ่ม Add bus



รูปที่ 17-41 แสดงการจองรถบัสของแพคเกจ

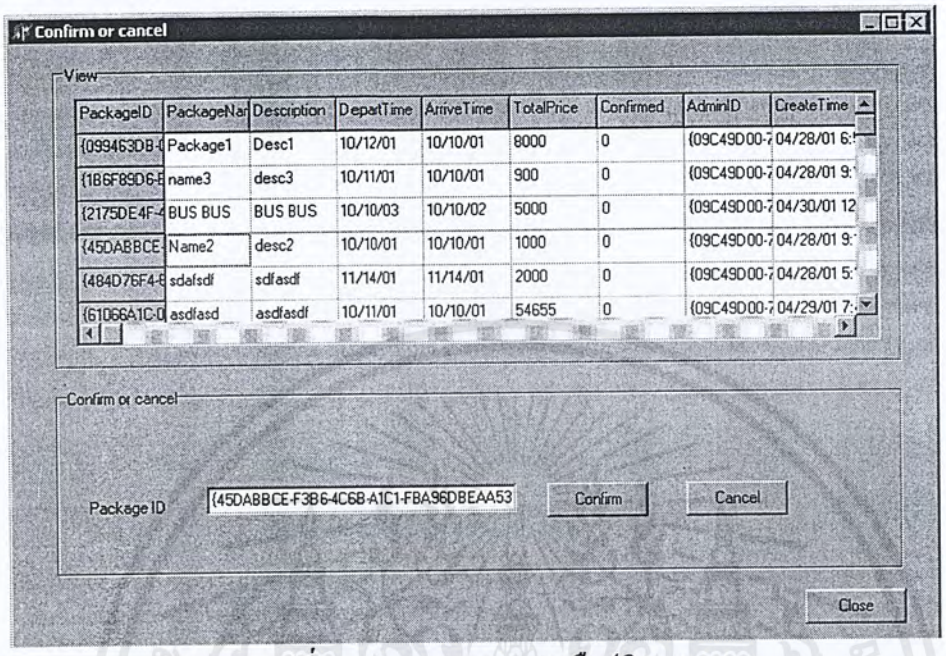
เสร็จแล้วทำการใส่ข้อมูลของแพคเกจให้ครบ จะได้หน้าจอดังนี้



รูปที่ 17-42 แสดงการจองแพคเกจเสร็จ

เสร็จแล้วทำการ Create package เป็นอันเสร็จการสร้างแพคเกจ

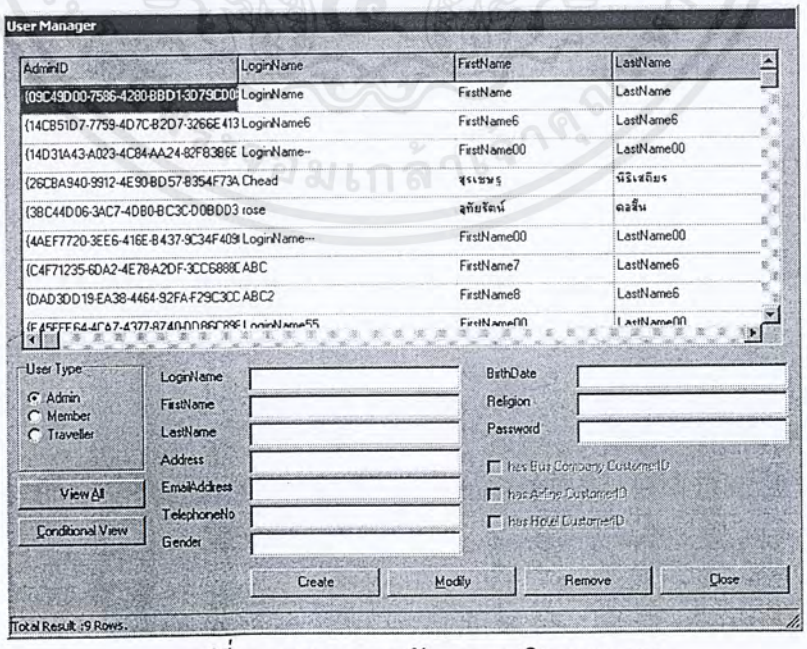
หลังจากการสร้างแพคเกจเสร็จแล้วสามารถกลับมา ตกลงหรือปฏิเสธได้ โดยการเลือก Confirm package ที่หน้าของ Package manager



รูปที่ 17-43 แสดงการตกลงหรือปฏิเสธการจอง

17.2.4 การจัดเกี่ยวกับสมาชิกของระบบ

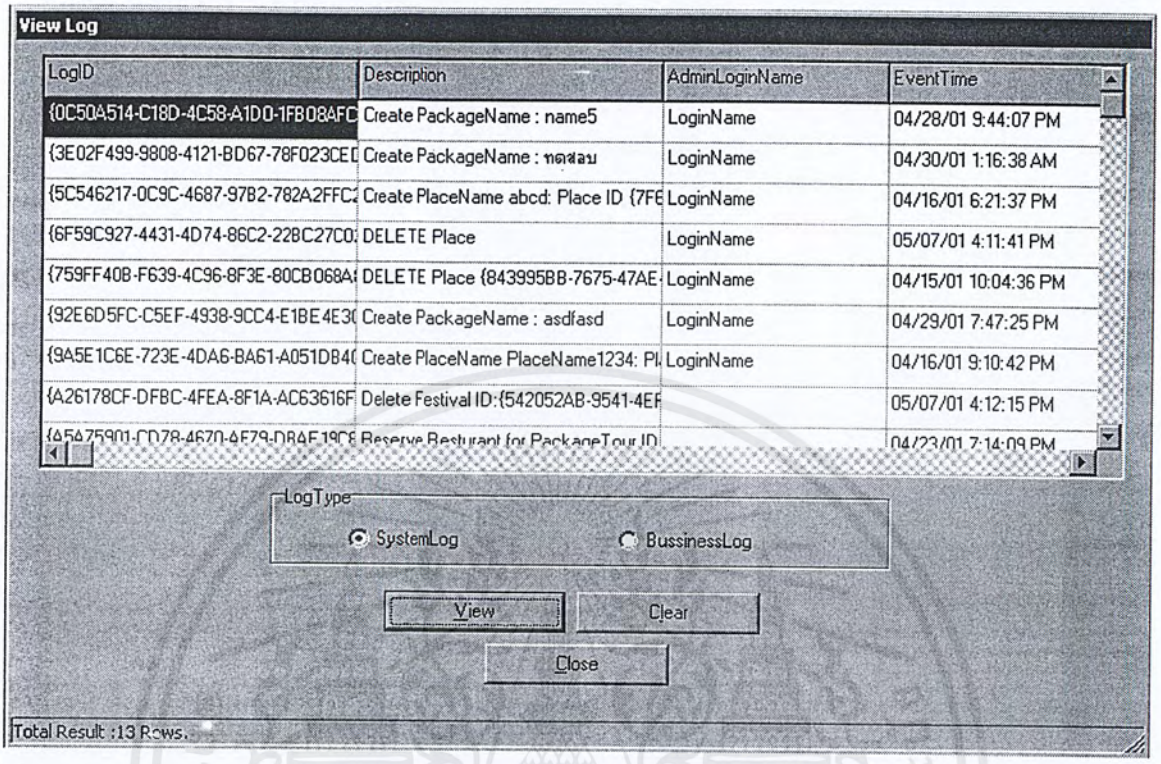
สามารถทำการสร้าง, ลบ, แก้ไข เกี่ยวกับตัวของสมาชิกในระบบได้โดยที่หน้า Admin application ทำการเลือก User management



รูปที่ 17-44 แสดงการจัดการสมาชิกของระบบ

17.2.5 การดูประวัติการทำงานของระบบ

สามารถดูการทำงานที่ทำไปแล้วของระบบได้โดยที่หน้า Admin application ทำการเลือก View



log

รูปที่ 17-45 แสดงการดูประวัติการทำงานของระบบ

สามารถดูได้ทั้ง System log และ Business log

บทที่ 18
เปรียบเทียบประสิทธิภาพระหว่าง COM+ และ EJB

นอกจาก COM+ และ DNA แล้วก็ยังมีซอฟต์แวร์คอมโพเนนท์อื่นอีกซึ่งได้แก่ EJB และ J2E การเปรียบเทียบประสิทธิภาพระหว่าง COM+ และ EJB นั้นเป็นเรื่องที่ค่อนข้างที่จะยุ่งยากในการเปรียบว่าฝ่ายใดมีประสิทธิภาพเหนือกว่าฝ่ายใดอย่างชัดเจน เนื่องจากทั้งสองระบบก็มีข้อด้อย และข้อเด่นที่แตกต่างกันไปซึ่งขึ้นอยู่กับความต้องการของระบบ ได้แก่

