

ห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์
ELECTRONIC THESIS LIBRARY



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ร.ท.

ภาควิชาวิศวกรรมคอมพิวเตอร์

๑๖๒๓๗

คณะวิศวกรรมศาสตร์

๖๕๖๖

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เลขหมู่.....
เลขทะเบียน..... 46139
วัน, เดือน, ปี 20 ส.ค. 2546

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

๑๖๒๓๗๑๒

ห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์

ELECTRONIC THESIS LIBRARY



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์

ELECTRONIC THESIS LIBRARY

ผู้จัดทำ

1. นาย คณัย อมตาริยกุล

รหัสประจำตัว 42015303

2. นางสาว สุพัตรา คำจันทร์

รหัสประจำตัว 42015327

อาจารย์ที่ปรึกษา


(ดร. วิศิษฐ์ หิรัญกิตติ)



ห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์

นายคนัย อมตาริยกุล 42015303

นางสาวสุพัตรา คำจันทร์ 42015327

ดร.วิศิษฎ์ หิรัญกิตติ อาจารย์ที่ปรึกษา

ปีการศึกษา 2544

บทคัดย่อ

ปริญญานิพนธ์นี้เสนอวิธีการออกแบบ และพัฒนาเว็บไซต์ สำหรับเก็บรวบรวม และเป็นระบบสืบค้นบทคัดย่อวิทยานิพนธ์ในรูปแบบอิเล็กทรอนิกส์ ระบบนี้สามารถทำการค้นหาชื่อวิทยานิพนธ์ที่ต้องการโดยอาศัยคำสำคัญต่าง ๆ จากบทคัดย่อของวิทยานิพนธ์นั้น ๆ ทำให้ผู้ใช้สามารถใช้งานระบบผ่านทางเว็บเบราว์เซอร์ได้ สำหรับผู้ใช้ที่เป็นผู้ปฏิบัติการสามารถใช้ระบบนี้ทำการจัดเก็บบทคัดย่อวิทยานิพนธ์สำหรับผู้ใช้นุคคลทั่วไปสามารถใช้ระบบนี้สืบค้นบทคัดย่อวิทยานิพนธ์ได้

บทคัดย่อวิทยานิพนธ์ในรูปแบบอิเล็กทรอนิกส์ มีการจัดเก็บในรูปแบบเอกสาร XML ตามชื่อแท็กที่กำหนดโดยมาตรฐานฉบับลินเมต้าด้า เวอร์ชัน 1.1 ที่เป็นภาษาไทย ข้อมูลบทคัดย่อวิทยานิพนธ์ในรูปแบบ XML ได้จัดเก็บในฐานข้อมูลเชิงสัมพันธ์ เพื่อช่วยประหยัดเนื้อที่หน่วยความจำ และอำนวยความสะดวกในเรื่องการสืบค้น สำหรับโปรแกรมที่จัดการระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์พัฒนาโดยอาศัยจาวาเซิร์ฟเลต (Java Servlet)

ELECTRONIC THESIS LIBRARY

Danai Amatariyakul	42015303
Supattra Khamchan	42015327
Dr. Visit Hirankitti	Advisor

ABSTRACT

This Project proposed a design and development of a Web Service for a digital library of thesis abstracts. The library user can access to this service via a Web Browser. With this service, the normal users can search through thesis abstract using keywords database while the administrator user can use the library service to archive thesis abstracts.

The electronic version of a thesis abstract is represented in an XML document form using tags defined by the Dublin-metadata 1.1 Thai Edition. All XML documents of the thesis abstracts are stored in a relational database in order to save storage space and speed up the abstract search. Our digital library is implemented using Java Servlet.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากบุคคลหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วิศิษฐ์ หิรัญกิตติ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้คำแนะนำ เอาใจใส่ และคอยช่วยเหลือเสมอมา ซึ่งต้องขอขอบคุณเป็นอย่างมาก

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้สนับสนุนสถานที่และทรัพยากรต่างๆ ในการทำงานวิจัยนี้ทั้งคำปรึกษาและคำแนะนำที่ดีจากอาจารย์ทุกๆ ท่านด้วย ขอขอบคุณ Software (DB2) จาก IBM ขอขอบคุณเพื่อนๆ และน้องๆ ทุกคนที่คอยช่วยเหลือและคอยกวนใจในทุกๆ โอกาส ขอขอบคุณคณาจารย์ในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ประสิทธิ์ประสาทวิชาความรู้มาให้แก่เรา และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ทำให้เรามีวันนี้ได้

สุดท้ายนี้ขอขอบคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

คนนี้ อมตารียกุล
สุพัตรา คำจันทร์

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพ	VII
สารบัญตาราง	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	1
1.3 ขอบเขตของปริญญานิพนธ์	2
1.4 วิธีการดำเนินงาน	2
บทที่ 2 Extensible Markup Language (XML)	3
2.1 บทนำ	3
2.2 XML คืออะไร	3
2.3 การสร้าง Well-Formed ให้เอกสาร XML	4
2.3.1 Well-Formed ของเอกสาร XML	5
2.3.2 การมาร์กอัปและข้อมูลที่เป็นตัวอักษร (Markup and Character Data)	5
2.3.3 การประกาศโปรล็อก (The Prolog)	6
2.3.4 การประกาศ XML (The XML Declaration)	6
2.3.5 การเขียนคอมเมนต์ (Comments)	7
2.3.6 การใช้งาน Processing Instructions	7
2.3.7 แท็กและอิลิเมนต์ (Tags and Element)	7
2.3.7.1 ชื่อแท็ก (Tag Names)	8
2.3.7.2 อิลิเมนต์ว่าง (Empty Element)	8
2.3.7.3 อิลิเมนต์รูต (The Root Element)	8
2.3.7.4 แอตทริบิวต์ (Attributes)	9
2.3.8 การสร้างโครงสร้างเอกสารให้เป็น Well-Formed	11
2.3.9 CDATA Sections	12
2.3.10 XML Namespaces	12

สารบัญ (ต่อ)

	หน้า
บทที่ 3 Document Type Definition (DTD)	14
3.1 การตรวจสอบความถูกต้องของเอกสาร XML	14
3.2 องค์ประกอบของ DTD	14
3.3 Document Type Declarations	15
3.4 Elements Declarations	15
3.5 DTD Comments	17
3.6 การกำหนด DTD ใวนอกเอกสาร XML	17
3.7 DTD Entities	18
3.7.1 General Entities	18
3.7.2 Parameter Entities	20
3.8 Attributes	20
บทที่ 4 XML Parser	22
4.1 SAX (Simple API for XML)	22
4.2 DOM (Document Object Model)	23
บทที่ 5 โพรโทคอล HTTP (HyperText Transfer Protocol)	26
5.1 ความหมายของ HTTP	26
5.2 โครงสร้าง HTTP (HTTP Structure)	27
5.2.1 บรรทัดเริ่มต้นในการร้องขอ (Initial Request Line)	29
5.2.2 บรรทัดเริ่มต้นในการตอบสนอง (Initial Response Line)	29
5.2.3 เฮดเดอร์ (Header)	31
5.2.4 เมสเสจบอดี้ (message body)	32
5.3 เมธอด (Method)	34
5.3.1 เมธอด GET	34
5.3.2 เมธอด POST	35
บทที่ 6 จาวาเซิร์ฟเลต (Java Servlet)	37
6.1 ความหมายของเซิร์ฟเลต	37
6.2 เซิร์ฟเลตเอนจิน (Servlet engine)	38
6.3 อินเตอร์เฟซ javax.servlet.Servlet	39
6.4 วงจรชีวิตของเซิร์ฟเลต (Servlet's Life Cycle)	40
6.4.1 สถานะเริ่มต้น (Initial State)	40
6.4.2 สถานะให้บริการ (Service State)	41
6.4.3 สถานะทำลาย (Destroy State)	41

สารบัญ (ต่อ)

	หน้า
บทที่ 7 การสร้างและการออกแบบ	42
7.1 ภาพรวมของระบบ	42
7.2 การนำ XML มาใช้เก็บข้อมูลที่เป็นวิทยานิพนธ์	43
7.3 การจัดเก็บข้อมูลที่เป็น XML ลงในดาต้าเบสเซิร์ฟเวอร์	48
7.4 การนำข้อมูลที่จัดเก็บในดาต้าเบสเซิร์ฟเวอร์มาแสดงผล	52
บทที่ 8 การทดลองและผลการทดลอง	53
8.1 หน้าเพจหลักของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์	53
8.2 การป้อนข้อมูลวิทยานิพนธ์ลงในระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์	54
8.2.1 การป้อนข้อมูลทั่วไป	54
8.2.1.1 ทดสอบการป้อนข้อมูลมหาวิทยาลัย	54
8.2.1.2 ทดสอบการป้อนข้อมูลสาขาวิชา	55
8.2.1.3 ทดสอบการป้อนข้อมูลสาขาวิชาย่อย	57
8.2.2 ทดสอบป้อนข้อมูลวิทยานิพนธ์	58
8.3 การค้นหาข้อมูลและการแสดงข้อมูลวิทยานิพนธ์	61
8.4 การทดสอบลบข้อมูลวิทยานิพนธ์	62
บทที่ 9 บทวิจารณ์และสรุป	64
9.1 สรุปผลการดำเนินงาน	64
9.2 บทวิจารณ์	64
9.3 แนวทางในการพัฒนาต่อ	65
ภาคผนวก ก การติดตั้ง DB2 Universal Database Enterprise Edition	66
ภาคผนวก ข การตั้งค่าการเชื่อมต่อกับ Database Server ที่เป็น DB2	70
บรรณานุกรม	77

สารบัญรูปภาพ

หน้าที่

รูปที่ 3-1 แสดงผลลัพธ์การใช้ general เอนทิตีใน Internet Explorer	19
รูปที่ 3-2 แสดงผลลัพธ์การใช้ general เอนทิตีที่สร้างขึ้นเองใน Internet Explorer	20
รูปที่ 4-1 แสดงการทำงานของ parser	22
รูปที่ 4-2 แสดงการทำงานของ parser ที่เป็น DOM	23
รูปที่ 4-3 แสดงลักษณะการมองข้อมูลเป็นทรีใน DOM	24
รูปที่ 4-4 UML class model of DOM Level 2 core interfaces and classed	25
รูปที่ 5-1 การติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์	27
รูปที่ 5-2 ลักษณะข้อมูลที่รับส่งระหว่างไคลเอนต์และเซิร์ฟเวอร์	33
รูปที่ 6-1 หน้าที่และการทำงานของเซิร์ฟเลต	38
รูปที่ 6-2 เซิร์ฟเลตเอนจิน และเซิร์ฟเลตที่อยู่ภายใน	39
รูปที่ 6-3 วงจรชีวิตของเซิร์ฟเลต	40
รูปที่ 6-4 วิธีการจัดการกับร้องขอแบบ GET และ POST ของเซิร์ฟเลต	41
รูปที่ 7-1 ภาพรวมของระบบ	42
รูปที่ 7-2 ER Diagram ของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์	51
รูปที่ 7-3 แสดงการจัดเก็บข้อมูล XML ในฐานข้อมูล	51
รูปที่ 7-4 แสดงการดึงข้อมูลจากดาต้าเบสเซิร์ฟเวอร์	52
รูปที่ 8-1 แสดงหน้าจอหลักของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์	53
รูปที่ 8-2 แสดงหน้าจอการป้อนข้อมูลมหาวิทยาลัย	54
รูปที่ 8-3 แสดงผลของการป้อนข้อมูลมหาวิทยาลัย	55
รูปที่ 8-4 แสดงหน้าจอการป้อนข้อมูลสาขาวิชา	56
รูปที่ 8-5 แสดงผลของการป้อนข้อมูลสาขาวิชา	56
รูปที่ 8-6 แสดงหน้าจอการป้อนข้อมูลสาขาวิชาย่อย	57
รูปที่ 8-7 แสดงผลของการป้อนข้อมูลสาขาวิชาย่อย	58
รูปที่ 8-8 แสดงการป้อนข้อมูลวิทยานิพนธ์	58
รูปที่ 8-9 แสดงการป้อนข้อมูลวิทยานิพนธ์ที่อยู่ในรูปไฟล์ pdf	59
รูปที่ 8-10 แสดงผลการป้อนข้อมูลวิทยานิพนธ์	60
รูปที่ 8-11 แสดงหน้าจอการค้นหาข้อมูลวิทยานิพนธ์	60
รูปที่ 8-12 แสดงผลการค้นหาข้อมูลวิทยานิพนธ์	61
รูปที่ 8-13 แสดงข้อมูลวิทยานิพนธ์ที่อยู่ในรูปแบบของไฟล์ pdf	62
รูปที่ 8-14 แสดงหน้าจอการลบข้อมูลวิทยานิพนธ์	62
รูปที่ 8-15 แสดงผลการลบข้อมูลวิทยานิพนธ์	63

สารบัญรูปภาพ (ต่อ)

หน้าที่

รูปที่ ก-1 แสดงการติดตั้ง DB2 Universal Database Enterprise Edition Version 7.1	66
รูปที่ ก-2 แสดงการเลือก DB2 Products ที่ต้องการติดตั้ง	67
รูปที่ ก-3 แสดงการเลือกประเภทที่ต้องการติดตั้ง	67
รูปที่ ก-4 แสดงการเลือกตำแหน่งดิสก์ที่จะทำการติดตั้ง	68
รูปที่ ก-5 แสดงการตั้งค่า Username และ Password ของผู้ดูแลระบบ	68
รูปที่ ก-6 แสดงการยืนยันการติดตั้ง	69
รูปที่ ก-7 แสดงการเสร็จสิ้นการติดตั้ง	69
รูปที่ ข-1 แสดงการเข้าสู่โปรแกรม Client Configuration Assistant	70
รูปที่ ข-2 แสดงโปรแกรม Client Configuration Assistant	71
รูปที่ ข-3 แสดงวิธีการตั้งค่าการเชื่อมต่อ	71
รูปที่ ข-4 แสดงการเลือกฐานข้อมูล	72
รูปที่ ข-5 แสดงการ Add System	72
รูปที่ ข-6 แสดงการเลือกฐานข้อมูล	73
รูปที่ ข-7 แสดงการตั้งค่า Database alias ให้กับฐานข้อมูลที่จะติดต่อ	73
รูปที่ ข-8 แสดงการ Register ฐานข้อมูลใน ODBC	74
รูปที่ ข-9 แสดงหน้าต่างการ Confirmation	74
รูปที่ ข-10 แสดงการทดสอบการเชื่อมต่อ	75
รูปที่ ข-11 แสดงการผลการทดสอบของการเชื่อมต่อฐานข้อมูล	75
รูปที่ ข-12 แสดงการฐานข้อมูลที่สามารถเชื่อมต่อได้	76

สารบัญตาราง

หน้าที่

ตารางที่ 3-1 แสดงสัญลักษณ์ต่างๆใช้ในการประกาศอิลิเมนต์	17
ตารางที่ 3-2 สรุปค่าต่างๆของ TYPE ใน Attributes	21
ตารางที่ 3-3 สรุปค่าต่างๆของ DEFAULT_VALUE ที่อยู่ใน Attributes	21
ตารางที่ 7-1 ความหมายของอิลิเมนต์ใน XML	44
ตารางที่ 7-2 ข้อมูลสาขาวิชาของวิทยานิพนธ์	49
ตารางที่ 7-3 ข้อมูลสาขาวิชาย่อยของวิทยานิพนธ์	49
ตารางที่ 7-4 ข้อมูลสำนักพิมพ์หรือสถาบันการศึกษาที่เป็นเจ้าของวิทยานิพนธ์	49
ตารางที่ 7-5 ข้อมูลผู้จัดทำวิทยานิพนธ์	49
ตารางที่ 7-6 ข้อมูลทั่วไปของวิทยานิพนธ์	50
ตารางที่ 7-7 ข้อมูลความสัมพันธ์ระหว่างวิทยานิพนธ์และผู้จัดทำวิทยานิพนธ์	50
ตารางที่ 7-8 ข้อมูลคำสำคัญในวิทยานิพนธ์	50



บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันมีปริมาณนิพนธ์เป็นจำนวนมากที่จัดทำขึ้นในประเทศไทย ซึ่งมีผู้คิดที่จะรวบรวม ปริมาณนิพนธ์เหล่านี้เอาไว้ เพื่อเป็นศูนย์รวมของข้อมูลและเพื่อเป็นการสนับสนุนผลงานวิจัยของไทย การเก็บรวบรวมปริมาณนิพนธ์ของนักศึกษาในรูปแบบดิจิทัลจึงเกิดขึ้น นั่นคือ ห้องสมุดวิทยานิพนธ์ อิเล็กทรอนิกส์ และปัจจุบันได้มีผู้ที่รวบรวมปริมาณนิพนธ์และจัดเก็บปริมาณนิพนธ์ในรูปแบบดิจิทัลในแบบ ต่างๆ เป็นจำนวนอยู่ไม่น้อย ดังนั้นเพื่อเป็นการพัฒนาการจัดเก็บปริมาณนิพนธ์ในรูปแบบดิจิทัล ให้ง่าย ต่อการใช้งานนั่นคือ ง่ายต่อจัดเก็บ ง่ายต่อการเพิ่มปริมาณนิพนธ์เข้าไปในห้องสมุด หรือแม้แต่การค้นหา ปริมาณนิพนธ์ที่ผู้ค้นหาต้องการ

ปัจจุบันมีเทคโนโลยีที่ใช้ในการรวบรวมข้อมูลให้เป็นฐานข้อมูลอยู่มาก ซึ่ง XML(Extensible Markup Language) เป็นภาษาหนึ่งที่กำลังเป็นที่นิยมในการเก็บข้อมูล เนื่องจาก XML เป็นภาษามาตรฐาน ที่ใช้ในการสร้างข้อมูลและสามารถอธิบายข้อมูลให้ โดยที่ทุกๆ Application สามารถที่จะเข้าใจข้อมูลที่อยู่ใน รูปแบบของเอกสารของ XML รวมถึงผู้ที่มาดูเอกสาร XML ก็สามารถที่จะเข้าใจถึงข้อมูลเหล่านี้ได้ เนื่องจากเอกสาร XML เป็นข้อมูลที่เป็นตัวอักษรและสามารถสื่อความหมายของข้อมูลได้ด้วยการให้ความ หมายของข้อมูลที่คนทั่วไปสามารถเข้าใจได้

นอกจากภาษา XML ที่ใช้ในการจัดเก็บข้อมูลแล้ว โครงการนี้ยังได้ใช้ภาษาจาวา ซึ่งจาวาเป็น ภาษาที่ใช้ในการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) ที่ได้รับความนิยมเป็นอย่างมาก เนื่องจากความสามารถในด้านที่เป็นอิสระไม่ขึ้นกับแพลตฟอร์มใดๆ (Platforms Independents) โดยโครง งานนี้ได้ใช้ภาษาจาวาในการจัดการกับข้อมูลในเอกสาร XML รวมถึงการค้นหาปริมาณนิพนธ์

เพื่อการพัฒนาห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ ให้มีความสามารถในการใช้งานที่เป็นสากล คือสามารถทำงานได้ในทุกๆ แพลตฟอร์ม และ ทุกๆ Application ที่ต้องการที่จะใช้ข้อมูลก็สามารถมาใช้ ได้ เพราะความสามารถของ XML จึงได้นำ XML และภาษาจาวา มาใช้ในการสร้างห้องสมุดวิทยานิพนธ์ อิเล็กทรอนิกส์

1.2 วัตถุประสงค์ของปริมาณนิพนธ์

1.2.1 เพื่อสร้างระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ ให้สามารถทำงานในฟังก์ชันพื้นฐานได้

1.2.2 เพื่อเป็นแนวทางในการพัฒนาห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ ในระดับที่สูงและ ซับซ้อนขึ้นไปอีก

1.2.3 ศึกษาโครงสร้างและรายละเอียดของ XML และศึกษาการสร้างโปรแกรมประยุกต์ สำหรับ XML เพื่อให้สามารถสร้างโปรแกรมประยุกต์สำหรับ XML ได้

1.2.4 เพื่อให้ นักศึกษาสามารถประยุกต์ใช้ความรู้ต่างๆ ที่ได้ศึกษามาช่วยในการแก้ปัญหาและพัฒนาระบบงาน

1.3 ขอบเขตของปริญญาพนธ์

งานวิจัยที่ได้ทำขึ้นมานี้ ต้องการที่จะศึกษาการจัดเก็บข้อมูลด้วย XML และนำมาประยุกต์ใช้งานกับฐานข้อมูลที่เป็นแบบ Relational อีกทั้งศึกษาเทคโนโลยีที่เกี่ยวข้องกับ XML เพื่อที่จะสามารถนำมาใช้งานร่วมกันในงานวิจัย และสามารถเขียนโปรแกรมภาษาจาวาเซิร์ฟเลตซึ่งประมวลผลบนฝั่งเซิร์ฟเวอร์ให้ทำการติดต่อกับฐานข้อมูลซึ่งอยู่ในดาต้าเบสเซิร์ฟเวอร์ (Database Server) ผ่าน JDBC API

สร้างระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ โดยให้มีการบริการพื้นฐานเช่น การใส่ข้อมูล, ค้นหาข้อมูล, แสดงข้อมูล, แก้ไขข้อมูล และลบข้อมูลวิทยานิพนธ์ ผ่านทางเว็บเบราว์เซอร์ (Web Browser) ที่เชื่อมต่อกับเครือข่ายอินเทอร์เน็ต

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงการนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็เริ่มจากศึกษารายละเอียดของภาษา XML จากนั้นทำการศึกษาเทคโนโลยีที่เกี่ยวข้องกับภาษา XML ซึ่งก็มีการสร้างเอกสาร DTD เพื่อใช้งานร่วมกับเอกสาร XML ศึกษาวิธีการนำเอกสาร XML ไปผ่านกระบวนการ Parsing เพื่อนำข้อมูลในเอกสาร XML ออกมาใช้ จากนั้นทำการศึกษาโปรโตคอล HTTP ที่ใช้ในการสื่อสารระหว่างเว็บเบราว์เซอร์กับเว็บเซิร์ฟเวอร์ อีกทั้งศึกษาการเขียนจาวาเซิร์ฟเลตเพื่อทำงานบนฝั่งเซิร์ฟเวอร์และใช้ในการติดต่อกับฐานข้อมูล ศึกษาการติดต่อและการใช้งานตัวจัดการฐานข้อมูล (DBMS) ที่จะใช้ในระบบ จากนั้นทำการศึกษาวิธีการแมปข้อมูลจากเอกสาร DTD เป็น Database Schema จากนั้นก็นำความรู้ที่ได้ศึกษามาทั้งหมดมาออกแบบสถาปัตยกรรมของระบบ และเริ่มพัฒนาระบบตามที่ได้ออกแบบได้ หลังจากนั้นเป็นส่วนของการทดสอบการทำงานของระบบ และตรวจหาข้อบกพร่องของระบบเพื่อนำกลับมาแก้ไขให้ดีขึ้น

บทที่ 2

Extensible Markup Language (XML)

2.1 บทนำ

XML ถูกกำหนดโดยกลุ่มทำงาน XML ของสถาบัน World Wide Web Consortium (W3C) กลุ่มทำงานกลุ่มนี้ได้บรรยายถึงภาษา XML ไว้ว่า “Extensible Markup Language (XML) เป็นเซตย่อยของ Standard Generalized Markup Language (SGML) จุดมุ่งหมายคือให้สามารถใช้ SGML ในการบริการรับข้อมูลและประมวลผลบนเครือข่ายอินเทอร์เน็ตซึ่งปัจจุบันสามารถทำได้ด้วยภาษา HTML เท่านั้น XML ถูกออกแบบมาเพื่อให้เป็นวิธีการที่ง่ายและสามารถประมวลผลร่วมกันทั้ง SGML และ HTML”

เนื่องจากอิลิเมนต์ของ XML ไม่มีความหมายจริงๆ จึงเป็นสาเหตุที่ทำให้ XML เป็นภาษาที่กำลังมาเป็นภาษาที่มีความสำคัญ เพราะ XML เป็นซับเซตของภาษา SGML (มากกว่าที่จะเป็นแอปพลิเคชันของ SGML) และสามารถจะถูกใช้ได้ทั้งสำหรับเก็บข้อมูลและเป็นพื้นฐานสำหรับแอปพลิเคชันที่ใช้ XML เช่น XML Stylesheet Language (XSL)

เอกสาร XML (XML document) สามารถเก็บข้อมูลโดยใช้อิลิเมนต์ที่มีความหมายอย่างที่ต้องการให้มันเป็นอย่างแท้จริง แม้ว่าตอนนี้แนวความคิดนี้จะค่อนข้างใหม่แต่นี้ก็เป็นข้อได้เปรียบประการหลัก หากผู้ส่งและผู้รับ รู้ความหมายของอิลิเมนต์นั้น เอกสาร XML สามารถถูกใช้ในการเก็บและโอนย้ายข้อมูลระหว่างแพลตฟอร์ม, ระบบปฏิบัติการและอุปกรณ์ที่เป็นอิสระจากกันอย่างแท้จริง และเนื่องจากมันเป็นแค่เท็กซ์เท่านั้น มันจึงสามารถส่งผ่านเครือข่ายชนิดต่างๆ ได้อย่างง่าย รวมถึงการส่งผ่านโพรโตคอล HTTP ในการติดต่อสื่อสารของเครือข่ายอินเทอร์เน็ตด้วย ด้วยเหตุนี้ XML จึงเป็นเวอร์ชันของ SGML ที่ใช้งานง่ายและเหมาะสำหรับเว็บ เช่นเดียวกับ SGML ตรงที่ XML ให้คุณออกแบบชุดอิลิเมนต์ของคุณเองเมื่อต้องการอธิบายเอกสารที่เป็นเรื่องเฉพาะเหมือนกับ SGML รูปแบบที่เป็นเรื่องเฉพาะซึ่งเป็นคำจำกัดความ โปรแกรมประยุกต์ XML (หรือเรียกอีกอย่างว่า Vocabulary)

2.2 XML คืออะไร

XML เป็นภาษามาตรฐานที่ใช้โครงสร้างในการอธิบายข้อมูลซึ่งสามารถทำให้แอปพลิเคชันที่แตกต่างกันสามารถเข้าใจกันได้ โดยใช้แท็กเป็นตัวกำหนดความหมายของข้อมูลที่อยู่ในแท็กนั้นๆ ข้อมูลที่เก็บในไฟล์ XML นั้นจะอยู่ในรูป hierarchical structure เมื่อพิจารณาตัวอย่าง HTML ต่อไปนี้

```
<p><b>Mrs. Mary McGoon</b>
```

```
<br>
```

```
1404 Main Street
```

```
<br>
```

```
Anytown, NC 34829</p>
```

เมื่อนำไปเปิดด้วยเว็บเบราว์เซอร์จะแสดงผลดังนี้

Mrs. Mary McGoon

1404 Main Street

Anytown, NC 34829

จะเห็นได้ว่า HTML เป็นภาษาที่อธิบายถึงวิธีการแสดงผล (render) ของอิลิเมนต์ ไม่มีข้อมูลใดระบุถึงความหมายของข้อมูลนั้นๆ แต่ถ้าข้อมูลข้างต้นเขียนในรูปแบบของ XML จะเขียนได้เป็น

```
<address>
<name>
<title>Mrs.</title>
<first-name>Mary</first-name>
<last-name>McGoon</last-name>
</name>
<street>1401 Main Street</street>
<city>Anytown</city>
<state>MC</state>
<zipcode>34829</zipcode>
</address>
```

จะเห็นได้ว่าเราสามารถสร้างแท็กที่แสดงถึงความหมายของข้อมูลที่อยู่ภายในแท็กนั้น และสามารถเขียนแอปพลิเคชันที่สามารถดำเนินการกับข้อมูลได้ง่ายกว่า และสามารถนำข้อมูลไปใช้งานได้หลายวัตถุประสงค์มากกว่า เช่น ในการแสดงผลอาจจะนำข้อมูลเดียวกันไปแสดงผลได้หลายประเภทตามที่ต้องการได้ ซึ่งเป็นคุณสมบัติที่สำคัญของ XML ที่สามารถเก็บสารสนเทศไว้ในเอกสารได้

2.3 การสร้าง Well-Formed ให้เอกสาร XML

เนื่องจากเอกสาร XML เข้มงวดเรื่อง syntax ในเอกสารมาก ดังนั้นการที่จะสร้างเอกสาร XML นั้นต้องทำให้ถูกต้องตามกฎเกณฑ์ที่ XML ได้กำหนดขึ้น เอกสาร XML ที่สามารถนำไปใช้งานได้นั้นจะถูกรเรียกว่า Well-Formed ซึ่งมีรายละเอียดต่างๆ ในการสร้างดังต่อไปนี้

2.3.1 Well-Formed ของเอกสาร XML

การที่เอกสาร XML จะเป็นเอกสารที่ Well-formed นั้น ขั้นแรกเอกสารจะต้องประกอบด้วย 3 ส่วนคือ

1. Prolog ซึ่งจะประกอบด้วย การประกาศ XML(XML Declaration) เช่น
`<?xml version="1.0"?>`
2. อิลิเมนต์รูต (Root Element) เป็นอิลิเมนต์ที่อยู่นอกที่สุด ซึ่งภายในรูตนี้ก็จะเก็บอิลิเมนต์อื่นอีกด้วย นั่นคือเอกสารที่ Well-formed จะต้องประกอบด้วยอิลิเมนต์รูตหนึ่งอิลิเมนต์เท่านั้น นั่นหมายความว่า ทุกๆอิลิเมนต์ในเอกสาร XML จะต้องอยู่ในอิลิเมนต์รูต
3. Optional miscellaneous ซึ่งจะประกอบด้วย Comments, Processing Instructions

ในหัวข้อต่อไปจะเป็นรายละเอียดของการที่จะทำให้เอกสาร XML เป็นเอกสารที่ Well-formed ซึ่งจะเกี่ยวกับเรื่องต่างๆ ดังต่อไปนี้ กฎเกณฑ์(Syntax Rules) ของ XML กฎการตั้งชื่อ(Naming Rules) กฎการซ้อนทับ(Nesting Rules) เป็นต้น

2.3.2 การมาร์กอัพและข้อมูลที่เป็นตัวอักษร (Markup and Character Data)

เอกสาร XML นั้นได้สร้างขึ้นจากการมาร์กอัพ (Markup) และข้อมูลที่เป็นตัวอักษร (Character data) ซึ่งการมาร์กอัพในเอกสารนั้นจะทำให้เอกสาร XML เกิดโครงสร้างขึ้น การมาร์กอัพนั้นประกอบด้วย แท็กเริ่มต้น (start tag), แท็กลงท้าย (end tag), แท็กว่าง (empty tag), Comment, Processing Instruction (PI), Document Type Declarations (DTD), ส่วนข้อมูลที่เป็นตัวอักษรนั้นก็คือ ตัวอักษรทั้งหมดที่ไม่ใช่ส่วนที่เป็นมาร์กอัพ ตัวอย่างของการใช้มาร์กอัพและข้อมูลที่เป็นตัวอักษร มีดังต่อไปนี้

```
<?xml version="1.0"?>
<DOCUMENT>
  <GREETING>
    Hello from XML
  </GREETING>
  <MESSAGE>
    Welcome to the wild and wooly world of XML
  </MESSAGE>
</DOCUMENT>
```

แท็กจะต้องเริ่มด้วย “<” และลงท้ายด้วย “>” ดังนั้นแท็กในการมาร์กอัพก็คือ

`<?xml version="1.0"?>`, `<DOCUMENT>`, `<GREETING>`, `<MESSAGE>` และส่วนที่เป็นข้อมูลที่เป็นตัวอักษร คือ Hello From XML และ Welcome to the wild and wooly world of XML หัวข้อต่อไปจะเป็นการกล่าวถึงรายละเอียดของโครงสร้างเอกสาร XML ซึ่งมีเนื้อหาดังต่อไปนี้

2.3.3 การประกาศโปรล็อก (The Prolog)

Prolog จะเป็นส่วนหัวของเอกสาร XML ซึ่งก็ไม่กำหนดไว้อย่างแน่นอน แต่อย่างน้อยควรมีส่วนที่เป็นการประกาศ XML (XML Declaration) โดยทั่วไปแล้ว Prolog นั้นจะประกอบด้วย XML Declaration, Comment, Processing Instruction, Whitespace และ Document Type Declaration ดังตัวอย่างด้านล่างนี้

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="greeting.css"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (CUSTOMER)*>
  <!ELEMENT CUSTOMER (NAME,DATE,ORDERS)>
  <!ELEMENT NAME (LAST_NAME,FIRST_NAME)>
  <!ELEMENT LAST_NAME (#PCDATA)>
  <!ELEMENT FIRST_NAME (#PCDATA)>
  <!ELEMENT DATA (#PCDATA)>
  <!ELEMENT ORDERS (ITEM)*>
  <!ELEMENT ITEM (PRODUCT,NUMBER,PRICE)>
  <!ELEMENT PRODUCT (#PCDATA)>
  <!ELEMENT NUMBER (#PCDATA)>
  <!ELEMENT PRICE (#PCDATA)>
]>
<!-- ***** end of Prolog ***** -->
<DOCUMENT>
...
...
```

3.3.4 การประกาศ XML (The XML Declaration)

เอกสาร XML ต้องเริ่มด้วยการประกาศ XML (XML Declaration) ซึ่งจะเป็นการแสดงว่าเอกสารนั้นถูกเขียนเป็นเอกสาร XML และควรประกาศเป็นบรรทัดแรกของเอกสาร ดังตัวอย่างด้านล่างนี้

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
```

