ศึกษาเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์สำหรับการพัฒนาเว็บแอพพลิเคชั่น

A STUDY OF JAVASERVER FACES TECHNOLOGY FOR WEB APPLICATION

พยอม ศรีพิมพ์
สุรศักดิ์ วัฒนศิริ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ศึกษาเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์สำหรับการพัฒนาเว็บแอพพลิเคชั่น

A STUDY OF JAVASERVER FACES TECHNOLOGY FOR WEB APPLICATION

ผู้จัดทำ

1. นายพยอม ศรีพิมพ์ รหัสนักศึกษา 48015351
2. นายสุรศักดิ์ วัฒนศิริ รหัสนักศึกษา 48015362

_____อาจารย์ที่ปรึกษา

(ดร. อรัญญา วลัยรัชต์)

ศึกษาเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์สำหรับการพัฒนาเว็บแอพพลิเคชั่น

นายพยอม ศรีพิมพ์ 48015351

นายสุรศักดิ์ วัฒนศิริ 48015362

ดร. อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2550

บทคัดย่อ

JavaServer Faces (JSF) เป็นที่รู้จักอย่างรวดเร็วในการแก้ปัญหาที่สำคัญสำหรับการพัฒนายูสเซอร์อินเตอร์เฟสที่รวดเร็วในแอพพลิเคชั่นของจาวาที่ทำงานบนฝั่งเซิร์ฟเวอร์ เราสามารถออกแบบส่วนของยูสเซอร์อินเตอร์เฟสโดยการวางคอมโพเนนต์ลงไปบนฟอร์มและเชื่อมต่อคอมโพเนนต์เหล่านั้นไปยังอ็อปเจ็คที่สร้างจากคลาสของภาษาจาวาได้ โดยมีเฟรมเวิร์คที่มีความยืดหยุ่นเป็นเครื่องมือที่ช่วยในการออกแบบและสามารถใช้การ drag-and-drop ในการสร้าง GUI ส่วนที่ต่างจากเทคโนโลยีอื่นคือ JSF ช่วยลดความซับซ้อนในการทำงานด้วยการแบ่งส่วนที่ใช้ในการแสดงผลออกจากส่วนที่ใช้ในการจัดการ

ในโครงงานนี้เราจะศึกษาส่วนประกอบต่างๆของ JSF โดยจะมีเว็บแอพพลิเคชั่นที่มีการออกแบบเป็นร้านขายหนังสือออนไลน์เพื่อเป็นตัวอย่างในการใช้ JSF ซึ่งเราจะใช้ Oracle JDeveloper 10g เป็นเครื่องมือในการพัฒนา

A Study of JavaServer Faces Technology for Web Application

Mr. Payom Sripim 48015351

Mr. Surasak Wattanasiri 48015362

Dr. Aranya Walairacht Advisor

Academic Year 2007

ABSTRACT

JavaServer Faces (JSF) is quickly emerging as the leading solution for rapid user interface development in Java-based server-side applications. We can design web user interfaces by putting components on a form and linking them to Java object. The flexible framework was designed for tool supports, and usable of drag-and-drop GUI builders that have been emerged. Unlike other technologies, JSF simplifies the complex implementation works with the separation of presentation and business logic, navigation, connections with external services, and configuration management.

In this project, we propose the studies of JSF and its components. A web application of online book store is designed and implemented as an example using JSF platform. We use Oracle JDeveloper 10g as a tool in the implementation.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก ดร.อรัญญา วลัยรัชต์ และ ผศ.ดร.สมศักดิ์ วลัยรัชต์ ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ที่ได้ให้ความเอาใจใส่ ให้คำแนะนำและความช่วยเหลือเสมอมา ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ทั้งสองท่าน และขอขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณ คณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกๆ เรื่อง

สุดท้ายนี้ขอขอบคุณพี่ๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และกำลังใจเสมอมา ทำให้ข้าพเจ้าสามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายพยอม ศรีพิมพ์
นายสุรศักดิ์ วัฒนศิริ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูปภาพ

รูปที่ **หน้า**

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันมีการใช้เทคโนโลยีการเขียนเว็บแอพพลิเคชั่นมีอยู่หลายภาษาซึ่งแต่ละภาษาก็มีคุณสมบัติเฉพาะที่ต่างกันไป เช่น การเขียนเว็บด้วยเซิร์ฟเล็ตจะต้องรู้และเข้าใจโค้ดภาษาจาวาเป็นอย่างดีเพราะการสร้างเว็บด้วยเซิร์ฟเล็ตจำเป็นต้องใช้โค้ดจาวาทั้งหมด การเขียนเว็บด้วยจาวาเซิร์ฟเวอร์เพจเป็นการนำเอาวิธีการสร้างเว็บแบบสเตติกด้วย HTML รวมเข้ากับโค้ดจาวาทำให้เป็นเว็บแอพพลิเคชั่นแบบไดนามิกจึงทำให้การสร้างเว็บง่ายขึ้น และการเขียนเว็บด้วยจาวาเซิร์ฟเวอร์เฟสซ์จะเป็นการเพิ่มส่วนการจัดการหน้าเพจให้ดียิ่งขึ้น เช่น สามารถจำสะถานะที่มีการร้องขอก่อนหน้าที่ฝั่งเซิร์ฟเวอร์ได้ เป็นต้น

ด้วยเหตุนี้จึงได้ศึกษาเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์เพราะมีการพัฒนาให้การเขียนเว็บแอพพลิเคชั่นมีประสิทธิภาพการทำงานที่ดีที่สุด

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานของจาวาเซิร์ฟเวอร์เฟสซ์
2. เพื่อพัฒนาการเขียนเว็บแอพพลิเคชั่นให้มีประสิทธิภาพ
3. เพื่อนำเสนอเทคโนโลยีใหม่ของภาษาจาวาที่ใช้เขียนเว็บแอพพลิเคชั่น
4. สามารถนำเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์ไปประยุกต์ใช้ได้

1.3 ขอบเขตของโครงงาน

โครงงานนี้ได้นำเสนอตัวอย่างเว็บแอพพลิเคชั่น คือ การสั่งซื้อหนังสือออนไลน์ ที่ใช้การทำงานของเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์มาช่วยการจัดการ โดยมีประเด็นที่สำคัญเกี่ยวกับคุณสมบัติที่โดดเด่นของเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์เป็นหลัก และเพื่อสนับสนุนให้มีการใช้เทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์ในการพัฒนาเว็บมากขึ้น

1.4 วิธีการดำเนินงาน

1. ศึกษาเทคโนโลยีของจาวาเซิร์ฟเวอร์เฟสซ์
2. ศึกษาเกี่ยวกับวิธีการเขียนเว็บโดยทั่วไป
3. เก็บรวบรวมข้อมูลการพัฒนาเว็บในเทคโนโลยีอื่นของภาษาจาวา

4. ศึกษาวิธีบริหารการจัดการของร้านหนังสือ
5. ออกแบบแผนภาพ UML ระบบของร้านหนังสือ
6. ทดลองวิธีการทำงานของแต่ละคุณสมบัติเพื่อประกอบความเข้าใจในทฤษฎี
7. เขียนโปรแกรมของร้านหนังสือด้วยเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์
8. นำไปติดตั้งบนเครื่องเซิร์ฟเวอร์
9. ทดสอบการใช้งานด้วยเว็บบราวเซอร์ที่เครื่องไคลเอ็นต์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ ความเข้าใจรวมทั้งกระบวนการใช้งานเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์
2. สามารถเขียนเว็บแอพพลิเคชั่นที่สร้างจากเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์ได้
3. สามารถเขียน UI Component ที่เป็นแบบเชิงวัตถุได้
4. ยูสเซอร์ได้รับความสะดวกในการใช้งานเว็บมากขึ้น

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บท ด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินงาน ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงานและเทคโนโลยีที่เกี่ยวข้อง

บทที่ 3 กล่าวถึงการออกแบบระบบร้านหนังสือ การทำงานของระบบเมื่อมีลูกค้าติดต่อมา

บทที่ 4 กล่าวถึงการทดลอง การเปรียบเทียบ เทคโนโลยีการเขียนเว็บที่ใกล้เคียงกัน

บทที่ 5 สรุปผลการศึกษา ทดลอง ปัญหา อุปสรรค แนวทางแก้ไข

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ภาษาจาวา

จาวาเป็นภาษาโปรแกรมขั้นสูงที่พัฒนาขึ้นมาโดยบริษัท Sun Microsystems โดยมีวัตถุประสงค์เพื่อให้เป็นภาษาระดับสูงที่ใช้พัฒนาซอฟต์แวร์สำหรับอุปกรณ์ที่ควบคุมโดยไมโครโพรเซสเซอร์ (Embedded System) ที่สามารถนำไปรันได้โดยไม่ขึ้นกับแพลตฟอร์ม (Platform)

ภาษาจาวาได้รับการยอมรับว่าเป็นภาษาที่ดีที่สุดในการพัฒนาซอฟต์แวร์ที่ต้องการความปลอดภัยสูงแบบกระจายผ่านระบบเครือข่าย โดยสามารถใช้พัฒนาได้ตั้งแต่โปรแกรมของอุปกรณ์เน็ตเวิร์กที่ควบคุมโดยไมโครโพรเซสเซอร์ โปรแกรมในเว็บบราวเซอร์ จนถึงแอพพลิเคชั่นทั่วไป ตัวภาษาจาวามีลักษณะเด่นดังนี้

- เป็นภาษาพัฒนาโปรแกรมเชิงวัตถุ
- สามารถนำไปใช้งานได้โดยไม่ขึ้นกับแพลตฟอร์ม
- สามารถประมวลผลแบบกระจาย โดยดึงคลาสไลบรารี่จากที่ต่างๆ ผ่านทาง HTTP และ FTP
- สามารถทำงานได้หลายงานพร้อมกันในเวลาเดียวกัน
- ไม่มีตัวแปรแบบพอยน์เตอร์ (Pointer) เพื่อขจัดปัญหาในการจัดการหน่วยความจำ
- สามารถตรวจสอบและจัดการกับความผิดพลาด (Exception) ที่เกิดขึ้น
- มีการรองรับมาตรฐานความปลอดภัยในการใช้งานรูปแบบต่างๆ

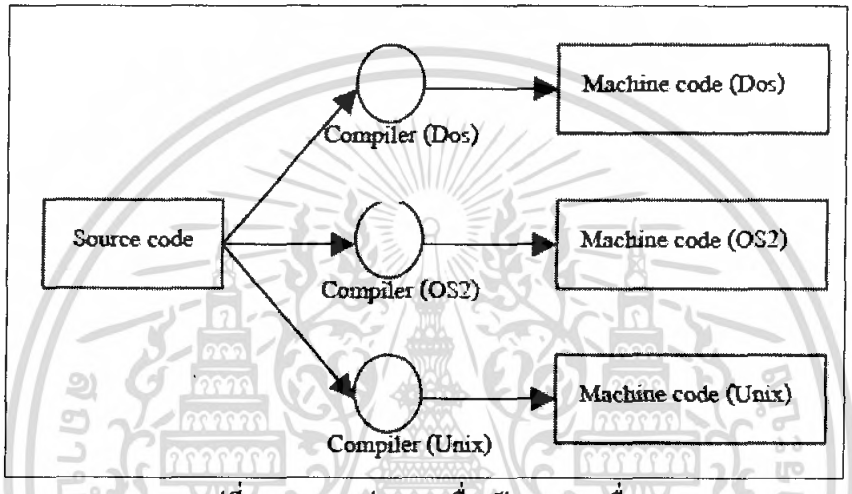
2.2 หลักการทำงานของภาษาจาวา

จาวาเป็นภาษาระดับสูงเชิงวัตถุซึ่งมีกลไกการแปลภาษาและการประมวลผลในลักษณะผสมผสานระหว่างแบบคอมไพเลอร์ (Compiler) และแบบอินเทอร์พรีเตอร์ (Interpreter) โดยทั้งสองแบบมีลักษณะดังนี้

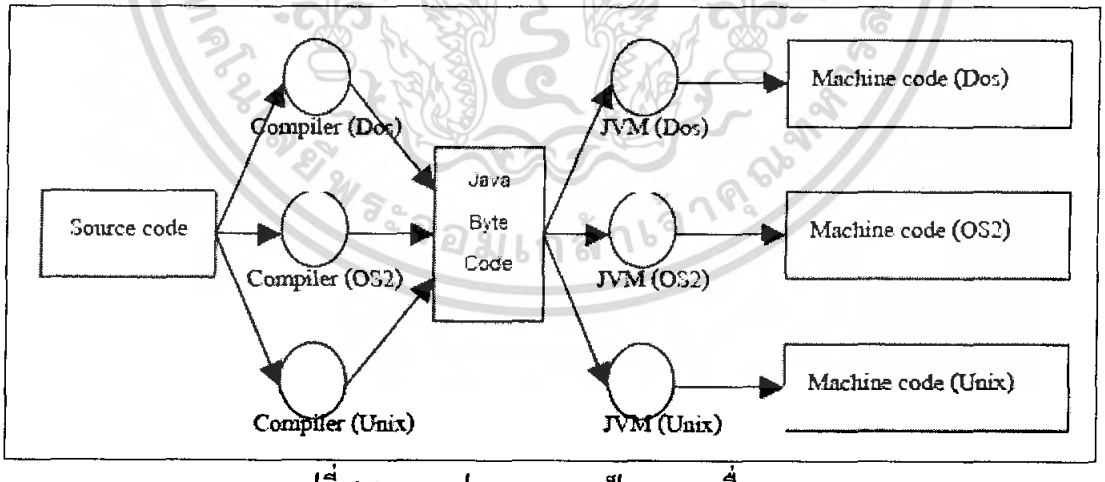
แบบคอมไพเลอร์นั้น ตัวคอมไพเลอร์จะทำหน้าที่วิเคราะห์โค้ดภาษาระดับสูงและแปลเป็นภาษาเครื่อง (Machine code) ซึ่งจะมีข้อดีคือ จะได้โปรแกรมภาษาเครื่องที่มีประสิทธิภาพ เนื่องจากตัวคอมไพเลอร์ได้ทำการวิเคราะห์และปรับแต่งตัวโค้ดให้ได้ภาษาเครื่องที่มีประสิทธิภาพที่สุด แต่ข้อเสียของแบบนี้คือภาษาเครื่องที่ได้จะไม่สามารถนำไปใช้กับแพลตฟอร์มอื่นได้

ส่วนแบบอินเทอร์พรีเตอร์นั้น ตัวอินเทอร์พรีเตอร์จะทำการอ่านโค้ดภาษาระดับสูงทีละบรรทัดแล้วแปลเป็นภาษาเครื่องและทำงานทันที แล้วก็จะกลับไปอ่านโค้ดบรรทัดถัดไป ทำอย่างนี้ไปจนจบโปรแกรม วิธีนี้จะต้องทำการอ่านโค้ดและแปลเป็นภาษาเครื่องสลับกันไป ทำให้ทำงานได้

ช้ากว่าแบบคอมไพเลอร์ แต่ก็มีข้อดีคือตัวอินเทอร์พรีเตอร์สามารถสร้างได้ง่ายกว่าตัวคอมไพเลอร์ ภาษาจาวาได้นำข้อดีของทั้งสองแบบมารวมกัน โดยเมื่อทำการแปลโค้ด ตัวจาวาคอมไพเลอร์ (Java Compiler) จะทำการแปลโค้ดภาษาจาวาเป็นไบต์โค้ด (Byte Code) ที่ไม่ขึ้นกับแพลตฟอร์ม และเมื่อจะทำการประมวลผล ตัวจาวาเวอร์ชวลแมชชีน (Java Virtual Machine) จะทำการแปลไบต์โค้ดทีละคำสั่งเป็นภาษาเครื่องของแพลตฟอร์มที่กำลังทำงานแล้วส่งให้ตัวประมวลผลทำงานตามคำสั่ง วิธีการนี้ทำให้โปรแกรมที่เขียนด้วยภาษาจาวาสามารถทำงานได้ในลักษณะ "เขียนครั้งเดียว ใช้งานได้ทุกที่" คือไม่ว่าจะทำการคอมไพล์โค้ดในแพลตฟอร์มใด ก็สามารถนำไบต์โค้ดที่ได้ไปใช้กับแพลตฟอร์มอื่นที่มีตัวจาวาเวอร์ชวลแมชชีนสำหรับใช้ประมวลผล



รูปที่ 2.1 การแปลภาษาอื่นเป็นภาษาเครื่อง



รูปที่ 2.2 การแปลภาษาจาวาเป็นภาษาเครื่อง

2.3 จาวาสคริปต์ (Javascript)

จาวาสคริปต์เป็นภาษาสคริปต์ที่เกิดขึ้นมาจากความนิยมในอินเตอร์เน็ต และเป็นหนึ่งในภาษาสำหรับอินเตอร์เน็ตที่มีการยอมรับและได้รับความนิยมอย่างสูง ด้วยลักษณะของภาษาสคริปต์ที่ง่ายในการศึกษา เรียนรู้และทำความเข้าใจ แต่ให้ศักยภาพและความสามารถเทียบเท่ากับภาษาระดับสูง

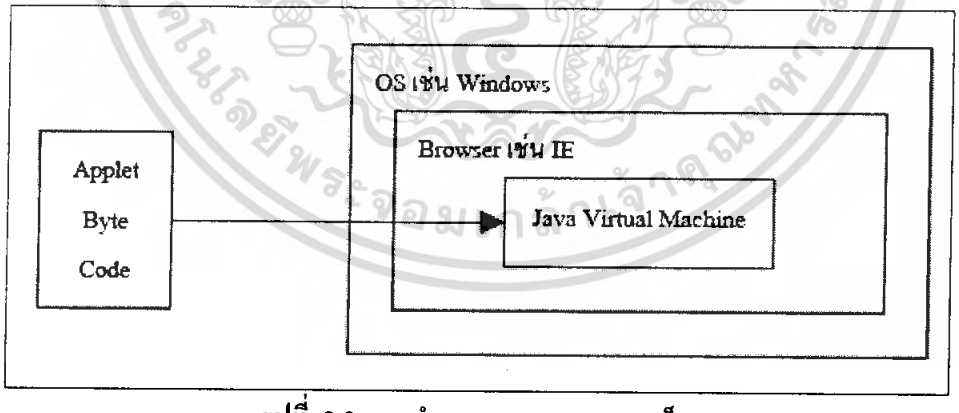
การเขียนจาวาสคริปต์นั้นมีข้อดีที่สามารถใช้บนระบบปฏิบัติการใดๆ ก็ได้และยังสามารถกระโดดไปมาระหว่างระบบปฏิบัติการได้อีกด้วย ซึ่งจะเป็นประโยชน์ในกรณีที่เราต้องการที่จะให้มีการทำงานของสคริปต์ภายใต้บราวเซอร์ในระบบปฏิบัติการต่างๆ โดยจาวาสคริปต์ทำงานได้ทั้งฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์

ถ้าจะเปรียบเทียบการใช้งานระหว่างจาวาสคริปต์กับภาษาจาวาแบบอื่นแล้วอาจจะพิจารณาได้จากเงื่อนไขต่างๆ เช่น ถ้าต้องการความง่ายและรวดเร็วก็ใช้จาวาสคริปต์ แต่ถ้าต้องการโปรแกรมที่มีความซับซ้อนก็จะใช้ภาษาจาวาแบบอื่นซึ่งแน่นอนกว่า

2.4 จาวาแอพเพล็ต (Java Applet)

จาวาแอพเพล็ต คือ โปรแกรมคอมพิวเตอร์ที่เขียนด้วยภาษาจาวาที่ถูกแปลอยู่ในรูปของไบต์โค้ดด้วยการคอมไพล์ และการทำงานของโปรแกรมจะทำงานผ่านจาวาเวอร์ชวลแมชชีนซึ่งอยู่บนบราวเซอร์ที่สนับสนุนการใช้งานจาวาแอพเพล็ต

