การติดต่อสื่อสารของเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก

Prolog based Multi-agent Communication

นายไพโรจน์ พุ่มขจร

นางสาววีระณี นุชิตประสิทธิชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การติดต่อสื่อสารของเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก

Prolog based Multi-agent Communication

โดย

นายไพโรจน์ พุ่มขจร

นางสาววีระณี นุชิตประสิทธิชัย

อาจารย์ที่ปรึกษา

ดร.วิศิษฏ์ หิรัญกิตติ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การติดต่อสื่อสารของเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก

Prolog based Multi-agent Communication

ผู้จัดทำ

1. นายไพโรจน์ พุ่มขจร 41014320
2. นางสาววีระณี นุชิตประสิทธิชัย 41014398

_____อาจารย์ที่ปรึกษา

(ดร. วิศิษฏ์ หิรัญกิตติ)

การติดต่อสื่อสารของเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก

นายไพโรจน์ พุ่มขจร 41014320
นางสาววีระณี นุชิดประสิทธิชัย 41014398
ดร.วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอแนวทางในการพัฒนาการติดต่อสื่อสารกันระหว่างเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก ทั้งนี้มีเป้าหมายเพื่อให้เอเจนต์หลายๆตัวสามารถถามคำถาม ให้ข้อมูลระหว่างกัน ในที่นี้จะให้เครื่องคอมพิวเตอร์ที่เป็นไคลเอ็นท์ และเซิร์ฟเวอร์เป็นเสมือนเอเจนต์ โดยฝั่งไคลเอ็นท์ใช้โปรล็อกสร้างเป็นบราวเซอร์ทำหน้าที่เป็นเอเจนต์ ส่วนฝั่งเซิร์ฟเวอร์ใช้โปรแกรม PROWEB® สร้างเป็นเว็บเซิร์ฟเวอร์ซึ่งทำหน้าที่เป็นเอเจนต์เช่นกัน เอจนต์ทั้งสองฝั่งสามารถสื่อสาร และเรียกใช้งานซึ่งกันและกันโดยใช้ภาษากลางเป็นไคว่รี่ในภาษาโปรล็อก นอกจากนั้นการสื่อสารนี้ยังไม่ได้จำกัดเพียงเอเจนต์สองตัวคือบราวเซอร์กับเซิร์ฟเวอร์เท่านั้น แต่ยังสามารถเป็นการติดต่อระหว่างเอเจนต์อื่นๆได้ เช่นระหว่างเซิร์ฟเวอร์กับเซิร์ฟเวอร์อื่น ทั้งนี้การที่เอเจนต์สามารถสื่อสารกันโดยใช้ไคว่รี่ในภาษาโปรล็อก เนื่องจากเอเจนต์ทั้งสองฝั่งมีการใช้งานเมตาอินเตอร์พรีทเตอร์ภาษาโปรล็อก

การวิจัยในโครงงานนี้จะครอบคลุมการสื่อสารที่เป็นการสื่อสารพื้นฐานของมนุษย์ เช่น การถามที่ผู้ถูกถามสามารถที่จะไปถามคำถามเดียวกันนี้กับผู้อื่นเพื่อหาคำตอบให้กับผู้ถาม การให้ข้อมูลที่ผู้ถูกบอกจะจดจำสิ่งที่มีผู้อื่นบอกไว้ และ การสั่งให้อีกผู้หนึ่งทำงานให้

PROLOG BASED MULTI-AGENT COMMUNICATION

Pairoj Poomkajorn 41014320
Veeranee Nuchitprasittichai 41014398
Dr. Visit Hirunkitti Advisor

ABSTRACT

This thesis presents an approach for communication between agents using Prolog. The purpose is to make multiple agents to ask questions and exchange information between each other. A client and a server computers are treated as the agents. We used Prolog to develop a web browser and used it as an agent. In addition, we used PROWEB®, a Prolog web utility to develop a web server and used it as another agent. Both agents are able to communicate and request executions of a command on each other by using a Prolog query as the common language. This communication is not limited to only the two agents the browser and the server but can be applied to other agents, for example, between servers. In order for the agents to be able to communicate using a Prolog query, we adopted a meta-interpreter as the core engine of every agent.

This research investigated basic communications of human, for example, the case when someone answering a question by obtaining the answer from another person, the case when someone is given an information and has to remember that information, and the case when someone is requested to do some activity.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่จะขาดเสียไม่ได้ในความสำเร็จดังกล่าวนี้ คือ อาจารย์ วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ไม่เพียงแต่สละเวลาเท่านั้น แต่ยังสละแรงกายและแรงใจในการเอาใจใส่ดูแล แนะนำ และช่วยเหลือเสมอมาจนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี อีกทั้งคณาจารย์ทุกท่านที่ถ่ายทอดวิชาความรู้มาให้ ซึ่งในความกรุณาของอาจารย์ที่มี ต้องขอบพระคุณเป็นอย่างสูง

ประการถัดมาที่มีส่วนสำคัญในความสำเร็จคือ เพื่อนๆ ทุกคนที่มีส่วนสนับสนุน ให้กำลังใจ และเป็นที่ปรึกษา รวมทั้งต้องขอบคุณห้อง AI ที่เป็นสถานที่ที่ให้ทำโครงงานและปริญญานิพนธ์นี้

และบุคคลที่สำคัญที่สุดที่ทำให้มีวันนี้ได้คือ บิดา มารดา อันเป็นที่เคารพรักอย่างสูง ซึ่งได้ให้ความรักความเอาใจใส่ ให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณ และกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

ไพโรจน์ พุ่มขจร

วีระณี นุชิตประสิทธิชัย

สารบัญ

หน้าที่

สารบัญ(ต่อ)

หน้าที่

สารบัญ(ต่อ)

หน้าที่

สารบัญรูปภาพ

หน้าที่

สารบัญรูปภาพ (ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

หลักการติดต่อสื่อสารกันระหว่างเอเจนต์ หรือที่เรียกว่า Multi-agent Communication ได้มีการศึกษาวิจัยอย่างกว้างขวางในสาขาปัญญาประดิษฐ์ แนวคิดดังกล่าวได้นำมาประยุกต์ใช้งานกับโปรแกรมแอฟพลิเคชั่นบนอินเตอร์เน็ตไม่นานมานี้ ความคิดนี้จะทำให้ความสามารถของเว็บไซต์ (web site) สูงขึ้น ในลักษณะที่ว่าโปรแกรมบนอินเตอร์เน็ตสามารถพูดคุยติดต่อสื่อสารกันได้ ทำให้เกิดความสามารถใหม่ๆขึ้นมาได้

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อนำหลักการ การติดต่อสื่อสารในรูปลักษณะของสถาปัตยกรรมไคลเอ็นท์–เซิร์ฟเวอร์ (Client-server Architecture) มาอธิบายลักษณะการติดต่อสื่อสาร (communication) ด้วยภาษาทางลอจิก ซึ่งนำไปสู่การพัฒนาภาษาเพื่อใช้ในการสื่อสารต่อไป
2. เพื่อให้สามรถประยุกต์ใช้งานโปรแกรมภาษาโปรล็อกมาใช้งานบนเว็บเป็นเสมือน CGI โดยผ่านโปรแกรม โปรเว็บเซิร์ฟเวอร์ (Proweb Server) ได้
3. เพื่อเรียนรู้การใช้ภาษาโปรล็อกมาเขียนบราวเซอร์ในฐานะเอเจนต์ และเว็บแอปพลิเคชั่น (web application) ที่ฝั่งเซิร์ฟเวอร์ซึ่งทำหน้าที่เป็นเอเจนต์เช่นกัน เพื่อให้นำมาใช้ในการสื่อสารระหว่างกันผ่านเว็บได้

1.3 ขอบเขตของโครงงาน

1. อธิบายหลักการติดต่อสื่อสารในรูปแบบไคลเอ็นท์–เซิร์ฟเวอร์ (Client-server) โดยใช้ภาษาโปรล็อก
2. นำกระบวนการสื่อสารในรูปแบบไคลเอ็นท์–เซิร์ฟเวอร์ (Client-server) มากำหนดและสร้างรูปแบบกระบวนการสื่อสารของเอเจนต์ ด้วยภาษาโปรล็อก

1.4 วิธีการดำเนินงาน

1. ศึกษาภาษาโปรล็อก (Prolog)
2. ศึกษาโพรโตคอล (Protocol) ที่ใช้ในการติดต่อสื่อสาร เช่น Hypertext Transfer Protocol (HTTP), TCP/IP Protocol
3. ศึกษาโปรแกรมโปรเว็บ (Proweb)
4. เขียนโปรแกรมขึ้นมาเพื่อทดลองการทำงานของ TCP/IP Protocol และ Hypertext Transfer Protocol (HTTP)

1.5 เนื้อหาโดยรวม

ในบทที่ 2 อธิบายถึงทฤษฎีพื้นฐานของโปรล็อก (Prolog), TCP/IP Protocol, Hypertext Transfer Protocol (HTTP), สถาปัตยกรรมไคลเอ็นท์–เซิร์ฟเวอร์ (Client-server Architecture), หลักการทำงานของบราวเซอร์กับเว็บเซิร์ฟเวอร์, เอเจนต์ (Agent) และ SMTP (Simple Mail Transfer Protocol)

บทที่ 3 อธิบายโครงสร้างของระบบการติดต่อสื่อสารกันระหว่างเอเจนต์ รวมทั้งโครงสร้างของโปรล็อกบราวเซอร์ และเซิร์ฟเวอร์

บทที่ 4 อธิบายรายละเอียดการทำงานของโปรล็อกบราวเซอร์ และการติดตั้ง (setup) โปรล็อกบราวเซอร์

บทที่ 5 อธิบายรายละเอียดการทำงานของเซิร์ฟเวอร์ และการติดตั้งเซิร์ฟเวอร์

บทที่ 6 อธิบายขั้นตอนการใช้งานโปรล็อกบราวเซอร์

บทที่ 2
หลักการและทฤษฎี

2.1 ภาษาโปรล็อก (Prolog)

2.1.1 โปรแกรมภาษาโปรล็อก

โปรล็อกโปรแกรม คือ เซตของคลอสจำกัด (definite clause) ซึ่งมีรูปแบบประโยคดังต่อไปนี้

A :- B1,...,Bn

เมื่อ n >= 0 ใช้สัญลักษณ์ ", " เป็นตัวเชื่อมมีความหมายว่า "และ" ถ้าคลอส (clause) อ่านจากซ้ายไปขวา สัญลักษณ์ " :- " จะอ่านว่า "ถ้า" ("if") แต่ถ้าอ่านจากขวาไปซ้ายสัญลักษณ์ " :- " จะอ่านว่า"หมายความว่า" ("implies") A และ B เป็นอะตอมทางลอจิก (logic atom) ซึ่งถูกเรียกว่าเป้าหมายหรือโกล (goal) จากประโยคข้างต้นมีความหมายว่า การที่จะทำให้โกล A สำเร็จได้ เกิดจากการทำโกล B ทุกๆ อัน เรียก A ว่าเป็นหัว (head) ของคลอส และ B1,...,Bn ว่าเป็นตัว (body) ของคลอส และแต่ละ Bi เรียกว่าสับโกล (sub goal) หากว่ารูปแบบข้างต้นไม่มี B นั่นหมายความว่า A เป็นจริงโดยไม่มีเงื่อนไข สามารถเขียนได้อยู่ในรูป A :- true เมื่อ true หมายความว่าเป็นจริงโดยไม่มีเงื่อนไข (empty goal)

ตัวอย่างคลอส

h(X,Y) :- q(X,Y)

หมายถึง

$\forall$ X,Y [h(X,Y) :- q(X,Y)]

เพรดดิเคท (predicate) คือ กลุ่มของคลอสซึ่งมีหัว (head) ที่มีฟังก์เตอร์ (functor) เหมือนกัน และจำนวนตัวแปร (arity) เท่ากัน ตัวอย่าง h(X,Y) มีฟังก์เตอร์คือ h และจำนวนตัวแปรเท่ากับสอง

อะตอม (atom) คือ ฟังก์เตอร์ที่มีจำนวนตัวแปรเท่ากับศูนย์

2.1.2 กลไกการทำงานและการแบ็คแทรกค์ (The execution and backtracking mechanism)

โปรล็อกจะพิจารณาคลอสที่สามารถยูนิฟายได้ตามลำดับที่พบในโปรแกรม เมื่อไม่สามารถหาได้ มันจะทำการแบ็คแทรกซ์กลับไปที่จุดที่เป็นทางเลือกล่าสุด ดังโปรแกรมต่อไปนี้

```
possible_pair(X,Y) :- boy(X), girl(Y).

boy(john).
boy(marmaduke).
boy(bertram).

girl(griselda).
```

```
girl(ermintrude).
girl(brunhilde).

?- possible_pair(X,Y).
```

โปรแกรมข้างบนเป็นการจับคู่เด็กผู้ชายและเด็กผู้หญิงว่าสามารถจับคู่ได้อย่างไรบ้าง ซึ่งจะได้คำตอบดังนี้

```
X = john, Y = griselda ;
X = john, Y = ermintrude ;
X = john, Y = brunhilde ;
X = marmaduke, Y = griselda ;
X = marmaduke, Y = ermintrude ;
X = marmaduke, Y = brunhilde ;
X = bertram, Y = griselda ;
X = bertram, Y = ermintrude ;
X = bertram, Y = brunhilde ;
```

การทำงานของโปรแกรมจะเริ่มจากการหาโกล (goal) boy(X) ซึ่งคลอส (fact) แรกที่พบก็คือ boy (john). ดังนั้นจะได้คำตอบสำหรับโกลแรกเป็น john จากนั้นก็ไปหาโกลถัดไปคือ girl(Y) ซึ่งจะได้คำตอบเป็น griselda เพราะฉะนั้นคำตอบแรกของ ?- possible_pair(X,Y). จึงได้เป็น X = john, Y = griselda และโปรล็อกก็จะพยายามหาว่ายังมี girl อื่นอีกหรือไม่ ถ้ามีก็จะได้เป็นคำตอบที่สอง และสามต่อไป ถ้าไม่มีมันก็จะทำการแบ็คแทรกซ์กลับไปหาโกล boy(X) อีกครั้ง ซึ่งจะได้คำตอบเป็น boy อันถัดมา จากนั้นก็จะไปหาโกล girl(Y) อีกครั้ง โดยเริ่มจาก girl แรกใหม่ ทำอย่างนี้ไปเรื่อยๆ จนไม่สามารถหาโกล boy(X) ได้อีก

2.1.3 การควบคุม (Control)

ลำดับของโกลในคลอสจะเป็นตัวกำหนดลำดับของการประมวลผล ลำดับของคลอสในโปรแกรมจะเป็นตัวกำหนดลำดับในการเลือกคลอส พิจารณาโปรแกรมต่อไปนี้

```
person(adam).
person(X). :- mother(X,Y).
person(eve).

Mother(cain, eve).
Mother(abel, eve).
Mother(jabal, adah).
```

Mother(tubalcain, zillah).

ถ้าเราถามคำถามว่า ?- person(X).

จะได้คำตอบดังนี้ X = adam ;

X = cain ;

X = abel ;

X = jabal ;

X = tubalcain ;

X = eve ;

โปรล็อกมีสิ่งอำนวยความสะดวกสำหรับการควบคุมลำดับได้แก่สัญลักษณ์คัท (cut) " ! "

คัทของภาษาโปรแกรมภาษาโปรล็อกเปรียบเสมือนคำสั่ง "go to" ซึ่งเป็นการควบคุมระดับต่ำ การใช้คัทอาจทำให้โปรแกรมไม่มีโครงสร้าง และเข้าใจยาก คล้ายกับ "go to" ที่สามารถใช้อิมพลีเมนท์การควบคุมระดับสูงเช่น "while" และ "repeat" ลูป ในการแปลงโปรแกรมภาษา คัทสามารถใช้นิยามการควบคุมระดับสูงของเพรดิเคทภาษาโปรล็อกได้

โครงสร้างหนึ่งที่เป็นการนิเสธ (negation) ได้แก่ "not" โปรซีเยอร์ not(A) จะเป็นโกลที่สำเร็จ ก็ต่อเมื่อ A ล้มเหลว โดย "not" จะสามารถนิยามได้จากคัทดังต่อไปนี้

```
not(P) <- P, !, fail .
not(P) .
```

เมื่อ "fail" ที่เป็นโกลที่ล้มเหลวเสมอ โปรแกรม not จะใช้ตัวแปรเมตา (meta-variable) ซึ่งเป็นคุณสมบัติของโปรล็อกที่อนุญาตให้โกลถูกพิจารณาอย่างไดนามิก (dynamic) ในขณะประมวลผล

เครื่องหมายต่อไปของโปรล็อกคือ การใช้สัญลักษณ์ " ; " แทนคำสั่ง or พิจารณาโปรแกรมต่อไปนี้

```
grandfather (X,Y) <- father(X,Z), (father(Z,Y) ; mother(Z,Y)) .
```

จะมีความหมายของการทำงานเหมือนกับโปรแกรมต่อไปนี้

```
grandfather (X,Y) <- father(X,Y), parent(Z,Y) .

parent(X,Y) <- father(X,Y) .
parent(X,Y) <- mother(X,Y) .
```

การควบคุมอื่นๆ ที่มีประโยชน์ในภาษาโปรล็อกได้แก่ if-then-else คือ p -> q ; r คือหาก p สำเร็จให้ไปทำ q มิเช่นนั้นจะทำ r

2.1.4 ไซด์อีเฟค (Side-effects)

โปรล็อกเป็นภาษาฟังก์ชันโดยธรรมชาติของมัน และมีหลายแอพพลิเคชั่น ซึ่งต้องการโครงสร้างข้อมูลเบื้องต้นเช่น สแตกและคิว สามารถอิมพลีเมนท์อย่างมีประสิทธิภาพโดยไม่มีไซด์เอฟเฟค แต่อย่างไรก็ตาม บางครั้งมันก็จำเป็นในการพัฒนาสเตท (state) ของระบบ โปรล็อกมีสองเพรดดิเคท คือ assert(X) คือเพิ่มคลอส X ไปที่โปรแกรม (การ assert มีสองแบบคือ asserta คือการเพิ่มเข้าไปเป็นคลอสแรกของฐานข้อมูล และ assertz คือเพิ่มเข้าไปเป็นคลอสสุดท้ายของฐานข้อมูล) ส่วน retract(X) เป็นการลบคลอส X ออกจากฐานข้อมูล

ไซด์เอฟเฟคถูกใช้โดยการพัฒนาระบบเพื่อพัฒนาโปรแกรมในการตีความหมาย มันสามารถนำมาใช้ในการบันทึกคำตอบของการไควรี่ (query) ของผู้ใช้ได้ทำให้ไม่ต้องถามคำถามเดิมซ้ำสองครั้ง

2.1.5 เพรดดิเคทลำดับที่สอง (Second order predicates)

เพรดดิเคทลำดับที่สองสามารถมองโกลเป็นข้อมูล ตัวอย่างของเพรดดิเคทประเภทนี้ได้แก่ bagof โกล bagof(X,P,S) ให้ค่า S เป็นลิสของอินสแตนซ์ของ X สำหรับแต่ละการประมวลผลของโกล P ตัวอย่างเช่น

```
?- bagof ( (X-Y),(member(X,[1,2,3]),member(Y,[a,b,c,d]) ),S) .
S = [1-a, 1-b, 1-c, 1-d, 2-a, 2-b, 2-c, 2-d, 3-a, 3-b, 3-c, 3-d]
```

2.1.6 เมตาโปรแกรมมิ่งในโปรล็อก (Meta-programming in Prolog)

เมตาโปรแกรม (Meta-program) เป็นโปรแกรมที่สามารถจัดการโปรแกรมอื่น เช่น การทำการวิเคราะห์ ควบคุมการทำงานของโปรแกรม หรือแก้ไขและเพิ่มเติมส่วนต่างๆ ของโปรแกรมได้ พูดอีกนัยหนึ่งก็คือ เมตาโปรแกรมมองโปรแกรมเป็นข้อมูลได้

เมตาโปรแกรมจะประกอบด้วย 2 ระดับ ระดับแรกคือ ระดับเมตา (meta level) ซึ่งทำการควบคุมและจัดการโปแกรมอื่น และระดับที่สองคือระดับวัตถุ (object level) จะมองโปรแกรมเหมือนเป็นข้อมูล

เมตาโปรแกรมมิ่งถูกใช้อย่างกว้างขวางในลอจิกโปรแกรมมิ่ง เช่น ใช้สำหรับเป็นดีบักเกอร์ (debuggers), คอมไพรเลอร์ (compilers) และโปรแกรมแปลงภาษา (program transformer) โดยปกติโปรแกรมหลายๆ โปรแกรมมักจะเกี่ยวข้องกับเมตาโปรแกรมมิ่ง

ตัวอย่างของเมตาอินเตอร์พรีทเตอร์อย่างง่ายๆ ที่เรียกว่า Vanilla meta-interpreter แสดงดังรูป

```
solve(true) .
solve((A,B)) :- solve(A), solve(B) .
solve(A) :- clause(A,B), solve(B) .
solve(A) :- call(A).
```

รูปที่ 2-1 Vanilla Meta-Interpreter

2.1.7 ไลบรารี (Library) มาตรฐานในการติดต่อสื่อสารของภาษาโปรล็อก

ไลบรารีมาตรฐานในการติดต่อสื่อสารของภาษาโปรล็อกนี้อนุญาตให้โปรแกรมภาษาโปรล็อกสามารถติดต่อสื่อสารกับโปรแกรมอื่นผ่านทางอินเตอร์เน็ต (Internet) ได้

ไลบรารีในการติดต่อสื่อสารของภาษาโปรล็อกมีดังนี้

- TCP/IP Sockets Library

  ไลบรารีนี้ให้เพรดิเคทที่จำเป็นในการติดต่อสื่อสารผ่านเน็ตเวอร์ก (network) สำหรับโปรแกรมที่เขียนด้วยภาษาโปรล็อก ถ้าใช้ไลบรารีนี้จะสามารถสร้างช่องทางในการติดต่อสื่อสารระหว่างโปรแกรมที่อยู่ต่างเครือข่ายได้, สามารถรับ-ส่งข้อมูลระหว่างกระบวนการที่ติดต่อสื่อสารกัน โดยมีหรือไม่มีเวลารอคอย (timeout) ก็ได้ แต่ไลบรารีนี้จะสนับสนุนการส่งข้อมูลเป็นสตรีม (stream) เท่านั้น

- HTTP Client Library

  HTTP (HyperText Transfer Protocol) เป็นเน็ตเวอร์กโพรโตคอลที่ใช้กับเว็บ (Web) สามารถเคลื่อนย้ายไฟล์ และข้อมูลอื่นๆ หรือที่เรียกว่า รีซอร์ส (resources) บนเว็บได้ โดยการใช้ต้องใช้งานร่วมกับ TCP/IP Library

  ตัวอย่างลักษณะงานที่ต้องใช้ไลบรารีนี้

  - ดาวน์โหลดไฟล์ (download files) จากเว็บเซิร์ฟเวอร์
  - สั่งข้อมูลไปให้ CGI ทำงาน
  - อื่นๆ ที่เกี่ยวกับเว็บ

- EMAIL Library

  อีเมล์ (Email) เป็นสิ่งที่ใช้กันอย่างกว้างขวางบนอินเตอร์เน็ต ข้อมูลในอีเมล์นั้นโดยทั่วไปจะใช้ในการสื่อสารระหว่างบุคคล แต่เรายังสามารถนำมาใช้เพื่อการสื่อสารระหว่างโปรแกรมได้ด้วย ไลบรารีนี้ได้มีการกำหนดเพรดิเคทซึ่งทำให้โปรแกรมสามารถรับ-ส่ง และประมวลผลข้อมูลในอีเมล์ได้ แต่ต้องถูกต้องตามมาตรฐานของระบบอินเตอร์เน็ต

  สิ่งที่ไลบรารีนี้สามารถทำได้

  - รับ-ส่งอีเมล์ที่มีไฟล์อื่นแนบมาด้วย หรือไม่มีก็ได้
  - สามารถเลือกเฉพาะอีเมล์ที่ต้องการ ได้ทั้งก่อนและหลังดาวน์โหลด

2.2 TCP/IP Protocol

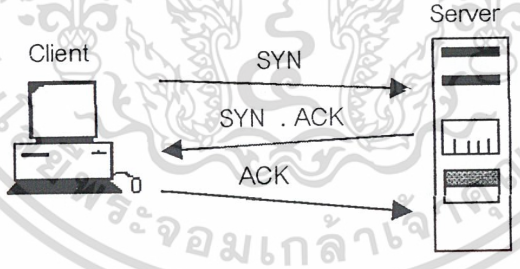
โพรโตคอล TCP เป็นโพรโตคอลในชั้นนำส่งข้อมูล (Transport Layer) ซึ่งเป็นการเชื่อมต่อแบบต่อเนื่อง (connection-oriented) และได้รับการออกแบบมาให้เป็นโพรโตคอลที่ไว้วางใจได้ (reliable) เพื่อใช้ในการสื่อสารผ่านระบบเครือข่ายทั่วไปที่อาจมีความผิดพลาดเกิดขึ้นในขณะนำส่งข้อมูลอยู่เสมอ

บริการ TCP เริ่มต้นจากการที่สร้างส่วนควบคุมการติดต่อระหว่างกันที่เรียกว่า ซ็อกเก็ต (socket) ซ็อกเก็ตแต่ละตัวจะมีหมายเลขที่อยู่ประกอบด้วย 2 ส่วนคือหมายเลข IP ของโฮสต์และหมายเลขขนาด 16 บิตที่ใช้ภายในโฮสต์นั้นๆ เรียกว่า พอร์ต (port)

พอร์ตหมายเลขต่ำกว่า 256 เป็นพอร์ตที่ใช้งานพื้นฐานทั่วไปซึ่งถูกสำรองใช้งานเฉพาะอย่างเท่านั้น เช่น โปรแกรม FTP ต้องระบุพอร์ตหมายเลข 21 หรือการเชื่อมต่อผ่าน TELNET จะต้องระบุพอร์ตหมายเลข 23 หรือถ้าเป็น HTTP ต้องใช้พอร์ตหมายเลข 80 พอร์ตใช้งานพื้นฐานอื่นๆอธิบายไว้ใน RFC 1700

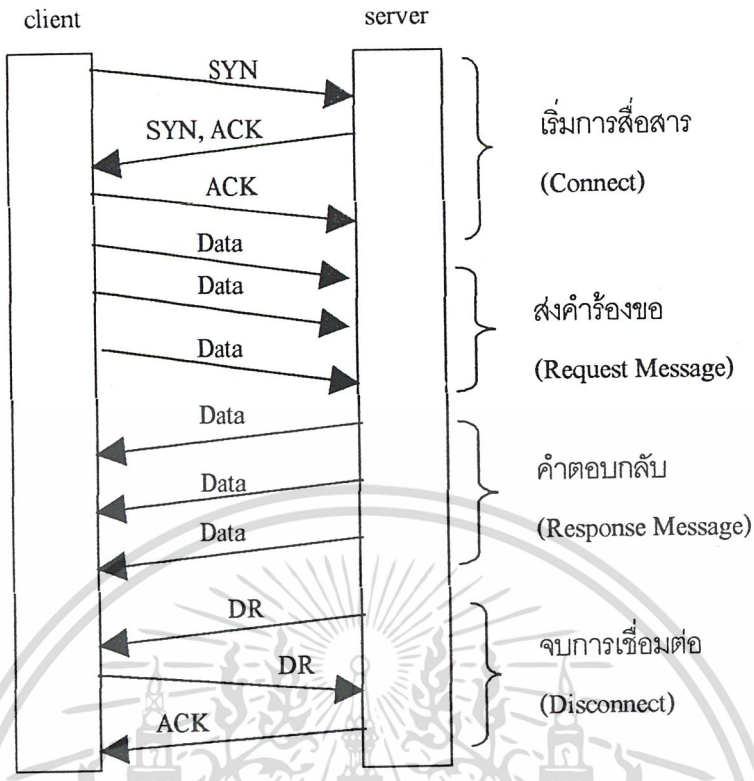
การเชื่อมต่อของ TCP เป็นการเชื่อมต่อแบบสองทาง (full-duplex) และเชื่อมต่อแบบจุดต่อจุด (point-to-point) หมายความว่าการเชื่อมต่อจะมีโฮสต์อยู่เพียง 2 โฮสต์ที่ปลายสายแต่ละข้างซึ่งสามารถส่งข้อมูลสวนทางกันได้ตลอดเวลา TCP ไม่สนับสนุนการเชื่อมต่อหลายจุด (multicasting) และแบบกระจายข่าว (broadcasting)

วิธีการเชื่อมต่อของ TCP เริ่มต้นจากการเชื่อมต่อซึ่งเป็นแบบ การจับมือร่วมสามขั้นตอน (three-way handshake) โดยต้องมีฝ่ายหนึ่งซึ่งปกติจะเป็นผู้ให้บริการ (server) เป็นฝ่ายรอคอยสัญญาณ SYN จากผู้ขอใช้บริการ (client) โดยสัญญาณที่ส่งไปจะประกอบไปด้วย หมายเลข IP และพอร์ตที่ใช้, ขนาดสูงสุดของเซกเมนต์ และจบลงด้วยการใช้สัญญาณ ACK เพื่อยืนยันการเชื่อมต่อ



รูปที่ 2-2 แสดงวิธีการเชื่อมต่อของ TCP

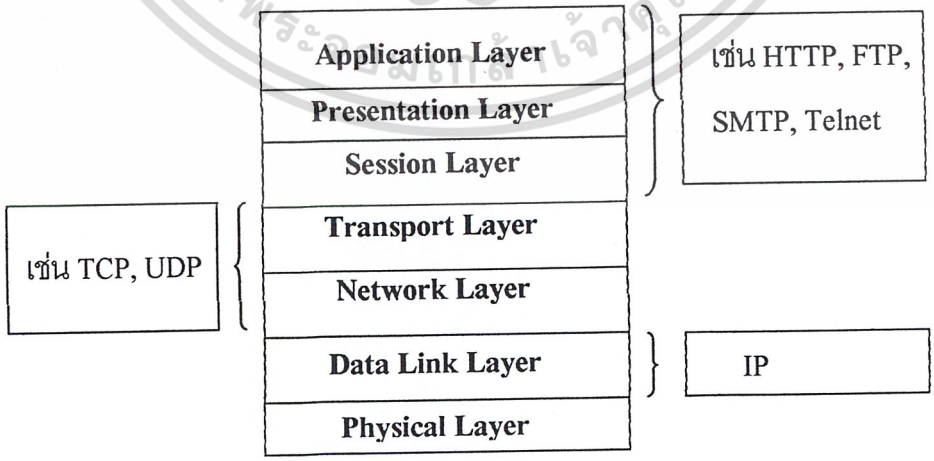
ในส่วนของการส่งข้อมูล ข้อมูลที่จะถูกส่งใน TCP จะถูกแบ่งออกเป็นหลายๆ เซกเมนต์ ซึ่งขนาดของเซกเมนต์จะมีขนาดไม่เกิน 64 Kbyte (หรือ 65,535 ไบต์) และจะต้องไม่เกินค่ากำหนดขนาดสูงสุดของแพ็กเก็ตของแต่ละระบบเครือข่ายที่เรียกว่า MTU (Maximum transfer unit) ค่าของ MTU อาจมากหรือน้อยกว่า 64 Kbyte ก็ได้



รูปที่ 2-3 ขั้นตอนในการรับ-ส่งข้อมูล

2.3 HTTP (Hypertext Transfer Protocol)

HTTP เป็น Protocol ในชั้น Application Layer ของ OSI Model การที่จะ implement HTTP จะต้องเรียกใช้บริการของ TCP/IP ซึ่งเป็นโพรโตคอลในชั้น Transport Layer



