การใช้ SPIDER ในการรวบรวมข้อมูลผ่านเว็บ

WEB SPIDER/BOT PROGRAMMING

ธนชาติ กิติดำรง

สมัชญ์ แสงสุพรรณ

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

WEB SPIDER/BOT PROGRAMMING

THANACHAT KITIDAMRONG

SMATH SANGSUBHAN

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2005

หัวข้อปัญหาพิเศษ	การใช้ SPIDER ในการรวบรวมข้อมูลผ่านเว็บ	
	WEB SPIDER/BOT PROGRAMMING	
ชื่อนักศึกษา	นายธนชาติ กิติดำรง	45050479
	นายสมัชญ์ แสงสุพรรณ	45050528
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	ดร.นวลสวาท หิรัญสกลวงศ์	
	ดร.พงษ์ชัย นิลาศ	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2548

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ดร.นันทิกา เบญจเทพานันท์	
กรรมการ	ผศ.สิริลักษณ์ อนันต์สถิตย์สิน	
กรรมการและอาจารย์ที่ปรึกษา	ดร.นวลสวาท หิรัญสกลวงศ์	
กรรมการและอาจารย์ที่ปรึกษา	ดร.พงษ์ชัย นิลาศ	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปัญหาพิเศษ	การใช้ SPIDER ในการรวบรวมข้อมูลผ่านเว็บ	
ชื่อนักศึกษา	นาย ธนชาติ กิติดำรง	45050479
	นาย สมัชญ์ แสงสุพรรณ	45050528
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2548	
อาจารย์ที่ปรึกษา	ดร.พงษ์ชัย นิลาศ	
	ดร.นวลสวาท หิรัญสกลวงศ์	

บทคัดย่อ

ปัจจุบันระบบการค้นคืนสารสนเทศต่างๆ บนอินเตอร์เน็ทมีมากมายและถือเป็นปัจจัยสำคัญในการนำไปสู่ความสำเร็จในวงการธุรกิจ โครงงานปัญหาพิเศษนี้จัดทำขึ้นเพื่อศึกษาและพัฒนาระบบการรวบรวมข้อมูลบนอินเตอร์เน็ทแบบอัตโนมัติโดยโปรแกรมที่เรียกว่า “Spider” ซึ่งเป็นโปรแกรมอย่างหนึ่งที่ทำหน้าที่ในการเก็บข้อมูลต่างๆ บนอินเตอร์เน็ท และตัวอย่างที่พบเห็นได้อย่างชัดเจนในการประยุกต์ใช้ โปรแกรมสไปเดอร์ (Spider) ได้แก่ การนำไปรวบรวมข้อมูลต่างๆ บนอินเตอร์เน็ทสำหรับระบบ Search Engine จึงทำให้คณะผู้จัดทำมีความสนใจที่จะศึกษาและได้ทำการพัฒนาโปรแกรมสไปเดอร์ขึ้นมาเองด้วยภาษาจาวา ซึ่งโปรแกรมนี้ถูกพัฒนาขึ้นเพื่อเก็บรวบรวมข้อมูลทุกประเภทที่อยู่บนอินเตอร์เน็ท โดยอัตโนมัติเพื่อลดค่าใช้จ่ายและเวลาในการทำงานจากการใช้มือคน

Special Project Title	WEB SPIDER/BOT PROGRAMMING
Student	Mr.Thanachat Kitidamrong 45050479
	Mr.Smath Sangsubhan 45050528
Degree	Bachelor of Science
Department	Mathematics and Computer Science, Faculty of Science
Programme	Computer Science
Academic Year	2005
Special Project Advisor	Dr.Pongchai Nilas
	Dr.Nualsawat Hiransakolwong

ABSTRACT

Nowadays information retrieval technology is an important factor to success in business. Our senior project is to study and tries to use advantage of automatically data crawling using the spider technology. This technology is a useful in reducing humans' workload and provided for the lower fixed cost. Our spider is developed using Java language along with other third-party programs including MS Access database and Artium installation wizard. This program is capable of automatically collecting information via websites and reprocessing information into the understandable report and database.

กิตติกรรมประกาศ

ในการทำโครงงานปัญหาพิเศษนี้สามารถสำเร็จลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบคุณ อาจารย์ พงษ์ชัย และ อาจารย์ นวลสวาท ถ้าไม่มีทั้งสองท่านงานนี้คงไม่สามารถสำเร็จได้ โดยเฉพาะอย่างยิ่งต้องขอขอบคุณอาจารย์พงษ์ชัย ที่ได้ริเริ่มแนวคิดและช่วยชี้แนะในการทำสไปเดอร์ อีกทั้งยังได้ช่วยจัดหาสถานที่ทำงาน ในสภาพแวดล้อมที่ดีให้ ซึ่งช่วยให้คณะผู้จัดทำสามารถทำงานได้อย่างมีประสิทธิภาพ และในที่สุดก็สำเร็จลุล่วงด้วยดี

คณะผู้จัดทำ

มีนาคม 2549

สารบัญ

เนื้อหา **หน้าที่**

สารบัญภาพ

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหาพิเศษ

เนื่องจากปัจจุบันอินเตอร์เน็ตเข้ามามีบทบาทในชีวิตประจำวันเป็นอย่างมาก และจำนวนเว็บไซต์ก็ได้มีการเพิ่มขึ้นอย่างรวดเร็ว การค้นหาข้อมูล หรือ การเก็บรวบรวมข้อมูลต่างๆ บนอินเตอร์เน็ตก็เป็นเรื่องที่สามารถหลีกเลี่ยงได้ยาก ไม่ว่าจะเป็นการเก็บข้อมูลประเภทไฟล์ html, รูปภาพ และ multimedia ต่างๆ ซึ่งหากเราต้องการเก็บรวบรวมข้อมูลเหล่านี้เป็นจำนวนมาก โดยใช้กำลังคนจะทำให้สิ้นเปลืองเวลาและค่าใช้จ่ายเป็นอย่างมาก ทั้งยังสิ้นเปลืองทรัพยากรบุคคลอีกด้วย

เนื่องจากความต้องการในการรวบรวมข้อมูลข้างต้น จึงมีการจัดทำโปรแกรมที่สามารถเก็บรวบรวมข้อมูลต่างๆ ที่อยู่บนอินเตอร์เน็ตโดยอัตโนมัติขึ้น เพื่อความสะดวกสำหรับผู้ใช้และช่วยประหยัดค่าใช้จ่าย

1.2 ความมุ่งหมายและวัตถุประสงค์ของปัญหาพิเศษ

ในการจัดทำปัญหาพิเศษนี้มีความมุ่งหมายและวัตถุประสงค์ดังต่อไปนี้

1. ศึกษาการทำงานของ web crawler
2. ศึกษาภาษา JAVA และฐานข้อมูล Access ในการพัฒนาโปรแกรม
3. ศึกษากลไกการทำงานแบบ Multithread
4. จัดทำโปรแกรมส่วนอินเตอร์เฟส เพื่อให้ง่ายต่อการใช้งาน

1.3 ขอบเขตของปัญหาพิเศษ

โครงงานปัญหาพิเศษนี้เป็นการศึกษาและจัดทำโปรแกรม Web Crawler ซึ่งสามารถเก็บรวบรวมข้อมูลจากอินเตอร์เน็ตโดยอัตโนมัติ ซึ่งโปรแกรมจะมีอินเตอร์เฟสอย่างง่ายสำหรับ user เพื่อให้ง่ายต่อการใช้งาน โดยโปรแกรมจะสามารถเก็บข้อมูลได้หลายประเภท ดังนี้

1. ไฟล์ *.html
2. ไฟล์ *.css
3. ไฟล์ *.pdf
4. รูปภาพทุกประเภท
5. ไฟล์ Audio ประเภท mp3, wma

ผู้ใช้สามารถกำหนดเวปไซต์เริ่มต้นในการ Crawler ได้ โดยกำหนดค่าผ่านทางอินเตอร์เฟสจากนั้นโปรแกรมจะทำการ Crawler ไปเรื่อยๆ โดยสามารถกำหนดได้ว่าจะต้องการ Crawl ภายในเวปไซต์นั้นแค่เวปไซต์เดียว หรือว่าต้องการตามไป Crawl เวปไซต์อื่นที่มี Link มาจากเวปไซต์เริ่มต้นก็ได้

ซึ่งผู้ใช้สามารถกำหนดขอบเขตการทำงานของโปรแกรมได้โดยกำหนดความลึกในโปรแกรมผ่านทางอินเตอร์เฟส โปรแกรมจะไม่สามารถเข้าไป Crawler ภายในเวปไซต์ที่ต้องมีการใส่ Username และ Password ได้เนื่องจากจะเป็นการละเมิดสิทธิ์ของเวปไซต์นั้น

1.4 ขั้นตอนในการดำเนินงาน

ในการทำปัญหาพิเศษนี้มีขั้นตอนในการดำเนินงานดังต่อไปนี้

1. ศึกษากลไกการทำงานของ web crawler
2. ศึกษาปัญหาและรวบรวมข้อมูลที่เกี่ยวกับปัญหา
3. ศึกษาเครื่องมือในการพัฒนาโปรแกรม
   - ภาษาที่ใช้ในการเขียนโปรแกรม ได้แก่ JAVA
   - ระบบปฏิบัติการที่ใช้ คือ Microsoft Window XP
   - โปรแกรม j2sdk1.4.2_07
   - ฐานข้อมูลที่ใช้ ได้แก่ Access2003
4. ออกแบบระบบงานฐานข้อมูล และ ส่วนติดต่อกับผู้ใช้
5. จัดทำโปรแกรมตามที่ออกแบบไว้
6. ทดลองใช้จริงเพื่อศึกษาข้อผิดพลาดต่างๆ เพื่อนำมาปรับปรุง และแก้ไขโปรแกรมให้มีประสิทธิภาพมากขึ้น
7. สรุปและวิเคราะห์ปัญหาพร้อมทำเอกสารประกอบในการทำปัญหาพิเศษ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากการทำปัญหาพิศษนี้ สามารถแบ่งออกได้ดังนี้

1. เรียนรู้การเขียนโปรแกรมที่ทำงานแบบ web crawler
2. เรียนรู้การเขียนโปรแกรมโดยใช้การประมวลผลแบบขนาน(Multithread)
3. เรียนรู้การบริหารและจัดการทรัพยากรอย่างประหยัด
4. เขียนและพัฒนาโปรแกรมที่มีความสามารถในการเก็บรวบรวมข้อมูลที่มีประสิทธิภาพ

1.6 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

สามารถทำการแบ่งอุปกรณ์ที่ใช้ในการทำปัญหาพิเศษนี้ได้เป็น 2 ส่วนคือ

1. ฮาร์ดแวร์
   - เครื่องคอมพิวเตอร์ความต้องการขั้นต่ำ Pentium4 (2.4 MHz) ขึ้นไป
   - ฮาร์ดดิสก์ขนาดความจุขั้นต่ำ 30 GB
   - หน่วยความจำขนาด 256 Mb
   - Modem ADSL ความเร็วขั้นต่ำ 256 kbps

2. ซอฟต์แวร์
   - เครื่องคอมพิวเตอร์ที่รันประกอบด้วย J2sdk1.4.2_07
   - ระบบปฏิบัติการ Microsoft Window XP
   - Access Database
   - โปรแกรม netbeans ใช้สำหรับเขียนโปรแกรมภาษา JAVA

บทที่ 2
ทฤษฎีและหลักเกณฑ์ที่เกี่ยวข้อง

2.1 หลักการทำงานของ Web Crawler หรือ Spider

Spider ถูกออกแบบให้ค้นหาหน้า web page ตาม content ที่ผู้ใช้ต้องการ โดยจะเริ่มจากระบุ single web page จากนั้นก็จะทำการหา reference ภายใน page นั้นเพื่อจะออกไปค้นหาข้อมูลยัง web page อื่นๆ การทำ Spidering จะหยุดก็ต่อเมื่อได้ทำการเยี่ยมทุก website ที่มีการอ้างถึงในหน้า webpage แต่ละหน้าแล้ว นั่นหมายความว่า อาจจะท่อง website ไปเรื่อยๆ จนครบทุก website บน internet ก็เป็นได้

ประเด็นสำคัญของการ Spider คือ การเปิด url แล้วทำการหา Link ภายใน Page นั้นว่ามีการชี้ไปที่ไหนบ้าง โดยจะเก็บ url ที่สามารถหาได้ในแต่ละ page มาเพื่อทำการเปิด url ที่เก็บมาได้แล้วหา Link ต่อไปเรื่อยๆ และ ยังสามารถบอก Broken Link ใน page ได้อีกด้วยในกรณีที่ Link นั้นไม่มีอยู่จริง

โดย Attribute ที่ใช้กันมากที่สุดในการ reference ไปสู่อีก webpage หนึ่ง เรียกว่า "a hypertext reference" (href)

2.1.1 ประเภทของ href แบ่งออกเป็น 3 ประเภท

1. Internal Link
2. External Link
3. Other Link

1. **Internal Link** – เป็น Link ที่เกิดจากการ Link หน้า Web Page ภายใน Website เดียวกัน เช่น main
2. **External Link** – เป็น Link ที่เกิดจากการ Link ไปยัง WebSite อื่นๆ เช่น main
3. **Other Link** – เป็น Link ที่ไม่ได้มีการชี้ไปยัง Web Page ใดๆ เช่น to

2.1.2 Spider Queue แบ่งออกเป็น 4 ชนิด

1. *Waiting Queue*
2. Running Queue

3. Complete Queue

4. Error Queue

1).Waiting Queue

เมื่อได้ทำการ Check ว่า URL ที่ Collect ได้จาก Web Page นั้นยังไม่ได้เยี่ยมถึงจะทำการ add URL นั้นๆ ลงใน Waiting Queue เพื่อรอการถูก Parsing (Collect Link)

2).Running Queue

เมื่อจะทำการ process url จะทำการดึง url ออกมาจาก Waiting Queue แล้วทำการ add URL นั้นลง Running Queue เพื่อแสดงว่าขณะนี้กำลังรัน Process url นี้อยู่

3).Complete Queue

เมื่อได้ทำการเยี่ยม URL นั้นๆแล้ว จะ Remove URL นั้นออกจาก Waiting Queue และทำการ add URL นั้นลงใน Complete Queue แทน เพื่อเป็นการแสดงว่า URL นี้ได้ถูกเยี่ยมและทำการ Collect Link ที่อยู่ภายใน URL นี้ไปแล้ว และจะไม่มีการกลับมาเยี่ยม URL นี้อีก (non-repetitive)

4).Error Queue

เมื่อ URL ที่กำลังเยี่ยมอยู่นั้นไม่มีอยู่จริงหรือไม่สามารถ Connect หน้า Web Page นี้ได้ จะถูก Remove URL นั้นออกจาก Waiting Queue และทำการ add URL นั้นลงใน Error Queue แทน เพื่อเป็นการแสดงว่า URL นี้ได้ถูกเยี่ยมไปแล้ว และจะไม่มีการกลับมาเยี่ยม URL นี้อีก (non-repetitive)

2.2 เทคโนโลยีแบบ Multithread

Process – คือกิจกรรมที่เป็นผลมาจากการทำงานของโปรแกรม

2.2.1 ที่มาของการทำงานของ Thread

ในระบบปฏิบัติการแบบ Multitasking ผู้ใช้สามารถส่งโปรแกรมให้แก่ระบบปฏิบัติการทำงานมากกว่า 1 โปรแกรมในเวลาหนึ่งได้ โดยโปรแกรมเหล่านี้จะไปเข้าคิวเพื่อรอเข้าทำงานในหน่วยประมวลผล โดยความเร็วของคอมพิวเตอร์ทำให้การทำงานลักษณะนี้ดูคล้ายกับว่าโปรแกรมเหล่านั้นทำงานไปพร้อมๆ กัน ทั้งที่จริงแล้วเป็นการจัดคิวให้เข้าทำงานทีละโปรแกรม โปรแกรมส่วนใหญ่มักต้องทำงานหลายอย่างไปพร้อมๆ กัน เช่น ในขณะหนึ่งอาจอ่านข้อมูลจาก Disk พร้อมกับแสดงตัวอักษรที่จอภาพ หากงานทั้งสองนี้ถูกแยกกันทำเป็น 2 Process และทำเป็นเวลานาน จะเห็นการแสดงตัวอักษรสะดุดเป็นช่วงๆ เนื่องจากมีการสลับกันทำงานระหว่าง Process ที่อ่าน

ข้อมูลจาก Disk กับ Process ที่พิมพ์ตัวอักษรออกทางจอภาพ การเปลี่ยนการทำงานของ Process หนึ่งไปสู่อีก Process หนึ่ง เรียกว่า "Context Switching"

ในขั้นตอนนี้ต้องมีการเก็บสถานการณ์การคำนวณของ Process เดิมไว้ เพื่อให้สามารถกลับมาทำงานต่อจากจุดนั้นได้ และต้องโหลดสถานะการคำนวณของ Process เดิม เพื่อให้กลับมาทำงานต่อจากจุดนั้นได้และต้องโหลดสถานการณ์คำนวณของ Process ใหม่เข้าไปทำงานแทนที่ ซึ่งภาระการทำ context switching นี้เป็น overhead ที่ทำให้โปรแกรมทำงานช้าลง ผู้ออกแบบระบบปฏิบัติการและภาษาสำหรับโปรแกรมจึงหาวิธีทำให้ overhead ของ context switching ลดลงด้วยการเสนอกลไกใหม่ที่เรียกว่า "Thread"

2.2.2 Multi Thread ทำงานอย่างไร

ระบบปฏิบัติการแบบ Multi-Threading จะสามารถแบ่ง Process ออกเป็นหน่วยที่เล็กกว่า ซึ่งเรียกว่า Thread ระบบปฏิบัติการแบบนี้สามารถสร้างหลาย threads ในหนึ่ง Process โดยปกติ Thread คือ execution flow ของการทำงานอย่างหนึ่ง หากนำหลายๆ threads มารวมกันใน Process หนึ่งจะช่วยให้สามารถทำงานได้มากกว่าหนึ่งอย่างไปพร้อมๆ กัน โดยไม่ต้องมีการทำ context switchingโปรแกรมจึงทำงานได้เร็วขึ้นและเป็นการใช้งานหน่วยประมวลผลได้อย่างมีประสิทธิภาพมากขึ้นด้วย

การทำงานแบบ MultiThread จะทำให้การทำงานของโปรแกรมสามารถทำงานได้เร็วขึ้น เนื่องจากเป็นการทำงานแบบขนานกัน นั่นคือจะมีหลายงานที่ได้รับการประมวลผลในเวลาเดียวกัน

2.2.3 การนำหลักการของ MultiThread มาประยุกต์ใช้กับโครงงาน

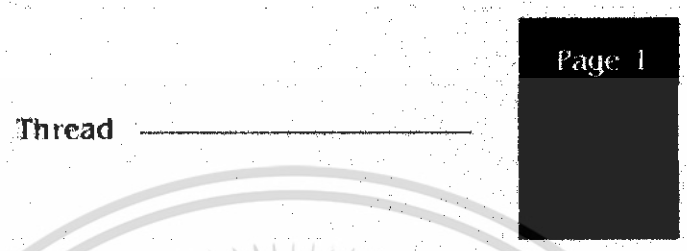
จะมีการเปิด Page หลายๆ Page เพื่อประมวลผลพร้อม ๆกันในเวลาเดียวกัน ซึ่งจะทำให้โปรแกรมสามารถทำงานได้เร็วขึ้นหลายเท่า

ผลลัพธ์ของความแตกต่างจากการใช้ MultiThread และไม่ใช้ MultiThread

สมมติว่าในแต่ละ Page มี Link ทั้งหมด 10 Link ถ้าไม่ใช้ MultiThread ในการประมวลผล ก็จะมีการประมวลผลทีละ 1 Page นั่นหมายความว่า ณ เวลาหนึ่งจากการประมวลผล จะได้ทั้งหมด 10 Link แต่ถ้าเป็นการประมวลผลแบบ MultiThread สมมติว่าทำการสร้าง Thread ขึ้นมา 10 Thread การทำงานคือ แต่ละ Thread จะทำการวิ่งไปเปิด Page แต่ละหน้า นั่นหมายความว่า ณ เวลาหนึ่งจากการประมวลผล จะได้ทั้งหมด 100 Link ซึ่งใช้เวลาในการทำงานเท่ากัน แต่สามารถเก็บข้อมูลได้มากกว่าหลายเท่าตัว

การทำงานของ MultiThread จึงเป็นประโยชน์สำคัญอย่างยิ่งในการพัฒนา Algorithm ของโปรแกรมเพื่อเพิ่มประสิทธิภาพในแง่ของความเร็วในการประมวลผล

จากรูปแสดงการประมวลผลแบบไม่ใช้ MultiThread



ภาพที่ 2.1 แสดงการทำงาน Single Thread

2.2.4 การ Synchronized

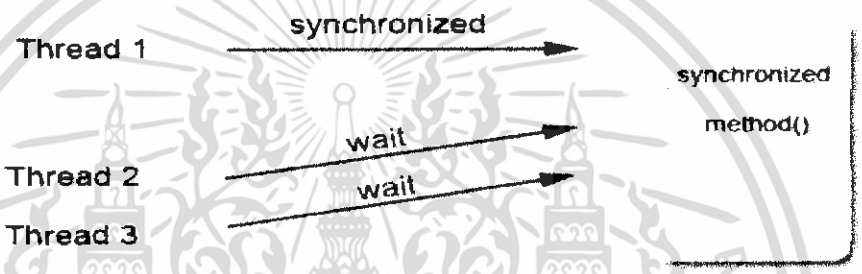
การทำงานของ Thread จะทำงานไปพร้อมๆ กัน ถ้า Thread แต่ละตัวใช้ Resource แยกกันก็จะไม่เกิดปัญหาแต่ถ้า Thread แต่ละตัวทำงานพร้อมๆ กันและมีการใช้ Resource ร่วมกัน ซึ่งจะทำให้เกิดปัญหาคือ อาจทำให้ค่าที่ได้ผิดพลาด เพราะมีการใช้ทรัพยากรร่วมกันในเวลาเดียวกัน เช่น สมมติเรามี Method ที่ใช้ในการบวกเลขขึ้นอีก 1 และให้ Thread มีการทำงานแบบ aSynchronized จะทำให้ค่าที่ return ออกมาผิดพลาด

ต.ย สมมติว่ามี Global Variable เป็น int ซึ่งมีค่าเป็น 1 และมี Thread ที่ 1 และ Thread ที่ 2 เข้าไปดึงค่าของตัวแปรเพื่อนำค่าออกมาบวกแต่ถ้าหาก Thread ทั้ง 2 สามารถเข้าใช้ทรัพยากรได้พร้อมกัน เมื่อ Thread ตัวแรกดึงค่าออกมาได้เป็น 1 และนำมาบวกเพิ่มขึ้นอีก 1 จะได้ค่าเป็น 2 และจัดเก็บแต่ Thread ตัวที่ 2 ก็ได้ค่าที่ดึงออกมาเป็น 1 เช่นกันเพราะได้เข้าใช้ทรัพยากรพร้อมกันไปตั้งแต่แรกแล้ว Thread ที่ 2 จึงนำค่ามาบวกขึ้นอีก 1 จะได้ค่าเป็น 2 ซึ่งไม่ถูกต้องเพราะจริงๆ แล้วคำตอบที่ถูกต้องจริงๆ ควรจะเป็น 3 เพราะมี Thread เข้ามาทำงาน 2 ตัว

การแก้ไขปัญหาการเข้าใช้ทรัพยากรพร้อมๆ กันของ Thread คือการประกาศให้ Method ที่ทำการบวกเลขขึ้นอีก 1 นั้นเป็น synchronized method ดังนี้

```
public synchronized  int plus_number(int num)
        {
                return num+1;
         }
```

เมื่อใส่ keyword synchronized ที่หน้า method จะทำให้ method นั้นถูกครอบครองได้โดย Thread เพียง Thread เดียวเท่านั้น และเมื่อ Thread นั้นทำงานเสร็จแล้วจึงจะปล่อยทรัพยากรให้กับ Thread อื่นที่รอการใช้งานอยู่ จึงทำให้การทำงานที่มีการใช้ทรัพยากรร่วมกันถูกต้อง



ภาพที่ 2.2 ภาพแสดงการ Synchronize Thread

2.3 JAVA กับฐานข้อมูล Access

เมื่อโปรแกรมภาษา JAVA ต้องการติดต่อกับระบบฐานข้อมูลใด ก็จะต้องมี driver สำหรับระบบฐานข้อมูลนั้น เพื่อทำหน้าที่เชื่อมต่อระหว่างโปรแกรมกับฐานข้อมูล แต่ระบบฐานข้อมูลมีความแตกต่างกันไปในแต่ละยี่ห้อ ดังนั้น driver ที่สร้างขึ้นสำหรับติดต่อกับ JDBC จึงถูกสร้างขึ้นด้วยวิธีที่แตกต่างกัน

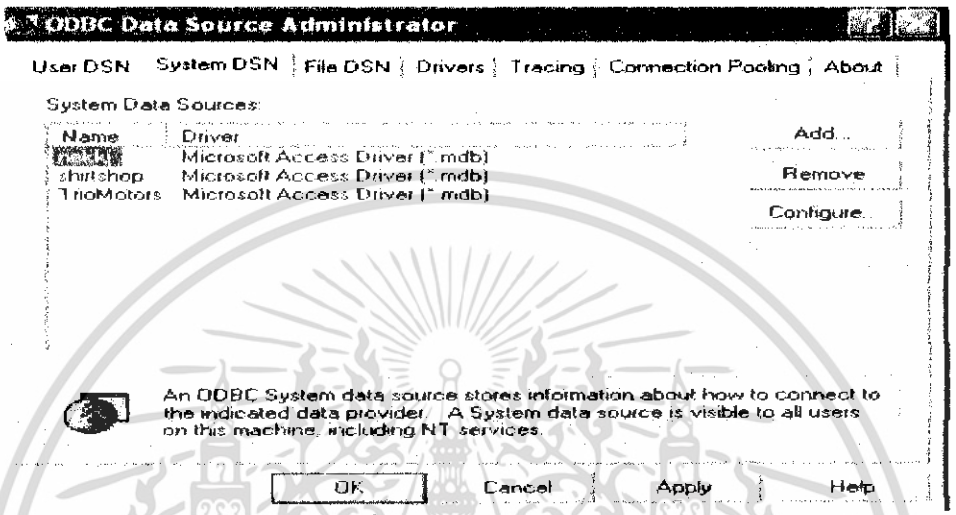
ODBC (Open Database Connectivity) เป็น API ที่ถูกใช้อย่างแพร่หลาย สามารถติดต่อกับระบบฐานข้อมูลได้หลายยี่ห้อ และสามารถใช้ได้หลาย platform แต่สาเหตุที่เราไม่เขียนโปรแกรมเพื่อเรียกใช้ ODBC ตรงๆ จากภาษา JAVA เนื่องจากว่า ODBC ถูกเขียนขึ้นด้วยภาษา C จึงมีปัญหาในด้านความปลอดภัยของข้อมูลและความเข้ากันได้ของโปรแกรม ดังนั้นจึงมี JDBC-ODBC Bridge เพื่อเป็นสะพานในการเชื่อมต่อการทำงานจากโปรแกรมภาษา JAVA ไปยัง ODBC Driver อีกที

ในตัวอย่างนี้จะใช้ Database ของ Access ในการเก็บข้อมูล โดยภาษา JAVA สามารถติดต่อกับฐานข้อมูล Access ได้ผ่านทาง jdbc:odbc Bridge

2.3.1 วิธีการ Set ค่า DSN เพื่อใช้ในการติดต่อกับฐานข้อมูล

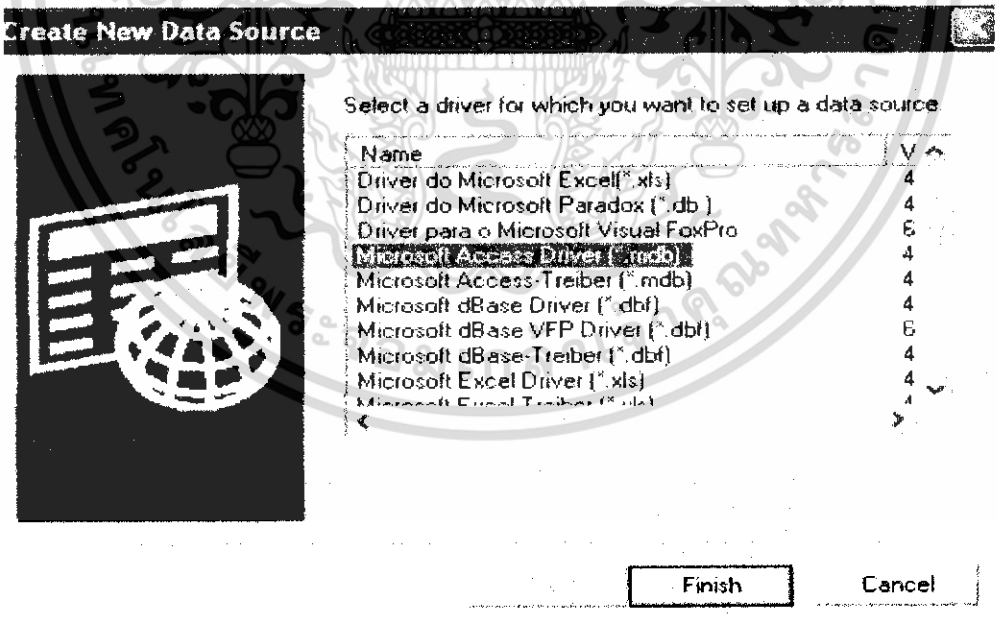
ก่อนใช้งานโปรแกรมจะต้องทำการ Set ODBC เพื่อติดต่อฐานข้อมูลก่อน ตามวิธีดังนี้

1. ไปที่ Control Panel แล้ว Double Click ที่ Administrative Tools และที่ Data Sources (ODBC) แล้วเลือก System DSN จะปรากฏดังรูป



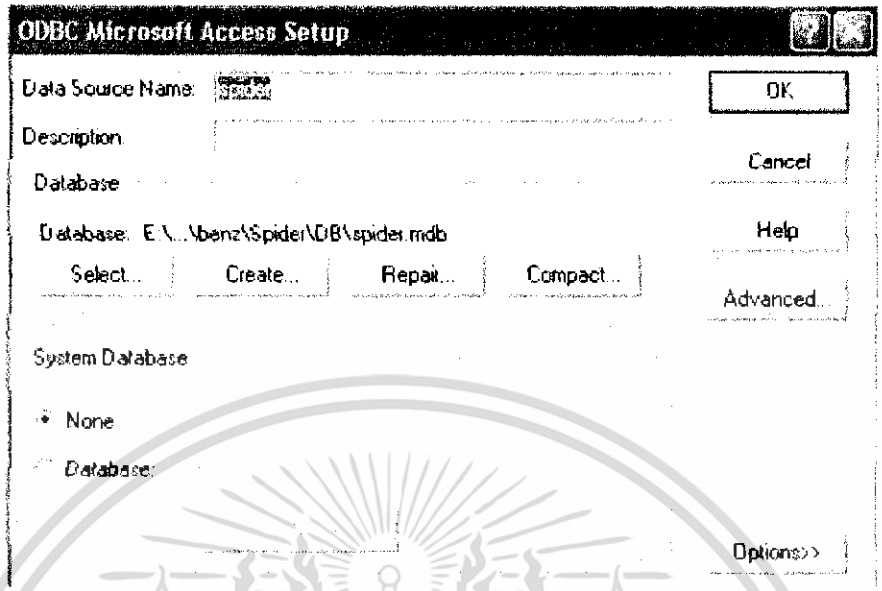
ภาพที่ 2.3 ภาพแสดงการ Set ODBC 1

2. Click ที่ Add... และ เลือก Microsoft Access Driver ดังรูปแล้วClick Finish



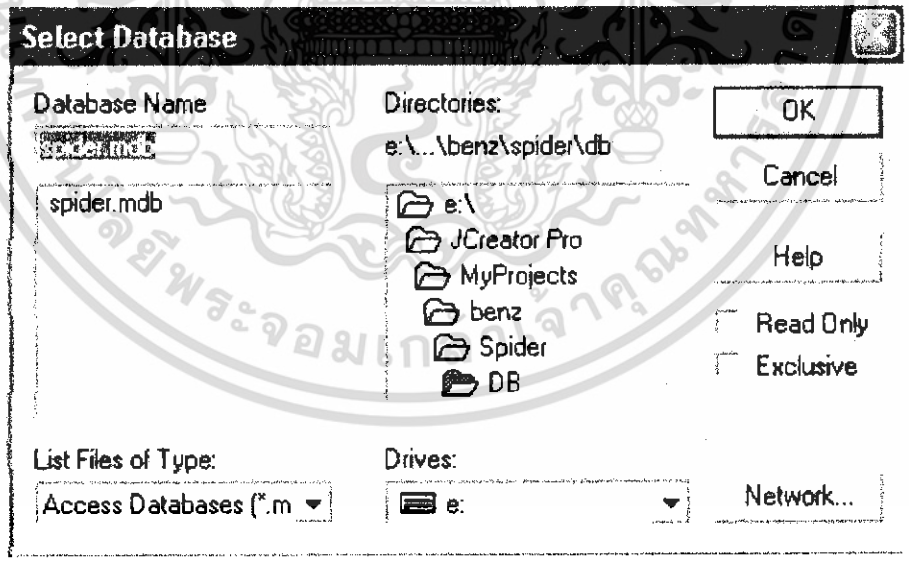
ภาพที่ 2.4 ภาพแสดงการ Set ODBC 2

3. ใส่ชื่อที่ Data Source Name ว่า "spider" ดังรูปแล้ว Click Select...



