J2EE & XML

จาวาระดับองค์กร

J2EE & XML จาวาระดับองค์กร
จำนวน **284** หน้า พร้อมซีดีรอมหนึ่งแผ่น
ราคาปก **275** บาท
ISBN : 974-915XX-X-X

ผู้เขียน
นรินทร์ โอฬารกิจอนันต์
ซัน จาวา 2 เซอร์ติฟายด์โปรแกรมเมอร์ (SJCP)
Email : narin@dekisugi.net

จัดทำโดย
สำนักพิมพ์ เดคิซูกิ ดอทเนต
URL : http://www.dekisugi.net/java
Email : webmaster@dekisugi.net

สงวนลิขสิทธิ์ ตามพระราชบัญญัติลิขสิทธิ์ พุทธศักราช 2521 โดย นาย นรินทร์ โอฬารกิจอนันต์ ห้ามมิให้นำส่วนหนึ่งส่วนใดหรือทั้งหมดของหนังสือไปใช้เพื่อประโยชน์เชิงพาณิชย์โดยไม่ได้รับอนุญาตเป็นลายลักษณ์อักษรจากเจ้าของลิขสิทธิ์

หนังสือเล่มนี้เป็นหนังสือเล่มที่สามในชุด หนังสือจาวา ต่อจาก
จาวา สำหรับผู้เริ่มต้น
เจเอสพี สำหรับเวบโปรแกรมเมอร์

โครงการหนังสือในอนาคต
J2ME จาวาบนมือถือ

เครื่องหมายการค้าทั้งหมดที่กล่าวถึงในหนังสือเล่มนี้เป็นขององค์กรหรือบริษัทที่กล่าวถึงทั้งหมด

สารบัญ

บทที่ 1 J2EE จาวาระดับองค์กร.....9

ที่มาของ J2EE 9
จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ 10
โครงสร้างของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ 11
คุณสมบัติเด่นของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ 13
J2EE vs .NET 16
J2EE กับประเทศไทย 16

EJB

บทที่ 2 ติดตั้ง J2EE server 19

เตรียมอุปกรณ์ 19
ดาวน์โหลด J2EE server 20
ติดตั้ง J2EE SDK 20
เรียกใช้งาน J2EE server 22
ทดสอบ J2EE server 22
ปลดโปรแกรม HelloWorld 26
จบการใช้งาน J2EE server 27

บทที่ 3 บีนบริการแบบไม่คงสถานะ 28

บีน 28
ประโยชน์ของการใช้บีน 29
แนวคิดเรื่อง Thin-client 29
ประเภทของบีน 30
บีนบริการ 30
ชนิดของบีนบริการ 31
บีน HelloWorld 32
คอมไพล์บีน 36
แพจเกจบีน 37
ไคลน์เอนท์ของ HelloWorld 42
การเรียกแมธธอสระยะไกล 48
โหลด HelloWorldApp 51
ทดสอบโปรแกรม HelloWorld 53
เวบไคลน์เอนท์สำหรับ HelloWorld 53

บทที่ 4 บีนบริการแบบไม่คงสถานะ (2).....

สถาปัตยกรรมการประมวลผลแบบกระจาย .....
บทบาทของโปรแกรมเมอร์ J2EE .....
วงจรชีวิตของบีนบริการแบบไม่คงสถานะ.....
ข้อกำหนดของการสร้างบีนบริการ .....
ธรรมเนียมนิยมในการตั้งชื่อคลาส .....

บทที่ 5 บีนบริการแบบคงสถานะ .....

บีนบริการแบบคงสถานะ .....
บีน Counter .....
ทดสอบ Counter .....

วงจรชีวิตของบีนบริการแบบคงสถานะ .....
บทที่ 6 บีนวัตถุแบบ BMP.....
บีนวัตถุ .....
ประเภทของบีนวัตถุ .....
บีน AccountBean.....
อินเตอร์เฟส Account .....
โฮมอินเตอร์เฟส AccountHome .....
โปรแกรม AccountClient .....
เตรียมฐานข้อมูล.....
โหลดโปรแกรม Account.....
วงจรชีวิตของบีนวัตถุ .....
สรุปข้อบังคับในการสร้าง BMP.....
บทที่ 7 บีนวัตถุแบบ CMP.....
CMP.....
บีน SavingsAccount.....
ทดสอบบีน SavingsAccount .....
แมธธอสค้นหาและแมธธอสเลือก .....
บทที่ 8 บีนที่สั่งด้วยแมสเสจ.....
แมจเสจ .....
บีนที่สั่งด้วยแมจเสจ .....
บีน Message .....
การกำหนดช่องส่งแมจเสจ.....
ทดสอบโปรแกรม.....
วงจรชีวิตของบีนแบบสั่งด้วยแมสเสจ .....
บทที่ 9 ระบบรักษาความปลอดภัย.....
ระบบรักษาความปลอดภัยบน J2EE.....
การตรวจสอบผู้ใช้ .....
การกำหนดสิทธิ์ในการรันแมธธอส .....
บทที่ 10 ติดต่อกับระบบอื่น.....
การติดต่อกับทรัพยากร.....
ติดต่อกับฐานข้อมูล .....
ติดต่อกับเมล์เซิร์ฟเวอร์ .....
บทที่ 11 การจัดการธุรกรรม.....
การจัดการธุรกรรม .....
การจัดการธุรกรรมในจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ .....
ประเภทของจัดการธุรกรรม.....
การกำหนดการจัดการธุรกรรม .....
บทที่ 12 บีนบริการเรียกบีนวัตถุ .....
ประโยชน์ .....
โปรแกรมตัวอย่าง.....
โหลดโปรแกรม TellerApp.....
บทที่ 13 คลัสเตอร์ .....
คลัสเตอร์ .....
คลัสเตอร์บีนบริการแบบไม่มีสถานะ.....
คลัสเตอร์บีนบริการแบบมีสถานะ .....
คลัสเตอร์ของบีนวัตถุ.....
คลัสเตอร์ของบีนที่สั่งด้วยแมสเสจ.....

XML

บทที่ 14 รู้จัก XML .....

XML ใช้ทำอะไร.....
รูปแบบของไฟล์ XML .....
XML กับ HTML .....
กฎอื่นๆ ของการเขียนไฟล์ XML .....
DTD.....
การใส่ XML ไว้ในไฟล์ HTML.....
XSL.....

บทที่ 15 จัดการกับเอกสาร XML ด้วย DOM.....

โหนด .....
DOM.....
ใช้ DOM สั่งโปรแกรม .....
ใช้ DOM สร้างเอกสาร XML .....

บทที่ 16 ส่ง SOAP ด้วย SAAJ .....

บริการผ่านเวบ .....
รูปแบบของข้อความ SOAP .....
ขั้นตอนของการใช้บริการผ่านเวบ.....
SAAJ.....
ส่งคำร้องขอการสืบค้น UDDI .....
JAXR.....

บทที่ 17 เรียกแมธธอสระยะไกล .....
ด้วย JAX-RPC.....

JAX-RPC.....
โครงสร้างของ JAX-RPC .....
บีนที่สนับสนุน JAX-RPC.....
ทดสอบโปรแกรม .....

J2EE APIs

บทที่ 18 ส่งแมธธอสระยะไกลด้วย .....
RMI-IIOP.....

รู้จัก RMI-IIOP.....
การติดต่อแบบไคลน์เอนท์-เซิร์ฟเวอร์.....
การเรียกแมธธอสระยะไกล .....
อินเตอร์เฟสสำหรับแมธธอสระยะไกล .....
เซิร์ฟเวอร์.....
RMI Registry .....
ไคลน์เอนท์.....
Stub และ Skeleton .....
ทดสอบโปรแกรม .....
การส่งผ่านตัวแปรและวัตถุของแมธธอสระยะไกล .....
การเรียกกลับ.....

บทที่19 ติดต่อกับสารบบด้วย JNDI .....

Naming Server .....
สารบบ.....
JNDI.....
การติดตั้ง SPI .....
การสืบค้นวัตถุ .....
ติดต่อกับ Directory Server .....

บทที่ 20 สร้าง Web Application ด้วย ..... จาวาเซิร์ฟเลต.....

Web Application .....
จาวาเซิร์ฟเลต .....
เซิร์ฟเลตคอนเทนเนอร์ .....
HelloWorldServlet.....
หลักการทำงานของเซิร์ฟเลต .....
HttpServlet.....
การใช้เซิร์ฟเลตจัดการกับฟอร์ม HTML.....
ใช้เซิร์ฟเลตสร้างฟอร์มแบบหลายหน้า.....
จาวาเซิร์ฟเวอร์ เพจ.....

บทที่ 21 ติดต่อกับฐานข้อมูลด้วย JDBC.....

ไดร์เวอร์ JDBC.....
ติดตั้งฐานข้อมูล mySQL .....
คำสั่งโหลดไดร์เวอร์ .....
ต่อไปยังฐานข้อมูล.....
สร้างตารางใหม่ .....
แทรกแถวข้อมูล.....
การ SELECT ข้อมูล.....
ใช้ ResultSet ในการอัพเดทข้อมูล .....
PrepareStatement.....
การจัดการ Transaction.....
ปิดการติดต่ออย่างมั่นใจ .....

บทที่ 22 ส่งอีเมล์ด้วย JavaMail .....

SMTP.....
POP.....
IMAP .....
ชุดคำสั่ง JavaMail .....
ติดตั้งเมล์เซิร์ฟเวอร์.....
ส่งอีเมล์ .....
รับอีเมล์ .....
ลบอีเมล์.....
การแนบเอกสาร.....

บทที่ 23 สื่อสารผ่านระบบแมสเสจจิงด้วย JMS.....

แมจเสจจิง.....
JMS .....
โดเมน .....
ติดตั้ง OpenJMS.....
กระจายข่าว .....
การรับข้อความ .....
การส่งข้อความแบบจุดต่อจุด .....
ชนิดของข้อความ.....

คำแนะนำในการอ่าน

เนื้อหาของหนังสือเล่มนี้
หนังสือเล่มนี้เป็นหนังสือสอนเกี่ยวกับการพัฒนาโปรแกรมประยุกต์ระดับองค์กรโดยใช้ภาษาจาวา ซึ่งเป็นหนังสือเล่มที่สามในชุดของหนังสือจาวา ต่อจากสองเล่มแรก ได้แก่ *"จาวา สำหรับผู้เริ่มต้น"* และ *"เจเอสพี สำหรับเวบโปรแกรมเมอร์"* ซึ่งเป็นการแนะนำภาษาจาวาเบื้องต้น และสอนการใช้ภาษาจาวาเพื่อสร้างเวบเพจ ตามลำดับ

เนื้อหาของหนังสือเล่มนี้แบ่งเป็นสามส่วน ส่วนแรกเป็นเรื่องเกี่ยวกับการสร้างโปรแกรมประยุกต์เพื่อใช้ทำงานบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์โดยเฉพาะที่เรียกว่า EJB ซึ่งจัดว่าเป็นหัวใจของการสร้างโปรแกรมประยุกต์ระดับองค์กรด้วยภาษาจาวา ส่วนที่สองเป็นเรื่องเกี่ยวกับ XML และชุดคำสั่งภาษาจาวาที่เกี่ยวข้องกับการจัดการ XML และส่วนที่สามเป็นภาคผนวกซึ่งแนะนำเกี่ยวกับการใช้ชุดคำสั่งต่างๆ ของ J2EE ในระดับพื้นฐาน เผื่อไว้สำหรับคนที่สนใจวิธีการสร้างโปรแกรมประยุกต์ระดับองค์กรโดยไม่พึ่งแอพพลิเคชั่นเซิร์ฟเวอร์

ความรู้พื้นฐานที่จำเป็นสำหรับการอ่าน
แม้ว่าเนื้อหาของหนังสือเล่มนี้จะไม่ใช่ตอนต่อจากหนังสือสองเล่มแรกโดยตรง แต่ผู้อ่านจำเป็นต้องมีความคุ้นเคยกับการใช้คำสั่งในภาษาจาวามาแล้วระดับหนึ่ง อย่างน้อยที่สุดควรมีความคุ้นเคยกับคำสั่งในการสร้างและสืบทอดคลาสของภาษาจาวา และจะยิ่งดีหากเคยสัมผัสการสร้างเวบเพจด้วยเจเอสพีมาบ้างแล้ว มิฉะนั้นผู้อ่านจะทำความเข้าใจโปรแกรมตัวอย่างในหนังสือเล่มนี้ได้ยาก เพราะการอธิบายโปรแกรมตัวอย่างในหนังสือเล่มนี้จะไม่มีการอธิบายวิธีการใช้คำสั่งพื้นฐานเหล่านี้ซ้ำอีก

แหล่งหาความรู้พื้นฐาน
ถ้าหากคุณไม่มีพื้นฐานภาษาจาวามาก่อน ขอแนะนำให้ดาวน์โหลดตัวอย่างของหนังสือสองเล่มข้างต้นมาอ่านก่อนได้ฟรีจากเวบไซต์ของหนังสือจาวาที่ http://www.dekisugi.net/java ตัวอย่างหนังสือที่ให้ดาวน์โหลดฟรีมีเนื้อหาไม่เท่ากับฉบับสมบูรณ์แต่เพียงพอสำหรับการสร้างพื้นฐานความรู้ก่อนที่จะอ่านหนังสือเล่มนี้

สำหรับผู้ที่สนใจจะได้หนังสือฉบับสมบูรณ์ของหนังสือตัวอย่างทั้งสองเล่ม สามารถอ่านรายละเอียดวิธีการสั่งซื้อได้จากตอนท้ายสุดของหนังสือเล่มนี้

การทดสอบโปรแกรมตัวอย่าง
ควรทดลองรันโปรแกรมตัวอย่างไปด้วยทุกโปรแกรมในขณะที่อ่านเพื่อให้เกิดความเข้าใจอย่างแท้จริง และควรปฏิบัติตามขั้นตอนการทดสอบโปรแกรมในหนังสืออย่างเคร่งครัดเพื่อให้การทดลองรันโปรแกรมทำได้อย่างราบรื่น ไม่ควรข้ามขั้นตอน เพราะหากเกิดข้อผิดพลาดแล้วจะหาสาเหตุได้ยาก โปรแกรมตัวอย่างทุกโปรแกรมจะอยู่ใต้โฟลเดอร์ชื่อ examples ในแผ่นซีดีรอมที่แถมมากับหนังสือ

นอกจากนี้ขอแนะนำให้ติดตั้งโปรแกรมต่างๆ จากแผ่นซีดีรอมที่แถมมาด้วย การเปลี่ยนไปใช้เวอร์ชั่นที่ใหม่กว่าของโปรแกรมเหล่านี้โดยการดาวน์โหลดมาจากเวบไซต์ของจาวาโดยตรงอาจได้โปรแกรมที่ทันสมัยกว่าแต่จะทำให้เกิดความสับสนเวลาทำตามวิธีในหนังสือได้ เพราะเมนูต่างๆ ของโปรแกรมแต่ละเวอร์ชั่นไม่เหมือนกัน พยายามศึกษาจากหนังสือจนเข้าใจโดยใช้โปรแกรมที่อยู่ในซีดีรอมให้ถ่องแท้ก่อนแล้ว จึงค่อยหันไปลองใช้เวอร์ชั่นใหม่จะเป็นการดีกว่า

เป้าหมายของหนังสือเล่มนี้
จาวาระดับองค์กรเป็นเรื่องใหญ่มาก ไม่มีทางที่คนคนเดียวจะเรียนรู้ทุกอย่างได้หมด การพัฒนาโปรแกรมประยุกต์ในระดับองค์กรส่วนมากอาศัยการทำงานเป็นทีม เพื่อให้โปรแกรมเมอร์แต่ละคนมี

เวลามากพอที่จะเรียนรู้เรื่องใดเรื่องหนึ่งเพื่อให้เกิดความชำนาญเฉพาะเรื่องได้ เป้าหมายของหนังสือเล่มนี้ คือ การเปิดโลกทัศน์เบื้องต้นของผู้อ่านไปสู่โลกของการเขียนโปรแกรมประยุกต์ระดับองค์กรด้วยภาษาจาวา เมื่ออ่านจบแล้วผู้อ่านจะเห็นภาพของกระบวนการพัฒนาทั้งระบบและสามารถเขียนโปรแกรมประยุกต์ระดับองค์กรอย่างง่ายๆ ได้ แต่หากผู้อ่านต้องการนำไปใช้งานจริงในอนาคต จำเป็นที่ผู้อ่านจะต้องมีการศึกษาหาความรู้เพิ่มเติมอีกในระดับที่ยากยิ่งๆ ขึ้นไป

ตรวจสอบความรู้ใหม่ๆ

จาวาระดับองค์กรยังอยู่ในขั้นของการพัฒนาและยังคงมีความเปลี่ยนแปลงอยู่ตลอดเวลา เพื่อให้ผู้อ่านทันความเปลี่ยนแปลงอยู่เสมอท่าน กรุณาตรวจสอบความเปลี่ยนแปลงได้จากเวบไซต์ http://www.dekisugi.net/java/support

1

J2EE จาวาระดับองค์กร

J2EE (อ่านว่า เจ-ทู-ดับ-เบิล-อี) ย่อมาจากคำว่า **Java 2 Enterprise Edition** คือ กลุ่มของเทคโนโลยีภาษาจาวาที่จำเป็นสำหรับการพัฒนาโปรแกรมประยุกต์ด้วยภาษจาวาให้ใช้งานได้ในระดับองค์กร