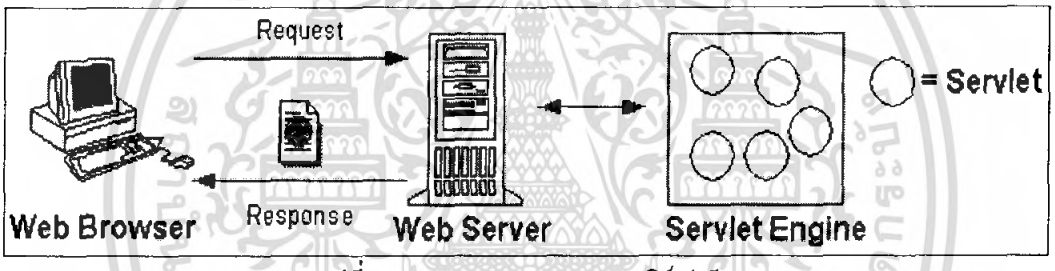
ข้อแตกต่างอีกประการของการสร้างแอพเพล็ตด้วยภาษาจาวากับภาษาจาวาแบบอื่นคือ ตัวจาวาแอพเพล็ตต้องถูกสืบทอดมาจากคลาส Applet



รูปที่ 2.3 การทำงานของจาวาแอพเพล็ต

2.5 จาวาเซิร์ฟเล็ต (Java Servlet)

จาวาเซิร์ฟเล็ตเป็นโปรแกรมที่ทำงานบนเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์ โดยจาวาเซิร์ฟเล็ตนั้นจะทำการประมวลผลบนเครื่องเซิร์ฟเวอร์โดยจะทำงานควบคู่กับเครื่องของผู้ใช้ และเมื่อผู้ใช้ต้องการประมวลผลก็ส่งคำร้องให้จาวาเซิร์ฟเล็ตที่อยู่บนเซิร์ฟเวอร์ทำงานแล้วส่งผลลัพธ์กลับมาให้เครื่องของผู้ใช้อีกที ทำให้เครื่องของผู้ใช้ไม่จำเป็นต้องเป็นเครื่องที่มีประสิทธิภาพมากนัก จาวาเซิร์ฟเล็ตนั้นอ้างอิงหลักการของ CGI โดยข้อดีของจาวาเซิร์ฟเล็ตที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือภาษาจาวานั่นเอง นอกจากนี้จาวาเซิร์ฟเล็ตยังมีความเร็วที่เหนือกว่า CGI เพราะเซิร์ฟเล็ตใช้หลักการของเธรดโดยจะทำการสร้าง 1 เธรดต่อหนึ่งคำร้องที่มาจากไคลเอ็นต์ แต่ตัว CGI จะทำการสร้าง 1 โพรเซสต่อหนึ่งคำร้อง ซึ่งจะสิ้นเปลืองทรัพยากรของระบบมากกว่า ถึงแม้ว่าเซิร์ฟเล็ตจะใช้หลักการของ CGI แต่ตัวเว็บเซิร์ฟเวอร์จะไม่สามารถส่งข้อมูลไปให้เซิร์ฟเล็ตโดยตรงได้เหมือนกับ CGI แต่ต้องทำการเพิ่มส่วนที่ใช้เป็นเสมือนตัวห่อหุ้มเซิร์ฟเล็ตต่างๆ ไว้ โดยส่วนที่เพิ่มขึ้นมานี้เราเรียกว่าเซิร์ฟเล็ตเอ็นจิ้น (Servlet Engine) หรือเซิร์ฟเล็ตคอนเทนเนอร์ (Servlet Container)



รูปที่ 2.4 การทำงานของจาวาเซิร์ฟเล็ต

2.6 จาวาบีน (Java Bean)

จาวาบีนเป็นรูปแบบการใช้งานออปเจ็คที่ถูกสร้างไว้แล้ว และนำมาประกอบกันเป็นซอฟต์แวร์ ถ้าเทียบไปแล้วจะคล้ายกับตัวแอคทีฟคอนโทรล (Active Control) ที่ใช้ในโปรแกรมภาษาวิชวลเบสิก (Visual Basic) แต่จาวาบีนจะสร้างด้วยภาษาจาวาเพื่อให้ใช้งานได้ทุกแพลตฟอร์ม ทำให้จาวาบีนมีการสร้างเพื่อขายเป็นการค้าจากผู้ผลิตรายอื่นๆ (Third Party) ด้วย จาวาบีนจะมีลักษณะดังนี้คือ

- เป็นคลาสในภาษาจาวาเพื่อเรียกใช้ได้ในการพัฒนาโปรแกรมจำพวกวิชวล เช่น บีนเกี่ยวกับปุ่ม บีนเกี่ยวกับรูปภาพ บีนเกี่ยวกับเมนู เป็นต้น ถ้าเปรียบเทียบกับวิชวลเบสิกก็คือสิ่งที่เรียกว่าคอนโทรลนั่นเอง
- เป็นส่วนประกอบทางกราฟฟิกที่สามารถนำมาประกอบกันเป็นโปรแกรมที่ติดต่อกับผู้ใช้แบบ GUI ได้ง่ายขึ้นสะดวกขึ้น เหมือนที่ใช้ในโปรแกรมภาษาวิชวลต่างๆ

- เป็นลักษณะคล้ายๆ กล่องที่เราสามารถเปลี่ยนลักษณะการทำงานและคุณสมบัติต่างๆ ข้างในได้โดยเซ็ตที่ตัวพร็อพเพอตี้ของมัน

2.7 จาวาเซิร์ฟเวอร์เพจ (JavaServer Page)

จาวาเซิร์ฟเวอร์เพจ (JavaServer Page) หรือ JSP เป็นเทคโนโลยีที่ทำงานบนฝั่งเซิร์ฟเวอร์ (Server Side Script) มีความสามารถในการจัดการกับเว็บแอพพลิเคชั่นแบบไดนามิกคอนเทนต์ (Dynamic Content) โดย JSP ถูกพัฒนามาจากเซิร์ฟเล็ต เพื่อแก้ไขปัญหาหนึ่งที่เกิดขึ้นกับเซิร์ฟเล็ต คือ เซิร์ฟเล็ตจะเป็นการผสมข้อมูลในส่วนของการจัดการ (Business Logic) คือข้อมูลทางตรรกะ เช่น จาวาบีน, ฐานข้อมูล กับส่วนของการแสดงผล (Presentation Layer) รวมเข้าไปด้วย นอกจากนี้เซิร์ฟเล็ตยังเปรียบเสมือนจาวาไฟล์ (Java File) ที่มีการฝังข้อความ HTML ลงไป จากปัญหาดังกล่าวทำให้ผู้ที่จะพัฒนาโปรแกรมจำเป็นต้องมีความรู้ทางด้านภาษาจาวามาพอสมควร และการแก้ไขในส่วนของหน้าตาที่ใช้แสดงผลจะทำได้ยาก ส่วน JSP จะมีการแยกข้อมูลส่วน Business Logic กับ Presentation Layer ออกจากกัน นอกจากนี้ JSP ยังเปรียบเสมือน HTML Page ที่มีการฝัง Java Code ลงไป ทำให้โปรแกรมทำงานได้ดีมากขึ้น โดยแยกหน้าที่ได้ตามความถนัดของผู้พัฒนา เช่น หากถนัดเขียนโค้ด ก็ให้ทำงานในส่วนของ Business Logic แต่หากถนัดที่จะออกแบบหน้าตาของเว็บเพจ ก็ให้ทำงานในส่วนของ Presentation Layer

2.8 จาวาเซิร์ฟเวอร์เฟสซ์ (JavaServer Faces)

จาวาเซิร์ฟเวอร์เฟสซ์ (JavaServer Faces) หรือ JSF เป็นเว็บแอพพลิเคชั่นเฟรมเวิร์ค (Framework) ที่ทำให้การออกแบบแอพพลิเคชั่นยูสเซอร์อินเตอร์เฟส (User Interface) นั้นง่ายขึ้น โดยที่มีการแบ่งส่วนที่ใช้ในการนำเสนอ (Presentation) ออกมาจากส่วนที่ใช้ในการจัดการ (Business logic) ซึ่งมีองค์ประกอบที่สำคัญดังนี้

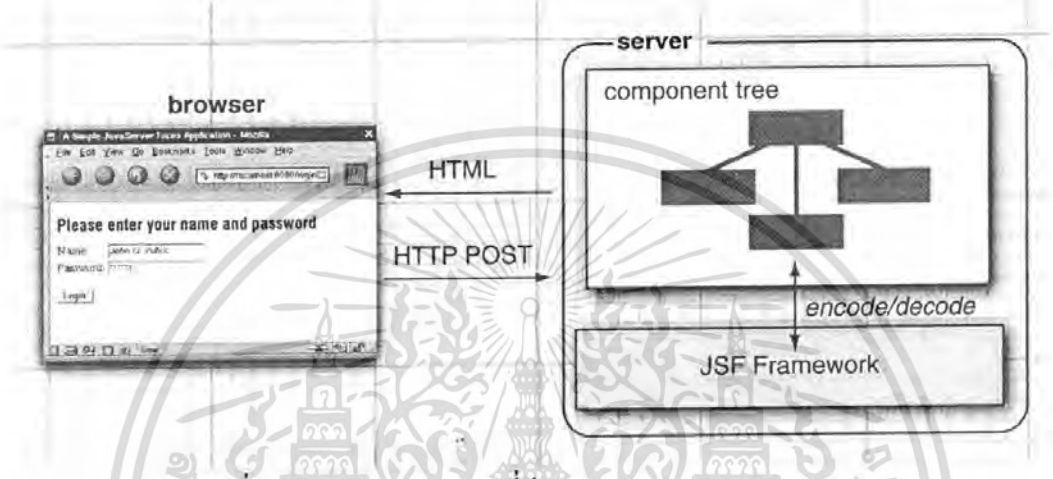
2.8.1 ยูสเซอร์อินเตอร์เฟสคอมโพเนนต์ (User Interface Component)

เป็นส่วนที่ติดต่อกับยูสเซอร์โดยตรง ซึ่งเมื่อเปรียบเทียบกับสวิง (Swing) จะมีส่วนที่เหมือนกัน คือ ถูกสร้างขึ้นมาจากจาวาบีน นั้นหมายความว่าจะมี พร็อพเพอตี้, เมธอด และ event ส่วนที่ไม่เหมือนสวิง ก็คือ คุณลักษณะที่ถูกออกแบบมาเพื่อใช้ในเว็บแอพพลิเคชั่นและทำงานอยู่บนเครื่องเซิร์ฟเวอร์ซึ่งนี้เป็นส่วนที่สำคัญมาก เพราะว่าส่วนยูสเซอร์อินเตอร์เฟสของเว็บแอพพลิเคชั่นโดยมากแล้วไม่ได้สร้างขึ้นมาจากคอมโพเนนต์แต่สร้างมาจากเอาท์พุท มาร์คอัพ (Output Markup) อย่างเช่น HTML

แพ็กเก็ตยูสเซอร์อินเตอร์เฟสอีเลเมนท์ (UI element) เป็นคอมโพเนนต์ที่ทำให้ง่ายในการพัฒนา เพราะว่าแกนกลางของฟังก์ชั่นถูกห่อหุ้มไว้ด้วยส่วนของโค้ดที่สามารถนำกลับมาใช้ใหม่ได้

2.8.2 เรนเดอเรอร์ (Renderer)

เรนเดอร์ถูกรวมเข้าไว้ในเรนเดอร์คิท (Render Kit) ซึ่งโดยปกติแล้วจะให้ความสนใจไปที่ชนิดของเอาท์พุต จาวาเซิร์ฟเวอร์เฟสซ์จะมีเรนเดอร์คิท มาตรฐานสำหรับ HTML 4.01 แต่เรนเดอร์คิทสามารถสร้าง HTML ที่มีรูปร่างหน้าตาแตกต่างกัน เช่น Wireless Markup Language (WML), Scalable Vector Graphics (SVG) หรือสามารถติดต่อกับ จาวาแอพเพล็ต, จาวาแอพพลิเคชั่น หรือเครื่องไคลเอ็นต์ที่มีความแตกต่างกัน



รูปที่ 2.5 แสดงเพจของ JSF ที่มีการ Encoding และ Decoding

เรนเดอร์จะคอยแปลข้อมูลระหว่างเครื่องไคลเอ็นต์กับเซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์มีการส่งข้อมูลออกไป มันจะทำการเข้ารหัส (Encoding) ซึ่งก็คือโพรเซส (Process) ที่สร้างขึ้นมาแทนคอมโพเนนต์เพื่อให้เครื่องไคลเอ็นต์เข้าใจ เมื่อเซิร์ฟเวอร์ได้รับการร้องขอจากเครื่องไคลเอ็นต์ เรนเดอเรอร์จะถอดรหัส (Decoding) ซึ่งก็คือโพรเซส ที่คอยตรวจสอบพารามิเตอร์ที่มีการร้องขอเข้ามา และเซ็ตค่าพื้นฐานของคอมโพเนนต์เหล่านั้น

สำหรับตัวอย่างนี้จะเป็นโค้ดส่วนหนึ่งของคอมโพเนนต์ HtmlInputText

```
<h:inputText id="inputText" size="20" maxlength="30"/>
```

เมื่อคอมโพเนนต์ถูกเข้ารหัส ตัวคอมโพเนนต์จะส่งโค้ด HTML ไปที่เครื่องไคลเอ็นต์ตามตัวอย่างดังนี้

```
<input id="myForm:inputText" type="text"
        name="myForm:inputText" maxlength="30" size="20" />
```

เมื่อยูสเซอร์ป้อนข้อความ “foo” ไปในฟิลด์อินพุต การถอดรหัส คือโพรเซสฟอร์มพารามิเตอร์ส่งไปใน HTTP response และตั้งค่าของคอมโพเนนต์ HtmlInputText บนเครื่องเซิร์ฟเวอร์ให้เป็น “foo”

2.8.3 วาลิเดเตอร์ (Validator)

ในการสร้างยูสเซอร์อินเตอร์เฟสคอมโพเนนต์ จะต้องให้แน่ใจว่าข้อมูลที่ป้อนเข้าไปนั้นจะต้องถูกต้องทั้งหมด ซึ่งจะต้องมีความสัมพันธ์กันกับส่วนที่กำหนดไว้ในฐานข้อมูล เช่น จะต้องไม่ป้อนค่าว่างเปล่า บ่อยครั้งจะมีการบังคับความถูกต้องของข้อมูลโดยการใช้คำสั่ง if ในจาวาสคริปต์เป็นจำนวนมากในการแสดงว่ามีข้อผิดพลาดอะไรเกิดขึ้น ถ้าเราไม่มีเฟรมเวิร์คเข้ามาช่วย

จาวาเซิร์ฟเวอร์เฟสซ์จะมีการจัดการวาลิเดเตอร์ 3 ทาง คือ

ในระดับยูสเซอร์อินเตอร์เฟสคอมโพเนนต์ หรือ ผ่านทางเมธอดวาลิเดเตอร์ใน backing beans หรือ ในคลาสวาลิเดเตอร์ ซึ่งยูสเซอร์อินเตอร์เฟสคอมโพเนนต์โดยทั่วไปจะจัดการกับวาลิเดชั่นง่ายๆ เช่น ไม่ว่าจะเป็นค่าที่ถูกร้องขอมา หรือ ลอจิกวาลิเดชั่นเหล่านั้นจะกำหนดให้กับตัวคอมโพเนนต์เอง (ไม่สามารถใช้กับคอมโพเนนต์อื่นได้) เมธอดวาลิเดเตอร์จะถูกใช้เมื่อเราจำเป็นต้องวาลิเดทฟิลด์หนึ่งฟิลด์หรือมากกว่าบนฟอร์ม (ไม่จำเป็นต้องใช้ลอจิกนั้นร่วมกับคอมโพเนนต์อื่น) วาลิเดเตอร์ภายนอก (External Validator) เป็นการใช้สำหรับกรณีทั่วไป เช่น ความยาวของฟิลด์หรือขอบเขตของตัวเลข วาลิเดชั่นจะถูกจัดการอยู่ที่ฝั่งเซิร์ฟเวอร์ เพราะว่าเครื่องไคลเอ็นต์ทั้งหมดไม่สนับสนุนการใช้สคริปต์ (คอมโพเนนต์จาวาเซิร์ฟเวอร์เฟสซ์สามารถใช้กับ (Support) วาลิเดชั่นบนเครื่องไคลเอ็นต์ได้ แต่ไม่ถือเป็นคอมโพเนนต์มาตรฐาน (Standard Component))

เมื่อวาลิเดเตอร์พบข้อผิดพลาด เช่น สตริง (String) ยาวเกินไป หรือใส่หมายเลขบัตรเครดิตผิด ตัววาลิเดเตอร์จะเพิ่มข้อความผิดพลาด (error message) ไปยังบัญชีข้อความปัจจุบัน (massage list) ซึ่งจะทำให้ง่ายต่อการแสดงวาลิเดชั่นที่ผิดพลาด (validation error) กลับไปที่ยูสเซอร์ที่กำลังใช้คอมโพเนนต์จาวาเซิร์ฟเวอร์เฟสซ์มาตรฐาน ซึ่งมีตัวอย่างดังนี้

```
<h:inputText>
        <f:validateLength minimum="2" maximum="10"/>
</h:inputText>
```

จะเห็นว่ามีความง่ายในการเชื่อมโยงวาลิเดเตอร์เข้ากับคอมโพเนนต์ ในตัวอย่างเป็นการกำหนดคอมโพเนนต์ HtmlInputText เข้ากับ Length วาลิเดเตอร์เพื่อตรวจสอบให้แน่ใจว่า อินพุตของยูสเซอร์ที่ป้อนเข้ามามีความยาวของตัวอักษรอยู่ระหว่าง 2 ถึง 10 ตัวอักษร

2.8.4 คอนเวอเตอร์ (Converter)

เมื่อยูสเซอร์มีการโต้ตอบกับจาวาเซิร์ฟเวอร์เฟสซ์แอพพลิเคชั่น จะกระทำกับเอาต์พุตของเรนเดอเรอร์ซึ่งเป็นตัวแทนที่สร้างขึ้นมาโดยเฉพาะสำหรับแต่ละเครื่องไคลเอ็นต์ (เหมือนเว็บบราวเซอร์) ในลำดับต่อไปเรนเดอเรอร์จะต้องมีความเข้าใจเกี่ยวกับคอมโพเนนต์ที่แสดงด้วย แต่คอมโพเนนต์สามารถเชื่อมโยงเข้ากับคุณสมบัติของ Backing Beans คุณสมบัติเหล่านั้นสามารถเป็น

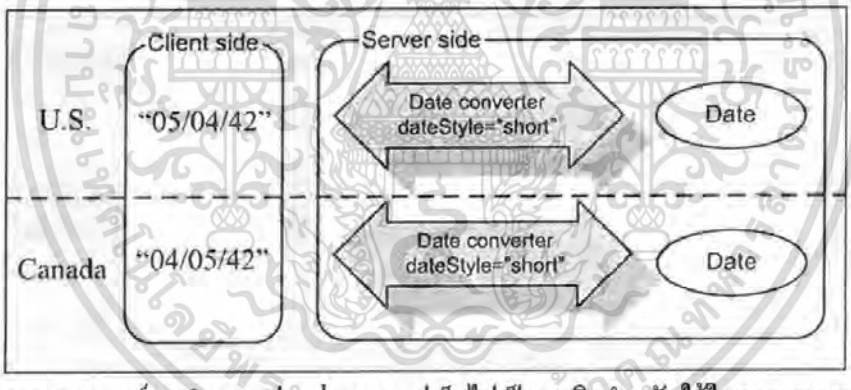
อะไรก็ได้ เช่น สตริง (String) ที่ใช้แทนชื่อ วันที่ (Date) ที่ใช้แทนวันเกิด เพราะว่ามันไม่มีข้อจำกัดของชนิดข้อมูล เรนเดอเรอร์จึงไม่สามารถรู้ก่อนล่วงหน้าว่าจะแสดงออปเจ็คอย่างไร

