

การปรับปรุงตัวแปลงภาษา HTML เป็น WML
Improvement of HTML-to-WML Converter



นายเกษม ตริตระการ

นางสาวจุฑารัตน์ เจริญไพฑูลย์กิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เลขหมู่.....
เลขทะเบียน...46151
วัน, เดือน, ปี 20 ส.ค. 2546

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

การปรับปรุงตัวแปลงภาษา HTML เป็น WML
Improvement of HTML-to-WML Converter



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การปรับปรุงตัวแปลงภาษา HTML เป็น WML

Improvement of HTML-to-WML Converter

ผู้จัดทำ

1. นายเกษม ตรีตระการ

รหัสประจำตัว 41014043

2. นางสาวจุฑารัตน์ เจริญไพฑูริย์กิจ

รหัสประจำตัว 41014080



(ดร.วิศิษฐ์ หิรัญกิตติ)

อาจารย์ที่ปรึกษา

การปรับปรุงตัวแปลงภาษา HTML เป็น WML

นาย เกษม ตรีตระกูล 41014043
 นางสาว จุฬารัตน์ เจริญไพฑูย์กิจ 41014080
 คร. วิศิษฐ์ หิรัญกิตติ อาจารย์ที่ปรึกษา
 ปีการศึกษา 2544

บทคัดย่อ

จุดประสงค์ในการปรับปรุงและพัฒนาตัวแปลงภาษา HTML เป็น WML คือเพื่อให้อุปกรณ์ที่สามารถใช้งานเว็บสามารถเข้าถึงข้อมูลและบริการต่างๆ ที่เป็นบริการของเว็บ และปรับปรุงตัวแปลงภาษา HTML เป็น WML ในโครงงานเดิม [8] ให้สามารถแสดงผลเท็กภาษา HTML ได้มากขึ้น

ปริญญานิพนธ์ฉบับนี้เป็นการปรับปรุงตัวแปลงภาษาจาก HTML เป็น WML เดิมโดยใช้ความรู้เกี่ยวกับ HTTP โพรโตคอล, WAP โพรโตคอล, ลักษณะโครงสร้างภาษา HTML, WML, XHTML, XML และใช้ภาษาจาวาในการเขียนโปรแกรม ในการทำงานของตัวแปลงภาษาจะใช้จาวาเซิร์ฟเลตร้องขอและรับเมสเซจ HTTP จากเว็บเซิร์ฟเวอร์ ซึ่งจะได้รับข้อมูลมาในรูปของไฟล์ภาษา จากนั้น จะใช้ไฟล์ HTML นี้แปลงให้อยู่ในรูปของไฟล์ภาษา XML และไฟล์ภาษา XHTML ตามลำดับ ก่อนท้ายที่สุดจะแปลงเป็น WML เพื่อส่งไปแสดงผลยังเว็บเบราว์เซอร์ที่อยู่บนอุปกรณ์เว็บ ผลพลอยได้อื่นๆที่ได้รับคือ สามารถได้ผลเป็นไฟล์ภาษา XHTML ด้วย ความสามารถพิเศษของตัวแปลงภาษานี้คือสามารถแปลงตารางขนาดใหญ่ที่อยู่ในรูปแบบที่ซับซ้อนได้ และสามารถแบ่งได้คได้ เพื่อรองรับขนาดของหน้าจอโทรศัพท์มือถือ ในขณะที่ตัวแปลงภาษาเดิมทำไม่ได้

Improvement of HTML-to-WML Converter

Kasame Tritrakan	41014043
Jutarat Jaroenpiboonkit	41014080
Dr. Visit Hirunkitti	Advisor

ABSTRACT

The main purpose of this project is to improve the HTML-to-WML Converter developed in a previous project [8], so that the new converter can handle more HTML tags and work more efficiency.

The development of this converter is base on knowledge of HTTP protocol, WAP protocol, language structure of HTML, WML, XHTML and XML, and it is implemented using Java. The process of the conversion is follows. First, our Java Servlet issues on HTTP request to server and obtains the HTTP response form the server; the data it obtains is in HTML, the HTML data is stored in a file. Then the HTML file is converted into XML and XHTML formats respectively before the XHTML being converted to WML format and return the result pages to a WAP browser. As a result we also convert HTML data into XHTML data

The special feature of the new converter is that it can handle large and complex tables of HTML and can divide large decks, so that elements in the tables and large decks can fit in small screens of a WAP browser. This ability is not found in the previous converter.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์วิศิษฐ์ หิรัญกิตติ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก และ IBM ที่ให้ความอนุเคราะห์ Software IBM WebSphere Transcoding Publisher ในโครงการ IBM Academic Program ที่ร่วมกับภาควิชาวิศวกรรมคอมพิวเตอร์

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

และบุคคลอื่นๆนอกเหนือจากบิดา มารดา และ อาจารย์ที่ปรึกษา คือเหล่าเพื่อนๆทุกคนที่ได้ให้กำลังใจและให้ยืมเครื่องคอมพิวเตอร์ลงโปรแกรมยามโปรแกรมหมดอายุแล้วซื้อเก็ยฟอร์เมทเครื่อง และช่วยพาไปกินอาหารดีๆยามดึก และให้กำลังใจกันตลอดมา รวมถึงเว็บไซต์ google.com ที่ช่วยให้ข้าพเจ้าสามารถหาข้อมูลมากมายเพื่อใช้ในการทำโปรเจกต์

นายเกษม ตรีตระการ

นางสาวจุฑารัตน์ เจริญไพบูลย์กิจ

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญรูปภาพ	VIII
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ขอบเขตของงานวิจัย	2
1.4 วิธีการดำเนินงาน	2
บทที่ 2 ทฤษฎี	4
2.1 โพรโทคอล WAP (Wireless Application Protocol)	4
2.1.1 WAP Model	4
2.1.2 สถาปัตยกรรมของแวย์พ (WAP Architecture)	5
2.1.3 ตัวอย่างของเครือข่ายแวย์พ	9
2.2 โพรโทคอล HTTP (HyperText Transfer Protocol)	10
2.2.1 โครงสร้าง HTTP (HTTP Structure)	11
2.2.2 รูปแบบของการร้องขอ	19
2.3 ภาษามาร์กอัพ (Markup Language)	22
2.3.1 HTML (HyperText Markup Language)	22
2.3.2 WML (Wireless Markup Language)	23
2.3.3 XML	24
2.3.4 XHTML	28
2.4 เซิร์ฟเลต (Servlet)	32
2.4.1 เซิร์ฟเลตเอนจิน (Servlet engine)	33
2.4.2 อินเตอร์เฟส javax.servlet.Servlet	34
2.4.3 วงจรชีวิตของเซิร์ฟเลต (Servlet's Life Cycle)	35

สารบัญ (ต่อ)

	หน้าที่
บทที่ 3 หลักการในการออกแบบ	37
3.1 การออกแบบและการทำงานของโปรแกรมแปลงภาษา1	37
3.2 การออกแบบและการทำงานของโปรแกรมแปลงภาษา2	37
3.2.1 บทนำ	37
3.2.2 โครงสร้างของโปรแกรมแปลงภาษา2	38
3.2.3 องค์ประกอบของโปรแกรม	40
3.2.4 การทำงานของโปรแกรม	45
บทที่ 4 การทำงานของโปรแกรมแปลงภาษา2	50
4.1 การทำงานทั่วไป	50
4.2 การทำงานที่มีแท็กลิงก์หรือเฟรม	51
4.3 การทำงานที่มีแท็กฟอร์ม	51
4.4 การทำงานเมื่อมีการ Submit ฟอร์ม	53
บทที่ 5 ผลการทดลองและการเปรียบเทียบ	54
5.1 การใช้งาน	54
5.2 ตัวอย่างการแปลง	54
5.2.1 ตัวอย่างการแปลง frame	54
5.2.2 ตัวอย่างการแปลง form	56
5.2.3 ตัวอย่างการแปลง table	57
5.2.4 ตัวอย่างการแบ่งเค็ค	57
5.3 การเปรียบเทียบ	58
5.3.1 IBM WebSphere Transcoding Publisher	58
5.3.2 โปรแกรมแปลงภาษา1	59
บทที่ 6 สรุปและบทวิเคราะห์	61
6.1 สรุป	61
6.2 ข้อดี	61
6.3 ข้อเสีย	63
6.4 ข้อจำกัด	63
6.5 แนวทางการพัฒนาต่อไป	63
ภาคผนวก ก.	65
ภาคผนวก ข	115

สารบัญ (ต่อ)

ภาคผนวก ก

บรรณานุกรม

หน้าที่

129

130



สารบัญตาราง

	หน้าที่
ตารางที่ 3.1 การเปลี่ยนจาก HTML เป็น XHTML	46
ตารางที่ 3.2 การเปลี่ยนจาก XHTML เป็น WML	47
ตารางที่ 5.1 การเปรียบเทียบกับ IBM WebSphere Transcoding Publisher	59
ตารางที่ 5.2 การเปรียบเทียบกับ โปรแกรมแปลงภาษา1	60



สารบัญรูปภาพ

	หน้าที่
รูปที่ 2.1 WAP Model	5
รูปที่ 2.2 WAP Architecture	6
รูปที่ 2.3 ส่วนประกอบของ WAE	7
รูปที่ 2.4 ตัวอย่างเครือข่ายของเว็บ	9
รูปที่ 2.5 การติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์	11
รูปที่ 2.6 ลักษณะข้อมูลที่รับส่งระหว่างไคลเอนต์และเซิร์ฟเวอร์	18
รูปที่ 2.7 หน้าที่และการทำงานของเซิร์ฟเลต	33
รูปที่ 2.8 เซิร์ฟเลตเอนจิน และเซิร์ฟเลตที่อยู่ภายใน	34
รูปที่ 2.9 วงจรชีวิตของเซิร์ฟเลต	35
รูปที่ 2.10 วิธีการจัดการกับร้องขอแบบ GET และ POST ของเซิร์ฟเลต	36
รูปที่ 3.1 ขั้นตอนการแปลงของโปรแกรมรูนก่อน	37
รูปที่ 3.2 ลำดับการแปลงจาก HTML เป็น WML	38
รูปที่ 3.3 โครงสร้างโดยรวมของโปรแกรมแปลงภาษา2	38
รูปที่ 3.4 โครงสร้างภายในของโปรแกรมแปลงภาษา2	39
รูปที่ 3.5 เมธอดใน Convert	39
รูปที่ 3.6 เมธอดในการแปลงเป็น WML	40
รูปที่ 3.7 การทำงานของส่วนที่ 1	46
รูปที่ 4.1 การทำงานโดยทั่วไป	50
รูปที่ 4.2 การทำงานในการแปลงแท็กฟอร์ม	52
รูปที่ 5.1 เว็บเพจของ [HT->W]ML Converter	54
รูปที่ 5.2 การแสดงผลเว็บเพจ ภาษา HTML โดยใช้ แท็ก frame	54
รูปที่ 5.3 การแสดงผลเว็บเพจ ภาษา WML โดยแปลงแต่ละเฟรมให้เป็นลิงก์	55
รูปที่ 5.4 การแสดงผลเว็บเพจ ภาษา WML	55
รูปที่ 5.5 การแสดงผลเว็บเพจ ภาษา HTML โดยใช้ แท็ก form	56
รูปที่ 5.6 การแสดงผลเว็บเพจ ภาษา WML	56
รูปที่ 5.7 ตัวอย่างการแปลงตาราง	57
รูปที่ 5.8 ตัวอย่างเว็บเพจเพื่อใช้ในการแบ่งเค็ล	57
รูปที่ 5.9 ตัวอย่างของเด็กที่ 1	58

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

ปัจจุบันแนวโน้มของเทคโนโลยีทางด้านคอมพิวเตอร์และอินเทอร์เน็ต จะเป็นไปในรูปแบบของเทคโนโลยีแบบไร้สาย(Wireless Technology) ผลที่ได้ตามมาก็คือผลิตภัณฑ์ใหม่ๆ เช่น เครื่องคอมพิวเตอร์มือถืออย่าง Palm รวมทั้ง PocketPC เป็นต้นมีปริมาณเพิ่มมากขึ้น นอกจากนี้ในส่วนของโทรศัพท์มือถือก็มีเทคโนโลยี WAP (Wireless Application Protocol) ซึ่งเกิดขึ้นเพื่อเพิ่มประสิทธิภาพของโทรศัพท์มือถือให้มากกว่าการเป็นแค่อุปกรณ์สื่อสารด้วยเสียง และบริการส่งข้อความสั้นๆ (Short Message Service) โดยเพิ่มความสามารถในการสื่อสารข้อมูล (data) เข้ามาด้วย ซึ่งข้อมูลที่ว่านี้คือข้อมูลในเครือข่ายอินเทอร์เน็ตนั่นเอง

แต่ในปัจจุบันการใช้งานเทคโนโลยี WAP ยังไม่เป็นที่แพร่หลายเท่าที่ควรอันเนื่องมาจากข้อจำกัดหลายอย่าง เช่น การทำงานของเทคโนโลยี WAP จะต้องอาศัยเครือข่ายไร้สายแบบดิจิทัลเท่านั้น, อัตราเร็วในการส่งข้อมูลซึ่งจำกัดอยู่ที่ 9.6 Kbps, ไม่กคบน โทรศัพท์มือถือใช้งานยากในการกดตัวอักษร ไม่เหมือนกับเป็นคีย์บอร์ดของคอมพิวเตอร์ และการคิดค่าบริการจะคิดตามเวลาการใช้งาน (air time) ซึ่งถือว่าสิ้นเปลืองมากพอสมควร เป็นต้น นอกจากนี้เว็บไซต์(WAP Site) ยังมีจำนวนและคุณภาพสู้เว็บไซต์(Web Site) ไม่ได้ เนื่องจากจำนวนคนที่สร้างเว็บเพจ(WAP Pages) มีจำนวนที่น้อยกว่าเว็บเพจ(Web Pages) เพราะการสร้างเว็บเพจจะต้องใช้ภาษา WML(Wireless Markup Language) ซึ่งเป็นภาษาที่ใหม่กว่าHTML ทำให้หลายๆคนไม่สนใจที่จะสร้างเว็บเพจขึ้นมาใหม่

ในปัจจุบันก็ได้มีเครื่องมือในการแปลงจากภาษา HTML ไปเป็นภาษา WML อยู่ด้วยกันหลายตัว โดยแต่ละตัวจะมีความแตกต่างกันออกไปทั้งในเรื่องของฟังก์ชันการใช้งาน, ความสามารถในการแปลง, ความครอบคลุมเนื้อหาต้นฉบับของเว็บ และการแสดงผลออกหน้าจอบนโทรศัพท์มือถือให้สามารถอ่านได้ง่าย ซึ่งตัวแปลงดังกล่าวยังไม่สามารถแปลงบางหน้าเว็บไซต์ที่มีความซับซ้อนมากๆได้อีกทั้งแต่ละโปรแกรมผู้ใช้ก็ไม่สามารถที่จะเพิ่มฟังก์ชันในการแปลงได้

จากเหตุผลข้างต้นจึงมีความคิดที่จะสร้างตัวแปลงภาษา HTML เป็น WML ที่มีความสามารถในการแปลงมากขึ้น สามารถแปลงหน้าเว็บไซต์ที่มีความซับซ้อนไม่เพิ่มมากขึ้นจากเดิมที่มีอยู่ อีกทั้งยังสามารถให้ผู้สนใจที่พัฒนาความสามารถของการแปลงสามารถนำโปรแกรมไปพัฒนาต่อได้ง่ายขึ้น สามารถเพิ่มฟังก์ชันการทำงานได้ง่าย และเป็นการปรับปรุงจากงานวิจัยตัวแปลงภาษา HTML เป็น WML ของนางสาววรรณฉัตร และ นางสาววรรณิศา ให้มีประสิทธิภาพมากขึ้น สามารถนำไปใช้งานในสภาพแวดล้อมจริงมากขึ้น อีกทั้งแนวโน้มของภาษา WML จะเข้าใกล้ภาษา XHTML และ XML เข้าไปเรื่อยๆ จึงได้มีการสร้างตัวแปลงภาษาที่สามารถรองรับการเปลี่ยนแปลงนี้ได้ โดยได้เพิ่มประสิทธิภาพในการแปลงเป็นภาษา XHTML และ XML เข้าไปด้วยโดยผู้ใช้งานสามารถใช้กับอุปกรณ์ที่ใช้ภาษา WML, XHTML และ XML ได้โดยไม่ต้องมีการกำหนดค่าใดๆเพิ่ม

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษาเทคโนโลยีเว็บ และเว็บเพื่อหาความแตกต่างและหาวิธีในการนำเสนอหน้าเว็บเพจ ด้วยเว็บเพจให้มีความสวยงาม และง่ายต่อการนำไปใช้งานจริง
2. นำแสดงบนโทรศัพท์มือถือได้ โดยใช้ภาษาที่ง่ายต่อการ โปรแกรม, สามารถเข้ากับสภาพแวดล้อมในการใช้งาน และสามารถใช้งานได้มีประสิทธิภาพ
3. พัฒนาเครื่องมือในการแปลง (Conversion tool) เพื่อเพิ่มประสิทธิภาพในการแปลงโค้ดภาษา HTML ไปเป็นภาษา WML เพื่อรองรับรูปแบบในการเขียนภาษา HTML ได้หลากหลาย
4. สามารถแสดงผลเว็บเพจที่ผ่านการแปลงแล้วได้มีประสิทธิภาพมากขึ้น อีกทั้งเพิ่มเติมฟังก์ชันการทำงานที่ยังไม่มีในตัวแปลงภาษา1 เพื่อเพิ่มบริการต่างๆให้ผู้ใช้ได้ใช้อย่างมีประสิทธิภาพมากขึ้น
5. เพิ่มความสามารถในการแปลงเป็นภาษา XHTML และ XML เข้าไปด้วยเพื่อรองรับเทคโนโลยีเว็บเวอร์ชันใหม่ๆที่กำลังจะมีการนำมาใช้ในไม่ช้า
6. เพิ่มความสามารถในการแปลงตาราง ในรูปแบบที่ซับซ้อน เช่น ตารางซ้อนตารางได้ แบ่งหน้า เว็บเพจ ออกเป็นเค็ลหลายๆเค็ลได้

1.3 ขอบเขตของงานวิจัย

- พัฒนาตัวแปลงโดยพยายามให้ครอบคลุมทุกแท็กของภาษา HTML
- จัดรูปแบบในการแสดงผลให้มีความสวยงามมากยิ่งขึ้น
- เพิ่มความสามารถในการแปลงเป็นภาษา XML และ XHTML เพื่อรองรับเว็บ เวอร์ชัน2 ที่จะมีการนำมาใช้ในอนาคต
- เพิ่มประสิทธิภาพเพื่อสามารถนำไปใช้จริงได้มากขึ้น
- พัฒนาโปรแกรมเพื่อความสะดวกในการพัฒนาต่อ
- สามารถแปลงหน้าเว็บเพจได้หลากหลายรูปแบบมากยิ่งขึ้น

1.4 วิธีการดำเนินงาน

1. ศึกษา ค้นคว้า และรวบรวมความรู้พื้นฐานเกี่ยวกับเว็บและเว็บ
2. ติดตามการเปลี่ยนแปลงของเทคโนโลยีเว็บ
3. ศึกษาภาษา HTML, WML, XML และ XHTML
4. ศึกษาเครื่องมือที่ใช้ในการเขียนและแสดงผลภาษา WML
5. ฝึกการเขียนภาษา HTML, WML, XML และ XHTML
6. ศึกษาการทำงานของเว็บเซิร์ฟเวอร์(Web Server) ที่ชื่อว่า TOMCAT และทดลองการร้องขอเอกสาร HTML จากอินเทอร์เน็ต
7. ศึกษาการทำเอกสาร WML แบบไดนามิกด้วย Servlet ทดลองเขียน Servlet บน TOMCAT

8. ศึกษาการเขียนภาษา JAVA และศึกษา API ที่มีประโยชน์ในการนำมาใช้งาน
9. ออกแบบวิธีการนำเสนอเอกสาร WML บนโทรศัพท์มือถือ
10. เขียนโปรแกรมเพื่อแปลงภาษา HTML เป็น WML
11. ทดสอบการทำงานของโปรแกรม และแก้ไขปัญหาที่เกิดขึ้น



บทที่ 2

ทฤษฎี

2.1 โพรโทคอล WAP (Wireless Application Protocol)

เว็บเป็นการรวมเอาเทคโนโลยีทางเน็ตเวิร์ก 2 ตัวคือ ข้อมูลแบบไร้สาย (wireless data) และ อินเทอร์เน็ตเข้าไว้ด้วยกัน ซึ่งตลาดทางด้านข้อมูลแบบไร้สายและอินเทอร์เน็ตได้มีอัตราการเจริญเติบโตอย่างรวดเร็วในปัจจุบัน ทำให้เกิดรูปแบบการให้บริการทางข้อมูลใหม่ๆ ขึ้นมาตลอดเวลา

เทคโนโลยีอินเทอร์เน็ตส่วนใหญ่ออกแบบมาสำหรับการทำงานบนเครื่องคอมพิวเตอร์ที่มี ขนาดหน้าจอใหญ่ มีย่านความถี่ (Bandwidth) ตั้งแต่ปานกลางถึงสูง และระบบเน็ตเวิร์กที่มีความมั่นคง ซึ่งต่างจากสภาพแวดล้อมการทำงานของเครือข่ายแบบไร้สาย เนื่องจากข้อจำกัดต่างๆ ของอุปกรณ์ไร้สาย ดังนี้

- ชีพียูมีพลังในการประมวลผลต่ำ
- หน่วยความจำน้อย (ROM และ RAM)
- ข้อจำกัดในด้านแหล่งจ่ายพลังงาน
- ขนาดหน้าจอแสดงผลมีขนาดเล็ก
- อุปกรณ์อินพุตใช้งานยาก (เมื่อเทียบกับแป้นคีย์บอร์ด)

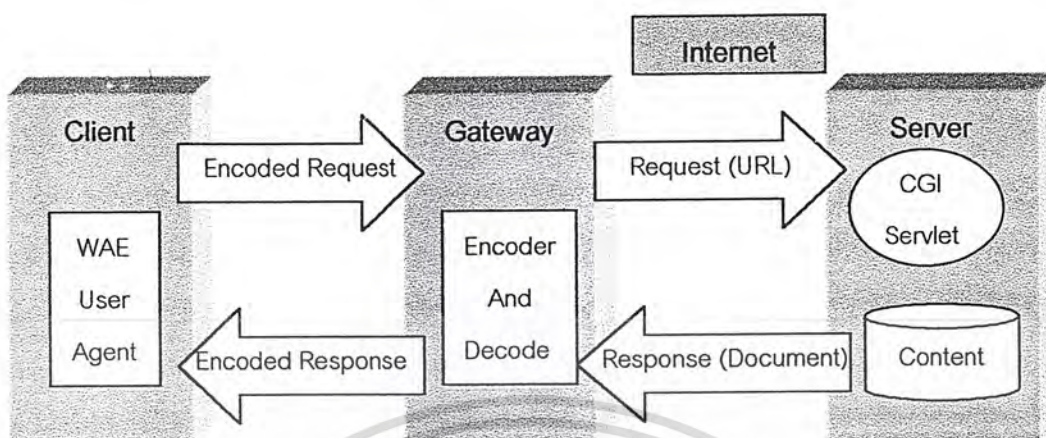
นอกจากนี้ในส่วน of เครือข่าย ไร้สาย ก็ยังมีข้อจำกัดเมื่อเทียบกับเครือข่ายอินเทอร์เน็ต ดังนี้

- มีย่านความถี่แคบ
- ระยะเวลาในการส่งข้อมูลผ่านเครือข่าย (latency) มากกว่า
- สภาพการเชื่อมต่อมีความเสถียรน้อยกว่า เช่น เมื่อขับรถเข้าอุโมงค์จะพบว่าสัญญาณเสียงในโทรศัพท์มือถือจะขาดหายไปบางครั้ง

เครือข่ายโทรศัพท์มือถือในปัจจุบันจะมีบริการต่างเพื่อให้บริการผู้ใช้ ซึ่งมีการพยายามที่จะ สร้างบริการใหม่ๆ เพื่อเพิ่มการใช้งาน ซึ่งหนึ่งในบริการนั้นก็คือเว็บที่เพิ่มความสามารถของโทรศัพท์มือถือให้มากกว่าที่จะแค่ส่งข้อมูลเสียงเท่านั้น

2.1.1 WAP Model

โมเดลของเว็บจะคล้ายคลึงกับโมเดลของ WWW เพื่อประโยชน์ต่างๆ ในการพัฒนาแอปพลิเคชัน และรูปแบบในการเขียนโปรแกรม รวมถึงเครื่องมือในการพัฒนาที่มีอยู่แล้ว (เช่น เว็บเซิร์ฟเวอร์, XML tool เป็นต้น) มีการลดขนาดและเพิ่มฟังก์ชันบางอย่างเข้าไปเพื่อให้เหมาะสมกับสภาพแวดล้อมของเครือข่ายไร้สาย



รูปที่ 2.1 WAP Model

รูปแบบและแอปพลิเคชันของเว็บจะอยู่บนฐานของ WWW การส่งคอนเท้นต์จะใช้มาตรฐานการเชื่อมต่อบนมาตรฐานของ WWW อีกทั้งไมโครบราวเซอร์ที่ใช้ในอุปกรณ์ไร้สายก็อยู่บนพื้นฐานของเว็บบราวเซอร์

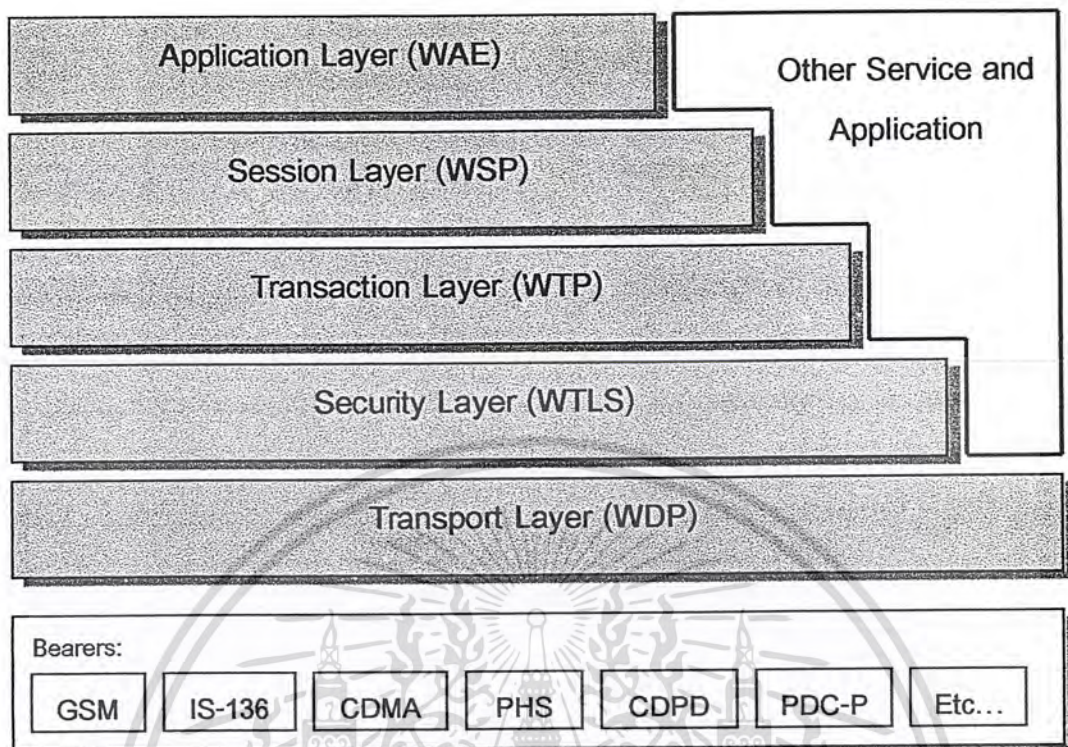
เว็บได้กำหนดกลุ่มของคอมโพเน้นต์ที่ทำหน้าที่ติดต่อสื่อสารระหว่างโทรศัพท์มือถือและเซิร์ฟเวอร์ ดังนี้

- มาตรฐานในการระบุที่อยู่ (Standard naming model) – ใช้ URLs ในการระบุคอนเท้นต์บนเซิร์ฟเวอร์ URIs ใช้ระบุ resource ในระบบ
- ชนิดของคอนเท้นต์ (Content typing) – คอนเท้นต์ทั้งหมดของเว็บจะเหมือนกับคอนเท้นต์ของ WWW เพื่อให้ผู้ใช้สามารถประมวลผลคอนเท้นต์ได้อย่างถูกต้อง
- มาตรฐานรูปแบบของคอนเท้นต์ (Standard content formats) – คอนเท้นต์ของเว็บจะอยู่บนพื้นฐานของเทคโนโลยี WWW รวมถึงการแสดงผล, จัดข้อมูล, รูปภาพ และภาษาสคริปต์
- มาตรฐานการเชื่อมต่อ (Standard communication protocols) – มาตรฐานการเชื่อมต่อของเว็บจะสื่อสารระหว่างบราวเซอร์กับเซิร์ฟเวอร์

คอนเท้นต์และโพรโตคอลของเว็บจะถูกลดขนาดลงเพื่อเหมาะสมกับอุปกรณ์ไร้สาย เว็บใช้ Proxy ในการเชื่อมต่อระหว่างเครือข่ายไร้สายกับ WWW

2.1.2 สถาปัตยกรรมของเว็บ (WAP Architecture)

สถาปัตยกรรมของเว็บถูกสร้างให้มีความสามารถในการขยายขนาดของการพัฒนาแอปพลิเคชันสำหรับการสื่อสารของอุปกรณ์ไร้สาย ด้วยสิ่งนี้ทำให้มีการแบ่งมาตรฐานออกเป็นระดับชั้นที่เรียกว่า "โพรโตคอล สแต็ก (Protocol stack)" แต่ละระดับชั้นจะถูกเข้าถึงโดยระดับชั้นที่อยู่ด้านบน



รูปที่ 2.2 WAP Architecture

แต่ละระดับชั้นของสแต็กจะมีอินเตอร์เฟสไปยังระดับชั้นที่สูงกว่า หมายความว่าชั้นที่อยู่ต่ำกว่าจะไม่สามารถเข้าถึงชั้นที่อยู่สูงกว่าได้ แต่แอปพลิเคชันภายนอกอื่นๆสามารถเข้าถึงระดับ Session, Transaction, Security และ Transport ได้โดยตรง

Wireless Application Environment (WAE)

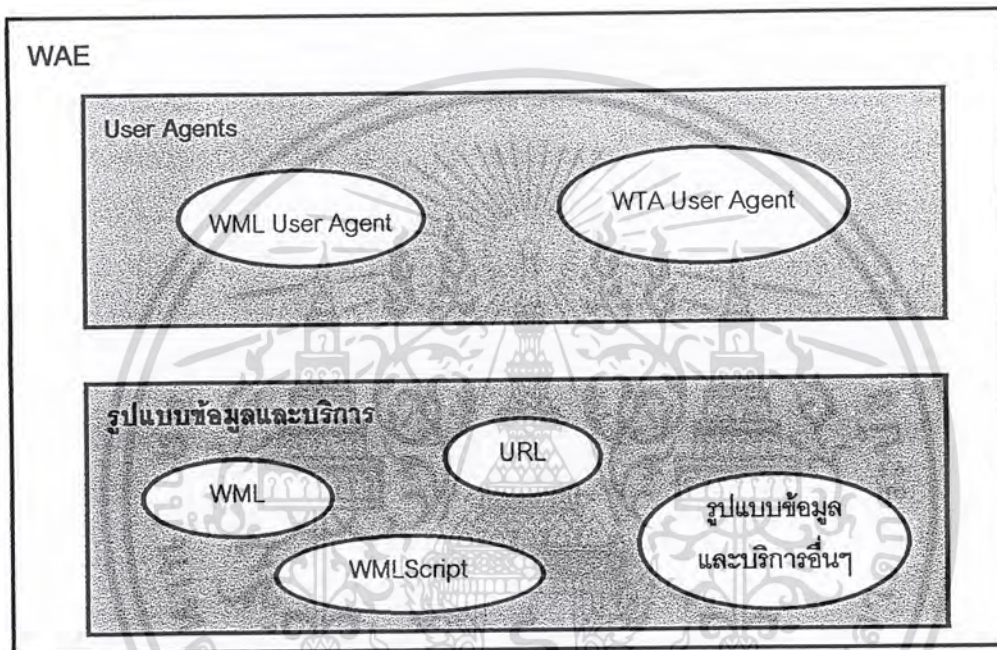
สภาพแวดล้อมของแอปพลิเคชันของแรว์พ (WAE) เป็นสภาพแวดล้อมของแอปพลิเคชันเพื่อใช้ในการเชื่อมต่อระหว่างเทคโนโลยี WWW กับเทคโนโลยีแบบไร้สาย จุดประสงค์ของ WAE ก็เพื่อสร้างสภาพแวดล้อมที่อนุญาตให้ผู้ทำงานและผู้พัฒนาบริการสร้างแอปพลิเคชันและบริการอยู่บนแพลตฟอร์มใดก็ได้ได้อย่างมีประสิทธิภาพและตรงตามความต้องการของผู้ใช้ WAE จะมีโมโครบราวเซอร์ที่ประกอบด้วยฟังก์ชันการทำงานต่อไปนี้

- Wireless Markup Language (WML) – เป็นภาษาที่มีขนาดเล็ก มีความคล้ายคลึงกับ HTML แต่จะถูกลดขนาดลงเพื่อความเหมาะสมต่อการใช้งานบนอุปกรณ์ไร้สาย
- WMLScript – เป็นภาษาสคริปต์ที่มีขนาดเล็ก คล้ายกับจาวาสคริปต์
- Wireless Telephony Application (WTA, WTAI) – เป็นบริการบนโทรศัพท์และอินเตอร์เฟสของการเขียนโปรแกรม

- Content Format – เป็นรูปแบบของข้อมูล รวมถึงรูปภาพ, สมุดโทรศัพท์ และการแสดงข้อมูล

WAE จะมี User agent 2 ตัวคือ

1. WML User Agent เช่น เว็บเบราว์เซอร์ หรือไมโครเบราว์เซอร์ซึ่งติดตั้งอยู่บนโทรศัพท์มือถือ หรือใน เวิร์มูเลเตอร์ (WAP Emulator) ต่างๆ
2. WTA User Agent จะทำงานในส่วนที่เกี่ยวข้องกับฟังก์ชันการทำงานของโทรศัพท์



รูปที่ 2.3 ส่วนประกอบของ WAE

Wireless Session Protocol (WSP)

WSP ประกอบด้วยอินเตอร์เฟซสำหรับ 2 ส่วนของการบริการ ส่วนแรกคือบริการการเชื่อมต่อแบบ Connection-Oriented ที่ทำงานอยู่บนระดับชั้น Transaction Protocol (WTP) ส่วนที่ 2 คือบริการการเชื่อมต่อแบบ Connectionless ที่ทำงานอยู่บนบริการแบบดาต้าแกรม (WDP)

WSP ประกอบด้วยบริการที่ร้องขอจากเบราว์เซอร์แอปพลิเคชัน (WSP/B) ซึ่ง (WSP/B) จะประกอบด้วยฟังก์ชันต่อไปนี้

- ฟังก์ชันในการเข้ารหัสและแปลความหมายของ HTTP/1.1
- Long-lived session state
- การพัก session และกลับมาทำงาน (Session suspend and resume with session migration)
- ความสะดวกในการใส่ข้อมูล

- การทำงานข้ามโพรโตคอล

มาตรฐานใน WSP จะถูกลดขนาดลงเพื่อให้เหมาะสมกับการใช้งานบนเครือข่ายที่มีย่านความถี่ตัวนำต่ำ และระยะเวลาในการส่งข้อมูลนาน WSP/B ถูกออกแบบให้ WAP proxy ติดต่อ WSP/B ของไคลเอนต์ กับ เซิร์ฟเวอร์ที่เป็น HTTP

Wireless Transaction Protocol (WTP)

WTP เกี่ยวข้องกับการรับประกันความน่าเชื่อถือของการส่งข้อมูล WTP ทำงานอยู่บนชั้นบนสุดของการให้บริการแบบค่าเช่าแก็ม และใช้ light-weight transaction-oriented protocol ที่เหมาะสมกับอุปกรณ์ไร้สาย WTP ทำงานได้อย่างมีประสิทธิภาพอยู่บนเครือข่ายแบบค่าเช่าแก็มแบบไร้สายทั้งที่มีระบบความปลอดภัยและไม่มี ซึ่งมีลักษณะสำคัญ ดังนี้

- มีคลาสในการให้บริการ 3 คลาส
 - Unreliable one-way requests
 - Reliable one-way requests
 - Reliable two-way request-reply transactions
- ผู้ใช้สามารถเลือกระดับความน่าเชื่อถือได้
- ผู้ใช้สามารถเลือกระดับข้อมูลผิดพลาดที่ยอมรับได้
- มีการหน่วงเวลาของ ACK เพื่อลดจำนวนข้อมูลที่ส่ง
- Asynchronous transaction

Wireless Transaction Layer Security (WTLS)

จุดประสงค์ของ WTLS คือการสร้างความปลอดภัยในระดับชั้นการส่งข้อมูลระหว่างไคลเอนต์กับเว็บเกตเวย์ ซึ่ง WTLS อยู่บนพื้นฐานความปลอดภัยในชั้นการขนส่ง (TLS: Transport Layer Security) แต่ถูกลดขนาดลงเพื่อเหมาะสมกับข้อสัญญาที่แคบ WTLS มีลักษณะสำคัญ ดังนี้

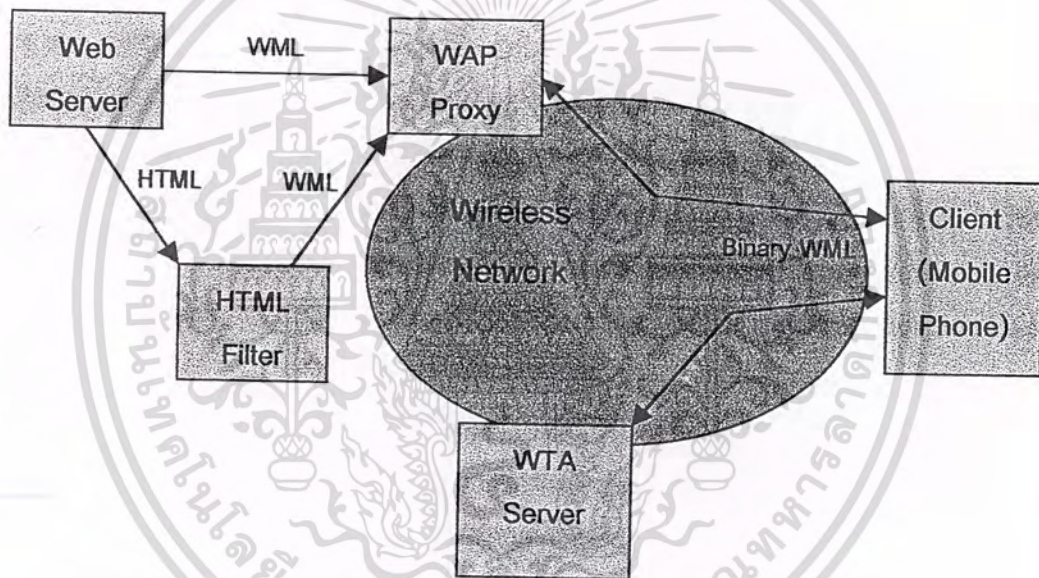
- ความถูกต้องของข้อมูล (Data integrity) - รับประกันว่าข้อมูลที่ส่งระหว่างไคลเอนต์และเซิร์ฟเวอร์จะไม่ถูกเปลี่ยนแปลง
- ความเป็นส่วนตัว (Privacy) - รับประกันว่าข้อมูลที่ส่งจะไม่สามารถถูกอ่านโดยคนอื่นได้ นอกจากเจ้าของเท่านั้น
- การพิสูจน์ตน (Authentication) - มีการพิสูจน์ตนระหว่างไคลเอนต์และเซิร์ฟเวอร์
- การป้องกัน Denial-of-service - มีการป้องกันการโจมตีแบบ Denial-of-service

แอปพลิเคชันสามารถเลือกที่จะเอา WTLS หรือไม่เอาได้ขึ้นอยู่กับระดับความปลอดภัยที่ต้องการและลักษณะของเครือข่าย

Wireless Datagram Protocol (WDP)

ระดับชั้นในการส่งข้อมูลของเว็บจะใช้เทคโนโลยีแบบค้ำแกรม ซึ่งทำงานอยู่บนการให้บริการส่งข้อมูลโดย ตัวกลาง (Bearer) ที่รองรับเครือข่ายแบบต่างๆ ถ้าเว็บถูกใช้บนตัวกลางที่สนับสนุน UDP (User Datagram Protocol) แล้ว WDP ก็ไม่จำเป็นต้องใช้ แต่บนตัวกลางอื่นๆ เช่น GSM, SMS นั้นฟังก์ชันค้ำแกรมถูกจัดเตรียมโดย WDP ดังนั้นเว็บจึงใช้ UDP หรือไม่ก็ WDP ในการให้บริการค้ำแกรมที่มีการซ่อนลักษณะของตัวกลางที่แตกต่างกันไว้ และเตรียมฟังก์ชัน port number และถ้าจำเป็น WDP ก็สามารถขยายฟังก์ชันสำหรับการแตก (fragment) และการรวม (reassemble) ของค้ำแกรมที่ใหญ่เกินไปสำหรับตัวกลางนั้นๆ

2.1.3 ตัวอย่างของเครือข่ายเว็บ



รูปที่ 2.4 ตัวอย่างเครือข่ายของเว็บ

ในตัวอย่างนี้ เว็บไคลเอนต์จะสื่อสารกับเซิร์ฟเวอร์ในเครือข่าย 2 ตัว Proxy ของเว็บทำหน้าที่ที่เปลี่ยน request ของเว็บไปเป็น request ของ WWW เพื่อให้สามารถส่ง request ไปยังเว็บเซิร์ฟเวอร์ได้ Proxy ยังเปลี่ยน response จากเว็บเซิร์ฟเวอร์ ไปเป็นรูปแบบของเว็บเพื่อไคลเอนต์สามารถรับได้

ถ้าเว็บเซิร์ฟเวอร์มีคอนเทนต์ของเว็บอยู่แล้ว (เช่น WML) Proxy ก็จะมีการส่งไปให้ไคลเอนต์โดยตรงโดยไม่มีการเปลี่ยนแปลงแต่อย่างใด แต่ถ้าเว็บเซิร์ฟเวอร์มีแต่คอนเทนต์ของ WWW (เช่น HTML) ตัวกรอง (Filter) จะทำการเปลี่ยนคอนเทนต์ WWW ไปเป็นคอนเทนต์ของเว็บ เช่นการเปลี่ยนจาก HTML ไปเป็น WML อย่างที่โปรแกรมนี้ทำ

WTA จะทำการติดต่อกับเว็บไคลเอนต์โดยตรงเพราะจะมีคอนเท้นต์ของเว็บอยู่แล้วไม่ต้องผ่านตัวกรอง

2.2 โพรโทคอล HTTP (HyperText Transfer Protocol)

เป็นโพรโทคอลที่ใช้ในการติดต่อสื่อสารระหว่างเว็บเบราว์เซอร์และเว็บเซิร์ฟเวอร์สำหรับเครือข่ายเวิลด์ไวด์เว็บ โดยใช้คอนเซ็ปต์ของไคลเอนต์-เซิร์ฟเวอร์เป็นหลัก หลักการทำงานพื้นฐานของ HTTP คือการส่งข้อความจากไคลเอนต์ไปยังเซิร์ฟเวอร์เพื่อทำการประมวลผล หลังจากนั้นเซิร์ฟเวอร์จะทำการส่งข้อความซึ่งเป็นผลลัพธ์จากการประมวลผลกลับมายังไคลเอนต์ ซึ่งโดยทั่วไปการส่งข้อความระหว่างไคลเอนต์และเซิร์ฟเวอร์มักใช้ ไลบรารีเป็นหลัก แต่ HTTP เป็นโพรโทคอลที่ถูกออกแบบมาเพื่อให้สามารถใช้งานได้ง่ายและมีประสิทธิภาพสูง ดังนั้นจึงมีการส่งข้อความเป็นเท็กซตรูมแทน ซึ่งหลักการทำงานอย่างง่ายของโพรโทคอล HTTP มีดังนี้

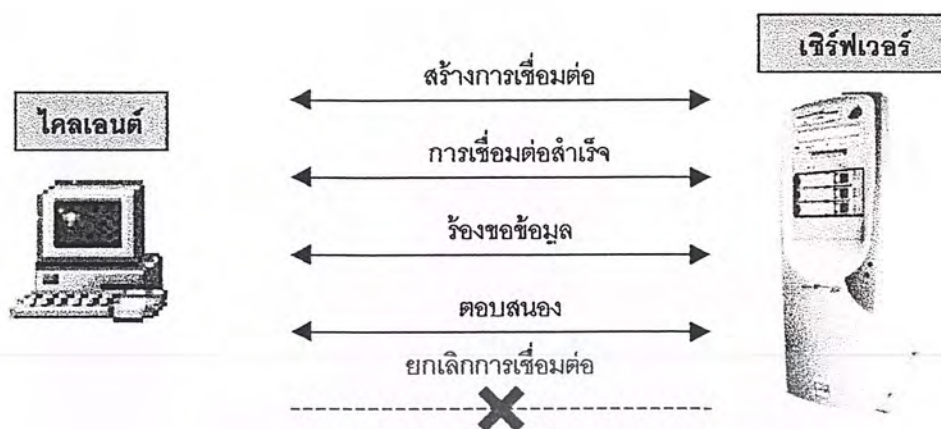
HTTP ย่อมาจากคำว่า Hypertext Transfer Protocol ซึ่งเป็นโพรโทคอลที่ใช้ในการรับ-ส่งข้อมูลต่างๆ ในโลกของเวิลด์ไวด์เว็บ (WWW) ข้อมูลต่างๆ เหล่านี้โดยทั่วไปมักจะถูกเรียกว่า Resource โดย Resource เหล่านี้อาจเป็นไฟล์ HTML, ไฟล์รูปภาพ หรือคำสั่งต่างๆ (Query String) เช่น คำสั่งที่ส่งไปที่ CGI ของโปรแกรม หรืออาจจะเป็นไบนารีสตรีมในกรณีของการดาวน์โหลดหรืออัปโหลดไฟล์ หรืออาจจะเป็นสิ่งอื่นๆ ตามแต่จะกำหนดขึ้น

HTTP เป็นโพรโทคอลที่อยู่ในส่วนของแอปพลิเคชันเลเยอร์ในโพรโทคอลสแต็ก โดยข้อมูลต่างๆ จากเลเยอร์นี้จะถูกส่งผ่านต่อไปยังเลเยอร์อื่นๆ ที่อยู่ต่ำกว่า ซึ่งส่วนหนึ่งในนั้นก็คือโพรโทคอล TCP/IP นั่นเอง

HTTP เป็นเน็ตเวิร์กโพรโทคอลที่ใช้หลักการของโมเดลไคลเอนต์-เซิร์ฟเวอร์ ในการติดต่อสื่อสารซึ่งหลักการทำงานอย่างคร่าวๆ มีดังนี้

1. HTTP ไคลเอนต์จะทำการสร้างการเชื่อมต่อไปหา HTTP เซิร์ฟเวอร์ซึ่งโดยทั่วไปจะทำงานผ่านทางซ็อกเก็ตของ TCP/IP
2. หลังจากนั้น HTTP ไคลเอนต์ จะทำการส่งคำสั่งการร้องขอ (request) ซึ่งอยู่ในรูปของข้อความ (เมสเสจ) ไปให้ HTTP เซิร์ฟเวอร์เพื่อขอ resource ที่ต้องการ
3. HTTP เซิร์ฟเวอร์จะทำการแปลความคำสั่งที่ได้และจะส่งผลตอบแทนกลับ (response) ซึ่งเป็น resource ที่ HTTP ไคลเอนต์ต้องการกลับมา (ผลที่ส่งกลับมากจะเป็นลักษณะของเมสเสจคล้ายกับการร้องขอของ HTTP ไคลเอนต์ที่ส่งมาให้ HTTP เซิร์ฟเวอร์)
4. หลังจากทำการส่งการตอบรับเสร็จสิ้น HTTP เซิร์ฟเวอร์จะทำการปิดการเชื่อมต่อที่มาจาก HTTP ไคลเอนต์

ในกรณีที่ HTTP ไคลเอนต์ต้องการ resource อื่นๆ HTTP ไคลเอนต์จะต้องทำการสร้างการเชื่อมต่อใหม่และส่งคำร้องขอไปยัง HTTP เซิร์ฟเวอร์อีกครั้ง



รูปที่ 2.5 การติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์

ในการทำงานของเว็บเบราว์เซอร์ก็คือ HTTP ไคลเอนต์นั่นเอง เหตุผลคือเว็บเบราว์เซอร์ถูกใช้เป็นตัวส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์ (หรือ HTTP เซิร์ฟเวอร์) เพื่อร้องขอ resource ที่ต้องการจากเว็บเซิร์ฟเวอร์ จากนั้นจะรับ resource นั้นกลับมาโดยใช้โปรโตคอล HTTP โดยทั่วไป resource จะกระจายอยู่ตามโหนดของเครือข่ายต่างๆทั่วโลกสิ่งที่จะขาดไม่ได้ในการอ้างถึง resource ใดๆก็ได้โดยใช้โปรโตคอลอะไรก็ได้ ไม่ใช่สำหรับ HTTP เพียงอย่างเดียวเช่น

<http://161.246.5.117/home.html>

เป็น URL ของโปรโตคอล HTTP เพื่อใช้โหลดไฟล์ HTML

<http://161.246.5.117:80/cat.jpg>

เป็น URL ของโปรโตคอล HTTP เพื่อใช้โหลดไฟล์รูปภาพ

<http://161.246.5.117/cgi-bin/sendMail.pl>

เป็น URL ของโปรโตคอล HTTP เพื่อใช้โหลดค่าที่ได้จากการประมวลผลของ CGI ของโปรแกรมชื่อ sendMail.pl

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

เป็น URL ของโปรโตคอล FTP เพื่อใช้ดาวน์โหลดไฟล์เอกสาร

จากตัวอย่างข้างต้น จะเห็นได้ว่าในหนึ่ง URL จะประกอบไปด้วยชื่อของโปรโตคอล, ชื่อของเซิร์ฟเวอร์, พอร์ตที่ใช้ และ resource ที่ต้องการ

2.2.1 โครงสร้าง HTTP (HTTP Structure)

จากหลักการทำงานของ HTTP จะเห็นได้ว่าการติดต่อสื่อสารระหว่างไคลเอนต์กับเซิร์ฟเวอร์แล้ว ตัวเมสเสจ(message) จะเป็นตัวกลางที่ใช้ในการติดต่อสื่อสารเสมอ และเพื่อให้ง่ายต่อ

การใช้งานรูปแบบของเมสเสจที่ใช้ในการร้องขอหรือการตอบรับจึงมีลักษณะคล้ายๆกัน โดยใช้เท็กส์เป็นหลักซึ่งโครงสร้างของเมสเสจจะประกอบด้วย

1. บรรทัดเริ่มต้น (initial line)
2. เฮดเดอร์ (header)
3. บรรทัดว่าง (a blank line) ซึ่งก็คือ CRLF หรือการเว้นหนึ่งบรรทัด (วิธีหนึ่งที่ทำให้คือการกด Enter นั่นเอง)
4. เมสเสจบอดี ซึ่งอาจจะใช้บรรทัด, คำสั่ง (Query String) หรืออาจจะเป็นเอาต์พุตที่มาจากเซิร์ฟเวอร์โดยส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับจุดประสงค์ของการใช้งาน

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

```
<initial line>
เฮดเดอร์1: value1
เฮดเดอร์2: value2
เฮดเดอร์3: value3
[blank line]
<เมสเสจ body, optional.....>
```

ตัวอย่างง่ายๆของเมสเสจที่ไคลเอนต์ใช้อาจจะเป็น

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

ตัวอย่างง่ายๆของเมสเสจที่เซิร์ฟเวอร์ใช้อาจจะเป็น

```
HTTP 1.0 200 OK ← บรรทัดเริ่มต้น
```


Date: Sunday, 23-July-00 04:01:12 GMT ← เซคเตอร์
 Server: Apache/1.3.12(Unix) (Red Hat/Linux) PHP/3.0.15 mod_perl/1.21 ← เซคเตอร์
 MINE-version: 1.0 ← เซคเตอร์
 Content-type: text/html ← เซคเตอร์
 Content-length: 115 ← เซคเตอร์
 [blank line]
 <HTML><HEAD><TITLE>HTTP Tutorial</TITLE></HEAD> ← เมสเชจบอดี
 <BODY>This is a tutorial, but please visit me again...</BODY> ← เมสเชจบอดี(cont.)
 </HTML> ← เมสเชจบอดี(cont.)

