เจเอสพี

สำหรับเวบโปรแกรมเมอร์

เจเอสพี สำหรับเวบโปรแกรมเมอร์
ISBN : 974-90480-0-8
จำนวน 201 หน้า
ราคา **180** บาท

ผู้เขียน
นรินทร์ โอฬารกิจอนันต์ (SJCP)

จัดทำโดย
สำนักพิมพ์ เดคิซูกิ ดอทเนต
ตู้ปณ 13 ปณจ. ภาษีเจริญ กรุงเทพมหานคร 10160
URL : http://www.dekisugi.net/java
email : webmaster@dekisugi.net

บรรณานุกรม

- Marty Hall, More Servlets™ and JavaServer Pages™/Marty Hall, Prentice Hall, ISBN 0-13-067614-4
- Barry Burd, JSP: JavaServer Pages™, M&T Books, ISBN 0-7645-3535-8
- James Goodwill, Pure JavaServer Pages™, Sams, ISBN 0-6723-1902-0
- JDK 1.4 documentation , http://java.sun.com

หนังสือเล่มนี้เป็นหนังสือเล่มที่สองในชุด หนังสือจาวา ต่อจาก
จาวา สำหรับผู้เริ่มต้น

โครงการหนังสือในอนาคต
สร้างโปรแกรมทางธุรกิจของคุณเองด้วย J2EE
สร้างโปรแกรมบนออร์แกนไนเซอร์ด้วย J2ME

สงวนลิขสิทธิ์ ตามพระราชบัญญัติลิขสิทธิ์ พุทธศักราช 2521 โดย นาย นรินทร์ โอฬารกิจอนันต์ ห้ามมิให้นำส่วนหนึ่งส่วนใดหรือทั้งหมดของหนังสือไปใช้เพื่อประโยชน์เชิงพาณิชย์โดยไม่ได้รับอนุญาตเป็นลายลักษณ์อักษรจากเจ้าของลิขสิทธิ์

เครื่องหมายการค้าทั้งหมดที่กล่าวถึงในหนังสือเล่มนี้เป็นขององค์กรหรือบริษัทที่กล่าวถึงทั้งหมด

สารบัญ

คำแนะนำในการอ่าน 4
บทที่ 1 รู้จัก เจเอสพี 7
บทที่ 2 เจเอสพีคอนเทนเนอร์ 13
บทที่ 3 บล็อกของคำสั่งเจเอสพี 23
บทที่ 4 ไดเร็กทีฟ 37
บทที่ 5 วัตถุแฝง 47
บทที่ 6 แท็กเจเอสพี 61
บทที่ 7 บีน 65
บทที่ 8 คุ๊กกี้ 75
บทที่ 9 JDBC 85
บทที่ 10 ป้ายโฆษณาอัตโนมัติ 99
บทที่ 11 สร้างแบบฟอร์มให้ส่งอีเมล์ 103
บทที่ 12 แบบสำรวจประชามติ 111
บทที่ 13 การเก็บสถิติผู้เข้าชมเวบไซต์ 115
บทที่ 14 สมุดลงนาม 125
บทที่ 15 กระดานเสวนา 131
บทที่ 16 ระบบสมาชิก 141
บทที่ 17 ร้านค้าออนไลน์ 149
บทที่ 18 ระบบรักษาความปลอดภัย SSL 161
บทที่ 19 สร้างแท็กของคุณเอง 169
ภาคผนวก ก. HTML 175
ภาคผนวก ข. SQL พื้นฐาน 187
ภาคผนวก ค. แพจเกจสำหรับการ Upload ไฟล์ (ของแถม) 197

คำแนะนำในการอ่าน

งานเกี่ยวกับการพัฒนาเวบไซต์ เป็นความใฝ่ฝันของใครหลายๆ คน เชื่อว่าคุณเองก็คงเคยสร้างโฮมเพจส่วนตัวเล่นๆ บ้างแล้ว หนังสือเล่มนี้จะทำให้คุณก้าวไปอีกขั้นหนึ่ง ด้วยการสร้างใช้ภาษาจาวาในการสร้างเวบไซต์ให้ทำอะไรต่อมิอะไรได้เหมือนเวบไซต์มืออาชีพ

หนังสือเล่มนี้สำหรับใคร

หนังสือเล่มนี้เขียนขึ้นสำหรับคนที่อยากเป็น **เวบโปรแกรมเมอร์** คำว่า เวบโปรแกรมเมอร์ ไม่เหมือนกับคำว่า เวบมาสเตอร์ เพราะเวบโปรแกรมเมอร์คือผู้ที่สร้างเวบไซต์ด้วยการเขียนโปรแกรมให้เวบเซิร์ฟเวอร์ทำงานอย่างที่ต้องการได้ ในขณะที่เวบมาสเตอร์คือผู้ที่คอยดูแลความเรียบร้อยของเวบไซต์ หรือพูดอีกนัยหนึ่งก็คือ เวบโปรแกรมเมอร์เป็นผู้สร้าง ส่วนเวบมาสเตอร์เป็นผู้รักษา

เวบโปรแกรมเมอร์ ไม่ได้สร้างเวบไซต์เองทั้งหมด การสร้างโฮมเพจระดับมืออาชีพต้องอาศัยความร่วมมือระหว่าง เวบโปรแกรมเมอร์ กับ เวบดีไซน์เนอร์ คำว่า **เวบดีไซน์เนอร์** หมายถึงคนที่ออกแบบเวบไซต์ให้น่าดึงดูด ใช้งานง่าย เวบดีไซน์เนอร์ต้องมีความถนัดทางศิลปะ มีประสบการณ์ทางด้านสื่อสิ่งพิมพ์ ใช้โปรแกรมสร้างภาพกราฟฟิคได้อย่างคล่องแคล่ว ชอบเครื่องแมคอินทอช (อันหลังนี้ไม่แน่เสมอไป) ในขณะที่เวบโปรแกรมเมอร์จะเป็นพวกที่ทำงานอยู่เบื้องหลัง เพราะส่วนของเวบไซต์ที่สร้างสรรค์โดยเวบโปรแกรมเมอร์จะซ่อนอยู่บนเวบเซิร์ฟเวอร์ ผู้เยี่ยมชมเวบไซต์แทบมองไม่ออกเลยว่าส่วนไหนของโฮมเพจที่เราเห็นเป็นผลงานของเวบโปรแกรมเมอร์ คนที่เป็นเวบโปรแกรมเมอร์ต้องมีพื้นฐานมาทางสายเทคโนโลยี โดยเฉพาะอย่างยิ่งต้องมีทักษะในการเขียนโปรแกรมคอมพิวเตอร์ เนื้อหาทั้งหมดของหนังสือเล่มนี้จะเกี่ยวกับคนที่อยากเป็นเวบโปรแกรมเมอร์เท่านั้นไม่เกี่ยวกับเวบดีไซน์เนอร์เลยสักนิดเดียว อย่าบ่นถ้าโฮมเพจตัวอย่างในหนังสือเล่มนี้มีหน้าตาที่ดูธรรมดาๆ

ต้องมีพื้นฐานอะไรบ้าง

คนที่จะอ่านหนังสือเล่มนี้ได้รู้เรื่องต้องมีพื้นฐานการเขียนโปรแกรมภาษาจาวาพอสมควร คุ้นเคยกับคำสั่งพื้นฐานและเข้าใจโครงสร้างของภาษา ถ้าคุณไม่มีพื้นฐานภาษาจาวาอยู่เลย ขอแนะนำให้อ่านหนังสือ ***จาวา สำหรับผู้เริ่มต้น*** ซึ่งดาวน์โหลดได้จาก http://www.dekisugi.net/java

นอกจากนี้ คุณต้องเคยสร้างโฮมเพจด้วยการเขียนคำสั่ง HTML มาบ้าง และพอมีความรู้เกี่ยวกับการสั่งงานฐานข้อมูลด้วยคำสั่ง SQL ให้คุณอ่านภาคผนวก ก และ ข ของหนังสือเล่มนี้ก่อนเป็นอันดับแรก ถ้าคุณไม่แน่ใจว่าคุณรู้จัก HTML และ SQL

คำแนะนำเวลาอ่าน

การศึกษาการเขียนโปรแกรมคอมพิวเตอร์ ไม่ว่าจะเป็น เจเอสพี หรือภาษาอะไรก็ตาม ต้องได้ลองเขียนโปรแกรมตัวอย่างและทดสอบโปรแกรมเหล่านั้นบนเครื่องคอมพิวเตอร์จึงจะได้ผล ดังนั้นคุณต้องมีเครื่องคอมพิวเตอร์ส่วนบุคคลที่ใช้ระบบปฏิบัติการวินโดว์ และมีไมโครซอฟท์อินเตอร์เนตเอ็กซ์พลอเลอร์เวอร์ชั่น 4 ขึ้นไปติดตั้งอยู่บนเครื่อง นอกจากนี้คอมพิวเตอร์ของคุณควรติดต่อกับอินเตอร์เนตได้ด้วย

ขณะที่อ่านหนังสือเล่มนี้ ควรทดลองเขียนโปรแกรมตัวอย่างและทดสอบโปรแกรมตัวอย่างไปด้วย เพื่อดูว่าใช้งานได้จริงหรือไม่ทุกโปรแกรม จุดประสงค์ไม่ใช่เพื่อการจับผิด แต่การได้ลงมือจะทำให้เกิดความชำนาญมากกว่าการอ่านเฉยๆ เป็นอย่างมาก ภาษาคอมพิวเตอร์เป็นเรื่องที่เรียนไม่ได้ด้วยการอ่านแต่เพียงอย่างเดียว จะให้ดีขอแนะนำให้พิมพ์โปรแกรมตัวอย่างลงในเครื่องด้วยตัวเองทีละบรรทัดทุกโปรแกรม

อนึ่ง หากเนื้อหาในหนังสือเล่มนี้มีส่วนผิด ส่วนเพิ่มเติม หรือเนื้อหาใหม่ๆ ผู้เขียนจะนำไปรวมไว้ใน http://www.dekisugi.net/java/support ลองหาโอกาสเข้าไปเยี่ยมชมเพื่อให้ตัวคุณได้รับความรู้ที่ทันสมัยอยู่เสมอ

-ผู้เขียน

"คนเรามีวิธีดำรงชีวิตอยู่แค่สองอย่าง อย่างหนึ่งคือราวกับว่าไม่มีอะไรในโลกนี้ที่น่าอัศจรรย์ อีกอย่างคือราวกับว่าทุกสิ่งทุกอย่างในโลกนี้ช่างน่าอัศจรรย์"

-อัลเบิร์ต ไอน์สไตน์

1

รู้จัก เจเอสพี

จาวาเซิร์ฟเวอร์ เพจ™ หรือเรียกย่อๆ ว่า **เจเอสพี** คือ การใช้ภาษาจาวาในการสร้างเวบเพจแบบที่มีเนื้อหาไม่ตายตัว

ภาษา HTML

ภาษา HTML ใช้สร้างเวบเพจแบบที่มีเนื้อหาตายตัว ตัวอย่างเช่น

โปรแกรมที่ 1-1 `helloworld.html`

```
<html>
<body>
<b>Hello World</b>
</body>
</html>
```

ไฟล์ helloworld.html เก็บคำสั่ง HTML ที่บอกให้เบราเซอร์ (เช่น ไมโครซอฟท์อินเตอร์เนตเอ็กซ์พลอเรอร์ หรือเนตสเคปคอมมิวนิเคเตอร์) แสดงเวบเพจที่มีคำว่า Hello World เขียนด้วยอักษรตัวหนา

ไฟล์ HTML ประกอบด้วยข้อความที่เป็นเนื้อหาของเวบไซต์ (เช่น คำว่า Hello World) กับคำสั่งกำกับการแสดงผลของเบราเซอร์ (เช่น คำว่า ... ที่บอกให้แสดงข้อความด้วยอักษรตัวหนา) เวบเพจที่เขียนด้วยคำสั่ง HTML ล้วนๆ มีลักษณะที่ตายตัว กล่าวคือเนื้อหาของเวบเพจมีลักษณะแน่นอน จะเยี่ยมชมกี่ครั้งก็ดูเหมือนเดิม

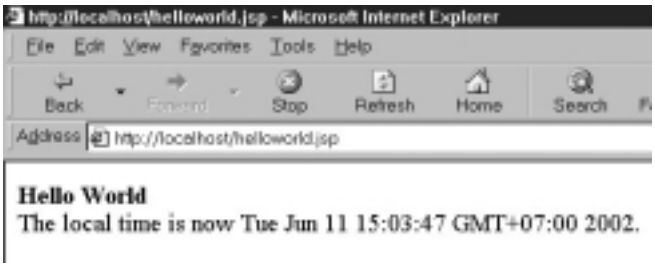
เจเอสพี

เวบเพจที่เขียนด้วย เจเอสพี มีเนื้อหาที่ไม่ตายตัว โดยคำสั่งเจเอสพีจะอยู่ปะปนกับคำสั่ง HTML เพื่อสร้างเนื้อหาเฉพาะส่วนที่มีเนื้อหาที่เปลี่ยนแปลงได้ ลองดูตัวอย่างการใช้คำสั่งเจเอสพีสร้างโฮมเพจที่สามารถแสดงวันเวลาปัจจุบันซึ่งเปลี่ยนไปเรื่อยๆ ทุกครั้งที่มีผู้เยี่ยมชมดังต่อไปนี้

โปรแกรมที่ 1-2 `helloworld.jsp`

```
<html>
<body>
<b>Hello World</b><br>
The local time is now <%= new java.util.Date() %>.
</body>
</html>
```

คำสั่ง เจเอสพี ในไฟล์ helloworld.jsp คือ คำสั่ง `<%= new java.util.Date() %>` ซึ่งเป็นคำสั่งที่บอกให้แสดงวันที่และเวลา เนื้อหาส่วนอื่นๆ จะเหมือนกันทุกครั้งที่มีผู้เยี่ยมชมเวบเพจ แต่ตรงตำแหน่งของคำสั่งเจเอสพี ข้อความจะเปลี่ยนไปเรื่อยๆ เมื่อมีผู้เยี่ยมชมเวบไซต์ เพราะมันจะแสดงวันที่และเวลาปัจจุบัน ณ ขณะที่เยี่ยมชมเวบเพจนั้น ดังภาพ



โฮมเพจนี้มีเนื้อหาต่างกันเมื่อถูกเข้าถึงต่างเวลา นั้นคือเราสามารถใช้ เจเอสพี ในการสร้างเนื้อหาส่วนที่ไม่ตายตัวในเวบไซต์ได้ โดยคำสั่งเจเอสพีสามารถอยู่ปะปนกับคำสั่ง HTML โดยบริเวณที่เป็นคำสั่งเจเอสพีจะล้อมรอบด้วยเครื่องหมาย <% และ %> เสมอ

เวบเพจที่มีคำสั่งเจเอสพีอยู่ จะต้องมีนามสกุล .jsp แทนที่จะเป็น .html ธรรมดา ซึ่งเวบเซิร์ฟเวอร์จะดูจากนามสกุลเป็นตัวบอกว่าเวบเพจหน้าใดที่คำสั่งเจเอสพีปนอยู่บ้าง ถ้ามีคำสั่งเจเอสพี เวบเซิร์ฟเวอร์จะแทนคำสั่งเจเอสพีในเพจหน้านั้นด้วยข้อความที่เหมาะสมก่อนที่จะส่งไปให้เบราเซอร์ที่เครื่องของผู้เยี่ยมชมแสดงผล ดังตัวอย่างข้างต้น ทุกครั้งที่มีคนเรียกเพจ helloworld.jsp เวบเซิร์ฟเวอร์จะแทนคำสั่ง <%= new java.util.Date() %> ด้วยวันที่และเวลาปัจจุบันก่อน แล้วจึงส่งไปให้เบราเซอร์แสดงผล แต่ถ้าเวบเพจหน้านั้นมีนามสกุลเป็น .html มันจะส่งไปให้เบราเซอร์ทันที เพราะเนื้อหาของไฟล์ .html นั้นตายตัวอยู่แล้ว ไม่ต้องเปลี่ยนแปลงอะไร

อนึ่ง การแสดงวันที่และเวลาเป็นเพียงตัวอย่างหนึ่งที่ทำให้เห็นภาพของประโยชน์ของการใช้คำสั่งเจเอสพี คำสั่งเจเอสพียังทำอะไรต่อมิอะไรได้อีกมาก

ทำไมต้องใช้เจเอสพี

อันที่จริงมีเทคโนโลยีอื่นๆ อีกหลายตัวที่ใช้สร้างโฮมเพจที่มีเนื้อหาแบบไม่ตายตัวได้เหมือนกับเจเอสพี เช่น CGI, ASP, ColdFusion หรือ PHP เราเรียกเทคโนโลยีเหล่านี้ว่า **เซิร์ฟเวอร์ไซด์สคริปต์** (Server-side scripts) เพราะมีลักษณะคล้ายๆ กับภาษาคอมพิวเตอร์ ซึ่งสั่งให้เวบเซิร์ฟเวอร์สร้างเวบเพจแบบไม่ตายตัวให้ แต่เจเอสพีมีข้อดีที่เหนือกว่าเซิร์ฟเวอร์สคริปต์ตัวอื่นๆ อยู่หลายประการ

NOTE

มีเทคโนโลยีอีกกลุ่มหนึ่งที่เรียกว่า ไคล์เอนท์ไซด์สคริปต์ (Client-side scripts) เช่น จาวาสคริปต์ หรือ VBScript เทคโนโลยีในกลุ่มนี้ใช้สร้างเวบที่มีเนื้อหาแบบไม่ตายตัวได้เหมือนกัน แต่แทนที่จะสั่งงานบนฝั่งเวบเซิร์ฟเวอร์ คำสั่งเหล่านี้จะถูกโหลดลงมาบนเบราเซอร์ก่อนแล้วทำงานบนฝั่งเบราเซอร์ทันทีก่อนการแสดงผล อย่างไรก็ดี มีการทำงานหลายอย่างที่ไคล์เอนท์ไซด์สคริปต์ทำไม่ได้

ประการแรก เจเอสพี ไม่ใช่ภาษาคอมพิวเตอร์ที่สร้างขึ้นมาใหม่ คำสั่งทุกคำสั่งของเจเอสพีคือคำสั่งภาษาจาวา ดังนั้นผู้ที่รู้ภาษาจาวาอยู่แล้วจึงเรียนรู้เจเอสพีได้อย่างรวดเร็ว อีกทั้งภาษาจาวายังเป็นภาษาเชิงวัตถุที่มีความสมบูรณ์แบบ มีโครงสร้างภาษาที่รัดกุม และมีความปลอดภัยสูง ไม่ยึดติดกับระบบปฏิบัติการ ดังนั้นเวบไซต์ที่เขียนด้วยเจเอสพีจึงได้รับอานิสงส์เหล่านั้นไปด้วย

ประการที่สอง เจเอสพี มีสถาปัตยกรรมที่เหมาะกับการใช้งานของผู้เยี่ยมชมจำนวนมากๆ ในเวลาเดียวกัน เวบเพจที่เขียนด้วยเจเอสพีทำงานเป็นมัลติเทรดโดยอัตโนมัติ (เวบโปรแกรมเมอร์ไม่ต้องสร้างเทรดเอง) อีกทั้งเวลาเวบเซิร์ฟเวอร์โหลดคำสั่งเจเอสพีเข้าไปในหน่วยความจำเพื่อทำงาน คำสั่งเจเอสพีจะยังคงค้างอยู่ในหน่วยความจำ เมื่อผู้เยี่ยมชมรายอื่นเรียกเวบเพจหน้านั้นซ้ำอีกจึงไม่เสียเวลาในการโหลดคำสั่งซ้ำอีก เวบเพจที่ใช้คำสั่งเจเอสพีจึงมีความเร็วในการทำงานเป็นอย่างมาก

ประการสุดท้าย เจเอสพี พัฒนาง่าย เพราะสามารถเขียนอยู่ปะปนกับคำสั่ง HTML ในไฟล์เดียวกัน การพัฒนาเวบไซต์จึงมีความสะดวก รวดเร็ว เข้าใจง่าย

ความเป็นมาของเจเอสพี

ก่อนหน้าที่จะมีเจเอสพี การใช้ภาษาจาวาทำงานเป็นเซิร์ฟเวอร์ไซด์สคริปต์ ใช้เทคโนโลยีที่มีชื่อว่า **เซิร์ฟเลต** ซึ่งเป็นการเขียนโปรแกรมภาษาจาวาล้วนๆ เพื่อสั่งให้เวบเซิร์ฟเวอร์สร้างเวบเพจที่มีเนื้อหาแบบไม่ตายตัว การสร้างเซิร์ฟเลตมีขั้นตอนเหมือนกับการเขียนโปรแกรมภาษาจาวาเต็มรูปแบบซึ่งค่อนข้างยุ่งยากเมื่อนำมาใช้พัฒนาเวบไซต์ ดังนั้นผู้พัฒนาจาวาจึงสร้าง เจเอสพี ขึ้นมาทดแทน โดยออกแบบให้เหมาะกับการนำไปใช้สร้างเวบเพจ

ที่จริงแล้ว เจเอสพี กับเซิร์ฟเลต เป็นเรื่องเดียวกัน เพราะไฟล์คำสั่งเจเอสพีที่เราเขียนขึ้นจะถูกแปลงให้เป็นเซิร์ฟเลตในที่สุด เพียงแต่ขั้นตอนการแปลงจะถูกซ่อนเอาไว้โดยเวบเซิร์ฟเวอร์ หน้าที่ของเวบโปรแกรมเมอร์ก็เพียงแต่เขียนไฟล์คำสั่งเจเอสพีขึ้นมาแบบการเขียนโฮมเพจปกติ แล้วนำไปวางไว้ในตำแหน่งที่ถูกต้อง ที่เหลือเวบเซิร์ฟเวอร์จะจัดการต่อเองโดยอัตโนมัติ เป็นการซ่อนความยุ่งยากทั้งหมดเอาไว้แบบที่เรามองไม่เห็นเลย

