การพัฒนาระบบงานบน WWW

Application Development on WWW

โดย

นายธวัชชัย อนุพงศ์อนันต์

นายวิทยา วิญญุนาวรรณ

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปีการศึกษา 2539

การพัฒนาระบบงานบน WWW

โดย

นายธวัชชัย อนุพงศ์อนันต์ 36014187
นายวิทยา วิญญนาวรรณ 36014406

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบงานบน WWW

ผู้จัดทำ

1. นายธวัชชัย อนุพงศ์อนันต์ รหัสนักศึกษา 36014187
2. นายวิทยา วิญญุนาวรรณ รหัสนักศึกษา 36014406

..... อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

การพัฒนาระบบงานบน WWW

นายธวัชชัย อนุพงศ์อนันต์
นายวิทยา วิญญุนาวรรณ
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

โครงงานนี้เป็นการนำเสนอวิธีการสร้างระบบงานให้ผู้ใช้สามารถเรียกใช้งานได้โดยผ่านบราวเซอร์ของบริการเวิลด์ไวด์เว็บ ซึ่งเป็นบริการของระบบเครือข่ายอินเตอร์เน็ตที่เป็นที่นิยมใช้งานกันอย่างกว้างขวางมากที่สุดในปัจจุบัน ระบบงานสามารถถูกสร้างขึ้นโดยประกอบด้วย 2 ส่วนสำคัญคือส่วนของ CGI (Common Gateway Interface) ซึ่งเป็นจะถูกประมวลผลที่ฝั่งเว็บเซิร์ฟเวอร์ ในส่วนของ CGI สามารถสร้างขึ้นโดยใช้ภาษาโปรแกรมมิ่งหลายภาษา เช่น วิชวลเบสิก หรือ Delphi ส่วนที่สอง คือส่วนของ JAVA ซึ่งจะถูกประมวลผลที่บราวเซอร์บนฝั่งไคลเอ็นท์

ระบบงานที่สร้างขึ้นสามารถทำงานกับฐานข้อมูลชนิดต่าง ๆ เช่นไมโครซอฟท์แอ็กเซส หรือ SQL เซิร์ฟเวอร์ เป็นต้น ระบบงานสามารถเชื่อมต่อกับฐานข้อมูลเหล่านี้โดยผ่าน ODBC ผู้ใช้สามารถเรียกใช้งานระบบงาน และเข้าทำงานกับฐานข้อมูลได้โดยไม่จำกัดชนิดของเครื่องคอมพิวเตอร์ที่ผู้ใช้ใช้งานอยู่ ไม่จำกัดชนิดของระบบปฏิบัติการ เพียงแต่เครื่องคอมพิวเตอร์ที่ผู้ใช้งานใช้อยู่ได้เชื่อมต่อกับระบบเครือข่ายอินเตอร์เน็ต หรืออินทรานเน็ตก็เพียงพอแล้ว

การที่ผู้ใช้งานสามารถเข้ามาใช้งานฐานข้อมูลได้ จึงต้องมีส่วนของระบบรักษาความปลอดภัยของข้อมูลด้วยซึ่งสามารถทำได้หลายวิธี ไม่ว่าจะเป็นการเข้ารหัสข้อมูลแบบคีย์เดียว การเข้ารหัสข้อมูลแบบสองคีย์ เป็นต้น

Application Development on WWW

Thawatchai Anuponganant
Vitaya Vinyunavan
Dr.Voravat Limpoka Advisor
1996

ABSTRACT

This project presents how to create an application that allows users to call and work through the Web browser of the WWW service in the INTERNET. The application can be create with the 2 important parts that is CGI (Common Gateway Interface) that will be executed at the Web Server. The CGI part can be written with several programming language such as Visual Basic , Delphi etc. The second part of the application is JAVA that will be executed at Web browser in the client site.

The application can work with several type of databases such as MS-Access database file , database on the SQL Server etc. The application can connect to these databases with ODBC. User can call the application for running and working with databases without the limited of the computer platform , or the operating system. The computer should be connect to the INTERNET or INTRANET.

The user can call the application and working with the database , so there must be a security system to protect user's data like cryptography include single key cryptography and public key cryptography method.

สารบัญ

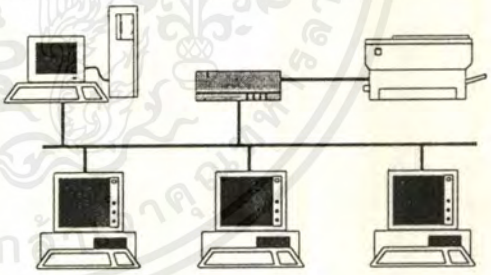
สารบัญรูปภาพ

สารบัญตาราง

บทนำ

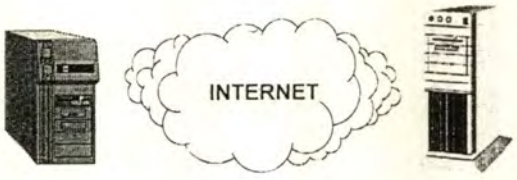
ในอดีตเมื่อเริ่มมีการใช้งานเครื่องคอมพิวเตอร์ การใช้งานส่วนใหญ่จะเป็นลักษณะของเครื่องเทอร์มินอล (Terminal) ซึ่งต่อเข้ากับเครื่องคอมพิวเตอร์ขนาดใหญ่เช่นเครื่องเมนเฟรม (Mainframe) การประมวลผลข้อมูลทั้งหมดจะถูกทำที่เครื่องเมนเฟรมนี้เท่านั้น ต่อมาเมื่อมีการพัฒนาทางด้านเทคโนโลยีสารกึ่งตัวนำ ซึ่งทำให้เครื่องคอมพิวเตอร์มีขนาดเล็กลง แต่มีความสามารถในการทำงานและการประมวลผลต่าง ๆ มากขึ้น ก็ได้มีการนำเครื่องคอมพิวเตอร์มาใช้งานในลักษณะของเครื่องคอมพิวเตอร์ส่วนบุคคล ผู้ใช้งานแต่ละคนก็สามารถทำงานและประมวลผลข้อมูลของตนเองได้ ในสมัยต่อมาได้เริ่มมีการนำเครื่องคอมพิวเตอร์หลาย ๆ เครื่องเข้ามาเชื่อมต่อกันเพื่อเป็นการแบ่งปันทรัพยากรที่มีอยู่ให้สามารถใช้งานร่วมกัน เกิดความ คุ้มค่ามากขึ้น กลายเป็นระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น (LAN : Local Area Network) และระบบเครือข่ายคอมพิวเตอร์ขนาดใหญ่ขึ้น (WAN : Wide Area Network) จนกระทั่งในปัจจุบันนี้ ระบบเครือข่ายคอมพิวเตอร์ได้ถูกพัฒนาขึ้นให้สามารถติดต่อสื่อสารกันได้ทั่วโลก กลายเป็นระบบเครือข่ายอินเตอร์เน็ต (INTERNET) นั่นเอง

Mainframe



LAN , WAN

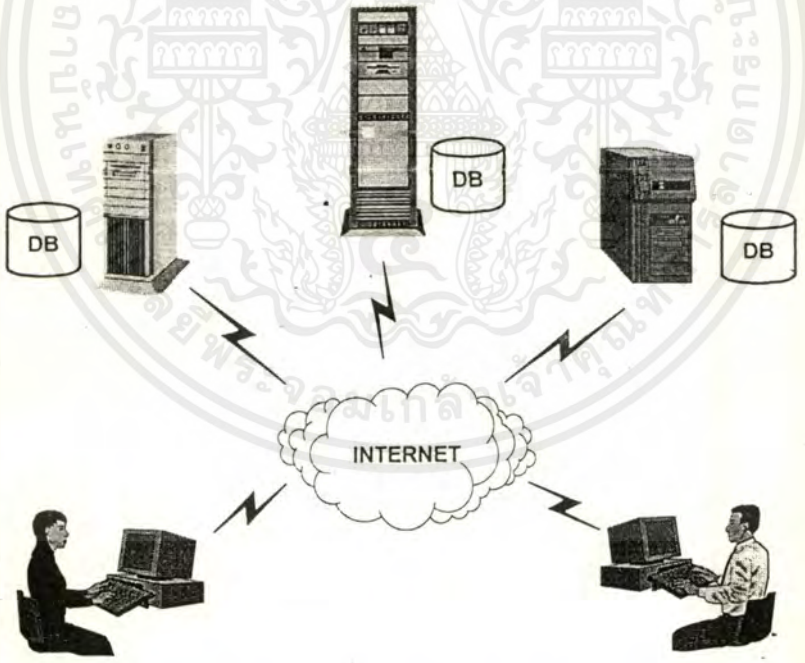
Personal Computer
(PC)



รูปที่ A - 1 ระบบคอมพิวเตอร์ในยุคต่าง ๆ

การใช้งานระบบเครือข่ายอินเตอร์เน็ต นี้ช่วยให้สามารถเชื่อมต่อเครื่องคอมพิวเตอร์ที่อยู่ในสถานที่ต่าง ๆ ซึ่งห่างกันไกลได้ทั่วโลก ทำให้สามารถใช้งานและแบ่งปันฐานข้อมูลต่าง ๆ ซึ่งช่วยให้เกิดการพัฒนาในด้านต่าง ๆ อย่างรวดเร็ว เช่นการค้นคว้าข้อมูลในการทำวิจัยของนักวิชาการ นักวิทยาศาสตร์ การค้นคว้าข้อมูลสำหรับการทำธุรกิจ การค้นคว้าข้อมูลของ นักศึกษาในการเรียนการศึกษา หรือแม้กระทั่งการค้นหาข้อมูลต่าง ๆ เพื่อความบันเทิงของ ผู้ใช้งานทั่วไป ดังนั้นจึงทำให้ระบบเครือข่ายอินเตอร์เน็ต จึงมีผู้ใช้งานมากมาย และมี แนวโน้มที่จะมีผู้ใช้งานเพิ่มขึ้นอย่างต่อเนื่อง

การใช้งานระบบเครือข่ายอินเตอร์เน็ต ที่เป็นที่นิยมมากในปัจจุบันนี้ได้แก่การใช้งานบริการ World Wide Web (WWW) ซึ่งช่วยให้ผู้ใช้งานสามารถค้นหาข้อมูลต่าง ๆ ได้ง่าย และรวดเร็ว ข้อมูลที่ได้จากบริการ WWW นี้มีทั้งข้อมูลที่เป็นข้อมูลตัวอักษร รูปภาพ ภาพเคลื่อนไหว หรือแม้กระทั้งข้อมูลเสียงก็มี นอกจากนี้รูปแบบการใช้งานมีการติดต่อกับผู้ใช้แบบกราฟฟิก (GUI : Graphic User Interface) ซึ่งทำให้ใช้งานได้ง่าย



รูปที่ A - 2 การเข้าใช้งานระบบฐานข้อมูลผ่านเครือข่ายอินเตอร์เน็ต

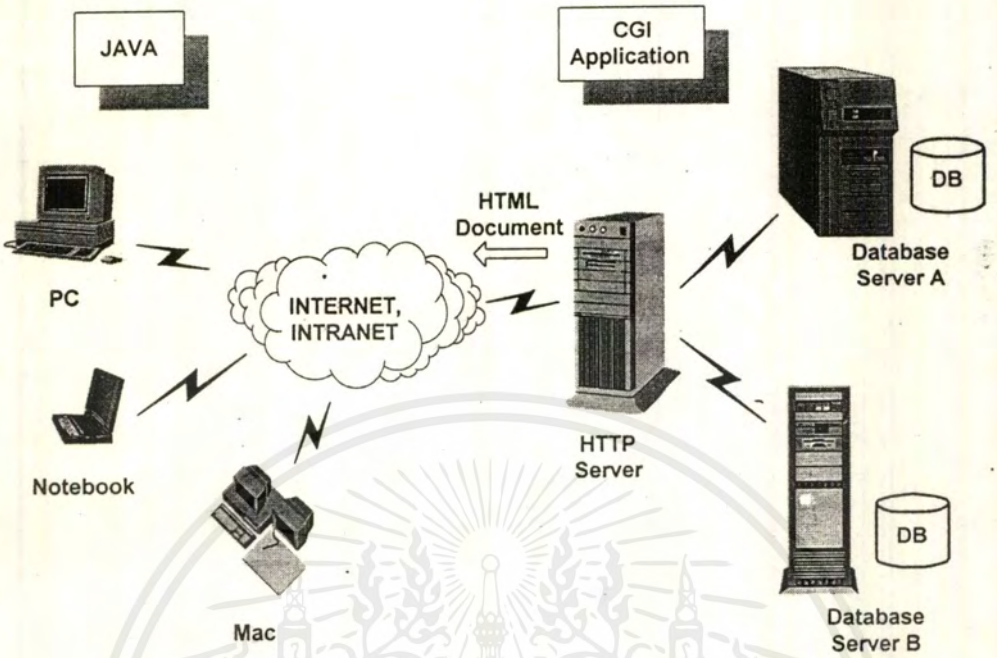
การใช้งาน WWW นี้นอกจากจะเรียกข้อมูลที่ต้องการขึ้นมาดูได้แล้ว ยังสามารถที่จะทำงานกับฐานข้อมูลที่มีอยู่ที่ฝั่งเซิร์ฟเวอร์ (Server) ได้ด้วย โดยสามารถเพิ่มเติม แก้ไขข้อมูลในฐานข้อมูลก็ได้ หรือจะสั่งให้แอพพลิเคชั่น (Application) ที่อยู่ที่ฝั่งเซิร์ฟเวอร์ทำการประมวลผล หรือทำงานอื่นใดก็ได้ตามที่ต้องการ เช่นระบบการสั่งซื้อสินค้าผ่านทาง อินเตอร์

เน็ต ที่ต้องมีการทำงานกับฐานข้อมูล มีการตัดยอดสินค้าคงคลัง มีการโอนเงินในบัญชีธนาคาร และระบบรักษาความปลอดภัยของข้อมูล เป็นต้น

เนื้อหาในรายงานฉบับนี้จะอธิบายถึงหลักการเบื้องต้น วิธีการสร้าง และข้อมูลต่าง ๆ ที่เกี่ยวข้อง ที่จำเป็นต้องใช้ในการพัฒนาระบบงาน ให้สามารถทำงานผ่านทางเครือข่ายอินเตอร์เน็ตโดยการใช้บริการของ WWW เช่น ในบทที่ 1 และ 2 จะกล่าวถึงเรื่องของ CGI (Common Gateway Interface) ซึ่งเป็นหลักการสร้างแอพพลิเคชั่นให้สามารถเรียกใช้งานผ่านทางบริการ WWW ได้ ในบทที่ 3 และ 4 จะกล่าวถึงเรื่องของ JAVA ซึ่งเป็นภาษาที่สามารถทำงานได้บนบราวเซอร์ (Browser) ได้ ในบทที่ 5 จะกล่าวถึงหลักการต่าง ๆ เกี่ยวกับระบบรักษาความปลอดภัยของข้อมูล การเข้ารหัสข้อมูลแบบต่าง ๆ รวมทั้งอัลกอริธึมสำหรับการเข้ารหัสที่เป็นที่นิยมใช้งานในปัจจุบัน ในบทสุดท้าย คือบทที่ 6 จะเป็นหลักการทำงานของตัวอย่างระบบงานที่สร้างขึ้น

จากเนื้อหาทั้งหมดที่กล่าวถึงในรายงานฉบับนี้จะช่วยให้สามารถสร้างระบบงานบนเครือข่าย WWW เพื่อทำงานกับฐานข้อมูลต่าง ๆ ซึ่งระบบงานซึ่งทำงานกับฐานข้อมูลโดยผ่านเครือข่าย WWW นี้ มีข้อดีตรงที่ความง่ายต่อการจัดการระบบการติดต่อกับผู้ใช้ (User Interface) เนื่องจากผู้ใช้งานสามารถใช้งานระบบงานนี้ได้โดยใช้บราวเซอร์ซึ่งมีมาตรฐานแล้ว นอกจากนี้เนื่องจากสามารถใช้งานระบบงานผ่านบราวเซอร์ได้ทันที ดังนั้นระบบงานจึงเป็นอิสระไม่ขึ้นกับเครื่องคอมพิวเตอร์ของผู้ใช้ ไม่ขึ้นกับระบบปฏิบัติการที่ผู้ใช้ใช้งานอยู่ เช่น Windows, Unix, Macintosh, OS/2 เป็นต้น

ระบบงานที่จะได้นำเสนอในรายงานฉบับนี้ประกอบด้วยส่วนต่าง ๆ และมีการติดต่อ สื่อสารกันดังรูป A-3



รูปที่ A - 3 ภาพรวมและส่วนประกอบต่าง ๆ ของระบบงาน

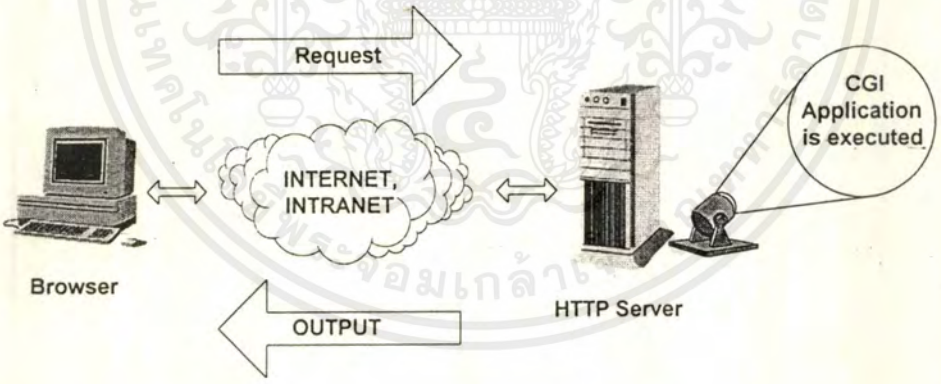
บทที่ 1

CGI

Common Gateway Interface

ความหมาย

CGI คือ รูปแบบของการประมวลผล Application บนฝั่ง Server โดยผู้ใช้สามารถเรียกใช้ได้จากฝั่ง Client ได้ จากนั้นผลลัพธ์ที่ได้จากการประมวลผลจะถูกส่งกลับไปให้กับผู้ใช้ที่ฝั่ง Client โดยการสื่อสารข้อมูลระหว่างฝั่ง Client และ Server นี้จะอาศัย Protocol ที่เรียกว่า HTTP (HyperText Transfer Protocol) ซึ่ง Protocol ชนิดนี้เป็น Protocol ที่ถูกคิดค้นขึ้นมาเมื่อปี ค.ศ. 1990 เพื่อใช้สำหรับการสื่อสารในเครือข่าย WWW (World Wide Web)



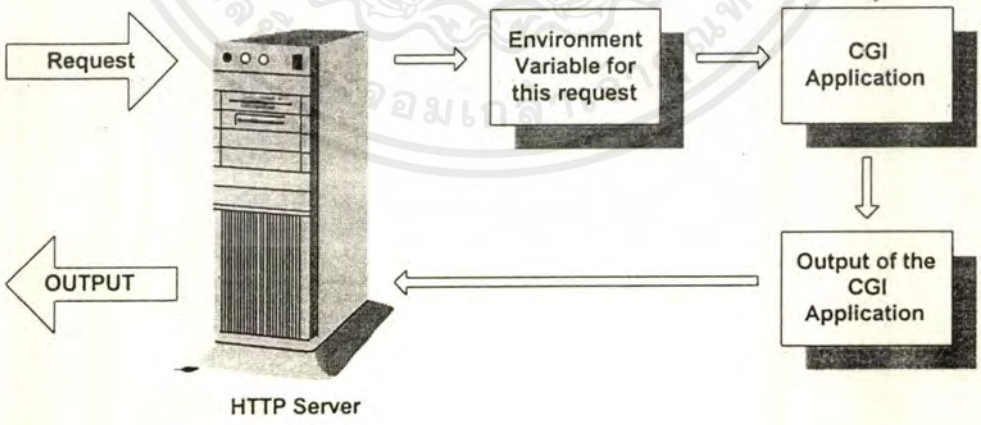
รูปที่ 1 - 1 ภาพโดยรวมในการทำงานของ CGI

CGI สามารถเขียนขึ้นได้โดยใช้ภาษาโปรแกรมมิ่ง (Programming Language) หลายชนิด เช่น Visual Basic , C , C++ , หรือ UNIX Shell Script เป็นต้น เราสามารถแบ่งภาษาโปรแกรมมิ่งเหล่านี้ได้เป็น 2 ประเภท คือ

1. ภาษาโปรแกรมมิ่งที่ไม่ต้อง Compile
2. ภาษาโปรแกรมมิ่งที่ต้อง Compile

ข้อแตกต่างของภาษาทั้ง 2 ชนิดนี้ก็คือ ภาษาที่ไม่ต้อง Compile จะสามารถทำการปรับปรุงเปลี่ยนแปลงแก้ไขได้ง่าย เมื่อทำการแก้ไขแล้วก็ไม่ต้อง Compile ใหม่ แต่การทำงานจะช้ากว่าภาษาที่ต้อง Compile ดังนั้นการเลือกใช้ภาษาใดนั้น ก็ขึ้นอยู่กับความสะดวกของผู้ใช้งานและความเหมาะสมกับลักษณะงานที่ต้องการ ในการใช้งาน CGI โปรแกรมโดยทั่วไป นิยมเก็บโปรแกรมที่ Compile แล้ว หรือ Script ไว้ที่ Directory ที่มีชื่อว่า cgi-bin ส่วนในกรณีที่ใช้ภาษาที่ต้อง Compile ก็นิยมเก็บ Source-code ไว้ที่ Directory ที่มีชื่อว่า cgi-src

ในการทำงานของ CGI จะมีการสื่อสารระหว่างฝั่ง Client และฝั่ง Server โดยจะมีตัวแปรสำหรับเก็บข้อมูลการติดต่อต่าง ๆ เรียกว่า Environment Variable ซึ่งตัวแปรแต่ละตัวนี้ก็จะมีหน้าที่สำหรับเก็บข้อมูลต่าง ๆ แตกต่างกันไป เช่น HTTP_USER_AGENT , SERVER_SOFTWARE , SERVER_NAME , GATEWAY_INTERFACE , SERVER_PROTOCOL , SERVER_PORT , REMOTE_HOST , REMOTE_ADDR , REMOTE_USER , REQUEST_METHOD , HTTP_ACCEPT , PATH_INFO , PATH_TRANSLATED , SCRIPT_NAME , QUERY_STRING , AUTH_TYPE , CONTENT_TYPE , CONTENT_LENGTH เป็นต้น ซึ่งค่าจากตัวแปรต่าง ๆ เหล่านี้ผู้ใช้งานสามารถนำไปใช้สำหรับ Application ต่าง ๆ ได้ตามต้องการ



รูปที่ 1 - 2 การใช้งาน Environment Variable

การใช้งาน CGI ที่เป็นที่นิยมในปัจจุบันมีอยู่ 3 รูปแบบ คือ

1. Imagemap คือการรับค่าพิกัด x,y บนภาพที่ผู้ใช้กด Mouse เพื่อส่งต่อให้กับ Applicationเพื่อทำงานตามที่ได้กำหนดไว้โดยจะมี File ที่เรียกว่า Map อยู่ที่ Server เพื่อรับค่าพิกัด x,y แล้วพิจารณาเรียกใช้ Resource ต่าง ๆ ตามที่ได้กำหนดไว้ แต่ในปัจจุบันสามารถทำ Imagemap ได้โดยใช้ Tag ของ HTML ซึ่งสามารถรับค่าพิกัด x,y และพิจารณาเรียกใช้ Resource ที่กำหนดได้ที่ฝั่ง Client ไม่ต้องมีการประมวลผลที่ Server
2. CGI Application คือการสร้าง Application เพื่อทำงานอย่างหนึ่งอย่างใดแล้วส่งผลลัพธ์ในรูปแบบต่าง ๆ ให้กับ Client ที่ได้ร้องขอมา เช่นการสร้างภาพเพื่อแสดงจำนวนผู้เข้าใช้งาน หรือที่เรียกว่า Counter เป็นต้น
3. FORM คือการสร้างแบบฟอร์มขึ้นมาโดยใช้ HTML จากนั้นเมื่อผู้ใช้ได้กรอกข้อมูลแล้วส่งมากลับมาที่ HTTP-Server แล้ว Application ที่ฝั่ง Server ก็จะรับข้อมูลดังกล่าวเพื่อทำการประมวลผลต่อไป

การส่งข้อมูลของ CGI มี 2 วิธีที่เป็นที่นิยมในปัจจุบัน คือ

1. การส่งโดยวิธี GET (GET Method)
2. การส่งโดยวิธี POST (POST Method)

ในการส่งข้อมูลจาก Client มายัง Server ในแต่ละครั้ง โปรโตคอล HTTP จะระบุวิธีการที่ใช้ส่งในตัวแปรที่มีชื่อว่า REQUEST_METHOD ซึ่งนอกจาก Method GET และ Method POST นี้แล้วยังอาจมี Method อื่น ๆ สำหรับการใช้งานเฉพาะอย่างใดอย่างหนึ่งซึ่งจะไม่พูดถึงในที่นี้ สำหรับ Method GET และ Method POST นั้นแต่ละวิธีนั้นมีข้อดี ข้อเสียและวิธีการส่งข้อมูลแตกต่างกันไป ดังนี้

การส่งข้อมูลโดยวิธี GET

ในการเรียกใช้ URL (Uniform Resource Locator) โดยปกติก็เป็นการใช้วิธี GET อยู่แล้ว การส่งข้อมูลของวิธี GET จะส่งข้อมูลรวมไปกับ URL โดยจะตามหลัง URL คั่นด้วยเครื่องหมายปรัศนี (?) จากนั้นจะตามด้วยชุดของข้อมูลเป็นคู่ ๆ แต่ละคู่ประกอบด้วยชื่อคีย์ (Key name) และ ค่าของคีย์ (Key Value) ข้อมูลแต่ละคู่จะถูกคั่นแยกจากกันด้วยเครื่องหมาย และ (&) ข้อมูลต่าง ๆ ในส่วนของ Key Value จะถูกทำการแปลงเล็กน้อยเรียกว่าการทำ URL Encode จากนั้นข้อมูลจะถูกส่งไปอยู่ในตัวแปรชื่อ QUERY_STRING ซึ่งการแปลงข้อมูลต่าง ๆ ทำได้ดังนี้

1. เครื่องหมายเว้นวรรคจะถูกเปลี่ยนเป็นเครื่องหมายบวก (+)
2. ตัวอักษรพิเศษต่าง ๆ จะถูกเปลี่ยนให้เป็นเครื่องหมายเปอร์เซ็นต์ตามด้วยเลขฐานสิบหกซึ่งเป็นรหัสแอสกี (ASCII) ของอักษรพิเศษนั้น ๆ เช่น เครื่องหมาย ! จะถูกแปลงเป็น %23 เป็นต้น

ข้อดีของการส่งข้อมูลโดยวิธี GET คือ สามารถสร้าง Application ให้รับข้อมูลได้ง่าย ทั้งนี้สามารถรับข้อมูลได้จากตัวแปร QUERY_STRING ได้โดยตรง

ตัวอย่างการส่งข้อมูลด้วยวิธี GET ที่เห็นได้ชัดเจน คือ การส่งข้อมูลของ INTERNET Search Engine เช่น ALTAVISTA , YAHOO , LYCOS เป็นต้น เมื่อผู้ใช้งาน INTERNET Search Engine ใส่คำที่ต้องการค้นหาและเริ่มทำการค้นหาข้อมูล เมื่อสังเกตค่า URL ที่ส่งกลับไปให้ Server จะเห็นข้อมูลที่ถูกส่งกลับไปให้ Server ด้วย ตัวอย่างเช่น การใช้ ALTAVISTA ค้นหาข้อมูลคำว่า "Application" จะได้ URL ดังนี้

```
http://www.altavista.digital.com/cgi-bin/query?pg=q&what=web&fmt=.&q=Application
```

เนื่องจากการส่งข้อมูลด้วยวิธี GET จะทำการส่งข้อมูลรวมไปกับ URL และข้อมูลจะถูกนำไปเก็บในตัวแปรที่มีชื่อว่า QUERY_STRING ดังนั้นข้อมูลที่สามารถส่งได้จึงมีขนาดจำกัด โดยมีขนาดได้ไม่เกิน 256 ตัวอักษร ดังนั้นสำหรับแบบฟอร์มขนาดใหญ่ที่ต้องส่งข้อมูลจำนวนมากกลับไปให้ Server จึงไม่สามารถใช้การส่งข้อมูลด้วยวิธี GET นี้ได้

การส่งข้อมูลโดยวิธี POST

การส่งข้อมูลโดยวิธี POST นี้สามารถแก้ปัญหาข้อมูลมีขนาดใหญ่ไม่สามารถส่งด้วยวิธี GET ได้ โดยการส่งข้อมูลโดยวิธี POST นี้จะทำการส่งจำนวนตัวอักษรของข้อมูลที่จะส่งไปให้กับ Server โดยใช้ตัวแปรที่มีชื่อว่า CONTENT_LENGTH หลังจากนั้น Application ที่ฝั่ง Server ก็จะสามารถรับข้อมูลได้จาก STDIN (Standard Input) ที่ HTTP Server กำหนดขึ้นมาใหม่เป็นการชั่วคราว โดยรับข้อมูลในปริมาณเท่ากับที่ได้กำหนดไว้ในตัวแปรชื่อ CONTENT_LENGTH นั่นเอง ดังนั้นการส่งข้อมูลโดยวิธี POST นี้จึงสามารถส่งข้อมูลเป็นจำนวนได้ไม่จำกัด

ในการเขียน Application สำหรับรับข้อมูลซึ่งส่งมาจากฝั่ง Client โดยวิธี GET และ POST จึงมีความแตกต่างกันซึ่งในที่นี้จะขอยกตัวอย่างวิธีการเขียนโปรแกรมสำหรับการรับข้อมูลด้วยภาษา C บน Unix

ตัวอย่างการรับข้อมูลที่ส่งมาด้วยวิธี GET

```
/*
* Process input form in GET Method
*/

#include <stdio.h>
#include <stdlib.h>

/* this is the structure we use for the CGI variables */
struct {
      char name[128];
      char val[128];
}elements[16];

/* temporary storage for the environment variable */
char *cp;
char *empty = "<empty>";

/* macro for displaying environment variables */
#define safenv(a) ((cp = getenv(a)) ? cp : empty)

void splitword(out,in,stop)
char *out;
char *in;
char stop;
{
int i,j;

        for (i =0;in[i] && (in[i] != stop); i++)
                out[i] = in[i];

        out[i] = '\0'; /* terminate it */
        if (in[i])
                i++;

        for (j = 0;in[j]; ) /* shift the rest of the in */
                in[j++] = in[i++];
}

char x2c(x)
char *x;
{
register char c;

/* note: (x & 0xdf) makes x upper case */
        c = (x[0] >= 'A' ? ((x[0] & 0xdf) - 'A') + 10 : (x[0] - '0'));
```

```
        c *= 16;
        c += (x[1] >= 'A' ? ((x[1] & 0xdf) - 'A') + 10 : (x[1] - '0'));
        return(c);
}

/* this function goes through the URL char-by-char and converts all the "escaped"
(hex-encoded) sequences to characters  this version also converts pluses to spaces. */

void unescape_url(url)
char *url;
{
register int i,j;

        for (i = 0,j = 0;url[j];++i, ++j)
        {
                if ((url[i] = url[j]) == '%')
                {
                        url[i] = x2c(&url[j+1]);
                        j += 2;
                }
                else if (url[i] == '+')
                        url[i] = ' ';
        }
        url[i] = '\0'; /* terminate it at the new length */
}

main()
{
char *qs; /* qs is for the query string */
int i;

        /* send the MIME header first! */
        printf("Content-type: text/html\n\n");

        printf("<html>\n");
        printf("<h1> These are informations for this submission </h1>\n");
        printf("<hr>");

        /* send the CGI variables */
        printf("HTTP_USER_AGENT = %s <br>\n", safenv("HTTP_USER_AGENT"));
        printf("REQUEST_MOTHOD = %s <br>\n", safenv("REQUEST_METHOD"));
        printf("SCRIPT_NAME = %s <br>\n", safenv("SCRIPT_NAME"));
        printf("QUERY_STRING = %s <br>\n", safenv("QUERY_STRING"));
        printf("REMOTE_HOST = %s <br>\n", safenv("REMOTE_HOST"));
        printf("REMOTE_ADDR = %s <br>\n", safenv("REMOTE_ADDR"));
        printf("CONTENT_TYPE = %s <br>\n", safenv("CONTENT_TYPE"));
        printf("CONTENT_LENGTH = %s <br>\n", safenv("CONTENT_LENGHT"));
        printf("<hr><p>\n");

        /* assign the query string to qs;or fail if it's empty */
        if ((qs = getenv("QUERY_STRING")) == NULL)
        {
                printf("No query information to decode. \n");
                exit(1);
        }
```

```
        /* split out each of the parameters from the query string */
        for (i = 0;qs[0] != '\0';i++)
        {
                /* first divide by '&' for each parameter */
                splitword(elements[i].val, qs, '&');
                /* convert the string for hex characters and pluses */
                unescape_url(elements[i].val);
                /* now split out the name and value */
                splitword(elements[i].name, elements[i].val, '=');
        }

        printf("Variables:\n\n");
        for (i =0;elements[i].name[0];i++)
        printf("%s=%s", elements[i].name, elements[i].val);
}
```

ตัวอย่างการรับข้อมูลที่ส่งมาด้วยวิธี POST

```
/*
* Process input form in POST Method
*/

#include <stdio.h>
#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif
#include <string.h>

#define MAX_ENTRIES 10000

typedef struct {
   char *name;
   char *val;
} entry;

char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

main(int argc, char *argv[])
{
entry entries[MAX_ENTRIES];
register int x,m=0;
int cl;

        printf("Content-type: text/html%c%c",10,10);

        if(strcmp(getenv("REQUEST_METHOD"),"POST"))
        {
```

```
                printf("This script should be referenced with a METHOD of POST.\n");
                printf("If you don't understand this, see this ");
                printf("<A
HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
                exit(1);
        }
        if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
        {
                printf("This script can only be used to decode form results. \n");
                exit(1);
        }
        cl = atoi(getenv("CONTENT_LENGTH"));

        for(x=0;cl && (!feof(stdin));x++)
        {
                m=x;
                entries[x].val = fmakeword(stdin,'&',&cl);
                plustospace(entries[x].val);
                unescape_url(entries[x].val);
                entries[x].name = makeword(entries[x].val,'=');
        }

        printf("<H1>Query Results</H1>");
        printf("You submitted the following name/value pairs:<p>%c",10);
        printf("<ul>%c",10);

        for(x=0; x <= m; x++)
        {
                printf("<li> <code>%s =
%s</code>%c",entries[x].name,entries[x].val,10);
                printf("</ul>%c",10);
        }
```

หลังจากทราบวิธีการเขียนโปรแกรมเพื่อทำการรับข้อมูลที่ส่งมาจากฝั่ง Client ด้วยวิธี GET หรือ POST แล้ว ก็สามารถที่จะสร้าง Application ง่าย ๆ ที่รับข้อมูลจากแบบฟอร์มแล้วนำข้อมูลมาประมวลผลขึ้นได้แล้ว

การทำงานของ CGI

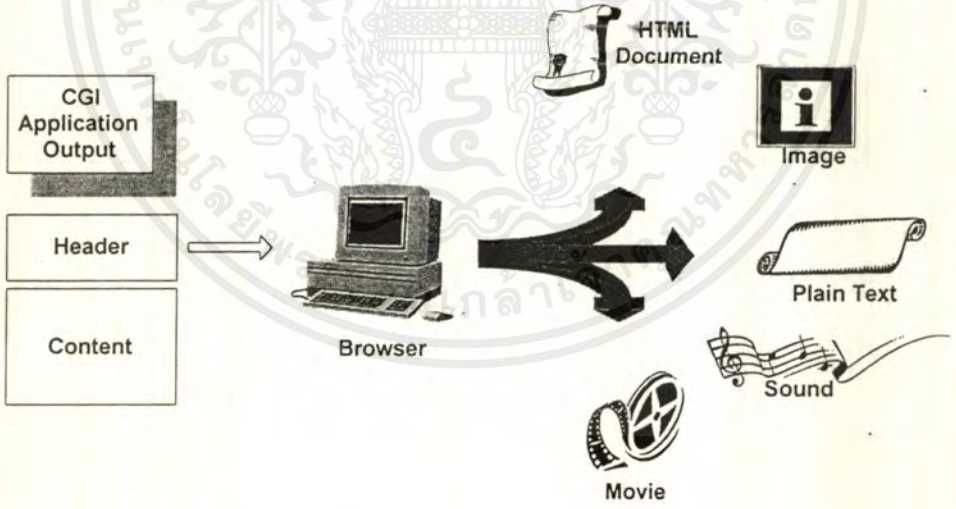
เมื่อ Client มีการร้องขอ URL มาที่ CGI โปรแกรมที่ฝั่ง Server แล้ว Server จะทำการประมวลผลโปรแกรมดังกล่าวแบบ Realtime โดยที่ CGI โปรแกรมสามารถรับข้อมูล ต่าง ๆ จากฝั่ง Client ได้จากตัวแปรที่เรียกว่า Environment Variable ดังที่ได้กล่าวไว้แล้วในตอนต้น จากนั้นก็จะส่งผลลัพธ์ที่ได้กลับไปให้ฝั่ง Client โดยผลลัพท์ที่ส่งกลับไปนี้มีได้หลายรูปแบบ ไม่ว่าจะเป็นเอกสาร HTML (HTML Document) , รูปภาพ , ตัวอักษร (TEXT) , เสียง ,

ภาพเคลื่อนไหว เป็นต้น การที่ฝั่ง Client จะแสดงผลผลลัพธ์ประเภทต่าง ๆ ได้อย่างถูกต้องนั้น CGI โปรแกรมจึงต้องมีการกำหนดชนิดของผลลัพธ์ที่ได้ไว้ในส่วนที่เรียกว่า ส่วนหัว (Header) ซึ่งข้อมูลในส่วนหัวนี้จะมีรูปแบบดังนี้

1. เป็นตัวอักษรในรหัส ASCII
2. เก็บข้อมูลบรรทัดละ 1 ข้อมูล
3. ส่วนหัวจะถูกแยกออกจากส่วนผลลัพธ์โดยใช้บรรทัดว่างคั่น 1 บรรทัด

ข้อมูลในส่วนหัวมีหลายข้อมูลเช่น

1. Content-Type : ใช้สำหรับระบุชนิดของผลลัพธ์ที่ได้ด้วย MIME Type (Multipurpose Internet Mail Extension) ทั้งนี้เพื่อเป็นการบอกให้ส่วนของ Browser รับทราบชนิดของข้อมูลว่าข้อมูลที่ได้รับจาก Server นั้นเป็นข้อมูลประเภทใด
2. Location : ใช้ระบุที่อยู่ของผลลัพธ์ที่จะอ้างถึง โดยใช้ URL
3. Status : ใช้สำหรับบอกให้ Server ส่งข้อมูลสถานะไปให้ Client ซึ่งตามปกติในส่วนนี้ตัว Server จะเป็นตัวจัดการสร้างขึ้นให้เองโดยอัตโนมัติ เช่นในกรณีที่หา Resource ที่อ้างถึงไม่พบ จะส่งรหัส 404 หรือผู้เรียกใช้ไม่มีสิทธิจะส่งรหัส 403 เป็นต้น



รูปที่ 1 - 3 Browser สามารถใช้ CGI Header ในการแยกประเภทของข้อมูล

ตัวอย่างการใส่ส่วนหัว (Header)

ในกรณีที่ต้องการให้ CGI โปรแกรมส่งเอกสาร HTML กลับไปให้ฝั่ง Client จะต้องสร้าง CGI โปรแกรมให้ได้ผลลัพธ์หลังจากการทำงานดังนี้

```
Content-type: text/html

<HTML><HEAD>
<TITLE>output of HTML from CGI script</TITLE>
</HEAD><BODY>
<H1>Sample output</H1>
What do you think of <STRONG>this?</STRONG>
</BODY></HTML>
```

ในกรณีที่ต้องการอ้างถึงไฟล์อื่น ๆ สามารถที่จะระบุได้ในส่วนของ Location : ตามด้วยบรรทัดว่าง ๆ 1 บรรทัด ดังตัวอย่าง

```
Location: /dir1/dir2/myfile.html

```

ตัวอย่าง CGI โปรแกรม

สรุปแล้ว CGI โปรแกรม ก็คือโปรแกรม หรือสคริปต์ ที่เขียนขึ้นเพื่อให้ Server ทำการประมวลผลเมื่อมีการร้องขอจาก Client โดยมีการส่งข้อมูลโดยใช้โปรโตคอล HTTP นั้นเอง การเขียน CGI โปรแกรมสามารถเขียนได้หลายภาษาดังตัวอย่างต่อไปนี้

การเขียน CGI โปรแกรมโดยใช้เชลล์สคริปต์

ตัวอย่างที่ 1 เป็นแสดงวันที่ปัจจุบันโดยใช้คำสั่ง DATE เขียนโดยใช้สคริปต์

```
#!/bin/sh

DATE=/bin/date

echo Content-type: text/plain
echo

if [ -x $DATE ]; then
        $DATE
else
        echo Cannot find date command on this system.
fi
```

ตัวอย่างที่ 2 เป็นการแสดงรายชื่อของผู้ใช้งานระบบในขณะนั้น เขียนขึ้นโดยเชลล์สคริปต์

```
#!/bin/sh

FINGER=/usr/ucb/finger

echo Content-type: text/html
echo

if [ -x $FINGER ]; then
        if [ $# = 0 ]; then
                cat << EOM
<HTML><HEAD><TITLE>Finger Gateway</TITLE></HEAD><BODY>
<H1>Finger Gateway</H1>

<ISINDEX>

This is a gateway to "finger". Type a user@host combination in your browser's
search dialog.<P>
EOM
        else
                echo \<PRE\>
                $FINGER "$*"
                echo \</PRE\>
        fi
else
        echo Cannot find finger on this system.
fi
cat << EOM
</BODY></HTML>
EOM
```

ตัวอย่างที่ 3 เป็นการสร้าง CGI โปรแกรม โดยใช้เชลล์สคริปต์ ตัวอย่างนี้เป็นตัวอย่างการสร้างเอกสาร HTML เพื่อให้สามารถอ้างต่อไปยัง Homepage ของผู้ใช้แต่ละคนได้ โดยมีการกำหนดให้หาไฟล์ชื่อ index.html ในไดเร็กเทอรี่ชื่อ www ใน home-directory ของแต่ละคน

```
#!/bin/csh
# Search for personal homepage
# By Atsawin
# Start Jan. 3, 1995
# Last update Mar. 13, 1996

cat << EOM
<HEAD>
<TITLE>Student 4D</TITLE>
</HEAD>
<BODY>
<H1>Student 4D</H1>
List of Student in class 4D.
<HR>
<UL>
```

```
EOM

cd /users/std4d
foreach user (*)
  if (-d $user) then
    echo -n \<LI\>
    /bin/grep \^$user\: /etc/passwd|awk -f /usr/local/etc/httpd/util/realname|awk -f
/usr/local/etc/httpd/util/realname1
    if (-e $user/www/index.html) then
      echo \<A HREF=\"http://diamond.ce.kmitl.ac.th/~$user\"\>\($user\)\</A\>
    else
      echo \($user\)
    endif
  endif
end

cat << EOM
</UL>
<HR>
Send problem, comment or suggestion to <B>webmaster@ce.kmitl.ac.th</B>.
Thank you very much.
</BODY>
</HTML>
EOM
```

ตัวอย่างที่ 4 เป็นการสร้าง CGI Program เพื่อแสดงค่าของ Environment Variable ต่าง ๆ ที่เกิดขึ้นระหว่างการติดต่อสื่อสารของ Client กับ Server

```
#!/bin/sh

echo HTTP/1.0 200 OK
echo Content-type: text/plain
echo Server: $SERVER_SOFTWARE
echo

echo CGI/1.0 test script report:
echo

echo argc is $#. argv is "$*".
echo

echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = $PATH_INFO
echo PATH_TRANSLATED = $PATH_TRANSLATED
echo SCRIPT_NAME = $SCRIPT_NAME
echo QUERY_STRING = $QUERY_STRING
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
```

```
echo REMOTE_USER = $REMOTE_USER
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

การเขียน CGI โปรแกรมโดยใช้ภาษา C

การสร้าง CGI โปรแกรมโดยใช้ภาษาโปรแกรมมิ่ง เช่นภาษา C ดังที่ได้กล่าวไว้แล้วในตอนต้น คือ จะต้องเก็บซอร์สโค้ด ไว้ในไดเร็กเทอรี่ชื่อ /cgi-src และเก็บโปรแกรมที่ได้จากการคอมไพล์ ไว้ที่ไดเร็กเทอรี่ชื่อ /cgi-bin ตัวอย่างนี้ เป็นการใช้ภาษา C บนยูนิกซ์สร้าง CGI โปรแกรมเพื่อรับข้อมูลที่ได้จากการส่งด้วยวิธี GET

```
#include <stdio.h>
#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif
#include <string.h>

typedef struct {
   char name[128];
   char val[128];
} entry;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


main(int argc, char *argv[]) {
   entry entries[10000];
   register int x,m=0;
   char *cl;

   printf("Content-type: text/html%c%c",10,10);

   if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
      printf("This script should be referenced with a METHOD of GET.\n");
      printf("If you don't understand this, see this ");
      printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
      exit(1);
   }

   cl = getenv("QUERY_STRING");
   if(cl == NULL) {
      printf("No query information to decode.\n");
      exit(1);
```

```
    }
    for(x=0;cl[0] != '\0';x++) {
        m=x;
        getword(entries[x].val,cl,'&');
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
    }

    printf("<H1>Query Results</H1>");
    printf("You submitted the following name/value pairs:<p>%c",10);
    printf("<ul>%c",10);

    for(x=0; x <= m; x++)
        printf("<li> <code>%s = %s</code>%c",entries[x].name,
            entries[x].val,10);
    printf("</ul>%c",10);
}
```

การเขียน CGI โปรแกรมโดยใช้ PERL

ตัวอย่างที่ 1 เป็นการแสดงค่าต่าง ๆ ของตัวแปรในการติดต่อสื่อสารระหว่าง Client และ Server

```
#!/usr/bin/perl
# Test for CGI
# This CGI will display some Environment Variable
#
# Vitaya Vinyunavan  17 January 1997
#=====================================
#
printf STDOUT "Content-type: text/html \n\n";
printf STDOUT "<h1>Environment Variable by PERL1.PL.CGI</h1><hr> \n";
printf STDOUT "SERVER_SOFTWARE = " . $ENV{SERVER_SOFTWARE} . "<br>\n";
printf STDOUT "SERVER_NAME = " . $ENV{SERVER_NAME} . "<br>\n";
printf STDOUT "GATEWAY_INTERFACE = " . $ENV{GATEWAY_INTERFACE} . "<br>\n";
printf STDOUT "SERVER_PROTOCOL = " . $ENV{SERVER_PROTOCOL} . "<br>\n";
printf STDOUT "SERVER_PORT = " . $ENV{SERVER_PORT} . "<br>\n";
printf STDOUT "SERVER_ADMIN = " . $ENV{SERVER_ADMIN} . "<br>\n";
printf STDOUT "CGI_VERSION = " . $ENV{CGI_VERSION} . "<br>\n";
printf STDOUT "REQUEST_PROTOCOL = " . $ENV{REQUEST_PROTOCOL} . "<br>\n";
printf STDOUT "REQUEST_KEEP_ALIVE = " . $ENV{REQUEST_KEEP_ALIVE} . "<br>\n";
printf STDOUT "REQUEST_METHOD = " . $ENV{REQUEST_METHOD} . "<br>\n";
printf STDOUT "HTTP_ACCEPT = " . $ENV{HTTP_ACCEPT} . "<br>\n";
printf STDOUT "PATH_INFO = " . $ENV{PATH_INFO} . "<br>\n";
printf STDOUT "PATH_TRANSLATED = " . $ENV{PATH_TRANSLATED} . "<br>\n";
printf STDOUT "SCRIPT_NAME = " . $ENV{SCRIPT_NAME} . "<br>\n";
printf STDOUT "QUERY_STRING = " . $ENV{QUERY_STRING} . "<br>\n";
printf STDOUT "REMOTE_HOST = " . $ENV{REMOTE_HOST} . "<br>\n";
printf STDOUT "REMOTE_ADDR = " . $ENV{REMOTE_ADDR} . "<br>\n";
```

```
printf STDOUT "AUTH_TYPE = " . $ENV{AUTH_TYPE} . "<br>\n";
printf STDOUT "REMOTE_USER = " . $ENV{REMOTE_USER} . "<br>\n";
printf STDOUT "REMOTE_IDENT = " . $ENV{REMOTE_IDENT} . "<br>\n";
printf STDOUT "CONTENT_TYPE = " . $ENV{CONTENT_TYPE} . "<br>\n";
printf STDOUT "CONTENT_LENGTH = " . $ENV{CONTENT_LENGTH} . "<br>\n";
printf STDOUT "REFERER = " . $ENV{REFERER} . "<br>\n";
printf STDOUT "FROM = " . $ENV{FROM} . "<br>\n";
printf STDOUT "USER_AGENT = " . $ENV{USER_AGENT} . "<br>\n";
```

ตัวอย่างที่ 2 เป็นการแสดงจำนวนครั้งที่เอกสาร HTML Document ที่กำหนดถูกเรียกดูโดยแสดงตัวเลขด้วยภาพกราฟิกในรูปแบบ xbitmap ขาวดำ ขนาด 8*16 pixel โดยจะมีการกำหนดไฟล์สำหรับเก็บจำนวนครั้งไว้ แล้วโปรแกรมนี้จะอ่านข้อมูลจากไฟล์ดังกล่าว

```
#!/usr/bin/perl
# $Id: counter,v 1.1 1994/04/18 14:48:00 root Exp $
#
# counter - increment a counter for WWW
#
# neat things left to do:
#       - create a new file if it does not exist
#       - maintain "backup" of #'s in case of spontaneous reset
#
#
# 6/12/95 - *finally* gotten around to making this accepct an argument
#           as the counter file to increment
#                   m.l.nelson@larc.nasa.gov
#
# 5/11/94 - removed inetd dependency; now uses cgi-bin Michael Nelson
#                   m.l.nelson@larc.nasa.gov
#
# Written 3/8/94 by Dan Rich (drich@corp.sgi.com)
#    Based on C code written by Frans van Hoesel (hoesel@chem.rug.nl)
#
# $Log: counter,v $
# Revision 1.1  1994/04/18  14:48:00  root
# Initial revision
#
#

sub usage {
   print STDERR "usage: $0 [-i]\n";
   exit 1;
}

if ( $ENV{QUERY_STRING} eq "" )
{
        $counterfile = "/users/std4d/s6014406/www/index.html.cnt";
}
else
{
        $counterfile = "/users/std4d/s6014406/www/" . $ENV{QUERY_STRING};
}
```

```
require('getopts.pl') || die "can\'t do getopts.pl: $@";
require('sys/file.ph') || die "can\'t do sys/file.ph: $@";

if ( ! &Getopts('i') ) {
   &usage();
}
$inv = $opt_i;

# bitmap for each digit

@invdigits = (0xff,0xff,0xff,0xc3,0x99,0x99,0x99,0x99,0x99,0x99,0x99,0x99,0xc3,0xff,0xff,0xff,
        0xff,0xff,0xff,0xcf,0xc7,0xcf,0xcf,0xcf,0xcf,0xcf,0xcf,0xcf,0xcf,0xff,0xff,0xff,
        0xff,0xff,0xff,0xc3,0x99,0x9f,0x9f,0xcf,0xe7,0xf3,0xf9,0xf9,0x81,0xff,0xff,0xff,
        0xff,0xff,0xff,0xc3,0x99,0x9f,0x9f,0xc7,0x9f,0x9f,0x9f,0x99,0xc3,0xff,0xff,0xff,
        0xff,0xff,0xff,0xcf,0xcf,0xc7,0xc7,0xcb,0xcb,0xcd,0x81,0xcf,0x87,0xff,0xff,0xff,
        0xff,0xff,0xff,0x81,0xf9,0xf9,0xf9,0xc1,0x9f,0x9f,0x9f,0x99,0xc3,0xff,0xff,0xff,
        0xff,0xff,0xff,0xc7,0xf3,0xf9,0xf9,0xc1,0x99,0x99,0x99,0x99,0xc3,0xff,0xff,0xff,
        0xff,0xff,0xff,0x81,0x99,0x9f,0x9f,0xcf,0xcf,0xe7,0xe7,0xf3,0xf3,0xff,0xff,0xff,
        0xff,0xff,0xff,0xc3,0x99,0x99,0x99,0xc3,0x99,0x99,0x99,0x99,0xc3,0xff,0xff,0xff,
        0xff,0xff,0xff,0xc3,0x99,0x99,0x99,0x99,0x83,0x9f,0x9f,0xcf,0xe3,0xff,0xff,0xff
        );
@digits = (0x00,0x00,0x00,0x3c,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x30,0x38,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x00,0x00,0x00,
        0x00,0x00,0x00,0x3c,0x66,0x60,0x60,0x30,0x18,0x0c,0x06,0x06,0x7e,0x00,0x00,0x00,
        0x00,0x00,0x00,0x3c,0x66,0x60,0x60,0x38,0x60,0x60,0x60,0x66,0x3c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x30,0x30,0x38,0x38,0x34,0x34,0x32,0x7e,0x30,0x78,0x00,0x00,0x00,
        0x00,0x00,0x00,0x7e,0x06,0x06,0x06,0x3e,0x60,0x60,0x60,0x66,0x3c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x38,0x0c,0x06,0x06,0x3e,0x66,0x66,0x66,0x66,0x3c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x7e,0x66,0x60,0x60,0x30,0x30,0x18,0x18,0x0c,0x0c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x3c,0x66,0x66,0x66,0x3c,0x66,0x66,0x66,0x66,0x3c,0x00,0x00,0x00,
        0x00,0x00,0x00,0x3c,0x66,0x66,0x66,0x66,0x7c,0x60,0x60,0x30,0x1c,0x00,0x00,0x00
        );

$hex = "0123456789abcdef";

# Read and increment the counter file
while (-f "${counterfile}.lock") { sleep 1;}
open(LOCK,">${counterfile}.lock");
open(COUNTERFILE,"<$counterfile") || die "can\'t open $counterfile: $!\n";
#flock(COUNTERFILE,&LOCK_EX) || die "can\'t lock $counterfile: $!\n";
$text = <COUNTERFILE>;
close(COUNTERFILE);
$text++;
$len = length($text) > 7 ? length($text) : 7;
open(COUNTERFILE,">$counterfile") || die "can\'t open $counterfile: $!\n";
printf COUNTERFILE "%0${len}u\n",$text;
$text = sprintf("%0${len}u",$text);
#flock(COUNTERFILE,&LOCK_UN);
close(COUNTERFILE);
unlink("${counterfile}.lock");

# Generate an X11 bitmap on STDOUT
print &PrintHeader; #mln
printf STDOUT "#define count_width %d\n#define count_height 16\n", $len*8;
printf STDOUT "static char count_bits[] = {\n";
```

```
for ($y=0; $y < 16; $y++) {
   for ($x=0; $x < $len; $x++) {
      $d = substr($text,$x,1) - '0';
      print STDOUT '0x';
      if ($inv) {          # $inv = 1 for inverted text
         printf STDOUT "%1x",($invdigits[($d * 16) + $y] >> 4) & 0xf;
         printf STDOUT "%1x",$invdigits[($d * 16) + $y] & 0xf;
      } else {
         printf STDOUT "%1x",($digits[($d * 16) + $y] >> 4) & 0xf;
         printf STDOUT "%1x",$digits[($d * 16) + $y] & 0xf;
      }
      if ($x < $len-1) {
         print STDOUT ',';
      }
   }
   if ($y==15) {
      print STDOUT '};';
   } else {
      print STDOUT ',';
   }
   print STDOUT "\n";
}

# PrintHeader
# Returns the magic line which tells WWW that we're an HTML document

sub PrintHeader {
  return "Content-type: image/x-xbitmap\n\n";
}
```

การเขียน CGI โปรแกรมโดยใช้ TCL

ตัวอย่างนี้เป็นการสร้างเอกสาร HTML เพื่อแสดงค่าตัวแปร (Environment Variable) ต่าง ๆ

