ช็อปปิ้งเอเจนต์

Shopping Agent

โดย

นาย ธนพล อัศวกุล

นาย ธนาพงศ์ จารุสถิระกุล

อาจารย์ที่ปรึกษา

ผศ. อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ช็อปปิ้งเอเจนต์

Shopping Agent

ผู้จัดทำ

1. นาย ธนพล อัศวกุล รหัสประจำตัว 40010292
2. นาย ธนาพงศ์ จารุสถิระกุล รหัสประจำตัว 40010298

_____ อาจารย์ที่ปรึกษา

(ผศ. อภิเนตร อูนากูล)

ช็อปปิ้งเอเจนต์

นาย ธนพล อัศวกุล 40010292
นาย ธนาพงศ์ จารุสถิระกุล 40010298
ผศ. อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

อินเทอร์เน็ตได้กลายเป็นสิ่งที่จำเป็นในปัจจุบันไปแล้ว ทั้งที่เพิ่งได้รับความนิยมเมื่อไม่กีปีที่ผ่านมาแต่อัตราการเติบโตของข่าวสารข้อมูลและจำนวนผู้ใช้นั้นเป็นไปอย่างรวดเร็ว ทำให้ธุรกิจต่างๆเริ่มหันมาสนใจที่จะทำธุรกิจผ่านทางอินเทอร์เน็ต (Ecommerce) กันมากขึ้น ด้วยเหตุนี้เอง ผู้ใช้อินเทอร์เน็ตจะต้องใช้เวลามากในการค้นหาข้อมูลข่าวสารที่ตนต้องการ

ดังนั้น จึงได้มีการพัฒนาซอฟท์แวร์เอเจนต์ขึ้นเพื่อแก้ไขปัญหาดังกล่าว โดยซอฟท์แวร์เอเจนต์ต่างๆที่มีใช้อยู่บนอินเทอร์เน็ตนั้นโดยมากจะอยู่ในประเภทของเสิร์ชเอ็นจิน ซึ่งมีหน้าที่หลักในการท่องไปตามลิงก์ของเว็บเพจแล้วทำการเก็บข้อมูลที่ใช้ในการบ่งชี้ประเภทและเนื้อหาของเว็บเพจนั้นๆ แต่ว่าข้อจำกัดของการใช้เสิร์ชเอ็นจินก็คือ ผู้ใช้จะได้รับผลลัพธ์ที่เป็นลิงค์ของเว็บเพจมาเท่านั้น จากนั้นผู้ใช้จะต้องเข้าไปอ่านแล้วกลั่นกรองข้อมูลที่ต้องการออกมาเอง

ด้วยเหตุนี้จึงได้มีการค้นคว้าวิจัยทางด้านการแยกสาระที่สำคัญออกมาจากเว็บเพจ (Information Extraction) เพื่อให้ผู้ใช้ได้รับข้อมูลที่ต้องการโดยไม่ต้องมาทำการกลั่นกรองเอง เมื่อมีการนำ Information Retrieval กับ Information Extraction มาใช้ร่วมกันก็จะทำให้ผู้ใช้ได้รับประโยชน์อย่างเต็มที่

และด้วยเหตุที่อีคอมเมอร์สกำลังเป็นเติบโตเป็นอย่างมาก โครงนี้จึงได้นำแนวคิดของ Information Retrieval และ Information Extraction มาใช้ด้วยกันกับอีคอมเมอร์ส ซึ่งจะนำมาอิมพลิเมนต์กับการสร้างช็อปปิ้งเอเจนต์สำหรับการค้นหาข้อมูลของอพาร์ทเมนต์และคอนโดมีเนียมเพื่อการเช่าและซื้อ

Shopping Agent

Thanapol Asavakul
Thanapong Charusadhirakul
Assit.Prof. Apinetr Unakul

Abstract

The World Wide Web has gained important role in the field of information technology ever since it emerged . Ecommerce is revolutionizing the way tomorrow's businesses run. The rapid growth of internet users has led the growth of online merchants. Large amount of information on the world wide web makes it difficult to find the needed information. Buyers are facing difficult task finding their needed products on the world wide web.

Software agents have been developed to help ease the task of human users. Search engine is one example of software agents used for the world wide web. The main task of search engine is to help users find the related documents reside on the world wide web. The main limitation of Search engine is that it does not help user find specific product rather it helps user find the related webpages,which may or may not contain the wanted product.

This has led to the birth of Shopping agent, yet another type of software agent, which its main task is to gather specific products from different online merchants, summarize the search result and present user the useful information.

This thesis is concerned with the analysis and design of shopping agent, focusing on data extraction from HTML webpages, which is considered not well-structured. The implementation uses condominium and apartment as product domain.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่ประสบความสำเร็จได้ ถ้าหากปราศจากความช่วยเหลือและร่วมมือจากหลายๆฝ่าย ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ อภิเนตร อูนากูล ที่ให้คำแนะนำ คำปรึกษาและดูแลอย่างเอาใจใส่ ขอขอบคุณพี่ต๋วยและพี่ชิตที่คอยแนะนำในหลายๆเรื่อง ขอบคุณพี่ๆทุกคนในห้องฮาร์ดแวร์ที่ให้ความช่วยเหลือทุกๆด้านทั้งความรู้ อาหาร และ บันเทิง ขอบคุณนายก้วย ผู้แสนรู้ในจาวา นายช่วยแห่งห้องโอ้ล่าล๊าที่คอยมาแหกปากให้เพื่อนๆไม่หลับ นายโป้งที่มีคอมให้เล่นเคาน์เตอร์สไตรค์ เพ็ญที่อยู่กับโป้ง นิ่มและก้อยที่มีขนมมาให้กิน นายหนอนที่ชอบมาร้องเพลงให้ฟัง นายหนู ฮันเตอร์ ไรอัน ไวโอเล็ต ป๊า และเดอะร็อคที่ชวนเล่นเกม ขอบคุณเพื่อนๆในห้องโอ้ล่าล๊า ห้องเน็ตเวิร์ค ห้องไอแซค และทุกๆคนที่ไม่ได้กล่าวถึงในที่นี้

สุดท้ายนี้ขอกราบขอบพระคุณบิดา-มารดา และทุกคนในครอบครัวที่ให้การเลี้ยงดู ให้การศึกษาและอบรมให้ข้าพเจ้าเป็นอย่างดี

นาย ธนพล อัศวกุล

นาย ธนาพงศ์ จารุสถิระกุล

สารบัญ

สารบัญรูปภาพ

บทที่ 1 บทนำ

1.1 ความสำคัญและที่มา

อินเทอร์เน็ตได้รับความนิยมและใช้กันอย่างแพร่หลายภายในระยะเวลาอันสั้น ในขณะที่จำนวนผู้ใช้เพิ่มขึ้น จำนวนของข้อมูลบนอินเทอร์เน็ตก็ได้เพิ่มขึ้นตามการเติบโตของผู้ใช้อย่างรวดเร็วเช่นกัน ทำให้อินเทอร์เน็ตกลายเป็นแหล่งข้อมูลที่ใหญ่ที่สุดแห่งหนึ่ง ซึ่งทำให้การค้นหาข้อมูลบนอินเทอร์เน็ตเป็นเรื่องที่ยาก โดยเฉพาะข้อมูลสำหรับการซื้อสินค้า เพราะถึงแม้ปัจจุบันจะมีเสิร์ชเอ็นจิน (Search Engine) สำหรับการค้นหาเว็บเพจที่ต้องการก็ตาม แต่ว่าผู้ใช้จะต้องเข้าไปอ่านเว็บเพจเหล่านั้นและกลั่นกรองข้อมูลที่ต้องการเอง อีกทั้งการซื้อขายสินค้ามีแนวโน้มที่จะทำผ่านทางอินเทอร์เน็ตมากขึ้นในอนาคต จึงจำเป็นต้องมีการพัฒนาระบบสำหรับการค้นหาสินค้าบนอินเทอร์เน็ตขึ้นซึ่งเป็นที่มาของโครงงานนี้

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษานิยามและคุณสมบัติของช็อปปิ้งเอเจนต์

1.2.2 เพื่อศึกษากระบวนการค้นหาเว็บเพจที่มีข้อมูลที่ต้องการ (Information Retreival)

1.2.3 เพื่อศึกษากระบวนการคัดเลือกข้อมูลที่ต้องการออกมาจากเว็บเพจ (Information Extraction)

1.2.4 เพื่อศึกษาและวิเคราะห์ระบบที่มีอยู่ในปัจจุบัน

1.2.5 เพื่อศึกษาเทคโนโลยีที่เกี่ยวข้องในการพัฒนาระบบ

1.2.6 เพื่อออกแบบและพัฒนาระบบให้สามารถใช้งานได้จริง

1.2.7 เพื่อศึกษาหาแนวทางปรับปรุงระบบให้ดียิ่งขึ้น

1.3 ขอบเขตของโครงงาน

1.3.1 โครงงานนี้เป็นเพียงแค่การนำเสนอข้อมูลของสินค้าเท่านั้น จะไม่ครอบคลุมถึงการซื้อขายสินค้า

1.3.2 สินค้าในโครงงานนี้จะเป็นอพาร์ทเมนต์และคอนโดมีเนียมในกรุงเทพ

1.3.3 ข้อมูลของสินค้าที่ได้จะเป็นข้อมูลที่มาจาก

1.3.3.1 เว็บเพจของพ่อค้าที่มีสินค้าเป็นคอนโดมีเนียมและอพาร์ทเมนต์ โดยข้อมูลจะอยู่ในรูปของ HTML

1.3.3.2 ผู้ที่มีสินทรัพย์ส่วนตัวและต้องการจะเสนอขายหรือเช่า มาประกาศเอาไว้ โดยที่จะต้องทำการสมัครเป็นสมาชิกก่อน

1.3.4 จำนวนของเว็บไซต์ที่เป็น HTML ที่ใช้ทดสอบนั้นจะจำกัดไว้ที่ 5 เว็บไซต์โดยใช้เว็บไซต์ของไทย

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาขั้นตอนการซื้อสินค้าบนอินเทอร์เน็ต

1.4.2 ศึกษาแนวคิดของเอเจนต์

1.4.3 ศึกษาหลักการค้นหาเว็บเพจที่เกี่ยวข้อง (Information Retrieval)

1.4.4 ศึกษาหลักการคัดเลือกข้อมูลที่ต้องการออกจากเว็บเพจ (Information Extraction)

1.4.5 ศึกษาระบบที่มีอยู่แล้ว

1.4.6 วิเคราะห์และออกแบบระบบการค้นหาสินค้าบนอินเทอร์เน็ต

1.4.7 ศึกษาเครื่องมือต่างๆ ที่ใช้ในการออกแบบและพัฒนาระบบ

1.4.8 ออกแบบและพัฒนาระบบโดยใช้วิธีการออกแบบเชิงวัตถุ

1.4.9 พัฒนาระบบตามที่ได้ออกแบบไว้

1.4.10 ทดสอบและปรับปรุงระบบ

1.4.11 สรุปและเสนอแนวทางพัฒนาต่อไป

บทที่ 2 หลักการและทฤษฎี

2.1 ความเป็นมาของช็อปปิ้งเอเจนต์

2.1.1 ความหมายของช็อปปิ้งเอเจนต์ (Shopping Agent)

ช็อปปิ้งเอเจนต์ คือ ผู้ช่วยในการซื้อสินค้าหรือบริการ โดยช็อปปิ้งเอเจนต์อาจจะช่วยผู้ใช้ในด้านการค้นหาสินค้าและเปรียบเทียบราคาจากผู้ขายหลายๆรายบนอินเทอร์เน็ต ซึ่งโดยปกติแล้วจะเป็นการช่วยหาข้อมูลเท่านั้น ส่วนของการติดต่อซื้อขายนั้นผู้ใช้จะต้องติดต่อกับผู้ขายเอง

2.1.2 การซื้อสินค้าผ่านอินเทอร์เน็ต

ในการซื้อสินค้านั้นสามารถแบ่งเป็นขั้นตอนต่างๆได้ดังนี้

1. *ระบุความต้องการ (Need Identification)* คือ การที่ผู้ซื้อจะต้องรู้ว่าตนมีความต้องการอะไร เช่น ต้องการจะหาที่อยู่
2. *หาสินค้าที่ต้องการ (Product Brokering)* คือ การที่ผู้ซื้อหาสินค้าที่ตนต้องการ เช่น จำนวนห้องนอนที่ต้องการ จำนวนห้องน้ำ ขนาดของห้อง หรือ สาธารณูปโภค
3. *หาผู้ขาย (Merchant Brokering)* คือ ขั้นตอนการหาว่าจะหาสินค้าที่ต้องการได้จากผู้ขายรายใด
4. *เจรจา (Negotiation)* คือ ขั้นตอนการเจรจาตกลงรายละเอียดการซื้อ
5. *ชำระเงินและส่งสินค้า (Payment & Delivery)* คือ ขั้นตอนการชำระเงินและรับสินค้า
6. *บริการและประเมินผล (Service & Evaluation)* คือ ขั้นตอนการรับบริการและประเมินผลความพอใจ

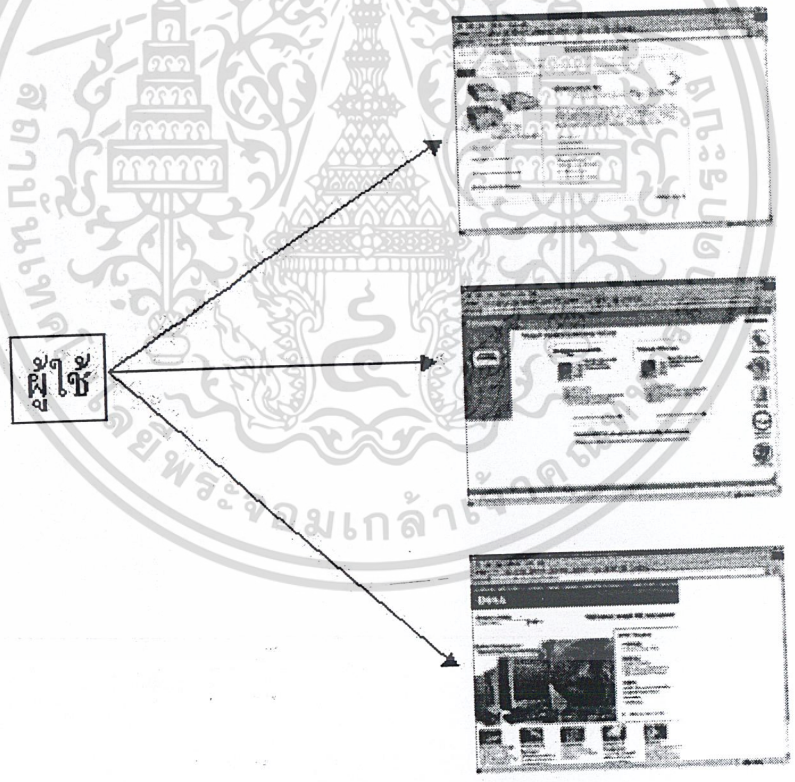
รูปภาพ 2-1 แผนภาพของการซื้อสินค้าบนอินเทอร์เน็ต

2.1.3 กำเนิดของช็อปปิ้งเอเจนต์

เนื่องจากการค้นหาข้อมูลบนอินเทอร์เน็ตในอดีตนั้นเป็นเรื่องยากเนื่องจากมีเว็บไซต์อยู่มากมาย จึงได้มีการสร้างเสิร์ชเอ็นจินสำหรับการค้นหาข้อมูลที่ต้องการบนอินเทอร์เน็ต แต่เมื่ออินเทอร์เน็ตได้พัฒนาไปจนมีการซื้อขายสินค้า การใช้เสิร์ชเอ็นจินในการค้นหาสินค้านั้นไม่เหมาะสมเนื่องจาก

- เสิร์ชเอ็นจินจะเก็บข้อมูลจากเว็บเพจที่เป็นแบบ Static เท่านั้น ในขณะที่ข้อมูลสินค้าบนอินเทอร์เน็ตแทบทั้งหมดจะเป็นเว็บเพจที่ถูกสร้างขึ้นในทันทีแบบ Dynamic
- เนื่องจากเหตุผลในข้อแรกทำให้เสิร์ชเอ็นจินสามารถค้นหาเว็บไซต์ขายสินค้าที่ผู้ใช้ต้องการเท่านั้น แต่ไม่สามารถดูรายละเอียดของสินค้าได้
- ผู้ใช้จะต้องท่องไปทุกเว็บไซต์ที่ขายสินค้าที่ต้องการเพื่อทำการเปรียบเทียบราคาและสินค้า ซึ่งเป็นเรื่องที่ยากลำบากและใช้เวลามาก

จากปัญหาดังกล่าว ทำให้มีการคิดสร้างระบบช่วยการค้นหาสินค้า หรือ ช็อปปิ้งเอเจนต์ขึ้นเพื่ออำนวยความสะดวกให้กับผู้ใช้



รูปภาพ 2-2 ก่อนมีช็อปปิ้งเอเจนต์

รูปภาพ 2-3 ระบบที่มีช็อปปิ้งเอเจนต์

2.1.4 ตัวอย่างของช็อปปิ้งเอเจนต์

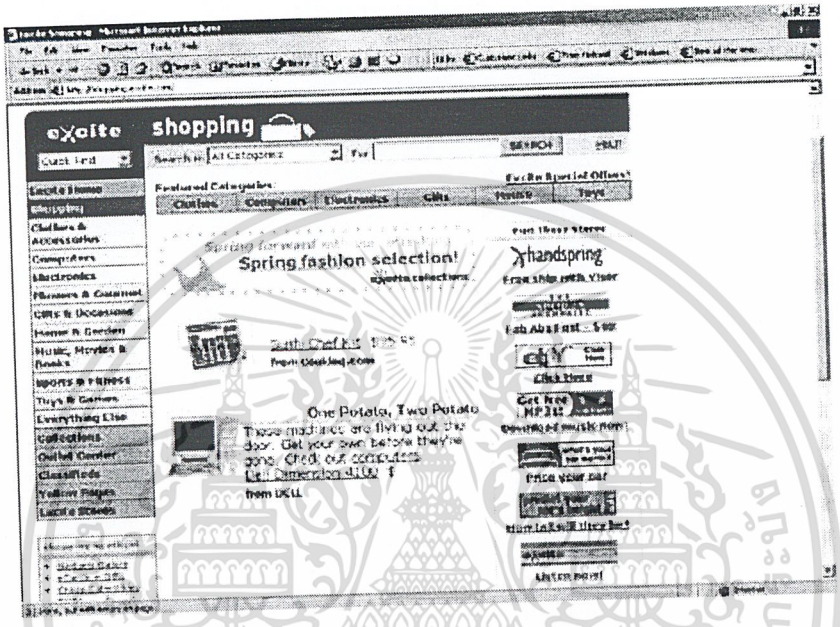
ในปัจจุบันมีเอเจนต์บนอินเทอร์เน็ตที่ช่วยผู้ใช้ในการซื้อสินค้าออนไลน์อยู่มากมาย ซึ่งแต่ละตัวมีโครงสร้างและแนวทางที่แตกต่างกันไป โดยเอเจนต์บางตัวจะเป็นการช่วยตัดสินใจว่าควรจะซื้ออะไร ในขณะที่บางตัวจะช่วยเปรียบเทียบราคา หรือ ค้นหาผู้ขายที่มีสินค้าตรงตามความต้องการ

2.1.4.1 Jango

Jango เป็นเอเจนต์ ของ NETbot โดยชื่อเดิมของ Jango คือ Shopbot สร้างโดยอิงบนงานวิจัยของ Oren Etzioni และ Dan Weld แห่งมหาวิทยาลัยวอชิงตัน ในเดือนตุลาคม 2540 Excite ซื้อ NETBot ในราคา 35 ล้านดอลลาร์ การทำงานของ Jango จะทำแบบออนไลน์ คือ เมื่อมีการร้องขอมาจากผู้ใช้ Jango จะทำการค้นหาข้อมูลบนอินเทอร์เน็ตในทันที โดยจะไม่มีการนำข้อมูลมาเก็บไว้ที่ Jango ก่อน
Jango ประกอบด้วย 4 ส่วน ได้แก่

- Natural Language Front-End ทำหน้าที่แปลงคำร้องขอของผู้ใช้ที่เป็นภาษาธรรมชาติให้ระบบเข้าใจ
- Query router ทำหน้าที่ตีความหมายว่าสินค้าอยู่ในหมวดใดและเกี่ยวข้องกับเว็บไซต์ใด
- Aggregation Engine ทำหน้าที่ร้องขอข้อมูลจากหลายๆเว็บไซต์พร้อมๆกัน
- Filter ดึงข้อมูลออกมาจากเว็บไซต์ต่างๆ โดยใช้การวิเคราะห์แบบ Shopbot

Jango จะเรียนรู้วิธีการค้นหาสินค้าจากเว็บไซต์ในช่วงของการเรียนรู้โดยใช้ URL ของเว็บไซต์ร้านค้าและความรู้ด้านสินค้า ในช่วงนี้ Jango จะเรียนรู้ถึงการฟอร์แมตของการแสดงผลว่ารายละเอียดของสินค้าถูกแสดงออกมาอย่างไรเพื่อที่จะคัดเลือกข้อมูลเกี่ยวกับสินค้าออกมาจาก HTML ได้ เช่น ราคา เป็นต้น ในระหว่างช่วงการช็อปปิ้งนั้นความรู้ที่ได้มาจะถูกนำมาใช้ในการคัดเลือกข้อมูล โดยการคัดเลือกข้อมูลจะทำกับหลายเว็บไซต์พร้อมๆกันและผลที่ได้ก็จะถูกนำมาเรียงโดยราคาแล้วเสนอให้กับผู้ใช้



รูปภาพ 2-4 Jango ของ Excite (www.excite.com)

2.1.4.2 Junglee

Junglee เริ่มในเดือน กรกฎาคม 2539 โดยบัณฑิตจากมหาวิทยาลัยแสตนฟอร์ด และถูกซื้อโดย Amazon ในเดือน สิงหาคม 2541 ในราคาประมาณ 180 ล้านดอลลาร์ Junglee ใช้ระบบ Virtual Database (VDB) คู่กับการใช้ HTMLและ XML ในการเสนอข้อมูลจากหลายๆแหล่ง

VDB ทำหน้าที่ในการรวบรวม,จัดการโครงสร้าง และ รวมข้อมูลจากแหล่งต่างๆให้กับผู้พัฒนาโปรแกรม โดยมี Wrapper เป็นตัวกลางระหว่างแหล่งข้อมูลในการแปลงให้ข้อมูลอยู่ในรูปของดาตาเบส

VDB มีส่วนประกอบหลัก 2 ส่วน ได้แก่ ระบบรวบรวมข้อมูล และ ระบบแสดงข้อมูล ข้อมูลจะถูกดึงออกมาจากแหล่งต่างๆโดยระบบรวบรวมข้อมูลและนำไปเก็บในดาตาเบส โดยการอัพเดตดาตาเบสนี้จะถูกกำหนดโดยระบบแสดงข้อมูล

ระบบรวบรวมข้อมูลมี 3 ส่วน ได้แก่ Wrapper, Mapper และ Extractor

1. Wrapper ทำหน้าที่เป็นตัวกลางสำหรับเว็บไซต์ต่างๆกับระบบ และสร้างโดยการเขียนโปรแกรมด้วยมือสำหรับแต่ละเว็บไซต์โดยเฉพาะ
2. Mapper แปลงข้อมูลที่ถูกดึงออกมาให้เป็นรูปแบบที่ต้องการ

3. Extractor ดึงโครงสร้างจากข้อมูลเทกซ์ที่ไม่มีโครงสร้างโดยการใช้ดิกชันนารีและกฎที่อธิบายถึงการดึงข้อมูลออกจากเทกซ์ โดยในทั้ง 2 กรณีจะใช้ภาษาที่ออกแบบมาสำหรับการสร้างกฎ ในระบบจะมี 1 wrapper ต่อ 1 เว็บไซต์ในขณะที่ extractor จะมีเพียงตัวเดียว

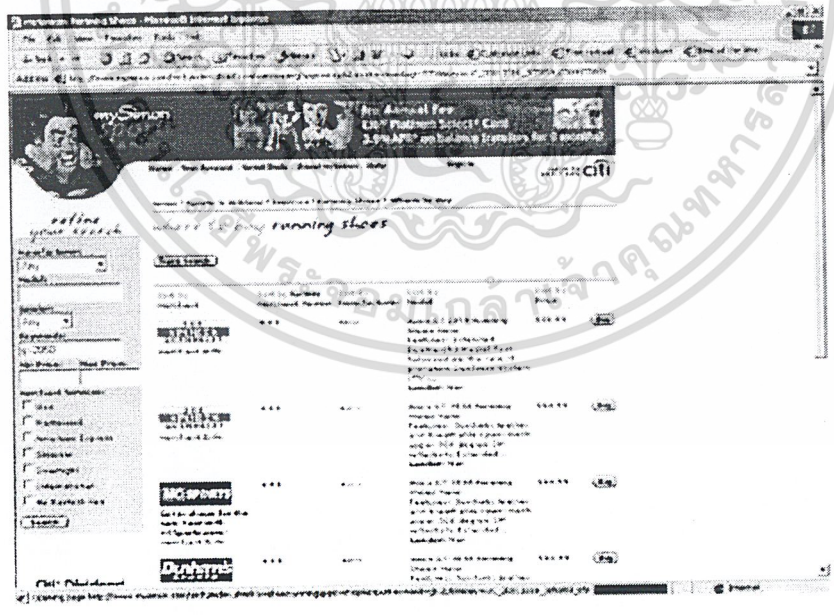
2.1.4.3 MySimon

MySimon สร้างโดย Michale Yang และ Yeogirl Yun ในเดือนเมษายน 2541 โดยใช้เทคโนโลยี Virtual Learning Agent(VLA) ที่ Yeogirl Yun เป็นผู้พัฒนาสำหรับเรียนรู้เว็บไซต์

VLA ทำงานโดยสร้าง intelligent agent หลายๆตัวที่สามารจำลองการซื้อสินค้าของมนุษย์ได้และสามารถเรียนรู้ที่จะดึงข้อมูลทีต้องการจากเว็บไซต์ได้

เอเจนต์จะถูกสอนให้ดึงข้อมูลจากแต่ละเว็บไซต์โดยผู้สอนในสภาวะแวดล้อมแบบ GUI โดยผู้สอน ซึ่งเรียกว่า Simon Product Intelligence (SPI) ไม่จำเป็นจะต้องเป็นโปรแกรมเมอร์ SPI จะทำการท่องเว็บไซต์ของพ่อค้าแล้วใช้ GUI ในการคัดลอกข้อมูลจากเว็บไซต์ไปสู่แบบฟอร์มออนไลน์ โคดจะถูกสร้างขึ้นอัตโนมัติโดยขึ้นอยู่กับพฤติกรรมของ SPIและข้อมูลที่ถูกคัดลอกออกมา

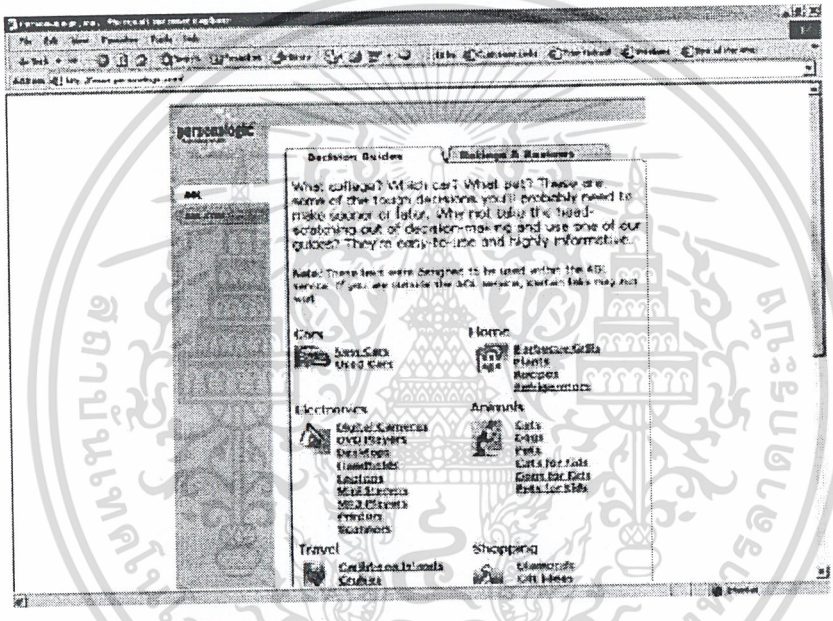
การทำงานของ mySimon จะทำงานแบบออนไลน์ คือ เมื่อมีผู้ใช้ป้อน Query เข้ามา ระบบจะทำการตีความหมายของ Query เพื่อหาโดเมนของสินค้า จากนั้นจะทำการค้นหาสินค้าจากเว็บไซต์ที่มีสินค้าอยู่ในโดเมนนั้นๆแบบทันที ซึ่งจะไม่มีการนำข้อมูลของสินค้ามาเก็บไว้ที่ดาตาเบสของระบบก่อน



รูปภาพ 2-5 ผลการค้นหาสินค้าด้วย mySimon (www.mysimon.com)

2.1.4.4 PersonaLogic

PersonaLogic เป็นเอเจนต์ที่ทำงานในส่วนของการแนะนำสินค้า (Product Brokering) โดยจะกรองผลิตภัณฑ์ที่ไม่ตรงตามข้อกำหนดของผู้ใช้ออกไปเรื่อยๆตามคำตอบที่ได้จากผู้ใช้ โดยผลที่ได้จะเป็นรายชื่อของสินค้าที่มีคุณสมบัติตามข้อกำหนดหลักของผู้ใช้ซึ่งจะเรียงลำดับโดยคุณสมบัติรองที่ผู้ใช้กำหนด จุดด้อยของ PersonaLogic ก็คือไม่ได้ช่วยผู้ใช้ว่าควรจะซื้อจากใคร และการที่ถามคำถามผู้ซื้อไปเรื่อยๆด้วยวิธีการ Deep Interview ทำให้เป็นการยากที่ผู้ซื้อจะกลับไปแก้คำตอบที่ต้องการจะแก้ ซึ่งหากผู้ใช้คุ้นเคยกับสินค้านั้นก็คงไม่มีปัญหา แต่ถ้าหากว่าผู้ใช้ไม่คุ้นเคยกับสินค้านั้นก็จะทำให้ผู้ใช้มีโอกาสที่จะต้องกลับไปแก้คำตอบสูง



รูปภาพ 2-6 ระบบช่วยตัดสินใจการซื้อสินค้า Personalogic ของ AOL

2.1.4.5 Firefly

Firefly เป็นเอเจนต์ที่ทำงานในขั้นตอนการแนะนำสินค้าเช่นกัน แต่ว่าแทนที่จะใช้วิธีการกรองสินค้าที่ไม่มีคุณสมบัติตามข้อกำหนดออกไป Firefly ใช้วิธีการแนะนำสินค้าด้วย automated collaborative filtering (ACF) ซึ่งวิธีการของ ACF คือขั้นแรก ACF จะทำการเปรียบเทียบการให้คะแนนสินค้าของผู้ใช้กับผู้ซื้อคนอื่น แล้วพิจารณาหาผู้ซื้อคนอื่นที่มีรสนิยมในการซื้อคล้ายๆกันกับผู้ใช้ จากนั้น ACF ก็จะแนะนำสินค้าที่ได้รับคะแนนสูงจากผู้ใช้ที่มีรสนิยมคล้ายๆกัน ก่อนที่ Firefly จะถูกไมโครซอฟท์ซื้อไปนั้น Firefly จะใช้แนะนำสินค้าพื้นๆ เช่น หนังสือ เพลง หรือ ภาพยนตร์ ส่วนจะด้อยของ Firefly ก็เหมือนกับ PersonaLogic ก็คือ ไม่ได้แนะนำว่าควรจะซื้อสินค้านั้นๆ จากใคร และนอกจากนั้นแล้ว Firefly ยังไม่เหมาะกับสินค้าที่มีความซับซ้อน เช่น คอมพิวเตอร์ประกันภัย หรือว่า รถยนต์ เพราะว่าในสินค้าประเภทนี้

ผู้ใช้ต้องการให้สินค้านั้นๆ มีคุณสมบัติต่างๆตามที่ผู้ใช้ต้องการ เช่น ราคา ขนาด การรับประกัน น้ำหนัก เพราะว่า ACF ไม่ได้นำคุณสมบัติเหล่านั้นมาพิจารณา

2.1.4.6 BargainFinder

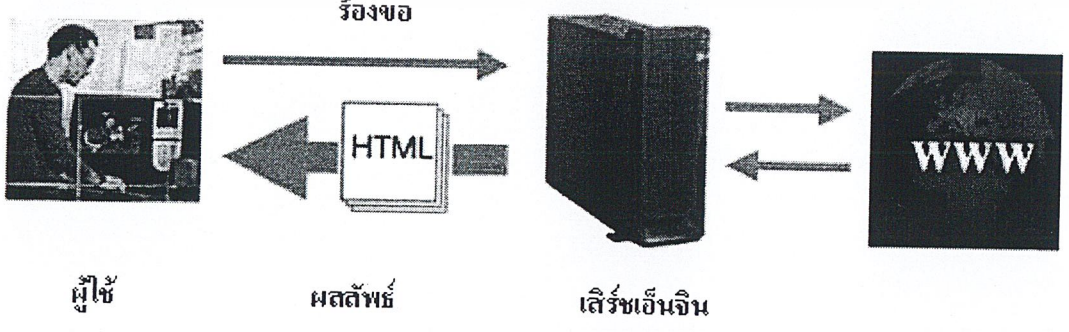
BargainFinder เป็นเอเจนต์ประเภท Merchant Brokering ตัวแรกที่มีการเปรียบเทียบราคา การทำงานก็คือ เมื่อผู้ใช้กำหนดรายละเอียดสินค้า BargainFinder จะเข้าไปหาราคาจากเว็บไซต์ของผู้ขาย แล้วแสดงผลออกมาเป็น ราคาสินค้าทั้งหมดที่พบออกมาให้ผู้ใช้ แต่ว่าข้อเสียของ BargainFinder ก็คือ BargainFinder จะทำการขอราคาของสินค้าจากเซิร์ฟเวอร์ของ BargainFinder ซึ่งทำให้ผู้ขายบางรายป้องกันการร้องขอราคาได้ เช่น Cdnow ไม่ยอมให้ราคาผ่านทาง BargainFinder โดยให้เหตุผลว่า Cdnow ให้สิ่งอื่นนอกเหนือจากการขาย CD ได้แก่ review ตัวอย่างเพลง หรือประวัตินักร้อง เป็นต้น

