การจัดการข้อมูลส่วนบุคคลและกลุ่มบนระบบเครือข่ายคอมพิวเตอร์

WEB ORGANIZER

ฉัตรชัย กองมณี
ชั่งคุณ ศิริเฑียรทอง
ณรงชัย พงศ์สุวากร

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

WEB ORGANIZER

CHATCHAI KONGMANEE
CHANGKUN SIRITIANTHONG
NARONGCHAI PONGSUWAGORN

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2000

หัวข้อปัญหาพิเศษ การจัดการข้อมูลส่วนบุคคลและกลุ่มบนระบบเครือข่ายคอมพิวเตอร์ WEB ORGANIZER

ชื่อนักศึกษา นายฉัตรชัย กองมณี 40056018

นายชั่งคุณ ศิริเฑียรทอง 40056020

นายณรงชัย พงศ์สุวากร 40056025

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2543

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2543

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผู้ช่วยศาสตราจารย์พัชรินทร์ เหมโชติ	
กรรมการ	ดร. นันทิกา เบญจเทพานันท์	
กรรมการและอาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)

หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปัญหาพิเศษ	การจัดการข้อมูลส่วนบุคคลและกลุ่มบนระบบเครือข่ายคอมพิวเตอร์	
ชื่อนักศึกษา	นายฉัตรชัย กองมณี	40056018
	นายชั่งคุณ ศิริเทียรทอง	40056020
	นายณรงชัย พงศ์สุวากร	40056025
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2543	
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์	

บทคัดย่อ

ในปัจจุบันปัญหาหนึ่งทางธุรกิจ คือ การนัดหมายพบปะบุคคลหรือกลุ่มคนที่มีส่วนเกี่ยวข้องในการทำงานร่วมกันซึ่งถ้าหากทราบช่วงเวลานัดหมายบุคคลหรือกลุ่มคนที่ทำงานร่วมกันก็จะเป็นประโยชน์อย่างยิ่ง และหากบุคคลหรือกลุ่มคนสามารถที่จะจัดการตารางนัดหมายของตนเองและของกลุ่มก็จะมีประโยชน์ในการจัดการเวลานัดหมายมากขึ้น ประกอบกับปัจจุบันการติดต่อสื่อทางอินเตอร์เน็ทกำลังเป็นที่นิยมกันอย่างแพร่หลาย โดยการเชื่อมต่อทำได้ง่ายและหลายทาง

ดังนั้นโครงงานปัญหาพิเศษฉบับนี้ได้นำปัญหาทางธุรกิจและการใช้งานอินเตอร์เน็ทเป็นสื่อในการติดต่อมารวมกันพัฒนาเป็นเว็บไซต์ที่ทำงานด้านการจัดการการนัดกมายส่วนบุคคลและกลุ่ม พร้อมทั้งสามารถสนทนาพูดคุยแบบสาธารณะหรือกลุ่ม และสามารถจัดการข้อมูลส่วนบุคคล เช่น อีเมล์ และ สมุดบันทึกส่วนตัว

Special Project Titles	Web Organizer
Students	Mr. Chatchai Kongmanee 40056018
	Mr. Changkun Siritianthong 40056020
	Mr. Narongchai Pongsuwakorn 40056025
Degree	Bachelor Degree of Science
Department	Mathmatics and Computer Sciences, Faculty of Science
Programme	Computer Sciences
Acadamic Year	2000
Special Project Advisor	Assistant Professor Praiboon Pantaragphong

Abstract

Nowadays, There are many problems in business for example an appointment that other people concern about working together. That is useful and they can manage the schedule themselves its very useful for the management.

Today, the communication in Internet is very popular around the world. It's very simple and easy to access. So, this project solve the problem in business and Internet assess for connect and develop to be a web-site that arrange the appointment of themselves. It also communicates in public. This can manage an information such as e-mail and personnel address books.

กิตติกรรมประกาศ

คณะผู้จัดทำปัญหาพิเศษเรื่องการจัดการข้อมูลส่วนบุคคลและกลุ่มบนระบบเครือข่ายคอมพิวเตอร์ ขอขอบพระคุณอาจารย์ทุกๆท่าน ที่ช่วยให้คำแนะนำต่างๆในการทำปัญหาพิเศษ โดยเฉพาะอย่างยิ่ง ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์ ซึ่งเป็นที่ปรึกษาปัญหาพิเศษที่ได้กรุณาให้คำแนะนำในการทำปัญหาพิเศษฉบับนี้แก่ผู้จัดทำ ขอบคุณอาจารย์ทุกท่านที่ได้กรุณาสละเวลาอันมีค่าเพื่อตรวจสอบปัญหาพิเศษฉบับนี้ และขอบคุณบิดามารดาที่เป็นผู้สนับสนุนทางด้านค่าใช้จ่ายต่างๆ เพื่อนๆที่เป็นกำลังใจในการทำปัญหาพิเศษนี้จนกระทั่งสำเร็จลงด้วยดี

คณะผู้จัดทำ

เมษายน 2544

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหาพิเศษ

ข้อมูลส่วนบุคคลและกลุ่มบุคคลที่มีการทำงานร่วมกัน เป็นสิ่งที่จำเป็นต่อการดำเนินงานทางธุรกิจเป็นอย่างมาก ดังนั้นการจัดเก็บข้อมูลและการนำมาใช้ให้เกิดประโยชน์นับว่าเป็นสิ่งที่สำคัญเป็นอย่างยิ่ง โดยเฉพาะถ้ามีการทำงานเป็นลักษณะกลุ่มงานหรือ Project การที่สามารถจัดการกับข้อมูลและติดต่อกับสมาชิกในกลุ่มงาน การกระจายข้อมูลการจัดการตารางเวลาการ นัดหมาย จะเป็นสิ่งที่อำนวยความสะดวกและช่วยให้การดำเนินงานทางธุรกิจเป็นไปอย่างมีประสิทธิภาพ ไม่ว่าจะเป็นระบบจัดการตารางเวลา(schedule and calendar) ระบบการจัดการจดหมายเล็กทรอนิกส์(e-mail) ระบบจัดการข้อมูลส่วนบุคคล(personal information and address book) ระบบการติดต่อผ่านเครือข่าย(chat) และด้วยเทคโนโลยีเครือข่ายอินเตอร์เน็ตที่พัฒนาขึ้นอย่างรวดเร็ว การที่เราสามารถจัดการข้อมูลผ่านระบบเครือข่ายอินเตอร์เน็ตได้ จึงยิ่งเป็นการเพิ่มความสะดวกในการจัดการข้อมูล และ ลดค่าใช้จ่ายในการที่จะต้องจัดซื้อจัดหา อุปกรณ์จัดการข้อมูลส่วนตัว (PDA) เพราะเราสามารถจะติดต่อเครือข่ายอินเตอร์เน็ตได้ง่ายดาย เช่นตาม อินเตอร์เน็ตคาเฟ่ต์

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1. เพื่อทำการพัฒนาระบบการจัดการข้อมูลส่วนบุคคล และ e-mail ผ่านเครือข่ายอินเตอร์เน็ต
2. เพื่อทำการพัฒนาระบบการจัดการตารางเวลา(schedule and calendar) ,ระบบfileข้อมูลและระบบ การ เตือนการนัดหมายหรือแสดงหมายกำหนดการ ทั้งระบบส่วนตัวและลักษณะการทำงานเป็นกลุ่มหรือProject ผ่านระบบเครือข่าย
3. เพื่อทำการพัฒนาระบบการจัดการติดต่อสื่อสารภายในกลุ่ม (chat) ผ่านเครือข่ายอินเตอร์เน็ต
4. เพื่อทำการศึกษาและพัฒนาระบบด้วย Java technology ผ่านเครือข่ายอินเตอร์เน็ต ไม่ว่าจะเป็น Java applet, jsp, Java servlet พร้อมทั้งการจัดการ Web server
5. เพื่อทำการศึกษาและพัฒนาระบบการจัดการฐานข้อมูลผ่าน JDBC (JavaDatabaseConnection) พร้อมทั้งการจัดการ Database server

1.3 ขอบเขตของปัญหาพิเศษ

ในการจัดการศึกษาและพัฒนาระบบการจัดการข้อมูลส่วนบุคคลผ่านระบบเครือข่ายจะประกอบไปด้วย 2 ส่วนหลักในการศึกษาอันประกอบไปด้วย

- front end หรือ ส่วนของ presentation logic ซึ่งเป็นส่วนในการแสดงผลและทำการเข้าใช้งานและจัดการข้อมูล ของผู้ใช้ระบบ อันประกอบไปด้วยส่วนหลัก ๆ คือ การจัดการตารางเวลา(schedule and calendar), ระบบจัดการจดหมายอิเล็กทรอนิกส์(e-mail) ระบบจัดการข้อมูลส่วนบุคคล(personal information and address book), ระบบการติดต่อผ่านเครือข่ายแบบเป็นกลุ่ม(chat) ซึ่งจะติดต่อผ่านทาง Web server
- back end หรือส่วนของ business logic ซึ่งเป็นส่วนของ Database, การจัดการสมาชิกของระบบ, การจัดการ Database server , E-mail server, การพัฒนา logic ในการให้บริการต่าง ๆ รวมถึง ระบบในการสร้างการเตือน(remainder)

1.4 ประโยชน์ที่ความว่าจะได้รับ

1. สามารถนำเว็บไซต์นี้ไปใช้งานได้จริงในระบบธุรกิจ
2. ทำให้มีความสะดวกในการทราบช่วงเวลานัดหมายของบุคคลหรือกลุ่มผู้ที่ต้องการจะนัดหมาย
3. ลดค่าใช้จ่ายในการจัดหาบุคคลที่คอยดูแลเรื่องการนัดหมาย และ ค่าใช้ในการจัดซื้อ วัสดุที่ไว้บันทึกการนัดหมายและข้อมูลต่างๆที่จำเป็น

1.5 ขั้นตอนในการศึกษา

ในการศึกษาได้ทำการจัดการเวลาในการศึกษาออกเป็นส่วน ๆ ดังต่อไปนี้

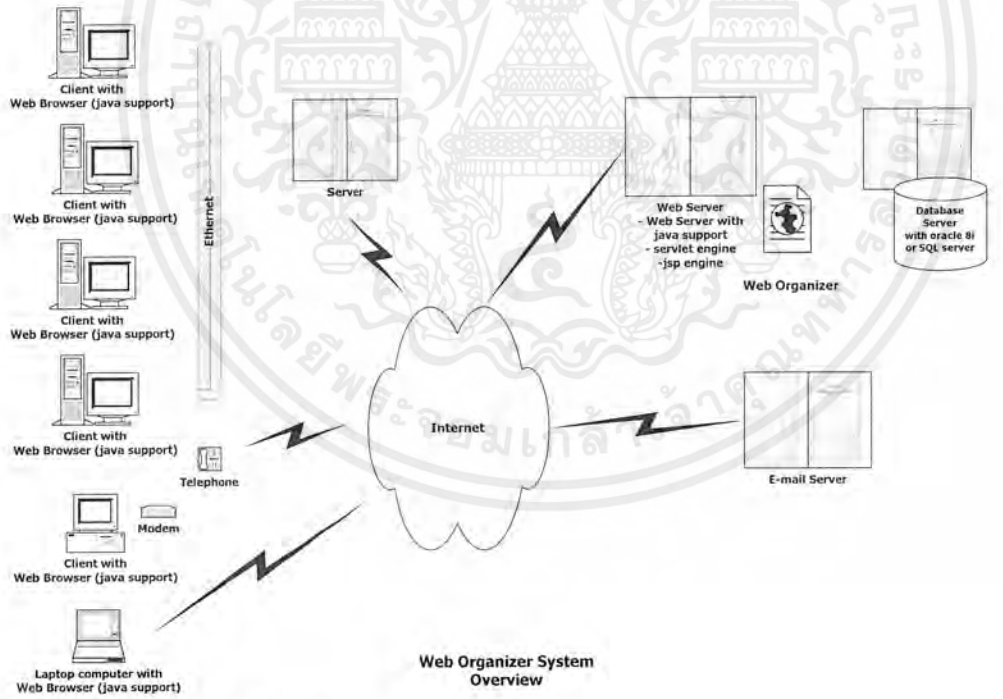
- มิถุนายน ถึง กรกฎาคม ทำการศึกษาและหาข้อมูลของระบบ และ technology ที่จะใช้ในการแก้ปัญหา
- สิงหาคม ถึง กันยายน ทำการออกแบบระบบ, ออกแบบฐานข้อมูล, ศึกษาการพัฒนาระบบ
- ตุลาคม ถึง ธันวาคม ทำการพัฒนาต้นแบบ (prototype) ของระบบ และพัฒนาระบบจริง
- มกราคม ถึง มีนาคม ทดสอบการใช้งานระบบและทำการแก้ไขปรับปรุงระบบพร้อมสรุปผล

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 การทำงานของระบบจัดการข้อมูลส่วนบุคคลและกลุ่มผ่านระบบเครือข่าย

ระบบจัดการข้อมูลส่วนบุคคลผ่านระบบเครือข่ายเป็นระบบจัดการข้อมูลที่ใช้ เทคโนโลยีของเครือข่ายอินเตอร์เน็ต ที่ได้รับความนิยมและแพร่หลายอยู่ในปัจจุบันมาใช้ให้เกิดประโยชน์เพราะรูปแบบของ Application หรือ โปรแกรม ในปัจจุบันซึ่งมีการพัฒนาและสร้างให้อยู่ในรูปแบบที่สามารถทำงานผ่าน Web ได้หรือที่มักเรียกกันว่า Web Application และโดยเฉพาะข้อมูลส่วนบุคคลและกลุ่มนับเป็นสิ่งที่จำเป็นต่อการดำเนินงานทางธุรกิจและชีวิตประจำวันของบุคคลเป็นอย่างมาก ถ้าสามารถนำการจัดการข้อมูลมาใช้ผ่านระบบ อินเตอร์เน็ต ก็เป็นการเพิ่มความสะดวกและลดค่าใช้จ่ายในการจัดการลงไปได้อย่างมาก

เนื่องจากลักษณะของ Web Application จะมีตัวของโปรแกรมที่ใช้ในการทำงานอยู่ภายในส่วนของ Web Server ทำให้ผู้ที่ต้องการใช้โปรแกรมไปต้องทำการติดตั้งโปรแกรมไว้ที่ตัวเครื่อง เป็นการลดค่าใช้จ่ายในการติดตั้งโปรแกรม เพียงแต่มีเพียง Web browser ติดต่อผ่านระบบเครือข่าย ก็จะสามารถใช้

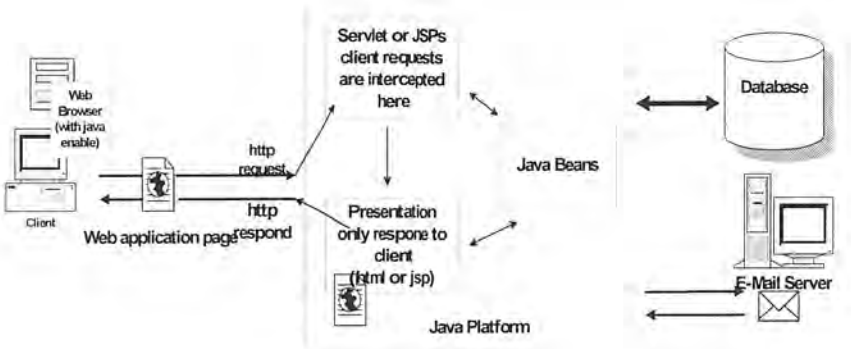


รูปที่ 2.1 ลักษณะโดยรวมของระบบ

งานระบบได้ทันทีซึ่งเหมาะกับลักษณะของการจัดการข้อมูลส่วนบุคคลและกลุ่มในระบบจัดการข้อมูลส่วนบุคคลผ่านระบบเครือข่าย อินเตอร์เน็ตนี้ใช้หลักการการทำงานของระบบผ่านเครือข่ายโดยการทำงานจะทำงานผ่าน Web Browser เพื่อเข้าสู่เครือข่ายอินเตอร์เน็ต และเข้าสู่ตัวของระบบจัดการข้อมูลส่วนบุคคล ดังรูปที่ 2.1 ที่เป็นลักษณะโดยรวมของระบบ จากภาพจะเห็นได้ว่าการเข้าถึงระบบจัดการข้อมูลส่วนบุคคลจะสามารถทำการเข้าถึงระบบได้จากทุกที่ที่สามารถติดต่อเข้ากับเครือข่าย อินเตอร์เน็ตได้ไม่ว่าจะเป็นจากที่ทำงาน ที่บ้าน หรือแม้แต่ผ่านอินเตอร์เน็ตคาเฟ่ หรือ ผ่านอุปกรณ์ไร้สาย ส่วนประกอบของระบบประกอบไปด้วย

1. Client with Web Browser ซึ่งเป็นส่วนของการเข้าใช้งานระบบผ่านทาง Web Browser ซึ่ง Web Browser ที่จะเข้าใช้งานระบบจะต้องสามารถ ใช้งาน Java ได้ โดยจะต้องมี JVM หรือ Java Virtual Machine ไว้สำหรับทำงานกับ applet หรือ แสดงผล GUI ที่เป็น Java component เช่น Java Swing , Java Servlet เป็นต้น
2. เครือข่าย อินเตอร์เน็ต เป็นสื่อกลางในการติดต่อสื่อสารระหว่าง Client กับ ระบบ โดยสามารถทำการติดต่อเครือข่ายอินเตอร์เน็ตผ่าน ISP (Internet Service Provider) หรือผ่านบริการอื่น ๆ เพื่อใช้ในการเข้าใช้ระบบ
3. ตัวระบบจัดการข้อมูลส่วนบุคคล ซึ่งจะประกอบไปด้วยส่วนของ Web Server, Database Server และ E-mail Server ดังมีรายละเอียดในแต่ละส่วนดังต่อไปนี้
   - Web Server ทำหน้าที่ในการรับ request จาก client และส่ง response กลับสู่ client เพื่อทำการให้ บริการแก่เครื่อง client ตามที่ร้องขอมา
   - Database Server ทำหน้าที่ในการ จัดเก็บข้อมูลของสมาชิกและของระบบเพื่อใช้ในการให้บริการ
   - E-mail Server ทำหน้าที่ในการให้บริการจดหมายอิเล็กทรอนิกส์ หรือ e-mail สำหรับใช้ในระบบจัดการข้อมูลส่วนบุคคล

ส่วนประกอบหลักของระบบจะอยู่ที่ส่วนของการให้บริการทางฝั่งของ Server ซึ่งจะประกอบไปด้วย logic ในการให้บริการและจัดการข้อมูลต่าง ๆ ของระบบ ดังรูปที่2.2 จะแสดงรายละเอียดการทำงานของระบบการให้บริการของระบบจัดการข้อมูลส่วนบุคคลผ่านเครือข่ายอินเตอร์เน็ต

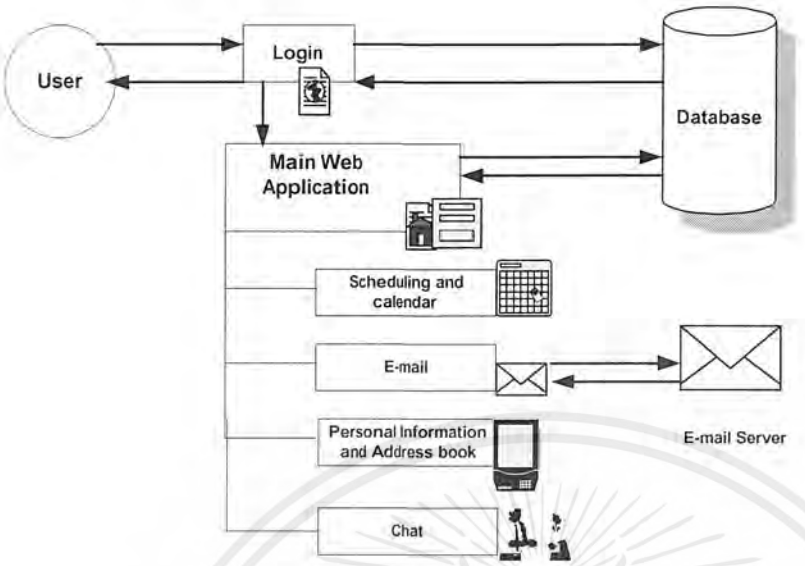


รูปที่ 2.2 การทำงานของระบบ

ในส่วนของการให้บริการหลังจากได้รับการร้องขอบริการ (request) เข้ามาผ่านทางเครือข่าย ด้วย protocol http ด้วยการใช้ Web Browser เมื่อเครื่อง Web Browser ได้รับการร้องขอก็จะทำการให้บริการ โดยส่วนของการรับการร้องขอ จะอยู่ในรูปของ Java Servlet หรือ JSP (JavaServer Page) โดยตัวระบบจัดการฐานข้อมูลส่วนบุคคลได้จัดการพัฒนาขึ้นภายใต้ภาษา Java หรือ อยู่ในรูปแบบของ Java Platform จึงมีการจัดการให้บริการของระบบดังต่อไปนี้คือ

1. HTTP service ของ Web Server จะทำการรับการร้องของของ client และทำการเรียกใช้ Java Servlet หรือ JSP ที่ client ร้องขอมา โดยตัวของ Java Servlet และ JSP จะเป็นส่วนของ logic ในการให้บริการ และติดต่อกับฐานข้อมูล หรือ e-mail
2. Java Servlet และ JSP สามารถทำการติดต่อกับฐานข้อมูล และ e-mail ได้โดยตรงหรืออาจจะทำการติดต่อกับ Java Bean ซึ่งจะเป็นโปรแกรมขนาดเล็กที่ทำงานเฉพาะอย่างในการให้ Java Bean ทำการติดต่อกับฐานข้อมูล หรือ ให้ Java Bean ทำการไปดึง e-mail จาก E-mail Server แล้วส่งข้อมูลกลับมาให้ Java Servlet หรือ JSP
3. เมื่อตัว Java Servlet และ JSP ทำการประมวลผลเสร็จแล้วก็จะทำการแสดงผลกลับมาโดยจะแสดงผลมาอยู่ในรูปของ เอกสาร HTML หรือ เป็น JSP เพื่อส่งกลับมาให้ client เพื่อแสดงผลผ่าน Web Browser
4. ในการติดต่อฐานข้อมูล ภายใต้ระบบจัดการข้อมูลส่วนบุคคลและกลุ่มผ่านเครือข่ายจะพัฒนาภายใต้ Java Platform จะใช้ JDBC (Java Database Connectivity) ในการติดต่อกับฐานข้อมูล

5. ในการติดต่อ เพื่อให้บริการ e-mail จะทำงานผ่าน Javamail API



รูปที่ 2.3 แสดงส่วนประกอบของระบบ

ในการทำงานของระบบจัดการข้อมูลส่วนบุคคลและกลุ่มผ่านเครือข่ายประกอบไปด้วยการให้บริการหลัก ๆ ดังภาพ

- *Scheduling and Calendar* เป็นส่วนที่จัดการเกี่ยวกับ ตารางเวลา การจัดการหมายนัด ปฏิทิน ซึ่งจะประกอบไปด้วยการจัดการการนัดหมายส่วนบุคคลและกลุ่ม หรือ การนัดหมายกับกลุ่มงาน (Appointment) การสร้างหมายเตือน หมายนัดกับผู้อื่น, การจัดการเหตุการณ์พิเศษ(Event) หรือ เป็นการจดจำเหตุการณ์ที่จะเกิดขึ้นในวันนั้น เช่นวันเกิด วัดครบรอบ, การจัดการตารางงานของบุคคลหรือของกลุ่ม อาจจะเป็นการจัดตามลำดับความสำคัญหรือดูว่าจะต้องทำอะไรก่อนหลัง(Task) อาจจะเป็นงานของโปรเจ็กต์ที่มีระยะเวลาการดำเนินการยาวนาน โดยแสดงในรูปของปฏิทิน และตารางเวลาของแต่ละวัน มีการแสดง Appointment, Event, Task พร้อมกับแสดงการเตือน หรือ แสดงหมายนัดเมื่อตรงตามเวลาที่ตั้งเอาไว้ โดยสามารถส่งหมายนัดหรือขอการนัดหมายกับ สมาชิกคนอื่น หรือ กับสมาชิกในกลุ่ม

- *E-mail* เป็นส่วนของการจัดการจดหมายอิเล็กทรอนิกส์ หรือ e-mail โดยจะประกอบไปด้วยการจัดการในส่วนของการรับและส่ง e-mail โดยทำการ forward mail จาก E-mail Server มาแสดง พร้อมทั้ง menu ในการจัดการ ทั้งการรับ attach file การลบจดหมาย การจัดการ folder การส่ง e-mail

โดยมีหน้าจอในการใช้เขียนจดหมาย พร้อมทั้งมีการเพิ่มความสะดวกในการส่งภายในกลุ่ม หรือส่งผ่านรหัสสมาชิก

- *Personal Information and Address Book* เป็นส่วนในการจัดการข้อมูลส่วนบุคคล อาทิ เช่น ชื่อ, ที่อยู่, ตำแหน่ง, วันเกิด, เบอร์โทรศัพท์, ส่วนตัว พร้อมกับ address book สำหรับจดบันทึก ชื่อ, ที่อยู่, เบอร์โทรศัพท์, บริษัท, ตำแหน่ง, web, address, e-mail address ของบุคคลหรือองค์กรที่ต้องการบันทึกเอาไว้

- *Chat* เป็นส่วนของการติดต่อสื่อสารระหว่างภายในกลุ่มผ่านเครือข่าย ด้วยโปรแกรม chat โดยที่สามารถติดต่อได้เฉพาะกับภายในกลุ่มที่ทำการสร้างเอาไว้เท่านั้น จึงเหมาะสำหรับการติดต่อสื่อสารภายในกลุ่มเพื่อปรึกษาเกี่ยวกับ project

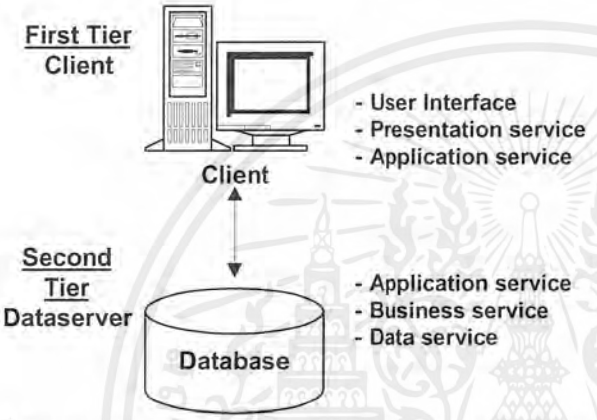
โดยที่ส่วนประกอบทั้งหมดจะรองรับงานทั้งในส่วนของบุคคล และถ้าสมาชิกมีการสร้าง กลุ่มงานขึ้นมากก็สามารถใช้งานในส่วนของ กลุ่มได้อีกด้วย เช่น สามารถเรียกดู ตารางเวลาของกลุ่มได้ (แก้ไขไม่ได้ยกเว้นหัวหน้างานหรือผู้ที่สร้าง Group), ส่ง e-mail ให้กับทุกคนในกลุ่มโดยเพียงเลือกกลุ่มที่ต้องการส่ง, สามารถใช้งาน file ที่ สมาชิกในกลุ่มได้ทำการ share file ไว้ เป็นต้น โดยในการสร้าง group จะต้องมีหัวหน้ากลุ่มหรือผู้ที่ทำการสร้างกลุ่มขึ้น และ add สมาชิกเข้ามาในกลุ่มได้ตามที่ต้องการ

2.2 ข้อมูลเกี่ยวกับ Software และ ทฤษฎีที่เกี่ยวข้อง

2.2.1 โมเดลชนิด Tier

สถาปัตยกรรมของแอปพลิเคชั่นแบบClient/Serverแบ่งการประมวลผลเป็น2โปรแกรม โดยทั่วไปจะจะทำงานบนเครื่องขึ้นไป แอปพลิเคชั่นที่ทำงานกับฐานข้อมูลแบบClient/Server เครื่องServerจะเก็บรวบรวมข้อมูลไว้ ส่วนเครื่องclientประมวลผลข้อมูลที่ได้มาหรือสร้างเป็นข้อมูลใหม่ วิธีทำงานโดยใช้สถาปัตยกรรมแบบClient/Serverนี้ทำให้สามารถติดต่อใช้งานข้อมูลได้จากผู้ใช้หลายแห่ง

2.2.1.1 การทำงาน Client/Server แบบTwo-Tier



รูปที่ 2.4 แสดงการทำงานของ Client/Server แบบ Two-Tier

รูปแบบธรรมดาทั่วไปของสถาปัตยกรรม Client/Server เป็นTwo-Tier ซึ่งมาจากการแบ่งการทำงานของแอปพลิเคชั่นออกเป็นส่วนClient/Serverยอมให้มีการติดต่อจากหลายๆที่เข้ากับส่วนให้บริการซึ่งข้อมูลไว้ ส่วนการแสดงผลอยู่ที่Client และส่วนเก็บรวบรวมข้อมูลจะอยู่ที่Serverแอปพลิเคชั่นทั่วไป ส่วนใหญ่บน internet เช่น e-mail, telnet, ftp, gopher หรือ web เป็นแอปพลิเคชั่นแบบ 2 ระดับ ซึ่งทำงานโดยไม่ต้องประมวลผลข้อมูลขนาดใหญ่ โดยทั่วไปจะทำงานติดต่อใช้ข้อมูลภายใน internet

ข้อดีของแอปพลิเคชั่นแบบ Two-Tier

เป็นแอปพลิเคชั่นง่ายๆธรรมดาที่ไม่ต้องการการบำรุงรักษามาก สามารถพิจารณาเลือกใช้ว่าเหมาะกับแอปพลิเคชั่นแบบTwo-Tierหรือไม่ควรขึ้นกับเงื่อนไขดังนี้

- เป็นแอปพลิเคชั่นที่ใช้ฐานข้อมูลเดียว
- ฐานข้อมูลบรรจุอยู่ภายในCPUเครื่องเดียว
- ฐานข้อมูลมีขนาดเดิม ไม่เปลี่ยนแปลงบ่อยๆ

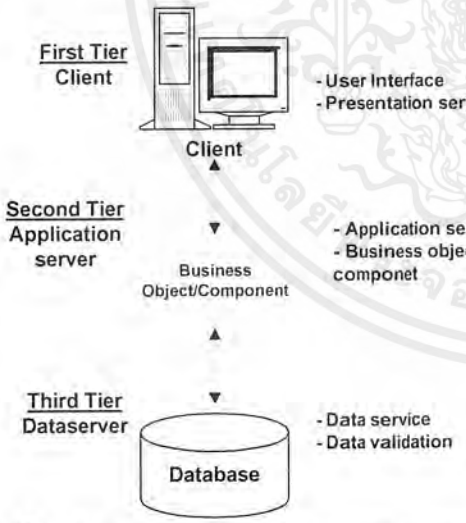
- user baseไม่มีการเปลี่ยนแปลงบ่อย
- requirement ไม่มีการเปลี่ยนแปลงหรือเปลี่ยนแปลงน้อยมาก
- แอปพลิเคชั่นที่สมบูรณ์แล้ว ไม่จำเป็นต้องดูแลบำรุงรักษามาก

ข้อเสียของแอปพลิเคชั่นแบบTwo-Tier

ความต้องการของผู้ใช้เพิ่มมากขึ้น ดังนั้นความซับซ้อนของแอปพลิเคชั่นจึงต้องมากตามไปด้วย จากการที่Client มีประสิทธิภาพและมีความซับซ้อนขึ้นเรื่อยๆ ในขณะที่serverมีขนาดเล็กลงเพื่อให้ราคาถูกลง และความสามารถในการจัดการกับฐานข้อมูลที่ซับซ้อนต่ำลง เช่น ปัจจุบันเครื่องคอมพิวเตอร์เมนเฟรมได้ถูกเปลี่ยนมาใช้เครื่แงคอมพิวเตอร์ขนาดเล็กจำนวนมากมาทำงานแทน และงานบางส่วนจะถูกผลักภาระไปที่เครื่องclient เพื่อเป็นการลดค่าใช้จ่าย แต่การทำเช่นนี้ทำให้เกิดปัญหา “fat client”

Client ที่มีปัญหา fat client นี่เกิดจากการที่clientไม่สามารถรองรับขนาดของข้อมูลและงานของผู้ใช้ที่มีจำนวนมากขึ้นได้เพราะว่างานของclientไม่ได้มีแค่แสดงข้อมูลให้เห็นเท่านั้น แต่ยังมีการดึงข้อมูลอื่นๆจำนวนมากที่ไม่กี่ยวข้องเลยกับงานนั้นๆ มาด้วยและในกรณีที่การเปลี่ยนแปลงฟังก์ชันการทำงานบางส่วน ผู้ใช้จำเป็นต้องมีการเปลี่ยนแปลง ทดสอบและแจกจ่ายโปรแกรมในส่วนของclientที่ได้ปรับปรุงแล้วไปยังclientทุกเครื่อง

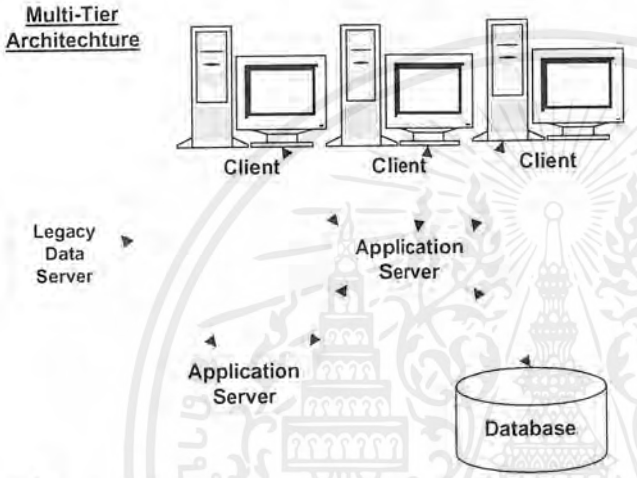
2.2.1.2 การทำงาน Client/Server แบบ Three-Tier



รูปที่ 2.5 แสดง การทำงาน Client/Server แบบ Three-Tier

เพื่อแก้ปัญหาของtwo-tier จึงเพิ่มจาก 2 Tier เป็น3 Tier โดยในแบบTwo-Tier เดิม client จะติดต่อโดยตรงกับฐานข้อมูล หากมีการเปลี่ยนแปลงใดๆเกิดขึ้นในฐานข้อมูล การแสดงผลทางด้าน client จำเป็นต้องเปลี่ยนแปลงไปด้วย ในการแก้ปัญหานี้จะเพิ่มTierใหม่เข้ามาขั้นระหว่างclientและ Server โดย Clientจะติดต่อกับserverโดยผ่านทางobjectที่อยู่บน Middle-Tier จากนั้น Middle-Tier จะติดต่อกับ Server โดย Client จะเห็นเฉพาะobjectใน Middle-Tier เท่านั้น การเปลี่ยนแปลงใดๆจะต้องทำผ่าน Middle-Tier เท่านั้น

2.2.1.3 การทำงาน Client/Server แบบ Multi-Tier (Multi-Tired Application)



รูปที่ 2.6 แสดง Client/Server แบบ Multi-Tier

โปรแกรมแอปพลิเคชั่นโดยทั่วไปที่ใช้งานอยู่ จะประกอบด้วยส่วนที่ติดต่อกับผู้ใช้(User Interface) สำหรับแสดงผลและเก็บรวบรวมข้อมูลเข้ามา กลุ่มของ function ต่างๆที่ทำหน้าที่ที่ประมวลผลข้อมูลและแบ่งงานต่างๆรวมถึงวิธีการเก็บรักษาข้อมูล ถึงแม้ว่าfunctionที่ใช้ในการเก็บรักษาข้อมูลโดยทั่วไปจะทำงานอยู่ภายใต้ Serverของฐานข้อมูลส่วนกลาง บางครั้งเรียกรูปแบบลักษณะการทำงานแบบนี้ว่าเป็นโมเดลแอปพลิเคชั่นแบบ2 ระดับ (Two-Tier application model) ซึ่งโปรแกรมแอปพลิเคชั่นแบบเก่าจะเป็นโปรแกรมเดียวซึ่งทำงานบนเครื่องของผู้ใช้ เนื่องจากโปรแกรมแอปพลิเคชั่นที่ทำงานเดี่ยวๆนั้นมีขนาดใหญ่มาก จึงพัฒนาได้ช้าและบำรุงรักษามาก อีกทั้งยังใช้เนื้อที่ฮาร์ดดิสก์สูงมาก เพียงแค่มีการเปลี่ยนแปลงเล็กน้อยก็จะต้องมีการเขียนโปรแกรมทับลงไปและคอมไพล์อีกครั้ง และเนื่องจากโปรแกรมแอปพลิเคชั่นเหล่านี้เขียนขึ้นมาเพื่อใช้งานกับระบบที่มีลักษณะต่างกันจึงไม่สามารถที่จะเปลี่ยนไปใช้งานบนระบบที่แตกต่างไปได้ วิธีการแก้ปัญหาดังกล่าวมาทำได้โดยการแบ่งโปรแกรมแอปพลิเคชั่นเดี่ยวๆนี้ออกเป็นโมดูลย่อยๆที่ทำงานร่วมกัน การแยกส่วนที่ติดต่อกับผู้ใช้ออกมาจาก

function อื่นๆในโปรแกรมแอปพลิเคชั่น ทำให้สามารถสร้างโปรแกรมไคลเอนด์ (client) เล็กๆ ซึ่งไม่ซับซ้อนและไม่ต้องทำงานมากเกินไปบนเครื่องของผู้ใช้ โดยในโมดูลนี้จะใช้ฮาร์ดดิสก์บนเครื่องผู้ใช้น้อยกว่าและสามารถพัฒนา บำรุงรักษาได้ง่ายกว่า ตัวอย่างเช่นส่วนที่ติดต่อกับผู้ใช้สามารถเปลี่ยนแปลงได้โดยไม่ต้องเขียนโปรแกรมแอปพลิเคชั่นใหม่ทั้งหมดในทางทฤษฎีแล้ว ส่วนโมดูลที่ติดต่อกับผู้ใช้นี้สามารถเขียนได้โดยใช้ภาษาที่ต่างกัน เช่น จาวา(Java) จึงทำให้ใช้ได้บนเครื่องที่แตกต่างกัน ไคลเอนด์โมดูลออกแบบโดยใช้Java applet จึงไม่ต้องการเนื้อที่บนฮาร์ดดิสก์เพื่อการติดตั้ง Java IDL สนับสนุนซอฟท์แวร์ที่ออกแบบเป็นโมดูล แต่ละโมดูลออกแบบให้รองรับแอปพลิเคชั่นได้หลายแบบ แบ่งลักษณะของโมดูลดังกล่าวได้3ประเภทใหญ่ๆ คือ