```
#!/usr/local/bin/tclsh
# tcl-cgi.tcl
# robert.bagwill@nist.gov, no warranty, no rights reserved
# print out command line args, stdin, and environment variables
#
# some fixes by dl@hplyot.obspm.fr - v1.1 - Apr 11 1995
#
set envvars {SERVER_SOFTWARE SERVER_NAME GATEWAY_INTERFACE
SERVER_PROTOCOL SERVER_PORT REQUEST_METHOD PATH_INFO
PATH_TRANSLATED SCRIPT_NAME QUERY_STRING REMOTE_HOST REMOTE_ADDR
REMOTE_USER AUTH_TYPE CONTENT_TYPE CONTENT_LENGTH HTTP_ACCEPT
HTTP_REFERER HTTP_USER_AGENT}

puts "Content-type: text/html\n"
puts "<HTML>"
```

```
puts "<HEAD>"
puts "<TITLE>CGI/1.1 TCL script report:</TITLE>"
puts "</HEAD>"

puts "<BODY>"
puts "<H1>Command Line Arguments</H1>"
puts "argc is $argc, argv is $argv."
puts ""

puts "<H1>Message</H1>"
puts "<PRE>"
if {[string compare $env(REQUEST_METHOD) "POST"]==0} {
set message [split [read stdin $env(CONTENT_LENGTH)] &]
} else {
set message [split $env(QUERY_STRING) &]
}
foreach pair $message {
        set name [lindex [split $pair =] 0]
        set val [lindex [split $pair =] 1]
        regsub -all {\+} $val { } val
        # kludge to unescape some chars
        regsub -all {\%0A} $val \n\t val
        regsub -all {\%2C} $val {,} val
        regsub -all {\%27} $val {'} val
        puts "$name\t= $val"
}
puts "</PRE>"

puts "<H1>Environment Variables</H1>"
puts "<DL>"
foreach var $envvars {
        if {[info exists env($var)]} {
                puts "<DT>$var<DD>$env($var)"
                }
        }
}
puts "</DL>"
puts "</BODY>"
puts "</HTML>"
######################
# end of tcl-cgi.tcl
######################
```

Windows CGI

การทำงานของ CGI ที่ได้กล่าวมาแล้วนั้นเป็นการทำงานทั่ว ๆ ไป เช่น บน UNIX แต่สำหรับ Windows CGI นั้นเป็นส่วนที่เพิ่มเติมขึ้นมาสำหรับการสร้าง CGI บนระบบปฏิบัติการ

Windows ซึ่งการทำงานของ Windows CGI จะแตกต่างจาก CGI เดิมเล็กน้อยซึ่งสามารถอธิบายได้ดังนี้

ในระบบปฏิบัติการ Windows นั้นไม่มีตัว Interpreter เหมือนใน Unix ดังนั้นจึงไม่สามารถเขียน CGI เป็น Script ได้ การเขียน CGI บน Windows จึงต้องเป็น Executable Program คือต้องมีการคอมไพล์ให้เป็นไฟล์ .EXE ก่อน อย่างไรก็ตาม เราก็สามารถที่จะเรียกใช้ CGI ที่เป็น Batch File ซึ่งสามารถเอ็กซีคิวต์ได้โดยอาศัย Shell ของ DOS เช่น 4DOS , NDOS ก็ได้

การทำงานของ Windows CGI เมื่อมีการเรียกใช้ Executable Program ใด ๆ จะมีการสร้างไฟล์ชั่วคราว (Temporary File) ขึ้นมา 3 ไฟล์คือ

1. CGI-DATA-FILE
2. CONTENT-FILE
3. OUTPUT-FILE

CGI-DATA-FILE

เป็นไฟล์ชั่วคราวที่ใช้เก็บข้อมูลต่าง ๆ ที่ใช้ในการติดต่อสื่อสารระหว่างฝั่ง Client และ Server ภายในไฟล์นี้จะประกอบด้วยส่วนต่าง ๆ ดังนี้

- CGI เป็นส่วนที่เก็บค่า Environment Variable ต่าง ๆ
- Accept แสดงชนิดของข้อมูลที่ Client สามารถรับได้
- System เก็บ Full Path/name ของ CONTENT-FILE และ OUTPUT-FILE
- Extra Headers เก็บข้อมูลพิเศษที่มาพร้อมกับ URL เช่นใน Method GET จะมีข้อมูลในส่วนนี้
- Form Literal ข้อมูลจาก FORM ที่ส่งมาโดย Method POST จะถูก Decode และเก็บเป็นคู่ ๆ ของ Key=Value ไว้ที่ส่วนนี้
- Form External เก็บชื่อของ CONTENT-FILE และความยาวของข้อมูล
- Form Huge เก็บตำแหน่งของข้อมูลใน CONTENT-FILE และความยาว

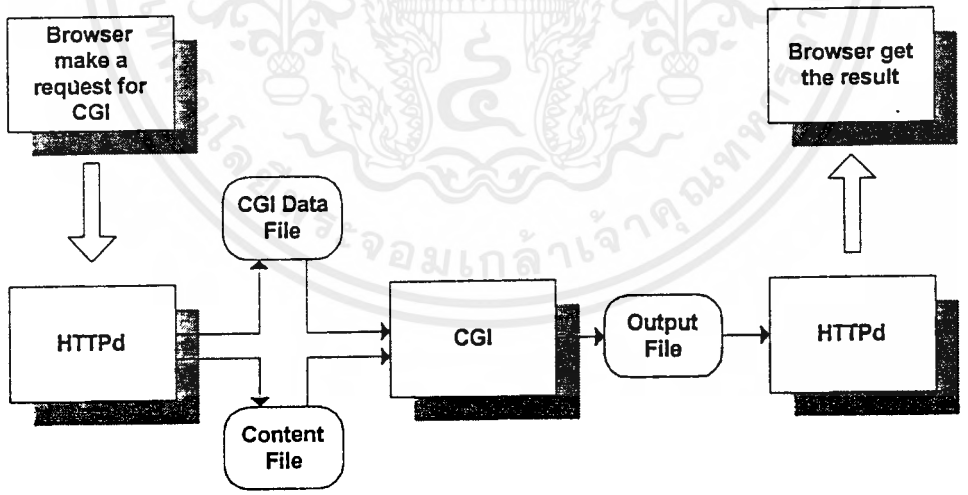
CONTENT-FILE

เป็นไฟล์ชั่วคราวที่เก็บข้อมูลที่ส่งมาจากฝั่ง Client ในการประมวลผลของโปรแกรมที่ฝั่ง Server สามารถที่นำข้อมูลจากไฟล์นี้ไปใช้งานได้

OUTPUT-FILE

ในการประมวลผลของโปรแกรมที่ฝั่ง Server ข้อมูลส่วนใดที่ต้องการให้เป็นผลลัพธ์ส่งกลับไปที่ฝั่ง Client จะถูกนำมาเขียนใส่ OUTPUT-FILE นี้ก่อน และเมื่อโปรแกรมทำงานเสร็จสิ้นแล้ว HTTPd Server ก็จะส่ง OUTPUT-FILE นี้ไปให้แก่ฝั่ง Client

ไฟล์ชั่วคราวทั้ง 3 ไฟล์นี้จะถูก HTTPd Server ลบทิ้งไปเมื่อโปรแกรม CGI ทำงานเสร็จสิ้นและ HTTPd Server ส่งผลลัพธ์กลับไปให้ Client แล้ว



รูปที่ 1 - 4 การใช้งานไฟล์ชั่วคราวใน Windows CGI

ดังนั้น ในการสร้าง CGI Application ขึ้นบน Windows จะต้องทำการอ่านข้อมูลและเขียนผลลัพธ์ลงที่ไฟล์ชั่วคราวที่กล่าวถึงนี้ ไม่สามารถที่จะรับค่าตัวแปรต่าง ๆ โดยตรงได้ อย่างไรก็ตาม ได้มีการสร้างเฟรมเวิร์ก (Framework) สำหรับการรับข้อมูลจากไฟล์ชั่วคราวนี้แล้ว โดยมีทั้งที่เป็นวิชวลเบสิก (Visual Basic) และเดลไฟ (Delphi) ซึ่งเฟรมเวิร์กนี้จะทำการ

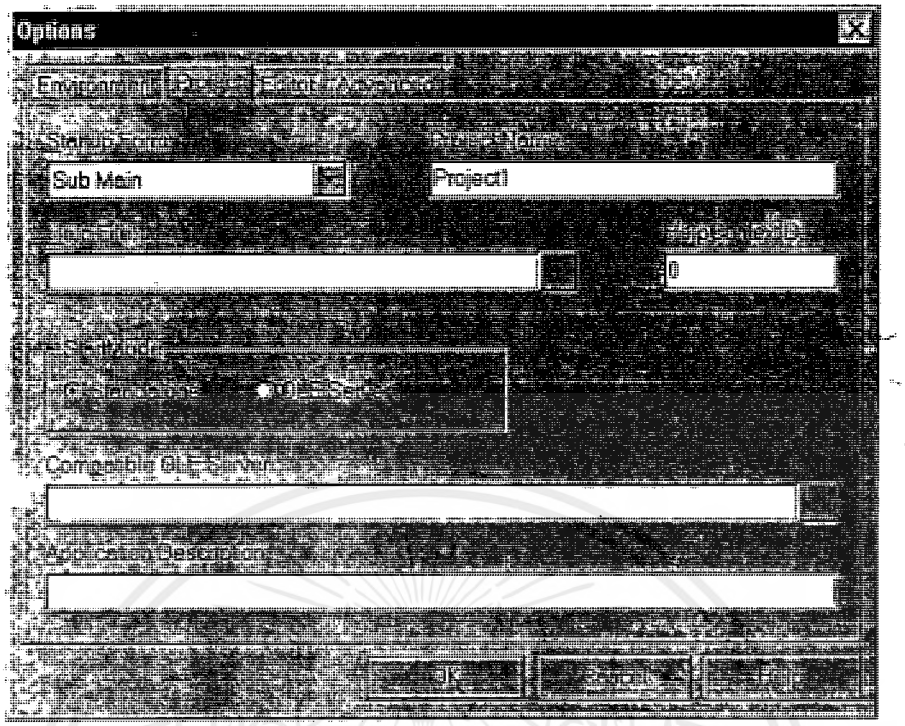
อ่านข้อมูลจาก CGI-DATA-FILE เพื่อกำหนดตัวแปรต่าง ๆ และกำหนดว่าจะต้องส่งผลลัพธ์ไปยังไฟล์ใด นอกจากนี้ยังมีส่วนของการรับข้อมูลจากแบบฟอร์มให้ใช้งานได้ด้วย เราสามารถใช้งานเฟรมเวิร์กนี้ได้โดยการนำไปรวมเข้ากับส่วนของ CGI Application ที่สร้างขึ้นได้ทันที

บทที่ 2
การสร้าง CGI
ด้วยภาษาโปรแกรมมิ่งบนวินโดว์

การสร้าง CGI แอพพลิเคชั่นโดยใช้ Visual Basic

เฟรมเวิร์กของการสร้าง CGI แอพพลิเคชั่นด้วยภาษาวิชวลเบสิก (Visual Basic) ที่จะกล่าวถึงในที่นี้ เป็นโมดูลภาษาวิชวลเบสิกที่สร้างขึ้นโดย Robert B. Denny (rdenny@dc3.com) (http://solo.dc3.com) ซึ่งภายในเฟรมเวิร์กนี้จะมีการกำหนดค่าเริ่มต้นต่าง ๆ ให้สามารถเรียกใช้งานได้ง่าย และยังมีรูทีนต่าง ๆ ให้ใช้งาน โดยผู้ใช้สามารถสร้าง CGI แอพพลิเคชั่น โดยใช้ เฟรมเวิร์กนี้ได้ โดยมีขั้นตอนดังนี้

1. เปลี่ยนให้การทำงานของแอพพลิเคชั่นเริ่มที่ส่วนของ Sub Main() ซึ่งเดิมนั้นแอพพลิเคชั่นจะถูกกำหนดให้เริ่มทำงานที่ Form1 การเปลี่ยนจุดเริ่มต้นการทำงานนี้ทำได้โดยเลือกที่เมนู Tools/Options... จากนั้นจะปรากฏหน้าต่างสำหรับตั้งค่าในส่วนต่าง ๆ ของแอพพลิเคชั่น ซึ่งจะมีส่วนของ Environment , Project , Editor และ Advanced ให้เลือกไปที่ส่วนของ Project จากนั้นตั้งค่าในส่วนของ Startup Form: ให้เป็น Sub Main ดังรูปที่ 2-1



รูปที่ 2 - 1 การกำหนดจุดเริ่มต้นการทำงานของแอพพลิเคชั่น

2. นำเฟรมเวิร์กไปรวมกับแอพพลิเคชั่น โดยเฟรมเวิร์กนี้จะเป็นไฟล์ภาษาวิชวลเบสิกชื่อ CGI32.BAS เราสามารถรวมไฟล์นี้เข้ากับแอพพลิเคชั่นได้โดยเลือกที่เมนู File/Add File จากนั้นก็เลือกไฟล์ CGI32.BAS นี้
3. ภายในเฟรมเวิร์กนี้จะมีส่วนของ Sub Main() อยู่แล้ว ซึ่งจะทำการกำหนดค่าเริ่มต้นต่าง ๆ ที่จำเป็น แล้วเรียกสับรูทีน (Sub Routine) ชื่อ Sub CGI_Main() ซึ่งสับรูทีนนี้ ก็คือส่วนของการทำงานของแอพพลิเคชั่นที่เราต้องการสร้างนั่นเอง
4. การเขียนโปรแกรมในส่วนของ Sub CGI_Main() นี้ก็สามารถเขียนได้ตามปกติของภาษาวิชวลเบสิก ไม่ว่าจะเป็นส่วนของการคำนวณ การติดต่อกับฐานข้อมูล หรือการทำงานอื่น ๆ เราสามารถสร้างส่วนของ Sub CGI_Main() ได้โดยการสร้างโมดูลใหม่ โดยเลือกที่เมนู Insert/Module จะปรากฏหน้าต่างสำหรับเขียนโปรแกรมสำหรับโมดูลใหม่นั้น
5. นอกจากสับรูทีนชื่อ Sub CGI_Main() แล้วยังมีสับรูทีนอีกอันที่จำเป็นต้องมี นั่นคือสับรูทีนชื่อ Sub Inter_Main() ซึ่ง Sub Inter_Main() นี้มีจะถูกเอ็กซีคิวต์เมื่อมีคำสั่งให้รันในโปรแกรมวิชวลเบสิกเท่านั้น โดยจะไม่ถูกเรียกใช้เมื่อมีการเรียกใช้งานแอพพลิเคชั่นผ่านบราวเซอร์จริง ๆ ส่วน Sub CGI_Main() จะไม่ถูกเอ็กซีคิวต์เมื่อใช้คำสั่งรันในโปรแกรมวิชวลเบสิก แต่จะถูกเอ็กซีคิวต์เมื่อมีการเรียกจากบราวเซอร์เท่านั้น ที่เป็นเช่นนี้เพราะเมื่อใช้คำสั่งรันในโปรแกรมวิชวลเบสิกนั้น ไฟล์ชั่วคราว (CONTENT-FILE , DATA-FILE, OUTPUT-FILE) ที่ใช้สำหรับเก็บข้อมูลของการติดต่อระหว่างบราวเซอร์ กับเว็บเซิร์ฟเวอร์นั้นยังไม่ถูกสร้างขึ้นมาจริง ๆ ดังนั้นโปรแกรมในส่วนของ Sub

CGI_Main() จึงยังไม่สามารถถูกเอ็กซีคิวต์ได้อย่างถูกต้อง และจากเหตุผลดังกล่าวมานี้เอง ทำให้เราไม่สามารถตรวจสอบการทำงานของแอพพลิเคชั่นที่สร้างขึ้นว่าทำงานได้ถูกต้องหรือไม่จากการใช้คำสั่งรันในโปรแกรมวิชวลเบสิก แต่เราสามารถตรวจสอบการทำงานได้จากการทดลองเรียกใช้ CGI แอพพลิเคชั่นนี้จากบราวเซอร์จริง ๆ เท่านั้น

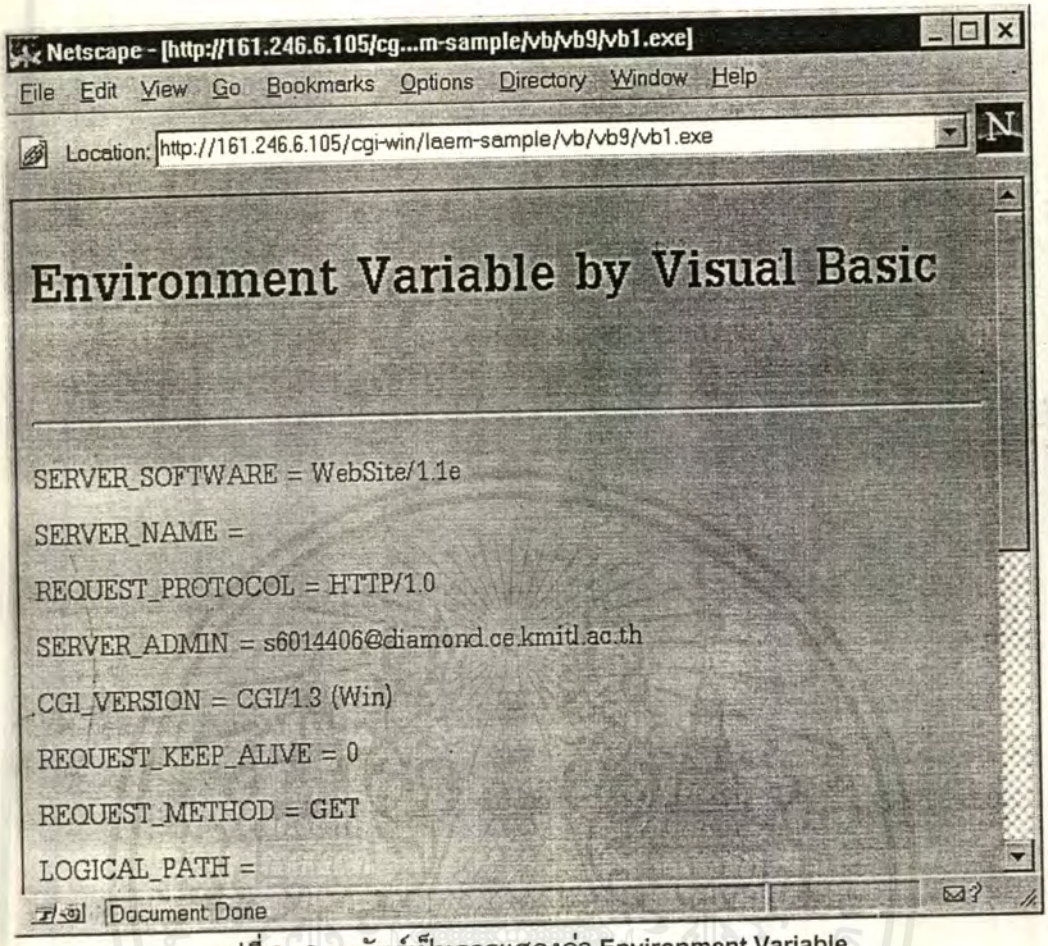
6. ในการสร้าง CGI แอพพลิเคชั่นนี้ เราสามารถส่งผลลัพธ์จากการเอ็กซีคิวต์ให้ออกไปเป็นผลลัพธ์ให้แก่ฝั่งไคลเอ็นท์ได้ โดยการส่งไปเป็นพารามิเตอร์ของสับรูทีนชื่อ Sub Send() ซึ่งเป็นสับรูทีนที่มีอยู่แล้วในเฟรมเวิร์ก (CGI32.BAS) โดยในสับรูทีนนี้จะทำการส่งสตริงที่รับมาเป็นพารามิเตอร์ ไปยังไฟล์ชั่วคราว OUTPUT-FILE นั้นเอง
7. ค่าต่าง ๆ จากแบบฟอร์มสามารถรับเข้ามาประมวลผลได้โดยใช้สับรูทีนสำเร็จรูปที่มีมาพร้อมกับเฟรมเวิร์กแล้ว เช่น สับรูทีนชื่อ Sub GetSmallField() ใช้สำหรับรับข้อมูลจากแบบฟอร์มที่ส่งมาด้วยวิธี POST เป็นต้น

ตัวอย่างแรกเป็นตัวอย่างง่าย ๆ ซึ่งจะแสดงการเรียกใช้งานสับรูทีนต่าง ๆ ที่มักใช้งานบ่อย ๆ โดยผลลัพธ์ที่ได้เมื่อเรียกใช้ผ่านบราวเซอร์จะแสดงค่า Enviroment Variable ต่าง ๆ สำหรับการติดต่อระหว่างบราวเซอร์กับเว็บเซิร์ฟเวอร์ แต่ถ้าใช้คำสั่งรันในโปรแกรมวิชวลเบสิก จะปรากฏแมสเสจบ็อก (Message Box) และแสดงข้อความว่า "Success" (ตามคำสั่งในสับรูทีน Sub Inter_Main()) ซอร์สโค้ดสำหรับตัวอย่างนี้มีดังนี้

```
'  Test for CGI written by Visual Basic 4.0 with Robert B. Denny
Framework.
'
' Vitaya Vinyunavan 17 January 1997
' ==========================================
'
Sub CGI_MAIN()

        Send ("Content-type: text/html")
        Send ("")
        Send ("<h1>Environment Variable by Visual Basic</h1><hr> ")
        Send ("SERVER_SOFTWARE = " & CGI_SERVERSOFTWARE
& "<br>")
        Send ("SERVER_NAME = " & CGI_SERVER_NAME & "<br>")
        Send ("REQUEST_PROTOCOL = " &
CGI_REQUESTPROTOCOL & "<br")
        Send ("SERVER_PORT = " & CGI_SERVERPORT & "<br>")
        Send ("SERVER_ADMIN = " & CGI_SERVERADMIN &
"<br>")
```

```
        Send ("CGI_VERSION = " & CGI_VERSION & "<br>")
        Send ("REQUEST_KEEP_ALIVE = " &
CGI_REQUESTKEEPALIVE & "<br>")
        Send ("REQUEST_METHOD = " & CGI_REQUESTMETHOD &
"<br>")
        Send ("LOGICAL_PATH = " & CGI_LOGICALPATH & "<br>")
        Send ("PHYSICAL_PATH = " & CGI_PHYSICALPATH &
"<br>")
        Send ( "QUERY_STRING = " & CGI_QUERYSTRING &
"<br>")
        Send ("REMOTE_HOST = " & CGI_REMOTEHOST & "<br>")
        Send ("REMOTE_ADDR = " & CGI_REMOTEADDR & "<br>")
        Send ("AUTH_TYPE = " & CGI_AUTHTYPE & "<br>")
        Send ("CONTENT_TYPE = " & CGI_CONTENTTYPE &
"<br>")
        Send ("CONTENT_LENGTH = " & CGI_CONTENTLENGTH &
"<br>")
        Send ("REFERER = " & CGI_REFERER & "<br>")
        Send ("FROM = " & CGI_FROM & "<br>")
        Send ("USER_AGENT = " & CGI_USERAGENT & "<br>")

End Sub

Sub Inter_Main()
   MsgBox ("Success")
End Sub
```

ซึ่งจะปรากฏผลลัพธ์บนบราวเซอร์ดังรูปที่ 2 - 2

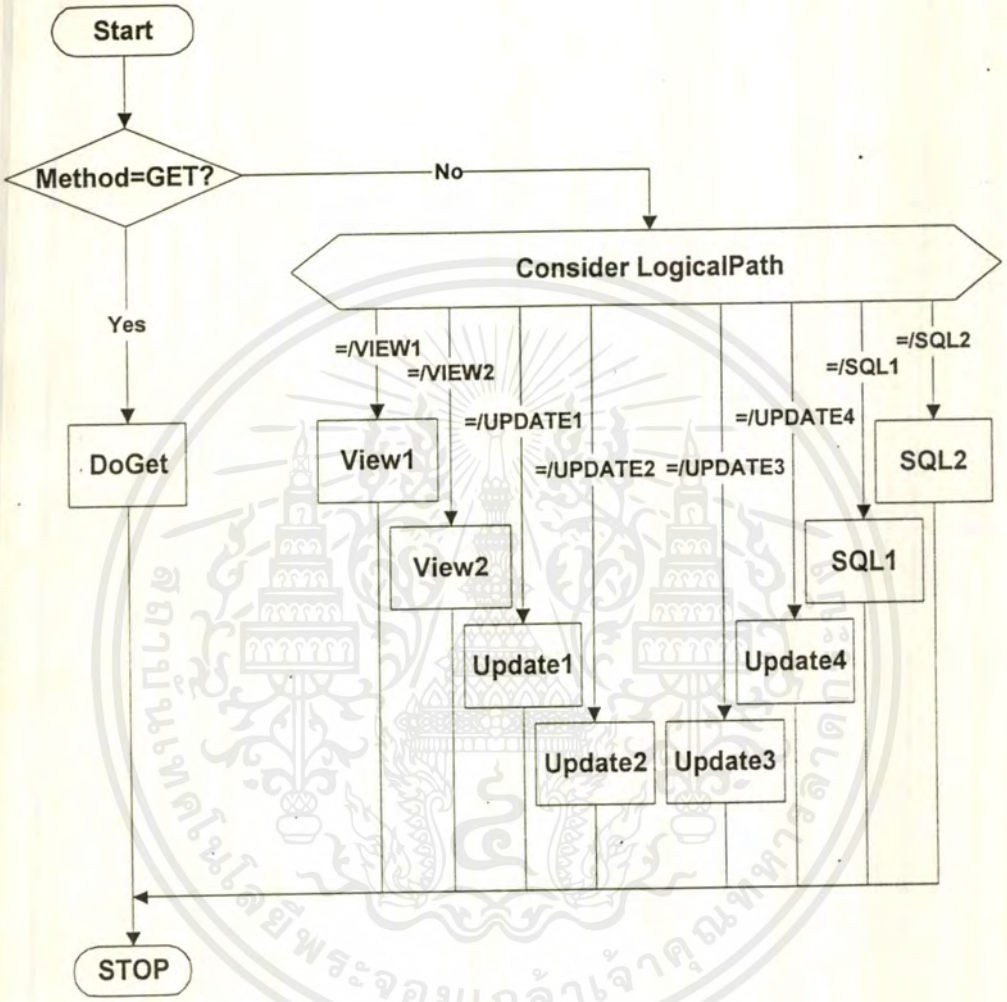


รูปที่ 2 - 2 ผลลัพธ์เป็นการแสดงค่า Environment Variable

ตัวอย่าง CGI แอพพลิเคชั่นอีกตัวอย่างหนึ่งที่สร้างขึ้นโดยใช้ภาษาวิชวลเบสิก ตัวอย่างนี้เป็นการสร้าง CGI แอพพลิเคชั่นเพื่อให้ผู้ใช้งานสามารถติดต่อเข้ามาเพื่อเรียกดูข้อมูล เกี่ยวกับนักศึกษาชั้นปีที่ 4 ภาควิชาวิศวกรรมคอมพิวเตอร์ ผ่านทางบราวเซอร์ โดยข้อมูลของนักศึกษา จะถูกจัดเก็บอยู่เป็นฐานข้อมูลบน SQL เซิร์ฟเวอร์ (SQL Server) โดยโปรแกรมวิชวลเบสิกนี้จะเชื่อมต่อเพื่อทำงานกับฐานข้อมูลผ่านทาง ODBC โดยตั้งชื่อ DSN (DatasourceName) = LaemSQL โดยชี้ไปยังฐานข้อมูลชื่อ Laem นอกจากนี้ ยังอนุญาตให้นักศึกษานั้นเข้ามาแก้ไขข้อมูลของตนเองให้ถูกต้อง และทันสมัย เช่น ข้อมูลเกี่ยวกับที่อยู่ หมายเลขโทรศัพท์ ข้อมูลเกี่ยวกับงานที่ได้ เป็นต้น โดยการแก้ไขข้อมูลต่าง ๆ สามารถทำได้โดยใช้ Dynaset ซึ่งเป็นตัวแปรชนิดหนึ่งสำหรับงานฐานข้อมูลในภาษาวิชวลเบสิก

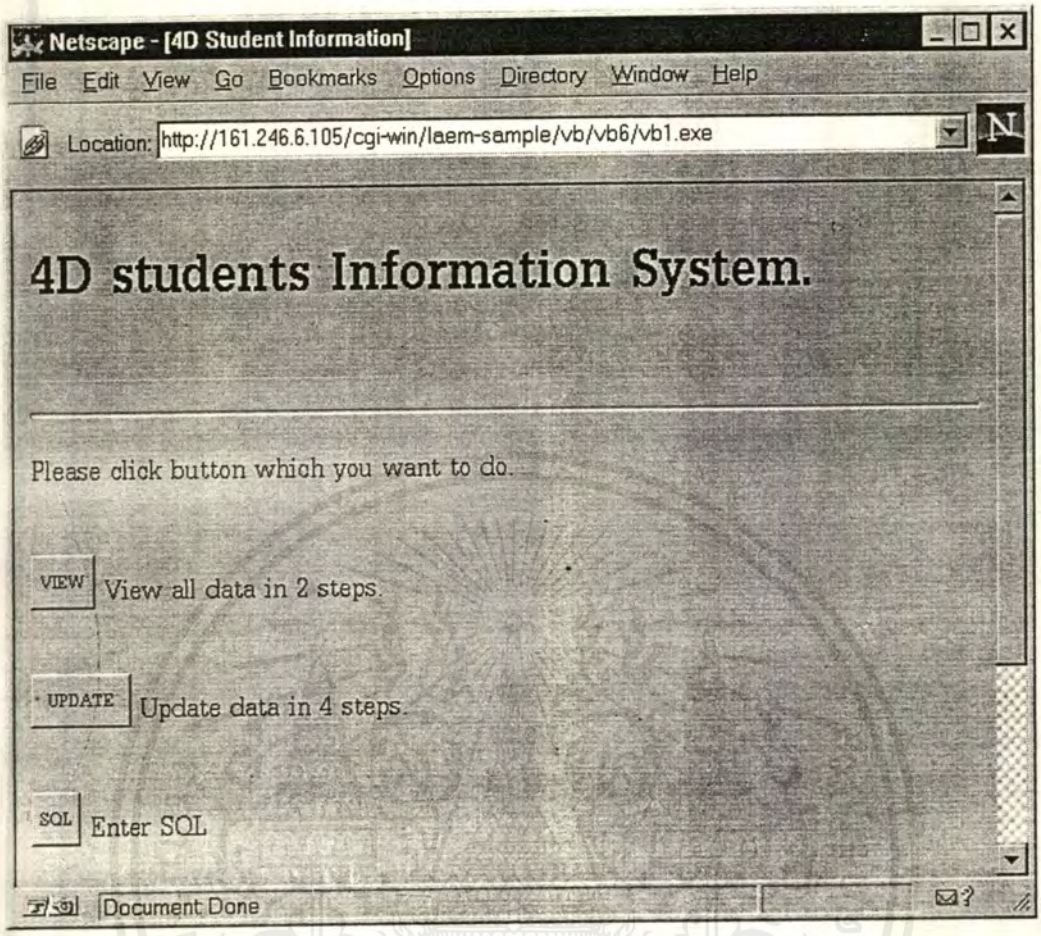
ตัวอย่างนี้ประกอบด้วยโมดูลที่สำคัญ 3 โมดูล นั่นคือโมดูลสำหรับ CGI เฟรมเวิร์ก , โมดูลสำหรับ Sub CGI_Main() และโมดูลสำหรับสับรูทีนทั่วไปอื่น ๆ ที่มักต้องใช้งานในหลาย ๆ แอพพลิเคชั่น เช่น สับรูทีนสำหรับการลงชื่อผู้จัดทำที่ด้านท้ายสำหรับแต่ละเพจ เป็น

ต้น ซอร์สโค้ดของตัวอย่างนี้สามารถดูได้จากภาคผนวก ฉ โดยในที่นี้จะอธิบายถึงขั้นตอนการทำงานคร่าว ๆ ดังรูปที่ 2 - 3



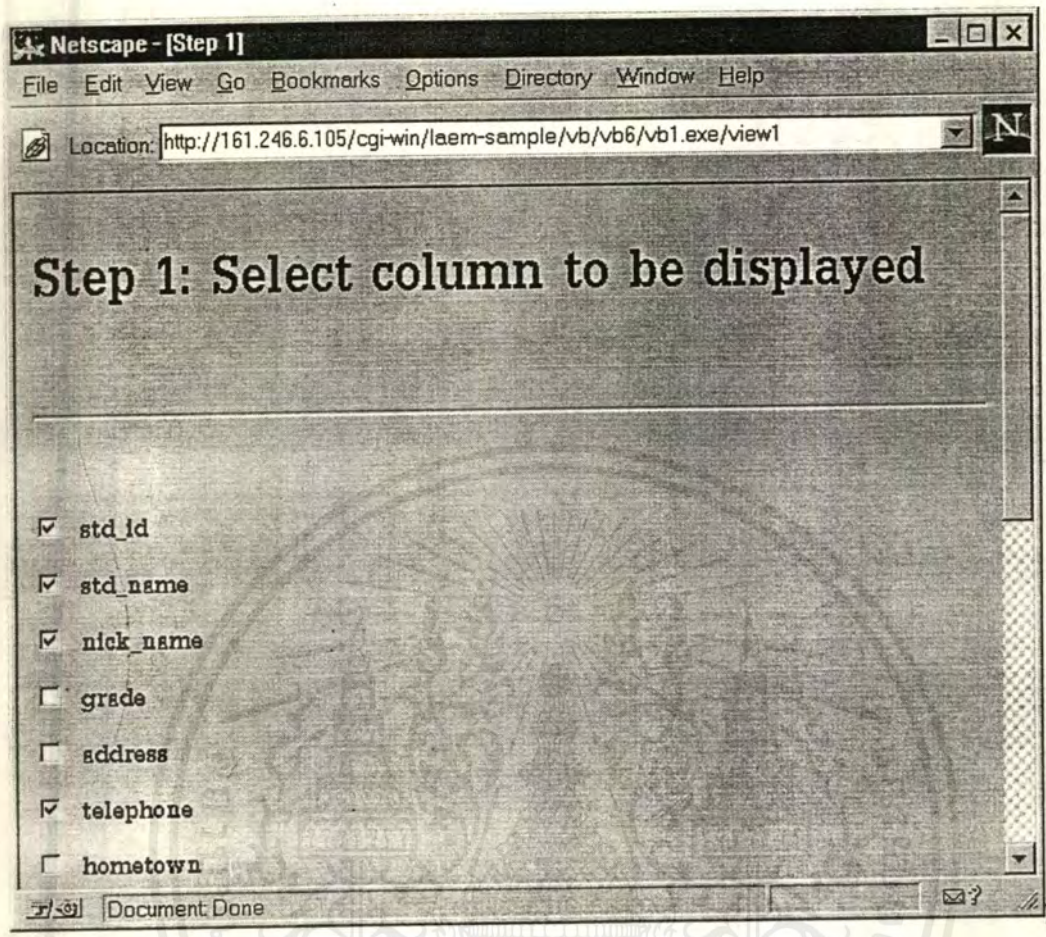
รูปที่ 2 - 3 การทำงานของแอพพลิเคชั่น

เมื่อผู้ใช้เริ่มเรียกใช้งาน CGI แอพพลิเคชั่นจะปรากฏผลลัพธ์บนบราวเซอร์ดังรูปที่ 2 - 4 โดยจะมีปุ่มให้ผู้ใช้งานเลือกกดเพื่อทำงานต่าง ๆ ดังนี้ VIEW สำหรับเรียกดูข้อมูลในคอลัมน์ที่กำหนดของนักศึกษาทุกคน , UPDATE สำหรับการแก้ไขข้อมูลของนักศึกษา และ SQL เป็นการเรียกดูข้อมูลโดยมีการกำหนดเงื่อนไขต่าง ๆ ได้ตามต้องการโดยใส่คำสั่งเป็นภาษา SQL สำหรับคิวรี่ (Query)

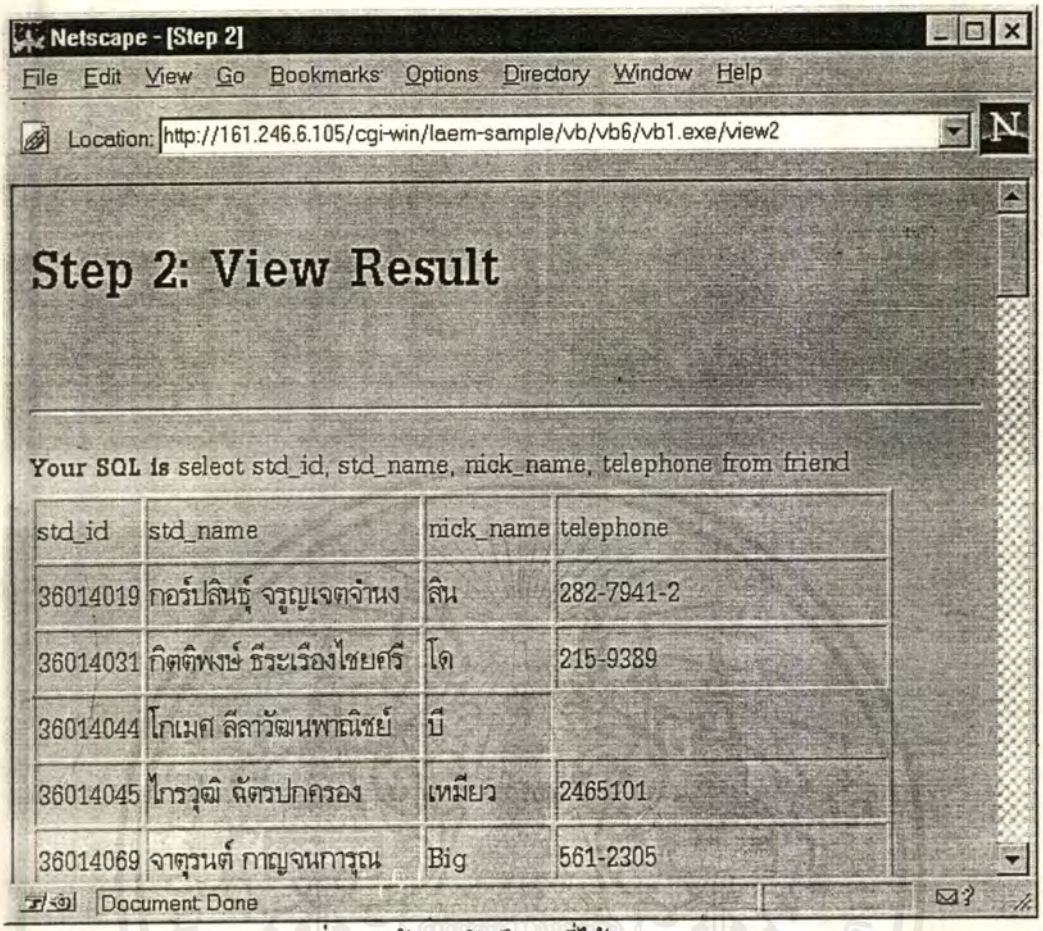


รูปที่ 2 - 4 หน้าแรกบนบราวเซอร์

การทำงานเมื่อกดปุ่ม VIEW จะมีการติดต่อจากบราวเซอร์กลับไปยังเว็บเซิร์ฟเวอร์เพื่อเรียกใช้งาน CGI แอพพลิเคชั่นนี้อีกครั้ง โดยคราวนี้จะเป็นการติดต่อด้วยวิธี POST (การติดต่อครั้งแรกเป็นการติดต่อด้วยวิธี GET) นอกจากนี้ก็ยังมีการส่งข้อมูลบางอย่างเป็น Environment Variable มีชื่อว่า LogicalPath ซึ่งจะนำมาใช้ในการบอกให้ CGI แอพพลิเคชั่นรู้ว่าการติดต่อเรียกใช้งานครั้งนี้ต้องการให้ CGI แอพพลิเคชั่นทำงานในส่วนใด ซึ่งในที่นี้เป็นการติดต่อเพื่อให้ทำงานในส่วนของ VIEW1 จึงมีการส่งค่า LogicalPath=VIEW1 จากนั้นจะปรากฏรายชื่อคอลัมน์ขึ้นมาให้เลือก ดังรูปที่ 2 - 5 เมื่อผู้ใช้งานเลือกคอลัมน์ที่ต้องการให้แสดงผลแล้วกดปุ่ม "NEXT >" ก็จะปรากฏข้อมูลของนักศึกษาในคอลัมน์ที่ต้องการแสดงขึ้นมาในรูปตาราง ดังรูปที่ 2 - 6



รูปที่ 2 - 5 เมื่อกดปุ่ม VIEW



รูปที่ 2 - 6 ข้อมูลนักศึกษาที่ได้จากการ VIEW

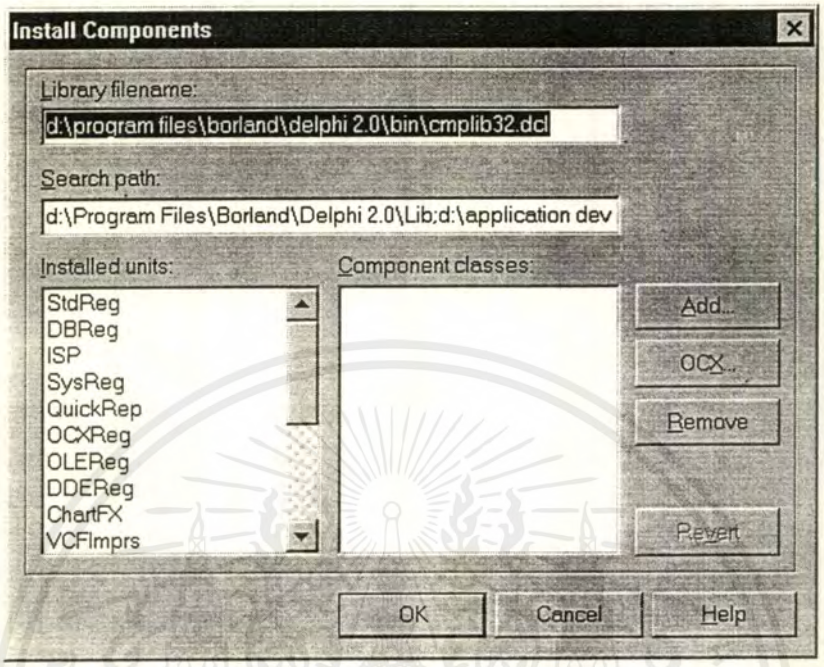
ในส่วนของปุ่ม UPDATE และ SQL ก็มีการทำงานคล้าย ๆ กับการทำงานเมื่อกดปุ่ม VIEW ผู้ใช้สามารถศึกษาการทำงานของ CGI แอพพลิเคชั่นนี้ได้จากซอร์สโค้ดของโปรแกรมในภาคผนวก ฉ

การสร้าง CGI แอพพลิเคชั่นโดยใช้ Delphi

เช่นเดียวกับการสร้าง CGI แอพพลิเคชั่นโดยใช้วิชวลเบสิก คือ มีการสร้างเฟรมเวิร์กไว้ให้ใช้งานได้สำเร็จรูป โดยเฟรมเวิร์กของ Delphi นี้ได้มีการแปลงมาจากเฟรมเวิร์กของวิชวลเบสิก โดย Ann Lynnworth (ann@href.com) (http://super.sonic.net/ann) โดยมีการเขียนเป็นยูนิต (UNIT) สามารถคอมไพล์ให้เป็นคอมโพเนนท์สำหรับใช้งานในโปรแกรมเดลไฟได้ทันที โดยมีชื่อว่า TCGIEnvData ซึ่งการใช้งานเฟรมเวิร์กนี้มีขั้นตอนดังนี้

1. ทำการเพิ่มคอมโพเนนท์ในโปรแกรม Delphi ให้ใช้งานคอมโพเนนท์ TCGIEnvData ได้ก่อนโดยการเพิ่มคอมโพเนนท์ของ Delphi สามารถทำได้โดยเลือกที่เมนู

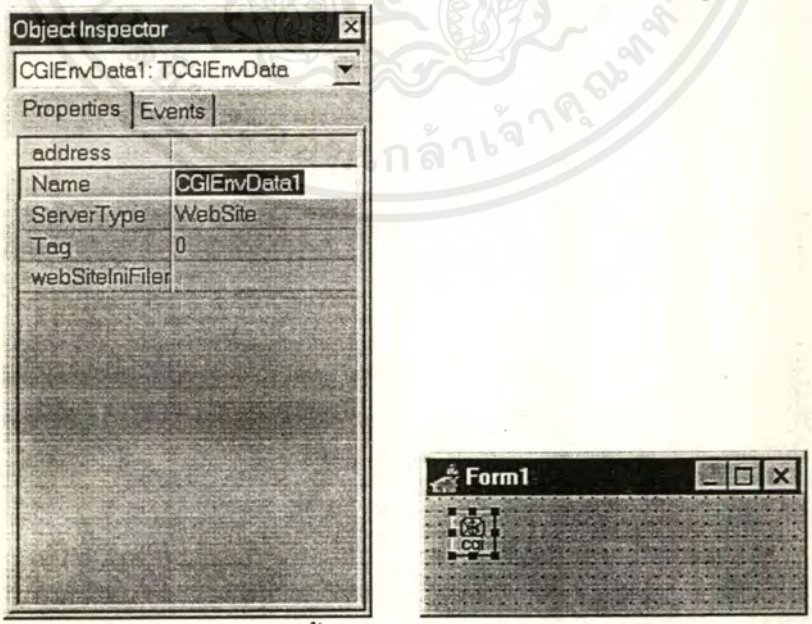
Component/Install... จากนั้นจะปรากฏหน้าต่างขึ้นมาเพื่อให้เพิ่มคอมโพเนนท์ ดังรูป 2 - 7



รูปที่ 2 - 7 หน้าต่างสำหรับเพิ่มคอมโพเนนท์ใน Delphi

จากนั้นทำการเพิ่มคอมโพเนนท์ที่ต้องการโดยการกดปุ่ม Add... แล้วใส่ชื่อไฟล์ของคอมโพเนนท์ซึ่งมักจะเป็นไฟล์นามสกุล .DCU

2. สร้างโปรเจคใหม่ นำคอมโพเนนท์ TCGIEnvData มาใส่ในฟอร์ม ดังรูปที่ 2-8



รูปที่ 2 - 8 พร็อพเพอร์ตี้และรูปของคอมโพเนนท์ TCGIEnvData

3. เริ่มเขียนแอพพลิเคชั่นได้ตามต้องการในส่วนของ Form.Create ซึ่งจะเป็นส่วนที่เริ่มต้นการประมวลผลเป็นส่วนแรกเมื่อมีการเรียกใช้แอพพลิเคชั่น ซึ่งก่อนจะเขียนโปรแกรมจะต้องมีการเรียกใช้ออบเจ็คท์ชื่อ CGIEnvData1 ก่อน

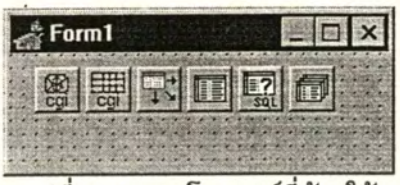
ตัวอย่างการสร้าง CGI แอพพลิเคชั่นโดยใช้ Delphi เป็นการส่งข้อความ "hello, world wide web!" กลับไปแสดงผลที่บราวเซอร์

```
procedure TForm1.create;
begin

        with CGIEnvData1 do
        begin
                websiteINIfilename := paramstr(1);
                application.onException := cgiErrorHandler;
                application.processMessages; {required}

                createStdout;
                sendPrologue;
                send( '<HTML><HEAD>' );
                sendTitle( 'Dynamic HTML Page' );
                send( '</HEAD><BODY>' );
                send( 'hello, world wide web!' );
                send( '</BODY></HTML>' );
                closeStdout;
                closeApp( application ); { don't leave form around }
        end;
end;
```

นอกจากคอมโพเนนท์ TCGIEnvData นี้แล้ว ยังมีคอมโพเนนท์อื่น ๆ ให้สามารถใช้งานได้ เช่นคอมโพเนนท์สำหรับการติดต่อกับฐานข้อมูล ชื่อ TCGIDB , คอมโพเนนท์สำหรับการส่งจดหมายอิเลกทรอนิคส์ เป็นต้น ผู้ใช้สามารถดาวน์โหลดคอมโพเนนท์เหล่านี้ได้จากอินเตอร์เน็ต

ตัวอย่าง CGI แอพพลิเคชั่นที่เขียนขึ้นโดย Delphi อีกตัวอย่างหนึ่ง เป็นแอพพลิเคชั่นสำหรับเรียกดูข้อมูลของนักศึกษาชั้นปีที่ 4 ภาควิชาวิศวกรรมคอมพิวเตอร์ และสามารถแก้ไขข้อมูลต่าง ๆ ของนักศึกษาได้ เหมือนกับตัวอย่าง CGI แอพพลิเคชั่นที่เขียนขึ้นด้วยภาษาวิชวลเบสิกดังที่กล่าวแล้วเมื่อตอนต้น การติดต่อกับฐานข้อมูลในแอพพลิเคชั่นที่เขียน

ขึ้นด้วย Delphi นี้ตัวแอพพลิเคชั่นจะติดต่อกับ BDE (Borland Database Engine) ก่อน แล้วจึงเชื่อมต่อกับ ODBC ซึ่งชี้ไปยังไฟล์ฐานข้อมูลของไมโครซอฟท์แอ็กเซส 7.0 แทน โดยแอพพลิเคชั่นเมื่อเขียนเสร็จแล้วจะประกอบด้วยคอมโพเนนท์ต่าง ๆ เช่น คอมโพเนนท์สำหรับ CGI เฟรมเวิร์ก , คอมโพเนนท์สำหรับติดต่อกับฐานข้อมูล ดังรูป 2 - 9



รูปที่ 2 - 9 คอมโพเนนท์ที่ต้องใช้

จากเนื้อหา พร้อมทั้งตัวอย่างต่าง ๆ ที่กล่าวมาทั้งหมดในส่วนนี้ ก็คงทำให้ผู้อ่านมีความเข้าใจ และเกิดแนวคิดเกี่ยวกับขั้นตอนการทำงานและการสร้าง CGI แอพพลิเคชั่นได้บ้างแล้ว

บทที่ 3

Java

ภาษาจาวา หรือ Java Programming Language เป็นภาษาใหม่ล่าสุด ที่บริษัท ซันไมโครซิสเต็มส์คิดค้นขึ้นเพื่อใช้ในการเขียนโปรแกรมบนอินเตอร์เน็ต ด้วยโปรแกรมที่เขียนขึ้นนี้จะทำให้เว็บเพจซึ่งแต่เดิมใช้งานในลักษณะของฐานข้อมูลเป็นส่วนใหญ่ จะมีสีสันขึ้น นั่นคือจะมีคุณสมบัติของเสียงและภาพเคลื่อนไหวประกอบเพิ่มขึ้น เราจะสามารถเล่นเกม คำนวณบัญชี พูดคุย รับข้อมูลที่ทันสมัยเสมอ ได้

Java Virtual Machine

Java Virtual Machine เป็นหลักการสร้างคอมพิวเตอร์จำลองของจาวา โดยการสมมติให้มีคอมพิวเตอร์อีกเครื่องหนึ่งขึ้นมาโดยเครื่องนี้จะใช้ในการคอมไพล์โปรแกรมภาษาจาวาทุกโปรแกรม เมื่อต้องการให้โปรแกรมภาษาจาวาไปทำงานบนคอมพิวเตอร์จริง ๆ เครื่องใด เราก็เพียงแต่สร้างตัวอินเตอร์พรีเตอร์ (Interpreter) ของคอมพิวเตอร์จำลองตัวนี้บนเครื่องนั้น ๆ ภาษาจาวาทุกโปรแกรมก็จะสามารถทำงานบนระบบคอมพิวเตอร์นั้น ๆ ได้ตามต้องการ ด้วยหลักการนี้ก็เป็นที่มาของคุณสมบัติของจาวา ที่ไม่ขึ้นกับแพลทฟอร์มและฮาร์ดแวร์ใด ๆ หรือพอร์ตเทเบิล (Portable) เราอาจจะเรียก Java Virtual Machine สั้น ๆ ว่า Java VM ก็ได้

สำหรับเหตุผลที่ต้องมี Java VM นี้ จริง ๆ แล้วเป็นการกำหนดคำขึ้นมา เพื่อเป็นคำจำกัดความเฉพาะในความคิดเท่านั้น เพื่อให้นักพัฒนาจะได้ไม่ถูกบังคับให้ต้องสร้างตัวอินเตอร์พรีเตอร์ตามแนวทางแบบใดแบบหนึ่งโดยเฉพาะ แต่ตัวอินเตอร์พรีเตอร์ที่สร้างขึ้นตามข้อกำหนดดังกล่าว ไม่ว่าจะอยู่บนแพลทฟอร์มใด จะสามารถรันโปรแกรมที่เขียนขึ้นในภาษาจาวาได้ โดยให้ผลลัพธ์ออกมาเหมือนกัน

แต่ในข้อกำหนดของ Java VM ก็มีการให้นิยามที่ชัดเจนมาก ๆ เกี่ยวกับการออกแบบอินเตอร์พรีเตอร์ในหลาย ๆ ส่วน โดยเฉพาะอย่างยิ่งส่วนที่เกี่ยวข้องกับการกระจายโค้ดของจาวาไปใส่ไว้ในรูปแบบที่กำหนด ข้อกำหนดนี้ได้แก่ไวยากรณ์ของออบโค้ดและโอเปอร์แรนด์ พร้อมกับค่าประจำตัว การจัดโครงสร้างของโค้ด การจัดวางรูปแบบของออบเจ็คท์

ของจาวา ข้อกำหนดต่าง ๆ เหล่านี้จะทำให้นักพัฒนาอินเตอร์พรีเตอร์ทั้งหลาย สามารถจะสร้างอินเตอร์พรีเตอร์ขึ้นมาใหม่บนแพลทฟอร์มใด ๆ ก็ได้ ดังนั้น การพัฒนาภาษาจาวา จึงไม่ได้ถูกปิดกั้นอยู่กับระบบของซันซึ่งเป็นผู้คิดค้นภาษานี้ขึ้นมาเท่านั้น

จากที่กล่าวมาข้างต้นถึงคุณสมบัติของจาวาที่ไม่ยึดติดอยู่กับแพลทฟอร์มใด ๆ ทำให้การทำงานกับระบบคอมพิวเตอร์ที่ทำงานแบบกระจาย (Distributed computing) ได้รับการตอบสนองอย่างเหมาะสม นั่นคือ ทำให้เกิดความพอร์ตเทเบิลนั่นเอง

อย่างไรก็ตาม Java VM ก็ยังมีข้อจำกัดอยู่ ข้อจำกัดของ Java VM นั้นจะอยู่ที่ข้อจำกัดของการออกแบบตัวอินเตอร์พรีเตอร์แทน เช่น การจำกัด ค่าของโอเปอร์แรนด์ และขนาดของสแต็ก เป็นต้น ข้อกำหนดเหล่านี้หมายถึงว่า Java VM สามารถจะอ้างถึงหน่วยความจำได้เฉพาะในห้วงแอดเดรสเท่าที่มีอยู่เท่านั้น

ข้อจำกัดภายในของตัว Java VM เองนั้นมีห้วงแอดเดรสให้ใช้อยู่ถึง 4 GB เพราะขนาดของความกว้างของการอ้างแอดเดรสเป็น 32 บิต method ต่าง ๆ ของจาวามีขนาดได้เพียง 32 KB เพราะมีข้อจำกัดของการใช้คำสั่งกระโดด เป็นแบบ 16 บิต (โดยมีบิดแรกเป็นตัวบอกว่ากระโดดไปข้างหน้า หรือข้างหลัง และบิดต่อ ๆ ไป บอกระยะทางการกระโดดจากจุดที่ทำงานอยู่) จำนวนตัวแปรในแต่ละชุดสแต็กจะถูกจำกัดอยู่ที่ 256 ตัว เพราะดัชนีที่ใช้ชี้ตัวแปรเหล่านี้มีขนาดเพียง 8 บิด นอกจากนั้น จำนวนของค่าคงที่ที่อยู่ในส่วนกลาง ยังถูกจำกัดด้วยขนาดของดัชนี 16 บิด ทำให้มีจำนวนได้เพียง 32,000 ค่าในแต่ละ method

การที่เราถือว่า การมีค่าเหล่านี้ เป็นข้อจำกัด ก็อาจจะเป็นการมองการณ์ไกลไปสักหน่อย เนื่องจากปัจจุบันเครื่องคอมพิวเตอร์ส่วนใหญ่มีหน่วยความจำกัดเพียง 16 หรือ 32 MB กันเท่านั้น หน่วยความจำขนาด 4 GB จึงยังเป็นเรื่องที่ไม่ต้องคิดหนักในตอนนี้ ในขณะที่ขนาดของ method ที่จำกัดที่ 32 KB ก็เป็นเพียง method เดียวเท่านั้น

หลักการทำงานของโปรแกรมภาษาจาวา

ในที่นี้ขออธิบายการทำงานของภาษาจาวาเริ่มตั้งแต่การคอมไพล์ตัวโปรแกรมจนกระทั่งเราเรียกใช้งานในเว็บเพจ เป็นดังนี้

การคอมไพล์

คอมไพเลอร์ของภาษาจาวาก็เช่นเดียวกับคอมไพเลอร์ในภาษาอื่น ๆ นั่นคือ มันจะสร้างรหัสภาษาเครื่อง (Machine Code หรือ Assembler Code) จากภาษาในระดับที่สูงกว่าเพื่อให้ซีพียู (CPU : Central Processing Unit) สามารถนำไปใช้งานได้

แต่ข้อแตกต่างที่สำคัญระหว่างคอมไพเลอร์ของภาษาจาวากับภาษาอื่น ๆ คือ ซีพียูหรือโปรเซสเซอร์ที่จะทำหน้าที่ในการปฏิบัติตามคำสั่งที่ได้จากการคอมไพล์ภาษาจาวานั้นไม่มีอยู่จริง เป็นเพียงสิ่งที่สมมติขึ้นมาที่เรียกว่า Java Virtual Machine นอกจากนี้การอ้างถึงส่วนต่าง ๆ ของโปรแกรมที่คอมไพล์ด้วยคอมไพเลอร์ของภาษาจาวาก็จะมีวิธีการที่แตกต่างออกไป

คอมไพเลอร์ของภาษาจาวาจะไม่เปลี่ยนการอ้างถึงส่วนของโปรแกรมจากการ ใช้ชื่อแบบในภาษาสูง ไปเป็นตัวเลขเหมือนคอมไพเลอร์ภาษาอื่น ๆ ทำกัน และคอมไพเลอร์ภาษาจาวาก็จะไม่มีการสร้างแผนที่ของการจัดวางโปรแกรมบนหน่วย ความจำขึ้นมาในระหว่างการคอมไพล์ด้วยเหตุผลที่สำคัญคือ เพื่อเป็นการสร้างความพอร์ตเทเบิลให้กับตัวโปรแกรม เพราะการจัดวางตำแหน่งของโปรแกรมจะต้องขึ้นอยู่กับลักษณะการทำงานของโปรเซสเซอร์ตัวใดตัวหนึ่ง การยังไม่จัดวางตำแหน่งช่วยทำให้โปรแกรมที่ได้จากการคอมไพล์มีความเป็นกลาง สามารถนำไปใช้บนคอมพิวเตอร์แพลทฟอร์มอื่น ๆ ได้ นอกจากนั้นยังทำให้เกิดความปลอดภัยอีกด้วย

สิ่งที่ได้จากการคอมไพล์ในภาษาจาวาเราเรียกว่า **ไบท์โค้ด (ByteCode)**

ออบโค้ด และโอเปอแรนด์ในจาวา

ส่วนต่าง ๆ ของโปรแกรมเมื่อเราไปอ้างอิงในคอมพิวเตอร์แบบต่าง ๆ จะประกอบด้วยส่วนประกอบ 2 ส่วนด้วยกัน คือ

1. ออบโค้ด คือ คำสั่งที่จะให้คอมพิวเตอร์ทำงานอย่างใดอย่างหนึ่ง
2. โอเปอแรนด์ คือ ข้อมูลที่จำเป็นต้องใช้ในการทำงานตามออบโค้ด

ทั้งออบโค้ดและโอเปอแรนด์จะถูกจัดวางเรียงอยู่เป็นแถวเพื่อให้เครื่องคอมพิวเตอร์เรียกเข้าไปทำงานเรียงไปตามลำดับ ตามคำสั่งอาจจะมีการเรียกข้อมูลตัวเลขออกมาจากหน่วยความจำแล้วเก็บใส่ไว้ในสแต็ก คำสั่งอาจจะให้ดึงตัวเลขออกมาอีกตัวหนึ่งแล้วก็ใส่เข้าไปในสแต็กอีก จากนั้นบวกข้อมูลตัวเลขทั้งสองเข้าด้วยกันแล้วเก็บผลลัพธ์กลับเข้าสู่หน่วยความจำ คำสั่งตามรูปแบบชุดคำสั่งของ Java Virtual Machine ที่ทำงานตามนี้ ก็คือ

```
lload address
lload address
ladd
lstore address
```

ซึ่งจะสามารถคอมไพล์ออกมาเป็นภาษาเครื่องได้ดังนี้