2.2 แนวคิดของการค้นหาข้อมูลจากแหล่งข้อมูล

การคัดเลือกข้อมูลออกจากแหล่งข้อมูลนั้นได้มีมานานพอสมควรแล้ว โดยแอพพลิเคชันที่เป็นที่นิยมมากก็คือ เสิร์ชเอ็นจินสำหรับการค้นหาเว็บเพจที่มีความเกี่ยวข้องกับสิ่งที่ผู้ใช้ต้องการ แต่ว่าการค้นหาเว็บเพจนั้นก็ยังไม่เพียงพอ เนื่องจากผู้ใช้ต้องเข้าไปอ่านทุกๆเว็บเพจเพื่อค้นหาสาระที่ตนต้องการ จึงได้มีแนวคิดใหม่ขึ้นมา โดยจะสามารถแบ่งแนวคิดทางด้านการคัดเลือกข้อมูลได้ ดังนี้

2.2.1 Information Retrieval

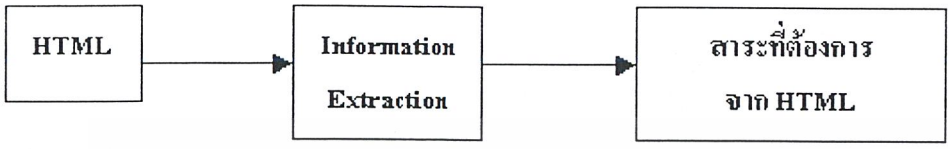
Information Retrieval (IR) เป็นกระบวนการค้นหาข้อมูลจากเซตของเอกสารที่มีอยู่ โดยผลลัพธ์ที่ได้จะเป็นเอกสารหรือเซ็ตของเอกสารที่มีความเกี่ยวข้องกับ Query โดยเมื่อได้ผลลัพธ์จาก IR แล้วผู้ใช้จะต้องกลั่นกรองและคัดเลือกสาระที่ต้องการด้วยตนเองอีกทีหนึ่ง ตัวอย่างเช่น ในระบบห้องสมุดดิจิตอล ผู้ใช้จะทำการป้อน Query ให้กับระบบแล้วระบบจะทำการค้นหาเอกสารที่เกี่ยวข้องกับ Query แล้วรีเทิร์นผลลัพธ์เป็นลิงก์ไปยังเอกสารนั้นๆ หลังจากนั้นผู้ใช้ก็จะต้องอ่านและคัดเลือกเอาเนื้อหาสาระที่ต้องการ



รูปภาพ 2-7 หนึ่งในตัวอย่างของ Information Retrieval คือ เสิร์ชเอ็นจิน

2.2.2 Information Extraction

Information Extraction (IE) เป็นกระบวนการคัดเลือกเอาสาระออกมาจากข้อมูลที่มี เช่น การคัดเลือกเอาสาระออกมาจากเว็บเพจที่มีข้อมูลของหุ้น โดยจะเป็นการคัดเอาเฉพาะส่วนที่สนใจ ซึ่งอาจจะเป็นราคาของหุ้นบางตัว เป็นต้น โดยองค์ประกอบหลักของ IE ก็คือ กฎในการดึงข้อมูลที่ต้องการออกมา



รูปภาพ 2-8 กระบวนการของ Information Extraction

แนวทางการอิมพลีเมนต์ Information Extraction

ในปัจจุบันมีการออกแบบระบบ IE อยู่ 2 แบบด้วยกันคือ

1. Knowledge Engineering Approach จะมีการสร้างกฎสำหรับการคัดเลือกสาระโดยการทำด้วยผู้เชี่ยวชาญของแอพพลิเคชันนั้นๆ ในวิธีนี้ความชำนาญของผู้พัฒนาระบบเป็นสิ่งสำคัญมาก แต่โดยมากแล้วระบบ IE ที่ทำงานได้อย่างมีประสิทธิภาพมักจะทำด้วยมือ ปัญหาที่สำคัญของวิธีนี้ก็คือจะต้องใช้ผู้เชี่ยวชาญในการพัฒนาระบบ
2. Automatic Training Approach สำหรับวิธีนี้แล้วไม่จำเป็นต้องใช้ความชำนานมากนักเมื่อมีการเปลี่ยนโดเมนของข้อมูล เพียงแต่ป้อนความรู้ให้กับระบบแล้วระบบจะสามารถเรียนรู้เองได้ วิธีนี้จะต้องใช้ความรู้ทางปัญญาประดิษฐ์ (Artificial Intelligent) เป็นอย่างมาก

2.2.3 Information Retrieval กับ Information Extraction

IR เป็นเทคโนโลยีที่มีการค้นคว้ากันมานานกว่า IE และมีการใช้กันอย่างแพร่หลายในปัจจุบัน โดยเฉพาะการทำเสิร์ชเอ็นจิน หรือห้องสมุดดิจิตอล อาจสรุปได้ว่า IR คือ การค้นหาเอกสารที่เกี่ยวข้องออกมาจากเซตของเอกสารที่มี ในขณะที่ IE คัดเอาสาระออกมาจากเอกสาร ดังนั้นจะเห็นได้ว่าทั้ง IR และ IE มีจุดประสงค์ที่แตกต่างกัน แต่ถ้านำมาใช้ร่วมกันก็จะเป็นระบบที่มีความสามารถมาก

2.3 ชนิดของแหล่งข้อมูล

แหล่งข้อมูลที่นำมาทำกระบวนการ Information Extraction นั้น สามารถจัดประเภทได้ตามโครงสร้างของข้อมูลดังนี้

2.3.1 ข้อมูลที่มีโครงสร้าง หรือ Structured Data

ข้อมูลชนิดนี้เป็นข้อมูลที่เหมาะที่สุดในการทำ Information Extraction เพราะว่ามีโครงสร้างที่ตายตัวทำให้ผู้พัฒนาโปรแกรมพัฒนาได้อย่างง่ายและเป็นระบบ ตัวอย่างของข้อมูลชนิดนี้ได้แก่ ข้อมูลบน DMBS ที่มีการกำหนด Schema จากการออกแบบฐานข้อมูล หรือ ข้อมูล XML ที่มีการกำหนดโครงสร้างโดย Document Type Difinition หรือ Schema ก็ตาม

```
<?xml version= "1.0"?>
<property>
  <condominium>
    <name> Sathon Place </name>
    <address> Sathon Rd. </address>
    <price>  15,000,000 Bht </price>
  </condominium>
</property>
```

รูปภาพ 2-9 ตัวอย่างของ XML ซึ่งจัดว่าเป็น Structured Data

2.3.2 ข้อมูลที่ไร้โครงสร้าง หรือ Free Text

ข้อมูลชนิดนี้เป็นภาษาธรรมชาติจึงจัดว่าเป็นข้อมูลที่ไม่มีโครงสร้าง ตัวอย่างเช่น ข่าวซึ่งไม่มีโครงสร้างที่แน่นอนว่าลำดับของแอตทริบิวต์มีการเรียงตัวอย่างไร หรือแม้แต่จำนวนของแอตทริบิวต์ที่ไม่สามารถคาดเดาได้ ด้วยเหตุนี้ข้อมูลชนิดนี้จึงไม่นิยมนำมาทำกระบวนการ Information Extraction

2.3.3 ข้อมูลแบบกึ่งโครงสร้าง หรือ Semi-Structured Data

เป็นข้อมูลชนิดที่อยู่กึ่งกลางระหว่างข้อมูลที่ไม่มีโครงสร้างและข้อมูลที่มีโครงสร้าง โดยในอดีตข้อมูลชนิดนี้ IE ไม่สามารถเข้าใจ จึงได้มีการพัฒนาเทคนิคทาง Natural Language Processing (NLP) เพื่อการสร้างกฎในการคัดเลือกข้อมูลออกมาจากข้อมูลที่ไร้โครงสร้าง อย่างไรก็ตามวิธีเหล่านี้เหมาะกับเอกสารที่มีโครงสร้างเท่านั้น จะไม่เหมาะกับข้อมูลแบบกึ่งโครงสร้าง ดั้งนั้นเทคนิคเดิมของ IE ไม่สามารถนำมาใช้ได้

```
<html>
  .
  .
    <table>
     <tr>
       <td><font color= “red”>Condominium</font></td>
       <td><b>Sathon Place</b></td>
       <td><<font color= “blue”><b>15,000,000</b></font></td>
     </tr>
    </table>
  .
  .
</html>
```

รูปภาพ 2-10 ตัวอย่างของ HTML ที่จัดว่าเป็นเอกสารแบบ Semi-Structured Data

2.3.4 ข้อมูลบนอินเทอร์เน็ต

อินเทอร์เน็ตเป็นแหล่งข้อมูลขนาดใหญ่ที่โดยมากข้อมูลมักจะเป็นประเภทกึ่งโครงสร้าง ถึงแม้ว่าอาจจะมีข้อมูลที่มีโครงสร้างหรือข้อมูลที่ไม่มีโครงสร้างบ้าง นักวิจัยบางคนได้กล่าวว่าทุกเว็บเพจเป็นข้อมูลแบบ กึ่งโครงสร้างเพราะว่ามีข้อมูลเกี่ยวกับการแสดงผล อย่างไรก็ตามได้มีผู้เชี่ยวชาญจัดประเภทของเว็บเพจดังนี้

- เว็บเพจที่มีโครงสร้าง (Structured Webpage) คือ เว็บเพจที่มีการทำข้อมูลเป็นชิ้นถือว่าเป็นข้อมูลชนิดที่มีโครงสร้างถ้าทุกๆแอตทริบิวต์ใน Tuple สามารถถูกคัดเลือกออกมาได้อย่างถูกต้องโดยใช้แนวทางบางอย่างเช่น การเว้นวรรค หรือ การเรียงของแอตทริบิวต์
- เว็บเพจแบบกึ่งโครงสร้าง (Semistructured Webpage) คือ อาจจะมีบาง tuple ที่มีแอตทริบิวต์ไม่ครบ แอตทริบิวต์ที่มีหลายค่า หรือ การสับเปลี่ยนของแอตทริบิวต์
- เว็บเพจที่ไม่มีโครงสร้าง (Unstructured Webpage) คือ จะต้องมีการใช้ความรู้ทางด้านภาษามาใช้ในการคัดเลือกข้อมูลออกมาจากเว็บเพจ

2.4 สไปเดอร์

ถ้าคุณต้องการให้เว็บไซต์ของคุณเป็นที่รู้จัก ไม่มีอะไรสำคัญไปกว่าการที่เว็บไซต์ของคุณมีรายชื่ออยู่ในเสิร์ชเอ็นจิน เป็นอันดับต้นๆ เมื่อมีผู้สนใจในสิ่งที่เกี่ยวข้องกับเว็บไซต์ของคุณ ซึ่งเสิร์ชเอ็นจิน จะเป็นเครื่องมือที่ช่วยในการค้นหาเว็บไซต์ของคุณ

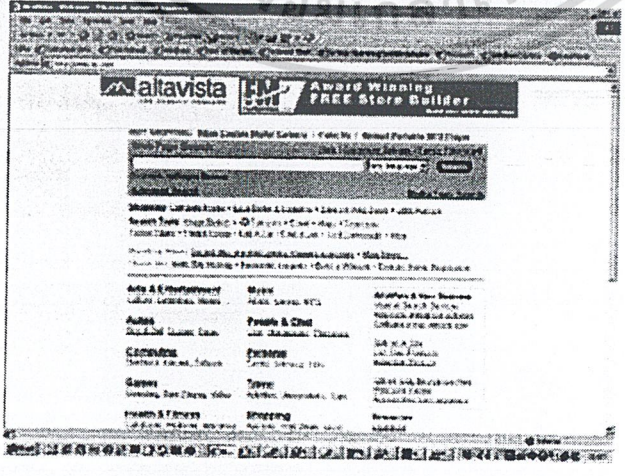
การที่เว็บไซต์ของคุณผลที่ได้จากการค้นหาก็ไม่สามารถรับประกันได้ว่าเว็บไซต์ของคุณจะมีคนเยี่ยมชมมากขึ้น โดยปกติทั่วไปแล้วผู้ใช้ที่เข้ามาค้นหาเว็บไซต์ที่ต้องการนั้น ถ้าไม่พบเว็บไซต์ที่ต้องการในผลหน้าแล้วก็จะไม่เข้าไปในผลหน้าต่อไป แต่จะทำการค้นหาด้วยคีย์เวิร์ดใหม่ ดังนั้นจึงจำเป็นมากที่เว็บไซต์ของคุณต้องมีรายชื่ออยู่ในหน้าแรกๆของผลที่ได้จากการค้นหาของผู้ใช้เสิร์ชเอ็นจิน นี้

ไม่มีสูตรใดที่สามารถคำนวณได้อย่างถูกต้องว่าเว็บไซต์ใดควรจะอยู่ในอันดับแรกๆ ของผลลัพธ์ที่ได้เสิร์ชเอ็นจิน แต่ละตัวมีความแตกต่างกัน สามารถทำงานได้ดีในสัปดาห์นี้ แต่อาจจะทำงานได้ไม่ดีในอีกสัปดาห์ก็เป็นได้ มีเว็บไซต์ใหม่ที่เข้ามาใหม่และตั้งค่าที่สามารถทำให้เว็บไซต์นั้นมีรายชื่ออยู่ในผลลัพธ์หน้าแรกๆ ได้เหมาะสมกว่า โดยเรียกค่าเหล่านั้นว่า Spider food อาจทำให้เว็บไซต์ของคุณตกอันดับได้ โดยทั่วไปแล้วผู้ที่สร้างเสิร์ชเอ็นจิน ขึ้นมานั้นไม่ได้เปิดเผยว่าความจริงแล้วมันทำงานอย่างไร ให้รู้เพียงข้อมูลทั่วไปเท่านั้นว่าทำการจัดลำดับอย่างไรเท่านั้น

2.5 เสิร์ชเอ็นจิน

เป็นฐานข้อมูลที่ถูกเก็บโดยสไปเดอร์หรือโรบอตซึ่งจะทำหน้าที่แสกนลิงก์ต่างแล้วทำการวิเคราะห์แต่ละหน้า และทำการเก็บข้อมูลลงในฐานข้อมูลนั้น

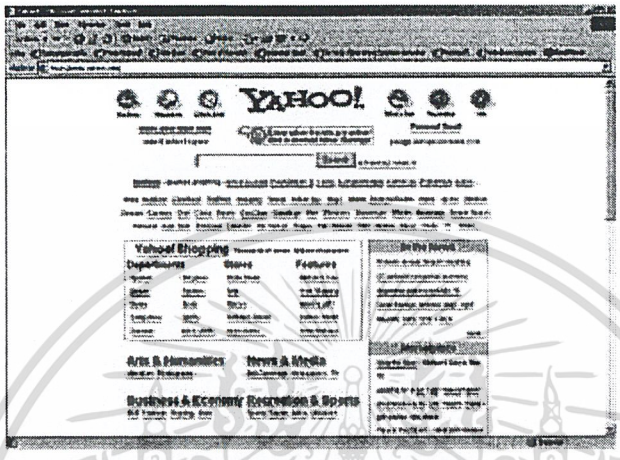
สไปเดอร์ที่ใช้กับเสิร์ชเอ็นจินนั้นสามารถเข้าถึงหน้า HTML ถึง 100,000 หน้าต่อวัน สไปเดอร์ส่วนมากจะสามารถเข้าถึงหน้า HTML ต่างๆ ได้โดยผ่านลิงก์ ที่อยู่ใน HTML หน้านั้นแล้วทำแบบเดียวต่อกันไปเรื่อยๆ มีความเป็นไปได้ที่สไปเดอร์สามารถหาเว็บไซต์ของคุณได้โดยที่คุณไม่ต้องส่ง URL ของคุณเข้าไปกับเสิร์ชเอ็นจิน ตัวนั้น ตัวอย่างของเสิร์ชเอ็นจิน ได้แก่ www.altavista.com หรือ www.hotbot.com



รูปภาพ 2-11 ตัวอย่างของเสิร์ชเอ็นจิน www.altavista.com

2.6 ไดเร็กทอรี

เป็นฐานข้อมูลที่ไม่ได้ถูกเก็บโดยสไปเดอร์แต่ใช้คนเพื่อแยกประเภทของเว็บเพจแต่ละหน้าที่ได้ทำการ Submit ไว้แล้วในไดเร็กทอรีที่เลือกไว้ และไม่สามารถเปลี่ยนได้ ซึ่งทั้งเสิร์ชเอ็นจิน และไดเร็กทอรี จะมีฟังก์ชันการทำงานที่เหมือนกัน ตัวอย่างของไดเร็กทอรี ได้แก่ www.yahoo.com หรือ www.sanook.com เป็นต้น



รูปภาพ 2-12 เว็บไดเร็กทอรี่ www.yahoo.com

2.7 Wrapper

Wrapper อาจจะถูกมองว่าเป็นกระบวนการที่ออกแบบสำหรับการคัดเลือกสาระจากแหล่งข้อมูลแล้วนำเสนอข้อมูลที่สนใจให้ผู้ใช้ ในวงการดาตาเบส Wrapper เป็นส่วนประกอบของซอฟต์แวร์ที่แปลงข้อมูล และ Query จากโมเดลหนึ่งไปยังอีกโมเดลหนึ่ง แต่สำหรับเว็บแล้ว หน้าที่ของ Wrapper คือ การแปลงข้อมูลที่เก็บอยู่ในเอกสาร HTML ให้เป็นข้อมูลที่มีโครงสร้างสำหรับการประมวลผลต่อไป

Wrapper สำหรับเว็บนั้นจะรับ Query ของข้อมูลในแหล่งข้อมูล แล้วดึงข้อมูลจากแหล่งนั้น ทำการคัดเลือกสาระออกมาแล้วรีเทอร์นผลลัพธ์ โดยการใช้กฎในการคัดเลือกข้อมูล (Extraction Rule) และโค๊ดสำหรับการใช้กฎเหล่านี้ซึ่งจะเฉพาะเจาะจงสำหรับแหล่งข้อมูลหนึ่งเท่านั้น สำหรับการจะคัดเลือกสาระของออกจากข้อมูลจากหลายๆแหล่งจะทำโดยการสร้าง Wrapper ไว้หลายๆตัว

คุณสมบัติของ Wrapper ที่ดี

1. สามารถรับมือกับการขัดข้องของระบบเน็ตเวิร์กได้
2. สามารถรับมือกับการเปลี่ยนแปลงรูปแบบการแสดงผลของเอกสารได้

2.8 สถาปัตยกรรมเว็บแอพพลิเคชัน

แบ่งออกเป็น 3 แพตเทิร์นหลักๆ ได้แก่

1. *เว็บไคลเอ็นต์แบบ Thin (Thin Web Client)* เป็นแพตเทิร์นที่นิยมที่สุด โดยในแพตเทิร์นนี้จะมีการควบคุมการทำงานของไคลเอ็นต์น้อยที่สุด โดยมากผู้ใช้จำเป็นจะต้องมีเพียงเว็บบราวเซอร์ที่รองรับการใช้แบบฟอร์มได้ก็เพียงพอแล้วกล่าวคือ Client จะได้รับแต่ HTML เท่านั้น
2. *เว็บไคลเอ็นต์แบบ Thick (Thick Web Client)* เป็นแพตเทิร์นที่มีใช้มากพอสมควร โดย Client จะได้รับข้อมูลที่นอกเหนือจาก HTML ทั่วๆไป เช่น Dynamic HTML, Java Applet หรือ Active X control แต่อย่างไรก็ตามการสื่อสารกับเซิร์ฟเวอร์ ก็ยังทำโดย HTTP
3. *Web Delivery* นอกเหนือจากการสื่อสารผ่าน HTTP แล้วในแพตเทิร์นนี้ยังได้มีการใช้โปรโตคอลอื่นๆด้วย เช่น IIOP และ DCOM โดยใช้ในการกระจายออบเจ็กต์ ในแพตเทิร์นนี้เว็บบราวเซอร์ทำหน้าที่เป็นแค่ผู้รับส่งออบเจ็กต์เท่านั้น

2.9.1 เว็บไคลเอ็นต์แบบ Thin

เว็บไคลเอ็นต์แบบ Thin เป็นแพตเทิร์นของสถาปัตยากรรมที่เหมาะกับแอพพลิเคชันบนอินเทอร์เน็ต เพราะว่าผู้พัฒนาแอพพลิเคชันแทบจะไม่สามารถปรับเปลี่ยนอะไรที่ไคลเอ็นต์ได้เลย โดยการประมวลผลทั้งหมดจะกระทำที่เซิร์ฟเวอร์

2.9.1.1 การประยุกต์ใช้เว็บไคลเอ็นแบบ Thin

แพตเทิร์นนี้เหมาะกับอินเทอร์เน็ตแอพพลิเคชันที่ไม่สามารถควบคุมการทำงานของไคลเอ็นต์ได้ หรือว่าประสิทธิภาพในการประมวลผลของไคลเอ็นต์ต่ำ

อินเทอร์เน็ตแอพพลิเคชันส่วนใหญ่จะใช้แพตเทิร์นนี้เพราะไม่ต้องการจะจำกัดจำนวนผู้ใช้ด้วยความสามารถของไคลเอ็นต์

2.9.1.2 โครงสร้างของเว็บไคลเอ็นต์แบบ Thin

ส่วนประกอบหลักของสถาปัตยกรรมเว็บไคลเอ็นต์แบบ Thin โดยมากจะอยู่บนเซิร์ฟเวอร์ โดยส่วนประกอบหลักในแพตเทิร์นนี้ประกอบด้วย

- *เว็บบราวเซอร์ของไคลเอ็นต์:* เพียงแค่เว็บบราวเซอร์มีความสามารถในการรองรับ Form ของ HTML ก็ถือว่าเพียงพอแล้ว เพราะว่าเว็บบราวเซอร์มีหน้าที่เป็นเพียงแค่ส่วนติดต่อกับผู้ใช้เท่านั้น โดยอาจจะมีการรับและส่งคุ๊กกี้ (Cookies) ด้วย การติดต่อระหว่างผู้ใช้กับระบบทั้งหมดจะทำผ่านเว็บบราวเซอร์
- *เว็บเซิร์ฟเวอร์:* ไคลเอ็นต์จะเข้าถึงระบบโดยผ่านทางเว็บเซิร์ฟเวอร์เท่านั้นซึ่งก็คือการส่งคำร้องขอ (Request) เอกสาร HTML ไปยังเว็บเซิร์ฟเวอร์ ซึ่งอาจจะเป็นเว็บเพจที่เป็นแบบ Static หรือ Server Page ก็ได้ การตอบสนอง (Response) จากเว็บเซิร์ฟเวอร์จะขึ้นอยู่กับว่าทางเว็บบราวเซอร์ได้ร้องขออะไรมา ถ้าเป็นการร้องขอเพจที่มีสคริปต์ฝั่งเซิร์ฟเวอร์ฝังอยู่ด้วย เช่น CGI ISAPI หรือ NSAPI ทางเว็บเซิร์ฟเวอร์ก็จะสร้างโปรเซสของอินเทอร์พรีเตอร์

ขึ้นมาเพื่อทำการตีความหมายของสคริปต์แล้วจึงส่งผลลัพทธ์ที่ได้ไปยังเว็บบราวเซอร์ซึ่งจะเป็น HTML เท่านั้น

- *การสื่อสารด้วย:* HTTP ซึ่งเป็นโปรโตคอลที่มีการใช้ในการสื่อสารระหว่างเว็บบราวเซอร์กับเว็บเซิร์ฟเวอร์ โดยจะมีการสร้างช่องทางการสื่อสารใหม่ทุกๆครั้งที่มีการสื่อสารกันระหว่างเว็บบราวเซอร์กับเว็บเซิร์ฟเวอร์ และเพื่อความปลอดภัยของข้อมูลจึงมีอีกวิธีการในการสื่อสารผ่าน HTTP ก็คือ การสร้างช่องทางการสื่อสาร HTTP แบบปลอดภัยผ่าน Secure Sockets Layer (SSL) ซึ่งการสื่อสารชนิดนี้จะต้องมีการเข้ารหัสข้อมูลรับ-ส่งระหว่างเว็บบราวเซอร์กับเว็บเซิร์ฟเวอร์ โดยใช้เทคโนโลยีคีย์สาธารณะและคีย์ส่วนบุคคล
- *เอกสาร HTML:* คือ เว็บเพจที่ไม่ผ่านการประมวลผลที่เซิร์ฟเวอร์ โดยปกติแล้วเว็บเพจเหล่านี้จะประกอบด้วยคำบรรยาย หรือวิธีการใช้ หรือว่าแบบฟอร์ม HTML ที่ไม่มีการเปลี่ยนแปลง
- *เซิร์ฟเวอร์เพจ:* เป็นเว็บเพจที่ต้องผ่านการประมวลผลที่เซิร์ฟเวอร์ก่อนส่งไปให้ไคลเอ็นต์ โดยทั่วไปแล้วการอิมพลิเมนต์เซิร์ฟเวอร์เพจจะใช้สคริปต์เพจ เช่น Active Server Pages, Java Server Pages, Profession Home Page หรือ Cold Fusion Pages เป็นต้น ซึ่งโดยมากเซิร์ฟเวอร์เพจจะสามารถเข้าถึงทรัพยากรต่างๆบนฝั่งเซิร์ฟเวอร์ได้ เช่น ดาตาเบสเซิร์ฟเวอร์
- *แอพพลิเคชันเซิร์ฟเวอร์:* เป็นกลไกหลักในการประมวลผลเซิร์ฟเวอร์เพจ โดยอาจจะทำงานอยู่บนเครื่องเดียวกับเว็บเซิร์ฟเวอร์ก็ได้ หรือแม้แต่ทำงานอยู่ในโปรเซสเดียวกันก็ได้

2.9.2 เว็บไคลเอ็นต์แบบ Thick

เว็บไคลเอ็นต์แบบ Thick เป็นแพตเทิร์นของสถาปัตยกรรมที่เพิ่มเติมความสามารถของแพตเทิร์นสถาปัตยกรรมแบบเว็บไคลเอ็นต์แบบ Thin โดยการใช้สคริปต์ฝั่งไคลเอ็นต์และออบเจ็กต์ที่ฝั่งไคลเอ็นต์ เช่น Active X หรือ Java Applet เพื่อที่จะแบ่งการทำงานบางส่วนมาทำที่ไคลเอ็นต์และปรับปรุงส่วนติดต่อกับผู้ใช้ให้มีความสวยงามและความสามารถมากกว่า HTML ธรรมดา

2.9.2.1 การประยุกต์ใช้เว็บไคลเอ็นต์แบบ Thin

เว็บไคลเอ็นต์แบบ Thick เหมาะกับการใช้งานที่สามารถคาดเดาได้ว่าผู้ใช้มีเว็บบราวเซอร์รุ่นใด, ต้องการส่วนติดต่อกับผู้ใช้ที่ซับซ้อนขึ้น หรือ ต้องมีการประมวลผลที่ฝั่งไคลเอ็นต์ซึ่งเป็นสิ่งที่สำคัญที่สุดในการแยกแยะความแตกต่างระหว่างเว็บไคลเอ็นต์แบบ Thick ออกจากเว็บไคลเอ็นต์แบบ Thin

งานที่ถูกนำมาประมวลผลบนไคลเอ็นต์มักจะไม่ใช่งานที่ต้องการทรัพยากรสูงมากนัก เช่น การตรวจสอบความถูกต้องของแบบฟอร์ม หรือ การสลับรูปเมื่อมีการนำเมาส์ไปวางโดยการใช้จาวาสคริปต์

โดยส่วนมากแล้วการใช้สคริปต์ฝั่งไคลเอ็นต์, แอปเพลต, แอคทีฟเอ็กซ์ หรือปลั๊กอิน ทั้งหลายก็เพื่อเพิ่มประสิทธิภาพในการแสดงผล เช่น การใช้จาวาสคริปต์ในการตรวจสอบความถูกต้องของแบบฟอร์ม หรือใช้จาวาแอปเพลตในการเช็คผลกีฬาแบบออนไลน์ หรือการใช้ปลั๊กอิน Shockwave เพื่อแสดงอนิเมชันบนบราวเซอร์

2.9.2.2 โครงสร้างของเว็บไคลเอ็นต์แบบ Thick

การติดต่อระหว่างเซิร์ฟเวอร์และไคลเอ็นต์จะทำบน HTTP เช่นเดียวกับเว็บไคลเอ็นต์แบบ Thin โดยองค์ประกอบหลักได้เพิ่มจากเว็บไคลเอ็นต์แบบ Thin ดังนี้

- *ไคลเอ็นต์สคริปต์:* ได้แก่ จาวาสคริปต์ หรือ VBScript ที่อยู่ใน HTML ซึ่ง W3C ได้กำหนดอินเทอร์เฟสระหว่าง HTML และ Document Object Model สำหรับไคลเอ็นต์สคริปต์
- *XML:* คือ เอกสาร ".XML" ที่นำเสนอข้อมูลโดยที่ไม่มีการกำหนดการแสดงผล
- *ActiveX Control:* ไคลเอ็นต์สคริปต์สามารถอ้างถึงและดาว์นโหลดออบเจ็กต์ COM มาที่ไคลเอ็นต์ได้ถ้าจำเป็น โดยออบเจ็กต์ COM จะสามารถเข้าถึงทรัพยากรของผู้ใช้ได้อย่างเต็มที่ กลไกในการปกป้องผู้ใช้ก็คือ จะมีการพิสูจน์ตัวตน (Authenticate) ผ่านทางบุคคลที่ 3 ที่ได้รับความไว้วางใจ (Trusted Third Party)
- *จาวาแอปเพล็ต:* คือ คอมโพเนนต์ที่ถูกคอมไพล์แล้วซึ่งจะทำงานภายใต้เว็บบราวเซอร์ โดยมีการจำกัดการเข้าถึงทรัพยากรของระบบเพื่อความปลอดภัย จุดประสงค์ของการใช้จาวาแอปเพล็ตก็เพื่อการสร้างส่วนอินเตอร์เฟสให้สวยงาม, การอ่านข้อมูล XML หรือซ่อนรายละเอียดการทำงานที่อยู่บนไคลเอ็นต์
- *จาวาบีน:* เป็นจาวาคอมโพเนนต์ที่มีจุดประสงค์ในการทำงานเฉพาะอย่าง โดยมีการอิมพลีเมนต์อินเทอร์เฟสต่างๆเพื่อให้ง่ายต่อการติดต่อกับระบบที่ใหญ่และซับซ้อน

2.9.3 Web Delivery

แพตเทิร์นนี้ได้รับชื่อนี้ก็เพราะว่าใช้เครือข่ายอินเทอร์เน็ตสำหรับส่งออบเจ็กต์ไปที่ไคลเอ็นต์ เพื่อกระจายออบเจ็กต์ไปทำงานแบบไคลเอ็นต์-เซิร์ฟเวอร์

2.9.3.1 การประยุกต์ใช้ Web Delivery

แพตเทิร์นนี้เหมาะสำหรับการประยุกต์ใช้เมื่อสามารถควบคุมไคลเอ็นต์และเน็ตเวิร์กได้ แต่แพตเทิร์นนี้ไม่เหมาะกับเว็บแอพพลิเคชันเพราะว่าไม่สามารถควบคุมไคลเอ็นต์ได้และไม่สามารถคาดเดาสถานะของเน็ตเวิร์กได้

จุดเด่นของสถาปัตยกรรมนี้ก็คือสามารถกระจายการประมวลผลไปทำงานบนไคลเอ็นต์ได้ แต่ว่าความเป็นไปได้ที่จะประยุกต์ใช้เพียงแค่แพตเทิร์นนี้เพียงแพตเทิร์นเดียวนั้นเป็นไปได้ยาก ทางที่เป็นไปได้คือ จะต้องใช้ร่วมกับแพตเทิร์นอื่น

2.9.3.2 โครงสร้างของ Web Delivery

ข้อแตกต่างระหว่าง Web Deliver กับ สถาปัตยกรรมอื่นๆสำหรับเว็บแอพพลิเคชันก็คือวิธีการสื่อสารระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ โดยในแพตเทิร์นอื่นๆนั้นจะใช้โปรโตคอล HTTP ซึ่งเป็นโปรโตคอล

ชนิด Connectionless ที่เป็นข้อจำกัดสำหรับการโต้ตอบระหว่างผู้ใช้กับเซิร์ฟเวอร์ โดยองค์ประกอบของสถาปัตยกรรมนี้เพิ่มเติมจากเว็บไคลเอ็นต์แบบ Thin ดังนี้

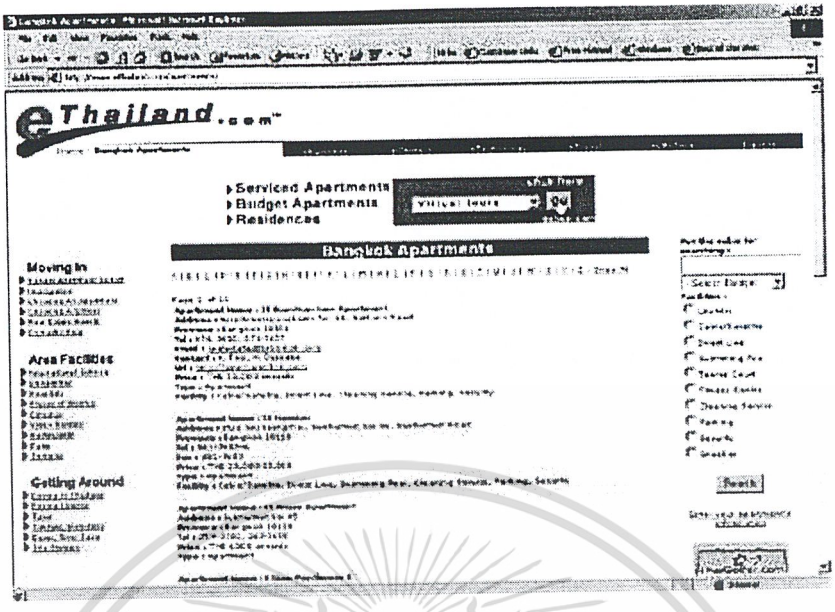
- *DCOM:* หรือ Distributred COM ซึ่งเป็นโปรโตคอลสำหรับการกระจายออบเจ็กต์ของบริษัทไมโครซอฟท์ โดยจะยอมให้ออบเจ็กต์บนเครื่องหนึ่งสามารถติดต่อและเรียกใช้เมธอดของออบเจ็กต์บนเครื่องอื่นได้
- *IIOP:* หรือ Internet Inter-ORB คือ โปรโตคอล CORBA สำหรับการโต้ตอบระหว่างออบเจ็กต์ผ่านทางอินเทอร์เน็ตหรือกเน็ตเวิร์กแบบ TCP/IP
- *RMI (JRMP):* หรือ Remote Method Invocation คือ การโต้ตอบระหว่างออบเจ็กต์ที่อยู่ต่างเครื่องกันบนภาษจาวา ถึงแม้ว่า JRMP (Java Remothod Method Protocol) เป็นโปรโตคอลมาตรฐานของ RMI แต่ว่า RMI ก็ยังสามารถอิมพลีเมนต์ด้วย IIOP ของ CORBA ได้เช่นกัน