ทั้งเซิร์ฟเลต และเจเอสพี เป็นส่วนหนึ่งของ J2EE ซึ่งเป็นเทคโนโลยีของการสร้างโปรแกรมสำหรับเครื่องแม่ข่ายบนเน็ตเวิร์กที่สนับสนุนการทำงานแบบ web services การเรียนรู้เจเอสพีนอกจากจะนำไปใช้สร้างเวบไซต์อย่างเดียวแล้ว ยังเป็นพื้นฐานในการเรียนรู้ J2EE อีกด้วย เป็นการยิงปืนทีเดียวได้นกหลายตัว เพราะ web services กำลังกลายเป็นมาตรฐานของการพัฒนาโปรแกรมบนโลกไอทีในอนาคตที่โปรแกรมเมอร์ทุกคนต้องรู้

NOTE

ยังมีเทคโนโลยีจาวาที่เกี่ยวกับการสร้างเวบไซต์อีกตัวหนึ่ง เรียกว่า จาวาแอพเพลต จาวาแอพเพลตเป็นโปรแกรมภาษาขนาดจิ๋วซึ่งถูกโหลดลงมารันบนตัวเบราเซอร์ คล้ายๆกับ จาวาสคริปต์ แต่เบราเซอร์ที่จะรันจาวาแอพเพลตได้ต้องมีการติดตั้ง จาวาปลักอิน ก่อน เราจะไม่มีการกล่าวถึงจาวาแอพเพลตในหนังสือเล่มนี้

2

เจเอสพี คอนเทนเนอร์

เวบเซิร์ฟเวอร์ที่จะเข้าใจคำสั่งเจเอสพีต้องเป็นเวบเซิร์ฟเวอร์ที่สนับสนุนเทคโนโลยีจาวา สิ่งที่ทำให้เวบเซิร์ฟเวอร์ที่สนับสนุนเทคโนโลยีจาวาแตกต่างจากเวบเซิร์ฟเวอร์ทั่วไปก็คือ เวบเซิร์ฟเวอร์ที่สนับสนุนเจเอสพีมีสิ่งที่เรียกว่า **เจเอสพี คอนเทนเนอร์** ในบทนี้เราจะมารู้จักกับ เจเอสพี คอนเทนเนอร์ และทดลองติดตั้งเวบเซิร์ฟเวอร์ที่สนับสนุนเจเอสพี ได้แก่ Tomcat

เจเอสพี คอนเทนเนอร์

เจเอสพี คอนเทนเนอร์ คือจาวาเวอร์ชัวน์แมทชีนบนเวบเซิร์ฟเวอร์ หน้าที่ของจาวาเวอร์ชัวน์แมทชีนคือการรันคำสั่งภาษาจาวา เจเอสพี คอนเทนเนอร์ จะทำหน้าที่รันคำสั่งเจเอสพี ซึ่งก็คือคำสั่งในภาษาจาวาที่อยู่ในไฟล์ .jsp ก่อนที่จะส่งผลลัพธ์ไปยังเบราเซอร์

เมื่อมีผู้เยี่ยมชมร้องขอโฮมเพจที่มีนามสกุล .jspบนเวบไซต์ผ่านทางเบราเซอร์ เบราเซอร์จะส่งคำสั่งร้องขอไปยังเวบเซิร์ฟเวอร์ เมื่อเวบเซิร์ฟเวอร์พบว่าไฟล์ที่ร้องขอมีนามสกุลเป็น .jsp มันจะรู้ทันทีว่ามีคำสั่งเจเอสพีอยู่ในไฟล์ซึ่งต้องรันก่อน เวบเซิร์ฟเวอร์จะอ่านไฟล์ .jsp แล้วคอมไพล์ให้กลายเป็นโปรแกรมภาษาจาวาล้วนๆ ในรูปของไฟล์นามสกุล .java (ซึ่งก็คือเซิร์ฟเลต) จากนั้นมันจะเรียกคอมไพล์เลอร์ที่อยู่บนเครื่องเดียวกันออกมาคอมไพล์ให้ได้ไฟล์นามสกุล .class ซึ่งก็คือจาวาไบต์โค้ดที่พร้อมจะรัน จากนั้นเวบเซิร์ฟเวอร์จะรันจาวาไบต์

โค้ดที่ได้บน เจเอสพี คอนเทนเนอร์ ผลลัพธ์ที่ได้จากการรันจะอยู่ในรูปของคำสั่ง HTML ล้วนๆ ซึ่งจะส่งกลับไปให้เบราเซอร์ที่ร้องขอมาเพื่อแสดงผลบนหน้าจอของผู้เยี่ยมชม

ในการร้องขอครั้งต่อไป ไม่ว่าจะเกิดจากผู้เยี่ยมชมคนเดิมหรือไม่ก็ตาม เวบเซิร์ฟเวอร์จะประหยัดเวลาด้วยการลดขั้นตอนลง กล่าวคือ มันจะมองหาจาวาไบต์โค้ดของไฟล์ๆ เดิมที่ค้างอยู่ในหน่วยความจำบน เจเอสพี คอนเทนเนอร์ แล้วรันซ้ำเพื่อให้ได้ผลลัพธ์ส่งไปให้เบราเซอร์ได้ทันที ไม่มีการแปลงไฟล์จาก .jsp ให้เป็น .java และไม่มีการคอมไพล์ใหม่ ดังนั้นการเข้าถึงไฟล์ .jsp ที่เคยเข้าถึงไปแล้วครั้งหนึ่งจะมีความเร็วในการเข้าถึงเร็วกว่าในครั้งแรกเป็นอย่างมาก เพราะเวบเซิร์ฟเวอร์ไม่ต้องเสียเวลาคอมไพล์อีก ขั้นตอนทั้งหลายที่กล่าวมานี้เกิดขึ้นโดยอัตโนมัติ

อย่างไรก็ดี เมื่อใดก็ตามที่มีการแก้ไขเนื้อหาในไฟล์ .jsp ใหม่ไม่ว่าด้วยเหตุผลใดก็ตาม เวบเซิร์ฟเวอร์จะถือว่าไฟล์ .jsp นั้นเป็นไฟล์ใหม่ที่ไม่เคยมีการเข้าถึงมาก่อน การเยี่ยมชมในครั้งต่อไปจะมีการแปลงไฟล์และคอมไพล์ใหม่เหมือนกับเป็นการเข้าชมครั้งแรก

ติดตั้ง Tomcat

เวบเซิร์ฟเวอร์ที่สนับสนุนเจเอสพีมีอยู่หลายตัวในท้องตลาด เช่น JRun, LiteWebServer แต่ตัวที่เราจะใช้เป็นตัวอย่างในหนังสือเล่มนี้ ได้แก่ **Tomcat** ซึ่งมีข้อดีคือฟรี ในบทนี้เราจะสาธิตการติดตั้ง Tomcat บนระบบปฏิบัติการวินโดว์ และใช้มันในการทดลองรันไฟล์ .jsp ตัวอย่างในบทต่อๆ ไป คุณสามารถดาวน์โหลดไบนารีของ Tomcat เพื่อนำมาติดตั้งบนเครื่องคอมพิวเตอร์ของคุณได้ที่ http://jakarta.apache.org/tomcat ขอแนะนำให้ใช้เวอร์ชั่นอะไรก็ได้ตั้งแต่ 4.0 ขึ้นไป แต่ถ้าจะให้ดีที่สุดควรเลือกใช้เวอร์ชั่นที่ใหม่ที่สุดในขณะนั้นและเป็นเวอร์ชั่นที่ออกแล้วอย่างเป็นทางการ เวอร์ชั่น 4.0 ขึ้นไปสนับสนุน เจเอสพี เวอร์ชั่น 1.2 ซึ่งเป็นเวอร์ชั่นอ้างอิงของหนังสือเล่มนี้

ในการติดตั้ง Tomcat บนเครื่องคอมพิวเตอร์ส่วนตัวของคุณ เราใช้เครื่องคอมพิวเตอร์เครื่องเดียวกันนี้เป็นทั้งเวบเซิร์ฟเวอร์และเบราเซอร์ คุณจะใช้เบราเซอร์ที่มีอยู่ในเครื่องของคุณในการติดต่อกับ Tomcat บนเครื่องเดียวกัน จึงสามารถใช้งานได้โดยไม่ต้องมีอุปกรณ์เนตเวิร์คหรือต้องติดต่อกับอินเตอร์เนตแต่ประการใด

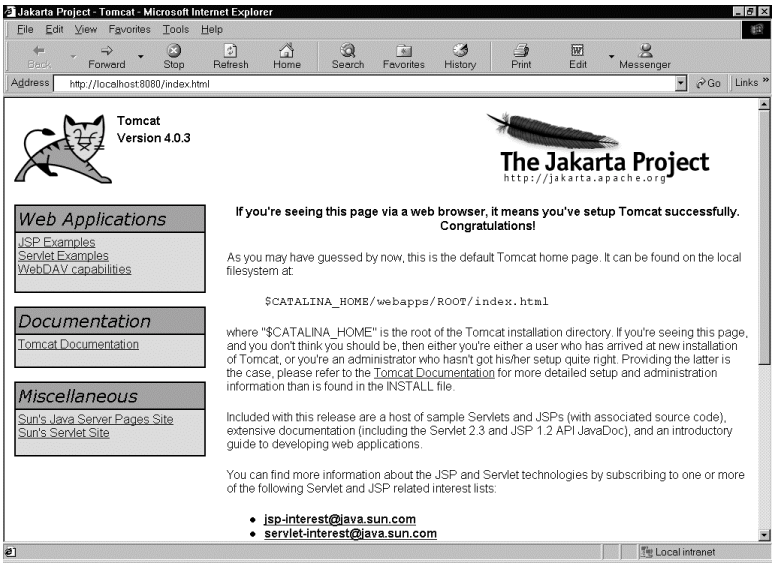
หนังสือเล่มนี้สาธิตการติดตั้ง Tomcat เวอร์ชั่น 4.0.3 ซึ่งคอมพิวเตอร์ที่จะใช้ติดตั้งได้ต้องมีคุณสมบัติดังต่อไปนี้

1. เป็นระบบปฏิบัติการวินโดว์อะไรก็ได้ตั้งแต่ วินโดว์ 95 ขึ้นไป ไม่ว่าจะเป็น 98, Me, XP, NT หรือ 2000
2. มีการติดตั้ง Java 2 SDK เวอร์ชั่น 1.2 หรือสูงกว่าเอาไว้แล้ว และสามารถคอมไพล์และรันโปรแกรมภาษาจาวาได้จริง อย่าลืมเซต PATH ให้ถูกต้อง
3. สร้างตัวแปรแวดล้อมของดอสชื่อ JAVA_HOME ไว้และให้มีค่าเท่ากับชื่อโฟลเดอร์ที่ติดตั้ง Java 2 SDK เอาไว้ (เช่น C:\java ในกรณีของผู้ที่ติดตั้ง Java 2 SDK ตามวิธีที่ระบุไว้ในหนังสือ จาวา สำหรับผู้เริ่มต้น)

การติดตั้ง Tomcat เริ่มจากการดาวน์โหลดตัวโปรแกรมจาก http://jakarta.apache.org/tomcat ซึ่งถูกบีบอัดไว้ในรูปของไฟล์ .zip เมื่อดาวน์โหลดเสร็จให้ทำการขยายออกบนที่ใดก็ได้ในฮาร์ดดิสก์ ในที่นี้ขอให้ขยายออกบน C:\ ซึ่งจะได้ไฟล์ที่ขยายออกแล้วทั้งหมดอยู่ภายใต้โฟลเดอร์ๆ หนึ่ง ซึ่งมีชื่อค่อนข้างยาวขึ้นต้นด้วยคำว่า jakarta- เพื่อความสะดวกในการอ้างอิงในหนังสือเล่มนี้ขอให้เปลี่ยนชื่อโฟลเดอร์นี้เสียใหม่เป็น tomcat เท่านี้การติดตั้งก็เสร็จเรียบร้อยแล้ว

เวลาจะสตาร์ท Tomcat ก็ให้เรียกหน้าต่างดอสออกมา แล้ว cd เข้าไปที่โฟลเดอร์ C:\tomcat\bin จากนั้นพิมพ์ว่า startup.bat รอสักครู่จะมีหน้าต่างดอสอีกหน้าต่างหนึ่งโผล่ออกมา หน้าต่างนี้เป็นหน้าต่างบอกสถานะของการทำงานของ Tomcat อย่าปิดหน้าต่างนี้ตลอดเวลาที่เวบเซิร์ฟเวอร์ทำงานอยู่

ลองทดสอบว่าเวบเซิร์ฟเวอร์ใช้การได้หรือยังด้วยการเรียกเบราเซอร์บนเครื่องของคุณออกมาแล้วพิมพ์ http://localhost:8080 คุณควรเห็นเบราเซอร์ของคุณแสดงผลดังภาพ



NOTE

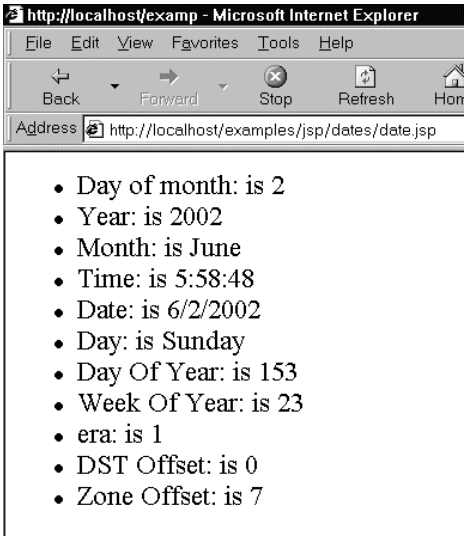
ถ้าไม่ได้ผลให้ลองแทนคำว่า localhost ด้วยคำว่า 127.0.0.1 ดังนี้ http://127.0.0.1:8080

NOTE

อันที่จริงเบราเซอร์ที่ใช้ทดสอบการทำงานของเจเอสพีจะเป็นเบราเซอร์ยี่ห้อใดก็ได้ เพราะ การใช้เจเอสพีไม่จำเป็นว่าเบราเซอร์จะต้องรู้จักภาษาจาวา เนื่องจากทุกสิ่งทุกอย่างเกิดขึ้นบนฝั่งเซิร์ฟเวอร์ ไม่เหมือนกับการใช้จาวาแอพเพลตที่ต้องมีการติดตั้งจาวาปลักอินบนเบราเซอร์ก่อน ตรงนี้นับเป็นข้อดีของการใช้เซิร์ฟเวอร์ไซด์สคริปต์ เพราะไม่ต้องคอยพะวงว่าเบราเซอร์ของผู้เยี่ยมชมจะไม่สนับสนุน

ถ้าคุณเห็นเหมือนกับภาพข้างต้นแสดงว่า Tomcat ของคุณทำงานเป็นเวบเซิร์ฟเวอร์ได้แล้ว แต่ยังไม่แน่ว่าจะสนับสนุนไฟล์ .jsp แล้วหรือไม่ เพราะเวบหน้านี้เป็นเวบตัวอย่างหน้าแรกของ Tomcat ซึ่งมีนามสกุล .html

ลองทดสอบการรันไฟล์นามสกุล .jsp ดูบ้างเพื่อดูว่า เจเอสพีคอนเทนเนอร์ ใน Tomcat ทำงานถูกต้องหรือไม่ ด้วยการคลิกเข้าไปที่ตัวเลือก JSP Examples แล้วลองคลิกคำว่า Execute ในบรรทัดที่มีคำว่า Date เบราเซอร์จะกระโดดไปยังเพจใหม่ที่มีนามสกุล .jsp ซึ่งสามารถแสดงวันเวลาปัจจุบันได้ ถ้าคุณพบเพจที่มีลักษณะคล้ายภาพข้างล่างนี้แสดงว่า Tomcat ของคุณทำงานรองรับเจเอสพีได้อย่างสมบูรณ์แบบแล้ว



NOTE

ถ้าไม่ได้ผลอาจเป็นเพราะการติดตั้ง Java 2 SDK บนเครื่องของคุณไม่ถูกต้อง หรืออาจเป็นเพราะได้เซตค่าตัวแปร path และ JAVA_HOME โปรดศึกษาวิธีการติดตั้ง Java 2 SDK ได้จากหนังสือ จาวา สำหรับผู้เริ่มต้น

ถ้าต้องการจบการทำงานของเวบเซิร์ฟเวอร์ให้เรียกดอส แล้ว cd เข้าไปในโฟลเดอร์ C:\tomcat\bin จากนั้นเรียกคำสั่ง shutdown.bat เวบเซิร์ฟเวอร์จะหยุดทำงาน เราควรจบการทำงานของเวบเซิร์ฟเวอร์เองทุกครั้งก่อนที่จะมีการปิดเครื่องคอมพิวเตอร์

NOTE

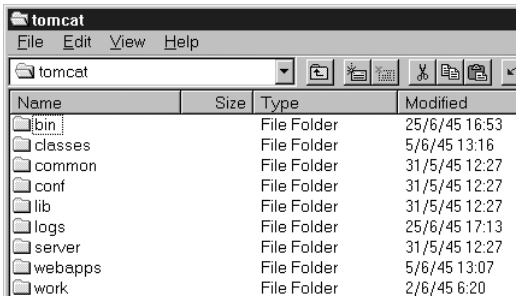
ถ้าต้องการให้เวบเซิร์ฟเวอร์ยอดนิยมอย่าง Apache , iPlanet หรือ ไมโครซอฟท์ IIS สนับสนุนเจเอสพี ให้หาโปรแกรมส่วนขยายของมันมาติดตั้ง ซึ่งได้แก่ ServletExec

NOTE

ถ้าต้องการทดลองสร้างเวบไซต์ที่สนับสนุนเจเอสพีที่เรียกใช้งานได้บนอินเตอร์เนตจริงๆ คุณต้องมีเวบเซิร์ฟเวอร์ที่ต่ออยู่กับอินเตอร์เนตตลอดเวลาซึ่งโดยมากจะใช้การเช่าเนื้อที่จากเวบไซต์ที่ให้บริการเวบโฮสต์ติ้งต่างๆ ถ้าไม่อยากเสียเงินให้ลองเข้าไปที่ http://www.mycgiserver.com มีบริการให้สร้างโฮมเพจส่วนตัวด้วยเจเอสพีให้สมัครฟรี

โครงสร้างของ Tomcat

ภายใต้โฟลเดอร์ C:\tomcat ซึ่งเป็นโฟลเดอร์หลักของ Tomcat มีโฟลเดอร์ย่อยที่สำคัญดังต่อไปนี้



โฟลเดอร์ webapps

โฟลเดอร์ webapps ใน Tomcat เป็นที่เก็บไฟล์นามสกุล .jsp .html หรือไฟล์ใดๆ ก็ตาม ที่ใช้ประกอบขึ้นเป็นเวบไซต์ของเรา ภายใต้โฟลเดอร์ webapps นี้เราจะสร้างโฟลเดอร์ย่อยอีกอย่างไรก็ได้เพื่อให้การจัดเก็บไฟล์ในเวบไซต์ของเรามีความเป็นระเบียบ

ถ้าผู้เยี่ยมชมพิมพ์ชื่อเวบไซต์ของเราบนเบราเซอร์โดยไม่ระบุชื่อไฟล์ Tomcat จะมองหาไฟล์ที่ชื่อว่า index ที่อยู่ในโฟลเดอร์ ROOT ซึ่งอยู่ใต้โฟลเดอร์ webapps อีกที แล้วนำมาแสดงผล ไฟล์ชื่อ index จึงใช้เป็นโฮมเพจหน้าหลักของเวบไซต์ โดยอาจมีนามสกุลเป็น .html .htm หรือ .jsp ก็ได้

โฟลเดอร์ work

โฟลเดอร์ work เป็นโฟลเดอร์ที่ Tomcat ใช้เก็บไฟล์นามสกุล .java และ .class ซึ่ง Tomcat สร้างขึ้นจากไฟล์ .jsp และเพื่อนำไปรันบน เจเอสพี คอนเทนเนอร์ ไฟล์ .java และ .class เหล่านี้จะเก็บอยู่อย่างเป็นระบบระเบียบภายใต้โฟลเดอร์ work โดย Tomcat จะสร้างโฟลเดอร์ย่อยที่มีชื่อเหมือนโฮสต์เนมของเครื่องคอมพิวเตอร์ของคุณอยู่ ภายใต้โฟลเดอร์นี้จะมีโฟลเดอร์ย่อยซึ่งมีโครงสร้างเหมือนกับโฟลเดอร์ย่อยที่สร้างไว้ในโฟลเดอร์ webapps โดย Tomcat จะเก็บไฟล์นามสกุล .java และ .class ที่เกิดจากไฟล์ .jsp ไว้ในตำแหน่งที่ตรงกันกับตำแหน่งของไฟล์ .jsp ที่อยู่ในโฟลเดอร์ webapps โดยใช้ชื่อเหมือนกับชื่อของไฟล์ .jsp แต่ตามด้วยคำว่า $jsp

โดยปกติแล้ว เราไม่ควรไปแตะต้องโฟลเดอร์ work เพราะ Tomcat จะเป็นผู้ใช้โฟลเดอร์นี้เองโดยอัตโนมัติเวลาที่มันคอมไพล์ไฟล์ .jsp