การประกาศเอกสาร XML จะใช้โอลิเมนต์ `<?xml?>` ในการประกาศว่าเอกสารนี้เป็นเอกสารของ XML และใน `<?xml?>` สามารถที่จะมีแอตทริบิวต์ได้ดังต่อไปนี้

- version เป็นตัวบอกว่าใช้ XML version ไหน ในที่นี้ก็จะใช้ version 1.0 ซึ่งต้องใช้แอตทริบิวต์นี้เสมอเมื่อมีการประกาศเอกสารว่าเป็นเอกสารของ XML

- encoding เป็นแอตทริบิวต์ที่ใช้บอกว่าจะเข้ารหัสเอกสารเป็นแบบไหน
- standalone เป็นแอตทริบิวต์ที่ใช้บอกว่าเราต้องการอ้างอิงข้อมูลหรือใช้ข้อมูลจากภายนอกหรือไม่ ถ้ามีค่า “yes” แสดงว่าไม่ใช่ และถ้า “no” แสดงว่าใช่

2.3.5 การเขียนคอมเมนต์ (Comments)

การสร้างหมายเหตุในเอกสาร XML(XML Comment) จะเหมือนกับการ Comment ใน HTML ซึ่งแสดงถึงข้อความที่เป็นหมายเหตุพิเศษ เพื่อให้ผู้เขียนเอกสารสามารถทำความเข้าใจ และง่ายต่อการสืบค้นในภายหลัง ซึ่งจะมีในเอกสารหรือไม่ก็ได้ และไม่ถือว่าเป็นส่วนจริงของเอกสาร นั่นหมายความว่า XML Parser จะไม่สนใจ และสามารถที่จะใส่ลงในส่วนใดก็ได้ โดยข้อความหมายเหตุต้องอยู่ภายใต้เครื่องหมาย “<!--” และ “-->”

การ Comment มีกฎดังต่อไปนี้

- ห้ามประกาศ Comment ก่อนการประกาศเอกสาร XML(XML Declaration)
- ห้ามใช้ Comment ภายในแท็ก เช่น

```
<DOCUMENT <!--Start Document -->
```
- ห้ามใช้ -- ภายใน Comment เพราะ Parser จะดูเหมือนเป็นการจบ Comment ซึ่งจะ
 ทำให้ผิด
- สามารถใช้ Comment ปิดส่วนของแท็กที่ไม่ต้องการใช้ได้

2.3.6 การใช้งาน Processing Instructions

เป็นคำสั่งที่ XML Processor จะต้องเข้าใจและสามารถทำตามคำสั่งนั้นได้ โดยคำสั่งจะเริ่มต้นด้วย “<?” และปิดท้ายด้วย “>” เช่นคำสั่ง `<?xml-stylesheet?>` เป็นคำสั่งที่ใช้เชื่อมต่อ เอกสาร XML กับ Style Sheet ของเอกสาร XML นั้น

ในส่วนที่เป็นรายละเอียดของ Prolog ได้แสดงครบแล้ว ไม่ว่าจะเป็น XML Declaration Processing Instruction และ Comment ยกเว้น DTD ซึ่งจะขออธิบายไว้ในบทต่อไป และในหัวข้อต่อไปจะขอกล่าวถึงรายละเอียดของโครงสร้างของเอกสาร XML คือการสร้างแท็กและอิลิเมนต์

2.3.7 แท็กและอิลิเมนต์ (Tags and Element)

การที่จะทำให้เอกสาร XML มีโครงสร้างได้นั้น สามารถทำได้โดยใช้การ Markup ซึ่งจะประกอบด้วยส่วนที่เรียกว่า อิลิเมนต์(Element) โดย XML อิลิเมนต์นั้นก็ประกอบด้วย แท็กเริ่มต้น(start tag) และแท็กปิดท้าย(end tag) ยกเว้นอิลิเมนต์นั้นถูกกำหนดเป็นอิลิเมนต์ว่างเปล่า(empty Element) จะมีแท็กเดียว แท็กเริ่มต้น หรือเรียกอีกอย่างหนึ่งว่า Opening tag จะเริ่มต้นด้วยเครื่องหมาย “<” และลงท้ายด้วยเครื่องหมาย “>” สำหรับแท็กปิดท้ายจะเริ่มต้นด้วยเครื่องหมาย “</” และปิดท้ายด้วยเครื่องหมาย “>”

2.3.7.1 ชื่อแท็ก (Tag Names)

XML มีข้อกำหนดเกี่ยวกับการตั้งชื่อแท็กดังต่อไปนี้ โดยอักษรตัวแรกของชื่อแท็กต้องเป็นตัวอักษร underscore และ colon เท่านั้น ส่วนตัวถัดไปสามารถเป็นได้ทั้งตัวอักษร, ตัวเลข, underscore, hyphens, periods และ colon โดยต้องไม่มีการเว้นวรรค ดังตัวอย่างต่อไปนี้

```
<DOCUMENT>
<document>
<_Record>
<customer>
```

จากตัวอย่างจะเห็นได้ว่า แท็ก <DOCUMENT> และแท็ก <document> จะมีค่าไม่เท่ากัน เนื่องจาก XML Processor จะอยู่ใน Case-sensitive และจากแท็กข้างต้นนี้ จะต้องทำการปิดด้วยแท็กดังต่อไปนี้

```
</DOCUMENT>
</document>
</_Record>
</customer>
```

2.3.7.2 อิลิเมนต์ว่าง (Empty Element)

อิลิเมนต์ว่างนั้นจะประกอบด้วยแท็กเพียงแท็กเดียว ซึ่งจะไม่มีการเริ่มต้นและปิดท้าย เหมือนอย่างใน HTML เช่นแท็ก , และ
 เป็นต้น และใน XML อิลิเมนต์นั้นสามารถเขียนได้เป็น <ชื่ออิลิเมนต์ /> ดังตัวอย่างต่อไปนี้

```
<?xml version = "1.0"?>
<document>
    <greeting text = "Hello from XML" />
</document>
```

2.3.7.3 อิลิเมนต์ราก (The Root Element)

เอกสาร XML ที่ Well-formed นั้น จะต้องมียิลิเมนต์ตัวหนึ่งที่ภายในอิลิเมนต์เก็บทุกๆ อิลิเมนต์ โดยเรียกอิลิเมนต์นี้ว่าอิลิเมนต์รากซึ่งเป็นส่วนที่สำคัญมากของเอกสาร XML โดยเฉพาะในส่วนของการพาร์ส (parse) เอกสาร XML ซึ่งจะเริ่มอิลิเมนต์รากก่อน ตัวอย่างของอิลิเมนต์รากก็เช่น อิลิเมนต์<DOCUMENT> ด้านล่าง

```
<?xml version = "1.0"?>
<DOCUMENT>
    <CUSTOMER>
        <NAME>
```

```

<LAST_NAME></LASTNAME>
<FIRST_NMAE></ FIRST_NMAE >

</NAME>

<DATE>

</DATE>

</CUSTOMER>

</DOCUMENT>

```

2.3.7.4 แอตทริบิวต์ (Attributes)

รูปแบบของแอตทริบิวต์ของเอกสาร XML ก่อนข้างที่จะเหมือนกับของ HTML คือจะเป็นคู่ของชื่อและค่าของแอตทริบิวต์ ซึ่งจะเป็นการกำหนดข้อมูลเพิ่มให้กับทั้งแท็กเริ่มต้น และแท็กว่าง โดยมีการกำหนดค่าของแอตทริบิวต์ในเครื่องหมาย “ ” มีรูปแบบการเขียนดังต่อไปนี้

ชื่อแอตทริบิวต์ = “ค่าแอตทริบิวต์”

ตัวอย่างด้านล่างจะเป็นตัวอย่างของการใช้แอตทริบิวต์ เมื่อเราเพิ่มแอตทริบิวต์ STATUS ให้กับอิลิเมนต์ของ <CUSTOMER>

```

<?xml version = “1.0”?>
<DOCUMENT>
  <CUSTOMER STATUS = “Good Credit”>
    <NAME>
      <FIRST_NAME>Sam</FIRST_NAME>
      <LAST_NAME>Smith</LAST_NAME>
    </NAME>
    .....
    .....
  </CUSTOMER>
</DOCUMENT>

```

เกิดคำถามขึ้นว่าเมื่อไรที่ต้องเก็บข้อมูลไว้ที่แอตทริบิวต์ และเมื่อไรที่ต้องเก็บข้อมูลไว้ที่อิลิเมนต์ การเก็บข้อมูลไว้ที่แอตทริบิวต์นั้นจะทำให้ข้อมูลไม่มีโครงสร้าง และถ้าเก็บข้อมูลไว้ที่แอตทริบิวต์มากเกินไปจะทำให้อ่านข้อมูลได้ยากเกินไป

2.3.7.4.1 ชื่อแอตทริบิวต์ (Attribute Name)

กฎการตั้งชื่อของแอตทริบิวต์นั้นจะคล้ายกับกฎการตั้งชื่อของอีลิเมนต์ คืออักษรแรกต้องเริ่มด้วยตัวอักษร, underscore, หรือ colon และอักษรตัวต่อไปอาจจะเป็นตัวอักษร, underscore, hyphens period และ colon และห้ามมีการเว้นวรรคระหว่างชื่อแอตทริบิวต์ ดังตัวอย่างด้านล่างต่อไปนี้จะเป็นการตั้งชื่อแอตทริบิวต์ที่ถูกต้อง

```
<circle origin_x= "1.0" origin_y= "20.0" radius= "10.0"/>
<image src= "image.jpg">
<pen color= "red" width= "5">
<book pages= "1231">
```

ตัวอย่างการตั้งชื่อแอตทริบิวต์ที่ผิด

```
<circle lorigin_x= "1.0" lorigin_y= "20.0" lradius= "10.0"/>
<image src name= "image.jpg">
<pen color@= "red" width@= "5">
<book pages(important)= "1231">
```

2.3.7.4.2 ค่าแอตทริบิวต์ (Attribute Values)

เนื่องจากการมาร์กอัปเกิดเป็นตัวหนังสือ ดังนั้นแอตทริบิวต์ก็จะเป็นเพียงตัวหนังสือเช่นกัน และค่าของแอตทริบิวต์ปกติจะอยู่ภายใต้เครื่องหมาย “ ” ถึงแม้ว่าเราจะให้ค่าของแอตทริบิวต์เป็นตัวเลขก็ตาม

```
<circle lorigin_x= "1.0" lorigin_y= "20.0" lradius= "10.0"/>
```

เหมือนตัวอย่างนี้ XML Processor จะให้ค่าของแอตทริบิวต์เป็นสตริงเท็กซ์ ซึ่งถ้าเราต้องการใช้ข้อมูลที่เป็นตัวเลขจะต้องไปเปลี่ยนที่ภาษาที่เข้าใช้ข้อมูลของเอกสาร XML เองเช่น ภาษาจาวา เป็นต้น

สามารถใช้เครื่องหมาย ‘ ’ ในการกำหนดค่าของแอตทริบิวต์ได้เมื่อต้องการที่จะเครื่องหมาย “ ” ภายในค่าของแอตทริบิวต์อีกครั้งหนึ่ง เช่น

```
<quotation text= ‘He said, “Not that!”’>
```

หรือต้องการที่จะใช้เครื่องหมาย ” เพียงตัวเดียว XML Processor ก็ได้กำหนดให้สามารถใช้ ' แทนเครื่องหมาย “ ดังตัวอย่าง

```
<person height= "5&apos;6&apos;">
```

2.3.8 การสร้างโครงสร้างเอกสารให้เป็น Well-Formed

ในการสร้างเอกสาร XML ให้เป็นเอกสารที่ Well-formed นั้นมีข้อกำหนดมากมายเกี่ยวกับโครงสร้างของเอกสาร XML ที่จะทำให้ตัวเอกสารนั้น Well-formed ซึ่งจะขอล่าเป็นหัวข้อดังต่อไปนี้

1. กฎข้อแรกของโครงสร้างเอกสารที่ Well-formed คือต้องมีการประกาศว่าเป็นเอกสารของ XML (XML Declaration) ซึ่งในทางปฏิบัติเราไม่ต้องประกาศก็ได้ แต่ถ้าเราประกาศจะทำให้เอกสารของเรานั้นเป็นเอกสารที่ Well-formed
2. เอกสาร XML ที่ Well-formed จะต้องประกอบด้วยอย่างน้อยอิลิเมนต์ 1 อิลิเมนต์ ซึ่งนั่นก็คืออิลิเมนต์รูต และสำหรับทุกๆ อิลิเมนต์จะต้องถูกเก็บภายในอิลิเมนต์รูตนี้
3. สำหรับทุกๆ อิลิเมนต์ต้องประกอบด้วยแท็กเริ่มต้นและแท็กปิด ลงท้ายเสมอ ยกเว้นแท็กว่าง ซึ่งทั้งแท็กเปิดและแท็กเริ่มต้นจะต้องเหมือนกันด้วย(case-sensitive)
4. ต้องปิดแท็กว่างด้วยเครื่องหมาย /> เสมอ เนื่องจากแท็กนี้ไม่มีข้อมูลที่เป็นส่วนของ CDATA ดังนั้นจึงไม่มีการปิดแท็กด้วยแท็กปิด
5. ทำตามกฎการซ้อนทับที่ถูกต้อง (Nest Element) ซึ่งเอกสารที่ Well-formed นั้นต้องทุกๆ อิลิเมนต์ต้องอยู่ในคู่ของอิลิเมนต์ที่ถูกต้อง (Correctly Nest) ดังตัวอย่างด้านล่างเป็นเอกสารที่อยู่ในกฎการซ้อนทับที่ถูกต้อง

```
<?xml version = "1.0" encoding = "UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
  <MESSAGE>
    Welcome to the wild and wooly world of XML
  </MESSAGE>
</DOCUMENT >
```

ตัวอย่างด้านล่างนี้เป็นตัวอย่างของเอกสารที่ไม่ถูกต้องตามกฎการซ้อนทับ

```
<?xml version = "1.0" encoding = "UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  <MESSAGE>
    Welcome to the wild and wooly world of XML
  </GREETING>
</MESSAGE>
</DOCUMENT >
```

6. ชื่อของแอตทริบิวต์ต้องไม่ซ้ำ ซึ่งจะผิดถ้าทำเช่นนี้

```
<PERSON LAST_NAME="WESTER" LAST_NAME="JONE">
```

เนื่องจาก XML เป็นกรณีที่ case-sensitive ซึ่งถ้าทำเช่นนี้จะถูก

```
<PERSON LAST_NAME="WESTER" last_name="JONE">
```

2.3.9 CDATA Sections

CDATA Sections จะเป็นส่วนของข้อมูลที่เป็น Character โดยที่ XML Processor จะไม่พาร์สข้อมูลในส่วนนี้ โดยที่ CDATA Sections จะเริ่มด้วย `<![CDATA` และปิดด้วยเครื่องหมาย `]]>` ซึ่งข้อมูลที่อยู่ภายใต้เครื่องหมายนี้ XML Processor จะทำการค้นหาข้อมูลก็จริงแต่ก็เป็นเพียงการค้นหาเครื่องหมาย `]]>` เพื่อที่จะไม่พาร์สข้อมูลในส่วนนี้ ซึ่งจะมีประโยชน์มากที่สามารถทำให้มีส่วนที่เป็นข้อมูลจริงๆ ในเอกสาร XML ที่ไม่ถูกพาร์ส

2.3.10 XML Namespaces

เนื่องจาก XML สามารถที่จะให้เรากำหนดแท็กที่ใช้งานเองได้ แต่ในปัจจุบันมีผู้สร้างเอกสาร XML มากมายซึ่งอาจจะทำให้มีผู้ที่จะกำหนดแท็กที่ซ้ำกับแท็กที่เราสร้างขึ้นก็ได้ ดังนั้นการกำหนดให้แท็กที่ใช้กันไม่ซ้ำกัน วิธีที่จะแก้ปัญหา ก็การใช้ Namespace โดย namespace จะเป็นการรับประกันว่าถึงแม้จะมีแท็กที่ตั้งชื่อที่เหมือนกันแล้วก็ตาม แต่ทั้ง 2 แท็กนั้นจะไม่ซ้ำกัน

3.3.10.1 การสร้าง namespace (Creating a Namespace)

เพื่ออำนวยความสะดวก จะขอสร้าง namespace จากตัวอย่างต่อไปนี้

```
<library>
  <book>
    <title>
      Earthquakes for lunch
    </title>
  </book>
</library>
```

ดังนั้นเพื่อที่จะสร้าง namespace นั้นเราจะใช้ xmlns:prefix แดคทริบิวต์ในการสร้าง namespace ใหม่ ซึ่งจะต้องไม่ซ้ำกันด้วย (unique) โดย prefix ก็คือสิ่งที่จะใช้นำหน้า (prefix) อีลิเมนต์ของเรา ซึ่งเราต้องใช้พร้อมกับ URI หรือต้องอ้างอิงโดยตรงกับ URI ของ XML Processor เมื่อกำหนดแล้วเราก็สามารถใช้ได้เป็นดังนี้

```
<library>
  <xmlns:book="http://www.amazingbooks.com/spec">
    <book:book>
      <book:title>
        Earthquakes for lunch
      </book:title>
    </book:book>
  </library>
```



บทที่ 3

Document Type Definition (DTD)

3.1 การตรวจสอบความถูกต้องของเอกสาร XML

ความถูกต้องของเอกสาร XML จะมีอยู่ 2 รูปแบบคือ Well-Formed XML และ Valid XML :

- Well-Formed XML การที่เอกสาร XML นั้นจะ well form ได้ก็ต่อเมื่อเอกสาร XML นั้นเขียนได้ถูกต้องตาม syntax ของ XML ได้แก่การปฏิบัติตาม nesting rule การตั้งชื่ออิลิเมนต์ การใส่เครื่องหมายคำพูดครอบค่าของแอตทริบิวต์ ฯลฯ ดังที่ได้กล่าวมาแล้ว
- ส่วน valid XML เอกสาร XML นั้นจะ valid ได้ก็ต่อเมื่อเอกสาร XML นั้นมีการตรวจสอบด้วยวิธีการ DTD (Document Type Definition) หรือ XML Schema

เนื่องจากจุดเริ่มต้นของการมีภาษา XML ขึ้นมานั้นเกิดจากเราต้องการให้ข้อมูลของเราสามารถสื่อความหมายในตัวเองได้ ทำให้เกิดประเด็นที่ต้องพิจารณาว่า ถ้าเรานำ XML มาอธิบายความหมายของข้อมูลแล้ว ตัวเอกสาร XML เองก็ควรมีสิ่งที่จะมาให้ความหมายแก่เอกสาร XML ด้วย ซึ่งในปัจจุบันเราจะใช้ DTD เข้ามาอธิบายเอกสาร XML ของเราว่าเอกสารของเรานั้น ประกอบด้วยอิลิเมนต์อะไรบ้าง อิลิเมนต์ในเอกสารมีลำดับการใช้งานอย่างไร และข้อมูลที่อิลิเมนต์ต่างๆเก็บนั้นคืออะไร

สำหรับ DTD นั้นเป็นวิธีการสร้างนิยามให้แก่เอกสารซึ่งถือกำเนิดมาจากภาษา SGML ซึ่งอย่างที่กล่าวไปแล้วว่า XML เป็นภาษาที่เป็นส่วนย่อยหรือเป็นsubsetของภาษา SGML ดังนั้นเราจึงสามารถนำ DTD มาอธิบายเอกสาร XML ได้ DTD ถือว่าเป็นส่วนเพิ่มเติม (optional) ของเอกสาร XML นั่นคือเราจะกำหนดให้เอกสาร XML ของเรามี DTD กำกับอยู่ด้วยหรือไม่ก็ได้ จะกำหนดให้อยู่ภายในเอกสารของเราเลย หรือว่าจะแยกเป็นอีกไฟล์ก็ได้ ถ้าเป็นกรณีหลังเราจะได้ไฟล์ที่มีนามสกุล .dtd เพิ่มขึ้นอีกไฟล์หนึ่ง

3.2 องค์ประกอบของ DTD

DTD นั้นเป็นสิ่งที่ให้นิยามหรือเป็นการสร้างข้อตกลงให้แก่เอกสาร XML ของเรา เช่นในเอกสารจะต้องประกอบไปด้วยอิลิเมนต์อะไรได้บ้าง เป็นต้น ทั้งนี้การกำกับเอกสาร XML ด้วย DTD จะทำให้เอกสาร XML ของเราสามารถมีการใช้ร่วมกับคนอื่นหรือแอปพลิเคชันอื่นได้โดยที่ผู้ที่มาขอใช้ (หรือที่ขอมให้เราใช้) ก็จะต้องมีการปฏิบัติตามข้อกำหนดที่ระบุไว้ใน DTD นั้นเอง

DTD มีองค์ประกอบหลายอย่างได้แก่ element, attribute, entity, processing instruction และ comment โดยองค์ประกอบบางอย่างก็เป็นสิ่งที่ DTD ต้องมี แต่บางอย่างไม่จำเป็นต้องมีก็ได้ ซึ่งมีรายละเอียดดังต่อไปนี้

3.3 Document Type Declarations

ในการที่เราจะกำหนด syntax และโครงสร้างของอิลิเมนต์ของ DTD นั้น เราจะต้องประกาศ (declare) ข้อกำหนดในเอกสาร โดยใช้ document type declaration โดยเราจะใช้ `<!DOCTYPE>` ในการสร้าง document type declaration ในอิลิเมนต์นี้จะมีรูปแบบการใช้งานอยู่หลายแบบดังนี้

- `<!DOCTYPE rootname [DTD]>`
- `<!DOCTYPE rootname SYSTEM URL>`
- `<!DOCTYPE rootname SYSTEM URL [DTD]>`
- `<!DOCTYPE rootname PUBLIC identifier URL>`
- `<!DOCTYPE rootname PUBLIC identifier URL [DTD]>`

โดยที่ rootname คือ root ของอิลิเมนต์, URL คือ URL ของ DTD

3.3.1 #PCDATA

โดยปกติ XML จะมองที็กซ์ (text) ทุกตัวเป็น parsable character คือต้องมีความถูกต้องโดยพาร์สเซอร์และเรียกที็กซ์ประเภทดังกล่าวว่า PCDATA (Parsable Character Data) และเมื่ออิลิเมนต์ใดถูกกำหนดให้เป็น PCDATA แล้ว ข้อความที่อยู่ระหว่างอิลิเมนต์นั้นก็จะถูกพาร์สหรือถูกตรวจสอบโครงสร้างของข้อมูล เช่นถ้าเรากำหนดในส่วนของ DTD ให้อิลิเมนต์ที่ชื่อ To เป็น PCDATA จะได้ดังนี้

```
<!ELEMENT To (#PCDATA)>
```

3.4 Elements Declarations

สำหรับ DTD เราใช้อิลิเมนต์เพื่อกำหนดรายละเอียดต่างๆของเอกสาร XML ของเรา ภายในอิลิเมนต์หนึ่งๆนั้น จะมีการประกาศชื่อและประเภทของข้อมูลที่สามารเก็บได้ ซึ่งเราจะเรียกตรงส่วนนี้ว่า “Content Specification”

การประกาศ content specification ดังกล่าวสามารถทำได้ใน 4 ลักษณะดังต่อไปนี้

1. อิลิเมนต์ที่ประกอบด้วยรายการของอิลิเมนต์อื่นๆ ตั้งแต่ 1 ตัวขึ้นไป ตัวอย่างเช่น

```
<!ELEMENT Email(To,From,CC,BCC,Subject,Body)>
```

ซึ่งมีความหมายว่าอิลิเมนต์ที่ชื่อ Email ประกอบไปด้วยรายการของอิลิเมนต์ต่างๆได้แก่ To, From, CC, BCC, Subject และ Body ซึ่งเราจะเรียกแต่ละอิลิเมนต์ในรายการดังกล่าวว่าเป็น “Sub Element” หรือ “Child Element” ของอิลิเมนต์ Email และเมื่อรวมกันเราจะเรียกว่า “Content Model” ซึ่งแต่ละอิลิเมนต์ใน Content Model จะต้องมีการประกาศรายละเอียดของตัวเองใน DTD ด้วย จากตัวอย่างข้างต้น เมื่อเราประกาศว่าอิลิเมนต์ Email ประกอบไปด้วยอิลิเมนต์ To, From, CC, BCC, Subject และ Body แล้ว เราจะต้องมีการประกาศต่อไปว่า อิลิเมนต์ To, From, CC, BCC, Subject และ Body มีการเก็บข้อมูลหรือเก็บอิลิเมนต์เป็นแบบไหน

2. อิลิเมนต์ที่ไม่มีการเก็บข้อมูลใดๆ โดยจะใช้คีย์เวิร์ด EMPTY ตัวอย่างเช่น

```
<!ELEMENT X EMPTY>
```

ซึ่งมีความหมายว่า X เป็นอิลิเมนต์ว่าง (empty element) คือเป็นอิลิเมนต์ที่ไม่มีการเก็บข้อมูลใดๆเลยนั่นเอง ถ้าเทียบกับแท็กใน HTML ก็เทียบได้กับแท็ก
 หรือ <HR> ซึ่งเป็นแท็กที่มีไว้เพื่อการใช้งานบางอย่าง แต่ไม่มีข้อมูลใดๆบรรจุอยู่ในแท็กนั้น เช่น เราใช้
 เพื่อขึ้นบรรทัดใหม่ เป็นต้น

3. อิลิเมนต์ที่สามารถเก็บข้อมูลใดๆก็ได้ ตัวอย่างเช่น

```
<!ELEMENT Y ANY>
```

ในกรณีของอิลิเมนต์ Y คือเราอาจไม่แน่ใจว่าข้อมูลที่อิลิเมนต์ Y จะเก็บนั้นเป็นข้อมูลแบบใด หรืออาจจะประกาศในลักษณะดังกล่าวถ้า อิลิเมนต์ Y มีการเก็บข้อมูลหลายๆแบบ และเราไม่ต้องการเจาะลงไปว่า อิลิเมนต์ Y จะเก็บข้อมูลแบบใดบ้าง

4. อิลิเมนต์ที่มีการเก็บข้อมูลแบบผสม กล่าวคือ อิลิเมนต์สามารถเก็บข้อมูลได้อย่างใดอย่างหนึ่งโดยใช้เครื่องหมายไปป์ (pipe, |) เป็นตัวขึ้นข้อมูล ตัวอย่างเช่น

```
<!ELEMENT Z (#PCDATA|X|Y)>
```

ซึ่งหมายความว่า Z เป็นอิลิเมนต์ที่สามารถเก็บข้อมูลได้ทั้งเป็น PCDATA หรือ child element X, Y ก็ได้

นอกจากนี้ XML ยังใช้สัญลักษณ์ต่างๆ เข้ามาเป็นตัวช่วยในการประกาศอิลิเมนต์ โดยสามารถสรุปเป็นตารางได้ดังนี้

สัญลักษณ์	ตัวอย่าง	ความหมาย
(	(X,Y)	อิลิเมนต์จะต้องมีลำดับของเนื้อหา (content) เป็น X และ Y
,	(X,Y,Z)	ตามลำดับ
	(X Y Z)	อิลิเมนต์ต้องมีเนื้อหาเป็น X, Y และ Z
?	X?	อิลิเมนต์ต้องมีเนื้อหาเป็น X หรือ Y หรือ Z อย่างใดอย่างหนึ่ง
*	X*	อิลิเมนต์อาจมีเนื้อหาเป็น X และหากมีเนื้อหาเป็น X ก็จะต้องปรากฏได้เพียงครั้งเดียวเท่านั้น

+	X+	อิลิเมนต์อาจมีเนื้อหาเป็น X และหากมีเนื้อหาเป็น X ก็ สามารถปรากฏได้มากกว่า 1 ครั้ง
ไม่มีสัญลักษณ์	X	อิลิเมนต์จะต้องมีเนื้อหาเป็น X อย่างน้อย 1 ครั้งขึ้นไป อิลิเมนต์จะต้องมีเนื้อหาเป็น X

ตารางที่ 3-1 แสดงสัญลักษณ์ต่างๆใช้ในการประกาศอิลิเมนต์

เราจะมาดูตัวอย่างของการใช้สัญลักษณ์ต่างๆดังนี้

```
<!ELEMENT Email(To+,From,CC*,BCC*,Subject?,Body?)>
```

ซึ่งมีความหมายว่า อิลิเมนต์ที่ชื่อ Email ประกอบด้วยรายการของอิลิเมนต์อื่นๆ ได้แก่ To, From, CC, BCC, Subject และ Body โดยที่

- To จะต้องมียูนิคอย่างน้อย 1 ค่า
- From จะต้องมียูนิคได้เพียง 1 ค่า
- BCC และ CC อาจจะมีหรือไม่มียูนิคก็ได้ ถ้ามีก็สามารถมีได้แค่ 1 ค่า
- Subject อาจจะมีหรือไม่มียูนิคก็ได้ แต่ถ้ามีก็สามารถมีได้แค่ครั้งเดียว
- Body อาจจะมีหรือไม่มียูนิคก็ได้ แต่ถ้ามีก็สามารถมีได้แค่ครั้งเดียว

3.5 DTD Comments

การเขียนคอมเมนต์ในส่วนของ DTD จะมีลักษณะเหมือนกับการเขียนคอมเมนต์ใน XML นั่นคือข้อความที่จะเป็นคอมเมนต์จะต้องอยู่ระหว่างเครื่องหมาย <!-- และ -->

3.6 การกำหนด DTD ไว้ในเอกสาร XML

การประกาศใช้งาน DTD มี 2 แบบ คือ ประกาศไว้ในเอกสาร XML และ ประกาศไว้เป็นไฟล์นามสกุล DTD ซึ่งแยกกันอยู่กับเอกสาร XML จากตัวอย่างที่ผ่านมาเราจะประกาศ DTD ไว้ในเอกสาร XML การที่เราจะกำหนด DTD ไว้ในเอกสาร XML นั้นเราจะใช้ DOCTYPE ดังนี้

```
<!DOCTYPE rootelement SYSTEM URL>
```

3.6.1 Public Document Type Definitions

เมื่อเรามี DTD ที่ไว้ใช้กับการใช้ทั่วไป เราจะใช้คีย์เวิร์ด PUBLIC มาแทน SYSTEM ที่อยู่ใน <!DOCTYPE> document type declaration ซึ่งจะใช้รูปแบบดังนี้

```
<!DOCTYPE rootelement PUBLIC identifier URL>
```

การใช้ศัพท์เวิร์ด PUBLIC เราจะต้องสร้าง formal public identifier (FPI) และกำหนดกฎให้กับ FPI ดังนี้

- ในส่วนแรกเป็นส่วนของการติดต่อของ DTD สำหรับ DTD ที่เราเป็นคนกำหนดเองในส่วนนี้ควรเป็นค่า “-“ ถ้า nonstandard body ถูกต้องตามแบบแผนของ DTD ใช้ “+” และสำหรับรูปแบบที่เป็นมาตรฐาน ในส่วนนี้จะอ้างถึงค่ามาตรฐานของมัน (อย่างเช่น ISO/IEC13449:2000)
- ในส่วนที่สองต้องใส่ชื่อของกลุ่มหรือบุคคลที่คอยดูแล หรือรับผิดชอบ DTD ในที่นี้ควรใช้ชื่อที่ไม่ซ้ำกัน (unique) และอ้างถึงกลุ่มได้ง่าย เช่น W3C ควรใช้ w3c
- ในส่วนที่สามต้องแสดงรูปแบบของเอกสารที่อธิบาย ควรที่จะตามด้วย unique identifier ของ ชนิด เช่น version 1.0 ซึ่งในส่วนนี้ควรประกอบด้วยตัวเลขของ version ที่คุณจะ update
- ในส่วนที่สี่ จะเป็นส่วนที่กำหนดภาษาที่ DTD ใช้ เช่นภาษาอังกฤษจะใช้ EN
- แต่ละส่วนของ FPI ต้องแยกด้วย double slash (//)

```
<!DOCTYPE Email PUBLIC "-//kmitl//Student XML version 1.0//EN"
    "email.dtd">
```

3.7 DTD Entities

เอนทิตีใน XML มีอยู่ 2 ชนิด คือ general entities และ parameter entities โดย general entities จะอยู่ใน content ของเอกสาร XML ซึ่งจะขึ้นต้นด้วยเครื่องหมาย & และลงท้ายด้วยเครื่องหมาย ; ส่วน parameter entities จะใช้ในเอกสาร DTD ซึ่งจะเริ่มต้นด้วยเครื่องหมาย % และลงท้ายด้วยเครื่องหมาย ;

เอนทิตี สามารถเป็นได้ทั้ง parsed และ unparsed เอนทิตี ซึ่งใน content ของ parsed เอนทิตี จะเป็นข้อความที่เป็น well-formed XML ส่วน unparsed เอนทิตีจะเก็บข้อมูลที่ไม่ต้องการ parsed เช่น ข้อมูลทั่วไปหรือ ข้อมูลที่เป็นไบนารี

3.7.1 General Entities

ปกติแล้ว XML จะ define เอนทิตีพื้นฐานไว้แล้ว 5 ตัวคือ <, >, &, " และ ' เอนทิตีเหล่านี้จะใช้แทนตัวอักษร <, >, &, “ และ ‘ ตามลำดับ ซึ่งเราไม่ต้อง defined มันลงไปใน DTD อีก จากตัวอย่างจะเป็นการใช้เอนทิตีที่มีการ define ไว้แล้ว 5 ตัวดังนี้