2.9 เว็บไซต์อสังหาริมทรัพย์ของไทย

โครงงานนี้จะยึดเว็บไซต์ของไทยเป็นหลัก เนื่องจากต้องการจำกัดข้อมูลของคอนโดมีเนียมและอพาร์ทเมนต์ให้อยู่ในกรุงเทพ ซึ่งมีเว็บไซต์ต่างๆดังนี้

- www.ethailand.com/apartments

  เป็นเว็บไซต์ท่าที่มีการรวบรวมอพาร์ทเมนต์ให้เช่าในกรุงเทพไว้ โดยจะสามารถค้นหาตามชื่อ ราคา หรือ สิ่งอำนวยความสะดวก โดยข้อมูลที่ได้จากเว็บไซต์นี้จะมี
  - ชื่อของอพาร์ทเมนต์
  - ที่อยู่
  - จังหวัด
  - URL สำหรับรายละเอียด
  - ราคาเช่า

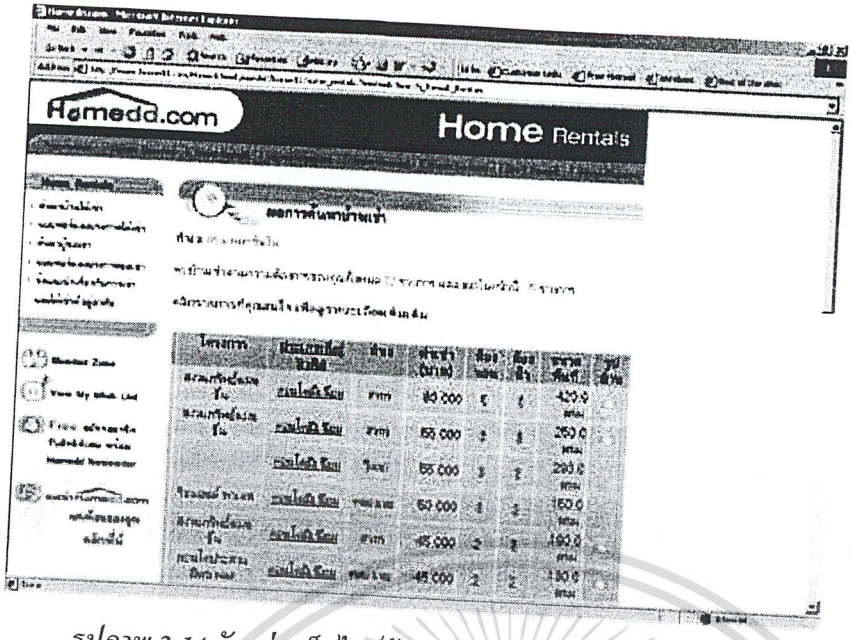


รูปภาพ 2-13 ตัวอย่างเว็บไซต์ค้นหาอพาร์ทเมนต์ (www.ethailand.com)

- www.homedd.com

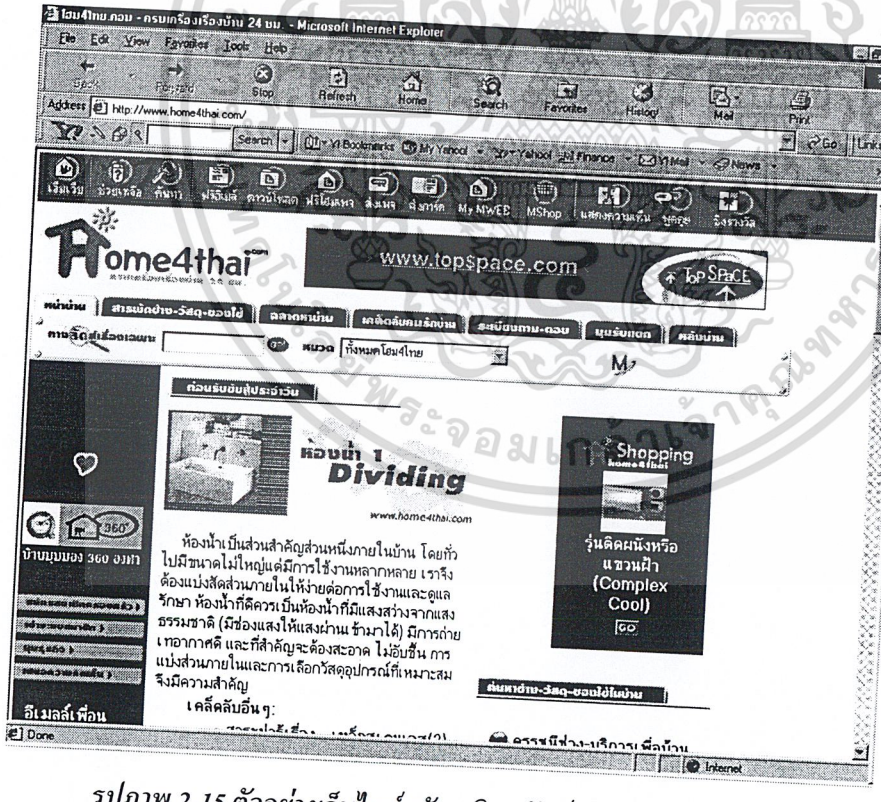
เป็นเว็บไซต์เกี่ยวกับบ้านที่มีบริการค้นหาสินทรัพย์ให้เช่าอีกแห่งที่มีข้อมูลของการเช่าอพาร์ทเมนต์ โดยจะมีข้อมูลสำหรับผู้ใช้ ดังนี้

- ชื่อโครงการ
- ประเภทที่อยู่อาศัย
- ทำเล
- ค่าเช่า
- จำนวนห้องนอน
- จำนวนห้องน้ำ
- ขนาดพื้นที่



รูปภาพ 2-14 ตัวอย่างเว็บไซต์ค้นหาอพาร์ทเมนต์ (www.homedd.com)

- www.home4thai.com



รูปภาพ 2-15 ตัวอย่างเว็บไซต์อสังหาริมทรัพย์ www.home4thai.com

เป็นเว็บไซต์เกี่ยวกับบ้านและอสังหาริมทรัพย์ ที่มีบริการค้นหาทรัพย์สินต่างๆ เช่น อพาร์ทเมนต์ คอนโดมิเนียม บ้านเดี่ยว ที่ดินเปล่า ฯลฯ โดยผู้ใช้สามารถกำหนดรายละเอียดในการค้นหาได้ดังนี้

- ชนิดของอสังหาริมทรัพย์
- การเข้าอยู่
- ทำเล
- งบประมาณ
- รูปภาพ
- แผนที่
- มุมมอง 360 องศา

บทที่ 3 XML

3.1 ประวัติความเป็นมาของ XML

XML ย่อมาจาก eXtensible Markup Language เป็นภาษา Markup ที่กำหนดมาตรฐานโดย W3C (World Wide Web Consotrium) ซึ่งมีรากฐานมาจาก SGML (Standard Generalized Markup Language) โดยมีจุดประสงค์ในการแก้ไขข้อบกพร่องของ HTML (Hyper Text Markup Language) ที่ไม่มีโครงสร้างของข้อมูลเนื่องจากมีแต่รายละเอียดในการแสดงผล และลดความยุ่งยากในของ SGML โดยจุดเด่นของ XML ก็คือ สามารถกำหนดแท็ก ได้เอง ซึ่งมีประโยชน์อย่างมากในการกำหนดโครงสร้างของข้อมูลเองโดยไม่ต้องรอให้มีการกำหนดมาตรฐานสากล แต่ใช่ว่าการที่สามารถกำหนดโครงสร้างข้อมูลได้เองจะทำได้อย่างไม่มีข้อจำกัดเพื่อเป็นการแก้ปัญหาความเป็นไม่เป็นมาตรฐานของโครงสร้างที่กำหนดขึ้นสำหรับแต่ละแอพพลิเคชัน จึงมีการกำหนดโครงสร้างของข้อมูลโดย DTD (Document Type Definition) หรือ Schema สำหรับแต่ละแอพพลิเคชัน

3.2 ส่วนประกอบของ XML

ส่วนประกอบของ XML มี 4 ส่วนหลักๆ ได้แก่

1. ตัวเอกสาร XML เอง
2. ส่วนกำหนดโครงสร้างของเอกสาร โดยในขณะนี้มี 2 แนวทางในการกำหนดโครงสร้าง ได้แก่ DTD (Document Type Definition) และ Schema
3. Stylesheet ซึ่งใช้ในการแสดงผล โดยแบ่งออกเป็น 2 ชนิด ได้แก่ XSL (eXtensible Style Language) และ CSS (Cascading Style Sheet)
4. ส่วนการลิงก์ข้อมูล ได้แก่ XLL (eXtensible Linking Language) ซึ่งประกอบด้วย Xlink และ Xpointer โดย Xlink จะเป็นการลิงก์ไปยังเอกสาร ในขณะที่ Xpointer เป็นการชี้ตำแหน่งของข้อมูลในเอกสาร

3.3 ไวยกรณ์ของ XML

เอกสาร XML ที่ถูกต้องตาม Syntax จะต้องมีคุณสมบัติดังนี้

1. จะต้องมี XML processing instruction ที่หัวไฟล์เพื่อเป็นการประกาศว่าเป็นเอกสาร XML คือ <? xml version= “1.0”?>
2. ในเอกสาร XML จะต้องมีแท็กที่เป็นราก 1 แท็ก ซึ่งคุณสมบัติของแท็กรากก็คือ จะต้องครอบคลุมทุกๆ แท็กในเอกสาร XML
3. ทุกๆแท็กจะต้องมีการปิดแท็ก โดยทำได้ 2 วิธีคือ <tag_name>content</tag_name> หรือ ถ้าเป็นแท็กว่าง(แท็กที่ไม่มีข้อมูล แต่อาจจะมีแอตทริบิวต์ได้) จะเป็น <empty_tag/>

4. จะต้องใช้เครื่องหมาย “ ” ครอบคลุมค่าของแอตทริบิวต์

ตัวอย่าง XML ที่ถูกต้องตาม Syntax

```
<?xml version="1.0"?>
<root_element>
    <child>
        <empty _element child_count= "0"/>
    </child>
</root_element>
```

3.4 DTD (Document Type Definition)

เนื่องจาก XML เป็นภาษาที่ผู้ใช้สามารถกำหนดโครงสร้างได้เอง การที่จะทำให้ภาษานั้นเป็นที่ยอมรับในกลุ่มของผู้ใช้และผู้พัฒนาในวงการใดๆนั้น จะต้องมีการกำหนดโครงสร้างของข้อมูลเพื่อความสะดวกในการสร้างและนำข้อมูล XML ไปใช้

DTD เป็นวิธีการกำหนดโครงสร้างข้อมูลของ XML ที่รวมอยู่ใน XML รุ่นที่ 1 โดย DTD สามารถกำหนดว่าให้ข้อมูลสามารถมีอิลิเมนต์ใดได้บ้าง ข้อมูลในอิลิเมนต์จะเป็นอะไรได้บ้าง สามารถมีอิลิเมนต์ลูกอะไรได้บ้าง มีจำนวนเท่าใด เรียงลำดับอย่างไร สามารถมีแอตทริบิวต์อะไรได้บ้าง

การจะตรวจสอบความถูกต้องทางโครงสร้างของ XML นั้น สามารถทำได้โดยการใช้ Parser ซึ่งจะแบ่งระดับความถูกต้องของเอกสาร XML ได้ดังนี้

3.5 ระดับความถูกต้องของเอกสาร XML

เอกสาร XML จะแบ่งความถูกต้องออกเป็น 3 ระดับ ได้แก่

1. Invalid คือ ไม่ถูกต้องตาม Syntax ของ XML เช่น <first><second>content(/second></first>
2. Well Formed คือ ถูกต้องตาม Syntax ของ XMLแต่มีโครงสร้างที่ไม่เป็นไปตาม DTD หรือ Schema
3. Valid คือ นอกจากถูกต้องตาม Syntax ของ XML แล้วยังมีโครงสร้างเป็นไปตาม DTD หรือ Schema

3.6 การกำหนดโครงสร้างข้อมูลโดย DTD

ในขั้นตอนการกำหนดโครงสร้างข้อมูลโดย DTD จะแบ่งออกเป็น 2 ขั้น คือ

1. การประกาศอิลิเมนต์ (Element)
2. การประกาศแอตทริบิวต์ (Attribute)

3.6.1 การประกาศอิลิเมนต์

การประกาศอิลิเมนต์มี Syntax ดังนี้

<!ELEMENT [Element Name] [Element Definition/Type]>

[Element Name] คือ ชื่อของอิลิเมนต์ที่อยู่ใน XML ถ้าหากมีการใช้ Namespace ก็ต้องใส่ namespace prefix ด้วยเพราะว่า DTD ตีความหมายอิลิเมนต์ที่มี namespace prefix กับอิลิเมนต์ที่ไม่มี namespace prefix เป็นคนละอิลิเมนต์กัน

[Element Type] เป็นส่วนที่ใช้ประกาศว่าอิลิเมนต์เป็นชนิดใด ซึ่งมี 2 ชนิด ได้แก่

- ANY คือ อิลิเมนต์นี้สามารถมีข้อมูลเป็นอะไรก็ได้
- EMPTY คือ อิลิเมนต์ที่ไม่มีข้อมูลและไม่มีอิลิเมนต์ลูก แต่จะมีแอตทริบิวต์ได้
- #PCDATA คือ ข้อมูลที่อยู่ในอิลิเมนต์จะมีอิลิเมนต์ลูกไม่ได้ จะต้องเป็นตัวอักษรเท่านั้น

ตัวอย่างของ Element Type

<!ELEMENT ROOT ANY>

<!ELEMENT FIRSTCHILD EMPTY>

<!ELEMENT SECONDCHILD (#PCDATA)>

[Element Definition] เป็นส่วนที่ใช้ประกาศว่าจะมีอิลิเมนต์ลูกอะไรบ้าง จำนวนเท่าใด และมีการเรียงลำดับอย่างไร

- การประกาศอิลิเมนต์ลูกมี Syntax ดังนี้ :

  <!ELEMENT [Element Name] ([Nested Element] [,Nested Element]…)>

  เช่น <!ELEMENT ROOT (FIRSTCHILD,SECONDCHILD)>

  จะเป็นการกำหนดว่าอิลิเมนต์ที่ชื่อว่า ROOT จะมีอิลิเมนต์ลูก 2 อิลิเมนต์ได้แก่ FIRSTCHILD และ SECONDCHILD ตามลำดับ

- การประกาศจำนวนของอิลิเมนต์ลูกมี 3 ประเภท ได้แก่

  1. มีอิลิเมนต์ลูกจำนวน 1 อิลิเมนต์หรือมากกว่า (One or More Childern) ประกาศโดยการใช้เครื่องหมาย “+” ต่อท้ายชื่ออิลิเมนต์ เช่น

  <!ELEMENT ROOT (FIRST+,SECOND)>

  2. มีอิลิเมนต์ลูกจำนวน 0 หรือ 1 อิลิเมนต์ (Zero or One Child) ประกาศโดยการใช้เครื่องหมาย “?” ต่อท้ายชื่ออิลิเมนต์ เช่น

  <!ELEMENT ROOT (FIRST?,SECOND)>

3. มีอิลิเมนต์ลูกจำนวนเท่าใดก็ได้ (Zero or More Children) ประกาศโดยการใช้เครื่องหมาย "*" ต่อท้ายชื่ออิลิเมนต์ เช่น

<!ELEMENT ROOT (FIRST*,SECOND)>

- การประกาศอิลิเมนต์แบบผสม คือ การประกาศที่อิลิเมนต์จะมีอิลิเมนต์ลูกได้หลายแบบ โดยการใช้เครื่องหมาย "|" ที่หมายความว่า "หรือ" เช่น

<!ELEMENT ROOT (FIRST|SECOND|THIRD)>

มีความหมายว่า ROOT จะมีอิลิเมนต์ลูกเป็น FIRST หรือ SECOND หรือ THIRD ก็ได้

3.6.2 การประกาศแอตทริบิวต์

การประกาศแอตทริบิวต์มี Syntax ดังนี้

<!ATTLIST [Enclosing Element]
[Attribute Name] CDATA [Modifier]
.....
>

[Enclosing Element] คือ ชื่อของอิลิเมนต์ที่เป็นเจ้าของแอตทริบิวต์

CDATA คือการกำหนดว่าค่าของแอตทริบิวต์จะต้องเป็นตัวอักษร

[Modifier] คือ การตั้งข้อกำหนดของแอตทริบิวต์ ได้แก่

- #IMPLIED คือ จะมีแอตทริบิวต์นี้หรือไม่ก็ได้
- #FIXED คือ จะกำหนดค่าของแอตทริบิวต์นี้ ไม่ให้มีการแก้ไข
- #REQUIRED คือ การกำหนดว่าจะต้องมีแอตทริบิวต์นี้

ตัวอย่างการประกาศแอตทริบิวต์

<!ATTLIST AUTHOR NAME CDATA #REQUIRED>
<!ATTLIST AUTHOR PHONE CDATA #IMPLIED>
<!ATTLIST AUTHOR COMPANY CDATA #FIXED "ABC">

3.7 XML Schema

XML Schema ถูกออกแบบขึ้นมาเพื่อแทนที่และเพิ่มความสามารถของ DTD โดย XML Schema เป็น XML สำหรับการกำหนดโครงสร้างของ XML เอง เนื่องจาก XML Schema ยังอยู่ระหว่างการปรับปรุงและเปลี่ยนแปลงทำให้ยังไม่เป็นมาตรฐานดังนั้นจะขอไม่กล่าวถึงรายละเอียดในการกำหนดโครงสร้างด้วย XMLSchema แต่จะกล่าวถึงความสามารถที่ XML Schema เหนือกว่า DTD ดังนี้

1. DTD ไม่รู้จักโครงสร้างที่เป็นแบบลำดับขั้น (Hierachy)
2. DTD ไม่สามารถจัดการกับ Namespace ได้
3. ไม่สามารถกำหนดความสัมพันธ์ระหว่างเอกสาร XML
4. สามารถกำหนดชนิดของข้อมูล (Data type) ที่ใช้ใน XML ได้ ดังนี้
   - string
   - boolean
   - float
   - double
   - decimal
   - timeInstant
   - timeDuration
   - recurringInstant
   - binary
   - uri

3.8 เปรียบเทียบระหว่าง DTD กับ Schema

DTD นั้นง่ายและไม่มีการเปลี่ยนแปลงแล้ว ในขณะที่ XML Schema นั้นจะมีความสามารถในการกำหนดชนิดของข้อมูล (Data Type) ได้ แต่ยังมีการเปลี่ยนแปลงอยู่ทำให้เป็นอุปสรรคในการใช้ Schema อยู่บ้าง

3.9 เทคโนโลยีที่เกี่ยวข้องกับ XML

3.9.1 Stylesheet

แบ่งออกเป็น 2 ชนิด ได้แก่ XSL (eXtensible Style Language) และ CSS (Cascading Style Sheet) โดยที่ XSL ยังแบ่งย่อยออกเป็น XSL Transformation และ XSL Formattion Object ในขณะที่ CSS ก็แบ่งออกเป็น 2 ระดับ จะไม่ขอกล่าวถึงรายละเอียดในที่นี้เนื่องจากโครงงานนี้ไม่ได้ใช้ XML ในการแสดงผลกับบราวเซอร์จึงไม่จำเป็นที่จะต้องมี Stylesheet

3.9.2 Namespace

Namespace เป็นการแมประหว่าง prefix ของอิลิเมนต์กับ URI (Uniform Resource Indentifier) โดยจุดประสงค์ของการแมปก็คือป้องกันการซ้ำในชื่อของอิลิเมนต์ ดังตัวอย่างต่อไปนี้ ถ้าหากมีเอกสาร XML ที่อิลิเมนต์ <price> สำหรับเก้าอี้ โดยอยู่ระหว่างอิลิเมนต์ <chair> และ </chair> ในขณะเดียวกันก็มีอิลิเมนต์ <price> สำหรับโต๊ะที่อยู่ระหว่างอิลิเมนต์ <table> และ </table> ก็จะมีปัญหาสำหรับ parser ว่า

การตีความหมายของอิลิเมนต์ price จะขึ้นอยู่กับอิลิเมนต์แม่ หรือว่าผู้สร้างเอกสารทำผิดพลาด ซึ่งอาจจะแก้ปัญหาโดยการเปลี่ยนชื่ออิลิเมนต์ไปเป็น <chair-price> และ <table-price> ซึ่งก็ไม่ใช่ทางแก้ที่ดีนัก ทางแก้ที่เหมาะสมคือการใช้ Namespace กำหนดว่าอิลิเมนต์นั้นๆเป็นของใคร การกำหนด Namespace จะทำโดย

1. กำหนด URI ให้แต่ละ Namespace ที่ใช้
2. กำหนด prefix ให้ Namespace
3. ประกาศ Namspace โดยใช้แอตทริบิวต์ xmlns:prefix= "URI" เช่น

   <table:price

   xmlns:table= "http://www.table.com">
4. เมื่อมีการประกาศ Namespace แล้ว สามารถใช้ <prefix:name> ได้ทันที เช่น <table:price>

3.10 ข้อดีข้อเสียของ XML

3.10.1 ข้อดีของ XML

- สามารถกำหนดโครงสร้างข้อมูลเองได้
- แยกข้อมูลออกจากการแสดงผลทำให้ข้อมูลมีโครงสร้างที่ดี
- สามารถนำไปแสดงผลได้หลากหลายรูปแบบ
- แอพพลิเคชันสามารถใช้ข้อมูลร่วมกันบนข้อมูลเดียวกันได้
- เป็น text จึงสามารถสร้าง และแก้ไขได้ด้วย text editor
- มี API สนับสนุนมากมาย ในหลายภาษา
- มีแนวโน้มที่จะเป็นมาตรฐานในอนาคต
- โปรแกรมต่างๆเริ่มมีสนับสนุน XML
- สนับสนุนการใช้การเข้ารหัส unicode ทำให้ใช้ได้กับทุกภาษา

3.10.2 ข้อเสียของ XML

- การค้นหาข้อมูลใน XML นั้นไม่สามารถทำได้เหมือนกับ DBMS ที่ใช้ SQL เป็นภาษาในการทำ Query ถึงแม้จะมี XQL สำหรับทำหน้าที่นี้แล้วก็ตาม แต่ XQL ยังอยู่ในช่วงของการเริ่มต้น
- ความไม่ปลอดภัยของข้อมูล เนื่องจากเป็นเท็กซ์ไฟล์ ความปลอดภัยจึงต่ำเมื่อเทียบกับการเก็บข้อมูลบน DBMS ที่มีการจัดการดีกว่า
- เว็บบราวเซอร์ในปัจจุบันยังสนับสนุน XML ได้ไม่เต็มที่และการอิมพลีเมนต์การแสดงผลก็แตกต่างกันไปตามบริษัทผู้ผลิตด้วย เช่น อินเทอร์เน็ตเอ็กซ์พลอเรอร์จะสนับสนุนการใช้ XSL Transformation ในขณะที่เน็ตสเคปสนับสนุนการแสดงผลด้วย CSS

3.11 แนวโน้มในอนาคตของ XML

แนวโน้มของการใช้ XML ในพาณิชย์อิเล็กทรอนิคส์ในอนาคตจะมีมากขึ้นเรื่อยๆ เนื่องจากว่ามีการยอมรับในมาตรฐานต่างๆที่เกี่ยวกับ XML จาก W3C มากขึ้น อีกทั้งเครื่องมือและซอฟต์แวร์ต่างๆที่ใช้พัฒนาระบบจะสนับสนุนการใช้งาน XML มากขึ้น และเมื่อบริษัทคู่ค้าทางธุรกิจนำ XML มาใช้ จะส่งผลบังคับให้คู่ค้าต้องปรับตัวมาใช้ XML ด้วย การใช้ XML ในพาณิชย์อิเล็กทรอนิคส์ที่น่าจะมีมากที่สุดน่าจะเป็นการใช้ XML ในการแลกเปลี่ยนข้อมูลระหว่างธุรกิจ (Business-to-Business) โดยมีหลายสิ่งที่ชี้ให้เห็นว่า XML จะเป็นมาตรฐานในอนาคตมีดังนี้

3.11.1 ซอฟต์แวร์ที่สนับสนุน XML

ปัจจุบันบริษัทซอฟต์แวร์ขนาดใหญ่หลายบริษัทได้พัฒนาให้ซอฟต์แวร์ของตนสนับสนุน XML ทั้งสิ้น โดยมีตัวอย่างดังนี้

- เว็บบราวเซอร์ Microsoft Internet Explorer เริ่มมีการสนับสนุน XML บ้างในรุ่นที่ 4 และเพิ่มขึ้นในรุ่นที่ 5 และ 5.5
- เว็บบราวเซอร์ Netscape เริ่มสนับสนุน XML ในรุ่นที่ 4.0.4 และเพิ่มขึ้นในรุ่นที่ 6
- เว็บบราวเซอร์ Mozilla สนับสนุน XML ในรุ่นปัจจุบัน
- เว็บบราวเซอร์ Opera สนับสนุน XML ในรุ่นปัจจุบัน
- Oracle เริ่มสนับสนุน XML ตั้งแต่รุ่น 8i ขึ้นไป
- Apache ใช้ XML ในการกำหนดรายละเอียดในการ build ซอฟต์แวร์ของ Apache
- Macromedia สนับสนุนการใช้ XML ใน Dreamweaver และยังใช้ XML ในการกำหนดรายละเอียดของโปรแกรม

3.11.2 การนำ XML มาใช้ในอีคอมเมิร์ส

Tesco

Tesco เป็นร้านขายหนังสือและเว็บไซต์อีคอมเมอร์ส ที่ใหญ่ที่สุดในอังกฤษได้นำ XML มาใช้ในการสื่อสารระหว่าง Tesco กับคู่ค้า

3.11.3 แอพพลิเคชันของ XML

ภายในระยะเวลาไม่กี่ปีที่มี XML วงการต่างๆได้นำ XML ไปกำหนดภาษาของตนเองมากมาย โดยจะขอยกตัวอย่างแอพพลิเคชันที่มีผู้ใช้มากๆ

3.11.3.1 WML (Wireless Markup Language)

เป็นมาตรฐานที่กำหนดโดย WAP Forum ที่มีกลุ่มผู้ผลิตโทรศัพท์มือถือชั้นนำของโลกเป็นแกนนำ สำหรับการใช้อินเทอร์เน็ตบนอุปกรณ์ไร้สาย เช่น โทรศัพท์มือถือ หรือ Personal Digital Assistant

(PDA) โดยมีจุดประสงค์ในการนำมาแทน HTML เพราะว่าทรัพยากรของอุปกรณ์เหล่านี้มีน้อยกว่าบน PC ทั่วไปและยังมีข้อจำกัดในด้านการแสดงผลที่มีหน้าจอขนาดเล็ก

3.11.3.2 CDF (Channel Definition Language)

เป็นแอพพลิเคชันสำหรับการเผยแพร่เว็บไซต์โดยการใช้แนวคิดแบบการผลัก (Push) ซึ่งจะตรงกันข้ามกับการท่องชมเว็บไซต์ตามปกติทั่วๆไปที่ผู้ใช้จะทำการดึง (Pull) ข้อมูลที่ต้องการ โดยจะเป็นการนำเสนอเว็บไซต์มาให้ผู้ใช้โดยการที่ผู้ใช้ไม่จำเป็นจะต้องเลือกสรรค์ข้อมูลที่ต้องการเพราะทางเว็บไซต์จะคัดเลือกแล้วนำมาเสนอให้ โดยเนื้อหาจะเปลี่ยนแปลงไปเองตามเวลาซึ่งจะเหมือนกับการชมโทรทัศน์

บริษัทไมโครซอฟท์ซึ่งเป็นผู้ผลิตเว็บบราวเซอร์อินเทอร์เน็ตเอ็กซ์พลอเรอร์ได้นำ CDF มาใช้ในอินเทอร์เน็ตเอ็กซ์พลอเรอร์ รุ่นที่ 4 ขึ้นไป

3.11.3.3 SVG (Scalable Vector Graphics)

เป็นอีกแอพพลิเคชันของ XML ที่ได้มีการนำมาใช้กับเว็บบราวเซอร์ของบริษัทไมโครซอฟท์ โดย SVML เป็นภาษาที่ใช้ในการวาดรูปที่เป็นแบบเว็คเตอร์

3.11.3.4 XQL (eXtensible Query Language)

XQL เป็นภาษาที่ออกแบบขึ้นสำหรับการ Query ข้อมูลในเอกสาร XML ให้เหมือนอย่างการทำดาตาเบส Query ถึงแม้ว่าในปัจจุบันจะยังไม่ถูกยอมรับเป็นมาตรฐานจาก W3C แต่ XQL ก็เป็นที่กล่าวถึงและนิยมเป็นอย่างมากจนถือว่าเป็นมาตรฐาน *de facto* สำหรับการเข้าถึงข้อมูลที่เก็บใน XML

โครงสร้างของ Query ถูกกำหนดโดยแนวคิดของ Xpath และผลลัพธ์ที่ได้จะถูกกำหนดโดยการใช้แท็กของ XQL ตัวอย่างของ XQL Query ต่อไปนี้จะหาข้อมูลในตาราง books แล้วรีเทิร์นทุกเรคอร์ดที่มี title เป็น “Java” โดยจะแสดงค่าจากคอลัมน์ author เป็นผลลัพธ์ของแต่ละเรคอร์ด

```
//book[title contains “Java”] (.//authors)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
<xql:result>
  <book>
    <author name=“Richard Monson-Haefel”  location= “Minnesota” />
  </book>
  <book>
    <author name= “Jason Hunter” location= “California” />
    <author name= “William Crawford” location= “Massachusetts” />
  </book>
</xql:result>
```

XQL จะมีการเปลี่ยนแปลงอยู่เรื่อยๆจนกว่าจะได้รับการยอมรับเป็นมาตรฐานจาก W3C โดยสามารถดูรายละเอียดได้ที่ http://metalab.unc.edu/xql/xql-proposal.html

บทที่ 4 การวิเคราะห์และออกแบบ

4.1 การวิเคราะห์

4.1.1 ความต้องการของระบบ

1. ระบบจะต้องสามารถตอบสนองความต้องการของผู้ใช้ได้ในด้านการค้นหาสินทรัพย์
2. ระบบจะต้องสามารถค้นหาสินทรัพย์ให้ผู้ใช้แบบออฟไลน์ได้ ถ้าการค้นหาแบบออนไลน์ไม่พบสินทรัพย์ที่ต้องการ
3. ระบบจะต้องสามารถตอบสนองความต้องการของผู้ใช้ได้ในด้านของการประกาศขาย/เช่าโดยยอมให้มีการแก้ไขรายละเอียดการประกาศได้
4. ระบบจะต้องตอบสนองต่อผู้ใช้ด้วยความเร็วตามสมควร
5. ระบบจะต้องสามารถรวบรวมข้อมูลของสินทรัพย์ที่มีอยู่บนอินเทอร์เน็ตที่อยู่ในรูปของ HTML ได้
6. ผู้ดูแลระบบสามารถควบคุมการทำงานของเอเจนต์ในการเก็บข้อมูลจากอินเทอร์เน็ตได้

4.1.2 ความสามารถและบริการของระบบช็อปปิ้งเอเจนต์

ความสามารถและบริการของระบบแบ่งออกเป็น 2 ส่วนได้แก่

1. ความสามารถและบริการสำหรับผู้ใช้
   - สมัครสมาชิก
   - แก้ไขข้อมูลของสมาชิก
   - ค้นหาสินทรัพย์แบบออนไลน์
   - ค้นหาสินทรัพย์แบบออฟไลน์
   - ประกาศขาย/เช่าสินทรัพย์ของสมาชิก
   - แก้ไขการประกาศขาย/เช่าสินทรัพย์ของสมาชิก
2. ความสามารถและบริการสำหรับผู้ดูแลระบบ
   - ควบคุมการเก็บข้อมูลจากเว็บไซต์ต่างๆได้
   - ควบคุมการทำงานของระบบค้นหาแบบออฟไลน์
   - สอนการช็อปปิ้งให้กับช็อปปิ้งเอเจนต์โดยใช้เครื่องมือที่มีอินเทอร์เฟสแบบกราฟิค (GUI)

4.1.3 ระเบียบของระบบช็อปปิ้งเอเจนต์

1. ผู้ใช้ทุกคนจะต้องทำการสมัครเป็นสมาชิก
2. สมาชิกจะต้องล็อกอินเข้าสู่ระบบก่อนใช้บริการ

3. สมาชิกสามารถใช้บริการค้นหาอพาร์ทเมนต์และคอนโดมีเนียม ได้ทั้งแบบออนไลน์และออฟไลน์
4. การค้นหาแบบออฟไลน์จะทำก็ต่อเมื่อการค้นหาแบบออนไลน์ไม่ได้ผล
5. สมาชิกสามารถประกาศเช่า/ขาย สินทรัพย์ที่มีอยู่ได้
6. การประกาศขาย/เช่า จะถูกลบออกเมื่อสมาชิกไม่ได้ล็อกอินเข้าสู่ระบบภายในระยะเวลาที่กำหนด