โฟลเดอร์ bin

โฟลเดอร์ bin ใช้เก็บยูทิลิตี้ต่างๆ ของ Tomcat ตัวที่สำคัญก็ได้แก่ ไฟล์ startup.bat และ shutdown.bat ที่ใช้สตาร์ทและหยุดการทำงานของ Tomcat

โฟลเดอร์ conf

โฟลเดอร์ conf ใช้เก็บไฟล์สำคัญที่ใช้กำหนดสถานะของ Tomcat ไฟล์ที่สำคัญมากได้แก่ ไฟล์ server.xml การแก้ไขเนื้อหาของไฟล์ต่างๆ ในโฟลเดอร์นี้ต้องทำด้วยความระมัดระวัง เพราะอาจทำให้ Tomcat ทำงานผิดพลาดได้

NOTE

เวบเซิร์ฟเวอร์ทีสนับสนุนเจเอสพีตัวอื่น จะมีโครงสร้างของโฟลเดอร์ต่างๆ แตกต่างไปจากของ Tomcat แต่หลักการคร่าวๆ จะเหมือนกัน ตัวอย่างเช่น JRun เก็บไฟล์ .jsp ไว้ใต้โฟลเดอร์ชื่อ server/default/default-app แทนที่จะเป็นโฟลเดอร์ webapps รายละเอียดโปรดศึกษาจากคู่มือการติดตั้งของเวบเซิร์ฟเวอร์ตัวนั้นๆ ที่คุณใช้อยู่

การตั้งค่าสถานะของ Tomcat

ใต้โฟลเดอร์ conf มีไฟล์ชื่อ server.xml อยู่ ไฟล์นี้เป็นไฟล์ที่ใช้กำหนดสถานะต่างๆ ของ Tomcat โดยเซิร์ฟเวอร์จะอ่านไฟล์นี้ทุกครั้งที่เริ่มทำงาน แต่จะไม่อ่านไฟล์นี้อีกตลอดเวลาที่ทำงาน ดังนั้นหากเราต้องการแก้ไขสถานะต่างๆ ของเวบไซต์ของเรา ต้องมีการรีสตาร์ท Tomcat เสียใหม่ด้วยเพื่อให้เวบเซิร์ฟเวอร์รับรู้ค่าใหม่

เนื้อหาของไฟล์ server.xml ถูกเขียนอยู่ในรูปแบบของคำสั่ง XML ซึ่งมีลักษณะเฉพาะที่ Tomcat เข้าใจ ค่าปกติต่างๆ ของ Tomcat มีอยู่อย่างมากมาย ในที่นี้เราจะมารู้จักกับวิธีการเซตค่าปกติตัวที่สำคัญๆ บางตัวเท่านั้น ดังต่อไปนี้

การตั้งชื่อโฮสต์

ชื่อโฮสต์ปกติที่ตั้งมาให้อยู่แล้วได้แก่ localhost แต่เวบเซิร์ฟเวอร์ตัวจริงของคุณจะมีชื่อโฮสต์เป็นชื่อที่มีโดเมนเนมพ่วงท้าย ถ้าคุณต้องการเซตชื่อโฮสต์จริงๆ ของคุณให้กับ Tomcat ให้ทำดังนี้

ใช้ Notepad เปิดดูเนื้อความในไฟล์ server.xml แล้วมองหาข้อความต่อไปนี้

```
<Engine name="Standalone" defaultHost="localhost" debug="0">
```

สมมติว่าเวบไซต์ของคุณมีชื่อว่า www.mysite.com ให้แก้ไขบรรทัดข้างต้นเสียใหม่เป็น

```
<Engine name="Standalone" defaultHost="www.mysite.com" debug="0">
```

บันทึกการแก้ไขแล้วรีสตาร์ท Tomcat เสียใหม่ ชื่อโฮสต์จะเปลี่ยนไปตามต้องการ ลองใช้เบราเซอร์พิสูจน์ด้วยการพิมพ์แอดเดรสว่า http://www.mysite.com:8080

อย่างไรก็ดีตัวอย่างในหนังสือเล่มนี้จะใช้ชื่อโฮสต์เป็น localhost เหมือนเดิมตลอดทั้งเล่ม คุณอาจเปลี่ยนกลับไปเป็น localhost ด้วยเพื่อให้เหมือนกับตัวอย่างในหนังสือ

การเซตค่าพอร์ต

พอร์ตค่าช่องทางเสมือนบนเน็ตเวิร์ก ซึ่งเซิร์ฟเวอร์เปิดรอไว้ให้บริการบนเน็ตเวิร์ก เซิร์ฟเวอร์ตัวหนึ่งมีพอร์ตหลายพอร์ต แต่ละพอร์ตจะมีหลายเลขประจำพอร์ต และจะให้บริการอย่างใดอย่างหนึ่งเพียงประเภทเดียว สำหรับบริการ HTTP ซึ่งเป็นบริการสำหรับการเข้าถึงเวบไซต์จะมีหมายเลขพอร์ตมาตรฐานเป็นเลข 80 ในขณะที่ค่าปกติของ Tomcat จะมีหมายเลขพอร์ตเป็น 8080 ด้วยเหตุนี้เวลาที่เราใช้เบราเซอร์เข้าถึงเวบไซต์เมื่อตอนต้นของบทนี้เราต้องระบุหมายเลขพอร์ตเอาเอง ด้วยการใช้เครื่องหมายโคลอน ตามด้วยหมายเลขพอร์ต ต่อท้ายชื่อโฮสต์ เราสามารถเปลี่ยนหมายเลขพอร์ตเสียใหม่เป็น 80 ด้วยวิธีการดังต่อไปนี้

มองหาข้อความต่อไปนี้ในไฟล์ server.xml

```
<Connector className="org.apache.catalina.connector.http.HttpConnector"
               port="8080" minProcessors="5" maxProcessors="75"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0" connectionTimeout="60000"/>
```

ทีนี้ลองแก้คำว่า port=”8080” ให้เป็น port=”80” แล้วบันทึก จากนั้นลองรีสตาร์ทเวบเซิร์ฟเวอร์ใหม่ดู แล้วใช้เบราเซอร์เข้าถึงโดยพิมพ์ว่า http://localhost เฉย ๆ จะพบว่าสามารถเข้าถึงหน้าตัวอย่างได้ตามปกติ ไม่จำเป็นต้องมีหมายเลขพอร์ต เพราะพอร์ต 80 เป็นหมายเลขพอร์ตมาตรฐานสำหรับบริการ HTTP ซึ่งเบราเซอร์รู้อยู่แล้ว

ต่อไปนี้เราถือว่า Tomcat ของเราใช้พอร์ต 80 ตลอดเนื้อหาของหนังสือเล่มนี้

ตอนนี้คุณก็มีเวบเซิร์ฟเวอร์ที่สนับสนุนเจเอสพีไว้ใช้ในการศึกษาเจเอสพีเป็นที่เรียบร้อยแล้ว ในบทต่อไปเราก็จะเริ่มศึกษาโครงสร้างคำสั่งเจเอสพีกันเลย

3

บล็อกของคำสั่งเจเอสพี

ในบทนี้คุณจะได้เรียนรู้วิธีการสร้างไฟล์ .jsp เพื่อใช้สร้างโฮมเพจของคุณด้วยการเขียนไฟล์ .jsp ไฟล์แรกในชีวิต ไฟล์นี้มีความสามารถพิเศษคือการนับจำนวนผู้ที่เคยเข้าเยี่ยมชมตัวมันเองได้

การสร้างไฟล์ .jsp

การสร้างไฟล์ .jsp ทั่วไปเริ่มจากการใช้ Notepad ในการสร้าง ตอนนี้ให้คุณสร้างไฟล์ชื่อ counter01.jsp แล้วพิมพ์ข้อความต่อไปนี้ลงไป

โปรแกรมที่ 3-1 `counter01.jsp`

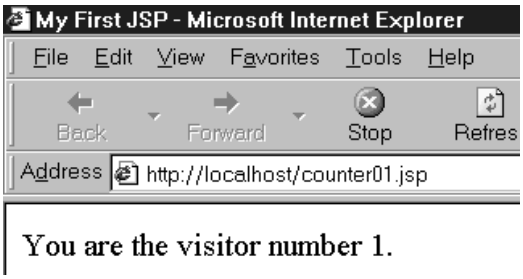
```
<%! int count = 0; %>
<html>
<title>My First JSP</title>
<body>
You are the visitor number <%= ++count %>.
</body>
</html>
```

จากนั้นก็เซฟไฟล์นี้ลงในโฟลเดอร์ C:\tomcat\webapps\ROOT

NOTE

ระวังอย่าให้นามสกุลของไฟล์ .jsp กลายเป็น .txt

ถ้าแน่ใจว่า Tomcat ของคุณกำลังทำงานอยู่ ลองเปิดเบราเซอร์ของคุณขึ้นมาแล้วพิมพ์ที่อยู่ http://localhost/counter.jsp ลงไปผลที่ได้ควรจะเป็นดังนี้



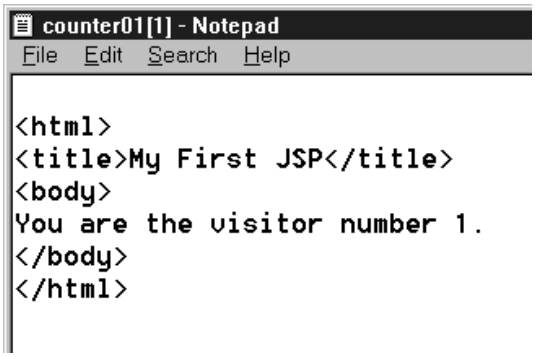
ถ้าลองรีเฟรชเบราเซอร์ดูหลายๆ ครั้ง จะเห็นได้ว่าหมายเลขผู้เยี่ยมชมเพิ่มขึ้นได้เองทีละหนึ่ง การรีเฟรชเปรียบได้กับการร้องขอโฮมเพจหน้านี้ซ้ำใหม่

โดยปกติการสร้างเวบไซต์ด้วยเจเอสพีก็ทำได้ง่ายๆ เพียงแค่นี้ เริ่มจากสร้างไฟล์ .jsp ขึ้นมาด้วย Notepad ซึ่งไฟล์เจเอสพีก็คือไฟล์ HTML ที่มีคำสั่งเจเอสพีปะปนอยู่ แล้วก็นำไฟล์นี้ไปวางไว้ใต้โฟลเดอร์ webapps เพียงแค่นี้ก็ได้โฮมเพจที่ต้องการสมใจ

คำสั่งเจเอสพีในไฟล์ counter01.jsp ได้แก่คำสั่งที่ล้อมรอบด้วยเครื่องหมาย <% และ %> คำสั่งแรกได้แก่คำสั่ง <%! int count = 0; %> เป็นคำสั่งประกาศตัวแปรแบบจำนวนเต็มชื่อ count และกำหนดให้มีค่าเริ่มต้นเป็น 0 จะเห็นได้ว่าคำสั่งเจเอสพีก็คือคำสั่งภาษาจาวาธรรมดาๆ นี่เอง แต่เขียนอยู่ภายใต้เครื่องหมาย <% และ %> (ต่อไปนี้จะเรียกว่าบล็อกเจเอสพี) สังเกตว่าในกรณีนี้บล็อกเจเอสพีมีเครื่องหมายอัศเจรีย์ตามหลังเครื่องหมาย <% อยู่ด้วย แต่ตอนนี้เราจะยังไม่อธิบายว่าทำไม

คำสั่งเจเอสพีอีกคำสั่งหนึ่งได้แก่ คำสั่ง <%= ++count %> ซึ่งเป็นคำสั่งให้บวกค่าปัจจุบันของ count ด้วย 1 และให้แสดงค่าของตัวแปร count ที่ได้ออกมาบนเบราเซอร์ สังเกตว่าคำสั่งเจเอสพีคือคำสั่งภาษาจาวาที่ถูกล้อมด้วยบล็อกเจเอสพีอีกเช่นเคย แต่คราวนี้มีเครื่องหมาย = ตามหลังเครื่องหมาย <%

สิ่งที่น่าสังเกตอีกอย่างหนึ่งคือ เบราเซอร์จะมองไม่เห็นคำสั่งเจเอสพี สิ่งที่ Tomcat ส่งไปให้เบราเซอร์จริงๆ จะเป็นคำสั่ง HTML ล้วนๆ ไม่มีคำสั่งเจเอสพีปะปน ลองพิสูจน์ด้วยการทดลองดูซอร์สโค้ดบนเบราเซอร์ของคุณด้วยการเลือก View > Source (สำหรับไมโครซอฟท์อินเตอร์เนตเอ็กซ์พลอเลอร์) จะเห็นได้ว่าซอร์สโค้ดเป็นคำสั่ง HTML ล้วนๆ ไม่มีคำสั่งเจเอสพีอยู่ ที่เป็นเช่นนี้เพราะ Tomcat รันคำสั่งเจเอสพีให้เสร็จเสียก่อนบนเซิร์ฟเวอร์ แล้วแทนคำสั่งเจเอสพีเหล่านั้นด้วยผลลัพธ์ที่เหมาะสม ก่อนที่จะส่งผลไปให้เบราเซอร์ในรูปของคำสั่ง HTML ล้วนๆ นับเป็นข้อดีอีกอย่างหนึ่งของการใช้เจเอสพี กล่าวคือเบราเซอร์ของผู้เยี่ยมชมไม่จำเป็นต้องสนับสนุนจาวาแต่ประการใด



บล็อกเจเอสพี

เราทราบแล้วว่าบล็อกของคำสั่งเจเอสพีล้อมรอบด้วยเครื่องหมาย <% และ %> ที่จริงแล้วบล็อกเจเอสพียังแบ่งออกได้อีกเป็นสามประเภท ได้แก่ บล็อกประกาศตัวแปร บล็อกแสดงค่าของตัวแปร และบล็อกของสคริปเลต เครื่องหมายอัศเจรีย์ และเครื่องหมายเท่ากับ ที่อยู่หลังเครื่องหมาย <% ในโปรแกรมที่ 3-1 ล้วนเป็นสิ่งที่ใช้แยกความแตกต่างระหว่างบล็อกเจเอสพีแต่ละประเภท

บล็อกของคำสั่งเจเอสพีมีหลายประเภท ดังต่อไปนี้

บล็อกประกาศตัวแปร

บล็อกประกาศตัวแปรคือบล็อกเจเอสพีที่ถูกล้อมด้วยเครื่องหมาย <%! และ %> คำสั่งเจเอสพีที่อยู่ภายในบล็อกประกาศตัวแปรต้องเป็นคำสั่งประกาศตัวแปรหรือแมธธอสในภาษาจาวา และอาจมีมากกว่าหนึ่งคำสั่งก็ได้ ตัวอย่างของบล็อกประกาศตัวแปรที่ผ่านมาแล้วในโปรแกรมที่ 3-1 ได้แก่

```
<%! int count = 0; %>
```

คำสั่งในบล็อกนี้เป็นการประกาศตัวแปรจำนวนเต็มชื่อ count และกำหนดให้มีค่าเริ่มต้นเป็น 0 โปรดสังเกตว่าต้องจบคำสั่งด้วยเครื่องหมาย ; เสมอตามหลักภาษาจาวาทั่วไป ถ้าต้องการประกาศตัวแปรมากกว่าหนึ่งตัวก็สามารถทำได้เช่น

```
<%! int i = 0;
    double j = 0.0;
 %>
```

แบบนี้เป็นการประกาศตัวแปรสองตัวในบล็อกประกาศตัวแปรบล็อกเดียว คือ ตัวแปรจำนวนเต็ม i และ ตัวแปรทศนิยมแบบยาว j โดยกำหนดให้มีค่าเริ่มต้นเท่ากับ 0 ทั้งคู่

นอกจากการประกาศตัวแปรแล้ว บล็อกประกาศตัวแปรยังใช้ประกาศแมธธอสได้อีกด้วย เช่น

```
<%! int square(int i) {
        return i*i;
    }
 %>
```

ตัวอย่างนี้เป็นการประกาศแมธธอสชื่อ square() ซึ่งสามารถคำนวนค่ายกกำลังสองของเลขจำนวนเต็มให้เราได้ เราสามารถเรียกใช้แมธธอส square() นี้ที่ใดก็ได้ในไฟล์ .jsp ของเราเสมือนเป็นแมธธอสช่วยเหลือ

สิ่งสำคัญที่ควรจดจำเกี่ยวกับบล็อกประกาศตัวแปรก็คือ ตัวแปรในบล็อกประกาศตัวแปรจะถูกสร้างขึ้นและกำหนดค่าเริ่มต้นเพียงครั้งเดียวตอนที่ เจเอสพี คอนเทนเนอร์ โหลดโปรแกรมนี้เป็นครั้งแรก ซึ่งก็คือตอนที่มีผู้เยี่ยมชมเวบเพจหน้านั้นๆ เป็นครั้งแรก การเยี่ยมชมในครั้งต่อๆ ไปของเวบเพจหน้าเดิมจะใช้ตัวแปรตัวเดิมที่ได้สร้างไว้แล้ว ด้วยเหตุนี้เราจึงอาศัยค่าของตัวแปร count ในการนับจำนวนผู้เยี่ยมชมได้ เพราะค่าของตัวแปร count เพิ่มขึ้นทีละหนึ่งทุกครั้งที่มีผู้เยี่ยมชมจากผลของคำสั่ง ++count

บล็อกแสดงค่าตัวแปร

บล็อกแสดงค่าตัวแปร คือ บล็อกเจเอสพีที่ล้อมรอบด้วยเครื่องหมาย <%= และ %> ดังในโปรแกรมที่ 3-1 ข้างต้นมีบล็อกแสดงค่าตัวแปรอยู่หนึ่งบล็อกได้แก่

```
<%= ++count %>
```

คำสั่งในบล็อกนี้อาจเป็นได้ทั้งตัวแปร แมธธอส หรือวัตถุต่างๆ ซึ่งมีค่าในตัวของมันเอง เวลาเจเอสพี คอนเทนเนอร์เจอบล็อกแสดงค่าตัวแปร มันจะแสดงค่าของตัวแปร แมธธอส หรือวัตถุนั้นๆ ในตำแหน่งนั้นออกมา เช่นในกรณีข้างต้นคำสั่งนี้เป็นการบอกให้บวกค่าปัจจุบันของตัวแปร count ด้วยหนึ่ง แล้วแสดงค่าของตัวแปร count สิ่งที่ได้คือเบราเซอร์จะแสดงค่าปัจจุบันของตัวแปร count ซึ่งก็คือตัวเลขบอกจำนวนผู้เยี่ยมชมออกมา สังเกตว่าคำสั่งในบล็อกนี้ไม่ต้องมีเครื่องหมาย ; ต่อท้าย

ตัวอย่างอีกตัวอย่างหนึ่งของบล็อกแสดงค่าของตัวแปรที่เคยผ่านมาแล้วคือ คำสั่งในโปรแกรมที่ 1-2 ได้แก่คำสั่ง <%= new java.util.Date() %> ซึ่งแสดงค่าของวัตถุของคลาส Date ซึ่งมีค่าเท่ากับวันเวลาปัจจุบันนั่นเอง

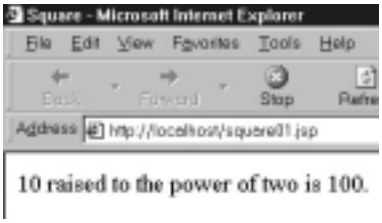
บล็อกแสดงค่าของตัวแปรยังสามารถแสดงค่าของแมธธอสได้อีกตัวย ดังตัวอย่างต่อไปนี้

โปรแกรมที่ 3-2 `square01.jsp`

```
<%! int i = 10;
    int square(int i) {
        return i*i;
    }
 %>
```

```
<html>
<title>Square</title>
<body>
<%=i%> raised to the power of two is <%=square(i)%>.
</body>
</html>
```

โปรแกรมนี้คำนวนค่ายกกำลังสองให้เรา โดยเริ่มจากการประกาศตัวแปรจำนวนเต็มชื่อ i โดยกำหนดให้มีค่าเท่ากับ 10 แล้วประกาศแมธธอส square() จากนั้นเราใช้บล็อกแสดงค่าของตัวแปรแสดงค่าของยกกำลังสองของสิบออกมา ผลที่ได้เป็นดังภาพ



บล็อกสคริปเลต

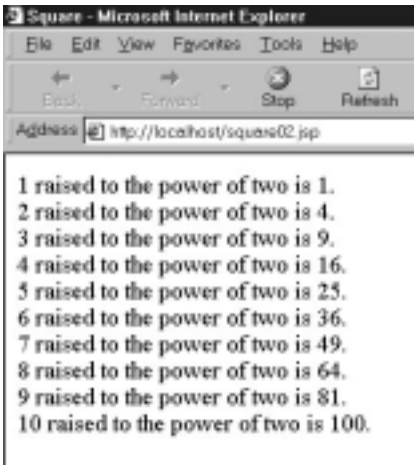
บล็อกเจเอสพีประเภทสุดท้ายได้แก่ บล็อกสคริปต์เลต ซึ่งถูกล้อมรอบด้วยเครื่องหมาย <% และ %> ธรรมดา

สคริปต์เลตคือคำสั่งหรือกลุ่มของคำสั่งจาวาอะไรก็ได้ ลองพิจารณาโปรแกรมต่อไปนี้

โปรแกรมที่ 3-3 square02.jsp

```
<%! int square(int i) {
        return i*i;
    }
 %>
<html>
<title>Square</title>
<body>
<% for (int i = 1; i <= 10; i++) { %>
<%=i%> raised to the power of two is <%=square(i)%>.
<% } %>
</body>
</html>
```