นั้นจึงเป็นที่มาว่าทำไมจึงต้องมีคอนเวอเตอร์เข้ามา ซึ่งมันจะแปลงออปเจ็คไปเป็นสตริงสำหรับใช้แสดงผลและจากอินพุตฟอร์มกลับไปเป็นออปเจ็ค คอนเวอเตอร์หนึ่งตัวสามารถเชื่อมโยงเข้ากับคอนโทรลกี่ตัวก็ได้ จาวาเซิร์ฟเวอร์เฟสซ์รวมคอนเวอเตอร์จาก Common Types เหมือนกับวันที่และเลขจำนวน

ยิ่งไปกว่านั้นแล้วคอนเวอเตอร์จะจัดการ formatting และ localization ด้วย ตัวอย่างเช่น DateTime Converter สามารถจัดรูปแบบของออปเจ็ควันที่ให้สัมพันธ์กับยูสเซอร์ ส่วนตัวอย่างต่อไปนี้จะแสดงการลงทะเบียน (Register) บนคอมโพเนนต์ HtmlOutputText

```
<h:outputText value="#{user.dateOfBirth}">
        <f:convert_datetime type="both" dateStyle="full"/>
</h:outputText>
```

สมมุติว่ายูสเซอร์เกิดวันที่ 4 พฤษภาคม ค.ศ. 1942 คอมโพเนนต์ HtmlOutputText จะแสดงสตริง "05/04/42" ถ้ายูสเซอร์อยู่ที่สหรัฐอเมริกา แต่จะแสดง "04/05/42" ถ้ายูสเซอร์อยู่ที่แคนาดา ซึ่งแสดงดังรูปต่อไปนี้



รูปที่ 2.6 คอนเวอเตอร์จะทำการแปลงค่าจากออปเจ็คไปเป็นสตริงสำหรับใช้ในการแสดงผล โดยจะทำการควบคุมรูปแบบการแสดงผลให้รองรับความหลากหลายของภาษา

2.8.5 เนวิกาชั่น (Navigation)

เว็บแอพพลิเคชั่นจะมีหน้าเพจ (Page) หลายหน้า และจะมีเส้นทางที่ใช้ในการไปยังแต่ละหน้า ตัวที่ทำหน้าที่เปลี่ยนจากหน้าหนึ่งไปยังอีกหน้าหนึ่งเรียกว่า เนวิกาชั่น

จาวาเซิร์ฟเวอร์เฟสซ์มีระบบเนวิกาชั่นที่เรียบง่ายและสวยงาม โดยที่ Navigation Handle จะรับผิดชอบในการตัดสินใจว่าเพจไหนจะถูกโหลดมาโดยพิจารณาจากเอาต์คัม ที่รับเข้ามาจากเมธอดแอคชั่น สำหรับเพจที่ให้มานั้น Navigation Rule จะเป็นตัวกำหนด ซึ่งจะกำหนดระหว่าง

เอาต์คัม และเพจไว้ใน Navigation Case ในไฟล์ JSF Configuration ซึ่งต่อไปนี้จะเป็น Navigation Rule ของ login.jpg ที่มีเอาต์คัมอยู่ 2 กรณี คือ “success” และ “failure”

```
<navigation-rule>
        <from-view-id>/login.jsp</from-view-id>
        <navigation-case>
                <from-outcome>success</from-outcome>
                <to-view-id>/mainmenu.jsp</to-view-id>
        </navigation-case>
        <navigation-case>
                <from-outcome>failure</from-outcome>
                <to-view-id>/login.jsp</to-view-id>
        </navigation-case>
</navigation-rule>
```

2.8.6 แบล็กกิ้งบีน (Backing bean)

จาวาเซิร์ฟเวอร์เฟสซ์ยอมให้เราเชื่อมโยงแบล็กกิ้งบีนเข้ากับยูสเซอร์อินเตอร์เฟสคอมโพเนนต์ผ่านทาง JSF expression language (EL) ซึ่งเหมือนกับ JSP 2.0 และ JSTL expression language เราสามารถใช้ JSF expression language เพื่อชี้เฉพาะไปที่คุณสมบัติบางที่ของแอพพลิเคชั่นซึ่งมีตัวอย่างดังนี้

```
<h:outputText id="helloBeanOutput"
        value="#{helloBean.numControls}"/>
```

โค้ดในส่วนนี้เกี่ยวกับค่าของคอมโพเนนต์ HtmlOutputText ที่พร็อพเพอตี้ numControls ของออปเจ็คที่เรียกว่า helloBean ซึ่งค่าของคอมโพเนนต์จะเปลี่ยนแปลงตามค่าของ helloBean.numControls

2.8.7 แมสเสจ (Message)

จาวาเซิร์ฟเวอร์เฟสซ์ได้จัดหาแมสเสจเพื่อช่วยในการประกาศข้อความประกอบด้วยใจความที่สรุปเนื้อหารายละเอียด

แมสเสจไม่จำเป็นต้องมีไว้เพียงบอกข้อผิดพลาด มันสามารถใช้เป็นแหล่งของข้อมูลข่าวสารได้ ตัวอย่างเช่น action listener method ที่มีการเพิ่มข้อความเข้ามาเมื่อมีการเพิ่มเรคคอร์ดใหม่เรียบร้อยแล้ว ตัวแมสเสจช่วยในการเชื่อมโยงเข้ากับคอมโพเนนต์ตัวใดตัวหนึ่ง (สำหรับอินพุตที่ผิดพลาด) หรือไม่ใช้กับทุกคอมโพเนนต์ (สำหรับแมสเสจของแอพพลิเคชั่น)

เราสามารถแสดงข้อความผิดพลาดที่เชื่อมโยงเข้ากับคอมโพเนนต์โดยใช้ HtmlMessage

```
<h:message id="errors" for="helloInput" style="color: red"/>
```

tag แสดงข้อผิดพลาดทั้งหมดนั้นถูกสร้างขึ้นสำหรับอินพุทคอมโพเนนต์ helloInput (ซึ่งต้องประกาศไว้ในหน้าเพจเดียวกัน) เราสามารถแสดงแมสเสจทั้งหมด หรือไม่เชื่อมโยงแมสเสจเหล่านั้นกับคอมโพเนนต์ ด้วยการใช้คอมโพเนนต์ HtmlMessage อื่น

2.8.8 อีเว็นท์และลิสต์เท็นเนอร์ (Event and listener)

เมื่อเราเขียนจาวาเซิร์ฟเวอร์เฟสซ์แอพพลิเคชั่นลอจิก แอพพลิเคชั่นจะรวมเข้าไปด้วย เพื่อกำหนดให้คอมโพเนนต์สร้างอีเว็นท์ที่ลิสต์เท็นเนอร์เข้าใจ โดยเราไม่ต้องไปคิดเกี่ยวกับ requests และ responses ทั้งหมด โดยมาตรฐานจะมี event 4 ประเภทคือ

2.8.8.1 value-change event

value-change event จะถูกสร้างขึ้นโดยส่วนควบคุมอินพุต (Input Control) เมื่อยูสเซอร์ป้อนค่าใหม่เข้าไปยังอินพุตคอมโพเนนต์ ซึ่งเราสามารถจัดการกับ value-change event ด้วย value-change listener

สำหรับตัวอย่างต่อไปจะยกตัวอย่างของคอมโพเนนต์ HtmlInputText และ HtmlPanelGrid ที่อยู่ในเพจเดียวกัน

```
<h:inputText valueChangeListener="#{myForm.processValueChanged}"/>
<h:panelGrid binding="#{myForm.changePanel}" rendered="false">
....
</h:panelGrid>
```

สำหรับคอมโพเนนต์ HtmlInputText จะทำการกำหนดค่าให้กับแอตทริบิวต์ valueChangeListener ไปที่พร็อพเพอตี้ที่ชื่อ processValueChanged ที่อยู่ในคลาสบีน ที่ชื่อ myForm ในส่วนของ HtmlPanelGrid จะรวมไว้กับพร็อพเพอตี้ที่ชื่อ changePanel

เมื่อยูสเซอร์ทำการเปลี่ยนค่าที่อยู่ในคอมโพเนนต์แล้วกด submit ฟอร์มจาวาเซิร์ฟเวอร์เฟสซ์จะทำการสร้าง value-change event จากนั้นมันก็จะไปที่เมธอดอีเว็นท์ลิสต์เท็นเนอร์เพื่อประมวลผลอีเว็นท์ที่เกิดขึ้น

```
public void processValueChanged(ValueChangeEvent event)
{
    HtmlInputText sender = (HtmlInputText)event.getComponent();
    sender.setReadonly(true);
    changePanel.setRendered(true);
}
```

ในตัวอย่างนี้เมธอดอีเว็นท์ลิสต์เท็นเนอร์จะทำการเซ็ตคุณสมบัติของ sender (โดยที่ HtmlInputText เป็น UI คอมโพเนนต์) ให้เป็น true เพื่อให้ยูสเซอร์ไม่สามารถแก้ไขอะไรได้ จากนั้นจะทำการเปลี่ยนคุณสมบัติของ changePanel ให้เป็น true เพื่อให้สามารถมองเห็น

2.8.8.2 action event

action event จะเกิดขึ้นโดยการที่ยูสเซอร์กระทำกับคอมโพเนนต์จากนั้นคอมโพเนนต์ก็จะสร้าง action event ที่เรียกว่า action source รวมเข้ากับคอนโทรล เช่น buttons, hyperlink โดยที่ action event นี้จะถูกจัดการโดย action listener

action listener มีอยู่ด้วยกัน 2 ประเภทคือ action listener ที่มีผลต่อระบบ เนวิกาชั่นและ action listener ที่ไม่มีผลต่อระบบเนวิกาชั่น

โดยที่ action listener ที่มีผลต่อระบบเนวิกาชั่นของจาวาเซิร์ฟเวอร์เฟสซ์นั้นโดยปกติแล้ว หลังจากที่มีการประมวลผล จะมีการคืนค่าเอาต์คัมกลับมา ซึ่งเป็นลอจิกคอลที่จะถูกนำไปใช้โดยระบบเนวิกาชั่นของจาวาเซิร์ฟเวอร์เฟสซ์เพื่อที่จะเลือกหน้าเว็บที่จะใช้ในการแสดงผลต่อไป ส่วน action listener ที่ไม่มีผลต่อระบบเนวิกาชั่นนั้นจะเป็นการใช้สำหรับปรับเปลี่ยนค่าของคอมโพเนนต์ที่อยู่ใน view หรือทำการประมวลผลเพื่อเป็นการปรับเปลี่ยนโมเดลออปเจ็คหรือเปลี่ยนคุณสมบัติที่อยู่ในคลาสบีน แต่ว่าไม่มีการเปลี่ยนหน้าเพจปัจจุบันที่ยูสเซอร์กำลังใช้งานอยู่ ดังนั้นหน้าเพจจะยังคงแสดงอยู่หลังจากที่สิ้นสุดการทำงานของลิสต์เท็นเนอร์

action event ที่เกิดขึ้นมานั้น ซึ่งโดยปกติแล้ว action listener จะเป็นตัวที่สร้างเอาต์คัมสตริงขึ้นมา มีด้วยกัน 2 ประเภท คือ สเตติกเอาต์คัม (static outcome) และไดนามิกเอาต์คัม (dynamic outcome)โดยที่ สเตติกเอาต์คัมนั้นจะเป็นสตริงที่ถูกกำหนดไว้ที่ตัวของคอมโพเนนต์

```
<h:commandButton type="submit"
    value="Login" action="success" immediate="true"/>
```

ในตัวอย่างนี้จะเป็นสเตติกเอาต์คัมที่กำหนดไว้ที่แอตทริบิวต์แอคชั่นมีค่าเป็น “success” มันจะถูกสร้างขึ้นเมื่อยูสเซอร์คลิ๊กไปที่ HtmlCommandButton ทำให้เกิด action event ขึ้นมาแต่ว่าไม่มีการไปเรียกใช้งานเมธอดแอคชั่น

ไดนามิกเอาต์คัมเป็นสตริงที่คืนค่ากลับมาจากตัวเมธอดแอคชั่นเอง ซึ่งเมธอดแอคชั่นจะคืนค่าที่แตกต่างกันออกมา โดยที่ action listener จะมองหาเมธอดแอคชั่นที่อยู่ในแอตทริบิวต์แอคชั่นของคอมโพเนนต์ และจะทำการประมวลผลในส่วนของเมธอดแอคชั่นแทน

```
<h:commandButton type="submit"
    value="Login" action="#{loginForm.login}"/>
```

เมื่อยูสเซอร์คลิ๊กที่ปุ่มนี้ จะทำให้ action event ถูกกระตุ้นการทำงาน และเมธอดต่อไปนี้จะทำงานและคืนค่าเป็นเอาต์คัมออกมา

```
public class LoginForm {
    ...
    public String login() {
        if (...) { // login is successful
            return "success";
        }
        else {
            return "failure";
        }
    }
    ...
}
```

สำหรับตัวอย่างนี้หลังจากที่เมธอดแอคชั่นประมวลผลเสร็จแล้วจะมีการคืนค่าเอาต์คัมออกมา ขึ้นอยู่กับการประมวลผลจากข้อมูลที่รับเข้ามา ค่าที่ได้อาจจะเป็น “success” หรือ “failure” ซึ่งค่าที่ได้นี้นำไปใช้กับระบบเนวิกาชั่นเพื่อใช้เลือกว่าจะให้หน้าเพจไหนแสดงออกมา

2.8.8.3 data model event

data model event จะเกิดขึ้นเมื่อคอมโพเนนต์รับรู้ว่ามีการกระทำกับข้อมูลที่อยู่ในตัวมัน เช่นการนำข้อมูลที่เราเลือก (select) ใน HtmlDataTable มาแสดงที่ List Item ซึ่งในส่วนนี้จะไม่เหมือนกับ value-change หรือ action event listener โดยที่ data model event จะต้องอิมพลีเมนท์มาจาก Java Interface

data model event จะมีส่วนที่ต่างจากอีเว็นท์อื่นๆ อยู่เล็กน้อย เพราะว่าอีเว็นท์ที่เกิดขึ้นจะไม่ได้มาจาก UI คอมโพเนนต์ แต่จะเกิดจาก instance ของ data model event

2.8.8.4 phase event

อย่างไรก็ตามแอพพลิเคชั่นที่พัฒนาด้วยจาวาเซิร์ฟเวอร์เฟสซ์จะรับรีเควสที่มาจากขั้นตอนการประมวลผลทั้ง 6 ที่เรียกว่า Request Processing Lifecycle ในระหว่างที่ประมวลผลอยู่นี้จาวาเซิร์ฟเวอร์เฟสซ์จะ restore request view, translates request พารามิเตอร์เข้าไปในค่าของคอมโพเนนต์ validate input, update class bean, model object, invokes action listener และมีการตอบกลับไปที่ยูสเซอร์ phase event จะถูกสร้างขึ้นก่อนและหลังของแต่ละขั้นตอนใน Lifecycle

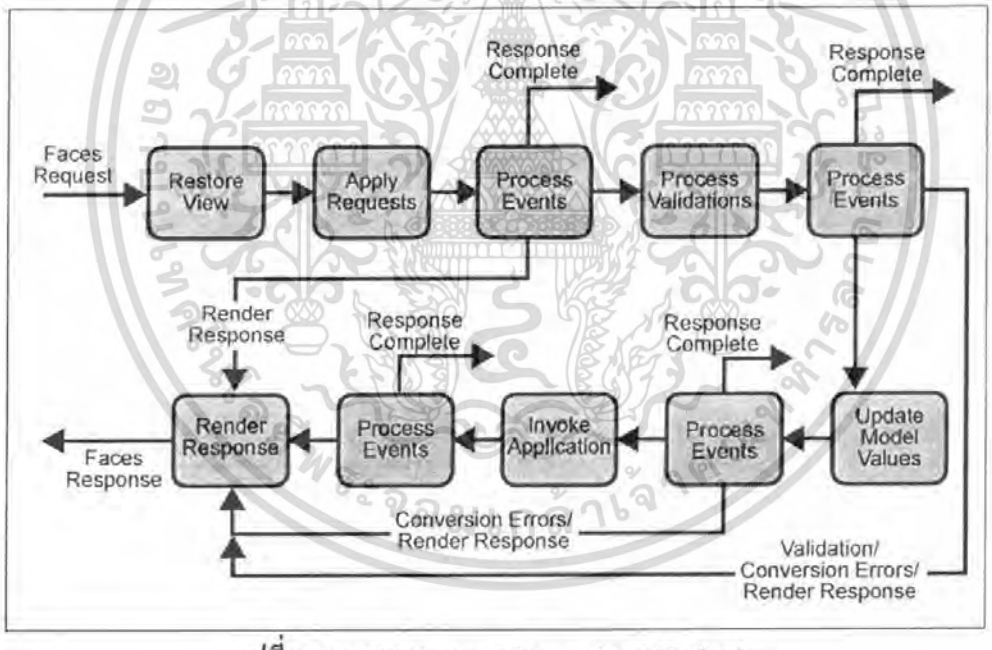
ถึงแม้ว่ายูสเซอร์อินเตอร์เฟสคอมโพเนนต์จะสามารถสร้าง phase event ขึ้นมาแต่ส่วนมากแล้วจาวาเซิร์ฟเวอร์เฟสซ์จะสร้างขึ้นมาด้วยตัวเองมากกว่า และโดยปกติแล้วจะถูกอิมพลีเมนท์โดยจาวาเซิร์ฟเวอร์เฟสซ์แต่บางครั้งก็สามารถอิมพลีเมนท์โดยแอพพลิเคชั่นที่พัฒนาขึ้นมาเอง อย่างเช่น เราสามารถทำการกำหนดค่าเริ่มต้น (initialize) ก่อนการแสดงผลได้

2.9 วงจรการทำงานของหน้าเพจ JSF (Lifecycle of JSF Page)

ไฟล์จาวาเซิร์ฟเวอร์เฟสซ์จะถูกแทนด้วยโครงสร้าง tree ของคอมโพเนนต์ที่เรียกว่า view เมื่อเครื่องไคลเอ็นต์ทำการร้องขอหน้าเพจ วงจรการทำงานก็จะเริ่มต้นขึ้น ในช่วงระยะเวลาของวงจรการทำงานนั้นจาวาเซิร์ฟเวอร์เฟสซ์จะต้องทำการอิมพลีเมนท์โดยการสร้าง view โดยพิจารณาจากสถานะที่ถูกบันทึกไว้ในครั้งก่อน ในขณะที่เครื่องไคลเอ็นต์ทำการ Postback หน้าเพจเข้ามาจาวาเซิร์ฟเวอร์เฟสซ์จะต้องทำการอิมพลีเมนท์โดยทำในส่วนของขั้นตอนของวงจรการทำงานดังนี้

- วาลิเดชั่น
- คอนเวอชั่น

2.9.1 วงจรการทำงานของการประมวลผลการร้องขอ (Request Processing Life Cycle)



รูปที่ 2.7 แสดง Request Processing Life Cycle

วงจรการทำงานของการประมวลผลการร้องขอจะจัดการกับ request 2 ชนิดคือ Initial request และ Postback

- Initial request : ขั้นตอนนี้จะเกิดขึ้นเมื่อยูสเซอร์ทำการ request หน้าเพจในครั้งแรก โดยที่วงจรการทำงานจะทำในขั้นตอนของ restore view และ render response

• Postback : ขั้นตอนนี้จะเกิดขึ้นเมื่อยูสเซอร์ทำการ submit ฟอร์มที่อยู่ในหน้าเพจที่ทำการโหลดเข้ามาในบราวเซอร์ในขั้นตอนของ Initial request ในครั้งก่อนนี้ โดยที่จะทำตามการจัดการทั้งหมดใน Lifecycle