- ลักษณะของระบบงาน หากมีความจำเป็นต้องใช้งานกับระบบงานที่หลากหลาย หรือต้องการขายซอฟต์แวร์คอมโพเนนท์ให้กับองค์กรภายนอก หรือองค์กรเหล่านั้นใช้งานระบบปฏิบัติการอื่นนอกจาก Windows 2000 EJB ก็เป็นทางเลือกที่เหมาะสมกว่า COM+ อยู่พอสมควร แต่หากว่าองค์กรนั้นพัฒนาซอฟท์แวร์คอมโพเนนท์ขึ้นเพื่อใช้งานเองภายใต้ระบบปฏิบัติการ Windows 2000 ทั้ง COM+ และ EJB ก็มีความเหมาะสมเสมอกัน
- ความสามารถเดิมของผู้พัฒนา หากผู้พัฒนามีความชำนาญในตัวจาวาแล้ว EJB ก็เป็นทางเลือกที่เหมาะสมกว่า เช่นเดียวกันหากผู้พัฒนามีความชำนาญในตัวภาษาโปรแกรมอื่น COM+ จะเหนือกว่าเล็กน้อย เนื่องจาก EJB นั้นมีข้อจำกัดที่สามารถพัฒนาโดยใช้ภาษาจาวาเท่านั้นซึ่งแตกต่างจาก COM+
- ค่าใช้จ่ายสำหรับเซิร์ฟเวอร์ เนื่องจากสภาพแวดล้อมใน Windows 2000 นั้นเหมาะสมกับการพัฒนา COM+ อยู่แล้วโดยไม่จำเป็นต้องเพิ่มเติมโปรแกรมอื่น หากแต่การพัฒนาด้วย EJB นั้นต้องการคอนเทนเนอร์และเว็บเซิร์ฟเวอร์เพิ่มเติม จากระบบปฏิบัติการ

ซึ่งสามารถสรุปได้ดังตารางที่ 18-1

คุณสมบัติ	COM+	EJB
ภาษาในการพัฒนาคอมโพเนนท์	ทุกภาษาโปรแกรม	จาวา เท่านั้น
แพลตฟอร์ม	Windows 2000	ทุกแพลตฟอร์ม
จำนวนบริษัทที่นำไปใช้ (ข้อมูลปลายปี พศ. 2543)	บริษัทไมโครซอฟท์	มากกว่า 30
เทคโนโลยีที่เกี่ยวข้อง	COM TI, MSMQ, OLE DB	RMI/JNI, COBRA
โพรโตคอล	DCOM	IIOP
สถานะในคอมโพเนนท์	ไม่มี	มี
คอมโพเนนท์ที่ถาวร (Persistence Component)	มี	มี
ทรานแซคแบบเกี่ยวเนื่อง (Nested Transaction)	สนับสนุน	ไม่สนับสนุน
การโหลดบาลานซ์ซิ่งในมิดเดิลแวร์	มี (โดยใช้ Application Center 2000)	มี

ตารางที่ 18-1 เปรียบเทียบคุณสมบัติระหว่าง COM+ และ EJB

คุณสมบัติ	COM+	EJB
การแคชข้อมูลในมิดเดิลเทียร์	ไม่มี (In memory database ถูกถอดจากวินโดว์ 2000 ก่อนออกจากจำหน่ายจากข้อจำกัดในด้านเทคโนโลยี	มี
การทำงานแบบอะซิงโครนัส (โดยใช้ Queued Component)	มี	ไม่มี
มีผู้ผลิตรายเดียว	ใช่	ไม่ใช่
มิดเดิลแวร์มาพร้อมกับระบบปฏิบัติการ	ใช่	ไม่ใช่
จำนวนโปรเซสเซอร์ต่อเครื่องสูงสุด	32	เกิน 256
จำนวนโปรเซสเซอร์สูงสุดต่อคลัสเตอร์	ไม่จำกัด	ไม่จำกัด
เครื่องมือในการพัฒนา	ไม่จำกัด	ไม่จำกัด แต่แนะนำว่าควรใช้ Microsoft Visual Studio

ตารางที่ 18-1(ต่อ) เปรียบเทียบคุณสมบัติระหว่าง COM+ และ EJB

บทที่ 19
บทวิจารณ์ และสรุป

19.1 สรุปผลการดำเนินงาน

จากการดำเนินงานโครงงานนี้การดำเนินงานได้แบ่งเป็น 3 ส่วนคือ

- ศึกษาทฤษฎีพื้นฐานของ COM, COM+ , DNA และ องค์ประกอบอื่น ๆที่เกี่ยวข้อง
- ทำการออกแบบระบบงานตัวอย่าง OlalaTour
- ทำการสร้างระบบงานโดยใช้ DNA ร่วมกับ Borland Delphi 5.0 และ Microsoft Visual Basic 6.0

สำหรับระบบงานที่ได้ทดลองสร้างนั้นก็ได้มีการใช้เทคโนโลยีต่าง ๆ ที่ได้ศึกษามา ซึ่งก็ประสบความสำเร็จพอสมควร โดยที่สามารถสร้างระบบงานซึ่งมีลักษณะเป็น มัลติเทียร์ โดยมีเทียร์ ต่าง ๆ 4 ส่วน คือ

- ไคลเอนต์ซึ่งเป็นทั้งเว็บเบราเซอร์ และ เป็นทั้งแฟลทไคลเอนต์
- เว็บเซิร์ฟเวอร์ ที่ใช้ IIS และมีเซิร์ฟเวอร์สคริปเป็น ASP
- แอปพลิเคชั่นเซิร์ฟเวอร์ ใช้ COM+ รันไทม์
- ดาตาเบสเซิร์ฟเวอร์เป็น Microsoft SQL Server 2000

ข้อจำกัดในการเรียกใช้บางส่วนที่ต้องมีสถานะ (Stateful) ซึ่งได้แก่รถเข็นทางกลุ่มแก้ปัญหาโดยใช้คุกกี้ (Cookie) เก็บสถานะของรถเข็นไว้ในฝั่งไคลเอนต์โดยเก็บเป็นสตริง เมื่อผู้ใช้ทำการจอง ก็จะส่งสตริงนี้มา ฝั่งคอมโพเนนท์ก็จะทำการตัดสตริง แล้วส่งต่อไปยังคอมโพเนนท์ที่ต้องเรียกใช้อื่นต่อไป

นอกจากนั้นยังมีข้อจำกัดในการเรียกใช้คอมโพเนนท์ข้ามโดเมน ทำให้สถาปัตยกรรมของระบบต้องเปลี่ยนจากเดิมที่จะให้คอมโพเนนท์อยู่ต่างโดเมน (โดยคอมโพเนนท์จะอยู่โดเมนเดียวกับเครื่องดาตาเบสเซิร์ฟเวอร์ของตัวมัน) เป็น คอมโพเนนท์อยู่โดเมนเดียวกันแล้วเรียกไปยังดาตาเบสเซิร์ฟเวอร์ซึ่งอยู่ต่าง โดเมน โดยต้องทำการทรัสต์โดเมนกัน