ผลที่ได้คือ



ไฟล์ .jsp ข้างต้นแสดงค่ายกกำลังสองของเลข 1 ถึง 10 โดยการเรียกแมธธอส square() ที่ประกาศไว้ในตอนต้นของไฟล์ แล้วใช้คำสั่ง for วนลูป 10 รอบ ให้สังเกตเทคนิคการใช้สคริปเลตปะปนกับคำสั่ง HTML

การใช้สคริปต์เลตทุ่นแรง

จากตัวอย่างข้างต้นจะเห็นได้ว่าบล็อกสคริปเลตสามารถทำให้คำสั่งเจเอสพีอยู่ปะปนกับคำสั่ง HTML ได้อย่างกลมกลืนราวกับว่าเป็นภาษาเดียวกัน สิ่งที่ต้องระวังก็คือ สคริปเลตต้องอยู่ภายใต้วงล้อมของเครื่องหมาย <% และ %> เสมอแม้ว่าจะเป็นเพียงส่วนเล็กส่วนน้อยของคำสั่งเจเอสพี อย่างวงเล็บปีกกาแค่อันเดียว เป็นต้น เราสามารถอาศัยสคริปเลตในการทุ่นเรงในการเขียนคำสั่ง HTML ซึ่งบ่อยครั้งคำสั่ง HTML ค่อนข้างจะซ้ำซ้อนและเยิ่นเย้อ

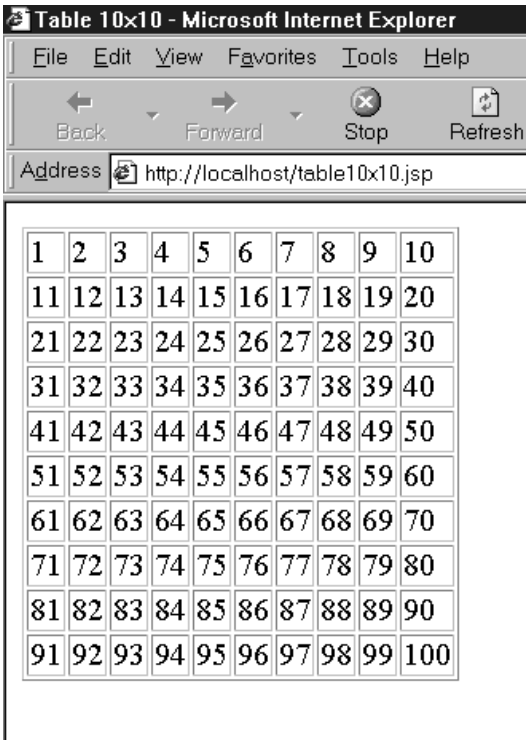
ตัวอย่างเช่น ถ้าเราต้องการสร้างหน้าเวบที่มีตารางขนาด 10 x 10 ช่อง ภายในตารางมีตัวเลขหนึ่งถึงร้อยเขียนเรียงต่อกันไปเรื่อยๆ จากช่องซ้ายไปขวาและจากบนลงล่าง ถ้าเราใช้คำ

สั่ง HTML ล้วนๆ คงได้ไฟล์ HTML ที่ยาวน่าดู เราสามารถเอาสคริปต์เลตมาช่วยทุนแรงได้ดังนี้

โปรแกรมที่ 3-4 table10x10.jsp

```
<html>
<title>Table 10x10</title>
<body>
<table border="1" >
<% for (int i = 0; i < 10; i++) { %>
<tr>
<% for (int j = 1; j <= 10; j++) { %>
<td><%=10*i + j %></td>
<% } %>
</tr>
<% } %>
</table>
</body>
</html>
```

ลองใช้เบราเซอร์เข้าถึงเพจข้างต้นจะได้ผลดังนี้



1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

โปรแกรมนี้มีการใช้คำสั่งวนลูปในการเขียนคำสั่ง HTML ที่ใช้วาดตารางๆ ซ้ำๆ สังเกตว่าสคริปต์เลตสามารถอยู่ปะปนกับคำสั่ง HTML ได้อย่างกลมกลืน ข้อสำคัญต้องอย่าลืมใส่เครื่องหมาย <% และ %> ล้อมรอบสคริปต์เลตทุกครั้ง

การประกาศตัวแปรภายในบล็อกสคริปเลต

ที่ผ่านมาเราประกาศตัวแปรภายในบล็อกประกาศตัวแปร แต่เนื่องจากคำสั่งที่อยู่ในบล็อกสคริปเลตจะเป็นคำสั่งอะไรก็ได้ในภาษาจาวารวมทั้งคำสั่งในการประกาศตัวแปรด้วย ตัวแปรจึงประกาศได้ทั้งในบล็อกประกาศตัวแปร และในบล็อกสคริปเลต แล้วมันต่างกันอย่างไร?

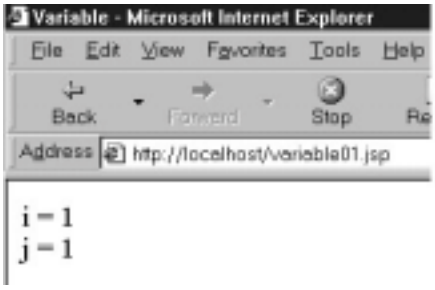
ตัวแปรที่ประกาศในบล็อกประกาศตัวแปรจะถูกสร้างขึ้นเพียงตัวเดียวต่อหนึ่งเพจ ผู้เยี่ยมชมทุกรายที่เข้าถึงเพจเดียวกันจะใช้ตัวแปรตัวนี้ร่วมกัน ในขณะที่ตัวแปรที่ประกาศในบล็อก

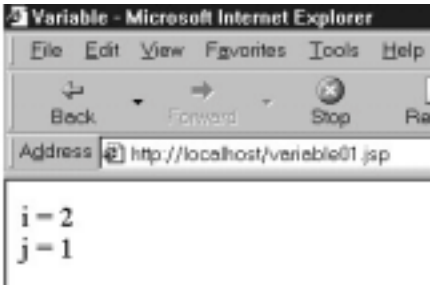
สคริปเลตจะถูกสร้างขึ้นและตายไปทุกครั้งที่มีผู้เยี่ยมชมเข้าถึงเวบเพจ ลองพิจารณาตัวอย่างต่อไปนี้

โปรแกรมที่ 3-5 `variable01.jsp`

```
<%! int i = 0; %>
<html>
<title>Variable</title>
<body>
<% int j = 0; %>
i = <%= ++i %> <br>
j = <%= ++j %> <br>
</body>
</html>
```

ลองเข้าถึงเพจนี้ติดต่อกันสองครั้งด้วยการรีเฟรชจะได้ผลดังนี้





สาเหตุที่ค่าของตัวแปร i เพิ่มขึ้น แต่ค่าของตัวแปร j ไม่เพิ่มเป็นเพราะตัวแปร i ถูกประกาศไว้ในบล็อกประกาศตัวแปร มันจึงเป็นตัวแปรตัวเดิมเวลาที่มีผู้เยี่ยมชมเข้าถึงมากกว่าหนึ่งครั้ง ค่าที่เพิ่มขึ้นของตัวแปร i จะต่อยอดไปเรื่อยๆ ในขณะที่ตัวแปร j ถูกประกาศไว้ในบล็อกสคริปเลตมันจึงถูกสร้างใหม่ทุกครั้งที่มีการเข้าถึงโดยผู้เยี่ยมชมรายใหม่ ค่าของมันจึงไม่เพิ่มขึ้นไปจาก 1

เรานิยมใช้บล็อกประกาศตัวแปรในการประกาศตัวแปรที่มีไว้ใช้เป็นพวกค่าคงที่ นอกนั้นเรานิยมประกาศตัวแปรในสคริปเลต เพื่อประหยัดหน่วยความจำ (เกิดขึ้นแล้วตายไปทันที) ยก

เว้นตัวแปรที่ใช้เป็นตัวนับจำนวนผู้เยี่ยมชม ซึ่งต้องมีค่าต่อเนื่องจากการเข้าถึงครั้งหนึ่งไปสู่อีกครั้งหนึ่ง

เราสามารถสร้างตัวแปรในบล็อกสคริปเลตที่มีชื่อเดียวกับตัวแปรที่ประกาศไว้แล้วในบล็อกประกาศตัวแปรได้ แต่ตัวแปรที่สร้างในสคริปเลตจะบดบังตัวแปรที่สร้างในบล็อกประกาศตัวแปร กล่าวคือเวลาอ้างถึงตัวแปรชื่อนั้นในสคริปเลต มันจะถือว่าหมายถึงตัวแปรตัวที่ประกาศในไว้ในสคริปเลตเสมอ

คอมเมนท์ในเจเอสพี

ตอนนี้เราได้เรียนรู้แล้วว่าบล็อกเจเอสพีมีสามแบบคือ บล็อกประกาศตัวแปร บล็อกแสดงค่าของตัวแปร และบล็อกสคริปเลต ซึ่งใช้เครื่องหมายกำกับบล็อกต่างกัน ที่จริงแล้วยังมีบล็อกของเจเอสพีอีกแบบหนึ่งคือ บล็อกของคอมเมนท์ ซึ่งใช้เครื่องหมาย **<%--** และ **--%>** ล้อมรอบคอมเมนท์ คอมเมนท์มีไว้ช่วยความจำของผู้เขียนโปรแกรมเองด้วยการเขียนคำอธิบายต่างๆ เวบเซิร์ฟเวอร์จะละเลยข้อความที่อยู่ข้างใน ตัวอย่างเช่น

โปรแกรมที่ 3-6 `variable02.jsp`

```
<%-- i is declared in declaration block--%>
<%! int i = 0; %>
<html>
<title>Variable</title>
<body>
<%-- i is declared in scriplet--%>
<% int j = 0; %>
i = <%= ++i %> <br>
j = <%= ++j %> <br>
</body>
</html>
```

โปรแกรมนี้จะให้ผลเหมือนโปรแกรมที่ 3-6 ทุกประการ คอมเมนท์ที่เพิ่มเติมขึ้นมาไม่มีผลต่อการทำงานแต่ประการใด

ปฏิทินอิเล็กทรอนิกส์

ก่อนจะจากบทนี้ไป เรามีตัวอย่างการใช้เจเอสพีในการสร้างปฏิทินอิเล็กทรอนิกส์ที่แสดงวันของเดือนปัจจุบัน โดยมีการจัดวางตำแหน่งของวันที่ให้ตรงกับวันของสัปดาห์ได้อย่างถูกต้อง และจะแสดงวันที่ของวันนี้ด้วยตัวหนา ดังนี้

โปรแกรมที่ 3-7 calendar.jsp

```
<%! String[] month =

{"","January","February","March","April","May","June","July","August","S
   eptember","October","November","December" };

    int[] numberOfDays ={0,31,28,31,30,31,30,31,31,30,31,30,31};
%>

<% java.text.SimpleDateFormat formatter = new
   java.text.SimpleDateFormat("dd-MM-yyyy");
   String today = formatter.format(new java.util.Date());

   java.util.Date firstdate = formatter.parse("01"+today.substring(2));
   formatter = new java.text.SimpleDateFormat("F");
   int firstDayOfMonth = Integer.parseInt(formatter.format(firstdate));

   int dd = Integer.parseInt(today.substring(0,2));
   int MM = Integer.parseInt(today.substring(3,5));
   String yyyy = today.substring(6);

   int start = 0;
   int stop = numberOfDays[MM];

%>

 <html>
 <title>Calendar</title>
 <body>
 <table border="1" >
 <tr><td colspan="7" align="center"><%= month[MM] %> - <%= yyyy
   %></td></tr>
 <tr><td>Su</td><td>Mo</td><td>Tu</td><td>W</td>
     <td>Th</td><td>Fr</td><td>Sa</td>
 </tr>
 <% for (int i = 1; i <=6 ; i++) { %>
     <tr>
 <% for (int j = 1; j <= 7; j++) { %>
     <td> <% if (i==1 && j==firstDayOfMonth) { start=1; };
             if (dd==start) { %><b><% };
             if (start>0&&start<=stop) {%> <%= start++ %><% }
             if (dd==start+1) { %></b><% }; %>
```

```
 </td>
<% } %>
</tr>
<% } %>
</table>
</body>
</html>
```

โปรแกรมนี้เริ่มต้นด้วยการประกาศอะเรย์สองตัวชื่อ month และ numberOfDays ซึ่งเก็บชื่อเดือน และวันที่ โดยเลขดรรชนีของชื่อเดือนและวันที่จะตรงกับลำดับของเดือนและวันที่นั้นๆ พอดี

จากนั้นในสคริปเลตเป็นการใช้คำสั่งเจเอสพีในการหาค่าวันเวลาปัจจุบัน โดยที่เวลาแสดงผลให้แสดงผลในรูปแบบ dd-MM-yyyy แทนที่จะเป็นรูปแบบปกติของมัน (แบบที่เห็นในบทที่ 1) การกำหนดการแสดงรูปแบบวันที่ใช้คลาส SimpleDateFormat ซึ่งอยู่ในแพจเกจมาตรฐาน java.text ช่วย เมื่อได้แล้วก็เก็บค่าวันที่ในรูปแบบนี้ไว้ในตัวแปรชื่อสตริง today

ขั้นตอนต่อไปเป็นการหาว่าวันที่ 1 ของเดือนปัจจุบันตรงกับวันอะไรของสัปดาห์ เพื่อให้สร้างปฏิทินได้ถูกต้อง เราใช้คำสั่ง substring() ในการเปลี่ยนวันที่ปัจจุบันที่เก็บไว้ในตัวแปร today ให้กลายเป็นวันที่ 1 ของเดือนเดียวกันปีเดียวกัน แล้วส่งเข้าไปในแมธธอส parse()

เพื่อให้คำนวนวันของสัปดาห์ออกมาให้เรา เมื่อได้แล้วก็เก็บวันของสัปดาห์ไว้ในตัวแปรจำนวนเต็มชื่อ FirstDayOfMonth (เลขหนึ่งแทนวันอาทิตย์ เรื่อยไปตามลำดับ)

ตอนนี้เรารู้แล้วว่าจะสร้างปฏิทินได้อย่างไร เพราะรู้ว่าวันที่ปัจจุบันเป็นวันที่เท่าไร เดือนอะไร ปีอะไร และวันในสัปดาห์วันแรกของเดือนเป็นวันอะไร เราก็แยกเก็บวันที่ เดือน และปี ไว้ในตัวแปรสตริงชื่อ dd MM yyyy ตามลำดับ พอเวลาแสดงผลก็ใช้การวนลูปเพื่อสร้างตารางปฏิทินขึ้น โดยอาศัยข้อมูลเกี่ยวกับวันที่และวันของสัปดาห์ที่เราหามาไว้ในตอนแรกของโปรแกรมเป็นหลักเกณฑ์ในการสร้าง ถ้าคุณยังรู้สึกว่าโปรแกรมปฏิทินนี้เข้าใจยาก อย่าเพิ่งตกใจ ตัวอย่างนี้เป็นเพียงตัวอย่างเพื่อให้เห็นภาพของการใช้งานเจเอสพีจริงๆ เท่านั้น ไม่ประสงค์ให้คุณเข้าใจทั้งหมดของโปรแกรมนี้ในเวลานี้

4

ไดเร็กทีฟ

ไดเร็กทีฟไม่ใช่คำสั่งในภาษาจาวา แต่เป็นคำสั่งที่บอกให้เวบเซิร์ฟเวอร์ทำอะไรบางอย่างก่อนที่จะรันคำสั่งเจเอสพีที่อยู่ในเพจ ไดเร็กทีฟมักอยู่ที่ส่วนบนสุดของไฟล์ .jsp และล้อมรอบด้วยเครื่องหมาย <%@ และ %> ซึ่งทำให้ดูคล้ายกับบล็อกของคำสั่งเจเอสพี ในบทนี้เราจะทำความรู้จักกับไดเร็กทีฟสองกลุ่มคือ include และ page

ไดเร็กทีฟ include

ไดเร็กทีฟ include มีไว้สำหรับสอดแทรกเนื้อหาของเวบหน้าอื่นเข้าไปในเวบหน้าปัจจุบัน ตัวอย่างการใช้งานที่เห็นได้ชัดก็คือ การสอดแทรกเมนูหลักหรือโลโก้ต่างๆ ที่ส่วนต้นของเวบเพจทุกเพจบนเวบไซต์ ข้อดีของการทำแบบนี้ก็คือเราเขียนเมนูหลักหรือโลโก้เหล่านั้นเพียงครั้งเดียวบนเพจหน้าหนึ่ง แล้วสามารถนำไปแทรกลงในเวบหน้าอื่นๆ ได้ด้วยการแทรกเพียงคำสั่งไดเร็กทีฟสั้นๆ ลงไป

ตัวอย่างเช่นถ้าเราเตรียมโลโก้หลักของเวบไซต์ของเราไว้ในไฟล์ชื่อ logo.html ซึ่งมีซอร์สโค้ดและหน้าตาดังตัวอย่างข้างล่าง

โปรแกรมที่ 4-1 `logo.html`

```
<center>
```

```
<img src="logo.gif" >
<br>
Welcome to Dekisugi.net
</center>
```

ที่นี่เราต้องการแทรกโลโก้นี้ไว้ในส่วนบนสุดของเวบเพจอื่น เราก็ทำได้โดยการแทรกไดเร็กทีฟ include ที่เรียกไฟล์ logo.html เข้าไปดังตัวอย่างต่อไปนี้

โปรแกรมที่ 4-2 `otherpage.jsp`

```
<html>
<title> Test Page </title>
<body>
<%@ include file="logo.html" %>
<br>
This is a test page.
</body>
</html>
```

เท่านี้โลโก้ที่เราเตรียมไว้ก็จะเข้าไปแทรกอยู่ในหน้าเวบของเราตำแหน่งที่ไดเร็กทีฟ include อยู่ พารามิเตอร์ file ก็คือชื่อของไฟล์ที่เก็บเนื้อหาที่ต้องการให้แทรกเข้าไป

อย่างไรก็ดี ข้อเสียของการใช้ไดเร็กทีฟ include ก็คือ วันหลังหากเราต้องการแก้ไขไฟล์ logo.html เสียใหม่ ไฟล์ .jsp อื่นๆ ที่ include โลโก้เข้าไปจะไม่ทำการอัพเดทให้โดยอัตโนมัติ เพราะมันดึงไฟล์ logo.html เข้าไปเพียงครั้งเดียวตั้งแต่ตอนที่มีผู้เยี่ยมชมเวบเพจเป็นครั้งแรก หลังจากนั้นหากไฟล์ logo.html ถูกแก้ไขมันจะไม่รับรู้ ถ้าต้องการให้รับรู้จำเป็นต้องมีการบังคับให้มันสร้างไฟล์ .java ขึ้นมาใหม่ซึ่งทำได้โดยการเข้าไปเปิดไฟล์ .jsp นั้นๆ แล้วบันทึกใหม่ทั้งๆ ที่ไม่มีการแก้ไข Tomcat จะตรวจสอบวันที่ของการเข้าถึงไฟล์ที่เปลี่ยนไปและสร้างไฟล์ .java ใหม่ทำให้มีการอัพเดทไฟล์ที่ include เข้ามา

เจเอสพี มีวิธีอื่นในการสอดแทรกเวบเพจที่สามารถอัพเดทไฟล์ปลายทางได้เองเวลาที่ไฟล์ที่ต้นทางถูกแก้ไข เรากล่าวถึงวิธีการนั้นอีกครั้งในบทถัดๆ ไป

ไดเร็กทีฟ page

ไดเร็กทีฟ page เป็นไดเร็กทีฟที่มีที่ใช้บ่อยกว่าไดเร็กทีฟ include และมีพารามิเตอร์อยู่หลายตัว ในบทที่แล้วเราได้เคยใช้ไดเร็กทีฟนี้ไปแล้วในการอิมพอร์ตแพจเกจซึ่งใช้พารามิเตอร์ import ในการระบุชื่อของแพจเกจที่ต้องการอิมพอร์ต พารามิเตอร์ตัวอื่นๆ ของไดเร็กทีฟ page ก็ได้แก่

language

พารามิเตอร์นี้ใช้บอกเวบเซิร์ฟเวอร์ว่าคำสั่งในบล็อกเจเอสพีที่ตามมาในเพจนั้นๆ เขียนด้วยภาษาคอมพิวเตอร์ภาษาใด ซึ่งค่าที่เป็นไปได้ของพารามิเตอร์ตัวนี้ในขณะนี้มีแค่ตัวเดียวคือ

java ซึ่งเป็นค่าปกติของมันอยู่แล้ว ไดเร็กทีฟนี้จึงอาจจะละไว้ไม่ใส่ก็ได้ ลักษณะการเขียนก็เป็นดังต่อไปนี้

```
<%@ page language="java" %>
```

contentType

เวลาเบราเซอร์รับข้อมูลจากเวบเซิร์ฟเวอร์มาแสดงผล ข้อมูลที่เวบเซิร์ฟเวอร์ส่งมามีลักษณะเป็นไฟล์ เช่น ไฟล์ .html ไฟล์ .gif ไฟล์ .class เป็นต้น เบราเซอร์ต้องอาศัยพารามิเตอร์ contentType ในการแยกแยะว่าไฟล์ที่ได้รับมาใช้ทำอะไร มีอะไรอยู่ในนั้น เราสามารถมาบอกเบราเซอร์ของผู้เยี่ยมชมได้ว่าไฟล์ .jsp ที่ส่งมาให้เป็นคำสั่ง HTML ล้วนๆ ด้วยการใช้พารามิเตอร์ contentType ดังนี้

```
<%@ page contentType="text/html" %>
```