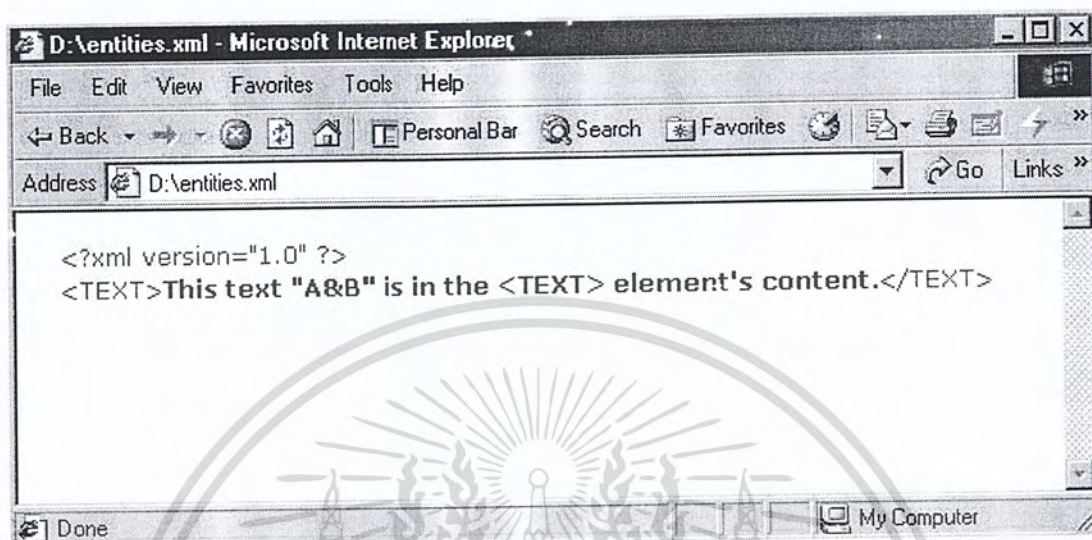
```
<?xml version="1.0" ?>
```

```
<TEXT>
```

This text "A&B"; is in the <TEXT> element's content.

```
</TEXT>
```

เมื่อทำการ parsed แต่ละเอนทิตีจะถูกแทนด้วยตัวอักษรที่ถูกกำหนดค่าไว้ในแต่ละ entities โดย XML processor โดยในรูปที่ 3-1 จะแสดงเอกสารนี้ที่เปิดใน Internet Explorer สังเกตว่าทุก entities จะถูกแทนด้วยค่าของแต่ละ entities ดังนี้



รูปที่ 3-1 แสดงผลลัพธ์การใช้ general เอนทิตีใน Internet Explorer

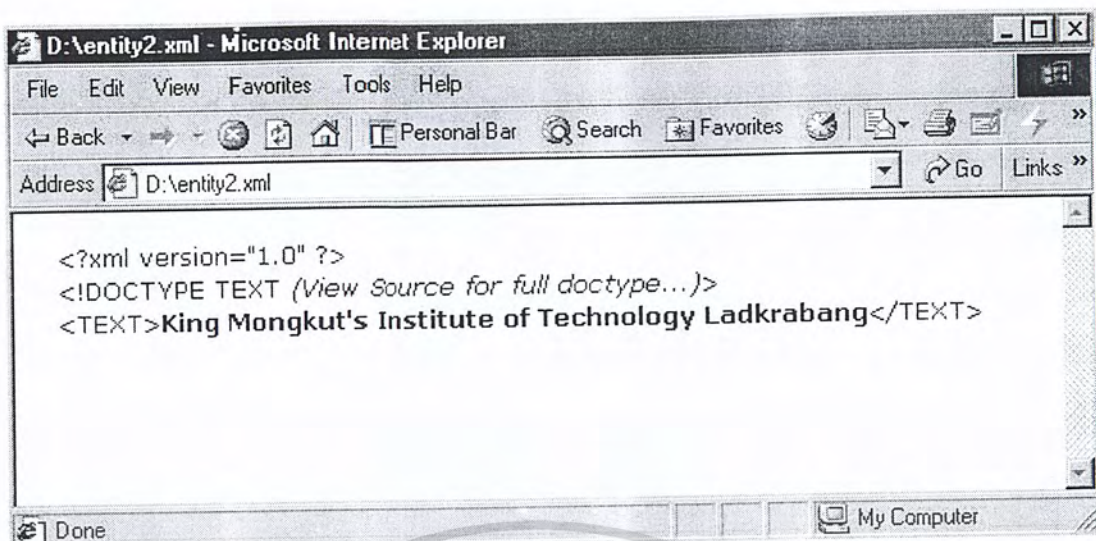
เราสามารถ define เอนทิตีของเราเองได้โดยการ declare ไว้ใน DTD ซึ่งเราจะใช้อิเลเมนต์ `<!ENTITY>` ในการ declare เอนทิตี (คล้ายๆกับการใช้ `<!ELEMENT>`) โดยจะใช้รูปแบบดังนี้

```
<!ENTITY NAME DEFINITION>
```

โดย NAME จะเป็นชื่อของเอนทิตี และ DEFINITION เป็นค่าของเอนทิตี ตัวอย่างต่อไปนี้จะเป็นการแสดงผลการใช้เอนทิตีที่ง่ายที่สุด ในที่นี้เราจะตั้งชื่อของเอนทิตีว่า KMITL และค่าของเอนทิตีเป็นชื่อเต็มของ KMITL ดังนี้

```
<?xml version="1.0"?>
<!DOCTYPE TEXT [
  <!ENTITY KMITL "King Mongkut's Institute of Technology Ladkrabang"
]>
<TEXT>&KMITL;</TEXT>
```

จากตัวอย่างข้างต้นเราจะได้ผลลัพธ์ที่แสดงผ่าน Internet Explorer ดังนี้ สังเกตว่าที่ `&KMITL` ใน source code เมื่อทำการ parsed แล้วจะถูกแทนที่ด้วยค่า (definition) ที่อยู่ในเอนทิตี KMITL



รูปที่ 3-2 แสดงผลลัพธ์การใช้ general เอนทิตีที่สร้างขึ้นเองใน Internet Explorer

3.7.2 Parameter Entities

parameter entity สามารถใช้ได้ในการเอกสาร DTD เท่านั้น การสร้าง parameter entity นั้นคล้ายกับการสร้าง general entity โดยจะมี % อยู่ภายในอิลิเมนต์ `<!ENTITY>` ดังนี้

```
<!ENTITY % NAME DEFINITION>
```

และนี่เป็นตัวอย่างการใช้ internal parameter entity ซึ่งเราจะ declare เอนทิตีชื่อว่า BR และมี definition คือ `<!ELEMENT BR EMPTY>` และเวลาเรียกใช้งานเราจะขึ้นต้นด้วย % ตามด้วยชื่อของเอนทิตี และลงท้ายด้วย ; ดังนี้

```
<?xml version="1.0" ?>
<!DOCTYPE TEXT [

<!ENTITY %BR "<!ELEMENT BR EMPTY">
<!ENTITY KMITL "King Mongkut's Institute of Technology Ladkrabang"

%BR;

]>
<TEXT>&KMITL;</TEXT>
```

3.8 Attributes

ทุกอิลิเมนต์ใน XML สามารถที่จะใส่ attribute ต่างๆได้ และเราสามารถที่จะกำหนดได้ว่าแต่ละอิลิเมนต์ในเอกสาร XML ควรที่จะมี attributes ชื่ออะไร และมีลักษณะเป็นแบบไหน มีค่าดีฟอลต์ (default) เป็นอะไรได้ โดยใช้อิลิเมนต์ `<!ATTLIST>` ในการกำหนดค่าลิสต์ของ attribute ดังนี้

<!ATTRIBUTE ELEMENT_NAME		
	ATTRIBUTE_NAME	TYPE DEFAULT_VALUE
	ATTRIBUTE_NAME	TYPE DEFAULT_VALUE

โดยที่ ELEMENT_NAME คือชื่อของอิลิเมนต์ที่เราจะ declare attributes, ATTRIBUTE_NAME คือชื่อของ attribute ที่จะ declare, TYPE คือ ชนิดของ attribute และ DEFAULT_VALUE เป็นการกำหนดค่าดีฟอลต์ให้กับ attribute ซึ่งตารางข้างล่างนี้จะเป็นตารางที่สรุปค่าของ TYPE ทั้งหมดที่มี

TYPE	Description
CDATA	เป็นตัวอักษรธรรมดาที่ไม่ประกอบด้วย ตัวอักษรที่เป็น markup
ENTITIES	ชื่อเอนทิตีหลายตัว(ซึ่งต้อง declare ใน DTD)แยกกันด้วยช่องว่าง(whitespace)
ENTITY	ชื่อเอนทิตี (ซึ่งต้อง declare ใน DTD)
Enumerated	แทน list of values (ต้องใช้ค่าใดค่าหนึ่งจาก list)
ID	เป็นคุณสมบัติของ XML ที่ชื่อจะต้องไม่ซ้ำกัน นั่นคือแต่ละ attribute ที่มีค่า ID จะมีชื่อที่ไม่ซ้ำกัน
IDREF	จะถือค่าของ ID attribute ของบางอิลิเมนต์ โดยปกติอิลิเมนต์ที่เป็นอิลิเมนต์ปัจจุบัน(current element) จะเกี่ยวข้อง
IDREFS	แสดง ID ของอิลิเมนต์หลายค่าซึ่งแยกโดยช่องว่าง(whitespace)
NMTOKEN	แสดงชื่อ XML ที่เกี่ยวข้อง
NMTOKENS	แสดงชื่อของ XML ที่เกี่ยวข้องกันหลายค่าในลิสต์ ซึ่งแยกกันโดยช่องว่าง(whitespace)
NOTATION	แสดงชื่อของ notation ซึ่งต้อง declare ใน DTD

ตารางที่ 3-2 สรุปค่าต่างๆของ TYPE ใน Attributes

ในส่วนของตารางด้านล่างนี้จะเป็นตารางที่สรุปค่า DEFAULT_VALUE ที่เป็นไปได้ทั้งหมดดังนี้

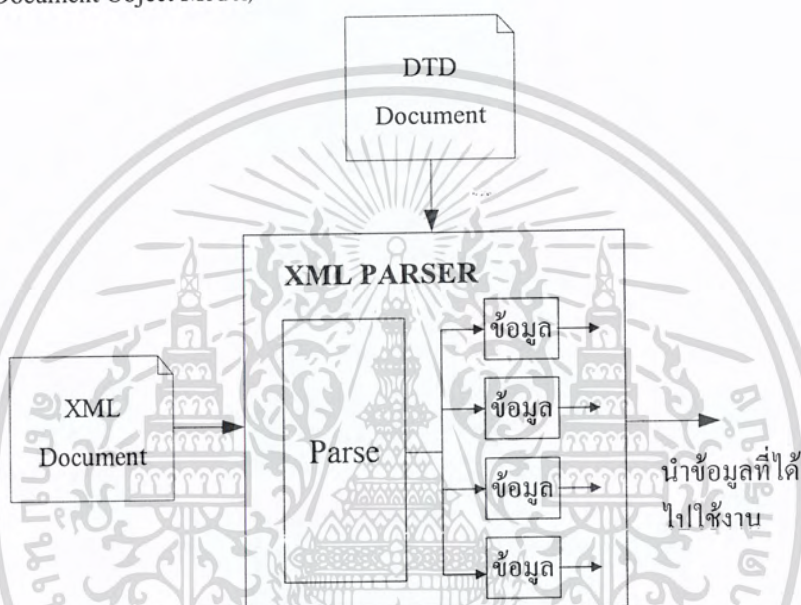
Method	Description
VALUE	แสดงข้อมูลพื้นฐานซึ่งถูกปิดด้วยเครื่องหมายคำพูด
#IMPLIED	แสดงว่า attribute นี้ไม่มีค่าดีฟอลต์ และ จะมีหรือไม่มีค่านี้ก็ได้
#REQUIRED	แสดงว่า attribute นี้ไม่มีค่าดีฟอลต์ แต่จะต้องใส่ค่าให้ attribute นี้เสมอ
#FIXED VALUE	ในที่นี้ VALUE ถือค่าของ attribute และ attribute นี้จะต้องใช้ค่านี้เสมอ

ตารางที่ 3-3 สรุปค่าต่างๆของ DEFAULT_VALUE ที่อยู่ใน Attributes

บทที่ 4

XML Parser

ก่อนที่จะนำเอกสาร XML ไปใช้งานได้นั้นจำเป็นจะต้องผ่านกระบวนการที่เรียกว่าการ “parsing” เสียก่อน โดยกระบวนการ parsing หรือการ parse เอกสารนั้นก็คือ การที่ทำให้คอมพิวเตอร์เข้าใจถึงโครงสร้างและองค์ประกอบต่างๆของเอกสารนั่นเอง และส่วนที่รับผิดชอบในการ parse เอกสารเรา จะเรียกว่า Parser ในปัจจุบันมีมาตรฐานหรือวิธีการหลักๆ อยู่ 2 อย่างได้แก่ SAX (Simple API for XML) และ DOM (Document Object Model)



รูปที่ 4-1 แสดงการทำงานของ parser

4.2 SAX (Simple API for XML)

SAX เป็นแอปพลิเคชันโปรแกรมมิ่งอินเทอร์เฟซ (API) ที่มีลักษณะการทำงานก็ต่อเมื่อมีเหตุการณ์ (เช่น การร้องขอข้อมูล) ใดๆ เกิดขึ้น จะเรียกโมเดลการทำงานแบบนี้ว่า event-based model กล่าวคือ SAX จะไม่มีการสร้างภาพรวมของเอกสารไว้ก่อนเลยว่าเอกสารมีโครงสร้างเป็นเช่นไร เมื่อแอปพลิเคชันมีการร้องขอมาจึงจะรายงานหรือทำงานตามเหตุการณ์ของการทำ parsing ไปยังแอปพลิเคชันดังกล่าว

หลักการทำงานของ SAX จะมองเพียง “จุดเริ่มต้น” กับ “จุดสิ้นสุด” ของเอกสารและของอิลิเมนต์เท่านั้น และจะสนใจเฉพาะสิ่งที่ต้องการเท่านั้น

```
<?xml version="1.0"?>
```

```
<DOC>
```

```
<ELEMENT1>value 1</ ELEMENT 1>
```

```
< ELEMENT 2>value 2</ ELEMENT 2>
```

```
</DOC>
```

SAX จะมองเอกสารข้างต้นไปที่ละบรรทัดๆ ในลักษณะของ linear events ดังนี้

start document

start element: DOC

start element: ELEMENT 1

characters: value 1

end element: ELEMENT 1

start element: ELEMENT 2

characters: value 2

end element: ELEMENT 2

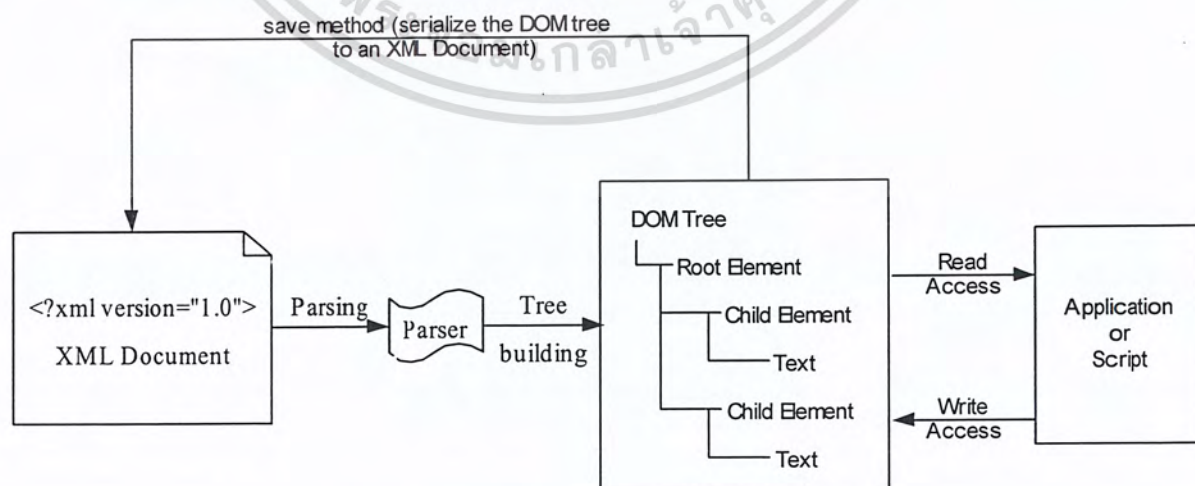
end element: DOC

end document

จากตัวอย่าง SAX จะทำการพาร์สเอกสารตั้งแต่บรรทัดแรก จนถึงบรรทัดสุดท้ายแบบเส้นตรง

4.3 Document Object Model

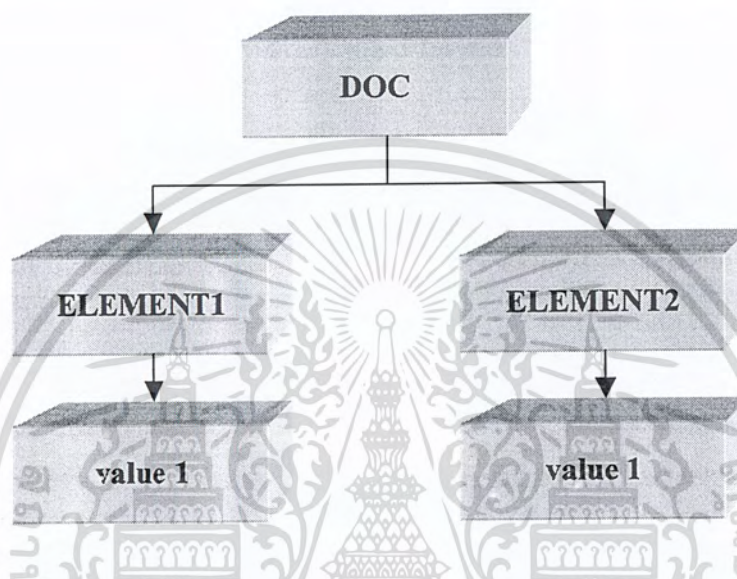
DOM คือแอปพลิเคชันโปรแกรมมิ่งอินเทอร์เฟซ (Application Programming Interface, API) สำหรับเอกสาร XML ซึ่งจะนำเสนอข้อมูลของ XML ในลักษณะของทรี (tree) เพราะว่าการ traversal และการจัดการ ของโครงสร้างแบบทรีง่ายต่อการเขียนโปรแกรม DOM จะอ่านเอกสาร XML ลงไปในหน่วยความจำ (memory) และเก็บข้อมูลทั้งหมดอยู่ในโหนด (node) ดังนั้นการเข้าถึงข้อมูลจะเร็วมาก ระดับบนสุดของทรีเราจะเรียกว่า “document Element” โอลิเมนต์ดังกล่าวจะแตกสาขาออกไปเป็นโอลิเมนต์ย่อยๆ ตั้งแต่ 1 โอลิเมนต์ขึ้นไป โดยโอลิเมนต์ย่อยที่แตกออกไปเราจะเรียกว่า “child node” ซึ่งรูปแบบการทำงานของ parser ที่เป็น DOM สามารถแสดงได้ดังรูปที่ 4-2 ซึ่งในโครงงานนี้ได้เลือกใช้ DOM เป็นตัวพาร์สเซอร์ เพราะสามารถจัดการ (เช่น การแก้ไข หรือการลบข้อมูล) กับข้อมูล XML ที่เก็บอยู่ในโหนดได้ง่ายกว่าการทำด้วย SAX พาร์สเซอร์



รูปที่ 4-2 แสดงการทำงานของ parser ที่เป็น DOM

จากตัวอย่างด้านล่างนี้ เมื่อผ่านกระบวนการ parsing โดย DOM จะทำการมองเอกสารออกมาเป็นทรีดังนี้

```
?xml version="1.0"?>
<DOC>
    <ELEMENT1>value 1</ ELEMENT 1>
    < ELEMENT 2>value 2</ ELEMENT 2>
</DOC>
```

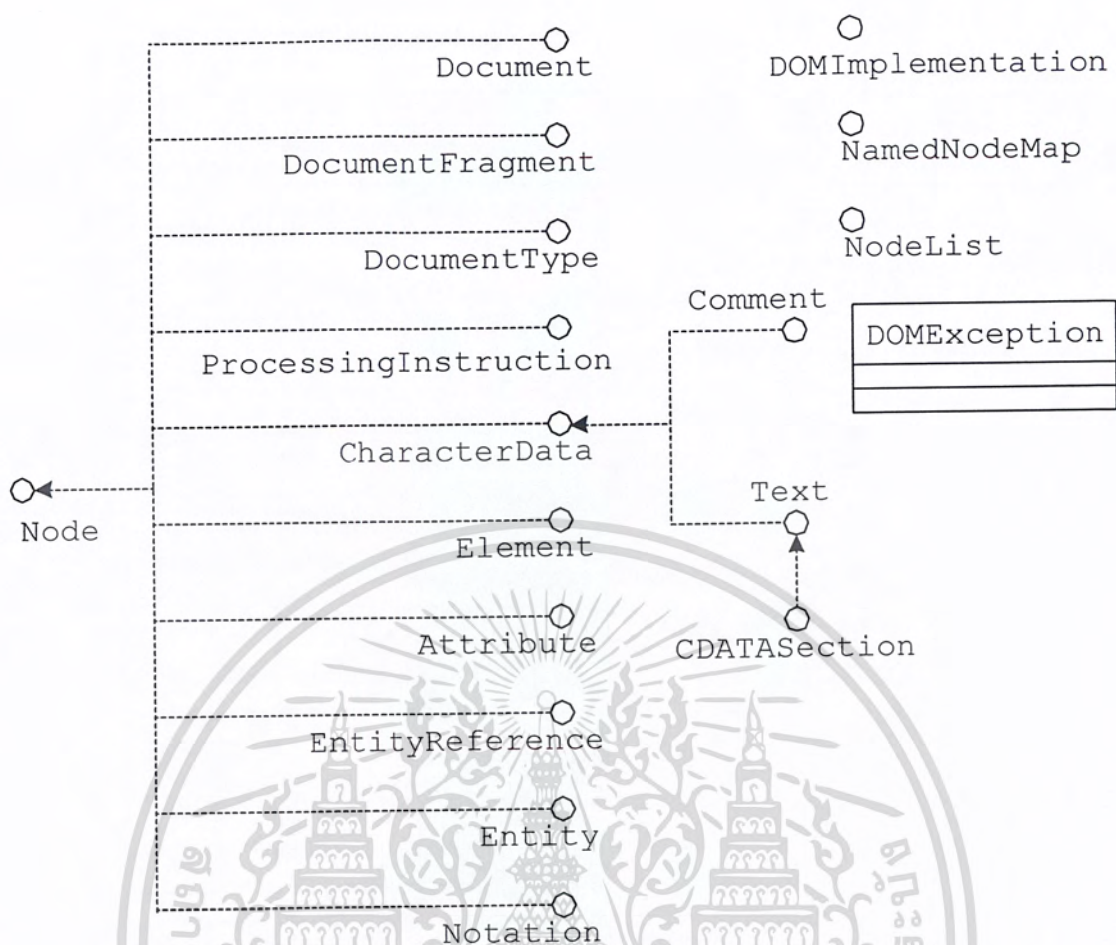


รูปที่ 4-3 แสดงลักษณะการมองข้อมูลเป็นทรีใน DOM

การจะเข้าถึงข้อมูลในแต่ละโหนดของทรี ถ้าเราทำงานภายใต้การทำงานของบราวเซอร์ อย่างเช่น Internet Explorer ตั้งแต่ version 5 เป็นต้นไปจะมีการฝัง XML Parser ไว้อยู่แล้ว แต่ถ้าเราต้องการจะเข้าถึงข้อมูลโดยใช้ภาษา Java เราจะต้องมีคลาสของ DOM Parser ก่อน ในที่นี้เราจะใช้ Xerces ของ Apache โดยในคลาสของ Xerces นี้จะได้จัดเตรียมฟังก์ชันต่างๆ ที่จำเป็นต้องใช้ในการพาร์สและจัดการกับข้อมูล

4.3.1 การใช้งาน DOM Tree

เมื่อได้ DOM Tree จากกระบวนการ parsing แล้ว เราสามารถที่จะย้าย หรือปรับเปลี่ยนโครงสร้างของทรี หรือเข้าถึงข้อมูล และนำข้อมูลจากทรีของข้อมูล XML มาแสดงผล การที่เราจะทำอะไรกับข้อมูลในทรีนั้น เราจะต้องเข้าใจ object พื้นฐานของข้อมูลใน XML ที่จะสามารถเข้าถึงได้ โดยอินเทอร์เฟซเหล่านี้ได้แสดงในรูปที่ 4-3 ซึ่งเราจะจัดการกับข้อมูลทั้งหมดภายใต้ DOM tree นี้



รูปที่ 4-3 UML class model of DOM Level 2 core interfaces and classes

จากรูปเราจะให้ความสนใจกับ Node interface เป็นพิเศษ จะสังเกตว่ามันเป็น base interface สำหรับ interfaces อื่นๆ จึงสามารถที่จะเขียน method กระทำกับโหนดครอบคลุมทุกประเภทของโหนดในโครงสร้างแบบ DOM

บทที่ 5

โพรโทคอล HTTP (HyperText Transfer Protocol)

HTTP เป็นโพรโทคอลที่ใช้ในการติดต่อสื่อสารระหว่างเว็บเบราว์เซอร์และเว็บเซิร์ฟเวอร์ สำหรับเครือข่ายเวิลด์ไวด์เว็บ โดยใช้คอนเซ็ปต์ของไคลเอนต์-เซิร์ฟเวอร์เป็นหลัก หลักการทำงานของ HTTP คือการส่งข้อความจากไคลเอนต์ไปยังเซิร์ฟเวอร์เพื่อตีความหมาย หลังจากนั้นเซิร์ฟเวอร์จะทำการส่งข้อความซึ่งเป็นผลการตีความกลับมายังไคลเอนต์ โดยทั่วไปการส่งข้อความระหว่างไคลเอนต์และเซิร์ฟเวอร์มักใช้ไบต์สตรีมเป็นหลัก แต่ HTTP เป็นโพรโทคอลที่ออกแบบมาเพื่อให้ใช้งานง่ายแต่มีประสิทธิภาพ ดังนั้นจึงมีการใช้การส่งข้อมูลแบบเท็กซ์สตรีมแทน

5.1 ความหมายของ HTTP

HTTP มาจากคำว่า Hypertext Transfer Protocol ซึ่งเป็นโพรโทคอลที่ใช้ในการส่งข้อมูลต่างๆ ในโลกเวิลด์ไวด์เว็บ ข้อมูลต่างๆ เหล่านี้โดยทั่วไปมักจะถูกเรียกว่า *Resource* โดย Resource เหล่านี้อาจเป็นไฟล์ เช่น ไฟล์ HTML, ไฟล์รูปภาพ หรือคำสั่งต่างๆ (Query String)

HTTP เป็นโพรโทคอลที่อยู่ในส่วนของเลเยอร์แอปพลิเคชัน ในโพรโทคอลสแต็กโดยข้อมูลต่างๆ จากเลเยอร์นี้จะถูกส่งผ่านไปยังเลเยอร์อื่นๆ ที่ต่ำกว่าซึ่งส่วนหนึ่งในนั้นก็คือโพรโทคอล TCP/IP นั่นเอง

HTTP เป็นเน็ตเวิร์คโพรโทคอลที่ใช้หลักการของโมเดลไคลเอนต์เซิร์ฟเวอร์ ในการติดต่อสื่อสาร ซึ่งมีหลักการทำงานอย่างคร่าวๆ ดังต่อไปนี้

- HTTP ไคลเอนต์ จะทำการสร้างคอนเนกชันไปยัง HTTP เซิร์ฟเวอร์ซึ่งโดยทั่วไปจะผ่านทางซ็อกเก็ตของ TCP/IP
- หลังจากนั้น HTTP ไคลเอนต์ จะทำการส่งคำสั่งการร้องขอ ซึ่งอยู่ในรูปของข้อความไปให้ HTTP เซิร์ฟเวอร์เพื่อขอ Resource ที่ต้องการ
- HTTP เซิร์ฟเวอร์จะทำการตีความคำสั่งที่ได้รับและส่งผลตอบสนองกลับ ซึ่งเป็น Resource ที่ HTTP ไคลเอนต์ ต้องการกลับมา
- หลังจากที่มีการส่งการตอบรับเสร็จสิ้น HTTP เซิร์ฟเวอร์จะทำการปิดคอนเนกชันที่มาจาก HTTP ไคลเอนต์
- ในกรณีที่ HTTP ไคลเอนต์ต้องการ Resource อื่นๆ HTTP ไคลเอนต์ จะต้องทำการสร้างคอนเนกชันใหม่และส่งคำสั่งไปหา HTTP เซิร์ฟเวอร์อีกครั้ง

จากหลักการข้างต้นจะเห็นว่าการติดต่อสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์จะเป็นลักษณะครั้งต่อครั้ง ในทางเครือข่ายเรียกว่าการสื่อสารแบบนี้ว่า Stateless Protocol

4. เมสเชจบอดี ซึ่งอาจจะใช้บรรจุไฟล์, คำสั่ง (Query String) หรืออาจจะเป็นเอาต์พุตที่มาจากเซิร์ฟเวอร์โดยส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับจุดประสงค์ของการใช้งาน

CRLF = Carriage return (\r) และ Line Feed (\n) ซึ่งก็คือ \u00A และ \u00D ใน ACSII นั่นเอง
รูปแบบโดยทั่วไปของเมสเชจจะเป็น

```
<initial line>
เซคเตอร์1: value1
เซคเตอร์2: value2
เซคเตอร์3: value3
[blank line]
<เมสเชจ body, optional.....>
.....>
```

ตัวอย่างง่าย ๆ ของเมสเชจที่ไคลเอนต์อาจจะเขียน

```
GET /images/cat.jpg ← บรรทัดเริ่มต้น
From: soup@jarticles.com ← เซคเตอร์
User-Agent: Mozilla/4.72 ← เซคเตอร์
[blank line]
หรือ
POST /servlet/searchEngine HTTP/1.0 ← บรรทัดเริ่มต้น
From: webmaster@jarticles.com ← เซคเตอร์
User-Agent: Mozilla/4.72 ← เซคเตอร์
[blank line]
keyword=java&topic=jsp ← เมสเชจบอดี
```

ตัวอย่างง่าย ๆ ของเมสเชจที่เซิร์ฟเวอร์อาจจะเขียน

```
HTTP 1.0 200 OK ← บรรทัดเริ่มต้น
Date: Sunday, 23-July-00 04:01:12 GMT ← เซคเตอร์
Server: Apache/1.3.12(Unix) (Red Hat/Linux) PHP/3.0.15 mod_perl/1.21 ← เซคเตอร์
MIME-version: 1.0 ← เซคเตอร์
Content-type: text/html ← เซคเตอร์
Content-length: 115 ← เซคเตอร์
[blank line]
<HTML><HEAD><TITLE>HTTP Tutorial</TITLE></HEAD> ← เมสเชจบอดี
```

<BODY>This is a tutorial, but please visit me again...</BODY> ← เมสเซจบอดี(cont.)
 </HTML> ← เมสเซจบอดี(cont.)

