

Desarrollo de Aplicaciones con J2EE

Aplicaciones Distribuidas Multicapa

0/40

Contenido

- Plataforma J2EE
- Aplicaciones Distribuidas multicapa
- Arquitectura Multicapa
- Componentes J2EE
 - Componentes de Clientes: aplicaciones y applets
 - Componentes Web: JSP y servlets
 - Enterprise Java Beans: Session y Entity
- Contenedores J2EE
 - EJB
 - Web
 - Cliente Aplicación
 - Applet
- Empaquetado
- Roles, APIs, Scripts
- Ejemplos

1/40

Plataforma J2EE

- Ofrece un modelo de aplicaciones distribuidas multicapa.
- Permite la reutilización de componentes.
- Utiliza XML para el intercambio de información.
- Ofrece un modelo unificado de seguridad.
- Control de transacciones flexible.

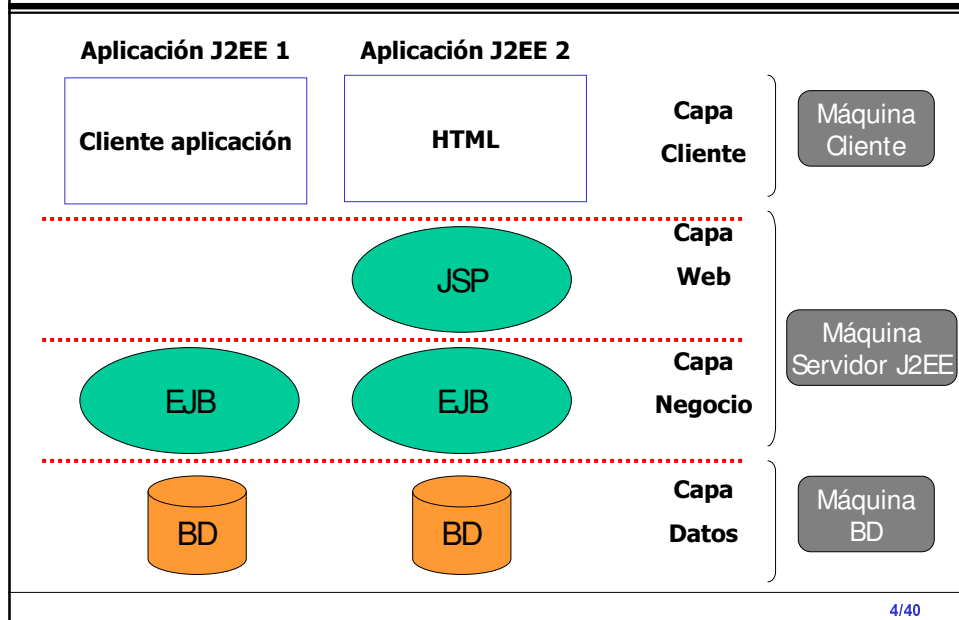
2/40

Aplicaciones Distribuidas multicapa

- Toda aplicación está dividida en componentes según la funcionalidad de la misma.
- Además estos componentes pueden estar distribuidos en diferentes máquinas.
- Capas:
 - Cliente
 - Web
 - Negocio
 - Datos

3/40

Arquitectura Multicapa



Componentes J2EE

- Las aplicaciones J2EE están formadas por componentes.
- Un componente J2EE es una unidad software autosuficiente que es ensamblada conjuntamente con las clases y ficheros necesarios.
- Los componentes se comunican entre si vía RMI.

Tipos de Componentes J2EE

- **Clientes aplicaciones y Applets** son componentes que se ejecutan en el cliente.
- **Java Servlets y JSP** son componentes Web que se ejecutan en el servidor.
- **Enterprise Java Beans** son componentes de negocio y que se ejecutan en el servidor.

6/40

Clientes J2EE

- **Clientes WEB:** consiste en dos partes.
 - Páginas Web dinámicas en algún lenguaje de marcado.
 - El navegador cliente.
 - Suele denominarse “*Cliente ligero*” (thin client) ya que no realiza accesos a BD ni procesamiento complejo.
- **Applets:** una página Web puede contener un applet, como aplicación de cliente. Se ejecutan en una JVM (plug-in), y según una política de seguridad.
- En general se prefiere la alternativa de Cliente Web.