ปัญหาของการทำงานจะอยู่ในส่วนการศึกษาทฤษฎีและการออกแบบระบบ ซึ่งทำให้มีผลกระทบการสร้างระบบงาน ปัญหาของการศึกษาทฤษฎี นั้นเนื่องจากเทคโนโลยี COM, COM+ และ DNA นั้นยังเป็นเทคโนโลยีซึ่งยังใหม่ ทำให้มีข้อมูลเหล่านี้น้อย และข้อมูลบางส่วนก็ยังเป็นเพียงข้อมูล ที่ถูกตีพิมพ์ก่อนที่ตัวผลิตภัณฑ์จริงจะ ออกซึ่งทำให้เกิดความคลาดเคลื่อน เมื่อตัวผลิตภัณฑ์จริงออกมา นอกจากเนื่องจากผลิตภัณฑ์บางตัวนั้นยังไม่เสร็จ และจำเป็นต้องใช้ในงานบางส่วน ทำให้ต้องรอ และเมื่อผลิตภัณฑ์ออกมาก็ยังใหม่และไม่สมบูรณ์เท่าที่ควร ส่วนปัญหาในการออกแบบนั้นก็เนื่องจากยังไม่มีทฤษฎี ที่เป็นทางการของการออกแบบระบบงานเชิงคอมโพเนนท์ ทางกลุ่มจึงทำการลองผิดลองถูก และ ขาดประสบการณ์ในการออกแบบระบบ ทำให้ระบบที่ออกแบบมาในครั้งแรกนั้นบกพร่อง จนต้องแก้โครงสร้างของระบบอยู่หลายครั้ง ซึ่งปัญหาทั้งสองส่วนนี้ส่งผลกระทบต่อ การทำงานในส่วนการสร้างระบบงาน ซึ่งมีเวลาจำกัด ทำให้งานที่ออกมายังไม่สมบูรณ์เสียทีเดียว

19.2 แนวทางการพัฒนาต่อ

สำหรับการพัฒนาซอฟต์แวร์เชิงคอมโพเนนท์ที่ทางกลุ่มได้ศึกษาและลองสร้าง ระบบงานจริงมานั้นทางกลุ่มยังไม่ได้ทดสอบความสามารถในเรื่องการ Load Balancing ซึ่งต้องใช้โปรแกรม Microsoft Application Center 2000 เนื่องจากตัวผลิตภัณฑ์ที่ทางกลุ่มได้มายังไม่สมบูรณ์ และขาดแคลนเครื่อง จึงไม่สามารถหาข้อสรุปในเรื่องนี้ได้ นอกจากนี้เนื่องจากข้อจำกัดทางเวลาทำให้ยังศึกษาในเรื่องของการเรียกใช้คอมโพเนนท์ที่เป็นเทคโนโลยี อื่นคือ EJB รวมทั้งการทำงานแบบอะซิงโครนัสโดยผ่าน MSMQ (Microsoft Massage Queue) ซึ่งคิดว่าเป็นแนวทางในการพัฒนาซึ่งน่าจะเป็นสิ่งที่ควรศึกษาและพัฒนาต่อไป

ภาคผนวก ก

ยูนิฟิเคชั่นโมเดลลิ่งแลงเกวจ (Unification Modeling Language)

1 ความรู้เบื้องต้น

ยูเอ็มแอล (UML) หรือยูนิฟิเคชั่นโมเดลลิ่งแลงเกวจเป็นภาษาที่ถูกสร้างขึ้นเพื่อใช้แสดงโมเดลทางออบเจ็กต์โอเรียนเท็ต เกิดจากการพัฒนาของผู้นำทางเทคโนโลยีทางออบเจ็กต์ 3 ท่าน คือเกรดี้ บูช (Grady Booch) อิวาร์ จาคอบเซน (Ivar Jacobsen) และจิม รัมกอช (Jim Rumgaugh).

ยูเอ็มแอลนั้นได้รวมแนวความคิดของวิธีการต่างๆทางออบเจ็กต์เข้าด้วยกัน จุดประสงค์ของยูเอ็มแอลคือการสร้างโมเดลในการพัฒนาที่เข้าใจง่ายแต่สามารถนำไปใช้ประยุกต์ได้กับทุกๆระบบตัว ยูเอ็มแอลเองนั้นได้รับการสนับสนุนจากผู้นำในอุตสาหกรรมทางด้านคอมพิวเตอร์หลาย ๆ ราย
ใน UML มีโมเดลที่แสดงมุมมองต่อระบบที่ไม่เหมือนกัน ซึ่งประกอบไปด้วย

- ยูสเคสไดอะแกรม (Use-case Diagram) ใช้ในการแสดงการติดต่อระหว่างระบบกับผู้ใช้
- คลาสไดอะแกรม (Class Diagram) ใช้แสดงโครงสร้างทางลอจิกของระบบ
- ออบเจ็กต์ไดอะแกรม (Object Diagram) ใช้แสดงออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์
- สเตทไดอะแกรม (State Diagram) ใช้แสดงพฤติกรรมของออบเจ็กต์แต่ละตัวในระบบ
- คอมโพเนนต์ไดอะแกรม (Component Diagram) ใช้แสดงโครงสร้างทางกายภาพของซอฟต์แวร์
- ดีพลอยเมนท์ไดอะแกรม (Deployment Diagram) ใช้แสดงการติดต่อระหว่างฮาร์ดแวร์และซอฟต์แวร์
- อินเตอร์แอคทิฟไดอะแกรม (Interactive Diagram) ใช้แสดงพฤติกรรมของระบบ
- แอคทิวิตี้ไดอะแกรม (Activity Diagram) ใช้แสดงการไหลเวียนของอีเว็นต์ในยูสเคส

2 การวิเคราะห์ระบบ

โดยปกติระบบจะต้องติดต่อกับสภาพแวดล้อมภายนอกซึ่งการที่เราจะรู้ว่าระบบติดต่อกับสภาวะแวดล้อมภายนอกอย่างไรนั้นก็จำเป็นต้องมีการศึกษาความต้องการของระบบ หรือ Requirement Analysis จากเดิมในการวิเคราะห์ระบบแบบโครงสร้างเมื่อเราวิเคราะห์ความต้องการของระบบเป็นที่เรียบร้อยแล้ว จะได้ไดอะแกรม 2 ตัว คือ คอนเท็กซ์ไดอะแกรม และ ดาต้าโฟลว์ไดอะแกรม (Data flow Diagram) แต่สำหรับการวิเคราะห์ระบบด้วยออบเจ็กต์โอเรียนเท็ต หลังจากวิเคราะห์ความต้องการของระบบแล้วจะได้เป็น Use-Case Diagram และ External Event Context จะจำลองระบบเป็นเหมือนกับออบเจ็กต์ใหญ่ ๆ ตัวหนึ่ง ซึ่งประกอบด้วยออบเจ็กต์ย่อยภายใน และออบเจ็กต์ของระบบนี้จะแสดงการติดต่อสื่อสารทั้งการรับและส่งเมสเสจ กับออบเจ็กต์ที่เป็นแอคเตอร์ (Actor) ภายนอกส่วน Use-case จะแสดงฟังก์ชั่นการทำงาน

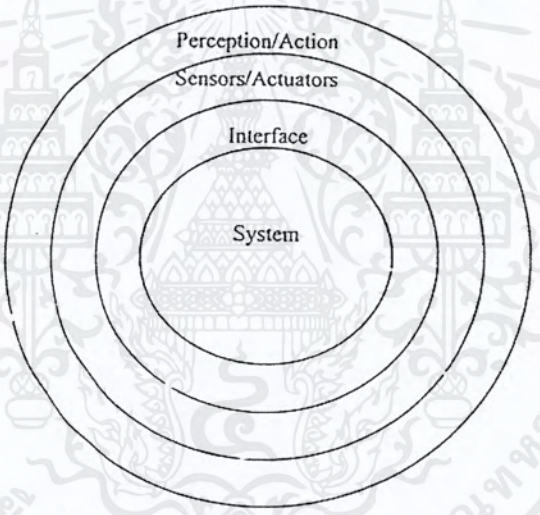
หลักของระบบและโพรโตคอล หรือข้อตกลงในการแลกเปลี่ยนเมสเสจที่จำเป็นเพื่อให้ระบบสามารถติดต่อกับแอคเตอร์ภายนอกได้อย่างถูกต้องตามความต้องการของระบบ

อีเว็นต์ภายนอก (External Event)

ซิสเท็มคอนเท็กซ์ (System Context) เป็นสิ่งที่แสดงขอบเขตของระบบที่ผู้พัฒนาหรือผู้ใช้สนใจต่อระบบ ซึ่งเราจะทำใช้คอนเท็กซ์ไดอะแกรม (Context Diagram) ซึ่งจะแสดงสภาวะแวดล้อมที่มีต่อระบบ ซึ่งประกอบไปด้วยผู้กระทำหรือแอคเตอร์ที่กระทำต่อระบบ นอกจากนี้ยังแสดงการแลกเปลี่ยนเมสเสจและอีเว็นต์ของระบบกับสภาวะแวดล้อมภายนอกอีกด้วย

ออบเจ็กต์ระดับคอนเท็กซ์ (Context-Level Objects)

