การพัฒนาซอฟต์แวร์อิดิเตอร์และอินเตอร์พรีเตอร์สำหรับการแปลภาษารวมเว็บเพจและเซิร์ฟเล็ต

DEVELOPMENT OF EDITOR AND INTERPRETER FOR JAVA SERVLET TEMPLATE ENGINE

ธนพัฒน์ รติพิชญ์พร

วิมลวรรณ ชินฉลองพร

อนุชาติ อัศววิวัฒน์พงศ์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

DEVELOPMENT OF EDITOR AND INTERPRETER FOR JAVA SERVLET TEMPLATE ENGINE

THANAPAT RATIPITCHABHORN

VIMOLWAN CHINCHALONGPORN

ANUCHARD AUSSAWAVIWATPONG

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2002

หัวข้อปัญหาพิเศษ	การพัฒนาซอฟต์แวร์อิดิเตอร์และอินเตอร์พรีเตอร์สำหรับการแปลภาษารวมเว็บเพจและเซิร์ฟเล็ต DEVELOPMENT OF EDITOR AND INTERPRETER FOR JAVA SERVLET TEMPLATE ENGINE	
ชื่อนักศึกษา	นายธนพัฒน์ รติพิชญ์พร	42050393
	นางสาววิมลวรรณ ชินฉลองพร	42050442
	นายอนุชาติ อัศววิวัฒน์พงศ์	42050462
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	ดร.กิตติมา เมฆาบัญชากิจ	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ปีการศึกษา 2545

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	อ. สิริลักษณ์ อนันต์สถิตย์สิน	
กรรมการ	ดร. นันทิกา เบญจเทพานันท์	
กรรมการและอาจารย์ที่ปรึกษา	ดร.กิตติมา เมฆาบัญชากิจ	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปัญหาพิเศษ	การพัฒนาซอฟต์แวร์อิดิเตอร์และอินเตอร์พรีเตอร์สำหรับการแปลภาษารวมเว็บเพจและเซิร์ฟเล็ต	
ชื่อนักศึกษา	นายธนพัฒน์ รติพิชญ์พร	42050393
	นางสาววิมลวรรณ ชินฉลองพร	42050442
	นายอนุชาติ อัศววิวัฒน์พงศ์	42050462
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2545	
อาจารย์ที่ปรึกษา	ดร.กิตติมา เมฆาบัญชากิจ	

บทคัดย่อ

อินเตอร์เน็ตมีบทบาทกับชีวิตประจำวันของคนเราและเข้ามามีอิทธิพลทางธุรกิจต่างๆ มากมาย ข้อมูลและข่าวสารทั้งหลายถูกส่งผ่านทางสื่อบนอินเตอร์เน็ตหรือที่รู้จักกันในนามของเว็บไซต์การพัฒนาเว็บไซต์โดยปกติแล้วจะมีขั้นตอนต่าง ๆ ในการดำเนินการมากมาย อีกทั้งผู้พัฒนาจำเป็นที่จะต้องมีความรู้พื้นฐานในการเขียนภาษา HTML ซึ่งเป็นภาษาโครงสร้างของการแสดงผลในรูปแบบของเว็บไซต์ แต่ในปัจจุบันรูปแบบของเว็บไซต์ได้ถูกพัฒนาให้สามารถประมวลผลข้อมูลต่างๆ ได้จนกลายเป็นที่รู้จักกันในนามของเว็บแอปพลิเคชัน ซึ่งการพัฒนาเว็บแอปพลิเคชันนั้นสามารถทำได้หลายวิธีแต่ในที่นี้จะพัฒนาโดยใช้ Java Servlet เนื่องจาก Servlet มีความสามารถในการเขียน Server Side Code เพื่อติดต่อกับ Client แต่ทั้งนี้การเขียน Servlet นั้นต้องมีทั้ง ภาษา Java และ HTML ร่วมกัน ทำให้การแก้ไขและการพัฒนาเป็นไปได้อย่างยุ่งยาก ในการศึกษาปัญหาพิเศษนี้ ได้อาศัยภาษา JAVA ในการพัฒนาโปรแกรมช่วยเหลือในการพัฒนาเว็บแอปพลิเคชันโดยใช้ Servlet ทำให้การพัฒนาเว็บแอปพลิเคชันสามารถแยกส่วนของภาษา Java และ HTML ออกจากกันได้ โดยอาศัยหลักการของ Template Engine ในการแยกส่วนของ ภาษา Java และ HTML ออกจากกันแล้วให้สามารถทำงานร่วมกันได้ ทั้งนี้การแยกส่วนของภาษาทั้ง 2 ออกจากกันนั้นจะมีผลทำให้ง่ายต่อการแก้ไขและการพัฒนาเว็บแอปพลิเคชัน

Special Project Title	DEVELOPMENT OF EDITOR AND INTERPRETER FOR JAVA SERVLET TEMPLATE ENGINE	
Students	Mr. Thanapat Ratipitchabhorn	42050393
	Miss Vimolwan Chinchalongporn	42050442
	Mr. Anuchard Aussawaviwatpong	42050462
Degree	Bachelor of Science	
Department	Mathematics and Computer of Science, Faculty of Science	
Programme	Computer Science	
Academic Year	2002	
Special Project Advisor	Dr. Kittima Mekabanchakij	

ABSTRACT

Internet plays a very important role in human lifestyle and also has a massive influence on business. All of the data and information is being funneled through the Internet media, otherwise known as 'Web Sites'. The development of a 'Web Site' normally has several processes. These process require a special skill based on HTML programming, which is the 'MarkUp Language' used to display a 'Web Site'. Currently, the development of a web site is much more complex. A web site can process the data transactions besides generating the HTML, known as 'Web Application'. This project, we study the techniques and methods related to the Java language, HTML, and a Template Engine based on Velocity.The software developed in this project has the functionality of editor and interpreter for a simple Template Engine. The result software is intended for convenience of web site development.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่อง การพัฒนาซอฟต์แวร์อิดิเตอร์และอินเตอร์พรีเตอร์สำหรับการแปลภาษารวมเว็บเพจและเซิร์ฟเล็ตนี้จะไม่มีทางสำเร็จลุล่วงไปได้ด้วยดีหากปราศจากคณาจารย์ในคณะวิทยาศาสตร์ที่คอยประสิทธิ์ประสาทวิชาให้ความรู้พื้นฐานต่างๆแก่คณะผู้จัดทำ ดร.กิตติมา เมฆาบัญชากิจ อาจารย์ผู้รับผิดชอบในการให้คำปรึกษากับปัญหาพิเศษ ที่กรุณาให้คำปรึกษาและอำนวยความสะดวกในการทำปัญหาพิเศษนี้

นอกจากนี้แล้วทางคณะผู้จัดทำต้องขอกราบขอบคุณ บิดา มารดา ที่ให้ความอุปการะและช่วยเหลือในการแก้ปัญหาเฉพาะหน้าต่างๆ เพื่อนๆ สาขาวิชาวิทยาการคอมพิวเตอร์ ที่คอยแลกเปลี่ยนความคิดเห็นและให้กำลังใจในการทำงานและเจ้าหน้าที่ดูแลห้องปฏิบัติการคอมพิวเตอร์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ ที่คอยอำนวยความสะดวกในการทำงาน และความช่วยเหลืออย่างเต็มความสามารถ ในการทำปัญหาพิเศษนี้

คณะผู้จัดทำ

มีนาคม 2546

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

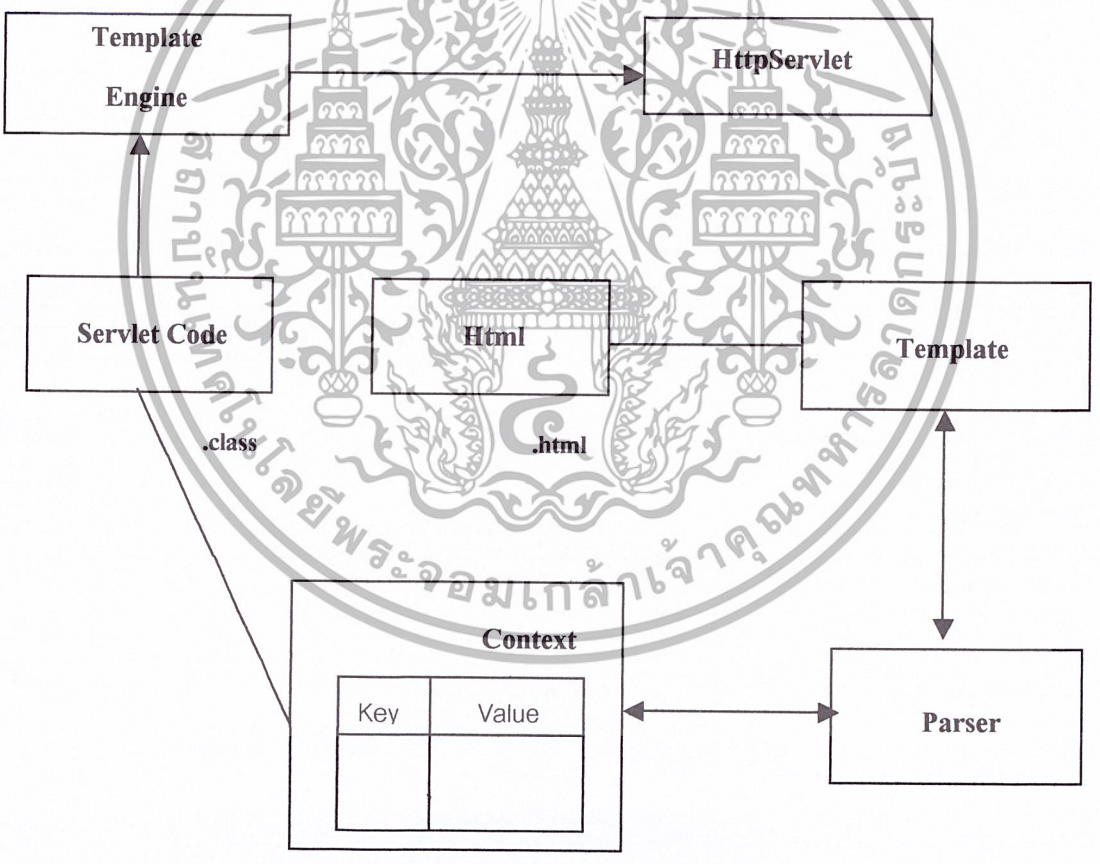
สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการพัฒนา Web Application โดยใช้ Java Servlet เป็นที่นิยมอย่างแพร่หลาย เนื่องจาก Servlet มีความสามารถในการเขียน Server Side Code เพื่อติดต่อกับ Client แต่ทั้งนี้ การเขียน Servlet ก็ต้องมีทั้ง ภาษา Java และ HTML ร่วมกัน ทำให้การแก้ไข และการพัฒนาเป็นไปได้อย่างยุ่งยาก เพราะผู้ที่จะมาทำการปรับปรุงต้องเข้าใจทั้งสองภาษา ปัญหานี้สามารถแก้ไขโดยการใช้ Template Engine ซึ่งจะแยกส่วนของโปรแกรม Java และ HTML ออกจากกันตามแผนผัง ดังนี้



รูปที่ 1.1 แผนผังการทำงานของ Template Engine

จากรูปที่ 1.1 Template Engine จะทำหน้าที่ 2 อย่างคือ ตีความ Servlet และสั่ง Parser ทำงาน โดยจะเห็นได้ว่า Web Designer (HTML) สามารถทำงานไปพร้อมๆ กับProgrammer (Java) ซึ่งจะเป็นไปตามหลักของ MVC (Model-View-Controller) ดังนั้น Web Design สามารถพัฒนา Web Page ซึ่งอาจเขียนจาก Text editor ทั่วไป หรือ HTML editor (FrontPage หรือ Dreamweaver) โดยไม่จำเป็นต้องรู้จักหรือเข้าใจภาษา Java เลย

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาการทำงานของ Template Engine ตามแนวคิด MVC

1.2.2 เพื่อศึกษาการเขียน Application โดยใช้ภาษา Java ที่ประมวลผลฝั่ง Server

1.2.3 เพื่อศึกษาการทำงานของ Servlets ซึ่งสามารถทำงานฝั่ง Server เพื่อติดต่อกับ Client

1.3 ขอบเขตการศึกษา

1.3.1 พัฒนาซอฟต์แวร์ Template Engine สำหรับแปลคำสั่งและ generate code ให้เป็น Java Servlets ภายใต้กรอบ MVC

1.3.2 การพัฒนาซอฟต์แวร์จะใช้เครื่องมือ

- J2SE Version 1.4.0
- Velocity Version 1.1
- Gel 0.79k2
- Apache Tomcat 4.0.4 Server
- WS – FTP Pro Version 7

1.4 ขั้นตอนการศึกษา

1.4.1 ศึกษาการทำงานของภาษา Java ในส่วนที่เกี่ยวข้องกับการทำงานบนระบบเครือข่ายอินเตอร์เน็ต

1.4.2 ศึกษาการทำงานของHTTP Protocol ที่ทำงานอยู่บนเครือข่ายอินเตอร์เน็ต

1.4.3 ศึกษาโครงสร้างและการเขียน HTML

1.4.4 ศึกษาการทำงานและการใช้งาน Java Servlet

1.4.5 ศึกษาโครงสร้างและการใช้งาน Velocity

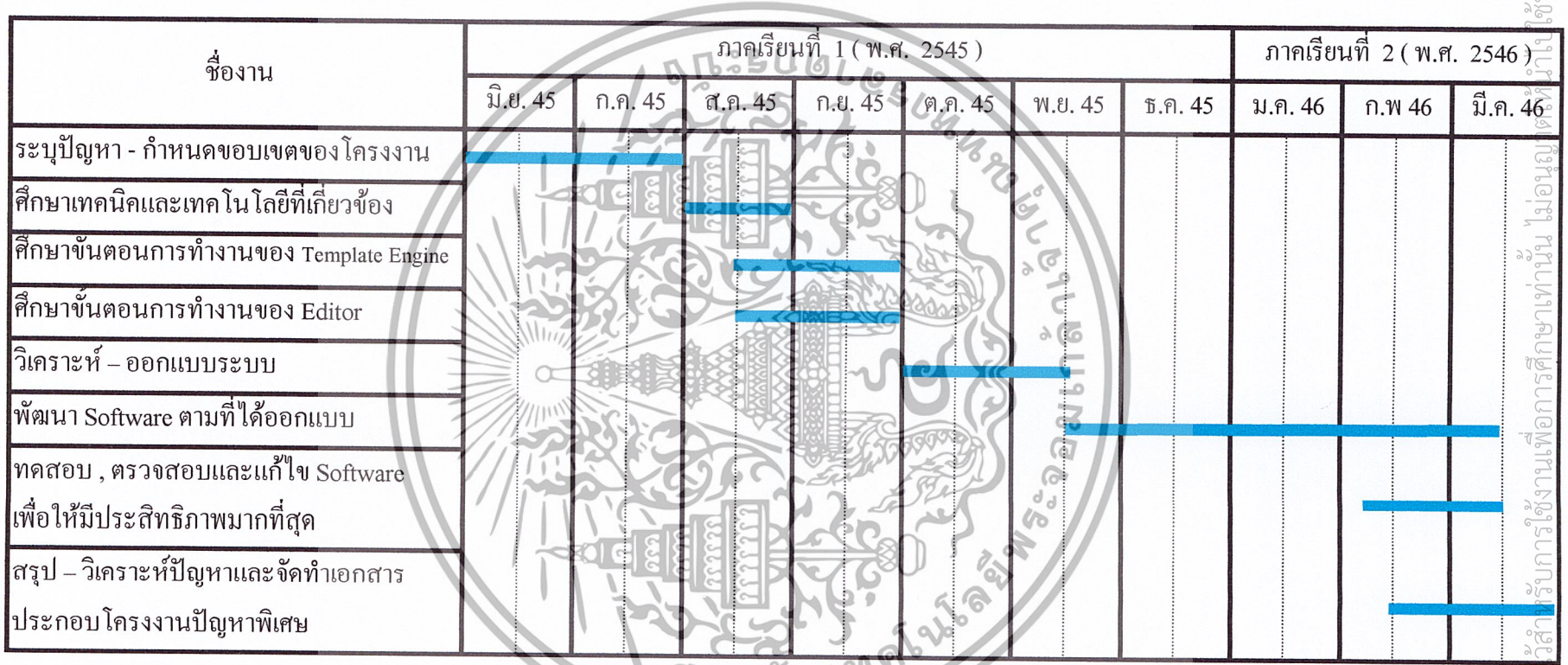
1.4.6 ศึกษาวิธีการทำงานของ Parser

1.4.7 พัฒนาซอฟต์แวร์ Template Engine โดยใช้ภาษา Java พร้อมทดสอบการทำงาน

1.4.8 รวบรวมข้อมูลในการทำปัญหาพิเศษทั้งหมดมาดำเนินการจัดทำเป็นเอกสาร

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 ตารางแสดงแผนการดำเนินงาน



บทที่2
ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

2.1 แนะนำภาษาจาวา

จาวานั้นนอกจากจะเป็นภาษาสำหรับเขียนโปรแกรม (Application Programming) ซึ่งเป็นลักษณะของโปรแกรมเชิงวัตถุ (Object Oriented Programming) แล้วยังสามารถนำมาใช้งานได้บน Web อีกด้วย ซึ่งเรียกว่า " Web Applet " หรือ " Applet " ด้วยเหตุผลง่ายๆดังกล่าวนี้ จึงทำให้จาวาเป็นโปรแกรมที่ได้รับความนิยม และถูกพัฒนาขึ้นอย่างต่อเนื่องจนถึงปัจจุบัน

2.1.1 ชนิดของโปรแกรมจาวา

ปัจจุบันโปรแกรม Java แบ่งออกเป็น 2 ชนิด คือ Java Application และ Java Applets ทั้ง 2 ชนิดนำมาใช้งานแตกต่างกันดังนี้

Java Application

เป็นการนำ Java มาเขียนเป็นโปรแกรมที่สามารถนำมาใช้งานได้อย่างอิสระ (Stand Alone Program) เหมือนกับการเขียนโปรแกรมภาษาระดับสูงอื่นๆ เช่น C++ , Pascal , Cobol ทั้งนี้สามารถนำ Application ไปใช้งานที่คอมพิวเตอร์ต่างแพลตฟอร์ม ไม่ว่าจะเป็น PC , Macintosh หรืออื่นๆ

Java Applets

เป็นการนำ Java มาเขียนเป็นโปรแกรมเช่นเดียวกัน แต่ไม่สามารถเรียกใช้ตามลำพังเหมือนกับ Applications แต่จะต้องนำไปใส่ไว้ในเอกสาร HTML (เพื่อทำให้ Web Page ทำงานได้ดียิ่งขึ้น) แล้วใช้โปรแกรม Web Browser (เช่น Netscape , Internet Explorer) หรือใช้ Utilities ของ Java ชื่อ appletviewer เพื่อเรียกผลลัพธ์ก็ได้

2.1.2 คุณสมบัติของภาษาจาวา

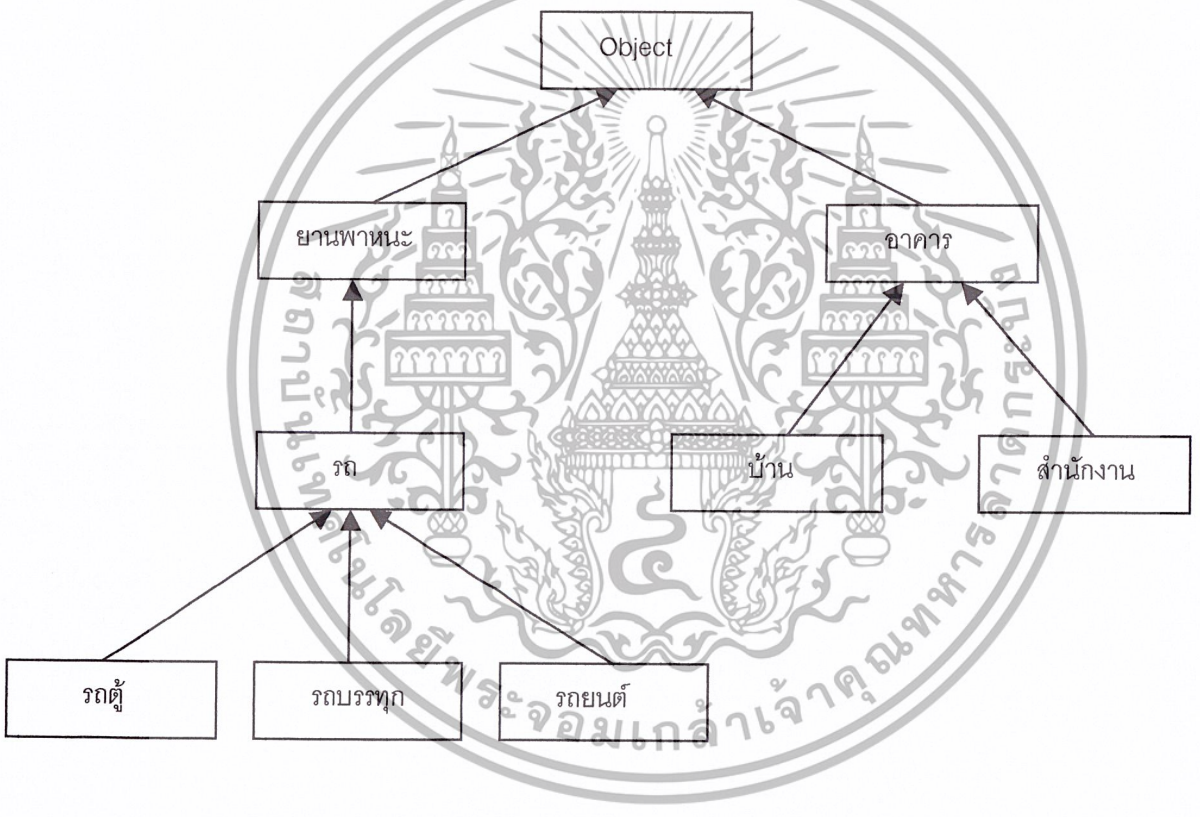
2.1.2.1 จาวาเป็นภาษาที่เรียนรู้ได้ง่าย

จาวามีความคล้ายคลึงกับภาษา C และ C++ มากแต่ง่ายกว่า เนื่องจากตัดคุณสมบัติบางอย่างที่ไม่จำเป็นทิ้ง เช่น จาวาไม่มี Operator Overloading , Header File , การดำเนินการกับตัวชี้ (Pointer Operation) และ อื่นๆ ทำให้ความซับซ้อนของภาษาลดลง รวมทั้งผู้เขียนโปรแกรมส่วนมากมักมีความคุ้นเคยกับภาษา C หรือ ภาษา C++ จึงทำให้ง่ายต่อการเรียนรู้ภาษาจาวา

2.1.2.2 จาวาเป็นภาษาที่เป็นเชิงวัตถุ (Object-Oriented Language)

จาวาเป็นภาษาที่เป็นเชิงวัตถุ กล่าวคือ มีการออกแบบให้มีโครงสร้างเป็นเชิงวัตถุ เพื่อให้การพัฒนาโปรแกรมเป็นไปได้ง่ายและรวดเร็ว และสามารถนำส่วนต่างๆมาใช้ใหม่โดยไม่ต้องแก้ไขหรือมีการแก้ไขที่น้อยที่สุด

ในภาษาจาวา เกือบทุกสิ่งทุกอย่างจะกำหนดในรูปเชิงวัตถุ ยกเว้นเฉพาะตัวแปรบางชนิด เช่น ตัวเลข และค่าทางตรรกะ จาวามีการจำแนกออกเป็นคลาส แต่ละคลาสจะมีลักษณะเฉพาะของเชิงวัตถุนั้น และมีเมธตอดซึ่งแสดงถึงพฤติกรรมเฉพาะของเชิงวัตถุนั้น คลาสสามารถถ่ายทอดคุณสมบัติจากคลาสอื่นได้ โดยที่คลาสเริ่มต้นจะเป็นคลาส Object เสมอ



รูปที่ 2.1 การสร้างคลาสยานพาหนะ และอาคาร

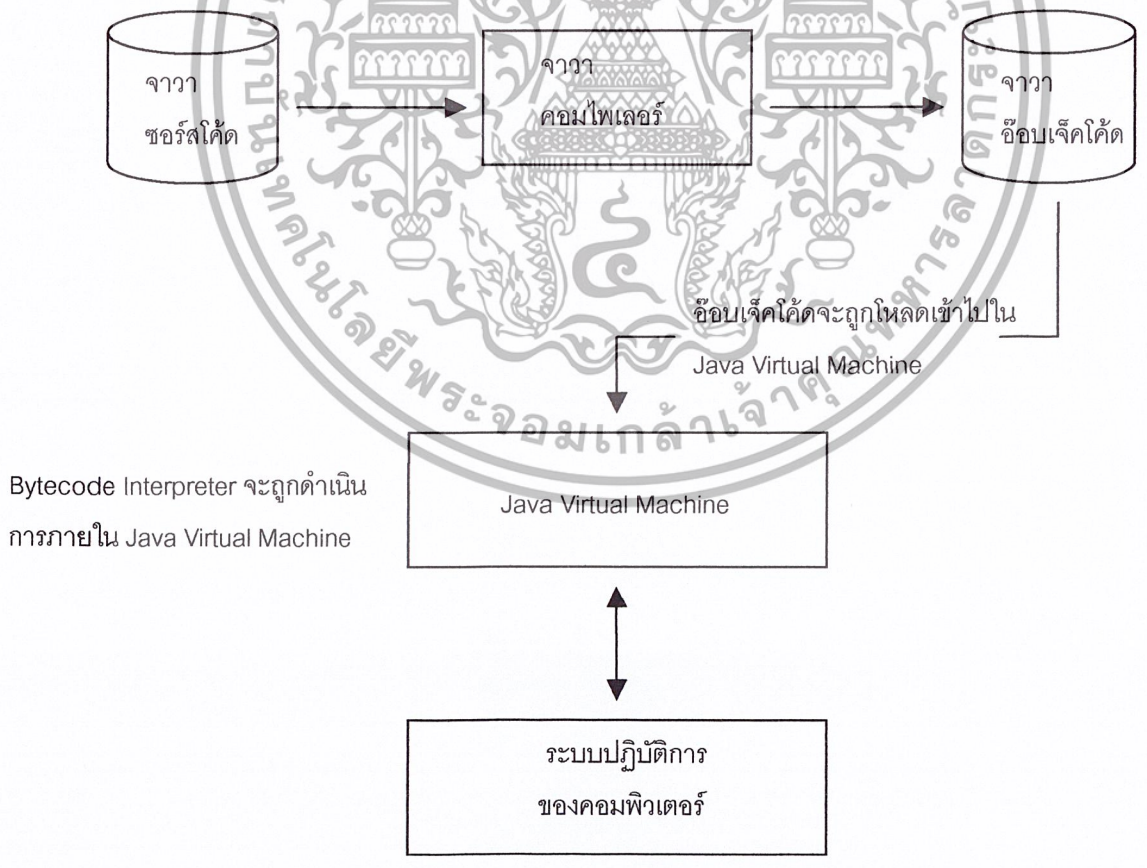
จากรูปที่ 2.1 จะพบว่าคลาสรถบรรทุกมีการถ่ายทอดคุณสมบัติมาจากคลาสของรถ ซึ่งรถถ่ายทอดคุณสมบัติมาจากคลาสยานพาหนะ และยานพาหนะถ่ายทอดคุณสมบัติมาจาก Object อีกทอดหนึ่ง

จาวาเป็นภาษาที่มีการถ่ายทอดแบบ Single Inheritance กล่าวคือ แต่ละคลาสจะถ่ายทอดคุณสมบัติจากคลาสใดคลาสหนึ่งเพียงคลาสเดียว ภาษาคอมพิวเตอร์บางภาษาอนุญาตให้มี Multiple-Inheritance กล่าวคือ แต่ละคลาสจะสามารถถ่ายทอดคุณสมบัติจากคลาสอื่นได้มากกว่าหนึ่งคลาส ซึ่งอาจก่อให้เกิดความสับสนได้ง่าย และทำให้ภาษาซับซ้อนโดยไม่จำเป็น

2.1.2.3 จาวาเป็นภาษาที่ทำงานได้กับคอมพิวเตอร์ทุกระบบปฏิบัติการ

ก่อนที่จะรันโปรแกรมจาวา โปรแกรมจาวาจะถูกคอมไพล์ และแปลงสภาพเป็นไบต์โค้ดก่อน ไบต์โค้ดมีสภาพคล้ายภาษาเครื่อง ดังนั้น โปรแกรมจาวาจึงมีประสิทธิภาพสูง เนื่องจากไบต์โค้ดไม่ขึ้นกับระบบปฏิบัติการของเครื่องคอมพิวเตอร์ ทำให้โปรแกรมจาวาสามารถปฏิบัติการได้บนคอมพิวเตอร์ทุกระบบ โดยไม่ต้องทำการคอมไพล์โปรแกรมใหม่

โปรแกรมจาวาสามารถทำงานได้กับคอมพิวเตอร์ทุกระบบปฏิบัติการ ไม่ว่าจะเป็น PC ที่ทำงานบน Window95 หรือบนเครื่อง Mac หรือบน Unix ทั้งนี้เนื่องจากโปรแกรมจาวาจะทำงานบน Java Virtual Machine ดังรูปที่ 2.2



รูปที่ 2.2 ลักษณะการทำงานของโปรแกรมจาวาบนเครื่องคอมพิวเตอร์

โปรแกรมจาวาจะถูกแปลงโดยจาวาคอมไพเลอร์ ไปเป็นไบต์โค้ด ซึ่งไบต์โค้ดเป็นคำสั่งภาษาเครื่องสำหรับ Java Virtual Machine เมื่อโปรแกรมจาวาทำงาน ตัวแปลภาษาจาวา (Java Interpreter) จะทำการอ่านไบต์โค้ด และดำเนินการตรวจสอบว่ามีสิ่งใดแอบแฝง หรือเป็นอันตรายหรือไม่ ถ้าไม่ก็จะทำงานตามคำสั่งในโปรแกรมภายใน Java Virtual Machine ตัวแปลภาษาจาวาสามารถทำงานเดี่ยวๆ หรือโดยเป็นส่วนหนึ่งของเว็บบราวเซอร์ก็ได้