ภาพที่ 2.5 ภาพแสดงการ Set ODBC 3

4. ทำการเลือก File จาก folder database ชื่อ spider.mdb แล้ว Click OK ดังรูป เป็นอันเสร็จขั้นตอนการSet ODBC และท่านสามารถทำการ Run Program เพื่อใช้งานได้แล้ว



ภาพที่ 2.6 ภาพแสดงการ Set ODBC 4

2.3.2 Programming with JDBC

ก่อนที่จะใช้งาน JDBC ได้นั้น จำเป็นต้อง import java.sql.* เข้ามาก่อน จึงจะสามารถใช้งาน Class ที่เกี่ยวกับการติดต่อกับ Database ได้

วิธีการติดต่อกับฐานข้อมูล Access

Step 1 : กำหนด Driver สำหรับฐานข้อมูลของ Access มี code ดังนี้

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Step 2 : กำหนดค่า DSN ที่ใช้สำหรับติดต่อกับฐานข้อมูล Access ผ่านทาง method getConnection() ของ Class DriverManager มี code ดังนี้

```
Connection connection;
        connection = DriverManager.getConnection("jdbc:odbc:spider","","");
```

Step 3 : เตรียมคำสั่ง SQL เพื่อรอการ Query ผ่านทาง Method PreparedStatement ของ Class Connection มี code ดังนี้

```
prepAssign = connection.prepareStatement("select url from spider where status=? ");
```

เครื่องหมาย ? จะถูกกำหนดค่าได้ผ่านทาง Method setString(int,String) เช่น สมมติว่าเราต้องการกำหนดค่าของ Status เป็น "W" สามารถทำได้ดังนี้

```
prepAssign.setString( 1 , "W" );
```

Parameter ตัวที่ 1 คือ ตำแหน่งของเครื่องหมาย? ส่วน Parameter ตัวที่ 2 คือ ค่าของ Parameter นั้นๆ หลังจากเรียกคำสั่งข้างบนแล้ว คำสั่ง SQL ที่ได้คือ

```
select url from spider where status= W
```

Step 4 : ประมวลผลคำสั่ง SQL มี code ดังนี้

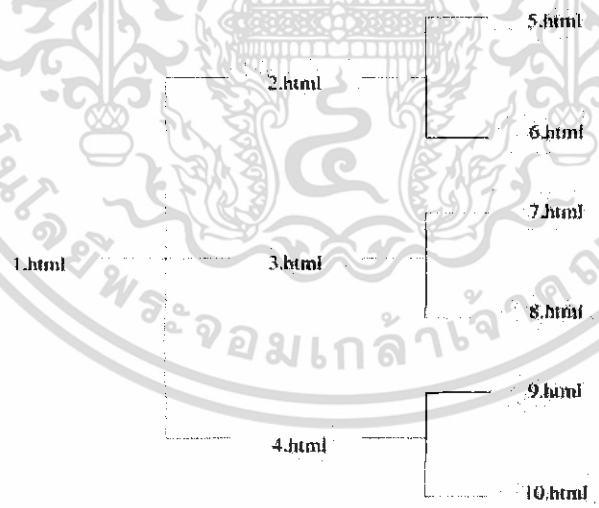
```
prepAssign.executeQuery();
```

Step 5 : นำ Object ResultSet มารับผลลัพธ์ที่ได้เพื่อนำไปแสดง

```
ResultSet r = prepAssign.executeQuery();

        if (r.next())
        {
            System.out.println(r.getString("url"));
        }
```

2.4 ลำดับชั้น (Depth)

ก่อนที่จะอธิบาย Algorithm การกำหนดลำดับชั้นนั้น จะขออธิบายในส่วนของชั้น (Depth) เพื่อให้มีความเข้าใจตรงกันก่อน ดังนี้



ภาพที่ 2.7 ภาพแสดง Depth

ถ้ากำหนดขอบเขตว่าต้องการ Spider โดยให้มีความลึกเป็น 1 ชั้น

- จะทำการ Process ภายในหน้า 1.html เท่านั้น และ ผลลัพธ์ที่ได้จะได้ url คือ 2.html , 3.html , 4.html

ถ้ากำหนดขอบเขตว่าต้องการ Spider โดยให้มีความลึกเป็น 2 ชั้น

- จะทำการค้นหา Process ภายในหน้า 1.html , 2.html , 3.html , 4.html เท่านั้น ซึ่งผลลัพธ์จะได้ url คือ 2.html , 3.html , 4.html , 5.html, 6.html , 7.html , 8.html , 9.html , 10.html

2.5 เทคนิคในการเปิดอ่านและเขียนข้อมูลลงไฟล์

2.5.1 Streams

คือ วัตถุสมมติที่มีลักษณะคล้ายท่อบรรจุข้อมูล มีข้อมูลเรียงลำดับกันในลักษณะแถวเรียงหนึ่ง สามารถนำ stream มาต่อระหว่างโปรแกรมกับอีกโปรแกรมหนึ่ง หรือหน่วยความจำ ก็ได้ ทำให้โปรแกรมสามารถอ่านหรือเขียนข้อมูลจาก stream ที่ต่อ โดยโปรแกรมนั้นไม่จำเป็นต้องสนใจว่าอีกปลายของ stream นั้นต่ออยู่กับอะไร

เช่น สมมติว่า stream ปลายด้านหนึ่งต่ออยู่กับโปรแกรมและปลายอีกด้านหนึ่งต่ออยู่กับไฟล์ เมื่อโปรแกรมส่งข้อมูลออกไปที่ stream ข้อมูลนั้นก็จะถูกเก็บลงในไฟล์ แต่หากเปลี่ยน stream นั้นไปต่อกับ socket ของ network เพื่อส่งข้อมูลไปยังอีกเครื่องหนึ่ง กรณีนี้เมื่อโปรแกรมส่งข้อมูลออกไปที่ stream ด้วยวิธีเดิม ข้อมูลนั้นก็จะถูกส่งไปที่อีกเครื่องตามที่เราต้องการ ซึ่งขึ้นอยู่กับผู้เขียนโปรแกรมว่าจะนำไปต่อกับอะไร วิธีนี้ช่วยให้เราสามารถสร้างโปรแกรมที่ติดต่อกับภายนอกด้วยวิธีที่สากล ไม่ขึ้นกับสิ่งที่มันติดต่อด้วยและสนับสนุนการนำโปรแกรมเดิมกลับมาใช้ได้อีกในหน้าที่อื่นๆ เช่น นำโปรแกรมสำหรับเขียนข้อมูลลงไฟล์มาใช้ในการส่งข้อมูลไปใน network เป็นต้น " *ภาษา JAVA นั้นจะกำหนดคลาสของ stream ชนิดต่างๆ ไว้ใน package ที่ชื่อ java.io* "

2.5.2 การนำ stream มาประยุกต์ใช้กับโครงงาน

ในการอ่านข้อมูลหรืออ่าน Content ของไฟล์ที่กำลัง process อยู่ได้นั้นต้องใช้ stream ในการอ่านเข้ามา โดยสมมติว่าถ้าต้องการอ่าน content ทั้ง page ที่กำลัง process อยู่ จะมีปลายของ stream ด้านหนึ่งต่ออยู่กับ url ที่กำลังเปิดอยู่ ส่วนปลายอีกด้านหนึ่งให้ต่อไปที่ไฟล์ และจากนั้นก็ทำการอ่านข้อมูลจาก url นั้นมาเพื่อเก็บลงไฟล์ ก็จะได้ content ทั้งหมดใน page นั้นมาเก็บลงไฟล์ที่ถูกสร้างไว้

2.6 J2SE : Java 2 Platform, Standard Edition (Core/Desktop)

J2SE เป็นเสมือน Library ที่รวมรวม Method และ Function Call ต่างๆของ ภาษา Java และ ยังช่วยสนับสนุนในการ จัดเตรียม สภาวะแวดล้อมที่เหมาะสม สำหรับการ พัฒนา Java Application ทั้งบนฝั่ง Desktop และ Server ซึ่งได้จัดเตรียมการทำงานด้านต่างๆ ไว้สำหรับช่วยในการ พัฒนา Application ในด้านต่างๆ เช่น การรักษาความปลอดภัย, การเชื่อมต่อฐานข้อมูล และด้านอื่นๆอีกมากมาย บางครั้งก็เรียก J2SE ว่า J2EE หรือ J2SDK ซึ่งในการทำงานต้องการ J2SE ในการ Compile และ Run โปรแกรม

สำหรับเวอร์ชั่น ที่ถูกใช้ในโครงงานนี้ คือ J2SE Version 1.4.0 เนื่องจากการพัฒนา Application ซึ่ง ณ ปัจจุบันนี้ Sun ได้ออก Version 1.5.0 มาแล้ว แต่พบว่ายังมีความบกพร่องในหลายๆเรื่อง อีกทั้ง ฝั่ง Client ส่วนมากยังรองรับกันแค่ Version 1.4.0 คณะผู้จัดทำจึงเลือกใช้ เวอร์ชั่น 1.4.0 ในการพัฒนา

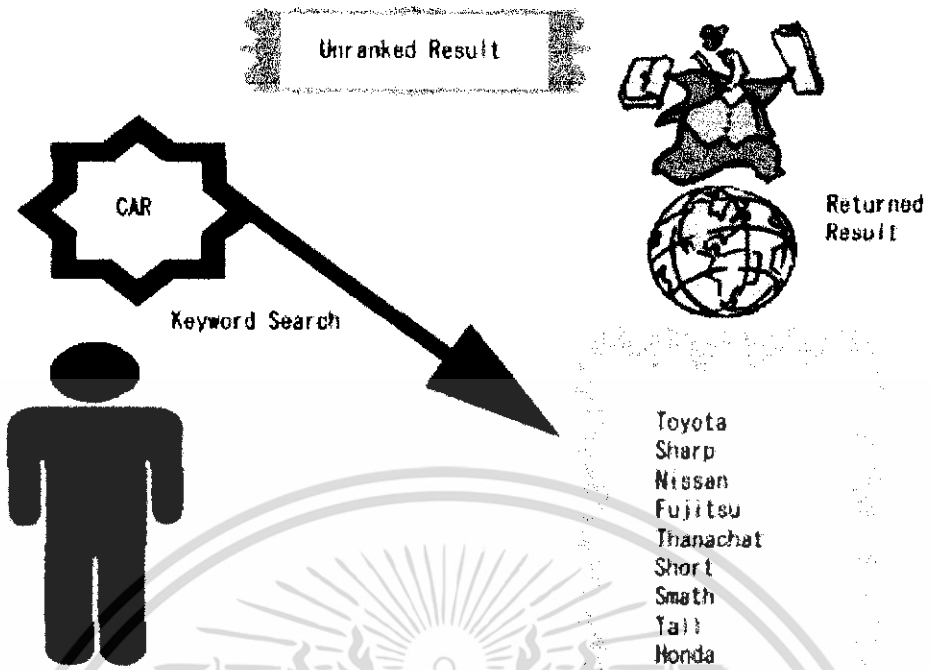
Sun Download Website: http://java.sun.com/j2se/index.jsp

2.7 Algorithm การทำ Ranking Webpage

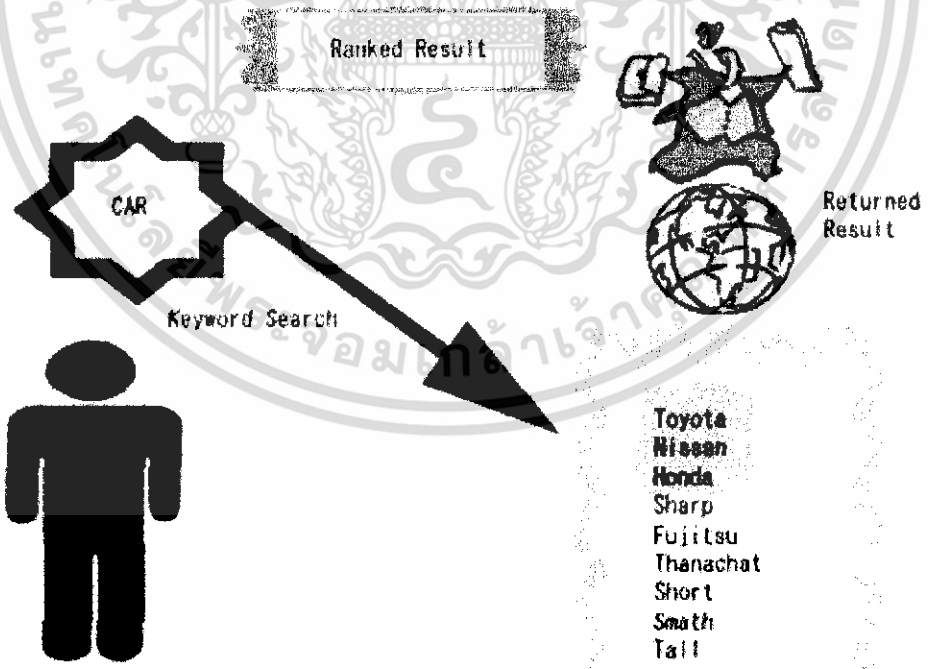
หลังจากที่ Spider ได้ทำการรวบรวมเว็บเพจ หรือ ไฟล์ ต่างๆมาให้แล้ว สิ่งต่อไปที่ควรจะทำคือ การจัดอันดับของเอกสารหรือ เว็บเพจทุกอัน ที่ได้รวบรวมมา โดยการจัดอันดับนี้ จะพยายามจัดอันดับให้ออกมาตรงใจ หรือตรงต่อความต้องการของ User ให้มากที่สุด คำว่าตรงใจ User หมายถึงว่า ผลการค้นหานั้นๆ ได้ผลออกมาตรงเรื่องหรือตรงกับสิ่งที่ User ต้องการจะค้นหา โดยต้องสามารถเรียงได้จากสิ่งที่ตรงมากสุดจนถึงสิ่งที่ไม่ค่อยตรง

การทำ Ranking Webpage นั้นอาจจะมีการทำได้หลายรูปแบบ โดยแบบที่ง่ายที่สุดก็คือการนับจำนวนคำซ้ำในแต่ละ Document ยิ่งถ้ามี Keyword อยู่ในเนื้อหาของ เอกสารมากเท่าไร ยิ่งแสดงว่ามีโอกาสจะเป็นเรื่องที่เกี่ยวข้องกับสิ่งที่ต้องการค้นหามากขึ้น แต่วิธีนี้ก็ยังมีจุดบกพร่องอีกมากทำให้เกิด กรณีที่ผิดพลาดอีกหลายกรณี โดยอาจจะใช้สมการหรือ Algorithm ที่ได้ พิสูจน์มาแล้วว่ามีประสิทธิภาพ มาใช้ก็ได้เพื่อให้ได้ผลออกมาตรงใจ User ส่วนใหญ่มากที่สุด

หลังจากที่ได้ทดลองวัดประสิทธิภาพแล้ว โดยคำนึงถึงความเหมาะสมกับตัว Spider จึงเลือกใช้ Vector Space Model Algorithm ในการ Ranking Webpage



ภาพ2.8 แสดงข้อมูลที่ได้เมื่อยังไม่ได้ทำการจัดอันดับ



ภาพ2.9 แสดงข้อมูลที่ได้เมื่อได้ทำการจัดอันดับแล้ว

2.7.1 Vector Space Model Algorithm

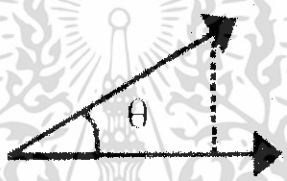
Vector Space Model (VSM) เป็น รูปแบบการจัดอันดับของ เอกสารแบบหนึ่งที่ได้รับความนิยมมาก โดยสามารถจัดอันดับได้ว่าเอกสารใด ใกล้เคียงกับสิ่งที่ต้องการค้นหาออกมาเป็นอันดับเรียงกัน โดยไม่ได้พิจารณาแต่เพียงจำนวนคำที่ซ้ำในแต่ละเอกสาร แต่ยังมีการใช้สมการและ วิธีการคำนวณที่ซับซ้อนเพื่อให้ได้คำตอบออกมาตรงใจ User มากที่สุด ก่อนที่เราจะรู้วิธีการทำเราจะต้องรู้ทฤษฎีต่อไปนี้ก่อน

Documents and queries are both vectors

$$\vec{d}_i = (w_{i,1}, w_{i,2} \cdots w_{i,t})$$

each wi,j is a weight for term j in document i

"bag-of-words representation"



Similarity of a document vector to a query vector = cosine of the angle between them

Cosine Similarity Measure

$$sim(d_i, q) = \cos\theta$$

$$(x \cdot y = |x||y|\cos\theta)$$

$$= \frac{d_i \cdot q}{|d_i||q|} = \frac{\sum_j w_{i,j} \times w_{q,j}}{\sqrt{\sum_j w_{i,j}^2}\sqrt{\sum_j w_{q,j}^2}}$$

Cosine is a normalized dot product

Documents ranked by decreasing cosine value

sim(d,q) = 1 when d = q

sim(d,q) = 0 when d and q share no terms

Term Weighting

Higher weight = greater impact on cosine

Want to give more weight to the more "important" or useful terms

What is an important term?

If we see it in a query, then its presence in a document

means that the document is relevant to the query.

Term Weighting

Higher weight = greater impact on cosine

Want to give more weight to the more "important" or useful terms

What is an important term?

If we see it in a query, then its presence in a document

means that the document is relevant to the query.

Term Frequency (tf) factor

$$tf_{i,j} = \frac{f_{i,j}}{\max_j f_{i,j}} \qquad tf_{i,j} = 1 + \log f_{i,j}$$

$$tf_{i,j} = 0.5 + \frac{0.5 \times f_{i,j}}{\max_j f_{i,j}} \qquad tf_{i,j} = K + \frac{(1-K) \times f_{i,j}}{\max_j f_{i,j}}$$

Inverse Document Frequency (idf) factor

$$idf_t = \log(1 + \frac{N}{n_t})$$

$$idf_t = \log(\frac{N - n_t}{n_t})$$

N = # documents in coll

nt = # documents containing term t

tf-idf weighting

A weighting scheme where

$w_{d,t} = tf_{d,t} \times idf_t$

Implementation VSM

$$sim(q,d) = \frac{1}{W_q W_d} \sum_t w_{q,t} \times w_{d,t}, W_d = \sqrt{\sum_t w_{d,t}^2}$$

Wq is the same for all documents

wq,t and wd,t can be accumulated as we process the inverted lists

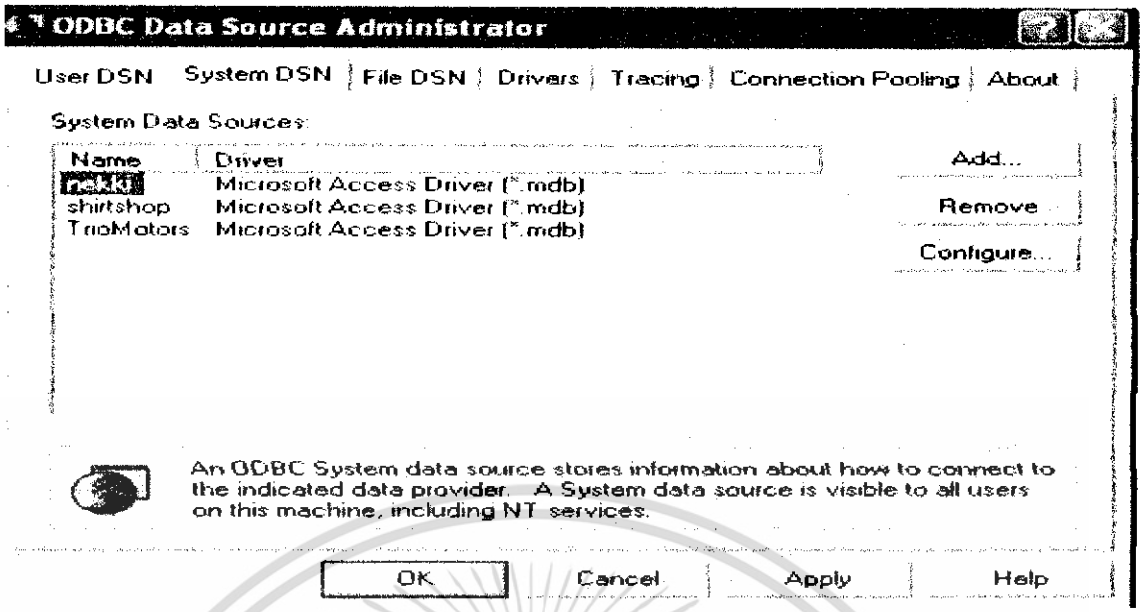
Wd can be precomputed

หลังจากที่ได้เข้าใจหลักการส่วนใหญ่แล้ว จึงสรุปขั้นตอนของการคำนวณ VSM Algorithm ให้เข้าใจง่ายๆ ได้ดังนี้

- แบ่งคำทั้งหมดออกเป็น Vector of Term ในแต่ละเอกสาร
- ตัดคำที่ไม่มีความหมาย(Stop word) ทั้งหมดออกจาก เอกสาร เช่น is, at ,are , am
- เปลี่ยนคำที่มีความหมายคล้ายคลึงกันเป็นคำที่เป็นราก (Stemming)
- ทำการคิดค่า TF, IDF และ WD(TFxIDF)
- นำค่าที่ได้ในแต่ละเอกสาร มาคำนวณสูตร Cosine Similarity Measure (sim)
- ยิ่งเอกสารได้ค่า Sim ใกล้ 1 ที่สุดแสดงว่าเอกสารนั้นยิ่งคล้ายกับสิ่งที่ต้องการ
- ถ้าเอกสารได้ Sim = 0 แสดงว่าเอกสารนั้นไม่ตรงเลย

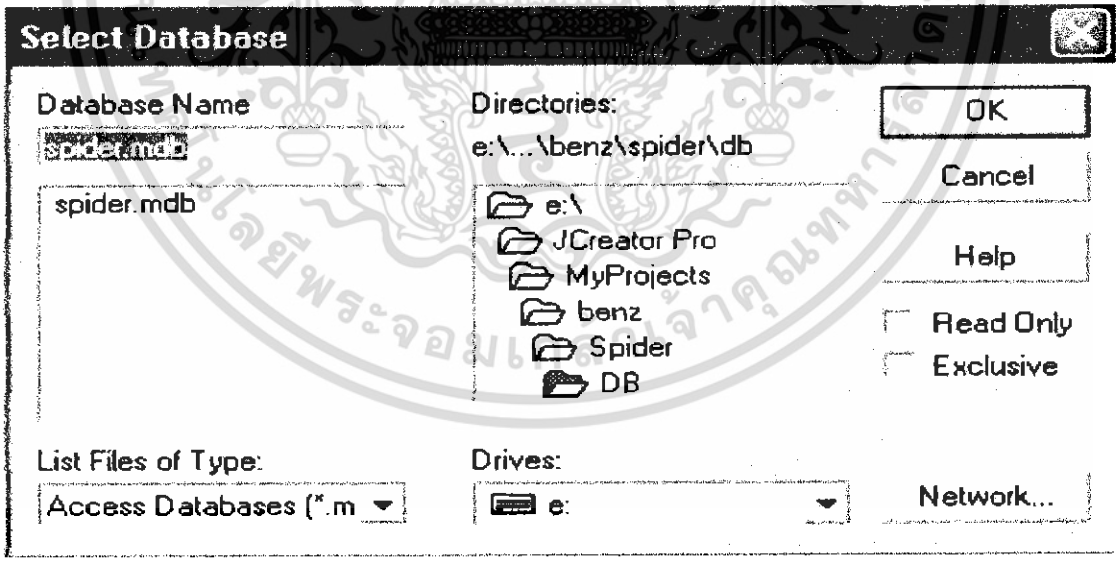
2.8 เทคนิคในการ ทำ ODBC-Less-Connection

จากในหัวข้อการ Set ODBC เพื่อให้ JAVA สามารถติดต่อกับ Database MS Access ได้นั้น ต้องอาศัย ODBC เป็นตัวช่วยในการเชื่อมต่อ เพื่อให้ JAVA สามารถเรียกใช้ API Library ของ Database ได้ ซึ่งก่อนที่จะใช้งานได้นั้น User ต้องทำการ Set DSN เพื่อเปิดการเชื่อมต่อระหว่าง ODBC และ JAVA ดังภาพ



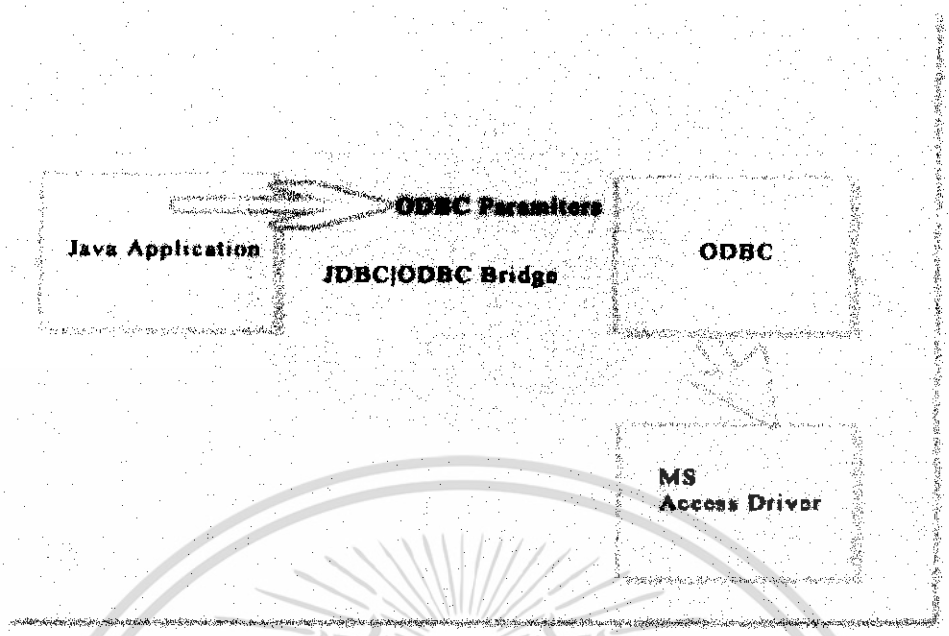
ภาพ2.10 แสดงการ Set Access ODBC Driver 1

User ต้องกำหนด DSN NAME ให้ตรงกับที่เขียนติดต่อไว้ใน โปรแกรม มิฉะนั้น จะไม่สามารถใช้งาน หรือ ติดต่อ ฐานข้อมูลดังกล่าวได้ ดังที่กล่าวมานั้น ผู้พัฒนาจะต้องสอน และ อธิบายวิธีการ Set DSN ให้กับ USER ทุกคน ซึ่งเป็นเรื่องที่ยากลำบาก และทำให้ USER ยุ่งยากในการใช้งาน ยิ่งไปกว่านั้น ถ้าหาก USER ไม่ใช่ Advance User แต่เป็น Basic User ที่ไม่ค่อยมีความรู้ด้านคอมพิวเตอร์ ก็อาจจะไม่เข้าใจถึงวิธีการทำได้



ภาพ2.11 แสดงการ Set Access ODBC Driver 2

ODBC-DSN-Less-Connection คือการเขียนโปรแกรมที่ ติดต่อ ODBC โดยตรงโดยไม่ผ่าน DSN User ไม่ต้องมาปรับค่าหรือ Set DSN อีกต่อไป เพราะ โปรแกรมจะสามารถติดต่อ Database โดยตรงได้เลย