2.2.1.1 บรรทัดเริ่มต้นในการร้องขอ (Initial Request Line)

บรรทัดเริ่มต้นของการร้องขอ(จากไคลเอนต์)จะแตกต่างจากบรรทัดเริ่มต้นของการตอบรับเล็กน้อย โดยบรรทัดเริ่มต้นของการร้องขอจะมี 3 ส่วนคือ

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

สามส่วนนี้จะประกอบกันเป็นบรรทัดเริ่มต้น โดยแต่ละส่วนจะถูกแยกออกจากกันโดยใช้ช่องว่าง (space) ดังตัวอย่างข้างล่าง

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

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

การใช้โคลอกลพาร (Local path)

ก่อนที่จะ HTTP ไคลเอนต์จะส่งการร้องขอไปที่ HTTP เซิร์ฟเวอร์ HTTP ไคลเอนต์จะสร้างการเชื่อมต่อขึ้นมาก่อนหนึ่งก่อน โดยใช้ชื่อเรียก ในการสร้างชื่อเรียกจะต้องทำการกำหนดชื่อของโฮสต์ที่จะติดต่อ ยกตัวอย่างเช่น ถ้า URL เป็น <http://161.246.5.117/path/to/file/index.html> ชื่อของโฮสต์ก็จะเป็น 161.246.5.117 ดังนั้นการกำหนดเพียงโคลอกลพารเพื่อบ่งบอกถึง resource ที่ต้องการในส่วน of บรรทัดเริ่มต้นในการร้องขอก็คือ /path/to/file/index.html ก็เพียงพอแล้ว

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

ข้อระวัง

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

2.2.1.2 บรรทัดเริ่มต้นในการตอบสนอง (Initial Response Line)

โดยทั่วไปบรรทัดเริ่มต้นในการตอบสนอง(จากเซิร์ฟเวอร์)มักจะเรียกว่า Status line ซึ่งจะประกอบไปด้วย 3 ส่วนย่อยคือ

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

HTTP/1.0 200 OK

หรือ

HTTP/1.0 404 Not Found

ข้อสังเกต

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

โค้ดแสดงสถานะ (Status code) ที่พบเห็นโดยทั่วไป คือ

- 100 Continue หมายถึง เซิร์ฟเวอร์ได้รับ request บางส่วนจากไคลเอนต์แล้ว บอกให้ไคลเอนต์ส่งที่เหลือมาด้วย
- 200 OK หมายถึง คำร้องขอที่มาจากไคลเอนต์ถูกต้องและ response ที่ไคลเอนต์ต้องการอยู่ในเมสเสจบอดีของ response นี้แล้ว
- 301 Moved Permanently หมายถึง resource ที่ไคลเอนต์ต้องการได้ถูกย้ายไปที่อื่นแล้ว
- 302 Moved Temporarily หมายถึง resource ที่ไคลเอนต์ต้องการได้ถูกย้ายไปอยู่ที่อื่นชั่วคราว
- 400 Bad Request หมายถึง เซิร์ฟเวอร์ไม่สามารถเข้าใจการร้องขอที่ไคลเอนต์ส่งมา

- 403 Forbidden หมายถึง เซิร์ฟเวอร์เข้าใจการร้องขอที่ไคลเอนต์ส่งมาแต่ไม่สามารถที่จะส่ง resource ที่ไคลเอนต์ต้องการกลับไปได้
- 404 Not Found หมายถึง เซิร์ฟเวอร์ไม่มี resource ที่ร้องขอมา
- 500 Server Error หมายถึงข้อผิดพลาดที่เกิดขึ้นในส่วนของเซิร์ฟเวอร์ (โดยทั่วไปข้อผิดพลาดนี้จะเกิดขึ้นในกรณีที่การร้องขอส่งมาเรียกแอปพลิเคชันฝั่งเซิร์ฟเวอร์ที่รันบนเซิร์ฟเวอร์แต่เกิดข้อผิดพลาดขึ้นระหว่างการประมวลผล ซึ่งโดยทั่วไปจะเกิดขึ้นเนื่องจากโปรแกรมมีบั๊ก (bug))

2.2.1.3 เฮดเดอร์ (Header)

เฮดเดอร์จะเป็นส่วนที่บอกถึงรายละเอียดของการร้องขอ (ที่กำลังส่งไปให้เซิร์ฟเวอร์) หรือการตอบรับ (ที่กำลังส่งกลับมายังไคลเอนต์) ยกตัวอย่างเช่น

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

From: admin@ce.kmitl.ac.th

User-Agent: Mozilla/4.72

Content-Type: text/html

Content-Length: 250

ข้อสังเกต

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

เฮดเดอร์ที่มักจะพบเห็นทั่วไปในส่วนของไคลเอนต์

From: เป็นเฮดเดอร์ที่ใช้เก็บอีเมลแอดเดรสของผู้ที่กำลังส่งการร้องขอ ยกตัวอย่างเช่น

From: admin@ce.kmitl.ac.th

User-Agent: เป็นเฮดเดอร์ที่ใช้บ่งบอกถึงชื่อของโปรแกรมที่ใช้เป็น HTTP ไคลเอนต์ ยกตัวอย่างเช่น User-Agent: Mozilla/4.72

เฮดเดอร์ที่มักจะพบเห็นทั่วไปในส่วนของเซิร์ฟเวอร์

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

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

2.2.1.4 เมสเสจบอดี้ (Message Body)

เมื่อไรก็ตามที่ไคลเอนต์หรือเซิร์ฟเวอร์ต้องการส่งข้อมูลไปกับเมสเสจ ส่วนที่เป็นเมสเสจบอดี้
 จะเป็นส่วนที่ใช้เก็บข้อมูลดังกล่าว ในกรณีของไคลเอนต์ตัวเมสเสจบอดี้อาจใช้สำหรับบรรจุไฟล์ที่
 ต้องการอัปโหลดหรือใช้เก็บข้อมูลที่ได้จากอิลิเมนต์ต่างๆ ของฟอร์ม HTML แล้วส่งไปยังเซิร์ฟเวอร์ก็
 ได้ ในกรณีของเซิร์ฟเวอร์ตัวเมสเสจบอดี้จะเป็นส่วนที่ใช้เก็บ resource ที่ไคลเอนต์ร้องขอหรืออาจใช้
 เก็บคำอธิบายต่างๆ ในกรณีที่มีข้อผิดพลาดเกิดขึ้นก็ได้ ถ้าเมสเสจมีส่วนของเมสเสจบอดี้ ไคลเอนต์
 หรือเซิร์ฟเวอร์มักจะเพิ่มเฮดเดอร์ที่ช่วยบ่งบอกถึงรายละเอียดของเมสเสจบอดี้ดังกล่าวด้วย ยกตัวอย่าง
 เช่นถ้าเซิร์ฟเวอร์ต้องการส่ง resource ที่เป็นไฟล์รูปภาพมาให้ไคลเอนต์ เฮดเดอร์ที่ถูกเพิ่มมาก็จะเป็น
 Context-Type: image/gif เป็นเฮดเดอร์ที่บ่งบอกถึงMIME type ของข้อมูลที่อยู่ในเมสเสจบอดี้ Context-
 Length: 1026 เป็นเฮดเดอร์ที่บ่งบอกถึงขนาดของเมสเสจบอดี้ในหน่วยไบต์

ตัวอย่างของการส่งเมสเสจระหว่าง ไคลเอนต์ (Web Browser) และเซิร์ฟเวอร์ (Web Server) เช่น

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

ขั้นแรกไคลเอนต์จะทำการเรียกใช้ช็อกเก็ตเพื่อเชื่อมต่อไปยังเซิร์ฟเวอร์ชื่อ 161.246.5.117 ซึ่ง
 โดยปกติจะติดต่อไปที่พอร์ต 80 ซึ่งเป็นพอร์ต Default ของโปรโตคอล HTTP (ในกรณีที่ต้องการเชื่อม
 ต่อไปยังพอร์ตอื่นก็สามารถทำได้โดยการเพิ่มส่วนของเลขพอร์ตที่ไคลเอนต์ต้องการเชื่อมต่อเข้าไป
 หลังจากส่วนของโฮสต์เนม เช่น <http://161.246.5.117:8080/tutorials/advance/helloworld.html> ทำให้
 ไคลเอนต์เชื่อมต่อไปยังพอร์ต 8080 แทน)

หลังจากนั้นไคลเอนต์จะทำการส่งเมสเสจร้องขอ

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

From: soup@jarticles.com

User-Agent: Mozilla/4.72

[blank line]

หลังจากที่เซิร์ฟเวอร์ได้รับเมสเสจข้างบนแล้ว เซิร์ฟเวอร์ส่งการตอบรับพร้อมทั้ง resource
 ที่ไคลเอนต์ต้องการกลับมามดังนี้

HTTP/1.0 200 OK

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

Content-Type: text/html

Content-Length: 1556

[blank line]

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

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

(more file contents)

...

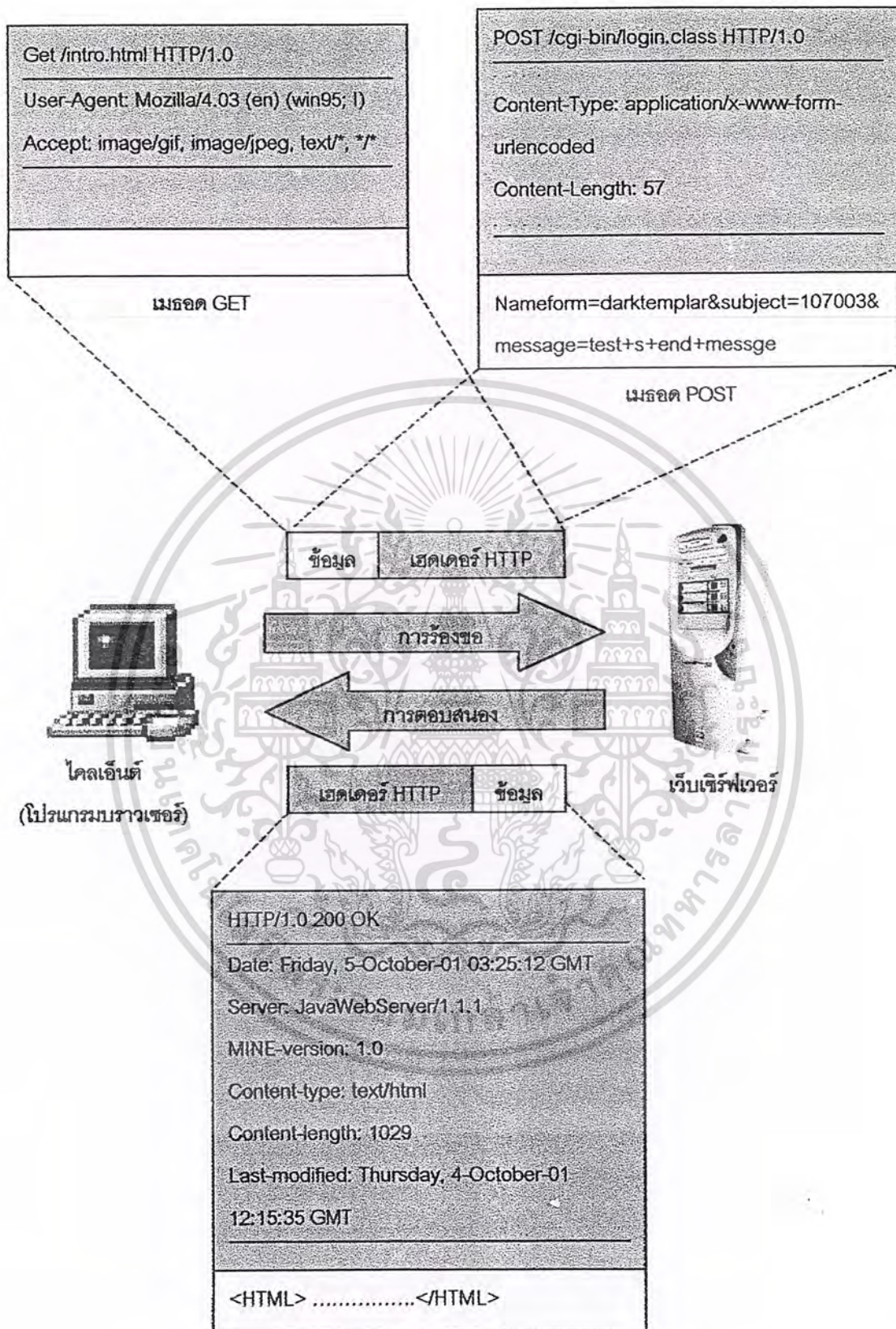
...

</BODY>

</HTML>

ท้ายสุดเซิร์ฟเวอร์จะทำการปิดการเชื่อมต่อของไคลเอนต์





รูปที่ 2.6 ลักษณะข้อมูลที่รับส่งระหว่างไคลเอ็นต์และเซิร์ฟเวอร์

2.2.2 รูปแบบของการร้องขอ

เป็นรูปแบบในการส่งข้อมูลไปยังเซิร์ฟเวอร์

เมธอด GET

ยกตัวอย่างเช่น <http://www.google.com/search.cgi?keyword=jarticles> เป็นการค้นหาที่ Search engine ของเว็บ www.google.com โดยใช้คีย์เวิร์ด = "jarticles"

ถ้าลองดูส่วนที่อยู่หลังจากส่วนของโฮสต์เนม (www.google.com) แล้วเราจะเห็นว่า URI ที่ถูกเรียก (/search.cgi?keyword=jarticles) ที่เห็นไม่ได้เป็น html ไฟล์อย่างทั่วไป แต่กลับกลายเป็นชื่อของโปรแกรม(search.cgi), เครื่องหมายคำถาม(?) และคีย์เวิร์ดที่เราใช้ค้นหาแทน

อย่างที่กล่าวมาแล้วในส่วนต้นของบทความว่า resource จริง ๆ แล้วก็คือตัวที่ใช้บ่งชี้ถึงค่าที่อยู่ในที่เซิร์ฟเวอร์ resource อาจเป็นไฟล์หรืออาจเป็น query string ที่ใช้ส่งไปเรียกโปรแกรมที่อยู่ในที่เซิร์ฟเวอร์ก็ได้ ซึ่งในกรณีของตัวอย่าง search engine ข้างต้นตัว resource ก็คือโปรแกรมที่ชื่อ search.cgi นั่นเองโดยทั่วไป response ที่ได้กลับมาจากเซิร์ฟเวอร์อาจจะอยู่ในรูปของไฟล์ถ้า resource ที่เราทวงถามเป็นไฟล์ ยกตัวอย่างเช่น <http://www.jarticles.com/tutorial/http.html> หรืออาจจะเป็น output ที่ได้จากการประมวลผลของโปรแกรมที่เราผ่าน query string เข้าไป เช่นถ้าเราส่ง query string ไปที่ search engine ผลที่เราจะได้กลับมาก็คือสิ่งต่างๆที่เกี่ยวข้องกับคีย์เวิร์ดที่เราใช้ค้นหานั่นเอง

โดยทั่วไปแล้วเว็บเบราว์เซอร์ (ไคลเอนต์) จะใช้เมธอด GET ในการส่งเมสเสจร้องไปที่เซิร์ฟเวอร์ในกรณีที่ resource ที่ต้องการเป็นไฟล์ อย่างไรก็ตามเมธอด GET ยังสามารถใช้ในการส่ง query string ขึ้นไปยังโปรแกรมที่รันอยู่ที่เซิร์ฟเวอร์ได้อีกด้วย ยกตัวอย่างเช่น ถ้าเราต้องการส่งคำคำขื่อ keyword โดยมีค่าเท่ากับ jarticles ไปที่เซิร์ฟเวอร์ชื่อ www.google.com โดยจะส่งไปที่ resource ซึ่งเป็น CGI โปรแกรมชื่อ search.cgi วิธีการทำคือ

1. กำหนด resource ลงไปในส่วนของ URI ที่เรียกใช้ ซึ่งจะอยู่หลังจากส่วนของโฮสต์เนมซึ่งจะได้ <http://www.google.com/search.cgi>
2. ใส่ตัวเครื่องหมายคำถาม(?) เพื่อบอกเซิร์ฟเวอร์ว่าส่วนที่เหลือถัดไปจะเป็นส่วนของค่าคำซึ่งจะได้ <http://www.google.com/search.cgi?>
3. ทำการเข้ารหัสโดยวิธีการที่เรียกว่า *URL-encoding* ไปที่ชื่อและค่าของตัวแปรต่างๆ ซึ่งในกรณีของเราตัวแปรก็คือคีย์เวิร์ด ซึ่งจะได้ <http://www.google.com/search.cgi?keyword=jarticles>

ถ้าหากพิมพ์ URL ข้างบนเข้าไปในเว็บเบราว์เซอร์แล้วกดปุ่ม Enter ตัวเว็บเบราว์เซอร์จะทำการความ URL ดังกล่าวซึ่งผลที่ได้คือการใช้ GET ส่งเมสเสจร้องขอไปที่เซิร์ฟเวอร์ที่ชื่อ www.google.com โดยจะแนบคีย์เวิร์ด "jarticles" ไปกับคำร้องขอ ซึ่งบรรทัดเริ่มต้นของเมสเสจร้องขออาจเป็นอย่างตัวอย่างข้างล่างนี้

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

From: soup@jarticles.com

User-Agent: Mozilla/4.72

[blank line]

หลังจากนั้นเราจะได้ลิงค์ต่างๆที่มีคำว่า "jarticles" อยู่ในเว็บนั้นกลับมา สังเกตว่า URL จริงๆที่ใช้กับ www.google.com จะต่างจาก URL ข้างบนเล็กน้อย วิธีการส่ง query string ไปกับ URL ในการใช้ จะเหมาะสำหรับ query string ที่ไม่ยาวมากนัก โดยทั่วไปความยาวของ URL มักจะจำกัดอยู่ที่ 80 ตัวอักษร ซึ่งโดยทั่วไปแล้วตัวอักษรที่ตัดจากนั้นจะถูกตัดออกไปโดยอัตโนมัติ (จะตัดหรือไม่จริงๆแล้วขึ้นอยู่กับจัดการของเซิร์ฟเวอร์แต่ละตัว)

URL-Encoding

ในการส่งค่าไปที่เซิร์ฟเวอร์โดยวิธีการ GET หรือวิธีการ POST (จะพูดถึงถัดไป) นั้นชื่อและค่าของตัวแปรต่างๆซึ่งอยู่หลังจากเครื่องหมายคำถาม(?) ในกรณีของ GET หรืออยู่ในส่วนของเมสเสจบอดี ในกรณีของ POST จะต้องถูกผ่านการเข้ารหัสโดยวิธีการที่เรียกว่า *URL-Encoding* เสียก่อน

โดยทั่วไปแล้ว URL เองจะมีตัวอักษรบางตัวที่ใช้สำหรับความหมายพิเศษ เช่น ., /, ~, &, ? ซึ่งในบางครั้งชื่อและค่าของตัวแปรต่างๆที่ถูกส่งไปที่เซิร์ฟเวอร์อาจจะมีตัวอักษรเหล่านี้ปะปนอยู่ด้วย ดังนั้นเพื่อหลีกเลี่ยงความสับสนในการตีความของเซิร์ฟเวอร์ ตัวอักษรต่างๆที่เป็นตัวอักษรที่ใช้ใน URL จะต้องถูกเข้ารหัสเสียก่อน โดยมีหลักการดังต่อไปนี้

1. ให้ทำการเปลี่ยนตัวอักษร unsafe ต่างๆที่อยู่ในชื่อและค่าของตัวแปรให้กลายเป็น "%xx" โดย "xx" ซึ่งคือค่าแอสกีของตัวอักษรนั้นๆในแบบเลขฐาน 16 ซึ่งตัวอักษร unsafe เหล่านี้รวมไปถึง =, &, %, +, และตัวอักษรต่างๆที่ไม่สามารถพิมพ์ได้หรือแม้กระทั่งตัวอักษรที่เราต้องการจะเข้ารหัสเอง
2. ให้ทำการเปลี่ยนช่องว่าง (space) ทั้งหมดให้กลายเป็นเครื่องหมายบวก (+)
3. จับคู่ชื่อและค่าของตัวแปรต่างๆ ด้วย =
4. ถ้ามีชื่อและค่าของตัวแปรมากกว่าหนึ่งตัว ให้นำชื่อและค่าของตัวแปรแต่ละคู่มาเชื่อมติดกันด้วยเครื่องหมาย &
5. นำชื่อและค่าของตัวแปรที่เชื่อมติดกันทั้งหมดไปใส่ที่ URL โดยมีเครื่องหมาย ? อยู่ข้างหน้า (ในกรณีของ GET) หรือนำไปใส่ที่เมสเสจบอดี (ในกรณีของ POST) ยกตัวอย่างเช่น ถ้าเรามีชื่อและค่า

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

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

หลังจากทำการ encode เราจะได้ String ออกมาเป็น keyword=jarticles&name=Soup+%26+friends โดย String นี้มีความยาวเท่ากับ 39 โดยที่ & = %26 ในเลขฐาน 16

เมธอด POST

บางครั้งหากไม่ต้องการส่งข้อมูลไปยังเซิร์ฟเวอร์ด้วยการติดข้อมูลเหล่านั้นไปกับ URI ด้วย เหตุผลที่ว่าข้อมูลเหล่านั้นอาจเกี่ยวข้องกับข้อมูลส่วนตัว ยกตัวอย่างเช่น login และ password ที่ใช้เป็น ตัวผ่านเข้าไปในเว็บไซด์ต่างๆ ซึ่งในกรณีที่ใช้ GET, URL หลังจากที่ทำ login ไปแล้ว URL ที่อยู่ ตรง Address Bar อาจจะเป็น `http://www.myserver.com/login.cgi?login=me&password=youandme` ซึ่งอาจมีคนเห็นแล้วนำข้อมูลไปใช้อย่างผิดๆ ได้ อีกกรณีหนึ่งคือ กรณีของค่าที่เราติดไปกับส่วนของ URI ที่ร้องขอทำให้ URL มีความยาวมากกว่าความยาวของ URL ที่เซิร์ฟเวอร์กำหนดไว้ ผลกระทบที่ อาจเกิดขึ้นคือค่าบางส่วนของเราอาจหายไปเนื่องจากการตัดทอนโดยเซิร์ฟเวอร์

วิธีการที่เราสามารถใช้เพื่อปัดบังข้อมูล(แบบง่ายๆ) หรือเพื่อส่งค่าที่มีความยาวมากๆ ก็คือ การใช้เมธอด POST โดยที่เมธอด POST จะต่างกับเมธอด GET ตรงที่ว่า POST จะไม่ทำการติดค่า ไปกับ URL แต่ เมธอด POST จะทำการใส่ค่าเข้าไปในส่วนของ เมสเสจบอดีของเมสเสจร้องขอ แทนซึ่งในกรณีนี้ URI ที่ร้องขอจะมีปรากฏเพียงชื่อของ โปรแกรมที่เราต้องการส่งค่าไปให้เท่านั้น

ในการส่งค่าไปในส่วนของเมสเสจบอดี เพื่อให้ง่ายต่อการตีความโดยเซิร์ฟเวอร์ ทางเมธอด POST จึงมีการบังคับให้ใส่เฮดเดอร์พิเศษเพื่อบรรยายรายละเอียดต่างๆ ของค่าที่อยู่ในเมสเสจบอดีนั้นอีกด้วยซึ่งโดยทั่วไป เฮดเดอร์ที่มักจะถูกเพิ่มเข้าไปก็คือ Content-Type และ Content-Length การ ใช้เมธอด POST ที่มักพบเห็นโดยทั่วไปคือการ ใช้ POST ในการส่งค่าที่มาจากฟอร์ม HTML ยกตัวอย่างเช่น

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

หลังจากที่เราทำการคลิกที่ปุ่ม GO! แล้วเว็บเซิร์ฟเวอร์จะทำการส่งเมสเสจไปที่เซิร์ฟเวอร์ ซึ่งจะคล้ายๆ กับเมสเสจข้างล่างนี้

```
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=bababa&password=wow

สังเกตว่าในกรณีของการส่งค่าจากฟอร์ม HTML ไคลเอนต์จะทำการกำหนด Content-Type เป็น application/x-www-form-urlencoded และ Content-Length จะมีค่าเท่ากับความยาวของชื่อและค่าของ อิลิเมนต์ฟอร์มที่อยู่ในฟอร์ม HTML รวมกันซึ่งจากตัวอย่างของเราคือฟอร์มอิลิเมนต์ที่ชื่อ userName และ password

นอกจากนี้ POST ยังสามารถใช้กับการส่งค่าแบบอื่นๆได้อีก โดยรูปแบบจะขึ้นอยู่กับวิธีการส่งที่ทางเว็บเบราว์เซอร์ และเว็บเซิร์ฟเวอร์ได้ทำการตกลงกันไว้ ยกตัวอย่างเช่น ในกรณีของไฟล์ Upload ตัว Content-Type ก็จะถูกกำหนดเป็น Multipart/form-data แทน ซึ่งตัวอย่างของฟอร์ม HTML อยู่ข้างล่างนี้

```
<FORM ACTION="/upload/uploadServlet" EncType="multipart/form-data"
METHOD="POST">
Which file do you want to upload? <INPUT TYPE="file" NAME="yourfile"><br>
<INPUT TYPE="submit">
</FORM>
```

2.3 ภาษามาร์กอัพ (Markup Language)

เป็นภาษาที่ใช้ในการเขียนไฟล์เอกสารต่างๆ เพื่อแสดงผลให้ผู้ดู

2.3.1 HTML (HyperText Markup Language)

HTML เป็นส่วนหนึ่งของภาษา SGML (Standard Generalized Markup Language) ที่มีโครงสร้างการใช้งานแบบง่ายๆ โดยภาษา HTML มีไว้สำหรับใช้สร้างเอกสารแบบไฮเปอร์เท็กซ์ของรูปแบบของเอกสารหรือข้อมูลส่วนต่างๆ ของเอกสารที่สนับสนุนทั้งที่เป็นข้อความ, ภาพ, เสียง, วิดีโอ และอื่นๆ โดยการใช้งานนั้นสามารถเชื่อมโยงเอกสารหรือข้อมูลต่างๆนี้ได้ โดยจุดที่แสดงการเชื่อมโยงนี้เองที่เรียกว่าการมาร์กอัพ (Markup)

HTML เป็น Script ที่เรียงลำดับแท็ก(Tag) ซึ่งก็คือคำสั่งไว้เป็นไฟล์ไฟล์หนึ่งเพื่อเป็นตัวบอกเบราว์เซอร์ว่าจะต้องแสดงอะไร ลักษณะอย่างไร แท็กจะประกอบด้วยเครื่องหมาย '<' ตามด้วยชื่อแท็ก และเครื่องหมาย '>' ซึ่งโดยทั่วไปแท็กจะมีเป็นคู่เพื่อเปิดและปิดคำสั่ง โดยแท็กเปิดจะมีลักษณะดังข้างต้น แต่แท็กปิดจะเพิ่ม Slash (/) หน้าแท็กเท่านั้น ตัวอย่างเช่น คำสั่ง Heading

```
<HEADING>.....</HEADING>
```

แท็กจะพิมพ์ด้วยตัวอักษรใหญ่ หรือเล็กก็ได้

โครงสร้างของ HTML

ลำดับแท็กในเอกสาร HTML

```
<HTML>.....</HTML>
```


คือจุดเริ่มต้นและสิ้นสุดของโปรแกรม เพื่อแจ้งแก่เบราว์เซอร์ว่าเป็นเอกสาร HTML

<HEAD>.....</HEAD>

หัวเรื่องแนะนำเรื่องราวของโฮมเพจ

ส่วนภายใน <HEAD>

<TITLE>.....</TITLE>

ข้อความจะปรากฏเป็นชื่อวินโดว์

ส่วนข้อความอื่น ๆ

<BODY>.....</BODY>

เนื้อความและแท็กอื่นๆจะอยู่ภายในนี้

HTML เป็น Script ที่ใช้สร้างโฮมเพจบนเว็บซึ่งเป็น Script ที่มีความสามารถในการเชื่อมโยงข้อมูลในคอมพิวเตอร์ ระหว่างคอมพิวเตอร์ในเครือข่าย และระหว่างเครือข่ายในอินเทอร์เน็ต โดยอ้างอิงจาก URL (Uniform Resource Locators) ด้วยโปรโตคอล HTTP ซึ่งเป็นโปรโตคอลของ www ดังที่ได้กล่าวไว้ข้างต้น

2.3.2 WML (Wireless Markup Language)

WML เป็นรูปแบบหนึ่งของภาษา XML ที่ถูกออกแบบเพื่อให้เหมาะสมกับข้อจำกัดทางด้านฮาร์ดแวร์ และแบนด์วิดท์(Bandwidth) ที่แคบของอุปกรณ์ไร้สาย เช่น โทรศัพท์มือถือ ซึ่งจำกัดความสามารถในการอินพุตและเอาต์พุต โดยเป็นภาษาที่เขียนในรูปแบบแท็ก (tag-based) เช่นเดียวกับภาษา HTML รวมทั้งโครงสร้างการเขียนและแท็กบางอย่างจึงคล้ายกับ HTML ด้วยข้อจำกัดของการเขียนแท็ก WML ก็จะเขียนด้วยตัวพิมพ์เล็กทั้งหมด และต้องเขียนให้ถูกต้องตามโครงสร้างที่กำหนดไว้ จะเขียนแท็กขาดหรือเกินเหมือน HTML ไม่ได้

โครงสร้างของเอกสาร WML ถูกออกแบบมาให้มีการทำงานเป็นการ์ด(card)ซึ่งเปรียบได้กับเว็บเพจหนึ่งหน้า และเมื่อรวมการ์ดหรือเว็บเพจหลายหน้าเข้าด้วยกันจะเรียกว่า เด็ค(deck) ซึ่งก็คือเว็บไซต่นั้นเอง ซึ่งการที่ WML ถูกออกแบบให้มีลักษณะหลายๆการ์ดในหนึ่งเด็ค เพราะอุปกรณ์ WAP จะได้ไม่เสียเวลาในการส่งคำร้องขอไปยังเซิร์ฟเวอร์หลายครั้ง ซึ่งเป็นการลดเวลา และแบนด์วิดท์ลง

ดังนั้น โครงสร้างที่ใหญ่ที่สุดของเอกสาร WML คือ เด็ค ซึ่งก็คือเนื้อหาทั้งหมดที่อยู่ในเอกสาร WML ในแต่ละเด็คจะประกอบด้วยกลุ่มเอกสารย่อยๆ ซึ่งเรียกว่าการ์ด ในแต่ละการ์ดจะเป็นที่ที่เก็บข้อมูลต่างๆที่ต้องการให้แสดงที่หน้าจอรวมไปถึงคำสั่งต่างๆที่ต้องการให้เอกสาร WML ตอบสนองต่อความต้องการของผู้ใช้ อุปกรณ์เว็บจะทำการแสดงผลทีละการ์ด โดยการ์ดที่อยู่บนสุดจะถูกแสดงก่อน ส่วนการ์ดต่อไปจะแสดงก็ต่อเมื่อถูกเรียกดูเท่านั้น ซึ่งขึ้นกับว่าเลือกเมนูหรือลิงค์อย่างไร ในการ์ดแต่

ละใบจะมีข้อมูลอยู่ในตัวมันเอง ซึ่งข้อมูลเหล่านั้นสามารถเป็นได้ทั้งข้อมูลสำหรับอ่าน หรือเป็นจอให้ผู้ให้กรอกข้อมูล หรือเมนูหรือลิงค์ให้เลือก ซึ่งจริงๆ แล้วก็คือส่วนติดต่อกับผู้ใช้ (user interface) บนจอให้อุปกรณ์เว็บนั่นเอง ผู้ให้บริการในระบบเว็บจะทำการส่งข้อมูลที่ละหนึ่งเค็คเท่านั้น ดังนั้นจึงจำเป็นอย่างยิ่งที่จะต้องออกแบบเค็คและการค์ให้ดี ดังนั้นจึงไม่ควรเขียน WML ให้แต่ละการค์มีขนาดใหญ่มากเกินไป คือให้แต่ละการค์สามารถถูกแสดงได้บนหน้าจอได้เพียงหน้าจอดีวเท่านั้น ถ้าไม่สามารถทำได้ก็ควรให้อยู่บนการค์ใหม่ ทั้งนี้เพื่อความสะดวกต่อการใช้งาน และถ้าขนาดของเค็คมีขนาดใหญ่มากเกินไปจะทำให้ใช้เวลาในการโหลดเค็คมาอยู่บนหน่วยความจำของอุปกรณ์นั้น และถ้าเค็คมีขนาดใหญ่มากกว่าหน่วยความจำของอุปกรณ์นั้นจะไม่สามารถโหลดเค็คได้ แต่ถ้าเล็กเกินไปหรือจะต้องเรียกเค็คอื่นๆ ก็จะทำให้ต้องมีการรับ-ส่งข้อมูลบ่อยขึ้นทำให้การทำงานไม่ต่อเนื่องและรวดเร็วเท่าที่ควร

2.3.3 XML

XML ย่อมาจาก Extensible Markup Language เป็นภาษาที่กำหนดโดยองค์กร World Wide Web Consortium (W3C, at www.w3c.org) โดยบอค์ของตัวเองนี้ได้กำหนดให้เป็นมาตรฐานตามเว็บ โดยมีความสามารถในการนิยามภาษาอื่นได้ อันที่จริง XML เป็น meta-markup language เพราะเราต้องสร้างภาษามาร์กอัปด้วยตนเองอีกที และ WML(Wireless Markup Language) ที่ใช้ในการแสดงข้อความบนโทรศัพท์มือถือระบบเว็บอย่างแพร่หลายนี้ ก็มีต้นกำเนิดมาจากภาษานี้ด้วยเช่นกัน

ความสัมพันธ์ที่แท้จริงระหว่าง HTML และ XML ก็คือ ทั้งสองภาษานี้มีพื้นฐานมาจาก Standard Generalized Markup Language (SGML) ซึ่งภาษา SGML เป็นภาษามาร์กอัปทั่วไปที่มีประสิทธิภาพมาก เพราะว่สิ่งต่างๆ จำนวนมากสามารถทำได้ด้วย SGML อย่างไรก็ตามภาษานี้ก็ยังมีควมยากที่จะเรียนรู้ และไม่มีคนนำมาใช้โดยทั่วไป ดังนั้น XML จึงเป็นส่วนย่อยของ SGML ที่ง่ายต่อการใช้มากกว่า (ในทางเทคนิค HTML ถูกเรียกเป็นแอปพลิเคชันของ SGML) ถ้าต้องการความสัมพันธ์ของ SGML และ XML ให้มากกว่านี้สามารถไปที่ www.w3c.org/TR/NOTE-sgml-xml

ภาษามาร์กอัปที่คนส่วนใหญ่คุ้นเคยทุกวันนี้คือ ภาษา HTML แต่ภาษานี้ไม่มีศักยภาพเพียงพอในการจัดการเว็บเพจ และมีจำนวนแท็กไม่เพียงพอ ดังนั้น จึงมีความจำเป็นที่จะต้องใช้ XML ซึ่งสามารถสร้างภาษามาร์กอัปได้ด้วยตนเอง

การแสดงผล XML ในบราวเซอร์

บราวเซอร์ เช่น Microsoft Internet Explorer version 5 หรือ เวอร์ชันหลังจากนี้ และ Netscape Navigator 6 สามารถแสดง XML ได้โดยตรง แต่จะแสดงในลักษณะของซอร์สโค้ดของ XML ซึ่งสิ่งที่แสดงออกมานั้นเหมือนเป็นภาษากลางที่ใครๆ ก็อ่านได้ และจะมีการนำสิ่งนั้นไปทำงานต่ออีกทีหนึ่ง

รูปแบบในการกำหนดเอกสาร XML มีอยู่ 2 แนวทางหลัก คือ

1. ใช้กับ cascading style sheet (CSS)

การนำ CSS style sheet ไปใช้ต้องใช้ผ่านคำสั่งการปฏิบัติการของ XML โดยแทรก `<?xml-stylesheet type="text/css href="greeting.css"?>` ลงในไฟล์ XML เพื่อบอกบราวเซอร์ถึงรูปแบบของสไตล์ชีต ตัวอย่างนี้ใช้ CSS ที่ชื่อว่า greeting.css

2. ใช้กับ Extensible Style Sheets Language (XSL)

การนำ XSL ไปใช้ต้องมีไฟล์ XSL เป็นตัวจัดการ และต้องใช้ผ่านคำสั่งการปฏิบัติการของ XML โดยแทรก `<?xml-stylesheet type="text/xsl href="greeting.xsl"?>` ลงในไฟล์ XML เหมือนกัน

บราวเซอร์ของ XML อีกตัวที่นิยมใช้ คือ Jumbo โดยบราวเซอร์ตัวนี้ถูกออกแบบมาเพื่อใช้งาน XML และ CML สามารถดาวน์โหลดบราวเซอร์นี้ได้ฟรีที่ www.xml-cml.org/jumbo.html ซึ่งบราวเซอร์ตัวนี้ไม่เพียงแต่แสดงผล XML ได้ แต่ยังแสดงผล CML ได้อีกด้วย

ใน IE5.0 สามารถใช้ความสามารถของ XML ได้ดังนี้

- ดูเอกสาร XML
- สนับสนุนมาตรฐาน W3C DTD อย่างเต็มรูปแบบ
- สามารถใช้ XML ที่ใส่มาใน HTML ได้
- สามารถรวมข้อมูล XML เข้ากับแท็กของ HTML ได้
- สามารถจัดรูปแบบข้อมูล XML โดยใช้ XSL ได้
- สามารถจัดรูปแบบข้อมูล XML โดยใช้ CSS ได้
- สนับสนุนการใช้งาน CSS ได้
- สามารถเข้าถึงข้อมูล XML โดยใช้ DOM ได้

โดยสรุปคือเราสามารถกล่าวได้ว่า Internet Explorer เวอร์ชัน 5 ขึ้นไปมีคุณสมบัติในการเป็น parser XML เต็มรูปแบบ

สาเหตุที่ภาษา XML นิยมใช้

ภาษา XML เป็นที่นิยมใช้กันอย่างแพร่หลายเนื่องมาจาก XML ง่ายต่อการควบคุมและแลกเปลี่ยนข้อมูล เช่น ใน B2B ก็มีการใช้ XML ในการติดต่อกัน เหตุผลในการนิยมใช้ XML มีดังนี้

- ความง่ายในการแลกเปลี่ยนข้อมูล

XML มีความสามารถในการแลกเปลี่ยนข้อมูลระหว่างองค์กร ที่มีระบบที่ต่างกัน โดยผ่านอินเทอร์เน็ต โดยไม่ต้องใช้ความพยายามมาก ซึ่งระบบที่ต่างกันนี้จะแปลงผลออกมา เพื่อส่งออกให้อยู่ในรูปของ XML ก่อน โดย XML ทำหน้าที่เหมือนภาษากลางที่ทุกคนรู้จัก

- สามารถทำภาษามาร์กอัป ด้วยภาษาที่เขยชิน

เราสามารถสร้างภาษามาร์กอัปด้วยคำที่เราใช้ในชีวิตประจำวัน หรือ คำในทางธุรกิจ หรือ ใน Technical term ได้ โดยเราและบุคคลอื่นๆ ก็สามารถเข้าใจความหมายได้ และภาษาที่นิยมนำมาใช้ใน XML จนเริ่มที่จะเป็นสากลมีดังนี้

- Banking Industry Technology Secretariat (BITS)
 - Financial Exchange (IFX)
 - Bank internet Payment System (BIPS)
 - Telecommunications Markup (TIM)
 - School Interoperability Framework (SIF)
 - Common Business Library (CBL)
 - Electronic Business XML Initiative (ebXML)
 - Product Data Markup Language (PDML)
 - Financial Information eXchange protocol (FIX)
 - The Text Encoding Initiative (TEI)
- มีการอธิบายข้อมูลด้วยตัวเอง
มีการตั้งชื่ออิลิเมนต์ และชื่อแท็กของ XML อย่างมีความหมายด้วยภาษาของเราได้ ทำให้รู้ว่าข้อมูลที่อยู่ในแท็กนี้เป็นอย่างไร ซึ่งก็หมายความว่าเราสามารถรู้ได้ทันทีว่าเอกสาร XML นี้เป็นอะไร
 - มีโครงสร้างและการรวมข้อมูล
ความสามารถอีกอย่างของ XML ไม่ได้คือการระบุข้อมูลได้เท่านั้น แต่ยังระบุโครงสร้างของข้อมูลและอิลิเมนต์ที่แตกต่างกันก็สามารถรวมกับอิลิเมนต์อื่นได้อีกด้วย ความสำคัญอยู่ตรงที่การจัดการความซับซ้อนในการเกี่ยวข้องกับข้อมูลสำคัญ โดยเราจะสร้าง semantic rules ที่ระบุโครงสร้างของเอกสาร เพื่อที่เอกสารนั้นจะถูกตรวจสอบได้อย่างถูกต้อง

กฎเกณฑ์และการใช้ XML

ในบรรทัดแรกของไฟล์ XML ต้องมีการประกาศเฮดเดอร์ก่อน เช่น `<?xml version="1.0" encoding="windows874"?">` เป็นการประกาศว่าเป็นเอกสาร XML เวอร์ชันที่ 1

การใช้อิลิเมนต์และแท็ก

1. เมื่อมีแท็กเปิดต้องมีแท็กปิดเสมอ ซึ่ง XML นั้นเข้มงวดกว่า HTML มาก
2. ในแท็กของ XML นั้นอักษรตัวใหญ่และเล็กนั้นมีความแตกต่างกัน (Case Sensitive)
3. ตำแหน่งของแท็กมีความเคร่งครัดมาก โดยจะต้องเป็นตามลำดับ และต้องเปิดและปิดแท็กเป็นวงๆ ไป
4. ใน XML เรื่องอิลิเมนต์ต้องมีการซ้อนกันเป็นวงๆ ฉะนั้นจึงจำเป็นต้องมี Root Tag หรือ แท็กกราก หรือ แท็กแม่ โดยมีแท็กลูกอื่นๆ อยู่ในวงของแท็กแม่นี้ ถ้ามีแท็กลูก แท็กหลานก็ต้องอยู่เป็นวงๆ ภายในกรอบของแท็กลูก จะไม่มีการถิกวงเหมือนที่เคยทำได้ใน HTML

การใช้แอตทริบิวต์ของอิลิเมนต์

อิลิเมนต์ของ XML ก็สามารมีแอตทริบิวต์(Attribute) ได้เช่นเดียวกับที่เป็นมาใน HTML แต่แอตทริบิวต์ของ XML จะต้องอยู่ภายในกรอบของเครื่องหมายฟันทนุ (") เสมอ

XML Parser

XML parser เป็นซอฟต์แวร์แพ็คเกจที่เราใช้เป็นส่วนหนึ่งของแอปพลิเคชัน เช่น Oracle 8i หรือส่วนของโปรแกรมของเรา ซึ่งตัวพาร์สเซอร์อื่นๆ มีดังต่อไปนี้

- SAX: The simple API for XML
- expat
- expat as Perl Module
- TclExpat
- LT XML
- XML for Java –(XML4J)
- XML Microsoft's validating XML processor
- Lark
- XP
- Python and XML Processing Preliminary XML Parser
- TclXML
- XML Testbed
- SXP
- The Microsoft XML Parser
- OmiMark 5 Programming Language
- Java Standard Extension for XML

XML Validator

XML Validator เป็นตัวที่ทำการตรวจสอบว่าเอกสาร XML อยู่ในรูปแบบที่ถูกต้องตามกฎ (well-formed) แล้วหรือยัง โดยมีตัว XML validator ที่อยู่ในเว็บ ดังนี้

- W3C XML Validator ที่ validator.w3.org
- Tidy ที่ www.w3.org/People/Raggett/tidy/
- www.xml.com/xml/pub/tools/ruwf/check.html โดย XML validator นี้ต้องตรวจสอบ online เท่านั้น
- www.ltg.ed.ac.uk/~richard/xml-check.html XML validator ตัวนี้มีพื้นฐานมาจาก RXP parser และเอกสาร XML จะต้องตรวจสอบ online เช่นกัน

- www.stg.brown.edu/service/xmlvalid/ และเอกสาร XML จะต้องตรวจสอบ online เช่นกัน

2.3.4 XHTML

XHTML คือ ตระกูลของแบบเอกสาร (document type) ทั้งในปัจจุบันและอนาคตและเป็นโมดูลที่จำลอง หรือ subset ที่ต่อจาก HTML4 แบบเอกสารของ XHTML มีพื้นฐานมาจาก XML และถูกออกแบบการทำงานร่วมกับ User agent ที่ใช้พื้นฐานของ XML

XHTML 1.0 เป็นแบบเอกสารตัวแรกในตระกูลของ XHTML นับเป็นการปฏิรูปของ HTML4 ซึ่งเป็นแอปพลิเคชันของ XML 1.0 โดยถูกใช้เป็นภาษาในคอนเท้นต์ของการทำให้เหมือนกับ XML และรูปแบบภาษาที่ง่ายจะยังเหมือนกับ HTML4 อยู่แต่สำหรับผู้ที่ต้องการเขียน XHTML1.0 จะต้องทราบข้อได้เปรียบต่างๆ ดังนี้

- เอกสาร XHTML จะเหมือนกับ XML อย่างเช่น สามารถแสดง, แก้ไข , หาข้อผิดพลาด อย่างง่ายๆ ด้วยการใช้ XML tool ที่เป็นมาตรฐาน
- เอกสาร XHTML มีการเขียนและการทำงานที่เหมือนหรือดีกว่า HTML4
- เอกสาร XHTML มีประโยชน์มากในการเขียนเป็นแอปพลิเคชัน (เช่น scripts และ applets) ซึ่ง การตอบรับ (reply) ขึ้นอยู่กับ HTML DOM (HTML Document Object Model) หรือ XML DOM (XML Document Object Model)
- การเริ่มต้นของ XHTML จะเป็นการทำงานระหว่างกันภายในและระหว่างสภาพแวดล้อมของ XHTML ที่ต่างกัน

XHTML เป็นก้าวถัดไปของการวิวัฒนาการของอินเทอร์เน็ต สำหรับการเปลี่ยนมาใช้ XHTML ทุกวันนี้ทางด้านคอนเท้นต์จะมีความเป็น XML โดยรวมข้อได้เปรียบดังที่กล่าวมาแล้ว แต่ขณะเดียวกันก็ยังคงคอนเท้นต์เดิมอยู่ และ สามารถจะใช้ได้ต่อไปในอนาคต

ข้อแตกต่างของ XHTML กับ HTML4

เนื่องจาก XHTML เป็นแอปพลิเคชันของ XML ดังนั้นจึงต้องมีการเปลี่ยนกฎของ SGML ที่ เป็น พื้นฐานของ HTML4 ดังนี้

1. เอกสารต้อง well-formed (Document must be well-formed)

Well-formalness เป็นแนวคิดใหม่ของ XML ซึ่งหมายถึง ทุกๆอิลิเมนต์ต้องมีแท็กปิด หรือเขียนในรูปแบบพิเศษ และ ทุกๆอิลิเมนต์ต้องซ้อนกันด้วยคู่ของมันให้ถูกต้อง

ถูกต้อง: nested elements.