ภาษาจาวาเป็นมาตรฐานเดียวกันบนทุกๆเครื่อง เช่น integer จะเป็น 32 บิตเสมอ และ long จะเป็น 64 บิตเสมอ ซึ่งแตกต่างจากภาษา C และ C++ ซึ่งคอมไพเลอร์ และส่วนประกอบอื่นๆ จะแตกต่างกันเล็กน้อยขึ้นอยู่กับเครื่อง และระบบปฏิบัติการ ทำให้การย้ายโปรแกรมจากระบบหนึ่งไปสู่อีกระบบหนึ่งเป็นเรื่องยุ่งยาก และต้องคอมไพล์โปรแกรมใหม่ แต่สำหรับจาวาเป็นเรื่องง่าย และไม่ต้องคอมไพล์โปรแกรมใหม่

ระบบของจาวายังตระเตรียมคลังโปรแกรม (Library) ของคลาสต่างๆ สำหรับระบบปฏิบัติการต่างๆ โดยการใช้คลังโปรแกรมเหล่านี้ จะทำให้โปรแกรมจาวาสามารถทำงานได้บนทุกระบบปฏิบัติการที่รองรับจาวา

2.1.2.4 จาวาเป็นมัลติเทรด (Multi-thread)

ระบบคอมพิวเตอร์สมัยใหม่ เช่น ยูนิกซ์ และ วินโดว์ 2000/NT มีการทำงานระบบหลายภารกิจ (Multi-tasking) ซึ่งหมายความว่า คอมพิวเตอร์สามารถทำงานได้มากกว่า 1 งานในขณะเดียวกัน จาวาเป็นภาษาที่ถูกออกแบบให้ทำงานระบบหลายภารกิจ

โปรแกรมจาวาสามารถมีได้มากกว่า 1 เทรด ของการทำงาน เช่น เทรดหนึ่งอาจทำงานคำนวณบางอย่างที่ใช้เวลานาน และอีกเทรดหนึ่งก็ทำการติดต่อกับผู้ใช้ ทำให้ผู้ใช้ไม่ต้องหยุดทำงานเพื่อรอให้จาวาเสร็จงานที่ใช้เวลาในการทำงานนานก่อน

โปรแกรมในลักษณะมัลติเทรด โดยทั่วไปจะยุ่งยากซับซ้อน เนื่องจากเหตุการณ์หลายเหตุการณ์สามารถเกิดขึ้นพร้อมๆกัน แต่จาวาได้ตระเตรียมระบบเพื่อจัดการในเหตุการณ์พร้อมกัน ช่วยให้การเขียนโปรแกรมเป็นไปได้โดยง่าย

2.1.2.5 จาวามีระบบเก็บขยะ (Garbage Collection)

ผู้เขียนโปรแกรมซึ่งเคยเขียนโปรแกรมโดยใช้ภาษา C และภาษา C++ จะต้องระมัดระวังในการอ้างอิงถึงหน่วยความจำ (Memory) เมื่อไม่มีการใช้งานหน่วยความจำส่วนใด จะต้องทำการคืนหน่วยความจำนั้นๆกลับคืนสู่ระบบ เพื่อให้สามารถเรียกใช้ได้ใหม่ ในโครงการใหญ่ๆการกระทำดังกล่าวเป็นสิ่งที่กระทำได้ยาก และมักเป็นสาเหตุของข้อผิดพลาดต่างๆ

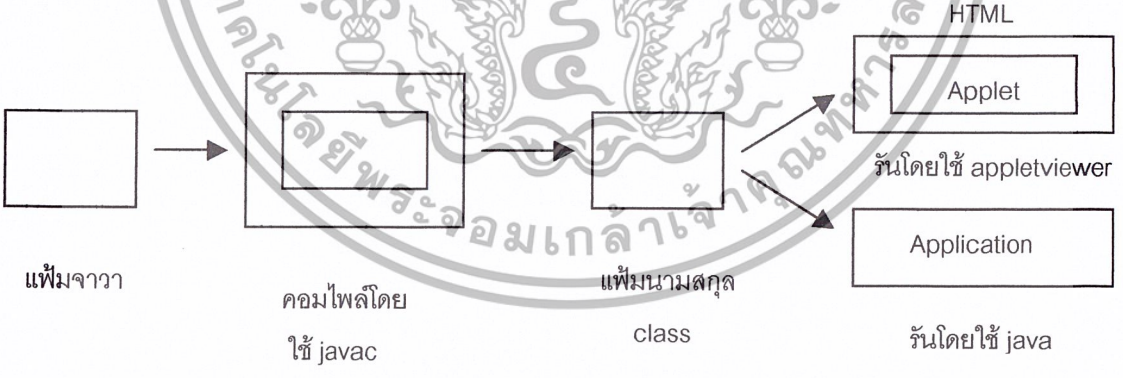
ผู้เขียนโปรแกรมภาษาจาวาไม่จำเป็นต้องกังวลเกี่ยวกับการจัดการกับหน่วยความจำของระบบ จาวาจะมีการสร้างระบบการเคลียร์ Object ที่ไม่มีการอ้างอิงในขณะนั้น ซึ่งจะทำการตรวจเช็คหน่วยความจำโดยอัตโนมัติ และทำการคืนหน่วยความจำที่ไม่ได้ใช้สู่ระบบ ซึ่งทำให้การเขียนโปรแกรมด้วยจาวาง่ายขึ้น

2.1.3 แนะนำ Java Development Kit (JDK)

ปัจจุบันมีซอฟต์แวร์ที่ใช้ในการสร้างโปรแกรมภาษาจาวาอยู่หลายตัวเช่น JDK (Java Development Kit) ของบริษัท Sun Microsoft System, Visual Caf้ ของ WebGain , JBuilder , Visual Age For Java ของ IBM , Microsoft Visual J++ และอื่นๆ แต่ที่ได้รับความนิยมมากๆได้แก่ JDK , Visual Caf้ , Visual Age For Java

JDK เป็นซอฟต์แวร์ที่สร้างขึ้นเพื่ออำนวยความสะดวกในการพัฒนาโปรแกรมจาวา JDK รวบรวมซอฟต์แวร์และคู่มือการใช้งานที่จำเป็นในการสร้างโปรแกรมจาวา และจาวาแอปเพล็ต JDK ถูกสร้างขึ้นโดยบริษัทย่อยของ Sun ชื่อ Javasoft ซึ่ง JDK ได้รับความนิยมเนื่องจากผู้ใช้สามารถดาวน์โหลดได้ฟรี และมีขนาดเล็ก ปัจจุบันเป็น Version 1.4.0

JDK จะครอบคลุมโปรแกรมที่สามารถเรียกใช้ได้ทั้งในวินโดว์ และยูนิกซ์ เพื่อใช้ในการคอมไพล์ , การรัน และตรวจสอบความถูกต้องของโปรแกรมจาวา



รูปที่ 2.3 ขั้นตอนการคอมไพล์ และการรันโปรแกรมจาวา

จากรูปที่ 2.3 แฟ้มนามสกุล java จะถูกคอมไพล์โดยใช้คำสั่ง javac ซึ่งเราจะได้แฟ้มนามสกุล class ถ้าเป็นโปรแกรมจาวาแอปเพล็ต จะต้องบรรจุแฟ้มนามสกุล class นั้นไว้ในแฟ้ม HTML เสียก่อน แล้วจึงใช้คำสั่ง appletviewer เพื่อรันโปรแกรม หรือใช้เว็บบราวเซอร์ เช่น Netscape หรือ Internet Explorer ในกรณีของโปรแกรมประยุกต์ สามารถรันโดยใช้คำสั่ง java ได้ทันที

2.1.4 การสร้างโปรแกรมภาษาจาวา

(1) การพัฒนาโปรแกรม

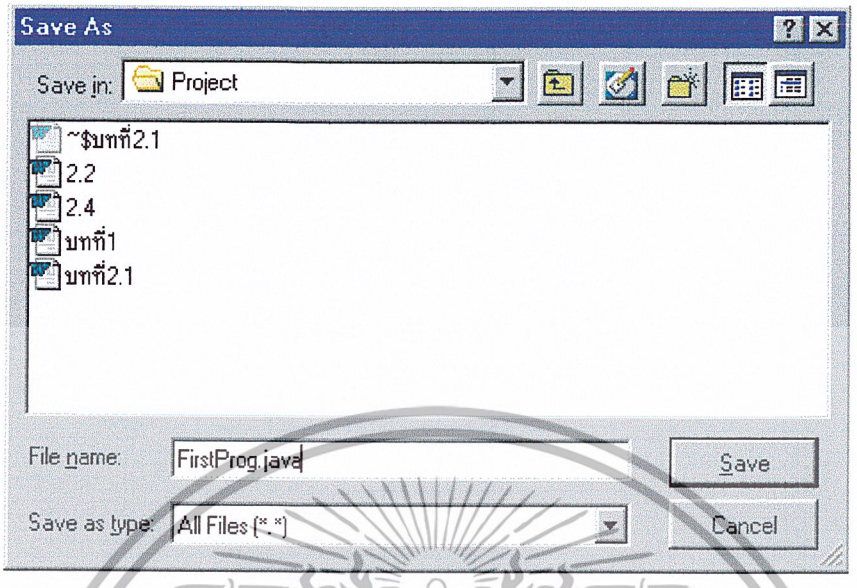
เริ่มจากการใช้อิดิเตอร์ (Editor) ในการเขียนโปรแกรม ใน Unix เราสามารถใช้ vi หรือ emacs ใน Dos เราสามารถใช้คำสั่ง edit และในวินโดว์เราสามารถใช้ Notepad หรือ อิดิเตอร์ที่สร้างขึ้นเฉพาะเพื่อใช้ในการเขียนโปรแกรมจาวา เช่นอิดิเตอร์ใน Microsoft Visual J++ เป็นต้น

การใช้ Notepad ในการเขียนโปรแกรมทำได้ดังนี้

1. เลื่อนเมาส์ไปกดที่เมนู Start
2. เลื่อนเมาส์ไปกดที่ program เลือกNotepad จากเมนู Accessories
3. พิมพ์โปรแกรมลงบนวินโดว์ของ Notepad ดังรูปที่ 2.4

รูปที่ 2.4 ตัวอย่างโปรแกรมประยุกต์ FirstProg.java

หลังจากเขียนโปรแกรมเสร็จเรียบร้อยแล้วให้บันทึกลงแฟ้มซึ่งมีชื่อเดียวกับคลาส (Class) ในโปรแกรมและระบุนามสกุล java เสมอ พร้อมทั้งระบุชนิดของแฟ้มเป็น All Files(*.*) ในตัวอย่างนี้ให้ชื่อแฟ้มว่า FirstProg.java



รูปที่ 2.5 แสดงการบันทึกแฟ้ม FirstProg.java

เมื่อบันทึกโปรแกรมเสร็จ เราจะต้องทำการคอมไพล์โปรแกรมโดยใช้คำสั่ง javac ใน command line ของ Dos ดังนั้นจึงต้องทำการเปลี่ยนจากการทำงานบนวินโดว์มาทำงานบน Dos ขั้นตอนการคอมไพล์โปรแกรมเป็นดังนี้

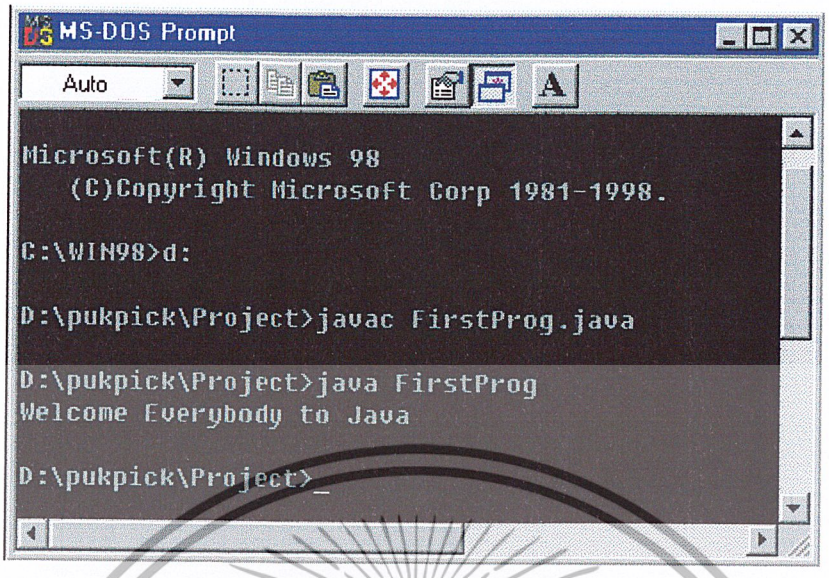
1. เลื่อนเมาส์ไปกดที่เมนู Start
2. เลือก Program จากเมนู Start
3. จากเมนู Program เลือก MS-DOS Prompt
4. เปลี่ยนไดเรกเทอรีไปเป็นไดเรกเทอรีที่บรรจุซอร์สโค้ด
5. คอมไพล์ซอร์สโค้ดโดยใช้คำสั่ง javac FirstProg.java

(2) การรันโปรแกรมจาวาที่พัฒนา

หลังจากคอมไพล์โปรแกรม เราจะได้แฟ้มนามสกุล class ซึ่งเราจะสามารถรันโปรแกรมได้ทันทีโดยใช้คำสั่ง

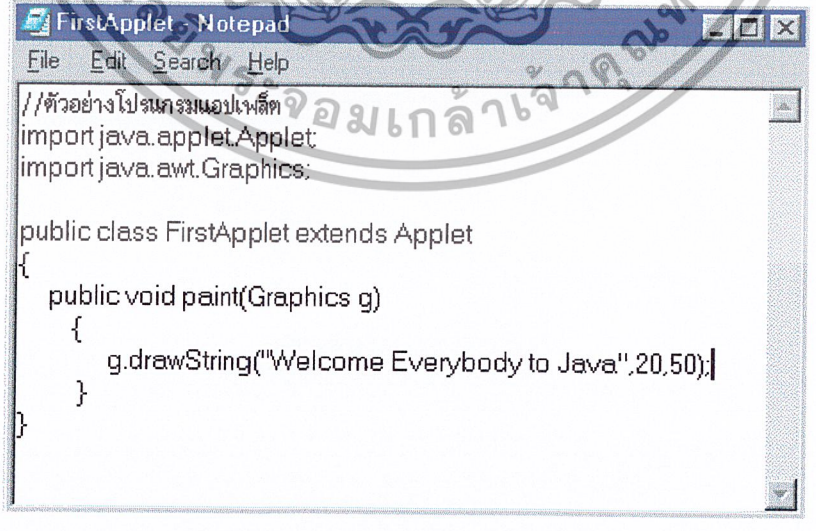
java FirstProg บน Dos กรณีที่เป็น Application

ในกรณีนี้จะพบว่าเราไม่จำเป็นต้องระบุนามสกุลของแฟ้ม ผลลัพธ์ของโปรแกรมจะแสดงต่อท้ายดังรูปที่ 2.6

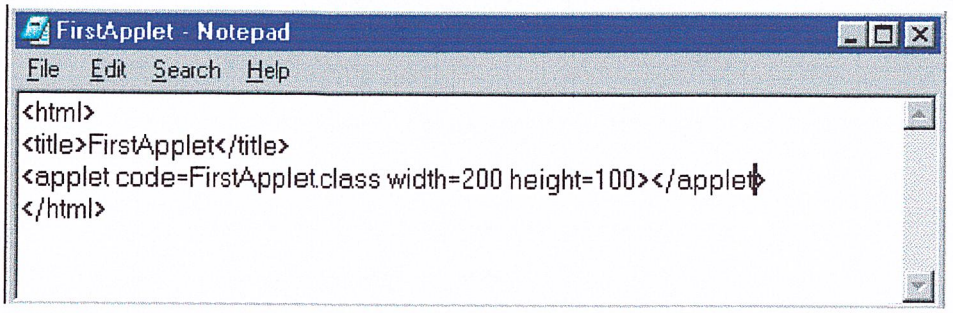


รูปที่ 2.6 การคอมไพล์ และรันโปรแกรม FirstProg.java

ในกรณีของจาวาแอปเพล็ต หลังจากการคอมไพล์โปรแกรมแล้วเราไม่สามารถรันได้ทันที เราจะต้องนำแอปเพล็ตนั้นๆมาบรรจุลงในแฟ้ม HTML เสียก่อน ในการนำแอปเพล็ตมาบรรจุลงแฟ้ม HTML จะใช้ Tag <APPLET> แล้วใส่ชื่อของแอฟเพล็ตแฟ้มลงไปดังรูปที่ 2.8 หลังจากนั้นทำการบันทึกลงแฟ้มนามสกุล HTML (ในกรณีนี้สามารถตั้งชื่อแฟ้มเป็นชื่ออะไรก็ได้ ตามที่เราต้องการ)



รูปที่ 2.7 ตัวอย่างแอปเพล็ต FirstApplet.java



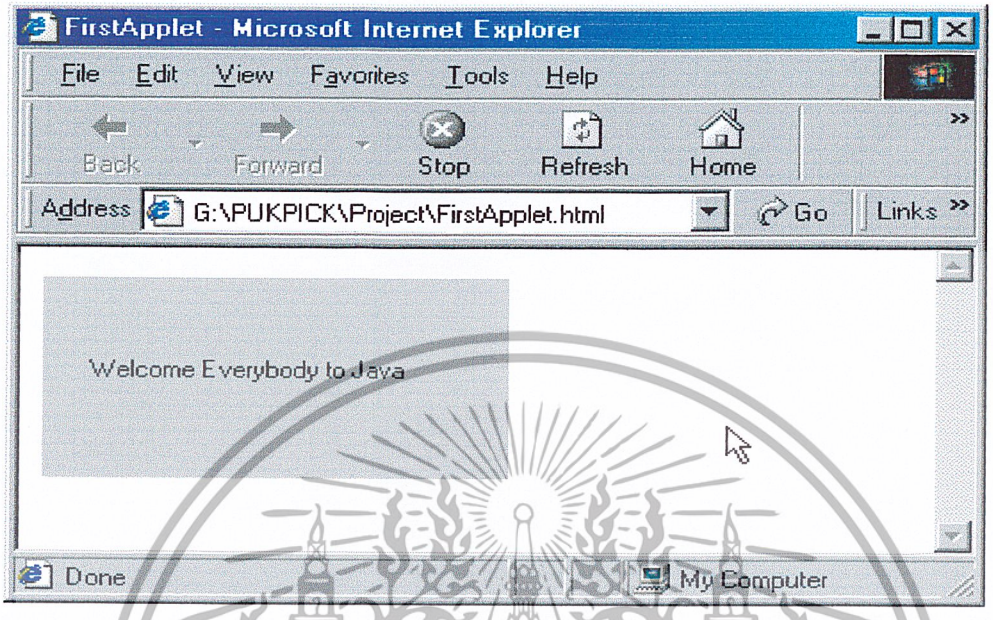
รูปที่ 2.8 การบรรจุแอปเพล็ตลงไปใน HTML

การรันจาวาแอปเพล็ตมี 2 วิธี

1.ใช้ applet viewver วิธีนี้สามารถรันโปรแกรมโดยใช้คำสั่ง appletviewer FirstApplet.html บน dos และได้ผลลัพธ์ ดังรูปที่ 2.9

รูปที่ 2.9 การรันโปรแกรมโดยใช้ applet viewer

2. การใช้เว็บบราวเซอร์



รูปที่ 2.10 การรันโปรแกรมโดยใช้ Internet Explorer

2.2 โพรโตคอล HTTP (Hypertext Transfer Protocol)

HTTP มาจากคำว่า Hypertext Transfer Protocol ซึ่งเป็น Protocol ที่ใช้ในการส่งข้อมูลต่าง ๆ ในโลกของ World Wide Web ข้อมูลต่างๆ เหล่านี้โดยทั่วไปมักจะถูกเรียกว่า Resource โดย Resource เหล่านี้ อาจจะเป็นไฟล์ เช่น HTML ไฟล์, image ไฟล์ หรือคำสั่งต่างๆ (Query String) เช่น คำสั่งที่ส่งไปที่ cgi โปรแกรมหรืออาจจะเป็น binary stream ในกรณีของการ download/upload ไฟล์ หรืออาจจะเป็นสิ่งอื่นๆ อีกมากมายตามแต่จะกำหนดขึ้น

HTTP เป็น Protocol ที่อยู่ในส่วนของ Application Layer ใน protocol stack โดยข้อมูลต่างๆ จาก Layer นี้จะถูกส่งผ่านไปยัง Layer อื่นๆ ที่ต่ำกว่าซึ่งส่วนหนึ่งในนั้นคือ TCP/IP protocol นั่นเอง

HTTP เป็น network protocol ที่ใช้หลักการของ client-server model ในการติดต่อสื่อสาร ซึ่งหลักการทำงานอย่างคร่าวๆ มีดังนี้

- HTTP Client จะทำการสร้างคอนเนคชั่นไปหา HTTP Server ซึ่งโดยทั่วไปจะผ่านทางsocket ของ TCP/IP
- หลังจากนั้น HTTP Client จะทำการส่งคำสั่ง (request) ซึ่งอยู่ในรูปของ message ไปให้ HTTP Server เพื่อทวงถามถึง resource ที่ต้องการ
- HTTP Server จะทำการตีความคำสั่งที่ได้และส่งผล (response) ซึ่งเป็น resource ที่ HTTP Client ต้องการกลับมา (ผลที่ส่งกลับมาจะเป็นลักษณะของ message คล้ายกับ request ของ HTTP Client ที่ส่งมาให้ HTTP Server)
- หลังจากที่การส่ง response เสร็จสิ้น resource อื่น ๆ HTTP Server จะทำการปิดคอนเนคชั่น ที่มาจาก HTTP Client
- ในกรณีที่ HTTP Client ต้องการ resource อื่นๆ HTTP Client จะต้องทำการสร้างคอนเนคชั่นใหม่ และส่งคำสั่งไปหา HTTP Server อีกครั้ง

จากหลักการข้างต้นจะเห็นว่าการติดต่อสื่อสารระหว่าง Client และ Server จะเป็นลักษณะครั้งต่อครั้ง ในทาง network เราเรียกการติดต่อสื่อสารแบบนี้ว่า Stateless Protocol จริง ๆ แล้ว Web Browser ก็คือ HTTP client นั่นเอง เหตุผลคือเราใช้ Web Browser เป็นตัวส่ง request ไปที่ Web Server (หรือ HTTP server) เพื่อจะรับ resource ที่ต้องการกลับมาโดยใช้ HTTP protocol. โดยทั่วไป resource จะกระจายอยู่ตาม network nodes ต่าง ๆ ทั่วโลก สิ่งหนึ่งที่จะขาดไม่ได้ในการอ้างถึง resource เหล่านี้คือ URL (Universal Resource Locator) URL คือตัวที่ใช้ชี้ถึงแหล่งที่อยู่ของ resource ว่าอยู่ที่ไหน หลายคนอาจเข้าใจผิดคิดว่า URL ถูกใช้สำหรับ HTTP อย่างเดียวแต่จริง ๆ แล้ว URL สามารถใช้เพื่ออ้างถึง resource อะไรก็ได้ โดยใช้ protocol อะไรก็ได้ ยกตัวอย่างเช่น

http://www.jarticles.com/index.html

เป็น URL ของ HTTP protocol เพื่อใช้โหลดไฟล์ html

http://www.jarticles.com:80/coffee.jpg

เป็น URL ของ HTTP protocol เพื่อใช้โหลดไฟล์ image

http://www.jarticles.com/cgi-bin/sendMail.pl

เป็น URL ของ HTTP protocol เพื่อใช้โหลดค่าจากการประมวลผลของ Cgi โปรแกรมชื่อ sendMail.pl

ftp://ftp.jarticles.com/jspTutorial.doc

เป็น URL ของ FTP protocol เพื่อใช้ดาวโหลดไฟล์ document

จากตัวอย่างข้างต้น เราจะเห็นว่าในหนึ่ง URL จะประกอบไปด้วยชื่อของ protocol, ชื่อของ server, portที่ใช้ และ resource ที่ต้องการ ดังรูปแบบข้างล่าง

http://www.jarticles.com:80/index.html

------ ----------

| | | |

Protocol Hostname Port Resource

2.2.1 HTTP Structure

จากหลักการทำงานของ HTTP ข้างต้น เราจะเห็นว่าสำหรับการติดต่อสื่อสารระหว่าง Client กับ Server แล้ว ตัว message จะเป็นตัวกลางที่ใช้ในการติดต่อสื่อสารเสมอ และเพื่อให้ง่ายต่อการใช้งาน รูปแบบของ message ที่ใช้ใน request หรือ response จึงมีลักษณะคล้าย ๆ กัน โดยจะใช้ text เป็นหลัก ซึ่งโครงสร้างของ message จะประกอบไปด้วย

- บรรทัดเริ่มต้น (initial line)
- Header
- บรรทัดว่าง (a blank line) ซึ่งก็คือ CRLF หรือการเว้นหนึ่งบรรทัด (วิธีหนึ่งที่ทำได้คือการกด Enter นั่นเอง)
- message body ซึ่งอาจจะใช้บรรจุไฟล์, คำสั่ง (Query String) หรืออาจจะเป็น output ที่มาจาก server โดยส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับจุดประสงค์ของการใช้งาน

รูปแบบโดยทั่วไปของ message จะเป็น

<initial line>

Header1: value1

Header2: value2

Header3: value3

[blank line]

<message body, optional .....>

ตัวอย่างง่าย ๆ ของ message ที่ Client ใช้อาจจะเป็น

GET /images/coffee.jpg <-- initial line

From: soup@jarticles.com <-- header

User-Agent: Mozilla/4.72 <-- header

[blank line] <-- บรรทัดว่าง

หรือ

POST /servlet/searchEngine HTTP/1.0 <-- initial line

From: webmaster@jarticles.com <-- header

User-Agent: Mozilla/4.72 <-- header

[blank line] <-- บรรทัดว่าง

keyword=java&topic=jsp <-- message body

ตัวอย่างง่าย ๆ ของ message ที่ Server ใช้อาจจะเป็น

HTTP 1.0 200 OK <-- initial line

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

Server: Apache/1.3.12(Unix) (Red Hat/Linux) PHP/3.0.15 mod_perl/1.21 <-- header

MIME-version: 1.0 <-- header

Content-type: text/html <-- header

Content-length: 115 <-- header

[blank line] <-- บรรทัดว่าง

<HTML><HEAD><TITLE>HTTTP Tutorial</TITLE></HEAD> <-- message body

<BODY>This is a tutorial, but please visit me again..</BODY> <-- message body (cont.)

<HTML> <-- message body (cont.)

2.2.2 Initial Request Line

initial line ของ request จะแตกต่างจาก initial line ของ response เล็กน้อย โดย initial line ของ request จะมีสามส่วนคือ

1. ชื่อของ method เช่น GET, POST, HEAD, TRACE
2. local path ของ resource ที่ client ต้องการ
3. เวอร์ชั่นของ HTTP/x.x ที่ HTTP Client ใช้

สามส่วนนี้จะประกอบกันเป็น initial line โดยแต่ละส่วนจะถูกแยกออกจากกันโดยใช้ช่องว่าง (space) ดังตัวอย่างข้างล่าง

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

ตัวอย่างนี้ใช้ method ที่ชื่อ GET เพื่อขอ resource ชื่อ path/to/file/index.html โดยใช้ HTTP/1.0 protocol

สาเหตุที่ต้องใช้ local path?