ภาพ2.12 แสดงการ Set Access ODBC Driver 3

วิธีการเขียนโปรแกรมแบบ DSN-Less-Connection

```
import java.sql.*;


public class lessConn
{
private static String dbsrc = "jdbc:odbc:DRIVER=Microsoft Access Driver (*.mdb); " +
"DBQ=C:/Documents and Settings/Administrator/Desktop/SpiderDatabase/test.mdb; " +
"UserCommitSync=Yes; " +
"Threads=3; " +
"SafeTransactions=0; " +
"PageTimeout=5; " +
"MaxScanRows=8; " +
"MaxBufferSize=2048; " +
"DriverId=281; " +
"DefaultDir=C:/ProgramFiles/CommonFiles/ODBC/DataSources";
//C:/ProgramFiles/CommonFiles/ODBC/DataSources
private static Connection conn;
private static Statement stmt;
```

```
private static ResultSet results;

public static void main(String[] args)
{
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

conn = DriverManager.getConnection(dbsrc, "", "");

stmt = conn.createStatement();
results = stmt.executeQuery("SELECT * FROM id");

while(results.next())
{
System.out.println(results.getString("id"));
System.out.println(results.getString("name"));
}
}
catch(Exception e)
{
e.printStackTrace();
}
finally
{
try
{
results.close();
stmt.close();
conn.close();
}
catch(SQLException e)
```

```
{
e.printStackTrace();
}
}
}
}
```

• รูปด้านล่างแสดงส่วนที่ทำหน้าที่ส่งค่า Parameters ต่างๆไปยัง ODBC บอกว่าตัว .mdb อยู่ที่ path ไหน และการปรับแต่งค่าการเชื่อมต่อต่างๆ

```
private static String dbsrc = "jdbc:odbc:DRIVER=Microsoft Access Driver (*.mdb); " +
"DBQ=C:/Documents and Settings/Administrator/Desktop/SpiderDatabase/test.mdb; " +
"UserCommitSync=Yes; " +
"Threads=3; " +
"SafeTransactions=0; " +
"PageTimeout=5; " +
"MaxScanRows=8; " +
"MaxBufferSize=2048; " +
"DriverId=281; " +
"DefaultDir=C:/ProgramFiles/CommonFiles/ODBC/DataSources";
```

• ส่วนที่ทำหน้าที่เปิดการเชื่อมต่อไปยัง ODBC โดยใช้ตัวแปรที่ได้กำหนดค่าไว้

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
conn = DriverManager.getConnection(dbsrc, "", "");
```

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

3.1 ขั้นตอนการวิเคราะห์และ ออกแบบ Spider

การออกแบบ โดเมน และ Scope

- รูปแบบ Spider เป็นอย่างไร
- Spider จะต้องเก็บข้อมูลอะไรบ้าง
- Spider จะรันบนเครื่องที่มี Resource ขนาดไหน
- Spider สามารถทำงานได้เร็วแค่ไหน

3.1.1 การออกแบบรูปแบบของ Spider

โครงงานนี้จัดทำขึ้นมาเพื่อจุดประสงค์หลักคือ การเก็บข้อมูลบนเว็บด้วย AI ซึ่งตัวSpider จะต้องสามารถทำการ Indexing ได้ด้วยตัวเองและยังสามารถ Download Page มาเก็บไว้ นอกจากนี้ยังสามารถ แยกข้อมูลต่างๆที่ได้มาไว้ใน ฐานข้อมูลเพื่อจัดเตรียมให้ Search Engine ใช้งานได้อีกด้วย สรุปคือ Spider จะเป็น Spider แบบ All-in-One ที่สามารถทำงานได้ด้วยตัวเอง ทั้งหมด

3.1.2 การออกแบบการจัดเก็บข้อมูล

ข้อมูลที่ต้องการเก็บได้แก่

- MetaTag Data
- Text Content
- URL
- Referrer URL
- วันที่และเวลาที่ได้ทำการ Crawler Page นี้มา
- วันที่ที่ Page นี้ได้อัพเดทล่าสุด
- Depth (ระดับความลึก) ที่เจอ Page นี้

3.1.3 Spec ของเครื่องที่ต้องการ

จริงๆแล้ว Spider หรือ Bot มักจะเป็นงานที่ไปรันบนเครื่อง Mainframe ที่มีความเร็วสูงเพื่อสามารถประมวลผลได้เร็ว แต่ Spider ของโครงงานนี้ สามารถรองรับการทำงานบน เครื่อง Desktop ธรรมดาได้ด้วย แต่เนื่องจากการทำงานแบบ Multi Thread นั้นจะกินทรัพยากรค่อนข้างมาก จึงต้องการเครื่องที่มี Main Memory มาก

Minimum Spec

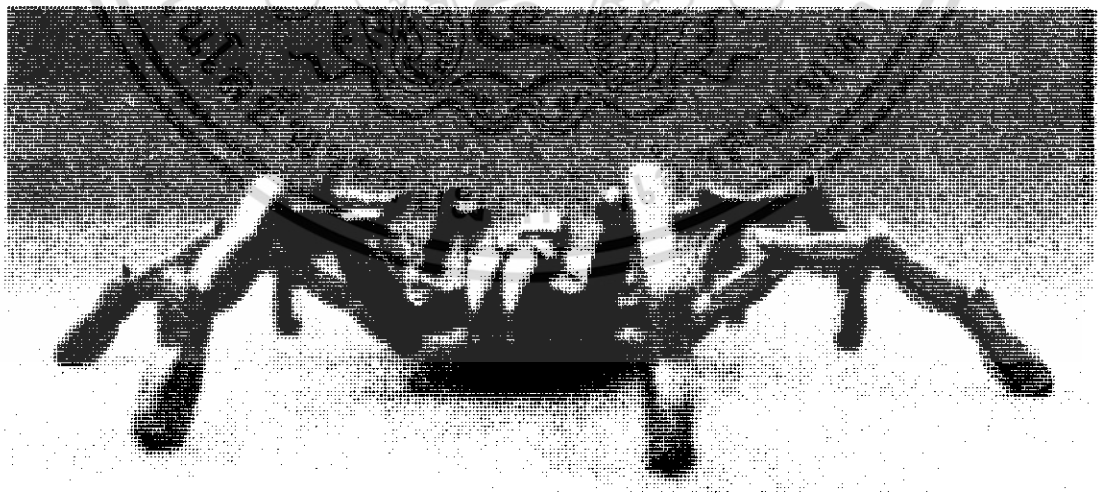
- CPU Intel or AMD 1.5GHz
- RAM 512MB
- HDD 10G
- Mainboard VIA chipset
- Video Graphic Card
- Cooling System

Recommended Spec

- CPU Intel of AMD 2.0GHz +
- RAM 1024MB
- HDD 30G
- Mainboard NForce chipset
- Video Graphic Card
- Cooling System

3.1.4 Spider จะทำงานได้เร็วแค่ไหน

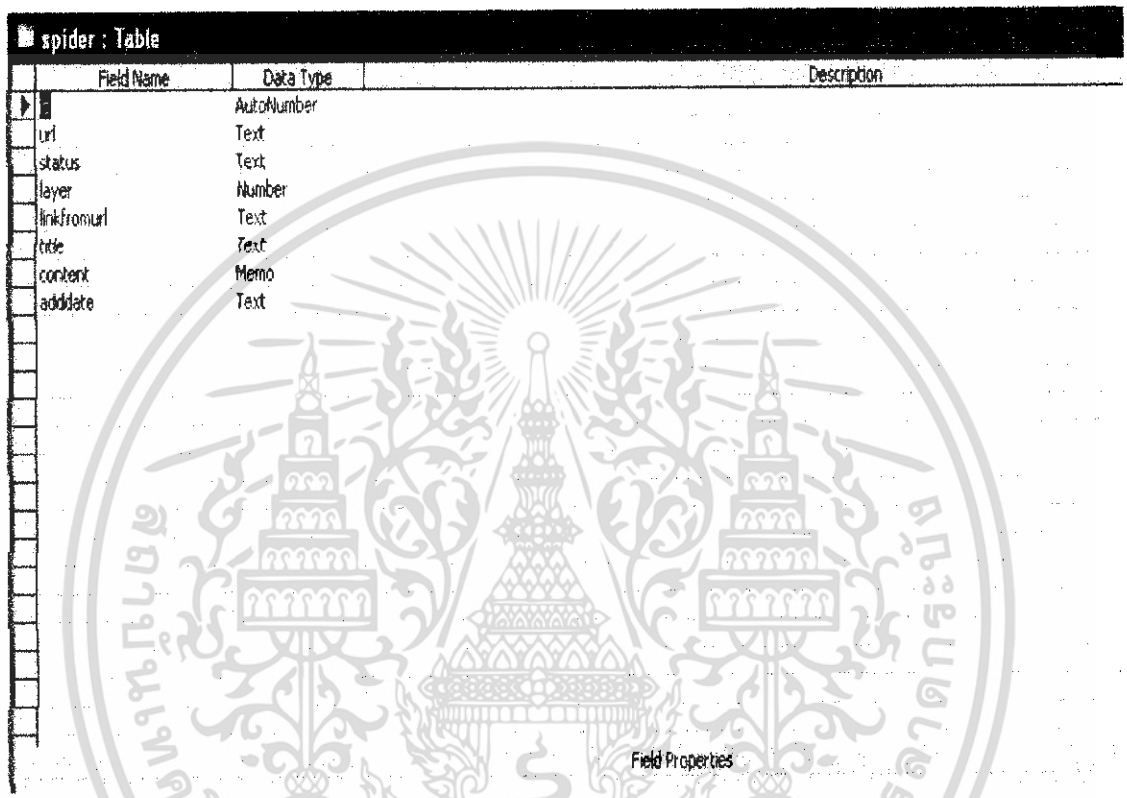
Spider จะสามารถทำงานด้วยความเร็ว 20Pages/Sec เป็นอย่างต่ำ ความเร็วที่ว่านี้คือ ความเร็วที่เข้าเปิดเพจ, วิเคราะห์เพจ, ดาวน์โหลดเพจมาเก็บไว้ และแยก ฟิลด์ ลง ฐานข้อมูล



3.2 ขั้นตอนการลงมือปฏิบัติจริง

3.2.1 สร้างฐานข้อมูลที่จะเก็บข้อมูลลง

ในโครงงานนี้ใช้ Microsoft Access เป็น Database ในการเก็บข้อมูลลงฐานข้อมูล โดยข้อมูลนี้จะมีการแยก Field, Content และ URL ให้เรียบร้อย สำหรับให้สามารถ Search Engine นำไปใช้งานต่อได้



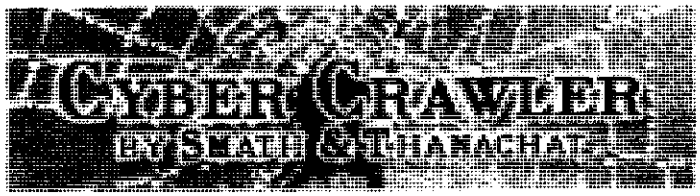
ภาพ3.1 แสดงโครงสร้างฐานข้อมูล Access

Column Data

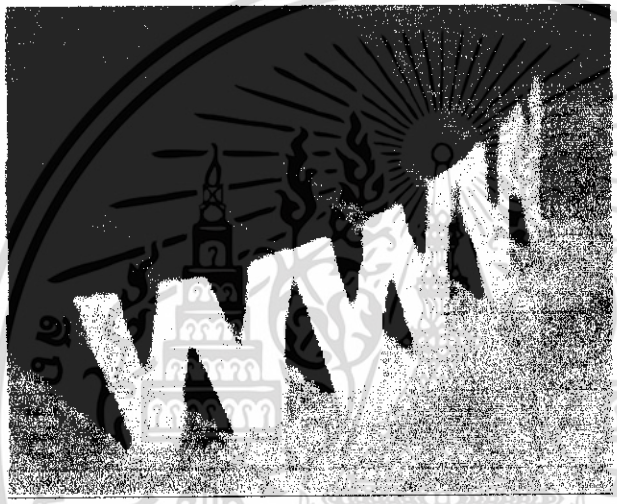
- **id:** เป็น Unique Number สำหรับแต่ละ Record
- **url:** URL address ของเพจนี้
- **status:** แสดง Status ของเพจนี้
- **linkfromurl:** แสดงว่าถูก Link มาจากเพจใด
- **title:** หัวข้อเรื่องของเพจ
- **content:** Text Content ของเพจ
- **adddate:** วันที่เพิ่มเพจนี้เข้ามา

3.2.2 ทำภาพและตกแต่งกราฟฟิกต่างๆที่ต้องใช้ใน GUI

- Banner Logo

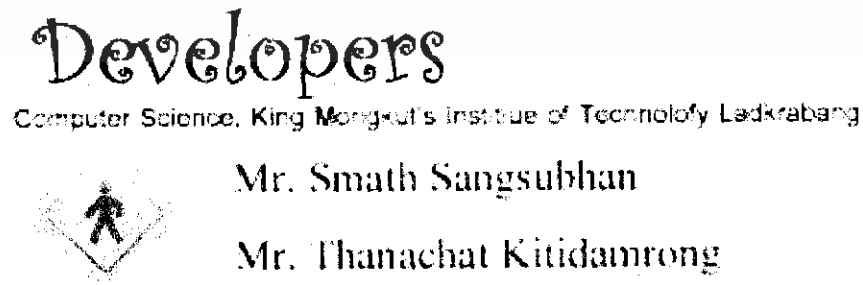


- กราฟิกตกแต่ง

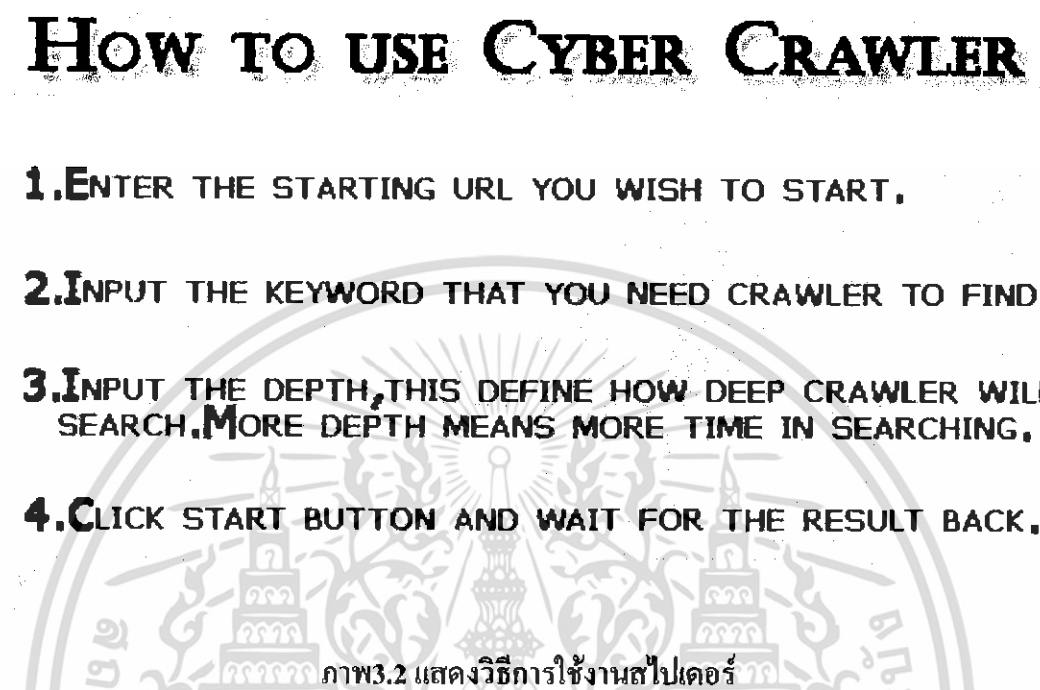


- Button ต่างๆ

- รายชื่อผู้จัดทำ



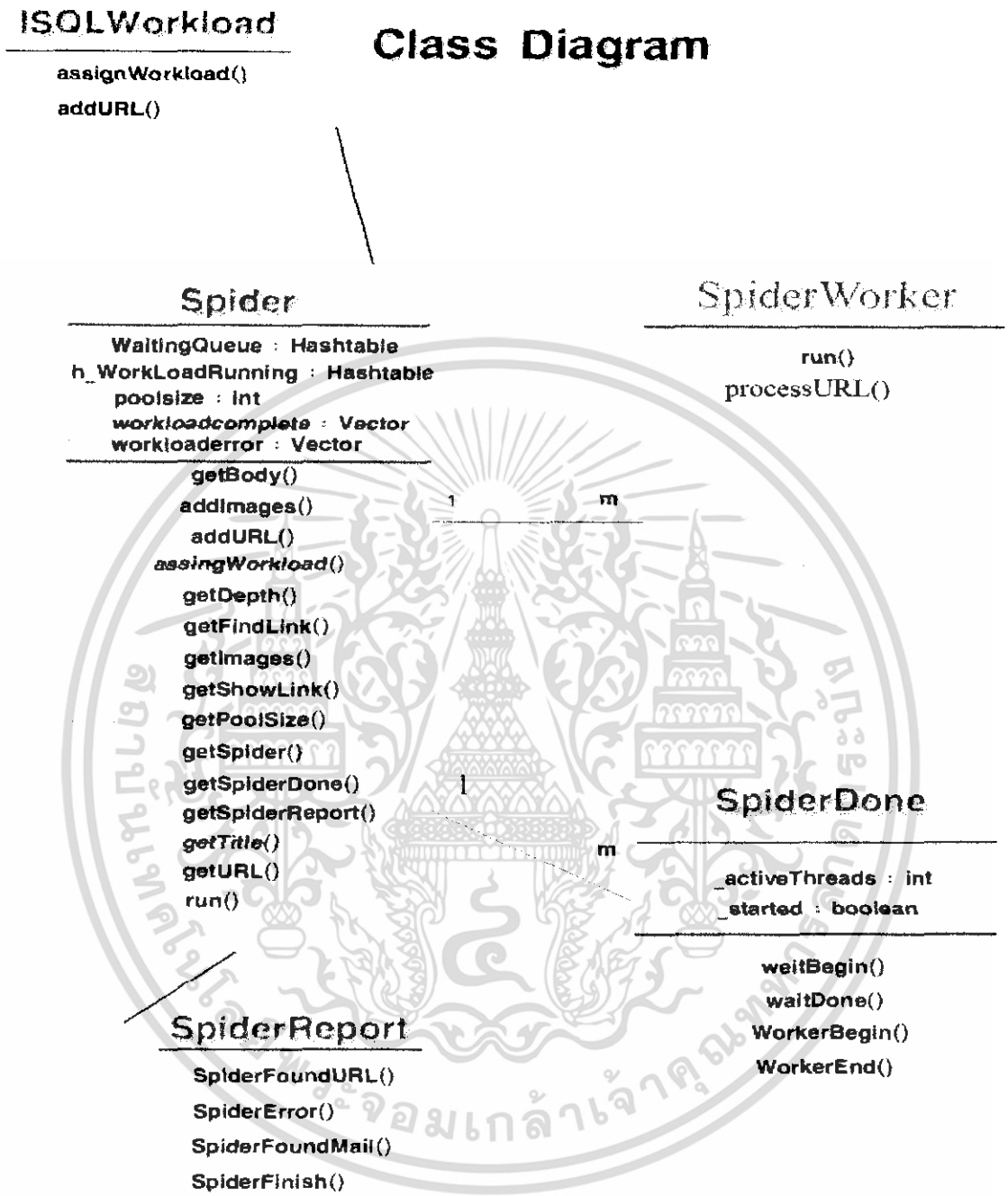
● Help & How to use



ภาพ3.2 แสดงวิธีการใช้งานสไปเดอร์

ภาพ3.3 แสดงคำถามที่มักถูกถามบ่อย

3.3 ขั้นตอนการเขียนโปรแกรม

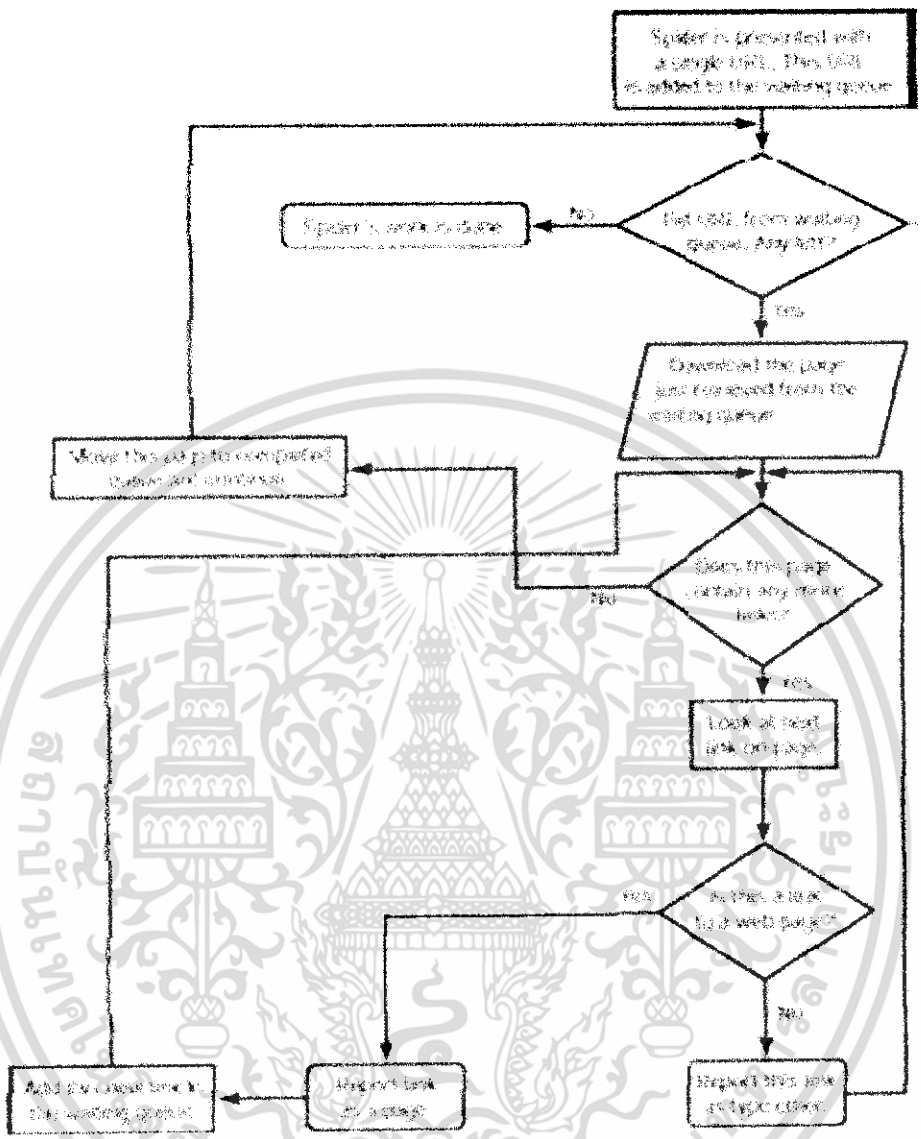


ภาพ3.4 แสดง Spider Class Diagram

3.3.1 ส่วนประกอบของ Class Diagram

- Class Spider: เป็น Classs ที่ดูแลการโหลด Content ต่างๆ
- Class SpiderWorker: มีหน้าที่คอยสั่งให้ Spider ทำงาน
- Class SpiderDone: ควบคุมเกี่ยวกับการจบการทำงาน
- Class SpiderReport: ใช้ในการแสดง Report ข้อมูล

3.3.2 Flowchart การทำงานของโปรแกรม



ภาพ3.5 แสดง Spider Flowchat

3.3.3 Spider Algorithm

Get the user's input: the starting URL and the desired file type. Add the URL to the currently empty list of URLs to search.

While the list of URLs to search is not empty,

{

Get the first URL in the list.

Move the URL to the list of URLs already searched.

```
  Check the URL to make sure its protocol is HTTP
    (If not, break out of the loop, back to "While").
  See whether there's a robots.txt file at this site that includes a "Disallow" statement.
    (If so, break out of the loop, back to "While".)
  Try to "open" the URL (that is, retrieve that document from the Web).
  If it's not an HTML file, break out of the loop,back to "While."
  Step through the HTML file.
While the HTML text contains another link.
  {
    Validate the link's URL and make sure robots are allowed (just as in the outer loop).
    If it's an HTML file,
    If the URL isn't present in either the to-search list or the already-searched list, add it to the
to-search list.
      Else if it's the type of the file the user requested,
        Add it to the list of files found.
  }
}
```

อธิบายการทำงาน

1). เริ่มแรก ให้ user กำหนดจุดเริ่มต้นของ url ว่าต้องการจะให้ Spider เริ่มต้นการ Crawler จากที่เพจไหน

2).Spider จะทำการ Add Page นั้นเข้าคิว และทำการประมวลผลเพจนั้น (Parsing) ว่ามีลิงค์ไปที่ใดบ้าง และนำเข้าคิวต่อไปโดยมีเงื่อนไขดังนี้

เงื่อนไขการทำงานและเช็คความถูกต้อง

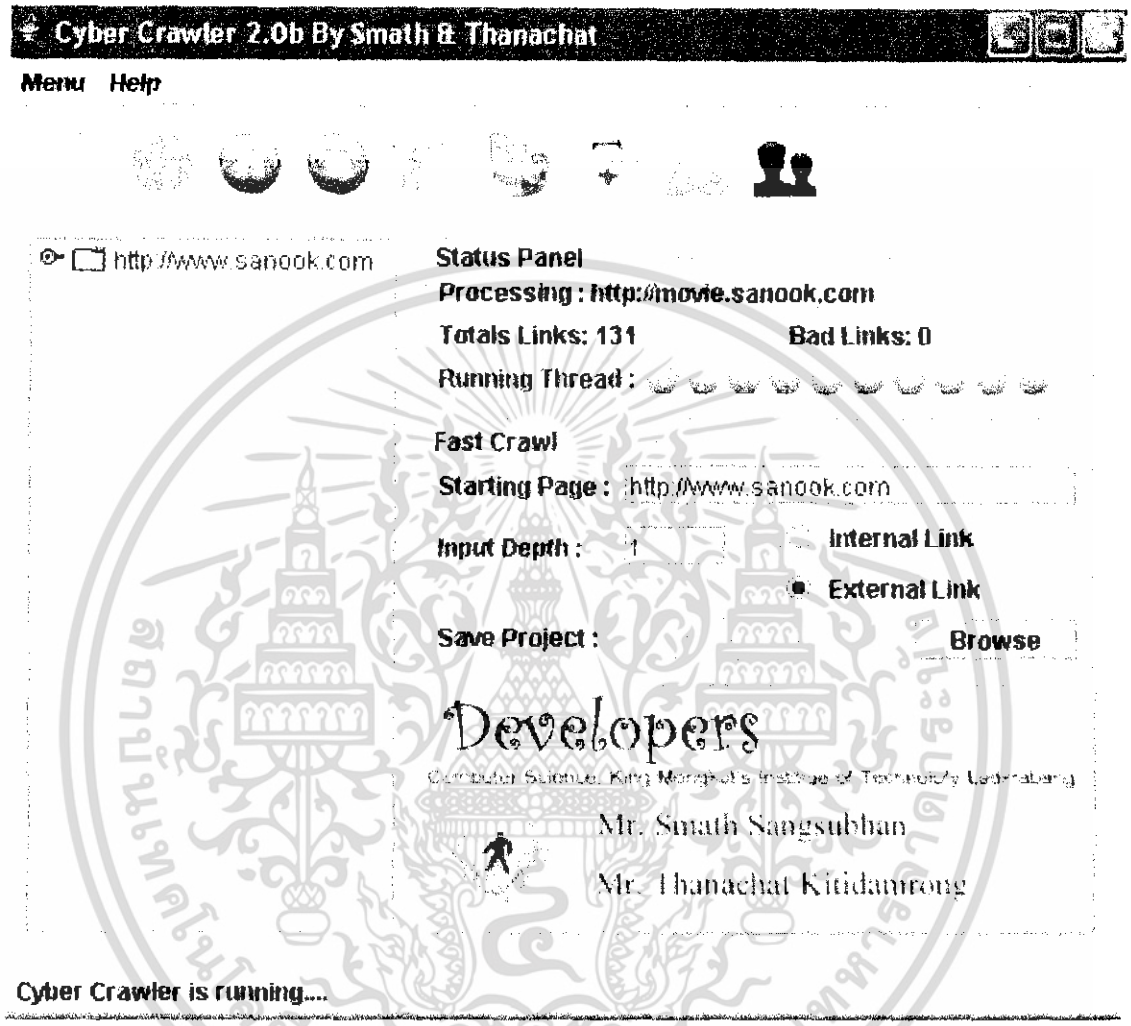
- **Page นั้นต้องไม่อยู่ในคิว (ป้องกันการประมวลผลซ้ำเพจเดิม)**
- **เป็น Protocol http**
- ***ผ่านข้อตกลงของ* Robot Exclusion Standard**

3).ต่อจากนั้น Spider ก็จะดึงเพจที่อยู่ในคิวออกมาทำต่อไปเรื่อยๆจนหมด

3.3.4 *Main Application*

หน้าจอหลักของ Spider จะมีลักษณะและส่วนประกอบดังนี้

(หมายเหตุ – โปรแกรมในรูปเป็น version 2.0beta)



ภาพ3.6 แสดงหน้าจอ Interface ของ Main Application

ส่วนประกอบต่างๆได้แก่

- **Enter Spider Input:** สำหรับใส่ URL แรกที่จะส่ง Spider ไป
- **Keyword:** สำหรับกรองเพจที่จะเก็บเพื่อเอาเฉพาะสิ่งที่สนใจ
- **Enter Depth:** ใส่ระดับความลึกที่ต้องการให้ Clawler
- **Total Links:** แสดงจำนวน Page ทั้งหมดที่ได้จากการ Crawler
- **Bad Links:** แสดงจำนวน Link ที่เป็น Broken Link
- **Running Thread:** แสดงจำนวน Thread ที่กำลังทำงานอยู่

3.4 การลงมือเขียนโปรแกรม

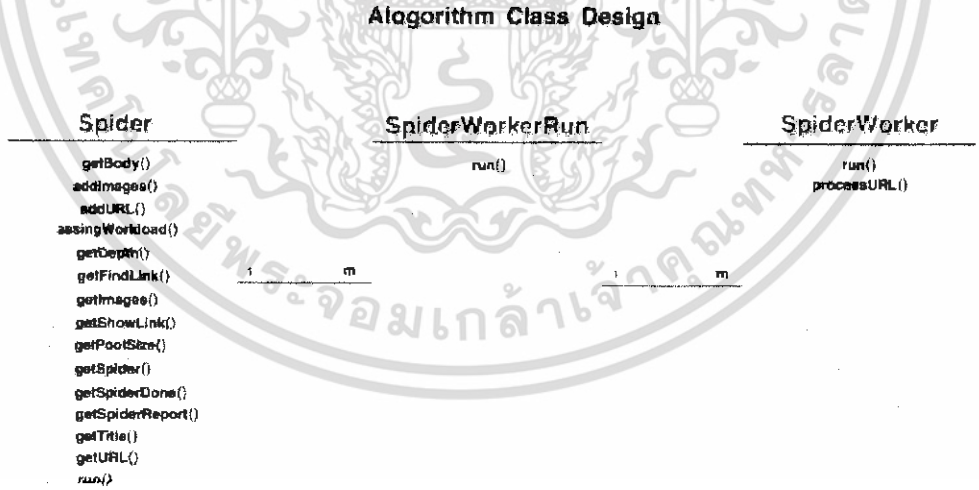
3.4.1 Algorithm การกำหนดลำดับชั้น (Depth) กับการทำงานแบบ MultiThread

การกำหนดลำดับชั้นการทำงานของโปรแกรมที่ทำงานแบบ MultiThread จะมีหลักการคือให้แต่ละชั้นทำการสร้าง Thread เพื่อทำงานเฉพาะภายในชั้นนั้นๆ เลย เพื่อที่จะสามารถควบคุมการประมวลผลในแต่ละชั้นได้ และ จะมีการสร้าง Hashtable ขึ้นมาตามจำนวนชั้น (Depth) ที่ต้องการ โดย Hashtable แต่ละอัน จะทำหน้าที่เก็บ url ที่สามารถหาได้ในแต่ละชั้น และ เก็บลำดับชั้นที่ url นั้นๆอยู่ โดยการเก็บจะมี Key เป็น url และ มี Value เป็นค่าลำดับชั้นที่

3.4.2 วิธีการกำหนดลำดับชั้น

ใน WaitingQueue จะสร้างเป็น Hashtable เพื่อเก็บ url เป็น key และจะเก็บลำดับชั้น(layer) ของ url นี้เป็นค่า value เมื่อ url นั้นพร้อมจะถูกประมวลผลก็จะดึง url ออกจาก WaitingQueue และย้ายไปเก็บใน RunningQueue จากนั้นก็จะทำการ process เพื่อหา url ใหม่ที่อยู่ใน page นี้ออกมา จากนั้นเราจะดึงค่า layer ของ url ที่กำลังถูก process นั้นไปบวกเพิ่มขึ้น 1 เพื่อเป็นค่า layer ของ url ใหม่ที่ Collect ได้ จากนั้นจึงทำการ add url ใหม่ และ ค่า layer ที่สามารถหาได้เก็บลงใน WaitingQueue

และจะต้องมีการ Check ด้วยว่า layer ใหม่ที่หาได้นั้น <= depth เพื่อให้โปรแกรมไม่ add url ที่มีลำดับชั้นสูงกว่านี้ลงใน WaitingQueue เพื่อรอการ process



ภาพ3.7 แสดงการทำงานของคลาสต่างๆ

ถ้ามี : แต่ละ Thread นั้นก็จะวิ่งเข้าไปดึง url นั้นพร้อมๆ กัน โดย Thread ตัวใดที่ไปถึงก่อนก็จะได้ url นั้นออกมาประมวลผลและ Thread ที่ว่างอยู่ก็จะรอให้มีการ add url เข้ามาใหม่เพื่อที่จะวิ่งเข้าไปดึงมาประมวลผลเป็นแบบนี้เรื่อยๆ จนกระทั่งมีการสั่งให้หยุดโปรแกรมหรือโปรแกรมได้ทำงานจนถึงระดับชั้นที่ได้กำหนดไว้เรียบร้อยแล้ว

ถ้าไม่มี : Thread นั้นก็จะรอจนกว่าจะมีการ Add url ใหม่ลงไปใน WaitingQueue ของชั้นที่ Thread นั้นทำงานอยู่ และ จากนั้นค่อยดึง url นั้นออกมาประมวลผล

3.4.4 อธิบายการทำงานของ Source Code โปรแกรม (เฉพาะบริเวณสำคัญ)

3.4.4.1 Spider.java