```
22 xxxx
22 xxxx
97
55 xxxx
```

จะเห็นได้ว่าออบโค้ดจะเป็นตัวเลขขนาด 8 บิตธรรมดา เพื่อบอกให้เครื่องรู้ว่าจะให้ทำงานอะไร ส่วนตัวแปรแอดเดรสเป็นตัวเลขที่ใช้บอกเครื่องว่าให้ไปดึงค่าของตัวแปรมาจากที่ไหนในหน่วยความจำ ส่วนนี้ก็คือโอเปอร์แรนด์ ซึ่งขนาดของตัวแปรแอดเดรสแต่ละตัวจะเป็นเลข 32 บิด ดังนั้น จากออบโค้ดและโอเปอร์แรนด์ข้างต้นรวมกันแล้วจะใช้พื้นที่ในหน่วยความจำ 16 ไบท์หรือ 128 บิด ซึ่งถ้าโปรแกรมส่วนเล็ก ๆ นี้เป็นสมาชิกประเภท method ของคลาส มันก็จะถูกผนึกรวมเข้าไปในทุก ๆ method ที่อยู่ในคลาสเดียวกัน เมื่อตัวคอมไพเลอร์ทำการสร้างโค้ด

ส่วนวิธีการที่คอมไพเลอร์จะทราบตำแหน่งที่อยู่และกระโดดไปทำงานที่ส่วนของโปรแกรมที่ถูกต้องเมื่อมีส่วนอื่น ๆ ของโปรแกรมเรียกใช้นั้น ก็คือ คอมไพเลอร์จะเก็บขนาดความยาวของทุก ๆ ส่วนของโค้ดเอาไว้แล้วจัดวางลงบนหน่วยความจำเรียงตามลำดับ หลังจากนั้นก็เพียงแต่บอกให้เครื่องกระโดดไปทำงานที่ตำแหน่งของแอดเดรสที่เป็นจุดเริ่มต้นของ method ที่ต้องการจะเรียกใช้งานเท่านั้น ในการเรียกใช้ method จะต้องใช้คำสั่ง jsr address ซึ่งแปลออกมาเป็นภาษาเครื่องได้ 168 xx โดยแอดเดรสในที่นี้เป็นเลข 16 บิด

การวางตำแหน่งในหน่วยความจำ

ในภาษาจาวาจะไม่มีการลดรูปแบบการอ้างถึงส่วนต่าง ๆ ของโปรแกรมจากการเรียกเป็นชื่อให้เหลือเพียงตัวเลขหรือแอดเดรสที่กำหนดขึ้นจากการจัดวางตำแหน่งของโปรแกรมลงในหน่วยความจำ คอมไพลเลอร์ภาษาจาวาจะทิ้งชื่อของแต่ละส่วนของโปรแกรม (โดยเฉพาะ method) เอาไว้ในตัวโปรแกรมที่สร้างขึ้น เมื่อโปรแกรมทำงาน จะเป็นหน้าที่ของตัว อินเตอร์พรีเตอร์ที่จะคอยเปิดตารางค้นหาที่อยู่ของ method ที่ต้องการเรียกใช้งาน โดยก่อนที่จะเริ่มทำงานจริง อินเตอร์พรีเตอร์จะต้องสร้างแผนที่ในการจัดวางสิ่งต่าง ๆ

ลงในหน่วยความจำขึ้นมาก่อนแล้วจึงสร้างตารางขึ้นมา เพื่อช่วยหาตำแหน่งของ method เมื่อมีการเรียกใช้งานโดยใช้ชื่อของ method

การรันโปรแกรม

การรันโค้ด ที่คอมไพล์เอาไว้สำหรับ Java Virtual Machine เป็นหน้าที่ของตัวอินเตอร์พรีเตอร์ การรันโปรแกรมจะแบ่งได้เป็น 3 ขั้นตอนหลัก ๆ คือ การอ่าน การตรวจสอบความถูกต้อง และการทำตามโค้ด หน้าที่ในการอ่านโค้ดเข้าสู่ระบบจะเป็นของ Class Loader หน้าที่การทำงานในส่วนนี้จะไม่ได้อ่านเข้ามาเฉพาะไฟล์จาวาที่กำลังจะเรียกใช้เท่านั้น แต่จะอ่านคลาสที่มีการอ้างถึงและคลาสที่มีการ inherited มาโดยคลาสที่อ้างถึง เมื่อผ่านขั้นตอนนี้แล้ว โค้ดทั้งหมดก็จะถูกส่งผ่านตัวตรวจสอบไบท์โค้ดเพื่อให้แน่ใจว่าโค้ดที่ส่งมามีความถูกต้องตามมาตรฐานของจาวา และจะไม่รบกวนเสถียรภาพของระบบ เมื่อผ่านการตรวจสอบแล้วโค้ดก็จะถูกส่งต่อไปยังระบบรันไทม์ (Run-Time System) ซึ่งจะสั่งงานไปยังฮาร์ดแวร์อีกต่อหนึ่ง ซึ่งหลักการทำงานในแต่ละขั้นตอนที่กล่าวมาข้างต้นเป็นดังนี้

Class Loader

ตัวคลาสโหลดเดอร์ จำทำหน้าที่ดึงโค้ดทั้งหมดที่จำเป็นในการทำงานของแอพพลิเคชั่น ไม่ว่าจะเป็นคลาสที่ถูก inherited มา หรือคลาสอื่น ๆ ที่มีการเรียกใช้ เมื่อโหลดเดอร์ดึงคลาสใดเข้ามาแล้วก็จะจัดคลาสนั้น ๆ ใส่เข้าไว้ใน namespace ของมันเอง โดยการเก็บ จะใช้ชื่อของคลาสเป็นสำคัญ ไม่ได้ใช้การอ้างถึงเป็นตัวเลข หลักการนี้ก็จะเหมือนกันกับการทำงานของ Virtual Machine ที่โอเอส (OS : Operating System) สร้างขึ้นให้แอพพลิเคชั่นแต่ละตัวทำงาน ถ้าไม่ได้มีการเจาะจงเรียกใช้คลาสที่อยู่นอก namespace นี้ การเรียกใช้ชื่อต่าง ๆ ในคลาส ก็จะไม่มีการรบกวนกันระหว่างคลาสเลย

คลาสทั้งหมดที่อยู่บนเครื่องโลคอลเอง จะได้รับห้วงแอดเดรส (Address Space) เป็นของตัวเอง ส่วนคลาสต่าง ๆ ที่ดึงมาจากภายนอกจะได้รับ namespace เป็นของตัวเอง การทำงานลักษณะนี้จะช่วยให้คลาสที่อยู่บนโลคอลทำงานได้ประสิทธิภาพดีขึ้น เพราะใช้ namespace ร่วมกันได้ แต่ก็ยังมีการป้องกันความผิดพลาดที่อาจจะเกิดจากคลาสที่ดึงเข้ามาจากภายนอก และในทางกลับกัน คลาสที่อิมพอร์ตเข้ามาก็ปลอดภัยจากความผิดพลาดที่อาจจะเกิดขึ้นจากคลาสโลคอล ด้วย

เมื่อคลาสทั้งหมดที่เกี่ยวข้องกับการทำงานถูกอิมพอร์ตเข้ามาเรียบร้อยแล้ว การจัดวางหน่วยความจำสำหรับเริ่มต้นการทำงานก็จะเกิดขึ้นได้ การเรียกชื่อต่าง ๆ ก็จะสามารถจับคู่กับแอดเดรสจริง ๆ ของหน่วยความมาจำได้ แล้วตัวโหลดเดอร์จะสร้างตารางสำหรับค้นหาที่อยู่ของสิ่งต่าง ๆ ขึ้น (Look-up Table) การสร้างตารางขึ้น เป็นขั้นตอนสุดท้ายนี้ ทำให้ลดความเสี่ยงจากการทำงานผิดปกติของซุปเปอร์คลาส (Super Class) และการอ้างแอดเดรสที่ไม่ถูกต้องได้

ตรวจสอบไบท์โค้ด

เมื่อโค้ดเดินทางมาจนถึงขั้นตอนการสร้างตารางจับคู่ชื่อกับแอดเดรสแล้ว ก็ยังไม่สามารถแน่ใจได้ว่าโค้ดที่อ่านเข้ามาจะมีความปลอดภัย ดังนั้น จึงต้องมีตัว verifier หรือตัวตรวจสอบไบท์โค้ด ทำหน้าที่ตรวจสอบความถูกต้องทีละบรรทัดว่าเป็นไปตามข้อกำหนดของจาวา และสอดคล้องกับการทำานของตัวโปรแกรมเองหรือไม่ การตรวจสอบโค้ดในเชิงทฤษฎีจะสร้างค้นหาปัญหาต่าง ๆ ได้หลายอย่าง เช่น จะไม่มีการสร้างพอยต์เตอร์ที่เกินกว่าหน่วยความจำจริง ไม่มีคำสั่งใดสามารถละเมิดสิทธิ์การทำงานของตัวโปรแกรมได้ ไม่มีการจับคู่ออบเจ็คท์ผิด จะไม่มีการให้โอเปอร์แรนด์มากหรือน้อยเกินไป การกำหนดค่าต่าง ๆ สำหรับไบท์โค้ดจะต้องถูกต้องครบถ้วน และจะไม่มีการแปลงข้อมูลผิดรูปแบบ

การใช้ตัวตรวจสอบตอบสนองจุดประสงค์ 2 ประการสำคัญ คือ สิ่งต่าง ๆ ดังที่กล่าวมาแล้วจะถูกตรวจสอบก่อนทำให้ตัวอินเตอร์พรีเตอร์มั่นใจได้ว่า ไบท์โค้ดที่ส่งเข้าไปทำงานจะไม่มีขั้นตอนการทำงานที่สร้างปัญหาให้กับตัวระบบ และจุดประสงค์ที่สองก็คือ ตัวอินเตอร์พรีเตอร์จะทำงานตามไบท์โค้ดได้รวดเร็วกว่า เพราะไม่ต้องคอยระวังว่าจะมีปัญหาเกิดขึ้น และไม่ต้องหยุดเป็นช่วง ๆ เมื่อพบปัญหาและต้องแก้ไข

ในการทำงาน ไบท์โค้ดจะถูกตรวจสอบเพียงครั้งเดียวเท่านั้น และจะทำงานไปได้ตลอดไม่ต้องมีการตรวจสอบซ้ำอีกเมื่อมีการเรียกกลับมาทำงานที่ส่วนเดิมของโปรแกรม

การทำงานตามโค้ด

เมื่อตัวโหลดเดอร์ได้รวบรวมโค้ดเข้ามาสู่ระบบทำการจัดวางในหน่วยความจำ และตัวตรวจสอบได้ทำการตรวจสอบความถูกต้องแล้ว โค้ดก็จะถูกส่งต่อไปยังตัวอินเตอร์พรีเตอร์เพื่อทำงานตามคำสั่ง การทำงานตามคำสั่งของโค้ดก็คือการเปลี่ยนโค้ดให้กลายเป็นคำสั่ง การทำงานจริงที่ตัวระบบไคลเอ็นท์ที่รันโค้ดนี้สามารถทำงานได้ ซึ่งวิธีการที่ทำได้ก็มีอยู่ 2 วิธีด้วยกัน คือ ตัวอินเตอร์พรีเตอร์ทำการคอมไพล์โค้ดเหล่านี้ให้กลายเป็นเนทีฟโค้ดที่ตัวเครื่องไคลเอ็นท์เข้าใจ แล้วค่อยทำงาน เพื่อให้ได้ความเร็วสูงสุดในการทำงาน กับอีกวิธีหนึ่งก็คือ ตัวอินเตอร์พรีเตอร์อ่านโค้ดเข้ามาแล้วดีความทำงานไปทีละคำสั่ง และทำการดีความไปเรื่อย ๆ ตลอดเวลาที่มีการทำงาน

โดยปกติแล้ว ผู้สร้างตัวอินเตอร์พรีเตอร์มักจะเลือกใช้วิธีการหลัง รูปแบบของไบท์โค้ดในภาษาจาวามีคยามยืดหยุ่นเพียงพอที่จะสามารถเปลี่ยนไปทำงานบนเครื่องไคลเอ็นท์แบบต่าง ๆ ได้โดยไม่มีการก่อให้เกิดโอเวอร์เฮดมากมายเกินความจำเป็น อย่างไรก็ตามไคลเอ็นท์ของจาวาบางระบบจะมีความสามารถในการทำงานได้ทั้งสองวิธี คือ โปรแกรมเมอร์สามารถจะเลือกใช้วิธีการคอมไพล์กับงานที่เน้นการคำนวณมาก ๆ เพื่อเป็นการเพิ่มสมรรถนะในการทำงานให้ได้เต็มที่ ซึ่งไคลเอ็นท์แบบนี้จะให้ได้ทั้งความพอร์ดเทเบิล และสมรรถนะที่ดี

การสร้างระบบรันไทม์ที่ดีจะต้องถ่วงดุลย์ความสำคัญ 3 ประการให้ได้พอเหมาะ นั่นคือ ความพอร์ดเทเบิล ความปลอดภัย และสมรรถนะ เรื่องของความพอร์ดเทเบิลทำได้โดยการใช้รูปแบบของไบท์โค้ดที่มีความเป็นกลางเพียงพอ สามารถนำไปใช้รันบนเครื่องคอมพิวเตอร์แบบต่าง ๆ ได้โดยง่าย นอกจากนั้น การที่ตัวอินเตอร์พรีเตอร์ทำการกำหนดการจัดวางตำแหน่งในหน่วยความจำในช่วงรันไทม์ (แทนที่จะเป็นระหว่างการคอมไพล์เหมือนภาษาอื่น ๆ ก็เป็นการเพิ่มความแน่นอนว่า คลาสต่าง ๆ ที่อิมพอร์ตเข้ามาจะยังคงใช้ได้อยู่ตลอดเวลา เรื่องของความปลอดภัยนั้น เป็นสิ่งที่มีการคำนึงถึงอยู่ตลอดกระบวนการทำงานของระบบรันไทม์ โดยเฉพาะอย่างยิ่งในส่วนของตัวตรวจสอบไบท์โค้ด ที่ทำให้แน่ใจได้ว่าโปรแกรมจะทำงานได้ถูกต้องตามข้อกำหนดของจาวา ส่วนเรื่องของสมรรถนะนั้นก็สามารถจัดการได้ในสองระยะ คือ พยายามเอาโอเวอร์เฮดทั้งหลายไปใส่ไว้ที่ตอนเริ่มต้น โหลดโปรแกรมเข้ามาสู่ระบบ หรือไม่ก็กำหนดให้ทำงานเป็นแบบแบ็กกราวนด์ (Back-Ground Thread)

ด้วยสิ่งต่าง ๆ เหล่านี้ ทำให้จาวาสามารถปล่อยสมรรถนะระดับที่น่าพอใจออกมาได้โดยที่ยังคงไว้ซึ่งความพอร์ดเทเบิล และสภาพแวดล้อมที่ปลอดภัย นอกจากนั้น ยังสามารถจะดึงสมรรถนะระดับสูงสุดออกมาใช้ได้ทันทีเมื่อต้องการ

Java Language Advantages

ภาษาจาวาเป็นภาษาโปรแกรมแบบ OOP (Object Oriented Programming) แต่มีจุดเด่นอยู่ที่ สามารถสร้างโปรแกรมขนาดเล็ก เพื่อใช้งานผ่านเน็ตเวิร์กร่วมกับบราวเซอร์ที่สามารถ อินเตอร์พรีตไบท์โค้ดที่ถูกสร้างขึ้นด้วยจาวาคอมไพเลอร์ได้ เทคนิคการออกแบบของจาวานั้น ไม่ขึ้นอยู่กับสถาปัตยกรรมของฮาร์ดแวร์ดังที่กล่าวมาแล้ว เราเรียกว่า Architecture Neutral ฉะนั้นโปรแกรมเมอร์ที่เขียนภาษาจาวาไม่จำเป็นจะต้องสนใจว่าโปรแกรมของตนจะถูกนำไปใช้บนเครื่องแบบใด บราวเซอร์จะทำหน้าที่แปลไบท์โค้ดให้กับฮาร์ดแวร์เอง ซึ่งถือเป็นกุญแจสำคัญของการออกแบบด้านเทคนิคของจาวา และเป็นเหตุผลสำคัญที่ทำให้จาวาเหมาะกับเว็บ และระบบออนไลน์มากกว่า

โดยเป้าหมายของการฝังภาษาจาวากับเว็บ ก็คือเพื่อให้เนื้อหาของเว็บสามารถเอ็กซีคิวต์ได้ ซึ่งทำให้ผู้ใช้สามารถโต้ตอบ หรือ Interactive ได้ สิ่งที่จำเป็นก็เพียงแต่มีบราวเซอร์ที่สนับสนุนภาษาจาวาติดตั้งอยู่บนเครื่องเท่านั้น และเชื่อมต่อกับอินเตอร์เน็ตก็เพียงพอแล้ว

คุณสมบัติของภาษาจาวาที่โดดเด่น

ภาษาจาวามีคุณสมบัติเด่นดังต่อไปนี้

1. ความง่าย (Simple)

ด้วยโครงสร้างของภาษาที่คล้ายคลึงกับภาษาที่เป็นที่นิยม และคุ้นตาคนทั่วไปอย่างภาษา C และ C++ แต่ได้กำจัดคุณลักษณะที่ไม่จำเป็น และฟุ่มเฟือยของภาษาต้นแบบทิ้งหมด รวมถึงกฎเกณฑ์ข้อบังคับและไวยากรณ์ที่เข้าใจง่ายต่อโปรแกรมเมอร์รุ่นใหม่และรุ่นเก่าที่เชี่ยวชาญอยู่แล้ว

จากที่กล่าวมาข้างต้นแล้วว่า ภาษาจาวาอาศัยภาษา C และ C++ เป็นต้นแบบ แต่ก็ได้ตัดฟีเจอร์ หรือคุณลักษณะหลาย ๆ อย่างทางด้าน Object-Oriented ที่ไม่ค่อยได้ใช้งานออกไป และที่ทำงานซับซ้อนออกไปด้วย เนื่องจากภาษาที่ออกแบบมาให้มีความสามารถสูง มักจะมีปัญหาของฟีเจอร์ที่ซับซ้อนเหล่านั้นได้ เช่น การสร้างโค้ดมักจะมีข้อผิดพลาด (Error) ที่ยากต่อความเข้าใจ มูลค่าของการดูแลระบบซอฟท์แวร์มักเกิดจากการดูแลโค้ดมากกว่าค่าใช้จ่ายในการเขียนขึ้นมาใหม่เสียอีก

ฉะนั้น จาวาจึงถูกออกแบบมาให้มีความแตกต่างจาก C และ C++ ในหลายประการ เช่น

1. จาวาไม่สนับสนุน struct , union และ pointer data types
2. จาวาไม่สนับสนุน typedef หรือ #define
3. จาวามีการจัดการโอเปอเรเตอร์ที่แน่นอน ไม่อนุญาตให้มีการโอเวอร์โหลด
4. จาวาไม่สนับสนุน multi inheritance
5. จาวาจัดการ command line argument แตกต่างจาก C หรือ C++
6. จาวามี string class เป็นส่วนหนึ่งของแพ็คเกจ java.lang ต่างจาก null character array ของ C++ ,C
7. จาวามีระบบจัดสรร และปล่อยหน่วยความจำโดยอัตโนมัติ (garbage collection) ดังนั้นไม่จำเป็นต้องใช้ฟังก์ชัน allocation และ de-allocation function เช่นเดียวกับ C หรือ C-Object-Oriented เช่นเดียวกับ C++ ในแนวทางการเขียนโปรแกรมแบบออบเจ็คท์ ที่สามารถเขียนซอฟท์แวร์คอมโพเนนท์ที่สามารถนำกลับมาใช้ใหม่ได้

2. คุณสมบัติแบบ OOP (Object-Oriented Programming)

จากโครงสร้างภาษาของจาวาไม่ว่าจะเป็นคลาส , method และออบเจ็คท์ หรือการกำหนดชนิดของข้อมูล ตัวแปร และเครื่องหมายทางคณิตศาสตร์ ล้วนแล้วแต่ดั้งอยู่บนพื้นฐานของการเขียนโปรแกรมแบบ OOP ทั้งสิ้น ซึ่ง OOP เป็นรูปลักษณ์การเขียนโปรแกรมยุคใหม่ที่มีประสิทธิภาพในการเขียนและนำไปใช้งานที่ดีกว่าแนวคิดเดิม

องค์ประกอบของข้อมูล และโอเปอเรชัน (operation) ที่เรียกว่า methods (จะกล่าวถึงอีกครั้ง) ที่จะเรียกใช้ข้อมูลเหล่านั้น method เหล่านี้จะผนึกรวมมากับข้อมูล encapsulation นอกจากจะป้องกันการเข้าถึงข้อมูลของออบเจ็คท์โดยตรง เพราะ method เป็นวิธีเดียวที่จะเปลี่ยนสถานะของข้อมูล

หลักการทั่วไปของ Object-Oriented คือ inheritance ออบเจ็คท์ใช้คุณลักษณะจาก ออบเจ็คท์อื่นโดยไม่จำเป็นต้องสร้างฟังก์ชันเหล่านั้นใหม่ ดังนั้น inheritance ช่วยให้ซอฟท์แวร์นำกลับมาใช้ใหม่ได้ ข้อได้เปรียบอีกประการหนึ่งของการจัดการซอฟท์แวร์แบบ inheritance จะถูกจัดการไปตามคลาส หมายถึงแต่ละออบเจ็คท์ในคลาสจะมีคุณลักษณะออบเจ็คท์แม่ (parent objects) ซึ่งจะทำให้สามารถสร้างเอกสารของงาน ทำความเข้าใจ และผลประโยชน์ของซอฟท์แวร์ที่สร้างไว้ก่อนหน้า เพราะฟังก์ชันของซอฟท์แวร์มักจะเป็นการเพิ่มเติมจากออบเจ็คท์ที่สร้างขึ้นก่อนหน้านั้น ออบเจ็คท์ที่อยู่ปลายของ inheritance มักจะมีคุณลักษณะพิเศษและทำงานด้วยความสามารถสูง

3. การไม่ขึ้นกับแพลทฟอร์มและฮาร์ดแวร์ใด ๆ (Platform Independent)

เป็นข้อเด่นที่ทำให้ภาษาจาวาแพร่หลายไปอย่างรวดเร็ว ด้วยคุณสมบัติของภาษาจาวาที่เมื่อเขียนขึ้นแล้วจะถูกคอมไพล์ให้เป็นรูปแบบไบท์โค้ด (Byte Code Format) และเมื่อนำไปใช้งานในแพลท์ฟอร์มใด ๆ ก็จะถูกอ่านและรันโดยอินเตอร์พรีเตอร์ในแต่ละแพลทฟอร์มนั้น ๆ ด้วยคุณสมบัตินี้ ทำให้การใช้งานจาวาไม่ขึ้นกับแพลทฟอร์ม หรือฮาร์ดแวร์ใด ๆ

4. ความปลอดภัย (Safety)

ภาษาจาวานี้ ถูกออกแบบให้สามารถใช้งานได้โดยที่จะไม่ติดไวรัส เพราะจะมีระบบรักษาความปลอดภัยของตัวภาษา ไม่ถูกลบหรือเปลี่ยนแปลงไฟล์ข้อมูล

5. คุณสมบัติ Distributed

ของจาวาก็ต่างจาก C และ C++ จาวาออกแบบมาให้ทำงานในสภาพของเน็ตเวิร์กอยู่แล้ว จาวามีคลาสไลบรารีขนาดใหญ่สำหรับการติดต่อด้วยชุดโปรโตคอล TCP/IP ที่ใช้บนอินเตอร์เน็ต รวมไปถึงโปรโตคอลเฉพาะของอินเตอร์เน็ตอย่าง HTTP และ FTP อีกด้วย ด้วยโค้ดของจาวา โปรแกรมเมอร์สามารถจัดการทรัพยากรผ่าน URLs ได้ง่ายดาย เช่นเดียวกับใช้ C++ และ C จัดการกับโลคอลไฟล์ซิสเต็มส์ (Local File System)

6. Interpreted

เมื่อจาวาคอมไพเลอร์แปลงไฟล์คลาสที่เป็นซอร์สของจาวา ให้อยู่ในรูปไบท์โค้ดแล้ว ไฟล์คลาสในรูปไบท์โค้ดนี้จะสามารถรันอยู่บนเครื่องใดก็ได้ ดังที่กล่าวมาแล้วในตอนต้น ทำให้จาวาไม่ขึ้นอยู่กับฮาร์ดแวร์แพลทฟอร์มใด และแยกคอมไพเลอร์ ออกจากการรันโปรแกรมที่ไคลเอ็นท์ เนื่องจากไบท์โค้ดไม่ได้จำเพาะเจาะจงกับเครื่อง ขอให้มี Virtual Machine ของจาวาก็ใช้ได้แล้ว

7. Robust

ซอฟท์แวร์ที่มีความทนทาน หมายถึงซอฟท์แวร์ที่ไม่ถูกหยุดจากบัก (Bug) ง่ายไปนัก ภาษาโปรแกรมที่ดี จะต้องช่วยให้ผู้ใช้สามารถสร้างโปรแกรมที่มีความทนทานมากขึ้น เช่น ข้อกำหนดในการเขียนโปรแกรม ซึ่งอาจหมายถึง data types และการใช้พอยเตอร์ เป็นที่รู้รับทั่วกันว่าภาษา C ขาดการตรวจสอบความเข้ากันได้ของ data types ขณะคอมไพล์ และรันไทม์ ซึ่งได้พัฒนาขึ้นใน C++ แต่อย่างไรก็ตาม C++ ก็ยังขาด สำหรับจาวา โปรแกรมเมอร์ไม่สามารถกำหนดพอยเตอร์ได้ตามใจชอบ แต่จะบังคับให้ใช้ในลักษณะของอาร์เรย์แทน โปรแกรมที่ใช้ภาษา C อาจคุ้นเคยกับการเรียกใช้หน่วยความจำได้อย่างอิสระ แต่ในจาวา โปรแกรมเมอร์ไม่สามารถ overwrite หรือ corrupt หน่วยความจำที่ใช้กับข้อมูลอื่นได้ก็เพราะจาวาออกแบบให้ทำงานกับ เน็ตเวิร์ก นอกจากนั้นการที่จาวาจำกัดการใช้งานพอยเตอร์ย่อมทำให้ผู้พัฒนาโปรแกรมไม่สามารถเรียกอ่านข้อมูลในหน่วยความจำโดยตรงได้ ช่วยเพิ่มความปลอดภัยในการทำงานบนเน็ตเวิร์ก ซึ่งเป็นสิ่งที่ต้องให้ความสนใจอย่างยิ่ง

8. Architecture-Neutral

จาวาคอมไพเลอร์สร้างไบท์โค้ดที่จะถูกส่งไปยังบราวเซอร์ที่เรียกใช้ และอินเตอร์พรีตบนเครื่องปลายทาง ซึ่งมีบราวเซอร์ หรืออินเตอร์พรีเตอร์ของจาวาติดตั้งอยู่

9. int

แทนอินทีเจอร์ที่เป็น 32 บิต twos' compliment

10. float

แทนจำนวนโฟลทติงพอยต์แบบ 32 บิตเสมอ ตามมาตรฐาน IEEE 754

11. High Performance

เนื่องจากไบท์โค้ดของจาวาจะถูกอินเตอร์พรีต ในบางครั้งจะมีประสิทธิภาพไม่เร็วเท่าการคอมไพล์โดยตรงหรือเอ็กซีคิวต์บนฮาร์ดแวร์เฉพาะการคอมไพล์จาวาประกอบไปด้วยออ

บชั่นให้แปลงไบท์โค้ดไปเป็นไบท์โค้ดของเครื่องสำหรับฮาร์ดแวร์เฉพาะบางเครื่อง ด้วยวิธีนี้จะเพิ่มประสิทธิภาพของจาวา ให้เทียบเท่ากับการใช้คอมไพเลอร์ และโหลดโปรเซสปกติ บริษัท ซันไมโครซิสเต็มส์ ได้ทดสอบประสิทธิภาพของไบท์โค้ดที่แปลงไปเป็นโค้ดของเครื่องกับโปรแกรมภาษา C และ C++ ที่คอมไพล์โดยตรง ปรากฏว่าแทบจะไม่เห็นความแตกต่างเลย

12. Multithreaded

ภาษาจาวาสามารถสร้างแอพพลิเคชั่นที่ทำงานได้หลายอย่างพร้อมกัน เนื่องจากรูทีนของระบบที่อนุญาตให้ทำงานแบบ multiple threads ด้วยจาวา โปรแกรมเมอร์สามารถเขียนโปรแกรมแบบเรียลไทม์และอินเตอร์แอคทีฟกับผู้ใช้ได้

13. Dynamic

ต่างจากโค้ดที่เขียนจาก C++ ที่ต้องคอมไพล์ซ้ำทุกครั้งที่มีการเปลี่ยน parent class จาวาใช้ method และ instance variables ในไลบรารี่ของออบเจ็คท์ได้ใหม่ โดยไม่มีผลต่อไคลเอ็นท์ออบเจ็คท์ที่เกี่ยวข้องกับออบเจ็คท์นั้นเลย

บทที่ 4
การเขียนโปรแกรมด้วยภาษาจาวา

สำหรับในที่นี้จะนำเสนอใน 2 รูปแบบด้วยกัน คือ

1. การเขียนโปรแกรมด้วยภาษาจาวา (Writing the Java Programs)
2. การเขียนแอพเพล็ท (Writing Applets)

การเขียนโปรแกรมภาษาจาวา (Writing Java)

สำหรับการอธิบายหลักการเขียนโปรแกรมด้วยภาษาจาวานี้ ขอแบ่งหัวข้อการอธิบายออกเป็นหัวข้อย่อย ๆ ดังนี้

1. หลักการของการเขียนโปรแกรมแบบ Object-Oriented Programming
2. องค์ประกอบของภาษาจาวา
3. ออบเจ็คท์ (Objects) , คลาส (Classes) และ อินเตอร์เฟซ (Interface) ในภาษาจาวา

1. หลักการของการเขียนโปรแกรมแบบ Object-Oriented Programming

เนื่องด้วยภาษาจาวานี้ อาศัยหลักการเขียนโปรแกรมแบบ Object-Oriented Programming หรือ OOP ซึ่งถือเป็นรูปแบบการเขียนโปรแกรมที่พัฒนาขึ้นมาค่อนข้างใหม่ล่าสุด โดย คุณสมบัติหลัก ๆ ของ OOP และจาวา ได้แก่

1.1 คุณสมบัติออบเจ็คท์

ออบเจ็คท์ของภาษา OOP และจาวานี้ จะประกอบด้วยตัวแปร และ method โดยที่ตัวแปรหมายถึงองค์ ประกอบพื้นฐานของออบเจ็คท์นั้น ๆ เช่น ออบเจ็คท์ของจักรยานก็จะประกอบด้วย เกียร์ , ล้อ เป็นต้น และ method ก็หมายถึง สิ่งที่ออบเจ็คท์สามารถทำได้โดยใช้ตัวแปร

นั้น ๆ เช่น ในกรณีของจักรยาน ก็สามารถเร่งความเร็ว , เปลี่ยนเกียร์ เป็นต้น ซึ่ง method ต่าง ๆ ในที่นี้ เช่น การเปลี่ยนเกียร์ก็ต้องอาศัยตัวแปรที่มีอยู่ก็คือ เกียร์เบอร์ต่าง ๆ

นอกจากนี้เรายังจะสามารถใช้ออบเจ็คท์ในการแทนสิ่งที่ไม่มีตัวตนก็ได้ เช่ นในภา ษา จา วา มีการแทนออบเจ็คท์ในเหตุการณ์ (event) ต่าง เช่น การกดเมาส์ การปล่อยเมาส์ เป็นต้น

สำหรับออบเจ็คท์นี้ จะมีโมเดลที่เป็น method ล้อมรอบตัวแปรเอาไว้ ซึ่งเรียกว่า คุณสมบัติ Encapsulation ซึ่งจะซ่อนรายละเอียดที่ไม่สำคัญของออบเจ็คท์นั้นเอาไว้ นั่นคือ เวลาเราใช้ออบเจ็คท์เราไม่จำเป็นต้องรู้คุณสมบัติทุกอย่าง เพียงแต่เราเลือกใช้ method ให้ถูกกับตัวแปรเท่านั้นก็พอ

และด้วยคุณสมบัติ Encapsulation ของออบเจ็คท์นี้ ทำให้การใช้คุณสมบัติออบเจ็คท์ของจาวานี้ได้เปรียบการเขียนโปรแกรมแบบอื่น ๆ คือ ออบเจ็คท์แต่ละออบเจ็คท์จะเป็นอิสระจากตัวซอร์สโค้ด นั่นคือ เราสามารถนำออบเจ็คท์ไปใช้ที่อื่นก็ได้ หรือนำไปใช้ที่ไหนก็ได้ และออบเจ็คท์ยังทำให้สามารถซ่อนข้อมูลบางอย่างที่ไม่จำเป็นต้องแสดงออกมา ทำให้เราสามารถเปลี่ยนแปลงข้อมูลในส่วนนี้ได้ตลอดเวลาโดยไม่มีผลกระทบต่อการทำงานของ method ปกติ

1.2 คุณสมบัติ message

การที่ออบเจ็คท์แต่ละออบเจ็คท์จะสามารถติดต่อระหว่างกัน หรือการที่เราจะสามารถใช้งาน ออบเจ็คท์แต่ละตัวได้ เราจำเป็นจะต้องมีการส่งสัญญาณบอกให้ออบเจ็คท์นั้นรับรู้ นั่นคือ การส่ง message ให้ออบเจ็คท์รับรู้นั่นเอง ซึ่งการส่ง message นี้จะประกอบด้วย

1. ออบเจ็คท์เป้าหมายที่ต้องการส่งไปให้
2. ชื่อของ method ที่ต้องการใช้
3. ตัวแปรที่ใช้กับ method นั้น ซึ่งเราเรียกว่า parameter

ยกตัวอย่างเช่น การใช้งานรถจักรยาน ออบเจ็คท์เป้าหมายก็คือ จักรยานของเรา ชื่อของ method ในที่นี้สมมติเราต้องการจะเปลี่ยนเกียร์ นั่นคือ เปลี่ยนเกียร์คือชื่อของ method นั่นเอง และพารามิเตอร์ ก็คือ เกียร์ต่ำ นั่นคือ เราต้องการให้รถจักรยานของเราเปลี่ยนเป็นเกียร์ต่ำนั่นเอง

1.3 คุณสมบัติคลาส

สำหรับคุณสมบัติคลาส เปรียบเสมือนเป็นต้นแบบของออบเจ็คท์ในโปรแกรม ที่เราเขียนขึ้นมา เช่น สมมติเราต้องการที่จะใช้ออบเจ็คท์จักรยาน เราก็จะสร้างต้นแบบของจักรยานหรือคลาส ของจักรยานขึ้นมาซึ่งจะทำให้เราสามารถใช้ออบเจ็คท์จักรยานมากเท่าใดก็ได้ โดยใช้ผ่านออบเจ็คท์ และออบเจ็คท์แต่ละตัวที่เราใช้ก็จะเป็นอิสระต่อกัน เปรียบเสมือนกับจักรยานของแต่ละคน แต่ละคนจะเปลี่ยนเกียร์ บางคนจะเบรคหรือ จะเร่งความเร็วก็ไม่ขึ้นต่อกัน

สำหรับองค์ประกอบของคลาส ก็เช่นเดียวกับออบเจ็คท์ ที่จะมีตัวแปร และ method โดยออบเจ็คท์ที่มาจากคลาสนั้น ๆ ก็จะมีตัวแปร และ method เหมือนต้นร่างของมันทุกประการ

1.4 คุณสมบัติ Inheritance

สำหรับคลาสแต่ละคลาส จาวาก็ยอมให้มีการกำหนดเป็นคลาสใหม่ ๆ ที่มาจากคลาส อันเดิม เช่น ในกรณีของจักรยาน เรายังมีจักรยานอื่นย่อย ๆ ไปอีก เช่น จักรยานเสือภูเขา จักรยานจ่ายตลาด ซึ่งก็ล้วนเป็นจักรยานเหมือนกัน มีคุณสมบัติพื้นฐานเหมือนกันแต่อาจจะมีองค์ประกอบย่อย ๆ แตกต่างกัน นั่นคือคลาสใหม่ที่เราสร้างขึ้นมา เราเรียกว่าสับคลาส (Subclass) ซึ่งเราสามารถที่จะเพิ่มตัวแปร หรือ method อื่น ๆ เพิ่มเติมให้กับมันได้นอกเหนือจากคุณสมบัติพื้นฐานที่มันมีอยู่เดิม

โดยคลาสเริ่มต้นที่มีคลาสย่อย ๆ ลงไป เราเรียกว่าซุปเปอร์คลาส (Superclass) และคลาส ย่อย ๆ เราเรียกว่าสับคลาส (Subclass)

โดยคุณสมบัติการส่งต่อจากคลาสบนซุปเปอร์คลาส ไปยังสับคลาสของมันเราเรียกว่าคุณสมบัติ Inheritance

2. องค์ประกอบของภาษาจาวา

สำหรับในหัวข้อนี้ เราจะกล่าวถึงการเขียนโปรแกรมด้วยภาษาจาวา ว่าจะประกอบด้วย องค์ประกอบพื้นฐานอะไรบ้าง ในที่นี้ขอยกตัวอย่างโปรแกรมภาษาจาวาซึ่งเป็นโปรแกรมที่ใช้ในการนับตัวอักษรที่เราใส่เข้าไปดังนี้

```
class Count {
    public static void main(String[] args)
        throws java.io.IOException
    {
        int count = 0;
        while (System.in.read() != -1)
            count++;
        System.out.println("Input has "+ count + " chars.");
    }
}
```

จากตัวอย่างโปรแกรมข้างต้น ประกอบด้วยส่วนต่าง ๆ ดังนี้

2.1 ตัวแปร และ Data Type

ตัวแปร ในภาษาจาวาการประกาศตัวแปรจะมีรูปแบบการประกาศ ที่ประกอบด้วย 2 ส่วนทุกครั้งที่ประกาศ คือประกอบด้วย ชนิดของตัวแปร และชื่อของตัวแปร เช่น จากในตัวอย่างการประกาศตัวแปร คือ int count และ String[] args

สำหรับชนิดของตัวแปร หรือ ข้อมูลในภาษาจาวามี 2 แบบด้วยกัน คือ Primitive type และ Reference type

Primitive type จะเป็นตัวแปรที่ประกอบด้วยค่าเพียงค่าเดียว ซึ่งได้แก่ชนิดของข้อมูลดังต่อไปนี้

ชนิด	ขนาดและรูปแบบ	คำอธิบาย
จำนวนเต็ม		
byte	8 บิต two's complement	เป็นจำนวนเต็มขนาด 1 ไบต์
short	16 บิต two's complement	จำนวนเต็มแบบ short
int	32 บิต two's complement	จำนวนเต็มตามปกติ
long	64 บิต two's complement	จำนวนเต็มแบบ long
จำนวนจริง		
float	32 บิต มาตรฐาน IEEE 754	
double	64 บิต มาตรฐาน IEEE 754	
ชนิดอื่น ๆ		
char	ตัวอักษรขนาด 16 บิต	ตัวอักษรตัวเดียว
boolean	true หรือ false	เป็นค่าบูลีน (จริง หรือ เท็จ)

ตารางที่ 4 - 1 ข้อมูลชนิด Primitive Type ในภาษาจาวา

Reference Type จะเป็นค่าที่จะอ้างอิงไปยังค่าอื่น (ในการเขียนภาษาอื่นอาจใช้พอยเตอร์แทน) ซึ่งเป็นค่าที่แท้จริงของมันซึ่งชนิดของข้อมูลนี้ก็ได้แก่อาร์เรย์ คลาส หรืออินเตอร์เฟซใด ๆ

2.2 Operator

โอเปอร์เรเตอร์ของจาวาอาจจะใช้กับโอเปอร์แรนด์เพียง 1 ตัว หรือ 2 ตัวก็ได้ โอเปอร์เรเตอร์ที่ใช้กับโอเปอร์แรนด์เพียงตัวเดียวเรียกว่า Unary operator โดยการใช้ Unary operator นี้อาจจะใช้นำหน้าโอเปอร์แรนด์ (prefix) หรือไว้หลังโอเปอร์แรนด์ (postfix) ก็ได้ ซึ่งรูปแบบของทั้งสองวิธีเป็นดังนี้

operator op เช่น ++i

op operator เช่น i++

สำหรับโอเปอร์เรเตอร์ที่ต้องการโอเปอร์แรนด์ 2 ตัวเรียกว่า Binary operator โดยการใช้โอเปอร์เรเตอร์นี้จะใช้ระหว่างโอเปอร์แรนด์ทั้งคู่นั้น (infix) ซึ่งรูปแบบเป็นดังนี้

op1 operator op2

โอเปอร์เรเตอร์มีด้วยกันหลายแบบดังนี้

2.2.1 Arithmetic Operator

มีโอเปอร์เรเตอร์ดังต่อไปนี้

โอเปอร์เรเตอร์	การใช้	
+	op1 + op2	การบวก
-	op1 - op2	การลบ
*	op1 * op2	การคูณ
/	op1 / op2	การหาร
%	op1 % op2	การหารแล้วเอาแต่เศษ (modula)

ตารางที่ 4 - 2 Arithmetic Operator ในภาษาจาวา

นอกจากนี้เรายังใช้ + กับ String ด้วย (String Concatenation) ซึ่งจะเห็นจากตัวอย่าง

```
System.out.println("Input has "+ count + " chars.");
```

เครื่องหมาย + และ - ยังใช้เป็น Unary version โดยใช้ในรูปแบบดังนี้

+	+ op	แสดงค่าบวก
-	- op	แสดงค่าลบ

ตารางที่ 4 - 3 การใช้โอเปอเรเตอร์ + และ - ในรูปแบบ Unary version

นอกจากนี้เรายังมี shortcut ที่แสดงการเพิ่มหรือลดค่าเป็นจำนวนแน่นอน ซึ่งมีการใช้ดังนี้

++	op ++	จะบวกโอเปอร์แรนด์ด้วย 1 จะมีการคำนวณโอเปอร์แรนด์ก่อนการเพิ่มค่า
++	++ op	จะบวกโอเปอร์แรนด์ด้วย 1 จะมีการคำนวณโอเปอร์แรนด์หลังการเพิ่มค่า
--	op --	จะลบโอเปอร์แรนด์ด้วย 1 จะมีการคำนวณโอเปอร์แรนด์ก่อนการเพิ่มค่า
--	-- op	จะลบโอเปอร์แรนด์ด้วย 1 จะมีการคำนวณโอเปอร์แรนด์หลังการเพิ่มค่า

ตารางที่ 4 - 4 การใช้ Shortcut ในการเพิ่มหรือลดค่าโอเปอร์แรนด์

2.2.2 Relational and Conditional Operator

จะเป็นการใช้โอเปอร์เรเตอร์เพื่อเปรียบเทียบค่า 2 ค่า เพื่อตัดสินค่าความสัมพันธ์ของ 2 ค่านั้น โดยค่าที่คืนมาจะเป็นเท็จหรือจริง โดยมีการใช้ดังนี้

โอเปอร์เรเตอร์	การใช้	จะคืนค่าเป็นจริงถ้า
>	op1 > op2	op1 มากกว่า op2
>=	op1 >= op2	op1 มากกว่าหรือเท่ากับ op2
<	op1 < op2	op1 น้อยกว่า op2
<=	op1 <= op2	op1 น้อยกว่าหรือเท่ากับ op2
==	op1 == op2	op1 และ op2 เท่ากัน
!=	op1 != op2	op1 และ op2 ไม่เท่ากัน

ตารางที่ 4 - 5 Relational Operator ของจาวา

ซึ่งยังมี Condition operator ที่ใช้กัน 3 แบบด้วยกัน คือ

&&	op1 && op2	op1 และ op2 จริงทั้งคู่
||	op1 || op2	op1 หรือ op2 เป็นจริง
!	! op	op เป็นเท็จ

ตารางที่ 4 - 6 Condition Operator ของจาวา

2.2.3 Bitwise Operator

จะเป็นการทำงานกับค่าข้อมูลในระดับบิต ซึ่งมีการใช้ดังนี้

โอเปอร์เรเตอร์	การใช้	การทำงาน
>>	op1 >> op2	ชิพค่าบิตของ op1 ไปทางขวาเป็นจำนวน op2
<<	op1 << op2	ชิพค่าบิตของ op1 ไปทางซ้ายเป็นจำนวน op2
>>>	op1 >>> op2	ชิพค่าบิตของ op1 ไปทางขวาเป็นจำนวน op2 (ไม่คิดเครื่องหมาย)
&	op1 & op2	กระทำ and ทีละบิตของ op1 และ op2
|	op1 | op2	กระทำ or ทีละบิตของ op1 และ op2
^	op1 ^ op2	กระทำ xor ทีละบิตของ op1 และ op2
~	~ op	กระทำ complement ของ op

ตารางที่ 4 - 7 Bitwise Operator ของจาวา

สำหรับการชิพของโอเปอร์เรเตอร์ เช่น

```
13 >> 1
```

หมายความว่าชิพค่า 13 ไปทางขวา 1 ตำแหน่ง จาก 13 สามารถเขียนในแบบไบนารีได้เป็น 1101 เมื่อเราชิพไปทางขวา ก็จะได้ 110 หรือ 6 ในฐานสิบนั่นเอง

2.2.4 Assignment Operator

เราจะใช้เครื่องหมาย = เพื่อเป็นการให้ค่า (Assignment) แก่ตัวแปร ซึ่งจากตัวอย่างเราจะเห็นประโยคดังนี้

```
int count = 0;
```

แต่ในจาวา เรายังให้การให้ค่าในกรณีดังนี้ สมมติเราต้องการเพิ่มค่าให้กับตัวแปรหนึ่งแล้วคืนค่านั้นกลับสู่ตัวแปรนั้น เช่น

```
i = i + 2;
```

เราสามารถใช้ Shortcut โดยการใช้เครื่องหมาย += แทน ซึ่งจากตัวอย่างข้างต้นก็สามารถแทนได้ด้วย

```
i += 2;
```

นอกจากนี้ยังมีเครื่องหมายอื่น ๆ อีกดังนี้

โอเปอร์เรเตอร์	การใช้	มีความหมายเท่ากับ
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2
<<=	op1 <<= op2	op1 = op1 << op2
>>=	op1 >>= op2	op1 = op1 >> op2
>>>=	op1 >>>= op2	op1 = op1 >>> op2

ตารางที่ 4 - 8 Shortcut ในการใช้งานโอเปอร์เรเตอร์ต่าง ๆ

2.3 Expression

expression ก็คือลำดับของตัวแปร , โอเปอร์เรเตอร์ และ method ที่เขียนเรียงกันเป็นลำดับที่ถูกต้อง และมีความหมายตามไวยากรณ์ของภาษา โดยผลของ expression นี้จะได้ออกมาเป็น ค่า ๆ เดียว ตัวอย่างของ expression เช่น

count++;

จากในตัวอย่างโปรแกรมข้างต้น

หรือ System.in.read() != -1

จะเป็น expression ที่ประกอบด้วย expression 2 expression คือ System.in.read() และ System.in.read() != -1 ที่ใช้เครื่องหมาย !=

สำหรับใน expression โดยปกติจะมีการกระทำตามเครื่องหมายต่าง ๆ ที่ปรากฏโดยพิจารณาตามความสำคัญ (Precedence) ของมัน โดยตารางที่ 4 - 9 จะแสดงถึงลำดับความ

สำคัญ ของแต่ละเครื่องหมาย และสำหรับเครื่องหมายที่อยู่ในบรรทัดเดียวกันจะมีค่าความสำคัญเท่ากัน แต่ถ้าปรากฏพร้อมกันจะคำนวณจากซ้ายไปขวา

postfix operators	[] . (params) expr++ expr--
unary operators	++expr --expr +expr -expr ~ !
creation or cast	new (type)expr
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instaceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
conditional	? :
assignment	= += -= *= /= %= ^= &= |= <<= >>= >>>=

ตารางที่ 4 - 9 ลำดับความสำคัญของโอเปอร์เรเตอร์ต่าง ๆ

2.4 Control Flow Statements

เป็นการกำหนดเงื่อนไขการเดินหน้าของโปรแกรมโดยการทำในแต่ละรูปแบบดังนี้

รูปแบบ	*คีย์เวิร์ด*
การตัดสินใจ	if-else, switch-case
ลูป	for, while, do-while
การยกเว้น	try-catch-finally, throw
อื่น ๆ	break, continue, label: , return

ตารางที่ 4 - 10 รูปแบบการกำหนดเงื่อนไขการทำงาน

การใช้ if-else

เราสามารถเขียนรูปแบบการใช้ if-else ได้ง่าย ๆ ดังนี้

```
if (expression)
        statement1
else  statement2
```

โดยโปรแกรมจะทำตาม statement1 ก็ต่อเมื่อ expression เป็นจริง(True) แต่ถ้าไม่จริง ก็จะทำตาม statement2 ของ else แต่ตามปกติก็ไม่จำเป็นต้องมี else ทุกครั้งก็ได้ นอกจากนี้เรายังสามารถใช้ประโยคที่มีการกำหนดเงื่อนไข if-else ที่ย่อยเข้าไปมาก ๆ ได้ (Nested) เช่น

```
if (exp1)
stat1
else if (exp2)
   stat2
else if (exp3)
   stat3
else ...
```

การใช้ switch

เป็นการใช้เงื่อนไขที่จะกระทำตามเงื่อนไขใด ๆ โดยขึ้นกับเงื่อนไขที่กำหนด ที่มีหลายเงื่อนไข ดังตัวอย่างรูปแบบประโยค

```
int month;
int numDays;
...
switch (month) {
case 1 : System.out.println("January"); break;
case 2 : numDays = 31;
case 3 : if ( ((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0) )
        numDays = 29;
    else
        numDays = 28;
    break;
}
```

จากตัวอย่างประโยคข้างต้นจะมีความหมายเดียวกับ

```
int month;
...
if (month == 1) {
 System.out.println("January");
} else if (month == 2) {
 ...
```

และเราจะสังเกตว่า เราสามารถกำหนดเป็นเงื่อนไขย่อย ๆ ลงในเงื่อนไขเหล่านี้ได้เช่นกัน

Loop Statements

การใช้ for Loop มีรูปแบบการเขียนง่าย ๆ ดังนี้

```
for (initialization; termination; increment)
    statements
```

นั่นคือ โปรแกรมจะทำตาม statements ไปเรื่อย ๆ โดยเริ่มต้นกำหนดค่าเริ่มต้นที่ initialization จนกระทั่งถึงจุดปลาย (termination) โดยขณะที่ทำตาม statements จนจบก็จะกลับมาตรวจดูค่าเงื่อนไขที่กำหนดแต่แรกเสมอ จากนั้นจะ increment ตามที่กำหนด เช่น

```
....
int i;
int length = a.length;
for (i = 0; i < length; i++) {
  ....
}
```

จากตัวอย่างข้างต้นเริ่มต้นให้ i=0 จากนั้นจะเพิ่ม i ขึ้นทีละหนึ่งจนกระทั่ง i < length

การใช้ do-while Loop

มีรูปแบบการใช้ดังนี้

```
do {
    statements
} while (booleanExpression);
```

นั่นคือ โปรแกรมจะยังคงทำตาม statements ตราบใดที่ค่าใน (Boolean Expression) ยังคงเป็นจริงตามเงื่อนไขที่กำหนด สำหรับตัวอย่างการใช้ while loop จะดูได้ตามตัวอย่างซึ่งใช้ดังนี้

```
while (System.in.read() != -1)
    count++;
System.out.println("Input has "+ count + " chars.");
```

Exception Hadling Statement

เป็นการตรวจสอบเมื่อเกิดกรณีที่ผิดไวยากรณ์ขึ้น หรือเป็นการยกเว้นในโปรแกรมซึ่งจะกล่าวถึงอีกครั้งในตอนหลัง

Branching statement

จะเป็นการใช้ break โดยเมื่อพบเครื่องหมาย break ในโปรแกรมเมื่อใด โปรแกรมก็จะหยุดแล้วกลับไปยัง statement ถัดจาก statement ปัจจุบันทันที นอกจากนี้เรายังจะสามารถใช้ break เพื่อใช้กระโดด (jump) ไปยังจุด หรือ statement ที่เราใส่ชื่อไว้ (label) โดยหลักการใส่ชื่อทำได้ดังตัวอย่าง สมมติต้องการตั้งชื่อ statements ว่า breakToHere ก็สามารถทำได้ดังนี้

breakToHere: statements...

เมื่อเราต้องการกระโดดไปยัง statements เหล่านั้นก็สามารถทำได้ดังนี้

break breakToHere;

การใช้ return จะเป็นการใช้เพื่อออกจาก method ปัจจุบัน และกระโดดกลับไปสู่ statement ถัดจาก method ที่เรียกใช้นั้น สำหรับการใช้ return นี้ เราสามารถใช้เพื่อคืนค่าหรือไม่คืนค่าก็ได้ สำหรับกรณีที่ต้องการคืนค่ากลับไป ให้ใส่ค่านั้นที่ท้ายของ return ดังตัวอย่าง

```
return ++count;
```

สำหรับกรณีที่ไม่ต้องการคืนค่าใด ๆ เพียงแต่ต้องการกลับไปสู่การเรียกใช้ method ก็เพียงแต่ใช้ return เฉย ๆ ดังนี้

```
return;
```

2.5 การใช้อาร์เรย์และสตริง

การใช้อาร์เรย์

การใช้อาร์เรย์ก็เช่นเดียวกันกับการใช้ตัวแปรทั่ว ๆ ไป นั่นคือ ก่อนใช้จะต้องมีการประกาศ อาร์เรย์นั้นก่อน โดยรูปแบบการประกาศจะใช้ ชนิดของตัวแปร ทั่ว ๆ ไป โดยบอกให้รู้ว่าเป็นอาร์เรย์ชนิดใด แต่จุดที่แตกต่างจากตัวแปรทั่ว ๆ ไป คือ จะต้องบอกให้รู้ว่าเป็นอาร์เรย์ โดยใช้เครื่องหมาย [] เช่น

```
int[] arrayOfInts;
```

เป็นการประกาศสร้างตัวแปร arrayOfInts ว่าเป็นอาร์เรย์ของ int หลังจากการประกาศแล้ว ก็จะต้องใช้โอเปอร์เรเตอร์ new เพื่อที่จะให้มีการ allocate เนื้อที่ในหน่วยความจำให้กับอาร์เรย์นั้น ๆ (ตอนแรกที่ประกาศอาร์เรย์ จะยังไม่มีการ allocate เนื้อที่หน่วยความจำแต่อย่างใด) ดังนี้

```
int[] arrayOfInts = new int[10]
```

สำหรับรูปแบบการประกาศดังข้างต้นจะเป็นดังนี้

```
elementType[] arrayName = new elementType[arraySize]
```

ในการใช้ new จะมีการกำหนดขนาดของอาร์เรย์นั้นให้ด้วย และเมื่อเราใช้โอเปอร์เรเตอร์ new เรียบร้อยแล้ว เราก็สามารถให้ค่า (assign) กับอาร์เรย์เหล่านั้นได้แล้ว ดังตัวอย่างการให้ค่าอาร์เรย์ข้างล่างนี้

```
for (int j = 0; j < arrayOfInts.length; j++) {
    arrayOfInts[j] = j;
    System.out.println("[j] = " + arrayOfInts[j]);
}
```

นอกจากนี้ อาร์เรย์ยังสามารถเก็บออบเจ็คท์หรืออาร์เรย์ด้วยกันเองได้ด้วย

การใช้สตริง

การใช้สตริงสามารถใช้ได้ 2 รูปแบบได้แก่

String[] args เป็นการใช้โดยการประกาศให้รู้ว่าเป็นสตริง และมีการกำหนดชื่อของสตริงนั้น ซึ่งจากตัวอย่างก็คือ args นั่นเอง

สำหรับอีกวิธีหนึ่งก็คือ การใช้เครื่องหมายอัญประกาศ เช่น "Input has " ในกรณีนี้จะการกำหนดขนาดของสตริงแน่นอน

นอกจากนี้แต่ละสตริงยังสามารถนำมารวมกันโดยใช้เครื่องหมาย + ซึ่งเรียกว่า Concatenation ดังที่ได้กล่าวมาข้างต้น ซึ่งในตัวอย่างก็ปรากฏดังนี้

System.out.println("Input has "+ count + " chars.");

2.6 องค์ประกอบอื่น ๆ

การใช้ main() Method

เราจะต้องเรียกใช้ main() method ทุกครั้ง เพื่อบอกการเริ่มโปรแกรม จากตัวอย่างข้างต้น จะเห็นการใช้ของ main() method ดังนี้

public static void main(String[] args)

จะเห็นว่า main() method ถูกใช้ก่อน method อื่น ๆ ภายหลังจากประกาศคลาสขึ้นมาแล้ว สำหรับทุก ๆ โปรแกรมที่เขียนด้วยภาษาจาวา main() method จะต้องมีรูปแบบดังข้างต้น ซึ่งประกอบด้วยส่วนต่าง ๆ ดังนี้

- **public** บอกให้รู้ว่า main() method สามารถเรียกโดยออบเจ็คท์ใด ๆ ก็ได้
- **static** บอกให้รู้ว่า main() method เป็น method ของคลาส
- **void** บอกให้รู้ว่าจะไม่มีการคืนค่าใด ๆ ออกมา

เมื่อเริ่มต้นการใช้โปรแกรมใด ๆ จะเริ่มต้นด้วยการเรียก main() method จากนั้น main() method จะทำหน้าที่ในการเรียกใช้ method อื่น ๆ ต่อไป

- **String[] args** เป็นการแสดงให้เห็นถึงการรับค่าข้อมูลเข้าไปยังแอพพลิเคชั่นนั้น ๆ โดยที่ สตริงดังกล่าวนี้เราเรียกว่า command-line argument สำหรับเรื่องของ command-line argument จะกล่าวถึงในตอนหลัง

การใช้ Exceptions

จากตัวอย่างโปรแกรมข้างต้น เราจะเห็นประโยคหนึ่งที่ว่า

throws java.io.IOException

ซึ่งในการทำงานในโปรแกรม บางครั้งเราจะเกิดกรณีที่ทำให้เกิดการหยุดของโปรแกรม หรือเกิดความไม่ต่อเนื่องของโปรแกรม เช่น กรณีหาร 7 ด้วย 0 เป็นต้น ซึ่งกรณีดังกล่าว จาวาจะมี throw ซึ่งจะช่วยทำให้ Exception ดังกล่าวสามารถละทิ้งไปเพื่อให้เกิดความต่อเนื่องของโปรแกรมขึ้น

การใช้ Input และ Output Stream

จากตัวอย่างจะมีประโยคดังนี้

System.in.read() != -1

ซึ่งเป็นการอ่านตัวอักษรเข้ามาจากอินพุท โดยถ้าไม่มีตัวอักษรใดให้อ่านเลย จะให้ค่า -1 ออกมา ฉะนั้น เราจึงเห็นประโยคตัวอย่างกำหนดให้ค่าที่อ่านมาไม่เท่ากับ -1 จึงจะนับ

และประโยค

System.out.println("Input has "+ count + " chars.");

เป็นการพิมพ์ข้อความออกสู่หน้าจอ โดยหลังจากพิมพ์เสร็จจะขึ้นบรรทัดใหม่ให้ด้วย ซึ่งยังมี System.out.print() ซึ่งจะพิมพ์ข้อความออกสู่หน้าจอ โดยที่จะไม่ขึ้นบรรทัดใหม่ สิ่งที่เราสังเกตเห็นจากสอง statement ข้างต้นก็คือ in และ out เราจะเห็นว่า in หมายถึงรับเข้ามาสู่โปรแกรม และ out หมายถึงออกจากโปรแกรมไป

3. ออบเจ็คท์ , คลาส และ อินเตอร์เฟซ ในภาษาจาวา

จากที่เราได้กล่าวถึงเรื่องของ OOP ไว้ในตอนต้นแล้ว ในตอนนี้เรากล่าวถึงลักษณะการเขียนแบบ OOP ของจาวา ไม่ว่าจะเป็นคุณสมบัติของออบเจ็คท์ หรือคลาส ก็ตาม

3.1 ออบเจ็คท์

ออบเจ็คท์ของจาวา ก็เช่นเดียวกับที่กล่าวมาในเรื่องของ OOP คือ เราจะมีตัวแปรและ method ที่แสดงถึงลักษณะของออบเจ็คท์ต่าง ๆ โดยการใช้งานออบเจ็คท์จะมีขั้นตอน หรือ ที่เรียกว่าวงจรชีวิตของออบเจ็คท์ 3 ขั้นตอนด้วยกันดังนี้

1. การสร้างออบเจ็คท์
2. การใช้ออบเจ็คท์
3. การกำจัดออบเจ็คท์

การสร้างออบเจ็คท์

ดามปกติ เราจะเห็นการสร้างออบเจ็คท์ด้วยประโยคพื้นฐานดังนี้

```
Date today = new Date();
```

โดยที่ก่อนจะสร้างออบเจ็คท์ขึ้นมาได้นี้ จะต้องมีคลาสเริ่มต้นก่อน ซึ่งในที่นี้ก็คือคลาส Date จากประโยคข้างต้น เมื่อเรามองดี ๆ แล้วจะเห็นว่า ประกอบด้วยการกระทำ 3 อย่าง ด้วยกัน คือ declaration, Instantiation และ initiallization

Declaration

โดยทั่วไป การประกาศของออบเจ็คท์ จะเหมือนกับการประกาศของตัวแปร ทั่ว ๆ ไป ดังตัวอย่างนี้

Date today;

นั่นคือจะมีรูปแบบดังนี้

Type name

ซึ่งในจาวาแล้ว สิ่งที่ถือว่าเป็นชนิด (Type) ของออบเจ็คท์ก็ได้แก่ทั้งคลาส และอินเตอร์เฟซ ซึ่งจะกล่าวถึงในตอนหลัง

Instantiating an Object

จะเป็นขั้นตอนที่ใช้โอเปอร์เรเตอร์ new เพื่อสร้างออบเจ็คท์ที่ถือกำเนิดจากคลาส โดยจะมีการ allocate เนื้อที่ในหน่วยความจำในขั้นตอนนี้จากโอเปอร์เรเตอร์ดังกล่าว เช่น

new Rectangle(0, 0, 100, 200);

ซึ่งเป็นการสร้างสี่เหลี่ยมขึ้นมาจากคลาส Rectangle

Initializing an Object

เป็นการกำหนดค่าเริ่มต้นให้แก่ออบเจ็คท์ เช่นในตัวอย่างการสร้างออบเจ็คท์ของสี่เหลี่ยมเป็นดังนี้

new Rectangle(0, 0, 100, 200);

ตัวเลขข้างหลัง ก็คือ การกำหนดค่าเริ่มต้นนั่นเอง

ตัวอย่างของการสร้างออบเจ็คท์อีกตัวอย่างหนึ่งก็คือ

Date MyBirthday = new Date(1963, 8, 30);

การใช้ออบเจ็คท์

การใช้ออบเจ็คท์ในที่นี้ก็คือ การกระทำกับตัวแปรต่าง ๆ นั่นเอง ในที่นี้ เราจะมาดูการกระทำกับตัวแปรในรูปแบบต่างๆ กันก่อนสำหรับในการแสดงการใช้งาน เราจะอ้างอิงจากออบเจ็คท์ของสี่เหลี่ยม (rectangle) ที่เราสร้างขึ้นเป็นตัวอย่างข้างต้น

การอ้างอิงตัวแปรของออบเจ็คท์

การอ้างอิงตัวแปรแต่ละตัวของออบเจ็คท์จะใช้รูปแบบดังนี้

objectReference.variable

นั่นคือ เราจะใช้เครื่องหมาย . คั่นระหว่างชื่อของออบเจ็คท์นั้นกับตัวแปร สมมติเรามี rect เป็นชื่อของสี่เหลี่ยม และมีตัวแปร x และ y การอ้างถึงแต่ละตัวแปร ก็แทนได้ดังนี้ rect.x และ rect.y เมื่อเราต้องการเปลี่ยนแปลงค่าต่าง ๆ ของตัวแปร ก็ทำได้เหมือนกับเป็นตัวแปร ทั่ว ๆ ไป อย่างในกรณีนี้เราต้องการจะเปลี่ยนจุดของสี่เหลี่ยมใหม่ ก็ทำได้ดังนี้

```
rect.x = 15;
rect.y = 37;
```

นอกจากนี้ การให้ค่า (assignment) การใช้เครื่องหมายต่าง ๆ (operator) ที่กล่าวมาข้างต้นสามารถมาใช้กับออบเจ็คท์นี้ทั้งหมด อย่างเช่นในกรณีนี้ สี่เหลี่ยมมีส่วนกว้างและยาว ให้แทนด้วย rect.width และ rect.height ฉะนั้นเราก็สามารถหาค่าของพื้นที่ของสี่เหลี่ยมนี้ได้ดังนี้

```
area = rect.height * rect.width;
```

การเรียกใช้ method ของออบเจ็คท์

ก็เช่นเดียวกับการอ้างอิงตัวแปร นั่นคือจะใช้รูปแบบดังนี้

objectReference.methodName(argumentList); หรือ
objectReference.methodName();

อย่างเช่นต้องการจะย้ายสี่เหลี่ยมไปยังจุดใหม่ ก็สามารถใช้ method ชื่อ move() ได้

rect.move(244,24);

การกำจัดออบเจ็คท์ที่ไม่ต้องการ

เราจะใช้ finalize() ซึ่งเป็น method ที่ใช้เพื่อทำการจบ หรือกำจัดออบเจ็คท์ที่ไม่จำเป็นต้องใช้อีกแล้ว เราเรียกว่า finalization ซึ่งทำให้เรามีเนื้อที่ว่างมากขึ้น

3.2 Classes

จากที่เราได้เคยกล่าวมาข้างต้นว่าคลาสทำหน้าที่เป็นต้นแบบ หรือ โปรโตไทป์ของออบเจ็คท์ ในการใช้งานคลาส จะประกอบด้วย 2 องค์ประกอบด้วยกัน คือ

1. ส่วนประกาศคลาส (Class Declaration)
2. ตัวคลาส (Class body)

นอกจากนี้ในการใช้งานคลาส เรายังมีองค์ประกอบอื่น ๆ ที่ต้องพิจารณาอีกมาก ซึ่งจะกล่าวเป็นลำดับ ๆ ดังนี้

การประกาศคลาส (Class declaration)

สำหรับการประกาศคลาสพื้นฐานจะประกอบด้วยคลาส ซึ่งเป็นคีย์เวิร์ดบอกให้รู้ว่าเป็นคลาส และชื่อของคลาสนั้น เพราะฉะนั้นเราอาจจะเขียนรูปแบบพื้นฐานของการประกาศออกมาได้ดังนี้

```
class NameOfClass {
    ...
}
```

สำหรับหลักการประกาศคลาส ก็มีดังนี้

1. ชื่อของคลาส ต้องขึ้นต้นด้วยตัวใหญ่
2. บอกให้รู้ว่าซุปเปอร์คลาสนั้นคืออะไร

ตามปกติ ทุก ๆ คลาส จะต้องมีซุปเปอร์คลาสของมันเอง ซึ่งถ้าเราไม่ประกาศ โปรแกรมจะคาดว่ามันคือออบเจ็คท์คลาส (Object class)

สำหรับการบอกถึงซุปเปอร์คลาส ทำได้โดยการใส่คีย์เวิร์ด extend บวกกับชื่อของ ซุปเปอร์คลาส โดยเขียนอยู่ระหว่างชื่อคลาสนั้น ๆ และเครื่องหมายวงเล็บปีกกาเปิด({) ซึ่งเราสามารถเขียนรูปแบบของมันได้ดังนี้

```
class NameOfClass extends SuperClassName {
    ...
}
```

การใส่อินเตอร์เฟซที่คลาสนั้นใช้อินเตอร์เฟซเป็นการประกาศเซ็ทของ method และค่าคงที่ โดยที่เราไม่จำเป็นต้องบอกถึงรายละเอียด เนื่องจากมีการกล่าวรายละเอียดอยู่ในอินเตอร์เฟซนั้นแล้ว เมื่อใดที่เราอ้างถึง method ใด ๆ ก็เป็นการอ้างอิงไปถึงอินเตอร์เฟซนั้นเอง

สำหรับรูปแบบ เราจะใช้คีย์เวิร์ด implements บวกกับชื่อของอินเตอร์เฟซนั้น ๆ เช่น ตัวอย่างข้างล่างนี้ ImaginaryNumber เป็นคลาสซึ่งมีซุปเปอร์คลาสชื่อ Number และใช้อินเตอร์เฟซชื่อ Arithmetic ซึ่งในอินเตอร์เฟซนี้จะมี method ชื่อ add() และ substract() ฉะนั้นเราสามารถเขียนแทนได้ดังนี้

```
class ImaginaryNumber extends Number implements Arithmetic {
    ...
}
```

สรุปการประกาศคลาส จะมีรูปแบบคร่าว ๆ ดังนี้

```
[modifiers] class ClassName [ extends SuperClassName ] [ implements InterfaceNames ] {
    ...
}
```

สิ่งที่อยู่ในเครื่องหมาย [] หมายถึงว่าจะมีหรือไม่ก็ได้

1. modifier จะประกาศให้รู้ว่าคลาสนี้เป็น public abstract หรือ final
2. ClassName จะเป็นชื่อของคลาส
3. SuperClassName เป็นชื่อของซุปเปอร์คลาสของ ClassName นั้น ๆ

4. InterfaceName เป็นชื่อของอินเตอร์เฟซที่ใช้ในกรณีที่มีมากกว่า 1 ให้ใส่เครื่องหมาย ,

ในกรณีที่เราไม่ได้ใส่ส่วนที่อยู่ใน [] คอมไพเลอร์ของจาวาจะกำหนดให้เป็น non-final , non-public , non-abstract , สับคลาส ของออบเจ็คท์ และไม่มีอินเตอร์เฟซใด ๆ

ตัวคลาส (Class Body)

ตามรูปแบบของโปรแกรม ตัวคลาสจะอยู่ในส่วนในของโปรแกรม ดังข้างล่างนี้

```
classDeclaration {
    classBody
}
```

สำหรับองค์ประกอบของตัวคลาสนี้มี 2 ส่วนใหญ่ ๆ ก็คือ ตัวแปร และ method ซึ่งตามปกติเรามักจะนิยมประกาศตัวแปรก่อน แล้วตามด้วยการประกาศ method ดังรูปแบบ ข้างล่างนี้

```
classDeclaration {
    memberVariableDeclarations
    methodDeclarations
}
```

ซึ่งข้างล่างนี้เราจะแสดงถึงตัวอย่างการประกาศตัวแปร และ method ดังนี้

```
class TicketOuttaHere {
    Float price;
    String destination;
    Data departureDate;
    void signMeUp(Float forPrice, String forDest, Date forDate) {
        price = forPrice;
        destination = forPrice;
        departureDate = forDate;
    }
}
```

การประกาศสมาชิกของตัวแปร (Declaring Member Variables)

การประกาศตัวแปร ตามปกติจะประกอบด้วย 2 องค์ประกอบย่อย คือ ชนิดของข้อมูล และ ชื่อของตัวแปรนั้น ดังรูปแบบของการประกาศจะเป็นดังนี้

type variableName;

สำหรับการตั้งชื่อตัวแปร จะต้องขึ้นต้นด้วยตัวเล็ก และห้ามมีตัวแปรใด ๆ มีชื่อเหมือนกัน แต่ชื่อของตัวแปรกับ method อาจจะมีชื่อเดียวกันได้ แต่จริง ๆ แล้ว การประกาศตัวแปร จะมีองค์ประกอบอย่างเต็มยศดังนี้

[accessSpecifier] [static] [final] [transient] [volatile] type variableName

ตัวที่อยู่ใน [] หมายถึง จะใส่หรือไม่ก็ได้ สำหรับองค์ประกอบต่าง ๆ นี้ สามารถอธิบายได้ดังนี้

1. accessSpecifier เป็นการกำหนดควบคุมการเข้าใช้ที่ อค วบคุ มารเข้ ถึ ตั ฉปร หรือ method ว่าได้หรือไม่
2. static จะแสดงให้เห็นว่าตัวแปร หรือ method นั้น ๆ เป็น class member variable ซึ่งตรงกันข้ามกับ instance member variable และ class method ตามลำดับ
3. final บอกให้รู้ว่าตัวแปรนี้เป็นค่าคงที่
4. volatile แสดงให้เห็นว่าตัวแปร สามารถเปลี่ยนแปลงได้แบบ Asynchronous

การใช้ method

การใช้งาน method ประกอบด้วย 2 ส่วนเช่นกัน คือ การประกาศ method และ method body โดยมีรูปแบบการใช้งานดังนี้

```
methodDeclaration {
    methodBody
}
```

การประกาศ method (The Method Declaration)

ส่วนนี้จะให้ข้อมูลเกี่ยวกับ method ทุกอย่างแก่คอมไพเลอร์ ไม่ว่าจะเป็น ชื่อ , ชนิดของการคืนค่าของ method ดังตัวอย่าง method ข้างล่างนี้ซึ่งชื่อว่า isEmpty() โดยจะคืนค่าบูลีนออกมา

```
class Stack {
    ...
    boolean isEmpty() {
        ...
    }
}
```

การคืนค่าของ method เป็นสิ่งจำเป็นมาก และ method จะต้องคืนค่าในรูปแบบใดรูปแบบหนึ่ง ถ้าไม่มีจริง ๆ ก็คืนค่า void ซึ่งหมายถึงไม่มีการคืนค่ากลับมา

สำหรับการตั้งชื่อของ method ก็มีสิ่งที่จะต้องคำนึงถึงด้วยเช่นกัน

1. method สามารถตั้งชื่อซ้ำ ๆ กันได้ แต่ผ่านค่าพารามิเตอร์ที่แตกต่างกันในแต่ละ method ก็ถือว่าแตกต่างกัน ดังตัวอย่าง