ก่อนที่ HTTP Client จะส่ง request ไปที่ HTTP Server, HTTP Client จะสร้างคอนเนคชั่นขึ้นมาอันหนึ่งก่อน โดยใช้ socket การสร้าง socket จะต้องทำการกำหนดชื่อของ host ที่จะติดต่อ ยกตัวอย่างเช่น ถ้า url เป็น http://www.jarticles.com/path/to/file/index.html ชื่อของ host ก็จะเป็น www.jarticles.com ดังนั้นการกำหนดเพียง local path เพื่อบ่งบอกถึง resource ที่ต้องการในส่วนของ initial request line ซึ่งก็คือ /path/to/file/index.html ก็เพียงพอแล้ว

กฎและเงื่อนไข

1) ชื่อของ method จะต้องใช้ตัวใหญ่เสมอ

2) เวอร์ชั่นของ HTTP จะอยู่ในรูป HTTP/x.x และจะต้องเป็นตัวใหญ่เสมอ

2.2.3 Initial Response Line

โดยทั่วไป initial response line มักจะเรียกว่า status line ประกอบไปด้วยสามส่วนย่อย คือ

1. เวอร์ชั่นของ HTTP/x.x ที่ server ใช้สำหรับส่ง message
2. response status code ซึ่งเป็นตัวบอกว่าผลของ request ที่ Client ส่งมาเป็นอย่างไร
3. reason phase เป็นตัวอธิบายความหมายของ response status code อีกทีหนึ่ง เช่น

HTTP/1.0 200 OK

หรือ

HTTP/1.0 404 Not Found

กฎและเงื่อนไข

1) เวอร์ชั่นของ HTTP จะต้องอยู่ในรูปของ HTTP/x.x และจะต้องเป็นตัวใหญ่เสมอ

2) response status code เป็น code ที่ส่งมาให้ (Client) อ่านซึ่งจะมีข้อความโดยย่อ อธิบายแสดงผลของความผิดพลาด(ถ้ามี)

3) response status code จะอยู่ในลักษณะของเลขสามหลัก โดยหลักแรกจะบอกถึงความหมายโดยทั่ว ๆ ไปของ response

- 1xx เป็น code ที่ใช้บอกถึงเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในการสื่อสารระหว่าง Client และ Server
- 2xx เป็น code ที่บ่งบอกว่า request ที่ส่งจาก Client ถูกทำได้เสร็จสมบูรณ์แล้ว
- 3xx เป็น code ที่บอกให้ Client ทำการ redirect ไปที่ URL อื่นแทน
- 4xx เป็น code ที่บ่งบอกถึงความผิดพลาดที่เกิดขึ้นจากส่วนของ Client
- 5xx เป็น code ที่บ่งบอกถึงความผิดพลาดที่เกิดขึ้นจากส่วนของ Server

status code ที่พบเห็นโดยทั่วไป คือ

100 Continue

Server ได้รับ request บางส่วนจาก Client แล้ว, Client กรุณาส่งส่วนที่เหลือมาให้ด้วย

200 OK

request ที่มาจาก Client ถูกต้องและ resource ที่ client ต้องการอยู่ใน message body ของ response นี้แล้ว

301 Moved Permanently

resource ที่ Client ต้องการเคยอยู่ที่ Server นี้แต่ถูกย้ายไปอยู่ที่อื่นแล้ว

302 Moved Temporarily

resource ที่ Client ต้องการถูกย้ายไปอยู่ที่ Server อื่นหรือไม่สามารถ access ได้ชั่วคราว

400 Bad Request

server ไม่สามารถเข้าใจ request ที่ Client ส่งมา

403 Forbidden

server เข้าใจ request ที่ Client ส่งมาแต่ไม่ต้องการที่จะส่ง resource ที่ Client ต้องการกลับไปให้

404 Not Found

server ไม่พบ resource ที่ Client ต้องการ

500 Server Error

เกิดข้อผิดพลาดขึ้นในส่วนของ Server (โดยปกติ Server Side Application ที่รันบน Server เกิดข้อผิดพลาดขึ้นระหว่างการประมวลผล ซึ่งมักจะเกิดขึ้นเพราะโปรแกรมมีบัก)

2.2.4 Header

โดยทั่วไป header จะเป็นส่วนที่บอกถึงรายละเอียดของ request (ที่กำลังส่งไปให้ Server) หรือ response (ที่กำลังส่งกลับมายัง Client) ยกตัวอย่างเช่น

- Client จะส่ง Header ที่บอกถึงชนิดและขนาดของข้อมูลที่อยู่ข้างใน message body ในกรณีที่ Client ต้องการ upload ไฟล์ไปยัง server
- Server อาจจะส่ง Header ที่เกี่ยวกับชนิดและขนาดของ resource ที่ Client กำลังจะได้รับโดยใช้ Content-Type และ Content-Length

header จะเป็นลักษณะของ text format โดยในหนึ่งบรรทัดจะถูกใช้สำหรับหนึ่ง header ซึ่งจะอยู่ในรูปของ "Header-Name: value" และจบด้วย CRLF ยกตัวอย่างเช่น

From: soup@jarticles.com

User-Agent: Mozilla/4.72

Content-Type: text/html

Content-Length: 250

หมายเหตุ

1) ชื่อของ header สามารถใช้ตัวใหญ่หรือตัวเล็กก็ได้

2) เราสามารถเว้นช่องว่างหรือ tab ระหว่าง ":" ของ Header-Name และ value กว้างเท่าไรก็ได้

3) ใน HTTP1.0 จะมี header กำหนดไว้ 16 แบบแต่ของ HTTP1.1 จะมีถึง 46 แบบด้วยกัน

header ที่มักพบเห็นทั่วไปในส่วนของ Client

From:

เป็น header ที่ใช้สำหรับเก็บ email address ของผู้ที่กำลังส่ง request

เช่น From: soup@jarticles.com

User-Agent:

เป็น header ที่ใช้บ่งบอกถึงชื่อของโปรแกรมที่ใช้เป็น HTTP Client

เช่น User-Agent: Mozilla/4.72

header ที่มักพบเห็นทั่วไปในส่วนของ Server

Server:

เป็น header ที่จะคล้ายกับ User-Agent แต่เป็นตัวบอกถึงชื่อของโปรแกรมที่ใช้เป็นHTTP Server เช่น Server: Apache/1.93

Last-Modified:

เป็น header ที่ใช้บอกถึงเวลาครั้งสุดท้ายที่ resource ที่ client ต้องการถูกเปลี่ยนแปลง (modify/update) โดยเวลาที่ใช้จะเทียบจาก GMT (Greenwich Mean Time) ยกตัวอย่างเช่น เวลาของเมืองไทยถือว่าเป็น GMT+7.00 ถ้าตอนนี้เวลาในเมืองไทยคือ Sun, 23 July 2000 20:39:56 เวลาที่เป็น GMT ก็คือ Last-Modified: Sun, 23 July 2000 13:39:56 GMT

Message Body

เมื่อไรก็ตามที่ client หรือ server ต้องการส่งข้อมูลไปกับ message ส่วนที่เป็น message body จะเป็นส่วนที่ใช้สำหรับเก็บข้อมูลดังกล่าว ในกรณีของ client ตัว message body อาจใช้สำหรับบรรจุไฟล์ที่ต้องการ upload หรือใช้สำหรับเก็บข้อมูลที่มาจาก element ต่าง ๆ ของ HTML form แล้วส่งไปยัง server ก็ได้ ในกรณีของ server ตัว message body จะเป็นส่วนที่ใช้เก็บ resource ที่ client ทวงถาม หรืออาจใช้สำหรับเก็บคำอธิบาย ต่าง ๆ ในกรณีที่มี error เกิดขึ้นก็ได้ ถ้า message มีส่วนของ message body, client หรือ server มักจะเพิ่ม header ที่ช่วยบอกถึงรายละเอียดของ message body ดังกล่าวด้วย ยกตัวอย่างเช่นถ้า server ต้องการส่ง resource ที่เป็นไฟล์ image มาให้ client, header ที่ถูกเพิ่มขึ้นมาก็จะเป็น

Content-Type: image/gif

เป็น header ที่บ่งบอกถึง MIME type ของ data ที่อยู่ใน message body

Content-Length: 1026

เป็น header ที่บ่งบอกถึงขนาดของ message body ในหน่วย byte

ตัวอย่างของการส่ง message ระหว่าง client และ server

เกิดอะไรขึ้นบ้าง ถ้าสมมุติว่าเราต้องการ resource จาก URL ข้างล่างนี้

http://www.jarticles.com/tutorials/basic/helloworld.html

(Client = Web Browser, Server = Web Server)

ขั้นแรก Client จะทำการเรียกใช้ socketเพื่อคอนเนคไปที่เซิฟเวอร์ ซึ่งโดยปกติจะติดต่อไปที่ port 80 ซึ่งเป็น default port ของ HTTP protocol (ในกรณีที่ต้องการคอนเนคไปที่ port อื่นก็ทำได้โดยเพิ่ม port เข้าไปหลังจาก hostname เช่น

http://www.jarticles.com:8080/tutorials/advance/hellocorba.html Client จะคอนเนคไปที่ port 8080 แทน)

หลังจากนั้น Client จะทำการส่ง request message

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

From: soup@jarticles.com

User-Agent: Mozilla/4.72

[blank line]

หลังจากที่ Server ได้รับ message ข้างบนแล้ว Server จะส่ง response พร้อมทั้ง resource ที่ Client ต้องการกลับมาดังนี้

HTTP/1.0 200 OK

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

Content-type: text/html

Content-length: 1556

[blank line]

<HTML><HEAD><TITLE>HTTTP Tutorial</TITLE></HEAD>

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

(more file contents)

...

...

...

</BODY>

<HTML>

ท้ายสุด Server จะทำการปิดคอนเนคชั่นของ Client

2.2.5 GET Method

เมื่อใช้การทำงานลักษณะ GET Method อธิบายได้จากตัวอย่างของการใช้ search Engine ค้นหาข้อมูลบางอย่างอาจจะสังเกตเห็นว่าหลังจากที่ใส่ keyword ลงไปแล้วคลิกปุ่ม search, URL ที่อยู่ตรง Location Toolbar (Netscape), Address Bar (IE) ของ Web Browser จะเปลี่ยนเป็นอะไรคล้าย ๆ ดังนี้

http://www.google.com/search.cgi?keyword=jarticles

(search ที่ www.google.com โดยใช้ keyword = "jarticles")

ถ้าลองดูส่วนที่อยู่หลังจากส่วนของ hostname (www.google.com) แล้วเราจะเห็นว่า requested URI (/search.cgi?keyword=jarticles) ที่เห็นไม่ได้เป็น html ไฟล์อย่างทั่ว ๆ ไป แต่กลับกลายเป็นชื่อของโปรแกรม(search.cgi), เครื่องหมายคำถาม(?)และ keyword ที่เราใช้ค้นหาแทน

อย่างที่กล่าวมาแล้วในตอนต้นของบทความว่า resource จริง ๆแล้ว ก็คือตัวที่ใช้บ่งชี้ถึงข้อมูลที่อยู่ที่ server.resource อาจเป็นไฟล์หรืออาจเป็น query string ที่ใช้ส่งไปเรียกโปรแกรมที่อยู่ที่ server ก็ได้ ซึ่งในกรณีของตัวอย่าง search Engine ข้างต้นตัว resource ก็คือโปรแกรมที่ชื่อ search.cgi นั่นเอง

โดยทั่วไป response ที่ได้กลับมาจาก server อาจจะอยู่ในรูปของไฟล์ถ้า resource ที่เราทวงถามเป็นไฟล์ ยกตัวอย่างเช่น http://www.jarticles.com/tutorial/http.html หรืออาจจะเป็น output ที่ได้จากการประมวลผลของประแกรมที่เราผ่าน query string เข้าไป เช่นถ้าส่ง query string ไปที่ search engine ผลที่จะได้กลับมาก็คือ link ต่าง ๆ ที่เกี่ยวข้องกับ keyword ที่เราใช้ค้นหานั่นเอง

โดยทั่วไป Web Browser (Client) จะใช้ GET Method ในการส่ง request message ไปที่ server ในกรณีที่ resource ที่ต้องการเป็นไฟล์ อย่างไรก็ตาม GET ยังสามารถใช้ในการส่ง query string สั้น ๆ ไปยังโปรแกรมที่รันอยู่ที่ server ได้อีกด้วย ยกตัวอย่างเช่น

ถ้าเราต้องการส่งข้อมูลชื่อ keyword โดยมีค่าเท่ากับ jarticles ไปที่ server ชื่อ www.google.com โดยเจาะจงไปที่ resource ซึ่งเป็น cgi โปรแกรมชื่อ search.cgi วิธีการทำก็คือ

1. กำหนด resource ลงไปในส่วนของ request URI ซึ่งจะอยู่หลังจากส่วนของ hostname ซึ่งจะได้ http://www.google.com/search.cgi

2. ใส่ตัวเครื่องหมายคำถามเพื่อบอก server ว่าส่วนที่เหลือถัดไปจะเป็นส่วนของข้อมูลซึ่งจะได้ http://www.google.com/search.cgi?

3. ทำการ encode โดยวิธีการที่เรียกว่า URL-encoding ไปที่ชื่อและค่าของตัวแปรต่าง ๆ ซึ่ง ในกรณีของเราตัวแปรก็คือ keyword ซึ่งจะได้ http://www.google.com/search.cgi?keyword=jarticles

ถ้าเราพิมพ์ URL ข้างบนเข้าไปใน Web Browser แล้วกดปุ่ม Enter ตัว Web Browser จะทำตีความ URL ดังกล่าว ซึ่งผลที่ได้คือการใช้ GET ส่ง request message ไปที่ server ที่ชื่อ www.google.com

โดยจะแนบ "jarticles" keyword ไปกับ request ซึ่ง initial line ของ request message จะเป็นอะไรคล้าย ๆ ข้างล่างนี้

GET /search.cgi?keyword=jarticles HTTP/1.0
From: soup@jarticles.com
User-Agent: Mozilla/4.72
[blank line]

หลังจากนั้นเราจะได้ลิงค์ต่าง ๆ ที่เกี่ยวกับ jarticles กลับมา

หมายเหตุ : URL จริง ๆ ที่ใช้กับ www.google.com จะต่างจาก URL ข้างบนเล็กน้อย

วิธีการส่ง query string ไปกับ URL เหมาะสำหรับ query string ที่ไม่ยาวมากนัก โดยทั่วไปความยาวของ URL มักจะจำกัดอยู่ที่ 80 characters ซึ่งโดยทั่ว ๆ ไปแล้ว character ที่ถัดจากนั้นจะถูกตัดออกไปโดยอัตโนมัติ (จะตัดหรือไม่ตัวจริง ๆ ขึ้นอยู่กับการ implement ของ server แต่ละตัว)

2.2.6 URL-Encoding

ในการส่งข้อมูลไปที่ server โดยวิธีการ GET หรือวิธีการ POST นั้นชื่อ และค่าของตัวแปรต่าง ๆ ซึ่งอยู่หลังจากเครื่องหมายคำถามในกรณีของ GET หรืออยู่ในส่วนของ message body ในกรณีของ POST จะต้องถูกผ่านการ encode โดยวิธีการที่เรียกว่า URL-Encoding เสียก่อน

โดยทั่วไป URL เองจะมี character บางตัวที่ใช้สำหรับความหมายพิเศษ เช่น : , / , ~ , & , ? ซึ่งในบางครั้งชื่อและค่าของตัวแปรต่าง ๆ ที่ถูกส่งไปที่ server อาจจะมี character เหล่านี้ปะปนอยู่ด้วย เพื่อหลีกเลี่ยงความสับสนในการตีความของ server, character ต่าง ๆ ที่เป็น character ที่ใช้ใน URL จะต้องถูก encode เสียก่อน โดยมีหลักการดังต่อไปนี้

1. ให้ทำการเปลี่ยน unsafe characters ต่าง ๆ ที่อยู่ในชื่อ และค่าของตัวแปรให้กลายเป็น "%xx" โดย "xx" คือรหัส ASCII ของ character นั้น ๆ ในแบบ hexadecimal ซึ่ง unsafe characters เหล่านี้รวมไปถึง = , & , % , + และ characters พิเศษต่าง ๆ ที่ไม่สามารถพิมพ์ได้ หรือแม้กระทั่ง characters ที่เราต้องการจะ encode เอง
2. ให้ทำการเปลี่ยนช่องว่าง (space) ทั้งหมดให้กลายเป็นเครื่องหมายบวก (+)
3. จับคู่ชื่อและค่าของตัวแปรต่าง ๆ ด้วย =

4. ถ้ามีชื่อและค่าของตัวแปรมากกว่าหนึ่งตัว ให้นำชื่อและค่าของตัวแปรแต่ละคู่มาเชื่อมติดกันด้วยเครื่องหมาย &

5. นำชื่อและค่าของตัวแปรที่เชื่อมติดกันทั้งหมดไปใส่ที่ URL โดยมีเครื่องหมาย ? อยู่ข้างหน้า (ในกรณีของ GET) หรือนำไปใส่ที่ message body (ในกรณีของ POST) ยกตัวอย่างเช่น ถ้าเรามีชื่อและค่า

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

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

หลังจากทำการ encode จะได้ String keyword=jarticles&name=Soup+%26+friends โดย String นี้มีความยาวเท่ากับ 39

หมายเหตุ : %26 แทนเลขฐาน 16 ของสัญลักษณ์ &

2.2.7 POST Method

บางครั้งเราอาจจะไม่ต้องการส่งข้อมูลไปยัง server ด้วยการพ่วงติดข้อมูลเหล่านั้นร่วมไปกับ URL เนื่องจากข้อมูลเหล่านั้นอาจเกี่ยวข้องกับข้อมูลความลับส่วนตัวของเรา ยกตัวอย่างเช่นข้อมูล login และ password ในกรณีที่เราใช้ GET หลังจากที่เราทำการ login ไปแล้ว URL ที่อยู่ตรงAddress Barอาจออกมาเป็น http://www.myserver.com/login.cgi?login=me&password=youandme

ถ้าบังเอิญว่าระหว่างที่เราล็อกอิน มีคนอื่นนั่งอยู่ข้าง ๆ ด้วยพอดี คนคนนั้นก็อาจจะนำเอาข้อมูลของเราไปใช้อย่างผิด ๆ ได้ อีกกรณีหนึ่งคือ กรณีของข้อมูลที่เราติดไปกับส่วนของ request URI ทำให้ URL มีความยาวมากกว่าความยาวของ URL ที่ server กำหนดไว้ ผลกระทบที่อาจเกิดขึ้นคือข้อมูลบางส่วนของเราอาจหายไปเนื่องจากการตัดทอนโดย server

วิธีการที่เราสามารถจะปิดบังข้อมูล(แบบง่าย ๆ) หรือเพื่อส่งข้อมูลที่มีความยาวมาก ๆ ก็คือการใช้ POST method

POST ต่างกับ GET ตรงที่ว่า POST จะไม่ทำการพ่วงติดข้อมูลไปกับ URL แต่ POST จะทำการใส่ข้อมูลเข้าไปในส่วนของ message body ของ request message แทน ซึ่งในกรณีนี้ request URI จะมีปรากฏเพียงชื่อของโปรแกรมที่เราต้องการส่งข้อมูลไปให้เท่านั้น

ในการส่งข้อมูลไปในส่วนของ message body, เพื่อให้ง่ายต่อการตีความโดย server ทาง POST method จึงมีการบังคับให้ใส่ header พิเศษเพื่อบรรยายรายละเอียดต่าง ๆ ของข้อมูลที่อยู่ใน message body นั้นอีกด้วยซึ่งโดยทั่วไป header ที่มักจะถูกเพิ่มเข้าไปก็คือ Content-Type และ Content-Length header

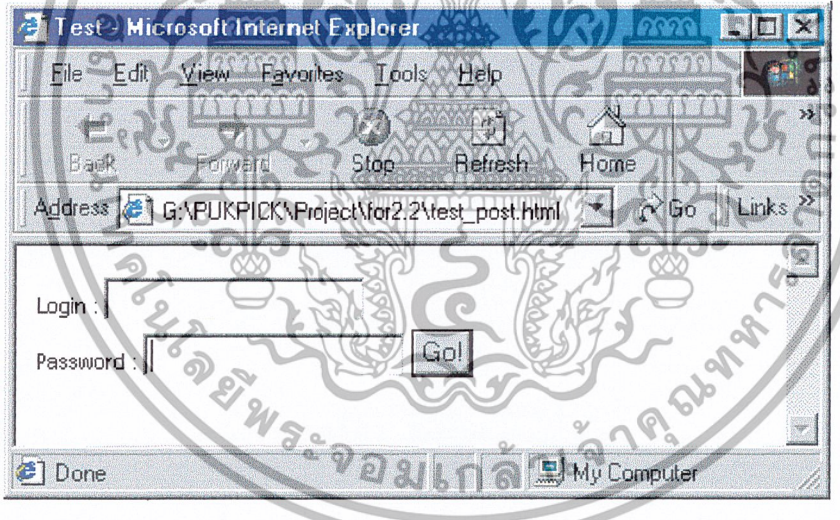
การใช้ POST มักจะเป็นการส่งข้อมูลที่มาจาก HTML Form ดังตัวอย่างในรูป 2.11 และ 2.12

test_post - Notepad

File Edit Search Help

```
<html>
<head>
<title>Test</title>
</head>
<body>
<FORM METHOD="POST" NAME="loginform"
ACTION="http://www.jarticles.com/cgi-bin/login.cgi">
Login : <INPUT TYPE="text" NAME="userName" SIZE="20" MAXLENGTH="20" VALUE="">
<br>
Password : <INPUT TYPE="PASSWORD" NAME="password" SIZE="20" MAXLENGTH="20"
VALUE="">
<INPUT TYPE="SUBMIT" VALUE="Go!">
</FORM>
</body>
</html>
```

รูปที่ 2.11 ตัวอย่างการใช้ POST ในการส่งข้อมูลจาก HTML Form



รูปที่ 2.12 ผลที่แสดงใน Internet Explorer ของ HTML ในรูป 2.11

เมื่อคลิกที่ปุ่ม GO! ในรูป 2.12 Web Browser จะทำการส่ง message ต่อไปนี้ไปที่ server

POST /cgi-bin/login.cgi HTTP/1.0

From: soup@jarticles.com

User-Agent: Mozilla/4.72

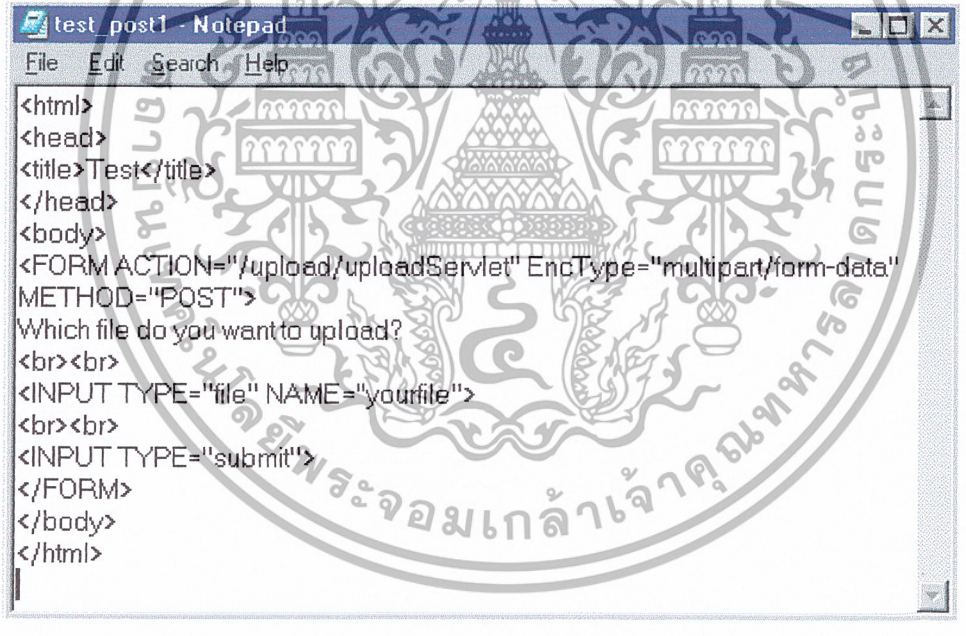
Content-Type: application/x-www-form-urlencoded

Content-Length: 32

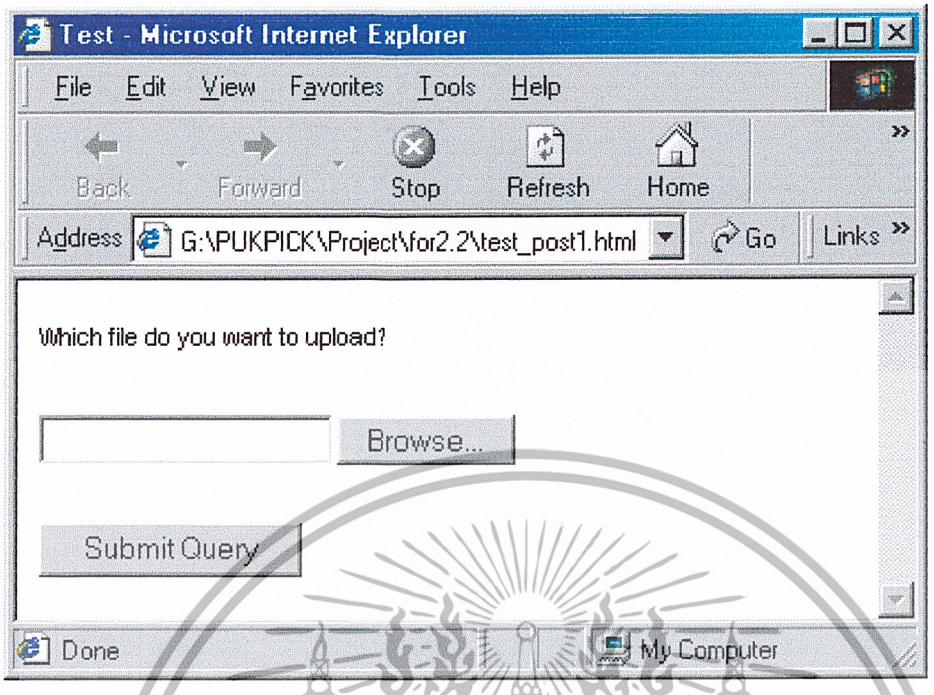
userName=abc&password=123

หมายเหตุ : ในกรณีของการส่งข้อมูลจาก HTML Form, Client จะทำการเซ็ต Content-Type เป็น application/x-www-form-urlencoded และ Content-Length จะมีค่าเท่ากับความยาวของชื่อและค่าของ form elements ต่าง ๆ ที่อยู่ใน HTML Form รวมกันซึ่งจากตัวอย่างในรูป 2.11 ก็คือ form element ที่ชื่อ userName และ password

นอกจากนี้ POST ยังสามารถใช้กับการส่งข้อมูลแบบอื่น ๆ ได้อีก โดย format จะขึ้นอยู่กับวิธีการส่งที่ทาง Web Browser และ Web Server ได้ทำการตกลงกันไว้ ยกตัวอย่างเช่น ในกรณีของไฟล์ Upload ตัว Content-Type ก็จะถูกเซ็ตเป็น Multipart/form-date แทน ซึ่งตัวอย่างของ HTML Form ประเภทนี้แสดงในรูปที่ 2.13 และรูปที่ 2.14



รูปที่ 2.13 ตัวอย่างการใช้ POST ในการ Upload ไฟล์



รูปที่ 2.14 ผลของการแสดงด้วย Internet Explorer ของ HTML ในรูปที่ 2.13

2.3 ภาษา HTML เบื้องต้น

HTML มาจากคำว่า HyperText Markup Language ซึ่งเป็นรูปแบบของภาษาที่ใช้ในการเขียนโปรแกรมเว็บเพจเพื่อแสดงผลบนเว็บบราวเซอร์ เอกสาร HTML จะเป็นเท็กซ์ไฟล์ธรรมดาที่ต้องอาศัยการแปลความจากเว็บบราวเซอร์ ในสมัยก่อนจุดประสงค์ในการใช้ HTML เพื่อแสดงผลที่เป็นข้อความส่วนใหญ่ แต่ในปัจจุบัน HTML มีความสามารถเพิ่มเติมมากมายที่รวมทั้งความสามารถในด้านมัลติมีเดีย

คำสั่งของ HTML เรียกว่า " แท็ก " ซึ่งแท็กนี้จะอยู่ในรูปแบบ <...>.....</...> ซึ่งเว็บบราวเซอร์จะแปลงแท็กนี้แล้วแสดงผลให้เห็น โดยทั่วไปการสร้างเว็บเพจด้วยภาษา HTML โดยใช้อิดิเตอร์ต่างๆ เช่น Notepad ของ Windows แต่โปรแกรมเมอร์จะต้องเข้าใจรูปแบบคำสั่งหรือแท็กของ HTML ทั้งหมด ซึ่งเป็นการยากและเสียเวลามาก ในปัจจุบันจึงได้มีเครื่องมือที่ช่วยสนับสนุนการเขียนโปรแกรมบนเว็บเพจในลักษณะ WYSIWYG (What You See Is What You Get) เช่น Microsoft Word หรือ Microsoft FrontPage เป็นต้น โปรแกรมสำเร็จรูปในลักษณะนี้จะทำให้ประหยัดเวลาในการสร้างเว็บเพจ เพราะสามารถใช้งานเหมือนกับการสร้างเอกสารทั่วไป คือ คีย์ข้อความ แล้วเปลี่ยนรูปแบบอื่นๆ ตามต้องการ หลังจากนั้นเมื่อมีการจัดเก็บเอกสารแค่เพียงเลือกรูปแบบเอกสารที่ต้องการจัดเก็บให้เป็นแบบ HTML หลังจากนั้นก็สามารถนำไปแสดงผลบนเว็บบราวเซอร์ได้ทันที โดยโปรแกรมสำเร็จรูปนี้จะสร้างโค้ด HTML ให้โดยอัตโนมัติ

2.3.1 โครงสร้างของเอกสาร HTML

เอกสาร HTML มีองค์ประกอบ 2 ส่วน คือ ส่วนที่เป็นเนื้อหา และ ส่วนที่เป็นคำสั่ง หรือ แท็ก รูปแบบพื้นฐานโครงสร้างของเอกสาร HTML จะเป็นดังตารางที่ 2.1

ตารางที่2.1 รูปแบบพื้นฐานของเอกสาร HTML

รูปแบบ	ความหมาย
<HTML>.....</HTML>	เป็นคำสั่งเริ่มต้นและสิ้นสุดของเอกสาร HTML เหมือนคำสั่ง Begin และ End ใน Pascal
<HEAD>.....</HEAD>	ใช้กำหนดข้อความ ในส่วนที่เป็น ชื่อเรื่อง ภายในคำสั่งนี้ จะมีคำสั่งย่อยอีกหนึ่งคำสั่ง คือ <TITLE>
<TITLE>.....</TITLE>	เป็นส่วนแสดงชื่อของเอกสาร โดยจะแสดงที่ไตเติลบาร์ของวินโดว์ที่เปิดเอกสารนี้อยู่เท่านั้น

รูปแบบ	ความหมาย
<BODY>…..</BODY>	ส่วนเนื้อหาของโปรแกรมจะเริ่มต้นด้วยคำสั่ง <BODY> และสิ้นสุดด้วย </BODY> ในระหว่าง 2 คำสั่งนี้ จะประกอบด้วยแท็กมากมายตามที่ต้องการให้แสดงผลบนบราวเซอร์

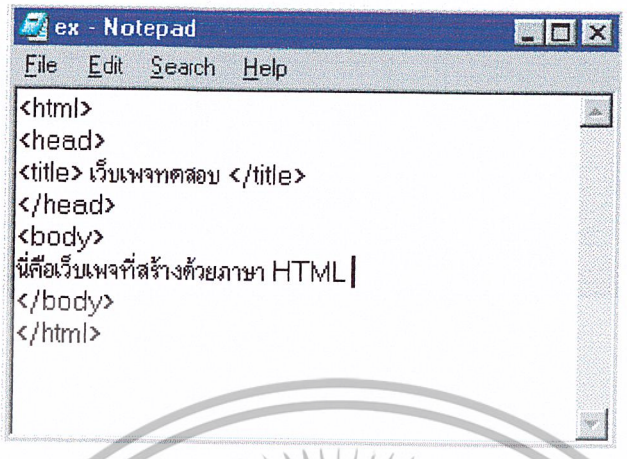
สำหรับรายละเอียดของ Tag ที่สมบูรณ์ สามารถศึกษาได้ที่ www.w3c.org

