สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ระบบสารสนเทศผ่านเว็บเซอร์วิส
WEB SERVICES INFORMATION SYSTEM



นาย สุรเชฐ เกียรติเรืองชัย
นาย วรายุส โรจนศิริบุตร
นาย โอฬาร ลีนะอังกูรรุจิ

เลขหมู่.....
เลขทะเบียน.....61969.....
วัน,เดือน,ปี.....25 ก.ค. 2549.....

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบสารสนเทศผ่านเว็บเซอร์วิส

WEB SERVICES INFORMATION SYSTEM

นาย สุรเชฐ เกียรติเรืองชัย
นาย วรายุส โรจนศิริบุตร
นาย โอฬาร ลีนะอังกูรรุจิ



อาจารย์ที่ปรึกษา
ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบสารสนเทศผ่านเว็บเซอร์วิส

WEB SERVICES INFORMATION SYSTEM

ผู้จัดทำ

1. นาย สุรเชฐ เกียรติเรืองชัย รหัสประจำตัว 44010555
2. นาย วรายุส โรจนศิริบุตร รหัสประจำตัว 44010588
3. นาย โอพาร ลีนะอังกูรรุจิ รหัสประจำตัว 44010628



_____ อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบสารสนเทศผ่านเว็บเซอร์วิส

นาย สุรเชฐ เกียรติเรืองชัย
นาย วรายุส โรจนศิริบุตร
นาย โอพาร ลีนะอังกูรรุจิ
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

ทุกวันนี้ธุรกิจส่วนใหญ่ได้ทำธุรกรรมบนอินเตอร์เน็ท ซึ่งระบบที่สนับสนุนการจัดการนั้นมีความซับซ้อนและถูกรวมกับระบบที่เกี่ยวข้องไว้ ดังนั้นสิ่งที่สำคัญที่สุดในการพัฒนาระบบคือการปรับปรุงแก้ไขให้การติดต่อสื่อสารระหว่างระบบต่างๆให้มีมาตรฐานเดียวกัน

เว็บเซอร์วิสเป็นมาตรฐานหนึ่งที่ถูกพัฒนาขึ้นมาจากบริษัทซอฟท์แวร์ชั้นนำ เช่น ไมโครซอฟท์ ไอบีเอ็ม ซัน เป็นต้น ซึ่งทำเป็นมาตรฐานเดียวกันและไม่ขึ้นอยู่กับภาษา และระบบปฏิบัติการ เว็บเซอร์วิสได้อธิบายเกี่ยวกับข้อมูลของโครงสร้าง และบริการต่างๆที่

จุดประสงค์ของปริญญานิพนธ์นี้คือการทำระบบเว็บเซอร์วิสให้มีความปลอดภัยโดยพัฒนาด้วยภาษาจาวา อีกทั้งยังมีการสาธิตวิธีการสร้างความปลอดภัยให้กับเว็บเซอร์วิส โดยการสร้างแอพพลิเคชันโอลาล้ามูฟวี่ ซึ่งเป็นระบบซื้อ ดีวีดี ซีดี ออนไลท์ และมีการใช้โปรโตคอล ws-security ในการรักษาความปลอดภัยให้กับแอพพลิเคชั่น ในแอพพลิเคชั่นนี้ประกอบไปด้วยการเรียกใช้เซอร์วิสต่างๆและใช้เทคโนโลยีโมบายเอเจนต์ในการดึงฐานข้อมูลออกมาประมวลผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Webservices information system

Surachet Kiatruangchai

Varayus Rojanasiributr

Olan Leenaungkoonruji

Dr. Worawat Limpoka Advisor

ABSTRACT

Nowadays, Most Business Transactions are done on the internet. The System that supports these transactions is complex and integrated with the related system so the most inportant aspect that develops the system is how to improve and unified the standard about communication between the systems.

The webservices is one standard which has been developed for a year by many major players on software development such as Microsoft , IBM , SUN Microsystem. It is a unified standard and don't depend on laguage , platform or operating system that used to develop the system but it relates with the information. The webservices describes about the structure 's information , the service that the application server provides the other application.

The purpose of this thesis is to make the security on the webservices application developed by JAVA. We demonstrate how to provide the security to the webservices application by developing security webservices application, Olala Movie, is DVD/CD online system and use the newly security protocol for webservice, ws-security, to provide security for Olala Movie. The application consists of many called services and uses a mobile agent to compute how to get the cheapest price in many dealers.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วรวัฒน์ ลิ้มโภคา และอาจารย์ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมาและอาจารย์ทุกท่านที่ได้ประสิทธิประสาทวิชา นอกจากนั้นยังได้ให้ความรู้และแนวคิดในการทำงานรวมถึงการใช้ชีวิต อันมีค่ายิ่งให้แก่ผู้เขียน ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบคุณพี่ ๆ เพื่อนๆ ห้อง OLALA และพี่เง๋ (นักศึกษาปริญญาโทจากเวียดนาม) ที่ช่วยเป็นกำลังใจเป็นที่ปรึกษาและให้ประสบการณ์ในการใช้ชีวิตในมหาวิทยาลัยที่มีค่ายิ่งใช้กับผู้เขียน

สุดท้ายนี้ ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นาย สุรเชฐ เกียรติเรืองชัย
นาย วรายุส โรจนศิริบุตร
นาย โอพาร ลีนะอังกูรรุจิ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญภาพ	IX
บทที่ 1 บทนำ	1
1.1 ความเป็นมาของโครงงาน	1
1.2 วัตถุประสงค์ของโครงงาน	1
1.3 ขอบเขตของโครงงาน	2
1.4 เทคนิค/เทคโนโลยีที่ใช้ในการพัฒนา	2
1.5 วิธีการดำเนินงาน	2
บทที่ 2 Web Services	3
2.1 การพัฒนาของเว็บ	3
2.2.1 สแตติกเว็บเพจ (Static Web Page)	3
2.2.2 ไดนามิกเว็บเพจ (Dynamic Web Page)	3
2.1.3 บริการบนเว็บ (Web Service)	4
2.2 เว็บเซอร์วิส	4
2.2.1 หลักการพื้นฐานของเทคโนโลยีต่างๆ ที่พัฒนามาสู่เว็บเซอร์วิส	5
2.2.2 ลักษณะของเว็บเซอร์วิส	5
2.2.3 เทคโนโลยีพื้นฐานของเว็บเซอร์วิส	6
2.3 มาตรฐานที่ใช้ในเว็บเซอร์วิส	7
2.3.1 XML (eXtensible Markup Language)	7
2.3.2 SOAP	7
2.3.3 UDDI	9
2.3.4 WSDL	9
บทที่ 3 Security และ Key	10
3.1 นิยามความมั่นคงปลอดภัยคอมพิวเตอร์	10
3.2 การพิสูจน์ตัวตน (Authentication)	11
3.3 การกำหนดสิทธิ์ (Authorization)	12
3.4 การเข้ารหัส (Encryption)	12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

	หน้า
3.5 การรักษาความสมบูรณ์ (Integrity)	13
3.6 การตรวจสอบ (Audit)	13
3.7 ประเภทของการพิสูจน์ตัวตน (Authentication Types)	13
3.8 ไม่มีการพิสูจน์ตัวตน (No Authentication)	14
3.9 การพิสูจน์ตัวตนโดยใช้รหัสผ่าน (Authentication by Passwords)	14
3.10 การพิสูจน์ตัวตนโดยใช้ Password Authenticators หรือ Tokens	14
3.11 การพิสูจน์ตัวตนโดยการเข้ารหัสโดยใช้กุญแจสาธารณะ (Public-key cryptography)	15
3.12 การพิสูจน์ตัวตนโดยการใช้ลายเซ็นอิเล็กทรอนิคส์ (Digital Signature)	17
3.13 โพรโตคอลในการพิสูจน์ตัวตน(Authentication Protocol)	21
3.14 Secure Socket Layer (SSL)	21
3.15 เอกสารสิทธิ์ (Digital Certificates)	25
3.16 Certificate Authority	29
บทที่ 4 WS-Security	31
4.1 WS-Security	32
4.2 การส่งซีเคียวริตี้ โทเคน (Sending Security Tokens)	33
4.3 การรักษาความสมบูรณ์ (Integrity)	35
4.4 การรักษาความลับ (Confidentiality)	37
บทที่ 5 เปรียบเทียบ .NET เว็บเซอร์วิส และจาวาเว็บเซอร์วิส	42
5.1 แพลตฟอร์มการพัฒนา	42
5.1.1 แพลตฟอร์มของ .NET	42
5.1.2 แพลตฟอร์มของ จาวา	43
5.2 เปรียบเทียบเทคโนโลยีจากทั้งสองค่าย	44
5.2.1 Vendor Solution	45
5.2.2 การสนับสนุนระบบเดิม	45
5.2.3 การยอมรับทางการตลาด	45
5.2.4 ภาษาที่รองรับ	45
5.2.5 การสนับสนุนแพลตฟอร์มเดิม	46
5.2.6 Portability	46
5.2.7 เครื่องมือที่ใช้ในการพัฒนา	46
5.2.8 การสนับสนุนการใช้งานเว็บเซอร์วิส	46
5.2.9 การขยายการรองรับการใช้งานของระบบ(Scalability)	47
5.2.10 สรุปการเลือกใช้งาน	47

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

	หน้า
บทที่ 6 Mobile Agent และ IBM Aglet	48
6.1 รูปแบบการทำงานบน Network	48
6.1.1 ไคลเอนต์-เซิร์ฟเวอร์	48
6.1.2 Code-on-demand	48
6.1.3 โมบายเอเจนต์	49
6.2 โมบายเอเจนต์คืออะไร	50
6.3 โมบายเอเจนต์ทำอะไรได้บ้าง	50
6.4 ทำไมต้องใช้โมบายเอเจนต์	51
6.5 ส่วนประกอบของ Mobile Agent	53
6.5.1 Agent	53
6.5.2 Place	53
6.5.3 Agent Transport Protocol (ATP)	54
6.6 กายวิภาคของ Mobile Agent	55
6.6.1 การเกิด	55
6.6.2 การตาย	55
6.6.3 การย้าย	55
6.7 IBM Aglet	56
6.7.1 แบบจำลอง Aglet	57
6.7.2 Aglet Package	57
6.8 วงจรชีวิตของ Aglet	59
6.8.1 Create	59
6.8.2 Dispose	59
6.8.3 Dispatch	60
6.8.4 Retract	60
บทที่ 7 เราจะใช้ Aglet อย่างไร	61
7.1 การติดต่อกับ Aglet	61
7.1.1 การส่ง Message	61
7.1.2 ประเภทของ Message	61
7.2 รูปแบบการทำงานของโมบายเอเจนต์	63
7.2.1 ท่องเที่ยว	63
7.2.2 จัดการงาน	63
7.2.3 พบปะกัน	63

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

	หน้า
บทที่ 8 การรวม Aglet เข้ากับ Application อื่นๆ	65
8.1 RMI	65
8.2 การสร้าง ATP Server	69
8.3 การควบคุม ATP Server ที่สร้างขึ้น	70
บทที่ 9 ความปลอดภัยในเรื่องของ Aglet	71
9.1 สิ่งผิดพลาดที่เกิดขึ้นได้ของหลักการ Mobile Agent	71
9.2 วิธีจัดการกับความปลอดภัยของ Aglet	71
9.2.1 การจัดการกับการพิสูจน์ตัวตน	70
9.2.2 การจัดการกับการรักษาความสมบูรณ์ของข้อมูล	71
9.2.3 การจัดการกับการรักษาความลับ	71
9.2.4 การจัดการการกำหนดสิทธิ์	72
9.2.5 การจัดการกับการรับผิดชอบต่อข้อมูล	72
9.2.6 การจัดการกับการตรวจสอบ	72
9.3 Aglets Security Policy File	72
บทที่ 10 Application	76
10.1 Overview ของระบบ OLALA Movie	76
10.2 ระบบ OLALA Movie	76
10.3 โครงสร้างของระบบ	77
10.4 วิเคราะห์และออกแบบระบบสั่งซื้อดีวีดี/ซีดี ออนไลน์	79
10.4.1 แผนผังระบบ(Use case)	79
10.4.2 ซีเควนซ์ไดอะแกรมของการซื้อ	79
10.5 ส่วนติดต่อผู้ใช้	80
10.5.1 หน้าหลักของเว็บ	80
10.5.2 หน้ารายการสินค้า	80
10.5.3 หน้ารายการสั่งซื้อ	81
10.5.4 หน้าวิธีชำระเงิน	81
บทที่ 11 บทวิจารณ์และสรุป	82
11.1 บทวิจารณ์และสรุป	82
11.2 แนวทางการพัฒนาต่อไป	82

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

ก. IBM Aglet

1. การติดตั้ง-ตั้งค่า Aglet 2.0.2 84
2. การใช้งาน Tahiti 85
3. การเขียนโปรแกรมสร้างเอเจ้นต์เซิร์ฟเวอร์ 87
4. การเขียนโปรแกรมเอเจ้นต์ตัวอย่าง 88

ข. การตั้งค่า BEA WebLogic ให้มีความปลอดภัย

1. สร้างเว็บเซอร์วิสตัวอย่างที่มี WS-Security 89
2. สร้างเว็บเซอร์วิสไคลเอนท์ 93
3. การตั้งเซิร์ฟเวอร์ให้มี SSL 96

ค. การเขียนโปรแกรม RMI

1. RMI Interface 97
2. RMI Implements 97
3. RMI เซิร์ฟเวอร์ 97
4. RMI ไคลเอนท์ 97
5. การใช้งาน RMI 97

ง. ไฟล์ที่อยู่บนเครื่องต่างๆในระบบ 98

จ. การตั้งค่าระบบ และ การทดลอง

1. เริ่มเอเจ้นต์เซิร์ฟเวอร์ และ RMI เซิร์ฟเวอร์ 100
2. เริ่มเว็บเซอร์วิสเซิร์ฟเวอร์ 100
3. เริ่มเว็บแอพลิเคชั่นเซิร์ฟเวอร์ 100
4. การเรียกดูหน้ารายการสินค้า 100

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้า

ตารางที่ 3.1 เปรียบเทียบข้อดีข้อเสียของการพิสูจน์ตัวตนแต่ละชนิด 18

ตารางที่ 3.2 อธิบายเวอร์ชันของ SSL และผู้พัฒนา 21

ตารางที่ 5.1 ตารางแสดงการสนับสนุนการทำงานด้านต่างๆของ 2 แพลตฟอร์ม 44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

	หน้า
รูปที่ 2.1 แสดงความสัมพันธ์ของผู้ให้บริการ ผู้ขอใช้บริการ และตัวแทนผู้ให้บริการ	6
รูปที่ 3.1 แสดง Security Pyramid	10
รูปที่ 3.2 กระบวนการพิสูจน์ตัวตน	11
รูปที่ 3.3ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจ	16
รูปที่ 3.4 ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจเพื่อการพิสูจน์ตัวตน	17
รูปที่ 3.5 การส่งข้อมูลเข้าไปใน Hash function	17
รูปที่ 3.6 การเข้ารหัสเมสเซสไดเจสต์ด้วยกุญแจส่วนตัวเพื่อเป็นการลงลายเซ็น	18
รูปที่ 3.7 ขั้นตอนการเปรียบเทียบความถูกต้อง	18
รูปที่ 3.8 กระบวนการเริ่มต้นการติดต่อสื่อสารของโพรโตคอล SSL	23
รูปที่ 3.9 แสดงส่วนประกอบต่างๆของเอกสารสิทธิ์	25
รูปที่ 3.10 แสดงเอกสารสิทธิ์ที่ตรวจสอบโดยบราวเซอร์	29
รูปที่ 4.1 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS	31
รูปที่ 4.2 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS และ การส่งต่อ	31
รูปที่ 4.3 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS และ ไม่ปลอดภัยบน SMTP	32
รูปที่ 4.4 Security Element	32
รูปที่ 4.5 SOAP Message with Digital Signature	35
รูปที่ 4.6 SOAP Message with Encrypted Body	38
รูปที่ 4.7 Encrypted SOAP Message with Key	39
รูปที่ 5.1 แพลตฟอร์มการพัฒนาระบบภายใต้เทคโนโลยี .NET	42
รูปที่ 5.2 แพลตฟอร์มการพัฒนาระบบภายใต้เทคโนโลยี J2EE	43
รูปที่ 6.1 รูปแบบไคลเอนต์-เซิร์ฟเวอร์	48
รูปที่ 6.2 รูปแบบCode-on-Demand	49
รูปที่ 6.3 รูปแบบโมบายเอเจนต์	50
รูปที่ 6.4 ตลาด Aglet Marketplace	51
รูปที่ 6.5 คุณสมบัติ Agent	53
รูปที่ 6.6 Agent Model Pyramid	54
รูปที่ 6.7 HTTP เป็นอุโมงค์ให้ ATP	55
รูปที่ 6.8 การเคลื่อนย้ายเอเจนต์	56
รูปที่ 6.9 ความเกี่ยวข้องระหว่างโฮส, เซิร์ฟเวอร์ และคอนเท็กซ์	57
รูปที่ 6.10 ขั้นตอนชีวิตของ Aglet	59
รูปที่ 6.11 เมธอดที่ถูกเรียกใช้เมื่อมีการเกิด	59
รูปที่ 6.12 เมธอดที่ถูกเรียกใช้เมื่อมีการตาย	60

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

	หน้า
รูปที่ 6.13 เมธอดที่ถูกเรียกใช้เมื่อมีการย้าย	60
รูปที่ 6.14 เมธอดที่ถูกเรียกใช้เมื่อมีการเรียกกลับ	60
รูปที่ 7.1 ข้อความระหว่างผ่านพร๊อกซี่	61
รูปที่ 7.2 Synchronous Messaging	61
รูปที่ 7.3 Asynchronous Messaging	62
รูปที่ 7.4 ประเภทของข้อความจากซ้ายไปขวา	63
รูปที่ 8.1 ความคิดรวบยอดของ Distributed Object	65
รูปที่ 8.2 สถาปัตยกรรมของ RMI	66
รูปที่ 8.3 Tahiti เซิร์ฟเวอร์สำหรับ Aglet ที่มาพร้อมกับ GUI	69
รูปที่ 8.4 เซิร์ฟเวอร์สำหรับ Aglet ที่รวม RMI แล้ว	69
รูปที่ 10.1 ภาพรวมภายในของโครงงาน	77
รูปที่ 10.2 ภาพรวมของเว็บเซอร์วิส 3 โครงงาน	78
รูปที่ 10.3 แผนผังของระบบ	79
รูปที่ 10.4 ซีเควนซ์ไดอะแกรมของการซื้อ	79
รูปที่ 10.5 โฮมเพจหน้าร้าน	80
รูปที่ 10.6 เว็บเพจหน้ารายการสินค้า	80
รูปที่ 10.7 เว็บเพจหน้ารายการสั่งซื้อ	81
รูปที่ 10.8 เว็บเพจการชำระเงิน	81

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

เนื่องจากปัจจุบันเทคโนโลยีด้านคอมพิวเตอร์และเน็ตเวิร์คเปลี่ยนแปลงไป จากแรกเริ่มเดิมทีที่คอมพิวเตอร์ทำงานเหมือนเป็นเกาะ (Island) คือไม่มีระบบเครือข่ายเชื่อมต่อกับเครื่องอื่นๆ พัฒนามาเป็นเครือข่ายขนาดเล็กหรือกลางที่มีมาตรฐานหลากหลายแตกต่างกัน และมาถึงยุคปัจจุบันที่เป็นเครือข่ายอินเตอร์เน็ต (Internet) ที่เข้าถึงได้จากเครื่องคอมพิวเตอร์แบบต่างๆกันรวมไปถึงโทรศัพท์เคลื่อนที่บางระบบด้วย และอินเตอร์เน็ตก็เป็นพื้นฐานการทำงานแบบไคลเอนท์-เซิร์ฟเวอร์ (Client-Server) ซึ่งมีพัฒนาการจากเครื่องไคลเอนท์ที่ไม่ต้องประมวลผลอะไรนัก (Thin Client) และเซิร์ฟเวอร์ก็ทำหน้าที่ส่งข้อมูลที่ผู้ใช้ต้องการกลับไป กลายมาเป็นซีจีไอ (CGI:Common Gateway Interface) ซึ่งมีข้อจำกัดที่ผู้ใช้ต้องทำการป้อนข้อมูลด้วยตัวเอง หากเป็นข้อมูลที่มาจากเซิร์ฟเวอร์เดียวหรือไม่กี่เซิร์ฟเวอร์ก็ไม่ใช่ปัญหาสักเท่าใด แต่ในยุคที่ข้อมูลข่าวสารบนอินเทอร์เนตมีมากมาย มาจากแหล่งต่างๆ มีเซิร์ฟเวอร์นับล้านเครื่องทั่วโลก และผู้ใช้ก็จำเป็นต้องรับรู้ข้อมูลข่าวสารหรือบริการนั้น การทำงานแบบ ไคลเอนต์-เซิร์ฟเวอร์ นั้นสร้างความลำบากให้กับผู้ใช้ทีเดียว จึงมีแนวคิดที่จะให้เซิร์ฟเวอร์มีการ “คุย” กัน แลกเปลี่ยนข้อมูลกันเพื่อนำข้อมูลไปให้กับผู้ใช้ จึงเกิดเป็นมาตรฐานเว็บเซอร์วิสขึ้นมา ซึ่งมีความคิดรวบยอดที่ต้องการให้เครื่องเซิร์ฟเวอร์มีการส่งข้อความที่เป็นมาตรฐานเดียวกันทั้งหมด ไม่ว่าเซิร์ฟเวอร์นั้นจะสร้างด้วยโปรแกรมภาษาอะไรก็ตาม และเซิร์ฟเวอร์ทั้งหมดก็จะต้องสื่อสารกันได้โดยไม่ต้องใช้วิศวกรของทั้งสององค์กรตกลงกันก่อน และเทคโนโลยีโมบายเอเจนต์ (Mobile Agent) ก็เป็นเทคโนโลยีใหม่ที่ผู้พัฒนาต้องการให้ช่วยให้ภาระงานด้านเน็ตเวิร์คมีประสิทธิภาพมากขึ้น โดยใช้ความคิดรวบยอดที่ว่า เราน่าจะส่งตัวแทนไปทำงานที่หน้าเบื่อหรือต้องเฝ้าดูติดตามตลอดได้ โครงงานนี้จึงใช้ 2 เทคโนโลยีร่วมกันเพื่อให้เกิดประโยชน์แก่ผู้ใช้มากยิ่งๆขึ้น

1.2 วัตถุประสงค์ของโครงงาน

เพื่อเป็นการศึกษา การพัฒนา และการใช้งาน Web Service และโมบายเอเจนต์โดยมีจุดประสงค์ดังนี้

1. ศึกษาทฤษฎีพื้นฐานของคอมโพเนนท์
2. ศึกษาทำความเข้าใจในจาวาเทคโนโลยี ซึ่งปัจจุบันมีการพัฒนาถึงรุ่นที่ 5 (JDK1.5)
3. ศึกษาเรื่องเว็บเซอร์วิสและเทคโนโลยีที่เกี่ยวข้อง ได้แก่ SOAP, XML, WSDL และ UDDI เป็นต้น
4. ศึกษาการทำงานและประโยชน์ของโมบายเอเจนต์และการพัฒนา
5. ศึกษาการสร้างเว็บเซอร์วิสโดยลงลึกในเทคโนโลยีของเว็บเซอร์วิส เช่น Transaction, Stateful และความปลอดภัยในการเรียกใช้เว็บเซอร์วิส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. ศึกษาการเรียกใช้เว็บเซอร์วิสข้ามค่าย เนื่องจากเทคโนโลยีเว็บเซอร์วิสนั้นตั้งใจให้เป็นมาตรฐานที่ไม่ขึ้นอยู่กับระบบใดๆอยู่ตั้งแต่แรกแล้ว รวมทั้งเปรียบเทียบเทคโนโลยีทั้งสองค่าย จาวาและ .NET

1.3 ขอบเขตของโครงงาน

1. ออกแบบและสร้างแอพพลิเคชั่นโดยใช้สถาปัตยกรรมแบบ 3 เทียร์ ร่วมกับการใช้หลักการพัฒนาซอฟต์แวร์เชิงคอมโพเนนท์ด้วยเทคโนโลยีเว็บเซอร์วิสโดยใช้จาวา
2. พัฒนาโมบายเอเจนต์เพื่อให้ใช้ร่วมกับเว็บเซอร์วิส เพื่อเพิ่มความสามารถให้กับแอพพลิเคชั่น
3. สร้างแอพพลิเคชั่นที่ทำการเรียกใช้เว็บเซอร์วิสคอมโพเนนท์ข้ามแพลตฟอร์มได้

1.4 เทคนิค/เทคโนโลยีที่ใช้ในการพัฒนา

1. Web Services
   - XML
   - SOAP
   - UDDI
   - WSDL
   - WS-Security
2. Mobile Agent
   - IBM Aglet
3. RMI (Remote Method Invocation)
4. PKI (Public Key Infrastructor)

1.5 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ อยู่ 3 เรื่องด้วยกัน คือ เว็บเซอร์วิส โมบายเอเจ้นต์ และความปลอดภัย

เว็บเซอร์วิสจะศึกษาทฤษฎีพื้นฐานของมัน การใช้เครื่องมือ (Tools) การตั้งค่าเซิร์ฟเวอร์ โมบายเอเจนต์จะเริ่มศึกษาลักษณะการทำงาน และวิธีการเขียนโปรแกรม ความปลอดภัยจะศึกษาเรื่องความจำเป็นในการรักษาความปลอดภัย

ต่อมาจะศึกษาการนำโมบายเอเจนต์มาใช้เชื่อมต่อกับเว็บเซอร์วิส และศึกษาเรื่องการใช้เว็บเซอร์วิสด้วยความปลอดภัย

ขั้นสุดท้ายคือนำองค์ความรู้ทั้งหมดในขั้นต้นมาสร้างเป็นแอพลิเคชั่นที่มีความซับซ้อนและแสดงถึงคุณลักษณะของสิ่งที่ได้ศึกษามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

Web Services

2.1 การพัฒนาของเว็บ

การพัฒนาทางเทคโนโลยีใด ๆ ล้วนแล้วแต่มีพัฒนาการเป็นยุคสมัย แต่ละยุคสมัยมีการใช้เทคโนโลยีที่มีจุดสำคัญของการพัฒนาที่เด่นชัด การพัฒนาเทคโนโลยีการประยุกต์ใช้งานบนเว็บก็เช่นเดียวกัน

2.1.1 สแตติกเว็บเพจ (Static Web Page)

เป็นยุคที่เริ่มต้นการใช้งานเว็บมีการวางข้อมูลตามมาตรฐานเอชทีเอมแอล (HTML) ไว้เป็นไฟล์บนเซิร์ฟเวอร์ ผู้ใช้งานใช้บราวเซอร์ (browser) เรียกข้อมูลด้วยโปรโตคอลเอชทีทีพี การเรียกใช้ข้อมูลจะได้ข้อมูลที่เก็บไว้ บนเซิร์ฟเวอร์ โดยลักษณะของข้อมูลยังเป็นข้อมูลแบบสแตติก คือเก็บไว้เพื่อรอการเรียกใช้ บนฝั่งของเซิร์ฟเวอร์เองยัง ไม่มีการฝังโปรแกรมพิเศษใด ๆ เพียงแต่มีโปรแกรมที่รับการติดต่อด้วยเอชทีทีพีเท่านั้นข้อมูลที่เรียกใช้ได้รับมาเป็น ไฟล์ โดยทางฝั่งไคลเอนต์จะนำไปแสดงผลตามข้อมูลที่ปรากฏในเท็กซ์ (text)ตามมาตรฐาน ในยุคที่หนึ่งนี้ทำให้มีการเก็บข้อมูลบนเซิร์ฟเวอร์เพื่อเป็นจุดที่ทำให้ผู้ใช้งานเรียกเข้าใช้ได้ง่ายการพัฒนาระบบบริการข้อมูลแบบเว็บเพจ (web page) หรือ โฮมเพจ (home page) จึงได้รับความสนใจ ทำให้มีผู้สร้างเว็บเพจและผู้ใช้ผ่านบราวเซอร์เป็นจำนวนมาก

2.1.2 ไดนามิกเว็บเพจ (Dynamic Web Page)

มีการพัฒนาโปรแกรมทางฝั่งเซิร์ฟเวอร์ เพื่อให้เซิร์ฟเวอร์ทำงานตามคำเรียกขอมาจากทางไคลเอนต์ผ่านทางโปรโตคอลเอชทีทีพี หรือการเรียกของบนเว็บ มีการเชื่อมโยงกับโปรแกรมเฉพาะ เช่น เมื่อผู้เรียกใช้คลิกบนบราวเซอร์ ทำให้เกิดการเรียกขอมาที่เซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำงานตอบสนอง โดยอาจจะไปรันโปรแกรม หรือเรียกโปรแกรมอื่นที่ทางฝั่งเซิร์ฟเวอร์เตรียมไว้ ในยุคสมัยนี้มีการสร้างระบบการเรียกเข้าหาโปรแกรมที่เรียกว่า ซีจีไอ หรือ คอมมอลเกตเวย์อินเทอร์เฟส (CGI-Common Gateway Interface) การเขียนโปรแกรมมีเทคนิควิธีการพิเศษที่ทำให้เข้าถึงฐานข้อมูล มีการพัฒนาเครื่องมือการพัฒนา และภาษาสำหรับพัฒนาเพื่อทำให้เรียกข้อมูลจากฐานข้อมูลหรือเรียกแฟ้มข้อมูลอื่นเพื่อส่งต่อให้ผู้เรียกใช้ ภาษาที่นิยมใช้ในการพัฒนามีภาษาซี (C Language) ภาษาเพิร์ล (PERL Language) และเครื่องมือพิเศษที่ใช้เชื่อมโยงกับฐานข้อมูลที่ผู้ผลิตซอฟต์แวร์ทางฐานข้อมูลดำเนินการให้ ตลอดจนมี80การใช้โปรแกรมทางด้านไคลเอนต์ที่เริ่มซับซ้อนขึ้นการที่เรียกว่าไดนามิกเว็บเพจก็ด้วยเหตุผลที่ว่าข้อมูลที่เรียกดูจะแปรเปลี่ยนตามการเรียกขอ การปรับเปลี่ยนข้อมูลกระทำตามโปรแกรมที่สร้างขึ้น สามารถคำนวณหรือประมาณผลข้อมูลได้ เช่น การดูข้อมูลสรุปของแต่ละวัน ดังนั้นการเรียกดูข้อมูลจะได้ผลที่ทันสมัยได้ ระบบเซิร์ฟเวอร์จึงเชื่อมโยงกับฐานข้อมูลหรือระบบงานอื่น แต่เมื่อจะส่งข้อมูลไปยังผู้ใช้ที่เรียกขอเข้ามาจะแปลงให้อยู่ในรูปแบบของเอชทีเอมแอลก่อนเพื่อให้บราวเซอร์แสดงผลได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3 บริการบนเว็บ (Web Service)

การพัฒนาเว็บมาจนถึงปัจจุบันได้ก้าวหน้าขึ้นไปอีกมาก จุดสำคัญอยู่ที่ความต้องการรองรับการใช้งานที่มีความต้องการสูงขึ้น โดยเฉพาะการดำเนินการทางธุรกิจการค้าต่างๆ ที่เพิ่มมากขึ้นบนเครือข่าย ลักษณะงานที่ต้องการมีลักษณะการทำงานร่วมกันระหว่างองค์กร (Interoperability) โดยให้โปรแกรมประยุกต์ขององค์กรหนึ่งส่งคำขอผ่านเครือข่ายอินเทอร์เน็ตด้วยโปรโตคอลเอชทีทีพี ไปยังเว็บบริการขององค์กรหนึ่งมีการโต้ตอบเพื่อรับส่งข้อมูลระหว่างกันแบบอัตโนมัติได้ การเรียกใช้บริการต่างๆ มีลักษณะการเรียกเข้าบริการ โดยมีตัวกลางช่วยและเชื่อมโยงเข้ากับเว็บเซิร์ฟเวอร์ การแลกเปลี่ยนข้อมูลตามคำเรียกขอใช้มาตรฐานเอ็กซ์เอ็มแอล หรือเอ็กซ์เทนซิเบิ้ลมาร์กอัพแลงเกวด (XML-eXtensible Mark Up Language) ข้อมูลที่เป็นเอ็กซ์เอ็มแอลจะถูกส่งผ่านมาภายใต้การเรียกขอเอชทีทีพี การบริการบนเว็บจึงเป็นการก้าวเข้ามาในระบบที่ทำให้เกิดการแลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งแต่เดิมมีวิธีการแลกเปลี่ยนข้อมูลกันตามรูปแบบที่เรียกว่า อีดีไอ (EDI) แต่จากการที่มี การใช้เอ็กซ์เอ็มแอล ทำให้การทำงานได้ตรงตามความต้องการลักษณะสำคัญของการบริการบนเว็บจึงเป็นงานประยุกต์ที่เซิร์ฟเวอร์บริการสามารถเชื่อมโยงไปกับการประยุกต์อื่น ๆ โดยสามารถเรียกหาข้อมูลจากที่อื่นเพื่อนำมาประมวลผล และส่งต่อให้ผู้ใช้ที่เรียกเข้ามาต่อไป

2.2 เว็บเซอร์วิส

การพัฒนาทางเทคโนโลยีใดๆ ล้วนแล้วแต่มีการพัฒนาเป็นยุคสมัย แต่ละยุคสมัยก็จะมีการใช้เทคโนโลยีที่มีจุดสำคัญของการพัฒนาที่เด่นชัด การพัฒนาเทคโนโลยีประยุกต์ใช้งานบนเว็บก็เช่นเดียวกัน เริ่มจากยุคแรกที่เป็นแบบสแตติกเว็บเพจที่เป็นเพียงแค่ไฟล์เอชทีเอมแอลไว้ให้ผู้ใช้ดูข้อมูลเฉยๆ จนสู่ยุคที่ 2 เป็นแบบไดนามิกเว็บเพจ (Dynamic Web Page) ที่สามารถแปรเปลี่ยนตามคำร้องขอของผู้ใช้งานได้ ซึ่งเป็นการเพิ่มประสิทธิภาพของเว็บเพจ จนกระทั่งในปัจจุบันมีการพัฒนาเว็บเพจเข้ามาสู่ยุคที่ 3 หรือแบบการบริการผ่านเว็บเซอร์วิส โดยจุดประสงค์เพื่อรองรับความต้องการการใช้งานที่สูงขึ้น และเป็นมาตรฐานในการทำงานร่วมกัน

เว็บเซอร์วิส คือ ซอฟต์แวร์คอมโพแนนท์ (Software Component) ที่ผู้ให้บริการนำมาสร้างเป็นแอพพลิเคชั่นสำหรับให้บริการการทำงานหนึ่งๆ ให้กับผู้ขอบริการทางอินเทอร์เน็ต หรือผู้ขอบริการสามารถที่จะขอบริการจากหลายๆที่มาประกอบกันได้ โดยที่แต่ละระบบนั้นมีความเป็นอิสระจากกัน (Loosely Coupled)ซึ่งปัจจัยพื้นฐานของเว็บเซอร์วิสที่ควรมี ซึ่งมีความเกี่ยงข้องกันในทางเทคนิคและทางธุรกิจคือ

- การรวมซอฟต์แวร์ต่างระบบกันจะต้องอนุญาตให้แต่ละระบบมีความเป็นอิสระจากกัน
- อินเทอร์เฟสทางด้านการบริการของซอฟต์แวร์ควรจะเผยแพร่สู่สาธารณชนและสามารถเข้าถึงได้ง่าย
- แมสเซจที่ใช้ติดต่อกันของการทำงานแบบแอพพลิเคชั่นกับแอพพลิเคชั่นต้องสอดคล้องกับมาตรฐานเปิดบนอินเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- แอพพลิเคชั่นสามารถสร้างได้จากการใช้ซอฟต์แวร์คอมโพแนนท์ทั้งจากภายในและภายนอกองค์กร โดยสร้างตามการดำเนินธุรกิจหลักขององค์กร

- มีแหล่งซอฟต์แวร์คอมโพแนนท์ที่หาได้ง่าย ซึ่งช่วยเพิ่มความยืดหยุ่นในการสร้างแอพพลิเคชั่นที่มีคุณลักษณะเฉพาะตามกระบวนการดำเนินการทางธุรกิจ

- การนำซอฟต์แวร์จากภายนอกมาใช้ใหม่ช่วยให้เกิดการลดต้นทุนและช่วยปรับปรุงบริการให้แก่ลูกค้า

- ซอฟต์แวร์สามารถขายเป็นบริการได้

2.2.1 หลักการพื้นฐานของเทคโนโลยีต่างๆ ที่พัฒนามาสู่เว็บเซอร์วิส

- การพัฒนาโปรแกรมแบบซอฟต์แวร์คอมโพแนนท์ ตามแนวคิดของการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ (Object-Oriented Concept)

- การออกแบบระบบแบบกระจายจากศูนย์กลาง (Distributed Computing) ซึ่งเป็นเป้าหมายสำคัญของการพัฒนาระบบตามสถาปัตยกรรมแบบไคลเอนต์เซิร์ฟเวอร์ (Client-Server)

- การทำอีดีไอ หรือ อิเล็กทรอนิกส์คาต้าอินเทอร์เช้นต์ (Electronic Data Interchange) ซึ่งสร้างขึ้นโดยกำหนดรูปแบบและมาตรฐานของข้อมูลสำหรับการทำธุรกิจ

- การบูรณาการของซอฟต์แวร์ต่างระบบ หรือ เอนเทอร์ไพรส แอพพลิเคชั่น อินทิเกรชั่น : อีเอไอ (Enterprise Application Integration : EAI) ที่อยู่บนพื้นฐานของความต้องการใช้ข้อมูลร่วมกันรวมทั้งการแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นให้สามารถทำงานได้อย่างถูกต้องเหมาะสม

- รูปแบบการให้บริการซอฟต์แวร์แบบเอเอสพี (ASP) หรือ แอพพลิเคชั่นเซอร์วิสโพรไวเดอร์ (Application Service Provider)

- แนวคิดการทำเหมืองข้อมูล (Data Mining) ซึ่งต้องการนำข้อมูลที่ถูกจัดเก็บไว้ในรูปแบบที่แตกต่างกันตามแหล่งต่างๆมาใช้งานร่วมกัน โดยที่เว็บเซอร์วิสได้นำหลักการต่างๆ เหล่านี้มาดำเนินการปรับปรุง แก้ไขข้อจำกัดต่างๆ โดยการคิดค้นเทคโนโลยีที่เป็นมาตรฐานเปิด (Opened Standard) และไม่มีใครเป็นเจ้าของ (Neural Vendor)

2.2.2 ลักษณะของเว็บเซอร์วิส

การพัฒนาเว็บเซอร์วิสใช้สถาปัตยกรรมบริการในลักษณะที่เรียกว่า สถาปัตยกรรมของแนวคิดทางด้านบริการ หรือ เอสโอเอ (Service-Oriented Architecture : SOA) เป็นแนวคิดเบื้องต้น แอพพลิเคชั่นส่วนใหญ่ในโลกของธุรกิจที่ใช้งานในปัจจุบัน เป็นแอพพลิเคชั่นและระบบย่อยที่ถูกสร้างขึ้น มีการทำงานที่ต้องสัมพันธ์กันอย่างไม่เป็นอิสระจากกัน การเปลี่ยนแปลงการทำงานในระบบย่อยหรือแอพพลิเคชั่นหนึ่ง อาจจะมีผลกระทบกับอีกแอพพลิเคชั่นหนึ่ง หรือบางครั้งอาจจะกระทบทั้งระบบ ทำให้การบำรุงรักษานั้นทำได้ยาก และ มีต้นทุนที่สูงขึ้น รวมทั้งยังเป็นข้อจำกัดในการเชื่อมต่อกับระบบของคู่ค้าอื่นๆ เอสโอเอไม่ใช่แนวคิดใหม่ แต่ได้เกิดขึ้นมานานแล้ว ซึ่งอยู่ในส่วนหนึ่งของแนวคิดการออกแบบระบบแบบกระจายจากศูนย์กลาง (Distributed Computing Concepts) เอสโอเอนั้น เป็นแนวคิดที่ได้รับการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ยอมรับและประสบความสำเร็จในระดับหนึ่ง การพัฒนาเอสโอเอ นั้นทำกันบนมาตรฐานเปิด ซึ่งได้รับการรับรองจากผู้ผลิตซอฟต์แวร์ชั้นนำและยังได้ร่วมมือกันรับรองมาตรฐานของเทคโนโลยีต่างๆ ที่ประกอบกันเป็นเว็บเซอร์ วิส เช่น ซิมเปิ้ลออบเจ็กต์แอคเซสโปรโตคอล (SOAP), ดับบิวเอสดีแอล (WSDL) และ ยูดีดีไอ (UDDI) เป็นต้ น