ที่มาของ J2EE

แต่เดิมนั้นภาษาจาวาถูกใช้ในการสร้างโปรแกรมประยุกต์ส่วนบุคคลเป็นหลัก ต่อมาเมื่อจาวาถูกนำไปใช้งานภายในองค์กรมากขึ้น ผู้พัฒนาจาวาจำเป็นต้องเพิ่มชุดคำสั่งใหม่ๆ เข้าไปเพื่อตอบสนองความต้องการใช้งานระดับองค์กร เมื่อชุดคำสั่งที่เพิ่มเติมเข้าไปมีจำนวนมากขึ้นเรื่อยๆ ผู้พัฒนาจาวาจึงได้ทำการสังคายนาจัดระเบียบชุดคำสั่งในภาษาจาวาเสียใหม่ โดยแบ่งเทคโนโลยีจาวาออกเป็นสามระดับ ได้แก่

1. **J2SE** (Java 2 Standard Edition) หรือ จาวาระดับมาตรฐาน คือ จาวาดั้งเดิมที่มีแต่ชุดคำสั่งปกติสำหรับการสร้างโปรแกรมประยุกต์ส่วนบุคคล

2. **J2EE** (Java 2 Enterprise Edition) หรือ จาวาระดับองค์กร คือ จาวาที่มีชุดคำสั่งเพื่อการสร้างโปรแกรมประยุกต์ที่ใช้งานระดับองค์กร
3. **J2ME** (Java 2 Micro Edition) หรือ จาวาระดับจิ๋ว คือ จาวาที่มีชุดคำสั่งเพื่อการสร้างโปรแกรมประยุกต์สำหรับอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก

J2EE ถือได้ว่าเป็นส่วนขยายของ J2SE ด้วย กล่าวคือ J2EE รวมเอาทั้งหมดของ J2SE เข้าไปอยู่ในตัวมันแล้วเพิ่มชุดคำสั่งที่จำเป็นสำหรับการพัฒนาโปรแกรมระดับองค์กรเข้าไป ความรู้เกี่ยวกับ J2SE จึงเป็นพื้นฐานที่จำเป็นสำหรับการศึกษา J2EE

ตารางที่ 1-1 ชุดคำสั่งในชุดของ J2EE

ชื่อชุดคำสั่ง	หน้าที่และประโยชน์ใช้สอย
RMI-IIOP	ใช้สำหรับการสื่อสารหรือเรียกใช้แมธธอสระยะไกลผ่านระบบเครือข่าย
Java IDL	ภาษากลางสำหรับสื่อสารกับโปรแกรมประยุกต์ระดับองค์กรที่พัฒนาด้วยเทคโนโลยีอื่น
JNDI	ระบบการเรียกชื่อของบริการหรือวัตถุที่อยู่บนระบบเครือข่าย
JDBC	ใช้เชื่อมติดต่อกับฐานข้อมูล
JMS	ใช้ประสานงานหรือสื่อสารระหว่างโปรแกรมประยุกต์ด้วยกัน
JavaMail	ใช้ติดต่อกับเมล์เซิร์ฟเวอร์ในองค์กร
Java Servlets	บริการเวบบนฝั่งเซิร์ฟเวอร์
JSP	บริการเวบบนฝั่งเซิร์ฟเวอร์ที่ดัดแปลงมาจาก Java Servlets อีกทีหนึ่ง โดยมีความง่ายในการพัฒนามากกว่า
JTA	ชุดคำสั่งสำหรับการบริหารจัดการธุรกรรม
JTS	บริการธุรกรรม
EJB	โครงสร้างโปรแกรมประยุกต์แบบคอมโพเนนท์
JAXP	ชุดคำสั่งสำหรับการจัดการไฟล์ XML
JAAS	ชุดคำสั่งสำหรับระบบรักษาความปลอดภัย
JCA	ใช้ติดต่อกับระบบอื่นๆ ที่มีอยู่เดิมในเครือข่ายขององค์กร

จะเห็นได้ว่า J2EE ประกอบด้วยชุดคำสั่งจำนวนมาก ตอนนี้อย่าเพิ่งสนใจว่าแต่ละตัวใช้ทำอะไร เราจะค่อยๆ รู้จักมันไปเรื่อยๆ ทีละตัว

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์

อย่างไรก็ตาม การนิยามคำว่า J2EE ขึ้นมาไม่ได้เป็นแค่การจัดกลุ่มชุดคำสั่งให้เป็นระบบเท่านั้น แต่ได้มีการสร้างนวัตกรรมของการพัฒนาโปรแกรมระดับองค์กรด้วย กล่าวคือมีการสร้างแนวคิดเรื่องแอพพลิเคชั่นเซิร์ฟเวอร์ขึ้นมา **แอพพลิเคชั่นเซิร์ฟเวอร์** เป็นการรวมเทคโนโลยีในชุดของ J2EE ไว้ในที่เดียวกัน เพื่อให้เกิดความง่ายในการพัฒนาโปรแกรม

แต่เดิมการเขียนโปรแกรมประยุกต์ให้ใช้งานได้ในระดับองค์กรเป็นเรื่องยากมาก เพราะนอกจากจะต้องเขียนคำสั่งให้โปรแกรมทำงานอย่างที่เราต้องการได้แล้ว โปรแกรมเมอร์ระดับองค์กรยังต้องระดมสรรพกำลังเพื่อเขียนคำสั่งที่ทำให้โปรแกรมนั้นทำงานบนเครือข่ายได้อย่างมีประสิทธิภาพด้วย ซึ่งไม่ใช่เรื่องง่ายเพราะในองค์กรประกอบด้วยเครื่องคอมพิวเตอร์จำนวนมากทำงานร่วมกันผ่านระบบเครือข่ายที่มีผู้ใช้จำนวนมากทำงานอยู่

NOTE

ถ้าอยากพอเข้าใจว่าการจะเขียนโปรแกรมเล็กๆ สักโปรแกรมหนึ่งให้ทำงานได้บนเครือข่ายนั้นยุ่งยากขนาดไหน ลองอ่าน บทที่ 19 การเรียกแมธธอสระยะไกลด้วย RMI-IIOP

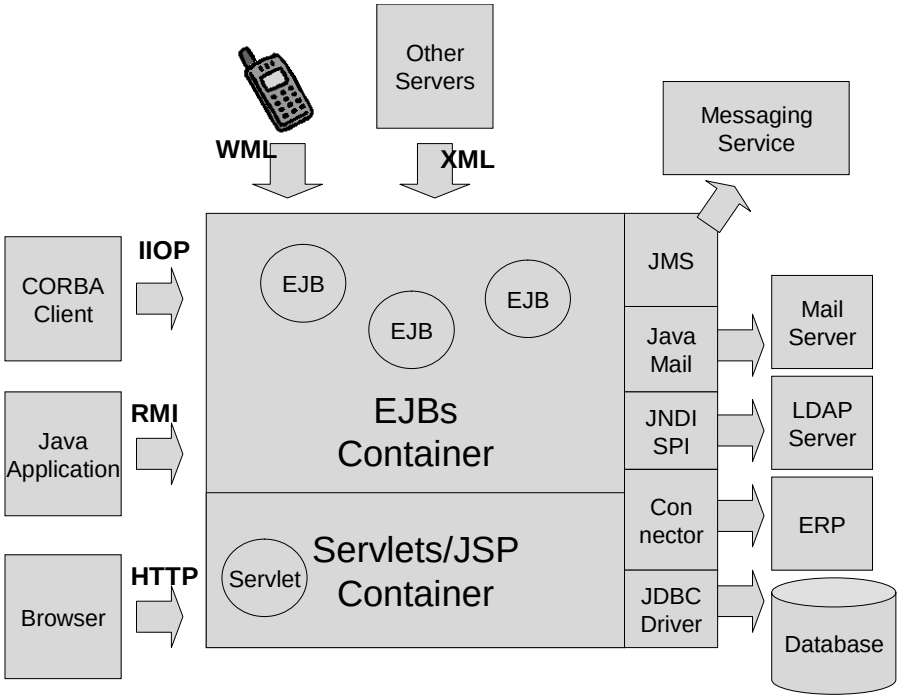
แนวคิดเรื่องแอพพลิเคชั่นเซิร์ฟเวอร์ทำให้การพัฒนาโปรแกรมประยุกต์สำหรับองค์กรทำได้ง่ายขึ้น เพราะแอพพลิเคชั่นเซิร์ฟเวอร์จะรับผิดชอบเกี่ยวกับการจัดการเครือข่ายทั้งหมดแทนเรา โปรแกรมเมอร์เพียงแต่เขียนโปรแกรมให้ทำงานอย่างที่ใจต้องการแล้วนำโปรแกรมที่ได้ไปรันบนแอพพลิเคชั่นเซิร์ฟเวอร์ แอพพลิเคชั่นเซิร์ฟเวอร์จะทำหน้าที่เนรมิตโปรแกรมอันนั้นให้ทำงานบนเครือข่ายคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ อาจกล่าวได้ว่าจาวาแอพพลิเคชั่นเซิร์ฟเวอร์เป็นหัวใจของ J2EE

โครงสร้างของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ มีลักษณะคล้ายกับจาวาเวอร์ชัวน์แมทชีนในจาวาระดับมาตรฐาน กล่าวคือ เป็นสิ่งที่โปรแกรมภาษาจาวาที่เราเขียนขึ้นทำงานบนตัวมันอีกที อาจกล่าวว่าจาวาแอพพลิเคชั่นเซิร์ฟเวอร์เป็นตัวที่ห่อหุ้มโปรแกรมภาษาจาวาระดับองค์กรที่เราเขียนเอาไว้จากสภาวะแวดล้อมภายนอก มันทำหน้าที่เป็นตัวกลางในการติดต่อสื่อสารกับภายนอกแทนโปรแกรมของเราแบบเดียวกับจาวาเวอร์ชัวน์แมทชีนที่คั่นกลางระหว่างโปรแกรมภาษาจาวาของเรากับระบบปฏิบัติการของเครื่อง

แต่จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ มีอะไรที่พิเศษกว่าจาวาเวอร์ชัวน์แมทชีนที่เราคุ้นเคย เพราะในระดับองค์กร นอกจากโปรแกรมประยุกต์จะต้องติดต่อกับระบบปฏิบัติการแล้ว มันยังต้องติดต่อกับกับสิ่งอื่นอีกมากมาย ซึ่งส่วนมากเป็นการติดต่อผ่านทางระบบเครือข่ายขององค์กร เช่น บางครั้งโปรแกรมต้องดึงข้อมูลจากฐานข้อมูลที่อยู่บนเครื่องคอมพิวเตอร์อีก

เครื่องหนึ่งที่ห่างออกไปบนเครือข่ายองค์กร เป็นต้น จาวาแอพพลิเคชั่นเซิร์ฟเวอร์จะทำหน้าที่เป็นธุระจัดการการติดต่อสื่อสารเหล่านี้ให้กับโปรแกรมของเรา



รูปภาพ 1-1 โครงสร้างของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ จะมาในรูปของผลิตภัณฑ์สำเร็จรูปมาตรฐานซึ่งผลิตโดยบริษัทผู้ผลิตซอฟท์แวร์ชั้นนำ โปรแกรมเมอร์ระดับองค์กรสามารถเลือกซื้อหาจาวาแอพพลิเคชั่นเซิร์ฟเวอร์เหล่านี้มาใช้งานในองค์กรได้ทันที ตัวอย่างของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ที่แพร่หลายอยู่ในท้องตลาดมีดังนี้

ตารางที่ 1-2 จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ยอดนิยม

ชื่อ	ผู้ผลิต
WebLogic	BEA Systems
WebSphere	IBM
Oracle 9i Application Server	Oracle
Sun ONE	Sun Microsystems
Jrun	Macromedia

JEUS	Tmax Soft
Jboss Application Server	JBoss
Enterprise Server, AppServer Edition	Borland

จาวาระดับองค์กรยังคงแนวคิดเรื่องความเป็นระบบเปิดเอาไว้กล่าวคือ โปรแกรมประยุกต์ภาษาจาวาระดับองค์กรที่คุณพัฒนาขึ้นมาโปรแกรมหนึ่ง สามารถนำมารันบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์เหล่านี้ตัวใดก็ได้ เพราะพวกมันเข้าใจชุดคำสั่งทั้งหมดในชุดของ J2EE เหมือนกัน โปรแกรมภาษาจาวาของคุณจึงไม่ยึดติดอยู่กับระบบใดระบบหนึ่ง สิ่งที่แตกต่างกันระหว่างจาวาแอพพลิเคชั่นเซิร์ฟเวอร์แต่ละตัวเป็นเพียงแค่โปรแกรมสนับสนุนต่างๆ ที่ผู้ผลิตเหล่านี้นำมาใช้เป็นจุดขายเพื่อทำให้การเขียนโปรแกรมทำได้ง่ายขึ้น ในแง่ของต้องจาวาแอพพลิเคชั่นเซิร์ฟเวอร์เองแล้ว ไม่มีความแตกต่าง

คุณสมบัติเด่นของจาวาแอพพลิเคชั่นเซิร์ฟเวอร์

คุณสมบัติเด่นของการใช้จาวาแอพพลิเคชั่นเซิร์ฟเวอร์นอกจากเรื่องของความง่ายในการพัฒนาและความเป็นระบบเปิดแล้ว ยังมีคุณสมบัติเด่นอื่นๆ อีกดังนี้

สนับสนุนมาตรฐาน COBRA

บนเครือข่ายองค์กรมีวิธีการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์หลายแบบ ทำให้ที่ผ่านมาการเชื่อมโยงข้อมูลระหว่างคอมพิวเตอร์ที่ใช้เทคโนโลยีต่างค่ายกันทำได้ยาก ดังนั้นผู้ผลิตจึงรวมตัวกันเพื่อกำหนดมาตรฐานสำหรับการสื่อสารระหว่างเครื่องคอมพิวเตอร์ใหม่ให้เป็นอย่างเดียวกัน มาตรฐานใหม่นี้มีชื่อว่า **CORBA** (Common Object Request Broker Architecture) ซึ่งเป็นการคุยกันผ่านโปรโตคอลมาตรฐานที่เรียกว่า **IIOP** (อ่านว่า ไอ-ออบ) และใช้ภาษากลางที่เป็นมาตรฐานชื่อว่า **IDL** (Interface Definition Language)

J2EE แต่เดิมมีวิธีการติดต่อสื่อสารเป็นแบบของตนเองเรียกว่า **RMI** (Remote Method Interface) ต่อมาได้ปรับปรุงใหม่ให้เข้ากันได้กับมาตรฐาน CORBA และเรียกชื่อใหม่ว่า **RMI-IIOP** พร้อมกับสร้างภาษา **Java IDL** ขึ้นเพื่อสนับสนุนภาษา IDL ทำให้โปรแกรมประยุกต์ที่เขียนขึ้นเพื่อรันบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์สามารถติดต่อกับระบบอื่นบนเครือข่ายองค์กรที่ไม่ได้พัฒนาด้วยจาวาได้ด้วย ถ้าระบบเหล่านั้นสนับสนุนมาตรฐาน CORBA เหมือนกัน

สนับสนุนแนวคิดของเวบแอพพลิเคชั่น

ปัจจุบันอินเตอร์เนตเข้ามามีบทบาทอย่างมากในองค์กร การที่เบราเซอร์เป็นโปรแกรมที่มีอยู่แล้วบนเครื่องพีซีทุกเครื่อง ทำให้เกิดแนวคิดที่ว่า โปรแกรมในระดับองค์กรควรใช้เบราเซอร์ทำหน้าที่เป็นส่วนที่ผู้ใช้ใช้ในการติดต่อกับเครื่องแม่ข่าย แทนที่จะต้องสร้างโปรแกรมสำหรับติดต่อกับเครื่องแม่ข่ายโดยเฉพาะขึ้นมา จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ก็สนับสนุนแนวคิดนี้ด้วย เพราะในจาวาแอพพลิเคชั่นเซิร์ฟเวอร์จะมีคอนเทนเนอร์สำหรับ Java Servlets และ Java Server Pages (JSP) พ่วงมาด้วย เราจึงสามารถใช้เบราเซอร์เป็นตัวติดต่อกับโปรแกรมที่พัฒนาบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์โดยใช้คอนเทนเนอร์เหล่านี้เป็นตัวกลางได้

ใช้หลักการของคอมโพเนนท์

โปรแกรมประยุกต์ระดับองค์กรที่ดีต้องมีสถาปัตยกรรมที่ยึดหลักของ **คอมโพเนนท์** คอมโพเนนท์ คือ หน่วยของโปรแกรมที่ทำงานเฉพาะอย่างหนึ่ง มีอิสระในตัวเอง สามารถทำงานร่วมกับคอมโพเนนท์อื่นเพื่อทำงานที่มีความซับซ้อนมากขึ้นได้ การใช้สถาปัตยกรรมแบบคอมโพเนนท์แทนการเขียนโปรแกรมขนาดใหญ่โปรแกรมเดียวเพื่อรองรับงานทั้งหมด ทำให้โปรแกรมที่พัฒนาขึ้นมีความยืดหยุ่นสูง สามารถขยายตัวเองเพื่อรองรับงานจากผู้ใช้จำนวนมากขึ้นได้ง่าย และสามารถใช้คอมพิวเตอร์หลายเครื่องช่วยกันทำงานอย่างเดียวกันให้เสร็จเร็วขึ้นด้วยการแบ่งคอมโพเนนท์ออกไปรันบนเครื่องคอมพิวเตอร์หลายเครื่องที่เชื่อมโยงกันด้วยระบบเครือข่ายได้ ทั้งหมดนี้เป็นไปได้โดยไม่ต้องมีการแก้ไขคำสั่งในโปรแกรมใหม่

คอมโพเนนท์บนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ มีชื่อเรียกว่า **EJB** (Enterprise Java Bean) หรือบางครั้งเรียกสั้นๆ ว่า **บีน**