เฟสวงจรการทำงานของการประมวลผลการร้องขอ(Request Processing Life Cycle Phases) จะมีขั้นตอนทั้งหมดดังนี้

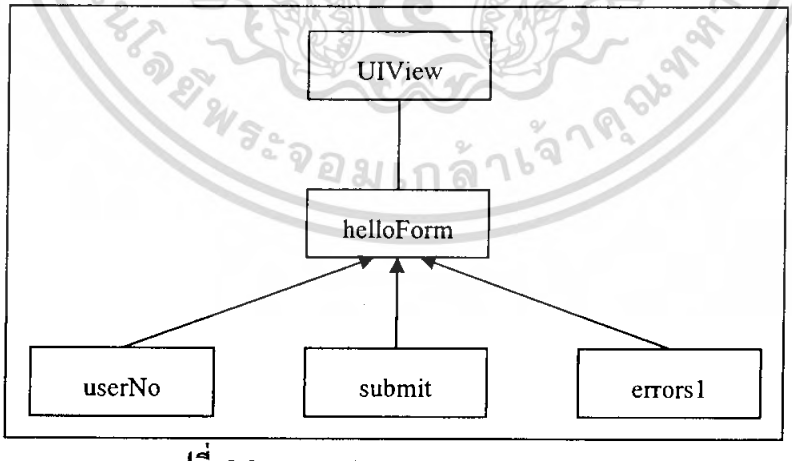
2.9.1.1 Reconstitute component tree phase หรือเรียกอีกชื่อหนึ่งว่า Restore view

เมื่อมี request สำหรับจาวาเซิร์ฟเวอร์เฟสซ์เพจเกิดขึ้นมา อย่างเช่น เมื่อมีการคลิ๊กไปที่ link หรือปุ่มของจาวาเซิร์ฟเวอร์เฟสซ์จะเริ่มต้นทำการอิมพลีเมนท์ในขั้นตอน restore view โดยที่ถ้าเป็นการ request ในครั้งแรก จะทำการสร้าง empty view ขึ้นมา ซึ่งในขั้นตอนปกติแล้วจาวาเซิร์ฟเวอร์เฟสซ์จะทำการอิมพลีเมนท์ตามขั้นตอนดังนี้

• สร้าง view ของเพจจาวาเซิร์ฟเวอร์เฟสซ์ถ้าการ request หน้าเพจเป็น initial request จาวาเซิร์ฟเวอร์เฟสซ์จะอิมพลีเมนท์โดยการสร้าง empty view และก็จะข้ามไปทำในขั้นตอน render response โดยที่ empty view จะยังอยู่ในขณะที่มีการประมวลผลการ postback ถ้าการ request หน้าเพจเป็น postback view ที่อยู่ในขั้นตอนของการ initial request จะยังคงอยู่ระยะเวลาในขั้นตอนนี้ Face จะอิมพลีเมนท์โดยการ restore view โดยใช้สถานะของข้อมูลที่บันทึกไว้ที่เครื่องไคลเอ็นต์หรือเครื่องเซิร์ฟเวอร์

• จะทำการเชื่อม handler และวาลิเดเตอร์เข้ากับคอมโพเนนต์ใน view และบันทึก view ไว้ใน Instance ของ FacesContext ซึ่งเก็บข้อมูลทั้งหมดที่จำเป็นในการประมวลผลการ request หนึ่งครั้ง

• จากนั้นก็จะนำ view ที่ได้ ไปเก็บไว้ใน FacesContext



รูปที่ 2.8 แสดง view ของเพจ greeting.jsp

2.9.1.2 Apply request values phase

แต่ละคอมโพเนนต์ใน tree จะเอาค่าใหม่ที่มาจาก request พารามิเตอร์ที่ได้จาก built-in decode method ของตัวมันเอง ภายใน decode method ค่าต่างๆจะถูกแปลงให้เป็นค่าที่ถูกต้องแล้วนำไปเก็บไว้ในคอมโพเนนต์นั้น สำหรับคอมโพเนนต์ userNo ในไฟล์ greeting.jsp จะทำการ convert จากสตริงเป็น integer ในส่วนที่เป็น conversion error จะนำเข้าไปในคิวของ FacesContext

2.9.1.3 Process validations phase

จาวาเซิร์ฟเวอร์เฟสซ์จะทำการอิมพลีเมนท์ทุกวาลิเดชั่นอินพุตการประมวลผล ที่ถูกรีจิสเตอร์ไว้กับคอมโพเนนต์ใน tree ในที่นี้จะเป็นวาลิเดชั่นอินพุต (ไม่ใช่ business logic วาลิเดชั่น) ในกรณีที่วาลิเดชั่นผิดพลาดขึ้น error message จะนำเข้าไปในคิวของ FacesContext และวงจรการทำงานจะข้ามไปที่ขั้นตอน Render Response ตัวอย่างเช่น userNo จะมีค่าอยู่ระหว่าง 1-10

2.9.1.4 Update model values phase

จาวาเซิร์ฟเวอร์เฟสซ์จะทำการอิมพลีเมนท์โดยการเข้าไปในคอมโพเนนต์ tree และจะทำการเซ็ตค่าคุณสมบัติต่างๆ ของออปเจ็คที่อยู่บนฝั่งเซิร์ฟเวอร์ให้สัมพันธ์กับค่าที่อยู่ในไคลเอ็นต์ ตัวอย่างเช่น ในไฟล์ greeting.jsp คุณสมบัติ userNumber ของ UserNumberBean จะถูกเซ็ตให้เป็นค่าในคอมโพเนนต์ userNo

2.9.1.5 Invoke application phase

จาวาเซิร์ฟเวอร์เฟสซ์จะทำการอิมพลีเมนท์ในส่วนของ Application-level event อย่างเช่น การ submit ฟอร์มหรือการ link ไปที่หน้าอื่น ในไฟล์ greeting.jsp จะมี Application-level event ที่เชื่อมโยงเข้ากับคอมโพเนนต์ UICommand โดยปกติแล้ว action listener จะทำการอิมพลีเมนท์โดยการรับค่าเอาต์คัมจากแอตทริบิวต์แอคชั่นของคอมโพเนนต์ UICommand จากนั้น listener จะส่งผ่านค่าเอาต์คัมไปที่ NavigtionHandler โดยที่ NavigtionHandler จะทำการmatche ไปที่ navigation rule ที่เหมาะสมที่ถูกกำหนดไว้ในไฟล์ Application configuration ของแอพพลิเคชั่นเพื่อกำหนดว่าจะให้หน้าเพจไหนแสดงต่อไป จากนั้นจาวาเซิร์ฟเวอร์เฟสซ์จะทำการเซ็ต response ไปที่หน้าเพจที่แสดงขึ้นมาใหม่

2.9.1.6 Render response phase

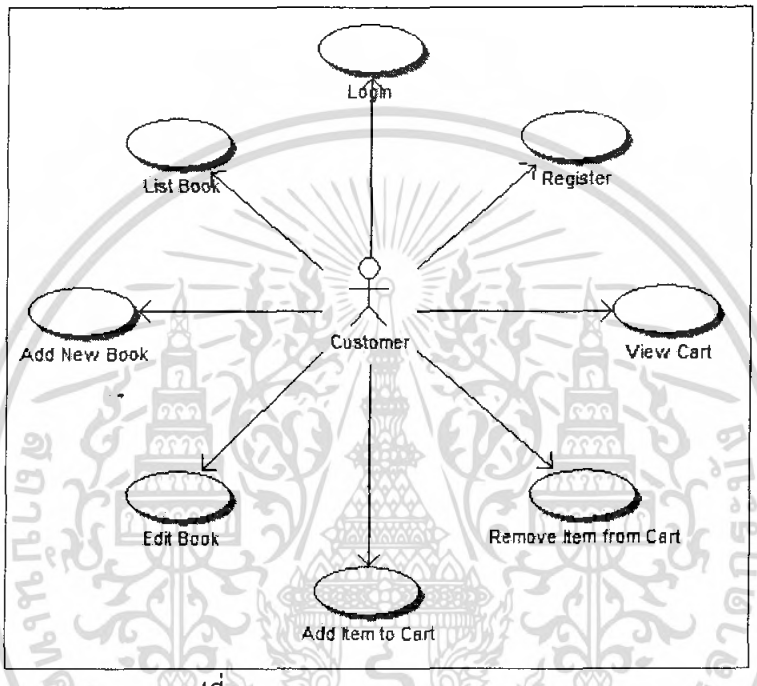
จาวาเซิร์ฟเวอร์เฟสซ์จะทำการอิมพลีเมนท์ในส่วนของ built-in encode method และทำการเรนเดอร์คอมโพเนนต์ จากคอมโพเนนต์ tree ที่ถูกบันทึกไว้ใน FacesContext ซึ่งจะเป็นการสร้างฟอร์มขึ้นมาจากคอมโพเนนต์ tree แต่ถ้ามีข้อผิดพลาดในขั้นตอนก่อนหน้านี้จะทำการแสดงหน้าเดิมขึ้นมาพร้อมกับ error message ที่อยู่ใน FacesContext สถานะของ response จะถูกบันทึก subsequent request จากนั้นจะไปทำงานในขั้นตอนของ restore view

บทที่ 3

การออกแบบ

3.1 แผนภาพ Use case diagram

การออกแบบระบบของ BookStore Online ด้วย Use case diagram สามารถบอกความสัมพันธ์ระหว่าง Customer และ action ที่อนุญาตให้ Customer กระทำได้เมื่อเข้ามาใช้งาน



รูปที่ 3.1 แผนภาพ Use case diagram

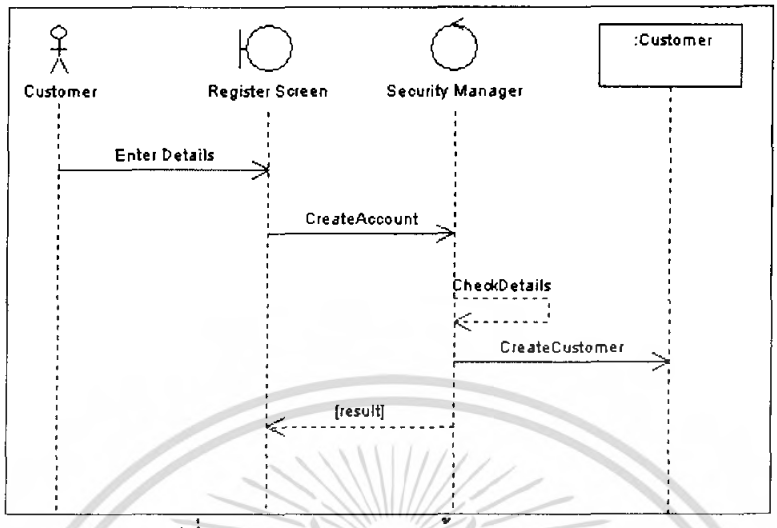
จากรูปที่ 3.1 แสดงแผนภาพ Use case diagram ที่บอกถึงภาพรวมของระบบที่ Customer เข้ามาใช้งานระบบ BookStore Online โดย Customer สามารถ Login, Register, View Cart, Remove Item from Cart, Add Item to Cart และ List Book ได้ ในส่วน Add New Book กับ Edit Book จะเป็นหน้าที่ของ admin ที่สามารถกระทำได้

3.2 แผนภาพ Sequence diagram

Sequence diagram ได้อธิบายถึงลำดับการทำงานของระบบ BookStore Online ในแต่ละขั้นตอนที่ Customer เข้ามาใช้งานระบบ โดยแบ่งออกเป็น 3 ส่วนหลักดังนี้

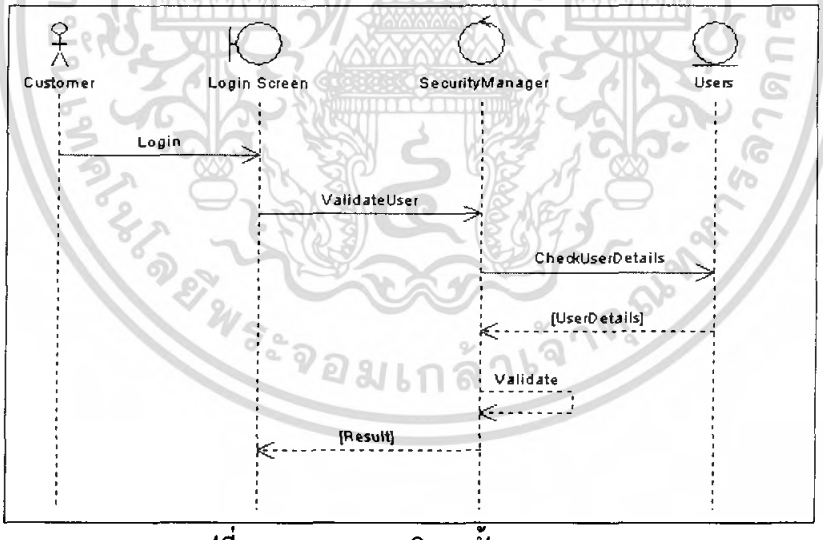
3.2.1 User Management

ในส่วนนี้เป็นส่วนที่ใช้จัดการกับ Customer เมื่อเข้ามาที่ Home Page



รูปที่ 3.2 แผนภาพอธิบายขั้นตอน Register

จากรูปที่ 3.2 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนการ Register เป็น Customer ไว้กับทางร้าน ซึ่งเมื่อเป็น Customer แล้วจะมีสิทธิ์สั่งซื้อหนังสือได้

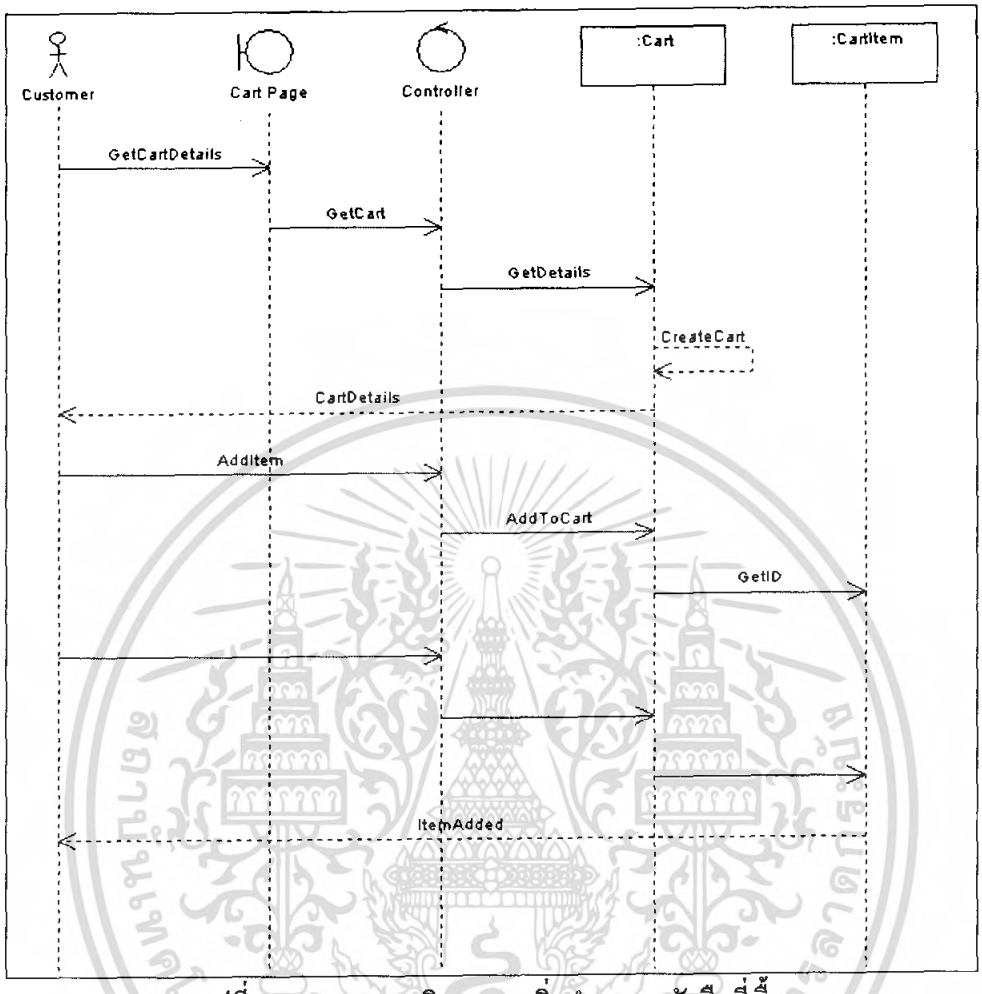


รูปที่ 3.3 แผนภาพอธิบายขั้นตอน Login

จากรูปที่ 3.3 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนการ Login ของ Customer โดยจะมีการตรวจสอบ Username Password ว่าถูกต้องหรือไม่ ถ้าระบบตรวจสอบแล้วถูกต้องจะอนุญาตให้ Login เข้ามาใช้งานระบบได้

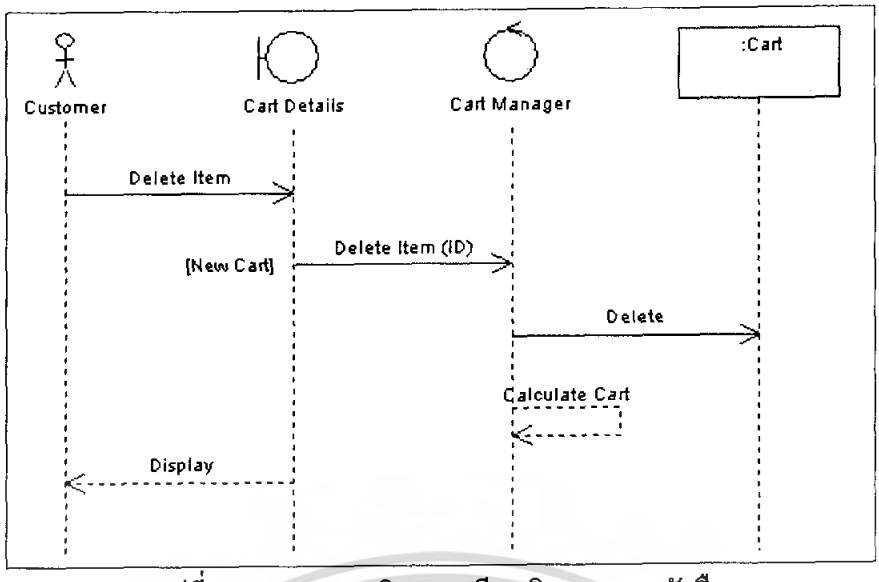
3.2.2 Manage Order

ในส่วนนี้เป็นส่วนที่ใช้จัดการเรื่องรายการซื้อหนังสือที่จะนำไปไว้ในตะกร้ารถเข็น



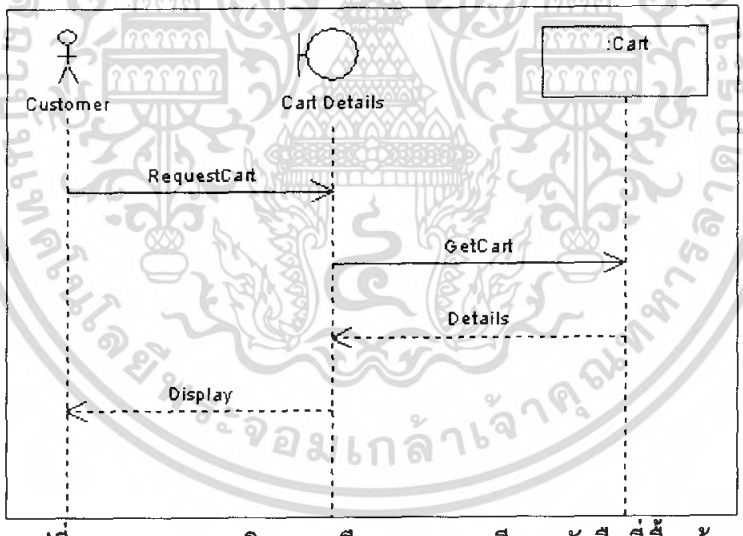
รูปที่ 3.4 แผนภาพอธิบายการเพิ่มจำนวนหนังสือที่ซื้อ