เอสโอเอมีส่วนประกอบหลัก 3 ส่วนคือ

1. ผู้ให้บริการ หรือที่เรียกว่า เซอร์วิสโพรไวเดอร์ (Service Provider)

2. ผู้ขอใช้บริการ หรือที่เรียกว่า เซอร์วิสรีเควสเตอร์ (Service Requester)

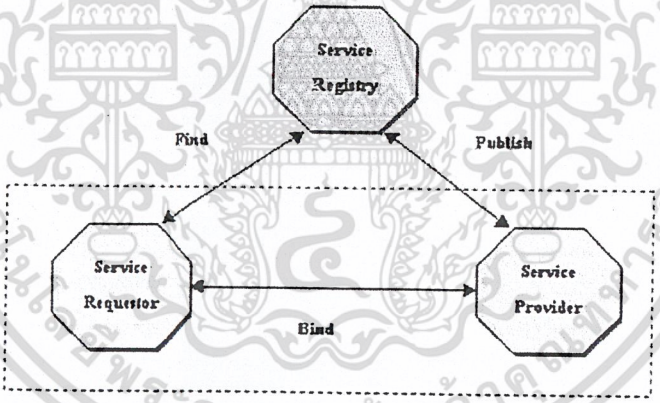
3. ตัวแทนของผู้ให้บริการ หรือที่เรียกว่า เซอร์วิสโบรกเคอร์ (Service Broker)

โดยส่วนประกอบหลักทั้ง 3 ส่วนนี้ สามารถติดต่อถึงกันได้โดยใช้ฟังก์ชันพื้นฐาน (Primary Function) คือการประกาศ (Publish) การค้นหา (Find) และการเรียกใช้ (Bind) ซึ่งฟังก์ชันทั้ง 3 มีการทำงานดังนี้

1. ผู้ให้บริการทำการประกาศบริการที่ตนเองให้บริการ ไปยังตั วแทนของผู้ให้บริการ ซึ่งตัวแทนของผู้ให้บริการจะทำการบันทึกเก็บไว้ในไดเรกทอรีของบริการ (Directory Service)

2. ผู้ขอใช้บริการจะทำการค้นหาบริการจากตัวแทนของผู้ให้บริการ

3. เมื่อพบบริการที่ต้องการแล้ว ผู้ให้บริการและผู้ขอใช้บริการจะทำการติดต่อกัน โดยผู้ขอใช้บริการจะทำการเรียกใช้บริการไปยังผู้ให้บริการนั้น



รูปที่ 2.2 แสดงความสัมพันธ์ของผู้ให้บริการ ผู้ขอใช้บริการ และตัวแทนผู้ให้บริการ

2.2.3 เทคโนโลยีพื้นฐานของเว็บเซอร์วิส

จากแนวคิดของเอสโอเอ ถูกนำมาใช้เป็นหลักการพื้นฐานของการพัฒนาเทคโนโลยีด้านเว็บเซอร์วิสซึ่งเทคโนโลยีที่ถูกพัฒนาขึ้นมานี้เป็นมาตรฐานเปิดบนอินเทอร์เน็ตที่เกิดจากการทำงานร่วมกันของนักวิจัยและที่ปรึกษาจากบริษัทซอฟต์แวร์ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 มาตรฐานที่ใช้ในเว็บเซอร์วิส

2.3.1 XML (eXtensible Markup Language)

ลักษณะที่สำคัญของภาษา XML

- สิ่งที่แตกต่างระหว่าง XML และ HTML คือ XML ถูกออกแบบมาเพื่อใช้แลกเปลี่ยนข้อมูล ไม่ใช่มาแทนที่ HTML ซึ่ง XML และ HTML ถูกออกแบบมาเพื่อจุดประสงค์ที่ต่างกัน คือ HTML จะเกี่ยวข้องกับการแสดงข้อมูล ส่วน XML จะเกี่ยวข้องกับการอธิบายข้อมูล
- XML ไม่มีแท็ก (tag) ที่สร้างขึ้น เราจะเป็นผู้กำหนดนิยาม และความหมายของแท็กตามข้อตกลงของ W3C ซึ่งทำให้การอ่านเอกสารเป็นไปได้ง่ายและเป็นมาตรฐาน
- XML สามารถรองรับการแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่น โดยไม่ขึ้นอยู่กับแพลตฟอร์ม
- XML เป็นภาษาที่รับการออกแบบมาเพื่อให้สามารถนิยามความหมายของข้อมูลได้ จึงมีการจัดโครงสร้างข้อมูล แบ่งข้อมูลออกเป็นหมวดหมู่ และส่วนประกอบย่อย
- ส่วนข้อมูล และส่วนการแสดงผลของเอกสาร XML ถูกแยกออกจากกันอย่างชัดเจน โดยที่ในเอกสาร XML นั้นจะมีแต่ตัวเนื้อข้อมูล จึงทำให้การเปลี่ยนแปลงข้อมูลไม่ส่งผลกระทบต่อการแสดงผล และการแก้ไขส่วนแสดงผลก็จะไม่ส่งผลกระทบกับตัวเนื้อข้อมูล
- การอ่านและแปลความหมายของเอกสาร XML สามารถทำได้โดยใช้โปรแกรมที่เรียกว่า XML parser
- XML มีความกะทัดรัด เข้าใจง่าย และใช้ประโยชน์ได้กว้างขวาง
- XML สามารถใช้ได้หลายภาษาผสมกัน เนื่องจาก XML สนับสนุน UNICODE
- XML ได้รับการสนับสนุนในโปรแกรมระบบฐานข้อมูลหลายๆ บริษัทให้สามารถดึงข้อมูลจากฐานข้อมูลมาอยู่ในรูปของ XML ได้

โครงสร้างของ XML ประกอบด้วย 3 ส่วน คือ

1. ส่วน Prolog เป็นส่วนที่ใช้ในการ Declaration
2. ส่วน Body เป็นส่วนของข้อมูลจริง
3. ส่วน Epilog เป็นส่วนที่ข้อความคอมเม้นท์ (comment) และ P1 (Processing Instruction) ซึ่งจริงๆ แล้วจะแทรกอยู่ในส่วนบอดี้ (Body) ของเอกสาร

2.3.2 SOAP

SOAP (Simple Object Access Protocol) เป็น XML-based โปรโตคอล (lightweight protocol) สำหรับการแลกเปลี่ยนข้อมูลในสภาวะแวดล้อมแบบกระจายศูนย์ (decentralized,distributed environment) SOAP ได้กำหนด เมเสจจิ้งโปรโตคอล ระหว่างผู้ขอบริการ (requester) กับผู้ให้บริการ (provider)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SOAP เป็นโพรโตคอลน้ำหนักเบาสำหรับกระจายข้อมูล ซึ่งก็คือ โพรโตคอล XML-base ประกอบด้วย 3 ส่วนคือ

1. เครื่องห่อหุ้ม (envelope) ซึ่งกำหนดเค้าโครงสำหรับอธิบายข้อความ และ มีกระบวนการอย่างไรบ้าง

2. กฎการเข้ารหัสสำหรับแสดงตัวอย่างของการกำหนดแอพพลิเคชั่น

3. รูปแบบสำหรับการแทนฟังก์ชันและตอบสนอง

SOAP มีขนาดเล็ก สามารถยืดขยายได้ เป็นโพรโตคอลพื้นฐานของ XML สำหรับการแลกเปลี่ยนข้อมูล ขั้นแรก SOAP จะกำหนดโครงสร้างและรูปแบบของข้อมูล และ ยังกำหนดเซตสำหรับกฎการเข้ารหัสข้อมูลที่เรียงต่อกัน และมีรูปแบบสำหรับการเรียกฟังก์ชัน รูปแบบของ SOAP ได้เพิ่มความสามารถในการติดต่อระหว่างโพรโตคอลอย่างแพร่หลาย เช่น โพรโตคอล HTTP

ข้อดีของโพรโตคอล SOAP

โพรโตคอล SOAP มีความสามารถให้เราเรียกใช้คอมโพเนนท์ หรือเว็บเซอร์วิสข้ามเครื่อง ข้ามแพลตฟอร์ม ข้ามภาษาได้อย่างสบาย โดยอาศัยโพรโตคอลที่มีอยู่เดิมในอินเทอร์เน็ตอย่าง HTTP และรูปแบบข้อความที่สื่อสารกันด้วยภาษา XML ซึ่งมีลักษณะเป็นข้อความธรรมดา ๆ ปิดล้อมด้วยแท็ก ทำให้เข้าใจได้ในทุกแพลตฟอร์ม ขอเพียงมี XML Parser มาแปลข้อความนั้น ก็เข้าใจได้แล้ว

นอกจากนี้ SOAP ยังมีข้อดีอีกอย่างหนึ่งที่เหนือกว่าโพรโตคอลแบบเดิม เช่น DCOM, RMI หรือ IIOP คือ ข้อความ SOAP สามารถผ่านระบบที่มีไฟร์วอลล์ (firewall) ป้องกันการบุกรุก ทั้งนี้เนื่องจากข้อความ SOAP ทำงานอยู่กับโพรโตคอล HTTP ซึ่งโดยธรรมชาติของไฟร์วอลล์จะเปิดให้การสื่อสารด้วยโพรโตคอล HTTP ผ่านได้อย่างสะดวก ดังนั้นข้อความ SOAP จึงผ่านได้เช่นกัน โดยไม่มีปัญหา ในขณะที่โพรโตคอลแบบเดิม ๆ เหล่านั้น ไฟร์วอลล์มักจะไม่ยอมให้ผ่านง่าย ๆ ซึ่งจุดนี้คือเป็นข้อเสียที่สำคัญที่สุดและทำให้โพรโตคอล SOAP ได้เปรียบขึ้นมาทันที

ข้อเสียของโพรโตคอล SOAP

ยังมีความจริงที่ปฏิเสธไม่ได้ว่า SOAP มีจุดด้อยอยู่บางประการ ประการแรก เนื่องจากลักษณะของข้อความ SOAP เป็นเอกสาร XML ที่ให้เสียเวลาในการแปลกลับมาเป็นรูปแบบที่โปรแกรมเข้าใจ และประการที่สองในกรณีที่ SOAP ทำงานอยู่กับโพรโตคอล HTTP ซึ่งโดยธรรมชาติแล้วโพรโตคอลนี้มีสมรรถนะในการรับ-ส่งข้อมูลต่ำโดยเฉพาะเมื่อเปรียบเทียบกับโพรโตคอลอื่น ๆ อย่าง IIOP

ด้วยเหตุนี้โพรโตคอล SOAP จึงมีอัตราการรับ-ส่งข้อมูลที่ต่ำด้วย และเหตุผลทั้งสองประการนี้ทำให้อัตราการรับส่งข้อมูลต่ำกว่าโพรโตคอล DCOM, RMI หรือ IIOP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.3 UDDI

UDDI (Universal Description, Discovery and Integration) เป็นโปรโตคอลที่ค่ายคอมพิวเตอร์ 4 ค่ายยักษ์ คือ ไอบีเอ็ม ซัน ไมโครซิสเต็มส์ ฮิวเลตแพคการ์ด และอะริบา พยายามผลักดันให้เป็นมาตรฐานอินเทอร์เน็ตสากล UDDI ถูกสร้างขึ้นมาเพื่อเป็นมาตรฐานในการค้นหาบริการเว็บเซอร์วิสสำหรับคู่ค้าทางธุรกิจ (business partner) โดย UDDI เปรียบเสมือนฐานข้อมูลขนาดใหญ่ซึ่งมีข้อมูลของเว็บเซอร์วิสที่ให้บริการทางธุรกิจ

รูปแบบการทำงานของยูดีดีไอจะเปลี่ยนแปลงวิธีการค้นหาข้อมูลบนอินเทอร์เน็ตที่เราใช้อยู่ในปัจจุบันจากการพิมพ์ชื่อ โดเมนเนมในช่องแอดเดรสบาร์ ไปเป็น การใช้คำสามัญหรือชื่อเฉพาะในการค้นหา และเข้าสู่เว็บไซต์แทน ยูดีดีไอจะทำหน้า ที่เหมือนสมุดหน้าเหลือง ที่จำแนกผู้จดทะเบียนตามรูปแบบของธุรกิจแยก ประเภทตามกิจการ

โดยสรุปยูดีดีไอเหมือนคือโครงสร้างพื้นฐานที่มารองรับการค้นหาข้อมูลแบบใหม่ โดยใช้ตัวเลขหรือคำสามัญแทน โดเมนเนม สำหรับโดเมนเนมเองก็ยังคงอยู่ไม่ได้หายไปไหน เพียงแต่ยูดีดีไอจะมาเป็นอีกมาตรฐานหนึ่งในการติดต่อกับ IP Address เหมือนที่เราเคยมีโดเมนเนมหรือยูอาร์แอล ในการทำให้ผู้ที่เข้าสู่เว็บไซต์รู้จักและจดจำได้โดยง่าย โดยที่ เว็บไซต์สำหรับค้นหา เว็บเซอร์วิส มีอยู่หลายที่อย่าง เช่น

- http://uddi.microsoft.com/search.aspx
- http://www-3.ibm.com/services/uddi/testregistry/find
- http://uddi.org

2.3.4 WSDL

WSDL ย่อมาจาก Web Services Description Language คือข้อมูลรูปแบบ XML ที่ใช้สำหรับอธิบายบริการบนเครือข่ายที่ให้บริการ เราสามารถใช้ WSDL เพื่อสร้างไฟล์ที่ระบุถึงบริการที่เซิร์ฟเวอร์ให้บริการและกลุ่มปฏิบัติการภายในบริการแต่ละตัวซึ่งเซิร์ฟเวอร์สนับสนุนสำหรับปฏิบัติการแต่ละตัว ไฟล์ WSDL ยังทำหน้าที่อธิบายรูปแบบข้อมูลที่ไคลเอนท์ต้องปฏิบัติตามในการร้องขอปฏิบัติการ คิดค้นขึ้นโดย 2 บริษัทยักษ์ใหญ่ คือ ไอบีเอ็ม และไมโครซอฟต์

เนื่องจากไฟล์ WSDL ทำหน้าที่กำหนดความต้องการทั้งฝั่งเซิร์ฟเวอร์และไคลเอนท์ ไฟล์นี้จึงเสมือนเป็นสัญญาสำหรับเครื่องทั้งสองฝั่ง เซิร์ฟเวอร์ตกลงที่จะให้บริการที่กำหนดไว้เมื่อไคลเอนท์ส่งคำร้องขอ SOAP ในรูปแบบที่เหมาะสมเท่านั้น

ในทางปฏิบัติหากเราต้องการสร้างเว็บเซอร์วิส ขึ้นมาเป็นของตนเอง ก็สามารถสร้างเอกสาร WSDL ได้โดยอัตโนมัติ เราจึงไม่ต้องไปกังวลในรายละเอียด ในข้อกำหนดใน WSDL มากนัก (สามารถหาอ่านเพิ่มเติมได้จากเว็บไซต์ของ W3C ที่ http://www.w3.org/TR/wsdl)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

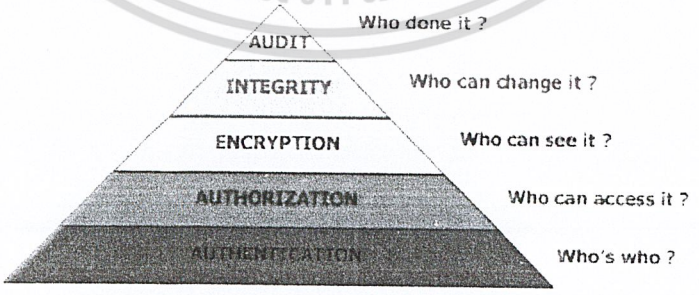
Security

3.1 นิยามความมั่นคงปลอดภัยคอมพิวเตอร์

ในปัจจุบันระบบคอมพิวเตอร์ได้ถูกคุกคามมากขึ้นทั้งจากไวรัสคอมพิวเตอร์หรือจากผู้ไม่ประสงค์ดี ซึ่งความมั่นคงปลอดภัยคอมพิวเตอร์ (Computer Security) ช่วยปกป้องเครื่องคอมพิวเตอร์รวมไปถึงอุปกรณ์ต่างๆที่เกี่ยวข้อง และที่สำคัญยังสามารถช่วยปกป้องข้อมูลที่ได้จัดเก็บไว้ภายในระบบหรือใช้ในความหมายความปลอดภัยทางข้อมูลสารสนเทศ (Information Security) ก็ได้ จุดประสงค์หลักของความปลอดภัยทางข้อมูลคือ ความลับ (Confidentiality) ความสมบูรณ์ (Integrity) ความพร้อมใช้ (Availability) และการห้ามปฏิเสธความรับผิดชอบ (Non-Repudiation) ของข้อมูลต่างๆภายในองค์กร (CIA-N) โดยมีรายละเอียดดังนี้

- **การรักษาความลับ (Confidentiality)** คือการรับรองว่าจะมีการเก็บข้อมูลไว้เป็นความลับ และผู้มีสิทธิเท่านั้นจึงจะเข้าถึงข้อมูลนั้นได้
- **การรักษาความสมบูรณ์ (Integrity)** คือการรับรองว่าข้อมูลจะไม่ถูกเปลี่ยนแปลงหรือทำลายไม่ว่าจะเป็นโดย อุบัติเหตุหรือโดยเจตนา
- **ความพร้อมใช้ (Availability)** คือการรับรองว่าข้อมูลและบริการการสื่อสารต่าง ๆ พร้อมที่จะใช้ได้ในเวลาที่ต้องการใช้งาน
- **การห้ามปฏิเสธความรับผิดชอบ (Non-Repudiation)** คือวิธีการสื่อสารซึ่งผู้ส่งข้อมูลได้รับหลักฐานว่าได้มีการส่งข้อมูลแล้ว และผู้รับก็ได้รับการยืนยันว่าผู้ส่งเป็นใคร ดังนั้นทั้งผู้ส่ง และผู้รับจะไม่สามารถปฏิเสธได้ว่าไม่มีความเกี่ยวข้องกับข้อมูลดังกล่าวในภายหลัง

ในทางปฏิบัตินั้นสามารถกำหนดลักษณะของการควบคุมความมั่นคงปลอดภัย (Security Controls) ได้ 5 ระดับตามรูป



รูปที่ 3.1 แสดง Security Pyramid

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

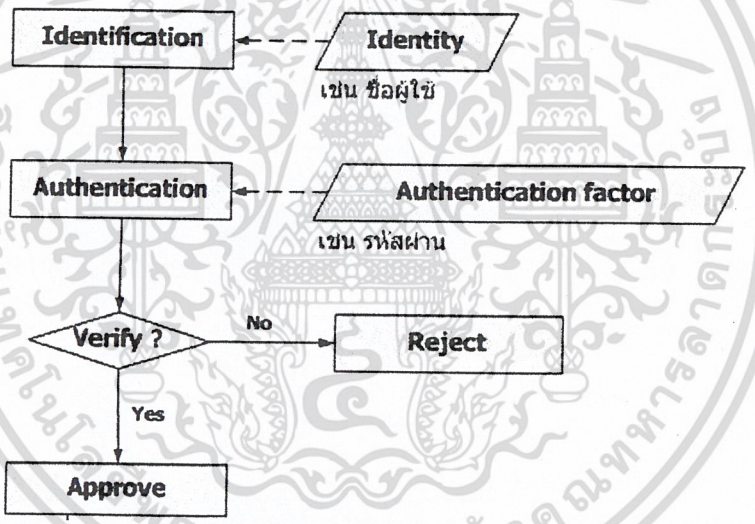
และถือเป็นองค์ประกอบที่สำคัญส่วนหนึ่งของความมั่นคงปลอดภัยคอมพิวเตอร์ เพราะจัดเป็นการกำหนดและควบคุมทั้งบุคคลที่สามารถเข้าสู่ระบบและเข้าสู่ข้อมูลภายในระบบ และเพื่อกระทำการใดได้บ้าง อนุญาตตามระดับชั้นของความสำคัญของข้อมูล รวมไปถึงการจัดเก็บพฤติกรรมการใช้งานระบบของบุคคลนั้นต่อข้อมูลบนระบบทั้งหมด

3.2 การพิสูจน์ตัวตน (Authentication)

การพิสูจน์ตัวตน คือขั้นตอนการยืนยันความถูกต้องของหลักฐาน (Identity) ที่แสดงว่าเป็นบุคคลที่กล่าวอ้างจริง ในทางปฏิบัติจะแบ่งออกเป็น 2 ขั้นตอน คือ

- การระบุตัวตน (Identification) คือขั้นตอนที่ผู้ใช้แสดงหลักฐานว่าตนเองคือใครเช่น ชื่อผู้ใช้ (username)

การพิสูจน์ตัวตน (Authentication) คือขั้นตอนที่ตรวจสอบหลักฐานเพื่อแสดงว่าเป็นบุคคลที่กล่าวอ้างจริง



รูปที่ 3.2 กระบวนการพิสูจน์ตัวตน

จากแผนผังแสดงกระบวนการพิสูจน์ตัวตน ในขั้นแรกผู้ใช้จะทำการแสดงหลักฐานที่ใช้ในการพิสูจน์ตัวตนต่อระบบ ซึ่งในขั้นนี้คือการระบุตัวตน และในขั้นตอนต่อมาระบบจะทำการตรวจสอบหลักฐานที่ผู้ใช้นำมากล่าวอ้างซึ่งก็คือการพิสูจน์ตัวตน หลังจากระบบได้ทำการตรวจสอบหลักฐานเรียบร้อยแล้วถ้าหลักฐานที่นำมากล่าวอ้างถูกต้องจึงอนุญาตให้เข้าสู่ระบบได้ หากหลักฐานที่นำมากล่าวอ้างไม่ถูกต้องผู้ใช้จะถูกปฏิเสธจากระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลักฐานที่ผู้ใช้นำมากล่าวอ้างที่เกี่ยวกับเรื่องของความปลอดภัยนั้นสามารถจำแนกได้ 2 ชนิด

- Actual identity คือหลักฐานที่สามารถบ่งบอกได้ว่าในความเป็นจริงบุคคลที่กล่าวอ้างนั้นเป็นใคร
- Electronic identity คือหลักฐานทางอิเล็กทรอนิกส์ซึ่งสามารถบ่งบอกข้อมูลของบุคคลนั้นได้ แต่ละบุคคลอาจมีหลักฐานทางอิเล็กทรอนิกส์
  ได้มากกว่า 1 หลักฐาน ตัวอย่างเช่น บัญชีชื่อผู้ใช้

กลไกของการพิสูจน์ตัวตน (Authentication mechanisms) สามารถแบ่งออกได้เป็น 3 คุณลักษณะคือ

- สิ่งที่คุณมี (Possession factor) เช่น กุญแจหรือเครดิตการ์ด เป็นต้น
- สิ่งที่คุณรู้ (Knowledge factor) เช่น รหัสผ่าน (passwords) หรือการใช้พิน (PINs) เป็นต้น
- สิ่งที่คุณเป็น (Biometric factor) เช่น ลายนิ้วมือ รูปแบบเรตินา (retinal patterns) หรือใช้รูปแบบเสียง (voice patterns) เป็นต้น

กระบวนการพิสูจน์ตัวตนนั้นจะนำ 3 ลักษณะข้างต้นมาใช้ในการยืนยันหลักฐานที่นำมากล่าวอ้าง ทั้งนี้ขึ้นอยู่กับระบบ วิธีการที่นำมาใช้เพียงลักษณะอย่างใดอย่างหนึ่ง(Single-factor authentication) นั้นมีข้อจำกัดในการใช้ ตัวอย่างเช่น สิ่งที่คุณมี (Possession factor) นั้นอาจจะสูญหายหรือถูกขโมยได้ สิ่งที่คุณรู้ (Knowledge factor) อาจจะถูกดักฟัง เดา หรือขโมยจากเครื่องคอมพิวเตอร์ สิ่งที่คุณเป็น (Biometric factor) จัดได้ว่าเป็นวิธีที่มีความปลอดภัยสูงอย่างไรก็ตามการที่จะใช้เทคโนโลยีนี้ได้นั้นจำเป็นต้องมีการลงทุนที่สูง เป็นต้น

ดังนั้นจึงได้มีการนำแต่ละคุณลักษณะมาใช้ร่วมกัน (multi-factor authentication) ตัวอย่างเช่น ใช้สิ่งที่คุณมีกับสิ่งที่คุณรู้มาใช้ร่วมกัน เช่น การใช้ลายมือชื่อร่วมกับการใช้บัตรเครดิตหรือการใช้รหัสผ่านร่วมกับการใช้บัตร ATMเป็นต้น การนำแต่ละลักษณะของการพิสูจน์ตัวตนมาใช้ร่วมกันมากกว่า 1 ลักษณะจะช่วยเพิ่มประสิทธิภาพในการรักษาความปลอดภัยของข้อมูล

3.3 การกำหนดสิทธิ์ (Authorization)

การกำหนดสิทธิ์ คือขั้นตอนในการอนุญาตให้แต่ละบุคคลสามารถเข้าถึงข้อมูลหรือระบบใดได้บ้าง ก่อนอื่นต้องทราบก่อนว่าบุคคลที่กล่าวอ้างนั้นคือใครตามขั้นตอนการพิสูจน์ตัวตนและต้องให้แน่ใจด้วยว่าการพิสูจน์ตัวตนนั้นถูกต้อง

3.4 การเข้ารหัส (Encryption)

การเข้ารหัส คือการเก็บข้อมูลให้เป็นส่วนบุคคลจากบุคคลอื่นที่ไม่ได้รับอนุญาต ส่วนประกอบ 2 ส่วนที่สำคัญที่จะช่วยทำให้ข้อมูลนั้นเป็นความลับได้ก็คือ การกำหนดสิทธิ์และการพิสูจน์ตัวตนเพราะว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก่อนการอนุญาตให้บุคคลที่กล่าวอ้างเข้าถึงข้อมูลหรือถอดรหัสข้อมูลนั้นต้องสามารถแน่ใจได้ว่าบุคคลที่กล่าวอ้างนั้นเป็นใครและได้รับอนุญาตให้สามารถเข้ามาดูข้อมูลได้หรือไม่ ในการเข้ารหัสนั้นวิธีการหนึ่งที่ทำได้คือการเข้ารหัสในรูปแบบของกุญแจลับ (Secret key) ซึ่งในการใช้คีย์รูปแบบนี้ต้องเฉพาะผู้ที่มีกุญแจลับนี้เท่านั้นที่สามารถรับข้อมูลที่เข้ารหัสแล้วได้

3.5 การรักษาความสมบูรณ์ (Integrity)

การรักษาความสมบูรณ์ คือการรับรองว่าข้อมูลจะไม่ถูกเปลี่ยนแปลงหรือทำลายไปจากต้นฉบับ (source) ไม่ว่าจะเป็น โดยบังเอิญหรือดัดแปลงโดยเจตนาที่อาจส่ง ผลเสียต่อข้อมูลการคุกคามความสมบูรณ์ของข้อมูลคือการที่บุคคลที่ไม่ได้รับอนุญาตสามารถที่จะเข้าควบคุมการจัดการของข้อมูลได้

3.6 การตรวจสอบ (Audit)

การตรวจสอบ คือการตรวจสอบหลักฐานทางอิเล็กทรอนิกส์ ซึ่งสามารถใช้ในการติดตามการดำเนินการเพื่อตรวจสอบความถูกต้องและแม่นยำ ตัวอย่างเช่นการตรวจ สอบบัญชีชื่อผู้ใช้ โดยผู้ตรวจบัญชี ซึ่งการตรวจสอบความถูกต้องของการดำเนินการเพื่อให้แน่ใจว่าหลักฐานทางอิเล็กทรอนิกส์นั้นได้ถูกสร้างและสั่งให้ทำงาน โดย บุคคลที่ได้รับอนุญาต และในการเชื่อมต่อเหตุการณ์เข้ากับบุคคลจะต้องทำการตรวจสอบหลักฐานของบุคคลนั้นด้วย ซึ่งถือเป็นหลักการพื้นฐานของขั้นตอนการทำงานของการพิสูจน์ตัวตนด้วย

การพิสูจน์ตัวตนจัดเป็นการตรวจสอบหลักฐานขั้นพื้นฐานที่สำคัญที่สุดใน 5 ระดับชั้นของการควบคุมความปลอดภัย ดังนั้นการพิสูจน์ตัวตนดีจะช่วยเพิ่มความมั่นคง ปลอดภัยขั้นพื้นฐานให้กับระบบมากยิ่งขึ้น

3.7 ประเภทของการพิสูจน์ตัวตน (Authentication Types)

ส่วนประกอบพื้นฐานของการพิสูจน์ตัวตนสมบูรณ์แบ่งได้เป็น 3 ส่วน คือ

- การพิสูจน์ตัวตน (Authentication) คือส่วนที่สำคัญที่สุดเพราะเป็นขั้นตอนแรกของการเข้าใช้ระบบ ผู้เข้าใช้ระบบต้องถูกยอมรับจากระบบว่าสามารถเข้าสู่ระบบได้ การพิสูจน์ตัวตนเป็นการตรวจสอบหลักฐานเพื่อแสดงว่าเป็นบุคคลนั้นจริง
- การกำหนดสิทธิ์ (Authorization) คือข้อจำกัดของบุคคลที่เข้ามาในระบบ ว่าบุคคลคนนั้นสามารถทำอะไรกับระบบได้บ้าง
- การบันทึกการใช้งาน (Accountability) คือการบันทึกรายละเอียดของการใช้ระบบและรวมถึงข้อมูลต่างๆที่ผู้ใช้กระทำลงไปในระบบ เพื่อผู้ตรวจสอบจะได้ตรวจสอบได้ว่า ผู้ใช้ที่เข้ามาใช้บริการได้เปลี่ยนแปลงหรือแก้ไขข้อมูลในส่วนใดบ้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากที่ได้กล่าวไปข้างต้นว่าการพิสูจน์ตัวตนมีความสำคัญที่สุดกับการเข้าใช้ระบบ จึงแจกแจงชนิดของการพิสูจน์ตัวตนใช้กันอยู่ในปัจจุบันว่ามีอะไรบ้างและแต่ละชนิดมีลักษณะอย่างไร ดังนี้

3.8 ไม่มีการพิสูจน์ตัวตน (No Authentication)

ตามหลักการแล้วการพิสูจน์ตัวตนไม่มีความจำเป็น ถ้าเงื่อนไขต่อไปนี้เป็นจริง

- ข้อมูลเหล่านั้นเป็นข้อมูลสาธารณะ ที่อนุญาตให้ทุกคนเข้าใช้บริการและเปลี่ยนแปลงได้
- ข้อมูลข่าวสารหรือแหล่งของข้อมูลนั้นๆ สามารถเข้าถึงได้เฉพาะบุคคลที่ได้รับอนุญาตเท่านั้น

3.9 การพิสูจน์ตัวตนโดยใช้รหัสผ่าน (Authentication by Passwords)

รหัสผ่านเป็นวิธีการที่ใช้มานานและนิยมใช้กันแพร่หลาย รหัสผ่านควรจำกัดให้เฉพาะผู้ใช้ที่มีสิทธิเท่านั้นที่ทราบแต่ว่าในปัจจุบันนี้ การใช้แค่รหัสผ่านไม่มีประสิทธิภาพมากพอที่จะรักษาความมั่นคงปลอดภัยให้กับระบบคอมพิวเตอร์หรือระบบเครือข่ายคอมพิวเตอร์ เนื่องจากการตั้งรหัสผ่านที่ง่ายเกินไป และวิทยาการและความรู้ที่ก้าวหน้าทำให้รหัสผ่านอาจจะถูกขโมยโดยระหว่างการสื่อสารผ่านเครือข่ายได้

3.10 การพิสูจน์ตัวตนโดยใช้ Password Authenticators หรือ Tokens

Authenticator หรือ โทเคนเป็นฮาร์ดแวร์พิเศษที่ใช้สร้าง "รหัสผ่านซึ่งเปลี่ยนแปลงได้ (dynamic password)" ในขณะที่กำลังเข้าสู่ระบบเครือข่ายมี 2 วิธี คือ ซิงโครนัส และ อะซิงโครนัส

- การพิสูจน์ตัวตนแบบซิงโครนัส แบ่งออกเป็น 2 ประเภทตามลักษณะของการใช้งาน คือ

  การพิสูจน์ตัวตนแบบซิงโครนัส โดยขึ้นอยู่กับสถานการณ์ (Event-synchronous authentication) เมื่อผู้ใช้ต้องการที่จะเข้าสู่ระบบ ผู้ใช้จะต้องกดโทเคนเพื่อให้ โทเคนสร้างรหัสผ่านให้ จากนั้นผู้ใช้นำรหัสผ่านที่แสดงหลังจากกดโทเคนใส่ลงในฟอร์ม เพื่อเข้าสู่ระบบ ระบบจะทำการตรวจสอบกับเซิร์ฟเวอร์ก่อน ว่ารหัสผ่านที่ใส่มีอยู่ในเซิร์ฟเวอร์จริง จึงจะยินยอมให้ผู้ใช้เข้าสู่ระบบ

  การพิสูจน์ตัวตนแบบซิงโครนัส โดยขึ้นอยู่กับเวลา (Time-synchronous authentication) เป็นวิธีการที่สร้างรหัสผ่าน โดยมีการกำหนดช่วงระยะเวลาการใช้งาน โดยปกติแล้วรหัสผ่านจะถูกเปลี่ยนทุกๆ หนึ่งนาที การสร้างรหัสผ่านจะเป็นไปอย่างต่อเนื่อง ทำให้บางครั้งรหัสผ่านที่สร้างออกมาอาจจะซ้ำกันกับรหัสผ่านตัวอื่นที่เคยสร้างมาแล้วก็ได้เมื่อผู้ใช้ ต้องการเข้าสู่ระบบก็ใส่รหัสผ่านและเวลาที่รหัสผ่านตัวนั้นถูกสร้างขึ้นมา (รหัสผ่านจะถูกสร้างขึ้นมาจากโทเคน)ลงในฟอร์มเพื่อเข้าสู่ระบบ ระบบจะทำการตรวจสอบเวลาและรหัสผ่านที่ผู้ใช้ใส่ลงไป กับเซิร์ฟเวอร์ว่ารหัสผ่านที่ใส่ตรงกับเวลาที่โทเคนสร้าง และมีอยู่ในเซิร์ฟเวอร์จริง จึงยินยอมให้ผู้ใช้เข้าสู่ระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การพิสูจน์ตัวตนแบบอะซิงโครนัส หรือเรียกอีกอย่างหนึ่งว่า "challenge-response" ถูกพัฒนาขึ้นเป็นลำดับแรกๆ ของระบบการใช้ "รหัสผ่านซึ่งเปลี่ยนแปลงได้" ซึ่งถือได้ว่าเป็นการป้องกันการจู่โจมที่ปลอดภัยที่สุด เพราะเนื่องจากว่าเมื่อผู้ใช้ต้องการจะเข้าสู่ระบบ ผู้ใช้จะต้องทำการร้องของไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์ก็จะส่ง challenge string มาให้ผู้ใช้ เพื่อให้ผู้ใช้ใส่ลงในโทเคนที่ผู้ใช้ถืออยู่ จากนั้นโทเคนจะทำการคำนวณรหัสผ่านออกมาให้ผู้ใช้ ผู้ใช้จึงสามารถนำรหัสผ่านนั้นใส่ลงในฟอร์มเพื่อเข้าสู่ระบบได้

การพิสูจน์ตัวตนแบบซิงโครนัสทั้งไคลเอ็นต์และเซิร์ฟเวอร์จะมีรหัสผ่านเก็บเอาไว้ แต่แบบอะซิงโครนัส ไคลเอ็นต์จะต้องติดต่อเซิร์ฟเวอร์ก่อน ก่อนจะได้รับรหัสผ่านจริง ทำให้การพิสูจน์ตัวตนแบบอะซิงโครนัสมีขั้นตอนที่ซับซ้อนกว่าแบบซิงโคนัส

ตัวอย่างของฮาร์ดแวร์พิเศษที่ใช้ในการสร้างรหัสผ่านซึ่งเปลี่ยนแปลงได้ ของการพิสูจน์ตัวตนโดยใช้ password authenticator หรือโทเคน

3.11 การพิสูจน์ตัวตนโดยการเข้ารหัสโดยใช้กุญแจสาธารณะ (Public-key cryptography)

เป็นการรักษาความปลอดภัยของข้อมูลระหว่างการส่งข้ามเครือข่ายวิธีหนึ่งที่นิยมใช้กันอยู่ในปัจจุบัน การเข้ารหัสแบบคู่รหัสกุญแจนี้จะมีความปลอดภัยมากกว่าการเข้ารหัสข้อมูลแบบธรรมดา แต่ก็ไม่ได้หมายความว่าการเข้ารหัสแบบคู่รหัสกุญแจนี้จะเป็นวิธีที่เหมาะสมที่สุดของวิธีการเข้ารหัส ทั้งนี้ขึ้นอยู่กับประเภทงานของแต่ละองค์กรหรือบุคคล

การเข้ารหัสโดยใช้กุญแจสาธารณะ ประกอบไปด้วยกุญแจ 2 ชนิด ที่ต้องใช้คู่กันเสมอในการเข้ารหัสและถอดรหัสคือ

- กุญแจสาธารณะ (public key) เป็นกุญแจที่ผู้สร้างจะส่งออกไปให้ผู้ใช้อื่นๆ ทราบหรือเปิดเผยได้
- กุญแจส่วนตัว (private key) เป็นกุญแจที่ผู้สร้างจะเก็บไว้ โดยไม่เปิดเผยให้คนอื่นรู้

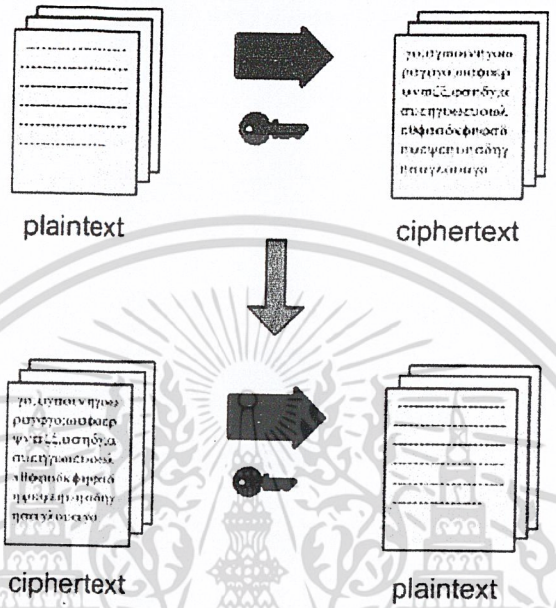
กระบวนการของการเข้ารหัสแบบคู่รหัสกุญแจ มีดังนี้

1. ผู้ใช้แต่ละคนจะสร้างคู่รหัสกุญแจของตัวเองขึ้นมา เพื่อใช้สำหรับการเข้ารหัสและการถอดรหัส
2. กุญแจสาธารณะจะถูกส่งออกไปยังผู้ใช้คนอื่นๆ แต่กุญแจส่วนตัวจะถูกเก็บที่ตนเอง
3. เมื่อจะส่งข้อมูลออกไปหาผู้ใช้คนใด ข้อมูลที่ส่งจะถูกเข้ารหัสด้วยกุญแจสาธารณะ ก่อนถูกส่งออกไป
4. เมื่อผู้รับได้รับข้อความแล้วจะใช้กุญแจส่วนตัวซึ่งเป็นคู่รหัสกันถอดรหัสออกมา