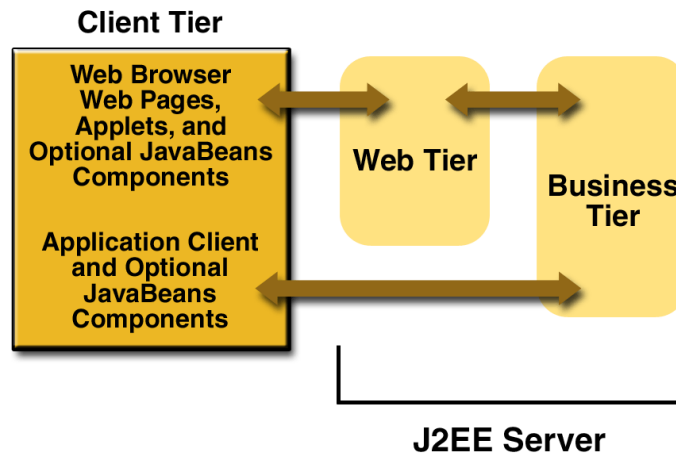
7/40

Nota acerca de los Clientes J2EE

- También podría utilizarse un cliente de aplicación que interactúa con las otras capas.
- Se pueden utilizar los Java Beans (get-set) pero no son imprescindibles ni son considerados componentes en la especificación J2EE.

8/40

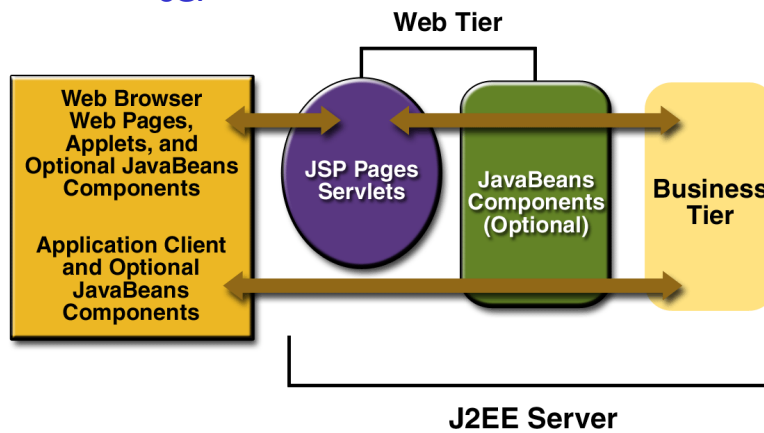
Ejemplo comunicación Capa Cliente



9/40

Componentes Web

- Servlets
- JSP



10/40

Componentes de Negocio

- Código Java que realiza las funciones de la lógica de la aplicación.
- La secuencia típica es:
 - El cliente envía petición al EJB
 - El EJB procesa los datos y accede a la capa de datos.
 - Otro EJB recupera los datos, los procesa y devuelve el resultado al cliente.

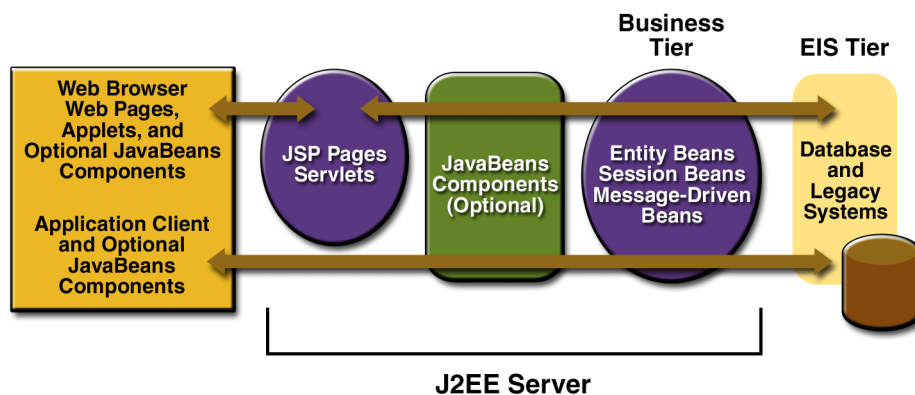
11/40

Tipos de Componentes de Negocio

- **Session Bean:**
 - Representa una conversación temporal con el cliente.
- **Entity Bean:**
 - Representa datos almacenados en las BD.
 - En este caso, los datos estarán disponibles más tarde, incluso si el servidor cae.
- **Message Bean:**
 - Es una combinación de session bean y de un mensaje del sistema de Java JMS que permite a un componente de negocio recibir mensajes asíncronos a través de un listener.