5.2.1 บรรทัดเริ่มต้นในการร้องขอ (Initial Request Line)

บรรทัดเริ่มต้นของการร้องขอ(จากไคลเอนต์)จะแตกต่างจากบรรทัดเริ่มต้นของการตอบรับเล็กน้อย โดยบรรทัดเริ่มต้นของการร้องขอจะมี 3 ส่วนคือ

- ชื่อของเมธอด เช่น GET, POST, HEAD, TRACE
- โคลอกลพารของ resource ที่ไคลเอนต์ต้องการ
- เวอร์ชันของ HTTP/x.x ที่ HTTP ไคลเอนต์ใช้

สามส่วนนี้จะประกอบกันเป็นบรรทัดเริ่มต้น โดยแต่ละส่วนจะถูกแยกออกจากกันโดยใช้ช่องว่าง (space) ดังตัวอย่างข้างล่าง

GET /path/to/file/index.html Http/1.0

ตัวอย่างนี้ใช้เมธอดที่ชื่อ GET เพื่อขอ resource (ในที่นี้ก็คือไฟล์) ชื่อ /path/to/file/index.html โดยใช้โปรโตคอล HTTP/1.0 ในการสื่อสาร

การใช้โคลอกลพาร (Local path)

ก่อนที่ HTTP ไคลเอนต์จะส่งการร้องขอไปที่ HTTP เซิร์ฟเวอร์ HTTP ไคลเอนต์จะสร้างการเชื่อมต่อขึ้นมาก่อนหนึ่งก่อน โดยใช้ชื่อที่เรียกว่าโฮสต์ ในการสร้างชื่อที่ถูกต้องจะต้องทำการกำหนดชื่อของโฮสต์ที่จะติดต่อ ยกตัวอย่างเช่น ถ้า URL เป็น <http://161.246.5.117/path/to/file/index.html> ชื่อของโฮสต์ก็จะเป็น 161.246.5.117 ดังนั้นการกำหนดเพียงโคลอกลพารเพื่อบ่งบอกถึง resource ที่ต้องการในส่วนของการร้องขอเริ่มต้นในการร้องขอก็คือ /path/to/file/index.html ก็เพียงพอแล้ว

API ที่ใช้ในแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ มักอ้างถึงส่วนที่เป็นโคลอกลพารนี้ว่า “Request URI (ตัว I มาจาก Identifier)”

ข้อระวัง

- ชื่อของเมธอดจะต้องใช้ตัวใหญ่เสมอ
- เวอร์ชันของ HTTP จะอยู่ในรูป HTTP/x.x และจะต้องเป็นตัวใหญ่เสมอ

5.2.2 บรรทัดเริ่มต้นในการตอบสนอง (Initial Response Line)

โดยทั่วไปบรรทัดเริ่มต้นในการตอบสนอง(จากเซิร์ฟเวอร์)มักจะเรียกว่า Status line ซึ่งจะประกอบไปด้วย 3 ส่วนย่อยคือ

- 1 เวอร์ชันของ HTTP/x.x ที่เซิร์ฟเวอร์ใช้สำหรับส่งเมสเซจ
- 2 Response status code ซึ่งเป็นตัวบอกว่าผลของการร้องขอที่ไคลเอนต์ส่งมาเป็นอย่างไร
- 3 Reason phase เป็นตัวอธิบายความหมายของ Response status code อีกทีหนึ่ง ยกตัวอย่างเช่น

HTTP/1.0 200 OK

หรือ

HTTP/1.0 404 Not Found

ข้อสังเกต

- เวอร์ชันของ HTTP จะต้องอยู่ในรูป HTTP/x.x และจะต้องเป็นตัวใหญ่เสมอ
- โค้ดสถานะการตอบสนอง (response status code) เป็นโค้ดที่ส่งมาให้คอมพิวเตอร์ทางฝั่งไคลเอนต์อ่านซึ่งจะมี reason phase เป็นตัวอธิบายให้ผู้ใช้ (Human Being) อ่านอีกทีหนึ่ง
- โค้ดสถานะการตอบสนองจะอยู่ในลักษณะของเลข 3 หลัก โดยหลักแรกจะบอกถึงความหมายโดยทั่วไปของ response
 - 1xx เป็น code ที่ใช้บอกถึงเหตุการณ์ต่างๆที่เกิดขึ้นในการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์
 - 2xx เป็น code ที่บ่งบอกว่า request ที่ส่งจากไคลเอนต์ถูกทำให้สมบูรณ์แล้ว
 - 3xx เป็น code ที่บอกให้ไคลเอนต์ทำการ redirect ไปที่ url ที่อื่นแทน
 - 4xx เป็น code ที่บ่งบอกถึงความผิดพลาดที่เกิดขึ้นในส่วนของไคลเอนต์
 - 5xx เป็น code ที่บ่งบอกถึงความผิดพลาดที่เกิดขึ้นในส่วนของเซิร์ฟเวอร์

โค้ดแสดงสถานะ (Status code) ที่พบเห็นโดยทั่วไป คือ

- 100 Continue หมายถึง เซิร์ฟเวอร์ได้รับ request บางส่วนจากไคลเอนต์แล้ว บอกให้ไคลเอนต์ส่งที่เหลือมาด้วย
- 200 OK หมายถึง คำร้องขอที่มาจากไคลเอนต์ถูกต้องและ response ที่ไคลเอนต์ต้องการอยู่ในเมสเสจบอดีของ response นี้แล้ว
- 301 Moved Permanently หมายถึง resource ที่ไคลเอนต์ต้องการได้ถูกย้ายไปที่อื่นแล้ว
- 302 Moved Temporarily หมายถึง resource ที่ไคลเอนต์ต้องการได้ถูกย้ายไปอยู่ที่อื่นชั่วคราว
- 400 Bad Request หมายถึง เซิร์ฟเวอร์ไม่สามารถเข้าใจการร้องขอที่ไคลเอนต์ส่งมา
- 403 Forbidden หมายถึง เซิร์ฟเวอร์เข้าใจการร้องขอที่ไคลเอนต์ส่งมาแต่ไม่สามารถที่จะส่ง resource ที่ไคลเอนต์ต้องการกลับไปได้
- 404 Not Found หมายถึง เซิร์ฟเวอร์ไม่มี resource ที่ร้องขอมา
- 500 Server Error หมายถึงข้อผิดพลาดที่เกิดขึ้นในส่วนของเซิร์ฟเวอร์ (โดยทั่วไปข้อผิดพลาดนี้จะเกิดขึ้นในกรณีที่มีการร้องขอส่งมาเรียกแอปพลิเคชันฝั่งเซิร์ฟเวอร์ที่รันบนเซิร์ฟเวอร์แต่เกิดข้อผิดพลาดขึ้นระหว่างการประมวลผล ซึ่งโดยทั่วไปจะเกิดขึ้นเนื่องจากโปรแกรมมีบั๊ก(bug))

5.2.3 เฮดเดอร์ (Header)

เฮดเดอร์จะเป็นส่วนที่บอกถึงรายละเอียดของการร้องขอ (ที่กำลังส่งไปให้เซิร์ฟเวอร์) หรือการตอบรับ (ที่กำลังส่งกลับมายังไคลเอนต์) ยกตัวอย่างเช่น

- ไคลเอนต์จะส่งเฮดเดอร์ที่บอกถึงชนิดและขนาดของข้อมูลที่อยู่ข้างในเมสเซจบอดี ในกรณีที่ไคลเอนต์ต้องการ upload ไฟล์ไปยังเซิร์ฟเวอร์
- เซิร์ฟเวอร์อาจจะส่งเฮดเดอร์ที่เกี่ยวกับชนิดและขนาดของ resource ที่ไคลเอนต์กำลังจะได้รับ โดยใช้ Content-Type และ Content-Length
- เฮดเดอร์จะมีรูปแบบเป็นเท็กซ์ โดยในหนึ่งบรรทัดจะถูกใช้สำหรับหนึ่งเฮดเดอร์ซึ่งจะอยู่ในรูปของ “เฮดเดอร์-Name: value” และจบด้วย CRLF ยกตัวอย่างเช่น

From: admin@ce.kmitl.ac.th

User-Agent: Mozilla/4.72

Content-Type: text/html

Content-Length: 250

ข้อสังเกต

- ชื่อของเฮดเดอร์จะใช้ตัวใหญ่หรือตัวเล็กก็ได้
- สามารถเว้นช่องว่างหรือ tab ระหว่าง “:” ของชื่อเฮดเดอร์และค่าของมันกว้างเท่าไรก็ได้
- ใน HTTP1.0 จะมีเฮดเดอร์กำหนดไว้ 16 แบบ แต่ของ HTTP1.1 จะมีถึง 46 แบบ

เฮดเดอร์ที่มักจะพบเห็นทั่วไปในส่วนของไคลเอนต์

From: เป็นเฮดเดอร์ที่ใช้เก็บอีเมลแอดเดรสของผู้ที่กำลังส่งการร้องขอ ยกตัวอย่างเช่น

From: admin@ce.kmitl.ac.th

User-Agent: เป็นเฮดเดอร์ที่ใช้บ่งบอกถึงชื่อของโปรแกรมที่ใช้เป็น HTTP ไคลเอนต์ ยกตัวอย่าง

เช่น User-Agent: Mozilla/4.72

เฮดเดอร์ที่มักพบเห็นทั่วไปในส่วนของเซิร์ฟเวอร์

Server: เป็นเฮดเดอร์ที่จะคล้ายกับ User-Agent แต่เป็นตัวบ่งชี้ชื่อ โปรแกรมที่ใช้เป็น HTTP เซิร์ฟเวอร์ ยกตัวอย่างเช่น Server: Apache/1.93

Last-Modified: เป็นเฮดเดอร์ที่ใช้บ่งบอกถึงเวลาครั้งสุดท้ายที่ resource ที่ไคลเอนต์ต้องการถูกเปลี่ยนแปลง (modify/update) โดยเวลาที่ใช้จะเทียบจาก GMT (Green wich Mean Time) ยกตัวอย่างเช่น เวลาของเมืองไทยเป็น GMT+7.00 ถ้าตอนนี้เวลาเมืองไทยคือ Sun, 23 July 2000 20:39:56 เวลาที่เป็น GMT ก็คือ Last-Modified: Sun, 23 July 2000 13:39:56 GMT

5.2.4 เมสเซจบอดี (message body)

เมื่อไรก็ตามที่ไคลเอนต์หรือเซิร์ฟเวอร์ต้องการส่งข้อมูลไปกับเมสเซจ ส่วนที่เป็นเมสเซจบอดีจะเป็นส่วนที่ใช้เก็บข้อมูลดังกล่าว ในกรณีของไคลเอนต์ตัวเมสเซจบอดีอาจใช้สำหรับบรรจุไฟล์ที่ต้องการอัปโหลดหรือใช้เก็บข้อมูลที่มีจากอิลิเมนต์ต่างๆ ของฟอร์ม HTML แล้วส่งไปยังเซิร์ฟเวอร์ก็ได้ ในกรณีของเซิร์ฟเวอร์ตัวเมสเซจบอดีจะเป็นส่วนที่ใช้เก็บ resource ที่ไคลเอนต์ร้องขอหรืออาจใช้เก็บคำอธิบายต่างๆ ในกรณีที่มีข้อผิดพลาดเกิดขึ้นก็ได้ ถ้าเมสเซจมีส่วนของเมสเซจบอดี ไคลเอนต์หรือเซิร์ฟเวอร์มักจะเพิ่มเฮดเดอร์ที่ช่วยบ่งบอกถึงรายละเอียดของเมสเซจบอดีดังกล่าวด้วย ยกตัวอย่างเช่นถ้าเซิร์ฟเวอร์ต้องการส่ง resource ที่เป็นไฟล์รูปภาพมาให้ไคลเอนต์ เฮดเดอร์ที่ถูกเพิ่มมาก็จะเป็น Context-Type: image/gif เป็นเฮดเดอร์ที่บอกถึง MINE type ของข้อมูลที่อยู่ในเมสเซจบอดี Context-Length: 1026 เป็นเฮดเดอร์ที่บ่งบอกถึงขนาดของเมสเซจบอดีในหน่วยไบต์

ตัวอย่างของการส่งเมสเซจระหว่างไคลเอนต์ (Web Browser) และเซิร์ฟเวอร์ (Web Server) เช่น

<http://161.246.5.117/tutorials/basic/helloworld.html>

ขั้นแรกไคลเอนต์จะทำการเรียกใช้ช็อกเก็ตเพื่อเชื่อมต่อไปยังเซิร์ฟเวอร์ชื่อ 161.246.5.117 ซึ่งโดยปกติจะติดต่อไปที่พอร์ต 80 ซึ่งเป็นพอร์ต Default ของโพรโทคอล HTTP (ในกรณีที่ต้องการเชื่อมต่อไปยังพอร์ตอื่นก็สามารถทำได้โดยการเพิ่มส่วนของเลขพอร์ตที่ไคลเอนต์ต้องการเชื่อมต่อเข้าไปหลังจากส่วนของโฮสต์เนม เช่น <http://161.246.5.117:8080/tutorials/advance/helloworld.html> ทำให้ไคลเอนต์เชื่อมต่อไปยังพอร์ต 8080 แทน)

หลังจากนั้นไคลเอนต์จะทำการส่งเมสเซจร้องขอ

GET /tutorials/basic/helloworld.html HTTP/1.0

From: soup@jarticles.com

User-Agent: Mozilla/4.72

[blank line]

หลังจากที่เซิร์ฟเวอร์ได้รับเมสเซจข้างบนแล้ว เซิร์ฟเวอร์ส่งการตอบรับพร้อมทั้ง resource ที่ไคลเอนต์ต้องการกลับมาดังนี้

HTTP/1.0 200 OK

Date: Sunday, 23-July-00 04:01:12 GMT

Content-Type: text/html

Content-Length: 1556

[blank line]

<HTML><HEAD><TITLE>HTTP Tutorial</TITAL></HEAD>

<BODY>This is HelloWorld tutorial, but please read it for fun

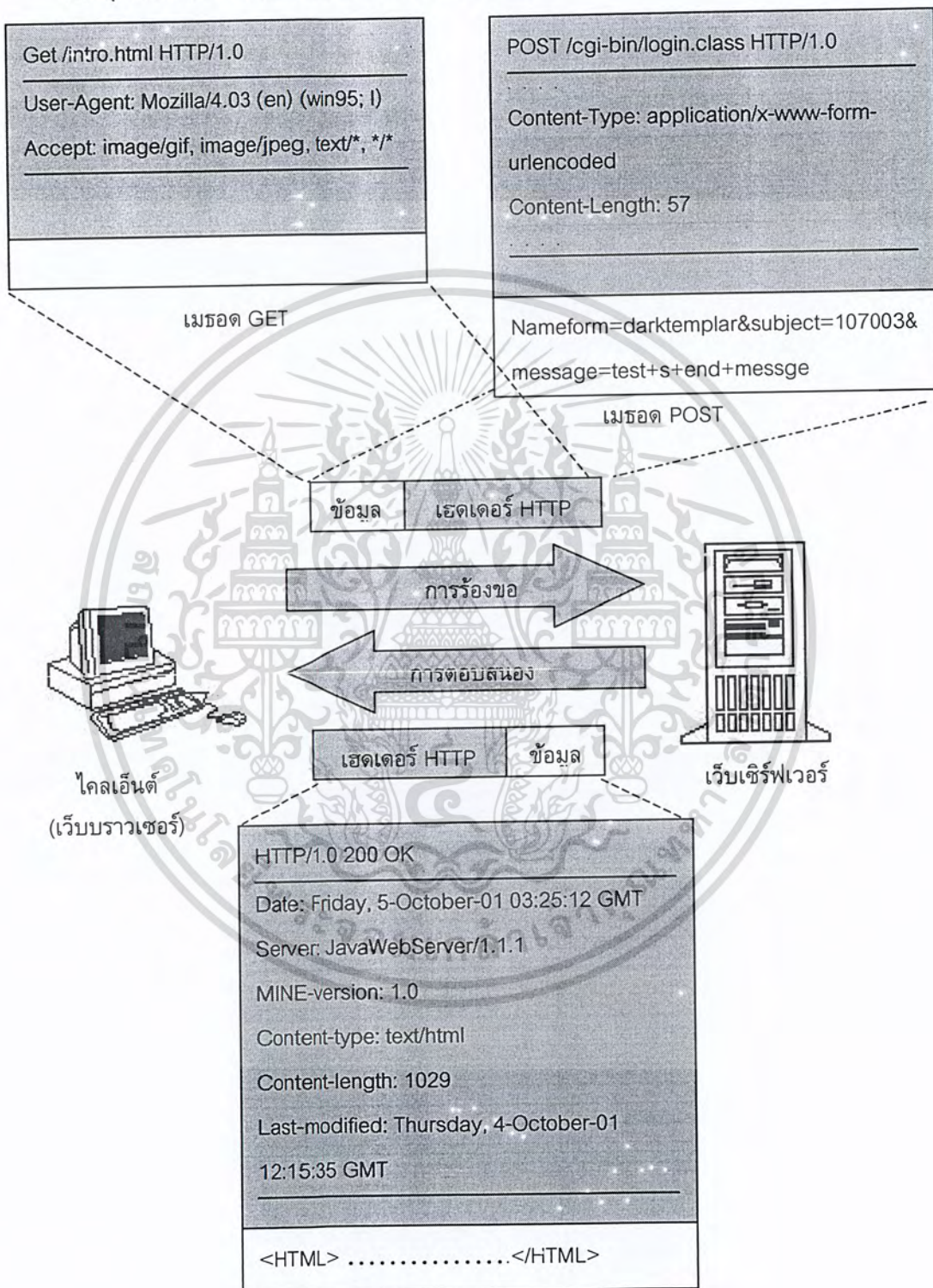
(more file contents)

...

</BODY>

</HTML>

ท้ายสุดเซิร์ฟเวอร์จะทำการปิดการเชื่อมต่อของไคลเอนต์



รูปที่ 5-2 ลักษณะข้อมูลที่รับส่งระหว่างไคลเอนต์และเซิร์ฟเวอร์

5.3 เมธอด (Method)

5.3.1 เมธอด GET

ตัวอย่างเช่น <http://www.google.com/search.cgi?keyword=computer> เป็นการค้นหาที่ www.google.com โดยใช้คีย์เวิร์ด = “computer”

ถ้าพิจารณาส่วนที่อยู่ต่อจากส่วนของโฮสต์เนม (www.google.com) แล้วจะเห็นว่า URL ที่ถูกร้องขอ (/search?keyword=computer) ที่เห็นไม่ได้เป็นไฟล์ html ไฟล์อย่างทั่วๆ ไปแต่กลับกลายเป็นชื่อของโปรแกรม (search.cgi), เครื่องหมายคำถาม (?) และคีย์เวิร์ดที่ใช้ค้นหาแทน อย่างที่กล่าวมาแล้วในส่วนต้นของบทความแล้วว่า Resource จริงๆ แล้วคือตัวที่ใช้บ่งชี้ถึงข้อมูลที่อยู่ในที่ server resource อาจเป็นไฟล์หรือเป็น Query String ที่ใช้ส่งไปเรียกโปรแกรมที่อยู่ในเซิร์ฟเวอร์ก็ได้ ซึ่งในกรณีตัวอย่างของ Search Engine ข้างต้น ตัว Resource ก็คือโปรแกรมที่ชื่อ search.cgi นั่นเอง โดยทั่วไปการตอบรับที่ได้กลับมาจากเซิร์ฟเวอร์อาจอยู่ในรูปของไฟล์ถ้า Resource ที่ร้องขอเป็นไฟล์

โดยทั่วไปเว็บเบราว์เซอร์(ไคลเอนต์) จะใช้เมธอด GET ในการส่งเมสเสจการร้องขอไปที่เซิร์ฟเวอร์ในขณะที่ Resource ที่ต้องการเป็นไฟล์ อย่างไรก็ตาม GET ยังสามารถใช้ในการส่ง Query String สั้นๆ ไปยังโปรแกรมที่รันอยู่เซิร์ฟเวอร์ได้อีกด้วย ตัวอย่างเช่น ถ้าต้องการส่งข้อมูลคีย์เวิร์ดโดยมีค่าเท่ากับ computer ไปที่เซิร์ฟเวอร์ชื่อ www.google.com โดยจะส่งไปที่ Resource ที่เป็น cgi โปรแกรมชื่อ search.cgi วิธีทำคือ

- กำหนด resource ลงไปส่วนของ request URI ซึ่งจะอยู่หลังจากส่วนของโฮสต์เนมซึ่งจะได้ <http://www.google.com/search.cgi>
- ใส่ตัวเครื่องหมายคำถามเพื่อบอก เซิร์ฟเวอร์ว่าส่วนที่เหลือถัดไปจะเป็นส่วนของข้อมูลซึ่งจะได้ <http://www.google.com/search.cgi?>
- ทำการเข้ารหัสโดยวิธีการที่เรียกว่า *URI-encoding* ไปที่ชื่อและค่าของตัวแปรต่างๆ ซึ่งในกรณีของตัวแปรก็คือคีย์เวิร์ดซึ่งจะได้ <http://www.google.com/search.cgi?keyword=name>

ถ้าหากพิมพ์ URL ข้างบนเข้าไปในเว็บเบราว์เซอร์แล้วกดปุ่ม enter ตัวเว็บเบราว์เซอร์จะทำการตีความ URL ดังกล่าวซึ่งผลที่ได้คือการให้ GET ส่งเมสเสจขอไปที่เซิร์ฟเวอร์ที่ชื่อ www.google.com โดยจะแนบคีย์เวิร์ด “computer” ไปกับการร้องขอซึ่งบรรทัดเริ่มต้นของเมสเสจการร้องขอจะเป็นอะไรคล้ายๆ ข้างล่างนี้

GET/search.cgi?keyword=jarticles HTTP/1.0

From: soup@jarticles.com

User-Agent: Mozilla/4.72

[blank line]

หลังจากนั้นจะได้สิ่งต่างๆ ที่เกี่ยวกับ computer กลับมา สังเกตว่า URL จริงๆ ที่ใช้กับ www.google.com จะต่างจาก URL ข้างล่างเล็กน้อย วิธีการส่ง query string ไปกับ URL เหมาะสำหรับ query string ที่ไม่ยาว

มากนัก โดยทั่วไปความของ URL มักจะจำกัดอยู่ที่ 80 ตัวอักษร ซึ่งโดยทั่วไปแล้วตัวอักษรที่ถัดจากนั้นจะถูกตัดออกไปโดยอัตโนมัติ

การเข้ารหัส URL (URL -Encoding)

ในการส่งข้อมูลไปที่ เซิร์ฟเวอร์โดยวิธีการ GET หรือ POST นั้นชื่อและค่าของตัวแปรต่างๆ ซึ่งอยู่หลังจากเครื่องหมายคำถามในกรณีของ GET หรืออยู่ในส่วนของ เมสเซจบอดีในกรณีของ POST จะต้องถูกผ่านการเข้ารหัสโดยวิธีการที่เรียกว่า *URI-Encoding* เสียก่อน โดยทั่วไปตัว URL เองจะมีตัวอักษรบางตัวที่ใช้สำหรับความหมายพิเศษ เช่น ;, /, ~, & และ ? ซึ่งในบางครั้งชื่อและค่าของตัวแปรต่างๆ ที่ถูกส่งไปที่ เซิร์ฟเวอร์อาจจะมีตัวอักษรเหล่านี้ปะปนอยู่ด้วย เพื่อหลีกเลี่ยงความสับสนในการตีความของ เซิร์ฟเวอร์ ตัวอักษรต่างๆ ที่เป็นที่ใช้ใน URL จะต้องถูกการเข้ารหัสเสียก่อน โดยมีหลักการดังต่อไปนี้ให้ทำการเปลี่ยน unsafe characters ต่างๆ ที่อยู่ในชื่อและค่าของตัวแปรให้กลายเป็น “%xx” โดย “xx” ก็คือค่าของแอสกีของตัวอักษรนั้นๆ ในแบบเลขฐาน 16 ซึ่ง ตัวอักษร unsafe เหล่านี้รวมไปถึง =, &, %, + และตัวอักษร ต่างๆ ที่ไม่สามารถพิมพ์ได้แม้กระทั่ง character ที่ต้องการจะเข้ารหัส

- ให้ทำการเปลี่ยนช่องว่าง(space) ทั้งหมดให้กลายเป็นเครื่องหมายบวก (+)
- จับคู่ชื่อและค่าของตัวแปรต่างๆ ด้วยเครื่องหมายเท่ากับ (=)
- ถ้าชื่อและค่าของตัวแปรมากกว่าหนึ่งตัว ให้นำชื่อและค่าของตัวแปรแต่ละคู่มาเชื่อมติดกันด้วยเครื่องหมาย &
- นำชื่อและค่าของตัวแปรที่เชื่อมติดกันทั้งหมดไปใส่ที่ URL โดยมีเครื่องหมาย ? อยู่ข้างหน้า(กรณีของ GET) หรือนำไปใส่ที่เมสเซจบอดี(กรณีของ POST) ยกตัวอย่างเช่น ถ้ามีชื่อและค่าดังนี้

(name, value) = ("keyword", "jarticles")

(name, value) = ("name", "Soup & friends")

หลังจากทำการเข้ารหัสจะได้ String ออกมาเป็น keyword=jarticles&name=Soup+%26+friends โดยStringนี้มีความยาวเท่ากับ 39 โดยที่ & = %26 ในเลขฐานสิบหก

5.3.2 เมธอด POST

บางครั้งหากไม่ต้องการส่งข้อมูลไปยังเซิร์ฟเวอร์ด้วยการติดข้อมูลเหล่านั้นไปกับ URL ด้วยเหตุผลที่ว่าข้อมูลเหล่านั้นอาจเกี่ยวข้องกับข้อมูลส่วนตัว ยกตัวอย่างเช่น login และ password ที่ใช้เป็นตัวผ่านเข้าไปในเว็บไซด์ต่างๆ ซึ่งในกรณีที่ใช้ GET, URL, หลังจากที่ทำ login ไปแล้ว URL ที่อยู่ตรง Address Bar อาจจะออกมาเป็น <http://www.myserver.com/login.cgi?login=me&password=youandme> ซึ่งอาจคนเห็นและนำข้อมูลไปใช้อย่างผิดๆ ได้อีกกรณีหนึ่งคือกรณีของข้อมูลที่ติดไปกับส่วนของ URL ที่ถูกร้องขอที่ทำให้ URL มีความยาวมากกว่าความยาวของ URL ที่เซิร์ฟเวอร์กำหนดไว้ ผลกระทบที่อาจเกิดขึ้นคือข้อมูลบางส่วนอาจหายไปเนื่องจากตัดทอนโดยเซิร์ฟเวอร์

วิธีการที่สามารถใช้เพื่อปิดบังข้อมูล หรือเพื่อส่งข้อมูลที่มีความยาวมากๆ ก็คือการใช้เมธอด POST โดยที่ POST ต่างกับ GET ตรงที่ POST จะไม่ทำการติดข้อมูลไปกับ URL แต่ POST จะทำการใส่ข้อมูลเข้าไปในส่วนของเมสเสจบอดีของเมสเสจร้องขอแทน ซึ่งในกรณีนี้ URI ที่ถูกร้องขอจะปรากฏเพียงชื่อของโปรแกรมที่ต้องการส่งข้อมูลไปให้เท่านั้น ในการส่งข้อมูลไปในส่วนของเมสเสจบอดีเพื่อต่อการตีความโดยเซิร์ฟเวอร์ทางเมธอด POST จึงมีการบังคับให้ใส่เฮดเดอร์พิเศษเพื่อบรรยายรายละเอียดต่างๆ ของข้อมูลที่อยู่ในเมสเสจบอดีนั่นอีกด้วย ซึ่งโดยทั่วไปเฮดเดอร์ที่มักถูกเพิ่มเข้าไปก็คือ Content-Type และ Content-Length การใช้ POST ที่มักพบเห็นโดยทั่วไปก็คือการใช้ POST ในการส่งข้อมูลที่มาจากฟอร์ม HTML ไคลเอนต์จะทำการเซต Content-Type เป็น application/x-www-form-urlencoded และ Content-Length จะเท่ากับความยาวของชื่อและค่าของอิลิเมนต์ต่างๆ ที่อยู่ในฟอร์มของ HTML รวมกันซึ่งตัวอย่างก็คืออิลิเมนต์ฟอร์มที่ชื่อ username และ password นอกจากนี้ POST ยังสามารถใช้กับการส่งข้อมูลแบบอื่นๆ ได้อีกด้วย โดยรูปแบบขึ้นอยู่กับวิธีการส่งที่ทางเว็บเบราว์เซอร์ และเว็บเซิร์ฟเวอร์ได้ทำการตกลงกันไว้ ยกตัวอย่างเช่น ในกรณีของไฟล์อัปโหลดตัว Content-Type ก็จะถูกเซตเป็น Multipart/form-data แทน



บทที่ 6

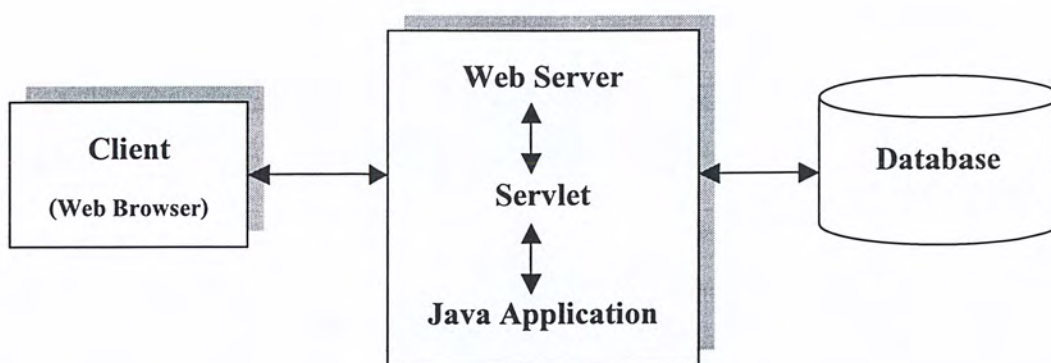
จาวาเซิร์ฟเลต (Java Servlet)

แม้ว่าโลกของอินเทอร์เน็ตจะเพิ่งเกิดขึ้นเพียงไม่กี่ปีก็ตาม แต่เทคโนโลยีที่ใช้กับอินเทอร์เน็ตกลับมีการเปลี่ยนแปลงไปอย่างรวดเร็วมาก ในสมัยแรกๆ หน้าเพจต่างๆ ที่อยู่ในเว็บจะเป็นลักษณะของสแตติกเพจหรือเพจที่ไม่มีการเปลี่ยนแปลงเนื้อหาที่ไม่ว่าจะนานเท่าไร นอกเสียจากว่าผู้ดูแลเพจนั้นจะทำการอัปเดตเพจดังกล่าว เพจลักษณะนี้เป็นเพจที่นิยมใช้กันทั่วไปในอินเทอร์เน็ตสมัยแรกเพราะอินเทอร์เน็ตยังนิยมกันอยู่ในวงแคบ โดยกลุ่มผู้ใช้จะเป็นกลุ่มบุคคลที่อยู่ในวงการศึกษาเท่านั้น ต่อมาจากนั้นไม่นานทางผู้ผลิตบราวเซอร์ได้ทำการเพิ่มความสามารถให้กับเพจโดยอนุญาตให้เพจสามารถแทรกสคริปต์เล็กๆ ลงไปพร้อมกับส่วนที่เป็น HTML ได้ซึ่งจุดนี้ก็คือจุดเริ่มต้นของจาวาสคริปต์ทางฝั่งไคลเอนต์นั่นเอง

แม้ว่าเพจจะเริ่มมีความสามารถในการโต้ตอบกับผู้ใช้โดยอิงความสามารถจากจาวาสคริปต์แล้วก็ตาม ถ้าพูดถึงในแง่ของส่วนเนื้อหาของตัวเพจจริงๆ แล้วตัวเพจเองก็ยังคงเป็นเพจสแตติก อยู่เช่นเดิม เมื่อกลุ่มผู้ใช้อินเทอร์เน็ตเริ่มมีมากขึ้นความต้องการที่จะทำให้เพจสามารถทำการรับส่งข้อมูลรวมไปถึงการเปลี่ยนแปลงเนื้อหาได้โดยอัตโนมัติก็เกิดขึ้น เทคโนโลยีที่เกิดขึ้นเพื่อรองรับความต้องการเหล่านี้ก็คือแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ (Server Side Application) นั่นเอง

6.1 ความหมายของเซิร์ฟเลต

เซิร์ฟเลตเป็นแอปพลิเคชันที่อยู่ทางฝั่งเซิร์ฟเวอร์แบบหนึ่งซึ่งอ้างอิงคอนเซ็ปต์มาจาก CGI (Common Gateway Interface) ข้อดีของเซิร์ฟเลตที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือภาษาจาวานั้นเอง จาวาเป็นภาษาที่ใช้คอนเซ็ปต์ของออบเจกต์-โอเรียนเตด (Object-Oriented) การเขียนโปรแกรมแบบออบเจกต์-โอเรียนเตด สามารถลดความซับซ้อนของโครงสร้างโปรแกรมรวมไปถึงการอำนวยความสะดวกในการนำส่วนของโปรแกรมที่เขียนไว้แล้วมาใช้ใหม่ได้ นอกจากนี้จาวายังเป็นภาษาที่เป็นลักษณะแบบไม่ขึ้นกับแพลตฟอร์มซึ่งจะช่วยให้เราสามารถที่จะทำการพัฒนาระบบโดยใช้สภาพแวดล้อมอะไรก็ได้ซึ่งโดยทั่วไปมักนิยมใช้ Window โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบนยูนิกซ์ เพื่อเพิ่มประสิทธิภาพของโปรแกรมแทน นอกจากนี้เซิร์ฟเลตยังมีความเร็วที่สูงกว่า CGI เพราะเซิร์ฟเลตใช้หลักการของเธรด (thread) โดยทำการสร้างหนึ่งเธรดต่อหนึ่งคำร้องขอที่มาจากไคลเอนต์ซึ่งในทางกลับกัน CGI จะทำการสร้างหนึ่งโพรเซส (process) ต่อหนึ่งคำร้องขอ ซึ่งจะทำให้เปลืองทรัพยากรมากกว่า และโพรเซสในการรันก็จะช้ากว่าด้วย ท้ายที่สุดจุดเด่นที่สำคัญของเซิร์ฟเลตก็คือ API (Application Programming Interface) โดยระบบที่ทำการพัฒนาโดยใช้คอนเซ็ปของเซิร์ฟเลตจะสามารถเรียกใช้ API ที่ทางจาวามีมาให้ซึ่งจะช่วยทำให้การพัฒนาระบบดังกล่าวง่ายและเร็วยิ่งขึ้น

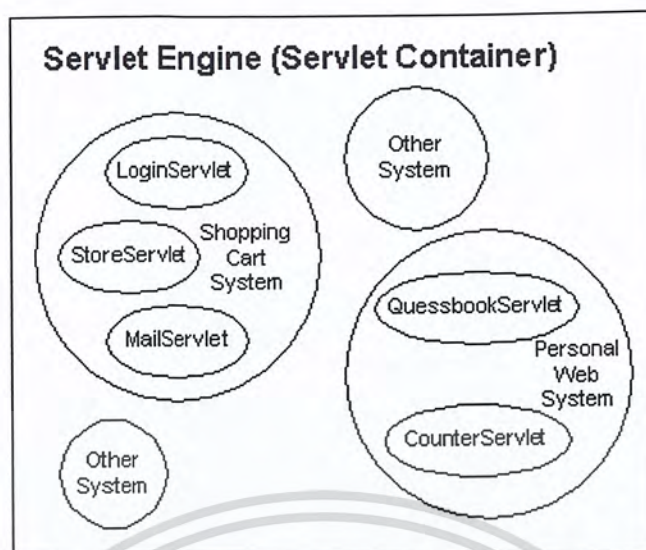


รูปที่ 6-1 หน้าที่และการทำงานของเซิร์ฟเลต

เมื่อมีการขอเรียกใช้บริการเซิร์ฟเลตมาถึงยังเซิร์ฟเวอร์ เซิร์ฟเลตเอนจินจะเรียกตัวโหลดคลาส(class loader) ตัวโหลดคลาสจะทำการตรวจสอบว่าคลาสดังกล่าวถูกโหลดขึ้นมาเรียบร้อยแล้วหรือไม่ และตรวจสอบเวอร์ชันของคลาสที่ถูกโหลดว่ามีค่าไทม์แสตมป์ (time stamp) เท่ากับ .class ไฟล์ (.class file) ที่อยู่บนดิสก์ (disk) หรือไม่ ถ้าคลาสของเซิร์ฟเลตยังไม่ถูกโหลด หรือมีเวอร์ชันต่ำกว่าไฟล์ที่อยู่บนดิสก์ เซิร์ฟเลตเอนจินจะทำการ โหลดคลาสไปเก็บไว้ใน JVM และสร้างอินสแตนซ์ (instance) ของคลาสดังกล่าว หลังจากนั้นวงจรชีวิตของเซิร์ฟเลตจะเริ่มขึ้นใหม่