```
<p>here is an emphasized <em>paragraph</em>.</p>
```

ไม่ถูกต้อง: overlapping elements

```
<p>here is an emphasized <em>paragraph.</p></em>
```


2. ชื่อของอิลิเมนต์และแอตทริบิวต์ ต้องเป็นตัวเล็ก
เอกสาร XHTML ต้องเป็นตัวอักษรตัวเล็ก สำหรับ ทุกๆชื่อของอิลิเมนต์และแอตทริบิวต์นี้เป็นข้อแตกต่างที่สำคัญมากเพราะ XML นั้น case sensitive เช่น กับ เป็น แท็กที่ต่างกัน
3. สำหรับแท็กที่ไม่ใช่แท็กว่าง (empty tag) ต้องมีแท็กปิด
ใน HTML4 อนุญาตให้ไม่มีแท็กปิดได้โดยอิลิเมนต์ที่ตามมาจะถือว่าเป็นการปิดแท็ก การยกเว้นไม่ต้องปิดแท็กนี้ไม่อนุญาตให้มีใน XHTML
4. ค่าของแอตทริบิวต์ต้องใส่เครื่องหมายฟั่นหนู (Double quote)
ค่าของแอตทริบิวต์ทุกๆค่าจะต้องใส่เครื่องหมายฟั่นหนู (") ถึงแม้ว่าค่านั้นจะเป็นหมายเลขก็ตาม
ถูกต้อง: terminated elements

```
<p>here is a paragraph.</p><p>here is another paragraph.</p>
```

ไม่ถูกต้อง: unterminated elements

```
<p>here is a paragraph.<p>here is another paragraph
```
5. การลดแอตทริบิวต์
XML ไม่สนับสนุนการมีแอตทริบิวต์จำนวนน้อยที่สุด ค่าของแอตทริบิวต์ทั้งคู่จะต้องมีการเขียนให้อยู่ในรูปแบบแอตทริบิวต์เช่น checked และ compact ไม่สามารถปรากฏได้หากไม่มีการกำหนดค่า
ถูกต้อง: unminimized attributes

```
<dl compact="compact">
```

ไม่ถูกต้อง: minimized attributes

```
<dl compact>
```
6. อิลิเมนต์ว่าง (empty element)
อิลิเมนต์ว่างต้องมีแท็กปิดหรือต้องมี / ปิดท้าย อย่างใดอย่างหนึ่ง ตัวอย่างเช่น
 หรือ <hr/>
ถูกต้อง: terminated empty tags

```
<br/><hr/>
```

ไม่ถูกต้อง: unterminated empty tags

```
<br><hr>
```
7. การจัดการ Whitespace อยู่ในค่าของแอตทริบิวต์
ในค่าของแอตทริบิวต์, User agents จะกำจัด Whitespace ส่วนหน้าและส่วนหลัง จากค่าของแอตทริบิวต์และ map ลำดับ Whitespace characters 1 character หรือมากกว่า (รวมถึง line breaks) ไปเป็น Inter-word space เดียวๆ (an ASCII space character for western scripts)

8. สคริปต์และอิลิเมนต์ Style

ใน XHTML สคริปต์และอิลิเมนต์ Style จะถูกประกาศให้มี #PCDATA รวมอยู่ด้วย < และ & จะถูกทำให้เป็นตัวเริ่มต้นของ markup และเอนติตี้ เช่น < และ & จะถูกยอมรับให้เป็นเอนติตี้อ้างอิงโดยตัวประมวลผล XML ไปเป็น < และ & ตามลำดับ การครอบเนื้อหาของสคริปต์หรืออิลิเมนต์ style ภายในส่วน CDATA เพื่อหลีกเลี่ยง การขยายเอนติตี้

```
<script>
```

```
<![CDATA[
```

```
... unescaped script content ...
```

```
]]>
```

```
</script>
```

ส่วนของ CDATA จะถูกยอมรับโดย XML processor และ ปรากฏเป็น node ใน Document Object Model

9. SGML exclusions

SGML นั้นไม่ยอมให้ผู้เขียน DTD บรรจ้อิลิเมนต์ที่ระบุโดยเฉพาะ ไว้ภายในอิลิเมนต์การห้าม (หรือที่ถูกเรียกว่า “exclusions”) เป็นไปไม่ได้ใน XML ตัวอย่างเช่น HTML4 Strict DTD ไม่อนุญาตให้ซ้อนอิลิเมนต์ ‘a’ ไว้ภายในอิลิเมนต์ ‘a’ อีกอันหนึ่ง ดังนั้นจึงเป็นไปไม่ได้ที่จะไม่อนุญาตใน XML ถึงแม้ว่าการป้องกันนี้ไม่สามารถกำหนดได้ใน DTD อิลิเมนต์ที่เฉพาะเจาะจงนี้ไม่ควรจะมีการซ้อนกัน โดยอิลิเมนต์ที่ไม่อนุญาตให้มีการซ้อนกันมีดังนี้

แท็ก <a>

ไม่สามารถมีอิลิเมนต์ a ไว้อย่างในได้

แท็ก <pre>

ไม่สามารถมีอิลิเมนต์ img, object, big, small, sub, or sup ไว้อย่างในได้

แท็ก <button>

ไม่สามารถมีอิลิเมนต์ input, select, textarea, label, button, form, fieldset, iframe or isindex ไว้อย่างในได้

แท็ก <label>

ไม่สามารถมีอิลิเมนต์ label อื่นๆ ไว้อย่างในได้

แท็ก <form>

ไม่สามารถมีอิลิเมนต์ form อื่นๆ ไว้อย่างในได้

10. อิลิเมนต์ที่มี ‘id’ และ ‘name’ เป็น attributes

HTML 4 กำหนดให้อิลิเมนต์ a, applet, form, frame, iframe, img, และ map มี name เป็นแอตทริบิวต์และ HTML 4 ยังให้มี id เป็นแอตทริบิวต์ได้อีกด้วย โดยแอตทริบิวต์ทั้งคู่ถูกออกแบบมาเพื่อใช้ส่วนหนึ่งของ identifier ใน XML นั้น ส่วนหนึ่งของ identifier เป็นของ ID type และมีแอตทริบิวต์ id ได้เพียงอันเดียวต่อ 1 อิลิเมนต์ ดังนั้น ในแอตทริบิวต์ id ของ

XHTML1.0 จะถูกกำหนดเป็น type ID เพื่อที่จะทำให้แน่ใจว่า XHTML 1.0 มีโครงสร้างแบบ XML โดยสมบูรณ์ XHTML1.0 ต้องใช้แอตทริบิวต์ id ในการกำหนดให้เป็นส่วนหนึ่งของ identifier แม้ว่าจะมี name เป็นแอตทริบิวต์อยู่แล้ว ในอิลิเมนต์ของ XHTML 1.0 ดังที่กล่าวมาแล้วนี้ ที่มี name เป็นแอตทริบิวต์จะถูกกำหนดให้เลิกใช้ (deprecated) และ จะเอาออกใน XHTML เวอร์ชันถัดไป

แท็ก XHTML (เฉพาะที่แตกต่างจาก HTML)

<area />
 <base />

 <col> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <col />
 <frame> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <frame />
 <hr />

 <input> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <input />
 <link> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <link />
 <meta> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <meta />
 <param> มีแท็กปิด หรือ ไม่มีก็ได้ ถ้าไม่มีแท็กปิดต้องประกาศเป็น <param />

แท็กที่มีใน HTML แต่ไม่มีใน XHTML

<applet> ใช้แท็ก <object> แทน
 <basefont> ใช้แท็ก styles แทน
 <center> ใช้แท็ก styles แทน
 <dir> ใช้แท็ก แทน
 ใช้แท็ก styles แทน
 <isindex> ใช้แท็ก <input> แทน
 <listing> ใช้แท็ก <pre> แทน
 <menu> ใช้แท็ก แทน
 <plaintext> ใช้แท็ก <pre> แทน
 <s> ใช้แท็ก แทน
 <strike> ใช้แท็ก แทน
 <u> ใช้แท็ก styles แทน
 <xmp> ใช้แท็ก <pre> แทน

2.4 เซิร์ฟเลต (Servlet)

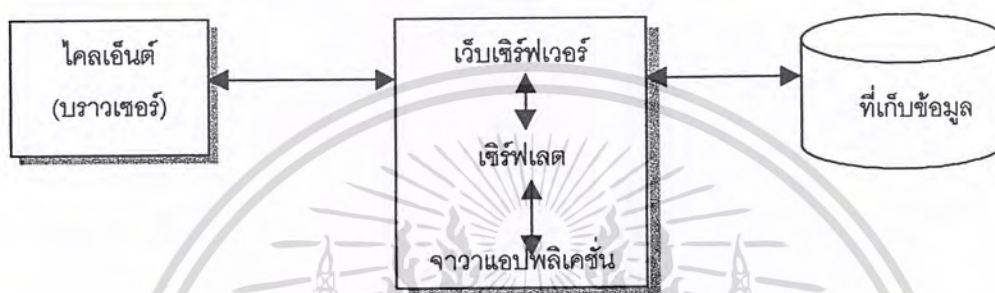
แม้ว่าโลกของอินเทอร์เน็ตจะเพิ่งเกิดขึ้นเพียงไม่กี่ปีก็ตาม เทคโนโลยีที่ใช้กับอินเทอร์เน็ตกลับมีการเปลี่ยนแปลงไปอย่างรวดเร็วมาก ในสมัยแรกๆ เพจต่างๆ ที่อยู่ในเว็บจะเป็นลักษณะของเพจสแตติกหรือเพจที่ไม่มีการเปลี่ยนแปลงเนื้อหาที่ไม่ว่าจะนานเท่าไร นอกเสียจากว่าผู้ดูแลเพจนั้นจะทำการอัปเดตเพจดังกล่าว เพจลักษณะนี้เป็นเพจที่นิยมใช้กันทั่วไปในอินเทอร์เน็ตสมัยแรกเพราะอินเทอร์เน็ตยังนิยมกันอยู่ในวงแคบ โดยกลุ่มผู้ใช้จะเป็นกลุ่มบุคคลที่อยู่ในวงการศึกษานั้น ต่อมาจากนั้นไม่นานทางผู้ผลิตบราวเซอร์ได้ทำการเพิ่มความสามารถให้กับเพจโดยอนุญาตให้เพจสามารถแทรกสคริปต์เล็กๆ ลงไปรวมกับส่วนที่เป็น HTML ได้ซึ่งจุดนี้ก็คือจุดเริ่มต้นของจาวาสคริปต์ทางฝั่งไคลเอนต์นั่นเอง

แม้ว่าเพจจะเริ่มมีความสามารถในการโต้ตอบกับผู้ใช้โดยอิงความสามารถจากจาวาสคริปต์แล้วก็ตาม ถ้าพูดถึงในแง่ของส่วนเนื้อหาของตัวเพจจริงๆ แล้วตัวเพจเองก็ยังคงเป็นเพจสแตติกอยู่เช่นเดิม เมื่อกลุ่มผู้ใช้อินเทอร์เน็ตเริ่มมีมากขึ้นความต้องการที่จะทำให้เพจสามารถทำการรับส่งข้อมูลรวมไปถึงการเปลี่ยนแปลงเนื้อหาได้โดยอัตโนมัติก็เกิดขึ้น เทคโนโลยีที่เกิดขึ้นเพื่อรองรับความต้องการเหล่านี้ก็คือแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ (Server Side Application) นั่นเอง

แอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ ในระยะแรกๆ มักถูกเขียนขึ้นด้วยคอนเซ็ปต์ของ CGI (Common Gateway Interface) โดยหลักการทำงานง่ายๆ ก็คือ เว็บเบราว์เซอร์จะทำการส่งข้อมูลที่เกิดจากแอ็กชันของผู้ใช้ เช่น การคลิกลิงก์หรือการกรอกแบบสอบถามไปยังเว็บเซิร์ฟเวอร์โดยแทนที่เว็บเซิร์ฟเวอร์จะทำการส่งเพจที่เป็นสแตติกกลับมา เว็บเซิร์ฟเวอร์จะทำการส่งต่อข้อมูลดังกล่าวไปยังโปรแกรมซึ่งถูกจัดไว้ โปรแกรมดังกล่าวจะทำการประมวลผลข้อมูลที่ได้แล้วส่งกลับไปยังเว็บเซิร์ฟเวอร์ ซึ่งทางเว็บเซิร์ฟเวอร์ก็จะส่งผลที่ได้นี้กลับไปยังเว็บเบราว์เซอร์อีกที หลายคนคงเคยเห็นแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ที่เป็นผลผลิตของ CGI ในรุ่นแรกๆ ที่ยังหลงเหลือในปัจจุบันนี้ ยกตัวอย่างเช่น ตัวนับ (Counter), แผนที่รูปภาพ (Image Map), สมุดเยี่ยม (Guest Book), ส่งเมล (SendMail) เป็นต้น จริงๆ แล้วตัว CGI เองสามารถเขียนด้วยภาษาอะไรก็ได้ แต่ที่นิยมที่สุดก็เห็นจะเป็น C และ Perl อาจจะเป็นเพราะว่า CGI เป็นสิ่งที่มีความกับอินเทอร์เน็ตช่วงแรกๆ ดังนั้นไม่ว่าจะเป็นเว็บเซิร์ฟเวอร์ไหนก็ตามแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์พื้นฐานที่ทางเว็บเซิร์ฟเวอร์เหล่านั้นจะต้องสนับสนุนก็คือ CGI เป็นที่นิยมใช้กันอย่างกว้างขวางจนกระทั่งปัจจุบันนี้

เซิร์ฟเลตเป็นแอปพลิเคชันที่อยู่ทางฝั่งเซิร์ฟเวอร์แบบหนึ่งซึ่งอ้างอิงคอนเซ็ปต์มาจาก CGI ข้อดีของเซิร์ฟเลตที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาวานั้นเอง จาวาเป็นภาษาที่ใช้คอนเซ็ปต์ของออบเจกต์-โอเรียนเตด ในการเขียนหลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมสำหรับโปรเจกต์ใหญ่ๆ จะทราบว่า ออบเจกต์-โอเรียนเตด สามารถลดความซับซ้อนของโครงสร้างโปรแกรมรวมไปถึงการอำนวยความสะดวกในการนำส่วนของโปรแกรมที่เขียนไว้แล้วมาใช้ใหม่ได้ดียิ่งขึ้น นอกจากนี้จาวายังเป็นภาษาที่เป็นลักษณะแบบไม่ขึ้นกับแพลตฟอร์มซึ่งจะช่วยให้เราสามารถที่จะทำการพัฒนาระบบโดยใช้สภาพแวดล้อมอะไรก็ได้ซึ่งโดยทั่วไปมักนิยมใช้ Window โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบนยูนิกซ์ เพื่อเพิ่มประสิทธิภาพของโปรแกรมแทน นอกจากนี้เซิร์ฟเลตยังมี

ความเร็วที่สูงกว่า CGI เพราะเซิร์ฟเลตใช้หลักการของเธรด (thread) โดยทำการสร้าง 1 เธรดต่อหนึ่งคำร้องขอที่มาจากไคลเอนต์ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 โพรเซส(process) ต่อหนึ่งคำร้องขอ ซึ่งจะทำให้เปลืองทรัพยากรมากกว่า และโพรเซสในการรันก็จะช้ากว่าด้วย ท้ายที่สุดจุดเด่นที่สำคัญของเซิร์ฟเลตก็คือ API(Application Programming Interface) โดยระบบที่ทำการพัฒนาโดยใช้คอนเซ็ปของเซิร์ฟเลตจะสามารถเรียกใช้ API ที่ทางจาวามีมาให้ (javax.servlet.*, javax.servlet.http.*) ซึ่งจะช่วยให้การพัฒนาระบบดังกล่าวง่ายและเร็วยิ่งขึ้น

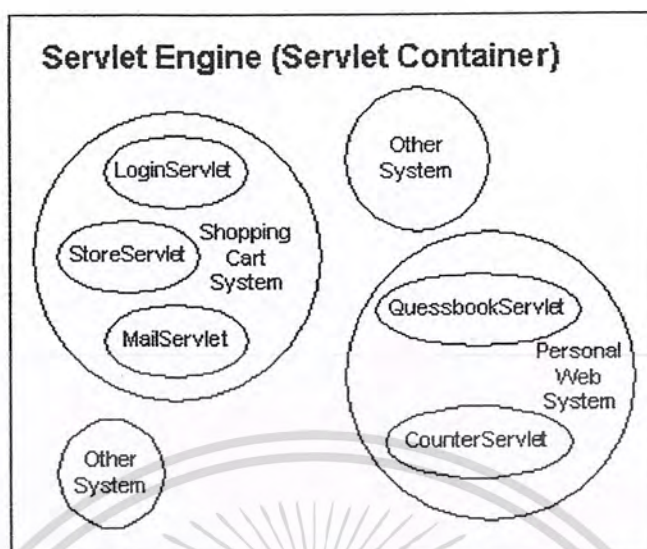


รูปที่ 2.7 หน้าที่และการทำงานของเซิร์ฟเลต

เมื่อการขอเรียกใช้บริการเซิร์ฟเลตมาถึงยังเซิร์ฟเวอร์ เซิร์ฟเลตเอนจินจะเรียกตัวโหลดคลาส (class loader) ตัวโหลดคลาสดำเนินการตรวจสอบว่าคลาสดังกล่าวถูกโหลดขึ้นมาเรียบร้อยแล้วหรือไม่ และตรวจสอบเวอร์ชันของคลาสที่ถูกโหลดว่ามีค่าไทม์แสตมป์ (time stamp) เท่ากับ .class ไฟล์ (class file) ที่อยู่บนดิสก์ (disk) หรือไม่ ถ้าคลาสของเซิร์ฟเลตยังไม่ถูกโหลด หรือมีเวอร์ชันเก่ากว่าไฟล์ที่อยู่บนดิสก์ เซิร์ฟเลตเอนจินจะทำการโหลดคลาสไปเก็บไว้ใน JVM และสร้างอินสแตนซ์ (instance) ของคลาสดังกล่าว หลังจากนั้นวงจรชีวิตของเซิร์ฟเลตจะเริ่มขึ้นใหม่

2.4.1 เซิร์ฟเลตเอนจิน (Servlet engine)

ในการรันระบบที่เขียนขึ้นโดยใช้หลักการของเซิร์ฟเลตเราจะต้องนำระบบดังกล่าวมาบรรจุอยู่ในสิ่งๆ หนึ่งที่เรียกว่าเซิร์ฟเลตเอนจิน ให้นึกว่าเซิร์ฟเลตเอนจินคล้ายๆ กับกล่องๆ หนึ่งที่ใส่ลูกปิงปองไว้หลายลูก โดยลูกปิงปองแต่ละลูกก็คือระบบๆ หนึ่งนั่นเอง หลายคนอาจสงสัยทำไมถึงใช้คำว่าระบบ โดยทั่วไปแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์ต่างๆ ที่ถูกเขียนขึ้นโดยใช้ API ของเซิร์ฟเลตจะถูกเรียกว่าเซิร์ฟเลต ในหนึ่งระบบอาจประกอบด้วยเซิร์ฟเลตหลายอัน ยกตัวอย่างเช่น ระบบที่เกี่ยวข้องกับ Shopping Cart อาจจะประกอบด้วยเซิร์ฟเลตที่ทำหน้าที่ในการเช็คสต็อกสินค้า, เซิร์ฟเลตที่ทำหน้าที่ในการเก็บข้อมูลสินค้า, เซิร์ฟเลตที่ทำหน้าที่ในการส่งอีเมลกลับไปยังลูกค้าเพื่อบอกว่าได้ทำการส่งของไปให้แล้ว เป็นต้น ดังนั้นถ้ามองโดยรวมแล้วเซิร์ฟเลตเอนจินก็คือที่รวมของระบบตั้งแต่หนึ่งระบบถึงหลายระบบ โดยแต่ละระบบจะประกอบด้วยเซิร์ฟเลตหนึ่งอันหรือมากกว่า ดังรูป ซึ่งระบบ ในที่นี้อาจจะหมายถึง Zone (Apache Jserv) หรือ Web Application (Tomcat) ก็ได้



รูปที่ 2.8 เซิร์ฟเลตเอนจิน และเซิร์ฟเลตที่อยู่ภายใน

เซิร์ฟเลตเอนจินเป็นเพียงกล่องๆหนึ่งที่ใช้บรรจุและรันกลุ่มของเซิร์ฟเลตเท่านั้น ในการที่จะทำการติดต่อสื่อสารกับไคลเอนต์ ตัวเซิร์ฟเลตเอนจินนี้จะต้องทำงานร่วมกับเว็บเซิร์ฟเวอร์ ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับไคลเอนต์อีกทีหนึ่ง เมื่อไคลเอนต์ที่มีคำร้องขอส่งมาจากไคลเอนต์ ถ้าคำร้องขอนั้นนั้นจะเจองมาที่ตัวเซิร์ฟเลต ทางเว็บเซิร์ฟเวอร์ก็จะทำการส่งคำร้องขอนั้นมาให้เซิร์ฟเลตเอนจิน ซึ่งทางเซิร์ฟเลตเอนจินก็จะทำการเรียกเซิร์ฟเลตที่ไคลเอนต์ต้องการขึ้นมาทำการประมวลผลคำร้องขอ นั้น โดยท้ายสุดเซิร์ฟเลตจะส่งผลกลับไปให้เซิร์ฟเลตเอนจิน เซิร์ฟเลตเอนจินก็จะส่งผลที่ได้กลับไปให้เว็บเซิร์ฟเวอร์ ซึ่งเว็บเซิร์ฟเวอร์ก็จะส่งผลกลับไปให้ไคลเอนต์

เซิร์ฟเลตเอนจินอาจจะเป็นส่วนที่ติดมากับเว็บเซิร์ฟเวอร์อยู่แล้วยกตัวอย่างเช่น เซิร์ฟเลตเอนจินที่อยู่ในเซิร์ฟเวอร์ของ Netscape Enterprise , IBM WebSphere หรืออาจจะเป็นส่วนที่เป็นส่วนที่เพิ่มเติมให้กับเว็บเซิร์ฟเวอร์ก็ได้เช่น Apache Jserv, Tomcat, JRun หรือแม้กระทั่งเป็นส่วนหนึ่งที่อยู่ในเว็บแอปพลิเคชัน เช่น BEA Weblogic เป็นต้น ทั้งนี้การเลือกใช้เซิร์ฟเลตเอนจินแต่ละชนิดก็มักขึ้นอยู่กับปัจจัยหลายอย่าง เช่น ความสะดวกในการรวมระบบที่จะสร้างขึ้นมาใหม่กับระบบที่มีอยู่แล้ว, งบประมาณที่มีอยู่สำหรับโครงการหรืออาจจะรวมไปถึงทักษะและประสบการณ์ส่วนตัวของนักพัฒนาแต่ละคน

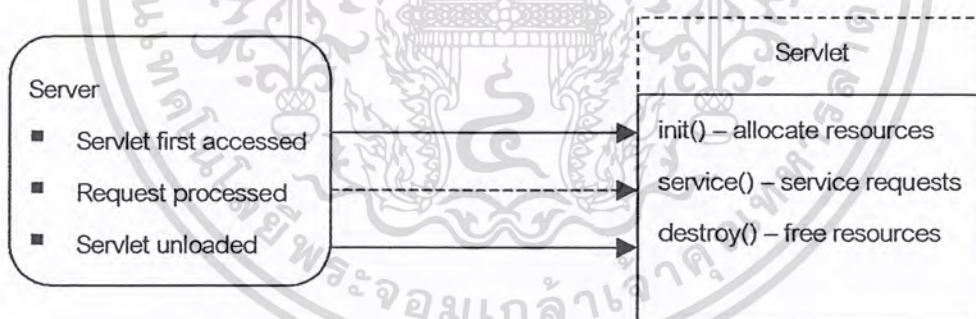
2.4.2 อินเตอร์เฟส javax.servlet.Servlet

หัวใจของเซิร์ฟเลตจริงๆอยู่ที่อินเตอร์เฟสที่ชื่อ javax.servlet.Servlet* โดยทุกเซิร์ฟเลตที่ถูกเขียนขึ้นจะต้องทำการอิมพลิเมนต์ตัวอินเตอร์เฟสนี้ไม่ว่าทางตรงก็ทางอ้อม (ทางตรงก็คือการอิมพลิเมนต์ตัวอินเตอร์เฟสนี้เลย ส่วนทางอ้อมก็คือการให้เซิร์ฟเลตทำการแบ่งย่อยคลาสบางคลาสที่ได้ทำการอิมพลิเมนต์ตัวอินเตอร์เฟสนี้ไว้แล้ว) เหตุผลว่าทำไมเราต้องอิมพลิเมนต์ตัวอินเตอร์เฟสนี้เพราะว่าเมื่อไรก็ตาม

ตามที่มีคำร้องขอจากไคลเอนต์เข้ามายังเซิร์ฟเลตเอนจิน ตัวเซิร์ฟเลตเอนจินจะทำการหาเซิร์ฟเลตที่ร้องขอดังกล่าวอ้างถึง หลังจากนั้นเซิร์ฟเลตเอนจินจะทำการเรียกฟังก์ชันต่างๆที่อยู่ในเซิร์ฟเลตเพื่อทำการประมวลผลคำร้องขอของไคลเอนต์ โดยฟังก์ชันที่เซิร์ฟเลตเอนจินจะทำการเรียกก็คือฟังก์ชันที่เซิร์ฟเลตได้ทำการอิมพลิเมนต์ ซึ่งเป็นฟังก์ชันที่ถูกกำหนดอยู่ใน `javax.servlet.Servlet` อินเตอร์เฟสนั่นเอง อาจมีคำถามว่าทำไมเซิร์ฟเลตเอนจินถึงเรียกฟังก์ชันที่ถูกกำหนดอยู่ในอินเตอร์เฟสนี้ เหตุผลง่าย ๆ ก็คือเซิร์ฟเลตเป็นส่วนที่ถูกโหลดเข้าไปในเซิร์ฟเลตเอนจินในช่วง Runtime ตัวเซิร์ฟเลตเอนจินเองไม่สามารถทราบได้ว่าเซิร์ฟเลตต่างๆมีฟังก์ชันอะไรประกอบอยู่ข้างนอกเสียจากว่าเซิร์ฟเลตนั้นได้ทำการอิมพลิเมนต์ฟังก์ชันที่เป็นมาตรฐานที่เซิร์ฟเลตเอนจินรับรู้ซึ่งก็คือเหตุผลว่าทำไมทุกเซิร์ฟเลตจะต้องทำการอิมพลิเมนต์ตัว `javax.servlet.Servlet` อินเตอร์เฟส อย่างไรก็ตามเราสามารถให้เซิร์ฟเลตเอนจินเรียกฟังก์ชันอื่นๆที่อยู่ในเซิร์ฟเลตได้ ซึ่งวิธีการก็คือการใส่ฟังก์ชันดังกล่าวเข้าไปในส่วนอิมพลิเมนต์ของฟังก์ชันต่างๆที่ถูกกำหนดอยู่ใน `javax.servletServlet` อินเตอร์เฟส

2.4.3 วงจรชีวิตของเซิร์ฟเลต (Servlet's Life Cycle)

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

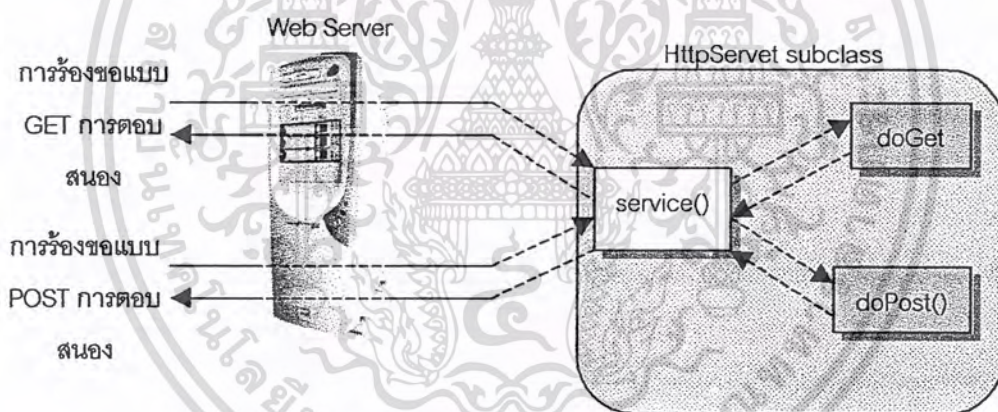


รูปที่ 2.9 วงจรชีวิตของเซิร์ฟเลต

1. สถานะเริ่มต้น เป็นการใช้เมธอด `init()` เมธอดจะถูกเรียกโดยเซิร์ฟเวอร์ทันทีที่อินสแตนซ์ (instant) ถูกสร้างเสร็จเรียบร้อยแล้ว โดยปกติเมธอดนี้จะถูกใช้เพื่อทำการเริ่มต้นเซิร์ฟเลตสาเหตุที่ไม่ใช้คอนสตรัคเตอร์ (constructor) เพราะคอนสตรัคเตอร์ของ JDK 1.0 (เริ่มมีเซิร์ฟเลต) ไม่สามารถรับอาร์กิวเมนต์ (argument) ในกรณีที่เกิดการโหลดแบบไดนามิก (dynamic) ได้ ดังนั้นการที่จะจัดหาข้อมูลเกี่ยวกับตัวมันเองและสิ่งแวดล้อมให้แก่เซิร์ฟเลตที่ถูกโหลดเป็นครั้งแรก เซิร์ฟเวอร์จำเป็นต้องเรียกเมธอด `init()` และส่งผ่านออบเจกต์ (object) ที่ใช้อินเตอร์เฟส `ServletConfig` นอกจากนี้ภาษาจาวาไม่อนุญาตให้อินเตอร์เฟสมีการประกาศคอนสตรัคเตอร์

นั้นหมายถึงอินเตอร์เฟซของแพ็คเกจ(package) javax.servlet.Servlet ก็ไม่สามารถประกาศคอนสตรัคเตอร์ที่รับค่าพารามิเตอร์(parameter) ServletConfig ได้เช่นกัน จึงต้องประกาศเมธอดชื่ออื่นแทนเช่น init() แต่ยังคงมีทางเป็นไปได้ที่จะประกาศคอนสตรัคเตอร์ให้แก่เซิร์ฟเลต แต่ในคอนสตรัคเตอร์ดังกล่าวห้ามมีการเข้าถึงออบเจกต์ ServletConfig และไม่สามารถ throw ServletException ได้ ออบเจกต์ ServletConfig จะส่งพารามิเตอร์ที่จำเป็นต้องใช้ในการเริ่มต้นการทำงานให้แก่เซิร์ฟเลต พารามิเตอร์ดังกล่าวจะถูกป้อนให้แก่เซิร์ฟเลต โดยไม่เกี่ยวข้องกับวิธีการร้องขอที่เข้ามา

- สถานะให้บริการ จะถูกเรียกจากเซิร์ฟเวอร์ทุกครั้งทีโคลเอนต์ทำการร้องขอใช้บริการเซิร์ฟเลต เมธอด service() จะรับค่าพารามิเตอร์ 2 ค่าได้แก่ออบเจกต์การร้องขอและออบเจกต์การตอบสนอง ออบเจกต์การร้องขอจะบอกเซิร์ฟเลตเกี่ยวกับการขอใช้บริการที่มีเข้ามา ในขณะที่ออบเจกต์การตอบสนองจะถูกเรียกใช้ในการส่งการตอบรับคืนกลับให้แก่เซิร์ฟเวอร์ โดยปกติแล้วเมธอด service() จะไม่ถูกโอเวอร์ไรด์(override) แต่จะทำการโอเวอร์ไรด์เมธอด doGet() เพื่อใช้จัดการการร้องขอแบบ GET และเมธอด doPost() สำหรับจัดการกับการร้องขอแบบ POST แทนดังตัวอย่างในรูป



รูปที่ 2.10 วิธีการจัดการกับร้องขอแบบ GET และ POST ของเซิร์ฟเลต

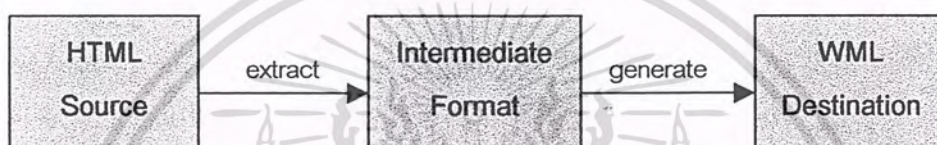
- สถานะทำลาย (destroy()) เซิร์ฟเวอร์จะเรียกเมธอด destroy() ของเซิร์ฟเลต เมื่อเซิร์ฟเลตกำลังจะถูกอันโหลด(unload) ในเมธอด destroy() ควรมีการคืนทรัพยากรต่างๆกลับคืนมาให้แก่ระบบ เพื่อให้โปรแกรมอื่นๆจะได้นำทรัพยากรเหล่านี้ไปใช้ได้ นอกจากนี้เมธอดยังให้โอกาสเซิร์ฟเลตในการบันทึกสถานะของข้อมูลต่างๆที่ถูกเรียกใช้เป็นประจำจากการเรียกเมธอด init() ในครั้งต่อไป

บทที่ 3

หลักการในการออกแบบ

3.1 การออกแบบและการทำงานของโปรแกรมแปลงภาษา 1

ตัวแปลงภาษาจะทำงานโดยการตัดแบ่งข้อความจากหน้า HTML ต้นฉบับและทำการจัดรูปแบบใหม่ให้เป็นภาษา WML โดยทำการแปลงให้อยู่ในรูปแบบของ Intermediate Format เสียก่อน ซึ่ง Intermediate Format ในที่นี้คือภาษา XML โดยมีการเพิ่มแท็กพิเศษบางแท็กเข้าไปเพื่อให้ง่ายต่อการแปลงเป็น WML ได้แก่ แท็ก <checkbox>, <radio> และ <text>



รูปที่ 3.1 ขั้นตอนการแปลงของโปรแกรมแปลงภาษา 1

ส่วนในการสร้างโค้ดภาษา WML จาก XML จะใช้เครื่องมือช่วยแปลง คือโปรแกรมโคคุนของอาปาเช่ (Apache's Cocoon) ซึ่งจะประกอบด้วย 3 ส่วน คือ

1. Procedure
เป็นตัวสร้างเอกสาร XML ซึ่งในการเขียนต้องสอดคล้องกับ API ของโปรคิวเซอร์โคคุน
2. Processor
เป็นตัวในการประมวลผลต้นไม้ (tree) DOM ซึ่งถูกสร้างขึ้นมาจากเอกสาร XML ที่สร้างโดยโปรคิวเซอร์
3. Formatter
เป็นตัวกำหนดรูปแบบของเอกสาร แต่เนื่องจาก XSL สไตลชีต ที่ระบุโดยตัวประมวลผล XSLT ทำงานที่จำเป็นทั้งหมดแล้ว ดังนั้นในกรณีนี้จึงไม่จำเป็นต้องใช้ส่วนนี้

3.2 การออกแบบและการทำงานของโปรแกรมแปลงภาษา 2

3.2.1 บทนำ

ตัวแปลภาษานี้เขียนขึ้นโดยใช้ภาษาจาวา ที่เลือกใช้ภาษาจาวาเนื่องจากภาษาจาวาสามารถสร้าง request เพื่อส่งไปยังเซิร์ฟเวอร์ และรับ response ที่ส่งมาจากเซิร์ฟเวอร์ได้ง่ายไม่ซับซ้อน อีกทั้งภาษาจาวามีเมธอด (Method) ในการจัดการกับสตริง (String) หลากหลายและง่ายต่อการใช้งาน

ขั้นตอนในการแปลงภาษาจากภาษา HTML ไปเป็นภาษา WML ของโปรแกรมนี้นี้จะเริ่มด้วยการให้ผู้กรอก URL ของเว็บไซต์ที่ต้องการจะแปลงจากนั้นโปรแกรมจะทำการส่ง request ไปยัง

เซิร์ฟเวอร์ เพื่อร้องขอเอกสาร HTML จากเซิร์ฟเวอร์นั้นๆ เมื่อโปรแกรมได้ไฟล์เอกสาร HTML มาแล้ว โปรแกรมจะทำการอ่านข้อมูลในไฟล์นั้นเก็บอยู่ในสตริงเพื่อง่ายต่อการอ้างถึงและทำการแปลงโดยใช้มรคคที่มีอยู่ของภาษาจาวาซึ่งทำงานบนอ็อบเจกต์สตริง และ โปรแกรมจะนำสตริงนี้ไปใช้ในการแปลงเป็นรูปแบบของภาษา WML ต่อไป

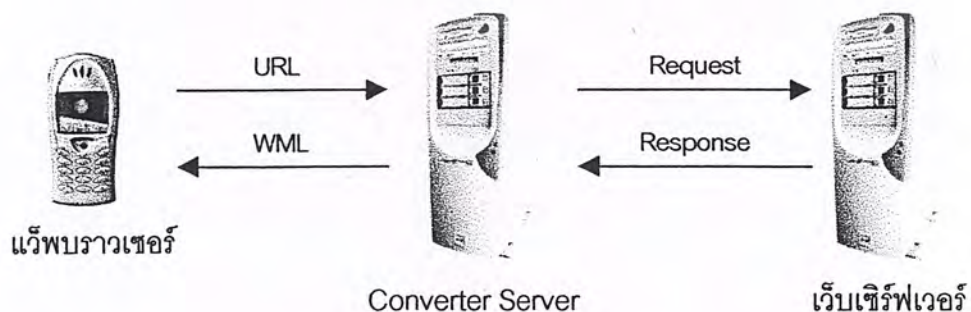
โปรแกรมนี้จะทำการแปลงเอกสาร HTML ไปเป็นเอกสาร XML ก่อนเพื่อความสะดวกในการแปลงเป็นภาษาอื่นๆ เนื่องจาก XML เปรียบเสมือนภาษากลางที่ง่ายต่อการนำไปแปลงเป็นเอกสารที่อยู่ในรูปแบบอื่น จากนั้น โปรแกรมจะทำการแปลงเอกสาร XML นี้ไปเป็นเอกสารในรูปแบบของเอกสาร XHTML เหตุผลที่ทำการแปลงเป็นภาษา XHTML ก่อนนี้ก็เพื่อจุดประสงค์ในอนาคต เมื่อมีการเปลี่ยนแปลงเวอร์ชันของ WML ไปเป็นเวอร์ชัน 2 ซึ่งรูปแบบของ WML เวอร์ชัน 2 จะอยู่บนพื้นฐานของภาษา XHTML ดังนั้นถ้าเราแปลงเป็นภาษา XHTML ไว้ก่อนก็จะเป็นการง่ายในการแปลงต่อไปเป็นภาษา WML เวอร์ชัน แต่เนื่องจากอุปกรณ์ส่วนใหญ่ในปัจจุบันยังสนับสนุนแค่ WML เวอร์ชัน 1.2 เท่านั้น โปรแกรมนี้จึงทำการแปลงเอกสารที่อยู่ในรูปแบบ XHTML ไปเป็น WML เวอร์ชัน 1.2 เพื่อให้ผู้ใช้ส่วนใหญ่สามารถใช้บริการตัวแปลงภาษาของเราได้ เมื่อได้เอกสารที่อยู่ในรูปแบบของ WML แล้ว ก็จะทำการเขียนลงไฟล์ .WML เพื่อให้เว็บเบราว์เซอร์สามารถดึงไปแสดงผลให้ผู้ใช้ได้ อีกทั้งโปรแกรมนี้สามารถทำในส่วนของ CGI (Common Gateway Interface) ซึ่งเป็นส่วนที่ทำให้ผู้ใช้สามารถส่งข้อมูลไปยังฝั่งเซิร์ฟเวอร์ได้



รูปที่ 3.2 ลำดับการแปลงจาก HTML เป็น WML

3.2.2 โครงสร้างของโปรแกรมแปลงภาษา

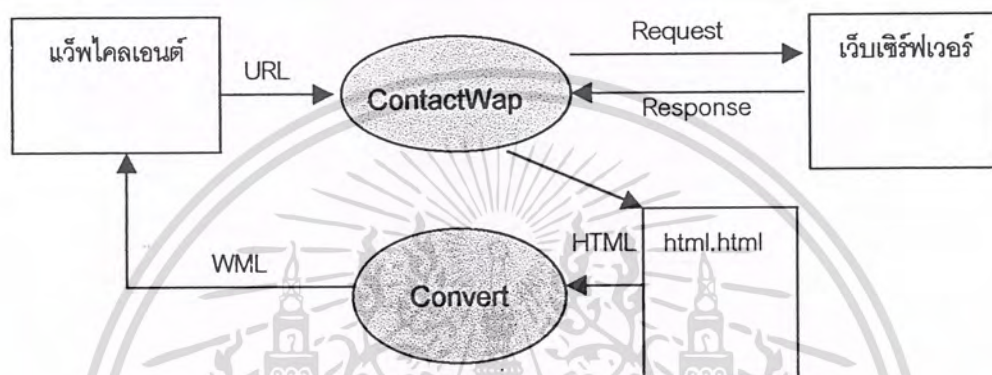
โปรแกรมแปลงภาษา2 มีโครงสร้างโดยรวมดังนี้



รูปที่ 3.3 โครงสร้างโดยรวมของโปรแกรมแปลงภาษา2

โดยตัวโปรแกรมแปลงภาษา2 นี้จะถูกติดตั้งอยู่บนเซิร์ฟเวอร์ที่เรียกว่า “Converter Server” โดยผู้ใช้จะต้องเข้ามายังหน้าเว็บเพจของตัวแปลงภาษานี้ก่อน จากนั้นจึงใส่ URL ของหน้าเว็บเพจที่ต้องการดู จากนั้นตัวแปลงภาษาจะรับ URL มาแล้วส่ง request ไปยังเว็บเซิร์ฟเวอร์ตาม URL นั้น เมื่อเว็บเซิร์ฟเวอร์รับ request แล้วก็จะส่ง response กลับมาในรูปแบบของ HTML จากนั้นตัวแปลงภาษาจะทำการแปลง HTML ให้อยู่ในรูปแบบของ WML และนำไปแสดงผลยังเว็บเบราว์เซอร์ของผู้ใช้

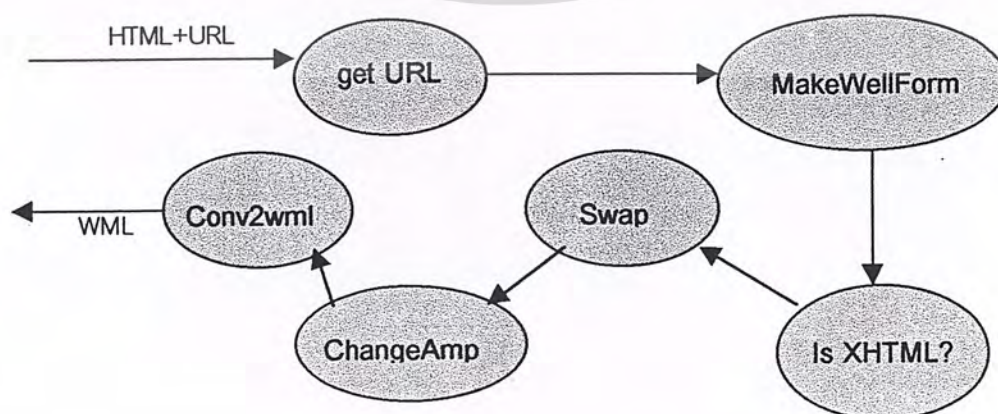
3.2.2.1 โครงสร้างภายในของตัวแปลงภาษา2



รูปที่ 3.4 โครงสร้างภายในของโปรแกรมแปลงภาษา2

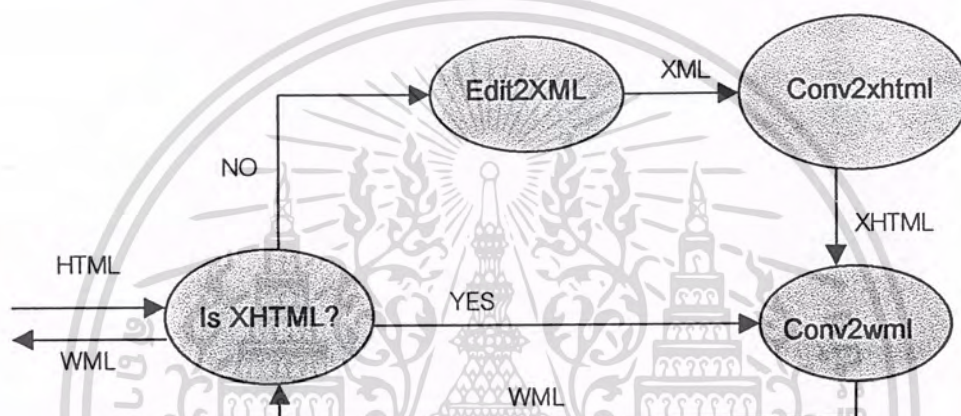
เมื่อรับ URL มาแล้ว ContactWap จะทำหน้าที่ในการส่ง request และรับ response ที่มาจากเว็บเซิร์ฟเวอร์ แล้วเขียน response ที่อยู่ในรูปของ HTML ลงบนไฟล์ html.html จากนั้น Convert จะทำการอ่านไฟล์ html.html นี้มาทำการแปลงให้อยู่ในรูปแบบของ WML แล้วส่งกลับไปแสดงผลยังเว็บไคลเอนต์

3.2.2.2 โครงสร้างภายในของตัว Convert



รูปที่ 3.5 เมธอดใน Convert

ในไฟล์ `html.html` ในบรรทัดแรกจะเป็น URL ของเว็บนั้น เพื่อจะเอา URL นั้นมาใส่ในส่วนอ้างอิง URL ของแท็ก HTML จากนั้นจะเข้าฟังก์ชัน `MakeWellForm` เพื่อทำ HTML ให้อยู่ในรูปแบบที่ Well-form ในระดับหนึ่ง เช่น เพิ่มแท็กปิดที่ขาดหายไป (ได้บางแท็กเท่านั้น) เป็นต้น ต่อมาจะเป็นส่วนในการตรวจสอบว่าเป็น XHTML หรือไม่ซึ่งจะกล่าวรายละเอียดต่อไป จากนั้นจะเข้าฟังก์ชัน `Swap` เพื่อทำการแปลง Encoding Name ให้อยู่ในรูปแบบของ Three-digit Numeric Values เช่น '©' เป็น '©' เพื่อให้เว็บเบราว์เซอร์สามารถแปลความหมายได้ หลังจากนั้นจะไปยังฟังก์ชัน `ChangeAmp` เพื่อทำการเปลี่ยน '&' ที่อยู่ในส่วนของ CGI ให้เป็น '&' หลังจากนั้นจะเข้า `Conv2wml` เพื่อแปลงเป็น WML และนำไปแสดงผลต่อไป



รูปที่ 3.6 เมธอดในการแปลงเป็น WML

ใน `Convert` จะมีฟังก์ชันในการตรวจสอบว่าเว็บเพจนั้นมีคอนเท้นต์เป็น XHTML หรือไม่ ถ้าใช่ก็จะไปที่ `Conv2wml` เลย เพื่อทำการแปลงเป็น WML แต่ถ้าไม่ใช่ก็จะไปที่ `Edit2XML` เพื่อทำการแปลงให้อยู่ในรูปแบบของ XML ก่อน จากนั้นจึงไปที่ `Conv2xhtml` เพื่อทำการแปลงเป็น XHTML แล้วจากนั้นจึงไปที่ `Conv2wml` เพื่อทำการแปลงเป็น WML แล้วส่งไปยังเว็บไคลเอนต์

3.2.3 องค์ประกอบของโปรแกรม

โปรแกรมที่ใช้ในการแปลงเอกสาร HTML เป็น WML ประกอบด้วยซอร์สโค้ดดังต่อไปนี้

1.1 ซอร์สโค้ดภาษา WML จำนวน 1 ไฟล์ ได้แก่

- `converter2wml.wml`

1.2 ซอร์สโค้ดภาษาจาวา จำนวน 7 ไฟล์ ได้แก่

- `ContactWap.java`
- `ContactForm.java`
- `Convert.java`

- edit2XML.java
- Conv2xhtml.java
- Conv2wml.java
- MakeWellForm.java

ContactWap.java

Constructor ไม่มี

เมธอด

doGet (HttpServletRequest req, HttpServletResponse res)

ทำหน้าที่ส่ง request ไปยังเว็บเซิร์ฟเวอร์เพื่อให้ส่งเอกสาร HTML กลับมา และเมื่อรับมาแล้วก็จะทำการเขียนลงไฟล์ html.html จากนั้นจะทำการ redirect ไปที่ Convert.java

ContactForm.java

Constructor ไม่มี

เมธอด

doGet (HttpServletRequest req, HttpServletResponse res)

ทำหน้าที่เหมือน ContactWap แต่จะเพื่อการทำงานบางส่วนเพื่อใช้ในการส่งค่า CGI ที่ผู้ใช้กรอกไปยังเว็บเซิร์ฟเวอร์

Convert.java

Constructor ไม่มี

เมธอด

doGet (HttpServletRequest req, HttpServletResponse res)

เป็นฟังก์ชันหลักที่ทำหน้าที่เรียกใช้เมธอดทั้งหมดที่ใช้ในการแปลงไฟล์ HTML เป็น XML, XHTML และ WML เมื่อแปลงเสร็จแล้วก็จะทำการแสดงผลไปยังเว็บเบราว์เซอร์

edit2XML.java

Constructor ไม่มี

เมธอด

ConvUnicode (String str)

เป็นฟังก์ชันที่ทำการแปลงอักขระพิเศษให้อยู่ในรูปของยูนิโค้ดเพื่อให้เบราว์เซอร์สามารถแสดงอักขระนั้นได้

addUrl (String str)

เป็นฟังก์ชันที่ทำการเพิ่ม URL ให้แก่บางแท็กที่มีการอ้างถึง URL โดยจะใส่ URL แบบเต็ม

checkComment (String str)

เป็นฟังก์ชันที่ทำการตรวจสอบคอมเมนต์และทำการแปลงแอตทริบิวต์ต่างๆให้อยู่ในรูปแบบของ XML

convToLower (String str)

เป็นฟังก์ชันที่ทำการแปลงแอตทริบิวต์ให้เป็นอักษรตัวเล็ก

editEmptyTag (String str)

เป็นฟังก์ชันที่ทำการเพิ่ม '/' ที่ตอนท้ายของแท็กเพื่อให้อยู่ในรูปแบบของ XML

Conv2xhtml.java

Constructor

convert2xhtml (String html)

เป็นเมธอดที่เปรียบเสมือน main ที่ทำหน้าที่เป็นเมธอดหลักที่จะถูกเรียก โดย Convert.java

เมธอด

FindClose (String str)

ทำหน้าที่หน้าที่เก็บรายละเอียดของแท็กต่างๆทั้งหมดที่อยู่ในไฟล์ HTML เช่น จำนวนแท็กเดียวกันที่อยู่ซ้อนกัน เป็นต้น

IsEmptyTag (String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นเป็น Empty tag หรือไม่ ถ้าใช่จะ return 'true' แต่ถ้าไม่ใช่จะ return 'false'

IsTag (String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นใช่แท็กของภาษา HTML หรือไม่ ถ้าใช่จะ return 'true' ถ้าไม่ใช่จะ return 'false'

NotConvert (String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นเป็นแท็กของภาษา HTML มาตรฐานหรือไม่ ถ้าใช่จะ return 'false' ถ้าไม่ใช่จะ return 'true' เพื่อจะได้ไม่ต้องแปลง

convTag (Doc doc, String tagName, String html, int type, String element)

ทำหน้าที่แปลงแท็กบางแท็กที่ง่ายๆ โดยทำการเทียบว่าแท็ก HTML นี้ควรแปลงเป็นแท็ก XHTML อะไร

convert (String html)

เป็นเมธอดที่ทำหน้าที่เรียกให้เมธอดต่างๆเพื่อทำการแปลงไฟล์ HTML แล้วส่งค่ากลับเป็นไฟล์ WML

convsize (String s)

ทำหน้าที่แปลงขนาดของตัวอักษรในแท็ก โดยกำหนดขนาดเป็นพิกเซล

findTag (String tag)

ทำหน้าที่หว่าแท็กนั้นเป็นแท็กอะไร โดยจะรับ String ทั้งหมดที่อยู่ภายในเครื่องหมาย '<' และ '>' แล้วส่งค่ากลับเป็นแท็กจริงๆ

head (String str)

ทำหน้าที่คัดส่วนที่อยู่ข้างบนแท็ก <html> ออก

Conv2wml.java

Constructor ไม่มี

เมธอด

FindClose(String str)

ทำหน้าที่หน้าที่เก็บรายละเอียดของแท็กต่างๆทั้งหมดที่อยู่ในไฟล์ XHTML เช่น จำนวนแท็กเดียวกันที่อยู่ซ้อนกัน เป็นต้น โดยแท็กทั้งหมดจะเก็บอยู่ในอาร์เรย์ชื่อ 'tempTagArr' ก่อน จากนั้นค่อยกรองเอาแต่แท็กเปิด และแท็กปิดบางแท็กได้แก่ <table>, </form>, </td>, </tr>, </th> โดยเก็บแท็กทั้งหมดลงในอาร์เรย์ชื่อ 'tagArr' ในที่นี้กำหนดจำนวนแท็กไว้มากที่สุด 500 แท็ก (รวมแท็กปิดด้วย)

IsEmptyTag(String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นเป็น Empty tag หรือไม่ ถ้าใช่จะ return 'true' แต่ถ้าไม่ใช่จะ return 'false'

IsTag(String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นใช่แท็กของภาษา XHTML หรือไม่ ถ้าใช่จะ return 'true' ถ้าไม่ใช่จะ return 'false'

canConvert(String inTable)

ทำหน้าที่ตรวจสอบตารางที่ส่งเข้ามาว่าตารางนั้นสามารถแปลงให้คงสภาพเป็นตารางเหมือนเดิมได้หรือไม่ โดยจะตรวจสอบว่ามีตารางซ้อนอยู่ข้างในอีกทีหรือไม่ และตรวจสอบว่ามีแท็กที่ต้องห้ามอยู่ภายในตารางหรือไม่ เช่น <table>, <form> เป็นต้น โดยจะทำการเรียกเมธอด 'notAllowInTable' เพื่อตรวจสอบว่าแท็กนั้นใช่แท็กต้องห้ามหรือไม่ ถ้ามีแท็กต้องห้ามอยู่ภายในตารางนั้นแสดงว่าตารางนั้นไม่สามารถคงรูปตารางไว้ได้ต้องทำการตัดตารางทิ้ง (ตัดกรอบของตารางออก) โดยจะ return 'false' แต่ถ้าตารางนั้นสามารถคงรูปของตารางไว้ได้ก็จะ return 'true'

notAllowInTable (String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นใช่แท็กต้องห้ามหรือไม่ เช่น <form>, <table>, <area> เป็นต้น โดยถ้าใช่จะ return 'true' แต่ถ้าไม่ใช่จะ return 'false'

tableGroup (String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นเป็นแท็กที่ใช้ในการสร้างตารางหรือไม่ถ้าใช่จะ return 'true' แต่ถ้าไม่ใช่จะ return 'false'

convTable (Doc doc, String html)

ทำหน้าที่แปลงตารางโดยถ้าค่าที่ส่งมาจาก canConvert เป็น 'true' ก็จะทำการแปลงโดยยังคงรูปของตารางไว้ และถ้าตารางนั้นมีจำนวน column มากกว่า 3 columns ก็จะทำการแบ่งให้แต่ละ row มี 3 column แต่ถ้าค่าที่ส่งมาจาก canConvert เป็น false ก็จะทำการดึงตารางออกแล้วแสดงเหมือนไม่มีแท็กตารางอยู่

convForm (Doc doc, String buffer, String html)

ทำหน้าที่ในการแปลงแท็กฟอร์ม โดยจะรับ String ที่อยู่ภายในแท็ก <form> กับ </form> เข้ามาภายในฟังก์ชัน จากนั้นจะทำการแปลงให้อยู่ในรูปแบบที่สามารถแสดงได้ในรูปของ WML และทำการส่งค่าต่างๆที่ผู้ใช้กรอกไปยังเซิร์ฟเวอร์เพื่อประมวลผลและส่งผลลัพธ์กลับมาเพื่อทำการแปลงและแสดงผลต่อไป

แท็ก <input> จะสามารถแปลงได้เฉพาะที่เป็น text, password, hidden, checkbox, radio เท่านั้น ซึ่งก็เพียงพอต่อการใช้งานและอุปกรณ์เว็บสามารถรองรับได้ในขณะนี้ โดยถ้ามีชนิดอื่นเข้ามาก็จะทำการตัดทิ้ง (ไม่แสดง) ส่วนแท็กในการสร้างฟอร์มอื่นๆที่สามารถแปลงได้คือ <select> และ <textarea> ส่วนแท็กอื่นๆจะทำการตัดทิ้ง

convTag (Doc doc, String tagName, String html, int type)

ทำหน้าที่แปลงแท็กบางแท็กที่ง่าย ๆ เช่น , <i>, เป็นต้น ให้อยู่ในรูปแบบของ WML โดยทำการเทียบว่าแท็ก XHTML นี้ควรแปลงเป็นแท็ก WML อะไร

convert (String html)

เมธอดจะรับ String ที่เก็บไฟล์ XHTML ทั้งหมดไว้ จากนั้นจะทำการเรียกให้เมธอดต่างๆเพื่อทำการแปลงไฟล์ XHTML แล้วส่งค่ากลับเป็นไฟล์ WML โดยแท็กอื่นๆที่ไม่ได้ระบุอยู่ในเมธอดนี้ (ไม่รวมแท็กที่เกี่ยวกับตารางและฟอร์ม) จะถูกตัดทิ้งและจะแสดงเฉพาะข้อความที่อยู่ระหว่างแท็กนั้นเป็นตัวอักษรปกติ เช่น delete ก็จะเหลือแค่ 'delete' (ตัดแท็กออก)

findTag (String tag)

ทำหน้าที่หว่าแท็กนั้นเป็นแท็กอะไร โดยจะรับ String ทั้งหมดที่อยู่ภายในเครื่องหมาย '<' และ '>' แล้วส่งค่ากลับเป็นแท็กจริงๆ เช่นรับมาเป็น ก็จะ return '' เพื่อประโยชน์ในการเทียบหาแท็กต่อไป

head (String str)

ทำหน้าที่ตัดส่วนที่อยู่ข้างบนแท็ก <body> ออก และทำการแปลงแท็กเฟรม (<frame>)

convert2wml (String st)

เป็นเมธอดที่เปรียบเสมือน main ที่ทำหน้าที่เป็นเมธอดหลักที่จะถูกเรียกโดย Convert.java

MakeWellForm.java*Constructor*

MakeWellForm(String str)

ทำหน้าที่ตรวจสอบและแก้ไขแท็กบางแท็กที่ไม่สมบูรณ์ คือ ไม่มีแท็กปิด

เมธอด

IsEmptyTag(String tag)

ทำหน้าที่ตรวจสอบว่าแท็กนั้นเป็นแท็กว่างหรือไม่ ถ้าใช่จะ return 'true' แต่ถ้าไม่ใช่จะ return 'false'

ChangeAmp(String xhtml)

ทำหน้าที่ตรวจสอบและเปลี่ยนตัว '&' ให้เป็น '&'

3.2.4 การทำงานของโปรแกรม

ในการทำงานของโปรแกรมที่ใช้ในการแปลงภาษา HTML เป็น WML นี้จะแบ่งออกเป็น 3 ส่วนการทำงานใหญ่ๆ คือ

1. ส่วนรับ URL จากผู้ใช้และส่ง request ไปยังเว็บเซิร์ฟเวอร์และรับเอกสาร HTML จากเว็บเซิร์ฟเวอร์มาเขียนลงไฟล์
2. ส่วนแปลงเอกสาร HTML เป็น XML และ XHTML
3. ส่วนแปลงเอกสาร XHTML เป็น WML และแสดงผล

ส่วนที่ 1

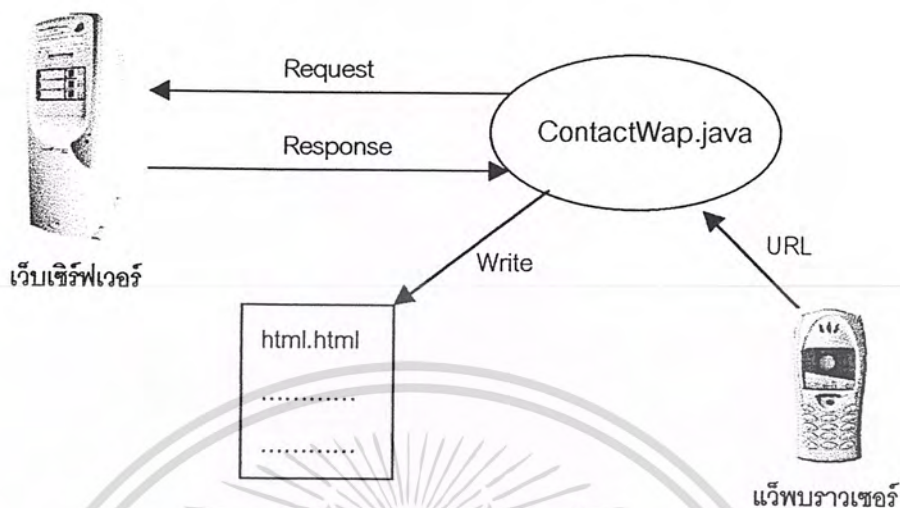
เป็นส่วนที่รับ URL จากผู้ใช้จากเอกสาร WML แล้วส่งค่า URL ไปยังไฟล์จาวาที่จะทำการสร้าง request ไปยังเว็บเซิร์ฟเวอร์ที่เก็บไฟล์เอกสารนั้นอยู่ โดยจะทำการตรวจสอบ URL ที่ผู้ใช้กรอกมาว่าถูกต้องตามรูปแบบที่ระบุอยู่ใน rfc2393 ถ้าไม่ถูกต้องจะพยายามเปลี่ยนให้อยู่ในรูปแบบที่ถูกต้องและเมื่อถูกต้องแล้วก็ทำการติดต่อไปยังเว็บเซิร์ฟเวอร์ตาม URL นั้น โดยใช้โค้ด ดังนี้