คอนเท็กซ์ไดอะแกรมจะทำหน้าที่แสดงการติดต่อระหว่างออบเจ็กต์ของระบบกับออบเจ็กต์ภายนอกได้ แบ่งออบเจ็กต์ของระบบและออบเจ็กต์ภายนอกโดนการจำลองระบบออกมาเป็นหัวหอม (Onions) ซึ่งจะมีผิวเป็นชั้น ๆ ซึ่งแต่ละชั้นก็มีความสำคัญต่อระบบที่แตกต่างกันไปดังรูปที่ ก-1



รูปที่ ก-1 การเปรียบเทียบระบบแบบหัวหอม

สำหรับผิวนอกสุดของภาพเปรียบเทียบเป็น Perception/Action ซึ่งเป็นส่วนแสดงแอคชันของระบบซึ่งชั้นนี้ทำให้ผู้ใช้สามารถเห็นการกระทำของระบบที่มีต่อสุขภาพแวดล้อมได้

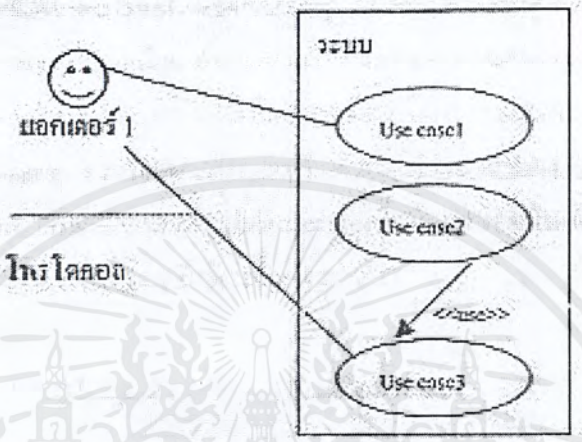
สำหรับชั้นต่อไป Sensors/Actuators เป็นชั้นของอุปกรณ์ที่ทำหน้าที่ติดต่อกับสิ่งแวดล้อม

สำหรับชั้นต่อไปอินเตอร์เฟสจะเป็นอุปกรณ์สำหรับใช้ในการสื่อสาร และชั้นสุดท้ายคือชั้นของระบบศูนย์กลางที่ทำหน้าที่ตัดสินใจและประมวลผลข้อมูล

ยูสเคส

คอนเท็กซ์ไดอะแกรมที่ได้จากการวิเคราะห์นั้นจะบอกถึงการติดต่อระหว่างสิ่งแวดล้อมภายนอกกับระบบ สำหรับในการวิเคราะห์ระบบแบบ ออบเจ็กต์โอเรียนเท็ดนั้นสามารถสร้างยูสเคสไดอะแกรมที่เพิ่มกรณีการติดต่อระหว่างระบบกับแอคเตอร์ภายนอก นอกจากนี้ยังใช้ในการตรวจสอบความถูกต้องของคอนเท็กซ์ไดอะแกรม

ยูสเคสไดอะแกรมจะแสดงมุมมองต่อฟังก์ชั่นการทำงานหลักๆของระบบ ซึ่งผู้พัฒนาระบบสามารถใช้ในการสื่อสารกับผู้ใช้ระบบได้ด้วยยูสเคสไดอะแกรม ในระดับบนสุดนั้นจะแสดงขอบเขตของระบบทั้งหมดซึ่งเป็นมุมมองจากฟังก์ชั่น การทำงานไม่ใช่มุมมองต่อคลาส ออบจ็กต์ที่จะมีในระบบโดยฟังก์ชั่นการทำงานของระบบนั้นจะแทนที่ด้วยวงรีและอยู่ในระบบใหญ่ ซึ่งแสดงโดยกรอบสี่เหลี่ยมโดยมีออบเจ็กต์กระจายอยู่รอบนอกระบบ ออบเจ็กต์ภายนอกเหล่านี้จะสัมพันธ์กับยูสเคสหนึ่ง ๆ อย่างจะมีความสัมพันธ์กับยูสเคสอื่น ๆ ซึ่งเราสามารถแทนได้โดยการลากเส้นแสดงความสัมพันธ์ระหว่างยูสเคสดังรูปที่ ก-2



รูปที่ ก-2 ยูสเคสไดอะแกรม

การกำหนดยูสเคส

ในการวิเคราะห์ระบบส่วนใหญ่จะใช้วิธีการกำหนดซีนาริโอ (Scenario) ขึ้นมาก่อนซึ่งซีนาริโอที่กำหนดขึ้นมานี้อาจจะมีมากกว่ายูสเคสจากนั้นนักพัฒนาจะเลือกซีนาริโอที่สำคัญต่อระบบขึ้นมาเพื่อทำเป็นยูสเคส

ซีนาริโอ

ซีนาริโอคือตัวตนของยูเคสในโมเดล ซีนาริโอจะกำหนดลำดับของเมสเสจของระบบ รวมถึงโครงสร้างของออบเจ็กต์ที่ทำงานร่วมกันเพื่อให้ระบบสามารถทำหน้าที่ที่รับผิดชอบได้ ข้อดีของซีนาริโอคือทำให้ผู้พัฒนาระบบและผู้ใช้สามารถเข้าใจการทำงานของระบบได้ง่ายขึ้น นอกจากนี้ยังใช้เป็นเครื่องมือในการตรวจสอบความต้องการของระบบ

ในขั้นตอนของการวิเคราะห์ระบบนั้น เมื่อเราได้คอนเท็กซ์ไดอะแกรมและยูสเคสไดอะแกรมแล้วเราก็วิเคราะห์ต่อโดยแยกระบบออกเป็นออบเจ็กต์ย่อยๆจากนั้นก็กำหนดซีนาริโอขึ้นสำหรับแต่ละยูสเคสโดยซีนาริโอนี้จะแสดงการรับเมสเสจระหว่างออบเจ็กต์ที่เรากำหนดขึ้นมาเพื่อทำงานที่รับผิดชอบตามยูสเคสได้สำเร็จ จะเห็นว่าในขั้นตอนนี้จะเป็นการตรวจสอบยูเคสและออบเจ็กต์ที่เรากำหนดขึ้นว่าครบหรือไม่

การวิเคราะห์ซีนาริโอไม่มีกฎเกณฑ์ที่แน่นอน แต่อาศัยความชำนาญของผู้วิเคราะห์และความเข้าใจในระบบ

ยูเอ็มแอลมีโมเดลที่รองรับการวิเคราะห์ซีนาริโออยู่ 2 โมเดลคือ

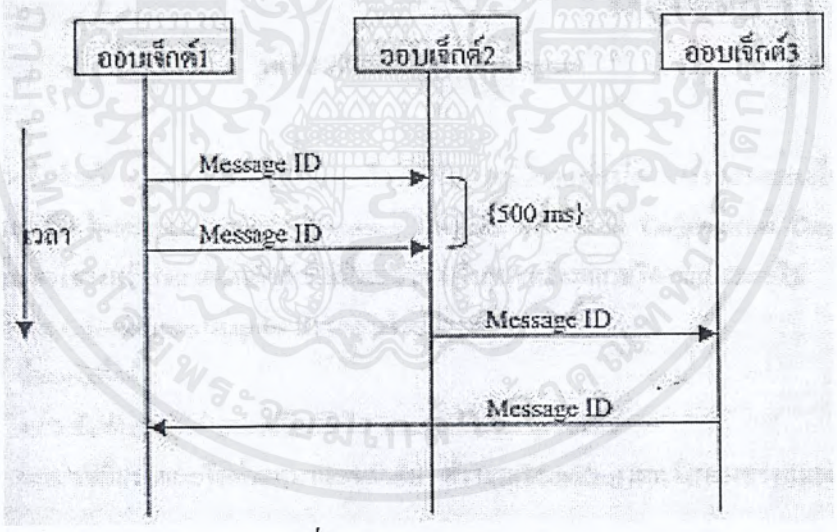
ซีเควนไดอะแกรม (Sequence Diagram) ที่แสดงถึงลำดับของเมสเสจที่ออบเจ็กต์ในระบบส่งและรับซึ่งกันและกัน

คอลแลบบอเรชั่นไดอะแกรม (Collaboration Diagram) จะแสดงโครงสร้างของออบเจ็กต์ที่ทำงานร่วมกันเป็นหลัก