1) User Interface(Client) Tier ส่วนที่ติดต่อกับผู้ใช้ในโมเดลแอปพลิเคชั่นแบบหลายระดับจะรวมไปถึงส่วนที่ติดต่อกับผู้ใช้แบบกราฟิก(GUI-Graphics User Interface) สำหรับแอปพลิเคชั่นทั้งแบบดั้งเดิมและแบบพื้นฐานสร้างมาเพื่อทำงานกับผู้ใช้ได้เร็วและได้ผลลัพธ์ที่ถูกต้อง โดยรวมfunctionที่ทำงานเกี่ยวกับGUIออกมาส่วนที่ให้บริการทางเครือข่าย การทำงานได้สอดคล้องกันของส่วนที่ติดต่อกับผู้ใช้ จะช่วยลดปัญหาในการเรียนรู้เพื่อใช้งานแอปพลิเคชั่นใหม่ๆทำงานร่วมกันกับแอปพลิเคชั่นได้ดีและให้ผลลัพธ์ที่มีคุณภาพสูงขึ้น แอปพลิเคชั่นแบบGUIนี้สามารถใช้ได้สำหรับงานทั่วไปของผู้ใช้ เช่น บนเครื่องในระบบเครือข่ายหรือบนinternet

2) Server Tier ส่วนที่ให้บริการหรือส่วนServerนี้เป็นส่วนสำคัญของแอปพลิเคชั่นเป็นส่วนกลาง ซึ่งคอยให้บริการการใช้แอปพลิเคชั่นและการสร้างแอปพลิเคชั่น ซึ่งการให้บริการนี้มีอยู่ในเครือข่ายและสามารถเข้าใช้ได้จากแอปพลิเคชั่นทุกระดับ

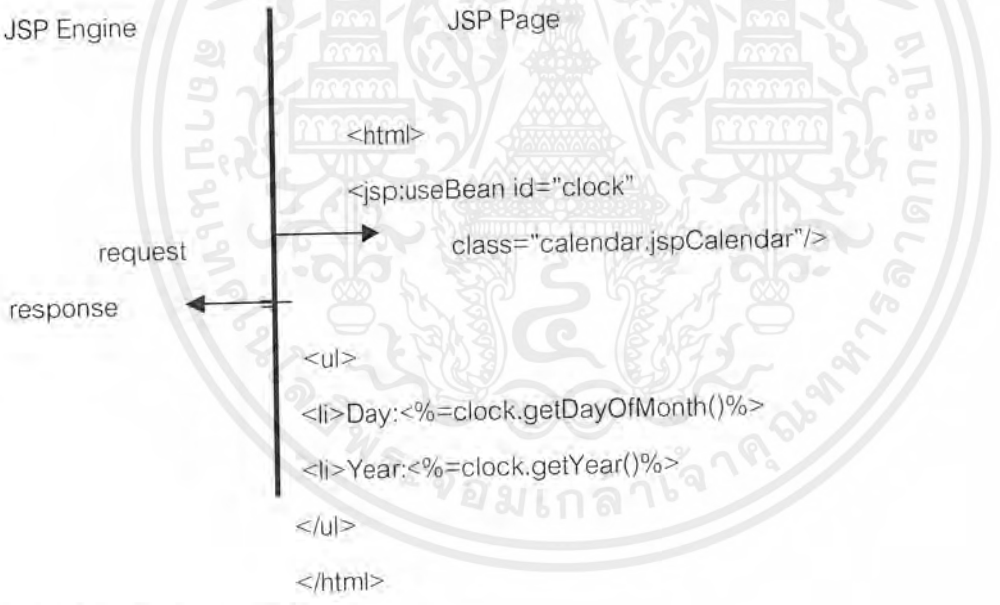
3) Data Store(Database)Tier แอปพลิเคชั่นแบบหลายระดับ(Multi-Tier)นี้จะแยกการติดต่อเข้าใช้ข้อมูลออกมาจากส่วน Server เรียกส่วนที่แยกออกมานี้ว่า"data store tier"มีหลายแบบที่ใช้เก็บและติดต่อใช้ข้อมูล เพื่อช่วยให้ผู้พิจารณาสามารถเลือกกลุ่มข้อมูลที่มีความสำคัญเป็นอันดับแรกสุด Objectที่สร้างขึ้นมาจะใช้เพื่อสนองความต้องการในการใช้ข้อมูลต่างๆประกอบด้วยความสามารถในการเรียกใช้ข้อมูลในRDBMS (Relational Database Management System) หรือ OODBMS (Object Oriented Database Management Systems)

2.2.2 JSP (JavaServer Pages)

JSP หรือ JavaServer Pages เป็น Java platform technology สำหรับสร้าง dynamic Web content application อย่างเช่น HTML, DHTML, XHTML หรือ XML JavaServer Pages เป็นเทคโนโลยีที่ทำให้สามารถสร้าง Web pages แบบ dynamic content ได้อย่างง่ายดาย แต่มีประสิทธิภาพในการทำงานและยืดหยุ่นสูง ซึ่งปัจจุบัน เป็นเวอร์ชัน 1.2

2.2.2.1 JSP คืออะไร

JSP page เป็น text-based document ซึ่งจะทำการอธิบายการประมวลผล หรือ process ขอ request (การร้องขอ) และทำการสร้าง response (การตอบสนอง) ซึ่งลักษณะของเอกสารจะประกอบไปด้วย ข้อมูล ซึ่งจะถูกดำเนินการในลักษณะ dynamic action ด้วย code ในรูปแบบ Java platform จาก ตัวอย่างด้านล่าง เป็นตัวอย่างของ JSP page เป็นการแสดง response page ซึ่งทำการแสดง วัน, เดือน,ปี ขณะนั้น เมื่อได้รับ request ตัว page เองประกอบไปด้วย fixed template text และ องค์ประกอบของ JSP ซึ่งแสดงด้วยการขีดเส้นใต้ ในบรรทัดที่ 2 เป็นการ create Bean ชื่อ clock ซึ่งเป็นชนิด calendar.jspCalendar และ อีก 2 บรรทัดเป็นการแสดงผลของ properties ของ Bean ตัวนี้



รูปที่ 2.7 ตัวอย่างของ JSP page

JavaServer Page เป็น Standard Extension ซึ่งถูกกำหนดไว้ที่ Servlet Standard Extension โดย JSP 1.0 จะใช้ class จาก Java Servlet 2.1 โดยทั้ง JSP 1.0 และ Servlet 2.1 จะทำงานภายใต้

Java Runtime Environment 1.1 (JRE 1.1) และยัง compatible กับ Java 2 Runtime Environment (J2RE)

2.2.2.2 JSP Model

JSP page จะถูก executed ด้วย JSP engine ซึ่งจะถูกติดตั้งไว้ที่ Web server หรือ Application Server ตัว JSP engine จะให้บริการโดย รับ request จาก client ไปให้ JSP page และ response จาก JSP page ไปสู่ client รูปแบบการตีความของ JSP จะคล้ายกับ servlet JSP page จะอธิบายการสร้าง response object จาก request object สำหรับรูปแบบ protocol ที่จะใช้ หรืออาจจะมีการสร้างหรือใช้ process จาก object อื่น ๆ

ทุกๆ JSP engine จะต้องรองรับ protocol HTTP สำหรับใช้ใน request และ response แต่ตัว engine สามารถที่จะรองรับ protocol สำหรับ request/response อื่นๆ ด้วยก็ได้ โดยปกติ request และ response object จะเป็น object ชนิด HttpServletRequest และ HttpServletResponse

JSP page อาจจะมีการกำหนดการรับเหตุการณ์ที่อยู่ในรูป handled โดยใน JSP 1.0 จะมีเพียง event init และ destroy โดยเมื่อครั้งแรกที่ request ได้รับบริการจาก JSP page ด้วย jspInit() method โดยปกติจะถูกเรียกเพื่อเตรียม page ซึ่ง JSP engine สามารถทำการเรียกคืนทรัพยากร (resource) ที่ถูกใช้โดย JSP page ในขณะที่ไม่ได้ให้บริการ request โดยการเรียกใช้ jspDestroy() method ซึ่งลักษณะดังกล่าวนี้จะคล้ายกับ life-cycle ของ servlet

JSP page จะถูก implemented โดยการใช้ JSP translation phase ซึ่ง จะทำเพียงครั้งเดียว ซึ่งจะตามด้วย request processing phase ซึ่งจะทำครั้งหนึ่งต่อ request translation phase จะทำสร้าง class ซึ่งจะทำการ implements javax.servlet.Servlet interface การทำการ translation JSP source page ไปสู่ Java implementation class file สำหรับตอบสนองด้วย JSP engine สามารถ กระทำขณะเวลาระหว่าง initial JSP page ไปสู่ runtime environment ของ JSP engine และทำการรับ และ processing client request สำหรับการสร้าง JSP page เพื่อทำการตอบสนอง

ตัว JSP page จะประกอบไปด้วยส่วนของการ declarations, fixed template data, action instance และส่วนของ script เมื่อ request ทำการร้องของ JSP page ทุกๆ ส่วนจะถูกใช้สร้าง response object ซึ่งจะถูกส่งกลับไปให้ client สิ่งที่ถือว่าเป็นส่วนที่สำคัญของ response object ก็คือ result stream

2.2.2.3 Object and Scopes

JSP page สามารถทำการสร้างและ access Java object เมื่อทำการ processing request จาก JSP Specification ได้กำหนดไว้ว่าสามารถทำการ สร้าง object ภายใน page ได้ด้วยวิธี

directive และ ใช้object ที่ถูกสร้างจากภายนอก ผ่านทาง action นั่นก็คือ object สามารถสร้างโดยตรงด้วยการใช้ scripting code ด้วยคำสั่งเพียงไม่กี่คำสั่ง โดยการสร้าง object จะมี scope attribute เป็นตัวกำหนด ขอบเขตในการอ้างอิง object และ บอกว่าเมื่อใด การอ้างอิงจะถูก ยกเลิก (removed)

การสร้าง object สามารถทำได้โดยตรงผ่าน scripting element ด้วยการกำหนด scripting-level variables ในการกำหนด action และ declatation, การให้ความหมาย, การกำหนด object, การกำหนด scope attribute สามารถทำผ่าน scripting elementsObject ที่ถูกสร้างภายใน JSP page ไว้สำหรับตอบสนอง request object ซึ่งจะมีการกำหนด scope ไว้ดังต่อไปนี้

- page Object ที่กำหนด scope เป็นแบบ page จะสามารถ access ได้เพียงภายใน page ที่ได้ทำการ create เอาไว้เท่านั้น ทุกการอ้างอิงถึง object จะถูกยกเลิก หลังจาก response ได้ทำการส่งกลับไปให้ client จาก JSP page หรือ การ request ถูกส่งต่อไปที่อื่น การอ้างอิงถึง object ด้วย page scope จะถูกเก็บเอาไว้ใน pagecontext object
- request Object ที่กำหนด scope เป็นแบบ request จะสามารถ access จาก pages ที่ทำการ processing ด้วย request เดียวกันทุกที่ที่ถูก create การอ้างอิง object จะถูกยกเลิกหลังจาก request ถูก process โดยเฉพาะถ้า request ถูก forward ไปให้ resource ใน runtime เดียวกัน object จะยังสามารถอ้างอิงถึงได้ การอ้างอิงถึง object ด้วย request scope จะถูกเก็บเอาไว้ใน request object
- session Object ที่กำหนด scope เป็นแบบ session จะสามารถ access จาก page ที่กำลัง processing request ซึ่งอยู่ใน session เดียวกันกับที่ ถูก create ขึ้น มันจะไม่ถูกต้องนักหากทำการ กำหนด object เป็นแบบ session scope จากภายใน page ที่ไม่เป็น session อยู่แล้ว (session-aware) การอ้างอิง object จะถูกยกเลิกหลังจาก จบ session การอ้างอิง object ด้วย session scope จะถูกเก็บไว้ใน session object ซึ่งจะทำงานร่วมกับ page activation
- application Object ที่กำหนด scope เป็นแบบ application จะสามารถ access จาก page ที่กำลัง processing request ซึ่งอยู่ภายใน application เดียวกันกับที่ถูก create ขึ้น การอ้างอิง object จะถูกยกเลิกเมื่อ runtime environment เรียกคืน ServletContext Object ที่กำหนด scope แบบ application จะสามารถกำหนด จาก pages ซึ่ง ไม่เป็น session-aware การอ้างอิง object ที่กำหนด scope เป็น application จะถูกจัดเก็บใน application object ซึ่งจะทำงานร่วมกับ page activationโดยใช้ชื่อในการอ้างอิง a unique object ณ ทุกจุดในการ execution ใน scope ที่แตกต่างกันจะมีการประพฤติตัวตาม scope ของตนภายใต้ single name space

2.2.2.4 Fixed Template Data

Fixed template data จะใช้ในการอธิบาย ชิ้นส่วน ซึ่งจะถูกใช้เป็นคำพูด ในการ response หรือเป็น input สำหรับ JSP actions ตัวอย่างเช่น ถ้าใน JSP page มีการแสดงผลในรูป ของ HTML เป็น list ซึ่งจะบอกว่า หนังสือเล่นใดตรงตามเงื่อนไขบ้าง template data อาจจะประกอบไปด้วย <ul>, , The following book…. เป็นต้น ซึ่ง fixed template data จะเป็นการแสดงข้อมูลที่ไม่เปลี่ยนแปลง ในรูปแบบของ output stream (ซึ่งสามารถอ้างอิง ด้วย variable out) ของ response ไปสู่ client ที่ทำการ request

2.2.2.5 Directives and Actions

องค์ประกอบของ JSP elements สามารถเป็น directive หรือ action ตัว Directive จะจัดการข้อมูลทั่ว ๆ ไป ซึ่งโดย concept ของการทำงานจะมีลักษณะเป็นอิสระขึ้นอยู่กับ request ที่ได้รับของ JSP page ตัวอย่างเช่น directive สามารถจะใช้ การเขียน scripting language ใน JSP page ส่วน Action ส่วนใหญ่จะเป็นส่วนที่ไว้สำหรับลงรายละเอียดสำหรับการให้บริการ request ที่ได้รับของ JSP page ถ้า JSP ได้ทำการ implement โดยการใช้ compiler หรือ translator ส่วน directive สามารถจะจัดการข้อมูล สำหรับ ขั้นตอนการ copilation / translation phase ในขณะที่ action จะเป็นข้อมูลสำหรับจัดลำดับการทำงานของ request processing phase ส่วน action อาจจะมีการสร้าง object และอาจจะทำการใช้งาน script ผ่าน scripting-specific variable Directive element จะอยู่ในรูปของ syntax

<%@ directive…%>

ซึ่งเป็นรูปแบบของ syntax สมัยใหม่ซึ่งทำตามรูปแบบของ XML syntax

ส่วนของ action element จะอยู่ในรูปแบบของ XML syntax คือจะมี start tag, body และ end tag

```
<mytag attr1="attribute value"…>
body
</mytag>
```

หรืออยู่ในลักษณะของ empty tag

```
<mytab attr1="attribute value".../>
```

ใน JSP element จะมี element type ไว้สำหรับอธิบายตัว tag name, attribute และ ความหมาย ของ tag ซึ่งเราจะอ้างอิงถึง type ของ tag ผ่านทาง tagname

2.2.2.6 Tag Extension Mechanism

ในอนาคตของ JSP จะมีการใช้งาน Tag Extension ซึ่งจะมีการอนุญาติให้ทำการเพิ่ม directive และ action ใหม่ ๆ ได้ ซึ่งทำให้ JSP เป็นภาษาที่ง่ายต่อการขยายความสามารถในเรื่อง portable ตัวอย่างเช่นการเพิ่มเติมความสามารถในเรื่อง database query

JSP จะมี tag library ซึ่งจะกำหนด JSP element type และ Customizer ไว้สำหรับใช้งานกับ element ในการควบคุมในขณะทำการ design ใน page composition tools ตัว Tag library สามารถนำมาใช้ใน JSP authoring tools และสามารถใช้ได้ตลอดทั่วทั้งเอกสาร JSP page ในขณะ runtime ที่ Web และ Application server ซึ่งตัว Tag Extension จะถูกรวมไว้ใน Java Runtime Environment Action และ directive เบื้องต้น จะกำหนดการใช้งาน Tag Extension mechanism ตาม semantic model ของ JSP model

2.2.2.7 Scripting Language

Scripting element จะถูกใช้ในการจัดการ object และ สำหรับสร้าง content ในการจัดการ ด้านการคำนวณ ในเอกสาร ส่วนประกอบของ scripting element จะประกอบไปด้วย declaration, scriptlets และ expression Declaration จะใช้ในการกำหนดค่าเบื้องต้น หรือการ declare โครงสร้างของภาษา script ซึ่งจะทำการใช้ใน scripting element Scriptlets จะใช้ในการอธิบายการทำงานเพื่อทำการตอบสนอง request ตัว Scriptlets จะใช้ในการสร้าง logic ของโปรแกรมซึ่งจะใช้สร้างเงื่อนไขในการทำงานหรือขั้นตอนการทำงานของ JSP page ส่วน Expression ก็จะเป็นส่วนของการแสดงค่าของ scripting language ซึ่งได้จากการอ่านค่าของตัวแปรออกมาในขณะ response ซึ่งโดยปกติผลลัพธ์ จะถูกแปลงอยู่ในรูป String และจะถูกจัดส่งในรูป output stream ทุก ๆ JSP engine จึงต้องรองรับ scripting elements ของ ภาษาจาวา และอาจมีการรองรับภาษา script อื่น ๆ ซึ่งจะต้องรองรับในเรื่อง

- การจัดการ Java object
- การเรียกใช้ method ของ Java object
- การดักจับ Java exception

การตีความของ script จะใช้พื้นฐานของภาษา Java ซึ่งความหมายของภาษาอื่น ๆ ยังไม่มีกำหนดไว้ JSP 1.0 ซึ่งแสดงถึงความสามารถในการทำงานข้าม platform ยังไม่สามารถรับประกันได้ ซึ่งจะมีการเพิ่มความสามารถใน version ต่อไป

2.2.2.8 Object and Variables

การ access object สามารถทำได้ด้วยการ code คำสั่งในส่วนของ scripting element ผ่าน scripting language variable การกำหนด scripting variable สามารถทำได้ 2 ที่ อันแรก เมื่อเริ่ม

ใน start tag และ หลังจาก end tag variable จะได้รับการทำงานเมื่อทำการ process request เมื่ออ้างอิงถึง object ที่กำหนดใน element ซึ่งจะอยู่ภายใต้การจัดสรรของ scope ของ object element type จะถูกกำหนดลงไปบนชื่อ และ ชนิดของ variable รวมไปถึงรายละเอียดบนชื่อของ variable ที่ทำการใช้ด้วย Scripting Language Scripting Language อาจจะมีการใช้งาน object ต่างกันในแต่ละภาษา อย่างเช่นใน JavaBeans เราจะสามารถ เข้าถึง properties ได้ผ่านทาง getter และ setter method ในขณะที่ สามารถเข้าถึงได้โดยตรง ด้วย JavaScript

2.2.2.9 JSP, HTML and XML

มาตรฐานของ JSP ได้กำหนดความสามารถในการรองรับโครงสร้างในการสร้างเอกสารด้วยการใช้ HTML และ XML ซึ่งโดยทั่วไปแล้ว ใน JSP page จะมีการส่งข้อมูลไปที่ server ในรูปของ HTTP request (ตัวอย่างเช่น Query argument หรือ Post method) ซึ่งจะทำงานกับข้อมูลที่จัดเก็บบน server ในแบบ interactive และทำการสร้าง dynamic content เพื่อส่งกลับมาให้ client ซึ่ง content สามารถจะจัดการให้อยู่ในรูปแบบของรูปแบบมาตรฐาน (HTML, DHTML, XHTML, XML, etc.) หรือในรูปของ text format และอื่น ๆ โดยเฉพาะ XML ใน JSP ได้นำความสามารถในเรื่อง extensibility และ โครงสร้างในการแสดงผลมาจาก XML ซึ่งความสัมพันธ์ ของ JSP และ XML ได้แก่ JSP page จะมี standard translation ในรูปแบบของ XML document ซึ่งตัว translation นี้จะมีประโยชน์อย่างมาก เพราะจะช่วยจัดการกลไกมาตรฐานในการใช้ XML tool และ APIs ที่ใช้อ่าน ใช้จัดการ และทำงานร่วมกับ JSP document อื่น ๆ

2.2.2.10 องค์ประกอบเบื้องต้นของการสร้าง Web Application ด้วย JSP

- Java Runtime Environment(s) ซึ่งจะทำงานใน server (จำเป็น)
- JSP page ทำหน้าที่ในการ handle request และทำการ generate dynamic content
- Servlet ทำหน้าที่ในการ handle reqest และทำการ generate dynamic content
- Server-side JavaBeans component ซึ่งเป็น component ที่มีทั้งพฤติกรรมและสถานะ
- Static HTML, DHTML, XHTML, XML และ page อื่น ๆ
- Client-side Java Applet, JavaBeans component, และ Java class file อื่น ๆ
- Java Runtime Environment สำหรับทำการ run ที่ client

ซึ่ง Web server ที่จะทำการให้บริการ JSP ควรจะมีคุณสมบัติดังกล่าว

2.2.2.11 URL Mapping

โครงสร้างของ Web Servers จะประกอบไปด้วยชุดของการ mapping ระหว่าง resource และ URL namespace ของ HTTP server ซึ่งอาจจะมากกว่า 1 server

ตัวอย่างเช่น http://www.myco.com/estore ซึ่งอาจจะมี prefix ตามมาข้างหลังเช่น

servlet/login.class

jsp/catalog.jsp

jsp/order.jsp

bean/shoppingcart.class

httpdocs/index.html

...

การ mapping แบบนี้จะถูกใช้ใน JSP โดยที่ HTTP client จะทำการเรียกใช้ application โดยการใช้ HTTP requests(GET,POST) ด้วย URLs ซึ่ง server จะทำการตอบสนองโดยการ mapping request URL ไปสู่ resource และทำการ dispatching/handling ตัว request และ จัดลำดับการตอบสนอง

2.2.2.12 Application and ServletContexts

ใน JSP 1.0 และ ใน Servlet 2.1 จะใช้ HTTP เป็น protocol ในการติดต่อกับ application ที่อยู่ภายใน server ด้วยการใช้ชุดของ URL ในการ mapping กับ resource ซึ่งชุดของ URL นี้จะถูกจัดการด้วย JSP engine (หรือ servlet runtime environment) ด้วย instance ของ javax.servlet.ServletContext Servlet และ JSP ใน application เดียวกัน สามารถใช้ instance ตัวนี้ด้วยกันได้ และยังสามารถ share global application state ด้วยการ share object ผ่าน ServletContextsetAttribute(),getAttribute() และ removeAttribute() method เราอาจจะสมมุติว่าข้อมูลที่ JSP page ใช้โดยตรง สามารถทำการ access ได้จากการตอบสนองของ ServletContext

2.2.2.13 Sessions

ในแต่ละ client หรือ แต่ละ connection จะมีการกำหนด session ขึ้นมา (javax.servlet.http.HttpSession) เฉพาะของแต่ละsession Java Servlet และ JSP ที่อยู่ใน Application เดียวกัน อาจจะมีการ share global session dependent state โดยทำการ sharing object ผ่าน HttpSession โดยใช้ putValue(), getValue() method ซึ่งจะต้องระมัดระวังเมื่อทำการ share หรือ ทำการจัดการ แต่ละ state ระหว่าง JSP และ Servlet ซึ่งทั้ง 2 threads หรือ มากว่า ที่ทำการ execute อาจจะ active ขึ้นมาพร้อม ๆ กันภายใน Servlet และ JSP ซึ่งการทำการ access แบบ synchronization จึงจำเป็นตลอดเวลาที่ทำงานเนื่องจากไม่สามารถคาดเดาพฤติกรรมได้ ซึ่งต้องจำไว้

เสมอว่า session จะ invalid หรือ expire เมื่อถึงเวลาหนึ่ง JSP และ Servlet handling จะอยู่ใน javax.servlet.ServletRequest จะสามารถส่งผ่านสถานะ โดยใช้ ServletRequest setAttribute(), getAttribute() และ removeAttribute() method

2.2.2.14 Application Models ด้วย JSP

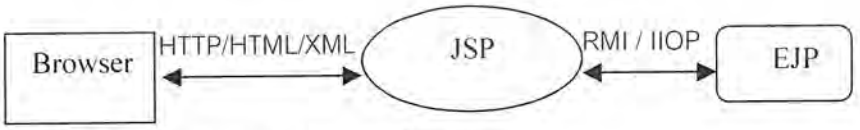
JSP page สามารถที่จะนำมาใช้ร่วมกับ Servlet, HTTP,HTML,XML, Applet, JavaBeans component และ Enterprise JavaBeans component ในการสร้าง application ได้อย่างกว้างขวาง หลายรูปแบบ

- 2-Tier Application

รูปที่ 2.8 แสดง 2 –Tier Application ด้วย JSP

สถาปัตยกรรมแบบ 2 tier (จากตัวอย่างเป็นการ access database) จะเป็นการใช้สถาปัตยกรรมของ cgi ซึ่งสามารถใช้ Java Servlet หรือ JSP ทำการ access external resource (ซึ่งอาจจะเป็น database หรือ application) เพื่อให้บริการ request จาก client ประโยชน์ของสถาปัตยกรรมแบบนี้ก็คือมีเพียงโปรแกรมเดียว และทำให้การสร้าง page 'แบบ dynamic content ง่ายในการจะทำการ request และ อ่านค่าสถานะของ resource อย่างไรก็ดี สถาปัตยกรรมแบบนี้ ไม่สามารถขยายเพื่อตอบสนองจำนวนของ client จำนวนมากที่เข้ามาใช้งานพร้อม ๆ กัน ซึ่งจะทำให้เกิดปัญหาในการ connection กับ resource

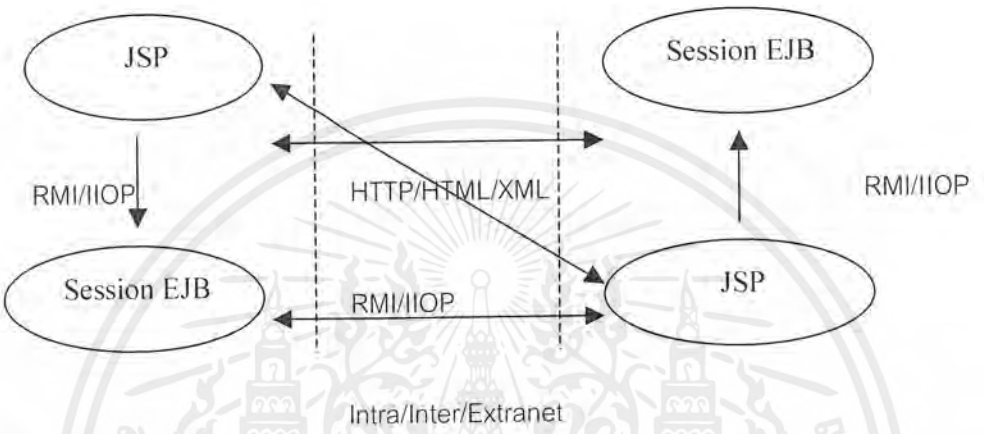
- N-Tier Application



รูปที่ 2.9 แสดง N-Tier Application ด้วย JSP

Application ที่มีรูปแบบดังนี้จะประกอบไปด้วย tier(n>=3) ซึ่งที่ middle tier หรือ JSP จะทำงานกับ back end resource ผ่าน Enterprise JavaBeans component Enterprise JavaBeans Server และตัว EJB component จะเป็นตัวจัดการ access resource โดยที่ตัว EJB server จะรองรับ transaction และ ควบคุมความปลอดภัยในการ access ซึ่งการ programming ที่ใช้รูปแบบนี้จะถูกรองรับด้วย Java 2 Enterprise Edition (J2EE) platform

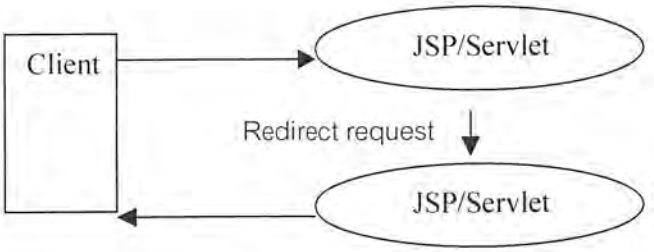
2.2.2.15 Loosely Coupled Applications



รูปที่ 2.10 แสดง Loosely Coupled Application ด้วย JSP

ด้วยรูปแบบนี้ เราจะมี 2 loosely coupled application (ซึ่งจะเหมือนกันทั้งใน Intranet, Extranet หรือ ใน Internet) ซึ่ง application อาจจะเป็นแบบ peer หรือ เป็น client หรือ server ตัวอย่างที่จะเห็นก็คือจะรองรับลักษณะของ chain application ระหว่างแต่ละ vendor ซึ่งจะเห็นได้ว่าจะเป็นอิสระต่อการเปลี่ยนแปลงในการจะนำไป implement สิ่งที่ช่วยให้เกิด loose coupling ใน application คือจะไม่ทำการ communication ระหว่าง application ด้วย RMI/IIOP หรือ Java IDL application จะติดต่อกันโดยผ่าน HTTP ด้วยการใช้ HTML หรือ XML จาก JSP

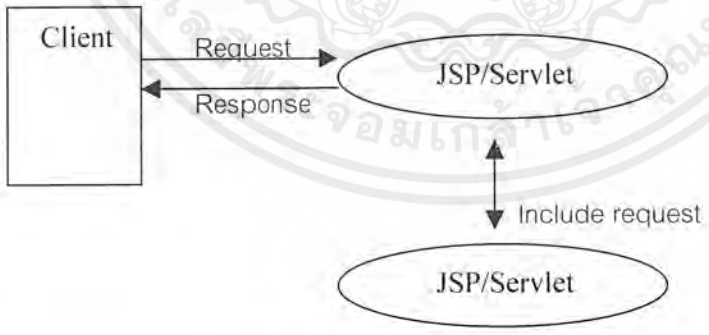
2.2.2.16 Redirecting Requests



รูปที่ 2.11 แสดง การ Redirecting Request

โดยปกติ data ที่ถูกส่งไปให้ client จะมีได้หลากหลายรูปแบบขึ้นอยู่กับ properties ของ client ซึ่งแต่ละอันจะถูก encode โดยตรง ใน request object หรือสามารถค้นหาได้ใน user/client profile (ตัวอย่างเช่น เก็บใน login database) ซึ่งในกรณีที่มีความหลายหลายมาก ๆ และเราสามารถทำการตรวจสอบรายละเอียด เกี่ยวกับ request ถ้าจำเป็น ก็ให้ทำการ redirect request ไปที่ JSP อื่น ด้วยรูปแบบโปรแกรมแบบนี้จะรองรับ Servlet APIs properties ของ HTTP protocol ซึ่งทำการ redirect จะไม่สามารถทำได้ถ้า response stream เริ่มต้นส่งกลับไปที่ client แล้ว ซึ่งพฤติกรรมนี้จะเหมาะกับความต้องการที่จะได้ผลลัพธ์อย่างรวดเร็ว โดยตัว JSP จะมีการจัดการ buffering ของ output stream ตัว JSP code สามารถ redirect request ก่อนที่จะทำการ flushing ออกจาก output buffer โดย Buffering จะจำเป็นมากสำหรับ error handling ซึ่งทำด้วยการ redirecting request

2.2.2.17 Including Requests



รูปที่ 2.12 แสดงการ Including Request

รูปแบบนี้เป็นรูแบบที่มีการ include request ซึ่งจะทำการ initial JSP page ซึ่งจะทำการ generating/composing ผลลัพธ์ บางอย่างแต่ บางจะมันอาจจะต้องการ content บางอย่างจาก page อื่น ซึ่ง content นั้นอาจจะเป็น static content แต่เราสามารถจะได้จากการ generate จาก JSP, Servlet หรือ ASP ในบางกรณีอาจจะใช้ประโยชน์สำหรับการทำ presentation content ซึ่งจะคล้ายกับการ generate ด้วย XML

2.2.2.18 JSP syntax

JSP pages จะเป็นการรวม HTML และ JSP-specific syntax โดยไฟล์จะมีสกุล .jsp จะต้องมีการcompile ก่อนจะไปทำการ execute JSP tag syntax จะประกอบไปด้วยส่วนประกอบดังที่เคยกล่าวบ้างแล้วในข้างต้นนั่นก็คือ directive, declaration, scriptlets, expression และ bean ซึ่งเราจะมาศึกษารายละเอียดของ Syntax กัน

2.2.2.19 JSP directive

JSP directive จะทำการ set global value ซึ่งอาจจะเป็น การกำหนด class declaration, method ที่จะทำการ implement หรือ ชนิดของ output รูปแบบ variable ที่จะเห็นได้แก่ langauge, method, import, content_type, implement และ extends

<%@ language="java" %> -- this is default, currently only Java language supported.

<%@ method = "doPost" import = "java.io.*"%><%@

implements="java.io.Serializable"%>

All directives have scope of the entire JSP page.