รูปที่ 2-4 OSI Model

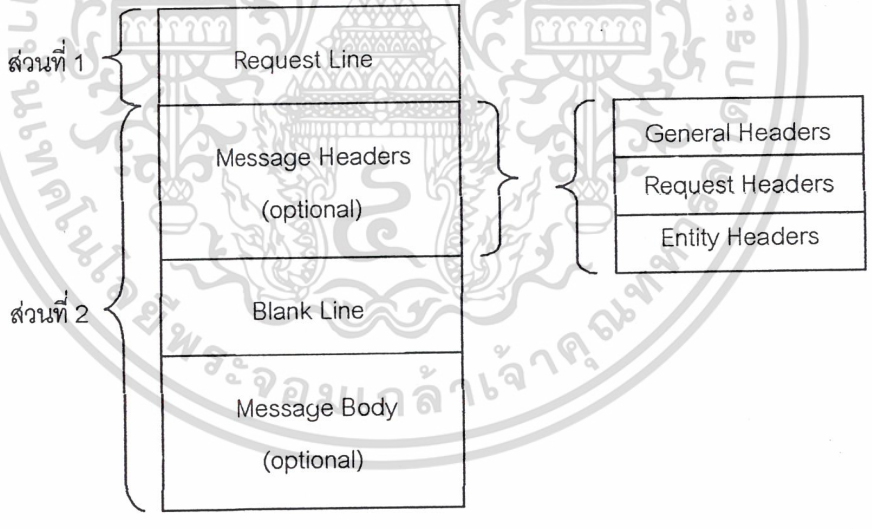
HTTP (Hypertext Transfer Protocol) เป็นโพรโตคอลแบบไคลเอ็นท์-เซิร์ฟเวอร์ มีการเชื่อมต่อแบบไม่ถาวร (connectionless) คือ เชื่อมต่อเฉพาะระหว่างการทำงานหนึ่งงานเท่านั้น เช่น เมื่อไคลเอ็นท์ขอบริการไปที่เซิร์ฟเวอร์ เครื่องเซิร์ฟเวอร์ก็จะทำการให้บริการ เสร็จแล้วจะตัดการติดต่อทันที เพื่อให้ไคลเอ็นท์อื่นสามารถติดต่อเข้ามาขอบริการได้อีก HTTP เป็นโพรโตคอลที่ใช้ในการร้องขอ (requests) และตอบสนอง (response) อย่างง่าย เมื่อไคลเอ็นท์สร้างการติดต่อกับเซิร์ฟเวอร์แล้ว ไคลเอ็นท์ก็จะส่งคำร้องขอไปยังเซิร์ฟเวอร์ตัวนั้น เซิร์ฟเวอร์จะตอบกลับด้วยข้อมูลที่ถูกร้องขอ

คำร้องขอ (requests) ของไคลเอ็นท์ประกอบด้วย 2 ส่วนดังนี้

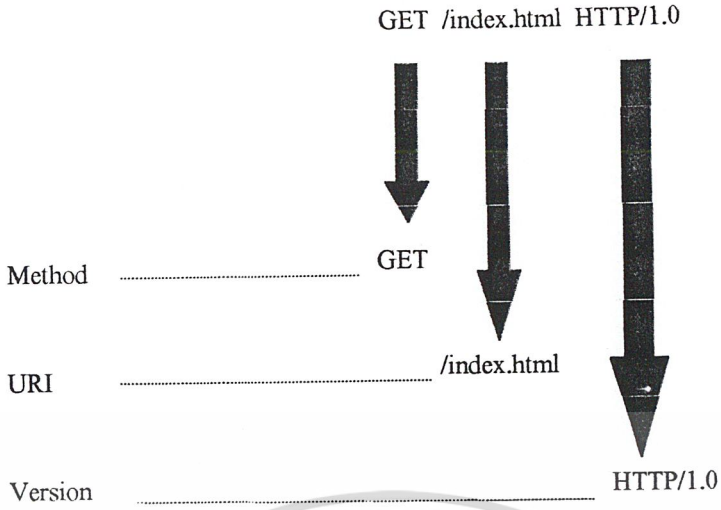
ส่วนที่ 1 ประกอบด้วย

- วิธีการร้องขอซึ่งจะเป็นตัวระบุการทำงาน
- URL ของแหล่งข้อมูล (resource) เช่น ไฟล์ (file) ที่ต้องการให้มันทำงาน
- เวอร์ชั่นของโพรโตคอล เช่น HTTP 1.0

ส่วนที่ 2 ประกอบด้วย ส่วนหัว (headers) ซึ่งให้ข้อมูลแก่เซิร์ฟเวอร์ เช่น ข้อมูลชนิดไหนที่ไคลเอ็นท์สามารถใช้หรือจัดการได้, ID ของไคลเอ็นท์ (client's identity), URL ของเอกสารที่เป็นผู้ส่งคำร้องขอ และเนื้อหา (body content)



รูปที่ 2-5 ส่วนประกอบของคำร้องขอ



รูปที่ 2-6 ส่วนประกอบใน Request-Line

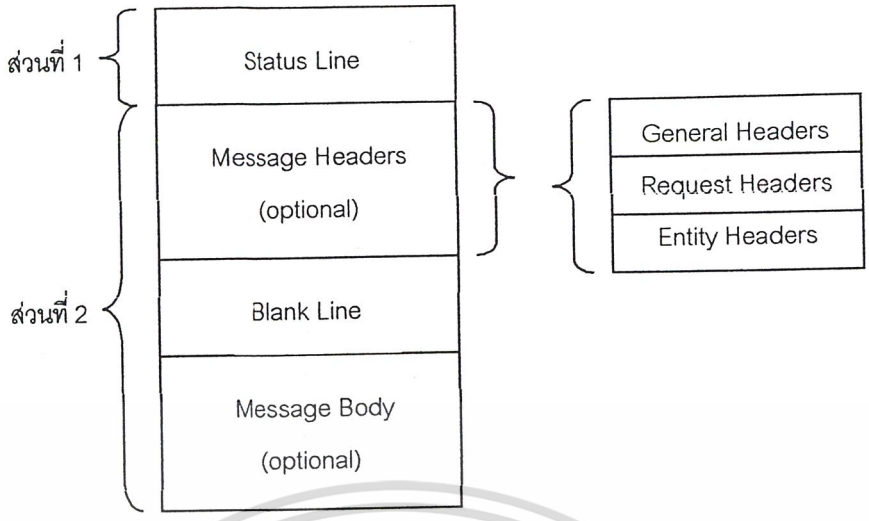
วิธีการขอบริการของไคลเอ็นท์ที่ใช้กันมากโดยปกติมี 3 วิธี คือ

- การขอบริการแบบขอข้อมูล (Method GET) ใช้ในการเรียกแหล่งข้อมูลมา อนุญาตให้ไคลเอ็นท์ส่งข้อมูลมาให้เซิร์ฟเวอร์ได้ในปริมาณที่จำกัด
- การขอบริการแบบถามรายละเอียด (Method HEAD) ใช้ร้องขอข้อมูลเฉพาะส่วนรายละเอียดของแหล่งข้อมูล
- การขอบริการแบบให้ข้อมูล (Method POST) อนุญาตให้ไคลเอ็นท์ส่งข้อมูลมาให้เซิร์ฟเวอร์ได้โดยไม่จำกัดขนาด

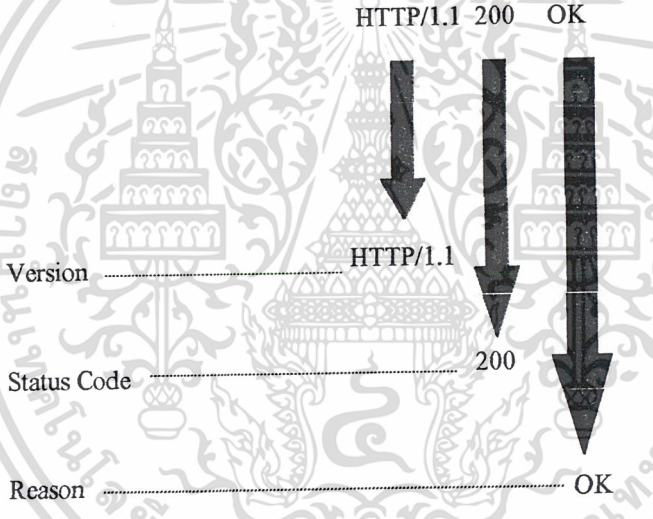
คำตอบกลับจากเซิร์ฟเวอร์ (server's response) ประกอบด้วย 2 ส่วนดังนี้

ส่วนที่ 1 เป็นส่วนที่แสดงสถานะ (status line) ประกอบด้วย เวอร์ชั่นของโพโตคอลของเซิร์ฟเวอร์ (server's protocol version), รหัส (code) ที่บอกถึงสถานะของคำร้องขอ และคำอธิบายเกี่ยวกับสถานะ

ส่วนที่ 2 เป็นข้อความ (message) ประกอบด้วย เมตาอินฟอร์เมชั่น (meta-information) ของแหล่งข้อมูล และเนื้อหา (body content) โดยทั่วไปเมตาอินฟอร์เมชั่นหมายความรวมไปถึง วันที่ที่แหล่งข้อมูลถูกร้องขอ, ชื่อและเวอร์ชั่นของเซิร์ฟเวอร์ที่แหล่งข้อมูลนั้นอยู่, วันเวลาที่แหล่งข้อมูลนั้นถูกแก้ไขเปลี่ยนแปลงครั้งสุดท้าย, ชนิดของเนื้อหา (content type) เช่น text / html และความยาวของเนื้อหา (content length) ถ้าเนื้อหาเป็นชนิด text / html แสดงว่าเป็นเอกสาร HTML



รูปที่ 2-7 ส่วนประกอบของคำตอบกลับจากเซิร์ฟเวอร์



รูปที่ 2-8 ส่วนประกอบใน HTTP Status-Line

ตัวอย่างวิธีการขอบริการแบบขอข้อมูล (A GET Request Example)

โดยทั่วไปการทำงานของลิงค์ (link) ที่พบเห็นบนเว็บบราวเซอร์ (web browser) คือ การเรียกข้อมูลและนำมาแสดงบนหน้าจอ เมื่อผู้ใช้ (user) คลิ๊กลงบนไฮเปอร์เท็คลิงค์ (hypertext link) บนเว็บบราวเซอร์ บราวเซอร์จะเปลี่ยน URL ที่ระบุอยู่ในลิงค์เป็นการขอบริการแบบขอข้อมูล โดยใช้เซิร์ฟเวอร์โฮสต์ และรากตามที่ระบุไว้ใน URL แล้วส่งคำร้องขอไปยังเซิร์ฟเวอร์ ตัวอย่างเช่น ถ้าผู้ใช้ คลิ๊กลงไปบนลิงค์ข้างล่างนี้

<A HREF="http://www.w3.org/Addressing/Addressing.html"> Web addressing </A>

การขอบริการแบบขอข้อมูลข้างล่างนี้จะถูกส่งไปหลังจากได้มีการเชื่อมต่อไปยังโฮสต์ www.w3.org

GET /Addressing/Addressing.html HTTP/1.0

/* a blank line here */

และเซิร์ฟเวอร์จะตอบกลับมาดังนี้

HTTP/1.0 200 Document follows

Server: CERN/3.0A

Date : Thu, 31 Jul 1997 00 : 59 : 05 GMT

Content-Type : text / html

Content-Length : 12998

Last-Modified : Wed, 04 Jun 1997 20 : 17 : 41 GMT

/* a blank line here */

<HTML>

/* the rest of the requested document is here */

รหัสบอกสถานะ (status code) เท่ากับ 200 เป็นการบอกว่าการร้องขอประสบความสำเร็จ (successful) ถ้าการร้องขอล้มเหลว (failure) รหัสบอกสถานะรหัสอื่นจะเป็นตัวบอกเหตุผลของการล้มเหลว เช่น มีการพบข้อผิดพลาด (error) บางอย่างขึ้น หรือแหล่งข้อมูลที่ร้องขอไปหาไม่พบ

ตัวอย่างวิธีการขอบริการแบบถามรายละเอียด (A HEAD Request Example)

การขอบริการแบบถามรายละเอียดตามตัวอย่างข้างล่างจะทำงานเหมือนการขอบริการแบบขอข้อมูล

HEAD /Addressing/Addressing.html HTTP/1.0

/* a blank line here */

แต่จะเรียกข้อมูลเฉพาะส่วนที่เป็นเมตาอินฟอร์เมชั่น (meta-information) จากเซิร์ฟเวอร์เท่านั้น ไม่มีเนื้อหา (body content) มาด้วย

เซิร์ฟเวอร์จะตอบกลับมาดังนี้

HTTP/1.0 200 Document follows

Server: CERN/3.0A

Date : Thu, 31 Jul 1997 00 : 59 : 05 GMT

Content-Type : text / html

Content-Length : 12998

Last-Modified : Wed, 04 Jun 1997 20 : 17 : 41 GMT

ตัวอย่างการส่งไควรี่ (Query Submission Examples)

การขอบริการแบบขอข้อมูลสามารถส่งไควรี่ (query) ไปยังโปรแกรมที่อยู่บนเครื่องเซิร์ฟเวอร์ โดยการใส่ไควรี่ลงไปใน URL ได้เลย

ตัวอย่าง

http://www.altavista.yellowpages.com.au/cgi-bin/query?mss=simple&pg=q&q=logicweb

หลังจากได้มีการเชื่อมต่อไปยังโฮสต์ www.altavista.yellowpages.com.au แล้ว การขอบริการข้างล่างจะถูกส่งไป

GET /cgi-bin/query?mss=simple&pg=q&q=logicweb HTTP/1.0

การขอบริการแบบขอข้อมูลถูกสร้างขึ้นเมื่อผู้ใช้ใส่ URL แบบ URL encoding query หรือเมื่อผู้ใช้ใส่ข้อมูลลงในฟอร์ม (form) บนเว็บบราวเซอร์ และส่ง (submits) ข้อมูลไปให้โปรแกรมที่อยู่บนเครื่องเซิร์ฟเวอร์ในราก (path) /cgi-bin/query

วิธีการขอบริการแบบขอข้อมูลไควรี่สตริง (Qurey String) ที่ส่งไปกับรากของ URL จะถูกจำกัดจำนวนข้อมูลที่สามารถส่งได้ เพราะความยาวของ URL มีความยาวจำกัด (โดยปกติจะมีความยาว 255 ตัวอักษร) แต่วิธีการขอบริการแบบให้ข้อมูลอนุญาตให้ผู้ใช้ส่งข้อมูลความยาวเท่าไหร่ก็ได้ รูปแบบข้างล่างนี้เป็นตัวอย่างการขอบริการแบบให้ข้อมูล ซึ่งส่งข้อมูลขนาด 10 ไบต์ (bytes)

```
POST /cgi-bin/post-query HTTP/1.0
Content-type : application/x-www-form-urlencoded
Content-length : 10
        /* a blank line here */
int=hello!
```

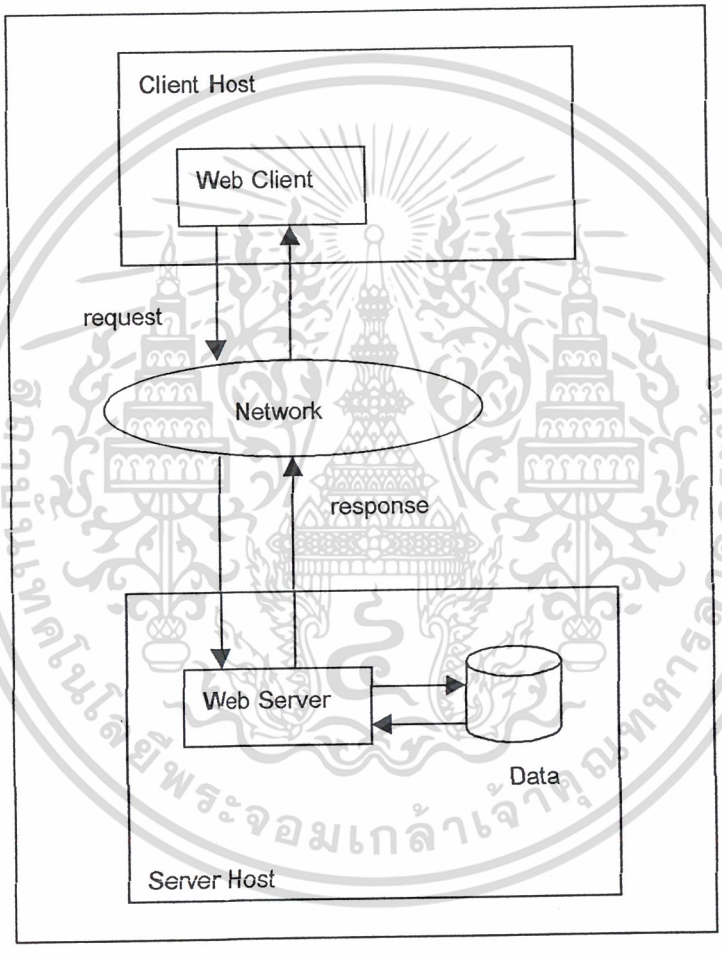
วิธีการส่งข้อมูลในลักษณะนี้ถูกเรียกว่า CGI (Common Gateway Interface) และโปรแกรมที่ทำงานอยู่ในฝั่งเซิร์ฟเวอร์ถูกเรียกว่า ซีจีไอสคริปต์ (CGI Script) หรือ ซีจีไอโปรแกรม (CGI Program)

2.4 สถาปัตยกรรมไคลเอ็นท์–เซิร์ฟเวอร์ (Client-server Architecture)

สถาปัตยกรรมไคลเอ็นท์–เซิร์ฟเวอร์ เป็นหลักการพื้นฐานที่ใช้ในการติดต่อสื่อสารบนอินเตอร์เน็ต (Internet) หรือเว็บ (web) โดยเซิร์ฟเวอร์ (server) คือ โปรแกรม (program) ที่รับการร้องขอต่างๆ (requests) ผ่านทางเน็ตเวอร์ก (network) และทำงานตามที่ได้รับการร้องขอให้สำเร็จ แล้วตอบกลับ

ไปยังผู้ร้องขอ เซิร์ฟเวอร์ควบคุมการเข้าถึงข้อมูลข่าวสารบนอินเตอร์เน็ต เป็นตัวแทนของไคลเอ็นท์ในการเข้าถึงข้อมูล และจัดการกับข้อมูลตามที่ได้รับการร้องขอ และส่งข้อมูลมายังไคลเอ็นท์ ส่วนไคลเอ็นท์ (client) คือ โปรแกรมที่สามารถส่งคำร้องขอไปยังเซิร์ฟเวอร์ และรอการตอบกลับของเซิร์ฟเวอร์ แล้วตอบรับไปยังเซิร์ฟเวอร์ตัวนั้น ไคลเอ็นท์สามารถรับอินพุท (input) หรือคำร้องขอจากผู้ใช้ (user) และแสดงข้อมูล (คำตอบกลับจากเซิร์ฟเวอร์) ให้แก่ผู้ใช้

เซิร์ฟเวอร์และไคลเอ็นท์จะมีลักษณะเฉพาะตัวขึ้นอยู่กับโพรโตคอล (protocol) ที่มันใช้ เว็บเซิร์ฟเวอร์ (web server) และไคลเอ็นท์ติดต่อสื่อสารกันในเน็ตเวอร์กโดยใช้ Hypertext Transfer Protocol (HTTP)



รูปที่ 2-9 สถาปัตยกรรมไคลเอ็นท์-เซิร์ฟเวอร์ของเว็บ

2.4.1 Uniform Resource Locators (URLs)

Uniform Resource Locator (URL) เป็นสตริง (string) ที่อ้างถึงแหล่งข้อมูล (resource) บนอินเตอร์เน็ต ที่ถูกเข้ารหัสเอาไว้ตามกฎ (URL encoding)

URL สำหรับแหล่งข้อมูล (resource) ประกอบด้วยส่วนประกอบต่างๆ ดังต่อไปนี้

1. **อินเตอร์เน็ตโพโตคอล** (Internet protocol) สำหรับการเข้าถึงแหล่งข้อมูล
2. **ชื่อโฮสต์** (host) ที่แหล่งข้อมูลนั้นอยู่

3. (optional) หมายเลขพอร์ต (port number) เป็นหมายเลขที่ถูกกำหนดให้ใช้รับข้อมูลในระดับ TCP ของโพรโตคอล HTTP โดยปกติมักจะเป็นพอร์ต 80
4. ราก (part) ที่ระบุแหล่งข้อมูลที่อยู่บนเครื่องโฮสต์

ตัวอย่าง URL http://www.w3.org/Addressing/Addressing.html URLนี้บอกว่าเอกสาร (document) ที่อยู่ในราก (part) /Addressing/Addressing.html ถูกเรียกมาโดยใช้ HTTP โพโตคอล จากเซิร์ฟเวอร์ที่วิ่งอยู่บนโฮสต์ด้วยแอ็ดเดรส (address) www.w3.org โดยใช้หมายเลขพอร์ทตามดิฟอล์ท (default)

รูปแบบอื่นๆ ของ URL ประกอบด้วย

- relative URLs

  relative URL เป็นส่วนหนึ่งของ URL ที่บอกว่าเอกสารอยู่ในรากไหน ตัวอย่าง relative URL เช่น /Addressing/Addressing.html อยู่ภายใน URL http://www.w3.org/

  URL ที่สมบูรณ์เขียนได้ดังนี้

  http://www.w3.org/Addressing/Adderssing.html
- URLs encoding queries

  URL สามารถใส่ไควรี่ (query) ที่จะส่งไปยังโปรแกรมที่อยู่ที่เครื่องเซิร์ฟเวอร์ โดยเครื่องหมาย " ? " จะเป็นตัวแบ่งแอ็ดเดรสของโปรแกรมออกจากส่วนไควรี่ที่จะส่งไปให้โปรแกรม ตัวอย่าง URL encoding query เช่น

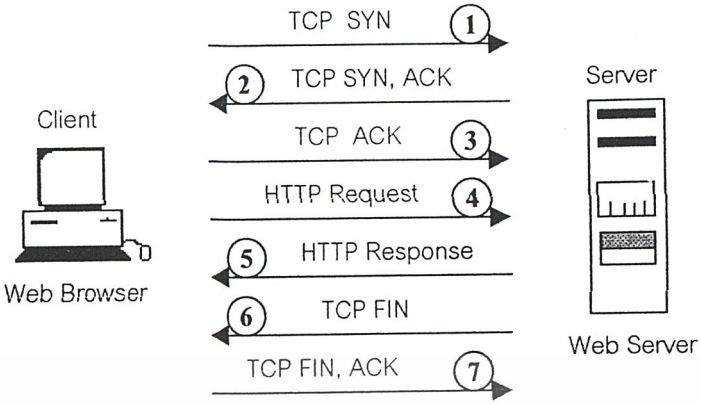
  http://www.altavista.yellowpages.com.au/cgi-bin/query?

  mss=simple&pg=q&q=logicweb

2.5 หลักการทำงานของบราวเซอร์กับเว็บเซิร์ฟเวอร์

สิ่งที่แตกต่างกันอย่างเห็นได้ชัดระหว่างบราวเซอร์กับเว็บเซิร์ฟเวอร์คือ หน้าที่ในการเริ่มการติดต่อสื่อสาร บราวเซอร์เท่านั้นที่จะเป็นคนเริ่มการติดต่อสื่อสาร เซิร์ฟเวอร์อาจจะมีข้อมูลจำนวนมากที่สามารถให้ได้ และสามารถทำงานอย่างอื่นได้อีกหลายอย่าง แต่มันจะทำก็ต่อเมื่อมีการร้องขอจากบราวเซอร์เท่านั้น

บราวเซอร์จะเริ่มการติดต่อสื่อสารโดยการสร้างการเชื่อมต่อของ TCP (TCP Connection) ซึ่งเป็นแบบการจับมือร่วมสามขั้นตอน (three-way handshake) หลังจากการเชื่อมต่อสามขั้นตอนเสร็จแล้ว บราวเซอร์ก็สามารถส่งคำร้องขอ (HTTP Request)ไปที่เว็บเซิร์ฟเวอร์ได้ เว็บเซิร์ฟเวอร์จะทำงานตามที่ได้รับการร้องขอให้สำเร็จ แล้วส่งผลการทำงาน (HTTP Response) มาให้บราวเซอร์ จากนั้นเว็บเซิร์ฟเวอร์จะส่งคำร้องขอจบการเชื่อมต่อของ TCP เว็บเซิร์ฟเวอร์เป็นคนเริ่มส่งคำร้องขอจบการเชื่อมต่อของ TCP เพราะเว็บเซิร์ฟเวอร์รู้ว่าเมื่อไหร่ที่เว็บเซิร์ฟเวอร์ได้ทำงานตามที่บราวเซอร์ร้องขอเสร็จเรียบร้อยแล้ว



รูปที่ 2-10 แสดงขั้นตอนการติดต่อสื่อสารระหว่างบราวเซอร์กับเว็บเซิร์ฟเวอร์

จากรูปมีขั้นตอนต่างๆ ดังนี้

ขั้นตอนที่ 1 บราวเซอร์ส่งสัญญาณ SYN (synchronize) ไปให้เว็บเซิร์ฟเวอร์เพื่อบอกว่าบราวเซอร์ต้องการเริ่มการติดต่อ

ขั้นตอนที่ 2 เว็บเซิร์ฟเวอร์ตอบกลับไปด้วยสัญญาณ SYN, ACK (acknowledge) เพื่อบอกว่ายินดีที่จะเริ่มการติดต่อ

ขั้นตอนที่ 3 บราวเซอร์สร้างการติดต่อสำเร็จโดยการส่งสัญญาณ ACK กลับไปที่เว็บเซิร์ฟเวอร์

ขั้นตอนที่ 4 บราวเซอร์ส่งคำร้องขอไปที่เว็บเซิร์ฟเวอร์

ขั้นตอนที่ 5 เว็บเซิร์ฟเวอร์ส่งผลการทำงานกลับมาให้บราวเซอร์

ขั้นตอนที่ 6 เว็บเซิร์ฟเวอร์ขอจบการเชื่อมต่อโดยส่งสัญญาณ FIN (finished) ไปให้บราวเซอร์

ขั้นตอนที่ 7 บราวเซอร์รับรู้การจบการเชื่อมต่อโดยส่งสัญญาณ FIN, ACK ไปให้เว็บเซิร์ฟเวอร์

2.6 เอเจนต์ (Agent)

เอเจนต์ คือ โปรแกรมที่สามารถรับข้อมูลและเป้าหมายของผู้ใช้ในการทำงานใดๆ แล้วตัวเอเจนต์จะทำการนำข้อมูลที่ได้ พร้อมทั้งรายละเอียดที่เก็บไว้ในตัวมารวมกัน และประมวลผลเพื่อทำงานให้ได้ผลลัพธ์เป็นเป้าหมายตามที่ผู้ใช้ได้ตั้งไว้

เอเจนต์มีลักษณะดังต่อไปนี้

- Autonomy คือ ความเป็นอิสระในการทำงาน สามารถทำงานให้สำเร็จลุล่วงได้ด้วยตนเอง
- Awareness คือ มีความสามารถที่จะรับรู้ และกระทำกับสภาพแวดล้อมของมันได้

- Persistence คือ การทำงานตามที่ได้รับมอบหมายโดยไม่เปลี่ยนแปลงการทำงาน
- Adaptiveness คือ สามารถปรับลักษณะการทำงานของตัวเองและสามารถเรียนรู้สิ่งใหม่ๆ ได้จากสภาพแวดล้อม
- Cooperation คือ มีการร่วมมือกันทำงานเพื่อให้บรรลุวัตถุประสงค์ของตัวเองได้

โครงสร้างของเอเจนต์ ประกอบด้วย เซ็นเซอร์ (sensor), ความรู้ และเอฟเฟคเตอร์ (effectors)

- เซ็นเซอร์ คือ สิ่งที่ใช้รับข้อมูลจากสิ่งแวดล้อม
- ความรู้ คือ เหตุผลที่นำมาใช้ในการประมวลผล
- เอฟเฟคเตอร์ คือ สิ่งที่ใช้ในการแสดงผลจากการประมวลผล

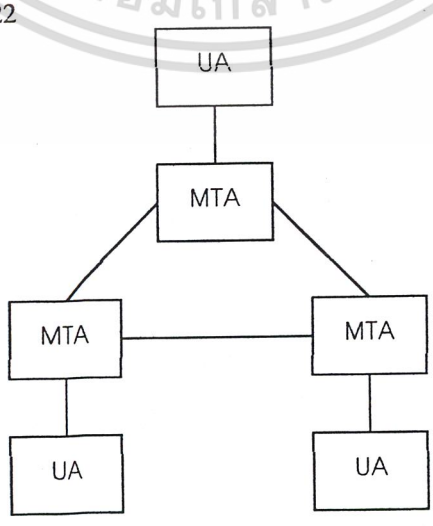
2.7 SMTP (Simple Mail Transfer Protocol)

2.7.1 สถาปัตยกรรมระบบเมล์

ไปรษณีย์อิเล็กทรอนิกส์ ซึ่งนิยมเรียกกันสั้นๆว่า อีเมล์(e-mail) เป็นโปรแกรมประยุกต์ที่แพร่หลายในระบบเครือข่าย ประโยชน์ของอีเมล์ช่วยให้ผู้ใช้ส่งและรับข้อความข้ามเครือข่ายได้ TCP/IP มีโพรโตคอลสนับสนุนการรับส่งเมล์หลายโพรโตคอล แต่โพรโตคอลที่นิยมใช้ในอินเตอร์เน็ตคือ SMTP หน้าที่ของ SMTP คือ กำหนดกรรมวิธีและรูปแบบการนำส่งข้อความระหว่างผู้รับและผู้ส่ง SMTP อาศัย TCP เพื่อลำเลียงจดหมายผ่านพอร์ตเบอร์ 25

ระบบเมล์ที่ใช้ใน TCP/IP มีองค์ประกอบ 2 ส่วน คือ UA (User Agent) และ MTA (Mail Transfer Agent) ทั้ง UA และ MTA เป็นชื่อที่นำมาจากระบบ X.400 ซึ่งเป็นมาตรฐานนานาชาติกำหนดการนำส่งเมล์

UA เป็นโปรแกรมติดต่อกับผู้ใช้ และอำนวยความสะดวกให้ผู้ใช้ เขียน แก้ไขและส่งจดหมาย รวมทั้งการเปิดอ่านจดหมายที่ได้รับ และจัดเก็บจดหมายเพื่อนำมาใช้ในภายหลัง ส่วน MTA ทำหน้าที่หาเส้นทางและส่งจดหมายไปถึงปลายทาง การติดต่อระหว่าง MTA ใช้พอร์ตเบอร์ 25 โดยแลกเปลี่ยนข้อมูลตามรูปแบบที่กำหนดใน RFC 822



รูปที่ 2-11 แสดงสถาปัตยกรรมทั่วไปของระบบเมล์ใน TCP/IP

การจัดแบ่งออกเป็น UA และ MTA มีข้อดีคือ แยกงานของทั้งสองส่วนให้เป็นอิสระจากกัน หน้าที่ของ UA เน้นการทำงานกับผู้ใช้เพื่อให้ผู้ใช้สามารถอ่านเขียนจดหมายได้อย่างสะดวกโดยไม่ต้องยุ่งเกี่ยวกับการทำงานระดับล่างของโพรโตคอล ส่วน MTA ทำงานตาม SMTP เช่น การตรวจสอบความถูกต้องของแอดเดรสของผู้รับและผู้ส่ง รวมทั้งการหาเส้นทางและนำส่งจดหมายไปยังปลายทาง