```
URL url = new URL(url);
```

```
URLConnection connection = url.openConnection();
```

```
connection.setDoOutput(true);
```

และรับไฟล์เอกสารที่ส่งมาจากเซิร์ฟเวอร์ที่อยู่ในรูปแบบของ HTTP Message ซึ่งถ้าไม่มีข้อผิดพลาดเกิดขึ้น ภายใน HTTP Message นั้นก็จะมีเอกสาร HTML บรรจุอยู่ จากนั้นจะทำการเขียนไฟล์เอกสาร HTML นี้ลงในไฟล์ html.html โดยมี URL อยู่บนบรรทัดแรกของไฟล์ เสร็จแล้วจะทำการ Redirect ไปยังส่วนต่อไป



รูปที่ 3.7 การทำงานของส่วนที่ 1

ส่วนที่ 2

ส่วนนี้ทำหน้าที่ในการแปลงไฟล์ HTML ไปเป็น XML และ XHTML การทำงานเริ่มจากอ่านเอกสาร HTML ที่อยู่ภายในไฟล์ html.html ที่ได้จากส่วนที่ 1 และเก็บเอกสาร HTML ทั้งหมดอยู่ภายในสตริง และนำสตริงนี้ส่งไปยังฟังก์ชันที่ใช้ในการแปลง HTML เป็น XML และเปลี่ยนต่อไปเป็น XHTML โดยฟังก์ชันนี้จะทำการแปลงรูปแบบของแท็กให้ตรงตามกฎของ XHTML ซึ่งได้กล่าวไว้แล้วในบทที่ 2 และทำการเปลี่ยนแท็กที่ไม่มีใน XHTML ไปเป็นแท็กที่มีใน XHTML และมีการทำงานเหมือนกัน ดังตาราง

HTML	XHTML
<APPLET>	<OBJECT>
<BASEFONT>	STYLE SHEET
<CENTER>	STYLE SHEET
<DIR>	
	STYLE SHEET
<ISINDEX>	<INPUT>
<LISTING>	<PRE>
<MENU>	
<PLAINTEXT>	<PRE>
<S>	
<STRIKE>	

<XMP>	<PRE>
-------	-------

ตารางที่ 3.1 การเปลี่ยนจาก HTML เป็น XHTML

เมื่อได้ทำการแปลงเป็น XHTML เสร็จแล้วจะได้สตริง 1 ตัวที่เก็บเนื้อเอกสาร XHTML ไว้ทั้งหมด จากนั้นจะนำสตริงนี้ส่งไปยังส่วนที่ 3 เพื่อแปลงเป็น WML ต่อไป

ส่วนที่ 3

ส่วนนี้ทำหน้าที่แปลงเอกสารให้อยู่ในรูปแบบ WML โดยการทำงานจะเริ่มจากนำสตริงที่บรรจุเอกสาร XHTML ที่ได้จากส่วนที่ 2 มาเข้าฟังก์ชันเพื่อหาแท็กทั้งหมดที่อยู่ในเอกสารและเก็บแท็กทั้งหมดลงในอาร์เรย์เพื่อใช้ในการแปลงเป็นไฟล์ WML ต่อไป จากนั้นจะนำสตริงตัวเดิมมาเข้าฟังก์ชันที่ใช้ในการส่วนหัวของเอกสาร โดยจะทำการแปลงส่วนที่อยู่เหนือแท็กบอดีขึ้นไปรวมทั้งแปลงเฟรม (Frame) โดยถ้าเอกสารนั้นมีการใช้เฟรมก็จะไม่มีการประกาศส่วนบอดี เมื่อแปลงส่วนหัวเสร็จแล้วและถ้ามีการใช้เฟรมก็จะนำส่วนที่แปลงได้ไปแสดงผลเลย แต่ถ้าไม่มีการใช้เฟรมก็จะส่งสตริงนี้ไปยังฟังก์ชันที่ทำหน้าที่แปลงส่วนบอดี โดยฟังก์ชันนี้จะ scan หาและแปลงทีละแท็กเริ่มจากตัวอักษรตัวแรก โดยเมื่อเจอแท็กก็จะเขียนดูว่าเป็นแท็กอะไรและจะแปลงให้อยู่ในรูปแบบอย่างไรเพื่อให้เหมาะสมต่อการนำไปแสดงผล ตารางเทียบแท็กบางแท็กที่สำคัญเพื่อใช้ในการแปลงแสดงได้ดังนี้

XHTML	WML
<!-->	<!-->
<a> href	<anchor><go><postfield>
<address>	<i>
	
<blockquote>	 ” “
<big>	<big>
<cite>	<i>
<dd>	
<dfn>	<i>
<dt>	
	
<frame> src	<anchor><go><postfield>

name	
<form>	<anchor><go><postfield>
<h1> - <h2>	<big>
<hr>	 -----
<i>	<i>
	
src	src
alt	alt
<input>	<input/>
name	name
value	value
type = [text password]	type = [text password]
<input>	<select>
name	name
value	multiple = true
type = checkbox	<option>
	value
<input>	<input/>
name	name
value	multiple = false
type = radio	<option>
	value
	 -
<meta>	<meta/>
<option>	<option>
value	value
<optgroup>	<optgroup>
label	title
<p>	<p>
align = [left right center]	align = [left right center]
<q>	“ “
<select>	<select>
name	name
multiple = true	mutiple = true

<option> value	<option> value
<select> name multiple = false <option> value	<select> name multiple = false <option> value
<table> cols	<table> columns
<td>	<td>
<th>	<td>
<tr>	<tr>
<textarea> name	<input/> name type = text
<var>	<i>

ตารางที่ 3.2 การเปลี่ยนจาก XHTML เป็น WML

หมายเหตุ รายละเอียดการทำงานของแต่ละแท็กดูได้ที่ภาคผนวก ก.

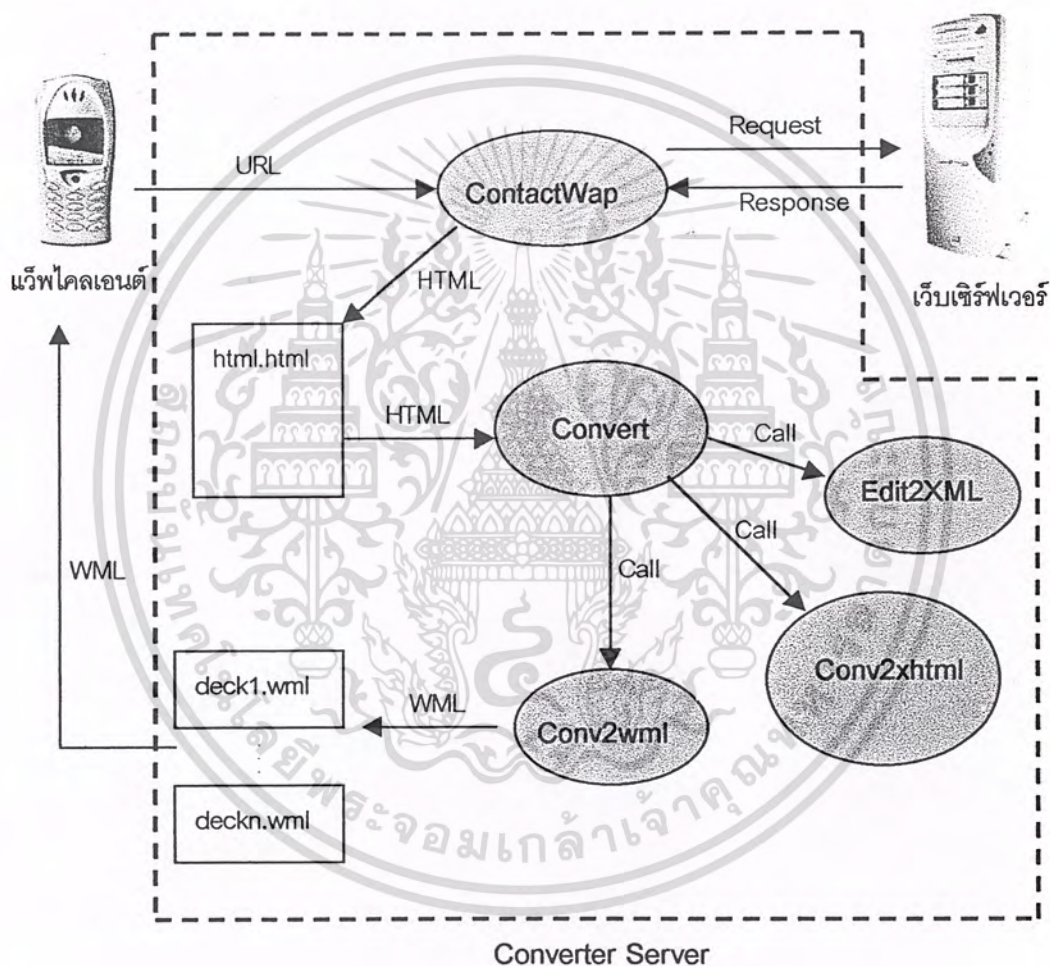
เมื่อทำการแปลงให้อยู่ในรูปแบบ WML เสร็จแล้วก็จะทำการเขียนลงไฟล์ .wml โดยมีการแบ่งขนาดไฟล์ถ้าขนาดของเดื้คมากกว่าขนาดหน่วยความจำของอุปกรณ์ที่ใช้เปิด

บทที่ 4

การทำงานของโปรแกรมแปลงภาษา2

4.1 การทำงานทั่วไป

ตัวแปลงที่สร้างขึ้นนี้จะถูกฝังอยู่ในเว็บเซิร์ฟเวอร์ซึ่งในที่นี้จะเรียกว่า “คอนเวอร์เตอร์เซิร์ฟเวอร์ (Converter Server)” เพื่อไม่ให้สับสนกับเว็บเซิร์ฟเวอร์ซึ่งให้บริการเว็บไซต์อยู่



รูปที่ 4.1 การทำงานโดยทั่วไป

ขั้นตอนการทำงาน

1. เมื่อผู้ใช้กรอกค่า URL ที่ต้องการแปลงผ่านเว็บเพจที่สร้างขึ้นเพื่อให้บริการแปลง แล้วทำการ Submit เว็บเบราว์เซอร์จะส่ง URL มาให้เซิร์ฟเวอร์ที่ชื่อ ContactWap
2. ContactWap จะนำค่า URL ที่ได้มาประกอบในการส่ง request ไปยังเว็บเซิร์ฟเวอร์ที่เว็บเพจนั้นอยู่

3. เมื่อเว็บเซิร์ฟเวอร์ส่ง response ที่มี HTML อยู่ด้วยมาแล้ว ContactWap จะทำการเขียน HTML นั้นลงบนไฟล์ html.html จากนั้น ContactWap จะทำการ redirect ตัวเองมาที่ Convert
4. Convert จะทำการอ่านไฟล์ HTML ที่ได้จากการ ContactWap มาเก็บไว้ในสตริง 1 ตัว แล้วเรียก Edit2XML โดยส่งสตริงนี้เข้าไปยังฟังก์ชันนี้ด้วย
5. Edit2XML จะทำการแปลงสตริงนั้นให้อยู่ในรูปแบบของ XML แล้วส่งคืนให้ Convert
6. Convert ทำการเรียก Conv2xhtml โดยส่งสตริงที่ได้จาก Edit2XML เข้าไปยังฟังก์ชันด้วย
7. Conv2xhtml จะทำการแปลงสตริงนั้นให้อยู่ในรูปแบบของ XHTML แล้วส่งกลับคืนให้ Convert
8. Convert ทำการเรียก Conv2wml โดยส่งสตริงที่ได้จาก Conv2xhtml เข้าไปยังฟังก์ชันด้วย
9. Conv2wml จะทำการแปลงสตริงนั้นให้อยู่ในรูปแบบของ WML แล้วทำการเขียนลงไฟล์ deck1.wml โดยถ้าขนาดของเค็ลใหญ่กว่าขนาดของหน่วยความจำของอุปกรณ์เว็บ ก็จะทำให้การแบ่งออกเป็นหลายๆเค็ล โดยจะเขียนลงไฟล์ชื่อ deck2.wml, deck3.wml, ... , deckn.wml ไปเรื่อยๆจนกว่าจะหมด เมื่อเสร็จแล้วก็จะกลับมาที่ Convert
10. Convert ทำการอ่านไฟล์ WML แล้วส่งไปแสดงผลยังเว็บเบราว์เซอร์

4.2 การทำงานที่มีแท็กลิงก์หรือเฟรม

เนื่องจากเว็บไม่สามารถแสดงผลในรูปแบบของเฟรมได้ ดังนั้นเมื่อมีแท็ก <frame> ปรากฏอยู่ภายในเอกสาร HTML จะยังคงแท็ก <frame> ไว้อยู่ใน XHTML แต่เมื่อทำการแปลงเป็น WML จะทำการแปลงแท็ก <frame> ให้เป็นลิงก์เพื่อให้ผู้ใช้สามารถเลือกดูแต่ละเฟรมได้

ขั้นตอนการลิงก์นั้นเหมือนกับการทำงานทั่วไป ต่างกันตรงที่เมื่อผู้ใช้คลิกที่ลิงก์นั้นจะมีการส่งค่า URL ของลิงก์นั้นไปให้ ContactWap เพื่อให้ ContactWap ส่ง request ไปยังเว็บเซิร์ฟเวอร์เพื่อนำไฟล์ HTML มาแปลงแล้วแสดงผลต่อไป

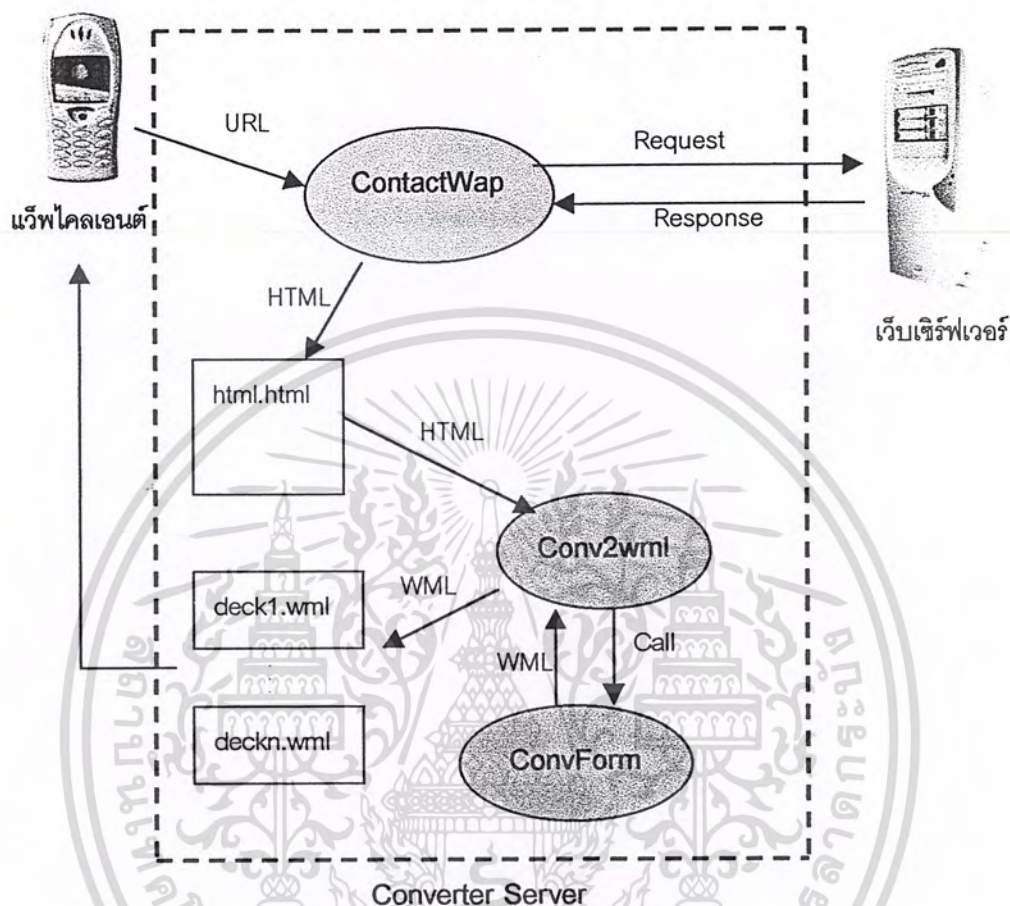
ขั้นตอนการทำงาน

1. เมื่อผู้ใช้เลือกคลิก เว็บเบราว์เซอร์จะส่ง URL ของลิงก์นั้นมาให้ ContactWap
2. ContactWap ทำการส่ง request ไปยังเว็บเซิร์ฟเวอร์ที่เว็บเพจนั้นอยู่
3. ContactWap รับ response จากเว็บเซิร์ฟเวอร์ที่มี HTML อยู่
4. ContactWap จะเขียน HTML นั้นลงไฟล์ html.html และ redirect ตัวเองมาที่ Convert
5. Convert ทำการแปลงให้อยู่ในรูปแบบของ WML แล้วส่งไปแสดงผลยังเว็บเบราว์เซอร์

4.3 การทำงานที่มีแท็กฟอร์ม

เนื่องจากภาษา WML แท็ก <input> ไม่มีการกำหนด type ที่เป็น radio กับ checkbox โดย WML จะอิมพลิเมนต์ radio กับ checkbox ไว้ในแท็ก <select> โดยขึ้นอยู่กับกำหนดยาค่าแอตทริบิวต์ของ multiple ถ้ากำหนดให้เป็น true ก็จะทำหน้าที่เป็น checkbox แต่ถ้ากำหนดให้เป็น false ก็จะทำหน้าที่เป็น radio ส่วนแท็ก <select> ของ HTML ก็ยังคงเดิมเป็น <select> เหมือนเดิมแต่รูปแบบการ

แสดงผลจะแตกต่างจาก HTML ส่วนแท็ก <textarea> เนื่องจากแท็กนี้มีการทำงานคล้ายกับแท็ก <input> ที่มี type เป็น text ดังนั้นเราจึงทำการแปลงแท็กนี้ให้เป็น <input> ที่มี type เป็น text



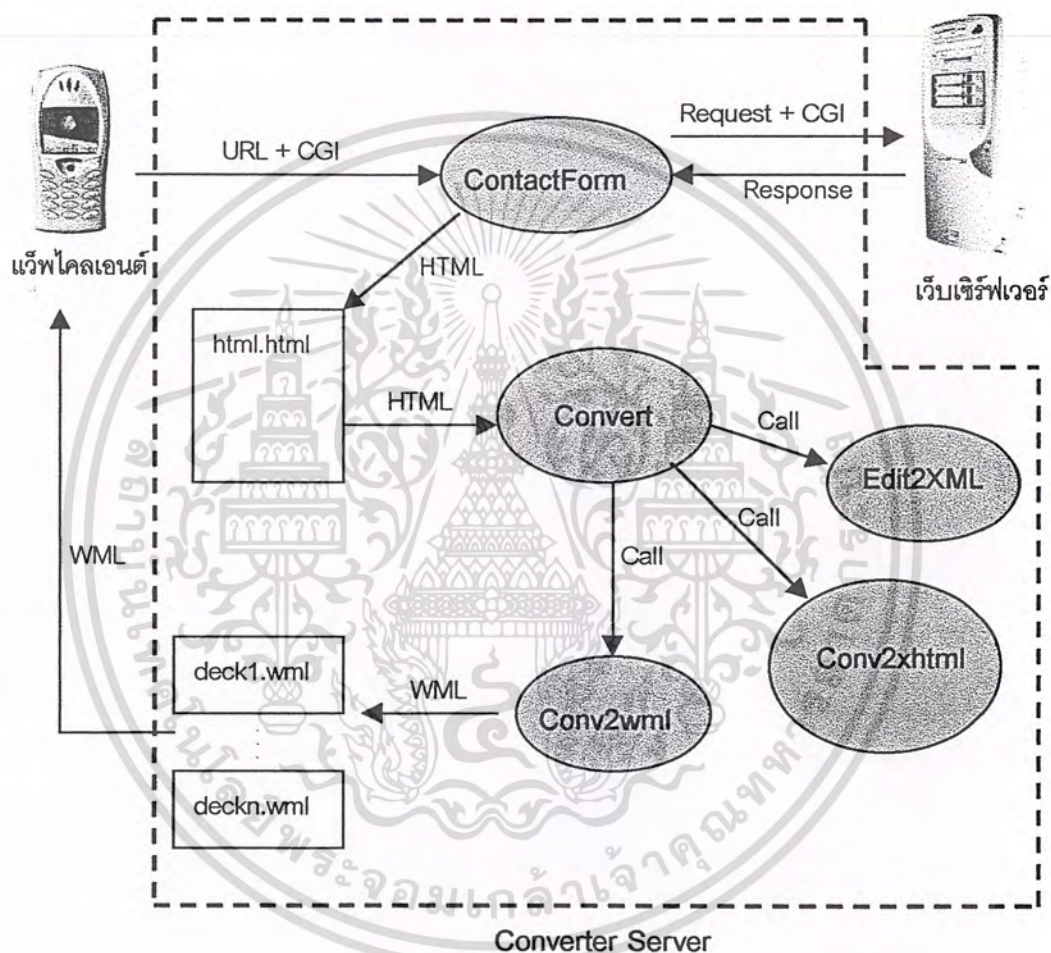
รูปที่ 4.2 การทำงานในการแปลงแท็กฟอร์ม

ขั้นตอนการทำงาน

1. การทำงานเริ่มแรกจะเหมือนกับการแปลงทั่วไป โดยผู้ใช้กรอก URL ส่งมายัง ContactWap และรับ response ที่ส่งมาจากเว็บเซิร์ฟเวอร์ และทำการแปลงตามปกติ
2. เมื่อ Conv2wml เจอแท็ก <form> ก็จะมีการตัดส่วนที่อยู่ระหว่างแท็ก <form> กับ </form> ส่งไปยังเมธอดที่ชื่อว่า ConvForm ซึ่งเป็นเมธอดที่ทำหน้าที่ในการแปลงแท็กฟอร์ม โดย ConvForm จะทำการแปลงแท็กฟอร์มแล้วส่งค่าสตริงที่เก็บข้อความที่แปลงเป็น WML แล้วกลับมายัง Conv2wml เพื่อทำการแปลงส่วนที่เหลือต่อไป และนำไปแสดงผลยังเว็บเบราว์เซอร์ของผู้ใช้ต่อไป

4.4 การทำงานเมื่อมีการ Submit ฟอรั่ม

เมื่อผู้ใช้กรอกข้อมูล CGI เสร็จแล้วและทำการกดปุ่ม Submit โปรแกรมจะส่งค่า URL และ CGI ที่ผู้ใช้กรอกไปยัง ContactForm แทนที่จะเป็น ContactWap ตามปกติ และ ContactForm จะทำการส่งค่า URL และ CGI ไปยังเว็บเซิร์ฟเวอร์ เพื่อให้เว็บเซิร์ฟเวอร์ทำการประมวลผลและส่งผลที่ได้ที่อยู่ในรูปของ HTML กลับมา โปรแกรมก็จะทำการแปลงผลที่ได้นั้นให้อยู่ในรูปของ WML และแสดงผลให้ผู้ใช้ต่อไป



รูปที่ 4.3 การทำงานเมื่อมีการ Submit ฟอรั่ม

ขั้นตอนการทำงาน

1. ผู้ใช้ทำการกรอกข้อมูล CGI และทำการกดคลิก SUBMIT เพื่อทำการส่ง URL และค่า CGI ที่ผู้ใช้กรอกมายัง ContactForm
2. ContactForm จะทำการรับ ค่า URL และ CGI จากเว็บเบราว์เซอร์ และส่งต่อไปยังเว็บเซิร์ฟเวอร์ เพื่อให้เว็บเซิร์ฟเวอร์ทำการประมวลผล CGI และส่งผลที่เป็น HTML กลับมายัง ContactForm และทำการแปลงตามปกติต่อไป

บทที่ 5

ผลการทดลองและการเปรียบเทียบ

5.1 การใช้งาน

ในการใช้งานตัวแปลงภาษานี้ ผู้ใช้สามารถเรียกผ่านเว็บเบราว์เซอร์ใดๆผ่านทางเว็บไซต์ URL: <http://localhost:8080/webdav/convert.wml> โดยการพิมพ์ URL ที่ต้องการแปลงลงไปในฟอร์ม



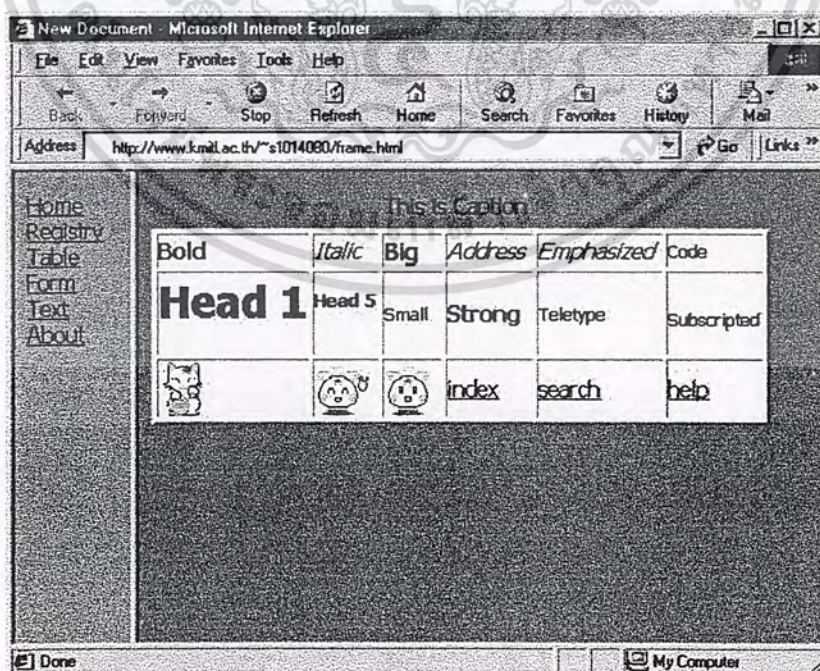
รูปที่ 5.1 เว็บไซต์ของ [HT->W]ML Converter

5.2 ตัวอย่างการแปลง

แสดงผลโดยใช้เบราว์เซอร์ IE5.5 และ อินเทอร์เน็ตเอกซ์พลอเรอร์

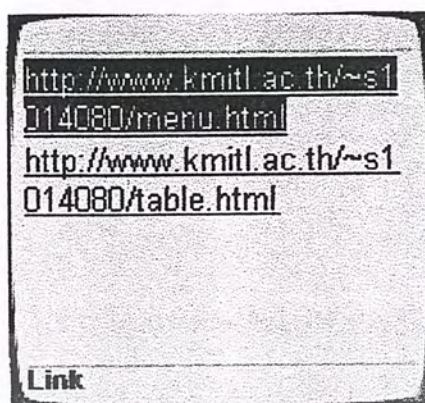
5.2.1 ตัวอย่างการแปลง frame

การแสดงผลหน้า HTML บนเว็บไคลเอนต์



รูปที่ 5.2 การแสดงผลเว็บเพจ ภาษา HTML โดยใช้แท็ก frame

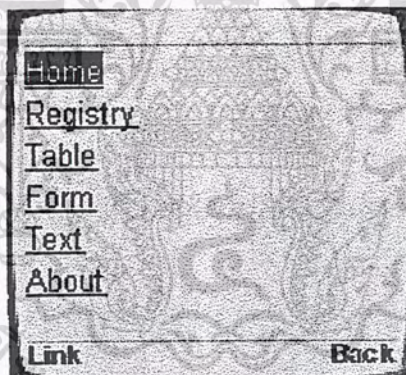
โดยข้อมูลเดียวกัน เมื่อแปลงเป็น WML เพื่อแสดงผลบนเว็บเบราว์เซอร์ ได้ดังต่อไปนี้



รูปที่ 5.3 การแสดงผลเว็บเพจ ภาษา WML โดยแปลงแต่ละเฟรมให้เป็นลิงก์

เนื่องจากในภาษา WML ไม่มีการใช้แท็กเฟรม จึงจำเป็นต้องทำการแปลงเฟรมให้อยู่ในรูปของลิงก์ที่จะลิงก์ไปยังหน้าเฟรมนั้นๆ โดยชื่อลิงก์จะเป็น URL ของหน้าเฟรมนั้นๆ

เมื่อเลือกลิงก์ <http://www.kmitl.ac.th/~s1014080/menu.html>



รูปที่ 5.4 การแสดงผลเมื่อเลือกเฟรม Menu

5.2.2 ตัวอย่างการแปลง form

Example Form - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address http://www.kmitl.ac.th/~s1014080/test_form.html Go Links

Name :

Gender : ☐ male ☐ female

Education

Interesting
☐ Sport ☐ Computer ☐ Reading

Address

Done My Computer

รูปที่ 5.5 การแสดงผลเว็บเพจ ภาษา HTML โดยใช้แท็ก form

หลังจากแปลงเป็น WML จะได้ผลลัพธ์ดังรูป

Name :

Gender :
[male]

Education
[Primary school]

Interesting
[]

Address
[Enter Address]

SUBMIT

Edit Back

รูปที่ 5.6 ตัวอย่างการแปลงฟอร์ม

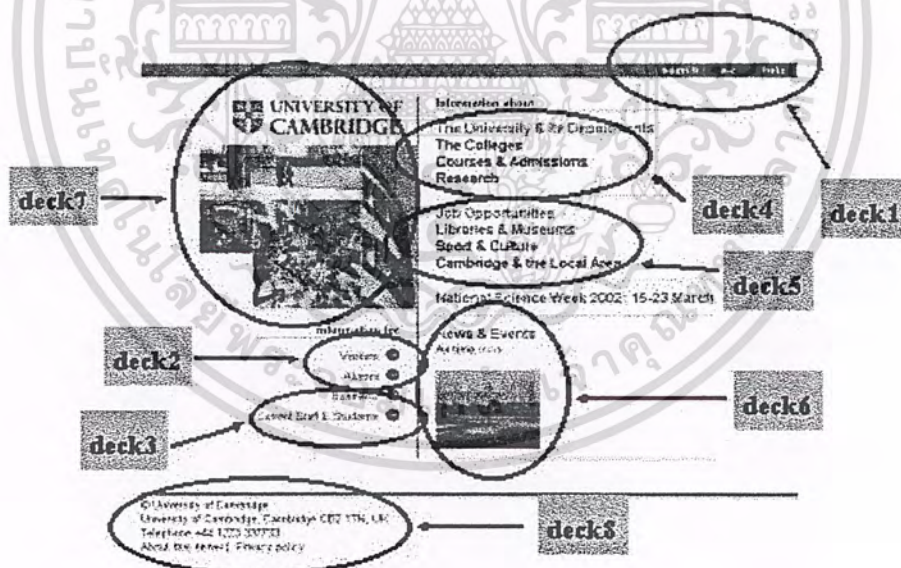
5.2.3 ตัวอย่างการแปลง table

จากรูปที่ 5.3 เมื่อทำการเลือกลิงค์ <http://www.kmitl.ac.th/~s1014080/table.html> ซึ่งเป็นเฟรมที่เป็นตารางจะได้ผลลัพธ์ดังรูป

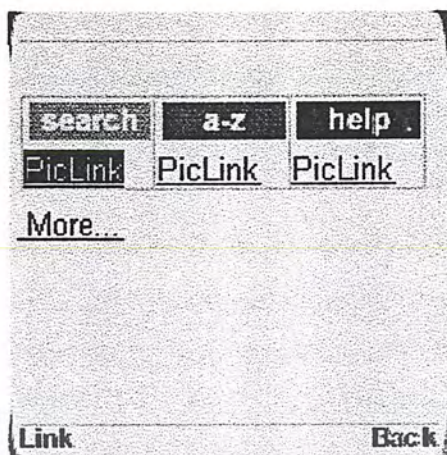
Bold	<i>Italic</i>	Big
Address	<i>Emphasized</i>	Code
Head 1	Head 5	Small
Strong	Teletype	Subscripted
		
index	search	help
CAPTION : "This is Caption"		
Link		Back

รูปที่ 5.7 ตัวอย่างการแปลงตาราง

5.2.4 ตัวอย่างการแบ่ง deck



รูปที่ 5.8 ตัวอย่างเว็บเพจเพื่อใช้ในการแบ่ง deck



รูปที่ 5.9 ตัวอย่างของเด็กที่1

จากรูป เนื่องจากเว็บเพจมีขนาดใหญ่มากเมื่อทำการแปลงแล้วขนาดของไฟล์ WML จะมีขนาดใหญ่เกินไป จึงต้องทำการแบ่งเด็ก โดยแต่ละเด็กจะเชื่อมโยงกันด้วยลิงก์ที่ชื่อว่า 'More...'

5.3 การเปรียบเทียบ

เป็นการเปรียบเทียบตัวแปลงภาษาแบบอัตโนมัติอย่างสมบูรณ์ระหว่างตัวแปลงภาษา2 ([HT->W]ML Converter) กับตัวแปลงภาษาอื่นที่ปรากฏ ดังต่อไปนี้

5.3.1 IBM WebSphere Transcoding Publisher

เป็นตัวแปลงโค้ดของ IBM สำหรับการติดตั้งโปรแกรมนี้จะต้องลง JDK1.3 หรือ เวอร์ชันที่มากกว่า ก่อน บนแพลตฟอร์ม ดังนี้ AIX(R), Microsoft Windows NT, Microsoft Windows 2000, Sun Solaris, and Linux และตัวแปลงโค้ดนี้ มีความสามารถในการแปลงโค้ดต่างๆ ดังต่อไปนี้

- Fragmentation Transcoder
- Image Transcoder
- Resource Repository
- HTML-to-Palm.Net Transcoder
- HTML-to-VoiceXML Transcoder
- Preference Monitor
- Annotation Transcoder
- XML Stylesheet Transcoder
- Machine Translation Transcoder
- HTML-to-HDML Transcoder
- HTML Simplification

- HTML-to-WML Transcoder
- HTML-to-CompactHTML (i-mode) Transcoder
- HTML Document Object Model (DOM) Generator

แต่ตัวแปลงโค้ดนี้สามารถแปลงโค้ดแบบสแตติก ได้เท่านั้น คือ แปลงออกมาเป็นโค้ดภาษา WML แล้วต้องบันทึกคอนเท้นต์นั้นลงไฟล์(.wml) ต่างหาก และถ้าต้องการดูผลการแปลง ต้องนำไปเปิดด้วยอิมูเลเตอร์โนเกีย อีกที จึงจะดูผลการแปลงได้

IBM WebSphere Transcoding Publisher	[HT->W]ML Converter 2
• ไม่แปลงแท็ก<table> แต่จะดึงแท็กอื่นๆที่อยู่ในตารางออกมา ถึงแม้ว่าจะเป็น ตารางซ้อนตารางก็ตาม • แปลงรูปภาพไม่ได้ • จะแปลงเฟรมแต่ละเฟรมเป็นลิงก์1, ลิงก์2, ลิงก์3 ไปเรื่อยๆ ไม่รู้ว่าแต่ละลิงก์คืออะไร • แสดงข้อมูลทั้งหมดใน 1 เด็ค • แปลงจากภาษา HTML เป็น ภาษา WML, HDML, VoiceXML, Palm.Net, CompactHTML (i-mode) และสามารถแปลงโค้ดแบบอื่นๆ ได้ตามที่กล่าวมาแล้ว • ต้องมีไฟล์ .html เพื่อนำมาแปลงเป็น .wml แล้วจึงใช้เว็บเบราว์เซอร์เปิดอีกที	• แปลงตารางได้ ทั้งตารางเดียวและ ตารางซ้อนตาราง • แปลงรูปภาพได้ • แปลงเฟรมเป็นลิงก์ โดยชื่อลิงก์จะเป็น URL ของแต่ละเฟรมนั้น • สามารถแบ่งหน้าเพจที่มีขนาดใหญ่ให้มีขนาดเหมาะสมกับเด็คได้ • แปลงจาก HTML เป็น XML,XHTML,WML ได้เท่านั้น • ไม่ต้องมีไฟล์ .html แต่โปรแกรมจะไปดึงจาก Web Server มาเองแล้วแสดงออกทางเว็บเบราว์เซอร์โดยอัตโนมัติ

ตารางที่ 5.1 การเปรียบเทียบกับ IBM WebSphere Transcoding Publisher

5.3.2 โปรแกรมแปลงภาษา1

เป็นตัวแปลงภาษาแบบอัตโนมัติ ซึ่งเรียกใช้ได้จากเว็บเพจ แต่ที่เครื่องเซิร์ฟเวอร์นั้นจะติดตั้งโปรแกรม JDK1.3 หรือมากกว่า, Apache HTTP Server เวอร์ชัน1.3.14, Jakarta-Tomcat v3.1.1, Cocoon1.8 และต้องดาวน์โหลด ApacheModuleJServ.dll3.1.1มาด้วย และตัวแปลงนี้จะแปลงแท็กต่างๆ แท็ก พบว่าสามารถแปลงเว็บเพจส่วนใหญ่ได้ แต่จะไม่สามารถแปลงโค้ด HTML ที่มีความซับซ้อนมาก (เช่น ใช้ทูตในการสร้าง) หรือไม่อยู่ในรูปแบบที่ถูกต้องตามไวยากรณ์ จะทำให้ไม่สามารถ

แปลงได้ และไม่มีการตรวจสอบขนาดเค็ลของไฟล์ WML ว่ามีขนาดใหญ่เกินกว่าอุปกรณ์จะใช้ได้หรือไม่ ซึ่งอาจให้การทำงานผิดพลาดได้

[HT->W]ML Converter 1	[HT->W]ML Converter 2
● แสดงข้อมูลทั้งหมดใน 1 เค็ล ● แปลงจาก HTML เป็น WML ได้เท่านั้น ● ไม่สามารถแปลงตารางซ้อนตารางได้ ● Attribute บางตัวต้องใส่ ถ้าไม่ใส่จะทำงานไม่ได้	● สามารถแบ่งหน้าเพจ ที่มีขนาดใหญ่ให้มีขนาดเหมาะสมกับเค็ลได้ ● แปลงจาก HTML เป็น XML, XHTML, WML ได้ ● แปลงตารางได้ ทั้งตารางเดี่ยวและ ตารางซ้อนตาราง ● Attribute บางตัวไม่ต้องใส่ก็ได้ โดยโปรแกรมจะใส่ค่าที่เหมาะสมให้เอง

ตารางที่ 5.2 การเปรียบเทียบกับโปรแกรมแปลงภาษา



บทที่ 6

สรุปและบทวิเคราะห์

6.1 สรุป

1. จากการทดลองปรับปรุงตัวแปลงภาษาจาก HTML เป็น WML แบบอัตโนมัติ โดยทำการปรับปรุงข้อเสียของโปรแกรมแปลงภาษา1 แล้วทำการทดลองแปลงในทุกๆแท็ก และเพิ่มความซับซ้อนของเว็บเพจให้มากขึ้น ก็สามารถแปลงได้ดีในระดับหนึ่ง แต่ในกรณีไฟล์ HTML ดั้งเดิมไม่อยู่ในรูปแบบที่กำหนดก็จะทำให้ไม่สามารถแปลงได้หรือถ้าแปลงได้ข้อมูลอาจผิดพลาดไปจากข้อมูลต้นฉบับ
2. แต่โปรแกรมนี้นี้มีขนาดที่ใหญ่และใช้หน่วยความจำมาก จึงต้องใช้เวลาและทรัพยากรในการแปลงแต่ละครั้งมากพอสมควร จึงนับเป็นข้อเสียของโปรแกรมนี้นี้

6.2 ข้อดี

ข้อดีของโปรแกรมแปลงภาษา2 ที่พัฒนาและแก้ไขข้อเสียของโปรแกรมแปลงภาษา1 และที่มีอยู่ในท้องตลาด คือ

- สามารถแปลงแท็กได้ทุกแท็กใน HTML4.01 และ XHTML1.0
โปรแกรมนี้สามารถรองรับแท็กได้ทุกแท็กที่ประกาศไว้ใน HTML4.01 และ XHTML1.0 โดยสามารถเปลี่ยนการทำงานในการเปลี่ยนแท็กที่ต้องการ ได้เพียงเขียนฟังก์ชันที่ต้องการเปลี่ยนมาแทนที่ฟังก์ชันเดิมที่มีอยู่ โดยไม่กระทบแท็กอื่น ยกเว้นว่าฟังก์ชันนั้นใช้แปลงมากกว่าหนึ่งแท็ก
- สามารถแปลงไฟล์รูปภาพได้
เนื่องจากเว็บไซต์ในปัจจุบันประกอบด้วยรูปภาพมากมายและบางรูปภาพสามารถสื่อความหมายได้มาก ดังนั้นถ้ามีการนำรูปภาพนี้มาแสดงด้วยก็สมารถทำให้ผู้ใช้สามารถเข้าใจเนื้อหาของเว็บนั้นได้ดีขึ้น โดยโปรแกรมนี้นี้จะทำการย่อขนาดรูปให้สามารถแสดงบนเว็บเบราว์เซอร์ได้เหมาะสม และพยายามคงเนื้อหาเดิมของเว็บไซต์ต้นฉบับไว้ให้มากที่สุด
- สามารถแปลงไฟล์ HTML ที่ไม่ well-form ได้ระดับหนึ่ง
โปรแกรมแปลงภาษา1 ไฟล์ HTML ที่นำมาแปลงจะต้องเป็นแบบ well-form เท่านั้น แต่โปรแกรมนี้นี้ไฟล์ HTML ที่จะนำมาแปลงไม่จำเป็นต้อง well-form เท่าที่ควรโดยแอดทริบิวต์ภายในแท็กไม่จำเป็นต้องใส่เครื่องหมายฟันหนู (") ก็ได้โดยโปรแกรมจะทำการใส่ให้เอง
- สามารถแปลงฟอร์มได้
เว็บไซต์ในปัจจุบันส่วนมากจะมีส่วนที่ให้ผู้ใส่ข้อมูลเพื่อส่งไปประมวลผลที่ฝั่งเซิร์ฟเวอร์ได้ ซึ่งส่วนนี้โปรแกรมที่มีอยู่ตามท้องตลาดยังไม่สามารถทำได้ แต่โปรแกรมนี้นี้สามารถ

แปลงฟอร์มในเว็บเพจให้อยู่บนเว็บเพจและให้ผู้ใช้ใส่ข้อมูลและส่งไปประมวลผลที่ฝั่งเซิร์ฟเวอร์ได้ โดยพยายามแปลงฟอร์มให้ทำหน้าที่เหมือนกับฟอร์มในเว็บเพจมากที่สุด และที่สำคัญคือข้อมูลที่ส่งไปให้เซิร์ฟเวอร์จากเว็บเพจต้องเหมือนกับส่งจากเว็บเพจ

- มีการเขียนเป็นไฟล์ WML ก่อนนำไปแสดงผล

โปรแกรมแปลงภาษา1 จะไม่มีการเขียนลงไฟล์ WML ไว้แต่นำไปแสดงผลเลย ส่วนโปรแกรมนี้อาจทำการเขียนลงไฟล์ WML ไว้ก่อนแล้วจึงนำไฟล์นี้ไปแสดงผลทำให้สามารถเก็บไฟล์ WML นี้ไว้ได้เพื่อสำหรับเว็บไซต์ที่มีการเปลี่ยนแปลงเนื้อหาภายในไม่บ่อย จะได้ไม่ต้องทำการแปลงอีกสามารถนำเอาไฟล์นี้ออกแสดงผลได้เลยเป็นการประหยัดเวลาและการประมวลผลของฮาร์ดแวร์ อีกทั้งเจ้าของเว็บอาจนำไฟล์นี้ไปใช้ทำเป็นเว็บเพจของเว็บเพจนั้นได้เลย

- สามารถแปลงเว็บไซต์ที่มีโครงสร้างเป็นตารางได้

เนื่องจากเว็บไซต์ในปัจจุบันจำนวนมากจะใช้ตารางมาช่วยในการจัดโครงสร้างของเว็บเพจ เพื่อให้การแสดงผลง่ายต่อการใช้งานของผู้ใช้ โดยโปรแกรมแปลงภาษา1 และ โปรแกรมตามท้องตลาดไม่สามารถแปลงเว็บเพจที่อยู่ในลักษณะนี้ได้ และบางโปรแกรมก็ตัดปัญหาโดยไม่มีการแสดงตารางอยู่ภายในเว็บเพจเลย แต่โปรแกรมนี้อาจตรวจสอบได้ว่าเว็บเพจนี้มีการใช้ตารางในการจัดโครงสร้างหรือไม่ถ้ามีก็จะแยกตารางนั้นออกมาแสดงเป็นหน้าจอปกติ (ไม่มีตาราง) แต่ถ้าตารางนั้นไม่ได้ใช้สำหรับจัดโครงสร้างเว็บเพจ ก็จะนำมาแสดงในรูปแบบของตารางบนหน้าเว็บเพจ โดยมีจัดตารางที่มีขนาดใหญ่ให้สามารถแสดงได้เหมาะสมกับขนาดหน้าจอแสดงผลของเว็บเบราว์เซอร์ที่ใช้งานอยู่

- มีการแบ่งขนาดของเด็ก

เว็บไซต์ในปัจจุบันส่วนใหญ่มักมีขนาดใหญ่กว่าเว็บไซค์มาก และข้อจำกัดในเรื่องของขนาดของหน่วยความจำบนอุปกรณ์เว็บซึ่งมีขนาดเล็ก ทำให้ไม่สามารถเปิดเว็บเพจที่มีขนาดใหญ่ได้ โดยโปรแกรมแปลงภาษา1 จะไม่มีการตรวจสอบความผิดพลาดในส่วนนี้จึงทำให้ไม่สามารถเปิดเว็บไซต์ที่มีขนาดใหญ่ได้ แต่โปรแกรมนี้อาจตรวจสอบได้ว่าขนาดของเด็กของ WML มีขนาดใหญ่เกินไปหรือไม่ถ้ามีขนาดใหญ่เกินกว่าข้อจำกัดของอุปกรณ์นั้นๆ ก็จะทำการแบ่งออกมาเป็นเด็กย่อยๆแล้วเชื่อมโยงกันด้วยลิงก์ที่ชื่อว่า More... เพื่อเปิดไปยังส่วนต่อไป และโปรแกรมนี้อาจกำหนดขนาดของหน่วยความจำได้ขึ้นอยู่กับว่าอุปกรณ์ที่ใช้เปิดมีขนาดของหน่วยความจำเท่าไร

- มีการทำเป็นไฟล์ XHTML

เนื่องจากแนวโน้มของภาษามาร์กอัพในอนาคตจะเปลี่ยนเป็น XHTML รวมถึงเว็บเบราว์เซอร์ใหม่ๆ (เวอร์ชัน 2.0) ก็จะอยู่บนพื้นฐานของภาษา XHTML ดังนั้นโปรแกรมนี้อาจรองรับการเปลี่ยนแปลงนี้ได้ โดยมีการสร้างเป็นไฟล์ XHTML ไว้เพื่อสามารถรองรับได้ทั้งเวอร์ชันใหม่และเวอร์ชันเก่า

6.3 ข้อเสีย

ข้อเสียที่สำคัญที่จำกัดการทำงานของโปรแกรมนี้ คือ

- โปรแกรมมีขนาดใหญ่
เนื่องจากขั้นตอนในการแปลงมีหลายขั้นตอนทำให้โปรแกรมมีขนาดใหญ่ และใช้ resource มาก ดังนั้นถ้ามีจำนวนผู้ใช้งานมากๆ จำเป็นต้องมีการเพิ่มประสิทธิภาพของฮาร์ดแวร์เพื่อรองรับการประมวลผลที่มากเหล่านี้
- ไม่สามารถกรองข้อมูลที่ไม่มีความจำเป็นออก
โปรแกรมนี้ไม่มีความสามารถในการกรองเอาข้อมูลที่ไม่จำเป็นออกโดยไม่นำมาแสดงได้ ทำให้บางครั้งจำนวนเคิลที่แบ่งมีมาก ซึ่งข้อมูลบางอย่างสามารถเอาออกได้โดยไม่ทำให้เนื้อหาภายในเว็บต้นฉบับเปลี่ยนไป และทำให้จำนวนเคิลน้อยลงง่ายต่อการใช้งาน
- มีการเขียนไฟล์ WML ไว้
เนื่องจากความต้องการที่จะเก็บไฟล์ WML ไว้เพื่อประโยชน์ดังที่ได้กล่าวมาแล้ว ทำให้ต้องใช้เนื้อที่ส่วนหนึ่งในการเก็บไฟล์ WML ไว้ แต่ถ้าไม่ต้องการใช้อีกก็สามารถลบทิ้งได้หลังจบการใช้งานแล้ว โดยโปรแกรมจะเขียนขึ้นมาใหม่อีกเมื่อมีการเรียกเว็บเพจนั้นอีกครั้งทำให้ได้ข้อมูลที่เป็นปัจจุบันเสมอ

6.4 ข้อจำกัด

- ถ้าต้องการเอาโค้ดชุดในรูปแบบที่แตกต่างออกไป จะต้องทำการเขียนโปรแกรมในฟังก์ชันส่วนที่แปลงนั้นใหม่ และต้องนำไปแทนที่ของเดิมที่มีอยู่ และการทำงานนั้นไม่กระทบการทำงานโดยรวมของโปรแกรม
- ไฟล์ HTML ต้องมีรูปแบบตามที่กำหนดในโปรแกรมเท่านั้น โดยจะต้องมีแท็กปิดเสมอ และต้องใช้ '<' แทน '<' และ '>' แทน '>' เสมอในส่วนของเนื้อข้อมูลที่แสดงผล (ส่วนที่ไม่ใช่แท็ก)

6.5 แนวทางการพัฒนาต่อไป

ข้อเสนอแนะในการพัฒนาเครื่องมือแปลงภาษานี้ต่อไป

การเพิ่มความสามารถในการทำงาน (Features)

- เพิ่มความสามารถในการกรองข้อมูล
ข้อมูลบางอย่างในเว็บเพจสามารถตัดทิ้งไม่นำมาแสดงบนเว็บเพจได้ เช่น ลิงก์ที่ซ้ำกัน, รูปภาพบางรูปที่ไม่จำเป็น, ข้อความที่ซ้ำซ้อน เป็นต้น โดยอาจต้องมีการใช้การวิเคราะห์รูปภาพ (Image processing) เพื่อหารูปที่ไม่มีความจำเป็นที่จะต้องนำมาเสนอบนเว็บ
- เพิ่มความสามารถในการแปลงไฟล์ HTML ที่ไม่ well-form ได้

เพื่อเพิ่มความสามารถในการแปลง HTML ที่มีโครงสร้างที่ไม่อยู่ในรูปแบบที่ถูกต้องได้ โดยอาจใช้ Tidy จาก W3C หรือ Jidy จากกรอิมพิลเมนต์จาวา ซึ่งเป็นเครื่องมือที่ช่วยเพิ่มแท็กที่ขาดหายไป, จัดการเกี่ยวกับแท็กที่ซ้อนกันอยู่ และอื่นๆที่จะทำให้เป็น well-form HTML (HTML ที่มีโครงสร้างที่ถูกต้อง) ที่สุด แม้ว่า Jidy จะไม่สามารถแก้ไขได้ทุกข้อผิดพลาด แต่สามารถแจ้งปัญหาและข้อผิดพลาดระหว่างการแปลงได้ ดังนั้นเราจึงสามารถเขียนหาวิธีจัดการกับพวกข้อผิดพลาดเหล่านี้ได้ โดยอาจใช้ Jidy ร่วมกับตัวพาร์เซอร์ HTML อื่นๆ จะช่วยให้การทำงานของโปรแกรมถูกต้องยิ่งขึ้น

- การกรอกรฟอร์ม

เนื่องจากบนเว็บเบราว์เซอร์มีความยุ่งยากในการกรอกข้อมูล ดังนั้นจึงควรลดการที่จะให้ผู้ใช้งานกรอกข้อมูลบนเว็บเบราว์เซอร์นั้น เช่น ในการค้นหากำหนดการเดินทางเครื่องบิน ซึ่งบนเว็บเพจต้องกรอกข้อมูลจำนวน 10 ช่อง ซึ่งอาจทำให้ผู้ใช้งานมีความยุ่งยากในการพิมพ์ผิดพลาด จึงควรวางวิธีที่จะลดจำนวนช่องกรอกข้อมูลลงแต่ยังสามารถใช้งานได้อยู่

- การเข้ารหัสข้อมูล

การสนับสนุน SSL โดยการเข้ารหัส, เลือกเอาบางข้อความที่เหมาะสม, ถอดรหัส เพื่อรักษาความปลอดภัยให้กับข้อมูลของผู้ใช้

- ความสามารถในการใช้คุกกี้ (cookie)

เพื่อสนับสนุนการเชื่อมโยง (navigation) และการจัดการเซสชัน (session) เนื่องจากการสนับสนุนคุกกี้ยังไม่แพร่หลายนัก ดังนั้นจึงไม่เป็นเรื่องง่ายสำหรับเว็บเบราว์เซอร์ในปัจจุบัน แต่อาจทำได้โดยอิมพิลเมนต์การทำงานของคุกกี้เอง เพื่อให้ได้เว็บไซต์ที่มีการทำงานถูกต้องยิ่งขึ้น

ความยืดหยุ่น

ความยืดหยุ่นในที่นี้หมายถึงความง่ายในการแปลงเว็บไซต์ใหม่ๆ โดยไม่ต้องยุ่งเกี่ยวกับการเปลี่ยนแปลงโค้ดจาวาโดยรวมเลย โดยจะเปลี่ยนเฉพาะฟังก์ชันที่ใช้แปลงแท็กนั้นเท่านั้น และสามารถเพิ่มความสามารถในการแปลงแท็กอื่นๆเมื่อเว็บเบราว์เซอร์นั้นสามารถรองรับการทำงานของแท็กนั้นได้แล้ว และสามารถตัดการทำงานของแท็กที่ไม่มีการใช้งานแล้วได้

ภาคผนวก ก.

ก.1 แท็ก HTML

<HTML>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อมูลที่น่าเสนอด้วยภาษา HTML

รูปแบบ <HTML> ... </HTML>

แอตทริบิวต์ ไม่มี

แท็กที่เกี่ยวข้อง ตามข้อกำหนด HTML 4.0 แล้ว ไฟล์ HTML ทุกไฟล์มีแท็ก <HTML> อยู่บนบรรทัดแรกของไฟล์ แต่ในบางครั้งก็อาจมีแท็ก <!DOCTYPE> วางไว้ก่อนแท็ก <HTML> ก็ได้ ซึ่งแท็กนี้จะทำหน้าที่ระบุว่าไฟล์นี้ใช้แท็กที่เป็นไปตามข้อกำหนดของ HTML ในเวอร์ชันใด

<HEAD>

ประเภท มีแท็กเปิด

หน้าที่ ใช้ครอบข้อความ, แท็ก และแอตทริบิวต์ที่อยู่ในบริเวณส่วนหัวเอกสาร

รูปแบบ <HEAD> ... </HEAD>

แอตทริบิวต์ ไม่มี

แท็กที่เกี่ยวข้อง สำหรับแท็กที่ใช้ระหว่างแท็ก <HEAD> และ </HEAD> นี้สามารถใช้ได้เพียงบางแท็กเท่านั้น ดังนี้คือ <BASE>, <ISINDEX>, <LINK>, <META>, <SCRIPT>, <STYLE> และ <TITLE>