NOTE

คำว่า Enterprise JavaBeans (EJB) ไม่เหมือนกับคำว่า JavaBeans คำว่า JavaBeans เป็นชื่อเรียกวิธีการเขียนโปรแกรมแบบหนึ่งซึ่งแยกส่วนของการเรียกใช้งานโปรแกรมกับการเขียนคำสั่งออกจากกัน ส่วน Enterprise JavaBeans เป็นชื่อเรียกเฉพาะของคอมโพเนนท์ที่เขียนด้วยภาษาจาวา แม้ว่าทั้งสองคำจะมีความเกี่ยวเนื่องกันอยู่ แต่ทั้งสองคำมีความหมายแตกต่างกัน

สนับสนุนแนวคิดของบริการผ่านเวบ

บริการผ่านเวบ (Web Services) เป็นแนวคิดใหม่ที่กำลังมาแรงในแวดวงคอมพิวเตอร์ระดับองค์กร เป็นวิธีการติดต่อสื่อสารข้ามองค์กรผ่านเครือข่ายอินเตอร์เนตโดยอาศัยมาตรฐาน

เดียวกันได้แก่ XML (eXtensible Markup Language) โดยชุดคำสั่งภาษาจาวาที่เกี่ยวกับการจัดการ XML เช่น **JAXP, JAXM, JAX-RPC** ได้ถูกนำมารวมไว้ในจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ทำให้จาวาแอพพลิเคชั่นเซิร์ฟเวอร์เป็นแพลตฟอร์มสำหรับการพัฒนาบริการผ่านเวบในองค์กรที่สะดวกที่สุด

มีการบริหารหน่วยความจำที่มีประสิทธิภาพ

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ช่วยเหลือโปรแกรมของคุณในการบริหารหน่วยความจำของเครื่องให้สามารถรองรับผู้ใช้จำนวนมากในเวลาเดียวกันได้มากกว่าปกติ เพราะสามารถเพิ่มลดตัวโปรแกรมที่อยู่ในหน่วยความจำได้ตามจำนวนผู้ใช้ที่เข้ามา โดยสลับเอาโปรแกรมตัวที่ถูกใช้งานไม่บ่อยออกไปพักไว้ในหน่วยความจำสำรองเพื่อให้เกิดพื้นที่ว่างสำหรับโปรแกรมที่ทำงานหนัก โดยกลไกเหล่านี้เป็นกลไกที่เกิดขึ้นโดยอัตโนมัติโดยการจัดการของเซิร์ฟเวอร์

มีการจัดการด้านธุรกรรมข้อมูล

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์มีระบบจัดการด้านธุรกรรมข้อมูลมาด้วยในตัว ทำให้การใช้งานข้อมูลสำคัญที่ต้องใช้ฐานข้อมูลเป็นไปอย่างมีเสถียรภาพ จาวาแอพพลิเคชั่นเซิร์ฟเวอร์สนับสนุนชุดคำสั่ง **JTA** (Java Transaction API) และ **JTS** (Java Transaction Service) ทำให้โปรแกรมเมอร์สามารถใช้ประโยชน์จากชุดคำสั่งเหล่านี้ได้โดยไม่ต้องเขียนคำสั่งจัดการธุรกรรมด้วยตนเอง

มีระบบรักษาความปลอดภัยและระบบจัดการผู้ใช้

จาวาแอพพลิเคชั่นเซิร์ฟเวอร์มีระบบรักษาความปลอดภัยและระบบจัดการผู้ใช้ที่สนับสนุน **JAAS** (Java Authentication and Authorization Service) ในตัวของมันเอง ทำให้โปรแกรมประยุกต์ที่เขียนขึ้นเพื่อรันบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์จะมีระบบจัดการผู้ใช้เหล่านี้ด้วยโดยไม่ต้องพัฒนาเอง

J2EE vs .NET

แนวคิดที่ใช้แอพพลิเคชั่นเซิร์ฟเวอร์ เป็นแนวคิดที่กำลังทำให้โปรแกรมประยุกต์ที่ใช้งานในองค์กรก้าวเข้าสู่ยุคใหม่ อันเป็นยุคที่เครือข่ายมีความซับซ้อนมากขึ้น แต่การพัฒนาโปรแกรมง่ายลง โดยเฉพาะอย่างยิ่งการเกิดขึ้นของสิ่งที่เรียกว่าบริการผ่านเวบซึ่งจะทำให้มีการเชื่อมโยงข้อมูลข้ามองค์กรมากขึ้น ทำให้การอาศัยแอพพลิเคชั่นเซิร์ฟเวอร์ในการลดความยุ่งยากในการพัฒนาระบบกำลังกลายเป็นความจำเป็นที่หลีกเลี่ยงไม่ได้

ค่ายเทคโนโลยีที่ทรงอิทธิพลที่สุดที่กำลังผลักดันแนวคิดเรื่องแอพพลิเคชั่นเซิร์ฟเวอร์มีสองค่ายได้แก่ J2EE ค่ายจาวา และ .NET ของค่ายไมโครซอฟท์ ว่ากันว่าการพัฒนาโปรแกรมแบบเก่าๆ ทั้งหมดที่ใช้กันอยู่ในระดับองค์กรจะถูกแทนที่ด้วยเทคโนโลยีของสองค่ายนี้ในที่สุด ว่ากันว่าในสหรัฐฯ คนที่กำลังเริ่มเรียนภาษาคอมพิวเตอร์เพื่อจะไปเป็นโปรแกรมเมอร์ระดับองค์กรในอนาคต จะเลือกเรียนแต่ภาษาจาวาหรือไม่ก็ไมโครซอฟท์วิชวลเบสิก (สำหรับ .NET) เท่านั้น

ทั้ง J2EE และ .NET มีความคล้ายคลึงกันมาก จะบอกว่าทั้งสองค่ายพยายามลอกเลียนแบบกันและกันก็ว่าได้ แต่ข้อได้เปรียบของ J2EE ก็คือมีอยู่บนหลายระบบปฏิบัติการทั้งไมโครซอฟท์วินโดว์ ลีนุกซ์ และยูนิกส์ของหลายๆ ค่าย จึงเป็นที่นิยมในองค์กรใหญ่ๆ ในสหรัฐฯ เพราะองค์กรเหล่านี้มีระบบปฏิบัติการเดิมที่ใช้อยู่ก่อนแล้วหลายระบบ ทำให้นำจาวาเข้าไปใช้ในองค์กรเหล่านี้ได้ง่าย ในขณะที่ .NET จะต้องพัฒนาบนระบบปฏิบัติการของไมโครซอฟท์เท่านั้น จึงอาจจะเหมาะกับองค์กรขนาดเล็กหรือองค์กรเกิดใหม่ที่ใช้แต่ระบบปฏิบัติการของไมโครซอฟท์ล้วนๆ อย่างไรก็ดีทั้งโปรแกรมประยุกต์ที่พัฒนาด้วย J2EE และ .NET สามารถสื่อสารกับข้ามระบบปฏิบัติการผ่านระบบเครือข่ายได้ทั้งคู่

J2EE กับประเทศไทย

การพัฒนาโปรแกรมประยุกต์ระดับองค์กรด้วย J2EE ไม่ใช่เรื่องใหม่สำหรับองค์กรในสหรัฐฯ แต่ยังนับว่าเป็นเรื่องใหม่มากสำหรับองค์กรในประเทศไทย โปรแกรมเมอร์ที่มีความรู้เกี่ยวกับเรื่องนี้ยังมีอยู่น้อยมาก ในขณะที่อุตสาหกรรมพัฒนาโปรแกรมประยุกต์ระดับองค์กรก็เป็นหนึ่งในสามอุตสาหกรรมประเภทซอฟท์แวร์ที่ประเทศไทยมีศักยภาพที่จะพัฒนาให้แข่ง

ขันกับโลกได้ (อีกสองอันได้แก่ มัลติมีเดีย และซอฟท์แวร์ไร้สาย) การหันมาสนใจที่จะสร้างความรู้ความชำนาญเกี่ยวกับ J2EE จึงเป็นตัวเลือกที่น่าสนใจอยู่ไม่น้อยสำหรับคนไทยที่สนใจอยากยึดอาชีพเป็นโปรแกรมเมอร์ระดับองค์กร

2

ติดตั้ง J2EE server

ในบทนี้เราจะติดตั้งและทดสอบจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ ไว้ใช้สำหรับการทดสอบโปรแกรมตัวอย่างในหนังสือเล่มนี้ จาวาแอพพลิเคชั่นเซิร์ฟเวอร์ที่จะใช้ในหนังสือเล่มนี้มีชื่อว่า **J2EE server**

เตรียมอุปกรณ์

ก่อนอื่นสิ่งที่คุณจะต้องมีก็คือ เครื่องพีซีที่ใช้ตัวประมวลผลที่มีความเร็วไม่ต่ำกว่า 350 MHz มีหน่วยความจำหลักไม่ต่ำกว่า 256 เมกกะไบต์ และมีเนื้อที่ว่างบนฮาร์ดดิสก์อย่างน้อยอีก 250 เมกกะไบต์ มีหัวอ่านซีดีรอม และโมเด็มสำหรับต่อไปยังเครือข่ายอินเตอร์เนต

ระบบปฏิบัติการที่ใช้บนเครื่องต้องเป็นไมโครซอฟท์วินโดว์ 98/Me/2000/XP และที่สำคัญจะต้องมี J2SE SDK เวอร์ชั่น 1.3.1 หรือสูงกว่า ติดตั้งอยู่แล้วบนเครื่อง อันที่จริง J2EE server มีทั้งที่ใช้ได้บนระบบปฏิบัติการลีนุกซ์ และโซราริสด้วย แต่ในหนังสือเล่มนี้จะสาธิตการใช้งานบนไมโครซอฟท์วินโดว์เท่านั้นเพราะเป็นระบบปฏิบัติการที่หาได้ง่ายที่สุด

J2EE server ถูกออกแบบมาเพื่อการใช้งานในองค์กร จึงเหมาะกับเครื่องคอมพิวเตอร์ที่มีสเปกค่อนข้างสูงอย่างเครื่องเซิร์ฟเวอร์ไม่เหมาะกับเครื่องพีซี ดังนั้นคุณจะประสบปัญหาเป็นอย่างมากหากเครื่องทดสอบของคุณเป็นเครื่องพีซีที่มีสเปกต่ำกว่าที่แนะนำข้างต้น โดยเฉพาะอย่างยิ่งในเรื่องของขนาดของหน่วยความจำหลัก เพราะ J2EE server มีระบบการจัดการหน่วยความจำสำหรับหน่วยความจำขนาดใหญ่ การนำมาทดสอบบนเครื่องที่มีหน่วยความจำขนาดเล็กจะทำให้ระบบทำงานช้ากว่าปกติ

ดาวน์โหลด J2EE server

J2EE server พ่วงมากับ J2EE SDK ซึ่งสามารถดาวน์โหลดมาใช้ได้ฟรีจากเวบไซต์ของจาวาชื่อ http://java.sun.com/j2ee ในหนังสือเล่มนี้จะใช้ J2EE server เวอร์ชั่น 1.3.1 ที่มากับ J2EE SDK 1.3.1 ซึ่งแม้ว่าจะเป็นเวอร์ชั่นเก่าแต่ก็เป็นเวอร์ชั่นที่มีขนาดกะทัดรัด และมีความสามารถเพียงพอสำหรับการศึกษาโปรแกรมตัวอย่างทั้งหมดในหนังสือเล่มนี้แล้ว ไม่แนะนำให้ใช้เวอร์ชั่นที่ใหม่กว่านี้ เพราะจะทำให้มีปัญหาเวลาทดสอบโปรแกรมตัวอย่างในหนังสือเนื่องจากเมนูต่างๆ ของโปรแกรมมีหน้าตาไม่เหมือนกันในแต่ละเวอร์ชั่น

คุณอาจไปที่ http://java.sun.com/j2ee/1.3/download.html เพื่อดาวน์โหลด J2EE SDK 1.3.1 FCS Release เองก็ได้ แต่เราได้ดาวน์โหลดไว้ให้คุณแล้วในแผ่นซีดีรอมที่แถมมากับหนังสือที่ไฟล์ชื่อ D:\j2ee\j2sdkee-1_3_1-win.exe (สมมติว่าหัวอ่านซีดีรอมของคุณอยู่ที่ไดร์ฟ D:) ไฟล์นี้มีขนาด 16.5 MB

กรณีที่คุณยังไม่มี J2**S**E SDK ติดตั้งไว้ก่อนบนเครื่องของคุณ ให้คุณไปที่ http://java.sun.com/j2se/1.4.2/index.jsp เพื่อดาวน์โหลด J2SE SDK 1.4.2 ซึ่งเป็นเวอร์ชั่นที่แนะนำให้ใช้กับ J2EE SDK 1.3.1 ด้วยหรือจะใช้ J2SE SDK 1.4.2 ที่อยู่ในซีดีรอมก็ได้ (อยู่ที่ไฟล์ D:\j2se\j2sdk-1_4_2_03-windows-i586-p.exe) โดยต้องติดตั้งให้เรียบร้อยก่อนก่อนที่จะติดตั้ง J2EE SDK ในขั้นตอนต่อไป

ติดตั้ง J2EE SDK

ดับเบิลคลิกไฟล์ j2sdkee-1_3_1-win.exe เพื่อเริ่มการติดตั้ง J2EE SDK 1.3.1 ตอบคำถามของโปรแกรมติดตั้งโดยเลือกตัวเลือกปกติที่ตัวติดตั้งแนะนำและคลิก Next ไปเรื่อยๆ จนจบที่

Finish เมื่อจบแล้ว J2EE SDK 1.3.1 จะถูกติดตั้งอยู่ที่โฟลเดอร์ C:\j2sdkee1.3.1 บนเครื่องของคุณ

จากนั้นเซตตัวแปรแวดล้อมโดยไปที่ Control Panel> System> Advanced> Environment Variables> ในช่องของ User Variables คลิก New เพื่อสร้างตัวแปรผู้ใช้ใหม่สองตัวดังนี้

ตารางที่ 2-1 User Variables ที่ต้องเซต

Variable	Variable Value
CLASSPATH	.;C:\j2sdk1.3.1\lib\j2ee.jar
JAVA_HOME	C:\j2sdk1.4.2_03
J2EE_HOME	C:\j2sdkee1.3.1

ตัวแปร CLASSPATH เป็นตัวแปรที่ใช้บอกคอมไพเลอร์ภาษาจาวาว่าจะหาแพจเกจที่จำเป็นสำหรับการคอมไพล์(ถ้ามี) ได้ที่ไหน เครื่องหมาย . หมายถึงให้หาที่โฟลเดอร์ปัจจุบัน ส่วน C:\j2sdk1.3.1\lib\j2ee.jar คือให้หาที่ไฟล์ j2ee.jar ด้วย ซึ่งจำเป็นเพราะไฟล์นี้เป็นไฟล์ที่บรรจุแพจเกจมาตรฐานของ J2EE ทั้งหมดที่จำเป็นในการคอมไพล์โปรแกรมที่ใช้ชุดคำสั่ง J2EE เอาไว้

ส่วนตัวแปร JAVA_HOME และ J2EE_HOME มีไว้บอก J2EE Server ว่า J2SE SDK และ J2EE SDK บนเครื่องนี้ถูกติดตั้งไว้ที่ใดตามลำดับ ถ้าคุณไม่ได้ติดตั้ง J2SE SDK และ J2EE SDK ไว้ในโฟลเดอร์ระบุไว้ในตารางที่ 2-1 ให้เปลี่ยนค่าของตัวแปรให้ตรงกับชื่อโฟลเดอร์บนเครื่องของคุณ อย่าลืมคลิก OK ทุกครั้งเมื่อเซตตัวแปรเหล่านี้เสร็จ

มีตัวแปรอีกตัวที่ต้องเซตคือตัวแปร Path ซึ่งเป็นตัวแปรระบบที่มีค่าเดิมของมันอยู่ก่อนแล้ว มองหาตัวแปร Path ในช่อง System Variables ใช้เมาส์เลือกตัวแปร Path แล้วคลิก Edit เพื่อแก้ไข โดยให้คงค่า Variable ไว้ ส่วน Variable Value ก็ให้คงค่าเดิมไว้แต่เติมต่อท้ายด้วย C:\j2sdk1.4.2_3\bin;C:\j2sdkee1.3.1\bin อย่าตัดของเดิมที่มีอยู่แล้วทิ้ง ตัวแปร Path ช่วยให้ดอสค้นหาที่อยู่ของคำสั่งต่างๆ ได้อย่างถูกต้อง การใส่ชื่อโฟลเดอร์ \bin ทั้งสองนี้ลงไปทำให้ดอสเรียกคำสั่งต่างๆ ของ J2SE SDK และ J2EE SDK ได้

วิธีการเซตตัวแปรระบบข้างต้นเป็นวิธีที่ใช้กับไมโครซอฟท์วินโดว์ XP และ 2000 ถ้าคุณใช้ 98 หรือ Me ให้ตรวจสอบวิธีเซตตัวแปรระบบได้ใน http://www.dekisugi.net/java/support

เรียกใช้งาน J2EE server

ตอนนี้ทดลองเรียกใช้งาน J2EE server ด้วยการเรียกหน้าต่างดอสออกมา ทดสอบดูก่อนว่าตัวแปรระบบที่เซตไว้ก่อนหน้านี้เซตถูกต้องหรือไม่ด้วยคำสั่ง

```
C:\>set JAVA_HOME
JAVA_HOME=C:\j2sdk1.4.2_03

C:\>set J2EE_HOME
J2EE_HOME=C:\j2sdkee1.3.1

C:\>set CLASSPATH
CLASSPATH=.;C:\j2sdkee1.3.1\lib\j2ee.jar

C:\>set Path
Path= C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;
C:\j2sdk1.4.2_03\bin;C:\j2sdkee1.3.1\bin;
```

เมื่อตรวจสอบดูพบว่าถูกต้องแล้ว ให้สตาร์ท J2EE server ขึ้นมาด้วยคำสั่ง

```
C:\>j2ee -verbose
```