2.2.2.20 JSP declarations

Declaration เป็นการกำหนด class-wide variables และ method

```
<SCRIPT runat=server>
private int I=4;
public void init(ServletConfig config)
throws ServletException
{
//override inti method
}
</SCRIPT>
```

2.2.2.21 JSP scriptlet

JSP page จะใช้ scriptlet ในการใส่ java code

```
<%
String name = request.getParameter("Name");
//output the "Name" parameter
out.println(name);
%>
```

2.2.2.22 JSP expressions

Expression การเขียน script อย่างสั้น สำหรับแสดงค่าของ output stream เมื่อ expression ถูกเรียกใช้งาน ผลลัพธ์จะถูกเปลี่ยนเป็น String และแสดงผลออกมา

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

2.2.2.23 ประโยชน์ของ JavaServer Pages technology

- Separation of dynamic and static content ด้วย JavaServer Pages จะช่วยทำให้สามารถแยกส่วนของ static content ออกจาก dynamic content ซึ่งจะแทรกอยู่ในรูปของ static template ซึ่งจะเป็นประโยชน์อย่างมากในการจัดการ Web content

- Support for scripting and tags JavaServer Pages รองรับการใช้งานในลักษณะของ script และ tags โดย tag จะเป็นการเรียกใช้งาน function และ script จะเป็นตัวจัดการกลไกในการเชื่อมต่อ แต่ละ function ในแต่ละ page

- Write Once, Run Anywhere JavaServer Pages เป็นอิสระต่อ platform ทั้ง dynamic Web pages, Web Servers, server component สามารถที่จะเขียน JSP pages บน platform จะสามารถ run บน Web server หรือ Web Application server อื่นๆ ได้ และ สามารถ access จาก Web Browser ใดก็ได้ ซึ่งคุณสามารถสร้าง server component ได้ทุก platform และ run ได้ทุก server

- High quality tool support ด้วยคุณสมบัติ write once, run anywhere ทำให้ผู้ใช้ JSP สามารถ เลือก tool ที่ดีที่สุดได้ ด้วยจุดประสงค์ในการออกแบบ JSP ให้สามารถทำการ portable กับ tools ต่างๆ ได้อย่างมีประสิทธิภาพ ซึ่งในเวอร์ชัน ต่อ ๆ ไปของ JSP จะมีการรองรับในเรื่องของ tag extension และ การจัดการและการติดตั้ง (deployment and installing support)

- Reuse of components and tags JavaServer Pages ช่วยให้มีการใช้งานของ component ในรูปแบบของ reuse component เช่น JavaBeans component, Enterprise JavaBeans component และ custom tags ซึ่ง component เหล่านี้สามารถนำมาใช้ใน interactive tools ต่าง ๆ

สำหรับการพัฒนา component ใหม่ ๆ หรือใช้ในการออกแบบ pages ซึ่งจะช่วยลดเวลาในการพัฒนา ซึ่งช่วยให้การทำงานแบบ cross-platform และสร้างความยืดหยุ่นระหว่าง Java กับ ๆภาษา script อื่น

- Web access layer for N-tier JavaServer Pages เป็นส่วนหนึ่งของ Java 2 Enterprise Edition (J2EE) ซึ่งเป็น Java technology สำหรับระบบ Enterprise คุณสามารถพัฒนา middle-tier server application ด้วยการใช้ JavaServer Pages Web site เป็น front-end สำหรับ Enterprise JavaBeans component ใน J2EE

2.2.3 JavaBean

คุณสมบัติที่สำคัญอีกอย่างของ เทคโนโลยี JSP ในการ access reusable component เช่น JavaBean component จากภายใน JSP page ซึ่งJavaBeans component มีคุณสมบัติ portable เป็นอิสระต่อ platform ซึ่งจะเขียนโดยใช้ Java programming language ผู้พัฒนาสามารถที่จะสร้าง reusable component เพียงครั้งเดียวก็สามารถนำไปใช้ได้ทุกที่ การที่ใช้ JavaBean กับ JSP page สามารถทำการแยก code ออกจาก layout ได้ โดยที่ tab สำหรับ content access และ presentation จะอยู่ภายใน JSP page Logic และ programming code สำหรับ content จะอยู่ภายใน JavaBean component

2.2.3.1 Characteristics of JavaBeans component

JavaBeans component เป็น reusable component เป็น Java object ซึ่งใช้มาตรฐานการตั้งชื่อ mehod สำหรับการเข้าถึง method สามารถทำให้เข้าถึง component และทำการจัดการผ่าน tools ได้ง่าย คุณสมบัติหลักของ JavaBeans คือ properties, methods และ events Properties เป็นส่วนประกอบของ JavaBean ซึ่งอาจจะเป็น Java object หรือ primitives data ซึ่งจะถูกกำหนดด้วย getter และ setter method Method ก็คือ action หรือ service ซึ่ง JavaBean component แสดงออกมา Event ก็จะเป็นเหตุการณ์หรือบางสิ่งที่สนใจหรือเกิดขึ้นกับ Bean

2.2.3.2 Beans in the JSP environment

การใช้ JavaBean component จะอยู่ในรูปของ data encapsulator property ของ JavaBean component จะเป็นข้อมูลหรือสถานะของ bean Properties สามารถ access ได้ด้วย 2 method คือ getter และ setter ค่าของ properties สามารถ access โดยใช้ getter method ถ้า properties สามารถใส่ค่าลงไปได้ จะสามารถเปลี่ยนแปลงค่าโดยใช้ setter method จากตัวอย่างจะมี bean ชื่อ time ซึ่งมี type เป็น Date (java.util.Date) Properties และ method ของ JavaBean component เป็น API สำหรับ เปลี่ยนแปลงข้อมูล และจัดการการให้บริการ การจะทำการ access bean ทำเพียงส่ง messages ไปให้ bean instance

```
public Date getTime()
public void setTime(Date newTime)
```

2.2.3.3 Using JavaBeans components in JSP page

เมื่อเราทำการใช้ servlet และ JSP page ใน application server JavaBeans component จะเปรียบเสมือน business logic สำหรับ application ซึ่งจะทำการจัดการ

- Properties ซึ่งจะเป็นชุดของ คุณสมบัติของ Bean
- Method ทำหน้าที่ในการให้บริการของ business logic
- Properties ที่ถูก encapsulate สำหรับ generated dynamic content

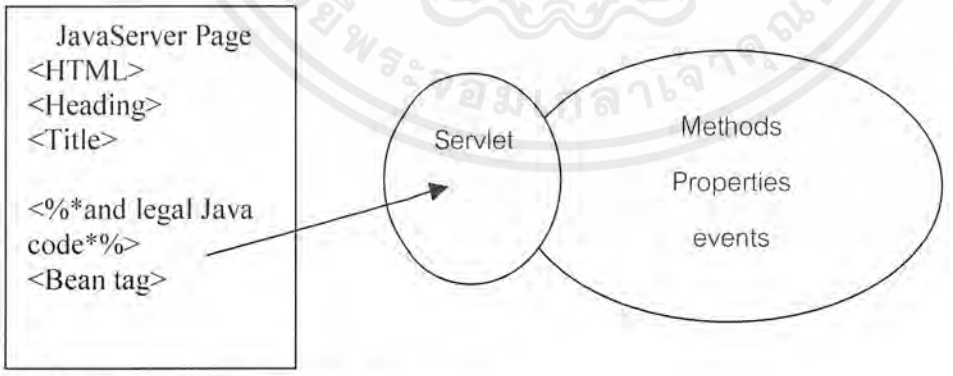
เมื่อผู้ใช้ส่ง request จาก browser ซึ่งอาจจะอยู่ในรูปของ static หรือ dynamic information ซึ่งถ้าเป็น dynamic การ handle request ด้วย servlet จะมีขั้นตอนดังต่อไปนี้

1. servlet จะทำการ locate JavaBeans ที่ต้องการจาก request
2. ทำการกำหนดค่าให้กับ bean โดยใช้ request parameter และทำการเรียกใช้ bussiness logic methods
3. servlet จะทำการส่งผลลัพธ์ไปให้ JSP page
4. JSP page จะใช้ dynamic content จากข้อมูลที่ได้จาก bean สำหรับสร้าง response กลับไปที่ Browser

2.2.3.4 การทำงานของ JavaBeans

Servlet และ back-end application เช่น database จะถูกนำมาใช้ผ่าน bean JSP page จะสามารถ access ผลลัพธ์ จาก bean object ผ่าน tag

- ถ้าต้องการ access bean ภายนอก มันจะทำการ retrieved จาก session หรือ request context
- ถ้าต้องการสร้าง bean มันจะถูกสร้างจาก serialized bean หรือ new bean จากโครงสร้าง



รูปที่ 2.13 แสดง การทำงานของ JavaBeans

2.2.3.5 Bean syntax

เป็นส่วนที่เราต้องการแยกการทำงานออกจากกัน โดย JSP bean tag ช่วยให้การแยก page ตามรายละเอียดของ content โดยใช้ access bean tag

<BEAN naem = "key" varname="alt_var_name"
type="java type" introspect="(yes|no)"
beanName="filename" create="(yes|no)"
scope="(request|session)">
Specify<PARAM> tags(optional)
Closing tag:
</BEAN>

2.2.3.6 <bean> tag attribute

bean tag ประกอบไปด้วย attribute มากมาย เช่น name, varname, type, introspect, beanName, create และ scope name เป็นชื่อของ bean เป็นค่าที่ไว้สำหรับทำการค้นหา bean อย่างเช่นเราจะทำการ getValue() หรือ getAttribute() varname เป็นชื่อของ local variable ซึ่งอ้างอิง bean และ สามารถเรียกใช้จาก scriptlet ถ้าไม่ได้กำหนด varname ไว้ ค่าของ name attibute จะถูกใช้แทน type เป็น type ของภาษา Java ของ bean ที่กำลังทำการ access โดย default type จะเป็น java.lang.Object เมื่อ introspect เป็น yes property แต่ละตัวของ bean ซึ่งมีชื่อ match กับชื่อของ request parameter จะสามารถใช้ setter method ทำการ set property value ให้ตรงกับ request parameter โดยค่าปกติของ introspect จะเป็น yes สำหรับ create attibute ถ้าเป็น yes bean จะถูกสร้าง (ผ่าน Beans.instantiate()) ถ้าไม่พบตามที่กำหนดไว้ใน scope และถ้าไม่ได้กำหนดและไม่พบ bean จะ return error กลับมา ถ้า scope เป็น request bean จะถูก retrieved จาก request context ซึ่งจะใช้เมื่อต้องการให้ bean สามารถเรียกใช้ได้ภายใน HTTP request เดียวกัน หากเป็น session scope bean จะถูก retrieved จาก session เดียวกัน โดยใช้ getValue() method ถ้า bean ถูกสร้างขึ้นแล้ว และจัดเก็บอยู่ใน session เดียวกัน ซึ่งจะใช้เมื่อ bean ถูกใช้จากหลาย ๆ HTTP request beanName เป็น filename ที่ใช้ในการ สร้าง JavaBean component จาก serialized bean file

2.2.3.7 JSP bean example

<HTML><HEAD>...</HEAD>
<bean name="Userinfo" varname"xtb"

```
type="TutorialXTBean" introspect="no" create="no" scope="session">
</bean>
<BODY>

<!—if this were a real application, you would confirm your
address below and provide billing information
to finalize your transaction.-->

<p>
<pre><%=xtb.getFirstname()%><%xtb.getLastname()%>
<%=xtb.getStreet()%>
<%=xtb.getCity()A%>,<%=xtb.getState()%><%=xtb.getZip()%>
</pre>
...
</BODY></HTML>
```

2.2.3.8 HTML template tag <insert> and <repeat>

จุดประสงค์อย่างหนึ่งของ JSP technology คือทำการแยก layout และ content ออกจากกัน HTML template tags จะช่วยในเรื่องนี้ ทำให้จะช่วยในเรื่องของ layout ทั้ง tab <insert> และ <repeat><insert> tag ทำให้ ผู้พัฒนา layout ในการออกแบบได้ง่ายขึ้น โดยลักษณะจะคล้ายกับ HTML แต่function จะคล้ายกับ java code

<insert bean="beanname" property="propname">

Behavior is same as :

<%= beanvar.getPropaname() =>

สำหรับ tag <repeat> จะทำให้ ผู้ออกแบบ layout สามารถใส่ Java code ที่เป็น loop ผ่าน indexed properties ซึ่งจะเป็นประโยชน์อย่างมากสำหรับ function เมื่อทำงานกับ block ข้อมูล เช่น rows ของ database

2.2.4 JDBC (Java Database Connectivity)

จาวามีการทำงานบนสภาวะแวดล้อมระบบไคลเอนต์/เซิร์ฟเวอร์ โดยที่ตัวโปรแกรมจะถูกดาวน์โหลดและประมวลผลบนเครื่องไคลแอนต์ บริษัท ซัน ไมโครซิสเต็มนับว่าเป็นผู้นำทางด้านการพัฒนาไลบรารี่ เพื่อให้เกิดความสะดวกสบายในการสร้างโปรแกรมประยุกต์ JDBC API ซึ่งเป็นไลบรารี่ตัวหนึ่งจะประกอบด้วยกลุ่มของคลาสและอินเทอร์เฟซ ซึ่งถูกใช้งานสำหรับการเขียนโปรมแกรมจาวา JDBC จะถูกออกแบบเพื่อที่จะสามารถอนุญาตให้นักพัฒนาโปรแกรม สามารถพัฒนาอินเทอร์เฟซที่เป็นมาตรฐาน โดยที่ JDBC API เป็นคลาสของจาวาสำหรับใช้ในการติดต่อกับฐานข้อมูลต่าง ๆ

JDBC เป็นพื้นฐานใหม่สำหรับนักพัฒนาโปรแกรมประยุกต์ (application) ในการติดต่อกับแหล่งข้อมูล (data source) โครงสร้างของ JDBC ถูกออกแบบมาเพื่อให้นักพัฒนาโปรแกรมสามารถสร้างมาตรฐานในการอินเทอร์เฟซ (interface) JDBC ถูกเรียกใช้โดยจาวา (Java) เพื่อการเข้าถึงฐานข้อมูลที่มีลักษณะคล้ายกับ ODBC (Open Database Connectivity) คือพัฒนามาจาก X\open SQL call-level interface ทางจาวาซอร์ฟ (JavaSoft) ได้สร้างข้อกำหนดในการออกแบบ JDBC (JDBC-Compliant) ขึ้นเพื่อให้มีหลักเกณฑ์เดียวกันโดยที่ไดรเวอร์(driver) ของ JDBC ต้องสนับสนุนมาตรฐาน ANSI SQL-92 และต้องผ่านการทดสอบจากทางจาวาซอร์ฟ ทำให้ผู้พัฒนาโปรมแกรมขจัดความยุ่งยากกับ SQLที่แตกต่างกันของแต่ละฐานข้อมูลส่งผลให้ JDBC มีความยืดหยุ่นสูง

2.2.4.1 ความหมายของJDBC

JDBCคือ API ของจาวาที่ใช้สำหรับตีความคำสั่ง SQL ซึ่งประกอบด้วยชุดของคลาส(classes)และอินเตอร์เฟซ ที่เขียนด้วยโปรแกรมภาษาจาวา ตัวJDBC ที่เป็น APIมาตรฐานสำหรับการพัฒนาระบบฐานข้อมูลทำให้มีความเป็นไปได้ในการเขียนโปรแกรมใช้งานเกี่ยวกับฐานข้อมูลโดยอาศัยจาวา API เพียงอย่างเดียวเท่านั้น การใช้ JDBC ทำให้ง่ายในการส่งคำสั่ง SQL ให้กับฐานข้อมูลชนิดต่าง ๆ ในรูปแบบเดียวกัน คือJDBC API จะไม่บังคับให้เขียนโปรแกรมขึ้นมาเฉพาะเพื่อใช้กับฐานข้อมูลของ Sybase และอีกโปรแกรมหนึ่งสำหรับฐานข้อมูลของ Oracle และอีกหลายโปรแกรมสำหรับฐานข้อมูลชนิดต่าง ๆ แต่สามารถที่จะเขียนโปรแกรมโดยใช้ JDBC API เพียงโปรแกรมเดียว และโปรแกรมนั้นสามารถส่งคำสั่ง SQL ไปยังฐานข้อมูลชนิดต่าง ๆ ได้ ด้วยโปรแกรมประยุกต์ที่ถูกเขียนโดยโปรแกรมภาษาจาวาสามารถกำจัดความยุ่งยากที่จะต้องเขียนโปรแกรมประยุกต์ชนิดต่างๆ สำหรับสภาวะแวดล้อมต่าง ๆ กัน การรวมกันระหว่างจาวาและ JDBC ทำให้นักเขียนโปรแกรมสามารภเขียนโปรแกรมเพียงโปรมแกรมเดียวแล้วสามารถทำไปใช้ได้ในทุก ๆแห่งที่ต้องการ จาวาจัดว่ามีความปลอดภัย ง่ายในการใช้งาน ง่ายต่อการทำความเข้าใจ และสามารถถ่ายโอนได้อย่างอัตโนมัติบนระบบเครือข่าย ซึ่งจาวาจัดเป็นภาษาที่เหมาะสมในการเขียนโปรแกรมที่ใช้จัดการเกี่ยวกับฐานข้อมูล JDBC เป็นทางเลือกที่ทำให้โปรมแกรมสามารถติดต่อกับฐานข้อมูลโดยไม่คำนึงถึงชนิดฐานข้อมูลที่ทำการติดต่อ

JDBC เป็นส่วนขยายความสามารถของจาวา จั่นคือเมื่อใช้จาวาและ JDBC API จะสามารถเผยแพร่เว็บเพจ (web page) ที่มีการเรียกใช้ข้อมูลจากฐานข้อมูลในลักษณะต่าง ๆ ตามที่ต้องการได้

2.2.4.2 ความสามารถของ JDBC

ในการใช้งานทั่วไป JDBC มีความสามารถทำงานได้3อย่างดังนี้

สร้างการเชื่อมต่อกับฐานข้อมูล

ส่งคำสั่ง SQL

จัดการกับผลลัพธ์ที่เกิดขึ้น

ตัวอย่างนี้เป็นโปรแกรมแสดงการทำงานทั้ง3อย่างข้างต้น

```
Connection con=DriverManager.getConnection("jdbc:odbc:wombat","login","password");
Statement stmt=con.createStatement();
Result rs=stmt.executeQuery("SELECT a,b,c FROM Table1");
While(rs.next())
{
 int x=getInt("a");
 String s=getString("b");
 Float f=getFloat("c");
}
```

2.2.4.3 JDBC จัดเป็น API ในระดับล่าง

JDBC จัดเป็นอินเตอร์เฟซในระดับล่าง ซึ่งหมายความว่า JDBC จะถูกนำไปใช้เกี่ยวกับคำสั่ง SQL โดยตรงซึ่ง JDBC จะทำงานได้เป็นอย่างดีและง่ายดายเมื่อเทียบกับ Database Connectivity API ตัวอื่นๆ แต่ในความเป็นจริงแล้วJDBC ได้ถูกออกแบบมาเพื่อเป็นพื้นฐานสำหรับในการสร้างอินเตอร์เฟซในระดับสูงมีความสารมรถสื่อความได้ง่ายและเป็น API ที่อำนวยความสะดวกในการปรับเปลี่ยนไปสู่อินเตอร์เฟซในระดับต่ำได้เช่น JDBC มี API ระดับสูง2ชนิดที่พัฒนาบนชั้นบนของ JDBC

1. SQL ชนิดฝังตัวจาวา JDBC ต้องการคำสั่ง SQL โดยส่งเป็นสายอักขระไปยังเมทธอด (method) ของจาวา SQL ชนิดฝังตัวจะอนุญาตให้นักเขียนโปรแกรมสามารถที่จะรวมคำสั่ง SQL โดยตรงเข้าไปเป็นส่วนหนึ่งของโปรแกรมจาวา เช่นตัวแปรของจาวาสามารถถูกใช้ในคำสั่ง SQL เพื่อใช้ในการรับหรือจัดการกับคำสั่งของSQL ได้
2. การแปลงจากตารางฐานข้อมูลเชิงสัมพันธ์ไปเป็นคลาสของจาวาโดยตรง คือการแปลงที่อยู่ในรูปแบบของ object\relational แต่ละแถวในตารางจะกลายเป็น instance ของคลาสนั้น และแต่ละค่า

ของสดมภ์ (column) จะเป็นคุณสมบัติของ instance นั้นๆนักเขียนโปรมแกรมสามารถที่จะจัดการโดยตรงกับออบเจ็ต (object) ของจาวา JDBC กำลังได้รับความนิยมเพิ่มมากขึ้น สำหรับนักพัฒนาจำนวนมากได้ทำงานกับเครื่องมือที่มี JDBC เป็นส่วนประกอบพื้นฐาน เพื่อสร้างโปรแกรมที่ง่ายทำให้นักเขียนโปรแกรมสามารถเขียนโปรแกรมเพื่อจัดการกับฐานข้อมูลโดยง่าย

2.2.4.4 การเปรียบเทียบระหว่าง JDBC กับ ODBC

ไมโครซอร์ฟ ODBC API ถูกใช้อย่างกว้างขวางในการเขียนโปรแกรมเพื่อจัดการกับฐานข้อมูลเชิงสัมพันธ์ โดยที่ ODBC สามารถเชื่อมต่อกับฐานข้อมูลได้หลากหลายบริษัท และหลายสภาวะแวดล้อมโดยที่สามารถที่จะใช้ ODBC ร่วมกับจาวาได้ แต่ทางเลือกที่ดีที่สุดคือการใช้ JDBC โดยมีเหตุผลดังนี้

- ODBC ไม่เหมาะสมสำหรับการใช้โดยตรงจากจาวาเพราะว่า ODBC ใช้ภาษาCเป็นอินเตอร์เฟซแต่การเรียกใช้จากจาวาจะเป็นในลักษณะของเนทีพ C (native c) ซึ่งจะมีปัญหาในด้านความปลอดภัย การปฏิบัติงาน ประสิทธิภาพและความเข้ากันของงานประยุกต์ต่างๆ
- การแปลงของ ODBC C API ไปเป็นจาวา API จะไม่ได้ผลลัพธ์ตามที่ต้องการ เช่นจาวาไม่มีตัวชี้(pointer)
- ODBC ยากในการเรียนรู้ ODBC จะรวมเอาคุณสมบัติพื้นฐานและขั้นสูงเข้าไว้ด้วยกัน และมีตัวเลือกที่ซับซ้อนสำหรับการคิวรี(query) ง่ายๆในทางตรงข้าม JDBC ถูกออกแบบมาให้รักษาสิ่งที่เป็นพื้นฐานและอนุญาติให้มีการเพิ่มความสามารถขั้นสูงได้ตามแต่ต้องการ
- การจัดการไดรเวอร์(driver) ODBCไดรเวอร์ต่างๆจะต้องติดตั้งทุกๆเครื่องที่เป็นไคลเอนต์(client) แต่ JDBC สามารถถูกติดตั้งโดยอัตโนมัติได้ตามทุกๆสภาวะแวดล้อมของจาวา

2.2.4.5 โครงแบบของจาวาซอร์ฟ

จาวาซอร์ฟเตรียมส่วนประกอบของผลิตภัณฑ์ JDBC ไว้ 3 ส่วนซึ่งเป็นส่วนหนึ่งของJDK(java Development Kit) ซึ่งมีรายละเอียดคือ

- JDBC driver manager จัดเป็นส่วนประกอบหลักของสถาปัตยกรรม JDBC ที่มีขนาดเล็กและเป็นพื้นฐาน จัดเป็นหน้าที่หลักในการเชื่อมต่องานประยุกต์จาวาเข้ากับไดรเวอร์ JDBC ที่ถูกต้อง
- JDBC driver test suite ให้ความมั่นใจว่าไดรเวอร์ JDBC สามารถใช้งานเข้ากับโปรแกรมได้
- JDBC-ODBC bridge อนุญาตให้ไดรเวอร์ODBCถูกใช้ในฐานะที่เป็นไดรเวอร์ของ JDBC

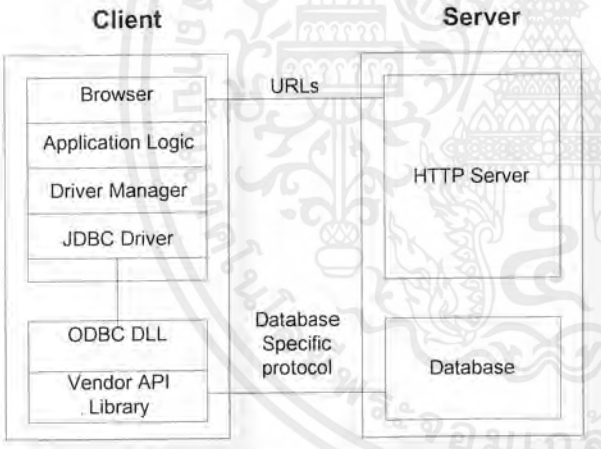
2.2.4.6 ชนิดของ JDBC Driver

JDBC drivers ในปัจจุบันมี 4 แบบคือ

- JDBC/ODBC bridge.
- Native-API,partly Java driver.
- Network-protocol, all-Java driver.
- Native-protocol, all-Java driver.

2.2.4.7 JDBC/ODBC bridge

JDBC/ODBC bridge ได้ถูกพัฒนาขึ้นโดยการร่วมมือกันระหว่าง Javasoft และ Intersolv เพื่อที่จะเพิ่มความสามารถให้กับฐานข้อมูลจำนวนมากที่ใช้ ODBC ในส่วนของ client โปรแกรมจะถูกเขียนขึ้นด้วย JDBC API โดย bridge ตัวนี้จะทำการ แปลงคำสั่งจาก JDBC ไปเป็นคำสั่งของ ODBC แล้วส่งคำสั่งนั้นไปยังตัว ODBC driver เพื่อจัดการฐานข้อมูล ข้อได้เปรียบหลักของ bridge ตัวนี้คือ โปรแกรมที่เขียนขึ้นนั้นจะง่ายต่อการเข้าถึงข้อมูลจากระบบฐานข้อมูลชนิดนี้ทำให้ต้องนึกถึงผลข้างเคียงที่อาจเกิดขึ้นและความยุ่งยาก เพราะว่าคำสั่งต้องส่ง จาก JDBC ไปยัง bridgeที่เชื่อมต่อไปยัง ODBC driver และสุดท้ายจาก ODBC ก็ส่งไปยัง native client-API เพื่อไปยังฐานข้อมูล

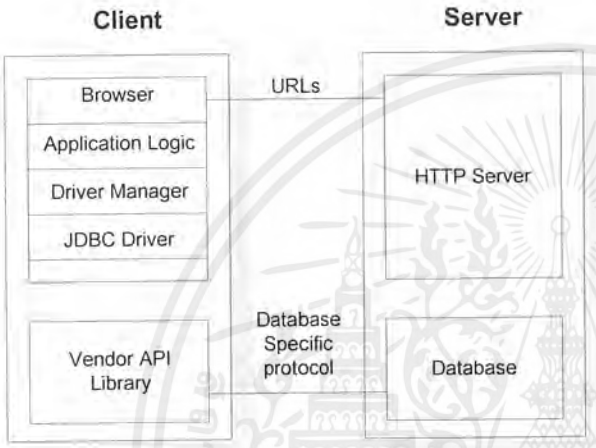


รูปที่ 2.14 ภาพแสดงการ access โดยใช้ Driver (Type I)

ตัว driver ชนิดนี้ทำให้โปรแกรมส่งข้อมูลไปไม่ทันทีทันใดกับที่เรียกไป เพราะ code หลัก ๆ ต้อง ถูกติดตั้งไว้ก่อนแล้ว บนทุก ๆ เครื่องของ client ที่ต้องการใช้การเชื่อมต่อด้วย JDBC/ODBC bridge จึงไม่ได้แก้ไขปัญหาการเปลี่ยนแปลงของ client program

2.2.4.8 Native-API,Partly Java Driver

เป็นแบบ two-tier นั้นคือ JDBC driver ต้องการ library เพื่อเปลี่ยนแปลง function ของ JDBC ไปเป็น querylanguage ต่าง ๆ ของ DBMS (เช่น library สำหรับ Sybase คือ dblib, สำหรับ Oracle คือ ocilib และอื่น ๆ) driver เหล่านี้โดยปกติจะเขียนขึ้นโดยภาษา Java และ C/C++ เนื่องจาก driver ต้องใช้ layer ของ C ในการเรียกใช้ไปยัง library driver ชนิดที่ 2 เช่น JDBC/ODBC Bridge ต้องการให้ code (ซึ่งคือ library ของแต่ละผู้ผลิต) ติดตั้งบนแต่ละ client ดังนั้นจึงมีปัญหา ทางด้านการดูแลซอฟท์แวร์เช่นเดียวกับ Bridge อย่างไรก็ตาม driver ชนิดที่ 2 นี้ จะเร็วกว่าชนิดที่ 1 เพราะ layer พิเศษของแปลงเป็น ODBC ถูกเอาออกไป

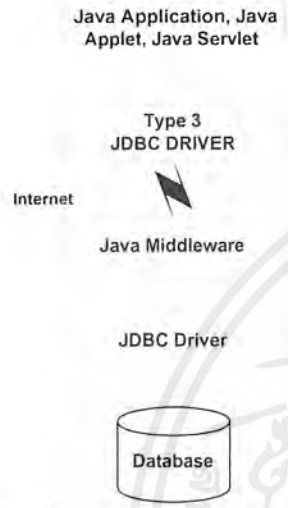


รูปที่ 2.15 ภาพแสดงการ access โดยใช้ Driver(Type II.)

2.2.4.9 Network-Protocol, All-Java Driver

driver ชนิดที่ 3 นี้เปลี่ยนการเรียก JDBC ไปเป็น Protocol เครือข่ายฐานข้อมูลอิสระ ซึ่งจะแปลงไปเป็นการเรียก database-specific API โดย middle-tier server (middle-tier server อาจใช้ driver ชนิดที่ 1 หรือ ชนิดที่ 2 ถ้าเขียนโดย Java) สถาปัตยกรรมโดยรวมประกอบ ด้วย 3 tiers คือ JDBC client และ driver, middleware, และฐานข้อมูลที่ถูกเข้าถึง JDBC driver ขนาดเล็ก (ประมาณ 200KB หรือ น้อยกว่า) ทำงานบน client และมีการใช้ logic ในการผ่านคำสั่ง SQL ในเครือข่ายไปยัง JDBC server รับข้อมูลกลับจาก server,และจัดการติดต่อ โดย driver ชนิดที่ 3 นี้ จะมีลักษณะเป็น just-in-time client deployment JDBC server จัดการติดต่อหลายอย่างกับฐานข้อมูลรวมทั้งการยกเว้นและสถานะของเหตุการณ์ที่มีผลต่อมาจากการทำคำสั่ง SQL JDBC Server ยังจัดรูปแบบของข้อมูลสำหรับการส่งในเครือข่ายไปยัง JDBC client Middleware Server สามารถ Implement เป็นส่วนประกอบดั้งเดิมหรือเขียนโดย Java การใช้แบบเดิมติดต่อกับ server ฐานข้อมูล

ใช้ client library ของผู้ผลิต หรือ ODBC เช่น dbAnywhere ของ Symantec และ SequeLink ของ Intersoft ถึงแม้ว่า SequeLink ไม่ต้องการ client library ของฐานข้อมูลติดตั้งบน server แต่จะใช้ library ของตัวเอง server จะต้องตั้งค่าสำหรับฐานข้อมูล (เช่น DSQuery กับ Sybase) พารามิเตอร์ เฉพาะของฐานข้อมูล (การ log การแปลง) และพารามิเตอร์อื่น ๆ ที่ server ต้องการ ถ้า Middleware เขียนโดย Java จะสามารถใช้ JDBC-compliant sever ในการสื่อสารกับ DBMS โดยผ่าน Protocol ของฐานข้อมูลของผู้ผลิต



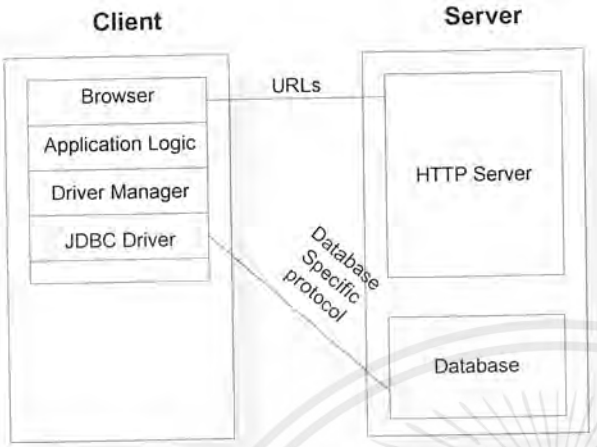
รูปที่ 2.16 ภาพแสดงการAccess โดยใช้ไดรฟ์เวอร์ JDBC แบบ three-tier Type 3

driver เหมาะสมสำหรับแอปพลิเคชั่นที่มีหลายผู้ใช้บนอินเตอร์เน็ต/อินทราเน็ต มากที่สุด ที่ซึ่งการกระทำของข้อมูลต่อเนื่องจำนวนมาก มาก เช่น queries,searches และอื่น ๆ ถูกคาดหมายประสิทธิภาพเป็นสิ่งสำคัญ Server สามารถจัดการฐานข้อมูลจำนวนมากรวมกันได้ สามารถให้การตรวจสอบและดูแลข้อมูล สามารถทำ loadbalancing และสนับสนุน catalog และ query caches และอย่างที่ได้กล่าวไปแล้วแอปพลิเคชั่นฐานข้อมูลบน web แบบ three-tier เกี่ยวข้องกับความปลอดภัย firewalls และ proxies ซึ่งType III driver สนับสนุนสิ่งเหล่านี้ สิ่งที่เป็นข้อเสียของ network-contric driver คือส่วนประกอบของ server เป็น Middleware ผู้ผลิตแต่ละรายใช้ Middleware ของตน เองสำหรับการติดต่อในเครือข่าย

2.2.4.10 Native-Protocol ,All-Java Driver(Type IV)

การทำงานจะแปลงการเรียก JDBC ไปเป็น Protocol เครือข่ายโดยตรงโดยใช้ Driver ซึ่งเขียนขึ้นเฉพาะโดย Driver เหล่านี้สามารถเขียน โดย Java ทั้งหมด และสามารถให้ การส่งข้อมูล แบบ just-in-time ของ Applet (เหมือน Type III) เนื่องจาก driver เหล่านี้แปลง JDBC ไปเป็น

Protocol ดั้งเดิมโดยตรงไปไม่มีการใช้ ODBC หรือ API ดั้งเดิม จึงสามารถให้การเข้าถึงฐานข้อมูลที่มีประสิทธิภาพสูง Driver เหล่านี้ทำขึ้นจากผู้ผลิต DBMS เท่านั้น จากความจริงที่ว่าความรู้ในเรื่อง Protocol เป็นผู้ผลิต ในปัจจุบันยังมี Type IV ใช้อยู่น้อยแต่จำนวนน่าจะมากขึ้น



รูปที่ 2.17 ภาพแสดงการ access โดยใช้ไดรฟเวอร์ Type 4

2.2.4.11 การเชื่อมต่อ

การเชื่อมต่อถูกแทนด้วยการเชื่อมต่อกับฐานข้อมูล โดยแต่ละการเชื่อมต่อจะประกอบด้วยคำสั่ง SQL ซึ่งสามารถทำงานและได้ผลลัพธ์ส่งกลับมาโดยการเชื่อมต่อนี้ งานประยุกต์หนึ่งๆสามารถมีหนึ่งการเชื่อมต่อหลายๆการเชื่อมต่อกับหนึ่งฐานข้อมูลหรือสามารถที่จะเชื่อมต่อกับหลายๆฐานข้อมูลที่แตกต่างได้

2.2.4.12 การเริ่มต้นการเชื่อมต่อ

วิธีมาตรฐานในการจัดตั้งการเชื่อมต่อกับฐานข้อมูลโดยเรียกใช้เมทธอด.getConnection โดยเมทธอดนี้จะมอง URLในรูปของสายอักขระ คลาสของ DriverManager จะถูกอ้างอิงในฐานะที่เป็นชั้น(layer)ของJDBC management โดยพยายามระบุที่อยู่ของไดรเวอร์ที่สามารถเชื่อมต่อกับฐานข้อมูลในรูปแบบของURLโดยคลาสนี้จะทำหน้าที่ดูแลรายการของคลาสDriveที่ลงทะเบียนแล้ว เมื่อเมธอด getConnection ถูกเรียกใช้ ตัวเมธอดนี้จะตรวจสอบแต่ละไดรเวอร์ในรายการจนกระทั่งพบไดรเวอร์ตัวที่สามารถใช้ในการเชื่อมต่อกับฐานข้อมูลตามURLนั้นๆเมทธอดDriverสามารถเชื่อมต่อโดยใช้URLเพื่อใช้ในการจัดตั้งการเชื่อมต่อได้ผู้ใช้สามารถที่จะหลีกเลี่ยงชั้น JDBC management โดยการเรียกใช้เมทธอด driverได้โดยตรง ในกรณีนี้จะเกิดประโยชน์เมื่อมี 2ไดรเวอร์ที่สามารถเชื่อมต่อกับฐานข้อมูล และผู้ใช้มีความต้องการเจาะจงใช้ไดรเวอร์ชนิดใดชนิดหนึ่งเป็นพิเศษในทางปรกติจะง่ายกว่าถ้าให้คลาส DriverManagerเป็นตัวจัดการการเริ่มต้นการเชื่อมต่อ

2.2.4.13 ข้อกำหนดURL

URL(Uniform Resource Location) จะให้ข้อมูลตำแหน่งที่อยู่ของทรัพยากรบนอินเทอร์เนต(internet) สามารถที่จะแทนที่อยู่ได้ โดยส่วนแรกของURL จะเป็นชนิดของโปรโตคอลที่ใช้สำหรับการเข้าใช้ข้อมูล และจะตามด้วยเครื่องหมายจุดคู่(colon) เสมอ มีโปรโตคอลที่ถูกใช้บ่อยๆคือftp(file transter protocol)และhttp(hypertext transfer protocol) ส่วนที่เหลือของURLหลังเครื่องหมายอัตภาค(semicolon)แรกจะให้ข้อมูลเกี่ยวกับที่อยู่ของแหล่งข้อมูล สำหรับโปรโตคอล ftp และ httpส่วนที่เหลือของURLจะแสดงถึงเครื่องแม่ข่าย(host)และอาจมีส่วนขยายอื่นๆเพิ่มเติมขึ้นได้

2.2.4.14 ข้อกำหนด URL สำหรับ JDBC

JDBC URLได้มีวิธีการในการระบุถึงฐานข้อมูลที่มีไดรเวอร์ที่เหมาะสมคอยจัดการและจัดตั้งการเชื่อมต่อ ผู้ใช้ไม่ต้องมีความกังวลเกี่ยวกับรูปแบบของJDBC URL JDBC URLถูกใช้หลากหลายชนิดกับไดรเวอร์ต่างๆ ซึ่งถือว่าเป็นความยืดหยุ่น สิ่งแรกคือจะอนุญาติให้ไดรเวอร์ที่แตกต่างกันใช้แบบแผนที่แตกต่างกันสำหรับชื่อของฐานข้อมูล odbc subprotocol อย่างที่สองJDBC URLอนุญาตให้ผู้เขียนไดรเวอร์ทำการเข้ารหัสข้อมูลสำหรับการเชื่อมต่อที่สำคัญๆและอย่างที่สามJDBC URL มีระดับของindirectionซึ่งมีความหมายคือ JDBC URL สามารถอ้างเครื่องแม่ข่ายหรือชื่อของฐานข้อมูลที่เป็นทางตรรก(logical)ซึ่งสามารถเปลี่ยนไปเป็นชื่อที่แท้จริงได้ การอนุญาตินี้ผู้ดูแลระบบจะหลีกเลี่ยงการระบุเครื่องแม่ข่ายที่แน่นอนลงไปในส่วนของJDBC NAME ที่มีจำนวนของความแตกต่างกันของnetwork name service(เช่นDNS) สัญลักษณ์มาตรฐานสำหรับ JDBC URL แสดงตามข้างล่าง โดยประกอบด้วย3ส่วนซึ่งแบ่งแยกโดยเครื่องหมายจุดคู่

jdbc:<subprotocol>:<subname>

มีรายละเอียดคือ

- jdbc the protocol เป็นโปรโตคอลที่ใช้ใน JDBC URL จะเป็น jdbc เสมอ
- <subprotocol> เป็นชื่อของไดรเวอร์หรือชื่อของกลไกการเชื่อมต่อกับฐานข้อมูล ซึ่งอาจ สนับสนุนหนึ่งหรือหลายไดรเวอร์ ตัวอย่างที่สำคัญคือ"odbc"ซึ่งถูกสงวนไว้สำหรับURLที่ชื่อของแหล่งข้อมูลอยู่ในลักษณะของ ODBC สำหรับตัวอย่างในการจัดการฐานข้อมูลโดย JDBC-ODBC bridge สามารถใช้ URLตามลักษณะดังนี้ jdbc:odbc:fred โดย subprotocol คือ"odbc"และ subname คือfred ถ้ามีความต้องการที่จะใช้ network name service(ชื่อฐานข้อมูลในJDBC URL ไม่ใช่ชื่อที่แท้จริง) โดย naming service สามารถที่จะ เป็น subprotocol ดังนั้นจากตัวอย่างสามารถจะมี URL ในลักษณะ jdbc:dcenaming:account_payable จากตัวอย่างนี้ URL จะเป็น naming service เฉพาะที่ใช้ติดต่อกับฐานข้อมูล