<BASE>

ประเภท ไม่มีแท็กปิด

หน้าที่ ใช้กำหนด URL กลางหรือชื่อแฟ้มกลาง ซึ่งมีค่า URL กลางนี้จะถูกเรียกใช้เมื่อมีแท็กใดๆที่ใช้แอตทริบิวต์ HREF และกำหนดค่า URL ในแอตทริบิวต์ HREF เป็นรูปแบบย่อ ทำให้ไม่ต้องกำหนดแบบเต็มรูปทุกครั้งในแอตทริบิวต์นี้ ส่วนชื่อแฟ้มกลางจะถูกใช้เมื่อมีลิงก์ที่ไม่ได้กำหนดค่าใดๆ ให้กับแอตทริบิวต์ TARGET โดยเว็บเบราว์เซอร์จะถือว่าให้ใช้ชื่อแฟ้มกลางนี้แทนโดยอัตโนมัติ

รูปแบบ <BASE HREF = "base_url ">

หรือ

<BASE TARGET = "frame_name">

แอตทริบิวต์

HREF	ใช้กำหนดค่า URL กลาง
TARGET	ใช้กำหนดชื่อแฟรมกลาง

<ISINDEX>

ประเภท	ไม่มีแท็กปิด
หน้าที่	ใช้สร้างฟิลด์รับข้อความ และเรียกใช้โปรแกรมค้นหาข้อมูลตามข้อความที่ผู้ใช้กรอกเข้ามา ปัจจุบันไม่นิยมใช้กันแล้ว เพราะสามารถใช้ฟอร์มทำงานแทนได้
รูปแบบ	<code><ISINDEX PROMPT = "Please enter the value to search for."></code>
แอตทริบิวต์	
PROMPT	ใช้กำหนดข้อความที่จะปรากฏบนเว็บเพจ พร้อมทั้งฟิลด์รับข้อความที่จะถูกป้อนเข้ามาโดนนักเล่นเว็บ

<META>

ประเภท	ไม่มีแท็กปิด
หน้าที่	แท็กนี้ใช้ทำงานได้หลายอย่าง เช่น ใช้ใส่ข้อมูลเพิ่มเติมให้เว็บเพจ ซึ่งข้อมูลเหล่านี้จะไม่ปรากฏบนเว็บเพจ โดยส่วนมากมักจะใช้เตือนความจำเท่านั้น เช่น ใส่คำสำคัญ(keyword)ของเว็บเพจนี้ ใส่ชื่อผู้เขียนหรือซอฟต์แวร์ที่ใช้สร้างเว็บเพจ ซึ่งโปรแกรมค้นหา (search engine) บางตัวก็ใช้คำสำคัญนี้ในการค้นหาเว็บเพจตามคำสำคัญที่ผู้เล่นเว็บระบุอีกด้วย นอกจากนั้นแท็กนี้ยังสามารถส่งข้อมูลเพื่อติดต่อกับเว็บเซิร์ฟเวอร์ได้ เช่น สั่งให้เว็บเซิร์ฟเวอร์ส่งไฟล์ HTML ตัวใหม่มาเองโดยอัตโนมัติภายในเวลาที่กำหนด เป็นต้น
รูปแบบ	<code><META HTTP-EQUIV="header" CONTENT="value"></code> หรือ <code><META NAME="name" CONTENT="value"></code>
แอตทริบิวต์	
HTTP-EQUIV	ใช้ระบุประเภทของheader ที่จะส่งไปพร้อมไฟล์เอกสาร ซึ่งค่าของ headerนี้จะเก็บไว้ในแอตทริบิวต์ CONTENT สำหรับค่าของHTTP-EQUIV ที่ใช้กันบ่อยๆก็คือ REFRESH ที่ใช้สั่งให้มีการปรับปรุงเว็บเพจบนหน้าจอใหม่ภายในเวลาที่ถูกกำหนดไว้ในแอตทริบิวต์ CONTENT นอกจากนี้ EXPIRES ก็เป็นอีกค่าหนึ่งที่ถูกใช้บ่อย ซึ่งจะใช้บอกให้ทราบว่าข้อมูลต่างๆที่อยู่ในเว็บเพจเชื่อถือได้ถึงวันที่ที่ระบุไว้เท่านั้น
NAME	กำหนดชื่อหรือข้อมูลให้กับตัวแปร meta (meta-variable) ดังนี้ AUTHOR, KEYWORDS, GENERATOR และ DESCRIPTION โดเฉพาะค่าที่กำหนดให้

กับ KEYWORDS จะถูกโปรแกรมค้นหา (search engine)ใช้ในการค้นว่าตรงตามที่ผู้ใช้ระบุมาหรือไม่

SCHEME กำหนดรูปแบบของข้อมูลเพื่อตีความว่าข้อมูลนั้นต้องอ่านว่าอย่างไร เช่น

<META SCHEME= “9-digit-Zipcod” NAME= “zip”

CONTENT= “02134-1078”>

CONTENT สามารถใช้งานได้ทั้ง 2 แบบคือ ใช้กำหนดค่าให้ข้อมูลเพิ่มเติม และกำหนดค่าที่จะส่งให้เว็บเซิร์ฟเวอร์

<LINK>

ประเภท ไม่มีแท็กปิด

หน้าที่ ใช้อ้างอิง หรืออ้างถึงข้อมูลในไฟล์อื่น

รูปแบบ <LINK HREF= “url_of_linked_file” TITLE= “title”
REL= “forward_relationship” REV= “reverse_relationship”>

แอตทริบิวต์

HREF ใช้อ้างอิงไฟล์ โดยกำหนดชื่อไฟล์แบบURL

TITLE ใช้กำหนดข้อความสั้นๆที่เป็นคำอธิบายไฟล์ที่อ้างอิง

REL ใช้อธิบายความสัมพันธ์ของไฟล์อ้างอิงกับไฟล์ปัจจุบันมักมีค่าเป็นไปได้หลายค่า

REV ใช้อธิบายความสัมพันธ์ของไฟล์ปัจจุบันกับไฟล์ที่อ้างอิง มักมีค่าตรงกันข้ามกับค่าของแอตทริบิวต์ REL มีค่าเป็นไปได้เหมือนแอตทริบิวต์ REL

<SCRIPT>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่เป็นสคริปต์ในไฟล์ HTML

รูปแบบ <SCRIPT LANGUAGE= “scripting_language”>

... script code goes here ...

</SCRIPT>

แอตทริบิวต์

DEFER ใช้เพื่อบอกให้ทราบเบราว์เซอร์รู้ว่าสคริปต์ชุดนี้ไม่มีการแสดงข้อมูลลงบนเว็บเพจเลย ดังนั้นระหว่างที่เว็บเบราว์เซอร์กำลังแปลไฟล์ HTML เป็นเว็บเพจอยู่นั้นเบราว์เซอร์ไม่มีความจำเป็นต้องอ่านสคริปต์นี้เลย วิธีนี้ช่วยทำให้เบราว์เซอร์แสดงเว็บเพจได้เร็วขึ้น

LANGUAGE ใช้กำหนดภาษาสคริปต์ที่ใช้ เช่น VBScript หรือ JavaScript

SRC ใช้อ้างอิงตำแหน่งของไฟล์ที่เก็บสคริปต์ไว้ โดยต้องการกำหนดในรูปของ URL

TYPE ใช้กำหนดชนิดของสคริปต์มาตรฐาน MIME เช่น text/javascript หรือ text/vbscript

เพิ่มเติม นักเขียนเว็บมักใช้ <!-- หรือ --> ครอบส่วนที่เป็นสคริปต์ไว้ทั้งหมด เพื่อใช้กรณีเว็บเบราว์เซอร์ไม่สามารถแปลสคริปต์ได้ ซึ่งเว็บเบราว์เซอร์จะเข้าใจว่าส่วนที่เป็นสคริปต์นั้นเป็นคำอธิบายธรรมดา (comment) และไม่แสดงส่วนที่เป็นสคริปต์ในเว็บเพจ

<STYLE>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบส่วนที่เป็นข้อกำหนดสไตล์ชีท ซึ่งเป็นวิธีกำหนดลักษณะแท็กใหม่ และมีข้อดีกว่าวิธีกำหนดด้วยแอตทริบิวต์และแท็กแบบเดิมหลายประการ ดังนั้น W3C จึงพยายามผลักดันให้นักเขียนเว็บเพจหันมาใช้วิธีนี้แทนแอตทริบิวต์ และแท็กเดิม

รูปแบบ `<STYLE TYPE="mime_type" MEDIA="media_type" TITLE="title">`
 ...style information goes here ...
`</STYLE>`

แอตทริบิวต์

MEDIA ใช้กำหนดชนิดของอุปกรณ์หรือสื่อที่ใช้แสดงซึ่งเหมาะสำหรับใช้กับข้อกำหนดสไตล์ชีทชุดนี้

TITLE ใช้กำหนดข้อความสั้นๆ เพื่ออธิบายสไตล์ชีทชุดนี้

TYPE ใช้กำหนดชนิดของข้อมูลภายใต้แท็กนี้ สำหรับสไตล์ชีทแล้วมักจะเป็น text/css1

<TITLE>

ประเภท มีแท็กปิด

หน้าที่ ใช้กำหนดชื่อหรือหัวเรื่องให้กับไฟล์ HTML ซึ่งมักจะแสดงไว้ที่ไคเลบาร์ หรือที่ขอบด้านล่างของเว็บเพจนั้น รวมทั้งรายการของ history หรือ favorites ด้วย และเว็บเบราว์เซอร์จะใช้ชื่อนี้เป็นเฉพาะเพื่ออ้างถึงเว็บเพจ นอกจากนี้ โปรแกรมค้นหาข้อมูล (Search Engine) บางอย่าง เช่น Yahoo หรือ Alta Vista ก็ใช้ข้อมูลส่วนนี้ในการจัดเว็บเพจนั้นๆ ว่าควรจะอยู่ในหมวดหมู่ใด (indexing) สำหรับแท็ก <TITLE> นี้ถือว่ามีค่าสำคัญต่อเว็บเพจมาก ดังนั้น HTML 4.0 จึงกำหนดไว้ว่าไฟล์ HTML ทุกไฟล์ต้องมีแท็ก <TITLE>

รูปแบบ `<TITLE> ...document title goes here ...</TITLE>`

แอตทริบิวต์ ไม่มี

<BODY>

ประเภท มีแท็กปิด

หน้าที่	ใช้ครอบข้อมูลและแท็กต่างๆซึ่งจะใช้แสดงเป็นเว็บเพจบนจอตนเอง
รูปแบบ	<pre> <BODY SGCOLOR="background_color" BACKGROUND="background_image" LIKE="unvisited_like_color" ALIKE="active_like_color" VLIKE="visited_like_color" TEXT="text_color"> ...document body goes here ... </BODY> </pre>
แอตทริบิวต์	สำหรับแท็ก <BODY> นี้มักจะใช้แอตทริบิวต์ด้านล่างนี้ในการกำหนดสีหรือพื้นหลัง ซึ่งเกี่ยวข้องกับแอตทริบิวต์เหล่านี้มี 13 สีมาตรฐาน คือ AQUA, BLACK, BLUE, FUCHSIA, GRAY, GREEN, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE และ YELLOW ALIKE ใช้กำหนดตำแหน่งข้อความที่เป็นจุดลิงค์ขณะถูกเรียกใช้งาน BACKGROUND ใช้กำหนดตำแหน่งและชื่อไฟล์รูปที่จะนำมาใช้เป็นภาพพื้นของเว็บเพจ โดยกำหนดชื่อตามแบบ URL ในกรณีที่รูปนั้นมีขนาดเล็กกว่าเว็บเพจ ก็จะนำรูปเหล่านั้นมาเรียงต่อกันในลักษณะเหมือนกับปูกระเบื้อง BGCOLOR กำหนดสีพื้น อาจจะกำหนดเป็นชื่อสี หรือค่า RGB อย่างใดอย่างหนึ่งก็ได้ LIKE กำหนดสีของข้อความที่เป็นจุดลิงค์ที่ยังไม่เคยถูกเรียกใช้งานมาก่อน TEXT กำหนดสีของข้อความทั่วไปที่อยู่ภายใต้กรอบของแท็ก <BODY> VLIKE ใช้กำหนดสีของข้อความที่เป็นจุดลิงค์ที่เคยถูกเรียกใช้งานมาแล้ว
หมายเหตุ	แอตทริบิวต์ทั้งหมดนี้กำลังจะถูกยกเลิกในเวอร์ชันถัดไป เพราะข้อกำหนดสโตนีทสามารถทำงานแบบเดียวกันนี้ได้ครบ
แท็กที่เกี่ยวข้อง	สำหรับแท็กต่างๆส่วนใหญ่สามารถใช้ระหว่างแท็ก <BODY> และ</BODY> ได้ ยกเว้นแท็กที่เกี่ยวข้องกับแท็ก Frame
	
ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความเพื่อให้ข้อความเป็นตัวหนา
รูปแบบ	...bold text goes here...
แอตทริบิวต์	ไม่มี
<BASEFONT>	
ประเภท	ไม่มีแท็กปิด

หน้าที่	ใช้กำหนด สี ขนาด และคุณสมบัติพื้นฐานอื่นๆของข้อความที่เป็นข้อมูลภายในตัวเอกสาร ซึ่งจะมีผลกับข้อความในแท็กต่างๆที่ไม่ใช่แท็กที่กำหนดคุณสมบัติของข้อความ เช่น แท็ก เป็นต้น
รูปแบบ	<BASEFONT SIZE= "size" COLOR= "color" FACE= "list_of_typefaces">
แอตทริบิวต์	
COLOR	ใช้กำหนดสีของข้อความ โดยอาจจะกำหนดเป็นชื่อสี หรือค่า RGB อย่างใดอย่างหนึ่งก็ได้ ซึ่งสีที่เป็นค่ามาตรฐานคือสีดำ
FACE	ใช้กำหนดฟอนต์ โคนสามารถกำหนดครั้งละหลายๆฟอนต์ แต่เว็บเบราว์เซอร์จะดูฟอนต์แรกที่กำหนด หากไม่สามารถแสดงฟอนต์แรกได้ ก็จะเลื่อนลำดับไปใช้ฟอนต์ที่ 2,3 แทนตามลำดับ
SIZE	ใช้กำหนดขนาดของข้อความ
แท็กที่เกี่ยวข้อง	 ที่ใช้มักเปลี่ยนคุณสมบัติต่างๆที่ถูกกำหนดโดยแท็ก <BASEFORNT> อีกทีหนึ่ง
<BIG>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้เพื่อบอกข้อความที่ต้องการให้มีขนาดใหญ่กว่าข้อความขนาดมาตรฐาน
รูปแบบ	<BIG>...big text goes here ...</BIG>
แอตทริบิวต์	ไม่มี
แท็กที่เกี่ยวข้อง	<SMALL>
	
ประเภท	มีแท็กปิด
หน้าที่	ใช้กำหนดสี ขนาด หรือฟอนต์ใหม่ให้กับข้อความที่ถูกแท็กนี้ครอบไว้ เพื่อให้ต่างจากคุณสมบัติพื้นฐานที่ถูกกำหนดโดยแท็ก <BASEFONT>
รูปแบบ	 ...text with modified font properties...
แอตทริบิวต์	จะเหมือนกับแอตทริบิวต์ที่ใช้กับแท็ก <BASEFONT> แต่สามารถกำหนดค่าที่ให้แอตทริบิวต์ SIZE เป็นค่าสัมพัทธ์ได้ เช่น +1 คือใหญ่กว่าค่าที่กำหนดไว้ด้วย <BASEFONT> หนึ่งระดับ และ -2 คือเล็กกว่าค่าที่กำหนดไว้ด้วย <BASEFONT> สองระดับ เป็นต้น
แท็กที่เกี่ยวข้อง	<BASEFONT>

<I>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการเน้นให้เป็นตัวเอียง
รูปแบบ	<i>...italicized text goes here ...</i>
แอตทริบิวต์	ไม่มี

<S>,<STRIKE>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการให้มีการใช้มีเส้นพาดกลางตลอดแนวข้อความนั้น
รูปแบบ	<S>...strikerthrough text goes here ...</S>
แอตทริบิวต์	ไม่มี

<SMALL>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการให้มีขนาดเล็กกว่าปกติ 1 ระดับ
รูปแบบ	<SMALL>...smaller text goes here ...</SMALL>
แอตทริบิวต์	ไม่มี

<SUB>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการให้มีขนาดเล็กและอยู่ต่ำกว่าระดับปกติ
รูปแบบ	_{...subscript text goes here ...}
แอตทริบิวต์	ไม่มี

<SUP>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการให้ข้อความขนาดและอยู่สูงกว่าระดับปกติ
รูปแบบ	^{...superscript text goes here ...}
แอตทริบิวต์	ไม่มี

<TT>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการทำให้ใช้ฟอนต์แบบที่มีขนาดตัวอักษรเท่ากันทุกตัว

รูปแบบ <TT>...text to be in fixed-width font goes here ...</TT>
 แอตทริบิวต์ ไม่มี

<U>

ประเภท มีแท็กปิด
 หน้าที่ ใช้ครอบข้อความที่ต้องการให้มีเส้นใต้ข้อความนั้น
 รูปแบบ <U>...text to be underlined...</U>
 แอตทริบิวต์ ไม่มี

<ACRONYM>

ประเภท มีแท็กปิด
 หน้าที่ ใช้ครอบข้อความที่ต้องการแสดงคำย่อที่จะปรากฏอยู่ในเนื้อข้อความของเว็บเพจ
 เช่น World Wide Web <ACRONYM> (WWW) </ACRONYM>
 รูปแบบ <ACRONYM> ... acronym goes here ...</ACRONYM>
 แอตทริบิวต์ ไม่มี

<ADDRESS>

ประเภท มีแท็กปิด
 หน้าที่ ส่วนมากจะใช้ครอบข้อความที่เป็นอีเมลแอดเดรส ซึ่งอาจเป็นแอดเดรสของผู้เขียนเว็บเพจ โดยเว็บเบราว์เซอร์จะแสดงข้อความชนิดนี้เป็นตัวเอียง
 รูปแบบ <ADDRESS>...address goes here...</ADDRESS>
 แอตทริบิวต์ ไม่มี

<CITE>

ประเภท มีแท็กปิด
 หน้าที่ ใช้ครอบข้อความที่ต้องการสื่อว่าเป็นคำที่ขี้มมาใช้ หรือเอกสารอ้างอิงมักจะเป็นตัวเอียง
 รูปแบบ <CITE> ...citation source goes here ...</CITE>
 แอตทริบิวต์ ไม่มี

<CODE>

ประเภท มีแท็กปิด

หน้าที่	ใช้ครอบข้อความที่เป็นโปรแกรม , source code หรือสคริปต์ โดยเว็บเบราว์เซอร์ จะแสดงข้อความด้วยฟอนต์ที่มีขนาดของตัวอักษรเท่ากัน
รูปแบบ	<CODE>...code fragment goes here ...</CODE>
แอตทริบิวต์	ไม่มี

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการแสดงให้เห็นว่าถูกลบออก ซึ่งเว็บเบราว์เซอร์จะแสดง
ข้อความนั้นพร้อมทั้งมีเส้นขีดฆ่าพาดอยู่ตรงกลาง ทั้งนี้เพื่อให้เห็นว่าเว็บเพจนั้น
มีการแก้ไขตรงตำแหน่งไหนบ้าง เพราะบางครั้งผู้ที่แก้ไขเว็บกับผู้ที่สร้างเป็น
คนละคนกัน และต้องการแสดงให้เห็นตำแหน่งที่มีการแก้ไข

รูปแบบ

...deleted text goes here ...

แอตทริบิวต์

ไม่มี

แท็กที่เกี่ยวข้อง

<INS>

<DFN>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการเน้นว่ามีความหมายพิเศษ และอธิบาย

รูปแบบ

<DFN>...term being introduced goes here ...</DFN>

แอตทริบิวต์

ไม่มี

<INS>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการแสดงให้เห็นว่าถูกเพิ่มเติมเข้าไป มีความหมายตรงข้าม
กับแท็ก โดยเว็บเบราว์เซอร์แสดงข้อความทั้งมีเส้นขีดอยู่ได้ข้อความนั้น

รูปแบบ

<INS>...inserted text goes here ...</INS>

แอตทริบิวต์

ไม่มี

แท็กที่เกี่ยวข้อง

<KBD>

ประเภท

มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่ต้องการแสดงให้เห็นว่าเป็นสิ่งที่ป้อนเข้ามาผ่านทางคีย์บอร์ด โดยบราวเซอร์จะแสดงข้อความในแท็กนี้ด้วยฟอนต์ที่มีขนาดตัวอักษรขนาดใหญ่เท่าตัวปกติ

รูปแบบ <KBD>...keyboard input goes here ...</KBD>

แอตทริบิวต์ ไม่มี

<Q>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่อ้างมาจากแหล่งอื่น โดยบราวเซอร์จะแสดงไว้ในเครื่องหมายคำพูด

รูปแบบ <Q CITE="URL_of_cited_document">...quotation goes here ...</Q>

แอตทริบิวต์

CITE อ้างถึงแหล่งต้นต่อหรือเว็บเพจของคำพูดนั้น โดยกำหนดเป็นค่า URL

แท็กที่เกี่ยวข้อง <BLOCKQUOTE>

<SAMP>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่เป็นตัวอย่าง โปรแกรมหรือสคริปต์ โดยเว็บเบราว์เซอร์มักแสดงข้อความนี้ด้วยฟอนต์ที่มีขนาดตัวอักษรเท่ากันทุกตัว

รูปแบบ <SAMP>...program output goes here ...</SAMP>

แอตทริบิวต์ ไม่มี

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่ต้องการเน้นเป็นพิเศษ โดยบราวเซอร์จะแสดงเป็นตัวหนา

รูปแบบ ...strongly emphasized text goes here ...

แอตทริบิวต์ ไม่มี

<VAR>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่เป็นชื่อตัวแปรภายในโปรแกรม โดยบราวเซอร์มักแสดงข้อความในแท็กนี้ด้วยฟอนต์ที่มีขนาดตัวอักษรเท่ากับทุกตัว

รูปแบบ <VAR> ... program variable goes here ...</VAR>

แอตทริบิวต์ ไม่มี

<BLOCKQUOTE>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่เป็นคำพูด โดยแยกข้อความนั้นออกจากข้อความส่วนอื่นแล้ว
นำมาขึ้นย่อหน้าใหม่ ซึ่งต่างจากการใช้แท็ก <Q>

รูปแบบ <BLOCKQUOTE CITE="URL_of_cited_document">
...quoted text goes here ...
</BLOCKQUOTE>

แอตทริบิวต์

CITE

ถ้าต้องการใส่ข้อความเพื่ออ้างถึงที่มาของข้อความนั้น สามารถทำได้โดยใช้
แอตทริบิวต์ CITE และกำหนด URL ที่เป็นที่อยู่ของข้อมูลนั้น

แท็กที่เกี่ยวข้อง

<Q>

ประเภท

ไม่มีแท็กปิด

หน้าที่

กำหนดให้บรรทัดขึ้นบรรทัดใหม่ เพราะโดยปกติแล้วบรรทัดจะแสดงข้อความเรียงจากซ้ายไปขวาและเรียงจากบนลงล่าง โดยไม่สนใจว่าข้อความภายในไฟล์ HTML นั้นมีการจกย่อหน้า หรือขึ้นบรรทัดใหม่ไว้อย่างไร (คือไม่ตีความ Carriage Return ที่อยู่ข้อความนั้นเลย) ดังนั้นถ้าต้องการกำหนดให้มีการขึ้นบรรทัดใหม่ ผู้สร้างเว็บเพจต้องใส่เองโดยใช้แท็ก

รูปแบบ

<BR CKEAR="LEFT|RIGHT|ALL|NONE">

แอตทริบิวต์

CLEAR

เป็นแอตทริบิวต์ที่ใช้เพื่อระบุตำแหน่งของข้อความที่อยู่ถัดจากแท็ก
 เช่น
ภายใต้เว็บเพจหน้าหนึ่งมีรูปที่อยู่ติดด้านซ้าย ข้อความที่ 1 แท็ก <BR CLEAR=
"left"> และตามด้วยข้อความที่ 2 ต่อท้าย เมื่ออยู่บนเว็บเพจจะปรากฏข้อความที่
2 ขึ้นบรรทัดใหม่ที่ต่อท้ายรูปที่อยู่ด้านซ้าย โดยค่าที่สามารถกำหนดได้คือ left,
right, all และ none

<CENTER>

ประเภท

มีแท็กปิด

หน้าที่

ครอบข้อความหรือรูปภาพที่ต้องการให้อยู่ตรงกลางหน้าเว็บเพจ

รูปแบบ

<CENTER>...centered page components goes here ...</CENTER>

แอตทริบิวต์ ไม่มี
 หมายเหตุ แท็กนี้ W3C ถือให้เป็นแท็กที่จะถูกยกเลิกไปในอนาคต โดยจะใช้แท็ก <DIV>
 แทน

<DIV>

ประเภท มีแท็กปิด
 หน้าที่ ใช้กำหนดกลุ่มของข้อความที่ หรือจัดแบ่งข้อความเป็นส่วน เพื่อช่วยให้สะดวก
 ต่อการจัดข้อความในกลุ่มหรือในส่วนนั้น

รูปแบบ <DIV SLIGN= "LEFT|RIGHT|CENTER|JUSTIFY">...</DIV>

แอตทริบิวต์

ALIGN ใช้จัดวางตำแหน่งของข้อความว่าน่าจะอยู่ชิดขอบด้านใด มีค่าที่เป็นไปได้คือ left
 , right , center และ justify

<HR>

ประเภท มีแท็กปิด
 หน้าที่ สร้างเส้นคั่นในแนวนอน เพื่อแบ่งข้อมูลออกเป็นส่วนๆ อย่างชัดเจน

รูปแบบ <HR ALIGN= "alignment" NOSHADE SIZE= "thickness"
 WIDTH= "pixels_or_percentage_of_screen">

แอตทริบิวต์

ALIGN กำหนดตำแหน่งเส้นคั่นว่าจะให้ชิดขอบเส้นใด ซึ่งจะมีผลก็ต่อเมื่อความยาวของ
 เส้นขอบนั้นสั้นกว่าความยาวของหน้าจอ มีค่าเป็นไปได้อีกคือ left, right และ
 center

NOSHADE กำหนดให้เส้นคั่นเป็นแบบเส้นทึบ

SIZE กำหนดความหนาของเส้นคั่น โดยกำหนดหน่วยเป็นพิกเซล

WIDTH กำหนดความยาวของเส้นคั่น โดยกำหนดหน่วยเป็นพิกเซลหรือเป็นเปอร์เซ็นต์
 ของขนาดหน้าจอ

<H1>-<H6>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่เป็นหัวข้อในระดับต่างกัน ตัวอย่างเช่น <H1> ใช้ครอบหัวข้อ
 ที่มีความสำคัญที่สุด และ<H2> ใช้ครอบหัวข้อที่มีความสำคัญรองลงมา
 โดยส่วนมากแล้วบรรทัดจะจัดข้อความที่เป็นหัวข้อเหล่านี้ให้มีขนาดตัวอักษร
 ใหญ่และหนากว่าปกติ และเพิ่มที่ว่างเหนือและใต้บรรทัดนั้นให้มากขึ้นด้วย

รูปแบบ <Hn ALIGN= "LEFT|CENTER|JUSTIFY">...Level n heading ...</Hn>

แอตทริบิวต์

ALIGN

จัดวางตำแหน่งข้อความว่าจะให้อยู่ชิดขอบด้านใด มีค่าเป็นไปได้อีกคือ left (ให้ชิดขอบซ้าย), right (ให้ชิดขอบด้านขวา), center (อยู่กลางระหว่างขอบซ้ายและขวา) และ justify (ให้ข้อความกระจายตามความกว้างของขอบทั้งสองด้าน) ถ้าไม่กำหนดไว้จะมีค่าเป็น left

<P>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความเพื่อสร้างย่อหน้าใหม่ ในกรณีที่มีการใช้แท็กนี้หลายๆครั้ง เพื่อเพิ่มบรรทัดว่างบนหน้าจอ เว็บเบราว์เซอร์จะไม่สนใจ ซึ่งต่างจากแท็ก
 เพราะเมื่อใช้แท็ก
 หลายๆตัวจะขึ้นบรรทัดใหม่เท่ากับจำนวนแท็กที่ใส่ไว้

รูปแบบ

<P ALIGN="LEFT|RIGHT|CENTER|JUSTIFY">

... paragraph text ...

</P>

แอตทริบิวต์

ALIGN

กำหนดตำแหน่งที่วางข้อความว่าจะให้ชิดขอบด้านใด มีค่าเป็นไปได้อีกคือ left, right, center และ justify ถ้าไม่กำหนดไว้จะมีค่าเป็น left

<PRE>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการให้มีการจัดรูปแบบ โดยเว็บเบราว์เซอร์จะแสดงข้อความในแท็กนี้ด้วยฟอนต์ที่มีขนาดตัวอักษรเท่ากัน และจัดให้เหมือนกับที่ใส่ไว้ในไฟล์ HTML

รูปแบบ

<PRE WIDTH="width_of_widest_line">

... preformatted text goes here ...

</PRE>

แอตทริบิวต์

ไม่มี

ประเภท

มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่ต้องการกำหนดให้เป็นรูปแบบพิเศษ โดยปกติมักจะใช้กับคำหรือข้อความสั้นๆที่อยู่ในย่อหน้าหนึ่ง

รูปแบบ `<SPAN STYLE="style information">`
range of text over which style is to be applied
`</SPAN>`

แอตทริบิวต์

STYLE ใช้กำหนดรูปแบบที่ต้องการให้มีผลกับข้อความที่อยู่ภายใต้แท็ก `<SPAN>` และ `</SPAN>` เท่านั้น เช่น กำหนดฟอนต์ ขนาด และสีของตัวอักษรที่จะใช้

ประเภท มีแท็กปิด

หน้าที่ ใช้แสดงรายการย่อยแบบมีลำดับ โดยแสดงหนึ่งต่อรายการ

รูปแบบ `<LI TYPE="list_type" START="start_value">`
...list item goes here...
`</LI>`

แอตทริบิวต์

START กำหนดค่าเริ่มต้นของเลขที่กำกับอยู่หน้ารายการนั้น โดยใช้ได้กับรายการย่อยของแท็ก `<OL>` เท่านั้น

TYPE กำหนดลักษณะตัวเลขหรือเครื่องหมายที่กำกับอยู่หน้ารายการย่อย สามารถใช้ได้กับรายการย่อยของแท็ก `<OL>` และ `<UL>` หากเป็นแท็ก `<OL>` มีค่าที่กำหนดได้คือ 1 (ใส่ตัวเลขอารบิกหน้าหน้า), A (ใส่ตัวพิมพ์ใหญ่ภาษาอังกฤษหน้าหน้า), a (ใส่ตัวพิมพ์เล็กภาษาอังกฤษหน้าหน้า), i (ใส่เลขโรมันแบบตัวเล็กหน้าหน้า), I (ใส่เลขโรมันแบบตัวใหญ่นานหน้า) ส่วนแท็ก `<UL>` มีค่าที่กำหนดได้คือ square, circle, disc

แท็กที่เกี่ยวข้อง `<OL>`, `<UL>`, `<DIR>` และ `<MENU>`

<DIR>

ประเภท มีแท็กปิด

หน้าที่ ใช้แสดงรายการย่อยแบบมีจุดหรือสัญลักษณ์กำกับหน้าแต่ละรายการ โดยแต่ละรายการจะมีตัวอักษรได้ไม่เกิน 20 ตัว ซึ่งแท็กนี้แต่เดิมมีวัตถุประสงค์ให้ใช้กับข้อความที่ไม่กว้างเกินไปนัก เช่น ดัชนี หรือหมายเลขโทรศัพท์

รูปแบบ `<DIR COMPACT>`
`<LI>List item 1</LI>`
`<LI>List item 2</LI>`

	...
	<DIR>
แอคทริบิวต์	
COMPACT	ควบคุมรายการย่อในบรรทัดต่างๆให้อยู่กันมากขึ้น โดยลดช่องว่างระหว่างบรรทัดลง
แท็กที่เกี่ยวข้อง	
<DL>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้สร้างข้อมูลรายการแบบมีนิยาม โดยสร้างรายการแบบนี้จะต่างไปจากข้อมูลรายการแบบอื่น คือในแต่ละรายการย่อจะประกอบด้วย 2 ส่วนคือ รายการย่อและรายการละเอียดอธิบายในแต่ละรายการย่อ
รูปแบบ	<DL COMPACT> ... terms and definitions go here ... </DL>
แอคทริบิวต์	
COMPACT	ควบคุมรายการย่อในบรรทัดต่างๆให้อยู่ชิดกันมากขึ้น โดยลดช่องว่างระหว่างบรรทัด
แท็กที่เกี่ยวข้อง	<DD> และ <DT>
<DT>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้กำหนดแต่ละรายการย่อที่อยู่ภายในรายการแบบมีนิยาม (แท็ก <DL>)
รูปแบบ	<DT> ...term being defined goes here ...</DT>
แอคทริบิวต์	ไม่มี
แท็กที่เกี่ยวข้อง	<DT> และ <DL>
<DD>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้แสดงส่วนที่เป็นคำอธิบายของแต่ละรายการย่อที่ต่อจากแท็ก <DT> และอยู่ภายในข้อมูลรายการแบบมีนิยาม (แท็ก <DL>)
รูปแบบ	<DD> ...term definition goes here ...</DD>
แอคทริบิวต์	ไม่มี

แท็กที่เกี่ยวข้อง <DL> และ <DT>

<MENU>

ประเภท มีแท็กปิด

หน้าที่ ใช้แสดงข้อความที่เป็นรายการย่อย โดยมีจุดหรือสัญลักษณ์อื่นกำกับหน้ารายการย่อยแต่ละข้อ โดยข้อความในรายการย่อยมักจะยาวไม่เกิน 20 ตัวอักษร ส่วนมากแล้วผลที่ได้จะเหมือนกับการแท็ก

รูปแบบ

<MENU COMPACT>

Menu list item 1

Menu list item 2

...

</MENU>

ประเภท มีแท็กปิด

หน้าที่ ใช้แสดงรายการย่อยแบบที่มีเลขหรือตัวอักษรกำกับหน้ารายการย่อยโดยเรียงตามลำดับจากน้อยไปมาก และมีค่าเริ่มต้นเป็น 1

รูปแบบ

<OL TYPE="I|A|a|i" START="start_value" COMPACT>

 List item 1

 List item 2

...

แอตทริบิวต์

COMPACT

ควบคุมรายการย่อยในบรรทัดต่างๆให้อยู่ชิดกันมากขึ้น โดยลดช่องว่างระหว่างบรรทัดลง

START

กำหนดค่าเริ่มต้นให้กับเลขหรือตัวอักษรที่กำกับหน้ารายการย่อย โดยปกติจะค่าเป็น 1 ซึ่งคุณสามารถกำหนดค่าเริ่มต้นใหม่เป็นหมายเลขอะไรก็ได้

TYPE

กำหนดชนิดหรือประเภทของค่าที่จะใช้กำกับหน้ารายการย่อย มีค่าเป็นไปได้คือ ตัวเลข (1), ตัวอักษรพิมพ์ใหญ่ (A), ตัวอักษรพิมพ์เล็ก (a), อักษรเลขโรมันตัวเล็ก (i) และ อักษรเลขโรมันตัวใหญ่ (I)

แท็กที่เกี่ยวข้อง

ประเภท

มีแท็กปิด

หน้าที่	ใช้แสดงรายการย่อแบบที่มีจุด หรือสัญลักษณ์กำกับหน้ารายการย่อ
รูปแบบ	<pre> <UL TYPE= "DISC SQUARE CIRCLE" COMPACT> List item 1 List item 2 ... </pre>
แอตทริบิวต์	
COMPACT	ควบคุมรายการย่อในบรรทัดต่างๆให้อยู่ชิดกันมากขึ้น โดยลดช่องว่างระหว่างบรรทัดลง
TYPE	ใช้กำหนดลักษณะของสัญลักษณ์ที่จะใช้กำกับหน้าแต่ละรายการย่อมีค่าเป็นไปได้อันนี้ disc , square และ circle
แท็กที่เกี่ยวข้อง	
<A>	
ประเภท	มีแท็กปิด
หน้าที่	ในกรณีที่ใช้กับแอตทริบิวต์ HREF จะสร้างลิงค์ หรือสร้างจุดเชื่อมต่อที่จะอ้างไปยังเว็บเพจอื่น แต่หากใช้แอตทริบิวต์ NAME จะกำหนดให้จุดนั้นเป็น anchor ที่ลิงค์อื่นมายังจุดนี้ได้
รูปแบบ	<pre> <!--Setting up a hyperlink --> ...hyperlinked element goes here... หรือ <!--Setting up a name anchor --> ...text to act as named anchor... </pre>
แอตทริบิวต์	
ACCESSKEY	ใช้กำหนดคีย์ลัดซึ่งจะให้ผลเหมือนกับการคลิกที่ลิงค์นั้น เช่นหากกำหนดแอตทริบิวต์ ACCESSKEY มีค่าเป็น c ผู้ใช้สามารถกด Alt + c ก็จะไปยังเว็บเพจที่ลิงค์อยู่จุดนั้นได้เลย

HREF	ใช้กำหนดชื่อเว็บเพจหน้าที่ต้องการอ้างถึง โดยต้องใส่ให้ตรงตามรูปแบบของ URL
NAME	กำหนดให้จุดนี้มีชื่อตามที่ระบุ โดยชื่อนี้ไม่ควรซ้ำกับชื่อแท็กหรือลิงค์ตัวอื่น
REL	กำหนดความสัมพันธ์ของเว็บเพจหน้าปัจจุบันกับเว็บเพจที่เชื่อมโยง
REV	ตรงข้ามกับ REL คือกำหนดความสัมพันธ์ของเว็บเพจที่เชื่อมโยงไว้กับเว็บเพจหน้าปัจจุบัน
TABINDEX	กำหนดลำดับของลิงค์ที่เคอร์เซอร์จะกระโดดไปเมื่อผู้ใช้กด Tab แต่ละครั้ง
TARGET	ในกรณีที่หน้าจอมีการแบ่งออกเป็นหลายเฟรม จะเป็นการกำหนดเฟรมที่ต้องการให้แสดงเมื่อถูกคลิก

ประเภท

ไม่มีแท็กปิด

หน้าที่

ใช้วางรูปไว้ในเว็บเพจ โดยไฟล์ภาพอาจเป็น .GIF หรือ .JPEG ก็ได้ ในกรณีที่ต้องการนำรูปมาใช้เป็นอิมเมจแมพ ต้องกำหนดแอตทริบิวต์ ISMAP หรือ USEMAP อย่างใดอย่างหนึ่งด้วยเสมอ

รูปแบบ