6.2 เซิร์ฟเลตเอนจิน (Servlet engine)

ในการรันระบบที่เขียนขึ้นโดยใช้หลักการของเซิร์ฟเลตเราจะต้องนำระบบดังกล่าวมาบรรจุอยู่ในสิ่งๆ หนึ่งที่เรียกว่าเซิร์ฟเลตเอนจิน ให้นึกว่าเซิร์ฟเลตเอนจินคล้ายๆกับกล่องๆหนึ่งที่ใส่ลูกปิงปองไว้หลายลูก โดยลูกปิงปองแต่ละลูกก็คือระบบๆ หนึ่งนั่นเอง หลายคนอาจสงสัยทำไมถึงใช้คำว่าระบบ โดยทั่วไปแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์หลายๆ ที่ถูกเขียนขึ้นโดยใช้ API ของเซิร์ฟเลตจะถูกเรียกว่าเซิร์ฟเลต ในหนึ่งระบบอาจประกอบด้วยเซิร์ฟเลตหลายอัน ยกตัวอย่างเช่น ระบบที่เกี่ยวกับ Shopping Cart อาจจะประกอบด้วยเซิร์ฟเลตที่ทำหน้าที่ในการเช็คสต็อกอิน, เซิร์ฟเลตที่ทำหน้าที่ในการเก็บข้อมูลสินค้า, เซิร์ฟเลตที่ทำหน้าที่ในการส่งเมลกลับไปยังลูกค้าเพื่อบอกว่าได้ทำการส่งของไปให้แล้ว เป็นต้น ดังนั้นถ้ามองโดยรวมแล้วเซิร์ฟเลตเอนจินก็คือที่รวมของระบบตั้งแต่หนึ่งระบบถึงหลายระบบ โดยแต่ละระบบจะประกอบด้วยเซิร์ฟเลตหนึ่งอันหรือมากกว่า ดังรูปที่ 6-2 ซึ่งระบบในที่นี้อาจจะหมายถึง Zone (Apache Jserv) หรือ Web Application (Tomcat) ก็ได้



รูปที่ 6-2 เซิร์ฟเลตเอนจิน และเซิร์ฟเลตที่อยู่ภายใน

เซิร์ฟเลตเอนจินเป็นเพียงกล่องๆหนึ่งที่ใช้บรรจุและรันกลุ่มของเซิร์ฟเลตเท่านั้น ในการที่จะทำการติดต่อสื่อสารกับไคลเอนต์ ตัวเซิร์ฟเลตเอนจินนี้จะต้องทำงานร่วมกับเว็บเซิร์ฟเวอร์ ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับไคลเอนต์อีกทีหนึ่ง เมื่อใดก็ตามที่มีคำร้องขอส่งมาจากไคลเอนต์ ถ้าคำร้องขอนั้นจะจงมาที่ตัวเซิร์ฟเลต ทางเว็บเซิร์ฟเวอร์ก็จะทำการส่งคำร้องขอนั้นมาให้เซิร์ฟเลตเอนจิน ซึ่งทางเซิร์ฟเลตเอนจินก็จะทำการเรียกเซิร์ฟเลตที่ไคลเอนต์ต้องการขึ้นมาทำการประมวลผลคำร้องขอนั้น โดยท้ายสุดเซิร์ฟเลตจะส่งผลกลับไปให้เซิร์ฟเลตเอนจิน เซิร์ฟเลตเอนจินก็จะส่งผลที่ได้กลับไปให้เว็บเซิร์ฟเวอร์ ซึ่งเว็บเซิร์ฟเวอร์ก็จะส่งผลกลับไปให้ไคลเอนต์

เซิร์ฟเลตเอนจินอาจจะเป็นส่วนที่ติดมากับเว็บเซิร์ฟเวอร์อยู่แล้วยกตัวอย่างเช่น เซิร์ฟเลตเอนจินที่อยู่ในเซิร์ฟเวอร์ของ Netscape Enterprise, IBM Web Sphere หรืออาจจะเป็นส่วนที่เป็นส่วนที่เพิ่มเติมให้กับเว็บเซิร์ฟเวอร์ก็ได้เช่น Apache Jserv, Tomcat, JRun หรือแม้กระทั่งเป็นส่วนหนึ่งที่อยู่ในเว็บแอปพลิเคชัน เช่น BEA Web logic เป็นต้น ทั้งนี้การเลือกใช้เซิร์ฟเลตเอนจินแต่ละชนิดก็มักขึ้นอยู่กับปัจจัยหลายอย่างเช่น ความสะดวกในการรวมระบบที่จะสร้างขึ้นใหม่กับระบบที่มีอยู่แล้ว, งบประมาณที่มีอยู่สำหรับโครงการหรืออาจจะรวมไปถึงทักษะและประสบการณ์ส่วนตัวของนักพัฒนาแต่ละคน

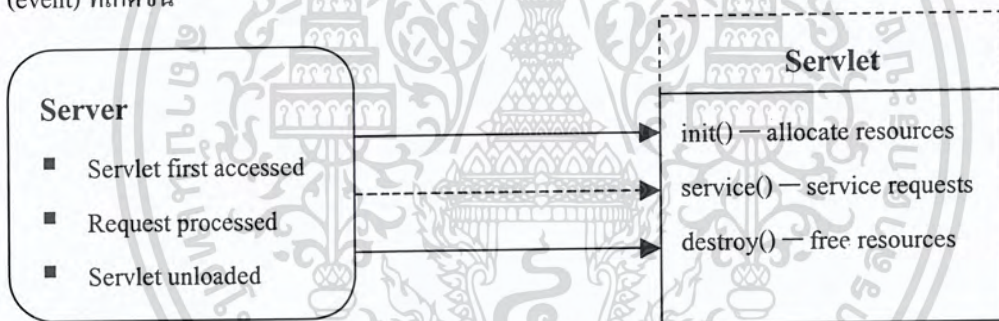
6.3 อินเทอร์เฟซ javax.servlet.Servlet

หัวใจของเซิร์ฟเลตจริงๆอยู่ที่อินเทอร์เฟซที่ชื่อ javax.servlet.Servlet* โดยทุกเซิร์ฟเลตที่ถูกเขียนขึ้นจะต้องทำการอิมพลีเมนต์ตัวอินเทอร์เฟซนี้ไม่ว่าทางตรงก็ทางอ้อม (ทางตรงก็คือการอิมพลีเมนต์ตัวอินเทอร์เฟซนี้เลย ส่วนทางอ้อมก็คือการให้เซิร์ฟเลตทำการแบ่งย่อยคลาสบางคลาสที่ได้ทำการอิมพลีเมนต์ตัวอินเทอร์เฟซนี้ไว้แล้ว) เหตุผลว่าทำไมเราต้องอิมพลีเมนต์ตัวอินเทอร์เฟซนี้เพราะว่าเมื่อไรก็ตามที่มีคำร้องขอจากไคลเอนต์เข้ามายังเซิร์ฟเลตเอนจิน ตัวเซิร์ฟเลตเอนจินจะทำการหาเซิร์ฟเลตที่การร้องขอดังกล่าวอ้างถึง หลังจากนั้นเซิร์ฟเลตเอนจินจะทำการเรียกฟังก์ชันต่างๆที่อยู่ในเซิร์ฟเลตเพื่อทำการประมวลผลค่า

ร้องขอของไคลเอ็นต์ โดยฟังก์ชันที่เซิร์ฟเลตเอนจินจะทำการเรียกก็คือฟังก์ชันที่เซิร์ฟเลตได้ทำการอิมพลีเมนต์ ซึ่งเป็นฟังก์ชันที่ถูกกำหนดอยู่ใน `javax.servlet.Servlet` อินเตอร์เฟสนั่นเอง อาจมีคำถามว่าทำไมเซิร์ฟเลตเอนจินถึงเรียกฟังก์ชันที่ถูกกำหนดอยู่ในอินเตอร์เฟสนี้ เหตุผลง่าย ๆ ก็คือเซิร์ฟเลตเป็นส่วนที่ถูกโหลดเข้าไปในเซิร์ฟเลตเอนจินในช่วง Runtime ตัวเซิร์ฟเลตเอนจินเองไม่สามารถทราบได้ว่าเซิร์ฟเลตต่าง ๆ มีฟังก์ชันอะไรประกอบอยู่บ้างนอกเสียจากว่าเซิร์ฟเลตนั้นได้ทำการอิมพลีเมนต์ฟังก์ชันที่เป็นมาตรฐานที่เซิร์ฟเลตเอนจินรับรู้ ซึ่งนี่ก็คือเหตุผลว่าทำไมทุกเซิร์ฟเลตจะต้องทำการอิมพลีเมนต์ตัว `javax.servlet.Servlet` อินเตอร์เฟส อย่างไรก็ตามเราสามารถให้เซิร์ฟเลตเอนจินเรียกฟังก์ชันอื่นๆที่อยู่ในเซิร์ฟเลตได้ ซึ่งวิธีการก็คือการใส่ฟังก์ชันดังกล่าวเข้าไปในส่วนอิมพลีเมนต์ของฟังก์ชันต่างๆที่ถูกกำหนดอยู่ใน `javax.servletServlet` อินเตอร์เฟส

6.4 วงจรชีวิตของเซิร์ฟเลต (Servlet's Life Cycle)

วงจรชีวิตของเซิร์ฟเลตจะประกอบด้วย 3 สถานะคือ สถานะเริ่มต้น (`init()`), สถานะให้บริการ (`service()`) และสถานะทำลาย (`destroy()`) เมธอดเหล่านี้เป็น “Callback method” หมายถึงเมธอดจะไม่ถูกเรียกโดยตัวเซิร์ฟเลตเอง แต่จะถูกเรียกโดยเซิร์ฟเลตเอนจิน ในการตอบสนองกับเหตุการณ์ (event) ที่เกิดขึ้น



รูปที่ 6-3 วงจรชีวิตของเซิร์ฟเลต

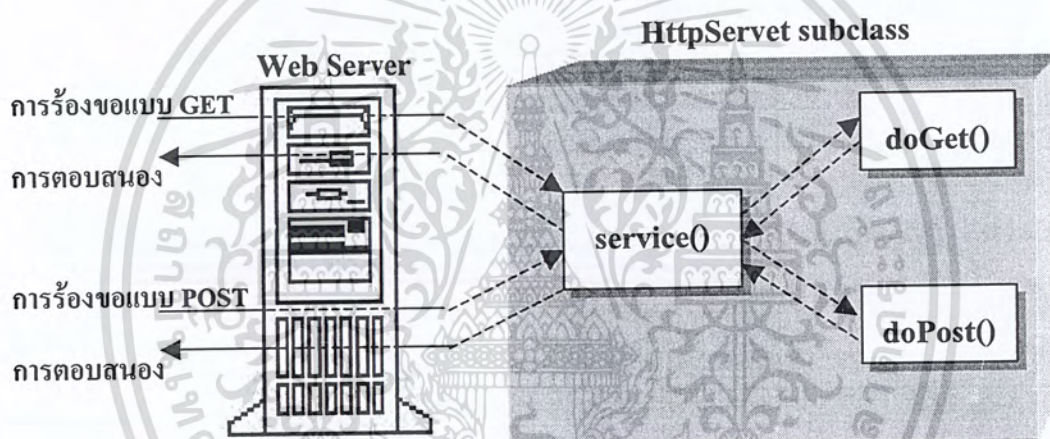
6.4.1 สถานะเริ่มต้น (Initial State)

เป็นการใช้เมธอด `init()` ซึ่งเมธอดจะถูกเรียกโดยเซิร์ฟเวอร์ทันทีที่อินสแตนซ์ (instance) ถูกสร้างเสร็จเรียบร้อยแล้ว โดยปกติเมธอดนี้จะถูกใช้เพื่อทำการเริ่มต้นเซิร์ฟเลต สาเหตุที่ไม่ใช้คอนสตรัคเตอร์ (constructor) เพราะคอนสตรัคเตอร์ของ JDK 1.0 (เริ่มมีเซิร์ฟเลต) ไม่สามารถรับอาร์กิวเมนต์ (argument) ในกรณีที่เกิดการโหลดแบบไดนามิก (dynamic) ได้ ดังนั้นการที่จะจัดหาข้อมูลเกี่ยวกับตัวมันเองและสิ่งแวดล้อมให้แก่เซิร์ฟเลตที่ถูกโหลดเป็นครั้งแรก เซิร์ฟเวอร์จำเป็นต้องเรียกเมธอด `init()` และส่งผ่านออบเจกต์ (object) ที่ใช้อินเตอร์เฟส `ServletConfig` นอกจากนี้ภาษาจาวาไม่อนุญาตให้อินเตอร์เฟสมีการประกาศคอนสตรัคเตอร์ นั่นหมายถึงอินเตอร์เฟสของแพ็คเกจ (package) `javax.servlet.Servlet` ก็ไม่สามารถประกาศคอนสตรัคเตอร์ที่รับค่าพารามิเตอร์ (parameter) `ServletConfig` ได้เช่นกัน จึงต้องประกาศเมธอดชื่ออื่นแทนเช่น `init()` แต่ยังคงมีทางเป็นไปได้ที่จะประกาศคอนสตรัคเตอร์ให้แก่เซิร์ฟเลต แต่ใน

คอนสตรัคเตอร์ดังกล่าวห้ามมีการเข้าถึงออบเจกต์ ServletConfig และไม่สามารถ throw ServletException ได้ ออบเจกต์ ServletConfig จะส่งพารามิเตอร์ที่จำเป็นต้องใช้ในการเริ่มต้นการทำงานให้แก่เซิร์ฟเลต พารามิเตอร์ดังกล่าวจะถูกป้อนให้แก่เซิร์ฟเลต โดยไม่เกี่ยวข้องกับการร้องขอที่เข้ามา

6.4.2 สถานะให้บริการ (Service State)

สถานะให้บริการจะถูกเรียกจากเซิร์ฟเวอร์ทุกครั้งที่ไคลเอนต์ทำการร้องขอใช้บริการเซิร์ฟเลต เมธอด service() จะรับค่าพารามิเตอร์ 2 ค่าได้แก่ออบเจกต์การร้องขอและออบเจกต์การตอบสนอง ออบเจกต์การร้องขอจะบอกเซิร์ฟเลตเกี่ยวกับการขอใช้บริการที่มีเข้ามา ในขณะที่ออบเจกต์การตอบสนองจะถูกเรียกใช้ในการส่งการตอบรับคืนกลับให้แก่เซิร์ฟเวอร์ โดยปกติแล้วเมธอด service() จะไม่ถูกโอเวอร์ไรด์ (override) แต่จะทำการโอเวอร์ไรด์เมธอด doGet() เพื่อใช้จัดการการร้องขอแบบ GET และเมธอด doPost() สำหรับจัดการกับการร้องขอแบบ POST แทน ดังตัวอย่างในรูป



รูปที่ 6-4 วิธีการจัดการกับร้องขอแบบ GET และ POST ของเซิร์ฟเลต

6.4.3 สถานะทำลาย (Destroy State)

สถานะทำลายเซิร์ฟเวอร์จะเรียกเมธอด destroy() ของเซิร์ฟเลต เมื่อเซิร์ฟเลตกำลังจะถูกอันโหลด(unload) ในเมธอด destroy() ควรมีการคืนทรัพยากรต่างๆกลับคืนมาให้แก่ระบบ เพื่อที่โปรแกรมอื่นๆจะได้นำทรัพยากรเหล่านี้ไปใช้ได้ นอกจากนี้เมธอดยังให้โอกาสเซิร์ฟเลตในการบันทึกสถานะของข้อมูลต่างๆที่ถูกเรียกใช้เป็นประจำจากการเรียกเมธอด init() ในครั้งต่อไป

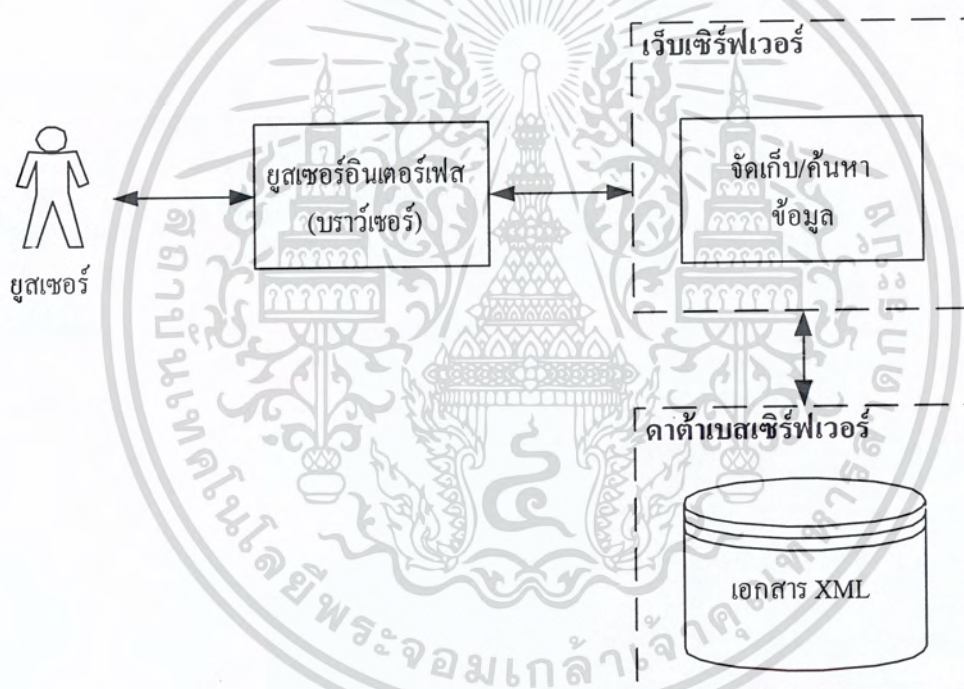
บทที่ 7

การสร้างและการออกแบบ

โครงการห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์นี้ คณะผู้จัดทำโครงการได้ใช้ภาษาจาวาเซิร์ฟเลต (Java Servlet) เป็นตัวหลักในการพัฒนาระบบ และใช้ DB2 ของ IBM เป็นตัวจัดการฐานข้อมูล (Database Management System : DBMS)

7.1 ภาพรวมของระบบ

ระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์นี้ จะมีโครงสร้างแบบ 3-tier architecture ซึ่งจะประกอบไปด้วยเว็บเซิร์ฟเวอร์, ดาต้าเบสเซิร์ฟเวอร์และยูสเซอร์อินเตอร์เฟส ซึ่งมีลักษณะดังรูป



รูปที่ 7-1 ภาพรวมของระบบ

ซึ่งขั้นตอนการออกแบบและพัฒนา สามารถแบ่งออกเป็นหัวข้อหลักๆ ได้ดังนี้

1. การนำ XML มาใช้เก็บข้อมูลที่เป็นวิทยานิพนธ์
2. การจัดเก็บข้อมูลที่เป็น XML ลงในดาต้าเบสเซิร์ฟเวอร์
3. การนำข้อมูลที่จัดเก็บในดาต้าเบสเซิร์ฟเวอร์มาแสดงผล

7.2 การนำ XML มาใช้เก็บข้อมูลที่เป็นวิทยานิพนธ์

ในขั้นตอนนี้เราจะต้องมาทำการออกแบบว่าเราต้องการที่จะเก็บข้อมูลวิทยานิพนธ์ส่วนไหนลงใน XML บ้างและมีโครงสร้างอย่างไร โครงสร้างของเอกสาร XML สามารถกำหนดได้โดยใช้ DTD ซึ่งในโครงงานนี้ได้ใช้ DTD ดังนี้

```
<!DOCTYPE LIBRARY_ITEM [
    <!ELEMENT LIBRARY_ITEM (TITLES, MAJOR, SUB_MAJOR, YEAR, CREATORS,
        PUBLISHER, ABSTRACT_CONTENT, KEYWORDS)>
    <!ELEMENT TITLES (THAI_TITLE, ENGLISH_TITLE)>
    <!ELEMENT THAI_TITLE (#PCDATA)>
    <!ELEMENT ENGLISH_TITLE (#PCDATA)>
    <!ELEMENT MAJOR (MAJOR_THAI_NAME, MAJOR_ENGLISH_NAME)>
    <!ELEMENT MAJOR_THAI_NAME (#PCDATA)>
    <!ELEMENT MAJOR_ENGLISH_NAME (#PCDATA)>
    <!ELEMENT SUB_MAJOR (SUB_MAJOR_THAI_NAME,
        SUB_MAJOR_ENGLISH_NAME)>
    <!ELEMENT SUB_MAJOR_THAI_NAME (#PCDATA)>
    <!ELEMENT SUB_MAJOR_ENGLISH_NAME (#PCDATA)>
    <!ELEMENT YEAR (#PCDATA)>
    <!ELEMENT CREATORS (CREATOR+)>
    <!ELEMENT CREATOR (FIRST_NAME, LAST_NAME)>
    <!ELEMENT FIRST_NAME (#PCDATA)>
    <!ELEMENT LAST_NAME (#PCDATA)>
    <!ELEMENT PUBLISHER (PUBLISHER_THAI_NAME,
        PUBLISHER_ENGLISH_NAME)>
    <!ELEMENT PUBLISHER_THAI_NAME (#PCDATA)>
    <!ELEMENT PUBLISHER_ENGLISH_NAME (#PCDATA)>
    <!ELEMENT ABSTRACT_CONTENT (THAI_ABSTRACT, ENGLISH_ABSTRACT)>
    <!ELEMENT THAI_ABSTRACT (#PCDATA)>
    <!ELEMENT ENGLISH_ABSTRACT (#PCDATA)>
    <!ELEMENT KEYWORDS (KEYWORD+)>
    <!ELEMENT KEYWORD (#PCDATA)>
]>
```

ในโครงการนี้ได้นำ “ ดัชนีคอร์เมตาดาต้า ฉบับ 1.1 ภาษาไทย ” ซึ่งเป็นมาตรฐานในการกำหนดคำจำกัดความเพื่อให้มีความเข้าใจที่ตรงกัน มาใช้ในการกำหนดความหมายของ DTD ซึ่งมีความหมายของแต่ละอิลิเมนต์ดังนี้

อิลิเมนต์	ความหมาย
LIBRARY_ITEM	ที่เก็บวิทยานิพนธ์
TITLES	ที่เก็บชื่อเรื่อง
THAI_TITLE	ชื่อเรื่องภาษาไทย
ENGLISH_TITLE	ชื่อเรื่องภาษาอังกฤษ
MAJOR	สาขาวิชาของวิทยานิพนธ์
MAJOR_THAI_NAME	ชื่อสาขาวิชาของวิทยานิพนธ์ภาษาไทย
MAJOR_ENGLISH_NAME	ชื่อสาขาวิชาของวิทยานิพนธ์ภาษาอังกฤษ
SUB_MAJOR	สาขาวิชาย่อยของวิทยานิพนธ์
SUB_MAJOR_THAI_NAME	ชื่อสาขาวิชาย่อยของวิทยานิพนธ์ภาษาไทย
SUB_MAJOR_ENGLISH_NAME	ชื่อสาขาวิชาย่อยของวิทยานิพนธ์ภาษาอังกฤษ
YEAR	ปีที่ทำวิทยานิพนธ์
CREATORS	ที่เก็บชื่อเจ้าของงาน
CREATOR	เจ้าของงาน
FIRST_NAME	ชื่อเจ้าของงาน
LAST_NAME	นามสกุลเจ้าของงาน
PUBLISHER	สำนักพิมพ์หรือสถาบันการศึกษาที่เป็นเจ้าของวิทยานิพนธ์
PUBLISHER_THAI_NAME	ชื่อภาษาไทยของสำนักพิมพ์หรือสถาบันการศึกษาที่เป็นเจ้าของวิทยานิพนธ์
PUBLISHER_ENGLISH_NAME	ชื่อภาษาอังกฤษของสำนักพิมพ์หรือสถาบันการศึกษาที่เป็นเจ้าของวิทยานิพนธ์
ABSTRACT_CONTENT	บทคัดย่อ
THAI_ABSTRACT	บทคัดย่อที่เป็นภาษาไทย
ENGLISH_ABSTRACT	บทคัดย่อที่เป็นภาษาอังกฤษ
KEYWORDS	ส่วนของคีย์เวิร์ด
KEYWORD	คีย์เวิร์ด

ตารางที่ 7-1 ความหมายของอิลิเมนต์ใน XML

ข้อมูลด้านล่างนี้เป็นตัวอย่างข้อมูลวิทยานิพนธ์

สาขาวิชาเทคโนโลยีพอลิเมอร์	
หัวข้อวิทยานิพนธ์	การเตรียมไฮโดรเจลจากเศษไหมเหลือทิ้งที่ผ่านการปรับปรุงกับพอลิไวน์ิลแอลกอฮอล์
นักศึกษา	นางสาวจิรพรรณ หน่ายคอน
รหัสประจำตัว	42065105
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีพอลิเมอร์
สถาบัน	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2544
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร.มาลินี ชัยศุกกิจสินธ์
บทคัดย่อ	
ในประเทศไทยมีเศษไหมที่เหลือทิ้งจากอุตสาหกรรมสิ่งทอมากมาย งานวิจัยนี้จึงเป็นการนำเศษไหมที่เหลือทิ้งนั้นมาใช้ไหมเพื่อช่วยเพิ่มคุณค่าของเศษไหม โดยนำเศษไหมมาผสมกับพอลิไวน์ิลแอลกอฮอล์ แล้วทำการฉายรังสีเพื่อก่อให้เกิดไฮโดรเจล จากนั้นศึกษาถึงผลของปริมาณรังสี อัตราส่วนของไหมและพอลิไวน์ิลแอลกอฮอล์ และน้ำหนักโมเลกุลของพอลิไวน์ิลแอลกอฮอล์ ที่มีต่อสมบัติทางกายภาพและสมบัติเชิงกลของไฮโดรเจล จากผลการทดลอง พบว่าลักษณะทางกายภาพที่แตกต่างกันของไฮโดรเจลเป็นผลเนื่องมาจากความเข้มข้นของไหม ความเข้มข้นของพอลิไวน์ิลแอลกอฮอล์ อัตราส่วนของไหมและพอลิไวน์ิลแอลกอฮอล์ น้ำหนักโมเลกุลของพอลิไวน์ิลแอลกอฮอล์ และความเข้มของรังสี นอกจากนี้ยังพบว่าปริมาณการเป็นเจล การดูดซับน้ำ และความแข็งแรงของเจล เป็นผลเนื่องจากพอลิไวน์ิลแอลกอฮอล์มากกว่าไหม การผสมพอลิไวน์ิลแอลกอฮอล์กับไหมช่วยเพิ่มคุณสมบัติดูดซับน้ำของไฮโดรเจล พอลิไวน์ิลแอลกอฮอล์ชนิดน้ำหนักโมเลกุลต่ำ (11,000 – 31,000) ไม่เหมาะต่อการเตรียมไฮโดรเจล การผสมไหม 1% และพอลิไวน์ิลแอลกอฮอล์ 3% ในอัตราส่วน 1:3 พร้อมทั้งทำการฉายรังสีด้วยความเข้ม 30 กิโลเกรย์ เป็นสูตรที่เหมาะสมในการเตรียมไฮโดรเจลเนื่องจากมีการดูดซับน้ำที่ดี มีความแข็งแรงพอสมควร และใช้ความเข้มรังสีไม่สูงจนเกินไป	
คำสำคัญ	ไฮโดรเจล, ไหม, พอลิไวน์ิลแอลกอฮอล์
Thesis Title	Preparation of Hydrogel from Modified Silk Waste and Poly(VinylAlcohol)
Student	Miss Jirapan Naikhorn
Student ID.	42065105

Programme Polymer Technology
Institute King Mongkut's Institute of Technology Ladkrabang
Year 2001
Thesis Advisor Asst. Prof. Dr. Malinee chaisupakitsin

ABSTRACT

In Thailand, there is much of silk waste from textile industry. This research focused on preparation of hydrogel from modified silk waste(SF) and poly(vinyl alcohol)(PVA) for value added to the silk waste. The hydrogels were prepared by γ -irradiation technique. The research aimed to examine the influence of radiation dose, ratio of SF/PVA, and molecular weight of PVA on physical and mechanical properties of the hydrogels. It was found that physical appearance of the hydrogels depended on concentration of the silk and PVA, ratio of silk/PVA, molecular weight of PVA, and irradiation dose. Gel fraction, water absorption, and gel strength of the hydrogels depended on PVA more than silk. Addition of PVA into silk could increase %water absorption of hydrogel. Low molecular weight PVA(11,000 — 31,000) was not suitable for preparing the hydrogel. The appropriate formula was 1% silk / 3%PVA at ratio 1:3 and irradiated at 30 kGy because it gave high water absorption, appropriate gel strength, at an appropriate irradiation dose.