2.7.2 เมล์รีเลย์

เมื่อผู้ใช้ส่งจดหมาย หน้าที่ของ UA คือส่งจดหมายไปยัง MTA เพื่อให้ MTA นำส่งต่อไป MTA ต้นทางอาจติดต่อกับ MTA ปลายทางโดยตรง หรือใช้วิธีรีเลย์ (relay) โดยส่งต่อเป็นทอดคือจาก MTA ต้นทางไปยัง MTA ระหว่างทางซึ่งจะเก็บเมล์ไว้ และนำส่งต่อตามจังหวะเวลาที่เหมาะสมจนกระทั่งเมล์ไปถึงปลายทาง ระบบเมล์ที่ใช้ส่งต่อเป็นทอดๆนี้เรียกว่า ระบบเก็บและส่งต่อ (store-and-forward-system) เมล์รีเลย์ประจำโดเมนหนึ่งๆเรียกว่า ตัวแลกเปลี่ยนเมล์ (mail exchanger)

การใช้เมล์รีเลย์มีข้อดีหลายประการเช่น

- ผู้ใช้งาน PC ทั่วไปที่ไม่มี MTA มักไม่ได้เปิดเครื่องใช้งานอยู่ตลอดเวลา เมื่อมีเมล์เข้ามาจำเป็นต้องอาศัยเมล์รีเลย์เป็นตัวเก็บพักเมล์ไว้จนกว่าจะเปิดใช้ PC เพื่อขอถ่ายเมล์มาจากเมล์รีเลย์
- เครือข่ายในหลายองค์กรใช้เมล์รีเลย์ทำหน้าที่ติดต่อกับเครือข่ายภายนอก เมล์รีเลย์อาจเป็นจุดเดียวที่ อนุญาติให้รับส่งเมล์โดยตรงกับภายนอกได้ โดยมีระบบไฟร์วอล์ห้ามเครื่องอื่นภายในเครือข่ายรับส่งเมล์โดยตรง เพื่อที่องค์กรจะได้สร้างระบบเมล์ศูนย์กลางและจะได้ไม่ให้ชื่อเครื่องอื่นภายในเครือข่ายแพร่ออกไปภายนอก
- การติดตั้ง MTA อย่างเช่น sendmail ในยูนิกซ์มีความซับซ้อน ผู้ดูแลระบบบางแห่งจะไม่ติดตั้ง MTA กันทั่วไป แต่ให้ใช้บริการผ่านเมล์รีเลย์แทน

2.7.3 รูปแบบจดหมาย

โครงสร้างของจดหมายประกอบด้วยส่วนสำคัญ 2 ส่วน คือ ส่วนหัวจดหมาย และส่วนเนื้อความจดหมาย ส่วนหัวจดหมายประกอบด้วยข้อความแสดงข่าวสารเกี่ยวกับการรับส่งจดหมาย เช่น ใครเป็นผู้ส่ง (From:), เมื่อเวลาใด (Date:), ส่งถึงใคร (To:), ใครบ้างที่ได้รับสำเนาจดหมาย (Cc:), และหัวเรื่องจดหมาย (Subject:) ส่วนเนื้อความจดหมายคือข้อความที่ได้รับซึ่งจะอยู่ต่อจากหัวจดหมาย SMTP กำหนดรูปแบบของหัวจดหมายไว้เพื่อให้ใช้เป็นหลักมาตรฐาน หัวจดหมายแต่ละฉบับอาจมีรายละเอียดมากน้อยต่างกันไป

2.7.4 ป๊อปและไอแม็ปโพรโตคอล (POP and IMAP Protocol)

โดยปกติแล้วเจ้าของจดหมายสามารถอ่านจดหมายโดยล็อกอินเข้าที่เซิร์ฟเวอร์ที่เก็บจดหมายอยู่แล้วใช้โปรแกรมที่ใช้อ่านจดหมาย เช่นโปรแกรม mail เปิดอ่านจดหมายหรืออีกวิธีหนึ่ง คือการเข้าใช้จดหมายแบบรีโมต

การเข้าใช้จดหมายแบบรีโมต เป็นการติดต่อขออ่านจดหมายวิธีหนึ่งที่ผู้ขอบริการไม่จำเป็นต้องล็อกอินเข้าเซิร์ฟเวอร์ที่เก็บจดหมายแต่สามารถอ่านจดหมายได้จากเครื่องใดก็ได้ที่มีโปรแกรมสนับสนุนโพรโตคอลอ่านจดหมายแบบรีโมต

ข้อดีของระบบนี้คือ อำนวยความสะดวกกรณีเครื่องไคลเอ็นต์ไม่มีโปรแกรมเทลเน็ตเพื่อล็อกอินเข้าเซิร์ฟเวอร์ อีกทั้งระบบจะนำจดหมายมาเก็บที่ไคลเอ็นต์โดยไม่จำเป็นต้องเชื่อมกับเซิร์ฟเวอร์ตลอดเวลาที่อ่านจดหมาย รูปแบบนี้มีประโยชน์อย่างมากสำหรับผู้ใช้อินเตอร์เน็ตแบบจ่ายค่าบริการตามระยะเวลาการใช้งาน

โพรโตคอลที่ใช้ในการเข้าถึงจดหมายจากระยะไกลมี 2 โพรโตคอลซึ่งนิยมใช้อย่างแพร่หลายได้แก่ ป๊อป (POP : Post Office Protocol) และไอแม็ป (IMAP : Internet Message Access Protocol) ทั้งป๊อปและไอแม็ปต่างก็เป็นบริการสำคัญที่ใช้แพร่หลายในอินเตอร์เน็ต ที่เซิร์ฟเวอร์ส่วนใหญ่มีให้บริการ

ป๊อปเป็นโพรโตคอลที่เก่ากว่า ส่วนไอแม็ปเป็นโพรโตคอลที่พัฒนาขึ้นมาภายหลังและเพิ่มเติมขีดความสามารถในการทำงานให้มากขึ้น ไอแม็ปรุ่นล่าสุดได้แก่ IMAP4 ส่วนป๊อปคือ POP3

- ป๊อป (POP)

ป๊อปได้รับการออกแบบมาให้ทำงานแบบออฟไลน์และทำงานโดยใช้ TCP พอร์ตเบอร์ 110 การดำเนินการใดๆกับจดหมายจะเกิดกับจดหมายที่อ่านมาเก็บไว้ที่ไคลเอ็นต์เท่านั้น โดยสรุปแล้วหน้าที่หลักของ ป๊อปคือ การส่งถ่ายจดหมายจากเซิร์ฟเวอร์มายังไคลเอ็นต์ นอกจากนี้ ป๊อปยังสามารถลบจดหมายที่เก็บไว้ที่เซิร์ฟเวอร์ได้อีกด้วย

- ไอแม็ป (IMAP)

ไอแม็ปมีความสามารถในการเข้าถึงแบบออนไลน์หรือเป็นแบบโต้ตอบกับเซิร์ฟเวอร์ ผู้ใช้สามารถดึงเฉพาะหัวข้อของจดหมาย ตัวจดหมาย หรือค้นหาจดหมายที่ตรงตามความต้องการและสามารถเปลี่ยนแปลงสถานะจดหมายที่เซิร์ฟเวอร์ได้ ซึ่งโดยสรุปก็คือ ไอแม็ปได้รับการออกแบบมาให้เข้าถึงจดหมายจากระยะไกลเหมือนกับการเข้าถึงจดหมายจากภายในเครื่องของผู้ใช้เอง ไอแม็ปทำงานโดยใช้ TCP พอร์ตเบอร์ 143

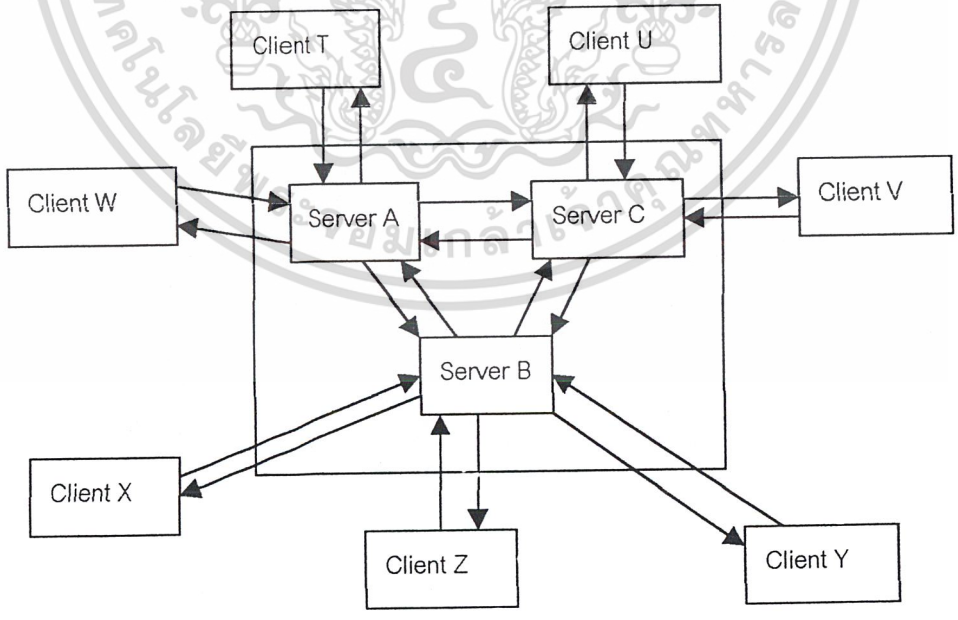
บทที่ 3
โครงสร้างของระบบการติดต่อสื่อสารกันระหว่างเอเจนต์

โครงงานนี้เป็นการจำลองการติดต่อสื่อสารต่างๆ ระหว่างบุคคล โดยมีการสร้างเอเจนต์ขึ้นมาเพื่อใช้ในการติดต่อสื่อสารระหว่างกัน ซึ่งในโครงงานนี้ได้ใช้ HTTP โพรโตคอลเป็นโพรโตคอลพื้นฐานในการติดต่อสื่อสาร ทำให้ต้องแบ่งประเภทของเอเจนต์ที่สร้างขึ้นออกเป็น 2 ประเภทคือ เอเจนต์ที่เป็นไคลเอ็นท์ กับเอเจนต์ที่เป็นเซิร์ฟเวอร์ เอเจนต์ที่เป็นไคลเอ็นท์จะใช้โปรแกรมโปรล็อกเขียนเป็นบราวเซอร์ ส่วนเอเจนต์ที่เป็นเซิร์ฟเวอร์จะใช้โปรแกรมโปรเว็บมาช่วยให้เอเจนต์ซึ่งเขียนด้วยภาษาโปรล็อกสามารถทำงานอยู่บนเว็บเซิร์ฟเวอร์ได้

3.1 โครงสร้างรวมของระบบการติดต่อสื่อสารกันระหว่างเอเจนต์

โครงสร้างของระบบการติดต่อสื่อสารกันระหว่างเอเจนต์ประกอบไปด้วยกลุ่มของเอเจนต์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์ซึ่งจะรอรับการติดต่อมาจากเอเจนต์อื่น และเอเจนต์ที่ทำหน้าที่เป็นไคลเอ็นท์ซึ่งต่อไปนี้จะเรียกว่า บราวเซอร์ การสื่อสารในระบบเริ่มจากบราวเซอร์เป็นผู้เริ่มการติดต่อกับเซิร์ฟเวอร์ โดยในขั้นตอนของการสื่อสารนั้นจะประกอบไปด้วยการแลกเปลี่ยนข้อมูลซึ่งกันและกันระหว่างบราวเซอร์กับเซิร์ฟเวอร์ และระหว่างเซิร์ฟเวอร์กับเซิร์ฟเวอร์

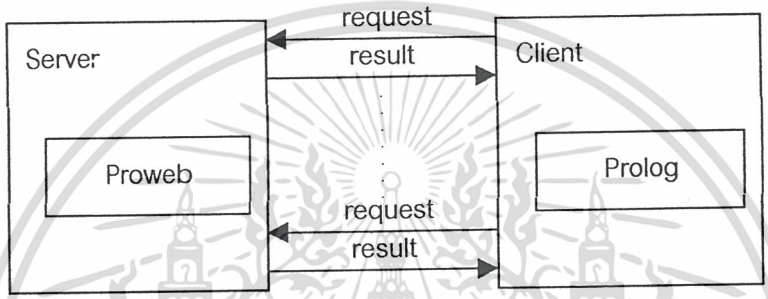
จากที่ได้กล่าวมานี้สามารถมองภาพรวมของระบบได้ดังรูปที่ 3-1



รูปที่ 3-1 ภาพรวมของระบบ

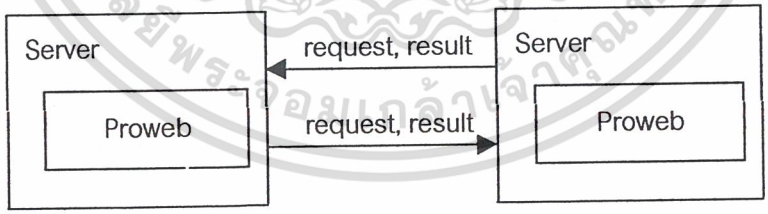
จากรูปที่ 3-1 ระบบการติดต่อสื่อสารกันระหว่างเอเจนต์ประกอบไปด้วยกลุ่มของเอเจนต์ที่เป็นเซิร์ฟเวอร์ 3 ตัว คือ Server A, Server B และ Server C โดยเอเจนต์ทั้ง 3 ตัวต้องรู้จักกัน และสามารถแลกเปลี่ยนข้อมูลซึ่งกันและกันได้ กับเอเจนต์ที่เป็นไคลเอ็นท์ซึ่งอาจมีได้หลายตัว เช่น Client T, Client U และ Client V โดยการแลกเปลี่ยนความรู้ระหว่างบราวเซอร์สามารถทำได้โดยผ่านการให้ข้อมูล และการถามความรู้จากเซิร์ฟเวอร์

ขั้นตอนการติดต่อสื่อสารระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องไคลเอ็นท์แต่ละเครื่องเริ่มจากไคลเอ็นท์จะเป็นผู้ขอบริการไปยังเซิร์ฟเวอร์ก่อน จากนั้นเซิร์ฟเวอร์จะรับคำร้องขอ (request) แล้วให้บริการในสิ่งที่ไคลเอ็นท์ขอมา ตามรูปข้างล่าง



รูปที่ 3-2 การติดต่อสื่อสารระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องไคลเอ็นท์แต่ละเครื่อง

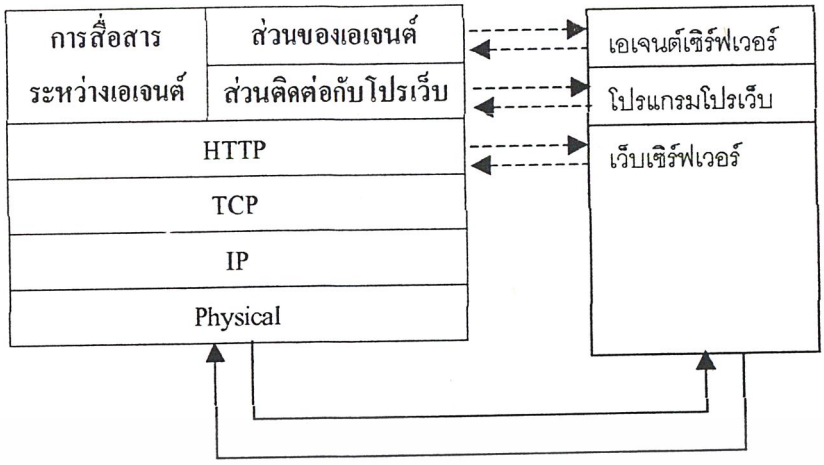
การติดต่อสื่อสารระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องเซิร์ฟเวอร์จะเหมือนกับการติดต่อสื่อสารระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องไคลเอ็นท์ โดยเครื่องที่ต้องการเริ่มการติดต่อจะส่งคำร้องไปให้กับเครื่องเซิร์ฟเวอร์เช่นเดียวกับเครื่องไคลเอ็นท์



รูปที่ 3-3 การติดต่อสื่อสารระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องเซิร์ฟเวอร์

โดยการเชื่อมต่อระหว่างเอเจนต์ใดๆ กับเอเจนต์ที่เป็นเซิร์ฟเวอร์จะมีลักษณะการเชื่อมต่อแบ่งออกเป็นการเชื่อมต่อในระดับ HTTP การเชื่อมต่อกับโปรเว็บ และการสนทนาระหว่างเอเจนต์ ดังรูปที่ 3-4

การเชื่อมต่อในระดับ HTTP ในฝั่งเซิร์ฟเวอร์ใช้โปรแกรมเว็บเซิร์ฟเวอร์เป็นผู้จัดการการรับส่งข้อมูล หลังจากนั้นจะส่งผ่านข้อมูลต่างๆ ไปให้กับโปรแกรมโปรเว็บที่มีหน้าที่แยกการสนทนาแต่ละครั้งออกจากกัน และในส่วนบนสุดก็จะเป็นการสนทนากันระหว่างเอเจนต์



รูปที่ 3-4 ภาพแสดงการเชื่อมต่อระหว่างเอเจนต์

3.2 โครงสร้างของโปรล็อกบราวเซอร์

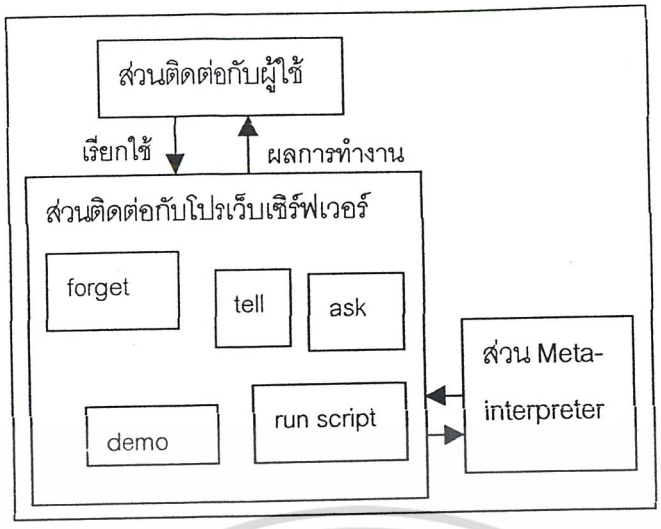
โปรล็อกบราวเซอร์ประกอบด้วยส่วนประกอบต่างๆ 2 ส่วนคือ ส่วนติดต่อกับผู้ใช้ และส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ หรือเว็บเซิร์ฟเวอร์อื่นๆ

3.2.1 ความสามารถของโปรล็อกบราวเซอร์

- สามารถเรียกดูเว็บเพจ (web page) ได้
- สามารถบอกความรู้ และถามข้อมูลจากโปรเว็บเซิร์ฟเวอร์
- สามารถสั่งให้โปรเว็บเซิร์ฟเวอร์ลืมความรู้ที่เคยบอกไปก่อนหน้านี้ได้
- สามารถสั่งให้โปรเว็บเซิร์ฟเวอร์ทำงานตามคำสั่งอื่นๆ
- สามารถสั่งให้บราวเซอร์ทำงานตามคำสั่งได้
- มีระบบล็อกอิน (login) เพื่อยืนยันตัวผู้ใช้ระบบ

3.2.2 ส่วนประกอบของโปรล็อกบราวเซอร์

ส่วนประกอบของโปรล็อกบราวเซอร์ สามารถแยกได้เป็นส่วนต่างๆ คือ ส่วนติดต่อกับผู้ใช้ ส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ และส่วน Meta-interpreter ตามรูป



รูปที่ 3-5 ส่วนประกอบของโปรล็อกบราวเซอร์

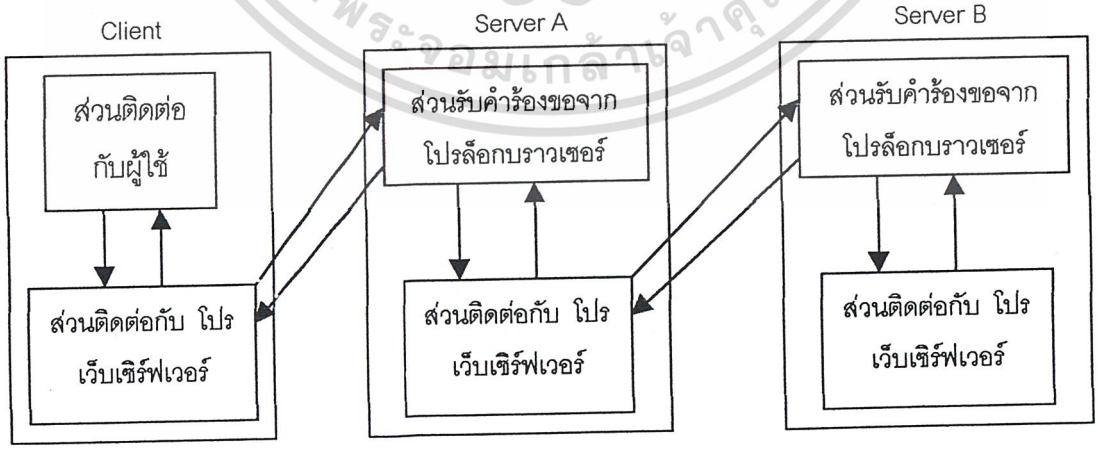
ส่วนติดต่อกับผู้ใช้มีหน้าที่รับข้อมูลจากผู้ใช้และเรียกใช้งานคำสั่งที่อยู่ในส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ หลังจากนั้นจึงแสดงผลการทำงาน

ส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์จะมีฟังก์ชันให้เรียกใช้งานอยู่ 5 ฟังก์ชันคือ demo, ask, tell, run script, forget ซึ่งรายละเอียดการเรียกใช้งานจะอยู่ในภาคผนวก ก

ส่วน Meta-interpreter เป็นส่วนที่ใช้ในการนำคำสั่งที่ได้รับมาทำงาน

3.3 โครงสร้างของเซิร์ฟเวอร์

เซิร์ฟเวอร์ประกอบด้วยส่วนประกอบต่างๆ 2 ส่วนคือ ส่วนรับคำร้องขอจากโปรล็อกบราวเซอร์ และส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ ซึ่งจะนำมาใช้ติดต่อกับโปรเว็บเซิร์ฟเวอร์ตัวอื่น



รูปที่ 3-6 ส่วนประกอบของเซิร์ฟเวอร์

การทำงานของเซิร์ฟเวอร์จะเริ่มจากการที่มีบราวเซอร์มาร้องขอบริการจากเซิร์ฟเวอร์ ส่วนรับคำร้องขอจะนำคำร้องขอนั้นมาประมวลผลให้ เสร็จแล้วก็จะส่งผลกลับไปให้บราวเซอร์

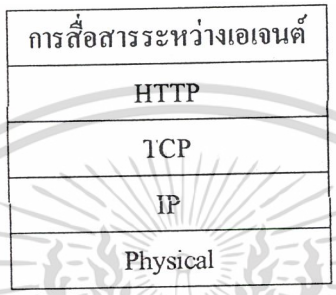
คำร้องขอที่บราวเซอร์ส่งเข้ามาให้โปรเว็บเซิร์ฟเวอร์มีอยู่ 3 ลักษณะคือ

- คำร้องขอในการถามข้อมูล
- คำร้องขอในการบอกความรู้
- คำร้องขอให้ทำงานตามคำสั่ง

ถ้าเป็นคำร้องขอในการถามข้อมูลและส่วนรับคำร้องขอของเซิร์ฟเวอร์นั้นไม่สามารถหาคำตอบให้บราวเซอร์ได้ ส่วนรับคำร้องขอก็จะส่งคำถามนั้นไปให้ส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ เพื่อให้ส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์นำคำถามนั้นไปหาคำตอบจากโปรเว็บเซิร์ฟเวอร์ตัวอื่นๆ เมื่อได้คำตอบแล้วก็จะส่งคำตอบนั้นกลับไปให้ส่วนรับคำร้องขอ เพื่อส่งกลับไปให้บราวเซอร์

บทที่ 4
โปรล็อกบราวเซอร์

โปรล็อกบราวเซอร์เป็นโปรแกรมที่สร้าง GUI (Graphics user interface) และสร้างการติดต่อกับเอเจนต์ที่เป็นเซิร์ฟเวอร์ ในบทนี้จะกล่าวถึงขั้นตอน แนวคิดในการสร้างโปรล็อกบราวเซอร์ ซึ่งการติดต่อจะใช้ HTTP โพรโตคอล เป็นช่องทางในการสื่อสาร โดยวิธีการสื่อสารระหว่างเอเจนต์เป็นดังรูป



รูปที่ 4-1 ลำดับชั้นในการติดต่อสื่อสาร

ขั้นตอนในการรับส่งข้อมูลของ HTTP

1. จะต้องโหลด TCP library ขึ้นมาด้วยคำสั่ง ensure_loaded(library('\tcplib\tcp')).
2. กำหนดค่าพารามิเตอร์ต่างๆที่ใช้ในการรับส่งข้อมูล ดังนี้

   tcp_set(recv_mode, (``, any, buffer)),

   tcp_set(recv_buffer, 65000).

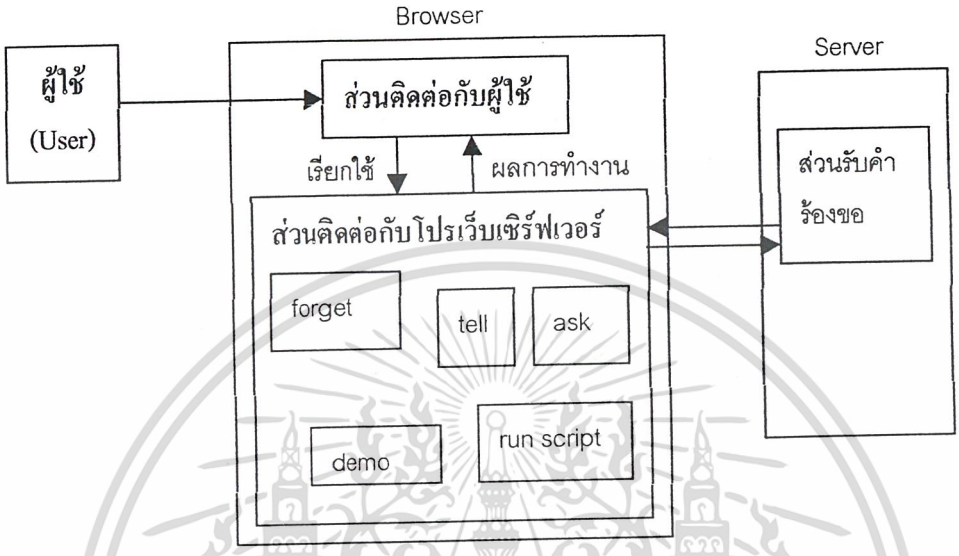
   ซึ่งจะเป็นการกำหนดให้ข้อมูลที่รับเข้ามาให้เก็บไว้ในบัฟเฟอร์ และกำหนดขนาดของบัฟเฟอร์เป็น 65000 ไบท์
3. สร้างการติดต่อกับเครื่องเซิฟเวอร์ tcp_connect(address(80, ` ชื่อเซิร์ฟเวอร์`), ID).
4. ส่งข้อมูลร้องขอไปเครื่องเซิฟเวอร์ tcp_send(ID, `GET URI HTTP/1.0~M~J~M~J`)
5. เตรียมบัฟเฟอร์ที่จะนำมาเก็บข้อมูลด้วยคำสั่ง fcreate(buffer, [], -1, 0) ซึ่งบัฟเฟอร์นี้จะนำมาใช้เก็บข้อมูลที่ถูกส่งมาทั้งหมด
6. อ่านข้อมูลและนำไปเก็บไว้ใน buffer

   tcp_select(term(ID, Term)),

   tcp_bufcopy(0,Outpos, Term, buffer, -1).
7. ทดสอบว่าข้อมูลถูกส่งมาครบแล้วหรือไม่ด้วยคำสั่ง tcp_select(end_of_file(ID)) ถ้ายังไม่ครบให้กลับไปทำในข้อ 6

4.1 ภาพรวมการทำงานของโปรล็อกบราวเซอร์

บราวเซอร์จะรับคำสั่งจากผู้ใช้ซึ่งเรียกใช้ฟังก์ชันต่างๆ แล้วส่งคำสั่งนั้นไปให้เซิร์ฟเวอร์ทำงาน จากนั้นจึงนำผลที่ได้มาแสดง โดยมีลักษณะการทำงานดังรูป



รูปที่ 4-2 ภาพรวมการทำงานของโปรล็อกบราวเซอร์

4.2 ขั้นตอนการสื่อสารระหว่างเอเจนต์

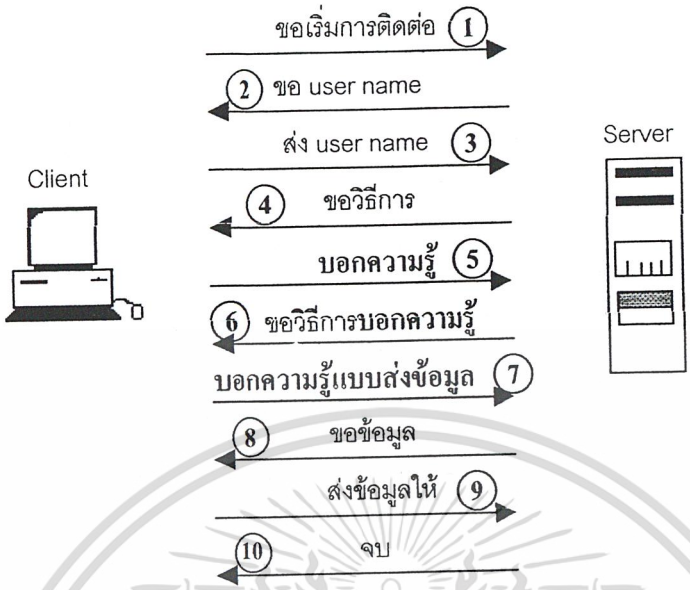
ส่วนติดต่อกับโปรเว็บเซิร์ฟเวอร์ของโปรล็อกบราวเซอร์มีฟังก์ชันให้เรียกใช้งานอยู่ 5 ฟังก์ชันคือ demo, ask, tell, run script, forget

ฟังก์ชันต่างๆ มีรายละเอียดการรับส่งข้อมูลดังนี้

- ฟังก์ชัน tell

  ใช้สำหรับการบอกความรู้ให้กับโปรเว็บเซิร์ฟเวอร์ ซึ่งจะมีรูปแบบการทำงานอยู่ 2 แบบ คือ การบอกความรู้แบบส่งข้อมูล และการบอกความรู้แบบส่งไฟล์

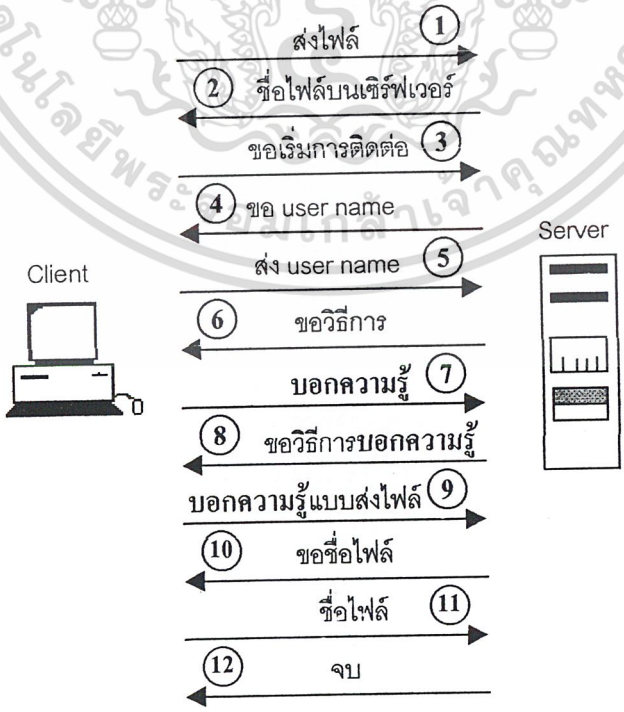
- การบอกความรู้แบบส่งข้อมูล



รูปที่ 4-3 แสดงขั้นตอนการบอกความรู้แบบส่งข้อมูล

การบอกความรู้แบบส่งข้อมูลเริ่มจากบราวเซอร์ขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอิน (user name) ไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการบอกความรู้ โดยบอกแบบส่งข้อมูล และส่งข้อมูลไปให้เซิร์ฟเวอร์