- <subname> เป็นวิธีในการเจาะจงฐานข้อมูล subname จะเปลี่ยนแปลงไปได้โดยขึ้นอยู่กับ subprotocol และสามารถที่จะมีsubsubname ได้ตามสัญลักษณ์ภายในตามที่ผู้เขียนไดรเวอร์เลือกใช้ โดยที่ subname จะสามารถให้ข้อมูลที่เพียงพอสำหรับที่อยู่ของฐานข้อมูล ในตัวอย่างก่อน"fred"ถือว่ามีความเพียงพอเพราะว่าodbc ได้จัดเตรียมข้อมูลส่วนที่เหลือไว้แล้ว กรณีฐานข้อมูลบนเซิร์ฟเวอร์ระยะไกลจะมีความต้องการรายละเอียดที่เพิ่มมากขึ้นอย่างไรก็ตามถ้าฐานข้อมูลที่ถูกเข้าใช้จากระบบอินเตอร์เจตจากตัวอย่างข้างต้น ที่อยู่ของระบบเครือข่ายควรจะถูกระบุลงใน JDBC URL ในส่วนของ subname และจะต้องเป็นไปตามมาตรฐานของ URLnamingดัง //hostname:pot/subname สมมุติว่า"dbnet" เป็นโปรโตคอลสำหรับใช้เชื่อมต่อกับเครื่องแม่ข่ายบนอินเตอร์เน็ต JDBC URL จะต้องอยู่ในรูปแบบดังนี้คือ jdbc:dbnet://wombat:306/fred

2.2.4.15 ODBC Subprotocol

subprotocol odbc ถือว่าเป็นกรณีพิเศษ มักจะถูกใช้กับ URL ที่ระบุชื่อแหล่งข้อมูลในลักษณะ ODBCและมีคุณสมบัติพิเศษ ในการที่อนุญาตให้จำนวนใด ๆ ของค่าแอททิบิว (attribute) สามารถเจาะจงได้ตามหลัง subname โดยสัญลักษณ์รูปแบบเต็มของโปรโตคอลodbcคือ jdbc:odbc:<data-source-name>[;<attrebute-name>=<attrebute-value>] ดังเช่น

dbc:odbc:wombat jdbc:odbc:qeora:UID-kgh;PWD=aaaa

2.2.4.16 การลงทะเบียน subprotocol

ผู้พัฒนาไดรเวอร์สามารถที่จำทำการจองชื่อที่จะใช้เป็น subprotocol ของ JDBC URLเมื่อคลาส DriverManager แสดงชื่อของ subprotocol ในรายการของไดรเวอร์ register ซึ่งไดรเวอร์เหล่านี้จะถูกจองชื่อและจัดตั้งเกี่ยวกับการเชื่อมต่อกับฐานข้อมูลตามที่ระบุในไดรเวอร์ นั้นๆ สำหรับตัวอย่าง odbc จะถูกจองสำหรับ JDBC-ODBC Bridge ปัจจุบันจาวาซอฟท์เป็นผู้จัดการลงทะเบียนชื่อ JDBC subprotocol อย่างไม่เป็นทางการ

2.2.4.17 การส่งคำสั่ง SQL

เมื่อการเชื่อมต่อถูกจัดตั้งขึ้น จะถูกใช้ในการส่งผ่านคำสั่ง SQL ไปยังฐานข้อมูล JDBC ไม่สามารถที่จะใส่ข้อจำกัดต่าง ๆ รวมเข้ากับคำสั่ง SQL ชนิดต่าง ๆ ซึ่งสามารถส่งไปได้ซึ่งเพื่อวัตถุประสงค์คือความยืดหยุ่น สามารถอนุญาติให้มีการใช้คำสั่งฐานข้อมูลที่เป็นพิเศษหรือคำสั่งที่ไม่เป็นภาษา SQL ตามที่ต้องการอย่างไรก็ตามผู้ใช้สามารถที่จะมีการตอบโต้เพื่อทำให้มั่นใจได้ว่าฐานข้อมูลสามารถที่จะดำเนินการกับคำสั่ง SQL ที่ถูกส่งไป ดังตัวอย่างงานประยุกต์ซึ่งพยายามที่จะส่ง store procedure เรียกไปยังระบบจัดการฐานข้อมูลซึ่งไม่สนับสนุน store procedure ทำให้เกิดความผิด

พลาด JDBC ต้องการให้ การออกแบบไดรเวอร์อย่างน้อยต้องสนับสนุนมาตรฐาน ANSI SQL-92 JDBC ได้จัดเตรียมคลาส 3 คลาสสำหรับการส่งคำสั่ง SQL ไปยังฐานข้อมูลและเมทธอด 3 เมทธอด ในการเชื่อมต่ออินเตอร์เฟซ ซึ่งสร้าง instance สำหรับคลาสเหล่านี้โดยมีรายละเอียดคือ

1. Statement ถูกสร้างโดยเมทธอด createStatement เป็นออบเจ็คที่ใช้ในการส่งคำสั่ง SQL พื้นฐานPreparedStatment ถูกสร้างโดยเมทธอด prepareStatement จะถูกใช้สำหรับคำสั่ง SQL ที่มีหนึ่งหรือหลายพารามิเตอร์ (parameter) เมทธอดนี้มีกลุ่มของเมทธอดซึ่งจำทำการกำหนดค่าใน พารามิเตอร์ทำการส่งไปยังฐานข้อมูลเมื่อถูกทำงาน instanceของเมทธอด

2. PrepareStatement จะเป็นการขยายคำสั่งและรวมถึงเมทธอดของ Statement ด้วย

3. CollableStatement ถูกสร้างโดยเมทธอด prepareCall จะ๔กใช้เมื่อทำงานกับ SQL store procedure โดยกลุ่มของของ SQL นี้จะเรียกโดยชื่อซึ่งคล้ายกับหน้าที่ของออบเจ็ค ตามรายการแสดง วิธีในการตัดสินใจเกี่ยวกับเมธอดทางด้านการเชื่อมต่อที่เหมาะสมสำหรับการสร้างคำสั่ง SQL ชนิด ต่าง ๆ

createStatement เมทธอด

- คำสั่ง SQL พื้นฐาน (ไม่มีพารามิเตอร์)

prepareStatement เมทธอด

- คำสั่ง SQL ที่มีหนึ่งหรือหลายพารามิเตอร์
- คำสั่ง SQL พื้นฐานที่ถูกเรียกใช้บ่อย ๆ

prepareCall เมทธอด

- เรียก store procedure

2.2.4.18 ทรานแซกชั่น (Transaction)

ทรานแซกชั่นประกอบด้วยหนึ่งหรือหลายคำสั่งซึ่งถูกใช้งาน จะสมบูรณ์เมื่อมีการ commit หรือ roll back เมื่อเมทธอด commit หรือ roll back ถูกเรียกใช้โดยทรานแซกชั่นปัจจุบันจะถูกว่าเสร็จสิ้นกระบวนการ ในการสร้างการเชื่อมต่อใหม่แต่ละครั้งจะถูกกำหนดวิธีเริ่มต้นคือ auto-commit หมายความว่าเมื่อคำสั่งจะเสร็จสิ้นเมื่อเมทธอด commit ถูกเรียกอย่างอัตโนมัติในกรณีนี้ แต่คำสั่งจะถูก commit ทีละคำสั่งซึ่งทำให้ทรานแซกชั่นประกอบด้วยเพียงหนึ่งคำสั่งเท่านั้น ถ้ารูปแบบ autocommit ถูกยกเลิกทรานแซกชั่นจะไม่มีการสิ้นสุดจนกระทั่งเมทธอด commit หรือ rollback ถูกเรียกใช้ ในกรณีที่สองนี้คำสั่งทุกคำสั่งใน ทรานแซกชั่นที่ถูก commit หรือ rollback ถูกรวมเข้าเป็นกลุ่มเดียวกัน เมทธอด commit ทำให้การเบปี่ยนแปลงจากคำสั่ง SQL ที่ทำกัฐานข้อมูลมีสภาพถาวร และฐานข้อมูลจะทำการปลดล็อคที่ถูกจัดการโดยทรานแซกชั่นนั้น ๆ เมทธอด rollback จะยกเลิกการเปลี่ยนแปลง บางเวลาผู้ใช้ไม่ต้องการที่จะให้การเปลี่ยนแปลงเพียง 1 สิ่งส่งผลกระทบกับสิ่งอื่น ๆ ซึ่งสามารถทำได้โดย

การยกเลิก auto-commit และทำการรวมการเปลี่ยนแปลงทั้งสองเป็นหนึ่งทรานแซกชั่น ถ้าการเปลี่ยนแปลงทั้งสองสำเร็จ เมทธอด commit จะถูกเรียกใช้ ซึ่งส่งผลกระทบต่อการเปลี่ยนแปลงทั้งสองเป็นการถาวร แต่ถ้าเกิดการล้มเหลวอย่างหนึ่งส่วน เมทธอด rollback จะถูกเรียก และทำให้ค่ากลับคืนสู่สภาพเดิมก่อนที่การเปลี่ยนแปลงเกิดขึ้น

2.2.4.19 Driver Manager

คลาส DriverManager เป็นชั้น management ของ JDBC ซึ่งทำงานระหว่างผู้ใช้และไดรเวอร์ จะทำการตรวจสอบไดรเวอร์ที่มีและจัดการจัดตั้งการเชื่อมต่อระหว่างฐานข้อมูลกับไดรเวอร์ที่เหมาะสม สิ่งที่เพิ่มเติมขั้นคือคลาส DriverManager มีส่วนร่วมเกี่ยวกับระยะเวลา ในการขอเข้าใช้ระบบ และเพิ่มรายการและบันทึกรายละเอียดในการเข้าใช้บริการต่าง ๆ

2.2.4.20 การตรวจสอบไดรเวอร์

คลาส DriverManager จะดูแลรักษารายการของคลาส Driver ซึ่งทำการลงทะเบียนโดยการเรียกเมทธอด DriverManager.registerDriver ทุก ๆ คลาส Driver จะต้องเขียนด้วยส่วนที่มีลักษณะคงที่ ซึ่งถูกสร้างจาก instance ของคลาสและทำการลงทะเบียน ด้วยคลาส DriverManager เมื่อมันถูกโหลด ดังนั้นผู้ใช้จะไม่สามารถเรียก DriverManager.registerDriver โดยตรงอย่างปรกติได้ แต่จะถูกเรียกโดยอัตโนมัติเมื่อไดรเวอร์ถูกโหลด คลาส Driver จะถูกโหลดและลงทะเบียนอย่างอัตโนมัติโดย DriverManager

2.2.4.21 การจัดตั้งการเชื่อมต่อ

เมื่อโหลดคลาส Driver และทำการลงทะเบียนโดยคลาส DriverManager แล้วหลังจากนั้นก็จะทำการจัดตั้งการเชื่อมต่อกับฐานข้อมูล เมื่อมีการร้องขอการเชื่อมต่อจะทำให้มีการไปเรียกเมทธอด DriverManager.getConnection โดยที่ DriverManager ทำการทดสอบแต่ละรอบในการเชื่อมต่อของแต่ละไดรเวอร์ในกรณีที่สามารถจัดตั้งการเชื่อมต่อได้ มีบางกรณีที่มีมากกว่าหนึ่งไดรเวอร์ JDBC ที่สามารถเชื่อมต่อตาม URL ที่ระบุมา ตัวอย่างคือ เมื่อทำการเชื่อมต่อกับฐานข้อมูลระยะไกล ซึ่งการเชื่อมต่อครั้งนี้มีความเป็นไปได้ที่จะใช้ไดรเวอร์ชนิด JDBC-ODBCbridge, JDBC-to-generic-network-protocol หรือไดรเวอร์ที่ได้รับการจัดหาให้จากตัวแทนผู้จำหน่าย ในกรณีนี้ลำดับของแต่ละไดรเวอร์จะถือมีความสำคัญเพราะว่า DriverManager จะใช้ไดรเวอร์แรกที่พบว่าสามารถทำให้การเชื่อมต่อกับ URL ที่ระบุประสบความสำเร็จ สิ่งแรกคือ DriverManager จะพยายามใช้ไดรเวอร์แต่ละชนิดตามลำดับที่ไดรเวอร์เหล่านั้นได้ลงทะเบียน โดยจะข้ามไดรเวอร์ตัวที่ไม่น่าเชื่อถือ การทดสอบไดรเวอร์โดยการเรียกเมทธอด Driver.connect ในแต่ละรอบ ซึ่งเรียกตาม URL ที่ผู้ใช้ผ่านเมทธอด

DriverManager.getConnection มาโดยที่ไดรเวอร์ตัวแรกจะทำหน้าที่จัดการกับ URL ดังกล่าวทำให้เกิดการเชื่อมต่อที่สมบูรณ์ขึ้นมา

2.2.4.22 ตัวอย่างไดรเวอร์ JDBC ของออราเคิล (Oracle)

ปัจจุบันไดรเวอร์ JDBC ของออราเคิลคือรุ่นที่ 7.3.4.0.0 ซึ่งได้ยึดมาตรฐานของ JDBC ตามข้อกำหนดของจาวาซอร์ฟ โดยไดรเวอร์ตัวนี้มีความถือตาม JDBC เวอร์ชั่น 1.22 ไดรเวอร์ JDBC ของออราเคิลมีอยู่ 2 ชนิดคือ

- JDBC Thin สำหรับจาวแอพเพ็ต และงานประยุกต์
- JDBC OCI สำหรับงานประยุกต์ชนิดจาวา (Java Application)

2.2.4.23 ลักษณะของไดรเวอร์ชนิด JDBC Thin

ไดรเวอร์ของออราเคิลชนิด JDBC Thin เป็นไดร์เวอร์ชนิดที่ 4 (Native-Protocol All-Java Driver) ผู้ใช้สามารถติดต่อโดยตรงกับออราเคิลโดยใช้จาวา socket ซึ่งทำการจัดการกับเวอร์ชั่นของ TCP/IP ของออราเคิล SQL net ด้วยตัวเอง เพราะว่าไดรเวอร์ชนิดนี้ถูกเขียนขึ้นโดย Java ทั้งหมด และไม่ขึ้นกับรูปแบบและสภาพแวดล้อมใด ๆ สามารถทำงานได้ทุก ๆ ระบบที่จาวาทำงานได้ โดยสามารถดาวน์โหลดบนบราวเซอร์ (browser) ทุกชนิดเช่นเดียวกับจาวา ทำให้มีความเหมาะสมกับแอพเพ็ตบนอินทราเน็ตการใช้งานไม่มีความจำเป็นที่ต้องมีการติดตั้งซอร์ฟแวร์ของออราเคิลทางฝั่งไคลเอนต์ โดยจะทำการเชื่อมต่อกับฐานข้อมูลของออราเคิลใด ๆ ที่มีเวอร์ชั่นตั้งแต่ 7.2.x หรือสูงกว่า

2.2.4.24 ลักษณะของไดรเวอร์ชนิดของ JDBC OCI

ไดรเวอร์ชนิดนี้จะเป็นไดร์เวอร์ชนิดที่ 2 (Native-API,Partly Java Driver) ซึ่งมีการเรียกใช้ OCI Library ทำให้ไดรเวอร์ชนิดนี้ถูกผูกติดอยู่กับรูปแบบ และไม่เหมาะสมที่จำนำไปใช้งานกับแอพเพ็ตที่การใช้งานต้องทำการดาวน์โหลดมาบนบราวเซอร์ที่กำลังทำงานอยู่ ซึ่งไม่สามารถรู้ได้ว่ากำลังทำงานอยู่บนแพลตฟอร์มใด ไดรเวอร์ชนิดนี้การทำงานต้องขึ้นกับรูปแบบที่ทาง ออราเคิลสนับสนุนด้วยคือ

- Solaris : version 2.5 หรือสูงกว่า
- Windows 95 or NT 3.51 หรือสูงกว่า

การใช้งานจำเป็นต้องมีการติดตั้งซอร์ฟแวร์ของออราเคิลเวอร์ชั่น 7.3.4 หรือสูงกว่าทางฝั่งของไคลเอนต์ อย่างไรก็ตามไดรเวอร์ชนิดนี้มีความเหมาะสมกับงานประยุกต์ชนิดจาวา ซึ่งสามารถทำการติดต่อกับออราเคิลเวอร์ชั่น 7. X หรือสูงกว่า

บทที่ 3
การออกแบบระบบ

3.1 การออกแบบระบบจัดการข้อมูลส่วนบุคคลผ่านเครือข่าย

ในการออกแบบระบบจัดการข้อมูลส่วนบุคคลได้ทำการแบ่งการทำงานของระบบออกอย่างชัดเจนดังที่ได้กล่าวไว้ในบทที่ 2 ในตอนต้นว่าได้แบ่งเป็น 5 ส่วนอันประกอบไปด้วย

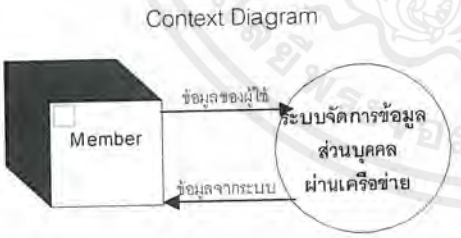
- Scheduling and Calendar
- Personal Information and Address Book
- E-mail
- Conferencing (chat)

ด้วยลักษณะของระบบจัดการข้อมูลส่วนบุคคลผ่านเครือข่าย เป็นการทำงานผ่าน Browser ดังนั้นตัวโปรแกรมของระบบจึงพัฒนาในรูปแบบของ Web Application โดยการทำงานของระบบจะประกอบไปด้วยขั้นตอนที่จะทำการอธิบายด้วย Data Flow Diagram และ Structure Chart สำหรับอธิบายลำดับการทำงานของระบบ

3.2 Data Flow Diagram

ระบบจัดการข้อมูลส่วนบุคคล สามารถนำมาสร้าง Flow สำหรับแสดงการทำงานของระบบโดยดูข้อมูลที่ใช้ในระบบและขั้นตอนของการทำงานทั้งหมดของระบบ

3.2.1 Context Diagram

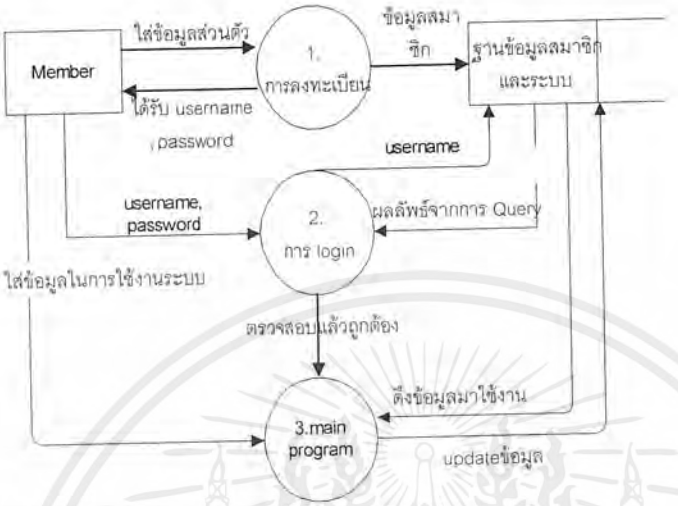


รูปที่ 3.1 แสดง context diagram ของระบบ

จาก context diagram จะเห็นได้ว่าระบบจัดการข้อมูลส่วนบุคคลจะสนใจเพียงสมาชิกของระบบ โดยสมาชิกจะทำการใส่ข้อมูลเข้ามาในระบบเพื่อนำมาจัดเก็บหรือจัดการ กิจกรรมภายในระบบและทำการรับข้อมูลที่ระบบทำงานเอาไว้นำไปใช้

ลักษณะของระบบสามารถแสดงออกมาในรูปแบบ ของ Data Flow Diagram ระดับ 0 ซึ่งจะแสดงการทำงานของระบบว่าประกอบไปด้วยอะไรบ้าง

3.2.2 DFD ระดับ 0



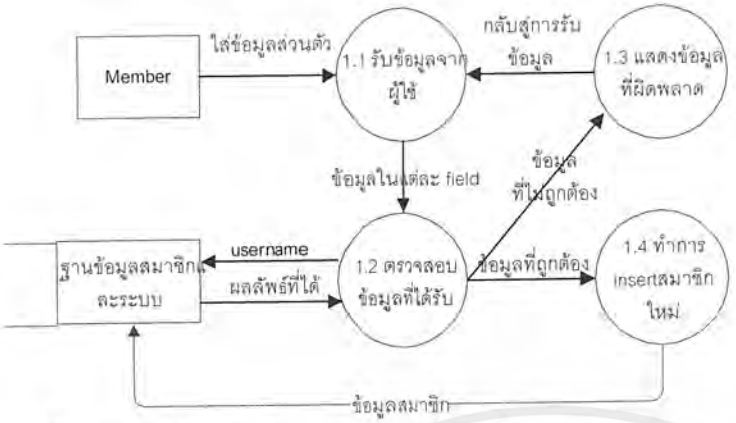
รูปที่ 3.2 แสดง DFD ระดับ 0

จากรูปสามารถแบ่งส่วนการทำงานได้

- Member ก็คือสมาชิกของระบบหรือผู้เข้ามาใช้ระบบ
- ส่วนของการลงทะเบียน ผู้ที่จะสามารถเข้ามาใช้ระบบได้จะต้องทำการ ลงทะเบียนเสียก่อน
- การ login สำหรับการเข้าใช้งานระบบทุกครั้งสมาชิกจะต้องทำการ login เพื่อตรวจสอบก่อน
- Main program คือส่วนของโปรแกรมการใช้งาน
- ระบบฐานข้อมูลสมาชิกและระบบ ที่จะทำการจัดเก็บ

ดังจะแสดงใน Data Flow Diagram ระดับ ได้เป็น 3 ส่วนใหญ่ ดัง ต่อไป

3.2.3 DFD ระดับ 1



รูปที่ 3.3 แสดง DFD ระดับ 1 ของการลงทะเบียน

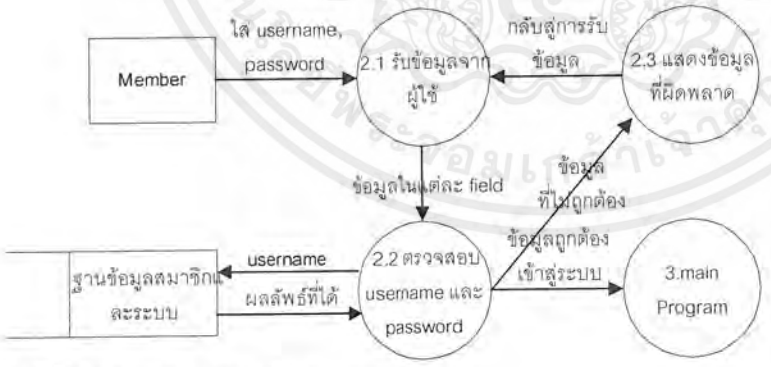
1. การลงทะเบียน จะประกอบไปด้วย 4 process ย่อย ๆ

   1.1 รับข้อมูลจากผู้ใช้ จากผู้ใช้

   1.2 ข้อมูลที่ได้รับจะถูกนำมาตรวจสอบว่าถูกต้องหรือไม่ ตามแต่ละชนิดข้อมูล และตรวจสอบ username ว่าไม่ซ้ำกับที่มีอยู่แล้ว

   1.3 ทำการแสดงข้อมูลที่ผิดพลาดถ้าพบว่ามีข้อมูลไม่ถูกต้องชนิดใดก็จะทำการแสดงว่าเป็นข้อมูล field ใด

   1.4 ถ้าข้อมูลถูกต้อง username ไม่ซ้ำ ก็จะทำการ insert เข้าฐานข้อมูล



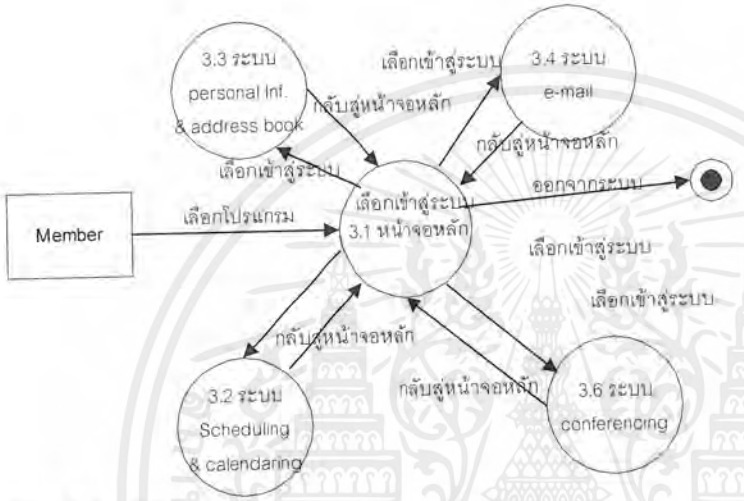
รูปที่ 3.4 แสดง DFD ระดับ 1 ของการ login

2. การ login ประกอบไปด้วย 3 Process ย่อย ๆ

2.1 การรับข้อมูลจากผู้ใช้ ทำการรับ username และ password จากผู้ใช้เข้ามาในระบบผ่าน form การรับข้อมูล

2.2 ทำการตรวจสอบ username และ password ว่าถูกต้องหรือไม่ โดยทำการส่ง username ไปทำการ query ข้อมูลแล้วเปรียบเทียบ password ว่าตรงหรือไม่ ถ้าหากไม่ถูกต้อง ก็ให้ไปทำการแสดงข้อมูลผิดพลาด หากถูกต้องก็จะทำการเข้าสู่ตัว main program

2.3 แสดงหน้าจอความผิดพลาด เช่น ไม่พบ username คนนี้ หรือ password ไม่ถูกต้อง



รูปที่ 3.5 แสดง DFD ระดับ 1 ของ Main Program

3. Main Program จะประกอบไปด้วยส่วนประกอบต่าง ๆ

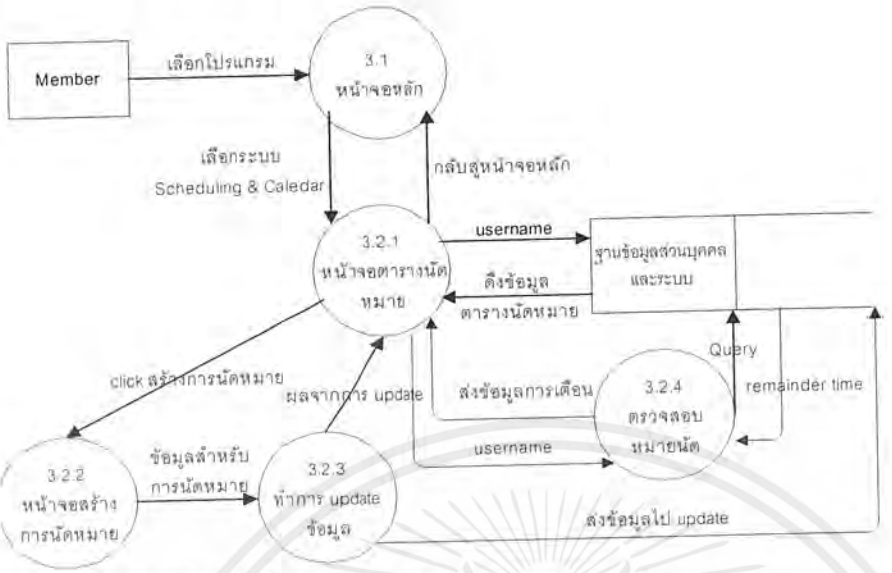
3.1 หน้าจอหลัก หลังจากได้รับการตรวจสอบการ login เรียบร้อย ก็จะเข้าสู่หน้าจอแรกที่เป็นหน้าจอหลักในการทำงาน ทำหน้าที่ในการแสดงรายละเอียดของ menu ทั้งหมดของระบบ เพื่อที่จะเลือกทำระบบใดต่อไป หรือ จะออกจากระบบ ด้วยการ log off

3.2 ระบบ Scheduling & calendaring เป็นระบบสำหรับจัดการตารางนัดหมาย

3.3 ระบบจัดการ personal information and address book

3.4 ระบบจัดการ Conferencing

3.2.4 DFD ระดับ 2



รูปที่ 3.6 แสดง DFD ระดับ 2 ของ Scheduling & Calendar

1. ระบบจัดการ Scheduling & Calendar มีรายละเอียดภายในดังต่อไปนี้

   1.1 หน้าจอตารางนัดหมาย ซึ่งจะเป็นหน้าจอหลักในการทำงานของระบบ Scheduling & Calendar โดยที่หน้าจอดังกล่าวจะแสดง ตารางนัดหมาย ปฏิทิน event task และ ทำการแสดง การเตือน (remainder)

   1.2 หน้าจอการสร้างการนัดหมาย เมื่อผู้ใช้ต้องการสร้างการนัดหมาย จะทำการเลือกการสร้างการนัดหมาย เพื่อเปิดฟอร์ม สำหรับทำการสร้างการนัดหมาย, event หรือ task

   1.3 ทำการ update ข้อมูลที่ได้ทำการสร้างเข้าไว้ใน Database หลังจากนั้นจะทำการ แสดงหน้าจอการทำงานหลักใหม่อีกครั้ง

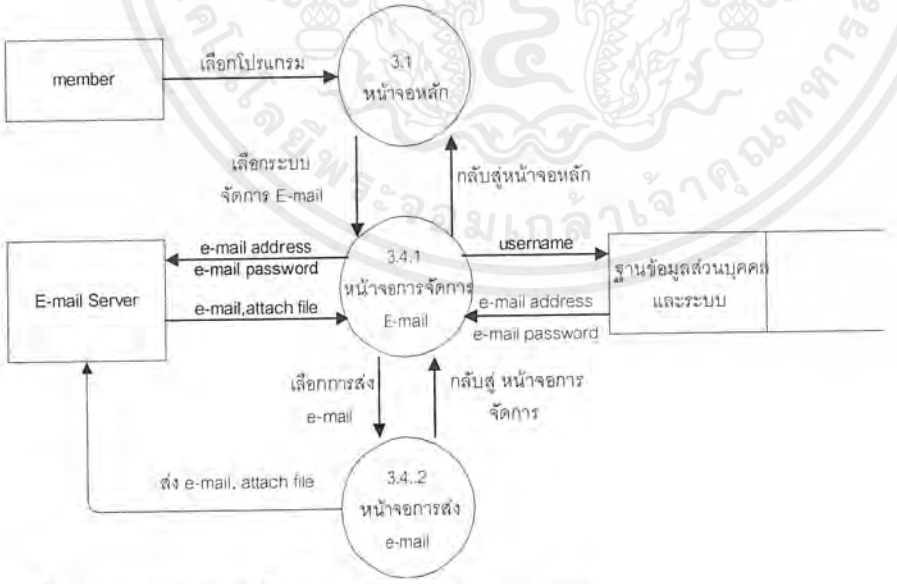
   1.4 เป็นส่วนของการตรวจสอบหมายนัด เมื่อทำการเปิดระบบเข้ามา หรือ ทุก ๆ คาบเวลาที่ตั้งเอาไว้จะทำการตรวจสอบว่ามีหมายนัด event หรือ task ที่จะต้องทำการเตือนเพื่อส่งให้หน้าจอหลัก

รูปที่ 3.7 แสดง DFD ระดับ 2 ของ Personal Information & Address Book

2. Personal Information & Address Book เป็นส่วนในการจัดการ ข้อมูลของสมาชิก และ Address Book ของสมาชิก โดยมีส่วนประกอบย่อย ๆ ดังต่อไปนี้

2.1 หน้าจอหลักในการแสดงข้อมูลส่วนบุคคล โดยจะแสดงข้อมูลทั่วไปของสมาชิกและ Address Book ของ สมาชิก โดยทำการดึงข้อมูลจาก Database และ ยังเป็นส่วนในการแก้ไข และ เพิ่มเติมข้อมูล

2.2 เป็นส่วนของการ update ข้อมูล จากหน้าจอหลักไปไว้ที่ Database แล้วทำการ update หน้าจอในการแสดงผล

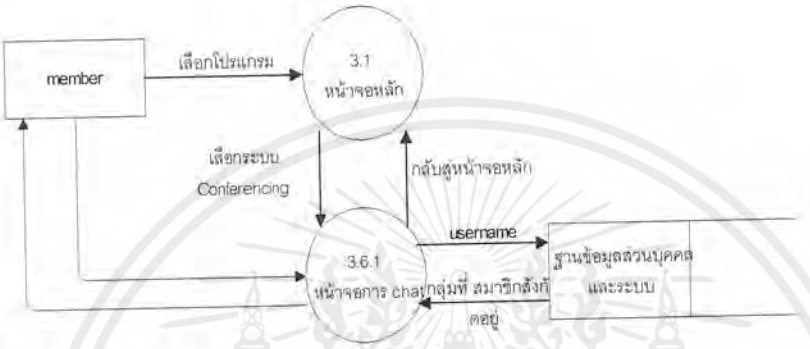


รูปที่ 3.8 แสดง DFD ระดับ 2 ของระบบจัดการ e-mail

3. ระบบจัดการ E-mail ประกอบไปด้วยส่วนต่าง ๆ ดังต่อไปนี้

3.1 หน้าจอในการจัดการ E-mail ทำการแสดง e-mail โดยทำการดึง e-mail address และ e-mail password จาก Database เพื่อไปทำการ ดึงข้อมูลจาก E-mail Server เพื่อนำมาแสดงผล

3.2 ส่วนของหน้าจอการส่ง e-mail ถ้าสมาชิกต้องการส่ง e-mail ก็จะทำการเปิดฟอร์มในการส่ง mail ขึ้นมาสำหรับทำการส่งโดยเฉพาะ



รูปที่ 3.9 แสดง DFD ระดับ 2 ของระบบ Conferencing (Chat)

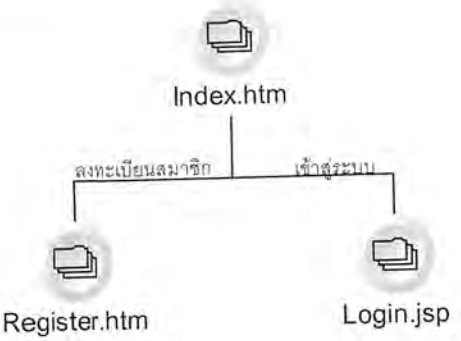
4. ระบบ Conferencing (chat) เป็นระบบสำหรับการติดต่อสื่อสารภายใน กลุ่มของสมาชิกโดยประกอบไปด้วย

4.1 หน้าจอการ chat เป็นหน้าจอที่ทำหน้าที่เป็น ฟอร์มสำหรับการสนทนาของสมาชิกกับสมาชิกคนอื่นในกลุ่ม โดยสามารถสนทนาได้กับ กลุ่มที่สมาชิกสังกัดอยู่

3.3 แผงผังโครงสร้าง

ด้วยการทำงานของระบบ เป็นลักษณะของ Web Programming จึงมีลักษณะการทำงานผ่านเครือข่ายโดยจะมีรูปแบบของการทำงานของโปรแกรมที่จะแสดงได้ดัง chart ด้านล่างต่อไปนี้

3.3.1 แผงผังโครงสร้างการเข้าสู่ระบบ



รูปที่ 3.10 แสดงแผงผังโครงสร้างของการเข้าสู่ระบบ

จากรูป

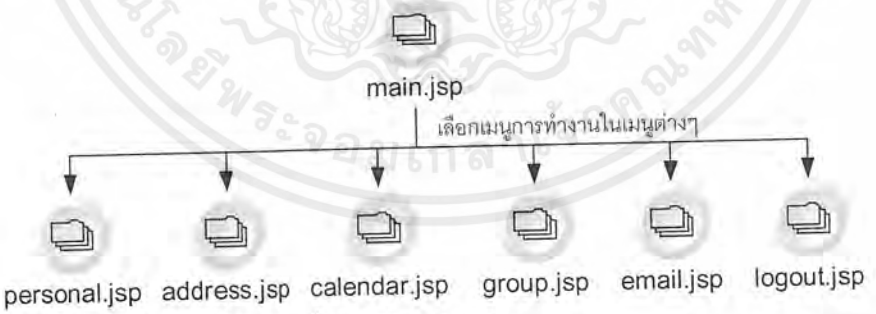
Index.htm เป็น home page เริ่มต้นของระบบ

Register.htm เป็น page สำหรับทำการลงทะเบียนเป็นสมาชิกระบบ

Login.jsp เป็น page สำหรับทำการ login เพื่อทำการเข้าสู่ระบบ

โครงสร้างแรกเป็นการแสดง โครงสร้างของระบบ ณ ตำแหน่งแรกในการเรียกเข้าสู่การทำงานของระบบ โดยเริ่มจาก index.htm ซึ่งเป็น หน้าเริ่มต้นในการเข้าสู่ระบบ ทำหน้าที่เป็น page สำหรับจะทำการ login เข้าสู่ระบบโดยจะเข้าสู่ Login.jsp หรือจะทำการสมัครสมาชิกโดยเข้าสู่ในส่วนของ Register.htm ซึ่งรูปแบบการทำงานจะอธิบายในบทต่อไป