Keyword : Hydrogel, Silk, Poly(VinylAlcohol)

เมื่อทำการแปลงไปเป็นเอกสารที่อยู่ในรูปแบบ XML จะได้ดังนี้

```
<?xml version="1.0" ?>
<LIBRARY_ITEM>
<TITLES>
  <THAI_TITLE>การเตรียมไฮโดรเจลจากเศษไหมเหลือทิ้งที่ผ่านการปรับปรุงกับพอลิไวนิล
  แอลกอฮอล์</THAI_TITLE>
  <ENGLISH_TITLE>Preparation of Hydrogel from Modified Silk Waste and
  Poly(VinylAlcohol)</ENGLISH_TITLE>
</TITLES>
<MAJOR>
  <MAJOR_THAI_NAME>วิทยาศาสตร์</MAJOR_THAI_NAME>
  <MAJOR_ENGLISH_NAME>Science</MAJOR_ENGLISH_NAME>
</MAJOR>
<SUB_MAJOR>
```

```

<SUB_MAJOR_THAI_NAME>เทคโนโลยีพอลิเมอร์</SUB_MAJOR_THAI_NAME>
<SUB_MAJOR_ENGLISH_NAME> Polymer Technology
</SUB_MAJOR_ENGLISH_NAME>
</SUB_MAJOR>
<YEAR>2544</YEAR>
<CREATORS>
  <CREATOR>
    <FIRST_NAME>จิรพรรณ</FIRST_NAME>
    <LAST_NAME>หน้าคอน</LAST_NAME>
  </CREATOR>
  <CREATOR>
    <FIRST_NAME>มาลินี</FIRST_NAME>
    <LAST_NAME>ชัยสุภกิจสินธ์</LAST_NAME>
  </CREATOR>
</CREATORS>
<PUBLISHER>
  <PUBLISHER_THAI_NAME>สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
  </PUBLISHER_THAI_NAME>
  <PUBLISHER_ENGLISH_NAME> King Mongkut's Institute of Technology Ladkrabang
  </PUBLISHER_ENGLISH_NAME>
</PUBLISHER>
<ABSTRACT_CONTENT>
  <THAI_ABSTRACT>ในประเทศไทยมีเศษไหมที่เหลือทิ้งจากอุตสาหกรรมสิ่งทอมากมาย งาน
  วิจัยนี้จึงเป็นการนำเศษไหมที่เหลือทิ้งนั้นมาใช้ใหม่เพื่อช่วยเพิ่มคุณค่าของเศษไหม โดยนำเศษ
  ไหมมาผสมกับพอลิไวนิลแอลกอฮอล์ แล้วทำการฉายรังสีเพื่อก่อให้เกิดไฮโดรเจล จากนั้นศึกษา
  ถึงผลของปริมาณรังสี อัตราส่วนของไหมและพอลิไวนิลแอลกอฮอล์ และน้ำหนักโมเลกุลของพอลิ
  ไวนิลแอลกอฮอล์ ที่มีต่อสมบัติทางกายภาพและสมบัติเชิงกลของไฮโดรเจล
  จากผลการทดลอง พบว่าลักษณะทางกายภาพที่แตกต่างกันของไฮโดรเจลเป็นผลเนื่องมาจาก
  ความเข้มข้นของไหม ความเข้มข้นของพอลิไวนิลแอลกอฮอล์ อัตราส่วนของไหมและพอลิไวนิล
  แอลกอฮอล์ น้ำหนักโมเลกุลของพอลิไวนิลแอลกอฮอล์ และความเข้มข้นของรังสี นอกจากนี้ยังพบ
  ว่าปริมาณการเป็นเจล การดูดซับน้ำ และความแข็งแรงของเจล เป็นผลเนื่องจากพอลิไวนิล
  แอลกอฮอล์มากกว่าไหม การผสมพอลิไวนิลแอลกอฮอล์กับไหมช่วยเพิ่มคุณสมบัติดูดซับน้ำของ
  ไฮโดรเจล พอลิไวนิลแอลกอฮอล์ชนิดน้ำหนักโมเลกุลต่ำ (11,000 – 31,000) ไม่เหมาะต่อการ
  เตรียมไฮโดรเจล การผสมไหม 1% และพอลิไวนิลแอลกอฮอล์ 3% ในอัตราส่วน 1:3 พร้อมทั้งทำ

```

การฉายรังสีด้วยความเข้ม 30 กิโลเกรย์ เป็นสูตรที่เหมาะสมในการเตรียมไฮโดรเจลเนื่องจากการดูดซับน้ำที่ดี มีความแข็งแรงพอสมควร และใช้ความเข้มรังสีไม่สูงจนเกินไป

<ENGLISH_ABSTRACT>In Thailand, there is much of silk waste from textile industry. This research focused on preparation of hydrogel from modified silk waste(SF) and poly(vinyl alcohol)(PVA) for value added to the silk waste. The hydrogels were prepared by γ -irradiation technique. The research aimed to examine the influence of radiation dose, ratio of SF/PVA, and molecular weight of PVA on physical and mechanical properties of the hydrogels. It was found that physical appearance of the hydrogels depended on concentration of the silk and PVA, ratio of silk/PVA ,molecular weight of PVA ,and irradiation dose. Gel fraction ,water absorption ,and gel strength of the hydrogels depended on PVA more than silk. Addition of PVA into silk could increase %water absorption of hydrogel. Low molecular weight PVA(11,000 — 31,000) was not suitable for preparing the hydrogel. The appropriate formula was 1%silk / 3%PVA at ratio 1:3 and irradiated at 30 kGy because it gave high water absorption, appropriate gel strength, at an appropriate irradiation dose</ENGLISH_ABSTRACT>

</ABSTRACT_CONTENT>

<KEYWORDS>

<KEYWORD> Hydrogel</KEYWORD>

<KEYWORD>Silk</KEYWORD>

<KEYWORD>Poly(VinylAlcohol)</KEYWORD>

<KEYWORD>ไฮโดรเจล</KEYWORD>

<KEYWORD>ไหม</KEYWORD>

<KEYWORD>พอลิไวนิลแอลกอฮอล์</KEYWORD>

</KEYWORDS>

</LIBRARY_ITEM>

7.3 การจัดเก็บข้อมูลที่เป็น XML ลงในดาต้าเบสเซิร์ฟเวอร์

ในขั้นตอนของการจัดเก็บ XML ลงในดาต้าเบสเซิร์ฟเวอร์ นั้น จะต้องทำการสร้าง Database Schema ซึ่งสามารถทำได้โดยการแมปจาก DTD มาเป็น Database Schema โดยในที่นี้ได้นำเอาหลักการจากวิทยานิพนธ์เรื่อง “AN XML INFORMATION RETRIEVAL SYSTEM ” มาใช้โดยเมื่อแมปเป็น Database Schema แล้วทำการ Normalize แล้วจะได้ ตารางดังนี้

MAJOR

COLUMN NAME	TYPE	SIZE	KEY
MAJOR_ID	INTEGER	-	P.K.
MAJOR_THAI_NAME	VARCHAR	100	-
MAJOR_ENGLISH_NAME	VARCHAR	100	-

ตารางที่ 7-2 ข้อมูลสาขาวิชาของวิทยานิพนธ์

SUB_MAJOR

COLUMN NAME	TYPE	SIZE	KEY
SUB_MAJOR_ID	INTEGER	-	P.K.
MAJOR_ID	INTEGER	-	P.K., F.K.
SUB_MAJOR_THAI_NAME	VARCHAR	100	-
SUB_MAJOR_ENGLISH_NAME	VARCHAR	100	-

ตารางที่ 7-3 ข้อมูลสาขาวิชาย่อยของวิทยานิพนธ์

PUBLISHER

COLUMN NAME	TYPE	SIZE	KEY
PUBLISHER_ID	INTEGER	-	P.K.
PUBLISHER_THAI_NAME	VARCHAR	100	-
PUBLISHER_ENGLISH_NAME	VARCHAR	100	-

ตารางที่ 7-4 ข้อมูลสำนักพิมพ์หรือสถาบันการศึกษาที่เป็นเจ้าของวิทยานิพนธ์

CREATOR

COLUMN NAME	TYPE	SIZE	KEY
CREATOR_ID	INTEGER	-	P.K.
FIRST_NAME	VARCHAR	30	-
LAST_NAME	VARCHAR	30	-

ตารางที่ 7-5 ข้อมูลผู้จัดทำวิทยานิพนธ์

LIBRARY_ITEM

COLUMN NAME	TYPE	SIZE	KEY
LIBRARY_ITEM_ID	INTEGER	-	P.K.
THAI_TITLE	VARCHAR	30	-
ENGLISH_TITLE	VARCHAR	30	-
THAI_ABSTRACT	LONG VARCHAR	-	-
ENGLISH_ABSTRACT	LONG VARCHAR	-	-
PDF_FILE	BLOB	1 Mbyte	-
YEAR	VARCHAR	4	-
SUB_MAJOR_ID	INTEGER	-	F.K.
PUBLISHER_ID	INTEGER	-	F.K.

ตารางที่ 7-6 ข้อมูลทั่วไปของวิทยานิพนธ์

LIBRARY_ITEM_CREATOR

COLUMN NAME	TYPE	SIZE	KEY
CREATOR_ID	INTEGER	-	P.K., F.K.
LIBRARY_ITEM_ID	INTEGER	-	P.K., F.K.
LAST_NAME	VARCHAR	30	-

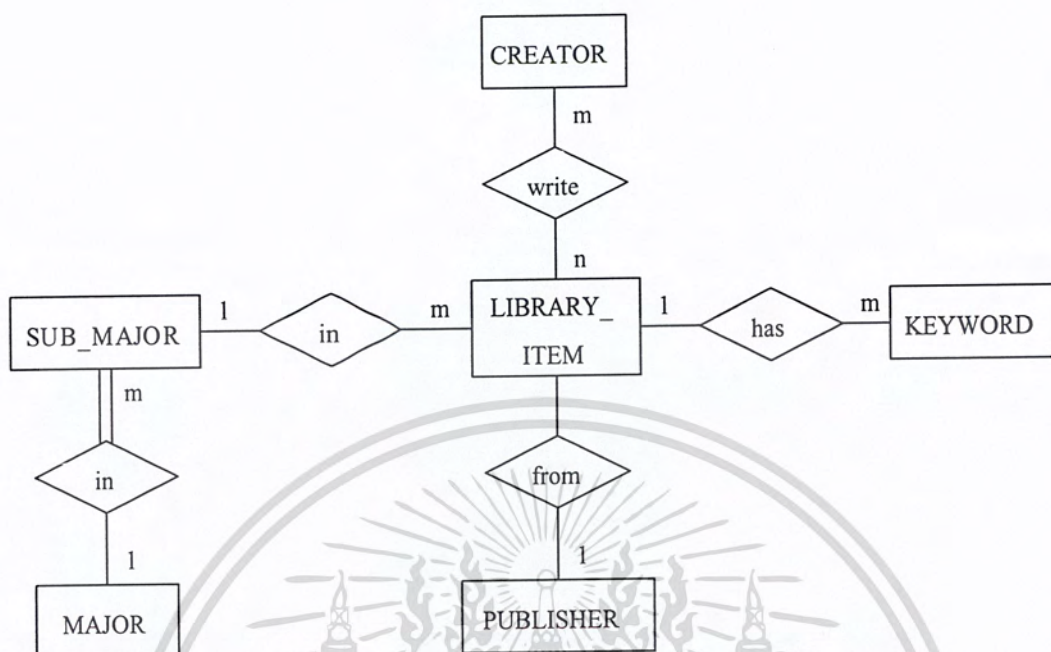
ตารางที่ 7-7 ข้อมูลความสัมพันธ์ระหว่างวิทยานิพนธ์และผู้จัดทำวิทยานิพนธ์

KEYWORDS

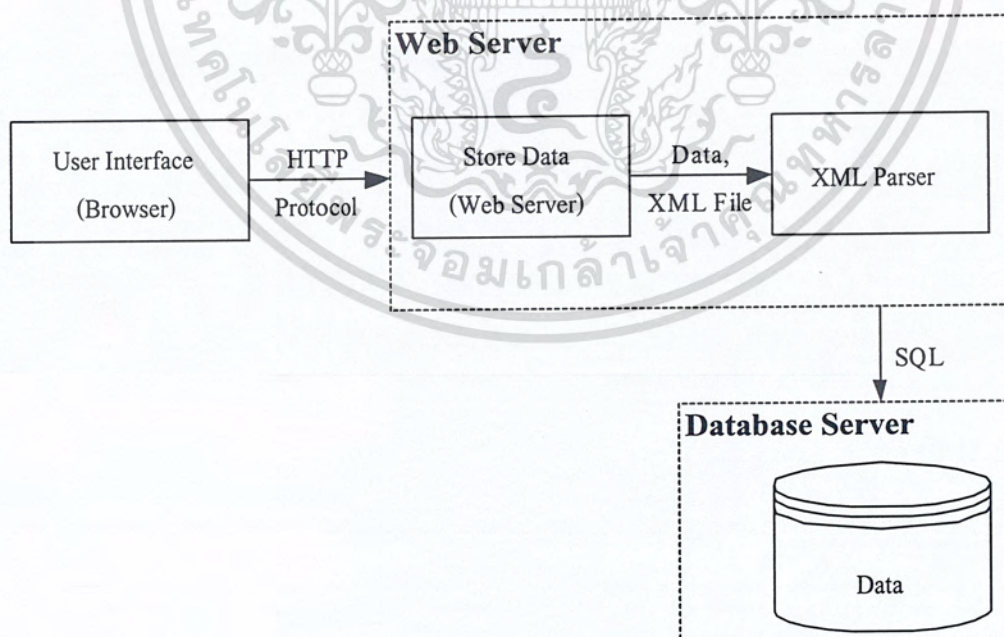
COLUMN NAME	TYPE	SIZE	KEY
KEYWORD	VARCHAR	50	P.K.
LIBRARY_ITEM_ID	INTEGER	-	P.K., F.K.

ตารางที่ 7-8 ข้อมูลคำสำคัญในวิทยานิพนธ์

จาก Database Schema สามารถเขียนความสัมพันธ์ของตาราง โดยใช้ ER Diagram ได้ดังนี้



รูปที่ 7-2 ER Diagram ของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์
การใส่ข้อมูลลงไปในฐานะข้อมูลจะทำได้ดังนี้

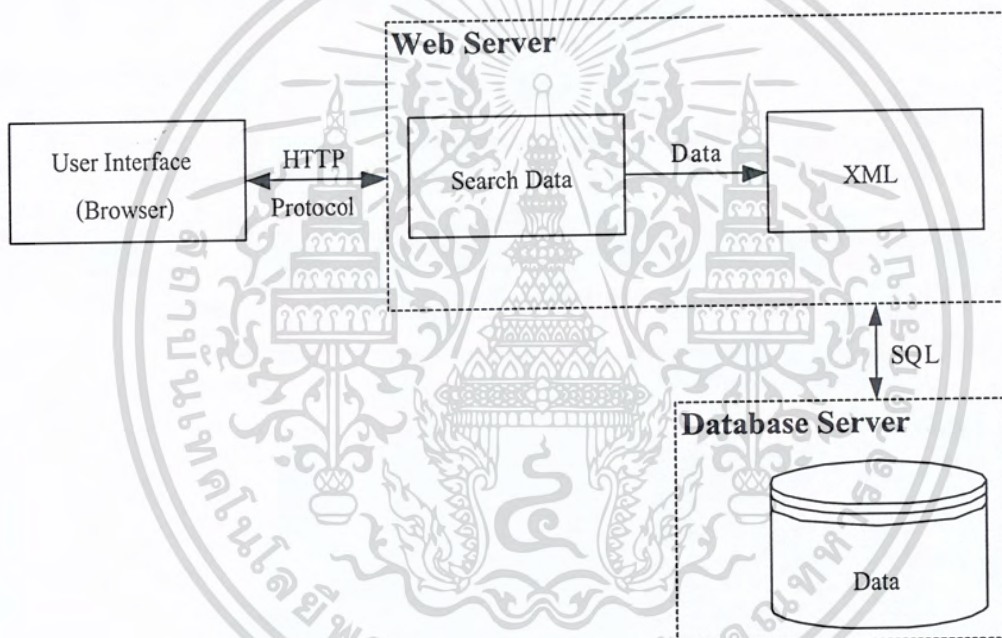


รูปที่ 7-3 แสดงการจัดเก็บข้อมูล XML ในฐานะข้อมูล

จากรูปที่ 7-3 เป็นโมเดลในการจัดเก็บข้อมูลลงในฐานข้อมูล โดยทางยูสเซอร์อินเตอร์เฟซ จะทำการส่งข้อมูลที่เป็น XML มาทางฝั่งเว็บเซิร์ฟเวอร์แล้วเว็บเซิร์ฟเวอร์จะจัดการส่งข้อมูลไปจัดเก็บในดาต้าเบสเซิร์ฟเวอร์ โดยก่อนที่จะจัดเก็บได้นั้นต้องทำการผ่าน XML Parser ก่อน เพื่อเป็นการแยกข้อมูลที่ได้มาลงในตารางที่เหมาะสม

7.4 การนำข้อมูลที่จัดเก็บในดาต้าเบสเซิร์ฟเวอร์มาแสดงผล

ในการนำข้อมูลที่จัดเก็บในดาต้าเบสเซิร์ฟเวอร์มาแสดงผลนั้นจะมีรูปแบบคร่าวๆดังรูปที่ 7-4 นั้นคือยูสเซอร์อินเตอร์เฟซต้องส่งข้อมูลที่และอิลิเมนต์ที่ต้องการค้นหาหรือแสดงผลมายังเว็บเซิร์ฟเวอร์ จากนั้นเว็บเซิร์ฟเวอร์จะทำการค้นหาข้อมูลในดาต้าเบสเซิร์ฟเวอร์ตามค่าอิลิเมนต์ที่ระบุมา และส่งผลลัพธ์ที่หาได้กลับไปยังยูสเซอร์อินเตอร์เฟซ



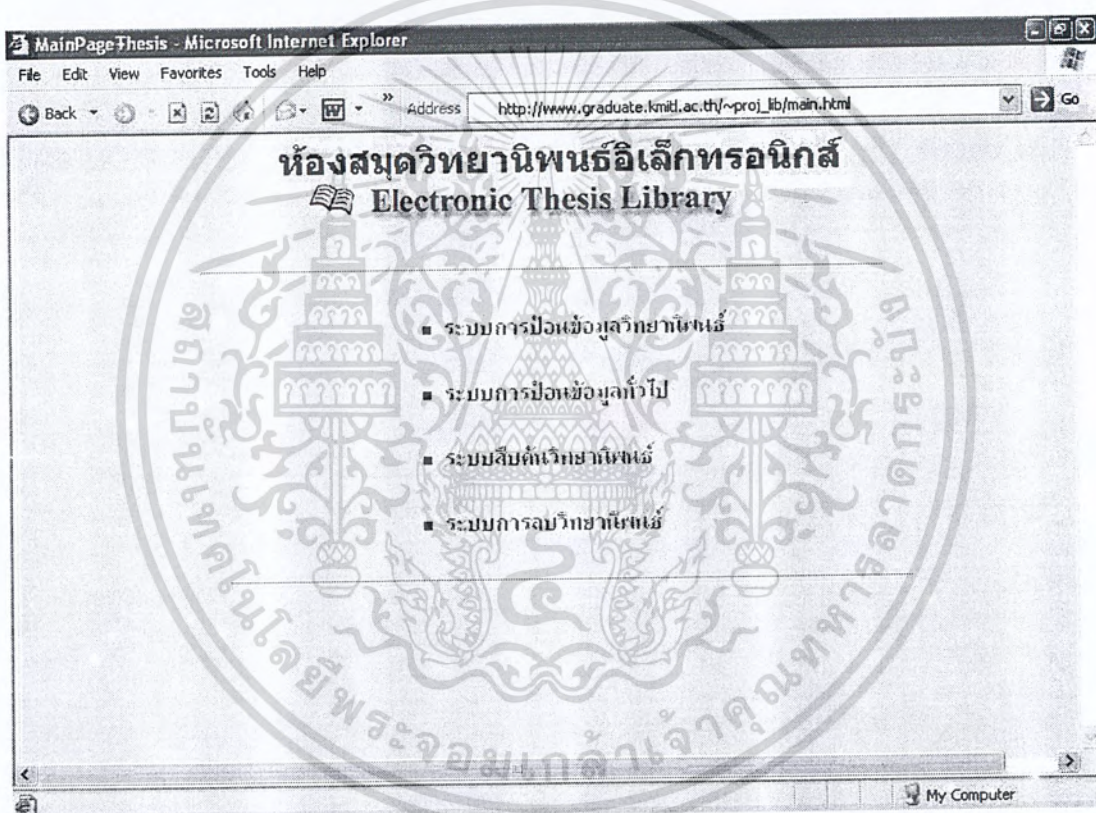
รูปที่ 7-4 แสดงการดึงข้อมูลจากดาต้าเบสเซิร์ฟเวอร์

บทที่ 8

ผลการทดลอง

ในบทนี้จะทำการทดลองการทำงานของระบบ ซึ่งระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ มีการทำงานหลักๆ อยู่ 4 ส่วนด้วยกันคือ การใส่ข้อมูลเข้าไปในระบบ, การค้นหาข้อมูลในระบบ, การนำข้อมูลที่ค้นหาได้มาแสดงผล และการลบข้อมูลวิทยานิพนธ์ ซึ่งการทำงานทั้งหมดจะกระทำผ่านเว็บเบราว์เซอร์ที่ได้เชื่อมต่อกับเครือข่ายอินเทอร์เน็ต

8.1 หน้าเพจหลักของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์



รูปที่ 8-1 แสดงหน้าเพจหลักของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์

ในหน้าเพจหลักจะมีลิงค์ (link) ไปยังการทำงานต่างๆ ที่มีอยู่ในระบบ ซึ่งก็มี การป้อนข้อมูลวิทยานิพนธ์ลงในระบบ, การป้อนข้อมูล, การค้นหาข้อมูลวิทยานิพนธ์, การลบข้อมูลวิทยานิพนธ์

8.2 การป้อนข้อมูลวิทยานิพนธ์ลงในระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์

การป้อนข้อมูลวิทยานิพนธ์ลงในระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ ได้ทำการแบ่งการป้อนข้อมูลออกเป็น 2 ส่วนหลักๆ คือ ส่วนของการป้อนข้อมูลวิทยานิพนธ์ และ ส่วนของการป้อนทั่วไป (ข้อมูลมหาวิทยาลัย, ข้อมูลสาขาวิชา และข้อมูลสาขาวิชาย่อย)

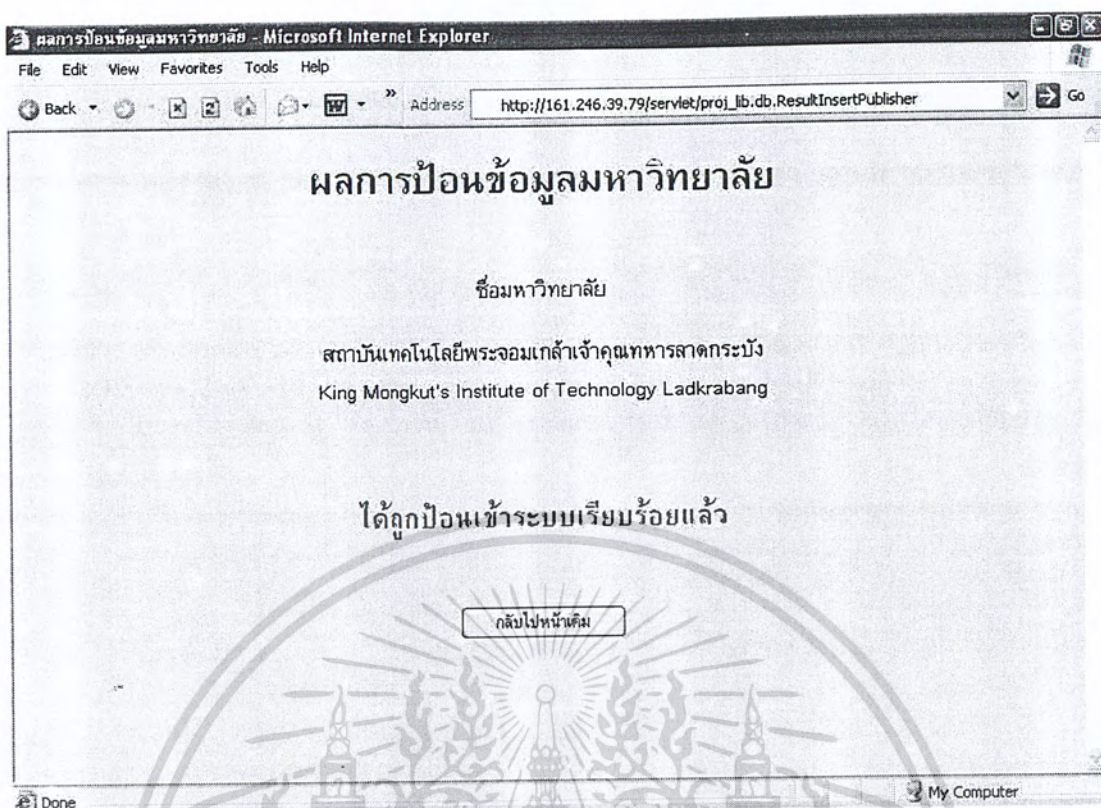
8.2.1 การป้อนข้อมูลทั่วไป

การใส่ข้อมูลในส่วนนี้จะแบ่งออกเป็น 3 ส่วนดังนี้

8.2.1.1 ทดสอบการป้อนข้อมูลมหาวิทยาลัย

รูปที่ 8-2 แสดงหน้าจอการป้อนข้อมูลมหาวิทยาลัย

ในส่วนของการป้อนข้อมูลมหาวิทยาลัยนั้น จะมีช่องรับข้อมูลอยู่ 2 ช่องคือ ชื่อของมหาวิทยาลัยที่เป็นภาษาไทยและชื่อของมหาวิทยาลัยที่เป็นภาษาอังกฤษ และมีรายชื่อของมหาวิทยาลัยที่มีอยู่ในฐานข้อมูลแสดงอยู่ด้วย เมื่อได้ทำการป้อนข้อมูลของมหาวิทยาลัยตามในรูปที่ 8-2 และทำการกดปุ่ม “ป้อน” จะได้ผลลัพธ์ดังรูปที่ 8-3 จากผลลัพธ์ในรูป 8-3 จะเห็นได้ว่าสามารถป้อนข้อมูลมหาวิทยาลัยเข้าไปในระบบได้



รูปที่ 8-3 แสดงผลของการป้อนข้อมูลมหาวิทยาลัย

8.2.1.2 ทดสอบการป้อนข้อมูลสาขาวิชา

ในส่วนของการป้อนข้อมูลสาขาวิชาหลักนั้น จะมีช่องรับข้อมูลและแสดงข้อมูลเหมือนกับในส่วนของการป้อนข้อมูลมหาวิทยาลัย เมื่อได้ทำการทดสอบป้อนข้อมูลของสาขาวิชาตามในรูปที่ 8-4 และทำการกดปุ่ม “ป้อน” จะได้ผลลัพธ์ดังรูปที่ 8-5 จากผลลัพธ์ในรูป 8-5 จะเห็นได้ว่าสามารถป้อนข้อมูลสาขาวิชาเข้าไปในระบบได้

ระบบการป้อนข้อมูล - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://161.246.39.79/servlet/proj_lib.db.InsertAdmin Go

สาขาวิชา

ชื่อภาษาไทย : วิศวกรรมศาสตร์

ชื่อภาษาอังกฤษ : Engineering ป้อน

-----สาขาวิชาที่มีในฐานข้อมูล-----

ครุศาสตร์อุตสาหกรรม
วิทยาศาสตร์
สถาปัตยกรรมศาสตร์

สาขาวิชาย่อย

เลือกสาขาวิชาของสาขาวิชาย่อย

ชื่อภาษาไทย :
ชื่อภาษาอังกฤษ : ป้อน

Done Internet

รูปที่ 8-4 แสดงหน้าจอการป้อนข้อมูลสาขาวิชา

ผลการป้อนข้อมูลสาขาวิชา - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://161.246.39.79/servlet/proj_lib.db.ResultInsertMajor Go

ผลการป้อนข้อมูลสาขาวิชา

ชื่อสาขาวิชา
วิศวกรรมศาสตร์
Engineering

ได้ถูกป้อนเข้าระบบเรียบร้อยแล้ว

กลับไปหน้าเดิม

Done My Computer

รูปที่ 8-5 แสดงผลของการป้อนข้อมูลสาขาวิชา

8.2.1.3 ทดสอบการป้อนข้อมูลสาขาวิชาย่อย

ระบบการป้อนข้อมูล - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Address http://161.246.39.79/servlet/proj_lib.db.InsertAdmin Go

ชื่อภาษาอังกฤษ : ป้อน

สาขาวิชาที่มีในฐานข้อมูล

ครุศาสตร์อุตสาหกรรม
วิทยาศาสตร์
สถาปัตยกรรมศาสตร์

สาขาวิชา

วิศวกรรมศาสตร์

ชื่อภาษาไทย : วิศวกรรมคอมพิวเตอร์

ชื่อภาษาอังกฤษ : Computer Engineering ป้อน

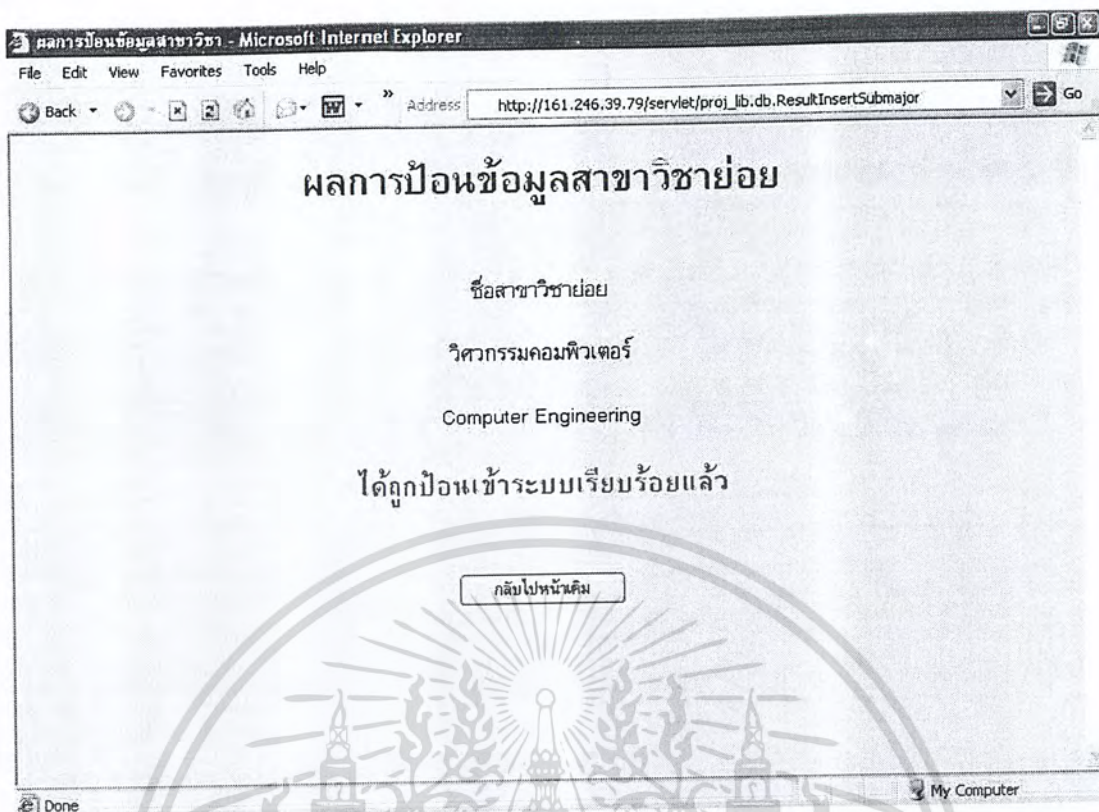
สาขาวิชาย่อยที่มีในฐานข้อมูล

วิศวกรรมไฟฟ้า
วิศวกรรมโทรคมนาคม
วิศวกรรมอิเล็กทรอนิกส์
วิศวกรรมโยธา

Done Internet

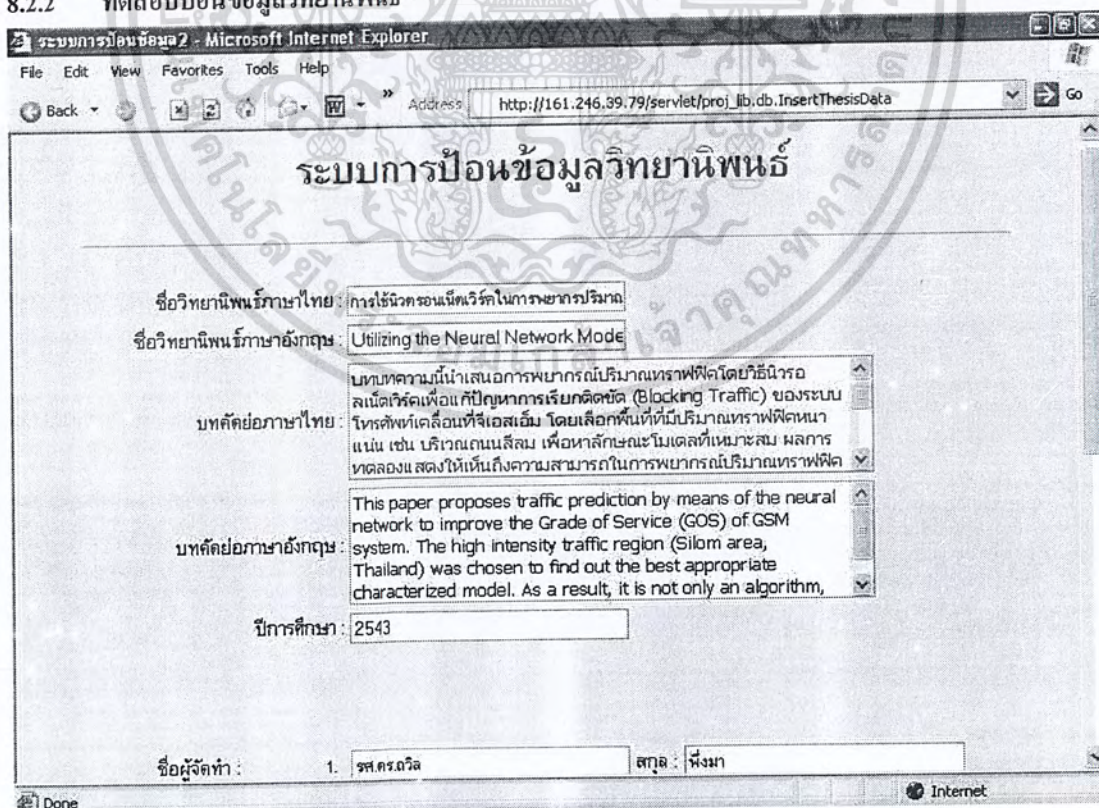
รูปที่ 8-6 แสดงหน้าจอการป้อนข้อมูลสาขาวิชาย่อย

ในส่วนของการป้อนข้อมูลสาขาวิชาย่อยนั้นมีความแตกต่างกับการป้อนข้อมูลสาขาวิชาคือจะต้องทำการเลือกข้อมูลจาก Drop List ว่าต้องการที่จะใส่ข้อมูลสาขาวิชาย่อยไว้ในสาขาวิชาหลักอะไร โดย Drop List จะทำการดึงข้อมูลสาขาวิชาหลักที่มีอยู่ในฐานข้อมูลมาแสดงให้เลือก เมื่อทดสอบใส่ข้อมูลของสาขาวิชาย่อยตามรูปที่ 8-6 แล้วทำการเลือกสาขาวิชาหลัก ถ้าสาขาวิชาย่อยที่ต้องการป้อนเข้าสู่ระบบไม่มีสาขาวิชาหลักที่ต้องการ จะต้องทำการป้อนข้อมูลของสาขาวิชาหลักเสียก่อน หลังจากนั้นทำการกดปุ่ม “ป้อน” หน้าเพจนี้จะทำการส่งข้อมูลไปประมวลผลแล้วได้ผลลัพธ์ดังรูปที่ 8-7 จากรูปจะเห็นได้ว่าสามารถที่จะป้อนข้อมูลสาขาวิชาย่อยเข้าไปในระบบได้อย่างถูกต้อง



รูปที่ 8-7 แสดงผลของการป้อนข้อมูลสาขาวิชาย่อย

8.2.2 ทดสอบป้อนข้อมูลวิทยานิพนธ์



รูปที่ 8-8 แสดงการป้อนข้อมูลวิทยานิพนธ์

ในรูปที่ 8-8 จะเป็นการทดสอบป้อนข้อมูลที่เป็นบทความของวิทยานิพนธ์ ซึ่งเมื่อทำการป้อนข้อมูลทั้งหมดครบถ้วน แล้วทำการกดปุ่ม “ป้อน” ที่อยู่ส่วนท้ายของเพจ ในส่วนของการป้อนข้อมูลวิทยานิพนธ์จะมีด้วยกันสองส่วนคือ ส่วนที่เป็นข้อมูลและส่วนที่เก็บวิทยานิพนธ์ที่อยู่ในรูปไฟล์นามสกุล pdf ดังรูปที่ 8-9