การเข้ารหัสโดยใช้กุญแจสาธารณะสามารถใช้ได้ทั้งในการเข้ารหัส (Encryption) และการพิสูจน์ตัวตน (Authentication)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประยุกต์ใช้ในการเข้ารหัสข้อมูล (Encryption) เป็นการนำข้อมูลที่จะส่งไปยังผู้รับมาเข้ารหัสด้วยกุญแจสาธารณะของผู้รับ และเมื่อผู้รับได้รับข้อความนั้นแล้วจะถอดรหัสออกมาด้วยกุญแจส่วนตัว จึงจะเห็นได้ว่ามีเพียงผู้รับเท่านั้นที่จะสามารถถอดรหัสออกมาได้

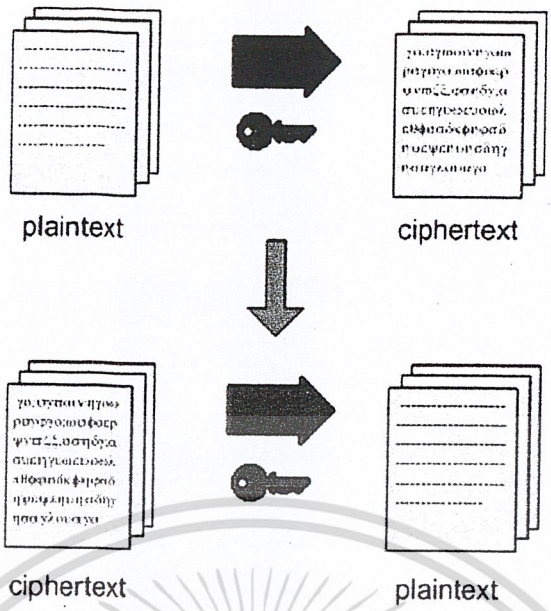


รูปที่ 3.3ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจ

การประยุกต์ใช้ในการพิสูจน์ตัวตน (Authentication) เป็นการนำข้อมูลที่ผู้ส่งต้องการส่งมาเข้ารหัสด้วยกุญแจส่วนตัวของผู้ส่ง แล้วนำข้อมูลนั้นส่งไปยังผู้รับ ซึ่งผู้รับจะใช้กุญแจสาธารณะซึ่งเป็นคู่รหัสกันถอดรหัสออกมา ผู้รับก็สามารถรู้ได้ว่าข้อความนั้นถูกส่งมาจากผู้ส่งคนนั้นจริง ถ้าสามารถถอดรหัสข้อมูลได้อย่างถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

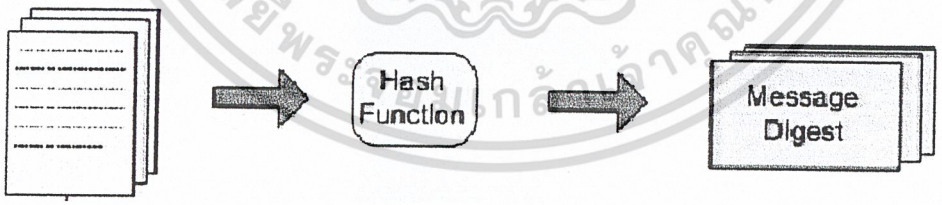


รูปที่ 3.4 ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจเพื่อการพิสูจน์ตัวตน

3.12 การพิสูจน์ตัวตนโดยการใช้ลายเซ็นอิเล็กทรอนิคส์ (Digital Signature)

เป็นการนำหลักการของการทำงานของระบบการเข้ารหัสแบบใช้คู่รหัสกุญแจเพื่อการพิสูจน์ตัวตนมาประยุกต์ใช้ ระบบของลายเซ็นดิจิตอลสามารถแบ่งเป็นขั้นตอนได้ดังนี้

1. เมื่อผู้ใช้ต้องการจะส่งข้อมูลไปยังผู้รับ ข้อมูลนั้นจะถูกนำไปเข้าฟังก์ชั่นทางคณิตศาสตร์ที่เรียกว่า "แฮชฟังก์ชัน" ได้เมสเซสไดเจสต์ (Message Digest) ออกมา



รูปที่ 3.5 การส่งข้อมูลเข้าไปใน Hash function

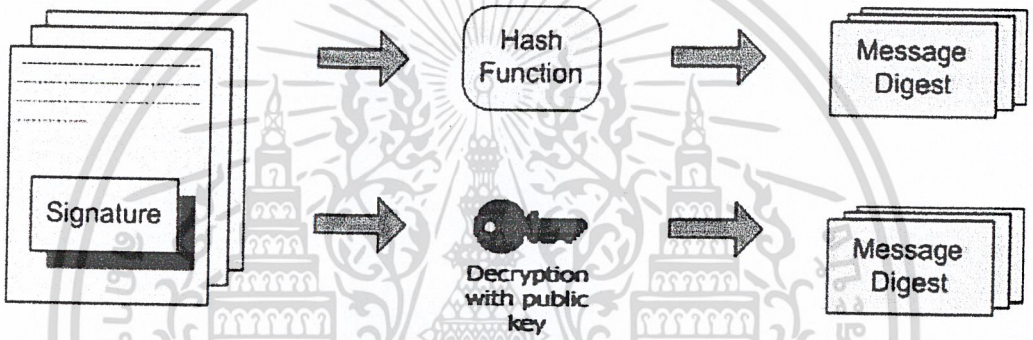
2. การใช้กุญแจส่วนตัวเข้ารหัสข้อมูล หมายถึงว่าผู้ส่งได้ลงลายเซ็นดิจิตอล ยินยอมที่จะให้ผู้รับสามารถทำการตรวจสอบด้วยกุญแจสาธารณะของผู้ส่งเพื่อพิสูจน์ตัวตนของผู้ส่งได้

61969
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.6 การเข้ารหัสเมสเซสไดเจสต์ด้วยกุญแจส่วนตัวเพื่อเป็นการลงลายเซ็น

3. การตรวจสอบข้อมูลว่าถูกส่งมาจากผู้ส่งคนนั้นจริงในด้านผู้รับ โดยการนำข้อมูลมาผ่านแฮชฟังก์ชันเพื่อคำนวณหาค่าเมสเซจไดเจสต์ และถอดรหัสลายเซ็นอิเล็กทรอนิคส์ด้วยกุญแจสาธารณะของผู้ส่ง ถ้าสามารถถอดได้อย่างถูกต้อง จะเป็นการยืนยันข้อมูลจากผู้ส่งคนนั้นจริง และถ้าข้อมูลเมสเซจไดเจสต์ที่ได้จากการถอดรหัสเท่ากันกับค่าเมสเซจไดเจสต์ในตอนต้นที่ทำการคำนวณได้ จะถือว่าข้อมูลดังกล่าวนั้นถูกต้อง



รูปที่ 3.7 ขั้นตอนการเปรียบเทียบความถูกต้อง

ลายเซ็นอิเล็กทรอนิคส์นิยมนำไปใช้ในระบบรักษาความปลอดภัยในการชำระเงินผ่านระบบอินเตอร์เนต ซึ่งในปัจจุบันนี้การทำธุรกรรมการเงินอิเล็กทรอนิกส์ได้รับความนิยมเป็นอย่างมาก

การพิสูจน์ตัวตน	ข้อดี	ข้อเสีย
ไม่มีการพิสูจน์ตัวตนตน	ง่ายต่อการใช้งานและค่าใช้จ่ายต่ำ	ความปลอดภัยของข้อมูลจะขึ้นอยู่กับผู้ใช้ว่าจะนำข้อมูลเหล่านั้นไปใช้ในทางที่ควรหรือไม่
การพิสูจน์ตัวตนโดยใช้รหัสผ่าน	สามารถใช้ได้กับทุกระบบ	จะไม่ปลอดภัยเมื่อมีการส่งข้ามระบบเครือข่ายที่เป็นสาธารณะหรือไม่มีการเข้ารหัสข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การพิสูจน์ตัวตนโดยใช้ PIN	- ง่ายต่อการจำและความปลอดภัยค่อนข้างดี (บัตร ATM) -สามารถสื่อสารข้ามเครือข่ายสาธารณะได้อย่างปลอดภัย	- ต้องใช้ฮาร์ดแวร์เฉพาะในการอ่าน PIN - ไม่สามารถใช้กับต่างระบบกันได้ - ราคาแพง
การพิสูจน์ตัวตนโดยใช้ password authenticators หรือโทเคนแบบซิงโครนัส	- มีความปลอดภัยมากกว่าการใช้การจำรหัสผ่านแบบธรรมดา - ไม่ต้องใช้เครื่องอ่านการ์ด - ผู้ที่ละเมิดเข้ามาไม่สามารถจะเข้ามาดูโจมได้	- การใช้งานยุ่งยากกว่าแบบจำรหัสผ่าน - authenticator เป็นวัตถุจึงง่ายต่อการสูญหายและการถูกขโมยได้
การพิสูจน์ตัวตนตนโดยใช้ password authenticators หรือโทเคนแบบอะซิงโครนัส	- มีความปลอดภัยมากกว่าการใช้การจำรหัสผ่านแบบธรรมดา - ไม่ต้องใช้เครื่องอ่านการ์ด - เป็นวิธีการป้องกันที่ดีที่สุดเมื่อเปรียบเทียบกับ วิธีการใช้การพิสูจน์ตัวตนโดยใช้ password authenticators หรือโทเคน	- การใช้งานยุ่งยากกว่าแบบจำรหัสผ่าน - authenticator เป็นวัตถุจึงง่ายต่อการสูญหาย และการถูกขโมยได้ไม่สามารถป้องกันผู้ที่ละเมิดเข้ามาในระบบได้ - การใช้งานค่อนข้างยุ่งยากกว่าวิธีการใช้ "รหัสผ่านซึ่งเปลี่ยนแปลงได้ (dynamic password)" วิธีอื่นๆ
การพิสูจน์ตัวตนโดยใช้	-มีความปลอดภัยสูงเพราะ	- ระบบมีความซับซ้อนสูง - ยังไม่ได้รับความนิยมกันอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลักษณะเฉพาะทางชีวภาพของแต่ละบุคคล	เลียนแบบกันได้ยาก	แพร่หลาย - ค่าใช้จ่ายสูง
การพิสูจน์ตัวตนโดยวิธี One-Time Password	-ทำให้การเดาหรือขโมยรหัสผ่านเป็นไปได้ยาก	- ไม่สะดวกต่อการใช้งาน เพราะผู้ใช้ต้องจำรหัสผ่านหลายตัว - ถ้าผู้ใช้จำรหัสผ่านไม่ได้ หรือ ทำรหัสผ่านสูญหาย ก็ไม่สามารถเข้าใช้ระบบได้
การพิสูจน์ตัวตนโดยการเข้ารหัสแบบคู่รหัสกุญแจ	- การจัดการกุญแจทำได้ปลอดภัย เพราะใช้กุญแจในการเข้ารหัส และถอดรหัสต่างกัน - สามารถระบุผู้ใช้โดยการใช้ร่วมกับลายมือชื่ออิเล็กทรอนิกส์	- ใช้เวลาในการเข้าและถอดรหัสค่อนข้างนาน เพราะต้องใช้การคำนวณอย่างมาก - ต้องใช้ระบบที่สนับสนุนการทำงาน
การพิสูจน์ตัวตนโดยการใช้ลายเซ็นดิจิตอล	- สามารถระบุตัวผู้ส่งได้ชัดเจน - ป้องกันข้อมูลถูกแก้ไขระหว่างการส่งได้ หรือสามารถตรวจสอบข้อมูลได้ว่าผ่านการแก้ไขมาหรือไม่	-ใช้เวลาในการเข้าและถอดรหัสค่อนข้างนาน เพราะต้องใช้การคำนวณอย่างมาก
การพิสูจน์ตัวตนโดยวิธี zero-knowledge	-ความปลอดภัยค่อนข้างสูง เพราะคำถามและคำตอบจะมีเพียง	-ความซับซ้อนของระบบเพิ่มขึ้นตามความฉลาดของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

proofs	ผู้ใช้ และ เซิร์ฟเวอร์เท่านั้น ที่ทราบ	

ตารางที่ 3.1 เปรียบเทียบข้อดีข้อเสียของการพิสูจน์ตัวตนแต่ละชนิด

3.13 โพรโตคอลในการพิสูจน์ตัวตน(Authentication Protocol)

ในระบบเครือข่ายแบบเปิดหรืออินเตอร์เน็ต การพิสูจน์ตัวตนถือได้ว่าเป็นกระบวนการเริ่มต้นและมีความสำคัญที่สุดในการปกป้องเครือข่ายให้ปลอดภัยโพรโตคอลในการพิสูจน์ตัวตน คือโพรโตคอลการสื่อสารที่มีกระบวนการพิสูจน์ตัวตนรวมอยู่ในชุดโพรโตคอล

3.14 Secure Socket Layer (SSL)

Secure Sockets Layer (SSL) เริ่มพัฒนาโดย Netscape Communications เพื่อใช้ในโพรโตคอลระดับแอพพลิเคชันคือ Hypertext Transfer Protocol (HTTP) ซึ่งเป็นการสื่อสารผ่านเว็บให้ปลอดภัย พัฒนาในช่วงต้นของยุคการค้าอิเล็กทรอนิคส์กำลังได้รับความนิยมในโลกอินเตอร์เน็ต

SSL ทำให้เกิดการสื่อสารอย่างปลอดภัยระหว่างไคลเอ็นต์และเซิร์ฟเวอร์ โดยการอนุญาตให้มีกระบวนการพิสุจน์ตัวตนร่วมกับการใช้งานลายเซ็นดิจิตอลสำหรับการรักษาความถูกต้องของข้อมูลและการเข้ารหัสข้อมูลเพื่อป้องกันความเป็นส่วนตัวระหว่างการสื่อสารข้อมูลโพรโตคอล SSL อนุญาตให้สามารถเลือกวิธีการในการเข้ารหัส วิธีสร้างไดเจสต์ [*1] และลายเซ็นดิจิตอล ได้อย่างอิสระก่อนการสื่อสารจะเริ่มต้นขึ้น ตามความต้องการของทั้งเว็บเซิร์ฟเวอร์และบราวเซอร์ ทั้งนี้เพื่อเพิ่มความยืดหยุ่นในการใช้งาน เปิดโอกาสให้ทดลองใช้วิธีการในการเข้ารหัสวิธีใหม่ รวมถึงลดปัญหาการส่งออกวิธีการเข้ารหัสไปประเทศที่ไม่อนุญาต

Netscape เริ่มพัฒนา SSL เวอร์ชันแรกคือเวอร์ชัน 2.0 และเวอร์ชันถัดมาเป็น 3.0 ซึ่งสนับสนุนความสามารถด้านความปลอดภัยมากขึ้น และเป็นเวอร์ชันสุดท้ายก่อนที่จะเป็นมาตรฐานกลางของโพรโตคอลบนอินเตอร์เน็ต โดยเปลี่ยนชื่อเป็น Transport Layer Security หรือ TLS ซึ่งดูแลมาตรฐานโดย Internet Engineering Task Force (IETF) อธิบายเวอร์ชันของ SSL และผู้พัฒนาได้ตามตาราง

เวอร์ชัน	ผู้พัฒนา	จุดเด่น	บราวเซอร์ที่สนับสนุน
SSL v2.0	Netscape Corp. [SSL2]	โพรโตคอล SSL รุ่นแรกที่พัฒนา	• NS Navigator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

		บนบราวเซอร์	1.x/2.x • MS IE 3.x • Lynx/2.8 + OpenSSL
SSL v3.0	Netscape Corp. เป็น Internet Drafted รุ่นก่อนเป็นมาตรฐานกลาง [SSL3]	ปรับปรุงใหม่เพิ่มความปลอดภัยมากขึ้น สนับสนุนการใช้ non-RSA ciphers ในการเข้ารหัสและห่วงโซ่เอกสารสิทธิ[*2]	• NS Navigator 2.x/3.x/4.x • MS IE 3.x/4.x • Lynx/2.8 + OpenSSL
TLS v1.0	IETF กำลังเสนอให้เป็นมาตรฐานโพรโตคอลบนอินเตอร์เน็ต (Proposed Internet Standard)	ปรับปรุงจาก SSL v3.0 สนับสนุนการทำงานในชั้น MAC และ HMAC เพิ่ม Padding ชนิด Block และวิธีการจัดลำดับข้อมูลและเพิ่มระดับการแจ้งเตือน	• Lynx/2.8 + OpenSSL

ตารางที่ 3.2 อธิบายเวอร์ชันของ SSL และผู้พัฒนา

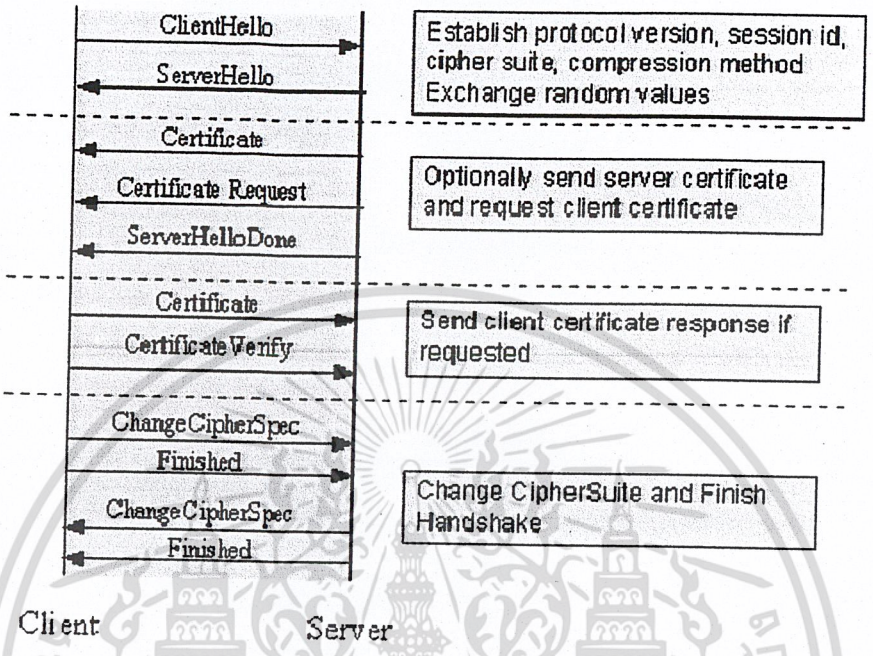
- [*1] ไดเจสต์ (Digest) คือข้อความที่เกิดจากการเข้ารหัสข้อมูลด้วยฟังก์ชันแฮชเช่น MD5 หรือ SHA-1
- [*2] ห่วงโซ่เอกสารสิทธิ (Certificate Chain) คือการเพิ่มข้อมูลเอกสารสิทธิที่เกี่ยวเนื่องกันเมื่อใช้ในขั้นตอนแลกเปลี่ยนข้อมูล ซึ่งจะช่วยลดเวลาในการค้นหาเอกสารสิทธิจากผู้ให้บริการ Certificate Authority (CA) ที่เกี่ยวเนื่องกันมากกว่า 1 ขึ้นไป

กระบวนการในการเริ่มต้นการสื่อสารผ่านชั้น SSL แบ่งเป็น 4 ขั้นตอนคือ

1. ประกาศชุดวิธีการเข้ารหัส ไดเจสต์ และลายเซ็นดิจิตอลที่สนับสนุนของทั้งไคลเอ็นต์และเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การพิสูจน์ตัวตนของเซิร์ฟเวอร์ต่อไคลเอ็นต์
3. การพิสูจน์ตัวตนของไคลเอ็นต์ต่อเซิร์ฟเวอร์ ถ้าจำเป็น
4. ไคลเอ็นต์และเซิร์ฟเวอร์ตกลงชุดวิธีการเข้ารหัส การสร้างไดเจสต์ และการใช้ลายเซ็นดิจิตอล



รูปที่ 3.8 กระบวนการเริ่มต้นการติดต่อสื่อสารของโพรโตคอล SSL

ขั้นตอน 1 : ประกาศชุดวิธีการเข้ารหัส ไดเจสต์ และลายเซ็นดิจิตอลที่สนับสนุนของทั้งไคลเอ็นต์และเซิร์ฟเวอร์

ไคลเอ็นต์และเซิร์ฟเวอร์ส่งข้อความเริ่มต้นการสื่อสาร (Hello message) ซึ่งประกอบไปด้วยเวอร์ชันของโพรโตคอลที่ใช้ วิธีการเข้ารหัสที่เว็บเซิร์ฟเวอร์และไคลเอ็นต์สนับสนุน หมายเลขระบุการสื่อสาร (Session identifier) รวมถึงวิธีการบีบอัดข้อมูลในการสื่อสารที่สนับสนุนหมายเลขระบุการสื่อสารที่เกิดขึ้นใช้สำหรับตรวจสอบการเชื่อมต่อระหว่างไคลเอ็นต์และเซิร์ฟเวอร์ ถ้ามีการเชื่อมต่อก่อนหน้านี้เกิดขึ้น แสดงว่าได้มีการตกลงวิธีการสื่อสารแล้วสามารถเริ่มต้นส่งข้อมูลได้ทันที เป็นการลดเวลาติดต่อสื่อสารลง

ขั้นตอน 2 : การพิสูจน์ตัวตนของเซิร์ฟเวอร์ต่อไคลเอ็นต์

ถัดมาเว็บเซิร์ฟเวอร์ทำการส่งเอกสารสิทธิ หรือใบยืนยันความมีตัวตนของเซิร์ฟเวอร์ ไคลเอ็นต์จะทำการตรวจสอบเอกสารสิทธิกับผู้ให้บริการองค์กรที่ออกเอกสารสิทธิ ที่ได้ตั้งค่าไว้ เพื่อยืนยันความถูกต้องของเอกสารสิทธิของเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอน 3 : การพิสูจน์ตัวตนของไคลเอ็นต์ต่อเซิร์ฟเวอร์ ถ้าจำเป็น

เซิร์ฟเวอร์สามารถร้องขอเอกสารสิทธิจากไคลเอ็นต์เพื่อตรวจสอบความถูกต้องของไคลเอ็นต์ด้วยก็ได้ ใช้ในกรณีที่มีการจำกัดการใช้งานเฉพาะไคลเอ็นต์ที่ต้องการเท่านั้นซึ่ง SSL สนับสนุนการตรวจสอบได้จากทั้งเซิร์ฟเวอร์และไคลเอ็นต์ ขึ้นอยู่กับการเลือกใช้งานในขณะติดต่อสื่อสารที่เกิดขึ้นนั้น

ขั้นตอน 4 : ไคลเอ็นต์และเซิร์ฟเวอร์ตกลงชุดวิธีการเข้ารหัส การสร้างไดเจสต์ และการใช้ลายเซ็นดิจิตอล

ขั้นตอนการตรวจสอบเอกสารสิทธิที่เซิร์ฟเวอร์ร้องขอจากไคลเอ็นต์จะมีหรือไม่มีก็ได้ ขึ้นอยู่กับการตั้งค่าบนเซิร์ฟเวอร์ หลังจากขั้นตอนการตรวจสอบเสร็จสิ้นเซิร์ฟเวอร์ และไคลเอ็นต์จะตกลงการใช้งานวิธีการเข้ารหัสระหว่างกัน โดยใช้ค่าที่ได้จากการประกาศในขั้นตอนแรก

วิธีการแลกเปลี่ยนกุญแจในการเข้ารหัส (Key exchange method) คือการกำหนดกลไกการแลกเปลี่ยนกุญแจที่ใช้ในการเข้ารหัสระหว่างการสื่อสาร โดยทั้งไคลเอ็นต์และเซิร์ฟเวอร์จะใช้กุญแจนี้ในการเข้ารหัสและถอดรหัสข้อมูล ใน SSL เวอร์ชัน 2.0 จะสนับสนุนวิธีการแลกเปลี่ยนกุญแจแบบ RSA ส่วน SSL เวอร์ชัน3.0 ขึ้นไปจะสนับสนุนวิธีการอื่นๆ เพิ่มเติมเช่นการใช้ RSA ร่วมกับการใช้เอกสารสิทธิ หรือ Diffie-Hellman เป็นต้น

วิธีการเข้ารหัสในปัจจุบันแบ่งเป็นสองวิธีคือ การใช้กุญแจเดียวกันในการเข้ารหัสและถอดรหัส อาจเรียกกุญแจนี้ว่า Session key หรือ กุญแจลับส่วนอีกวิธีการคือ การใช้กุญแจคนละตัวในการเข้ารหัสและถอดรหัส ประกอบไปด้วยกุญแจสาธารณะและกุญแจส่วนตัวซึ่งเป็นคู่กันเสมอ เข้ารหัสด้วยกุญแจใดจะต้องถอดรหัสด้วยกุญแจที่คู่กันและตรงกันข้ามเท่านั้น มักใช้วิธีการเข้ารหัสด้วยกุญแจคนละตัวมาใช้ในการเข้ารหัส Session key และส่งไปให้ฝั่งตรงข้ามก่อนการสื่อสารจะเกิดขึ้นรวมเรียกว่าวิธีการแลกเปลี่ยนกุญแจในการเข้ารหัส SSL ใช้วิธีการเข้ารหัสด้วยกุญแจสมมาตร หรือกุญแจเดียวในการเข้ารหัส และถอดรหัส ตามที่กล่าวข้างต้น วิธีการเข้ารหัสคือ การเข้ารหัสด้วย DES และ 3DES(Data Encryption Standard), วิธีการเข้ารหัสด้วย IDEA ส่วน RC2 และ RC4 เป็นวิธีการเข้ารหัสของ RSA รวมถึงวิธีการเข้ารหัสแบบ Fortezza สำหรับความยาวของการเข้ารหัสที่ใช้คือ 40 บิต, 96 บิต และ 128 บิต

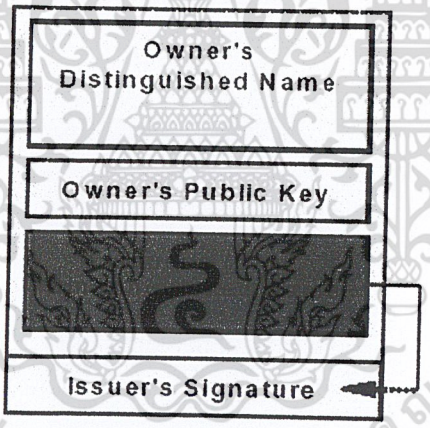
การสร้าง Message Authentication Code (MAC) เพื่อใช้สำหรับการยืนยันความถูกต้องของข้อมูลระหว่างการสื่อสารและป้องกันการปลอมข้อมูล ส่วนฟังก์ชันสร้างไดเจสต์ที่ SSL สนับสนุนและเลือกใช้ได้ในปัจจุบันคือ MD5 ขนาด 128 บิต และ SHA-1 (Secure Hash Algorithm) ขนาด 160 บิต ซึ่งจะได้วิธีการที่ทั้งสองฝ่ายสนับสนุนและเหมาะสม ซึ่งเป็นขั้นตอนสุดท้ายก่อนการสื่อสารที่มีการเข้ารหัสจะเริ่มต้นขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.15 เอกสารสิทธิ์ (Digital Certificates)

ลักษณะของเอกสารสิทธิ์ดิจิตอล เอกสารสิทธิ์เป็นเสมือนกับบัตรประจำตัวประชาชน ซึ่งจะบ่งบอกรายละเอียดของบุคคล เพื่อใช่ในการยืนยันตัวตน เราสามารถส่งเอกสารสิทธิ์ไปควบคู่กับลายมือชื่อดิจิตอลเพื่อใช่ในการยืนยันว่าเอกสารสิทธิ์นั้น ไม่มีการเปลี่ยนแปลง การสร้างเอกสารสิทธิ์ทำโดยผู้ใช้ทุกคนทำการขอเอกสารสิทธิ์กับองค์กรพิสูจน์สิทธิ์(Certificate Authority : CA) โดยการส่งคีย์สาธารณะและข้อมูลตามที่องค์กรพิสูจน์สิทธิ์นั้นกำหนดไป และทำการขอเอกสารสิทธิ์มา การติดต่อต้องทำเป็นการส่วนตัวหรือติดต่อผ่านระบบการพิสูจน์ตัวตนที่ปลอดภัย เราสามารถกำหนดสิ่งที่จำเป็นในการสื่อสารได้ดังต่อไปนี้

1. ผู้ใช้ทุกคนสามารถคำนวณหาชื่อและคีย์สาธารณะของเจ้าของเอกสารสิทธิ์ได้
2. ผู้ใช้ทุกคนสามารถตรวจสอบได้ว่าเอกสารสิทธิ์มาจากองค์กรพิสูจน์สิทธิ์จริงๆไม่ได้ถูกปลอมแปลงมา
3. ผู้ใช้สามารถตรวจสอบได้ว่าเอกสารสิทธิ์นั้นหมดอายุหรือไม่
4. ผู้ที่สามารถอ้าง และอัพเดต(update)เอกสารสิทธิ์ได้มีเพียงองค์กรมีอำนาจในการรับรองสิทธิ์เท่านั้น
5. ผู้ใช้ทุกคนสามารถตรวจสอบเอกสารสิทธิ์ได้เป็นประจำ



รูปที่ 3.9 แสดงส่วนประกอบต่างๆของเอกสารสิทธิ์

ความสำคัญของเอกสารสิทธิ์ในการเข้ารหัสแบบคีย์ต่างๆนั้นเราจะต้องมีการสร้างทั้งคีย์ส่วนตัวและคีย์สาธารณะ ซึ่งโดยทั่วไปการสร้างคีย์จะทำโดยโปรแกรมที่จะใช้คีย์นั้น เช่น โปรแกรมเว็บบราวเซอร์หรือโปรแกรมติดต่ออิเล็กทรอนิกส์เมล์ หลังจากที่สร้างคีย์ทั้งสองเรียบร้อยแล้วการเก็บรักษาคีย์เป็นเรื่องสำคัญ เราจะต้องเก็บรักษาคีย์ส่วนตัวไว้ให้ดีอย่าให้ใครแอบมาเห็นหรือขโมยไปได้ จากนั้นจะเป็นการตัดสินใจว่าจะทำการแจกจ่ายคีย์สาธารณะของเราไปสู่ผู้อื่นด้วยวิธีใด เช่นการแจกคีย์โดยส่งอิเล็กทรอนิกส์เมล์ไปให้เพื่อน หรือบุคคลที่ติดต่อกับเรา แต่วิธีนี้เราอาจส่งคีย์ไปให้ไม่ได้ครบทุกคน และยังต้องเป็นภาระการจัดส่งคีย์ให้กับบุคคลใหม่ๆที่ต้องการติดต่อด้วย นอกจากนี้ยังไม่สามารถทำให้ผู้รับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มั่นใจได้ว่าคีย์ที่ส่งไปนั้นเป็นคีย์ของเราจริงๆ เนื่องจากอาจมีผู้อื่นแอบสร้างคีย์โดยใช้ชื่อเราและแอบอ้างส่งคีย์ดังกล่าวให้กับผู้อื่นเพื่อให้เข้าใจว่าเป็นคีย์ของเราได้

วิธีที่ดีกว่าและใช้อยู่ในปัจจุบันก็คือการใช้ระบบแจกจ่ายคีย์ที่เชื่อถือได้โดยจะมีองค์กรที่ทำหน้าที่เฉพาะเป็นองค์กรที่สาม(Third Party) ในการรับรองและระงับการรับรองคีย์ที่เรียกว่าองค์กรพิสูจน์สิทธิ์ (Certificate Authority - CA) โดยองค์กรพิสูจน์สิทธิ์นี้จะตรวจสอบคีย์สาธารณะของเราด้วยหลักฐานว่าคีย์ของเรานั้นๆเป็นของจริง พร้อมทั้งตรวจสอบข้อมูลส่วนตัวของเรา(ข้อมูลที่ตรวจสอบจะมากน้อยแค่ไหนขึ้นอยู่กับระดับชั้นของการรับรอง) เมื่อผู้อื่นได้รับคีย์ของเราก็สามารถที่จะตรวจสอบกับผู้ออกเอกสารสิทธิ์นี้ว่าคีย์ที่ได้รับเป็นของจริงหรือไม่ ซึ่งตัวเอกสารสิทธิ์นี้จะเปรียบเสมือนบัตรประชาชนดิจิตอลของเราที่ใช้บอกได้ว่าเราเป็นบุคคลที่อ้างจริงๆในระบบเครือข่ายหรือการส่งข้อมูลทางอิเล็กทรอนิกส์

ในปัจจุบันบริษัทที่ออกเอกสารสิทธิ์ดิจิตอล (Digital Certificate) คือ บริษัท Verisign ,Cyvertrust,Global Sign และ Nortel โดยในเอกสารสิทธิ์ดิจิตอลจะประกอบด้วยข้อมูลต่างๆ ได้แก่ ชื่อของผู้ถือเอกสารสิทธิ์ ชื่อของบริษัทผู้ออกเอกสารสิทธิ์ คีย์สาธารณะของผู้ถือเอกสารสิทธิ์ วันหมดอายุของผู้ถือเอกสารสิทธิ์(โดยทั่วไปจะกำหนดระยะเวลา 6 เดือนหรือ 1 ปี) ระดับชั้นของเอกสารสิทธิ์ และเลขหมายของตัวเอกสารสิทธิ์ดิจิตอล

เอกสารสิทธิ์ดิจิตอลแบ่งออกได้เป็นสี่ระดับชั้นตามระดับการตรวจสอบข้อมูลของเจ้าของเอกสารสิทธิ์

1. ระดับที่หนึ่ง เป็นชั้นที่ออกเอกสารสิทธิ์ได้ง่ายที่สุดเนื่องจากมีการตรวจสอบน้อยที่สุด โดยจะตรวจแค่ชื่อผู้ถือเอกสารสิทธิ์ และที่อยู่อิเล็กทรอนิกส์เมล์(e-mail address) ว่าถูกต้องจริงเท่านั้น
2. ระดับชั้นที่สอง จะตรวจสอบเลขประจำตัวประชาชน เลขประจำตัวของงระบบสวัสดิการหรือประกันสังคม(social security number) และวันเดือนปีเกิด
3. ระดับชั้นที่สาม จะมีการตรวจสอบเพิ่มเติมเกี่ยวกับประวัติการใช้เครดิต และการชำระเงิน
4. ระดับชั้นที่สี่ ยังไม่มีการออกมาเป็นมาตรฐานอย่างแน่ชัด แต่จะเป็นการตรวจสอบข้อมูลเพิ่มเติมเกี่ยวกับตำแหน่งงานในองค์กรด้วย

ในการขอเอกสารสิทธิ์นั้นจะต้องกระทำบนกระบวนการที่ปลอดภัย ซึ่งในปัจจุบันมี 2 วิธีคือ

1. การขอเอกสารสิทธิ์ผ่าน โปรแกรมประเภทบราวเซอร์ เช่น อินเตอร์เน็ตเอ็กซ์พลอเรอร์, เนสเคปท์ เป็นต้น โดยผู้ใช้เข้าไปยังโฮสต์ที่ให้บริการการขอเอกสารสิทธิ์หรือเว็บไซต์ที่เปิดให้บริการดังกล่าวอยู่ ตัวโปรแกรมประเภทบราวเซอร์จะทำการสร้างคู่คีย์ คีย์ส่วนตัวจะเก็บไว้ที่เครื่องของเรา จากนั้นจะส่งคีย์สาธารณะและข้อมูลส่วนตัวของเราไปยังโฮสต์หรือไซต์ที่เป็นขององค์กรพิสูจน์ จากนั้นองค์กรพิสูจน์สิทธิ์จะทำการรับรองข้อมูลแล้วจึงส่งเอกสารสิทธิ์กลับมาที่เครื่องของเรา

2. การขอเอกสารสิทธิ์อีกรูปแบบหนึ่งคือ การส่งเอกสารสิทธิ์ที่ไม่ผ่านระบบเครือข่ายกล่าวคือ ผู้ขอต้องไปทำการขอที่สำนักงานขององค์กรพิสูจน์สิทธิ์โดยตรง โดยทำเรื่องของเอกสารสิทธิ์ และกรอกข้อมูลส่วนตัว องค์กรพิสูจน์สิทธิ์จะทำการรับรองและบันทึกข้อมูลของผู้ขอแล้วทำการส่งเอกสารสิทธิ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พร้อมทั้งคีย์ส่วนตัวให้กับผู้ขอในรูปแบบของไฟล์ในแผ่นดิสก์ให้ผู้ขอนำไปบันทึกในเครื่องของตนเองต่อไป

บริการพิสูจน์สิทธิ์แบบ X.509 (X.509 Authentication Service) ระบบ X.509 เป็นระบบพิสูจน์สิทธิ์ที่สำคัญมากในระบบเครือข่าย โดย X.509 เป็นอนุกรมย่อยของ X.500 ซึ่งกำหนดมาตรฐาน ITU-T โดยขณะที่ X.500 เป็นตัวกำหนดโครงสร้างในลักษณะที่เป็นไดเรกทอรีหรือโครงสร้างนั้น X.509 จะทำหน้าที่ในการพิสูจน์สิทธิ์ให้กับส่วนต่าง ๆ ของไดเรกทอรีนั้น สำหรับรูปแบบการใช้งานจะเน้นไปที่การพิสูจน์บุคคล เพื่อยืนยันการติดต่อเป็นสำคัญ การทำงานของ X.509 จะมีโครงสร้างการทำงานที่เป็นไดเรกทอรี โดยในที่นี้ไดเรกทอรีจะทำหน้าที่เป็นที่เก็บข้อมูลที่ใช้ในการยืนยัน ซึ่งโดยทั่วไปจะอยู่ในรูปของเอกสารสิทธิ์ซึ่งในเอกสารสิทธิ์จะบรรจุคีย์สาธารณะของผู้ใช้ที่เข้ารหัสโดยคีย์ส่วนตัวขององค์กรที่จ่ายใบเอกสารสิทธิ์มาให้ สำหรับการทำงานของ X.509 นั้นจะมีขอบเขตการนำไปใช้งานที่กว้างขวางมาก เช่น ใช้ในการทำ Mail Security ใช้ในการทำ IP Security ใช้ในการทำ Web Security หรือหากจะกล่าวว่าเมื่อใดที่ต้องการพิสูจน์บุคคลหรือยืนยันเครื่องคอมพิวเตอร์แล้ว ก็มักจะอยู่ในขอบข่ายการทำงานของ X.509 เสมอ X.509 ได้ถูกนำเสนอเมื่อปี 1988 จากนั้นได้ผ่านการปรับปรุงเป็นลำดับขั้น ในประเด็นต่าง ๆ รวมทั้งเรื่องความปลอดภัยด้วย จากนั้นก็ได้ออกมาเป็นข้อเสนอที่ปรับปรุงแล้วในปี 1993 และปรับปรุงอีกครั้งในปี 1995 โดยการทำงานของ X.509 จะใช้การเข้ารหัสแบบคีย์สาธารณะแล้วใช้มาตรฐานลายมือชื่อดิจิตอลในการรับรองข้อมูล สำหรับอัลกอริธึมนั้น ไม่ได้ระบุแน่นอนโดยสามารถเลือกใช้ได้หลายตัวแต่ที่แนะนำคืออาร์เอสเอ

รูปแบบทั่วไปของเอกสารสิทธิ์

- เวอร์ชัน (Version) แสดงหมายเลขเวอร์ชัน เพราะในแต่ละเวอร์ชันจะมีรูปแบบของข้อมูลที่ไม่เหมือนกันก็ได้ โดยปกติจะเป็นเวอร์ชัน 1 แต่หากในเอกสารสิทธิ์มีการใช้