ซีเควนไดอะแกรม

ส่วนประกอบของซีเควนไดอะแกรมจะมีเส้นในแนวตั้งหมายถึงออบเจ็กต์โดยมีชื่อของออบเจ็กต์เขียนอยู่ด้านบนหรือด้านล่างของเส้น ส่วนเส้นในแนวนอนนั้นแสดงถึงเมสเสจ เส้นตรงที่แทนเมสเสจทุกเส้นจะต้องมีออบเจ็กต์ต้นกำเนิดของเมสเสจ ส่วนจุดสิ้นสุดของเมสเสจก็จะเป็นออบเจ็กต์ปลายทาง โดยแต่ละเส้นของเมสเสจจะมีชื่อกำกับอยู่ โดยชื่อนี้อาจมีการเพิ่มรายการของพารามิเตอร์และค่าคงที่คืนกลับมาด้วย สำหรับเวลานั้นจะนับจากบนลงล่าง โดยสเกลของแกนตั้งจะบอกเพียงลำดับเมสเสจเท่านั้น ไม่สามารถใช้เป็นอัตราส่วนในการคำนวณหาเวลาระหว่างเมสเสจได้

ซีเควนไดอะแกรมส่วนใหญ่จะประกอบไปด้วยองค์ประกอบต่าง ๆ คือ ออบเจ็กต์ เมสเสจ คาบเวลา และสัญลักษณ์ของซีนาริโอ นอกจากนี้ยังสามารถเพิ่มข้อมูลที่จำเป็นเข้าไปในซีเควนไดอะแกรมได้ สำหรับตัวอย่างซีเควนไดอะแกรมที่เขียนขึ้นดังรูปที่ ก-3



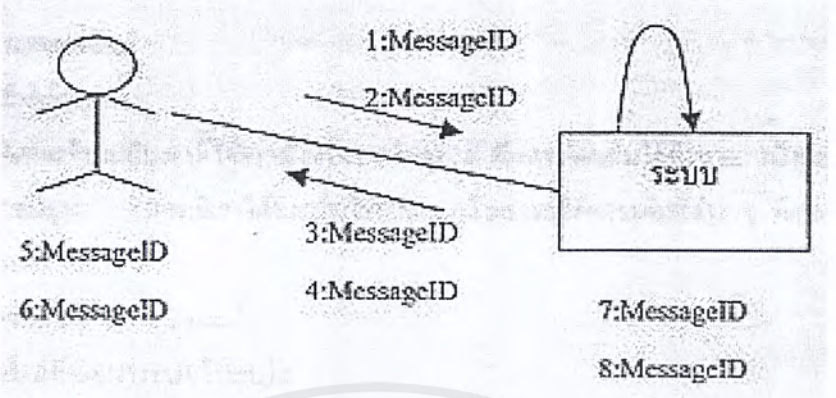
รูปที่ ก- 3 ซีเควนไดอะแกรม

สำหรับซีเควนไดอะแกรมที่ได้กล่าวไปจะมีเมสเสจในรูปแบบเพียร์ทูเพียร์ (Peer-to-Peer) คือมีออบเจ็กต์ตัวหนึ่งส่งเมสเสจให้ออบเจ็กต์อีกตัวหนึ่ง แต่ก็สามารถมีเมสเสจแบบกระจายได้ ซึ่งเมสเสจที่จะส่งออกจากออบเจ็กต์หนึ่งไปให้กับออบเจ็กต์อีกหลาย ๆ ตัว

คอลแลบบอเรชั่นไดอะแกรม

คอลแลบบอเรชั่นไดอะแกรมเป็นเครื่องมือสำหรับใช้แสดงซีนาริโออีกตัวหนึ่งที่มีประโยชน์ คอลแลบบอเรชั่นไดอะแกรมนั้นจะแสดงข้อมูลพื้นฐานเช่นเดียวกับซีเควนไดอะแกรม แต่ความแตกต่างคือ ซี

เควนไดอะแกรมจะแสดงลำดับของเมสเสจเป็นหลัก ส่วนคอลแลบบอเรชั่นไดอะแกรมนั้นจะแสดงโครงสร้างของออบเจ็กต์ที่ทำงานร่วมกันเป็นหลัก ตัวอย่างของคอลแลบบอเรชั่นไดอะแกรมดังรูปที่ ก-4



รูปที่ ก-4 คอลแลบบอเรชั่นไดอะแกรม

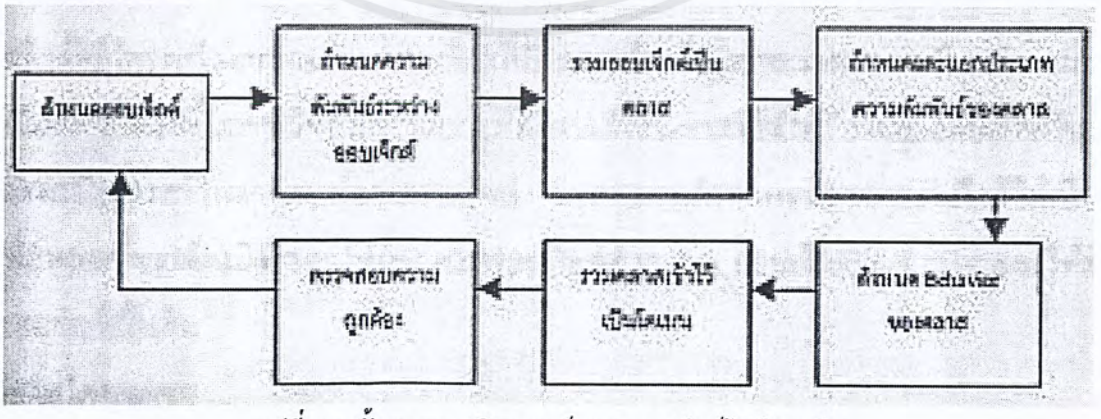
ข้อเสียคอลแลบบอเรชั่นไดอะแกรมคือ เมื่อมีการเปลี่ยนแปลงโครงสร้างของออบเจ็กต์แล้ว จำเป็นต้องมีการแก้ไขไดอะแกรมมากกว่าซีเควนไดอะแกรม อย่างไรก็ดีคอลแลบบอเรชั่นไดอะแกรมก็สามารถใช้แสดงโครงสร้างของออบเจ็กต์เพื่อเป็นแนวทางในการสร้างคลาสไดอะแกรมต่อไป

สำหรับคอลแลบบอเรชั่นไดอะแกรมประกอบด้วยส่วนต่าง ๆ ดังนี้ คือ

- ชื่อออบเจ็กต์
- ความสัมพันธ์ระหว่างออบเจ็กต์ที่มีการแลกเปลี่ยนเมสเสจกัน
- เมสเสจที่ประกอบไปด้วยหมายเลขลำดับ ตัวระบุ (Identifier) และทิศทางของเมสเสจ

กระบวนการทางออบเจ็กต์

ยูสเคสไดอะแกรมและคอนเท็กซ์ไดอะแกรมที่เราสร้างขึ้นมานั้นเป็นจุดเริ่มต้นในการวิเคราะห์ระบบแบบออบเจ็กต์โอเรียนเท็ตซึ่งเมื่อวิเคราะห์ระบบเสร็จเรียบร้อยแล้วจะได้เป็นโมเดล ซึ่งประกอบด้วยออบเจ็กต์ คลาส ความสัมพันธ์ระหว่างออบเจ็กต์และความสัมพันธ์ระหว่างคลาส ลำดับชั้นของการสืบทอดคุณสมบัติซึ่งถือว่าเป็นคุณสมบัติที่สำคัญที่สุดของออบเจ็กต์โอเรียนเท็ต กระบวนการทางออบเจ็กต์แสดงได้ดังรูปที่ ก-5



รูปที่ ก-5 ขั้นตอนการวิเคราะห์แบบออบเจ็กต์โอเรียนเท็ต

การกำหนดออบเจ็กต์

การกำหนดออบเจ็กต์โดยมากใช้วิธีการขีดเส้นใต้คำนาม วิธีการนี้ใช้กับคำอธิบายปัญหาที่ได้จากผู้ใช้ระบบโดยตรง คือการขีดเส้นใต้คำนาม หรือ วลี ที่เป็นคำนามอยู่ในคำอธิบายปัญหา หลังจากที่เราได้ขีดเส้นใต้คำนามแล้วเราจะได้ออบเจ็กต์ต่างๆ ที่แบ่งออกเป็น 3 ประเภท คือ