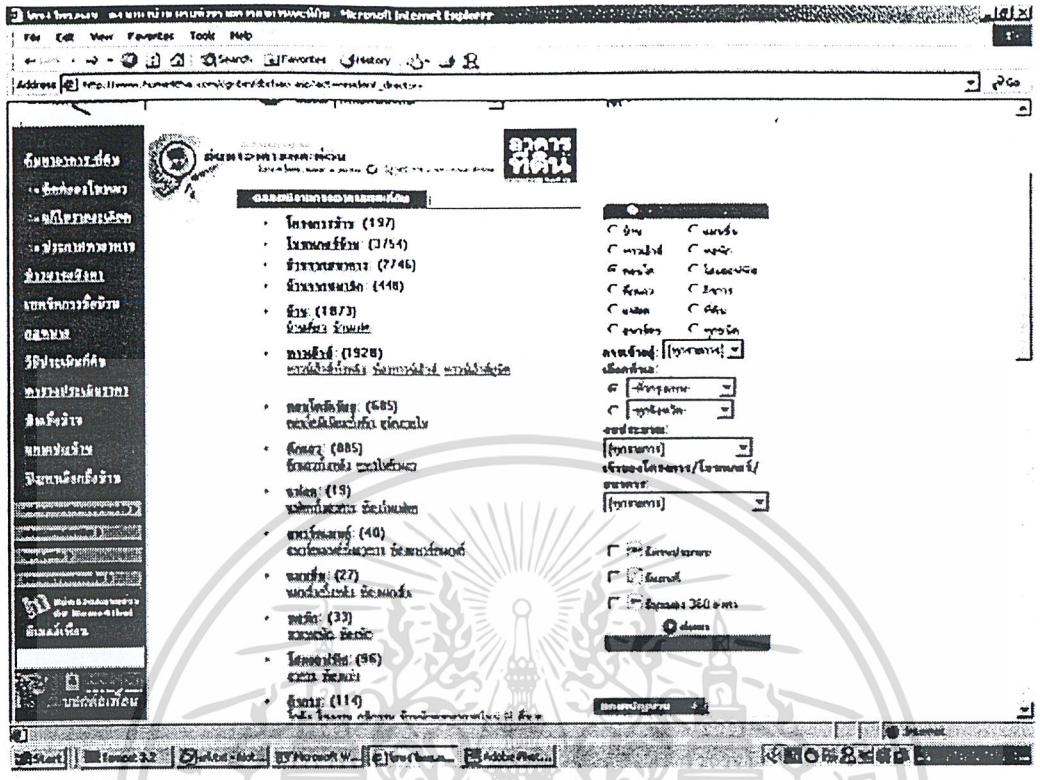
4.1.4 ข้อจำกัดของระบบช็อปปิ้งเอเจนต์

1. ช็อปปิ้งเอเจนต์ไม่สามารถคัดเลือกข้อมูลออกมาได้ทุกเว็บเพจ หรือข้อมูลที่คัดเลือกออกมาได้อาจมีความผิดพลาด เนื่องจากข้อมูลไร้โครงสร้างหรือความบกพร่องในการป้อนข้อมูลของเจ้าของเว็บไซต์
2. ระบบช็อปปิ้งเอเจนต์จะทำการค้นหาสินทรัพย์มาเก็บไว้ในดาตาเบสทำให้ต้องใช้ฐานข้อมูลขนาดใหญ่
3. เนื่องจากการนำข้อมูลมาเก็บไว้ที่ฐานข้อมูล อาจส่งผลให้ข้อมูลไม่มีความทันสมัยบ้าง

4.2 องค์ประกอบของโครงงาน

4.2.1 สไปเดอร์

ในโครงงานนี้สไปเดอร์จะทำหน้าที่เก็บรวบรวมข้อมูลของสินทรัพย์ประเภท Condominium และ Apartment จาก websites ที่ให้บริการค้นหาข้อมูลของอสังหาริมทรัพย์โดยทำการส่งค่าตัวแปรต่างๆ เข้าไปยัง CGI แล้วทำการเก็บไฟล์ HTML ที่ได้มาลงดิสก์ยังเครื่องที่สไปเดอร์ตั้งอยู่ ซึ่งไฟล์ HTML ที่ได้มานั้นส่วนใหญ่จะมีรูปแบบเป็นตารางที่มีรายการของสินทรัพย์ต่างๆ เช่น ชื่อสินทรัพย์ ราคา ทำเลที่ตั้ง เป็นต้น และในแต่ละสินทรัพย์ก็จะมีลิงก์ไปยังหน้าที่มีข้อมูลเพิ่มเติมของสินค้านั้น และสไปเดอร์ก็จะทำการเก็บลิงก์นั้นด้วย



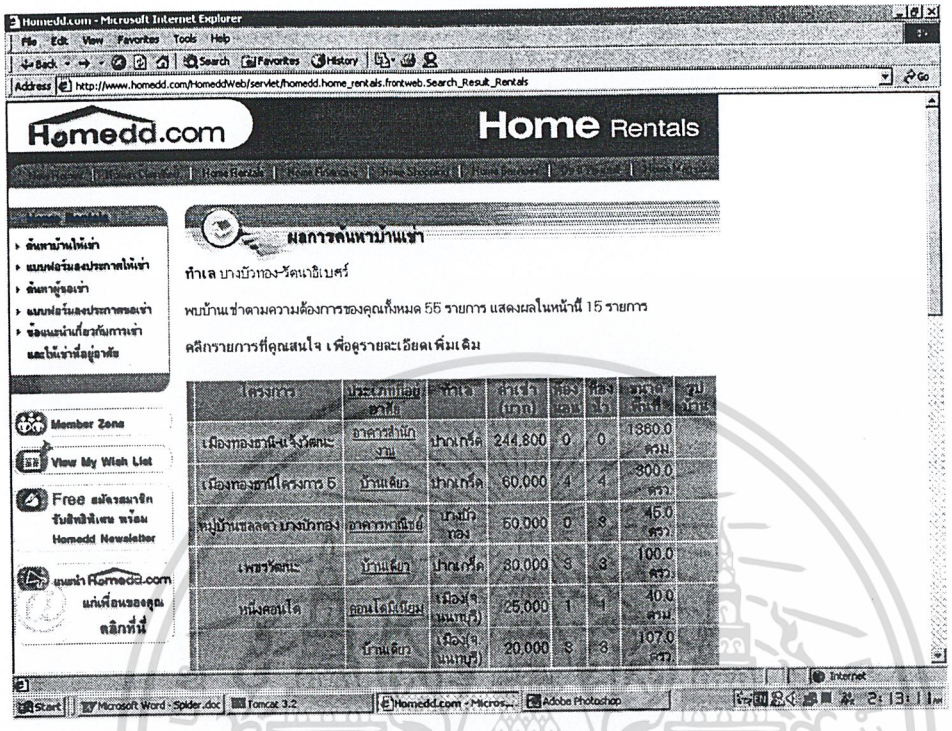
รูปภาพ 4-1 ตัวอย่างหน้าเว็บเพจที่มีฟอร์มการค้นหา

สไปเดอร์จะทำการสร้างเธรดขึ้นมาเพื่อรองรับลิงก์ที่เข้ามาโดยที่เธรดแต่ละเธรดจะอ่าน Resource จากไฟล์ข้อกำหนด ซึ่งเป็น XML 1 หนึ่งเว็บไซต์จะมี 1 Resource ซึ่ง Resource นั้นจะเป็นตัวที่บอกว่า เว็บไซต์นี้มี URL อะไร มีการเก็บข้อมูลเป็นอย่างไร URL ในเว็บไซต์ตัวอย่างนี้คือ

http://www.home4thai.com/cgibin/dbchao.asp?act=resident_search_result&PAGENO=2&QUERY_MAINTYPE=04&QUERY_TYPE=&QUERY_AREA=01000&QUERY_PROVINCE=01&QUERY_MOVEIN_TYPE=&QUERY_BUDGET=&QUERY_RESIDENT_COMPANY=&OPTION1=1&QUERY_MEMBER=&QUERY_PICTURE_1=&QUERY_MAP=

จะเห็นว่าใน URL จะมี ตัวแปรที่ส่งผ่านฟอร์มใน HTML ไปยังเซิร์ฟเวอร์ด้วย แล้วเซิร์ฟเวอร์ก็จะส่งไฟล์ HTML กลับมาโดยไฟล์ HTML ที่ได้มาส่วนใหญ่จะอยู่ในรูปของตารางที่มีข้อมูลของสินทรัพย์อยู่ สาเหตุที่เลือกข้อมูลที่มาจากฟอร์มนั้นเพราะว่าผลลัพธ์ที่ได้ส่วนใหญ่นั้นจะอยู่ในรูปของตารางทำให้ง่ายต่อการดึงข้อมูล ซึ่งข้อมูลจะถูกแยกเป็นคอลัมน์อยู่แล้ว สิ่งที่สไปเดอร์ทำคือทำการดึงข้อมูลที่อยู่ในเซลล์นั้นออกมา ส่วนใหญ่ข้อมูลในแต่ละแถวนั้น จะมีการแสดงผลที่เหมือนกัน เพราะผลลัพธ์ที่ได้นั้นมาจาก CGI ซึ่งส่วนใหญ่แล้วจะทำการ Query ออกมาและวนลูปแสดงผลออกมา ซึ่งโดยทั่วไปแล้วหนึ่งแถ

วจะหมายถึงข้อมูล 1 ชุด ถ้าเป็นอสังหาริมทรัพย์นั้นก็จะหมายถึงสินทรัพย์หนึ่งแห่ง เป็นต้น และยังเป็นการหลีกเลี่ยงการตัดคำจากข้อมูลด้วยเพราะว่าข้อมูลได้ถูกแยกแล้วว่าคอลัมน์นี้เป็นข้อมูลอะไรดังตัวรูป



รูปภาพ 4-2 ตัวอย่างรูปผลการค้นหาสินทรัพย์

เมื่อเธรดแต่ละเธรดได้ File HTML มาแล้วก็จะทำการ parse โดยใช้ HTML parser ของ Java ซึ่งมีอยู่แล้ว และจะทำการเก็บข้อมูลของแต่ละสินทรัพย์ตาม Resource ที่ได้ถูก config ไว้แล้วพร้อมทั้งเก็บลิงก์ที่เป็นข้อมูลเพิ่มเติมของแต่ละสินทรัพย์ถ้าจำเป็นต้องเข้าไปเก็บข้อมูลในหน้านั้นเธรดแต่ละเธรดนั้นจะเก็บข้อมูลเท่าที่สามารถเก็บได้ เนื่องจากว่าข้อมูลที่ได้นั้นเป็นข้อมูลดิบยังไม่สามารถนำไปใช้ได้จริงจึงมี Class Formattor ทำหน้าที่นำข้อมูลไปทำการแปลงให้เป็นข้อมูลที่สามารถเก็บลง ดาต้าเบสแล้วสามารถนำมาใช้กับฟังก์ชันอื่นๆ ในโครงงานได้ ตัวอย่างเช่นถ้าสามารถเก็บข้อมูลราคาของแต่ละสินทรัพย์ได้ก็จะสามารถทำการเปรียบเทียบราคาและทำการค้นหาตามราคาที่ผู้ค้นหาต้องการ ในส่วนของ Class Formatter นั้นจะทำหน้าที่เป็นเธรดหนึ่งของสไปเดอร์ ซึ่งเป็นเธรดที่ทำการดึงเอาข้อมูลของแต่ละสินทรัพย์ออกมาทำการเปลี่ยนไปเป็นรูปแบบของข้อมูลที่สามารถนำไปใช้ได้จริง โดยสำหรับวิธีที่จะทราบว่าต้องทำการเปลี่ยนแปลงได้อย่างไรนั้น Formattor ก็จะทำการอ่าน Resource ที่อยู่ในไฟล์ข้อกำหนดที่เป็น XML ซึ่งในแต่ละเว็บไซต์ก็จะมีรูปแบบของข้อมูลต่างกัน

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <resource_pointer>
  - <resource>
      <name>Home 4 Thai</name>
      <url>http://www.home4thai.com/cgi-bin/dbchao.asp?
        act=resident_search_result&PAGENO=2&QUERY_MAINTYPE=04&QUERY_TYPE=&QUERY_AREA=010
      <local_site />
      <order>0 5 7 4 8 6 1 2 3 9</order>
      <end_property>4</end_property>
    - <start_property available="Yes" place="result" value="" skip="0">
        <before>open_tr^^open_td^^open_font^^open_a</before>
        <after>close_a^^close_font</after>
      </start_property>
    - <property_size available="Yes" place="extend" value="" skip="0">
        <before>open_tr^^open_td^^open_font</before>
        <after>simple_img</after>
        <data_format prefix_idx="0" suffix_idx="6" accept_charactor="" cut_off_charactor="" end_charactor="- -" />
      </property_size>
    - <property_price available="Yes" place="extend" value="" skip="0">
        <before>open_td^^open_font</before>
        <after>simple_br^^simple_img</after>
        <data_format prefix_idx="0" suffix_idx="4" accept_charactor="" cut_off_charactor="" end_charactor="-" />
      </property_price>
    - <property_province available="No" place="result" value="Bangkok" skip="0">
        <before>open_body</before>
        <after>close_body</after>
        <data_format prefix_idx="0" suffix_idx="0" accept_charactor="" cut_off_charactor="" end_charactor="" />
      </property_province>
    - <property_area available="Yes" place="result" value="" skip="0">
        <before>close_a^^close_td^^open_td^^open_font</before>
        <after>close_font</after>
        <data_format prefix_idx="0" suffix_idx="0" accept_charactor="" cut_off_charactor="" end_charactor="" />
      </property_area>
```

รูปภาพ 4-3 ไฟล์ Resource ที่เป็น XML

เมื่อได้ข้อมูลที่เป็นข้อมูลแล้วก็จะสามารถเก็บลงดาต้าเบสได้ เพื่อนำไปใช้ต่อไป ในการเก็บข้อมูลนั้นในดาต้าเบสนั้น จะมี 2 ตารางที่จะคอยสับเปลี่ยนขณะที่มีผู้ใช้ต้องการค้นหาข้อมูลจากดาต้าเบสในขณะที่สไปเดอร์กำลังดึงข้อมูลจากดาต้าเบสอยู่ สาเหตุที่ต้องทำแบบนี้เพราะว่า ก่อนที่จะให้สไปเดอร์ทำงานนั้นจะทำการลบข้อมูลทั้งหมดในตารางนั้นก่อน ดั้งนั้นการมีตารางอีกตารางหนึ่งนั้นก็สามารถป้องกันไม่ให้ user ที่จะทำการค้นหาข้อมูลไม่เจอข้อมูลที่ว่างเปล่า ยังสามารถมีข้อมูลเก่ารองรับอยู่ได้

แต่เมื่อทำเช่นนี้แล้วก็จะเกิดปัญหาขึ้น คือข้อมูลที่ user ได้จากการค้นหาจากดาต้าเบสนั้นอาจเป็นข้อมูลที่ไม่ทันสมัยเพราะว่าในขณะที่ผู้ใช้ได้ข้อมูลนั้นสไปเดอร์กำลังดึงข้อมูลที่ทันสมัยกว่ามาจากเว็บไซต์ต่างๆ ดังนั้นการให้สไปเดอร์ทำงานจึงต้องเลือกเวลาที่มีผู้ใช้เข้ามาใช้บริการน้อยที่สุด จึงมีโอกาสที่จะเกิดปัญหาน้อยลง

4.2.1.1 อัลกอริทึม และ Resource File

ในการดึงข้อมูลจากไฟล์ HTML ที่ได้มานั้น ข้อมูลที่ต้องการจะดึงมีดังนี้

1. ชื่อของสินทรัพย์
2. ชนิดของสินทรัพย์
3. ที่ตั้งของสินทรัพย์ ซึ่งประกอบไปด้วย จังหวัด และ เขต (หรืออำเภอ)

4. หน้าที่ลิงก์ไปยังข้อมูลของสินทรัพย์นั้น
5. ที่อยู่ที่แท้จริงของสินทรัพย์นั้น
6. ขนาดของสินทรัพย์
7. จำนวนห้องนอน

ในการเก็บข้อมูลแต่ละแบบจะมีตำแหน่งที่อยู่ในไฟล์ HTML ที่ต่างกันดั้งนั้นจะต้องมีกฎที่ระบุว่าข้อมูลต้องการอยู่ระหว่าง Tag ใด ดั้งนั้นจะต้องมีการระบุว่าก่อนที่จะพบข้อมูลนั้น Parser จะพบ Tag ใดก่อนบ้าง และ พบ Tag ใดบ้างทีพบตามมา โดยจะไม่สนใจแอตทริบิวต์ที่อยู่ใน Tag แต่ละ Tag นั้น โดยจะมีกฎต่างๆที่จะช่วยทำให้ดึงข้อมูลตามที่ต้องการได้ตรงมากขึ้น ในการ Format ข้อมูลนั้น เนื่องจากการตัดคำภาษาไทยนั้นมีความซับซ้อน จึงใช้วิธีที่ง่ายในการตัดโดยการกำหนดกฎขึ้นมาสำหรับการตัดตัวอักษรด้วย ตัวอย่างกฎทั้งการเก็บและตัดข้อมูล