- หมายเลขลำดับ (Serial Number) เป็นเลขจำนวนเต็ม โดยจะต้องไม่ซ้ำกันในองค์กรที่ออกเอกสารสิทธิ์ โดยเลขนี้จะเป็นเลขที่จะใช้อ้างถึงแต่ละเอกสารสิทธิ์ที่สร้างขึ้นมา

- อัลกอริธึมที่ใช้สร้าง (Signature Algorithm Identifier) เป็นฟิลด์ที่ระบุอัลกอริธึมที่ใช้ในการสร้างเอกสารสิทธิ์

- ชื่อผู้ออกเอกสารสิทธิ์ (Issue Name) เป็นชื่อขององค์กรที่ออกเอกสารสิทธิ์

- ช่วงเวลาที่รับรองเอกสารสิทธิ์ (Period of Validity) เป็นตัวบอกว่า ให้ใช้เอกสารสิทธิ์นี้ตั้งแต่วันที่เท่าไหร่และสิ้นสุดวันที่เท่าไหร่

- ชื่อเจ้าของเอกสารสิทธิ์ (Subject Name) เป็นชื่อของบุคคลที่เอกสารสิทธิ์ใบนี้อ้างถึงหรือแทนตัวบุคคลนั้น

- ข้อมูลของคีย์สาธารณะ (Subject's Public Key Information) เป็นฟิลด์ที่เก็บคีย์สาธารณะและระบุถึงอัลกอริธึมที่ใช้กับคีย์นี้ขึ้นมา รวมถึงพารามิเตอร์อื่น ๆ ด้วย

- ตัวระบุผู้ออกเอกสารสิทธิ์ (Issuer Unique Identifier) เป็นฟิลด์ออปชันที่ใช้ในการระบุถึงองค์กรที่ออกเอกสารสิทธิ์ ในกรณีที่ชื่อ X.500 มีการนำไปใช้กับส่วนอื่น ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ตัวระบุผู้ขอเอกสารสิทธิ์ (Subject Unique Identifier) เป็นฟิลด์ที่ใช้ในการระบุถึงตัวบุคคลที่เป็นเจ้าของเอกสารสิทธิ์ ในกรณีที่ชื่อ X.500 มีการนำไปใช้กับส่วนอื่น ๆ

- ส่วนขยาย (Extension) เป็นกลุ่มของฟิลด์ที่เพิ่มเติมข้อมูลอื่น ๆ เข้ามาด้วย

- ลายมือชื่อ (Signature) จะบรรจุเมสเซจไดเจสต์ของข้อมูลในทุกฟิลด์ที่เข้ารหัสด้วยคีย์ส่วนตัวขององค์กรพิสูจน์สิทธิ์เพื่อเป็นการยืนยันว่า เอกสารสิทธิ์นี้สร้างมาจากองค์กรดังกล่าวจริง ๆ โดยจะมีข้อมูลที่ระบุวิธีการแฮชและวิธีการเข้ารหัสด้วย

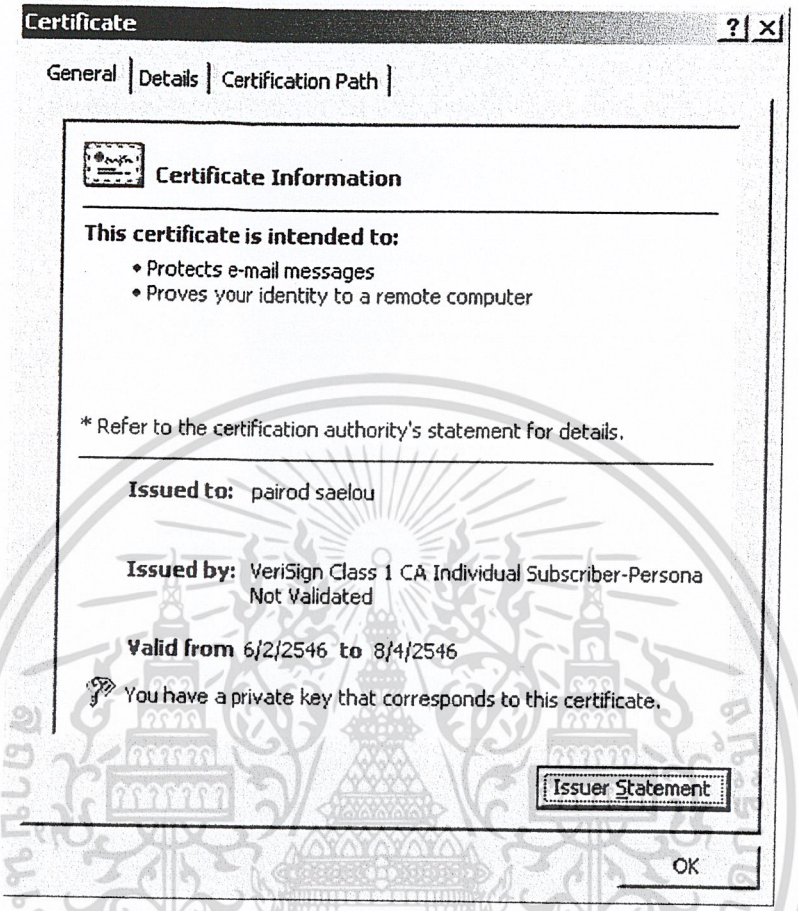
ในการใช้งานเอกสารสิทธิ์จะมีช่วงเวลาใช้งานที่จำกัดแน่นอน ดังนั้นหากผู้ใช้ต้องการใช้เอกสารสิทธ์ต่อไป ก็ต้องขอต่ออายุเอกสารสิทธิ์ก่อนที่จะหมดอายุ แต่หากจะมีการหมดอายุโดยที่ไม่ขอต่อหรือมีการเลิกใช้เอกสารสิทธิ์อาจจะเนื่องมาจากพนักงานลาออก หรืออาจจะเนื่องจากเอกสารสิทธิ์นี้ไม่ปลอดภัยแล้วก็ต้องทำการเรียกคืน (Revoke) และองค์กรพิสูจน์สิทธิ์จะต้องมีการจัดทำรายการเอกสารสิทธิ์ที่ถูกเรียกคืน (Certificate Revocation List - CRL) ซึ่งจะเก็บไว้ในไดเรกทอรีและรับรองโดยองค์กรพิสูจน์สิทธิ์ ซึ่งผู้ที่ต้องการตรวจสอบเอกสารสิทธิ์ว่าเป็นเอกสารสิทธิ์ที่ไม่ใช้งานแล้วหรือไม่ ก็ต้องขอรายการเอกสารสิทธิ์ที่ถูกเรียกคืนไปตรวจสอบ

และเนื่องจากเอกสารสิทธิ์ไม่สามารถปลอมได้ ดังนั้นการเก็บเอกสารสิทธิ์ไว้ที่องค์กรพิสูจน์สิทธิ์จึงไม่ต้องมีกลไกพิเศษมาป้องกันแต่อย่างใด กล่าวคือผู้ใช้คนใดที่เป็นสมาชิกขององค์กรพิสูจน์สิทธิ์ก็สามารถเข้าถึงเอกสารสิทธิ์ของผู้ใช้คนอื่น ๆ ได้ทุกคน โดยเอกสารสิทธิจะเก็บอยู่ในไฟล์เพียงไฟล์เดียวที่มีขนาดเล็ก นอกจากจะสามารถขอเอกสารสิทธิ์จากองค์กรพิสูจน์สิทธิ์แล้วผู้ใช้ยังสามารถส่งเอกสารสิทธิ์ไปให้กันเองได้อีกด้วย โดยผ่านทางสื่อต่าง ๆ เช่น จดหมายอิเล็กทรอนิกส์, ส่งผ่านแผ่นดิสก์เก็ตเป็นต้น

อย่างไรก็ตาม เนื่องจากระบบเครือข่ายในปัจจุบันมีขนาดใหญ่โตกว้างขวางมาก และการติดต่อสื่อสารก็ไม่ได้มีลักษณะเฉพาะกลุ่มอีกแล้ว ดังนั้นการที่จะให้ผู้ใช้ทุกคนมาใช้เอกสารสิทธิ์ที่รับรองโดยองค์กรพิสูจน์สิทธิ์เดียวกันทั้งหมดก็อาจเป็นเรื่องยาก หากผู้ใช้ 2 คนใช้เอกสารสิทธิ์ที่รับรองจากองค์กรพิสูจน์สิทธิ์คนละแห่งก็จะไม่สามารถตรวจสอบเอกสารสิทธิ์ของอีกฝ่ายได้ว่าเป็นฉบับจริงหรือไม่ซึ่งในกรณีเช่นนี้ ก็อาจจะใช้วิธีสำเนาคีย์สาธารณะขององค์กรพิสูจน์สิทธิ์ของผู้ใช้อีกคนหนึ่งมาทำการตรวจสอบเองก็สามารถทำได้ เช่น กำหนดให้มี CA-A และ CA-B โดยให้บริการกับผู้ใช้ A และ B แต่เนื่องจากในครั้งแรกที่ A สำเนาคีย์สาธารณะของ CA-B มานั้นอาจเกิดการปลอมแปลงได้ เพราะสิ่งที่เรามีอยู่ก็คือคีย์สาธารณะของ CA-A ของเรา แต่เอกสารสิทธิ์ของ CA-B ทำให้เราไม่สามารถตรวจสอบว่าเอกสารสิทธิ์ที่ได้รับมานั้นเป็นฉบับที่ถูกต้องหรือไม่ ดังนั้นวิธีดังกล่าวจึงถือว่า ไม่มีความปลอดภัยเพียงพอ สำหรับอีกวิธีการอีกแบบ คือให้ CA-A เก็บเอกสารสิทธิ์ของ CA-B เอาไว้ด้วยและ CAB ก็เก็บเอกสารสิทธิ์ของ CA-A เอาไว้เช่นกัน ด้วยวิธีนี้เราก็สามารถให้องค์กรพิสูจน์สิทธิ์ตรวจสอบเอกสารสิทธิ์ได้ไม่ว่าเอกสารสิทธิ์นั้นจะรับรองจาก CA-A หรือ CA-B ก็ตาม เช่น ผู้ใช้ A ต้องการตรวจสอบเอกสารสิทธิ์ที่รับรองจาก CA-A ก็สามารถทำได้เลยเพราะรู้คีย์สาธารณะของ CA-A อยู่แล้วเนื่องจากเป็นสมาชิก CA-A และหากผู้ใช้ A ต้องการตรวจสอบเอกสารสิทธิ์ที่รับรองโดย CA-B ผู้ใช้ A ก็ขอเอกสารสิทธิ์ของ CA-B

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จาก CA-A โดยเอกสารสิทธิ์ดังกล่าวจะรับรองโดย CA-A ดังนั้นจึงแน่ใจได้ว่าเอกสารสิทธิ์ของ CA-B ที่ได้รับนั้นเป็นของจริงและคีย์สาธารณะของ CA-B ก็เป็นของจริง



รูปที่ 3.10 แสดงเอกสารสิทธิ์ที่ตรวจสอบโดยบราวเซอร์

จากนั้นจึงนำเอาคีย์สาธารณะของ CA-B ไปตรวจสอบเอกสารสิทธิ์อีกทีก็จะทราบได้ว่าเอกสารสิทธิ์นั้นเป็นของจริงหรือไม่ และนอกเหนือจาก CA-A และ CA-B แล้วการเชื่อถือ(Trust)กันเช่นนี้ยังสามารถกระทำกับองค์กรพิสูจน์สิทธิ์อื่นๆ ไปเรื่อยๆ อย่างไรก็ตามหากองค์กรพิสูจน์สิทธิ์มีเป็นจำนวนมากแล้ว ก็มีความจำเป็นที่จะต้องจัดโครงสร้างการเชื่อถือกันขององค์กรพิสูจน์สิทธิ์ให้เป็นระบบ ไม่เช่นนั้นก็อาจมีการเชื่อถือกันแบบยุ่งเหยิงและทำให้การทำงานเป็นไปอย่างไม่มีประสิทธิภาพได้ ซึ่งมาตรฐาน X.509 ก็ได้แนะนำให้มีการใช้ระบบเชื่อถือกันในรูปของความสัมพันธ์แบบระดับชั้น (Hierachial)

3.16 Certificate Authority

เอกสารสิทธิดิจิทอลจะถูกออกโดยองค์กรที่ออกเอกสารสิทธิที่ซึ่งเป็นองค์กร(บุคคลที่สาม)ที่เชื่อถือได้เมื่อองค์กรที่ออกเอกสารสิทธิสร้างเอกสารสิทธิดิจิทอล แล้ว จะทำการเข้ารหัสด้วยกุญแจส่วนตัว ขององค์กรที่ออกเอกสารสิทธินั้น แล้วจะทำการส่งไปให้กับผู้ที่ขอเอกสารสิทธ์นั้น โดยผู้ขอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารสิทธิ์จะยืน ยัน ตัวขององค์กรที่ออกเอกสารสิทธิ นั้น ด้วยกุญแจสาธารณะ ซึ่งถูกออกด้วย high-level องค์กรที่ออกเอกสารสิทธิ ซึ่งมีความน่าเชื่อถือกว่า



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

WS-Security

ความจำเป็นที่จะต้องใช้ WS-Security แทน Secure Sockets Layer(SSL)

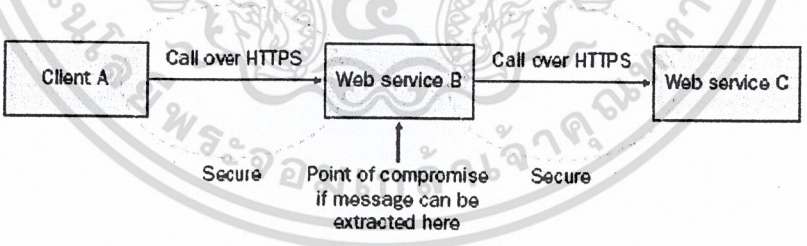
เนื่องจาก SSL นั้นยังมีข้อเสียอยู่บางอย่างที่ไม่เหมาะสมที่จะนำมาใช้กับเว็บเซอร์วิส โดยมีเหตุผล 2 ข้อ ดังต่อไปนี้

1. SSL นั้นเป็นการรักษาความปลอดภัยแบบ จุด - จุด(point-to-point security) นั้นหมายถึงว่าจะมีความปลอดภัยแต่จากจุดเริ่มถึงจุดที่มันส่งไปหา ซึ่งการรักษาความปลอดภัยแบบนี้จะใช้ได้แค่กรณีแบบติดต่อกันทีละแค่ 2 เครื่องเท่านั้น ดังรูปด้านล่าง



รูปที่ 4.1 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS

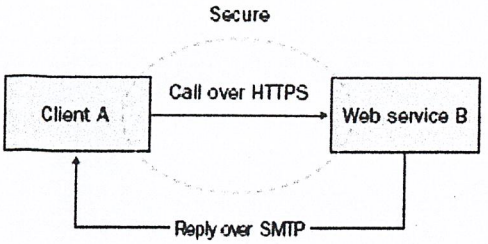
แต่ถ้าเว็บเซอร์วิสบีส่งต่อไปให้เว็บเซอร์วิสซี กรณีนี้ถึงแม้ว่าจะมีความปลอดภัยจาก เอ ไปบี และจากบี ไปซีก็ตามแต่เราก็ไม่สามารถมั่นใจได้เลยว่าข้อมูลที่ถูกส่งจากเอ ไปซีจะมีความปลอดภัย เพราะสามารถเปลี่ยนแปลงข้อมูลได้ในเว็บเซอร์วิสบี ได้ ดังรูปด้านล่าง



รูปที่ 4.2 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS และ การส่งต่อ

2. เว็บเซอร์วิสนั้นมีความจำเป็นที่ต้องทำงานบนโปรโตคอลหลายอย่าง ซึ่ง SSL นั้นทำงานอยู่บน http เท่านั้น เพราะฉะนั้นเมื่อมีการทำงานบนโปรโตคอลอื่น เช่น SMTP เป็นต้นก็ไม่สามารถที่จะมีความปลอดภัยได้ ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 การรักษาความปลอดภัยแบบ จุด - จุดบน HTTPS และ ไม่ปลอดภัยบน SMTP

จากข้อเสียดังกล่าวข้างต้น WS-Security นั้นสามารถแก้ไขข้อเสียดังกล่าวได้ เพราะ WS-Security นั้นเป็นวิธีการรักษาความปลอดภัยแบบ จบ-จบ(end-end) และมันยังไม่ยึดติดกับโปรโตคอลใดอีกด้วย

4.1 WS-Security

เป็นเทคโนโลยีพื้นฐานเพื่อเพิ่มความปลอดภัยให้แก่ SOAP อธิบายโดย **WS-Security** จุดประสงค์เพื่อเพิ่มระดับความปลอดภัยแบบ end-to-end ของข้อความ SOAP และจนถึงตอนนี้ ความสามารถของ **WS-Security** ยังไม่สามารถรองรับได้เกิน 20 หน้า เพราะว่า **WS-Security** เป็นเทคโนโลยีใหม่ แต่สามารถแทนที่การรักษาความปลอดภัยด้วย SOAP

สิ่งที่เป็นพื้นฐานของ **WS-Security** เป็นส่วนประกอบซึ่งเป็นส่วนหนึ่งของ SOAP โครงสร้างข้อมูลของ SOAP จะรวมอยู่ในซีเคียวริที อีลิเม้นท์ (Security Element) ดังตัวอย่าง รูปที่ 4.4 นี้

```
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
<s:Header>
 <wsse:Security>
  •••
 </wsse:Security>
</s:Header>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
<s:Body>
  •••
</s:Body>
</s:Envelope>
```

รูปที่ 4.4 Security Element

โครงสร้างพื้นฐานของ WS-Security สามารถอธิบายโดยเนมสเปซ (namespace) ของมัน ผู้สร้าง WS-Security จะขึ้นกับมาตรฐานที่มีอยู่ และเทคโนโลยีต่างๆ ที่เป็นไปได้

WS-Security อธิบายได้จากความมั่นคงที่ได้รับ ซึ่งอนุญาตจากผู้รับถึงความถูกต้องของข้อมูลที่ได้รับว่าไม่เปลี่ยนแปลงในขณะที่ทำการส่ง และมีสภาพที่เป็นความลับ ซึ่งรับรองว่าข้อมูลไม่ถูกเปิดเผยในระหว่างที่ทำการส่ง สามารถอธิบายวิธีการส่งของซีเคียวริตี้ โทคเคน เช่น การใช้ชื่อผู้ใช้ และรหัสผ่าน, ตั๋วเคอเบอร์รอส หรือ การรับรองจาก X.509 ในการอธิบาย

4.2 การส่งซีเคียวริตี้ โทเคน (Sending Security Tokens)

คำถามส่วนใหญ่ที่เกี่ยวกับการรับข้อความ เป็นคำถามพื้นฐาน ซึ่งมีแนวทางที่แตกต่างกันในการแสดงบนเครือข่าย เช่น ชื่อผู้ใช้ และ รหัสผ่าน ,ตั๋วเคอเบอร์รอส , X.509 และอื่นๆ แนวทางที่แตกต่างกันในหลักฐานการพิสูจน์ตัวตน โดยใช้ชื่อผู้ใช้ ดังตัวอย่าง คุณอาจจัดหารหัสผ่านใช้ในการพิสูจน์ตัวตนที่แท้จริงว่าเป็นคุณหรือไม่

วิธีที่ฉลาดกว่าคือ ผู้สร้าง WS-Security เลือกที่จะไว้ใจมากกว่าวิธีการในการส่งข้อมูล และการพิสูจน์ตัวตน ในความจริงแล้ว WS-Security ไม่ได้กำหนดวิธีการในการพิสูจน์ตัวตนไว้ทั้งหมด แต่จะกล่าวถึงการอธิบายเกี่ยวกับซีเคียวริตี้ โทเคน หลายๆ แบบว่าจะใช้ส่งได้อย่างไร

ในขณะที่อนุญาตให้ใช้ซีเคียวริตี้ โทเคน หลายๆ แบบนั้น WS-Security เจาะจงที่จะอธิบายการทำงานเพียงสามการทำงาน การทำงานที่ง่ายที่สุดคือ การส่งซีเคียวริตี้ โทเคนที่ประกอบไปด้วยชื่อผู้ใช้ และรหัสผ่าน โดย WS-Security ได้กำหนด < UsernameToken> ซึ่งประกอบไปด้วยชื่อผู้ใช้ และรหัสผ่าน ดังตัวอย่างข้างล่าง

```
<wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
    <wsse:UsernameToken>
        <wsse:Username>Diana</wsse:Username>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            <wsse:Password>mY5ecRet</wsse:Password>
        </wsse:UsernameToken>
</wsse:Security>
```

การทำงานแบบที่สองที่ถูกอธิบายโดย **WS-Security** คือ การส่งซีเคียวริตี้ โทเคน ที่ประกอบไปด้วยตั๋ว Kerberos ตัวตั๋วนี้จะส่งในส่วนของ SOAP header ของซีเคียวริตี้ อีลิเม้นท์ ดังตัวอย่างนี้

```
<wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
        <wsse:BinarySecurityToken
                ValueType="wsse:Kerberosv5ST"
                EncodingType="wsse:Base64Binary">
                        QMwcAG ...
        </wsse:BinarySecurityToken>
</wsse:Security>
```

การทำงานในแบบที่สาม **WS-Security** อธิบายเพื่อแสดงให้เห็นถึงซีเคียวริตี้ โทเคน, เอกสารสิทธิ X.509 รวมถึงการไว้ใจได้ของ <BinarySecurityToken> ดังตัวอย่างนี้

```
  <wsse:Security
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
        <wsse:BinarySecurityToken
                ValueType="wsse:X509v3"
                EncodingType="wsse:Base64Binary">
                        KkFPle ...
        </wsse:BinarySecurityToken>
  </wsse:Security>
```

ทั้งสามการทำงานนี้จะใช้สำหรับฝังซีเคียวริตี้ โทเคน ในส่วนของเฮดเดอร์ของข้อความของ SOAP นั้น **WS-Security** ได้กำหนด <SecurityTokenReference> ด้วย มันจะเป็นชื่อที่ใช้ในการเสนอแนะในอีลิเม้นท์ จะประกอบด้วยยูอาร์ไอ (URI) ที่จะบอกว่าจะหาซีเคียวริตี้ โทเคนได้ที่ไหน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การรักษาความสมบูรณ์ (Integrity)

การส่งข้อมูล และการพิสูจน์ตัวตนของข้อความ SOAP อาจจะก่อให้เกิดปัญหาหลายอย่างได้ ดังนั้น **WS-Security** ได้นำวิธีการที่น่าสรรเสริญยกย่องมาใช้ ภายใต้เงื่อนไขของการรักษาความสมบูรณ์สำหรับข้อความของ SOAP วิธีในการแก้ไขคือ การใช้ลายมือชื่อ ซึ่งเป็นมาตรฐานของ W3C โดยมีชื่อเรียกว่า ลายมือชื่อ XML (XML Signature) มันจัดการเกี่ยวกับรายละเอียดกลไกสำหรับการลงลายมือชื่อดิจิทอลของเอกสาร XML

สิ่งสำคัญในการอธิบายโดยลายมือชื่อ XML คือ <Signature> ซึ่งรวมทั้งลายเซ็นอิเล็กทรอนิคส์และข้อมูลเกี่ยวกับลายมือชื่อนี้ ถึงแม้ว่ามันจะมีการทำงานเพียงไม่กี่อย่าง ตัวอย่างของ <Signature> เช่น การใช้ SOAP ที่ประกอบด้วย <SignedInfo> และ <KeyInfo>

<SignedInfo> จะอธิบายข้อมูลที่จะใช้ในการลงลายมือชื่อ ในอีลิเม้นท์นี้จะประกอบด้วยอีลิเม้นท์ย่อยอีกหลายอีลิเม้นท์ ซึ่งแต่ละอีลิเม้นท์จะเกี่ยวกับข้อมูลที่เฉพาะเจาะจงแตกต่างกันไป

<KeyInfo> เป็นตัวบอกว่ากุญแจไหนควรจะใช้กับลายเซ็นนี้ ส่วนตัวกุญแจนั้นควรจะมีเอกลักษณ์เฉพาะตัว เช่น การอ้างถึงเอกสารสิทธิ ซึ่งประกอบด้วยกุญแจสาธารณะ ในอีลิเม้นท์นี้สามารถที่จะตัดทิ้งได้ตั้งแต่ที่ผู้รับอาจจะรู้วิธีการใช้กุญแจ โดยวิธีอื่น

ค่าเหล่านี้จะเป็นพื้นฐานของลายมือชื่อ XML ตัวอย่าง อีลิเม้นท์นี้ของ **WS-Security** ดูได้ในรูปที่ 4.5 ซึ่งเป็นตัวอย่างของข้อความ SOAP ที่ประกอบด้วยลายเซ็นอิเล็กทรอนิคส์ ดังนี้

```
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
<s:Header>
<wsse:Security>
<wsse:BinarySecurityToken
ValueType="wsse:X509v3"
EncodingType="wsse:Base64Binary"
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 wsu:Id="X509Cert">
 KkFPle ...
</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <ds:SignedInfo>
  <ds:CanonicalizationMethod
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14N"/>
  <ds:SignatureMethod
   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <ds:Reference URI="#MessageBody">
   <ds:DigestMethod
    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
   <ds:DigestValue>
    aOb4Luuk...
   </ds:DigestValue>
  </ds:Reference>
 </ds:SignedInfo>
 <ds:SignatureValue>
  A9qqIrtE3xZ...
 </ds:SignatureValue>
 <ds:KeyInfo>
  <wsse:SecurityTokenReference>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    <wsse:Reference URI="#X509Cert"/>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
  </ds:Signature>
 </wsse:Security>
</s:Header>
<s:Body wsu:Id="MessageBody">
 ...
</s:Body>
</s:Envelope>
```

รูปที่ 4.5 SOAP Message with Digital Signature

4.4 การรักษาความลับ (Confidentiality)

การเก็บข้อมูลให้ปลอดภัยจากคนที่สอดรู้สอดเห็นในขณะที่มันกำลังอยู่ในระหว่างการส่งข้อมูลเป็นเรื่องที่สำคัญ ดังนั้นการบริการสามอย่างที่กำหนดโดย **WS-Security** คือ การรักษาความลับซึ่งจัดการเกี่ยวกับการเข้ารหัสข้อมูลบางส่วน หรือทั้งหมดของข้อความ SOAP ก่อนที่จะทำการส่งต่อไป ซึ่งผู้สร้าง **WS-Security** จะเป็นผู้สร้างมาตรฐานสำหรับความไว้วางใจได้ ซึ่งมาตรฐานหนึ่งที่ถูกเลือกนำมาใช้คือการเข้ารหัส XML ซึ่งมาจาก W3C การใช้มาตรฐานนี้นั้น **WS-Security** อนุญาตให้เข้ารหัสทั้งหมด หรือเฉพาะบางส่วนของข้อมูลในเฮดเดอร์ ,บอดี้ และข้อมูลที่แนบมาของข้อความ SOAP นั้น การเข้ารหัส XML ไม่ได้มีความซับซ้อนมากเท่าใดนัก สำหรับตัวอย่างที่ใช้สามอีลิเม้นท์ ของ XML ในการอธิบายมาตรฐานนี้คือ <EncryptedData>, <EncryptedKey> และ <ReferenceList>

สิ่งที่สำคัญที่สุดของทั้งสามสิ่งนี้คือ <EncryptedData> แค่ชื่อก็พอจะบอกได้แล้วว่าอีลิเม้นท์นี้ประกอบด้วยข้อมูลที่ถูกเข้ารหัสแล้วอยู่ในอีลิเม้นท์ย่อยซึ่งจะมีชื่อเรียกว่า <CipherData> ในอีลิเม้นท์นี้จะประกอบด้วยอีลิเม้นท์ย่อยซึ่งจะแสดงอัลกอลิทึมในการเข้ารหัส , กุญแจที่จะใช้ในการเข้ารหัส และอื่นๆ

หมายเหตุ ไม่มี SOAP header ใน message ซึ่งในขณะนี้ถือว่ายังไม่เป็นแอพพลิเคชั่น ที่แท้จริง มันเป็นไปได้ที่จะใช้การเข้ารหัส XML ในการจัดการโดย **WS-Security** จะไม่มีซีเคียวริตี้ อีลิเม้นท์ หรือ Header อีลิเม้นท์ อื่นๆ ในตัวอย่างง่ายๆ เช่น ในส่วนของ body ของข้อความ จะประกอบด้วยเพียง <EncryptedData> กับอีลิเม้นท์ย่อยอีก 3 อีลิเม้นท์ คือ 1.อีลิเม้นท์แรกคือ <EncryptionMethod> จะบอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถึงอัลกอลิธึม ในการเข้ารหัสข้อมูล ในกรณีนี้อัลกอลิทึมชนิด TripleDES จะถูกเลือก และ กุญแจที่เหมือนกันจะถูกใช้ในการเข้ารหัส และถอดรหัส 2. อีลิเม้นท์ที่สองคือ <KeyInfo> จะมีการใช้ข้อมูลจาก XML Signature มันจะสมมติว่าทั้งสองฝั่งที่ติดต่อกันนั้นรู้ กุญแจที่ใช้ในการเข้ารหัส และถอดรหัส ตัวอย่างชื่อของกุญแจ เช่น "CN=Key13, C=US" ไม่ว่าชื่อใดก็ตามที่ถูกใช้นั้น สิ่งที่สำคัญที่สุดคือทั้งสองฝั่งต้องเข้าใจที่กุญแจนี้อ้างถึง 3.อีลิเม้นท์สุดท้ายคือ <CipherData> เป็นอีลิเม้นท์ย่อยที่จะมี <CipherValue> เป็นอีลิเม้นท์ย่อยอีกทีที่จะเป็นการเข้ารหัสข้อมูลที่แท้จริง

มันเป็นไปได้ที่จะส่งกุญแจที่ถูกเข้ารหัสในข้อความเดียวกันที่ข้อมูลนั้นถูกเข้ารหัสด้วยกุญแจ ตัวอย่างในกรณีนี้เมื่อข้อมูลถูกเข้ารหัสโดยใช้คู่คีย์โดยใช้ TripleDES ดังตัวอย่างรูปที่ 4.6 ดังนี้

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<s:Body>
 <xenc:EncryptedData>
  <EncryptionMethod
   Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
  <ds:KeyInfo>
   <ds:KeyName>
    CN=Key13, C=US
   </ds:KeyName>
  </ds:KeyInfo>
 <xenc:CipherData>
  <xenc:CipherValue>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    r5KipsDV . . .
   </xenc:CipherValue>
  </xenc:CipherData>
 </xenc:EncryptedData>
</s:Body>
</s:Envelope>
```

รูปที่ 4.6 SOAP Message with Encrypted Body

แล้วเมื่อคู่คีย์ถูกเข้ารหัสโดยใช้กุญแจสาธารณะแล้วทำการส่งข้อมูลต่อไป เมื่อได้รับข้อความแล้ว ผู้รับสามารถที่จะใช้กุญแจส่วนตัวในการถอดรหัสเพื่อดึงเอาคู่คีย์ออกมา แล้วใช้คู่คีย์นี้ในการถอดรหัสข้อมูลที่แท้จริง การส่งกุญแจที่เข้ารหัสนั้น การเข้ารหัส XML จะกำหนดอีลิเม้นท์กับชื่อที่ชัดเจนของ <EncryptedKey> เพื่อใช้กับข้อความ SOAP ซึ่ง **WS-Security** จะกำหนดว่าอีลิเม้นท์นี้ควรจะอยู่ใน <Security> ดังนั้น ข้อความ SOAP จะมีทั้งข้อมูลที่เข้ารหัสแล้ว และคู่คีย์ที่เข้ารหัสแล้ว ดูตัวอย่างได้จากรูปที่ 4.7 ดังนี้