12/40

Componentes de Negocio



13/40

Contenedores J2EE

- El desarrollo de clientes finos multicapa son complicados de realizar porque implica una gran cantidad de código para realizar todas las funciones de la aplicación en cuestión:

Transacciones
Estados
Multithreading
Pool de recursos
Seguridad

14/40

Contenedores J2EE

- J2EE proporciona un conjunto de servicios básicos denominados contenedores para cada tipo de componente.
- Los contenedores constituyen una interfaz entre un componente y la plataforma J2EE.
- Antes de poder ejecutar una aplicación J2EE, todo componente debe estar ensamblado en una aplicación J2EE y desplegado en su contenedor.

15/40

Contenedores J2EE

- El proceso de ensamblado comprende la parametrización de cada componente en J2EE.
- Este proceso permite utilizar los servicios:
 - Seguridad: configurar los componentes Web o EJB para que sólo usuarios autorizados.
 - Transacciones: permite especificar relaciones entre los métodos, de forma que la ejecución de los mismos se realiza de forma atómica.
 - Java Naming and Directory: ofrece una interfaz única para el nombrado de elementos.
 - Conectividad remota: permite las comunicaciones de bajo nivel entre los clientes y los EJB.

16/40

Nota sobre Contenedores J2EE

- Según los parámetros de configuración de los componentes, su comportamiento puede ser diferente.
 - La persistencia no es un servicio configurable, pero dentro de J2EE podemos sobrescribir un Entity Java Bean para que sea gestionado por el contenedor y escribir el código que deseamos haciendo que el Bean gestione los datos.
 - Podemos crear un bean que gestione la persistencia para crear métodos de búsqueda personalizados.

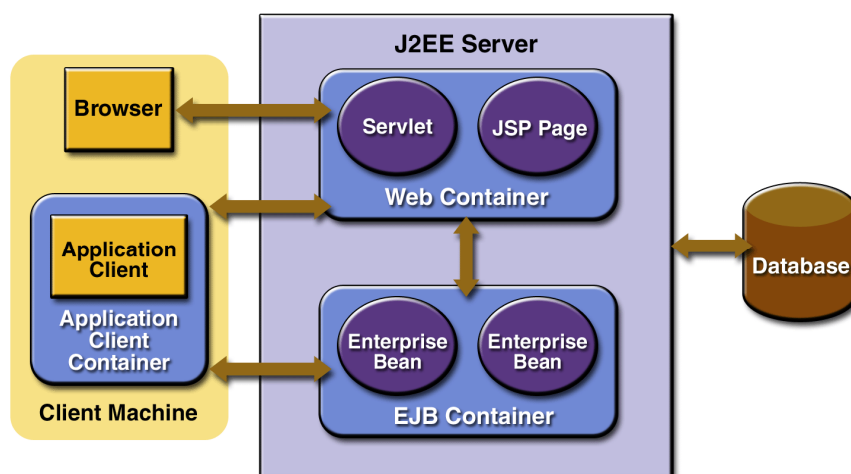
17/40

Tipos de Contenedores J2EE

- **EJB**: gestiona la ejecución de los EJB en las aplicaciones J2EE.
- **Web**: gestiona la ejecución de componentes tipo servlets y JSP en aplicaciones J2EE.
- **Cliente Aplicación**: gestiona la ejecución de clientes de aplicación.
- **Applet**: gestiona la aplicación de los applets.

18/40

Contenedores



19/40

Empaquetado en J2EE

- Los componentes J2EE son empaquetados de forma independiente para su posterior despliegue.
- Una aplicación J2EE está compuesta por uno o más EJB y componentes Web o aplicaciones.
- Una aplicación J2EE y sus módulos tienen un fichero descriptor de despliegue.
- Es un fichero XML que describe las características de despliegue de la aplicación.

20/40

Empaquetado en J2EE

- Una aplicación J2EE y todos sus módulos se ofrece en un EAR: Enterprise Archive.
 - Es un archivo “.jar” de Java pero con la extensión “.ear”.
- En la herramienta “**deploytool**” se crea el EAR y se añaden los JAR y WAR.
 - Cada fichero EAR contiene un fichero de despliegue, los ficheros EJB y recursos asociados.
 - Cada fichero JAR de cliente aplicación contiene un descriptor de despliegue, las clases y sus recursos asociados.
 - Cada fichero WAR contiene un descriptor de despliegue, los ficheros de componentes Web y los recursos necesario.