3.3.2 แผงผังโครงสร้างหน้าจอ Main Menu



รูปที่ 3.11 แสดงแผงผังโครงสร้างของหน้าจอ Main Menu

จากรูปเป็นโครงสร้างของส่วน Main Program ซึ่งเป็นส่วนประกอบสำคัญใน Web Application นี้โดยที่เมื่อผ่านการ login จาก login.jsp ก็จะเข้าสู่ในส่วนของ main.jsp ซึ่งจะประกอบด้วย page ต่าง ๆ ดังต่อไปนี้

Main.jsp	เป็น page สำหรับทำการแสดงข้อมูลเริ่มต้นสำหรับเข้าใช้โปรแกรม หรือทำการค้นหาข้อมูล และแสดงรายการนัดหมายประจำวัน
Personal.jsp	เป็น page สำหรับทำการแสดงข้อมูลส่วนตัว รายละเอียดต่าง ๆ ของสมาชิกที่บันทึกไว้ในระบบ
Address.jsp	เป็น page สำหรับทำการแสดงรายการบันทึกที่อยู่ หรือ รายชื่อของบุคคล หรือ บริษัท ที่สมาชิกต้องการจัดเก็บไว้
Calendar.jsp	เป็น page สำหรับทำการแสดงตารางนัดหมาย ปฏิทิน ของมูลของการนัดหมายต่าง ๆ หรือ โครงงานที่สมาชิกได้จัดเก็บไว้ทั้งของสมาชิกเอง หรือ ของกลุ่ม และ สมาชิกคนอื่นที่ทำการ share ไว้
Group.jsp	เป็น page สำหรับทำการแสดงการจัดการกลุ่มสมาชิก โดยจะแสดงกลุ่มที่สมาชิกอยู่หรือ สามารถสร้างกลุ่มผู้ใช้ขึ้นมาได้ พร้อมทั้งจัดการสิทธิในกลุ่มของผู้ใช้
Email.jsp	เป็น page สำหรับทำการแสดงระบบจดหมายอิเล็กทรอนิกส์ หรือ e-mail โดยสามารถทำการจัดส่งหรือจะทำการตรวจดู e-mail ได้จากโปรแกรม
Logout.jsp	เป็น page สำหรับทำการ logout ออกจากระบบโดยจะทำการเข้าสู่หน้า index.htm

3.4 ระบบฐานข้อมูล และ E-R Diagram

ระบบจัดการข้อมูลส่วนบุคคลสามารถแบ่งข้อมูลที่สนใจออกมาจากส่วนประกอบของระบบทั้ง 5 ส่วนนั่นก็คือ

3.4.1 ระบบจัดการตารางนัดหมายส่วนบุคคล

จะประกอบไปด้วยข้อมูลหลัก ๆ ที่ได้จัดเก็บไว้ 3 ส่วนอันประกอบไปด้วย

1. appointment หรือการกำหนดการนัดหมายส่วนบุคคลของผู้ใช้เองและยังสามารถส่งหมายนัดไปให้สมาชิก ของระบบคนอื่น ๆ หรือจะส่งไปให้สมาชิกภายในกลุ่มโดยการกำหนดช่วงเวลาที่จะทำการนัดหมาย และกำหนดช่วงเวลาที่จะทำการเตือน

2. event คือเหตุการณ์ที่สำคัญเฉพาะแต่ละสมาชิกเพื่อจะเตือนผู้ใช้ว่าจะมีเหตุการณ์นั้นเกิดขึ้นเมื่อไร เช่น วันเกิด, วันครบรอบ และยังสามารถกำหนด event ของกลุ่มได้ด้วย

3. Task คืองานหรือโครงงานที่ผู้ใช้ต้องทำ โดยการกำหนด ช่วงเวลาในการทำงานนั้นไว้ และลำดับความสำคัญของงานนั้น ๆ และสามารถกำหนดงานของกลุ่มได้อีกด้วย

3.4.2 ระบบจัดการข้อมูลส่วนตัว

จะมีข้อมูลที่สนใจอยู่ 2 ส่วนคือ

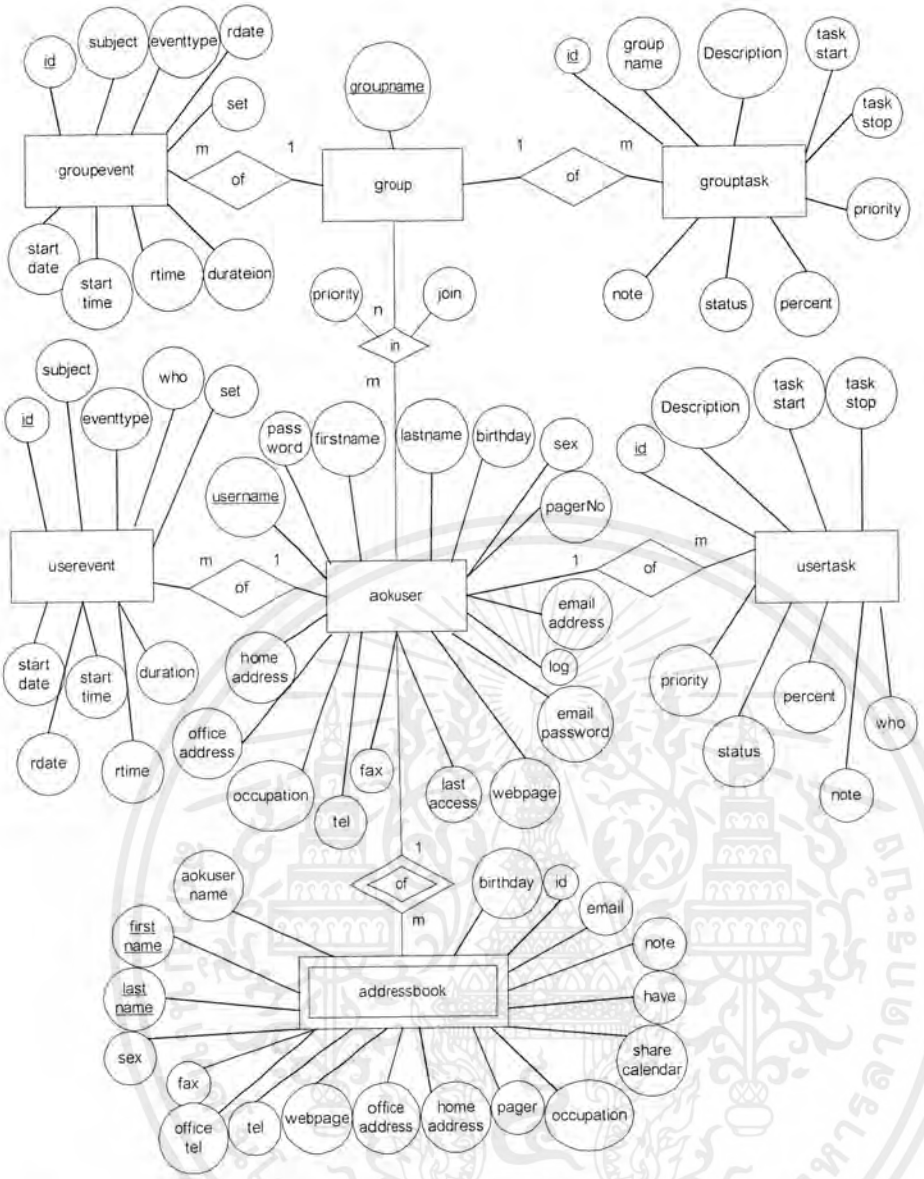
1. ข้อมูลของผู้ใช้เอง จะประกอบไปด้วยชื่อ ทีอยู่ ที่ทำงาน วันเกิด เบอร์โทรศัพท์ e-mail address e-mail password อาชีพ เป็นต้น

2. ส่วนของ Addressbook เป็นส่วนของข้อมูลที่ ผู้ใช้ต้องการเก็บข้อมูลผู้อื่นมาเก็บไว้ในการใช้งานอันประกอบไปด้วย ชื่อ ที่อยู่ เบอร์โทรศัพท์ หรือ username ของระบบจัดการข้อมูลส่วนบุคคล

3.4.3 ระบบจัดการ E-mail เราจะสนในเพียง E-mail Address และ E-mail password สำหรับในการดึงข้อมูลจาก E-mail Address สำหรับในการส่ง ไม่จำเป็นต้องมีการจัดเก็บข้อมูล

3.4.4 ระบบการสนทนาภายในกลุ่ม (Chat) จะสนใจเพียงว่าเป็นสมาชิกในแต่ละกลุ่มที่ได้ทำการสร้างเอาไว้หรือไม่ โดยดูได้จากตารางที่บอกเอาไว้

จากข้อมูลดังกล่าว ในระบบจัดการฐานข้อมูลส่วนบุคคลผ่านเครือข่าย จึงจะมีการใช้ Database ในการจัดเก็บข้อมูลโดยในการจัดเก็บข้อมูลใช้ในรูปแบบของตารางหรือ RDBMS (Relational Database Management System) โดยข้อมูลจะประกอบไปด้วย E/R diagram และตารางดังต่อไปนี้



รูปที่ 3.12 แสดง E-R diagram ของระบบจัดการข้อมูลส่วนบุคคลและกลุ่ม

3.5 ตารางต่างๆของระบบจัดการข้อมูลส่วนบุคคลและกลุ่ม

ตารางที่ 3.1 แสดงตารางของ AOKUSER

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
Username	Text(10)	ชื่อที่ใช้ในการเข้าใช้ระบบ	✔		
Password	Text(8)	รหัสที่ใช้ในการเข้าใช้ระบบ			
Firstname	Text(20)	ชื่อจริงของผู้ใช้ระบบ			
Lastname	Text(20)	นามสกุลจริงของผู้ใช้ระบบ			
Birthday	Text(10)	วันเกิดของผู้ใช้ระบบ			
Sex	Text(10)	เพศของผู้ใช้ระบบ			
Homeaddress	Text(80)	ที่อยู่ของผู้ใช้ระบบ			
Officeaddress	Text(80)	ที่ทำงานของผู้ใช้ระบบ			
Occupation	Text(50)	อาชีพของผู้ใช้ระบบ			
Tel	Text(50)	เบอร์โทรศัพท์ของผู้ใช้ระบบ			
Fax	Text(30)	เบอร์โทรสารของผู้ใช้ระบบ			
Webpage	Text(50)	IP address หรือ URLs ของผู้ใช้ระบบ			
Emailaddress	Text(30)	e-mail address ของผู้ใช้ระบบ			
Emailpwd	Text(10)	e-mail password ของผู้ใช้ระบบสำหรับเข้าไปเอาข้อมูล			
PagerNo	Text(30)	เบอร์ pager ของผู้ใช้ระบบ			
Log	Boolean	เป็นการ check การเข้าใช้ระบบ			
Lastaccess	Date/Time	เก็บวันที่เข้าสู่ระบบครั้งล่าสุด			

ตารางที่ 3.2 แสดงตารางของ Group

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
groupname	Text(20)	เป็นชื่อกลุ่ม	✔		
Username	Text(10)	เป็นชื่อที่ใช้ในการเข้าใช้ระบบ	✔	✔	
Priority	Text(10)	ลำดับความสำคัญในกลุ่ม			
Join	Boolean	การเข้าร่วมกลุ่ม			

ตารางที่ 3.3 แสดงตารางของ Address Book

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
Id	Text(50)	หมายเลขของ addressbook			
Username	Text(10)	ชื่อที่ใช้ในการเข้าใช้ระบบ	✔		
firstname	Text(50)	ชื่อของผู้ที่จะทำการจัดเก็บ	✔		
lastname	Text(50)	นามสกุล	✔		
Have	Boolean	การเป็นสมาชิกของระบบ			
aokusername	Text(10)	ชื่อในระบบของผู้ใช้งาน			
sharecalendar	Boolean	การใช้ตารางนัดหมายร่วมกัน			
occupation	Text(50)	อาชีพ			
Homeaddress	Text(80)	ที่อยู่			
officeaddress	Text(80)	ที่ทำงาน			
Birthday	Text(20)	วันเกิด			
Sex	Text(6)	เพศ			
Tel	Text(40)	เบอร์โทรศัพท์			
Officetel	Text(40)	เบอร์โทรศัพท์ที่ทำงาน			
Email	Text(50)	e-mail address			
pager	Text(40)	เบอร์ pager			
Webpage	Text(50)	IP address หรือ URLs			
Note	Text(80)				

ตารางที่ 3.4 แสดงตารางของ Groupevent

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
groupname	Text(10)	ชื่อกลุ่ม			
subject	Text(80)	ชื่อหัวข้อของเหตุการณ์			
eventtype	Text(50)	ประเภทของเหตุการณ์			
startdate	Date/Time	วันเริ่มต้น			
starttime	Number	เวลาเริ่มต้น			
duration	Number	ระยะเวลา			
set	Yes/No	กำหนดการเตือน			
rdate	Date/Time	วันที่ให้ทำการเตือน			
rtime	Date/Time	เวลาที่ให้ทำการเตือน			
Id	Text(50)	หมายเลขเหตุการณ์	✔		

ตารางที่ 3.5 แสดงตารางของ Grouptask

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
groupname	Text(50)	ชื่อกลุ่ม			
description	Text(50)	เนื้อหาของ task			
taskstart	Date/Time	วันเริ่มต้น			
taskstop	Date/Time	วันสิ้นสุด			
priority	Text(50)	ลำดับความสำคัญของ task			
status	Text(50)	สถานะของ task			
percent	Text(50)	ระดับความสำเร็จ			
Note	Text(100)	หมายเหตุ			
Id	Text(50)	หมายเลขประจำ task	✔		

ตารางที่ 3.6 แสดงตารางของ Userevent

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
username	Text(10)	ชื่อของผู้ใช้ระบบ			
subject	Text(80)	เนื้อเรื่อง			
eventtype	Text(50)	ประเภทของเหตุการณ์			
startdate	Date/Time	วันเริ่มต้น			
starttime	Number	เวลาเริ่มต้น			
duration	Number	ระยะเวลา			
set	Yes/No	การเตือน			
rdate	Date/Time	วันที่ทำการเตือน			
rtime	Date/Time	เวลาที่ทำการเตือน			
who	Text(10)	ผู้ที่ส่งหมายนัด			
Id	Text(50)	หมายเลขเหตุการณ์	✔		

ตารางที่ 3.7 แสดงตารางของ Usertask

ชื่อข้อมูล	ชนิดข้อมูล	รายละเอียด	PK	FK	หมายเหตุ
Username	Text(50)	ชื่อของผู้ใช้ระบบ			
description	Text(50)	เนื้อหาของ task			
taskstart	Date/Time	เป็นวันที่เริ่มต้น			
taskstop	Date/Time	เป็นวันที่สิ้นสุด			
priority	Text(50)	ลำดับความสำคัญของ task			
status	Text(50)	สถานะของ task			
percent	Text(50)	ความคืบหน้าของ task			
Note	Text(100)	หมายเหตุ			
who	Text(10)	ผู้ทำการส่ง task			
Id	Text(50)	หมายเลข task	✔		

โดยตารางข้อมูลทั้งหมดจะถูกนำมาจัดสร้างเอาไว้ด้วยระบบจัดการฐานข้อมูล (RDBMS) ซึ่งภายในระบบนี้มีการทดลองใช้งานฐานข้อมูลอยู่ 2 ตัวได้แก่

- SQL Server 7.0
- Oracle 8i

โดยฐานข้อมูลทั้งหมดจะถูกจัดเก็บไว้ที่ Database Server โดยจะมีข้อมูลบางส่วนที่จะถูกจัดเก็บอยู่ที่ส่วนอื่น อันประกอบไปด้วย

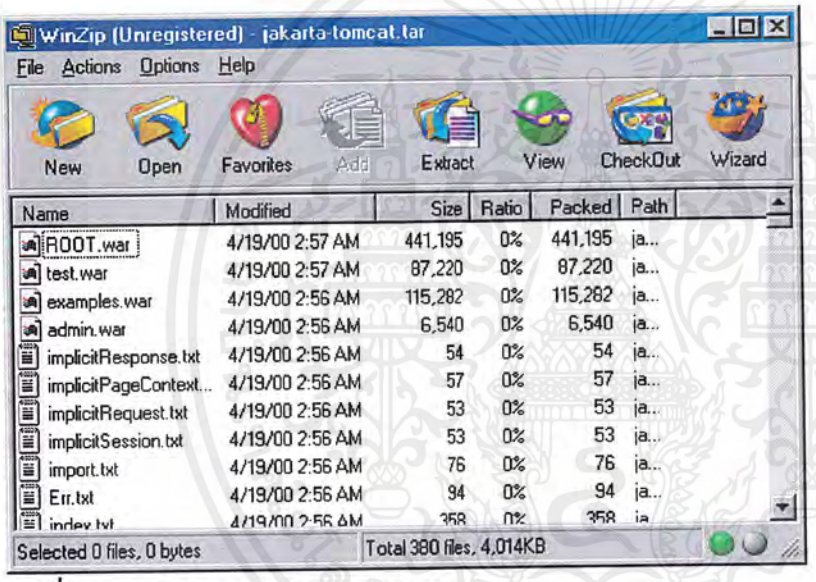
- ข้อมูลของ ระบบจัดการ e-mail โดยจะมีตัว E-mail Server ทำหน้าที่ในการจัดเก็บ e-mail, attach file
- ข้อมูลของ file และ folder จะถูกจัดเก็บไว้ใน server ซึ่งอาจจะอยู่ที่ Database Server หรือ Web Server โดยจัดเก็บใน Hardisk ตาม folder ของ user ที่ระบบสร้างเอาไว้

บทที่ 4
ขั้นตอนการพัฒนาระบบ

4.1 การพัฒนา Tomcat Web Server

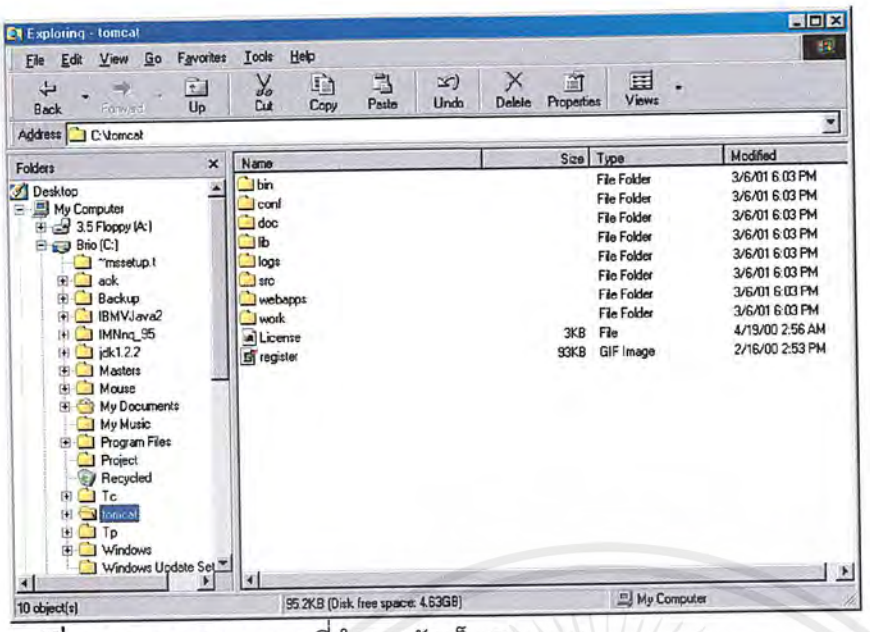
เนื่องจากระบบเป็นการพัฒนาระบบ Web Application ด้วย Java Technology จึงทำการเลือกใช้ Web Server ที่มี JSP engine โดยใช้ Jakarta- Tomcat (Web Server) เป็น Web Server โดยทำการติดตั้งดังต่อไปนี้

4.1.1 ทำการ Unzip file tomcat.zip ไปไว้ที่ C:\tomcat โดยสามารถ download ได้จาก www.apache.org



รูปที่ 4.1 แสดงการ Unzip file tomcat.zip

โดยจะได้ Directory เป็นดังรูปด้านล่าง

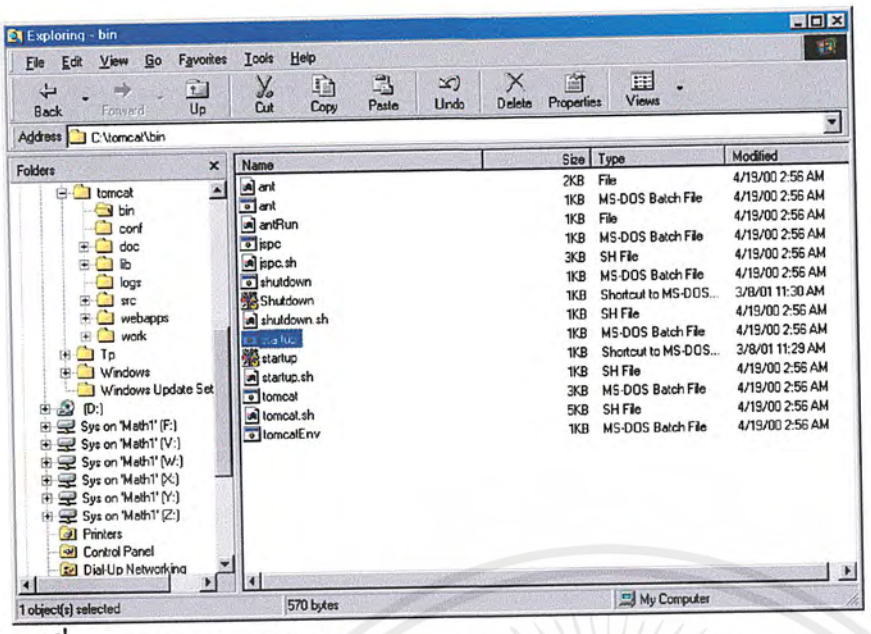


รูปที่ 4.2 แสดง Directory ที่ทำการจัดเก็บ Tomcat

4.1.2 เข้า windows explorer เพื่อเปิดแก้ไข file autoexec.bat ทำการเพิ่ม classpath ดังนี้

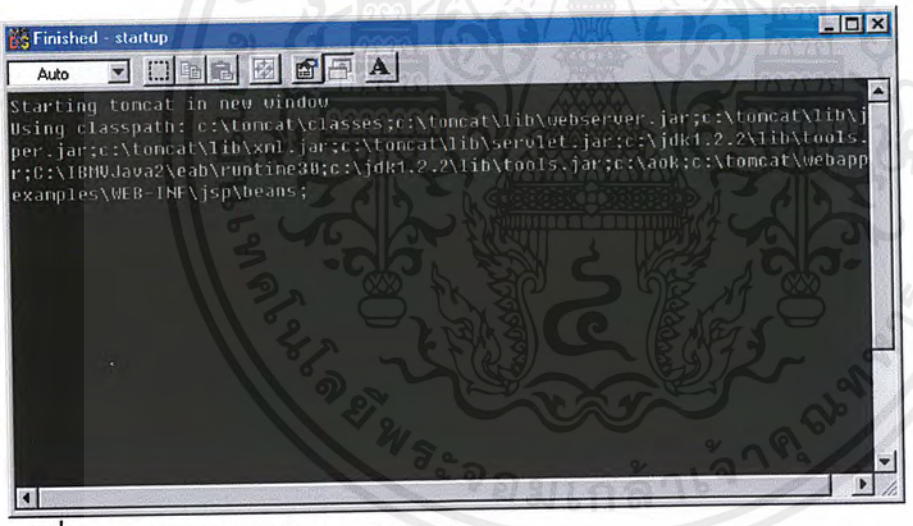
```
SET CLASSPATH=c:\jdk1.2.2\lib\tools.jar;c:\aok;c:\tomcat\webapps\examples\WEB-
INF\jsp\beans;
set TOMCAT_HOME=c:\tomcat
set JAVA_HOME=c:\jdk1.2.2
java ChatServer
```

4.1.3 ทำการเปิดระบบโดยการทำการ restart อีกครั้ง และทำการ start up file startup.bat ใน Directory c:\tomcat\bin\startup.bat สำหรับทำการ start server และ run shutdown.bat สำหรับทำการปิด server

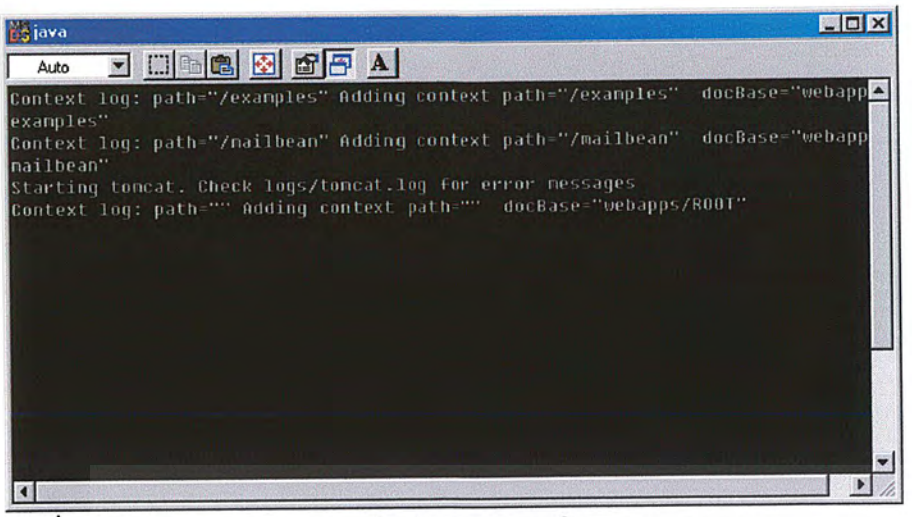


รูปที่ 4.3 แสดง startup.bat

จะได้ console สำหรับแสดงการทำงานของ tomcat server และ ระบบ Web Organizer ดังนี้

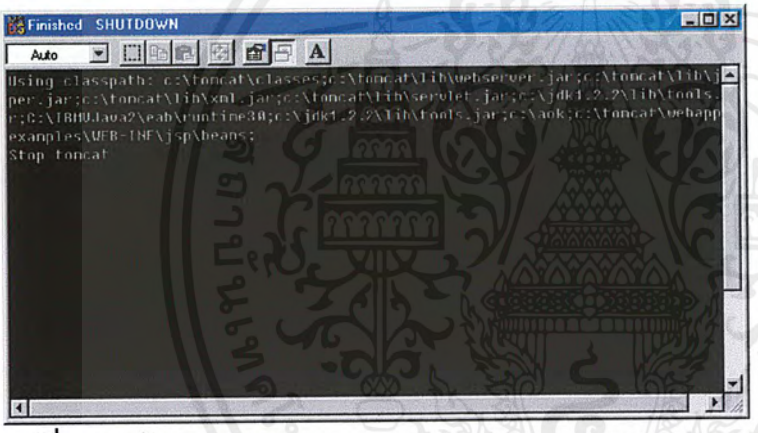


รูปที่ 4.4 แสดงการ start tomcat server



รูปที่ 4.5 แสดง Java Console สำหรับแสดงการทำงานของ server

สำหรับการ shut down server ทำได้โดยการ run file shutdown.bat ใน c:\tomcat\bin

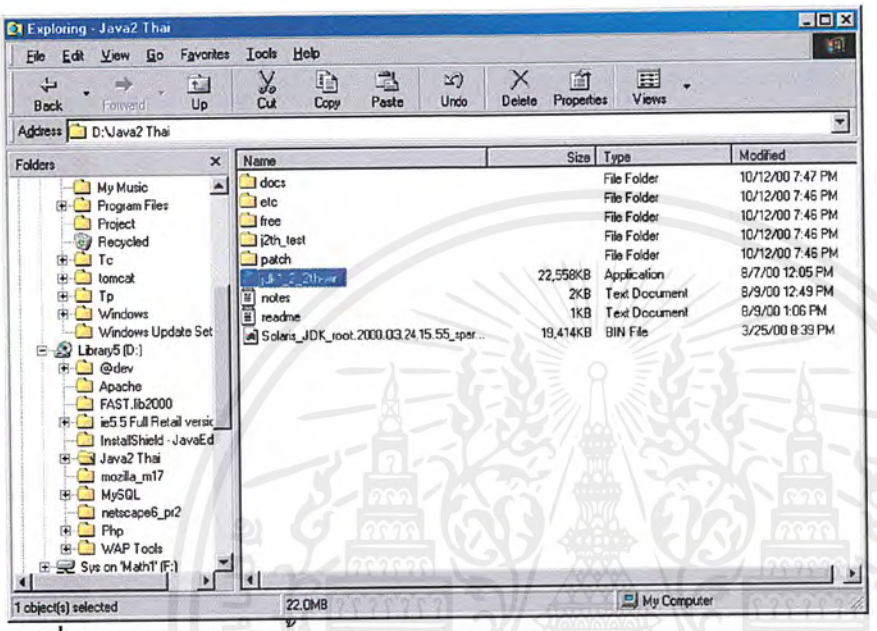


รูปที่ 4.6 หน้าจอแสดงสถานะการ Shutdown tomcat Web Server

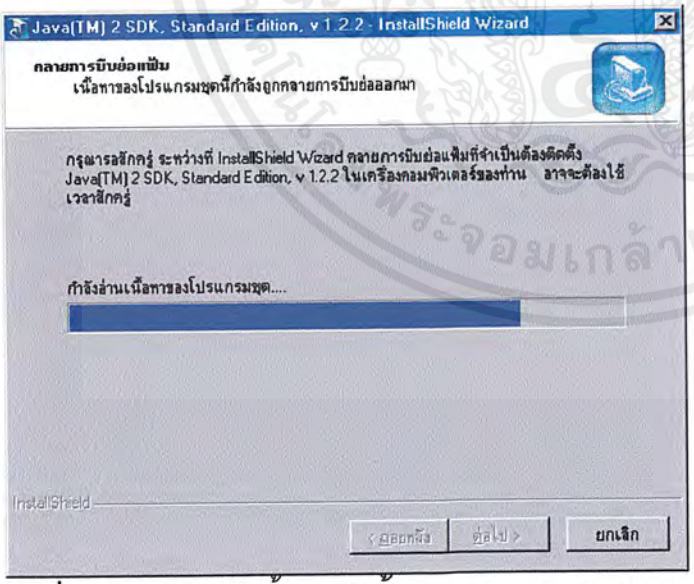
4.2 การติดตั้ง Java Development Kit

หลังจากการติดตัง Tomcat Web Server นั้นจะต้องทำการติดตั้ง Java Development Kit 1.2.2 Original version เพื่อใช้เป็นcompiler และ runtime environment สำหรับระบบงานนี้

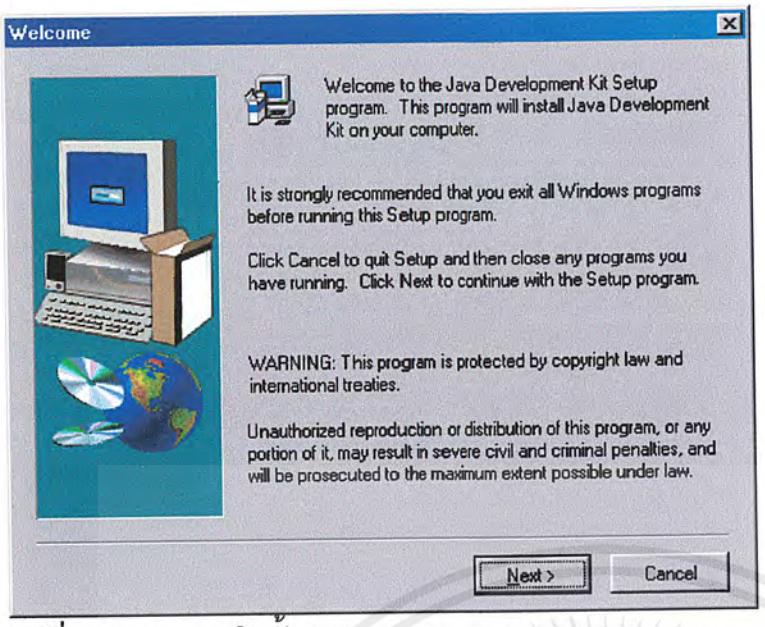
4.2.1 ทำการ download jdk1.2.2 จาก www.javasoft.com แล้วทำการ double click ตัว setup ที่ file jdk1_2_2win.exe



รูปที่ 4.7 แสดง File ติดตั้ง Java development kit 1.2.2

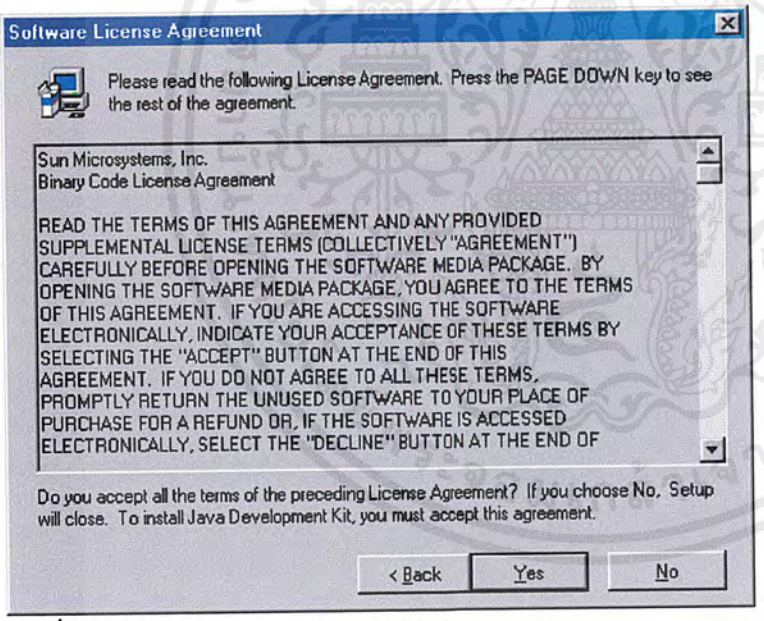


รูปที่ 4.8 แสดงการติดตั้งการติดตั้ง Java development kit 1.2.2



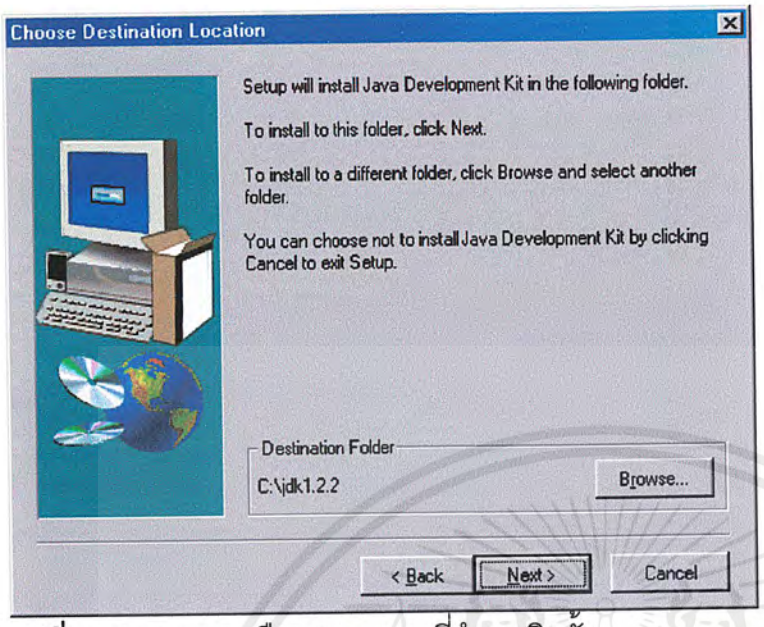
รูปที่ 4.9 แสดงการติดตั้ง Java development kit 1.2.2