ป้อนไฟล์ PDF ของวิทยานิพนธ์ - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Address http://161.246.39.79/servlet/proj_lib.db.InsertPDF Go

ระบบการป้อนข้อมูลวิทยานิพนธ์

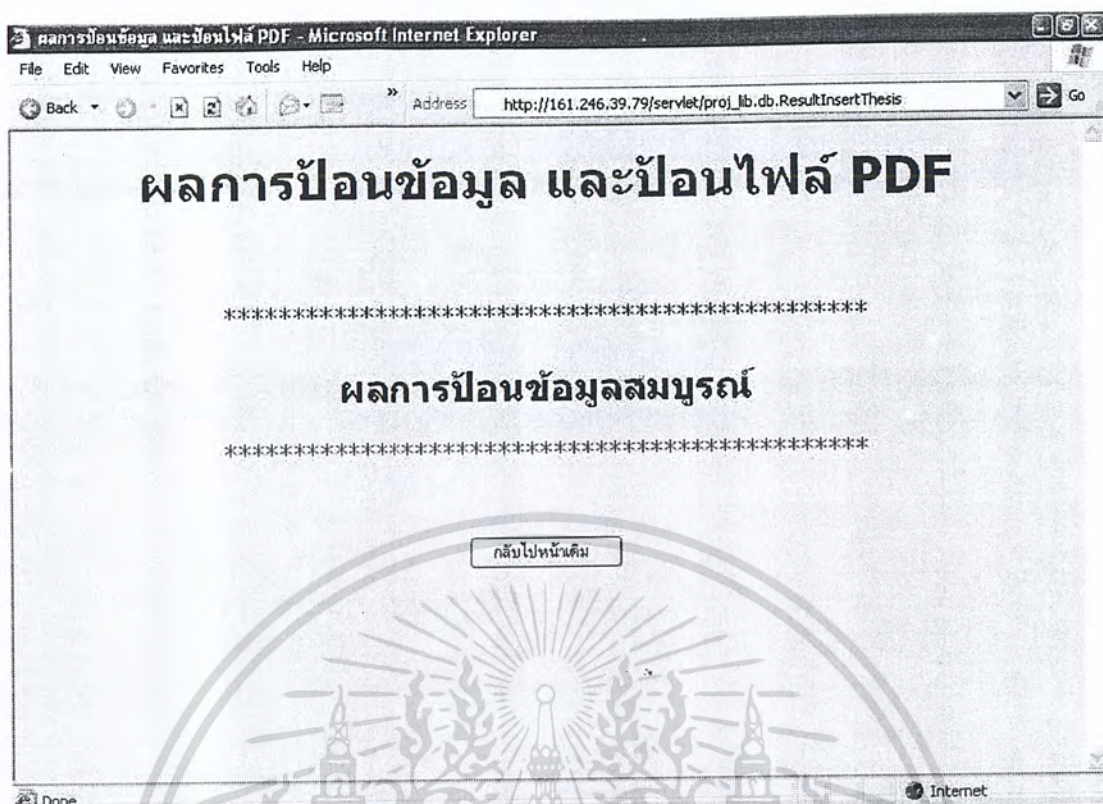
ป้อนไฟล์ PDF ของวิทยานิพนธ์

PDF :

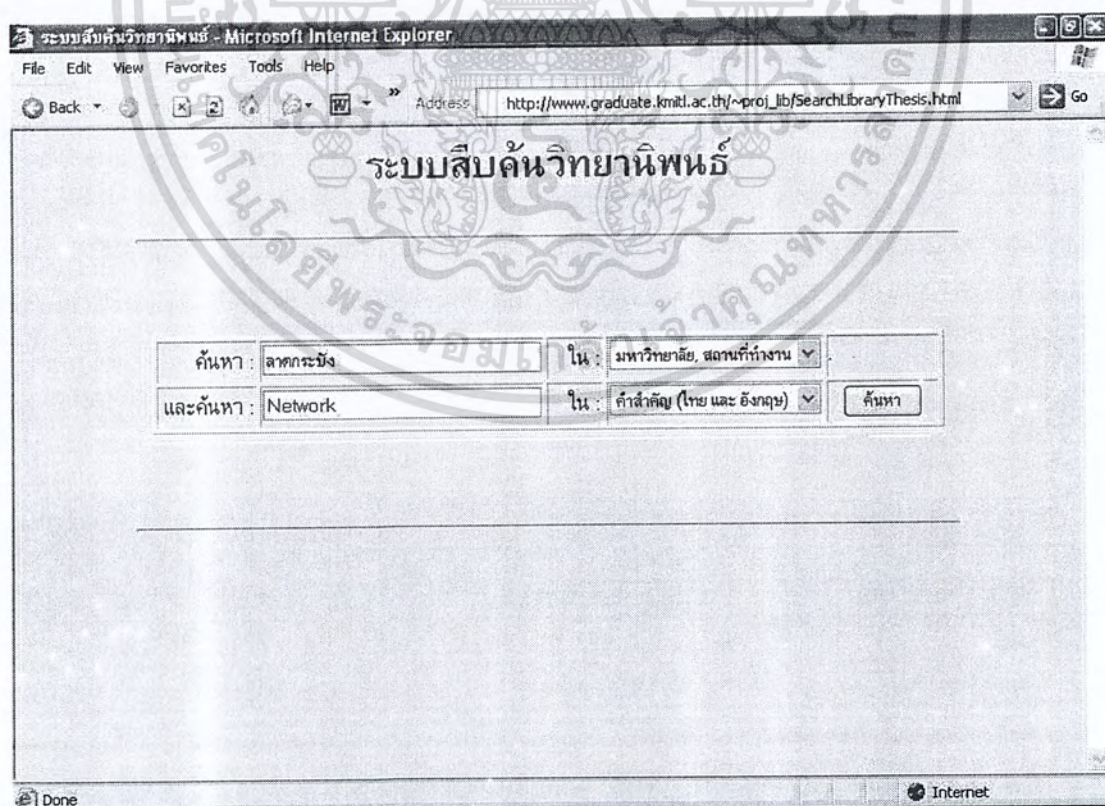
Done My Computer

รูปที่ 8-9 แสดงการป้อนข้อมูลวิทยานิพนธ์ที่อยู่ในรูปไฟล์ pdf

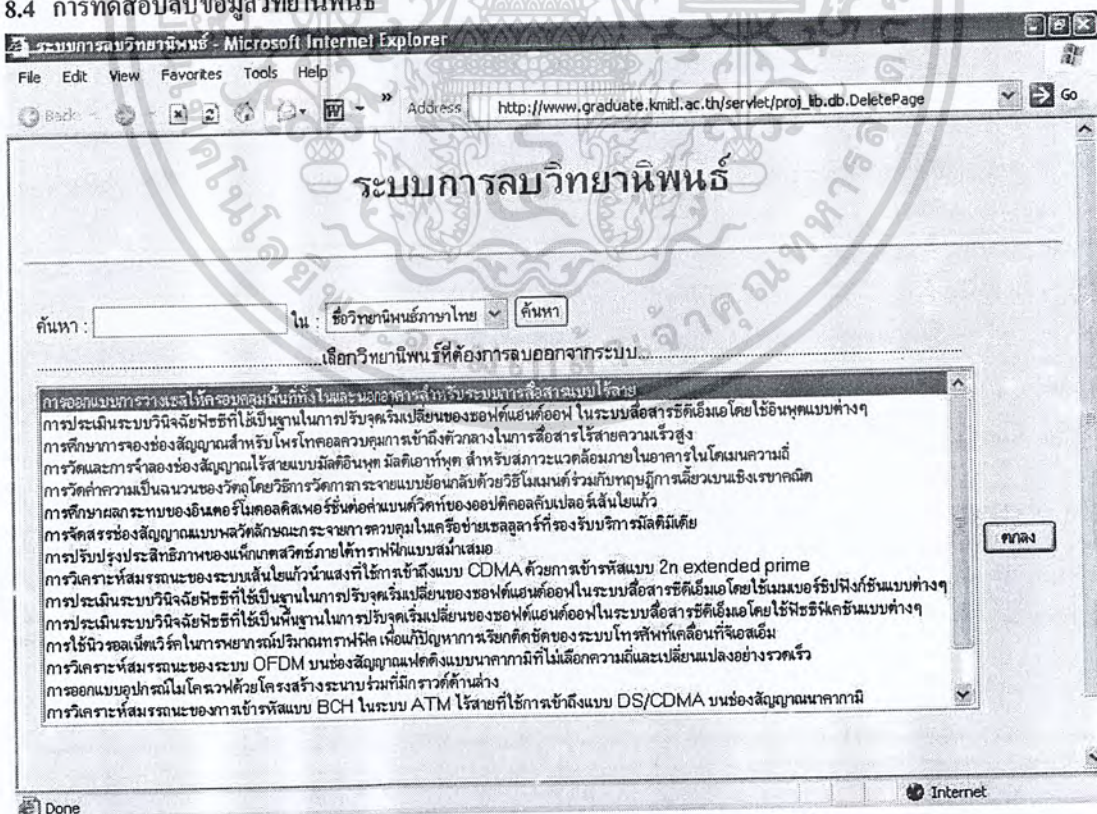
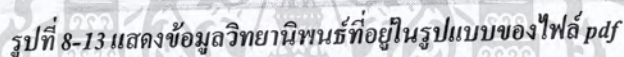
ในรูปที่ 8-9 จะเป็นการอัปโหลด (upload) ไฟล์จากเครื่องของผู้ใช้ไปยังเซิร์ฟเวอร์ เมื่อคลิกปุ่ม “Submit” ระบบก็จะทำการใส่ข้อมูลต่างๆ ที่ได้ป้อนมา ใส่เข้าไปอยู่ในตารางที่เหมาะสม ซึ่งผลของการทำงานจะอยู่ในรูปที่ 8-10 จากรูปแสดงให้เห็นว่า ระบบสามารถที่จะใส่ข้อมูลวิทยานิพนธ์ได้



รูปที่ 8-10 แสดงผลการป้อนข้อมูลวิทยานิพนธ์

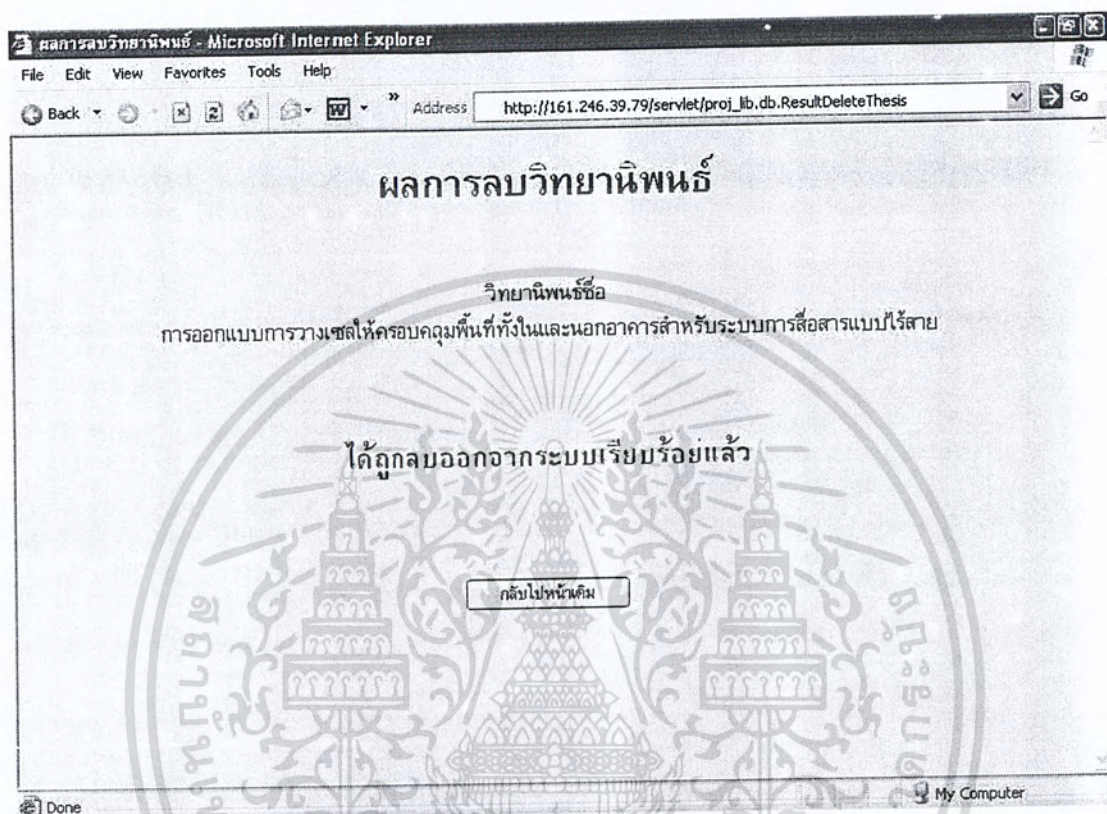


รูปที่ 8-11 แสดงหน้าเพจการค้นหาข้อมูลวิทยานิพนธ์



รูปที่ 8-14 แสดงหน้าเพจการลบข้อมูลวิทยานิพนธ์

จากรูปที่ 8-14 จะแสดงหน้าจอของการลบข้อมูลวิทยานิพนธ์ จากรูปจะเห็นว่า มี List ของข้อมูลที่เป็นชื่อของวิทยานิพนธ์ทั้งหมดที่มีอยู่ โดยถ้าต้องการลบวิทยานิพนธ์ สามารถทำได้โดยเลือกข้อมูลที่อยู่ใน List แล้วกดปุ่ม “ตกลง” แต่ถ้าเราทราบข้อมูลของวิทยานิพนธ์ที่เราต้องการลบก็สามารถทำการค้นหาข้อมูลของวิทยานิพนธ์จากช่องค้นหา ก่อน แล้วค่อยทำการลบวิทยานิพนธ์



รูปที่ 8-15 แสดงผลการลบข้อมูลวิทยานิพนธ์

จากรูปที่ 8-15 แสดงให้เห็นว่าระบบสามารถทำการลบข้อมูลวิทยานิพนธ์ที่ต้องการออกจากฐานข้อมูลได้

บทที่ 9

บทวิจารณ์ และสรุป

9.1 สรุปผลการดำเนินงาน

โครงการนี้เสนอวิธีการออกแบบ และพัฒนาเว็บไซต์ สำหรับเก็บรวบรวม และเป็นระบบสืบค้น บทความวิทยานิพนธ์ในรูปแบบอิเล็กทรอนิกส์ สามารถทำการค้นหาวิทยานิพนธ์ที่ต้องการโดยอาศัยคำสำคัญต่าง ๆ จากบทความของวิทยานิพนธ์นั้น ๆ ทำให้ผู้ใช้สามารถใช้งานระบบผ่านทางเว็บเบราว์เซอร์ได้ สำหรับผู้ใช้ที่เป็นผู้ปฏิบัติการ สามารถใช้ระบบนี้จัดเก็บบทความวิทยานิพนธ์ สำหรับผู้ใช้นักศึกษาสามารถใช้ระบบนี้สืบค้นบทความวิทยานิพนธ์ได้

บทความวิทยานิพนธ์ในรูปแบบอิเล็กทรอนิกส์ มีการจัดเก็บในรูปแบบเอกสาร XML ตามข้อกำหนดที่กำหนดโดยมาตรฐานฉบับลีนเมตาดาต้า ฉบับ 1.1 ภาษาไทย ข้อมูลบทความวิทยานิพนธ์ในรูปแบบ XML ได้จัดเก็บในฐานข้อมูลแบบ Relational เพื่อประหยัดเนื้อที่ และอำนวยความสะดวกในเรื่องการสืบค้น สำหรับโปรแกรมที่จัดการระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์พัฒนาโดยอาศัยเทคโนโลยีของจาวา คือจาวาเซิร์ฟเล็ต (Java Servlet)

9.2 บทวิจารณ์

หลังจากได้ลงมือทำและทดสอบโครงการวิจัยนี้แล้ว ผู้วิจัยได้พบว่าการทำงานของระบบห้องสมุดวิทยานิพนธ์อิเล็กทรอนิกส์ยังมีข้อจำกัดและข้อบกพร่องที่ควรแก้ไขอยู่หลายประการด้วยกันคือ

1. เซิร์ฟเวอร์ที่ใช้งานไม่มีเสถียรภาพเพียงพอ เนื่องจากปัญหาทางด้านเครือข่าย ทำให้เสียเวลากับการแก้ไขปัญหา
2. การแสดงผลข้อมูลวิทยานิพนธ์ที่เป็นไฟล์ PDF จะเสียเวลาในการโหลดข้อมูลนาน
3. การค้นหาข้อมูลวิทยานิพนธ์ สามารถค้นหาได้เฉพาะในส่วนของบทความเท่านั้น
4. การ upload ข้อมูลที่เป็นไฟล์ PDF สามารถ upload ข้อมูลที่ไม่เกิน 1 MB เท่านั้นซึ่งไม่เพียงพอสำหรับข้อมูลวิทยานิพนธ์ฉบับเต็ม
5. ระบบยังไม่สามารถใช้งานได้จริงเนื่องจากยังไม่ได้มีการแยกการทำงานกันระหว่าง ผู้ดูแลระบบ และผู้ใช้ระบบทั่วไป

9.3 แนวทางการพัฒนาต่อไป

1. การเก็บข้อมูลวิทยานิพนธ์ในโครงการนี้ได้เก็บข้อมูลของวิทยานิพนธ์เฉพาะส่วนของบทคัดย่อเท่านั้น เพื่อเป็นการเพิ่มประสิทธิภาพในการค้นหาข้อมูลควรจะขยายขอบเขตของการเก็บข้อมูลไปเป็นจัดเก็บข้อมูลวิทยานิพนธ์ทั้งวิทยานิพนธ์ (full text)
2. พัฒนาวีธีการแสดงผลข้อมูลในวิทยานิพนธ์ เนื่องจากในโครงการนี้ได้ใช้ไฟล์ pdf ในการเก็บข้อมูลวิทยานิพนธ์เมื่อนำไปแสดงผลผ่านเว็บเบราว์เซอร์ทำให้ต้องเสียเวลาในการโหลดข้อมูลมาก ถ้าได้พัฒนาการจัดเก็บข้อมูลวิทยานิพนธ์ทั้งวิทยานิพนธ์ (Full Text) แล้วก็ควรเปลี่ยนวิธีการแสดงผลข้อมูลในวิทยานิพนธ์โดยใช้ XSL มาช่วยในการแสดงผล
3. พัฒนาเพิ่มในส่วนของการแลกเปลี่ยนข้อมูลวิทยานิพนธ์ในรูปแบบของเอกสาร XML กับระบบอื่น เช่น สามารถแปลงเอกสาร Microsoft Word เป็นเอกสาร XML ที่ใช้กับระบบนี้ได้ทันที
4. พัฒนาฟังก์ชันการทำงานของระบบให้สมบูรณ์และครบถ้วนยิ่งขึ้น เช่น การใช้งานในหน้าต่างๆ ของเว็บเบราว์เซอร์ ผู้ใช้ในส่วนการปฏิบัติงานจะสามารถใช้ในส่วนการเพิ่มและลบบทคัดย่อวิทยานิพนธ์ออกจากระบบ และผู้ใช้ทั่วไปสามารถที่จะค้นหาบทคัดย่อวิทยานิพนธ์ได้



ภาคผนวก ก

การติดตั้ง DB2 Universal Database Enterprise Edition

1. การเตรียมการติดตั้ง

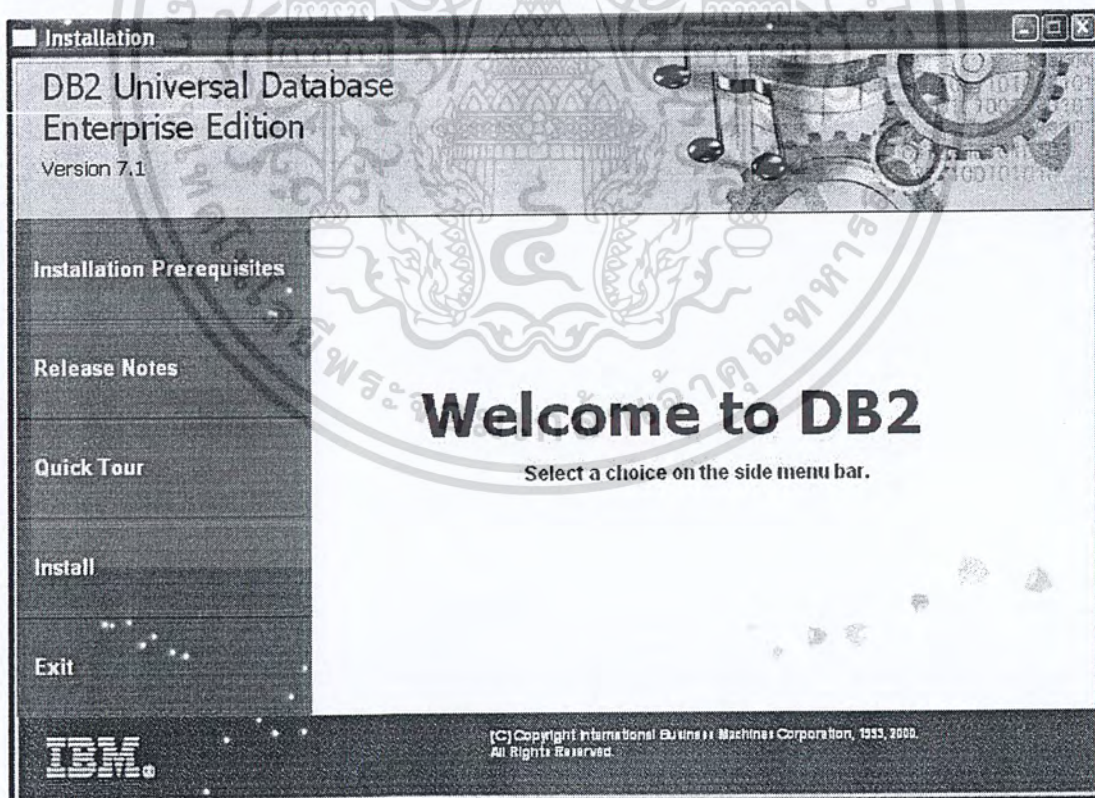
ในการติดตั้งบน Windows NT, Windows 2000 หรือ Windows XP ก่อนอื่นผู้ที่ทำการติดตั้งจะต้องอยู่ใน Group ของ Administrators ก่อน ซึ่งมีความสามารถในการติดตั้ง Database Server โดยระบบมีความต้องการดังนี้

- Windows NT Version 4.0 with Service Pack 4 or later, Windows 2000, Windows XP, Windows 9x
- ขนาดของหน่วยความจำ 128 เมกะไบต์
- พื้นที่ของฮาร์ดดิสก์ 420 เมกะไบต์ (สำหรับการติดตั้งแบบปกติ) รวม JRE
- การเชื่อมต่อแบบ NetBIOS, IPX/SPX, Named Pipes, and TCP/IP

2. ขั้นตอนการติดตั้ง

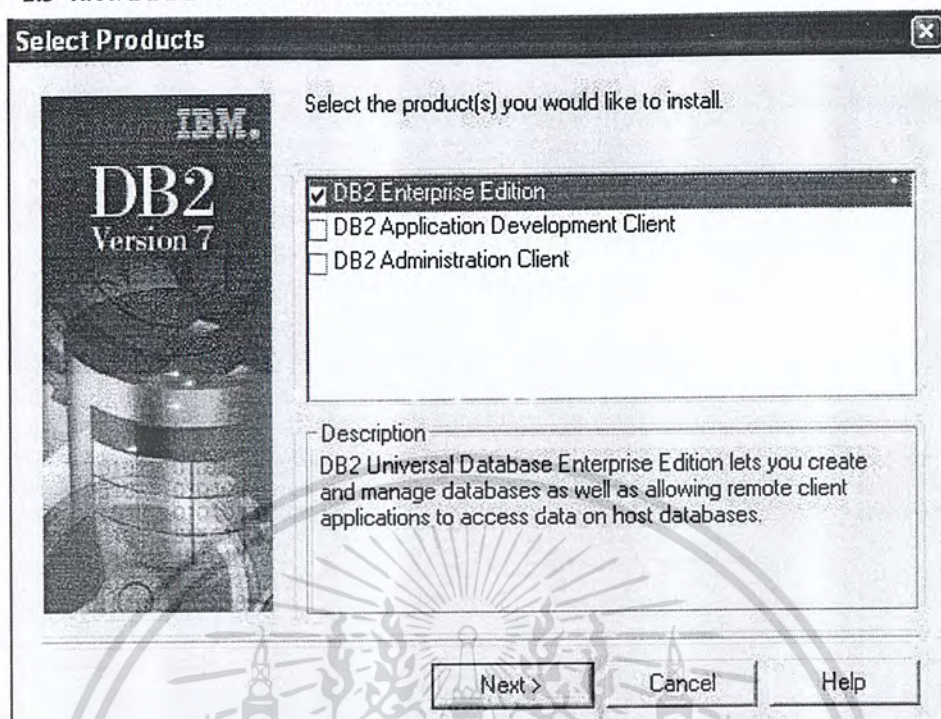
2.1 ผู้ติดตั้งต้องอยู่ในกลุ่มของ Administration Group

2.2 Run setup.exe เพื่อทำการติดตั้ง



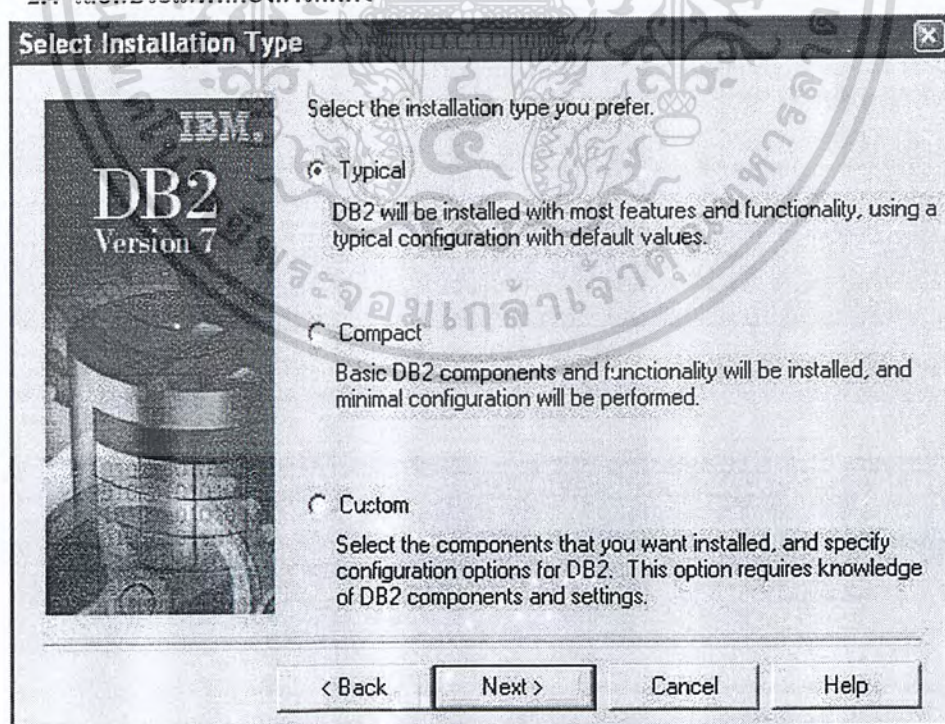
รูปที่ ก-1 แสดงการติดตั้ง DB2 Universal Database Enterprise Edition Version 7.1

2.3 เลือก DB2 Products ที่ต้องการติดตั้ง



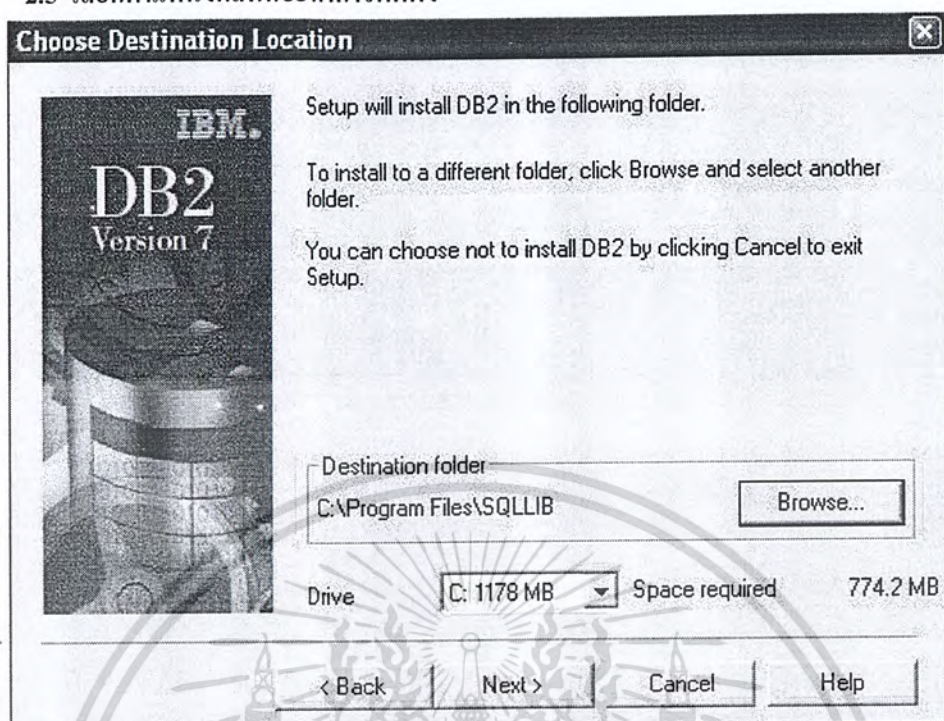
รูปที่ ก-2 แสดงการเลือก DB2 Products ที่ต้องการติดตั้ง

2.4 เลือกประเภทที่ต้องการติดตั้ง



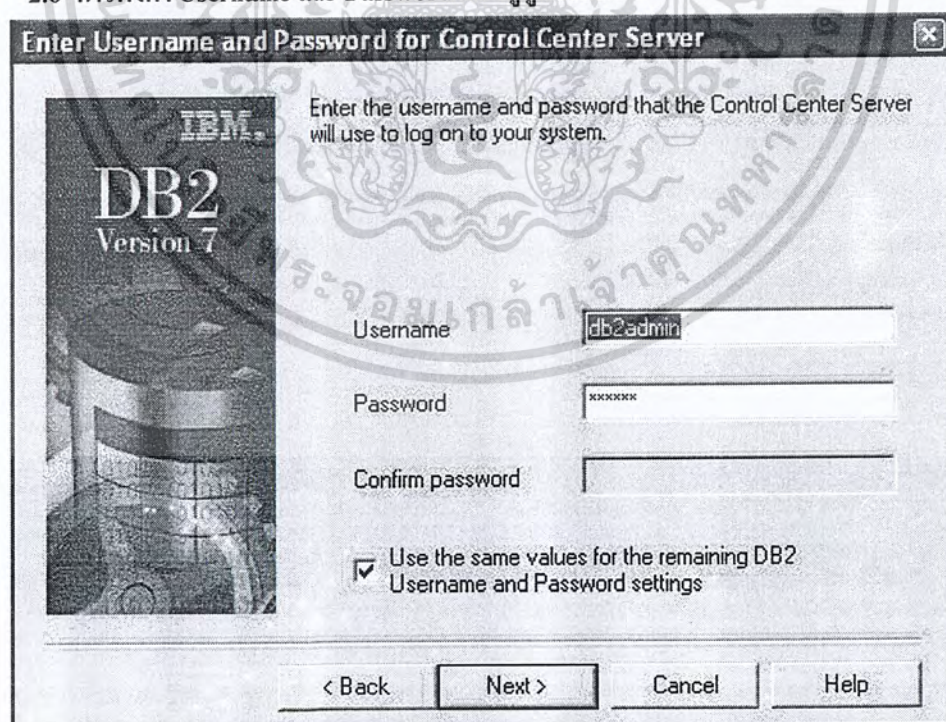
รูปที่ ก-3 แสดงการเลือกประเภทที่ต้องการติดตั้ง

2.5 เลือกตำแหน่งดิสก์ที่จะทำการติดตั้ง



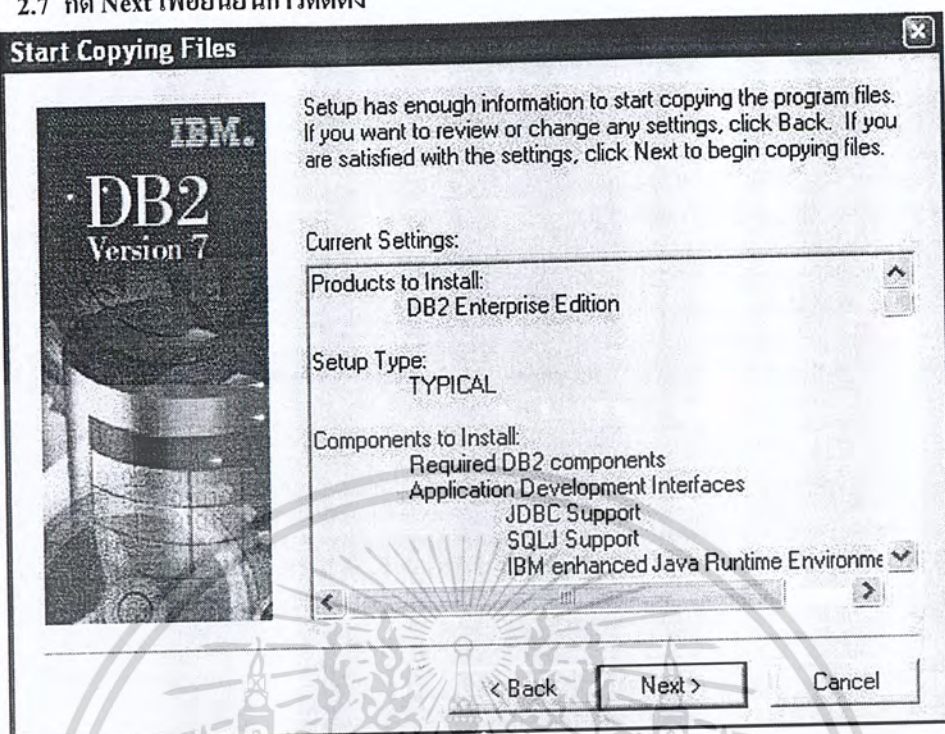
รูปที่ ก-4 แสดงการเลือกตำแหน่งดิสก์ที่จะทำการติดตั้ง