21/40

Roles en J2EE

- La capacidad de módulos reutilizables permiten dividir el desarrollo y despliegue de la aplicación entre diferentes tipos de roles.
 - **Suministrador J2EE:**
 - Compañías que implementan el servidor J2EE.
 - **S. Herramientas:**
 - Ofrecen herramientas de desarrollo, ensamblado y empaquetado.
 - **S. de Componentes de Aplicación:**
 - Desarrolla componentes Web, EJB, applets, cliente de aplicación, applets, para usarlas dentro de una aplicación J2EE.
 - **S. Desarrollador de EB:**
 - Escribe y compila códigos EB
 - Especifica el fichero de despliegue.
 - Agrupa los ficheros “.class” y el descriptor en el fichero JAR.

22/40

Roles en J2EE

- **Desarrollador de componentes Web:**
 - Escribe y compila código de servlets.
 - Escribe páginas JSP y HTML.
 - Especifica el descriptor de despliegue del componente Web.
 - Agrupa .class, .jsp, .html y el fichero de despliegue en un fichero WAR.
- **Desarrollador de cliente como aplicación:**
 - Escribe y compila código fuente java.
 - Especifica el descriptor de despliegue del cliente.
 - Agrupa .class y el fichero de despliegue en un fichero WAR.

23/40

Roles en J2EE

- **Ensamblador de aplicaciones:**
 - Recibe los componentes JAR y WAR y los integra en la aplicación EAR.
 - Puede modificar el fichero de despliegue.
 - Comprueba que todo cumple J2EE.
- **Deployer y administrador de aplicaciones:**
 - Configura y despliega la aplicación J2EE.
 - Administra la máquina e infraestructura de red.
 - Controla el entorno de ejecución.
 - Añade el fichero EAR al servidor J2EE.
 - Configura la aplicación J2EE.
 - El fichero EAR cumple J2EE.
 - Despliega la aplicación J2EE del EAR en el servidor.

24/40

SUN J2EE server

<http://java.sun.com/j2ee/download.html#sdk>

- Es un servidor no comercial utilizado como referencia de la especificación J2EE que SUN ofrece gratuito para demostraciones, prototipado, y uso educacional.
- Permite a los desarrolladores de servidores J2EE comprobar la compatibilidad de sus productos (portabilidad de aplicaciones J2EE).



25/40

J2EE APIs

- EJB 2.0
- JDBC API 2.0
- Java Servlets 2.3
- Java Server Pages 1.2
- Java Message Service 1.0
- Java Naming and Directory Interface 1.2
- Java Transaction API 1.0
- Java Mail API 1.2
- Java Beans Activation Framework 1.0
- Java API for XML Processing 1.1
- J2EE Connector Architecture 1.0
- Java Authentication and Authorization Service 1.0

26/40

J2EE Scripts

- **j2ee:**
 - arranca y para el servidor J2EE.
- **cloudscape:**
 - arranca y para la base de datos.
- **j2eeadmin:**
 - Añade JDBC drivers, destinos JMS y otros.
- **keytool:**
 - Crea clave públicas y privadas y genera certificados X509.
- **realmtool:**
 - Importa ficheros de certificados, añade y elimina usuarios de J2EE de una aplicación J2EE.
- **packager:**
 - Empaqueta componentes de aplicaciones J2EE en ficheros EAR, JAR y WAR.
- **verifier:**
 - Comprueba que los ficheros están bien formados según J2EE.
- **runclient:**
 - Ejecuta un cliente aplicación.
- **cleanup:**
 - Elimina todas las aplicaciones desplegadas.

27/40

Desarrollo de un ejemplo

- Ejemplo está en:
j2eetutorial/examples/src/ejb/convert

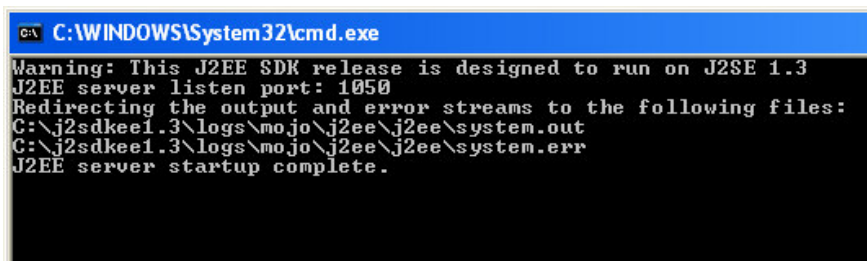
Environment Variable	Value
JAVA_HOME	The location of the J2SE SDK installation.
J2EE_HOME	The location of the J2EE SDK installation.
ANT_HOME	The location of the ant installation.
PATH	Should include the bin directories of the J2EE SDK, J2SE, and ant installations.