4.2.2 ทำการตกลง License Agreement

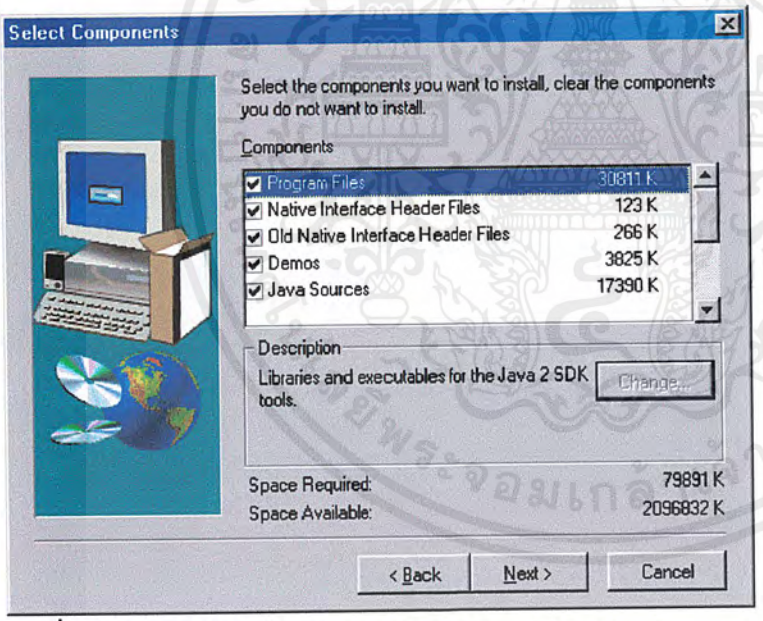


รูปที่ 4.10 แสดง Software License Agreement

4.2.3 ทำการเลือก Directory ที่จะทำการติดตั้ง และส่วนประกอบที่ต้องการลง

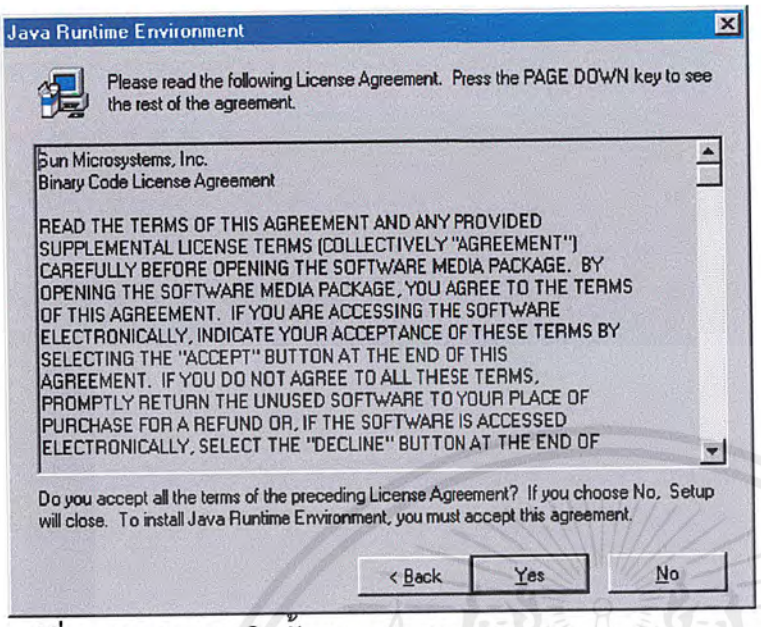


รูปที่ 4.11 แสดงการเลือก Directory ที่ทำการติดตั้ง

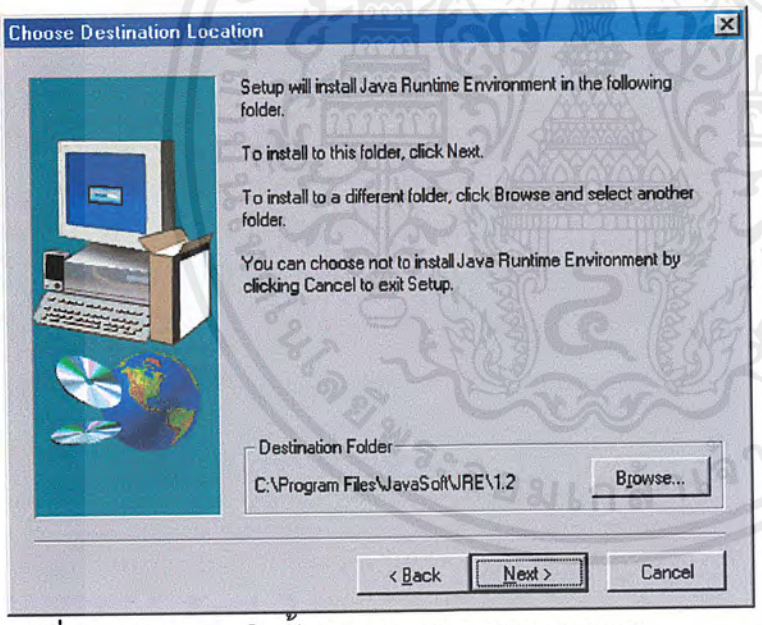


รูปที่ 4.12 แสดงส่วนประกอบของ Java Development kit

4.2.4 ทำการติดตั้ง Java Runtime Environment

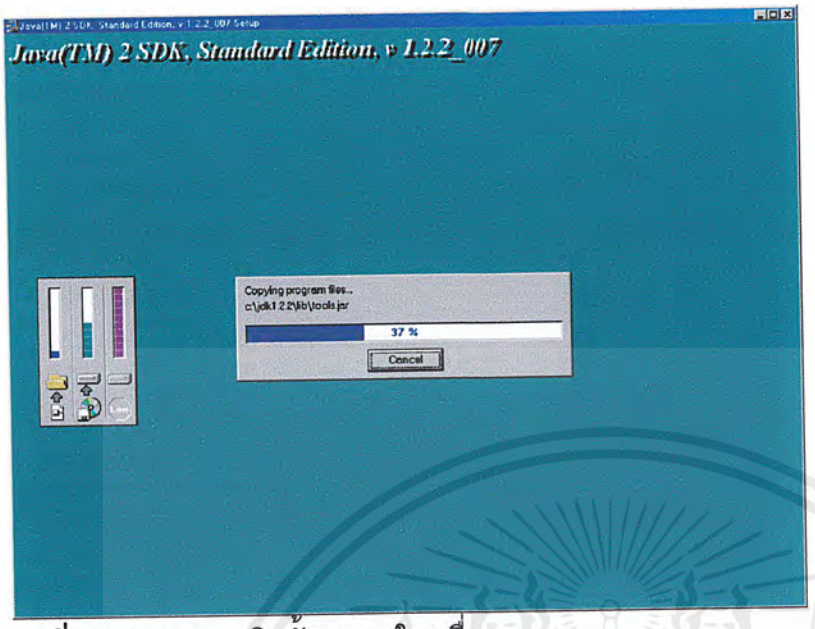


รูปที่ 4.13 แสดงการติดตั้ง Java Runtime Environment

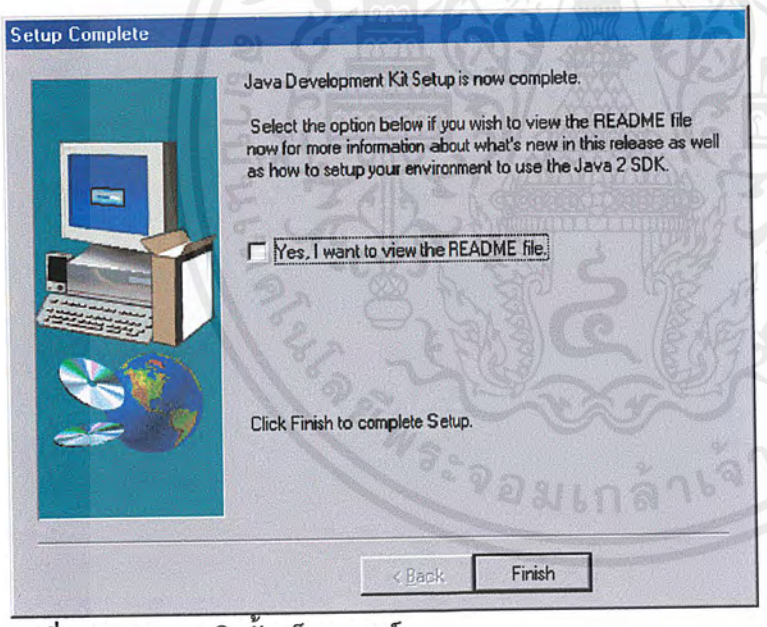


รูปที่ 4.14 แสดงการติดตั้ง Java Runtime Environment

4.2.5 ทำการติดตั้ง file ลงใน เครื่องพร้อมใช้งาน



รูปที่ 4.15 แสดงการติดตั้ง file ลงในเครื่อง



รูปที่ 4.16 แสดงการติดตั้งเสร็จสมบูรณ์

4.2.6 หลังจากติดตั้งเสร็จเรียบร้อยให้ทำการ set path ดังต่อไปนี้ลงใน autoexec.bat

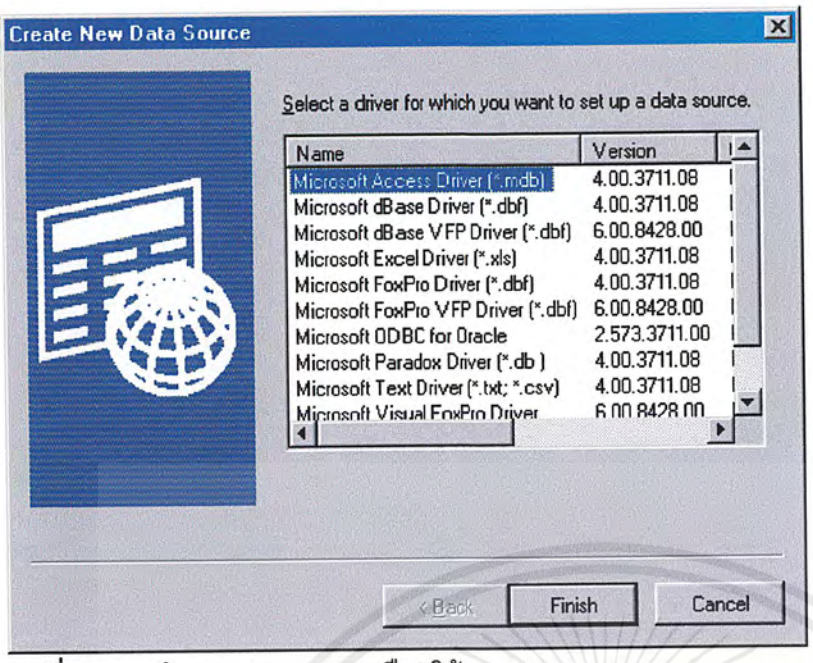
SET PATH=%PATH%;C:\JDK1.2.2\BIN;

4.3 การติดตั้งฐานข้อมูล

ขั้นตอนต่อมาทำการพัฒนาระบบฐานข้อมูลโดยทำการสร้างฐานข้อมูลจากE/Rdiagram ด้วย Microsoft SQL Server หรือ Microsoft Access แล้วจากนั้นให้ทำการ set ODBC สำหรับติดต่อกับฐานข้อมูลจากจอภาพให้คลิกที่ ODBC Data Source (32bit) เพื่อเข้าไปเพิ่ม Database ซึ่งจะขึ้นวินโดส์ให้เลือกชนิดตัวจัดการฐานข้อมูลที่ใช้ดังนี้

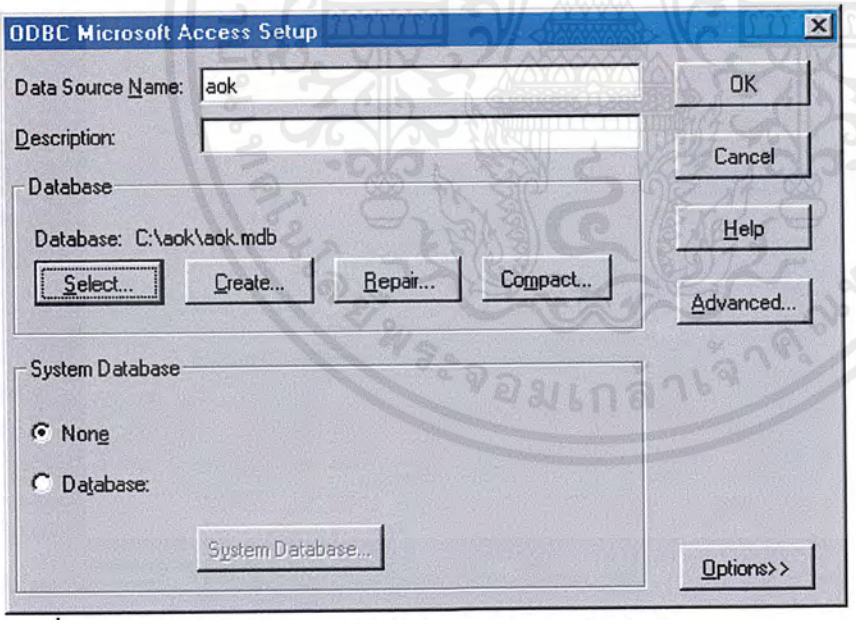
รูปที่ 4.17 หน้าจอการเข้าสู่ ODBC Data Source (32 Bit)

ต่อมาเลือก Add จะขึ้นวินโดส์ใหม่เลือกใช้ Microsoft Access Driver(*.mdb) เสร็จแล้วให้คลิกที่ปุ่ม Finish ซึ่งจะขึ้นวินโดส์ขึ้นมาใหม่ดังนี้



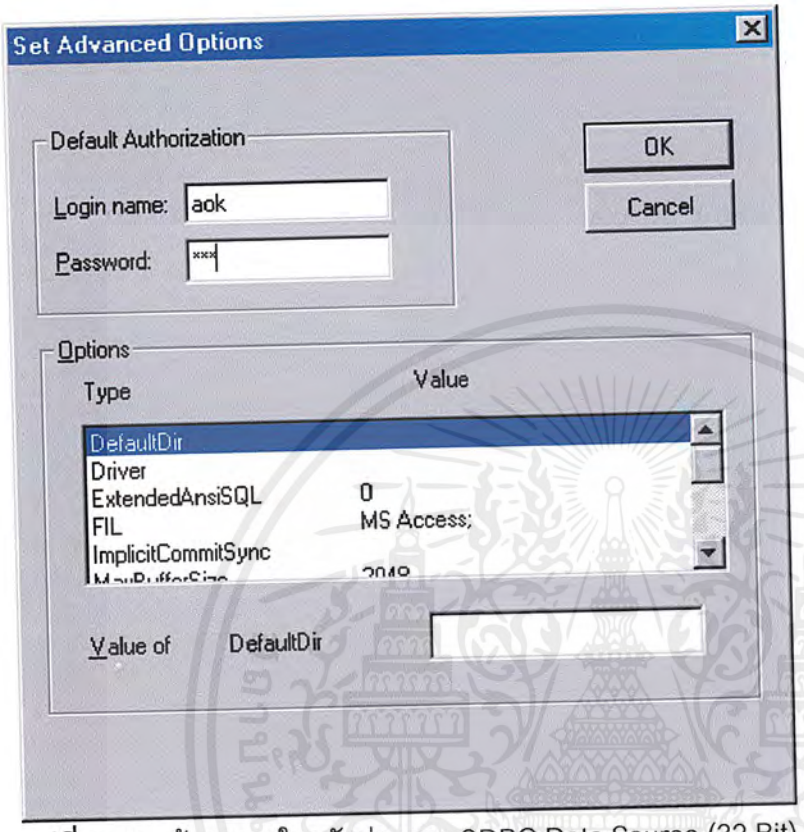
รูปที่ 4.18 หน้าจอการแสดงการเลือกใช้ Data Source

ให้ทำการพิมพ์ชื่อฐานข้อมูลที่ใช้คือ aok และกดปุ่ม select เพื่อเลือก path ที่มีไฟล์ของฐานข้อมูล ดังนี้ ในกรณีที่ลงที่เครื่องท่านจะอยู่ที่ path ดังนี้ C:\ เสร็จแล้วให้ทำการกดปุ่ม OK



รูปที่ 4.19 หน้าจอการ Setup ODBC ของ Microsoft Access

ต่อมาเลือกที่ Advanced ซึ่งจะขึ้นวินโดส์ขึ้นมาใหม่ดังนี้ ให้ทำการพิมพ์ชื่อ Username เป็น aok และ Password เป็น aok เสร็จแล้วให้ทำการกดปุ่ม OK ซึ่งเป็นการเสร็จสิ้นการใส่รหัสผ่านผู้ใช้งาน และวินโดส์จะหายไปและกดปุ่มOK ซึ่งเป็นการเสร็จสิ้นสำหรับการติดตั้งฐานข้อมูล



รูปที่ 4.20 หน้าจอการในรหัสผ่านของ ODBC Data Source (32 Bit)

โดยในการติดต่อฐานข้อมูลในระบบจะเป็นการใช้ JDBC type 1 คือทำงานผ่าน ODBC โดยจะทำการ connection กับฐานข้อมูลผ่าน Database Connection Pool เมื่อจะทำการเรียกใช้ฐานข้อมูล Object จะทำการสร้างขึ้นมาอัตโนมัติ ด้วยการเรียกใช้ package pool และทำการเรียกใช้ฐานข้อมูลผ่าน class DBUtilities ดังตัวอย่าง

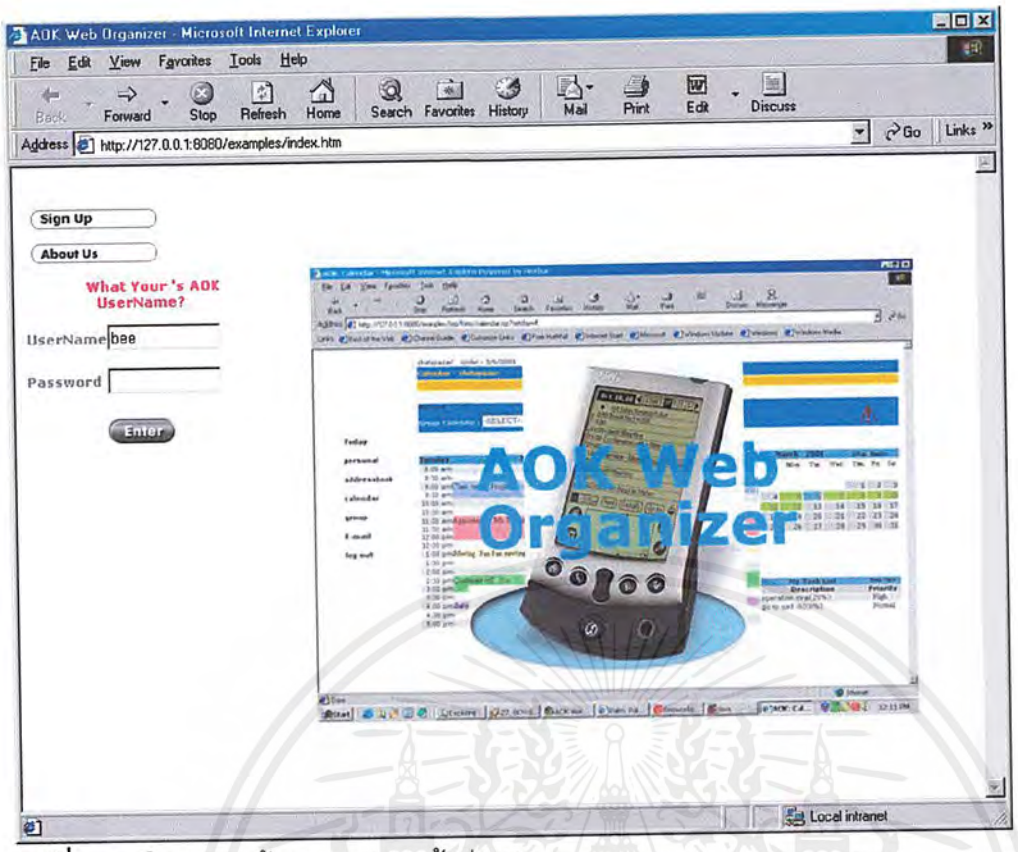
```
try{
        int result = pool.DBUtilities.accessDB("delete * from grouptask where
        id='"+id+"';");
        pool.DBUtilities.returnToPool();
        response.sendRedirect("gcalendar.jsp?ok=1");
```

```
}catch(SQLException e) {
        System.out.println("error is "+e);
}
```

โดยทำการติดต่อผ่าน pool.DBUtilities.accessDB() หรือ pool.DBUtilities.connectedDB() และเมื่อทำการใช้งานเรียบร้อยจะต้องทำการคืน ทรัพยากรให้กับระบบด้วย method pool.DBUtilities.returnToPool() โดยทุกครั้งที่ทำการใช้งานฐานข้อมูลจะต้องมีการใส่ tag คำสั่ง try{}catch{} เพื่อทำการดัก error ที่อาจจะเกิดขึ้นจากการติดต่อกับฐานข้อมูล

4.4 เริ่มต้นพัฒนาระบบงาน

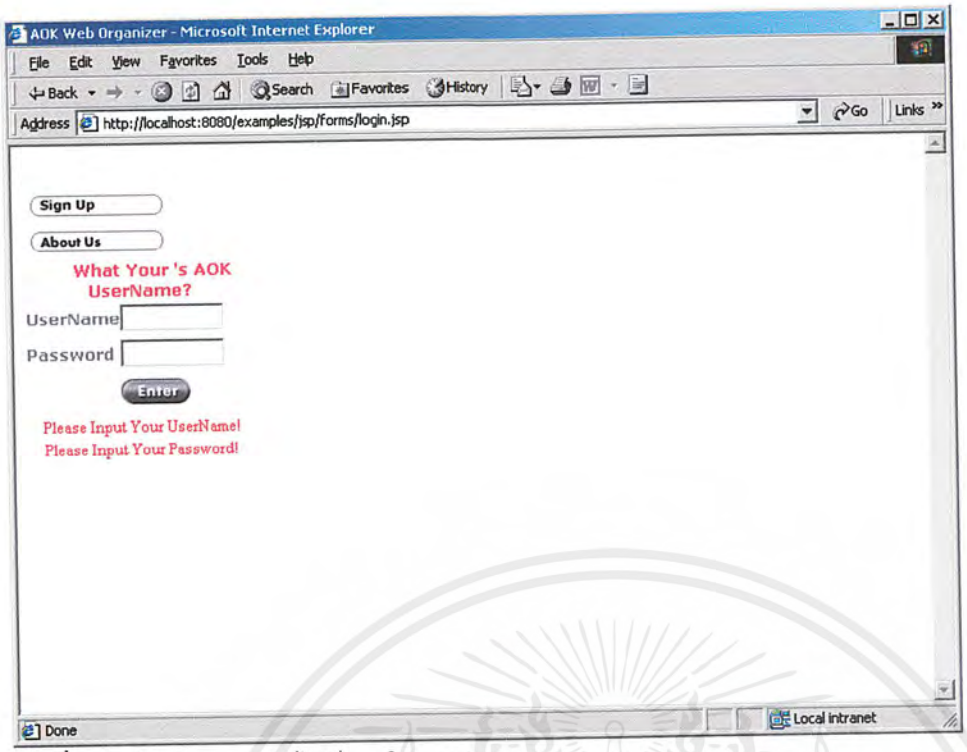
หลังจากจัดเตรียมระบบเรียบร้อยก็เข้าสู่ขั้นตอนการพัฒนา Application ในแต่ละส่วนโดยเริ่มจากโฮมเพจหน้าแรกของการเข้าสู่ระบบ Web Organizer เพื่อทำการสร้างในส่วนของการเข้าใช้ระบบโดยหน้าตาของหน้าจอในการ login เป็นดังรูป



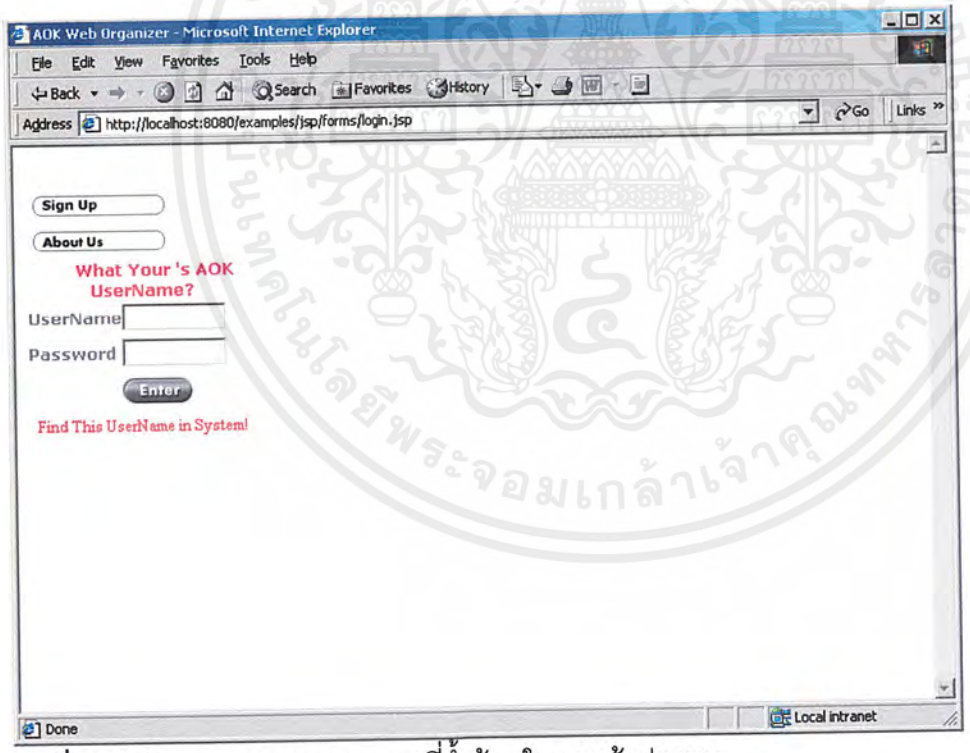
รูปที่ 4.21 โฮมเพจหน้าแรกของการเข้าสู่ระบบ

จากรูปที่ 4.21 จะเห็นว่าโฮมเพจนี้แบ่งเป็น 2 ส่วน คือ ส่วนของการลงทะเบียนใหม่เพื่อขอเข้ามาใช้ระบบ และส่วนของการ Login เพื่อเข้ามาใช้งานระบบ ซึ่งมีรายละเอียดในส่วนต่างๆดังนี้

- การลงทะเบียนใหม่เพื่อขอเข้ามาใช้ระบบ(Sing up) จะเป็นการไปยังโฮมเพจที่มีการกรอกข้อมูลที่จำเป็นเพื่อการลงทะเบียนเป็นสมาชิก
- ส่วนของการ Login จะเป็นการใส่ Username และ Password เพื่อเข้ามาใช้งานระบบ ซึ่งในการใส่ Username และ Password จะต้องไม่เป็นค่าว่าง (รูปที่ 4.22) และหากในขณะนั้นหากมีผู้ใช้งานที่เป็นชื่อ Username นี้อยู่ในระบบจะมีโฮมเพจแจ้งว่าผู้ใช้งานกำลังใช้งานอยู่ในระบบ (รูปที่ 4.23)



รูปที่ 4.22 โฮมเพจแสดงไม่มีการใส่ UseName และ Password



รูปที่ 4.23 โฮมเพจแสดงการ Login ที่ซ้ำซ้อนในการเข้าสู่ระบบ

โดยลักษณะการทำงานใน index.htm จะทำการเรียก login.jsp ซึ่งจะทำการตรวจสอบการ login จากบรรทัดคำสั่งดังต่อไปนี้

```
<jsp:useBean id="user" class="foo.UserBean" scope="session"/>
<% user.setUsername(request.getParameter("Username")); System.out.println("Usernaem
is "+user.getUsername());%>
<% user.setPassword(request.getParameter("Password"));
System.out.println("Password is "+user.getPassword());%>
<% if (user.checkUser()) {System.out.println("main");%>
<jsp:forward page="main.jsp"/>
<% } else {System.out.println("error");
%>
<jsp:forward page="errorlogin.jsp"/>
<%
}
%>
```

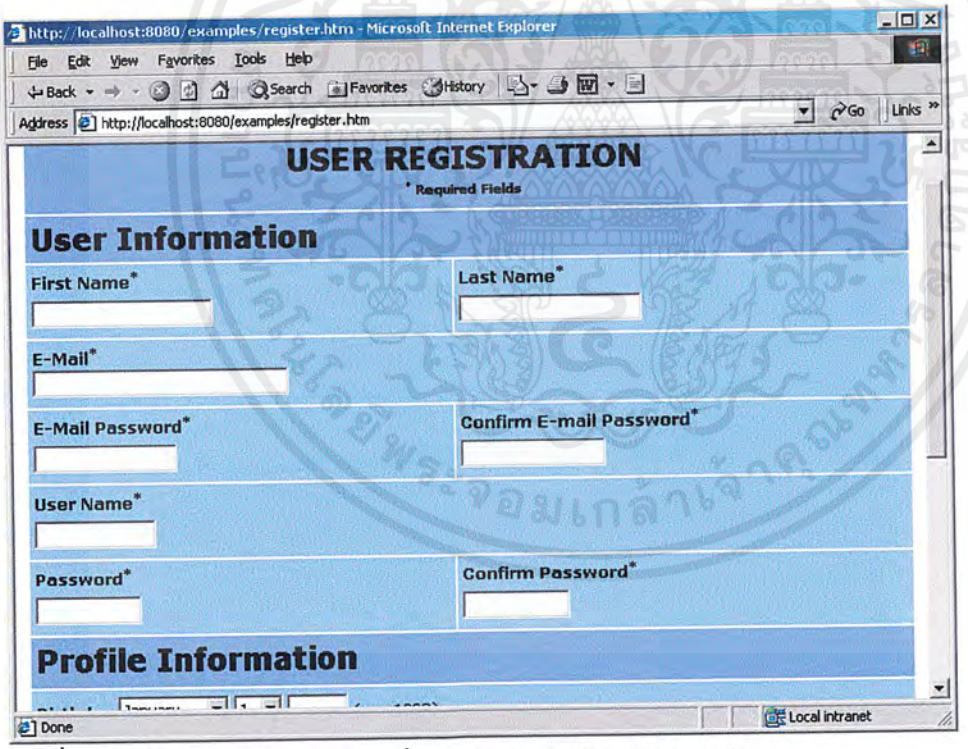
โดยใน 5 บรรทัดแรกจะเป็นการสร้าง JavaBean ของ ผู้ใช้ขึ้นมาต่อจากนั้นจะทำการอ่านค่าของ username และ password และทำการ check ด้วย method checkUser() ถ้าถูกต้องจะทำการ forward page ไปที่ main.jsp หากไม่ถูกต้อง จะไปยัง errorlogin.jsp และทำการแสดงค่า error ดังภาพข้างต้น

4.5 โฮมเพจหน้าของการลงทะเบียนเพื่อเป็นสมาชิกระบบ Web Organizer

จากการเลือกการลงทะเบียนใหม่เพื่อขอเข้ามาใช้ระบบ(Sing up) จะเป็นการเข้ามาสู่โฮมเพจเพื่อกรอกข้อมูลที่สำคัญสำหรับใช้ในการลงทะเบียน ซึ่งการกรอกข้อมูลนั้นหากในช่องมีเครื่องหมายดอกจันทร์คือ จำเป็นต้องมีข้อมูล ซึ่งการกรอกข้อมูลจะแบ่งเป็น 2 ส่วน มีดังนี้

ส่วนของ User Information (รูปที่ 4.24)

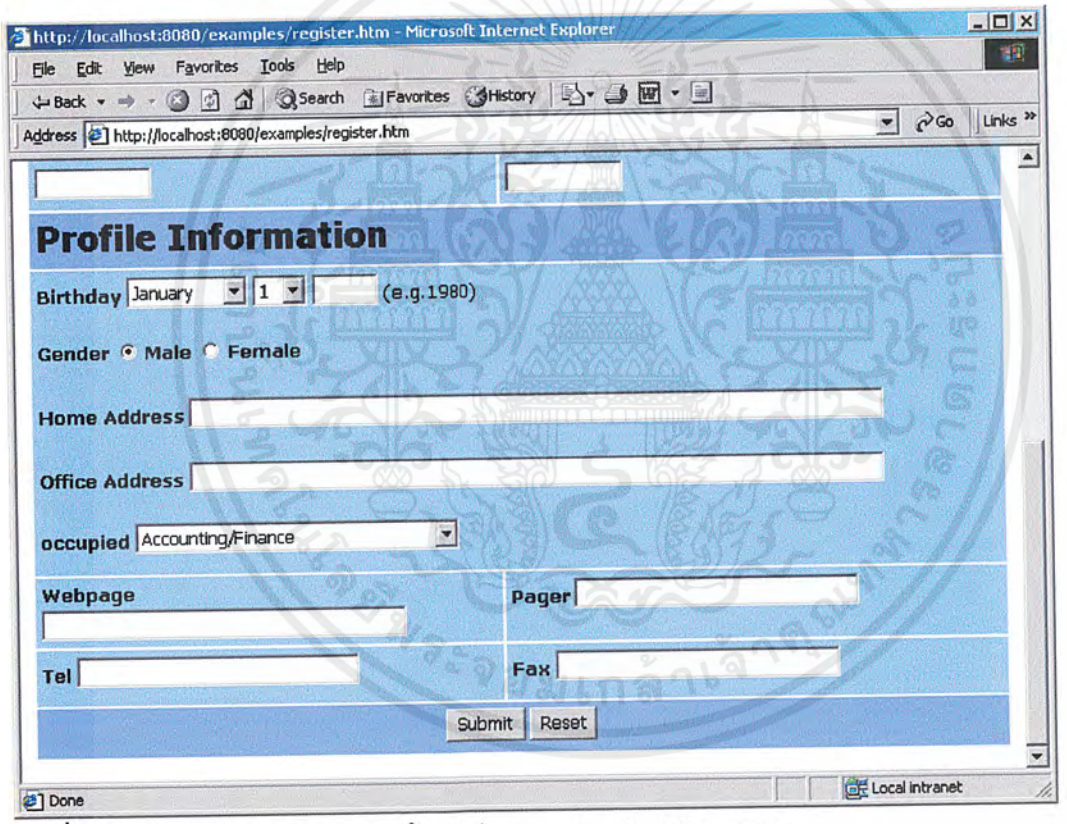
- First Name ชื่อจริงของผู้ใช้ระบบ
- Last Name นามสกุลจริงของผู้ใช้ระบบ
- E-Mail e-mail address ของผู้ใช้ระบบ
- E-Mail Password e-mail password ของผู้ใช้ระบบสำหรับเข้าไปอ่านเมล์
- User Name เป็นชื่อที่ใช้ในการเข้าใช้ระบบ
- Password รหัสที่ใช้ในการเข้าใช้ระบบ
- Confirm Password ยืนยันรหัสที่ใช้ในการเข้าใช้ระบบ ต้องเหมือนกับ Password



รูปที่ 4.24 โฮมเพจแสดงการกรอกข้อมูลส่วนของ User Information

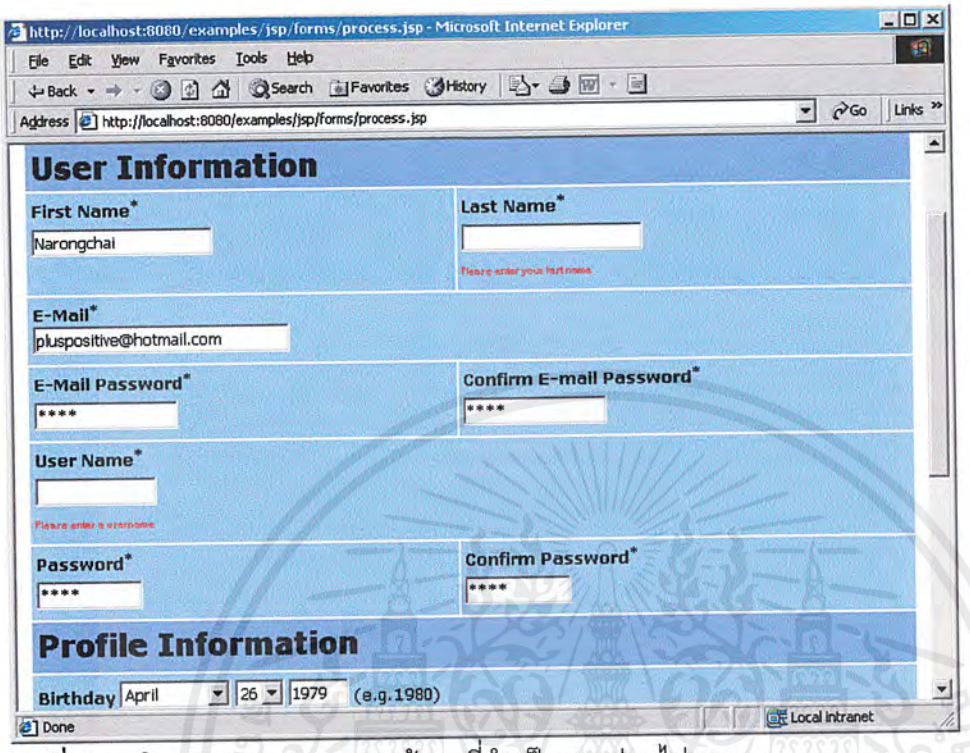
ส่วนของ Profile Information (รูปที่ 4.25) ซึ่งจะเป็นข้อมูลทั่วไปของสมาชิก

- Birthday วันเกิดของผู้ใช้ระบบ แบบ เดือน วัน ปี
- Gender เพศของผู้ใช้ระบบ
- Home Address ที่อยู่ของผู้ใช้ระบบ
- Office Address ที่ทำงานของผู้ใช้ระบบ
- Occupied อาชีพของผู้ใช้ระบบ
- Webpage IP address หรือ URLs ของผู้ใช้ระบบ
- Pager เบอร์ pager ของผู้ใช้ระบบ
- Tel เบอร์โทรศัพท์ของผู้ใช้ระบบ
- Fax เบอร์โทรสารของผู้ใช้ระบบ



รูปที่ 4.25 โฮมเพจแสดงการกรอกข้อมูลส่วนของ Profile Information

ในการกรอกข้อมูลถ้ากรอกข้อมูลที่จำเป็นไม่ครบระบบมีการแจ้งกลับมาเพื่อให้กรอกข้อมูลที่จำเป็นให้ครบจนหมดก่อนจึงจะสามารถสมัครเป็นสมาชิกได้ (รูปที่ 4.26)



รูปที่ 4.26 โฮมเพจแสดงการกรอกข้อมูลที่จำเป็นบางส่วนไม่ครบ

ในการกรอก User Name ถ้ากรอก User Name ที่มีชื่อใช้งานอยู่ในระบบเรียบร้อยแล้วระบบมีการแจ้งกลับมาเพื่อให้กรอก User Name ใหม่ที่ยังไม่มีชื่อการใช้งานอยู่ในระบบจึงจะสามารถสมัครเป็นสมาชิกได้ (รูปที่ 4.27) ซึ่งรูปแบบการตรวจสอบ ในpage register.htm จะทำการส่งต่อไปให้ process.jsp ทำการตรวจสอบและติดต่อกับฐานข้อมูลดังต่อไปนี้