จากรูปที่ 3.4 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนการเพิ่มจำนวนหนังสือที่ซื้อในตะกร้ารถเข็น ซึ่งจะไปตรวจสอบยอดรวมหนังสือในตะกร้ารถเข็นมาก่อนแล้วจึงเพิ่มรายการหนังสือที่ซื้อใหม่เข้าไป



รูปที่ 3.5 แผนภาพอธิบายกรณียกเลิกรายการหนังสือ

จากรูปที่ 3.5 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนการยกเลิกรายการหนังสือบางรายการที่มีอยู่ในตะกร้ารถเข็น โดยในขั้นตอนนี้จะเป็นการ delete รายการหนังสือออกจากตะกร้ารถเข็นพร้อมคำนวณราคารวมเงินใหม่ให้ด้วย

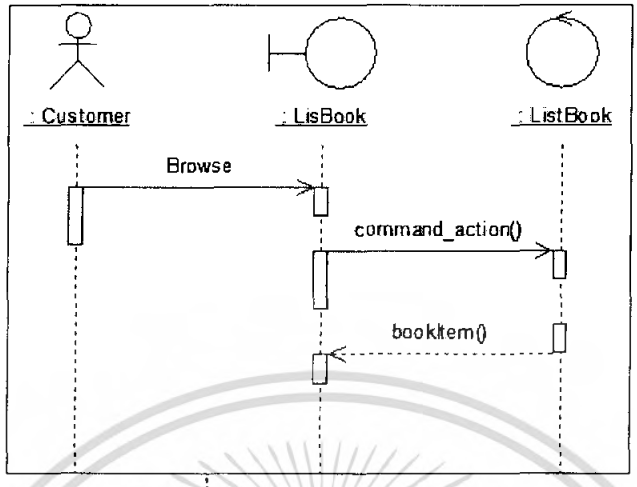


รูปที่ 3.6 แผนภาพอธิบายกรณีแสดงรายละเอียดหนังสือที่ซื้อแล้ว

จากรูปที่ 3.6 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนที่ใช้แสดงรายละเอียดของการซื้อหนังสือที่มีอยู่ในตะกร้ารถเข็นแล้ว โดยจะเป็นการไปนำรายการหนังสือที่เก็บไว้ในตะกร้ารถเข็นมาแสดงให้กับ Customer

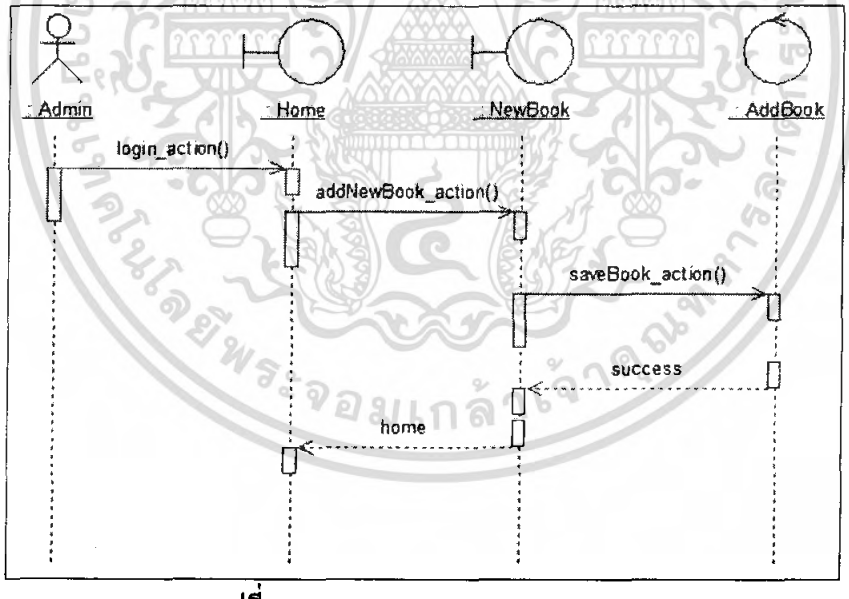
3.2.3 Manage Book

ในส่วนนี้เป็นส่วนที่จัดการเกี่ยวกับข้อมูลของหนังสือแสดงที่ Home Page



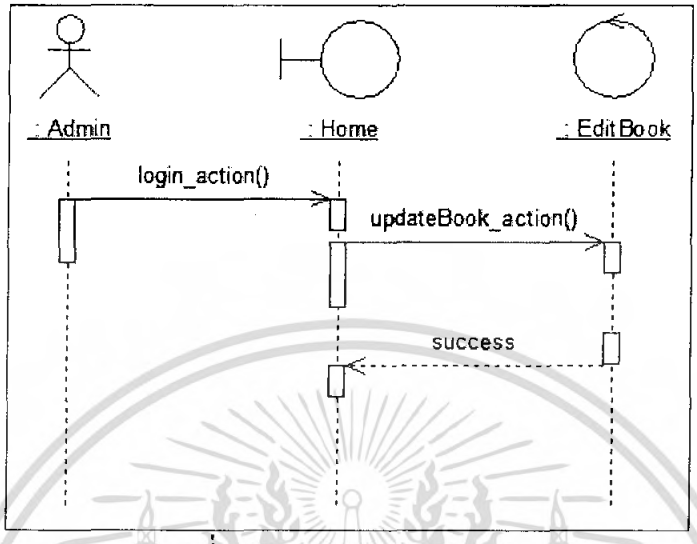
รูปที่ 3.7 แผนภาพ List Book

จากรูปที่ 3.7 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนในการแสดงรายการหนังสือที่มีจำหน่ายในร้าน โดย Customer จะเรียกดูรายการหนังสือทั้งหมดเพื่อเลือกซื้อได้



รูปที่ 3.8 แผนภาพการ Add New Book

จากรูปที่ 3.8 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนในการเพิ่มจำนวนหนังสือใหม่เข้าไปในรายการหนังสือของร้าน โดย Admin เท่านั้นที่สามารถกระทำได้ หลังจากที่มีการ create รายการหนังสือใหม่แล้วก็จะทำการเพิ่มเข้าในฐานข้อมูลรายการหนังสือ

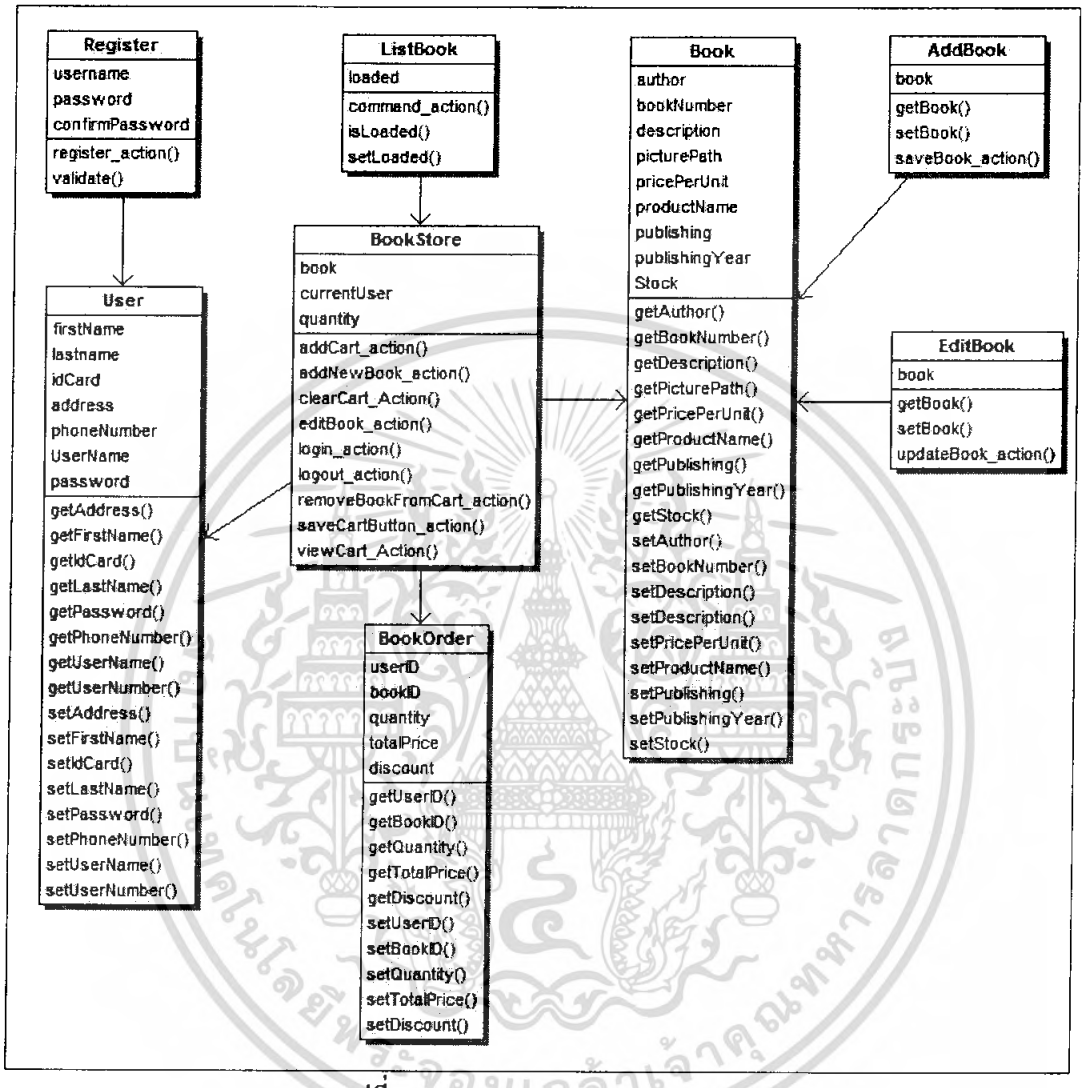


รูปที่ 3.9 แผนภาพการ Edit Book

จากรูปที่ 3.9 แสดงแผนภาพ Sequence diagram ที่อธิบายลำดับขั้นตอนการแก้ไขข้อมูลรายการหนังสือที่แสดงใน List Book โดย Admin เท่านั้นที่สามารถกระทำได้ สำหรับการแก้ไขนี้จะเป็นการ update รายการหนังสือที่มีอยู่ในฐานข้อมูลเดิม

3.3 แผนภาพ Class diagram

Class diagram ประกอบด้วย Class และความสัมพันธ์ระหว่าง Class ที่สามารถแสดงรายละเอียดได้ว่ามีเมธอดและแอตทริบิวต์อย่างไร



รูปที่ 3.10 แผนภาพ Class diagram

จากรูปที่ 3.10 แสดงแผนภาพ Class diagram ที่ระบบของ BookStore Online โดยประกอบไปด้วยคลาสต่างๆ ต่อไปนี้

1. Book จะเป็นคลาสที่สร้างขึ้นมาเพื่อใช้ติดต่อกับฐานข้อมูล โดยที่แอตทริบิวต์แต่ละตัวของคลาสจะถูกผูกติดเข้าไปกับฟิลด์ต่างๆ ของตารางที่อยู่ในฐานข้อมูล ซึ่งในการอ่านและเขียนข้อมูลนั้นจะทำผ่านทางเมธอด get และ set ที่กำหนดไว้

2. User เป็นคลาสที่จัดการกับข้อมูลของยูสเซอร์ โดยการทำงานจะเป็นลักษณะเดียวกันกับคลาส Book

3. BookOrder เป็นคลาสที่ใช้เก็บข้อมูลการซื้อ-ขาย โดยการทำงานจะเป็นลักษณะเดียวกันกับคลาส Book และ User

4. AddBook เป็นคลาสที่ใช้สำหรับจัดการเกี่ยวกับการเพิ่มรายการหนังสือเข้าไปในฐานข้อมูล

5. EditBook เป็นคลาสที่ใช้สำหรับจัดการเกี่ยวกับการแก้ไขรายการหนังสือที่อยู่ในฐานข้อมูล

6. BookStore เป็นคลาสที่ใช้จัดการระบบซื้อ-ขายหนังสือในตะกร้ารถเข็น

7. Register เป็นคลาสสำหรับใช้ลงทะเบียนสมาชิก

8. ListBook เป็นคลาสสำหรับใช้แสดงรายการหนังสือ

บทที่ 4

ผลการทดลองและทดสอบ

4.1 การทดลองส่วนของ UI Component

ในส่วนต่อไปจะแสดงตัวอย่างของคอมโพเนนต์ประเภทต่างๆ ซึ่งจะแบ่งออกเป็นประเภทตามลักษณะการใช้งาน

ในตารางที่ 4.1 และ 4.2 เป็นตัวอย่างของคอมโพเนนต์ที่ใช้สำหรับรับข้อมูลมาจากผู้ใช้งาน ซึ่งได้แก่ h:inputText, h:inputSecret และ h:inputTextarea โดยที่คอมโพเนนต์ที่แสดงออกมาจะมีรูปร่างหน้าตาแตกต่างกันไป ขึ้นอยู่กับการกำหนดค่าให้กับแอตทริบิวต์ต่างๆของคอมโพเนนต์แต่ละตัว

ส่วนที่เป็นความแตกต่างระหว่าง h:inputText กับ h:inputSecret คือ h:inputSecret จะซ่อนอินพุตที่ยูสเซอร์ป้อนเข้าไปไม่ให้มองเห็น ซึ่งอินพุตประเภทนี้ได้แก่ รหัสผ่าน ในตารางที่ 4.1 แถวที่ 5 เป็นตัวอย่างของ h:inputText ที่กำหนดค่าของแอตทริบิวต์ value ให้เท่ากับ "1234567" และ size เป็น "5" ซึ่งผลลัพธ์ที่ได้คอมโพเนนต์จะเป็นดังรูปที่อยู่ทางซ้ายมือ

ตารางที่ 4.1 ตัวอย่าง h:inputText และ h:inputSecret

Example	Result
<h:inputText value="#{form.testString}" readonly="true"/>	12345678901234567890
<h:inputSecret value="#{form.passwd}" redisplay="true"/>	**********
<h:inputSecret value="#{form.passwd}" redisplay="false"/>	
<h:inputText value="inputText" style="color: Yellow; background: Teal;"/>	inputText
<h:inputText value="1234567" size="5"/>	123456
<h:inputText value="1234567890" maxlength="6" size="10"/>	123456

ส่วนที่เป็นความแตกต่างระหว่าง h:inputText กับ h:inputTextarea คือ h:inputText จะรับอินพุตได้แค่บรรทัดเดียว ส่วน h:inputTextarea นั้นจะสามารถรับข้อมูลได้มากกว่า 1 บรรทัด โดยการกำหนดค่าให้กับแอตทริบิวต์ rows และ cols ดังตารางที่ 4.2

ตารางที่ 4.2 ตัวอย่าง h:inputTextarea

Example	Result
<h:inputTextarea rows="5"/>	
<h:inputTextarea cols="5"/>	
<h:inputTextarea value="123456789012345" rows="3" cols="10"/>	456789012345
<h:inputTextarea value="#{form.dataInRows}" rows="2" cols="15"/>	line one line two line three

คอมโพเนนต์ประเภทต่อไปจะเป็นคอมโพเนนต์ที่ใช้สำหรับแสดงข้อมูลให้กับยูสเซอร์ ซึ่งที่ข้อมูลนั้น อาจจะเป็นข้อความหรือรูปภาพคอมโพเนนต์เหล่านี้ได้แก่ h:outputText และ h:graphicImage โดยแสดงตัวอย่างในตารางที่ 4.3

ตารางที่ 4.3 ตัวอย่าง h:outputText และ h:graphicImage

Example	Result
<h:outputText value="#{form.testString}"/>	12345678901234567890
<h:outputText value="Number #{form.number}"/>	Number 1000
<h:outputText value="<input type='text' value='hello'/>"/>	hello
<h:outputText escape="true" value="<input type='text' value='hello'/>"/>	<input type="text" value="hello">
<h:graphicImage value="/tjefferson.jpg"/>	
<h:graphicImage value="/tjefferson.jpg" style="border: thin solid black"/>	

ส่วนต่อไปจะเป็นคอมโพเนนต์ h:commandButton โดยคอมโพเนนต์ตัวนี้จะทำการตรวจจับ event ที่เกิดขึ้นกับตัวมันโดยใช้ event handle จากนั้นก็จะทำการ invokes ไปยัง action method ที่ได้กำหนดไว้ในแอตทริบิวต์ action โดยตัวอย่างของคอมโพเนนต์ h:commandButton แสดงในตารางที่ 4.4

ตารางที่ 4.4 ตัวอย่าง h:commandButton

Example	Result
<h:commandButton value="submit" type="submit"/>	submit
<h:commandButton value="reset" type="reset"/>	reset
<h:commandButton value="click this button..." onclick="alert('button clicked')" type="button"/>	click this button to execute JavaScript
<h:commandButton value="disabled" disabled="#{not form.buttonEnabled}"/>	disabled
<h:commandButton value="#{form.buttonText}" type="reset"/>	press me

ส่วนต่อไปเป็น h:commandLink ซึ่งจะทำการสร้าง link ที่มีลักษณะคล้ายกับ action from ของ html ซึ่งแสดงตัวอย่างในตารางที่ 4.5

ตารางที่ 4.5 ตัวอย่าง h:commandLink

Example	Result
<h:commandLink> <h:outputText value="register"/> </h:commandLink>	register
<h:commandLink style="font-style: italic"> <h:outputText value="#{msgs.linkText}"/> </h:commandLink>	*click here to register*
<h:commandLink> <h:outputText value="#{msgs.linkText}"/> <h:graphicImage value="/registration.jpg"/> </h:commandLink>	 click here to register
<h:commandLink value="welcome" actionListener="#{form.useLinkValue}" action="#{form.followLink}">	welcome
<h:commandLink> <h:outputText value="welcome"/> <f:param name="outcome" value="welcome"/> </h:commandLink>	welcome

ในส่วนของ h:outputLink จะมีลักษณะคล้ายกับ h:commandLink ตรงที่จะสร้าง HTML anchor element แต่จะไม่เหมือนกับ h:commandLink ตรงที่ h:outputLink จะไม่สร้าง JavaScript เพื่อให้ทำงานเหมือนกับ submit button แสดงตัวอย่างในตารางที่ 4.6

ตารางที่ 4.6 ตัวอย่าง h:outputLink

Example	Result
<h:outputLink value="http://java.net"> <h:graphicImage value="java-dot-net.jpg"/> <h:outputText value="java.net"/> </h:outputLink>	 java.net
<h:outputLink value="#{form.welcomeURL}"> <h:outputText value="#{form.welcomeLinkText}"/> </h:outputLink>	go to welcome page
<h:outputLink value="#introduction"> <h:outputText value="Introduction" style="font-style: italic"/> </h:outputLink>	*Introduction*
<h:outputLink value="#conclusion" title="Go to the conclusion"> <h:outputText value="Conclusion"/> </h:outputLink>	Conclusion Go to the conclusion
<h:outputLink value="#toc" title="Go to the table of contents"> <f:verbatim> Table of Contents </f:verbatim> </h:outputLink>	**Table of Contents**

4.2 การทดลองส่วนของ Renderer

เพจ HTML คือการ Render สำหรับ tag h:form, h:inputText, h:inputSecret และ h:commandButton จะถูก convert เป็น HTML ดังแสดงในรูปที่ 4.1