```
<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
 xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<s:Header>
 <wsse:Security>
  <xenc:EncryptedKey>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
<ds:KeyInfo>
<ds:KeyName>
CN=Key13, C=US
</ds:KeyName>
</ds:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>
ir4DfG . . .
</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList>
<xenc:DataReference URI="#EncryptedBody"/>
</xenc:ReferenceList>
</xenc:EncryptedKey>
</wsse:Security>
</s:Header>
<s:Body>
<xenc:EncryptedData wsu:Id="EncryptedBody">
<xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
<xenc:CipherData>
 <xenc:CipherValue>
r5KipsDV . . .
 </xenc:CipherValue>
 </xenc:CipherData>
</xenc:EncryptedData>
</s:Body>
</s:Envelope>
```

รูปที่ 4.7 Encrypted SOAP Message with Key

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

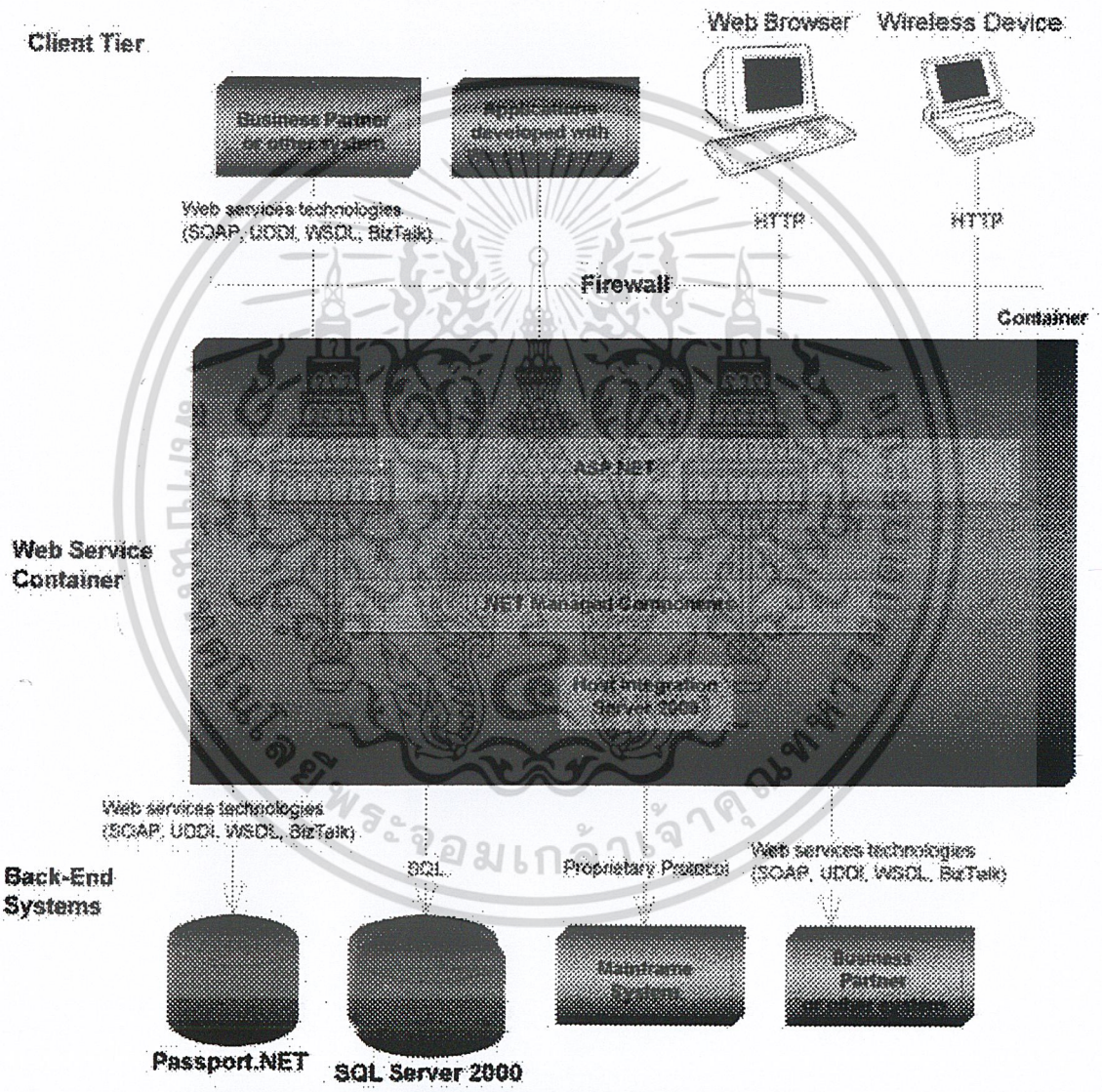
บทที่5

เปรียบเทียบ .NET เว็บเซอร์วิส และจาวาเว็บเซอร์วิส

5.1 แพลตฟอร์มการพัฒนา

แบ่งเป็น 2 แพลตฟอร์ม คือ .NET และ จาวา

5.1.1 แพลตฟอร์มของ .NET



รูปที่ 5.1 แพลตฟอร์มการพัฒนาระบบภายใต้เทคโนโลยี .NET

<u>.NET แอพพลิเคชั่น</u>

ถูกดีพลอยอยู่ภายในคอนเทนเนอร์ ที่มีเซอร์วิสที่จำเป็นให้บริการสำหรับเอนเตอร์ไพรส์ แอพพลิเคชั่น เช่น ทรานแซกชัน ความปลอดภัย และ เมสเสจจิ้งเซอร์วิส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชั้นของธุรกิจ

ของ.NET แอพพลิเคชั่นถูกสร้างโดยใช้ .NET เมเนจคอมโพเนนต์(managed component) ชั้นนี้ทำในส่วนของการประมวลผลทางธุรกิจและการจัดการข้อมูล การติดต่อไปยังฐานข้อมูลโดยผ่าน ADO.NET มีการติดต่อกับระบบเก่า(legacy system) โดยผ่าน Microsoft Host Integration Server 2000 เช่นติดต่อกับ COM Transaction Integrator (COM TI) หรือติดต่อคู่ค้าทางธุรกิจผ่านเว็บเซอร์วิส

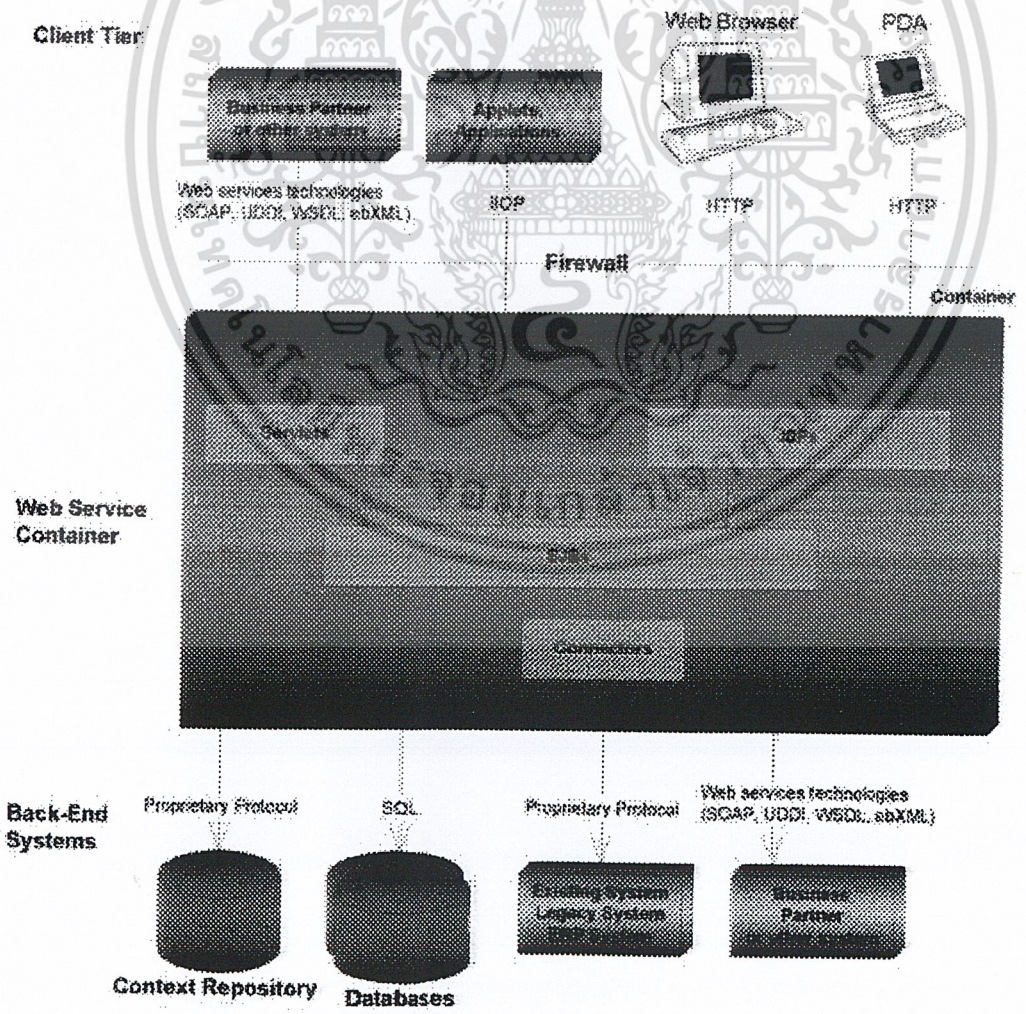
คู่ค้าทางธุรกิจ

สามารถใช้ .NET แอพพลิเคชั่นติดต่อโดยผ่านเว็บเซอร์วิส

ไคลเอนต์ที่เรียกใช้งาน

ก็เป็นได้ทั้งผ่านเว็บ ASP.NET ซึ่งให้ผลได้ในรูปของ HTML,XHTML,WML และรูปแบบของวินโดส์ฟอร์ม

5.1.2 แพลตฟอร์มของ จาวา



รูปที่ 5. 2 แพลตฟอร์มการพัฒนาระบบภายใต้เทคโนโลยี J2EE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

J2EE แอพพลิเคชั่น

ถูกดีพลอยอยู่ภายในคอนเทนเนอร์ ที่มีเซอร์วิสที่จำเป็นให้บริการสำหรับเอนเตอร์ไพรส์แอพพลิเคชั่น เช่น ทรานแซกชัน ความปลอดภัย

ชั้นของธุรกิจ

ในแอพพลิเคชั่นขนาดใหญ่ถูกสร้างโดยใช้ Enterprise JavaBeans (EJB) คอมโพเน็นต์ ติดต่อฐานข้อมูลโดย JDBC หรือ SQL/J หรือติดต่อกับระบบเดิมผ่าน Java Connector Architecture (JCA) ติดต่อผ่านเว็บเซอร์วิสโดยใช้ จาวา API สำหรับXML (JAX APIs)

คู่ค้าทางธุรกิจ

ติดต่อผ่านเว็บเซอร์วิส ตัวเซิร์ฟเลท(servlet) เป็นตัวคอยรับการร้องขอที่เข้ามาละส่งผลลัพธ์กลับไป โดยใช้ JAX APIs เป็นตัวจัดการเกี่ยวกับเว็บเซอร์วิส

ไคลเอนต์ที่เรียกใช้งาน

นอกเหนือจากการผ่านเว็บ และเว็บเซอร์วิสแล้ว ยังมีแอพเพลตที่สามารถจะติดต่อกับจาวาบีนผ่านยังโปรโตคอล Internet Inter-ORB Protocol (IIOP) ได้ด้วย

5.2 เปรียบเทียบเทคโนโลยีจากทั้งสองค่าย

เริ่มจากแสดงให้เห็นถึงความคล้ายคลึงกันของทั้งสองแพลตฟอร์มดังตารางข้างล่าง และจะทำการพิจารณาเปรียบเทียบให้เห็นในแต่ละหัวข้อ

Feature	J2EE	.NET
Type of technology	Standard	Product
Middleware Vendors	30+	Microsoft
Interpreter	JRE	CLR
Dynamic Web Pages	JSP	ASP.NET
Middle-Tier Components	EJB	.NET Managed Components
Database access	JDBC SQL/J	ADO.NET
SOAP, WSDL, UDDI	Yes	Yes
Implicit middleware (load-balancing, etc)	Yes	Yes

ตาราง 5.1 ตารางแสดงการสนับสนุนการทำงานด้านต่างๆของ 2 แพลตฟอร์ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.1 Vendor Solution

ทางด้านของ J2EE นั้นเป็นเสปกที่ออกมาโดยมีเวนเดอร์(vendor)หลายเจ้าที่นำเอามาตรฐานเหล่านั้นไปพัฒนาผลิตภัณฑ์(product)ออกมารองรับ รวมทั้งมีเครื่องมือให้ใช้ค่อนข้างหลากหลาย ข้อดีคือมีตัวเลือกในการพัฒนาโซลูชั่นค่อนข้างหลากหลาย แต่ข้อเสียก็มาจากข้อดีของมันนั่นเองคือ ความไม่เข้ากันของแต่ละผลิตภัณฑ์ ไม่สามารถที่จะย้ายโค้ดโปรแกรมที่พัฒนาข้ามผลิตภัณฑ์ได้อย่างครบถ้วนซึ่งการพัฒนาระบบจริงนั้นเหมาะกับการเลือกใช้ผลิตภัณฑ์จากค่ายเดียวกันมากกว่า

ทางฝั่ง .NET ออกผลิตภัณฑ์ออกมานำ เนื่องจากเป็นผู้ออกเสปกและพัฒนาเองทั้งหมด รวมทั้งระบบที่ทำการสนับสนุนการทำงานส่วนมากก็ปเป็นของไมโครซอฟท์เอง

5.2.2 การสนับสนุนระบบเดิม

ในระบบเดิมที่มีการใช้งานโดยหลากหลายภาษาหลายระบบในส่วนของจาวานั้นก็มีหลากหลายทางในการทำให้สามารถใช้งานร่วมกันกับระบบเดิมเหล่านั้นได้

- ใช้ Java Message Service (JMS) เชื่อมกับระบบเก่า
- ใช้เว็บเซอร์วิสในการอินทริเกรตกับระบบต่างๆ
- ติดต่อผ่าน CORBA

รวมทั้งยังมี J2EE Connector Architecture (JCA) ทำหน้าที่เป็น ตัวติดต่อกับทรัพยากรอื่นๆ เช่นSAP R/3 ,CICS/COBOL และอื่นๆ

ฝั่ง .NET มีตัว Host Integration Server 2000 เป็นตัวช่วยจัดการเกี่ยวกับทรานแซกชันไปยังระบบเมนเฟรม หรือตัว Microsoft Message Queue (MSMQ) ในการทำงานร่วมกันกับระบบเดิมที่ใช้ IBM MQSeries และสุดท้ายคือตัว BizTalk Server2000 ทำการอินทริเกรตระบบบนโปรโตคอลของ B2B

5.2.3 การยอมรับทางการตลาด

ทางด้านจาวามีมากกว่า 50 บริษัทที่ให้การยอมรับในแพลตฟอร์ม

อีกด้านนั้นทีมการตลาดของไมโครซอฟท์ทำงานด้านนี้ได้ค่อนข้างดี รวมทั้งวินโดวส์ก็มีผู้ใช้อยู่มากมาย รวมทั้งทางด้านไมโครซอฟท์ประกาศตัวเกี่ยวกับเรื่องเว็บเซอร์วิสมาก่อนจาวา ในข้อนี้สรุปได้ว่าไมโครซอฟท์มีการทำการตลาดที่ดีกว่า

5.2.4 ภาษาที่รองรับ

จาวามีจุดเด่นที่เป็นภาษากลางที่สามารถนำไปใช้งานได้หลากหลายแพลตฟอร์ม

.NET มีจุดเด่นตรงที่รองรับได้หลายภาษาที่เป็นไปตามข้อกำหนดของ .NET เฟรมเวิร์ก โดยข้อดีของมันคือ ภายใต้โปรเจ็กเดียวกันสามารถพัฒนาได้จากหลายภาษา แต่ข้อเสียก็เยอะเช่นกัน เช่น ความเสี่ยงของระบบใหม่นี้ที่ทางไมโครซอฟท์พัฒนาขึ้นมา ของใหม่ย่อมมีความเสี่ยงเกิดขึ้น

- การดูแลรักษาระบบ เนื่องจากเขียนด้วยหลากหลายภาษา ถ้าบุคคลผู้เชี่ยวชาญในบางภาษาลาออกไปทำให้คนอื่นๆที่ดูแลลำบากในการทำความเข้าใจโค้ดภาษานั้นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การสร้างองค์ความรู้ การใช้งานหลากหลายภาษาส่งผลให้โปรแกรมเมอร์ไม่สามารถแชร์ประสบการณ์หลายๆอย่างในการพัฒนาแก่กันได้

ในการพัฒนาระบบนั้นก็ต้องพิจารณาให้เหมาะสม รวมถึงคำนึงถึงผลกระทบของการตัดสินใจด้วย

5.2.5 การสนับสนุนแพลตฟอร์มเดิม

J2EE ไม่มีปัญหาเรื่องการเชื่อมต่อกับระบบเดิมมากนัก แต่ตัว JCA และ เว็บเซอร์วิสก็ยังต้องการเขียนโค้ดใหม่เพิ่มเติม ถึงแม้ว่าไมโครซอฟท์ .NET ได้รับการพัฒนามาจาก MTS และ COM+ แต่อย่างไรก็ตามรูปแบบการพัฒนาก็เปลี่ยนไปโดยสิ้นเชิง มีคอมมอนรันไทม์(CLR)เข้ามาจัดการ มีการจัดการเกี่ยวกับคอมโพเน็นต์ที่เปลี่ยนไป ทำให้อาจต้องมีการเขียนโค้ดใหม่เกือบ 60% เพื่อให้สามารถใช้ประโยชน์จากรันไทม์เอนเวอรอนเมนต์ใหม่นี้ได้อย่างเต็มที่

5.2.6 Portability

J2EE เป็นมาตรฐานสำหรับหลายๆแพลตฟอร์ม และรองรับการอิมพลีเมนต์ได้หลากหลาย แต่ถึงอย่างไรก็ตาม ยังมีปัญหาเรื่องของการนำโค้ดไปใช้ใหม่กับต่างผลิตภัณฑ์ เนื่องจากแต่ละผลิตภัณฑ์ก็ยังมีการเพิ่มความสามารถในผลิตภัณฑ์ของตัวเองด้วยที่ไม่เป็นไปตามมาตรฐาน J2EE ทางซันเองก็มีซอฟแวร์ที่ให้การทดสอบว่าผลิตภัณฑ์เป็นไปตามมาตรฐาน J2EE ไหม

ถึงวันนี้ ตัวผลิตภัณฑ์ที่รองรับการใช้งาน .NET ก็มาจากไมโครซอฟท์แห่งเดียวจึงไม่มีปัญหาส่วนนี้ แต่มีการพัฒนาเฟรมเวิร์กบนลินุกซ์ด้วยแต่ยังไม่สมบูรณ์นักการพัฒนาคลาสไลบารี่ มีการทดสอบการรันโค้ดโปรแกรมง่ายข้ามจาก วินโดส์ไปยังลินุกซ์ซึ่งก็ได้ผลถูกต้อง

ถึงแม้ว่าไมโครซอฟท์จะรับปากถึงการทำให้ .NET สามารถใช้งานข้ามระบบได้ก็ตาม แต่จากที่ผ่านมาเช่น COM คอมโพเน็นต์ ไมโครซอฟท์ก็สามารถทำให้ใช้ได้บนระบบปฏิบัติการอื่นเช่นกัน แต่ไม่มีการพัฒนาเซอร์วิสอื่นๆขึ้นมารองรับการใช้งานเลย ดังนั้นในส่วนของ .NET เราก็ต้องติดตามกันต่อไปว่าจะเป็นอย่างไร

5.2.7 เครื่องมือที่ใช้ในการพัฒนา

จาวามีเครื่องมือให้เลือกใช้ค่อนข้างเยอะ เช่น Forte, Visual Cafe, Visaul Age, JBulider

ไมโครซอฟท์ ใช้เครื่องมือ Visual Studio.NET ในการพัฒนาซึ่งมีเอนเวอร์รอนเมนต์การใช้งานคล้ายกับ VisualStudio ตัวเดิมซึ่งโปรแกรมเมอร์คุ้นเคยอยู่แล้ว รวมทั้งเครื่องมือของไมโครซอฟท์เองก็มีประสิทธิภาพในการพัฒนาค่อนข้างมาก

5.2.8 การสนับสนุนการใช้งานเว็บเซอร์วิส

J2EE สนับสนุนการใช้งานเว็บเซอร์วิสโดยใช้ Java APIs สำหรับการแปลภาษา XML แต่อย่างไรก็ตามในส่วนนี้ยังอยู่ระหว่างการพัฒนา ยังต้องใช้ปรับแต่งเขียนโค้ดเพิ่มเติมด้วยตัวเองเพื่อให้เป็นเว็บเซ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อร์วิสได้ ผลิตภัณฑ์ของแต่ละแห่งก็มีการพัฒนาไลบรารี่ในส่วนนี้เพิ่มเติมขึ้นเองเพื่อให้การใช้งานเป็นไปอย่างอัตโนมัติและรวดเร็ว ซึ่งส่งผลต่อการผูกติดซอร์ซโค้ดกับระบบ

เฟรมเวิร์กของ .NET ถูกพัฒนามาเพื่อสนับสนุนเว็บเซอร์วิสอย่างเต็มตัวตั้งแต่แรกเริ่มอยู่แล้ว ไม่ว่าอย่างไรก็ตามมาตรฐานของเว็บเซอร์วิสยังต้องพัฒนาอีกยาวไกล เช่นเรื่องของการจัดการทรานแซกชัน เป็นต้นดังนั้นทั้งสองค่ายก็จะต้องเร่งออกผลิตภัฑ์มารองรับในส่วนนี้ด้วย

5.2.9 การขยายการรองรับการใช้งานของระบบ(Scalability)

การเพิ่มประสิทธิภาพของระบบโดยทั่วไปเราสามารถที่จะเพิ่มได้การเลือกใช้ฮาร์ดแวร์ใหม่ๆที่มีประสิทธิภาพ ทาง J2EE สามารถใช้งานได้หลากหลายแพลตฟอร์มซึ่งแพลตฟอร์มนั้นก็สามารถเลือกใช้โพรเซสเซอร์ได้หลายตัวด้วย

ทาง .NET นั้นใช้งานบนวินโดส์เท่านั้นจึงมีข้อจำกัดทางด้านการสนับสนุนโพรเซสเซอร์

5.2.10 สรุปการเลือกใช้งาน

ทั้งสองแฟล็ตฟอร์มสามารถนำไปสู่จุดมุ่งหมายที่เดียวกันได้ การที่จะเลือกใช้ผลิตภัณฑ์จากค่ายใดก็ต้องพิจารณาจากหลากหลายองค์ประกอบได้แก่ นักพัฒนาในองค์กรของคุณมีความถนัดระบบใด , ระบบเดิมที่ใช้งาน, ความสัมพันธ์กับบริษัทเวนเดอร์ , และก็ลูกค้าของบริษัทคุณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

Mobile Agent และ IBM Aglet

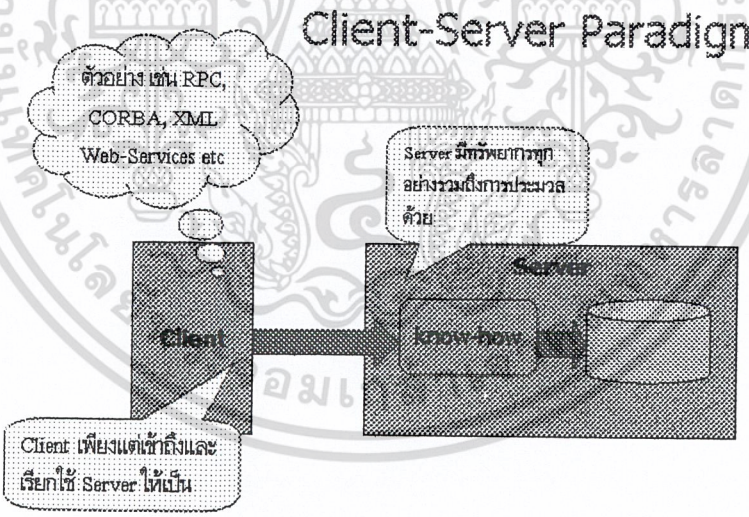
บทนี้จะแนะนำให้รู้จักโมบายเอเจนต์และ Aglet ที่เป็นโมบายเอเจนต์ที่สร้างด้วยภาษาจาวาโดยห้องวิจัยโตเกียวของไอบีเอ็มรวมไปถึงชีวิตของมัน

6.1 รูปแบบการทำงานบน Network

รูปแบบการทำงานบนเครือข่ายนั้นมีอยู่หลักๆด้วยกันอยู่ 3 แบบ คือ ไคลเอนต์-เซิร์ฟเวอร์, Code-on-Demand (Code-on-demand) และ โมบายเอเจนต์

6.1.1 ไคลเอนต์-เซิร์ฟเวอร์

จะใช้หลักการที่ ไคลเอนต์ ไม่ต้องประมวลผลอะไรมากมาย แค่รู้ว่าจะต้องใช้บริการอะไรเท่านั้น แต่หน้าที่ที่เหลือทั้งหมด ตั้งแต่ฐานข้อมูล ฟังก์ชันจัดการและหน่วยประมวลผล จะเป็นของ เซิร์ฟเวอร์

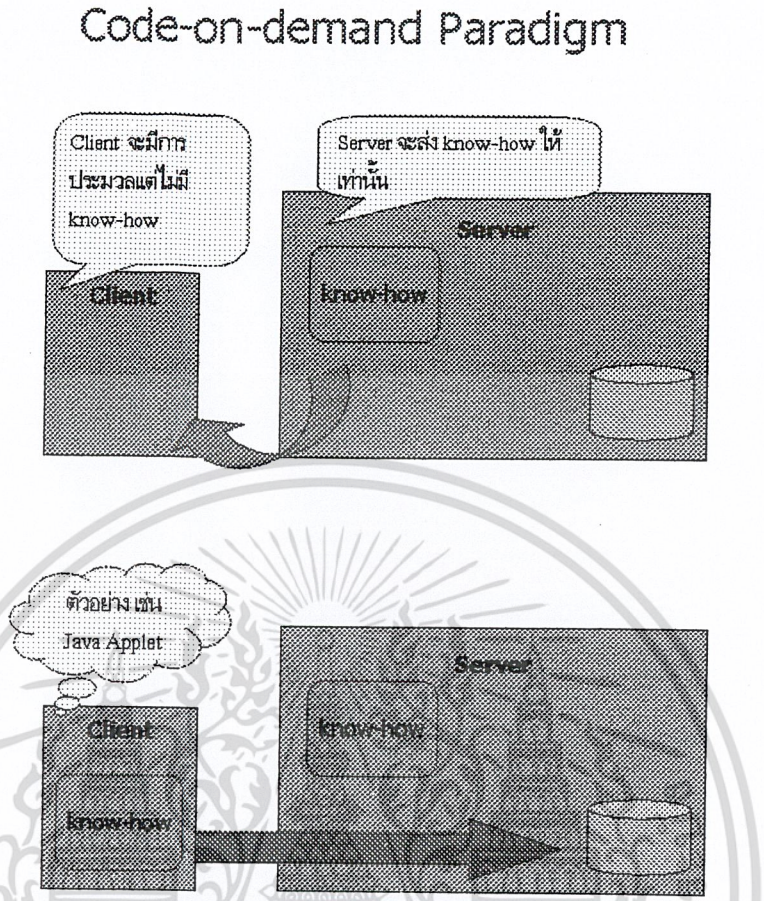


รูปที่ 6.1 รูปแบบไคลเอนต์-เซิร์ฟเวอร์

6.1.2 Code-on-Demand

ในรูปแบบนี้จะเป็นแบบที่ มีหน่วยประมวลผลที่ ไคลเอนต์ ทำให้ ไคลเอนต์ ไม่จำเป็นต้องมีการติดตั้งฟังก์ชันหรือโปรแกรมใดๆล่วงหน้า แต่จะโหลดมาขณะที่จะใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

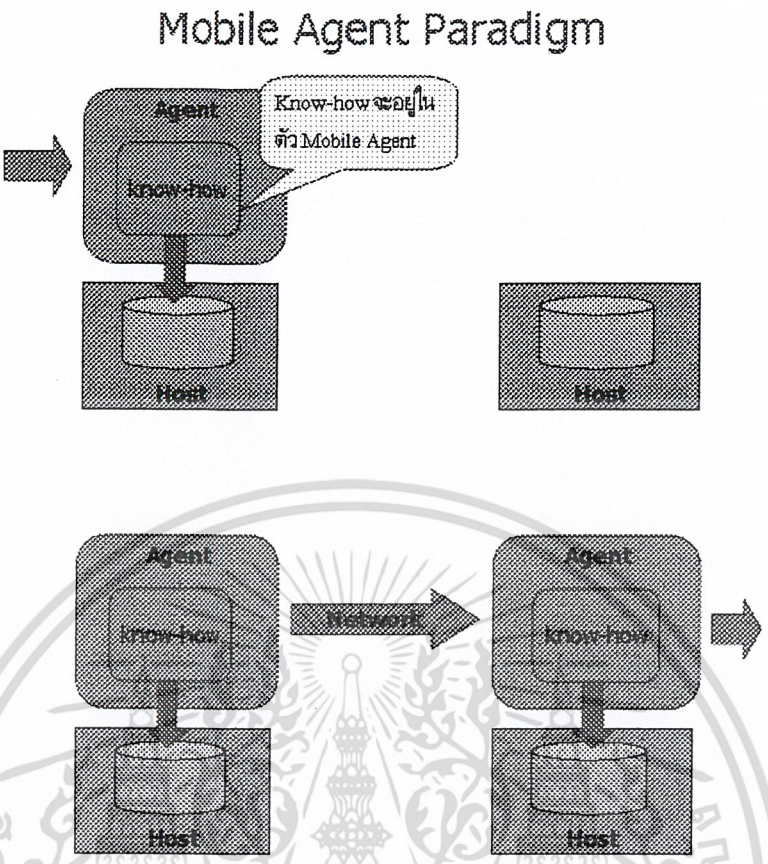


รูปที่ 6.2 รูปแบบCode-on-Demand

6.1.3 โมบายเอเจนต์

ในรูปแบบนี้จะเป็นรูปแบบที่อ่อนตัวที่สุดโดยตัวโมบายเอเจนต์ จะเป็นการผสมผสานของหน่วยประมวลผล, ฟังก์ชันการทำงาน และ ทรัพยากร ฟังก์ชันการทำงานจะไม่ได้ผูกอยู่ที่เครื่องๆเดียวแต่จะอยู่ทั่วไปในระบบเครือข่าย ไคลเอนต์ และ เซิร์ฟเวอร์ จะรวมกันเป็น Host, การทำงานแบบ Applet และ Servlet จะรวมกันและพัฒนาต่อมาเป็น โมบายเอเจนต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.3 รูปแบบโมบายเอเจนต์

6.2 โมบายเอเจนต์คืออะไร

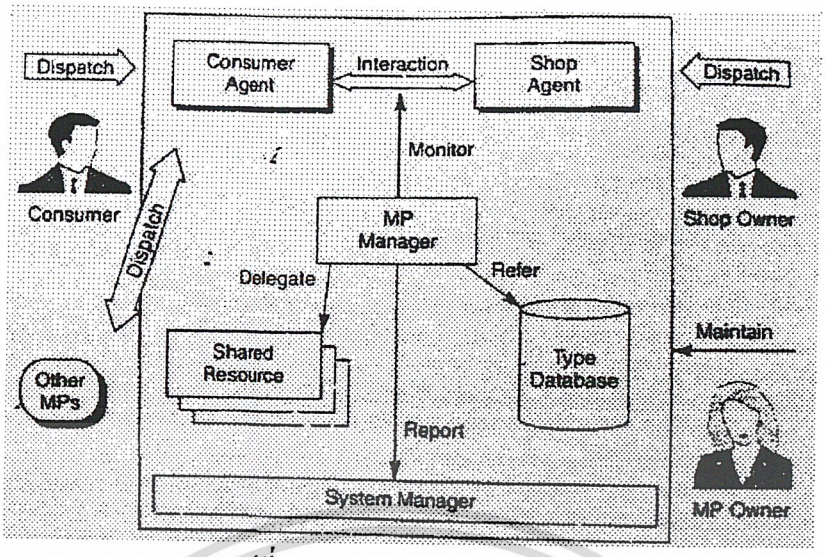
เอเจนต์ประจำที่ (Stationary Agent) คือโปรแกรมตัวแทนที่จะทำงานในเครื่องที่ถูกสั่งให้เริ่มทำงานเท่านั้น และหากมันต้องการที่จะสื่อสารกับโปรแกรมอื่นๆ มันจะทำการติดต่อผ่านกลไกที่เรียกว่า RPC (Remote Procedure Calling) แต่นั่นไม่ใช่กรณีของโมบายเอเจนต์ซึ่งสามารถเริ่มทำงานที่สภาวะของเครื่องๆหนึ่ง แต่มีอิสระที่จะเดินทางไปยังสภาวะของเครื่องอื่นๆพร้อมกับนำพาวัตถุใดๆไปทำงานต่อ

6.3 โมบายเอเจนต์ทำอะไรได้บ้าง

เหล่านี้คือตัวอย่างการนำโมบายเอเจนต์ไปประยุกต์ใช้

1. พาณิชย์อิเล็กทรอนิกส์ การใช้โมบายเอเจนต์เป็นตัวแทนคุยกัน แต่ละตัวก็จะมีหน้าที่แตกต่างกัน ในตลาด ตัวอย่างเช่น Tabican (www.tabican.ne.jp) แต่ใช้ได้เฉพาะในญี่ปุ่นเท่านั้น ซึ่งผู้บริโภค, เจ้าของร้าน และ เจ้าของตลาด จะมีตัวแทนเป็นของตัวเองดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.4 ตลาด Aglet Marketplace

2. ผู้ช่วยส่วนตัว เจ้าของก็จะรู้สึกดีที่ไม่ต้องเปิดคอมพิวเตอร์ทิ้งไว้ แต่ส่งตัวแทนไปเจรจาตกลงตารางนัดหมายแทนได้
3. นายหน้าที่ปลอดภัย แต่ละฝ่ายก็จะส่งตัวแทนไปเจรจา ณ โฮสลับที่ปลอดภัยที่ได้ตกลงกันไว้ก่อน โดยไม่ต้องกลัวว่าโฮสจะฝักใฝ่ฝ่ายใด
4. การดึงข้อมูลแบบกระจาย ข้อนี้เป็นตัวอย่างที่นิยมแสดงการใช้โมบายเอเจนต์โดยส่งโมบายเอเจนต์ไปยังโฮสเก็บข้อมูลปลายทางแล้วก็เครื่องต้นทางก็สามารถทำงานอย่างอื่นได้ เมื่อโมบายเอเจนต์ทำงานเสร็จก็เดินทางกลับมา
5. การบริการงานเครือข่ายโทรคมนาคม งานเช่นนี้เป็นงานที่มีโฮสจำนวนมาก และต้องมีการปรับแต่งทีละจำนวนมากๆเช่นกัน การใช้โมบายเอเจนต์จะทำให้ระบบนั้นอ่อนตัวและเกิดประสิทธิภาพ
6. งานที่มีการไหลของงานและงานที่เป็นกลุ่ม
7. งานเฝ้าสังเกต
8. งานกระจายข่าว โมบายเอเจนต์สามารถติดตั้ง อัพเดท จัดการโปรแกรมให้กับเครื่องลูกข่ายได้อย่างอัตโนมัติ
9. งานประมวลผลแบบขนาน โมบายเอเจนต์ สามารถโคลนตัวเองไปในเครือข่ายเพื่อจะทำงาน

6.4 ทำไมต้องใช้ โมบายเอเจนต์

1. ลดภาระการทำงานของเครือข่าย

ตัวอย่างการประมวลผลที่โมบายเอเจนต์จะแสดงให้เห็นประโยชน์อย่างมากที่สุดคือการที่ข้อมูลอยู่ตามที่ต่างๆและเชื่อมกันเป็นเครือข่าย การที่จะโหลดข้อมูลดิบมาเพื่อจะประมวลที่ใดที่หนึ่งนั้นจะเป็นการสิ้นเปลืองภาระงานเครือข่ายมาก แต่ถ้าใช้โมบายเอเจนต์จะเป็นการย้าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประมวลผลไปยังเครื่องต่างๆแล้วนำข้อมูลที่ประมวลได้กลับมาเท่านั้น คำคมง่ายๆคือ "move the computations to the data rather than the data to the computations"

2. ลดเวลาที่เสียไปในเครือข่าย

ในระบบที่ต้องการการประมวลแบบเวลาจริง การที่มีเวลาที่เสียไปในเครือข่ายเป็นสิ่งที่ยอมรับไม่ได้ Mobie Agent มีทางออกให้กับปัญหานี้ เพราะมันสามารถเคลื่อนที่จากศูนย์กลางควบคุมไปยังระบบปลายทางและสั่งการได้โดยตรง

3. เป็นโปรโตคอลแบบแคปซูล

ในการประมวลผลแบบกระจาย แต่ละเครื่องจะมีโปรแกรมเป็นของตัวเอง ในการที่จะจัดการข้อมูลเข้าและออก และเนื่องจากโปรโตคอลมันมีส่วนเกี่ยวข้องกับประสิทธิภาพและความปลอดภัย มันจึงเป็นการยุ่งยากที่จะจัดการปรับเปลี่ยนหรือเพิ่มการรักษาความปลอดภัย และแน่นอนมันก็จะเป็นมรดกปัญหาต่อไป ผิดกับโมบายเอเจนต์ที่สามารถเคลื่อนย้ายผ่านช่องทางที่สร้างขึ้นบนโปรโตคอลของตัวเอง

4. ทำงานได้แบบอะซิงโครนัสและอัตโนมัติ

ตัวอย่างเช่น อุปกรณ์ที่เคลื่อนที่ได้เช่นเครือข่ายไร้สายซึ่งอาจมีการ"หลุด"ได้ตลอดเวลา งานที่ต้องการการเชื่อมต่อตลอดเวลาจะทำงานไม่ได้หรือทำได้อย่างยากลำบาก ทางแก้โดยโมบายเอเจนต์คือการนำงานนั้นรวมไปกับโมบายเอเจนต์แล้วส่งไปทำงานแทน เมื่อทำงานเสร็จก็จะเชื่อมต่ออีกครั้งและเดินทางกลับ

5. มันปรับตัวได้ตลอดเวลา

เนื่องจากมันเป็นโปรแกรมที่เคลื่อนที่ได้ มันจึงต้องมีคุณลักษณะที่สามารถทำงานได้ในสภาวะที่แตกต่างกัน

6. มันทำงานได้ในสภาวะที่หลากหลาย

เครือข่ายคอมพิวเตอร์เป็นอะไรที่แตกต่างกันโดยพื้นฐาน บ่อยครั้งเนื่องมาจากฮาร์ดแวร์และซอฟต์แวร์ที่ต่างกัน แต่เนื่องจากโมบายเอเจนต์เป็นระบบที่ทำงานได้กับเครื่องคอมพิวเตอร์ทั่วไปและไม่อิงกับชั้นทรานสปอร์ตและอิงกับสภาวะแวดล้อมเท่านั้น ฉะนั้นมันจึงทำงานได้อย่างมีประสิทธิภาพโดยไร้รอยต่อ

7. มันทนทานต่อข้อผิดพลาด

ความสามารถในการจัดการกับปัญหาที่เกิดขึ้นโดยไม่คาดฝันของโมบายเอเจนต์ทำให้มันง่ายต่อการสร้างระบบประมวลผลแบบกระจายที่ทนทานต่อข้อผิดพลาดได้ ถ้าโฮสกำลังจะปิดโมบายเอเจนต์ที่กำลังทำงานจะได้รับการเตือนและย้ายการทำงานของตัวเองไปยังที่อื่นๆและทำงานต่อได้ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.5 ส่วนประกอบของ โมบายเอเจนต์

6.5.1 Agent

ส่วนนี้จะประกอบด้วย 5 ตัวแปรคือสถานะ(state), การทำงาน(implementation), อินเตอร์เฟซ (interface), ID (identifier) และหัวหน้า (principal)

- สถานะ

  เมื่อโมบายเอเจนต์เคลื่อนที่และจะเริ่มการทำงานต่อ มันจะต้องรู้ว่าทำอะไรต่อ จึงต้องมีการจดจำสถานะตัวเองไว้

- การทำงาน

  ส่วนนี้ก็เหมือนกับเป็นโค้ดโปรแกรมโมบายเอเจนต์จำเป็นต้องมีส่วนนี้เพื่อจะทำงานต่อ การทำงานนี้ต้องทำให้โมบายเอเจนต์ทำงานที่โฮสได้และต้องปลอดภัยต่อโฮสที่จะทำงานด้วย

- อินเตอร์เฟซ

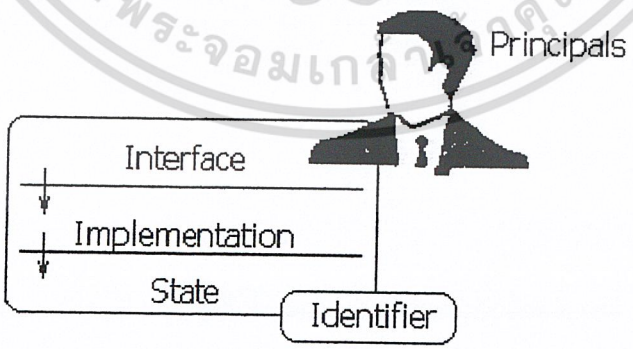
  เป็นส่วนที่ให้โมบายเอเจนต์ยอมรับการติดต่อจากโมบายเอเจนต์อื่นๆหรือจากระบบใดๆ

- ID

  รหัส ID นี้จะไม่ซ้ำกันอย่างน้อยก็ในช่วงชีวิตมัน ID นั้นเกิดจากการรวมระหว่างหัวหน้าและเลขอนุกรมของมัน เนื่องจากมันจะไม่ซ้ำกันและไม่เปลี่ยนแปลง มันจึงใช้ในการกำหนดตัวโมบายเอเจนต์แบบเจาะจงได้

- หัวหน้า

  อาจจะเป็นคนๆหนึ่ง หรือหน่วยงานหนึ่ง หรือบริษัทหนึ่งๆ แต่อย่างน้อยจะมี 2 อย่างคือ ผู้สร้าง(บอกว่าใครเป็นคนให้ได้คมา)และเจ้าของ(บอกว่าใครเป็นผู้สั่งสร้าง ซึ่งจะเป็นคนที่มีความรับผิดชอบทางกฎหมายและศีลธรรม)



รูปที่ 6.5 คุณสมบัติ Agent

6.5.2 Place

คือสภาวะแวดล้อมที่เอเจนต์ทำงาน ประกอบด้วย 4 สิ่งคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. Engine

   เป็นม้างานและเป็นจักรกลเสมือนสำหรับ Place หนึ่งหรือหลายๆอัน

2. Resources

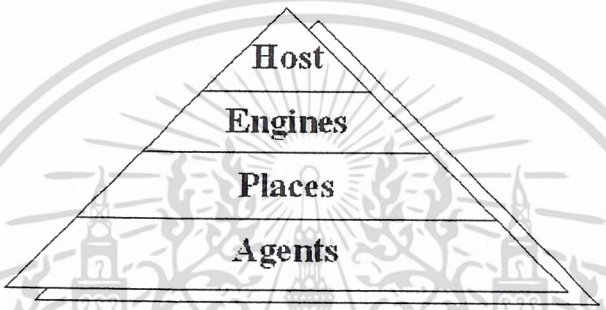
   เช่นพวกฐานข้อมูล, หน่วยประมวลผล และ บริการอื่นๆที่โฮสมีให้

3. Location

   คือที่อยู่เครือข่าย

4. Principals

   คือผู้รับผิดชอบตามกฎหมายสำหรับการทำงานของ Place



รูปที่ 6.6 Agent Model Pyramid

6.5.3 Agent Transport Protocol (ATP)

ATP เป็นโปรโตคอลระดับแอพลิเคชั่นที่ทำงานเป็นอุโมงค์บน HTTP โปรโตคอลนี้ถูกออกแบบเพื่อจะเคลื่อนย้ายเอเจนต์ได้อย่างเสรีไม่อิงกับระบบของเอเจนต์การร้องขอของโปรโตคอลนี้ก็จะมีการร้องขอ, ส่วนหัวและเนื้อหา

การร้องขอมี 4 วิธีการ คือ

1. Dispatch

   เป็นการให้ระบบเอเจนต์ปลายทางสร้างเอเจนต์ใหม่จากข้อมูลที่ผู้ร้องขอส่งไปและเริ่มทำงาน agent ถ้าการร้องขอสำเร็จผล ผู้ร้องขอจะทำลาย agent ตัวเองและคืนทรัพยากรที่ agent ตัวนั้นจับจองอยู่

2. Retract

   เป็นการดึงเอาเอเจนต์ที่อยู่ปลายทางกลับมายังผู้ส่งคำสั่ง

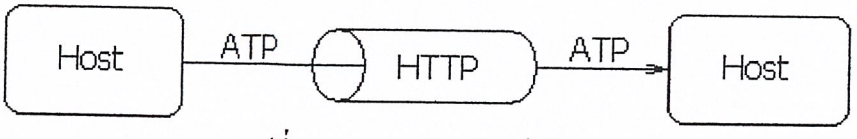
3. Fetch

   เหมือนกับ GET ของ HTTP มันจะให้ตัวรับหาและส่งข้อมูล ID มาให้

4. Message

   เป็นการส่งผ่านข้อความไปยังตัวเอเจนต์ที่เจาะจงโดยดูจาก ID และคืนค่าคำตอบให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.7 HTTP เป็นอุโมงค์ให้ ATP

6.6 กายวิภาคของ โมบายเอเจนต์

6.6.1 การเกิด

เอเจนต์ต้องถูกสร้างที่ Place การเกิดนี้อาจมี agent ตัวอื่นอยู่ก่อนแล้วหรืออาจไม่มีก็ได้ การเกิดนั้นเกี่ยวข้องกับขั้นตอน 3 ขั้นตอน

1. การสร้างวัตถุ โค้ดจะต้องถูกโหลดโดยมีทั้งอินเตอร์เฟซและอิมพลีเมนต์ แล้ว Place จะกำหนด ID ให้
2. การตั้งค่าเริ่มต้น agent นั้นยินยอมรับค่าที่ผู้สั่งป้อนให้ในขณะเริ่มสร้างตัวเอเจนต์
3. การทำงานแบบอัตโนมัติ เมื่อเอเจนต์ถูกสร้างและอยู่ที่ place แล้วมันจะสามารถทำงานได้อย่างอัตโนมัติโดยไม่เกี่ยวกับ เอเจนต์ตัวอื่นๆ

6.6.2 การตาย

การตายของเอเจนต์มีสาเหตุจากสิ่งเหล่านี้

1. เอเจนต์หมดอายุ
2. เอเจนต์ไม่ถูกอ้างอิง
3. เอเจนต์ละเมิดกฎความปลอดภัย
4. ระบบถูกปิด

การตายเกี่ยวข้องกับขั้นตอน 2 ขั้น คือ

1. การเตรียมที่จะตายเอเจนต์จะทำงานสุดท้ายก่อนตายได้
2. การหยุดการทำงาน place จะยุติการทำงานของเอเจนต์

6.6.3 การย้าย

เมื่อเอเจนต์ได้รับการร้องขอให้ย้าย place ต้นทางจะทำการส่ง และ ปลายทางจะทำการรับ

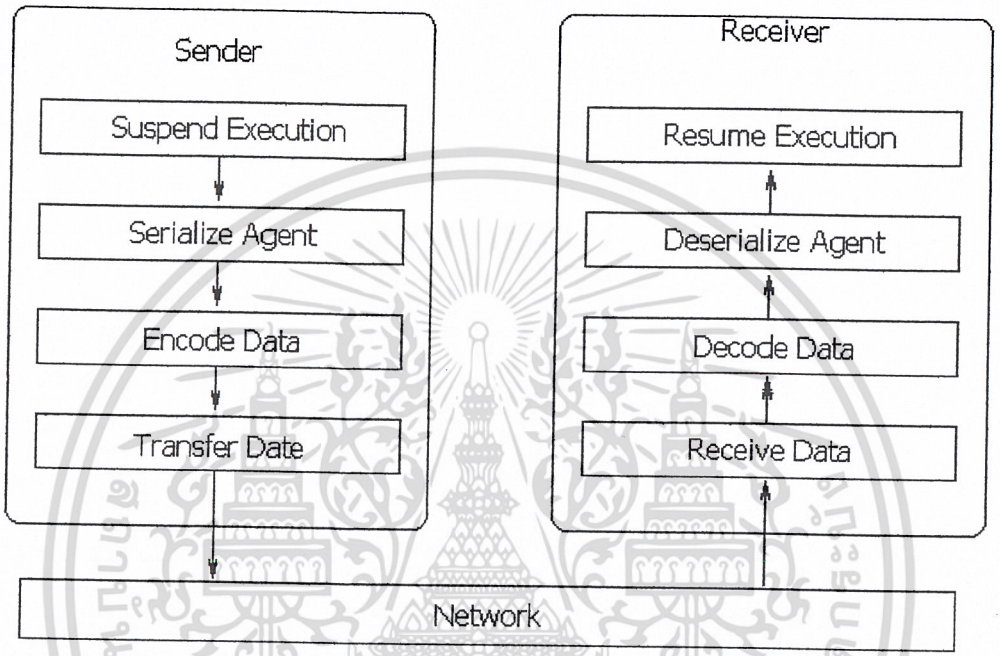
1. การส่ง มันจะทำตามขั้นตอนดังนี้

- หยุดการทำงาน agent ก่อนที่จะหยุดมันจะทำงานที่ค้างอยู่ให้เสร็จแล้วเอเจนต์จะหยุดการทำงาน
- อนุกรมตัวเอเจนต์เพื่อจะส่งไปบนเครือข่ายได้
- เข้ารหัสเพื่อการส่งเอเจนต์ที่อนุกรมแล้ว
- จะมีการสถาปนาการเชื่อมต่อเพื่อส่งเอเจนต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การรับ มันจะทำตามขั้นตอนดังนี้

- เมื่อปลายทางยอมรับการย้าย ก็จะได้รับ agent ที่เข้ารหัส
- ถอดรหัสเอเจนต์ที่เข้ารหัสให้กลายเป็นสายอนุกรม
- ทำการอนุกรมกลับด้านเพื่อให้กลายเป็นวัตถุและคืนค่าที่เอเจนต์เก็บไว้
- เอเจนต์ทำงานต่อ



รูปที่ 6.8 การเคลื่อนย้ายเอเจนต์

6.7 IBM Aglet

Aglet(TM) เป็นโมบายเอเจนต์แบบหนึ่งซึ่งผลิตภัณฑ์ของไอบีเอ็ม สร้างขึ้นด้วยภาษาจาวา และเป็นแบบเปิด เหตุผลที่ใช้ภาษาจาวาในการสร้างโมบายเอเจนต์ตัวนี้เพราะเหตุผลเหล่านี้

1. ภาษาจาวาไม่ขึ้นกับระบบปฏิบัติการ
2. เป็นภาษาที่มีความปลอดภัย จาวาไม่อนุญาตให้เข้าถึงหน่วยความจำได้โดยตรง
3. มีกลไกที่นำคลาสมาสร้างวัตถุได้ในขณะทำงาน
4. ภาษาจาวารองรับการทำงานแบบหลายงาน (Multi-Thread)
5. ภาษาจาวามีการทำอนุกรมวัตถุได้โดยเมื่อคืนสภาพก็ยังมีค่าที่เก็บไว้อยู่ด้วย
6. ไบนารี่โค้ด ของคลาสภาษาจาวามีข้อมูลเกี่ยวกับคลาสและเมธอด(Method)ทั้งหมด

แต่ก็มีข้อเสียเช่นกันคือ

1. ภาษาจาวาเองไม่มีการควบคุมการใช้ทรัพยากรของวัตถุ ทำให้วัตถุสามารถจองทรัพยากรได้อย่างไม่มีจำกัด อาจเกิดปัญหา Denial Of Service ได้

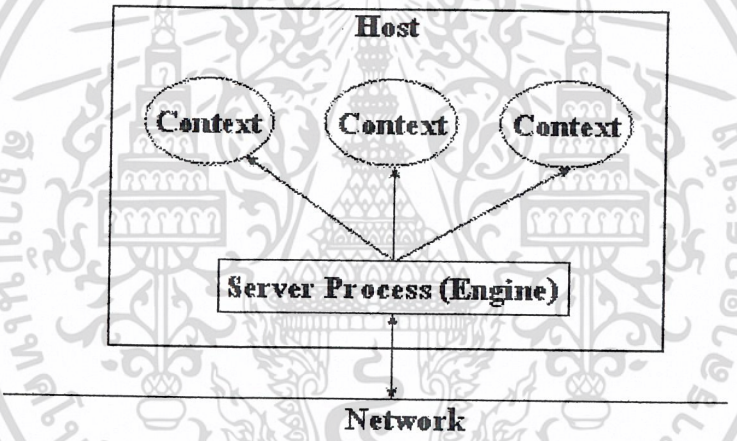
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ไม่มีการป้องกันการอ้างถึง การที่ประกาศเมธอดในจาวาเป็น public จะทำให้ใครก็สามารถเรียกใช้ได้ วิธีการแก้ปัญหานี้ของ Aglet ก็คือ การใช้พร็อกซี่
3. ภาษาจาวาไม่รองรับการเก็บและคืนสภาพ ปัจจุบัน การเก็บสถานะนั้นใช้การเชื่อตัวแปรภายในเท่านั้น

6.7.1 แบบจำลอง Aglet

แบบจำลองของ Aglet นั้นประกอบด้วย 3 สิ่งคือ

1. ตัว Aglet เอง เป็นวัตถุของจาวา
2. พร็อกซี่ (Proxy) เป็นตัวแทนในการติดต่อกับตัว Aglet ทำให้เราไม่ต้องรู้ว่า Aglet จริงๆอยู่ที่ใด Proxy มีทั้งแบบท้องถิ่น (Local Proxy) และแบบห่างไกล (Remote Proxy)
3. คอนเท็กซ์ (Context) เป็นพื้นที่ทำงาน ซึ่งก็คือ place นั่นเอง



รูปที่ 6.9 ความเกี่ยวข้องระหว่างโฮส, เซิร์ฟเวอร์ และคอนเท็กซ์

6.7.2 Aglet Package

Aglet อย่างน้อยต้องมีคลาสที่เกี่ยวข้องคือ Aglet, AgletProxy, AgletContext, Message, FutureReply และ AgletID

1. **Aglet** เป็นคลาสกุญแจและเป็นคลาสนามธรรมสำหรับสร้างตัว Aglet เราสามารถนำคลาสนี้มาสืบทอดและเพิ่มการทำงานเข้าไป ซึ่งมันจะเกี่ยวกับคลาส AgletInfo ซึ่งใช้เก็บข้อมูลพื้นฐานของ Aglet เช่นเวลาเกิด ที่ที่โหลดคลาส (Code Base) เวลาถึง ที่อยู่ปัจจุบันของมัน ตัวอย่างโค้ดที่ง่ายๆ ของ Aglet คือ