ลักษณะของโค้ดภาษา HTML จะเป็นดังนี้

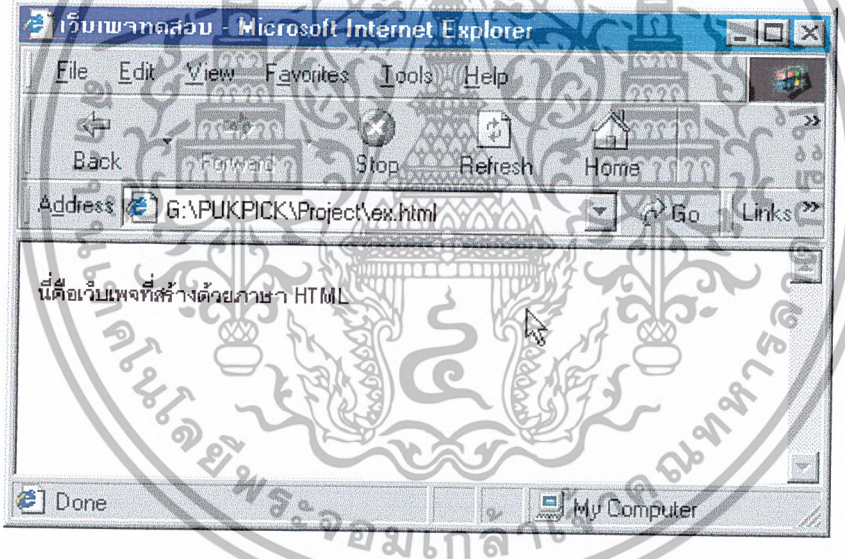
```
<html>
<head>
<title> ชื่อแสดงบนไตเติ้ลบาร์ของเว็บบราวเซอร์ </title>
</head>
<body>
    ……..
    คำสั่งหรือข้อความที่ต้องการแสดงผลบนเว็บบราวเซอร์
    ……..
</body>
</html>
```

2.3.2 เริ่มเขียนโปรแกรม HTML

ในการเขียนโปรแกรมภาษา HTML สามารถใช้เอดิเตอร์ได้หลายตัว ที่สะดวกที่สุดคือการใช้ NotePadที่มากับวินโดว์ (หรือจะทดลองใช้ EditPlus ก็ได้) เมื่อสร้างเรียบร้อยแล้วจัดเก็บเอกสารนั้นให้มีนามสกุล .htm หรือ .html ก็ได้



รูปที่ 2.15 ตัวอย่างโปรแกรม HTML



รูปที่ 2.16 ผลของการแสดง HTML ในรูป 2.15 ใน Internet Explorer

2.4 Servlets

Servlet มีบทบาทสำคัญในการพัฒนา web application โดยใช้ Java เนื่องจาก Servlet มีความสามารถในการเขียน server side code เพื่อติดต่อกับ client ดังนั้น Servlet จึงเป็นหัวใจสำคัญในการทำ server side programming

Servlet API มีรูปแบบทั่วไปของ class ที่สามารถทำงานในการให้บริการได้ ซึ่งการให้บริการก็นับเป็นหน้าที่สำคัญของ server อย่างหนึ่ง แต่ถ้าดูความหมายตามชื่อแล้ว Servlet ก็คือ server ที่มีขนาดเล็กและมีลักษณะเฉพาะ

Servlet container ช่วยให้ servlet programmer ไม่ต้องกังวลเกี่ยวกับรายละเอียดในการติดต่อกับ network, การตรวจจับ request จาก client, การสร้าง response ที่มีรูปแบบที่ถูกต้อง เนื่องจากสิ่งเหล่านี้เป็นหน้าที่ของ servlet container เอง หรือที่รู้จักกันว่า servlet engine โดย servlet container ทำการแปลง request ที่ส่งมาตาม protocol ต่าง ๆ ให้เป็น object ที่ servlet เข้าใจได้ และจะส่ง object ไปยัง servlet ซึ่ง servlet จะนำไปใช้ในการส่ง response ต่อไป นอกจากนี้ servlet container ยังมีหน้าที่รับผิดชอบในการจัดการวัฏจักรการทำงานของ servlet

ทั้ง servlet API library และ servlet engine เป็นส่วนหนึ่งใน Java Servlet Development Kit (JSDK) โดยสามารถดาวน์โหลดได้จาก http://java.sun.com/products/servlet

2.4.1 The Servlet Container

servlet container มีหน้าที่ในการจัดการกับ request จาก client ด้วยการส่ง request ไปยัง servlet และตอบรับการ request กลับไปยัง client นั้น การ implement ของ servlet container จะเปลี่ยนไปตามโปรแกรม แต่การทำงานระหว่าง servlet container และ servlet ต่าง ๆ จะถูกระบุโดย servlet API ในการทำงานระหว่าง servlet container จะมีการกำหนด method ต่าง ๆ ขึ้น แล้ว servlet container จะเรียกให้ servlet ทำงานตาม method นั้น ๆ ต่อไป นอกจากนี้ยังกำหนด class ต่าง ๆ ของ object ที่ servlet container จะส่งไปยัง servlet ต่อไป

วัฏจักรการทำงานของ servlet หนึ่ง ๆ มีลักษณะดังต่อไปนี้

- servlet container สร้าง instance ของ servlet
- servlet container เรียก method ชื่อ init() ของ instance นั้น
- ถ้า servlet container มี request ที่ต้องเรียกใช้ servlet ในการให้บริการ servlet container ก็จะเรียก method ชื่อ service() ของ instance นั้น
- ก่อนจะลบ instance ทิ้งไป servlet container จะเรียก method ชื่อ destroy() ของ instance นั้น

- instance นั้นจะถูกลบและถูกกำหนดให้กระบวนการ garbage collection มาจัดการต่อไป

การทำงานระหว่าง servlet container การันตีว่าก่อนที่ method ชื่อ service() จะถูกเรียกนั้น method ชื่อ init() จะต้องทำงานเสร็จสมบูรณ์เสียก่อน และก่อนที่ instance ของ servlet จะถูกลบไปนั้น method ชื่อ destroy() จะถูกเรียกให้ทำงานเสียก่อน ในทางปฏิบัติ มองว่า servlet container สร้าง instance ของ servlet เมื่อ servlet เริ่มทำงานหรือถูกเรียกให้ทำงานในครั้งแรก และถือว่า servlet container จะเก็บ instance ไว้ใน memory เพื่อให้บริการสำหรับ request อื่น ๆ ต่อไป servlet container อาจตัดสินใจที่จะลบ instance ออกไปจาก memory เมื่อใดก็ได้ เช่น เมื่อ servlet นั้นไม่ถูกเรียกใช้งานเลยในช่วงเวลาที่กำหนดไว้ หรือเมื่อ servlet container กำลัง shut down

ดังนั้น servlet container จะสร้างเพียงหนึ่ง instance สำหรับแต่ละ servlet (แม้จะไม่มีเหตุผลว่าทำไมต้องสร้างเพียงหนึ่ง instance ก็ตาม) ถ้า method ชื่อ service() กำลังทำงานอยู่และมีอีก request เข้ามา servlet container สามารถรอจนกระทั่ง method ชื่อ service() นั้นทำงานจนเสร็จสมบูรณ์ก่อนจะทำการเรียก method ชื่อ service() มาทำงานอีกครั้งหนึ่ง หรือสร้างอีก thread เพื่อให้บริการโดยการเรียก method ชื่อ service() เช่นกัน เนื่องจากไม่มีข้อกำหนดว่ามีเพียง thread เดียวที่สามารถเรียกใช้ method ชื่อ service() ในเวลาหนึ่ง ๆ

ในทางปฏิบัติ servlet container จะไม่สร้าง thread ใหม่ทุก ๆ ครั้งที่ได้รับ request แต่ servlet container จะใช้ pool ของ thread ในการจัดการกับ request ที่เข้ามาพร้อม ๆ กันแทน แต่ผลกระทบที่มีต่อ servlet ของทั้งสองวิธีก็มีมากเท่า ๆ กัน ดังนั้นภาพการทำงานของ servlet container จะเป็นดังภาพด้านล่าง โดย diagram นี้แสดงให้เห็นว่า servlet ถูกเรียก, ให้บริการแก่ 2 request เสร็จอย่างรวดเร็ว และสุดท้าย servlet จะถูกลบไปเมื่อ server ถูก shut down

เนื่องจาก web server ส่วนใหญ่ถูกสร้างขึ้นด้วยภาษาอื่น ๆ เช่น ภาษา C หรือ C++ ดังนั้นขณะที่ web server ที่สร้างด้วย Java (อาทิ Sun's Java Web Server และ ServletRunner ที่มีอยู่ใน JSDK) มี servlet container เป็นของตัวเอง web server อื่น ๆ (อาทิ Apache และ Microsoft IIS) จึงต้องการโปรแกรมบางอย่างเพิ่มเติมเพื่อควบคุมการทำงานของ servlet นอกจากนั้นก็ยังต้องการ plug-in หรือ module บางอย่างในการจัดการการติดต่อกันระหว่าง web server กับ servlet container ยกตัวอย่างเช่น Apache Jserv ที่มี internet protocol พิเศษที่ชื่อว่า AJPv1.1 ในการจัดการเกี่ยวกับการติดต่อกันนี้ โดย web server จะ request ไปยัง servlet และ servlet container จะส่งผลลัพธ์กลับไปยัง server ซึ่งจะเห็นว่าในกรณีนี้ servlet container และ web server ไม่จำเป็นต้องทำงานอยู่บนเครื่องเดียวกัน

เนื่องจาก servlet ทำงานใกล้ชิดกับ web server มาก ดังนั้น servlet จึงสามารถช่วยเพิ่มความสามารถของ web server ได้เป็นอย่างดี servlet มีความสามารถกระทั่งการทำงานพื้นฐานส่วนใหญ่ กล่าวคือ servlet สามารถจัดการกับ process มาตรฐานของ web server ได้ เช่น การแปลง URL ที่ request มาให้เป็น filepath, การส่งไฟล์ HTML คืนกลับไป Sun's Java Web Server ใช้ servlet ในการจัดการกับทุกอย่าง รวมไปถึงการประมวลผล CGI, การทำงานของ Java Server Pages (JSP), template ของ HTML และ Server Side Includes (SSI) นั่นแสดงว่า servlet สามารถทำงานได้ทุกอย่างบน web server

2.4.2 The Servlet API

Class และ Interface ต่าง ๆ อยู่ใน 2 package อันได้แก่ javax.servlet และ javax.servlet.http โดยที่ package แรกมี interface พื้นฐาน และ package หลังมี class ต่าง ๆ ที่ได้รับมาจาก generic servlet interface ซึ่งมี tool พิเศษที่ใช้ในการให้บริการ request บน HTTP

2.4.2.1 Servlet Interface

public interface Servlet

วัฏจักรการทำงานของ servlet ถูกกำหนดไว้ใน interface ชื่อ javax.servlet.Servlet เมื่อสร้าง servlet เราจะต้อง implement interface ไม่ว่าจะโดยทางตรงหรือทางอ้อม ส่วนมากจะ implement ทางอ้อมด้วยการ extend class javax.servlet.GenericServlet หรือclassjavax.servlet.http.HttpServlet ขณะที่ทำการ implement interface servlet 5 method ต่อไปจะต้องถูก implement ไปด้วย

init() method

public void init(Servletconfig config) throws ServletException

เมื่อ servlet ถูกสร้างขึ้น servlet container จะเรียก method ชื่อ init() โดย servlet container จะส่ง object ชนิด ServletConfig ไปยัง method ชื่อ init() เพื่อใช้เก็บข้อมูลที่เกี่ยวกับการ configuration ของ engine ซึ่งจะถูกนำไปใช้ต่อไปในภายหลัง method ชื่อ init() จะส่ง event ที่เกิดจาก ServletException เนื่องจากมีการประกาศ throws ServletException เมื่อเกิดเหตุการณ์เช่นนี้ servlet นั้นจะไม่สามารถให้บริการได้ แล้วจะเรียกไปยัง servlet ที่ทำให้ servlet นั้นถูกเรียกขึ้นมาใช้งานอีกครั้งโดย servlet container และ method ชื่อ init() ก็จะถูกเรียกทำงานอีกครั้งหนึ่ง interface ชื่อ servlet การันตีว่า method ชื่อ init() จะถูกเรียกเพียงครั้งเดียวจริง และการันตีว่า method ชื่อ init() จะสามารถทำงานได้เสร็จสมบูรณ์โดยไม่มีการส่ง event ที่เกิดจาก ServletException ก่อนที่จะมี request ใด ๆ ถูกส่งไปยัง servlet

service() method

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

method ชื่อ service() จะถูกเรียกก็ต่อเมื่อ servlet พร้อมสำหรับการให้บริการ request ที่เข้ามาแล้ว method ชื่อ service() รับ parameter 2 ตัว ได้แก่ object ชื่อ ServletRequest และ ServletResponse โดยServletRequest มี method และ field ที่จะเข้าถึงข้อมูลเกี่ยวกับ request ส่วน ServletResponse มี method ที่ servlet ใช้ในการสร้าง response

destroy() method

public void destroy()

ณ เวลาใด ๆ ที่กำหนดไว้ servlet container อาจตัดสินใจที่จะลบ servlet ออกไป ซึ่งอาจเกิดขึ้นเมื่อต้องการ memory เพิ่มหรือเมื่อ web server ถูก shut down ก่อนที่ servlet container จะเรียก method นี้ servlet container จะรอให้ thread ที่ทำหน้าที่ในการให้บริการได้ทำงานให้เสร็จเสียก่อน ดังนั้น interface ชื่อ servlet การันตีว่า method ชื่อ destroy() จะไม่ทำงานขณะที่ method ชื่อ service() กำลังทำงานอยู่

getServletConfig() method

public ServletConfig getServletConfig()

ระหว่างที่ servlet กำลังเริ่มต้นทำงาน object ชื่อ ServletConfig ที่ถูกส่งจาก servlet engine จะเก็บ instance ของ servlet ไว้ โดย ServletConfig สามารถเข้าถึง Init parameters และ object ชื่อ ServletContext ได้ Init parameters มักจะถูกระบุไว้ในไฟล์ของ servlet container โดยมีจุดประสงค์เพื่อให้ servlet ได้รับข้อมูลที่จำเป็นต้องใช้ขณะ runtime ขณะที่ ServletContext ช่วยให้ servlet มีความสามารถในการค้นหาข้อมูลที่เกี่ยวกับ servlet container การเรียกใช้ object หรือข้อมูลทั้งสองโดย method ชื่อ getServletConfig นี้จะทำเมื่อใดก็ได้

getServletInfo() method

public String getServletInfo()

method นี้คืนค่าเป็น object ชนิด String ที่เก็บข้อมูลเกี่ยวกับ servlet (เช่น ชื่อผู้สร้าง servlet, วันที่สร้าง servlet, คำบรรยาย เป็นต้น) servlet container ก็สามารถใช้ method นี้ได้เช่นกัน โดยอาจเก็บข้อมูลเป็น list ของ servlet ต่าง ๆ ที่อยู่ด้วยกันพร้อมคำอธิบาย

2.4.2.2 GenericServlet Class

Public abstract class GenericServlet implements Servlet, ServletConfig, Serializable

Class ชื่อ GenericServlet สามารถ implement interface ชื่อ Servlet ได้ จะสังเกตเห็นได้ว่าคลาสชื่อ GenericServlet ถูกประกาศไว้เป็น abstract เนื่องจากว่า method ชื่อ service() ถูกประกาศเป็น abstract ซึ่งหมายความว่าเมื่อเรา extend class นี้ เราต้อง implement method ชื่อ service() ด้วย ส่วน method อื่น ๆ จะถูก implement ดังนี้

method ชื่อ init(ServletConfig conf) เก็บ object ชนิด ServletConfig ไว้ใน instance variable แบบ private transient (ที่เรียกกันว่า config) ซึ่ง method ชื่อ getServletConfig() สามารถเข้าถึงและคืนค่า reference ที่อ้างถึง object นั้นได้ นั่นหมายความว่าถ้าเรา override method ชื่อ getServletConfig() โดยไม่ระวัง เราจะไม่สามารถใช้ method นี้ในการเรียกใช้ object ชนิด ServletConfig ได้ ด้วยเหตุผลนี้เองเราจะต้องทำการเรียก super.config(conf) ด้วย

API version 2.1 ของ Java Servlet มี method ชื่อ init() ที่ไม่มี argument อยู่ใน class ชื่อ GenericServlet เพื่อให้ override ได้ การ override จะทำให้เราสามารถเขียน code ใน servlet นั้นได้โดยไม่ต้องเรียกใช้ super.init(conf)

ความสามารถที่เพิ่มขึ้นมาใน API version 2.1 คือ class ชื่อ GenericServlet สามารถ implement interface ชื่อ ServletConfig ได้ ทำให้ผู้พัฒนา Servlet สามารถเรียกใช้ method ชื่อ ServletConfig() ได้โดยตรงโดยไม่ต้องเรียกผ่าน Object ชนิด ServletConfig โดย method เหล่านี้ได้แก่ getInitParameter(), getInitParameterNames(), getServletContext() ซึ่งแต่ละ method จะไปเรียก method ที่เกี่ยวข้องที่อยู่ใน นobject ชนิด ServletConfig

คสาส GenericServlet ยังมีอีก 2 method ที่ใช้ในการสร้าง servlet log ซึ่งในความเป็นจริงจะไปเรียก method ที่เกี่ยวข้องของ ServletContext มาช่วยทำงาน โดย method แรกก็คือ log(String msg) จะเขียนชื่อของ Servlet และ argument ชื่อ msg ลงไปใน log ของ Servlet Container ส่วนอีก method หนึ่งก็คือ log(String msg, Throwable cause) ซึ่งมี exception เพิ่มเข้าไปใน log นอกเหนือจากชื่อของ Servlet และ argument ชื่อ msg

class ต่อไปนี้อยู่ใน package ชื่อ javax.servlet.http

2.4.2.3 HttpServlet Class

class ชื่อ HttpServlet extend class ชื่อ GenericServlet และมี method ที่เกี่ยวกับ HTTP ที่อยู่ใน interface ชื่อ Servlet ให้เรียกใช้ class นี้เป็น class ที่เรามักจะเรียกใช้โดย extend ด้วย servlet

การ extend class ชื่อ GenericServlet ทำให้เรามองเห็นขอบเขตการทำงานที่เกี่ยวกับการให้บริการบนเครือข่าย แต่ HTTP เป็น protocol ที่นิยมใช้กับ servlet มากที่สุด

service() method

protected void service (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

method ชื่อ service() จะถูก implement โดย HttpServlet ที่ทำหน้าที่ในการจัดการกับ HTTP request ดังนั้นเราจึงไม่ควร override method นี้ เมื่อมี request เข้ามา method ชื่อ service() จะตรวจสอบชนิดของ request นั้น (ได้แก่ GET, POST, HEAD, OPTIONS, DELETE, PUT และ TRACE) และจัดการกับ request นั้นด้วย method ที่ทำงานสำหรับ request ชนิดนั้น ๆ (ได้แก่ doGet(), doPost(), doHead(), doOptions(), doDelete(), doPut() และ doTrace()) การจะเขียน servlet ที่โต้ตอบกับ HTTP request ตามชนิดที่เจาะจง เราทำได้ด้วยการ override method doXxx() ของ request ชนิดนั้น ๆ แต่ถ้ามี HTTP request ชนิดใดที่เราไม่ได้ override ไว้ servlet จะคืนค่า error ที่เป็นมาตรฐานที่บอกว่า that method is not valid for this resource.

getLastModified() method

protected long getLastModified(httpServletRequest req)

method นี้จะคืนค่ามาเป็นเวลาที่ servlet ถูกแก้ไขล่าสุดในรูปแบบ 00:00:00 ตามมาตรฐาน GMT แต่ถ้าไม่รู้เวลาที่เกิดจากการแก้ไข ค่า default ที่คืนกลับมาจะเป็นตัวเลขที่

เป็นค่าลบ เมื่อมีการให้บริการ request ชนิด GET method นี้จะสามารถบอก server ได้ว่า servlet ที่ให้บริการนั้นถูกแก้ไขล่าสุดเมื่อใด

2.4.2.4 HttpServletRequest Interface

public interface HttpServletRequest extends ServletRequest

เพื่อจะเข้าใจ interface นี้ เราจะต้องรู้ด้วยว่า HTTP ยอมให้ข้อมูลส่งผ่านไปยัง web server ได้อย่างไร HTTP ยอมให้เราส่ง parameter ต่าง ๆ ไปพร้อม ๆ กับ request ได้ ใน request แบบ GET นั้น parameter เหล่านี้ถูกเพิ่มเข้าไปตามหลัง URL ของ request ใน form ของ query string ใน

request แบบ POST parameter เหล่านี้จะอยู่ในส่วนของ body ของ request ในกรณีใด ๆ ก็ตาม parameter จะอยู่ในรูปคู่ของชื่อ parameter กับค่าของ parameter (key/value) อย่างไรก็ตาม HTTP ไม่ได้กำหนดว่าชื่อของ parameter จะต้องเป็น unique ดังนั้นบาง parameter จึงสามารถมีค่าเป็นชุด ๆ ได้ ซึ่งอาจเกิดขึ้นเนื่องจาก form ของ HTML มีลักษณะแบบนี้

object ใดที่ implement interface ชื่อ HttpServletRequest (เช่น object ของ HTTP request ที่ถูกส่งจาก servlet engine) จะช่วยให้ servlet สามารถเข้าถึงข้อมูลที่เกี่ยวกับ request ไปจนถึง method ต่าง ๆ ต่อไปนี้เป็น method พื้นฐานที่เราจะต้องใช้เพื่อรับข้อมูลจาก form

getParameter() method

public String getParameter(String key)

method นี้จะพยายามหา parameter ตามชื่อที่ส่งมากับ query string แล้วจึงคืนค่าของ parameter นั้น ๆ กลับมา ถ้าค่าของ parameter นั้น ๆ มีหลายค่า ค่าที่ถูกส่งกลับมาจะเป็นค่าแรก

getParameterValues() method

public String[] getParameterValues(String key)

ถ้า parameter หนึ่งมีหลายค่าที่สามารถคืนกลับมาได้ เช่น set ของ checkbox เราควรใช้ method นี้ในการรับค่าทั้งหมดของ parameter นั้น

getParameterNames() method

public Enumeration getParameterNames()

method นี้จะคืนค่าเป็น object ชนิด Enumeration และชื่อของ parameter ทั้งหมด

2.4.2.5 HttpServletResponse Interface

public interface HttpServletResponse extends ServletResponse

servlet engine มี object ที่ implement interface นี้และส่งไปยัง servlet หรือ method ที่ให้บริการ servlet ที่ให้บริการนั้นสามารถดัดแปลง response header และส่งผลลัพธ์คืนไปโดย object ชนิด HttpServletResponse และ method ของ object นี้ ต่อไปนี้เป็น method พื้นฐาน 2 method ที่เราต้องรู้

setContentType() method

public void setContentType(String type)

ก่อนจะเขียน response ส่งกลับไปยังผู้เรียก method นี้ต้องถูกเรียกเสียก่อน เพื่อจะกำหนด MIME type ของ HTTP response โดยจะเป็น MIME type ใดก็ได้ เมื่อเราเขียน HTML ส่งกลับไปยัง browser MIME type ควรจะเป็น text/html

getWriter() method

public PrintWriter getWriter() throws IOException

method นี้จะคืนค่าเป็น object ชนิด PrintWriter ที่เราสามารถใช้ในการเขียน response รูปแบบ text เพื่อตอบกลับไปยังผู้เรียก PrintWriter แปลงตัวอักษรชนิด Java's internal UniCode ให้ถูกต้องอย่างอัตโนมัติ เพื่อให้ client สามารถอ่านข้อความเหล่านั้นได้ อนึ่ง การใช้ object ชนิด PrintWriter เราจะเขียนข้อมูลลงไปยัง object ชนิด response โดยใช้ method ชื่อ println(String txt)

getOutputStream() method

public ServletOutputStream getOutputStream() throws IOException

method นี้คืนค่าเป็น ServletOutputStream ซึ่งเป็น subclass ของ java.io.OutputStream object นี้สามารถใช้ในการส่งข้อมูลชนิด binary กลับไปยัง client

setHeader() method

public void setHeader(String name, String value)

method นี้สามารถใช้ในการกำหนด HTTP header ที่จะส่งกลับไปยัง client ถึงแม้จะมีหลาย method ที่ใช้ในการกำหนด header แต่ในบางสถานการณ์เราอาจจะต้องการใช้ method นี้แทน

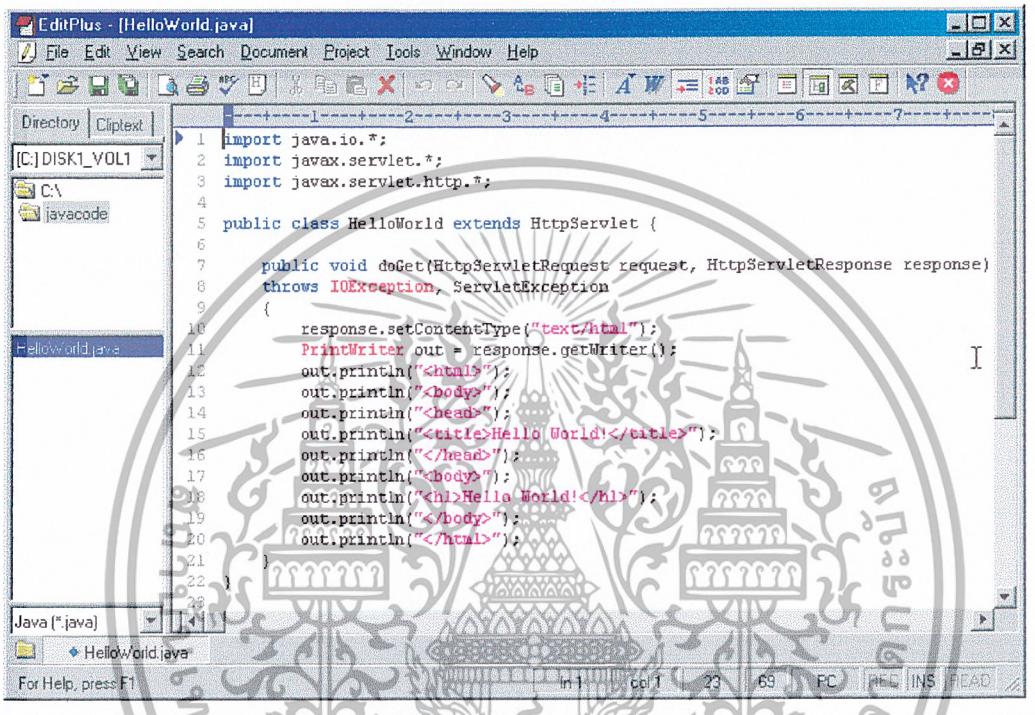
2.4.3 การทดสอบ Servlet

การที่เราจะทดสอบ servlet ได้เราจะต้องมีซอฟต์แวร์ JSDK และ Apache Tomcat ซึ่งสามารถโหลดได้จาก www.java.sun.com และ http://jakarta.apache.org ตามลำดับ การติดตั้งจะต้องทำตามลำดับคือ ติดตั้ง JSDK ก่อนแล้วจึงติดตั้งซอฟต์แวร์ Apache Tomcat ในที่นี้จะใช้ J2DK 1.4.0_01 และ Apache Tomcat 4.0.4 ซึ่งเป็นเวอร์ชั่นใหม่ล่าสุดของทั้ง 2 ตัวในการทดสอบ servlet เมื่อเราติดตั้งทั้ง 2 ตัวเสร็จแล้วให้เรา set ตามต่อไปนี้

(1)ให้คัดลอกไฟล์ servlet.jar ซึ่งถูกบรรจุไว้ใน Apache Tomcat 4.0.4\common\lib ไปไว้ที่ j2sdk 1.4.0_01\jre\lib\ext เมื่อเราทำเสร็จแล้วก็จะมาเริ่มทำ servlet

(2) โดยเริ่มแรกเราจะต้องทำการ Start Tomcat ก่อนที่จะเริ่มทำ servlet ทุกครั้ง เพราะการ Start Tomcat จะเป็นการจำลองเครื่องเราให้กลายเป็นเครื่อง Server

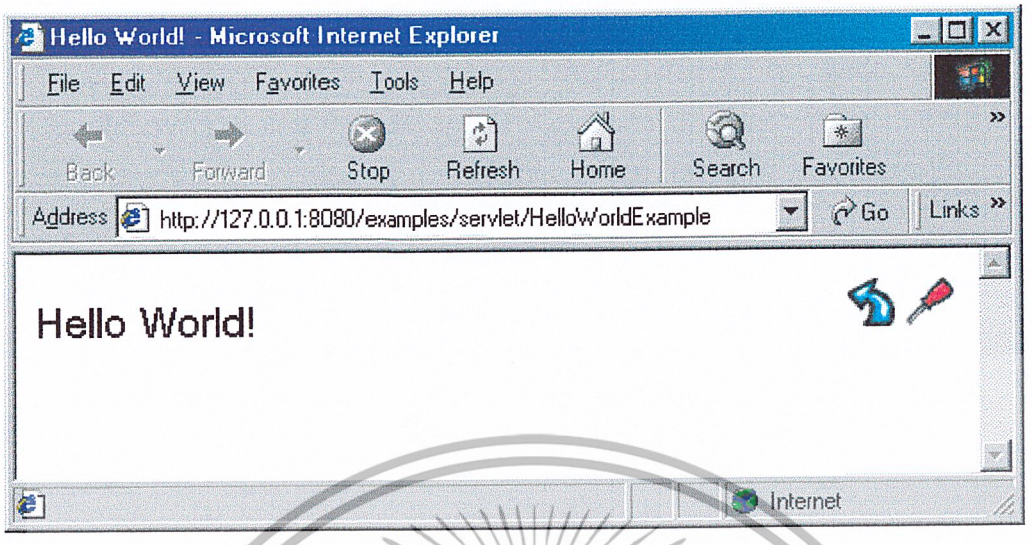
(3) พิมพ์โค้ดต่อไปนี้ลงไปใน Text Editor ในที่นี้จะใช้ Editplus ดังรูป 2.17 เสร็จแล้วให้จัดเก็บไฟล์แล้ว compile เหมือนปกติ



รูปที่ 2.17 แสดงตัวอย่างการเขียนโปรแกรม Servlet

(4) เมื่อ compile โปรแกรมแล้วเราจะได้ไฟล์ .class แล้วจึงนำไฟล์ .class ที่ได้ไปเก็บในไดเรกทอรี Apache Tomcat 4.0.4\webapps\examples\Web-inf\classes

(5) ดูการประมวลผลของ servlet ให้เปิด browser ขึ้นมาแล้วใส่ URL ดังรูป 2.18



รูปที่ 2.18 ผลการทำงานของโปรแกรม Servlet ในรูป 2.17

2.5 Template Engine

ในปัจจุบันเทคโนโลยี Template Engine นิยมใช้กันอย่างแพร่หลาย มีบริษัทจำนวนมากผลิตซอฟต์แวร์ประเภทนี้ เช่น WebMacro , Velocity , Freemaker ซึ่งในหัวนี้จะอธิบายโดยย่อเกี่ยวกับ Velocity

2.5.1 ชุดคำสั่งของ Velocity

2.5.1.1 การกำหนดค่าให้ตัวแปร

เราสามารถกำหนดค่าให้ตัวแปรโดยใช้คำสั่ง #set เช่น #set($a = “Velocity”)

2.5.1.2 การเขียน Comment

ในการเขียน Comment ของ Velocity จะคล้ายคลึงกับภาษา Java และ ภาษา C โดยมี 3 แบบคือ

1.Single line comment
คือ การเขียน comment บรรทัดเดียว โดยจะใช้สัญลักษณ์ ##
เช่น ##This is a single line comment

2.Multi line comment
คือ การเขียน comment ทีละหลายๆ บรรทัด ซึ่ง comment จะอยู่ระหว่างสัญลักษณ์ #* กับ #
เช่น #* This is
a multi line
comment *#

3.Comment block
คือการเขียน comment เพื่อบอกรายละเอียดเกี่ยวกับ version,ผู้ผลิต,วิธีการใช้ของโปรแกรม
โดย comment block จะอยู่ระหว่างสัญลักษณ์ #** กับ *#
เช่น #** information
@author Mohamed E.Fayad
@version 5
@readme *#

2.5.1.3 References

1. Variable

คือการอ้างถึงตัวแปรทั่วไป ซึ่งตัวแปรจะถูกนำหน้าด้วยเครื่องหมาย $

เช่น $foo = $bar

2. Properties

คือการอ้างถึง Attribute ของ Object

เช่น $purchase.Total

$customer.Address

3. Methods

คือการอ้างถึง method ของ object

เช่น $customer.getAddress()

$purchase.getTotal()

2.5.1.4 Condition

คือคำสั่งที่ใช้แสดงเงื่อนไข ซึ่งเหมือนกับในภาษาโปรแกรมอื่นๆ ทั่วไป

1.เงื่อนไขทางเดียว

ซึ่งสามารถเขียนโดยใช้คำสั่ง #if

เช่น #if($foo)

Velocity!