2.6 การตั้งค่า Username และ Password ของผู้ดูแลระบบ



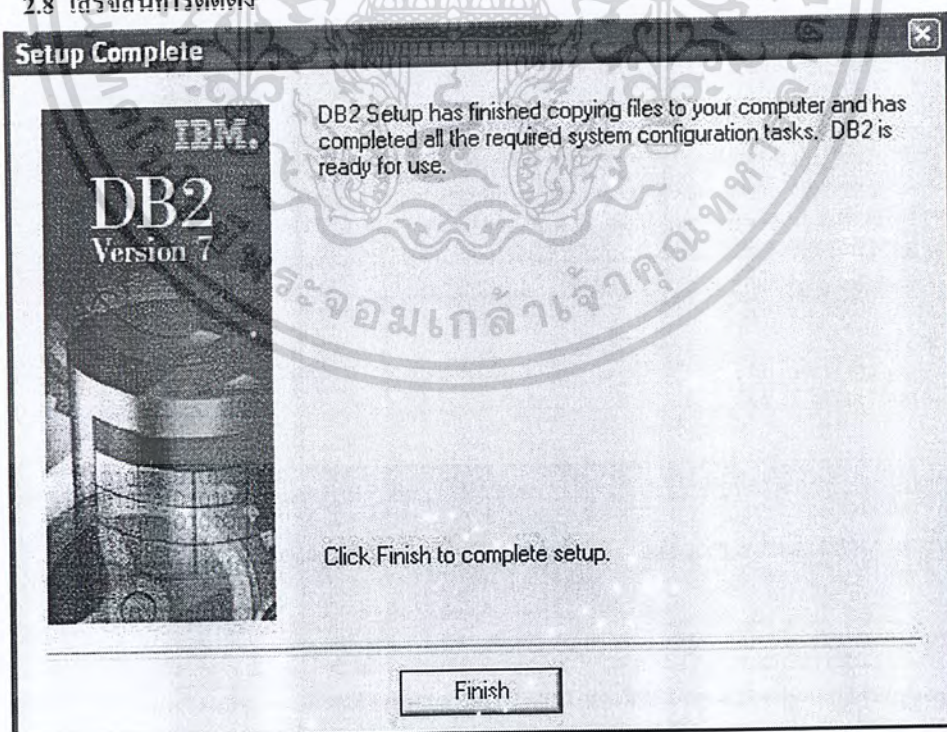
รูปที่ ก-5 แสดงการตั้งค่า Username และ Password ของผู้ดูแลระบบ

2.7 กด Next เพื่อยืนยันการติดตั้ง



รูปที่ ก-6 แสดงการยืนยันการติดตั้ง

2.8 เสร็จสิ้นการติดตั้ง



รูปที่ ก-7 แสดงการเสร็จสิ้นการติดตั้ง

ภาคผนวก ข

การตั้งค่าการเชื่อมต่อกับ Database Server ที่เป็น DB2

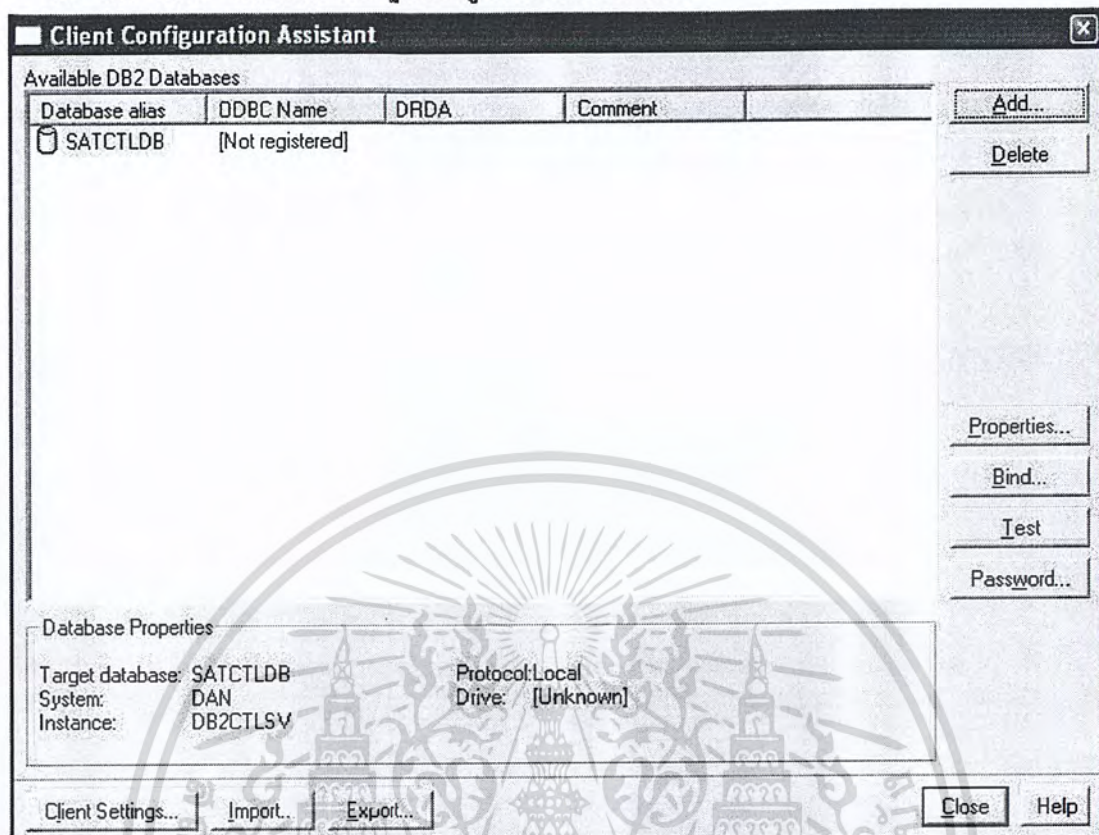
การติดตั้ง DB2 Universal Database Enterprise Edition ก็เพื่อให้เราสามารถติดต่อกับ Database Server เพื่อที่จะเข้าไปจัดการกับฐานข้อมูลเช่น สร้างตาราง (Table), สร้างฐานข้อมูล (Database), ใส่ข้อมูล, หรือว่าดูข้อมูลที่อยู่ในฐานข้อมูลได้สะดวกขึ้น แต่ก่อนที่เราจะติดต่อกับฐานข้อมูลได้นั้นเราต้องทำการตั้งค่าการเชื่อมต่อก่อน ซึ่งมีวิธีในการตั้งค่าการเชื่อมต่อกับ Database Server ดังนี้

1. เปิดโปรแกรม “Client Configuration Assistant”



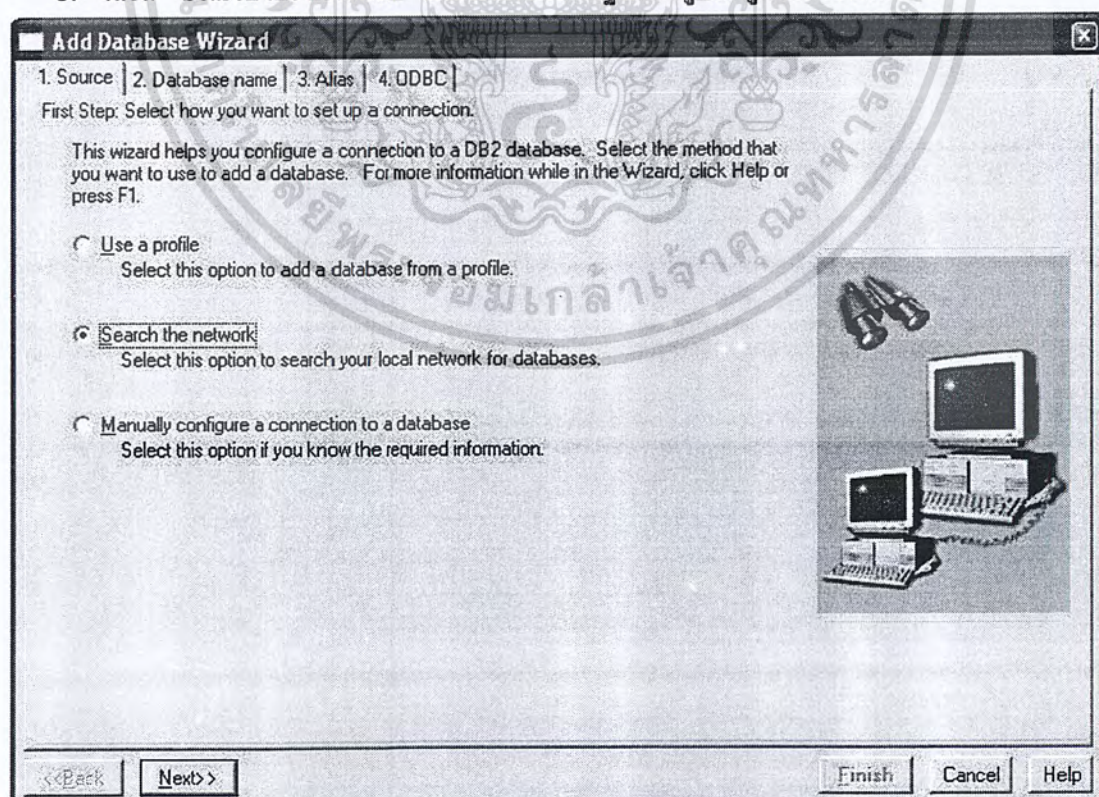
รูปที่ ข-1 แสดงการเข้าสู่โปรแกรม Client Configuration Assistant

2. กด “Add” เพื่อทำการเพิ่มฐานข้อมูลที่ต้องการจะติดต่อ



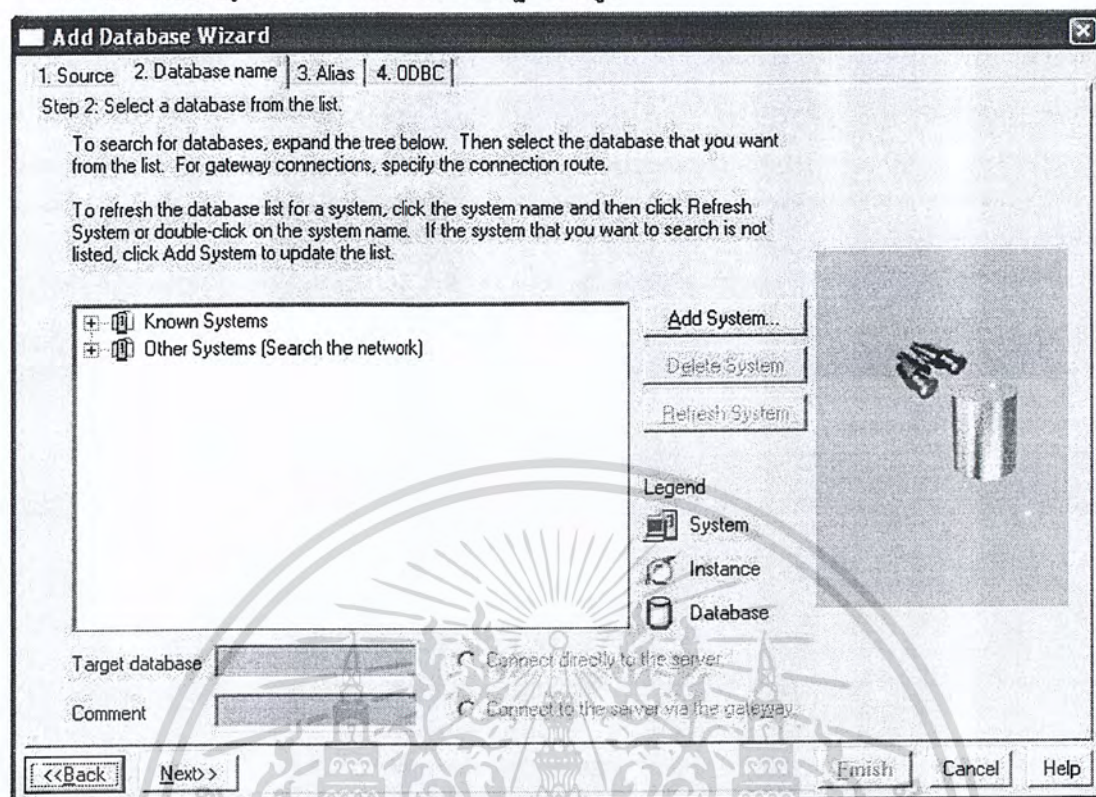
รูปที่ ข-2 แสดงโปรแกรม Client Configuration Assistant

3. เลือก “Search the network” เพื่อทำการค้นหาฐานข้อมูลที่อยู่ภายใน Network



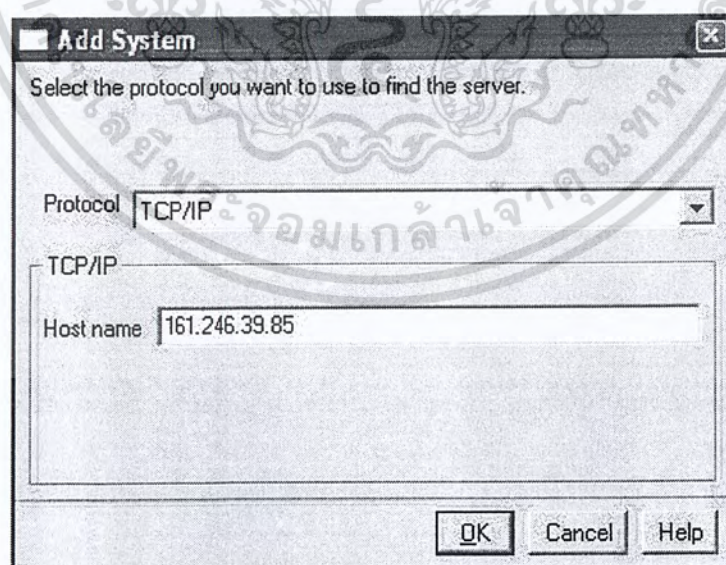
รูปที่ ข-3 แสดงวิธีการตั้งค่าการเชื่อมต่อ

4. กด “Add System...” เพื่อทำการเพิ่มฐานข้อมูล



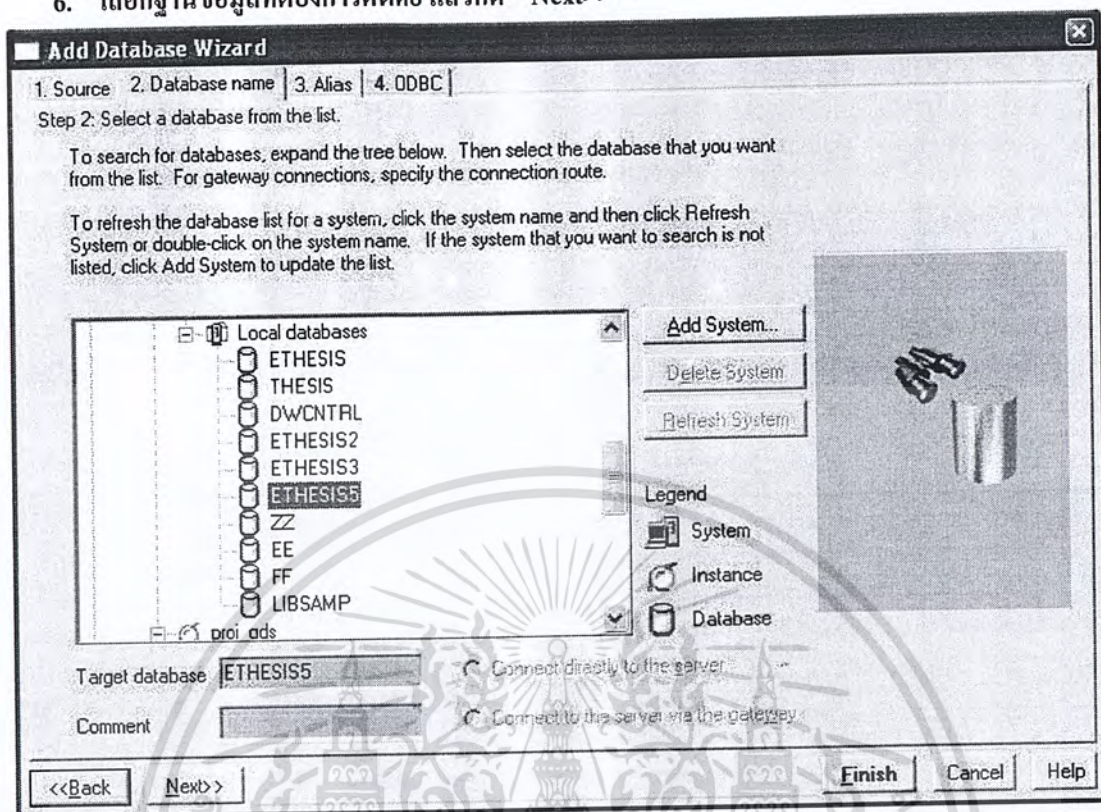
รูปที่ ข-4 แสดงการเลือกฐานข้อมูล

5. เลือกโปรโตคอลที่ใช้ติดต่อ และใส่ชื่อโฮสต์เนมหรือหมายเลข IP ของเซิร์ฟเวอร์ที่ต้องการติดต่อ



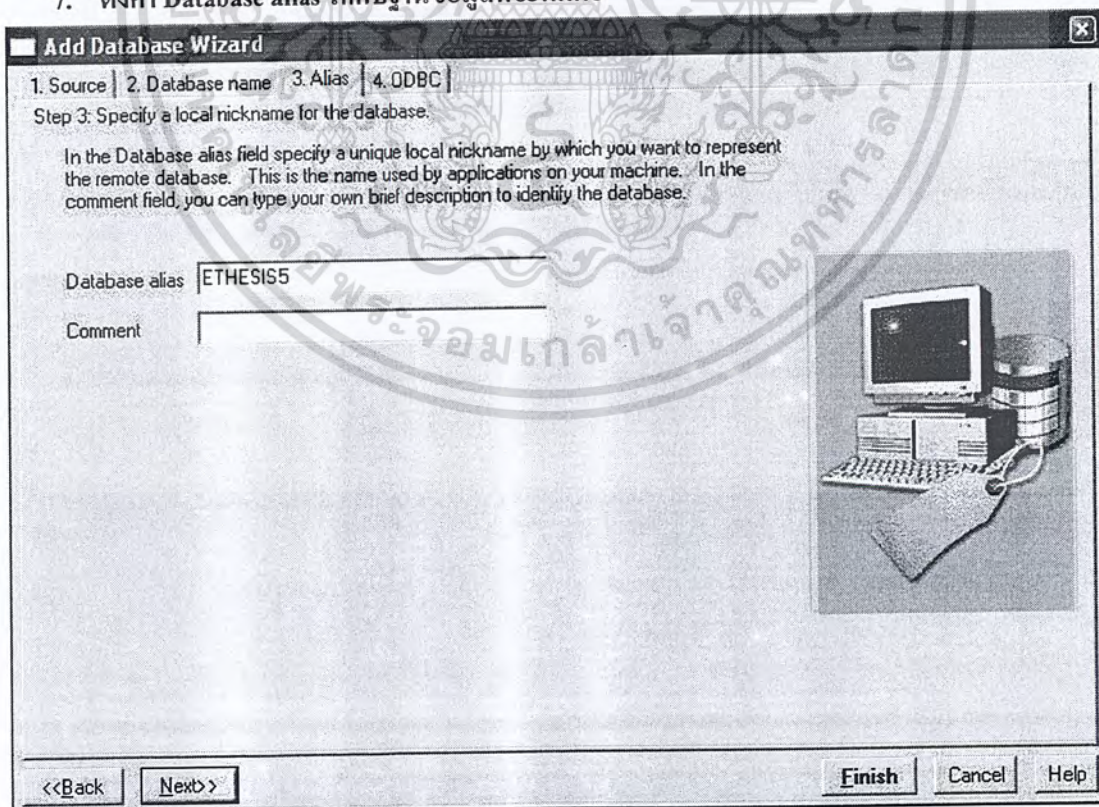
รูปที่ ข-5 แสดงการ Add System

6. เลือกฐานข้อมูลที่ต้องการติดต่อ แล้วกด “Next>>”



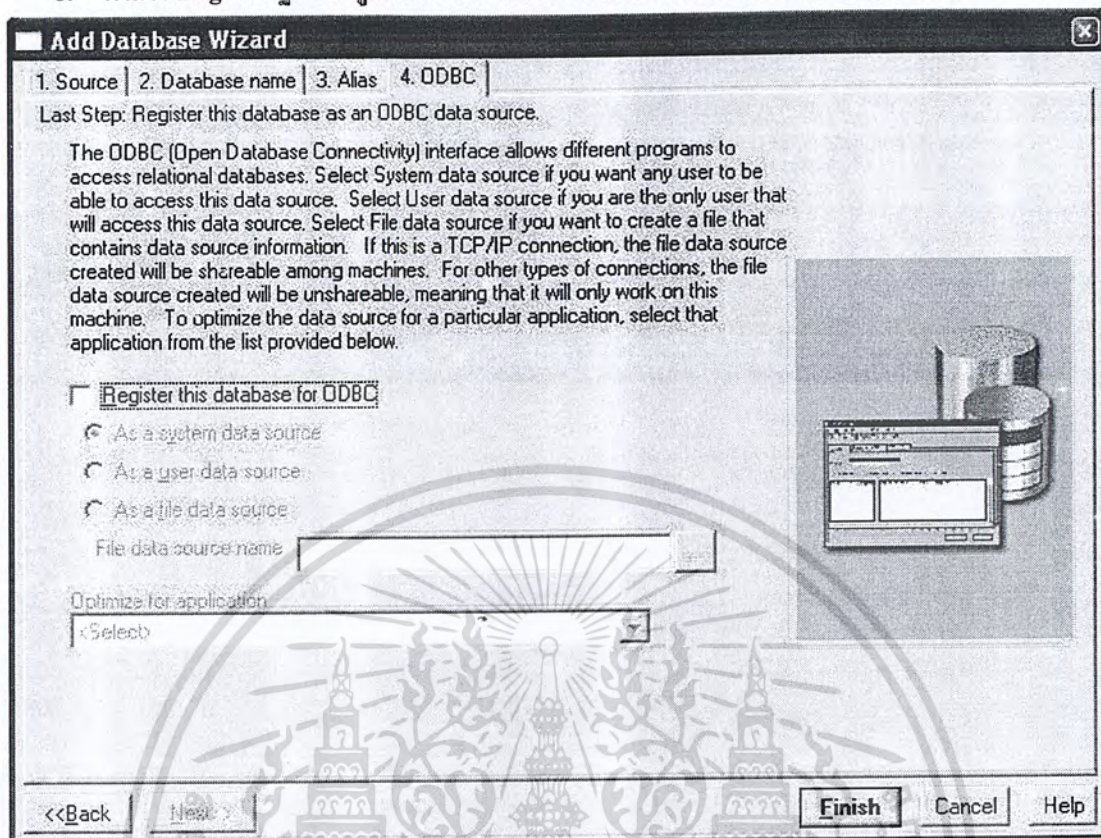
รูปที่ ข-6 แสดงการเลือกฐานข้อมูล

7. ตั้งค่า Database alias ให้กับฐานข้อมูลที่จะติดต่อ



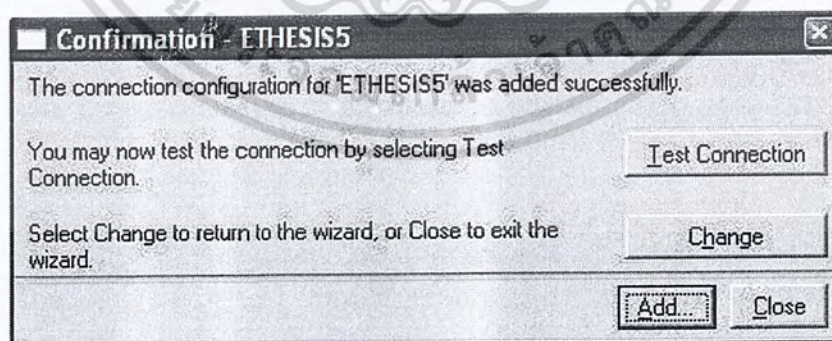
รูปที่ ข-7 แสดงการตั้งค่า Database alias ให้กับฐานข้อมูลที่จะติดต่อ

8. ทำการ Register ฐานข้อมูลกับ ODBC ในที่นี้ไม่ได้ใช้ ODBC จึงไม่ต้องทำการ Register



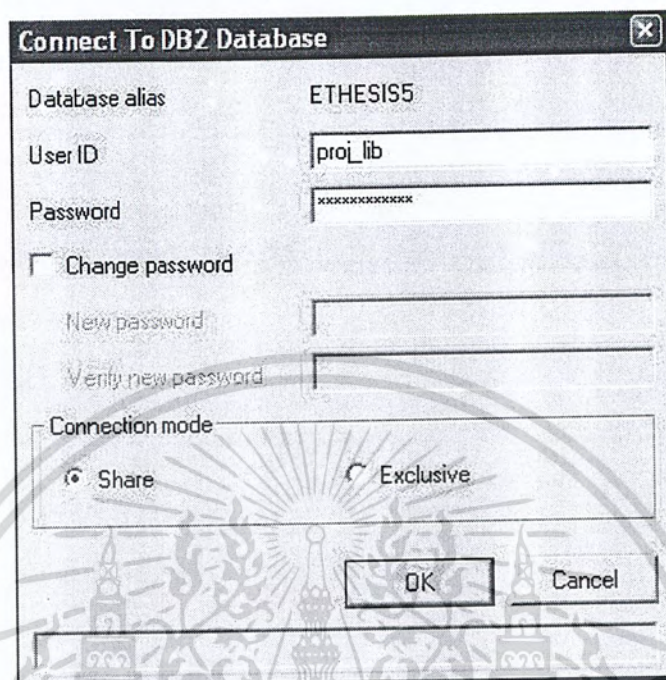
รูปที่ ข-8 แสดงการ Register ฐานข้อมูลใน ODBC

9. หลังจากกด “Finish” แล้วจะขึ้นหน้าต่าง Confirmation ถ้ากดปุ่ม “Add” โปรแกรมจะทำการเพิ่มฐานข้อมูลที่ได้กำหนดไว้ หรือว่าจะทดสอบการเชื่อมต่อกับฐานข้อมูลที่กำหนดไว้ โดยการกดปุ่ม “Test Connection”



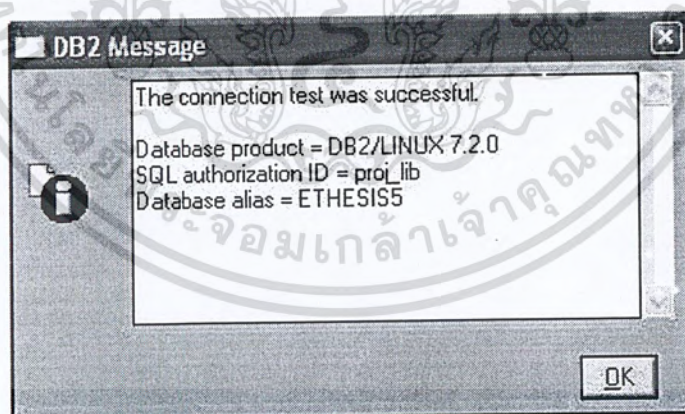
รูปที่ ข-9 แสดงหน้าต่างการ Confirmation

10. โปรแกรมจะแสดงหน้าต่าง “Connect To DB2 Database” จากนั้นใส่ User ID และ Password เพื่อทำการทดสอบการเชื่อมต่อ



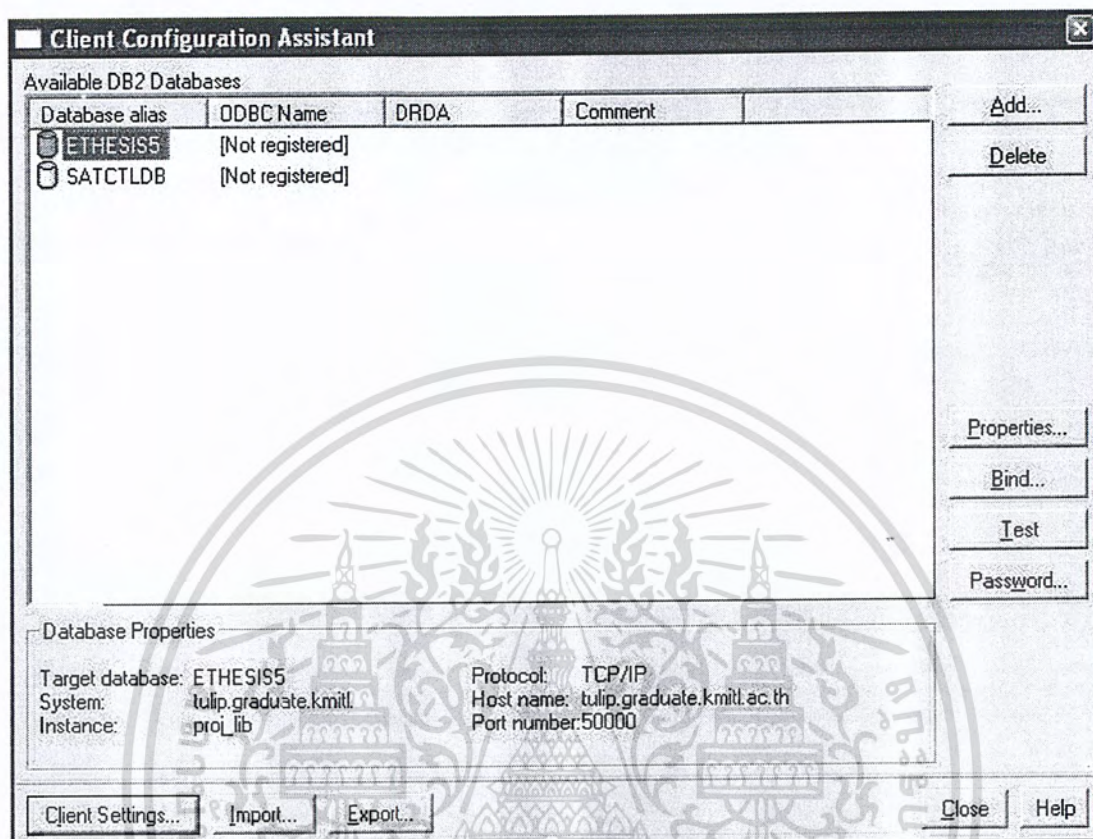
รูปที่ ข-10 แสดงการทดสอบการเชื่อมต่อ

11. ถ้าการเชื่อมต่อไม่มีปัญหาจะขึ้นหน้าต่างดังนี้



รูปที่ ข-11 แสดงการผลการทดสอบของการเชื่อมต่อฐานข้อมูล

12. เมื่อทำการ “Add” จากหน้าต่างรูปที่ ข-9 เรียบร้อยแล้ว ฐานข้อมูลที่เราได้ตั้งค่าการเชื่อมต่อไว้ก่อนหน้านี้จะขึ้นมาอยู่ในหน้าจอของโปรแกรม Client Configuration Assistant ดังรูป



รูปที่ ข-12 แสดงการฐานข้อมูลที่สามารถเชื่อมต่อได้

เมื่อได้ตั้งค่าการเชื่อมต่อกับฐานข้อมูลที่ต้องการติดต่อได้แล้ว เราสามารถใช้โปรแกรม Control Center เข้าไปจัดการกับฐานข้อมูลเหล่านี้ ไม่ว่าจะเป็นการ สร้างตาราง, ลบตาราง, หรือว่าดูข้อมูลที่อยู่ในตารางได้

บรรณานุกรม

- [1] Michael Floyd : “BUILDING WEB SITES WITH XML”, Prentice Hall PTR
- [2] Stephen A. Thomas : “HTTP Essentials”, Wiley Computer Publishing
- [3] Brett McLaughlin : “Java and XML”, O’Reilly & Associates, 2000
- [4] Marty Hall : “Core Servlet and Java Server Pages”, Prentice Hall PTR
- [5] Steve Holzner : “Inside XML”, New Rider
- [6] Jason Hunter, William Crawford : “Java Servlet Programming”, O’Reilly, 2001
- [7] Pornpavee Wattanajareonrung : “AN XML INFORMATION RETRIEVAL SYSTEM”, Asian Institute of Technology School of Advanced Technologies
- [8] Working Group on Dublin Core in Multiple Languages: “Dublin Core Metadata Element Set”