- การบอกความรู้แบบส่งไฟล์



รูปที่ 4-4 แสดงขั้นตอนการบอกความรู้แบบส่งไฟล์

การบอกความรู้แบบส่งไฟล์เริ่มจากการใช้โปรแกรม CGI ส่งข้อมูลในไฟล์ที่ต้องการส่ง แบบการขอบริการแบบให้ข้อมูล (Method POST) ไปที่เซิร์ฟเวอร์ก่อน แล้วทำการสร้างไฟล์ขึ้นมาใหม่ที่เครื่องเซิร์ฟเวอร์เพื่อเก็บข้อมูลที่ส่งมา โดยไฟล์ที่สร้างขึ้นใหม่นี้จะใช้การสุ่มชื่อไฟล์ จากนั้นจะส่งชื่อไฟล์ที่ได้จากการสุ่มกลับมาที่บราวเซอร์ และบราวเซอร์จะขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอินไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการบอกความรู้ โดยบอกแบบส่งไฟล์ และส่งชื่อไฟล์ที่เป็นชื่อที่ได้จากการสุ่มไปให้เซิร์ฟเวอร์

เหตุผลที่ต้องส่งไฟล์ไปไว้ที่เครื่องเซิร์ฟเวอร์ก่อน เพราะโปรเว็บเซิร์ฟเวอร์ไม่สามารถรับข้อมูลที่ส่งไปแบบการขอบริการแบบให้ข้อมูลได้ จึงต้องใช้โปรแกรม CGI อื่นช่วยรับข้อมูลไปเก็บไว้ในไฟล์ที่เครื่องเซิร์ฟเวอร์ก่อน แล้วใช้ชื่อไฟล์ที่อยู่ในเครื่องเซิร์ฟเวอร์นั้นเป็นเหมือนอินเด็กซ์ (index) โดยให้บราวเซอร์บอกชื่อไฟล์ที่อยู่บนเครื่องเซิร์ฟเวอร์แบบการขอบริการแบบขอข้อมูล (Method GET) แล้วให้โปรเว็บเซิร์ฟเวอร์ไปอ่านข้อมูลจากไฟล์นั้น

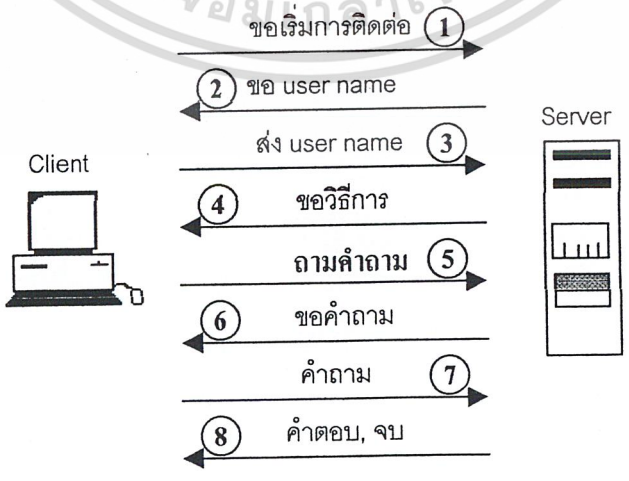
- ฟังก์ชัน ask

  ใช้สำหรับการถามคำถามที่ต้องการคำตอบ ซึ่งจะมีการทำงานได้ 3 กรณีคือ

  - เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปมีคำตอบให้
  - เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้แต่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง
  - เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้และไม่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง

  รายละเอียดการทำงานของแต่ละกรณีเป็นดังนี้

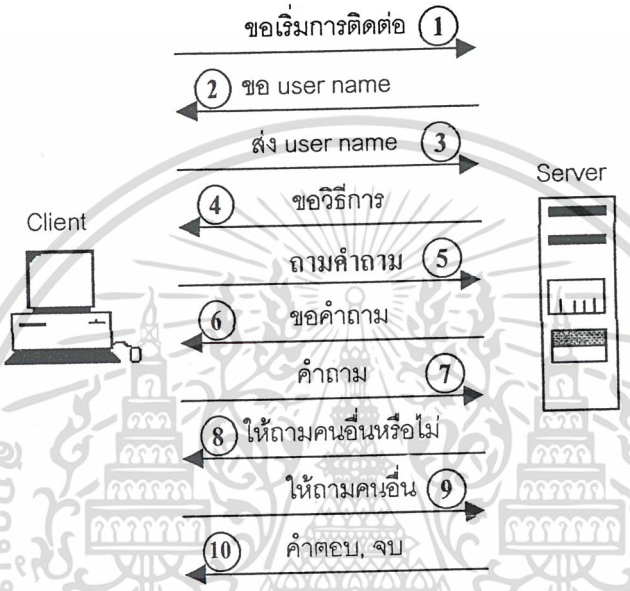
  ➢ เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปมีคำตอบให้



รูปที่ 4-5 แสดงขั้นตอนการถามที่เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปมีคำตอบให้

การทำงานเริ่มจากบราวเซอร์ขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอิน (user name) ไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการถามคำถาม แล้วส่งคำถามไปให้เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ส่งคำตอบมาให้ก็จบการทำงาน

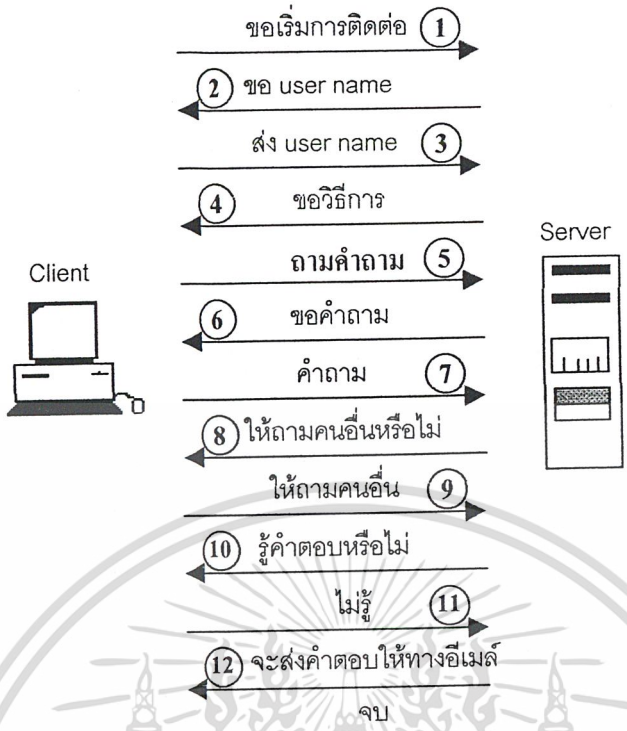
- เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้แต่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง



รูปที่ 4-6 แสดงขั้นตอนการถามที่เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้แต่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง

การทำงานเริ่มจากบราวเซอร์ขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอิน (user name) ไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการถามคำถาม แล้วส่งคำถามไปให้เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์หาคำตอบให้ไม่ได้ก็บอกให้เซิร์ฟเวอร์ไปหาคำตอบจากโปรเว็บเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง เมื่อเซิร์ฟเวอร์ส่งคำตอบมาให้ก็จบการทำงาน

- เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้และไม่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง



รูปที่ 4-7 แสดงขั้นตอนการถามที่เมื่อถามไปแล้วเซิร์ฟเวอร์ที่ถามไปไม่มีคำตอบให้และไม่สามารถหาคำตอบได้จากเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียง

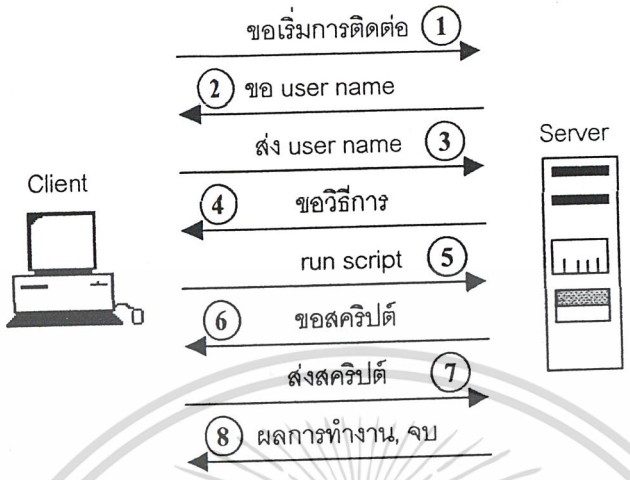
การทำงานเริ่มจากบราวเซอร์ขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอิน (user name) ไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการถามคำถาม แล้วส่งคำถามไปให้เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์หาคำตอบให้ไม่ได้ก็บอกให้เซิร์ฟเวอร์ไปหาคำตอบจากโปรเว็บเซิร์ฟเวอร์อื่นๆที่อยู่ใกล้เคียง ซึ่งโปรเว็บเซิร์ฟเวอร์อื่นๆ ที่อยู่ใกล้เคียงก็ไม่รู้คำตอบเช่นกัน เมื่อเซิร์ฟเวอร์ถามมาว่ารู้คำตอบหรือไม่บราวเซอร์จะตอบกลับไปว่าไม่รู้ ถ้าต้องการรู้คำตอบต้องไปตรวจสอบเมล์ (check mail) ที่ส่งมาจากโปรเว็บเซิร์ฟเวอร์ในภายหลัง

- ฟังก์ชัน demo

  ใช้เพื่อให้เว็บเซิร์ฟเวอร์แสดงความรู้ที่มีอยู่ออกมาจากคำถามที่ถามไป มีขั้นตอนการทำงานดังนี้ เริ่มจากตรวจสอบว่าเซิร์ฟเวอร์ที่เราต้องการให้แสดงความรู้นั้นใช่โปรเว็บเซิร์ฟเวอร์หรือไม่ โดยดูจากเลข IP ของเซิร์ฟเวอร์ว่าอยู่ในไฟล์ชื่อ conf.pl หรือไม่ ถ้าไม่อยู่จะถือว่าคำถามคือ URL ก็จะไปดึงหน้าเว็บเพจนั้นมาแสดง ถ้ามี IP นั้นอยู่ คือเซิร์ฟเวอร์นั้นเป็นโปรเว็บเซิร์ฟเวอร์จะส่งคำถามไปให้กับเซิร์ฟเวอร์ โดยวิธีการทำงานจะไม่มีการให้เซิร์ฟเวอร์ไปถามโปรเว็บเซิร์ฟเวอร์ที่อยู่ใกล้เคียงอีกถ้าไม่รู้คำตอบ คือหากเซิร์ฟเวอร์ไม่สามารถหาคำตอบได้จะถือว่าไม่มีคำตอบและจบการทำงานไป

- ฟังก์ชัน run script

  ใช้ส่งคำสั่งให้ไปทำงานบนโปรเว็บเซิร์ฟเวอร์



รูปที่ 4-8 แสดงขั้นตอนการส่งคำสั่งให้ไปทำงานบนโปรเว็บเซิร์ฟเวอร์

การทำงานเริ่มจากบราวเซอร์ขอเริ่มการติดต่อกับเซิร์ฟเวอร์ จากนั้นก็ส่งชื่อล็อกอินไปให้เซิร์ฟเวอร์และบอกไปว่าต้องการสั่งให้เซิร์ฟเวอร์ทำงานตามคำสั่ง (run script) แล้วส่งคำสั่งหรือสคริปต์ไปให้เซิร์ฟเวอร์

- ฟังก์ชัน forget

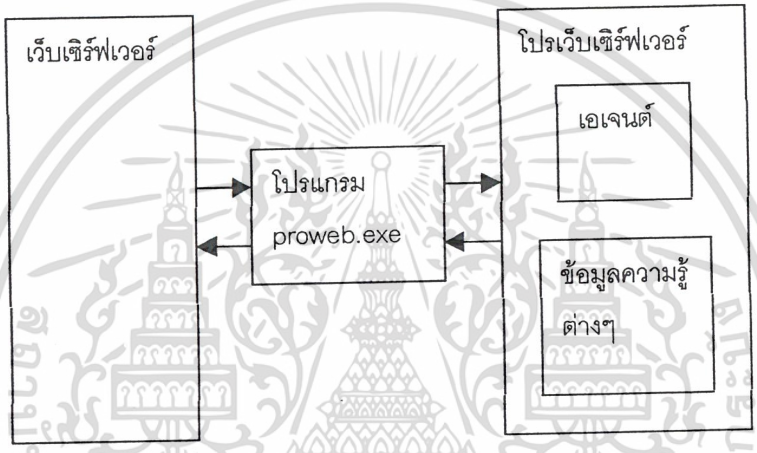
  ใช้สั่งให้โปรเว็บเซิร์ฟเวอร์ลืมความรู้ที่เคยบอกไป ซึ่งจะไปเรียกใช้คำสั่ง run script โดยสั่งให้โปรเว็บเซิร์ฟเวอร์ทำคำสั่งลบข้อมูลที่ระบุไว้

4.3 การติดตั้ง (setup) โปรล็อกบราวเซอร์

การลงโปรแกรม โปรล็อกบราวเซอร์จะต้องลงโปรแกรม Win-Prolog ก่อน จากนั้นให้เปิดไฟล์ที่ชื่อ browser.pl ขึ้นมาแล้วกดที่ปุ่ม run->complie จากนั้นจะมีหน้าจอล็อกอินขึ้นมาให้ หลักจากนั้นจะอยู่ในส่วนการใช้งานที่กล่าวถึงต่อไปในบทที่ 6

บทที่ 5
เซิร์ฟเวอร์

ในระบบการติดต่อสื่อสารระหว่างเอเจนต์หลายตัวเราได้ใช้โปรแกรมโปรเว็บเป็นส่วนติดต่อกับเอเจนต์ที่ทำงานบนเซิร์ฟเวอร์ และเนื่องจากโปรแกรมโปรเว็บเป็นเพียง CGI ตัวหนึ่ง การเรียกใช้งานจึงจำเป็นต้องมีเว็บเซิร์ฟเวอร์เป็นตัวจัดการการเชื่อมต่อในระดับ HTTP ส่วนโปรแกรมโปรเว็บจะมีหน้าที่ในการติดต่อกับเว็บเซิร์ฟเวอร์ และสร้างสภาพแวดล้อมในการทำงานของเอเจนต์ซึ่งถูกเขียนด้วยภาษาโปรล็อก โดยมีลักษณะความสัมพันธ์ ดังรูป



รูปที่ 5-1 ลักษณะการทำงานและความสัมพันธ์ระหว่างเว็บเซิร์ฟเวอร์ โปรเว็บ และเอเจนต์

5.1 การติดตั้งเซิร์ฟเวอร์

ในส่วนของเว็บเซิร์ฟเวอร์เราสามารถเลือกใช้เว็บเซิร์ฟเวอร์อะไรก็ได้ แต่โปรแกรมโปรเว็บนั้นเป็นโปรแกรมที่ทำงานบนระบบปฏิบัติการวินโดว์ (Microsoft Windows) ดังนั้นเราจึงเลือกใช้ Apache Web Server for Windows โดยดาวน์โหลดได้จาก http://www.apache.org/dist/httpd/binaries/win32/ รายละเอียดของการลง และตั้งค่าโปรแกรมจะกล่าวถึงในหัวข้อ 5.1.1