```
<property_size available="Yes" place="extend" value="" skip="0">
        <before>open_tr^^open_td^^open_font</before>
        <after>simple_img</after>
        <data_format prefix_idx="0" suffix_idx="6" accept_charactor=""
cut_off_charactor="" end_charactor="- -" />
</property_size>
```

Tag ของ XML นี้เป็นกฎที่ระบุว่าจะดึงข้อมูลที่เป็นราคาของสินค้าได้อย่างไร โดยจะมี

- Tag before ที่จะมีสตริงระบุว่าก่อนจะพบข้อมูลส่วนนี้จะประกอบด้วย Tag ใดบ้าง ในที่นี้จะหมายความว่า จะพบ <tr><td><font> ส่วนต่อจากนี้ก็จะเป็นส่วนของข้อมูล ตัวอย่างเช่น

  <tr><td><font>125 ตารางเมตร ข้อมูลที่ต้องการก็คือ 125 ตารางเมตร เพราะฉนั้นข้อมูลที่นำหน้าด้วย Tag เหล่านี้ก็อาจจะเป็นข้อมูลที่ต้องการได้ ดังนั้นจึงต้องใช้ Tag ที่ตามหลังข้อมูลที่ต้องการมาช่วยในการระบุว่าเป็นข้อมูลที่ต้องการจริงหรือไม่

- Tag after หลังจากที่ได้พบข้อมูลที่มีความเป็นไปได้ที่จะเป็นข้อมูลที่ต้องการแล้ว เราต้องทำการระบุด้วยว่าข้อมูลที่ต้องการนั้นตามหลังด้วย Tag อะไรบ้าง ตัวอย่างเช่น ตามด้วย <img> ถึงจะเป็นข้อมูลที่ต้องการจริงๆ และสามารถนำข้อมูลนั้นมาใช้งานได้

  ในกรณีที่ข้อมูลที่เราต้องการทั้งสองแอตทริบิวต์ ตัวอย่างเช่น ราคา และ ขนาดของสินทรัพย์ มีรูแบบของ html ที่เหมือนกัน เราไม่จำเป็นต้องระบุ Tag นำหน้าหรือตามหลังให้มีความเฉพาะเจาะจงมากขึ้น เราสามารถใช้การตั้งเงื่อนไขที่เหมือนกันได้ทั้งสองแอตทริบิวต์ได้เลย เพียงแต่กำหนดไว้ว่าแอตทริบิวต์ไหนมาก่อนเท่านั้นก็จะสามารถรู้ได้ว่าข้อมูลที่ได้มาก่อนหรือหลังนั้นเป็นข้อมูลอะไร

- แอตทริบิวต์ available ใน Tag property_size นี้จะเป็นตัวระบุว่าจะสามารถพบข้อมูลนี้ได้ในหน้า HTML หน้านี้หรือเปล่า ในที่นี้จะหมายถึงไม่สามารถเก็บข้อมูลราคามาจากหน้านั้นได้หรือไม่

- แอตทริบิวต์ place ใน Tag property_size จะเป็นตัวที่ระบุว่าข้อมูลที่จะได้อยู่ในหน้าที่เป็นหน้าผลรวมของทุกสินทรัพย์ หรือว่าอยู่ในหน้าที่มีข้อมูลเพิ่มเติมของสินทรัพย์นั้น
- แอตทริบิวต์ skip เป็นตัวแปรที่ระบุว่า ถ้ามีข้อมูลใดที่ไม่ต้องการที่สามารถเก็บมาได้ตามกฎที่เราตั้งไว้นั้น สามารถทำการข้ามข้อมูลส่วนไปได้โดยไม่นำมาคิดเป็นข้อมูล และทำการหาข้อมูลที่ตรงกับกฎนั้นอีกครั้ง
- Tag data_format เป็น Tag ที่ระบุถึงกฎที่ใช้ในการทำการ Format หลังจากที่ได้ข้อมูลที่ต้องการมาจากหน้า HTML แล้ว
- แอตทริบิวต์ prefix_idx นั้นเป็นตัวแปรที่บอกว่าข้อมูลที่ได้มานั้นจะทำการตัดตัวอักษรข้างหน้าเป็นจำนวนกี่ตัว ตัวอย่างเช่น ถ้าได้ข้อมูลมาจากหน้าที่เป็นหน้าข้อมูลเพิ่มเติมเป็น Price :1500000 ดังนั้นตัวแปรนี้จึงมีค่าเท่ากับ 7 เพราะว่าต้องการตัดคำว่า Price : ออกไป ดังนั้นข้อมูลใหม่ที่จะได้นั้นจึงมีแต่ราคาอย่างเดียว
- แอตทริบิวต์ suffix_idx นั้นเป็นตัวแปรแบบเดียวแบบ prefix_idx แต่ตัดจากข้างหลัง ตัวอย่างเช่น Price :1500000 บาท จะได้ตัวแปร prefix_idx เป็น 7 และ suffix_idx เป็น 4 เพราะต้องการจะตัด “ บาท” ออกไปจะได้เหลือแต่ตัวเลขอย่างเดียว
- แอตทริบิวต์ accept_character เป็นตัวแปรที่ระบุว่าต้องการตัวอักษรใดที่ต้องการในข้อความนั้น ตัวอย่างเช่น ข้อมูล Price : 1500000 บาท ตัวแปรก็จะเป็น ”0123456789” ดังนั้นข้อมูลที่จะได้ 1500000
- แอตทริบิวต์ cut_off_character เป็นตัวแปรที่ระบุว่าต้องการจะตัดตัวอักษรใดในข้อความนั้น ตัวอย่างเช่น 1,500,000 ตัวแปรก็จะเป็นเครื่องหมาย “,” ข้อมูลก็จะเปลี่ยนไปเป็น 1500000 หลังจากทำการตัดแล้ว
- แอตทริบิวต์ end_character เป็นตัวแปรที่ระบุว่าถ้าพบ character ตัวใดในข้อความให้ทำการหยุดการอ่านข้อมูลตำแหน่งนั้นและไม่นำข้อมูลที่ต่อท้ายจาก character ตัวนั้นมาเป็นข้อมูลอีก ตัวอย่าง ถ้าข้อมูลเป็นช่วงเช่นราคา 45000-50000 ถ้าเราต้องการดึงข้อมูลราคาต่ำมา ตัวแปรก็จะมีค่าเป็น “-“

4.2.2 ส่วนติดต่อกับผู้ใช้

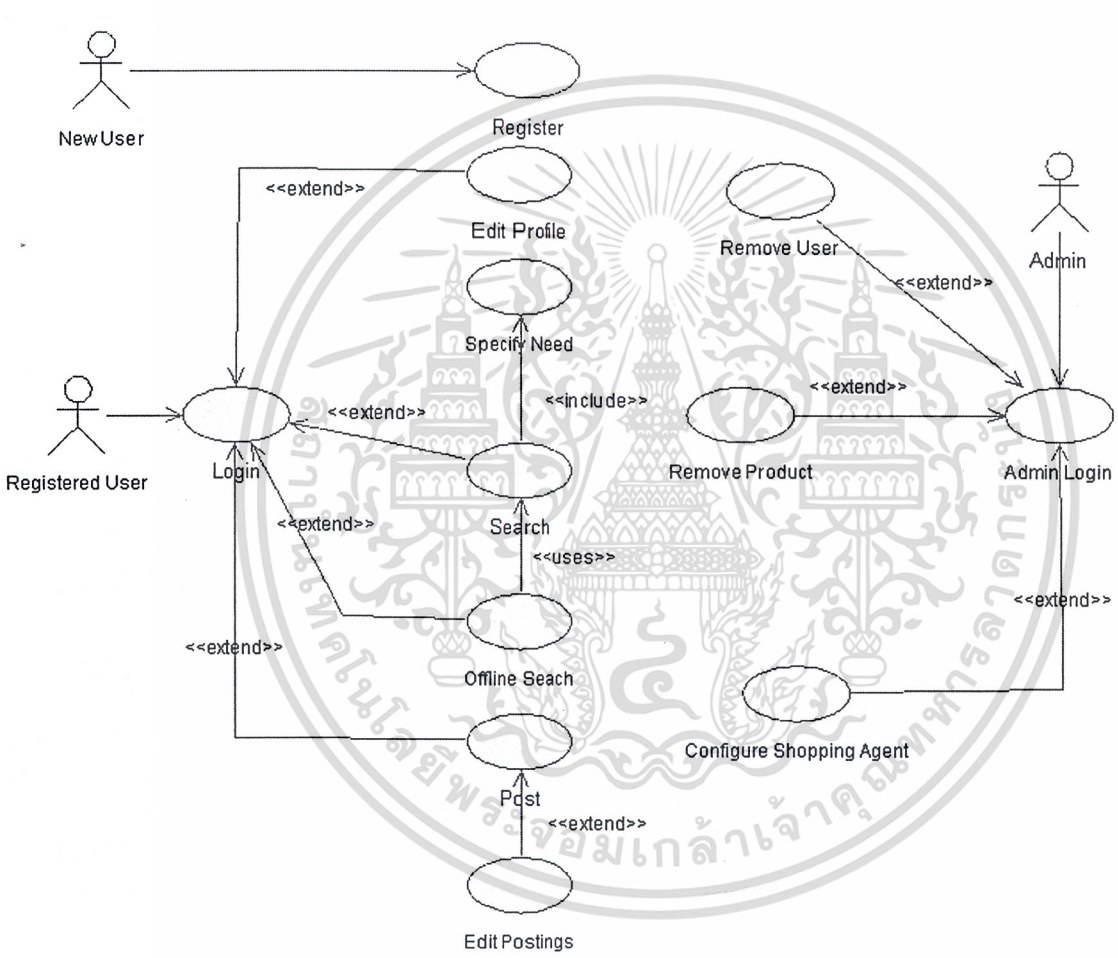
การติดต่อกับผู้ใช้จะเป็นการติดต่อทางเว็บเพจเท่านั้น ซึ่งหมายความว่าจะใช้ HTML ในการติดต่อกับเว็บบราวเซอร์ของผู้ใช้ ซึ่งประกอบด้วยหน้าต่างๆดังนี้

- หน้าแรกของเว็บไซต์ เป็นหน้าต้อนรับเข้าสู่ระบบ
- หน้าลงทะเบียน
- หน้าล็อกอินเข้าสู่ระบบ
- หน้ารายละเอียดของบริการ
- หน้ารายละเอียดการติดต่อผู้ดูแลระบบ
- หน้าแก้ไขข้อมูลสมาชิก

- หน้าค้นหาคอนโดมิเนียม/อพาร์ทเมนต์
- หน้าประกาศขาย/เช่า
- หน้ายืนยันการเลิกการติดต่อกับระบบ

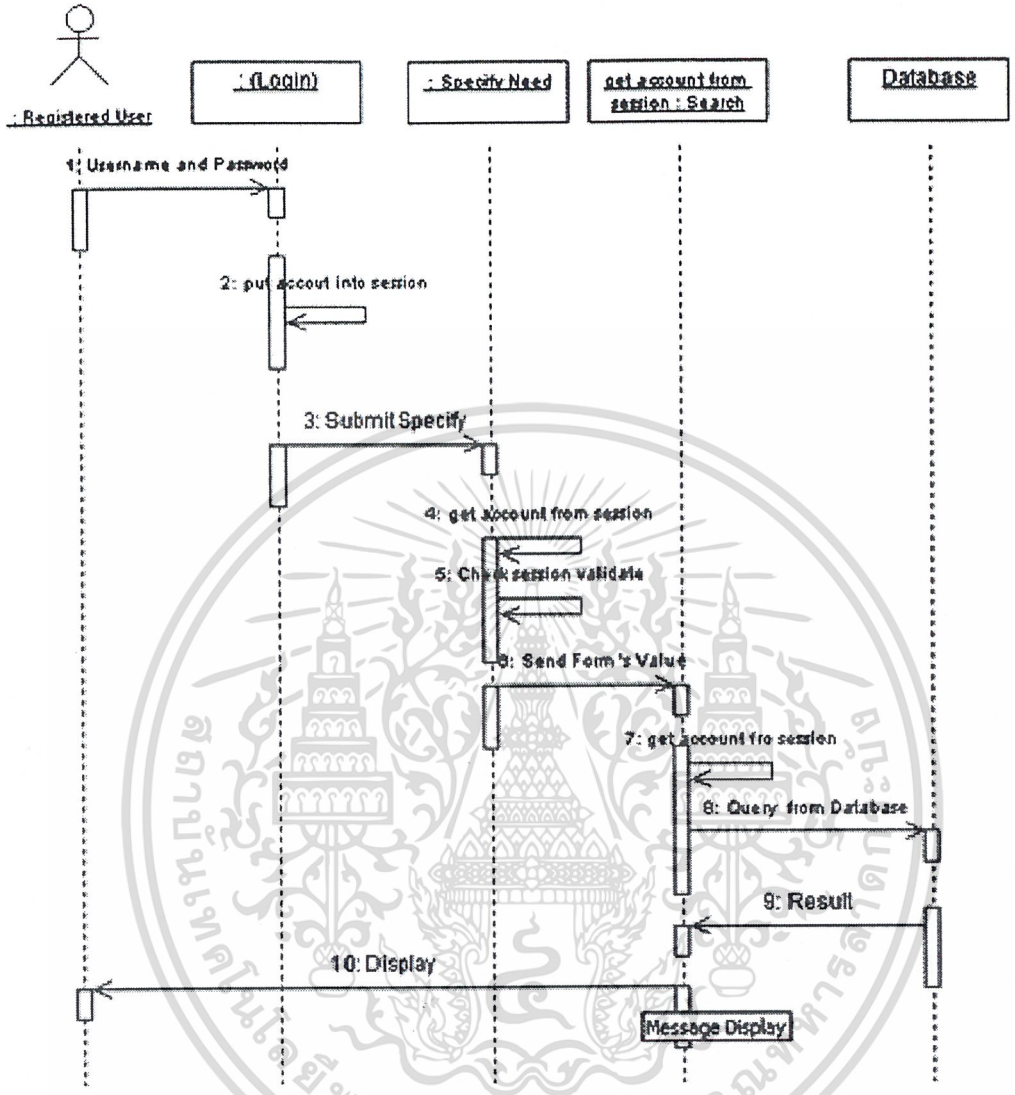
4.3 การออกแบบระบบโดยใช้ UML Diagram

4.3.1 Use Case Diagram



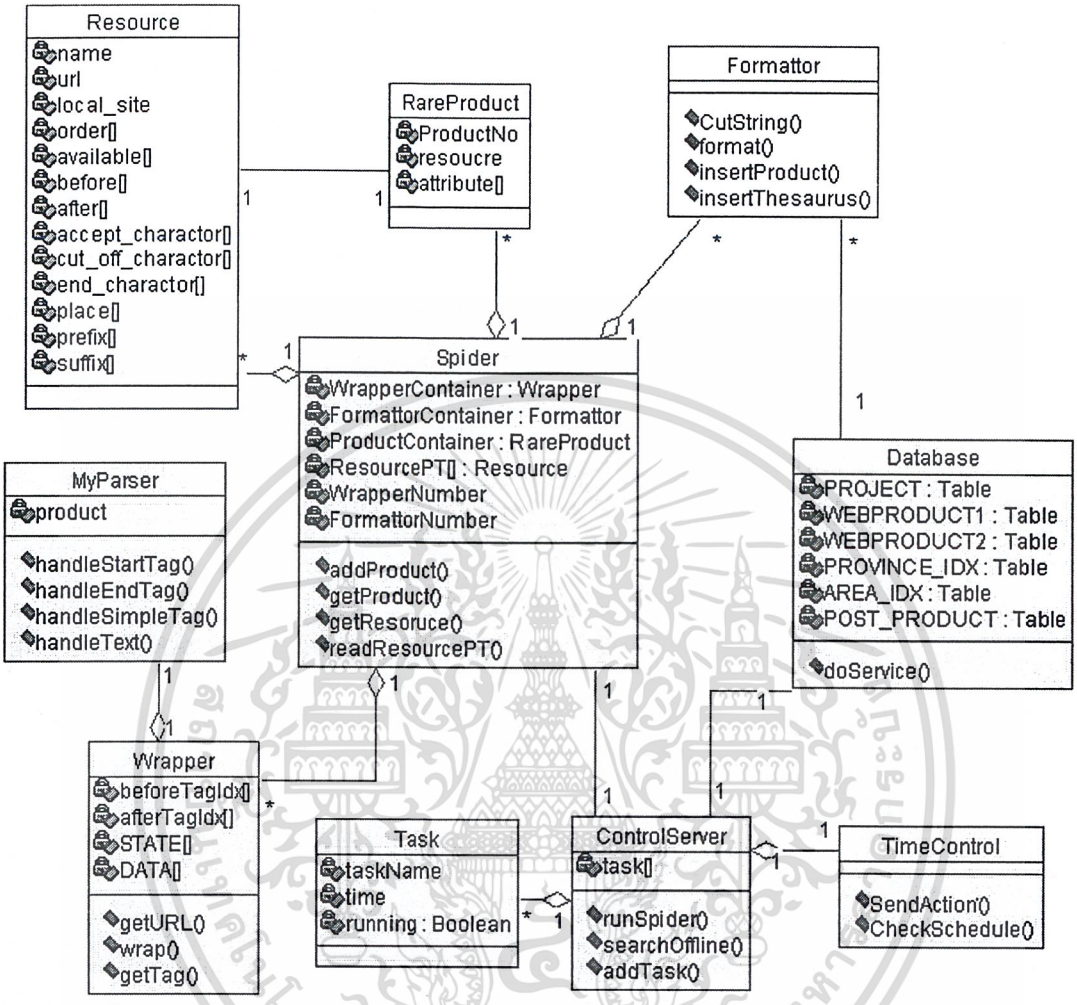
รูปภาพ 4-4 Usecase Diagram ของระบบ

4.3.2 Sequence Diagram



รูปภาพ 4-5 Sequence Diagram ของการค้นหาอพาร์ทเมนต์และคอนโดมิเนียม

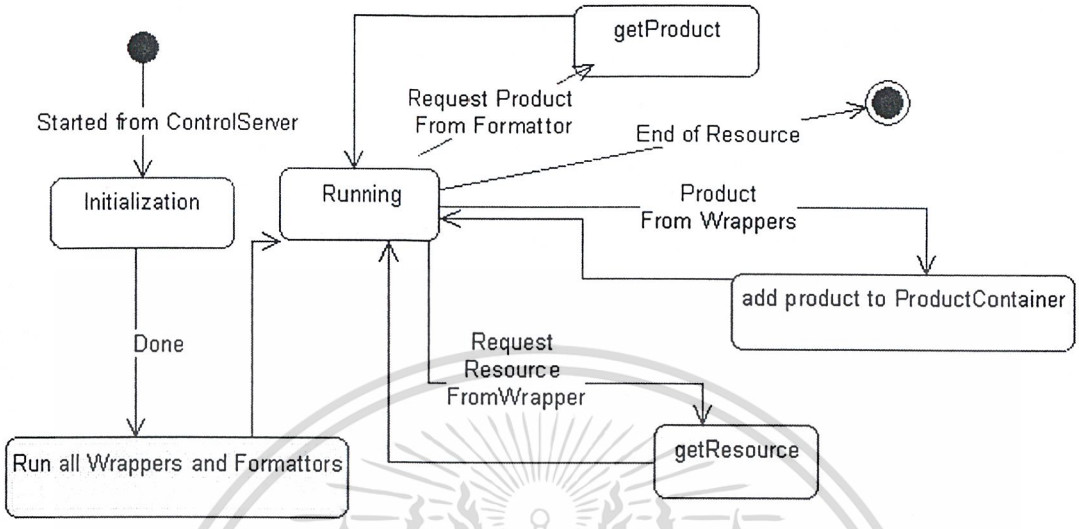
4.3.3 Class Diagram



รูปภาพ 4-6 Class Diagram ของระบบ

4.3.4 State Diagram

4.3.4.1 Spider



รูปภาพ 4-7 State Diagram ของสไปเดอร์

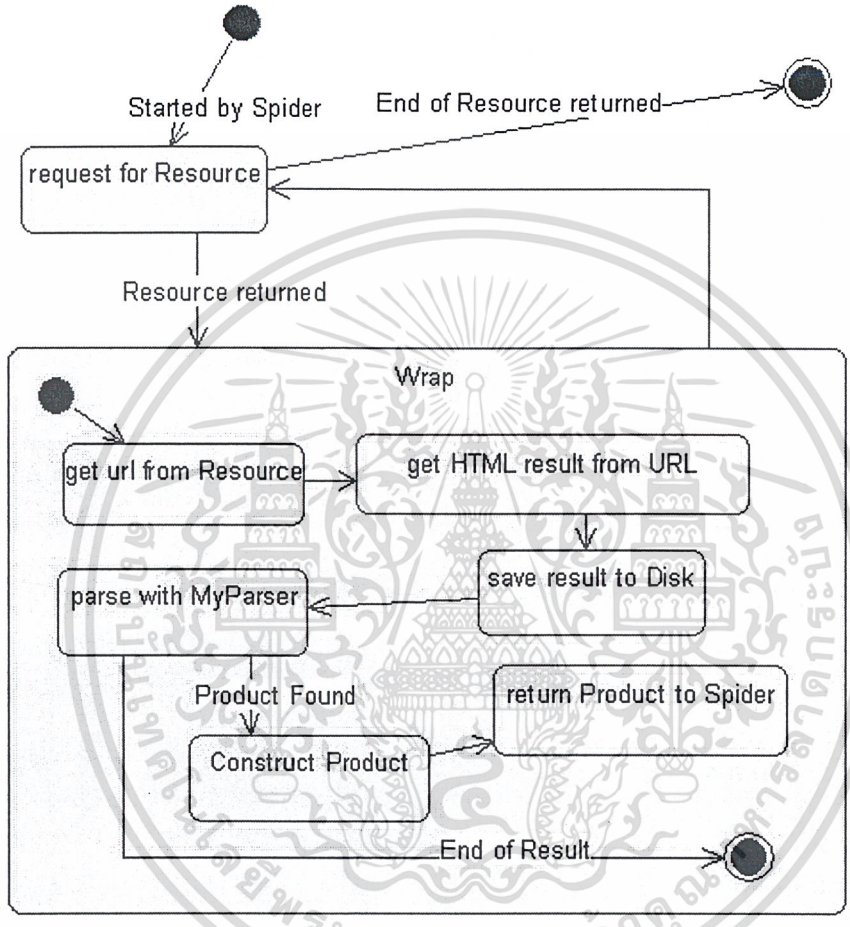
สไปเดอร์จะถูกสร้างและควบคุมโดย ControlServer ซึ่งจะสั่งงานตามเวลา เมื่อได้รับคำสั่งจาก ControlServer ให้เริ่มการทำงาน สไปเดอร์ก็จะทำดังขั้นตอนดังดัง State Diagram ข้างต้นนี้

คำอธิบาย State

- **Initialization** กำหนดข้าเริ่มต้นต่างๆ พร้อมทั้งสร้าง Wrapper และ Formattor ขึ้นมา ซึ่งทั้งสองตัวนี้จะเป็นเธรดที่ถูกเก็บไว้ใน WrapperContainer และ FormattorContainer ตามลำดับ
- **Run all Wrappers and Formattors** สไปเดอร์ทำการสั่งให้ Wrapper และ Formattor ทำงานโดยผ่าน WrapperContainer และ FormattorContainer
- **Running** เป็นช่วงที่รอการทำงานของ Wrapper และ Formattor ซึ่ง Wrapper จะอ่าน Resource ไปตัวละ 1 Resource จนกว่าจะหมดโดยมี Wrapper หลายตัวช่วยกัน เมื่อ Resource ก็หมายความว่า จบการทำงานของ Spider
- **GetResource** หลังจากที่ Wrapper ถูกสั่งให้ทำงานแล้ว ก็จำทำการ Request Resource จากสไปเดอร์เพื่อนำ Resource ที่ได้ไปเป็นกฎที่ไปใช้ในการดึงข้อมูลในเว็บไซต์ต่อไป
- **Add product to ProductContainer** เมื่อ Wrapper ทำการ Parse หมดแล้วก็จะส่ง Product ที่ยังไม่ได้ Format นี้ไปไว้ทีสไปเดอร์ก่อน หลังจากนั้น Formattor ก็จะทำการ Request Product เอง

- **GetProduct** เมื่อ Formattor ทำการ Request สไปเดอร์ก็จะทำการส่ง Product ที่มาที่หลังสุดให้กับ Formattor เพื่อไปทำการ Format ต่อไป

4.3.4.2 Wrapper



รูปภาพ 4-8 State Diagram ของ Wrapper

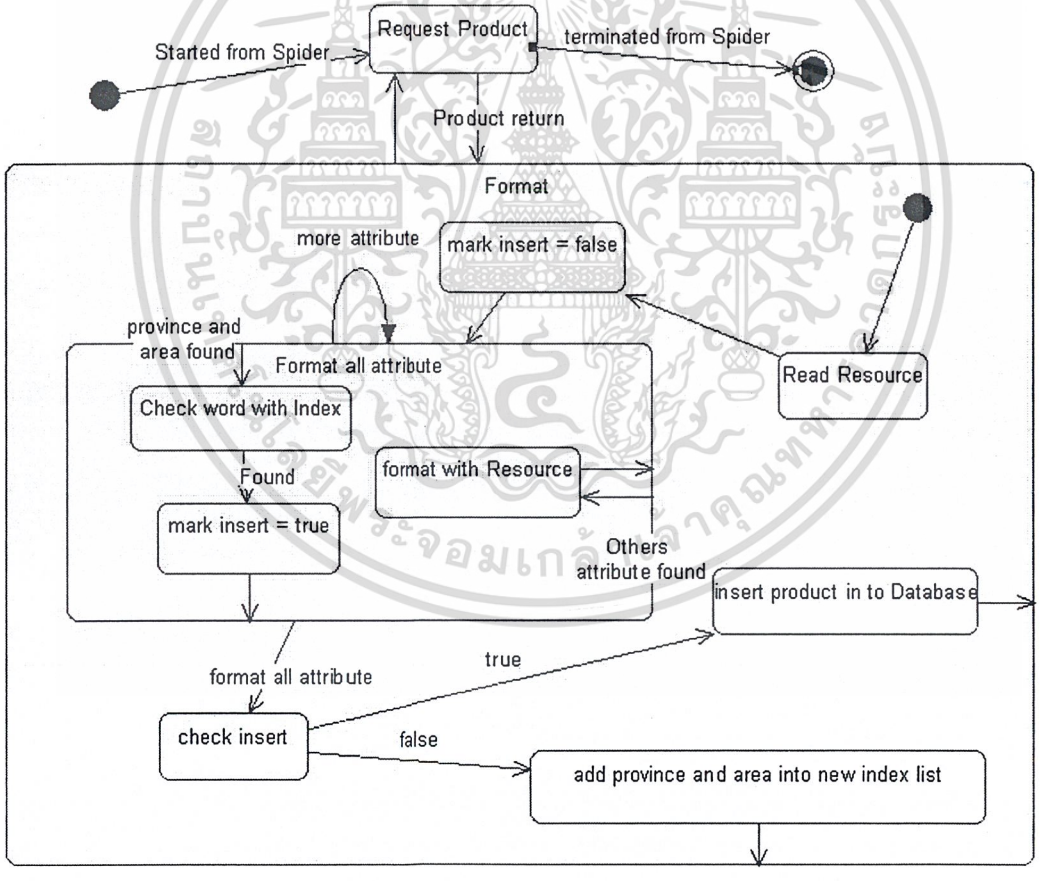
Wrapper เป็นเธรดของสไปเดอร์ตัวหนึ่ง ซึ่งจะมีหน้าที่ในการดึงข้อมูลจากเว็บไซต์ผ่าน URL ที่อ่านมาจาก Resource และทำการสร้าง Product ขึ้นมาจากข้อมูลที่ดึงมาได้นั้น

คำอธิบาย State

- **Request for Resource** Wrapper จะทำการตรวจสอบตลอดเวลาว่ามี Resource ที่ยังไม่ได้ทำการดึงข้อมูลหรือไม่ จนกว่าจะไม่มี Resource เหลือแล้วจึงจะจบการทำงานของ Wrapper
- **Wrap** แต่ถ้ายังมี Resource ที่ยังไม่ได้ดึงข้อมูลนั้น ก็จะทำการ Wrap ข้อมูลที่ได้มานั้นโดยมีกฎจาก Resource ที่ได้มา โดยมีการทำงานดังนี้

- **get url from Resource** Wrapper จะดึง URL จาก Resource ที่ได้มาจาก Spider
- **get HTML result from URL** ทำการดึงข้อมูลมาจากเว็บไซต์ผ่านทาง URL ก็จะได้ผลลัพธ์มาเป็น HTML
- **save result to Disk** จากนั้นทำการเก็บผลลัพธ์ที่ได้ลงเนื้อที่ในดิสก์เพื่อให้ MyParser ทำการ Parse ต่อไป
- **parse with Myparser** Myparser ตัวนี้ก็จะทำการ Parse ข้อมูลที่เป็นผลลัพธ์ที่ได้มาจาก Wrapper ตัวนี้ในขณะนี้ เพื่อทำการหา Product และแอตทริบิวต์ที่อยู่ใน Product
- **construct Product** เมื่อทำการดึงข้อมูลที่เป็นแอตทริบิวต์ของ Product เท่าที่จะทำการดึงมาได้นั้นแล้ว ก็จะทำการรวบรวมแอตทริบิวต์ที่ได้เป็น Product ส่งไปให้สไปเดอร์อีกที

4.3.4.3 Formatter



รูปภาพ 4-9 State Diagram ของ Formatter

Formattor เป็นเธรดที่ทำหน้าที่นำ Product ที่ Wrapper ดึงมาจากเว็บไซต์ต่างๆ แล้วนำมาทำการจัดรูปแบบให้เป็นข้อมูลที่มีรูปแบบที่สามารถนำไปใช้กับระบบได้

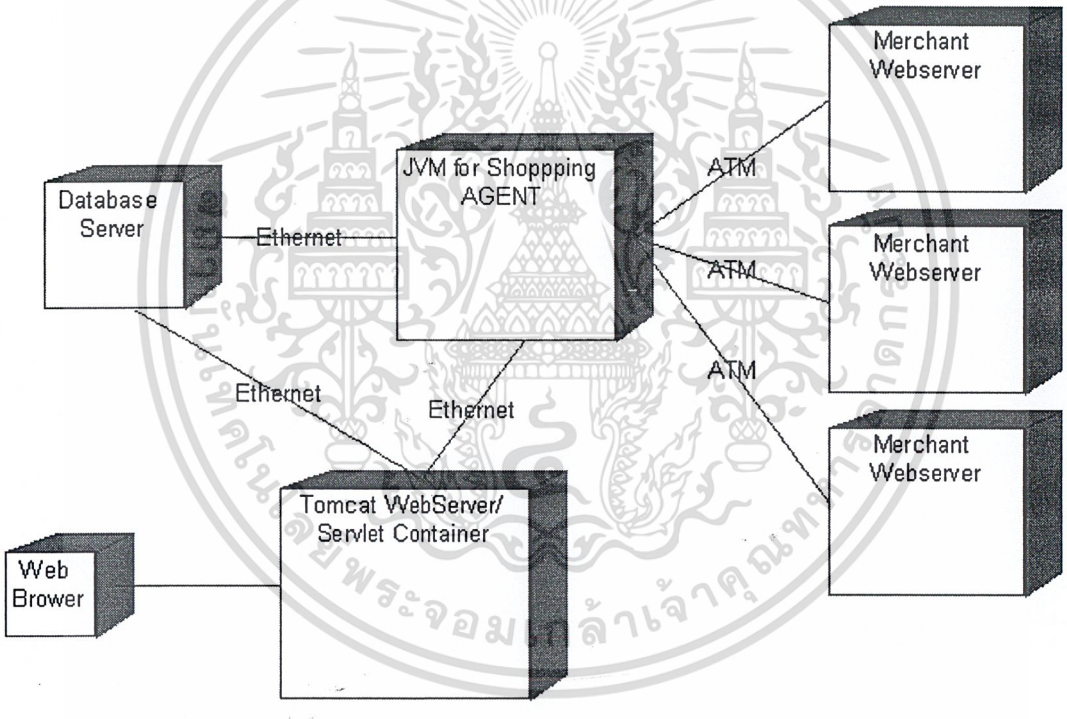
คำอธิบาย State

- **Request Product** เมื่อ Wrapper ทำการดึงข้อมูลของสินทรัพย์ที่ได้มาจากเว็บไซต์ต่างๆ แล้วก็จะทำให้ข้อมูลนั้นอยู่ในรูปของ Product เพื่อให้ Formattor ทำการเรียกข้อ Product นั้นมาทำการ Format Formattor จะไม่มีการหยุดทำงาน เมื่อใดที่มี Product เพิ่มขึ้นมา Formattor ก็จะทำการเรียกขอ Product มาเรื่อยๆ ถึงแม้ว่าจะไม่มี Product เพิ่มขึ้น Formattor จะหยุดทำงานก็ต่อเมื่อถูกทำลายจากสไปเดอร์
- **Format**
  - **Read Resource** Formattor จะอ่าน Resource ว่า Product นี้มีการ Format อย่างไร ก่อนที่จะเอากฎเหล่านั้นไปทำการ format attribute หมดทั้ง Product
  - **Mark insert = false** ทำการ mark Product ก่อนที่จะทำการ Format ให้ไม่สามารถนำลงดาต้าเบสได้
  - **Format all attribute** Formattor ทำการ Formattor ทุก Attribute โดยอาศัย Resource ที่ได้อ่านมาแล้วในตอนต้นใช้ในการ Format
    - **Check word with index** ถ้าแอตทริบิวต์ที่ทำการ Format เป็น Province (จังหวัด) หรือว่า Area (เขต) ก็จะต้องทำการตรวจสอบก่อนว่าข้อมูลที่เป็นตัวบ่งชี้ว่าสินทรัพย์นั้นอยู่ในจังหวัดหรือเขตใดนั้นมีอยู่ในดาต้าเบสหรือไม่ ถ้าไม่อาจหมายความว่าคำๆ นั้นอาจจะเป็นข้อมูลที่ผิดพลาด หรือไม่ก็อาจจะเป็นคำที่เป็นตัวบ่งชี้เหมือนกันแต่ว่าสะกดในอีกรูปแบบหนึ่ง สะกดด้วยอีกภาษาหนึ่งหรือว่าเป็นตัวย่อที่มีความหมายในอย่างเดียวกันกับค่าที่เป็นคีย์เวิร์ดในดาต้าเบส ตัวอย่างเช่นคำที่เป็นตัวบ่งชี้ว่า Product นั้นตั้งอยู่ในจังหวัดอยู่ในกรุงเทพสามารถเขียนได้หลายแบบ เช่น “กรุงเทพ” “กทม.” “Bangkok” “Bkk” เป็นต้นจึงต้องมีตารางที่เป็นตัวบอกว่า คำเหล่านี้มีความหมายว่า กรุงเทพ ซึ่งเป็นจังหวัดในประเทศไทย ดังนั้นต้องทำการตรวจสอบก่อนที่จะทำการ insert ลงดาต้าเบสเพื่อไม่ให้ข้อมูลที่ไม่ถูกต้องเข้าไปอยู่ในดาต้าเบสได้ และอาจเป็นเหตุให้เกิดข้อผิดพลาดกับส่วนอื่นของระบบที่ใช้ข้อมูลส่วนนั้นได้
    - **Format with Resource** ถ้าแอตทริบิวต์ไม่ได้เป็น Province หรือว่า Area แล้วให้ทำการ Format ข้อมูลที่เป็น Attribute ทั่วไปตามกฎของที่ได้อ่านมาแล้วจาก Resource ตามปกติ
    - **Mark insert = true** ถ้าค่าของแอตทริบิวต์ Province และ Area นั้นมีอยู่ในตารางแล้วก็จะทำการ mark เอาไว้ว่าข้อมูลนี้สามารถทำการนำไปลงดาต้าเบสได้แล้ว
  - **Check insert** หลังจากที่ทำการ Format ข้อมูลครบทุกแอตทริบิวต์แล้วนั้น ก็จะทำการตรวจสอบว่าข้อมูลของสินทรัพย์ที่ทำการ Format แล้วสามารถทำการนำลงดาต้าเบสได้หรือไม่ ถ้าได้ค่า insert จะเป็น true ตรงกันข้ามถ้าค่าเป็น false แสดงว่าไม่สามารถ insert ลงดาต้าเบสได้

- **Insert Product into Database** ถ้าค่า Insert เป็น true แสดงว่าแอตทริบิวต์ Province และ Area นั้นมีค่าอยู่ในดาต้าเบสแล้วจึงทำการ insert Product นั้นลง Database
- **Add province and area into new word list** ถ้าแอตทริบิวต์ Province หรือ Area นั้นไม่มีในดาต้าเบส ก็ให้ทำการใส่ค่าแอตทริบิวต์ตัวนั้นไปใน list เพื่อให้ผู้ดูแลระบบเข้ามาทำการตรวจสอบว่าคำใหม่ที่พบควรมีความหมายอย่างไร

จากนั้น Formattor ก็จะทำการขอ Product ตัวใหม่จากสไปเดอร์เรื่อยๆ ไม่มีหยุดจนกว่าจะถูกยกเลิกจาก Spider

4.3.5 Deployment Diagram



รูปภาพ 4-10 Deployment Diagram ของระบบ

ในการอิมพลีเมนต์ระบบจริงได้ใช้ซอฟต์แวร์,ฮาร์ดแวร์และเน็ตเวิร์กดังนี้

ซอฟต์แวร์

- Webserver/Servlet Container ได้เลือกใช้ Jakarta Tomcat 3.2.3 ที่เป็นรุ่นของวินโดวส์สำหรับรองรับการทำงานของ Servlet และไฟล์ HTML

- คาต้าเบสเซิร์ฟเวอร์ได้เลือกใช้ Oracle 8i สำหรับเก็บข้อมูลของสมาชิก ข้อมูลสินทรัพย์ที่ได้มาจากการประกาศของสมาชิกและการค้นหาจากเว็บไซต์ต่างๆ โดยช็อปปิ้งเอเจนต์
- JVM ของ JDK รุ่น 1.3 สำหรับรองรับการทำงานของช็อปปิ้งเอเจนต์ที่พัฒนาโดยภาษาจาวา

ฮาร์ดแวร์

- Intel Celeron 300 MHz
- Memory 196 MB
- Harddisk 3.0 GB

เน็ตเวิร์ก

- Ethernet LAN 10 Mbps

4.4 การออกแบบอินเทอร์เฟส

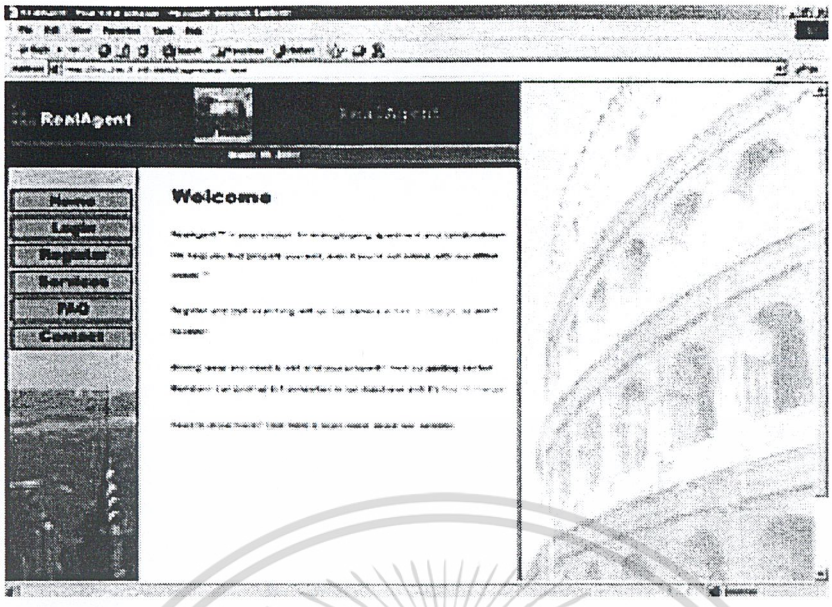
การออกแบบยูเซอร์อินเทอร์เฟสจะเน้นความเข้ากันได้ และการใช้งานง่ายเป็นหลัก โดยเรื่องของความสวยงามเป็นเรื่องที่คำนึงถึงเป็นเรื่องสุดท้าย ซึ่งมีหลักในการออกแบบโดยคร่าวๆ ดังนี้

- ใช้ HTML โดยไม่มีการใช้ปลั๊กอินใดๆเสริม เพื่อเป็นการลดปัญหาความเข้ากันได้ของไคลเอ็นต์
- หลีกเลี่ยงการใช้ VBScript สำหรับไคลเอ็นต์สคริปต์เนื่องจากมีเพียงเว็บบราวเซอร์ของไมโครซอฟท์เท่านั้นที่สนับสนุน โดยการใช้จาวาสคริปต์เป็นไคลเอ็นต์สคริปต์เพื่อให้เว็บบราวเซอร์ทั่วไปสามารถทำงานได้อย่างถูกต้อง
- ออกแบบที่ความละเอียดของหน้าจอ 640 x480 พิกเซล เพื่อให้ไคลเอ็นต์ส่วนมากสามารถเข้าชมเว็บไซต์ได้เหมือนกัน
- หลีกเลี่ยงการใช้เฟรมเป็นการใช้ตารางแทน เนื่องจากเว็บบราวเซอร์บางรุ่นไม่สนับสนุนการใช้เฟรม

4.4.1 รายละเอียดของแต่ละหน้าเว็บเพจ

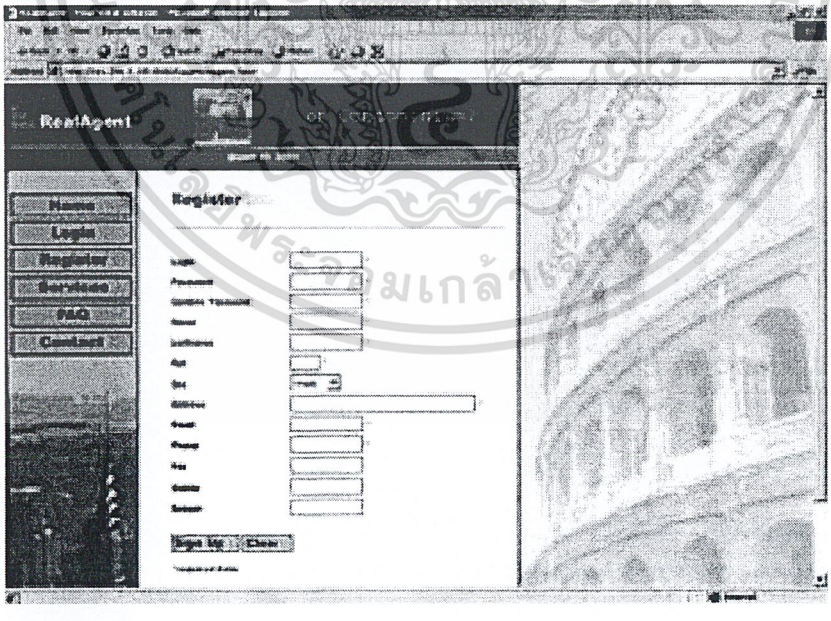
ในหน้าแรกของเว็บเพจจะมีลิงก์ไปยัง

1. หน้าล็อกอินเพื่อเข้าสู่ระบบ
2. หน้าสมัครสมาชิก
3. หน้าแสดงรายละเอียดบริการ
4. หน้า FAQ
5. หน้าการติดต่อกับผู้ดูแลระบบ



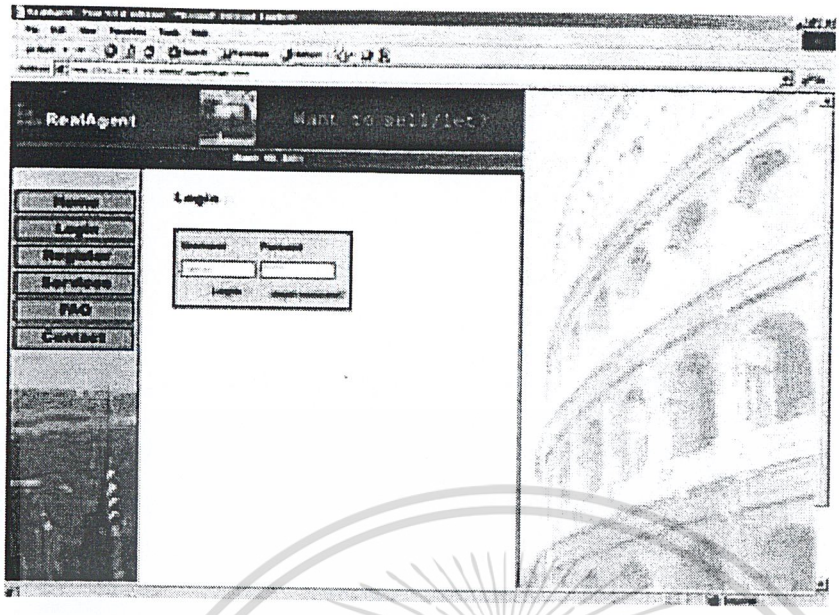
รูปที่ 4-11 หน้าแรกของเว็บเพจ

ในหน้าลงทะเบียนจะมีการตรวจสอบการป้อนข้อมูลของผู้ใช้ที่ต้องการจะลงทะเบียนด้วยการใช้จาวาสคริปต์ เช่น การตรวจสอบว่าในช่องพาสเวิร์ดและช่องยืนยันพาสเวิร์ดจะต้องมีค่าเท่ากัน หรือ ตรวจสอบในช่องอายุว่าผู้ใช้ป้อนตัวเลขเท่านั้น



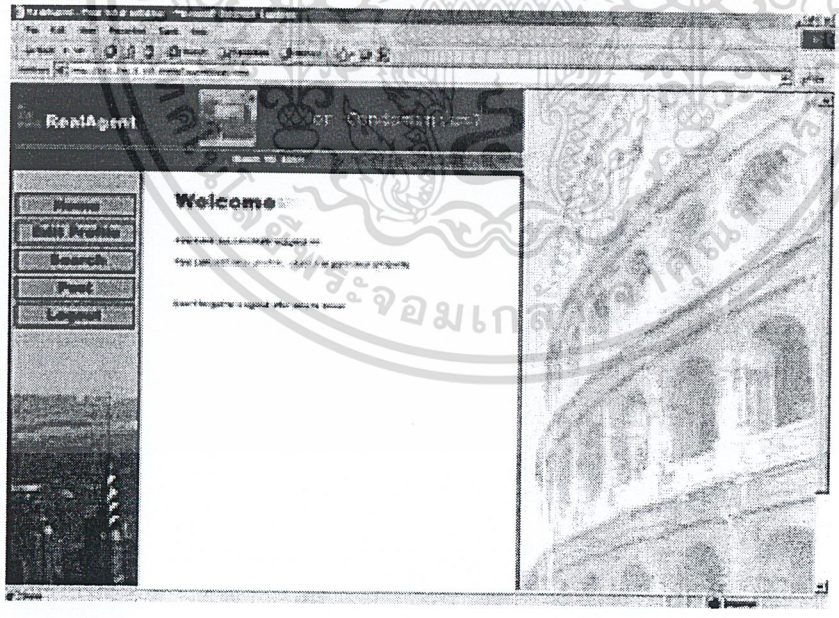
รูปที่ 4-12หน้าลงทะเบียน

หลังจากทำการลงทะเบียนแล้ว สมาชิกจะถูกนำมาที่หน้าการล็อกอินเข้าสู่ระบบ ซึ่งจะมีลิงก์สำหรับผู้ลืมรหัสผ่านให้กรอก username เพื่อทำการกู้รหัสผ่านได้โดยทางระบบจะส่งไปให้ทางอีเมล์



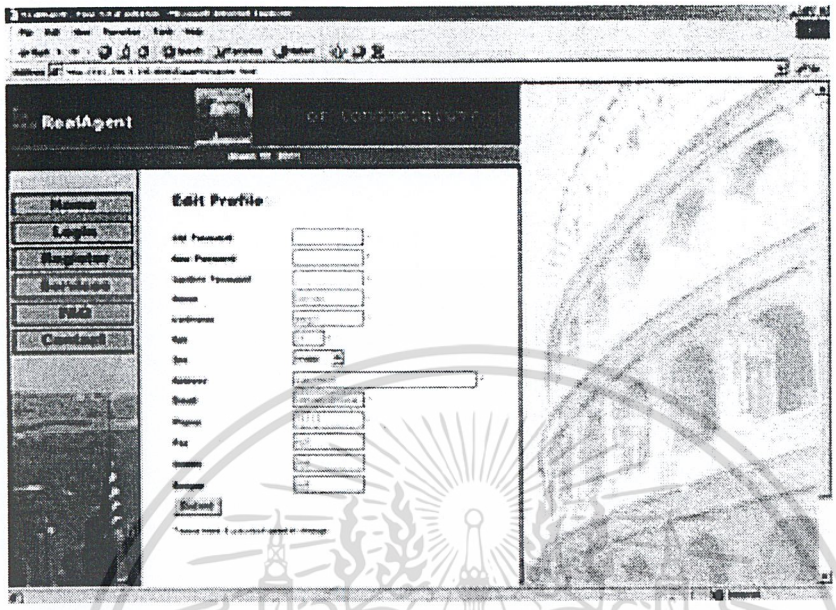
รูปที่ 4-13หน้าการล็อกอิน

หลังจากล็อกอินสำเร็จแล้ว จะนำสมาชิกมาที่หน้าหลัก ซึ่งสมาชิกสามารถแก้ไขข้อมูลส่วนตัว, ค้นหาสินทรัพย์, ประกาศขาย/เช่าสินทรัพย์และออกจากระบบได้



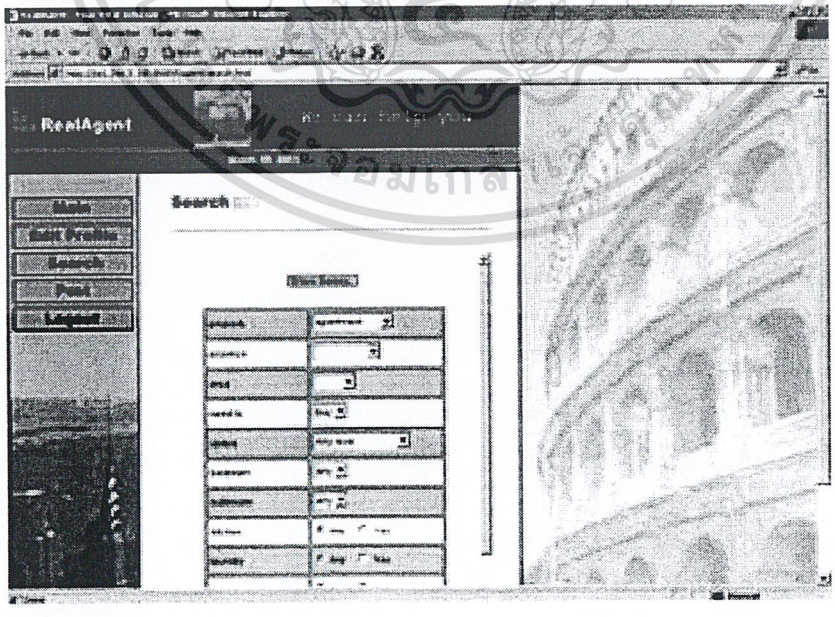
รูปที่ 4-14 หน้าหลักหลังจากการล็อกอินสำเร็จ

ในกรณีที่ผู้ใช้เลือกที่จะแก้ไขข้อมูลส่วนตัวก็สามารถเข้ามาที่หน้า Edit Profile ได้ โดยจะต้องมีการยืนยันรหัสผ่านเก่าก่อนจึงจะสามารถแก้ไขข้อมูลได้ โดยถ้าไม่ต้องการจะเปลี่ยนรหัสผ่านก็เพียงแค่เว้นว่างไว้ในฟิลด์สำหรับเปลี่ยนรหัสผ่าน



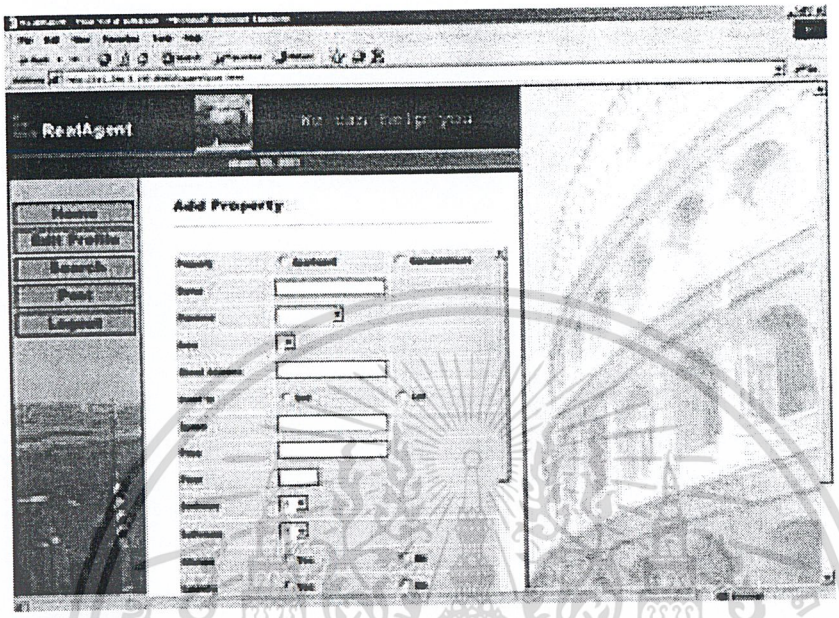
รูปที่ 4-15 หน้าสำหรับแก้ไขข้อมูลสมาชิก

ในกรณีที่สมาชิกต้องการที่จะค้นหาสินทรัพย์ก็จะต้องเข้ามาที่หน้า Search ซึ่งในหน้านี้สมาชิกจะต้องกำหนดความต้องการในทุกๆฟิลด์โดยจะมีการตรวจสอบความครบถ้วนของอินพุทจากสมาชิกโดยใช้จาวาสคริปต์



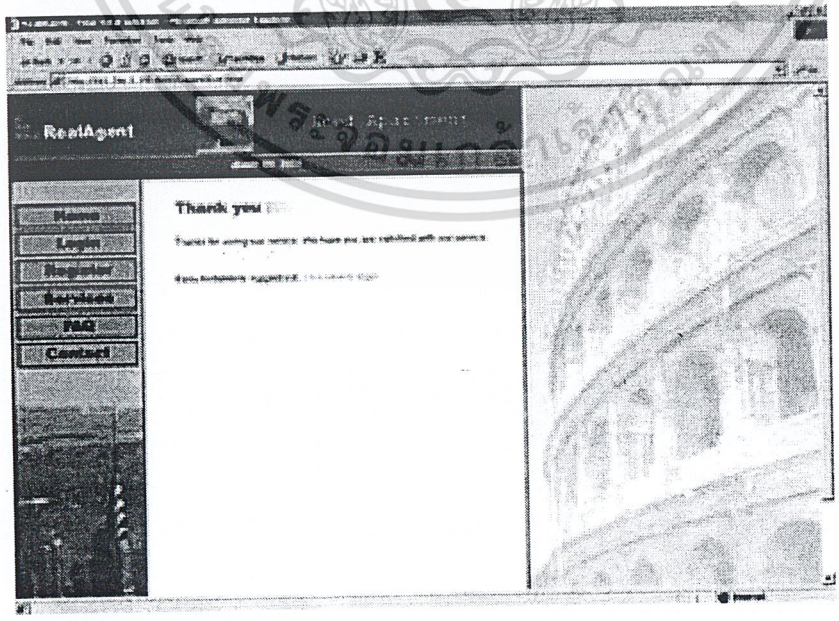
รูปที่ 4-16 หน้าสำหรับการกำหนดความต้องการเพื่อการค้นหา

ในกรณีที่สมาชิกต้องการจะประกาศขาย/เช่าสินทรัพย์กับทางเว็บไซต์ จะต้องเข้ามาที่หน้า Post เพื่อทำการกรอกรายละเอียดของสินทรัพย์ที่ต้องการจะประกาศ เมื่อผู้ใช้ทำการคลิกปุ่ม Submit ทางเซิร์ฟเวอร์จะตรวจสอบว่าได้มีการประกาศรายการนี้แล้วหรือไม่ ถ้าไม่ก็จะเก็บลงดาตาเบส จากนั้นจะพาสมาชิกไปยังหน้าหลัก



รูปที่ 4-17 หน้าการประกาศขาย/เช่าสินทรัพย์

เมื่อสมาชิกต้องการจะออกจากการใช้ระบบเพียงแค่คลิกที่ Logout ทางระบบจะนำมาที่หน้า Thank you โดยในหน้านี้จะมีลิงก์สำหรับการกลับไปล็อกอินในกรณีที่ไม่ได้ตั้งใจออกจากระบบ



รูปที่ 4-18 หน้าขอบคุณที่ใช้ระบบช็อปปิ้งเอเจนต์

บทที่ 5 ทรัพยากรและเทคโนโลยีที่ใช้ในระบบ

5.1 ทรัพยากรที่ใช้ในระบบ

5.1.1 Jakarta Tomcat

Tomcat เป็นเว็บเซิร์ฟเวอร์ที่พัฒนาโดย Apache ภายใต้โครงการจาการ์ตา ซึ่งมีให้ดาว์นโหลดฟรีที่ www.jakarta.apache.org โดยโครงงานนี้ใช้ Tomcat 3.2.1 ซึ่งเป็นรุ่นล่าสุดที่ทำงานบนปฏิบัติการวินโดวส์ 2000 ซึ่ง Tomcat 3.2.1 สนับสนุนทั้ง Java Servlet 2.2 และ Java Server Pages 1.1

5.1.2 JVM (Java Virtual Machine)

ระบบชื่อปปิ้งเอเจนต์ใช้ JDK 1.3 สำหรับการคอมไพล์และรันชื่อปปิ้งเอเจนต์บน JVM ที่มาพร้อมกับ JDK 1.3

5.1.3 Oracle 8i

เป็น DBMS ที่ใช้ในการเก็บฐานข้อมูลต่างๆในระบบ

5.2 เทคโนโลยีที่ใช้ในระบบ

5.2.1 Java HTML Parser

เป็น API ที่มาพร้อมกับ JDK โดยอยู่ในแพกเก็จ javax.swing.text.html.parser สำหรับการ parse ข้อมูลจากเอกสาร HTML โดยเฟรมเวิร์คของ Java HTML Parser เป็นแบบอีเวนต์

5.2.2 Java Servlet

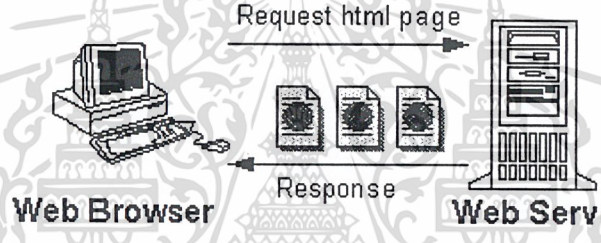
Java Servlet เป็นโค๊ดที่สามารถเข้าถึงผ่านทางอินเทอร์เฟสพื้นฐานของเน็ตเวิร์กเหมือนกับ CGI บนเว็บไซต์ การใช้งาน Servlet ที่นิยมก็คือ การเป็นมิดเดิ้ลแวร์บริดจ์ (Middleware Bridge) ระหว่างเว็บบราวเซอร์และระบบฐานข้อมูลขององค์กร โดยบราวเซอร์จะติดต่อกับเว็บเซิร์ฟเวอร์แล้วจึงให้ Servlet ทำการติดต่อกับฐานข้อมูลแล้ว Query ข้อมูลที่ต้องการออกมาแล้วทำการส่งไปให้บราวเซอร์ โดยที่เว็บเซิร์ฟเวอร์ไม่จำเป็นที่จะต้องสนับสนุนการติดต่อฐานข้อมูล

Servlet จะทำงานภายใต้ Sandbox ที่เซิร์ฟเวอร์ซึ่งก็เหมือนกับจาวาแอพเพล็ตบนไคลเอ็นต์ โดย Servlet จะอยู่บนเว็บเซิร์ฟเวอร์เลยหรือถูกดาว์นโหลดมาก็ได้เพียงแต่ Servlet ที่ถูกดาว์นโหลดมาจะถูกจำกัดการเข้าถึงทรัพยากรที่อาจเป็นอันตราย เช่น การเขียนไฟล์ เป็นต้น

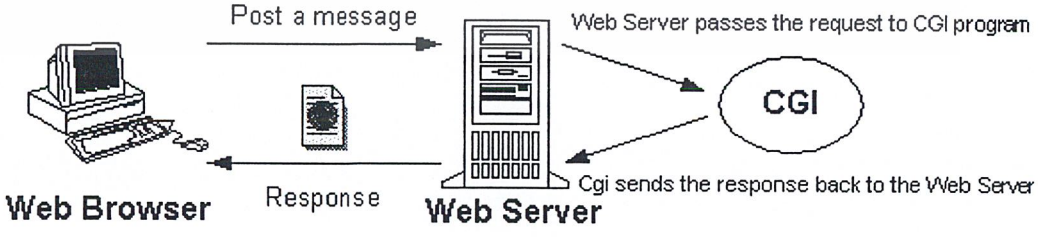
เปรียบเทียบระหว่าง CGI กับ Servlet

โมเดลของ CGI ทุกครั้งที่มีการร้องขอมาที่สคริปต์ CGI จะมีการสร้างโปรเซสระดับระบบปฏิบัติการ (OS-level process) เพื่อตอบสนองต่อการร้องขอ และเมื่อการตอบสนองสิ้นสุดโปรเซสก็จะถูกทำลาย เซิร์ฟเวอร์บางตัวสนับสนุนให้โปรเซสของ CGI มีชีวิตยาวนานขึ้น (Long-running CGI) สามารถรองรับการทำงานได้หลายการร้องขอ กล่าวคือจะไม่ทำลายโปรเซสเมื่อการทำงานสิ้นสุดลง แต่ถ้ามีการร้องขอเข้ามาพร้อมๆกัน ก็จะต้องมีการสร้างโปรเซสขึ้นมาเพื่อรองรับการร้องขอ

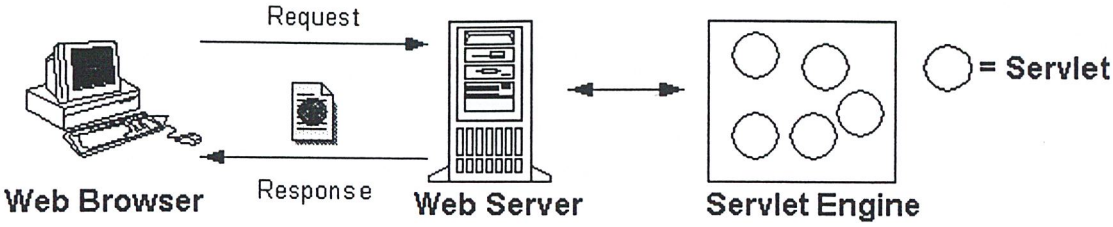
การทำงานของ Servlet ก็จะคล้ายๆกันเพียงแต่ว่าเปลี่ยนจากการสร้างโปรเซส เป็นการสร้างเธรด ซึ่งมีโอเวอร์เฮดในการ สร้าง หยุด หรือจัดการน้อยกว่าโปรเซส ทำให้มีประสิทธิภาพในการทำงานสูงขึ้น ข้อได้เปรียบของ Servlet ที่มีเหนือ CGI ก็คือ หนึ่งอินส์แตนซ์จะรองรับหลายการร้องขอ, สามารถใช้หน่วยความจำร่วม (Shared memory ของตัวแปรคลาสและตัวแปรอินส์แตนซ์) สำหรับการเก็บสถานะสำหรับการจัดการ Session ได้ ซึ่ง CGI จะต้องทำโดยการเขียนลงไฟล์ หรือ ใช้หน่วยความจำร่วมของระบบปฏิบัติการ (OS Shared Memory) ซึ่งวิธีหลังจะขึ้นอยู่กับแพลตฟอร์ม



รูปภาพ 5-1 สถาปัตยกรรมแบบเริ่มแรกของอินเทอร์เน็ต



รูปภาพ 5-2 สถาปัตยกรรมที่ มีการใช้ CGI



รูปภาพ 5-3 สถาปัตยกรรมของ Java Servlet

เปรียบเทียบระหว่าง Servlet กับ Server plug-ins

เพื่อเป็นการแก้ปัญหาการสร้างหลายโปรเซสของ CGI จึงมีการพัฒนา Server plug-ins ที่เพิ่มความสามารถของเว็บเซิร์ฟเวอร์โดยการลิงก์ไลบรารีสู่เซิร์ฟเวอร์โปรเซสแบบไดนามิกโดยเน็ตสเคปได้พัฒนาNSAPI ในขณะที่ไมโครซอฟท์ได้พัฒนา ISAPI

แต่เนื่องจากเซิร์ฟเวอร์ทั้ง 2 ชนิดดังกล่าวอิมพลิเมนต์บนภาษาทั่วไป เมื่อเซิร์ฟเวอร์ปลั๊กอินมีปัญหาอาจส่งผลให้เซิร์ฟเวอร์ล่มลงด้วย ข้อเสียอีกข้อของเซิร์ฟเวอร์ปลั๊กอินก็คือไม่มีความสามารถในการทำงานบนหลายแพลตฟอร์ม ในขณะที่การพัฒนาแอพพลิเคชันด้วยจาวาสามารถนำไปใช้บนแพลตฟอร์มใดก็ได้

การเรียกใช้ Servlet

โดยทั่วไปการเรียกใช้ Servlet จะมี 2 วิธี ได้แก่

1. *การเรียกโดยตรงผ่านทาง HTTP*

   วิธีนี้เป็นวิธีที่นิยมใช้กันมากที่สุด โดยการร้องขอผ่าน URL ซึ่งเหมือนกับการเรียกใช้ CGI โดยมีไวยกรณ์ในการเรียกใช้ดังนี้ http://hostname:port/servlet/ServletName

   เมื่อมีการร้องขอเข้ามา เว็บเซิร์ฟเวอร์จะตรวจว่ามีอินส์แตนซ์ของ Servlet นี้หรือไม่ ถ้ามี การร้อง ขอก็จะถูกส่งต่อไปยัง Servlet แต่ถ้ายังไม่มี จะทำการโหลด Servlet และสร้างอินส์แตนซ์ขึ้นมาเพื่อรองรับการร้องขอ

2. *การเรียกผ่าน Server-side include*

   Server-side include (SSI) ถูกนำมาใช้ตั้งแต่ยุคแรกเริ่มของเว็บเซิร์ฟเวอร์ โดยสนับสนุนการสร้าง HTML เมื่อมีการร้องขอ โดย HTML นี้จะมีแท็กพิเศษที่เว็บเซิร์ฟเวอร์จะแทนที่ด้วยข้อมูลที่ถูกสร้างขึ้นแบบไดนามิกแล้วจึงส่งไปยังไคลเอ็นต์ การเรียก Servlet จะทำโดยการใช้แท็ก <servlet> ดังตัวอย่างนี้