```
class DataRenderer {
    void draw(String s) {
        ...
    }
    void draw(int i) {
        ...
    }
    void draw(float f) {
        ...
    }
}
```

ซึ่ง draw() แต่ละตัวจะแตกต่างกัน

2. method ที่มีชื่อเดียวกับคลาสของมันจะเป็นคอนสตรักเตอร์ (Constructor) และใช้ในการ initialize ออบเจ็คท์ใหม่ของคลาส คอนสตรักเตอร์จะกล่าวถึงภายหลัง
3. คลาสสามารถจะนำ method นี้ไปใช้ในซุปเปอร์คลาสของมันได้ โดยมีคุณสมบัติเหมือนเดิม ทุกประการ

นอกจากการประกาศ method แบบพื้นฐานแล้ว ยังมีการประกาศ method ที่เป็นขั้น advance ดังนี้

```
[accessSpecifier] [static] [abstract] [final] [native] [synchronized] returnType
methodName ([paramlist]) [throws exceptionsList]
```

การส่งผ่านข้อมูลลงไปใน method

การส่งผ่านข้อมูลเข้าสู่ method จะอยู่ในรูปของอาร์กูเมนต์ ดังตัวอย่างข้างล่างนี้

```
double computePayment(double loanAmt, double rate, double futureValue, int
numPeriods) {
    double I, partial1, denominator, answer;
    I = rate / 100.0;
    partial1 = Math.pow((1+I), (0.0 - numPeriods));
    donominator = (1 - partial1) / I;
    answer = ((-1 * loanAmt) / denominator) - ((futureValue * partial1) / denominator)
    return answer;
}
```

ในที่นี้จะมี 4 อาร์กูเมนต์ คือ loan amount, interest rate, future value และ number of period ในกรณีนี้มีอาร์กูเมนต์มากกว่า 1 เราจะใช้เครื่องหมาย , ช่วย สำหรับการประกาศจะเห็นว่าเป็นไปตามรูปแบบง่าย ๆ ดังนี้

type name

Method Body

ใน body ของ method เราอาจจะประกาศค่าตัวแปร ซึ่งในที่นี้จะเป็นตัวแปรโลคอล (Local Variable) ซึ่งจะมีผลแค่เพียงใน method นั้นเท่านั้น เมื่อใดที่เราออกจาก method ค่าตัวแปรก็จะถูกยกเลิกไป ดังตัวอย่าง

```
Object findObjectInArray(Object o, Object[] arrayOfObjects) {
    int i;      // local variable
    for (i = 0; i < arrayOfObjects.length; i++) {
        if (arrayOfObjects[i] == 0)
            return o;
    }
    return null;
}
```

หลังจากคืนค่าของ method เรียบร้อยแล้ว i ก็จะไม่มีอยู่อีกต่อไป

การควบคุมการเข้าสู่สมาชิกของคลาส

สิ่งนี้ถือได้ว่าเป็นข้อได้เปรียบของคลาส ที่สามารถควบคุมและป้องกันการเข้าสู่การใช้งาน ตัวแปร และ method จากออบเจ็คท์อื่น ซึ่งถือว่ามีความสำคัญมาก เนื่องจากจะทำให้เราสามารถเพิ่มความสามารถทางด้านการรักษาความปลอดภัย (security) ได้

ในภาษาจาวา เราสามารถกำหนดระดับความสามารถในการรักษาความปลอดภัยโดยการกำหนดระดับการเข้าใช้ข้อมูลขณะที่เราประกาศคลาสนั้น ซึ่งมีระดับการเข้าถึง 4ระดับด้วยกัน คือ private, protected, public และ package

และในตาราง 4 - 11 ก็ได้แสดงถึงว่าแต่ละระดับอนุญาตให้อะไรเข้าใช้บ้างดังนี้

Specifier	class	subclass	package	world
private	X			
protected	X	X	X	
public	X	X	X	X
package	X		X	

ตารางที่ 4 - 11 ระดับความสามารถในการเข้าใช้ข้อมูลของคลาส

โดยในคอลัมน์แรกแสดงถึงว่า ตัวคลาส มันเองสามารถเข้าใช้ได้ คอลัมน์ที่สองหมายถึงว่าสับคลาส ของมันสามารถเข้าใช้ได้ คอลัมน์ที่สาม หมายถึงว่าคลาส ในแพ็คเกจเดียวกันสามารถเข้าใช้ได้ และสุดท้ายหมายถึงให้เข้าใช้ได้หมดเลย

คอนสตรักเตอร์ (Constructor)

คลาสของจาวาจะมี method พิเศษที่เรียกว่า คอนสตรักเตอร์ ซึ่งใช้ในการ initialize ออบเจ็คท์ใหม่ของคลาสนั้น ๆ คอนสตรักเตอร์จะมีชื่อเดียวกับคลาส และคลาสแต่ละคลาส จะมี คอนสตรักเตอร์กี่ตัวก็ได้ และทุกตัวจะมีชื่อเดียวกันดังตัวอย่างของ Rectangleclass อาจจะมีคอนสตรักเตอร์ดังนี้

```
public Rectangle()
public Rectangle(int width, int height)
public Rectangle(int x, int y, int width, int height)
public Rectangle(Dimension size)
public Rectangle(Point location)
public Rectangle(Point location, Dimension size)
```

ซึ่งแต่ละตัวจะมีค่าอาร์กูเมนต์ที่แตกต่างกัน และในการประกาศคอนสตรักเตอร์ก็จะมีการกำหนดระดับการเข้าใช้ตัวแปร หรือ method 4 แบบเหมือนดังที่กล่าวมาก่อนหน้านี้ คือ private protected public และ package

สับคลาส (Subclass) , ซุปเปอร์คลาส (Superclass) และ Inheritance

จากที่ได้กล่าวมาแล้วในส่วนของ OOP ที่ได้บอกเกี่ยวกับสับคลาสที่จะเกิดจากซุปเปอร์คลาส โดยอาศัยคุณสมบัติของ Inheritance ที่จะรับคุณสมบัติทุกอย่างของซุปเปอร์คลาสของมันมานั่นคือ ตัวแปร และ method

การสร้างสับคลาส

เราจะอาศัยการสร้างผ่านการประกาศคลาสที่ได้กล่าวมาแล้วข้างต้น ดังนี้

```
class Subclass extends SuperClass {
    ...
}
```

แต่อย่างไรก็ตามคลาสนี้ก็สามารถอ้างอิงได้เพียงซุปเปอร์คลาสขั้นเดียวเท่านั้น จากการประกาศข้างต้น จะเห็นว่า เราอาศัยคีย์เวิร์ด extends เมื่อเรามีสับคลาสแล้ว เราสามารถที่จะเพิ่ม หรือแทนที่ method เดิมที่มีอยู่ (ที่ได้รับจากซุปเปอร์คลาส) ได้มากเท่าที่ต้องการ

อินเตอร์เฟซ (Interface)

อินเตอร์เฟซคือการรวบรวมความหมายของ method และค่าคงที่ต่าง ๆ เอาไว้อินเตอร์เฟซนี้มีประโยชน์ดังนี้

1. เป็นการเก็บรวมความเหมือนกันของคลาสต่าง ๆ ที่ไม่เกี่ยวข้องกัน
2. เป็นการประกาศ method ที่ต้องอาศัยคลาสหนึ่งหรือมากกว่าในการทำงาน

ในการสร้างอินเตอร์เฟซขึ้นมาเราต้องกระทำสองอย่างด้วยกัน คือ การประกาศอินเตอร์เฟซ และ Interface Body ดังรูปแบบข้างล่าง

```
interfaceDeclaration {
    interfaceBody
}
```

สำหรับการประกาศอินเตอร์เฟซ (Interface Declaration) จะต้องมีองค์ประกอบอย่างน้อยดังนี้

```
interface Countable {
    ...
}
```

สำหรับการตั้งชื่ออินเตอร์เฟซเราจะใช้ชื่อขึ้นต้นเป็นอักษรใหญ่ เช่นเดียวกับชื่อของคลาส นอกจากนี้อินเตอร์เฟซยังสามารถทำได้อย่างคลาสทั่ว ๆ ไปในการใช้คุณสมบัติ Inheritance รับคุณสมบัติจากตัวอินเตอร์เฟซต้น โดยอาศัยคีย์เวิร์ด extends ดังตัวอย่างข้างล่างนี้

```
[public] interface InterfaceName [extends listOfSuperInterfaces] {
    ...
}
```

สำหรับการเขียน interface body เราจะใช้การประกาศ method และการประกาศค่าคงที่เช่นเดียวกับที่เคยกล่าวถึงมาข้างต้น ดังตัวอย่างข้างล่างนี้

```
interface Collection {
    int MAXIMUM = 500;

    void add(Object obj);
    void delete(Object obj);
    Object find(Object obj);
    int currentCount();
}
```

การเขียนแอพเพล็ท (Writing Applets)

สำหรับการเขียนแอพเพล็ทก็เช่นเดียวกับการเขียนจาวา ซึ่งเราต้องอาศัยหลักการพื้นฐานจากการเขียนภาษาจาวา และในแอพเพล็ทก็จะมีองค์ประกอบบางอย่างพื้นฐานที่แตกต่างจากจาวา ซึ่งจะขอกล่าวเป็นหัวข้อย่อย ๆ ดังนี้

1. หลักการพื้นฐานของแอพเพล็ท (Overview of Applets)-
2. การสร้าง Applet User Interface
3. การติดต่อสื่อสารกับโปรแกรมอื่น ๆ

1. หลักการพื้นฐานของแอพเพล็ท (Overview of Applets)

ก่อนที่จะเริ่มหลักพื้นฐานใด ๆ ของแอพเพล็ท ก็จะขอยกตัวอย่างแอพเพล็ทที่แสดงถึงสิ่งที่แอพเพล็ทสามารถทำได้ ดังนี้

```
import java.applet.Applet;
import java.awt.Graphics;

public class Simple extends Applet {
    StringBuffer buffer;

    public void init() {
        buffer = new StringBuffer();
        addItem("initializing... ");
    }

    public void start() {
        addItem("Starting... ");
    }

    public void stop() {
        addItem("Stopping... ");
    }
    public void destroy() {
        addItem("Preparing for Unloading... ");
    }
    void addItem(String newWord) {
        System.out.println(newWord);
        buffer.append(newWord);
        repaint();
    }
    public void paint(Graphics g) {
        g.drawRect(0, 0, size().width - 1, size().height - 1);
        g.drawString(buffer.toString(), 5, 15);
    }
}
```

สำหรับการเขียน และหลักพื้นฐานทั่ว ๆ ไปของการใช้แอพเพล็ท จะกล่าวถึงในรายละเอียดดังต่อไปนี้

1.1 วงจรชีวิตพื้นฐานของแอพเพล็ท (Life Cycle of an Applet)

วงจรชีวิตพื้นฐานในที่นี้ หมายถึง วงจรที่แอพเพล็ทต้องทำงานด้วย method ต่าง ๆ ที่มันอ้างอิงถึงซึ่งตามปกติจะมีการทำงานวนเวียนดังนี้

1. initialize ตัวมันเอง
2. เริ่มรันแอพเพล็ท
3. หยุดรันแอพเพล็ท
4. ทำลายตัวเอง เพื่อเตรียมพร้อมที่หยุดการทำงาน จนกว่าจะมีการเรียกใช้ใหม่

นั่นคือ มันจะเริ่มต้นที่การ Initialize ตัวเอง และจบลงด้วยการทำลายตัวเอง แต่ในขณะที่ทำงาน บางครั้งเราทิ้งหน้าจอไปสักครู่ แล้วค่อยกลับมาดูอีกครั้งหนึ่ง เราจะพบว่า แอพเพล็ทจะไม่ทำลายตัวเอง เพียงแต่ละหยุดการรันแอพเพล็ทเท่านั้น และเมื่อเรากลับมาดูอีกครั้งก็จะเริ่มมันใหม่ โดยไม่จำเป็นต้อง Initialize ตัวเองอีกครั้ง ซึ่งจะทำให้การรันแอพเพล็ทเร็วขึ้น แต่อย่างไรก็ตาม มันก็ต้องใช้ทรัพยากรสิ้นเปลืองเช่นกัน

สำหรับวงจรชีวิตที่กล่าวถึงข้างต้นนี้ จะต้องอาศัย method พื้นฐานดังนี้

1. init() ใช้ initialize แอพเพล็ททุก ๆ ครั้งที่โหลดมันขึ้นมา
2. start() ใช้เริ่มรันแอพเพล็ท
3. stop() ใช้หยุดการรักแอพเพล็ท
4. destroy() ใช้ทำลายตัวเอง

1.2 method สำหรับใช้ใน Drawing และ Event Handling

ในขั้นแรกจะขอยกตัวอย่างง่าย ๆ ก่อน

```
class Simple extends Applet {
    ...
    public void paint(Graphics g) {...}
    ...
}
```

method พื้นฐานที่เราใช้ในการแสดงผลออกสู่หน้ามี 2 อย่าง คือ

1. paint() ซึ่งเป็น method พื้นฐานที่ใช้ในการแสดงผลออกสู่หน้าจอ หรือหน้าของบราวเซอร์
2. update() เป็น method ที่ใช้กับ paint() ซึ่งใช้พัฒนาประสิทธิภาพในการ Drawing

1.3 method ที่ใช้ในการเพิ่มส่วนติดต่อกับผู้ใช้ หรือ User interface

ในหัวข้อนี้ เราจะแบ่งเรื่องที่กล่าวถึงเป็นเรื่องย่อย ๆ ดังนี้

1.3.1 User Interface พื้นฐานที่ภาษาจาวาทำให้อยู่แล้ว ซึ่งจะอยู่ใน Abstract Window Toolkit Class (AWT) โดยเราสามารถนำมาใช้ได้เลยโดยบอกให้คอมไพเลอร์รับรู้ที่เมื่อเริ่มต้นโปรแกรม สำหรับ ส่วนติดต่อกับผู้ใช้ (UI : User Interface) ที่สามารถนำมาใช้ได้ได้แก่

- Buttons (java.awt.Button)
- Checkboxes (java.awt.Checkbox)
- Single-line text fields (java.awt.TextField)
- Larger text display and editing areas (java.awt.TextArea)
- Labels (java.awt.Label)
- Lists (java.awt.List)
- Pop-up lists of choices (java.awt.Choice)
- Sliders and scrollbars (java.awt.Scrollbar)
- Drawing areas (java.awt.Canvas)
- Menus (java.awt.Menu, java.awt.MenuItem, Java.awt.checkboxMenuItem)
- Containers (java.awt.Panel, java.awt.Window)

สำหรับ UI ที่กล่าวมาข้างต้น เราสามารถอ้างอิงได้จากคลาสที่เขียนไว้ในวงเล็บด้านหลัง

1.3.2 method ที่ใช้ในการเรียกใช้ UI ได้แก่

- add() เป็นการเพิ่ม UI ต่าง ๆ
- remove() เป็นการเอา UI ออกไป
- setLayout() เป็นการใช้งานกับ layout manager ซึ่งเป็นการจัดหน้าจอ

1.4 การใส่แอพเพล็ทลงในหน้าจอของ HTML

เราสามารถทำได้โดยการใส่ tag <APPLET> ลงในโค้ดของ HTML ในที่นี้ขอยกรูปแบบมาตรฐานของการเขียนโค้ด ดังนี้

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
</APPLET>
```

จากตัวอย่างข้างต้น เราจะเห็นองค์ประกอบพื้นฐานต่อไปนี้

1. เริ่มต้นด้วยการประกาศว่าเป็น APPLET-tag
2. ตามด้วยการบอกถึงไฟล์ที่จะใช้ในการรัน ซึ่งจะต้องเป็นไฟล์ที่คอมไพล์เรียบร้อยแล้ว นั่นคือเป็นไฟล์ .class จากนั้นก็บอกถึงขนาดความกว้างและความยาวของหน้าจอ หรือพื้นที่ที่ต้องการให้แสดงแอพเพล็ทนั้น ๆ (WIDTH และ HEIGHT)
3. จากนั้นก็ปิดท้ายด้วย </APPLET>
4. นอกจากนี้ เรายังมีฟังก์ชันที่ใช้บอกถึงที่อยู่ในไฟล์ไบต์โค้ดนั้นด้วย ซึ่งมีรูปแบบดังนี้

```
<APPLET CODE=AppletSubclass.class CODEBASE=aURL
    WIDTH=anInt HEIGHT=anInt>
</APPLET>
```

ซึ่งในส่วนของ CODEBASE นี้ นอกจากจะใช้เป็น URL แล้ว ยังสามารถแทนด้วยที่อยู่ไดเร็กทอรี่ได้ด้วย และในกรณีที่มีการรับค่าเข้าสู่แอพเพล็ท ก็สามารถทำได้ผ่าน <PARAM> ตามรูปแบบ

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
<PARAM NAME=parameter1Name VALUE=aValue>
<PARAM NAME=parameter2Name VALUE=anotherValue>
</APPLET>
```

สำหรับการเขียนจริง ๆ ไม่จำเป็นที่จะต้องเป็นตัวใหญ่ทั้งหมด ดังตัวอย่างข้างล่าง

```
<applet code=AppletButton.class codebase=example width=350 height=60>
<param name=windowType value=BorderWindow>
<param name=windowtext value="BordreLayout">
<param name=buttonText value="Click here to see a BorderLayout in action">
<blockquote>
<hr>
<em>
Your browser can't run 1.0 Java applets,
so here's a picture of the window the program brins up:</em>
<p>
<img src=images/BorderEx1.gif width=302 height=138>
<hr>
</blockquote>
</applet>
```

จากตัวอย่างเราจะเห็น <blockquote>-tag ซึ่งใช้เป็นตัวบอกในกรณีที่บราวเซอร์นั้นไม่สามารถรันแอพเพล็ทนั้นได้ ก็จะข้ามบล็อกนี้ไป

2. การสร้าง Applet User Interface

ในที่นี้จะขอกล่าวคร่าว ๆ ในแต่ละหัวข้อดังนี้

การสร้าง GUI (Graphic User Interface)

ตามปกติการใช้งานแอพเพล็ทในบราวเซอร์ต่าง ๆ ก็ล้วนเป็น GUI (Graphic User Interface) อยู่แล้ว สำหรับคุณสมบัติพื้นฐานของ Applet GUI ก็มีดังนี้

1. การทำงานแบบ Panel เนื่องจากการใช้งานทางด้าน User Interface ของแอพเพล็ทอาศัย method จากคลาส AWT ทำให้การทำงานของแอพเพล็ทที่สร้างขึ้นจะได้รับ layout manager ของ AWT Panel มา ทำให้การทำงานจะเป็นรูปแบบของ panel ทั้งหมด

2. เนื่องด้วยการปรากฏของแอพเพล็ทจะปรากฏอยู่บนบราวเซอร์ที่อยู่บนวินโดว์ ทำให้เราสามารถแสดงแอพเพล็ทในหลายรูปแบบ ไม่ว่าจะเป็นการแสดงโดยอาศัยองค์ประกอบพื้นฐานที่เหมือนกับต้นฉบับ หรือเหมือนกับวินโดว์ต้นทางนั้น เช่น การแสดงหน้าต่าง ปุ่ม หรือ ลิสต์ต่าง ๆ ในอีกทางหนึ่งเราก็สามารถที่จะแสดงโดยสร้างรูปแบบขึ้นมาใหม่ก็ได้
3. สีของแบ็กกราวน์ และฟอร์กราวน์ที่แตกต่างกัน เดิมในแอพเพล็ทที่แสดงบนหน้าจอวินโดวส์จะเป็นสีเทาอ่อน เช่นเดียวกับการแสดงหน้าจอของ HTML ที่จะแสดงสีที่แตกต่างกัน ทำให้แอพเพล็ท สามารถอาศัยสีที่มีอยู่เดิมได้ ในขณะเดียวกัน มันก็สามารถที่จะแสดงอื่น ๆ ที่แตกต่างกันไปโดยอาศัย method ที่เหมาะสม เช่น setBackground()
4. ขนาดของหน้าจอ หรือขนาดของแอพเพล็ทที่แสดง สามารถกำหนดได้เอง หรือมันอาจจะกำหนดขนาดที่เหมาะสมให้เอง
5. สามารถโหลดรูปภาพ โดยใช้ method ที่ชื่อ getImage()

การใช้งานพารามิเตอร์ของแอพเพล็ท

สำหรับพารามิเตอร์ที่ใช้ในการส่งค่าไปยังแอพเพล็ทนี้ เราสามารถใช้ได้หลากหลายรูปแบบของข้อมูล ไม่ว่าจะเป็น ข้อมูลเกี่ยวกับแหล่งที่เก็บของไฟล์ วิธีการแสดงผลของแอพเพล็ทบนหน้าจอ หรือรายละเอียดอื่น ๆ ของการแสดงผล เป็นต้น สำหรับค่าที่สามารถส่งเป็นพารามิเตอร์ไป ก็ได้แก่

1. ค่า URL
2. จำนวนเต็ม (integer)
3. เลขจุดทศนิยม
4. ค่าบูลีน จะเป็น จริง หรือ เท็จ
5. สตริง
6. หลาย ๆ ค่าที่กล่าวมาทั้งหมดรวมกัน

ส่วนการตั้งชื่อของพารามิเตอร์นี้ โดยทั่ว ๆ ไปเรามีชื่อพารามิเตอร์ที่ใช้กันทั่ว ๆ ไป เช่น

1. SOURCE หรือ SRC ใช้สำหรับไฟล์ข้อมูล เช่น ไฟล์รูปภาพ
2. XXXSOURCE เช่น IMAGESOURCE จะใช้ในกรณีที่มีการอ้างอิงถึงไฟล์ข้อมูลไม่ซ้ำแบบกัน เช่น อาจมีไฟล์รูปภาพ และไฟล์เอกสาร (Document) เป็นต้น
3. XXXS เช่น IMAGES ใช้สำหรับกรณีที่มีการอ้างอิงถึงข้อมูลมาก ๆ

4. NAME จะใช้เฉพาะกับชื่อของแอพเพล็ท โดยเฉพาะในกรณีที่ต้องมีการติดต่อระหว่างแอพเพล็ท

ในกรณีที่เราไม่ได้กำหนดค่าใด ๆ ให้แก่พารามิเตอร์ คอมไพเลอร์ก็จะเลือกค่าที่เหมาะสมให้แอพเพล็ทนั้นเอง เช่น กรณีที่เราต้องการแสดงภาพเคลื่อนไหวบนหน้าจอ คอมไพเลอร์ก็จะเลือกจำนวนภาพที่จะแสดงที่เหมาะสมให้

ในที่นี้ขอยกตัวอย่างซอร์สโค้ดแอพเพล็ทง่าย ๆ ซึ่งเราจะใช้อ้างอิงในการเขียนถึงการใช้พารามิเตอร์

```
/*
 * Copyright (c) 1995, 1996 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
//TO DO: Close all windows when you leave the page?
import java.awt.*;
import java.util.*;
import java.applet.Applet;

public class AppletButton extends Applet implements Runnable {
    int frameNumber = 1;
    String windowClass;
    String buttonText;
    String windowTitle;
    int requestedWidth = 0;
    int requestedHeight = 0;
    Button button;
    Thread windowThread;
    Label label;
    boolean pleaseCreate = false;

    public void init() {
        windowClass = getParameter("WINDOWCLASS");
        if (windowClass == null) {
            windowClass = "TestWindow";
        }

        buttonText = getParameter("BUTTONTEXT");
```

```
    if (buttonText == null) {
      buttonText = "Click here to bring up a " + windowClass;
    }

    windowTitle = getParameter("WINDOWTITLE");
    if (windowTitle == null) {
      windowTitle = windowClass;
    }
    String windowWidthString = getParameter("WINDOWWIDTH");
    if (windowWidthString != null) {
      try {
        requestedWidth = Integer.parseInt(windowWidthString);
      } catch (NumberFormatException e) {
        //Use default width.
      }
    }

    String windowHeightString = getParameter("WINDOWHEIGHT");
    if (windowHeightString != null) {
      try {
        requestedHeight = Integer.parseInt(windowHeightString);
      } catch (NumberFormatException e) {
        //Use default height.
      }
    }

    setLayout(new GridLayout(2,0));
    add(button = new Button(buttonText));
    button.setFont(new Font("Helvetica", Font.PLAIN, 14));

    add(label = new Label("", Label.CENTER));
  }

  public void start() {
    if (windowThread == null) {
      windowThread = new Thread(this, "Bringing Up " + windowClass);
      windowThread.start();
    }
  }
  public synchronized void run() {
    Class windowClassObject = null;
    Class tmp = null;
    String name = null;

    // Make sure the window class exists and is really a Frame.
    // This has the added benefit of pre-loading the class,
    // which makes it much quicker for the first window to come up.
    try {
      windowClassObject = Class.forName(windowClass);
    } catch (Exception e) {
      // The specified class isn't anywhere that we can find.
      label.setText("Can't create window: Couldn't find class "
                    + windowClass);
      button.disable();
      return;
    }
```

```
// Find out whether the class is a Frame.
for (tmp = windowClassObject, name = tmp.getName();
    !( name.equals("java.lang.Object") ||
      name.equals("java.awt.Frame") ); ) {
   tmp = tmp.getSuperclass();
   name = tmp.getName();
}
if ((name == null) || name.equals("java.lang.Object")) {
   //We can't run; ERROR; print status, never bring up window
   label.setText("Can't create window: "
                 + windowClass +
              " isn't a Frame subclass.");
   button.disable();
   return;
} else if (name.equals("java.awt.Frame")) {
   //Everything's OK. Wait until we're asked to create a window.
   while (windowThread != null) {
      while (pleaseCreate == false) {
         try {
            wait();
         } catch (InterruptedException e) {
         }
      }

      //We've been asked to bring up a window.
      pleaseCreate = false;
      Frame window = null;
      try {
         window = (Frame)windowClassObject.newInstance();
      } catch (Exception e) {
         label.setText("Couldn't create instance of class "
                  + windowClass);
         button.disable();
         return;
      }
      if (frameNumber == 1) {
         window.setTitle(windowTitle);
      } else {
         window.setTitle(windowTitle + ": " + frameNumber);
      }
      frameNumber++;

      //Set the window's size.
      window.pack();
      if ((requestedWidth > 0) | (requestedHeight > 0)) {
         window.resize(Math.max(requestedWidth,
                        window.size().width),
                  Math.max(requestedHeight,
                        window.size().height));
      }

      window.show();
      label.setText("");
   }
}
```

```
    }

    public synchronized boolean action(Event event, Object what) {
        if (event.target instanceof Button) {
            //signal the window thread to build a window
            label.setText("Please wait while the window comes up...");
            pleaseCreate = true;
            notify();
        }
        return true;
    }
}

class TestWindow extends Frame {
    public TestWindow() {
        resize(300, 300);
    }
}
```

จากตัวอย่างโปรแกรม เมื่อเรามาเขียน APPLET-tag ในโค้ด HTML และมีการกำหนดค่าพารามิเตอร์ เราจะเขียนได้ดังนี้

```
<APPLET CODE=AppletButton.class CODEBASE=example WIDTH=350 HEIGHT=60>
<PARAM NAME=windowClass VALUE=BorderWindow>
<PARAM NAME=windowTitle VALUE="BorderLayout">
<PARAM  NAME=buttonText  VALUE="Click here to see a BorderLayout in action">
</APPLET>
```

โดยไฟล์ตัวอย่างนี้มีชื่อว่า AppletButton.java เมื่อคอมไพล์แล้วจึงได้ AppletButton.class

สำหรับหลักการเขียนโค้ดเพื่อรับพารามิเตอร์ในซอร์สโค้ดภาษาจาวา เขียนได้โดยเราจะอาศัย method ที่ชื่อว่า getParameter() ดังรูปแบบ

public String getParameter(String name)

สำหรับในตัวอย่างโปรแกรมก็มีการใช้ดังกล่าวดังข้างล่างที่ยกมาเฉพาะส่วน

```
String windowClass;
String buttonText;
String windowTitle;
int requestedWidth = 0;
int requestedHeight = 0;
...
public void init() {
    windowClass = getParameter("WINDOWCLASS");
```

```
if (windowClass == null) {
   windowClass = "TestWindow";
}

buttonText = getParameter("BUTTONTEXT");
if (buttonText == null) {
   buttonText = "Click here to bring up a " + windowClass;
}

windowTitle = getParameter("WINDOWTITLE");
if (windowTitle == null) {
   windowTitle = windowClass;
}

String windowWidthString = getParameter("WINDOWWIDTH");
if (windowWidthString != null) {
   try {
      requestedWidth = Integer.parseInt(windowWidthString);
   } catch (NumberFormatException e) {
      //Use default width.
   }
}

String windowHeightString = getParameter("WINDOWHEIGHT");
if (windowHeightString != null) {
   try {
      requestedHeight = Integer.parseInt(windowHeightString);
   } catch (NumberFormatException e) {
      //Use default height.
   }
}
```

จากส่วนที่ยกมาจะเห็นการรับค่าพารามิเตอร์ และการกำหนดพารามิเตอร์

3. การติดต่อสื่อสารกับโปรแกรมอื่น ๆ

แอพเพล็ทต่าง ๆ สามารถติดต่อกับโปรแกรมอื่น ๆ ได้ใน 3 ทางดังนี้

1. โดยการเรียกใช้ method ของแอพเพล็ทอื่นบนเพจเดียวกัน
2. โดยการใช้ API ซึ่งจะมี method ที่ช่วยให้แอพเพล็ทสามารถติดต่อกับบราวเซอร์ หรือ Applet Viewer ที่เก็บแอพเพล็ทนี้
3. โดยการใช้ API ช่วยให้สามารถติดต่อกับโปรแกรมอื่น ๆ ผ่านเน็ตเวิร์ก โดยจุดสำคัญก็คือ จะต้องอยู่ในเน็ตเวิร์กที่ใช้โฮสต์เดียวกัน

สำหรับการติดต่อสื่อสารจะมีหัวข้อย่อย ๆ ที่จะต้องพิจารณาดังที่กล่าวมาข้างต้น ดังนี้

การส่ง message ไปยังแอพเพล็ทอื่น ๆ บนเพจเดียวกัน

ปกติแอพเพล็ทจะพยายามค้นหาแอพเพล็ทอื่น ๆ และส่ง message ไปยังแอพเพล็ทนั้น โดยดั้งอยู่บนพื้นฐานทางด้านการรักษาความปลอดภัยดังนี้

1. แอพเพล็ทจะต้องรันอยู่ในบนเพจเดียวกัน และบนบราวเซอร์ตัวเดียวกัน
2. มี Applet Viewer หลายตัวที่ต้องการให้แอพเพล็ทเหล่านั้นเกิดจากเซิร์ฟเวอร์เดียวกันด้วย การค้นหาแอพเพล็ทอื่น สามารถทำได้โดยใช้ method ชื่อ getApplet() โดยการหาอาจหาโดยหาตามรายชื่อแอพเพล็ท หรือการค้นหาทั้งหมดเลยก็ได้ ซึ่งตามปกติแอพเพล็ทจะไม่มีชื่อ การที่แอพเพล็ทจะมีชื่อได้จะต้องมีการกำหนดให้มันไว้ในไฟล์ HTML โดยการกำหนดสามารถทำได้ 2 แบบ ดังนี้

- โดยการใส่ชื่อไว้ใน APPLET-tag

```
<applet codebase=example/ code=Sender.class width=450 height=200
name="buddy">
...
</applet>
```

- โดยการใส่ใน PARAM-tag

```
<applet codebase=example/ code=Receiver.class width=450 height=35>
<param name="name" value="old pal">
...
</applet>
```

ตัวอย่างข้างล่างนี้จะเป็นตัวอย่างของโปรแกรม sender และ receiver

```
/*
 * Copyright (c) 1995, 1996 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licerising information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.applet.*;
import java.awt.*;
import java.util.Enumeration;
```

```
public class Sender extends Applet {
   private String myName;
   private TextField nameField;
   private TextArea status;

   public void init() {
      GridBagLayout gridBag = new GridBagLayout();
      GridBagConstraints c = new GridBagConstraints();

      setLayout(gridBag);

      Label receiverLabel = new Label("Receiver name:", Label.RIGHT);
      gridBag.setConstraints(receiverLabel, c);
      add(receiverLabel);

      nameField = new TextField(getParameter("RECEIVERNAME"), 10);
      c.fill = GridBagConstraints.HORIZONTAL;
      gridBag.setConstraints(nameField, c);
      add(nameField);

      Button button = new Button("Send message");
      c.gridwidth = GridBagConstraints.REMAINDER; //end row
      c.anchor = GridBagConstraints.WEST; //stick to the text field
      c.fill = GridBagConstraints.NONE; //keep the button small
      gridBag.setConstraints(button, c);
      add(button);

      status = new TextArea(5, 60);
      status.setEditable(false);
      c.anchor = GridBagConstraints.CENTER; //reset to the default
      c.fill = GridBagConstraints.BOTH; //make this big
      c.weightx = 1.0;
      c.weighty = 1.0;
      gridBag.setConstraints(status, c);
      add(status);

      myName = getParameter("NAME");
      Label senderLabel = new Label("(My name is " + myName + ".)",
                  Label.CENTER);
      c.weightx = 0.0;
      c.weighty = 0.0;
      gridBag.setConstraints(senderLabel, c);
      add(senderLabel);

      validate();
   }

   public boolean action(Event event, Object o) {
      Applet receiver = null;
      String receiverName = nameField.getText(); //Get name to search for.
      receiver = getAppletContext().getApplet(receiverName);
      if (receiver != null) {
   //Use the instanceof operator to make sure the applet
            //we found is a Receiver object.
         if (!(receiver instanceof Receiver)) {
```

```
            status.appendText("Found applet named "
                        + receiverName + ", "
                        + "but it's not a Receiver object.\n");
        } else {
            status.appendText("Found applet named "
                        + receiverName + ".\n"
                        + " Sending message to it.\n");
                    //Cast the receiver to be a Receiver object
                    //(instead of just an Applet object) so that the
                    //compiler will let us call a Receiver method.
            ((Receiver)receiver).processRequestFrom(myName);
        }
    } else {
        status.appendText("Couldn't find any applet named "
                    + receiverName + ".\n");
    }
    return false;

  }

  public Insets insets() {
    return new Insets(3,3,3,3);
  }

  public void paint(Graphics g) {
    g.drawRect(0, 0, size().width - 1, size().height - 1);
  }

  public String getAppletInfo() {
    return "Sender by Kathy Walrath";
  }
}
```

```
/*
 * Copyright (c) 1995, 1996 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.applet.*;
import java.awt.*;

public class Receiver extends Applet {
    private final String waitingMessage = "Waiting for a message...          ";
    private Label label = new Label(waitingMessage, Label.RIGHT);
```

```
    public void init() {
        add(label);
        add(new Button("Clear"));
        add(new Label("(My name is " + getParameter("name") + ".)",
                Label.LEFT));
        validate();
    }

    public boolean action(Event event, Object o) {
        label.setText(waitingMessage);
        repaint();
        return false;
    }

    public void processRequestFrom(String senderName) {
        label.setText("Received message from " + senderName + "!");
        repaint();
    }

    public void paint(Graphics g) {
        g.drawRect(0, 0, size().width - 1, size().height - 1);
    }

    public String getAppletInfo() {
        return "Receiver (named " + getParameter("name") + ") by Kathy Walrath";
    }
}
```

ซึ่งเราจะเห็นถึงหลักการใช้ getApplets() เพื่อหาแอพเพล็ทบนเพจ โดยการใช้ method นี้จะให้ค่าของแอพเพล็ททั้งหมดที่ปรากฏบนเพจนั้น ตัวอย่างข้างล่างนี้ยกมาให้เห็นถึงรูปแบบการเรียกใช้แอพเพล็ทโดยใช้ method getApplets()

```
    public void printApplets() {
        Enumeration e = getAppletCohtext().getApplets();
        ...
        while (e.hasMoreElements()) {
            Applet applet = (Applet)e.nextElement();
            String info = ((Applet)applet).getAppletInfo();
            if (info != null) {
                textArea.appendText("- " + info + "\n");
            } else {
                textArea.appendText("- " + applet.getClass().getName() + "\n");
            }
        }
        ...
    }
```

ข้อแตกต่างของการเขียนแอพพลิเคชั่น และแอพเพล็ท

เราสามารถเปรียบเทียบการเขียนแอพพลิเคชันด้วยภาษาจาวา และ การเขียนแอพเพล็ทด้วยภาษาจาวาได้ตามตารางที่ 4 - 12

การเขียนแอพพลิเคชันด้วยภาษาจาวา	การเขียนแอพเพล็ทด้วยภาษาจาวา
คอมไพล์ด้วยจาวาคอมไพเลอร์ (Java Compiler) และได้ไบท์โค้ด ในรูปไฟล์ .class	เช่นเดียวกัน
สามารถนำไปเอ็กซีคิวต์ เพื่อทำงานเป็นแอพพลิเคชั่นทั่วๆ ไปโดยอาศัยการเรียก แอพพลิเคชั่นขึ้นใช้งานโดยใช้จาวาอินเตอร์พรีเตอร์ (Java Interpreter)	เรียกใช้ผ่านบราวเซอร์ที่สนับสนุนการทำงานของภาษาจาวา โดยฝังแอพเพล็ทที่ผ่านการคอมไพล์แล้วไว้ในไฟล์ HTML
ใช้หลักการ (syntax) ของการเขียนภาษาจาวาทั่วไปโดยเป็นการเขียนแบบโอโอพี (OOP)	เช่นเดียวกัน
เอ็กซีคิวต์ ณ เครื่องที่เรียกใช้งาน โดยอินเตอร์พรีเตอร์จากไบท์โค้ดของมัน	จะเอ็กซีคิวต์ ณ เครื่องที่เรียกใช้นั้นโดยอาศัยการอินเตอร์พรีเตอร์จากไบท์โค้ด

ตารางที่ 4 - 12 เปรียบเทียบการเขียนแอพพลิเคชั่น และแอพเพล็ทด้วยภาษาจาวา

จาวาสคริปต์ (JavaScript)

จะเป็นการนำภาษาจาวาไปเขียนไว้ใน HTML page โดยมีรูปแบบการเขียนที่เรียกว่าจาวาสคริปต์ ซึ่งเป็นการเขียนแทค (tag) แบบ HTML แต่อาศัยหลักการของภาษาจาวา ปัจจุบันมีผู้นิยมเขียนจาวาสคริปต์กันมากเนื่องจากมีวิธีการเขียนที่ง่าย

ตัวอย่างการเขียนจาวาสคริปต์

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- to hide script contents from old browsers
   function square(i) {
document.write("The call passed ", i ," to the function.","&lt;BR&gt;")
      return i * i      }
   document.write("The function returned ",square(5),".")
// end hiding contents from old browsers -->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
<!-- hide script form old browsers
document.write(bar(),output("Make Me Big",3,"Make me ordinary."))
// end hiding from old browsers -->
</SCRIPT>
<P>Thanks.
</BODY>
```

คุณสมบัติของจาวาสคริปต์เปรียบเทียบกับการเขียนจาวา สามารถเปรียบเทียบและแสดงให้เห็นได้ดังตารางที่ 4 - 13

การเขียนจาวา	การเขียน JavaScript
เป็นการเขียนภาษาจาวาตามหลักการเขียนปกติ และมีบิวท์-อินฟังก์ชั่น (built-in function) ช่วยในการเขียน	เป็นการเขียนภาษาสคริปต์ (Script language) โดยอาศัย syntax ของจาวา และมีบิวท์-อินฟังก์ชั่น ช่วยในการเขียน
เป็นโอโอพี	เป็น Object-based language
ถ้าเป็นจาวาสคริปต์ก็จะใส่ไบท์โค้ดไว้ในไฟล์ HTML ถ้าเป็นจาวาแอพพลิเคชั่นก็เขียนเป็นไฟล์และคอมไพล์ใช้งานตามปกติ	เขียนฝังตัวไว้ใน HTML page
การตรวจสอบตัวแปรจะทำขณะคอมไพล์หรือเป็นระบบคอมไพล์ไทม์ (compile-time)	จะเป็นระบบรันไทม์ (run-time) นั่นคือจะตรวจสอบตัวแปรขณะรันโปรแกรม
ตัวแปรต่าง ๆ จะอยู่ในรูปของวัตถุ (object)	การกำหนดตัวแปรจะมีแบบ numeric, boolean และ string
การเขียนจะซับซ้อนกว่าแบบจาวาสคริปต์	การเขียนจะไม่ยุ่งยาก
มีคุณสมบัติ inheritance ตามแบบโอโอพีทั่ว ๆ ไป	ไม่มีคุณสมบัติ inheritance
	จะทำงานแบบเช็คเหตุการณ์ (Event Handler) เช่น การคลิกเมาส์ การใส่ค่าในแบบฟอร์ม เป็นต้น
การเรียกใช้งาน ถ้าเป็นแอพเพล็ทจะเรียกผ่าน HTML page โดยจะเรียกผ่านไบท์โค้ดที่ถูกคอมไพล์เรียบร้อยแล้ว ที่ไคลเอ็นท์ จะทำหน้าที่ในการอินเตอร์พรีต (interpret) ไบท์โค้ดนั้น โดยอาศัยบราวเซอร์ที่สนับสนุนการทำงานของภาษาจาวา (จะมีจาวาอินเตอร์พรีเตอร์อยู่) ถ้าเป็นแอพพลิเคชันจะใช้งานโดยมีจาวาอินเตอร์พรีเตอร์ในการเรียกใช้งานไบท์โค้ดนั้น ๆ	จะถูกอินเตอร์พรีตที่ตัวบราวเซอร์ที่สนับสนุนการใช้งานภาษาจาวา โดยจะเป็นในลักษณะเดียวกับการ อินเตอร์พรีตภาษา HTML

ตารางที่ 4 - 13 เปรียบเทียบการเขียนจาวาสคริปต์กับการเขียนจาวา

บทที่ 5
ระบบความปลอดภัย

ระบบความปลอดภัย (Security) ในรายงานฉบับนี้จะเกี่ยวข้องกับระบบความปลอดภัยในเครื่องคอมพิวเตอร์และในระบบเน็ตเวิร์ก โดยจะขอกล่าวถึงรายละเอียดของเรื่องระบบความปลอดภัยเป็นเรื่องย่อย ๆ ดังต่อไปนี้

1. เรื่องทั่ว ๆ ไปเกี่ยวกับระบบความปลอดภัย
2. รูปแบบของการเข้ารหัสข้อมูล (Cryptography)

เรื่องทั่ว ๆ ไปเกี่ยวกับระบบความปลอดภัย

ระบบความปลอดภัย ทั้งในระบบคอมพิวเตอร์และระบบติดต่อสื่อสารทั่ว ๆ ไป มีสิ่งที่ควรคำนึงถึงดังนี้

1 รูปแบบการคุกคามระบบคอมพิวเตอร์และระบบติดต่อสื่อสาร ได้แก่

1.1 **Interruption** จะเป็นการไปหยุดการทำงานตามปกติของระบบทำให้เกิดการสูญหายของข้อมูล ทำให้ข้อมูลเกิดการแพร่กระจาย หรือใช้ข้อมูลนั้นไม่ได้อีก

1.2 **Interception** เป็นกรณีที่องค์กร หรือบุคคลอื่น ๆ ที่ไม่มีสิทธิ์ใช้ข้อมูล เข้ามาใช้ข้อมูลโดยไม่ถูกต้อง

1.3 **Modification** เป็นการเข้ามาใช้ข้อมูล รวมทั้งทำการเปลี่ยนแปลงรายละเอียดของข้อมูลนั้นด้วย

1.4 **Fabrication** เป็นการปลอมแปลงว่าเป็นผู้ใช้ปกติของระบบ ทำให้เกิดการเข้าใจผิดของระบบ และทำให้เกิดความเสียหายต่อระบบได้

2 ความสำคัญของระบบรักษาความปลอดภัย

ระบบรักษาความปลอดภัยในคอมพิวเตอร์หรือการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ มีขึ้นเพื่อรักษา 3 สิ่งดังต่อไปนี้

2.1 ความเป็นความลับ (Secrecy) หมายถึง การรักษาไว้ซึ่งข้อมูล หรือระบบที่จะอนุญาตให้เฉพาะผู้ที่มีสิทธิ์เท่านั้น

2.2 ความถูกต้องแน่นอนของข้อมูล (Integrity) หมายถึงระบบนั้นจะได้รับการดูแลให้เฉพาะผู้ที่มีสิทธิ์เท่านั้นที่จะสามารถเปลี่ยนแปลงใด ๆ กับระบบนั้นได้

2.3 การตอบสนองอย่างเหมาะสม (Availability) หมายถึง ผู้ที่มีสิทธิ์ใช้ข้อมูลหรือระบบนั้น ๆ ในระดับต่างๆ ก็จะได้รับการตอบสนองตามระดับอย่างเหมาะสม ขณะเดียวกันก็ป้องกันผู้ที่ไม่สิทธิ์นั้นด้วย

3 รูปแบบของการป้องกันความปลอดภัย
การควบคุม (Control) ประกอบด้วย

3.1 การเข้ารหัส (Encryption) เป็นวิธีที่แพร่หลายที่สุดในการป้องกันความปลอดภัย โดยการเข้ารหัสกับข้อมูลที่ทำการส่ง โดยเรื่องนี้จะกล่าวถึงในตอนท้ายต่อไป

3.2 การควบคุมทางซอฟท์แวร์ (Software Control) ซึ่งประกอบด้วย

3.2.1 การควบคุมทางการพัฒนาโปรแกรม (Development control)

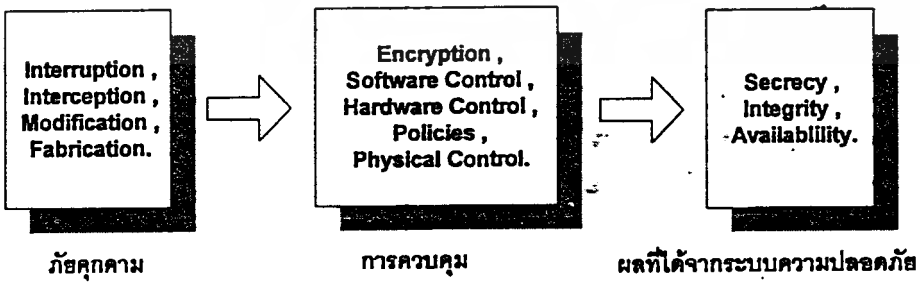
3.2.2 การควบคุมตัวระบบ OS (Operating System Control)

3.2.3 การควบคุมโปรแกรมภายใน (Internal Program Control)

3.3 การควบคุมทางฮาร์ดแวร์ (Hardware Control)

3.4 การกำหนดนโยบายการใช้งาน (Policies) ในกรณีนี้จะมีผลใช้ในระดับที่กว้าง เช่น การกำหนดให้ใส่รหัสลับ (password) เป็นต้น

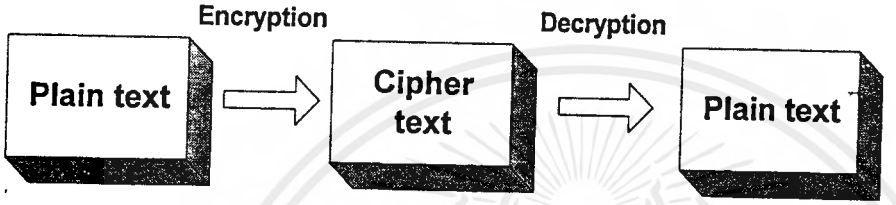
3.5 Physical Controls เป็นการควบคุมทุก ๆ ส่วนในระบบ ไม่ว่าจะเป็นโปรแกรม ข้อมูล ห้องเก็บระบบ พนักงานรักษาความปลอดภัย ฯลฯ



รูปที่ 5 - 1 ข้อควรคำนึงในระบบความปลอดภัย

รูปแบบของการเข้ารหัสข้อมูล (Cryptography)

การเข้ารหัสข้อมูล (Cryptography) คือ การเข้ารหัสให้แก่ข้อมูล เพื่อป้องกันการเข้าใช้ หรือ แอบดูข้อมูลอย่างไม่ถูกต้อง โดยในการเข้ารหัสข้อมูลนี้ จะมีหลักการคร่าว ๆ ดังรูป 5 - 2 นี้

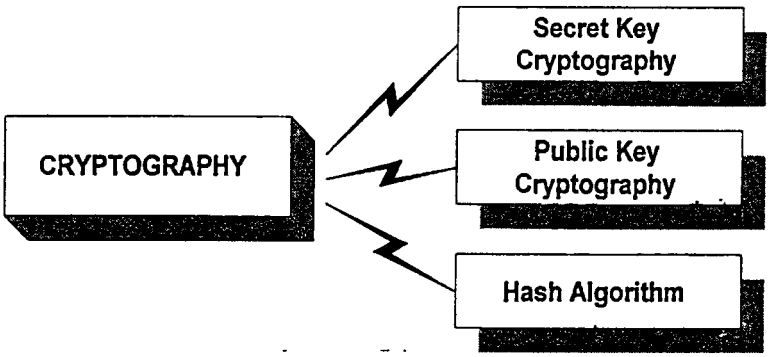


รูปที่ 5 - 2 การเข้ารหัสข้อมูล

โดยข้อมูลเริ่มต้นจะเรียกว่า plain text จากนั้นเมื่อเข้ารหัสให้แก่ข้อมูลนั้น จะเรียกว่า การเข้ารหัส (Encryption) โดยข้อมูลใหม่หลังจากนำข้อมูลมาเข้ารหัสแล้ว เราเรียกว่า cipher ext และเมื่อต้องการจะเปลี่ยนข้อมูลที่ถูกเข้ารหัสนั้นกลับสู่ข้อมูลปกติ เราเรียกว่า การถอดรหัส Decryption) โดยรหัสที่เรานำมาใส่กับข้อมูลนี้ เราเรียกว่า คีย์ (key)

การเข้ารหัสข้อมูล มี 3 รูปแบบด้วยกัน คือ

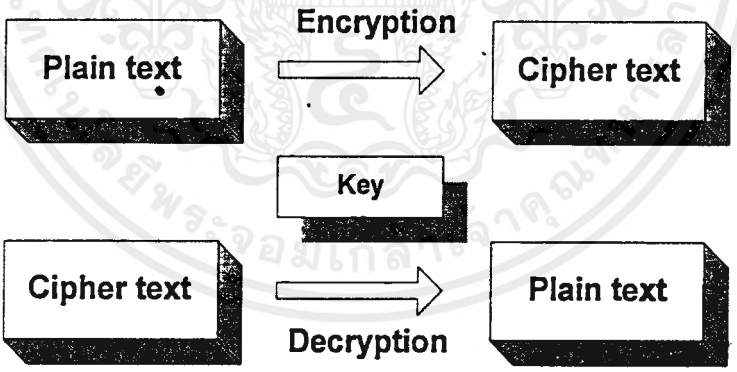
. Secret Key Cryptography
. Public Key Cryptography
. Hash Algorithms



รูปที่ 5 - 3 รูปแบบการเข้ารหัสข้อมูล

Secret Key Cryptography

Secret Key Cryptography เป็นการรักษาความปลอดภัยของข้อมูล โดยการเข้ารหัสให้กับข้อมูลโดยใช้รหัส หรือคีย์ (Key) เพียงคีย์เดียว หรือ เรียกว่า Single key ในการเข้ารหัสจะใช้คีย์เดียวกับที่ใช้ในการถอดรหัส นั่นคือ จะเป็นรูปแบบดังรูปที่ 5 - 4



รูปที่ 5 - 4 Secret Key Cryptography

โดยข้อมูลเริ่มต้นเราเรียกว่า plain text เมื่อผ่านการเข้ารหัส (Encryption) โดยใช้คีย์ก็จะได้ข้อมูลที่ถูกเข้ารหัสเราเรียกว่า cipher text เมื่อต้องการจะถอดรหัส (Decryption) เราก็ใช้คีย์ เดียวกันในการถอดรหัส Secret Key Cryptography บางครั้งเราเรียกว่า Conventional cryptography หรือ Symmetric cryptography

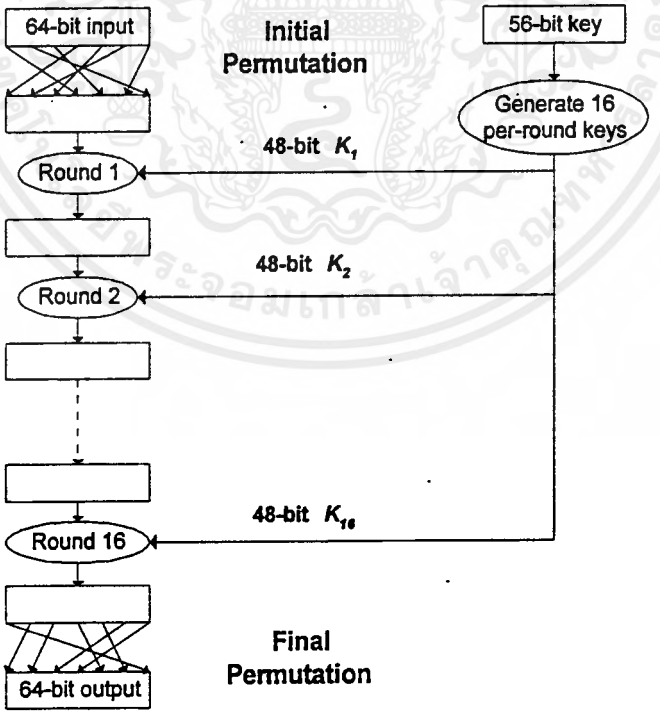
สำหรับการใช้หลักคีย์เดียวนี้ ขอยกตัวอย่างรูปแบบการใช้งานผ่านรูปแบบการเข้ารหัสแบบคีย์เดียวชนิดหนึ่ง ที่เราเรียกว่า Data Encryption Standard หรือ DES ดังนี้

DES

DES พัฒนาขึ้นในปี 1977 โดย National Bureau of Standards โดยอาศัยหลักการพื้นฐานที่เรียกว่า Lucifer cipher ที่พัฒนาขึ้นโดย IBM

DES นี้ จะใช้คีย์ขนาด 56 บิต นำมาเข้ารหัสข้อมูลที่แบ่งเป็นบล็อกของข้อมูล ขนาดบล็อกละ 64 บิต และผลของข้อมูลหลังจากเข้ารหัสแล้วก็จะออกมามีขนาด 64 บิตเช่นเดียวกัน

สำหรับรูปแบบกว้าง ๆ ของการใช้งานคีย์เดี่ยวของ DES เป็นดังรูป 5 - 5



รูปที่ 5 - 5 รูปแบบการทำงานของ DES

ซึ่งมีหลักการทำงานดังนี้ อินพุท ซึ่งเป็นข้อมูลที่เราจะส่งไปซึ่งเราแบ่งเป็นบล็อกข้อมูลขนาด 64 บิตนี้ จะถูกสลับสับเปลี่ยนบิตของข้อมูลซึ่งสุดท้ายก็ยังคงได้ข้อมูลขนาด 64 บิตที่ถูกสับบิตข้อมูลแล้ว จากนั้นจะนำบิตคีย์ขนาด 56 บิต มาสร้างเป็นคีย์ขนาด 48 บิต จำนวน 16 คีย์ สำหรับใช้ในการเข้ารหัสให้กับอินพุทข้างต้นรอบละ 1 คีย์เป็นจำนวน 16 รอบ โดยที่แต่ละรอบของการเข้ารหัสข้อมูลขนาด 48 บิต กับอินพุทขนาด 64 บิตนี้จะได้เอาท์พุทขนาด 64 บิตออกมา ซึ่งจะกลายเป็นอินพุทของรอบต่อไปเรื่อย ๆ จนจบ 16 รอบ เมื่อได้เอาท์พุทในรอบที่ 16 ออกมาแล้ว ข้อมูลอันนี้จะนำไปสับเปลี่ยนบิตอีกครั้งหนึ่ง โดยครั้งนี้จะใช้หลักการสับเปลี่ยนบิต (จะกล่าวถึงต่อไป) ที่กลับกับการสับในครั้งแรก หรือ อาจกล่าวได้ว่าเป็นอินเวอร์สของการสับข้างต้นนั่นเอง

สำหรับการถอดรหัส เราต้องเริ่มต้นโดยการสับเปลี่ยนบิตโดยใช้หลักการสับเปลี่ยนดังที่ใช้กับการสับเปลี่ยนบิตในตอนแรก นั่นคือเป็นการยกเลิก (undo) การสับเปลี่ยนในตอนสุดท้าย จากนั้นนำคีย์ที่ใช้ในการเข้ารหัสในตอนแรกมาใส่ โดยใส่กลับกันคือ ใส่รหัสขนาด 48 บิตที่รหัสที่ 16 ก่อน และสุดท้ายที่รหัสที่ 1 จากนั้นจะได้ข้อมูลซึ่งจะต้องไปทำการสับ โดยใช้วิธีการสับที่กลับกับการสับครั้งแรก หรือใช้หลักการสับของการเข้ารหัสในตอนสุดท้ายนั่นเอง

สำหรับการทำงานของ DES จะขออธิบายอย่างละเอียดดังนี้

การสับบิตของข้อมูล (Permutations of the Data)

การสับบิตของข้อมูล เราเรียกในภาษาอังกฤษว่า Permutation โดยการทำการสับบิตข้อมูลนี้ จะทำ 2 ครั้ง คือ ครั้งแรกที่ข้อมูลเข้ามา และตอนสุดท้ายที่จะใช้การสับแบบอินเวอร์ส แต่อย่างไรก็ตาม การสับบิตของข้อมูลนี้ก็ไม่ได้เพิ่มความปลอดภัยของข้อมูลมากเท่าใด เนื่องจากถ้าผู้ที่จะลักลอบดูข้อมูลที่รู้หลักการทำงานของ DES ซึ่งมีการเผยแพร่อยู่แล้ว สามารถที่จะศึกษา และถอดการสับบิดข้อมูลนี้ได้ แต่เหตุที่ต้องมีการสับบิดเพื่อที่จะทำให้การถอดรหัสของข้อมูลที่ผ่าน DES สามารถทำได้ยากขึ้นในซอฟท์แวร์เนื่องด้วยต้องอาศัยการทำงานของฮาร์ดแวร์อย่างมาก

การสับบิดของข้อมูลทำได้ดังนี้

Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

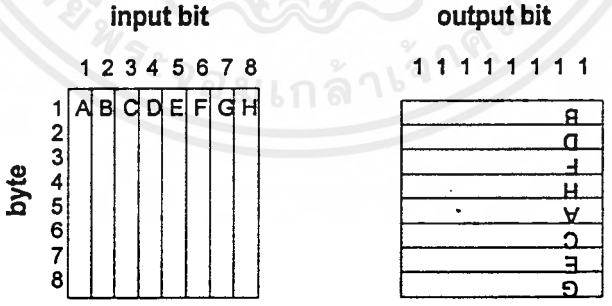
Final Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

รูปที่ 5 - 6 การสับบิตของข้อมูล

รูปที่ 5 - 6 แสดงหมายเลขบิตของข้อมูลที่ผ่านการสับบิตของการสับบิตครั้งแรก (Initial Permutation) และการสับบิตครั้งสุดท้าย (Final Permutation) พิจารณาภาพแรกของรูปที่ 5 - 6 เอาท์พุทของข้อมูลที่ผ่านการสับบิตที่จะนำไปยังขั้นตอนต่อไปจะนำบิตของอินพุทบิตที่ 58 มาไว้เป็นบิตที่ 1 ในเอาท์พุท เอาบิตที่ 50 มาไว้เป็นบิตที่ 2 ในเอาท์พุท เอาบิตที่ 7 ของอินพุทมาไว้เป็นบิตสุดท้ายของเอาท์พุทเป็นต้น

แต่อย่างไรก็ดี การสับบิตของข้อมูลนี้ไม่ใช่เป็นการทำอย่างสุ่ม (Random) แต่เรามีหลักดังรูป 5 - 7



รูปที่ 5 - 7 อัลกอริธึมในการสับบิตข้อมูล

อินพุทจะแบ่งเป็น 8 ไบต์ เช่นเดียวกับเอาท์พุทที่มี 8 ไบต์เช่นกัน โดยแต่ละบิตของไบต์แรกของอินพุทจะนำไปเป็นบิตที่ 8 ของแต่ละไบต์ในเอาท์พุท (ดูภาพประกอบ) แต่ละบิตในไบต์ที่ 2 จะนำไปเป็นบิตที่ 7 ของแต่ละไบต์ในเอาท์พุท นั่นคือ แต่ละบิตของไบต์ที่ *i* ของอินพุทจะนำไปเป็นบิตที่ (9-*i*) ของทุกไบต์

โดยการนำแต่ละบิตไปใส่เป็นเอาท์พุทในแต่ละไบต์นี้ จะนำบิตคู่ไปใส่ในไบต์ที่ 1-4 ของเอาท์พุท และบิตคี่ไปเป็นไบต์ที่ 5-8 (ดูรูป 5 - 7 ประกอบ)

การสร้างคีย์ในแต่ละรอบ (Generating the Per-Round Keys)

ในการสร้างคีย์นี้ ในตอนแรกเราจะมีคีย์ขนาด 64 บิต หรือ 8 ไบต์เช่นกัน แต่เราจะไม่นำบิตที่ 8,16,...,64 ซึ่งเป็นบิตพาริตี้มาใช้ นั่นคือ จะเหลือ 56 บิตนั่นเอง จากนั้นบิตทั้ง 56 บิตนี้จะนำมาสับเปลี่ยน โดยแบ่งคีย์เป็น 2 ส่วน ส่วนละ 28 บิต ซึ่งเราเรียกว่า C_0 และ D_0 (0 หมายถึง เป็นคีย์เริ่มต้น) และทำการสับดังรูปที่ 5 - 8

C_0

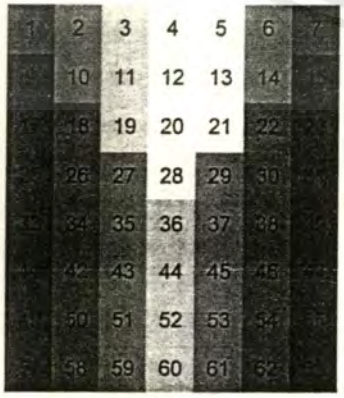
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D_0

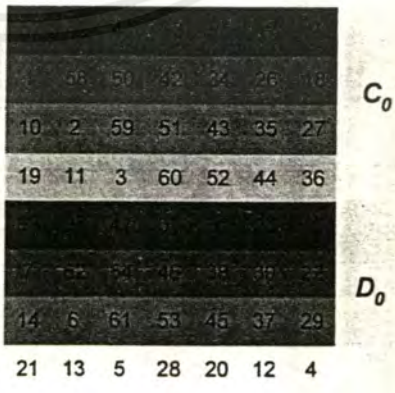
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	60	52	44	36

รูปที่ 5 - 8 คีย์ขนาด 56 บิต

โดยหลักการก็ดูเช่นเดียวกับการสับบิตข้อมูลที่กล่าวถึงในตอนแรกนั่นเอง และเราจะสังเกตว่าบิตที่ 8, 16, ..., 64 ไม่ได้ใช้เช่นกัน และการสับบิตข้อมูลนี้ก็ไม่ได้ทำอย่างสุ่มเช่นกัน โดยจะใช้อัลกอริธึมในการสับบิตดังรูปที่ 5 - 9

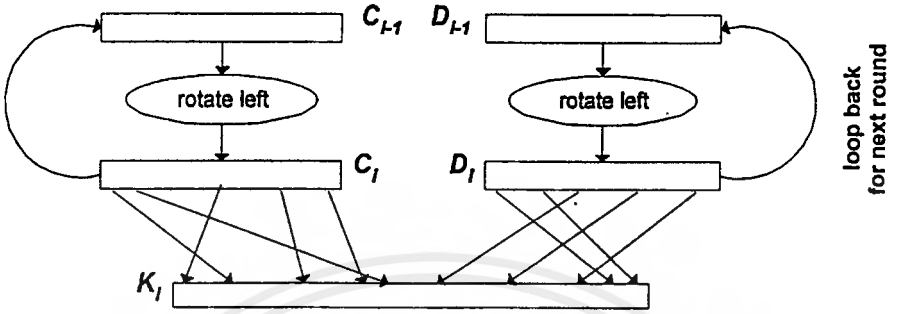


คีย์ขนาด 56 บิต



รูปที่ 5 - 9 อัลกอริธึมในการสับบิตคีย์

จากนี้ เราจะมาเริ่มสร้างคีย์ที่ใช้ในแต่ละรอบ ซึ่งเราขอเรียกคีย์แต่ละคีย์ในแต่ละรอบว่า K_i โดย i แทนรอบแต่ละรอบจาก 1-16 ดังรูปที่ 5 - 10