5.1.1 การตั้งค่าเว็บเซิร์ฟเวอร์สำหรับการใช้งานร่วมกับโปรเว็บเซิร์ฟเวอร์

การตั้งค่าสำหรับโปรแกรม Apache จะสามารถแก้ค่าของเซิร์ฟเวอร์โดยการเข้าไปแก้ในไฟล์ httpd.conf โดยจะต้องเพิ่มคำสั่งเหล่านี้ลงไป

```
ScriptAlias /proweb/ "C:/Apache/proweb/"

ScriptAlias /ProWeb/ "C:/Apache/proweb/"
```

```
<Directory "C:/Apache/proweb">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

เพื่อให้เซิร์ฟเวอร์รู้ว่าควรจะไปหาคำว่า proweb ได้ที่ใด และกำหนดให้อนุญาต เรียกโปรแกรมที่อยู่ในไดเรคทรอรี่ (Directory) ได้

5.1.2 การลงโปรแกรมโปรเว็บเซิร์ฟเวอร์

การลงโปรแกรมมีขั้นตอนต่างๆ ดังนี้

1. คัดลอก(copy) ไดเรคทรอรี่ Proweb ที่อยู่ในโปรแกรมโปรล็อกลงไปที่ไดเรคทรอรี่ของโปรแกรม Apache
2. คัดลอกไฟล์ pro386w.exe ลงไปในไดเรคทรอรี่ พร้อมทั้งเปลี่ยนชื่อไฟล์เป็น proweb.sys
3. คัดลอกไฟล์ pw_demo.pl, proweb_interface.pl, http_util.pl, util.pl, sendmail.pl conf.pl ลงไปที่ไดเรคทรอรี่ proweb\examples\ และคัดลอกไฟล์ user_profile.pl ลงไปที่ไดเรคทรอรี่ proweb\
4. คัดลอกไดเรคทรอรี่ library ที่อยู่ในโปรแกรมโปรล็อกลงไปที่ไดเรคทรอรี่ proweb\
5. เมื่อทำตามขั้นตอนนี้เสร็จแล้ว ให้เรียกไปที่ URL http://localhost/proweb/proweb.exe?lpa=examples เพื่อเป็นการเริ่มต้นโปรแกรม โปรเว็บเซิร์ฟเวอร์

5.2 โปรเว็บเซิร์ฟเวอร์

ไคลเอ็นท์ ในโปรเว็บจะหมายถึงเว็บบราวเซอร์ที่ใช้ในการติดต่อสื่อสาร และร้องขอข้อมูลจากโปรแกรมที่ทำงานอยู่บนเว็บเซิร์ฟเวอร์

โปรเว็บเป็นการเพิ่มความสามารถให้กับเว็บเซิร์ฟเวอร์ โดยทำให้เว็บเซิร์ฟเวอร์สามารถใช้งานโปรแกรมที่เขียนด้วยภาษาโปรล็อกได้ โปรเว็บอนุญาตให้หน้า HTML ที่แสดงอยู่ที่เว็บบราวเซอร์ของไคลเอ็นท์สามารถเชื่อมโยงกับโปรแกรมที่เขียนด้วยภาษาโปรล็อกที่ทำงานอยู่บนเซิร์ฟเวอร์ ถ้าไคลเอ็นท์ดูหน้า HTML ที่แสดงอยู่ที่เว็บบราวเซอร์ธรรมดามันจะไม่มีการกระทำใดๆ ระหว่างไคลเอ็นท์กับเซิร์ฟเวอร์โดยทั่วไปไคลเอ็นท์จะสามารถขอหน้า HTML หน้าถัดไปผ่านทาง URL และเซิร์ฟเวอร์จะเป็นผู้ทำหน้าที่ส่ง HTML หน้าถัดไปมาให้ถ้าใช้โปรเว็บ ไคลเอ็นท์จะสามารถสนทนากับโปรแกรมโปรเว็บที่ทำงานอยู่บนเซิร์ฟเวอร์ได้โดยตรงอย่างมีประสิทธิภาพ สามารถใช้หน้า HTML เหมือนเป็นทางผ่านของการกระทำระหว่างกันในการสนทนา

5.2.1 ลักษณะการทำงานของโปรเว็บเซิร์ฟเวอร์

เมื่อเริ่มต้นทำงาน PROWEB.EXE จะไปเรียกไฟล์ที่ชื่อ pw_demo.pl มาทำงานเท่านั้น ดังนั้นถ้าเราต้องการให้ไฟล์ไหนทำงาน เราจึงจำเป็นต้องเปลี่ยนชื่อไฟล์นั้นให้อยู่ในชื่อ pw_demo.pl ก่อน

การทำงานจะเริ่มจากการหาเพรดิเคท (predicate) ที่ชื่อว่า main_goal/0 และทำงานตามฟังก์ชันที่อยู่ในเพรดิเคทนั้นไปจนเสร็จก็จบการทำงาน โดยเพรดิเคท main_goal มักจะมีคำสั่ง proweb_send_form('ชื่อฟอร์ม') ไว้สำหรับสั่งให้โปรเว็บนำฟอร์ม (form) ตามที่ระบุชื่อมาทำให้อยู่ในรูปของหน้า HTML แล้วส่งไปที่บราวเซอร์ของไคลเอ็นท์

การที่โปรเว็บสามารถสนทนากับบราวเซอร์แต่ละตัวได้อย่างถูกต้อง คือไม่มีการสลับบทสนทนากันได้ เพราะโปรเว็บมีการสร้างตัวแปรที่ชื่อ proweb_data_uco และ proweb_data_page ขึ้นมาเองเพื่อใช้ในการบอกว่ากำลังคุยบทสนทนาที่เท่าไหร่ หน้าอะไร และยังมีตัวแปรที่ชื่อ Qxxxx ซึ่งสร้างมาจากเพรดิเคท proweb_question/2 เพื่อใช้เป็นตัวแปรที่จะรับข้อมูลจากฝั่งบราวเซอร์ โดยรายละเอียดของแต่ละตัวแปรเป็นดังนี้

- proweb_data_uco เป็นตัวแปรที่ใช้ระบุการสนทนาในครั้งนั้น โดยโปรเว็บจะสร้างตัวเลขขึ้นมาตัวเลขหนึ่งซึ่งจะไม่ซ้ำกับที่เคยมีอยู่ เช่น proweb_data_uco=%5B12%5D หมายความว่าบทสนทนานี้เป็นบทสนทนาที่ [12] โดย %5B และ %5D หมายถึง "[" และ "]" ตามลำดับ
- proweb_data_page เป็นตัวแปรที่ใช้ระบุหน้าของการสนทนาในครั้งนั้น เช่น proweb_data_uco=%5B12%5D&proweb_data_page=%5B1%5D หมายความว่า บทสนทนานี้เป็นบทสนทนาที่ [12] และเป็นการสนทนาในหน้าที่ [1] เมื่อสนทนาต่อไปค่า proweb_data_page จะเพิ่มขึ้นทีละหนึ่ง
- Qxxxx เป็นตัวแปรที่ใช้รับข้อมูลจากฝั่งบราวเซอร์ โดยชื่อตัวแปรจะเริ่มจาก Q0001 ไปเป็นลำดับตามจำนวนของตัวแปรที่เราต้องการใช้ในหน้านั้น เช่น ถ้าหน้านั้นมีฟิลด์ (field) ที่ใช้รับข้อมูลอยู่ 5 ตัว ก็จะมีตัวแปร Q0001 ถึง Q0005

5.3 รายละเอียดการทำงานของเซิร์ฟเวอร์

เอเจนต์เซิร์ฟเวอร์มีความสามารถที่จะรับการร้องขอให้มันทำงาน ซึ่งจะแบ่งการร้องขอออกเป็น 3 แบบ เป็น การร้องขอเพื่อขอคำตอบ(ask), การร้องขอให้เอเจนต์เซิร์ฟเวอร์เก็บความรู้เอาไว้(tell), การร้องขอให้เอเจนต์เซิร์ฟเวอร์ทำงานตามสคริปต์(run script) ซึ่งจะมีรายละเอียดดังนี้

การทำงานทั้งหมดจะเริ่มจากการที่มีบราวเซอร์มาร้องขอการติดต่อ ซึ่งเอเจนต์เซิร์ฟเวอร์จะส่งคำถามไปว่า คนถามนั้นคือใคร หลังจากที่รู้ว่าบราวเซอร์เป็นใครแล้วก็จะถามว่าต้องการให้เอเจนต์เซิร์ฟเวอร์ทำอะไร ซึ่งสามารถเลือกได้ 3 แบบ คือ บอกความรู้ให้เอเจนต์เซิร์ฟเวอร์ (tell), ถามคำถาม (ask) และสั่งให้เอเจนต์เซิร์ฟเวอร์ทำงานตามคำสั่ง (run script)

ถ้าบราวเซอร์ต้องการบอกความรู้ให้เอเจนต์เซิร์ฟเวอร์ เอเจนต์เซิร์ฟเวอร์ก็จะถามต่อว่าจะบอกความรู้แบบไหน แบบส่งข้อมูล หรือส่งไฟล์ ถ้าบราวเซอร์ตอบมาว่าแบบส่งข้อมูล เอเจนต์เซิร์ฟเวอร์ก็จะ

รับข้อมูลนั้นไปเก็บไว้ แต่ถ้าบราวเซอร์ตอบว่าแบบส่งไฟล์ เอเจนต์เซิร์ฟเวอร์ก็จะรับไฟล์มา และอ่านข้อมูลจากไฟล์นั้นมาเก็บไว้เป็นความรู้

ถ้าบราวเซอร์ต้องการถามคำถามเอเจนต์เซิร์ฟเวอร์ เอเจนต์เซิร์ฟเวอร์ก็จะถามต่อว่าจะถามอะไร เมื่อเอเจนต์เซิร์ฟเวอร์ได้รับคำถามมาแล้วจะนำคำถามนั้นไปหาคำตอบ ถ้าเอเจนต์เซิร์ฟเวอร์หาคำตอบให้ได้ ก็จะส่งคำตอบนั้นกลับไปให้บราวเซอร์ แต่ถ้าหาคำตอบไม่ได้ เอเจนต์เซิร์ฟเวอร์จะถามกลับไปที่บราวเซอร์ว่าผู้ใช้นั้นรู้คำตอบหรือไม่ ถ้ารู้ก็ให้บอก หรือต้องการให้ไปถามเอเจนต์เซิร์ฟเวอร์อื่น ถ้าบราวเซอร์ตอบกลับมาว่าให้ไปถามเอเจนต์เซิร์ฟเวอร์อื่น เอเจนต์เซิร์ฟเวอร์จะนำคำถามนั้นส่งไปถามเอเจนต์เซิร์ฟเวอร์ทุกเครื่องที่อยู่ใกล้เคียง เมื่อได้คำตอบแล้วก็ทำการส่งกลับไปให้บราวเซอร์ แต่ถ้าถามจากเอเจนต์เซิร์ฟเวอร์ทุกเครื่องที่อยู่ใกล้เคียงแล้วยังไม่ได้คำตอบอีก เอเจนต์เซิร์ฟเวอร์จะถามกลับไปที่บราวเซอร์ว่าผู้ใช้นั้นรู้คำตอบหรือไม่อีกครั้ง ถ้ารู้ก็ให้บอก ถ้าไม่รู้เอเจนต์เซิร์ฟเวอร์จะเก็บข้อมูลคำถามและชื่อล็อกอิน (user name) ของผู้ถามไว้ เมื่อเอเจนต์เซิร์ฟเวอร์รู้คำตอบแล้ว จะส่งคำตอบไปให้ที่อีเมล์ของผู้ถาม ซึ่งผู้ถามจะสามารถตรวจสอบเมล์ (check mail) ที่ส่งมาจากเอเจนต์เซิร์ฟเวอร์ได้ทางโปรล็อกบราวเซอร์

ถ้าบราวเซอร์ต้องการสั่งให้เอเจนต์เซิร์ฟเวอร์ทำงานตามคำสั่ง เอเจนต์เซิร์ฟเวอร์ก็จะรับคำสั่งมาทำงานให้ การทำงานจะสำเร็จก็ต่อเมื่อคำสั่งนั้นเป็นคำสั่งในรูปแบบสคริปต์ของภาษาโปรล็อก และเป็นเพรดิเคทที่เอเจนต์เซิร์ฟเวอร์รู้จักเท่านั้น

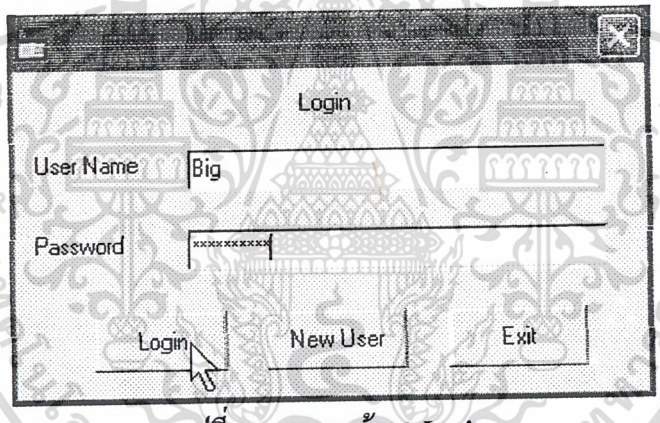
บทที่ 6
ขั้นตอนการใช้งานโปรล็อกบราวเซอร์

ในบทนี้จะกล่าวถึงขั้นตอนและวิธีการใช้งานระบบที่จะประกอบไปด้วย ส่วนของการยืนยันผู้ใช้ระบบ ส่วนการเรียกดูข้อมูลจากเซิร์ฟเวอร์ ส่วนการให้ข้อมูลกับเซิร์ฟเวอร์ ส่วนการถามคำถาม ส่วนการบอกให้เซิร์ฟเวอร์ลืมข้อมูล ส่วนการสั่งงานบราวเซอร์ และส่วนการสั่งงานเซิร์ฟเวอร์

6.1 การยืนยันผู้ใช้ระบบ

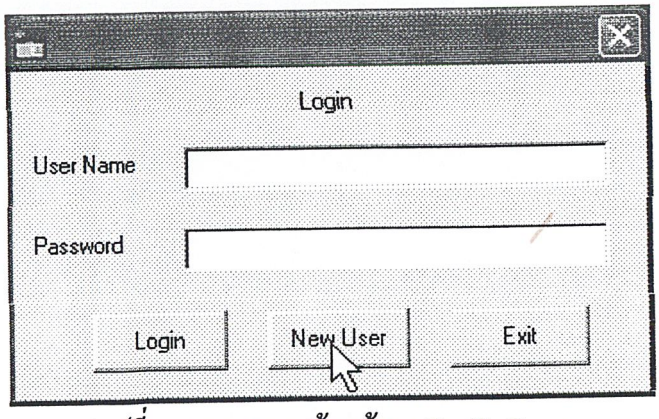
การยืนยันผู้ใช้ระบบ หรือการล็อกอิน มีจุดประสงค์เพื่อที่จะทราบได้ว่าผู้ที่ใช้ระบบอยู่ในขณะนั้นเป็นใคร โดยงานในส่วนนี้จึงจำเป็นต้องให้ผู้ใช้ทุกคนที่จะเริ่มใช้งานระบบต้องผ่านส่วนนี้ไปก่อน โดยมีรายละเอียดการใช้งานดังนี้

หน้าจอล็อกอินจะเป็นหน้าจอแรกในการที่จะเข้ามาใช้งานโปรแกรมโปรล็อกบราวเซอร์ ถ้าผู้ใช้คนนั้นเคยใช้ระบบนี้อยู่แล้ว ให้กรอก User Name, Password แล้วกด Login ดังรูป



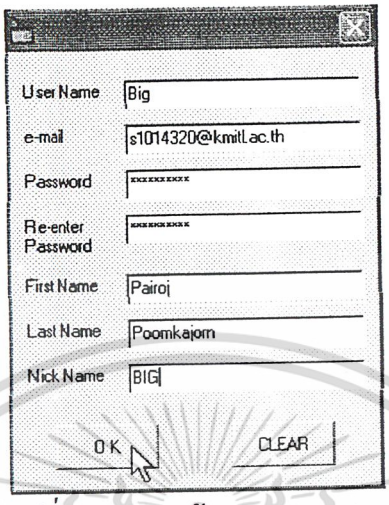
รูปที่ 6-1 แสดงหน้าจอ Login

ถ้าเป็นผู้ใช้ใหม่ต้องทำการสมัครสมาชิกก่อน โดยกดที่ปุ่ม New User



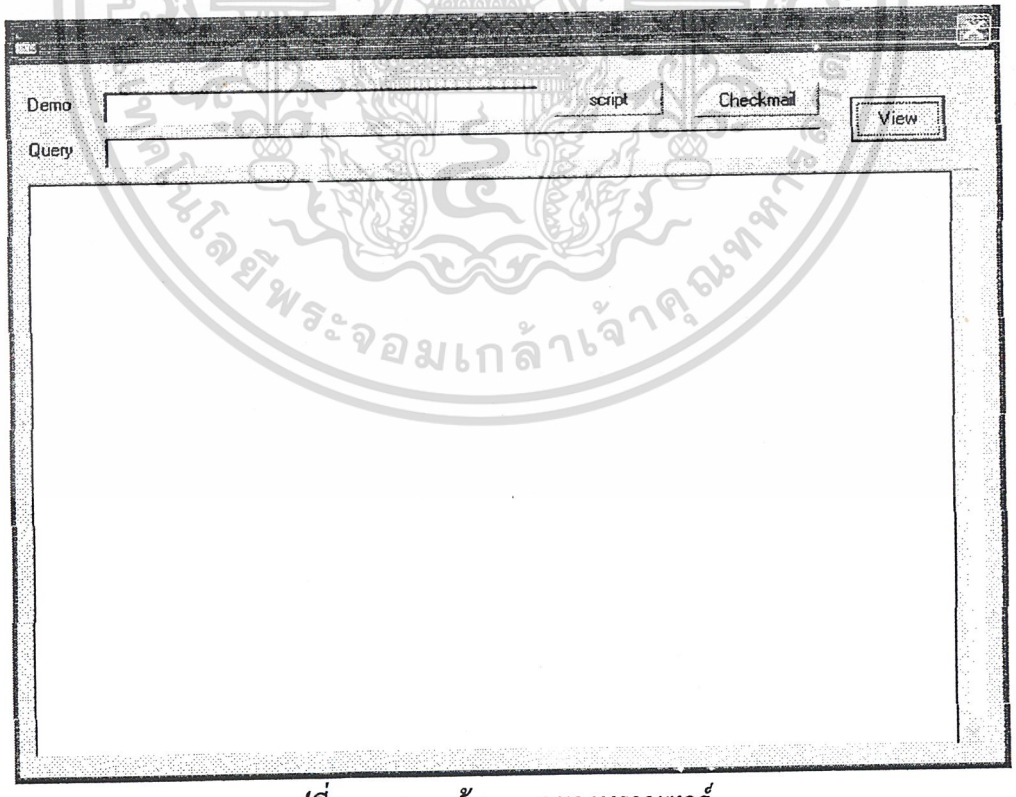
รูปที่ 6-2 แสดงการเข้าหน้าจอ Profile Form

จากนั้นจะมีหน้าจอ PROFILE FORM ขึ้นมาดังรูป หลักจากนั้นให้กรอกข้อมูลของตนลงไปซึ่งจำเป็นจะต้องใส่ในส่วนของ User Name, e-mail, และ Password ให้ครบดังรูป



รูปที่ 6-3 แสดงหน้าจอ Profile Form

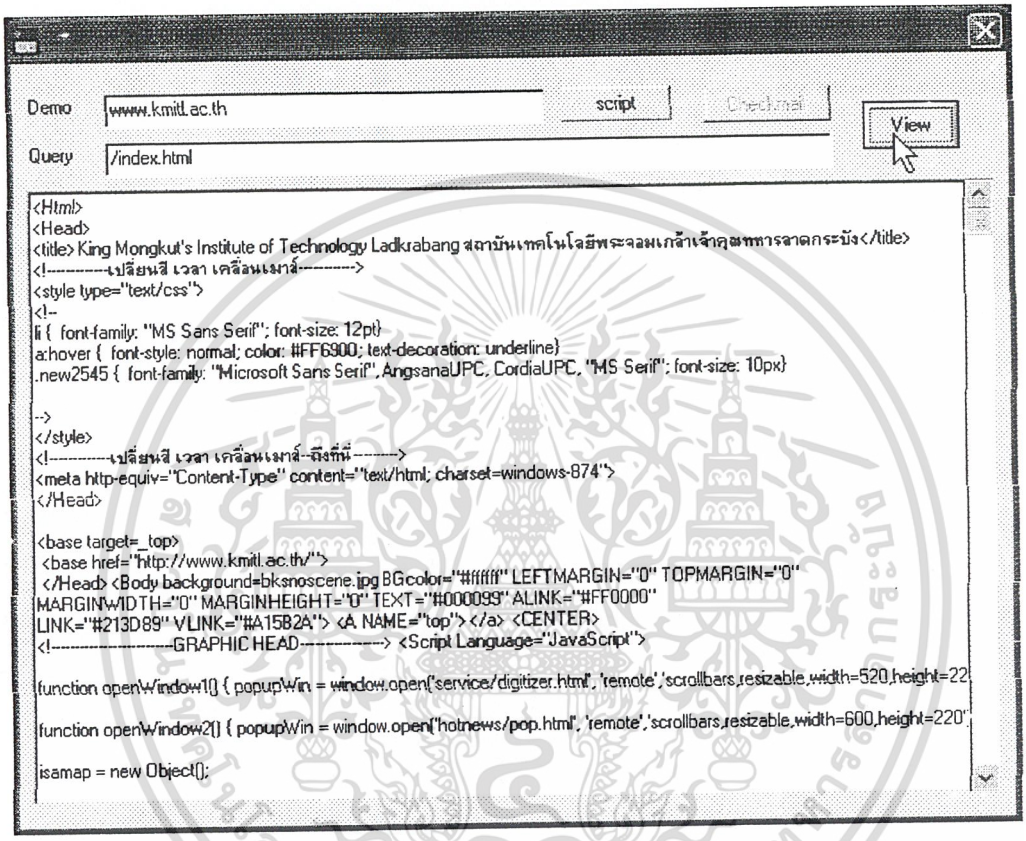
เมื่อกรอกข้อมูลเสร็จเป็นที่เรียบร้อยแล้วก็จะกลับมาในส่วนของการล๊อกอินเข้าระบบอีกครั้งหนึ่ง ให้กด Login เพื่อเริ่มต้นเข้าสู่ระบบ ซึ่งจะปรากฏหน้าจอดังรูป



รูปที่ 6-4 แสดงหน้าจอแรกของบราวเซอร์

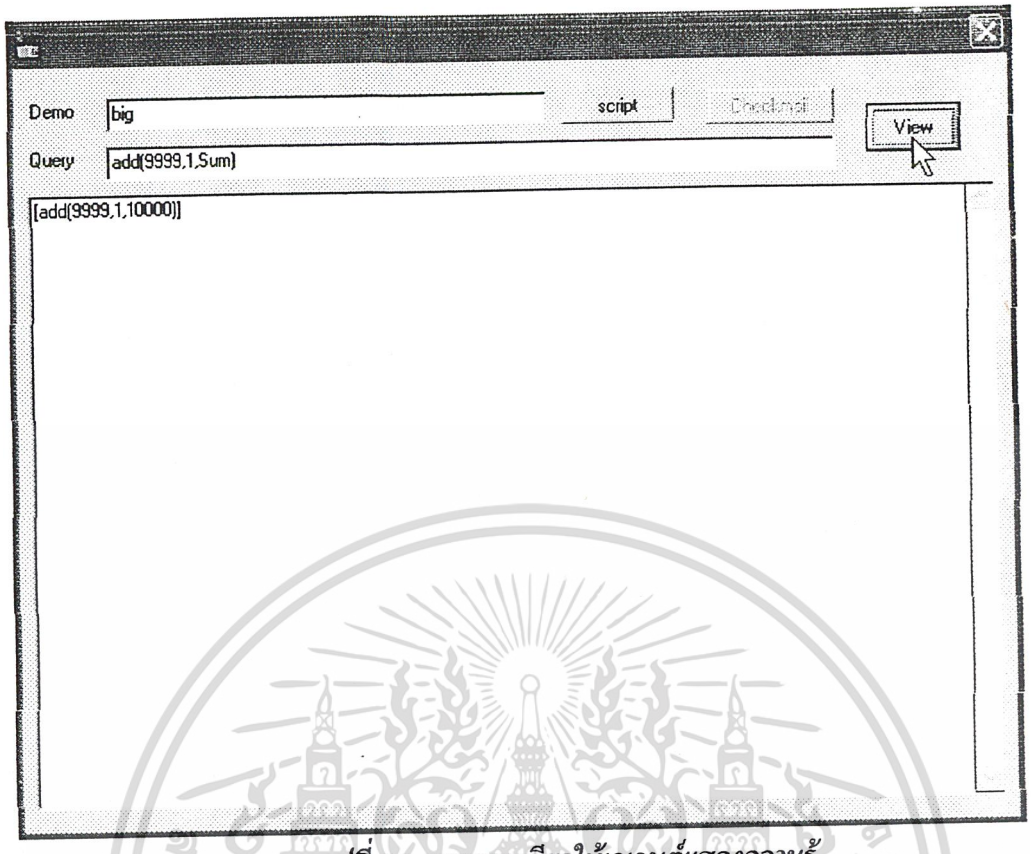
6.2 ส่วนการเรียกดูข้อมูลจากเซิร์ฟเวอร์

เมื่อผ่านขั้นตอนล็อกอินมาแล้ว จะเข้ามาในส่วนเริ่มต้นใช้งาน ซึ่งสามารถที่จะเรียกดูเว็บเพจได้ โดยพิมพ์ชื่อโดเมนเนม และ URL ของหน้าที่ต้องการ และกดที่ปุ่ม View ในตัวบราวเซอร์จะไปอ่านข้อมูลนั้นมาให้ และนำมาแสดงผลในรูปของตัวอักษร ดังรูป



รูปที่ 6-5 แสดงการเรียกดูเว็บเพจ

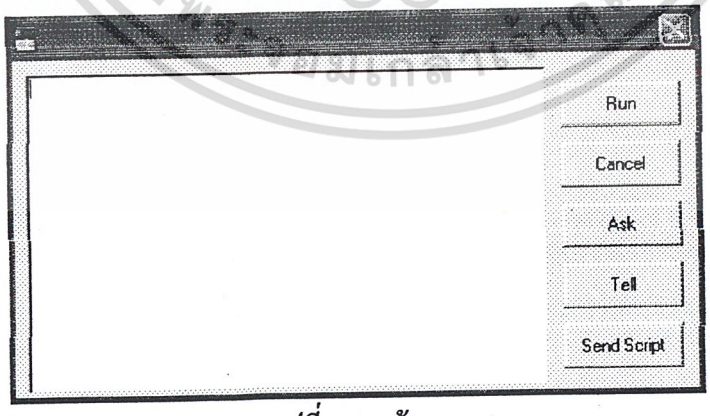
แต่หากชื่อเซิร์ฟเวอร์เป็นชื่อของ โปรเว็บเซิร์ฟเวอร์แล้วการทำงานจะต่างออกไป โดยจะดูที่ข้อมูลที่เป็น Query หากอยู่ในรูปแบบ Query ของภาษาโปรล็อกแล้วจะเป็นการส่งคำถามไปถามกับเซิร์ฟเวอร์ที่เป็นเอเจนต์ตัวนั้นให้นำความรู้ที่มีอยู่ของเอเจนต์ตัวนั้นมาแสดงดังรูป



รูปที่ 6-6 แสดงการเรียกให้เอเจนต์แสดงความรู้

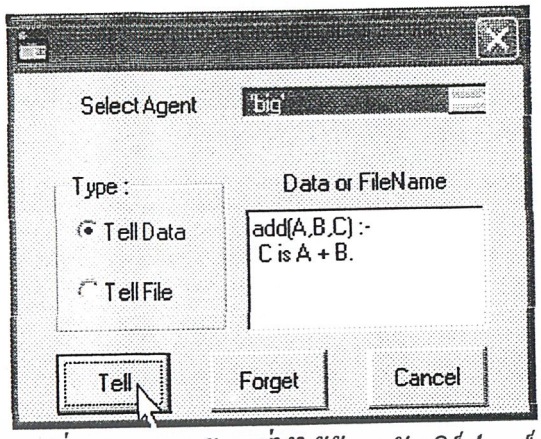
6.3 ส่วนการให้ข้อมูลกับเซิร์ฟเวอร์

การให้ข้อมูลกับเซิร์ฟเวอร์เป็นการที่เราต้องการให้เอเจนท์ที่เป็นเซิร์ฟเวอร์เก็บข้อมูล ความรู้ หรือวิธีการคิดเอาไว้ ซึ่งเซิร์ฟเวอร์จะต้องจดจำสิ่งที่ได้บอกไป ทำได้โดยกดไปที่ปุ่ม script จากนั้นจะมีหน้าจอขึ้นมาดังรูป



รูปที่ 6-7 หน้าจอ script

หน้าจอนี้เป็นหน้าที่ใช้เขียน และส่งคำสั่งต่างๆไปทำงาน เมื่อเราต้องการให้ข้อมูลกับเซิร์ฟเวอร์ให้กดไปที่ปุ่ม Tell จากนั้นจะมีหน้าจอขึ้นมาอีกหน้าจอหนึ่ง ดังรูป



รูปที่ 6-8 แสดงหน้าจอที่ใช้ให้ข้อมูลกับเซิร์ฟเวอร์

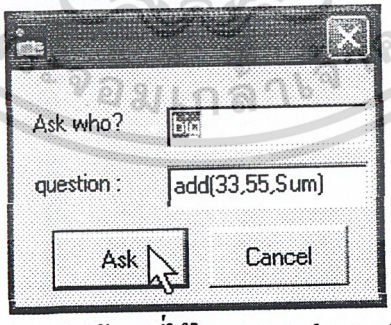
หน้านี้เป็นหน้าที่ใช้ให้ข้อมูลให้กับเอเจนต์ที่เป็นเซิร์ฟเวอร์ จากนั้นให้เลือกชื่อเอเจนต์ที่ต้องการบอกความรู้ไป และพิมพ์ข้อมูลที่ต้องการจะบอกลงไปในช่อง Data or FileName หรืออาจจะพิมพ์เป็นชื่อไฟล์ที่เก็บข้อมูลที่ต้องการจะบอกไปก็ได้ แล้วจึงค่อยมาเลือกในช่อง Type ว่าได้ใส่เป็นชื่อไฟล์ลงไป หรือใส่เป็นข้อมูลลงไป ข้อมูลที่พิมพ์ไปจะต้องอยู่ในรูปแบบสคริปต์ของภาษาโปรล็อกเช่น

add(A,B,C) :- C is A + B.

เมื่อใส่ข้อมูลครบแล้วให้กดที่ปุ่ม Tell ซึ่งระบบจะจัดการนำความรู้ที่ได้บอกไปนี้ไปเก็บไว้ในเอเจนต์ที่เลือกไปนั้น

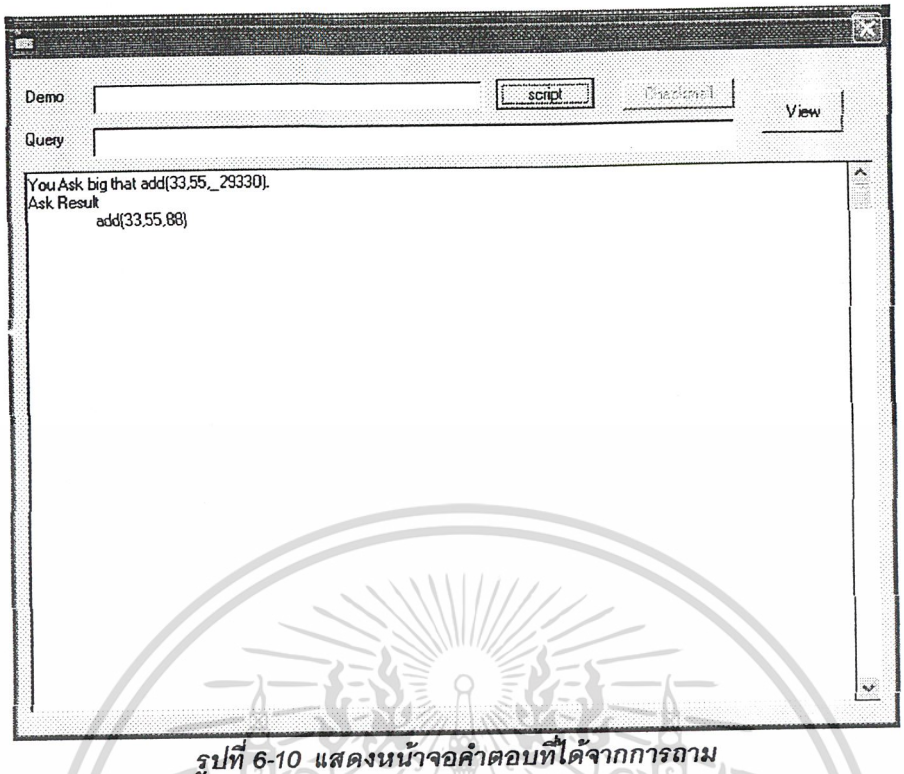
6.4 ส่วนการถามคำถาม

ในส่วนนี้เป็นการเรียกให้เอเจนต์ที่เป็นเซิร์ฟเวอร์ไปหาคำตอบมาให้ โดยเริ่มจากกดไปที่ปุ่ม Ask ในหน้าจอ script จากนั้นจะมีหน้าจอมาให้รอกข้อมูลที่จะถามดังรูป



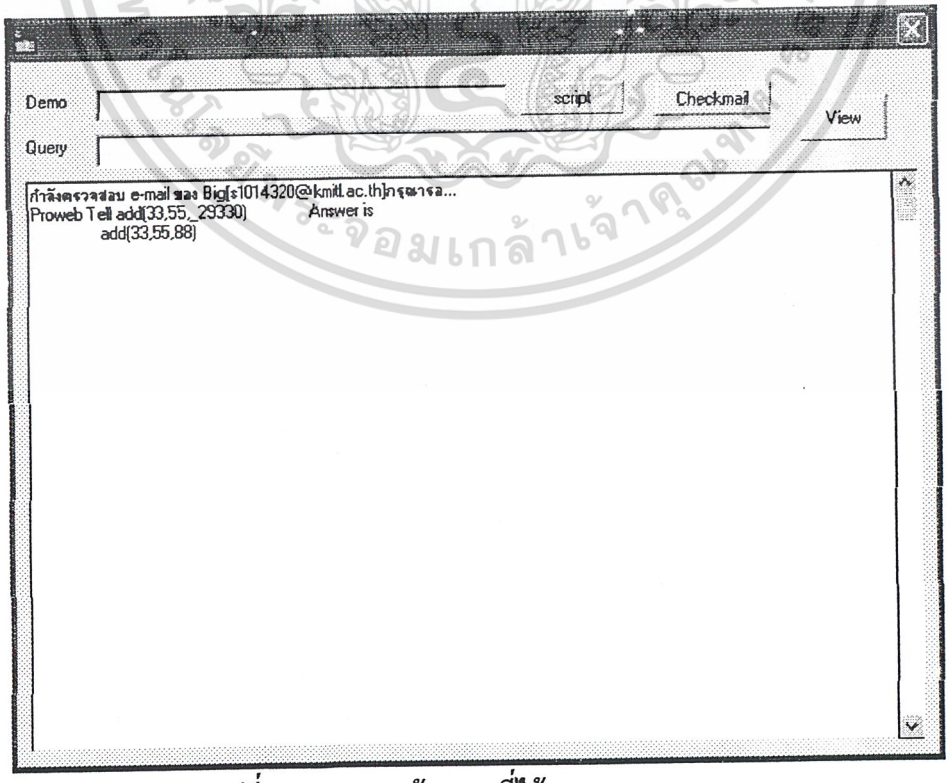
รูปที่ 6-9 แสดงหน้าจอที่ใช้ในการถามคำถามเซิร์ฟเวอร์

ให้ใส่ชื่อโปรเว็บเซิร์ฟเวอร์ที่ต้องการถาม และคำถามที่เราต้องการคำตอบ จากนั้นกดปุ่ม Ask จากนั้นก็จะรอเอเจนต์เซิร์ฟเวอร์ส่งคำตอบคืนกลับมา ถ้าหากในตัวเอเจนต์เซิร์ฟเวอร์นั้นไม่มีคำตอบ ก็จะไปหาคำตอบจากเซิร์ฟเวอร์ตัวอื่นมาให้ ดังแสดงตามรูป



รูปที่ 6-10 แสดงหน้าจอคำตอบที่ได้จากการถาม

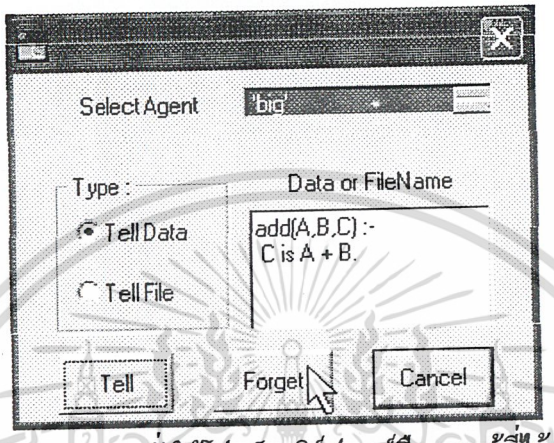
แต่ถ้าทุกเซิร์ฟเวอร์ยังไม่มีคำตอบของคำถามนี้ เอเจนต์เซิร์ฟเวอร์ตัวที่บราวเซอร์ถามไปจะเก็บคำถาม และชื่อผู้ถามเอาไว้ เมื่อใดที่เอเจนต์เซิร์ฟเวอร์ตัวนั้นมีคำตอบ ก็จะส่งคำตอบนั้นกลับมาทางอีเมล์ของผู้ใช้คนนั้น ซึ่งผู้ใช้จะเรียกดูคำตอบจากการถามนี้ได้โดยกดที่ปุ่ม Check mail และจะแสดงคำตอบตามรูป



รูปที่ 6-11 แสดงหน้าจอผลที่ได้จากการ check mail

6.5 ส่วนการบอกให้เซิร์ฟเวอร์ลืมข้อมูล

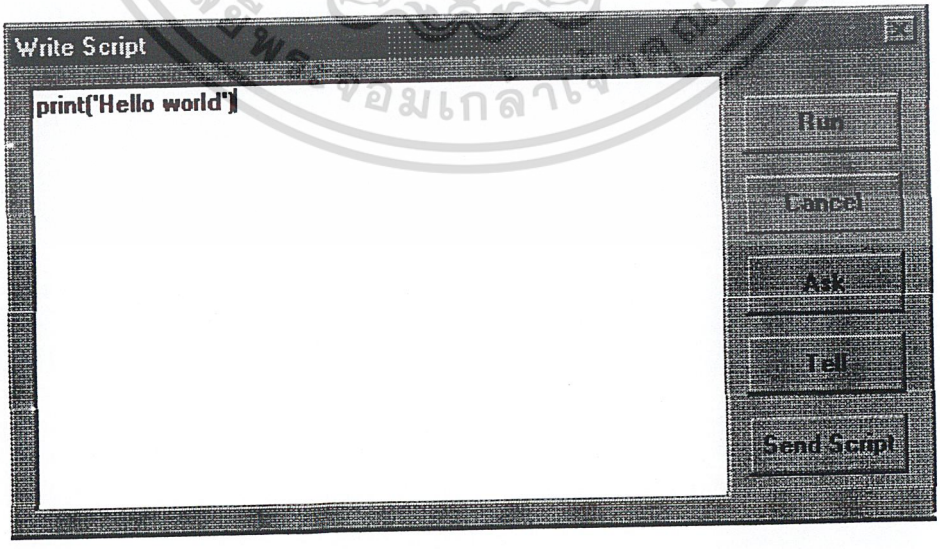
การสั่งให้เอเจนต์เซิร์ฟเวอร์ลืมข้อมูลที่ได้บอกไป ให้กดปุ่ม Tell จากหน้าจอ script หลังจากนั้นให้กรอกข้อมูลเช่นเดียวกับในขั้นตอนการให้ข้อมูล แล้วกดที่ปุ่ม Forget ดังรูป



รูปที่ 6-12 แสดงการสั่งให้โปรเว็บเซิร์ฟเวอร์ลืมความรู้ที่ได้บอกไป

6.6 ส่วนการสั่งงานบราวเซอร์

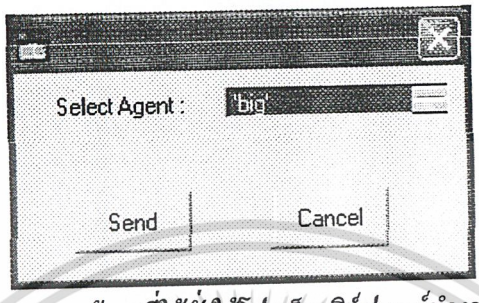
การสั่งให้บราวเซอร์ทำงานตามคำสั่งให้กดไปที่หน้าจอ script และพิมพ์สคริปต์ที่ต้องการเช่นดังรูป



รูปที่ 6-13 รูปการพิมพ์สคริปต์

6.7 ส่วนการสั่งงานเซิร์ฟเวอร์

การสั่งให้โปรเว็บเซิร์ฟเวอร์ทำงานตามคำสั่งให้ทำเช่นเดียวกับการสั่งให้บราวเซอร์ทำงาน แต่ให้เปลี่ยนจากการกดปุ่ม Run มาเป็นกดที่ปุ่ม Send Script จากนั้นให้เลือกโปรเว็บเซิร์ฟเวอร์ที่ต้องการ แล้วกดปุ่ม Send ดังรูป



รูปที่ 6-14 แสดงหน้าจอที่ใช้สั่งให้โปรเว็บเซิร์ฟเวอร์ทำงานตามคำสั่ง

บทที่ 7
บทวิจารณ์และสรุป

7.1 บทสรุป

โครงงานนี้ได้เสนอแนวทางในการพัฒนาการติดต่อสื่อสารกันระหว่างเอเจนต์หลายตัวโดยใช้ภาษาโปรล็อก ซึ่งสิ่งที่ได้ทำในโครงงานนี้คือ สร้างเอเจนต์ที่ทำงานเป็นบราวเซอร์ และสร้างเอเจนต์ที่ทำงานอยู่บนเซิร์ฟเวอร์ รวมถึงกระบวนการในการติดต่อระหว่างเอเจนต์ โดยเราได้ใช้ภาษาโปรล็อกมาสร้างเอเจนต์ที่ทำงานเป็นบราวเซอร์ และใช้โปรแกรมโปรเว็บมาช่วยในการสื่อสารของเอเจนต์ที่ทำงานอยู่บนเซิร์ฟเวอร์ ซึ่งเอจนต์ทั้งสองฝั่งสามารถสื่อสาร และเรียกใช้งานซึ่งกันและกันได้ โดยใช้ภาษากลางเป็น Query ในภาษาโปรล็อก และยังสามารถติดต่อระหว่างเอเจนต์อื่นๆได้ เช่นระหว่างเซิร์ฟเวอร์ กับเซิร์ฟเวอร์อื่น โดยในโครงงานนี้เราได้นำการติดต่อสื่อสารระหว่างมนุษย์มาจำลองเป็นการติดต่อสื่อสารระหว่างเอเจนต์ ซึ่งทำให้เอเจนต์มีความสามารถในการถามคำถาม ให้ข้อมูลระหว่างกัน และสามารถสั่งให้เอเจนต์ตัวอื่นทำงานให้ได้ แต่การถามคำถามนั้นยังทำได้แค่เพียงให้เอเจนต์ที่ทำงานเป็นบราวเซอร์สามารถถามคำถามเอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ได้ และเอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์สามารถถามคำถามกันเองได้ แต่เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ไม่สามารถถามคำถามมายังบราวเซอร์ได้ ซึ่งถ้าเป็นกรณีที่เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ไม่สามารถหาคำตอบให้ผู้ถามได้ เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์นั้นก็ควรที่จะสามารถถามคำถามมายังบราวเซอร์ได้เพื่อให้ผู้ใช้บราวเซอร์นั้นช่วยบอกความรู้ให้ การให้ข้อมูลนั้นเอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์สามารถบอกข้อมูลให้แก่ผู้ถามผ่านทางอีเมล์ของผู้ใช้ได้ในกรณีที่เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์นั้นไม่สามารถหาคำตอบให้แก่ผู้ถามได้ในขณะที่ถาม โดยที่หน้าจอโปรล็อกบราวเซอร์จะมีปุ่ม Check mail ไว้เพื่ออำนวยความสะดวกในการตรวจสอบอีเมล์ที่เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ส่งมาให้ ส่วนการสั่งให้เอเจนต์ตัวอื่นทำงานให้นั้น ยังทำได้แค่เพียงให้เอเจนต์ที่ทำงานเป็นบราวเซอร์สามารถสั่งให้เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ทำงานให้เท่านั้น แต่เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ไม่สามารถสั่งให้เอเจนต์ที่ทำงานเป็นบราวเซอร์ทำงานได้ การแสดงผลที่หน้าจอโปรล็อกบราวเซอร์เป็นการแสดงผลเป็นตัวหนังสือเท่านั้น ไม่สามารถแสดงผลเป็นรูปภาพได้

ประโยชน์ที่ได้รับจากการทำโครงงานนี้คือ ได้นำเอาเทคนิคทาง AI (Artificial Intelligence) มาทำให้การสื่อสารของเว็บมีความสามารถมากขึ้น

7.2 แนวทางในการพัฒนาต่อ

1. ปรับปรุงวิธีการสนทนาให้รัดกุมยิ่งขึ้น โดยเปลี่ยนเป็นการส่งเพรดิเคทที่มีความหมายไปแทนการส่งเป็นวิธีการ และข้อมูลไป เช่น การให้ข้อมูลเราอาจจะส่งไปแค่เพียง

   tell(ข้อมูลที่ต้องการบอก) เป็นต้น

2. พัฒนาให้เอเจนต์ที่ทำงานเป็นบราวเซอร์สามารถถามวิธีการทำงานได้ ไม่ใช่เป็นการถามเพื่อต้องการคำตอบอย่างเดียว ซึ่งจะทำให้เอเจนต์ที่ทำงานเป็นบราวเซอร์สามารถพัฒนาตัวเองขึ้นได้ ทำให้บราวเซอร์มีความรู้มากขึ้น ไม่ใช่รู้แค่เพียงเพรดิเคทที่เคยมีอยู่เท่านั้น
3. พัฒนาให้เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์สามารถถามคำถามมายังบราวเซอร์ได้เพื่อให้ผู้ใช้บราวเซอร์นั้นช่วยบอกความรู้ให้ ทำให้เอเจนต์ที่ทำงานเป็นเซิร์ฟเวอร์ไม่ต้องรอการบอกความรู้จากผู้ใช้ฝ่ายเดียว แต่ยังสามารถหาคำตอบในคำถามที่ตัวเองไม่รู้จากผู้ใช้ได้ด้วย

7.3 บทวิจารณ์

7.3.1 ข้อดีของโครงงาน

1. สร้างแนวคิดของการติดต่อสื่อสารระหว่างเอเจนต์
2. เพิ่มความสามารถของทั้งเว็บบราวเซอร์ และ เว็บเซิร์ฟเวอร์ให้มีความสามารถในการทำงานต่างๆตามความต้องการของผู้ใช้ระบบได้

7.3.2 ข้อจำกัดของโครงงาน

1. การใช้ภาษาโปรล็อกในการทำงานทำให้ประสิทธิภาพในด้านความเร็วต่ำ
2. การนำภาษาโปรล็อกมาใช้สร้างบราวเซอร์ทำให้ไม่สามารถแสดงผล HTML ออกมาเป็นรูปภาพได้เนื่องจากภาษาโปรล็อกไม่มีไลบรารีที่สนับสนุนด้านนี้

ภาคผนวก

ภาคผนวก ก
เพรดิเคทต่างๆ

ไฟล์ browser.pl

- คำอธิบาย ไฟล์นี้เป็นโปรแกรมที่ทำส่วนติดต่อผู้ใช้ ที่จะประกอบไปด้วยหน้าจอต่างๆ คือ หน้าล็อกอิน, หน้ารับชื่อผู้ใช้ใหม่, หน้าบราวเซอร์, หน้าสคริปต์, หน้าถามคำถาม, หน้าบอกความรู้, หน้าส่งสคริปต์ไปทำงาน
- ฟังก์ชันที่มี

user_dialog

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าต่างบราวเซอร์ มีหน้าที่สำหรับเป็นหน้าแสดงผลของโปรแกรมนี้

script_dialog

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าสคริปต์ มีหน้าที่สำหรับรับสคริปต์จากผู้ใช้ไปทำงาน

ask_dialog

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าถามคำถาม มีหน้าที่สำหรับรับข้อมูลที่จะนำไปถามเซิร์ฟเวอร์อื่น

tell_dialog

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าบอกความรู้ มีหน้าที่สำหรับรับข้อมูลที่จะนำไปบอกเซิร์ฟเวอร์อื่น

send_script

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าส่งสคริปต์ไปทำงาน ส่งสคริปต์ จากผู้ใช้ไปทำงานที่เซิร์ฟเวอร์อื่น

login

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้าล็อกอิน รับชื่อผู้ใช้และรหัสผ่าน ไปทำการล็อกอินเข้าระบบ

newuser

- คำอธิบาย เป็นฟังก์ชันที่ใช้สร้างหน้ารับชื่อผู้ใช้ใหม่ เพื่อรับข้อมูลของผู้ใช้คนใหม่

message_handler(Windows, Message, Data, Result)

- คำอธิบาย เป็นฟังก์ชันที่ใช้จัดการกับ windows message ทั้งหมด

system_login(Uname,Pass)

- คำอธิบาย เป็นฟังก์ชันส่งข้อมูลของผู้ใช้ไปทำการล็อกอินกับเครื่องเซิร์ฟเวอร์

new_user_profile(Uname,Email,Pass,Fname,Lname,Nname)

- คำอธิบาย เป็นฟังก์ชันที่ใช้ส่งข้อมูลของผู้ใช้คนใหม่ไปให้เซิร์ฟเวอร์

check_user_exist(Uname)

- คำอธิบาย เป็นฟังก์ชันที่ใช่ทดสอบว่าในระบบมีชื่อผู้ใช้คนนี้อยู่หรือไม่

email_check

- คำอธิบาย ตรวจสอบว่ามี e-mail ของผู้ใช้ระบบคนปัจจุบันจากเอเจนต์เซิร์ฟเวอร์ ส่งมาหรือไม่ ซึ่ง e-mail ที่ส่งมาจะเป็นข้อมูลที่ผู้ใช้เคยถามคำถามทิ้งไว้ที่เอเจนต์แต่ไม่สามารถตอบได้ในขณะนั้น จึงส่งมาเป็น e-mail ในภายหลัง และบราวเซอร์จะนำคำตอบที่ได้มาแสดงผล

check_have_new_proweb_mail(Email,Pass)

- คำอธิบาย ตรวจสอบว่ามี e-mail จากเอเจนต์เซิร์ฟเวอร์ส่งมาหรือไม่ แต่จะไม่นำมาแสดงผล

- **library ที่ต้องใช้**
  - **library('tcp\mail')** ใช้เพื่อการตรวจสอบ และอ่านเมล์
  - **proweb_conversation** ใช้เป็นฟังก์ชันที่ใช้ติดต่อกับเอเจนต์เซิร์ฟเวอร์
  - **http_util** ใช้รับส่งข้อมูลในระดับ HTTP โพรโตคอล
  - **conf** ใช้กำหนดค่าเริ่มต้นของระบบ

ไฟล์ proweb_interface.pl

- คำอธิบาย ไฟล์นี้เป็นโปรแกรมที่ทำส่วนติดต่อกับเอเจนต์เซิร์ฟเวอร์
- ฟังก์ชันที่มี

demo(Agent, Query, Ans)

- คำอธิบาย เป็นฟังก์ชันที่ให้เอเจนต์เซิร์ฟเวอร์แสดงความรู้จากที่ได้ถามไป
- ข้อมูลที่ใส่ **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  Query ชื่อไฟล์ หรือ เป้าหมาย (Goal)
- ผลที่ได้ **Ans** ข้อมูลในไฟล์ หรือลิสต์ ของคำตอบที่เป็นจริง

ask(Agent, Question, AnsList)

- คำอธิบาย เป็นฟังก์ชันที่ให้เอเจนต์เซิร์ฟเวอร์ไปหาคำตอบมา
- ข้อมูลที่ใส่ **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  Qusetion เป้าหมาย
- ผลที่ได้ **AnsList** ลิสต์ ของคำตอบ

tell(Agent, Flag, Knowlage)

- คำอธิบาย เป็นฟังก์ชันบอกความรู้ให้กับเอเจนต์เซิร์ฟเวอร์
- ข้อมูลที่ใส่
  - **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  - **Flag** ชนิดของความรู้ที่ส่งไป (file | data)
  - **Knowlage** ความรู้ที่ส่งไป อาจจะเป็นชื่อไฟล์ หรือเพรดิเคท

forget(Agent, Type, FileKnowlage)

- คำอธิบาย เป็นฟังก์ชันบอกให้เอเจนต์เซิร์ฟเวอร์ลบความรู้ที่มี
- ข้อมูลที่ใส่
  - **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  - **Flag** ชนิดของความรู้ที่ส่งไป (file | predicate)
  - **FileKnowlage** ความรู้ที่ต้องการให้ลบ อาจจะเป็นชื่อไฟล์ หรือเพรดิเคท

run_script(Agent, Data, ErrorCode)

- คำอธิบาย เป็นฟังก์ชันที่จัดการส่งสคริปต์ไปทำงานที่เอเจนต์เซิร์ฟเวอร์
- ข้อมูลที่ใส่
  - **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  - **Data** สคริปต์ที่ต้องการให้ไปทำงานบนเอเจนต์เซิร์ฟเวอร์
- ผลที่ได้
  - **ErrorCode** ผลการทำงาน
    - 0 สคริปต์สามารถทำงานได้ ผลการทำงานเป็น ture
    - -1 สคริปต์สามารถทำงานได้ ผลการทำงานเป็น fail
    - 1 สคริปต์ไม่สามารถทำงานได้

say_hello(Agent,Respond)

- คำอธิบาย เป็นฟังก์ชันที่ใช้เริ่มต้นการติดต่อกับเอเจนต์เซิร์ฟเวอร์
- ข้อมูลที่ใส่
  - **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
- ผลที่ได้
  - **Respond** ข้อมูลที่ตอบกลับมาจากเอเจนต์เซิร์ฟเวอร์อยู่ในรูปแบบของ HTML

speak1(Agent,Last_Msg,DataSend, New_Msg)

- คำอธิบาย เป็นฟังก์ชันที่ใช้สนทนาและรับส่งข้อมูลกับเอเจนต์เซิร์ฟเวอร์
- ข้อมูลที่ใส่
  - **Agent** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  - **Last_Msg** ข้อมูลตอบกลับจากการสนทนาก่อนหน้านี้ เพื่อเป็นการบอกว่าพูดต่อจากครั้งใดหากไม่ใส่ข้อมูลจะถือว่าเป็นการสนทนาต่อจากครั้งสุดท้ายที่สนทนา กับ Agent คนนี้
  - **DataSend** ข้อมูลที่ส่งไปให้เอเจนต์เซิร์ฟเวอร์
- ผลที่ได้
  - **New_Msg** ข้อมูลที่ตอบกลับมาจากเอเจนต์เซิร์ฟเวอร์อยู่ในรูปแบบของ HTML

get_page_id(UCO,Page)

- คำอธิบาย เป็นฟังก์ชันที่ใช้วิเคราะห์ข้อมูลเพื่อหาค่า ID ของการติดต่อกับเอเจนต์เซิร์ฟเวอร์ครั้งนั้น โดยจะอ่านข้อมูลจาก Standard Input Stream
- ผลที่ได้ **UCO** หมายเลขของการติดต่อในครั้งนั้น
  Page ลำดับของการสนทนา (เป็นเลขที่เพิ่มขึ้นเรื่อยๆ เริ่มจาก 0)

send_file(Who,MyPath,HostPath)

- คำอธิบาย เป็นฟังก์ชันที่ส่งไฟล์ไปให้กับฝั่งเซิร์ฟเวอร์
- ข้อมูลที่ใส่ **Who** ชื่อ หรือหมายเลข IP ของเซิร์ฟเวอร์
  MyPath ชื่อ และที่อยู่ของไฟล์ที่ต้องการส่ง
- ผลที่ได้ **New_Msg** ชื่อ และที่อยู่ของไฟล์ซึ่งถูกส่งไป บนเครื่องเซิร์ฟเวอร์

- **Dynamic Predicate ที่เกี่ยวข้อง**
  - **connection(fail)** ใช้บอกว่าการสนทนาครั้งนั้นสำเร็จหรือไม่
  - **my_name(Name)** เป็นเพรดิเคทที่ใช้บอกชื่อผู้ใช้ระบบขณะนี้
- **library ที่ต้องใช้**
  - **http_util** ใช้รับส่งข้อมูลในระดับ HTTP โพรโตคอล
  - **conf** ใช้กำหนดค่าเริ่มต้นของระบบ

ไฟล์ http_util.pl

- คำอธิบาย ไฟล์นี้เป็นฟังก์ชันที่ถูกเรียกใช้งานเพื่อให้สามารถรับส่งข้อมูลตาม HTTP โพรโตคอล ได้ซึ่งจะมีกระบวนการทำงานหลักๆเช่น GetURL PostMessage
- ฟังก์ชันที่มี

get_url(HostName, Path, Output)

- คำอธิบาย เป็นฟังก์ชันที่ทำหน้าที่สร้าง และ ส่ง HTTP GET Request ไปให้เครื่องเซิร์ฟเวอร์พร้อมทั้งรอรับ ข้อมูลตอบกลับด้วย
- ข้อมูลที่ใส่ **HostName** หมายเลข IP ของเซิร์ฟเวอร์ที่ต้องการติดต่อ
  Path ข้อมูล หรือ Resource ที่ต้องการ
- ผลที่ได้ **Output** ข้อมูลตอบกลับจากเซิร์ฟเวอร์(อยู่ในรูปของ HTTP Respond Message)

post_message(HostName, Path, InputFile, Output)

- คำอธิบาย เป็นฟังก์ชันที่ทำหน้าที่สร้าง และ ส่ง HTTP POST Request ไปให้เครื่องเซิร์ฟเวอร์พร้อมทั้งรอรับ ข้อมูลตอบกลับด้วย
- ข้อมูลที่ใส่ **HostName** หมายเลข IP ของเซิร์ฟเวอร์ที่ต้องการติดต่อ
  Path ตำแหน่งที่อยู่ของโปรแกรมที่ให้มารับ ข้อมูล
  InputFile ข้อมูลที่ต้องการส่งเข้าไป
- ผลที่ได้ **Output** ข้อมูลตอบกลับจากเซิร์ฟเวอร์(อยู่ในรูปของ HTTP Respond Message)

receive_url(ID)

- คำอธิบาย เป็นฟังก์ชันที่รอรับข้อมูลจาก TCP Socket มาเขียนบน Standard Output Stream
- ข้อมูลที่ใส่ **ID** หมายเลขของการติดต่อ

parse_header(ResponseMessage, String)

- คำอธิบาย ฟังก์ชันนี้มีหน้าที่ตรวจสอบ HTTP Respond Message ว่าที่ได้ส่งคำขอ (Request) ไปนั้นสามารถหาคำตอบได้หรือไม่ พร้อมทั้งแยก เนื้อข้อมูลออกมาจาก Respond Message ด้วย
- ข้อมูลที่ใส่ **ResponseMessage** ข้อมูลตอบกลับที่ได้มาจากการ get_url หรือ post_message
- ผลที่ได้ **String** เนื้อข้อมูลที่แยกออกมาได้

set_time_out(Ms)

- คำอธิบาย กำหนดเวลารอคอย

time_out

- คำอธิบาย ทดสอบว่าเวลารอคอยกำหนดหมดแล้วหรือยัง

have_time_out(Ms)

- คำอธิบาย กำหนดเวลารอคอยของการรอรับ HTTP Respond

no_time_out

- คำอธิบาย กำหนดเวลารอคอยของการรอรับ HTTP Respond ให้เป็นค่าสูงสุดที่เป็นไปได้

- **Dynamic Predicate** ที่เกี่ยวข้อง
  - **timeout_set(Ms)** อ่านค่าเวลารอคอยที่ถูกกำหนดไว้
- **library** ที่ต้องใช้
  - **library(tcp/tcp)** ใช้รับส่งข้อมูลในระดับ TCP/IP
  - **library(tcp/http)** ใช้สร้าง HTTP Request Message
  - **util** ใช้เรียกฟังก์ชันพื้นฐานในการจัดการสตริง

ไฟล์ util.pl

- คำอธิบาย ไฟล์นี้เก็บฟังก์ชันที่ถูกเรียกใช้งานทั่วๆไป
- ฟังก์ชันที่มี

string_cut(String,Start,Len,Out)

- คำอธิบาย ตัดสตริงจากตำแหน่งเริ่มต้นไปเป็นจำนวน เท่ากับ Len
- ข้อมูลที่ใส่ **String** สตริงเริ่มต้น
  Start จุดเริ่มต้น ไบต์แรกนับที่ 0
  Len ความยาวของสตริงที่ต้องการ ถ้า Len = -1 จะตัดจนจบสตริง
- ผลที่ได้ **Out** สตริงที่ถูกตัดออกมา

find_word(String,Word,Point)

- คำอธิบาย ค้นหาสตริงย่อยจากสตริงทั้งหมด แล้วคืนตำแหน่งของสตริงย่อยนั้น
- ข้อมูลที่ใส่ **String** สตริงทั้งหมด
  Word สตริงที่ต้องการค้นหา
- ผลที่ได้ **Point** จุดเริ่มต้นของสตริงที่พบ (เริ่มจาก 1)

convert_str_to_atom(Data,Atom)

- คำอธิบาย แปรข้อมูลให้อยู่ในรูปของเพรดิเคท
- ข้อมูลที่ใส่ **Data** ข้อมูลที่ต้องการเปลี่ยน
- ผลที่ได้ **Atom** ข้อมูลหลังจากเปลี่ยนไปแล้ว

to_write(Data)

- คำอธิบาย เขียนข้อมูลลง Standard Output
- ข้อมูลที่ใส่ **Data** ข้อมูลที่ต้องการเขียน

write_to_file(Data, File)

- คำอธิบาย เขียนข้อมูลลงไฟล์
- ข้อมูลที่ใส่ **Data** ข้อมูลที่ต้องการเขียน
  File ชื่อไฟล์

write_data_from_file(File)

- คำอธิบาย อ่านข้อมูลจากไฟล์ขึ้นมาและเขียนลงไปใน Standard Input
- ข้อมูลที่ใส่ **File** ชื่อไฟล์

solve(Goal)

- คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับทำเมตาอินเตอร์พรีทเตอร์ (Meta Interpreter)
- ข้อมูลที่ใส่ **Goal** เป้าหมาย หรือสิ่งที่ต้องการให้ทำงาน

ไฟล์ pw_demo.pl

- คำอธิบาย ไฟล์นี้เป็นโปรแกรมที่ทำงานเป็นเอเจนต์อยู่บนเซิร์ฟเวอร์
- ฟังก์ชันที่มี

main_goal

- คำอธิบาย ฟังก์ชันเริ่มต้นของการทำงานบนโปรเว็บ

add_knowlage(File, Realname)

- คำอธิบาย ฟังก์ชันที่จัดการเก็บข้อมูลความรู้จากไฟล์นี้ ลงในหน่วยความจำของโปรเว็บ
- ข้อมูลที่ใส่ **File** ชื่อที่อยู่ของไฟล์ที่อยู่บนเครื่องเซิร์ฟเวอร์
  Realname ชื่อของไฟล์ต้นฉบับ

assert_knowlage(Knowlage, Name)

- คำอธิบาย ฟังก์ชันที่จัดการเก็บข้อมูลลงในหน่วยความจำของโปรเว็บ
- ข้อมูลที่ใส่ **Knowlage** ความรู้ที่ต้องการเพิ่มเข้าไป
  Name ชื่อที่มาของความรู้นั้น

retract_ knowlage (Knowlage, Name)

- คำอธิบาย ลบความรู้ที่มีอยู่ออกจากหน่วยความจำ
- ข้อมูลที่ใส่ **Knowlage** ความรู้ที่ต้องการลบ
  Name ชื่อที่มาของความรู้นั้น

check_sendmail

- คำอธิบาย ทดสอบหาคำตอบว่ายังมีคำถามใดที่เซิร์ฟเวอร์เคยหาคำตอบไม่ได้ และสามารถหาได้แล้วในขณะนี้ จะส่งคำตอบกลับไปให้ผู้ถามทางอีเมล์

findAns(Goal, AnsList)

- คำอธิบาย หาคำตอบทุกคำตอบที่เป็นไปได้ของเป้าหมาย (Goal) จากความรู้ที่มีแล้วส่งคำตอบคืนไว้ที่ AnsList ซึ่งจะเก็บเป็นลิสต์ของคำตอบ

ask_other_server(Goal, AnsList)

- คำอธิบาย หาคำตอบทุกคำตอบที่เป็นไปได้ของเป้าหมาย (Goal) โดยการไปถามจากผู้อื่นแล้วส่งคำตอบคืนไว้ที่ AnsList ซึ่งจะเก็บเป็นลิสต์ของคำตอบ

add_user_profile(Uname, Email, Pass, Fname, Lname, Nname)

- คำอธิบาย เก็บข้อมูลของผู้ใช้แต่ละคนลงในไฟล์ user_profie.pl เพื่อใช้เป็นข้อมูลของผู้ใช้ระบบ

ไฟล์ conf.pl

- คำอธิบาย ไฟล์นี้เป็นไฟล์ที่ใช้ตั้งค่าเริ่มต้นของระบบ มีรายละเอียดดังนี้

- ฟังก์ชันที่มี

define(server).

define(browser).

- คำอธิบาย กำหนดว่าเอเจนต์ที่อ่านค่าจากไฟล์นี้ทำหน้าที่เป็นเซิร์ฟเวอร์ หรือบราวเซอร์

:- define(serber), assert(my_name(Name)).

- คำอธิบาย กำหนดชื่อของเครื่องเอเจนต์เซิร์ฟเวอร์เครื่องนั้น

proweb_server(big, '161.246.5.112').

proweb_server(pokky, '161.246.5.96').

- คำอธิบาย ชื่อและ IP บอกเอเจนต์เซิร์ฟเวอร์ทั้งหมดในระบบ

proweb_dir(Agent, '/proweb').

- คำอธิบาย บอกที่อยู่ของโปรแกรมโปรเว็บบนเครื่องที่เอเจนต์ตัวนั้นอยู่

receive_path(Agent, '/cgi-bin/receive.cgi').

- คำอธิบาย กำหนด URI ของโปรแกรม receive.cgi ของเครื่องที่เอเจนต์ตัวนั้นอยู่

ไฟล์ script.pl

- คำอธิบาย ไฟล์นี้เป็นได้เก็บฟังค์ชันต่างๆที่สามารถสั่งให้บราวเซอร์ทำงานได้
- ฟังก์ชันที่มี

checkmail_all

- คำอธิบาย เปิดอีเมล์ของผู้ใช้ระบบคนนั้นขึ้นมาแสดงทั้งหมด

checkmail_bysubject(Condition)

- คำอธิบาย เปิดอีเมล์ของผู้ใช้ระบบคนนั้น โดยเลือกจากชื่อหัวข้อที่มีคำตรงกับ Condition

checkmail_bydate(Condition)

- คำอธิบาย เปิดอีเมล์ของผู้ใช้ระบบคนนั้น โดยเลือกจากวันที่ ที่มีคำตรงกับ Condition

sendmail(To,Subj,Body)

- คำอธิบาย ใช้ส่งอีเมล์

ภาคผนวก ข

ซอร์สโค้ดโปรแกรม

ไฟล์ Browser.pl