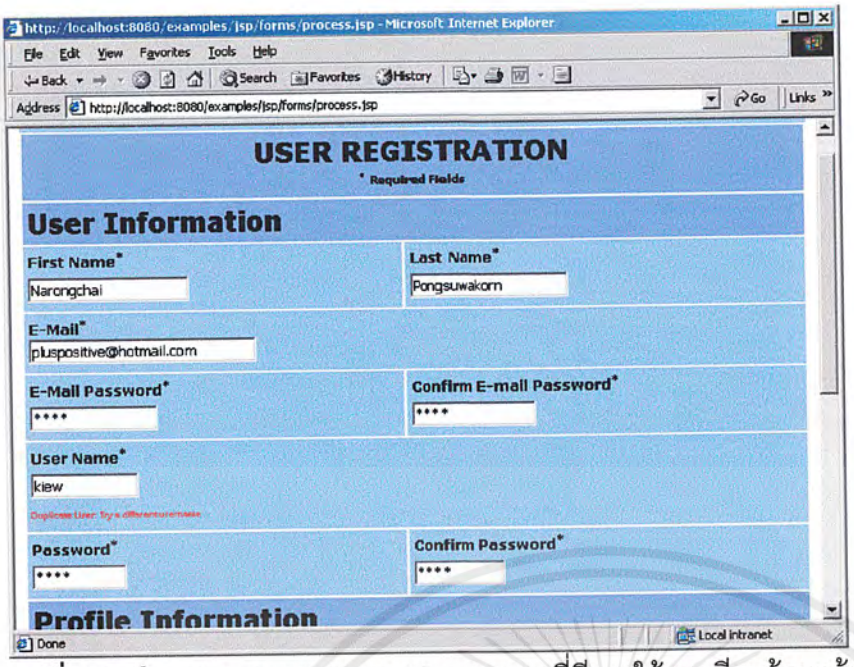
```
<jsp:useBean id="formHandler" class="foo.FormBean" scope="request">
<jsp:setProperty name="formHandler" property="*"/>
<jsp:setProperty name="formHandler" property="b_Month" param="B_Month"/>
<jsp:setProperty name="formHandler" property="b_Date" param="B_Date"/>
<jsp:setProperty name="formHandler" property="b_Year" param="B_Year"/>
<jsp:setProperty name="formHandler" property="h_Address" param="H_Address"/>
```

```
<jsp:setProperty name="formHandler" property="o_Address" param="O_Address"/>
<jsp:setProperty name="formHandler" property="occupied" param="Occupied"/>
<jsp:setProperty name="formHandler" property="fax_no" param="Fax_no"/>
<jsp:setProperty name="formHandler" property="gender" param="Gender"/>
<jsp:setProperty name="formHandler" property="pager_no" param="Pager_no"/>
<jsp:setProperty name="formHandler" property="web_Page" param="Web_Page"/>
<jsp:setProperty name="formHandler" property="tel_no" param="Tel_no"/>
</jsp:useBean>
```

ส่วนแรกนี้จะเป็นส่วนในการรับข้อมูลจากการกรอกข้องสมาชิกที่ทำการลงทะเบียน

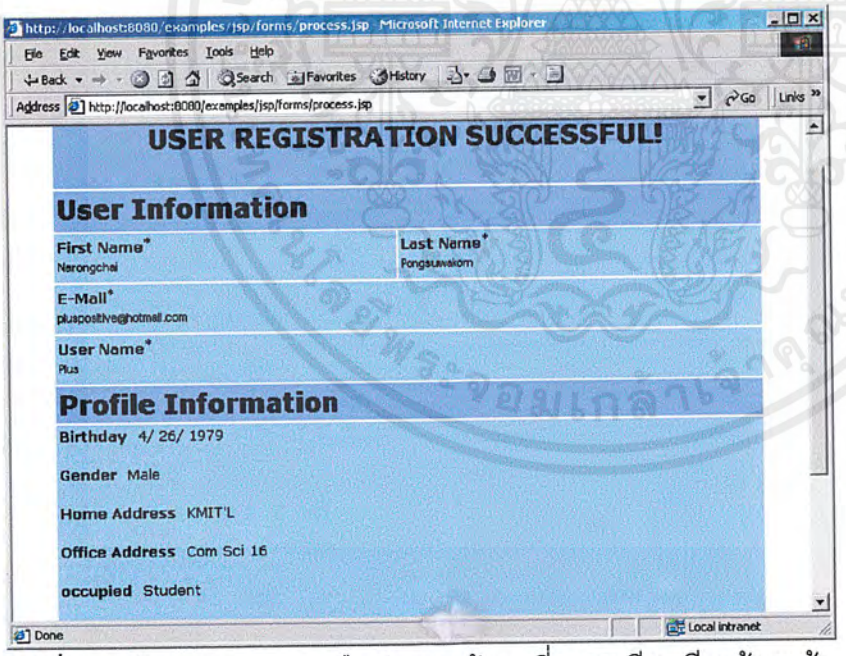
```
<% if (formHandler.validate()) {
%>
<jsp:forward page="/servlet/DBHandler"/>
<%
  } else {
%>
      <jsp:forward page="<%=bundle.getString(\"process.retry\")%>"/>
<%
  }
%>
```

ต่อจากการรับข้อมูลจะเป็นการตรวจสอบข้อมูล ทั้งในเรื่องของความถูกต้อง และความซ้ำซ้อนของข้อมูลหาก ถูกต้องก็จะทำการจัดเก็บลงฐานข้อมูล หากไม่ถูกต้องก็จะทำการส่งผลไปให้ผู้ทำการสมัคร และแสดงข้อผิดพลาด



รูปที่ 4.27 โฮมเพจแสดงการกรอก User Name ที่มีการใช้งานเรียบร้อยแล้วอยู่ในระบบ

เมื่อสามารถสมัครเป็นสมาชิกได้เรียบร้อยแล้วจะมีหน้าจอยืนยันรายละเอียดของการสมัครเป็นสมาชิก และเลือกการแสดงผลไปยังโฮมเพจที่เป็นหน้าของการเข้าสู่ระบบ (รูปที่ 4.28)



รูปที่ 4.28 โฮมเพจแสดงการยืนยันกรอกข้อมูลที่ลงทะเบียนเรียบร้อยแล้ว

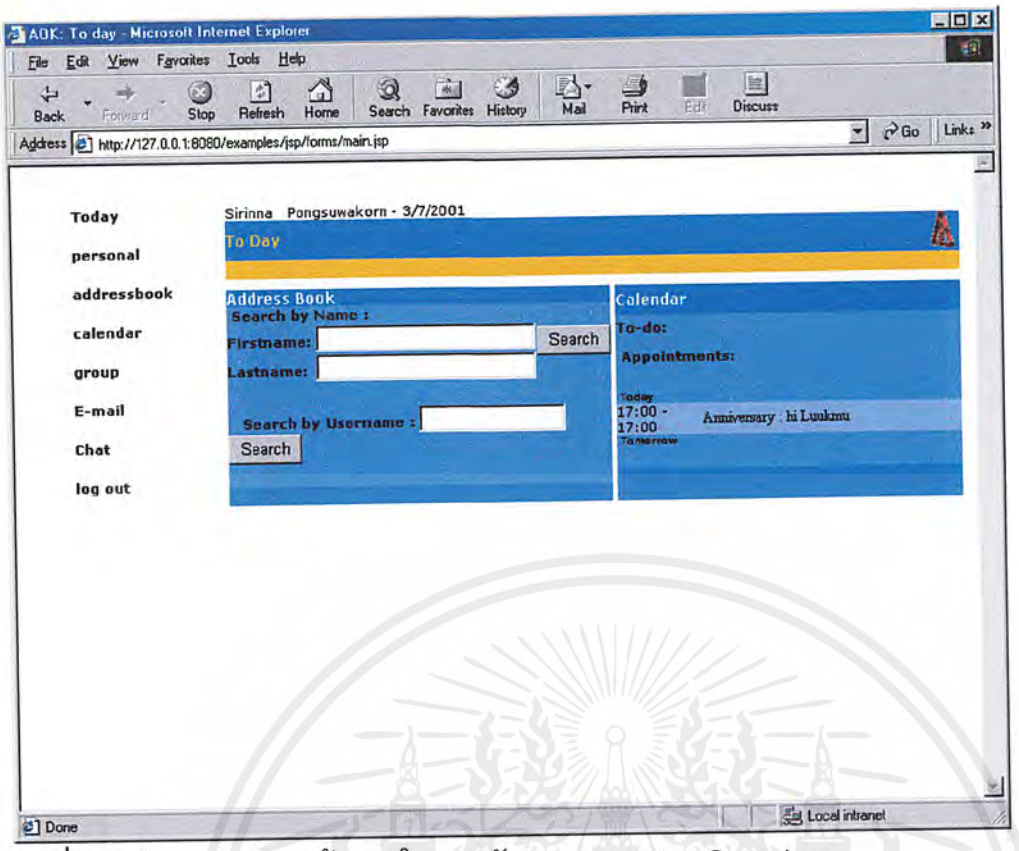
4.6 โฮมเพจหน้าการทำงานหลักของระบบ Web Organizer

เมื่อผู้สมาชิกของระบบผ่านการ Login ที่เรียบร้อย จะมาที่โฮมเพจหน้าหลักของระบบ Web Organizer ที่จะประกอบรูปแบบการทำงาน 2 ส่วน และเมนูการทำงาน 7 เมนู ซึ่งรูปแบบการทำงาน 2 ส่วน ประกอบไปด้วย (รูปที่ 4.29)

- การทำงานร่วมกับ Address Book ที่สามารถค้นหาจากชื่อและนามสกุลผู้ที่ต้องการค้นหา หรือ ค้นหาจาก Username
- การทำงานร่วมกับ Calendar ที่จะแสดงการนัดหมายที่เกิดขึ้นในวันที่ทำการใช้งานระบบ และสามารถที่จะเรียกเข้าไปดูการนัดหมายได้

เมนูการทำงาน 7 เมนู ประกอบไปด้วย

- Today เป็นเมนูที่ใช้แสดงหน้าจอหลักโฮมเพจหน้าหลักของระบบ Web Organizer
- Personal เป็นเมนูที่จะใช้ในการเปลี่ยนแปลง แก้ไขข้อมูลของสมาชิกที่ได้กรอกไว้ในการสมัครเป็นสมาชิกของระบบ
- Addressbook เป็นเมนูที่จะใช้ในการสร้าง เปลี่ยนแปลง แก้ไขข้อมูล ของบุคคลที่เราต้องการจะเก็บ
- Calendar เป็นเมนูที่ใช้แสดงปฏิทินนัดหมายส่วนบุคคล
- Group เป็นเมนูที่ใช้ในการสร้างและจัดการกลุ่มผู้ใช้งาน
- E-mail เป็นเมนูที่ใช้ในการอ่านและรับ E-mail
- Chat เป็นเมนูที่ใช้ในการสนทนาของผู้ใช้ระบบ
- Logout เป็นเมนูที่ใช้ในออกจากระบบ



รูปที่ 4.29 โฮมเพจแสดงหน้าการทำงานหลักของระบบ Web Organizer

เมื่อเข้ามาสู่หน้าจอหลักซึ่งจะเป็นการแสดงข้อมูลปัจจุบันจะมีการแสดง ชื่อของสมาชิก วันที่ หากมีหมายนัดจะมีการแสดงทางด้านส่วนของ Calendar ไม่ว่าจะเป็น task, Appointments ของวันนี้และวันพรุ่งนี้ นอกจากนั้นในทุกหน้าจะมีการแสดง เครื่องหมาย "!" ซึ่งจะแสดงว่ามีการเตือนว่ามีนัดหมายเกิดขึ้น นอกจากนี้ยังมีส่วนของการ search หาบุคคลจากชื่อ หรือ username ภายใน address book ของสมาชิกได้อีกด้วย โดยในการทำงานในทุก ๆ page จะมีการ สร้าง JavaBean ของสมาชิกในระบบและ ระบบจัดการเวลาขึ้นมา ดังตัวอย่างด้านล่าง

```
<jsp:useBean id="user" class="foo.UserBean" scope="session"/>
<jsp:useBean id="acal" class="cal.JspCalendar" scope="session"/>
```

ซึ่งทั้ง 2 Bean นี้จะทำหน้าที่ในการจัดการข้อมูลและติดต่อกับข้อมูลของสมาชิกในระหว่างอยู่ภายใน Web Application ตัวนี้ตัวอย่างเช่นถ้าต้องการชื่อของ สมาชิกก็ทำการเรียก method getUsername(), getFirstname(), getLastname() ดังตัวอย่าง

```
<tr>
    <td colspan="2"> <font color="#000000" face="Verdana, Arial, Helvetica, sans-serif"
    size="2">
  <%=user.getFirstName()%>   <%=user.getLastName()%> -
   <%=acal.getCurrentDate()%>
  </font> </td>
</tr>
```

จากตัวอย่างจะมีการอ่านค่า Firstname และ Lastname จาก Bean user และมีการอ่านค่าเวลาปัจจุบันจาก Bean acal ซึ่งทั้งสอง Bean นี้สามารถจะอ้างอิงได้ตลอดทั้ง application ที่มีการประกาศไว้

4.7 โฮมเพจหน้าการทำงานของ Persanol

เมื่อผู้ที่ได้ลงทะเบียนเป็นสมาชิกของระบบ Web Organizer มีความประสงค์ที่จะแก้ไขข้อมูลของตนเองสามารถที่จะเลือกทำการแก้ไขได้จากเมนู Personal ซึ่งจากการได้เลือกการทำงานที่เมนูนี้จะแสดงรายละเอียดทั้งหมดของผู้ที่ได้ลงทะเบียนเป็นสมาชิกของระบบ Web Organizerให้สามารถทำการแก้ไขได้ (รูปที่ 4.30) โดยใน page นี้จะมีการสร้าง Bean นอกเหนือจาก 2 Bean ที่กล่าวมาก็คือ Bean info โดยสร้างจาก class foo.FormBean

```
<jsp:useBean id="info" class="foo.FormBean" scope="request"/>
```

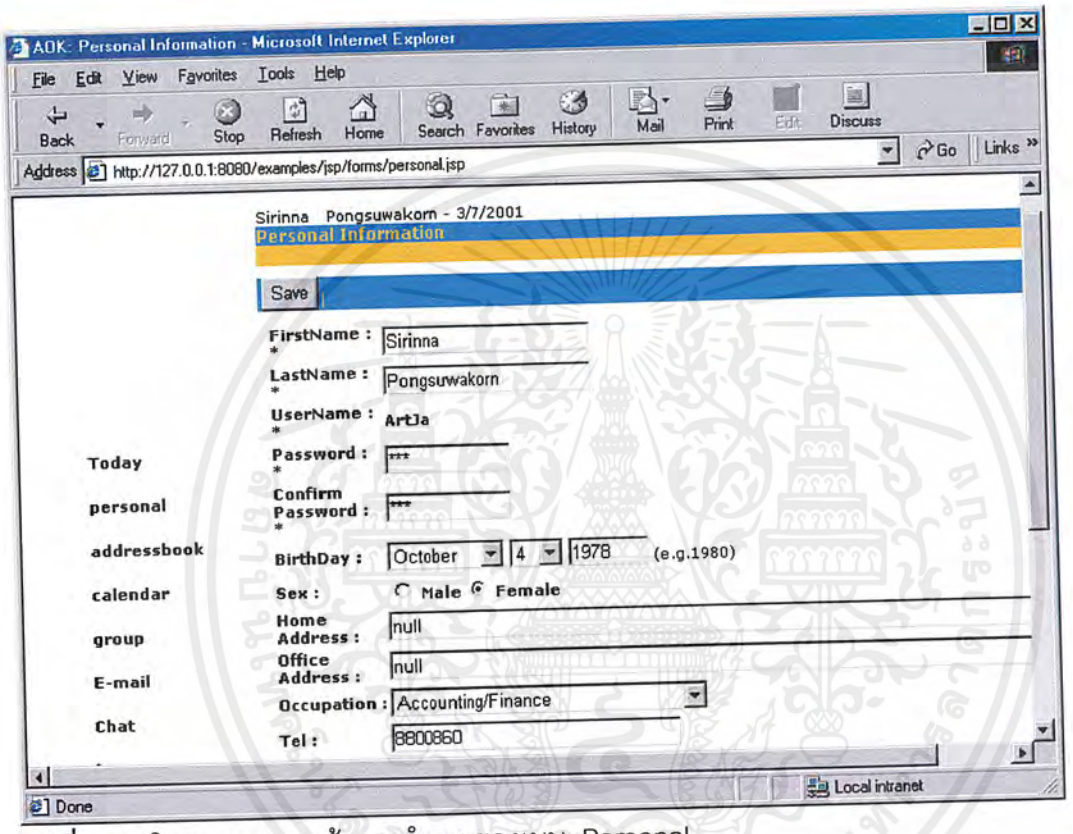
โดย class นี้จะทำการติดต่อกับฐานข้อมูลแล้วทำการอ่านค่ามาแสดงผล นอกจากนี้ยังสามารถทำการ update ข้อมูล ผ่าน method จาก info bean นี้ได้อีกด้วย ดังตัวอย่าง

```
<%=info.getErrorMsg("firstName")%>
<%=info.getErrorMsg("lastName")%>
<%=info.getO_Address()%>
```

ส่วนตัวอย่างด้านล่างจะเป็นการ รับค่าจาก page ที่ทำการแสดง information ของ สมาชิกแล้วทำการ update ผ่าน info.save() เมื่อทำการกดปุ่ม save

```
<%
info.setAllattribute(request);
if (info.validate()) {
```

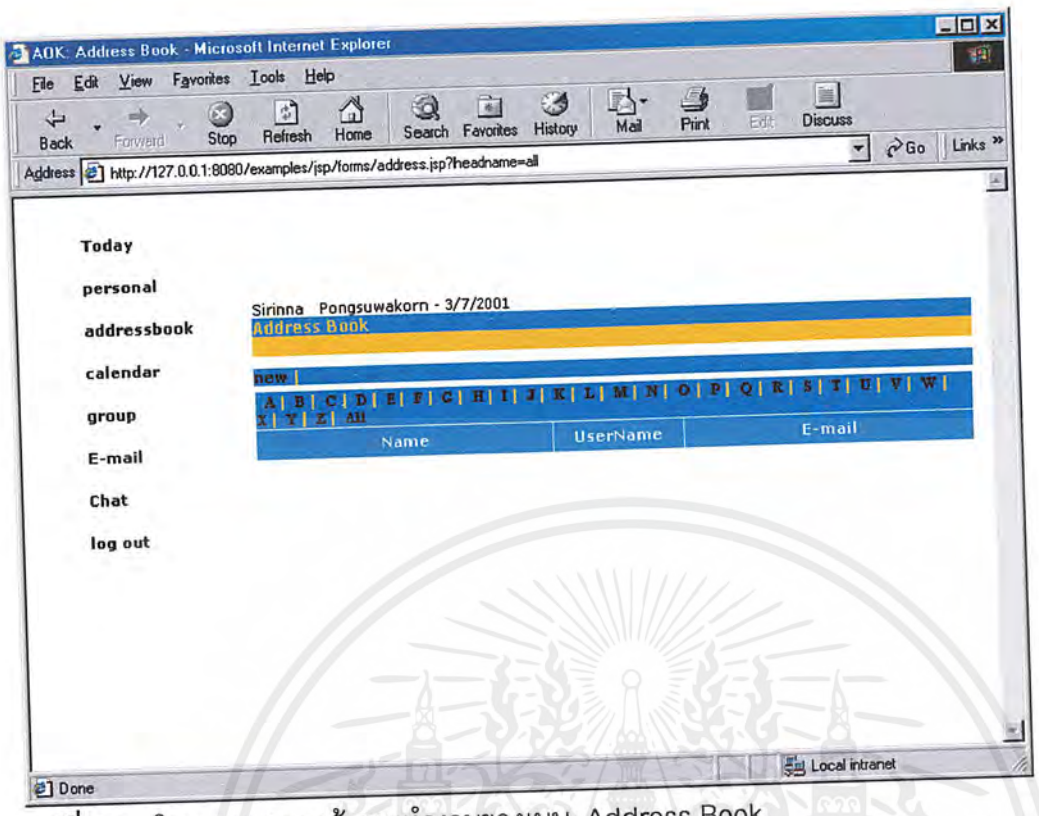
```
info.save(user.getUsername());
%>
        <jsp:forward page="personal.jsp"/><%
} else {%>
        <jsp:forward page="personal.jsp"/><%
}%>
```



รูปที่ 4.30 โฮมเพจแสดงหน้าการทำงานของเมนู Persanal

4.8 โฮมเพจหน้าการทำงานของ Address Book

จากการเลือกเมนู Address Book ซึ่งเมนูนี้จะแสดงรายละเอียดทั้งหมดของรายชื่อ และ อีเมล์ของผู้ที่ได้ทำการบันทึกลงใน Address Book (รูปที่ 4.31) โดยเจ้าของ Address Book สามารถที่จะเรียกดูรายชื่อใน Address Book ได้ตามตัวอักษรภาษาอังกฤษ และสามารถแก้ไขและเรียกดูข้อมูลทั้งหมดของผู้ที่ถูกบันทึกลงใน Address Book (รูปที่ 4.32)



รูปที่ 4.31 โฮมเพจแสดงหน้าการทำงานของเมนู Address Book

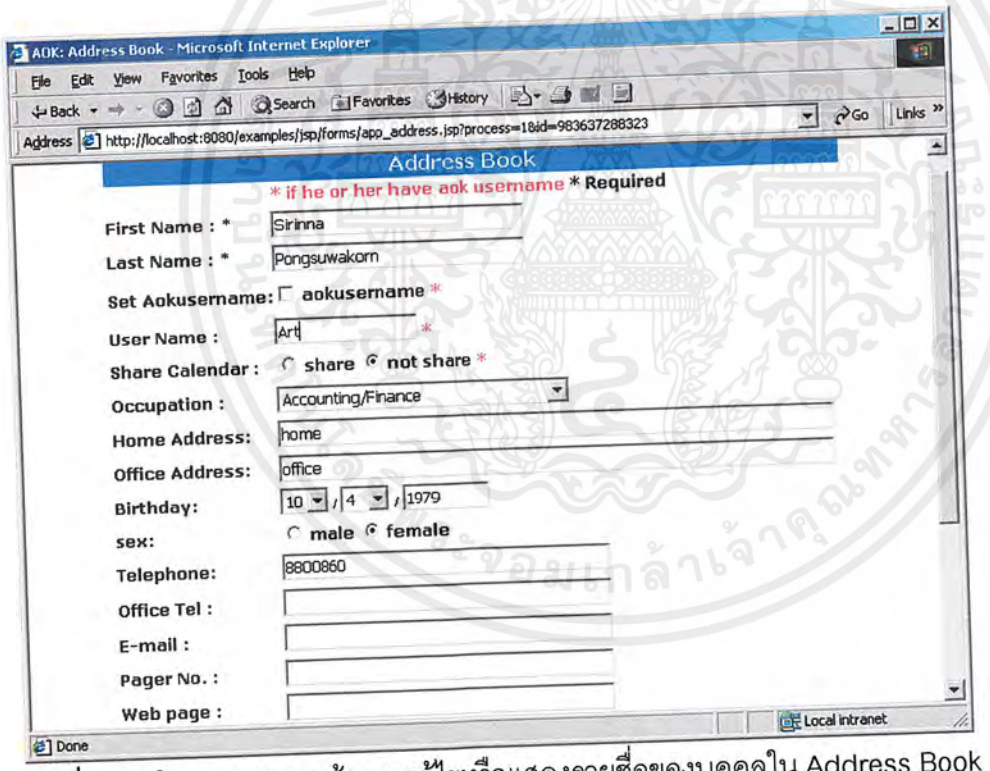
จากรูปจะเห็นได้ว่าจะมีการเรียงข้อมูลใน address book ตามชื่อโดยที่เมื่อทำเกิดเปิด page ครั้งแรกจะทำการแสดง รายชื่อทั้งหมด หากต้องการเฉพาะตัวอักษรก็ทำการเลือกตามตัวอักษร ตัวอย่างเช่น

```
<a class="noline" href="address.jsp?headname=a">A</a>
<a class="noline" href="address.jsp?headname=b">B</a>
<a class="noline" href="address.jsp?headname=c">C</a>
<a class="noline" href="address.jsp?headname=d">D</a>
```

โดยจะมีการส่งตัวอักษรไปให้ address.jsp ผ่านพารามิเตอร์ headname ซึ่งจะทำการแสดงผลเฉพาะตัวอักษรที่ต้องการโดยการ query จากฐานข้อมูลจากตัวอย่าง

```
<%
    String headname = request.getParameter("headname");
    String sql = "";
    if(headname!=null)
    {
```

```
            if(headname.equals("all"))
            {
                  sql = "select * from addressbook where username =
'"+user.getUsername()+"' order by firstname,lastname;";
            }else
            {
                  sql = headname;
            }
      }
%>
```



รูปที่ 4.32 โฮมเพจแสดงหน้าการแก้ไขหรือแสดงรายชื่อของบุคคลใน Address Book

จากรูปที่ 4.32 จะเป็นการจัดการเพิ่ม แก้ไข หรือ ลบ address book ออกจากฐานข้อมูลโดยการเรียก page app_address.jsp ขึ้นมาทำงานและทำการติดต่อกับฐานข้อมูล โดยในการตรวจสอบการทำงานจะมีการส่งพารามิเตอร์ process เพื่อดูว่าจะทำอะไร

```
String process = request.getParameter("process");
```

โดยถ้า เป็น 0 จะเป็นการ new ส่วนถ้าเป็น 1 จะเป็นการแก้ไขหรือเปิดดูหรือลบ โดยจะมีการส่งพารามิเตอร์อีกตัวหนึ่งคือ id สำหรับเป็น key ในการค้นหาในฐานข้อมูล

```
String id = request.getParameter("id");
```

นอกจากนี้อาจจะมีการจัดการกับความผิดพลาดหรือที่เกิดจากผู้ใช้ด้วยการส่งค่า errorมาให้กับ page นี้ด้วย

```
String error1 = request.getParameter("error1");
String error2 = request.getParameter("error2");
```

4.9 โฮมเพจหน้าการทำงานของ calendar

ในการที่ผู้ใช้งานระบบต้องการที่จะสร้างตารางนัดหมายนั้น สามารถที่จะสร้างโดยผ่านจากการเลือกทางเมนู Calendar เมนู ซึ่งเมนูนี้จะแสดงรายละเอียด โดยมีการแบ่งช่วงเวลานัดหมายจาก 8.00 – 17.00 ซึ่งแบ่งเป็นช่วงละ 1 ชั่วโมง และมีปฏิทินเพื่อที่ใช้เลือกวันเวลาล่วงหน้าเพื่อบันทึกการนัดหมาย ปฏิทินจะมีวันที่ปรากฏอยู่ถ้าวันที่ปรากฏมีกรอบเขียวล้อมรอบหมายความว่าวันนั้นมีการนัดหมายหรืองานที่ทำในช่วงเวลานั้น

โดยการทำงานหลักของ Calendar จะมีการใช้งาน Bean cal.JspCalendar ซึ่งเป็นคลาสที่ทำงานกับระบบปฏิทินและระบบการนัดหมายโดยมีการเรียกใช้ดังนี้

```
<jsp:useBean id="acal" class="cal.JspCalendar" scope="session"/>
```

โดยจะมี method ที่สำคัญ ๆ สำหรับทำงานกับปฏิทินและระบบนัดหมายดังตัวอย่าง

```
<%=acal.getCalendarD(user.getUsername())%>
```

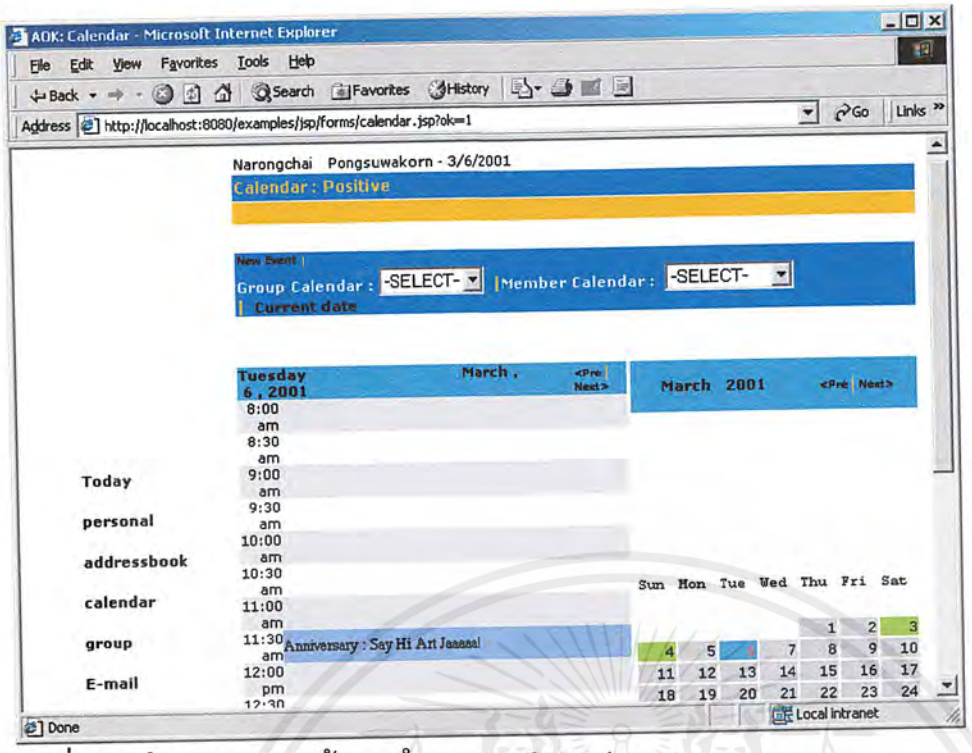
เป็นการเรียก method getCalendarD() สำหรับทำการแสดงรายการ event ของแต่ละวัน

```
<%=acal.getCalendarM(user.getUsername())%>
```

เป็นการเรียก method getCalendarM() สำหรับทำการแสดงรายการ ปฏิทิน ของเดือนปัจจุบันและตามที่ผู้ใช้จะทำการใช้งานพร้อมทั้งแสดงสีเพื่อแสดงว่ามีการนัดหมายหรือไม่ด้วย

```
<%=acal.getTaskList(user.getUsername())%>
```

เป็นการเรียก method getTaskList() สำหรับทำการแสดงรายการ Task ของผู้ใช้ในแต่ละวัน



รูปที่ 4.33 โฮมเพจแสดงหน้าการทำงานของ Calendar

ในการเพิ่ม แก้ไข ลบ การนัดหมายสามรถทำได้โดยเลือกไปยังช่วงเวลาที่นัดหมาย และจะมีโฮมเพจที่ใช้สำหรับเพิ่ม แก้ไข ลบการนัดหมายปรากฏออกมา (รูปที่ 4.34) ในหน้าจอการใส่ตารางนัดหมายจะประกอบไปด้วย

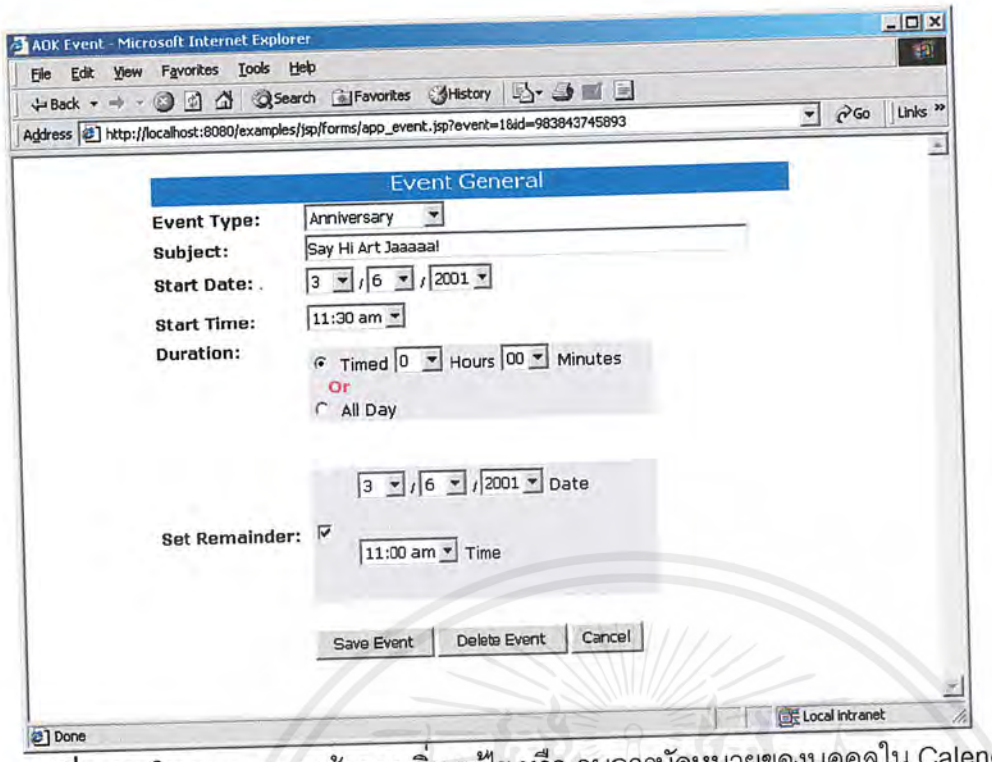
- Event Type จะเป็นการกำหนดรูปแบบการนัดหมาย
- Subject จะเป็นการใส่ชื่อหัวข้อการนัดหมาย
- Start Date จะเป็นการกำหนดวันที่ทำการนัดหมาย
- Start Time จะเป็นการกำหนดเวลาที่ทำการนัดหมาย
- Duration จะเป็นการกำหนดความคงสภาพของนัดหมาย
- Set Remainder จะเป็นการกำหนดการเตือนการนัดหมาย

โดยในการจัดเก็บการนัดหมายจะมีการรับค่าพารามิเตอร์ event จากผู้ใช้ว่าต้องการทำ action อะไรเช่นถ้าต้องการเพิ่ม event ใหม่ก็จะส่งค่า 0 มาให้โดยทำการรับค่าดังนี้

```
String event = request.getParameter("event");
```

แล้วทำการจัดเก็บ หากเป็นการเปิดดูหรือแก้ไขก็จะมีการใช้พารามิเตอร์ id ในการอ้างอิงกับตัว event

```
String id = request.getParameter("id");
```



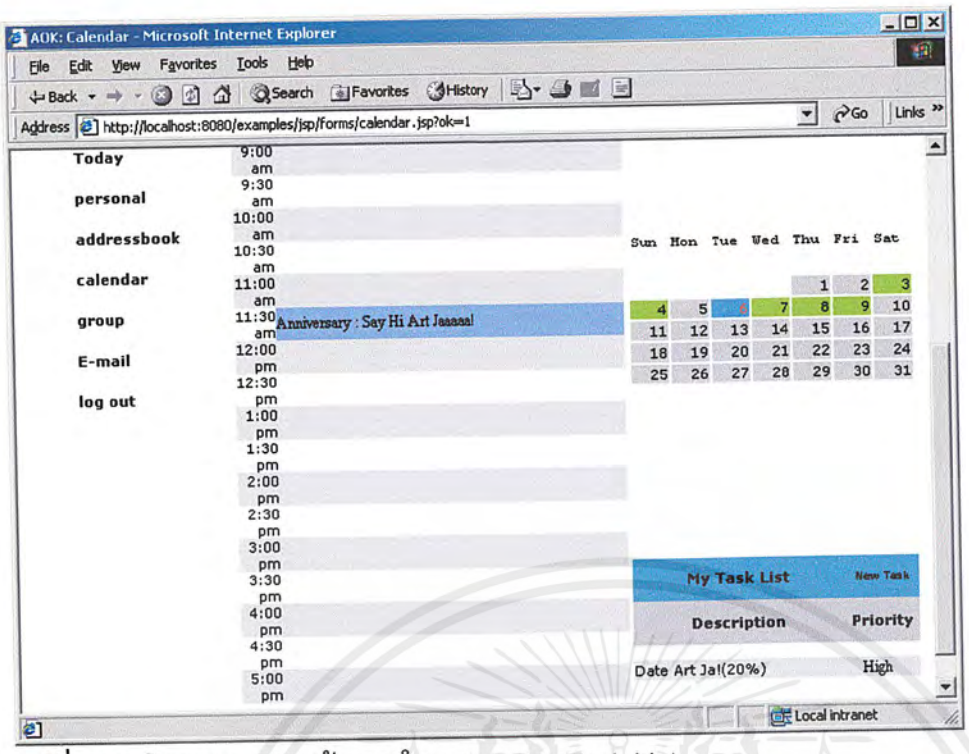
รูปที่ 4.34 โฮมเพจแสดงหน้าการเพิ่ม แก้ไข หรือ ลบการนัดหมายของบุคคลใน Calendar

การเพิ่ม แก้ไข ลบ บันทึกเตือนการทำงานสามารถทำได้โดยเลือกไปยัง My Task List (รูปที่ 4.35) และจะมีโฮมเพจที่ใช้สำหรับเพิ่ม แก้ไข ลบการบันทึกเตือนทำงานปรากฏออกมา (รูปที่ 4.36) ในหน้าจอการใส่ตารางนัดหมายจะประกอบไปด้วย