รูปที่ 5 - 10 การสร้างคีย์ที่ใช้ในแต่ละรอบ

นั่นคือ ใน C และ D จะหมุนบิดไปทางซ้าย (Rotate left) โดยจำนวนของบิตที่หมุนจะต่างกัน โดยในรอบที่ 1 , 2 , 9 และ 16 จะหมุนไปทางซ้าย 1 บิด และที่เหลือจะหมุน 2 บิต

และจะนำไปสับเปลี่ยนบิต ซึ่งส่วนนี้ถือว่าเป็นส่วนที่ทำให้เกิดความปลอดภัยของข้อมูลใน DES นี้ โดย จะนำ C และ D ที่ผ่านการหมุน และสับเปลี่ยนบิดมาสร้างเป็นคีย์ (K_i) โดย C เป็นครึ่งหน้า และ D เป็นครึ่งหลัง และการสับเปลี่ยนบิดของ C และ D จะทำดังนี้ (ดูรูปที่ 5 - 10 ประกอบด้วย)

การสับเปลี่ยนบิดของบิตครึ่งหน้า หรือ C ของคีย์ จะสังเกตว่า บิดที่ 9 , 18 , 22 และ 25 ไม่ได้ใช้ จะได้ C ดังนี้

14 17 11 24 1 5
3 28 15 6 21 10
23 19 12 4 26 8
16 7 27 20 13 2

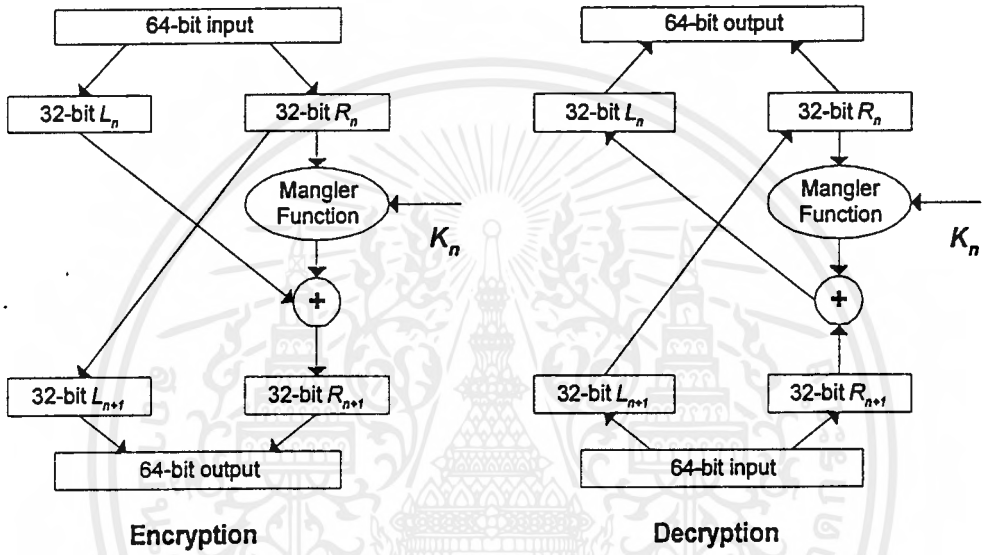
และ D จะไม่นำบิดที่ 35 , 38 , 43 และ 54 ไม่นำมาใช้ ซึ่งจะได้ดังนี้

41 52 31 37 47 55
30 40 51 45 33 48
44 49 39 56 34 53
46 42 50 36 29 32

เมื่อรวม C และ D ก็จะได้ 48 บิตตามต้องการ

การเข้ารหัสในแต่ละรอบ (DES Round)

ในแต่ละรอบของการเข้ารหัสจะมีวิธีเข้ารหัสดังรูปที่ 5 - 11



รูปที่ 5 - 11 วิธีการเข้ารหัสในแต่ละรอบ

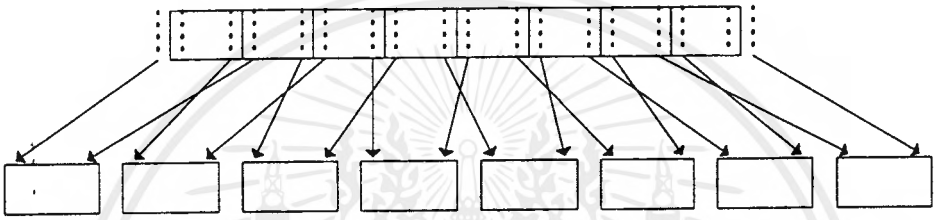
ซึ่งในภาพจะแสดงทั้งการเข้ารหัส และถอดรหัส โดยหลักการดังนี้

อินพุท 64 บิต จะถูกแบ่งเป็น 2 ส่วน ส่วนละ 32 บิต ซึ่งเรียกว่า L_n และ R_n เมื่อดูจากรูปที่ 5 - 11 จะเห็นว่า R_n จะกลายเป็นเอาท์พุท L_{n+1} และ R_n ก็จะนำไปผ่านขั้นตอนที่เรียกว่า mangler function (จะกล่าวถึงต่อไป) กับคีย์ที่เราสร้างขึ้นในแต่ละรอบ เมื่อได้ผลก็จะนำไป XOR กับ L_n และได้เป็น R_{n+1} ออกมา จากนั้นนำ L_{n+1} และ R_{n+1} มารวมกัน (concatenation) ก็จะได้เป็นเอาท์พุทขนาด 64 บิตตามต้องการในแต่ละรอบ

จะสังเกตว่า ในขั้นตอน mangler function จะเป็นการนำอินพุทขนาด 32 บิต ไปทำกับคีย์ขนาด 48 บิต และได้เป็นเอาท์พุทขนาด 32 บิตออกมา

Mangler Function

จากที่เราได้กล่าวถึงข้างต้น อินพุทจะเป็น R_n ขนาด 32 บิต ในที่นี้ขอเรียกว่า R และคีย์ K_n ขนาด 48 บิต ซึ่งขอเรียกว่า K และจะได้เป็นเอาท์พุทขนาด 32 บิต ซึ่งจะนำไป XOR กับ L_n ต่อไป การทำ mangler function ดอนแรกจะนำ R ขนาด 32 บิต มาขยายเป็น 48 บิต ดังรูปที่ 5 - 12



รูปที่ 5 - 12 การขยายขนาดของ R ใน mangler function

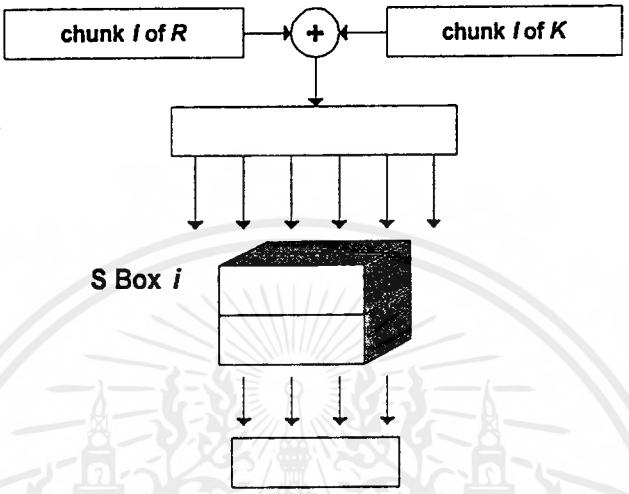
เราทำได้โดยแบ่ง R เป็นส่วน ที่เรียกว่า chunk ขนาดส่วนละ 4 บิต และขยาย chunk ขนาด 4 บิต นี้เป็น 6 บิต โดยนำเอาบิตที่ติดกับบิตต้นและปลายของแต่ละ chunk มารวมด้วย (ดูรูปที่ 5 - 12 ประกอบด้วย) และบิตที่อยู่ขวาสุดและซ้ายสุด (ของ R) ถือเป็นบิตที่อยู่ติดกันของ chunk ริมสุดสองข้างด้วย

K ก็จะถูกแบ่งเป็น 8 ส่วน ส่วนละ 6 บิตเช่นกัน และเรียกว่า chunk เช่นกัน

จากนั้นจะนำแต่ละ chunk ของ R และ K มา XOR กัน โดย chunk i ของ R จะนำมา XOR กับ chunk i ของ K ซึ่งจะได้เป็นเอาท์พุทขนาด 6 บิต จากนั้นจะนำเอาท์พุทขนาด 6 บิตนี้ไปใส่ใน S box เพื่อสร้างเป็นเอาท์พุทขนาด 4 บิต จากอินพุท 4 บิตที่ใส่เข้าไป

โดยเราจะเห็นว่า อินพุทที่สามารถเป็นได้ (6 บิต) คือ 64 แบบ และเอาท์พุท (4 บิต) จะเป็นได้ 16 แบบ S box จะทำหน้าที่ในการเปรียบเทียบอินพุทที่เข้ามาเพื่อสร้างเป็นเอาท์พุท โดยจะมีรูปแบบ หรือตารางการเปรียบเทียบเป็นการเฉพาะในแต่ละบิตของข้อมูล (จะกล่าวถึงต่อไป) ซึ่งอินพุท 4 ค่าที่เป็นไปได้ จะจับได้เป็นเอาท์พุท 1 ค่า

ดังภาพข้างล่างเป็นการแสดงหลักการทำงานของ mangler function และตารางการเปรียบเทียบ



รูปที่ 5 - 13 หลักการทำงานของ mangler function

Input	Input bits 2 thru 5
1,6	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	1110 0100 1101 0001 0010 1111 1011 1000 0011 1010 0110 1100 0101 1001 0000 0111
01	0000 1111 0111 0100 1110 0010 1101 0001 1010 0110 1100 1011 1001 0101 0011 1000
10	0100 0001 1110 1000 1101 0110 0010 1011 1111 1100 1001 0111 0011 1010 0101 0000
11	1111 1100 1000 0010 0100 1001 0001 0111 0101 1011 0011 1110 1010 0000 0110 1101

ตารางที่ 5 - 1 ผลลัพธ์ขนาด 4 บิตจาก S box 1 (บิต 1 - 4)

Input	Input bits 8 thru 11
7,12	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	1111 0001 1000 1110 0110 1011 0011 0100 1001 0111 0010 1101 1100 0000 0101 1010
01	0011 1101 0100 0111 1111 0010 1000 1110 1100 0000 0001 1010 0110 1001 1011 0101
10	0000 1110 0111 1011 1010 0100 1101 0001 0101 1000 1100 0110 1001 0011 0010 1111
11	1101 1000 1010 0001 0011 1111 0100 0010 1011 0110 0111 1100 0000 0101 1110 1001

ตารางที่ 5 - 2 ผลลัพธ์ขนาด 4 บิตจาก S box 2 (บิต 5 - 8)

Input	Input bits 14 thru 17
13,18	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	1010 0000 1001 1110 0110 0011 1111 0101 0001 1101 1100 0111 1011 0100 0010 1000
01	1101 0111 0000 1001 0011 0100 0110 1010 0010 1000 0101 1110 1100 1011 1111 0001
10	1101 0110 0100 1001 1000 1111 0011 0000 1011 0001 0010 1100 0101 1010 1110 0111
11	0001 1010 1101 0000 0110 1001 1000 0111 0100 1111 1110 0011 1011 0101 0010 1100

ตารางที่ 5 - 3 ผลลัพธ์ขนาด 4 บิตจาก S box 3 (บิต 9 - 12)

Input	Input bits 20 thru 23
19,24	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	0111 1101 1110 0011 0000 0110 1001 1010 0001 0010 1000 0101 1011 1100 0100 1111
01	1101 1000 1011 0101 0110 1111 0000 0011 0100 0111 0010 1100 0001 1010 1110 1001
10	1010 0110 1001 0000 1100 1011 0111 1101 1111 0001 0011 1110 0101 0010 1000 0100
11	0011 1111 0000 0110 1010 0001 1101 1000 1001 0100 0101 1011 1100 0111 0010 1110

ตารางที่ 5 - 4 ผลลัพธ์ขนาด 4 บิตจาก S box 4 (บิต 13 - 16)

Input	Input bits 26 thru 29
25,30	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	0010 1100 0100 0001 0111 1010 1011 0110 1000 0101 0011 1111 1101 0000 1110 1001
01	1110 1011 0010 1100 0100 0111 1101 0001 0101 0000 1111 1010 0011 1001 1000 0110
10	0100 0010 0001 1011 1010 1101 0111 1000 1111 1001 1100 0101 0110 0011 0000 1110
11	1011 1000 1100 0111 0001 1110 0010 1101 0110 1111 0000 1001 1010 0100 0101 0011

ตารางที่ 5 - 5 ผลลัพธ์ขนาด 4 บิตจาก S box 5 (บิต 17 - 20)

Input	Input bits 32 thru 35
31,36	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	1100 0001 1010 1111 1001 0010 0110 1000 0000 1101 0011 0100 1110 0111 0101 1011
01	1010 1111 0100 0010 0111 1100 1001 0101 0110 0001 1101 1110 0000 1011 0011 1000
10	1001 1110 1111 0101 0010 1000 1100 0011 0111 0000 0100 1010 0001 1101 1011 0110
11	0100 0011 0010 1100 1001 0101 1111 1010 1011 1110 0001 0111 0110 0000 1000 1101

ตารางที่ 5 - 6 ผลลัพธ์ขนาด 4 บิตจาก S box 6 (บิต 21 - 24)

Input	Input bits 38 thru 41
37,42	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	0100 1011 0010 1110 1111 0000 1000 1101 0011 1100 1001 0111 0101 1010 0110 0001
01	1101 0000 1011 0111 0100 1001 0001 1010 1110 0011 0101 1100 0010 1111 1000 0110
10	0001 0100 1011 1101 1100 0011 0111 1110 1010 1111 0110 1000 0000 0101 1001 0010
11	0110 1011 1101 1000 0001 0100 1010 0111 1001 0101 0000 1111 1110 0010 0011 1100

ตารางที่ 5 - 7 ผลลัพธ์ขนาด 4 บิตจาก S box 7 (บิต 25 - 28)

Input	Input bits 44 thru 47
43,48	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00	1101 0010 1000 0100 0110 1111 1011 0001 1010 1001 0011 1110 0101 0000 1100 0111
01	0001 1111 1101 1000 1010 0011 0111 0100 1100 0101 0110 1011 0000 1110 1001 0010
10	0111 1011 0100 0001 1001 1100 1110 0010 0000 0110 1010 1101 1111 0011 0101 1000
11	0010 0001 1110 0111 0100 1010 1000 1101 1111 1100 1001 0000 0011 0101 0110 1011

ตารางที่ 5 - 8 ผลลัพธ์ขนาด 4 บิตจาก S box 8 (บิต 29 - 32)

เมื่อนำเอาท์พุท 4 บิตมารวมกันทั้งหมดแล้ว ก็จะได้เอาท์พุทขนาด 32 บิต จาก S box 8 S box ซึ่ง 32 บิตนี้จะนำมาสับเปลี่ยนบิต (permutation) โดยการสับเปลี่ยนในขั้นตอนนี้จะมีความสำคัญมาก เพราะจะกลายเป็นอินพุทของรอบต่อ ๆ ไป โดยหลักการสับเปลี่ยนจะเป็นดังรูปที่ 5 - 14

16,7,20,21,29,12,28,17	1,15,23,26,5,18,31,10	2,8,24,14,32,27,3,9	19,13,30,6,22,11,4,25

รูปที่ 5 - 14 การสับบิตผลลัพธ์ขนาด 32 บิตจาก S box

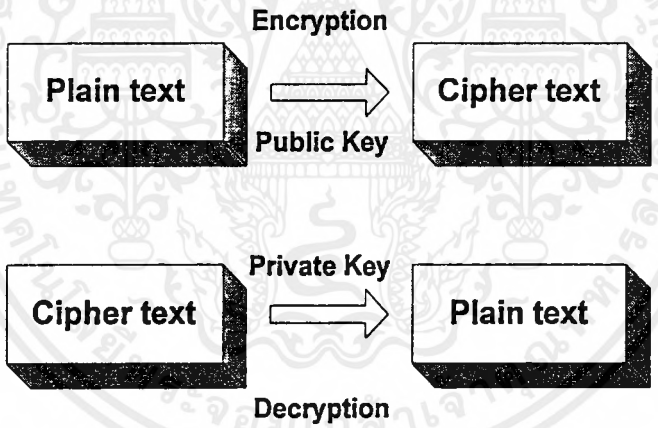
ซึ่งการทำงานทั้งหมดจะเป็น 16 รอบด้วยกัน เมื่อผ่านทั้ง 16 รอบแล้วก็จะผ่าน Final Permutation ดังที่กล่าวมาแล้วข้างต้น ก็จะได้ข้อมูลที่ผ่านการเข้ารหัสที่เรียบร้อยของ DES

สำหรับการทำ Secret Key Cryptography ยังมีอีกหลายแบบ เช่น Internation Data Encryption Algorithm (IDEA) เป็นต้น

Public Key Cryptography

บางครั้งเราเรียกว่า Asymmetric cryptography โดย Public Key Cryptography จะใช้คีย์ 2 คีย์ คือ Private key ซึ่งเป็นคีย์ที่รู้เฉพาะคน ไม่เปิดเผยให้คนอื่นรู้ และ Public key ซึ่งจะเปิดเผยให้คนอื่นได้รับรู้ เพื่อสามารถติดต่อกับเจ้าของคีย์ได้

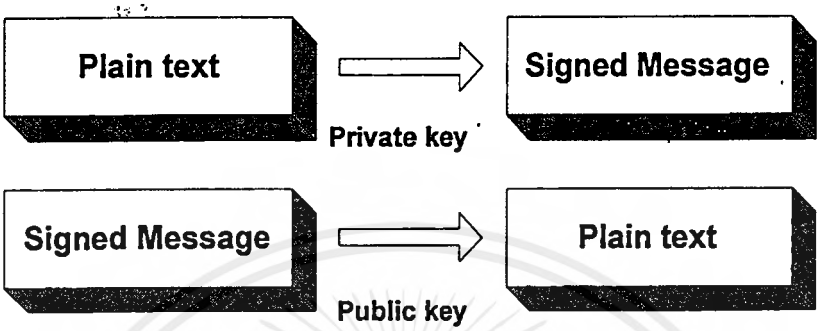
รูปแบบการทำงานของ Public Key Cryptography สามารถอธิบายได้ด้วยภาพง่าย ๆ ดังรูปที่ 5 - 15



รูปที่ 5 - 15 Public Key Cryptography

โดยตอนแรกข้อมูลเริ่มต้น คือ plain text จะถูกเข้ารหัสเพื่อส่งไป โดยการเข้ารหัสเราจะใช้รหัส public key ของคนที่เราต้องการจะส่งไปให้ (แต่ละคนจะมีรหัส public key เฉพาะตัว ที่จะเปิดเผยต่อคนทั่ว ๆ ไปได้) ซึ่งจะได้เป็นข้อมูลที่ถูกเข้ารหัสแล้ว ก็คือ cipher text จากนั้น เมื่อผู้รับได้รับข้อมูลแล้ว ก็จะใช้รหัส private key ของตัวเองในการถอดรหัส ก็จะได้เป็นข้อมูลต้นฉบับ (plaintext) กลับคืนมา

นอกจากนี้ เรายังใช้หลักของ Public Key Cryptography ในการสร้างเป็นลายเซ็นดิจิตอล (Digital signature) เวลาที่เราต้องการส่งข้อมูลไปยังที่ใด และปลายทางสามารถตรวจสอบว่าเป็นของเราอย่างแน่นอนได้ โดยมีหลักการดังรูปที่ 5 - 16



รูปที่ 5 - 16 การใช้ Public Key Cryptography ในการสร้างลายเซ็นดิจิตอล

โดยตอนแรก เราจะส่งข้อมูลไปโดยใช้คีย์ของเรา (private key) ก็จะได้ข้อมูลที่ถูกเซ็นชื่อเรียบร้อยแล้วว่าเป็นของเรา จากนั้นปลายทางจะใช้ public key ของเราเพื่อตรวจสอบว่าเป็นของเราจริงหรือเปล่า ถ้าใช่ก็สามารถถอดรหัสออกมาได้ โดยขั้นตอนนี้เราเรียกว่า Verification โดย public key และ private key ของแต่ละคนจะจับคู่กันอยู่แล้ว ทำให้เราสามารถตรวจสอบได้

อย่างไรก็ตามการใช้งาน Public Key Cryptography จำเป็นจะต้องอาศัยพื้นฐานความรู้ทางด้านทฤษฎีจำนวน (Number Theory) ค่อนข้างมาก

การใช้งาน Public Key Cryptography ไม่มีรูปแบบที่แน่นอน แต่อาศัยหลักการพื้นฐานเหมือนกันดังที่กล่าวมาข้างต้น ซึ่งวิธีการเข้ารหัสแบบ Public Key Cryptography มีหลายวิธีด้วยกัน ได้แก่ RSA, El Gamal, DSS, Diffie-Hellman เป็นต้น แต่ในรายงานนี้จะขอนำเสนอรูปแบบเดียวที่ค่อนข้างจะเป็นที่นิยม คือ RSA รวมถึงทฤษฎีจำนวนอย่างคร่าว ๆ บางเรื่องที่จำเป็นในการใช้งาน RSA ด้วย

ทฤษฎีจำนวนที่จำเป็นต้องรู้ (Modular Arithmetic)

การใช้งาน Public Key Cryptography ส่วนใหญ่มักจะอาศัยหลักการพื้นฐานของ Modular Arithmetic ค่อนข้างมาก Modular Arithmetic คือ เลขจำนวนเต็มที่ไม่เป็นจำนวนลบ ที่มีค่าน้อยกว่าเลขจำนวนเต็มบวก n เสมอ สำหรับการหา Modular Arithmetic จะอาศัยการคำนวณทางคณิตศาสตร์ทั่ว ๆ ไป เช่น การบวก หรือคูณ แล้วแต่ชนิด จากนั้นนำมาหารด้วย n ค่าที่ต้องการก็คือเศษเหลือจากการหารนั่นเอง โดยผลที่ได้นี้เราเรียกว่า modulo n หรือ mod n โดยหากเขียนว่า x mod n จะหมายถึงเศษจากการหาร x ด้วย n นั่นเอง

Modular Addition

คือ การหาค่าเศษเหลือ หรือ mod n จากผลบวก ในที่นี้เราจะขออธิบายในกรณี n = 10 เนื่องจากสามารถอธิบายให้เห็นภาพได้ง่ายที่สุด ค่าที่ได้จาก mod n นี้จะต้องอยู่ระหว่าง 0 ถึง 9 เช่น หาค่า mod n ของค่าดังต่อไปนี้

5 + 5 = 0
3 + 9 = 2
2 + 2 = 4
9 + 9 = 8 เป็นต้น

เราสามารถดูผลที่เกิดจากการหาค่าทั้งหมดในกรณี n = 10 ได้ดังตารางที่ 5 - 9

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

ตารางที่ 5 - 9 ผลลัพธ์ของ Modular Addition ใน modulo 10

นอกจากนี้ เรายังจะสามารถหาค่าที่เรียกว่า อินเวอร์สการบวก (Additive Inverse) ซึ่ง อินเวอร์สการบวกของเลข x ใน modulo n คือเลขที่นำไปบวกกับ x แล้วได้ผลลัพธ์เท่ากับ 0 mod n เช่น อินเวอร์สการบวกของ 4 ใน modulo 10 ก็คือ 6 เพราะว่า $4 + 6 = 0 \bmod 10$

เราสามารถนำหลักการของ Modular Addition มาใช้ในการเข้ารหัสข้อมูลอย่างง่าย ๆ ได้ โดยการกำหนดคีย์ที่จะใช้ในการเข้ารหัส แล้วนำคีย์นั้นไปบวกกับ plain text ซึ่งจะได้ผลลัพธ์จากการ mod n ตามตาราง ส่วนการถอดรหัส ก็สามารถทำได้โดยการนำ cipher text ที่ได้มาบวกกับอินเวอร์สการบวกของคีย์ที่ใช้ในการเข้ารหัส ยกตัวอย่างเช่นนำ 4 เป็นคีย์ในการเข้ารหัส (ทำโดยบวกข้อมูลเข้ากับ 4 และนำผลลัพธ์จากการบวกไป mod 10) และการถอดรหัสก็จะใช้ 6 ซึ่งเป็นอินเวอร์ส การบวกของ 4 ในการถอดรหัสนั่นเอง

Modular Multiplication

คือ การนำผลคูณมาหารด้วย n แล้วนำเศษมาใช้นั่นเอง ตารางหน้าถัดไปจะแสดงผลการหา Modular Multiplication ในกรณี $n = 10$ เราจะเห็นว่า การคูณด้วย 1 , 3 , 7 และ 9

จะสามารถใช้ในการเข้ารหัสได้ เนื่องจากค่าผลลัพธ์ที่ได้เป็นการแทนข้อมูลแบบหนึ่งต่อหนึ่ง คือจะให้ค่าตัวเลขที่แตกต่างกัน ในการคูณค่าที่ไม่ซ้ำกันในช่วง 0 ถึง 9

เช่น 1 * 1 = 1

1 * 2 = 2

.

.

.

1 * 9 = 9

ซึ่งแต่ละค่าไม่ซ้ำกันเลย แต่ถ้าคูณกับค่าอื่น ๆ (0 , 2 , 4 , 5 , 6 , 8) จะให้ค่าที่มีค่าซ้ำ ทำให้ในกรณีที่เราเข้ารหัสหนึ่ง เมื่อนำไปถอดรหัสอาจจะไม่สามารถทำได้อย่างถูกต้องได้ ตัวอย่างเช่น หากนำ 5 ไปใช้ในการเข้ารหัส ก็จะได้ผลลัพธ์เป็น 0 หรือ 5 เท่านั้น และเมื่อต้องการถอดรหัส cipher ที่มีค่า 5 เราก็จะไม่สามารถทราบได้ว่าควรจะถอดรหัสให้เป็น 1 , 3 , 5 , 7 หรือ 9

*	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

ตารางที่ 5 - 10 ผลลัพธ์ของ Modular Multiplication ใน modulo 10

ในส่วนของการถอดรหัส สามารถทำได้โดยการคูณ cipher ด้วยค่าที่เรียกว่า อินเวอร์สการคูณ (Multiplicative inverse) ของคีย์ที่ใช้ในการเข้ารหัส ซึ่งค่าอินเวอร์สการคูณของเลข x ใด ๆ ใน modulo n ก็คือเลขที่คูณกับ x แล้วได้ผลลัพธ์เท่ากับ 1 mod n เช่น อินเวอร์สการคูณของ 7 ใน modulo 10 คือ 3 เนื่องจาก 3 * 7 = 1 mod 10 ซึ่งจากตารางจะเห็นว่า

เฉพาะ 1 , 3 , 7 และ 9 เท่านั้นที่มีค่าอินเวอร์สการคูณ คือ 3 เป็นอินเวอร์สการคูณของ 7 , 9 เป็นอินเวอร์ส การคูณของ 9 และ 1 เป็นอินเวอร์สการคูณของ 1

ในกรณีที่เลข n มีค่ามาก ๆ เช่น เลข n ที่มีขนาด 100 หลัก จะเห็นว่าการหาอินเวอร์สการคูณโดยการคำนวณตามปกติจะยากมากจนแทบเป็นไปไม่ได้ อย่างไรก็ตาม มีวิธีการหาค่าอินเวอร์สการคูณใน modulo n ซึ่งมีประสิทธิภาพเรียกว่า อัลกอริธึมของยูคลิด (Euclid's Algorithm) ซึ่งเราจะไม่กล่าวถึงอย่างละเอียดในรายงานนี้

จากที่เรากล่าวถึงข้างต้นว่าเฉพาะเลข 1 , 3 , 7 และ 9 เท่านั้นที่มีอินเวอร์สการคูณ ทั้งนี้เนื่องจากเลขทั้ง 4 ตัวนี้เป็น Relatively Prime กับ 10 ซึ่ง Relative Prime ของ n หมายถึงตัวเลขนั้นและ n มี 1 เป็นตัวหารร่วมมาก (ห.ร.ม.)

จากข้อสรุปที่ว่า ตัวเลขทุกตัวที่เป็น Relatively Prime กับ n จะสามารถหาค่าอินเวอร์ส การคูณของเลขนั้นได้ ส่วนตัวเลขอื่น ๆ นอกเหนือจากนี้จะไม่สามารถหาอินเวอร์สการคูณได้ และตัวเลขที่เป็น Relatively Prime กับ n ยังสามารถนำไปใช้ในการเข้ารหัสได้โดยใช้หลักการ Modular Multiplication ได้ เนื่องจากเมื่อเราใช้ตัวเลขนี้เป็นคีย์เข้ารหัสแล้ว เราจะสามารถใช้อินเวอร์สการคูณของตัวเลขเหล่านี้เป็นคีย์ในการถอดรหัสได้

ในที่นี้เราจะขอกล่าวถึงคุณสมบัติอีกข้อหนึ่ง เรียกว่า Totient Function (มาจากคำว่า Total และ Quatient) ซึ่งเราใช้สัญลักษณ์แทนว่า $\phi(n)$ โดยค่านี้จะเป็นตัวที่บอกว่ามีเลขกี่จำนวนที่น้อยกว่า n และมีคุณสมบัติ Relatively Prime กับ n ดังนั้นถ้า n เป็นจำนวนเฉพาะ (prime) จะได้ว่าจำนวนเต็ม 1 , 2 , ... , n-1 ล้วนเป็น Relatively Prime กับ n เพราะฉะนั้นจะได้ $\phi(n) = n-1$ นั่นเอง

แต่ถ้า n เกิดจากการคูณกันของจำนวนเฉพาะ 2 จำนวน คือ p และ q เพราะฉะนั้น จำนวนเต็มที่ Relatively Prime กับ n สามารถหาได้โดยพิจารณาว่า

มีเลขทั้งหมด pq จำนวนใน {0 , 1 , 2 , 3 , ... , n-1}
เลขที่มี p เป็นตัวประกอบ มีทั้งหมด q จำนวน
เลขที่มี q เป็นตัวประกอบ มีทั้งหมด p จำนวน

จึงมีเลขทั้งหมด $p+q-1$ (ต้องนับเลข 0 ครั้งเดียว) ที่เป็น Relatively Prime กับ pq

ดังนั้นจึงมีเลขทั้งหมด $pq - (p+q-1) = (p-1)(q-1)$ จำนวนที่เป็น Relatively Prime กับ n

จะได้ $\phi(n) = (p-1)(q-1)$

Modular Exponentiation

เช่นเดียวกับค่า Modular อื่น ๆ ที่กล่าวถึงข้างต้น ดูตารางที่ 5 - 11 ประกอบ

x^y	0	1	2	3	4	5	6	7	0	9	10	11	12
0		0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	4	8	6	2	4	8	6	2	4	8	6
3	1	3	9	7	1	3	9	7	1	3	9	7	1
4	1	4	6	4	6	4	6	4	6	4	6	4	6
5	1	5	5	5	5	5	5	5	5	5	5	5	5
6	1	6	6	6	6	6	6	6	6	6	6	6	6
7	1	7	9	3	1	7	9	3	1	7	9	3	1
8	1	8	4	2	6	8	4	2	6	8	4	2	6
9	1	9	1	9	1	9	1	9	1	9	1	9	1

ตารางที่ 5 - 11 ผลลัพธ์ของ Modular Exponentiation ใน modulo 10

เราสามารถเขียนได้ว่า $4^6 = 6 \bmod 10$ เพราะว่า $4^6 = 4096$ แต่จากตารางที่ 5 - 11 เราจะสังเกตว่า $x^y \bmod n$ จะไม่เท่ากับ $x^{y+n} \bmod n$ ซึ่งไม่เหมือนกับใน Ordinary arithmetic เช่น $3^1 = 3 \bmod 10$ แต่ $3^{11} = 177147 = 7 \bmod 10$

จากตารางเราจะเห็นว่า เราสามารถใช้ 3 เป็นคีย์ในการเข้ารหัสได้ เนื่องจาก 3 จะให้ค่าที่แตกต่างกันจาก 0 ถึง 9 ในขณะที่ค่าอื่น ๆ จะให้ค่าซ้ำกันในบางค่าได้ เช่นหากใช้ 2 เป็นคีย์ในการเข้ารหัสโดยใช้ Modular Exponentiation จะมีกรณีเช่น 2^2 และ 8^2 ได้ 4 mod 10 ซึ่งทำให้ไม่สามารถทราบได้ว่าควรจะถอดรหัส cipher 4 ให้เป็น 2 หรือ 8 เป็นต้น

เมื่อพิจารณาจากตารางเราจะเห็นว่า ค่าในคอลัมน์ที่ 1 และ 5 เหมือนกัน เช่นเดียวกับค่าในคอลัมน์ที่ 2 และ 6 ที่เหมือนกัน และค่าในคอลัมน์ที่ 3 และ 7 ก็เหมือนกัน เราจึงสามารถสรุปได้ว่า $x^y \bmod n$ มีค่าเท่ากับ $x^{(y \bmod \phi(n))} \bmod n$ นั่นก็คือ ในกรณีของ n = 10 เราจะได้ค่าที่ Relatively prime กับ 10 ก็คือ 1 , 3 , 7 และ 9 เพราะฉะนั้น $\phi(n)$ = 4 นั่นคือ คอลัมน์ที่ i จะเท่ากับคอลัมน์ที่ $i + 4$ โดยข้อสรุปนี้จะเป็นจริงสำหรับค่า n ที่เป็นจำนวนเฉพาะ หรือเป็นผลคูณของจำนวนเฉพาะที่ไม่ซ้ำกันเท่านั้น

และเราสามารถสรุปได้ว่า ถ้า $y = 1 \bmod \phi(n)$ แล้วสำหรับจำนวน x ใด ๆ จะได้ว่า $x^y = x \bmod n$

RSA

RSA มาจาก Rivest , Shamir และ Adleman ซึ่งเป็นชื่อของศาสตราจารย์ 3 ท่านจากสถาบันเทคโนโลยีแห่งแมสซาชูเซส (MIT) ซึ่งเป็นผู้คิดค้นวิธีนี้ RSA อาศัยหลักของ Public Key Cryptography ซึ่งขนาดของคีย์ จะสามารถกำหนดได้ตามต้องการ ถ้าเรากำหนดคีย์ยาวก็จะทำให้ยากต่อการถอดรหัส หรือ ทำให้ปลอดภัยมากขึ้น แต่ถ้าเรากำหนดคีย์สั้น ก็จะทำให้มีประสิทธิภาพในการใช้งานมากยิ่งขึ้น แต่ตามปกติแล้วนิยมใช้ความยาวของคีย์เท่ากับ 512 บิต

เช่นเดียวกัน บล็อกของข้อมูล ก็สามารถกำหนดได้ตามต้องการ แต่บล็อกข้อมูล หรือ plain text นี้ จะต้องมีขนาดไม่เกินกว่าความยาวของคีย์ ส่วน cipher text ที่ได้จะมีขนาดเท่ากับขนาดของคีย์ การคำนวณสำหรับการเข้ารหัสแบบ RSA นี้ใช้เวลามากกว่าการคำนวณสำหรับการเข้ารหัสแบบ Secret Key algorithms เช่น DES , IDEA เป็นต้น ดังนั้น RSA จึงไม่นิยมใช้ในการเข้ารหัสข้อมูลที่ยาว ๆ แต่นิยมเข้ารหัสข้อมูลโดยใช้ secret key ก่อนแล้วจากนั้นจึงค่อยนำ secret key นั้นไปเข้ารหัสด้วยวิธี RSA อีกครั้งหนึ่ง

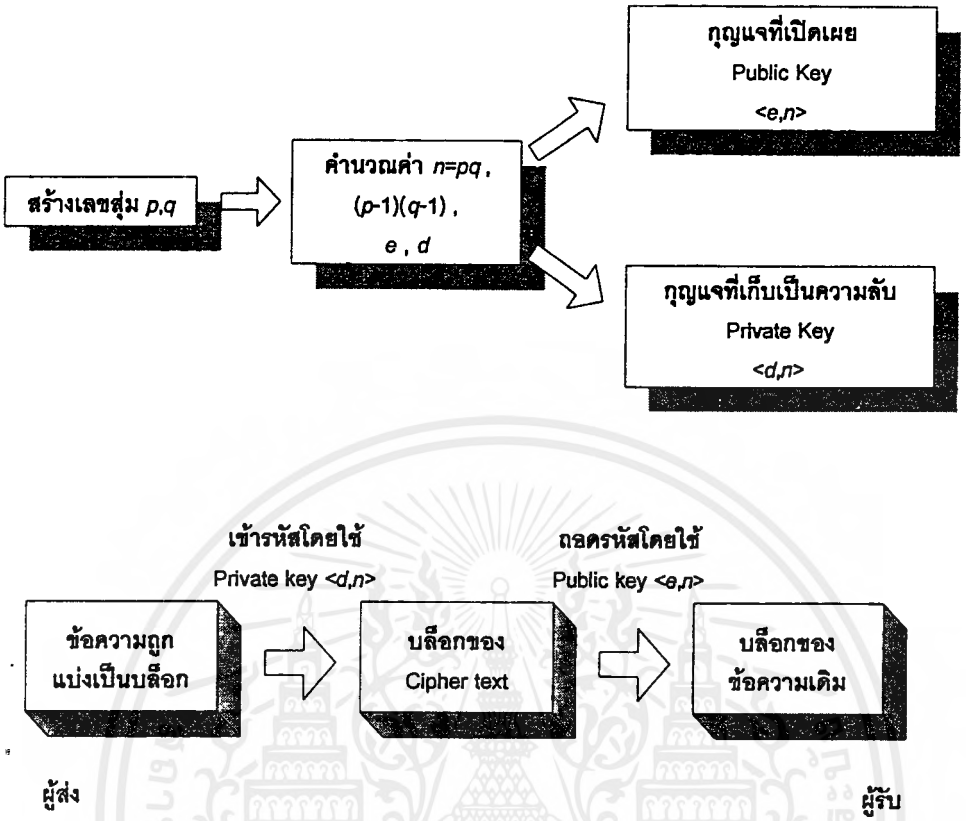
การทำงานของ RSA

การทำงานคร่าว ๆ ก็คือ เราต้องเริ่มต้นด้วยการสร้าง public key และ private key โดยเริ่มต้นด้วยการเลือกค่าจำนวนเฉพาะ p และ q ขนาดใหญ่พอควร (มักจะเลือกจำนวนเฉพาะที่มีขนาดประมาณ 256 บิต) คูณค่าทั้งสองนี้เข้าด้วยกัน จะได้ค่าผลลัพธ์ n ค่าที่เราเลือกมา คือ p และ q นี้จะถูกเก็บเป็นความลับตลอดเวลา

การสร้าง public key ทำได้โดยเลือกจำนวน e ที่มีคุณสมบัติ Relatively prime กับ $\phi(n)$ เนื่องจากเรารู้ค่า p และ q ทำให้เราสามารถรู้ค่า $\phi(n)$ ได้ นั่นคือเท่ากับ $(p-1)(q-1)$ จะได้ public key คือ $<e,n>$

การสร้าง private key ทำได้โดย หาค่า d ที่เป็นอินเวอร์สการคูณของ e mod $\phi(n)$ จะได้ค่า private key คือ $<d,n>$ เราสามารถมั่นใจว่าจะหาอินเวอร์สการคูณของ e mod $\phi(n)$ ได้เพราะว่าในการเลือกค่า e ในขั้นตอนแรกนั้นเราเลือกค่า e ที่เป็น Relatively prime กับ $\phi(n)$

การเข้ารหัสให้กับข้อมูล m ซึ่งมีขนาดบิดเล็กกว่า n โดยใช้ public key ทำได้โดยคำนวณ ciphertext $c = m^e \bmod n$ เมื่อต้องการจะถอดรหัสโดยใช้ private key ทำได้โดยคำนวณ message $m = c^d \bmod n$ เมื่อต้องการจะสร้างลายเซ็นดิจิตอล (Digital signature) ทำได้โดยเข้ารหัสข้อมูลด้วย private key จะได้ signed message $s = m^d \bmod n$ และผู้รับสามารถตรวจสอบลายเซ็นบนข้อมูลโดยการถอดรหัสโดยใช้ public key ของผู้ส่ง ได้โดยการคำนวณ message $m = s^e \bmod n$



รูปที่ 5 - 17 การสร้าง และการใช้งาน RSA

ในที่นี้จะขอยกตัวอย่างการเข้ารหัสแบบ RSA สักตัวอย่างหนึ่ง ดังนี้

ถ้าต้องการเข้ารหัสข้อความว่า "KING MONGKUT INSTITUTE OF TECHNOLOGY" โดยแทนตัวอักษรด้วยตัวเลขต่อไปนี้

A = 01 B = 02

C = 03 D = 04

E = 05 F = 06

G = 07 ..... Z = 26

BLANK = 00 จะสามารถแปลงข้อความข้างต้นเป็นตัวเลขได้ดังนี้

11091407001315140711212000091419200 9

20212005001506002005030814151215072 5

เลือก $p = 73$ และ $q = 151$ ฉะนั้น $n = pq = 11023$ เลือก $e = 11$

ขั้นตอนการเข้ารหัสแบ่งข้อความที่เข้ารหัสออกเป็นบล็อก ๆ ละ 4 หลัก จากนั้นหา mod n ดังนี้

$C_1 = 1109^{11} \bmod 11023 = 5902$

$C_2 = 1407^{11} \bmod 11023 = 8015$

$C_3 = 0013^{11} \bmod 11023 = 4385$

$C_4 = 1514^{11} \bmod 11023 = 8433$

$C_5 = 0711^{11} \bmod 11023 = 7922$

$C_6 = 2120^{11} \bmod 11023 = 1436$

$C_7 = 0009^{11} \bmod 11023 = 576$

$C_8 = 1419^{11} \bmod 11023 = 9127$

$C_9 = 2009^{11} \bmod 11023 = 3133$

$C_{10} = 2021^{11} \bmod 11023 = 9538$

$C_{11} = 2005^{11} \bmod 11023 = 7420$

$C_{12} = 0015^{11} \bmod 11023 = 10206$

$C_{13} = 0600^{11} \bmod 11023 = 4855$

$C_{14} = 2005^{11} \bmod 11023 = 7420$

$C_{15} = 0308^{11} \bmod 11023 = 9892$

$C_{16} = 1415^{11} \bmod 11023 = 10833$

$C_{17} = 1215^{11} \bmod 11023 = 5601$

$C_{18} = 0725^{11} \bmod 11023 = 3911$

จะสังเกตได้ว่าขนาดของการแบ่งบล็อกของข้อความจะต้องน้อยกว่าค่า n เพราะเซ็ททั้งหมดของ mod n จะอยู่ในช่วง $[0,n-1]$ เมื่อจะถอดรหัสจะทำได้โดยวิธีเดียวกันโดยใช้กุญแจถอดรหัสอีกตัวหนึ่งคือ 5891 เพราะว่า

$\phi(n) = (p\text{-}1)(q\text{-}1)$

$= (73\text{-}1)(151\text{-}1)$

$= 10800$

$ed = 1 \bmod \phi(n)$

และจาก $e = 11$

จะได้ $d = 5891$

$M1 = 5902^{5891} \bmod 11023 = 1109$

$M_2 = 8015^{5891} \bmod 11023 = 1407$

$M_3 = 4385^{5891} \bmod 11023 = 0013$

$M_4 = 8433^{5891} \bmod 11023 = 8433$

$M5 = 7922^{5891} \bmod 11023 = 0711$

$M_6 = 1436^{5891} \bmod 11023 = 2120$

$M_7 = 576^{5891} \bmod 11023 = 0009$

$M_8 = 9127^{5891} \bmod 11023 = 1419$

$M_9 = 3133^{5891} \bmod 11023 = 2009$

$M_{10} = 9538^{5891} \bmod 11023 = 2021$

$M_{11} = 7420^{5891} \bmod 11023 = 2005$

$M_{12} = 10206^{5891} \bmod 11023 = 0015$

$M_{13} = 4855^{5891} \bmod 11023 = 0600$

$M_{14} = 7420^{5891} \bmod 11023 = 2005$

$M_{15} = 9892^{5891} \bmod 11023 = 0308$

$M_{16} = 10833^{5891} \bmod 11023 = 1415$

$M_{17} = 5601^{5891} \bmod 11023 = 1215$

$M_{18} = 3911^{5891} \bmod 11023 = 0725$

จากตัวอย่างเดิม ถ้าเลือกค่า p และ q ที่มีจำนวนหลักมาก ๆ จะลดจำนวนครั้งของการเข้ารหัสซึ่งทำให้การเข้ารหัสเร็วขึ้น

เลือก p = 23203941528510160092833 (จำนวนเฉพาะขนาด 23 หลัก)

q = 1735739017547947065587 (จำนวนเฉพาะขนาด 22 หลัก)

จะได้ $n = pq$

= 40275986671936234424621110207398375752479552 (44 หลัก)

$(n) = (p-1)(q-1)$

= 40275986671936234424621110207398375752479552

จากคุณสมบัติ $ed = 1 \bmod \phi(n)$ เลือกค่า $e = 7$

จะได้ d = 57537123817051763463744443163426251074970 79

ซึ่ง ed = 40275986671936234424621110207398375752479553

จากข้อความเดิม "KING MONGKUT INSTITUTE OF TECHNOLOGY" เมื่อแปลงเป็นตัวเลขโดยแทน

A = 01 B = 02

C = 03 D = 04

E = 05 F = 06

G = 07 ..... Z = 26

BLANK = 00 จะสามารถแปลงข้อความข้างต้นเป็นตัวเลขได้ดังนี้

11091407001315140711212000091419200 9

20212005001506002005030814151215072 5

จากจำนวนหลักของ n เท่ากับ 44 หลัก เราสามารถแบ่งข้อความที่เป็นตัวเลขได้ 2 บล็อก โดยบล็อกแรกใช้ 40 หลัก ส่วนบล็อกที่ 2 ใช้ 32 หลัก แล้วเข้ารหัสได้ดังนี้

ข้อความบล็อกที่ 1 M_1 : 1109140700131514071121200009141920092021

ทำการเข้ารหัสได้ดังนี้ $C_1 = M_1^e \bmod n$

จะได้ข้อความที่เข้ารหัส C_1: 9380655288089560794796743877454715306769400

ข้อความบล็อกที่ 2 M_2 : 20050015060020050 3

เมื่อเข้ารหัสโดย $C_2 = M_2{}^e \bmod n$

จะได้ข้อความที่เข้ารหัส C_2 : 367470296989927959351939548942953373785791 67

Hash Algorithms

สำหรับการทำ Hash Algorithms นี้ จะขอกล่าวถึงเพียงเล็กน้อยเท่านั้น

Hash Algorithms บางทีเรียกว่า message digests หรือ one-way tranformations จะอาศัยหลักการแปลงข้อมูลโดยอาศัยหลักทางคณิตศาสตร์ โดยเราจะนำข้อมูลที่มี (โดยแปลงเป็น string of bits) มาคำนวณโดยใช้หลักทางคณิตศาสตร์ ซึ่งจะได้ค่าตัวเลขจำนวนหนึ่งที่มีขนาดตามที่กำหนด เมื่อเราต้องการนำผลไปใช้ ก็นำไปเพียงผลจากการคำนวณที่มีขนาดจำกัด ซึ่งมักจะมีขนาดเล็กกว่าข้อมูลจำนวนมาก ๆ ทำให้เราไม่ต้องเสียพื้นที่เก็บมาก ๆ และเมื่อเราจะตรวจสอบความถูกต้อง ก็คำนวณผ่านตัวที่ต้องการตรวจสอบว่าได้ผลเหมือนค่า hash ที่เก็บไว้หรือไม่ ถ้าเท่ากัน แสดงว่าถูกต้อง

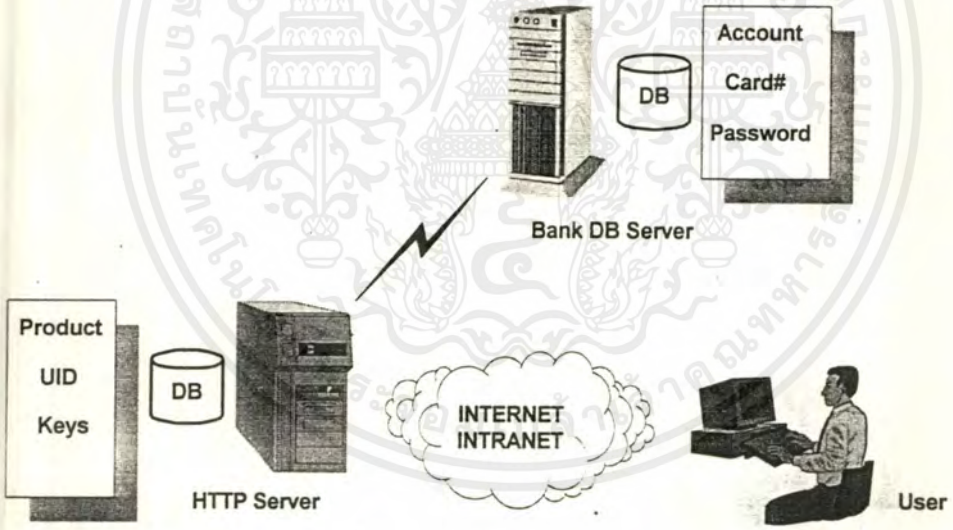
การทำ hash algorithms กับ ข้อมูล (m) เราใช้ $h(m)$ แทน

การทำ Hash Algorithms นิยมทำในการการเก็บข้อมูลเพื่อตรวจสอบข้อมูลในแต่ละวัน หรือ การตรวจสอบการใช้ password เป็นต้น

บทที่ 6
ตัวอย่างระบบงาน

หลักการทำงานของตัวอย่างระบบงาน

เป็นการจำลองระบบขายสินค้าผ่านเครือข่ายอินเทอร์เน็ต โดยที่ผู้ขายซึ่งมีฐานข้อมูลเก็บรายการสินค้า, ราคา, จำนวนในคลังสินค้า และติดต่อเข้ากับฐานข้อมูลลูกค้าของธนาคาร ซึ่งจะมีหมายเลขบัตรเครดิต, รหัสลับสำหรับการใช้งานบัญชี เป็นต้น



รูปที่ 6 - 1 แสดงภาพโดยรวมของระบบงานตัวอย่าง

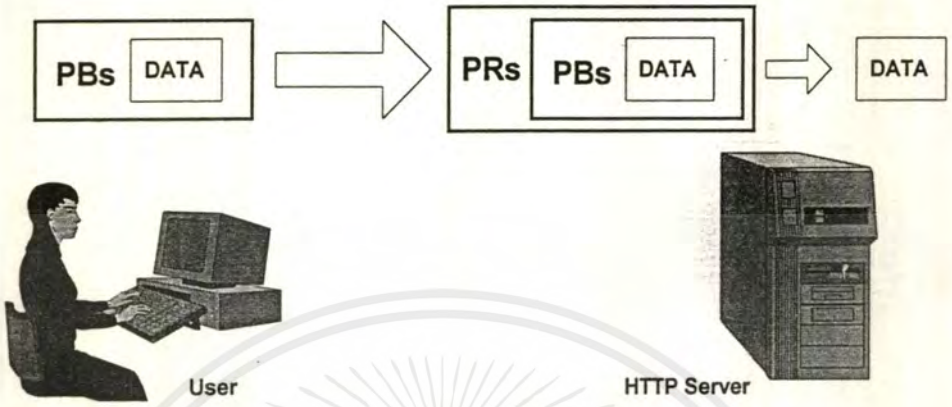
เมื่อผู้ใช้งานปลายทางติดต่อเข้ามาเพื่อจะเลือกซื้อสินค้าที่เว็บเซิร์ฟเวอร์ของผู้ขาย จะมีการส่งจาวาแอพเพล็ทเพื่อแสดงรายการสินค้า, ราคา, และแบบฟอร์มสำหรับให้ผู้ใช้สามารถเลือกซื้อสินค้า รวมทั้งแบบฟอร์มสำหรับใส่หมายเลขบัตรเครดิต และรหัสลับ ในการเลือกซื้อสินค้านั้นผู้ใช้จะสามารถทำได้โดยผ่านจาวาแอพเพล็ทนี้ และการทำงานต่าง ๆ ในช่วงแรกนี้จะมีการประมวลผลที่ฝั่งไคลเอ็นท์ ทั้งนี้เพื่อเป็นการลดภาระของเซิร์ฟเวอร์ ข้อมูลต่าง ๆ ที่ผู้ใช้ใส่ลงในแบบฟอร์มนี้จะต้องมีระบบรักษาความปลอดภัยของข้อมูลโดยคาดว่าจะต้องประกอบหัวข้อต่าง ๆ ดังนี้คือ

1. **Encryption** การเข้ารหัสข้อมูลที่ส่งจากฝั่งไคลเอ็นท์มายังเซิร์ฟเวอร์ ทั้งนี้เพื่อไม่ให้ผู้อื่นที่เชื่อมต่อเครื่องคอมพิวเตอร์เข้ากับเครือข่ายอินเทอร์เน็ตที่อยู่ระหว่างทางสามารถแอบดูข้อมูลได้
2. **Authentication** เพื่อเป็นการยืนยันว่าผู้ใช้ที่อยู่ที่ฝั่งไคลเอ็นท์เป็นตัวจริงตามที่อ้างถึง (เป็นเจ้าของบัตรเครดิตตัวจริง)
3. **Digital Signature** เพื่อใช้สำหรับตรวจสอบข้อมูลที่เซิร์ฟเวอร์ได้รับว่าเป็นข้อมูลที่ถูกต้องตรงกับที่ผู้ใช้ที่ฝั่งไคลเอ็นท์ต้องการ ไม่ได้ถูกแก้ไขระหว่างทาง

จากหัวข้อที่กล่าวมานี้ จึงได้มีการออกแบบระบบให้มีการทำงานดังนี้

Encryption

ในส่วนของการเข้ารหัสนี้ใช้หลักการของ Public-Key Cryptography คือ ที่ฝั่งเว็บเซิร์ฟเวอร์จะมีการสร้างคีย์ที่ใช้ในการเข้ารหัสและถอดรหัสข้อมูลขึ้นมาสองคู่เรียกว่า PBS (Server's Publickey) กับ PRS (Server's Privatekey) และ PBC (Client's Publickey) กับ PRC (Client's Privatekey) เมื่อผู้ใช้ติดต่อเข้ามาที่เว็บเซิร์ฟเวอร์นี้ จะมีการสร้าง PRC และ PBC สำหรับผู้ใช้งานคนนั้น จากนั้นจะส่งคีย์ PRC และ PBS กลับไปให้ฝั่งไคลเอ็นท์ด้วย จากนั้นเมื่อผู้ใช้กรอกข้อมูลลงในแบบฟอร์มแล้ว ก่อนที่จะส่งข้อมูลเหล่านั้นกลับมาให้เซิร์ฟเวอร์ ข้อมูลนั้นก็จะถูกเข้ารหัสโดยจาวาแอพเพล็ทนั่นเอง



รูปที่ 6 - 2 แสดงการเข้ารหัสและถอดรหัสข้อมูลของระบบงาน

การเข้ารหัสของระบบมีหลักสำคัญดังนี้

เป็นการผสมระหว่าง Secret-Key Cryptography และ Public-Key Cryptography โดยการเข้ารหัสจะทำการสุ่มตัวเลขขึ้นมาแล้วทำการเลื่อนตัวอักษรแต่ละตัวไปทางซ้ายตามตาราง ASCII ตามจำนวนที่ได้สุ่มขึ้นมานั้น เช่น มีข้อมูล "Hello" และสุ่มตัวเลขได้ 8 จะได้ข้อมูลที่ถูกเข้ารหัสคือ "Pmttw" เป็นต้น วิธีการในการเข้ารหัสนี้เป็นการเข้ารหัสข้อมูลแบบใช้คีย์เดียว (Secret-Key Cryptography) เนื่องจากตัวอักษรส่วนใหญ่ที่ใช้งานจะมีรหัสแอสกีที่ 33 (!) จนถึง 122 (z) และรหัสแอสกีมีค่าสูงสุด 255 ดังนั้นเลขสุ่มที่ใช้จึงไม่ควรมีค่าเกิน 255-122 = 133 นั่นเอง

หลังจากที่เข้ารหัสข้อมูลโดยใช้เลขสุ่มนี้แล้ว ก็จะนำเลขสุ่มนี้ไปเข้ารหัสโดยใช้ Public-Key Cryptography โดยใช้คีย์ PBS ในการเข้ารหัส ซึ่งมีผลทำให้ผู้อื่นที่ไม่รู้ PRS ไม่สามารถถอดรหัสข้อมูลดังกล่าวได้ ทั้งนี้เป็นไปตามหลักการของ Public-Key Cryptography คือมีคีย์อยู่คู่หนึ่ง ถ้าใช้คีย์ใดในการเข้ารหัสแล้ว จะต้องใช้อีกคีย์หนึ่งเท่านั้นในการถอดรหัส จะใช้คีย์เดิม หรือคีย์อื่น ๆ ที่ไม่ใช่คู่กันไม่ได้

คีย์ที่ใช้ในส่วนของ Public-Key Cryptography ของระบบนี้สามารถสร้างขึ้นโดยใช้ทฤษฎีของ RSA คือเริ่มต้นด้วยการหาค่า p, q จากนั้นคำนวณ n, $\phi(n)$ จากนั้นหาค่า e และ

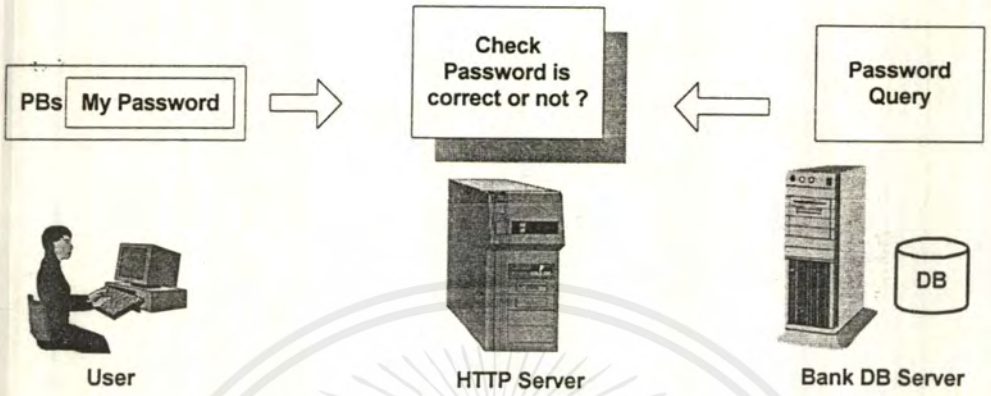
d แต่ว่าขนาดของค่าต่าง ๆ นี้ยังไม่สามารถทำให้มีขนาดใหญ่เหมือนที่ใช้งานจริงได้เนื่องจากข้อจำกัดบางประการซึ่งสามารถอธิบายได้ดังนี้

เนื่องจากโปรแกรมที่ใช้ในการเข้ารหัสและถอดรหัสสามารถใช้ตัวแปรที่มีขนาดสูงสุดได้ 15 หลัก และในการเข้ารหัสและถอดรหัสจะต้องนำผลลัพธ์จากการ mod n มาคูณกันอย่างน้อย 1 ครั้ง ดังนั้นผลลัพธ์จากการ mod n จะต้องไม่เกิน 15/2 = 7.5 = 7 หลัก ดังนั้น ค่า n จะมีค่าสูงสุดได้ไม่เกิน 7 หลัก โดยค่า n นั้นได้จากค่า $p*q$ ดังนั้นขนาดของ p และ q จึงมีขนาดรวมกันไม่เกิน 7 หลัก ดังนั้นจึงเลือกให้ p มีขนาด 4 หลัก และ q มีขนาด 3 หลัก ซึ่งเมื่อนำไปคำนวณค่า $\phi(n)$ ก็จะได้ขนาด 7 หลัก ทำให้ค่า e, d มีขนาดไม่เกิน 7 หลักด้วยนั้นเอง

อย่างไรก็ตามในการใช้งานจริง ๆ นั้นการคำนวณต่าง ๆ อาจอยู่ในรูปของเลขฐานสองทำให้สามารถคำนวณตัวเลขขนาดใหญ่ ๆ ได้อย่างไม่จำกัดขนาด เช่น การใช้ค่า p , q ขนาด 64, 128 หรือ 512 bit เป็นต้น ซึ่งในส่วนนี้สำหรับโครงงานอาจมีการพัฒนาขึ้นต่อไปในอนาคต

Authentication

คือการยืนยันว่าผู้ใช้ที่อยู่ที่ฝั่งไคลเอ็นท์เป็นผู้ใช้ตัวจริงตามที่อ้าง เราสามารถตรวจสอบได้โดยใช้รหัสผ่าน (Password) โดยในระบบจำลองนี้ นอกจากผู้ใช้จะต้องใส่หมายเลขบัตรเครดิตให้ถูกต้องตรงกับข้อมูลที่มีอยู่ที่ฐานข้อมูลของธนาคารแล้ว ยังต้องใส่รหัสผ่านให้ถูกต้องตรงกับข้อมูลที่อยู่ในฐานข้อมูลของธนาคารด้วย ดังนั้นในกรณีที่ผู้ใช้ไม่ใช่เจ้าของบัตรตัวจริง ก็จะไม่สามารถใช้งานได้เนื่องจากไม่รู้รหัสผ่าน



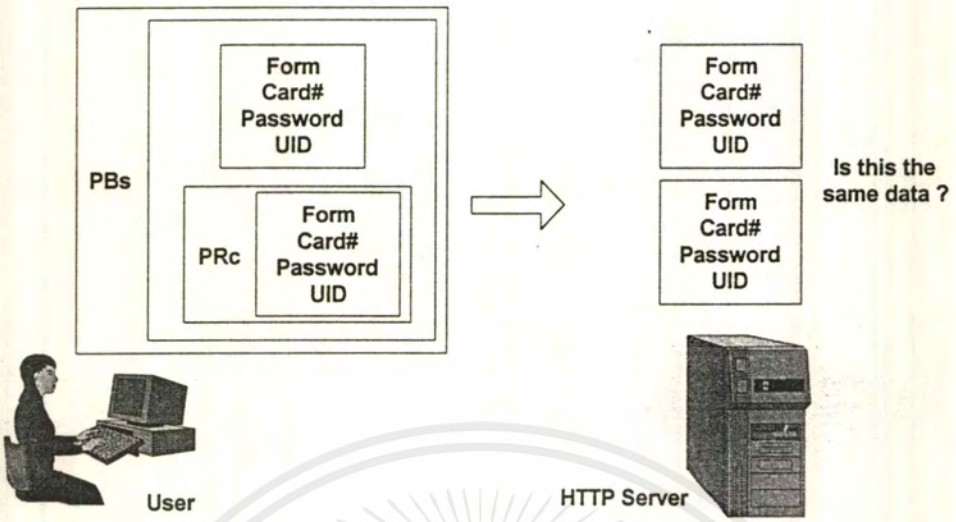
รูปที่ 6 - 3 แสดงขั้นตอนการตรวจสอบรหัสผ่าน

Digital Signature

ใช้สำหรับตรวจสอบข้อมูลที่ส่งมายังเว็บเซิร์ฟเวอร์ว่าถูกแก้ไขโดยผู้อื่นระหว่างทางหรือไม่ ซึ่งในระบบจำลองนี้มีวิธีการดังนี้

เมื่อผู้ใช้งานเรียกใช้งาน ที่ฝั่งเซิร์ฟเวอร์จะมีการสร้างคีย์คู่หนึ่งสำหรับผู้ใช้งานคนนั้น เรียกว่า PBC , PRC พร้อมกับหมายเลขของผู้ใช้คนนั้น (UID) จากนั้นจะเก็บข้อมูลเหล่านี้ไว้ที่ฐานข้อมูล พร้อมทั้งส่ง PRC และ UID กลับไปให้ผู้ใช้ด้วย เมื่อผู้ใช้งานกรอกข้อมูลลงในแบบฟอร์ม ข้อมูลในแบบฟอร์มจะถูกทำสำเนาขึ้นมาเป็น 2 ชุด ชุดหนึ่งจะถูกเข้ารหัสโดยใช้ PRC ก่อน จากนั้น ข้อมูลทั้งหมดจะถูกเข้ารหัสอีกครั้งโดย PBS