```
<IMG SRC="URL_of_image_file"
      WIDTH="width_in_pixels" HEIGHT="height_in_pixels"
      ALT="text_description" BORDER="thickness_in_pixels"
      ALIGN="TOP|MIDDLE|LEFT|RIGHT"
      HSPACE="horizontal_Spacing_in_pixels"
      ISMAP USEMAP="map_name">
```

แอตทริบิวต์

SRC

กำหนดตำแหน่งและชื่อไฟล์รูป โดยกำหนดชื่อตาม URL WIDTH และ HEIGHT กำหนดความกว้างและความสูงของรูปโดยมีหน่วยเป็นพิกเซล ซึ่งการกำหนดแบบนี้ทำให้เบราว์เซอร์สามารถกำหนดพื้นที่ให้ภาพนั้น พร้อมกับจัดวางข้อความอื่นบนเว็บเพจไปได้เลย โดยไฟล์รูปนั้นก็ยังคงถูกทอยทอดาวน์โหลดเข้ามา

ALT

กำหนดข้อความที่จะแสดงบริเวณรูปหรือโปรแกรม plug-in ต่างๆ ซึ่งข้อความเหล่านี้จะปรากฏเมื่อเบราว์เซอร์ยังโหลดรูปหรือโปรแกรมไม่เสร็จ หรือหากเว็บเบราว์เซอร์ที่ใช้ไม่สามารถแสดงกราฟิกได้ เช่น Lynx ก็จะแสดงข้อความนี้แทน

BORDER

กำหนดความหนาของกรอบรูป แต่โดยปกติรูปจะไม่ใส่กรอบไว้ให้ ยกเว้นแต่รูปที่เป็นไฮเปอร์ลิงค์จะถูกใส่กรอบที่มีสีไว้ด้วย ซึ่งสามารถจะยกเลิกโดยการกำหนดให้ BORDER=0 แทน

ALIGN	กำหนดตำแหน่งที่ใช้แสดงข้อความที่ตามหลังรูป โดยหากกำหนดเป็น Top, Middle หรือ Bottom ข้อความจะถูกวางไว้เสมอกับด้านบนบนของรูป, อยู่กึ่งกลางระหว่างความสูงของรูป และอยู่เสมอกับด้านล่างของรูปตามลำดับ โดยข้อความในบรรทัดถัดไปจะแสดงอยู่ใต้รูป แต่หากกำหนดเป็นแบบ left หรือ right ข้อความทั้งหมดจะอยู่ล้อมรูปทางด้านซ้ายและทางขวาตามลำดับ
HSPACE และ VSPACE	กำหนดระยะห่างระหว่างข้อมูลกับรูป หรือเป็นการกำหนดพื้นที่ว่างรอบรูปนั่นเอง โดยมีหน่วยเป็นพิกเซล สำหรับHSPACE จะใช้กำหนดพื้นที่ว่างที่อยู่ทางด้านซ้ายและขวาของรูป ส่วน VSPACE ใช้กำหนดพื้นที่ว่างเหนือและใต้รูป
ISMAP	กำหนดให้รูปที่กำหนดด้วยแท็ก เป็นรูปที่จะใช้เป็นอิมเมจแมพแบบปร ความหมายโดยเว็บเซิร์ฟเวอร์
USEMAP แท็กที่เกี่ยวข้อง	กำหนดให้รูปที่ถูกนำมาใช้เป็นอิมเมจแมพแบบความหมายโดยเว็บเบราว์เซอร์ <MAP>
<MAP>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบแท็ก <AREA> ซึ่งทำหน้าที่กำหนดพิกัดของพื้นที่ย่อยแต่ละส่วนภายในรูปอิมเมจแมพ
รูปแบบ	<MAP NAME="map_name"> ...hot region definition go here ... </MAP>
แอตทริบิวต์	
NAME	ใช้กำหนดชื่อให้แท็ก <MAP> ซึ่งจะถูกนำไปใช้อ้างอิงจากแท็กอื่นๆ เช่นแอตทริบิวต์ USEMAP ของแท็ก
แท็กที่เกี่ยวข้อง	 และ <AREA>
<AREA>	
ประเภท	ไม่มีแท็กปิด
หน้าที่	ใช้กำหนดพิกัดของพื้นที่ย่อยในรูปอิมเมจแมพ โดยในแต่ละพื้นที่ย่อยเหล่านี้จะทำได้เป็นลิงค์ไปยังเว็บเพจหรือไฟล์ที่ถูกกำหนดไว้คู่กับลิงค์นี้
รูปแบบ	<AREA SHAPE="RECT CIRCLE POLY DEFAULT" COORDS="coordinate_list" HREF="URL_of_linked_document" TARGET="frame_name" ALT="text_alternative" TABINDEX="tab_order_position" NOHREF>

แอตทริบิวต์

ALT	ใช้กำหนดข้อความสำหรับแต่ละพื้นที่ ซึ่งจะถูกแสดงเมื่อเว็บเบราว์เซอร์ไม่สามารถแสดงรูปภาพได้ หรือผู้เล่นสั่งไม่ให้อำนาจการโหลดรูปมาแสดงบนเว็บเพจ
COORDS	ใช้กำหนดพิกัดของพื้นที่ในรูป โคนตำแหน่งพิกัดเหล่านี้จะมาเป็นจุด และแยกแต่ละจุดด้วยเครื่องหมาย , (comma)
HREF	ใช้กำหนดชื่อไฟล์หรือเว็บเพจที่เชื่อมไว้กับพื้นที่ข้อย่อยนั้น ซึ่งผู้เล่นเว็บคลิกในบริเวณพื้นที่ข้อย่อยนั้น ก็จะเรียกไฟล์หรือเว็บเพจนั้นมาแสดง ให้กำหนดชื่อไฟล์ตามแบบ URL
NORHREF	ใช้กำหนดให้พื้นที่ข้อย่อยนั้น ไม่มีการลิงค์ไปยังไฟล์หรือเว็บเพจใด
SHAPE	ใช้กำหนดลักษณะของแต่ละพื้นที่ข้อย่อย ซึ่งมีค่าที่กำหนดได้คือ rect (สี่เหลี่ยมผืนผ้า), circle (วงกลม), poly (รูปสามเหลี่ยม), default (พื้นที่ที่เหลือ)
TABINDEX	ใช้กำหนดลำดับให้กับแต่ละพื้นที่ข้อย่อย เมื่อผู้ใช้กด Tab เคอร์เซอร์จะเลื่อนไปยังพื้นที่ข้อย่อยต่างๆตามลำดับที่กำหนดไว้
TARGET	ใช้กำหนดเฟรมที่ต้องการให้แสดงข้อมูล
แท็กที่เกี่ยวข้อง	<MAP>
<TABLE>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้กรอบข้อมูลและแท็กทั้งหมดที่ใช้ในตาราง
รูปแบบ	<pre> <TABLE ALIGN= "LEFT CENTER RIGHT" BORDER="thickness_in_pixels" BGCOLOR="color" WIDTH= "pixels_or_percentage_of_browser_width" COLS= "number_of_columns" CELLPADDING= "pixels" CELLSPACING= "pixels" FRAME= "outer_border_rendering" RULES= "inner_border_rendering"> ... </TABLE> </pre>

แอตทริบิวต์

ALIGN	ใช้กำหนดตำแหน่งของตารางบนเว็บเพจ มีค่าเป็นไปได้ 3 ค่าคือ left , right และ center โดยหากกำหนดเป็นค่า left หรือ right ข้อความที่อยู่ถัดจากแท็ก <TABLE> จะแสดงล้อมตารางนั้น
BORDER	กำหนดความหนาของเส้นกรอบตาราง กำหนดเป็นหน่วยพิกเซล

BGCOLOR	กำหนดสีพื้นในตาราง ซึ่งทำได้ 2 วิธีคือ กำหนดเป็นชื่อสี หรือใช้ค่า RGB
CELLPADDING	กำหนดระยะห่างระหว่างเส้นกรอบของตารางกับข้อมูลที่อยู่ในแต่ละเซลล์ กำหนดหน่วยเป็นพิกเซล
CELLSPACING	กำหนดระยะห่างระหว่างเซลล์ข้อมูล กำหนดหน่วยเป็นพิกเซล
COLS	กำหนดจำนวนคอลัมน์ในแต่ละแถว ทั้งนี้จะช่วยให้เว็บเบราว์เซอร์แสดงตารางได้รวดเร็วขึ้น
FRAME	กำหนดความต้องการแสดงกรอบที่ขอบด้านใดบ้าง มี 4 ด้านคือ ด้านบน, ด้านล่าง, ด้านซ้าย และด้านขวา ค่าที่เป็นไปได้มีดังนี้ above ใส่เส้นกรอบด้านบน below ใส่เส้นกรอบด้านล่าง border ใส่เส้นกรอบทั้ง 4 ด้าน box เช่นเดียวกับ border hsides ใส่เส้นกรอบเฉพาะด้านบนกับด้านล่าง lhs ใส่เส้นกรอบเฉพาะด้านซ้าย rhs ใส่เส้นกรอบเฉพาะด้านขวา vsides ใส่เส้นกรอบเฉพาะด้านซ้ายและด้านขวา void ไม่ใส่เส้นกรอบที่ด้านใดเลย
RULES	กำหนดความต้องการแสดงกรอบกับส่วนใดของตาราง ค่าที่เป็นไปได้ดังนี้ คือ all ใส่เส้นกรอบทุกช่องในตาราง cols ใส่เส้นกั้นระหว่างคอลัมน์ groups ใส่เส้นกรอบรอบกลุ่มของช่องในตารางที่กำหนดด้วยแท็กต่างๆ เช่น <THEAD>, <TBOY>, <TFOOT> และ <COLGROUPS> none ไม่ใส่กรอบที่ใดเลย row ใส่เส้นกั้นระหว่างแถว
WIDTH	ใช้กำหนดความกว้างของตาราง โดยกำหนดเป็นหน่วยพิกเซลหรือเป็นเปอร์เซ็นต์ที่เทียบกับความกว้างของจอ
แท็กที่เกี่ยวข้อง	ภายใต้กรอบของแท็ก <TABLE> และ </TABLE> จะเป็นข้อมูลตาราง และแท็กต่างๆ ดังนี้ <CAPTION>, <THEAD>, <TFOOT>, <TBOY>, <COLGROUP>, <COL>, <TR>, <TH> และ <TD>
<CAPTION>	
ประเภท	มีแท็กปิด

หน้าที่ ใช้ครอบข้อความสั้นๆ ที่เป็นชื่อตารางหรือข้อความกำกับตาราง โดยแท็กนี้ต้องอยู่ระหว่างแท็ก<TABLE> และ</TABLE> อีกที่หนึ่ง

รูปแบบ <CAPTION ALIGN= "TOP|BOTTOM|LEFT|RIGHT">
...caption text goes here ...
</CAPTION>

แอตทริบิวต์

ALIGN ใช้กำหนดตำแหน่งของชื่อตาราง เช่น ALIGN = top ให้แสดงชื่อตารางไว้ที่ด้านบนเหนือตาราง ค่าที่เป็นไปได้คือ top , bottom , left และ right

แท็กที่เกี่ยวข้อง <TABLE>

<THEAD>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อมูลส่วนหัวตาราง แท็กนี้ต้องอยู่ระหว่างแท็ก <TABLE> และ </TABLE>

รูปแบบ <THEAD ALIGN= " LEFT|CENTER|RIGHT|JUSTIFY">
VALIGN= "TOP|MIDDLE|BOTTOM|BASRLINE">
...rows that compose the header ...
</THEAD>

แอตทริบิวต์

ALIGN ใช้กำหนดตำแหน่งข้อในเซลล์ส่วนหัวตาราง ว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้คือ left , right , center และ justify

VALIGN ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนหัวตาราง ว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านบนและล่างอย่างไร มีค่าเป็นไปได้คือ top, middle, bottom และ baseline

แท็กที่เกี่ยวข้อง <TABLE>, <TR>, <TH> และ <TD>

<TFOOT>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อมูลส่วนท้ายตาราง แท็กนี้ต้องอยู่ระหว่างแท็ก<TABLE> และ </TABLE>

รูปแบบ <TFOOT ARIGN= "LEFT|CENTER|RIGHT|JUSTIFY">
VALIGN= "TOP|MIDDLE|BOTTOM|BASELINE">
...
</TFOOT>

แอตทริบิวต์

ALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนท้ายตาราง ว่าจะให้อยู่ชิดเส้นกรอบตาราง ด้านข้างอย่างไร มีค่าเป็นไปได้คือ left , right , center และ justify
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนท้ายตาราง ว่าจะให้อยู่ชิดกับกรอบตาราง ด้านบนและล่างอย่างไร มีค่าเป็นไปได้ คือ top, middle, bottom และ baseline
แท็กที่เกี่ยวข้อง	<TABLE>, <TR>, <TH> และ <TD>
<TBOBY>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้กรอบมุมส่วนตัวข้อมูลของตาราง
รูปแบบ	<pre> <TBODY ALIGN="LEFT CENTER RIGHT JUSTIFY" VALUE="TOP MIDDLE BOTTOM BASELINE"> ... </TBODY> </pre>
แอตทริบิวต์	
ALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในตาราง ว่าจะให้อยู่ชิดกับเส้น กรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้คือ left , right ,centerและjustify
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในตาราง ว่าจะให้อยู่ชิดกับเส้น กรอบตารางด้านบนและด้านล่างอย่างไร มีค่าเป็นไปได้คือ top,middle,bottom และ baseline
แท็กที่เกี่ยวข้อง	<TABLE>, <TR>, <TH> และ <TD>
<COLGROUP>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้กลุ่มข้อมูลคอลัมน์ที่ต้องการให้มีขนาดและลักษณะเหมือนกันเข้าด้วยกันซึ่งจะ ทำให้สะดวกกว่าวิธีกำหนดขนาดและลักษณะของแต่ละคอลัมน์แยกกันซึ่งปกติ ระหว่างแท็ก <COLGROUP> และ </COLGROUP> จะไม่มีส่วนที่เป็นเนื้อหา หรือแท็กอื่นๆนอกจากแท็ก <COL> เท่านั้น โดยแท็ก <COL> นี้จะใช้กำหนด ลักษณะให้กับกลุ่มคอลัมน์ย่อยที่อยู่ใต้กลุ่มคอลัมน์ใหญ่ที่กำหนดโดยแท็ก <COLGROUP> อีกทีหนึ่ง
รูปแบบ	<pre> <COLGROUP SPAN="number_of_columns" WIDTH="width_of_column_group" ALIGN="LEFT RIGHT CENTER JUSTIFY" VALIGN="TOP MIDDLE BOTTOM BASELINE"> ... </pre>

</COLGROUP>

แอตทริบิวต์

ALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในกลุ่มว่าจะอยู่ชิดกับเส้นกรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้อีกคือ left, right, center และ justify
SPAN	กำหนดจำนวนคอลัมน์ในแต่ละแถวที่จะอยู่ในกลุ่ม
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในกลุ่มว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านบนและด้านล่างอย่างไร มีค่าเป็นไปได้อีกคือ top, middle, bottom และbaseline
WIDTH	กำหนดความกว้างของเซลล์ที่อยู่ในกลุ่ม โดยกำหนดเป็นหน่วยฟิกเซล

แท็กที่เกี่ยวข้อง

<COL>

<COL>

ประเภท

ไม่มีแท็กปิด

หน้าที่

ใช้กำหนดความกว้างและลักษณะอื่นให้กับคอลัมน์หรือบางคอลัมน์ที่อยู่ในกลุ่มตารางแท็ก<COL>เป็นตัวแทนของกลุ่มย่อยของเซลล์ที่อยู่ในกลุ่มใหญ่ที่กำหนดโดยแท็ก <COLGROUP> โดยต้องกำหนดแท็ก <COL> ไว้ได้แท็ก <COLGROUP> อีกทีหนึ่ง

รูปแบบ

```
<COL SPAN="number_of_column_subgroup"
ALIGN="LEFT|RIGHT|CENTER|JUSTIFY"
VALIGN="TOP|MIDDLE|BOTTOM|BASELINE">
```

แอตทริบิวต์

ALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในกลุ่มว่าจะอยู่ชิดกับเส้นกรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้อีกคือ left, right, center และ justify
SPAN	กำหนดจำนวนคอลัมน์ในแต่ละแถวที่จะอยู่ในกลุ่ม
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ส่วนที่เป็นข้อมูลในกลุ่มว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านบนและด้านล่างอย่างไร มีค่าเป็นไปได้อีกคือ top, middle, bottom และbaseline

WIDTH

กำหนดความกว้างของเซลล์ที่อยู่ในกลุ่ม โดยกำหนดเป็นหน่วยฟิกเซล

แท็กที่เกี่ยวข้อง

<COLGROUP>

<TR>

ประเภท

มีแท็กปิด

หน้าที่

ใช้กำหนดแถวของตาราง โดยอาจเป็นส่วนหัวของตาราง, ส่วนท้าย และส่วนที่เป็นข้อมูลในตาราง

รูปแบบ	<pre><TR BGCOLOR="color" ALIGN="LEFT RIGHT CENTER JUSTIFY" VALIGN="TOP MIDDLE BOTTOM BASELINE"> ... </TR></pre>
แอตทริบิวต์	หากมีการระบุแอตทริบิวต์ในแท็กนี้ ข้อมูลในแถวนั้นจะใช้แอตทริบิวต์นี้แทนแอตทริบิวต์ที่กำหนดไว้ในแท็กอื่น
ALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ในแต่ละแถวว่าจะอยู่ชิดกับเส้นกรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้คือ left, right, center และ justify
BGCOLOR	ใช้กำหนดสีภายในแต่ละแถว
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ในแต่ละแถวว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านบนและด้านล่างอย่างไร มีค่าเป็นไปได้คือ top, middle, bottom และ baseline
แท็กที่เกี่ยวข้อง	<TD> และ <TH>
<TD>, <TH>	มีแท็กปิด
ประเภท	
หน้าที่	ใช้กำหนดแถวเซลล์ตาราง โดย<TH>ใช้กับเซลล์มีเยื้องหัวตาราง ซึ่งมีผลให้ข้อความในเซลล์นั้นถูกแสดงเป็นหัวหนา และอยู่ตรงกลางระหว่างความกว้างของเซลล์ ส่วนแท็ก <TD> ใช้กับส่วนที่เซลล์ข้อมูลของตาราง โดยแสดงเป็นแบบตัวธรรมดา ชิดกับเส้นกรอบด้านซ้ายของตาราง และอยู่กลางหากความสูงของเซลล์นั้นยาวเกินกว่าจำนวนบรรทัดของข้อความที่อยู่ในเซลล์นั้น
รูปแบบ	<pre><TD ALIGN="LEFT RIGHT CENTER JUSTIFY" VALIGN="TOP MIDDLE BOTTOM BASELINE" BGCOLOR="color" NOWRAP ROWSPAN="number_of_rows" COLSPAN="number_of_columns"> ... </TD> หรือ <TH ALIGN="LEFT RIGHT CENTER JUSTIFY" VALIGN="TOP MIDDLE BOTTOM BASELINE" BGCOLOR="color" NOWRAP ROWSPAN="number_of_rows" COLSPAN="number_of_columns"> ... </TH></pre>
แอตทริบิวต์	
ALIGN	ใช้กำหนดตำแหน่งข้อมูลของเซลล์ว่าจะอยู่ชิดกับเส้นกรอบตารางด้านข้างอย่างไร มีค่าเป็นไปได้คือ left, right, center และ justify

BGCOLOR	ใช้กำหนดสีภายในแต่ละเซลล์
COLSPAN	ใช้กำหนดว่าจะรวมอีกกี่คอลัมน์เป็นหนึ่งเซลล์
NOWSPAN	สั่งให้เบราว์เซอร์ไม่ต้องนำข้อความที่ขวาล้นเซลล์ไปขึ้นบรรทัดใหม่
ROWSPAN	ใช้กำหนดว่าจะรวมอีกกี่แถวเป็นหนึ่งเซลล์ที่ใช้แสดงข้อมูล
VALIGN	ใช้กำหนดตำแหน่งข้อมูลในเซลล์ว่าจะให้อยู่ชิดกับเส้นกรอบตารางด้านบน และด้านล่างอย่างไร มีค่าเป็นไปได้คือ top, middle, bottom และbaseline
แท็กที่เกี่ยวข้อง	<TR>

<FORM>

ประเภท	มีแท็กปิด
หน้าที่	ใช้กรอบข้อมูลและแท็กที่ใช้สร้างฟิลด์รับข้อมูลและปุ่มต่างๆที่ประกอบกันเป็นฟอร์ม นอกจากนี้แท็ก <FORM> ยังเป็นคำกำหนดว่าจะส่งข้อมูลให้เว็บเซิร์ฟเวอร์อย่างไร และเรียกใช้โปรแกรมตัวใดทำงานด้วย
รูปแบบ	<pre> <FORM ACTION="URL_of_processing_script" METHOD="GET POST" TARGET="frame_name" ENCTYPE="MIME_type_of_file_to_upload" ACCEPT-CHARSET="acceptable_character_sets"> ... </FORM> </pre>
แอตทริบิวต์	
ACCEPT-CHARSET	กำหนดชุดของตัวอักษรที่ฟอร์มสามารถรับได้
ACTION	กำหนดชื่อไฟล์หรือโปรแกรมที่ทำหน้าที่ประมวลผลข้อมูลที่ผู้เล่นเว็บป้อนผ่านฟอร์มเข้ามา โดยโปรแกรมหรือไฟล์นี้อยู่ที่ฝั่งเซิร์ฟเวอร์ และต้องกำหนดชื่อไฟล์ตามแบบ URL
ENCTYPE	กำหนดชนิดของไฟล์ที่จะถูกรับและส่งในฟอร์มตามแบบ MIME
METHOD	กำหนดวิธีส่งข้อมูลให้เว็บเซิร์ฟเวอร์ มีค่าเป็นไปได้ 2 แบบ คือ POST หรือ GET
TARGET	กำหนดชื่อเฟรมที่ต้องการให้แสดงผลลัพธ์ที่ได้จากการประมวลผลของเว็บเซิร์ฟเวอร์ตามข้อมูลที่ฟอร์มส่งไปให้
แท็กที่เกี่ยวข้อง	<INPUT>, <SELECT>, <OPTION>, <TEXTAREA>, <BUTTON>, <LABEL>, <FIELDSET> และ <LEGEND>

<INPUT>

ประเภท	ไม่มีแท็กปิด
หน้าที่	ใช้สร้างฟิลด์รับข้อมูลแบบต่างๆตามค่าที่กำหนดให้กับแอตทริบิวต์ TYPE

รูปแบบ

<!-- Text and password fields -->

```
<INPUT TYPE="TEXT|PASSWORD" NAME="field_name"
      VALUE="default_value" SIZE="field_size"
      MAXLENGTH="maximum_input_length" DISABLED READONLY>
```

หรือ

<!-- Hidden field -->

```
<INPUT TYPE="HIDDEN" NAME="field_name" VALUE="field_value">
```

หรือ

<!-- Checkbox -->

```
<INPUT TYPE="CHECKBOX" NAME="field_name" VALUE="field_value"
      CHECKED DISABLED>
```

หรือ

<!-- Radio button -->

```
<INPUT TYPE="RADIO" NAME="field_name" VALUE="field_value"
      CHECKED DISABLED>
```

หรือ

<!-- File upload -->

```
<INPUT TYPE="FILE" NAME="field_name" VALUE="default_value"
      ACCEPT="acceptable_MIME_types" DISABLED>
```

หรือ

<!-- Image-based button -->

```
<INPUT TYPE="IMAGE" SRC="URL_of_image_file"
      ALT="description" ALIGN="TOP|MIDDLE|BOTTOM|LEFT|RIGHT"
      USEMAP="map_name" DISABLED>
```

หรือ

<!-- Scripted button -->

```
<INPUT TYPE="BUTTON" VALUE="button_label"
      ONCLICK="script_name" DISABLED>
```

หรือ

<!-- Submit/reset button -->

```
<INPUT TYPE="SUBMIT|RESET" VALUE="button_label" DISABLED>
```

แอตทริบิวต์

NAME

ใช้กำหนดชื่อของฟิลด์รับข้อมูล ฟิลด์รับข้อมูลทุกตัวควรมีชื่อ ทั้งนี้เพราะโปรแกรมประมวลผลทางฝั่งเซิร์ฟเวอร์จะอ้างอิงข้อมูลภายในฟิลด์ด้วยชื่อนี้ได้

	VALUE เป็นแอตทริบิวต์ที่ใช้เพื่อกำหนดค่าเริ่มต้นให้ฟิลด์รับข้อมูล หรือกำหนดข้อความที่จะแสดงไว้บนปุ่มรับคำสั่ง
TYPE	ใช้กำหนดว่าชนิดของฟิลด์ มีค่าเป็นไปได้อย่างไร
text	ฟิลด์รับข้อความแบบบรรทัดเดียว โดยมีแอตทริบิวต์อื่นที่ต้องกำหนดร่วมด้วยคือ NAME, VALUE, LENGTH, MAXLENGTH, DISABLE และ READONLY
password	ฟิลด์รับข้อความแบบบรรทัดเดียวคล้ายกับแบบ text แต่ใช้กับการป้อนรหัสผ่านโดยเฉพาะ โดยรหัสผ่านหรือข้อความที่ถูกต้องป้อนเข้าไปจะถูกแสดงเป็นเครื่องหมาย * แทนตัวอักษรแต่ละตัวเสมอ
hidden	ฟิลด์นี้จะไม่ถูกแสดงบนเว็บเพจ แต่ข้อมูลที่เก็บอยู่ภายในฟิลด์จะถูกส่งไปให้เว็บเซิร์ฟเวอร์ สำหรับฟิลด์แบบนี้ต้องการกำหนดแอตทริบิวต์ NAME และ VALUE ร่วมด้วย
checkbox	ฟิลด์แบบนี้ผู้ใช้จะต้องคลิกเพื่อเลือกหรือไม่เลือกรายการนั้น โดยต้องกำหนดแอตทริบิวต์ NAME เพื่อกำหนดชื่อให้กับฟิลด์ และแอตทริบิวต์ VALUE เพื่อกำหนดค่าที่ส่งให้กับเว็บเซิร์ฟเวอร์หากฟิลด์นั้นถูกคลิกเลือก กรณีที่ต้องการให้มีค่าเริ่มต้นของฟิลด์ใดเป็นแบบถูกเลือกไว้ ก็ให้ใส่แอตทริบิวต์ CHECKED กำกับไปด้วย
radio	ฟิลด์แบบนี้จะมีตัวเลือกหลายตัว แต่ผู้เล่นเว็บจะเลือกได้ตัวเดียวเท่านั้น โดยต้องใส่แท็ก <INPUT> เท่ากับจำนวนตัวเลือกที่มี และกำหนดแอตทริบิวต์ NAME เพื่อกำหนดชื่อตัวแปร ในแท็กเหล่านั้นให้เหมือนกัน แต่ใส่แอตทริบิวต์ VALUE พร้อมทั้งค่าในแต่ละแท็กไม่เหมือนกัน ซึ่งค่านี้จะเป็นค่าที่ส่งให้กับเว็บเซิร์ฟเวอร์เมื่อรายการนั้นถูกเลือก หากต้องการให้ตัวเลือกใดถูกเลือกไว้เป็นค่าเริ่มต้นก็สามารถทำได้ โดยใส่แอตทริบิวต์ CHECKED กำกับไว้ที่ค่านั้น
file	ฟิลด์ที่ใช้ใส่ชื่อไฟล์เพื่อส่ง (upload) ไปยังเว็บเซิร์ฟเวอร์พร้อมกับข้อมูลอื่นๆที่ผู้เล่นกรอกผ่านฟอร์ม โดยสามารถใส่แอตทริบิวต์ NAME เพื่อกำหนดชื่อให้กับฟิลด์, VALUE เพื่อกำหนดค่าเริ่มต้นให้กับฟิลด์, ACCEPT เพื่อกำหนดประเภทของไฟล์ และ DISABLED เพื่อกำหนดให้ฟิลด์นั้นไม่รับข้อมูลใดๆ
image	ใช้กำหนดรูปแทนปุ่มรับคำสั่งโดยใส่แอตทริบิวต์ SRC เพื่อกำหนดตำแหน่งและชื่อไฟล์ที่เก็บรูปนั้น, ALT เพื่อระบุชื่อหรือข้อความที่ต้องการให้แสดงแทนในกรณีที่ไม่สามารถหารูปนั้นพบ, ALIGN เพื่อกำหนดตำแหน่งที่จะให้รูปนั้นปรากฏ และ DISABLED เพื่อกำหนดให้รูปนั้นไม่รับข้อมูลใดๆ

button	ใช้สร้างปุ่มรับคำสั่ง โดยใส่แอตทริบิวต์ VALUE เพื่อกำหนดข้อความที่จะให้ปรากฏอยู่บนปุ่ม, แอตทริบิวต์ ONCLICK เพื่อกำหนดคำสั่งหรือโปรแกรมบนฝั่งเซิร์ฟเวอร์ที่จะถูกเรียกให้ทำงานเมื่อกดปุ่มนี้คลิก และ DISABLED เพื่อกำหนดให้ปุ่มนี้ไม่ทำงาน
submit	ใช้สร้างปุ่มรับคำสั่ง SUBMIT ซึ่งเมื่อผู้ใช้คลิกที่ปุ่มนี้เบราว์เซอร์จะส่งข้อมูลทั้งหมดบนฟอร์มไปให้เว็บเซิร์ฟเวอร์
reset	ใช้สร้างปุ่มรับคำสั่ง RESET ซึ่งเมื่อผู้ใช้คลิกที่ปุ่มนี้เบราว์เซอร์จะล้างข้อมูลทั้งหมดที่อยู่ในฟอร์ม ให้เหมือนเมื่อตอนเริ่มต้น

DISABLED	ใช้ระงับการทำงานของฟิลด์ทุกชนิด โดยเมื่อเพิ่มแอตทริบิวต์ DISABLED บนเบราว์เซอร์จะยังคงแสดงฟิลด์เหล่านั้น แต่จะไม่ยอมให้ผู้ใช้ป้อนข้อมูลหรือเปลี่ยนแปลงตัวเลือกในฟิลด์นั้นได้
LENGTH	ใช้กำหนดความยาวเป็นตัวอักษรให้กับฟิลด์รับข้อมูลที่จะแสดงบนหน้าจอ
MAXLENGTH	ใช้กำหนดความยาวสูงสุดเป็นจำนวนตัวอักษรที่ฟิลด์นั้นจะรับได้
READONLY	กำหนดให้ฟิลด์นี้แสดงข้อมูลเพียงอย่างเดียว ห้ามผู้ใช้ป้อนหรือแก้ไขข้อมูล

<SELECT>

ประเภท

มีแท็กปิด

หน้าที่

สร้างฟิลด์แบบมีหลายตัวเลือก โดยผู้ใช้สามารถเลือกตัวเลือกเหล่านั้นได้ครั้งละ 1 หรือหลายตัวเลือกก็ได้

รูปแบบ

```
<SELECT NAME="field_name" SIZE="visible_row" MULTIPLE
DISABLED>
...
</SELECT>
```

แอตทริบิวต์

DISABLED	ห้ามไม่ให้ผู้ใช้ป้อนหรือแก้ไขข้อมูลภายในฟิลด์ได้
MULTIPLE	กำหนดให้ผู้ใช้สามารถเลือกตัวอักษรได้มากกว่า 1 รายการ หากจะเลือกหลายตัวเลือกให้กด Ctrl ค้างไว้พร้อมกับคลิกรายการที่ต้องการเลือก
NAME	กำหนดชื่อให้ฟิลด์
SIZE	กำหนดจำนวนตัวเลือกที่จะแสดงบนเว็บเพจ
แท็กที่เกี่ยวข้อง	<OPTION>

<OPTION>

ประเภท

มีแท็กปิด

หน้าที่

ใช้สร้างตัวเลือกในฟิลด์แบบ <SELECT>

รูปแบบ <OPTION VALUE="option_value" SELECTED DISABLED>
... option text ...

แอตทริบิวต์

DISABLED ห้ามไม่ให้ผู้ใช้เลือกตัวเลือกรายการนี้ และบราวเซอร์จะแสดงฟิลด์เป็นสีจางกว่าฟิลด์ปกติ

SELECTED กำหนดให้รายการนี้เป็นค่าเริ่มต้น คือถูกเลือกไว้

VALUE กำหนดค่าให้รายการนี้ ซึ่งจะเป็ค่าเดียวกับค่าที่ถูกเก็บไว้ในหน่วยความจำและถูกส่งไปยังเว็บเซิร์ฟเวอร์

แท็กที่เกี่ยวข้อง <SELECT>

<TEXTAREA>

ประเภท มีแท็กปิด

หน้าที่ ใช้สร้างฟิลด์รับข้อความแบบหลายบรรทัด

รูปแบบ <TEXTAREA NAME="field_name" ROWS="number_of_rows"
COLS="number_of_columns" DISABLED READONLY
... default text to appear in windows ...

</TEXTAREA>

แอตทริบิวต์

COLS กำหนดจำนวนตัวอักษรที่ผู้ใช้จะป้อนได้ในแต่ละบรรทัด

DISABLED ห้ามไม่ให้ผู้ใช้ป้อนข้อมูลในฟิลด์นี้ได้ และบราวเซอร์จะแสดงฟิลด์นี้เป็นสีจางกว่าฟิลด์ปกติ

NAME กำหนดชื่อให้กับฟิลด์เพื่อที่จะใช้อ้างอิงในโปรแกรมได้

READONLY กำหนดให้ฟิลด์นี้แสดงข้อมูลเพียงอย่างเดียว ห้ามผู้ใช้แก้ไขข้อมูล

ROWS กำหนดจำนวนบรรทัดที่ให้ป้อนข้อความ

<BUTTON>

ประเภท มีแท็กปิด

หน้าที่ ใช้สร้างปุ่มรับคำสั่งบนฟอร์ม ปุ่มรับคำสั่งที่สร้างด้วยแท็กนี้จะได้ผลใกล้เคียงกับแท็ก <INPUT> ต่างกันตรงรูปปุ่มที่ได้ คือถ้าเพิ่มแท็ก เข้าไปภายใต้กรอบของแท็ก <BUTTON> ก็สามารถใช้รูปนั้นแทนรูปปุ่มได้

รูปแบบ <BUTTON TYPE="SUBMIT|RESET" NAME="button_name"
VALUE="button_value" DISABLED
... text for button face or tag ...
</BUTTON>

แอตทริบิวต์

DISABLED	ห้ามไม่ให้ผู้ใช้คลิกที่ปุ่ม และบราวเซอร์จะแสดงปุ่มนี้เป็นสีจางกว่าปุ่มปกติ
NAME	กำหนดชื่อให้กับปุ่ม
TYPE	กำหนดชนิดของปุ่ม มีค่าที่เป็นไปได้คือ RESET หรือ SUBMIT ซึ่งจะมีผลเหมือนปุ่ม RESET และ SUBMIT ที่สร้างจากแท็ก <INPUT>
VALUE	กำหนดข้อความที่จะให้แสดงบนปุ่ม

<LABEL>

ประเภท	มีแท็กปิด
หน้าที่	กำหนดข้อความที่แสดงกำกับฟิลด์ โดยส่วนใหญ่จะใช้กำกับฟิลด์ที่เป็นแบบ text เพื่อให้ผู้ใช้ทราบว่าข้อมูลที่ใส่กรอกนั้นเป็นข้อมูลที่เกี่ยวข้องอะไร เช่น ชื่อหรือที่อยู่ เป็นต้น ถ้าผู้ใช้คลิกที่ข้อความนี้บราวเซอร์จะเลื่อนเคอร์เซอร์ไปยังฟิลด์ที่คู่กันทันที
รูปแบบ	<pre><LABEL FOR="field_id" ACCESSKEY="shortcut_key" DISABLED> ... label text goes here ... </LABEL></pre>
แอตทริบิวต์	ACCESSKEY กำหนดปุ่มในแป้นพิมพ์เพื่อให้เคอร์เซอร์เลื่อนไปยังฟิลด์หรือปุ่มภายในฟอร์มที่ต้องการ เช่น หากกำหนดว่า ACCESSKEY = "A" ผู้ใช้สามารถกด Alt + A หรือ Ctrl + A เพื่อให้เคอร์เซอร์เลื่อนไปยังฟิลด์รับข้อมูลหรือปุ่มนั้นได้ทันที DISABLED ห้ามไม่ให้ผู้ใช้คลิกข้อความ LABEL นี้ได้ FOR กำหนดค่าให้เป็น ID ของฟิลด์ที่ต้องการให้คู่กับ LABEL นี้ แท็กที่เกี่ยวข้อง <INPUT>, <SELECT>, และ <TEXTAREA>

<FIELDSET>

ประเภท	มีแท็กปิด
หน้าที่	ใช้จัดกลุ่มฟิลด์บางฟิลด์ไว้ด้วยกัน
รูปแบบ	<pre><FIELDSET> ... related input fields ... </FIELDSET></pre>
แอตทริบิวต์	ไม่มี
แท็กที่เกี่ยวข้อง	<LEGEND>

<LEGEND>

ประเภท	มีแท็กปิด
--------	-----------

หน้าที่	สร้างข้อความกำกับกลุ่มของฟิลด์ที่สร้างโดยแท็ก <FIELDSET>
รูปแบบ	<pre><LEGEND ALIGN="LEFT RIGHT TOP BOTTOM" ACCESSKEY="shortcut_key"> ... legend text goes here ... </LEGEND></pre>

แอตทริบิวต์

ACCESSKEY	กำหนดปุ่มในแป้นพิมพ์เพื่อให้เคอร์เซอร์เลื่อนไปยังกลุ่มของฟิลด์ที่ต้องการ เช่น หากกำหนด ACCESSKEY="A" ผู้ใช้สามารถกด Alt + A หรือ Ctrl + A เพื่อให้เคอร์เซอร์เลื่อนไปยังกลุ่มของฟิลด์นั้น
ALIGN	ใช้กำหนดตำแหน่งของข้อความ โดยมีค่าที่กำหนดได้คือ left, right, top และ bottom
แท็กที่เกี่ยวข้อง	<FIELDSET>
<FRAMESET>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้แบ่งพื้นที่ในเว็บเพจออกเป็นส่วนย่อย
รูปแบบ	<pre><FRAMESET ROWS="list_of_row_sizes" COLS="list_of_column_sizes"> ... </FRAMESET></pre>
แอตทริบิวต์	
ROWS	ใช้กำหนดว่าจะให้มีกี่แบ่งพื้นที่ทั้งหมดออกเป็นกี่แถว และในแต่ละแถวมีความสูงเท่าใด
COLS	ใช้กำหนดว่าจะให้มีการแบ่งพื้นที่ทั้งหมดออกเป็นกี่คอลัมน์ และในแต่ละคอลัมน์มีความกว้างเท่าใด
	การใช้แอตทริบิวต์ทั้งสองนี้มีข้อห้ามคือใช้ด้วยกันไม่ได้ สำหรับวิธีกำหนดขนาดของแต่ละพื้นที่ย่อยด้วย ROWS และ COLS มีหลายแบบ เช่น กำหนดเป็นเปอร์เซ็นต์เมื่อเทียบกับพื้นที่ใหญ่ กำหนดเป็นหน่วยพิกเซล หรือกำหนดเป็นอัตราส่วนก็ได้
แท็กที่เกี่ยวข้อง	<FRAME> และ <NOFRAME>

<FRAME>

ประเภท	มีแท็กปิด
หน้าที่	ใช้กำหนดข้อมูลที่จะถูกแสดงไว้ในเฟรม

รูปแบบ	<pre> <FRAME SRC="URL_of_document" NAME="frame_name" FRAMEBORDER="0 1" MARGINWIDTH="width_in_pixels" MARGINHEIGHT="height_in_pixels" NORESIZE SCROLLING="YES NO AUTO"> </pre>
แอตทริบิวต์	
FRAMEBORDER	กำหนดความต้องการแสดงเส้นกรอบของเฟรมหรือไม่ มีค่าเป็นไปได้อีกคือ 0 หมายถึงไม่ต้องแสดงเส้นกรอบ ส่วน 1 หมายถึงให้แสดงเส้นกรอบ
MARGINHEIGHT	กำหนดระยะห่างระหว่างเส้นกรอบด้านบนของเฟรมกับตัวข้อมูล กำหนดเป็นหน่วยพิกเซล
MARGINWIDTH	กำหนดระยะห่างระหว่างเส้นกรอบด้านซ้ายของเฟรมกับตัวข้อมูล กำหนดเป็นหน่วยพิกเซล
NAME	กำหนดชื่อให้กับเฟรม เพื่อให้แท็กอื่นๆสามารถอ้างอิงได้ เช่น แท็ก <A>, <FORM> และ <AREA> เป็นต้น
NORESIZE	กำหนดว่าไม่ยอมให้ผู้ใช้อยู่หรือขยายขนาดของเฟรม โดยปกติผู้ใช้สามารถใช้เมาส์ย่อหรือขยายขนาดของเฟรมได้
SCROLLING	กำหนดความต้องการแสดงแถบเลื่อนด้านข้างของเฟรมหรือไม่ มีค่าเป็นไปได้อีกคือ yes หมายถึง แสดงแถบเลื่อนเสมอ no หมายถึง ไม่ต้องแสดงแถบเลื่อนเลย auto หมายถึง แสดงแถบเลื่อนในกรณีที่มีข้อมูลล้นเฟรมเท่านั้น
SRC	กำหนดชื่อไฟล์ HTML หรือไฟล์ข้อมูลอื่นที่ชี้ในการแสดงเฟรม เป็นแอตทริบิวต์ที่จำเป็นต้องมีเสมอ

แท็กที่เกี่ยวข้อง <FRAMESET>

<NOFRAME>

ประเภท	มีแท็กปิด
หน้าที่	ใช้แสดงข้อความในกรณีที่เว็บเบราว์เซอร์ที่ใช้ไม่สามารถแสดงเฟรมได้ซึ่งเบราว์เซอร์จะแสดงข้อความที่กำหนดไว้ในแท็กนี้แทน
รูปแบบ	<pre> <NOFRAME> ...non-frames content goes here... </NOFRAME> </pre>
แอตทริบิวต์	ไม่มี

แท็กที่เกี่ยวข้อง

<FRAMESET>

<IFRAME>

ประเภท

มีแท็กปิด

หน้าที่

ใช้สร้างเฟรมแบบที่ลอยได้ (floating frame) คือเฟรมนี้ไม่จำเป็นต้องมีการจัดพื้นที่ไว้ก่อนล่วงหน้าเหมือนกับเฟรมที่สร้างด้วยแท็ก <FRAME> แต่จะมีลักษณะเหมือนกับรูปภาพบนเว็บเพจ คือสามารถสร้างและวางไว้ตรงไหนก็ได้ โดยจะแสดงเป็นกรอบสี่เหลี่ยม และมีแถบเลื่อนด้านข้างถ้าข้อมูลในเฟรมมีมากกว่าที่จะแสดงในเฟรมได้หมดในทีเดียว

รูปแบบ