#end

2.เงื่อนไขสองทางขึ้นไป

สามารถเขียนโดยใช้คำสั่ง #elseif หรือ #else

เช่น #if($foo<100)

 Go North

else if ($foo>10)

Go East

#else

Go West

#end

2.5.1.5 Relational and Logical Operators

1.Relational

ใน Velocity มีเฉพาะการเปรียบเทียบความเท่ากันของค่าเท่านั้น โดยใช้เครื่องหมาย ==

เช่น #if($foo==$bar)

equal value!!!

2.Logical Operator

ใน Velocity มีเฉพาะ logical And(&&) กับ logical Or(||) เท่านั้น

2.5.1.6 Loops

ใน Velocity ใช้คำสั่ง Foreach ซึ่งทำหน้าที่เหมือน for loop ในภาษาโปรแกรมอื่นๆ

เช่น #foreach($product in $all Products)

$product

#end

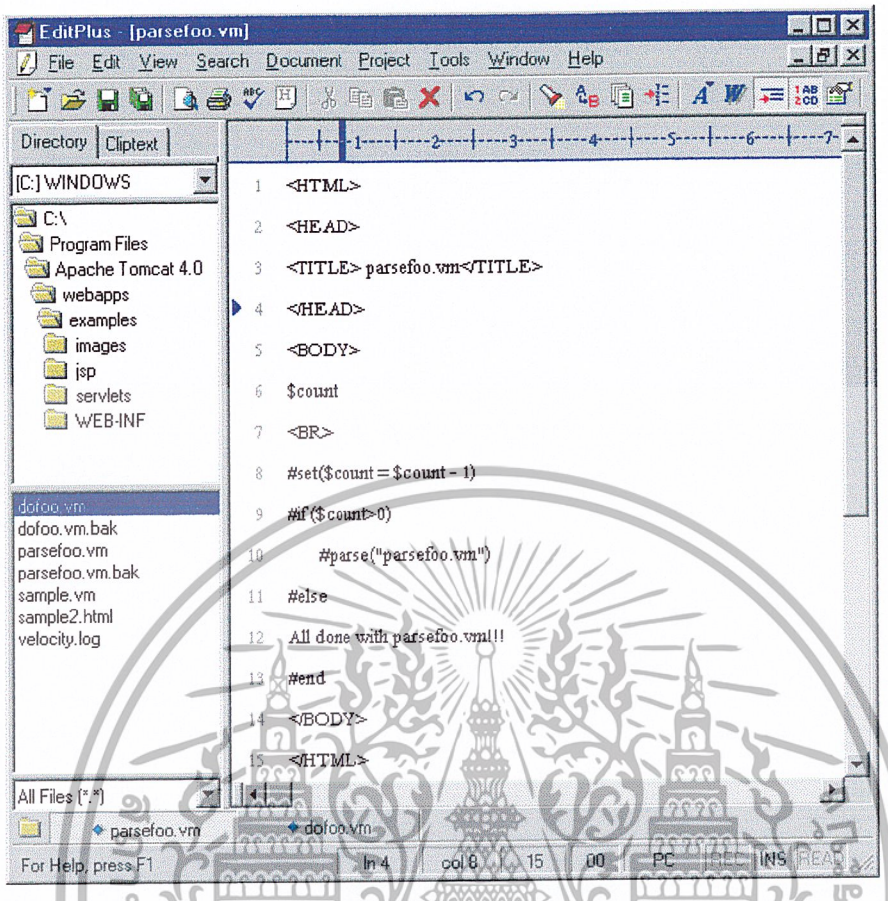
2.5.1.7 Include

คือการนำไฟล์อื่นๆ เข้ามาร่วมใช้ด้วย

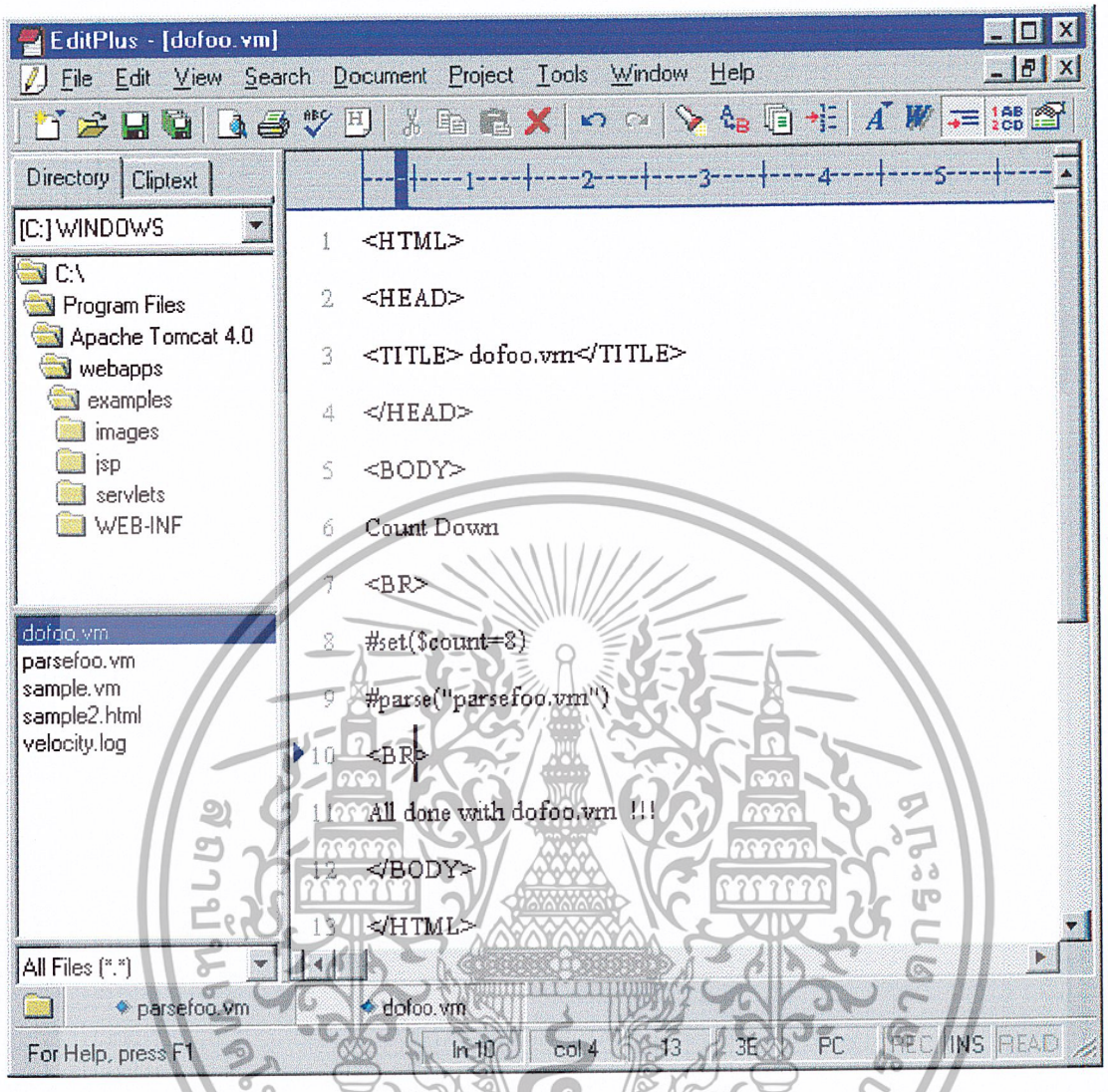
เช่น #include("one.gif", "two.txt", "three.htm")

2.5.1.8 Parse

Velocity ใช้คำสั่ง parse เพื่อการนำ file ที่เป็น VTL (ไฟล์ .vm , .html หรือ .htm) มาประมวลผลร่วมกัน ดังตัวอย่างในรูปที่ 2.19 และ 2.20 ซึ่งแสดงถึงโค้ดในไฟล์ VTL ชื่อ dofoo.vm และ parsefoo.vm และจากการใช้คำสั่ง parse จะให้ผลลัพธ์เป็น HTML ในรูปที่ 2.21



รูปที่ 2.19 โค้ดในไฟล์ VTL ชื่อ parsefoo.vm



รูปที่ 2.20 โค้ดในไฟล์ VTL ชื่อ dofoo.vm

จากโค้ด dofoo.vm :

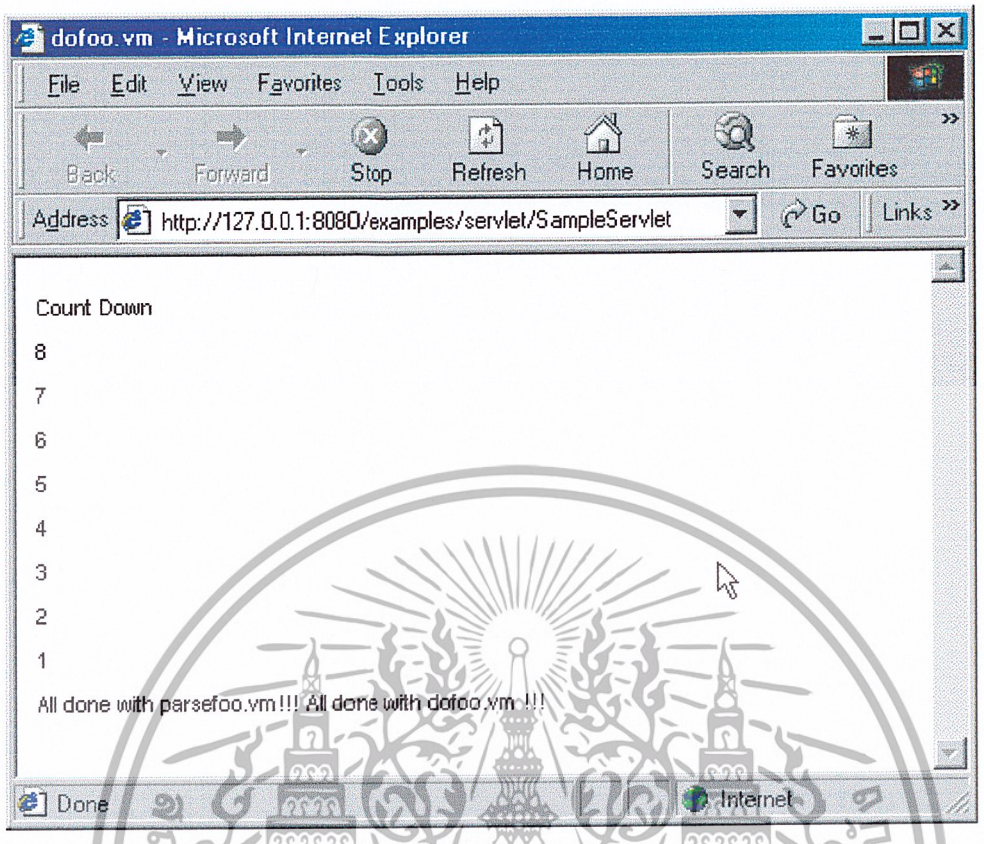
บรรทัด 8 : เซตค่าตัวแปร count เป็น 8

บรรทัด 9 : เรียกไฟล์ VTL ชื่อ parsefoo.vm ให้ประมวลผล

จากโค้ด parsefoo.vm

บรรทัด 6 : แสดงค่าข้อมูลของตัวแปร ซึ่งจะให้ผลเป็น 8 (ดูรูปที่2.19) ในรอบแรกของการทำงานของ parsefoo.vm

บรรทัด 8-13 : ค่า count ลดลงทีละ 1 แล้วเรียกตัวเองทำงาน โดยขณะที่ count > 0 ก็แสดงค่าของ count (7,6,5,…,1) และในที่สุดในรอบสุดท้ายจะแสดงข้อความ " All done with parsefoo.vm!!! "



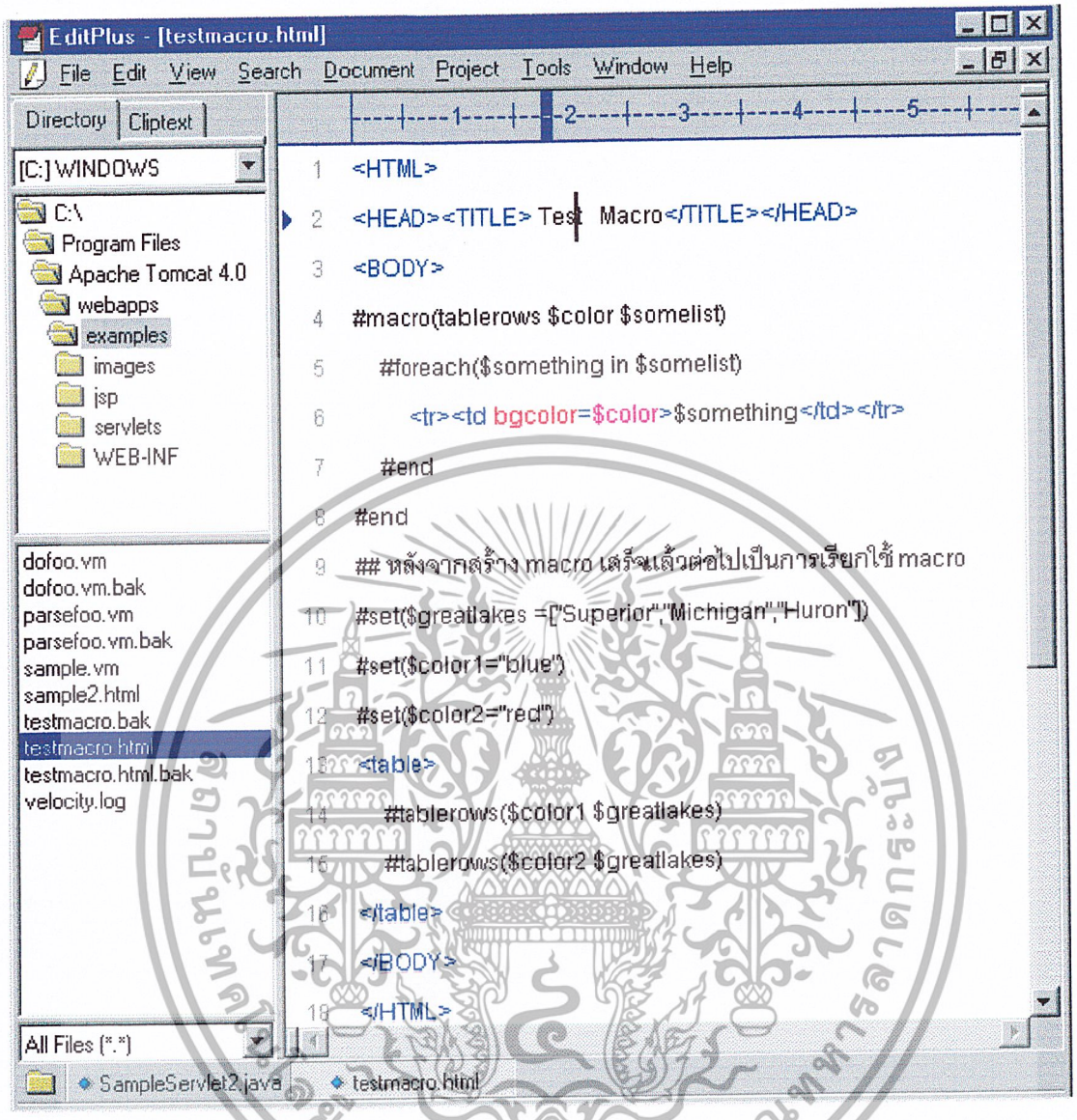
รูปที่ 2.21 Output จากการ parse ไฟล์ VTL

2.5.1.9 stop

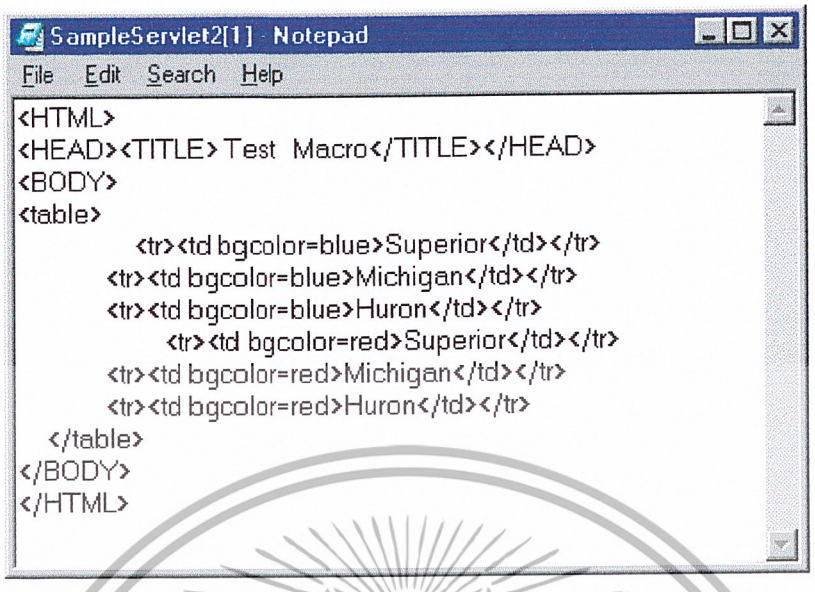
คือ การหยุด execute โดยคำสั่งที่ใช้ คือ #stop

2.5.1.10 macro

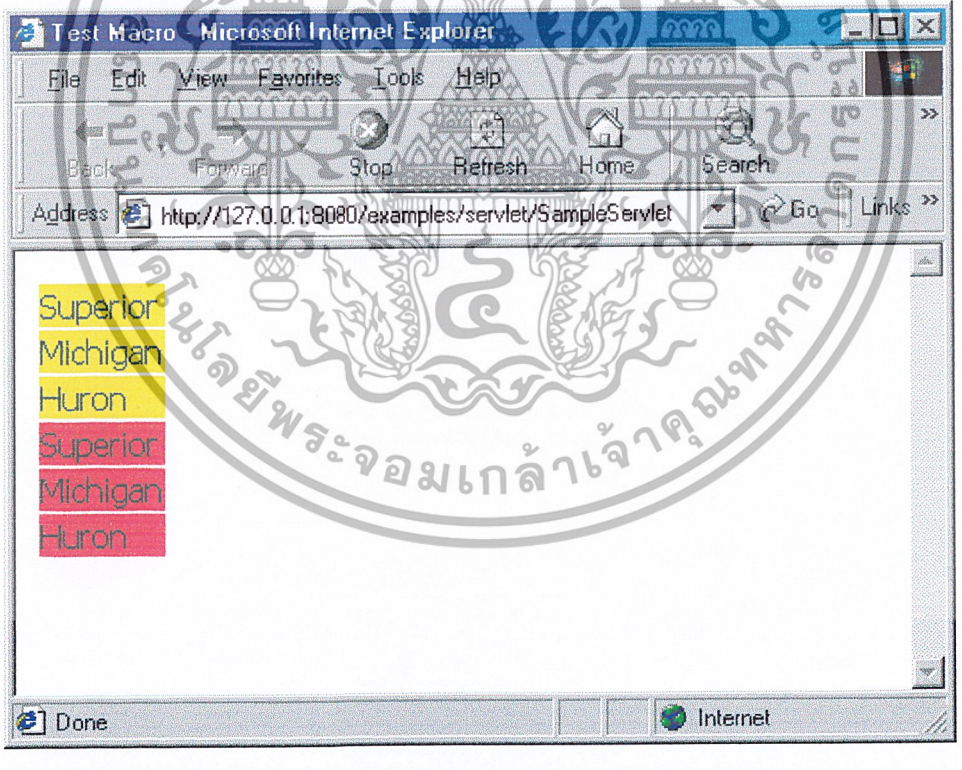
Velocity มีวิธีการเขียน macro ที่เรียกว่า Velocimacros ซึ่งคล้ายกับ Macro ในภาษาโปรแกรมอื่นๆ การสร้าง macro ใน Velocity นี้จะใช้คำสั่ง #macro ดังตัวอย่างในรูป 2.22



(ก) source code ในภาษา Velocity



(ข) ผลลัพธ์ในรูปของ HTML



(ค) ผลลัพธ์ใน browser

รูปที่2.22 ตัวอย่างการใช้ macro

โค้ดในรูป 2.22 (ก) จะให้ผลลัพธ์เป็น HTML ในรูป 2.22 (ข) ซึ่งแสดงผลใน browser ในรูป 2.22 (ค)

คำอธิบายการใช้ macro

กำหนด macro ชื่อ tablerows ที่มี 2 พารามิเตอร์ คือ color และ somelist ซึ่ง macro ทำหน้าที่สร้าง row ใน table ที่มีสี background (bgcolor) ตามที่กำหนดด้วย color และแสดงข้อความตามที่กำหนดใน array somelist

2.5.1.11 Math operator

โอเปอเรเตอร์ทางคณิตศาสตร์ใน Velocity แสดงไว้ในตารางที่ 2.2

ตารางที่2.2 โอเปอเรเตอร์ทางคณิตศาสตร์ของ Velocity

Operator	Description
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หาค่าเศษของการหาร (mod)

2.5.1.12 Range Operator

เป็นการกำหนดขอบเขตที่ต้องการ ซึ่งมักใช้ร่วมกับคำสั่ง #set , #foreach เช่น

```
#set($foo in[1..5])
#foreach($i in $foo)
    $i
#end
```

คำอธิบายโค้ด : กำหนดชื่อ range ชื่อ foo สามารถอยู่ในช่วง 1 ถึง 5 การใช้ foreach เป็นการกำหนดให้ตัวแปร i อยู่ในช่วง range foo ดังนั้น โค้ดนี้จะให้ผลลัพธ์เป็น 1 2 3 4 5

2.5.2 การทดสอบ Template Engine

การทดสอบ Template Engine จะต้องมีการติดตั้งซอฟต์แวร์ต่อไปนี้ก่อน

1. J2SDK ของ Java
   - ในที่นี้จะใช้ J2SDK เวอร์ชั่น 1.4.0_01

2. Web Server ที่สามารถรัน Java Servlet
   - ในที่นี้จะใช้ Apache Tomcat เวอร์ชั่น 4.0.4

3. Template Engine สำหรับ Java Servlet
   - ในที่นี้จะใช้ Velocity 1.1 เป็นตัวทดสอบซึ่งสามารถ download ได้ที่ http://apache.org

เมื่อติดตั้งซอฟต์แวร์ทั้ง 3 ตัวแล้ว การทดสอบจะเริ่มต้นโดยทำการ Start เซิร์ฟเวอร์ Tomcat ตัวอย่างที่ใช้ทดสอบนี้จะอาศัยโค้ดภาษา Java Servlet ดังรูปที่ 2.23

```
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.Properties;
import java.util.Vector;
import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.velocity.Template;
import org.apache.velocity.context.Context;
import org.apache.velocity.servlet.VelocityServlet;
import org.apache.velocity.app.Velocity;
import org.apache.velocity.exception.ResourceNotFoundException;
import org.apache.velocity.exception.ParseErrorException;

public class SampleServlet extends VelocityServlet
{
    protected Properties loadConfiguration(ServletConfig config )
        throws IOException, FileNotFoundException
    {
```

```
    Properties p = new Properties();
    String path = config.getServletContext().getRealPath("/");
    if (path == null)
    {
       System.out.println(" SampleServlet.loadConfiguration() : unable to "
                   + "get the current webapp root.  Using '/'. Please fix.");
       path = "/";
    }
    p.setProperty( Velocity.FILE_RESOURCE_LOADER_PATH,  path );
    p.setProperty( "runtime.log", path + "velocity.log" );
    return p;
}
public Template handleRequest( HttpServletRequest request,
        HttpServletResponse response, Context ctx )
{
    String p1 = "Bob";
    String p2 = "Harold";
    Vector personList = new Vector();
    personList.addElement( p1 );
    personList.addElement( p2 );
    ctx.put("theList", personList );
    Template outty = null;
    try
    {
       outty =  getTemplate("sample.vm");
    }
    catch( ParseErrorException pee )
    {
       System.out.println("SampleServlet : parse error for template " + pee);
    }
    catch( ResourceNotFoundException rnfe )
```

```
        {
            System.out.println("SampleServlet : template not found " + rnfe);
        }
        catch( Exception e )
        {
            System.out.println("Error " + e);
        }
        return outty;
    }
}
```

รูปที่2.23 แสดงตัวอย่างโค้ดภาษา Java Servlet

วิธีการทดสอบ

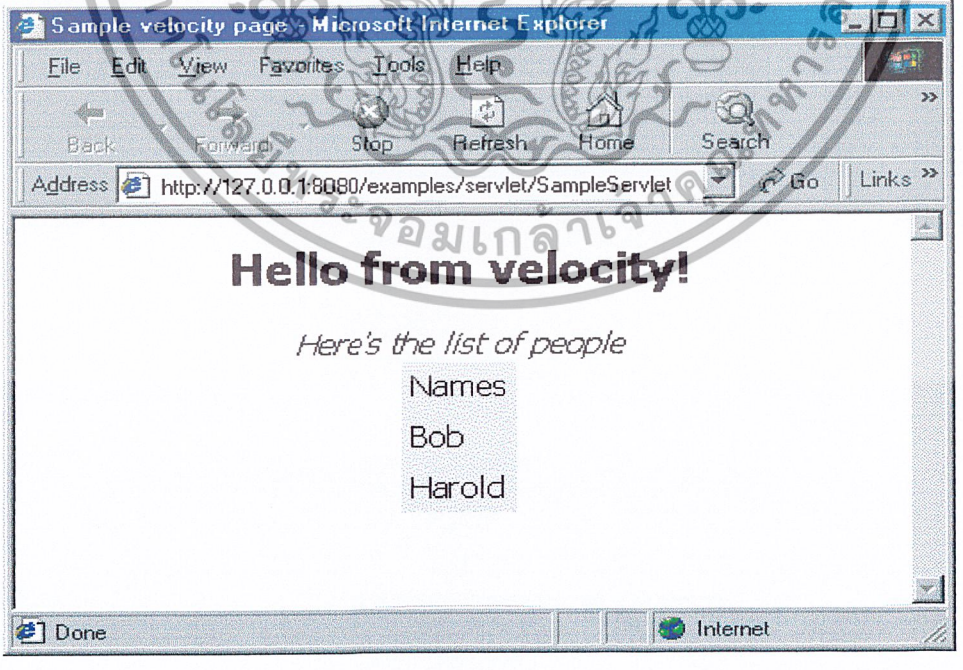
1. คอมไพล์ไฟล์ .java ให้เป็น .class แล้วนำไปเก็บไว้ที่ไดเรกทอรี Apache Tomcat 4.0\webapps\examples\WEB-INF\classes
2. สร้างเอกสาร VTL ที่จะทำกับเอกสาร Servlet ตัวอย่างเอกสาร VTL จะเป็นโค้ดภาษา html ดังรูปที่ 2.24
3. เอกสารในข้อ 2 จะต้องเก็บบันทึกไว้ในไดเรกทอรี Apache Tomcat 4.0\webapps\examples
4. ดูผลลัพธ์ของการทดสอบโดยใช้ browser เรียกดู Servlet ดังรูปที่ 2.25

```
<html>
<head><title>Sample velocity page</title></head>
<body bgcolor="#ffffff">
<center>
<h2>Hello from velocity!</h2>
<i>Here's the list of people</i>
<table cellspacing="0" cellpadding="5" widht="100%">
  <tr>
    <td bgcolor="#eeeeee" align="center">
      Names
```

```
    </td>
  </tr>
  #foreach ($name in $theList)
  <tr>
    <td bgcolor="#eeeeee">$name</td>
  </tr>
  #end
</table>
</center>
</html>
```

รูปที่ 2.24 แสดงตัวอย่างโค้ด html

ผลลัพธ์ของ Servlet ในรูปที่ 2.25 เกิดจากโค้ด SampleServlet เรียกใช้ Template “Sample.vm” ในเมธอด handleRequest ด้วยคำสั่งว่าที่ใช้เมธอด getTemplate() ของคลาส VelocityServlet