รูปที่ 4.1 แสดงเพจที่มาจาก tag ของ JSF Render เป็น HTML แล้ว

4.3 การทดลองส่วนของ Validation

เป็นการตรวจสอบอินพุตที่ป้อนเข้ามาว่าตรงกับข้อมูลที่ต้องการหรือไม่ สำหรับในไฟล์ index.jsp สามารถแสดงโค้ดโปรแกรมได้ดังนี้ โดยที่ required= "true" หมายถึง กรณีที่ไม่ป้อนอินพุตใดๆ ให้แสดง errorMessage ด้วยและไม่ ต้อง process ต่อ

```
<h:outputText value="#{msgs.creditCard}"/>
<h:inputText id="card" value="#{payment.card}" required="true">
  <f:validateLength minimum="13"/>
</h:inputText>
<h:message for="card" styleClass="errorMessage"/>
```

โปรแกรมที่ 4.1 ตัวอย่างการทำ Validation

ตัวอย่างในรูปที่ 4.2 ได้กำหนดให้ขนาดของหมายเลข Credit Card จะต้องมีความยาวไม่เกิน 13 หลัก ถ้าหากมีการป้อนอินพุตเกินที่กำหนดไว้จะมีข้อความขึ้นมาเตือน

รูปที่ 4.2 แสดงการป้อนตัวเลขน้อยกว่า 13 ตัวอักษร

หลังจากที่ป้อนอินพุตได้ถูกต้อง ในที่นี้ Credit Card มีตัวเลขถูกป้อนเข้าไปครบ 13 หลัก เมื่อกดปุ่ม process จะไปยังเพจใหม่ ซึ่งได้จัดรูปแบบที่ถูกต้องตามต้องการแล้ว ดังแสดงในรูป 4.3

รูปที่ 4.3 แสดงการป้อนอินพุตที่ถูกต้องและทำการจัดรูปแบบแล้ว

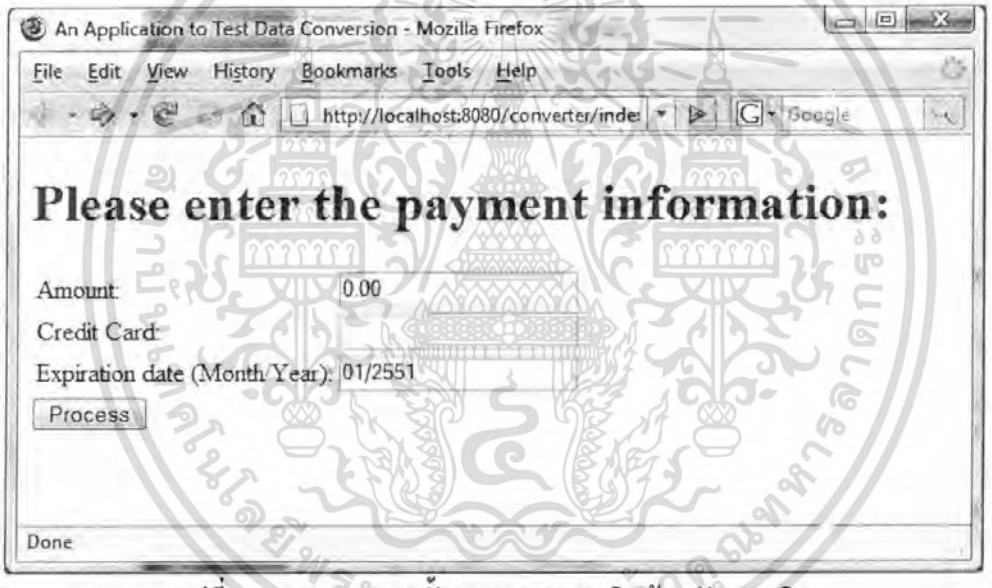
4.4 การทดลองส่วนของ Conversion

Conversion เป็นกระบวนการที่จะเปลี่ยนแปลงสตริงที่รับมาจากฟอร์มไปเป็นข้อมูลชนิดอื่น โดยขั้นตอนของการเปลี่ยนแปลงนั้นจะยังไม่ถูกทำโดยทันที แต่จะถูกส่งไปจัดการโดย Business logic แทน โดยทีแรกค่าต่างๆ จะถูกเก็บไว้ในอ็อปเจ็คของคอมโพเนนต์

ตัวอย่างต่อไปนี้จะประกอบไปด้วยกระบวนการ Conversion โดยเว็บแอพพลิเคชั่นนี้จะเป็นตัวอย่างการแสดงขั้นตอนของการชำระเงินด้วยบัตรเครดิต โดยการชำระเงินจะประกอบไปด้วยข้อมูลต่อไปนี้

- จำนวนเงิน
- หมายเลขบัตร
- วันหมดอายุของบัตร

โดยรายละเอียดต่างๆ แสดงในรูปที่ 4.4



รูปที่ 4.4 แสดงเพจของขั้นตอนการชำระเงินด้วยบัตรเครดิต

เราจะทำการเพิ่มขั้นตอนของการ Conversion ไปที่ text field อันแรก โดยค่าที่รับเข้ามานั้นจะต้องมีอย่างน้อย 2 หลัก โดยทำได้ดังนี้

```
<h:inputText value="#{payment.amount}">
    <f:convertNumber minFractionDigits="2"/>
</h:inputText>
```

โปรแกรมที่ 4.2 ตัวอย่างการทำ Conversion number

โดยที่ f:convertNumber เป็น converter ตัวหนึ่งที่อยู่ใน standard converter ที่จะถูกจัดการโดย JSF ในส่วนของ text field ตัวที่สองจะไม่มีการใช้ converter และใน text field ตัวที่สามจะใช้ f:convertDateTime เป็น converter โดยที่รูปแบบของแอตทริบิวต์ที่ใช้ในการเซ็ตสตริงจะเป็นสตริง MM/yyyy

```
<h:inputText value="#{payment.date}">
    <f:convertDateTime pattern="MM/yyyy"/>
</h:inputText>
```

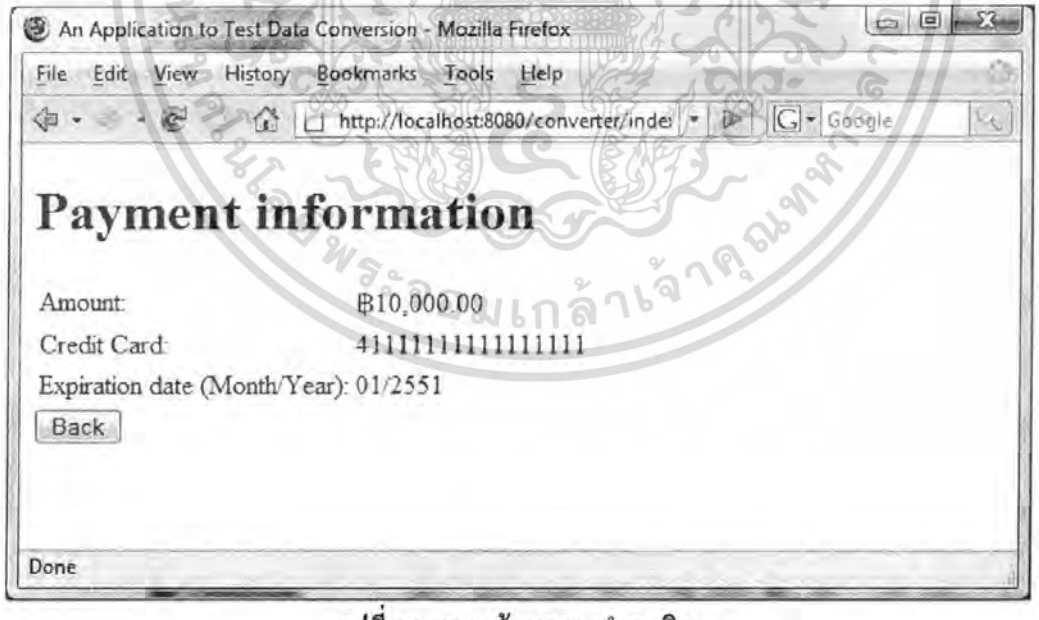
โปรแกรมที่ 4.3 ตัวอย่างการทำ Conversion Date Time

ในไฟล์ result.jsp จะแสดงค่าอินพุตที่ยูสเซอร์ป้อนเข้าไป โดยจะมีความแตกต่างกันไปขึ้นอยู่กับสกุลเงิน แสดงดังโค้ดต่อไปนี้

```
<h:outputText value="#{payment.amount}">
    <f:convertNumber type="currency"/>
</h:outputText>
```

โปรแกรมที่ 4.4 ตัวอย่างการทำ Conversion number

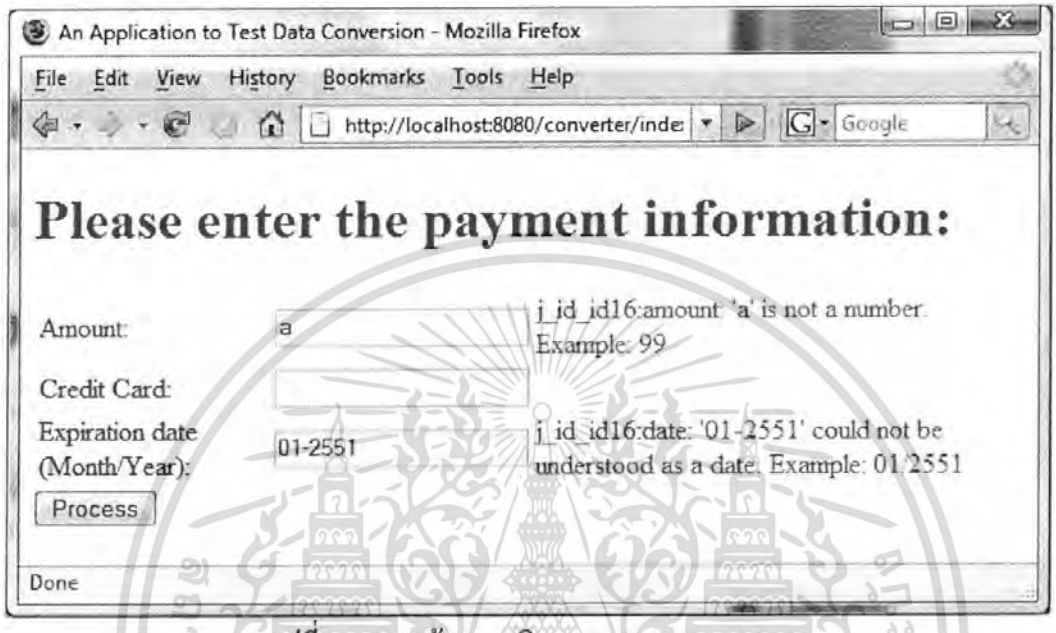
Converter จะจัดการกับสัญลักษณ์ที่ใช้แทนสกุลเงินของแต่ละประเทศดังแสดงในรูปที่ 4.5



รูปที่ 4.5 แสดงข้อมูลการชำระเงิน

เมื่อ conversion เกิดข้อผิดพลาดขึ้นมาจะมีขั้นตอนต่างๆ เกิดขึ้นดังต่อไปนี้

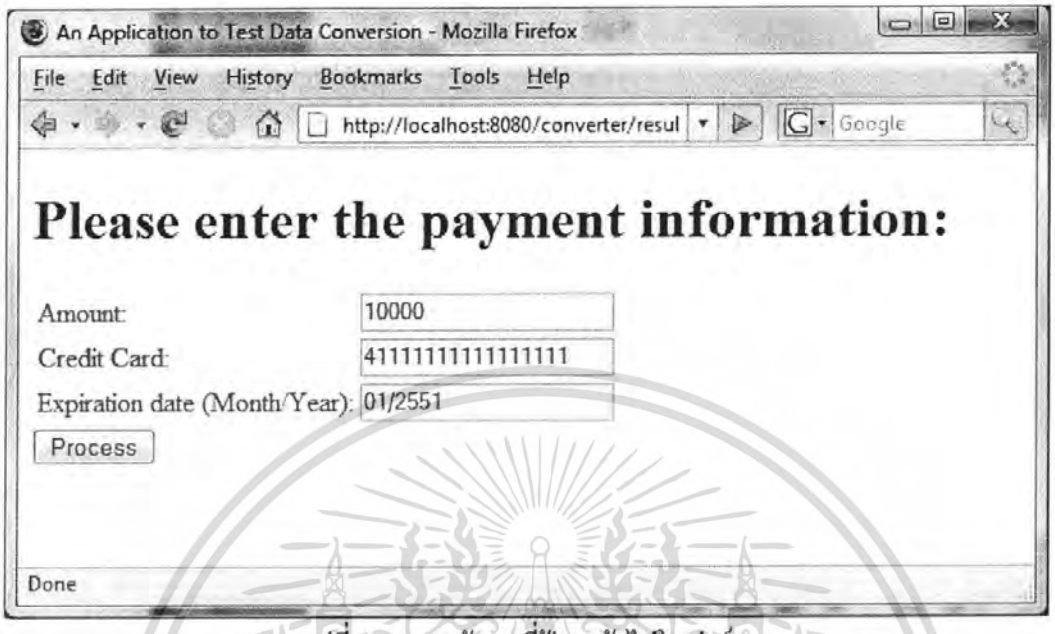
- คอมโพเนนต์ต่างๆ ที่เกิด conversion failed จะทำการแสดงข้อความผิดพลาดขึ้นมา
- JSF จะทำการ redisplays เพจปัจจุบันทันทีหลังจากที่มีการประมวลผลในขั้นตอน Validations เรียบร้อยแล้ว



รูปที่ 4.6 แสดงข้อความผิดพลาดของ conversion

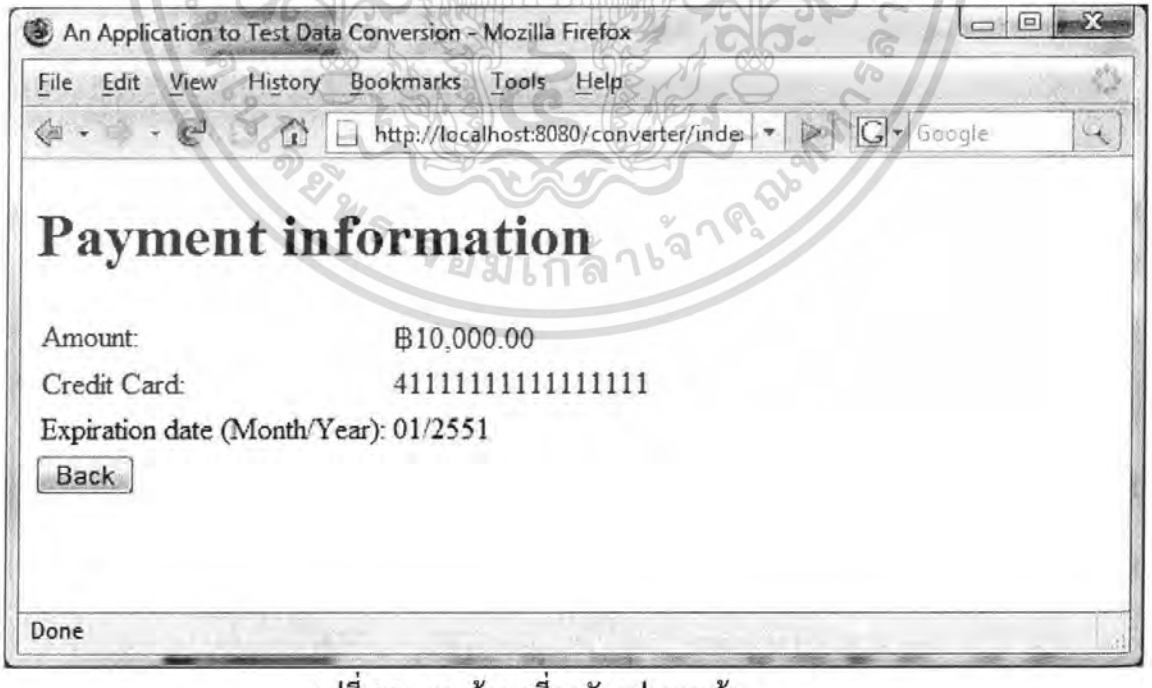
จากรูปที่ 4.6 จะเป็นตัวอย่างการรันโปรแกรม โดยที่ข้อมูลที่ป้อนเข้าไปใน text field จะเป็นข้อมูลที่รูปแบบ(format)ไม่ตรงกับที่กำหนดไว้ใน conversion ทำให้คอมโพเนนต์แสดงข้อความผิดพลาดขึ้นมา

ในตัวอย่างต่อไปจะเป็นการป้อนข้อมูลที่มีรูปแบบที่ถูกต้องเข้าไป โดยข้อมูลที่ป้อนเข้าไปจะแสดงดังรูปที่ 4.7



รูปที่ 4.7 แสดงข้อมูลที่ป้อนเข้าไปในฟอร์ม

หลังจากที่ทำการกดปุ่ม Process แล้ว ข้อมูลที่อยู่ในฟอร์มจะถูกจัดรูปแบบใหม่แล้วนำไปแสดงในเพจถัดไป ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 แสดงข้อมูลที่ถูกจัดรูปแบบแล้ว

จากการทดลองจะเห็นว่า conversion ของ JSF มีประโยชน์มากในการช่วยตรวจสอบรูปแบบของข้อมูลที่ป้อนเข้ามาในฟอร์ม ก่อนที่จะนำข้อมูลเหล่านั้นไปเก็บไว้ในอ็อปเจ็ค ซึ่งข้อมูลที่ป้อนเข้าไปในฟอร์มจะต้องมีรูปแบบ(format)ที่ตรงกับอ็อปเจ็คที่ได้กำหนดไว้แล้ว

4.5 การทดลองส่วนของ Navigation

Navigation จะเกิดขึ้นเมื่อยูสเซอร์พยายามจะสวิตช์จากเพจหนึ่งไปยังอีกเพจหนึ่ง โดยการคลิกที่ปุ่มหลังจากที่ป้อนอินพุตเข้าไป การคลิกที่ hyperlink หรือป้อน URL ของเว็บแอพพลิเคชันที่ต้องการเข้าถึงโดยตรงไปที่ web browser ไม่ว่ากรณีใดก็ตาม เพจต่อไปที่จะแสดงออกมาหรือตอบสนองสำหรับหน้าเพจปัจจุบันจะต้องถูกจัดการโดยเว็บแอพพลิเคชัน ซึ่งในกรณีของเทคโนโลยี Java Servlets เมื่อ web browser ทำการร้องขอมาที่ Servlets โดยการพิมพ์ URL ของ Servlets ไปที่ web browser เมธอด doGet() จะถูกร้องขอ และจะทำการตอบสนองกลับที่ web browser

ลองจินตนาการว่าเมื่อ Servlets มีการตอบสนองกับไปที่ยูสเซอร์ และยูสเซอร์ป้อนข้อมูลบางอย่างเข้าไป เพื่อที่จะนำข้อมูลที่ได้มาแสดงที่หน้าถัดไป ซึ่งทั้งหมดนี้คือกลไกการทำงานทั้งหมดของ Navigation ในส่วนของเว็บแอพพลิเคชัน แต่ในความเป็นจริงแล้ว Navigation ควรจะมีการเลือกเพจที่จะนำมาแสดงโดยพิจารณาจากข้อมูลที่ได้ทำการรับเข้ามา เช่น ในกรณีของการล็อกอิน ข้อมูลที่ป้อนเข้าไปจะต้องมีความถูกต้องจึงจะสามารถเข้าสู่ระบบได้ แต่ถ้าข้อมูลไม่ถูกต้องก็ให้แสดงข้อความขึ้นมา

Types of Navigation

JSF จะมี Navigation model อยู่ด้วยกัน 2 ชนิดคือ

- Static Navigation
- Dynamic Navigation

4.5.1 Static Navigation

มักจะใช้ในกรณีที่รู้ล่วงหน้าว่าจะให้เพจไหนมาแสดงต่อไป ซึ่ง Navigation จะสามารถเลือกเพจขึ้นมาแสดงต่อไปได้ทันที ตัวอย่างเช่น ระบบล็อกอิน web browser