เมื่อเซิร์ฟเวอร์ได้รับข้อมูล ก็จะทำการถอดรหัสโดยใช้ PRS ก่อน จากนั้นก็จะตรวจสอบ UID และเลือกใช้ PBC ในฐานข้อมูลที่ตรงกันในการถอดรหัสข้อมูลส่วนหนึ่ง จากนั้นก็จะตรวจสอบข้อมูลทั้งสองชุดนี้ว่าถูกต้องตรงกันหรือไม่



รูปที่ 6 - 4 แสดงการทำ Digital Signature

ภาคผนวก ก
หลักการของระบบไคลเอ็นท์/เซิร์ฟเวอร์

แนวความคิดของไคลเอ็นท์/เซิร์ฟเวอร์

แนวความคิดของไคลเอ็นท์/เซิร์ฟเวอร์ คือรูปแบบการทำงานที่สนับสนุนสภาวะแวดล้อมที่แอพพลิเคชั่น ที่เรียกว่าไคลเอ็นท์ ร้องขอบริการจากแอพพลิเคชั่น ที่เรียกว่าเซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์นั้นมีหลายแบบได้แก่

1. ไฟล์เซิร์ฟเวอร์ (File Server) จะเตรียมบริการในการเข้าถึงไฟล์ และจัดการไฟล์
2. พริ้นต์เซิร์ฟเวอร์ (Print Server) จะเตรียมบริการงานพิมพ์บนเครือข่าย
3. ดาต้าเบสเซิร์ฟเวอร์ (Database Server) จะเตรียมบริการในการเข้าถึงฐานข้อมูลและจัดการฐานข้อมูล
4. คอมมิวนิเคชั่นเซิร์ฟเวอร์ (Communication Server) จะเตรียมบริการทางด้านการสื่อสารข้อมูลบนเครือข่าย
5. แอพพลิเคชั่นเซิร์ฟเวอร์ (Application Server) จะเตรียมบริการการเข้าถึงแอพพลิเคชั่นและยอมให้แอพพลิเคชั่นกระจายไปที่โฮสต์มากกว่า 1 โฮสต์

ส่วนของไคลเอ็นท์

ส่วนนี้ทำหน้าที่รันแอพพลิเคชั่นของไคลเอ็นท์ ส่วนนี้ทำงานด้วยระบบ GUI (Graphic User Interface) หรือ OOUI (Object-Oriented User Interface) และ DSM (Distributed System Management) ในรูปแบบใดรูปแบบหนึ่ง

ส่วนของมิดเดิลแวร์ (Middleware)

เป็นส่วนที่อยู่ตรงกับเครื่องหมายทับ (/) ทำงานทั้งส่วนของไคลเอ็นท์ และเซิร์ฟเวอร์ของแอพพลิเคชั่น เราแบ่งมิดเดิลแวร์ออกเป็น 4 ประเภทด้วยกันคือ

1. Transport-Stack
2. Network Operating System (NOS)
3. Distributed System Management (DSM)
4. Service-Specific

Transport-Stack กับ NOS ให้พื้นฐานในการสื่อสารทั่วไป ส่วน DSM จะทำงานบนทุก ๆ โหนดของเครือข่ายแบบไคลเอ็นท์/เซิร์ฟเวอร์ มันต้องการมิดเดิลแวร์ของตัวมันเองบน NOS ที่จะทำการแลกเปลี่ยนข้อมูลระหว่างสเตชั่นต่าง ๆ ที่จัดการ ส่วน Service-Specific จะขึ้นอยู่กับรูปแบบของแอพพลิเคชั่น

ดาต้าเบสแอพพลิเคชั่น (Database Application) ใช้มิดเดิลแวร์ในรูปของ SQL พร้อมกับ ODBC (Open Database Connectivity) , DRPA (Distributed Relational Database Architecture) , RDA (Remote Database Access)

ส่วนของเซิร์ฟเวอร์

ทำหน้าที่รันแอพพลิเคชั่นที่จัดการกับทรัพยากรต่าง ๆ ที่ใช้ร่วมกัน มีรูปแบบของแอพพลิเคชั่นอยู่ 4 รูปแบบ คือ

1. ฐานข้อมูล SQL
2. ระบบติดตามการประมวลผล Transaction (TP Monitor)
3. กรุ๊ปแวร์ (Groupware)
4. ออบเจ็คท์แบบกระจาย (Distributed Object)

และในส่วนของเซิร์ฟเวอร์ยังบรรจุส่วนของ DSM เอาไว้ด้วย

มีข้อสังเกตคือ ส่วนประกอบทั้ง 3 ส่วนนี้อาจจะทำงานอยู่บนเครื่องคอมพิวเตอร์เครื่องเดียวกันเลยก็ได้ เพราะโหนดใด ๆ สามารถเป็นได้ทั้งไคลเอ็นท์ และเซิร์ฟเวอร์โดยการโต้ตอบระหว่างเซิร์ฟเวอร์กับเซิร์ฟเวอร์ส่วนใหญ่ จะอยู่ในรูปของไคลเอ็นท์/เซิร์ฟเวอร์ โดยสลับกันเป็นไคลเอ็นท์และเซิร์ฟเวอร์ อย่างไรก็ดีการโต้ตอบระหว่างเซิร์ฟเวอร์กับเซิร์ฟเวอร์ ต้องมีมิดเดิล์แวร์พิเศษโดยเฉพาะ

ชนิดของการประมวลผลไคลเอ็นท์/เซิร์ฟเวอร์

1. Stand-Alone ไคลเอ็นท์/เซิร์ฟเวอร์
2. Stand-Alone LAN ไคลเอ็นท์/เซิร์ฟเวอร์
3. Manual Extract ไคลเอ็นท์/เซิร์ฟเวอร์
4. Single-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์
5. Multi-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์
6. Distributed Database ไคลเอ็นท์/เซิร์ฟเวอร์

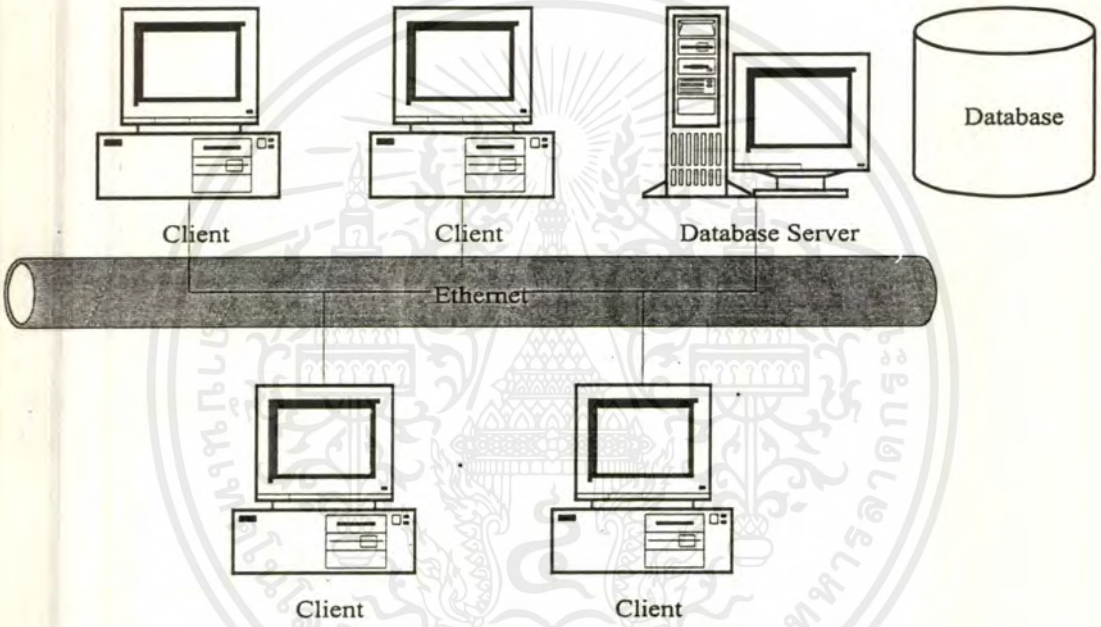
Stand-Alone ไคลเอ็นท์/เซิร์ฟเวอร์

แอพพลิเคชั่นประเภทนี้จะมีผู้ขอใช้บริการประมวลผลอยู่บนเครื่องเดียวกันที่ให้บริการ ลักษณะการทำงานประเภทนี้ จะบั่นทอนประสิทธิภาพการประมวลผลระบบจัดการฐานข้อมูลลงบ้าง แต่ความเร็วในการสื่อสารระหว่างผู้ขอใช้บริการและผู้ให้บริการจะสูงมาก และสามารถเพิ่มประสิทธิภาพการประมวลผลได้โดยการใช้มัลติโปรเซสเซอร์ (Multiprocessor)

รูปที่ ก - 1 Stand-Alone ไคลเอ็นท์/เซิร์ฟเวอร์

Stand-Alone LAN ไคลเอ็นท์/เซิร์ฟเวอร์

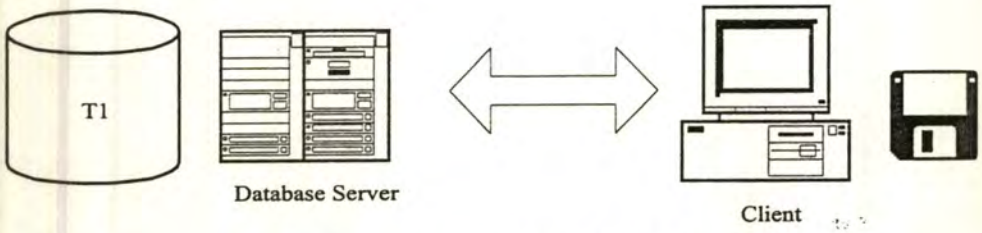
ระบบไคลเอ็นท์/เซิร์ฟเวอร์แบบนี้ จะเป็นรูปแบบของไคลเอ็นท์/เซิร์ฟเวอร์ในวง LAN วงหนึ่ง การทำงานของผู้ขอบริการแต่ละตัวอาจจะรับผิดชอบงานด้านการนำเสนอข้อมูลประมวลผลธุรกิจ และลอจิกทางด้านฐานข้อมูล ในขณะที่ผู้ให้บริการ จะรับผิดชอบในเรื่องของการเรียกใช้ข้อมูลสำหรับผู้ขอบริการภายในวง LAN แต่มีข้อเสียคือ การสื่อสารระหว่างผู้ขอบริการและผู้ให้บริการที่ทำผ่านการเชื่อมต่อของวง LAN จะช้ากว่าแอพพลิเคชั่นไคลเอ็นท์/เซิร์ฟเวอร์ที่ทำงานบนเครื่องเดียวกัน



รูปที่ ก - 2 Stand-Alone LAN ไคลเอ็นท์/เซิร์ฟเวอร์

Manual Extract ไคลเอ็นท์/เซิร์ฟเวอร์

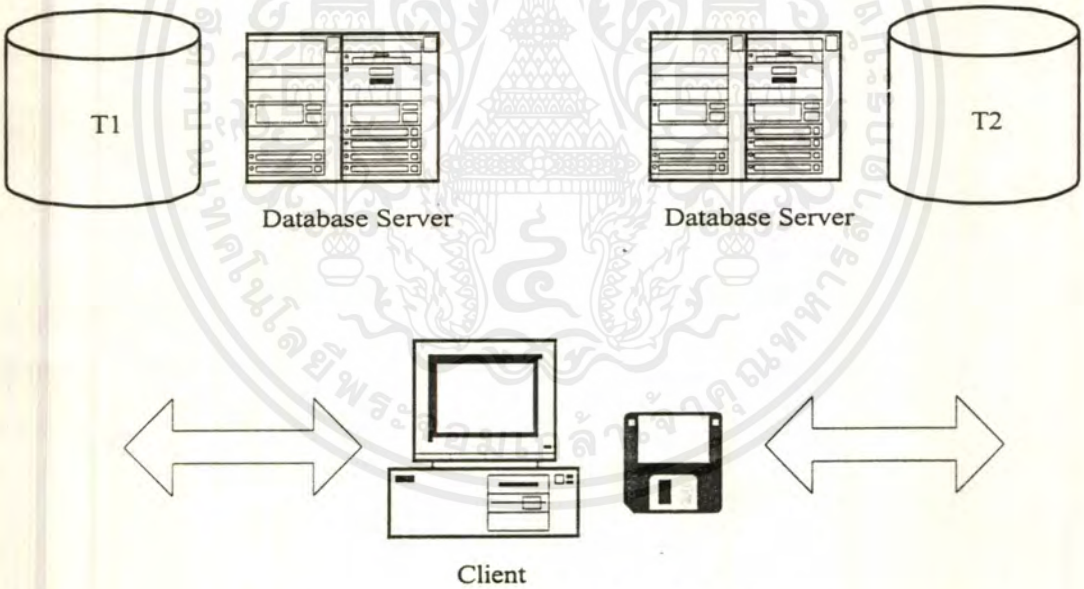
แอพพลิเคชั่นของระบบไคลเอ็นท์/เซิร์ฟเวอร์แบบนี้ จะเรียกใช้ข้อมูลบางส่วนของทั้งหมด โดยย้ายไปเก็บที่เครื่องของผู้ขอใช้บริการ ซึ่งข้อมูลส่วนนี้ถูกสร้างขึ้นด้วยวิธีการกระจายข้อมูลแบบ Manual Extract ลักษณะของการทำงานของแอพพลิเคชั่น สามารถเกิดขึ้นโดยผู้ใช้ส่งคำสั่งไปยังผู้ให้บริการเพื่อเรียกใช้ข้อมูลซึ่งมักจะกำหนดให้อ่านอย่างเดียว โดยการคัดข้อมูลและการย้ายข้อมูลเป็นสิ่งที่สำคัญและจำเป็นต้องทำ



รูปที่ ก - 3 Manual Extract ไคลเอ็นท์/เซิร์ฟเวอร์

Single-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์

แอพพลิเคชั่นของระบบไคลเอ็นท์/เซิร์ฟเวอร์แบบนี้ จะมีความสามารถสูงขึ้น โดยมันจะทำการส่งคำสั่ง ที่ประกอบด้วยคำสั่งหลายคำสั่งส่งไปยังผู้ให้บริการหลาย ๆ ตัวที่อยู่ห่างไกลกัน แต่ข้อมูลที่ทำการเรียกใช้จากผู้ให้บริการแต่ละตัวมักจะไม่มีความสัมพันธ์กัน เพราะไม่ได้เชื่อมต่อกันเป็นเครือข่ายเดียวกัน และไม่มีผู้ให้บริการใดทำหน้าที่เป็นตัวกลางในการสื่อสาร

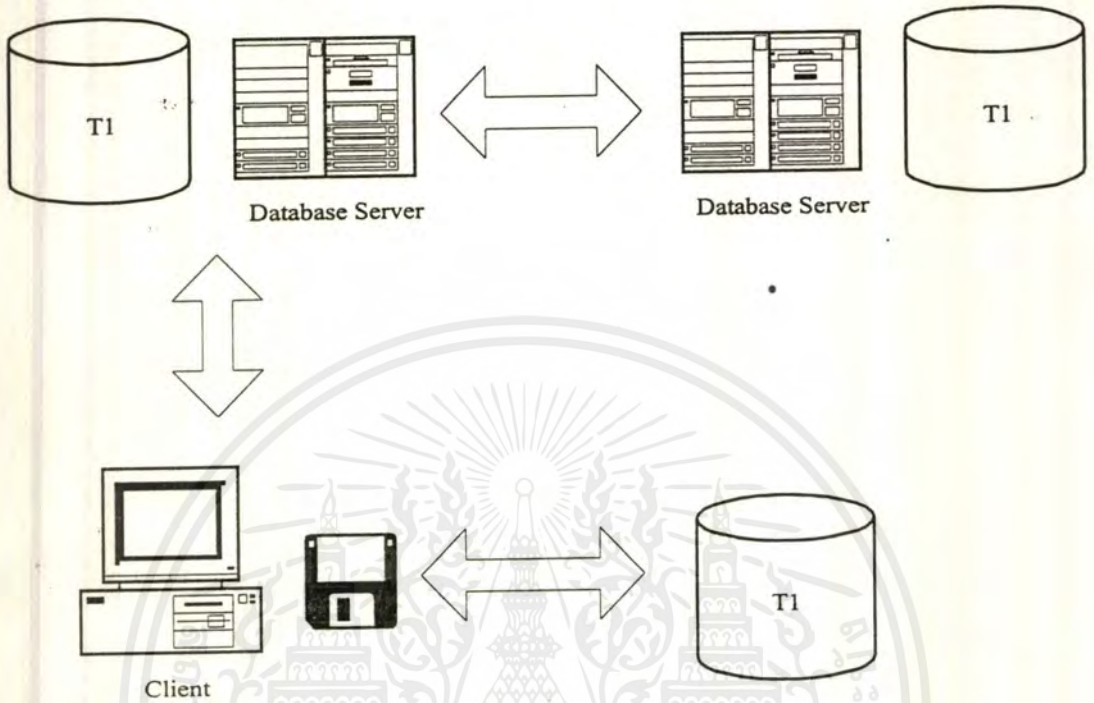


รูปที่ ก - 4 Single-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์

Multi-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์

แอพพลิเคชั่นของระบบไคลเอ็นท์/เซิร์ฟเวอร์แบบนี้ จะสนับสนุนการติดต่อกันระหว่างผู้ให้บริการแต่ละตัว ดังนั้นผู้ขอใช้บริการสามารถออกคำสั่งที่จะแก้ไขข้อมูลที่เก็บอยู่หลาย ๆ ที่ได้ โดยอาจจะมองได้ว่า เป็นลักษณะไคลเอ็นท์/เซิร์ฟเวอร์ที่มีความสามารถในการกระจาย

ฐานข้อมูล และเมื่อมีความสามารถประเภทนี้แล้ว การกระจายข้อมูลจะทำด้วยวิธี Snapshot จากผู้ขอใช้บริการฐานข้อมูล แทนที่จะเป็นวิธี Manual Extract

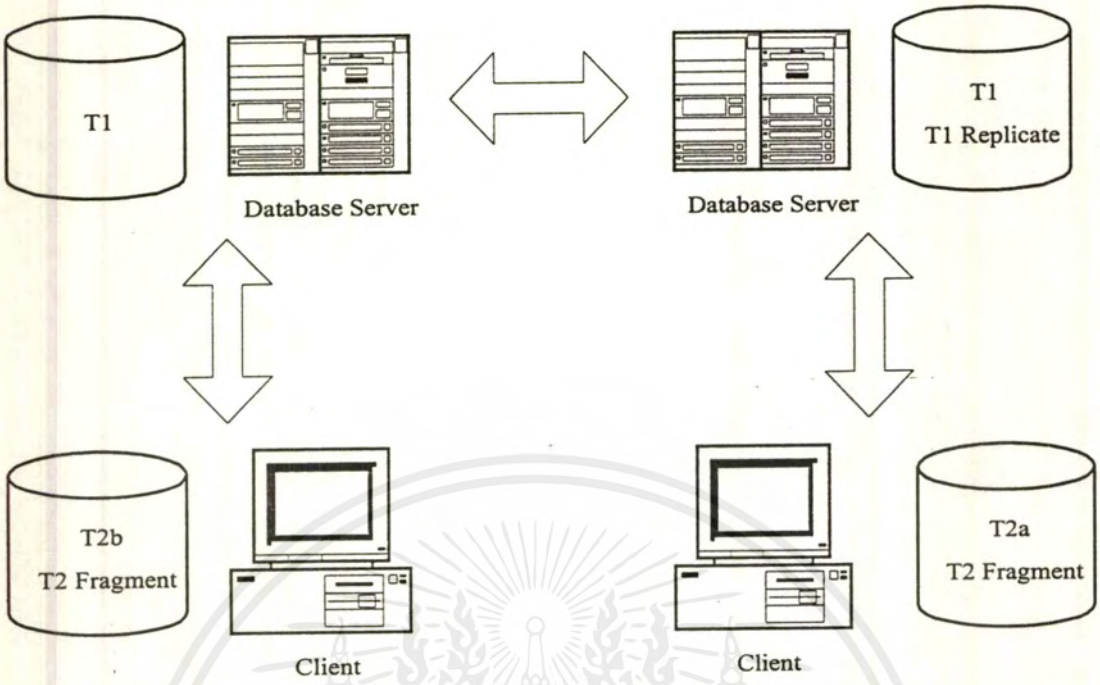


รูปที่ ก - 5 Multi-Site Update ไคลเอ็นท์/เซิร์ฟเวอร์

Distributed Database ไคลเอ็นท์/เซิร์ฟเวอร์

ระบบไคลเอ็นท์/เซิร์ฟเวอร์แบบนี้ จะใช้แอพพลิเคชั่นฐานข้อมูลแบบกระจาย และใช้การประมวลผลแบบ Distributed Request ลักษณะของไคลเอ็นท์/เซิร์ฟเวอร์ประเภทนี้ ผู้ให้บริการฐานข้อมูล จะสนับสนุนทั้งการคัดแบ่งข้อมูล หรือการสำเนา (Copy) ข้อมูลทั้งหมดไปเก็บไว้ตามหน่วยเก็บข้อมูลของผู้ให้บริการต่าง ๆ ซึ่งทำให้การอ่านข้อมูลสามารถทำได้รวดเร็ว แต่การแก้ไขข้อมูลอาจจะต้องใช้เวลามากกว่า เพราะว่าจะต้องมีการติดต่อระหว่างผู้ให้บริการซึ่งอาจจะไม่ใช่แค่ 2 ตัว ดังนั้นเทคโนโลยีทางด้านการสื่อสารจึงมีบทบาทสำคัญในการขจัดปัญหาด้านความเร็ว

ความสามารถที่จำเป็นสำหรับแอพพลิเคชั่นประเภทนี้คือ การที่แอพพลิเคชั่นไม่จำเป็นต้องรู้ตำแหน่งของผู้ให้บริการ หรือตำแหน่งที่เกิดการประมวลผลฐานข้อมูล การควบคุมประสิทธิภาพโดยรวมของระบบ การควบคุมความถูกต้องของข้อมูลที่กระจายอยู่ตามที่ต่าง ๆ และการควบคุมการทำการกระจายข้อมูลซึ่งเป็นส่วนสำคัญของระบบไคลเอ็นท์/เซิร์ฟเวอร์ประเภทนี้



รูปที่ ก - 6 Distributed Database ไคลเอ็นท์/เซิร์ฟเวอร์

ภาคผนวก ข

ODBC

Open Database Connectivity

ODBC คืออะไร

Open Database Connectivity คือวิธีการติดต่อและเข้าถึงจากแอพพลิเคชั่นสู่ระบบจัดการฐานข้อมูลโดยใช้ภาษา SQL เป็นมาตรฐานการเข้าถึงฐานข้อมูล ความสามารถในการต่อเชื่อมแบบนี้ทำให้แอพพลิเคชั่นสามารถเข้าถึงฐานข้อมูลได้หลายรูปแบบ ซึ่งทำให้ผู้พัฒนาโปรแกรมสามารถพัฒนาโปรแกรมไปได้โดยไม่ต้องทำการระบุชนิดของระบบจัดการฐานข้อมูล

แต่เดิมนั้นการพัฒนาโปรแกรมประยุกต์ที่ใช้งานเกี่ยวกับฐานข้อมูล การเข้าใช้ฐานข้อมูลของโปรแกรมเหล่านี้จะทำผ่านการเรียกใช้ SQL แบบฝังตัว (Embedded SQL) ซึ่งในขณะนั้นวิธีทางแบบนี้ก็ดูจะไปได้ดีทีเดียว เพราะว่าตัวโปรแกรมสามารถทำการเปลี่ยนรูปแบบของระบบไม่ว่าจะเป็นทางด้านฮาร์ดแวร์ หรือซอฟท์แวร์ได้หลายรูปแบบ รวมทั้งระบบปฏิบัติการด้วย (โดยการคอมไพล์ใหม่ทุกครั้งที่มีการย้ายระบบ)

อย่างไรก็ตามในการพัฒนาโปรแกรมในระบบที่ความแตกต่างกัน เช่นการเรียกใช้ข้อมูลของออราเคิลจากไมโครซอฟท์เอ็กซ์เซล (Microsoft Excel) วิธีการเข้าถึงข้อมูลแบบเดิมนั้นจะต้องทำการพรีคอมไพล์โค้ดของเอ็กซ์เซลและออราเคิล โดยใช้ IBM precompiler และ Oracle precompiler ตามลำดับ ซึ่งจะเห็นว่าเป็นการยุ่งยากมากทีเดียว

วิธีการต่อเชื่อมแบบ ODBC จะให้ความสะดวกในการติดต่อข้อมูลมากกว่าวิธีการดั้งเดิมโดยการกำหนดมาตรฐานการต่อเชื่อมของข้อมูล (Data protocol, DBMS capability) และแนวทางนี้ได้ทำให้เกิดความคิดที่จะสร้างไดรฟ์เวอร์สำหรับการติดต่อกับของฐานข้อมูลขึ้นมา (DLL)

ทฤษฎีการทำงานของ ODBC

ส่วนติดต่อของการเชื่อมต่อฐานข้อมูลเปิด (ODBC : Open Database Connectivity) เป็นตัวทำให้โปรแกรมประยุกต์สามารถที่จะเข้าถึงข้อมูลในระบบจัดการฐานข้อมูล (DBMS : Database Management System) โดยใช้ภาษา SQL (Structure Query Language) เป็นมาตรฐานหลักในการเข้าถึงข้อมูล

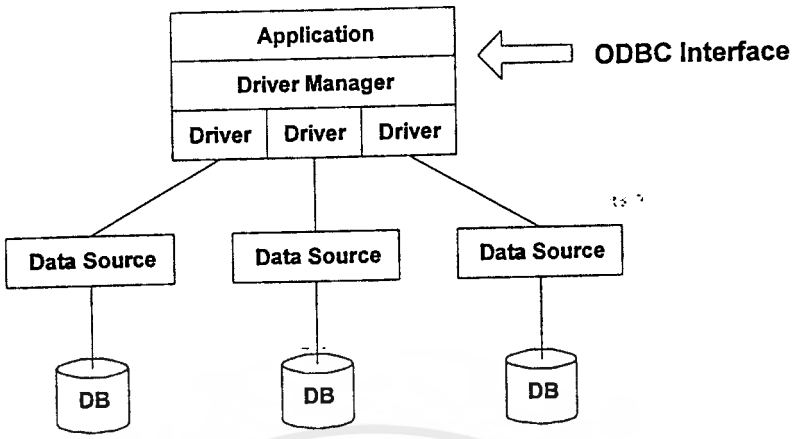
ข้อดีของการติดต่อโดยใช้ ODBC

1. ฟังก์ชั่นของ ODBC อนุญาตให้แอพพลิเคชั่นติดต่อกับระบบจัดการฐานข้อมูลได้โดยสะดวก (การทำคำสั่ง SQL และการรับผลลัพธ์)
2. ใช้ภาษา SQL ตามมาตรฐาน SQL CAE, X/Open และ SQL Access Group (SAG)
3. มีการกำหนดการส่งกลับรหัสความผิดพลาด (Error Code) เป็นมาตรฐานเดียวกัน
4. เป็นวิธีการมาตรฐานในการติดต่อกับระบบจัดการฐานข้อมูล
5. มีการกำหนดชนิดข้อมูล (Data Type) เป็นมาตรฐาน
6. ชุดคำสั่ง SQL สามารถกำหนดได้แม้ในขณะ Runtime
7. สามารถเขียนโปรแกรมชุดเดียว แต่สามารถเข้าใช้ระบบจัดการฐานข้อมูลได้หลายตัว
8. ตัวโปรแกรมไม่ต้องรับผิดชอบในการดูแลการติดต่อข้อมูลกับระบบจัดการฐานข้อมูล
9. ค่าข้อมูลสามารถถูกส่งหรือรับได้ในรูปแบบที่สะดวกขึ้น

องค์ประกอบของ ODBC

สถาปัตยกรรมของ ODBC ประกอบด้วย 4 ส่วนสำคัญ

1. แอพพลิเคชั่น ทำหน้าที่ประมวลผลและเรียกใช้ฟังก์ชั่นของ ODBC ตามคำสั่งภาษา SQL พร้อมทั้งทำการรับผลลัพธ์ด้วย
2. ตัวจัดการไดรฟ์เวอร์ (Driver Manager) ทำหน้าที่โหลดไดรฟ์เวอร์เชื่อมต่อกับแหล่งข้อมูล
3. ไดรฟ์เวอร์ (Driver) ทำหน้าที่ประมวลผลการเรียกใช้ฟังก์ชั่นของ ODBC , ส่งคำสั่ง SQL ไปสู่แหล่งข้อมูลที่ถูกต้อง และทำการส่งผลลัพธ์กลับให้แอพพลิเคชั่น และในบางครั้งไดรฟ์เวอร์ จะทำหน้าที่แปลงคำสั่งที่ส่งมาให้อยู่ในรูปที่สนับสนุนโดยระบบจัดการฐานข้อมูลแต่ละชนิดอีกด้วย
4. แหล่งข้อมูล (Data Source) เป็นแหล่งข้อมูลที่ผู้ใช้ต้องการเข้าถึง



รูปที่ ข - 1 องค์ประกอบของ ODBC

แอพพลิเคชั่น

ตัวโปรแกรมจะเรียกใช้การต่อเชื่อม ODBC ในการทำงานต่อไปนี้

1. ร้องขอการต่อเชื่อมกับแหล่งข้อมูล
2. สุ่งคำสั่ง SQL สู่แหล่งข้อมูล
3. กำหนดพื้นที่การจัดเก็บและรูปแบบของข้อมูลที่เป็นผลลัพธ์จาก SQL Request
4. ร้องขอผลลัพธ์
5. ประมวลผลและจัดการกับข้อผิดพลาด
6. รายงานผลให้กับผู้ใช้ (ถ้าจำเป็น)
7. ร้องขอการ Commit หรือ Rollback สำหรับควบคุมการประมวลผล Transaction
8. ยกเลิกการติดต่อกับแหล่งข้อมูล

ตัวจัดการไดรฟ์เวอร์

ตัวจัดการไดรฟ์เวอร์ คือ DLL (Dynamic Link Library) และไลบรารี่อื่น ๆ หน้าที่หลักของตัวจัดการไดรฟ์เวอร์ก็คือการโหลดไดรฟ์เวอร์ ส่วนหน้าที่อื่น ๆ ก็มีดังนี้

1. เรียกใช้ไฟล์ ODBC.INI เพื่อกำหนดชื่อของแหล่งข้อมูล (Data Source Name) ให้กับไดรฟ์เวอร์ DLL
2. ทำการประมวลผลการเริ่มต้นการเชื่อมต่อของ ODBC
3. เป็นจุดต่อเชื่อมระหว่างฟังก์ชั่นของ ODBC กับไดรฟ์เวอร์แต่ละตัว
4. ทำการตรวจสอบพารามิเตอร์และลำดับการเรียกใช้ ODBC

ไดรฟ์เวอร์

ไดรฟ์เวอร์คือ DLL ที่สร้างฟังก์ชั่นของ ODBC และทำการโต้ตอบกับแหล่งข้อมูล ไดรฟ์เวอร์ทำการตอบสนองการเรียกใช้ฟังก์ชั่นของ ODBC โดยจะทำงานต่อไปนี้

1. สร้างการต่อเชื่อมกับแหล่งข้อมูล
2. ส่งคำขอร้องให้กับแหล่งข้อมูล
3. แปลงข้อมูลจากรูปแบบหนึ่งสู่อีกรูปแบบหนึ่ง
4. ส่งผลลัพธ์กลับให้แอพพลิเคชั่น
5. จัดการส่งข้อมูลความผิดพลาดให้อยู่ในรูปแบบรหัสมาตรฐานแล้วส่งกลับไปให้แอพพลิเคชั่น
6. ทำหน้าที่จัดการและดูแลเคอร์เซอร์ (Cursor)

แหล่งข้อมูล (Data Source)

แหล่งข้อมูล หมายถึงการรวมกันของระบบจัดการฐานข้อมูล, ระบบปฏิบัติการ และระบบเครือข่าย โดยมีการชี้เฉพาะชนิด และประเภทลงไป หรืออีกนัยหนึ่งก็หมายความว่า การที่แอพพลิเคชั่นทำการติดต่อกับระบบจัดการฐานข้อมูล ยี่ห้อหนึ่งบนระบบปฏิบัติการหนึ่ง และเข้าถึงโดยระบบเครือข่ายชนิดหนึ่ง เช่น ออราเคิลที่วิ่งบนระบบปฏิบัติการ OS/2 โดยใช้ระบบเครือข่ายของ Novell Netware เป็นต้น

ภาคผนวก ค

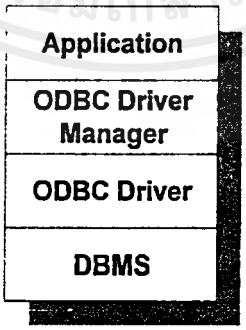
JDBC

Java Database Connectivity

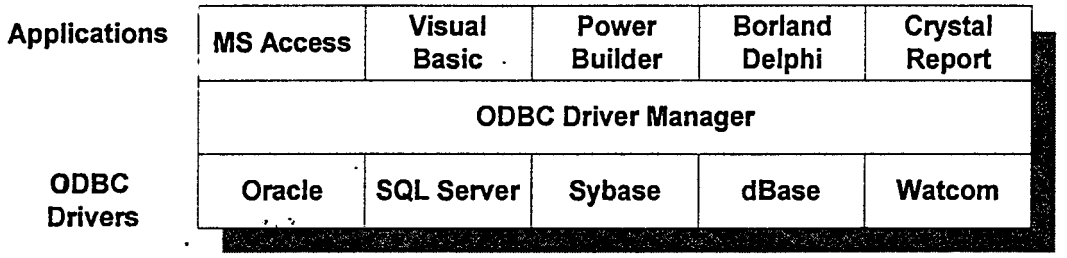
เจดีบีซี เป็นมาตรฐานที่ช่วยในการติดต่อระหว่างจาวา กับระบบจัดการฐานข้อมูล (DBMS : Database Management System) โดยเจดีบีซีนี้เป็นคำย่อมาจาก จาวาดาต้าเบสคุอนเนคติวิตี้ (JDBC : Java Database Connectivity) โดยตัวที่ใช้ทำงานจริง ๆ ระหว่างจาวากับระบบจัดการฐานข้อมูล ก็คือ เจดีบีซีเอพีไอ (JDBC API : Java Database Connectivity Application Programming Interface)

การทำงานของเจดีบีซี

ในการทำงาน เจดีบีซีจะสร้างอินเตอร์เฟซเพื่อติดต่อกับฐานข้อมูลในลักษณะเดียวกับการทำงานของโอดีบีซี (ODBC - Open Database Connectivity) ดังรูปที่ ค - 1 และ ค - 2 ที่แสดงโมเดลของโอดีบีซี



รูปที่ ค - 1 แสดงโมเดลการทำงานของโอดีบีซี

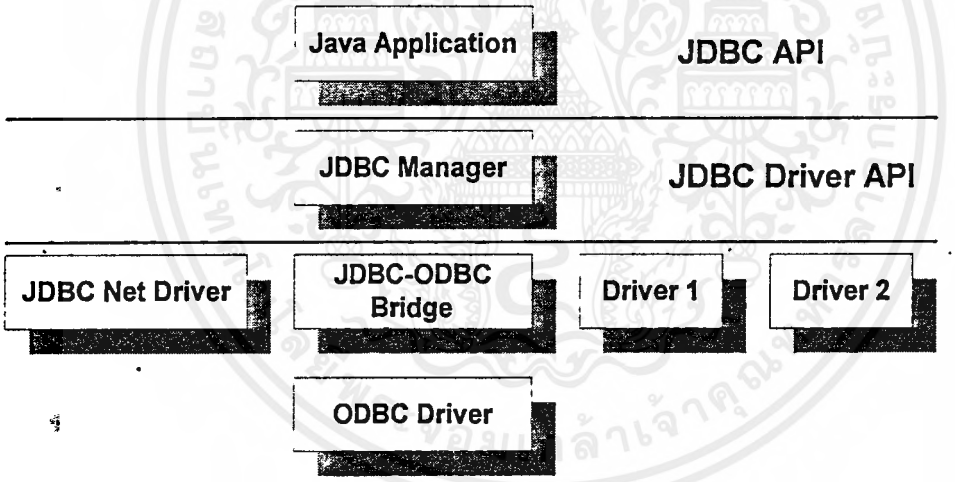


รู

ปที่ ค - 2 สถาปัตยกรรมของโอดีบีซีและตัวอย่างของฐานข้อมูลที่ใช้ไดรฟ์เวอร์โอดีบีซี

สำหรับการทำงานของเจดีบีซีแล้ว จะมีการทำงานอยู่ที่ 2 เลย์เยอร์หลัก คือ

1. **เจดีบีซีเอพีไอ** ซึ่งจะทำการติดต่อระหว่างแอพพลิเคชั่น หรือ แอพเพล็ทใด ๆ ที่เขียนขึ้นด้วยภาษาจาวา กับ เจดีบีซีแมเนเจอร์ (JDBC Manager)
2. **เจ่ดีบีซีไดรฟ์เวอร์เอพีไอ** จะทำการติดต่อระหว่างเจดีบีซีแมเนเจอร์ กับไดรฟ์เวอร์ของระบบฐานข้อมูลที่ติดต่อด้วย ดังรูปที่ ค - 3



รูปที่ ค - 3 แสดงโมเดลชั้นการทำงานของเจดีบีซี

จากรูปที่ ค - 3 เราจะเห็นว่า ใน 2 เลย์เยอร์หลัก จะมีองค์ประกอบที่จำเป็นของการทำงานของเจดีบีซี คือชั้นเจดีบีซีเอพีไอจะมีแอพพลิเคชั่นหรือแอพเพล็ท และชั้นเจดีบีซีไดรฟ์เวอร์เอพีไอก็จะมีเจดีบีซีแมเนเจอร์ซึ่งจะทำหน้าที่ในการเป็นเจดีบีซีไดรฟ์เวอร์เอพีไอเพื่อให้แอพพลิเคชั่นหรือแอพเพล็ทในชั้นบนสามารถติดต่อกับไดรฟ์เวอร์ของระบบฐานข้อมูลได้

ในชั้นล่างสุดเราจะเห็นไดรฟ์เวอร์ต่าง ๆ ที่เจดีบีซีจะอาศัยเจดีบีซีแมเนเจอร์ในการติดต่อเข้ากับแอพพลิเคชั่น หรือ แอพเพล็ท และยังมี เจดีบีซี-โอดีบีซี บริดจ์ (JDBC-ODBC Bridge) ซึ่งเป็นตัวเชื่อมระหว่างแอพพลิเคชั่นหรือ แอพเพล็ทใด ๆ ที่ต้องการติดต่อกับฐาน

ข้อมูลที่สนับสนุนมาตรฐานของไดรฟ์เวอร์โอดีบีซี โดยการทำงานก็เช่นเดียวกับการติดต่อกับไดรฟ์เวอร์ทั่ว ๆ ไป

เจดีบีซีทำอะไรบ้าง

เจดีบีซีมีหน้า 3 อย่างด้วยกัน คือ

1. สร้างจุดเชื่อมต่อกับฐานข้อมูลที่ต้องการติดต่อด้วย
2. ส่งคำสั่งเอสคิวแอล (SQL Statement) ไปยังฐานข้อมูลนั้น
3. รับผลจากคำสั่งที่สั่งไปในขั้นที่ 2

ในที่นี้จะขอยกตัวอย่างโค้ดสั้น ๆ ของการใช้งานทั้ง 3 อย่างที่กล่าวมาแล้ว ดังนี้

```
Connection con = DriverManager.getConnection {
  "jdbc:odbc:wombat", "login", "password");          .....(1)
Statement stmt = con.createStatement();                .....(2)
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table");  ....(3)
while (rs.next()) {
      int x = getInt("a");
      String s = getString("b");
      float f = getFloat("c");
  }
```

จากโค้ดข้างต้น จะเป็นการเรียกใช้เมทอด (method) จากเอพีไอของเจดีบีซี เพื่อทำงาน 3 ขั้นตอน กล่าว คือ

1. ขั้นตอนการสร้างการติดต่อกับฐานข้อมูล จะเป็น การใช้เมทอด (1) คือ getConnection
2. ขั้นตอนการสร้างและส่งคำสั่ง จะใช้เมทอด (2) คือ createStatement
3. ขั้นตอนรับผลจากคำสั่ง จะใช้เมทอด (3)

เอพีไอของเจดีบีซี

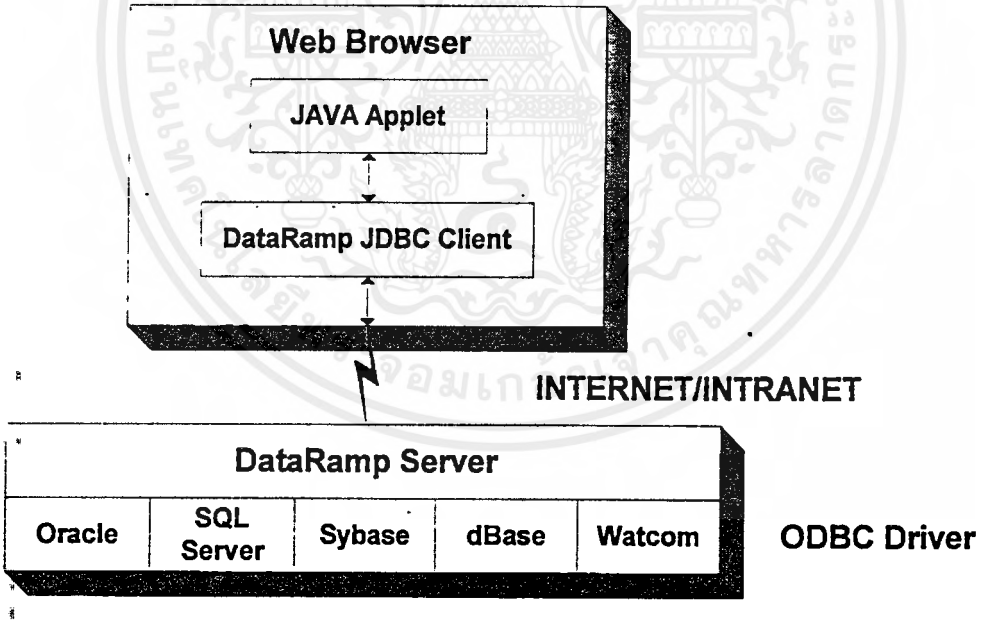
เอพีไอหลักของเจดีบีซี คือ **java.sql.*** ซึ่งทางบริษัทซัน ไมโครซิสเต็มส์ สร้างขึ้น จะเป็นเอพีไอในระดับล่าง (low-level API) ซึ่งจะเป็นฐานที่บริษัทผู้ผลิตฐานข้อมูลอื่น ๆ จะสามารถนำไปสร้างเอพีไอในระดับสูงขึ้นได้ โดยใช้เอพีไอของเจดีบีซีเป็นพื้นฐานหลัก โดยเอพีไอในระดับสูงขึ้นของบริษัทอื่น ๆ ที่สร้างขึ้นนี้จะทำให้การเรียกใช้งาน หรือ เขียนโปรแกรมติดต่อกับฐาน

ข้อมูลนั้น ๆ ทำได้ง่ายขึ้น ในขณะที่การใช้เอพีไอของเจดีบีซีนี้จะเป็นการติดต่อโดยตรงกับฐานข้อมูลนั้น ๆ เลย

สำหรับคลาส (class) หลัก ๆ ของเอพีไอของเจดีบีซี ก็จะเป็นคลาสที่ใช้ในการทำหน้าที่หลัก 3 อย่างที่กล่าวมาแล้ว คลาสหลัก ๆ จึงได้แก่

1. ***java.sql.Environment*** ใช้ในการสร้างการติดต่อกับฐานข้อมูล
2. ***java.sql.Connection*** ใช้ในการเทียบการติดต่อดาต้าสตรักเจอร์ของฐานข้อมูลต่าง ๆ
3. ***java.sql.Statement*** ใช้เก็บคำสั่งเอสคิวแอลที่สามารถเรียกใช้ได้
4. ***java.sql.ResultSet*** การเข้าจัดการกับผลที่ได้รับจากการเรียกใช้คำสั่งเอสคิวแอล
5. ***java.sql.Driver*** จะเก็บไดรฟ์เวอร์ที่เจดีบีซีสามารถติดต่อด้วยได้

สำหรับปัจจุบันมีหลาย ๆ บริษัทได้สร้างเอพีไอในระดับสูงที่สามารทำงานโดยอาศัยพื้นฐานของ เจดีบีซีเอพีไอ ในที่นี้จะขอยกตัวอย่างการทำงานของเอพีไอของบริษัทดาต้าแรมป์ ซึ่งมีการทำงานดามรูปที่ ค - 4



รูปที่ ค - 4 โมเดลการทำงานของเอพีไอของบริษัทดาต้าแรมป์

จากภาพจะเห็นว่าเอพีไอของดาต้าแรมพ์ จะทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่างแอพเพล็ทกับฐานข้อมูลที่ต้องการ โดยอาศัยเจดีบีซีเอพีไอเป็นฐานหลัก

ภาคผนวก ง
ซอร์สโค้ดของ CGI เฟรมเวิร์ก
สำหรับภาษาวิชวลเบสิก

```
Attribute VB_Name = "CGI_Framework"
'---------------------------------------------------------------
'     *************
'     * CGI32.BAS *
'     *************
'
' VERSION: 1.7  (December 3, 1995)
'
' AUTHOR:  Robert B. Denny <rdenny@netcom.com>
'
' Common routines needed to establish a VB environment for
' Windows CGI programs that run behind the WebSite Server.
'
' INTRODUCTION
'
' The Common Gateway Interface (CGI) version 1.1 specifies a minimal
' set of data that is made available to the back-end application by
' an HTTP (Web) server. It also specifies the details for passing this
' information to the back-end. The latter part of the CGI spec is
' specific to Unix-like environments. The NCSA httpd for Windows does
' supply the data items (and more) specified by CGI/1.1, however it
' uses a different method for passing the data to the back-end.
'
' DEVELOPMENT
'
' WebSite requires any Windows back-end program to be an
' executable image. This means that you must convert your VB
' application into an executable (.EXE) before it can be tested
' with the server.
'
' ENVIRONMENT
'
' The WebSite server executes script requests by doing a
' CreateProcess with a command line in the following form:
'
'   prog-name cgi-profile
'
' THE CGI PROFILE FILE
'
' The Unix CGI passes data to the back end by defining environment
' variables which can be used by shell scripts. The WebSite
' server passes data to its back end via the profile file. The
' format of the profile is that of a Windows ".INI" file. The keyword
' names have been changed cosmetically.
```

```
'
' There are 7 sections in a CGI profile file, [CGI], [Accept],
' [System], [Extra Headers], and [Form Literal], [Form External],
' and [Form huge]. They are described below:
'
' [CGI]              <== The standard CGI variables
' CGI Version=       The version of CGI spoken by the server
' Request Protocol=  The server's info protocol (e.g. HTTP/1.0)
' Request Method=    The method specified in the request (e.g., "GET")
' Request Keep-Alive= If the client requested connection re-use (Yes/No)
' Executable Path=   Physical pathname of the back-end (this program)
' Logical Path=      Extra path info in logical space
' Physical Path=     Extra path info in local physical space
' Query String=      String following the "?" in the request URL
' Content Type=      MIME content type of info supplied with request
' Content Length=    Length, bytes, of info supplied with request
' Request Range=     Byte-range specfication received with request
' Server Software=   Version/revision of the info (HTTP) server
' Server Name=       Server's network hostname (or alias from config)
' Server Port=       Server's network port number
' Server Admin=      E-Mail address of server's admin. (config)
' Referer=          URL of referring document
' From=             E-Mail of client user (rarely seen)
' User Agent=       String describing client/browser software/version
' Remote Host=      Remote client's network hostname
' Remote Address=   Remote client's network address
' Authenticated Username=Username if present in request
' Authenticated Password=Password if present in request
' Authentication Method=Method used for authentication (e.g., "Basic")
' Authentication Realm=Name of realm for users/groups
'
' [Accept]           <== What the client says it can take
' The MIME types found in the request header as
'   Accept: xxx/yyy; zzzz...
' are entered in this section as
'   xxx/yyy=zzzz...
' If only the MIME type appears, the form is
'   xxx/yyy=Yes
'
' [System]           <== Windows interface specifics
' GMT Offset=        Offset of local timezone from GMT, seconds (LONG!)
' Output File=       Pathname of file to receive results
' Content File=      Pathname of file containing raw request content
' Debug Mode=        If server's CGI debug flag is set (Yes/No)
'
' [Extra Headers]
' Any "extra" headers found in the request that activated this
' program. They are listed in "key=value" form. Usually, you'll see
' at least the name of the browser here as "User-agent".
'
' [Form Literal]
' If the request was a POST from a Mosaic form (with content type of
' "application/x-www-form-urlencoded"), the server will decode the
' form data. Raw form input is of the form "key=value&key=value&...",
' with the value parts "URL-encoded". The server splits the key=value
' pairs at the '&', then spilts the key and value at the '='.
```

```
' URL-decodes the value string and puts the result into key=value
' (decoded) form in the [Form Literal] section of the INI.
'
' [Form External]
' If the decoded value string is more than 254 characters long,
' or if the decoded value string contains any control characters
' or quote marks the server puts the decoded value into an external
' tempfile and lists the field in this section as:
'   key=<pathname> <length>
' where <pathname> is the path and name of the tempfile containing
' the decoded value string, and <length> is the length in bytes
' of the decoded value string.
'
' NOTE: BE SURE TO OPEN THIS FILE IN BINARY MODE UNLESS YOU ARE
'       CERTAIN THAT THE FORM DATA IS TEXT!
'
' [Form File]
' If the form data contained any uploaded files, they are described in
' this section as:
'   key=[<pathname>] <length> <type> <encoding> [<name>]
' where <pathname> is the path and name of the tempfile contining the
' uploaded file, <length> is the length in bytes of the uploaded file,
' <type> is the content type of the uploaded file as sent by the browser,
' <encoding> is the content-transfer encoding of the uploaded file, and
' <name> is the original file name of the uploaded file.
'
' [Form Huge]
' If the raw value string is more than 65,536 bytes long, the server
' does no decoding. In this case, the server lists the field in this
' section as:
'   key=<offset> <length>
' where <offset> is the offset from the beginning of the Content File
' at which the raw value string for this key is located, and <length>
' is the length in bytes of the raw value string. You can use the
' <offset> to perform a "Seek" to the start of the raw value string,
' and use the length to know when you have read the entire raw string
' into your decoder. Note that VB has a limit of 64K for strings, so
'
' Examples:
'
'   [Form Literal]
'   smallfield=123 Main St. #122
'
'   [Form External]
'   field300chars=c:\website\cgi-tmp\1a7fws.000 300
'   fieldwithlinebreaks=c:\website\cgi-tmp\1a7fws.001 43
'
'   [Form Huge]
'   field230K=c:\website\cgi-tmp\1a7fws.002 276920
'
' =====
' USAGE
' =====
' Include CGI32.BAS in your VB4 project. Set the project options for
' "Sub Main" startup. The Main() procedure is in this module, and it
' handles all of the setup of the VB CGI environment, as described
```

```
' above. Once all of this is done, the Main() calls YOUR main procedure
' which must be called CGI_Main(). The output file is open, use Send()
' to write to it. The input file is NOT open, and "huge" form fields
' have not been decoded.
'
' NOTE: If your program is started without command-line args,
' the code assumes you want to run it interactively. This is useful
' for providing a setup screen, etc. Instead of calling CGI_Main(),
' it calls Inter_Main(). Your module must also implement this
' function. If you don't need an interactive mode, just create
' Inter_Main() and put a 1-line call to MsgBox alerting the
' user that the program is not meant to be run interactively.
' The samples furnished with the server do this.
'
' If a Visual Basic runtime error occurs, it will be trapped and result
' in an HTTP error response being sent to the client. Check out the
' Error Handler() sub. When your program finishes, be sure to RETURN
' TO MAIN(). Don't just do an "End".
'
' Have a look at the stuff below to see what's what.
'
'---------------------------------------------------------------
' Author:   Robert B. Denny <rdenny@netcom.com>
'         April 15, 1995
'
' Revision History:
'   15-Apr-95 rbd   Initial release (ref VB3 CGI.BAS 1.7)
'   02-Aug-95 rbd   Changed to take input and output files from profile
'                   Server no longer produces long command line.
'   24-Aug-95 rbd   Make call to GetPrivateProfileString conditional
'                   so 16-bit and 32-bit versions supported. Fix
'                   computation of CGI_GMTOffset for offset=0 (GMT)
'                   case. Add FieldPresent() routine for checkbox
'                   handling. Clean up comments.
'   29-Oct-95 rbd   Added PlusToSpace() and Unescape() functions for
'                   decoding query strings, etc.
'   16-Nov-95 rbd   Add keep-alive variable, file uploading description
'                   in comments, and upload display.
'   20-Nov-95 rbd   Fencepost error in ParseFileValue()
'   23-Nov-95 rbd   Remove On Error Resume Next from error handler
'   03-Dec-95 rbd   User-Agent is now a variable, real HTTP header
'                   Add Request-Range as http header as well.
'---------------------------------------------------------------
Option Explicit
'
'=====================
' Manifest Constants
'=====================
'
Const MAX_CMDARGS = 8       ' Max # of command line args
Const ENUM_BUF_SIZE = 4096  ' Key enumeration buffer, see GetProfile()
' These are the limits in the server
Const MAX_XHDR = 100        ' Max # of "extra" request headers
Const MAX_ACCTYPE = 100     ' Max # of Accept: types in request
Const MAX_FORM_TUPLES = 100 ' Max # form key=value pairs
Const MAX_HUGE_TUPLES = 16  ' Max # "huge" form fields
```

```
Const MAX_FILE_TUPLES = 16  ' Max # of uploaded file tuples
'
'
'=====
' Types
'=====
'
Type Tuple              ' Used for Accept: and "extra" headers
   key As String        ' and for holding POST form key=value pairs
   value As String
End Type

Type FileTuple          ' Used for form-based file uploads
   key As String        ' Form field name
   file As String       ' Local tempfile containing uploaded file
   length As Long       ' Length in bytes of uploaded file
   type As String       ' Content type of uploaded file
   encoding As String   ' Content-transfer encoding of uploaded file
   name As String       ' Original name of uploaded file
End Type

Type HugeTuple          ' Used for "huge" form fields
   key As String        ' Keyword (decoded)
   offset As Long       ' Byte offset into Content File of value
   length As Long       ' Length of value, bytes
End Type
'
'
'================
' Global Constants
'================
'
'---------
' Error Codes
'---------
'
Global Const ERR_ARGCOUNT = 32767
Global Const ERR_BAD_REQUEST = 32766      ' HTTP 400
Global Const ERR_UNAUTHORIZED = 32765     ' HTTP 401
Global Const ERR_PAYMENT_REQUIRED = 32764  ' HTTP 402
Global Const ERR_FORBIDDEN = 32763        ' HTTP 403
Global Const ERR_NOT_FOUND = 32762        ' HTTP 404
Global Const ERR_INTERNAL_ERROR = 32761   ' HTTP 500
Global Const ERR_NOT_IMPLEMENTED = 32760  ' HTTP 501
Global Const ERR_TOO_BUSY = 32758         ' HTTP 503 (experimental)
Global Const ERR_NO_FIELD = 32757         ' GetxxxField "no field"
Global Const CGI_ERR_START = 32757        ' Start of our errors

'====================
' CGI Global Variables
'====================
'
'------------------
' Standard CGI variables
'------------------
'
```

```
Global CGI_ServerSoftware As String
Global CGI_ServerName As String
Global CGI_ServerPort As Integer
Global CGI_RequestProtocol As String
Global CGI_ServerAdmin As String
Global CGI_Version As String
Global CGI_RequestMethod As String
Global CGI_RequestKeepAlive As Integer
Global CGI_LogicalPath As String
Global CGI_PhysicalPath As String
Global CGI_ExecutablePath As String
Global CGI_QueryString As String
Global CGI_RequestRange As String
Global CGI_Referer As String
Global CGI_From As String
Global CGI_UserAgent As String
Global CGI_RemoteHost As String
Global CGI_RemoteAddr As String
Global CGI_AuthUser As String
Global CGI_AuthPass As String
Global CGI_AuthType As String
Global CGI_AuthRealm As String
Global CGI_ContentType As String
Global CGI_ContentLength As Long
'
' ----------
' HTTP Header Arrays
' ----------
'
Global CGI_AcceptTypes(MAX_ACCTYPE) As Tuple    ' Accept: types
Global CGI_NumAcceptTypes As Integer        ' # of live entries in array
Global CGI_ExtraHeaders(MAX_XHDR) As Tuple      ' "Extra" headers
Global CGI_NumExtraHeaders As Integer           ' # of live entries in array
'
' ----------
' POST Form Data
' ----------
'
Global CGI_FormTuples(MAX_FORM_TUPLES) As Tuple ' POST form key=value pairs
Global CGI_NumFormTuples As Integer             ' # of live entries in array
Global CGI_HugeTuples(MAX_HUGE_TUPLES) As HugeTuple ' Form "huge tuples
Global CGI_NumHugeTuples As Integer             ' # of live entries in array
Global CGI_FileTuples(MAX_FILE_TUPLES) As FileTuple ' File upload tuples
Global CGI_NumFileTuples As Integer             ' # of live entries in array
'
' ----------
' System Variables
' ----------
'
Global CGI_GMTOffset As Variant         ' GMT offset (time serial)
Global CGI_ContentFile As String        ' Content/Input file pathname
Global CGI_OutputFile As String         ' Output file pathname
Global CGI_DebugMode As Integer         ' Script Tracing flag from server
'
'
' ==========
```

```
' Windows API Declarations
' ========================
'
' NOTE: Declaration of GetPrivateProfileString is specially done to
' permit enumeration of keys by passing NULL key value. See GetProfile().
' Both the 16-bit and 32-bit flavors are given below. We DO NOT
' recommend using 16-bit VB4 with WebSite!
'
#If Win32 Then
Declare Function GetPrivateProfileString Lib "kernel32" _
   Alias "GetPrivateProfileStringA" _
  (ByVal lpApplicationName As String, _
   ByVal lpKeyName As Any, _
   ByVal lpDefault As String, _
   ByVal lpReturnedString As String, _
   ByVal nSize As Long, _
   ByVal lpFileName As String) As Long
#Else
Declare Function GetPrivateProfileString Lib "Kernel" _
  (ByVal lpSection As String, _
   ByVal lpKeyName As Any, _
   ByVal lpDefault As String, _
   ByVal lpReturnedString As String, _
   ByVal nSize As Integer, _
   ByVal lpFileName As String) As Integer
#End If
'
'
' ================
' Local Variables
' ================
'
Dim CGI_ProfileFile As String        ' Profile file pathname
Dim CGI_OutputFN As Integer          ' Output file number
Dim ErrorString As String

'-----------------------------------------------------------
'
' Return True/False depending on whether a form field is present.
' Typically used to detect if a checkbox in a form is checked or
' not. Unchecked checkboxes are omitted from the form content.
'
'-----------------------------------------------------------
Function FieldPresent(key As String) As Integer
   Dim i As Integer

   FieldPresent = False          ' Assume failure

   For i = 0 To (CGI_NumFormTuples - 1)
      If CGI_FormTuples(i).key = key Then
         FieldPresent = True    ' Found it
         Exit Function          ' ** DONE **
      End If
   Next i
                                ' Exit with FieldPresent still False
End Function
```

```
'---------------------------------------------------------------
'
'  ErrorHandler() - Global error handler
'
' If a VB runtime error occurs dusing execution of the program, this
' procedure generates an HTTP/1.0 HTML-formatted error message into
' the output file, then exits the program.
'
' This should be armed immediately on entry to the program's main()
' procedure. Any errors that occur in the program are caught, and
' an HTTP/1.0 error messsage is generated into the output file. The
' presence of the HTTP/1.0 on the first line of the output file causes
' NCSA httpd for WIndows to send the output file to the client with no
' interpretation or other header parsing.
'---------------------------------------------------------------
Sub ErrorHandler(code As Integer)

    Seek #CGI_OutputFN, 1    ' Rewind output file just in case
    Send ("HTTP/1.0 500 Internal Error")
    Send ("Server: " + CGI_ServerSoftware)
    Send ("Date: " + WebDate(Now))
    Send ("Content-type: text/html")
    Send ("")
    Send ("<HTML><HEAD>")
    Send ("<TITLE>Error in " + CGI_ExecutablePath + "</TITLE>")
    Send ("</HEAD><BODY>")
    Send ("<H1>Error in " + CGI_ExecutablePath + "</H1>")
    Send ("An internal Visual Basic error has occurred in " + CGI_ExecutablePath + ".")
    Send ("<PRE>" + ErrorString + "</PRE>")
    Send ("<I>Please</I> note what you were doing when this problem occurred,")
    Send ("so we can identify and correct it. Write down the Web page you were using,")
    Send ("any data you may have entered into a form or search box, and")
    Send ("anything else that may help us duplicate the problem. Then contact the")
    Send ("administrator of this service: ")
    Send ("<A HREF=""mailto:" & CGI_ServerAdmin & """>")
    Send ("<ADDRESS>&lt;" + CGI_ServerAdmin + "&gt;</ADDRESS>")
    Send ("</A></BODY></HTML>")

    Close #CGI_OutputFN

    '=======
    End          ' Terminate the program
    '=======
End Sub

'---------------------------------------------------------------
'
'  GetAcceptTypes() - Create the array of accept type structs
'
' Enumerate the keys in the [Accept] section of the profile file,
' then get the value for each of the keys.
'---------------------------------------------------------------
Private Sub GetAcceptTypes()
'
```

```
    Dim sList As String
    Dim i As Integer, j As Integer, l As Integer, n As Integer

    sList = GetProfile("Accept", "") ' Get key list
    l = Len(sList)                   ' Length incl. trailing null
    i = 1                            ' Start at 1st character
    n = 0                            ' Index in array
    Do While ((i < l) And (n < MAX_ACCTYPE)) ' Safety stop here
      j = InStr(i, sList, Chr$(0))   ' J -> next null
      CGI_AcceptTypes(n).key = Mid$(sList, i, j - i) ' Get Key, then value
      CGI_AcceptTypes(n).value = GetProfile("Accept", CGI_AcceptTypes(n).key)
      i = j + 1                      ' Bump pointer
      n = n + 1                      ' Bump array index
    Loop
    CGI_NumAcceptTypes = n           ' Fill in global count

End Sub

'-----------------------------------------------------------
'
' GetArgs() - Parse the command line
'
' Chop up the command line, fill in the argument vector, return the
' argument count (similar to the Unix/C argc/argv handling)
'-----------------------------------------------------------
Private Function GetArgs(argv() As String) As Integer
    Dim buf As String
    Dim i As Integer, j As Integer, l As Integer, n As Integer

    buf = Trim$(Command$)            ' Get command line

    l = Len(buf)                     ' Length of command line
    If l = 0 Then                    ' If empty
      GetArgs = 0                    ' Return argc = 0
      Exit Function
    End If

    i = 1                            ' Start at 1st character
    n = 0                            ' Index in argvec
    Do While ((i < l) And (n < MAX_CMDARGS)) ' Safety stop here
      j = InStr(i, buf, " ")         ' J -> next space
      If j = 0 Then Exit Do          ' Exit loop on last arg
      argv(n) = Trim$(Mid$(buf, i, j - i)) ' Get this token, trim it
      i = j + 1                      ' Skip that blank
      Do While Mid$(buf, i, 1) = " "  ' Skip any additional whitespace
        i = i + 1
      Loop
      n = n + 1                      ' Bump array index
    Loop

    argv(n) = Trim$(Mid$(buf, i, (l - i + 1))) ' Get last arg
    GetArgs = n + 1                  ' Return arg count

End Function

'-----------------------------------------------------------
```

```
'
'  GetExtraHeaders() - Create the array of extra header structs
'
' Enumerate the keys in the [Extra Headers] section of the profile file,
' then get the value for each of the keys.
'--------------------------------------------------------------------
Private Sub GetExtraHeaders()
   Dim sList As String
   Dim i As Integer, j As Integer, l As Integer, n As Integer

   sList = GetProfile("Extra Headers", "") ' Get key list
   l = Len(sList)                    ' Length incl. trailing null
   i = 1                             ' Start at 1st character
   n = 0                             ' Index in array
   Do While ((i < l) And (n < MAX_XHDR))  ' Safety stop here
      j = InStr(i, sList, Chr$(0))       ' J -> next null
      CGI_ExtraHeaders(n).key = Mid$(sList, i, j - i) ' Get Key, then value
      CGI_ExtraHeaders(n).value = GetProfile("Extra Headers", CGI_ExtraHeaders(n).key)
      i = j + 1                      ' Bump pointer
      n = n + 1                      ' Bump array index
   Loop
   CGI_NumExtraHeaders = n           ' Fill in global count

End Sub

'--------------------------------------------------------------------
'
'  GetFormTuples() - Create the array of POST form input key=value pairs
'
'--------------------------------------------------------------------
Private Sub GetFormTuples()
   Dim sList As String
   Dim i As Integer, j As Integer, k As Integer
   Dim l As Integer, m As Integer, n As Integer
   Dim s As Long
   Dim buf As String
   Dim extName As String
   Dim extFile As Integer
   Dim extlen As Long

   n = 0                             ' Index in array

   '
   ' Do the easy one first: [Form Literal]
   '
   sList = GetProfile("Form Literal", "")  ' Get key list
   l = Len(sList)                    ' Length incl. trailing null
   i = 1                             ' Start at 1st character
   Do While ((i < l) And (n < MAX_FORM_TUPLES)) ' Safety stop here
      j = InStr(i, sList, Chr$(0))       ' J -> next null
      CGI_FormTuples(n).key = Mid$(sList, i, j - i) ' Get Key, then value
      CGI_FormTuples(n).value = GetProfile("Form Literal", CGI_FormTuples(n).key)
      i = j + 1                      ' Bump pointer
      n = n + 1                      ' Bump array index
   Loop
   '
```

```
' Now do the external ones: [Form External]
'
sList = GetProfile("Form External", "") ' Get key list
l = Len(sList)                    ' Length incl. trailing null
i = 1                             ' Start at 1st character
extFile = FreeFile
Do While ((i < l) And (n < MAX_FORM_TUPLES)) ' Safety stop here
   j = InStr(i, sList, Chr$(0))       ' J -> next null
   CGI_FormTuples(n).key = Mid$(sList, i, j - i) ' Get Key, then pathname
   buf = GetProfile("Form External", CGI_FormTuples(n).key)
   k = InStr(buf, " ")               ' Split file & length
   extName = Mid$(buf, 1, k - 1)          ' Pathname
   k = k + 1
   extlen = CLng(Mid$(buf, k, Len(buf) - k + 1)) ' Length
   '
   ' Use feature of GET to read content in one call
   '
   Open extName For Binary Access Read As #extFile
   CGI_FormTuples(n).value = String$(extlen, " ") ' Breathe in...
   Get #extFile, , CGI_FormTuples(n).value 'GULP!
   Close #extFile
   i = j + 1                      ' Bump pointer
   n = n + 1                      ' Bump array index
Loop

CGI_NumFormTuples = n              ' Number of fields decoded
n = 0                             ' Reset counter
'
' Next, the [Form Huge] section. Will this ever get executed?
'
sList = GetProfile("Form Huge", "")    ' Get key list
l = Len(sList)                    ' Length incl. trailing null
i = 1                             ' Start at 1st character
Do While ((i < l) And (n < MAX_FORM_TUPLES)) ' Safety stop here
   j = InStr(i, sList, Chr$(0))       ' J -> next null
   CGI_HugeTuples(n).key = Mid$(sList, i, j - i) ' Get Key
   buf = GetProfile("Form Huge", CGI_HugeTuples(n).key) ' "offset length"
   k = InStr(buf, " ")               ' Delimiter
   CGI_HugeTuples(n).offset = CLng(Mid$(buf, 1, (k - 1)))
   CGI_HugeTuples(n).length = CLng(Mid$(buf, k, (Len(buf) - k + 1)))
   i = j + 1                      ' Bump pointer
   n = n + 1                      ' Bump array index
Loop

CGI_NumHugeTuples = n              ' Fill in global count

n = 0                             ' Reset counter
'
' Finally, the [Form File] section.
'
sList = GetProfile("Form File", "")    ' Get key list
l = Len(sList)                    ' Length incl. trailing null
i = 1                             ' Start at 1st character
Do While ((i < l) And (n < MAX_FILE_TUPLES)) ' Safety stop here
   j = InStr(i, sList, Chr$(0))       ' J -> next null
   CGI_FileTuples(n).key = Mid$(sList, i, j - i) ' Get Key
```

```
    buf = GetProfile("Form File", CGI_FileTuples(n).key)
    ParseFileValue buf, CGI_FileTuples(n)  ' Complicated, use Sub
    i = j + 1                   ' Bump pointer
    n = n + 1                   ' Bump array index
  Loop

  CGI_NumFileTuples = n             ' Fill in global count

End Sub

'---------------------------------------------------------------
'
'  GetProfile() - Get a value or enumerate keys in CGI_Profile file
'
' Get a value given the section and key, or enumerate keys given the
' section name and "" for the key. If enumerating, the list of keys for
' the given section is returned as a null-separated string, with a
' double null at the end.
'
' VB handles this with flair! I couldn't believe my eyes when I tried this.
'---------------------------------------------------------------
Private Function GetProfile(sSection As String, sKey As String) As String
  Dim retLen As Long
  Dim buf As String * ENUM_BUF_SIZE

  If sKey <> "" Then
    retLen = GetPrivateProfileString(sSection, sKey, "", buf, ENUM_BUF_SIZE, CGI_ProfileFile)
  Else
    retLen = GetPrivateProfileString(sSection, 0&, "", buf, ENUM_BUF_SIZE, CGI_ProfileFile)
  End If
  If retLen = 0 Then
    GetProfile = ""
  Else
    GetProfile = Left$(buf, retLen)
  End If

End Function

'---------------------------------------------------------------
'
' Get the value of a "small" form field given the key
'
' Signals an error if field does not exist
'
'---------------------------------------------------------------
Function GetSmallField(key As String) As String
  Dim i As Integer

  For i = 0 To (CGI_NumFormTuples - 1)
    If CGI_FormTuples(i).key = key Then
      GetSmallField = Trim$(CGI_FormTuples(i).value)
      Exit Function         ' ** DONE **
    End If
  Next i
  '
  ' Field does not exist
```

```
    '
    Error ERR_NO_FIELD
End Function

'--------------------------------------------------------------------
'
'  InitializeCGI() - Fill in all of the CGI variables, etc.
'
' Read the profile file name from the command line, then fill in
' the CGI globals, the Accept type list and the Extra headers list.
' Then open the input and output files.
'
' Returns True if OK, False if some sort of error. See ReturnError()
' for info on how errors are handled.
'
' NOTE: Assumes that the CGI error handler has been armed with On Error
'--------------------------------------------------------------------
Sub InitializeCGI()
    Dim sect As String
    Dim argc As Integer
    Static argv(MAX_CMDARGS) As String
    Dim buf As String

    CGI_DebugMode = True    ' Initialization errors are very bad

    '
    ' Parse the command line. We need the profile file name (duh!)
    ' and the output file name NOW, so we can return any errors we
    ' trap. The error handler writes to the output file.
    '
    argc = GetArgs(argv())
    CGI_ProfileFile = argv(0)

    sect = "CGI"
    CGI_ServerSoftware = GetProfile(sect, "Server Software")
    CGI_ServerName = GetProfile(sect, "Server Name")
    CGI_RequestProtocol = GetProfile(sect, "Request Protocol")
    CGI_ServerAdmin = GetProfile(sect, "Server Admin")
    CGI_Version = GetProfile(sect, "CGI Version")
    CGI_RequestMethod = GetProfile(sect, "Request Method")
    buf = GetProfile(sect, "Request Keep-Alive")    ' Y or N
    If (Left$(buf, 1) = "Y") Then                   ' Must start with Y
        CGI_RequestKeepAlive = True
    Else
        CGI_RequestKeepAlive = False
    End If
    CGI_LogicalPath = GetProfile(sect, "Logical Path")
    CGI_PhysicalPath = GetProfile(sect, "Physical Path")
    CGI_ExecutablePath = GetProfile(sect, "Executable Path")
    CGI_QueryString = GetProfile(sect, "Query String")
    CGI_RemoteHost = GetProfile(sect, "Remote Host")
    CGI_RemoteAddr = GetProfile(sect, "Remote Address")
    CGI_RequestRange = GetProfile(sect, "Request Range")
    CGI_Referer = GetProfile(sect, "Referer")
    CGI_From = GetProfile(sect, "From")
    CGI_UserAgent = GetProfile(sect, "User Agent")
```

```
    CGI_AuthUser = GetProfile(sect, "Authenticated Username")
    CGI_AuthPass = GetProfile(sect, "Authenticated Password")
    CGI_AuthRealm = GetProfile(sect, "Authentication Realm")
    CGI_AuthType = GetProfile(sect, "Authentication Method")
    CGI_ContentType = GetProfile(sect, "Content Type")
    buf = GetProfile(sect, "Content Length")
    If buf = "" Then
       CGI_ContentLength = 0
    Else
       CGI_ContentLength = CLng(buf)
    End If
    buf = GetProfile(sect, "Server Port")
    If buf = "" Then
       CGI_ServerPort = -1
    Else
       CGI_ServerPort = CInt(buf)
    End If

    sect = "System"
    CGI_ContentFile = GetProfile(sect, "Content File")
    CGI_OutputFile = GetProfile(sect, "Output File")
    CGI_OutputFN = FreeFile
    Open CGI_OutputFile For Output Access Write As #CGI_OutputFN
    buf = GetProfile(sect, "GMT Offset")
    If buf <> "" Then                    ' Protect against errors
       CGI_GMTOffset = CVDate(Val(buf) / 86400#) ' Timeserial GMT offset
    Else
       CGI_GMTOffset = 0
    End If
    buf = GetProfile(sect, "Debug Mode")   ' Y or N
    If (Left$(buf, 1) = "Y") Then          ' Must start with Y
       CGI_DebugMode = True
    Else
       CGI_DebugMode = False
    End If

    GetAcceptTypes        ' Enumerate Accept: types into tuples
    GetExtraHeaders       ' Enumerate extra headers into tuples
    GetFormTuples         ' Decode any POST form input into tuples

End Sub

'-------------------------------------------------------
'
'  main() - CGI script back-end main procedure
'
' This is the main() for the VB back end. Note carefully how the error
' handling is set up, and how program cleanup is done. If no command
' line args are present, call Inter_Main() and exit.
'-------------------------------------------------------
Sub Main()
    On Error GoTo ErrorHandler

    If Trim$(Command$) = "" Then    ' Interactive start
       inter_main                   ' Call interactive main
       Exit Sub                     ' Exit the program
```

```
    End If

    InitializeCGI      ' Create the CGI environment

    '============
    CGI_Main           ' Execute the actual "script"
    '============

Cleanup:
    Close #CGI_OutputFN
    Exit Sub                    ' End the program
'------------
ErrorHandler:
    Select Case Err             ' Decode our "user defined" errors
        Case ERR_NO_FIELD:
            ErrorString = "Unknown form field"
        Case Else:
            ErrorString = Error$    ' Must be VB error
    End Select

    ErrorString = ErrorString & " (error #" & Err & ")"
    On Error GoTo 0             ' Prevent recursion
    ErrorHandler (Err)          ' Generate HTTP error result
    Resume Cleanup
'------------
End Sub

'-----------------------------------------------------------
'
' Send() - Shortcut for writing to output file
'
'-----------------------------------------------------------
Sub Send(s As String)
    Print #CGI_OutputFN, s
End.Sub

'-----------------------------------------------------------
'
'  SendNoOp() - Tell browser to do nothing.
'
' Most browsers will do nothing. Netscape 1.0N leaves hourglass
' cursor until the mouse is waved around. Enhanced Mosaic 2.0
' oputs up an alert saying "URL leads nowhere". Your results may
' vary...
'
'-----------------------------------------------------------
Sub SendNoOp()

    Send ("HTTP/1.0 204 No Response")
    Send ("Server: " + CGI_ServerSoftware)
    Send ("")

End Sub

'-----------------------------------------------------------
```

```
'  WebDate - Return an HTTP/1.0 compliant date/time string
'
' Inputs:   t = Local time as VB Variant (e.g., returned by Now())
' Returns:  Properly formatted HTTP/1.0 date/time in GMT
'-------------------------------------------------------------
Function WebDate(dt As Variant) As String
   Dim t As Variant

   t = CVDate(dt - CGI_GMTOffset)     ' Convert time to GMT
   WebDate = Format$(t, "ddd dd mmm yyyy hh:mm:ss") & " GMT"

End Function



'-------------------------------------------------------------
'
' PlusToSpace() - Remove plus-delimiters from HTTP-encoded string
'
'-------------------------------------------------------------
Public Sub PlusToSpace(s As String)
   Dim i As Integer

   i = 1
   Do While True
      i = InStr(i, s, "+")
      If i = 0 Then Exit Do
      Mid$(s, i) = " "
   Loop

End Sub

'-------------------------------------------------------------
'
' Unescape() - Convert HTTP-escaped string to normal form
'
'-------------------------------------------------------------
Public Function Unescape(s As String)
   Dim i As Integer, l As Integer
   Dim c As String

   If InStr(s, "%") = 0 Then           ' Catch simple case
      Unescape = s
      Exit Function
   End If

   l = Len(s)
   Unescape = ""
   For i = 1 To l
      c = Mid$(s, i, 1)                ' Next character
      If c = "%" Then
         If Mid$(s, i + 1, 1) = "%" Then
            c = "%"
            i = i + 1                  ' Loop increments too
         Else
```

```
            c = x2c(Mid$(s, i + 1, 2))
            i = i + 2               ' Loop increments too
          End If
        End If
        Unescape = Unescape & c
    Next i

End Function

'-------------------------------------------------------------
'
' x2c() - Convert hex-escaped character to ASCII
'
'-------------------------------------------------------------
Private Function x2c(s As String) As String
    Dim t As String

    t = "&H" & s
    x2c = Chr$(CInt(t))

End Function


Private Sub ParseFileValue(buf As String, ByRef t As FileTuple)
    Dim i, j, k, l As Integer

    l = Len(buf)

    i = InStr(buf, " ")             ' First delimiter
    t.file = Mid$(buf, 1, (i - 1))      ' [file]
    t.file = Mid$(t.file, 2, Len(t.file) - 2)  ' file

    j = InStr((i + 1), buf, " ")        ' Next delimiter
    t.length = CLng(Mid$(buf, (i + 1), (j - i - 1)))
    i = j

    j = InStr((i + 1), buf, " ")        ' Next delimiter
    t.type = Mid$(buf, (i + 1), (j - i - 1))
    i = j

    j = InStr((i + 1), buf, " ")        ' Next delimiter
    t.encoding = Mid$(buf, (i + 1), (j - i - 1))
    i = j

    t.name = Mid$(buf, (i + 1), (l - i - 1))  ' [name]
    t.name = Mid$(t.name, 2, Len(t.name) - 1) ' name

End Sub

'-------------------------------------------------------------
'
'  FindExtraHeader() - Get the text from an "extra" header
'
' Given the extra header's name, return the stuff after the ":"
' or an empty string if not there.
'-------------------------------------------------------------
```

```
Public Function FindExtraHeader(key As String) As String
  Dim i As Integer

  For i = 0 To (CGI_NumExtraHeaders - 1)
    If CGI_ExtraHeaders(i).key = key Then
      FindExtraHeader = Trim$(CGI_ExtraHeaders(i).value)
      Exit Function        ' ** DONE **
    End If
  Next i
  '
  ' Not present, return empty string
  '
  FindExtraHeader = ""
End Function
```