รูปที่ 2.25 แสดงผลลัพธ์ของการทดสอบ Template Engine

บทที่ 3
โครงสร้างหลักของซอฟต์แวร์

การพัฒนาซอฟต์แวร์ Template Engine สำหรับ Java Servlet เป็นซอฟต์แวร์ที่ทำให้ Java programmer และ Web designer สามารถพัฒนาโปรแกรมไปพร้อมๆกันได้ ซึ่งจะเป็นไปตามหลักของ MVC (Model-View-Controller) โดยซอฟต์แวร์ Template Engine นี้ จะมีส่วนของ Editor ซึ่งมีไว้สำหรับ Web designer เพื่อสร้างโค้ด HTML และ preview ก่อนจะ Upload ไปยัง Application Server

การวิเคราะห์ออกแบบซอฟต์แวร์ในโครงงานปัญหาพิเศษนี้ได้กำหนดขอบเขตเบื้องต้นไว้ในหัวข้อ 3.1 ความต้องการในการใช้ซอฟต์แวร์ Template Engine สำหรับ Java Servlet และกำหนดรายละเอียดใน หัวข้อ 3.2 ซึ่งอธิบายด้วย Conceptual Model (Class Diagram) ของซอฟต์แวร์ , หัวข้อ 3.3 อธิบายถึงการออกแบบ โดยใช้ Sequence Diagram

ตั้งแต่นี้ ระบบที่พัฒนาในโครงงานนี้ จะเรียกโดยย่อว่า SSEDIT (Simple Servlet Template Engine & Editor)

3.1 ความต้องการในการใช้ระบบ Template Engine สำหรับ Java Servlet

โดยทั่วไปการเขียนโปรแกรมทางฝั่ง Server โดยการใช้ Java Servlet นั้น ผู้เขียนจะต้องมีความชำนาญทั้งทางภาษา Java และ HTML ซึ่งส่วนใหญ่ Web Designer ซึ่งเป็นผู้ออกแบบเว็บเพจจะเขียนโค้ด HTML เพื่อพัฒนารูปแบบของ Page ให้ตรงตามความต้องการเท่านั้น แต่จะไม่สามารถสร้างไดนามิคเพจ ซึ่งต้องอาศัย โปรแกรมเมอร์เข้ามาช่วยในการพัฒนาโปรแกรม

โดยทั่วไปนั้นเว็บเพจมักมีความซับซ้อน เนื่องจากมีหลาย frame และในแต่ละ frame มีหลายองค์ประกอบ ซึ่งหากนำโค้ด HTML เหล่านี้มาเขียนรวมใน Java Servlet ก็จะยิ่งเป็นการเพิ่มความซับซ้อนของโปรแกรมให้มากขึ้น อีกทั้งหากเราต้องการแก้ไขโค้ด HTML เพียงบางส่วน ก็จะส่งผลให้ต้อง compile ตัว Java Servlet ใหม่ด้วย ทั้งๆ ที่ไม่ได้แก้ไขส่วนที่เป็น Java เลย

ด้วยสาเหตุนี้จึงพัฒนา "Template Engine" เพื่อให้ Java Programmer ทำงานแยกจาก Web Designer โดยทั้งสองส่วนสามารถทำงานไปพร้อมๆ กันได้

3.1.1 Requirements (Domain)

1.โปรแกรมจะมีโค้ดบางส่วนเพิ่มเติมจากโค้ด servlet เดิมที่มีอยู่แล้ว เพื่อใช้งานร่วมกับ Template Engine ที่เราสร้างขึ้น ถึงจะทำการ compile โปรแกรมได้

2. มีอิดิเตอร์สำหรับสร้างโค้ด HTML

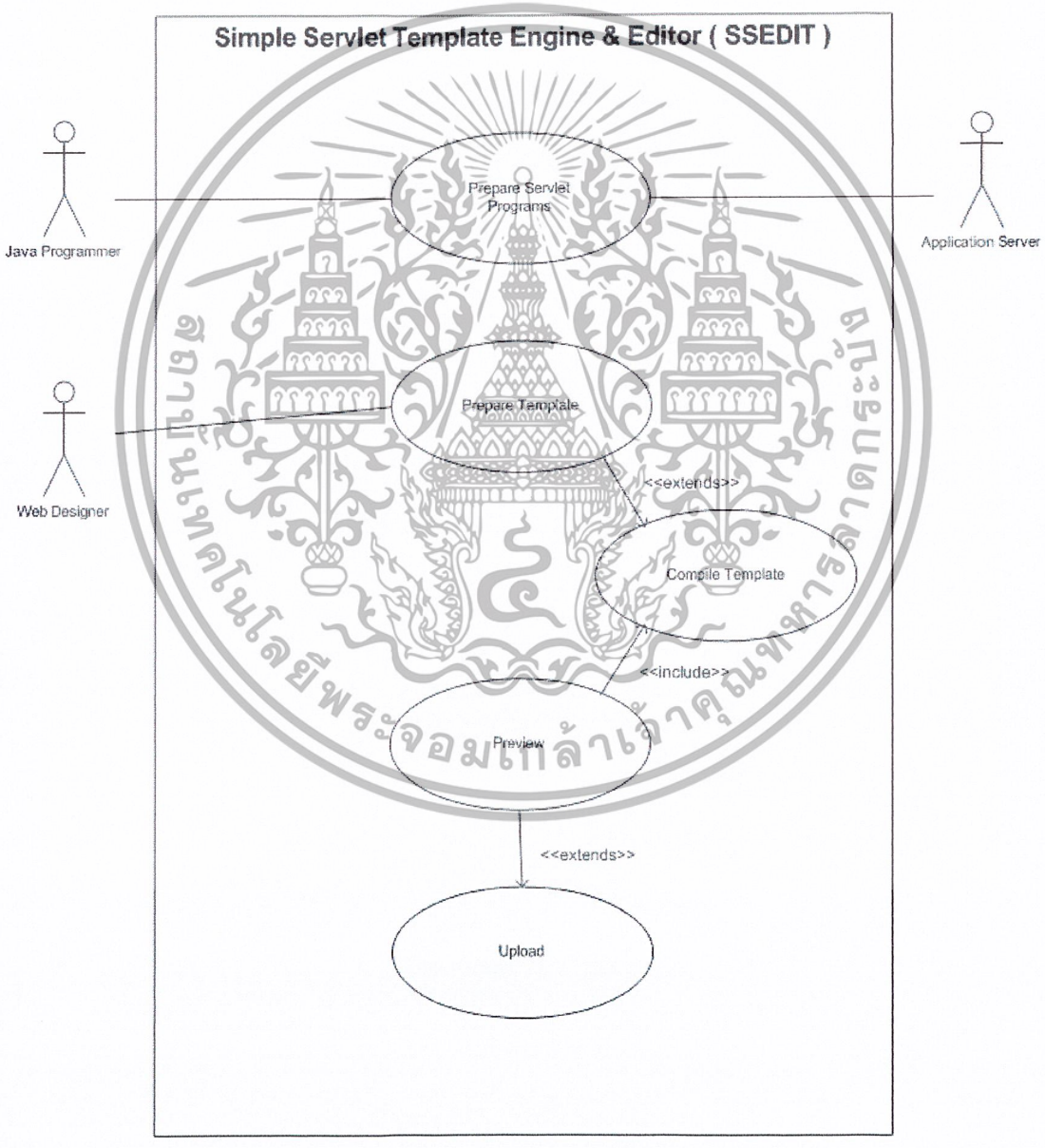
3. โค้ด HTML สามารถมีคำสั่งโปรแกรมอย่างง่าย เช่น คำสั่ง loop, คำสั่ง if เป็นต้น

4. สามารถทำการ preview ดูผลของเว็บเพจก่อน upload ไปยัง application server ได้

5. หากเว็บเพจที่กำลังทำการพัฒนานั้นยังไม่เสร็จสมบูรณ์ ก็สามารถ upload เก็บไว้ที่ application server ก่อนได้ เพื่อให้สามารถแก้ไขและพัฒนาด้วยการ download จากที่ใดก็ได้

6. เมื่อพัฒนาโค้ด HTML จนเสร็จสมบูรณ์แล้ว จึงทำการ upload ลง application server

3.1.2 การจัดระบบโดย UML



รูปที่ 3.1 Use Case Diagram ของซอฟต์แวร์

3.1.2.1 นิยาม Actors

1. Actors : Java Programmer หมายถึง ผู้แก้ไขและพัฒนาโปรแกรมในส่วนของภาษา Java

2. Actors : Web Designer หมายถึง ผู้แก้ไขและพัฒนาโปรแกรมในส่วนของ โค้ด HTML

3. Actors : Application Server หมายถึง Server ที่เป็นโค้ด servlet และโค้ด HTML อีกทั้งเป็นที่ตีความโค้ด servlet

3.1.2.2 คำอธิบาย Use Case

Use Case : **Prepare Servlet Programs**

Actors : Java Programmer, Application Server

คำอธิบาย : Java Programmer เขียนโค้ด servlet โดยต้อง extends Template Engine ที่เราสร้างขึ้น เมื่อสร้าง code เสร็จแล้วก็จะ upload ไปยัง Application Server

Requirement : 1

Use Case : **Prepare Template**

Actors : Web Designer

คำอธิบาย: Web Designer ทำการเขียนและพัฒนาเว็บเพจโดยเขียนขึ้นจากอิดิเตอร์ที่เราสร้างขึ้น อีกทั้งสามารถใช้คำสั่งโปรแกรมอย่างง่าย เช่นคำสั่ง loop หรือ คำสั่ง if เป็นต้น

Requirement : 2,3

Use Case : **Compile Template**

Actors : Application Server

คำอธิบาย : ก่อนที่จะทำการ Preview ก็จะต้องมีการคอมไพล์ก่อน ถึงจะสามารถดูผลที่เกิดขึ้นได้

Requirement : 4

Use Case : **Preview**

Actors : Application Server

คำอธิบาย : เมื่อ Web Designer สร้างและพัฒนาโค้ด HTML เสร็จแล้วจะสามารถทำการ preview ดูผลที่เกิดขึ้น ก่อนที่จะทำการ upload ไฟล์นั้นไปยัง Application Server ได้ ซึ่งถ้า Web Designer ยังพัฒนาโปรแกรมไม่เสร็จสมบูรณ์ก็สามารถเก็บโค้ดส่วนนี้ไว้ใน Application Server ได้ โดยจะมีไดเรกทอรีสร้างขึ้น สำหรับการ preview จึงทำให้ผู้พัฒนาโปรแกรมสามารถ download เพื่อแก้ไขจากที่ใดก็ได้

Requirement : 4,5

Use Case : **Upload**

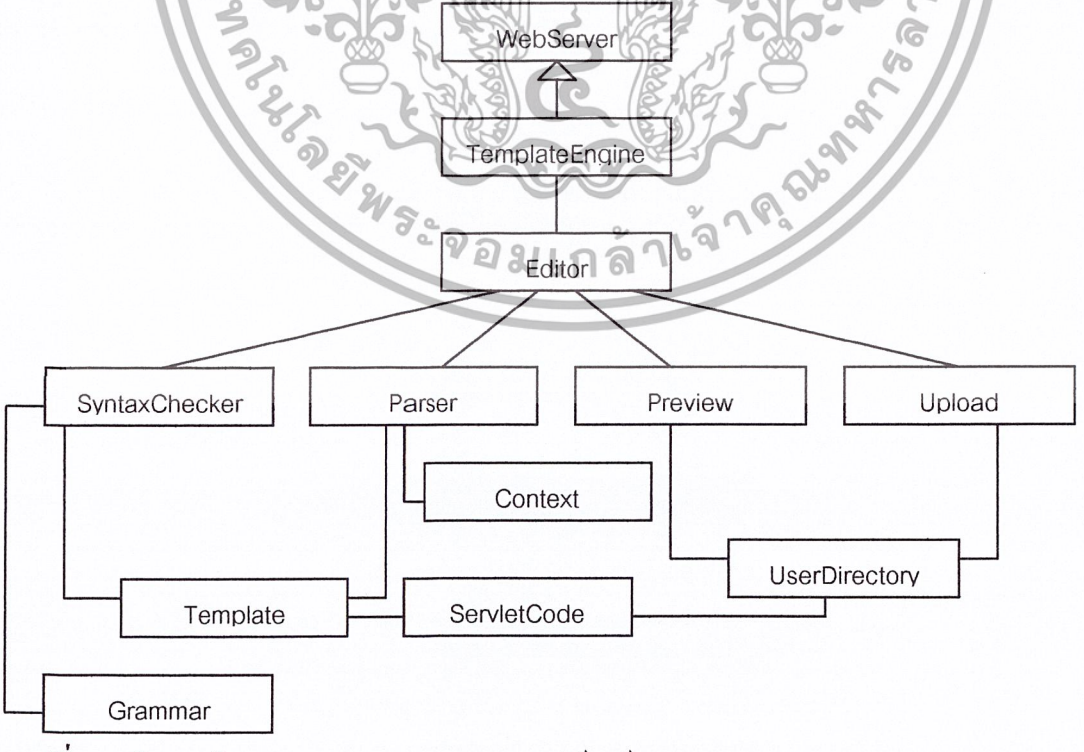
Actors : Application Server

คำอธิบาย : เมื่อ preview ได้ผลตามต้องการแล้ว ก็จะ Update ลง Application Server เพื่อใช้งานจริง

Requirement : 6

3.2 การออกแบบซอฟต์แวร์โดย Class Diagram

จากหัวข้อ 3.1 เราสามารถเขียน Conceptual Model ซึ่งแสดง class หลักของซอฟต์แวร์ได้ดังรูปที่ 3.2 และตาราง 3.1 อธิบายความหมายของ class เหล่านั้น



รูปที่ 3.2 โครงสร้าง Conceptual Model ของซอฟต์แวร์

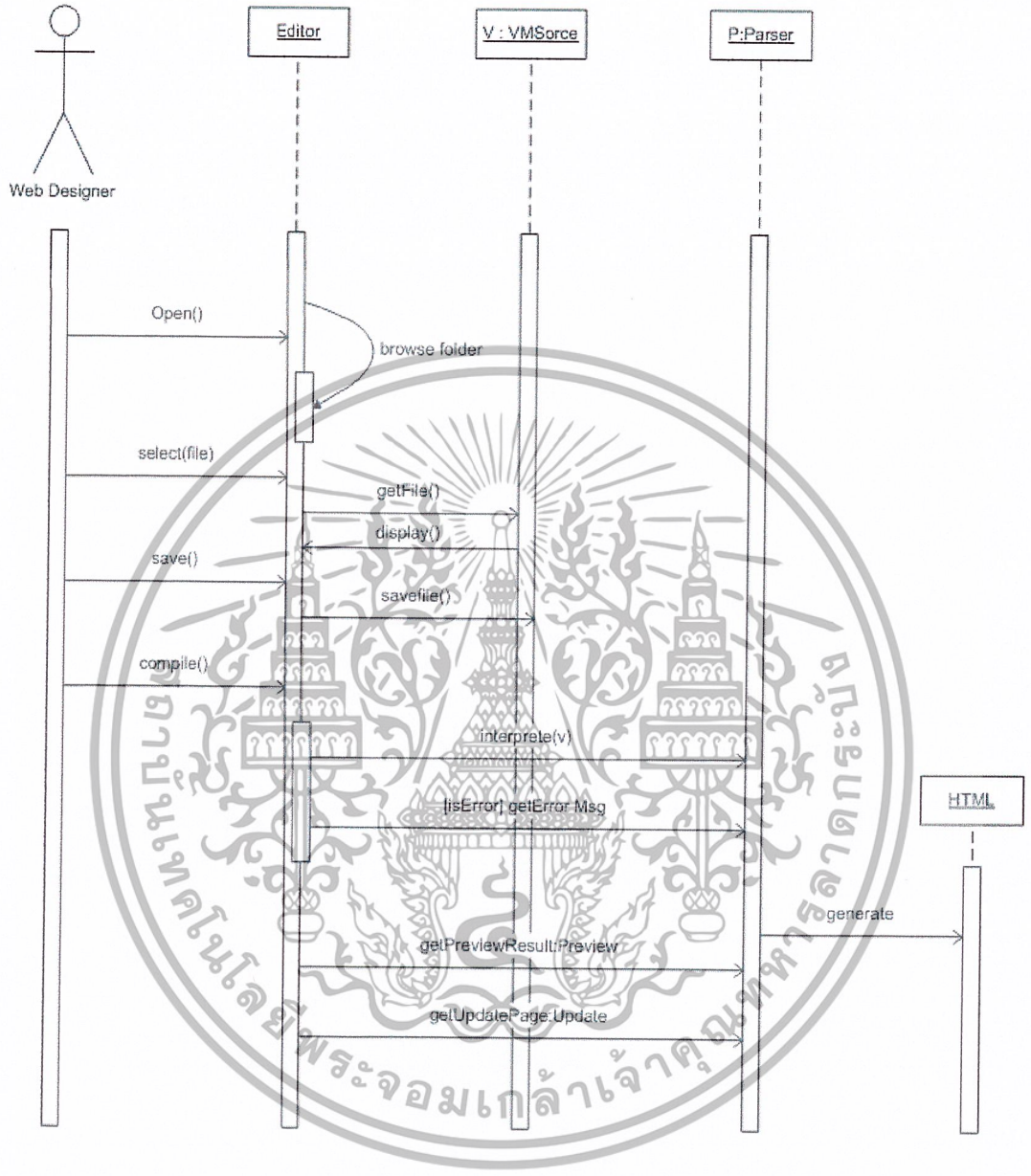
ตาราง 3.1 ความหมายของ Class หลัก

ชื่อคลาส	หน้าที่การทำงาน
WebServer	เป็นคลาส HttpServlet เดิมที่มีอยู่แล้ว ใน Java Servlet
TemplateEngine	เป็นคลาส Template Engine ที่สร้างขึ้นเพื่อให้สามารถพัฒนาโค้ด Java และโค้ด HTML ไปพร้อมๆกันได้ โดยจะต้องเป็น subclass ของ HttpServlet และทำการ overriding และ เพิ่มเติมบางส่วน
Editor	เป็นคลาสที่แสดง User Interface เพื่อให้ผู้พัฒนาโปรแกรมสามารถพัฒนาและปรับปรุงโปรแกรมผ่านอิดิเตอร์ของซอฟต์แวร์ Template Engine ได้
Preview	เป็นคลาสที่แสดงผลการรันผ่าน Web browser ก่อนจะทำการการ save file
Upload	เป็นคลาสที่ทำการ Upload ไฟล์ที่เสร็จสมบูรณ์แล้ว ไปยัง Server
UserDirectory	เป็นคลาสที่ระบุตำแหน่งไดเรกทอรีใน Application Server ของผู้ใช้แต่ละคน
SyntaxChecker	เป็นคลาสที่ทำการตรวจสอบ Syntax ของโปรแกรมที่สร้างขึ้น ซึ่งจะต้องตรวจกับ grammar ของ Template Engine
Grammar	เป็นคลาสที่เก็บรวบรวม grammar ทั้งหมดไว้
Template	เป็นคลาสที่ระบุตำแหน่งของ HTML file
Parser	เป็นคลาสที่ทำหน้าที่แปลคำสั่ง และ generate โค้ด HTML
Context	เป็นคลาสที่เก็บชื่อ และค่าของตัวแปร ซึ่งใช้ร่วมกันระหว่าง Java files และ HTML files
ServletCode	เป็นคลาสของ Java code ซึ่งจะเป็น subclass ของคลาส TemplateEngine

3.3 การออกแบบซอฟต์แวร์โดยใช้ Sequence Diagram

Use case diagram ของซอฟต์แวร์ Template Engine สำหรับ Java Servlet ซึ่งแสดงรายละเอียดไว้ในหัวข้อที่ 3.1 ความต้องการในการใช้ซอฟต์แวร์ Template Engine สำหรับ Java Servlet สามารถนำแต่ละ Use case มาแสดงในลักษณะของ Sequence Diagram ดังรูปที่ 3.3

รูปที่ 3.3(ก) Sequence Diagram สำหรับ Use case " Prepare Servlet Programs"



รูปที่ 3.3(ข) Sequence Diagram สำหรับ Use case "Prepare template", "Compile Template", "Preview", และ "Upload"

บทที่4
โครงสร้างของระบบงาน

4.1 เครื่องมือที่ใช้ในการพัฒนาโปรแกรม

คอมพิวเตอร์ที่สามารถใช้ซอฟต์แวร์ Template Engine สำหรับ Java Servlet ควรจะต้องมีประสิทธิภาพดังต่อไปนี้

4.1.1 ฮาร์ดแวร์

PC ที่มีรายการฮาร์ดแวร์ที่จำเป็นดังตารางที่ 4.1

ตาราง 4.1 รายการฮาร์ดแวร์ที่จำเป็นต้องใช้สำหรับรันซอฟต์แวร์ SSEDIT

รายการฮาร์ดแวร์	รายละเอียด	คำแนะนำ
Processor	เป็น Processor Pentium ขึ้นไป	ควรใช้ Processor Pentium II เป็นอย่างต่ำ
Harddisk	มีขนาดมากกว่า 250 MB	ควรมีขนาด 500 MB เป็นอย่างต่ำ
หน่วยความจำ	อย่างต่ำ 32 MB	ควรมีขนาดอย่างน้อย 64 MB

4.1.2 ซอฟต์แวร์

ควรมีรายการซอฟต์แวร์ที่จำเป็นดังตารางที่ 4.2

ตาราง 4.2 รายการฮาร์ดแวร์ที่จำเป็นต้องใช้สำหรับรันซอฟต์แวร์ SSEDIT

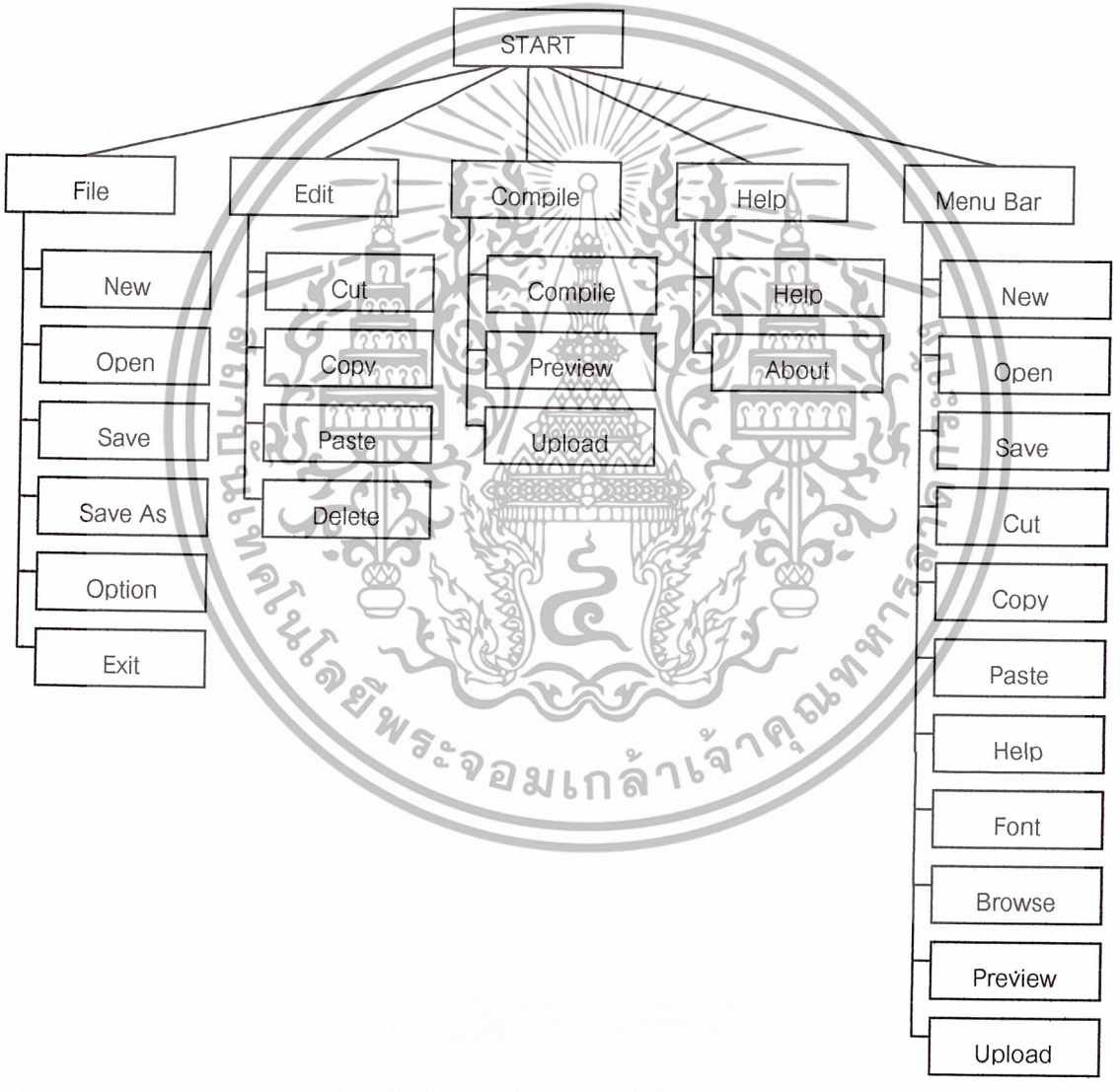
รายการซอฟต์แวร์	รายละเอียด	คำแนะนำ
ระบบปฏิบัติการ	Windows 98/ME/XP, Windows NT Server 4.0, Windows 2000 Server	ควรใช้ Windows NT Server หรือ Windows 2000 Server
J2SDK	J2SDK 1.3.0 ขึ้นไป	ควรใช้ J2SDK 1.4.0 ขึ้นไป
Application Server	ซอฟต์แวร์เว็บเซิร์ฟเวอร์ที่สนับสนุนการทำงานของ Servlet	สำหรับ Java Servlet ควรใช้ Tomcat Apache

4.2 การพัฒนาซอฟต์แวร์ในส่วนของอิดิเตอร์

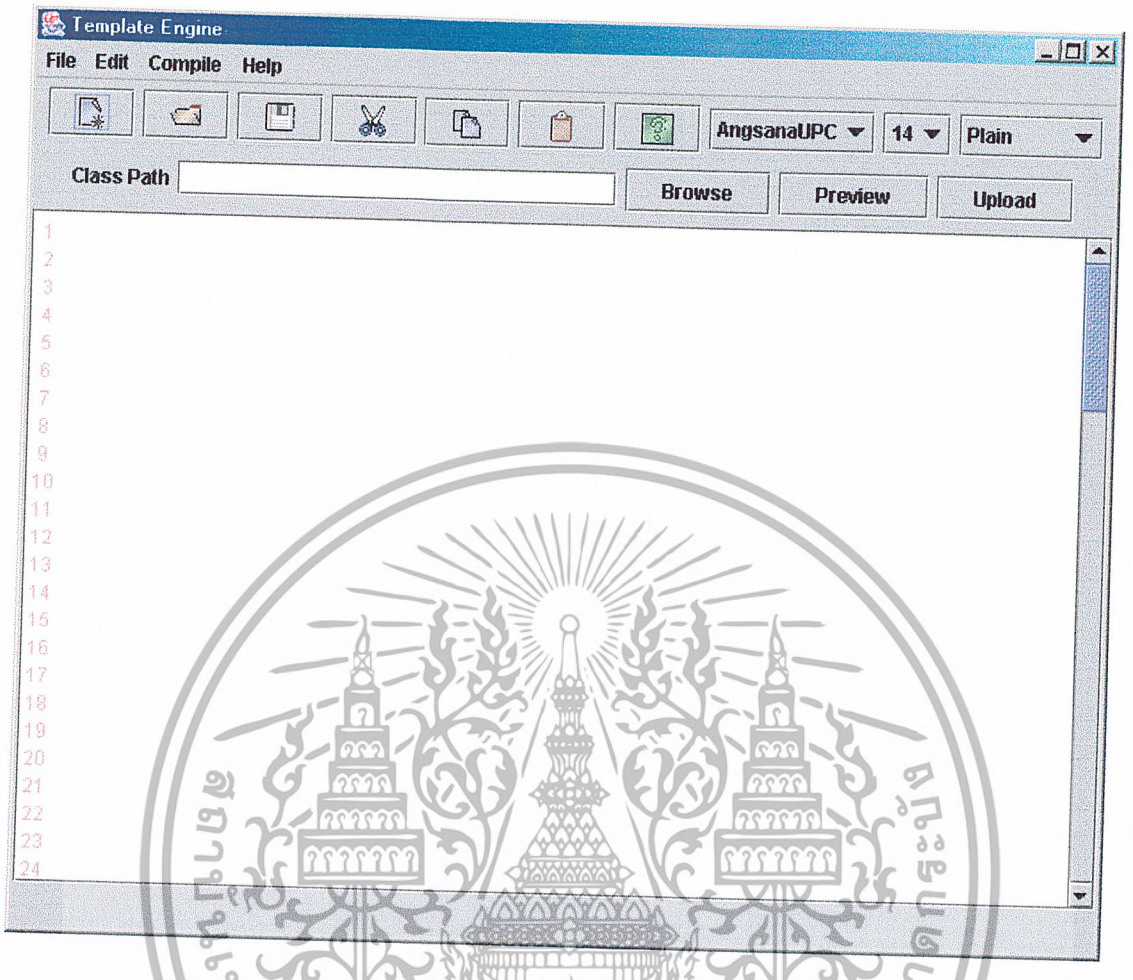
การพัฒนาอิดิเตอร์ทำขึ้นเพื่ออำนวยความสะดวกให้กับ Web Developer โดยสามารถ install ซอฟต์แวร์ Template Engine สำหรับ Java Servlet ไว้ในที่ที่ต้องการ

4.2.1 โครงสร้างซอฟต์แวร์

ซอฟต์แวร์นี้มีโครงสร้างซอฟต์แวร์ (Program Structure Chart) ตามรูปที่ 4.1 ซึ่งเมื่อเปิดโปรแกรม Template Engine จะปรากฏอิดิเตอร์เพื่อให้พัฒนาโปรแกรม ดังรูปที่ 4.2



รูปที่ 4.1 Program Structure Chart ของซอฟต์แวร์ SSEDIT



รูปที่ 4.2 User Interface ของอิดิเตอร์สำหรับพัฒนาโปรแกรม

4.2.2 รายละเอียด Component ของซอฟต์แวร์

อธิบายรายละเอียดส่วนประกอบต่างๆของเมนูตาม Structure Chart

4.2.2.1 เมนู File

เมนู File จะประกอบด้วยเมนูย่อยที่สำคัญอยู่ 6 เมนูย่อย ซึ่งจะแสดงใน รูปที่4.3 โดยแต่ละเมนูย่อยมีหน้าที่ดังนี้