```
1   :- chdir( '.\' ).
2   :- ensure_loaded( conf ).
3   :- ensure_loaded( http_util ).
4   :- ensure_loaded( proweb_interface ).
5   :- ensure_loaded(library('tcp\mail')), mail_init.
6   :- ensure_loaded( script ).
7
8   user_dialog :-
9     _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_thickframe,ws_caption,ws_border,dlg_modalframe],
10    _S2 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
11    _S3 =
12  [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_multiline,ws_vscroll,es_autohscroll,es_autovscro
13  ll,es_readonly,es_wantreturn],
14    _S4 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll],
15    _S5 = [ws_child,ws_visible,ss_left],
16    wdcreate( user_dialog,     `Prolog Web Browser`,  89, 31, 638, 504, _S1 ),
17    wccreate( (user_dialog,1000), button, `View`,     540, 30,  60,  30, _S2 ),
18    wccreate( (user_dialog,1002), button, `script`,   350, 18,  70,  22, _S2 ),
19    wccreate( (user_dialog,1003), button, `Checkmail`, 440, 20,  80,  22, _S2 ),
20    wccreate( (user_dialog,8000), edit,  ``,          10, 80, 610, 380, _S3 ),
21    wccreate( (user_dialog,8001), edit,  ``,          60, 20, 280,  20, _S4 ),
22    wccreate( (user_dialog,8002), edit,  ``,          60, 50, 460,  20, _S4 ),
23    wccreate( (user_dialog,10000), static, `Demo`,      10, 20,  40,  20, _S5 ),
24    wccreate( (user_dialog,10001), static, `Query`,     10, 50,  40,  20, _S5 ).
25
26
27  script_dialog :-
28    _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_thickframe,ws_caption,ws_border,dlg_modalframe],
29
```

```
30  _S2 =
31  [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_multiline,es_autohscroll,es_autovscroll,es_wantr
32  eturn],
33    _S3 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
34    wdcreate( script_dialog,     `Write Script`,     227, 51, 468, 257, _S1 ),
35    wccreate( (script_dialog,8000), edit, ``,          10, 10, 350, 210, _S2 ),
36    wccreate( (script_dialog,1000), button, `Run`,     370, 20, 80, 30, _S3 ),
37    wccreate( (script_dialog,1001), button, `Cancel`,  370, 60, 80, 30, _S3 ),
38    wccreate( (script_dialog,1002), button, `Ask`,     370, 100, 80, 30, _S3 ),
39    wccreate( (script_dialog,1003), button, `Tell`,    370, 140, 80, 30, _S3 ),
40    wccreate( (script_dialog,1004), button, `Send Script`, 370, 180, 80, 30, _S3 ).
41
42  ask_dialog :-
43    _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_thickframe,ws_caption,ws_border,dlg_modalframe],
44    _S2 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll,es_autovscroll],
45    _S3 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_multiline,es_autohscroll,es_autovscroll],
46    _S4 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
47    _S5 = [ws_child,ws_visible,ss_left],
48    wdcreate( ask_dialog,     `Ask`,          356, 56, 192, 151, _S1 ),
49    wccreate( (ask_dialog,8000), edit,  `Who`,     80, 20, 100, 20, _S2 ),
50    wccreate( (ask_dialog,8001), edit,  `Question?`, 80, 50, 100, 20, _S3 ),
51    wccreate( (ask_dialog,1000), button, `Ask`,    20, 30, 70, 30, _S4 ),
52    wccreate( (ask_dialog,10000), static, `Ask who?`, 10, 20, 60, 20, _S5 ),
53    wccreate( (ask_dialog,10001), static, `question :`, 10, 50, 60, 20, _S5 ),
54    wccreate( (ask_dialog,1001), button, `Cancel`,  100, 80, 70, 30, _S4 ).
55
56  tell_dialog :-
57    _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_thickframe,ws_caption,ws_border,dlg_modalframe],
58    _S2 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
59    _S3 = [ws_child,ws_visible,bs_autoradiobutton],
60    _S4 = [ws_child,ws_visible,ws_tabstop,ws_border,lbs_sort,ws_vscroll,lbs_usetabstops],
61    _S5 =
62  [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_multiline,es_autohscroll,es_autovscroll,es_wantr
63  eturn],
```

```
64      _S6 = [ws_child,ws_visible,ss_center],
65      _S7 = [ws_child,bs_groupbox,ws_visible],
66      wdcreate( tell_dialog,      ``,                 260, 106, 268, 207, _S1 ),
67      wccreate( (tell_dialog,1000), button, `Tell`,          20, 140,  60,  30, _S2 ),
68      wccreate( (tell_dialog,1001), button, `Cancel`,        180, 140,  60,  30, _S2 ),
69      wccreate( (tell_dialog,1002), button, `Forget`,        100, 140,  60,  30, _S2 ),
70      wccreate( (tell_dialog,2000), button, `Tell Data`,      30,  70,  70,  20, _S3 ),
71      wccreate( (tell_dialog,2001), button, `Tell File`,      30, 100,  70,  20, _S3 ),
72      wccreate( (tell_dialog,4000), listbox, `List1`,        120,  10, 120,  20, _S4 ),
73      wccreate( (tell_dialog,8001), edit,   ``,              120,  70, 120,  60, _S5 ),
74      wccreate( (tell_dialog,10000), static, `Select Agent`,   20,  10,  90,  20, _S6 ),
75      wccreate( (tell_dialog,10001), static, `Data or FileName`, 120,  50, 120,  20, _S6 ),
76      wccreate( (tell_dialog,11000), button, `Type :`,        20,  50,  90,  80, _S7 ),
77      wbtnsel( (tell_dialog,2000), 1 ),
78      forall( proweb_server(ServerName,_),
79                      ( to_write(ServerName)~>S,
80                        wlbxadd( (tell_dialog,4000), -1, S) )
81              ).
82
83  send_script :-
84      _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_thickframe,ws_caption,ws_border,dlg_modalframe],
85      _S2 = [ws_child,ws_visible,ss_center],
86      _S3 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
87      _S4 = [ws_child,ws_visible,ws_tabstop,ws_border,lbs_sort,ws_vscroll,lbs_usetabstops],
88      wdcreate( send_script,      `Send Script`,          260, 138, 238, 137, _S1 ),
89      wccreate( (send_script,10000), static, `Select Agent :`,  10,  10,  90,  20, _S2 ),
90      wccreate( (send_script,1000), button, `Send`,          30,  60,  60,  30, _S3 ),
91      wccreate( (send_script,1001), button, `Cancel`,        130,  60,  60,  30, _S3 ),
92      wccreate( (send_script,4000), listbox, `List1`,        110,  10, 110,  20, _S4 ),
93      forall( proweb_server(ServerName,_),
94                      ( to_write(ServerName)~>S,
95                        wlbxadd( (send_script,4000), -1, S) )
96              ).
97
```

```
98  login :-
99    _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_caption,ws_border,dlg_modalframe],
100   _S2 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll],
101   _S3 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll,es_password],
102   _S4 = [ws_child,ws_visible,ss_left],
103   _S5 = [ws_child,ws_visible,ss_center],
104   _S6 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
105   wdcreate( login,      `L O G I N`,    220, 41, 326, 195, _S1 ),
106   wccreate( (login,8000), edit, ``,       90, 40, 210, 20, _S2 ),
107   wccreate( (login,8001), edit, ``,       90, 80, 210, 20, _S3 ),
108   wccreate( (login,10000), static, `User Name`, 10, 40, 70, 20, _S4 ),
109   wccreate( (login,10001), static, `Password`, 10, 80, 70, 20, _S4 ),
110   wccreate( (login,10002), static, `Login`,    10, 10, 300, 20, _S5 ),
111   wccreate( (login,1000), button, `Login`,    40, 120, 70, 30, _S6 ),
112   wccreate( (login,1001), button, `Exit`,     220, 120, 70, 30, _S6 ),
113   wccreate( (login,1002), button, `New User`, 130, 120, 70, 30, _S6 ).
114 newuser :-
115   _S1 = [dlg_ownedbydesktop,ws_sysmenu,ws_caption,ws_border,dlg_modalframe],
116   _S2 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll],
117   _S3 = [ws_child,ws_visible,ws_tabstop,ws_border,es_left,es_autohscroll,es_password],
118   _S4 = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
119   _S5 = [ws_child,ws_visible,ss_left],
120   _S6 = [ws_child,ws_visible,ws_tabstop,ss_left],
121   wdcreate( newuser,      `PROFILE FORM`,    221, 41, 256, 325, _S1 ),
122   wccreate( (newuser,8000), edit, ``,        80, 20, 160, 20, _S2 ),
123   wccreate( (newuser,8001), edit, ``,        80, 50, 160, 20, _S2 ),
124   wccreate( (newuser,8002), edit, ``,        80, 80, 160, 20, _S3 ),
125   wccreate( (newuser,8003), edit, ``,        80, 110, 160, 20, _S3 ),
126   wccreate( (newuser,8004), edit, ``,        80, 150, 160, 20, _S2 ),
127   wccreate( (newuser,8005), edit, ``,        80, 180, 160, 20, _S2 ),
128   wccreate( (newuser,8006), edit, ``,        80, 210. 160, 20, _S2 ),
129   wccreate( (newuser,1000), button, `O K`,      30, 250, 70, 30, _S4 ),
130   wccreate( (newuser,1002), button, `CLEAR`,    150, 250, 70, 30, _S4 ),
131   wccreate( (newuser,10000), static, `User Name`,  10, 20, 70, 20, _S5 ),
```

```
132     wccreate( (newuser,10001), static, `e-mail`,          10,  50,  70,  20, _S5 ),
133     wccreate( (newuser,10002), static, `Password`,        10,  80,  70,  20, _S5 ),
134     wccreate( (newuser,10003), static, `Re-enter Password`, 10, 110,  70,  30, _S6 ),
135     wccreate( (newuser,10004), static, `First Name`,      10, 150,  70,  20, _S5 ),
136     wccreate( (newuser,10005), static, `Last Name`,       10, 180,  70,  20, _S5 ),
137     wccreate( (newuser,10006), static, `Nick Name`,       10, 210,  70,  20, _S5 ).
138
139   prolog_browser :-
140           login,user_dialog,script_dialog,ask_dialog,tell_dialog,newuser,send_script,
141           window_handler( login          ,message_handler ),
142           window_handler( user_dialog    ,message_handler ),
143           window_handler( script_dialog  ,message_handler ),
144           window_handler( ask_dialog     ,message_handler ),
145           window_handler( tell_dialog    ,message_handler ),
146           window_handler( newuser        ,message_handler ),
147           window_handler( send_script    ,message_handler ),
148           show_dialog(login).
149
150   message_handler( user_dialog, msg_close, _, _) :-
151           wshow( user_dialog, 0 ),
152           retractall( user_password( _, _) ),
153           retractall( my_name(_) ),
154           show_dialog(login),!.
155   message_handler( Window, msg_close, _, _ ):- wshow( Window, 0 ),!.
156   message_handler( (Window,1001), msg_button, 0, _ ) :- wshow( Window, 0 ),!.
157
158   message_handler( Windows, Message, Data, Result) :-
159           handler( Windows, Message, Data, Result).
160
161   handler( (send_script,1000), msg_button, _, _ ) :-
162           get_edit_box( (script_dialog,8000), Script ),
163           not_empty(Script),
164           get_selected_data_from_listbox((send_script,4000), String),
165           wshow( send_script, 0 ),
```

```
166             convert_str_to_atom(String, Agent),
167             convert_str_to_atom(Script,  Atom),
168             rewrite_edit_box( (user_dialog,8000), `Script running.~M~J` ),
169             run_script( Agent, Atom, ErrorCode ),
170             Window = (user_dialog,8000),
171             ( ErrorCode == 0 -> write_edit_box(Window,`~M~JRun script ready~M~J`);
172              ErrorCode == 1 -> write_edit_box(Window,`~M~JRun script fail~M~J`);
173                     write_edit_box(Window,`~M~JRun script error~M~J`) ).
174
175     handler( (newuser,1000), msg_button, _, _ ) :-
176             get_edit_box( (newuser,8000),       Uname ),
177             get_edit_box( (newuser,8001),       Email ),
178             get_edit_box( (newuser,8002),       Pass1 ),
179             get_edit_box( (newuser,8003),       Pass2 ),
180             get_edit_box( (newuser,8004),       Fname ),
181             get_edit_box( (newuser,8005),       Lname ),
182             get_edit_box( (newuser,8006),       Nname ),
183             nonvar(Uname), Uname \= ``,
184             nonvar(Email), Email \= ``,
185             nonvar(Pass1), Pass1 \= ``,
186             nonvar(Pass2), Pass2 \= ``,!,
187             ( check_user_exist( Uname ) ->
188                     % This User had Exist
189                     message_box(ok, `This Username are already exist!!`,_),
190                     !,clear_new_user_form;
191                     ( (Pass1 == Pass2, check_email_password(Email,Pass1)) ->
192                             !,
193                             new_user_profile(Uname,Email,Pass1,Fname,Lname,Nname),
194                             !,wshow(newuser,0),
195                             rewrite_edit_box((login,8000),Uname),
196                             rewrite_edit_box((login,8001),Pass1);
197                             message_box(ok, `Incorrect Password`,_)
198                     )
199             ).
```