```
<IFRAME SRC="URL_of_document" NAME="frame_name"
FRAMEBORDER="0|1"
WIDTH="frame_width_in_pixels_or_percentage"
HEIGHT="frame_height_in_pixels_or_percentage"
MARGINWIDTH="margin_width_in_pixels"
MARGINHEIGHT="margin_height_in_pixels"
SCROLLING="YES|NO|AUTO"
ALIGN="TOP|MIDDLE|BOTTOM|LEFT|RIGHT">
... text or image alternative to the floating frame ...
</IFRAME>
```

แอตทริบิวต์

ALIGN

ใช้กำหนดการจัดวางของข้อความที่ต่อท้ายเฟรม
 ถ้ากำหนดค่าเป็น TOP, MIDDLE หรือ BOTTOM แล้ว ข้อความที่ต่อท้ายเฟรมนี้จะถูกแสดงไว้เสมอกับขอบด้านบน หรือตรงกลาง, ด้านล่างของเฟรมตามลำดับ และเมื่อข้อความยาวจนล้นจอ ข้อความที่เหลือจะถูกปิดไปขึ้นบรรทัดใหม่ที่ด้านล่างได้เฟรม
 ถ้ากำหนดค่าเป็น LEFT หรือ RIGHT แล้ว ข้อความที่ต่อท้ายเฟรมนี้จะแสดงล้อมเฟรมทางด้านซ้าย และขวาตามลำดับ

FRAMEBORDER

กำหนดว่าต้องการให้แสดงเส้นกรอบของเฟรมหรือไม่ มีค่าที่กำหนดได้คือ 0 หมายถึงไม่แสดงเส้นกรอบของเฟรม ส่วน 1 หมายถึงให้แสดงเส้นกรอบ

HEIGHT

กำหนดความสูงของเฟรม กำหนดเป็นหน่วยพิกเซล

MARGINHEIGHT

กำหนดระยะห่างระหว่างเส้นกรอบด้านบนของเฟรมกับข้อมูล

MARGINWIDTH

	กำหนดระยะห่างระหว่างเส้นกรอบด้านซ้ายของเฟรมกับข้อมูล
NAME	กำหนดชื่อให้เฟรม เพื่อให้แท็กอื่นสามารถอ้างอิงได้ เช่น <A>, <FORM> และ <AREA> เป็นต้น
SCROLLING	กำหนดความต้องการแสดงแถบเลื่อนด้านข้างของเฟรมหรือไม่ มีค่าเป็นไปได้อคือ yes หมายถึง แสดงแถบเลื่อนเสมอ no หมายถึง ไม่ต้องแสดงแถบเลื่อนเลย auto หมายถึง แสดงแถบเลื่อนในกรณีที่มีข้อมูลล้นเฟรมเท่านั้น
SRC	กำหนดชื่อไฟล์ข้อมูลที่จะนำมาแสดงในเฟรม โดยกำหนดตามรูปแบบ URL
WIDTH	กำหนดความกว้างของเฟรม กำหนดเป็นหน่วยพิกเซล

<APPLET>

ประเภท	มีแท็กปิด
หน้าที่	ใช้เรียกโปรแกรมเสริมที่เป็น Java applet เข้ามาทำงานในเว็บเบราว์เซอร์
รูปแบบ	<pre><APPLET WIDTH="width_in_pixels" HEIGHT="height_in_pixels" CODEBASE="base_URL_for_applet" CODE="applet_class_file" ARCHIVE="archive_list" ALT="text_alternative" HSPACE="pixels" VSPACE="pixels"> ... </APPLET></pre>
แอตทริบิวต์	
ALIGN	ใช้กำหนดการจัดวางข้อความที่อยู่ต่อท้าย Java applet ถ้ากำหนดค่าเป็น TOP, MIDDLE หรือ BOTTOM แล้ว ข้อความที่ต่อท้ายโปรแกรมนี้จะถูกแสดงไว้เสมอบนขอบด้านบน, ตรงกลาง หรือด้านล่างของวินโดว์ของโปรแกรม applet นั้นตามลำดับ แล้วขึ้นบรรทัดใหม่ต่อท้ายทางด้านล่างของวินโดว์นั้น ถ้ากำหนดค่าเป็น LEFT หรือ RIGHT แล้ว ข้อความที่ต่อท้ายนั้น จะแสดงล้อมรอบวินโดว์ของโปรแกรมทางด้านซ้าย และด้านขวาตามลำดับ
ALT	กำหนดข้อความที่จะแสดงไว้แทนตัวโปรแกรม ในกรณีที่เบราว์เซอร์ไม่สามารถเรียกใช้โปรแกรมได้
ARCHIVE	ชี้ไปที่ที่มีโปรแกรมเพิ่มหรือไฟล์ข้อมูลเพิ่มสำหรับออบเจกต์ที่ต้องถูกดึงมาก่อนตัวชี้โปรแกรม
CODE	กำหนดชื่อโปรแกรมที่เรียกใช้
CODEBASE	กำหนดที่อยู่ของไฟล์โปรแกรมตามแบบ URL
HEIGHT	กำหนดความสูงของวินโดว์ที่ใช้แสดงโปรแกรม กำหนดเป็นพิกเซล

HSPACE	กำหนดพื้นที่ว่างบริเวณด้านซ้ายและด้านขวาของวินโดวที่โปรแกรมทำงานอยู่
NAME	กำหนดชื่อให้โปรแกรม applet นี้ เพื่อให้ applet อื่นสามารถอ้างอิงได้
OBJECT	เป็นวิธีเรียกใช้ Java applet แบบหนึ่งสามารถใช้แทนแอตทริบิวต์ CODE แต่วิธีอ้างอิง Java applet ด้วยแอตทริบิวต์นี้ต้องกำหนดเป็นหมายเลข Serial No.
VSPACE	กำหนดพื้นที่ว่างบริเวณด้านบนและล่างของวินโดวที่โปรแกรมทำงานอยู่
WIDTH	กำหนดความกว้างของวินโดวที่โปรแกรมจะทำงาน กำหนดเป็นหน่วยพิกเซล
แท็กที่เกี่ยวข้อง	<PARAM>

<PARAM>

ประเภท	ไม่มีแท็กปิด
หน้าที่	ใช้กำหนดค่าพารามิเตอร์ที่จะต้องส่งให้กับโปรแกรมเสริม
รูปแบบ	<pre><PARAM NAME="parameter_name" VALUE="parameter_value" VALUETYPE="DATA REF OBJECT" TYPE="intermedia_type"></pre>
แอตทริบิวต์	
NAME	กำหนดชื่อพารามิเตอร์
TYPE	กำหนดชนิดของข้อมูลที่เบราว์เซอร์ต้องรับ กำหนดค่าตาม MINE
VALUE	กำหนดค่าพารามิเตอร์
VALUETYPE	กำหนดชนิดของข้อมูล เช่น DATA, REF และ OBJECT
แท็กที่เกี่ยวข้อง	<OBJECT> และ <APPLET>

<OBJECT>

ประเภท	มีแท็กปิด
หน้าที่	ใช้เรียกโปรแกรมเสริม หรือออบเจกต์เข้ามาทำงานหรือไว้ในเว็บเพจ เช่น ภาพกราฟิก, ไฟล์เอกสาร, applet หรือ ActiveX
รูปแบบ	<pre><OBJECT CLASSID="implementation_info" CODEBASE="URL_of_object" CODETYPE="MINE_type" TYPE="data_MINE_type" STANDBY="message_while_loading" USEMAP="map_name" ALIGN="TEXTTOP MIDDLE TEXTMIDDLE BASELINE TEXTBOTTOM LEFT CENTER RIGHT" WIDTH="width_in_pixels_or_percentage" NAME="object_name" HEIGHT="height_pixels_or_percentage" HSPACE="pixels" VSPACE="pixels" BORDER="pixels"></pre>

</OBJECT>

แอตทริบิวต์

ALIGN	ใช้กำหนดการจัดวางข้อความที่อยู่ต่อท้ายออบเจกต์นั้น
WIDTH และ HEIGHT	กำหนดความกว้างและความสูงของพื้นที่ที่ใช้แสดงออบเจกต์ กำหนดเป็นหน่วยพิกเซล หรือเป็นเปอร์เซ็นต์
BORDER	กำหนดความหนาของกรอบ กำหนดเป็นหน่วยพิกเซล
HSPACE และ VSPACE	กำหนดระยะห่างระหว่างเส้นกรอบกับออบเจกต์ทั้งในแนวนอนและแนวตั้ง สามารถกำหนดเป็นหน่วยพิกเซล หรือเป็นเปอร์เซ็นต์ของขนาดพื้นที่ของโปรแกรมเสริมอย่างใดอย่างหนึ่งก็ได้
CLASSID	กำหนดชื่อโปรแกรมหรือไฟล์ที่ต้องการใช้
CODEBASE	กำหนดที่อยู่ของไฟล์ที่เก็บออบเจกต์นั้นตามแบบ URL
CODETYPE	กำหนดชนิดของไฟล์หรือโปรแกรม ตามแบบ MINE
STANDBY	กำหนดข้อความที่จะแสดงแทนตัวโปรแกรมเสริมหรือรูป ในกรณีที่เบราว์เซอร์ยังไม่สามารถแสดงโปรแกรมได้ หรือในขณะที่เบราว์เซอร์กำลังดาวน์โหลดโปรแกรมอยู่
TYPE	กำหนดชนิดข้อมูลที่จะถูกส่งให้ออบเจกต์ตามแบบ MINE
USEMAP	ใช้กำหนดให้ภาพนั้นทำหน้าที่เป็นอิมเมจแมพแบบทางฝั่งเบราว์เซอร์
แท็กที่เกี่ยวข้อง	<PARAM>

ก.2 แท็ก WML

<a>

ประเภท มีแท็กปิด

หน้าที่ สร้างลิงค์ โดยเปรียบเสมือนเป็นคำย่อ หรือชอร์ตคัตของแท็ก <anchor> กับแท็กคำสั่งของงาน <go> นั่นเอง

แอตทริบิวต์

href กำหนดชื่อ URI หรือการค์ที่ต้องการเชื่อมโยงไปถึง

title ใส่ข้อความที่เป็นชื่อ ที่จะอธิบายลิงค์อย่างย่อ

<access>

ประเภท มีแท็กปิด

หน้าที่ ควบคุมการเข้าถึงข้อมูลของผู้เข้าชม (access) หรือต้องการอนุญาตให้ผู้เข้าชมเฉพาะกลุ่มเข้ามาอ่านข้อมูลได้เฉพาะเว็บเพจบางเท่านั้น

แอตทริบิวต์

domain กำหนดชื่อของ โดเมนเนมหรือที่อยู่(IP Address) ของเอกสาร WML

path ชื่อของตำแหน่งหรือไฟล์เดอร์ ที่อยู่ภายใน โดเมนเนมที่ระบุไว้ใน domain

<anchor>

ประเภท มีแท็กปิด

หน้าที่ สร้างลิงค์ เพื่อเชื่อมโยงระหว่างการ์ดหรือหน้าเว็บเพจ ซึ่งสามารถทำงานได้ เช่นเดียวกับแท็ก <do> แต่ต่างกันตรงที่ข้อความในแท็ก <anchor> จะไม่ปรากฏในเมนู Options ของซอฟต์แวร์ แต่จะปรากฏบนหน้าจอโดยมีลักษณะเป็นลิงค์ที่เราสามารถเลือกได้เลย และเนื่องจากแท็กนี้ จะปรากฏบนหน้าจอ จึงต้องใส่แท็กนี้ระหว่างแท็ก<p>ด้วยเหมือนข้อความทั่วไป และในแท็กนี้จะต้องคำสั่งของงาน (Task) ที่เปลี่ยนหน้าการ์ดหรือหน้าเว็บเพจ เมื่อลิงค์ของแท็กนี้ถูกเลือก ซึ่งแท็กของงานที่สามารถอยู่ในแท็กนี้คือ go, prev และ refresh

แอตทริบิวต์

title ใส่ข้อความที่เป็นชื่อ ที่จะอธิบายลิงค์อย่างย่อ

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่ต้องการทำให้หนา

แอตทริบิวต์ ไม่มี

<big>

ประเภท มีแท็กปิด

หน้าที่ ใช้ครอบข้อความที่ต้องการทำให้ตัวอักษรมีขนาดใหญ่ขึ้น

แอตทริบิวต์ ไม่มี

ประเภท ไม่มีแท็กปิด

หน้าที่ กำหนดให้ขึ้นบรรทัดใหม่สำหรับข้อความที่อยู่ต่อจากแท็กนี้

แอตทริบิวต์ ไม่มี

<card>

ประเภท มีแท็กปิด

หน้าที่ กำหนดจุดเริ่มต้นและจุดสิ้นสุดของการ์ดในเคิล เด็กหนึ่งสามารถมีการ์ดได้หลายๆ การ์ด ซึ่งแต่ละการ์ดก็บรรจุข้อมูลและวิธีการตอบโต้กับผู้ใช้ที่แตกต่างกันไป

แอตทริบิวต์

title ใช้สำหรับใส่ข้อความที่เป็นชื่อ ที่จะอธิบายภาพรวมของข้อมูลในการ์ด

id ใช้สำหรับกำหนดชื่อให้กับการ์ด เพื่ออ้างอิงการ์ดอื่นได้

<do>

ประเภท มีแท็กปิด

หน้าที่ ใช้ในการช่วยให้ผู้ใช้สามารถเข้าถึง และเชื่อมโยงจากการ์ดที่แสดงอยู่ในขณะนั้นไปสู่การ์ด หรือ URI อื่น ๆ ได้ แท็ก <do> จะทำหน้าที่เชื่อมต่อระหว่างสิ่งที่แสดงออกมาบนหน้าจอภาพกับงานที่เราต้องการให้หน้า WAP ทำ โดยแท็ก <do> นี้ สามารถใช้ได้กับระดับเคิลและการ์ด

แอตทริบิวต์

type เป็นตัวแสดงแก่เครื่องโทรศัพท์มือถือ (WAP บราวเซอร์) ให้ทราบว่าแท็กนั้นถูกใช้เพื่อทำงานอะไร และยังช่วยให้บราวเซอร์สร้างอินเทอร์เฟซที่เหมาะสมกับงานนั้นๆ ด้วย ซึ่งแอตทริบิวต์นี้ต้องมีเสมอ มิฉะนั้นจะเกิด Syntax error

ค่าแอตทริบิวต์	ความหมาย
Accept	เป็นค่าที่ใช้กับมากมีไว้สำหรับแสดงการยอมรับให้ทำงานตามที่กำหนด
Prev	เป็นปุ่มแสดงถึงความต้องการย้อนกลับไปยังหน้าก่อน
Help	เป็นปุ่มแสดงถึงความต้องการความช่วยเหลือ
Reset	เป็นปุ่มแสดงถึงตัวเลือก หรือการทำงานอื่น ๆ ที่สามารถเพิ่มเติมได้
Option	เป็นปุ่มแสดงถึงการลบ หรือเริ่มต้นการทำงานใหม่
Delete	เป็นปุ่มแสดงถึงการลบวัตถุหรือตัวเลือกต่างๆ
Unknown	เป็นชนิดที่เราไม่ได้กำหนดลงไป หรือเกิดจากการพิมพ์ค่าของแอตทริบิวต์
""(empty string)	ที่ไม่มีนิยามหรือพิมพ์ผิด มีความหมายเหมือนกับ unknow
x-* , X-*	เป็นชนิดที่ใช้ทดลอง
vna.* , VNA.*	ผู้ขายหรือผู้เสนอแนะกำหนดชนิดของ vnd.co-type ซึ่ง co คือ ชื่อบริษัท และ type เป็น do-type

label	ใช้กำหนดชื่อของลิงค์ ที่แสดงออกมาบนหน้าจอในเมนูได้
option	เป็นแอตทริบิวต์ที่อนุญาตให้ WAP บราวเซอร์ คิดว่าแท็กนี้เป็นตัวเลือก ซึ่งจะนำมาใช้หรือไม่ก็ได้ โดยสามารถกำหนดค่าเป็น "true" หรือ "false" ถ้ากำหนดเป็น "true" เครื่องอาจจะไม่นำแท็กนี้มาใช้ก็ได้
name	ใช้กำหนดชื่อของแท็ก <do> เพื่ออ้างอิง
	
ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการเน้นหรือให้ความสำคัญ
แอตทริบิวต์	ไม่มี

<fieldset>

ประเภท	มีแท็กปิด
หน้าที่	ใช้จัดกลุ่มให้กับส่วนรับข้อมูล ซึ่งจะทำให้บราวเซอร์แสดงส่วนรับข้อมูลต่างๆ ได้ดีขึ้น นอกจากนี้จะมีแท็กที่กำหนดให้รับข้อมูลแล้ว ยังสามารถใส่แท็กต่างๆ ได้ อีกดังต่อไปนี้
	- แท็ก fieldset (fieldset ซ้อน fieldset) - แท็ก a - แท็ก img - แท็ก tab

- แท็กbr
- แท็กการตกแต่งตัวอักษร เช่น em,b,u

แอตทริบิวต์

title กำหนดชื่อให้กับแท็ก

<go>

ประเภท มีแท็กปิด

หน้าที่ ใช้กำหนดให้ WAP บราวเซอร์เปลี่ยนหน้าของเว็บเพจ ไปตามตำแหน่งที่กำหนดขึ้นโดยแอตทริบิวต์ href ซึ่งตำแหน่งที่กำหนดใน href อาจจะเป็น URI หรือชื่อของ card หรือ deck ในไซตนั้นก็ได้

แอตทริบิวต์

href ใช้กำหนด URI หรือชื่อของการค์ที่ต้องการเชื่อมต่อไปถึง

sendreferer สามารถกำหนดค่าเป็น trueหรือ false ก็ได้ ซึ่งแอตทริบิวต์ที่อนุญาตให้เครื่อง PDA ส่ง URI ของหน้า WAP เพจนั้นไปยังเซิร์ฟเวอร์ เมื่อทำการติดต่อกับเครื่องเซิร์ฟเวอร์ โดยปกติเราจะกำหนดค่าแอตทริบิวต์นี้เป็น true เมื่อเราต้องการควบคุมบุคคลที่จะเข้าถึงข้อมูลในเครื่องเซิร์ฟเวอร์

method กำหนดวิธีการของการส่งค่าผ่านโปรโตคอล HTTP ซึ่งค่าได้ 2 อย่าง คือ POST หรือ GET

- GET เป็นวิธีการส่งค่าของข้อมูลไปพร้อมกับค่าของ URI โดยมีเครื่องหมาย ? กั้นระหว่างชื่อของตัวแปรที่ทำการส่งกับ URI โดยมีเครื่องเซิร์ฟเวอร์จะสามารถรับค่าเหล่านี้ได้โดยการใช้คำสั่ง QUERY_STRING
- POST เป็นวิธีการส่งข้อมูลมาทาง stdin ซึ่งมีหน่วยเป็นไบต์ และจำนวนไบต์ของข้อมูลที่จะถูกส่งเก็บค่าไว้ในตัวแปร CONTENT_LENGTH การส่งข้อมูลด้วยวิธีนี้จะเป็นการป้องกันมิให้ข้อมูลอ่านง่ายกว่าวิธีการส่วแบบ GET เพียงเล็กน้อย

accept-charset กำหนดคิสต์ของการเข้ารหัสตัวอักษรที่เซิร์ฟเวอร์สามารถรองรับได้

<head>

ประเภท มีแท็กปิด

หน้าที่ เป็นส่วนหัวของเอกสาร เป็นส่วนที่เก็บข้อมูลและรายละเอียดที่มีความสัมพันธ์กับเอกสาร WML ทั้งเอกสาร (deck) แท็กที่สามารถมีได้ภายในแท็ก <head> คือ <access> หรือ <meta>

แอตทริบิวต์ ไม่มี

<i>

ประเภท	มีแท็กปิด
หน้าที่	ใช้ครอบข้อความที่ต้องการทำให้เป็นตัวอักษรเอียง
แอตทริบิวต์	ไม่มี

ประเภท	ไม่มีแท็กปิด
หน้าที่	แสดงรูปภาพลงในหน้า WAP
แอตทริบิวต์	
sre	แสดงตำแหน่งของที่อยู่และชื่อของไฟล์รูปภาพที่ต้องการนำมาใช้ หรือที่เรียกว่า URI ของรูปภาพ
alt	ทางเลือก (alternative) ของการแสดงผลของรูปภาพ ในกรณีที่เบราว์เซอร์ไม่สามารถแสดงภาพที่ต้องการออกมาได้ เบราว์เซอร์จะแสดงข้อความที่กำหนดในแอตทริบิวต์นี้ออกมาแทน
localsre	ใช้ระบุไอคอนที่ถูกกำหนดไว้แล้ว ถ้าแอตทริบิวต์นี้ระบุไอคอนถูกต้อง จะถูกแทนแอตทริบิวต์ sre และ alt
align	ทำหน้าที่จกวางตำแหน่งของรูปภาพกับข้อความ โดยมีกำหนดได้ 3 แบบคือ - การวางข้อความเสมอขอบบนของรูปภาพ (top) - การวางข้อความในตำแหน่งตรงกลางของรูปภาพ (middle) - การวางข้อความเสมอขอบล่างของรูปภาพ (bottom)
vspace	กำหนดขนาดของช่องว่างที่ต้องการใส่ระหว่างขอบของรูปภาพในตำแหน่งบนและล่าง
hspace	กำหนดขนาดของช่องว่างที่ต้องการใส่ระหว่างขอบของรูปภาพในตำแหน่งซ้ายและขวา
height	ทำหน้าที่จองขนาดความสูงพื้นที่ของเบราว์เซอร์ล่วงหน้าเพื่อทำการโหลดรูป นั้น การกำหนดพื้นที่สำหรับการดาวน์โหลดรูปจะมีประโยชน์ ก็เบราว์เซอร์สามารถแสดงข้อความต่อจากพื้นที่ ที่เรากำหนดได้เลย โดยไม่ต้องรอให้เบราว์เซอร์โหลดรูปเสร็จ
width	ทำหน้าที่จองขนาดความกว้างพื้นที่ของเบราว์เซอร์ล่วงหน้าเพื่อทำการ โหลดรูป

<input/>

ประเภท	ไม่มีแท็กปิด
หน้าที่	ใช้รับข้อมูลที่เป็นตัวอักษร ซึ่งพิมพ์โดยผู้ใช้
แอตทริบิวต์	

name	กำหนดชื่อตัวแปรที่ใช้สำหรับเก็บข้อมูลที่ผู้ใช้ใส่เข้ามา ดังนั้นจึงต้องกำหนดค่าให้กับแอตทริบิวต์นี้ทุกครั้ง
value	กำหนดค่าเริ่มต้นหรือค่าอัตโนมัติให้ตัวแปรที่ระบุในแอตทริบิวต์ name และเป็นค่าที่จะแสดงหน้าจอครั้งแรก เมื่อผู้ใช้ได้เข้ามาในหน้านี้
type	ใช้กำหนดการแสดงผลตัวอักษรในช่องรับข้อมูล สามารถ กำหนดได้ 2 ค่าคือ - text หมายถึง กำหนดให้การแสดงผลตัวอักษรเป็นไปตามปกติ - password หมายถึง กำหนดให้แสดงผลตัวอักษรไม่ตรงตามที่ผู้ใช้พิมพ์เข้ามา โดยทั่วไปแล้วตัวอักษรที่บราวเซอร์แสดงคือเครื่องหมาย *
title	ใช้สำหรับตั้งชื่อแท็กนี้
maxlength	ใช้กำหนดจำนวนตัวอักษรที่มากที่สุดที่ผู้ใช้สามารถใส่ได้ ถ้าหากไม่กำหนดค่าให้กับแอตทริบิวต์นี้ บราวเซอร์จะถือว่าจำนวนตัวอักษรที่ใส่ได้ไม่จำกัดจำนวน
format	ใช้กำหนดรูปแบบในการรับข้อมูล

ค่าแอตทริบิวต์	ความหมาย
A	อนุญาตให้ใส่ได้เฉพาะอักษรตัวใหญ่ และสัญลักษณ์ต่าง ที่ไม่ใช่ตัวเลข
a	อนุญาตให้ใส่ได้เฉพาะอักษรตัวเล็ก และสัญลักษณ์ต่าง ที่ไม่ใช่ตัวเลข
N	อนุญาตให้ใส่ได้เฉพาะตัวเลข
X	อนุญาตให้ใส่ได้เฉพาะอักษรตัวใหญ่
x	อนุญาตให้ใส่ได้เฉพาะอักษรตัวเล็ก
M	อนุญาตให้ใส่อักษรอะไรก็ได้ แต่บราวเซอร์จะถือว่าเป็นอักษรตัวใหญ่
m	อนุญาตให้ใส่อักษรอะไรก็ได้ แต่บราวเซอร์จะถือว่าเป็นอักษรตัวเล็ก
*f	อนุญาตให้ใส่อักษรเป็นจำนวนเท่าไรก็ได้ โดยที่ f คือค่าต่างๆ ที่ผ่านมา เช่น *A, *a, *N เป็นต้น
nf	อนุญาตให้ใส่อักษรเป็นจำนวนเท่าค่า n โดย n มีค่าระหว่าง 1-9 และ f คือค่าต่างๆ ที่ผ่านมา เช่น 3a, 5M เป็นต้น
\c	กำหนดให้แสดงผลตัวอักษรหลังเครื่องหมาย \ เช่น \- ในช่องใส่ข้อมูลจะมีเครื่องหมาย (-) แสดง

emptyok	สำหรับอินพุตที่อนุญาตให้ผู้ใช้ใส่ค่าหรือไม่ให้ใส่ค่าก็ได้ ค่าที่สามารถกำหนดให้แอตทริบิวต์นี้คือ - true หมายความว่า ผู้ใช้ไม่ต้องใส่ข้อมูลก็ได้ - false หมายความว่า ผู้ใช้จะต้องใส่ข้อมูล
tabindex	ใช้กำหนดลำดับที่ของแท็ก ในกรณีที่ผู้ใช้คลิกปุ่มแท็บ

size	ใช้กำหนดความกว้างของช่องใส่ข้อมูล โดยจะวัดเป็นจำนวนตัวอักษร
<meta/>	
ประเภท	ไม่มีแท็กปิด
หน้าที่	เป็นส่วนที่ให้ข้อมูลทั่วไปที่เกี่ยวข้องกับเอกสาร WML ทั้งเอกสาร (deck) ซึ่งข้อมูลส่วนนี้ไม่สามารถถูกเก็บไว้ในส่วนของเนื้อหาภายในการค์ได้ งานหลักๆ ของแท็ก <meta> มีดังต่อไปนี้ 1. กำหนดคำอธิบาย และคำที่แสดงความหมายของ WAP เพจ 2. ใช้ป้องกันมิให้เบราว์เซอร์ดาวน์โหลดเอกสาร WML จากแคช (cache) 3. เปลี่ยนค่า URI หรือต้องการดึงข้อมูลจากเว็บเซิร์ฟเวอร์ เมื่อถึงเวลาที่กำหนดโดยอัตโนมัติ
แอตทริบิวต์	
content	ในการกำหนดคุณสมบัติของเอกสารที่ต้องการส่งให้เซิร์ฟเวอร์ หรือใช้ในการกำหนดค่าให้คุณสมบัติของเอกสารที่กำหนดในแอตทริบิวต์ name
name	ใช้ในการระบุชื่อของคุณสมบัติของเอกสาร เช่น keyword,description หรือ author
http-equiv	จะใช้แทนแอตทริบิวต์ name เพื่อจะนำค่าของข้อมูลไปสร้าง HTTP header ซึ่ง เป็นข้อมูลที่สำคัญที่รับส่งข้อมูลโดยโปรโตคอล HTTP
forua	มีค่าได้ 2 ค่า คือ “true” หรือ “false” ซึ่งจะกำหนดค่าเป็นแท็ก เมื่อต้องการให้เกิด เว็ลตข้อมูลของแท็ก <meta> ออกเมื่อทำการส่งข้อมูลไปยังเครื่องโทรศัพท์ของผู้รับ และมีค่าเป็นจริง เมื่อเราบังคับให้เกิดเว็บต้องส่งค่าของแท็ก <meta> ไปยังเครื่องของผู้รับ
scheme	ใช้ในการกำหนดรูปแบบ หรือมาตรฐานที่ใช้ในการตีความที่อยู่ในแอตทริบิวต์ โดยค่าของ scheme ที่จะใช้ อาจจะแตกต่างกันไปในแต่ละความหมายของแท็ก <meta>
<noop>	
ประเภท	มีแท็กปิด
หน้าที่	ใช้เพื่อทำการยกเลิกการทำงานของเทมเพลต (template) ในระดับเล็ก โดยมากจากคำว่า “no operation” หรือแปลว่า “ไม่ทำอะไรเลย”
แอตทริบิวต์	ไม่มี
<onevent>	
ประเภท	มีแท็กปิด

หน้าที่