```
public class MyFirstAglet extends com.ibm.aglet.Aglet{
        public void onCreation(Object init){
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            // ใส่คำสั่งต่างๆที่จะให้ทำงานเมื่อ Aglet เกิด เช่น
            dispatch(new URL("atp://some.host.com/context"));
            // เป็นการให้ย้ายตัว Aglet ไปยังคอนเท็กซ์นั้น
        }
        public void run(){
            // ใส่คำสั่งต่างๆที่จะให้ทำงานเมื่อ Aglet เริ่มทำงาน
        }
}
```

2. **AgletProxy** เป็นคลาสที่ทำหน้าที่ติดต่อกับ Aglet เมื่อถูกเรียกใช้งาน AgletProxy จะปรึกษากับ Security Manager เพื่อจะดูว่าสิ่งที่ติดต่อเข้ามามีสิทธิ์ในการดำเนินการหรือไม่
3. **AgletContext** เนื่องจาก place ก็คือคอนเท็กซ์ Aglet จะใช้ชีวิตส่วนใหญ่อยู่ที่นี่ เกิดและตายที่นี่ด้วย เราจะใช้คลาสนี้เพื่อจะเอาข้อมูลเกี่ยวกับสภาพแวดล้อม รวมทั้ง Aglet ที่อยู่ในที่ๆนี้ด้วย

```
context = getAgletContext();
// เป็นการยึดเอาคอนเท็กซ์ปัจจุบัน
context.createAglet(...);
// เป็นการสร้าง Aglet
context.retractAglet(remoteContextURL, agletID);
// เป็นการดึง Aglet จาก เครื่องอื่นโดยต้องอ้างID ด้วย
Enumeration proxies = context.getAgletProxies();
// เป็นการขอพร็อกซี่ทั้งหมด
```

4. **Message** คลาสนี้หมายถึงข้อความที่ส่งไปติดต่อกับ Aglet โดยมีตัวแปรที่เรียกว่า "kind" เพื่อจะใช้แบ่งประเภทข้อความ และมีค่าของ "kind" นั้นๆ ติดไปด้วยได้

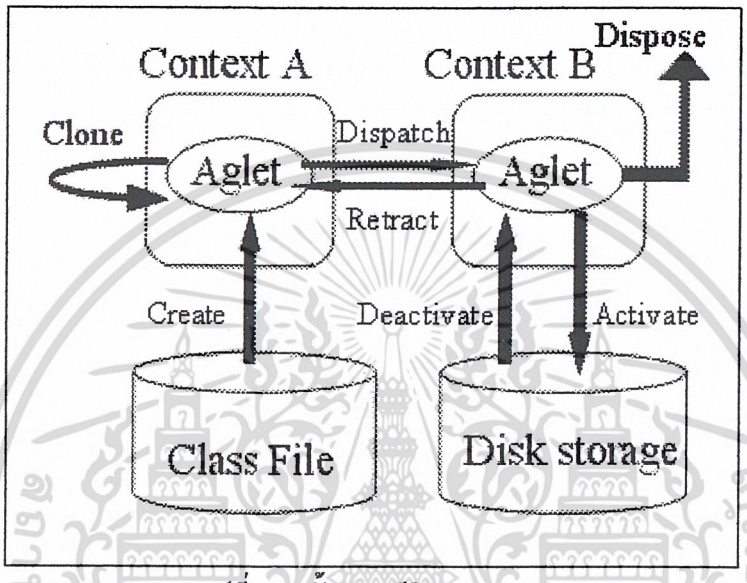
```
Message myName = new Message("my name", "Jacob");
Message yourName = new Message("your name?");
```

5. **FutureReply** คลาสนี้เป็นคลาสที่ส่งข้อความแบบไม่ต้องรอคำตอบ ซึ่งเรื่อง **Message** จะกล่าวถึงรายละเอียดในหัวข้อ 7.1
6. **AgletID** เป็นคลาสที่เก็บข้อมูลเกี่ยวกับ Aglet ที่เป็นแบบเฉพาะเจาะจงตัว มีข้อมูลเช่น ID, เวลาเกิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

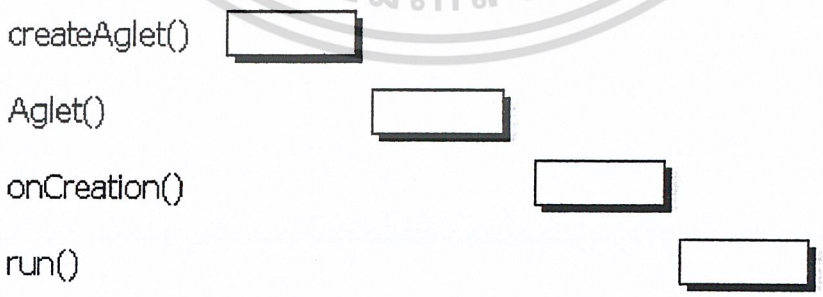
```
AgletID aid = proxy.getAgletID();
proxy = context.getAgletProxy(aid);
```

6.8 วงจรชีวิตของ Aglet



รูปที่ 6.10 ขั้นตอนชีวิตของ Aglet

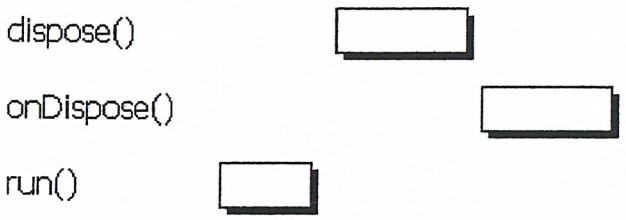
6.8.1 Create จะเป็นการโหลดคลาสมาจากที่ที่โหลดคลาส (*.class) ซึ่งอาจจะเป็นเครื่องท้องถิ่นหรือเครื่องที่ห่างไกลผ่าน ATP หรือ HTTP โปรโตคอลก็ได้ เมื่อ Aglet ถูกสร้าง โดยเมธอด createAglet() แล้วเมธอด Aglet() ซึ่งเป็น Constructor จะถูกเรียก แล้วเมธอด onCreation(Object init) ซึ่งทั้ง 3 เมธอดนี้จะเรียกครั้งเดียวตอนเกิด แล้วเมธอด run() จะถูกเรียกขึ้นมาเป็นอันดับสุดท้าย



รูปที่ 6.11 เมธอดที่ถูกเรียกใช้เมื่อมีการเกิด

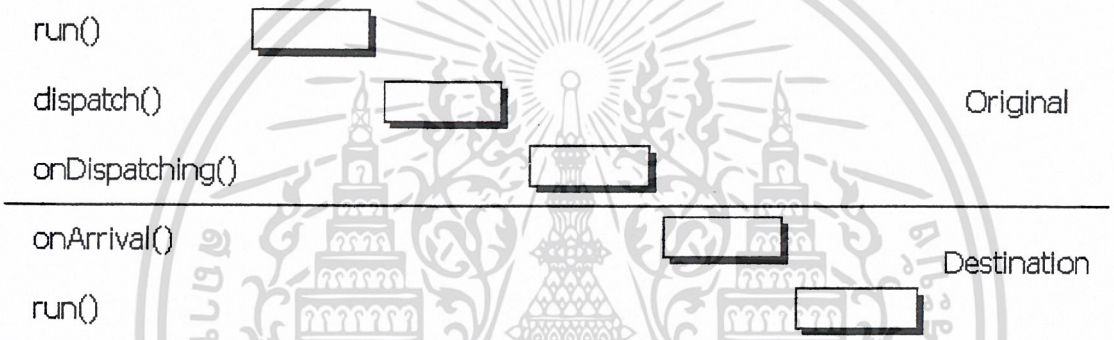
6.8.2 Dispose จะเป็นการตายของ Aglet โดยเมธอด dispose() ซึ่งเป็นเมธอด final ที่ overide ไม่ได้อีก เมื่อถูกเรียก มันจะไปเรียกเมธอด onDispose() ซึ่งเราเขียนเองให้ไปจัดการอะไรก็ได้ก่อนตายก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



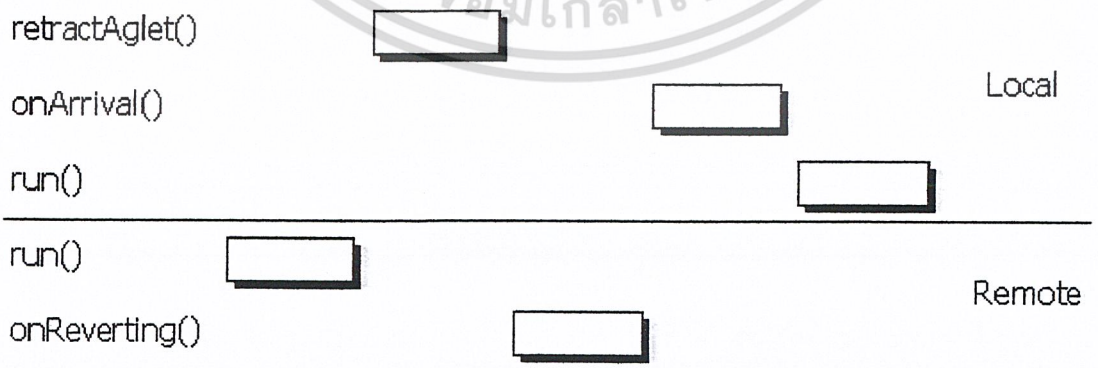
รูปที่ 6.12 เมธอดที่ถูกเรียกใช้เมื่อมีการตาย

6.8.3 Dispatch จะเป็นการย้าย โดยเมื่อเมธอด dispatch() ซึ่งเป็น final ถูกเรียก มันจะไปเรียกเมธอด onDispatching() ซึ่งจะทำงานที่ฝั่งต้นทาง และ onArrival() ซึ่งจะทำงานที่ฝั่งปลายทาง แล้วก็ไปเรียกเมธอด run() ต่อไป



รูปที่ 6.13 เมธอดที่ถูกเรียกใช้เมื่อมีการย้าย

6.8.4 Retract เป็นการดึง Aglet กลับ โดยการเรียกเมธอด retractAglet() Aglet นั้นที่กำลังรันเมธอด run() ก็จะหยุดหลังจากนั้นเมธอด onReverting() ของ Aglet ตัวนั้นจะทำงาน ตามด้วยเมธอด onArrival() แล้วตามด้วยเมธอด run()



รูปที่ 6.14 เมธอดที่ถูกเรียกใช้เมื่อมีการเรียกกลับ

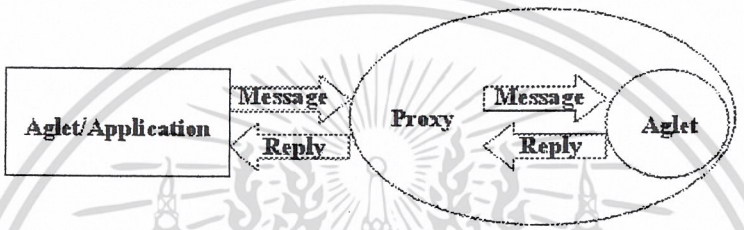
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
เราจะใช้ Aglet อย่างไร

7.1 การติดต่อกับ Aglet

7.1.1 การส่ง Message

การที่เราจะติดต่อกับ Aglet เราจะใช้วิธีส่ง Message ให้กับ Aglet แต่มันไม่ได้รับโดยตรง แต่ Aglet มันจะรับโดยผ่าน Proxy



รูปที่ 7.1 ข้อความระหว่างผ่านพร็อกซี่

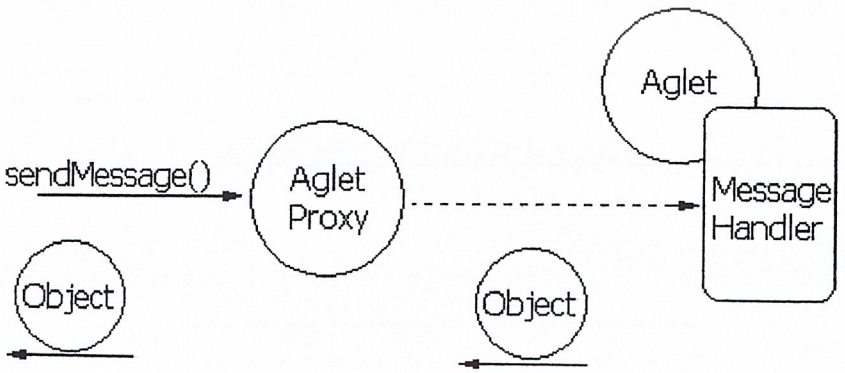
7.1.2 ประเภทของ Message

ข้อความที่ส่งให้ Aglet มีอยู่ 3 ประเภท

1. Now-Type Messaging

   ข้อความประเภทนี้จะเป็นแบบ Synchronous Messaging คือ ส่งไปแล้วรอรับคำตอบกลับ ระหว่างนี้ผู้ส่งข้อความจะหยุดการทำงานแล้วรอดังรูปที่ 7.2 (ซ้าย) ใน Aglet จะใช้คำสั่งว่า

```
proxy.sendMessage(msg);
```



รูปที่ 7.2 Synchronous Messaging

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

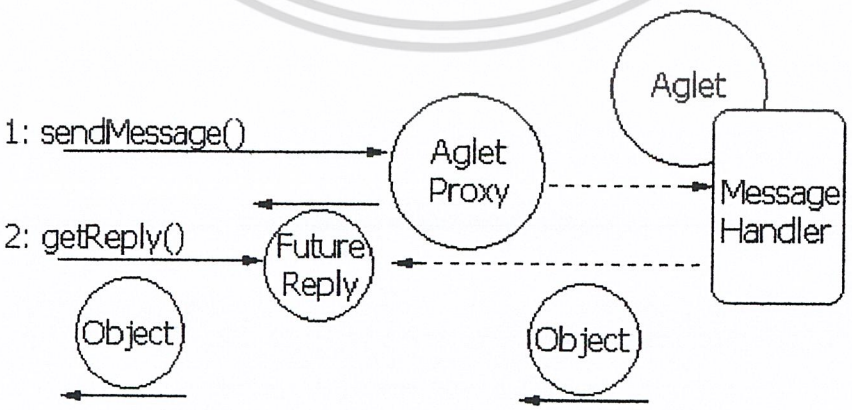
ฝั่ง Aglet ตัวรับก็จะใช้ Message Handler เป็นตัวจัดการ จะใช้เมธอด handleMessage() เพิ่มลงไป

```
public boolean handleMessage(Message msg){
   if (msg.samKind("Hello")){
            //Respond to sender
            return true;
   }else
            return false;
}
```

2. Future-Type Messaging

ข้อความประเภทนี้จะเป็นแบบ Asynchronous Messaging คือ ส่งไปแล้วไม่ต้องรอรับคำตอบกลับ แต่ถ้ามีก็ได้ ระหว่างนี้ผู้ส่งข้อความจะทำงานอย่างอื่นได้เลย ดังรูปที่ 7.2 (กลาง) ใน Aglet จะใช้คำสั่งว่า

```
FutureReply future=proxy.sendFutureMessage(msg);
Future.waitForReply(4000);
if (future.isAvailable()){
   String reply = (String) future.getReply();
}
else{
   //Time out...forget reply
}
```

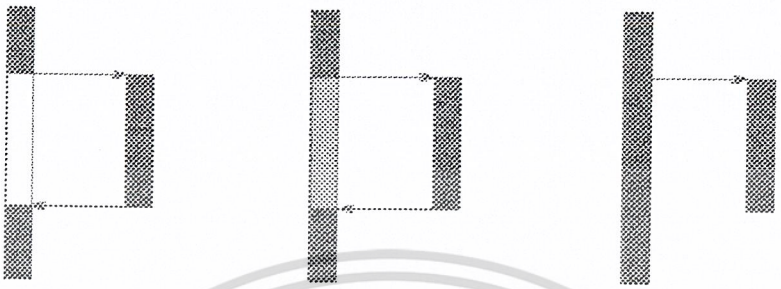


รูปที่ 7.3 Asynchronous Messaging

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. One-Way-Type Messaging

ข้อความประเภทนี้เป็น Asynchronous Messaging คือ เป็นแบบส่งแล้วไม่ต้องรอคำตอบกลับเช่นกัน เหมือนเป็นการบอกข่าวให้กับ Aglet เฉยๆ



รูปที่ 7.4 ประเภทของข้อความจากซ้ายไปขวา

Now-Type, Future-Type และ One-Way-Type

7.2 รูปแบบการทำงานของโมบายเอเจนต์

รูปแบบการทำงานของโมบายเอเจนต์ก็มีอยู่ 3 แบบด้วยกัน คือ ท่องเที่ยว, จัดการงาน และ พบปะกัน

7.2.1 ท่องเที่ยว

เป็นรูปแบบที่เอเจนต์ตัวเดียวเดินทางไปยังโฮสต่างๆ มี 3 แบบ คือ

1. Itinerary รูปแบบนี้เอเจนต์จะเดินทางไปทำงานที่โฮสต่างๆตามที่กำหนดเส้นทางไว้
2. Forwarding รูปแบบนี้คือที่โฮสทำการส่งต่อเอเจนต์ไปยังโฮสอื่นๆโดยไม่ต้องทำงานที่โฮสนั้น
3. Ticket รูปแบบนี้จะทำให้เอเจนต์รู้จักมีมีสิทธิ์ใน Host ต่างๆก่อน แล้ว Aglet จะเป็นตัวที่ตัดสินใจจากข้อมูลในตัวเองว่าจะไปที่ไหน

7.2.2 จัดการงาน

เป็นรูปแบบที่เอเจนต์จัดการงาน 2 แบบคือ

1. Master-Slave รูปแบบนี้เจ้านายจะแบ่งงานเป็นส่วนเล็กๆให้ทาสจำนวนมากช่วยกันทำ
2. Plan รูปแบบนี้จะซับซ้อนขึ้น คือให้เอเจนต์ที่แตกต่างกันทำงานหลายงานที่โฮสต่างกันหลายโฮสตามการไหลของงาน

7.2.3 พบปะกัน

1. Meeting เป็นรูปแบบที่เอเจนต์ตั้งแต่ 2 ตัวขึ้นไปพบกันตรงที่โฮสหนึ่งๆแล้วสื่อสารกัน
2. Locker รูปแบบนี้จะมีโฮสหนึ่งที่เอเจนต์ทุกตัวที่ผ่านมามาทิ้งข้อมูลไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. Messenger รูปแบบนี้จะให้เอเจนต์ทำตัวเป็นบุรุษไปรษณีย์
4. Finder รูปแบบนี้จะเป็นรูปแบบที่ให้เอเจนต์ตัวหนึ่งเก็บข้อมูลและตั้งชื่อข้อมูล หลังจากนั้นเราจะหาข้อมูลก็จะไปถามจากเอเจนต์ตัวนั้นแทน
5. Organized Group จะมีเอเจนต์หลายๆตัวที่ต่างกันรวมเป็นกลุ่มและเดินทางไปด้วยกัน



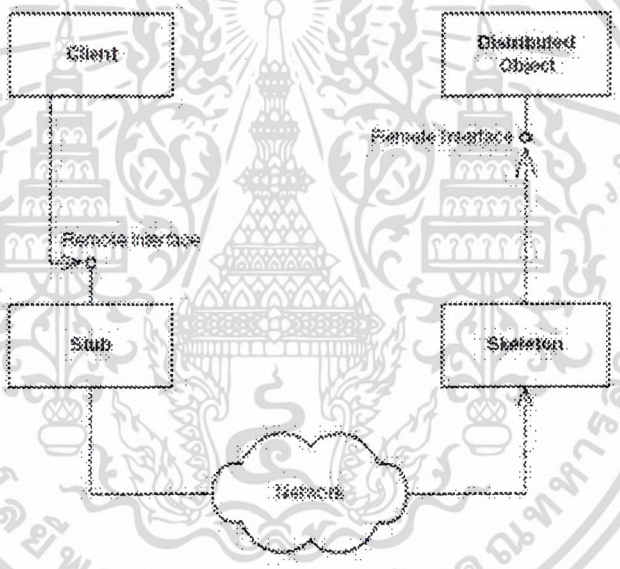
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
การรวม Aglet เข้ากับ Application อื่นๆ

8.1 RMI

RMI: Remote Method Invocation คือการเรียกใช้คอมโปเน้นท์ข้ามเครื่องในระบบกระจายคอมโปเน้นท์ เป็นการทำให้โปรแกรมภาษาจาวาสามารถสร้างวัตถุที่ไม่ได้อยู่ที่เครื่องท้องถิ่นได้ แต่วัตถุนั้นจะอยู่ที่เครื่องที่อยู่ระยะไกล และใช้งานได้เหมือนกับเป็นวัตถุในเครื่องท้องถิ่น

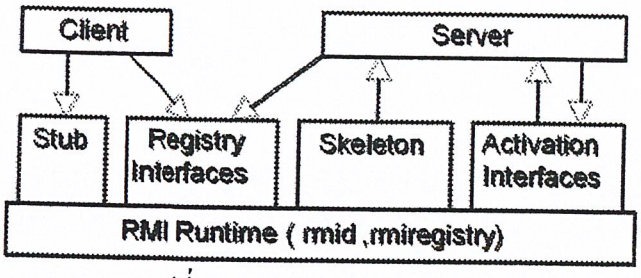
หลักการของ Distributed Object



รูปที่ 8.1 ความคิดรวบยอดของ Distributed Object

จากรูปที่ 8.1 แสดงการใช้งาน ของ Distributed Object จะพบว่า ในส่วนของไคลเอนท์นั้นจะมี Stub ซึ่งก็เป็นเหมือนคลาสฝั่งไคลเอนท์และในส่วนของ Distributed Object จะมี Skeleton ก็เหมือนกับเป็นคลาสฝั่งเซิร์ฟเวอร์ ซึ่งในส่วนของ Stub จะมี Remote Interfaces ที่เหมือนกับทางด้าน Distributed Object ด้วยวิธีนี้ ทำให้ไคลเอนท์ไม่รับรู้ว่าได้เรียกใช้วัตถุที่ต่างเครื่องกันอยู่ ซึ่งสามารถทำได้โดยใช้มาตรฐานต่างๆ ได้มากมาย เช่น RMI/IIOP, CORBA หรือ DCOM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.2 สถาปัตยกรรมของ RMI

การสร้าง RMI ประกอบด้วย 4 ขั้นตอน

1. สร้าง Interface ของ Remote Object. ส่วนนี้จะเหมือนกับเป็นโครงร่างของคลาส ใช้เพื่อให้ไคล์เอนท์ได้รู้ว่าคลาสชื่ออะไรมีเมธอดอะไรเรียกใช้และคืนค่าอะไร ต้องเป็นการสืบทอดมาจาก java.rmi.Remote และ throws RemoteException ดังตัวอย่าง

```
public interface AgletRMIServer extends Remote
{
  public String showAll() throws RemoteException;
  public boolean updateStock(String retailShopCode, int id, int quantity) throws
RemoteException;
}
```

2. สร้าง Implementation ของ Remote Object. ส่วนนี้จะเหมือนการเขียนโปรแกรมธรรมดาแต่จะบอกว่าเป็น implements ของคลาส interface ในข้อ 1 และสืบทอดจาก java.rmi.server.UnicastRemoteObject

```
public class AgletRMIServerImpl
    extends UnicastRemoteObject
    implements AgletRMIServer {
    static ServerApp app;
//คอนสตรัคเตอร์
  AgletRMIServerImpl() throws RemoteException {
    super();
  }
//เมธอด returnAgletResult ใช้เป็นการภายในจึงไม่ได้ประกาศในอินเตอร์เฟซ
  public static Object returnAgletResult(String name,Object init){
    try{
      if(name.equals(""))return null;  //ถ้าไม่มีชื่อเอเจ้นต์ไม่ต้องทำอะไรต่อ
      AgletProxy proxy=null;
      //สั่งให้ เซิร์ฟเวอร์สร้างเอเจ้นต์ที่มีโค้ดเบส,ชื่อเอเจ้นต์ และค่าเริ่มต้น
      //และเก็บพร็อกซี่ของเอเจ้นต์ที่สร้างไว้ด้วย
      //ดูรายละเอียดเกี่ยวกับเมธอด startAglet ที่ภาคผนวก ก
      proxy=app.startAglet(new URL("atp://127.0.0.1:4000"),name,init);
      Object reply=null;
      short count=0;
      //ส่งข้อความถามเอเจ้นต์ตัวที่สร้างไว้เป็นระยะๆ จนกว่าจะได้คำตอบ
      while(reply==null){
        try{
          reply = proxy.sendMessage(new Message(
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            "God Father wanna know"));
        }catch(Exception e){out(""+e);}
        //หากไม่ได้คำตอบภายใน30ครั้งที่ถามก็ให้เลิก
        if(count++>30){out("Time out");break;}
        Thread.sleep(500);
      }
      proxy.dispose(); //ทำลายเอเจ้นต์ตัวที่สร้างป้องกันทรัพยากรล้นระบบล่ม
      return reply;
    }
    catch(Exception e){out(""+e);}
    return null;
  }
//อิมพลีเมนต์ของเมธอด showAll()
  public String showAll(){
    try{
      //เป็นการเรียกเมธอดเพื่อสร้างเอเจ้นต์ดึงข้อมูลและขอคำตอบกลับ
      String temp=(String) (new AgletRMIServerImpl().returnAgletResult("FetchDBParent",
null));
      return temp.trim();
    }catch(Exception e){out(""+e);}
    return "";
  }
  public boolean updateStock(String retailShopCode, int id, int quantity){
    try{
      if(!retailShopCode.equals("OlalaMovie"))return false;
      Object[] init=new Object[]{
          new Integer(id), new Integer(quantity)
      };
      //เป็นการเรียกเมธอดเพื่อสร้างเอเจ้นต์ตัดสต๊อกและดูว่าตัดได้หรือไม่
      return ((Boolean) (new
AgletRMIServerImpl().returnAgletResult("UpdateStockParent", init))).booleanValue();
    }catch(Exception e){out(""+e);}
    return false;
  }
```

3. สร้าง Host Server ในขั้นนี้จะเป็นการบอกว่าเซิร์ฟเวอร์เรามีการ Bind ว่าชื่ออะไร

```
public static void main(String args[]) {
  try {
    System.setSecurityManager(new RMISecurityManager());

    //สร้างอินสแตนซ์
    AgletRMIServerImpl Server = new AgletRMIServerImpl();

    //put the local instance in the registry
    Naming.rebind("//127.0.0.1:1099/RMIATPServer", Server);
    app = new ServerApp();
    app.initServer(app) ;
    System.out.println("Server waiting.....");
  }
  catch (java.net.MalformedURLException me) {
    System.out.println("Malformed URL: " + me.toString());
  }
  catch (RemoteException re) {
    System.out.println("Remote exception: " + re.toString());
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    catch (Exception e) {
      System.out.println("Exception from ATP :" + e);
    }
  }
```

4. Client Application ส่วนนี้จะเป็นการสร้างไคลเอนท์ซึ่งก็เหมือนการสร้างวัตถุในโปรแกรมปกติ เพียงแต่เพิ่มการหาโฮสและเปลี่ยนรูปแบบ (Casting) ของวัตถุเท่านั้น

```
public class AgletRMIClientKea{
  public static void main(String[] arg){
    try{
      long start = System.currentTimeMillis();
      System.out.println("Got remote object");
      //ตัวอย่างนี้จะเป็นการเรียกเมธอด showAll แบบย่อ
      out(""+((AgletRMIServer)Naming.lookup("//127.0.0.1:1099/RMIATPServer")).showAll());
      //แสดงเวลาที่ใช้ไปในการสร้างวัตถุระยะไกล
      out("\nTIME USED:"+(System.currentTimeMillis()-start)+" mS.");
    }catch(Exception e){
        System.out.println("Exception "+e);
    }
  }
```

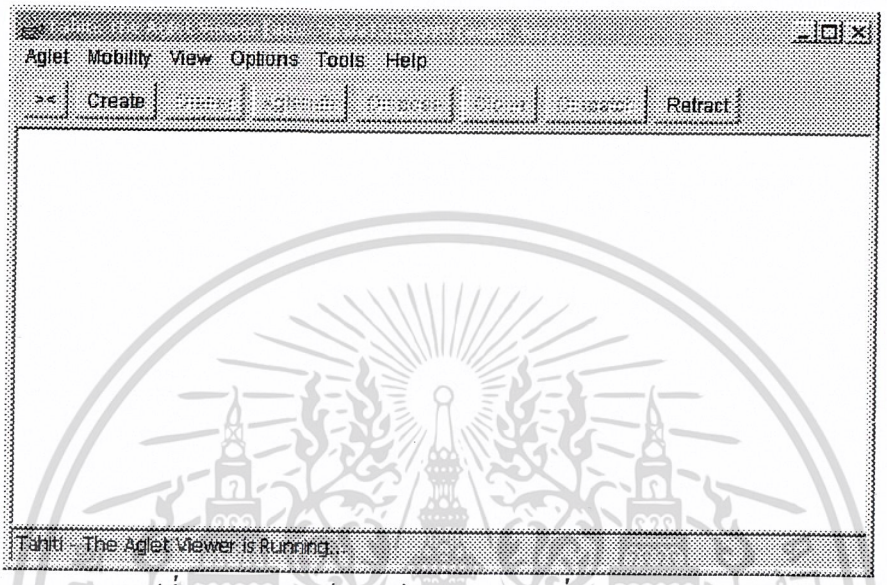
เมื่อมีการคอมไพล์แล้วก็จะได้ Interface, Implementation, Server, Stub และ Skeleton เราก็นำ Stub ไปให้ฝั่งไคลเอนท์ใช้

เมื่อต้องการจะเริ่มเซิร์ฟเวอร์ เราต้องเปิด RMI Registry เพื่อให้ Interface เชื่อมเข้ามาก่อน หลังจากนั้นเมื่อมีการเปิดเซิร์ฟเวอร์ Skeleton ก็จะเตรียมพร้อมไว้ เมื่อมีการเรียก (invoke) จากไคลเอนต์ ก็จะมีการเชื่อมต่อ Stub

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

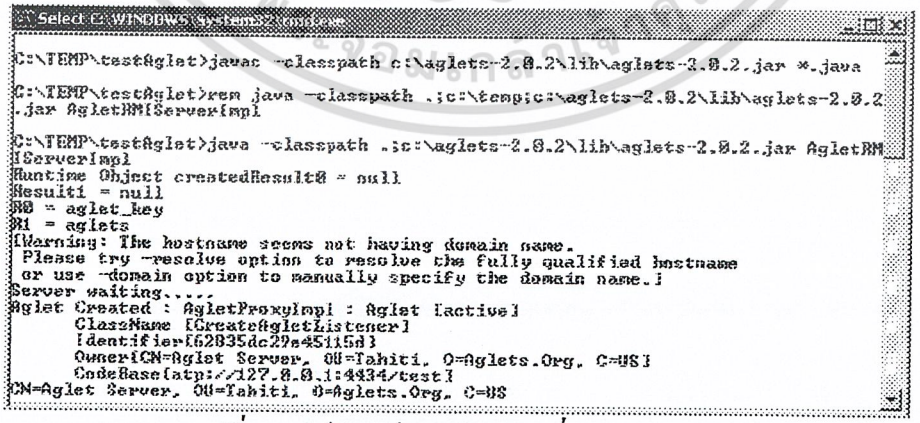
8.2 การสร้าง ATP Server

ดังที่กล่าวไว้แล้วว่า Aglet จะเดินทางผ่านโปรโตคอล ATP และจะใช้ชีวิตอยู่ที่ Place ที่เรียกว่า Context ที่อยู่บน Engine และ Engine นี่เองคือ ATP Server โดย IBM Aglet Framework นั้นมีตัว ATP Server ที่เป็น GUI ให้ผู้ใช้ได้ทดลอง ซึ่งมีหน้าตาดังรูปที่ 8.3



รูปที่ 8.3 Tahiti เซิร์ฟเวอร์สำหรับ Aglet ที่มาพร้อมกับ GUI

แต่ถ้าเราต้องการติดต่อกับ Aglet โดยใช้โปรแกรมที่เราพัฒนาเอง เราจะต้องเขียนเซิร์ฟเวอร์เอง โชคดีที่ ServerApp.java นั้นเป็นโค้ดที่ผู้พัฒนา Aglet ได้สร้างไว้เป็นทางออกในเรื่องนี้แล้ว ทำให้เราสามารถ "ฝัง" ATP Server เข้าไปในตัวโปรแกรมใดๆที่เขียนด้วยภาษาจาวาได้ หน้าตาของผลลัพธ์จากการรันโปรแกรมที่ฝัง ATP Server ลงไปมีดังรูปที่ 8.4



รูปที่ 8.4 เซิร์ฟเวอร์สำหรับ Aglet ที่รวม RMI แล้ว

รูปนี้เป็นขณะที่กำลังสร้าง Aglet

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3 การควบคุม ATP Server ที่สร้างขึ้น

เมื่อเราตั้ง ATP Server เองได้แล้ว หากเราต้องการโปรแกรมให้ Server ทำการสร้าง Aglet สักตัวหนึ่ง เราจะต้องหาวิธีติดต่อกับ ATP Server ให้ได้ ทางแก้ปัญหานี้คือการนำ RMI เข้ามาช่วย โดยการตั้ง ATP Server ที่เป็น RMI Server ซึ่งมีเมธอดสร้าง Aglet ไว้ก่อน หลังจากนั้น เราก็เขียนไคลเอนท์ของ RMI ที่จะเรียกใช้เมธอดเหล่านั้น

เนื่องจากภาษาจาวามีแนวคิดที่ต้องการให้เขียนโปรแกรมครั้งเดียวแต่ใช้ได้หลากหลาย เราจึงสามารถใช้ RMI ไคลเอนท์ที่เป็น Application, Applet, Bean, Servlet หรือแม้กระทั่ง JSP ได้ และจุดนี้เองที่เราสามารถเชื่อมต่อโมบายเอเจนต์เข้ากับเว็บเซอร์วิสได้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9
ความปลอดภัยในเรื่องของ Aglet

9.1 สิ่งผิดพลาดที่เกิดขึ้นได้ของหลักการ Mobile Agent

-ต่อ Agent

- การที่ Remote Host เป็นภัยต่อเอเจ้นต์เช่นการที่เซอร์เวอร์ตัวนั้นแกะข้อมูลที่เอเจ้นต์นำพามาด้วย
- เอเจ้นต์คุกคามเอเจ้นต์ด้วยกันเอง เช่นการที่เอเจ้นต์สายลับ พยายามหาและสื่อสารกับเอเจ้นต์ของเราเพื่อขโมยข้อมูล
- อื่นๆที่เป็นภัยคุกคามต่อเอเจ้นต์คือภัยคุกคามที่ไม่ได้เกิดจากเซอร์เวอร์และเอเจ้นต์ แต่ก็สร้างความเสียหายต่อเอเจ้นต์หรือข้อมูลของเราได้

-ต่อ Host

- การมาของเอเจ้นต์ที่เราไม่รู้จักซึ่งจะมาขโมยข้อมูลหรือทำอะไรก็ตาม
- อื่นๆที่เป็นภัยคุกคามต่อ Host เช่นการส่งเอเจ้นต์จำนวนมากมาทำให้ Host ของเรารองรับไม่ได้และหยุดทำงานไป

-ต่อ Network

- กรณีนี้ก็อย่างเช่นการทำเอเจ้นต์แบบน้ำท่วม ก็อาจจะทำให้ระบบเครือข่ายล่มได้

9.2 วิธีจัดการกับความปลอดภัยของ Aglet

การจัดการกับการรักษาความปอลดภัยในโมบายเอเจนต์จะเหมือนๆกับแอพลิเคชันทั่วไป

9.2.1 การจัดการกับการพิสูจน์ตัวตน

- Authentication of user จะใช้เทคนิค PKI หรือรหัสผ่านมาช่วย
- Authentication of host ในขั้นนี้เราจะต้องรู้ว่าโฮสนั้นๆเชื่อถือได้หรือไม่
- Authentication of code จะใช้เทคนิค ลายเซ็นดิจิตอล เพื่อดูว่าเป็นของบุคคลที่เชื่อถือได้หรือไม่ สมควรรันโค้ดนั้นไหม
- Authentication of agent ในข้อนี้ทางแก้ปัญหายังไม่มีความชัดเจนว่าจะจัดการอย่างไร

9.2.2 การจัดการกับการรักษาความสมบูรณ์ของข้มูล

ในขั้นนี้จะใช้เทคนิค Integrity check ว่า aglet ไม่ได้มีการเปลี่ยนแปลง

9.2.3 การจัดการกับการรักษาความลับ

ในขั้นนี้จะใช้หลักที่ว่า Aglet เดินทางบนเซอร์เวอร์ที่เชื่อถือได้ก็จะเดินทางไปยังเซอร์เวอร์อื่นๆ โดยเส้นทางที่ไม่ถูกดักดูข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9.2.4 การจัดการการกำหนดสิทธิ์

ข้อนี้จัดการ โดยใช้ Access Control ใน policy ที่จะจำกัดความสามารถของ Aglet ที่จะทำงานหรือเข้าถึงต่างๆ

9.2.5 การจัดการกับการรับผิดชอบต่อข้อมูล

Aglet และ เซิร์ฟเวอร์ไม่สามารถปฏิเสธการสื่อสารที่เกิดขึ้นได้

9.2.6 การจัดการกับการตรวจสอบ

บริการนี้จะเป็นการบันทึกสิ่งที่เกี่ยวกับความปลอดภัยไว้เพื่อตรวจสอบภายหลัง

9.3 Aglets Security Policy File

เป็นไฟล์ที่กำหนดขีดความสามารถทุกอย่างซึ่งจะคลอบคลุมถึงการปกป้องเอเจนต์, การปกป้องโฮส และการปกป้องระบบเครือข่าย

ไฟล์นี้จะต้องอยู่ใน