อย่างไรก็ตาม พารามิเตอร์นี้ก็ไม่จำเป็นอีกเช่นกัน เพราะปกติเบราเซอร์จะคิดว่าเป็นไฟล์ HTML อยู่แล้ว กรณีที่จะใช้พารามิเตอร์ตัวนี้จริงๆ ก็คือ กรณีที่มีภาษาไทยอยู่ในหน้าเวบของเรา เราต้องการบอกให้เบราเซอร์เข้ารหัสภาษาให้ถูกต้อง เราทำได้โดยการใส่ไดเร็กทีฟดังนี้

```
<%@ page contentType="text/html;char-set=windows-874" %>
```

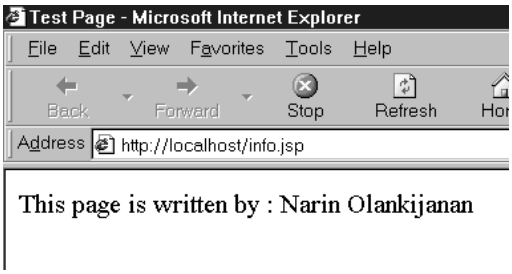
windows-874 เป็นการเข้ารหัสมาตรฐานของภาษาไทยบนอินเตอร์เนตเอ็กซ์พลอเลอร์ ถ้าเราใส่ไดเร็กทีฟนี้ไว้ในเพจที่มีภาษาไทย ผู้เยี่ยมชมจะสามารถอ่านภาษาไทยได้เลยไม่จำเป็นต้องเสียเวลาในการเปลี่ยน encoding ให้เป็นภาษาไทยด้วยตนเอง

info

พารามิเตอร์ตัวต่อไปคือ info ซึ่งมีไว้เก็บข้อความอะไรก็ได้ที่ผู้เขียนต้องการสื่อให้กับเบราเซอร์ เบราเซอร์สามารถเรียกข้อความใน info ออกมาแสดงผลได้ด้วยคำสั่ง (HttpJspPage)page.getServletInfo() เช่น ถ้าต้องการระบุชื่อผู้พัฒนาเวบก็ทำได้ดังตัวอย่างต่อไปนี้

โปรแกรมที่ 4-3 info.jsp

```
<%@ page info="Narin Olankijanan" %>
<html>
<title> Test Page </title>
<body>
This page is written by : <%= ((HttpJspPage)page).getServletInfo() %>
</body>
</html>
```



buffer

พารามิเตอร์ buffer มีไว้สำหรับกำหนดขนาดแรมชั่วคราวที่เวบเซิร์ฟเวอร์ใช้เก็บผลลัพธ์ที่ได้จากเจเอสพีคอนเทนเนอร์เวลารันเพจนั้นๆ ค่าปกติของมันคือ 8kb ซึ่งเพียงพอสำหรับเวบเพจทั่วไป ในกรณีที่เราต้องการประหยัดแรมเราสามารถลดขนาดของบัฟเฟอร์ต่อเพจลง และในทางตรงกันข้ามถ้าเรามีเวบเพจบางเพจที่มีความยาวของเนื้อหามากกว่าที่บัฟเฟอร์ขนาด 8kb จะรับได้ เราต้องขยายขนาดเพื่อให้เนื้อหาในเพจถูกแสดงอย่างครบถ้วน ดังตัวอย่างต่อไปนี้

```
<%@ page buffer="32kb" %>
```

ตัวอย่างข้างต้นขยายขนาดของบัฟเฟอร์เป็น 32 กิโลไบต์ต่อเพจ

autoFlush

พารามิเตอร์ autoFlush ใช้บอกว่าให้เวบเซิร์ฟเวอร์ทำอย่างไรเมื่อแรมที่ใช้เป็นบัฟเฟอร์เต็ม ค่าปกติของ autoFlush คือ “true” ซึ่งเป็นการบอกให้ลบข้อมูลเก่าๆ ในแรมทิ้งเพื่อให้บัฟ

เฟอร์สามารถมีเนื้อที่ใหม่ๆ สำหรับทำงานต่อไปได้ แต่ถ้าเราใส่ข้อความต่อไปนี้ลงในไฟล์ .jsp ของเรา

```
<%@ page autoFlush="false" %>
```

เมื่อใช้งานเวบเซิร์ฟเวอร์ไปนานๆ จนเนื้อที่ในบัพเฟอร์เต็ม การเข้าถึงเวบเพจจากผู้เยี่ยมชมรายต่อไปจะทำให้เกิด error ขึ้น

โดยปกติก็ไม่มีเหตุผลอันใดที่คุณจะเซตพารามิเตอร์ autoFlush ให้เป็น false

extends

โดยปกติแล้วคลาสที่เกิดจากไฟล์ .jsp จะต้องสืบทอดคลาส HttpJspBase ถ้าเราต้องการให้มันสืบทอดคลาสที่เราสร้างขึ้นเองเราสามารถใส่ชื่อของคลาสแม่เป็นค่าของพารามิเตอร์ extends ทั้งนี้คลาสที่เราสร้างขึ้นเองต้องสืบทอดคลาส HttpJspBase มาก่อนและต้องมีแมธธอสนามธรรมชื่อ _jspService() อยู่ นอกจากนี้เราต้องคอมไพล์คลาสที่เราสร้างขึ้นเองให้เป็นไฟล์ .class และเก็บไว้ในโฟลเดอร์ WEB-INF ใต้โฟลเดอร์ ROOT

ในที่นี้ไม่ขอยกตัวอย่างการสืบทอดคลาสส่วนตัวเพราะไม่เป็นที่นิยมใช้ และอาจเกิดความผิดพลาดได้ง่าย

isThreadSafe

โดยปกติไฟล์ .jsp สามารถถูกเข้าถึงได้โดยผู้เยี่ยมชมมากกว่าหนึ่งคนในเวลาเดียวกัน เพราะ Tomcat สนับสนุนการทำงานแบบมัลติเทรด มัลติทาส์กกิ้ง แต่ในบางกรณีเราต้องการกันไม่ให้มีการเข้าถึงเพจหน้าเดียวกันจากผู้ใช้มากกว่าหนึ่งคนในเสียววินาทีเดียวกันเช่น เพจที่ทำงานเกี่ยวกับระบบการเงินของธนาคารหรือระบบจองตั๋วเครื่องบินที่จะมั่วไม่ได้เป็นอันขาด เราใช้พารามิเตอร์ isThreadSafe ในการสั่งให้ Tomcat ล็อกเพจนี้ไว้ให้เข้าถึงได้ทีละหนึ่งคนด้วยการกำหนดค่าให้เป็น false ดังนี้

```
<%@ page isThreadSafe="false" %>
```

ความเร็วในการตอบสนองผู้เยี่ยมชมอาจตกลงบ้างถ้าเวบไซต์ของคุณมีผู้เยี่ยมชมจำนวนมาก แต่โดยทั่วไปแล้วแทบสังเกตไม่เห็นความเปลี่ยนแปลง สิ่งที่ได้มาคือความปลอดภัยของระบบฐานข้อมูล

session

เวบเซิร์ฟเวอร์ที่สนับสนุนเจเอสพี มีการจดจำเบราเซอร์ที่เข้าถึงเวบเพจของมันด้วย เราเรียกการติดต่อมาจากเบราเซอร์ตัวเดียวกันว่า session ไม่ว่าเบราเซอร์นั้นจะเข้าถึงเพจกี่เพจ ตราบใดที่เบราเซอร์นั้นยังไม่หยุดการทำงานเวบเซิร์ฟเวอร์จะจดจำได้ว่าเป็นการเข้าถึงจากเบราเซอร์ตัวเดิม การจดจำ session ทำให้เวบเซิร์ฟเวอร์ต้องทำงานมากขึ้นกว่าปกติ เพราะต้องคอยติดตามตรวจสอบเบราเซอร์ เราสามารถบอกให้ Tomcat ไม่ต้องตรวจสอบ session ได้ด้วยการสั่ง

```
<%@ page session="false" %>
```

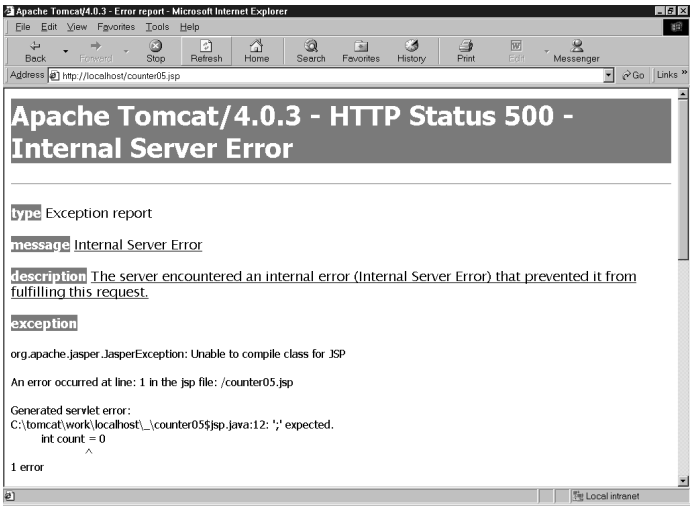
errorPage และ isErrorPage

ถ้าคุณเขียนคำสั่งเจเอสพีไม่ถูกต้อง เวบเซิร์ฟเวอร์จะฟ้องความผิดพลาดออกมาบนเบราเซอร์แทนที่จะแสดงผล ตัวอย่างเช่น โปรแกรมข้างล่างนี้เหมือนโปรแกรมที่ 3-1 ของเราทุกประการแต่ลืมใส่เครื่องหมาย ; ตามหลังคำสั่ง int count = 0

โปรแกรมที่ 4-4 `counter02.jsp`

```
<%! int count = 0 %>
<html>
<title>My First JSP</title>
<body>
You are the visitor number <%= ++count %>.
</body>
</html>
```

เบราเซอร์จะฟ้องความผิดพลาดดังนี้



บางครั้งเราต้องการซ่อนความผิดพลาดไว้ไม่ให้ผู้เยี่ยมชมเวบไซต์ของคุณได้เห็นหากเวบไซต์ของคุณมีบั๊กโดยไม่ได้เจตนา วิธีการหนึ่งก็คือการพาผู้เยี่ยมชมที่เยี่ยมชมเวบหน้าที่บั๊กไม่ว่าจะเป็นหน้าใดในเวบไซต์ของคุณไปยังเวบหน้าพิเศษหน้าหนึ่งซึ่งแจ้งให้ผู้เยี่ยมชมทราบว่าเกิดความผิดพลาดบางประการ เราใช้พารามิเตอร์ errorPage ในการบอกให้เวบเซิร์ฟเวอร์ส่งผู้เยี่ยมชมไปที่เพจหน้าพิเศษ และใช้พารามิเตอร์ isErrorPage ในการระบุให้เพจหน้าพิเศษนั้นเป็นเพจสำหรับรองรับความผิดพลาดจากเพจอื่น ทำได้ดังนี้

โปรแกรมที่ 4-5 `counter03.jsp`

```
<%@ page errorPage="error.jsp" %>
<%! int count = 0 %>
<html>
<title>My First JSP</title>
<body>
You are the visitor number <%= ++count %>.
</body>
</html>
```

โปรแกรมข้างต้นเหมือนกับโปรแกรมที่ 4-4 แต่มีการระบุว่าให้ส่งผู้เยี่ยมชมไปที่ไหนถ้าเพจมีบั๊ก

โปรแกรมที่ 4-6 `error.jsp`

```
<%@ page isErrorPage="true" %>
<html>
<title>Error Page</title>
<body>
Sorry, sir. Something is wrong.
</body>
</html>
```

โปรแกรมข้างต้นคือเพจสำหรับรองรับความผิดพลาดจากเพจอื่นๆ ที่นี้ลองเข้าถึงเพจ counter06.jsp ดูจะพบว่าเบราเซอร์จะถูกส่งไปยัง error.jsp แทน

5

วัตถุแฝง

จาวาเป็นภาษาเชิงวัตถุ การจะใช้งานแมธธอสจะต้องเริ่มจากการประกาศวัตถุของคลาสที่มีแมธธอสนั้นอยู่ขึ้นมาก่อน แล้วจึงเรียกใช้แมธธอสนั้นผ่านทางวัตถุที่เราประกาศขึ้น

สำหรับเจเอสพี คลาสบางครั้งมีความจำเป็นที่จะต้องใช้งานบ่อยมาก ดังนั้นเจเอสพีจึงลดขั้นตอนลงด้วยการสร้างวัตถุของคลาสเหล่านั้นขึ้นมาให้เราเรียกใช้แมธธอสได้ทันทีโดยไม่ต้องเขียนคำสั่งประกาศวัตถุเหล่านั้นก่อน เราเรียกวัตถุพวกนี้ว่า **วัตถุแฝง**

ในบทนี้เราจะมาเรียนรู้ว่าวัตถุแฝงมีอะไรบ้าง แต่ก่อนอื่นขออธิบายความหมายของคำว่า scope ซึ่งจำเป็นต่อการทำเข้าใจเรื่องวัตถุแฝงก่อน

scope

scope หมายถึงช่วงชีวิตของวัตถุมีสี่ระดับ ได้แก่

ตารางที่ 5-1 scope

ระดับ	ความหมาย
application	เกิดขึ้นและคงอยู่ตลอดไปตราบเท่าที่เวบเซิร์ฟเวอร์ยังทำงานอยู่
session	เกิดขึ้นเมื่อผู้เยี่ยมชมใช้เบราเซอร์ติดต่อเข้ามาในเวบไซต์ และคงอยู่ตราบเท่าที่เบราเซอร์ของผู้เยี่ยมชมยังไม่หยุดทำงาน
request	เกิดขึ้นเมื่อผู้เยี่ยมชมเข้าถึงเพจหน้าหนึ่งๆ และจบลงทันทีเมื่อเบราเซอร์แสดงผลของเพจนั้นออกหน้าจอเรียบร้อยแล้ว ถ้าเพจนั้นๆ ประกอบด้วยไฟล์ .jsp หลายๆ ไฟล์ (มีการใช้ไดเร็กทีฟ include) จะถือว่าไฟล์ .jsp ทุกไฟล์ที่ประกอบขึ้นเป็นเพจนั้นอยู่ใน request เดียวกัน

page	เกิดขึ้นเมื่อผู้เยี่ยมชมเข้าถึงเพจหน้าหนึ่งๆ และจบลงทันทีเมื่อเบราเซอร์แสดงผลของเพจนั้นออกหน้าจอเรียบร้อยแล้ว ถ้าเพจนั้นๆ ประกอบด้วยไฟล์ .jsp หลายๆ ไฟล์ (มีการใช้ไดเร็กทีฟ include) จะถือว่าไฟล์ .jsp แต่ละไฟล์ที่ประกอบขึ้นเป็นเพจนั้นเป็นคนละ page

เวลาเราสร้างวัตถุขึ้นมาด้วยคำสั่งในบล็อกประกาศตัวแปร วัตถุนั้นจะมี scope อยู่ในระดับ application กล่าวคือ มันจะคงอยู่ตราบเท่าที่เวบเซิร์ฟเวอร์ยังทำงานอยู่ การเยี่ยมชมเพจหน้านั้นๆ แต่ละครั้งไม่ทำให้เกิดวัตถุตัวใหม่ แต่จะใช้วัตถุตัวเดิมไปตลอด คล้ายกับตอนที่เราประกาศตัวแปร count ในบล็อกประกาศตัวแปรสำหรับนับจำนวนผู้เยี่ยมชม

ส่วนถ้าเราสร้างวัตถุในบล็อกสคริปเลต วัตถุนั้นจะมี scope อยู่ในระดับ page กล่าวคือจะสร้างใหม่ทุกครั้งที่มีผู้เยี่ยมชมเข้าถึงเพจ และจะหายไปทันทีที่การเยี่ยมชมสิ้นสุดลง คล้ายกับตัวแปรทั้งหลายที่ประกาศในบล็อกสคริปเลต

จะเห็นได้ว่าวัตถุที่เราประกาศเองจะมี scope อยู่แค่สองแบบเท่านั้นคือ application และ page แต่วัตถุแฝงบางตัวจะมี scope เป็นแบบอื่นด้วย

request

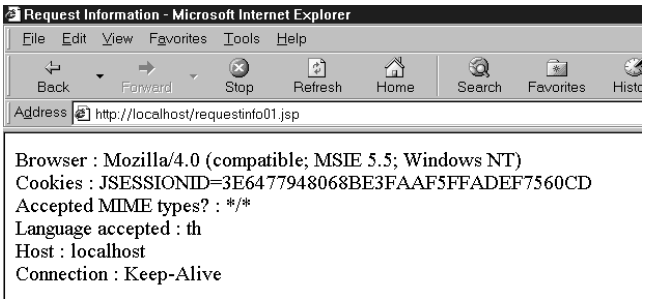
วัตถุแฝงตัวแรกที่เราจะทำความรู้จักคือ request ซึ่งเป็นวัตถุแฝงของคลาส HttpServletRequest วัตถุแฝงที่ชื่อ request นี้มี scope เป็นแบบ request ด้วยกล่าวคือ ในการเยี่ยมชมแต่ละครั้งวัตถุแฝง request จะถูกสร้างขึ้นและจะตายไปทันทีที่การเยี่ยมชมสิ้นสุดลง

วัตถุแฝง request มีแมธธอสดีๆ ที่เป็นประโยชน์จำนวนมาก แมธธอสตัวแรกคือ getHeader() ซึ่งใช้สืบค้นค่าพารามิเตอร์ต่างๆ ของเบราเซอร์ของผู้เยี่ยมชม ตัวอย่างต่อไปนี้เป็นตัวอย่างการเรียกใช้แมธธอส getHeader()

โปรแกรมที่ 5-1 `requestinfo01.jsp`

```
<html>
<title>Request Information</title>
```

```
<body>
Browser : <%= request.getHeader("User-Agent") %> <br>
Cookies : <%= request.getHeader("Cookie") %> <br>
Accepted MIME types? : <%= request.getHeader("Accept") %> <br>
Language accepted : <%= request.getHeader("Accept-Language") %> <br>
Host : <%= request.getHeader("Host") %> <br>
Connection : <%= request.getHeader("Connection") %> <br>
</body>
</html>
```



เพจนี้จะแสดงข้อมูลต่างๆ ของการร้องขอของเบราเซอร์ เช่น ชนิดของเบราเซอร์ ภาษาปกติที่เบราเซอร์แปลผล เป็นต้น ข้อมูลเหล่านี้เป็นข้อมูลซึ่งปกติจะส่งมากับเนื้อความของเพจ โดยจะอยู่ที่ส่วนหัวของไฟล์ เราเรียกข้อมูลกลุ่มนี้ว่า Header ดังนั้นเราใช้แมธธอสของวัตถุ request ที่ชื่อ getHeader() ในการเรียกข้อมูลแต่ละตัวออกมา จะเห็นได้ว่าเราสามารถเรียกใช้วัตถุแฝง request นี้ได้ทันทีที่เมื่อไรก็ได้ราวกับว่ามีใครสร้างมันเอาไว้ให้เรา

พารามิเตอร์ที่อยู่ในวงเล็บของแมธธอส getHeader() เป็นชื่อของชนิดของข้อมูลที่ต้องการจะสืบค้น ซึ่งมีดังนี้

ตารางที่ 5-2 พารามิเตอร์ของแมธธอส getHeader() ในคลาส HttpServletRequest

ชื่อ	ความหมาย
Cookie	เลขประจำตัวของ session
From	อีเมล์แอดเดรสของผู้ร้องขอ
Accept	ชนิดของไฟล์ที่เบราเซอร์ยอมรับ */* หมายถึงอะไรก็ได้
Accept-Charset	ประเภทกลุ่มตัวอักษรที่เบราเซอร์ยอมรับ
Accept-Encoding	การเข้ารหัสปกติของเบราเซอร์
Authorization	ข้อมูลที่เป็นความลับที่ส่งมาเช่น พาสเวิร์ด
User-Agent	ชนิดของเบราเซอร์ ส่วนมากมีค่าเท่ากับ Mozilla เพราะเป็นเบราเซอร์ต้นแบบของโลก
Accept-Language	ภาษาปกติที่เบราเซอร์ใช้

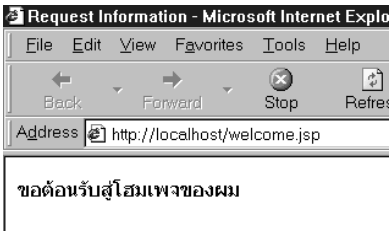
Referer	URL ของเพจที่มาก่อน
Charge-To	ข้อมูลทางบัญชี
If-Modified-Since	เป็นการบอกว่าข้อมูลที่ค้างอยู่ในแคชของเบราเซอร์ทันสมัยหรือไม่
Pragma	คำสั่งพิเศษที่บอกเซิร์ฟเวอร์ ที่สำคัญคือ no-cache ที่สั่งห้าม proxy server แคชข้อมูล
Host	ชื่อโฮสต์ของเซิร์ฟเวอร์
Connection	คำสั่งที่บอกให้เซิร์ฟเวอร์รักษาสภาพการติดต่อกับเบราเซอร์ไว้
Content-Length	จำนวนไบต์ของไฟล์ที่ส่งมา

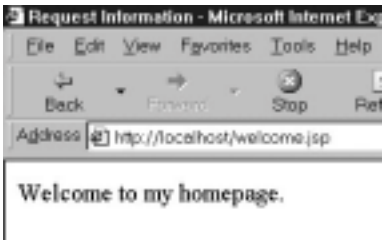
Accept-Language เป็นค่าปกติของภาษาที่ผู้เยี่ยมชมตั้งเบราเซอร์ของตนไว้ เราสามารถนำข้อมูลตรงนี้มาทำให้เวบไซต์ของเราดูฉลาดขึ้นได้ด้วยการเปลี่ยนภาษาที่ทักทายผู้เยี่ยมชมไปตามค่าของ Accepted-Language ตัวอย่างเช่น