The Login Page

ในไฟล์ login.jsp จะประกอบไปด้วยอินพุตฟิลด์ สำหรับรับชื่อยูสเซอร์กับ password และมีปุ่มที่ใช้สำหรับส่งข้อมูลที่อยู่ในฟิลด์ไปยังเซิร์ฟเวอร์ โดยจะใช้แอตทริบิวต์ action ของคอมโพเน็นต์ 'command button' ที่กำหนดค่าให้เป็น 'loginWelcome' เพื่อใช้เป็น logic ในการบอก Navigation ให้รู้ว่าจะต้องไปที่ไฟล์ 'loginWelcome.jsp'

ไฟล์ login.jsp แสดงโค้ดโปรแกรมได้ดังนี้

```
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
<html>
<head><title>Login Application</title></head>
<body>
<h1>Login Application</h1>
<f:view>
<h:form>
   <p>Enter your username:
      <h:inputText  value="#{LoginBean.username}"  id="usernameTextField" required="true"/>
      <h:message for="usernameTextField" />
   </p>
   <p>Enter your password:
      <h:inputSecret
         value="#{LoginBean.password}" id="passwordTextField" required="true"/>
      <h:message for="passwordTextField" />
   </p>
   <h:commandButton value="Submit Values" action="loginWelcome"/>
</h:form>
</f:view>
</body>
</html
```

โปรแกรมที่ 4.5 login.jsp

Login Bean Class

คลาส UserBean จะประกอบไปด้วย แอตทริบิวต์ username กับ password เพื่อใช้สำหรับเก็บ username และ password ที่รับมาจาก ยูสเซอร์โดยทำการแม็ปไปที่ UserBean.username และ UserBean.password โดยใช้ '#{UserBean.username}' และ '#{UserBean.password}'

ไฟล์ LoginBean.java แสดงโค้ดโปรแกรมได้ดังนี้

```
package net.javabeat.articles.jsf.navigation;
public class LoginBean {
   private String username;
   private String password;
   public LoginBean()  { }
   public String getUsername()   {
      return username;
   }
   public void setUsername(String username) {
      this.username = username;
   }
   public String getPassword() {
      return password;
   }
   public void setPassword(String password)  {
      this.password = password;
   }
}
```

โปรแกรมที่ 4.6 LoginBean.java

Faces Configuration File

faces-config.xml จะเป็นไฟล์ที่ใช้สำหรับกำหนดการเข้าถึง LoginBean ผ่านทาง 'managed-bean' ส่วนต่อมาจะเป็นส่วนของระบบ navigation ซึ่งเราจะทำการกำหนด navigation rule สำหรับ login.jsp ผ่านทาง 'navigation-rule' และ 'from-view-id' โดยที่ 'from-view-id' นี้ จะใช้ตรวจสอบ outcome ที่มาจาก login.jsp ส่วนต่อไปคือ 'navigation-case' ใช้ตรวจสอบกรณีที่เป็นไปได้ของ outcome ที่เข้ามาว่าตรงกันที่ได้กำหนดไว้ใน 'from-outcome' หรือไม่ ถ้าตรงก็จะโหลดเพจที่ได้กำหนดไว้ใน 'to-view-id' ขึ้นมา

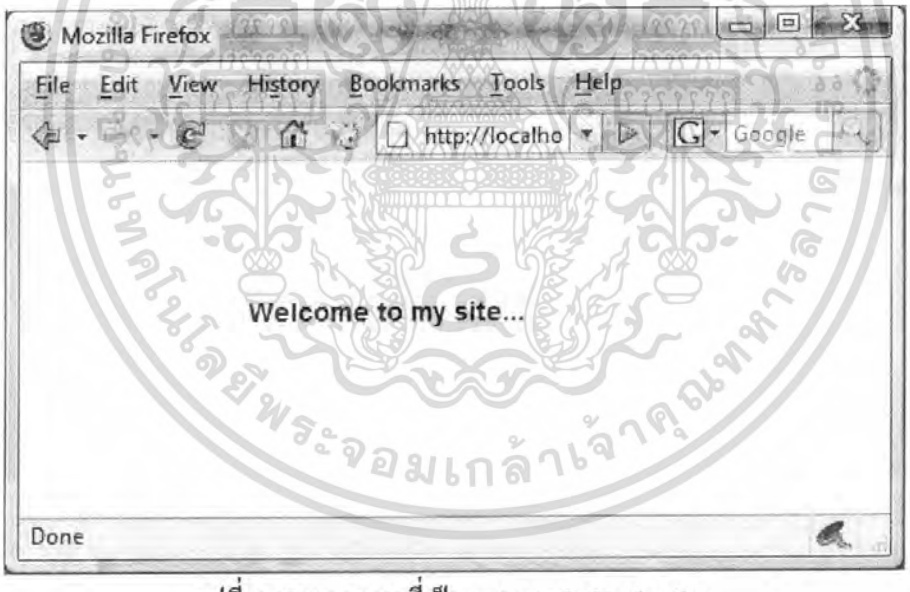
ไฟล์ faces-config.xml แสดงโค้ดโปรแกรมได้ดังนี้

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="1.2"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">
<managed-bean>
   <managed-bean-name>LoginBean</managed-bean-name>
   <managed-bean-class>
      net.javabeat.articles.jsf.navigation.LoginBean</managed-bean-class>
   <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
<navigation-rule>
   <description></description>
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>loginWelcome</from-outcome>
      <to-view-id>/loginWelcome.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

โปรแกรมที่ 4.7 faces-config.xml

สำหรับเพจแบบ Static Navigation ยูสเซอร์ป้อนข้อมูลใดๆ ก็จะไปยังอีกเพจหนึ่งที่ได้กำหนดไว้แล้ว ดังแสดงในรูปที่ 4.9 และ 4.10

รูปที่ 4.9 แสดงเพจการป้อนข้อมูลที่ไปยังเพจต่อไป



รูปที่ 4.10 แสดงเพจที่เป็นแบบ static Navigation

4.5.2 Dynamic Navigation

จากที่ได้กล่าวไปแล้วข้างต้นจะเห็นว่า Navigation ของ JSF ก็ทำงานได้ไม่ต่างกับ Web Application อื่นๆ แต่ที่จะกล่าวต่อไปนี้เป็นส่วนที่โดดเด่นที่สุดของ JSF ซึ่งมีความยืดหยุ่นสูงในการทำงานเมื่อเทียบกับเทคโนโลยีอื่นๆ โดยในตัวอย่างเราจะเพิ่มโค้ดในส่วนข้างล่างนี้เข้าไปใน

ไฟล์ LoginBean.java แสดงโค้ดโปรแกรมได้ดังนี้

```
public String nextPage() {
   if (username.equals("guest") && password.equals("guest"))  {
      return "loginSuccess";
   }
   return "loginFailure";
}
```

โปรแกรมที่ 4.8 LoginBean.java

เมธอด nextPage() จะเป็นตัวกำหนดว่าจะให้เพจใดแสดงออกมาโดยพิจารณาจากค่าของ username และ password ที่รับมาจากยูสเซอร์ โดยถ้าทั้ง username และ password เป็น 'guest' ก็จะถือว่า login ได้สำเร็จ โดยจะส่งสตริง 'loginSucess' มาเป็น logic สำหรับ Navigation และจะส่ง 'loginFailure' ออกมาในกรณีที่ login ไม่สำเร็จ

ไฟล์ Faces Configuration file แสดงโค้ดโปรแกรมได้ดังนี้

```
<navigation-rule>
   <description></description>
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>loginSuccess</from-outcome>
      <to-view-id>/loginSuccess.jsp</to-view-id>
   </navigation-case>
   <navigation-case>
      <from-outcome>loginFailure</from-outcome>
      <to-view-id>/loginFailure.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

โปรแกรมที่ 4.9 faces-config.xml

เราจะมี Navigation case อยู่ 2 กรณีสำหรับ เพจ login.jsp คือในกรณีที่ login สำเร็จ และไม่สำเร็จ โดยถ้า outcome ออกมาเป็น 'loginSuccess' ก็จะแสดง 'loginSuccess.jsp' ออกมา แต่ถ้าเป็น 'loginFailure' ก็จะแสดง 'loginFailure.jsp' ออกมา

ส่วนต่อไปคือไฟล์ 'loginSuccess.jsp' ซึ่งไฟล์นี้จะถูกโหลดขึ้นมาในกรณีที่ login ได้สำเร็จ โดยจะมีการนำเอาค่าที่อยู่ใน LoginBean.java มาแสดงด้วย

ไฟล์ loginSuccess.jsp แสดงโค้ดโปรแกรมได้ดังนี้

```
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
<html>
<head>
<title>Login Successful</title>
</head>
<body>
<h3>Login Successful</h3>
  <P>
    <f:view>
    <h:form>
      <P>You have succesfully logged in.<P/>
      <P>Welcome <h:outputText value = "#{LoginBean.username}" /> <P/>
    </h:form>
    </f:view>
  </P>
</body>
</html>
```

โปรแกรมที่ 4.10 loginSuccess.jsp

ส่วนต่อไปคือไฟล์ 'loginFailure.jsp' ซึ่งไฟล์นี้จะถูกโหลดขึ้นมาในกรณีที่ login ไม่สำเร็จ โดยจะมีการนำเอาค่าที่อยู่ใน LoginBean.java มาแสดงด้วยเช่นกัน

ไฟล์ loginFailure.jsp แสดงโค้ดโปรแกรมได้ดังนี้

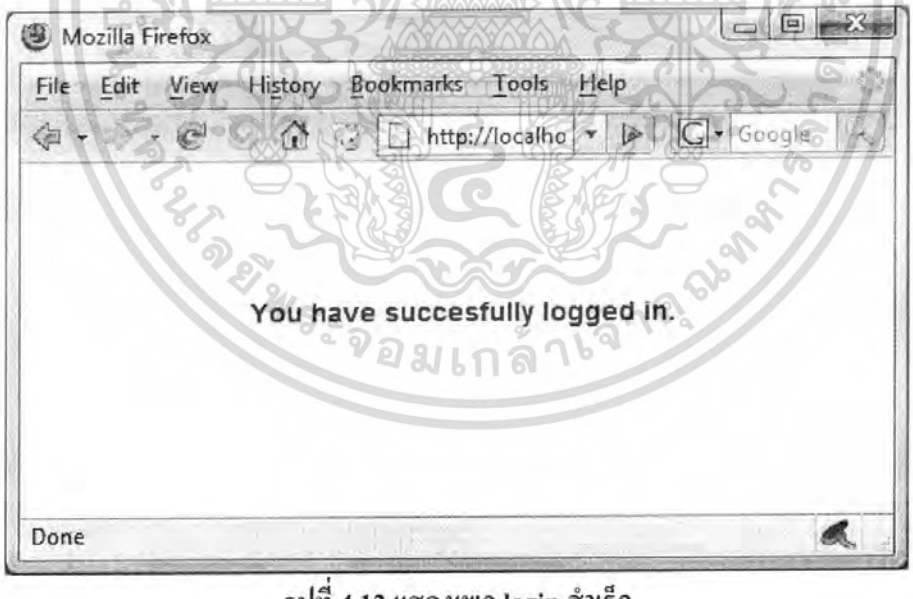
```
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
<html>
<head>
<title>Login Failure</title>
</head>
<body>
<h3>Login Failure</h3>
  <P>
    <f:view>
    <h:form>
    <P>The Login id <h:outputText value = "#{LoginBean.username}" />
      is not found.<P/>
    <P>Please try again. </P>
    </h:form>
    </f:view>
  </P>
</body>
</html>
```

โปรแกรมที่ 4.11 loginFailure.jsp

สำหรับเพจแบบ Dynamic Navigation จะพิจารณาการป้อนข้อมูลจากยูสเซอร์ ในที่นี้ได้ป้อน username password เข้าไป แสดงในรูปที่ 4.11

รูปที่ 4.11 แสดงการป้อนข้อมูลของเพจ Dynamic Navigation

ในกรณีป้อนข้อมูลถูกต้องจะโหลดเพจที่แสดงว่า login สำเร็จ แสดงในรูปที่ 4.12



รูปที่ 4.12 แสดงเพจ login สำเร็จ

ในกรณีป้อนข้อมูลผิดจะโหลดเพจที่แสดงว่า username password ที่ป้อนเข้ามานั้นผิดให้ป้อน username password อีกครั้ง แสดงในรูปที่ 4.13

รูปที่ 4.13 แสดงเพจการ login ผิด

สรุปแล้วก็คือ ระบบ Navigation ของ JSF เป็นระบบที่มีประสิทธิภาพสูง มีความยืดหยุ่นสูงเมื่อเทียบกันระบบอื่นๆ ทำให้ผู้พัฒนามีความสะดวกมียิ่งขึ้น

4.6 การทดลองส่วนของ Event และ Listener

JSF จะตรวจจับ event จากการกดปุ่มบนเพจ ถ้ามี event เกิดขึ้นจะไปเรียกการทำงานจาก backing bean ให้ทำการประมวลผลเพื่อให้ได้ผลลัพธ์กลับไปที่เพจ

ในไฟล์ index.jsp แสดงโค้ดโปรแกรมได้ดังนี้

```
<h:commandButton value="Confirm"
    actionListener="#{UserBean.addConfirmUserListenerAction}"/>
```

โปรแกรมที่ 4.12 index.jsp

ใน backing bean แสดงโค้ดโปรแกรมได้ดังนี้