```
<html>
...
<servlet
        name= "ServletName"
        code= "ServletClass"
        codebase= "ssi/"
        initParam1= "val1"
        ...>
    <param name= "requestParam1" value= "val3">
    ...
</servlet>
...
</html>
```

การจะแยกแยะระหว่าง HTML ธรรมดากับ HTML ที่มีการเรียก Servlet ทำได้โดยใช้นามสกุล ".shtml" เมื่อเว็บเซิร์ฟเวอร์ได้รับการร้องขอมาที่ไฟล์ ".shtml" จะมีการอ่านไฟล์นี้เพื่อหาแท็กพิเศษ เมื่อพบแท็ก Serlet ก็จะมีการเรียกใช้ Servlet ที่ระบุและผลลัพทธ์ที่ได้จาก Servlet ก็จะถูกแทรกเข้าไปแทนแท็ก Servlet แล้วจึงส่งต่อไปยังไคลเอ็นต์

Java Servlet เป็น API สำหรับการสร้างเว็บเพจที่เซิร์ฟเวอร์โดยจะต้องทำงานภายใต้ Servlet Container เช่น Java Web Server หรือ Jakarta Tomcat เป็นต้น หากจะวิเคราะห์ให้ดีแล้ว Java Servlet ก็จะคล้ายกับ CGI (Common Gateway Interface) เพียงแต่ว่ามีข้อได้เปรียบที่ Java Servlet ทำงานเป็น 1 โปรเซสแล้วทำการสร้างเธรดขึ้นมารองรับการร้องขอจากเว็บบราวเซอร์ ในขณะที่ CGI จะทำการสร้างโปรเซสใหม่ขึ้นทุกครั้งที่มีการร้องขอจากเว็บบราวเซอร์ ซึ่งเป็นวิธีการที่ใช้ทรัพยากรมากกว่าและยังมีประสิทธิภาพที่ต่ำกว่าอีกด้วย

5.2.3 JDBC

JDBC (Java DataBase connectivity) คือ API ของ Java โดยบริษัท Sun สำหรับการติดต่อกับ DMBS ซึ่งมีการทำงานคล้ายกับ ODBC ของ Microsoft

JDBC ได้แบ่งชนิดของไดร์เวอร์ (Driver) ออกเป็น 4 ชนิดได้แก่

- ไดร์เวอร์ชนิดที่ 1 (JDBC/ODBC bridge) จะใช้ bridge ในการติดต่อกับดาตาเบส โดยจะต้องมีการติดตั้ง bridge ลงบนทุกๆเครื่องไคลเอ็นต์ ทำให้ยากต่อการดูแลรักษา ทางบริษัทซันรวม JDBC-ODBC bridge เข้าไว้ใน JDK ด้วย ทำให้สามารถติดต่อฐานข้อมูลผ่านทาง ODBC ซึ่งเป็นมาตรฐานของไมโครซอฟท์ได้ ข้อดีของการใช้ bridge ก็คือ DBMS ส่วน

ใหญ่จะรองรับ ODBC อย่างไรก็ตามการใช้ ODBC นั้นไม่สามารถใช้งานได้ครบทุกฟังก์ชันของ API ของ DBMS ซึ่งเป็นผลเสียในแง่ของประสิทธิภาพและอาจเกิดปัญหาในเธรดได้

- ไดร์เวอร์ชนิดที่ 2 (Native-API, partly Java driver) เป็น Native API ที่พัฒนาโดย C หรือ C++ ซึ่งมีประสิทธิภาพสูงและสามารถใช้งานได้ครบทุกฟังก์ชันของ DBMS แต่ว่าจะต้องมีไดร์เวอร์สำหรับแต่ละแพลตฟอร์ม โดยบริษัทผู้ผลิต DBMS จะเป็นผู้แจกจ่ายไดร์เวอร์ชนิดนี้ ทำให้ต้องใช้ไดร์เวอร์เฉพาะสำหรับแต่ละDBMS
- ไดร์เวอร์ชนิดที่ 3 (Network-protocol, all-Java driver) เป็น Native API ที่ใช้เน็ตเวิร์กโปรโตคอลในการสื่อสารกับมิดเดิ้ลแวร์เซิร์ฟเวอร์ (Middleware Server) ที่ติดต่อกับดาตาเบส เนื่องจากไดร์เวอร์ชนิดที่ 3 ไม่ขึ้นอยู่กับบริษัทผู้ผลิตจึงต้องการแค่ไดร์เวอร์เดียวเท่านั้นไม่ว่าจะต้องติดต่อกับดาต้าเบสเซิร์ฟเวอร์ที่แตกต่างกันกี่ชนิดก็ตาม แต่ก็ยังคงต้องการไดร์เวอร์ที่ตรงกับแพลตฟอร์มของเครื่องไคลเอ็นต์
- ไดร์เวอร์ชนิดที่ 4 (Native-protocol, all-Java driver) เป็นไดร์เวอร์ที่เป็นจาวาแท้ๆ ซึ่งติดต่อโดยตรงกับดาตาเบส ทำให้ต้องการไดร์เวอร์ที่ตรงกับดาตาเบส แต่เนื่องจากไดร์เวอร์ชนิดนี้เป็นจาวาทำให้ไม่ขึ้นกับแพลตฟอร์มของไคลเอ็นต์

5.2.4 Java API สำหรับการ parse XML

ในขณะนี้มีอยู่ 3 API หลักๆ ได้แก่ SAX, DOM และ JDOM ที่เป็นที่นิยมอย่างแพร่หลาย ซึ่งแต่ละ API ก็มีหลายรุ่นด้วยกัน

SAX (Simple API for XML)

SAX มีเฟรมเวิร์คแบบอีเวนต์ สำหรับการ parse XML โดยในกระบวนการ parse จะอ่าน XML และแบ่งออกเป็นหน่วยย่อยๆที่สามารถนำไปใช้ได้ SAX จะกำหนด event ที่จะสามารถเกิดขึ้นได้ โดยกำหนดอินเทอร์เฟส org.xml.sax.ContentHandler ที่กำหนดเมธอดต่างๆ เช่น startDocument() และ endDocument โดยจะต้องทำการอิมพลีเมนต์อินเทอร์เฟสนี้เพื่อที่จะสามารถควบคุมกระบวนการ parse

ระหว่างการทำงานของ SAX เมื่อ parser ทำการ parse แล้วเจออิลิเมนต์ ต่างๆ จะมีการ callback ให้ เมธอดที่กำหนดไว้ในอินเทอร์เฟสทำงาน โดยการทำงานของ SAX จะเป็นแบบ sequential กล่าวคือจะอ่าน XML เรียงจากหัวไปท้ายไฟล์ นักพัฒนาไม่สามาถย้อนกลับไปส่วนที่ทำการ parse ผ่านไปแล้วได้

มักมีการสับสนว่า SAX คือ XML parser ซึ่งจริงๆแล้ว SAX เป็นเฟรมเวิร์คสำหรับ parser และกำหนดอีเวนต์ต่างๆที่จะเกิดในระหว่างการ parse กล่าวคือจะต้องมี parser มาเชื่อมต่อกับ SAX เพื่อที่จะทำการ parse ข้อมูล

ข้อดีของ SAX

- ใช้ทรัพยากรน้อย เพราะว่าไม่ต้องเก็บอะไรไว้ในหน่วยความจำ

- ทำงานเร็วเพราะไม่ต้องอ่านทั้งไฟล์ก่อนที่จะเริ่มทำงานได้

ข้อเสียของ SAX

- เนื่องจาก SAX เป็นการทำงานแบบ sequential ทำให้ไม่สามารถเลือกอ่านข้อมูลตามต้องการได้ เช่น ไม่สามารถย้อนกลับไปอ่านข้อมูลที่ผ่านไปแล้วได้ หรือ ไม่สามารถข้ามไปอ่านข้อมูลข้างหน้าได้
- สามารถอ่าน XML ได้เท่านั้น ไม่สามารถสร้างหรือแก้ไขได้

DOM (Document Object Model)

DOM เป็น API สำหรับ Document Object Model ที่กำหนดโดย W3C ที่มี 3 ระดับ ในขณะที่ SAX ทำให้นักพัฒนาสามารถเข้าถึงข้อมูลได้ แต่ DOM มีความสามารถในการแก้ไขข้อมูลด้วย โดย DOM นำเสนอโครงสร้างของ XML โดยการใช้ Tree ที่นักพัฒนาส่วนใหญ่คุ้นเคย

ในการทำงานของ DOM จะทำการอ่าน XML ทั้งเอกสารแล้วนำไปเก็บในหน่วยความจำแล้วเก็บข้อมูลไว้ในโนดทำให้การเข้าถึงส่วนต่างๆของเอกสารสามารถทำได้อย่างรวดเร็ว

ข้อเสียของ DOM ก็คือ เมื่อเอกสาร XML มีขนาดใหญ่ หน่วยความจำที่ใช้ในการเก็บ DOM Tree ก็จะต้องเพิ่มขึ้นด้วย ซึ่งอาจจะทำให้ทำงานได้ช้าหรือระบบล่มได้

ข้อดีของ DOM

- มีโครงสร้างเป็น Tree ทำให้สามารถเข้าถึงข้อมูลส่วนใดก็ได้
- สามารถสร้างหรือแก้ไข XML ได้ ไม่ว่าจะเป็นการเพิ่ม หรือลบโนดออก

ข้อเสียของ DOM

- ใช้ทรัพยากรมากเพราะว่าจะต้องเก็บทั้ง DOM Tree ในหน่วยความจำ จึงไม่เหมาะกับการทำงานกับ XML ขนาดใหญ่

JDOM

เป็น API ที่ใหม่ที่สุดใน API ทั้ง 3 นี้ โดย JDOM ถูกสร้างขึ้นมาแก้ปัญหาที่ SAX และ DOM มี และเพื่อเพิ่มประสิทธิภาพ โดย JDOM ยังสนับสนุนข้อมูลที่เป็นแบบไฮราคีใดๆก็ได้ และสามารถทำการ serialize ได้ง่ายเหมือนกับการสร้าง XML

ข้อดีของ JDOM

- มีโครงสร้างเป็นแบบ tree ทำให้สามรถเข้าถึงข้อมูลส่วนใดใน XML ก็ได้เช่นเดียวกับ DOM
- สามารถสร้างและแก้ไข XML ได้
- ใช้ทรัพยากรน้อยกว่า DOM

- มีประสิทธิภาพสูงเพราะว่าออกแบบสำหรับ Java เท่านั้น

ข้อเสียของ JDOM

- ใช้ทรัพยากรมากกว่า SAX
- ยังเป็นรุ่นทดสอบอยู่ ในรุ่นเต็มอาจจะมีการเปลี่ยนแปลงไปจากรุ่นทดสอบได้

	SAX	DOM	JDOM
การใช้ทรัพยากร	น้อย	มาก	ปานกลาง
อ่าน XML	Sequential	Tree	Tree
แก้ไข XML	ไม่ได้	ได้	ได้

ตาราง 5-1 เปรียบเทียบคุณสมบัติของ API ต่างๆ สำหรับการ parse XML

5.2.5 Javamail

เป็น API สำหรับการรับและส่งอีเมล์ของ Sun โดยรุ่นปัจจุบันได้พัฒนามาถึงรุ่นที่ 1.2 แล้ว โดย Javamail 1.2 สนับสนุนการติดต่อกับเมล์เซิร์ฟเวอร์หลายโปรโตคอลได้แก่ POP3, IMAP4 และ SMTP โดยการทำงานของ Javamail จะต้องใช้ Java Activation Framework ร่วมด้วย ซึ่งรุ่นล่าสุดที่ใช้ในโครงงานนี้คือรุ่นที่ 1.0.1

บทที่ 6 บทวิจารณ์และสรุป

6.1. บทวิจารณ์และสรุป

โครงงานนี้เป็นการศึกษา ออกแบบ และพัฒนาระบบช็อปปิ้งเอเจนต์ โดยมีสินค้าที่ใช้ในโครงงานนี้เป็นคอนโดมีเนียมและอพาร์ทเมนต์สำหรับการเช่าหรือซื้อ ซึ่งมีแหล่งข้อมูลหลักจากเอกสาร HTML ที่ได้จากเว็บไซต์อสังหาริมทรัพย์ในประเทศไทย ได้แก่ www.homedd.com, www.home4thai.com, www.thailandproperty.com เป็นต้น รวมถึงข้อมูลที่ได้จากสมาชิกเข้ามาประกาศไว้

แนวคิดหลักในโครงงานนี้ประกอบด้วย Information Retrieval และ Information Extraction ซึ่งส่วนของ Information Retrieval นั้นเป็นขั้นตอนในการหา HTML เว็บเพจที่มีข้อมูลของสินค้าซึ่งในที่นี้คือคอนโดมีเนียมและอพาร์ทเมนต์ เพื่อนำมาเข้ากระบวนการ Information Extraction ซึ่งก็คือการคัดเลือกสาระออกมาจากเอกสาร HTML โดยผลลัพธ์ที่ได้จะเป็นรายละเอียดของสินค้า ได้แก่ ชื่อของอาคาร, ที่อยู่ของอาคาร, ขนาดของห้อง และราคา

เนื่องจากแหล่งข้อมูลของโครงงานเป็นเอกสาร HTML ซึ่งจัดได้ว่าเป็นแหล่งข้อมูลแบบ Semi-Structured Data ที่สามารถทำกระบวนการ Information Extraction ได้ แต่กระนั้นก็ตาม ยังไม่เหมาะสมกับกระบวนการ Information Extraction เท่ากับข้อมูลแบบ Structured Data ทำให้การคัดเลือกข้อมูลออกจากแหล่งข้อมูลอาจมีความผิดพลาดได้บ้างและทางด้านการออกแบบระบบก็ทำได้ยุ่งยากกว่า

ข้อจำกัดของระบบในโครงงานนี้คือ ไม่สามารถคัดเลือกข้อมูลออกจากแหล่งข้อมูลที่เป็นเอกสารที่ไม่มีโครงสร้าง หรือ Free Text ได้ซึ่งโดยมากจะเป็นเอกสารที่สร้างโดยมือแทนที่จะเป็นการสร้างโดยการใช้ CGI หรือ Server Page ที่มีโครงสร้างมากกว่า แต่อย่างไรก็ตามช็อปปิ้งเอเจนต์ที่เป็นที่นิยมทุกตัวก็จะทำการค้นหาข้อมูลจากแหล่งข้อมูลแบบ Semi-Structured ทั้งสิ้น และเนื่องจากเหตุที่แหล่งข้อมูลไม่ใช่แหล่งข้อมูลชนิด Structured Data นี้เอง ทำให้ช็อปปิ้งเอเจนต์โดยมากยังไม่สามารถเรียนรู้การคัดเลือกข้อมูลด้วยตัวเองได้ ดังจะเห็นได้ว่าช็อปปิ้งเอเจนต์ที่เป็นที่นิยมจะต้องมีการโปรแกรมด้วยโปรแกรมเมอร์ด้วย

ทางด้านของแนวทางในการสร้างช็อปปิ้งเอเจนต์นั้น จะเป็นไปทางการใช้กฎในการคัดเลือกข้อมูลออกจากเว็บเพจ แล้วให้ช็อปปิ้งเอเจนต์ทำตามกฎในการคัดเลือกข้อมูลออกมา ซึ่งอาจจะมีเครื่องมือในการสร้างกฎให้กับช็อปปิ้งเอเจนต์ที่มีส่วนติดต่อกับผู้ใช้แบบ GUI ดังที่ mySimon กำลังใช้อยู่ รวมทั้งระบบของโครงงานนี้ด้วยที่ใช้ระบบ GUI ในการสร้างกฎให้กับช็อปปิ้งเอเจนต์

ข้อดีของการใช้กฎในการคัดเลือกข้อมูลออกมาจาก HTML คือ เป็นแนวทางที่ง่ายกว่าในทางด้านการออกแบบ และถ้าหากสร้างเครื่องมือสำหรับการสร้างกฎให้ใช้งานได้ง่ายก็จะทำให้ไม่จำเป็นต้องใช้โปรแกรมเมอร์ในการสร้างกฎให้กับช็อปปิ้งเอเจนต์ ทำให้สามารถรับมือกับการเพิ่มขึ้นกับจำนวนเว็บไซต์ของผู้ขายสินค้าบนอินเทอร์เน็ตได้

ข้อเสียของการใช้กฎในการคัดเลือกข้อมูลออกจาก HTML ก็คือ ถ้าเว็บไซต์ที่มีการสร้างกฎไปแล้วเกิดมีการเปลี่ยนแปลงรูปแบบการแสดงผล จะส่งผลให้การทำงานผิดพลาด โดยจะต้องทำการแก้ไขกฎในการคัดเลือกข้อมูลใหม่ ทุกครั้งที่เว็บไซต์มีการเปลี่ยนแปลงการแสดงผลนั่นเอง

ส่วนด้านของประสิทธิภาพนั้น ในโครงงานได้ใช้แนวทางของ Junglee ในการนำข้อมูลที่ผ่านกระบวนการ Information Retrieval และ Information Extraction มาเก็บไว้ในฐานข้อมูล ซึ่งจะลดปัญหาความล่าช้าในการค้นหาข้อมูลจากผู้ใช้ที่เกิดจากปัญหาของระบบเน็ตเวิร์กได้ โดยจะสามารถกำหนดเวลาให้ระบบไปเก็บข้อมูลจากเว็บไซต์ต่างๆ ในเวลาที่มีผู้ใช้น้อย แต่ปัญหาที่จะเกิดขึ้นก็คือ จะต้องมีขนาดของฐานข้อมูลที่ใหญ่ และปัญหาที่อาจจะเกิดได้คือ ข้อมูลจะไม่ทันสมัยเท่ากับแนวทางแบบการค้นหาแบบทันทีทันใดที่ mySimon และ Jango ใช้

สุดท้ายนี้ก็หวังว่าโครงงานนี้จะสามารถนำมาใช้งานได้จริงในอนาคตอันใกล้นี้ หรืออาจจะนำมาเป็นต้นแบบในการพัฒนาต่อไป

6.2. แนวทางในการปรับปรุงระบบ

เนื่องจากข้อจำกัดทางด้านทรัพยากรทำให้การพัฒนาระบบของโครงงานนี้ไม่สามารถพัฒนาระบบแบบกระจายออบเจ็กต์ได้ (Distributed Object) ดังนั้นเมื่อมีทรัพยากรที่เพียงพอก็ควรปรับเปลี่ยนโครงสร้างการทำงานของช็อปปิ้งเอเจนต์ให้เป็นการทำงานแบบการกระจายออบเจ็กต์ อิมพลีเมนต์โดยการใช้ JAVA RMI ซึ่งจะเป็นการช่วยเพิ่มประสิทธิภาพในการค้นหาและคัดเลือกสาระจากข้อมูลได้มากและรวดเร็วขึ้น ด้วยการให้ช็อปปิ้งเอเจนต์กระจายไปทำงานบนเซิร์ฟเวอร์หลายๆตัวและในแต่ละเซิร์ฟเวอร์ก็ยังทำงานหลายๆเธรด จะทำให้ระบบสามารถตอบสนองการร้องขอของผู้ใช้ได้รวดเร็วพอที่จะสามารถทำการค้นหาตามคำร้องขอจากเว็บไซต์ต่างๆได้แบบเรียลไทม์ ซึ่งจะมีข้อดีเหนือกว่าระบบปัจจุบันคือ ไม่จำเป็นจะต้องนำข้อมูลจากเว็บไซต์ต่างๆมากเก็บไว้ในดาตาเบสเซิร์ฟเวอร์ของระบบ และยังจะได้ข้อมูลที่ทันสมัยถูกต้องมากกว่าอีกด้วย

นอกจากการปรับเปลี่ยนที่ได้กล่าวไปแล้ว ในอนาคต XML จะเป็นมาตรฐานในการแลกเปลี่ยนข้อมูลซึ่งจะทำให้การออกแบบและอิมพลีเมนต์ช็อปปิ้งเอเจนต์ง่ายขึ้นมาก ไม่ว่าในกรณีที่เว็บเพจในอนาคตจะเป็นการส่งไฟล์ XML ที่ประกอบด้วยไสตล์ชีตมาฟอร์แมตการแสดงผลที่เว็บบราวเซอร์ของผู้ใช้ หรือว่าจะเป็นการเก็บข้อมูลเพื่อการนำมาใช้ระหว่างธุรกิจ (Business-to-Business) เพราะว่าข้อมูลจะมีโครงสร้างที่ดี ทำให้การที่จะคัดเลือกข้อมูลออกมาใช้ทำได้อย่างมีระบบและง่าย

ภาคผนวก ก

UML (Unified Modeling Language)

UML (Unified Modeling Language)

UML หรือ Unified Modeling Language เป็นภาษาในการสร้างโมเดลจำลอง (Abstract Model) ที่มีสัญลักษณ์ (Notation) ที่เป็นมาตรฐาน ซึ่งจะช่วยในส่วนของการวิเคราะห์ (Analysis) และออกแบบ (Design) ระบบแบบออบเจ็กต์ โอเรียนเท็ด (Object Oriented) ภาษา UML เกิดจากการพัฒนาร่วมมือกันของผู้นำเทคโนโลยีทางด้านวัตถุ 3 คน คือ Grady Booch , Ivar Jacobson และ Jim Rumbaugh และบริษัท Rational Software เพื่อพัฒนาโมเดลร่วมกัน ให้เกิดความเป็นหนึ่งเดียวในการพัฒนาระบบเชิงวัตถุ การออกแบบระบบแบบออบเจ็กต์ โอเรียนเท็ด จะทำให้เราสามารถวิเคราะห์ส่วนประกอบโดยรวมของระบบ และสามารถรองรับภาษาที่จะใช้พัฒนาได้หลายภาษา รวมทั้งหลีกเลี่ยงความซ้ำซ้อน และลดความเฉพาะเจาะจงลงด้วย

ภาษา UML ประกอบไปด้วย 4 ส่วนหลัก คือ

1. วิว (Views) เป็นส่วนแสดงมุมมองต่าง ๆ ของระบบที่ถูกออกแบบขึ้นมา โดยจะใช้แผนภาพต่าง ๆ ในการอธิบาย
2. แผนภาพ (Diagram) เป็นสัญลักษณ์ต่าง ๆ ที่ถูกจัดเรียงขึ้น เพื่ออธิบายระบบในมุมมองต่าง ๆ
3. สัญลักษณ์ เพื่อใช้ในการแสดงหรือเป็นตัวแทนของสิ่งต่าง ๆ เช่น คลาส (Class) , ออบเจ็กต์ (Object) หรือ ความสัมพันธ์ (Relationship) เป็นต้น
4. ส่วนของคำอธิบายเพิ่มเติม หรือการแสดงข้อมูลอื่น ๆ ที่จำเป็น

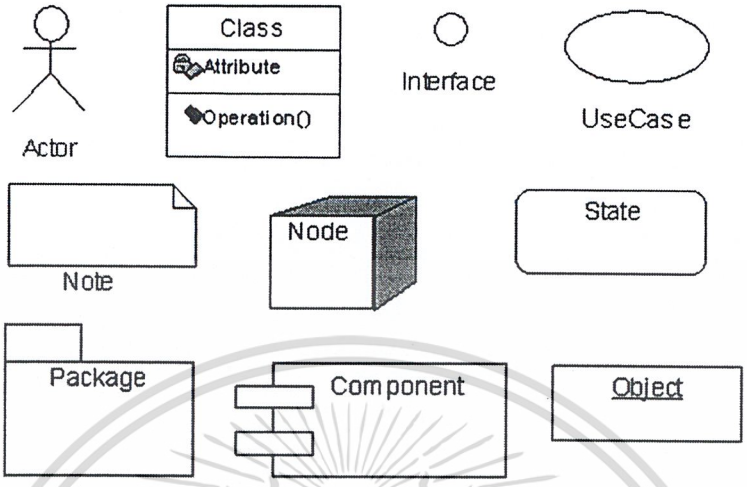
จุดประสงค์ของ UML ก็คือ ต้องการสร้างโมเดลในการพัฒนาที่เข้าใจและสร้างได้ง่าย แต่สามารถนำไปใช้ได้กับทุก ๆ ระบบ ซึ่งการสร้างโมเดลจำลองของระบบ มีข้อดี ดังนี้

- ได้ระบบที่ถูกต้องและตรงกับความต้องการจริงของผู้ใช้
- ทุก ๆ ส่วนของระบบมีความสอดคล้องกัน
- การสื่อสารระหว่างผู้พัฒนาและผู้ใช้ระบบมีความสะดวกและเป็นมาตรฐาน
- สามารถเปลี่ยนแปลงระบบที่ทำการออกแบบได้ง่าย
- สามารถทำความเข้าใจได้ง่ายทั้งผู้พัฒนาและผู้ใช้ระบบ

สัญลักษณ์ที่ใช้ในภาษา UML

ภาษา UML ประกอบไปด้วยสัญลักษณ์และเครื่องหมายจำนวนมาก ทำให้สามารถสร้างโมเดลและแผนภาพ (Diagram) ได้อย่างยืดหยุ่นและครอบคลุมโดเมนของปัญหาทั้งหมด นอกจากนี้ เครื่องหมายและสัญลักษณ์เหล่านี้ ยังแสดงหรือเป็นตัวแทนของสิ่งต่าง ๆ ที่อยู่ในโมเดลด้วย

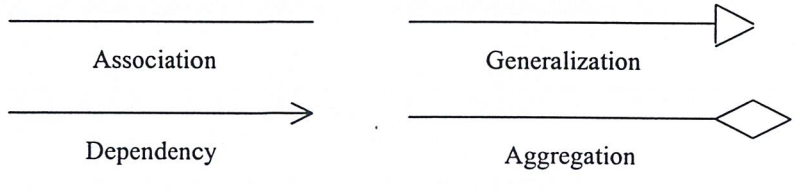
สัญลักษณ์ในภาษา UML ที่ใช้ในแต่ละแผนภาพมีเป็นจำนวนมาก ในที่นี้จะแสดงถึงสัญลักษณ์ที่พบเห็นบ่อยในโมเดลและแผนภาพต่าง ๆ ดังภาพ



รูปที่ ก-1 ตัวอย่างสัญลักษณ์ที่ใช้ในแผนภาพต่าง ๆ

นอกจากนี้ยังมีสัญลักษณ์ที่แสดงถึงความสัมพันธ์ต่าง ๆ ที่มีในระบบ ดังนี้

- **แอสโซซิเอชั่น (Association)** เป็นความสัมพันธ์ในลักษณะของการเชื่อมต่อกันระหว่างอุปกรณ์ที่มีความสัมพันธ์กัน ซึ่งเป็นความสัมพันธ์แบบ 2 ทิศทาง
- **เจเนรอลไลเซชั่น (Generalization)** หรือ **อินเฮอริแท้น (Inheritance)** เป็นความสัมพันธ์แบบอุปกรณ์หนึ่งสืบทอดคุณสมบัติมาจากอีกอุปกรณ์หนึ่ง โดยจะให้กำเนิดคลาสย่อย (Subclasss) จากคลาสแม่ (Superclass) โดยคลาสย่อยอาจมีการเพิ่มแอตทริบิวต์ (Attribute) และโอเปอเรชั่น (Operation) ในคลาสของมันเอง ความสัมพันธ์แบบนี้อาจถูกเรียกได้เป็น is-a หรือ kind-of
- **ดีเพนเด้นซี่ (Dependency)** เป็นความสัมพันธ์ในลักษณะของการพึ่งพากันระหว่างอุปกรณ์หนึ่ง ที่มีต่ออีกอุปกรณ์หนึ่ง
- **อากีเกชั่น (Aggregation)** เป็นรูปแบบการแสดงอุปกรณ์ที่ประกอบจากอุปกรณ์อื่นหลายอุปกรณ์ หรือเรียกว่า อุปกรณ์หนึ่งเป็นส่วนหนึ่งของอีกอุปกรณ์ ความสัมพันธ์แบบนี้ อาจเรียกได้เป็นความสัมพันธ์แบบเป็นส่วนประกอบ หรือที่เรียกว่า part-of

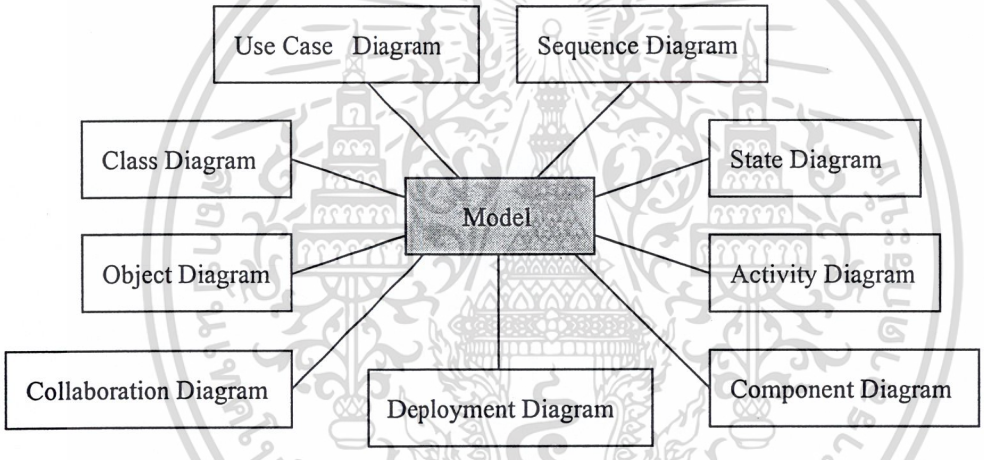


รูปที่ ก-2 สัญลักษณ์ของความสัมพันธ์แบบต่าง ๆ

ถึงแม้ว่าใน UML จะมีสัญลักษณ์และเครื่องหมายต่าง ๆ เป็นจำนวนมาก แต่ก็ยังไม่เพียงพอที่จะใช้แทนทุก ๆ สิ่งที่มีในโลกความเป็นจริง ดังนั้น จึงมีการกำหนด สเตอริโอไทป์ (Stereotype) ขึ้น เพื่อแทนความหมายของสิ่งเหล่านั้นตามต้องการได้ นอกจากนั้น สัญลักษณ์และเครื่องหมายของ UML ทั้งหมด ก็สามารถเขียนให้อยู่ในรูปแบบของสเตอริโอไทป์ได้เช่นเดียวกัน โดยสัญลักษณ์ของสเตอริโอไทป์นั้น จะเขียนชื่อของสัญลักษณ์ภายในวงเล็บ << >> เช่น <<Interface>> , <<Control>> หรือ <<Actor>> เป็นต้น

แผนภาพของ UML

UML มีโมเดลที่สื่อสารด้วยภาพได้สำหรับระบบหลายโมเดล โดยแต่ละโมเดลจะแสดงมุมมองต่อระบบที่ไม่เหมือนกัน และจะใช้แผนภาพ (Diagram) เพื่ออธิบายส่วนต่าง ๆ ของโมเดลที่สร้างขึ้น



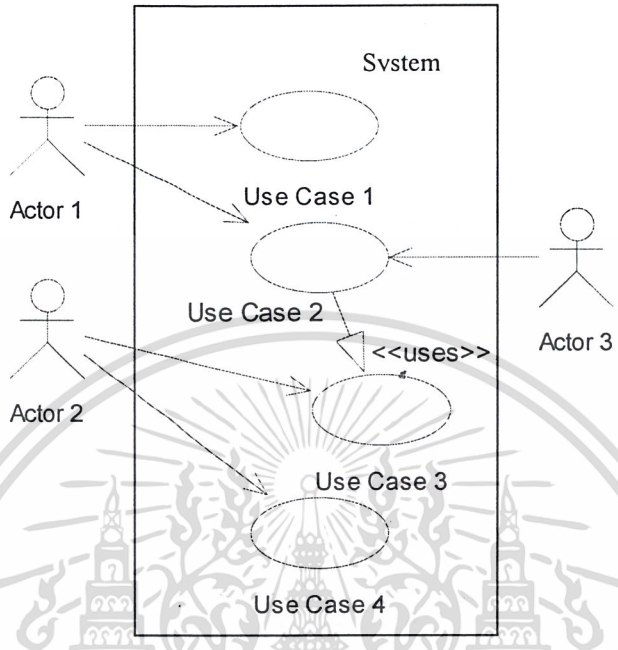
รูปที่ ก-3 แผนภาพต่าง ๆ ใน UML

Use Case Diagram

แผนภาพยูซเคส จะแสดงความสัมพันธ์ระหว่างผู้ใช้งาน (User) กับระบบ โดยใช้แอ็กเตอร์ (Actors) แทนผู้ใช้งาน แอ็กเตอร์จะไม่ใช่ส่วนประกอบของระบบ เป็นเพียงส่วนที่ใช้ติดต่อกับระบบ ซึ่งอาจเป็นเพียงการป้อนข้อมูลเข้าในระบบ หรือการรับข้อมูลจากระบบ หรืออาจเป็นทั้ง 2 อย่างก็ได้ ซึ่งการกำหนดคุณลักษณะของแอ็กเตอร์ (Identify Actor) ให้ได้โดยกำหนดว่า ใครเป็นผู้ใช้งานระบบ , ใครที่มีความสามารถเหมาะสมที่จะใช้งานระบบ , ใครที่มีส่วนสนับสนุนข้อมูลในระบบ , ใครที่สนับสนุนการบำรุงรักษาระบบ (Maintenance) และระบบมีการใช้งานกับภายนอกอย่างไร

แผนภาพยูซเคส เป็นมุมมองต่อระบบในระดับสูง และมีการใช้ภาษาอธิบายระบบทั่ว ๆ ไป ให้ผู้ใช้ทุก ๆ ระดับสามารถตรวจสอบแผนภาพยูซเคสได้ง่ายขึ้น และสามารถใช้แผนภาพยูซเคสในการสื่อสาร

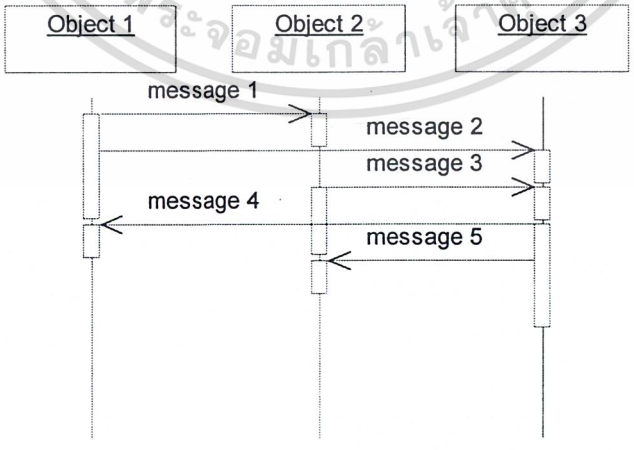
ระหว่างทีมผู้พัฒนาระบบเอง เนื่องจากสัญลักษณ์ต่าง ๆ ที่ใช้ในแผนภาพยูซเคส เป็นมาตรฐาน และมีรายละเอียดต่าง ๆ ครบถ้วน



รูปที่ ก-4 แผนภาพ ยูซเคส

Sequence Diagram

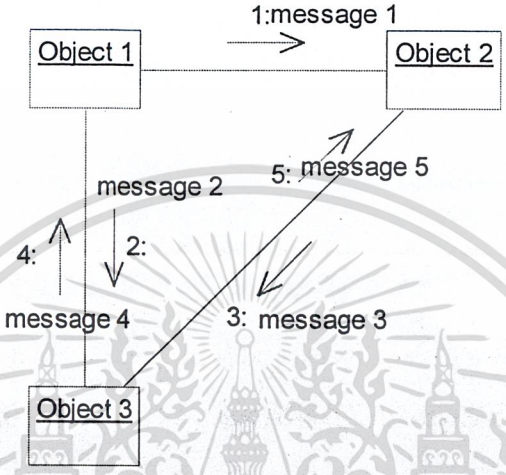
เป็นแผนภาพแสดงลำดับขั้นตอนในการทำงาน แสดงการติดต่อหรือตอบโต้ระหว่างออบเจ็กต์ โดยเน้นที่ลำดับก่อนหลัง และเวลาในการเปลี่ยนแปลง แต่ละออบเจ็กต์ มีการส่งแมสเซสติดต่อระหว่างกัน โดยมีเงื่อนไขคือคาบเวลามาเกี่ยวข้อง



รูปที่ ก-5 แผนภาพลำดับขั้นตอนการทำงาน

Collaboration Diagram

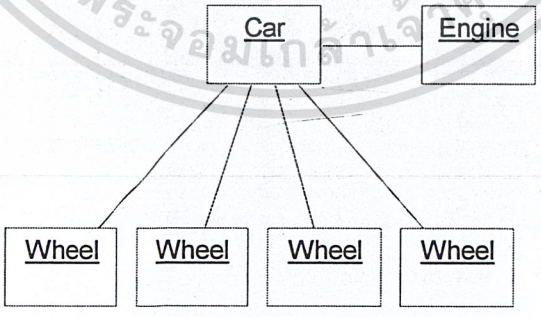
เป็นแผนภาพแสดงการทำงานร่วมกันของวัตถุ ซึ่งแสดงข้อมูลพื้นฐานเช่นเดียวกับแผนภาพแสดงลำดับขั้นตอนการทำงาน แต่มีความแตกต่าง คือ แผนภาพแสดงลำดับขั้นตอนการทำงาน จะแสดงลำดับของแมสเซสเป็นหลัก ส่วนแผนภาพแสดงการทำงานร่วมกันของวัตถุ จะแสดงโครงสร้างของวัตถุที่ทำงานร่วมกันเป็นหลัก โดยใช้สัญลักษณ์ลูกศรเป็นตัวกำหนดทิศทางการเคลื่อนที่ของแมสเซส และมีเลขลำดับการเคลื่อนที่ของแมสเซสเป็นตัวกำหนดลำดับขั้นตอนการทำงาน