โปรแกรมที่ 5-2 `welcome.jsp`

```
<%@ page contentType="text/html ; char-set=windows-874" %>
<% String lang = request.getHeader("Accept-Language") ; %>
<html>
<title>Request Information</title>
<body>
<% if (lang.equals("th")) { %>
ขอต้อนรับสู่โฮมเพจของผม
<% } else { %>
Welcome to my homepage.
<% } %>
</body>
</html>
```

โฮมเพจ welcome.jsp ของคุณสามารถเลือกทักทายผู้เยี่ยมชมด้วยภาษาที่ถูกต้องด้วยการตรวจสอบจากค่าของ Accept-Language หากมีค่าเท่ากับ th ก็จะทักทายผู้เยี่ยมชมเป็นภาษาไทย แต่ถ้าไม่ก็จะทักทายเป็นภาษาอังกฤษ





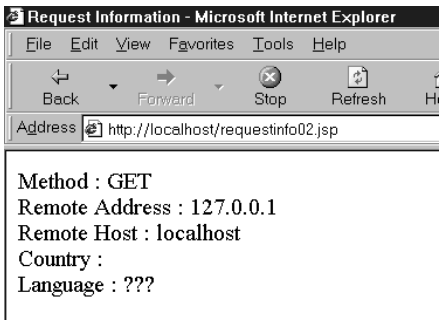
NOTE

ถ้าลองแล้วเกิด error ขึ้น ให้ตรวจสอบดูใน Internet Options ของเบราเซอร์ของคุณว่าคุณได้เลือก Language ปกติไว้หรือไม่

นอกจากแมธธอส getHeader() แล้ว request ยังมีแมธธอสที่น่าสนใจตัวอื่นอีก เช่น

โปรแกรมที่ 5-3 `requestinfo02.jsp`

```
<html>
<title>Request Information</title>
<body>
Method : <%= request.getMethod() %> <br>
Remote Address : <%= request.getRemoteAddr() %> <br>
Remote Host : <%= request.getRemoteHost() %> <br>
Country : <%= request.getLocale().getDisplayCountry() %> <br>
Language : <%= request.getLocale().getDisplayLanguage() %> <br>
</body>
</html>
```



การตอบสนองแบบฟอร์มของผู้ใช้

ประโยชน์ที่สำคัญอย่างยิ่งของวัตถุแฝง request คือการตอบสนองแบบฟอร์มของผู้ใช้ที่สร้างจากคำสั่ง <form> ของ HTML ตัวภาษา HTML เองตอบสนองไม่ได้เพราะการตอบสนองฟอร์มต้องทำบนฝั่งเซิร์ฟเวอร์ วัตถุแฝง request มีแมธธอสชื่อ getParameter() เป็นอุปกรณ์สำคัญในการตอบสนองแบบฟอร์ม

ลองดูตัวอย่างการตอบสนองแบบฟอร์มของผู้ใช้ด้วยการสร้างแบบฟอร์มที่ใช้แทนปฏิทินร้อยปี ด้วยการให้ผู้เยี่ยมชมกรอกวันเดือนปีเกิดแล้วเครื่องจะแสดงวันของสัปดาห์ และปีนักษัตรของผู้เยี่ยมชม

เริ่มต้นด้วยการสร้างไฟล์ .html ง่ายๆ ซึ่งรับชื่อและวันเดือนปีเกิด

โปรแกรมที่ 5-4 `100yearcalendar.html`

```
<html>
<title>100-year Caledar</title>
<body>
<form method=”post” action=”result.jsp” >
ชื่อ <input type=”text” name=”name”><br>
วันที่เกิด
<input type=”text” name=”date” size=”2” maxlength=”2”>
 เดือน
<select name="month">
          <option value="1" >มกราคม</option>
          <option value="2" >กุมภาพันธ์</option>
          <option value="3" >มีนาคม</option>
          <option value="4" >เมษายน</option>
          <option value="5" >พฤษภาคม</option>
          <option value="6" >มิถุนายน</option>
          <option value="7" >กรกฎาคม</option>
          <option value="8" >สิงหาคม</option>
          <option value="9" >กันยายน</option>
          <option value="10" >ตุลาคม</option>
          <option value="11" >พฤศจิกายน</option>
          <option value="12" >ธันวาคม</option>
</select>
 ปี พ.ศ.
<input type=”text” name=”year” size=”4” maxlength=”4”><br>
<input type=”submit”>
```

```
</form>
</body>
</html>
```

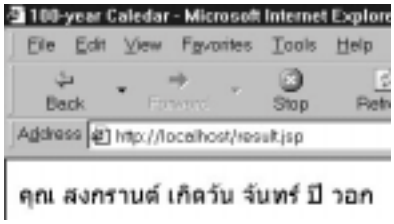
คราวนี้ก็สร้างไฟล์ .jsp ชื่อ result.jsp ขึ้นมารับมือ ดังนี้

โปรแกรมที่ 5-5 result.jsp

```
<%@ page import="java.util.Date" %>
<%@ page import="java.text.SimpleDateFormat" %>

<%! String[] zodiac = {“วอก","ระกา","จอ","กุน","ชวด","ฉลู",
      "ขาล","เถาะ","มะโรง","มะเส็ง","มะเมีย","มะแม"};
    String[] dayOfWeek ={"","อาทิตย์","จันทร์","อังคาร","พุธ",
      "พฤหัสบดี","ศุกร์","เสาร์"};
%>

<%
String name = request.getParameter("name");
String date = request.getParameter("date");
String month = request.getParameter("month");
String thaiyear = request.getParameter("year");
int year = Integer.parseInt(thaiyear)-543;
int y = year%12;

SimpleDateFormat formatter = new SimpleDateFormat("d-M-yyyy");
Date birthdate = formatter.parse(date+"-"+month+"-"+year);
formatter = new SimpleDateFormat("F");
int day = Integer.parseInt(formatter.format(birthdate));

%>
<html>
<title>100-year Caledar</title>
<body>
คุณ <%=name%> เกิดวัน <%=dayOfWeek[day]%> ปี <%=zodiac[y]%>
</body>
```

```
</html>
```

พารามิเตอร์ของแมธธอส getParameter() คือ ชื่อของข้อมูลที่ส่งมาจากฟอร์มซึ่งตรงกับค่าของพารามิเตอร์ name ของฟอร์มแต่ละตัวนั่นเอง เมื่อเรารับข้อมูลอันได้แก่ ชื่อ และวัน เดือน ปีมาแล้ว เราก็นำมาคำนวณหาวันของสัปดาห์ และปีนักษัตร ผลลัพธ์ที่ได้เป็นดังนี้



NOTE

ปูพื้นกันหน่อยสำหรับคนที่มีพื้นฐานเรื่องฟอร์มของ HTML ไม่แน่น ฟอร์มที่มีไว้ให้ผู้เยี่ยมชมมีหลายรูปแบบ เช่น ตัวเลือก เมนู ปุ่ม ฯลฯ แบบพื้นฐานที่สุดก็คือมีช่องว่างให้เติมตัวอักษรลงไป ตัวอย่างเช่น

<input type="text" name="address">

เป็นคำสั่งสร้างช่องกรอกข้อความ (textbox) ให้ชื่อเรียกว่า address ชื่อเรียกเป็นสิ่งที่เอาไว้ให้เวบเซิร์ฟเวอร์อ้างอิง ฟอร์มทุกรูปแบบต้องตั้งชื่อเสมอด้วยการใส่พารามิเตอร์ name

คำสั่งในการสร้างฟอร์มทุกตัวจะต้องอยู่ภายใต้แท็ก <form> เพื่อบ่งบอกว่าตัวเลือกเหล่านั้นอยู่ในฟอร์มเดียวกัน พารามิเตอร์ที่สำคัญของแท็ก <form> มีอยู่สองตัวได้แก่ action และ method พารามิเตอร์ action มีไว้บอกชื่อของไฟล์ .jsp บนเซิร์ฟเวอร์ที่ทำหน้าที่รับมือกับฟอร์ม ส่วนพารามิเตอร์ method ใช้กำหนดวิธีการส่งข้อมูลในฟอร์มไปให้เวบเซิร์ฟเวอร์มีสองวิธีได้แก่

1. POST เป็นวิธีการส่งข้อมูลของฟอร์มไปยังเวบเซิร์ฟเวอร์แบบครั้งละมากๆ เหมาะกับฟอร์มยาวๆ ข้อมูลที่ส่งไปจะมองไม่เห็น

2. GET เป็นวิธีส่งแบบที่มองเห็นข้อมูลได้จาก URL เช่น ในตัวอย่างข้างต้นหากเปลี่ยนไปใช้วิธีส่งแบบ GET เวลาส่งข้อมูลจะเห็น URL ของไฟล์ปลายทางเป็น

http://localhost/result.jsp?name=สงกรานต์

&date=13&month=4&year=2475

กล่าวคือข้อมูลจะถูกแสดงหลังชื่อ URL โดยคั่นด้วยเครื่องหมายคำถาม ถ้ามีข้อมูลหลายตัวแต่ละตัวจะคั่นด้วยเครื่องหมาย & การส่งข้อมูลแบบ GET เป็นวิธีปกติของเบราเซอร์

response

response เป็นวัตถุแฝงของคลาส HttpServletResponse ใช้จัดการเกี่ยวกับการตอบสนองการร้องขอไฟล์ .jsp นั้นๆ มี scope เป็นแบบ request และมีแมธธอสที่น่าสนใจดังนี้

sendRedirect()

ใช้ส่งผู้เยี่ยมชมไปยังเพจอื่นทันที เช่น ต้องการส่งผู้เยี่ยมชมจากเพจปัจจุบันไปยัง http://www.yahoo.com ทำได้โดยใช้คำสั่ง

```
<% response.sendRedirect(http://www.yahoo.com); %>
```

sendError()

ใช้บอกให้เบราเซอร์แสดง Error เช่น

```
<% response.sendError(500); %>
```

ใช้เรียก Error 500

NOTE

Error code เป็นเลขบอกความผิดพลาดที่เกิดขึ้นในการติดต่อกับเวบเซิร์ฟเวอร์ ตามมาตรฐาน HTTP 1.1 มีอยู่อย่างมากมายหลายเลขหมาย แบ่งออกเป็นหมวดๆ ได้ดังนี้

100-199 เป็นเพียงการเตือนว่าเบราเซอร์ควรติดต่อเข้ามาด้วยวิธีการอื่น
200-299 เป็นการแจ้งว่าการร้องขอเป็นผล
300-399 เป็นการแจ้งเกี่ยวกับเพจที่ได้ย้ายไปแล้ว
400-499 เป็นการแจ้งว่าความผิดพลาดเกิดจากฝั่งเบราเซอร์
500-599 เป็นการแจ้งว่าความผิดพลาดเกิดจากฝั่งเซิร์ฟเวอร์

session

session เป็นวัตถุแฝงของคลาส HttpSession มี scope เป็นแบบ session กล่าวคือมันจะถูกสร้างขึ้นเมื่อผู้เยี่ยมชมใช้เบราเซอร์คลิกเข้ามาในเวบไซต์เป็นครั้งแรก หนึ่งตัวต่อหนึ่งผู้เยี่ยมชม และจะอยู่ตราบเท่าที่เบราเซอร์ยังไม่จบการทำงาน วัตถุแฝง session ช่วยทำให้เวบเซิร์ฟเวอร์จำผู้เยี่ยมชมได้เพราะทุกครั้งที่เวบเซิร์ฟเวอร์สร้าง session มันจะกำหนด sessionID เป็นหมายเลขกำกับ session ขึ้นมา เวบเซิร์ฟเวอร์จะอาศัย sessionID ในการจดจำว่าการร้องขอที่ส่งเข้ามาเป็นการร้องขอที่มาจากเบราเซอร์ตัวใด ซึ่งทำให้เวบไซต์ของเราติดตามพฤติกรรมของผู้เยี่ยมชมแต่ละรายได้ตลอดเวลาที่ผู้เยี่ยมชมยังอยู่ภายในเวบไซต์

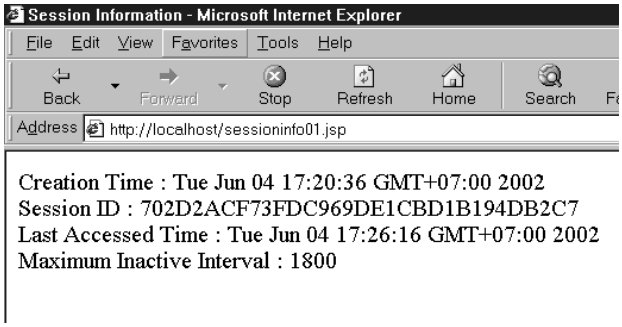
NOTE

ถ้าผู้เยี่ยมชมคนเดิมเปิดหน้าจออินเตอร์เนตเอ็กซ์พลอเรอร์หลายหน้าจอพร้อมกัน จะถือว่าเป็น session เดียวกัน ทุกหน้าจอจะใช้วัตถุแฝง session ตัวเดียวกัน มี sessionID ตัวเดียวกัน แต่ถ้าเปิดทั้งอินเตอร์เนตเอ็กซ์พลอเรอร์ และเบราเซอร์ตัวอื่นเช่น เนสท์เคป บนคอมพิวเตอร์เครื่องเดียวกันจะถือว่าเป็นคนละ session แม้ว่าจะมาจากคอมพิวเตอร์เครื่องเดียวกันก็ตาม

ลองมาดูว่าวัตถุแฝงที่ชื่อ session มีอะไรให้เล่นบ้าง

โปรแกรมที่ 5-6 `sessioninfo01.jsp`

```
<html>
<title>Session Information</title>
<body>
Creation Time : <%= new java.util.Date(session.getCreationTime()) %>
   <br>
Session ID : <%= session.getId() %> <br>
Last Accessed Time : <%= new
   java.util.Date(session.getLastAccessedTime()) %> <br>
Maximum Inactive Interval : <%= session.getMaxInactiveInterval()  %>
   <br>
</body>
</html>
```



Session ID คือเลขทะเบียนประจำ session ซึ่งเวบเซิร์ฟเวอร์อาศัยในการแยกแยะว่าคำร้องขอหนึ่งๆ มาจากเบราเซอร์ตัวใด

Creation Time คือเวลาที่ session นี้ถูกสร้างขึ้นหรือก็คือเวลาที่ผู้เยี่ยมชมเข้าเยี่ยมชมเวบไซต์แห่งนี้เป็นครั้งแรก ส่วน Last Accessed Time คือเวลาที่ผู้เยี่ยมชมเข้าถึงเพจล่าสุด

Maximum Inactive Interval คือ เวลาเป็นวินาทีที่ session จะหมดอายุ ค่าปกติคือ 1800 วินาทีหรือ 30 นาที เหตุที่ต้องกำหนดเวลาหมดอายุของ session เป็นเพราะถ้าผู้เยี่ยมชมเปิดเบราเซอร์ทิ้งไว้เป็นเวลานานมากๆ ข้อมูลของ session อาจล้าสมัยไปแล้ว ถ้าผู้เยี่ยมชมกลับมาใช้เบราเซอร์อยู่อีกเวบเซิร์ฟเวอร์ควรถือกำหนด session ID ใหม่ให้จะดีกว่า เราสามารถตั้งค่าเวลาหมดอายุตามใจเราได้ด้วยคำสั่ง

```
<% session.setMaxInactiveInterval(24*60*60); %>
```

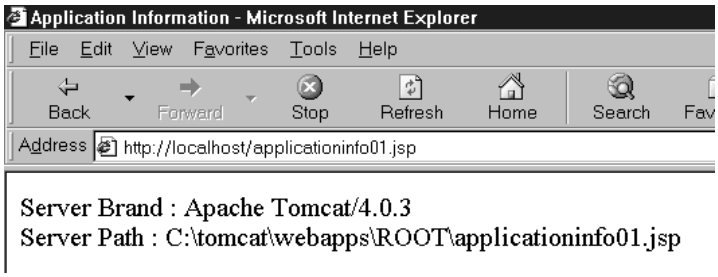
คำสั่งนี้เป็นการตั้งค่าเวลาหมดอายุให้เป็น 24 ชั่วโมงหรือหนึ่งวัน เป็นต้น

application

application เป็นวัตถุแฝงที่จัดการเกี่ยวกับค่าต่างๆ ของเวบเซิร์ฟเวอร์มี scope เป็นแบบ application มีแมธธอสที่น่าสนใจดังนี้

โปรแกรมที่ 5-7 applicationinfo01.jsp

```
<html>
<title>Application Information</title>
<body>
Server Brand : <%= application.getServerInfo() %> <br>
Server Path : <%= application.getRealPath(request.getServletPath()) %>
   <br>
</body>
</html>
```



จากภาพที่เห็น ชนิดของเวบเซิร์ฟเวอร์นี้คือ Tomcat เวอร์ชั่น 4.0.3 และไฟล์ๆ นี้ถูกเก็บไว้ที่โฟล์เดอร์ C:\tomcat\webapps\ROOT

page

เราเคยเรียกวัตถุแฝง page มาแล้วครั้งหนึ่งในโปรแกรมที่ 4-3 วัตถุแฝงตัวนี้มี scope เป็นแบบ page

out

out เป็นวัตถุแฝงของคลาส Writer หน้าที่ของ out คือการสั่งให้เบราเซอร์แสดงข้อความอะไรก็ได้ทั้งที่คำสั่งนี้อยู่ภายในสคริปเลต ตัวอย่างเช่นโปรแกรมที่ 1-1 (โปรแกรม Hello World) เขียนใหม่โดยใช้คำสั่ง out ได้ดังนี้

โปรแกรมที่ 5-8 `helloworld.jsp`

```
<html>
<body>
<% out.print("<b>Hello World</b>"); %>
</body>
</html>
```

แมธธอส print() เป็นแมธธอสที่สั่งให้เบราเซอร์แสดงข้อความที่อยู่ในวงเล็บออกมา ดังนั้นจึงไม่ต่างอะไรกับการเขียนข้อความนั้นด้วยคำสั่ง HTML ธรรมดา

นอกจากแมธธอส print() แล้ว ยังมีแมธธอส println() ซึ่งให้ผลเหมือนกัน

config

วัตถุแฝง config มีแมธธอสที่น่าสนใจอยู่หนึ่งตัวคือ getServletName() ซึ่งจะแสดงค่าของโฟล์เดอร์ที่เก็บไฟล์ .class ของเวบหน้านั้น

pageContext

pageContext รวบรวมข้อมูลทั้งหมดเอาไว้ในตัวเดียวกัน หมายความว่าคุณสามารถเข้าถึงข้อมูลใดๆ ของ session, page, config, application เข้าถึงได้ด้วย pageContext เพราะมันมีแมธธอสอยู่อย่างมากมายไว้ให้ใช้ แต่การใช้งานย่อมซับซ้อนกว่า เราจะไม่ขอกล่าวถึงในที่นี้

exception

exception เป็นวัตถุแฝงที่เก็บข้อมูลเกี่ยวกับบั๊กต่างๆ ที่เกิดขึ้น

6

แท็กเจเอสพี

ยุคนี้กระแสของ XML มาแรง คำว่า XML ย่อมาจาก Extensible Markup Language เป็นภาษาที่มีโครงสร้างคล้ายๆ กับภาษา HTML คือประกอบด้วยแท็กต่างๆ ปะปนกับภาษามนุษย์ จุดประสงค์หลักของ XML คือใช้เป็นภาษากลางระหว่างเซิร์ฟเวอร์ต่างชนิดต่างยี่ห้อกันให้สามารถรับส่งข้อมูลระหว่างกันได้ โดยไม่ต้องสนใจว่าเซิร์ฟเวอร์เหล่านั้นจะใช้เทคโนโลยีของค่ายไหน

คำสั่งเจเอสพีได้รับอิทธิพลของ XML ด้วยเหมือนกัน กล่าวคือ เจเอสพีมีสิ่งที่เรียกว่า **แท็กเจเอสพี** ซึ่งเป็นคำสั่งเจเอสพีที่อยู่ในรูปของแท็ก XML โดยจะนำหน้าด้วยแท็ก <jsp: และต่อท้ายด้วย /> เพื่อบอกให้ทราบว่าเป็น แท็กเจเอสพี เราสามารถบอกให้เวบเซิร์ฟเวอร์ทำอะไรหลายๆ อย่างได้ด้วยแท็กเจเอสพีเหล่านี้ ในบทนี้เราจะมาดูว่าแท็กเจเอสพีมีอะไรบ้าง

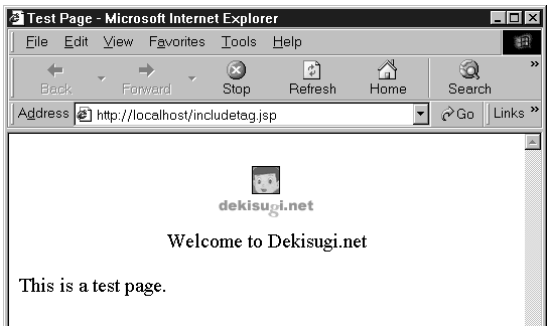
include

คงยังจำได้ว่าถ้าเราต้องการสอดแทรกเนื้อหาในไฟล์อื่นเข้าไปในไฟล์ .jsp ของเรา เราใช้ไดเร็กทีฟ include ข้อเสียของการสอดแทรกเนื้อหาของไฟล์อื่นด้วยวิธีนี้คือ ถ้ามีการแก้ไขไฟล์ต้นทาง เนื้อหาใหม่จะไม่ไปปรากฏอยู่ในไฟล์ปลายทางโดยอัตโนมัติ เพราะการสอดแทรกเกิดขึ้นเมื่อตอนที่มีผู้เยี่ยมชมไฟล์ปลายทางเพียงครั้งเดียวเท่านั้น