พารามิเตอร์ –verbose ไม่จำเป็นต้องมีก็ได้ แต่ขอแนะนำให้มี เพราะเป็นการสั่งให้ J2EE server ที่สตาร์ทขึ้นมา คอยรายงานเราอยู่เสมอว่ามีเหตุการณ์อะไรเกิดขึ้นบนเซิร์ฟเวอร์บ้าง ซึ่งจะทำให้เราตรวจสอบความผิดพลาดต่างๆ ที่อาจเกิดขึ้นได้ง่าย

เมื่อ J2EE server สตาร์ทเสร็จแล้ว จะขึ้นคำว่า J2EE server startup complete ห้ามปิดหน้าต่างดอสนี้ตลอดการทำงานของ J2EE server

ทดสอบ J2EE server

มาถึงขั้นตอนนี้ คุณมี J2EE server ซึ่งเป็นจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ทำงานอยู่แล้วบนเครื่องของคุณ ลำพังตัวจาวาแอพพลิเคชั่นเปล่าๆ ทำอะไรไม่ได้ คุณต้องเขียนโปรแกรมภาษาจาวาขึ้นมาแล้วนำไปรันบน J2EE server อีกที เพื่อให้ J2EE server ทำงานอะไรก็ตามที่คุณอยากให้ทำ

เราจะทดสอบว่า J2EE server ของเราทำงานได้จริงหรือไม่ ด้วยการทดลองติดตั้งโปรแกรมภาษาจาวาตัวอย่างที่มีชื่อว่า HelloWorld ลงไป โปรแกรมนี้เป็นโปรแกรมเล็กๆ ที่ไม่ทำอะไรนอกจากเขียนคำว่า HelloWorld ให้คุณดู

โปรแกรมนี้มีพร้อมอยู่แล้วเป็นไฟล์ๆ หนึ่งในซีดีรอมชื่อว่า HelloWorldApp.ear ในโฟลเดอร์ D:\examples\ears ปกติโปรแกรมภาษาจาวาที่พร้อมจะรันบน J2EE server จะอยู่ในรูปของไฟล์นามสกุล .ear เสมอ (ในบทนี้เรายังไม่สนใจว่าเราจะสร้างไฟล์นี้ขึ้นมาได้อย่างไร)

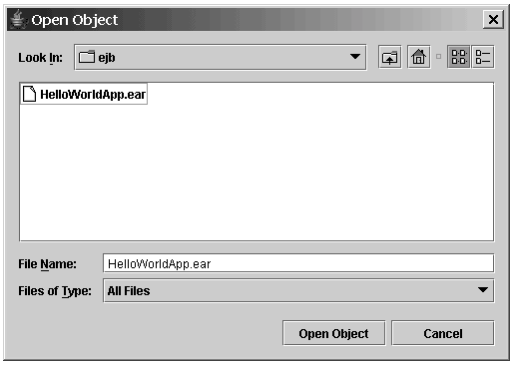
การจะติดตั้งโปรแกรมภาษาจาวาอะไรก็ตามลงบน J2EE server เราจะสั่งงานผ่านโปรแกรมช่วยเหลือตัวหนึ่งมีชื่อว่า J2EE Application Deployment Tool ซึ่งเรียกใช้งานโดยใช้คำสั่ง (คุณต้องเปิดหน้าต่างดอสขึ้นมาใหม่อีกหน้าต่างหนึ่ง)

```
C:\> deploytool
```

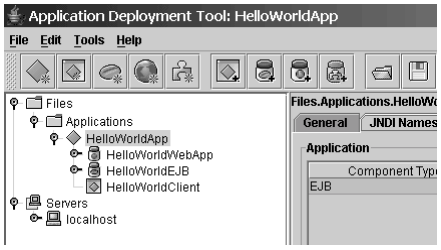
รอสักครู่โปรแกรมจะสตาร์ทขึ้นมา โปรแกรม J2EE Application Deployment Tool หรือ deploytool นี้เป็นโปรแกรมที่มีส่วนติดต่อกับผู้ใช้เป็นแบบกราฟฟิค

เราใช้โปรแกรมนี้ ติดต่อกับ J2EE server เพื่อสั่งการทุกอย่างเกี่ยวกับ J2EE server เพราะเราติดต่อกับ J2EE server โดยตรงไม่ได้ ตอนนี้เราจะติดตั้งโปรแกรม HelloWorld ลงบน J2EE server ผ่านโปรแกรมตัวนี้

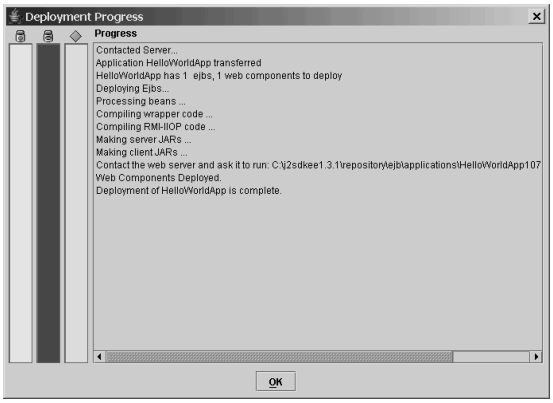
เมื่อพร้อมแล้วไปที่เมนู File>Open แล้วเข้าไปในไดร์วซีดีรอม เลือกไฟล์ที่ชื่อ HelloWorldApp.ear ในโฟลเดอร์ examples\ears



จากนี้คลิก Open Object จะเห็นโปรแกรม HelloWorldApp ปรากฏขึ้นที่หน้าต่างด้านซ้ายของหน้าจอดังภาพ



ใช้เมาส์คลิกเพื่อเลือกที่ HelloWorldApp แล้วกดปุ่ม (Deploy) เพื่อโหลดโปรแกรม HelloWorldApp ลงบน J2EE server คลิก Finish เลย ไม่ต้องสนใจตัวเลือกใดๆ



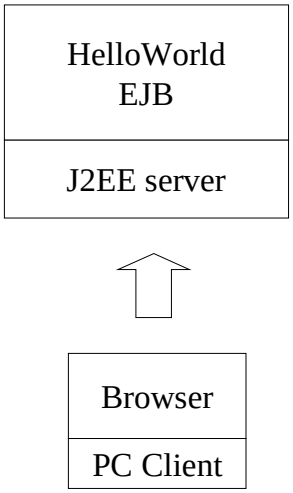
รอให้กระบวนการโหลดเสร็จสิ้น แล้วจึงคลิก OK ออกมา

ตอนนี้การโหลดเสร็จสมบูรณ์แล้ว โปรแกรม HelloWorld พร้อมจะทำงาน ผู้ใช้สามารถติดต่อกับโปรแกรมได้โดยผ่านทางเบราเซอร์

เรียกอินเตอร์เนตเอ็กซ์พลอเรอร์ หรือเบราเซอร์อะไรก็ได้ที่อยู่บนเครื่องของคุณ แล้วพิมพ์ที่อยู่ URL ว่า http://localhost:8000/hello เบราเซอร์ควรแสดงคำว่า HelloWorld ดังภาพ

อย่างนี้แสดงว่าโปรแกรม HelloWorld ทำงานได้จริงๆ และ J2EE server ก็ติดตั้งได้อย่างถูกต้องแล้วบนเครื่องของคุณ ในที่นี้เราทดสอบโปรแกรมโดยให้เซิร์ฟเวอร์กับไคลน์เอนท์(เบราเซอร์)อยู่บนเครื่องเดียวกัน แต่ความจริงแล้วเซิร์ฟเวอร์กับเบราเซอร์จะอยู่บนคนละเครื่องก็ได้ เพราะเซิร์ฟเวอร์กับเบราเซอร์ติดต่อกันได้ผ่านระบบเครือข่าย ถ้าต้องการทดสอบข้าม

เครื่องให้ใช้เบราเซอร์จากเครื่องอื่นติดต่อเข้ามาโดยเปลี่ยนคำว่า localhost ใน URL Address ให้เป็น ไอพีแอดเดรส ของเครื่องที่รันเซิร์ฟเวอร์อยู่แทน

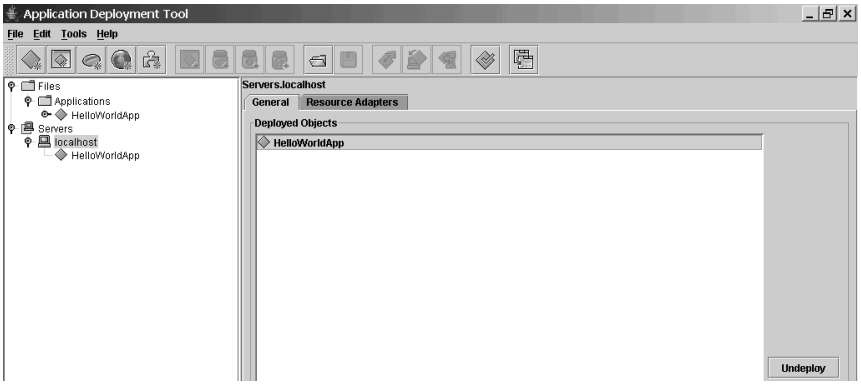


การทดสอบโปรแกรมตัวอย่างในหนังสือเล่มนี้จะใช้พีซีเครื่องเดียวในการทดสอบทั้งหมด แต่ขอให้เข้าใจว่าความจริงแล้วเซิร์ฟเวอร์กับไคลน์เอนท์จะอยู่บนคนละเครื่องที่ติดต่อถึงกันผ่านระบบเครือข่ายก็ได้

ปลดโปรแกรม HelloWorld

ก่อนจากกันในบทนี้ให้คุณปลดโปรแกรม HelloWorld ออกก่อน

ไปที่หน้าต่างซ้ายเลือก HelloWorldApp ใต้ localhost แล้วไปที่หน้าต่างด้านขวาเลือก HelloWorldApp อีกที จากนั้นคลิก Undeploy



โปรแกรมจะถามยืนยันว่าต้องการปลด HelloWorldApp ออกจาก J2EE server แน่หรือไม่ ให้ตอบ Yes

คราวนี้กลับไปเลือก HelloWorldApp ใต้ Files ที่หน้าต่างด้านซ้ายแล้วเลือก File>Close การปลด HelloWorldApp ก็จะเสร็จสมบูรณ์

จบการใช้งาน J2EE server

เวลาจะเลิกใช้งาน J2EE server ให้เปิดหน้าต่างดอสขึ้นมาอีกหน้าต่างหนึ่งแล้วใช้คำสั่งดังนี้

```
C:\>j2ee -stop
```

แล้วรอให้หน้าต่างเดิมกลับมาอยู่ที่ C:\> จึงปิดหน้าต่างดอสได้ อย่าหยุด J2EE server กะทันหันด้วยการปิดหน้าต่างดอสไปเฉยๆ โดยไม่สั่งคำสั่ง j2ee –stop ก่อนโดยเด็ดขาด เพราะอาจทำให้เกิดความเสียหายกับ J2EE server ได้

3

บีนบริการแบบไม่คงสถานะ

อย่างที่ได้เกริ่นไปบ้างแล้วในบทแรกว่าโปรแกรมภาษาจาวาที่ใช้รันบนจาวาแอพพลิเคชั่นจะเขียนให้อยู่ในรูปของ EJB (Enterprise Java Beans) หรือ บีน (ต่อไปนี้ขอเรียกสั้นๆ ว่า บีน) ในบทนี้เราจะเรียนรู้การวิธีการสร้างบีน ผ่านบีนชนิดแรกที่มีเรียกว่า **บีนบริการ** (Session Bean) ซึ่งเป็นบีนชนิดที่มีความซับซ้อนน้อยที่สุด

บีน

มาทำความรู้จักกันก่อนว่า บีน คืออะไร?

บีน คือ โปรแกรมภาษาจาวาขนาดเล็ก (คอมโพเนนท์) บนเครื่องแม่ข่าย (เซิร์ฟเวอร์) ที่เขียนขึ้นมาเพื่อทำหน้าที่เฉพาะเจาะจงอย่างใดอย่างหนึ่ง บีน มีความเป็นอิสระในตัวเอง เพราะมันสามารถทำงานอยู่อย่างโดดเดี่ยวบนเซิร์ฟเวอร์ หรือจะทำงานประสานกับบีนตัวอื่นบนเซิร์ฟเวอร์เดียวกัน หรือข้ามเซิร์ฟเวอร์กันก็ได้

ประโยชน์ของการใช้บีน

โปรแกรมประยุกต์ในระดับองค์กรโดยเฉพาะโปรแกรมขนาดใหญ่ที่มีความซับซ้อนมาก มักประกอบด้วยบีนจำนวนมากทำงานประสานกันอยู่ภายใน โครงสร้างที่ประกอบด้วยชิ้นส่วนย่อยแบบนี้ทำให้โปรแกรมมีความยืดหยุ่นสูง ทั้งในแง่ของการตอบสนองความต้องการของผู้ใช้จำนวนมาก และในแง่ของการพัฒนา J2EE จึงกำหนดให้โปรแกรมทุกโปรแกรมที่จะรันบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ได้ต้องเขียนให้อยู่ในรูปของบีนเท่านั้น ไม่ว่าโปรแกรมนั้นจะทำงานเล็กน้อยหรือใหญ่โตแค่ไหนก็ตาม

ด้วยเหตุที่โปรแกรมภาษาจาวาระดับองค์กรประกอบด้วยบีน เวลาที่มีจำนวนผู้ใช้โปรแกรมเพิ่มมากขึ้น เราสามารถทำให้โปรแกรมที่เราสร้างขึ้นมีความสามารถในการรองรับงานที่เพิ่มมากขึ้นเรื่อยได้ง่ายๆ ด้วยการรันอินสแตนท์ของบีนตัวเดียวกันหลายๆ ตัวบนเซิร์ฟเวอร์ เพื่อให้อินสแตนท์เหล่านั้นช่วยกันตอบสนองความต้องการของผู้ใช้ที่เข้ามาในระบบ ถ้าเซิร์ฟเวอร์เครื่องเดียวไม่พอ ก็สามารถใช้เซิร์ฟเวอร์หลายเครื่องที่รันอินสแตนท์ของบีนตัวเดียวกันช่วยกันทำงานเหมือนกับเป็นเครื่องเซิร์ฟเวอร์ขนาดใหญ่เครื่องเดียวได้อีกด้วย

ในแง่ของการพัฒนา แนวคิดเรื่องบีนทำให้การแบ่งงานระหว่างโปรแกรมเมอร์ในทีมพัฒนาทำได้ง่ายขึ้น เพราะบีนเป็นหน่วยของโปรแกรมอิสระ จึงสามารถมอบหมายให้โปรแกรมเมอร์แต่ละคนรับผิดชอบบีนเป็นตัวๆ ไปได้เลย โปรแกรมเมอร์เพียงแต่ต้องทราบเพิ่มเติมว่าบีนของโปรแกรมเมอร์คนอื่นๆ ในทีมทำอะไรได้บ้าง และจะเรียกใช้งานได้อย่างไรก็พอ ไม่จำเป็นต้องไปรู้ไปเห็นว่าคำสั่งภายในบีนของคนอื่นมีว่าอย่างไร

แนวคิดเรื่อง Thin-client

การเขียนโปรแกรมบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ที่ดีควรให้บีนซึ่งทำงานอยู่บนฝั่งเซิร์ฟเวอร์ทำงานส่วนใหญ่ของโปรแกรม เพราะเซิร์ฟเวอร์เป็นเครื่องคอมพิวเตอร์ขนาดใหญ่ที่มีทรัพยากรระบบมาก ทำงานได้เร็ว การให้บีนทำงานส่วนใหญ่ของ

โปรแกรมจะทำให้ได้ใช่ทรัพยากรของเซิร์ฟเวอร์อย่างเต็มที่ งานที่เกี่ยวข้องกับการแสดงผลออกหน้าจอให้ผู้ใช้งานดูเป็นงานที่ไม่กินทรัพยากรระบบมากควรยกให้เป็นของหน้าที่ของโปรแกรมที่อยู่บนฝั่งเครื่องลูกข่าย (ไคลน์เอนท์) แนวคิดแบบนี้เป็นแนวคิดที่เรียกว่า **Thin-client** กล่าวคือ โปรแกรมบนฝั่งไคลน์เอนท์ไม่ทำอะไรมากนอกจากรับข้อมูลจากเซิร์ฟเวอร์มาแสดงผลอย่างเดียว

แนวคิดแบบ Thin-client ทำให้องค์กรไม่ต้องลงทุนซื้อเครื่องที่มีสเปกสูงๆ เพื่อทำเป็นเครื่องลูกข่าย แต่ให้ทุ่มเงินลงทุนไปกับการซื้อเซิร์ฟเวอร์เครื่องเดียวให้มีสเปกดีๆ ไปเลย เพราะเซิร์ฟเวอร์เป็นทรัพยากรที่แบ่งกันใช้ได้ นอกจากนี้แนวคิดนี้ยังช่วยจำกัดปริมาณข้อมูลที่วิ่งบนเครือข่ายข้ามไปมาระหว่างเซิร์ฟเวอร์กับไคลน์เอนท์ได้ด้วย เพราะไคลน์เอนท์ส่งคำสั่งร้องขอซึ่งโดยทั่วไปมีขนาดเล็กไปหาเซิร์ฟเวอร์เพียงครั้งเดียว เซิร์ฟเวอร์จะประมวลผลอยู่ข้างในตัวจนเรียบร้อยแล้วค่อยส่งผลลัพธ์ทั้งหมดไปให้ไคลน์เอนท์แสดงผลทีเดียว ไม่ต้องมีการส่งข้อมูลข้ามไปข้ามมาหลายครั้งขณะประมวลผลอย่างในกรณีที่ไคลน์เอนท์และเซิร์ฟเวอร์ช่วยกันประมวลผล

ประเภทของบีน

บีนมี 3 ประเภทดังนี้

1. บีนบริการ (Session Bean)
2. บีนวัตถุ (Entity Bean)
3. บีนที่ทำงานด้วยแมสเสจ (Message-Driven Bean)