New - สร้างไฟล์ใหม่

Open - เปิดไฟล์เดิมที่มีอยู่แล้ว

Save - บันทึกไฟล์ที่กำลังแก้ไขอยู่โดยใช้ชื่อเดิม

Save As - บันทึกไฟล์โดยใช้ชื่อใหม่

Option - จัดเก็บค่าพาทของ FTP และ เว็บบราวเซอร์

Exit - ออกจากโปรแกรม

รูปที่ 4.3 เมนู File

4.2.2.2 เมนู Edit

เมนู Edit จะประกอบด้วยเมนูย่อยที่สำคัญอยู่ 4 เมนูย่อย ซึ่งจะแสดงใน รูปที่ 4.4 โดยแต่ละเมนูย่อยมีหน้าที่ดังนี้

Cut - ลบข้อมูลที่เลือกไว้ออก แต่จดจำข้อมูลนั้นไว้

Copy - สำเนาข้อมูลที่เลือกไว้

Paste - นำข้อมูลที่เลือกไว้มาแสดง

Delete - ลบข้อมูลที่เลือกไว้

รูปที่ 4.4 เมนู Edit

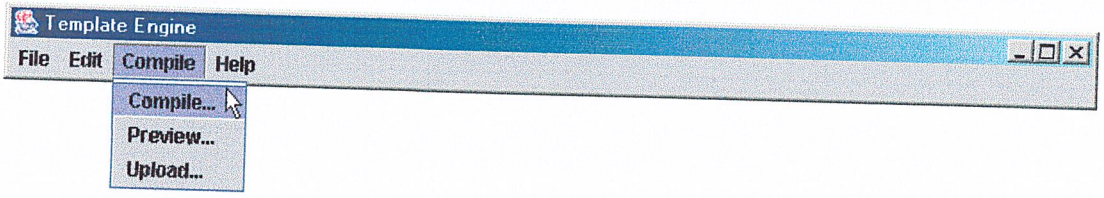
4.2.2.3 เมนู Compile

เมนู Compile จะประกอบด้วยเมนูย่อยที่สำคัญอยู่ 3 เมนูย่อย ซึ่งจะแสดง ในรูปที่4.5 โดยแต่ละเมนูย่อยมีหน้าที่ดังนี้

Compile - ตรวจสอบ Syntax ของไฟล์ HTML

Preview - คอมไพล์และทำการเปิดเว็บบราวเซอร์เพื่อดูผลลัพธ์

Upload - อัพโหลดไฟล์ที่เรียบร้อบแล้วสู่ Application Server โดยใช้ FTP



รูปที่ 4.5 เมนู Compile

4.2.2.4 เมนู Help

เมนู Help จะประกอบด้วยเมนูย่อยที่สำคัญอยู่ 2 เมนูย่อย ซึ่งจะแสดงใน
รูปที่4.6 โดยแต่ละเมนูย่อยมีหน้าที่ดังนี้

Help - แนะนำเกี่ยวกับการใช้ซอฟต์แวร์ Template Engine สำหรับ Java Servlet

About - แนะนำเกี่ยวกับ Version และ ผู้จัดทำซอฟต์แวร์นี้

รูปที่ 4.6 เมนู Help

4.2.2.5 เมนูบาร์

เมนู File จะประกอบด้วยเมนูย่อยที่สำคัญอยู่ 11 เมนูย่อย ซึ่งจะแสดงใน
รูปที่4.7 โดยแต่ละเมนูย่อยมีหน้าที่ดังนี้

New - สร้างไฟล์ใหม่

Open - เปิดไฟล์เดิมที่มีอยู่แล้ว

Save - บันทึกไฟล์ที่กำลังแก้ไขอยู่โดยใช้ชื่อเดิม

Cut - ลบข้อมูลที่เลือกไว้ออก แต่จดจำข้อมูลนั้นไว้

Copy - สำเนาข้อมูลที่เลือกไว้

Paste - นำข้อมูลที่เลือกไว้มาแสดง

Delete - ลบข้อมูลที่เลือกไว้

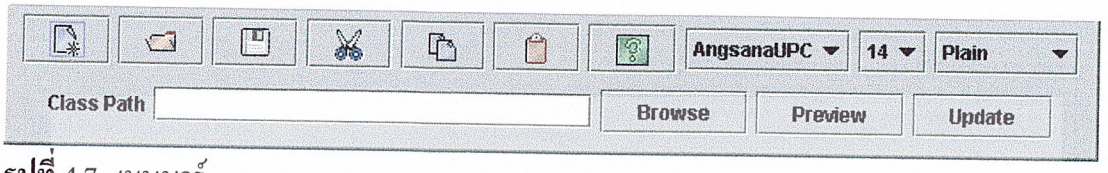
Help - แนะนำเกี่ยวกับการใช้ซอฟต์แวร์ Template Engine สำหรับ Java Servlet

Font - เปลี่ยนรูปแบบของตัวอักษรในอิดิเตอร์

Browse - ค้นหา Class Path ที่ใช้ร่วมกับ HTML ไฟล์

Preview - คอมไพล์และทำการเปิดเว็บบราวเซอร์เพื่อดูผลลัพธ์

Upload - อัพโหลดไฟล์ที่เรียบร้อบแล้วสู่ Application Server โดยใช้ FTP



รูปที่ 4.7 เมนูบาร์

4.3 ชุดคำสั่งของ SSEDIT

SSEDIT เป็นอิดิเตอร์ที่พัฒนาขึ้นในโครงงานนี้ ซึ่งมีชุดคำสั่งที่สามารถใช้ในไฟล์ HTML ดังตารางที่ 4.3

ตารางที่ 4.3 ชุดคำสั่งของ SSEDIT

คำสั่ง	ความหมาย
#set (variable = value)	เซตค่าให้ตัวแปร เช่น #set ($I = “name”) #set ($I = 5)
#for (initial_loop operation final_loop) #endloop	คำสั่งในการทำซ้ำ เช่น #for ($I &le 5) #endloop #for ($I &lt $J) #endloop
#if (variable operation value,variable) #endif	คำสั่งเงื่อนไข #if ($I &eq 5) #endif #if ($I &ne $J) #else #endif

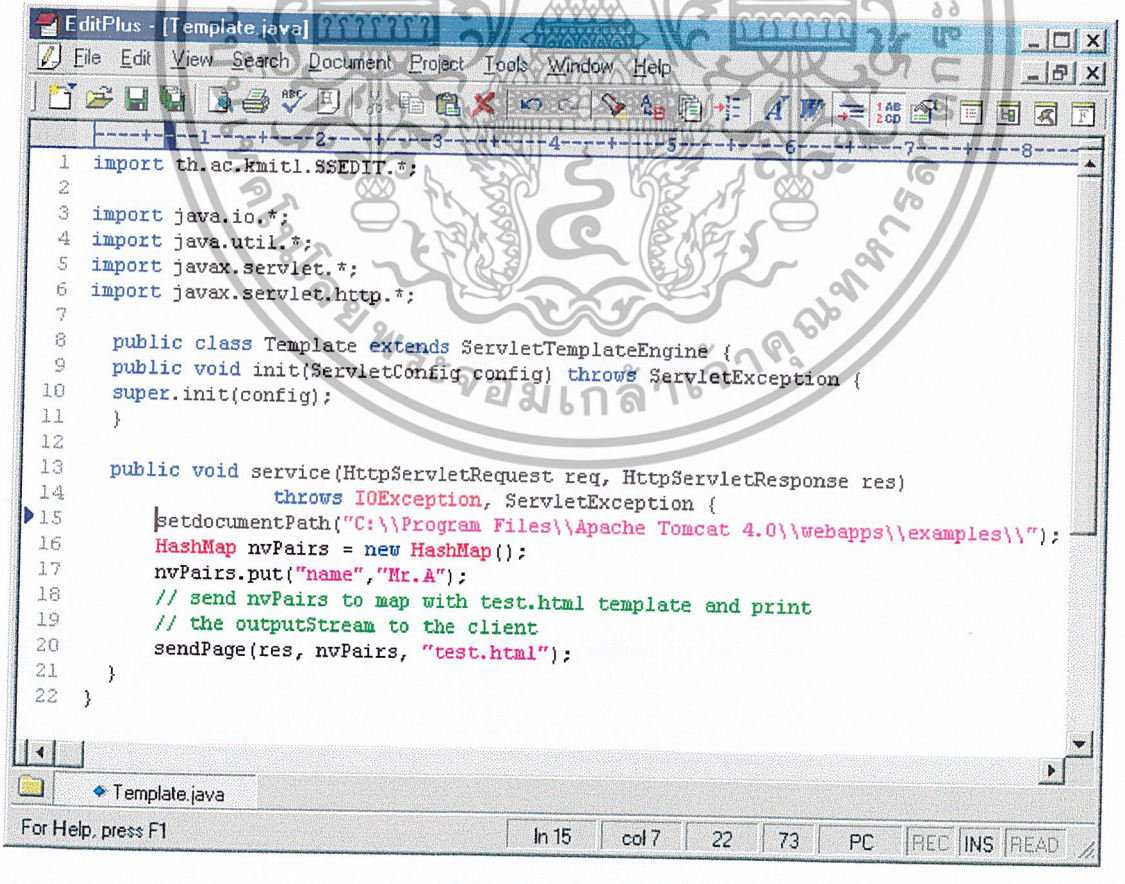
โดย operation ทั้งหมดที่สามารถใช้ได้ใน SSEDIT มีทั้งหมด 6 ประเภท แสดงในตารางที่ 4.4

ตารางที่ 4.4 operation ใน SSEDIT

สัญลักษณ์	ความหมาย
&eq	เท่ากับ
&ne	ไม่เท่ากับ
&le	น้อยกว่าเท่ากับ
&lt	น้อยกว่า
&ge	มากกว่าเท่ากับ
&gt	มากกว่า

4.4 ขั้นตอน และวิธีการใช้ซอฟต์แวร์ SSEDIT

4.4.1 สร้างไฟล์จาวาพร้อมทั้งคอมไพล์ให้เป็นคลาสไฟล์ ดังตัวอย่างดังรูปที่ 4.8



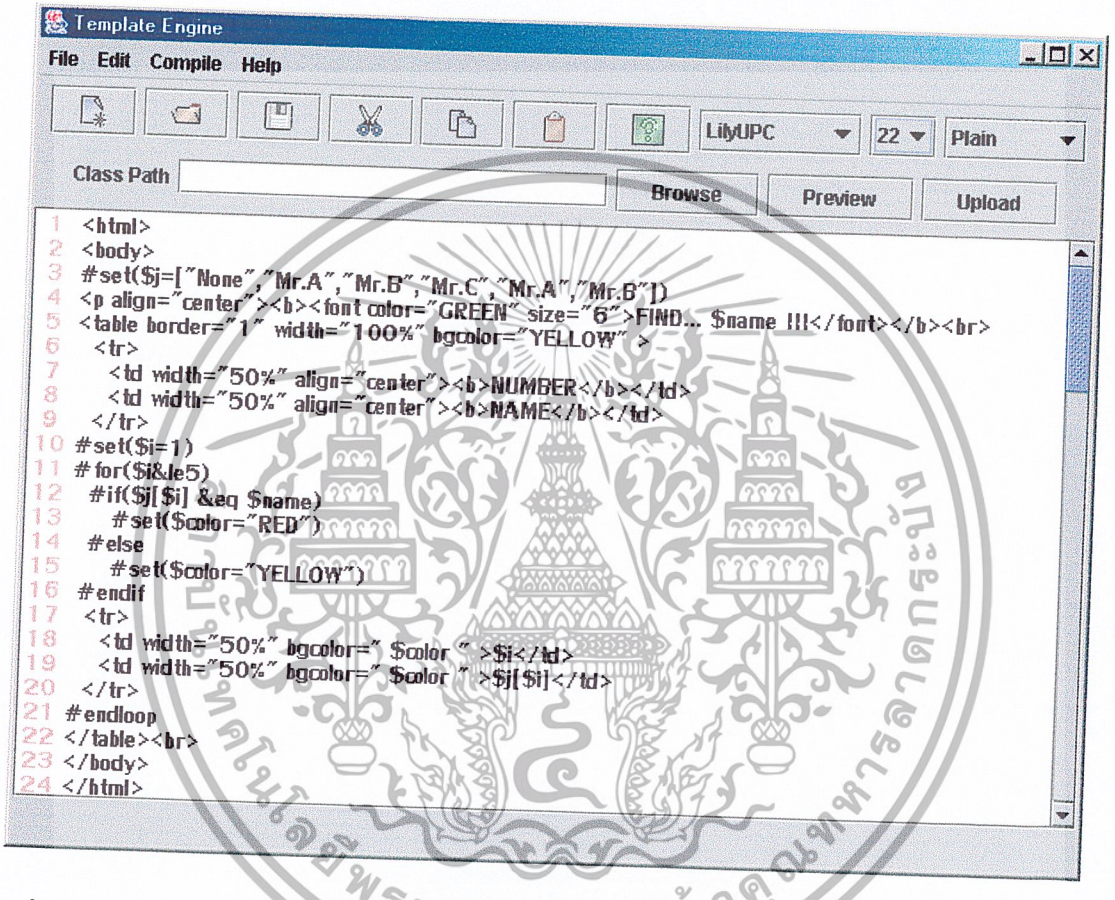
รูปที่ 4.8 ไฟล์ Template.java ซึ่งจะนำมาทดสอบใช้ควบคู่กับ Test.html

จากโค้ด Template.java:

บรรทัด 15 : กำหนด Path ของไฟล์ HTML

บรรทัด 16 : สร้างคอนเท็กซ์ nvPairs เก็บตัวแปรที่ใช้ร่วมกันระหว่างไฟล์จาวา และไฟล์ HTML

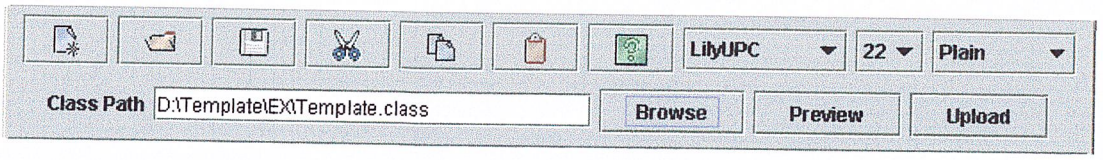
4.4.2 สร้างไฟล์ HTML บนอิดิเตอร์เพื่อใช้ร่วมกับไฟล์จาวาที่สร้างขึ้นแล้ว ดังรูปที่ 4.9



รูปที่ 4.9 ไฟล์ test.html ที่ใช้ Template และ ตัวแปร name

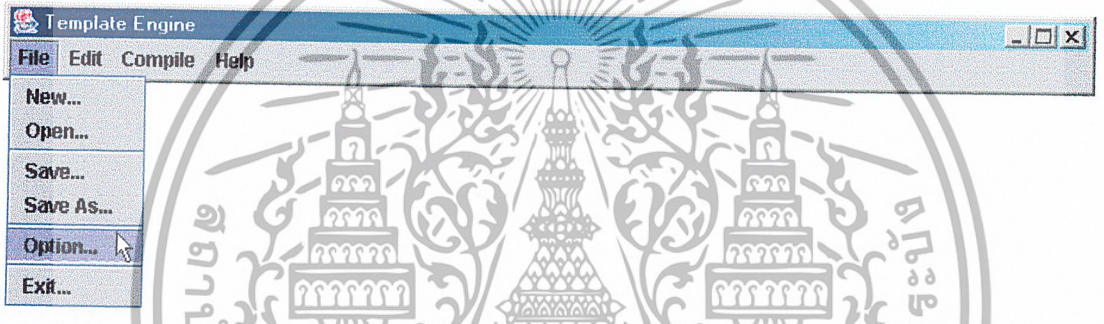
จากโค้ด test.html : โค้ดนี้จะทำการสร้างตาราง โดยค้นหาแถวที่มีชื่อตรงกับตัวแปร name ที่อยู่ในคอนเท็กซ์ โดยจะให้แถวนั้นแสดงเป็นสีแดง

4.4.3 ก่อนที่จะทำการ Preview เพื่อดูผลลัพธ์จะต้องมีการเซตค่าพาธของคลาสไฟล์ที่นำมาประมวลผลร่วมกับไฟล์ HTML ดังรูปที่4.10

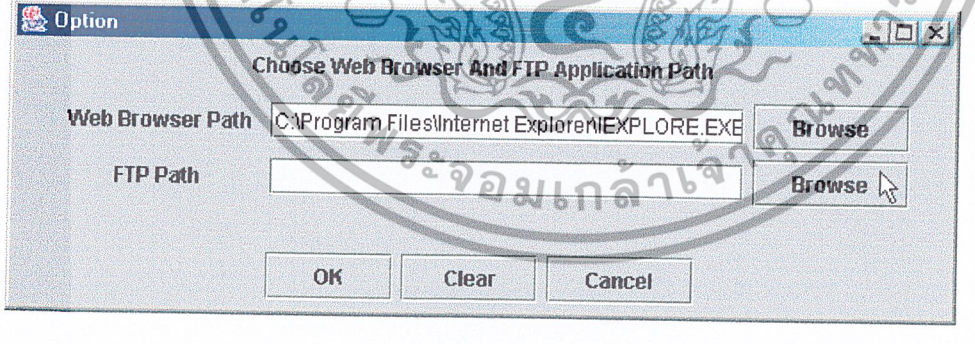


รูปที่ 4.10 การกำหนดค่าคลาสพาธของไฟล์จาวาที่จะนำมาประมวลผลร่วมกัน

4.4.4 เนื่องจากว่าการ Preview และ Upload ไฟล์ไปยัง Application Server จะต้องมีการเรียกใช้โปรแกรมเว็บบราวเซอร์ และ FTP จึงจำเป็นที่ต้องทราบพาธของ Application ทั้งสอง ซึ่งสามารถเซตพาทดังกล่าวที่ เมนู File > Option ดังรูปที่ 4.11



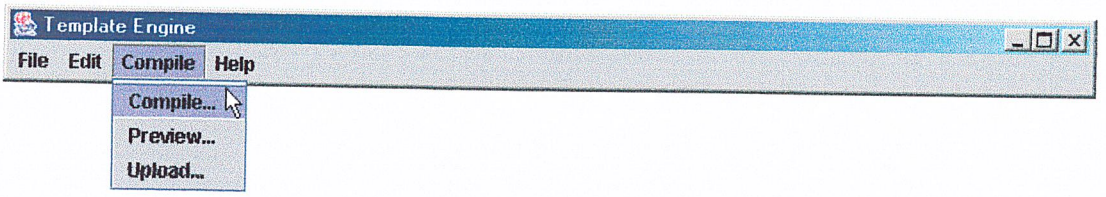
(ก) ตำแหน่งของเมนู Option ที่ใช้กำหนดพาธของ Application



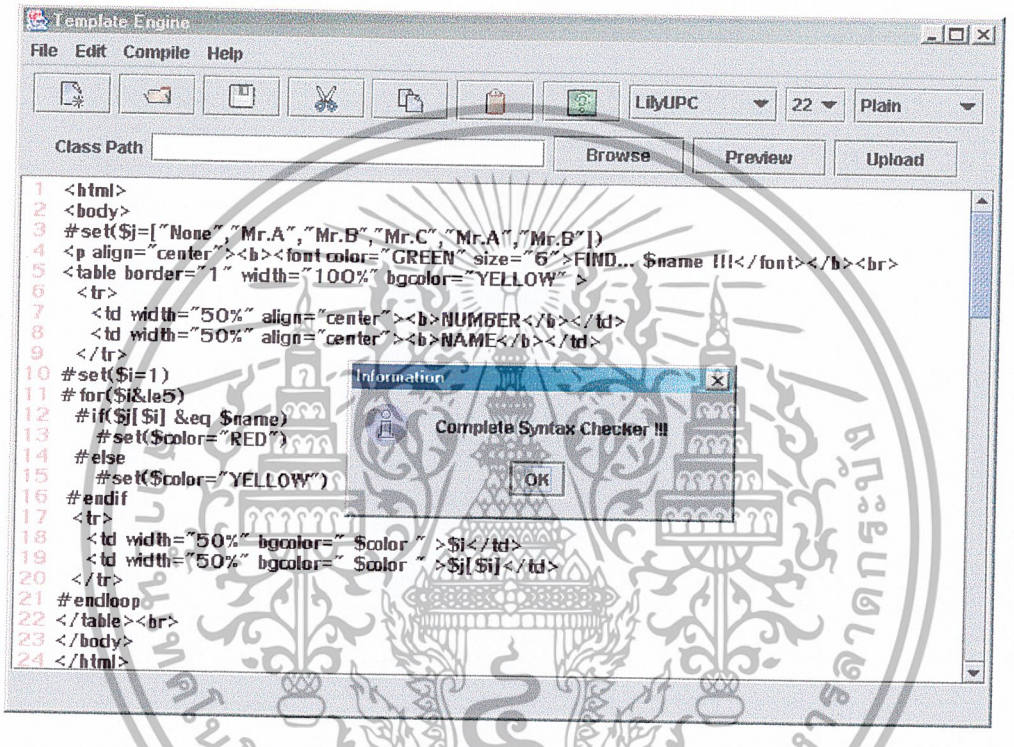
(ข) หน้าต่าง Option หลังจากเลือก เมนู Option

รูปที่ 4.11 การกำหนดพาธของ Web browser และ FTP

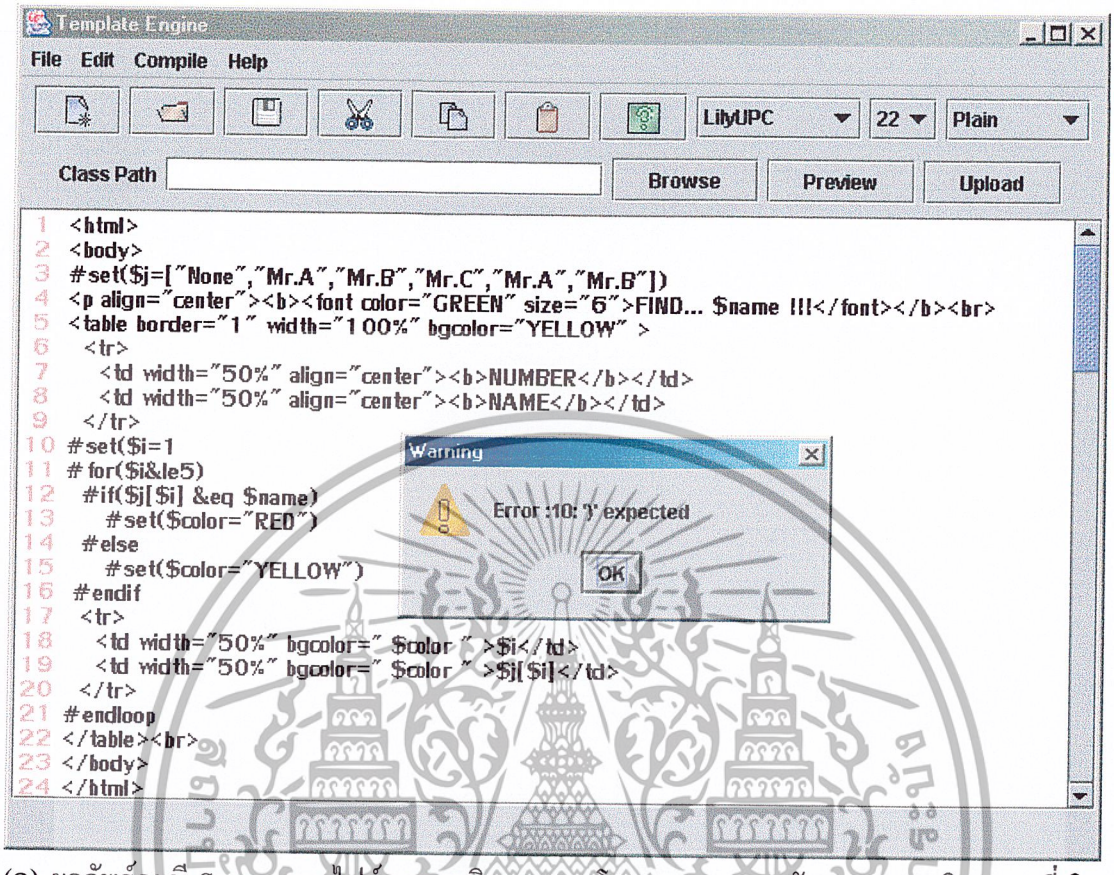
4.4.5 ก่อนทำการ Preview สามารถตรวจสอบ Syntax ของไฟล์ HTML ที่สร้างขึ้นจากเมนู Compile > Compile ดังรูปที่ 4.12



(ก) ตำแหน่งของเมนู Compile ที่ใช้ในการตรวจสอบ Syntax ของไฟล์ HTML



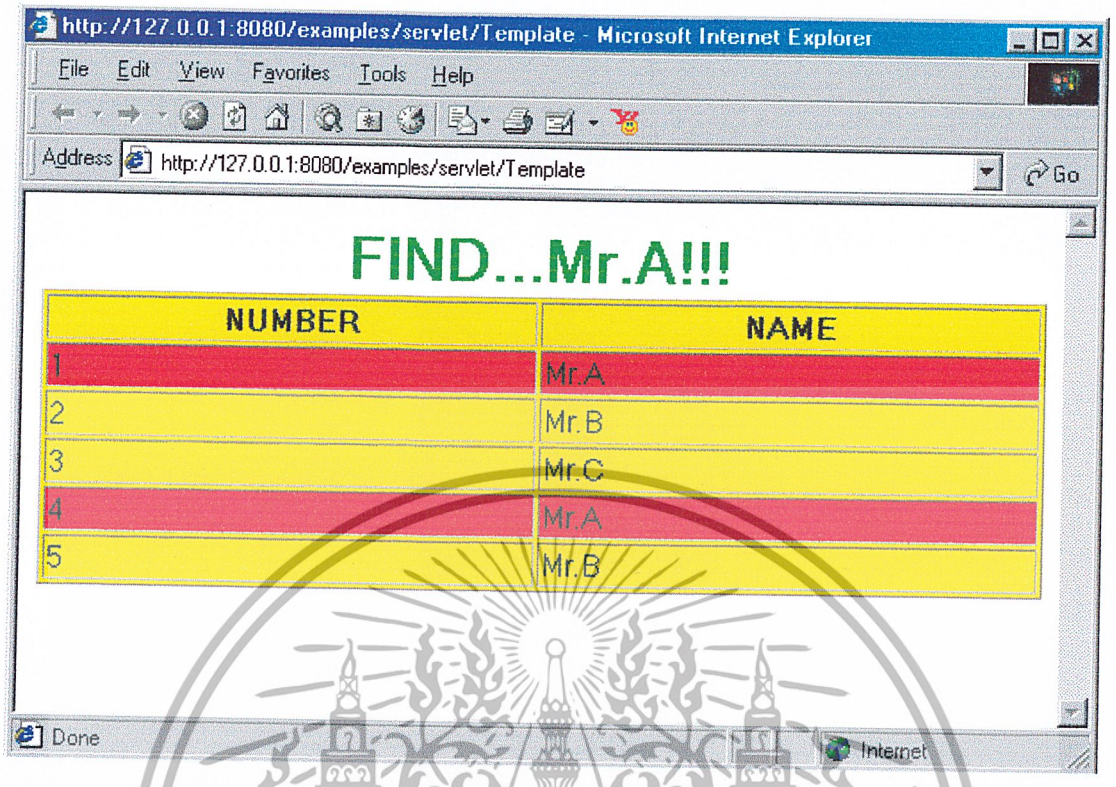
(ข) ผลลัพธ์กรณี Syntax ของไฟล์ HTML ถูกต้อง



(ค) ผลลัพธ์กรณี Syntax ของไฟล์ HTMLผิดพลาด โดยจะแสดงบรรทัด และความผิดพลาดที่เกิดขึ้น

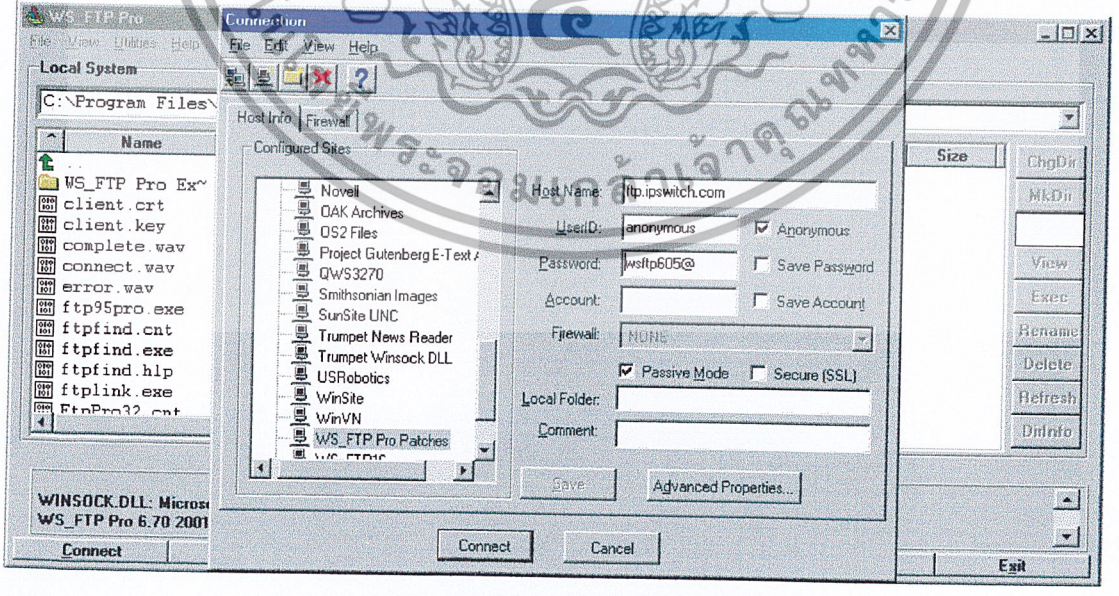
รูปที่ 4.12 การคอมไพล์โค้ด HTML และ วิธีการใช้ Template

4.4.6 เมื่อต้องการ Preview โปรแกรมจะทำการนำคลาสไฟล์ และไฟล์ HTML ไปไว้ที่ไดเรกทอรีของ Apache Tomcat ที่สามารถทำการรันได้ จากนั้นเรียกเว็บบราวเซอร์เพื่อแสดงผล จะได้ผลลัพธ์ดังรูปที่ 4.13



รูปที่ 4.13 ผลลัพธ์ที่เกิดขึ้นจากไฟล์ Template.class และไฟล์ test.html

4.4.7 เมื่อเลือกฟังก์ชัน Upload โปรแกรมจะทำการเรียก FTP เพื่อช่วยในการอัพโหลดไฟล์ไปยัง Application Server ดังรูปที่ 4.14



รูปที่ 4.14 อิดิเตอร์ตอนเรียกโปรแกรม FTP ตามที่ระบุไว้ในเมนู Option

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

5.1.1 การศึกษารวบรวมข้อมูล

การศึกษาเก็บรวบรวมข้อมูล เพื่อใช้เป็นแนวทางในการออกแบบโครงสร้าง Class Diagram และพัฒนาเป็น Application โดยใช้วิธีการศึกษาตัวอย่างการทำงานของ Template Engine ค้นคว้า Component จากอินเตอร์เน็ตและสอบถามจากผู้มีประสบการณ์ในภาษา Java

5.1.2 การวิเคราะห์และออกแบบระบบงาน

ในการวิเคราะห์และออกแบบระบบงานตลอดจนลำดับขั้นตอนในการปฏิบัติงาน เพื่อเป็นต้นแบบในการพัฒนาโปรแกรม โดยวางเค้าโครง Class Diagram, Use case Diagram, Sequence Diagram, User Interface

5.1.3 การพัฒนาโปรแกรมคอมพิวเตอร์

การพัฒนาโปรแกรม SSEDIT ใช้ซอฟต์แวร์ ดังนี้

1) J2SDK 1.4.1 ใช้ในการพัฒนาโปรแกรมจาวา
2) Apache Tomcat 4.0 ใช้ในการทดสอบระบบ
3) Java Servlet 1.2 ใช้ในการทดสอบระบบ servlet
4) Edit Plus ใช้เป็นอิดิเตอร์ในการเขียนโค้ดทั้งหมด
5) Poseidon 1.6 ใช้ในการพัฒนา Object Model

5.1.4 การประยุกต์ใช้ซอฟต์แวร์ SSEDIT

ระบบที่พัฒนาขึ้น จะนำไปใช้ในการพัฒนาโปรแกรม Java Servlet โดยจะทำให้ Java Programmer และ Web Designer สามารถพัฒนาโปรแกรมไปพร้อมๆ กันได้

5.2 ข้อเสนอแนะ

5.2.1 ในการติดตั้งซอฟต์แวร์ Template Engine สำหรับ Java Servlet ควรติดตั้งให้อยู่ในไดเรกทอรีที่ Application Server สามารถรัน servlet ได้

5.2.2 เพื่อความสมบูรณ์ในการติดตั้งซอฟต์แวร์ Template Engine สำหรับ Java Servlet ควรติดตั้ง โปรแกรม FTP และ Web Browser บนเครื่อง Client

5.2.3 ซอฟต์แวร์นี้ทำหน้าที่หลักที่จะช่วยให้ Web Designer สามารถนำโปรแกรมที่ Java Programmer พัฒนาขึ้นมาใช้ร่วมกัน โดยไม่มีการจัดการเกี่ยวกับการคอมไพล์ จาวาไฟล์

บรรณานุกรม

กิตติ ภักดีวัฒนะกุล. 2543. Java ฉบับพื้นฐาน. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร

กิตติ ภักดีวัฒนะกุล. 2544. Java ฉบับโปรแกรมเมอร์. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร

ดนุพล กิ่งสุคนธ์. 2542. การเขียนโปรแกรมด้วยภาษา JAVA. กรุงเทพฯ

รุ่งโรจน์ โพนคำ และ ปราลี มณีรัตน์. 2544. Advanced Java Programming. กรุงเทพฯ

วีระศักดิ์ ซึงถาวร. 2543. Java Programming Volume1. กรุงเทพฯ : ซีเอ็ดยูเคชั่น

Brett Spell. 2000. Professional Java Programming. USA : Wrox Press Inc

Bruce Eckel. 2000. "Thinking in Java- Second Edition". USA : Prentice Hall

Cay HorstMann. 2000. "Computing Concepts with Java 2 Essentials-Second Edition". USA : Wiley&Sons Ltd.