เจเอสพีมีแท็กเจเอสพีที่ทำงานได้เหมือนกับไดเร็กทีฟ include แต่จะอัพเดทไฟล์ปลายทางโดยอัตโนมัติเวลาที่มีการแก้ไขไฟล์ต้นทาง แท็กนี้คือ <jsp:include ลองดูรูปแบบของคำสั่งจากตัวอย่างต่อไปนี้

โปรแกรมที่ 6-1 `includetag.jsp`

```
<html>
<title> Test Page </title>
<body>
<jsp:include page="logo.html" flush="true" />
<br>
This is a test page.
</body>
</html>
```



โปรแกรมนี้ให้ผลเหมือนกับโปรแกรมที่ 5-2 แต่ดีกว่าตรงที่เมื่อใดที่เราแก้ไขไฟล์ logo.html หน้าของไฟล์ includetag.jsp จะเปลี่ยนตามโดยอัตโนมัติ

แท็ก <jsp: include ต้องมีพารามิเตอร์ flush="true" อยู่ด้วยเสมอ

forward

แท็ก forward มีไว้สำหรับส่งผู้เยี่ยมชมไปยังเวบหน้าอื่น ตัวอย่างเช่น สมมติว่าเราสร้างไฟล์ชื่อ forward.jsp ขึ้นมา แล้วกำหนดให้เมื่อไรก็ตามที่มีผู้เข้ามาเยี่ยมชมเวบเพจหน้านี้ ให้เวบเซิร์ฟเวอร์ส่งผู้เยี่ยมชมยัง includetag.jsp แทน เราทำได้ดังนี้

โปรแกรมที่ 6-2 forward.jsp

```
<html>
<title> Forward Page </title>
<body>
<jsp:forward page="includetag.jsp" />
<br>
You will be forward to includetag.jsp.
</body>
</html>
```

ถ้าลองเข้าถึงเพจนี้ดูจะพบว่าเบราเซอร์จะกระโดดไปยังเพจ includetag.jsp ในทันที

ใช้แท็กเจเอสพีแทนไดเร็กทีฟ

เราสามารถใช้แท็กเจเอสพีแทนการใช้ไดเร็กทีฟ page หรือ include ได้ด้วย โดยมีรูปแบบดังตัวอย่างต่อไปนี้

```
<jsp:directive.page import="java.util.*" />
```

ตัวอย่างนี้ให้ผลเหมือนกับคำสั่ง

```
<%@ page import="java.util.*" %>
```

ทุกประการ

ถ้าเป็นไดเร็กทีฟ include ก็เพียงแต่ใช้คำว่า include แทน page เช่น

```
<jsp:directive.include file="hello.html" />
```

ให้ผลเหมือนกับคำสั่ง

```
<%@ include file"hello.html" %>
```

ทุกประการ

แท็กอื่น ๆ

แท็กเจเอสพี ยังมีอีกหลายตัว โดยเฉพาะอย่างยิ่ง แท็กที่ใช้กับ **บีน** เราจะได้เรียนรู้แท็กเหล่านั้นไปพร้อมๆ กับการรู้จักกับบีนในบทต่อไป

7

บีน

บีน คือโปรแกรมขนาดเล็กบนเวบเซิร์ฟเวอร์ ซึ่งคอยช่วยเหลือไฟล์ .jsp ด้วยการทำงานบางอย่างแทนให้ โดยที่ไฟล์ .jsp เพียงแต่ร้องขอความช่วยเหลือจากบีนโดยการส่งพารามิเตอร์ที่จำเป็นไปให้ บีนก็จะส่งผลลัพธ์ที่ต้องการกลับไปให้ทันที

ประโยชน์สำคัญของบีนคือการลดความยุ่งเหยิงของไฟล์ .jsp งานที่เคยต้องใช้คำสั่งเจเอสพีเป็นสิบๆ บรรทัด จะลดลงเหลือเพียงคำสั่งร้องขอความช่วยเหลือจากบีนเพียงหนึ่งหรือสองบรรทัดเท่านั้น ทำให้ไฟล์ .jsp มีขนาดเล็กลงและดูง่ายขึ้น การทำเช่นนี้มีประโยชน์มากเพราะปกติแล้วโปรแกรมเมอร์ภาษาจาวาที่พัฒนาคำสั่งเจเอสพี กับนักออกแบบเวบไซต์ที่ใช้คำสั่ง HTML มักเป็นคนละคนกัน การลดคำสั่งเจเอสพีในเวบเพจลงด้วยการย้ายมาเขียนโปรแกรมบนบีนแทน ทำให้การแบ่งงานระหว่างเวบโปรแกรมเมอร์ กับผู้ออกแบบเวบไซต์มีความคล่องตัวมากขึ้น

การสร้างบีนก็เหมือนกับการเขียนโปรแกรมภาษาจาวาปกติ คือ ต้องเขียนเป็นไฟล์นามสกุล .java แล้วนำไปคอมไพล์ให้ได้ไฟล์นามสกุล .class พอจะใช้งานก็นำไฟล์ .class ที่ได้ไปวางไว้บนโฟลเดอร์ WEB-INF ใต้โฟลเดอร์ย่อยที่ชื่อ classes ลองดูตัวอย่างการสร้างบีนอย่างง่ายๆ ที่ใช้งานเป็นตัวนับจำนวนผู้ชมได้เหมือนกับโปรแกรมที่ 3-1 ต่อไปนี้

โปรแกรมที่ 7-1 `Counter.java`

```
package mybean;

public class Counter implements java.io.Serializable {

   private int count = 0;

   public Counter() {
   }

   public int getCount() {
     return ++count;
   }

   public void setCount(int count) {
     this.count = count;
   }
}
```

สร้างไฟล์ Counter.java นี้ขึ้นมาด้วย Notepad จากนั้นเซฟลงบนดิสก์แล้วคอมไพล์ด้วยคำสั่ง

```
javac Counter.java
```

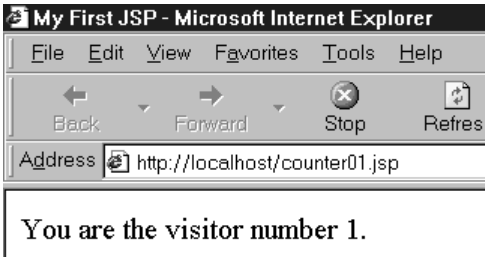
จะได้ไฟล์ Counter.class ขึ้นมา ให้นำไปไว้ในโฟล์เดอร์ C:\tomcat\webapps\root\WEB-INF\classes\mybean (คุณต้องสร้างขึ้นมาเอง) คลาส Counter ที่เราสร้างขึ้นนี้ก็คือ บีน นั่นเอง

ลองสร้างไฟล์ชื่อ counter04.jsp ขึ้นมาเพื่อเรียกใช้ Counter ดังนี้

โปรแกรมที่ 7-2 counter04.jsp

```
<jsp:useBean id="counter" class="mybean.Counter" scope="application" />
<html>
<title>My First JSP</title>
<body>
You are the visitor number <jsp:getProperty name="counter"
   property="count" />.
</body>
</html>
```

เวบหน้านี้ให้ผลเหมือนโปรแกรมที่ 3-1 ทุกประการ



ลองค่อยๆ ดูกันซิว่าเกิดอะไรขึ้น เริ่มจากไฟล์ Counter.java บรรทัดแรก เราประกาศให้บีนตัวนี้อยู่ในแพจเกจชื่อ mybean ซึ่งเป็นชื่อที่ตั้งขึ้นมาเอง และต่อไปตัวอย่างทุกตัวอย่างที่อยู่ในหนังสือเล่มนี้จะกำหนดให้บีนอยู่ในแพจเกจชื่อ mybean เพื่อความเป็นระเบียบ เวลาทำงานจริงๆ ขึ้นจะเขียนบีนให้อยู่ในแพจเกจชื่ออะไรก็ได้ตามใจชอบ

บีนจะต้อง implements อินเตอร์เฟสชื่อ java.io.Serializable เสมอ และต้องประกาศให้เป็นคลาสสาธารณะด้วยคำสั่ง public เพื่อพร้อมให้ไฟล์ .jsp ไหนๆ ในเวบของเราเรียกใช้งานได้ คลาส Counter มีตัวแปรคลาสอยู่หนึ่งตัวเป็นตัวแปรจำนวนเต็มชื่อว่า count ซึ่งกำหนดให้มีค่าเริ่มต้นเท่ากับ 0 ตัวแปรจำนวนเต็มตัวนี้จะใช้เป็นตัวนับจำนวนผู้เยี่ยมชมเวบไซต์นั่นเอง

ตามหลักการเขียนโปรแกรมเชิงวัตถุที่ดี ตัวแปรคลาสควรประกาศให้เป็น private แมธธอสใดๆ ที่อยู่นอกคลาสนี้จะไม่สามารถเข้าถึงตัวแปร count ได้โดยตรง แต่จะเข้าถึงโดยผ่านแมธธอสสาธารณะที่คลาส Counter สร้างไว้ให้ ในที่นี้ได้แก่แมธธอส getCount() และ setCount() ซึ่งใช้อ่านค่าของตัวแปร count และตั้งค่าของตัวแปร count ใหม่ตามลำดับ ปกติแล้วแมธธอสที่เขียนขึ้นมาเพื่อใช้เข้าถึงตัวแปร private จะมีชื่ออย่างไรก็ได้ แต่ถ้าเป็นบีนแล้วเราต้องตั้งชื่อให้ขึ้นต้นด้วยคำว่า get และ set ตามด้วยชื่อของตัวแปร private ตัวนั้นในการอ่าน และการแก้ไขค่าตามลำดับ

โปรดสังเกตว่า แมธธอส getCount() นอกจากจะส่งค่าของตัวแปร count ออกมายังบวกค่าของตัวแปร count ด้วยหนึ่งอีกด้วย

เวลาคอมไพล์ไฟล์ Counter.java จะได้ไฟล์ Counter.class ให้นำไปไว้ใต้โฟล์เดอร์ C:\tomcat\webapps\ROOT\WEB-INF\classes\mybean บีนทุกตัวของเวบไซต์จะต้องอยู่ใต้โฟลเดอร์ WEB-INF\classes นี้ โฟลเดอร์นี้เป็นโฟลเดอร์ที่มองไม่เห็นหากเข้าถึงด้วยเบราเซอร์แต่มีไว้สำหรับเก็บบีนโดยเฉพาะ ในกรณีที่กำหนดให้บีนอยู่ในแพจเกจด้วยต้องมีการสร้างโฟลเดอร์ย่อยที่มีชื่อเหมือนชื่อของแพจเกจ เช่น ถ้ากำหนดให้บีนอยู่ในแพจเกจ org.apache.beans ก็ต้องสร้างโฟลเดอร์ C:\tomcat\webapps\ROOT\WEB-INF\classes\org\apache\beans แล้วนำไฟล์ .class ของบีนตัวนั้นไปไว้ เป็นต้น

เมื่อได้บีนที่พร้อมจะให้เรียกใช้แล้ว เวลาจะเรียกใช้ในไฟล์ .jsp เราใช้แท็กเจเอสพีชื่อ useBean ในการเรียกใช้ดังนี้

```
<jsp:useBean id="counter" class="mybean.Counter" scope="application" />
```

คำสั่งนี้เป็นการประกาศวัตถุของคลาส mybean.Counter และให้ชื่อว่า counter พารามิเตอร์ id คือชื่อของวัตถุซึ่งจริงๆ แล้วจะตั้งชื่ออะไรก็ได้ แต่นิยมตั้งชื่อเหมือนชื่อของบีนแต่ใช้อักษรตัวเล็กขึ้นต้น ส่วนพารามิเตอร์ class คือชื่อเต็มของบีนที่ต้องการเรียกใช้

บีนมี scope เหมือนกับวัตถุแฝง แต่พิเศษกว่าตรงที่เราสามารถเลือก scope ได้เองตามใจชอบ ในกรณีของบีนนับจำนวนผู้เยี่ยมชมต้องมี scope เป็น application เพื่อที่จะได้เหมือนกับตัวแปร count ที่ประกาศในบล็อกประกาศตัวแปร กล่าวคือค่าของมันจะไม่หายไปไหนตลอดการทำงานของเวบเซิร์ฟเวอร์

ในเมื่อวัตถุของบีนที่ชื่อ counter มีแค่ตัวเดียวตลอดทั้งเวบไซต์ ตัวแปรคลาส count จึงมีอยู่แค่ตัวเดียวและมีค่าเริ่มต้นเป็น 0 เมื่อถูกสร้างขึ้น

คำสั่ง

```
<jsp:getProperty name="counter" property="count" />
```

เป็นการเรียกแมธธอสชื่อ getCount() ซึ่งจะบวกค่าของตัวแปร count ด้วยหนึ่ง แล้วคืนค่าของตัวแปร count ออกมาทางเบราเซอร์ พารามิเตอร์ name ใช้บอกชื่อของวัตถุของบีนที่เราต้องการเรียกแมธธอส get ส่วน property ใช้บอกชื่อของตัวแปรที่ต้องการ get ค่า

เนื่องจากตัวแปรคลาส count มีแค่ตัวเดียวทั้งเวบไซต์ ค่าของมันจึงเพิ่มขึ้นทุกครั้งที่มีใครก็ได้เยี่ยมชมเวบหน้านี้ เราจึงใช้มันเป็นตัวนับจำนวนผู้เยี่ยมชมได้

ลองทดลองความแตกต่างระหว่าง scope แต่ละชนิดดูด้วยการแก้ไข scope ของโปรแกรมที่ 7-2 ใหม่ดังนี้

โปรแกรมที่ 7-3 `counter05.jsp`

```
<jsp:useBean id="counter" class="mybean.Counter" scope="session" />
<html>
<title>My First JSP</title>
<body>
You are the visitor number <jsp:getProperty name="counter"
   property="count" />.
</body>
</html>
```

คราวนี้ลองเรียกเพจนี้ดู จะพบว่าค่าของตัวแปร count จะเริ่มที่ 0 ใหม่ แทนที่จะนับต่อจากที่เราเคยเข้าถึงมันไปแล้วก่อนหน้านี้ เมื่อลองรีเฟรชดูก็จะพบว่าค่าของมันจะเพิ่มขึ้นทีละหนึ่งเหมือนอย่างเคย แต่ทีนี้ลองปิดเบราเซอร์แล้วเปิดใหม่ แล้วลองเข้าถึงเพจนี้ซ้ำอีก จะพบว่าตัวแปร count จะกลับไปเริ่มต้นที่ 0 ใหม่อีกแล้ว ทั้งนี้เป็นเพราะการปิดเบราเซอร์เป็นการจบ session บีนตัวเก่าจะตาย และเมื่อเปิดเบราเซอร์ขึ้นมาใหม่ session ใหม่จะเกิดขึ้น บีนจะถูกสร้างใหม่ทุกอย่างจึงเริ่มต้นที่ 0 ใหม่ทุกครั้ง

ถ้าคุณมีทั้งอินเตอร์เนตเอ็กซ์พลอเรอร์ และเน็ตเคปคอมมิวนิเคเตอร์ คุณอาจลองเข้าถึงเพจนี้พร้อมๆ กัน เบราเซอร์ต่างยี่ห้อกันแม้จะอยู่บนคอมพิวเตอร์เครื่องเดียวกันจะถือว่าเป็นการเข้าถึงเวบเซิร์ฟเวอร์จากคนละ session กัน คุณจะพบว่าเลขนับจำนวนครั้งที่เยี่ยมชมเวบไซต์ของเบราเซอร์ทั้งสองตัวจะแยกต่างหากกัน

คราวนี้ลองแก้ค่าของ scope ใหม่เป็น page

โปรแกรมที่ 7-4 counter06.jsp

```
<jsp:useBean id="counter" class="mybean.Counter" scope="page" />
<html>
<title>My First JSP</title>
<body>
You are the visitor number <jsp:getProperty name="counter"
   property="count" />.
</body>
</html>
```

ทดลองซ้ำจะเห็นว่าค่าตัวนับจำนวนผู้เยี่ยมชมจะเป็น 1 ตลอดกาลไม่ว่าเราจะรีเฟรชเบราเซอร์กี่ครั้งก็ตาม ที่เป็นเช่นนี้เพราะ page ทำให้บีนถูกสร้างใหม่ทุกครั้งที่เข้าถึงเพจ ค่าของตัวแปร count จึงกลับไปตั้งต้นที่ศูนย์ใหม่ทุกครั้งไป

บีน Counter ที่เราสร้างขึ้นมีแมธธอสอีกตัวหนึ่งที่ยังไม่ได้กล่าวถึงคือ setCount() แมธธอสนี้ใช้เซตค่าของตัวนับใหม่ให้เป็นค่าอะไรก็ตามที่เราต้องการ ตัวอย่างการใช้งานก็เช่น ถ้าวันดีคืนดีเราเกิดอยากจะเซตค่าตัวนับของเราเสียใหม่ให้เริ่มจาก 0 เราก็เรียกแมธธอสตัวนี้โดยผ่านค่า 0 ลงไป การเรียกแมธธอสเซตค่าด้วยการใช้แท็กเจเอสพีใช้คำสั่งต่อไปนี้

```
<jsp:setProperty name="counter" property="count" value="0"/>
```

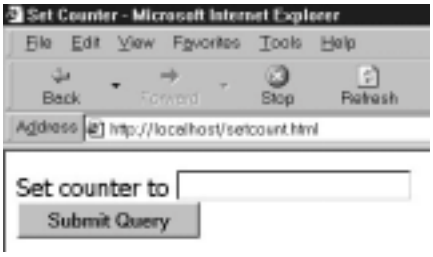
คำสั่งนี้ก็ดูคล้ายๆ กับคำสั่ง getProperty พารามิเตอร์ value ที่เพิ่มขึ้นมาก็คือค่าที่เราต้องการส่งผ่านเข้าไปในแมธธอส getCount() ซึ่งในที่นี้ก็คือ 0 เพราะเราต้องการล้างตัวนับ ตอนนี้ลองสร้างฟอร์มขึ้นมาไว้สำหรับเซตตัวนับโดยเฉพาะดูดังนี้

เริ่มจากฟอร์ม HTML ก่อน

โปรแกรมที่ 7-5 setcount01.html

```
<html>
<title>Set Counter</title>
<body>
```

```
<form action="setcount01.jsp">
Set counter to <input type="text" name="number"><br>
<input type="submit">
</form>
</body>
</html>
```

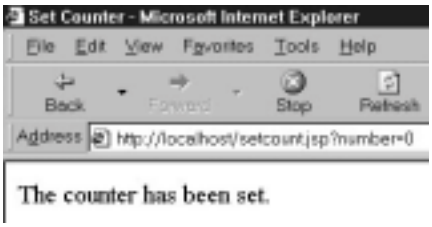


จากนั้นก็สร้างไฟล์ .jsp ขึ้นมารับมือ

โปรแกรมที่ 7-6 setcount01.jsp

```
<jsp:useBean id="counter" class="mybean.Counter" scope="application" />
<% int number = Integer.parseInt(request.getParameter("number")); %>
<jsp:setProperty name="counter" property="count" value="<%=number%>" />

<html>
<title>Set Counter</title>
<body>
The counter has been set.
</body>
</html>
```

ลองใส่ 0 ในแบบฟอร์มแล้วคลิก Submit Query ไฟล์ setcount.jsp จะตอบสนองดังในภาพ



คราวนี้ถ้าลองเรียกไฟล์ counter04.jsp ใหม่อีกครั้งจะพบว่าตัวนับกลับไปตั้งต้นที่ 1 ใหม่

ในกรณีที่ข้อมูลในฟอร์มมีชื่อเหมือนกับตัวแปรในบีนพอดี เราสามารถลดขั้นตอนของการรับส่งตัวแปรได้ด้วยการใช้เครื่องหมาย * เช่น ในโปรแกรมที่ 7-6 จะเขียนใหม่ได้ดังนี้

โปรแกรมที่ 7-7 `setcount02.jsp`

```
<jsp:useBean id="counter" class="mybean.Counter" scope="application" />

<jsp:setProperty name="counter" property="*" />

<html>
<title>Set Counter</title>
<body>
The counter has been set.
</body>
</html>
```

เครื่องหมาย * ที่ส่งให้พารามิเตอร์ property จะทำให้บีนมองหาข้อมูลในฟอร์มที่ส่งเข้ามาว่ามีข้อมูลตัวใดที่มีชื่อเหมือนกับตัวแปรในบีนบ้าง ถ้ามีมันจะเซตค่าตัวแปรในบีนให้ตรงโดยอัตโนมัติ ถ้ามีตัวแปรส่งผ่านเข้ามาหลายตัวมันก็จะตรวจสอบให้ทุกตัว โปรแกรมจึงสั้นลงอย่างมาก

การเรียกใช้งานบีนโดยตรง

อันที่จริงแล้วเราสามารถสร้างบีนแล้วเรียกแมธธอสของบีนโดยตรงในสคริปเลตได้โดยไม่ต้องผ่านแท็ก setProperty และ getProperty ตัวอย่างเช่น

โปรแกรมที่ 7-8 `counter07.jsp`

```
<jsp:useBean id="counter" class="mybean.Counter" scope="application" />
<html>
<title>My First JSP</title>
<body>
You are the visitor number <%=counter.getCount()%>.
</body>
</html>
```

หลังจากเรียกใช้บีนผ่านแท็ก useBean แล้ว เราสามารถเรียกแมธธอส getCount() ของบีนผ่านชื่อ counter ได้โดยตรงราวกับว่าเป็นวัตถุที่เราสร้างขึ้นกับมือ โปรแกรมข้างต้นให้ผลเหมือนกับโปรแกรมที่ 7-2 ทุกประการ