```
public Hashtable[] WaitingQueue;
protected SpiderWorkerRun spider_workerrun[];
public Spider(FindLink findlink,SpiderReport report,int poolsize,String url,int depth)
{
        this.findlink = findlink;
        this.report = report;
        this.poolsize = poolsize;
        this.depth = depth;

      WaitingQueue = new Hashtable[depth+1];

        // ---------- Input 2 Layer ,It's must Create 2 Waiting Queue ----------
        for(int j=0;j < WaitingQueue.length;j++)
        {
                WaitingQueue[j] = new Hashtable();      //Create Array Of Hashtable
        }

        // ---------- Create Pool To Work In Depth ----------
        // ---------- So each Object have there own layer value ----------
        spider_workerrun = new SpiderWorkerRun[depth];
        for(int i=0;i<spider_workerrun.length;i++)
        {
```

```
                spider_workerrun[i] = new SpiderWorkerRun(this,i);// i is layer
            }

            // ---------- Add First URL In Waiting Queue ----------
            if(url.toString().length()>0)
            {
                addURL(url,url);
            }
        }

        public void run()
        {
            int z=0;
            if(_halted)
            return;

                // ---------- Process Thread In Each Layer Parallel ----------
                for(int i=0;i<depth;i++)
                {
                    spider_workerrun[i].start();
                }
        }
```

SpiderWorkerRun.java

```
        protected SpiderWorker pool[];
        protected int poolsize;

        public SpiderWorkerRun(Spider spider,int layer)
        {
            this.poolsize = spider.getPoolSize();
```

```
                // ---------- Create Pool Of Each Layer ----------
                pool = new SpiderWorker[poolsize];
                for(int i=0;i<pool.length;i++)
                {

                        pool[i] = new SpiderWorker(spider,layer);

                }
        }

        public void run()
        {
                        for(int i=0;i < pool.length;i++)
                        {
                                pool[i].start();
                        }
        }
```

3.4.4.2 SpiderWorker.java

```
public SpiderWorker(Spider spider,int layer)
        {
                this.spider = spider.getSpider();
                this.report = spider.getSpiderReport();
                this.keyword = spider.getFindLink().keyword.getText();
                this.depth = spider.getDepth();
                this.layer = layer;
        }
public void run()
```

```
{
        for(;;)
        {
                String url = spider.getURL(layer);

                if(url==null)
                return;

                try
                {

                        spider.getSpiderDone()[layer].workerBegin();
                        processURL(new URL(url)); // Collect URL In This Page
                        spider.getSpiderDone()[layer].workerEnd();

                }catch(MalformedURLException e){}

                report.SpiderUpdateQueue(spider.h_WorkLoadRunning.size());

        }
}
```

3.4.4.3 อธิบาย Source Code โปรแกรมเพิ่มเติม

ใน Constructor ของ Class Spider จะมีการวน loop สร้าง Object ของ Class SpiderWorkerRun ตามระดับชั้นความลึกนั่นหมายความว่าสร้าง Thread ขึ้นมาควบคุมการทำงานของแต่ละชั้น และ ใน Method run() จะเรียกให้ Object ของ Class SpiderWorkerRun แต่ละตัวทำงาน

ใน Constructor ของ Class SpiderWorkerRun จะวน loop ทำการสร้าง Pool ซึ่งเป็น Object ของ Class SpiderWorker เพื่อให้มีหลายๆ Pool ที่สามารถ Process url ที่พบในชั้นนั้นๆ ได้ และใน Method run() ของ Class SpiderWorkerRun จะเรียกให้ Object ของ Class SpiderWorker แต่ละตัวทำงานซึ่งจะเป็นการทำงานแบบขนาน

สังเกตว่าใน Class Spider จะส่ง this เป็น parameter ในการสร้าง Object ของ Class SpiderWorkerRun (*ไม่สร้าง Object ใหม่*) เพราะว่าจำเป็นต้องใช้ Queue ร่วมกันเพื่อที่จะได้ add และ remove url เพราะฉะนั้นจึงจำเป็นต้องใช้ Object ของ Class Spider ตัวเดียวกัน จึงส่ง this เป็น parameter นั่นเอง และใน Method processURL() จะทำ หน้าที่ process url ปัจจุบันที่ดึงออกมาจาก WaitingQueue

หมายเหตุ : เนื่องจากมีการสร้าง Object ของ Class SpiderWorker ใหม่ ใน Constructor ของ Class SpiderWorkerRun เพราะฉะนั้นแต่ละ Threadจะมี Method processURL() เป็นของตัวเอง ซึ่งจะเป็น Method ที่ทำการ Collect URL มาจึงทำให้ Thread แต่ละตัวทำงานแยกจากกัน แต่ตอนที่จะดึง url ออกมาจาก WaitingQueue ตรงส่วนนั้นเราจำเป็นต้องใช้ Object ของ Class Spider ตัวเดียวกัน Thread จึงต้องรอกัน ณ เวลาที่เรียก Method getURL() ซึ่งเป็น Method() ที่มีการ synchronized เอาไว้

จากตัวอย่างที่ผ่านมาได้ศึกษาเกี่ยวกับการทำ MultiThread ซึ่งสามารถกำหนดระดับชั้นได้แล้ว แต่ปัญหาที่เกิดขึ้นมาในภายหลังก็คือ การที่เราใช้ Hashtable และ Vector ในการเก็บข้อมูลที่มีขนาดมากๆ

3.4.5 ปัญหาจากการใช้ Hashtable และ Vector

ข้อมูลต่างๆ ที่ถูกเก็บโดย Hashtable และ Vector นั้นจะถูกเก็บอยู่ในหน่วยความจำ (RAM) ซึ่งหากมีการเก็บข้อมูลมากกว่า 10,000 url ขึ้นไป อาจจะทำให้เกิดปัญหา Out Of Memory ได้ เนื่องจากหน่วยความจำได้ถูกใช้จนหมด วิธีการแก้ไขเพื่อทำให้โปรแกรมมีประสิทธิภาพได้นั้นคือ การเปลี่ยนมาเก็บข้อมูลต่างๆ ที่สามารถ Collect ได้ลง Database แทน

3.4.6 JAVA กับฐานข้อมูล Access

เมื่อโปรแกรมภาษา JAVA ต้องการติดต่อกับระบบฐานข้อมูลใด ก็จะต้องมี driver สำหรับระบบฐานข้อมูลนั้น เพื่อทำหน้าที่เชื่อมต่อระหว่างโปรแกรมกับฐานข้อมูล แต่ระบบฐานข้อมูลมีความแตกต่างกันไปในแต่ละยี่ห้อ ดังนั้น driver ที่สร้างขึ้นสำหรับติดต่อกับ JDBC จึงถูกสร้างขึ้นด้วยวิธีที่แตกต่างกัน

ODBC (Open Database Connectivity) เป็น API ที่ถูกใช้อย่างแพร่หลาย สามารถติดต่อกับระบบฐานข้อมูลได้หลายยี่ห้อ และสามารถใช้ได้หลาย platform แต่สาเหตุที่ไม่เขียนโปรแกรมเพื่อเรียกใช้ ODBC ตรงๆ จากภาษา JAVA เนื่องจากว่า ODBC ถูกเขียนขึ้นด้วยภาษา C จึงมีปัญหาในด้านความปลอดภัยของข้อมูลและความเข้ากันได้ของโปรแกรม ดังนั้นจึงมี JDBC-ODBC Bridge เพื่อเป็นสะพานในการเชื่อมต่อการทำงานจากโปรแกรมภาษา JAVA ไปยัง ODBC Driver

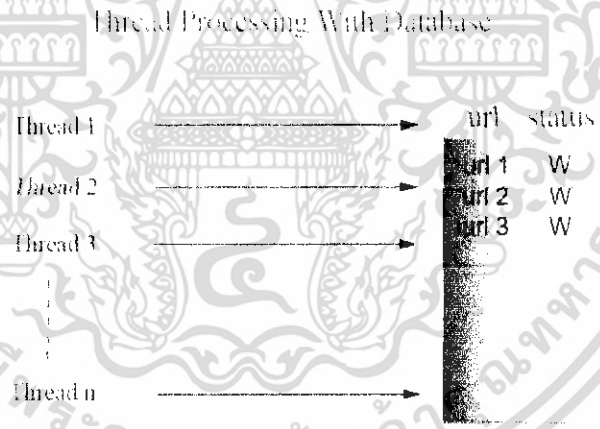
ในตัวอย่างนี้จะใช้ Database ของ Access ในการเก็บข้อมูล โดยภาษา JAVA สามารถติดต่อกับฐานข้อมูล Access ได้ผ่านทาง jdbc:odbc Bridge

3.4.7 Algorithm การทำงานของ Thread ในโปรแกรม

Thread จะไปดึง url ที่มีสถานะ (status) เป็น "W" (Waiting) ในฐานข้อมูลออกมาและเมื่อดึงออกมาแล้วจะทำการเปลี่ยนสถานะของ url นั้นเป็น "R" (Running) ทันที เพื่อเป็นการบอกว่า url นั้นกำลังถูก process อยู่

เมื่อทำการ process url นั้นเสร็จเรียบร้อยแล้ว ก็จะทำการเปลี่ยนสถานะของ url นั้นเป็น "C" (Complete) หรือ "E" (Error, ในกรณีที่ url นั้นเกิด Error) เพื่อเป็นการบอกว่า url นั้นได้ผ่านการ process ไปแล้ว เมื่อมีการพบ url ใหม่ ก็จะทำการ add url ใหม่นั้นลงใน Database โดยเราจะ Set status เป็น "W" เพื่อรอการ process

โดยการทำงานก็ยังคงใช้หลักการเดิมซึ่งก็คือ จะมี Thread หลายๆ ตัวที่คอย Check ค่าที่อยู่ในฟิลด์ status ในตาราง spider โดยจะดึง url ที่มีค่า status เป็น "W" ออกมาเพื่อประมวลผล ดังแสดงในรูปข้างล่าง

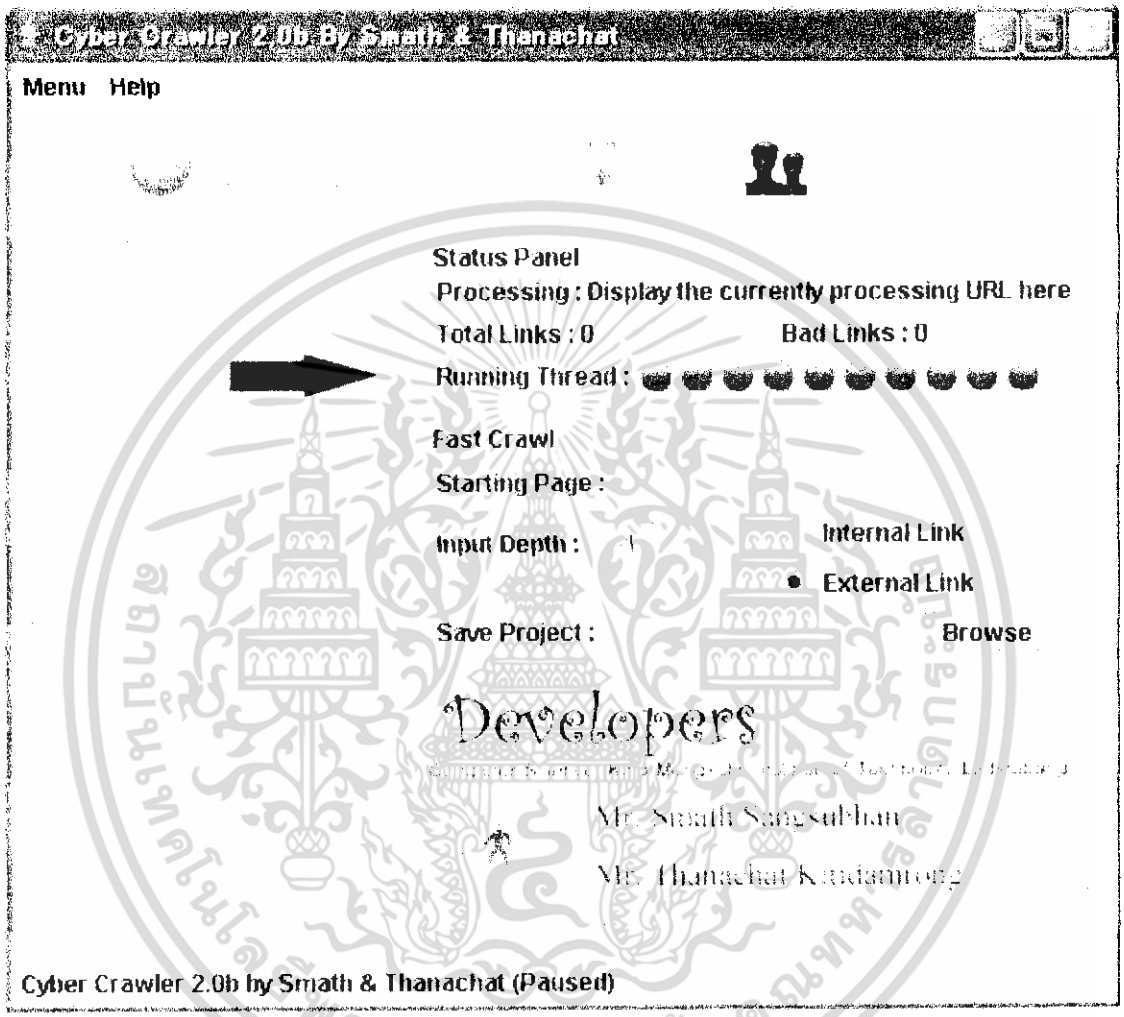


ภาพ3.9 แสดงการทำงานแบบ Multi Tread

จากรูป การทำงานของ Thread ต่างๆ จะเข้าไปดึง url ที่มีสถานะเป็น "W" พร้อมๆ กัน โดย Method ที่ทำงานในการดึง url ออกมาจากจาก Database จะต้องถูก synchronized เอาไว้ เพื่อให้ ณ เวลาเดียวกันนั้น จะมี Thread ได้เพียง Thread เดียวที่สามารถครอบครองการทำงานได้ เพื่อให้การทำงานนั้นถูกต้อง

3.4.8 การแสดงผล Running Thread ผ่าน Graphic

เนื่องจากโปรแกรม Spider นี้ใช้ การทำงานแบบ Multi Thread ซึ่งการทำงานแบบนี้ในเวลาเดียวกันจะมีการทำงานหลายงานพร้อมกัน ซึ่ง Thread แต่ละตัวก็จะทำงานที่ต่างกันไป เราจึงเพิ่ม Feature หนึ่งขึ้นมาคือการแสดงผลของจำนวน Thread แบบ Real Time ซึ่งจะแสดงผลว่าขณะนี้มีจำนวน Thread กำลังทำงานอยู่กี่ Thread



ภาพ3.10 แสดงผลจำนวน Thread ก่อนเริ่มรัน Spider

Source Code แสดงจำนวน Running Thread

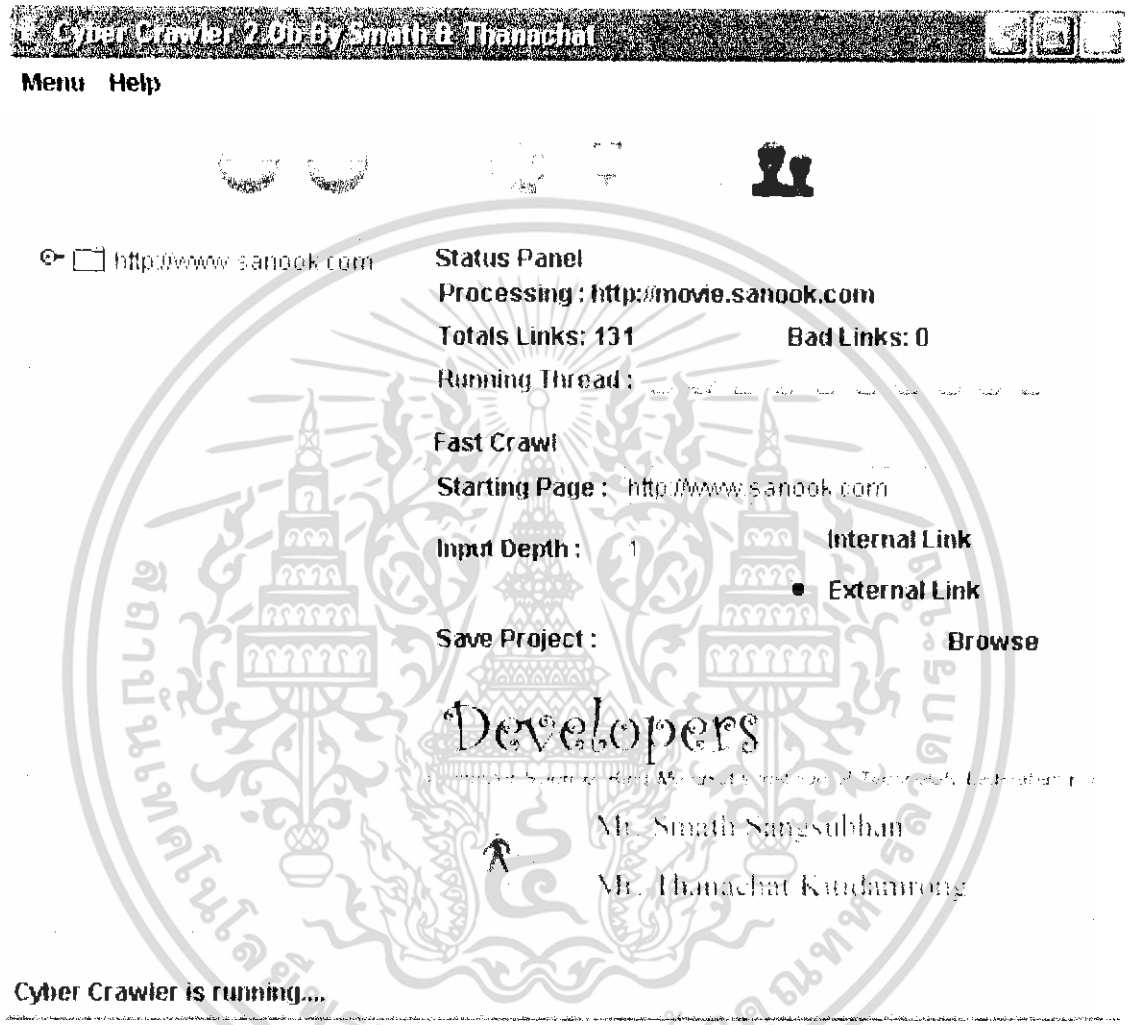
```
Private...ImageIcon...green=new
ImageIcon(getClass().getResource("/Interface_Spider/pic/label-green-16.png"));
  Private...ImageIcon...red=new
ImageIcon(getClass().getResource("/Interface_Spider/pic/label-red-16.png"));
```

● ทำการ Initial รูป Thread สีแดง และสีเขียว โดยสีแดงคือไม่ทำงาน สีเขียวคือทำงานอยู่โดยใช้ Class ImageIcon เพื่อเตรียมนำรูปไปใช้ โดยใช้ตัวแปรชื่อ red, green

```
for(rt=0;rt< t.length;rt++)
    {

      JButton button2 = new JButton(green);
      greent[rt] = button2;
      redthreadPanel.add( greent[rt] );

      JButton button = new JButton(red);
      t[rt] = button;
      redthreadPanel.add( t[rt] );

      t[rt].setBounds(red_axis,8,16,16);
      red_axis+=20;
      t[rt].setBorderPainted(false);
      t[rt].setBackground(mainColor);

      greent[rt].setBounds(green_axis,8,16,16);
      green_axis+=20;
      greent[rt].setBorderPainted(false);
      greent[rt].setBackground(mainColor);

      // ----- Hide green color -----
      greent[rt].setVisible(false);

    }
```

● ทำการสร้างรูป Thread สีแดงโดยวนลูปสร้างตามจำนวนที่ต้องการ โดยกำหนดให้แต่ละภาพห่างเป็นระยะทางเท่ากัน

●สร้าง รูป Thread สีเขียวทับลงไปที่ตำแหน่งของรูป Thread สีแดง แล้วปรับรูป Thread สีเขียวให้ Visible=False ไว้ เพื่อให้ปกติตอนที่ยังไม่มี Thread ทำงานจะแสดงภาพ Thread สีแดง และเมื่อมีการ new Thread ใหม่ขึ้นมาทำงาน ก็จะแทรกบรรทัด ที่ปรับ **greent[rt].setVisible(true);** ไปเพื่อให้ภาพ Thread สีเขียวแสดงขึ้นมา



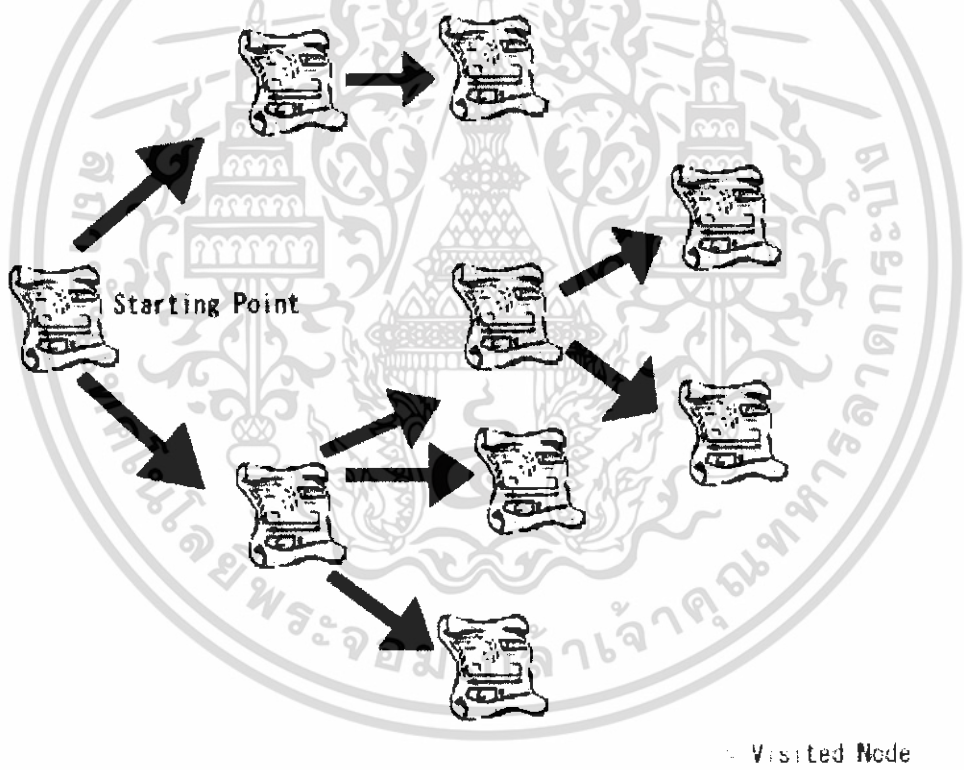
ภาพ 3.11 แสดงจำนวน Running Thread ขณะโปรแกรมกำลังทำงาน

3.4.9 เงื่อนไขการหยุดค้นหาของ Spider

เมื่อ Spider ทำการค้นหาไประยะหนึ่งแล้ว Spider จะสามารถตัดสินใจได้ว่าจะหยุดหรือจะค้นหาต่อไป ซึ่งถ้าหาก Spider ไม่เตรียมเงื่อนไขในการหยุดไว้ก็อาจจะทำให้ตัว Spider ติด Loop ได้ และทำการค้นหาต่อไปเกินความจำเป็น นอกจากนี้เมื่อ Spider รันต่อเนื่องกันเป็นระยะเวลานาน ยังทำให้ Memory ของระบบถูกใช้งานมากขึ้นเรื่อยๆด้วย จึงทำให้ต้องมีการกำหนด เงื่อนไขในการหยุดค้นหาของ Spider ขึ้นมา โดยจะหยุดเมื่อเจอเงื่อนไขต่อไปนี้

3.4.9.1 เมื่อไม่พบเว็บเพจอีกแล้ว

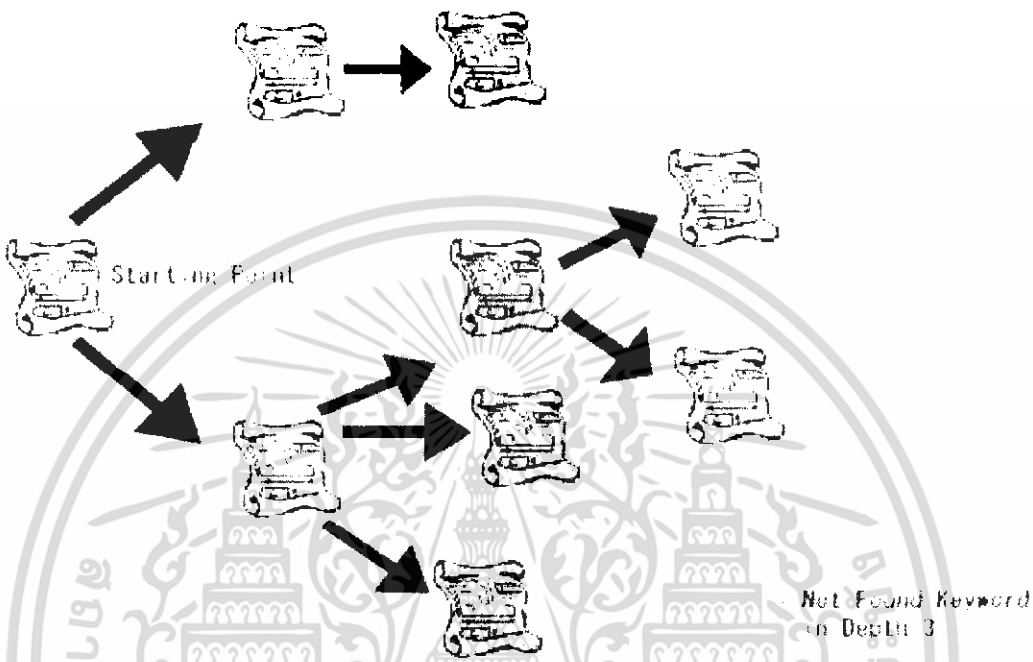
เมื่อSpider ทำการค้นหาไปเรื่อยๆแล้ว จนกระทั่งครบทุก Page หรือทุก Node แล้ว และไม่มีทาง Link ไปยังเอกสารหรือว่า Page ต่อไปแล้ว Spider จะทำการหยุดการค้นหา และคืน Memory ให้ระบบ



ภาพ3.12 แสดงเงื่อนไขการหยุดเมื่อหาครบแล้ว

3.4.9.2 เมื่อไม่พบเว็บเพจทั้งระดับชั้น

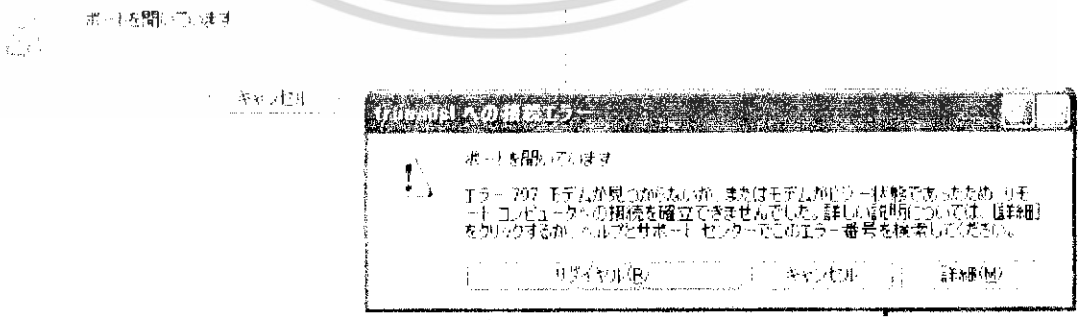
เมื่อSpider ทำการค้นหาแบบกรอง Keyword (คือการค้นหาเฉพาะเว็บที่มีคำที่ค้นหา) แล้ว เมื่อในระดับความลึกใดระดับหนึ่ง ไม่พบเว็บใดเลยที่มีข้อความที่ต้องการค้นหา (Keyword) Spider จะทำการหยุดการค้นหา และคืน Memory ให้ระบบ



ภาพ3.13 แสดงการหยุดค้นหาเมื่อไม่เจอคำที่ต้องการทั้งระดับชั้น

3.4.9.3 เมื่อเกิดปัญหากับ Connection

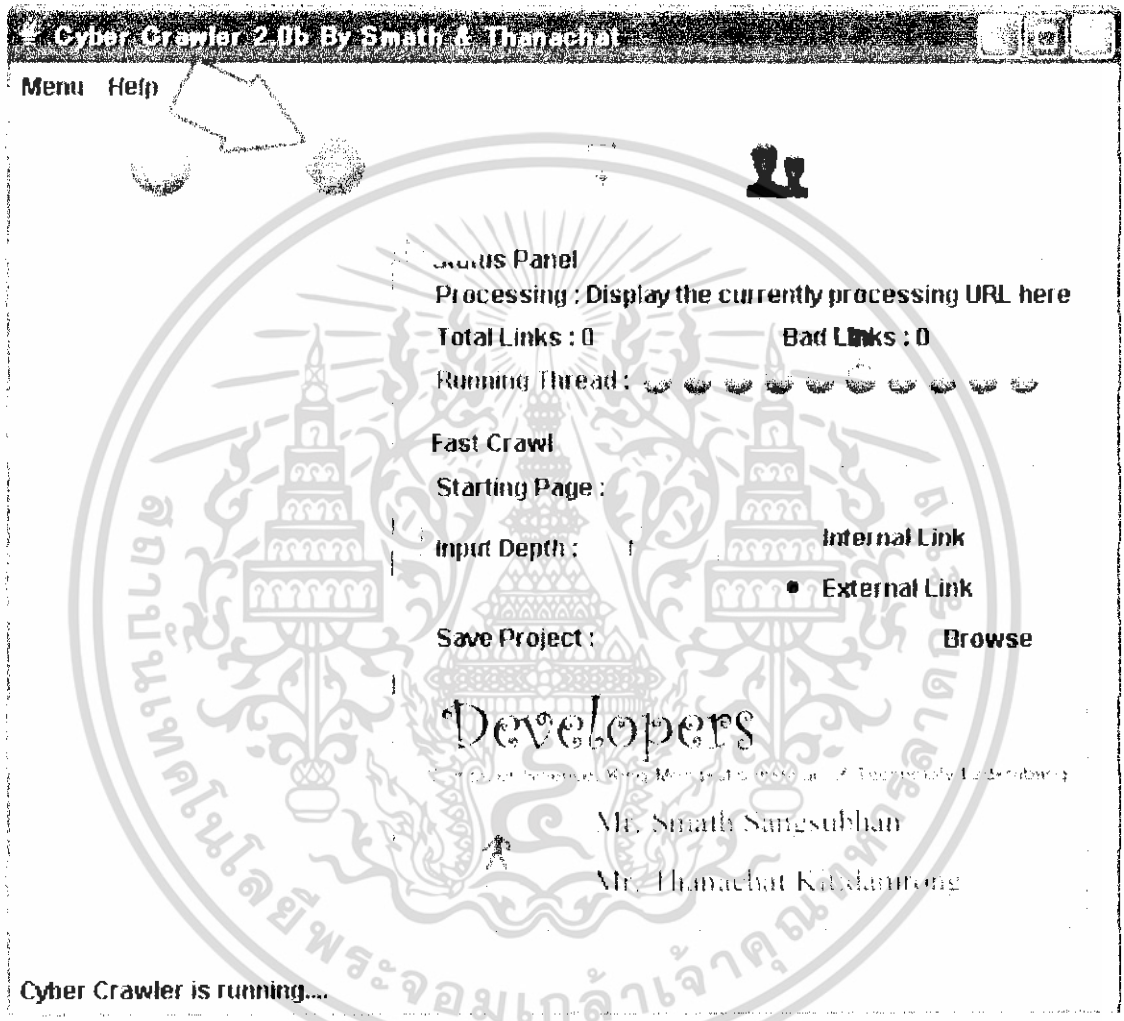
Spider จะหยุดการค้นหาเมื่อ มีปัญหาเกิดขึ้นกับ Internet Connection ของเครื่องที่รัน Spider อยู่ ซึ่งได้แก่ การเกิดการ Disconnect, Time out เป็นต้น



ภาพ3.14 แสดงการหยุดค้นหาเมื่อเกิดปัญหากับ Internet Connection

3.4.9.4 เมื่อ User กด Cancel ยกเลิกการค้นหา

อีกวิธีที่สามารถหยุดการทำงานของ Spider ได้ก็คือการกดปุ่มหยุดทำงานในโปรแกรม Spider เมื่อกดปุ่มหยุดทำงาน Spider จะส่งคำสั่งหยุดทำงานไปให้ทุก Thread ที่กำลังทำงานอยู่ให้หยุด แล้วทุกๆ Thread ก็จะทำงานที่ค้างอยู่ให้เสร็จแล้วจึงหยุดการทำงาน และคืน Memory ให้แก่ระบบและ หยุดการทำงานทุกอย่าง



ภาพ3.15 แสดงการหยุดค้นหาเมื่อเกิดเมื่อ User กดหยุด

3.4.10 ประยุกต์ใช้การเขียนติดต่อ Database แบบ Less-Connection

เมื่อเข้าใจหลักการทำงานเบื้องต้นของการติดต่อ Database แบบ Odbc Less-Connection แล้ว ขอนำหลักการดังกล่าวมาประยุกต์ใช้งานกับตัว Spider โดยอย่างแรกที่ต้องรู้คือ Class ใดบ้างที่ต้องมีการติดต่อกับ Database ซึ่ง Class หลักๆที่มีการติดต่อ ฐานข้อมูลคือ Class Spider โดยเราต้องปรับ Code ในโปรแกรมดังนี้