Thomas C. Wu. 1999. "An introduction to Object-Oriented Programming with Java". USA : McGraw-Hill.

Vartan Piroumain. 1999. "Java GUI Development-The Authoritative Solution". India : Technomedia.

David J.Barnes.1999. "Object-Oriented Programming with JAVA : An Introduction". USA : Prentice Hall.

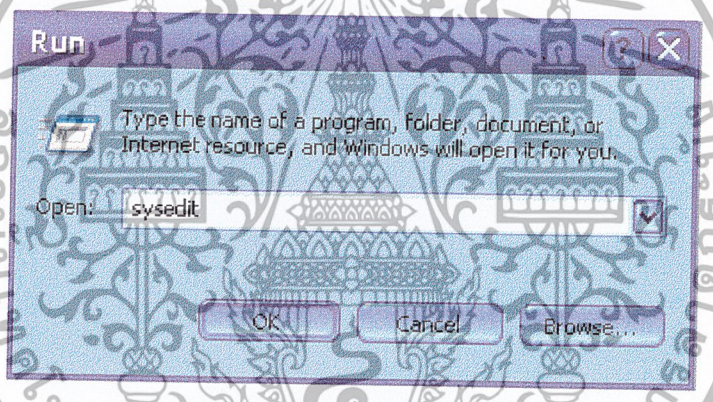
ภาคผนวก ก

การดาวน์โหลดและติดตั้งโปรแกรม Java สำหรับ Windows

ให้ดาวน์โหลดโปรแกรม Java ได้ที่ http://java.sun.com สำหรับเวอร์ชันที่ดาวน์โหลดมาคือj2sdk-1_4_1-rc-windows-i586.exe แล้วทำการติดตั้งโปรแกรมไว้ที่ไดเรกทอรี C:\j2sdk1.4.1

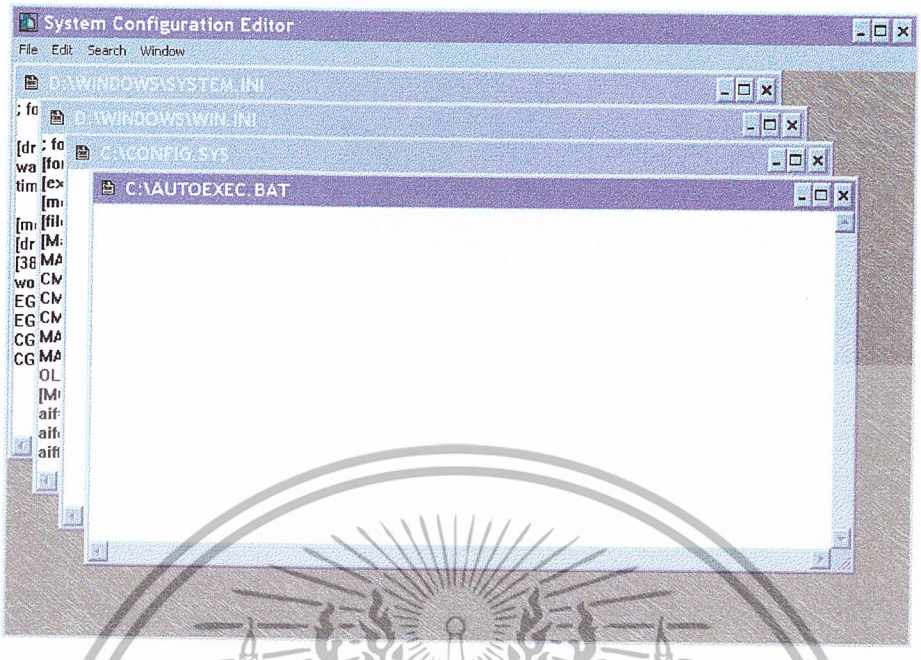
เมื่อได้ทำการติดตั้ง j2sdk เรียบร้อยแล้ว ขั้นต่อไปก็คือ การปรับแต่งสภาพแวดล้อมที่เป็นประโยชน์บางประการให้เหมาะสมกับโปรแกรม Java ก่อน ทั้งนี้เพื่อความสะดวกในการทำงาน ความปลอดภัยของโปรแกรม Java และโปรแกรมอื่นบนระบบ ซึ่งมีขั้นตอนการปรับแต่งสภาพแวดล้อมดังนี้

1. คลิกปุ่ม Start ที่มุมซ้ายด้านล่างของจอ
2. เลือก Run บนเมนูที่ปรากฏขึ้นมา
3. พิมพ์คำว่า sysedit ลงในบ๊อกซ์ของ Open ดังรูปที่ 1



รูปที่ ก-1 รันโปรแกรม sysedit.exe จากเมนู Start > Run

4. คลิกปุ่ม OK ภายในบ๊อกซ์จะได้รายชื่อไฟล์บนระบบแสดงขึ้นมาดังรูปที่ 2



รูปที่ ก-2 ข้อมูลต่างๆ ใน Sysedit

5. เลือกไฟล์ AUTOEXEC.BAT สำหรับเครื่องที่ภายในไฟล์ AUTOEXEC.BAT ยังไม่มีการเซตค่าใด ๆ ให้ทำ

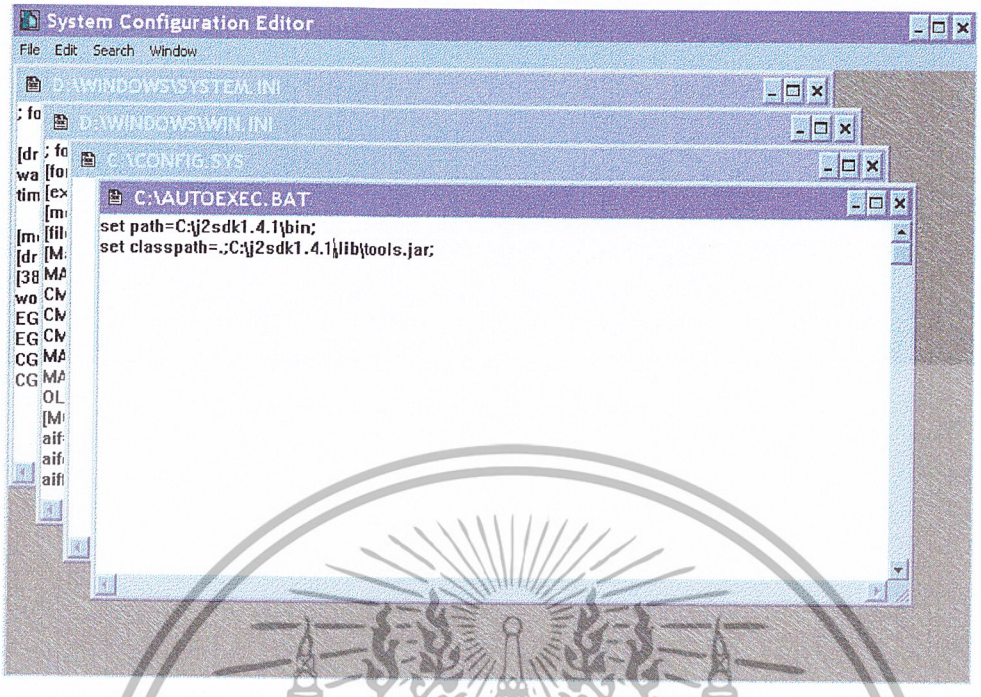
ทำการเซตค่า path และ classpath ดังนี้

- set path=C:\j2sdk1.4.1\bin;
- set classpath=.;C:\j2sdk1.4.1\lib\tools.jar;

สำหรับเครื่องที่ภายในไฟล์ autoexec.bat มีการเซตค่า path และ classpath อยู่แล้วให้ใส่ ; ต่อท้ายตัวอักษรสุดท้ายในคำสั่งดังนี้

- set path=C:\windows; C:\j2sdk1.4.1\bin;
- set classpath=C:\windows; .;C:\j2sdk1.4.1\lib\tools.jar;

ซึ่งสามารถเซตค่า path และ classpath ได้ดังรูปที่ 3



รูปที่ ก-3 การเซต path และ classpath ให้กับไฟล์ AUTOEXEC.BAT

6. เสร็จแล้วให้ทำการ save ไฟล์ AUTOEXEC.BAT ที่ทำการแก้ไขแล้วทำการ restart เครื่องก็เป็นการเสร็จสิ้นการติดตั้งโปรแกรม Java

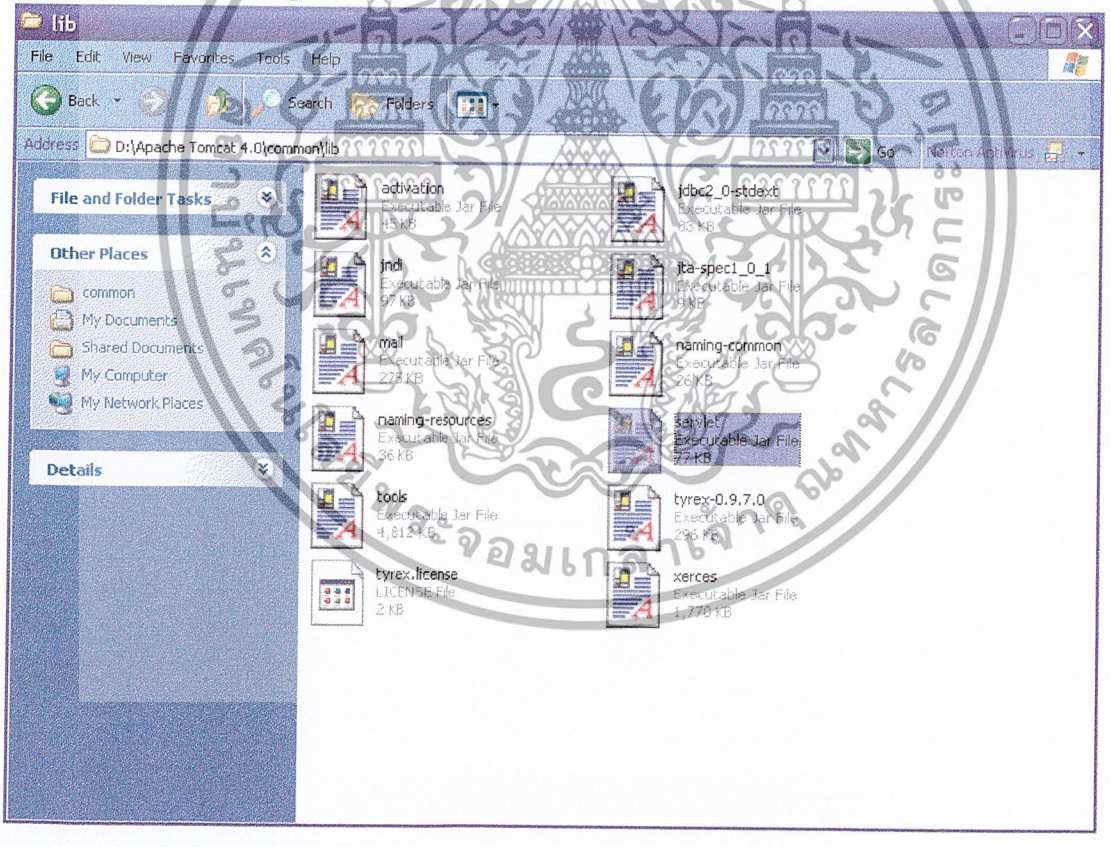
การดาวน์โหลดและติดตั้ง Apache Tomcat

ดาวน์โหลดโปรแกรม Apache Tomcat ได้ที่ http://jakarta.apache.org สำหรับเวอร์ชันที่ดาวน์โหลดมาคือ jakarta-tomcat-4.0.3.exe จากนั้นให้ทำการติดตั้งไว้ที่ไดเรกทอรี C:\Apache Tomcat 4.0

หมายเหตุ : ภายในเครื่องจะต้องมีการติดตั้งโปรแกรม Java ก่อนอยู่แล้วมิฉะนั้นจะทำการติดตั้งโปรแกรม Apache Tomcat ไม่ได้

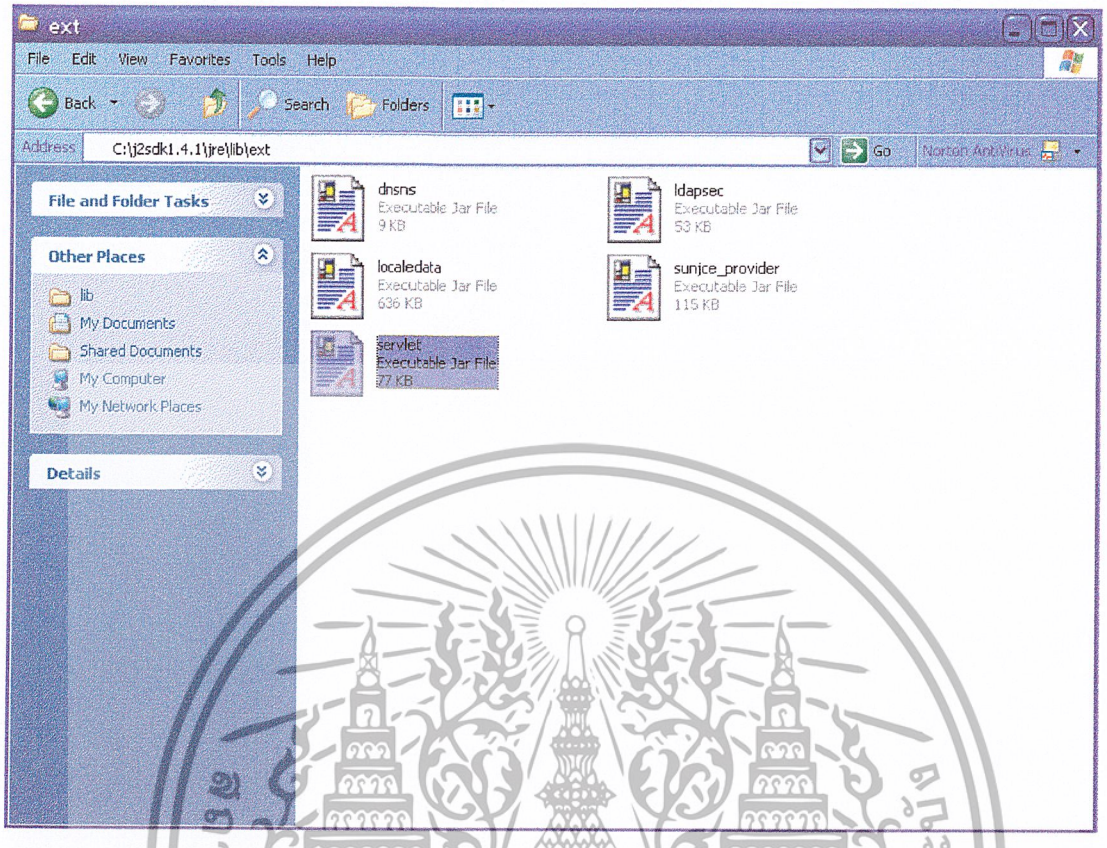
เมื่อติดตั้งโปรแกรม Apache Tomcat เสร็จแล้วจะต้องเตรียมความพร้อมในการใช้งาน Servlet โดยมีขั้นตอนในการเตรียมความพร้อมดังนี้

1. ให้เข้าไปยังไดเรกทอรี C:\Apache Tomcat 4.0\common\lib แล้วทำการคัดลอกไฟล์ Servlet.jar ดังรูปที่ 4



รูปที่ ก-4 ไดเรกทอรีที่เก็บไฟล์ servlet.jar

2. นำไฟล์ servlet.jar ที่คัดลอกมาไปไว้ในไดเรกทอรี C:\j2sdk1.4.1\jre\lib\ext ดังรูปที่ 5



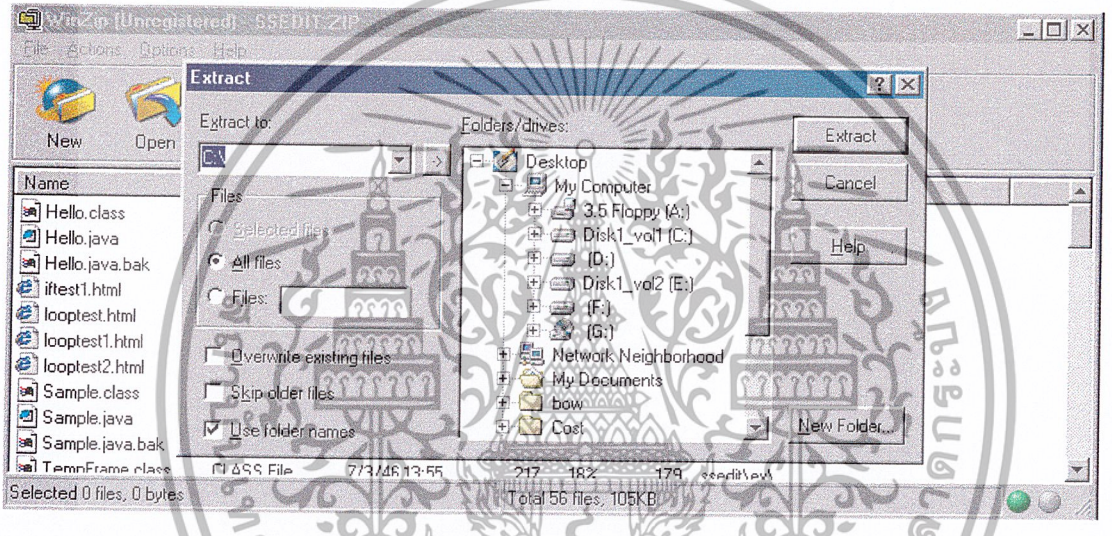
รูปที่ ก-5 ไดเรกทอรีที่นำไฟล์ servlet.jar ไปไว้

3. เสร็จสิ้นการเตรียมความพร้อมให้ java servlet สามารถประมวลผลร่วมกับ Apache Tomcat Server ได้นั้น

การติดตั้งโปรแกรมอิดิเตอร์และอินเตอร์พรีเตอร์สำหรับการแปลภาษารวมเว็บเพจและเซิร์ฟเล็ต

มีขั้นตอนในการติดตั้งดังนี้

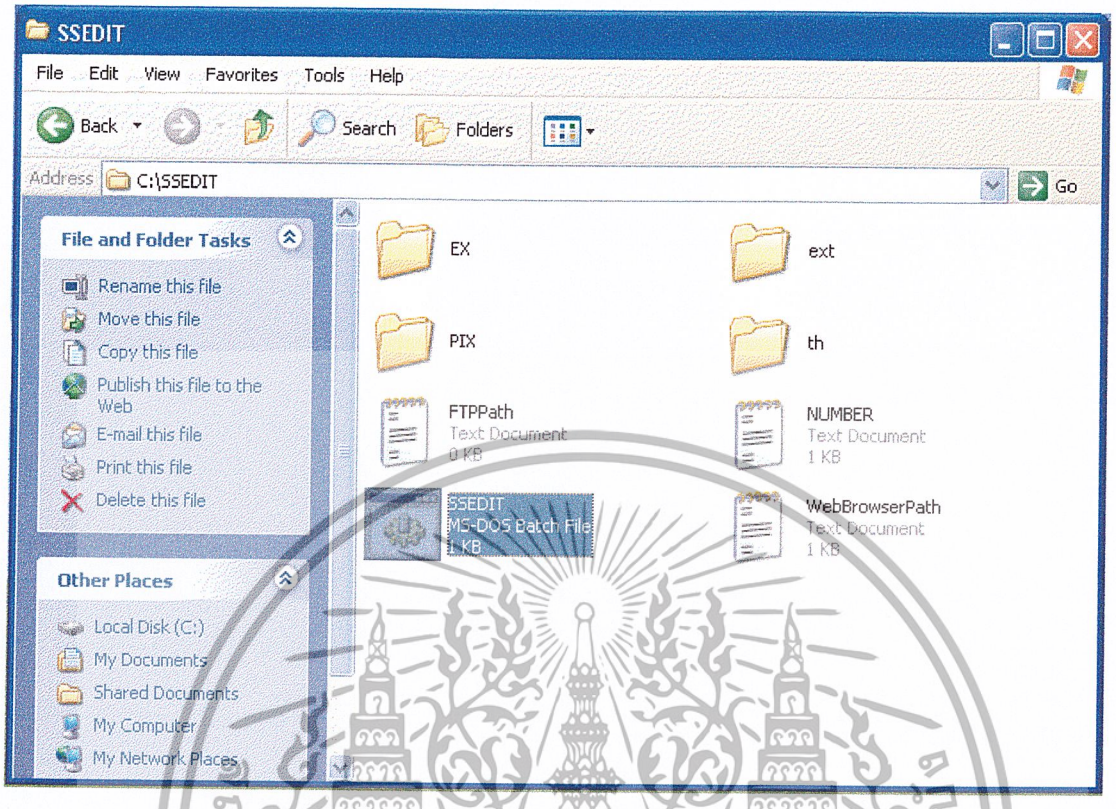
1. ใส่แผ่น CD ที่แนบมาจากเอกสารนี้ จากนั้นให้เข้าไปยัง Drive CD-ROM จะเห็นโฟลเดอร์ Special Problem ให้เข้าไปหาไฟล์ SSEDIT.zip ที่อยู่ในไดเรกทอรี Special Problem\Install\SSEDIT แล้วทำการแตกไฟล์ไปไว้ที่ไดเรกทอรี C:\ โดยใช้โปรแกรม Winzip ดังรูปที่ 6



รูปที่ ก-6 การแตกไฟล์ SSEDIT.zip โดยใช้โปรแกรม Winzip

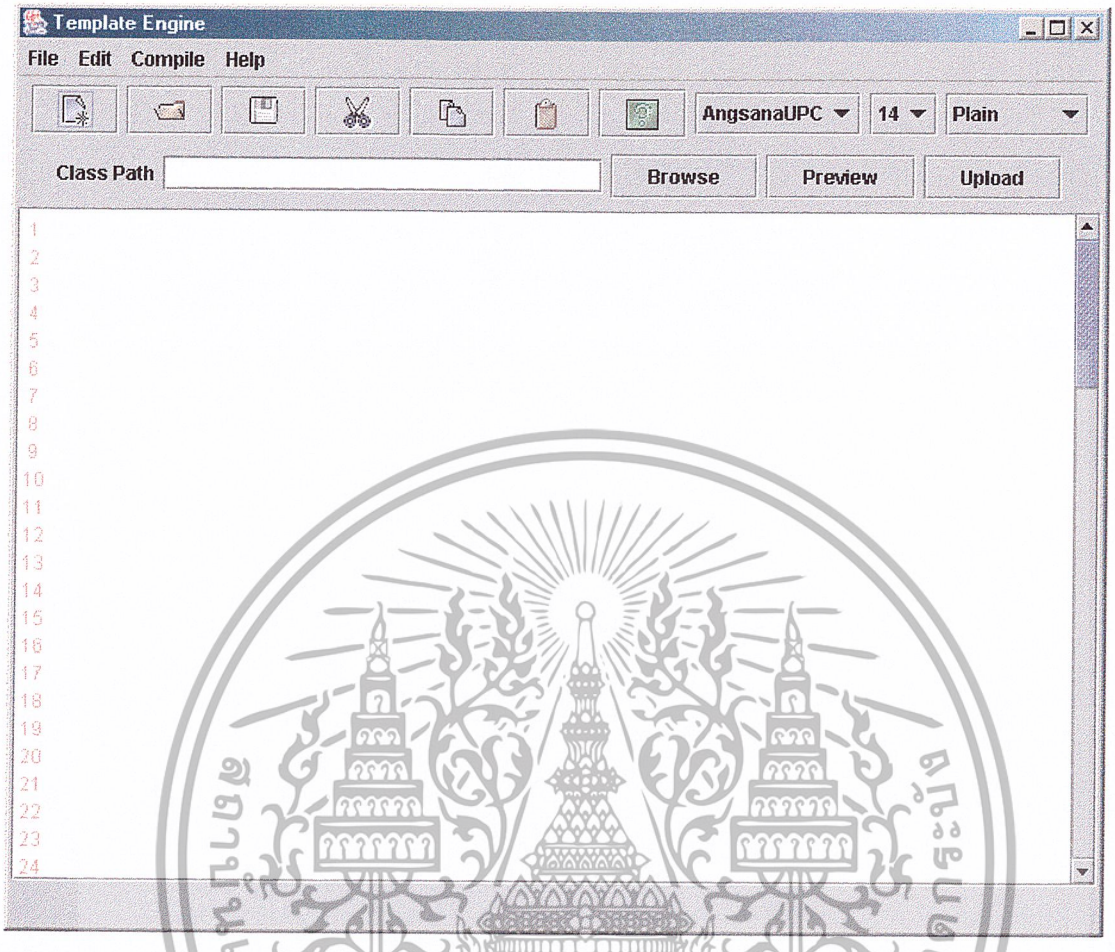
2. ให้เข้าไปคัดลอกไฟล์ SSEDIT.jar ที่เก็บอยู่ในไดเรกทอรี C:\SSEDIT\ext ไปไว้ที่ไดเรกทอรี C:\j2sdk1.4.1\jre\lib\ext

3. ให้เข้าไปยังไดเรกทอรี C:\SSEDIT แล้วให้ดับเบิ้ลคลิ๊กที่ไฟล์ SSEDIT.MS-DOS Batch ดังแสดงดังรูปที่ 7



รูปที่ ก-7 การดับเบิ้ลคลิ๊กที่ไฟล์ SSEDIT.MS-DOS Batch

4. จากนั้นจะปรากฏอิดิเตอร์ SSEDIT ขึ้นมาดังรูปที่ 8



รูปที่ ก-8 การเริ่มต้นการใช้งานโปรแกรม SSEDIT

**หมายเหตุ การติดตั้งโปรแกรมจะต้องทำการติดตั้งตามลำดับที่จัดเรียงไว้ในหนังสือเล่มนี้มิฉะนั้นจะทำให้เกิดการผิดพลาดในการทำงาน