```
200
201  handler( (newuser,1002), msg_button, _, _ ) :- clear_new_user_form.
202  clear_new_user_form :-
203          clear_edit_box( (newuser,8000) ),
204          clear_edit_box( (newuser,8001) ),
205          clear_edit_box( (newuser,8002) ),
206          clear_edit_box( (newuser,8003) ),
207          clear_edit_box( (newuser,8004) ),
208          clear_edit_box( (newuser,8005) ),
209          clear_edit_box( (newuser,8006) ).
210
211  handler( (login,Id), Msg, Data, _ ) :-
212          ((Id == 1000, Msg == msg_button);
213           (Msg == msg_key, Data = (_,_,13))),
214          wenable((login,1000), IsEnable),!, IsEnable == 1,
215          wenable((login,1000), 0),
216          get_edit_box((login,8000),Uname),
217          get_edit_box((login,8001),Pass),
218          ( system_login( Uname, Pass ) ->
219                  clear_edit_box((user_dialog,8000)),
220                  clear_edit_box((user_dialog,8001)),
221                  clear_edit_box((user_dialog,8002)),
222                  wshow( login, 0 ),                    /* Close this windows*/
223                  show_dialog( user_dialog );           /* And open Browser */
224                  message_box(ok, `User ID does not exist Or Password Incorrect`,_)
225          ),!,
226          wenable((login,1000), 1),
227          clear_edit_box( (login,8001) ).
228
229  handler( (login,1002), msg_button, _, _ ) :-
230          show_dialog(newuser).
231
232  handler( (user_dialog,1000), msg_button, _, _ ) :-
233          Window = (user_dialog,8000),
```

```
234             get_address(Host,URI),
235             clear_edit_box(Window),
236             write_edit_box(Window,`กำลังค้นหาคำตอบอยู่`),
237             atom_string(Agent, Host),
238             ( convert_str_to_atom(URI,Query); atom_string(Query, URI) ),
239             assert( data(demo(Agent, Query, Ans )) ),
240             ( demo(Agent, Query, Ans ) ->
241               rewrite_edit_box(Window,Ans);
242               clear_edit_box(Window),
243               write_edit_box(Window,`ไม่สามารถหาคำตอบได้`)
244             ),!.
245
246     handler( (user_dialog,1003), msg_button, _, _ ) :-
247             Window = (user_dialog,8000),
248             my_name(ID),
249             user_email(ID,Address,Password),
250             write(ID)~> Name,
251             cat( [`กำลังตรวจสอบ e-mail ของ `, Name,`[`, Address, `]`, `กรุณารอ...~M~J` ],
252     WaitMessage,_ ),
253             rewrite_edit_box(Window,WaitMessage),
254             email_check.
255     handler( (user_dialog,1002), msg_button, _, _ ) :-
256             show_dialog(script_dialog).
257     handler( (script_dialog,1000), msg_button, _, _ ) :-
258             get_edit_box( (script_dialog,8000) ,Data),
259             not_empty(Data),
260             Window = (user_dialog,8000),
261             clear_edit_box(Window),
262             convert_str_to_atom(Data,Atom),
263             wshow( script_dialog, 0 ),
264             rewrite_edit_box( Window, `Script running.~M~J` ),
265             catch(ErrorCode, solve(Atom)),
266             ( ErrorCode == 0 -> write_edit_box(Window,`~M~JRun script ready~M~J`);
267               ErrorCode == 1 -> write_edit_box(Window,`~M~JRun script fail~M~J`);
```

```
268                     write_edit_box(Window,`~M~JRun script error~M~J`) ).
269
270 handler( (script_dialog,1002), msg_button, _, _ ) :-
271             wshow( script_dialog, 0 ),
272             show_dialog(ask_dialog).
273 handler( (script_dialog,1003), msg_button, _, _ ) :-
274             wshow( script_dialog, 0 ),
275             show_dialog(tell_dialog).
276 handler( (script_dialog,1004), msg_button, _, _ ) :-
277             show_dialog(send_script).
278 handler( (ask_dialog,1000), msg_button, _, _ ) :-
279             get_edit_box( (ask_dialog,8000) ,Agent_s ),
280             get_edit_box( (ask_dialog,8001) ,Question_s),
281             atom_string( Agent, Agent_s ),
282             convert_str_to_atom( Question_s, Question ),
283             ( write(`You Ask `),write(Agent),write(` that `),
284               write(Question),write(`.`) )->>Msg,
285             show_answer(Msg),
286             wshow( ask_dialog, 0 ),
287             ( ask(Agent, Question, Ans) ->
288                     convert_str_to_atom(Ans,ResultList),
289                     Window = (user_dialog,8000),
290                     write_edit_box(Window,`~M~JAsk Result ~M~J`),
291                     forall( member(A,ResultList),
292                             ( write_edit_box(Window,`~J`),
293                               write_edit_box(Window,A),
294                               write_edit_box(Window,`~M~J`)
295                             )
296                     );
297                     catch(Error, wait_for_mail),
298                     ( Error == 0 ->
299                             retractall(wait_for_mail),
300                             show_error(`No Result now! You can wait your result later~M~J`);
301                             show_error(`No Result`)
```

```
302                     )
303             ).
304     handler( (tell_dialog,1000), msg_button, _, _ ) :-
305             get_edit_box( (tell_dialog,8001), Data_s ),
306             not_empty(Data_s),
307             get_selected_data_from_listbox((tell_dialog,4000), String),
308             convert_str_to_atom(String, Agent),
309             wbtnsel((tell_dialog,2000),A),
310             wbtnsel((tell_dialog,2001),B),
311             ( ( A == 1, B == 0) -> Flag = data, convert_str_to_atom(Data_s, Data);
312              ( A == 0, B == 1) -> Flag = file, atom_string( Data , Data_s );
313              show_error(`You must select type!`),!,fail
314             ),
315
316             ( write(`Sending request: Tell `), write(Flag), write(` to `), write(Agent),
317              write(` that `), write(Data), write(`.`),nl )~>Send_data,
318             show_answer(Send_data),
319             wshow( tell_dialog, 0 ),
320             assert( cmd(tell(Agent, Flag, Data)) ),
321             ( tell(Agent, Flag, Data)->
322              (write(`Sending `),write(Flag),write(` ok`))~>Message,
323              show_answer(Message);
324              show_error(`ไม่สามารถส่งข้อมูลได้`) ).
325     handler( (tell_dialog,1002), msg_button, _, _ ) :-
326             get_edit_box( (tell_dialog,8001), Data_s ),
327             not_empty(Data_s),
328             get_selected_data_from_listbox((tell_dialog,4000), String),
329             convert_str_to_atom(String, Agent),
330             wbtnsel((tell_dialog,2000),A),
331             wbtnsel((tell_dialog,2001),B),
332             ( ( A == 1, B == 0) -> Flag = predicate, convert_str_to_atom(Data_s, Data);
333              ( A == 0, B == 1) -> Flag = file,  Data = Data_s ;
334              show_error(`You must select type!`),!,fail
335             ),
```

```
336
337             ( write(`Sending request: Forget `), write(Flag), write(` to `), write(Agent),
338              write(` that `), write(Data), write(`.`),nl )~>Send_data,
339             show_answer(Send_data),
340             wshow( tell_dialog, 0 ),
341             ( forget(Agent, Flag, Data)->
342              (write(`Sending `),write(Flag),write(` ok`))~>Message,
343              show_answer(Message);
344              show_error(`ไม่สามารถส่งข้อมูลได้`) ).
345
346     get_selected_data_from_listbox(Window, String) :-
347             find_selected_position(Window, 0, Point),
348             wlbxget( Window, Point, String ).
349
350     find_selected_position(Window, Start, Point) :-
351             wlbxsel( Window, Start, State ),
352             ( State == 1 ->
353                     Point = Start;
354                     Next is Start + 1,
355                     find_selected_position(Window, Next, Point)
356             ),!.
357      get_edit_box(Window,Data) :-
358             wcount(Window,LastChar,_,_),
359             wedtsel( Window,0,LastChar),
360             wedttxt( Window,Data).
361      get_address(Host,URI) :-
362             get_edit_box((user_dialog,8001),Host),
363             get_edit_box((user_dialog,8002),Temp ),
364             ( (Temp == ``; var(Temp)) -> URI = `/`; URI = Temp ).
365      clear_edit_box(Window) :-
366             wcount(Window,LastChar,A,B),
367             wedtsel( Window,0,LastChar),
368             wedttxt( Window,``). % rewrite data with NULL
369      show_answer(Data) :-
```

```
370             write(Data)~>String,
371             rewrite_edit_box((user_dialog,8000),String).
372     show_error(Data) :-
373             (nl,write(Data))~>String,
374             write_edit_box((user_dialog,8000),String).
375     write_edit_box(Window,Data) :- write(Data)~>String, wedttxt( Window,String).
376     rewrite_edit_box(Window,String) :-
377             clear_edit_box(Window),
378             write_edit_box(Window,String).
379
380     system_login(Uname,Pass) :-
381             nonvar(Uname),  Uname \= ``,
382             nonvar(Pass),       Pass    \= ``, !,
383             near_server(Agent),
384             demo( Agent, user_profile(Uname,_,_,_,_), ProfileList ),!,
385             demo( Agent, user_password(Uname,Pass), PassList ),!,
386             len( ProfileList, 1 ),          len( PassList, 1 ),
387             member(A, ProfileList),     member(B, PassList),
388             ( retract(A); true ),           assert(A),
389             assert(B),
390             retractall( my_name(_) ),
391             assert( my_name(Uname) ),!.
392
393     new_user_profile(Uname,Email,Pass,Fname,Lname,Nname) :-
394             near_server(Agent),!,
395             run_script(Agent, add_user_profile(Uname,Email,Pass,Fname,Lname,Nname), ReturnCode ).
396     check_user_exist(Uname) :-
397             near_server(Agent),
398             demo(Agent, exist_user(Uname), AllUserList),!,
399             len(AllUserList,1),!.
400
401     email_check :-
402             my_name(ID),
403             user_email(ID,Address,Password),
```

```
404             find_mail_server( Address, User, Server ),
405             check_email(Server,User,Password),!.
406     find_mail_server( Address, User, Server ) :-
407         find_word(Address,`@`,Point),
408         string_cut(Address,Point,-1,Emailserver),
409         Len is Point-1,
410         string_cut(Address,0,Len,User),
411         (Emailserver == `kmitl.ac.th`->
412             Server = `mail.kmitl.ac.th`;
413          Emailserver == `ce.kmitl.ac.th` ->
414             Server = `diamond.ce.kmitl.ac.th`
415         ).
416
417     check_email_password(Email,Pass) :-
418             find_mail_server( Email, User, Server ),
419             mail_check( Server, _, User, Pass, _),!.
420     check_have_new_proweb_mail( Email,Pass ) :-
421             find_mail_server( Email, User, Server ),
422             del('mail.txt'),
423             mail_get_if(Server,_,User,Pass,0,'mail.txt',(`Subject`,contains,any_of,[`Proweb Tell`]),_,Got),
424             !, Got \= 0,
425             del('mail.txt').
426     check_email(Server,Username,Pass) :-
427             Window = (user_dialog,8000),
428             MailFile = 'mail.txt',
429             mail_check( Server, _, Username, Pass, NoOfMsgs),
430             (NoOfMsgs == 0 ->
431              ( write_edit_box(Window,`ไม่มีข้อมูลใหม่`) );
432           (mail_get_if(Server,_,Username,Pass,1,MailFile,(`Subject`,contains,any_of,[`Proweb
433      Tell`]),Total,Got),
434             (Got == 0 ->
435                     ( write_edit_box(Window,`ไม่มีข้อมูลใหม่`) );
436               (mail_parse_all(MailFile,Msgs,Parse),
437                del(MailFile),
```

```
438                     ( write( ` There are `),
439                       write( Got ),
440                       write( ` new mail from proweb server. `)
441                     ) ~> Message,
442             message_box(ok, Message, _ ),
443             forall( member( (From,To,Cc,Bcc,Subj,Date,Header,Body,Attach) , Msgs ),
444                 ( write_edit_box(Window,Subj),
445                                 write_edit_box(Window,`~IAnswer is~M~J`),
446                                 fcreate(Body,Body,0,0),
447                                 input(Body),
448                                 fread(s,0,0,ResultList_s),
449                                 convert_str_to_atom(ResultList_s,ResultList),
450                                 forall( member(A,ResultList),
451                                  ( write_edit_box(Window,`~I`),
452                                    write_edit_box(Window,A),
453                                    write_edit_box(Window,`~M~J`))
454                                 ),
455                                 write_edit_box(Window,`~M~J`),
456                                 fclose(Body)
457                   )
458                 ) /*forall*/
459             ) /*mail_parse_all*/
460           ) /*Got == 0*/
461         ) /*mail_get_if*/
462       ).
463
464   find_serv_user_pass(Server,User,Password) :-
465             my_name(ID),
466             user_email(ID,Address,Password),
467             find_mail_server( Address, User, Server ).
468
469   :- prolog_browser
```

ไฟล์ pw_demo.pl

```
:- multifile( proweb_page     / 2 ).
:- multifile( proweb_form     / 2 ).
:- multifile( proweb_question / 2 ).
:- ensure_loaded( examples(sendmail) ).
:- ensure_loaded( examples(util) ).
:- ensure_loaded( examples(conf) ).
:- ensure_loaded( examples(http_util) ).
:- ensure_loaded( examples(proweb_interface) ).
:- dynamic( ask_other_server_debug/1 ).

main_goal :-
        proweb_send_form( user_form ),
        proweb_returned_answer(userdata, Userdata ),
        proweb_post_reply(userdata, Userdata ),
        proweb_send_form( select_form(select1) ),
        proweb_returned_answer( select1, Select1 ),
        ( Select1 == 'tell' ->
                ( proweb_send_form(select_form(select2)),
                  proweb_returned_answer( select2, Select2 ),
                  function_tell(Select2)
                ); /* End Select1 == 'tell'*/
          Select1 == 'ask' ->
                ( proweb_send_form(ask_form),
                  proweb_returned_answer(askdata, Askdata ),
                  proweb_post_reply(askdata, Askdata ),
                  convert_str_to_atom(Askdata,Atom2),
                  ( ( findAns( Atom2, Answer2 ),len(Answer2,Len2),Len2 \=0 ) ->
                        ( to_write(Answer2)~>Temp_Answer2,
                          proweb_post_reply( solution, Temp_Answer2),
                          proweb_send_form( ans_ask_form )
                        );
                        ( proweb_send_form( select3_form ),
```

```
33          proweb_returned_answer( select3, Select3 ),
34          ( Select3 = 'Client tell Proweb Server' ->
35                  ( proweb_send_form(select_form(select2)),
36                    proweb_returned_answer( select2, Select5 ),
37                    function_tell(Select5)
38                  );
39                  ( /*'Tell another Proweb Server'*/
40                    ( ( proweb_data_uco(MyUCO),
41                            ask_other_server_debug( MyUCO) ) ->
42                            AllResultList = [];
43                            ask_other_server( Askdata,  AllResultList )
44                    ),
45                    len( AllResultList, Len3 ),
46                    ( Len3 \= 0  ->
47                      ( to_write( AllResultList )~> AllResultList_s,
48                        proweb_post_reply( solution, AllResultList_s),
49                        proweb_send_form( ans_ask_form ) );
50                        proweb_data_uco( MyUCO ),
51                        assert( ask_other_server_debug( MyUCO ) ),
52                        proweb_send_form(know_unknow_form),
53                        proweb_returned_answer( select4, Select4 ),
54                        retractall( ask_other_server_debug( MyUCO ) ),
55                        ( Select4 = 'Tell' ->
56                                ( proweb_send_form(select_form(select2)),
57                                  proweb_returned_answer( select2, Select6 ),
58                                  function_tell(Select6)
59                                );
60                                ( convert_str_to_atom(Askdata,Atom5),
61                                  proweb_posted_reply(userdata, Userdata ),
62                                  catch(Error1, mail(Atom5, Userdata)),
63                                  ( Error1 == 0 ->
64                                          proweb_send_form(ans_unknow_form);
65                                          ( (user_email(Userdata,Mail,_),
66                                            nonvar(Mail)) ->
```

```
67                                                    asserta(mail(Atom5, Userdata));
68                                                          true
69                                                    ),
70                                                  proweb_send_form(ans_unknow_form)
71                                                ) /*Error1 == 0*/
72                                              )
73                                            ) /*Select4 = 'Tell'*/
74                                          ) /*Len3 \= 0*/
75                                        ) /*'Tell another Proweb Server' */
76                                      ) /* Select3 = 'Client tell Proweb Server' */
77                                    )
78                                  ) /* ( findAns( Atom2, Answer2 ) */
79                                ); /* End Select1 == 'ask'*/
80                        Select1 = 'run script' ->
81                          ( proweb_send_form( script_form ),
82                            proweb_returned_answer(scriptdata, Scriptdata ),
83                            proweb_post_reply(scriptdata, Scriptdata ),
84                            convert_str_to_atom(Scriptdata.Atom3),
85                            catch( Error, solve(Atom3) ),
86                            ( Error == 0  -> Respond = `Run script ready`;
87                              Error == -1 -> Respond = `Run script fail` ;
88                                             Respond = `Run script Error`
89                            ),
90                            proweb_send_form( ans_script_form(Respond) )
91                          ) /* End Select1 = 'run script'*/
92                        ).
93
94  proweb_page( _,
95          [ include('html/project/head.htm'),
96            'Project',
97            include('html/project/body.htm'),
98            proweb(select_form),
99            include('html/project/foot.htm')
100         ] ).
```

```
101
102  proweb_form( user_form,
103          [ p, /p,
104           table(border='0', 'width=100%'), tr(align='center'), td,
105           h2 @ 'Web based Multi-agent Communication',
106           /td, /tr,
107           tr(align='center'), td,
108           h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
109           /td, /tr, /table,
110           p, /p,
111            table('WIDTH=90%',align='center'), tr(align='center'), td,
112           unencoded @ 'User Name',
113           /td, /tr,
114           tr(align='center'), td,
115           ?userdata,
116            /td, /tr, /table,
117            table('WIDTH=50%',align='center'), tr, td('WIDTH=50%',align='center'),
118            input(type=submit, value=`OK`),
119            /td,
120           td('WIDTH=50%',align='center'),
121            input(type=reset),
122             /td, /tr, /table,
123            p, /p,
124           table(border='0', 'width=100%'), tr(align='center'), td,
125           h2 @ 'Department of Computer Engineering',
126           /td, /tr,
127           tr(align='center'), td,
128           h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
129            /td, /tr, /table ] ).
130
131   proweb_form( script_form,
132           [ p, /p,
133            table(border='0', 'width=100%'), tr(align='center'), td,
134            h2 @ 'Web based Multi-agent Communication',
```

```
135             /td, /tr,
136             tr(align='center'), td,
137             h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
138             /td, /tr, /table,
139             p, /p,
140              table('WIDTH=90%',align='center'), tr(align='center'), td,
141             unencoded @ 'Run Script ',
142             /td, /tr,
143             tr(align='center'), td,
144             ?scriptdata,
145              /td, /tr, /table,
146              table('WIDTH=50%',align='center'), tr, td('WIDTH=50%',align='center'),
147              input(type=submit, value=`OK`),
148              /td,
149             td('WIDTH=50%',align='center'),
150              input(type=reset),
151               /td, /tr, /table,
152               p, /p,
153              table(border='0', 'width=100%'), tr(align='center'), td,
154              h2 @ 'Department of Computer Engineering',
155              /td, /tr,
156              tr(align='center'), td,
157             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
158              /td, /tr, /table ] ).
159
160     proweb_form( ans_script_form(R),
161             [ p, /p,
162              table(border='0', 'width=100%'), tr(align='center'), td,
163              h2 @ 'Web based Multi-agent Communication',
164              /td, /tr,
165              tr(align='center'), td,
166              h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
167              /td, /tr,  /table,
168              p,  /p,
```

```
169             table('WIDTH=100%'), tr, td(align='center'),
170             R,
171              /td, /tr, /table,
172             p, /p,
173             table(border='0', 'width=100%'), tr(align='center'), td,
174             h2 @ 'Department of Computer Engineering',
175             /td, /tr,
176             tr(align='center'), td,
177             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
178             /td, /tr, /table ] ).
179
180     proweb_form( select_form(S),
181             [ p, /p,
182              table(border='0', 'width=100%'), tr(align='center'), td,
183              h2 @ 'Web based Multi-agent Communication',
184              /td, /tr,
185              tr(align='center'), td,
186              h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
187              /td, /tr, /table,
188              p, /p,
189              table('WIDTH=50%',align='center'), tr, td(align='center'),
190             ?S,
191              /td, /tr,
192             tr, td(align='center'),
193              input(type=submit, value=`OK`),
194              /td, /tr, /table,
195              p, /p,
196              table(border='0', 'width=100%'), tr(align='center'), td,
197              h2 @ 'Department of Computer Engineering',
198              /td, /tr,
199              tr(align='center'), td,
200              h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
201              /td, /tr, /table ] ).
202
```

```
203  proweb_form( tell_form,
204          [ p, /p,
205           table(border='0', 'width=100%'), tr(align='center'), td,
206           h2 @ 'Web based Multi-agent Communication',
207           /td, /tr,
208           tr(align='center'), td,
209           h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
210           /td, /tr, /table,
211           p, /p,
212           table('WIDTH=90%',align='center'), tr(align='center'), td,
213           unencoded @ 'Tell',
214           /td, /tr,
215           tr(align='center'), td,
216           ?telldata,
217           /td, /tr, /table,
218           table('WIDTH=100%',align='center'), tr, td('WIDTH=50%',align='center'),
219           input(type=submit, value=`OK`),
220           /td,
221           td('WIDTH=50%',align='center'),
222           input(type=reset),
223           /td, /tr, /table,
224           p, /p,
225           table(border='0', 'width=100%'), tr(align='center'), td,
226           h2 @ 'Department of Computer Engineering',
227           /td, /tr,
228           tr(align='center'), td,
229           h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
230           /td, /tr, /table ] ).
231
232  proweb_form( sendfile_form,
233          [ p, /p,
234           table(border='0', 'width=100%'), tr(align='center'), td,
235           h2 @ 'Web based Multi-agent Communication',
236           /td, /tr,
```

```
237             tr(align='center'), td,
238             h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
239             /td, /tr, /table,
240             p, /p,
241             table('WIDTH=90%',align='center'), tr(align='center'), td,
242             unencoded @ 'Type file name',
243             /td, /tr,
244             tr(align='center'), td,
245             ?filename,
246              /td, /tr, /table,
247              table('WIDTH=100%',align='center'), tr, td('WIDTH=50%',align='center'),
248              input(type=submit, value=`OK`),
249              /td,
250             td('WIDTH=50%',align='center'),
251              input(type=reset),
252               /td, /tr, /table,
253               p, /p,
254              table(border='0', 'width=100%'), tr(align='center'), td,
255              h2 @ 'Department of Computer Engineering',
256              /td, /tr,
257              tr(align='center'), td,
258             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
259              /td, /tr, /table ] ).
260
261     proweb_form( ask_form,
262             [ p, /p,
263              table(border='0', 'width=100%'), tr(align='center'), td,
264              h2 @ 'Web based Multi-agent Communication',
265              /td, /tr,
266              tr(align='center'), td,
267              h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
268              /td, /tr,  /table,
269              p, /p,
270              table('WIDTH=90%',align='center'), tr(align='center'), td,
```

```
271             unencoded @ 'Ask',
272            /td, /tr,
273             tr(align='center'), td,
274             ?askdata,
275              /td, /tr, /table,
276             table('WIDTH=100%',align='center'), tr, td('WIDTH=50%',align='center'),
277             input(type=submit, value=`OK`),
278              /td,
279              td('WIDTH=50%',align='center'),
280              input(type=reset),
281              /td, /tr, /table,
282              p, /p,
283             table(border='0', 'width=100%'), tr(align='center'), td,
284             h2 @ 'Department of Computer Engineering',
285             /td, /tr,
286             tr(align='center'), td,
287             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
288              /td, /tr,  /table ] ).
289
290    proweb_form( select3_form,
291            [ p, /p,
292             table(border='0', 'width=100%'), tr(align='center'), td,
293              h2 @ 'Web based Multi-agent Communication',
294              /td, /tr,
295              tr(align='center'), td,
296              h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
297              /td, /tr,  /table,
298               p, /p,
299               table('WIDTH=100%'), tr, td(align='center'),
300               'Server can not find the solution for you.  ',
301               /td, /tr, tr, td(align='center'),
302               'What do you want to Proweb Server do? ',
303                /td, /tr, /table,
304               p, /p,
```

```
305             table('WIDTH=50%',align='center'), tr, td(align='center'),
306             ?select3,
307             /td, /tr,
308             tr, td(align='center'),
309              input(type=submit, value=`OK`),
310              /td, /tr, /table,
311              p, /p,
312             table(border='0', 'width=100%'), tr(align='center'), td,
313             h2 @ 'Department of Computer Engineering',
314             /td, /tr,
315             tr(align='center'), td,
316             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
317             /td, /tr, /table ] ).
318
319     proweb_form( know_unknow_form,
320             [ p, /p,
321              table(border='0', 'width=100%'), tr(align='center'), td,
322              h2 @ 'Web based Multi-agent Communication',
323              /td, /tr,
324              tr(align='center'), td,
325              h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
326              /td, /tr,  /table,
327               p, /p,
328               table('WIDTH=100%'), tr, td(align='center'),
329               ' All proweb server don"t know the solution. ',
330               /td, /tr, tr, td(align='center'),
331               ' Do you know the solution?  ',
332               /td, /tr, tr, td(align='center'),
333               ' If you know the solution, please tell me. ',
334                /td, /tr, /table,
335               p, /p,
336               table('WIDTH=50%',align='center'), tr, td(align='center'),
337               ?select4,
338               /td, /tr,
```

```
339             tr, td(align='center'),
340              input(type=submit, value=`OK`),
341              /td, /tr, /table,
342              p, /p,
343             table(border='0', 'width=100%'), tr(align='center'), td,
344             h2 @ 'Department of Computer Engineering',
345             /td, /tr,
346             tr(align='center'), td,
347            h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
348             /td, /tr, /table ] ).
349
350    proweb_form( ans_tell_form,
351            [ p, /p,
352             table(border='0', 'width=100%'), tr(align='center'), td,
353             h2 @ 'Web based Multi-agent Communication',
354             /td, /tr,
355             tr(align='center'), td,
356             h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
357             /td, /tr,  /table,
358             p,  /p,
359              table('WIDTH=100%'), tr, td(align='center'),
360              'You are tell me that  ',
361              ??telldata,
362              ' .',
363               /td, /tr, /table,
364               p, /p,
365              table(border='0', 'width=100%'), tr(align='center'), td,
366              h2 @ 'Department of Computer Engineering',
367              /td, /tr,
368              tr(align='center'), td,
369              h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
370              /td, /tr, /table ] ).
371
372     proweb_form( ans_sendfile_form,
```

```
373          [ p, /p,
374           table(border='0', 'width=100%'), tr(align='center'), td,
375           h2 @ 'Web based Multi-agent Communication',
376           /td, /tr,
377           tr(align='center'), td,
378           h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
379           /td, /tr,  /table,
380           p,  /p,
381            table('WIDTH=100%'), tr, td(align='center'),
382            'You send file ',
383            ??filename,
384            ' to me.',
385             /td, /tr, /table,
386            p, /p,
387           table(border='0', 'width=100%'), tr(align='center'), td,
388           h2 @ 'Department of Computer Engineering',
389           /td, /tr,
390           tr(align='center'), td,
391           h2 @ 'King Mongkut''s Institute of Technology Ladkrabang',
392           /td, /tr, /table ] ).
393
394    proweb_form( ans_ask_form,
395          [ p, /p,
396           table(border='0', 'width=100%'), tr(align='center'), td,
397           h2 @ 'Web based Multi-agent Communication',
398           /td, /tr,
399           tr(align='center'), td,
400           h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
401           /td, /tr,  /table,
402            p, /p,
403            table('WIDTH=100%'), tr, td(align='center'),
404            'You are ask me that  ',
405            ??askdata,
406            '.',
```

```
407             /td, /tr,
408             tr, td(align='center'),
409            'The answer is ',
410             ??solution,
411             ' .',
412             /td, /tr,/table,
413             p, /p,
414            table(border='0', 'width=100%'), tr(align='center'), td,
415            h2 @ 'Department of Computer Engineering',
416            /td,  /tr,
417            tr(align='center'), td,
418            h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
419            /td, /tr, /table ] ).
420
421    proweb_form( ans_unknow_form,
422           [ p, /p,
423            table(border='0', 'width=100%'), tr(align='center'), td,
424            h2 @ 'Web based Multi-agent Communication',
425            /td, /tr,
426            tr(align='center'), td,
427            h2 @ unencoded @ 'การติดต่อสื่อสารของเอเจนต์หลายตัวบนอินเตอร์เน็ต',
428            /td, /tr,  /table,
429            p,  /p,
430             table('WIDTH=100%'), tr, td(align='center'),
431             ' When I know the solution. I will send the solution to your e-mail. ',
432              /td, /tr, /table,
433              p, /p,
434             table(border='0', 'width=100%'), tr(align='center'), td,
435             h2 @ 'Department of Computer Engineering',
436             /td, /tr,
437             tr(align='center'), td,
438             h2 @ 'King Mongkut"s Institute of Technology Ladkrabang',
439             /td, /tr, /table ] ).
440
```

```
441  proweb_question( userdata,
442              [ method   = input,
443                type     = string,
444                rows     = 1,
445               cols      = 50,
446               maxlength = 175
447              ] ).
448
449  proweb_question( scriptdata,
450              [ method   = input,
451                type     = string,
452                rows     = 1,
453               cols      = 100,
454               maxlength = 175
455              ] ).
456
457  proweb_question( select1,
458              [ method = menubox,
459                select = ['tell','ask','run script'],
460                prefill = 'tell',
461                rows    = 3
462              ] ).
463
464  proweb_question( select2,
465              [ method = menubox,
466                select = ['tell by URL','send file'],
467                prefill = 'tell by URL',
468                rows    = 2
469              ] ).
470
471  proweb_question( select3,
472              [ method = menubox,
473                select = ['Tell another Proweb Server','Client tell Proweb Server'],
474                prefill = 'Tell another Proweb Server',
```

```
475                 rows     = 2
476                ] ).
477
478     proweb_question( select4,
479                [ method = menubox,
480                  select   = ['Tell','Client doesn"t know the solution'],
481                  prefill  = 'Tell',
482                  rows     = 2
483                ] ).
484
485     proweb_question( telldata,
486                [ method   = input,
487                  type     = string,
488                  rows     = 1,
489                  cols     = 100,
490                  maxlength = 175
491                ] ).
492
493     proweb_question( askdata,
494                [ method   = input,
495                  type     = string,
496                  rows     = 1,
497                  cols     = 100,
498                  maxlength = 175
499                ] ).
500
501     proweb_question( filename,
502                [ method = input,
503                  type     = string,
504                  rows     = 1,
505                  cols     = 50,
506                  maxlength = 175
507                ] ).
508
```