อย่างไรก็ตาม ขอแนะนำให้พยายามใช้งานบีนผ่านแท็กเจเอสพีจะดีกว่า เพราะโปรแกรมจะดูเป็นระบบมากกว่า ตอนนี้อาจจะยังมองไม่เห็นภาพอย่างชัดเจน เพราะโปรแกรมตัวอย่างที่ผ่านมายังไม่ค่อยซับซ้อน แต่ต่อไปจะเห็นชัดขึ้นตอนที่นำไปประยุกต์ใช้งานจริง

8

คุ๊กกี้

คุณคงเคยใช้บริการฟรีหลายๆ อย่างบนอินเตอร์เนต เช่น อีเมล์ หรือฟรีโฮมเพจต่างๆ หรือไม่ก็เคยซื้อหนังสือบนอินเตอร์เนต คุณอาจแปลกใจที่พบว่าเวบไซต์เหล่านั้นสามารถจดจำชื่อ อีเมล์แอดเดรส หรือพาสเวิร์ดของคุณได้ แม้ว่าคุณจะไม่ได้เยี่ยมชมเวบไซต์เหล่านั้นมาหลายเดือนแล้ว เวบไซต์พวกนี้อาศัยการบันทึกข้อมูลเกี่ยวกับตัวคุณไว้ในไฟล์ขนาดจิ๋วซึ่งมันแอบหยอดทิ้งไว้ในฮาร์ดดิสก์ของคุณเวลาที่เข้าเยี่ยมชมเวบไซต์โดยที่คุณไม่รู้ตัว เวลาที่คุณกลับมาที่เวบไซต์เหล่านี้อีกครั้ง มันจะอ่านไฟล์ที่มันหยอดเอาไว้เพื่อดึงข้อมูลเกี่ยวกับตัวคุณกลับมา เราเรียกไฟล์ขนาดจิ๋วเหล่านี้ว่า **คุ๊กกี้**

Cookie

คุ๊กกี้เป็นวัตถุบนเจเอสพี คลาสที่นิยามคุ๊กกี้ได้แก่คลาส Cookie ซึ่งมีวิธีการตั้งชื่อคุ๊กกี้และกำหนดค่าให้กับคุ๊กกี้ด้วยการประกาศวัตถุของคลาสคุ๊กกี้ ตัวอย่างเช่น

```
Cookie cookie = new Cookie(“name”, “Songkarn”);
```

เป็นการประกาศวัตถุชื่อ cookie ของคลาส Cookie โดยให้คุ๊กกี้นี้มีชื่อเรียกว่า name และมีค่าเท่ากับ Songkarn

เวลาเราจะหยอดคุ๊กกี้ลงบนเบราเซอร์ของผู้เยี่ยมชม เราอาศัยแมธธอสของวัตถุแฝง response ชื่อ addCookie() ซึ่งจะทำให้เบราเซอร์สร้างไฟล์คุ๊กกี้ที่เก็บค่าของคุ๊กกี้เอาไว้บน

ฮาร์ดดิสก์ของผู้เยี่ยมชม แต่ก่อนจะหยอดคุ๊กกี้จำเป็นที่จะต้องกำหนดวันหมดอายุของคุ๊กกี้ก่อน คุ๊กกี้ทุกอันที่เราหยอดจะมีวันหมดอายุของมันอยู่เพื่อมิให้ข้อมูลในคุ๊กกี้เก่าเกินไป อายุของคุ๊กกี้นับเป็นวินาทีนับจากวินาทีที่หยอดคุ๊กกี้ และเราใช้แมธธอส setMaxAge() ของคลาส Cookie ในการกำหนดวันหมดอายุ เช่น

```
cookie.setMaxAge(60*60*24*30);
```

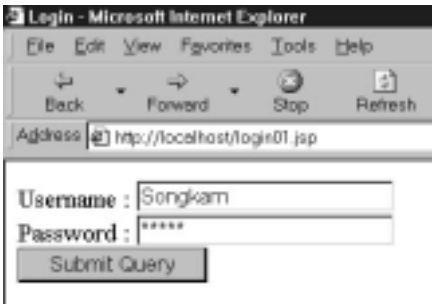
เป็นการกำหนดให้คุ๊กกี้ที่ชื่อ cookie มีอายุ 30 วันนับจากวันที่หยอด

ตัวอย่างง่ายๆ สมมติว่าเวบไซต์ของคุณมีระบบสมาชิก ซึ่งทุกครั้งที่ผู้เยี่ยมชมเข้ามาในเวบไซต์จะต้องระบุชื่อสมาชิกและรหัสผ่าน ถ้าผู้เยี่ยมชมเข้ามาทุกวันวันละหลายหน บางทีผู้เยี่ยมชมอาจรู้สึกรำคาญที่ต้องคอยกรอกชื่อสมาชิกอยู่ตลอดเวลา เราจะลองแก้ปัญหานี้ด้วยการใช้คุ๊กกี้ในการจดจำชื่อสมาชิกในการกรอกรหัสผ่านครั้งแรกของผู้เยี่ยมชม เพื่อที่ในการเยี่ยมชมครั้งต่อไปจะมีชื่อสมาชิกและรหัสเก่าโผล่ออกมารอไว้เลย ผู้เยี่ยมชมเพียงแต่กดปุ่มยืนยันอย่างเดียวก็พอ

เริ่มด้วยการสร้างแบบฟอร์มระบบสมาชิกก่อน

โปรแกรมที่ 8-1 `login01.jsp`

```
<html>
<title>Login</title>
<body>
<form method="post" action="auth01.jsp">
Username : <input type="text" name="username"><br>
Password : <input type="password" name="password"><br>
<input type="submit">
</form>
</body>
</html>
```

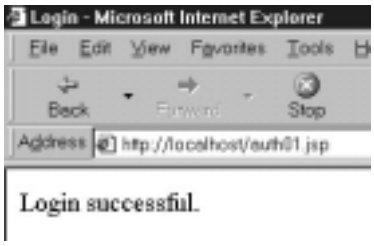


จากนั้นก็สร้างไฟล์ auth01.jsp ขึ้นมารับมือ ลองดูแบบที่ยังไม่มีคุ๊กกี้ก่อนเพื่อความเข้าใจ

โปรแกรมที่ 8-2 `auth01.jsp`

```
<% String username = request.getParameter("username");
   String password = request.getParameter("password");
%>
<html>
<title>Login</title>
<body>
<% if (username.equals("Songkarn")&&password.equals("12345")) { %>
Login successful.
<% } else { %>
Login failed.
<% } %>
</body>
</html>
```

โปรแกรมนี้รับค่า username และ password จากฟอร์มมาเปรียบเทียบ ถ้าชื่อสมาชิกคือ songkarn และ รหัสผ่านคือ 12345 มันจะแสดงข้อความว่าล็อกอินสำเร็จ มิฉะนั้นจะแสดงข้อความว่าการล็อกอินล้มเหลว (ระบบสมาชิกจริงๆ ซับซ้อนกว่านี้มาก แต่เรายังไม่ขอลงรายละเอียด เพราะบทนี้เราสนใจแต่เรื่องการทำคุ๊กกี้)



ที่นี้เราจะเติมคุ๊กกี้ลงไปเพื่อให้ระบบสมาชิกจดจำชื่อสมาชิกที่เคยล็อกอินได้ด้วย เริ่มจากการหยอดคุ๊กกี้ก่อนดังนี้

โปรแกรมที่ 8-3 `auth02.jsp`

```
<% String username = request.getParameter("username");
   String password = request.getParameter("password");
   Cookie cookie = new Cookie("username",username);
   cookie.setMaxAge(60*60*24*30);
   response.addCookie(cookie);
   cookie = new Cookie("password",password);
   cookie.setMaxAge(60*60*24*30);
   response.addCookie(cookie);
%>
<html>
<title>Login</title>
<body>
<% if (username.equals("Songkarn")&&password.equals("12345")) { %>
Login successful.
<% } else { %>
Login failed.
<% } %>
</body>
</html>
```

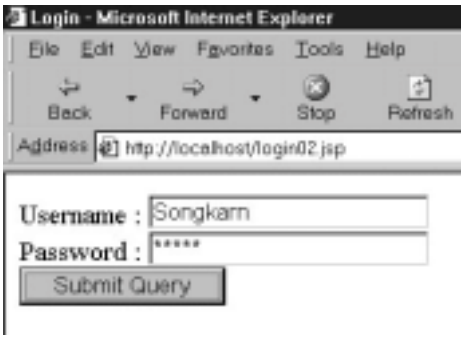
โปรแกรมนี้แทนที่จะรับข้อมูลจากฟอร์มมาเฉยๆ ก็จดใส่คุ๊กกี้ลงไปด้วย คุ๊กกี้ทั้งสองตัวจะมีอายุอยู่บนเครื่องคอมพิวเตอร์ของนายสงกรานต์เป็นเวลา 30 วัน

คราวนี้ก็มาแก้ฟอร์มสมาชิกใหม่ให้มองหาคุ๊กกี้ในเครื่องคอมพิวเตอร์ก่อน ถ้าพบคุ๊กกี้ที่มีชื่อว่า username และ password ก็ให้นำมากรอกไว้ในแบบฟอร์มรอไว้ให้เลย ดังนี้

โปรแกรมที่ 8-4 `login02.jsp`

```
<% String username = new String();
   String password = new String();

   Cookie[] cookies;
   Cookie cookie;
   cookies = request.getCookies();
   if (cookies!=null) {
       for (int i = 0; i < cookies.length; i++) {
            cookie = cookies[i];
            if ("username".equals(cookie.getName())) {
                username = cookie.getValue();
            }
            if ("password".equals(cookie.getName())) {
                password = cookie.getValue();
            }
       }
   }

%>
<html>
<title>Login</title>
<body>
<form method="post" action="auth02.jsp">
Username : <input type="text" name="username" value="<%=username%>"><br>
Password : <input type="password"
   name="password"value="<%=password%>"><br>
<input type="submit">
</form>
</body>
</html>
```

วัตถุแฝง request มีแมธธอสชื่อ getCookies() ซึ่งจะคืนค่าของคุ๊กกี้ทุกตัวในฮาร์ดดิสก์ของผู้เยี่ยมชมมาให้ในรูปของอะเรย์ Cookies[ ] หน้าที่ของเราก็คือตรวจสอบดูว่ามีคุ๊กกี้ตัวไหนบ้างที่มีชื่อว่า username หรือ password ถ้าพบก็ให้นำค่าของมันมากำหนดค่าให้กับ ตัวแปรสตริงที่เราสร้างขึ้นมาชื่อว่า username และ password แมธธอสที่ใช้ในการอ่านชื่อและค่าของคุ๊กกี้ได้แก่ getName() และ getValue() ตามลำดับ เมื่อได้ค่าของ username และ password ซึ่งเหมือนกับชื่อสมาชิกและรหัสผ่านที่เคยใช้แล้ว เราก็นำค่าของมันมากำหนดให้เป็นค่าปกติของฟอร์มด้วยการกำหนดให้กับพารามิเตอร์ value เช่นนี้แล้ว หากผู้เยี่ยมชมกลับเข้ามาตอบคำถามในหน้าเดิมอีก ชื่อเก่าของผู้เยี่ยมชมจะปรากฏขึ้นเองในช่องของชื่อผู้ตอบคำถาม ทำให้ผู้เยี่ยมชมไม่ต้องเสียเวลากรอกชื่อสมาชิกและรหัสซ้ำอีก



สิ่งที่ควรทราบเกี่ยวกับคุ๊กกี้ก็คือ เวลาเบราเซอร์เก็บคุ๊กกี้ไว้ในฮาร์ดดิสก์มันจะแยกคุ๊กกี้ที่มาจากเวบไซต์ต่างเวบไซต์ออกจากกันโดยเด็ดขาด นั้นคือคุ๊กกี้ที่มีชื่อซ้ำกันแต่เกิดจากคนละเวบไซต์จะไม่ปะปนกัน และเบราเซอร์จะยอมให้เวบไซต์ที่หยดคุ๊กกี้นั้นๆ เท่านั้นเป็นผู้อ่านคุ๊กกี้ ไม่มีการใช้งานข้ามเวบไซต์เป็นอันขาด นอกจากนี้เบราเซอร์ยังถือว่าคุ๊กกี้ชื่อเดียวกันที่มาจากเวบไซต์เดียวกันแต่เวบเพจที่หยอดคุ๊กกี้เป็นคนละโฟลเดอร์ เป็นคุ๊กกี้คนละตัวกัน

การใช้ session จดจำข้อมูล

เราสามารถใช้วัตถุแฝง session จดจำข้อมูลได้คล้ายกับคุ๊กกี้ แต่แทนที่จะซ่อนข้อมูลเก่าไว้ในฮาร์ดดิสก์ของผู้เยี่ยมชมจะใช้การเก็บข้อมูลไว้บนฝั่งเวบเซิร์ฟเวอร์แทน ข้อดีของการทำแบบนี้ก็คือ ในบางครั้งผู้เยี่ยมชมที่เป็นห่วงเรื่องของความปลอดภัยและความเป็นส่วนตัวจึงกำหนดให้เบราเซอร์บนเครื่องคอมพิวเตอร์ของตนไม่ให้รับคุ๊กกี้จากเวบไซต์ การใช้ session ทำงานแทนคุ๊กกี้จะไม่มีปัญหาตรงจุดนี้เพราะไม่มีการหยอดคุ๊กกี้ลงบนเบราเซอร์ของผู้เยี่ยมชมแต่ประการใด

อย่างไรก็ดี วัตถุแฝง session มี scope เป็นแบบ session ดังนั้นจึงจดจำข้อมูลของผู้เยี่ยมชมได้เฉพาะชั่วขณะที่ผู้เยี่ยมชมยังติดต่อกับเวบไซต์อยู่หรือตราบเท่าที่ยังไม่ปิดเบราเซอร์ เมื่อผู้เยี่ยมชมปิดการทำงานของเบราเซอร์ข้อมูลเหล่านั้นก็จะหายไป ดังนั้นการใช้ session เก็บข้อมูลอาจจะไม่เหมาะกับการจดจำชื่อสมาชิกและรหัสลับอย่างในกรณีของคุ๊กกี้ในตัวอย่างที่ผ่านมา แต่จะเหมาะกับการใช้จดจำว่าผู้เยี่ยมชมรายใดได้ล็อกอินไปแล้วบ้าง ตราบเท่าที่ผู้เยี่ยมชมรายนั้นยังไม่ปิดเบราเซอร์ผู้เยี่ยมชมไม่ต้องล็อกอินอีก

เราจะลองใช้ session กับระบบสมาชิกที่เราได้สร้างไว้ก่อนหน้านี้ โดย session จะทำหน้าที่จดจำเบราเซอร์ที่ล็อกอินแล้ว เมื่อเวลาที่ผู้เยี่ยมชมต้องการข้ามไปดูเวบหน้าอื่น เวบหน้านั้นๆ จะตรวจสอบว่าผู้เยี่ยมชมล็อกอินแล้วหรือยังด้วยการสืบค้นจาก session

วัตถุแฝง session มีแมธธอสชื่อ setAttribute() และ getAttribute() ซึ่งใช้บันทึกและเรียกดูข้อมูลอะไรก็ได้ ดูไปแล้วก็คล้ายกับคุ๊กกี้ ลองพิจารณาโปรแกรมที่ปรับปรุงใหม่ต่อไปนี้

โปรแกรมที่ 8-5 `auth03.jsp`

```
<% String username = request.getParameter("username");
   String password = request.getParameter("password");
%>
<html>
<title>Login</title>
<body>
<% if (username.equals("Songkarn")&&password.equals("12345")) {
      session.setAttribute("isLoggedin","yes");
%>
Login successful.
<% } else { %>
Login failed.
<% } %>
</body>
</html>
```

โปรแกรมนี้เหมือนกับโปรแกรมที่ 8-2 เพียงแต่เมื่อพบว่าชื่อและรหัสสมาชิกถูกต้อง ให้สร้างตัวเก็บข้อมูลชื่อ isLoggedin ขึ้นมาโดยกำหนดให้มีค่าเท่ากับ yes ด้วยคำสั่ง setAttribute() ตัวเก็บข้อมูลนี้จะคงอยู่ตราบเท่าที่ผู้เยี่ยมชมผู้นี้ยังไม่ปิดการทำงานของเบราเซอร์

คราวนี้เวบหน้าใดก็ตามที่เราไม่ต้องการให้ผู้เยี่ยมชมดูได้โดยไม่ล็อกอินก่อน ก็สามารถตรวจสอบการล็อกอินของผู้ที่เข้ามาด้วยการใช้แมธธอส getAttribute() เพื่อตรวจดูว่า session ของผู้เยี่ยมชมรายนั้นๆ มีตัวเก็บข้อมูลชื่อ isLoggedin อยู่หรือไม่ และมีค่าเท่ากับ yes หรือเปล่า

ลองสร้างไฟล์ .jsp ขึ้นมาไฟล์หนึ่งโดยสมมติว่าเป็นเวบเพจหน้าที่ต้องการมีการล็อกอินก่อน ซึ่งจะดูได้ ดังนี้

โปรแกรมที่ 8-6 sample01.jsp

```
<% String isLoggedin = new String();
   if (session.getAttribute(“isLoggedin”)!=null) isLoggedin =
   (String)session.getAttribute(“isLoggedin”);
%>
<html>
<title>Sample Page</title>
<body>
<%  if (isLoggedin.equals(“yes”)) { %>
Welcome. You are authorized to view this page. <br>
The money in your account is now $500.00.
<% } else { %>
Sorry. You haven’t logged in.<br>
<a href=”login03.jsp”>Click here to log in.</a>
<% } %>

</body>
</html>
```

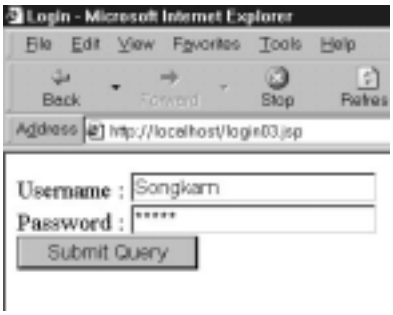
ในส่วนต้นของโปรแกรมนี้เป็นการตรวจสอบดูว่ามีตัวเก็บข้อมูลของ session ที่ชื่อ isLoggedin อยู่หรือไม่ ถ้ามี ให้รับค่าของมันมาใส่ไว้ในตัวแปรสตริงชื่อเดียวกัน จากนั้นทำการตรวจสอบดูว่ามีค่าเป็น yes หรือไม่ ถ้าใช่ก็ให้แสดงเวบเพจหน้านั้น แต่ถ้าไม่ ก็ให้เตือนผู้เยี่ยมชมว่ายังไม่ได้ล็อกอินแทน

ลองสร้างไฟล์ชื่อ login03.jsp ขึ้นมาโดยมีเนื้อหาเหมือนโปรแกรมที่ 8-1 แต่เปลี่ยนพารามิเตอร์ action ให้มีค่าเป็น auth03.jsp ดังนี้

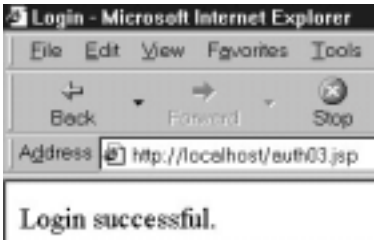
โปรแกรมที่ 8-7 login03.jsp

```
<% String username = new String();
   String password = new String();

%>
<html>
<title>Login</title>
<body>
<form method="post" action="auth03.jsp">
Username : <input type="text" name="username" value="<%=username%>"><br>
Password : <input type="password"
name="password"value="<%=password%>"><br>
<input type="submit">
</form>
```

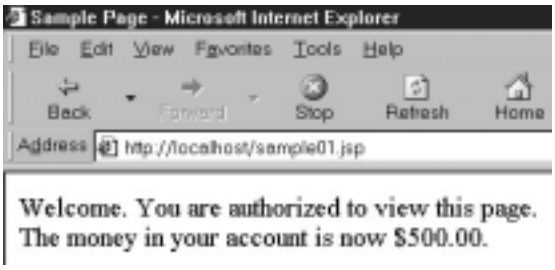
```
</body>
</html>
```



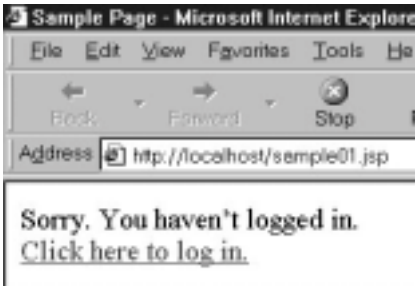
ลองใส่ชื่อ Songkarn และรหัสลับ 12345 ลงไปแล้วคลิก Submit Query มันจะเรียกไฟล์ auth03.jsp มาตอบสนอง



ตอนนี้ session ได้สร้างและเก็บ isLoggedin ที่มีค่าเท่ากับ yes ไว้แล้ว ให้ลองทดสอบโดยการเรียกไฟล์ sample01.jsp จะได้ผลดังนี้



แต่ถ้าลองปิดเบราเซอร์แล้วเปิดใหม่ แล้วเรียกไฟล์ sample01.jsp อีกที คราวนี้จะพบว่าเข้าไม่ได้แล้ว เพราะ isLoggedin ตัวที่มีอยู่ตายไปพร้อมกับการปิดเบราเซอร์ ผู้เยี่ยมชมต้องทำการล็อกอินใหม่



แน่นอนระบบล็อกอินจริงๆ มีความซับซ้อนกว่านี้ นี่เป็นเพียงตัวอย่างเพื่อให้เข้าใจว่าเราจะใช้ session จดจำข้อมูลได้อย่างไรเท่านั้น