28/40

Desarrollo de un ejemplo

- Arrancar y parar el servidor J2EE:

```
j2ee -verbose
```



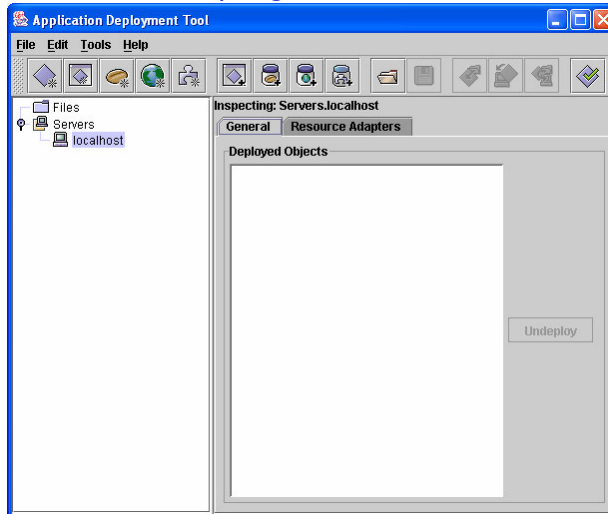
```
C:\WINDOWS\System32\cmd.exe
Warning: This J2EE SDK release is designed to run on J2SE 1.3
J2EE server listen port: 1050
Redirecting the output and error streams to the following files:
C:\j2sdkee1.3\logs\mojo\j2ee\j2ee\system.out
C:\j2sdkee1.3\logs\mojo\j2ee\j2ee\system.err
J2EE server startup complete.
```

```
j2ee -stop
```

29/40

Desarrollo de un ejemplo

- Arrancar la herramienta de despliegue:
deploytool



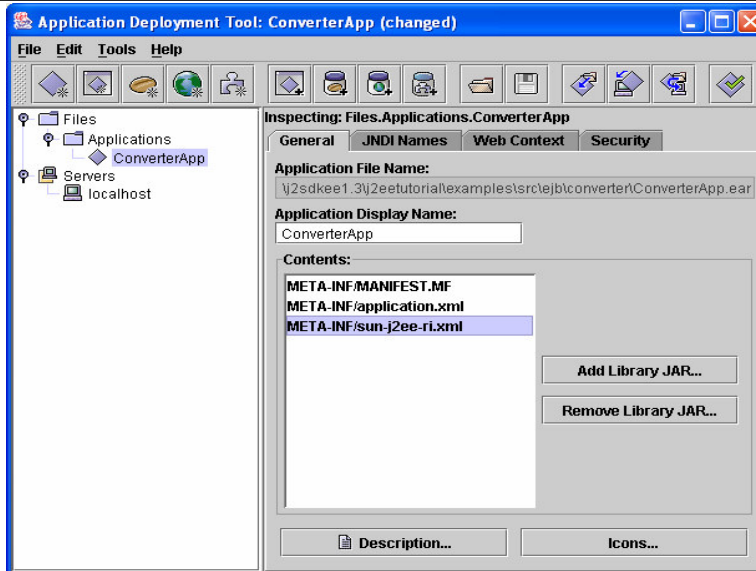
30/40

Creando la aplicación J2EE

- La aplicación “ConverterApp” contiene 3 componentes J2EE:
 - Un enterprise Java bean,
 - Un cliente de aplicación J2EE
 - Un componente Web
- En la herramienta de despliegue **DEPLOYTOOL**:
 - Seleccionar: **File** → **New Application**.
 - Presionar **Browse**.
 - Seleccionar “**j2eetutorial/examples/src/ejb/converter**”
- Poner nombre de la aplicación: “**ConverterApp.ear**”
- Pulsar “**New Application**” y **OK**

31/40

Despliegue de una nueva aplicación



32/40

ConverterApp

- **Crear el Enterprise Bean**
 - Un *enterprise bean* es un componente del servidor que contiene la lógica de negocio de la aplicación. En tiempo de ejecución, el cliente ejecuta las reglas de negocio invocando los métodos del EB.
 - Este ejemplo muestra un session bean sin conexión que se denomina **ConverterEJB**.
- **Codificando el Enterprise Bean**
 - El enterprise bean está compuesto por los siguientes códigos:
 - Remote interface
 - Home interface
 - Enterprise bean class