```
public class Spider extends Thread implements ISQLWorkLoad
        {
            public static String usageTime;
                protected FindLink findlink;
                protected SpiderReport report;
            private Form_Frame form_frame;

        //      protected Vector workloadcomplete = new Vector();
        //      protected Vector workloadwaiting = new Vector();
        //      protected Vector workloaderror = new Vector();
        //      protected Vector workloadrunning = new Vector();
                protected Vector showlink = new Vector();
                protected Vector images = new Vector();
                protected boolean _halted = false;

        //      public Vector getWorkLoadComplete(){return workloadcomplete;}
        //      public Vector getWorkLoadWaiting(){return workloadwaiting;}
        //      public Vector getWorkLoadError(){return workloaderror;}
        //      public Vector getWorkLoadRunning(){return workloadrunning;}
                public Vector getShowLink(){return showlink;}
                public Vector getImages(){return images;}

                protected SpiderDone done = new SpiderDone();
                protected SpiderWorker pool[];
                private int poolsize;
                private int activeThreads=0;
                private int depth;
                private int layer=0;
            private String savepath;
            //private Vector[] WaitingQueue;
                private int count_layer=0;
            private Hashtable root;
```

```
        public static long stime;
        private String firsturl;

            Connection connection;
            PreparedStatement prepAssign;
            PreparedStatement prepGetStatus;
            PreparedStatement prepCount;
            PreparedStatement prepFindLayer;
            PreparedStatement prepInsert;
            PreparedStatement prepUpdate;
            PreparedStatement prepUpdate_data;
            PreparedStatement prepGetId_Spider;
        PreparedStatement prepDelete;
        PreparedStatement prepGetAllData;

        private static String dbsrc = "jdbc:odbc:DRIVER=Microsoft Access Driver
(*.mdb); " +
                    "DBQ=C:/Program Files/Cyber Crawler/database/spider.mdb; " +
                    "UserCommitSync=Yes; " +
                    "Threads=3; " +
                    "SafeTransactions=0; " +
                    "PageTimeout=5; " +
                    "MaxScanRows=8; " +
                    "MaxBufferSize=2048; " +
                    "DriverId=281; " +

    //"DefaultDir=C:/ProgramFiles/CommonFiles/ODBC/DataSources";
                "DefaultDir=C:/Program Files/Cyber Crawler/database";
        String driver="sun.jdbc.odbc.JdbcOdbcDriver";
        String source="jdbc:odbc:spider";
```

ในตรงจุดที่เป็นตัวหนาของ Source Code ด้านบนคือส่วนของการกำหนดค่าต่างๆ ของการติดต่อฐานข้อมูลแบบ ODBC Less-Connection ไว้ในตัวแปร String "dbsrc" โดย Path ต่างๆจะต้องสัมพันธ์กับตัว Installation ที่จะกล่าวในตอนหลัง เช่น โฟลเดอร์ที่เก็บ Database ที่ " **C:/Program Files/Cyber Crawler/database"** ซึ่งตัว Installation จะทำการสร้างโฟลเดอร์นี้และวางฐานข้อมูลไว้ที่นี่ด้วย มิฉะนั้นโปรแกรมจะหาฐานข้อมูลไม่เจอ ต่อมาจะแสดงการเรียกใช้ตัวแปร dbsrc

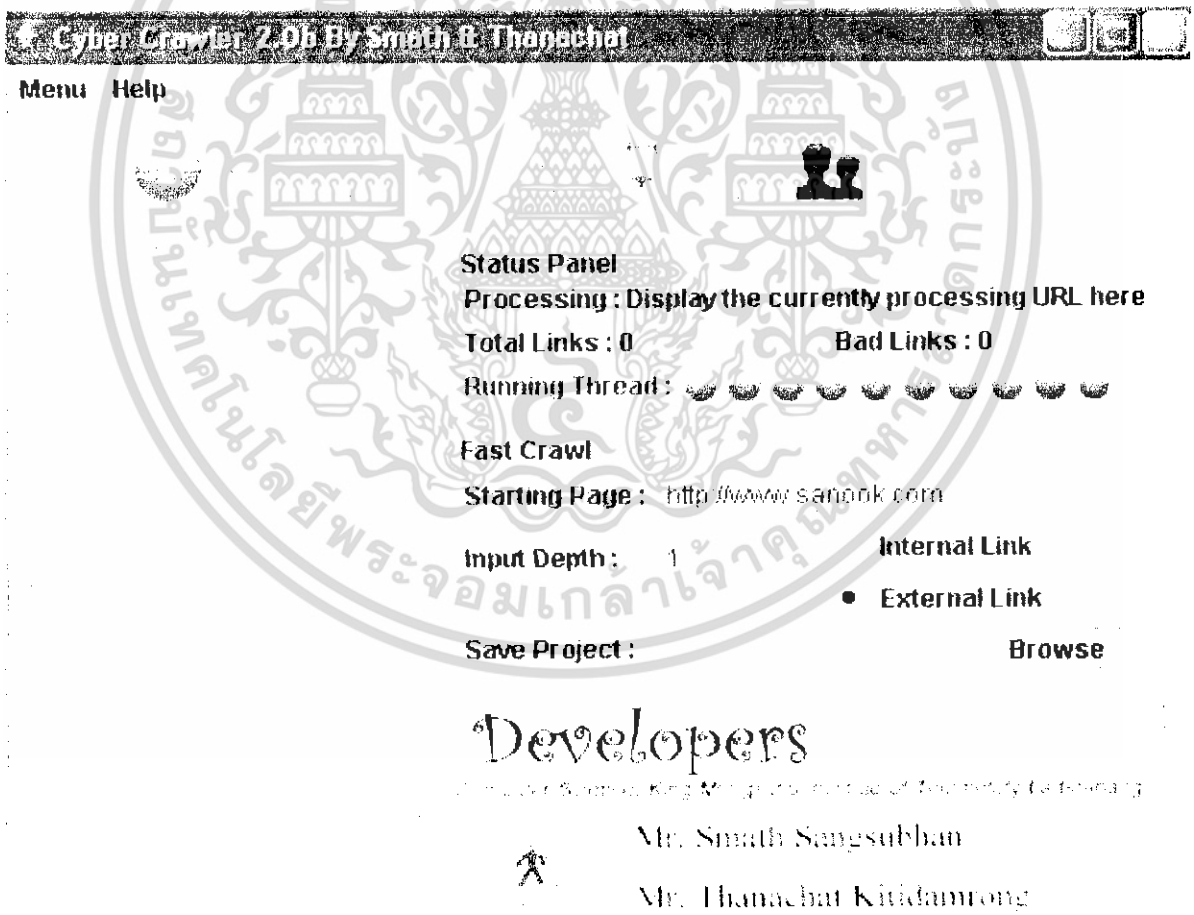
```
public void init_Database()
    {
            try
            {
                    Class.forName(driver);
                    connection = DriverManager.getConnection(dbsrc,"","");
                    prepGetId_Spider = connection.prepareStatement("select id from spider
where url=?");
                    prepAssign = connection.prepareStatement("select url from spider
where status='W'");
                    prepGetStatus = connection.prepareStatement("select status from spider
where url=?");
                    prepCount = connection.prepareStatement("select count(*) from spider
where url=?");
            prepInsert = connection.prepareStatement("insert into
spider(url,status,layer,linkfromurl,title,content,adddate)values(?,?,?,?,?,?,?)");
            prepUpdate_data = connection.prepareStatement("update spider set
title=?,content=?,adddate=? where url=?");
            prepFindLayer = connection.prepareStatement("select layer from spider where
url=?");
            prepUpdate = connection.prepareStatement("update spider set status=? where
url=?");
            prepDelete = connection.prepareStatement("delete from spider");
                    prepGetAllData = connection.prepareStatement("select * from spider
order by layer");
```

การเรียกใช้โปรแกรมจะทำได้ต่อเมื่อ ได้ติดต่อ Database โดยทำการเรียก Method init_Database() เพื่อ Initial ค่าเริ่มต้นในการติดต่อ ตัวแปร dbsrc ได้ถูกกำหนดค่าเอาไว้ในบรรทัดนี้ " **connection = DriverManager.getConnection(dbsrc,"","");**" จากนั้นเราก็สามารถเรียกใช้และทำงานกับฐานข้อมูลได้เลย

3.4.11 การแยก Feature ในการทำงาน

3.4.11.1 Fast Crawl Spidering

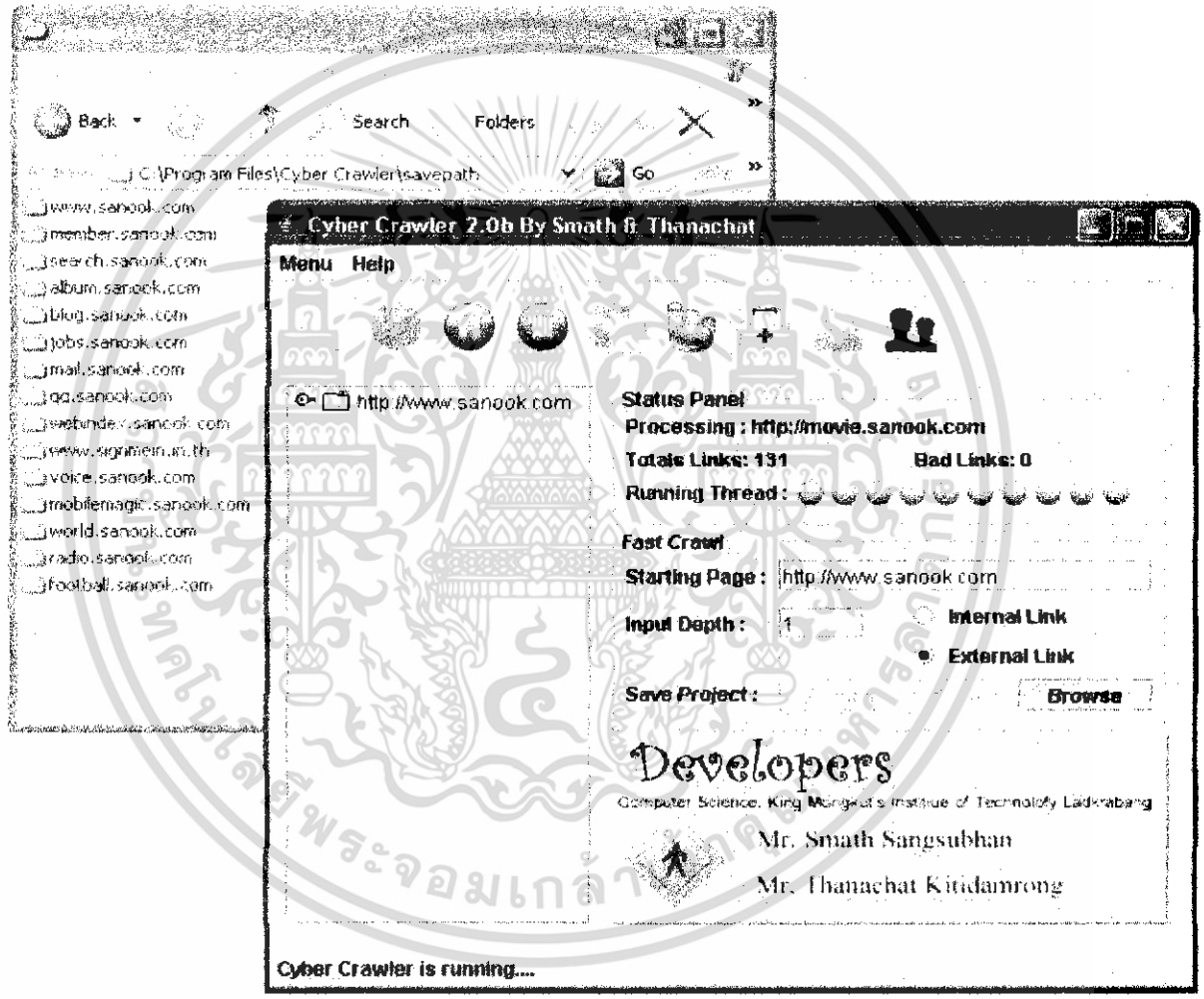
Fast Crawl Spidering จะเป็นการค้นหาอย่างรวดเร็ว เป็นเหมือนเมนูลัดที่ช่วยให้ User สามารถทำการค้นหาได้อย่างรวดเร็วและสะดวกเพราะไม่ต้องสร้างโปรเจคใหม่ การเลือกแบบ Fast Crawl จะทำการค้นหา Content ทุกประเภทที่พบไม่ว่าจะเป็น HTML , jpg , gif , png , mp3 , wma , xml , css , pdf , swf , zip , rar และเก็บไฟล์ทุกไฟล์ที่ได้พบมาลงในเครื่องของ User ตาม Path ที่ได้ระบุเอาไว้



ภาพ3.16 แสดงการ Search แบบ Fast Crawl Spidering

- ผู้ใช้ใส่ URL ของ Website ที่จะทำการ Spider คือ http://www.sanook.com
- กำหนดความลึกของลำดับชั้นที่จะ Spider เป็น 1
- เลือกลักษณะการค้นหาเป็นแบบ External Link
- เลือก Save Project ไปยัง Path "c:\program files\cyber crawler\savepath\

จากนั้นกดปุ่ม *RUN* โปรแกรมจะก็เริ่ม Spider และ Spider จะเริ่มทำการค้นหาจากจุดเริ่มต้นที่เราได้กำหนดเอาไว้ไปเรื่อยๆจนกระทั่งถึงเงื่อนไขการหยุดที่ได้กำหนดเอาไว้



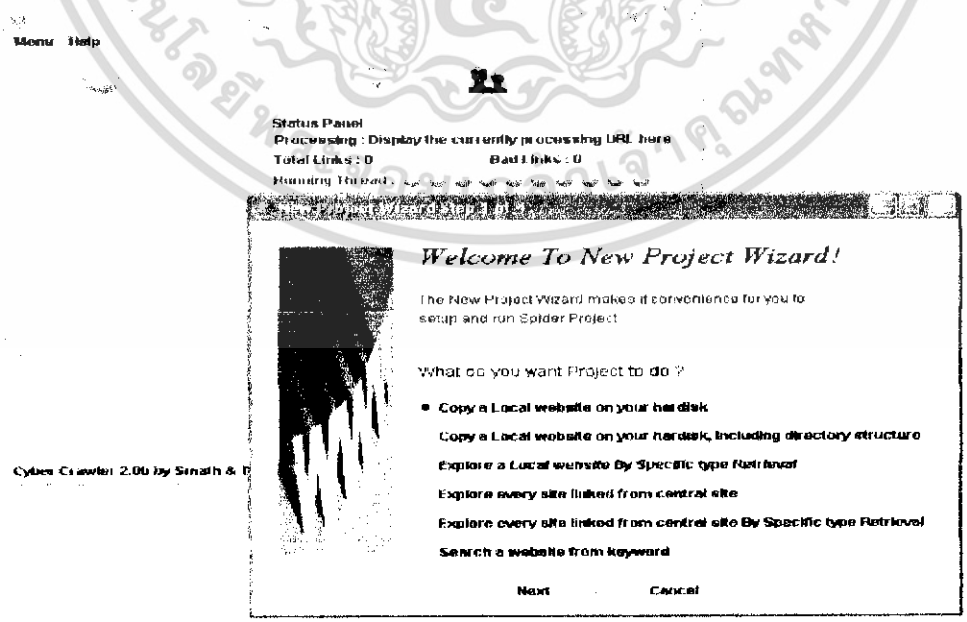
ภาพ3.17 แสดงผลลัพธ์จากการรัน Fast Crawl Spidering

จากรูปด้านบนจะเห็นว่า Content ที่ได้จากการค้นหานั้นถูกเก็บอยู่ใน Path ดังกล่าว และได้จัดเก็บแยกเป็น Folder ตามเว็บไซต์แต่ละเว็บ และในโฟลเดอร์ของแต่ละเว็บก็จะเก็บไฟล์ทุกประเภทที่ Spider ได้ทำการค้นหาได้มา ตามโครงสร้างของเว็บนั้นๆ

ภาพ3.18 แสดงการจัดเก็บ Folder ของไฟล์ที่ได้รวบรวมมา

3.4.11.2 New Project Work

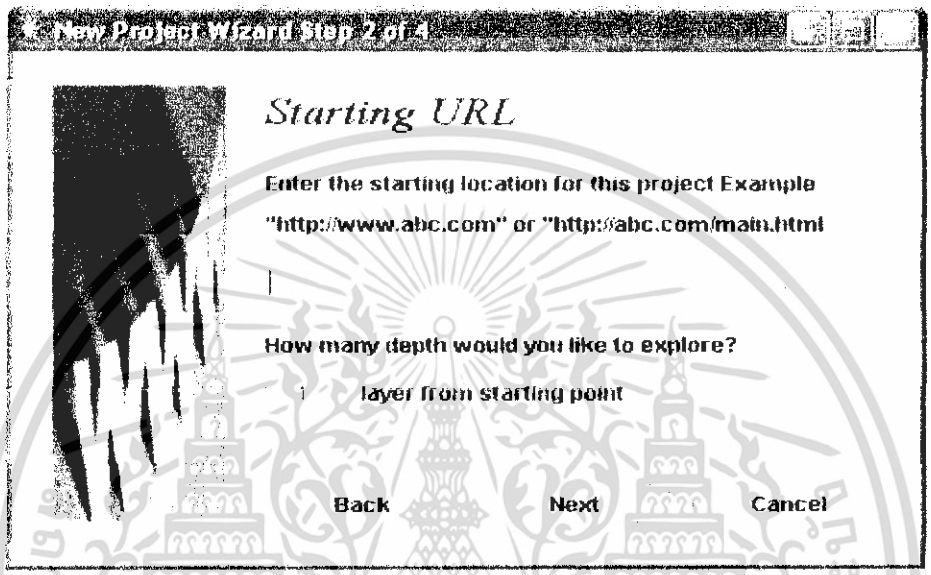
แบบที่ 2 คือการสร้างการค้นหาแบบละเอียด โดยต้องสร้างงานใหม่ขึ้นมาโดยการ New Project ใหม่ขึ้นมาและเลือกจากเมนูการค้นหาทั้ง 6 แบบ โดยแต่ละแบบนั้นก็จะมี Feature ที่แตกต่างกันไป ซึ่งการสร้างโปรเจคการค้นหาใหม่นั้นจะแบ่งออกเป็น 6 Features ดังภาพ



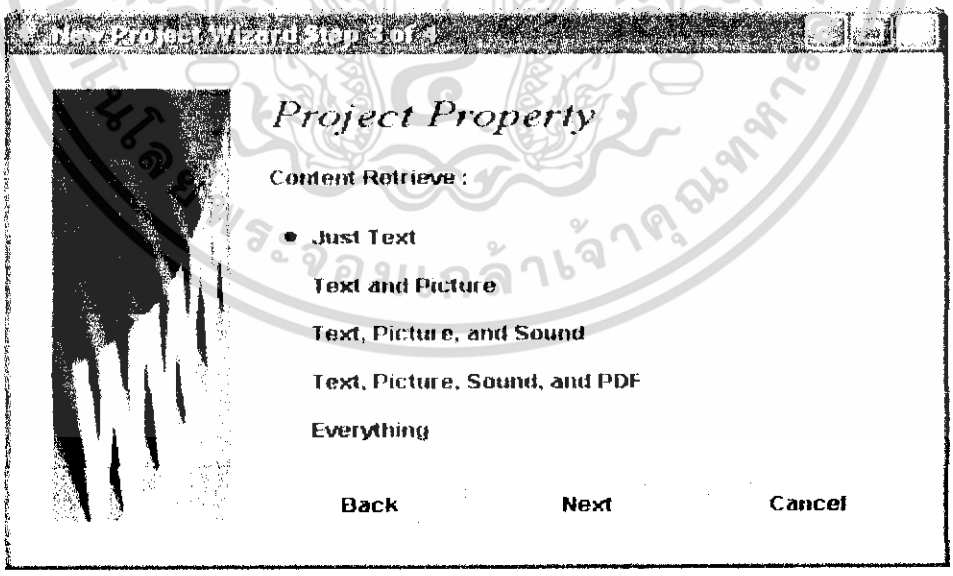
ภาพ3.19 แสดงการ New Project

- **Copy a Local website on your hardisk**

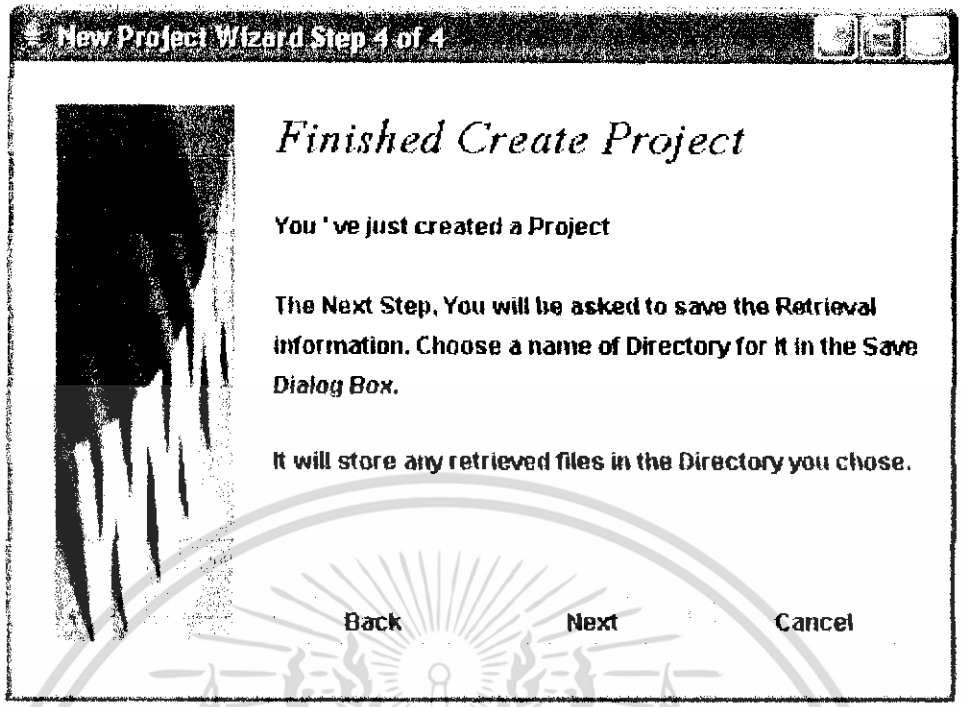
โปรแกรมจะทำการเก็บ Content ที่อยู่ภายใน Local ของ Website ที่ป้อนเข้าไปเท่านั้นและทุกไฟล์จะถูกเก็บใน Folder เดียวกัน จะไม่สร้าง Folder ที่เป็น Sub directory ตาม path ของ website นั้น ซึ่งอาจจะทำให้ดูยุ่งยากในการค้นหา การค้นหาแบบนี้ ไม่เหมาะกับการค้นหาเว็บเพจที่มีขนาดหน้าหลายๆ หน้ามาก เพราะจะให้ยุ่งยากต่อการมาค้นหาไฟล์



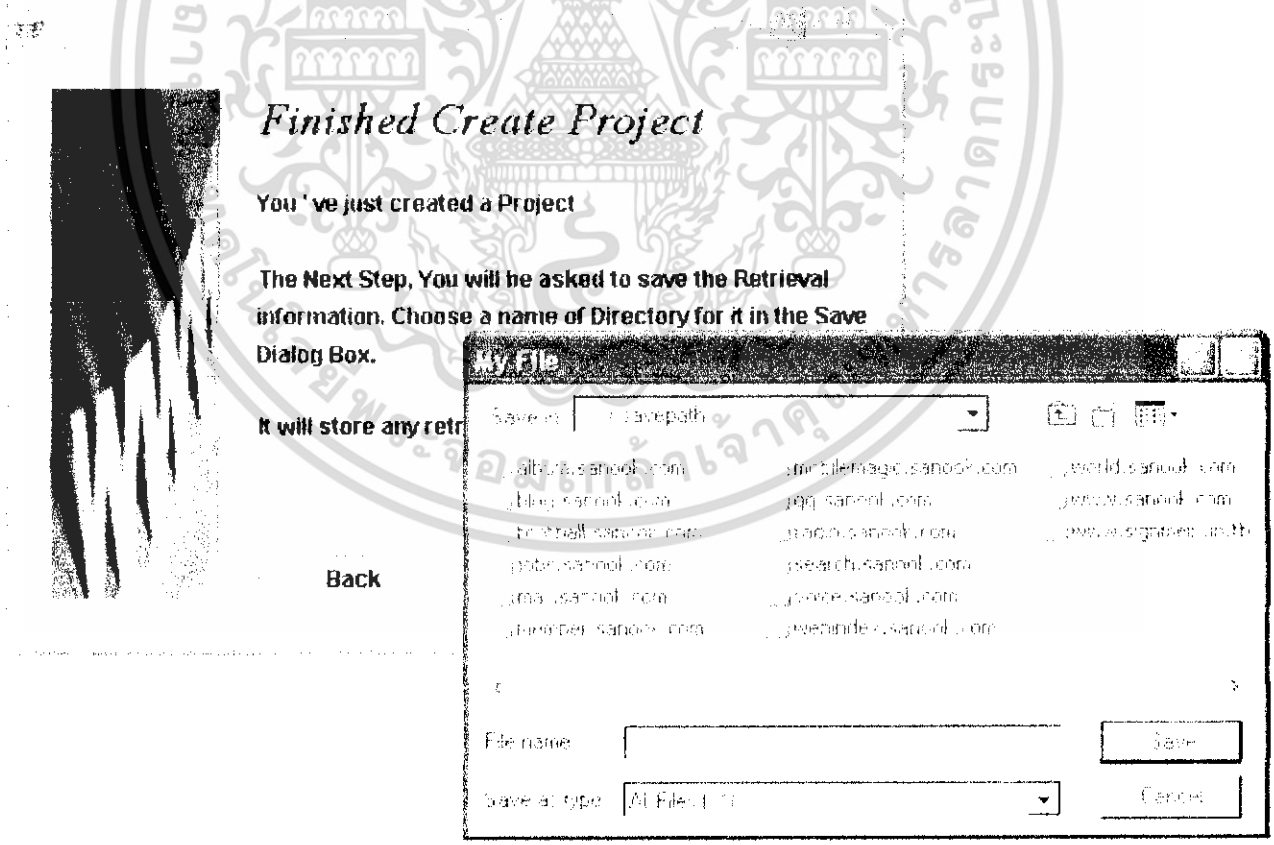
ภาพ3.20 แสดง (Feature 1) Step 1 : ใส่ URL ที่จะทำการ Spider และ ลำดับชั้นความลึก



ภาพ3.21 แสดง (Feature 1) Step 2 : เลือกประเภท Content ที่จะเก็บข้อมูล



ภาพ3.22 แสดง (Feature 1) Step 3 : แจ้งว่าการสร้างโปรเจคเสร็จเรียบร้อยแล้ว



ภาพ3.23 แสดง (Feature 1) Step 4 : ทำการ Save Project

เมื่อเลือกเสร็จแล้วก็กด Run โปรแกรมก็จะเริ่มทำการค้นหา โดยจะทำงานตามค่าที่ได้ถูกปรับไว้ใน Profile ที่ได้ Save ไว้เมื่อขั้นตอนที่แล้ว และจะทำการค้นหาไปเรื่อยจนพบเงื่อนไขให้หยุดตามเงื่อนไขที่ได้ระบุเอาไว้

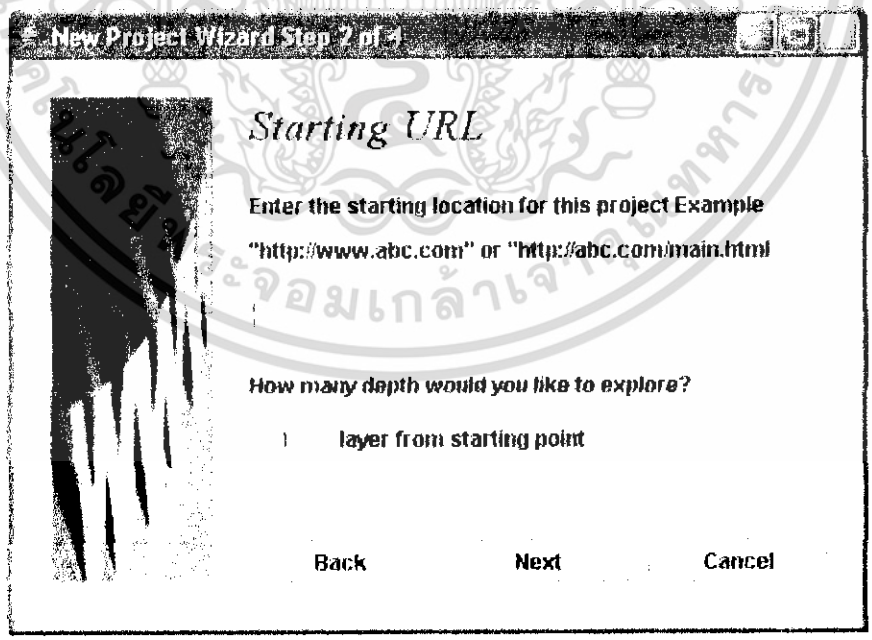
- **Copy a Local website on your hardisk, including directory structure**

โปรแกรมจะทำการเก็บ Content ที่อยู่ภายใน Local ของ Website ที่ป้อนเข้าไปเท่านั้นและไฟล์จะถูกเก็บใน Folder ตาม Sub directory ของ Website นั้น ซึ่งการทำงานทั้งหมดนั้นจะเหมือนแบบที่แล้วทุกประการ แต่จะมีการเพิ่มเติมคือการเก็บไฟล์ที่ค้นหาได้มานั้น จะจัดเก็บแยกตาม Directory ให้ด้วยเพื่อความสะดวกแก่การค้นหา Step การสร้างโปรเจคจะเหมือนกับหัวข้อที่แล้ว

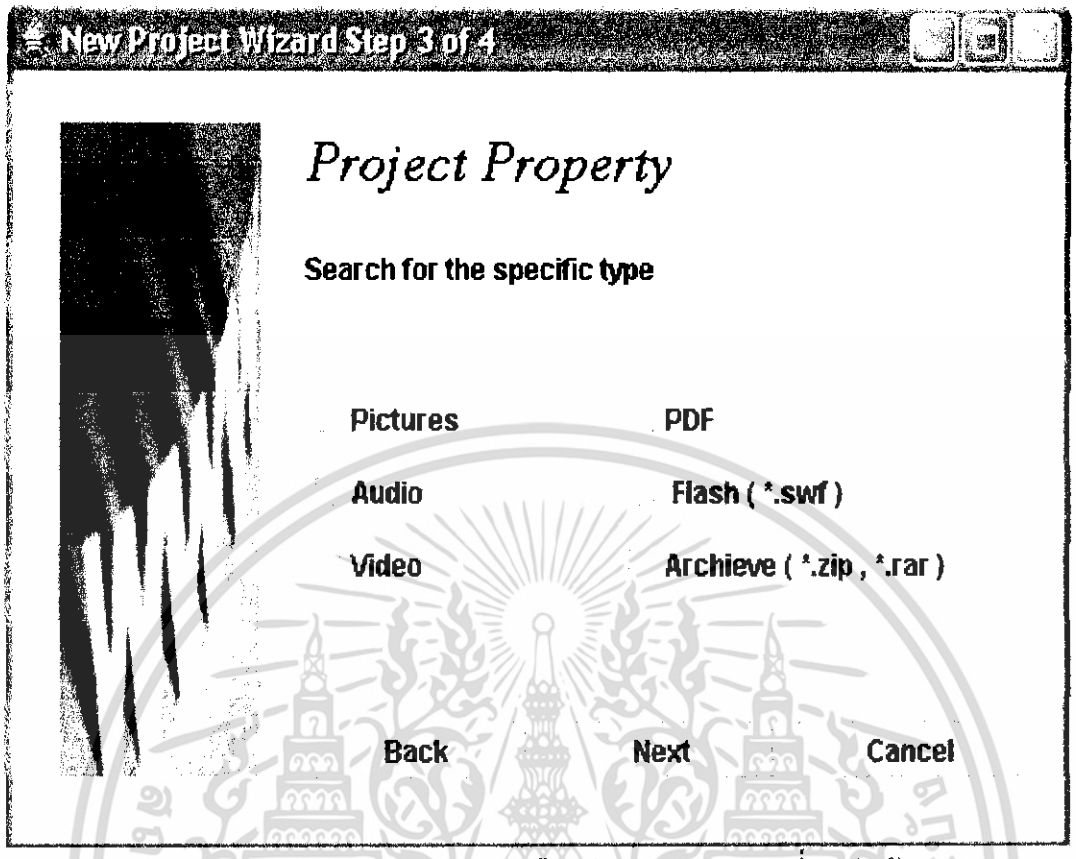
- **Explore a Local website By Specific type Retrieval**

โปรแกรมจะทำการเก็บ Content ที่อยู่ภายใน Local ของ Website ที่ป้อนเข้าไปเท่านั้นแต่ผู้ใช้สามารถเลือก Content ที่จะเก็บได้ ว่าจะเก็บไฟล์ประเภทไหนลง ฐานข้อมูลบ้าง ซึ่งสามารถจำแนกได้ตามนามสกุลของไฟล์ โดยสามารถระบุได้ว่าต้องการให้ Spider เก็บไฟล์ประเภทไหนมาบ้างตามที่ต้องการ โดยไฟล์สามารถจำแนกได้ดังนี้

---ไฟล์รูปภาพ, ไฟล์เสียงเพลง, ไฟล์วีดีโอ, ไฟล์ PDF, ไฟล์ Flash และไฟล์ที่ถูกบีบอัดมาเพื่อลดขนาด เช่น .Zip, .Rar



ภาพ3.24 แสดง (Feature 3) Step 1 : ใส่ URL ที่จะทำการ Spider และ ลำดับชั้นความลึก



ภาพ3.25 แสดง (Feature 3) Step 2 : เลือกประเภท Content ที่จะเก็บข้อมูล

- **Explore every site linked from central site**

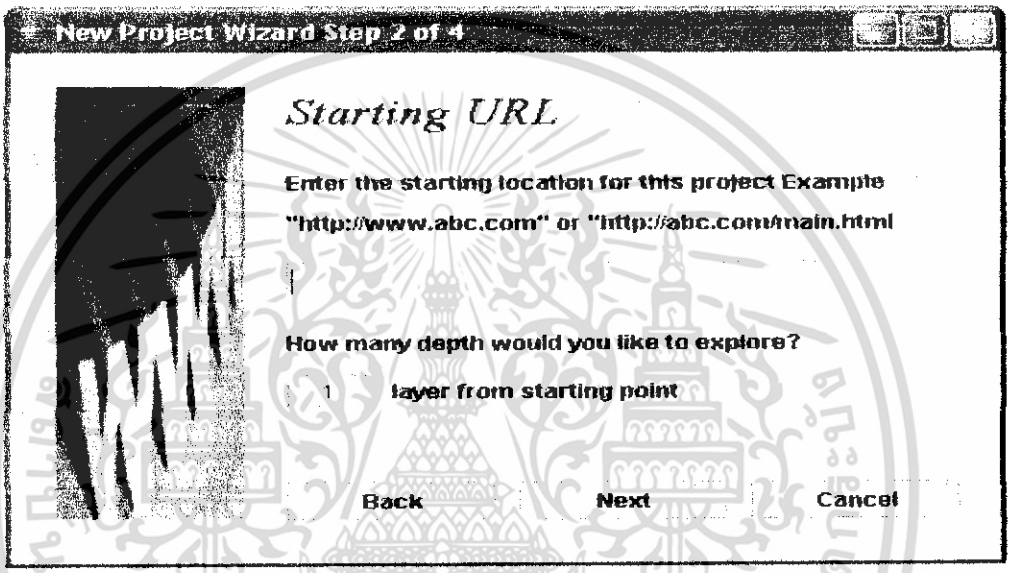
โปรแกรมจะทำการเก็บ Content ที่อยู่ภายใน Website ที่ป้อนเข้าไปและเก็บ Content ที่ Web page นั้น Link ไปนอก website ด้วย หรือกล่าวคือจะทำการเก็บไฟล์ทุกไฟล์ และท่องไปยังเว็บไซต์ทุกเว็บที่ถูกลิงค์ไปถึง โดยไม่คำนึงว่าเป็น Local หรือไม่ ส่วนขั้นตอนการทำงานก็จะเหมือนกับแบบอื่นๆ คือให้กรอกเว็บไซต์ที่เป็นจุดเริ่มต้น แล้วทำการค้นหาโดยเริ่มจากที่เว็บที่ระบุ

- **Explore every site linked from central site by specific type retrieval**

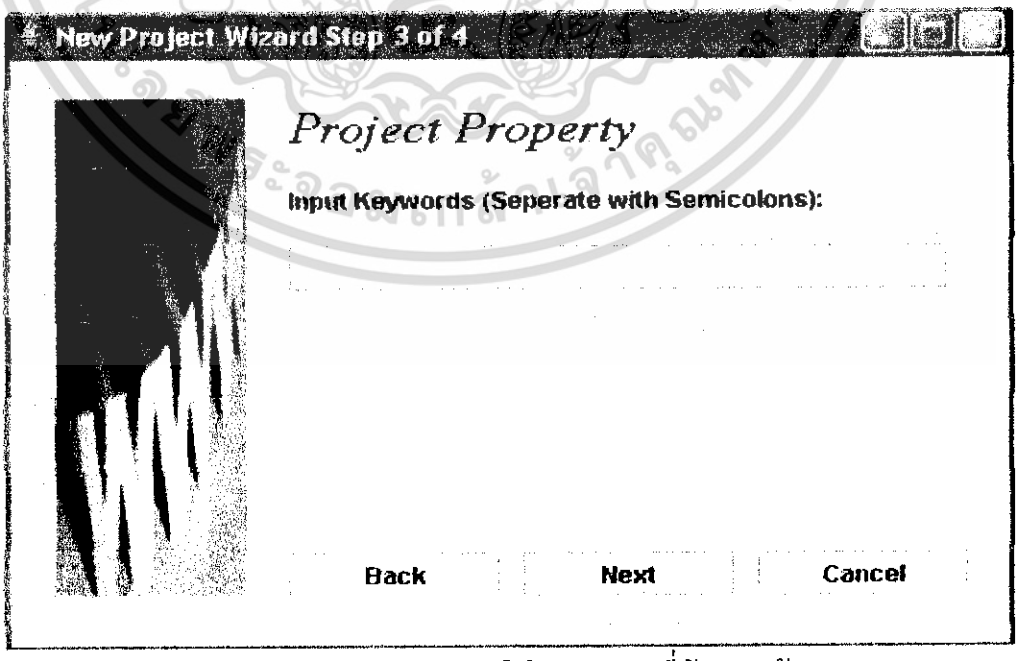
โปรแกรมจะทำภารเก็บ Content ที่อยู่ภายใน Website ที่ป้อนเข้าไปและเก็บ Content ที่ Web page นั้น Link ไปนอก website ด้วย หรือกล่าวคือจะทำการเก็บไฟล์ทุกไฟล์ และท่องไปยังเว็บไซต์ทุกเว็บที่ถูกลิงค์ไปถึง โดยไม่คำนึงว่าเป็น Local หรือไม่ ส่วนขั้นตอนการทำงานก็จะเหมือนกับแบบอื่นๆ คือให้กรอกเว็บไซต์ที่เป็นจุดเริ่มต้น แล้วทำการค้นหาโดยเริ่มจากที่เว็บที่ระบุ แต่ว่าแบบนี้จะต่างกับแบบข้างบนตรงที่สามารระบุได้ว่าต้องการไฟล์ประเภทไหนบ้าง

- **Search website from keyword**

โปรแกรมจะทำการเริ่มการค้นหาจากเว็บเพจแรกที่ User ระบุให้ไปค้นหา โดย User สามารถใส่ Keyword ที่สนใจจะค้นหาลงไปได้ด้วย เมื่อ โปรแกรมทำงานที่เว็บเพจแรกอยู่นั้น โปรแกรมจะทำการค้นหาว่าในเว็บเพจแรกนั้นมี ข้อความหรือเนื้อหาที่ตรงกับ Keyword ที่ User ต้องการค้นหาหรือไม่ ถ้าหากไม่ตรงก็จะไม่เก็บเพจนี้ลงฐานข้อมูล ต่อจากนั้นก็จะไปเช็คยังเพจต่อๆไปที่ได้ถูกลิงค์มา เรื่อยๆ ถ้าเพจไหนมีเนื้อหาตรงหรือคล้ายคลึงกับ Keyword ที่ User ต้องการก็จะเก็บเว็บเพจและ Content ของมันมาลงฐานข้อมูล พร้อมทั้งทำการ Ranking ด้วย Vector Space Model Algorithm เพื่อความสะดวกของ User ในการค้นหา



ภาพ3.26 แสดง (Feature 6) Step 1 : ใส่ URL ที่จะทำการ Spider และ ลำดับชั้นความลึก



ภาพ3.27 แสดง (Feature 6) Step 2 : ใส่ Keyword ที่ต้องการค้นหา

3.5 การทำ Installation CD

เมื่องานส่วนการ Programming ของ Spider เสร็จเรียบร้อยแล้ว งานต่อมาก็คือการทำระบบ Installation CD เพื่อเพิ่มความสะดวกสบายแก่ User ให้สามารถนำโปรแกรม ไปใช้งานได้ทันที เนื่องจากถ้าหากให้ User จัดการลงโปรแกรม Spider เอง User ที่ไม่มีความรู้เรื่องคอมพิวเตอร์ขั้นสูง อาจจะมีปัญหากับการ Set Database หรือการลง Java Run Time Library (J2sdk) ข้อดีของการทำระบบ Installation CD มีดังนี้

- ช่วยติดตั้ง Database อัตโนมัติ เรื่องจากการเขียนโปรแกรมแบบใช้ เทคนิค ODBC Less-Connection นั้น ไม่จำเป็นต้องไปตั้งค่าใน ODBC DSN แต่จำเป็นต้องสร้าง Path ที่อยู่ของ Database ให้ตรงกับที่ระบุไว้ในโปแกรม โดยตัว Installation จะจัดการนำไฟล์ Database ไปวางไว้ในที่ที่ต้องใช้เองโดยอัตโนมัติ
- ช่วยตรวจสอบ Spec และความสามารถของคอมพิวเตอร์ว่ามีทรัพยากรเพียงพอที่จะรัน Spider หรือไม่และถ้าไม่พอ จะแสดงว่าตรงจุดไหนที่ไม่ผ่านบ้าง
- ช่วยตรวจสอบว่า เครื่องที่จะทำงานมีการลง Java Runtime Environment ไว้แล้วหรือยัง และถ้าหากมีการลงไว้แล้ว ระบบ Installation จะไม่ทำการลงซ้ำ แต่ถ้าหากว่ายังไม่ได้ลง ระบบจะทำการลงให้เองโดยอัตโนมัติ

3.5.1 การทำ Jar Executable File

ปกติแล้ว Java Source Code เมื่อถูก Compile แล้ว ตัวไฟล์ .java จะถูกแปลงเป็น Machine Code ที่อยู่ในรูป ของไฟล์ .class ของ Java แต่ว่าการจะรันไฟล์ .class นั้นจะค่อนข้างยุ่งยาก เพราะ Class แต่ละ Class จะเกิดไฟล์ .Class ขึ้นมา 1 ไฟล์ ซึ่งทำให้จะต้องใช่ไฟล์ในการรันเยอะมาก และยุ่งยากต่อการทำความเข้าใจ และถ้าหากขาดไฟล์ใดไปไฟล์หนึ่งก็จะรันไม่ได้เลย

จากดังที่กล่าวมา ดังนั้นจึงจะรวมไฟล์ทั้งหมดเป็น .jar ไฟล์เดียว ซึ่งตัวไฟล์ .jar ตัวนี้เป็น Executable File ในตัวเอง หรือก็คือ สามารถรันไฟล์นี้ได้เลยเหมือนไฟล์ .exe ทุกประการ โดยการทำไฟล์ .jar จากไฟล์ Java Source Code มีดังนี้

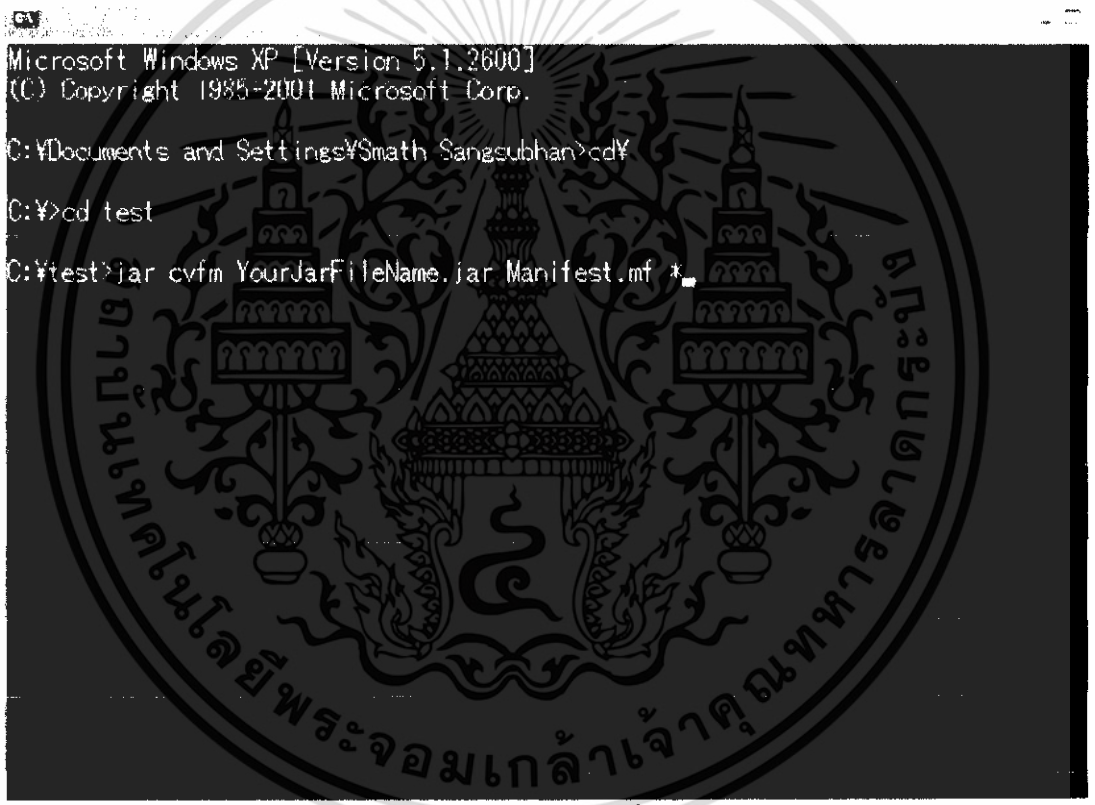
- ทำการCompile Java Source Code ให้เป็น .Class ทั้งหมด
- สร้างไฟล์ Manifest.mf ขึ้นมาโดยใช้ โปรแกรมText Editor โดยให้ใส่ Code ดังต่อไปนี้ลงไปในตัวไฟล์ Manifest.mf