รูปที่ ก-6 แผนภาพการทำงานร่วมกัน

Object Diagram

เป็นการแสดงความสัมพันธ์ระหว่างอุปกรณ์หรือส่วนต่าง ๆ ของวัตถุที่เกี่ยวข้องกัน เพื่อแทนออบเจ็กต์ที่มีอยู่จริง และแสดงความสัมพันธ์ต่าง ๆ ให้เห็นจริง



รูปที่ ก-7 แผนภาพวัตถุ

จากภาพเป็นตัวอย่างแผนภาพวัตถุของรถยนต์ ซึ่งมีล้อ 4 ล้อ และเครื่องยนต์เป็นส่วนประกอบและมีความเกี่ยวข้องกัน

Class Diagram

คลาส (Class) คือกลุ่มของออบเจ็กต์ที่มีคุณสมบัติ(Attributes) และพฤติกรรม (Behavior) ร่วมกัน โดยแผนภาพของคลาส เป็นแผนภาพที่มีความสำคัญมากที่สุดในการวิเคราะห์และออกแบบด้วยวิธีเชิงวัตถุ เนื่องจากแผนภาพของคลาสจะแสดงโครงสร้างของวัตถุและคลาสที่มีในระบบรวมทั้งแสดงความสัมพันธ์ด้วย

ในคลาสหนึ่ง ๆ จะประกอบด้วย

- ชื่อคลาส
- คุณสมบัติ (Attributes)
- การทำงาน (Operation)

โดยแต่ละคลาสจะมีความสัมพันธ์ในรูปแบบต่าง ๆ ได้ ดังนี้

1 หมายถึง จะมีออบเจ็กต์ในคลาสได้หนึ่งออบเจ็กต์เท่านั้น

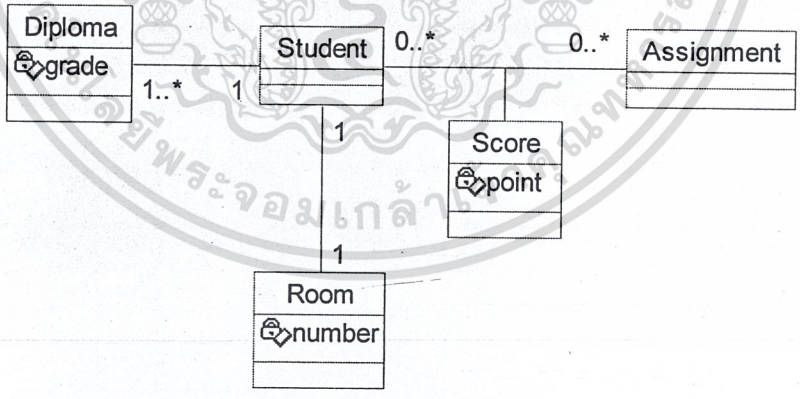
0..1 หมายถึง จะมีออบเจ็กต์ในคลาสไดอะแกรมได้เพียงหนึ่งหรืออาจไม่มีก็ได้

M..N หมายถึง จะมีออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ M ถึง N (เมื่อ M และ N เป็นจำนวนเต็มบอก)

* หมายถึง จะมีออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ 0 ขึ้นไป

0..* หมายถึง จะมีออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ 0 ขึ้นไป

1..* หมายถึง จะมีออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ 1 ขึ้นไป

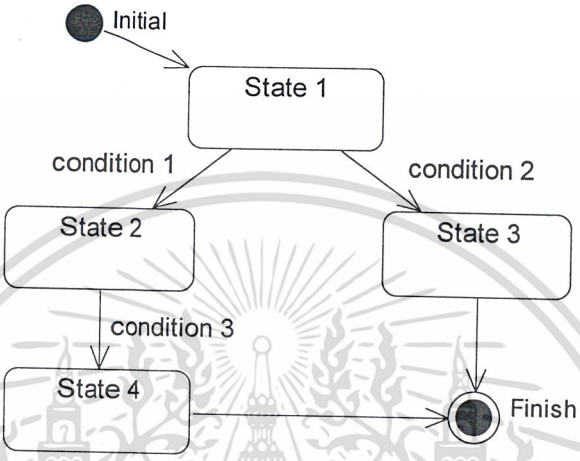


รูปที่ ก-8 แผนภาพคลาส

จากภาพตัวอย่างแผนภาพคลาส แสดงให้เห็นว่านักเรียน (Student) แต่ละคน จะต้องมีห้องเรียน (Room) คนละห้องเท่านั้น และนักเรียนอาจจะยังไม่มีใบเกรด (Diploma) หรือมีแล้วก็ได้

State Diagram

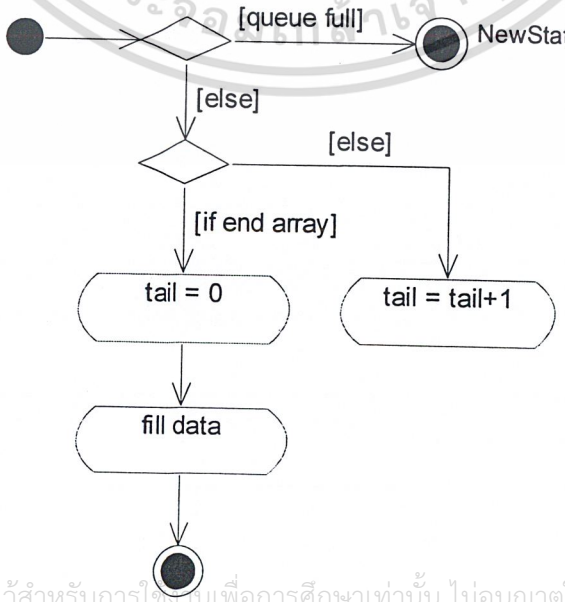
แผนภาพสถานะ ใช้อธิบายสถานะของคลาสต่าง ๆ ในระบบ โดยแสดงทุก ๆ สถานะที่เป็นไปได้ และเป็นเหตุการณ์ที่ทำให้เหตุการณ์ที่ทำให้ออบเจ็กต์เหล่านั้นเกิดการเปลี่ยนแปลง เรียกการเปลี่ยนแปลงนี้กว่า ทรานซิชั่น (Transitions)



รูปที่ ก-9 แผนภาพสถานะ

Activity Diagram

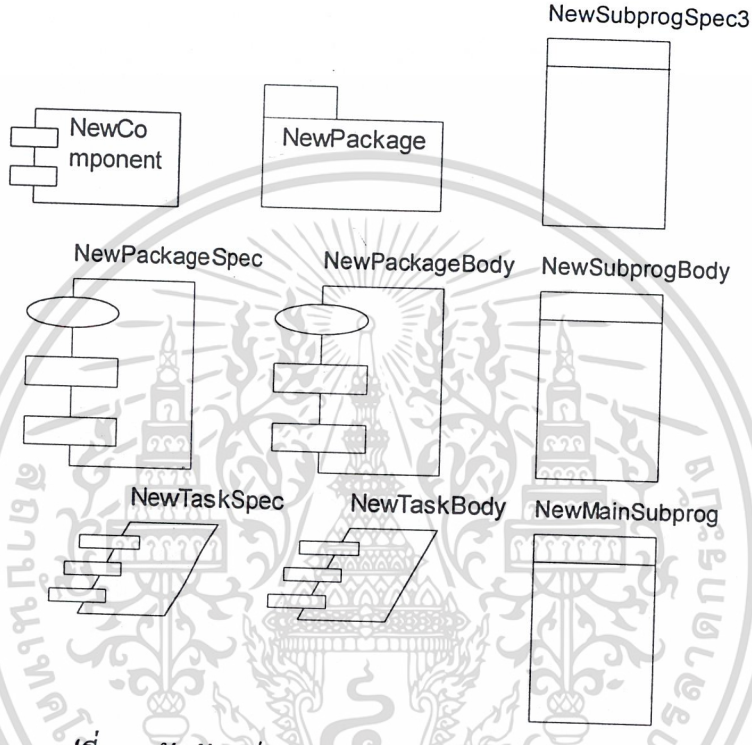
เป็นแผนภาพที่ช่วยในส่วนของการออกแบบอัลกอริทึ่ม โดยแบ่งย่อยระบบงานออกเป็นย่อย ๆ และมีลักษณะคล้ายกับโฟลล์ชาร์ต (Flow Chart) ซึ่งแต่ละสถานะจะถูกเปลี่ยนแปลงไปเมื่อเกิดการกระทำอย่างใดอย่างหนึ่งตามเงื่อนไขหรือการตัดสินใจที่กำหนด



รูปที่ ก-10 แผนภาพกิจกรรม สำหรับการเพิ่มข้อมูลลงในคิว

Component Diagram

แสดงโครงสร้างทางกายภาพของโค้ด (Code) ในรูปคอมโพเน้นท์ (Component) ของโค้ด โดยมีการแสดงความสัมพันธ์ของแต่ละคอมโพเน้นท์ เพื่อช่วยในการวิเคราะห์การเปลี่ยนแปลงของคอมโพเน้นท์เมื่อคอมโพเน้นท์ที่เกี่ยวข้องเกิดการเปลี่ยนแปลง โดยมีตัวอย่างของสัญลักษณ์ของแผนภาพคอมโพเน้นท์ ดังภาพ



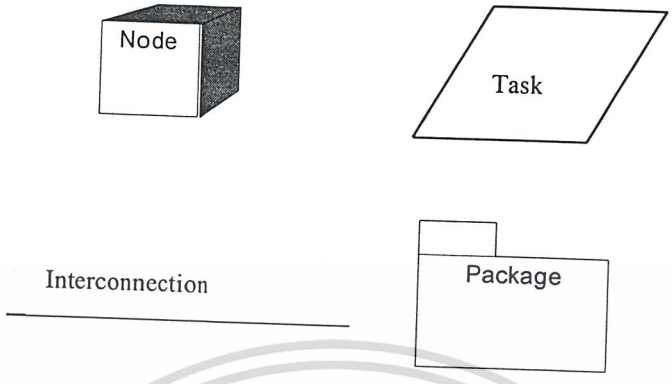
รูปที่ ก-11สัญลักษณ์ของแผนภาพคอมโพเน้นท์

Deployment Diagram

เป็นแผนภาพที่ใช้แทนระบบในระดับสถาปัตยกรรม โดยมีส่วนประกอบ ดังนี้

- โหนด (Node) ใช้แทนโพรเซสเซอร์ (Processor) , เซนเซอร์ (Sensor) , เราท์เตอร์ (Router) , จอแสดงผล (Monitor) หรืออุปกรณ์ฮาร์ดแวร์ (hardware) อื่น ๆ
- ระบบย่อยหรือแพ็กเก็ต (Package)
- ทากส์ (Task) หรืองานซึ่งเป็นวัตถุที่สามารถทำงานได้ด้วยตัวเอง (Active object)
- เส้นการเชื่อมต่อระหว่างโหนด (Interconnection)

โดยทั่วไป ผู้ออกแบบจะใช้สเตอริโอไทป์ (Stereotype) เพื่อเป็นเครื่องบ่งบอกชนิดของโหนดว่าเป็นโหนดประเภทใด เช่น เครื่องอ่านบัตร <<sensor>> , จอภาพ <<display>> , เครื่องนับเงิน <<actuator>> หรือ โมเด็ม <<actuator>> เป็นต้น



รูปที่ ก-12สัญลักษณ์ของแผนภาพดีพลอยเม้นท์

ภาคผนวก ข

การติดตั้ง TOMCAT

การติดตั้ง TOMCAT

Tomcat นี้เกิดมาจากโปรเจ็คของอาปาเช่ (Apache) ซึ่งต้องการนำเทคโนโลยีจาวามาใช้อย่างเต็มรูปแบบ ทั้งตัวเซิร์ฟเล็ต และเจเอสพี (JSP) ของจาวา และทางอาปาเช่ก็ได้ตั้งชื่อโปรเจ็คนี้ว่าจากาตาร์โปรเจ็ค (Jakarta Project) ผู้เข้าร่วมใน Project นี้อย่างเช่น ไอบีเอ็ม ซัน เป็นต้น (ร่วมทั้งสมาชิกจาก Apache Jserv Project) โดยในจากาตาร์โปรเจ็คนี้จะมีโครงงานย่อยๆอยู่มากมาย อย่างเช่น Ant, ORO ,Regexp ,Slide ,Struts ,Taglibs ,Tomcat ,Velocity ,Watchdog ซึ่งในที่นี้เราจะกล่าวถึงแต่ตัว Tomcat เพียงอย่างเดียว

ในขั้นตอนแรกต้องนำโปรแกรม Tomcat มาก่อน ซึ่งปัจจุบันก็มีถึงเวอร์ชัน 3.1 (เราสามารถเช็คดูได้ที่ http://jakarta.apache.org/) ใน Tomcat 3.1 นี้จะประกอบไปด้วย Servlet v.2.2 และ JSP v.1.1 ส่วนของตัวโปรแกรม สามารถดาวน์โหลดได้ที่

- http://aajc.au.ac.th/program/
- http://jakarta.apache.org/ (Official Wesite ของ Tomcat)

ก่อนที่จะเข้าไปดาวน์โหลดก็ต้องดูก่อนว่าเราใช้ระบบปฏิบัติการใดอยู่ จะได้นำโปรแกรมมาใช้ได้ถูกต้องและตรงกับระบบปฏิบัติการ

เมื่อได้ตัว Tomcat มา ถ้าเป็นพวกซิปของวินโดว์ก็อันซิปออก (มันจะสร้างไดเรกทอรี ชื่อ jakarta-tomcat ให้เรา) โดยไม่ต้องอินสตอลอะไรทั้งสิ้นแค่ทำการอันซิปเท่านั้นพอ หรือถ้าเป็นยูนิกส์ก็จัดการ Gunzip แล้วก็ tar xvf ออกมาเราก็จะได้ไดเรกทอรีของ tomcat ซึ่งจะประกอบด้วยสับไดเรกทอรีที่สำคัญดังนี้

Bin	เอาไว้เก็บ File ต่างที่เอาไว้สำหรับ Execute Tomcat
Conf	เอาไว้เก็บ File ต่างที่เอาไว้สำหรับ Config ตัว Tomcat, Apache JServ, JNI สำหรับ Tomcat
Lib	จัดเก็บ API ทั้งหมดที่เกี่ยวกับ Tomcat
Logs	Log File เกี่ยวกับ Tomcat (จะถูกสร้างขึ้นเมื่อมีการ startup Tomcat แล้วหนึ่งครั้ง)
Webapps	Directory ที่จัดเก็บ file ที่เป็น application ของเราอย่างพวก Servlet,JSP รวมทั้ง Web Page ด้วย (ซึ่งเป็น default ของ Tomcat)

ตาราง ข-1 แสดงรายละเอียดสับไดเร็กทอรี่

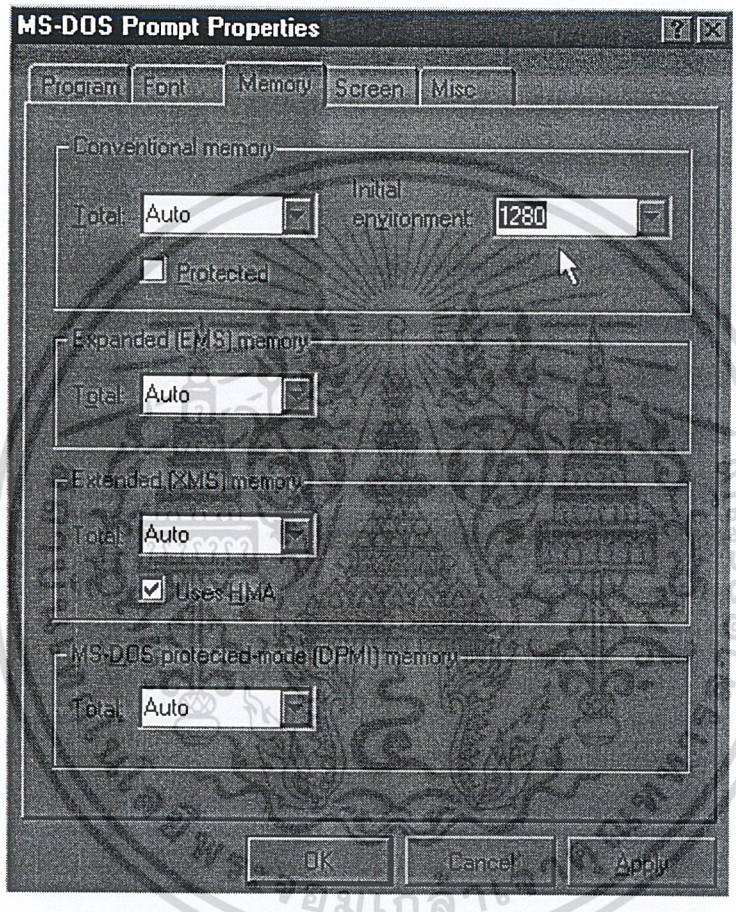
<u>วิธีการเรียก TOMCAT SERVER</u>

ไฟล์ที่เราจะใช้เรียก Tomcat ในวินโดว์ ได้แก่ ไฟล์ :

- startup.bat (จะมีหน้าที่เปิด Tomcat server)
- shutdown.bat (จะทำหน้าที่ปิด Tomcat server)

โดยจะมีการเรียกผ่านดอสพรอมพ์ ส่วนไฟล์สำหรับเรียก Tomcat ในระบบยูนิกส์ก็จะมีชื่อเหมือนกันเพียงจะต่างตรงนามสกุลที่ใช้เท่านั้น อย่างใน Solaris ก็จะเป็นพวก .sh เป็นต้น Ex. startup.sh

บางทีอาจจะเกิดปัญหาเมมโมรี ใน environment ของดอสไม่พอ เราจะต้องแก้ปัญหาโดยการตั้งค่าเมมโมรีใหม่ใน Dos Properties โดยตั้งค่าเมมโมรี ของ environment ให้เป็น 1280 ขึ้นไป ดังรูป



รูปภาพ ข-1 แสดงการตั้งค่า Initial environment

เมื่อเราทำการ Startup Tomcat เมื่อไหร่ ตัว Tomcat ก็จะทำการตั้งตัวเองเป็นเว็บเซิร์ฟเวอร์เพื่อรอการร้องขอข้อมูล และทำหน้าที่ตอบกลับไปโดยเราสามารถกดเข้าไปดูว่าเว็บเซิร์ฟเวอร์ ขึ้นหรือยังโดยการเรียกผ่านบราวเซอร์ อย่าง IE หรือ Netscape ก็ได้โดยพิมพ์ URL ดังนี้

URL: http://localhost:8080

(8080 คือ default port ของ Tomcat ซึ่งเราสามารถ set ให้เป็น port 80 ** ได้ใน file: server.xml ที่อยู่ใน directory "conf" ซึ่ง Port 80 : เป็น port มาตรฐานสำหรับส่งข้อมูลแบบ protocol HTTP)

วิธีการ Config Tomcat server (tomcat.properites)

การ Config Tomcat เราจะทำที่ jakarta-tomcat/conf ซึ่งจะประกอบด้วบไฟล์ config จำนวนมากมาย แต่ ไฟล์ที่เราจะใช้ครานี้จะมีอยู่ 2 ไฟล์ เท่านั้นนั่นคือ

- tomcat.properties (ทำหน้าที่ config environment ของ tomcat รวมทั้ง path ต่างๆใน Java environment)
- server.xml (ทำหน้าที่ set context ต่างๆ ใน tomcat)

configuration ใน tomcat.properties โดยปกติ เราจะใช้ Tomcat ได้เลยในระบบของวินโดว์แต่สำหรับในระบบของ Unix อาจจะต้องมีการตั้งค่าพาร์ธ ของ JVM ให้กับ Tomcat ด้วย โดยจะตั้งค่าตามนี้

(ของเก่า)

The Java Virtual Machine interpreter.

Syntax: wrapper.bin=[filename] (String)

Note: specify a full path if the interpreter is not visible in your path.

wrapper.bin=@JAVA@

(ของใหม่)

The Java Virtual Machine interpreter.

Syntax: wrapper.bin=[filename] (String)

Note: specify a full path if the interpreter is not visible in your path.

wrapper.bin=@c:\jdk1.2.2\bin@

ส่วน classpath จะตั้งค่าใน tomcat.properties เช่นกันแต่อยู่ในบรรทัดนี้

wrapper.classpath=@JSDK_CLASSES@ แก้เป็น wrapper.classpath=@c:\jdk1.2.2\bin\servlet.jar@

วิธีการ Config Tomcat server (server.xml)

server.xml fileตัวนี้ค่อนข้างมีความสำคัญอย่างมากเพราะจะเป็นตัวตั้งค่าพาร์ธของแอดเดรสของ Servlet Application ที่จะใช้ในเว็บไซต์และเป็นตัวตั้งค่าพอร์ทให้กับ Tomcat ด้วย

วิธีการตั้งค่าพอร์ทใหม่ให้กับ Tomcat เราจะทำดังนี้

(ของเก่า)

<Connector className="org.apache.tomcat.service.SimpleTcpConnector" >

<Parameter name="handler" value="org.apache.tomcat.service.http.HttpConnectionHandler"/ >

<Parameter name="port" value="8080" / >

</Connector>

(ของใหม่)

<Connector className="org.apache.tomcat.service.SimpleTcpConnector" >

<Parameter name="handler" value="org.apache.tomcat.service.http.HttpConnectionHandler"/ >

<Parameter name="port" value="80" / >

</Connector>

วิธีการเซ็ตแอดเดรสใหม่ให้กับ Tomcat เราจะแบ่งออกเป็นสองแบบ คือ

1. การเซ็ตโฮมเพจหน้าแรก ซึ่งเป็นการเซ็ตข้อมูลของโฮมเพจว่าอยู่ที่ไหน ซึ่งค่ามาตรฐาน (default) ของ Tomcat จะเป็นดังนี้

(ของเก่า)

<Context path="" docBase="webapps/ROOT" debug="0" reloadable="true" >

</Context >

(ของใหม่ ในแบบยูนิกส์)

<Context path="" docBase="home/aajc" debug="0" reloadable="true" >

</Context >

(ของใหม่ ในแบบวินโดว์)

<Context path="" docBase="c:\home\aajc" debug="0" reloadable="true" >

</Context >

พาร์ธของโฮมเพจที่เราสมมุติขึ้นอยู่ที่ /home/aajc/index.html แอตทริบิวต์ของข้อความมีดังนี้

path	เป็นการบอก Url ต่อจาก Url หลักที่เราได้ regist ไว้ใน DNS เช่น http://aajc.au.ac.th/ ถ้าใส่เป็น path= "/jboy" ก็จะเป็น http://aajc.au.ac.th/jboy และถ้าใส่เป็น "" ก็จะกลายเป็น homepage หน้าแรก
docBase	เป็นการ set directory ที่เราเก็บ document file ไว้ถ้าเป็น file แบบ html เราไม่จำเป็นต้องสร้าง context เลยเพราะตัว Tomcat จะจัดการเองให้หมด แต่ถ้าเป็นแบบ Servlet เราต้อง set docBase ให้ [จะแสดงให้ดูในภายหลัง]
debug	เป็นการ set piority ของ debug
reloadable	เป็นการ set ให้ Servlet สามารถ reload ได้ถ้าเรามีการแก้ไข Servlet แล้ว

ตาราง ข-2 แสดงพาธต่างๆ

2. การเซ็ต Servlet Application จะมีวิธีเซ็ตเหมือนกับการเซ็ตโฮมเพจโดยใช้ข้อความ เช่น HelloWorld.java ซึ่งเป็น Servlet Application แล้วนำไปเก็บไว้ในไดเรกทอรี /jakarta-tomcat/webapps/jservlet/WEB-INF/classes/HelloWorld.java (ในที่นี้สมมุติตั้งชื่อพาร์ธว่า jservlet แล้วต้องสร้างไดเรกทอรี "jservlet" ขึ้นมาด้วย) จากนั้นคอมไพล์ HelloWorld.java ให้เป็น .class Servlet Application ทุกๆตัวจะถูกเก็บอยู่ในสับไดเรกทอรี " /WEB-INF/classes" ซึ่งเราจะตั้งกี่ไดเรกทอรีก็ได้แต่จะต้องมีสับไดเรกทอรีนี้อยู่

(ของเก่า)

<Context path="/examples" docBase="webapps/examples" debug="0" reloadable="true">

< /Context>

(ของใหม่ ในแบบ Unix)

<Context path="/jservlet" docBase="webapps/jservlet" debug="0" reloadable="true" >

</Context >

ในตัวอย่างนี้ได้ set path="/jservlet" หมายความว่าเราจะได้ URL สำหรับที่เก็บ Servlet เป็น http://aajc.au.ac.th/jservlet/servlet (ใน Tomcat จะต่อ path "/servlet" ให้ด้วย) ส่วน docBase="webapps/jservlet" เราจะบอกว่า Servlet Application เราเก็บไว้อยู่ที่

" /jakarta-tomcat/webapps/jservlet/WEB-INF/classes "

ภาคผนวก ค.

Oracle Server

Oracle Server

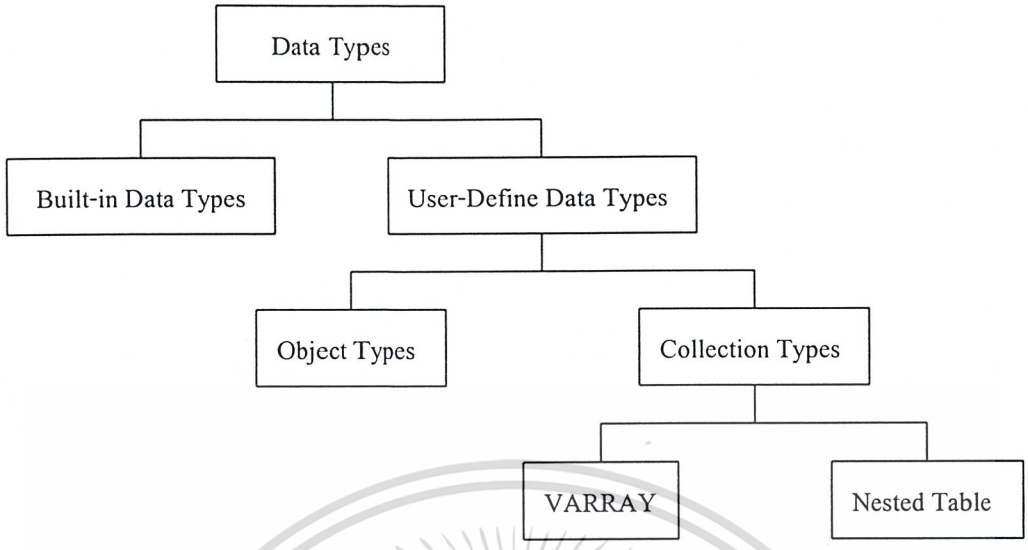
ออราเคิลเซิร์ฟเวอร์ (Oracle Server) คือ ฐานข้อมูลเชิงสัมพันธ์ (RDBMS : Rational Database Management System) ตัวหนึ่งจากบริษัทออราเคิล และเป็นเชิงฐานข้อมูลเชิงสัมพันธ์พาณิชย์ตัวแรกของโลกด้วย ออราเคิลเซิร์ฟเวอร์มีความสามารถโดดเด่น จุดเด่นคือความเชื่อถือได้ (Reliable) สูง อีกทั้งยังมีให้เลือกใช้ในเกือบทุกแพลตฟอร์ม (Platform) ตั้งแต่บนเมนเฟรม (Mainframe) , มินิคอมพิวเตอร์ (Minicomputer) , พีซี (Personal Computer) บนระบบปฏิบัติการเรื่อยไปตั้งแต่วินโดว์ส 9x (Windows9X) , วินโดว์สเอ็นที (WindowsNT) , วินโดว์สซีอี (WindowsCE) , ยูนิกซ์ (UNIX) , โซลาริส (Solaris), ลินุกซ์ (Linux) โดยที่ในทุกพอร์ตมีโครงสร้างการเหมือนกัน ๆ หมด คำสั่งที่ใช้ก็เป็นแบบเดียวกัน ทำให้ทำงานร่วมกันได้ สามารถนำข้อมูลจากพอร์ตหนึ่งไปพอร์ตอื่นได้อย่างไม่มีปัญหา เหมาะแก่การทำระบบต้นแบบ (Prototype) เช่น นักพัฒนาสามารถเขียน , ทดสอบ , พัฒนา ระบบบนเครื่องเดส์กทอป (Desktop) ได้ โดยไม่ต้องสนใจว่าสุดท้ายจะใช้นำไปใช้ที่แพลตฟอร์มใด (สำหรับเรื่องทำงานได้บนหลายแพลตฟอร์มเป็นหัวใจสำคัญของซอฟต์แวร์ (Software) จากออราเคิลก็ว่าได้)

Oracle 8I เป็นระบบฐานข้อมูลที่พัฒนาขึ้นมาจากระบบฐานข้อมูลแบบเดิม ซึ่งมีความสามารถในการจัดเก็บข้อมูลที่มีความซ้ำซ้อนมาก ๆ เช่น ข้อมูลมัลติมีเดีย , รูปภาพ หรือเสียง ได้อย่างมีประสิทธิภาพ โดยมีการเพิ่มชนิดข้อมูล (Data Types) ใหม่ขึ้นมา เพื่อรองรับการจัดเก็บข้อมูล และมีชนิดของข้อมูลที่ผู้ใช้สามารถกำหนดขึ้นมาเอง (User-Defined Data types) ข้อมูลชนิดนี้จะมีคุณสมบัติของออบเจ็กต์โอเรียนเท็ดเหมือนกันออบเจ็กต์ ที่เขียนขึ้นจากภาษาที่เป็นโปรแกรมเชิงวัตถุแบบต่าง ๆ

ชนิดของข้อมูล (Data Types) ใน Oracle8i

ข้อมูลที่ถูกจัดเก็บในฐานข้อมูล จะมีหลายชนิด แต่ละชนิดจะมีคุณสมบัติต่าง ๆ กันออกไป เช่น NUMBER ใช้เก็บตัวเลขที่นำไปคำนวณ , VARCHAR2 ใช้เก็บสตริง (String) หรือ DATE ใช้เก็บค่าของวันที่ เป็นต้น โดยชนิดของข้อมูลเหล่านี้ จะมีรายละเอียดและการกำหนดเนื้อที่ในการเก็บต่างกัน ผู้ใช้จะต้องเลือกใช้ให้เหมาะสมกับข้อมูลที่นำมาจัดเก็บ เพราะการกำหนดชนิดข้อมูลบางประเภทที่ไม่เหมาะสมใช้เนื้อที่มากเกินความจำเป็น อาจทำให้สิ้นเปลืองเนื้อที่ในการจัดเก็บได้ หรือถ้ากำหนดชนิดข้อมูลและกำหนดขนาดในการจัดเก็บไม่เพียงพอ อาจเกิดปัญหากับการจัดเก็บข้อมูลภายหลังได้

ชนิดของข้อมูลในออราเคิล 8i แบ่งเป็น 2 ประเภทใหญ่ ๆ คือ ชนิดข้อมูลมาตรฐานที่มีอยู่ในตัวของออราเคิล (Built-in data Types) และชนิดของข้อมูลที่ผู้ใช้กำหนดขึ้นเอง (User-Define Data Types)



Built-in Data Types

เป็นชนิดของข้อมูลมาตรฐานที่มีอยู่ในออราเคิลแล้ว ประกอบด้วย

ชนิดของข้อมูล (Data Types)	***คุณสมบัติ***
CHAR	ข้อมูลตัวอักษรที่มีขนาดความยาวคงที่ตามที่ได้กำหนดไว้ โดยสามารถกำหนดความยาวได้ตั้งแต่ 1 ถึง 2,000 ไบต์ ซึ่งถ้าหากมีการเปลี่ยนแปลงข้อมูลที่มีขนาดน้อยกว่ากำหนด ข้อมูลจะถูกเติมช่องว่างตามขนาดที่ได้กำหนดไว้ ถ้าข้อมูลมีขนาดมากกว่าที่กำหนดก็จะตัดออกให้ได้ขนาดตามที่กำหนดไว้ หากข้อมูลยาวมาก ๆ จะเกิดเป็นข้อผิดพลาด (Error) ขึ้น
VARCHAR2	ข้อมูลตัวอักษรที่มีความยาวเปลี่ยนแปลงได้ มีการกำหนดความยาวได้ตั้งแต่ 1 ถึง 4,000 ไบต์ โดยต้องมีการกำหนดความยาวเริ่มต้นไว้ ถ้าข้อมูลที่เปลี่ยนแปลงมีขนาดน้อยกว่าที่กำหนดไว้ จะเก็บเท่ากับจำนวนของข้อมูลจริง
VARCHAR	มีลักษณะเหมือนกัน VARCHAR2 แต่สามารถเก็บขนาดของข้อมูลที่เปลี่ยนแปลงได้ และเปรียบเทียบความแตกต่างของสตริงได้ด้วย
NCHAR **NVARCHAR2**	ชนิดของข้อมูลแบบ NSL (National Language Support) ซึ่งจะเป็นรูปแบบที่ตัวอักษรที่ 1 ไบต์ เก็บมากกว่า 1 ตัวอักษร เช่น ตัวอักษรภาษาไทย ซึ่ง NCHAR จะเก็บข้อมูลที่มีความยาวคงที่หรือเปลี่ยนแปลงได้ โดยมีขนาดสูงสุดได้ 2,000 ไบต์ และ NVARCHAR2 จะเก็บข้อมูลที่มีความยาวเปลี่ยนแปลงได้ โดยมีขนาดสูงสุด 4,000 ไบต์
LONG	ชนิดของข้อมูลที่สามารถเก็บได้สูงสุด 2 กิกะไบต์ เหมาะสำหรับการเก็บเท็กซ์ไฟล์ (Text File)

ชนิดของข้อมูล (Data Types)	*คุณสมบัติ*
NUMBER	เก็บข้อมูลตัวเลขที่เป็นจำนวนเต็ม (Fixed Numbers) และเลขทศนิยม (Floating-Point Numbers) ซึ่งสามารถเก็บได้สูงสุด 32 หลัก
DATE	เก็บข้อมูลที่เป็นวันและเวลา แต่ละฟิลด์จะกำหนดไว้ 7 ไบต์
LOB	ใช้เก็บข้อมูลที่มีขนาดใหญ่ เช่น เท็กซ์ไฟล์ , ภาพกราฟฟิก , เสียง โดยมีขนาดได้สูงสุด 4 กิกะไบต์ ประกอบด้วย - **BLOB** เก็บข้อมูลที่เป็นไบนารี่ (Binary) ที่ไม่มีโครงสร้างในฐานข้อมูล - **CLOB** เก็บตัวอักษร แบบ Single-byte - **NCLOB** เก็บตัวอักษร แบบ Fixed-length - **BFILE** เก็บข้อมูลประเภทไบนารี่ที่เป็นแฟ้มข้อมูลนอกฐานข้อมูล โดยจะเก็บค่าตัวชี้ตำแหน่ง (File Locator) ที่ชี้ตำแหน่งของแฟ้มข้อมูล ข้อมูลประเภทนี้เก็บไได้สูงสุด 4 กิกะไบต์ และเป็นประเภทอ่านอย่างเดียว ไม่สามารถแก้ไขได้
RAW **LONG RAW**	เป็นชนิดข้อมูลที่จะไม่มีการเปลี่ยนแปลงแม้ขณะที่มีการย้ายข้อมูลระหว่างระบบต่าง ๆ ซึ่งชนิดข้อมูลแบบนี้เหมาะกับข้อมูลที่เป็นแบบไบนารี่ และที่เป็นไบต์สตริง โดย RAW จะกำหนดให้มีขนาดสูงสุดได้ 2,000 ไบต์ และสามารถนำไปทำ Index ได้ ส่วน LONG RAW เก็บได้ถึง 2 กิกะไบต์ แต่ไม่สามารถนำไปทำ Index ได้
ROWID	ข้อมูลที่บอกถึงตำแหน่งของแถวทุก ๆ แถวในตาราง ซึ่งจะเก็บข้อมูลเป็นไบนารี่ โดยมีขนาด 10 ไบต์สำหรับ Extended ROWID และ ขนาด 6 ไบต์สำหรับ Restricted ROWID

User-Defined Data Types

เป็นคุณสมบัติที่เพิ่มขึ้นมาใน ORDBMS ให้ผู้ใช้สามารถกำหนดชนิดข้อมูลขึ้นมาเองได้ รวมถึงการกำหนดรายละเอียดของโครงสร้างข้อมูลและการทำงานของตัวมันเอง และสามารถใช้ชนิดของข้อมูลที่กำหนดขึ้นเหล่านี้ในเรชั่นแนล โมเดล (Rational Model) ได้ ซึ่งเพิ่มประสิทธิภาพในการพัฒนาโปรแกรมในรูปแบบของออบเจ็กต์ โอเรียนเท็ด โปรแกรมมิ่ง (Object Oriented Programming)

ชนิดของข้อมูลแบบ User-Defined Data Types ประกอบไปด้วย

Object Type

เป็นชนิดของข้อมูลที่มีความซับซ้อนและใกล้เคียงกับความเป็นจริงมากขึ้น ซึ่งจะมีการนำเอาข้อมูล (Data) และการกระทำ (Operation) มาไว้ด้วยกัน ผู้พัฒนาสามารถทำงานได้โดยไม่ต้องรู้รายละเอียดภายในของข้อมูลชนิดออบเจ็กต์นี้เลย และยังสามารถนำข้อมูลชนิดออบเจ็กต์นี้กลับมาใช้ใหม่ได้ (Reuse)

ตัวอย่างการสร้างออบเจ็กต์ ไทป์ ที่ใช้เก็บข้อมูลที่อยู่