- ออบเจ็กต์ที่พัฒนาระบบสนใจ
- ออบเจ็กต์ที่พัฒนาระบบไม่สนใจ
- คุณสมบัติของออบเจ็กต์

โดยจุดประสงค์หลักในการกำหนดออบเจ็กต์คือ ต้องการหาออบเจ็กต์ที่ผู้ที่พัฒนาสนใจ ส่วนออบเจ็กต์ที่ไม่ได้สนใจนั้นก็อาจจะมีประโยชน์ต่อการวิเคราะห์ระบบในภายหลัง

กลยุทธ์ในการขีดเส้นใต้คำนามเพื่อการกำหนดออบเจ็กต์นั้นเป็นวิธีที่ง่ายและรวดเร็วในการค้นหาออบเจ็กต์รวมทั้งคุณสมบัติของออบเจ็กต์จากคำอธิบายปัญหา แต่ว่าการใช้กลยุทธ์นี้ทำให้เราได้ออบเจ็กต์ที่ไม่สนใจมากอีกทั้งยังมีความกำกวมซึ่งผู้วิเคราะห์ระบบจำเป็นต้องแก้ปัญหาเหล่านี้ด้วย

การกำหนดความสัมพันธ์ระหว่างออบเจ็กต์

จะดูจากการส่งเมสเสจเป็นหลักเพราะว่าเมสเสจทุกเมสเสจจะหมายถึงความสัมพันธ์ระหว่างออบเจ็กต์ซึ่งเป็นผู้ส่งเมสเสจกับออบเจ็กต์ซึ่งเป็นผู้รับเมสเซจ

การกำหนดคุณสมบัติของออบเจ็กต์

คุณสมบัติของออบเจ็กต์เป็นส่วนข้อมูลของออบเจ็กต์ที่ไม่สามารถแบ่งแยกย่อยลงไปได้อีก ถ้าในระหว่างการวิเคราะห์พบว่าคุณสมบัติที่กำหนดยังสามารถแยกย่อยลงไปได้ แสดงว่าควรกำหนดคุณสมบัตินั้นเป็นออบเจ็กต์ที่อยู่ภายใน

ในระบบงานทั่วไปบางครั้งพบว่าบางออบเจ็กต์ไม่จำเป็นต้องมีคุณสมบัติ แต่มีเพียงเมธอดก็เพียงพอแล้ว

การกำหนดคลาส

ในการกำหนดออบเจ็กต์โดยใช้คำอธิบายของปัญหานั้น เราจะได้ออบเจ็กต์ออกมามากมาย โดยมีหลายออบเจ็กต์ที่มีโครงสร้างเหมือนกัน ออบเจ็กต์ที่มีโครงสร้างข้อมูลและเมธอดเหมือนกัน เราสามารถโมเดลให้เป็นคลาสได้ ในขั้นตอนนี้ของการวิเคราะห์คือการกำหนดคลาสสำหรับแต่ละออบเจ็กต์ในระบบ

คลาสเป็นการสร้างเอกลักษณ์ของออบเจ็กต์ เราสามารถกำหนดชนิดของออบเจ็กต์ให้เป็นคลาสได้ คลาสที่กำหนดขึ้นจะต้องมีตัวระบุให้กับแต่ละออบเจ็กต์ด้วย เช่น ออบเจ็กต์บัญชี เราอาจให้ตัวระบุเป็นหมายเลขบัญชีเป็นต้น

คลาสไดอะแกรม

คลาสไดอะแกรมเป็นไดอะแกรมที่มีความสำคัญมากที่สุดในการวิเคราะห์และออกแบบโดยใช้วิธีทางออบเจ็กต์โอเรียนเท็ต เนื่องจากจะแสดงโครงสร้างของออบเจ็กต์และคลาสที่มีในระบบรวมทั้งแสดง

ความสัมพันธ์ด้วย คลาสไดอะแกรมจะเป็นโครงสร้างหลักของระบบและใช้ในการแยกย่อยรายละเอียดของระบบด้วยวิธีทางออบเจ็กต์โอเรียนเท็ต อย่างไรก็ดีคลาสไดอะแกรมนั้นอาจจะมีลักษณะคล้ายออบเจ็กต์ไดอะแกรมแต่ว่าคลาสไดอะแกรมจะแสดงเฉพาะคลาส โดยไม่แสดงอินสแตนซ์หรือออบเจ็กต์ทั้งหมด

ในบางกรณีอาจจำเป็นต้องเพิ่มคลาสที่แสดงความสัมพันธ์ (Association Class) เนื่องจากความสัมพันธ์นี้สามารถมีคุณสมบัติและเมธอดเป็นของตัวเอง

ความสัมพันธ์ระหว่างออบเจ็กต์

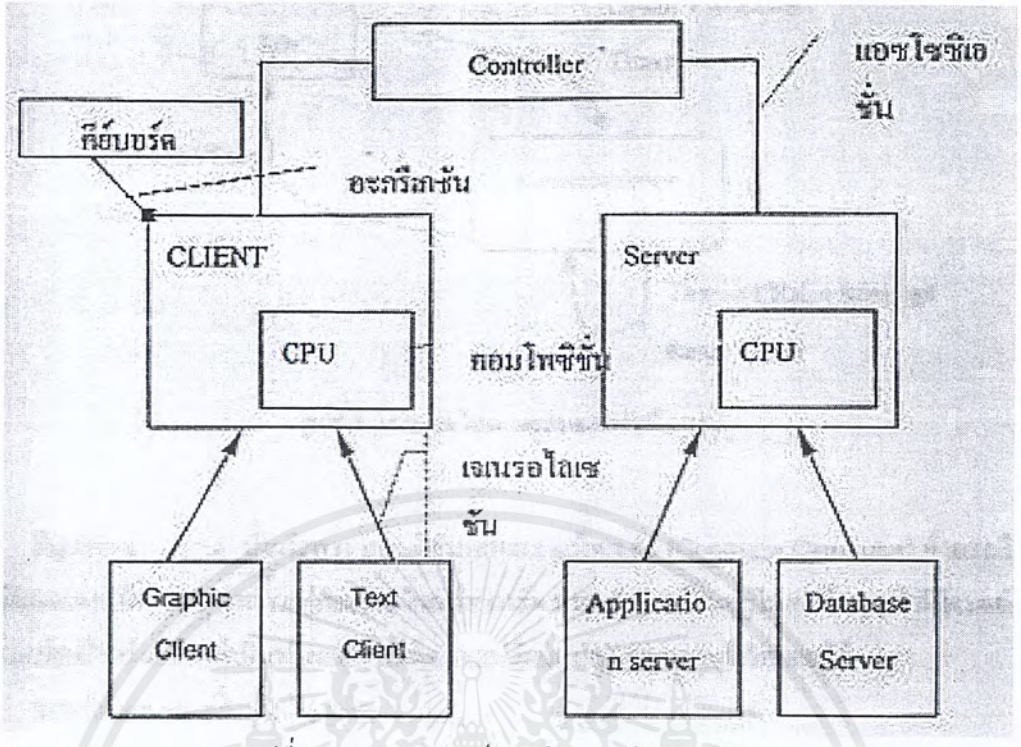
ในระบบออบเจ็กต์โอเรียนเท็ตแบ่งความสัมพันธ์ระหว่างออบเจ็กต์ออกเป็น4 ประเภทดังนี้

1. แอซโซซิเอชั่น (Association)
2. อะกรีเกชั่น (Aggregation)
3. คอมโพซิชั่น (Composition)
4. เจเนรอลไลเซชั่น (Generalization)

ความสัมพันธ์แอซโซซิเอชั่นเป็นความสัมพันธ์ระหว่างคลาส เมื่อออบเจ็กต์ของคลาสมีการส่งแมสเซจถึงกัน ในการสร้างความสัมพันธ์แบบแอซโซซิเอชั่นสามารถทำได้โดยประกาศอินสแตนซ์ไว้เป็นตัวอ้างอิงไปยังออบเจ็กต์ที่เป็นผู้รับแมสเซจ และเมื่อออบเจ็กต์ต้องการส่งแมสเซจก็เรียกฟังก์ชั่นจากอินสแตนซ์ออบเจ็กต์ผู้รับแมสเซจ