```
Manifest-Version:1.0
Main-Class: YourClassNameWithMain
```

โดยให้ใส่ ชื่อไฟล์ Main Class ลงไปในบรรทัดล่างโดยการพิมพ์ห้ามมีการเว้นบรรทัดว่าง โดยเด็ดขาด เช่น

```
Manifest-Version:1.0
Main-Class: MainClass
```

● หลังจากนั้นให้นำไฟล์ .class ทั้งหมดและไฟล์ Manifest ที่ได้สร้างไว้มาใส่ไว้ในโฟลเดอร์เดียวกันแล้วเปิด Command Line Console แล้วพิมพ์คำสั่งดังนี้ (ต้องมีการเซ็ทค่า Global Variables ระหว่าง Window และ Java Library ไว้ก่อนเพื่อให้สามารถเรียกใช้ชุดคำสั่งจากโฟลเดอร์ใดก็ได้)



ภาพ3.28 แสดงการใช้คำสั่ง JAR

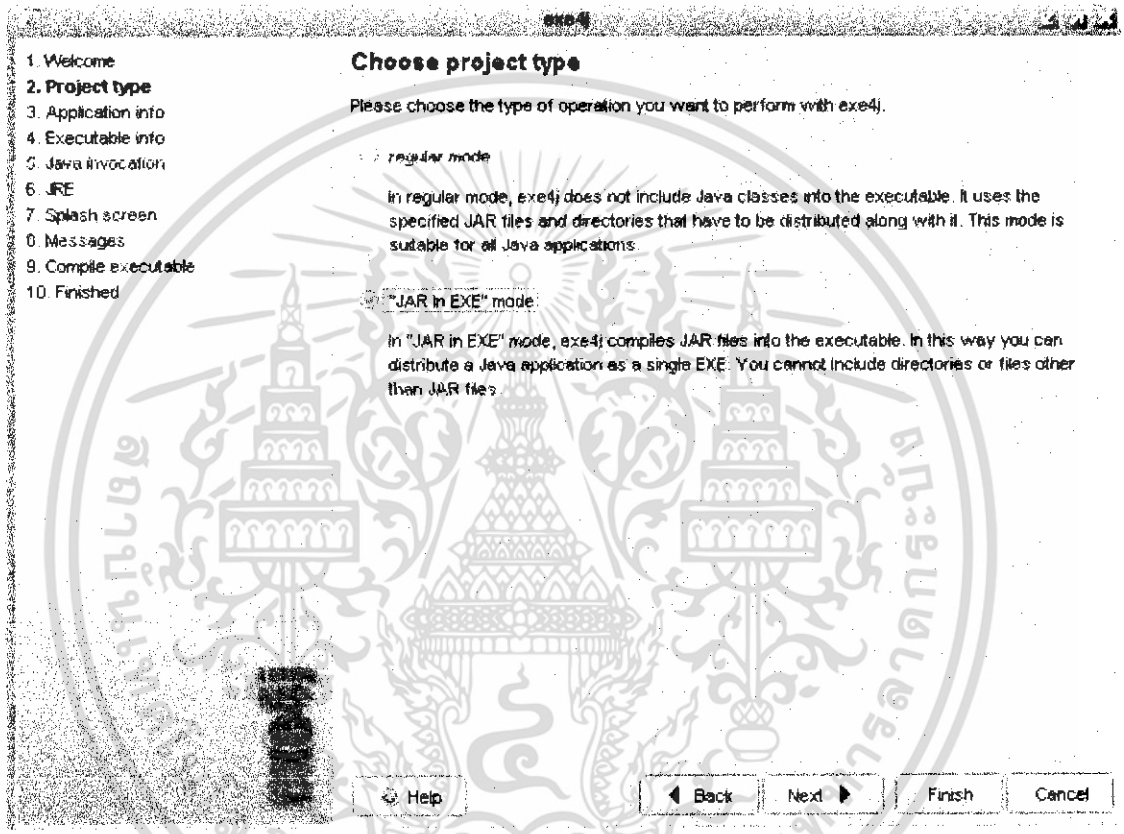
คำสั่ง cvfm แต่ละตัวคือ Attribute ของการปรับค่าการแปลงเป็น JAR File ส่วน * คือการบอกว่าเอา Class ไฟล์ทุกไฟล์ ถ้าเราไม่ใช้ * เราอาจใช้การระบุชื่อไฟล์เองก็ได้

3.5.2 การแปลง Jar to Exe

หลังจากที่ได้ไฟล์ .jar มาเรียบร้อยแล้ว สามารถ รันได้โดยการ Double Click แล้วก็สามารถรันได้เลยเหมือนไฟล์ .exe ของ วินโดว์ทั่วไป แต่ว่า ถ้าหากเครื่องที่ต้องการจะรันนั้นไม่มี

Java Runtime Environment ลงอยู่ เราจะไม่สามารถรันไฟล์ .jar ได้ ดังนั้นเพื่อความสะดวก เราจึงควรทำไฟล์เป็น .exe เพื่อสามารถรันตัว Installation ได้โดยยังไม่ต้องลง Java Runtime Environment เพราะตัว Installation จะจัดการลงให้เองในขั้นตอนการ Install

ในการทำ .exe นั้น Java ไม่ Support ในการแปลงจากไฟล์ .jar ไปเป็นไฟล์ .exe ดังนั้นต้องใช้โปรแกรม Third-Party Program ในการช่วยแปลง ซึ่งโปรแกรมพวกนี้ก็มีของหลายเจ้าให้เลือกใช้ โดยโปรแกรมที่เราเลือกใช้คือโปรแกรม EXE4J



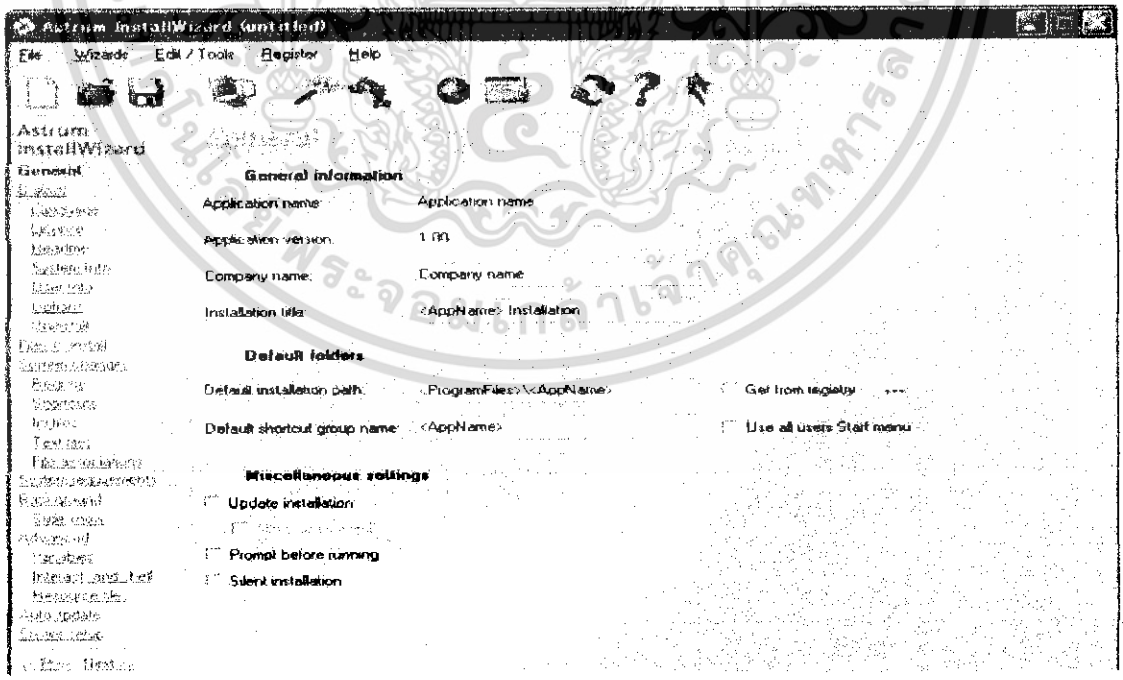
ภาพ3.29 แสดงโปรแกรม EXE4J

3.5.3 การทำ Installation Disc

หลังจากที่ได้ไฟล์ .exe ซึ่งเป็นไฟล์ Install มาไฟล์เดียวแล้ว ก็สามารถรันตัว Installation ได้ด้วยการรันเพียง Double Click แล้ว แต่ว่า นั่นยังไม่เพียงพอ เพราะว่า ถ้าหากมีเพียงแค่นี้ User ยังต้องไปลง Database เอง ไปเซท ค่าต่างๆ และรวมทั้งต้องลง Java Runtime Environment เอง ดังนั้นเราควรจัดการทุกอย่างให้ User เพื่อให้ User ไม่ต้องมายุ่งกับการเซทค่าต่างๆ

โปรแกรมที่ใช้ในการทำระบบ Installation คือโปรแกรม Astrum InstallWizard โดยเราจะอธิบายการปรับตั้งและการใช้งานโปรแกรมในส่วนหลักๆให้ดู โดยความสามารถหลักๆที่ระบบ Installation ที่ Astrum InstallWizard ทำได้ มีดังนี้

- สามารถนำไฟล์ที่ต้องการไปวางในที่ที่ระบุได้ โดยอัตโนมัติ
- ช่วยตรวจสอบ Spec และความสามารถของคอมพิวเตอร์ว่ามีทรัพยากรเพียงพอที่จะรัน Spider หรือไม่และถ้าไม่พอ จะแสดงว่าตรงจุดไหนที่ไม่ผ่านบ้าง
- ช่วยตรวจสอบว่า เครื่องที่จะทำงานมีการลง Java Runtime Environment ไว้แล้วหรือยัง และถ้าหากมีการลงไว้แล้ว ระบบ Installation จะไม่ทำการลงซ้ำ แต่ถ้าหากว่ายังไม่ได้ลง ระบบจะทำการลงให้เองโดยอัตโนมัติ
- สามารถแสดง Splash Screen แสดงภาพก่อนเข้าโปรแกรมได้
- สามารถแสดง Version ของโปรแกรมได้
- สามารถปรับค่าใน Registry ของเครื่องที่รันระบบ Installation ได้
- สามารถปรับภาษาที่แสดงได้หลายภาษาตามที่ต้องการ
- สามารถแสดง License, Readme และ Agreement ให้กดยอมรับก่อนที่จะทำการ Install ได้
- สามารถสร้าง Desktop Shortcut และ Menu Shortcut ได้
- สามารถ Uninstall หรือถอดถอนการติดตั้งโปรแกรมออกได้

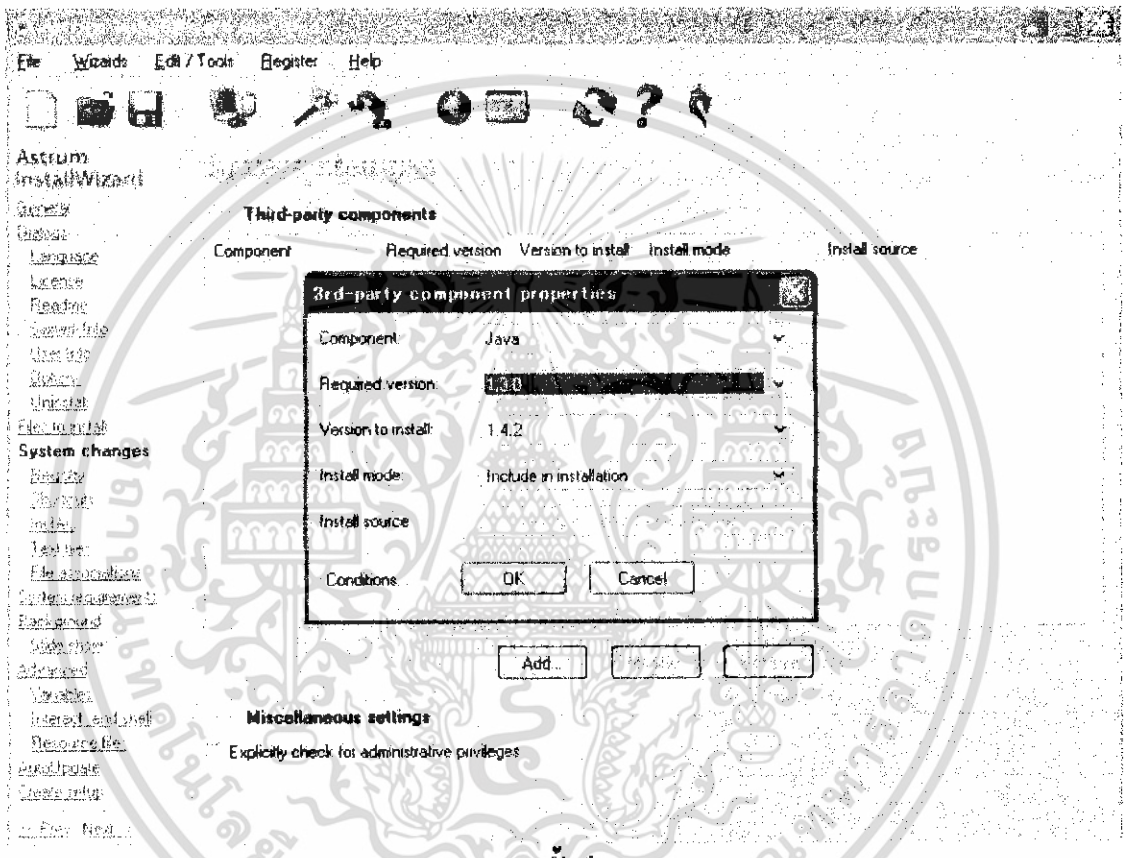


ภาพ3.30 แสดงโปรแกรม Astrum InstallWizard

3.5.3.1 การใช้งานโปรแกรม Astrum InstallWizard ในฟังก์ชั่นหลักๆ

● **การเช็ค Java Runtime Environment ของระบบ**

สามารถเลือกได้ว่าเครื่องของ User จะต้องมี Java Runtime Environment ขั้นต่ำ Version เท่าไรถึงจะเหมาะสม และถ้าหากไม่มี ก็จะทำการลงตัว Java Runtime Environment ให้เองโดยอัตโนมัติ นอกจากนี้ยังสามารถเลือกเช็ค Library ของ ตัวภาษาอื่นด้วยก็ได้



ภาพ3.31 แสดงการตั้งค่า เมนู System Change

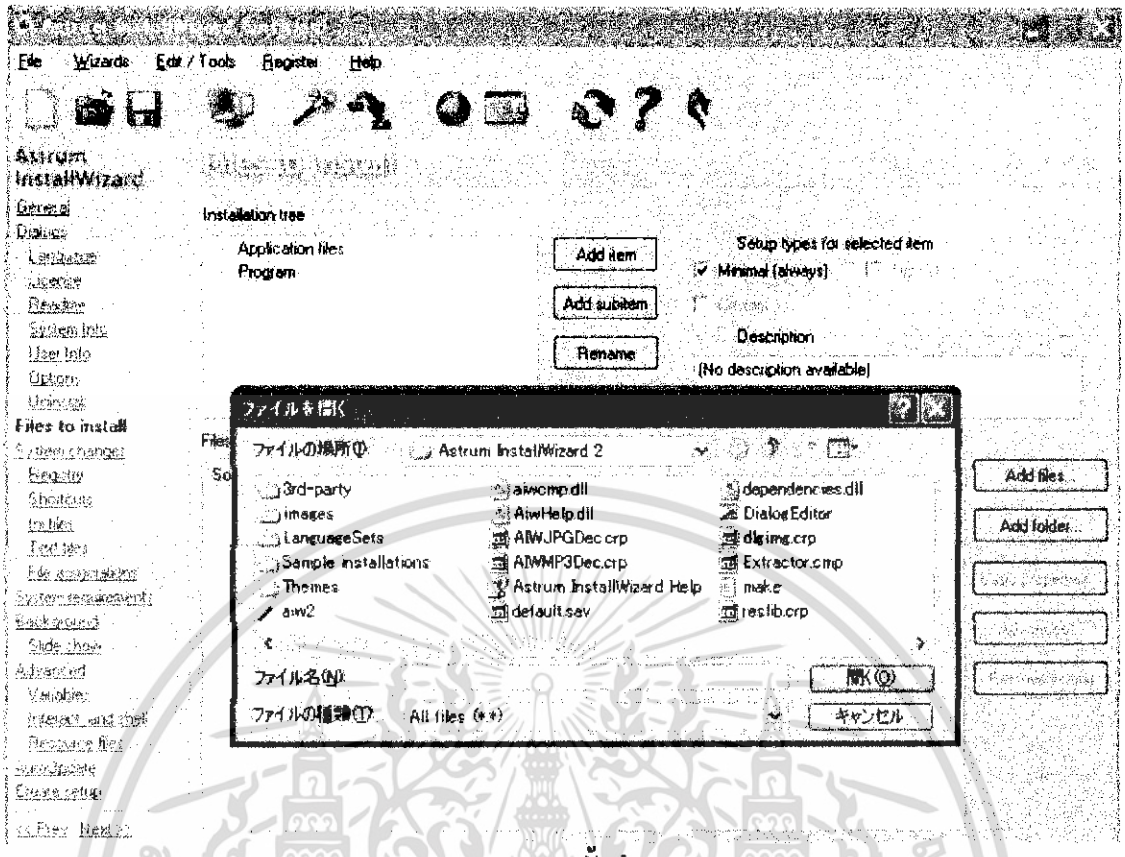
● **การเลือกไฟล์ที่จะ Install ลงในเครื่องของ User**

สามารถเลือกได้ว่าจะลงไฟล์ใดบ้างไว้ที่ โฟลเดอร์ใดบ้าง โดยไฟล์หลักๆที่ต้องลงและจำเป็นต้องใช้ได้แก่

- ไฟล์ Database ที่ ใช้ในการเก็บข้อมูล
- ไฟล์ตัว .exe ที่เป็นไฟล์ Install Spider ที่ได้ทำไว้

นอกจากนี้ยังสามารถสร้าง Folder เพื่อจัดเตรียมไว้สำหรับการใส่ไฟล์ต่างๆที่อาจเกิดขึ้นจาก Spider เอง เพื่อให้สะดวกแก่การค้นหา

-



ภาพ3.32 แสดงการตั้งค่า เมนู File to Install

3.5.4 การทำ Autorun CD

การทำ CD Install ให้สามารถรันได้อัตโนมัตินั้น ก็เป็นอีก ทางเลือกที่จะช่วยเพิ่มความสะดวกสบายให้แก่ผู้ใช้งาน ซึ่งในการจะรันตัว Installation เองแบบ Manual นั้น ผู้ใช้จะต้องรันไฟล์ที่เป็น สกุล .exe แต่ถ้าหากผู้ใช้ไม่มีความรู้เรื่องคอม ก็อาจจะไม่ทราบได้ว่าจะรันตัว Installation อย่างไร ในการทำ Autorun CD นั้นทำได้ง่ายๆ โดยการ Write ตัวแผ่น Installation CD นั้นเราจะต้องใส่ ไฟล์ Text ไว้ตัวหนึ่งซึ่งจะทำหน้าที่เก็บคำสั่งในการรันอัตโนมัติไว้ โดยจะต้องเซฟ ไฟล์นี้ไว้ในชื่อ autorun.inf โดยในไฟล์ autorun.inf นั้นจะต้องระบุชุดคำสั่งไว้ดังนี้