```
{user.home}/.aglets/security/aglets.policy
```

เช่นใน WindowsXP ก็จะเป็น “c:/Documents and Settings/UserName/.aglets/security/aglets.policy” เป็นต้น

ไฟล์ตัวอย่าง sample.policy ได้จากการติดตั้ง Aglets 2.0.2 ตามปกติ จะมีตัวอย่างการตั้งค่าแบบต่างๆ สรุป ได้ดังนี้

การยินยอมสามารถระบุได้ว่าจะเป็น Aglet ที่มาจากคลาสที่โหลดจากที่ใด ทั้งแบบ ATP, HTTP, และ FILE

```
grant
 // codeBase "atp://host.foo.bar:4434/-"
 // codeBase "atp://*.ibm.com:>=1024/"
 // codeBase "atp://*.ibm.com:2000-3000/"
 // codeBase "*://*:*/"
 codeBase "atp://*:*/"
 // , signedBy "onono,moshima" /* code is signed by onono and moshima */
 // , ownedBy "kosaka,mima" /* the aglet is created by kosaka or mima */
{
```

signedBy หมายถึง Aglet นี้ใครเป็นผู้สร้างซึ่งจะยอมหรือไม่ยอมให้ทำอะไร

ownedBy หมายถึง Context นี้เป็นของใครซึ่งจะยอมหรือไม่ยอมให้ทำอะไร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างนี้เป็นการยินยอมให้เข้าถึงไฟล์ที่ path ไหนและทำอะไรได้บ้าง

```
permission java.io.FilePermission "C:/temp", "read,write";
```

ตัวอย่างนี้จะยอมให้ Aglet ชื่อ Abc สามารถ dispatch, dispose, deactivate, activate, clone, และ retract ได้

```
// aglet
permission com.ibm.aglets.security.AgletPermission
  "Abc", "dispatch,dispose,deactivate,activate,clone,retract";
```

ถ้าไม่ต้องการให้ Abc ทำอะไรก็จะใช้ protection ดังนี้

```
// aglet protections
protection com.ibm.aglet.security.AgletProtection
  "Abc", "dispatch,dispose,deactivate,activate,clone,retract";
```

ตัวอย่างนี้ทำให้ Abc สามารถจัดการกับ Message ใดๆก็ได้

```
// messages
permission com.ibm.aglets.security.MessagePermission
  "Abc", "*";
```

ตัวอย่างนี้ทำให้ Abc ถูกป้องกันจาก Message ใดๆ

```
// message protections
protection com.ibm.aglet.security.MessageProtection
  "Abc", "*";
```

ตัวอย่างนี้เป็นการยินยอมให้ Context ใดๆสามารถ Multicast และ Subscribe ได้ (ดูรายละเอียดในบทที่ 6 Aglet Messaging)

```
// aglet context
permission com.ibm.aglets.security.ContextPermission
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  "*", "multicast,subscribe";
 permission com.ibm.aglets.security.ContextPermission
  "*", "create,receive,retract";
 permission com.ibm.aglets.security.ContextPermission
  "property.*", "read,write";
```

ตัวอย่างนี้จะเป็นการกำหนด Runtime ให้ทำอะไรได้บ้าง (โหลดคลาส, เข้าถึงคลาสในแพ็คเกจต่างๆ, โหลดไลบรารี่)

```
// runtime
permission java.lang.RuntimePermission
  "createClassLoader";
permission java.lang.RuntimePermission
  "accessClassInPackage.java.*";
permission java.lang.RuntimePermission
  "accessClassInPackage.com.ibm.aglets.util.*";
permission java.lang.RuntimePermission
  "accessClassInPackage.com.ibm.aglets.AgletProxyImpl";
permission java.lang.RuntimePermission
  "accessClassInPackage.com.ibm.aglet.*";
permission java.lang.RuntimePermission
  "loadLibrary.JdbcOdbc";       // for JDBC/ODBC
permission java.lang.RuntimePermission
  "accessClassInPackage.sun.jdbc.odbc"; // for JDBC/ODBC
```

ตัวอย่างนี้จะเป็นการยินยอมให้ Aglet เปิด Dialogue ได้

```
// window
permission java.awt.AWTPermission "showWindowWithoutWarningBanner";
```

ตัวอย่างนี้จะเป็นการให้ Aglet ที่อยู่ใน codeBase ที่กำหนดด้านบนสามารถเซต System.Property อะไรได้บ้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// property
permission java.util.PropertyPermission "awt.*", "read";
permission java.util.PropertyPermission "hotjava.*", "read";
permission java.util.PropertyPermission "apple.*", "read";
permission java.util.PropertyPermission "file.*", "read";
permission java.util.PropertyPermission "line.separator", "read";
permission java.util.PropertyPermission "path.separator", "read";
permission java.util.PropertyPermission "http.maxConnections", "read";
permission java.util.PropertyPermission "user.timezone", "read";
permission java.util.PropertyPermission "socksProxyHost", "read";
permission java.util.PropertyPermission "socksProxyPort", "read";
// for JDBC/ODBC
permission java.util.PropertyPermission "browser", "read";
// for RMI
permission java.util.PropertyPermission "java.rmi.*", "read";
permission java.util.PropertyPermission "sun.rmi.*", "read";
permission java.util.PropertyPermission "http.proxyHost", "read";
permission java.util.PropertyPermission "proxyHost", "read";
// for examples.patterns.Finger, examples.patterns.Writer
permission java.util.PropertyPermission "user.*", "read";
permission java.util.PropertyPermission "os.*", "read";
permission java.util.PropertyPermission "java.*", "read";
```

ตัวอย่างนี้จะเป็นการยอมให้ใช้ Socket ซึ่งรวมไปถึง RMI ด้วย

```
// socket
permission java.net.SocketPermission "localhost:*", "listen,resolve";
permission java.net.SocketPermission "codebase:*", "connect";
```

ถ้าอยากให้ทำอะไรก็ได้ทุกอย่างก็สามารถใช้ AllPermission แทนบรรทัดเดียวได้

```
// can do anything
permission java.security.AllPermission "*", "*";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 10
แอพพลิเคชัน

10.1 Overview ของระบบ OLALA Movie

ระบบ OLALA Movie เป็นระบบซื้อดีวีดีและซีดี ที่เป็นเว็บเซอร์วิสให้บริการในการค้นหาสินค้าจากแหล่งต่างๆ โดยที่ผู้ใช้บริการจะได้ราคาสินค้าที่ถูกที่สุด ด้วยเทคโนโลยีโมบายเอเจนต์ที่ใช้ในการคำนวณหาสินค้าจากแหล่งที่ถูกที่สุด โดยผู้ใช้บริการเพียงเลือกดีวีดีหรือซีดีที่ต้องการเท่านั้น

เมื่อผู้ใช้บริการต้องการแผ่นภาพยนต์สักเรื่อง ข้อมูลที่ผู้ใช้ต้องกรอกคือ ชื่อเรื่องของภาพยนต์และชนิดของแผ่นว่าจะเป็นดีวีดี หรือซีดี

เมื่อผู้ใช้บริการทำการจองเรียบร้อยแล้ว จะสามารถเข้าตรวจสอบดูรายละเอียด , เปลี่ยนแปลงและยกเลิกการจองครั้งนั้นๆได้ ส่วนเรื่องการชำระค่าบริการจะเป็นการชำระผ่านบัตรเครดิต เมื่อชำระค่าบริการเรียบร้อยแล้ว หากผู้ใช้บริการต้องการยกเลิกรายการที่จองไว้ ผู้ใช้บริการจะไม่ได้ค่าบริการคืน ซึ่งเป็นกฎ ของทางระบบ OLALA Movie

10.2 ระบบ OLALA Movie

ระบบ OLALA Movie เป็นระบบที่มีบริการอยู่ 3 อย่าง ได้แก่ การค้นหาจากชื่อภาพยนต์ การค้นหาจากชื่อร้านค้า การค้นหาจากช่วงของราคา ในการบริการแต่ละแบบนั้นสามารถทำการค้นหาได้ตามเงื่อนไขที่ลูกค้าระบุ ซึ่งเงื่อนไขจะแตกต่างกันไปตามแต่ละบริการ ได้แก่

1.การค้นหาจากชื่อภาพยนต์ สามารถทำการค้นหาได้โดยใส่ข้อมูลเพียงชื่อของภาพยนต์ที่ต้องการ

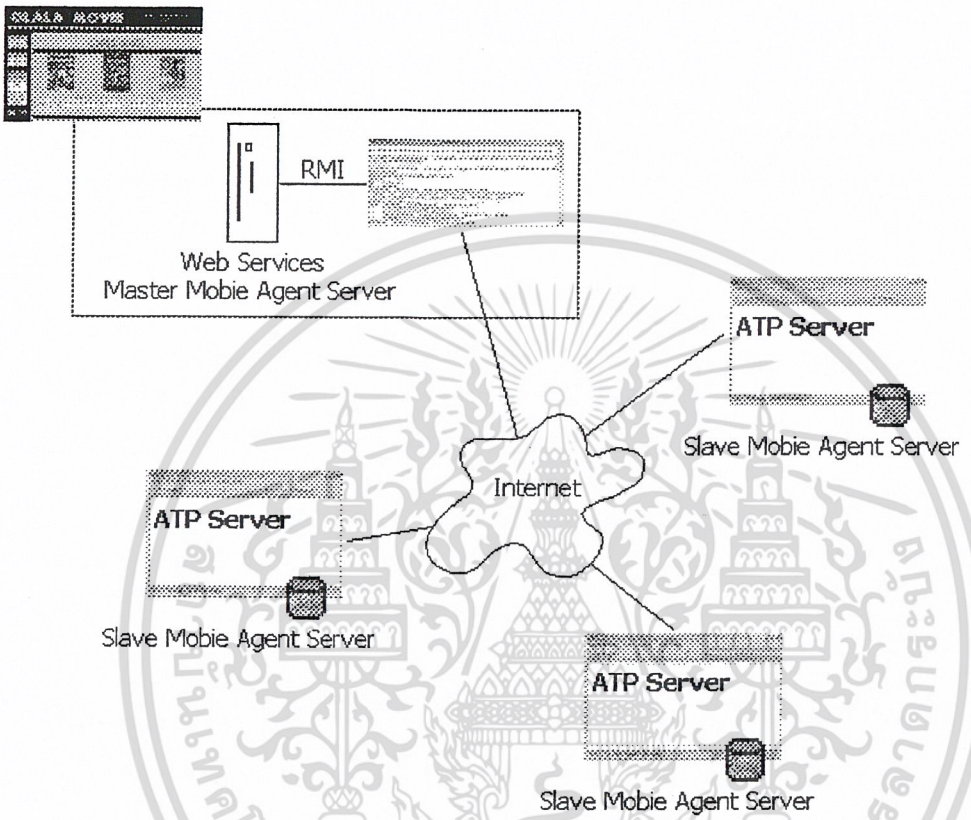
2.การค้นหาจากชื่อร้านค้า สามารถทำการค้นหาจากชื่อของร้านค้าที่ต้องการ

3.การค้นหาจากช่วงของราคา สามารถทำการค้นหาได้จากช่วงของราคา ราคาสูงสุด หรือราคาต่ำสุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10.3 โครงสร้างของระบบ

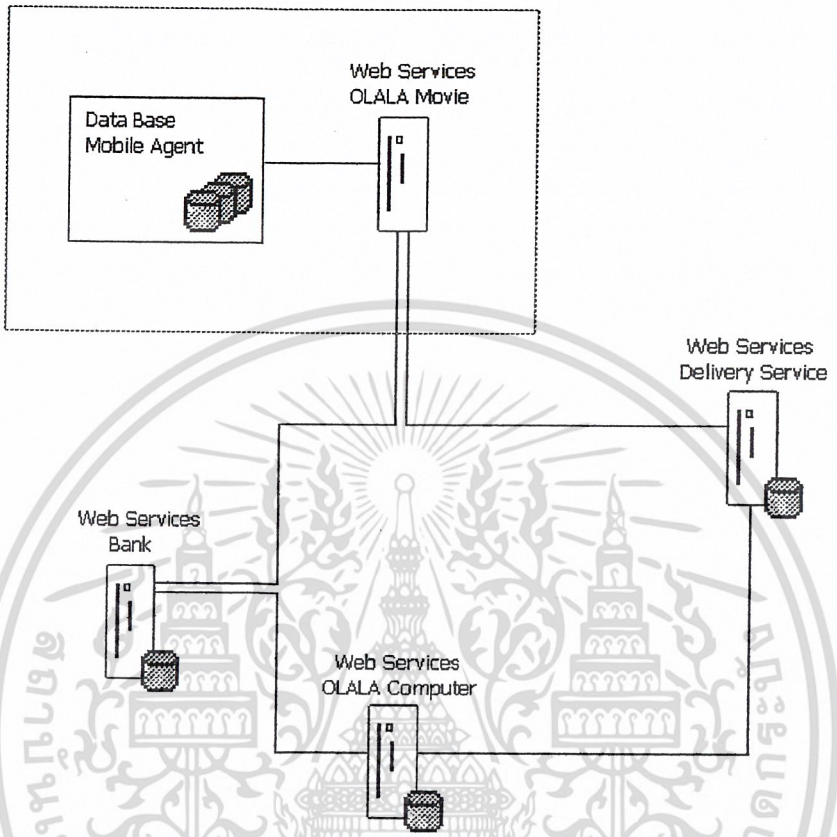
มีเว็บเซอร์วิสหนึ่งระบบที่ทำหน้าที่ดึงและเก็บข้อมูลของโดยผ่านโมบายเอเจนต์ โดยจะมีรูปแบบเอเจนต์เป็นแบบ Master-Slave เพื่อจะไปดึงข้อมูลมา



รูปที่ 10.1 ภาพรวมภายในของโครงงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบของโครงงานนี้จะมีการเชื่อมต่อกับเว็บเซอร์วิสภายนอก 2 แห่ง คือ ธนาคาร และ ผู้บริการส่งสินค้า

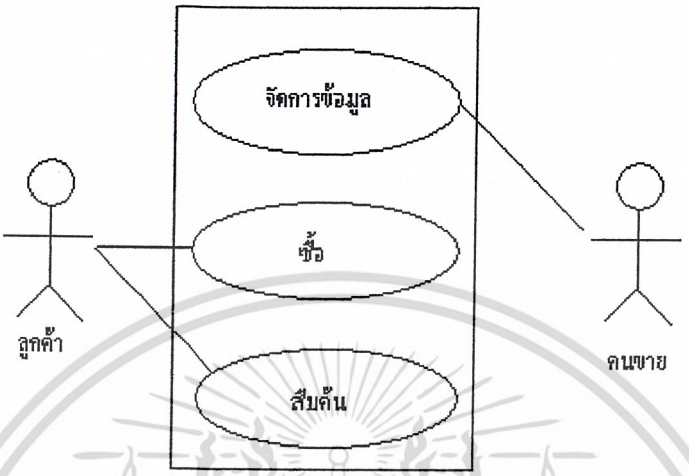


รูปที่ 10.2 ภาพรวมของเว็บเซอร์วิส 3 โครงงาน
ขอบเขตของโครงงานนี้อยู่ในกรอบสีแดง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

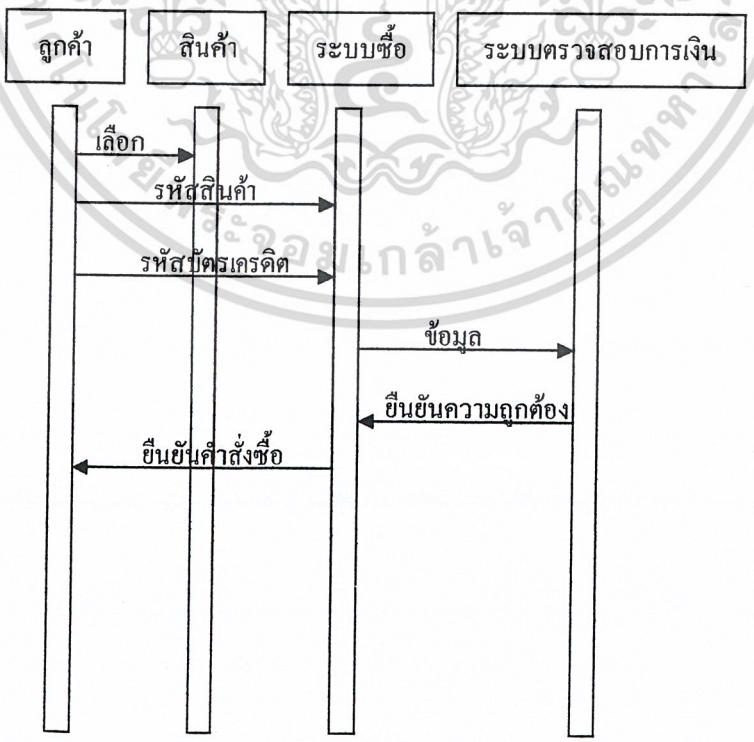
10.4 วิเคราะห์และออกแบบระบบสั่งซื้อดีวีดี/ซีดี ออนไลน์

10.4.1 แผนผังระบบ(Use case)



รูปที่ 10.3 แผนผังของระบบ

10.4.2 ซีเควนซ์ไดอะแกรมของการซื้อ

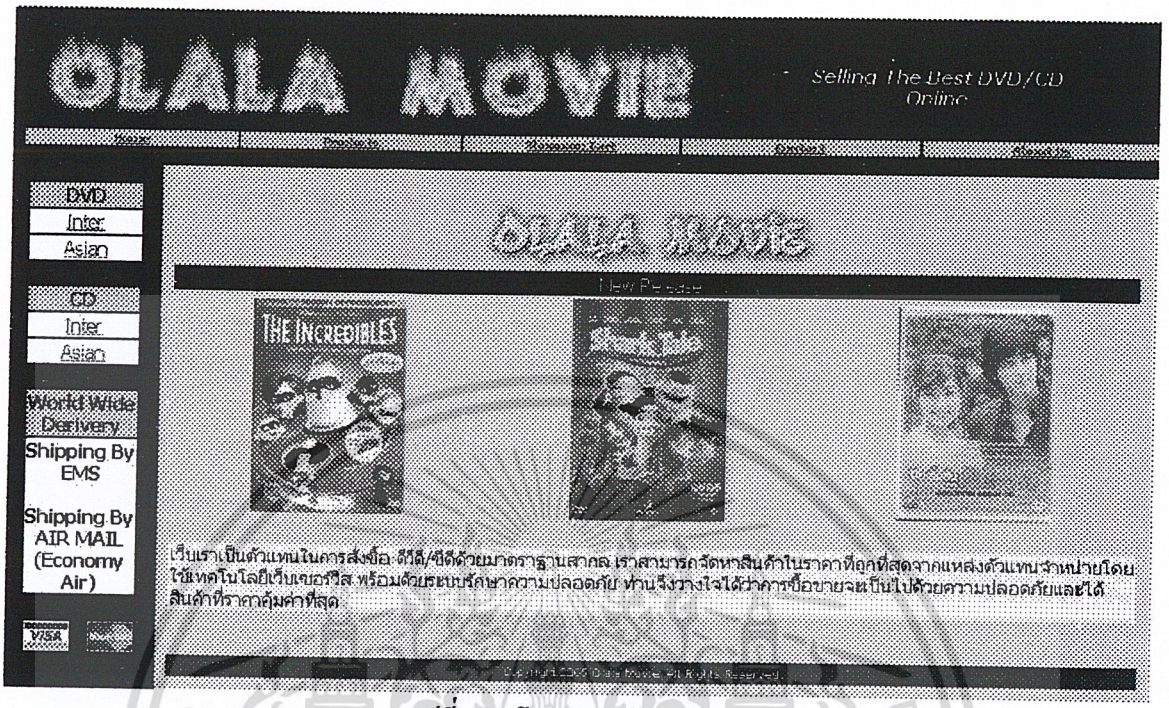


รูปที่ 10.4 ซีเควนซ์ไดอะแกรมของการซื้อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

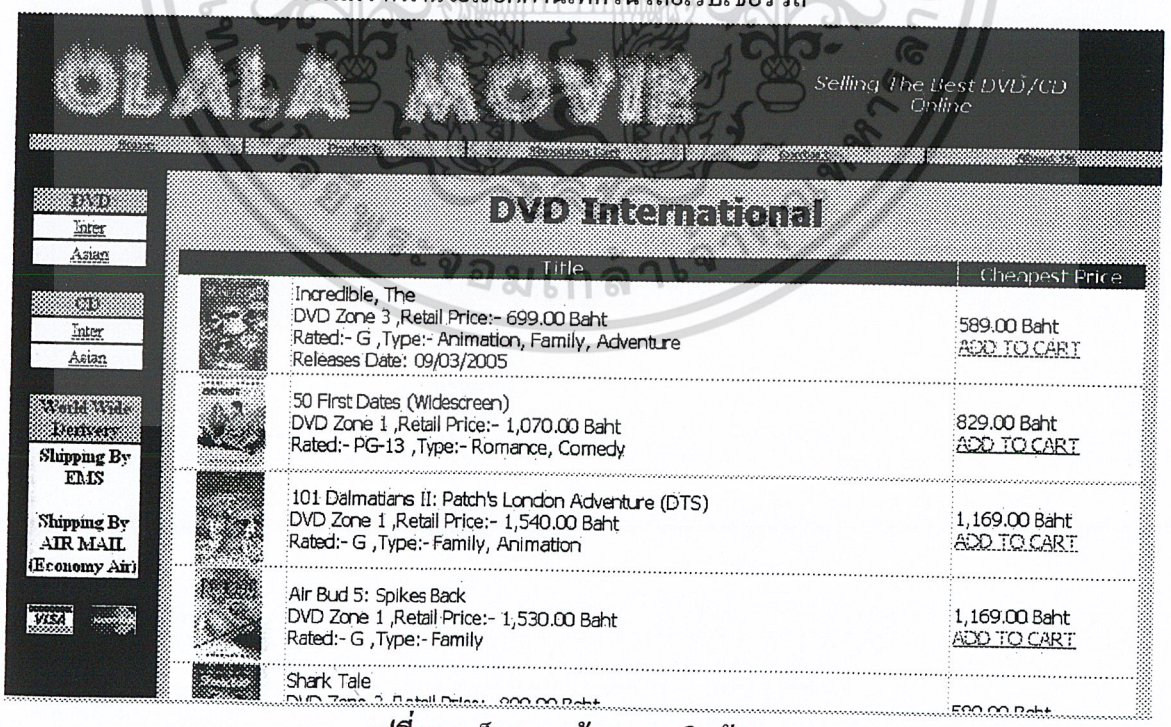
10.5 ส่วนติดต่อผู้ใช้

10.5.1 หน้าหลักของเว็บ



รูปที่ 10.5 โฮมเพจหน้าร้าน

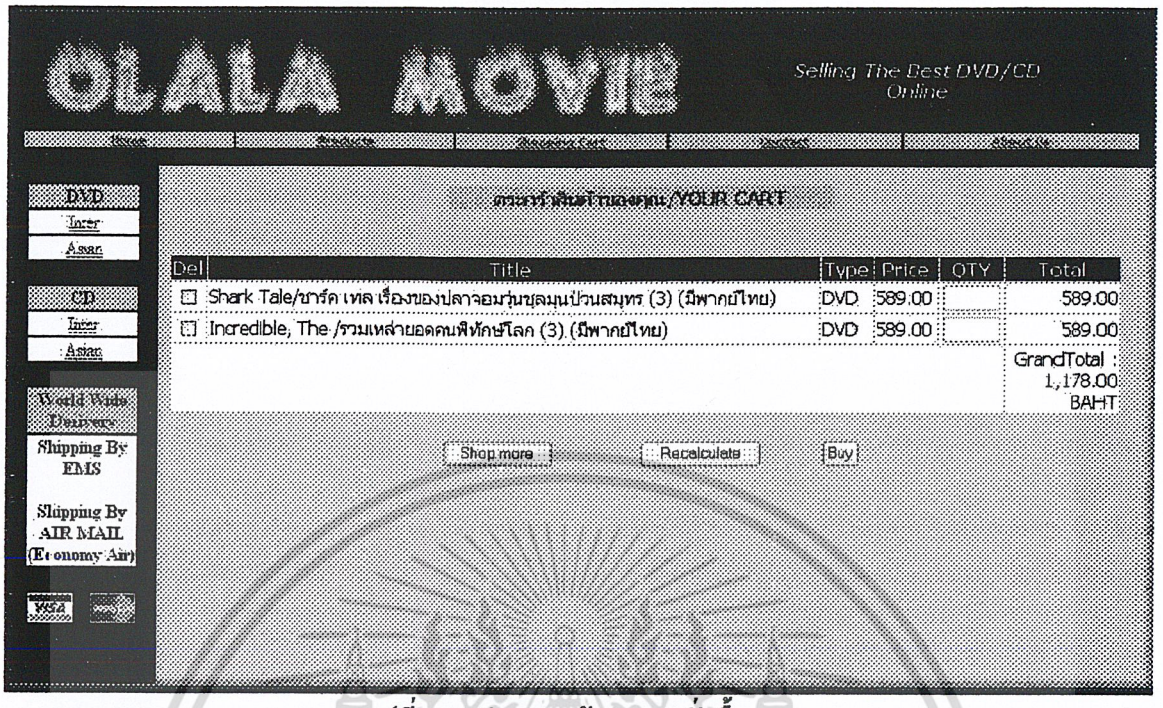
10.5.2 หน้ารายการสินค้า หน้านี้จะเรียกผ่านเทคโนโลยีเว็บเซอร์วิส



รูปที่ 10.6 เว็บเพจหน้ารายการสินค้า

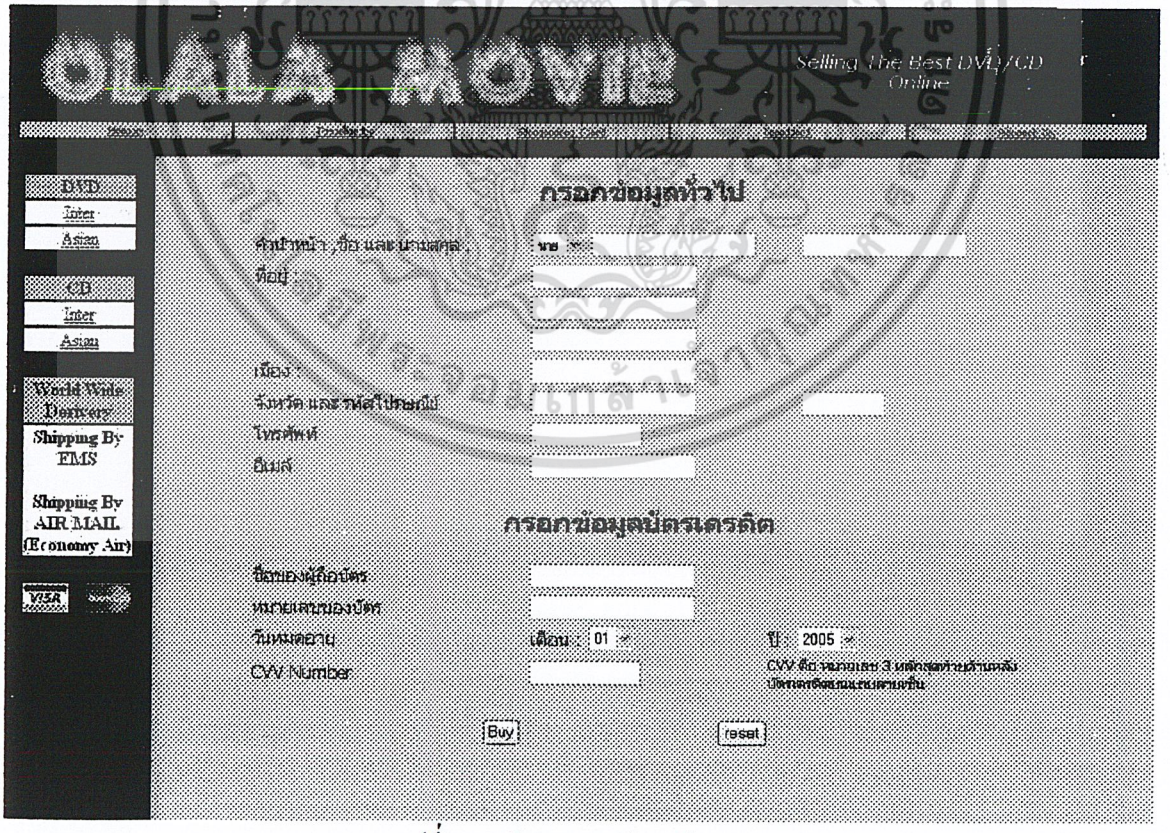
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10.5.3 หน้ารายการสั่งซื้อ จะแสดงรายการที่เราสั่งซื้อไปแล้ว



รูปที่ 10.7 เว็บเพจหน้ารายการสั่งซื้อ

10.5.4 หน้าวิธีชำระเงิน จะใช้วิธีชำระเงินผ่านบัตรเครดิต



รูปที่ 10.8 เว็บเพจการชำระเงิน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 11
บทวิจารณ์และสรุป

11.1 บทวิจารณ์และสรุป

เนื่องจากปัจจุบันแนวคิดในการเชื่อมต่อระหว่างแอพพลิเคชั่นเพื่อให้มีการทำงานอย่างอัตโนมัติ และสามารถที่จะเรียกใช้แอพพลิเคชั่นเพื่อเรียกข้อมูลได้ตลอดเวลาเป็นที่ต้องการมากขึ้นในทุก ๆ ด้าน รวมไปถึงด้านอุตสาหกรรมที่ต้องการให้เกิดการควบคุมเครื่องมือและอุปกรณ์ต่างภายในโรงงานและสถานประกอบการเพื่อลดความเสียหายที่กระทบต่อระบบอุตสาหกรรมน้อยที่สุด และยังช่วยให้ทราบถึงกระบวนการการทำงานของระบบในขณะนั้นนำได้อย่างรวดเร็วมากขึ้น

เทคโนโลยีเว็บเซอร์วิสเป็นอีกหนึ่งแนวคิดในการเชื่อมต่อแอพพลิเคชั่นเข้าด้วยการกำหนดมาตรฐานร่วมกันเพื่อระบบต่างๆ ภายในเครือข่ายที่มีความแตกต่างทางด้านเทคโนโลยีสามารถติดต่อสื่อสารถึงกันได้ และยังช่วยแก้ปัญหาในการเชื่อมโยงข้อมูลที่มาจากแต่ละระบบเข้าหากันได้ โดยเว็บเซอร์วิสได้อาศัยข้อดีในส่วนของโปรโตคอลเอชทีทีพี (HTTP Protocal) ที่นิยมใช้กันอย่างแพร่หลาย และการนิยามข้อมูลที่เป็นอิสระของภาษาเอ็กซ์เอ็มแอล

เทคโนโลยีความปลอดภัยบนเว็บเซอร์วิสนั้น ที่ใช้สื่อสารข้ามระบบนั้นยังมีเอกสารออกมาน้อย ทำให้ยากต่อการพัฒนา แต่คาดว่าจะสามารถพัฒนาได้ง่ายขึ้นในอนาคตอันใกล้นี้

11.2 แนวทางการพัฒนาต่อไป

ปัจจุบันส่วนใหญ่ยังไม่ได้ใช้ความสามารถของเทคโนโลยีเว็บเซอร์วิสอย่างเต็มที่ อีกทั้งยังมาตรฐานความปลอดภัยที่เกี่ยวกับเว็บเซอร์วิสนั้นยังไม่แพร่หลายเพียงพอ การที่จะทำการส่งข้อมูลแบบปลอดภัยข้ามเว็บเซอร์วิสที่พัฒนาจากคนละระบบ(จาวา และ ดอทเนต)กัน 100% ยังกระทำได้ยาก และเอกสารที่เกี่ยวข้องกับเรื่องพัฒนาข้ามระบบก็ยังมีน้อยมากๆ ซึ่งต้องรออีกสักระยะให้ทุกอย่างมีความลงตัวมากขึ้น โดยเฉพาะเครื่องมือในการพัฒนา เนื่องจากขณะที่ทำโครงงานนี้แต่ละผลิตภัณฑ์ได้มีการเปลี่ยนแปลงไปโดยต่างจากเดิม เช่น JDK ก็ได้พัฒนาถึงรุ่น 5 เพื่อเพิ่มคุณลักษณะในการเข้ารหัสแบบต่างๆขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก. IBM Aglet

1. การติดตั้ง-ตั้งค่า Aglet 2.0.2

การใช้ Aglet นั้นต้องมี J2SDK ติดตั้งอย่างถูกต้องไว้ก่อนแล้ว เมื่อดาวน์โหลด aglets-2.0.2.zip มาแล้วก็ให้ขยายไว้ที่ c:\aglets-2.0.2 แล้วตั้งค่า Environment ดังนี้

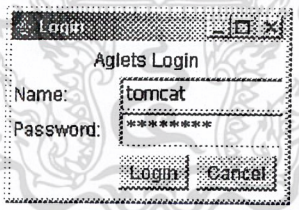
AGLET_HOME=c:\aglets-2.0.2	เป็นไดเร็คทอรี่หลัก
AGLET_PATH=c:\aglets-2.0.2\public	เปรียบเหมือน CLASSPATH
AGLET_EXPORT_PATH= c:\aglets-2.0.2\public	ใช้อ้างเมื่อมีการเรียก CodeBase จากเซิร์ฟเวอร์นั้น
CLASSPATH=	ไม่ควรตั้งเป็นค่าใดๆ เพราะอาจทำให้เอเจ้นต์เดินทางไม่ได้

จำเป็นต้องมีคีย์ใน User Home (เช่น C:/Documents and Settings/BlackJack) เราสามารถสร้างโดย คำสั่ง keytool –genkey –alias aliasname ในจาวาเอง รายละเอียดอ่านใน keytool –help

และเมื่อสร้างคีย์แล้วก็ให้ใช้คำสั่ง ant.bat (ระบบ Windows) ในการติดตั้ง ที่ c:\aglets-2.0.2\bin

เมื่อมีการสร้างแล้วเราก็จะนำ alias และ password มาใช้ในการเริ่ม Tahiti Aglet Viewer ซึ่งเป็นเซิร์ฟเวอร์แบบ GUI ที่มีมาให้

การเริ่ม Tahiti ใช้คำสั่ง agletsd.bat (ระบบ Windows) ที่ c:\aglets-2.0.2\bin จะได้รูป



รูปภาคผนวก ก-1 หน้าล็อกอิน

สังเกตว่าพอร์ตปกติจะเป็น 4434 แต่ถ้าต้องการเป็นอย่างอื่นให้ใส่อากิวเมนต์ต่อท้าย เช่น –port 4999 นอกจากนี้เรายังสามารถตั้งค่า alias และ password ในไฟล์ C:\aglets-2.0.2\cnf\aglets.props ตามตัวอย่าง เพื่อให้เริ่ม Tahiti ได้เลย ด้วยอากิวเมนต์ –f C:\aglets-2.0.2\cnf\aglets.props

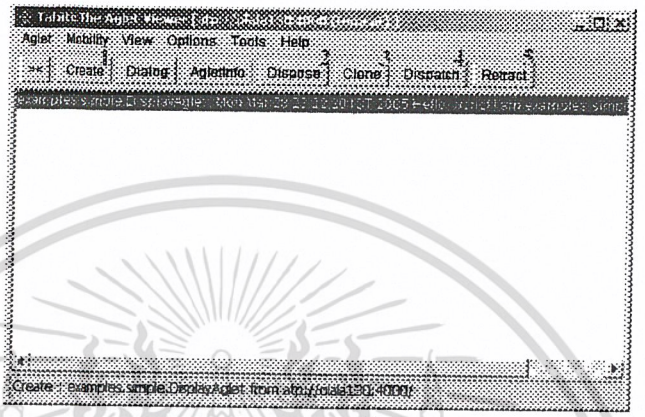
```
#เปลี่ยนให้ตรงกับของคุณ
aglets.owner.name=tomcat
aglets.owner.password=changeit
aglets.keystore.password=changeit
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การใช้งาน Tahiti