ความสัมพันธ์แบบอะกรีเกชั่นและคอมโพซิชั่นเป็นความสัมพันธ์ระหว่างคลาสที่มีลักษณะคล้ายกัน กล่าวคือการที่มีออบเจ็กต์ของคลาสหนึ่งเป็นส่วนหนึ่งของอีกออบเจ็กต์ของคลาสหนึ่งในการสร้างความสัมพันธ์แบบอะกรีเกชั่น และคอมโพซิชั่นสามารถทำได้โดยการประกาศอินสแตนซ์ของออบเจ็กต์ไว้ภายในคลาสที่โดนสร้างเป็นออบเจ็กต์เจ้าของ โดยออบเจ็กต์เจ้าของจะเป็ยผู้สร้างและทำลายออบเจ็กต์ที่อยู่ภายใน

ความสัมพันธ์แบบเจเนรอลไลเซชั่น เป็นความสัมพันธ์เมื่อมีออบเจ็กต์หนึ่งสืบทอดคุณสมบัติมาจากอีกออบเจ็กต์หนึ่งโดยในความสัมพันธ์เจเนรอลไลเซชั่นนั้น คลาสที่ได้รับการสืบทอดคุณสมบัติมาหรือสับคลาสนั้นได้รับการสร้างเป็นออบเจ็กต์ที่มีคุณสมบัติและเมทธอดที่เหมือนกับออบเจ็กต์ที่สร้างจากซุปเปอร์คลาสและยังมีคุณสมบัติและเมทธอดที่เพิ่มเข้ามาด้วย



รูปที่ ก-6 ความสัมพันธ์ระหว่างคลาสในแบบต่าง ๆ

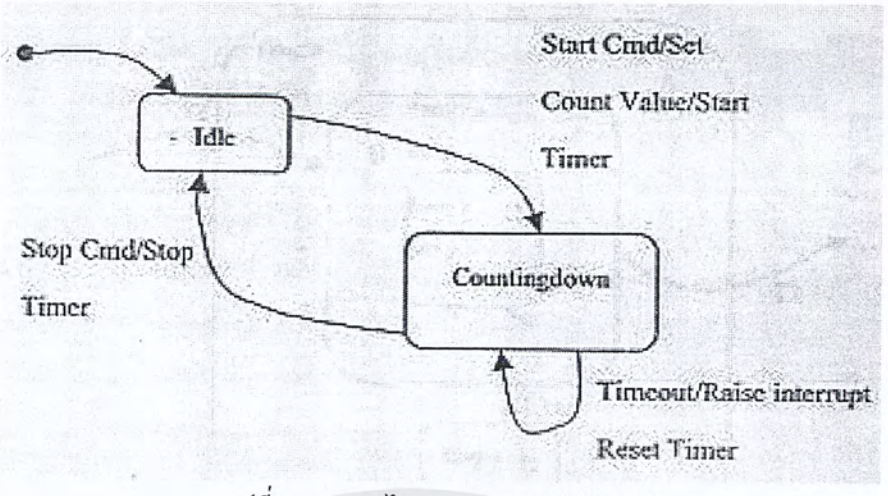
การกำหนดเมธอดของออบเจ็กต์

เมธอดเป็นส่วนที่บ่งบอกถึงคุณสมบัติและหน้าที่รับผิดชอบของออบเจ็กต์ ในการโมเดลโอเปอเรชั่น และเมธอดจะใช้สเตทไดอะแกรม ด้วยการใช้สเตทไดอะแกรมจะทำให้เราสามารถตรวจสอบว่าออบเจ็กต์สามารถทำงานร่วมกันได้อย่างถูกต้องหรือไม่

จากที่นำเสนอไปก่อนหน้านี้เป็นการกำหนดออบเจ็กต์ คุณสมบัติของออบเจ็กต์รวมทั้งโครงสร้างของออบเจ็กต์และความสัมพันธ์ ความสัมพันธ์ระหว่างออบเจ็กต์จะเกิดขึ้นได้ก็ต่อเมื่อมีการเรียกเมธอดของออบเจ็กต์

การกำหนดสเตทของออบเจ็กต์

สำหรับการวิเคราะห์และออกแบบระบบแบบออบเจ็กต์โอเรียนเท็ตเราจะกำหนดให้ 1 คลาสมี 1 สเตทไดอะแกรม ตัวสเตทนั้นแสดงด้วยสี่เหลี่ยมที่มีขอบมน สำหรับการเปลี่ยนสเตทจะแสดงด้วยเส้นตรงที่มีหัวลูกศรโดยเริ่มต้นจากสเตทที่เป็นต้นทางไปยังสเตทที่เป็นปลายทาง สำหรับรายละเอียดการเปลี่ยนสเตทนั้นจะประกอบไปด้วย อีเว็นต์ที่ทำให้เกิดการเปลี่ยนสเตทตามด้วย แอคชั่น (ฟังก์ชั่นหรือโอเปอเรชั่น) ที่จะต้องทำเมื่อมีการเปลี่ยนสเตท ส่วนเครื่องหมายวงกลมระบายสีดำเป็นจุดเริ่มต้นของสเตทไดอะแกรมดังรูปที่ ก-7



รูปที่ ก-7 สเตทไดอะแกรมของตัวจับเวลา

สัญลักษณ์ <> เรียกว่าคอนดิชันนอลคอนเน็คเตอร์(Condition Connector) ซึ่งยอมให้มีการเปลี่ยนสเตทได้หลายสเตทขึ้นอยู่กับเงื่อนไขหรือการ์ด ตัวการ์ดจะเขียนอยู่ในวงเล็บ [] ถ้ามีอีเว็นต์เกิดขึ้นแล้วการ์ดจะเป็นจริงหรือไม่เป็นจริงจะทำให้เกิดการเปลี่ยนแปลงสเตทตามที่กำหนดไว้

สเตทไดอะแกรมพื้นฐานในยูเอ็มแอล

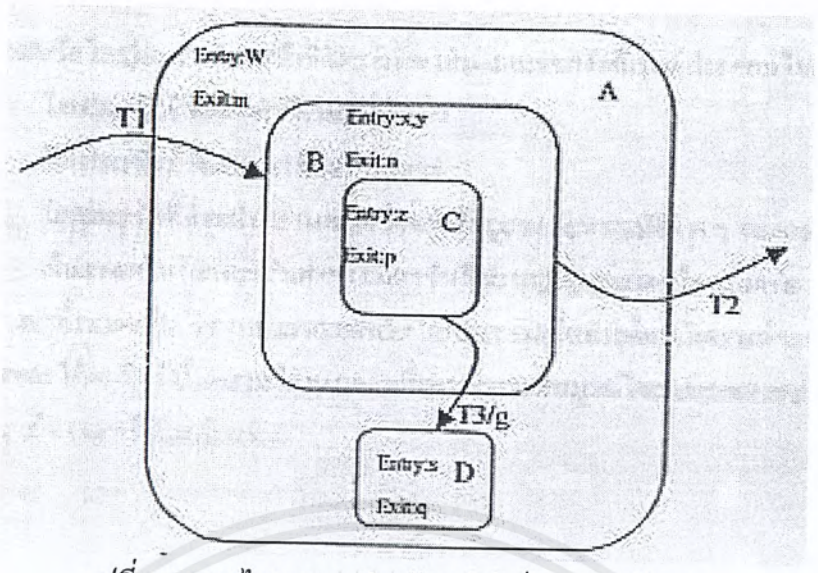
ไวยากรณ์ของสเตทไดอะแกรมในยูเอ็มแอล ได้อธิบายไปแล้วในบางส่วนในรายงานส่วนนี้จะนำเสนอส่วนที่เพิ่มความสามารถต่าง ๆ ที่สำคัญสำหรับสเตทไดอะแกรม

- สามารถทำสเตทซ้อนสเตทได้โดยสเตทภายในเรียกว่า สับสเตท ส่วนสเตทภายนอกเรียก ซุปเปอร์สเตท
- มีการกำหนดสเตทเริ่มต้นให้กับซุปเปอร์สเตทได้ เมื่อมีการเปลี่ยนสเตทมาที่ ซุปเปอร์ สเตทนี้ ก็ให้เริ่มที่สเตทเริ่มต้นนี้
- สามารถใช้ฮิสทอรีคอนเน็คเตอร์ (History Connector) คือ เมื่อกลับเข้ามาซุปเปอร์สเตทนี้อีกกำหดให้เริ่มสเตทที่ค้างอยู่ก่อนที่จะมีการเปลี่ยนสเตทออกไป ถ้าไม่มีก็ให้เริ่มที่ฮิสทอรีสเตท