```
[autorun]
OPEN=autorun.exe      "ชื่อไฟล์ที่จะให้รันอัตโนมัติเมื่อใส่แผ่น"
ICON=Autorun\Civ4Installer.ico      "ไอคอนที่ต้องการให้แสดง"
LABEL=Sid Meier's Civilization 4      "Text Label ที่ต้องการให้แสดง"
```

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

ในบทนี้จะกล่าวถึงขั้นตอนการติดตั้งและการใช้งานจริง ซึ่งจะรวมถึงการอ่านค่าและการแสดงผลต่างๆที่ได้ออกมาจากการค้นหาด้วย หลังจากนั้นจะมาดูถึงปัญหาของกระบวนการการทำงานขั้นตอนต่างๆ ว่ามีปัญหาหรืออุปสรรค์เกิดขึ้นในขั้นตอนใดบ้าง

4.1 คุณสมบัติของระบบที่นำมาทดสอบ

- Operating System – Microsoft Window XP Home Service Pack 2
- CPU – Pentium M (Centrino-Sonoma) 1.73 GHz
- Main Memory – 1024 MByte
- Graphic Card – ATI Mobility X700 –256M Hypermemory
- Internet Connection – TRUE ADSL 2.5 Mbps

4.2 ขั้นตอนการดำเนินการทดสอบ

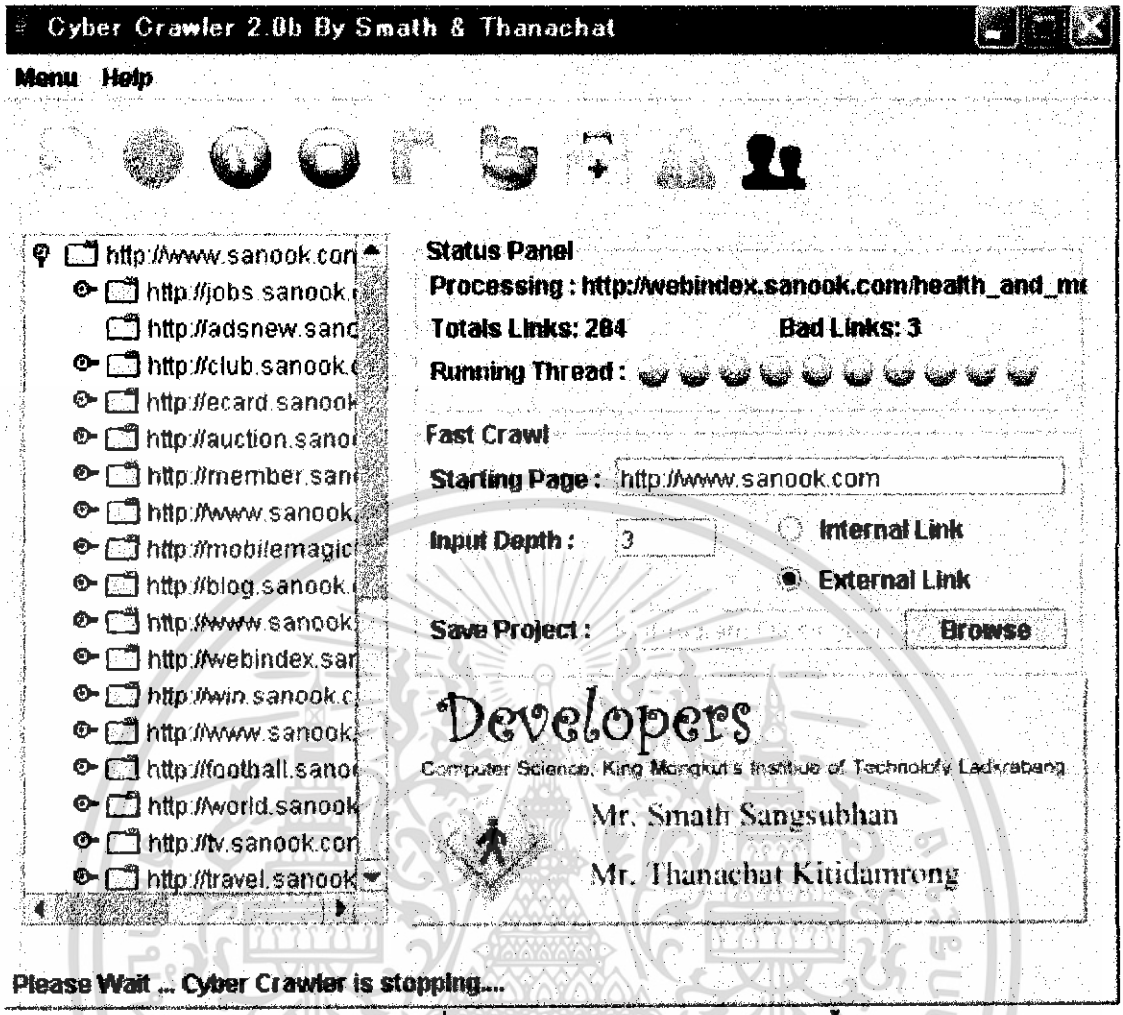
- ทดสอบการทำงาน โดยใช้เว็บที่เป็นจุดเริ่มต้นในการค้นหาเป็นเว็บเดียวกัน
- ดูผลการแสดงค่าต่างๆ ว่าถูกต้องหรือไม่
- ปรับเปลี่ยนค่าความลึกและลองทำการค้นหาแบบ Advance Search
- ลองกับเว็บหลายๆรูปแบบเพื่อทดสอบความถูกต้อง
- ดูไฟล์ทั้งหมดที่ได้เก็บมาลงเครื่องว่าเก็บได้ครบและถูกต้องหรือไม่
- วัดค่าความเร็วว่าสามารถทำงานได้กี่หน้าต่อวินาที

4.3 จุดมุ่งหมายของการทดสอบ

- เพื่อทดสอบประสิทธิภาพของการทำงานของสไปเดอร์
- เพื่อวัดความเร็วของการทำงาน
- เพื่อหาว่าสามารถทำงานโหมดต่างๆได้ตรงใจผู้ใช้งานหรือไม่
- เพื่อหาจุดบกพร่องที่เกิดจากการทดลองและนำไปปรับปรุง

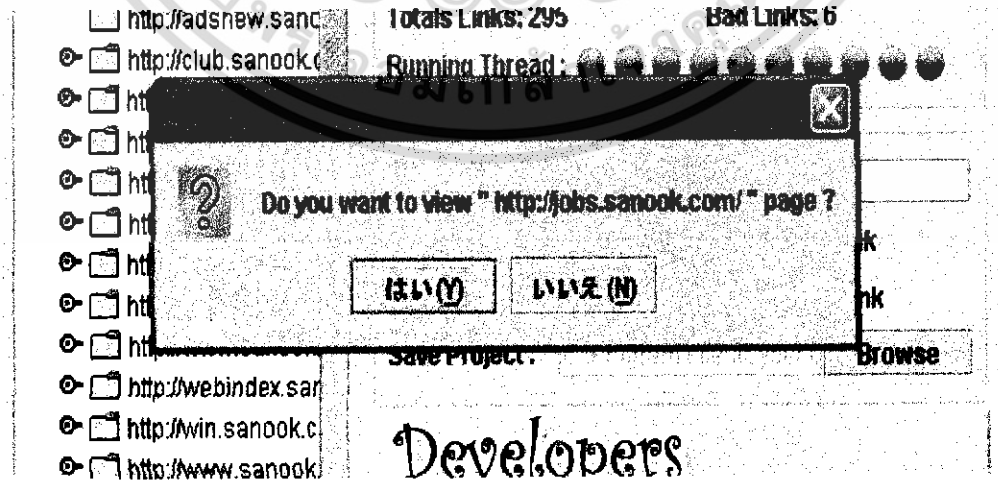
4.4 การแสดงผลลัพธ์ของการทำงาน

4.4.1 การแสดงผลการค้นหาในรูปของแผนผังต้นไม้



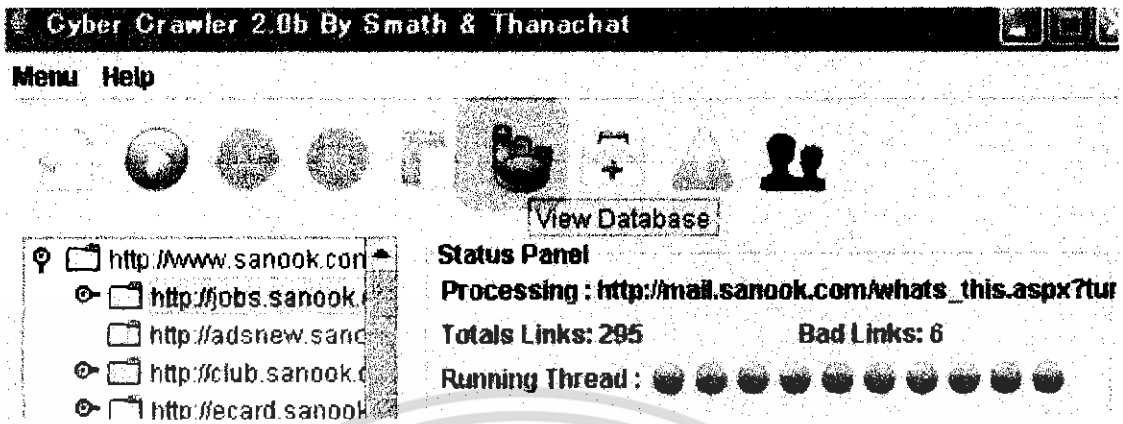
ภาพ4.1 แสดงเมื่อโปรแกรมหยุด หรือทำงานเสร็จสิ้น

เมื่อ Spider หยุดทำงาน จะสามารถดู Tree ที่แสดงลำดับชั้นของความลึกของเว็บที่ได้ทำการค้นหาได้และสามารถ Click ที่เว็บจากใน Tree เพื่อสามารถไปยังหน้าเว็บนั้นได้เลยดังรูป



ภาพ4.2 แสดงการลิงค์ไปที่เว็บที่เจอจาก Tree

4.4.2 การแสดงผลการค้นหาในรูปตารางฐานข้อมูล

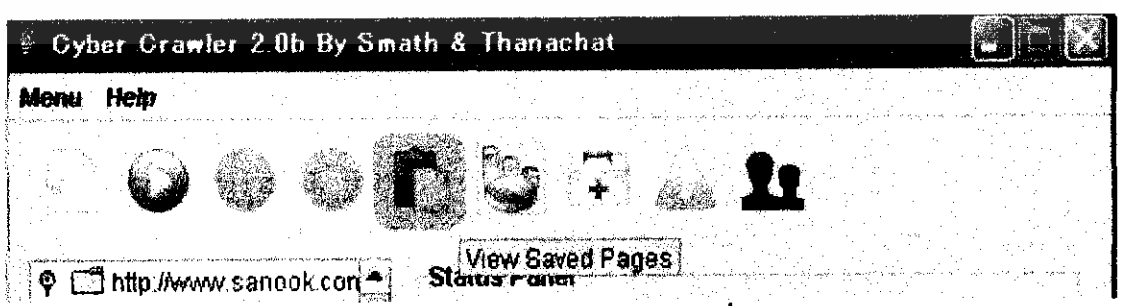


ภาพ4.3 แสดงการดูผลการค้นหาในฐานข้อมูล

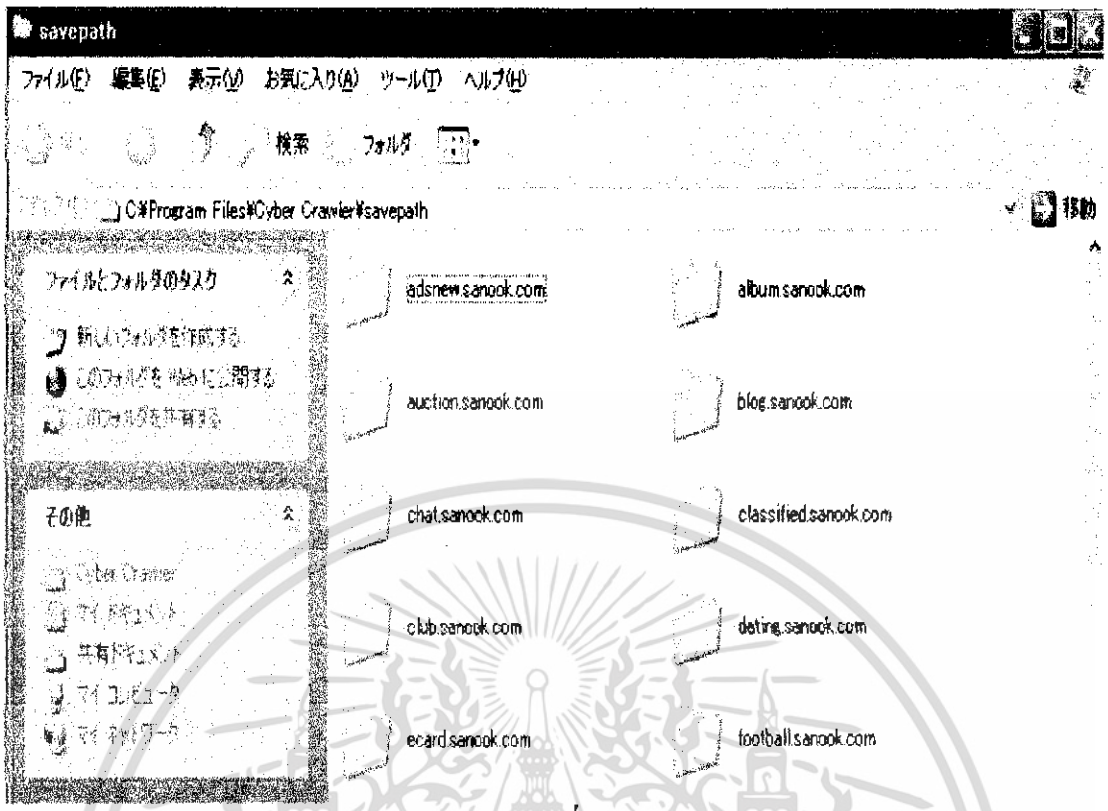
เมื่อค้นหาเสร็จแล้ว สามารถดูผลการค้นหาทั้งหมดที่ถูกเก็บลงฐานข้อมูลได้ โดยClick ที่ปุ่ม View Database บนโปรแกรม ซึ่งเมนูนี้จะเป็นการแสดงข้อมูลในฟิลด์ต่างๆที่ได้เก็บลงฐานข้อมูล ซึ่งก็มีข้อมูลที่สำคัญต่างๆเช่น URL, เวลาที่ได้เจอ, สถานะ การค้นหา และข้อมูลอื่นๆที่สำคัญอีกมากมาย โดยที่ไม่ต้องเสียเวลาเปิด MS Access

ภาพ4.4 แสดงฐานข้อมูลที่ได้เก็บไว้

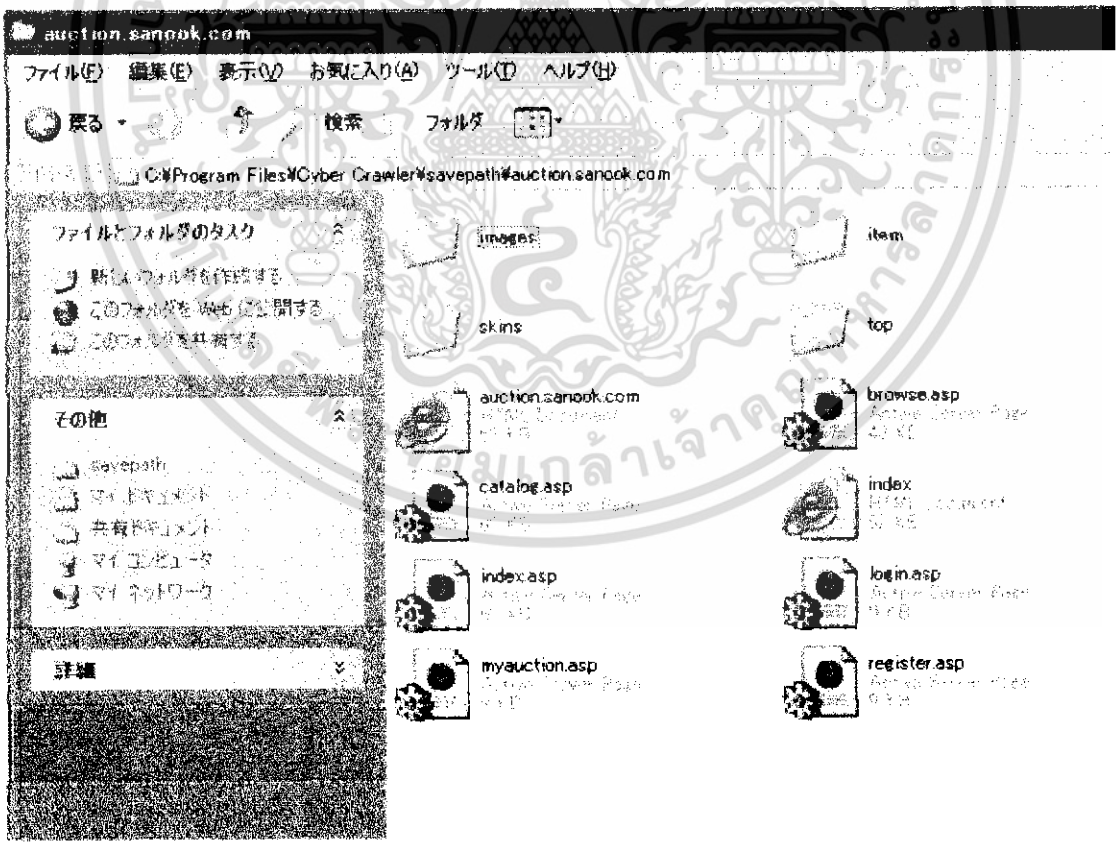
4.4.3 การแสดงไฟล์ทั้งหมดที่ได้ค้นเจอและเซฟเก็บไว้



ภาพ4.5 แสดงการเลือกดูไฟล์ที่ได้เก็บไว้



ภาพ4.6 แสดงไฟล์ที่สไปเดอร์ทำการเก็บไว้ 1



ภาพ4.7 แสดงไฟล์ที่สไปเดอร์ทำการเก็บไว้ 2

จะเห็นได้ว่าเมื่อเลือกเมนูนี้ สไปเดอร์จะทำการเปิด Window Explorer ไปยังโฟลเดอร์ที่ได้ระบุให้สไปเดอร์เก็บไฟล์ไว้เองโดยอัตโนมัติ ซึ่งในนั้นก็จะเก็บไฟล์ทุกไฟล์ที่ค้นหาได้มาไว้ทั้งหมด เพื่อความสะดวกจึงแยกโฟลเดอร์ไว้ให้ตาม Domain Name และ Sub Domain Name ของแต่ละเว็บที่ค้นเจอไว้ให้

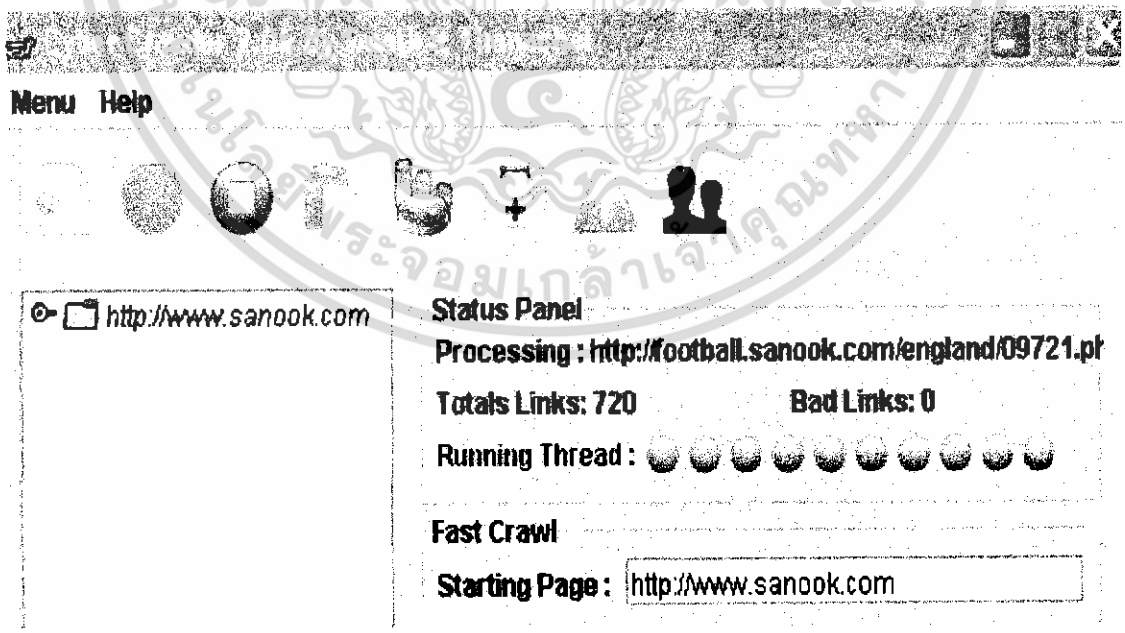
4.5 การเปรียบเทียบประสิทธิภาพเมื่อเปลี่ยนฐานข้อมูลที่ใช้

- **Microsoft Access 2003**

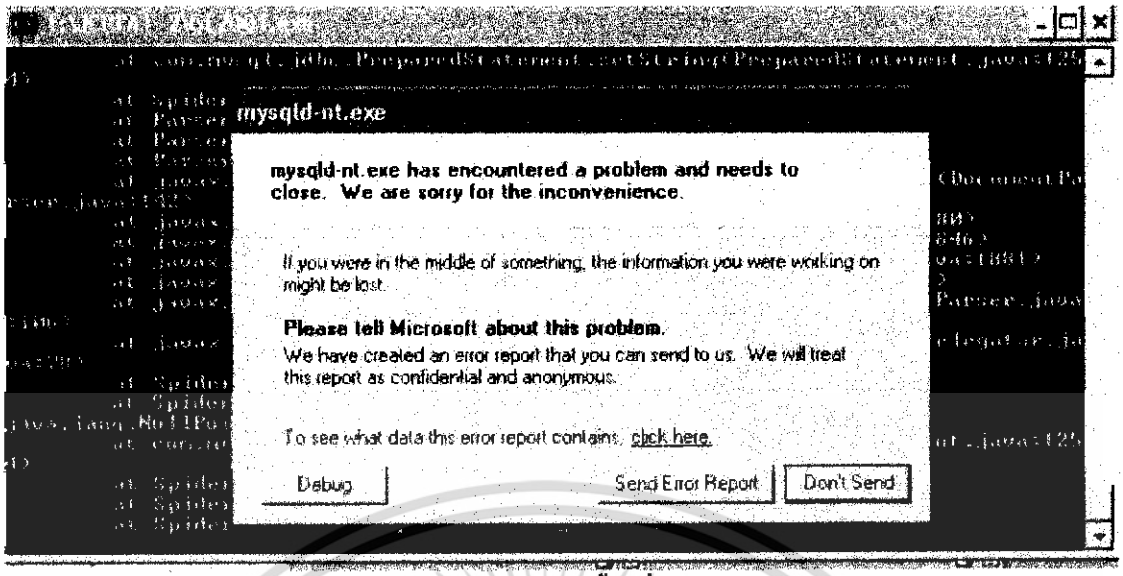
ในโครงงานนี้จะใช้ฐานข้อมูลตัวนี้ในการเก็บข้อมูลและใช้ในการทำงานของโปรแกรม เนื่องจาก ข้อดีเรื่องขนาดของ MS Access มีขนาดเล็ก และสามารถแจกจ่ายให้คนอื่นได้ง่าย นอกจากนี้ยังสามารถทำการเขียนโปรแกรมด้วยเทคนิค ODBC Less- Connection ทำให้เมื่อจะติดตั้งฐานข้อมูลนั้น ไม่จำเป็นต้องติดต่อผ่าน ODBC และไม่ต้องเซ็ทค่า DSN ให้ยุ่งยากอีกต่อไป โดยประสิทธิภาพของมันนั้น สามารถทำงานได้ประมาณ 33 หน้าต่อวินาที

- **MySQL**

เนื่องจากความแพร่หลายของ MySQL และเป็นฐานข้อมูลที่ใช้งานง่ายและมีคนใช้มากชนิดหนึ่ง ในตอนแรกตั้งใจจะใช้ฐานข้อมูล MySQL กับโปรแกรมในโครงงานนี้ แต่ว่า เรื่องจากสไปเดอร์เป็นงานที่ต้องมีการเก็บข้อมูลลงฐานข้อมูลทีละมากๆและบ่อยมาก จากการทดลอง MySQL จะค้างและไม่มีเสถียรภาพเมื่อรันไปได้ไม่ถึง 5นาที (ที่แวดล้อมเดียวกับ Access) ดังนั้นจึงไม่เลือกใช้ฐานข้อมูล MySQL กับโปรแกรมในโครงงานนี้



ภาพ4.8 แสดงการทำงานเมื่อใช้ MySQL



ภาพ4.9 แสดงการเกิดปัญหาขึ้นเมื่อรันไปได้ประมาณ 2 นาที

4.6 การวิเคราะห์ถึงปัญหาที่เกิดจากการทดลอง

จากการที่ได้ทำการทดลองดังข้างบนมา จะเห็นว่าสไปเดอร์ทำงานได้นับว่าสมบูรณ์ตรงตามความต้องการ และคาดว่าน่าจะตอบสนองถึงความต้องการในการจัดเตรียมฐานข้อมูลอัตโนมัติของผู้ใช้งานได้เป็นอย่างดี แต่ถึงกระนั้นก็ตาม โปรแกรมก็ยังมีจุดที่อาจจะต้องการการปรับปรุงและการพัฒนาต่อเพื่อให้สามารถใช้งานได้ เกิดประสิทธิภาพสูงสุด ซึ่งจุดต่างๆเหล่านั้นได้แก่

- **ปัญหาการสิ้นเปลืองทรัพยากรในการทำงานมาก**

เนื่องการทำงานของโปรแกรม เน้นที่ความเร็วในการท่องไปยังเพจต่างๆ ดังนั้นจึงนำเทคโนโลยี Multi-Thread มาใช้ ซึ่งผลที่ตามมาคือทำให้กินทรัพยาการของระบบมากขึ้น แต่ก็สามารถทำงานได้เร็วขึ้นกว่าเดิมหลานสิบเท่า ดังนั้นเครื่องที่จะทำการรันตัวสไปเดอร์ควรจะมีสเป็คที่สูงพอสมควร แต่ก็คงไม่ใช่ ปัญหาใหญ่มาก เพราะเครื่องที่จะใช้สไปเดอร์ทำฐานข้อมูลนั้นก็มักจะเป็นเครื่อง ระดับ Server Enterprise อยู่แล้ว