```
509  add_knowlage(File,Realname) :-
510          retract_knowlage( _, Realname),
511          fcreate(File,File,0,0),
512          input(In), input(File),
513          repeat,
514          read(Knowlage),
515          ( Knowlage == end_of_file ->
516                  true;
517                  real_assert_knowlage(Knowlage,Realname),
518                  fail
519          ),
520          input(In),
521          fclose(File),
522          forall(   retract( ':-'(A) ) , call(A) ) ,!.
523
524  retract_knowlage( Knowlage, Name) :-
525          ( nonvar(Name) ->
526                  ( string(Name) ->
527                   ( find_word( Name, '\',Point3) ->
528                          string_cut(Name, Point3, -1 , Temp2 );
529                          Temp2 = Name
530                   ),
531                   ( find_word( Temp2, `.`,Point4) ->
532                          LenRealName is Point4 - 1,
533                          string_cut(Temp2, 0, LenRealName, Filename_s );
534                          Filename_s = Temp2
535                   ); Filename_s = Name
536                  ),
537                  ( (Filename_s=='*';Filename_s==`*`) ->
538                          var( Filename );
539                          ( atom(Filename_s) -> Filename= Filename_s; atom_string(Filename,
540   Filename_s ) )
541                  );
542                  Filename = `nofile`
```

```
543             ),
544             proweb_posted_reply(userdata, Userdata ),
545             findAns( data(Knowlage, Userdata, Filename) , DataList1 ),
546             ( forall( member( A, DataList1 ),
547                         ( data(Knowlage, Userdata, Filename) = A,
548                           findAns( data(Knowlage, _, _) , DataList2 ),!,
549                           ( (len(DataList2, Len2), Len2 == 1) ->
550                                 retract(Knowlage); true ),
551                           retract(A)
552                         )
553                     );true
554             ),!.
555     retract_knowlage(Knowlage,Name) :- !.
556
557     assert_knowlage(Knowlage,Filename) :-
558             retract_knowlage(Knowlage,`nofile`),
559             real_assert_knowlage(Knowlage,`nofile`).
560     real_assert_knowlage(Knowlage,Filename) :-
561             nonvar( Knowlage ),
562             proweb_posted_reply(userdata, Userdata ),
563             assert( data(Knowlage, Userdata,Filename)),
564             ((findAns( Knowlage, AnsList ),AnsList \= []) -> true;
565                     ( ( functor(Knowlage,':-',2) ->
566                             ( ( retract(Knowlage) ; true),
567                               assert( Knowlage ) );
568                             assert( Knowlage )
569                     ),
570                       ( check_sendmail ; true)
571                     )
572             ), !.
573     check_sendmail :-
574             findAns( mail(_, _) , MailList ),
575             MailList \= [],
576             forall( member( mail(Question, Uname), MailList ),
```

```
577                     ( findAns( Question, AnsList ), len(AnsList, Len), Len \=0 ->
578                             retract( mail(Question, Uname) ),
579                             write_to_file(AnsList, 'Message.txt'),
580                             (print(`Proweb Tell `),to_write(Question) )~>Subject,
581                             user_email( Uname, Address, _ ),
582                             ( var( Address ) ->
583                                     true;
584                                     send_mail(`bundit.graduate.kmitl.ac.th`,`s1014398@ce.kmitl.ac.th`,
585                                             [Address],_,_, Subject ,'Message.txt',_ )
586                             )
587                             ; true
588                     )
589             ).
590
591     findAns( Goal, AnsList ) :-
592             repeat,
593             catch( Error, Goal ),
594             ( Error > 0 -> print('~M~JPredicate Not Defined.~M~J'),!,fail;true),
595             ( Error == 0 ->
596                     assert( temp_answer(Goal) ),
597                     fail;
598               Error == -1 ->
599                     datatolist( temp_answer,AnsList)
600             ),!.
601     datatolist(A,Out) :- retract(A(B)) ->          Out=[B|Tail],     datatolist(A,Tail);Out=[].
602
603     function_tell(Choice) :-
604             (Choice = 'tell by URL' ->
605                     ( proweb_send_form(tell_form),
606                       proweb_returned_answer(telldata, Telldata ),
607                       proweb_post_reply(telldata, Telldata ),
608                       convert_str_to_atom(Telldata,Atom1),
609                       assert_knowlage(Atom1,_),
610                       proweb_send_form( ans_tell_form ) );
```

```
                    ( proweb_send_form(sendfile_form),
                      proweb_returned_answer(filename, FullFilename ),
                      proweb_post_reply(filename, FullFilename ),
                      find_word(FullFilename,`[`,Point1),
                      find_word(FullFilename,`]`,Point2),
                      LenFilename is Point1 - 1,
                      LenTemp1 is Point2 - Point1 - 1,
                      string_cut(FullFilename, 0,LenFilename, FileName_s ),
                      string_cut(FullFilename, Point1,LenTemp1, Temp1 ),
                      ( find_word( Temp1, `\`,Point3) ->
                            string_cut(Temp1, Point3, -1 , Temp2 );
                            Temp2 = Temp1
                      ),
                      ( find_word( Temp2, `.`,Point4) ->
                            LenRealName is Point4 - 1,
                            string_cut(Temp2, 0, LenRealName, RealName_s );
                            RealName_s = Temp2
                      ),
                      atom_string(RealName,RealName_s),
                      atom_string(FileName,FileName_s),
                      add_knowlage(FileName,RealName),
                      proweb_send_form( ans_sendfile_form ) )
        ). /* Choice = 'tell by URL' */

ask_other_server( Askdata, AllResultList ) :-
        retractall( agent_data_reply( _, _, _ ) ),
        convert_str_to_atom(Askdata, Query),
        forall( near_server(Agent),
                ( ask(Agent, Query, ResultList_s, 1, 0, 0) ->
                  convert_str_to_atom(ResultList_s,ResultList),
                  assert( agent_data_reply( Agent, Askdata, ResultList ));true ) ),
        get_ask_other_server_result( Askdata, AllResultList ).

get_ask_other_server_result( Askdata, AllResultList ) :-
```

```
645             retract( agent_data_reply( _, Askdata, ResultList ) ),!,
646             get_ask_other_server_result( Askdata, OtherResultList ),
647             append( ResultList, OtherResultList, AllResultList ).
648     get_ask_other_server_result( _, [] ) :- !.
649
650     add_user_profile(Uname,Email,Pass,Fname,Lname,Nname) :-
651             nonvar( Uname ),
652             ( user_profile(Uname, _, _, _, _ ) ->
653                     !,fail;                         /* Error, This User is Already Exist */
654                     ( ( ask_other_server( user_profile(Uname, _, _, _, _ ), UList ),
655                       len(UList,Len),Len \=0 ) ->
656                             !,fail;                 /* Error, This User is Already Exist */
657                             !,absolute_file_name( 'user_profile.pl' , AbsFileName ),
658                             ( nl, to_write( user_profile(Uname,Email,Fname,Lname,Nname) ), write
659     (`.~M~J`)) ~> 'user_profile.pl',
660                             ( to_write( user_password(Uname,Pass) ), write(`.~M~J`), nl) ~>
661     'user_profile.pl',
662                             fclose(AbsFileName),
663                             ensure_loaded( user_profile )
664                     )
665             ),!.
666
667     make_all_user_list :-
668             ask_other_server( user_profile(_,_,_,_,_), List ),
669             retractall( other_server_user_email( _, _ ) ),
670             forall( member( user_profile(Uname,Email,_,_,_), List ),
671             assert( other_server_user_email( Uname, Email ) ) ),!.
```

ไฟล์ proweb_interface.pl

```
:- ensure_loaded( conf ).
:- ensure_loaded( http_util ).

demo(Agent, Query, Ans ):-
        ( proweb_server(Agent,AgentIP) ->
                ( ask( AgentIP, Query,  AnsList, 0) ->
                        convert_str_to_atom(AnsList,Ans);
                        get_url( AgentIP, Query, Response ),
                        parse_header(Response,Ans)
                );
                ( have_time_out,
                  get_url( Agent, Query, Response ),
                  parse_header(Response,Ans)
                )
        ),!.
forget( Agent, Type, FileKnowlage) :-
/* Type == file -> Remove All Predicate that had been assert previous in This file
  Type == predicate -> Remove that Predicate  */
        ( proweb_server(Agent,AgentIP) ->
                (  Type == file ->
                        run_script( Agent, retract_knowlage(_,FileKnowlage),_ );
                   Type == predicate ->
                        run_script( Agent, retract_knowlage(FileKnowlage,_),_ )
                );
                !,fail
        ),!.

run_script( Agent, Data, ErrorCode ) :-
        ( my_name(_) -> true; assert(my_name(nobody)) ),
        (atom(Data); compound(Data)),!,
        to_write(Data) ~>SendData,
        say_hello(Agent,Msg1),
        my_name(Me),
```

```
34          speak1( Agent, Msg1,          Me, Msg2 ),
35          speak1( Agent, Msg2, `run script`, Msg3 ),
36          speak1( Agent, Msg3,      SendData, Msg4 ),
37          input(Was),
38          input( (Msg4,0) ),
39          repeat,
40          ( eof -> true;
41                  readln(String),
42                  ( find_word(String,`Run script ready`,_) -> ErrorCode= 0;
43                    find_word(String, `Run script fail`,_) -> ErrorCode= -1;
44                    find_word(String,`Run script Error`,_) -> ErrorCode= 1;
45                    fail
46                  )
47          ),nonvar(ErrorCode), !.
48
49  tell(Agent, FileKnowlage) :-
50          ( (file(FileKnowlage),      file(FileKnowlage, -1, Info), Info == 1)->
51                  Method = file;
52                  Method = data
53          ),
54          ( proweb_server(Agent,AgentIP) ->
55                  tell(AgentIP, Method, FileKnowlage);
56                  !,fail
57          ),!.
58
59  tell(Agent, Flag, Knowlage) :- tell(Agent, Flag, Knowlage, _).
60  tell(Agent, Flag, Knowlage, MakeNewConnect) :-
61          ( my_name(_) -> true; assert(my_name(nobody)) ),
62          (atom(Knowlage); compound(Knowlage)),!,
63          to_write(Knowlage) ~>What,
64          ( Flag == file ->
65                  send_file(Agent,Knowlage,HostPath),
66                  Method = `send file`,
67                  write(Knowlage)~>Knowlage_s,
```

```
68                  cat( [HostPath,`[`,Knowlage_s,`]`],DataSend,_);
69                  Flag == data,
70                  Method = `tell by URL`,
71                  DataSend = What
72          ),
73          (MakeNewConnect = 1 ; true), /* Set Default Value */
74          ( MakeNewConnect == 1 ->
75                  say_hello(Agent,Msg1),
76                  my_name(Me),
77                  speak1( Agent, _, Me, Msg2 ),
78                  speak1( Agent, _, `tell`, Msg3);
79                  true
80          ),
81          speak1( Agent, _,  Method, Msg4),
82          speak1( Agent, _, DataSend, _),!.
83
84  ask(Agent, Question, AnsList) :-    /* Set Default Ask Argument */
85          ask(Agent, Question, AnsList, _, _, _).
86  ask(Agent, Question, AnsList, AskOther) :-
87          ask(Agent, Question, AnsList, _, _, AskOther).
88  ask(Agent, Question, AnsList, DoesIknow, AskOther) :-
89          ask(Agent, Question, AnsList, _, DoesIknow, AskOther).
90
91  ask(Agent, Question, AnsList, MakeNewConnect, DoesIknow, AskOther) :-
92          ( my_name(_) -> true; assert(my_name(nobody)) ),
93          (atom(Question); compound(Question)),!,
94          to_write(Question) ~>What,
95          (MakeNewConnect = 1 ; true), /* Set Default Value */
96          (DoesIknow      = 0 ; true),
97          (AskOther       = 1 ; true),
98          ( MakeNewConnect == 1 ->
99                  say_hello(Agent,Msg1),
100                 my_name(Me),
101                 speak1( Agent, _, Me, Msg2 ),
```

```
102                 speak1( Agent, _, `ask`, Msg3);
103                 true
104         ),
105         speak1( Agent, _, What, Msg4),
106         !,
107         ( get_all_ans(Msg4,AnsList) ->
108                 true;
109                 ( AskOther == 1 ->
110                         !,speak1( Agent, _, `Tell another Proweb Server`, Msg5),
111                         ( get_all_ans(Msg5,AnsList) ->
112                                 true;
113                                 ( DoesIknow == 1 ->
114                                         speak1( Agent, _, `Tell`, Msg6),
115                                         get_secreat_answer(Type,Secreat_answer),
116                                         tell(Agent, Type, Secret_answer, 0);
117                                         speak1( Agent, _, `Client doesn't know the solution`, _),
118                                         assert( wait_for_mail ),
119                                         !,fail
120                                 )
121                         );
122                 DoesIknow == 1 ->
123                         speak1( Agent, _, `Client tell Proweb Server`, Msg5),
124                         get_secreat_answer(Type,Secreat_answer),
125                         tell(Agent, Type, Secreat_answer, 0);
126                         !,fail
127                 )
128         ),!.
129 say_hello(Agent,Respond ) :-
130         proweb_dir(Agent,Dir),
131         ( write(Dir),write(`\proweb.exe?lpa=examples`) )~>Hello,
132         demo(Agent,Hello,Respond),!,
133         get_page_id( UCO, Page)<~Respond,
134         retractall( last_connection( Agent, _, _) ),
135         assert( last_connection( Agent, UCO, `[0]`) ).
```

```
136
137
138 speak1(Agent,Last_Msg,DataSend, New_Msg) :-
139         ( nonvar(Last_Msg) ->
140                 get_page_id(UCO,Page) <~ Last_Msg;
141                 last_connection( Agent, UCO, LastPage),
142                 find_word(LastPage,`]`,Point1),
143                 Len is Point1 - 2,
144                 string_cut(LastPage, 1, Len, LastPage_s ),
145                 number_string( LastPageNum, LastPage_s ),
146                 PageNum is LastPageNum + 1,
147                 (write(`[`),write(PageNum),write(`]`))~>Page
148         ),
149         proweb_dir(Agent,Dir),
150         http_encode_form_data([(proweb_data_uco,UCO),(proweb_data_page,Page),
151                                 (`Q0001`,DataSend)],Temp),
152         ( write(Dir),write(`\proweb.exe?`),write(Temp) )~>Send,
153         demo(Agent,Send,New_Msg),!,
154         retractall( last_connection( Agent, _, _) ),
155         assert( last_connection( Agent, UCO, Page) ).
156
157 get_all_ans(Text,Ans) :-
158         input(Was),
159         input( (Text,0) ),
160         ( repeat,
161          ( eof -> true;
162           readln(String),
163           ( find_word(String,`The answer is `,Point1) ->
164                 /* I already get answer */
165                 get_all_ans(String)~>Ans
166           ;find_word(String,`can not find the solution`,_)->
167                 !,input(Was), fail
168           ;fail
169           )
```

```
170             )
171             ), !,
172             input(Was),
173             nonvar(Ans).
174     get_all_ans(FirstData) :-
175             find_word(FirstData,`The answer is `,Point1),
176             len(`The answer is `,Len),
177             StartPoint is Point1 + Len - 1,
178             string_cut(FirstData,StartPoint, -1, Str),
179             write(Str),
180             len(Str, A),
181             B is A -1,
182             string_cut(Str,B,1,Out),
183             ( Out == `.` -> true;
184               repeat,
185               readln(String),
186               string_cut(String, 0, -1, Str2),
187               write(Str2),
188
189           len(Str2, C),
190               D is C -1,
191               string_cut(Str2, D, 1,End),
192               End == `.`
193             ),!.
194
195
196     get_page_id(UCO,Page) :-
197             ( repeat,
198               ( eof -> !,fail; true ),
199               readln(String),
200               get_page_id2(String,`"proweb_data_uco"`,UCO) ), !,
201             ( repeat,
202               ( eof -> !,fail; true ),
203               readln(String2),
```

```
204             get_page_id2(String2,`"proweb_data_page"`,Page) ),!,
205             !,nonvar(UCO),nonvar(Page).
206     get_page_id2(String,Word,Out) :-
207             retractall( temp(get_page_id,_) ),
208             find_word(String,Word,Point1),
209             len(Word,Len1),
210             Point2 is Point1 + Len1,
211             forall( find_word(String,`"`,Point),
212                     ( Point > Point2 -> assertz(temp(get_page_id,Point)) ;true ) ),
213             retract( temp(get_page_id,Point3) ),
214             retract( temp(get_page_id,Point4) ),
215             Len2 is Point4 - Point3 - 1,
216             string_cut(String,Point3,Len2,Out).
217
218     send_file(Who,MyPath,HostPath) :-
219             receive_path( Who, Path),
220             proweb_server(Who,AgentIP),
221             post_message( AgentIP, Path, MyPath, Response ),
222             parse_header(Response,Buffer),
223             input(Was),
224             input( (Buffer,0) ),
225             repeat,
226             (eof -> !,input(Was),fail;true),
227             readln(String),
228             find_word(String,`Filename=`,Point),
229             len(`Filename=`,Len),
230             StartPoint is Point + Len - 1,
231             string_cut(String,StartPoint,-1,HostPath),
232             input(Was),!.
```

ไฟล์ http_util.pl

```
1   :- ensure_loaded(library('tcp/tcp')),ensure_loaded(library('tcp/http')), tcp_reset.
2   :- ensure_loaded( util ).
3
4   get_url( HostName, Path, Output ) :- get_url( HostName, Path, _, Output ).
5   get_url( HostName, Path, Port, Output ) :-
6     nonvar(HostName),
7     (nonvar(Port); Port = 80),
8     (nonvar(Path): Path = `/`),
9     len(Path,Length),
10    (Length == 0 -> P = `/` ; write(Path) ~> P ),
11    http_get( P, _, Command),          /* Make PostMessage */
12    write(HostName) ~> Host,
13    tcp_reset,                         /* Send RequestMessage */
14    catch(Error, tcp_connect( address(Port,Host), ID )),!,
15    Error == 0,
16    tcp_send( ID, Command ),
17    receive_url( ID ) ~> Output,
18    !.
19
20  post_message( HostName, Path, InputFile, Output ):- post_message( HostName, Path, _, InputFile,
21  Output).
22  post_message( HostName, Path, Port, InputFile, Output ):-
23    nonvar(HostName),
24    (nonvar(Port); Port = 80),
25    (nonvar(Path); Path = `\`),
26    len(Path,Length),
27    (Length == 0 -> P = `/` ; write(Path) ~> P ),
28    file(InputFile,-1,Info),Info == 1,                  /* check file exist */
29    file(InputFile,4,DataLen),
30    write_data_from_file(InputFile)~>DataMessage,       /* Read Message Data */
31    http_post(P, DataMessage,`plain/text`,_,Command),/* Make PostMessage */
32    write(HostName) ~> Host,
33    tcp_reset,!,                                        /* Send RequestMessage */
```

```
34      catch(Error, tcp_connect( address(Port,Host), ID )),!,
35      Error == 0,
36      tcp_send( ID, Command ),!,
37      receive_url( ID ) ~> Output,
38      !.
39
40    receive_url( ID ):-
41      repeat,
42      ( tcp_select( term(ID,Data) ) ->
43              write( Data ),
44              fail
45       ; tcp_select( end_of_file( ID ) )
46      ), !.
47    set_time_out(Ms) :-
48              time(0,ThisTime), (_,Now) = ThisTime,
49              retractall( time_out( _ ) ),
50              TimeOut is Now + Ms,
51              assert( time_out(TimeOut) ),!.
52    time_out :-
53              time_out(Time),!,
54              time( 0,ThisTime),(_,Time2) = ThisTime,
55              Time2 > Time.
56    no_time_out :- retractall( timeout_set(_) ).
57    have_time_out :- retract( timeout_set(Time) ) -> have_time_out(Time); have_time_out(2000).
58    have_time_out(Time) :- no_time_out, asserta( timeout_set(Time) ).
59    delay(Time) :-
60              set_time_out(Time),
61              repeat,
62              time_out,!.
63    parse_header(ResponseMessage, String) :- parse_header(ResponseMessage)~>String.
64    parse_header(ResponseMessage) :-
65              input(Was),
66              input( (ResponseMessage,0) ),
67              ( eof -> input(Was), fail; true ),
```

```
68              readln(Respone_header),
69              ( find_word(Respone_header,`200`,_) ;
70               ( input(Was), fail)
71              ),
72              ( eof -> true;
73               repeat,
74               readln(S),
75               (S == `~M~J`; eof),
76               ( eof -> true;
77                repeat,
78                fread(s,1,0,String),
79                ( String == `~J` -> write( `~M~J` );
80                  String == `~M` -> true;
81                  write(String) ),
82                eof
83               )
84              ),
85              input(Was),!.
86
87  readln(String) :- fread(s,0,0,String).
88
89  :- have_time_out.
```

ไฟล์ util.pl

```
1   string_cut(String,Start,Len,Out) :-
2           string_chars( String, String_chars ),
3           string_cut_1( String_chars,Start,String_chars2 ),
4           string_cut_2( String_chars2,Len )~>Out,
5           !.
6   string_cut_1( Output, 0, Output ) :- !.
7   string_cut_1( [Head|Tails],Start, Output ) :- Temp is Start - 1, string_cut_1(Tails,Temp,Output).
8   string_cut_1( _, _, [] ) :- !.
9   string_cut_2( [], _ ) :- !.
10  string_cut_2( _, 0 ) :- !.
11  string_cut_2( [Head|Tails], Len ) :- Len < 0, (Head == 13; Head == 10), !.
12  string_cut_2( [Head|Tails], Len ) :- Temp is Len -1, fwrite(b,0,0,[Head]), string_cut_2(Tails,Temp).
13
14  find_word(String,Word,Point) :-
15          string_chars( String, String_chars ),
16          string_chars( Word, Search_chars ),
17          mapword(String_chars,Search_chars,Point2,Flag),
18          length(Search_chars,Points),
19          Point is Point2 - Points.
20  mapword([Head|Tail],[Headw|Tailw],Point,Flag) :-
21          ( ( Head == Headw , mapword(Tail,Tailw,Point2,yes) )
22           ;( nonvar(Flag), Flag == yes, !, fail )
23           ;( !,mapword(Tail,[Headw|Tailw],Point2,no) )
24          ),
25          Point is Point2 + 1.
26  mapword(_,[],1,yes) :- !.
27
28  convert_str_to_atom(Data,Atom) :-
29          ( write(Data),write(`~M~J.~M~J`) )~> String,
30          input(In),input((String,0)),
31          catch( Error, read(Atom) ),
32          input(In),
33          !,Error == 0,Atom \= end_of_file.
```

```
34
35  to_write(Data) :-  /* Invert Function of convert_str_to_atom */
36          type(Data,T),
37          ( ( T == 3 ) ->     write_atom(Data);
38            ( T == 4 ) ->     write_string(Data);
39            ( T == 6 ) ->     write(`[`), write_list(Data),write(`]`);
40            ( T == 7 ) ->     functor(Data,A,Count),
41                              write(`'`),          write(A),          write(`'(`),
42                              write_tuple(Data,Count,0),write(`)`);
43            write(Data)
44          ),!.
45  write_atom(Atom) :-
46          write(""),
47          atom_chars(Atom,Chars),
48          write_atom2(Chars),
49          write("").
50  write_atom2([]) :- !.
51  write_atom2([Head|Tails]) :-
52          atom_chars(C, [Head] ),
53          ( C == '"' ->
54                  write("""");
55                  write( C )
56          ),
57          write_atom2(Tails).
58
59  write_string(String) :-
60          write(````),
61          string_chars(String,Chars),
62          write_string2(Chars),
63          write(````).
64  write_string2([]) :- !.
65  write_string2([Head|Tails]) :-
66          string_chars(C, [Head] ),
67          ( C == ```` ->
```

```
68                      write(`````);
69                      write( C )
70              ),
71              write_string2(Tails).
72
73      write_list(List) :-
74              List = [Head|Tails],
75              to_write(Head),
76              ( Tails \= [] ->
77                      write(','),
78                      write_list(Tails);
79                      true
80              ),!.
81      write_tuple(Tuple, Argv, Last) :-
82              ArgAt is Last + 1,
83              arg(ArgAt, Tuple, Data),
84              to_write(Data),
85              ( Argv == ArgAt ->
86                      true;
87                      write(','),
88                      write_tuple(Tuple, Argv, ArgAt)
89              ),!.
90      write_to_file(Data,File) :-
91              file( File, -1, Info ),
92              (Info == 1 -> del(File); true),
93              fcreate(File,File,-1,0),
94              output(Was),
95              output(File),
96              write(Data),
97              output(Was),
98              fclose(File).
99      write_data_from_file(File) :-
100             fcreate( File, File, 0, 0),
101             input(Was),
```

```
102             input(File),inpos(0),
103             repeat,
104             ( not eof ->
105                     fread(s,0,0,Data),
106                     write(Data),
107                     fail;
108                     true
109             ),!,
110             input(Was),
111             fclose(File).
112
113     solve(true)  :- !.
114     solve((A,B)) :- !,solve(A), solve(B).
115     solve((A;B)) :- !,( solve(A); solve(B) ).
116     solve(A) :- repeat, catch( Error, clause( A, B ) ),
117                     ( Error == 0 ->
118                             solve(B);
119                       Error == -1 ->
120                             !,fail;
121                             !,call(A)
122                     ).
123     left_functor(Predicate,Functor) :-
124             functor(Predicate, Temp2, Arity),
125                     ( (Temp2 == ':-',Arity==2) ->
126                             arg(1,Predicate,Temp);
127                             Temp = Temp2 ),
128         ( atom(Temp) ->
129                             Functor = Temp;
130                             left_functor(Temp,Functor)  ),!.
131
132     not_empty(A) :-
133             (var(A) -> !, fail; true),
134             ( string(A) ->
135                     string_chars(A,Lists),!,
```

```
136                     all_not_space(Lists)
137                     ;true
138             ),!.
139     all_not_space( [] ) :- !, fail.
140     all_not_space(CharList) :-
141             [H|T] = CharList,
142             ( check_is_space( H ) ->
143                     all_not_space( T );
144               true
145             ).
146
147     check_is_space( Char ) :-
148             string_chars(S,[Char]),
149             ( S == ` `; S == `~I`;
150               S == `~M`; S == `~J` ).
```

ไฟล์ conf.pl

```
1   %define( server ).                    /* Change Here for Select Browser or Server */
2   define( browser ).
3
4   :- dynamic( [my_name/1] ).
5
6   :- define( server ), assert( my_name(Name) )./* Set Server Name */
7
8   :- define( server ), ensure_loaded( user_profile ).
9
10  /* All Server In System */
11
12  proweb_server(big,'161.246.5.120').
13  proweb_server(neung,'161.246.5.103').
14
15  proweb_server( ID, IP ) :-
16          nonvar(ID),!,
```

```
17          proweb_server( _, ID ),
18          IP =ID.
19  proweb_dir( Agent, '\proweb').                    /* Default proweb directory */
20
21  receive_path( Agent, '\cgi-bin\receive.cgi').   /* Default CGI for Send File */
22
25          proweb_server(AgentName,Agent),
26          not my_name(AgentName).
27
28  user_email( Uname, Email, Pass ) :-          /* Get UserName of Current User */
29          (my_name(Uname); true),                    /* Or Specific User                */
30          ( user_profile( Uname, Email, _, _, _) ->      /* Get Email of that User   */
31            (user_password( Uname, Pass ); true); /* If program in server then cann't get password */
32            define(server), other_server_user_email( Uname, Email )
33          ),!.
34  exist_user(Uname) :- user_profile( Uname, _, _, _, _).
```

ไฟล์ script.pl

```
1   checkmail_all :-
2           find_serv_user_pass(Server,Username,Pass),
3           checkemail(Server,Username,Pass,`Subject`,[``]).
4
5   checkmail_bysubject(Condition) :-
6           find_serv_user_pass(Server,Username,Pass),
7           checkemail(Server,Username,Pass,`Subject`,Condition).
8
9   checkmail_bydate(Condition) :-
10          find_serv_user_pass(Server,Username,Pass),
11          checkemail(Server,Username,Pass,`Date`,Condition).
12
```

```
13  checkemail(Server,Username,Pass,TypeCheck,Condition) :-
14          Window = (user_dialog,8000),
15          MailFile = 'mail.txt',
16          mail_check( Server, _, Username, Pass, NoOfMsgs),
17          (NoOfMsgs == 0 ->
18           ( write_edit_box(Window,`ไม่มีข้อมูลใหม่`) );
19      (mail_get_if( Server, _, Username,Pass ,0, 'mail.txt',
20  (TypeCheck,contains,any_of,Condition),Total,Got ),
21      (Got == 0 ->
22              ( write_edit_box(Window,`ไม่มีข้อมูลใหม่`) );
23          (mail_parse_all(MailFile,Msgs,Parse),
24          del(MailFile),
25              ( write( ` There are `),
26              write( Got ),
27              write( ` mail for you. `)
28              ) ~> Message,
29          message_box(ok, Message, _ ),
30          forall( member( (From,To,Cc,Bcc,Subj,Date,Header,Body,Attach) , Msgs ),
31              ( write_edit_box(Window,`Subject:~I`),
32                  write_edit_box(Window,Subj),
33                  write_edit_box(Window,`~M~JFrom:~I`),
34                  write_edit_box(Window,From),
35                  write_edit_box(Window,`~M~JDate:~I`),
36                  write_edit_box(Window,Date),
37                  write_edit_box(Window,`~M~J`),
38                  write_data_from_file(Body)~>MailText,
39                  write_edit_box(Window,MailText),
40
41  write_edit_box(Window,`~M~J~I═══════════════════════════ END `),
42                  write_edit_box(Window,`MESSAGE
43  ═══════════════════════~M~J~M~J`)
44              )
45          ) /*forall*/
46          ) /*mail_parse_all*/
```

```
47          ) /*Got == 0*/
48        ) /*mail_get_if*/
49     ).
50  sendmail(To,Subj,Body)  :- sendmail(To,_,_,Subj,Body,_).
51  sendmail(To,CC,BCC,Subj,Body,Attach) :-
52          atom(Body),
53          mail_set(log, on),
54          my_name(ID),
55          user_email(ID,Address,Password),
56          mail_send(`bundit.graduate.kmitl.ac.th`, _,(Address, To, CC, BCC, Subj, Body, Attach), Sent ).
57  sendmail(To,CC,BCC,Subj,Body,Attach) :-
58          string(Body),
59          write_to_file(Body, 'body.txt'),
60          mail_set(log, on),
61          my_name(ID),
62          user_email(ID,Address,Password),
63          mail_send(`bundit.graduate.kmitl.ac.th`, _,(Address, To, CC, BCC, Subj, 'body.txt', Attach),
64  Sent ).
```

บรรณานุกรม

[1] William F. Clocksin and C.S. Mellish (1987) : "*Programming in prolog Third, Revised and Extended Edition.*", Springer-Verlag, Berlin Heidelberg, 1987.

[2] Stephen A. Thomas : "*HTTP Essentials Protocols for Secure, Scaleable Web Sites*", John Wiley & Sons, Inc, 2001.

[3] Seng Wai Loke : "*Adding Logic Programming Behaviour to the World Wide Web*", A Thesis Submitted in total fulfilment of the requirements of the degree of Doctor of Philosophy Department of Computer Science School of Electrical Engineering and Computer Science The University of Melbourne AUSTRALIA, 1998.

[4] Z. Meltem ISMIK : "*LPA WIN-PROLOG 4.0 TCP/IP Libraries*", Logic Programming Associates Ltd, 1999.

[5] Rebecca Shalfield : "*LPA WIN-PROLOG 4.0 ProWeb Server User Guide*", Logic Programming Associates Ltd, 1998-2000.

[6] Andrew S. Tanenbaum : "*Computer Networks*", Pearson Education Indochina Ltd, 1996.

[7] ผศ.ดร. สัลยุทธ์ สว่างวรรณ : "*เครือข่ายคอมพิวเตอร์*", บริษัท ซีเอ็ดยูเคชั่นจำกัด (มหาชน), 2542.