สเตทไดอะแกรมสามารถมีแอคชั่นในตัวสเตทเองหรือในช่วงการเปลี่ยนแปลงสเตทก็ได้ โดยสามารถแบ่งเป็น

- แอคชั่นที่ทำเมื่อมีการเข้าสู่สเตท หรือ Entry Action (ใช้สัญลักษณ์ "Entry:")
- แอคชั่นที่ทำเมื่อมีการออกจากสเตท หรือ Exit Action (ใช้สัญลักษณ์ "Exit:")
- แอคชั่นที่ทำเมื่อสเตทนั้นๆแอคทีฟ หรือ Activities (ใช้สัญลักษณ์ "Do:")

จากที่กล่าวมาสามารถสรุปได้ดังรูปที่ ก-8



รูปที่ ก-8 สเตทไดอะแกรมสำหรับซุปเปอร์สเตทและลับสเตท

สำหรับทรานซิชั่นอาจจะมีพารามิเตอร์และการ์ดรวมทั้งแอคชั้นโดยไวยากรณ์ สรุปได้ดังนี้

ชื่ออีเว็นต์(รายการพารามิเตอร์)[การ์ด]/รายการแอคชั่นสำหรับการเปลี่ยนสเตท ^ รายการอีเว็นต์

ซึ่งสรุปได้ดังตารางที่ ก-1

ค่า	คำอธิบาย
ชื่ออีเว็นต์	เป็นชื่ออีเว็นต์ที่ทำให้เกิดการเปลี่ยนสเตท
การ์ด	เป็นเงื่อนไขที่อยู่ในรูปประโยคบูลีน ซึ่งจะต้องมีค่าเป็นจริงจึงจะทำให้เกิดการเปลี่ยนสเตทได้ โดยปกติจะใช้ร่วมกับคอนดิชันอลซุปเปอร์สเตทคอนเนคเตอร์ เพื่อตัดสินใจว่าจะเปลี่ยนไปยังสเตทใด
รายการแอคชั่น	เป็นรายการที่ใช้เครื่องหมายลูกน้ำแยกแต่ละส่วน โดยเป็นแอคชั่นที่จะต้องเกิดขึ้นเมื่อมีการเปลี่ยนสเตท
รายการอีเว็นต์	เป็นรายการอีเว็นต์ที่ใช้ลูกน้ำขั้นลำดับ โดยจะเป็นอีเว็นต์ที่จะต้องเกิดขึ้นเมื่อมีการเปลี่ยนสเตท

ตารางที่ ก-1 ความหมายของพารามิเตอร์ต่าง ๆ ของทรานซิชั่น

การกำหนดเมธอดหรือโอเปอเรชั่น

การกำหนดเมธอดหรือโอเปอเรชั่นมีหลักการดังนี้

- กำหนดเมธอดหรือโอเปอเรชั่นสำหรับออบเจ็กต์โดยเป็นอิสระต่อกัน
- พยายามซ่อนรายละเอียดโครงสร้างภายในออบเจ็กต์โดยผู้ต้องการติดต่อต้องผ่านอินเตอร์เฟสที่กำหนดไว้เท่านั้น
- ซุปเปอร์คลาสควรมีเมธอดหรือโอเปอเรชั่นที่สับคลาสส่วนใหญ่ต้องการใช้งาน
- เมธอดหรือโอเปอเรชั่นที่ทำหน้าที่จัดการเกี่ยวกับแมสเซจหรืออีเว็นต์ประกอบไปด้วย
  - โอเปอเรชั่นที่จัดการกับอีเว็นต์ที่เกิดขึ้น

- โอเปอเรชั้นที่จัดการการรับส่งแมสเซจ
- โอเปอเรชั้นที่ทำหน้าที่อ่านหรือเขียนค่าข้อมูลของคุณสมบัติต่างๆของออบเจ็กต์
- เมธอดหรือโอเปอเรชั่นสำหรับแอคชั่นที่ปรากฏอยู่บนสเตทไดอะแกรม

จากที่กล่าวมาทั้งหมดเป็นวิธีการหาเมธอดหรือโอเปอเรชั่น เพื่อทำให้การวิเคราะห์ระบบมีความถูกต้องมากที่สุด โดยจะได้ผลลัพธ์เป็นคลาสไดอะแกรมที่ค่อนข้างจะสมบูรณ์ โดยแต่ละคลาสประกอบไปด้วยคุณสมบัติ และเมธอดหรือโอเปอเรชั่น

3 การออกแบบ

ในกระบวนการการออกแบบนั้นจะเป็นการปรับเปลี่ยนโครงสร้างและการทำงานของโมเดลที่ได้จากการวิเคราะห์ระบบ โดยสามารถแบ่งออกเป็น 3 ระดับ คือ

- การออกแบบในระดับสถาปัตยกรรม (Architectural Design) เป็นการออกแบบในระดับโครงสร้างของระบบในระดับใหญ่คือ ระบบย่อย (Subsystem) แพคเกจ (Package) และทาสก์ (Task)
- การออกแบบในระบบโครงสร้างการทำงาน (Mechanistic Design) เป็นการออกแบบที่เน้นเกี่ยวกับโครงสร้างการทำงานระหว่างคลาสและออบเจ็กต์ที่มีอยู่ในระบบเพื่อให้บรรลุหน้าที่รับผิดชอบ
- การออกแบบในระดับรายละเอียด (Detail Design) จะกล่าวถึงโครงสร้างข้อมูลและอัลกอริทึมในการทำงานของคลาสต่าง ๆ

จากที่กล่าวมาทั้งหมดแนวทางการออกแบบระบบเพื่อปรับปรุงสิ่งที่ได้จากการวิเคราะห์ให้ถูกต้องมากที่สุดและสามารถนำกลับมาใช้ใหม่ได้มากที่สุด ซึ่งเป็นจุดประสงค์ที่สำคัญที่สุดในการวิเคราะห์และออกแบบระบบแบบออบเจ็กต์โอเรียนเท็ต ดังนั้นในปัจจุบันได้มีการเก็บรูปแบบการออกแบบในระบบเก่าเพื่อนำกลับมาประยุกต์ใช้กับระบบใหม่โดยอาจจะมีการปรับเปลี่ยนในบางส่วนบ้างก็ตาม รายละเอียดที่เก็บเกี่ยวกับ ข้อกำหนดของคลาสและออบเจ็กต์รวมทั้งความสัมพันธ์ระหว่างกันในระบบ ดังนั้นผู้ที่ออกแบบระบบมือใหม่ก็สามารถออกแบบระบบได้อย่างมีประสิทธิภาพ ด้วยการศึกษารูปแบบการออกแบบที่มีผู้เสนอไว้ ไปประยุกต์ใช้ทำให้ไม่ต้องเสียเวลาในการศึกษาและสร้างวิธีในการออกแบบระบบของตนเองขึ้นมา รูปแบบการออกแบบนี้เราเรียกว่า ดีไซน์แพทเทิร์น (Design Pattern)

บรรณานุกรม

[1] Richard C. Leinecker (2000) : "COM+ Unleashed", Sams Publishing

[2] Alan Gordon (2000) : "The COM and COM+ Programming Primer", Prentice Hall PTR

[3] Eric Harmon (2000) : "Delphi COM Programming", Macmillan Technical Publishing

[4] Inprise Coperation (1999) : "Borland Delphi's 5 Developer Guide", Inprise Coperation

[5] Steve Teixeira and Xavier Pacheco (2000) : "Borland Delphi's 5 Developer Guide", Sams Publishing

[6] Marco Cantซ (1999) : "Mastering Delphi 5", Sybex Inc.

[7] Alex Fedorov and Natalia Elmanova, Ph. D. (2000) : "Advance Delphi Developer's Guid to ADO", World Ware Publishing Inc.

[8] Alan W. Brown (2000) : "Large-Scale, Component-Based Development", Prentice Hall PTR

[9] Guy Eddon and Henry Eddon (1998) : "Programming Components with Microsoft Visual Basic 6.0" , Microsoft's Press

[10] Noel Jerke, George Szabo, David Jung and Don Kiely (1999) : "The Waite Group's Visual Basic 6 Client/Server How-To", Techmedia

เว็บไซต์อ้างอิง

[11] http://msdn.microsoft.com