33/40

ConverterApp

- **Codificando la Interfaz Remota:**
 - Una interfaz remota define los métodos de negocio que un cliente puede llamar. Su implementación está en el código del EB. Se denomina Converter.
- **Codificando la Interfaz Home (o local):**
 - Una interfaz home define los métodos que permiten al cliente crear, buscar o eliminar un enterprise bean.
 - En el caso de la interfaz **ConverterHome** contiene un único método que devuelve un objeto de tipo interfaz remota.
- **Codificando la clase del Enterprise Bean:**
 - La clase del enterprise bean se llama ConverterBean.
 - Esta clase implementa 2 métodos que son reglas de negocio `dollarToYen()` y `yenToEuro()`, que están definidos en la interfaz remota.

34/40

Interfaz Remota "Converter"

```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
import java.math.*;
public interface Converter extends EJBObject {
    public BigDecimal dollarToYen(BigDecimal dollars)
        throws RemoteException;
    public BigDecimal yenToEuro(BigDecimal yen)
        throws RemoteException;
}
```

35/40

Interfaz Home "ConverterHome"

```
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
public interface ConverterHome extends EJBHome {
    Converter create() throws RemoteException,
        CreateException;
}
```

36/40

Enterprise JavaBean "ConverterBean"

```
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext; import java.math.*;
public class ConverterBean implements SessionBean {
    BigDecimal yenRate = new BigDecimal("121.6000");
    BigDecimal euroRate = new BigDecimal("0.0077");
    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
    public BigDecimal yenToEuro(BigDecimal yen) {
        BigDecimal result = yen.multiply(euroRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
    public ConverterBean() {} public void ejbCreate() {}
    public void ejbRemove() {} public void ejbActivate() {}
    public void ejbPassivate() {}
    public void setSessionContext(SessionContext sc) {} .....
}
```

37/40

ConverterApp

- **Codificando la Interfaz Remota:**
 - Una interfaz remota define los métodos de negocio que un cliente puede llamar. Su implementación está en el código del EB. Se denomina Converter.
- **Codificando la Interfaz Home (o local):**
 - Una interfaz home define los métodos que permiten al cliente crear, buscar o eliminar un enterprise bean.
 - En el caso de la interfaz **ConverterHome** contiene un único método que devuelve un objeto de tipo interfaz remota.
- **Codificando la clase del Enterprise Bean:**
 - La clase del enterprise bean se llama ConverterBean.
 - Esta clase implementa 2 métodos que son reglas de negocio dollarToYen() y yenToEuro(), que están definidos en la interfaz remota.

38/40

Desarrollo del cliente de aplicación J2EE

- Un cliente de aplicación es un programa escrito en Java
- En tiempo de ejecución se ejecuta en una JVM diferente de la del servidor J2EE.
- La aplicación cliente contiene 2 ficheros JAR:
 - El primer JAR es para el componente J2EE del cliente. Contiene:
 - Fichero descriptor de despliegue del y las clases de cliente. Con la herramienta **deploytool** → File → New Application Client wizard, se crea automáticamente el fichero JAR y lo almacena en el fichero EAR.
 - El segundo fichero JAR contiene las clases stub que permiten al cliente comunicarse con el servidor y el bean correspondiente. Este JAR no está definido por la especificación J2EE y es específico de cada plataforma.

39/40

Desarrollo del cliente de aplicación J2EE

```
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import java.math.BigDecimal;
import Converter;
import ConverterHome;
public class ConverterClient {
    public static void main(String[] args) {
        try {
            Context initial = new InitialContext();
            Object objref = initial.lookup ("java:comp/env/ejb/SimpleConverter");
            ConverterHome home = (ConverterHome)PortableRemoteObject.narrow(objref,
                ConverterHome.class);
            Converter currencyConverter = home.create();
            BigDecimal param = new BigDecimal ("100.00");
            BigDecimal amount = currencyConverter.dollarToYen(param); System.out.println(amount);
            amount = currencyConverter.yenToEuro(param);
            System.out.println(amount);
            System.exit(0);
        } catch (Exception ex) {
            System.err.println("Caught an unexpected exception!");
            ex.printStackTrace();
        }
    }
}
```

40/40