บีนบริการ

บีนบริการ ถูกสร้างขึ้นเพื่อทำงานบางอย่างให้กับผู้ใช้บนฝั่งไคลน์เอนท์ที่ติดต่อเข้ามาที่เซิร์ฟเวอร์ กล่าวคือ เมื่อมีผู้ใช้ติดต่อเข้ามาในระบบ จาวาแอพพลิเคชั่นเซิร์ฟเวอร์จะสร้างอินสแตนท์ของบีนบริการขึ้นมาหนึ่งตัวเพื่อรับมือกับผู้ใช้รายนั้นและอินสแตนท์จะยังคงมีชีวิตอยู่จนกว่าผู้ใช้จะเลิกติดต่อเข้ามา ก่อนที่จะถูกเซิร์ฟเวอร์ทำลายเพื่อให้เกิดพื้นที่ว่างในหน่วยความจำหลัก

ลักษณะที่สำคัญของบีนบริการ คือ สถานะของอินสแตนท์นั้นจะคงอยู่ตราบเท่าที่อินสแตนท์นั้นยังมีชีวิตอยู่ และหายไปเมื่ออินสแตนท์ถูกทำลาย ไม่มีการเก็บบันทึกสถานะเพื่อเอาไว้อ้างอิงถึงได้ในภายหลัง บีนชนิดนี้จึงเหมาะสำหรับการใช้งานง่ายๆ ที่ไม่ต้องมีการบันทึกข้อมูลไว้เพื่ออนาคต ตัวอย่างเช่น ระบบรถเข็นอิเล็กทรอนิกส์ในร้านค้าออนไลน์มักสร้างด้วยบีนบริการ เป็นต้น เพราะเมื่อลูกค้าซื้อของเสร็จหรือเปลี่ยนใจผละจากร้านไป รถเข็นอิเล็กทรอนิกส์ของลูกค้ารายนั้นก็หมดหน้าที่ลงทันที

คำว่า สถานะของอินสแตนท์ หมายถึง ค่าของตัวแปรคลาสของอินสแตนท์นั้นๆ

ชนิดของบีนบริการ

บีนบริการยังแบ่งย่อยออกเป็นสองชนิดคือ

1. บีนบริการแบบคงสถานะ (Stateful Bean)
2. บีนบริการแบบไม่คงสถานะ (Stateless Bean)

บีนบริการแบบคงสถานะ คือ อินสแตนท์ของบีนแบบนี้จะถูกสร้างขึ้นเพื่อจัดการกับผู้ใช้ที่เข้ามาแบบ*ตัวต่อตัว* สามารถจดจำสถานะต่างๆ ของผู้ใช้ได้ (เช่น ผู้ใช้หยิบสินค้าอะไรใส่รถเข็นบ้าง) ตลอดเวลาที่ผู้ใช้รายนั้นยังติดต่อกับเซิร์ฟเวอร์อยู่ และจะหายไปเมื่อผู้ใช้จากไป ซึ่งโดยทั่วไปกินเวลาไม่เกินหนึ่งถึงสองชั่วโมง รถเข็นอิเล็กทรอนิกส์สร้างขึ้นจากบีนแบบนี้

บีนบริการแบบไม่คงสถานะ คือ บีนบริการที่ไม่มีการจดจำสถานะต่างๆ ของผู้ใช้ที่เข้ามา ดังนั้นในเวลาเดียวกันบีนบริการแบบไม่คงสถานะตัวเดียวสามารถรับมือผู้ใช้ได้*มากกว่าหนึ่งคน* เพราะมันไม่ต้องสนใจที่จะจดจำข้อมูลจำเพาะของผู้ใช้แต่ละคน บีนแบบนี้เหมาะสำหรับงานที่ง่ายมากๆ อย่างเช่น โปรแกรม HelloWorld ในบทที่แล้วก็ใช้บีนไม่มีสถานะ เพราะไม่ว่าผู้ใช้คนใดจะติดต่อเข้ามา โปรแกรมก็ไม่ทำอะไรพิเศษไปกว่าการเขียนคำว่า HelloWorld!

ในแง่ของความประสิทธิภาพในการทำงาน บีนบริการแบบไม่คงสถานะจะทำงานได้เร็วกว่าบีนบริการแบบคงสถานะ และกินเนื้อที่หน่วยความจำน้อยกว่า เพราะบีนตัวเดียวรับงานจากผู้ใช้ได้หลายคนในเวลาเดียวกัน ดังนั้นถ้างานของคุณเหมาะที่จะใช้บีนบริการแต่ไม่จำเป็นต้องมีการจดจำสถานะของผู้ใช้ ก็ควรเลือกใช้บีนบริการแบบไม่คงสถานะ คุณควรจะใช้บีนบริการแบบคงสถานะก็ต่อเมื่อจำเป็นต้องมีการจำจดสถานะบางอย่างของผู้ใช้เท่านั้น

บีน HelloWorld

ในบทนี้เราจะมาหัดเขียนบีนตัวแรกในชีวิตของเรา ซึ่งก็คือโปรแกรม HelloWorld ที่ใช้ทดสอบ J2EE server ไปในบทที่แล้ว โปรแกรม HelloWorld สร้างขึ้นจากบีนบริการแบบไม่คงสถานะเพียงตัวเดียว จึงเป็นบีนที่เขียนได้ง่าย

แต่เนื่องจากคุณยังไม่เคยสร้างบีนมาก่อน ดังนั้นในบางตอนคุณอาจสงสัยว่าทำไมต้องทำอย่างนั้นอย่างนี้ ขอให้พยายามทำตามขั้นตอนไปเรื่อยๆ ก่อน และคุณจะเริ่มเข้าใจทุกอย่างเองในภายหลัง

HelloWorldBean

ลองพิจารณาโปรแกรม HelloWorldBean.java ข้างล่างนี้

โปรแกรมที่ 3-1 HelloWorldBean.java

```
package hello;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class HelloWorldBean implements SessionBean {

    public String greet() {
        return "Hello World!";
    }

    public HelloWorldBean() {}
    public void ejbCreate() {}
    public void ejbRemove() {}
    public void ejbActivate() {}
    public void ejbPassivate() {}
    public void setSessionContext(SessionContext sc) {}
```

```
}
```

โปรแกรมภาษาจาวาที่เขียนให้เป็นบีนจะมีข้อบังคับหลายอย่างที่ทำให้โปรแกรมมีความแตกต่างจากโปรแกรมภาษาจาวามาตรฐาน

บีนบริการต้องนิยามให้เป็นคลาสที่สืบทอดอินเตอร์เฟส SessionBean ซึ่งอยู่ในแพจเกจ javax.ejb เสมอ อินเตอร์เฟส SessionBean นี้มีแมธธอสนามธรรม 5 ตัวที่คลาสใดก็ตามที่สืบทอดอินเตอร์เฟส SessionBean จะต้องประกาศ ได้แก่ ejbCreate(), ejbRemove(), ejbActivate(), ejbPassivate() และ setSessionContext(SessionContext sc) จะเห็นได้ว่าแมธธอสทั้งห้าที่ประกาศในคลาส HelloWorldBean ไม่มีเนื้อความ แมธธอสเหล่านี้มีไว้ให้จาวาแอพพลิเคชั่นนำไปใช้ในการบริหารจัดการบีน เช่น การสร้างหรือลบบีนใหม่ ซึ่งจาวาแอพพลิเคชั่นจะเรียกแมธธอสเหล่านี้เองโดยที่เราไม่ต้องไปยุ่ง แค่ประกาศเอาไว้ในคลาสให้มันก็พอ

นอกจากแมธธอสบังคับทั้งห้าตัวแล้ว ส่วนที่เหลือในบีนก็คือแมธธอสอะไรก็ตามที่เป็นตัวงานของโปรแกรมของเรา บางทีเราเรียกแมธธอสกลุ่มนี้ว่า **แมธธอสธุรกิจ** (Business Method) เพราะเป็นแมธธอสที่เป็นตัวงานของโปรแกรมจริงๆ และเป็นแมธธอสที่จะแตกต่างกันไปในแต่ละบีน ในที่นี้มีแมธธอสธุรกิจแค่แมธธอสเดียว คือ greet() แมธธอสนี้ไม่ได้ทำอะไรนอกจากส่งคำว่า HelloWorld ไปให้ผู้ใช้เท่านั้น ในบีนที่ใช้งานจริงๆ จะมีแมธธอสธุรกิจเป็นจำนวนมาก

HelloWorld

ตอนนี้เรามีโปรแกรม HelloWorldBean ซึ่งคือบีนของเราแล้ว แต่การสร้างบีนของเรายังไม่เสร็จสมบูรณ์ เพราะ J2EE กำหนดไว้ว่าไคลน์เอนท์จะเข้าถึงบีนโดยตรงไม่ได้ แต่จะต้องผ่านตัวกลางที่เรียกว่า **อินเตอร์เฟส** (อย่าสับสนกับคำว่า อินเตอร์เฟส ที่สืบทอดคลาส ทั้งสองคำเป็นคำเดียวกันแต่คนละความหมาย) โปรแกรม HelloWorld.java ข้างล่างนี้คืออินเตอร์เฟสของ HelloWorldBean

โปรแกรมที่ 3-2 HelloWorld.java

```
package hello;
```

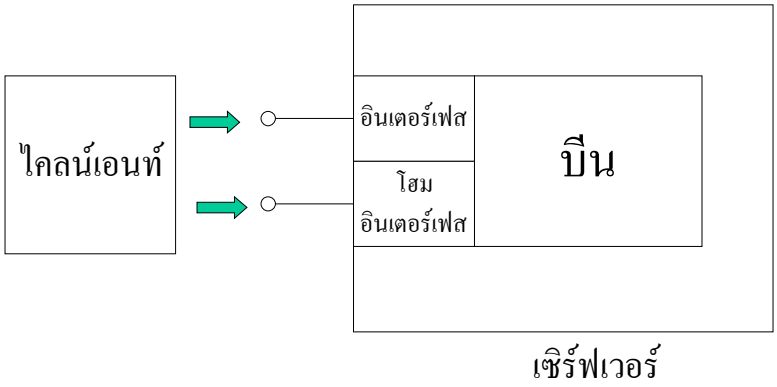
```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface HelloWorld extends EJBObject {

   public String greet() throws RemoteException;
}
```

การเขียนอินเตอร์เฟสให้บีนนั้นง่ายมาก อินเตอร์เฟสเป็นอินเตอร์เฟสที่สืบทอดคลาส javax.ejb.EJBObject

สิ่งที่อยู่ภายในอินเตอร์เฟสไม่มีอะไรเลยนอกจากแมธธอสธุรกิจอะไรก็ตามของบีนที่เราอยากให้ไคลน์เอนท์เห็นเพื่อที่จะสามารถเรียกใช้งานได้ ในที่นี้มีตัวเดียวคือ greet() แมธธอสใดๆ ในอินเตอร์เฟสนี้จะต้องโยน java.rmi.RemoteException เสมอ ส่วนเนื้อหาของแมธธอส ก็ไม่ต้องนำใส่ไว้อีก ไคลน์เอนท์ต้องการรู้แค่ว่าจะเรียกแมธธอสเหล่านี้ได้อย่างไรก็พอ



การที่ไคลน์เอนท์ไม่เข้าถึงบีนโดยตรงแต่ผ่านอินเตอร์เฟส เป็นการซ่อนความซับซ้อนของบีนเอาไว้ไม่ให้ไคลน์เอนท์เห็น ไคลน์เอนท์จะรู้แค่ว่ามีแมธธอสอะไรให้เรียก

ใช้งานได้บ้าง แต่ไม่รู้ว่าข้างในแมธธอสเหล่านั้นทำงานอย่างไร ประโยชน์ของการทำแบบนี้ก็คือ ถ้าในอนาคตโปรแกรมเมอร์ที่สร้างบีนต้องการแก้ไขโปรแกรม เขาสามารถแก้ไขที่คลาสบีนได้เลยโดยไม่ต้องห่วงว่าจะมีผลอะไรเกิดขึ้นบ้างกับไคลน์เอนท์ เพราะตราบใดที่ชื่อแมธธอสที่ไคลน์เอนท์เรียกได้ยังเหมือนเดิม เนื้อหาข้างในแมธธอสนั้นจะเปลี่ยนไปอย่างไรก็ได้

HelloWorldHome

ตอนนี้คุณคงคิดว่าเสร็จแล้ว แต่ก็ยังไม่เสร็จอีก นอกจากต้องมีอินเตอร์เฟสแล้ว บีนยังต้องการสิ่งที่เรียกว่า **โฮมอินเตอร์เฟส** อีก หน้าที่ของโฮมอินเตอร์เฟสคือการประกาศแมธธอสมาตรฐานที่ไคลน์เอนท์เรียกเวลาติดต่อเข้ามา ในที่นี้คือแมธธอส create() ดังนี้

โปรแกรมที่ 3-3 HelloWorldHome.java

```
package hello;

import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface HelloWorldHome extends EJBHome {

    HelloWorld create() throws RemoteException, CreateException;
}
```

อินเตอร์เฟส HelloWorldHome คือ โฮมอินเตอร์เฟสของบีน HelloWorldBean โฮมอินเตอร์เฟสจะต้องสืบทอดคลาส javax.ejb.EJBHome แมธธอสที่ต้องประกาศในโฮมอินเตอร์เฟสได้แก่ แมธธอสชื่อ create() ซึ่งโยน java.rmi.RemoteException และ javax.ejb.CreateException ซึ่งเป็นเอ็กซ์เซฟชั่นเกี่ยวกับความผิดพลาดบนเครือข่ายและความผิดพลาดในการสร้างอินสแตนท์ของบีนตามลำดับ

เอาล่ะ ตอนนี้การสร้างบีนก็เสร็จสมบูรณ์แล้ว คุณอาจจะรู้สึกว่ามันค่อนข้างยุ่งยาก แต่ที่จริงแล้วไม่ว่าบีนอะไรก็ตามวิธีสร้างก็จะเหมือนเดิมหมดทุกอย่าง ต่างกันแค่แมธธอสธุรกิจที่เป็นตัวงานจริงๆ เท่านั้น (ในที่นี้ได้แก่ greet())

HelloWorld มีแมธธอสธุรกิจแค่แมธธอสเดียว เพราะเป็นแค่โปรแกรมสาธิตเพื่อให้เห็นภาพของการเขียนบีนเท่านั้น โปรแกรมระดับองค์กรจริงๆ จะมีแมธธอสธุรกิจเป็นจำนวนมาก

คอมไพล์บีน

ตอนนี้ได้เวลาคอมไพล์บีนแล้ว คุณไม่ต้องเสียเวลาพิมพ์โปรแกรมตัวอย่างในหนังสือเล่มนี้เองเพราะเราทำมาให้แล้วในซีดีรอม ก่อนอื่นขอให้คุณก๊อปปี้โฟลเดอร์ชื่อ examples **ทั้ง**โฟลเดอร์ในซีดีรอมไปไว้ที่ C: โปรแกรมตัวอย่างทุกโปรแกรมในหนังสือเล่มนี้อยู่ในโฟลเดอร์นี้ และต่อไปนี้เราจะทำงานส่วนมากในโฟลเดอร์นี้เป็นหลัก

คอมไพล์บีน HelloWorld โดยการเปิดหน้าต่างดอสขึ้นมาแล้วเข้าไปในโฟลเดอร์ examples\src\HelloWorld ที่เป็นที่อยู่ของไฟล์โปรแกรม HellWorld แล้วคอมไพล์โปรแกรม ดังนี้

```
C:\> cd examples\src\HelloWorld

C:\examples\src\HelloWorld> javac –d . HelloWorld.java

C:\examples\src\HelloWorld> javac –d . HelloWorldHome.java

C:\examples\src\HelloWorld> javac –d . HelloWorldBean.java
```

ถ้าคุณไม่สามารถคอมไพล์ได้ ให้ตรวจสอบการเซตตัวแปรระบบต่างๆ ของคุณว่าถูกต้องหรือไม่ โดยเฉพาะอย่างยิ่ง ตัวแปร CLASSPATH เพราะคอมไพล์เลอร์จะมีการเรียกคลาสในแพจเกจ javax.ejb และ java.rmi ซึ่งเก็บไว้ที่ C:\j2sdkee1.3.1\lib\j2ee.jar ซึ่งถ้าคุณเซต CLASSPATH ไว้ไม่ถูกต้องคอมไพล์เลอร์จะหาแพจเกจเหล่านี้ไม่พบ

เมื่อคอมไพล์เสร็จแล้ว จะเกิดโฟลเดอร์ hello ขึ้น เพราะโปรแกรมทั้งสามของเราถูกเรานิยามไว้ให้อยู่ในแพจเกจ hello การใช้ พารามิเตอร์ –d . เป็นการสั่งให้คอมไพล์เลอร์นำไฟล์ .class ที่สร้างเสร็จแล้วไปไว้ในโฟลเดอร์ที่เหมาะสมกับการถูกเรียกใช้งาน ตอนนี้เราก็มีไฟล์ .class ทั้งสามไฟล์ที่พร้อมสำหรับจะนำไปรันบน J2EE server แล้ว

สังเกตว่าต้องคอมไพล์ HelloWorld.java ก่อน HelloWorldHome.java เพราะ ในโปรแกรม HelloWorldHome.java มีการเรียกใช้ HelloWorld ซึ่งมีการนิยามไว้ใน HelloWorld.java ด้วย

แพจเกจบีน

ไฟล์ .class เหล่านี้ไม่สามารถนำไปรันบน J2EE server ได้ทันที แต่ต้องมีการอัดรวมกันให้อยู่ในรูปของไฟล์ .ear ก่อน