ภาคผนวก จ

ซอร์สโค้ดของ CGI เฟรมเวิร์กสำหรับ Delphi

```
unit Cgi;

{ cgi.pas

  Original Author: Ann Lynnworth
  Copyright (c) 1995-1996, Ann Lynnworth.  All Rights Reserved.

  Thanks to Fred Thompson for adding getSmallMultiField().

  Thanks to Dagur Georgsson for testing and debugging the
  internationalization of getSmallField.

  Thanks to Shane Hall <shaneh@clyde.its.unimelb.edu.au> for
  making it work with Microsoft Internet Information Server 1.0.
  Shane's company is Web Down Under P/L in Australia.

  This program may be freely used and modified by anyone.  It would be
  considerate to keep at least these credits and copyright notice intact.

  It is distributed with a "don't laugh at my code" disclaimer.
  This was my first attempt at writing a Delphi component back
  in June '95.  If I change it now, hundreds of web-applications
  will break.  So I'm leaving the data structures alone.

  What would I do different?  For starters, I wouldn't use pstrings
  on the published properties!

  URLs of note:

  http://super.sonic.net/ann/delphi/cgicomp/ -- home of this component
  http://www.href.com/ -- home of my company, HREF Tools Corp., with newer cgi tools
  http://website.ora.com/ -- home of WebSite 32-bit server
  http://www.borland.com/ -- you remember Borland; they made Delphi for us <g>
}

{ Technical support -- sorry, there isn't any.  This is a FREE component.

  Here are the 3 things I usually tell people to get them started:

  1. download the free demo project from http://super.sonic.net/ann/delphi/cgicomp/code.html
  2. If you're following the directions in cgihelp.hlp, make sure you also
     connect the form create method to the form's onCreate event handler.
     That's easy to overlook and of course your app won't work.
  3. To test, you need to run the .exe from a browser using an http command
     in the form: http://127.0.0.1/cgi-win/demo1.exe

     That IP references your local drive.  You can use any other IP
```

```
    or domain name.

    You will not be able to test or debug your web-application within Delphi.

    These components do not work as-is with Netscape server, at least
    not with Netscape's implementation of win-cgi as of 2/23/96.
    They only work with WebSite server from O'Reilly & Associates.
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  forms, iniFiles, Dialogs;

type
  NTWebServerType = (WebSite);   {only one choice; I meant to have more}

type
  TWebServer = class(TComponent);

type
  TCGIEnvData = class(TComponent)
  private
    { Private declarations -- custom for this component }
    fServerType : NTWebServerType;
    fServerComponent : TWebServer;
    fStdOut : integer;
    fAddress : string;
    { for use with WebSite only }
    finiFilename : string;
    {CGI section of web site INI file}
    fCGICGIVersion : string;
    fCGIRequestProtocol : string;
    fCGIRequestMethod : string;  { 'GET' or 'POST' -- should be POST }
    fCGIExecutablePath : string;
    fCGILogicalPath : string;
    fCGIPhysicalPath : string;
    fCGIQueryString : string;
    fCGIContentType : string;
    fCGIContentLength : longInt;
    fCGIServerSoftware : string;
    fCGIServerName : string;
    fCGIServerPort : string;
    fCGIServerAdmin : string;
    fCGIReferer : string;
    fCGIFrom : string;
    fCGIRemoteHost : string;
    fCGIRemoteAddress : string;
    fCGIAuthenticatedUsername : string;
    fCGIAuthenticatedPassword : string;
    fCGIAuthenticationMethod : string;
    fCGIAuthenticationRealm : string;
    fSystemGMTOffset : double;
    fSystemOutputFile : string;
    fSystemContentFile : string;
```

```
 fSystemDebugMode : string;
 {Custom Private Procedures & Functions }
 procedure getCGIItem( p : pString; key : string; okEmpty : boolean );
 { CGI }
 function  getCGICGIVersion : pstring;
 function  getCGIRequestProtocol : pstring;
 function  getCGIRequestMethod : pstring;
 function  getCGIExecutablePath : pstring;
 function  getCGILogicalPath : pstring;
 function  getCGIPhysicalPath : pString;
 function  getCGIQueryString : pString;
 function  getCGIContentType : pString;
 function  getCGIContentLength : longInt;
 function  getCGIServerSoftware : pstring;
 function  getCGIServerName : pstring;
 function  getCGIServerPort : pString;
 function  getCGIServerAdmin : pString;
 function  getCGIReferer : pString;
 function  getCGIFrom : pString;
 function  getCGIRemoteHost : pString;
 function  getCGIRemoteAddress : pString;
 function  getCGIAuthenticatedUsername : pString;
 function  getCGIAuthenticatedPassword : pString;
 function  getCGIAuthenticationMethod  : pString;
 function  getCGIAuthenticationRealm   : pString;
 { system }
 {function  getSystemGMTOffset: pstring;}
 function  getSystemOutputFile : pstring;
 function  getSystemDebugMode : pstring;
 function  getSystemContentFile : pstring;
protected
 { Protected declarations }
 constructor Create(AOwner: TComponent); override;
 destructor  Destroy; override;
public
 { Public declarations }
 { misc }

 { WebSite only }
 procedure setIniFilename( value : string );
 {CGI}
 { Regarding all these pstrings.  I didn't know better.  I was trying to save
  255 bytes x this many properties.  "Don't laugh at my code."  Or laugh away,
  just do it in private. <g> }
 property  CGICGIVersion     : pstring read getCGICGIVersion stored false;
 property  CGIRequestProtocol : pstring read getCGIRequestProtocol stored false;
 property  CGIRequestMethod  : pstring read getCGIRequestMethod stored false;
 property  CGIExecutablePath  : pstring read getCGIExecutablePath stored false;
 property  CGILogicalPath    : pstring read getCGILogicalPath stored false;
 property  CGIPhysicalPath   : pstring read getCGIPhysicalPath stored false;
 property  CGIQueryString    : pString read getCGIQueryString stored false;
 property  CGIContentType    : pString read getCGIContentType stored false;
 property  CGIContentLength  : longInt read getCGIContentLength stored false;
 property  CGIServerSoftware : pString read getCGIServerSoftware stored false;
 property  CGIServerPort     : pString read getCGIServerPort stored false;
 property  CGIServerName     : pString read getCGIServerName stored false;
```

```
  property  CGIServerAdmin    : pstring read getCGIServerAdmin stored false;
  property  CGIReferer        : pString read getCGIReferer stored false;
  property  CGIFrom           : pString read getCGIFrom stored false;
  property  CGIRemoteHost     : pString read getCGIRemoteHost stored false;
  property  CGIRemoteAddress  : pString read getCGIRemoteAddress stored false;
  property  CGIAuthenticatedUsername : pString read getCGIAuthenticatedUsername stored false;
  property  CGIAuthenticatedPassword : pString read getCGIAuthenticatedPassword stored false;
  property  CGIAuthenticationMethod  : pString read getCGIAuthenticationMethod  stored false;
  property  CGIAuthenticationRealm   : pString read getCGIAuthenticationRealm   stored false;
  {System}
  property  SystemGMToffset : double read fSystemGMToffset stored false;
  property  SystemOutputFile : pstring read getSystemOutputFile stored false;
  property  SystemContentFile : pstring read getSystemContentFile stored false;
  property  SystemDebugMode : pstring read getSystemDebugMode stored false;
 published
  { Published declarations }

  { set this to your address, e.g. ann@href.com }
  property    address : string read fAddress write fAddress;

  { ServerTypes WebSite and httpd16 are functionally equivalent. }
  { The whole issue of ServerType is silly. }
  { Property is left in for compatibility only. 1-Jan-96 }
  property    ServerType : NTWebServerType read fServerType write fServerType;

  function    swapChar( s : string; fromChar : char; toChar : char ) : string;

  { set this to paramstr(1) at the beginning of your program }
  property    webSiteIniFilename : string read finiFilename write setIniFilename;

  { ****************************** }

  { Use the LOCATION: URL feature to "bounce" a user to a URL }
  procedure   bounceToLocation( goHere : string );

  { set application.onException to TCGIEnvData1.cgiErrorHandler as soon as you can in your app! }
  procedure   cgiErrorHandler( sender : TObject; e : Exception ) ;

  { call this at the end of your program }
  procedure   closeStdout;

  { This opens the stdout file based on filename created by WebSite;
    if you forget this line, the first send command will take care of it
    for you automatically. }
  function    createStdout : boolean;

  { get data from a named External field {size between 255 and 64K chars
    and put it into a PChar.  If you're basically working with input from
    a TextArea on a form, see getTextArea below.  It will be much more
    convenient. }
  function    getExternalField( key : string; var externFilename : string; dest : PChar ) : boolean;

  { get everything in a section ('Form Literal' or 'Form External').  Refer to
    readSectionValues in Delphi Help. This is the same thing -- it just automatically
    goes to the correct INI file for you. }
  function    getSectionValues( sectionName : string; strings : TStringList ) : boolean;
```

```
{ get data from an HTML form based on field name ("key") }
function   getSmallField( key : string ) : string;

{*************************************************************************}
{*** getSmallMultiField - added Dec. 17, 1995 - Fred Thompson **********}
{*************************************************************************}
{ get Multiple data from an HTML form based on field name ("key")      }
{ Return value contains all the values selected.                       }
function   getSmallMultiField( key : string ) : Tstringlist;
{*************************************************************************}

{ TextAreas are tricky.  If the user only enters one line of text, that
  text is stored as a "small field" in the [Form Literal] section.  This
  function hides that complexity and lets you simply work with a string
  list (which might only have one string in it).  }
function   getTextArea( key : string; dest : TStringList ) : boolean;

{ send a line of code to stdout (including required CR/LF) }
function   send( s : string ) : boolean ;
function   sendString( s : string; appendNewline : boolean ) : boolean;

{ send contents of Address property }
function   sendAddress : boolean;

function   sendAuthRequest : boolean;

{ send wallpaper background command (HTML 3.0) -- no color control yet }
function   sendBackground( imageFilename : string ) : boolean;

{ send a string to stdout, as a comment.  This is used internally to
  alert you to warnings/errors. }
procedure  sendComment( s : string );

{ send header, e.g. H1, H2, etc. }
function   sendHdr( hdrLevel : char; hdrText : string ) : boolean;

{ send horizonal ruler line command }
function   sendHR : boolean;

{ send A HREF command including optional image and optional (netscape) attributes
  such as align=left or hspace=5 }
function   sendHREF( imageFilename : string; imageAttrib : string;
             visiblePhrase : string; linkedURL : string ) : boolean;

{ send a simple IMG SRC phase.  attrib can be hspace=5 or align=left }
function   sendIMG( imageFilename : string; imageAttrib : string ) : boolean;

procedure  sendMailto( emailAddress : string );

{ do nothing; copied from Bob Denny's cgi.bas.  Bob Denny is the author of
  Win-Httpd and WebSite server.  He has my endless gratitude for answering
  my endless questions in May '95. }
procedure  sendNoOp;

{ send HTTP/1.0 200 OK etc. }
```

```
    function    sendPrologue : boolean;

    { send TITLE phrase }
    function    sendTitle( title : string ) : boolean;

    { convert Delphi date/time to GMT for use in HTML header }
    function    webDate (dt : TDateTime ) : string ;

  end;

{*********************************************************************}
{*********************************************************************}

type
  TWebsite = class(TWebServer)
  private
    fServerType : NTWebServerType;
    fCGI : TCGIEnvData;
    fIniFile : TIniFile;
    { custom }
    procedure CGIData( p : pString; key : string; okEmpty : boolean );
    procedure initData;
    function  readWebSiteCGIString( key : string; okEmpty : boolean ) : string;
    function  getExternalField( key : string; var externFilename : string; dest : PChar ) : boolean;
    function  getTextArea( key : string; dest : TStringList ) : boolean;
  public
    { Public declarations }
    property    INIFile: TIniFile read fIniFile stored false;
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    function    getSmallField( key : string ) : string;
    function    getSmallMultiField( key: string) :Tstringlist;    {*FWT*}
end;

const
    MAXTABLEFIELDS = 255;  { no more than 255 fields displayed in HTML Table }

    CGINOTFOUND = 'AAAKEY NOT FOUND';

    MAX_CMDARGS = 8;       { Max # of command line args }
    ENUM_BUF_SIZE = 4096;  { Key enumeration buffer, see GetProfile() }
    { These are the limits in the server }
    MAX_XHDR = 100;        { Max # of "extra" request headers }
    MAX_ACCTYPE = 100;     { Max # of Accept: types in request }
    MAX_FORM_TUPLES = 100; { Max # form key=value pairs }
    MAX_HUGE_TUPLES = 16;  { Max # "huge" form fields }

procedure closeApp( app : TApplication );
procedure Register;

implementation

{$IFDEF WIN32}
{$R CGI32.RES}
{$ELSE}
```

```
{$R CGI.DCR}
{$ENDIF}

constructor TCGIEnvData.Create(AOwner: TComponent);
begin

 inherited Create(AOwner);

  fStdOut := -99;
  fAddress := '';

 { CGI section }
  fCGICGIVersion := '';
  fCGIRequestProtocol := '';
  fCGIRequestMethod := '';
  fCGIExecutablePath := '';
  fCGILogicalPath := '';
  fCGIPhysicalPath := '';
  fCGIQueryString := '';
  fCGIContentType := '';
  fCGIContentLength := -1;   { init to -1 }
  fCGIServerSoftware := '';
  fCGIServerName := '';
  fCGIServerPort := '';
  fCGIServerAdmin := '';
  fCGIReferer := '';
  fCGIFrom := '';
  fCGIRemoteHost := '';
  fCGIRemoteAddress := '';
  fCGIAuthenticatedUsername := '';
  fCGIAuthenticatedPassword := '';
  fCGIAuthenticationMethod  := '';
  fCGIAuthenticationRealm   := '';

 { System section }
  fSystemGMTOffset := 0;
  fSystemOutputFile := '';
  fSystemContentFile := '';
  fSystemDebugMode := '';

  { Please realize that I thought there might be different
    components for different web servers, and the correct one
    would be linked in.  That whole strategy was abandoned. }
  fServerComponent := nil;
  if NOT (csDesigning in ComponentState) then
    fServerComponent := TWebsite.create( self );
end;

destructor TCGIEnvData.Destroy;
begin
   if fStdOut > 0 then
     closeStdOut;
   if NOT (csDesigning in ComponentState) then
    fServerComponent.free;
   inherited Destroy;
end;
```

```
{ ****************************************************************
      get CGI information from variables from INI file
 **************************************************************** }

function  TCGIEnvData.getCGICGIVersion : pString;
begin
 result := addr( fCGICGIVersion );
end;

procedure TCGIEnvData.getCGIItem( p : pString; key : string; okEmpty : boolean );
var
 x : TWebsite;
begin
 x := TWebsite( fServerComponent );
 x.CGIData( p, key, okEmpty );
end;

function  TCGIEnvData.getCGIRequestProtocol : pstring ;
begin
 getCGIitem( addr( fCGIRequestProtocol ), 'Request Protocol', TRUE );
 result := addr( fCGIRequestProtocol );
end;

function  TCGIEnvData.getCGIRequestMethod : pString;
begin
 result := addr( fCGIRequestMethod );
end;

function  TCGIEnvData.getCGIExecutablePath : pString;
begin
 result := addr( fCGIExecutablePath );
end;

function  TCGIEnvData.getCGILogicalPath : pstring ;
begin
 getCGIItem( addr( fCGILogicalPath ), 'Logical Path', FALSE );
 result := addr( fCGILogicalPath );
end;

function  TCGIEnvData.getCGIPhysicalPath : pString ;
begin
 getCGIItem( addr( fCGIPhysicalPath ), 'Physical Path', FALSE );
 result := addr( fCGIPhysicalPath );
end;

function  TCGIEnvData.getCGIQueryString : pString;
begin
 { it's because of QueryString being blank sometimes that the
   okEmpty parameter was added throughout. }
 getCGIItem( addr( fCGIQueryString ), 'Query String', TRUE );
 result := addr( fCGIQueryString );
end;

function  TCGIEnvData.getCGIContentType : pString;
begin
```

```
 getCGIItem( addr( fCGIContentType ), 'Content Type', FALSE );
 result := addr( fCGIContentType );
end;

function  TCGIEnvData.getCGIContentLength : longInt;
var
 x : TWebSite;
begin
 if fCGIContentLength <> -1 then begin
  { we've already loaded the information }
  result := fCGIContentLength;
  exit;
  end;
 x := TWebsite( fServerComponent );
 fCGIContentLength := x.fIniFile.readInteger( 'CGI', 'Content Length', 0 );
 result := fCGIContentLength;
end;

function  TCGIEnvData.getCGIServerSoftware : pString;
begin
 result := addr( fCGIServerSoftware );
end;

function  TCGIEnvData.getCGIServerName : pstring ;
begin
 getCGIItem( addr( fCGIServerName ), 'Server Name', FALSE );
 result := addr( fCGIServerName );
end;

function  TCGIEnvData.getCGIServerPort : pstring ;
begin
 getCGIItem( addr( fCGIServerPort ), 'Server Name', FALSE );
 result := addr( fCGIServerPort );
end;

function  TCGIEnvData.getCGIServerAdmin : pString;
begin
 result := addr( fCGIServerAdmin );
end;

function  TCGIEnvData.getCGIReferer : pstring ;
var
 x : TWebSite;

begin
 getCGIItem( addr( fCGIReferer ), 'Referer', FALSE );
 if fCGIReferer = cginotfound then begin
  x := TWebsite( fServerComponent );
  fCGIReferer := x.fIniFile.readString( 'Extra Headers', 'Referer', cginotfound );
  end;
 result := addr( fCGIReferer );
end;

function  TCGIEnvData.getCGIFrom : pstring ;
begin
 getCGIItem( addr( fCGIFrom ), 'From', FALSE );
```

```
 result := addr( fCGIFrom );
end;

function  TCGIEnvData.getCGIRemoteHost : pstring ;
begin
 getCGIItem( addr( fCGIRemoteHost ), 'Remote Host', FALSE );
 result := addr( fCGIRemoteHost );
end;

function  TCGIEnvData.getCGIRemoteAddress : pstring ;
begin
 getCGIItem( addr( fCGIRemoteAddress ), 'Remote Address', FALSE );
 result := addr( fCGIRemoteAddress );
end;

function  TCGIEnvData.getCGIAuthenticatedUsername : pstring ;
begin
 getCGIItem( addr( fCGIAuthenticatedUsername ), 'Authenticated Username', TRUE );
 result := addr( fCGIAuthenticatedUsername );
end;

function  TCGIEnvData.getCGIAuthenticatedPassword : pstring ;
begin
 getCGIItem( addr( fCGIAuthenticatedPassword ), 'Authenticated Password', TRUE );
 result := addr( fCGIAuthenticatedPassword );
end;

function  TCGIEnvData.getCGIAuthenticationMethod : pstring ;
begin
 getCGIItem( addr( fCGIAuthenticationMethod ), 'Authentication Method', TRUE );
 result := addr( fCGIAuthenticationMethod );
end;

function  TCGIEnvData.getCGIAuthenticationRealm : pstring ;
begin
 getCGIItem( addr( fCGIAuthenticationRealm ), 'Authentication Realm', TRUE );
 result := addr( fCGIAuthenticationRealm );
end;

function  TCGIEnvData.getSectionValues( sectionName : string; strings : TStringList ) : boolean;
var
 x : TWebsite;
begin
 strings.clear;
 x := TWebsite( fServerComponent );
 x.fIniFile.readSectionValues( sectionName, strings );
 result := (strings.count > 0);
end;


{ ****************************************************************
      get SYSTEM information from variables from INI file
 **************************************************************** }

function  TCGIEnvData.getSystemOutputFile : pString;
begin
```

```
 result := addr( fSystemOutputFile );
end;

function  TCGIEnvData.getSystemContentFile : pstring ;
var
  x : TWebSite;
begin
 if fSystemContentFile = '' then begin
   x := TWebsite( fServerComponent );
   fSystemContentFile := x.fIniFile.readString( 'System', 'Content File', cginotfound );
   end;
 result := addr( fSystemContentFile );
end;

function  TCGIEnvData.getSystemDebugMode : pstring ;
var
  x : TWebSite;
begin
 if fSystemDebugMode = '' then
 begin
   case fServerType of
   webSite :
     begin
       x := TWebsite( fServerComponent );
       fSystemDebugMode := x.fIniFile.readString( 'System', 'Debug Mode', cginotfound );
     end;
   else
     raise exception.create( 'Can not get Debug Mode; invalid web server type' );
   end;
 end;
 result := addr( fSystemDebugMode );
end;

{ Get the value of a "small" form field given the key
  Signals an error if field does not exist }
function TCGIEnvData.getSmallField( key : string ) : string;
var
  x : TWebsite;
  FileName: string;
  i,FileHandle:  integer;
  read:        byte;
  buffer:array[0..255] of char;
  r1 : string;
begin
 x := TWebsite( fServerComponent );
 result := x.getSmallField( key );
 {************* code added to handle long or control chars **********FWT*}
 if result = cginotfound then  begin
    result := x.fIniFile.readString( 'Form External', key, cginotfound );
    if result = cginotfound then
       exit;
    i := pos( ' ', result );
    FileName := copy( result, 0, i - 1 );
    i := strToInt( copy( result, i, 10 ) ) ;
    read := 255;
    FileHandle := fileOpen( FileName, fmOpenRead );
```

```
    if FileHandle > 0 then begin
      fileRead( FileHandle, buffer[0], read );
      fileClose( FileHandle );
      buffer[read] := #0;                {mark the ending}
      if i > read then begin             {indicate truncation}
       buffer[254] := '*';
       buffer[255] := '$';
       end
      else begin
       result := copy(strpas(buffer), 1, i);  {...'i' contains the correct string length...}
       end
      end
     else
      result := cginotfound;
    end;

  {********************* end of code added for long or control chars***FWT*}
end;

{************************** routine added - start of change **************FWT*}
{ Get the values of a "small" multiple selection form field given the key
  Signals an error if field does not exist }
function TCGIEnvData.getSmallMultiField( key : string ) : Tstringlist;
var
  x : TWebsite;
begin
  x := TWebsite( fServerComponent );
  result := x.getSmallMultiField( key );
end;
{************************** routine added - end of change ***************FWT*}

function TCGIEnvData.getExternalField( key : string; var externFilename : string; dest : PChar ) :
boolean;
var
  x : TWebsite;
begin
  x := TWebsite( fServerComponent );
  result := x.getExternalField( key, externFilename, dest );
end;

function TCGIEnvData.getTextArea( key : string; dest : TStringList ) : boolean;
var
  x : TWebsite;
begin
  x := TWebsite( fServerComponent );
  result := x.getTextArea( key, dest );
end;

{ ************************************************************}

function TCGIEnvData.createStdout : boolean ;
begin
  { create output file and save pointer to it }
  {// ssh}
  if fCGIServerSoftware = 'Microsoft-Internet-Information-Server/1.0' then
     fStdout := fileOpen( fSystemOutputFile, fmOpenWrite or fmShareDenyNone )
```

```
  else
    fStdout := fileCreate( fSystemOutputFile );
  if fStdOut < 0 then begin
   raise exception.create( 'Error code [' + intToStr( fStdOut ) +
    '] when creating file (' + fSystemOutputFile + ')' );
  end;
  result := TRUE;
end;


function TCGIEnvData.send( s :.string ) : boolean ;
begin

 result := sendString( s, TRUE );

end;

function TCGIEnvData.sendAuthRequest : boolean;
begin

   closeStdout;
   createStdout;

   result := send( 'HTTP/1.0 401 Unauthorized' );

   closeStdout;

end;

function TCGIEnvData.sendString( s : string; appendNewline : boolean ) : boolean;
const
 newLine : string[4] = #13#10;   {what's the minimum size here? 2? 3? 4? }
var
 s2 : string;
 count : longInt;
begin

 if fStdOut < 0 then
   if NOT createStdout then
    raise exception.create( 'Can not create stdout' );

 if appendNewline then
  s2 := s + newLine
 else
  s2 := s; { will performance suffer? should there be a separate routine here? }

 count := length( s2 );

 { since the first byte of s2 contains the length, we shouldn't write
 that out. Start instead with the next byte, which is s2[1]. }
 result := ( fileWrite( fStdout, s2[1], count ) = count );

end;

procedure TCGIEnvData.closeStdout;
```

```
begin
   fileClose( fStdout );
end;

{ SendNoOp() - Tell browser to do nothing.
 Most browsers will do nothing. Netscape 1.0N leaves hourglass
 cursor until the mouse is waved around. Enhanced Mosaic 2.0
 oputs up an alert saying "URL leads nowhere". Your results may
 vary...}
procedure TCGIEnvData.sendNoOp;
begin
   Send ('HTTP/1.0 204 No Response');
   Send ('Server: ' + fCGIServerSoftware );
   Send ('');
end;

{ WebDate - Return an HTTP/1.0 compliant date/time string

 Inputs:  dt = Local time as TDateTime (e.g., returned by Now)
 Returns:  Properly formatted HTTP/1.0 date/time in GMT }

function TCGIEnvData.webDate (dt : TDateTime ) : String ;
begin
   WebDate := FormatDateTime('ddd dd mmm yyyy hh:mm:ss "GMT"',
          dt - fSystemGMTOffset );
end;

procedure TCGIEnvData.bounceToLocation( goHere : string );
begin
   closeStdout;
   createStdout;
   Send ('LOCATION: ' + goHere );
   Send ('');
   closeStdout;
end;

function  TCGIEnvData.sendAddress : boolean;
begin
 if fAddress = '' then
   result := FALSE
 else
   result := send( '<ADDRESS>' + fAddress + '</ADDRESS>' );
end;

function  TCGIEnvData.sendHR : boolean;
begin
 result := send( '<HR>' );
end;

function  TCGIEnvData.sendHdr( hdrLevel : char; hdrText : string ) : boolean;
begin
 if ( hdrLevel < '1' ) OR ( hdrLevel > '6' ) then
 begin
  sendComment( 'hdrLevel should be between 1 and 6.  Ref: ' + hdrText );
  result := FALSE;
 end
```

```
 else
  result := send( '<H' + hdrLevel + '>' + hdrText + '</H' + hdrLevel + '>' );
end;

function  TCGIEnvData.sendHREF( imageFilename : string;
                                imageAttrib : string;
                                visiblePhrase : string;
                                linkedURL : string ) : boolean;
begin

 if linkedURL = '' then begin
   result := FALSE;
   exit;
  end;

{ Here is a sample of what this can result in:
<A HREF="http://www.sonic.net/~ann/htmlsmnr.html">
<IMG SRC="/html/ann/infobahn.gif"
>InfoBahn Construction Workshop</A>!
}
 send( '<A HREF="' + linkedURL + '">' );
 if imageFilename <> '' then
  send( '<IMG ' + imageAttrib + ' SRC="' + imageFilename + '">' );

 result := send( visiblePhrase + '</A>' );
end;

function  TCGIEnvData.sendIMG( imageFilename : string; imageAttrib : string ) : boolean;
begin
 result := send( '<IMG ' + imageAttrib + ' SRC="' + imageFilename + '">' );
end;

function  TCGIEnvData.sendPrologue : boolean;
begin
 try
  send( 'HTTP/1.0 200 OK' );
  send( 'SERVER: ' + fCGIServerSoftware );
  send( 'DATE: ' + webDate( now ) );
  send( 'Content-type: text/html' );
  send( '' );          { required blank line }
  result := TRUE;
 except
  result := FALSE;
 end;
end;

function  TCGIEnvData.sendTitle( title : string ) : boolean;
begin
 result := send( '<TITLE>' + title + '</TITLE>' );
end;

function  TCGIEnvData.sendBackground( imageFilename : string ) : boolean;
begin
 {<body background="bkground.gif">}
 result := send( '<BODY BACKGROUND="' + imageFilename + '"' );
```

```
end;

procedure TCGIEnvData.sendComment( s : string );
begin
  send( '<!-- ' + s + ' -->' );
end;

procedure TCGIEnvData.sendMailto( emailAddress : string );
begin
  send( '<A HREF="mailto:' + emailAddress + '">' + emailAddress + '</A>' );
end;

procedure TCGIEnvData.cgiErrorHandler( sender: TObject; e : Exception );
begin
    if fStdout = -99 then
      { haven't even gotten as far as opening stdout at all yet! }
      { this would be a bad time to count on writing to that file !! }
     closeApp( application );
try
   createStdout;
   Send ('HTTP/1.0 500 Internal Error');
   Send ('SERVER: ' + fCGIServerSoftware);
   Send ('DATE: ' + WebDate(Now) );
   Send ('Content-type: text/html' );
   Send ('');
   Send ('<HTML><HEAD>');
   Send ('<TITLE>Error in ' + fCGIExecutablePath + '</TITLE>' );
   Send ('</HEAD><BODY>');
   SendHdr( '2', 'Error in ' + fCGIExecutablePath );
   Send ('An internal error has occurred in this program: ' + fCGIExecutablePath + '.');
   Send ('<PRE>' + e.message + '</PRE>');
   Send ('<I>Please</I> note what you were doing when this problem occurred, ');
   Send ('so we can identify and correct it. Write down the Web page you were using, ');
   Send ('any data you may have entered into a form or search box, the' );
   Send ('date and time listed below, and ');
   Send ('anything else that may help us duplicate the problem. Then contact the ');
   Send ('administrator of this service: ');
   Send ('<A HREF="mailto:' + fCGIServerAdmin + '">' + fCGIServerAdmin + '</A> ' );
   SendHR;
   send( 'Generated on: ' + webDate( now ) );
   Send ('</BODY></HTML>');
   fileClose( fStdOut );
   fStdOut := -99;
finally
  { the bottom line! }
   closeApp( application );
end;

end;

procedure TCGIEnvData.setIniFilename( value : string );
var
  x : TWebSite;
begin
  fINIFilename := value;
  if NOT ( csDesigning in componentState ) then begin
```

```
    x := TWebSite( fServerComponent );
    x.initData;
    end;
  end;

  function TCGIEnvData.swapChar( s : string; fromChar : char; toChar : char ) : string;
  var
    i : shortint;
  begin
    for i := 1 to length( s ) do
      if s[i] = fromChar then
        s[i] := toChar;
    result := s;
  end;

  {*****************************************************************}
  {*****************************************************************}

  constructor TWebsite.create(AOwner: TComponent);
  begin
    if AOwner = nil then
      raise exception.create( 'Tried to create TWebsite object with nil owner.' );

    inherited Create(AOwner);

    fIniFile := nil;
    fServerType := WebSite;

    { this works only if AOwner is a valid pointer, which it should be
    since we're only created from within a CGIEnvData component }
    fCGI := TCGIEnvData(AOwner);  { connect back to CGIEnvData }
  end;

  procedure TWebSite.initData;
  begin
    if fCGI.WebSiteINIFilename = '' then
      raise exception.create( 'WebSiteINIFilename is blank' );

    try
      { create pointer to INI file }
      fIniFile := tInifile.create( fCGI.WebSiteIniFilename );
    except
      raise exception.create( 'Can not create tIniFile object' );
    end;

    with fCGI do begin
      { [CGI]            <== The standard CGI variables }
      fCGICGIVersion     := readWebSiteCGIString( 'CGI Version', FALSE );
      fCGIRequestMethod  := readWebSiteCGIString( 'Request Method', FALSE );
      { Request Protocol handled elsewhere }
      {//ssh}
      fCGIServerSoftware := readWebSiteCGIString( 'Server Software', FALSE );
      if fCGIServerSoftware = 'Microsoft-Internet-Information-Server/1.0' then
        fCGIExecutablePath := readWebSiteCGIString( 'Referer', FALSE )
      else
        fCGIExecutablePath := readWebSiteCGIString( 'Executable Path', FALSE );
```

```
    fCGIServerAdmin    := readWebSiteCGIString( 'Server Admin', TRUE );
    end;

  with fIniFile do begin
    { [System]          <== Windows interface specifics }
    { in visual basic: CGI_GMTOffset = CVDate(CDbl(buf) / 86400#)' Timeserial offset }
    fCGI.fSystemGMToffset := ( readInteger( 'System', 'GMT Offset', 0 ) / 86400 );  { fixed 6/12/95 aml
}
    fCGI.fSystemOutputFile  := readString( 'System', 'Output File', 'ann_x.out' );
    fCGI.fSystemContentFile := readString( 'System', 'Content File', '' );
  end;
end;

destructor TWebsite.Destroy;
begin
    fIniFile.free;
    inherited Destroy;
end;

function TWebsite.readWebSiteCGIString( key : string; okEmpty : boolean ) : string;
begin
  result := fINIfile.readString( 'CGI', key, cginotfound );
{ notfound is not always bad, e.g. user might not be authenticated first time around }
  if result = cginotfound then
    if NOT okEmpty then
      fCGI.sendComment( '[CGI] ' + key + ' key not found in WebSite INI file' );
end;

procedure TWebsite.CGIData( p : pString; key : string; okEmpty : boolean );
begin
    if p^ = '' then
      p^ := readWebSiteCGIString( key, okEmpty );
end;

{ returns KEY NOT FOUND and logs sendComment if that happens; otherwise full text }
function TWebsite.getSmallField( key : string ) : string;
begin
  with fIniFile do
    result := readString( 'Form Literal', key, cginotfound );

  if result = cginotfound then
    fCGI.sendComment( 'Field ' + key + ' is not in [Form Literal] section of WebSite .ini file.' );
end;

{ returns KEY NOT FOUND and logs sendComment if that happens; otherwise full text }
function TWebsite.getSmallMultiField( key : string ) : Tstringlist;
var
  varval, varname: string;
begin
  result := TStringList.create;
  varname := key;
  varval  := 'start';
  while varval <> cginotfound do begin
    with fIniFile do
      varval := readString( 'Form Literal', varname, cginotfound );
```

```
      if varval <> cginotfound then begin
        result.add( varval );
        varname := key+'_'+IntToStr(result.count);
        end;
    end;
end;

{ if key not found, then 3 things happen.  1. returns false
  2. externFilename set to "   3. error comment sent out }
function TWebsite.getExternalField( key : string;
                          var externFilename : string;
                          dest : PChar ) : boolean;
var
  info : string;
  buffer : string;
  x : byte;
  dataSize : integer;
  fileHandle : integer;

begin

{ [Form External]  notes written by Bob Denny and included in cgi.bas
  If the decoded value string is more than 254 characters long,
  or if the decoded value string contains any control characters,
  the server puts the decoded value into an external tempfile and
  lists the field in this section as:
    key=<pathname> <length>
  where <pathname> is the path and name of the tempfile containing
  the decoded value string, and <length> is the length in bytes
  of the decoded value string.

  Data larger than 65,536 bytes goes to [Form Huge] section. }

    with fIniFile do
      info := readString( 'Form External', key, cginotfound );

    if info = cginotfound then
    begin
      result := FALSE;
      externFilename := '';
      fCGI.sendComment( 'Field ' + key + ' is not in [Form External] section of WebSite .ini file.' );
      exit;
    end;

    x := pos( ' ', info );
    externFilename := copy( info, 0, x - 1 );

    dataSize := strToInt( copy( info, x, 10 ) ) ;
    dest := strAlloc( dataSize + 1 );

    { !!! need more error checking in this routine }
    fileHandle := fileOpen( externFilename, fmOpenRead );
    fileRead( fileHandle, dest, dataSize );
    fileClose( fileHandle );
    result := TRUE;
end;
```

```
function TWebsite.getTextArea( key : string; dest : TStringList ) : boolean;
var
 info : string;
 buffer : string;
 x : byte;
 dataSize : integer;
 f : textFile;
 externfilename : string;

begin

  result := TRUE;

  if dest = nil then
  begin
   dest := TStringList.create;
   fCGI.sendComment( 'TstringList was nil in call to getExternalStrList. ' +
                     'You should be using TStringList.create and .free yourself.' );
  end;

  dest.clear;

  { first see whether it's there as a one-liner }
  buffer := getSmallField( key );
  if buffer <> cginotfound then
  begin
   dest.add( buffer );    { all done }
   exit;
  end;


  with fIniFile do
   info := readString( 'Form External', key, cginotfound );

  if info = cginotfound then
  begin
   result := FALSE;
   fCGI.sendComment( 'Field ' + key + ' is not in [Form External] section of WebSite .ini file.' );
   exit;
  end;

  x := pos( ' ', info );
  externFilename := copy( info, 0, x - 1 );

  dataSize := strToInt( copy( info, x, 10 ) ) ;

  { !!! need more error checking in this routine }
  assignFile( f, externFilename );
  reset(f);
  while NOT eof(f) do
  begin
   readLn( f, buffer );
   dest.add( buffer );
  end;
  closeFile( f );
```

```
    result := TRUE;

end;

procedure closeApp( app : TApplication );
begin
  {Thanks to Charlie Calvert for the postMessage syntax. }
  { FYI: app.close; doesn't work and halt(1) is bad because resources aren't freed. }
  postMessage( app.Handle, wm_Close, 0, 0);
end;

{*********************************************************************}
{*********************************************************************}

procedure Register;
begin
  RegisterComponents('CGI', [TCGIEnvData]);
end;

end.
```

ภาคผนวก ฉ
ซอร์สโค้ดตัวอย่างระบบงานข้อมูลนักศึกษา

ตัวอย่าง CGI แอพพลิเคชั่นที่สร้างขึ้นด้วยภาษาวิชวลเบสิก แบ่งออกเป็น 2 โมดูล คือโมดูลสำหรับสับรูทีน Sub CGI_Main() และอีกโมดูลหนึ่ง เป็นโมดูลสำหรับสับรูทีนที่ใช้งานทั่ว ๆ ไป

```
Attribute VB_Name = "CGI_Main_Here"
Option Explicit

Dim sSelector As String
Dim CGI_File As String
Dim Database_File As String
' This subroutine will be called by CGI_Framework written by Robert B.Denny
' This program is Student 4D Information System.
' Work with database on DORAEMON SQL Server
' Connect to SQL Server by ODBC
' DSN : Data Source Name = LaemSQL
' Database name = LaemDB
Sub CGI_Main()

'sSelector is some word after CGI File name in <A HREF...> tag
'for example you call <a href="thisfile.exe/sSelector> ...
   sSelector = UCase$(Mid$(CGI_LogicalPath, 2))    ' Remove leading "/"

'Define the name of the file frequently use in this program.
   CGI_File = "/cgi-win/laem-sample/vb/vb8/vb1.exe"
   Database_File = "ODBC;DSN=LaemSQL;UID=s6014406;PWD=;DATABASE=Laem;"
'    Database_File = "d:/applic~1/database/laem.mdb"

'Consiter for RequestMethod GET or POST. Others is denied.
   Select Case UCase$(CGI_RequestMethod)
      Case "GET":
         DoGet
      Case "POST":
         DoPost
      Case Else:
         Send ("<H2>Cannot do """ & CGI_RequestMethod & """ method</H2>")
   End Select

End Sub


Sub DoGet()
   SendHeader ("4D Student Information")
   Send ("<BODY>")
   Send ("<HEAD><H1> 4D students Information System. </H1></HEAD>")
```

```
    Send ("<HR>")
    Send ("Please click button which you want to do.")

'VIEW
    Send ("<form action=""" & CGI_File & "/view1"" method=post>")
    Send ("<input type=submit value=""VIEW""> View all data in 2 steps.")
    Send ("</form>")

'UPDATE
    Send ("<FORM ACTION=""" & CGI_File & "/update1"" METHOD=POST>")
    Send ("<INPUT TYPE=SUBMIT VALUE=""UPDATE""> Update data in 4 steps.")
    Send ("</FORM>")

'SQL
    Send ("<form action=""" & CGI_File & "/sql"" method=post>")
    Send ("<input type=submit value=""SQL""> Enter SQL ")
    Send ("</form>")
    SendFooter
End Sub

Sub DoPost()
' Goto which step (There are 3 step for update)
    Select Case sSelector
        Case "SQL":
            DoSQL
        Case "SQL2":
            DoSQL2
        Case "VIEW1":
            View1
        Case "VIEW2":
            View2
        Case "UPDATE1":
            Update1
        Case "UPDATE2":
            Update2
        Case "UPDATE3":
            Update3
        Case "UPDATE4":
            Update4
        Case Else:
            SendHeader (" error ")
            Send ("<H2> Your Command ? </H2>")
            SendFooter
    End Select

End Sub
Sub DoSQL()
    SendHeader ("Enter SQL")
    Send ("<form action=""" & CGI_File & "/sql2"" method=post>")
    Send ("SQL : <input type=text name=""SQL""> ")
    Send ("<input type=submit>")
    SendFooter

End Sub

Sub DoSQL2()
```

```
   SendHeader ("Result")
   Call QueryResult(Database_File, GetSmallField("SQL"))
   SendFooter
End Sub


' Get student ID. Number
Sub Update1()
   SendHeader ("Step 1")
   Send ("<H1> Step 1: Enter your student ID.</H1>")
   Send ("<HR>")

'enter student ID.
   Send ("<FORM action=""" & CGI_File & "/update2"" method=post>")
   Send ("Student ID : <input type=text name=""ID"" maxlength=8 size=9> <BR>")
   Send ("<input type=submit value=""Next >"" > Go to select the column to be changed")
   Send ("</form>")

'button to go back
   Send ("<form action=""" & CGI_File & """ method=get>")
   Send ("<input type=submit value=""Cancel""> Back to the Main page.")
   Send ("</form>")
   SendFooter
End Sub


' Display the old data and
' select column to be changed
Sub Update2()
   Dim Temp_Database As Database
   Dim Temp_Snapshot As Snapshot
   Dim Temp_listset As Snapshot
   Dim Column_number As Integer
   Dim SQL_Query As String

' Connect to MS-Access database file and select where std_id = xxx
   SQL_Query = "select * from friend where std_id = '" & GetSmallField("ID") & "'"
   'Set Temp_Database = OpenDatabase("", False, False, Database_File)
   Set Temp_Database = OpenDatabase(Database_File)
   Set Temp_Snapshot = Temp_Database.CreateSnapshot(SQL_Query)


' Query return 0 row or not?
   If Temp_Snapshot.RecordCount <> 0 Then
      SendHeader ("Step 2")
      Send ("<h1> Step 2: Select column to be changed.</h1>")
      Send ("<hr>")

'Save information about Attribute on Database Table. in Temp_listset
'for display the old data
      Set Temp_listset = Temp_Snapshot.ListFields()

' Display choice for change and old data in the table format
      Send ("<form action=""" & CGI_File & "/update3"" method=post>")
      Send ("<table boarder=0>")
      Column_number = 0
```

```
    Temp_Snapshot.MoveFirst
    Temp_listset.MoveFirst
    Do While Not Temp_listset.EOF
      Column_number = Column_number + 1
      Send ("<tr>")
      Send ("<td>")

'check radio button at the last attribute
      If Column_number = Temp_listset.RecordCount Then
        Send ("<input type=radio checked name=""choice"" value=""" & Temp_listset("name") &
""">" & Temp_listset("name"))
      Else
        Send ("<input type=radio name=""choice"" value=""" & Temp_listset("name") & """> " &
Temp_listset("name"))
      End If

      Send ("</td>")
      Send ("<td><B> Old value : </b></td>")
      Send ("<td> " & Temp_Snapshot(Temp_listset("name")).value & "</td>")
      Send ("</tr>")
      Temp_listset.MoveNext
    Loop
    Send ("</table>")
    Send ("<input type=hidden name=""ID"" value=""" & GetSmallField("ID") & """>")
    Send ("<input type=submit value=""Next >""> Go to enter new data value.")
    Send ("</form>")

'enter new student ID.
    Send ("<form action=""" & CGI_File & "/update1"" method=post>")
    Send ("<input type=submit value=""< Back""> Enter new student ID.")
    Send ("</form>")

'button to go back
    Send ("<form action=""" & CGI_File & """ method=get>")
    Send ("<input type=submit value=""Cancel""> Back to the Main page")
    Send ("</form>")
    SendFooter
  Else

'If there is no row match with the input std_id , do here!
    SendHeader ("Step 2: No Match")
    Send ("<h1> There is no data for " & GetSmallField("ID") & "</h1>")
    Send ("<form action=""" & CGI_File & "/update1"" method=post>")
    Send ("<input type=submit value=""< Back""> Enter new student ID.")
    Send ("</form>")
    Send ("<form action=""" & CGI_File & """ method=get>")
    Send ("<input type=submit value=""Cancel""> Back to the Main.")
    Send ("</form>")
    SendFooter
  End If

  Temp_Database.Close

End Sub
```

```
' Enter the new value
Sub Update3()
   Dim Temp_Database As Database
   Dim Temp_Snapshot As Snapshot
   Dim Temp_listset As Snapshot
   Dim Number_of_Row As Integer
   Dim SQL_Query As String

   SendHeader ("Step3 ")
   Send ("<h1> Step 3: Enter the new value </h1>")
   Send ("<hr>")

   SQL_Query = "select " & GetSmallField("choice") & " from friend where std_id = '" &
GetSmallField("ID") & "'"
   'Set Temp_Database = OpenDatabase("", False, False, Database_File)
   Set Temp_Database = OpenDatabase(Database_File)
   Set Temp_Snapshot = Temp_Database.CreateSnapshot(SQL_Query)

   Send ("<h2> Update data for " & GetSmallField("choice") & " of " & GetSmallField("ID") &
"</h2>")

'Display old data value
   Send ("<b> Old value : </b>")
   Set Temp_listset = Temp_Snapshot.ListFields()
   Temp_listset.MoveFirst
   Do While Not Temp_listset.EOF
      Send (Temp_Snapshot(Temp_listset("name")).value & "<br>")
      Temp_listset.MoveNext
   Loop

'Form for input new data value
   Send ("<form action=""" & CGI_File & "/update4"" method=post>")
   Send ("<b> New value : </b><input type=text name=""new_value"" size=100>")
   Send ("<br><input type=submit value=""Next >""> Execute update operation.")
   Send ("<input type=hidden name=""choice"" value=""" & GetSmallField("choice") & """>")
   Send ("<input type=hidden name=""ID"" value=""" & GetSmallField("ID") & """>")
   Send ("</form>")

'change other column
   Send ("<form action=""" & CGI_File & "/update2"" method=post>")
   Send ("<input type=hidden name=""ID"" value=""" & GetSmallField("ID") & """>")
   Send ("<input type=submit value=""< Back""> Change other column.")
   Send ("</form>")

'button to go back
   Send ("<form action=""" & CGI_File & """ method=get>")
   Send ("<input type=submit value=""Cancel""> Back to the Main page.")
   Send ("</form>")
   SendFooter

   Temp_Database.Close

End Sub


'Execute Update operation
```

```
Sub Update4()
   Dim Temp_Database As Database
   Dim Temp_Snapshot As Snapshot
   Dim Temp_listset As Snapshot
   Dim Number_of_Row As Integer
   Dim SQL_Query As String
   Dim Row_Affected As Long

'starting HTML Document
   SendHeader ("Step 4")
   Send ("<h1> Step 4: Update Result </h1>")
   Send ("<hr>")

'Makeup UPDATE SQL Statement
   SQL_Query = "update friend set " & GetSmallField("choice") & " = "
   If GetSmallField("choice") = "grade" Then
      SQL_Query = SQL_Query & GetSmallField("new_value")
   Else
      SQL_Query = SQL_Query & "'" & GetSmallField("new_value") & "'"
   End If
   SQL_Query = SQL_Query & " where std_id = '" & GetSmallField("ID") & "'"

'Execute UPDATE Operation
   'Set Temp_Database = OpenDatabase("", False, False, Database_File)
   Set Temp_Database = OpenDatabase(Database_File)
   Temp_Database.Execute (SQL_Query)
   Row_Affected = Temp_Database.RecordsAffected

'UPDATE operation affect 0 row or not?
   If Row_Affected <> 0 Then

'Display new data for std_id = xxx
      SQL_Query = "select * from friend where std_id = '" & GetSmallField("ID") & "'"
      Set Temp_Snapshot = Temp_Database.CreateSnapshot(SQL_Query)
      Set Temp_listset = Temp_Snapshot.ListFields()

      Send ("<table boarder=0>")
      Temp_Snapshot.MoveFirst
      Temp_listset.MoveFirst
      Do While Not Temp_listset.EOF
         Send ("<tr>")
         Send ("<td><b> " & Temp_listset("name") & " : </b></td>")
         Send ("<td>" & Temp_Snapshot(Temp_listset("name")).value & "</td>")
         Send ("</tr>")
         Temp_listset.MoveNext
      Loop
      Send ("</table>")

   Else
'if no row affected , will do here
      Send ("No record affected this operation! ")
   End If

'enter new value again
   Send ("<form action=""" & CGI_File & "/update3"" method=post>")
   Send ("<input type=hidden name=""ID"" value=""" & GetSmallField("ID") & """>")
```

```
    Send ("<input type=hidden name=""choice"" value=""" & GetSmallField("choice") & """>")
    Send ("<input type=submit value=""< Back""> Enter new <b> " & GetSmallField("choice") & "
</b> value again.")
    Send ("</form>")

'finish operation , back to the main page
    Send ("<form action=""" & CGI_File & """ method=get>")
    Send ("<input type=submit value=""Finish""> Finish operation , Back to the Main page.")
    Send ("</form>")
    SendFooter

    Temp_Database.Close
End Sub


' select column to view
Sub View1()
    Dim Temp_Database As Database
    Dim Temp_Snapshot As Snapshot
    Dim Temp_listset As Snapshot
    Dim Number_of_Row As Integer
    Dim SQL_Query As String

    SendHeader ("Step 1")
    Send ("<h1> Step 1: Select column to be displayed </h1>")
    Send ("<hr>")

'makeup choice of column to be displayed
    Send ("<form action=""" & CGI_File & "/view2"" method=post>")
    SQL_Query = "select * from friend"
    'Set Temp_Database = OpenDatabase("", False, False, Database_File)
    Set Temp_Database = OpenDatabase(Database_File)
    Set Temp_Snapshot = Temp_Database.CreateSnapshot(SQL_Query)
    Set Temp_listset = Temp_Snapshot.ListFields()
    Temp_listset.MoveFirst

    Do While Not Temp_listset.EOF
       Send ("<input type=checkbox name=""choice"" value=""" & Temp_listset("name") & """> <b> "
& Temp_listset("name") & " </b><br>")
       Temp_listset.MoveNext
    Loop

    Send ("<input type=submit value=""Next >""> Display the result. ")
    Send ("</form>")

    Send ("<form action=""" & CGI_File & """ method=get>")
    Send ("<input type=submit value=""Cancel""> Back to the Main page. ")
    Send ("</form>")
    SendFooter

    Temp_Database.Close
End Sub

Sub View2()
    Dim Temp_Database As Database
    Dim Temp_Snapshot As Snapshot
```

```
   Dim Temp_listset As Snapshot
   Dim Number_of_Row As Integer
   Dim SQL_Query As String

   Dim i As Integer
   Dim col_list As String

'gather all column to be displayed
   i = 0
   On Error GoTo TOPPING_DONE
   Do
      If i = 0 Then
         col_list = GetSmallField("choice")
      Else
         col_list = col_list & "," & GetSmallField("choice_" & i)
      End If
      i = i + 1
   Loop
TOPPING_DONE:
   Resume TOPPING_DONE1
TOPPING_DONE1:
   On Error GoTo 0          ' Revert error handling

   If col_list <> "" Then
'column list is not 0 , then do here
      SendHeader ("Step 2")
      Send ("<h1> Step 2: View Result </h1>")
      Send ("<hr>")

      SQL_Query = "select " & col_list & " from friend"

'display SQL statement
      Send ("<b> Your SQL is </b> " & SQL_Query)

'display result in HTML table format
      Call QueryResult(Database_File, SQL_Query)

' enter new column list
      Send ("<form action=""" & CGI_File & "/view1"" method=post>")
      Send ("<input type=submit value=""< Back""> Select new column list. ")
      Send ("</form>")

' finish operation
      Send ("<form action=""" & CGI_File & """ method=get>")
      Send ("<input type=submit value=""Finish""> Finish operation , Back to the Main page.")
      Send ("</form>")
   Else
'There is no column list selected , so report error
      SendHeader ("Step 2")
      Send ("<h1> Step 2: ERROR! No column be selected. </h1>")
      Send ("<hr>")

'enter new column list
      Send ("<form action=""" & CGI_File & "/view1"" method=post>")
      Send ("<input type=submit value=""< Back""> Select new column list. ")
      Send ("</form>")
```

```
'Cancel operation
      Send ("<form action=""" & CGI_File & """ method=get>")
      Send ("<input type=submit value=""Cancel""> Back to the Main page.")
      Send ("</form>")
   End If

   SendFooter
End Sub
```

```
Attribute VB_Name = "Laem_Framework"
Option Explicit

'Execute SQL such as Update , Insert , Delete
'Vitaya Vinyunavan Fri. 20 Dec. 96

Sub Action_Query_Result(database_name, Sql_Action As String)
   Dim Temp_Database As Database

   Set Temp_Database = OpenDatabase(database_name)
   Temp_Database.Execute (Sql_Action)

   Temp_Database.Close
End Sub

'This procedure will be execute when compile but
'not effect in CGI operation.
'Vitaya Vinyunavan Fri. 20 Dec. 96

Sub inter_main()
   MsgBox ("Congratulation! Compile Successful!")
End Sub



' Receive database name and SQL query string
' and generate output data in the HTML Table format
' Vitaya Vinyunavan Fri 20 Dec. 96

Sub QueryResult(database_name As String, SQL_Query As String)
   Dim Temp_Database As Database
   Dim Temp_Snapshot As Snapshot
   Dim Temp_listset As Snapshot
   Dim Number_of_Row As Integer

' Connect database with ODBC should write in this form.
   'Set Temp_Database = OpenDatabase("", False, False, "ODBC; DSN='Laem
Database';UID=';PWD='")
   'Set Temp_Database = OpenDatabase("", False, False, "ODBC; DSN='" & database_name &
"';UID=';PWD='")

' Connect MS Access Database with Database Jet Engine.
   'Set Temp_Database = OpenDatabase("", False, False, database_name)
```

```
    Set Temp_Database = OpenDatabase(database_name)
    Set Temp_Snapshot = Temp_Database.CreateSnapshot(SQL_Query)

'Save information about Attribute on Database Table.
    Set Temp_listset = Temp_Snapshot.ListFields()

' This query return more than 0 row or not?
    If Temp_Snapshot.RecordCount <> 0 Then
       Temp_Snapshot.MoveLast
       Number_of_Row = Temp_Snapshot.RecordCount
    Else
       Number_of_Row = 0
    End If

' Begin HTML Table Tag.
    Send ("<TABLE BORDER>")
    Send ("<TR>")

'Go to First Attribute (column in table) and write all attribute name
    Temp_listset.MoveFirst
    Do While Not Temp_listset.EOF
       Send ("<TD SIZE=" & Str(Temp_listset("size")) & ">")
       Send (Temp_listset("name") & "</TD>")
       Temp_listset.MoveNext
    Loop
    Send ("</TR>")

'If data contain 0 row then exit this loop ,
'else repeat write data in each column of every rows.
    If Number_of_Row <> 0 Then
       Temp_Snapshot.MoveFirst
       Do While Not Temp_Snapshot.EOF
          Send ("<TR>")
          Temp_listset.MoveFirst
          Do While Not Temp_listset.EOF
             Send ("<TD> " & Temp_Snapshot(Temp_listset("name")).value & "</TD>")
             Temp_listset.MoveNext
          Loop
          Send ("</TR>")
          Temp_Snapshot.MoveNext
       Loop
    End If
    Send ("</TABLE>")
    Send ("<P><I> Total " & Str(Number_of_Row) & " record(s)</I>")

    Temp_Database.Close
End Sub



' Make link to ruturn to the referer document
' Vitaya Vinyunavan Tue.24 Dec. 1996
Sub Return_Back_Link()
'    Send ("<P><A HREF=""" & CGI_Referer & """> Back </A>")
End Sub
```

```
' Logo for my group.
' Vitaya Vinyunavan Fri.20 Dec.1996
Sub SendFooter()
   Send ("<HR>")
   Send ("<CENTER>")
   Send ("<FONT SIZE=3 COLOR=RED>")
   Send ("APPLICATION DEVELOPMENT ON WWW. ")
   Send ("</FONT><BR>")
   Send ("Thawatchai Anuponganant 36014187 <BR>")
   Send ("Vitaya Vinyunavan 36014406 <BR>")
   Send ("</CENTER>")
End Sub

' Generate head of HTML Document
' This procedure receive string from caller
' Vitaya Vinyunavan Fri. 20 Dec. 96

Sub SendHeader(Title As String)
   Send ("Content-type: text/html")
   Send ("").
   Send ("<HTML><HEAD><TITLE>" & Title & "</TITLE></HEAD>")

End Sub
```