เป็นแท็กของเหตุการณ์ที่สามารถกำหนดชนิดของเหตุการณ์เป็น `onenterforward`, `onenterbackward` หรือ `ontimer` ได้ พร้อมทั้งสามารถกำหนดคำสั่งของงาน (Task) ที่จะใช้ได้อีกด้วย และการใช้แท็ก `<onevent>` นี้ มีประโยชน์มากในการควบคุมการทำงานของตัวเครื่องโทรศัพท์มือถือได้มากขึ้นเพราะว่า แท็ก `<onevent>` นี้จะถูกใช้งานเป็นแท็กที่ทำหน้าที่กำกับแท็กคำสั่งของงาน (เช่น `go`, `prev`, `refresh` หรือ `noop`) อีกทีจึงมีประโยชน์มากกว่าเหตุการณ์ (event) เพื่อแอตทริบิวต์ในแท็ก `<card>` หรือ `<template>` อย่างเดียว เพราะสามารถกำหนดคำสั่งที่ต้องการทำได้มากขึ้น

แอตทริบิวต์

type

ใช้กำหนดเหตุการณ์ที่ใช้สำหรับเปลี่ยน URI

- `ontimer`

เหตุการณ์ `ontimer` จะใช้เป็นแอตทริบิวต์ในแท็ก `<card>` และ `<template>` เพื่อกำหนดหน้าเว็บเพจจะเปลี่ยนไปสู่การ์ดใบใหม่ หรือที่อยู่ใหม่ชื่ออะไร ในระยะเวลาที่กำหนด ซึ่งจะใช้คู่กับแท็ก `<timer>`.

- `onenterforward`

เหตุการณ์ `onenterforward` จะเกิดขึ้นเมื่อผู้ใช้เปิดการ์ดที่มีแอตทริบิวต์นี้ขึ้นมาใช้งาน โดยอาจจะเข้ามาโดยคำสั่ง `<go>` หรือวิธีอื่นๆก็ได้ และเหตุการณ์ `onenterforward` นี้ก็เป็นแอตทริบิวต์หนึ่งของแท็ก `<card>` และ `<template>` ซึ่งจะเป็นตัวกำหนดค่าของ URI ที่ต้องการจะไป เมื่อการ์ดนี้ถูกเปิดขึ้นมา

- `onenterbackward`

เป็นเหตุการณ์ที่เกิดขึ้นเช่นเดียวกับเหตุการณ์ของ `onenterforward` แต่จะแตกต่างกันตรงที่เหตุการณ์นี้จะเกิดขึ้น เมื่อการ์ดนี้ถูกเปิดโดยคำสั่ง `<prev/>` หรือโดยวิธีอื่นๆก็ได้ ซึ่งเกิดจากการที่ผู้เยี่ยมชมอ่านค่า URI ที่ได้รับจากที่เก็บไว้ใน history นั้นเอง

`<option>`

ประเภท

มีแท็กปิด

หน้าที่

ใช้ร่วมกับแท็ก `<select>`

แอตทริบิวต์

value

ใช้กำหนดค่าของตัวเลือก ซึ่งค่านี้จะเป็นค่าที่ตัวแปรในแอตทริบิวต์ `name` ของแท็ก

select

รับไปเมื่อผู้ใช้เลือกตัวเลือกนี้

index

ใช้กำหนดลำดับของตัวเลือก

title

ใช้ตั้งชื่อให้กับแท็ก

onpick เป็นเหตุการณ์ที่จะทำงานเมื่อผู้ใช้เลือกหรือยกเลิกการเลือกเมนูรับค่า (input) เมื่อตัวเลือกที่มีแอตทริบิวต์นี้ถูกเลือก เหตุการณ์ onpick นี้จะทำหน้าที่กำหนด URI ของหน้าเว็บเพจใหม่ที่จะนำมาแสดงผลนั่นเอง ใช้สำหรับลิงก์ไปยังหน้าอื่นๆ เมื่อมีการเลือกตัวเลือก หรือการยกเลิกการเลือกเกิดขึ้น ในกรณีที่มีการเลือกได้เลือกเพียงตัวเลือกเดียว แอตทริบิวต์นี้จะทำงาน (ลิงก์ไปยังหน้าที่กำหนดไว้) เฉพาะมีการเลือกเกิดขึ้นเท่านั้น ส่วนในกรณีที่สามารถเลือกได้หลายตัวเลือก แอตทริบิวต์นี้จะทำงานทั้งเมื่อมีการเลือก และยกเลิกการเลือกเกิดขึ้น

<optgroup>

ประเภท มีแท็กปิด

หน้าที่ จัดกลุ่มให้กับตัวเลือก จะช่วยให้เบราว์เซอร์แสดงตัวเลือกต่างๆ ได้ง่ายขึ้นมีการใช้ร่วมกับแท็กselect และ option

แอตทริบิวต์

title กำหนดชื่อให้กับกลุ่มตัวเลือก

<p>

ประเภท มีแท็กปิด

หน้าที่ ระบุพารากราฟของข้อมูลสู่จอภาพ โดยเป็นตัวกำหนดการจัดวางตัวอักษร และการแสดงตัวอักษร

แอตทริบิวต์

align ใช้กำหนดการจัดวางตัวอักษร ซึ่งสามารถกำหนดได้ 3 แบบ คือ ซิดซ้าย ซิดขวา และ ตรงกลาง แต่ถ้าหากว่าไม่ได้กำหนดค่าให้กับแอตทริบิวต์นี้ เบราวเซอร์จะใช้ค่าอัตโนมัติคือ ซิดซ้าย

mode ใช้สำหรับจัดการกับข้อความในแต่ละบรรทัดว่าจะให้ขึ้นบรรทัดใหม่หรือไม่ ถ้าข้อความนั้นยาวเกินกว่าหน้าจอจะแสดงได้ในหนึ่งบรรทัด ค่าของแอตทริบิวต์นี้คือ wrap และ nowrap

wrap หมายถึง กำหนดให้การแสดงตัวอักษร จะต้องขึ้นบรรทัดใหม่ ถ้าข้อความมีความยาวมากกว่าที่จะแสดงได้ในหนึ่งบรรทัด

nowrap หมายถึง กำหนดให้การแสดงตัวอักษร ไม่ต้องขึ้นบรรทัดใหม่ ถ้าข้อความมีความยาวมากกว่าที่จะแสดงได้ในหนึ่งบรรทัด

ถ้าหากเราไม่ได้กำหนดค่าให้กับแอตทริบิวต์นี้ ค่าที่เบราว์เซอร์จะใช้ก็คือ ค่าที่เรากำหนดไว้ในครั้งสุดท้ายในแท็ก <p> ที่อยู่ในการ์ดนั้น หรือถ้าหากเราไม่เคยกำหนดค่านี้มาก่อนเลย ค่าที่เบราว์เซอร์ใช้คือ wrap

<postfield>**ประเภท** มีแท็กปิด**หน้าที่** ส่งค่าตัวแปรไปยังเพจอื่นและสคริปต์ ทั้งแบบ GET และ POST โดยการส่งแบบ GET นั้นจะส่งโดยใช้แท็ก <postfield> หรือไม่ก็ได้ แต่การส่งแบบ GET นั้นจะต้องใช้แท็ก <postfield>**แอตทริบิวต์****name** ชื่อตัวแปรที่จะส่งค่าที่ผู้ใช้เลือก**value** ค่าตัวแปรที่จะถูกส่งกลับไปยังไฟล์ที่ทำหน้าที่เป็นโปรแกรมจัดการข้อมูล**<prev/>****ประเภท** ไม่มีแท็กปิด**หน้าที่** สร้างลิงก์กลับไปหน้าเก่า เพื่อให้สามารถย้อนกลับไปยังหน้าที่ผ่านมาแล้ว เมื่อเราคลิกที่ตัวเลือกเพื่อใช้แท็กนี้ ค่าของ URI ที่อยู่ใน history จะถูกนำออกมาอ่าน และเว็บเพจจะเปลี่ยนเป็นหน้าที่นำออกมาจาก history**แอตทริบิวต์** ไม่มี**<refresh/>****ประเภท** ไม่มีแท็กปิด**หน้าที่** ใช้ในการ โหลดหน้าเว็บเพจขึ้นมาใหม่ หรือเพื่อโหลดข้อมูลรวมทั้งค่าของตัวแปรล่าสุดออกมาแสดงอีกครั้งหนึ่ง**แอตทริบิวต์** ไม่มี**<select>****ประเภท** มีแท็กปิด**หน้าที่** แสดงตัวเลือกในลักษณะเมนู ใช้ร่วมกับแท็ก <option>**แอตทริบิวต์****name** ใช้สำหรับตั้งชื่อตัวแปรที่จะรับค่าตัวเลือกที่ผู้ใช้เลือก**iname** ใช้สำหรับตั้งชื่อตัวแปรที่ใช้เก็บค่าดัชนีของตัวเลือกที่ถูกเลือก**value** ใช้สำหรับกำหนดค่าเริ่มต้นให้กับตัวแปรที่ตั้งไว้ในแอตทริบิวต์ name ในกรณีที่กำหนดการเลือกให้สามารถเลือกได้หลายตัวเลือก ค่าแต่ละค่าต้องกันด้วยเครื่องหมายเซมิโคลอน (;)

ivalue	ใช้สำหรับกำหนดค่าดัชนีของตัวเลือกเริ่มต้นให้กับตัวแปรที่ตั้งไว้ในแอตทริบิวต์
iname	ถ้าหากเป็นการเลือกได้หลายตัวเลือก ค่าของดัชนีจะต้องกันด้วยเครื่องหมายเซมิโคลอน (;)
title	ใช้ตั้งชื่อให้กับแท็ก
multiple	ใช้สำหรับกำหนดจำนวนตัวเลือกที่สามารถเลือกได้ โดยค่าที่สามารถกำหนดได้คือ true หรือ false - true หมายความว่า ผู้ใช้สามารถเลือกได้หลายตัวเลือก - false หมายความว่า ผู้ใช้สามารถเลือกได้เพียงตัวเลือกเดียว
tabindex	ใช้กำหนดลำดับที่ของแท็บ ในกรณีที่ผู้ใช้กดปุ่มแท็บ

<setvar>

ประเภท

มีแท็กปิด

หน้าที่

กำหนดตัวแปร ซึ่งจะใช้เพียงแคภายในแท็ก<refresh>, <go> เท่านั้น

แอตทริบิวต์

name

ชื่อตัวแปรที่กำหนด

value

ค่าของตัวแปรที่ระบุ

<small>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการทำให้ตัวอักษรมีขนาดเล็กลง

แอตทริบิวต์

ไม่มี

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการเน้นหรือให้ความสำคัญเป็นพิเศษ

แอตทริบิวต์

ไม่มี

<table>

ประเภท

มีแท็กปิด

หน้าที่

สร้างตาราง เพื่อนำเสนอข้อมูลในลักษณะตาราง โดยเป็นตัวกำหนดจุดเริ่มต้นและจุดสิ้นสุดของตาราง

แอตทริบิวต์

title

กำหนดชื่อตาราง

align	ใช้สำหรับจัดวางตัวอักษรหรือรูปภาพที่แสดงในแต่ละคอลัมน์ ค่าที่กำหนดให้แอตทริบิวต์ ได้แก่ C,L,R C หมายถึง จัดให้อยู่กึ่งกลางคอลัมน์ L หมายถึง จัดให้อยู่ชิดซ้ายคอลัมน์ R หมายถึง จัดให้อยู่ชิดขวาคอลัมน์ การกำหนดการจัดวางนี้สามารถระบุให้แต่ละคอลัมน์ได้ด้วยการระบุค่าเรียงกันไป
columns	ใช้สำหรับระบุจำนวนคอลัมน์ที่มีตาราง ซึ่งจะต้องกำหนดค่าให้แอตทริบิวต์นี้ทุกครั้ง ถ้าหากจำนวนแท็ก td มีน้อยกว่าค่าที่กำหนดให้กับแอตทริบิวต์นี้ บราวเซอร์จะทำการเพิ่มคอลัมน์ว่างให้โดยอัตโนมัติ แต่ถ้าหากจำนวนแท็ก td มีมากกว่าค่าที่เรากำหนด บราวเซอร์จะใส่คอลัมน์ที่เกินมาลงทางซ้ายของตาราง
<td> ประเภท หน้าที่	มีแท็กปิด ใช้กำหนดจุดเริ่มต้นและจุดสิ้นสุดของคอลัมน์หนึ่งคอลัมน์ในตาราง ข้อมูลต่างๆ ที่ต้องการนำเสนอจะต้องอยู่ภายในแท็ก td ดังนั้นภายในแท็ก td จึงมีแท็กย่อยอื่นๆ เพื่อใช้ควบคุมการแสดงผลภาพ หรือการโต้ตอบกับผู้ใช้แท็กเหล่านั้น ได้แก่ แท็ก a ,แท็ก anchor ใช้สำหรับเชื่อมโยงไปยังข้อมูลส่วนอื่นๆ หรือการค์อื่นๆ แท็ก br ใช้ขึ้นบรรทัดใหม่ แท็ก img ใช้สำหรับแสดงรูปภาพ
แอตทริบิวต์	ไม่มี
<template> ประเภท หน้าที่	มีแท็กปิด กำหนดคำสั่งงาน (task) หรือ เหตุการณ์ (event) ต่างๆ ซึ่งคำสั่งของเหตุการณ์หรืองานที่เขียนในเทมเพลตนี้จะสามารถนำมาใช้ได้กับทุกการ์ดในเค้นนั้น ยกเว้นคำสั่งของงานนั้นจะถูกยกเลิกโดยคำสั่งอื่นที่เขียนขึ้นมาภายในการ์ดอีกทีภายในเทมเพลตนี้สามารถมีแท็ก<do> เพื่อทำหน้าที่ใส่คำสั่งงาน หรือแท็ก<onevent> เพื่อกำหนดคำสั่งงานเมื่อมีเหตุการณ์ต่างๆเกิดขึ้น
แอตทริบิวต์	ไม่มี
<timer/> ประเภท	ไม่มีแท็กปิด

หน้าที่

เป็นแท็กที่ช่วยในการกำหนดระยะเวลาที่เราต้องการนั่นเอง การนับเวลาของ timer จะเป็นการนับถอยหลังตั้งแต่เราเริ่มเปิดการ์ดที่มีแท็กนี้ขึ้นมา และจะหยุดเวลาเมื่อออกจากการ์ดใบนั้น timer จะนับเวลาถอยหลังไปเรื่อยๆ จนกระทั่งหมดเวลา หรือได้เวลาเท่ากับ 0 จากนั้นจึงจะทำการบังคับให้เหตุการณ์ที่กำหนดในแท็ก ontimer ทำงาน โดคนที่หนึ่งการ์ด สามารถมีแท็ก timer ได้เพียงแค่ 1 แท็กเท่านั้น

แอตทริบิวต์

name กำหนดชื่อของตัวแปรที่ใช้เก็บค่าของเวลา
value กำหนดค่าของเวลาที่เราต้องการ ซึ่งการกำหนดเวลาของ timer จะมีค่าเป็นหน่วยละ 1/10 วินาที และแก timer นี้ จะต้องมียแอตทริบิวต์นี้

<tr>

ประเภท

มีแท็กปิด

หน้าที่

ใช้กำหนดจุดเริ่มต้นและจุดสิ้นสุดของแถวหนึ่งแถวในตาราง

แอตทริบิวต์

ไม่มี

<u>

ประเภท

มีแท็กปิด

หน้าที่

ใช้ครอบข้อความที่ต้องการขีดเส้นใต้

แอตทริบิวต์

ไม่มี

<wml>

ประเภท

มีแท็กปิด

หน้าที่

ระบุจุดเริ่มต้นและจุดสิ้นสุดของเด็ก

แอตทริบิวต์

ไม่มี

ภาคผนวก ข

การติดตั้งโปรแกรม

ข.1 ความต้องการของระบบ

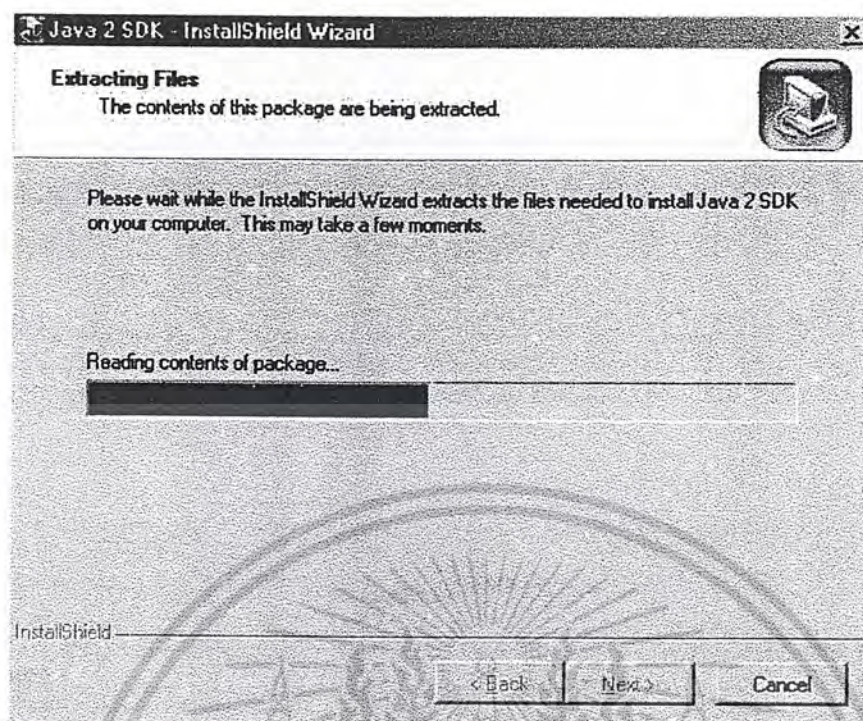
ระบบปฏิบัติการ (OS)	: วินโดว์ 2000/ME
หน่วยประมวลผลกลาง	: ทำงานได้ดีที่ความเร็วตั้งแต่ 450 เมกะเฮิร์ต ขึ้นไป
หน่วยความจำ (RAM)	: 256 MB หรือ มากกว่า
หน่วยความจุ (Hard Disk)	: 5 GB ขึ้นไป
ความละเอียดจอภาพ (Monitor)	: 65,000 สี หรือมากกว่า
การเชื่อมต่อโครงข่าย	: เครือข่ายอินเทอร์เน็ต
การติดตั้งโปรแกรมฝั่งเซิร์ฟเวอร์	: JDK1.4, เว็บเซิร์ฟเวอร์ที่รองรับการทำงานเซิร์ฟเล็ท
การติดตั้งโปรแกรมฝั่งไคลเอนต์	: แอ็พบราวเซอร์

ข.2 การติดตั้งระบบเพื่อใช้งาน

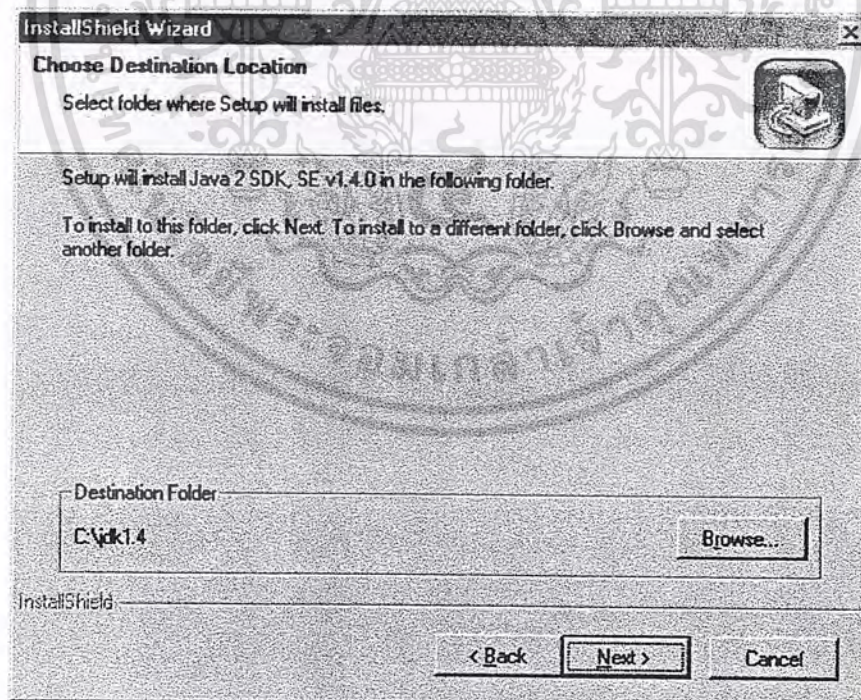
ข.2.1 การดาวน์โหลดและติดตั้ง Java Development Kit (JDK) 1.4

ดาวน์โหลดไฟล์ติดตั้ง โดยเลือกใช้ JDK เวอร์ชัน 1.4 สำหรับวินโดว์ ชื่อ “j2sdk1_4_0-win.exe” ในกรณีที่เครื่องคอมพิวเตอร์เป็นแพลตฟอร์มอื่นก็ให้เลือกไฟล์ของ JDK ตามแพลตฟอร์มที่ตรงกับเครื่องคอมพิวเตอร์มาใช้แทน การดาวน์โหลดสามารถทำได้โดย ดาวน์โหลดผ่านทางบราวเซอร์จากเว็บไซต์ของบริษัท Sun Microsystem (<http://java.sun.com/j2se/1.4/>) หรืออาจใช้บริการFTPจากบราวเซอร์ก็ได้ โดยกำหนด URL เป็น <ftp://ftp.javasoft.com>

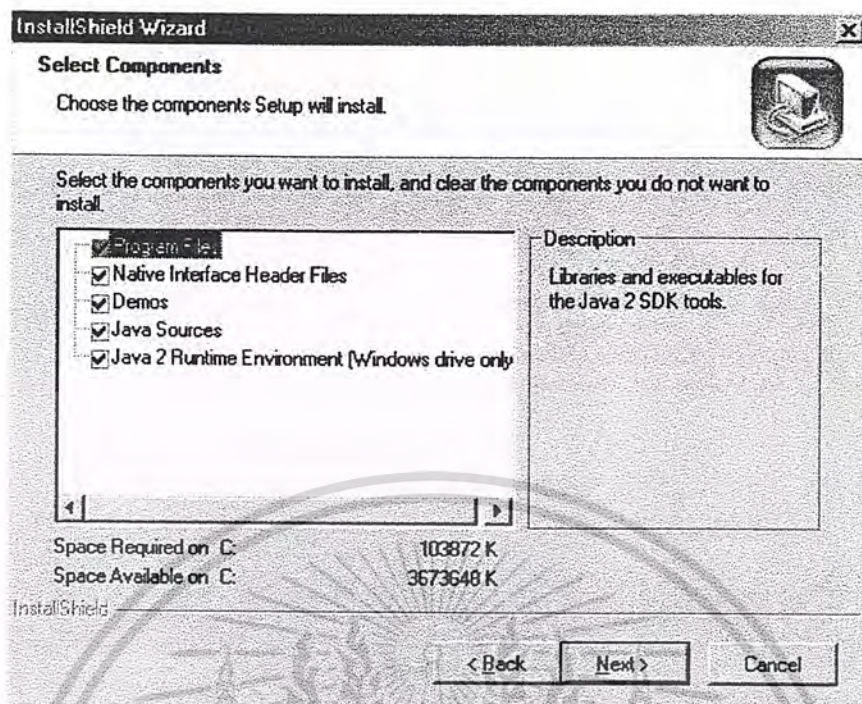
ไฟล์ j2sdk1_4_0-win.exe ที่ดาวน์โหลดได้มาจะมีไฟล์ที่ใช้งานถูกแพ็คเกจรวมกันอยู่ ดังนั้นจึงต้องแตกไฟล์เหล่านั้นออกมา โดยรันไฟล์ที่ได้มา จากนั้นการขยายไฟล์จะถูกทำให้โดยอัตโนมัติ แต่ถ้าหากมีการติดตั้ง JDK เวอร์ชันเก่าไว้แล้วในเครื่อง ให้ลบก่อนทำการขยายไฟล์



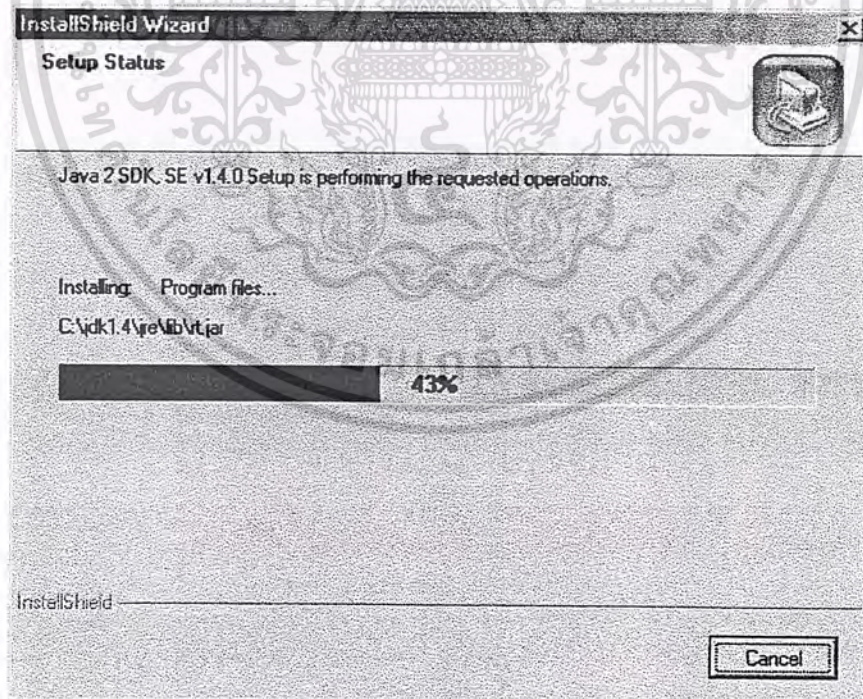
รูปที่ น.1 แสดงการขยายไฟล์ JDK 1.4



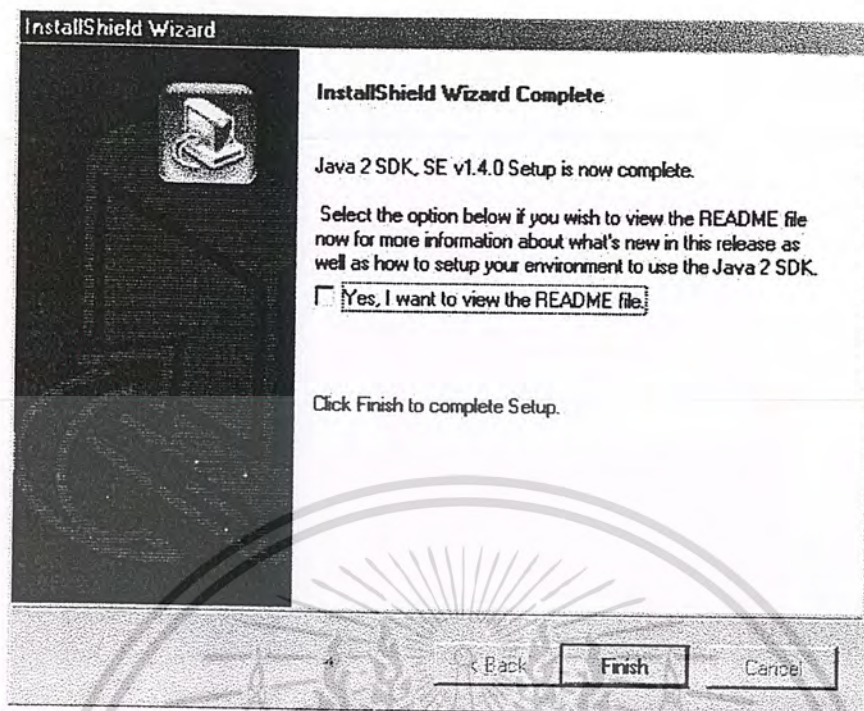
รูปที่ น.2 แสดงขั้นตอนการระบุพาธในการติดตั้ง JDK 1.4



รูปที่ น.3 แสดงขั้นตอนการเลือกคอมโพเนนต์ต่างๆ ในการติดตั้ง JDK 1.4



รูปที่ น.4 แสดงกระบวนการคัดลอกไฟล์ในระหว่างการทำติดตั้ง JDK 1.4



รูปที่ 4.5 แสดงการเสร็จสิ้นการติดตั้ง JDK 1.4

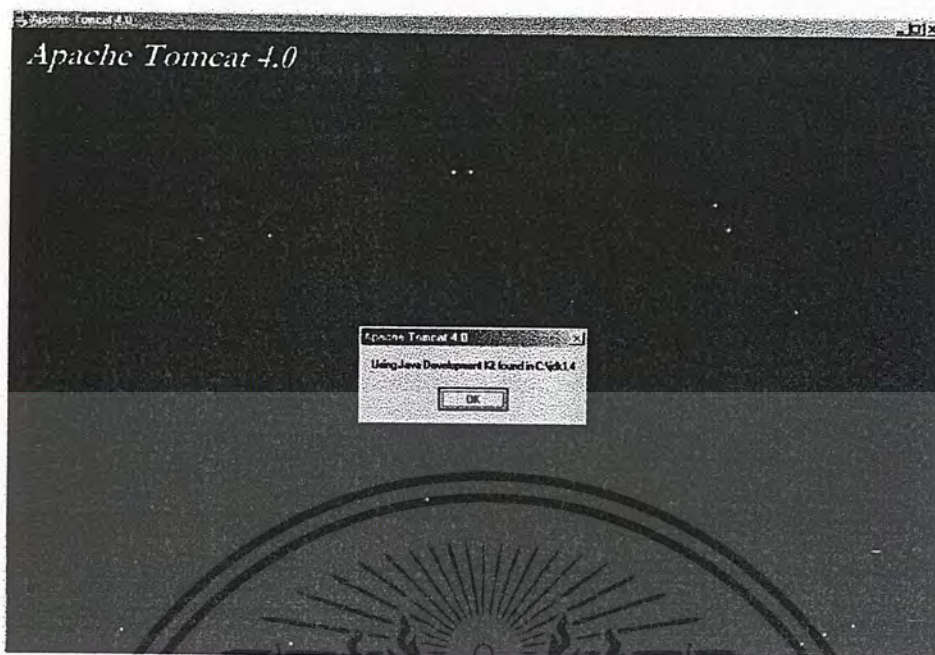
การปรับแต่งสภาพแวดล้อมให้เหมาะสม เช่น การเพิ่มเติมพารามิเตอร์ในไฟล์ autoexec.bat ของระบบที่ใช้งานอยู่ เพื่อเรียกใช้ทูลต่างๆ ของชุดพัฒนา JDK ได้จาก โดเรททอรี ไดร์ไดร์ที่มีอยู่บนระบบ โดยทำการแก้ไข ไฟล์ autoexec.bat โดยใช้เอดิเตอร์ใดก็ได้เปิดไฟล์นี้ขึ้นมา แล้วเข้าไปเพิ่มเติมพารามิเตอร์ชื่อ jdk1.4\bin โดยอาจทำการเพิ่ม

- Set CLASSPATH=[drive]:\[path]jdk1.4\bin; หรือ
- ;[drive]:\[path]jdk1.4\bin; ต่อท้ายด้วยอักขรชุดท้ายในคลาสพารามิเตอร์ที่มีอยู่เดิม

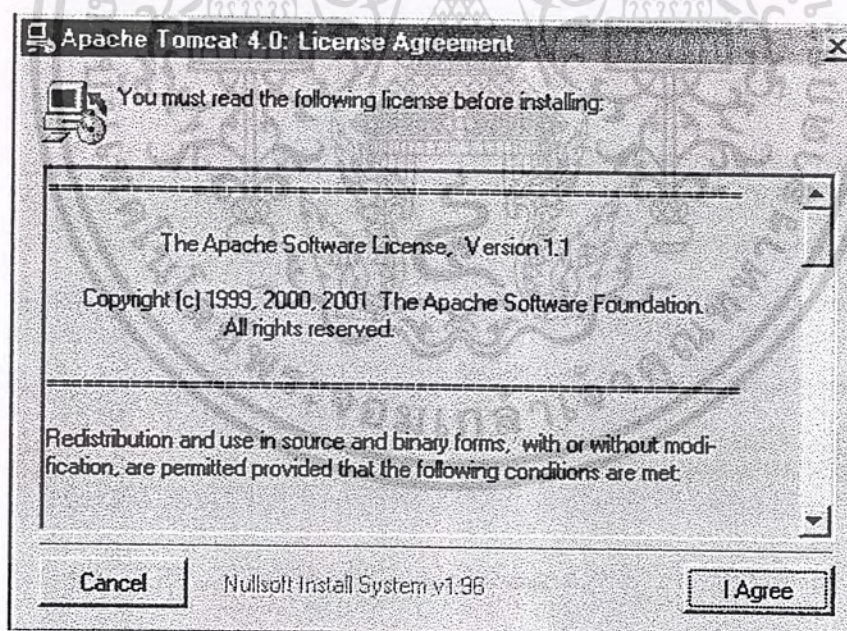
ข2.2 การติดตั้งเซิร์ฟเวอร์เลตออนอิน

ดาวน์โหลดไฟล์ติดตั้ง โดยเลือกใช้ Jakarta-Tomcat เวอร์ชัน 4.0.3 ชื่อ "jakarta-tomcat-4.0.3.exe" ซึ่งสามารถดาวน์โหลดได้จาก <http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.3/bin/jakarta-tomcat-4.0.3.exe> และสามารถดาวน์โหลดเวอร์ชันอื่นๆ ได้จาก <http://jakarta.apache.org/builds/>

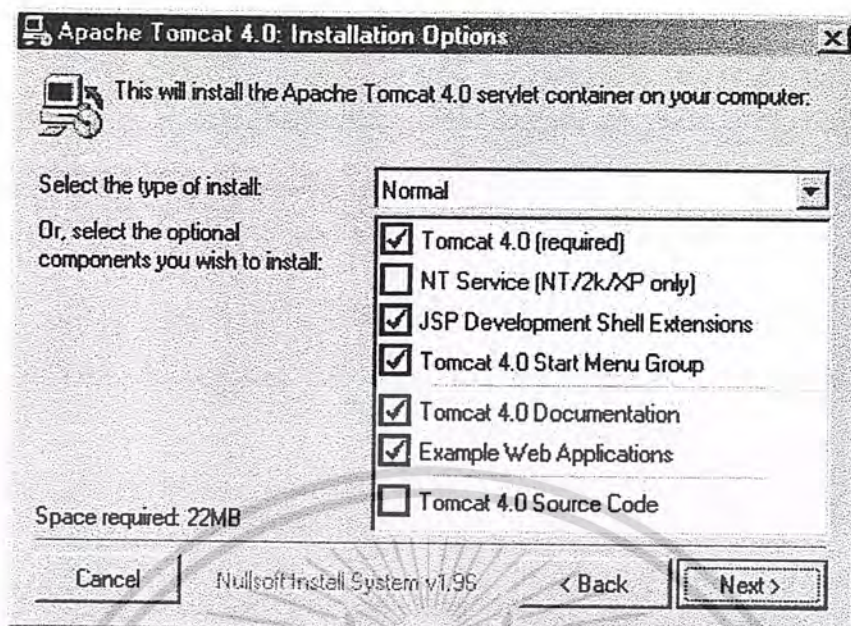
ไฟล์ jakarta-tomcat-4.0.3.exe ที่ดาวน์โหลดได้มาจะมีไฟล์ที่ใช้งานถูกแพ็คเกจเก็บรวมกันอยู่ ดังนั้นจึงต้องแตกไฟล์เหล่านั้นออกมา โดยรันไฟล์ที่ได้มา จากนั้นการขยายไฟล์จะถูกทำให้โดยอัตโนมัติ แต่ถ้าหากมีการติดตั้ง jakarta-tomcat เวอร์ชันเก่าไว้แล้วในเครื่อง ให้ลบก่อนทำการขยายไฟล์



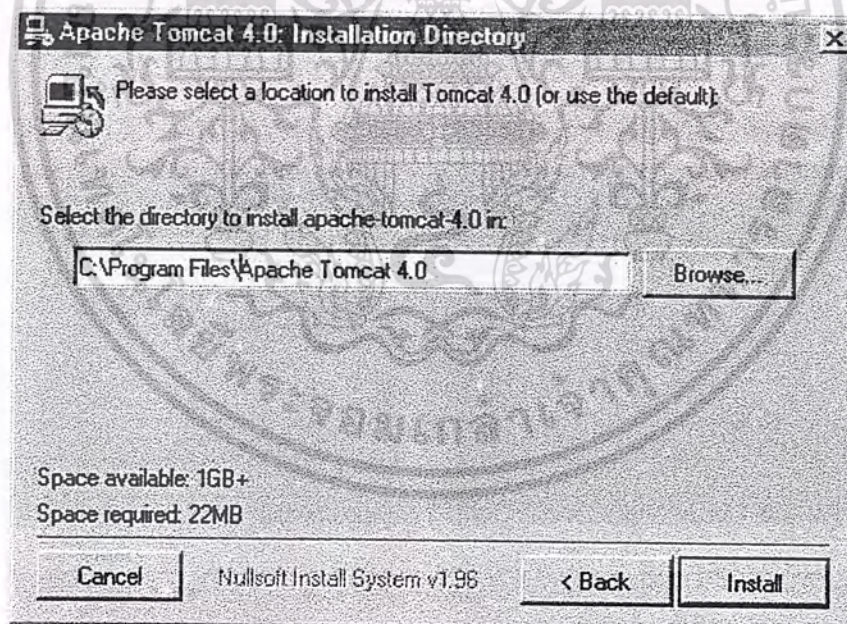
รูปที่ น.6 Tomcat จะหา ไดเรกทอรี ของ JDK เพื่อใช้ระบุพาร



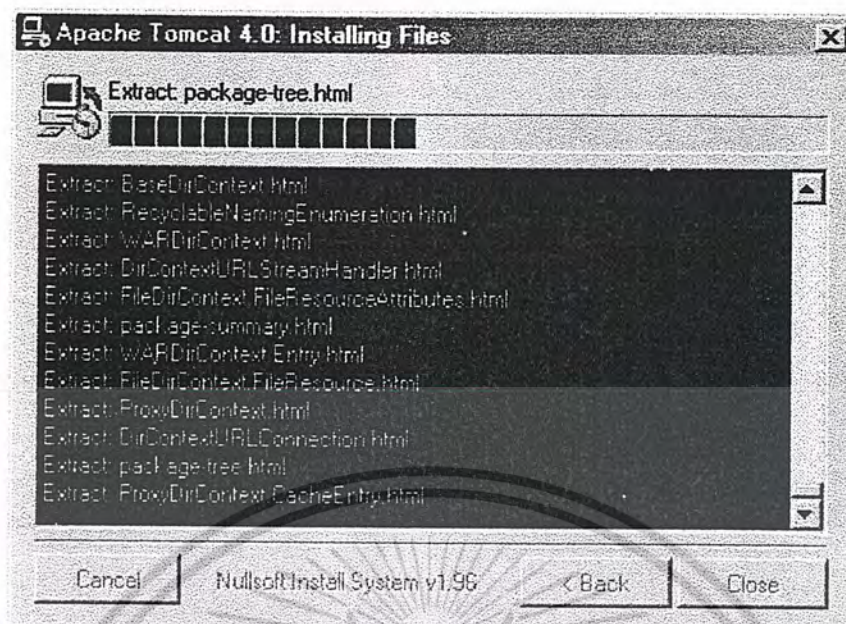
รูปที่ น.7 แสดงข้อตกลงทางกฎหมาย



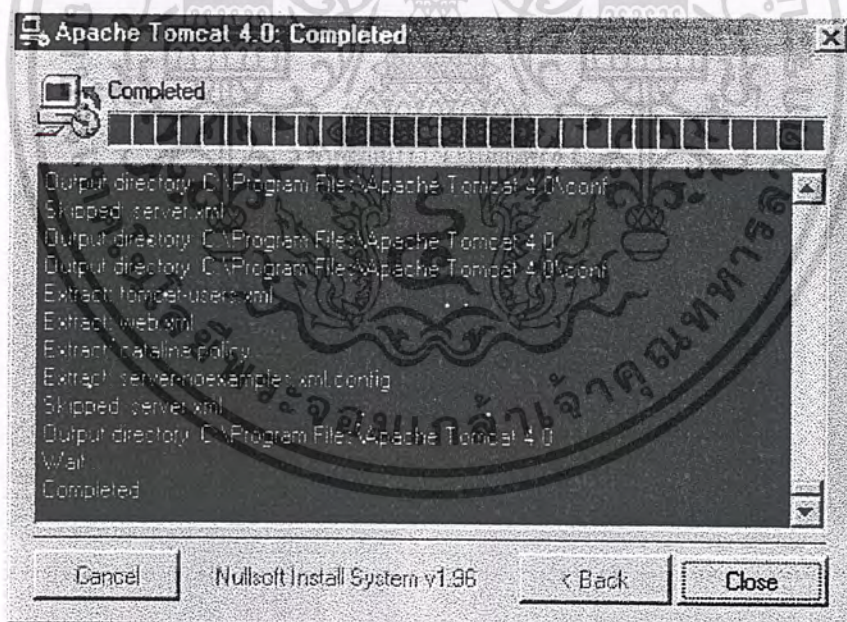
รูปที่ ๔.๘ แสดงขั้นตอนการเลือกคอมโพเนนต์ต่างๆ ในการติดตั้ง Tomcat



รูปที่ ๔.๙ แสดงขั้นตอนการระบุพาธในการติดตั้ง Tomcat

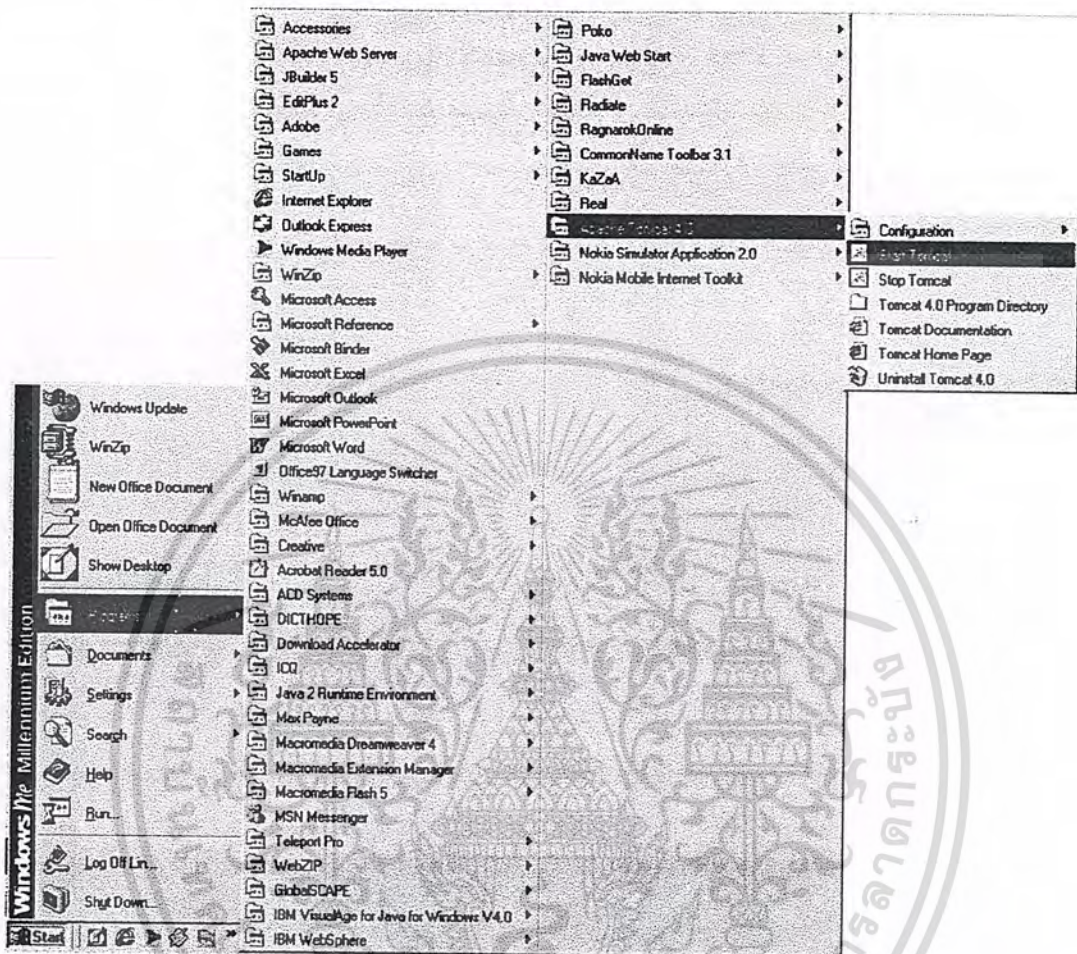


รูปที่ ข.10 แสดงกระบวนการขยายไฟล์ในระหว่างการทำการติดตั้ง Tomcat



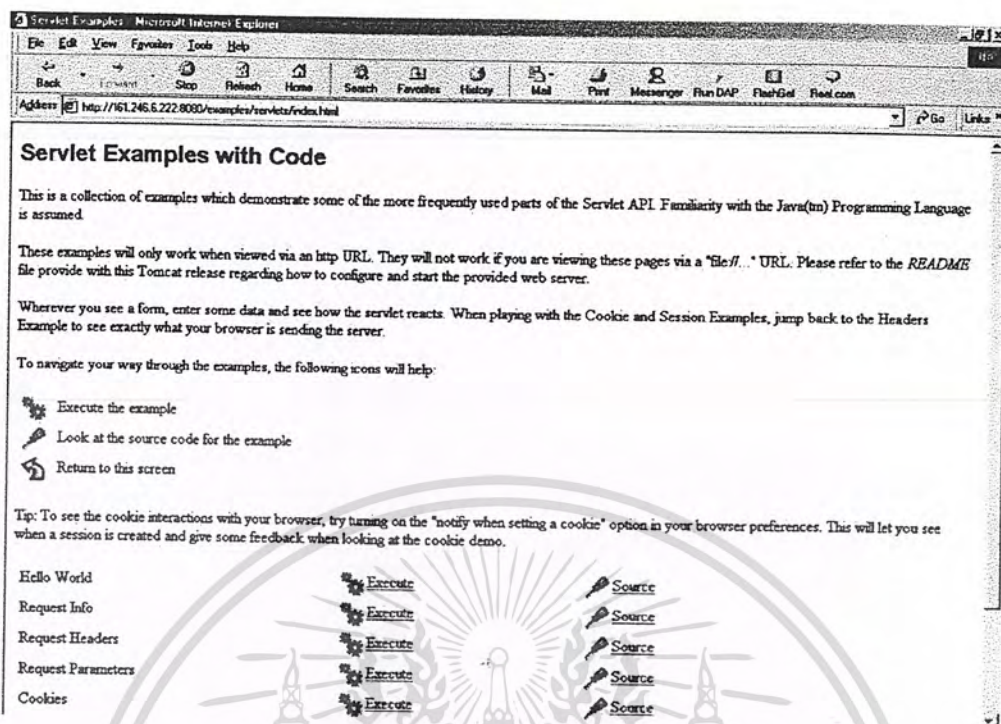
รูปที่ ข.11 แสดงการเสร็จสิ้นการติดตั้ง Tomcat

เมื่อทำการติดตั้งเรียบร้อยแล้ว ให้ start Tomcat ที่ Start -> Programs->Apache Tomcat 4.0 ->Start Tomcat



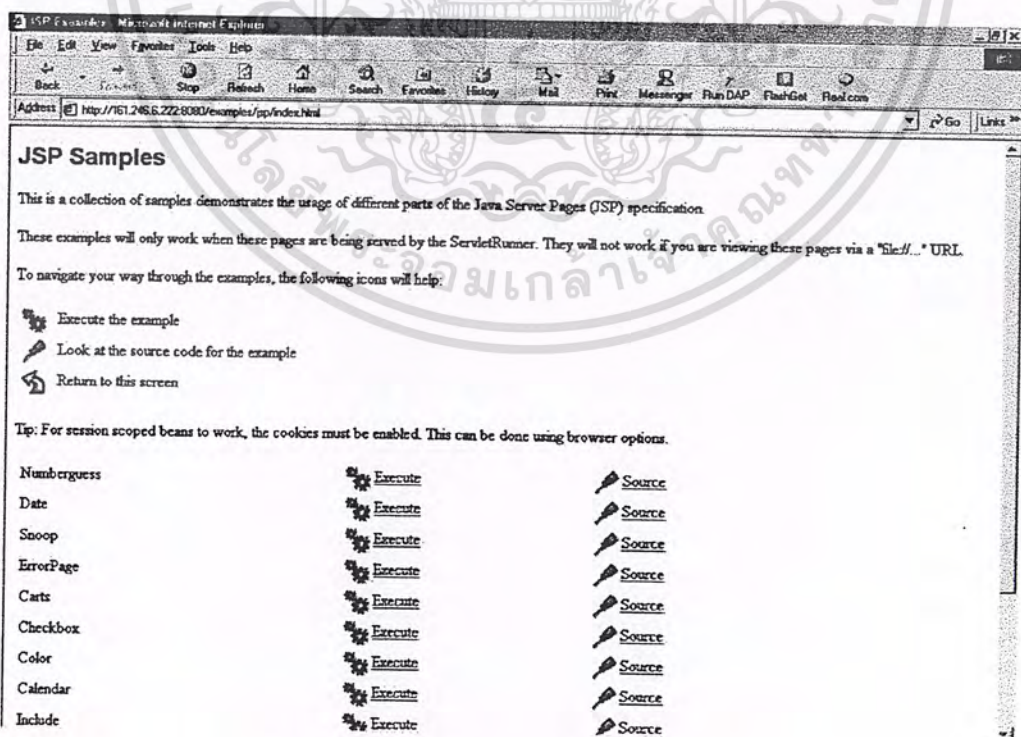
รูปที่ ข.12 การรัน Tomcat

หลังจากติดตั้งตามขั้นตอนแล้วทดสอบว่าการติดตั้ง Tomcat ประสบผลสำเร็จหรือไม่ด้วยการเปิด IE แล้วเข้ามาที่ URL <http://localhost:8080/examples/servlets/> เพื่อเป็นการทดสอบว่าสามารถใช้งานเซิร์ฟเลตบนเว็บเซิร์ฟเวอร์ได้หรือไม่ จะได้หน้าจอดังนี้



รูปที่ น.13 เว็บเพจแสดงการใช้งานเซิร์ฟเวอร์ Tomcat

และทดสอบการใช้งาน JSP ด้วยการเข้า URL `http://<localhost>:8080/examples/jsp/` จะได้หน้าจอ

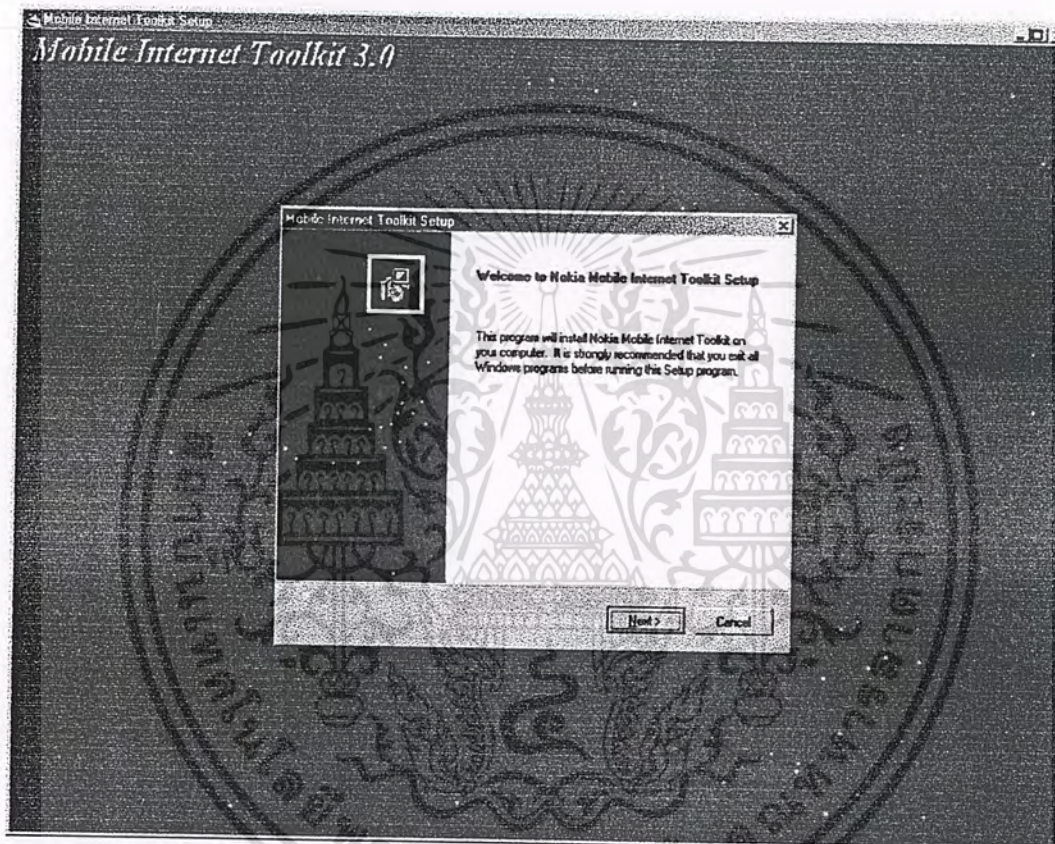


รูปที่ น.14 เว็บเพจแสดงการใช้งาน JSP Tomcat

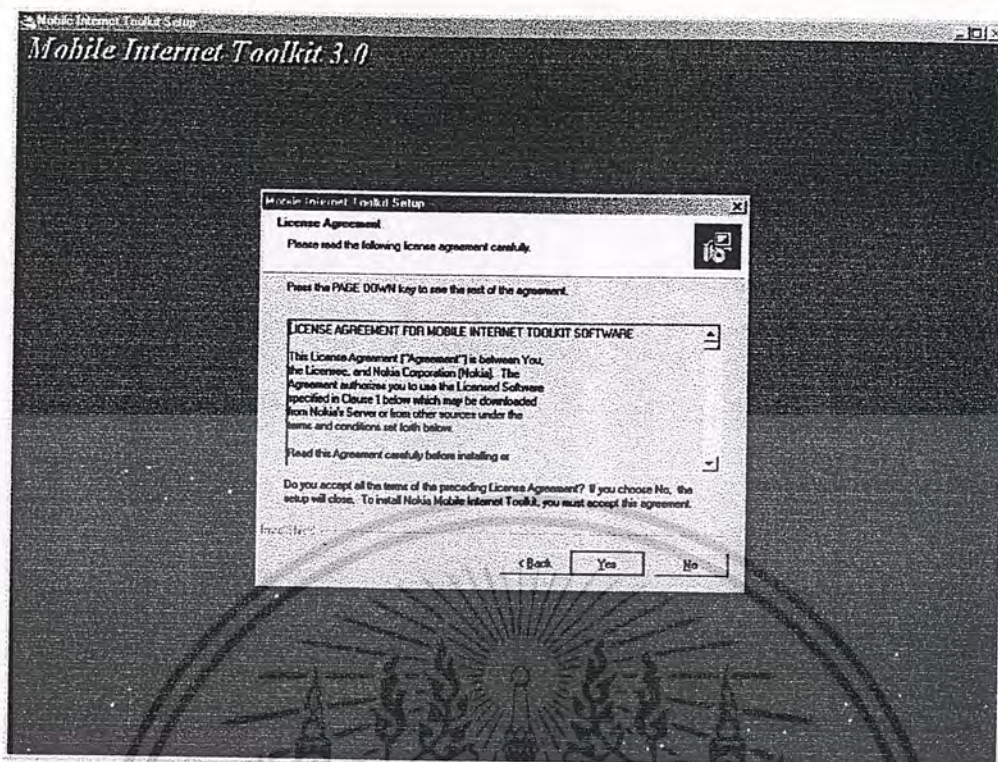
ข.2.3 การดาวน์โหลดและติดตั้ง Nokia Toolkit

ดาวน์โหลดไฟล์ติดตั้ง จากเว็บไซต์ของบริษัทโนเกีย (www.nokia.com) โดยต้องทำการลงทะเบียนก่อนถึงจะมีสิทธิในการดาวน์โหลดโปรแกรมฟรีได้ จากนั้นจึงค่อยทำการดาวน์โหลด Nokia WAP Toolkit เวอร์ชัน 3.0 ชื่อ NokiaToolkit3_0.zip

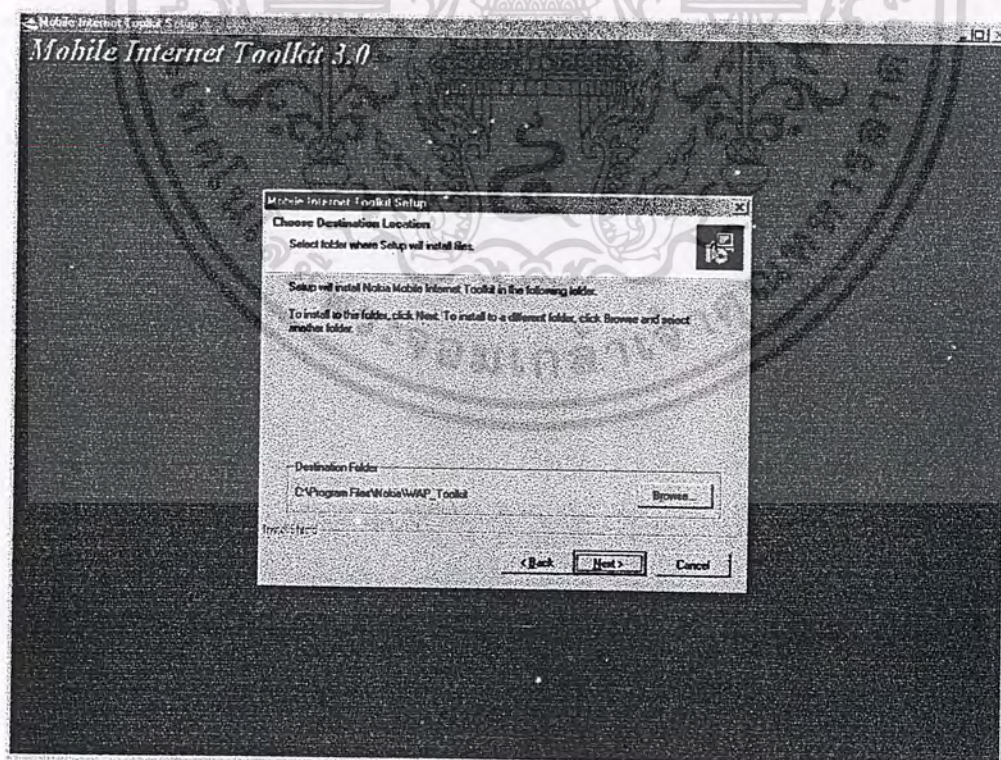
ทำการ unzip ไฟล์ที่ถูกบีบอัดของ Nokia Toolkit ก่อน จากนั้นดับเบิลคลิกที่ไอคอน setup.exe ที่ unzip ออกมา แล้วโปรแกรมจะเริ่มทำงาน



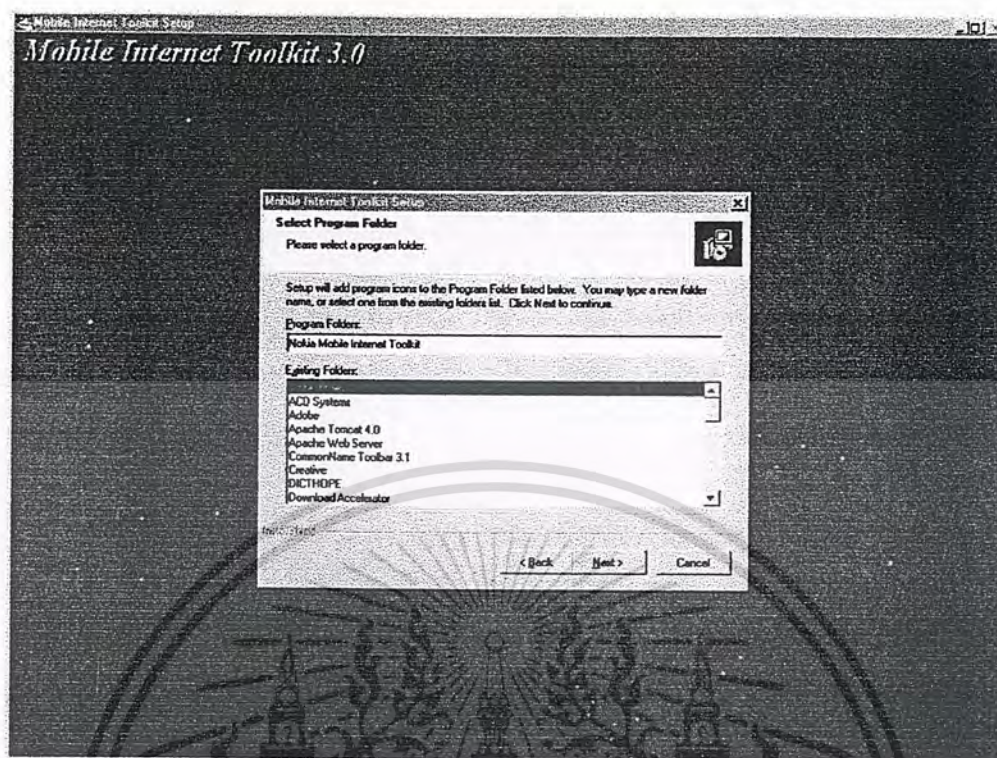
รูปที่ ข.15 แสดงหน้าต่างติดตั้ง Nokia Toolkit



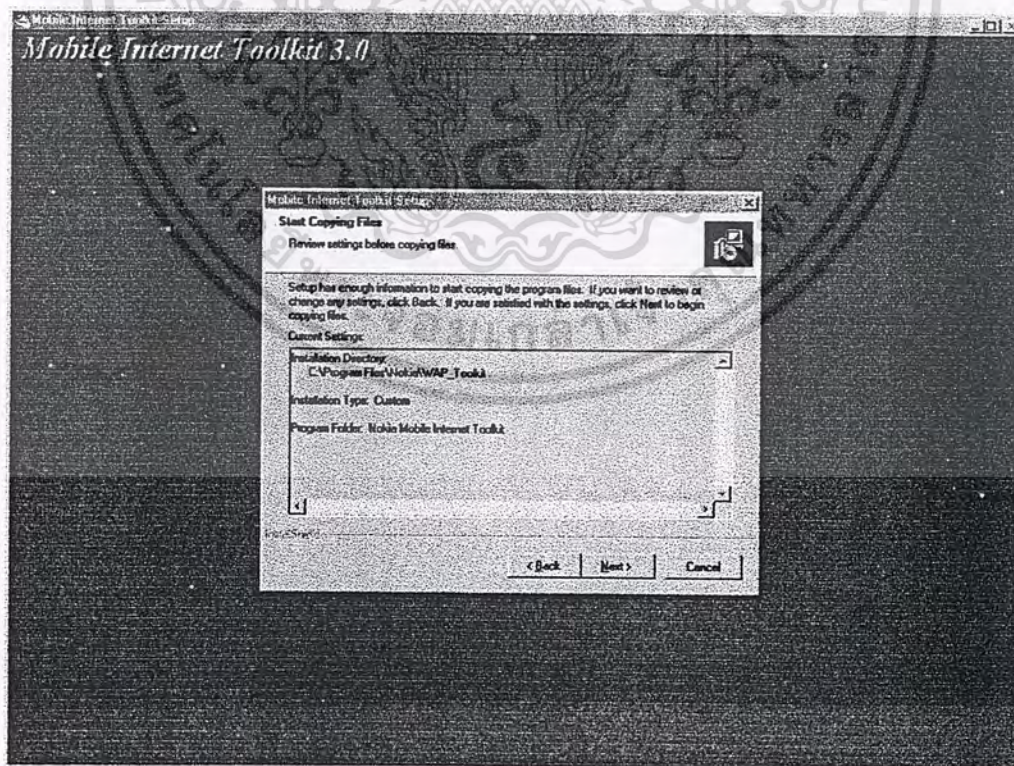
รูปที่ ข.16 แสดงข้อตกลงทางกฎหมาย



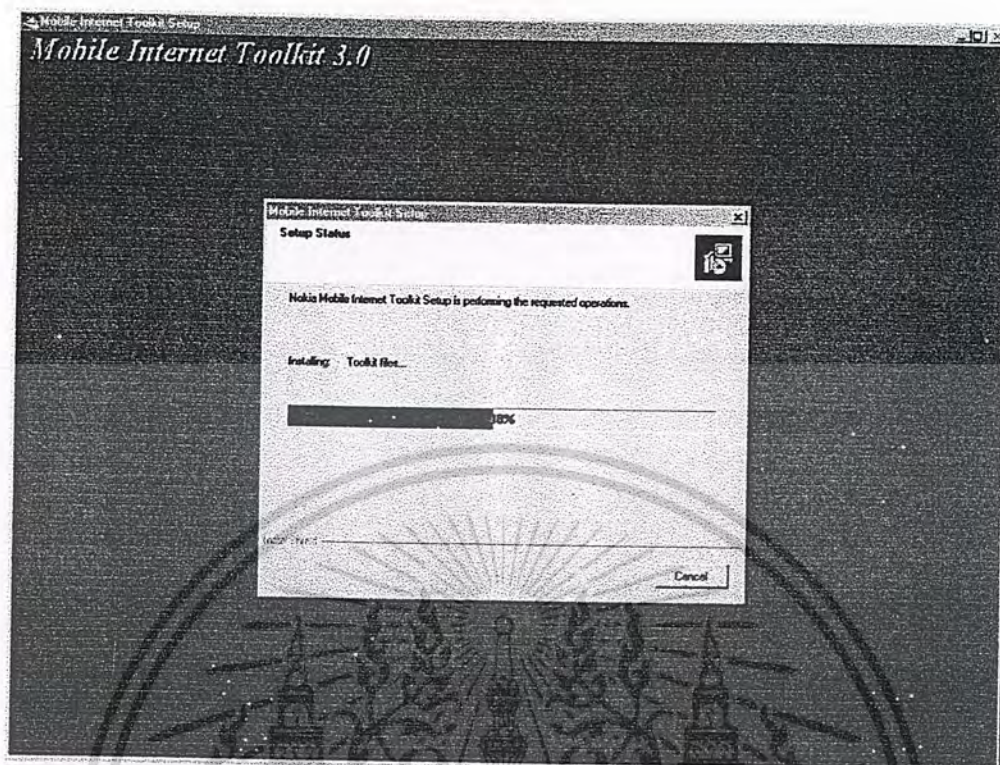
รูปที่ ข.17 แสดงขั้นตอนการระบุพาธในการติดตั้ง Nokia Toolkit



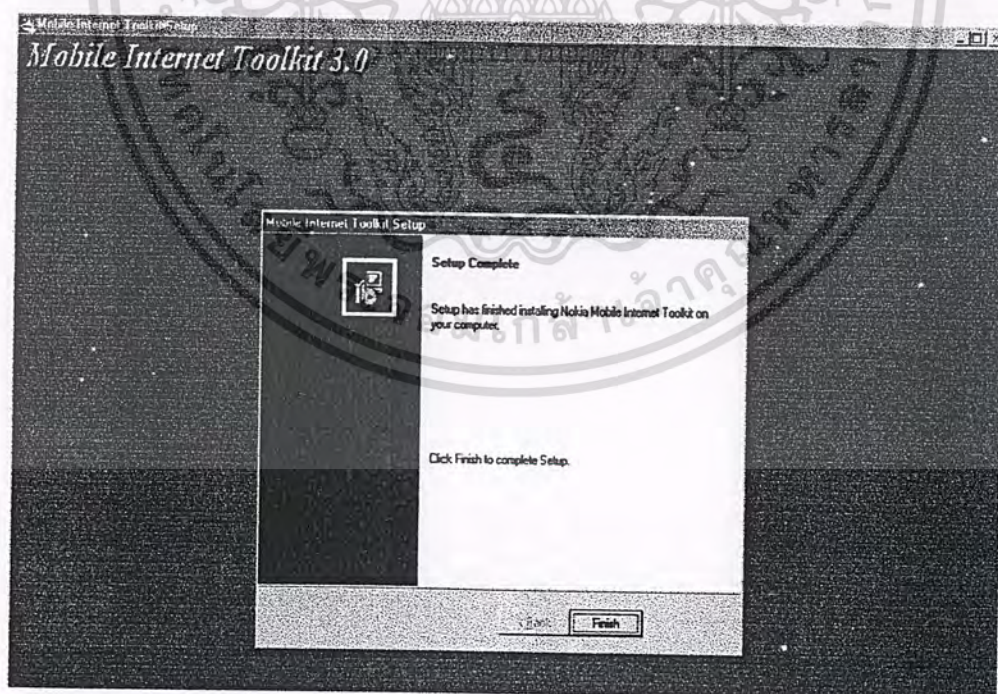
รูปที่ น.18 สร้างกลุ่มของไอคอนบนปุ่ม Start



รูปที่ น.19 แสดงภาพของการลง Nokia Toolkit



รูปที่ น.20 แสดงกระบวนการคัดลอกไฟล์ในระหว่างทำการติดตั้ง Nokia Toolkit



รูปที่ น.21 แสดงการเสร็จสิ้นการติดตั้ง Nokia Toolkit

ข.2.4 การใช้งานตัวแปลงภาษา

หลังจากเซตระบบทั้งเรียบร้อยแล้วก็จะสามารถใช้งานตัวแปลงภาษาได้ด้วยการนำไฟล์

ContactWap.class	Convert.class
Doc.class	Conv2wml.class
HtmlDoc.class	edit2XML.class
Conv2wml.class	Conv2xhtml.class

นำไฟล์ .class ทั้งหมดไว้ใน [drive]:[path]\jakarta-tomcat-4.0.3\webapps\webdav\WEB-INF\classes\ ซึ่งมีอยู่แล้ว และสุดท้ายนำไฟล์ convert.wml มาใส่ไว้ใน [drive]:[path]\jakarta-tomcat-4.0.3\webapps\webdav\

หลังจากรันเซิร์ฟเวอร์เสร็จแล้ว ให้เปิดบราวเซอร์ของ Nokia Wap Toolkit แล้วเข้ามาที่ URL <http://localhost:8080/webdav/convert.wml> จากนั้นจึงพิมพ์ URL ของเว็บไซต์ที่ต้องการ



รูปที่ ข.22 ตัวอย่าง แร็ปเพจของ [HT -> W] ML converter

ภาคผนวก ค

ดาวน์โหลดโปรแกรม Java Development Kit (JDK) 1.4

“<http://java.sun.com/j2se/1.4/download.html>” หรือ “<ftp://ftp.javasoft.com>”

ดาวน์โหลด Jakarta-Tomcat v.4.0.3

“<http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.3/bin/jakarta-tomcat-4.0.3.exe>”

ดาวน์โหลด Nokia Toolkit

<http://www.nokia.com>



บรรณานุกรม

- [1] Timothy T. Gottleber and Timothy N. Trainor : “*More excellent HTML with an introduction to JavaScript*”, McGraw-Hill, International Editions 2000.
- [2] Marty Hall : “*Core Servlets and JavaServer Pages*”, Sun Microsystems Press a Prentice Hall Title.
- [3] H.M. Deitel and P.J. Deitel : “*Java How to Program 3rd edition*”, Prentice-Hall.
- [4] ดร.วีระศักดิ์ ชิงदार : “*Java Programming Volumn 1*”, บริษัท ซีเอ็ดดูเคชั่น จำกัด(มหาชน)
- [5] ดร.วีระศักดิ์ ชิงदार : “*Java Programming Volumn 2*”, บริษัท ซีเอ็ดดูเคชั่น จำกัด(มหาชน)
- [6] สราวุธ อ้อยศรีสกุล : “*เปิดมิติ Mobile Internet ด้วย WAP*”, บริษัท วิศดี กรุป จำกัด
- [7] Steve Holzn : “*Inside XML 1st edition*”, New Rider Publishing
- [8] วรณฉัตร เทพสวัสดิ์ และ วรณิตา วีระมนตรี : “*ตัวแปลงภาษา HTML เป็น WML(HTML-TO-WML CONVERTER)*”, ปรินญานินพนธ์ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์, 2543
- [9] สุปราณี ชีร ไกรศรี : “*HTML4 Visual Guide*”, Provision, พิมพ์ครั้งที่6, ธันวาคม 2543
- [10] <http://www.w3school.org>
- [11] <http://www.wapforum.org>
- [12] <http://www.nokia.com>