เราจะใช้ deploytool สร้างไฟล์ .ear ให้เรา ตอนนี้ให้คุณสตาร์ท J2EE server ขึ้นมาก่อน รอจนการสตาร์ทเสร็จสมบูรณ์แล้วค่อยเรียก deploytool (จากคนละหน้าต่างดอส)

```
C:\> j2ee –verbose
```

```
C:\> deploytool
```

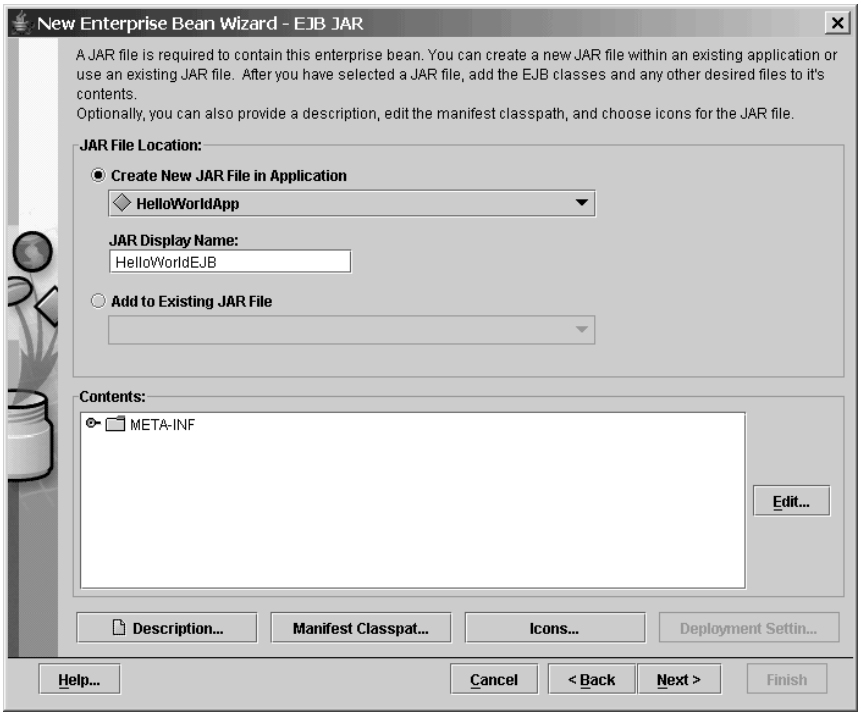
ไปที่ File>New>Application

ในช่อง Application File Name ใส่ว่า C:\examples\src\HelloWorld\HelloWorldApp.ear
ในช่อง Application Display Name ใส่ว่า
HelloWorldApp ดังภาพ แล้วคลิก OK

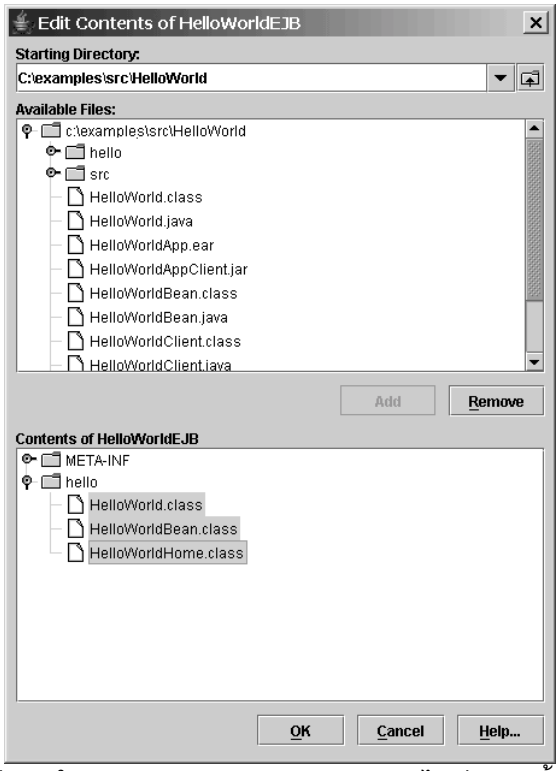
ตอนนี้ที่หน้าต่างซ้ายจะเห็นโปรแกรมประยุกต์เกิดใหม่ อยู่ใต้ Files มีชื่อว่า HelloWorldApp นั้นคือเราได้สร้างโปรแกรมประยุกต์ขึ้นโดยตั้งชื่อว่า HelloWorldApp โปรแกรมนี้จะอยู่ในรูปของไฟล์ชื่อ HelloWorldApp.ear แต่ตอนนี้โปรแกรมนี้ยังว่างอยู่ ประเดี๋ยวเราจะอัดบีนเข้าไป

กดปุ่ม ที่ทูลบาร์ เพื่อสร้างบีนใหม่ บีนที่สร้างใหม่นี้จะเข้าไปอยู่ในโปรแกรมประยุกต์ชื่อ HelloWorldApp ที่เราสร้างไว้ตอนต้น เพราะในหน้าต่างด้านซ้ายเราเลือกโปรแกรม HelloWorldApp อยู่

มีหน้าต่างหนึ่งปรากฏขึ้นมา เพื่ออธิบายว่านี่คือการสร้างบีน ไม่ต้องสนใจให้คลิก Next ไปยังหน้าต่างต่อไปได้เลย

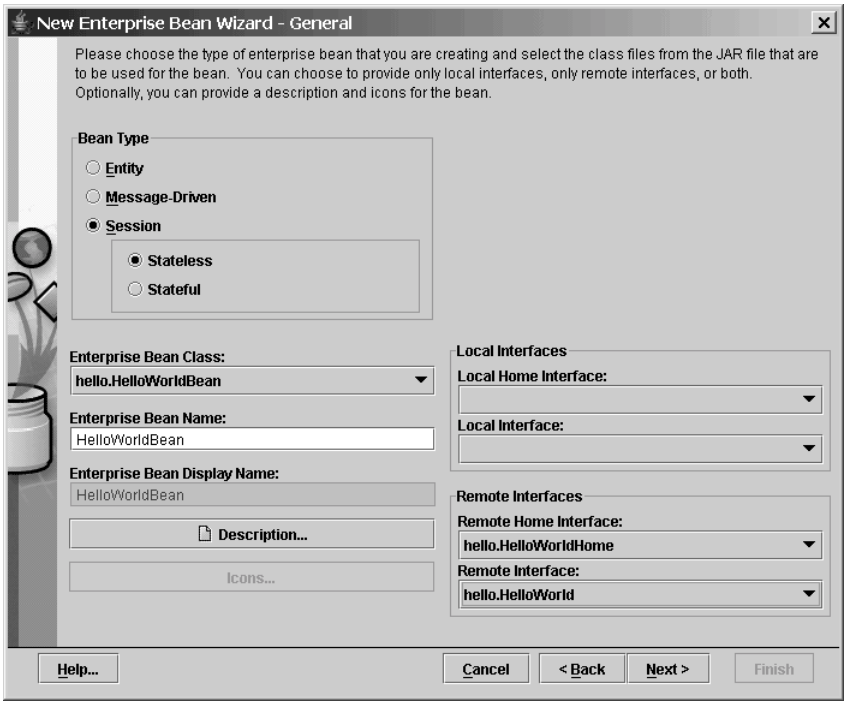


ใส่คำว่า HelloWorldEJB ที่ช่อง JAR Display Name เป็นการตั้งชื่อบีนของเรา จากนั้นคลิกที่ปุ่ม Edit



เลือกโฟลเดอร์ hello ในช่อง Available Files แล้วคลิก Add ไฟล์ .class ทั้งสามที่เราคอมไพล์ไว้จะกระโดดเข้าไปอยู่ในช่อง Contents of HelloWorldEJB จากนั้นคลิก OK กลับมาที่หน้าต่างก่อนหน้า ตอนนี้ไฟล์ .class ก็เข้าไปอยู่ใน HelloWorldApp.ear แล้ว

คลิก Next



เลือก Stateless ในช่อง Bean Type เพื่อบอกให้ทราบว่าบีน HelloWorld ของเราเป็นบีนประเภทบีนบริการแบบไม่คงสถานะ

ในช่อง Enterprise Bean Class เลือก hello.HelloWorldBean เพื่อบอกว่าบีนอยู่ในคลาสชื่อ HelloWorldBean.class ในโฟลเดอร์ hello แล้วตั้งชื่อบีนว่า HelloWorldBean ในช่อง Enterprise Bean Name

ในช่อง Remote Interfaces เลือก hello.HelloWorldHome ในช่อง Remote Home Interface และเลือก hello.HelloWorld ในช่อง Remote Interface เพื่อบอกว่า โฮมอินเตอร์เฟส และอินเตอร์เฟส อยู่ในคลาส HelloWorldHome และ HelloWorld ตามลำดับ จากนั้นคลิก Next ต่อไปเลย

ไม่ต้องสนใจหน้าต่างถัดไปที่ปรากฏออกมา คลิก Finish ได้ทันที

ตอนนี้เราได้อัดบีนเข้าไปใน HelloWorldApp.ear เรียบร้อยแล้ว ไปที่ File>Save เพื่อบันทึก

ไคลน์เอนท์ของ HelloWorld

แต่ก่อนที่เราจะทดสอบบีน เราต้องสร้างโปรแกรมบนฝั่งไคลน์เอนท์ขึ้นมาก่อน เพราะบีนทำงานบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ เราไม่อาจติดต่อกับบีนได้โดยตรง ถ้าไม่มีไคลน์เอนท์ โปรแกรมบนฝั่งไคลน์เอนท์สำหรับติดต่อกับบีน HelloWorld ก็เป็นโปรแกรมภาษาจาวามาตรฐานธรรมดา ดังนี้

โปรแกรมที่ 3-4 HelloWorldClient.java

```
import hello.HelloWorld;
import hello.HelloWorldHome;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;

public class HelloWorldClient {

 public static void main(String[] args) {
   try {
    Context initial = new InitialContext();

    Context myEnv = (Context) initial.lookup("java:comp/env");
    Object objref = myEnv.lookup("ejb/SimpleHello");

    HelloWorldHome home = (HelloWorldHome)
PortableRemoteObject.narrow(objref, HelloWorldHome.class);

    HelloWorld helloWorld = home.create();

    System.out.println(helloWorld.greet());    // (1)

    System.exit(0);
```

```
    } catch (Exception ex) {}
  }
}
```

โปรแกรมนี้เริ่มด้วยการอิมพอร์ต อินเตอร์เฟส HelloWorld และ โฮมอินเตอร์เฟส HelloWorldHome ซึ่งอยู่ไคลน์เอนท์จะต้องติดต่อเวลาคุยกับบีน

โปรแกรม HelloWorldClient ไม่ใช่โปรแกรมจาวาระดับองค์กร แต่เป็นโปรแกรมภาษาจาวามาตรฐานธรรมดาที่ทำงานบนจาวาเวอร์ชัวน์แมทชีน ดังนั้นจึงมีแมธธอส main() เป็นแมธธอสหลักเหมือนกับโปรแกรมภาษาจาวามาตรฐานทั่วไป

สิ่งที่โปรแกรมนี้ทำก็คือการติดต่อไปยังบีนที่รันอยู่บนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ เพื่อขอเรียกแมธธอส greet() ของบีน ที่ทำหน้าที่แสดงคำว่า HelloWorld ให้แสดงคำว่า HelloWorld ออกหน้าจอ

ตอนนี้ยังไม่ต้องสนใจคำสั่งในบรรทัดอื่นใดนอกจากในบรรทัด (1) สังเกตว่าในโปรแกรมนี้ไม่มีการประกาศแมธธอสชื่อ greet() แต่คำสั่งในบรรทัด (1) เรียกแมธธอส greet() ได้ เพราะเป็นการเรียกแมธธอสจากระยะไกล คือ เรียกผ่านระบบเครือข่ายไปที่บีนที่รันอยู่บนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ แมธธอส greet() เป็นแมธธอสที่เราประกาศไว้ในบีน HelloWorldBean ซึ่งไม่ได้ทำอะไรนอกจากส่งคืนค่าสตริงคำว่า Hello World! การที่โปรแกรม HelloWorldClient เรียกแมธธอสนี้ภายใต้คำสั่ง System.out.println() ย่อมทำให้เกิดการแสดงผลคำว่า HelloWorld ออกที่หน้าจอ เพราะแมธธอส greet() คืนค่าสตริงคำว่า HelloWorld ส่วนคำถามที่ว่าโปรแกรม HelloWorldClient เรียกแมธธอส greet() จากระยะไกลได้อย่างไรนั้น เดี๋ยวจะอธิบายให้ฟังอีกที (เกิดจากคำสั่งที่มาก่อนหน้าคำสั่ง greet() ทั้งหมด) ตอนนี้คอมไพล์โปรแกรม HelloWorldClient.java กันก่อน

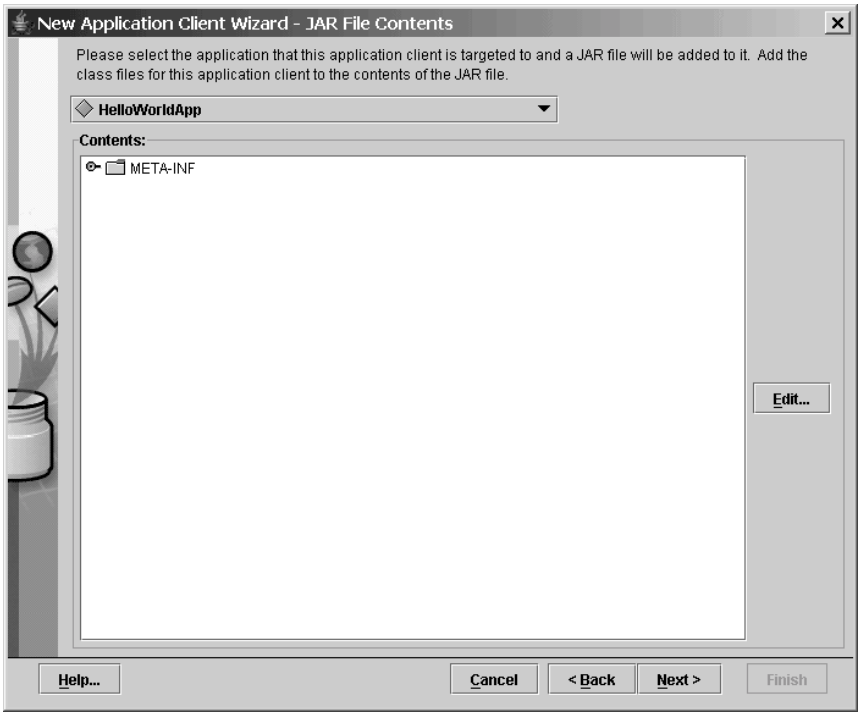
```
C:\examples\src\HelloWorld> javac –d . HelloWorldClient.java
```

สังเกตว่าไฟล์ HelloWorldClient.class ที่เกิดขึ้นจะไม่อยู่ในโฟลเดอร์ hello เพราะเราไม่ได้นิยามให้คลาส HelloWorldClient อยู่ในแพจเกจ hello อย่างคลาสของบีน

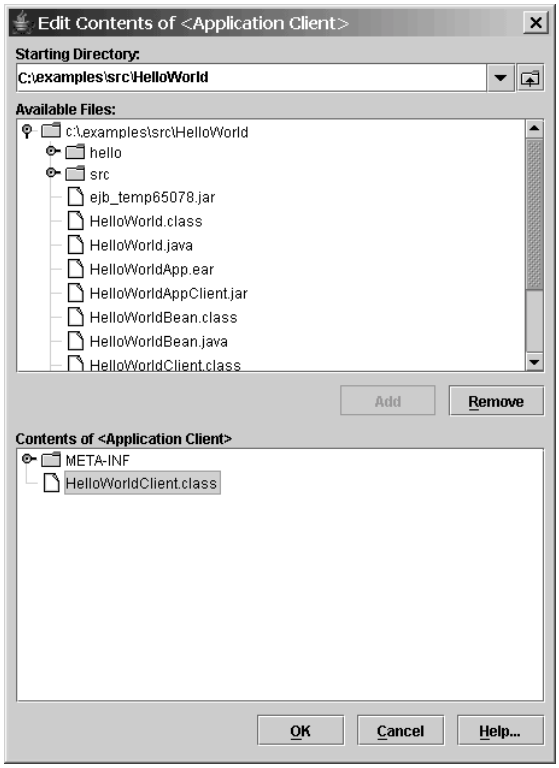
การร้องขอบริการจากเซิร์ฟเวอร์จะอยู่ในรูปของการเรียกแมธธอสของเซิร์ฟเวอร์จากระยะไกลเสมอ ดังนั้นถ้าเราต้องการให้เซิร์ฟเวอร์ทำอะไรให้ไคลน์เอนท์ก็ให้เขียนแมธธอสธุรกิจที่ทำงานอย่างนั้นขึ้นมาบนเซิร์ฟเวอร์เพื่อให้ไคลน์เอนท์เรียกใช้

ตอนนี้เราจะอัด HelloWorldClient ลงไปในไฟล์ HelloWorldApp.ear ร่วมกับบีนด้วย ให้กลับไปที่ deploytool แล้วกดปุ่ม เพื่อเพิ่มโปรแกรมเครื่องลูกข่ายลงใน HelloWorldApp (ก่อนกดปุ่มตรวจสอบดูที่หน้าต่างด้านซ้ายว่ากำลังเลือก HelloWorldApp อยู่หรือไม่ ถ้าไม่ ให้ใช้เมาส์คลิก HelloWorldApp ให้ระบายสีก่อน แล้วค่อยกดปุ่ม)