สำหรับซอร์สโค้ดของ CGI แอพพลิเคชั่นที่เขียนขึ้นด้วย Delphi จะมีการทำงานเหมือนกับ CGI แอพพลิเคชั่นที่เขียนขึ้นด้วยวิชวลเบสิกทุกประการ โดยคำสั่งการทำงานต่าง ๆ จะเขียนในส่วนของ Application.CreateForm

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

```
unit Unit1; {This is 4D Student Information System Program.}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Cgi, DBTables, DB, Cgidb;

const
    CGI_File = '/cgi-win/laem-sample/delphi/delphi6/project1.exe';
```

```
type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   DataSource1: TDataSource;
   Table1: TTable;
   Query1: TQuery;
   Database1: TDatabase;
   procedure FormCreate(Sender: TObject);

{These are my own procedure }
   procedure SendFooter;
   procedure DoGet;
   procedure DoPost;
   procedure view1;
   procedure view2;
   procedure update1;
   procedure update2;
   procedure update3;
   procedure update4;
   procedure sql;
   procedure sql2;

 private
   { Private declarations }

 public
      { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

{ Send my project footer }
procedure Tform1.SendFooter;
begin
    with CGIEnvData1 do
    begin
        sendHR;
        send('<center>');
        send('<font color=red> Application Development on WWW </font> <br>');
        send('Thawatchai Anuponganant 36014187 <br>');
        send('Vitaya Vinyunavan 36014406 <br>');
    end;
end;


{ Procedure View1 , send list of column in table friend with checkbox}
procedure tform1.view1;
var i : integer;
   temp_str : string;
begin
```

```
  with cgienvdata1 do
  begin
    {Send some header}
    send ('<h1> Step 1: Select column to be displayed </h1>');
    Send ('<hr>');

    {Begin of the form with the list of checkbox}
    send ('<form action="' + CGI_File + '/view2" method=post>');

    {loop until all column is display}
    for i := 0 to table1.fieldcount - 1 do
      with table1.fielddefs[i] do
      begin
          temp_str := name;
          send('<input type=checkbox name="choice" ');
          send('value="' + temp_str + '"> ');
          send('<b> ' + temp_str + '</b><br>');
       end;

    send ('<input type=submit value="Next >"> Display the result.');
    send ('</form>');

    {Button for back}
    send('<form action="' + cgi_file + '" method=get>');
    send('<input type=submit value="Cancel"> Back to the main page.');
    send('</form>');

  end;
end;


{display data in table with method "DRAWTABLE" of component "CGIDB"}
procedure tform1.view2;
var i : integer;
   temp_str : string;
   col_list : string;
   col_list_done : boolean;
   query_str : string;

begin
    with cgienvdata1 do
    begin
       col_list_done := false;
       i := 1;

       {if function getsmallfield error it will return this word "AAAKEY NOT FOUND"}
       if getsmallfield('choice') = 'AAAKEY NOT FOUND' then
         col_list_done := true
       else
          col_list := getsmallfield('choice');

       {make up column list}
       repeat
         str(i,temp_str);
         temp_str := 'choice_' + temp_str;
         if getsmallfield(temp_str)='AAAKEY NOT FOUND' Then
```

```
        col_list_done := TRUE
      else
        col_list := col_list + ',' + getsmallfield(temp_str);
      i := i+1;
    until col_list_done or (i=table1.fieldcount);

    {if column list is not null, then make sql query}
    if col_list <> '' then
    begin
        {put sql query statement to the property sql of component query}
        query1.sql.clear;
        query_str := 'select ' + col_list + ' from friend';
        query1.sql.add(query_str);
        query1.active := true;
        datasource1.dataset := query1;

        sendtitle('Step 2');
        send('<h1> Step 2: View Result </h1>');
        send('<hr>');

        cgidb1.drawtable;
        query1.active := false;

        {button 1}
        send('<form action="' + cgi_file + '/VIEW1" method=post>');
        send('<input type=submit value="< Back"> Select new column list.');
        send('</form>');

        {button 2}
        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Finish"> Finish operation, Back to the main page.');
        send('</form>');

    end
    else
    {if column list is null then do here}
    begin
        sendtitle('Step 2');
        send('<h1> Step 2: ERROR! No column be selected. </h1>');
        send('<hr>');

        send('<form action="' + cgi_file + '/VIEW1" method=post>');
        send('<input type=submit value="< Back"> Select new column list.');
        send('</form>');

        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Cancel"> Back to the Main page.');
        send('</form>');
    end

  end;
end;


{procedure update1 , get student id. from user.}
procedure tform1.update1;
```

```
begin
   with cgienvdata1 do
   begin
      sendtitle('Step 1');
      send('<h1> Step 1: Enter your student ID.</h1>');
      send('<hr>');

      send('<form action="' + cgi_file + '/UPDATE2" method=post>');
      send('Student ID : <input type=text name="ID" maxlength=8 size=9><br>');
      send('<input type=submit value="Next >"> Go to select the column to be changed');
      send('</form>');

      send('<form action="' + cgi_file + '" method=get>');
      send('<input type=submit value="Cancel"> Back to the Main page.');
      send('</form>');
   end;
end;

{update 2 , display the old data for student}
procedure tform1.update2;
var SQL_Query : string;
   temp_str : string;
   i : integer;

begin
   with cgienvdata1 do
   begin
      {make up sql query statement for the student's data}
      sql_query := 'select * from friend where std_id="' + getsmallfield('ID') + '"';
      query1.sql.clear;
      query1.sql.add(sql_query);
      query1.active := true;

      {check whether this student id. is in database ? if yes , do here!}
      if query1.recordcount <> 0 then
      begin
         send('<h1> Step 2: Select Column to be changed. </h1>');
         send('<hr>');

         send('<form action="' + cgi_file + '/UPDATE3" method=post>');
         send('<table>');

         {loop for radio box for all column}
         for i := 0 to query1.fieldcount - 1 do
            with query1.fielddefs[i] do
            begin
               send('<tr><td>');

               {the last radio box is always be checkd at the first time ,
                so we can sure that there is column be selected here.}
               if i = query1.fieldcount - 1 then
                  send('<input type=radio checked name="choice" value="' + name + '"> ' + name)
               else
                  send('<input type=radio name="choice" value="' + name + '"> ' + name);
               send('</td>');
```

```
            {display the old value of this column}
            send('<td><b> Old value : </b></td>');
            temp_str := query1.fieldvalues[name];
            send('<td> ' + temp_str + '<td>');
            send('</tr>');
          end;

        send('</table>');

        {some important data will be send to browser with hidden and
         browser can send it back to server again later.}
        send('<input type=hidden name="ID" value="' + getsmallfield('ID') + '">');
        send ('<input type=submit value="Next >"> Go to enter new data value.');
        send ('</form>');

        {button 1}
        send('<form action="' + cgi_file + '/UPDATE1" method=post>');
        send('<input type=submit value="< Back"> Enter new student ID.');
        send('</form>');

        {button 2}
        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Cancel"> Back to the Main page.');
        send('</form>');

      end
      else
      {if student id. is not in database , then do here!}
      begin
        sendtitle('Step 2: No Match');
        send('<h1> There is no data for ' + getsmallfield('ID') + '</h1>');

        {button 1}
        send('<form action="' + cgi_file + '/UPDATE1" method=post>');
        send('<input type=submit value="< Back"> Enter new student ID.');
        send('</form>');

        {button 2}
        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Cancel"> Back to the Main.');
        send('</form>');

      end;
      query1.active :=false;
    end;
end;

{get new value from user to update}
procedure tform1.update3;
var SQL_query : string;
  temp_str : string;

begin
  with cgienvdata1 do
  begin
    send('<h1> Step 3: Enter the new value </h1>');
```

```
        send('<hr>');

        {query the old data value using student id.}
        SQL_query := 'select ' + getsmallfield('choice') + ' from friend where std_id="' +
getsmallfield('ID') + '";
        query1.sql.clear;
        query1.sql.add(sql_query);
        query1.active := true;

        send('<h2> Update data for ' + Getsmallfield('choice') + ' of ' + getsmallfield('ID') + '</h2>');

        {display olf data value}
        send('<b> Old value : </b>');
        temp_str := query1.fieldvalues[getsmallfield('choice')];
        send(temp_str + '<br>');

        {form for receiving new data value}
        send('<form action="' + cgi_file + '/UPDATE4" method=post>');
        send('<b> New value : </b><input type=text name="new_value" size=100>');
        send('<br> <input type=submit value="Next >"> Execute update operation.');
        send('<input type=hidden name="choice" value="' + getsmallfield('choice') + '">');
        send('<input type=hidden name="ID" value="' + getsmallfield('ID') + '">');
        send('</form>');

        {button 2}
        send('<form action="' + cgi_file + '/UPDATE2" method=post>');
        send('<input type=hidden name="ID" value="' + getsmallfield('ID') + '">');
        send('<input type=submit value="< Back"> Change other column.');
        send('</form>');

        {button 3}
        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Cancel"> Back to the Main page.');
        send('</form>');

        query1.active := false;
    end;
end;

{execute sql statement for update}
procedure tform1.update4;
var sql_query : string;
   row_affected : integer;
   i : integer;
   temp_str : string;

begin
    with cgienvdata1 do
    begin
        send('<h1> Step 4: Update Result </h1>');
        send('<hr>');

        {make up update statement}
        sql_query := 'update friend set ' + getsmallfield('choice') + '=';
        if getsmallfield('choice') = 'grade' then
            sql_query := sql_query + getsmallfield('new_value')
```

```
    else
       sql_query := sql_query + "" + getsmallfield('new_value') + "";
    sql_query := sql_query + ' where std_id="' + getsmallfield('ID') + "";

    {execute update statement}
    query1.close;
    query1.sql.clear;
    query1.sql.add(sql_query);
    query1.execsql;


    {check for row affected}
    row_affected := query1.rowsaffected;
    if row_affected <> 0 then

    {if operation is affected}
    begin
        {query new data value}
        query1.active := false;
        sql_query := 'select * from friend where std_id="' + getsmallfield('ID') + "";
        query1.sql.clear;
        query1.sql.add(sql_query);
        query1.active := true;

        {display data value}
        send('<table>');
        for i := 0 to query1.fieldcount -1 do
           with query1.fielddefs[i] do
           begin
              send('<tr>');
              send('<td><b>' + name + ' : </b></td>');
              temp_str := query1.fieldvalues[name];
              send('<td> ' + temp_str + '</td>');
              send('</tr>');
           end;

        send('</table>');

    end
    else
    {if the operation isn't affected any row , then do here}
    begin
        send('No Record affected this operation!');
    end;

    query1.active := false;

    {button 1}
    send('<form action="' + cgi_file + '/UPDATE3" method=post>');
    send('<input type=hidden name="ID" value="' + getsmallfield('ID') + '">');
    send('<input type=hidden name="choice" value="' + getsmallfield('choice') + '">');
    send('<input type=submit value="< Back"> Enter new <b> ' + getsmallfield('choice') + '</b>
value again.');
    send('</form>');

    {button 2}
```

```
        send('<form action="' + cgi_file + '" method=get>');
        send('<input type=submit value="Finish"> Finish operation, Back to the Main page.');
        send('</form>');

    end;
end;

{interactive sql}
procedure tform1.sql;
begin
    with cgienvdata1 do
    begin
        {make form to get sql statement from user}
        send('<form action="' + cgi_file + '/SQL2" method=post>');
        send('SQL : <input type=text name="SQL" value="select * from friend">');
        send('<input type=submit>');
    end;
end;

{result from interactive sql}
procedure tform1.sql2;
begin
    with cgienvdata1 do
    begin
        query1.sql.clear;
        query1.sql.add(getsmallfield('SQL'));
        datasource1.dataset := query1;
        query1.active := true;
        cgidb1.drawtable;
    end;
end;

{When the request is method 'GET' then do this procedure}
procedure TForm1.DoGet;
begin
    with CGIEnvData1 do
    begin
        sendtitle('4D Student Information System');
        send ('<h1> 4D Student Information System </h1>');
        send ('<hr>');
        send ('Please click at button which you want to do');
{VIEW}
        Send ('<form action="' + CGI_File + '/view1" method=post>');
        Send ('<input type=submit value="VIEW"> View all data in 2 steps.');
        Send ('</form>');

{UPDATE}
        Send ('<FORM ACTION="' + CGI_File + '/update1" METHOD=POST>');
        Send ('<INPUT TYPE=SUBMIT VALUE="UPDATE"> Update data in 4 steps.');
        Send ('</FORM>');

{SQL}
        Send ('<form action="' + CGI_File + '/sql" method=post>');
        Send ('<input type=submit value="SQL"> Enter SQL ');
        Send ('</form>');
    end;
```

```
end;


{When the request is method 'POST' then do this procedure}
procedure tform1.DoPost;
var sSelector : String;
begin
   with cgienvdata1 do
   begin
      sselector := cgilogicalpath^;
      sselector := uppercase(sselector);

      if sselector = '/VIEW1' then
         view1
      else if sselector = '/VIEW2' then
         view2
      else if sselector = '/UPDATE1' then
         update1
      else if sselector = '/UPDATE2' then
         update2
      else if sselector = '/UPDATE3' then
         update3
      else if sselector = '/UPDATE4' then
         update4
      else if sselector = '/SQL' then
         sql
      else if sselector = '/SQL2' then
         sql2
      else
         send('<h1> ERROR </h1>');

   end;
end;


{This CGI program will start here!}
procedure TForm1.FormCreate(Sender: TObject);
var  Temp_Method : String;
begin
   with CGIEnvData1 do
   begin

      websiteINIFilename := paramstr(1);

      application.onException := cgiErrorHandler;
      application.processMessages;

      createStdout;
      sendPrologue;

      send( '<HTML><HEAD>' );
      sendTitle( 'Simple CGI by Delphi' );
      send( '</HEAD><BODY>' );

      temp_method := cgirequestmethod^;
```

```
      if temp_method='GET' then
        doget
      else
      if temp_method='POST' then
        dopost
      else
        send('Your method is not recognized');

      sendfooter;
      closeStdout;

  end;
  closeApp( application );
end;

end.
```

ภาคผนวก ช
การสร้างแบบฟอร์มด้วยภาษา HTML

ในการส่งข้อมูลจากบราวเซอร์ทางฝั่งผู้ใช้งาน มาให้กับ CGI แอพพลิเคชั่นทางฝั่งเซิร์ฟเวอร์นั้น มักจะสร้างเป็นแบบฟอร์มสำหรับรับข้อมูล โดยภาษา HTML นั้นจะมีแทค (tag) สำหรับการสร้างแบบฟอร์มซึ่งช่วยให้ผู้พัฒนาระบบงานสามารถสร้างแบบฟอร์มสำหรับรับข้อมูลได้ง่าย และสะดวกมาก โดยแทคดังกล่าวนี้มีรูปแบบดังนี้

```
<FORM ACTION="filename" METHOD="method">
.....
<INPUT TYPE=SUBMIT>
</FORM>
```

ส่วนประกอบที่สำคัญของแทคนี้มีดังนี้

1. ส่วนของ filename ซึ่งใช้ในการระบุ URL ของ CGI แอพพลิเคชั่นที่จะรับข้อมูล จากแบบฟอร์มนี้ไปทำการประมวลผล
2. ส่วนของ method เป็นการระบุวิธีการในการส่งข้อมูลจากแบบฟอร์ม ไปให้กับ CGI แอพพลิเคชั่น โดยมี 2 วิธีที่นิยมใช้คือ วิธี GET และ วิธี POST ซึ่งได้อธิบายไว้ในบทที่ 1 แล้ว
3. ส่วนของตัวรับข้อมูล ระหว่างแทค <FORM> และ </FORM> สามารถสร้างตัวรับข้อมูลชนิดต่าง ๆ เพื่อใช้งานได้ ซึ่งการสร้างแบบฟอร์มรับข้อมูลในภาษา HTML นี้มีตัวรับข้อมูลชนิดต่าง ๆ ให้เลือกใช้มากมาย ตัวรับข้อมูลที่สำคัญที่สุดคือ ตัวรับข้อมูลชนิด SUBMIT ซึ่งจะแสดงเป็นปุ่มให้ผู้ใช้กดเมื่อต้องการจะส่งข้อมูลเหล่านั้นไปให้ CGI แอพพลิเคชั่นที่กำหนด

ตัวรับข้อมูลนี้มีหลายชนิด สามารถกำหนดได้ในพารามิเตอร์ TYPE โดยตัวรับข้อมูลนี้ มีชนิด และรูปแบบการใช้งานดังนี้

ตัวรับข้อมูลชนิดข้อความ 1 บรรทัด

เป็นการรับข้อความตัวอักษร , ตัวเลข หรือสัญญลักษณ์ต่าง ๆ ที่มีขนาดไม่เกิน 256 ตัวอักษร มีรูปแบบการกำหนดดังนี้

<INPUT TYPE=TEXT NAME="text1">

พารามิเตอร์ NAME เป็นการกำหนดชื่อของตัวรับข้อมูลนี้ ซึ่งตัวอย่างนี้กำหนดให้มีชื่อว่า text1 การที่ต้องกำหนดชื่อให้กับตัวรับข้อมูลก็เพื่อ CGI แอพพลิเคชั่นจะได้สามารถรับข้อมูลจากตัวรับข้อมูลชนิดนี้โดยอ้างถึงชื่อของตัวรับข้อมูลนั่นเอง

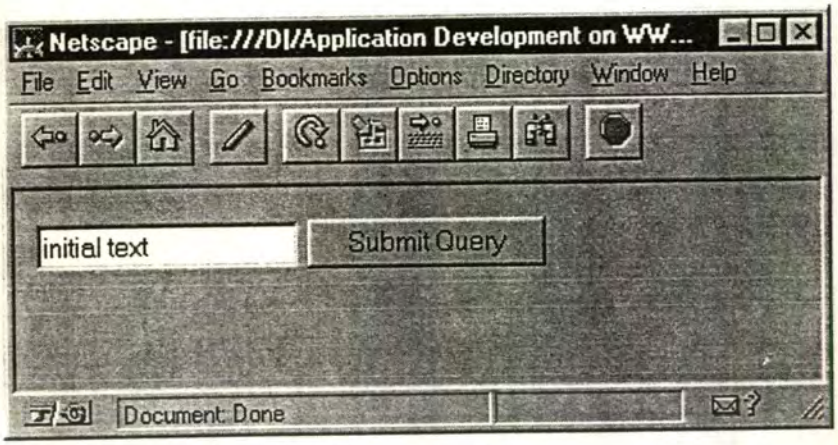
นอกจากพารามิเตอร์ NAME แล้วยังมีพารามิเตอร์อื่น ๆ ที่อาจใช้ หรือไม่ใช้ก็ได้ เช่น

1. SIZE ใช้สำหรับกำหนดขนาดของตัวรับข้อมูลที่จะแสดงผลบนบราวเซอร์
2. VALUE ใช้สำหรับแสดงข้อความเริ่มต้นในตัวรับข้อมูล
3. MAXLENGTH ใช้สำหรับกำหนดขนาดความยาวของข้อความสูงสุดที่ตัวรับข้อมูลสามารถรับได้

เมื่อรวมพารามิเตอร์ต่าง ๆ นี้จะได้แทครูปแบบดังนี้

```
<FORM ACTION="my_cgi" METHOD=POST>
<INPUT TYPE=TEXT  NAME="text1" SIZE=20 VALUE="initial text" MAXLENGTH=15>
<INPUT TYPE=SUBMIT>
</FORM>
```

เป็นการสร้างตัวรับข้อมูลชนิดรับข้อความ 1 บรรทัด ชื่อ "text1" มีข้อความเริ่มต้นว่า "initial text" สามารถรับข้อความสูงสุดได้ 15 ตัวอักษร และตัวรับข้อมูลนี้จะแสดงผลบนบราวเซอร์โดยมีขนาด 20 ตัวอักษร ซึ่งจะได้ผลลัพธ์ดังรูป



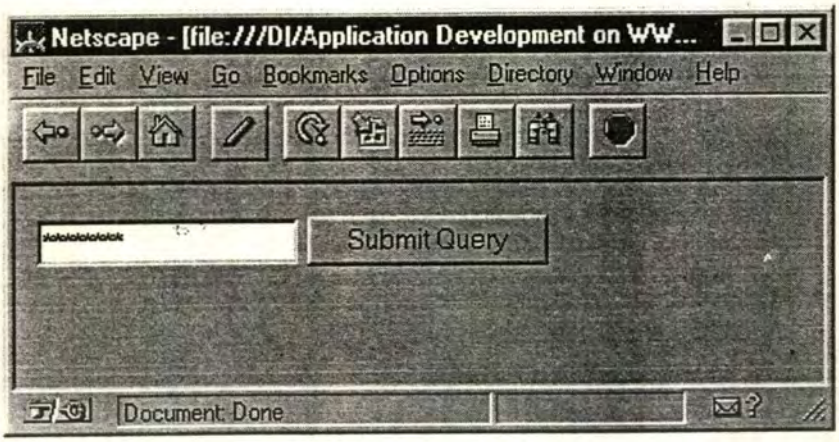
รูปที่ ช - 1 ตัวรับข้อมูลชนิดข้อความ 1 บรรทัด

ตัวรับข้อมูลชนิดรหัสลับ

คล้ายกับตัวรับข้อมูลชนิดข้อความ 1 บรรทัด แต่ว่า ข้อความที่ผู้ใช้งานใส่ลงในตัวรับข้อมูลนี้จะถูกแสดงผลเป็นเครื่องหมายดอกจัน ตัวรับข้อมูลชนิดนี้สามารถสร้างได้โดยกำหนดพารามิเตอร์ TYPE=PASSWORD และตั้งชื่อตัวรับข้อมูลโดยใช้พารามิเตอร์ NAME ส่วนพารามิเตอร์อื่น ๆ ที่อาจมีได้เช่น SIZE , MAXLENGTH เป็นต้น ดังตัวอย่างการใช้งาน

```
<FORM ACTION="my_cgi" METHOD=POST>
<INPUT TYPE=PASSWORD NAME="pwd" SIZE=20 MAXLENGTH=8>
<INPUT TYPE=SUBMIT>
</FORM>
```

ซึ่งจะได้ผลลัพธ์เป็นตัวรับข้อมูลที่สามารถรับรหัสลับได้สูงสุด 8 ตัวอักษร และเมื่อผู้ใช้งานใส่ข้อมูล ก็จะแสดงเป็นเครื่องหมายดอกจัน



รูปที่ ช - 2 ตัวรับข้อมูลชนิดรหัสลับ

ตัวรับข้อมูลชนิดตัวเลือก

เป็นชุดของตัวรับข้อมูล ซึ่งจะให้ผู้ใช้งานสามารถเลือกได้ กำหนดโดย

```
<INPUT TYPE=CHECKBOX NAME="checklist" VALUE="value1">
<INPUT TYPE=CHECKBOX NAME="checklist" VALUE="value2">
.....
```

โดยตัวรับข้อมูลในชุดเดียวกันจะต้องตั้งชื่อให้เหมือนกัน และพารามิเตอร์ VALUE จะเป็นการกำหนดค่าที่จะส่งให้กับ CGI แอพพลิเคชั่นเมื่อผู้ใช้เลือกตัวเลือกนั้น ๆ ตัวเลือกใดที่ผู้ใช้ไม่ได้เลือก ก็จะไม่ถูกส่งค่าไปยัง CGI แอพพลิเคชั่น

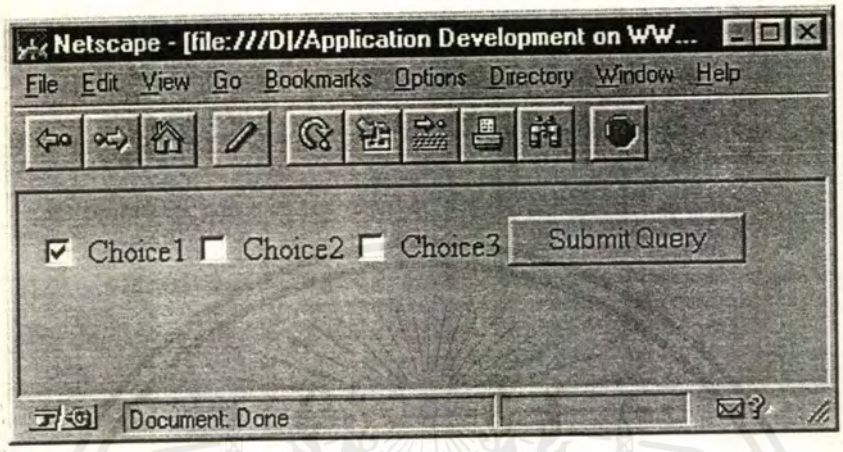
ในกรณีที่ผู้ใช้เลือกตัวเลือกหลาย ๆ ตัว ข้อมูลที่ส่งไปให้ CGI แอพพลิเคชั่นจะอยู่ในรูปแบบเช่น checklist=value1,checklist_1=value2,checklist_2=value3 ... เป็นต้น

พารามิเตอร์ที่อาจใช้เพิ่มเติมในการสร้างตัวรับข้อมูลชนิดนี้ เช่น CHECKED เป็นการกำหนดให้ตัวเลือกนั้นถูกเลือกโดยอัตโนมัติเมื่อเริ่มต้น ตัวอย่างการใช้งาน เช่น

```
<FORM ACTION="my_cgi" METHOD=POST>
<INPUT TYPE=CHECKBOX NAME="checklist" VALUE="value1" CHECKED> Choice1
<INPUT TYPE=CHECKBOX NAME="checklist" VALUE="value2"> Choice2
<INPUT TYPE=CHECKBOX NAME="checklist" VALUE="value3"> Choice3
<INPUT TYPE=SUBMIT>
```

</FORM>

ซึ่งจะได้ผลลัพธ์เป็นตัวรับข้อมูลชนิดตัวเลือกจำนวน 3 ตัวเลือก และตัวเลือกที่ 1 ถูกเลือกโดยอัตโนมัติตั้งแต่เริ่มต้นแล้ว



รูปที่ ช - 3 ตัวรับข้อมูลชนิดตัวเลือก

ตัวรับข้อมูลชนิดตัวเลือกเดียว

มีลักษณะคล้ายกับตัวรับข้อมูลชนิดตัวเลือก แต่ว่าผู้ใช้งานสามารถเลือกตัวเลือกได้เพียงตัวเดียวเท่านั้น สามารถกำหนดได้ดังนี้

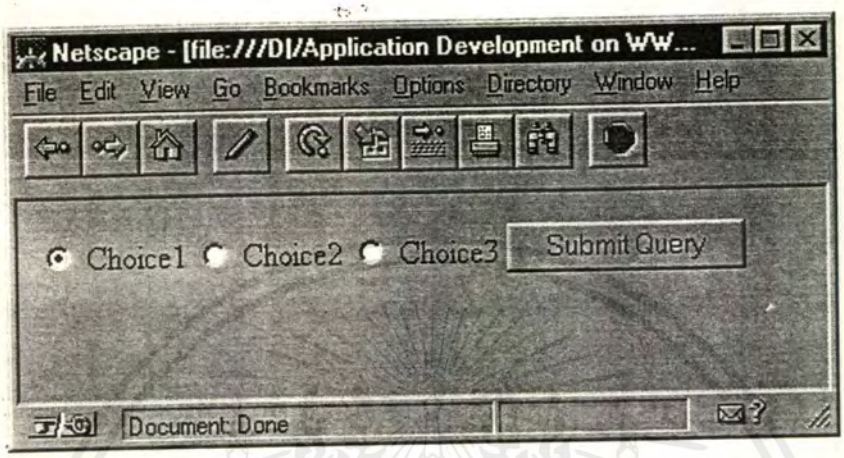
```
<INPUT TYPE=RADIO NAME="checklist" VALUE="value1">
<INPUT TYPE=RADIO NAME="checklist" VALUE="value2">
....
```

การตั้งชื่อตัวรับข้อมูลชนิดตัวเลือกเดียวที่อยู่ในชุดเดียวกันจะต้องตั้งชื่อให้เหมือนกัน และสามารถกำหนดให้ตัวเลือกตัวใดตัวหนึ่งถูกเลือกตั้งแต่เริ่มต้นได้โดยใช้พารามิเตอร์ CHECKED เช่นเดียวกับตัวรับข้อมูลชนิดตัวเลือก ตัวอย่างการใช้งานเช่น

```
<FORM ACTION="my_cgi" METHOD=POST>
<INPUT TYPE=RADIO NAME="checklist" VALUE="value1" CHECKED> Choice1
<INPUT TYPE=RADIO NAME="checklist" VALUE="value2"> Choice2
<INPUT TYPE=RADIO NAME="checklist" VALUE="value3"> Choice3
<INPUT TYPE=SUBMIT>
```

</FORM>

ซึ่งจะได้ผลลัพธ์เป็นตัวรับข้อมูลชนิดตัวเลือกเดียวจำนวน 3 ตัวเลือก และตัวเลือกแรกถูกเลือกตั้งแต่เริ่มต้น



รูปที่ ช - 4 ตัวรับข้อมูลชนิดตัวเลือกเดียว

ตัวรับข้อมูลชนิดแถบตัวเลือก

เป็นตัวรับข้อมูลที่ให้ผู้ใช้งานเลือกตัวเลือกตัวใดตัวหนึ่งจากแถบตัวเลือก เมื่อผู้ใช้กดเมาส์ที่แถบตัวเลือก จะปรากฏรายการตัวเลือกขึ้นมาให้เลือก การสร้างตัวรับข้อมูลชนิดนี้จะมีรูปแบบ ที่แตกต่างจากตัวรับข้อมูลชนิดอื่น ๆ โดยสามารถสร้างในรูปแบบดังนี้

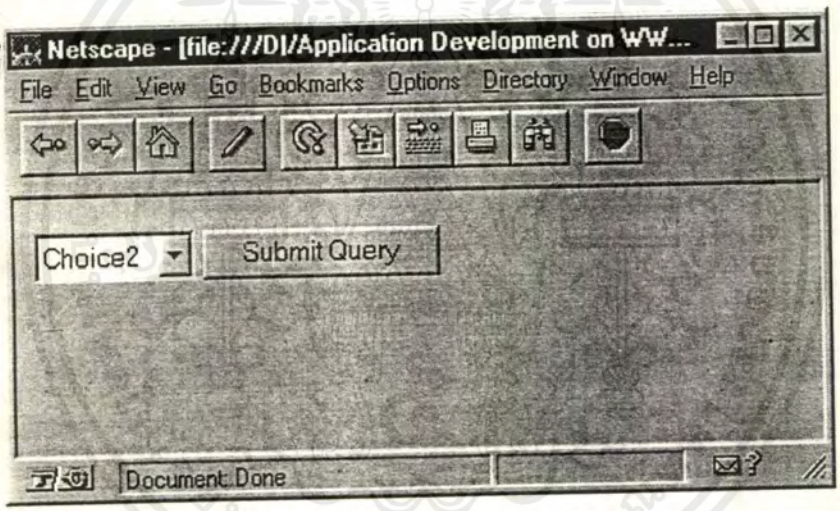
```
<SELECT NAME="checklist">
<OPTION VALUE="value1">
<OPTION VALUE="value2">
...
</SELECT>
```

การสร้างตัวรับข้อมูลชนิดแถบตัวเลือกนี้สามารถสร้างตัวเลือกกี่ตัวก็ได้ โดยระบุไว้ในแทค OPTION นอกจากนี้ยังมีพารามิเตอร์ที่อาจเพิ่มเติมได้เช่น SELECTED ใช้ระบุไว้ที่ตัวเลือกที่ต้องการให้ถูกเลือกเมื่อเริ่มต้น ในกรณีที่ไม่ได้ระบุพารามิเตอร์นี้ ตัวเลือกแรกจะเป็นตัวเลือกที่ถูกเลือกเมื่อเริ่มต้น ตัวอย่างการใช้งานเช่น

```
<FORM ACTION="my_cgi" METHOD=POST>
```

```
<SELECT NAME="checklist">
<OPTION VALUE="value1"> Choice1
<OPTION VALUE="value2" SELECTED> Choice2
<OPTION VALUE="value3"> Choice3
</SELECT>
<INPUT TYPE=SUBMIT>
</FORM>
```

จะเป็นตัวรับข้อมูลชนิดแถบตัวเลือก โดยเมื่อผู้ใช้กดเมาส์ที่แถบตัวเลือก จะปรากฏตัวเลือกขึ้นมา 3 ตัวเลือก คือ Choice1 , Choice2 และ Choice3 โดยตัวเลือก Choice2 จะเป็นตัวเลือกที่ถูกเลือกเมื่อเริ่มต้น



รูปที่ ช - 5 ตัวรับข้อมูลชนิดแถบตัวเลือก

ตัวรับข้อมูลชนิดข้อความหลายบรรทัด

เป็นตัวรับข้อมูลที่สามารถรับข้อมูลจากผู้ใช้ โดยข้อมูลเป็นแบบข้อความ , ตัวเลข หรือสัญญลักษณ์ต่าง ๆ หลาย ๆ บรรทัด และมีขนาดมากกว่า 256 ตัวอักษรได้มีรูปแบบการกำหนดคือ

```
<TEXTAREA NAME="area1">
</TEXTAREA>
```

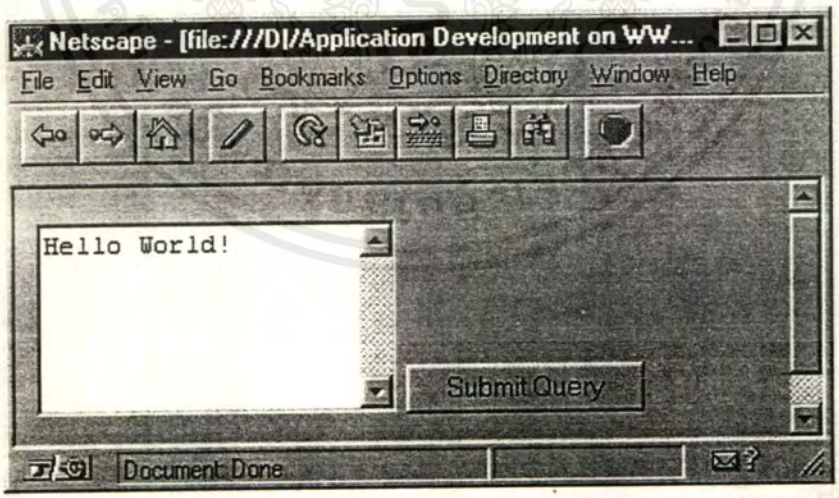
นอกจากนี้ยังมีพารามิเตอร์เพิ่มเติมต่าง ๆ ที่สามารถใส่เข้าไปเพื่อจัดรูปแบบการแสดงผลให้เหมาะสมมากขึ้นเช่น

1. ROWS ใช้สำหรับกำหนดจำนวนบรรทัดของตัวรับข้อมูล
2. COLS ใช้สำหรับกำหนดจำนวนคอลัมน์ของตัวรับข้อมูล
3. WRAP=VIRTUAL เป็นการกำหนดเพื่อให้มีการตัดคำโดยอัตโนมัติ

ตัวอย่างการใช้งานเช่น

```
<FORM ACTION="my_cgi" ACTION=POST>
<TEXTAREA NAME="area1" COLS=20 ROWS=5 WRAP=VIRTUAL>
Hello World!
</TEXTAREA>
<INPUT TYPE=SUBMIT>
</FORM>
```

จากตัวอย่าง เป็นการสร้างตัวรับข้อมูลแบบข้อความหลายบรรทัด โดยตัวรับข้อมูลนี้มีขนาดกว้าง 20 ตัวอักษร และมีขนาด 5 บรรทัด และสามารถตัดคำได้โดยอัตโนมัติ และในตัวรับข้อมูลนี้เริ่มต้นจะมีข้อความว่า "Hello World!"



รูปที่ ช - 6 ตัวรับข้อมูลชนิดข้อความหลายบรรทัด

ตัวรับข้อมูลชนิดยกเลิก

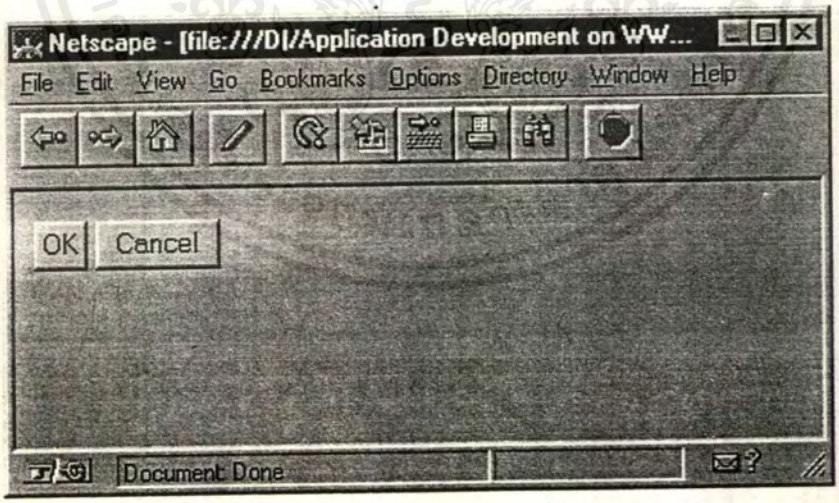
ใช้สำหรับการยกเลิกข้อความทั้งหมดที่ผู้ใช้ใส่ลงในแบบฟอร์ม และให้มีค่าต่าง ๆ ในฟอร์มเป็นค่าเริ่มต้นที่กำหนดใน HTML มีรูปแบบดังนี้

```
<INPUT TYPE=RESET>
```

พารามิเตอร์เพิ่มเติมที่สามารถนำมาใช้ร่วมกับตัวรับข้อมูลชนิดนี้ได้ เช่น VALUE ใช้ระบบข้อความที่จะแสดงบนตัวรับข้อมูล ซึ่งหากไม่ได้กำหนดพารามิเตอร์นี้ จะแสดงข้อความ "RESET" ตัวอย่างการใช้งานเช่น

```
<FORM ACTION="my_cgi" METHOD=POST>
<INPUT TYPE=SUBMIT VALUE="OK">
<INPUT TYPE=RESET VALUE="Cancel">
</FORM>
```

เป็นการแสดงตัวรับข้อมูลชนิดตกลง และชนิดยกเลิก โดยมีข้อความว่า "OK" และ "Cancel" บนตัวรับข้อมูลทั้งสองตามลำดับ



รูปที่ ช - 7 ตัวรับข้อมูลชนิดยกเลิก

ภาคผนวก ซ
การรันจาวาแอพพลิเคชั่นที่ตัวจาวาเซิร์ฟเวอร์

ในขณะนี้ ทางบริษัทซันไมโครซิสเต็มส์ได้ออกผลิตภัณฑ์ใหม่ คือ จาวาเซิร์ฟเวอร์ (JAVA Server) และ เซิร์ฟเล็ทเดเวลอปเปอร์คิท (Servlet Developer Kit) ซึ่งทำให้เราสามารถสร้างเซิร์ฟเวอร์ที่อาศัยฐานภาษาจาวาเป็นหลัก และมีเซิร์ฟเล็ท เอพีไอ (Servlet API) เป็นเครื่องมือสำคัญในการใช้งานจาวาเซิร์ฟเวอร์นั้น ๆ

จาวาเซิร์ฟเวอร์

จาวาเซิร์ฟเวอร์ เป็นเซิร์ฟเวอร์ที่เขียนขึ้นโดยใช้ภาษาจาวาทั้งหมด ซึ่งสามารถ Download ได้ที่ java.sun.com โดยจะยังคงเป็นจาวาเซิร์ฟเวอร์ เวอร์ชัน เบต้า 2 อยู่ โดย จาวาเซิร์ฟเวอร์นี้จะเป็นเอชทีทีพีเซิร์ฟเวอร์ (HTTP Server) สนับสนุนการทำงานของจาวาแอพพลิเคชัน และ CGI (Common Gateway Interface)

หลักการทำงานของจาวาเซิร์ฟเวอร์

จากคุณสมบัติของจาวาที่สามารถทำงานแบบมัลติเพิลเทรดได้ จาวาเซิร์ฟเวอร์จะรับการติดต่อจากเครื่องไคลเอ็นท์ที่ต่อเข้ากับเซิร์ฟเวอร์นี้ โดยจะใช้เทรด (thread) เป็นตัวรับการติดต่อ (connection) โดยทางจาวาเซิร์ฟเวอร์จะมีเทรดหลัก 1 เทรด เพื่อใช้รอรับการติดต่อ โดยการรับการติดต่อแต่ละครั้ง เซิร์ฟเวอร์จะตรวจสอบตลอดว่ายังคงมีเทรดเหลือ 1 เทรดเพื่อรอรับการติดต่อ เมื่อเทรดนั้นได้รับการติดต่อจากไคลเอ็นท์แล้ว ก็จะนำเทรดนั้นเข้าไปไว้ในพูล (Pool) การทำงาน (เป็นคิวการทำงานของเทรดที่นำมารวมกันไว้) โดยขนาดของพูลนี้จะสามารถเปลี่ยนแปลงแก้ไขได้ตลอดเวลา

ตราบใดที่เทรดการรับการติดต่อยังคงไม่เกินที่เซิร์ฟเวอร์ตั้งค่าจำกัดไว้ เซิร์ฟเวอร์ก็ยังคงรับการติดต่อได้เรื่อย ๆ แต่เมื่อเต็มแล้ว การติดต่อนั้นก็จะถูกบล็อกไว้ และถ้าเทรดที่รับ

การติดต่อใด ๆ ต้องรออยู่นานเกินกำหนดและในคิวการทำงานยังคงทำงานกันอยู่ เทรดนั้นก็จะหมดอายุ หรือ ถูกยกเลิกไปโดยอัตโนมัตินั่นเอง

ด้วยจาวาเซิร์ฟเวอร์ นี้ ทำให้หลักการของการใช้จาวาแอพพลิเคชัน ทำงานที่ฝั่งเซิร์ฟเวอร์ได้ ซึ่งจะขออธิบายต่อไป

เซิร์ฟเล็ท เอพีไอ (Servlet API)

เซิร์ฟเล็ทเอพีไอนี้ เป็นเอพีไอที่เพิ่มประสิทธิภาพการทำงานที่ฝั่งเซิร์ฟเวอร์ของจาวาแอพพลิเคชัน โดยเพิ่มประสิทธิภาพทางด้านการติดต่อกับทางไคลเอ็นท์

เซิร์ฟเล็ทนี้ เป็นคอมโพเนนท์ (component) ของจาวาแอพพลิเคชันทางฝั่งเซิร์ฟเวอร์ ซึ่งช่วยทำให้เซิร์ฟเวอร์สามารถรันจาวาได้ โดยมีหลักการทำงานเพื่อรับการเรียกใช้บริการต่าง ๆ นั่นเอง

และเนื่องจากเซิร์ฟเล็ทนี้จะรันอยู่ที่ตัวเซิร์ฟเวอร์ จึงไม่จำเป็นต้องมีส่วนติดต่อกับผู้ใช้แบบกราฟฟิค (Graphic User Interface - GUI) แสดงการทำงาน

นอกจากนี้ เซิร์ฟเล็ทยังถูกออกแบบมาให้สามารถให้บริการพื้นฐานต่าง ๆ ได้ตาม มาตรฐานของเว็บเซิร์ฟเวอร์ เช่น การให้บริการข้อมูลแบบสแตติก (static) ในรูป HTML Pageผ่านโปรโตคอล HTTP หรือสามารถนำไปใช้เป็น Plug-in ในแอพพลิเคชันอื่น ๆ ได้ด้วย

ถึงแม้จะเขียนด้วยภาษาจาวาเป็นหลัก การติดต่อกับไคลเอ็นท์ ก็สามารถติดต่อกับไคลเอ็นท์ที่เขียนด้วยภาษาอื่น ได้ด้วย ไม่จำเป็นต้องเป็นภาษาจาวาอย่างเดียว

การนำเซิร์ฟเล็ทไปใช้งาน

อาจจะนำไปใช้ได้ในหลาย ๆ รูปแบบ เช่น

- จัดการกับข้อมูลที่ส่งผ่านโปรโตคอล HTTP โดยใช้ HTML-Form และมีการส่งผ่านข้อมูลต่าง ๆ เช่น การส่งราคาสินค้า รายการสั่งซื้อสินค้า เป็นต้น
- การรับการติดต่อหลาย ๆ การติดต่อพร้อม ๆ กัน ทำให้สามารถสนับสนุนการทำงานที่ใช้แอพพลิเคชันเดียวกันได้ เช่น การทำการประชุมออนไลน์ เป็นต้น
- สามารถสร้างกลุ่มของการติดต่อ (community) โดยมีการกำหนดรหัสแสดงกลุ่มการติดต่อ ซึ่งจะอยู่ในรูปของเซิร์ฟเล็ทนั่นเอง

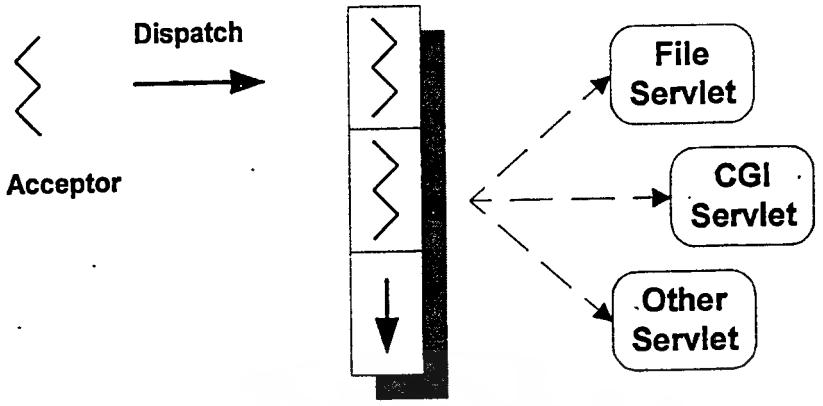
ลักษณะของเซิร์ฟเล็ทเอพีไอ

จาวาเซิร์ฟเล็ทเอพีไอ เวอร์ชันใหม่ล่าสุดนี้ จะเป็นเอพีไอขยายเพิ่มเติมจากเอพีไอมาตรฐาน ซึ่งปัจจุบัน เว็บเซิร์ฟเวอร์ที่สนับสนุนการทำงานของจาวาก็สนับสนุนการทำงานของเซิร์ฟเล็ทเอพีไอนี้เกือบหมดแล้ว โดยการเขียนโดยใช้เอพีไอนี้ จะอยู่ในรูปของการรับการเรียกใช้บริการและการตอบสนองการขอบริการ ดังตัวอย่างการเขียน ดังนี้

```
public class Myservlet extends GeneriServlet {
        public void service (
                ServletRequest          request,
                ServletResponse  response
        ) throws ServletException,  IOException
    {
                .....
    }
}
```

การใช้งาน HTTP Server กับ Servlet

HTTP Server จะใช้หลักการของเทรดเพื่อรับการติดต่อจาก HTTP โดยเซิร์ฟเล็ทที่ใช้กับโปรโตคอล HTTP นี้ จะสนับสนุน Method ของ HTTP ตั้งแต่ GET,POST ,HEAD โดยเมื่อ HTTP-Request ได้รับการตอบสนองจากเทรดที่มารับการติดต่อ เทรดก็จะตรวจสอบสิทธิ์และการกำหนดชื่อของการรับการติดต่อ นี้ โดยขึ้นอยู่กับการกำหนดของผู้ที่ดูแลเว็บเซิร์ฟเวอร์อีกทีหนึ่ง โดยการกำหนดชื่อและกลุ่มของเทรด หรือ การติดต่อนี้ ตัวจัดการ เทรดจะตัดสินเองว่า เซิร์ฟเล็ทใดเหมาะกับการเรียกใช้.



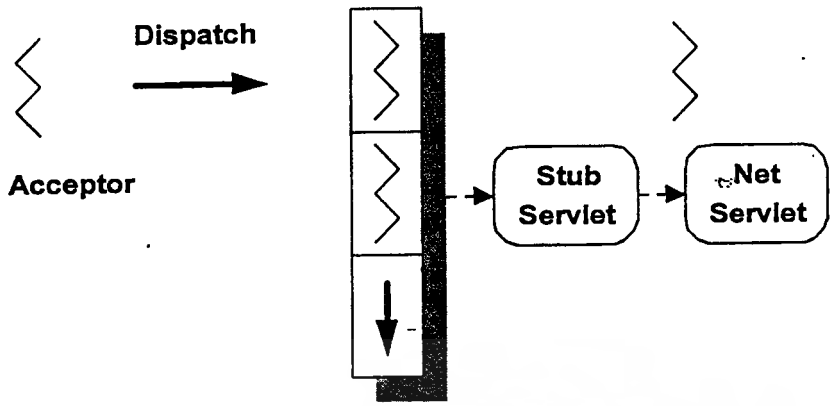
จากภาพจะเห็นว่า มี แฮนเดิลเลอร์เทรดพูล (Handler threads pool) เป็นตัวเก็บเทรดทั้งหลายที่รับการคอนเนคชั่นมา และจะถูกตัดสินใจว่าควรจะเป็นเซิร์ฟเล็ทประเภทใด

เซิร์ฟเล็ทนี้ จะเหมือนกับแอพเพล็ทตรงที่เป็นออบเจ็กต์ของไบท์โค้ดซึ่งสามารถโหลดขึ้นมาได้ทั้งภายในเน็ตเวิร์กหรือภายนอก แต่จุดแตกต่างที่สำคัญคือ เซิร์ฟเล็ทไม่มีส่วนติดต่อกับผู้ใช้ ทำให้มองไม่เห็นการทำงานของมัน และการอ้างถึงเซิร์ฟเล็ทนี้ก็จะอ้างผ่านทางยู อาร์ แอล (URL) เช่นกัน

เซิร์ฟเล็ทนี้ อาจจะมีแบ่งออกเป็นหลาย ๆ แบบ เช่น

- **ไฟล์เซิร์ฟเล็ท (File Servlet)** จะรับข้อมูลมาตรฐานทั่ว ๆ ไป
- **ซีจีไอเซิร์ฟเล็ท (CGI Servlet)** จะทำหน้าที่เสมือนเป็นเกทเวย์สำหรับอินเตอร์เฟซของ CGI 1.1
- **อิมเมจแมปเซิร์ฟเล็ท (Imagemap servlet)** จะรับรูปภาพในฝั่งของเซิร์ฟเวอร์นั่นเอง เป็นต้น

ในการทำงาน เมื่อเซิร์ฟเล็ทใดได้ตอบรับการคอนเนคชันแล้ว และเลือกแล้วว่าจะเป็นเซิร์ฟเล็ทแบบใด ก็จะเลือกเซิร์ฟเล็ทปลายทางที่จะติดต่อด้วย ดังภาพ



ตัวอย่างการเขียนเซิร์ฟเล็ท และการเขียนฝังไว้ในไฟล์เอชทีเอ็มแอล ซึ่งในที่นี้จะใช้ไฟล์เอสเอชทีเอ็มแอล (S-HTML File)

ตัวอย่างของเซิร์ฟเล็ท

```
/*
 * @(#)TestServlet.java 1.6 96/10/14
 *
 * Copyright (c) 1996 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */

import java.servlet.*;
import java.io.*;
import java.util.Date;
import java.util.Hashtable;
```

```
import sun.server.http.HttpServlet;
import sun.server.http.HttpRequest;
import sun.server.http.HttpResponse;

/**
 * TestServlet Servlet
 *
 * This is a simple servlet to demonstrate server-side includes
 * @author Scott Atwood
 * @version 1.6, 10/14/96
 */
public class TestServlet extends GenericServlet {

   public void init(ServletStub stub) {
      try {
         super.init(stub);
      } catch (Exception e) {
         e.printStackTrace();
      }
      System.out.println("TestServlet.init()");
   }

   public void service(ServletRequest req, ServletResponse res)
   throws ServletException, IOException
   {
      System.out.println("Test Servlet.service()");
      ServletOutputStream out = res.getOutputStream();
      out.println("init1: "+getInitParameter("init1"));
      out.println("init2: "+getInitParameter("init2"));
      out.println("request params: "+req.getParameters());
   }

   public String getServletInfo() {
      return "Sample server-side includes";
   }
}
```

ตัวอย่างการเขียนไฟล์เอสเอชทีเอ็มแอล ซึ่งจะฝังเซิร์ฟเล็ทที่เรียกใช้ไว้ภายใน ดังโค้ด

```
<h3>Test Servlet</h3>
<servlet code=TestServlet init1=one init2=two>
<param name=foo value=bar>
<param name=foo2 value=bar2>
</servlet>
```

จากที่กล่าวมาถึงทั้ง 2 ส่วน คือ เซิร์ฟเล็ท และ จาวาเซิร์ฟเวอร์ ทำให้เราสามารถนำจาวาไปรันบนตัวเซิร์ฟเวอร์ได้ โดยการรับส่งข้อมูลจากเซิร์ฟเวอร์ไปยังจาวา หรือ ในทางกลับกัน ซึ่งแต่เดิมเราอาจจะใช้การส่งจากจาวาแอพพลิเคชัน หรือ แอพเพล็ท ไปยัง ซีจีไอ จากนั้นส่งไป

ยังเซิร์ฟเวอร์อีกต่อหนึ่ง แต่ถ้าเราใช้จาวาเซิร์ฟเวอร์ หรือ เซิร์ฟเล็ท ก็จะทำให้เราสามารถใช้เซิร์ฟเล็ทและจาวาเซิร์ฟเวอร์นั้นควบคุมการส่งต่อข้อมูลได้เลย ทำให้การควบคุมสะดวกมากขึ้น

กิตติกรรมประกาศ

โครงงานชิ้นนี้สามารถสำเร็จลุล่วงไปได้ด้วยดีนั้นก็เพราะได้รับความช่วยเหลือ และการสนับสนุนจากบุคคลหลาย ๆ ท่าน

ขอขอบพระคุณ บิดา-มารดา ที่ได้อบรมสั่งสอน และให้ความสนับสนุนในทุก ๆ เรื่อง

ขอขอบพระคุณ ครูบาอาจารย์ทุกท่าน ที่ได้อบรมสั่งสอนทั้งด้านวิชาการและจริยธรรม

ขอขอบพระคุณ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา ที่ได้ให้คำชี้แนะ คำแนะนำต่าง ๆ

ขอขอบคุณ Robert B. Denny และ Ann Lynnworth ที่ให้ความรู้เรื่อง CGI

ขอขอบคุณ Phillip Zimmermann ที่ให้ความรู้เกี่ยวกับระบบความปลอดภัยของข้อมูล

ขอขอบคุณ บริษัท ซันไมโครซิสเต็มท์ , O'Reilly & Association ให้ซอฟแวร์มาทดลองใช้

ขอขอบคุณ พี่ตัวเล็ก , พี่หนึ่ง ที่ช่วยเหลือในด้านสถานที่

ขอขอบคุณ ป๊อก (ธนันท์) , นึก (สมนึก) , ดี่เล็ก (ชาญชัย) , ไก่ (สุชยันต์), โด (กิตติพงษ์) , คม (นิคม) , Big (จาตุรนต์) , ส้ม (สมภพ) , นึก (เบญศพล) , โด้ (อัศวิน) , หงอก (ทรงชัย) , เผือก (ธนพร) และเพื่อน ๆ ทุกคน ที่ให้ความช่วยเหลือ ให้กำลังใจ

หนังสืออ้างอิง

"Network Security PRIVATE Communication in a PUBLIC World" (Charlie Kaufman , Radia Perlman , Mike Speciner) Prentice Hall p.57-142