- **ไม่สามารถเชื่อมต่อไปยังเว็บที่มีการใส่ Username, Password**

เนื่องจากตอนนี้ยังไม่ได้ทำส่วนในการเข้าเว็บที่ต้องมีการเช็ค Username, Password ดังนั้นสไปเดอร์ จึงไม่สามารถเข้าหน้าเว็บที่มีการ Log-in ได้ ซึ่งจุดนี้จึงต้องมีการพัฒนาต่อไป ในอนาคต

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการทำงาน

จากการทำงานทั้งหมดที่ผ่านมา ได้ทำการศึกษาและรวบรวมข้อมูลเกี่ยวกับการทำสไปเดอร์ทั้งหมด เพื่อที่จะได้สามารถเข้าใจและสามารถพัฒนาระบบสไปเดอร์ขึ้นมาเองได้ โดยได้เล็งเห็นถึงความสำคัญของระบบรวบรวมข้อมูลอัตโนมัติที่จะเข้ามามีบทบาทในการทำงานของระบบสารสนเทศ จึงได้พยายามที่จะพัฒนาสไปเดอร์ขึ้นมา โดยมีขั้นตอนการดำเนินงานแบ่งเป็นขั้นตอนย่อยๆดังนี้

- การศึกษาและรวบรวมข้อมูล
- การวิเคราะห์และออกแบบสไปเดอร์
- การสร้างกราฟิกต่างๆและการค้นหาอัลกอริทึมที่เหมาะสม
- การทดสอบประสิทธิภาพของสไปเดอร์
- การแยก Feature การทำงานให้หลากหลาย
- การเก็บงานและตรวจสอบความถูกต้องของสไปเดอร์

5.2 ข้อเสนอแนะและสิ่งที่ควรพัฒนาต่อ

เนื่องจากสไปเดอร์มักจะเป็นโปรเจคระดับปริญญาโทถึงเอก แต่คณะผู้จัดทำสนใจและอยากที่จะลองทำขึ้นมาเมื่อเรียนอยู่ระดับปริญญาตรี ดังนั้นจึงอาจจะมีปัญหาบางประการที่ควรจะพัฒนาสไปเดอร์ต่อไปเพื่อให้สามารถนำไปสู่ภาคธุรกิจและใช้งานในระบบใหญ่ๆได้อย่างมีประสิทธิภาพมากขึ้น ซึ่งสิ่งที่ควรพัฒนาต่อไป ได้แก่

- การพัฒนา Algorithm ให้มีประสิทธิภาพมากขึ้นและกินทรัพยากรของระบบให้น้อยลง เพื่อให้สามารถทำงานได้ดีขึ้นและทำงานได้เร็วขึ้น
- การทำให้สไปเดอร์สามารถ Login เข้าไปยังเว็บที่มีการใช้ระบบ Username, Password เพื่อให้สามารถเก็บข้อมูลได้ทุกเว็บที่ต้องการ
- การทดลองให้ Tester ใช้งานจริงและรวบรวม Feedback กลับมาว่าผู้ใช้งานต้องการให้ปรับปรุงส่วนไหนบ้าง และทำการพัฒนาต่อไป

บรรณานุกรม

[1] SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers. A tutorial review, *Appeared in Proceedings of the Seventh International World Wide Web Conference (WWW7), Brisbane, Australia, April 1998. Printed in Computer Network and ISDN Systems v.30, pp. 119-130, 1998. Brisbane, Australia, April 1998.*

[2] Programming a Spider in Java by Jeff Heaton.

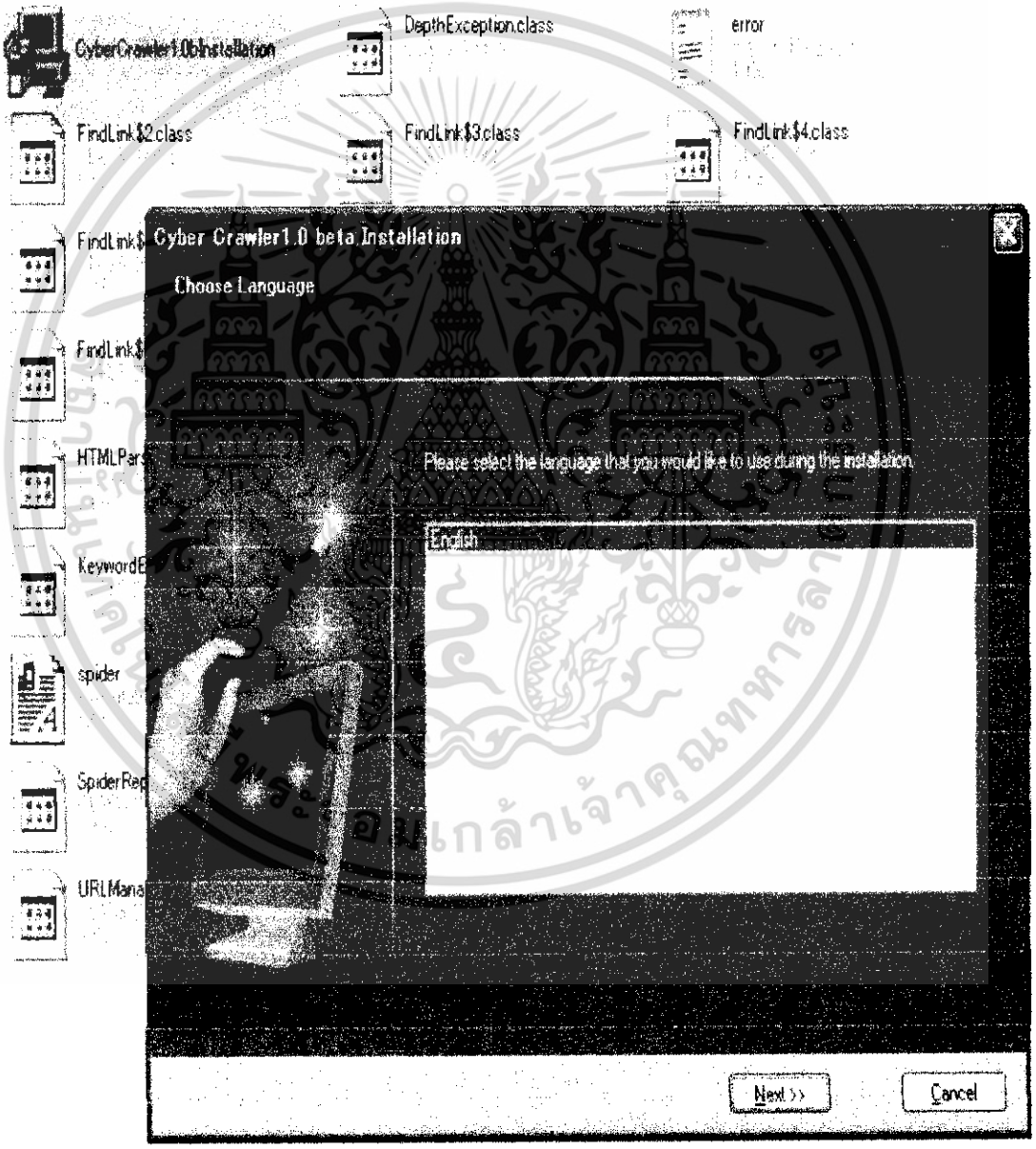
A tutorial review, *A nice book to practice with spider programming in Java language.*

[3] Writing a Web Crawler in the Java Programming Language by Thom Blum, Doug Keislar, Jim Wheaton and Ering Wold of Muscle Fish, LLC. A tutorial review, *another web crawler article which based on java language. This article provided the basic algorithm and the basic needed for web crawler. Such as, how web crawler work? Which function should provide in web crawler? What kind of error that may occur?*

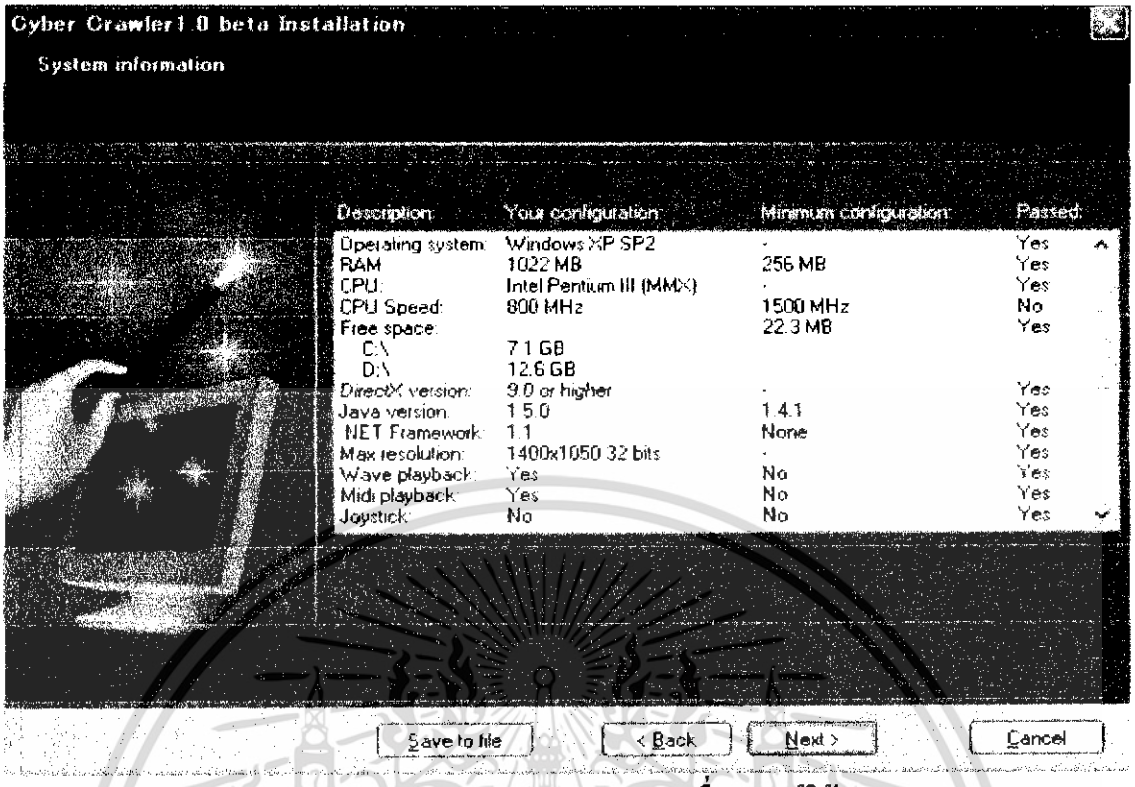
ภาคผนวก

ก. การติดตั้งโปรแกรม

เนื่องจากโปรแกรมได้จัดทำตัว Installation ไว้ให้ ซึ่งจะเป็นไฟล์ๆ เดียวเลย ดังนั้นผู้ใช้งานจึงหมดปัญหาเรื่องความยุ่งยากในการติดตั้งนอกจากนี้ได้จัดเตรียม Java Runtime ลงให้อัตโนมัติอีกด้วย รวมทั้งการเขียนโปรแกรมติดต่อ ฐานข้อมูลแบบ DSN-Less Connection ทำให้ผู้ใช้งานไม่ต้องไปติดตั้งฐานข้อมูลด้วยตัวเอง ขอเพียงมี MS Access ลงไว้ก็พอแล้ว

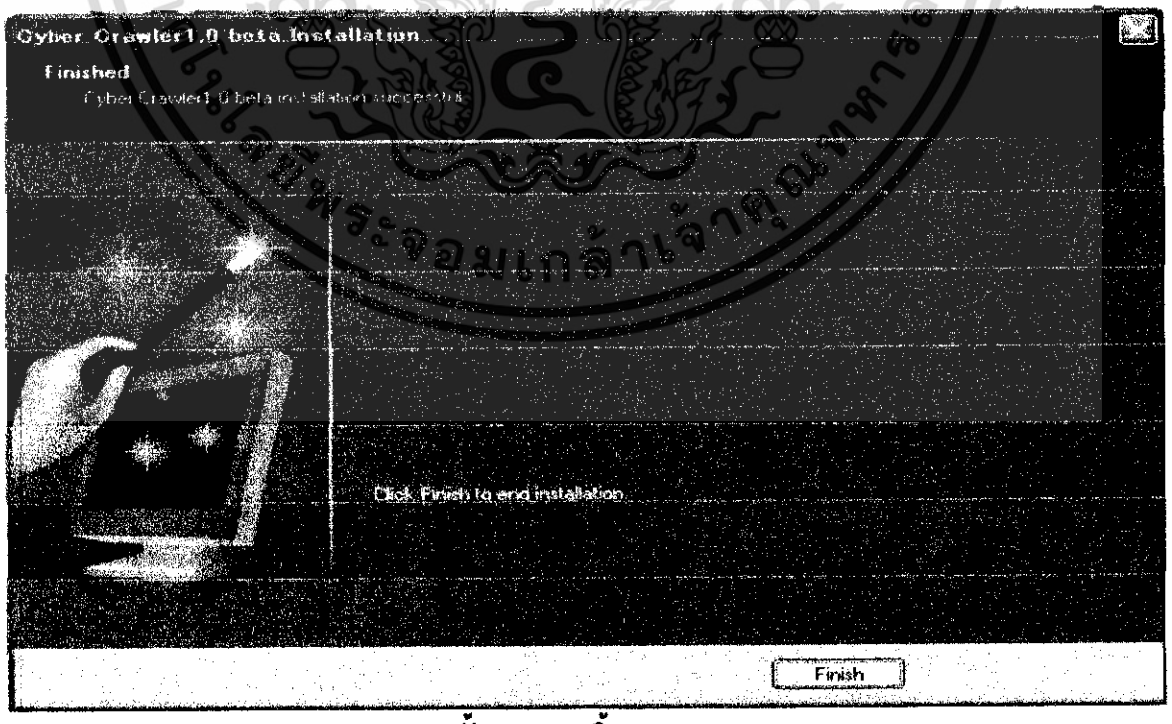


รูปภาพแสดงตัว Installation Wizard



รูปภาพแสดงการตรวจสอบ Spec เครื่องของผู้ใช้

หลักจากตรวจสอบความต้องการขั้นต่ำของโปรแกรมว่าเพียงพอต่อเครื่องก็ทำการกดต่อไป เพื่อให้ ตัว Installation Wizard ดำเนินการต่อไปจนกระทั่งเสร็จสิ้นการติดตั้ง ซึ่งในการติดตั้ง ตัว Installation Wizard จะทำการติดตั้ง Java Runtime และตัว ฐานข้อมูลให้เองโดยอัตโนมัติ



ภาพแสดงขั้นตอนการสิ้นสุดการ Install

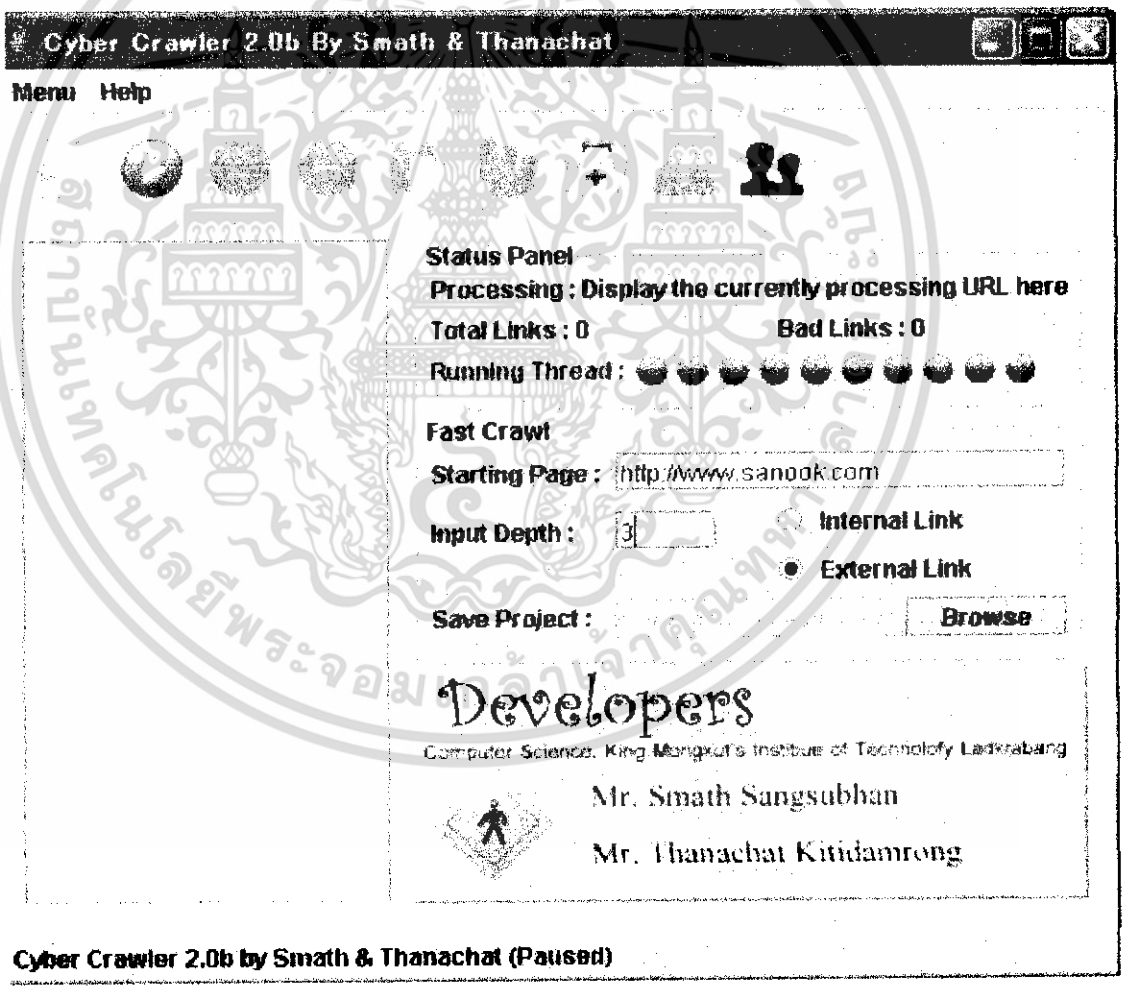
ภาคผนวก

ข. เริ่มต้นการใช้งาน

เมื่อจะเริ่มใช้งานโปรแกรม สิ่งแรกที่ควรจะรู้ก่อนคือ จุดแรกที่จะให้สไปเดอร์เริ่มทำงาน หรือก็คือเว็บเริ่มต้นนั่นเอง ต่อจากนั้น ก็ต้องกำหนดระดับความลึกและที่ๆต้องการเก็บเว็บเพจที่ได้ค้นหาเจอ ว่าจะเก็บไว้ที่ path ไหน โดยการใช้งานหลักๆจะมี 2 รูปแบบคือ

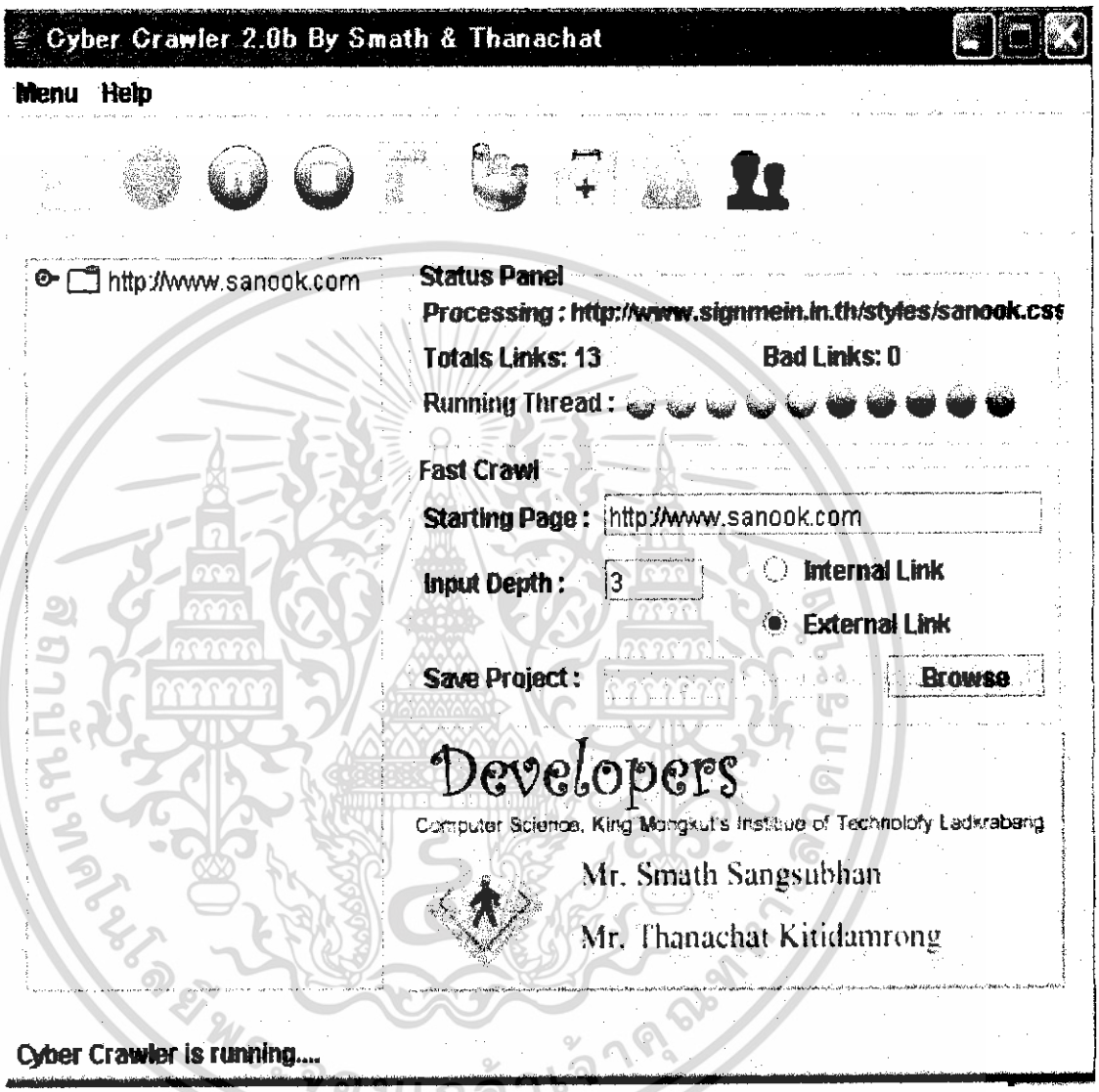
- Fast Search
- *Advance Search*

ดังที่ได้กล่าวไว้ในบทที่ 3 ดังนั้นจะข้ามถึงการอธิบายตรงจุดนี้ไป และยกตัวอย่างการทำงานแบบ Fast Search แทนเพื่อให้เห็นภาพได้ชัดเจน



ภาพแสดงหน้าจอหลักโปรแกรม

เมื่อเปิดโปรแกรมขึ้นมาแล้ว ก็ให้ใส่จุดเริ่มต้นที่ต้องการค้นหาลงไป โดยในตัวอย่างจะค้นหาจากเว็บ Sanook.com เป็นจุดเริ่มต้น โดยกำหนดระดับความลึก 3 ชั้น และกำหนด path ในการเก็บเว็บเพจให้เป็น Default และปรับให้ค้นหาทุกเว็บที่เจอไม่จำกัดเฉพาะเว็บใน Local Domain



ภาพแสดงโปรแกรมขณะกำลังรัน

เมื่อโปรแกรมกำลังรันนั้น Spider จะวิ่งไปยังเว็บต่างๆที่เจอจากในหน้าเพจของ Sanook.com และทำการเซฟเก็บ เว็บเพจและไฟล์ ทั้งหมด เช่น รูปภาพ เพลง และไฟล์ต่างๆบนเว็บลงไปใน path ที่ได้กำหนดเอาไว้ และจะแสดง ตัวเลขสถานะต่างๆ เช่น จำนวนเว็บที่เสีย หรือ จำนวนเว็บที่เก็บได้ทั้งหมดใน Status Panel