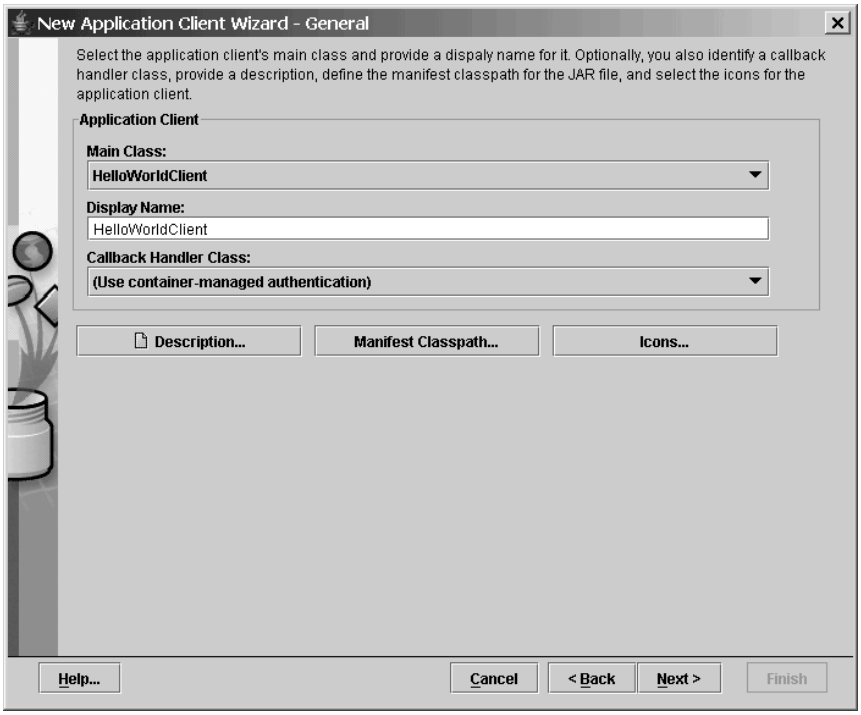
เมื่อมีหน้าต่างอธิบายความปรากฏขึ้นมาให้คลิก Next ไปยังหน้าต่างถัดไปดังภาพเลย จากนั้นคลิก Edit



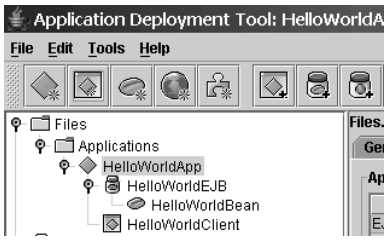
เลือกไฟล์ HelloWorldClient.class ในช่อง Available Files แล้วคลิก Add ไฟล์ HelloWorldClient.class จะเข้าไปอยู่ในช่อง Contents of <Application Client> ให้คลิก OK ออกมา



จากนั้นคลิก Next เพื่อเข้าสู่หน้าต่างต่อไป ตรวจดูว่าช่อง Main Class มีคำว่า HelloWorldClient และ ช่อง Display Name มีคำว่า HelloWorldClient หรือไม่ จากนั้นคลิก Next แล้วคลิก Finish ต่อเลย



ตอนนี้ที่หน้าต่างด้านซ้ายจะเห็น HelloWorldClient อยู่ใน HelloWorldApp เรียบร้อยแล้ว ดังภาพ



ไปที่ File>Save เพื่อบันทึกทุกอย่างเอาไว้ ตอนนี้คุณจะได้ไฟล์ HelloWorldApp.ear อยู่ในโฟลเดอร์ C:\examples\src\HelloWorld เป็นไฟล์ที่มีทั้งบีนและไคลน์เอนท์สำหรับคุยกับบีนเรียบร้อยแล้ว แต่ไฟล์นี้ยังไม่พร้อมจะโหลดลงบน J2EE server ยังมีอะไรที่เราต้องทำอีก

การเรียกแมธธอสระยะไกล

เมื่อกี้ผมค้างคุณไว้ตรงคำถามที่ว่า HelloWorldClient เรียกแมธธอส greet() ซึ่งเป็นของบีนจากระยะไกลได้อย่างไร ปกติแล้วในระดับองค์กร จะต้องมีโปรแกรมตัวหนึ่งซึ่งทำหน้าที่เก็บสารบบของโฮมอินเตอร์เฟส เพื่อให้ไคลน์เอนท์ทราบว่าบนเครือข่ายมีโฮมอินเตอร์เฟสอะไรอยู่บ้าง จะได้สั่งให้เซิร์ฟเวอร์สร้างอินสแตนท์ของบีนผ่านทางโฮมอินเตอร์เฟสนั้น และเรียกแมธธอสระยะไกลของบีนผ่านอินเตอร์เฟส

เวลาที่คุณสตาร์ท J2EE server จะมีโปรแกรมตัวหนึ่งสตาร์ทขึ้นมาด้วยเสมอเรียกว่า **JNDI** (Java Naming and Directory Interface) service ซึ่งเป็นทำหน้าที่เก็บสารบบของโฮมอินเตอร์เฟสที่มีอยู่บน J2EE server

เวลาไคลน์เอนท์จะอ้างถึงโฮมอินเตอร์เฟสผ่าน JNDI จะแทนโฮมอินเตอร์เฟสด้วยชื่อที่อยู่ในรูปแบบ java:comp/env/ejb/<ชื่อบีน> เสมอ ลองกลับไปดูโปรแกรม HelloWorldClient.java (โปรแกรมที่ 3-4) อีกครั้ง การติดต่อไปยัง JNDI เริ่มต้นด้วยการสร้างอินสแตนท์ของคลาส InitialContext ขึ้นมาก่อนด้วยคำสั่ง

```
Context initial = new InitialContext();
```

อินสแตนท์ของคลาส InitialContext มีแมธธอสที่ใช้ในการติดต่อกับ JNDI คือ แมธธอสต้องใช้มากที่สุดคือแมธธอส lookup() ซึ่งรับพารามิเตอร์สตริงเป็นชื่อของโฮมอินเตอร์เฟสที่เราต้องการเรียกหา และจะคืนค่าเป็นอินสแตนท์ของโฮมอินเตอร์เฟสตัวนั้นให้

คำสั่ง

```
Context myEnv = (Context) initial.lookup("java:comp/env");
```

เป็นการบอกเข้าไปค้นสารบบของ JNDI โดยเข้าไปในหมวด java:comp/env เพราะเรารู้อยู่แล้วว่าชื่อของโฮมอินเตอร์เฟสขึ้นต้นด้วยคำว่า java.comp/env เสมอ จากนั้นเราค่อยใช้คำสั่ง

```
Object objref = myEnv.lookup("ejb/SimpleHello");
```

เป็นการบอกให้ไปหา JNDI เข้าไปเอาอินสแตนท์ของโฮมอินเตอร์เฟสที่ชื่อว่า ejb/SimpleHello (หรือก็คือชื่อเต็มว่า java:comp/env/ejb/SimpleHello นั้นเอง) แมธธอส lookup() จะคืนค่าเป็นอินสแตนท์ของโฮมอินเตอร์เฟสที่มีชื่อว่า SimpleHello มาให้กับ objref

จากนั้นก็ใช้คำสั่ง

```
HelloWorldHome home = (HelloWorldHome)
PortableRemoteObject.narrow(objref, HelloWorldHome.class);
```

เพื่อจับเอาอินสแตนท์ของโฮมอินเตอร์เฟสที่ได้มาให้กับตัวแปร home ที่เราสร้างขึ้นมาอีกที ถึงตอนนี้ไคลน์เอนท์ก็เข้าถึงโฮมอินเตอร์เฟสของบีนได้แล้ว

จากนั้นใช้คำสั่ง

```
HelloWorld helloWorld = home.create();
```

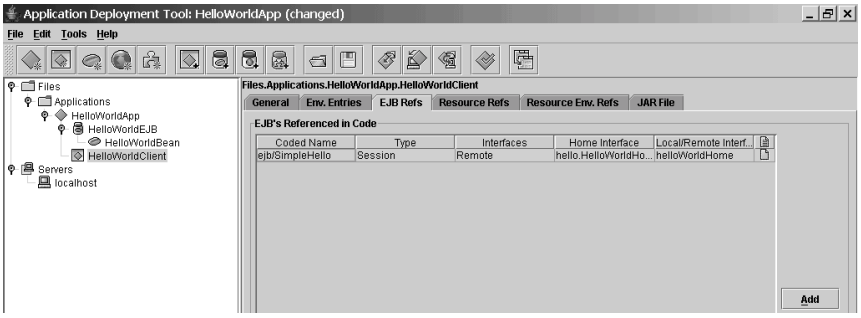
เป็นการเรียกแมธธอส create() ของโฮมอินเตอร์เฟสซึ่งเป็นผลทำให้เซิร์ฟเวอร์สร้างอินสแตนท์ของบีนขึ้นมาบนฝั่งเซิร์ฟเวอร์ ค่าที่แมธธอสนี้ส่งคืนคืออินสแตนท์ของอินเตอร์เฟส HelloWorld ซึ่งเราจับให้เท่ากับตัวแปร helloWorld ทำให้ตอนนี้ไคลน์เอนท์พร้อมจะเรียกแมธธอสระยะไกลได้แล้ว เพราะแมธธอส greet() ของบีนถูกประกาศไว้ในอินเตอร์เฟส HelloWorld (ดูโปรแกรมที่ 3-2)

แมธธอสระยะไกลแต่ละตัวที่ถูกเก็บสารบบไว้จะมีชื่อประจำตัวของมัน เรียกว่า ชื่อ JNDI (JNDI Name) ไคลน์เอนท์ที่ต้องการเรียกแมธธอสระยะไกลต้องเรียกผ่านชื่อ JNDI นี้โดยการสอบถามไปยัง JNDI

เวลาไคลน์เอนท์เรียกชื่อชื่อบีนที่ระบุไว้กับ JNDI จะอยู่ในรูปแบบ เมื่อไคลน์เอนท์สามารถเข้าถึงอินสแตนท์ของบีนได้ด้วยการสอบถามไปยัง JNDI โดยระบุชื่อที่มีรูปแบบดังกล่าว ไคลน์เอนท์ก็สามารถเรียกใช้แมธธอสของบีนผ่านอินสแตนท์ของบีนนั้นได้

กระบวนการเรียกแมธธอสระยะไกลของไคลน์เอนท์จึงเป็นดังที่กล่าวมาแล้วทั้งหมด คำสั่งเหล่านี้จึงมีอยู่เสมอที่ส่วนต้นของไคลน์เอนท์ เพราะทำให้ไคลน์เอนท์สามารถเรียกแมธธอสของบีนบนเซิร์ฟเวอร์ได้

มีข้อน่าสังเกตว่าชื่อที่ไคลน์เอนท์ใช้เรียกบีน HelloWorld คือ SimpleHello แทนที่จะเป็น HelloWorld ชื่อนี้เป็นชื่อที่คุณสามารถตั้งขึ้นมาได้เองจะเป็นอะไรก็ได้เรียกว่า Coded Name โดยวิธีการตั้งชื่อให้เรากลับไปที่ deploytool แล้วเลือก HelloWorldClient ที่หน้าต่างด้านซ้าย และเลือกแถบชื่อ EJB Refs ที่หน้าต่างด้านขวา แล้วคลิกที่ Add



ในช่อง Coded Name เติมคำว่า ejb/SimpleHello

ในช่อง Type เติมคำว่า Session

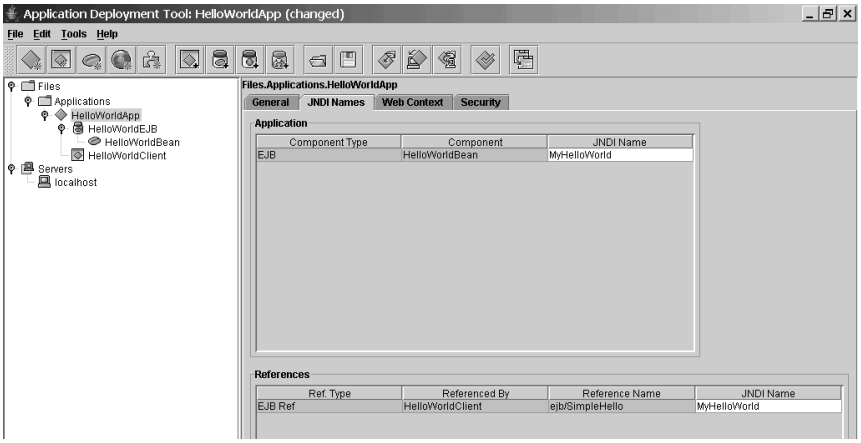
ในช่อง Interfaces เติมคำว่า Remote

ในช่อง Home Interface เติมคำว่า hello.HelloWorldHome

ในช่อง Local/Remote Interface เติมคำว่า hello.HelloWorld

การทำอย่างนี้เป็นการบอกว่าชื่อที่เห็นในโปรแกรมว่า ejb/SimpleHello นั้นให้ใช้แทนบีนบริการที่มีคลาส hello.HelloWorldHome และ hello.HelloWorld เป็นโฮมอินเตอร์เฟสและอินเตอร์เฟส ตามลำดับ

ที่หน้าต่างด้านซ้ายของ deploytool คลิกคำว่า HelloWorldApp จากนั้นที่หน้าต่างด้านขวาคลิกแถบ JNDI Names ดังภาพ



ในช่อง Application เติมคำว่า MyHelloWorld ในช่อง JNDI Name ส่วนในช่อง References เติมคำว่า MyHelloWorld อีกเช่นกันในช่อง JNDI Name

การทำเช่นนี้เป็นการผูกชื่อ SimpleHello เข้ากับชื่อ JNDI ว่า MyHelloWorld ซึ่งถูกผูกเข้ากับ HelloWorldBean อีกที ตอนนี้เซิร์ฟเวอร์จะเข้าใจแล้วว่า SimpleHello หมายถึง HelloWorldBean การที่ต้องมีชื่อ MyHelloWorld เป็นชื่อ JNDI ที่คั่นกลางอีกทีก็เพื่อว่าในอนาคตหากไคลน์เอนท์ต้องการเปลี่ยนชื่อเรียกบีนเสียใหม่จะได้ไม่ต้องแก้ที่โปรแกรม แต่มาเปลี่ยนที่ชื่อ JNDI ตรงนี้แทน อาจทำให้ดูซับซ้อนแต่ก็มีประโยชน์

NOTE

ถ้าอยากเข้าใจเรื่อง JNDI อย่างละเอียดมากขึ้น ลองดูในบทแถมที่ซีดีรอมบทที่ชื่อว่า JNDI จะเข้าใจมากขึ้น ในที่นี้เป็นการอธิบายแบบคร่าวๆ เท่านั้น ซึ่งอาจไม่ถูกต้องทั้งหมด

ถึงตอนนี้ให้เลือก File>Save เพื่อบันทึกทุกอย่างไว้ ตอนนี้ทุกอย่างพร้อมแล้วสำหรับการโหลดบีนลงบนเซิร์ฟเวอร์

โหลด HelloWorldApp

ที่ผ่านมาทั้งหมดเป็นการสร้างไฟล์ .ear ที่ภายในบรรจุไฟล์ .class ที่เป็นตัวโปรแกรม HelloWorld ของเรา ต่อไปนี้จะเป็นการโหลดโปรแกรมลงบนเซิร์ฟเวอร์ ซึ่งจะทำให้

โปรแกรมพร้อมที่จะให้บริการไคลน์เอนท์บนเครือข่าย การโหลดโปรแกรมเรียกเป็นศัพท์เฉพาะว่า การ Deploy

ที่หน้าต่างซ้ายเลือก HelloWorldApp จากนั้นกดปุ่ม

เลือกตัวเลือก Return Client Jar และ Save Object before deploying แล้วคลิก Next แล้วคลิก Finish เลย

ตัวเลือก Return Client Jar เป็นการบอกให้มีการสร้างไฟล์ชื่อ HelloWorldAppClient.jar ขึ้นมาบน C:\examples\src\HelloWorld ไฟล์นี้บรรจุสิ่งที่เรียกว่า Stub มีหน้าที่ทำให้ไคลน์เอนท์ติดต่อกับเครือข่ายได้ ส่วนตัวเลือก Save Object before deploying เป็นการบอกให้บันทึกทุกอย่างก่อนที่จะโหลด เพื่อความแน่ใจว่าคุณไม่ได้ลืมบันทึก ตอนที่คุณสร้างไฟล์ .ear

รอจนการโหลดเสร็จสมบูรณ์แล้วจึงคลิก OK ออกมา

ทดสอบโปรแกรม HelloWorld

คราวนี้มาลองทดสอบโปรแกรม HelloWorld ที่โหลดไว้แล้วกัน เรียกหน้าต่างดอสขึ้นมาแล้วเข้าไปที่โฟลเดอร์ C:\examples\src\HelloWorld

ในโฟลเดอร์นี้เป็นที่อยู่ของไฟล์ HelloWorldAppClient.jar ซึ่งเก็บ Stub ที่จำเป็นสำหรับไคลน์เอนท์ในการติดต่อกับเครือข่ายเอาไว้ ให้เซตตัวแปรแวดล้อมชื่อ APPCPATH ให้มีค่าเท่ากับ HelloWorldAppClient.jar จากนั้นเรียกโปรแกรม HelloWorldClient ด้วยคำสั่ง runclient ดังข้างล่างนี้

```
C:\> cd\examples\src\HelloWorld

C:\examples\src\HelloWorld> set APPCPATH=HelloWorldAppClient.jar

C:\examples\src\HelloWorld> runclient –client HelloWorldApp.ear   –name
HelloWorldClient –textauth
```

สักครู่โปรแกรมจะถาม username และ password ให้ใส่อะไรไปก็ได้มั่วๆ สักครู่โปรแกรมจะแสดงคำว่า Hello World! ออกมา

พารามิเตอร์ –client ใช้ระบุชื่อโปรแกรมบนจาวาแอพพลิเคชั่นเซิร์ฟเวอร์ซึ่งในที่นี้คือ HelloWorldApp.ear ส่วน –name ใช้ระบุชื่อไคลน์เอนท์ ซึ่งก็คือ HelloWorldClient และ –textauth เป็นการบอกให้มีการตรวจสอบผู้ใช้ว่าเป็นใครด้วยการถาม username และ password เป็นตัวอักษร