```
public void addConfirmUserListenerAction(ActionEvent ae) {
        // This method would call a database or other service
        // and add the confirmed user information.
        System.out.println("Adding new User...");
}
```

โปรแกรมที่ 4.13 UserBean.java

จากรายละเอียดที่ได้กล่าวไปแล้วในตอนต้นจะเห็นว่า JavaServer Face ได้จัดเตรียมเครื่องมือที่ใช้สำหรับอำนวยความสะดวกในการพัฒนา Web Application ไว้ให้แล้วพอสมควร ซึ่งเป็นประโยชน์อย่างมากในการพัฒนา Web Application ขนาดใหญ่ที่มีกระบวนการทำงานที่ซับซ้อน ซึ่งถ้าผู้พัฒนาได้มีการศึกษาให้เข้าใจเป็นอย่างดีแล้ว และบวกกับการออกแบบระบบที่ดีพอก็จะช่วยให้ Web Application ที่พัฒนาขึ้นมานั้นทำงานได้อย่างมีประสิทธิภาพ

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทวิจารณ์และสรุป

เทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์เป็นเฟรมเวิร์คที่เข้ามาช่วยให้การจัดการเพจ HTML ได้ง่ายกว่าสำหรับนักพัฒนาโปรแกรม โดยจะเขียนโค้ดในลักษณะที่ใกล้เคียงกับ tag ของ HTML ในไฟล์นามสกุล jsp และเขียนกฎเกณฑ์ไว้ในไฟล์ faces-config.xml และเขียน class ภาษาจาวาที่มี algorithm ไว้เพื่อใช้เป็น Bean ในการทำงานของแอพพลิเคชั่น

5.2 ข้อจำกัด

นักพัฒนาโปรแกรมจำเป็นต้องรู้หลักการของ Object Oriented Programming(OOP) และความหมายของ Tag HTML เพราะจาวาเซิร์ฟเวอร์เฟสซ์ได้ใช้ทั้งสองหลักการในการสร้างเว็บ

5.3 ปัญหาและอุปสรรคที่พบ

ด้วยความใหม่ของเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์ทำให้ยากต่อการค้นหาข้อมูลที่จะบอกรายละเอียดอย่างชัดเจนจึงต้องใช้วิธีการแยกเขียนโค้ดเป็นส่วนๆ เพื่อทำการทดลองศึกษาการทำงานอย่างแท้จริง และการออกแบบเว็บแอพพลิเคชั่นนั้นต้องสอดคล้องกับหลักการของ MVC

5.4 แนวทางการพัฒนาต่อ

ในส่วนของเทคโนโลยีจาวาเซิร์ฟเวอร์เฟสซ์นั้นมีความสามารถเกี่ยวกับการจัดการส่วน presentation ได้ดี แนวทางการพัฒนาต่อควรเป็นเรื่องการจัดการส่วน business logic ที่สามารถสั่งการทำงานด้วย command แบบสั้นได้

ในส่วน web application ของ BookStore Online ควรเป็นการเพิ่มการจัดการจ่ายเงินของ Customer ให้สามารถเลือกจ่ายเงินผ่านธนาคาร หรือ บัตรเครดิต

บรรณานุกรม

Bill Dudney, Jonathan Lehr, Bill Willis, LeRoy Mattingly. **Mastering JavaServer Faces.** :Wiley

David Geary, Cay HorstMann. **Core JavaServer Faces.** :Addison Wesley

H.M.Deitel, P.J.Deitel. **Java How To Program 7 Edition.** :Prentice-Hall Internation,c2003

Kito D.Mann Foreword by Ed Burns. **JavaServer Faces In Action.** :Manning

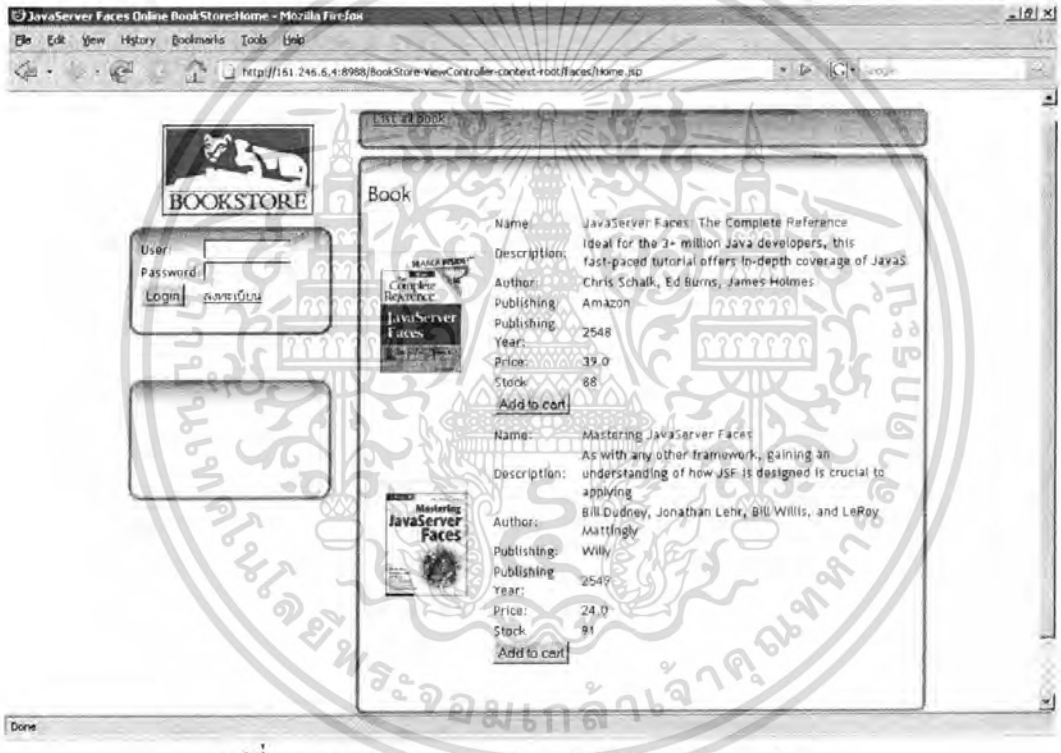
Sang Shin. 2549. **JavaTM EE (J2EE) Programming.** [Online].Available :http://www.javapassion.com/

Java Technology. 1994. **JavaServer Faces Technology.** [Online].Available :http://java.sun.com/

ภาคผนวก ก
คู่มือการใช้งานโปรแกรม

วิธีการลงทะเบียน(Register)

User ที่ไม่ได้เป็น Customer จะต้องทำการ Register เป็น Customer ก่อนเพื่อที่จะซื้อหนังสือได้ โดย Click "ลงทะเบียน" จะเข้าสู่อีก page หนึ่ง ซึ่งเป็น page สำหรับกรอกข้อมูลส่วนบุคคลพร้อมทั้งกำหนด User name และ Password ที่จะใช้ login เข้าสู่ระบบ เมื่อ Submit แล้วจะเก็บข้อมูลลงสู่ฐานข้อมูล หลังจากนั้นจะได้สิทธิ Customer ที่เป็นสมาชิกของร้าน ดังรูปที่ ก.1 และ ก.2

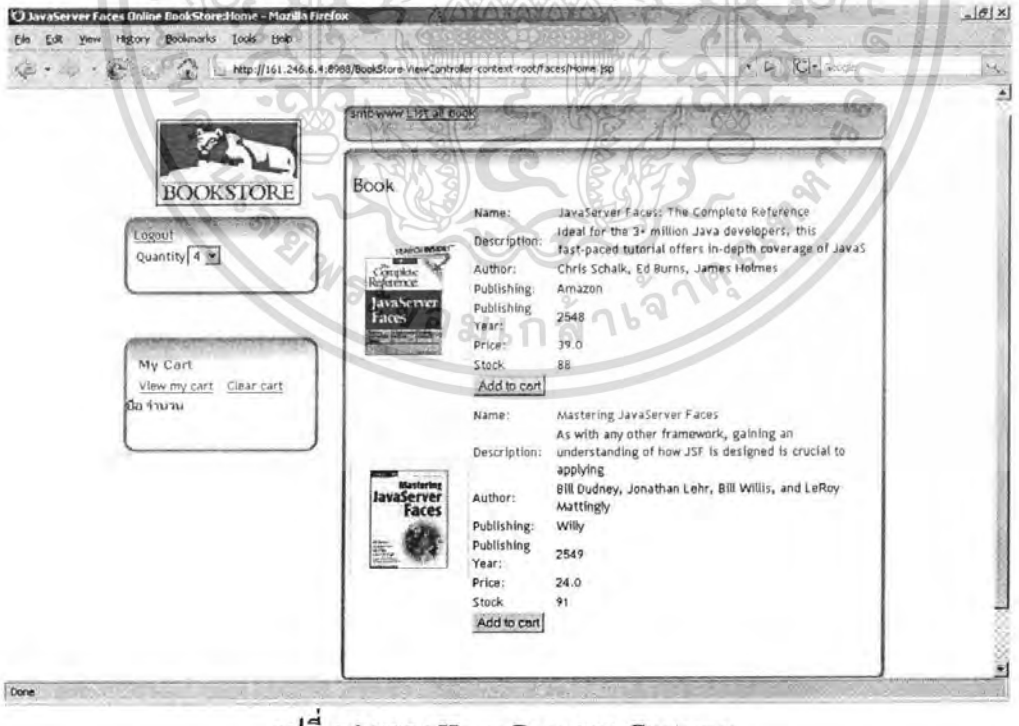


รูปที่ ก.1 แสดง Home Page ของระบบ BookStore Online

รูปที่ ก.2 แสดง page ของการกรอกข้อมูลเพื่อทำการ Register กับทางร้าน

Customer เข้ามาใช้งานระบบ

หลังจากที่ Customer login เข้ามาในระบบแล้วจะแสดง Home Page ดังรูปที่ ก.3

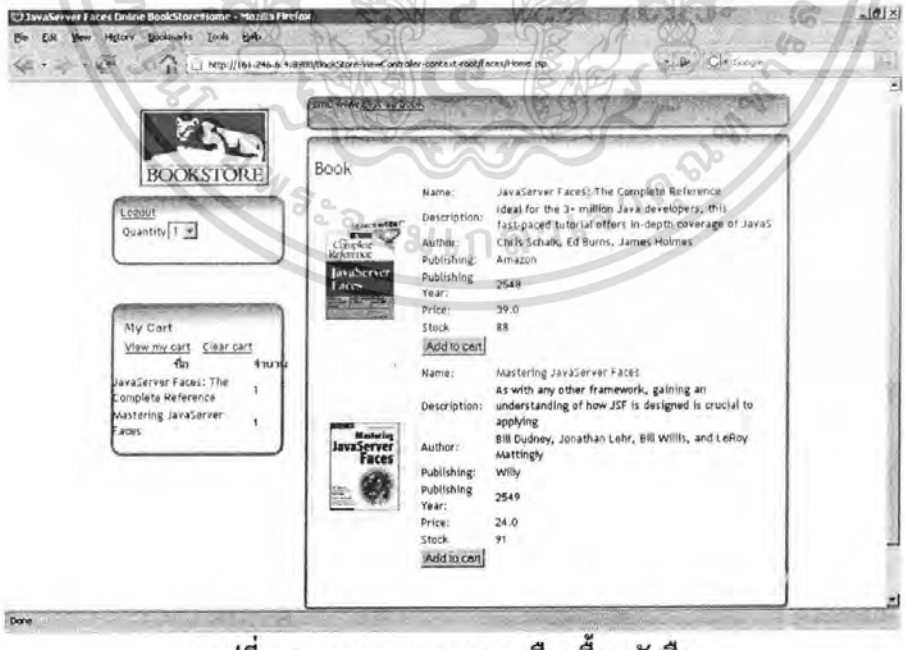


รูปที่ ก.3 แสดง Home Page ของ Customer

สามารถให้ระบบแสดงรายการหนังสือทั้งหมดได้เพื่อเลือกซื้อ โดย Click “List all book” ดังรูปที่ ก.4

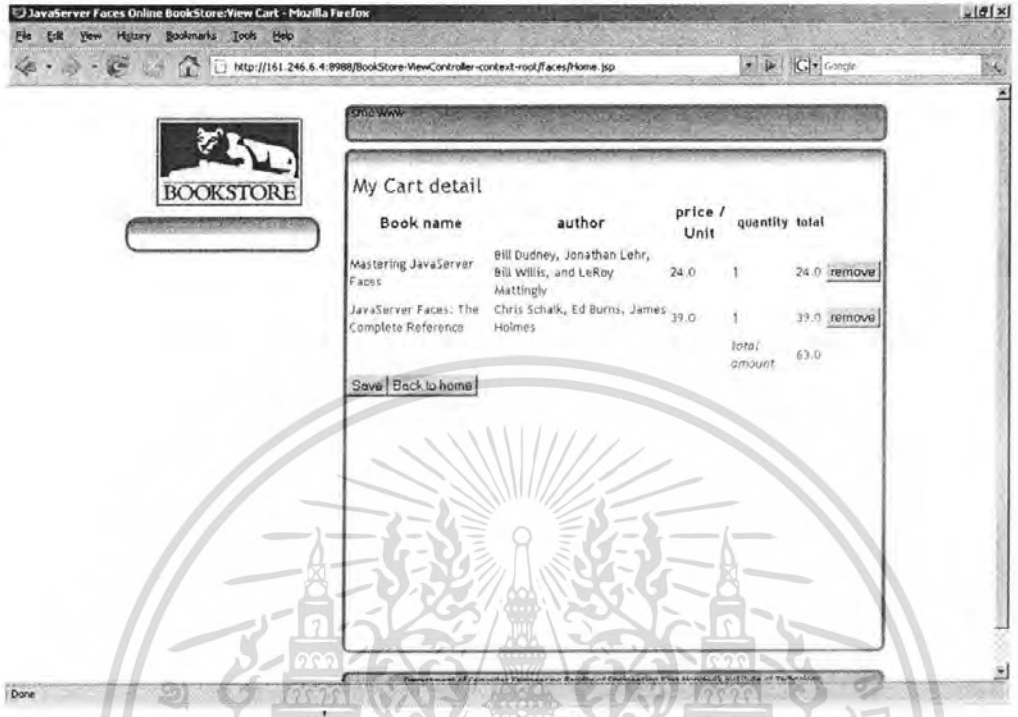
รูปที่ ก.4 แสดงรายการหนังสือทั้งหมดในร้าน

สามารถเลือกซื้อหนังสือได้ โดยจะเป็นการนำไปใส่ไว้ในตะกร้ารถเข็น Click “Add to cart” เมื่อ click แล้วจะแสดงชื่อหนังสือกับจำนวน ในกรอบ My Cart ทางซ้ายมือ ดังรูปที่ ก.5

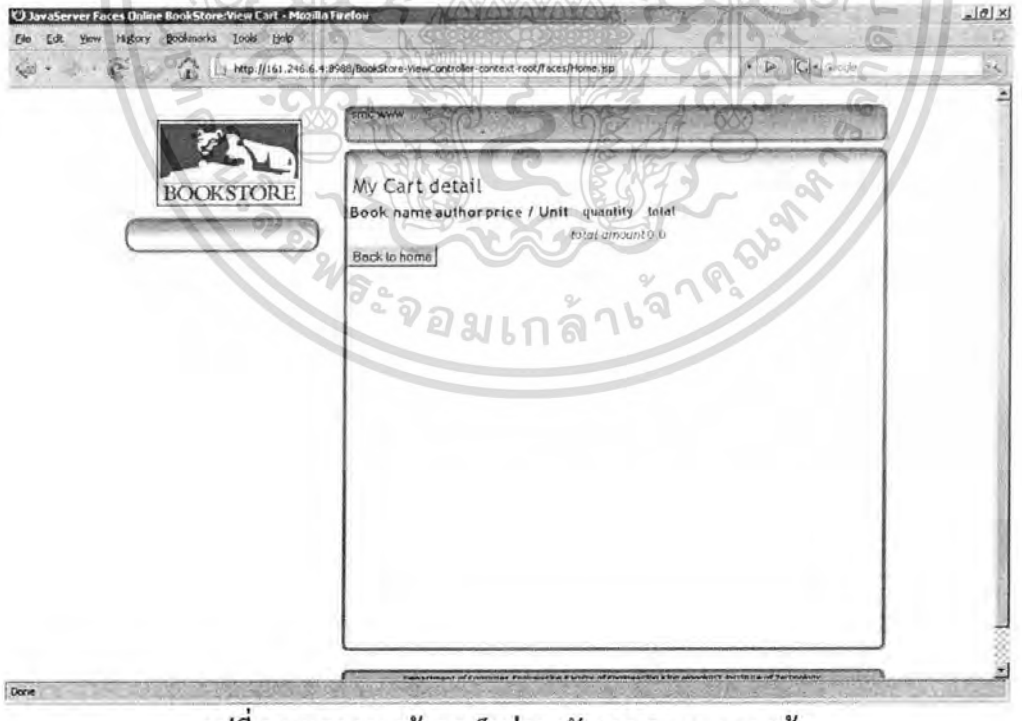


รูปที่ ก.5 แสดง page ของการเลือกซื้อหนังสือ

สามารถให้ระบบแสดงรายละเอียดในตะกร้ารถเข็นได้ โดย Click “View my cart” ที่ page นี้ยังยกเลิกหนังสือบางรายการได้ โดย Click “remove” และแสดงราคาหนังสือให้ด้วย ดังรูปที่ ก.6



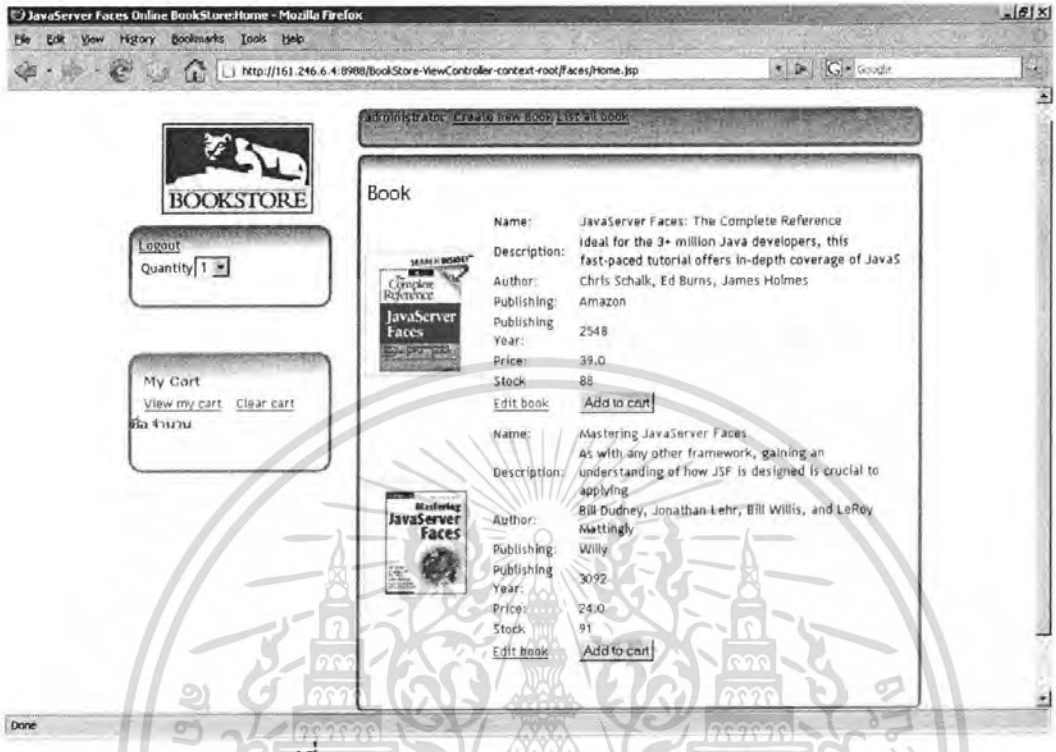
รูปที่ ก.6 แสดงรายละเอียดในตะกร้ารถเข็น



รูปที่ ก.7 แสดงตะกร้ารถเข็นว่างหลังจาก Clear cart แล้ว

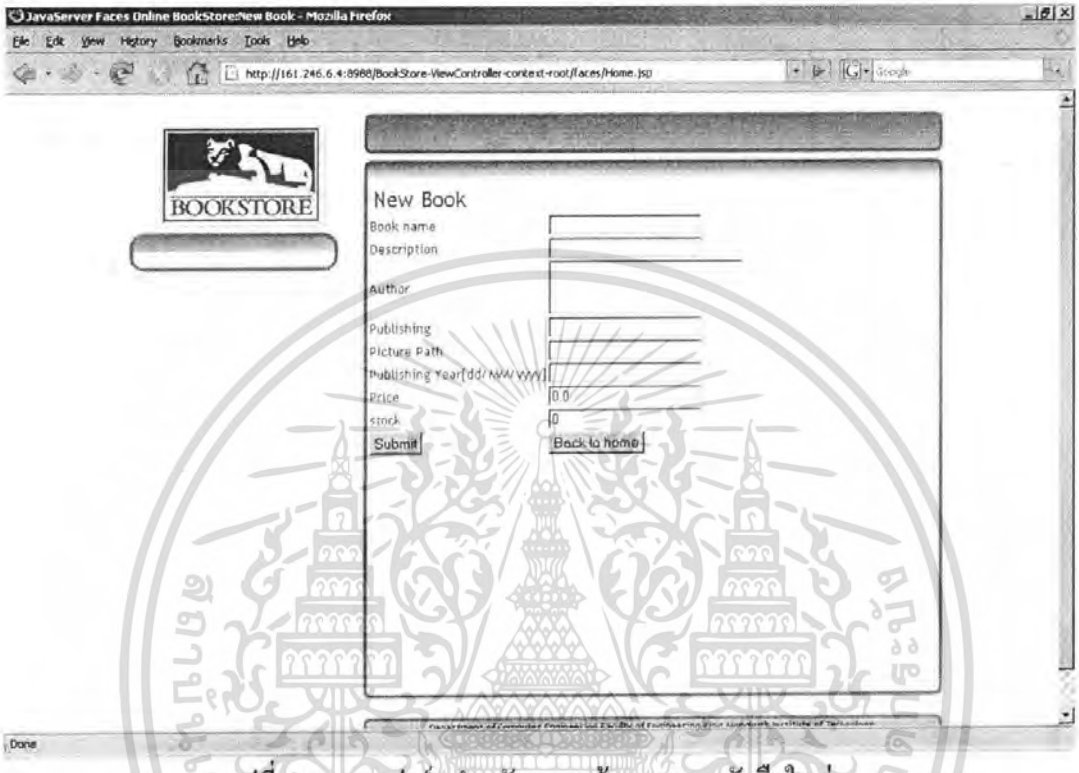
Administrator เข้ามาใช้งานระบบ

หลังจากที่ Administrator login เข้ามาใช้งานระบบจะแสดง Home Page ดังรูปที่ ก.8



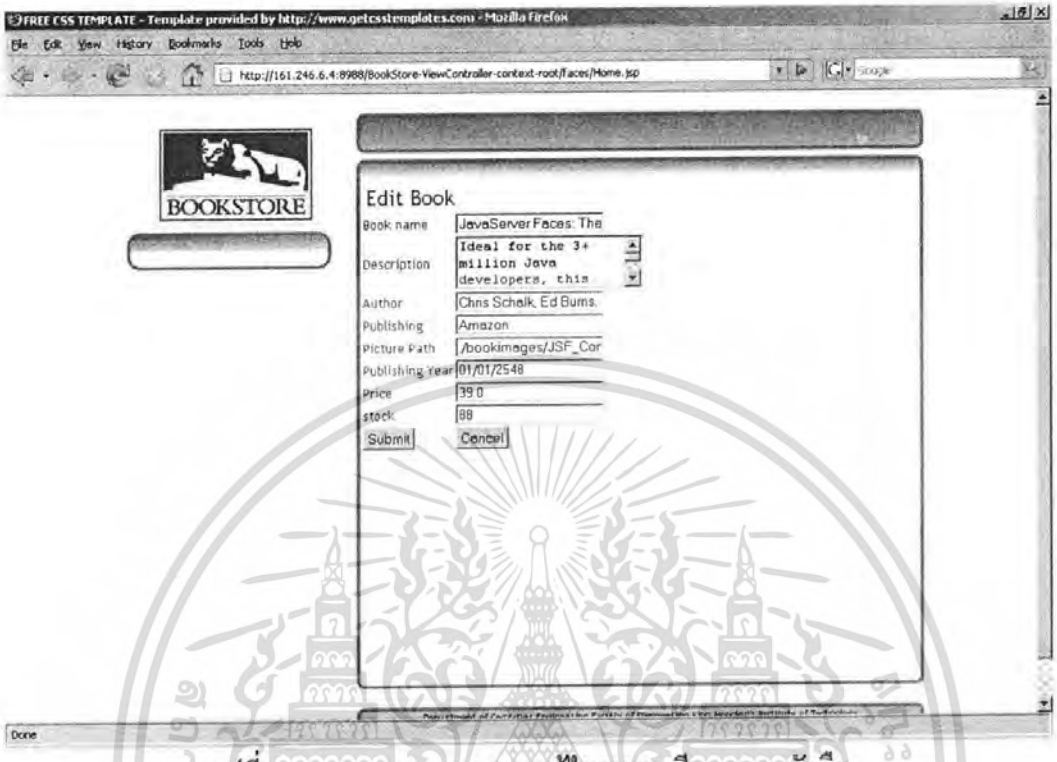
รูปที่ ก.8 แสดง Home Page ของ Administrator

สามารถเพิ่มรายการหนังสือใหม่เข้าไปใน list book ได้ โดย Click "Create new Book" Administrator จะต้องกรอกรายละเอียดของหนังสือ เช่น ชื่อหนังสือ, รายละเอียดย่อ, ผู้แต่ง, สำนักพิมพ์, รูปปกหนังสือ, ปีที่พิมพ์, ราคา และจำนวนหนังสือที่มีในร้าน หลังจาก Click "Submit" แล้วรายการหนังสือจะถูกเก็บไว้ในฐานข้อมูลหนังสือ ดังรูปที่ ก.9



รูปที่ ก.9 แสดงฟอร์มสำหรับกรอกข้อมูลของหนังสือใหม่

สามารถแก้ไขรายละเอียดของหนังสือได้ โดย Click "Edit book" สำหรับการแก้ไขนี้จะเป็นการไป update รายละเอียดของหนังสือในฐานข้อมูล ดังรูปที่ ก.10



รูปที่ ก.10 แสดง page ของการแก้ไขรายละเอียดของหนังสือ