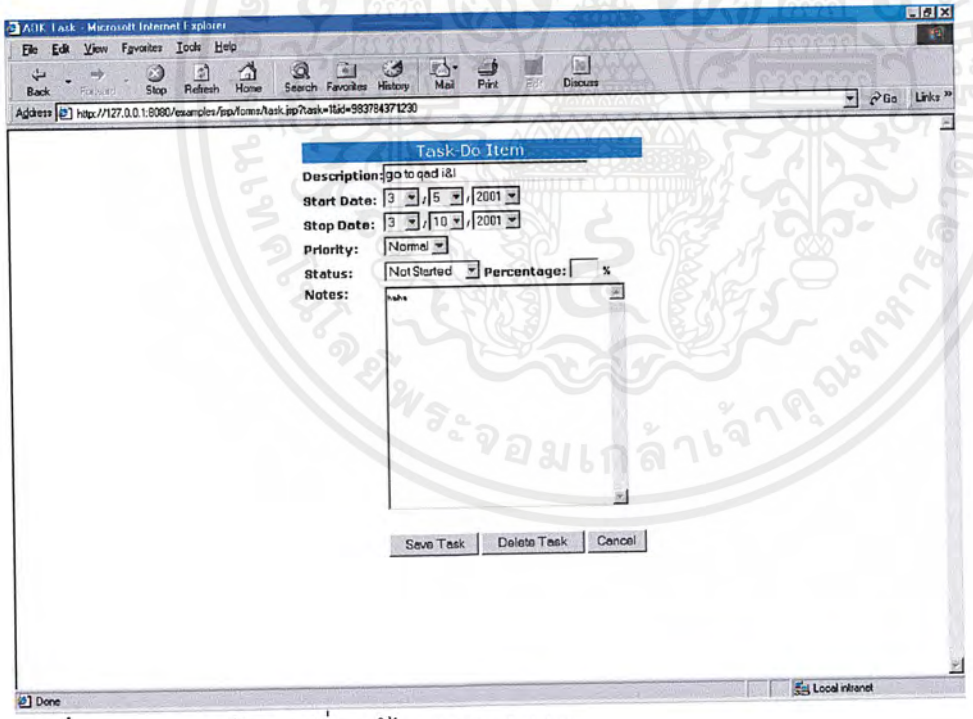
- Descriptionจะเป็นการอธิยายการทำงาน
- Start Date จะเป็นการกำหนดวันที่เริ่มทำงาน
- Stop Time จะเป็นการกำหนดวันที่สิ้นสุดทำงาน
- Priority จะเป็นการกำหนดความสำคัญของงาน
- Status จะเป็นการกำหนดสถานะการทำงาน
- Percentage จะเป็นการสถานะจำนวนเปอร์เซ็นต์ที่เสร็จของทำงาน
- Note จะเป็นการใส่ข้อความหมายเหตุของงาน

โดยในการจัดเก็บ task ก็มีการทำงานคล้ายกับในส่วนของ event โดยมีการใช้พารามิเตอร์เช่นเดียวกัน

```
String event = request.getParameter("task");
String id = request.getParameter("id");
```



รูปที่ 4.35 โฮมเพจแสดงหน้าการทำงานของ My Task List

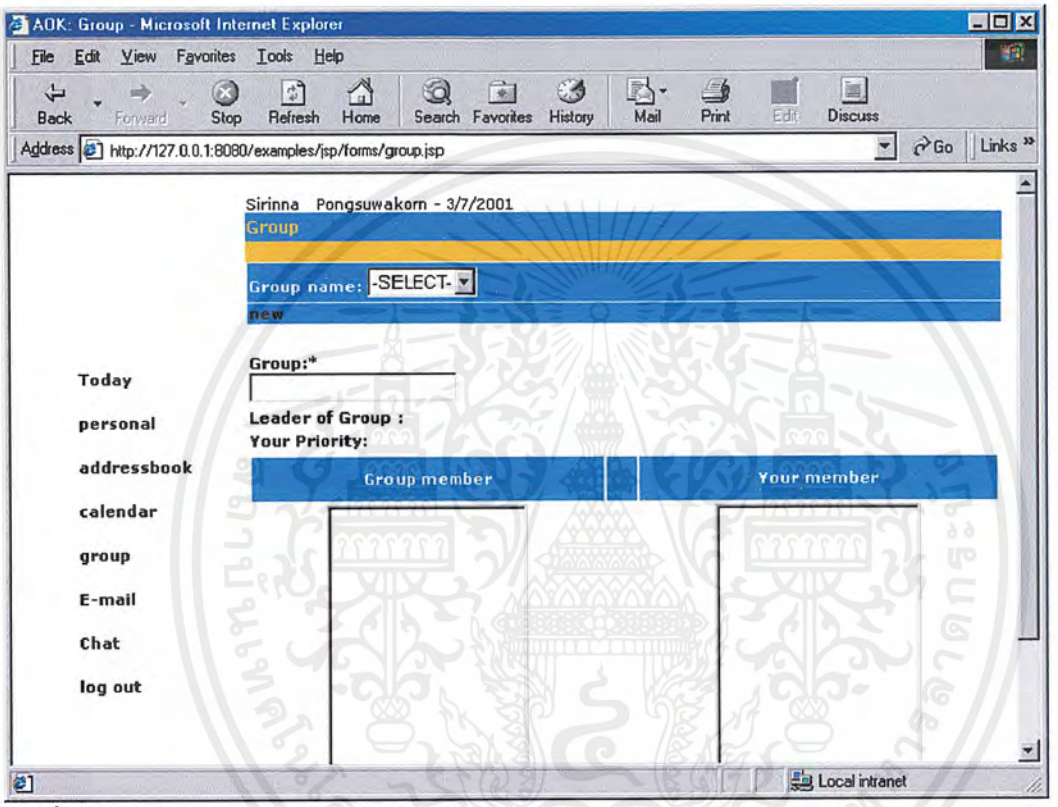


รูปที่ 4.36 แสดงหน้าการเพิ่ม แก้ไข ลบ Task List

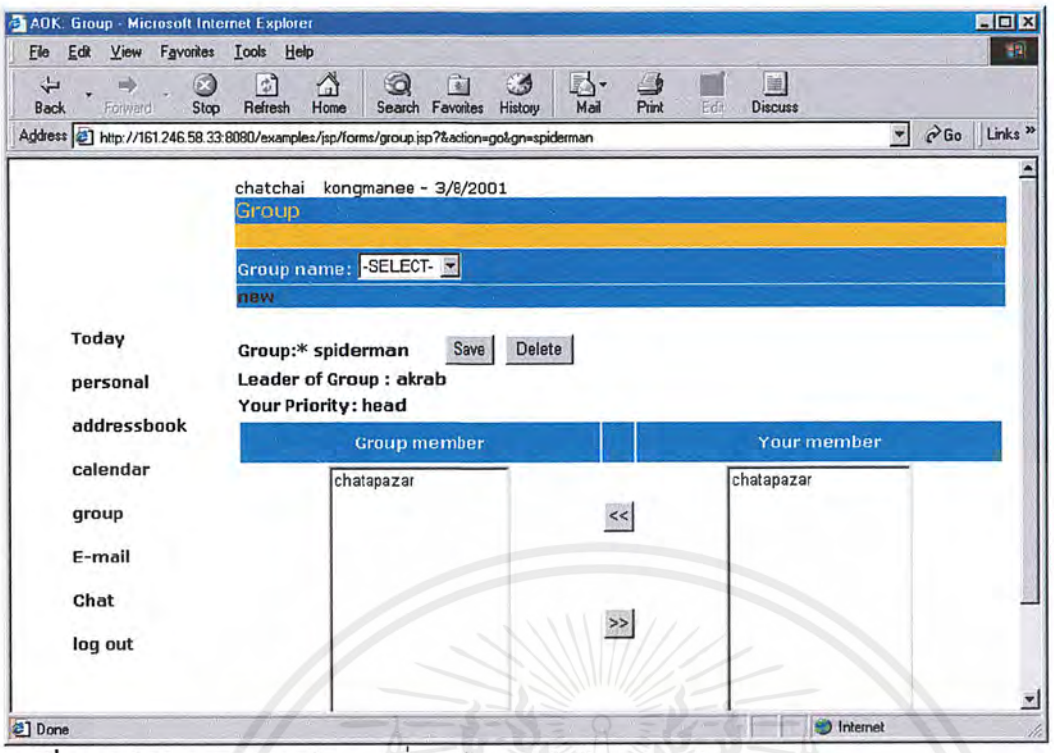
4.10 โฮมเพจหน้าการทำงานของ Group

ในการสร้างกลุ่มของผู้ใช้งาน สามารถสร้างได้ด้วยเมนู Group (รูปที่ 4.37) โดยการสร้างกลุ่มผู้ใช้งานร่วมกันสามารถเลือกที่ New จะปรากฏโฮมเพจให้ใส่ชื่อกลุ่มของผู้ใช้งาน(รูปที่ 4.38) และสามารถที่จะเพิ่มหรือลดสมาชิกเข้ากลุ่มได้โดยเลือก ปุ่ม <<เป็นการเพิ่มสมาชิก ปุ่ม>>เป็นการลดสมาชิก

**ผู้ที่ทำการแก้ไขกลุ่มต้องเป็นผู้นำของกลุ่มเท่านั้น



รูปที่ 4.37 โฮมเพจแสดงหน้าการทำงานของ Group



รูปที่ 4.38 โฮมเพจแสดงหน้าการเพิ่ม แก้ไขกลุ่มผู้ใช้งานร่วมกัน

การทำงานกับ group จะมีการรับพารามิเตอร์ action เข้ามาเพื่อทำการตรวจสอบว่าต้องการทำอะไร

```
String action = request.getParameter("action");
```

โดยจะมี action หลัก ๆ อยู่ 2 action คือการสร้างกลุ่มขึ้นมาใหม่กับการเปิดกลุ่มที่มีอยู่แล้วขึ้นมาจัดการ โดยถ้า action เท่ากับ new ก็จะทำการสร้างกลุ่มผู้ใช้ขึ้นมาใหม่ หากเท่ากับ go ก็จะทำการเปิดกลุ่มผู้ใช้นั้นขึ้นมาดังตัวอย่าง

```
<select name="selectgroup" onChange="MM_jumpMenu('parent',this,0)">
    <option selected>-SELECT-</option>
    <%
                                String gn = "";
                                try{
                                ResultSet rs =
pool.DBUtilities.connectedDB("SELECT group.groupname "+
                                "FROM [group] "+
```

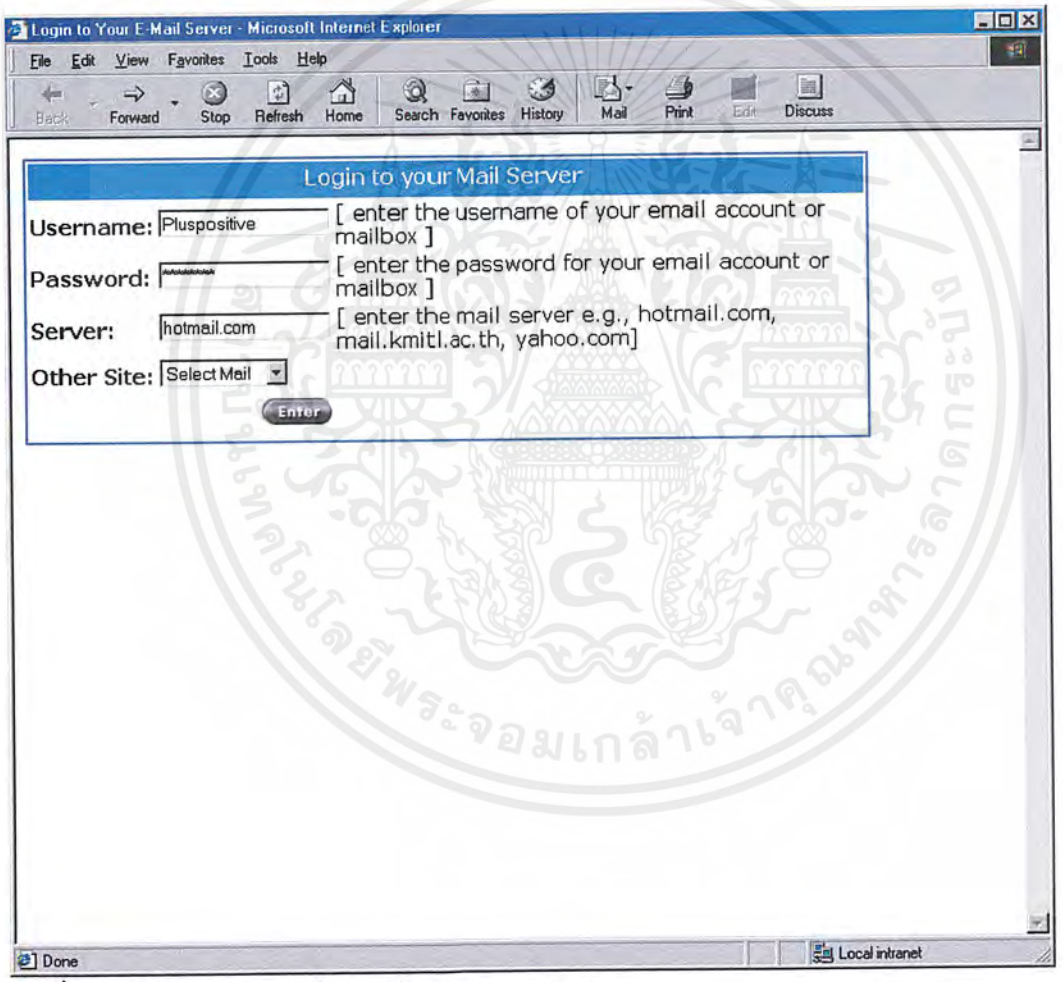
```
                                        "WHERE
(((group.username)='"+user.getUsername()+"')) order by groupname;");
                                        while(rs.next())
                                        {
                                                gn = rs.getString("groupname");
                                        %>
        <option value="group.jsp?&action=go&gn=<%=gn%>"><%=gn%></option>
        <%}
                                        rs.close();
                                        pool.DBUtilities.returnToPool();
                                        }catch(Exception e)
                                        {
                                                System.out.println("error is "+e);
                                        }
                                %>
        </select>
```

จาก code จะเห็นได้ว่าเป็นการสร้าง menu สำหรับให้ผู้ใช้เลือกไปยังกลุ่มที่ผู้ใช้สังกัดอยู่โดยในการทำงานกับกลุ่มถ้าผู้ใช้เป็นผู้สร้างขึ้นมาจะมีสิทธิเป็น leader ของกลุ่มแต่ถ้าไม่ใช่จะเป็น member โดยจะถูกจัดเก็บไว้ในตัวแปรเพื่อใช้กับกระบวนการจัดการ

4.11 โฮมเพจหน้าการทำงานของ E-Mail

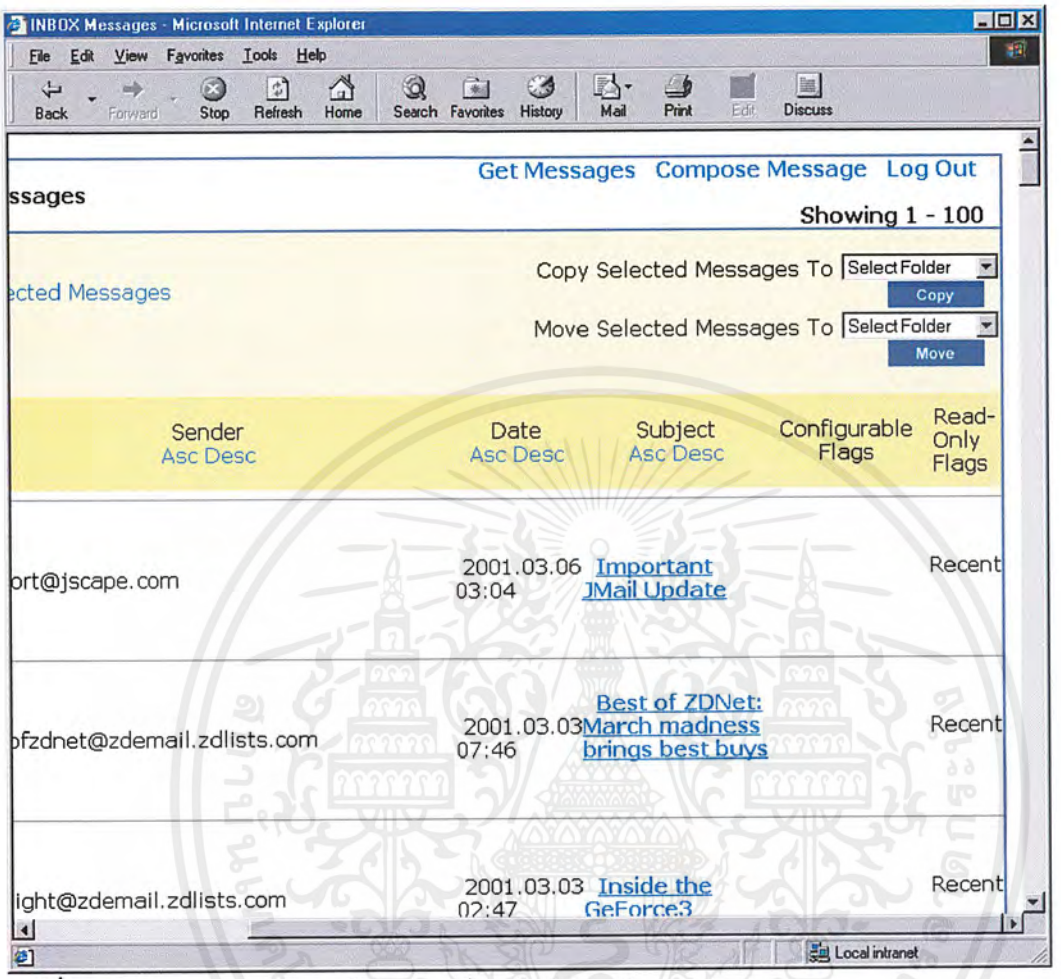
ในการรับและส่งอีเมล์ สามารถทำได้โดยเลือกเมนู E-Mail โดยจากเมนูจะปรากฏโฮมเพจ ให้ทำการ Login เข้าไปสู่อีเมล์เซิร์ฟเวอร์(รูปที่ 4.39) ซึ่งในการ Login เข้าไปสู่อีเมล์เซิร์ฟเวอร์จะมีการกรอกข้อมูลต่างๆ คือ

- Username ชื่อการใช้งาน
- Password รหัสผ่านในการเข้าใช้งาน
- Server ชื่ออีเมล์เซิร์ฟเวอร์ที่ต้องการติดต่อ
- Other Site ชื่ออีเมล์เซิร์ฟเวอร์ที่ทางระบบมีมาให้เลือก

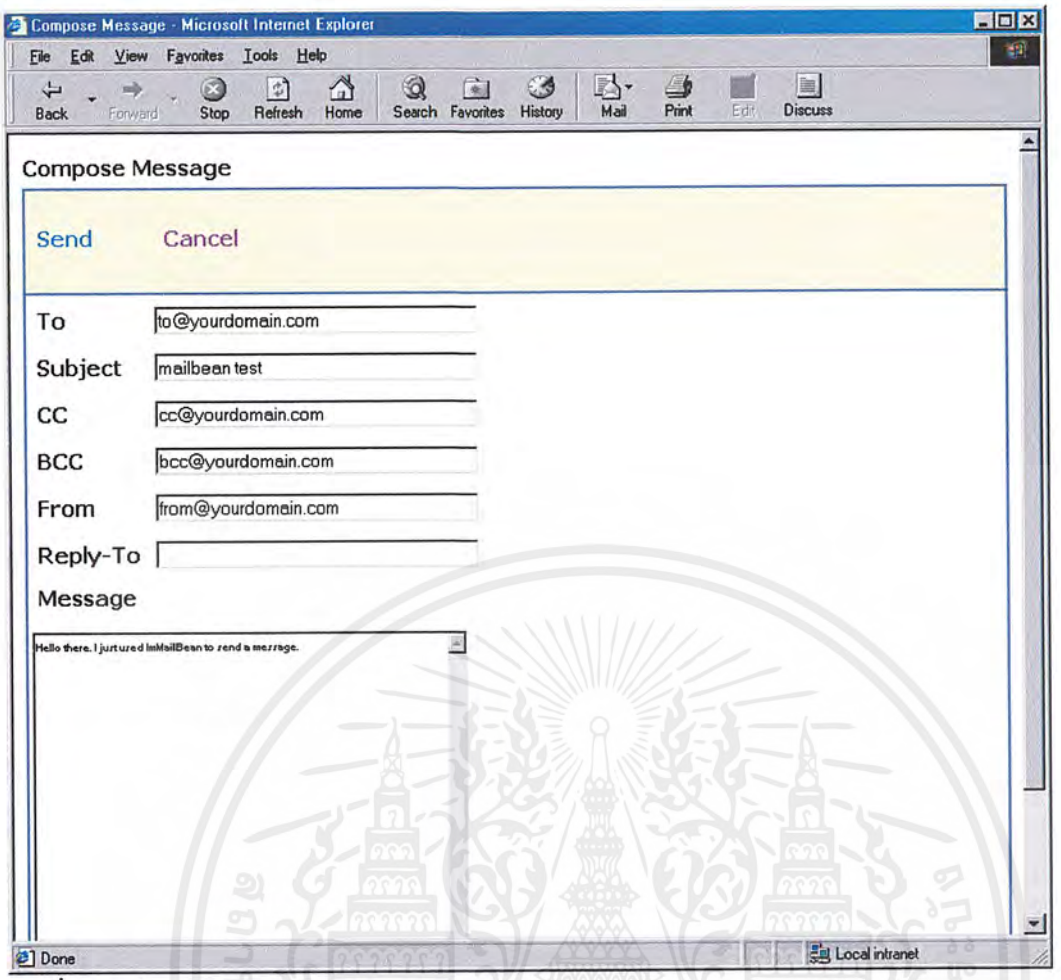


รูปที่ 4.39 โฮมเพจแสดงหน้าการทำงานของ E-Mail

เมื่อได้ติดต่อเข้าสู่อีเมล์เซิร์ฟเวอร์จะมีโฮมเพจแสดงรายการอีเมล์ทั้งหมดแสดงขึ้นมา(รูปที่ 4.40) และสามารถที่จะอ่าน หรือ ส่งอีเมล์จากเมนู Compose (รูปที่ 4.41)



รูปที่ 4.40 โฮมเพจแสดงการเข้าติดต่อสู่อีเมล์เซิร์ฟเวอร์



รูปที่ 4.41 โฮมเพจแสดงการส่ง E-Mail

ในการทำงานกับระบบ E-mail จะมีการสร้าง ImMailBean ขึ้นมาเพื่อทำหน้าที่ในการจัดการกับระบบ E-mail เช่นถ้าต้องการอ่านค่า Folder ของ E-mail ออกมาก็ทำการเรียก method

```
imMailBean.getFolders();
```

เพื่อทำการรับค่า Folder ที่บรรจุ E-mail ไว้ หากต้องการอ่านเนื้อหาใน E-mail ก็ทำการเรียก method

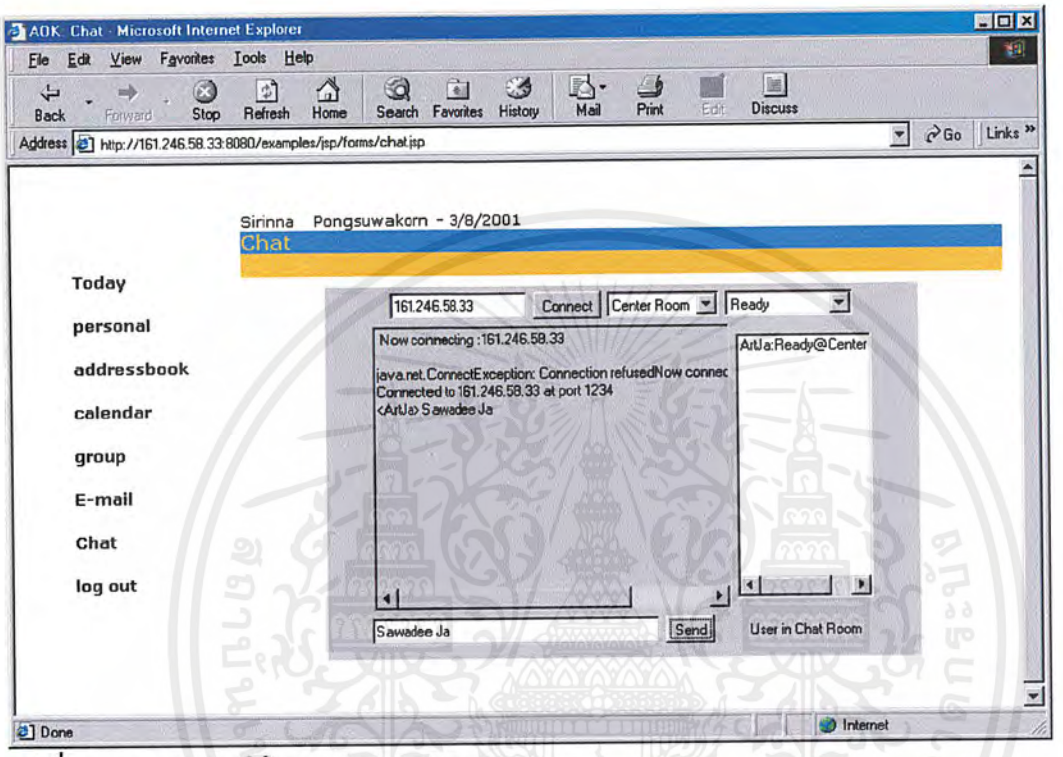
```
imMailBean.getMessages();
```

ก็จะได้ข้อความที่บรรจุอยู่ใน E-mail ออกมา

ส่วนการ login เพื่อเข้าสู่ E-mail Server ก็จะทำงานผ่าน mailbeanservlet ซึ่งเป็น Servlet ซึ่งทำหน้าที่ในการติดต่อกับระบบการจัดการไม่ว่าจะเป็นการ login, send, delete, trash เป็นต้น

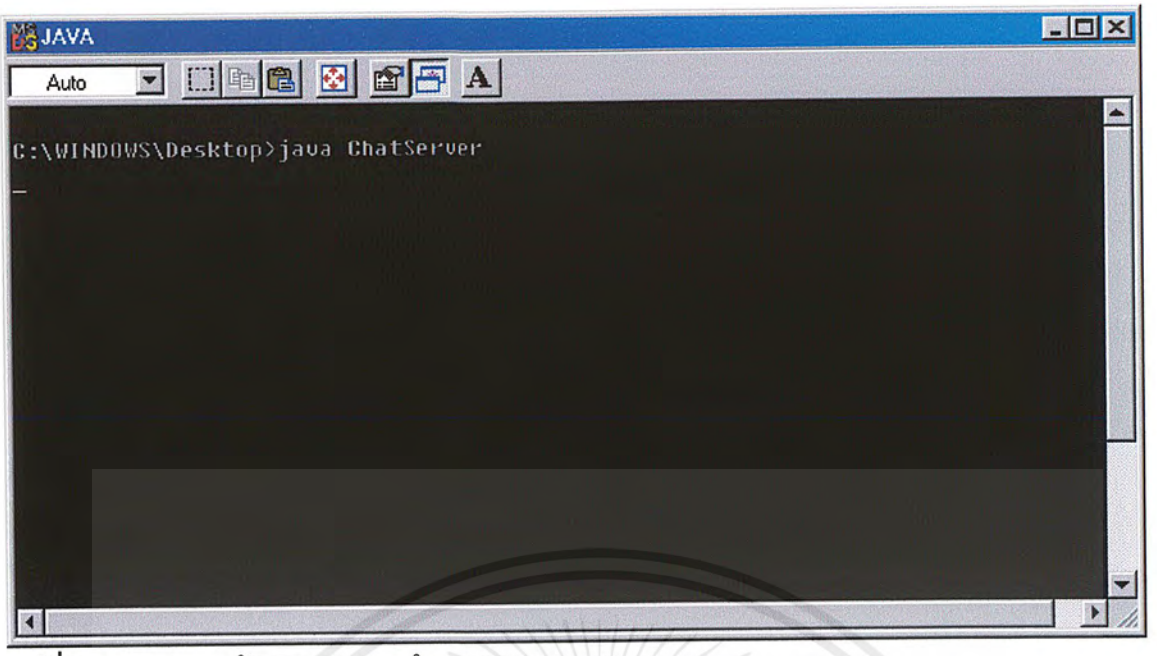
4.12 โฮมเพจหน้าการทำงานของ Chat

ในการทำงานของระบบสนทนา หรือ Chat จะประกอบไปด้วยหน้าจอสำหรับทำการสนทนา โดยในการใช้งานจะทำการใส่ชื่อผู้ใช้ และ ทำการเลือกห้องที่จะสนทนา จากนั้นทำการระบุ IP Address ของเครื่องที่ทำหน้าที่เป็น Server แล้วทำการ connect



รูปที่ 4.42 แสดงการใช้งานระบบสนทนา Chat

จากรูปจะเป็นในส่วนของ Client ซึ่งทำหน้าที่ในการใช้เป็นหน้าจอในการสนทนา โดยพัฒนาด้วย Java Applet โดยจะต้องทำการติดต่อกับ Chat Server โดยในส่วนของ Chat Server จะมีการทำการเปิด port คอยรับ request จาก Applet มีรูปแบบของ Chat Server ดังรูป



รูปที่ 4.43 แสดง หน้าจอแสดงการทำงานของ Chat Server

โดยการทำงานของ Chat Server จะมีรูปแบบดัง code ด้านล่างนี้

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {
  public static void main (String args[]) throws IOException {
    int port = 1234;
    ServerSocket server = new ServerSocket (port);
    while (true) {
      Socket client = server.accept ();
      System.out.println ("Accepted from " + client.getInetAddress ());
      ChatHandler handler = new ChatHandler (client);
      handler.start ();
    }
  }
```

}

จาก code จะเห็นได้ว่าจะมีการเปิด port รอรับการติดต่อจาก applet เข้ามา และทำการใช้ class ChatHandler สำหรับสร้าง thread การทำงานของ chat ขึ้นมาสำหรับตัว หน้าจอการทำงานจะทำเพียงเรียกใช้ applet Client.class โดยทั้งนี้จะต้องมีการระบุ IP ของ Chat Server ผ่าน ChatIP.properties ดังต่อไปนี้

```
# IP ของ Chat Server
ChatIP.IP = 161.246.58.29
```

ตัวอย่างการเรียก applet Client.class

```
<applet code = "Client" width = 500 height = 300>
<param name="username" value="<%=user.getUsername()%>">
</applet>
```

จะเห็นได้ว่าจะมีการ ส่งผ่านพารามิเตอร์ผ่าน applet ด้วยนั่นคือ username ของสมาชิกนั่นเองสำหรับติดต่อกับ Chat Server

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 บทสรุป

จากปัญหาพิเศษอันนี้ ทำให้เกิดระบบสำหรับทำการจัดการข้อมูลส่วนบุคคลและกลุ่ม ผ่านเครือข่ายอินเตอร์เน็ต ซึ่งเป็นการใช้ประโยชน์จาก อินเตอร์เน็ต ซึ่งปัจจุบันทุกคนสามารถเข้าถึงได้ง่ายและราคาถูก ทำให้การจัดการข้อมูลส่วนบุคคลและกลุ่มง่ายขึ้นและมีราคาถูก ทำให้ลดค่าใช้จ่ายในการที่จัดหาซื้อ Software หรือ Hardware เฉพาะทางมาใช้ ในส่วนของระบบยังประกอบไปด้วย ส่วนประกอบในการจัดการที่มีประโยชน์อาทิเช่น Scheduling, E-mail, Group, Address Book , Chat เป็นต้น

โดยในขั้นตอนการพัฒนาได้ทำการศึกษาหาข้อมูลในการพัฒนา Web Application จากหลาย ๆ แห่ง อาทิเช่น www.zkey.com, www.jspin.com, www.zaplet.com ซึ่งเป็น web ที่ทำการพัฒนาด้วย jsp และ Java Technology ซึ่งนอกจากจะได้ technology สำหรับในการพัฒนายังได้ข้อมูล และ แนวคิดจากพัฒนาจาก web เหล่านี้อีกด้วย

ในขั้นตอนการพัฒนาเริ่มจากการสร้าง โครงสร้างของระบบการออกแบบระบบฐานข้อมูลด้วย Entity Relationship Diagram การสร้าง Class Diagram สำหรับสร้าง Class สำหรับใช้งานในระบบตลอดจนการศึกษาและพัฒนาระบบ Web Server ด้วย Apache Tomcat และการศึกษาพัฒนาฐานข้อมูลด้วย JDBC

แต่เนื่องจากระยะเวลาและความสามารถอันจำกัดทำให้ระบบยังขาดส่วนประกอบที่ยังเป็นประโยชน์หรือ function ที่อำนวยความสะดวกที่ควรปรับปรุงในจุดต่าง ๆ ดังที่จะแสดงในส่วนข้อเสนอแนะ

5.2 ข้อเสนอแนะ

- การจัดการในส่วนของระบบ E-mail ยังไม่ดีพอไม่สามารถจัดการกับ Attach file ได้ และยังไม่สามารถติดต่อในการ forward mail จากผู้ให้บริการภายนอกไม่ได้จึงยังใช้ mail ที่ตั้งขึ้นเองกับของทางสถาบัน อยู่
- การจัดการ session ของระบบยังไม่ดีพอ ในเรื่องการตรวจเช็คผู้ใช้ในระบบ หากผู้ใช้ขาดการติดต่อกับระบบเป็นเวลานานหรือ ไม่ได้ logout จากระบบ จะทำให้สถานะของผู้ใช้ไม่ได้รับการแก้ไข ซึ่งทำให้ไม่สามารถ เข้าสู่ระบบในครั้งต่อไปได้

- ระบบยังขาดส่วนของจัดการไฟล์ข้อมูล ซึ่งจะสามารถเพิ่มประสิทธิภาพของระบบในเรื่องการจัดการไฟล์ข้อมูล การใช้งานไฟล์ข้อมูลร่วมกันซึ่งจะเป็นประโยชน์อย่างมาก

- ยังขาดในเรื่องของความปลอดภัยในการ encrypt ข้อมูลในระหว่างการส่งข้อมูลของผู้ใช้กับระบบ ซึ่งถือว่าเป็นเรื่องสำคัญหากนำระบบไปใช้ในเชิงพาณิชย์

- ระบบการเตือน สามารถทำงานได้เฉพาะตอนติดต่อกับระบบอยู่เท่านั้น ซึ่งควรจะมีการพัฒนาให้สามารถเตือนผ่าน มือถือ หรือ pager หรือผ่านทางโปรแกรมโดยไม่ต้องติดต่อกับระบบอยู่ (แต่ติดต่อกับ Internet อยู่)

- ระบบ สนทนาส่วนบุคคล หรือ กลุ่ม ยังไม่สามารถสร้างกลุ่มของการสนทนาขึ้นมาได้ ต้องมีการพัฒนาให้มีการสนทนาได้เฉพาะกลุ่มซึ่งจะต้องมีการพัฒนาต่อไป

บรรณานุกรม

Pratik Patel and Karl Moss. **Java Database Programing with JDBC**. Scottsdale, AZ.: Coriolis Group (1996).

Mike Cohn and Bryan Morgan. **Java Developer's Reference**, USA.:Sams.net(1996).

Piroz Mohseni. **Web Database Primer Plus**, USA.:Waite Group(1996).

James Duncan Davidson and Danny Coward. 1999. **Java Servlet Specification, V2.2 Final Release**.CA

Karl Moss. 1999. **Java Servlets**. McGraw-Hill Companies

JDK 1.2 Documentation [on-line]. Available HTTP:
http://www.javasoft.com/products/jdk/1.2/docs/index.html

JDBC Guide:GettingStarted[on-line] . Available HTTP:
http://www.math.sc.chula.ac.th/javadoc/guide/jdbc/getstart/introTocdoc.html

Table of Contents [on-line].Available HTTP:
http://www.sybase.com/products/internet/jdbcconnect/docs/jcref_2.htm

Oracle 7 JDBC driver [on-line]. Available HTTP:
http://www.oracle.com/st/products/st/products/jdbc/html/jdbc.html

Dale Dougherty. 1997. **Understanding Java Servlets**. [Online]. Available :
http://www.webreview.com/97/10/10/feature/colton.html

Cynthia Bloch. 1999. **Servlets**. [Online]. Available :
http://java.sun.com/docs/books/tutorial/servlets/index.html

Marty Hall. 1999. **Servlet Tutorial : A brief introduction to Java Servlets and Java Server Pages**. [Online]. Available :
http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/

Thomas Moore.1999. **Servlets Taverne : The Servlet Faq**. [Online]. Available :
http://wwwusers.imaginet.fr/~lse/JSS/textes/servfaq.htm

Donald Fischer. 2000. **Oracle Java Roadmap : Java Servlets**. [Online]. Available :
http://technet.oracle.com/tech/java/jroadmap/servlets/listing.htm.

Javasoft. 2000. **Fundamentals of Java Servlets**. [Online]. Available :
http://developer.java.sun.com/developer/Servlets/Fundamentals/introduction.html

Javasoft. 2000. **Interface javax.servlet.Servlet**. [Online]. Available :
http://java.sun.com/product/servlet/2.1/api/javax.servlet.Servlet.html

Javasoft. 2000. **JavaServer Developer Documentation**. [Online]. Available :
http://jserv.javasoft.com/products/webserver1.0.2/index_developer.html

Tom Laszewski. 2000.**Oracle Java Roadmap : Java Server Pages**. [Online]. Available :
http://technet.oracle.com/tech/java/jroadmap/jsp/listing.htm