เวบไคลน์เอนท์สำหรับ HelloWorld

ตอนนี้คุณอาจจะงงว่าทำไมการทดสอบโปรแกรม HelloWorld ไม่เหมือนกับในบทที่แล้วที่ทดสอบด้วยเบราเซอร์ ในบทนี้เราทดสอบโดยใช้โปรแกรมภาษาจาวาเป็นตัวติดต่อไปยังเซิร์ฟเวอร์ แต่ที่จริงเราสามารถใช้เบราเซอร์เป็นตัวติดต่อเข้าไปได้ด้วยเหมือนกัน

แต่การจะทำให้สามารถใช้เบราเซอร์เป็นไคลน์เอนท์ได้ ต้องมีการเขียนไฟล์ .jsp ขึ้นมาเพื่อทำหน้าที่เป็นตัวกลาง ดังนี้

โปรแกรมที่ 3-5 index.jsp

```
<%@ page
import="hello.HelloWorld,hello.HelloWorldHome,javax.ejb.*,
javax.naming.*, javax.rmi.PortableRemoteObject,
java.rmi.RemoteException" %>

<%!
   private HelloWorld hello = null;
   public void jspInit() {
      try {
         InitialContext ic = new InitialContext();
         Object objRef =
ic.lookup("java:comp/env/ejb/TheHello");
         HelloWorldHome home =
(HelloWorldHome)PortableRemoteObject.narrow(objRef,
HelloWorldHome.class);

         hello = home.create();

      } catch (Exception ex) {
      }

   }

   public void jspDestroy() {
         hello = null;
   }
%>

<html>
<head>
    <title>HelloWorld</title>
</head>

<body bgcolor="white">
<h1><b><center><%=hello.greet()%></center></b></h1>
<hr>


</body>
</html>
```

ลองดูคำสั่งในไฟล์ index.jsp จะเห็นว่ามีอะไรคล้ายๆ กับ คำสั่งที่อยู่ใน HelloWorldClient.java แต่มีวิธีการเขียนตามแบบของคำสั่ง JSP

เราใช้แมธธอส jspInit()ซึ่งเป็นแมธธอสที่จะทำงานทุกครั้งก่อนที่จะมีการโหลดไฟล์ JSP ในการบรรจุคำสั่งที่จะเรียกหาแมธธอส greet() จากเซิร์ฟเวอร์ ซึ่งรูปแบบคำสั่งจะคล้ายกับในกรณีของ HelloWorldClient.java เป็นอย่างมาก

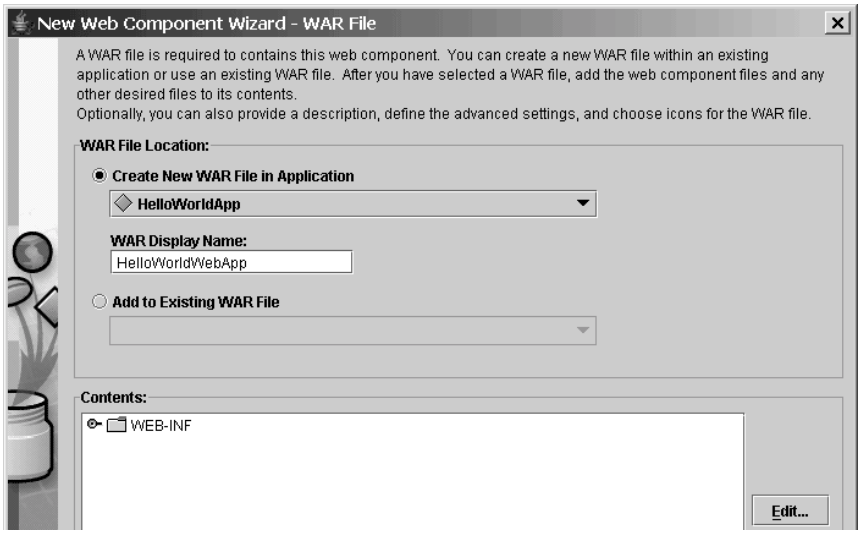
ส่วนแมธธอส jspDestroy() เป็นแมธธอสที่จะทำงานทุกครั้งเมื่อจบการทำงานของไฟล์ JSP เราจึงใส่คำสั่ง hello=null เข้าไป เพื่อเป็นการคืนทรัพยากรเท่านั้น

ที่ตัวเนื้อหาของเวบเราใช้คำสั่ง <%=hello.greet()%> ในการเรียกแมธธอส greet() ของบีน ซึ่งจะคืนค่าเป็นสตริงคำว่า Hello World! ให้ปรากฏบนเวบเพจนั้นเอง

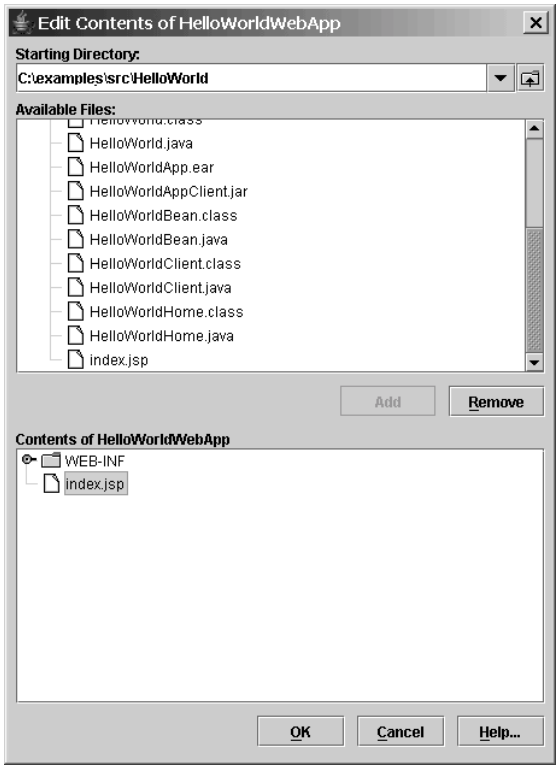
เราต้องอัดไฟล์ index.jsp นี้เข้าไปใน HelloWorldApp.ear เพื่อทำให้โปรแกรม HelloWorld ของเราใช้งานแบบเวบแอพพลิเคชั่นได้ด้วย ดังนี้

ที่หน้าต่างด้านซ้ายเลือก HelloWorldApp แล้ว กดปุ่ม

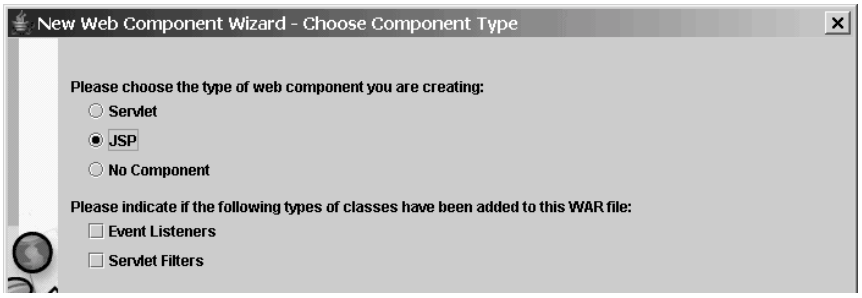
ถ้ามีหน้าต่างอธิบายให้คลิก Next ไปยังหน้าต่างต่อไป



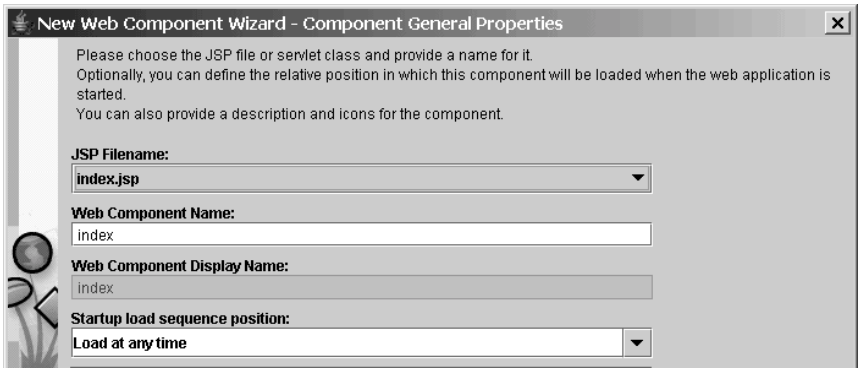
ใส่คำว่า HelloWorldWebApp ในช่อง WAR Display Name เพื่อตั้งชื่อ WAR คือ เวบคอมโพเนนท์ ที่ทำให้โปรแกรมของคุณสนับสนุนการทำงานผ่านเบราเซอร์ จากนั้นคลิก Edit



เลือกไฟล์ index.jsp ในช่อง Available Files แล้วกด Add เพื่อให้ไฟล์ index.jsp เข้าไปอยู่ในช่อง Contents of HelloWorldWebApp แล้วคลิก OK

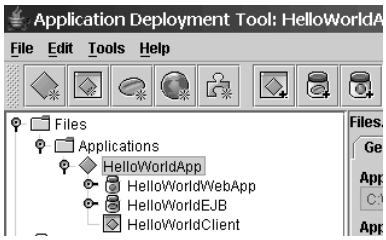


คลิก Next ไปยังหน้าต่างต่อไปเพื่อเลือกชนิดของเวบคอมโพเนนท์ ที่จริงแล้วเราจะใช้ Servlet หรือ JSP ทำหน้าที่เป็นเวบคอมโพเนนท์ก็ได้ แต่ในที่นี้เราใช้ JSP ดังนั้นเลือก JSP แล้วคลิก Next

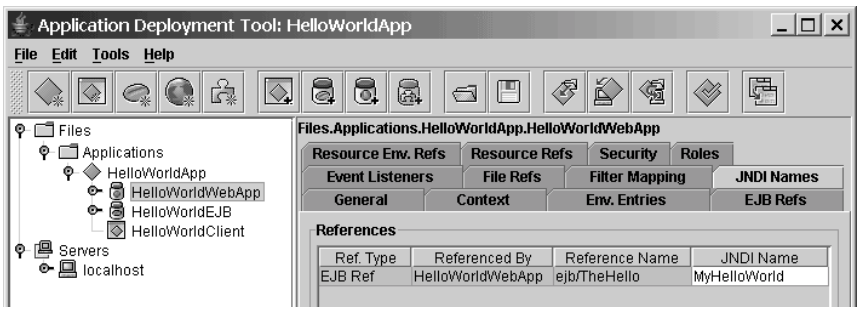


เลือก index.jsp ในช่อง JSP Filename เพื่อบอกให้ทราบว่าเวบคอมโพเนนท์ที่ว่าก็คือไฟล์ index.jsp จากนั้นคลิก Next และ Finish เลย

คราวนี้คุณต้องเห็น HelloWorldWebApp อยู่ใต้ HelloWorldApp ในหน้าต่างด้านซ้ายดังภาพ



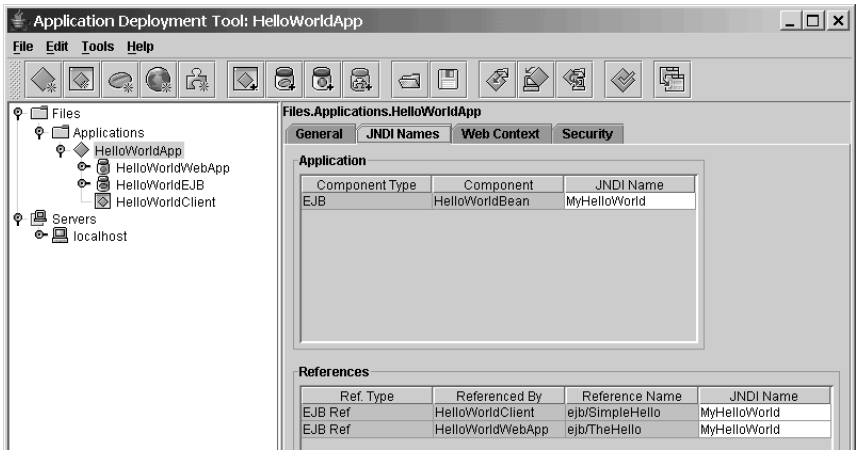
เลือก HelloWorldWebApp ที่หน้าต่างด้านซ้าย แล้วเลือกแถบ EJB Refs ที่หน้าต่างด้านขวา



ในช่อง Reference Name เติม ejb/TheHello

ในช่อง JNDI Name เติม MyHelloWorld

จากนั้นคลิก HelloWorldApp ในช่องหน้าต่างด้านซ้าย แล้วเลือกแถบ JNDI Names ที่หน้าต่างด้านขวา



ในช่อง JNDI Name แถว HelloWorldWebApp เติม MyHelloWorld

จะเห็นได้ว่า Reference Name ของไคลน์เอนท์ กับเวบคอมโพเนนท์ ไม่จำเป็นต้องเหมือนกัน ขอเพียงแต่ JDNI Name เหมือนกัน ก็จะเข้าถึง EJB ได้เหมือนกัน ข้อดีของการทำแบบนี้คือ ทำให้สามารถเปลี่ยนแปลงแก้ไขที่ฝั่งไคลน์เอนท์ได้(ถ้าต้องการ) โดยไม่ต้องไปเปลี่ยนอะไรที่ฝั่งเซิร์ฟเวอร์

จากนั้นเลือกแถบ Web Context เติมคำว่า hello ลงในช่อง Context Root

Context Root เป็นชื่อที่บอกว่าเราจะใช้เบราเซอร์เข้าถึงโปรแกรม HelloWorldWebApp ได้ด้วย URL อะไร ในกรณีนี้คือ http://localhost:8000/hello เพราะ Context Root ถูกกำหนดให้มีค่าเป็น hello เหตุที่ต้องมีการกำหนด Context Root เป็นเพราะเซิร์ฟเวอร์ตัวเดียวอาจมีโปรแกรมประยุกต์หลายโปรแกรมก็ได้ จึงต้องมีการกำหนดที่อยู่ให้เกิดความแตกต่าง

ส่วน localhost ก็คือชื่อโฮสต์ของเซิร์ฟเวอร์ และพอร์ต 8000 ก็คือพอร์ตปกติที่ J2EE server ให้บริการเวบ

เลือก File>Save

ตอนนี้ถ้าคุณโหลด (Deploy) HelloWorldApp ซ้ำใหม่ โปรแกรม HelloWorld ของคุณก็จะใช้เบราเซอร์เป็นตัวติดต่อได้ด้วย (ไม่ขอสาธิตซ้ำอีก)

โปรแกรมตัวอย่างตั้งแต่บทหน้าเป็นต้นไป เพื่อความกระชับของเนื้อหาจะใช้โปรแกรมภาษาจาวามาตรฐานเป็นไคลน์เอนท์ทดสอบอย่างเดียว ไม่มีการสร้างไฟล์ JSP เพื่อให้สามารถใช้เบราเซอร์เป็นไคลน์เอนท์ทดสอบด้วย แต่ขอให้ละไว้ในฐานที่เข้าใจว่าที่จริงแล้วจะใช้เบราเซอร์เป็นไคลน์เอนท์ด้วยก็ได้

แบบฟอร์มการสั่งซื้อหนังสือทางไปรษณีย์ (ถ่ายสำเนาได้)

หนังสือจาวาฉบับสมบูรณ์มีจำนวนบทครบถ้วนตามที่แสดงไว้ในสารบัญ สามารถสั่งซื้อได้ทางไปรษณีย์ **(ค่าส่งฟรีทั่วประเทศ)**

ชื่อ-นามสกุล ที่ให้จัดส่ง _____

ที่อยู่ _____

_____ อีเมล์แอดเดรส _____

(ท่านจะได้รับแจ้งยืนยันการสั่งซื้อและวันส่งด้วยหากระบุอีเมล์แอดเดรส)

001 จาวา สำหรับผู้เริ่มต้น

002 เจเอสพี สำหรับเวบโปรแกรมเมอร์

รายการหนังสือที่ต้องการสั่งซื้อ

รหัส	ชื่อหนังสือ	ราคาต่อเล่ม	จำนวนเล่ม
001	จาวา สำหรับผู้เริ่มต้น	235	_____เล่ม
002	เจเอสพี สำหรับเวบโปรแกรมเมอร์	180	_____เล่ม
003	J2EE & XML จาวาระดับองค์กร (แถมแผ่นซีดี)	275	____ _____เล่ม

___ ต้องการใบเสร็จรับเงิน

การชำระเงิน

☐ วิธีที่ 1 ทางธนาณัติ(ปณ.ตลิ่งชัน)/ตั๋วแลกเงินไปรษณีย์/เช็คขีดคร่อมเท่าจำนวนเงินของหนังสือที่สั่งซื้อ สั่งจ่ายนาย นรินทร์ โอฬารกิจอนันต์

☐ วิธีที่ 2 โอนหรือนำฝากเงินเข้าบัญชี ออมทรัพย์ ธ.กรุงเทพ สาขาปิ่นเกล้า เลขที่บัญชี 162-0-749901 ชื่อบัญชีนาย นรินทร์ โอฬารกิจอนันต์

ส่งแบบฟอร์มนี้พร้อมหลักฐานการชำระเงินมาที่ไปรษณีย์ที่ นาย นรินทร์ โอฬารกิจอนันต์ 21/5 ซอยลดาวัลย์ 2 ถนนบรมราชชนนี แขวงศาลาธรรมสพน์ เขตทวีวัฒนา กรุงเทพมหานคร 10170 หรือแฟกซ์/แสกน มาที่ แฟกซ์ 02-433-9122 หรือ webmaster@dekisugi.net

กรุณาอย่าส่งเงินสด หนังสือจะจัดส่งให้ภายใน 5 วันทำการโดยพัสดุลงทะเบียน หลังจากได้ทำการตรวจสอบการชำระเงินแล้ว