```
CREATE TYPE address_type AS OBJECT
(       home_address    VARCHAR2(10),
        road            VARCHAR2(30),
        region          VARCHAR2(30),
        province        VARCHAR2(30),
        zipcode         NUMBER(5));
```

การนำออบเจ็กต์ ไทป์นี้มาใช้ ให้ใช้ชื่อชนิดของข้อมูลที่ตั้งไว้ข้างต้น มาใส่เป็นชนิดของข้อมูลในสดมภ์ (Column) ได้เลย เช่น

```
CREATE TABLE member
(       id#     NUMBER(8) PRIMARY KEY,
        name    VARCHAR2(70),
        address ADDRESS_TYPE);
```

เมื่อต้องการใส่ข้อมูลลงในตารางที่สร้างไว้ให้ใช้ ARRESS_TYPE(‘490‘ , ’Sukhumvit’ , ‘Wattana’ , ‘Bangkok’ , 10110) ในตำแหน่งของข้อมูลที่ต้องการ

Collection Type

มีลักษณะเป็นหน่วยของข้อมูล (Data Unit) ที่มีจำนวนสมาชิกไม่จำกัด และสมาชิกทั้งหมดมีชนิดข้อมูลเหมือนกัน ประกอบด้วย

- **Varrays** มีลักษณะเป็นเซ็ตของข้อมูลแบบมีลำดับ (Array) สมาชิกทุกตัวจะเป็นข้อมูลชนิดเดียวกัน โดยจะมีเครื่องชี้ (Index) ที่เป็นหมายเลขที่มีความสอดคล้องกับตำแหน่งของสมาชิกในอาร์เรย์ การกำหนดขนาดของอาร์เรย์ไม่มีจำกัดแน่นอน แต่สามารรระบุขนาดที่ใหญ่ที่สุดเมื่อเรามีการกำหนดชนิดของข้อมูลเป็นแบบนี้ และสามารถนำชนิดของข้อมูลแบบนี้ไปใช้เป็น ชนิดของข้อมูลของสดมภ์ (Column) ในตารางรีเรชั่นเนล , แอตทริบิวต์ ของออบเจ็กต์ไทป์ หรือชนิดของข้อมูลที่ส่งค่ากลับมาจากฟังก์ชั่น

- **Nested Tables** เป็นเซ็ตของข้อมูลแบบที่ไม่เรียงลำดับ ซึ่งสมาชิกของทุก ๆ ตัวจะเป็นข้อมูลชนิดเดียวกัน Nested Tables มีเพียงคอลัมน์เดียว และข้อมูลในคอลัมน์นั้น จะถูกมองเหมือนเป็นตารางที่ภายในแต่ละแถวจะมีแถวย่อย ๆ ลงไปอีก การกำหนดชนิดของข้อมูลแบบ Nested Table เป็นดังตัวอย่าง

```
CREATE TYPE test_score AS OBJECT
(       student_id      NUMBER(8),
        score           NUMBER);

CREATE TYPE test_score_table AS TABLE OF test_score;

CREATE TABLE test_results
(       instructor_id   VARCHAR2(8),
        class_id        VARCHAR2(6),
        name            VARCHAR2(30),
        scores          TEST_SCORE_TABLE);
    NESTED TABLE scores STORE AS test_scores;
```

ส่วนการใส่ข้อมูลเข้าไปในตารางให้ใส่ซ้อนกันลงไปตามตำแหน่งของข้อมูล เช่น

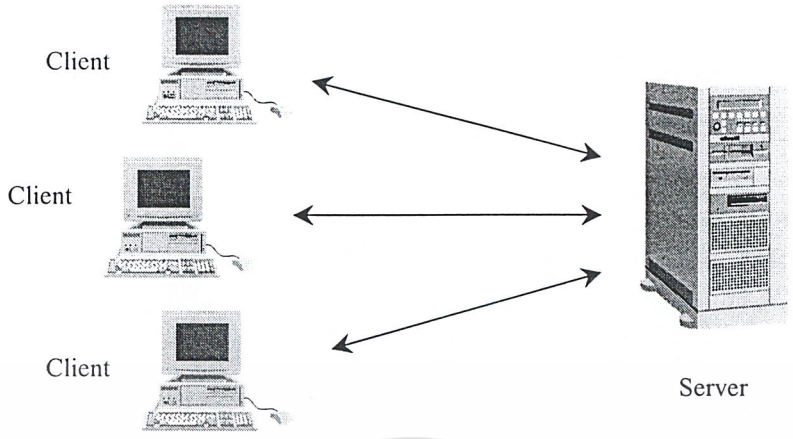
```
(TEST_SCORE_TABLE
  (TEST_SCORE ('40010249' , 90),
  TEST_SCORE ('40010331' , 95),
  TEST_SCORE ('40010554' , 90)));
```

โครงสร้างการเชื่อมต่อฐานข้อมูล

Oracle 8I เป็นฐานข้อมูลไคลเอ็นต์ / เซิร์ฟเวอร์ ดังนั้นดาต้าเบส เซิร์ฟเวอร์ (Database Server) จึงทำงานเป็นอิสระจากแอพพลิเคชั่น (Application) ที่ติดต่อกับฐานข้อมูล เซิร์ฟเวอร์จะรอ (Listen) , ตอบรับ (Accept) การร้องขอ (Request) จากไคลเอ็นต์ โดยช่วงแรกการติดต่อระหว่างออราเคิลกับไคลเอ็นต์จะเป็นโครงสร้างแบบทูเทียร์โมเดล (Two-tier Model) แต่ต่อมา เมื่อมีการพัฒนาทางด้านเว็บและการเชื่อมต่ออินเทอร์เน็ตมากขึ้น การเข้าถึงข้อมูลด้วยแอพพลิเคชั่นที่หลากหลาย ทำให้มีการใช้โครงสร้างแบบทรีเทียร์โมเดล (Three-tier Model) ขึ้น

โครงสร้างแบบทูเทียร์ (Two-tier Model)

โครงสร้างแบบทูเทียร์ มีเซิร์ฟเวอร์ 1 ตัว สำหรับรันฐานข้อมูล และไคลเอ็นต์ 1 ตัวหรือมากกว่า ซึ่งรันโปรแกรมที่ติดต่อกับฐานข้อมูล สำหรับออราเคิล ใช้ Net8 เป็นโปรโตคอลในการกำหนดการติดต่อสื่อสารระหว่างไคลเอ็นต์และเซิร์ฟเวอร์

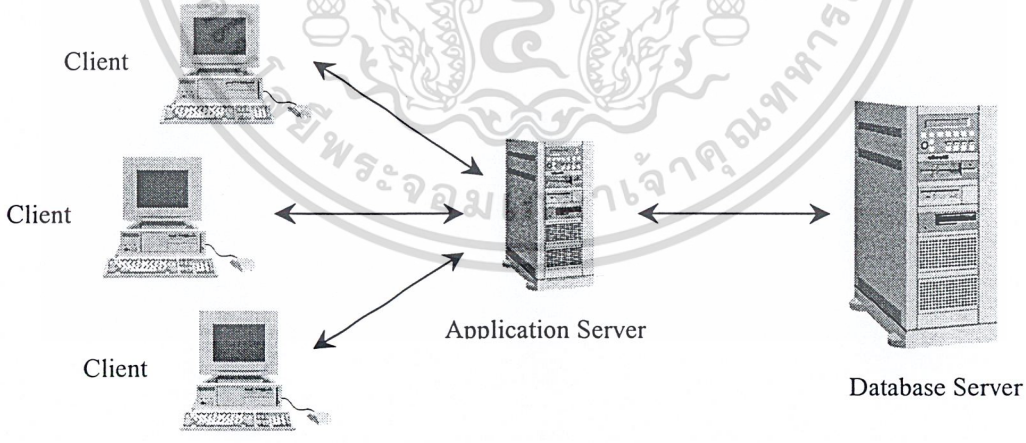


โครงสร้างแบบทูเทียร์

ในการติดต่อ ฝั่งไคลเอ็นต์ต้องติดตั้ง Net8 Client Software ซึ่งมีอินเทอร์เฟส (Interface) ที่ง่ายและสะดวกกับผู้ใช้ ส่วนฝั่งเซิร์ฟเวอร์ ต้องติดตั้ง Net8 Server Software และใช้ส่วนใหญ่จะใช้ TCP/IP โปรโตคอลในการในการสื่อสารผ่านเน็ตเวิร์ก (Network)

โครงสร้างแบบทรีเทียร์ (Three-tier Model)

โครงสร้างแบบทรีเทียร์เป็นที่นิยมใช้กันมาก นับตั้งแต่มีการติดต่อสื่อสารผ่านอินเทอร์เน็ต เพราะมีความสามารถในการโต้ตอบกับผู้ใช้ได้อย่างมีประสิทธิภาพ ผ่านทางเว็บไซต์



โครงสร้างแบบทรีเทียร์

จากภาพจะเห็นได้ว่า ทรีเทียร์โมเดลยังคงมีไคลเอ็นต์และออราเคิลเซิร์ฟเวอร์เหมือนเดิม และมีแอพพลิเคชั่นเซิร์ฟเวอร์ เพิ่มขึ้นมา ทำให้เมื่อมีเพิ่มไคลเอ็นต์มากขึ้น ก็ไม่จำเป็นต้องไปเปลี่ยนแปลงเซิร์ฟเวอร์ฐาน

ข้อมูล แต่ใช้วิธีการเพิ่มแอพพลิเคชั่นเซิร์ฟเวอร์แทน ทำให้มีความยืดหยุ่นมากขึ้น นอกจากนี้ ยังเป็นการลดโหลด (Load) บนดาต้าเบสเซิร์ฟเวอร์อีกด้วย

บรรณานุกรม

[1] William W. Cohen: *" Automatically Extracting Features for Concept Learning from the Web"*

[2] Dayne Freitag: *"Information Extraction from HTML:Application of a General Machine Learning Approach"*

[3] Mark Craven, Dan DiPasquo,Dayne Freitag, Andrew McCallum Tom Mitchell, Kamal Nigam Sean Slattery: *"Learning to Extract Symbolic Knowledge from the World Wide Web"*, 1998

[4] Jeffrey Dean and Monika R. Henzinger: *"Finding Related Pages in the World Wide Web"*

[5] Ion Muslea, Steven Minton, Craig A. Knoblock: *"Hierarchical Wrapper Induction for Semistructured Information Sources"*, September 1999

[6] Line Eikvil : *"Information Extraction from World Wide Web -A Survey"*, 1999

[7] Stephen Soderland: *"Learning Information Extraction Rules for Semi-structured and Free Text"*

[8] Ralf Kuhnel: *"Java and Agent Oriented Programming"*

[9] Robert B. Doorenbos,Oren Etzioni and Daniel S. Weld: *"A Scalable Comparison-Shopping Agent for the World Wide Web"*

[10] Xiaoying Gao and Leon Sterling: *"Semi-Structured Data Extraction from Heterogeneous Sources"*

[11] Peter Buneman: *"Semistructured Data"*

[12] Jaeyoung Yang, Eunseok Lee and Joongmin Choi: *"A Shopping Agent That Automatically Contructs Wrappers for Semi-Structured Online Vendors"*

[13] Theodore Hong: *"Visualizing Real Estate Property Information on the Web"*

[14] Naveen Ashish and Craig Knoblock: *"Wrapper Generation for Semi-structured Internet Sources"*