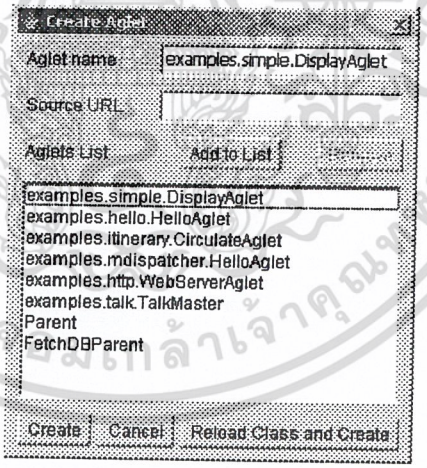
หน้าต่างหลักที่ใหญ่ที่สุดจะเป็นหน้าที่แสดงตัว Aglet ที่อยู่ที่คอนเท็กซ์นี้ การสร้างตัวเอเจ้นต์สักตัวหนึ่งจะต้อง

คลิ๊กที่ปุ่ม Create (ปุ่มที่ 1)



รูปภาคผนวก ก-2 หน้าต่าง Tahiti

จะได้หน้าต่าง Create Aglet ดังรูป แล้วพิมพ์ชื่อคลาสที่จะสร้างเป็นตัวเอเจ้นต์ในช่อง Aglet name



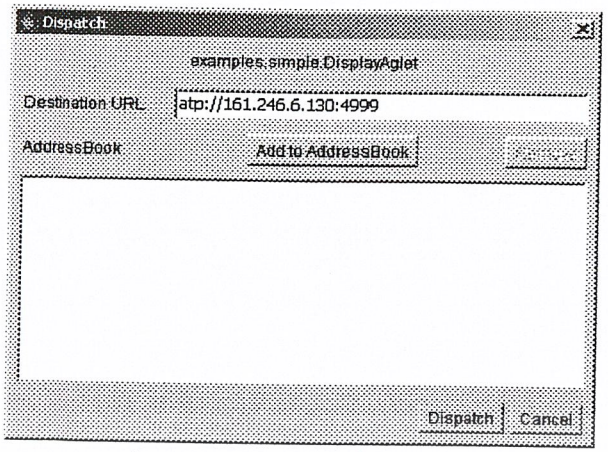
รูปภาคผนวก ก-3 หน้าต่าง Create Aglet

แล้วคลิ๊กที่ปุ่ม Create

เมื่อสร้างแล้ว ถ้า Aglet สามารถทำงานได้ด้วยตัวเองก็จะเห็นว่ามันมีการทำงานต่างๆ แต่ถ้าต้องการบังคับให้ Aglet จบชีวิตตัวเองให้คลิ๊กที่ Dispose (ปุ่มที่ 2) แล้วยืนยัน

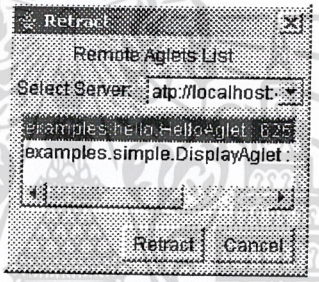
ถ้าต้องการย้ายตัวเองให้คลิ๊กที่ Dispatch (ปุ่มที่ 4) แล้วพิมพ์ URL ของปลายทางที่จะไปในช่อง Destination URL แล้วคลิ๊กที่ Dispatch

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปภาคผนวก ก-4 หน้าต่าง Dispatch

สำหรับปุ่มที่ 3 จะเป็นการ โคลนนิ่งตัว Aglet และปุ่มที่ 5 จะเป็นการดึงกลับ โดยเลือกเซิร์ฟเวอร์ใน Select Server และเลือกตัว Aglet ที่ต้องการแล้วคลิ๊ก Retract



รูปภาคผนวก ก-5 หน้าต่าง Retract

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. การเขียนโปรแกรมสร้างเอเจ้นต์เซิร์ฟเวอร์

ให้ดาวน์โหลดไฟล์ ServerApp.java จากเว็บไซต์ผู้พัฒนาแล้วนำมาสร้างวัตถุ ซึ่งมันจะไม่ค่อยมีความสามารถเท่ากับ Tahiti นัก เมื่อสร้างวัตถุแล้วจะมีพอร์ตเป็น 4434 เป็นปกติ เมื่อสร้างวัตถุของ ServerApp แล้วก็ให้เริ่มเซิร์ฟเวอร์ ก็จะได้ ATP Server ที่มีคอนเท็กซ์สำหรับให้ Aglet ทำงานได้

```
ServerApp app = new ServerApp();
app.initServer(app) ;
```

แต่เราต้องเพิ่มเมธอด startAglet ในโค้ด ServerApp ดังที่โครงงานเราได้ทำ เพื่อให้อย่างน้อยเซิร์ฟเวอร์เราสามารถให้กำเนิด Aglet ได้

```
public AgletProxy startAglet(URL urlCodeBase,String agletName,Object objArg){
 if(success){
  try {
   station = cxt.createAglet(urlCodeBase,agletName,objArg);
   return station;
  }catch (Exception e) {
   System.out.println("Start context error:" + e);
  }
 }else{
  System.out.println("The server was not started properly");
 }
 return null;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. การเขียนโปรแกรมเอเจ้นต์ตัวอย่าง

Aglet ที่ทำงานกับเซิร์ฟเวอร์ที่เราสร้างต้องมีการติดต่อผ่านคลาสข้อความ ซึ่ง Aglet ต้องโอเวอร์รายด์เมธอด handleMessage เพื่อมาจัดการ ตัวอย่าง **handleMessage** และ **proxy.sendMessage** สองกรอบด้านล่างนี้เป็นเทคนิคหนึ่งในการจัดการ

```
public boolean handleMessage(Message msg){
 try{
  if(msg.sameKind("God Father wanna know")){
   if(System.currentTimeMillis()-start>TIMEOUT)resultAllDone=true;
   if(resultAllDone)
    msg.sendReply(result);
   else
    msg.sendReply(null);
   return true;
  }
  return false;
 }catch(Exception e){}
 return false;
}
```

เมธอดที่ใช้จัดการรับข้อความในเอเจ้นต์

```
if(name.equals(""))return null;
if(init==null)init="";
AgletProxy proxy=null;
proxy=app.startAglet(new URL("atp://127.0.0.1:4000"),name,init);
Object reply=null;
short count=0;
while(reply==null){
 try{
  reply = proxy.sendMessage(new Message(
     "God Father wanna know"));
 }catch(Exception e){out(""+e);}
 if(count++>10){out("Time out");break;}
 Thread.sleep(500);
}
```

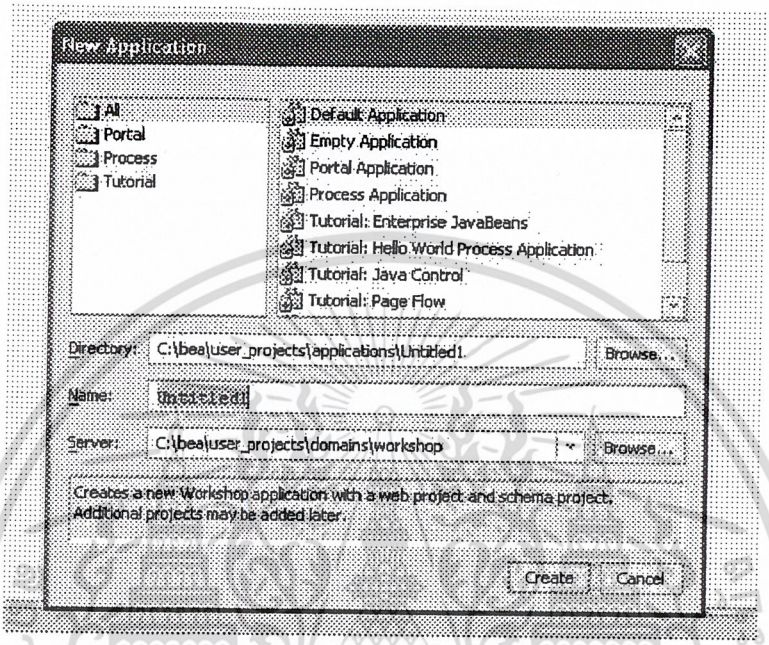
เซิร์ฟเวอร์ที่ใช้คู่กับเอเจ้นต์ที่มีข้อความด้านบน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข. การสร้างเว็บเซอร์วิสตัวอย่างที่มี ws-security โดย Bea Weblogic

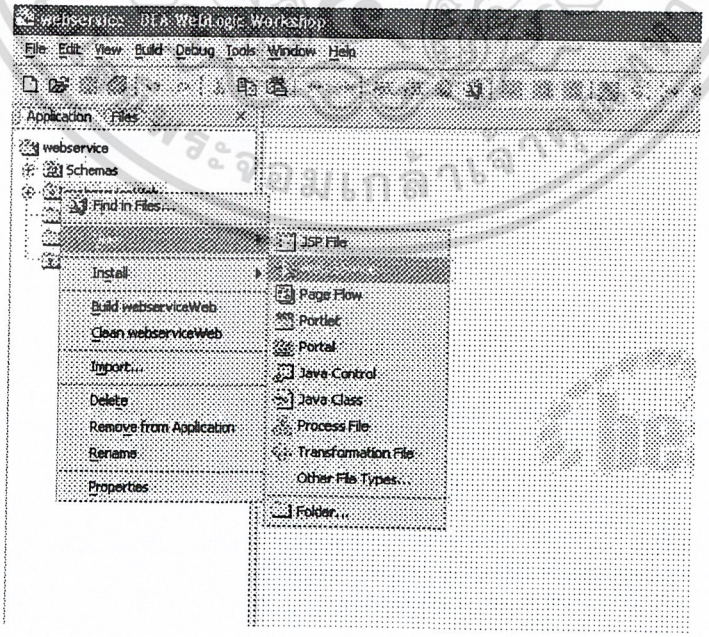
1.เปิดโปรแกรม Bea Weblogic Workshop

2.สร้าง New Application



รูปภาคผนวก ข-1

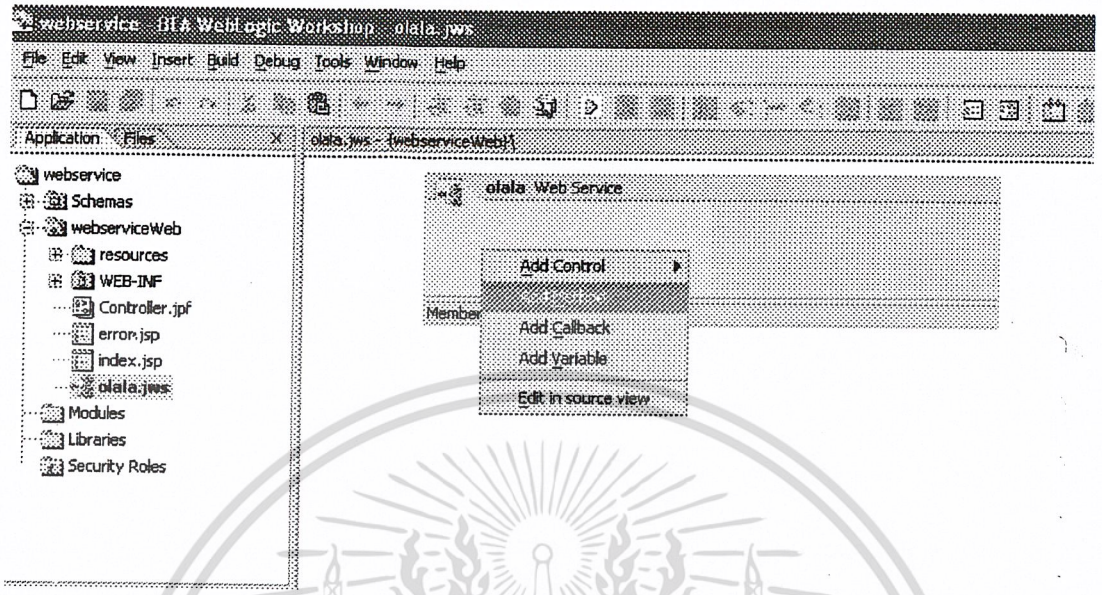
3.สร้าง New -> Web Service



รูปภาคผนวก ข-2

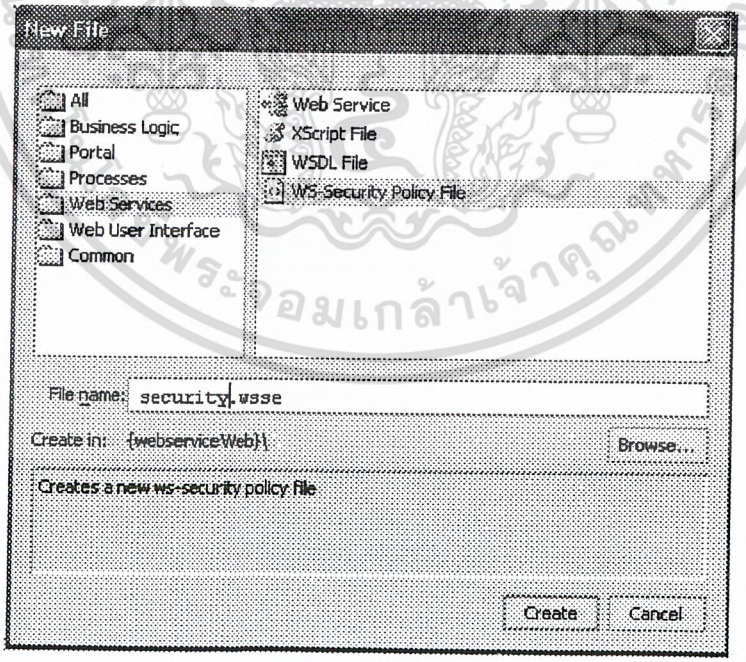
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.สร้าง method ของ webservice ที่ต้องการให้บริการ



รูปภาคผนวก ข-3

4.สร้าง ws-security policy file



รูปภาคผนวก ข-4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. ตัวอย่าง code ของ ws-security policy file

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
xmlns="http://www.bea.com/2003/03/wsse/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <wsSecurityIn>
        <encryptionRequired>
            <decryptionKey>
                <alias>mycompany</alias>
                <password>password</password>
            </decryptionKey>
        </encryptionRequired>

        <signatureRequired>true</signatureRequired>
    </wsSecurityIn>

<wsSecurityOut>
        <encryption>
            <encryptionKey>
                <alias>client1</alias>
            </encryptionKey>
        </encryption>

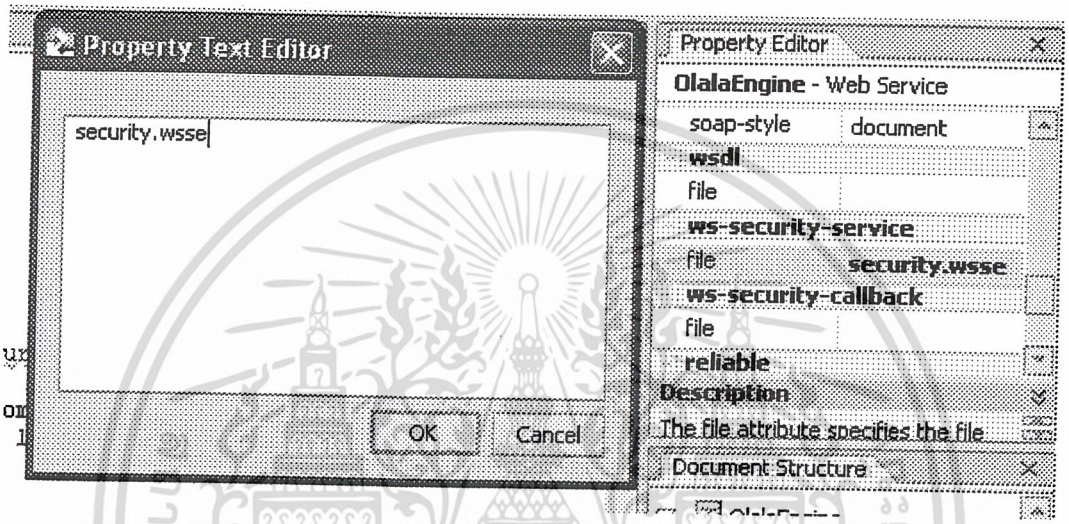
  <signatureKey>
            <alias>mycompany</alias>
            <password>password</password>
        </signatureKey>
</wsSecurityOut>

    <keyStore>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            <keyStoreLocation>C:\samples_mycompany.jks</keyStoreLocation>
            <keyStorePassword>password</keyStorePassword>
        </keyStore>
</wsSecurityPolicy>
```

6. ทำการเรียกใช้ ws-security ดังรูปภาคผนวก ข5 เป็นการเรียกใช้ security.wsse



รูปภาคผนวก ข-5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสร้างเว็บเซอร์วิสไคลเอนท์

1.ตัวอย่างโค้ดของไคลเอนท์(.jsp)ที่เรียกเว็บเซอร์วิสที่มี ws-security

```
<%@ page language="java" contentType="text/html;charset=UTF-8"
import = "java.io.IOException"
import = "java.io.FileInputStream"
import = "java.security.KeyStore"
import = "java.security.PrivateKey"
import = "java.security.KeyStoreException"
import = "java.security.NoSuchAlgorithmException"
import = "java.security.cert.X509Certificate"
import = "java.security.cert.CertificateException"
import = "java.util.List"
import = "java.util.ArrayList"
import = "java.security.KeyStore"
import = "java.security.PrivateKey"
import = "java.security.cert.X509Certificate"
import = "javax.xml.rpc.ServiceException"
import = "javax.xml.namespace.QName"
import = "javax.xml.rpc.handler.HandlerInfo"
import = "javax.xml.rpc.handler.HandlerRegistry"
import = "weblogic.webservice.context.WebServiceSession"
import = "weblogic.webservice.context.WebServiceContext"
import = "weblogic.webservice.core.handler.WSSEClientHandler"
import = "weblogic.webservice.WLMessageContext"
import = "weblogic.xml.security.wsse.Security"
import = "weblogic.xml.security.wsse.Token"
import = "weblogic.xml.security.wsse.BinarySecurityToken"
import = "weblogic.xml.security.wsse.SecurityElementFactory"
import = "weblogic.xml.security.specs.EncryptionSpec"
import = "weblogic.xml.security.specs.SignatureSpec"
import = "weblogic.jws.proxies.*"
//ต้องทำการ import java proxy จากฝั่ง webservice (ขอจาก ฝั่ง webservice)โดยทำการ add ไว้ที่
library
%>
<%!
public static class KeyUtil {

   public static KeyStore loadKeystore(String filename, String password)
   throws KeyStoreException, IOException, NoSuchAlgorithmException, CertificateException
   {
      final KeyStore ks = KeyStore.getInstance("JKS");
      ks.load(new FileInputStream(filename), password.toCharArray());
      return ks;
   }

   public static PrivateKey getPrivateKey(String alias, String password, KeyStore keystore)
   throws Exception
   {
      PrivateKey result =(PrivateKey) keystore.getKey(alias, password.toCharArray());
      return result;
   }

   public static X509Certificate getCertificate(String alias, KeyStore keystore)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        throws Exception
        {
            X509Certificate result = (X509Certificate) keystore.getCertificate(alias);
            return result;
        }

    }
    %>
    <%
    //service's variable
    //กำหนดค่าตัวแปรต่างๆ

    //key
    final String CLIENT_KEYSTORE = "C:\\samples_client.jks";
    //หาได้จากการสร้างโดย keytool หรือว่า เอามาจาก defualt ของ weblogic
    final String KEYSTORE_PASS = "password";
    final String KEY_ALIAS = "client1";
    final String SERVER_KEY_ALIAS = "mycompany";
    final String KEY_PASSWORD = "password";
    try
    {
        final KeyStore keystore = KeyUtil.loadKeystore(CLIENT_KEYSTORE, KEYSTORE_PASS);

        Service myservice = new
    Service_Impl("http://161.246.6.108:7001/projectWeb/orderservice/service.jws?WSDL=");
    //การเรียกใช้ เว็บเซอร์วิส โดยกรณีนี้ เป้าหมาย เว็บเซอร์วิส มีชื่อว่า service

        WebServiceContext context = myservice.context();
        WebServiceSession WSsession = context.getSession();

        PrivateKey clientprivate  = KeyUtil.getPrivateKey(KEY_ALIAS, KEY_PASSWORD, keystore);
        X509Certificate clientcert = KeyUtil.getCertificate(KEY_ALIAS, keystore);
        X509Certificate servercert = KeyUtil.getCertificate(SERVER_KEY_ALIAS, keystore);

        SecurityElementFactory factory = SecurityElementFactory.getDefaultFactory();

        Token client_x509token = factory.createToken(clientcert, clientprivate);
        Token x509token = factory.createToken(clientcert, clientprivate);
        EncryptionSpec encSpec = EncryptionSpec.getDefaultSpec();
        SignatureSpec sigSpec = SignatureSpec.getDefaultSpec();
        Token serverToken = factory.createToken(servercert, null);
        Security security = factory.createSecurity(null);
        //encryption client to server
        security.addEncryption(serverToken, encSpec);
            //signature 2 ways
        security.addSignature(x509token, sigSpec);
        security.addToken(x509token);

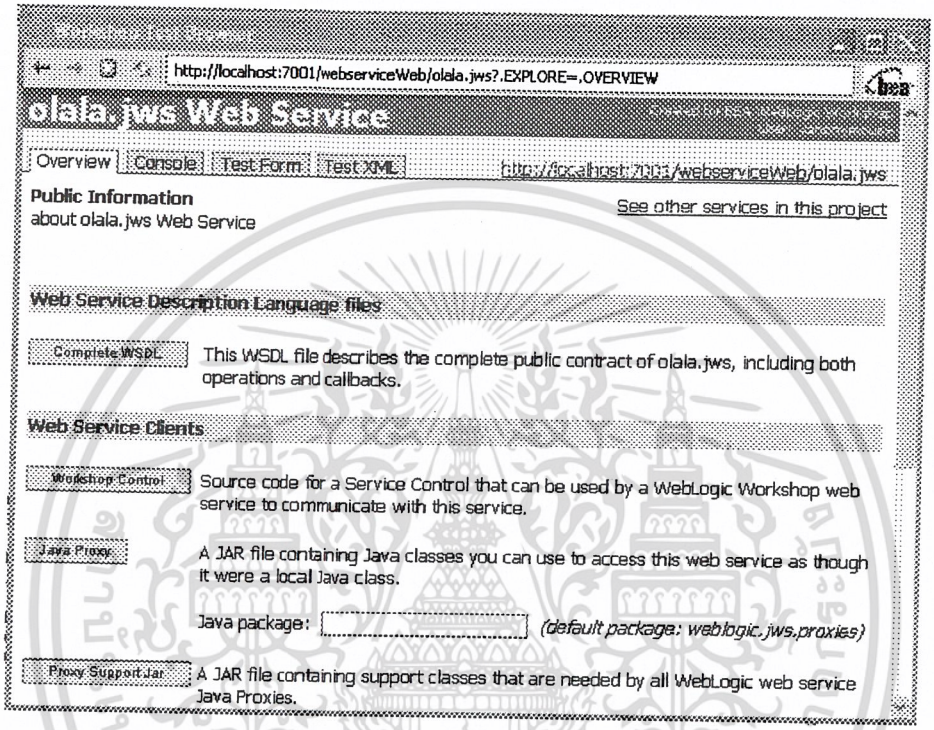
        ServiceSoap msg = myservice.getserviceSoap();

        context.getSession().setAttribute(WSSEClientHandler.REQUEST_SECURITY, security);
        WSsession.setAttribute(WSSEClientHandler.KEY_ATTRIBUTE, clientprivate);
        int
    i=msg.addorder(sfname,slname,spackage,s_add,dfname,dlname,d_add,sphone,dphone);
    //การเรียกใช้ method ของเว็บเซอร์วิส
        System.out.println(i);
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
catch(Exception e)
{
   System.out.println("------------------------------");
   e.printStackTrace();
}
%>
```

หมายเหตุ การขอ java proxy จากเว็บเซอร์วิส ขอได้รูปภาคผนวก ข6 โดยกดปุ่ม Java Proxy



รูปภาคผนวก ข-6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การตั้งเซิร์ฟเวอร์ให้มี SSL

1.config ที่ไฟล์ WEB-INF/web.xml โดยการเพิ่มโค้ดดังนี้

```
<?xml version="1.0" ?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

<security-constraint>
        <display-name>
           Test SSL
        </display-name>
                <web-resource-collection>
          <web-resource-name>allfile</web-resource-name>
          <description>A web service secured by SSL and basic authentication</description>
          <!--
          Defines the scope of the web resource to be secured with SSL.
          Secure all methods calls to the HelloWorldSecure web service.
          -->
          <url-pattern>/*</url-pattern>
          <http-method>GET</http-method>
          <http-method>POST</http-method>
                </web-resource-collection>
      <!--Encrypt the traffic between the client and this web resource. -->
      <user-data-constraint>
          <transport-guarantee>CONFIDENTIAL</transport-guarantee>
      </user-data-constraint>
        </security-constraint>
</web-app>
```

2. จากนั้นให้ทำการ run server ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค. การเขียนโปรแกรม RMI

1. RMI Interface ตัวอย่างจากแอพลิเคชั่นในโครงงานนี้ เมธอดที่ต้องการให้เรียกจากระยะไกลได้ต้องประกาศในนี้

```
public interface AgletRMIServer extends Remote{
  public String showAll() throws RemoteException;
  public boolean updateStock(String retailShopCode, int id, int quantity) throws
RemoteException;
}
```

2. RMI Implements ต้องมีการสร้างจาก Interface

```
public class AgletRMIServerImpl extends UnicastRemoteObject implements
AgletRMIServer {
  AgletRMIServerImpl() throws RemoteException {
    super();
  }
  public String showAll(){ ... }
  public Boolean updateStock(...){ ... }
}
```

3. RMI เซิร์ฟเวอร์ จะเป็นการตั้งคลาสในข้อ 1 และ 2 ให้เป็นเซิร์ฟเวอร์

```
public static void main(String args[]) {
    AgletRMIServerImpl Server = new AgletRMIServerImpl();
    Naming.rebind("//127.0.0.1:1099/RMIATPServer", Server);
}
```

4. RMI ไคลเอนท์ นี่เป็นตัวอย่างที่เขียนรวบให้เหลือบรรทัดเดียวในการเรียกเมธอด

```
((AgletRMIServer)Naming.lookup("//161.246.6.130:1099/RMIATPServer")).showAll();
```

5. การใช้งาน RMI

เมื่อมีการเขียนโค้ดครบทั้ง 4 ข้อและคอมไพล์แล้ว เราจะต้องตั้ง CLASSPATH ไปยังคลาส Implements ก่อน

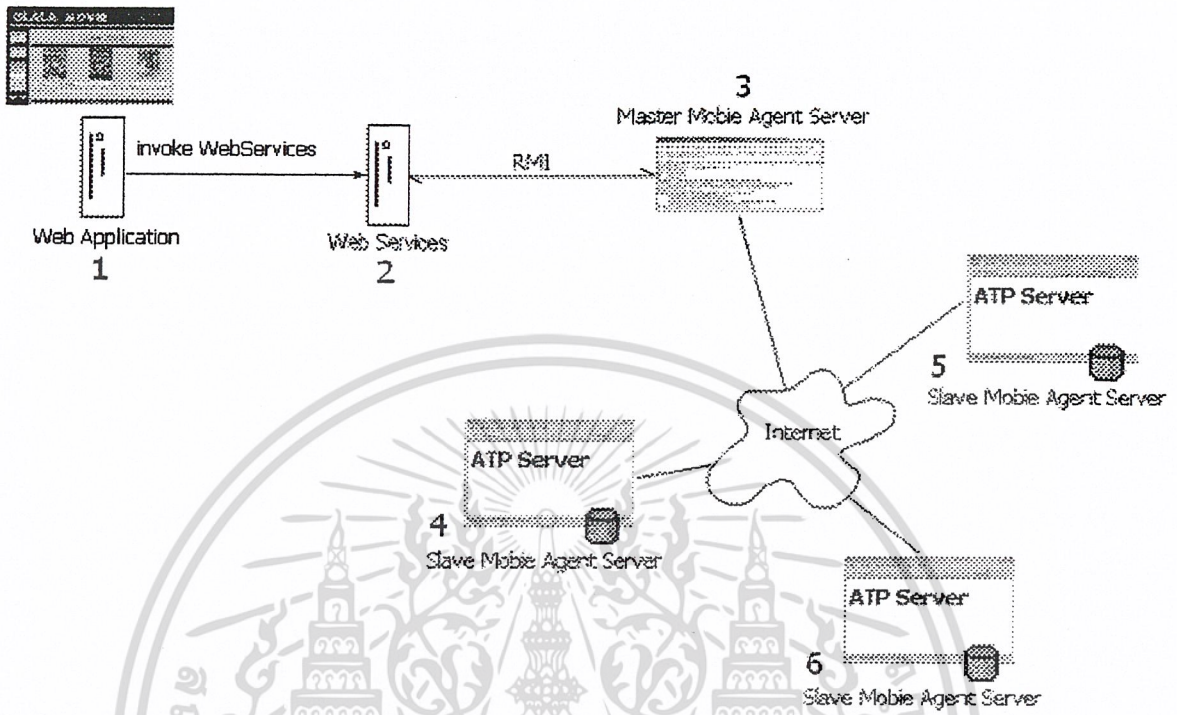
เสร็จแล้วก็ต้องมีการเรียก rmiregistry เพื่อให้ RMI Registry ทำงานเพื่อให้ Skeleton มาเชื่อมต่อได้แล้วเราจึงเรียกโปรแกรมเซิร์ฟเวอร์

เมื่อคอมไพล์ Implements แล้วจะมี _Stub และ _Skeleton ออกมา เราจะนำ คลาส Stub และ Interface ไปไว้ที่เดียวกับไคล์เอนท์ก่อนที่จะรัน

เมื่อเปิด RMI Registry และเซิร์ฟเวอร์แล้วเราก็จะเรียกไคลเอนท์ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ง. ไฟล์ที่อยู่บนเครื่องต่างๆในระบบ



รูปภาคผนวก ง-1 ระบบเว็บเซอร์วิส-โมบายเอเจ้นต์

เครื่องหมายเลข 1 จะเป็น เครื่องที่ติต่อกับลูกค้าและใช้ข้อมูลผ่านเว็บเซอร์วิส ประกอบด้วยไฟล์

addtocart.jsp	เป็นส่วนที่ถูกใช้เมื่อมีการเลือกสินค้า มันจะเพิ่มรายการในคุ๊กกี้
buynow.jsp	เป็นส่วนที่ใช้เมื่อจะชำระเงิน มันจะกระโดดไปยังหน้า information.html
catalog.jsp	เป็นส่วนที่เรียกใช้เว็บเซอร์วิสโดยนำคิวรีที่ได้ไปเปลี่ยนเป็นหน้า HTML
CatalogBean.java	เป็นบีนที่ใช้ในการจัดการคุกกี้และเปลี่ยนรูปแบบคิวรีเป็น HTML
content.jsp	เป็นเฟรมหนึ่งที่แสดงหน้าแรกของเว็บ ซึ่งจะเป็นข้อความต้อนรับ
incart.jsp	เป็นส่วนที่ใช้แสดงสินค้าทั้งหมดในรายการที่เลือกไว้แล้วพร้อมแสดงจำนวนเงิน
information.html	เป็นส่วนฟอร์มสำหรับกรอกข้อมูลเครดิตการ์ด
main.html	เป็น HTML ที่รวม 3 เฟรม
menu.jsp	เป็นส่วนแสดงสารบัญของเว็บ
top.jsp	เป็นเฟรมบนสุดของเว็บ ใช้แสดงโลโก้และโฆษณา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องหมายเลข 2 จะเป็นเครื่องที่ให้บริการเว็บเซอร์วิสในการแสดงรายการสินค้าและการสั่งซื้อสินค้า ประกอบด้วยไฟล์

OlalaEngine.jws	เป็นเซอร์วิสหลักของเว็บเซอร์วิสมี 2 เมธอด คือ String showAll() ซึ่งจะคืนค่าเป็นสตริงที่เป็นดิวรีที่ได้มาจากการเรียกโมบายเอเจ้นต์ผ่าน RMI Boolean buy(String retailCode, int id, int quantity) ซึ่งจะคืนค่าเป็นบูเลี่ยน โดยรับรหัสร้านค้า(String), รหัสสินค้า(int) และจำนวน(int)
security.wsse	เป็นไฟล์ที่กำหนดเรื่องความปลอดภัยข้อมูลเข้าออกเว็บเซอร์วิสนี้
mobile.jar	เป็น JAVA Archive ที่รวม Interface และ Stub ของแพ็กเกจ mobileagent (ด้านล่าง)

เครื่องหมายเลข 3 จะเป็นเครื่องเซิร์ฟเวอร์หลักของโมบายเอเจ้นต์ ซึ่งต้องเขียนโค้ดเอง ประกอบด้วยคลาส

ServerApp	เป็นเซิร์ฟเวอร์ของโมบายเอเจ้นต์
AgletRMIServer	เป็นอินเตอร์เฟซของคลาสที่ติดต่อกับเอเจ้นต์เซิร์ฟเวอร์
AgletRMIServerImpl	เป็นอิมพลีเมนต์ของ AgletRMIServer มันจะสร้างอินสแตนซ์ของ ServerApp และไบน์ดิงเป็น RMI เซิร์ฟเวอร์

เครื่องหมายเลข 4,5,6 จะเป็นเครื่องเซิร์ฟเวอร์ของโมบายเอเจ้นต์ ซึ่งจะติดต่อฐานข้อมูลซึ่งมีฟิลด์ต่างๆ ดังนี้

id int(6)	เป็นฟิลด์ที่ใช้เก็บรหัสสินค้า
imageLink varchar(50)	เป็นฟิลด์ที่ใช้เก็บลิงค์ของรูป
description varchar(200)	เป็นฟิลด์ที่ใช้เก็บคำบรรยายเกี่ยวกับสินค้า
price float(5)	เป็นฟิลด์ที่ใช้เก็บราคา
quantity int(4)	เป็นฟิลด์ที่ใช้เก็บจำนวนที่เหลือในคลังสินค้า

ตัวเอเจ้นต์ที่ใช้มี

FetchDBParent	เป็นเอเจ้นต์พ่อที่สร้างเอเจ้นต์ลูกชื่อ FetchDBBaby หลายตัวแล้วแต่จำนวน (**รูปแบบ Master-Slave**) เซิร์ฟเวอร์เพื่อจะไปเก็บข้อมูลจากฐานข้อมูลแล้วรวบรวมผลที่ได้ตอบกลับไปยังเซิร์ฟเวอร์
FetchDBBaby	เป็นเอเจ้นต์ลูกที่เกิดมาแล้วก็จะเดินทางไปยังเป้าหมายแล้วรวบรวมข้อมูลส่งเป็นข้อความกลับมาให้เอเจ้นต์พ่อ
UpdateStockParent	เป็นเอเจ้นต์พ่อที่สร้างเอเจ้นต์ลูกชื่อ UpdateStockBaby ตัวเดียว (**รูปแบบ Itinerary**) เพื่อจะเดินทางไปอัพเดทข้อมูลในฐานข้อมูลเพื่อตัดยอดสินค้าในคลังสินค้า
UpdateStockBaby	เป็นเอเจ้นต์ลูกที่เกิดมาแล้วก็จะเดินทางไปยังเซิร์ฟเวอร์ต่างๆเพื่อจะอัพเดทข้อมูลในฐานข้อมูลเพื่อตัดยอดสินค้าในคลังสินค้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จ. การตั้งค่าระบบ และ การทดลอง

1. เริ่มเอเจ้นต์เซิร์ฟเวอร์ และ RMI เซิร์ฟเวอร์
   เมื่อติดตั้งโปรแกรมที่จำเป็นต้องใช้ทั้งหมดแล้ว (ภาคผนวก ก-ง) ก็ให้เริ่มระบบโดยเริ่มจากเครื่อง RMI เซิร์ฟเวอร์ซึ่งมีเอเจ้นต์เซิร์ฟเวอร์หลักติดอยู่ ให้สั่ง

   >SET CLASSPATH=C:/Documents and Settings/BlackJack/jbproject/mobileagent/classes

   >start rmiregistry

   ตัว RMI Registry ก็จะเริ่มทำงานโดยจะต้องตั้งคลาสพาธให้อิงกับที่อยู่ของคลาส Skeleton และ Implementation หลังจากนี้ให้รัน AgletRMIServerImpl

   >java mobileagent.AgletRMIServerImpl

   ตัวเซิร์ฟเวอร์ของเอเจ้นต์และเซิร์ฟเวอร์ของ RMI ก็จะเริ่มทำงานได้ผลดังรูป

```
R0 = tomcat
R1 = changeit
[Warning: The hostname seems not having domain name.
Please try -resolve option to resolve the fully qualified hostname
or use -domain option to manually specify the domain name.]
Server waiting.....
```

   หลังจากนี้เราต้องไปเปิดเอเจ้นต์เซิร์ฟเวอร์เพิ่มอีกโดยเป็นเอเจ้นต์เซิร์ฟเวอร์ที่ติดต่อกับฐานข้อมูลที่เรานำมาทำเป็นคลังสินค้าแบบกระจายของเรา การเริ่มเอเจ้นต์เซิร์ฟเวอร์ Tahiti ดูได้ที่ภาคผนวก ก โดยเครื่องนั้นๆต้องเปิด MySQL ไว้แล้ว

2. เริ่มเว็บเซอร์วิสเซิร์ฟเวอร์ โดยเปิดโปรเจคที่มีไฟล์ OlalaEngine.jws ซึ่งเป็นเว็บเซอร์วิสที่เป็น RMI Client แล้วเปิดเว็บเซอร์วิสเซิร์ฟเวอร์ โดยกด Ctrl + F5 ใน BEA Weblogic Workshop

3. เริ่มเว็บแอพลิเคชั่นเซิร์ฟเวอร์ ในการสร้างเราได้รวมเข้ากับเว็บเซอร์วิสเซิร์ฟเวอร์ ซึ่งเราอาจจะแยกกันก็ได้ เพราะมันได้ติดต่อกันผ่านเครือข่ายอินเตอร์เนตได้อยู่แล้ว เราสามารถเริ่มเว็บแอพลิเคชั่นโดย กด Start ในหน้าของ main.html

4. การเรียกดูหน้ารายการสินค้า ก็สามารถเข้าเว็บเบราเซอร์ใดๆ จากเครื่องใดๆ ก็ได้แล้วกรอก URL http://161.246.6.101:7001/goon/main.html

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] H.M. Deitel and P.J. Deitel : *"Java How To Program 5th Edition"*, "Deitel", 2003

[2] สุวัฒนา สุขสมจินตน์, ผู้แปล : "*คัมภีร์การใช้ XML ฉบับสมบูรณ์*", "ซีเอ็ด", 2002

[3] Marty Hall and Larry Brown : *"Servlets and JavaServer Pages Volumn I Core Technologies 2nd Edition"*, "Prentice Hall", 2004

[4] Danny B. Lange and Mitsuru Oshima : *"Programming and Deploying Aglets Using Java"*, "Addison-Wesley", 2003



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้